Stratix IV Device Handbook

Volume 1

101 Innovation Drive
San Jose, CA 95134
www.altera.com

SIV5V1-4.0

Copyright © 2009 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

I.S. EN ISO 9001

Contents

Chapter Revision Dates xiii

Additional Information
About this Handbook . . . . . Info-xv
How to Contact Altera . . . . . Info-xv
Typographic Conventions . . . . . Info-xv

Section I. Device Core
Revision History I-1

Chapter 1. Stratix IV Device Family Overview
Feature Summary 1-1
Stratix IV GX Devices 1-2
Stratix IV E Device 1-4
Stratix IV GT Devices 1-5
Architecture Features 1-6
High-Speed Transceiver Features 1-6
Highest Aggregate Data Bandwidth 1-6
Wide Range of Protocol Support 1-6
Diagnostic Features 1-7
Signal Integrity 1-7
FPGA Fabric and I/O Features 1-8
Device Core Features 1-8
Embedded Memory 1-8
Digital Signal Processing (DSP) Blocks 1-8
Clock Networks 1-8
PLLs 1-9
I/O Features 1-9
High-Speed Differential I/O with DPA and Soft-CDR 1-9
External Memory Interfaces 1-9
System Integration 1-10
Integrated Software Platform 1-19
Ordering Information 1-19
Document Revision History 1-20

Chapter 2. Logic Array Blocks and Adaptive Logic Modules in Stratix IV Devices
Logic Array Blocks 2-1
LAB Interconnects 2-3
LAB Control Signals 2-4

Adaptive Logic Modules 2-5
ALM Operating Modes 2-8
Normal Mode 2-8
Extended LUT Mode 2-10
Arithmetic Mode 2-11
Shared Arithmetic Mode 2-13
LUT-Register Mode 2-14
Register Chain 2-15
ALM Interconnects 2-17
Clear and Preset Logic Control 2-17
LAB Power Management Techniques 2-18
Document Revision History 2-18

Chapter 3. TriMatrix Embedded Memory Blocks in Stratix IV Devices

Overview 3-1
TriMatrix Memory Block Types 3-3
Parity Bit Support 3-3
Byte Enable Support 3-3
Packed Mode Support 3-5
Address Clock Enable Support 3-5
Mixed Width Support 3-6
Asynchronous Clear 3-6
Error Correction Code (ECC) Support 3-7
Memory Modes 3-8
Single-Port RAM 3-9
Simple Dual-Port Mode 3-10
True Dual-Port Mode 3-12
Shift-Register Mode 3-15
ROM Mode 3-16
FIFO Mode 3-16
Clocking Modes 3-17
Independent Clock Mode 3-17
Input/Output Clock Mode 3-17
Read/Write Clock Mode 3-17
Single Clock Mode 3-18
Design Considerations 3-18
Selecting TriMatrix Memory Blocks 3-18
Conflict Resolution 3-18
Read-During-Write Behavior 3-19
Same-Port Read-During-Write Mode 3-19
Mixed-Port Read-During-Write Mode 3-20
Power-Up Conditions and Memory Initialization 3-21
Power Management 3-21
Document Revision History 3-22

Chapter 4. DSP Blocks in Stratix IV Devices

Stratix IV DSP Block Overview 4-2
Stratix IV Simplified DSP Operation 4-3
Stratix IV Operational Modes Overview 4-8

Stratix IV DSP Block Resource Descriptions 4-9
Input Registers 4-10
Multiplier and First-Stage Adder 4-12
Pipeline Register Stage 4-13
Second-Stage Adder 4-13
Rounding and Saturation Stage 4-14
Second Adder and Output Registers 4-14
Stratix IV Operational Mode Descriptions 4-15
Independent Multiplier Modes 4-15
9-, 12-, and 18-Bit Multiplier 4-15
36-Bit Multiplier 4-19
Double Multiplier 4-20
Two-Multiplier Adder Sum Mode 4-22
18 x 18 Complex Multiply 4-23
Four-Multiplier Adder 4-24
High-Precision Multiplier Adder Mode 4-26
Multiply Accumulate Mode 4-27
Shift Modes 4-29
Rounding and Saturation Mode 4-31
DSP Block Control Signals 4-33
Software Support 4-34
Document Revision History 4-35

Chapter 5. Clock Networks and PLLs in Stratix IV Devices

Clock Networks in Stratix IV Devices 5-1
Global Clock Networks 5-2
Regional Clock Networks 5-3
Periphery Clock Networks 5-4
Clock Sources Per Region 5-7
Clock Regions 5-8
Clock Network Sources 5-9
Dedicated Clock Input Pins 5-9
LABs 5-9
PLL Clock Outputs 5-9
Clock Input Connections to the PLLs 5-11
Clock Output Connections 5-11
Clock Control Block 5-12
Clock Enable Signals 5-15
Clock Source Control for PLLs 5-16
Cascading PLLs 5-17

PLLs in Stratix IV Devices 5-18
Stratix IV PLL Hardware Overview 5-20
PLL Clock I/O Pins 5-22
PLL Control Signals 5-24
pfdena 5-24
areset 5-24
locked 5-24
Clock Feedback Modes 5-25
Source Synchronous Mode 5-25
Source-Synchronous Mode for LVDS Compensation 5-26
No-Compensation Mode 5-27
Normal Mode 5-28
Zero-Delay Buffer Mode 5-28
External Feedback Mode 5-29
Clock Multiplication and Division 5-30
Post-Scale Counter Cascading 5-31
Programmable Duty Cycle 5-32
Programmable Phase Shift 5-32
Programmable Bandwidth 5-34
Background 5-34
Implementation 5-35
Spread-Spectrum Tracking 5-36
Clock Switchover 5-36
Automatic Clock Switchover 5-37
Manual Clock Switchover 5-39
Guidelines 5-40
PLL Reconfiguration 5-41
PLL Reconfiguration Hardware Implementation 5-41
Post-Scale Counters (C0 to C9) 5-44
Scan Chain Description 5-45
Charge Pump and Loop Filter 5-47
Bypassing PLL 5-48
Dynamic Phase-Shifting 5-48
PLL Specifications 5-51
Chapter Revision History 5-51

Section II. I/O Interfaces

Revision History II-1

Chapter 6. I/O Features in Stratix IV Devices

I/O Standards Support 6-2
I/O Standards and Voltage Levels 6-3
I/O Banks 6-5
Modular I/O Banks 6-8

I/O Structure 6-17
3.3-V I/O Interface 6-19
External Memory Interfaces 6-19
High-Speed Differential I/O with DPA Support 6-19
Current Strength 6-20
Slew Rate Control 6-21
I/O Delay 6-22
Programmable IOE Delay 6-22
Programmable Output Buffer Delay 6-22
Open-Drain Output 6-22
Bus Hold 6-22
Pull-Up Resistor 6-23
Pre-Emphasis 6-23
Differential Output Voltage 6-23
MultiVolt I/O Interface 6-23
On-Chip Termination Support and I/O Termination Schemes 6-24
On-Chip Series (R_S) Termination Without Calibration 6-25
On-Chip Series Termination with Calibration 6-26
Left-Shift Series Termination Control 6-27
On-Chip Parallel Termination with Calibration 6-27
Expanded On-Chip Series Termination with Calibration 6-28
Dynamic On-Chip Termination 6-29
LVDS Input OCT (R_D) 6-30
OCT Calibration 6-31
OCT Calibration Block Location 6-31
Sharing an OCT Calibration Block on Multiple I/O Banks 6-33
OCT Calibration Block Modes of Operation 6-34
Power-Up Mode 6-34
User Mode 6-34
OCT Calibration 6-36
Serial Data Transfer 6-36
Example of Using Multiple OCT Calibration Blocks 6-36
RS Calibration 6-37
Termination Schemes for I/O Standards 6-37
Single-Ended I/O Standards Termination 6-37
Differential I/O Standards Termination 6-39
LVDS 6-41
Differential LVPECL 6-42
RSDS 6-42
Mini-LVDS 6-43
Design Considerations 6-44
I/O Bank Restrictions 6-44
Non-Voltage-Referenced Standards 6-44
Voltage-Referenced Standards 6-45
Mixing Voltage-Referenced and Non-Voltage-Referenced Standards 6-45
Document Revision History 6-46

Chapter 7. External Memory Interfaces in Stratix IV Devices

Memory Interfaces Pin Support 7-3
Using the R_{UP} and R_{DN} Pins in a DQS/DQ Group Used for Memory Interfaces 7-24
Combining ×16/×18 DQS/DQ Groups for a ×36 QDRII+/QDRII SRAM Interface 7-24
Rules to Combine Groups 7-25

Stratix IV External Memory Interface Features 7-26
DQS Phase-Shift Circuitry 7-27
DLL 7-28
Phase Offset Control 7-39
DQS Logic Block 7-41
DQS Delay Chain 7-42
Update Enable Circuitry 7-42
DQS Postamble Circuitry 7-43
Leveling Circuitry 7-44
Dynamic On-Chip Termination Control 7-46
I/O Element Registers 7-46
Delay Chain 7-50
I/O Configuration Block and DQS Configuration Block 7-52
Document Revision History 7-54

Chapter 8. High-Speed Differential I/O Interfaces and DPA in Stratix IV Devices

Overview 8-1
Locations of the I/O Banks 8-3
LVDS Channels 8-4
LVDS SERDES 8-8
ALTLVDS Port List 8-9
Differential Transmitter 8-11
Programmable V_{OD} and Programmable Pre-Emphasis 8-13
Programmable VOD 8-14
Programmable Pre-Emphasis 8-15
Differential Receiver 8-16
Differential I/O Termination 8-18
Receiver Hardware Blocks 8-18
DPA Block 8-18
Synchronizer 8-19
Data Realignment Block (Bit Slip) 8-20
Deserializer 8-21
Receiver Data Path Modes 8-22
Non-DPA Mode 8-22
DPA Mode 8-23
Soft-CDR Mode 8-24
LVDS Interface with the Use External PLL Option Enabled 8-25
Left and Right PLLs (PLL_Lx and PLL_Rx) 8-28
Stratix IV Clocking 8-29
Source-Synchronous Timing Budget 8-30
Differential Data Orientation 8-30
Differential I/O Bit Position 8-30
Transmitter-Channel-to-Channel Skew 8-32
Receiver Skew Margin for Non-DPA Mode 8-32

Differential Pin Placement Guidelines 8-37
Guidelines for DPA-Enabled Differential Channels 8-37
DPA-Enabled Channels and Single-Ended I/Os 8-37
DPA-Enabled Channel Driving Distance 8-37
Using Corner and Center Left and Right PLLs 8-38
Using Both Center Left and Right PLLs 8-39
Guidelines for DPA-Disabled Differential Channels 8-40
DPA-Disabled Channels and Single-Ended I/Os 8-40
DPA-Disabled Channel Driving Distance 8-40
Using Corner and Center Left and Right PLLs 8-41
Using Both Center Left and Right PLLs 8-43
Chapter Revision History 8-44

Section III. System Integration

Revision History III-1

Chapter 9. Hot Socketing and Power-On Reset in Stratix IV Devices

Stratix IV Hot-Socketing Specifications 9-1
Stratix IV Devices can be Driven Before Power Up 9-2
I/O Pins Remain Tri-Stated During Power Up 9-2
Insertion or Removal of a Stratix IV Device from a Powered-Up System 9-2
Hot-Socketing Feature Implementation in Stratix IV Devices 9-2
Power-On Reset Circuitry 9-4
Power-On Reset Specifications 9-5
Document Revision History 9-6

Chapter 10. Configuration, Design Security, and Remote System Upgrades in Stratix IV Devices

Overview 10-1
Configuration Devices 10-2
Configuration Schemes 10-2
Configuration Features 10-3
Power-On Reset Circuit 10-4
VCCPGM Pins 10-5
VCCPD Pins 10-5
Fast Passive Parallel Configuration 10-5
FPP Configuration Using a MAX II Device as an External Host 10-6
FPP Configuration Timing 10-11
FPP Configuration Using a Microprocessor 10-14
Fast Active Serial Configuration (Serial Configuration Devices) 10-14
Estimating Active Serial Configuration Time 10-20
Programming Serial Configuration Devices 10-21
Passive Serial Configuration 10-23
PS Configuration Using a MAX II Device as an External Host 10-23
PS Configuration Timing 10-28
PS Configuration Using a Microprocessor 10-29
PS Configuration Using a Download Cable 10-29
JTAG Configuration 10-32
Jam STAPL 10-37
Device Configuration Pins 10-37
Configuration Data Decompression 10-44

Remote System Upgrades 10-46
Functional Description 10-47
Enabling Remote Update 10-48
Configuration Image Types 10-49
Remote System Upgrade Mode 10-50
Remote Update Mode 10-50
Dedicated Remote System Upgrade Circuitry 10-52
Remote System Upgrade Registers 10-53
Remote System Upgrade Control Register 10-54
Remote System Upgrade Status Register 10-55
Remote System Upgrade State Machine 10-56
User Watchdog Timer 10-57
Quartus II Software Support 10-57
ALTREMOTE_UPDATE Megafunction 10-58
Design Security 10-58
Stratix IV Security Protection 10-59
Security Against Copying 10-59
Security Against Reverse Engineering 10-59
Security Against Tampering 10-60
AES Decryption Block 10-60
Flexible Security Key Storage 10-60
Stratix IV Design Security Solution 10-61
Security Modes Available 10-62
Volatile Key 10-62
Non-Volatile Key 10-62
Non-Volatile Key with Tamper Protection Bit Set 10-62
No Key Operation 10-62
Supported Configuration Schemes 10-63
Document Revision History 10-64

Chapter 11. SEU Mitigation in Stratix IV Devices

Error Detection Fundamentals 11-2
Configuration Error Detection 11-2
User Mode Error Detection 11-2
Automated Single-Event Upset Detection 11-5
Error Detection Pin Description 11-5
CRC_ERROR Pin 11-5
Error Detection Block 11-6
Error Detection Registers 11-6
Error Detection Timing 11-8
Software Support 11-10
Recovering From CRC Errors 11-11
Document Revision History 11-12

Chapter 12. JTAG Boundary-Scan Testing in Stratix IV Devices

BST Architecture 12-1
BST Operation Control 12-1
I/O Voltage Support in a JTAG Chain 12-3
BST Circuitry 12-3
BSDL Support 12-4
Document Revision History 12-4

Chapter 13. Power Management in Stratix IV Devices

Overview 13-1
Stratix IV Power Technology 13-2
Programmable Power Technology 13-2
Stratix IV External Power Supply Requirements 13-3
Temperature Sensing Diode 13-3
External Pin Connections 13-4
Document Revision History 13-5

Chapter Revision Dates

The chapters in this book, *Stratix IV Device Handbook Volume 1*, were revised on the following dates. Where chapters or groups of chapters are available separately, part numbers are listed.

Chapter 1 Stratix IV Device Family Overview
Revised: *November 2009*
Part Number: *SIV51001-3.0*

Chapter 2 Logic Array Blocks and Adaptive Logic Modules in Stratix IV Devices
Revised: *November 2009*
Part Number: *SIV51002-3.0*

Chapter 3 TriMatrix Embedded Memory Blocks in Stratix IV Devices
Revised: *November 2009*
Part Number: *SIV51003-3.0*

Chapter 4 DSP Blocks in Stratix IV Devices
Revised: *November 2009*
Part Number: *SIV51004-3.0*

Chapter 5 Clock Networks and PLLs in Stratix IV Devices
Revised: *November 2009*
Part Number: *SIV51005-3.0*

Chapter 6 I/O Features in Stratix IV Devices
Revised: *November 2009*
Part Number: *SIV51006-3.0*

Chapter 7 External Memory Interfaces in Stratix IV Devices
Revised: *November 2009*
Part Number: *SIV51007-3.0*

Chapter 8 High-Speed Differential I/O Interfaces and DPA in Stratix IV Devices
Revised: *November 2009*
Part Number: *SIV51008-3.0*

Chapter 9 Hot Socketing and Power-On Reset in Stratix IV Devices
Revised: *November 2009*
Part Number: *SIV51009-3.0*

Chapter 10 Configuration, Design Security, and Remote System Upgrades in Stratix IV Devices
Revised: *November 2009*
Part Number: *SIV51010-3.0*

Chapter 11 SEU Mitigation in Stratix IV Devices
Revised: *November 2009*
Part Number: *SIV51011-3.0*

Chapter 12 JTAG Boundary-Scan Testing in Stratix IV Devices
Revised: *November 2009*
Part Number: *SIV51012-3.0*

Chapter 13 Power Management in Stratix IV Devices
Revised: *November 2009*
Part Number: *SIV51013-3.0*

About this Handbook

This handbook provides comprehensive information about the Altera® Stratix® IV family of devices.

How to Contact Altera

For the most up-to-date information about Altera products, see the following table.

Contact *(Note 1)*	Contact Method	Address
Technical support	Website	www.altera.com/support
Technical training	Website	www.altera.com/training
	Email	custrain@altera.com
Product literature	Website	www.altera.com/literature
Non-technical support (General)	Email	nacomp@altera.com
(Software Licensing)	Email	authorization@altera.com

Note:

(1) You can also contact your local Altera sales office or sales representative.

Typographic Conventions

The following table shows the typographic conventions that this document uses.

Visual Cue	Meaning
Bold Type with Initial Capital Letters	Indicates command names, dialog box titles, dialog box options, and other GUI labels. For example, **Save As** dialog box. For GUI elements, capitalization matches the GUI.
bold type	Indicates directory names, project names, disk drive names, file names, file name extensions, dialog box options, software utility names, and other GUI labels. For example, **\qdesigns** directory, **d:** drive, and **chiptrip.gdf**.
Italic Type with Initial Capital Letters	Indicates document titles. For example, *AN 519: Stratix IV Design Guidelines.*
Italic type	Indicates variables. For example, $n + 1$. Variable names are enclosed in angle brackets (< >). For example, *<file name>* and *<project name>***.pof**.
Initial Capital Letters	Indicates keyboard keys and menu names. For example, Delete key and the Options menu.
"Subheading Title"	Quotation marks indicate references to sections within a document and titles of Quartus II Help topics. For example, "Typographic Conventions."

Visual Cue	Meaning
`Courier type`	Indicates signal, port, register, bit, block, and primitive names. For example, `data1`, `tdi`, and `input`. Active-low signals are denoted by suffix n. For example, `resetn`. Indicates command line commands and anything that must be typed exactly as it appears. For example, `c:\qdesigns\tutorial\chiptrip.gdf`. Also indicates sections of an actual file, such as a Report File, references to parts of files (for example, the AHDL keyword `SUBDESIGN`), and logic function names (for example, `TRI`).
1., 2., 3., and a., b., c., and so on.	Numbered steps indicate a list of items when the sequence of the items is important, such as the steps listed in a procedure.
■ ■	Bullets indicate a list of items when the sequence of the items is not important.
☞	The hand points to information that requires special attention.
CAUTION	A caution calls attention to a condition or possible situation that can damage or destroy the product or your work.
WARNING	A warning calls attention to a condition or possible situation that can cause you injury.
↵	The angled arrow instructs you to press **Enter**.
	The feet direct you to more information about a particular topic.

Section I. Device Core

This section provides a complete overview of all features relating to the Stratix® IV device family, which is the most architecturally advanced, high-performance, low-power FPGA in the market place. This section includes the following chapters:

- Chapter 1, Stratix IV Device Family Overview
- Chapter 2, Logic Array Blocks and Adaptive Logic Modules in Stratix IV Devices
- Chapter 3, TriMatrix Embedded Memory Blocks in Stratix IV Devices
- Chapter 4, DSP Blocks in Stratix IV Devices
- Chapter 5, Clock Networks and PLLs in Stratix IV Devices

Revision History

Refer to each chapter for its own specific revision history. For information on when each chapter was updated, refer to the Chapter Revision Dates section, which appears in the full handbook.

1. Stratix IV Device Family Overview

SIV51001-3.0

Altera® Stratix® IV FPGAs deliver a breakthrough level of system bandwidth and power efficiency for high-end applications, allowing you to innovate without compromise. Stratix IV FPGAs are based on the Taiwan Semiconductor Manufacturing Company (TSMC) 40-nm process technology and surpass all other high-end FPGAs, with the highest logic density, most transceivers, and lowest power requirements.

The Stratix IV device family contains three optimized variants to meet different application needs:

- Stratix IV E (Enhanced) FPGAs—up to 813,050 logic elements (LEs), 33,294 Kbits RAM, and 1,288 18 × 18 bit multipliers
- Stratix IV GX transceiver FPGAs—up to 531,200 LEs, 27,376 Kbits RAM, 1,288 18 × 18-bit multipliers, and 48 full-duplex CDR-based transceivers at up to 8.5 Gbps
- Stratix IV GT FPGAs—up to 531,200 LEs, 27,376 Kbits RAM, 1,288 18 × 18-bit multipliers, and 48 full-duplex clock data recovery (CDR)-based transceivers at up to 11.3 Gbps

The complete Altera high-end solution includes the lowest risk, lowest total cost path to volume using HardCopy® IV ASICs for all the family variants, a comprehensive portfolio of application solutions customized for end-markets, and the industry leading Quartus® II software for increasing productivity and performance.

This chapter contains the following sections:

- "Feature Summary" on page 1–1
- "Architecture Features" on page 1–6
- "Integrated Software Platform" on page 1–19
- "Ordering Information" on page 1–19

Feature Summary

The following list summarizes the Stratix IV device family features:

- Up to 48 full-duplex CDR-based transceivers in Stratix IV GX and GT devices supporting data rates up to 8.5 Gbps and 11.3 Gbps, respectively
- Dedicated circuitry to support physical layer functionality for popular serial protocols, such as PCI-Express (PIPE) Gen1 and Gen2, Gigabit Ethernet, Serial RapidIO, SONET/SDH, XAUI/HiGig, (OIF) CEI-6G, SD/HD/3G-SDI, Fibre Channel, SFI-5, and Interlaken
- Complete PCI Express (PIPE) protocol solution with embedded PCI Express hard IP blocks that implement PHY-MAC layer, Data Link layer, and Transaction layer functionality

For more information, refer to the *PCI Express Compiler User Guide*.

- Programmable transmitter pre-emphasis and receiver equalization circuitry to compensate for frequency-dependent losses in the physical medium
- Typical physical medium attachment (PMA) power consumption of 100 mW at 3.125 Gbps and 135 mW at 6.375 Gbps per channel
- 72,600 to 813,050 equivalent LEs per device
- 7,370 to 33,294 Kbits of enhanced TriMatrix memory consisting of three RAM block sizes to implement true dual-port memory and FIFO buffers
- High-speed DSP blocks configurable as 9 × 9-bit, 12 × 12-bit, 18 × 18-bit, and 36 × 36-bit full-precision multipliers at up to 600 MHz
- Up to 16 global clocks (GCLK), 88 regional clocks (RCLK), and 132 periphery clocks (PCLK) per device
- Programmable power technology that minimizes power while maximizing device performance
- Up to 1,120 user I/O pins arranged in 24 modular I/O banks that support a wide range of single-ended and differential I/O standards
- Support for high-speed external memory interfaces including DDR, DDR2, DDR3 SDRAM, RLDRAM II, QDR II, and QDR II+ SRAM on up to 24 modular I/O banks
- High-speed LVDS I/O support with serializer/deserializer (SERDES), dynamic phase alignment (DPA), and soft-CDR circuitry at data rates up to 1.6 Gbps
- Support for source-synchronous bus standards, including SGMII, Gigabit Ethernet, SPI-4 Phase 2 (POS-PHY Level 4), SFI-4.1, XSBI, UTOPIA IV, NPSI, and CSIX-L1
- Pinouts for Stratix IV E devices designed to allow migration of designs from Stratix III to Stratix IV E with minimal PCB impact

Stratix IV GX Devices

Stratix IV GX devices provide up to 48 CDR-based transceiver channels per device:

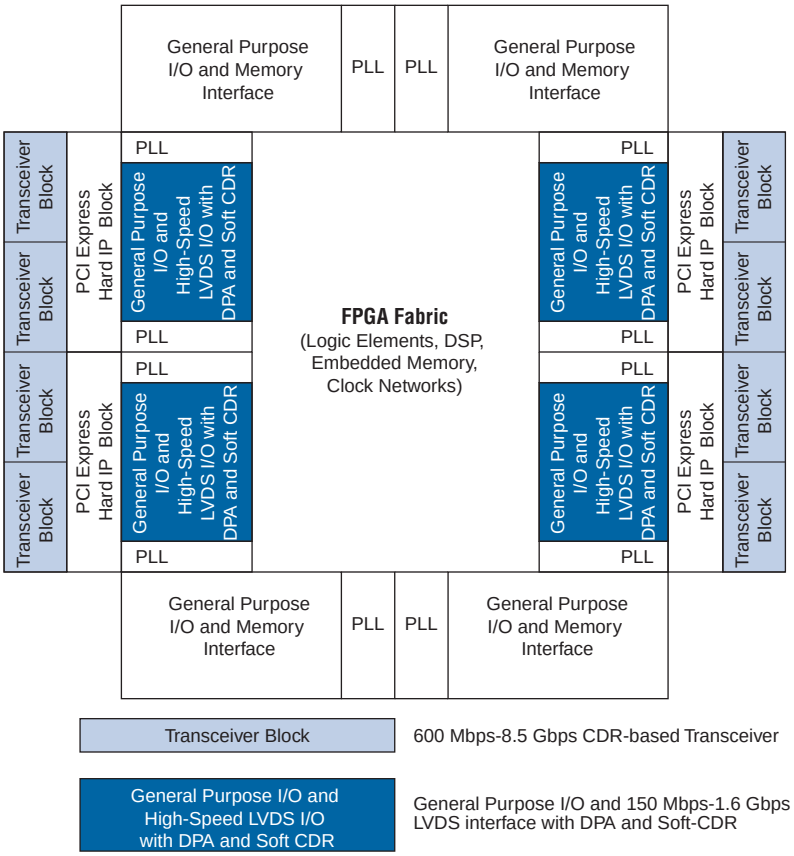
- Thirty-two out of the 48 transceiver channels have dedicated physical coding sublayer (PCS) and physical medium attachment (PMA) circuitry and support data rates between 600 Mbps and 8.5 Gbps
- The remaining sixteen transceiver channels have dedicated PMA-only circuitry and support data rates between 600 Mbps and 6.5 Gbps

☞ The actual number of transceiver channels per device varies with device selection. For more information about the exact transceiver count in each device, refer to Table 1–1 on page 1–11.

☞ For more information about transceiver architecture, refer to the *Stratix IV Transceiver Architecture* chapter.

Figure 1–1 shows a high-level Stratix IV GX chip view.

Figure 1–1. Stratix IV GX Chip View *(Note 1)*



Note to Figure 1–1:

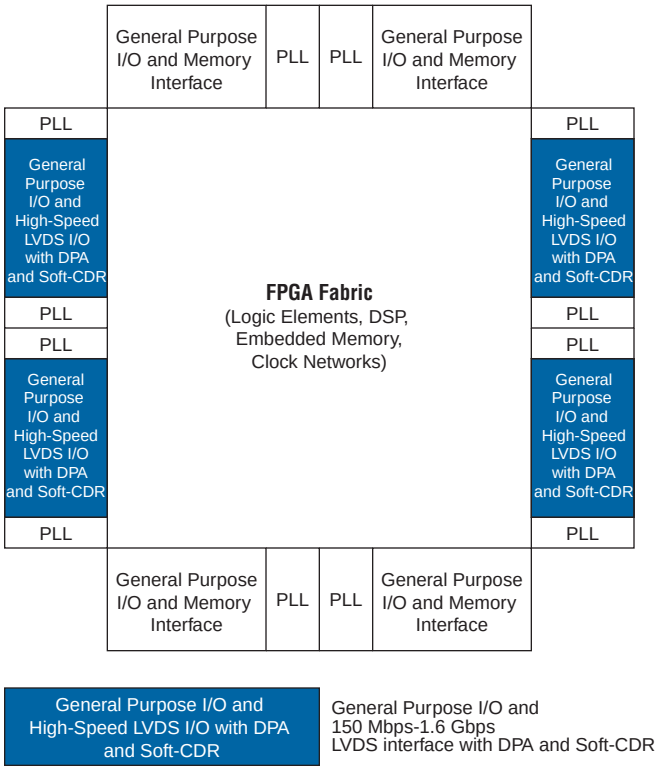
(1) Resource counts vary with device selection, package selection, or both.

Stratix IV E Device

Stratix IV E devices provide an excellent solution for applications that do not require high-speed CDR-based transceivers, but are logic, user I/O, or memory intensive.

Figure 1–2 shows a high-level Stratix IV E chip view.

Figure 1–2. Stratix IV E Chip View *(Note 1)*



Note to Figure 1–2:

(1) Resource counts vary with device selection, package selection, or both.

Stratix IV GT Devices

Stratix IV GT devices provide up to 48 CDR-based transceiver channels per device:

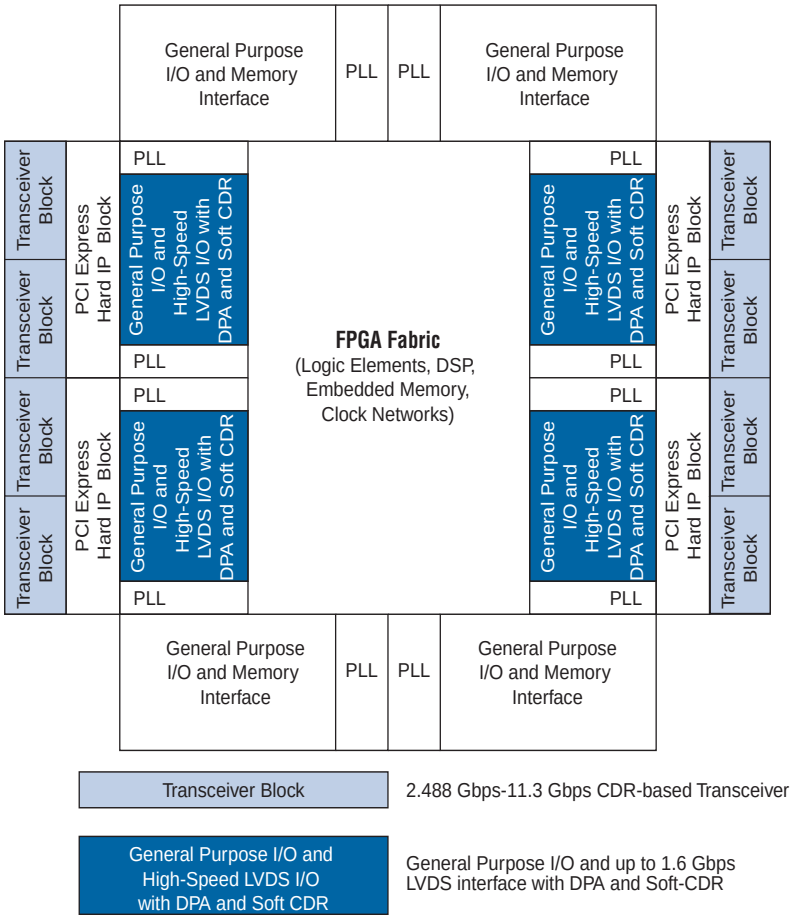
- Thirty-two out of the 48 transceiver channels have dedicated PCS and PMA circuitry and support data rates between 2.488 Gbps and 11.3 Gbps
- The remaining sixteen transceiver channels have dedicated PMA-only circuitry and support data rates between 2.488 Gbps and 6.5 Gbps

The actual number of transceiver channels per device varies with device selection. For more information about the exact transceiver count in each device, refer to Table 1–10 on page 1–20.

For more information about transceiver architecture, refer to the *Stratix IV Transceiver Architecture* chapter.

Figure 1–3 shows a high-level Stratix IV GT chip view.

Figure 1–3. Stratix IV GT Chip View *(Note 1)*



Note to Figure 1–3:

(1) Resource counts vary with device selection, package selection, or both.

For information about Stratix IV GT devices, refer to the *Stratix IV Transceiver Architecture* chapter.

Architecture Features

The Stratix IV device family features are divided into high-speed transceiver features and FPGA fabric and I/O features.

The high-speed transceiver features apply only to Stratix IV GX and Stratix IV GT devices.

High-Speed Transceiver Features

Stratix IV GX and Stratix IV GT high-speed transceiver features include:

Highest Aggregate Data Bandwidth

- Up to 48 full-duplex transceiver channels supporting data rates up to 8.5 Gbps in Stratix IV GX devices and up to 11.3 Gbps in Stratix IV GT devices.

Wide Range of Protocol Support

Physical layer support for the following serial protocols:

- Stratix IV GX—PCI Express (PIPE) Gen1 and Gen2, Gigabit Ethernet, Serial RapidIO, SONET/SDH, XAUI/HiGig, (OIF) CEI-6G, SD/HD/3G-SDI, Fibre Channel, SFI-5, GPON, SAS/SATA, HyperTransport 1.0 and 3.0, and Interlaken
- Stratix IV GT—40G/100G Ethernet, SFI-S, Interlaken, SFI-5.1, Serial RapidIO, SONET/SDH, XAUI/HiGig, (OIF) CEI-6G, 3G-SDI, and Fibre Channel
- Extremely flexible and easy-to-configure transceiver data path to implement proprietary protocols
- PCI Express (PIPE) Support
  - Complete PCI Express (PIPE) Gen1 and Gen2 protocol stack solution compliant to PCI Express base specification 2.0 that includes PHY-MAC, Data Link, and transaction layer circuitry embedded in PCI Express hard IP blocks

    For more information, refer to the *PCI Express Compiler User Guide*.

  - Root complex and end-point applications
  - ×1, ×4, and ×8 lane configurations
  - PIPE 2.0-compliant interface
  - Embedded circuitry to switch between Gen1 and Gen2 data rates
  - Built-in circuitry for electrical idle generation and detection, receiver detect, power state transitions, lane reversal, and polarity inversion
  - 8B/10B encoder and decoder, receiver synchronization state machine, and ± 300 parts per million (ppm) clock compensation circuitry
  - Transaction layer support for up to two virtual channels (VCs)

- XAUI/HiGig Support
  - Compliant to IEEE802.3ae specification
  - Embedded state machine circuitry to convert XGMII idle code groups (||I||) to and from idle ordered sets (||A||, ||K||, ||R||) at the transmitter and receiver, respectively
  - 8B/10B encoder and decoder, receiver synchronization state machine, lane deskew, and ± 100 ppm clock compensation circuitry
- Gigabit Ethernet Support
  - Compliant to IEEE802.3-2005 specification
  - Automatic idle ordered set (/I1/, /I2/) generation at the transmitter, depending on the current running disparity
  - 8B/10B encoder and decoder, receiver synchronization state machine, and ± 100 ppm clock compensation circuitry
- Support for other protocol features such as MSB-to-LSB transmission in SONET/SDH configuration and spread-spectrum clocking in PCI Express (PIPE) configurations

Diagnostic Features

- Serial loopback from the transmitter serializer to the receiver CDR for transceiver PCS and PMA diagnostics
- Reverse serial loopback pre- and post-CDR to transmitter buffer for physical link diagnostics
- Loopback master and slave capability in PCI Express hard IP blocks

For more information, refer to the *PCI Express Compiler User Guide*.

Signal Integrity

Stratix IV devices simplify the challenge of signal integrity through a number of chip, package, and board-level enhancements to enable efficient high-speed data transfer into and out of the device. These enhancements include:

- Programmable 3-tap transmitter pre-emphasis with up to 8192 pre-emphasis levels to compensate for pre-cursor and post-cursor inter-symbol interference (ISI)
- Up to 900% boost capability on the first pre-emphasis post-tap
- User-controlled and adaptive 4-stage receiver equalization with up to 16 dB of high-frequency gain
- On-die power supply regulators for transmitter and receiver phase-locked loop (PLL) charge pump and voltage controlled oscillator (VCO) for superior noise immunity
- On-package and on-chip power supply decoupling to satisfy transient current requirements at higher frequencies, thereby reducing the need for on-board decoupling capacitors
- Calibration circuitry for transmitter and receiver on-chip termination (OCT) resistors

FPGA Fabric and I/O Features

The Stratix IV FPGA fabric and I/O features include:

Device Core Features

- Up to 531,200 LEs in Stratix IV GX and Stratix IV GT devices and up to 813,050 LEs in Stratix IV E devices, efficiently packed in unique and innovative adaptive logic modules (ALMs)
- Ten ALMs per logic array block (LAB) deliver faster performance, improved logic utilization, and optimized routing
- Programmable power technology, including a variety of process, circuit, and architecture optimizations and innovations
- Programmable power technology available to select power-driven compilation options for reduced static power consumption

Embedded Memory

- TriMatrix embedded memory architecture provides three different memory block sizes to efficiently address the needs of diversified FPGA designs:
  - 640-bit MLAB
  - 9-Kbit M9K
  - 144-Kbit M144K
- Up to 33,294 Kbit of embedded memory operating at up to 600 MHz
- Each memory block is independently configurable to be a single- or dual-port RAM, FIFO, ROM, or shift register

Digital Signal Processing (DSP) Blocks

- Flexible DSP blocks configurable as 9 × 9-bit, 12 × 12-bit, 18 × 18-bit, and 36 × 36-bit full-precision multipliers at up to 600 MHz with rounding and saturation capabilities
- Faster operation due to fully pipelined architecture and built-in addition, subtraction, and accumulation units to combine multiplication results
- Optimally designed to support advanced features such as adaptive filtering, barrel shifters, and finite and infinite impulse response (FIR and IIR) filters

Clock Networks

- Up to 16 GCLKs and 88 RCLKs optimally routed to meet the maximum performance of 800 MHz
- Up to 112 and 132 PCLKs in Stratix IV GX and Stratix IV E devices, respectively
- Up to 66 (16 GCLK + 22 RCLK + 28 PCLK) per device quadrant in Stratix IV GX and Stratix IV GT devices
- Up to 71 (16 GCLK + 22 RCLK + 33 PCLK) per device quadrant in Stratix IV E devices

PLLs

- Three to 12 PLLs per device supporting spread-spectrum input tracking, programmable bandwidth, clock switchover, dynamic reconfiguration, and delay compensation
- On-chip PLL power supply regulators to minimize noise coupling

I/O Features

- Sixteen to 24 modular I/O banks per device with 24 to 48 I/Os per bank designed and packaged for optimal simultaneous switching noise (SSN) performance and migration capability
- Support for a wide range of industry I/O standards, including single-ended (LVTTL/CMOS/PCI/PCIX), differential (LVDS/mini-LVDS/RSDS), voltage-referenced single-ended and differential (SSTL/HSTL Class I/II) I/O standards
- On-chip series (R_S) and on-chip parallel (R_T) termination with auto-calibration for single-ended I/Os and on-chip differential (R_D) termination for differential I/Os
- Programmable output drive strength, slew rate control, bus hold, and weak pull-up capability for single-ended I/Os
- User I/O:GND:V_{CC} ratio of 8:1:1 to reduce loop inductance in the package—PCB interface
- Programmable transmitter differential output voltage (V_{OD}) and pre-emphasis for high-speed LVDS I/O

High-Speed Differential I/O with DPA and Soft-CDR

- Dedicated circuitry on the left and right sides of the device to support differential links at data rates from 150 Mbps to 1.6 Gbps
- Up to 98 differential SERDES in Stratix IV GX devices, up to 132 differential SERDES in Stratix IV E devices, and up to 47 differential SERDES in Stratix IV GT devices
- DPA circuitry at the receiver automatically compensates for channel-to-channel and channel-to-clock skew in source synchronous interfaces
- Soft-CDR circuitry at the receiver allows implementation of asynchronous serial interfaces with embedded clocks at up to 1.6 Gbps data rate (SGMII and Gigabit Ethernet)

External Memory Interfaces

- Support for existing and emerging memory interface standards such as DDR SDRAM, DDR2 SDRAM, DDR3 SDRAM, QDRII SRAM, QDRII+ SRAM, and RLDRAM II
- DDR3 up to 1,067 Mbps/533 MHz
- Programmable DQ group widths of 4 to 36 bits (includes parity bits)
- Dynamic OCT, trace mismatch compensation, read-write leveling, and half-rate register capabilities provide a robust external memory interface solution

System Integration

- All Stratix IV devices support hot socketing
- Four configuration modes:
  - Passive Serial (PS)
  - Fast Passive Parallel (FPP)
  - Fast Active Serial (FAS)
  - JTAG configuration
- Ability to perform remote system upgrades
- 256-bit advanced encryption standard (AES) encryption of configuration bits protects your design against copying, reverse engineering, and tampering
- Built-in soft error detection for configuration RAM cells

For more information about how to connect the PLL, external memory interfaces, I/O, high speed differential I/O, power, and the JTAG pins to PCB, refer to the *Stratix IV GX and Stratix IV E Device Family Pin Connection Guidelines* and the *Stratix IV GT Device Family Pin Connection Guidelines*.

Table 1–1 lists the Stratix IV GX device features.

Table 1–1. Stratix IV GX Device Features (Part 1 of 2)

Feature	EP4SGX70		EP4SGX110			EP4SGX180				EP4SGX230				EP4SGX290						EP4SGX360						EP4SGX530			
Package Option	**F780**	**F1152**	**F780**	**F1152**		**F780**	**F1152**		**F1517**	**F780**	**F1152**		**F1517**	**F780**	**F1152**		**F1517**	**F1760**	**F1932**	**F780**	**F1152**		**F1517**	**F1760**	**F1932**	**F1152**	**F1517**	**F1760**	**F1932**
ALMs	29,040		42,240			70,300				91,200				116,480						141,440						212,480			
LEs	72,600		105,600			175,750				228,000				291,200						353,600						531,200			
0.6 Gbps-8.5 Gbps Transceivers (PMA + PCS) *(1)*	—	16	—	—	16	—	—	16	24	—	—	16	24	—	—	16	24	24	32	—	—	16	24	24	32	16	24	24	32
0.6 Gbps-6.5 Gbps Transceivers (PMA +PCS) *(1)*	8	—	8	16	—	8	16	—	—	8	16	—	—	16	16	—	—	—	—	16	16	—	—	—	—	—	—	—	—
PMA-only CMU Channels (0.6 Gbps-6.5 Gbps)	—	8	—	—	8	—	—	8	12	—	—	8	12	—	—	8	12	12	16	—	—	8	12	12	16	8	12	12	16
PCI Express hard IP Blocks	1	2	1	2		1	2			1	2			2					4	2					4	2	4		
High-Speed LVDS SERDES (up to 1.6 Gbps) *(4)*	28	56	28	28	56	28	44		88	28	44		88	—	44		88	88	98	—	44		88	88	98	44	88	88	98
SPI-4.2 Links	1		1			1	2		4	1	2		4	—	2		4			—	2		4			2	4		

Table 1–1. Stratix IV GX Device Features (Part 2 of 2)

Feature	EP4SGX70		EP4SGX110			EP4SGX180				EP4SGX230				EP4SGX290						EP4SGX360						EP4SGX530			
Package Option	**F780**	**F1152**	**F780**	**F1152**		**F780**	**F1152**		**F1517**	**F780**	**F1152**		**F1517**	**F780**	**F1152**		**F1517**	**F1760**	**F1932**	**F780**	**F1152**		**F1517**	**F1760**	**F1932**	**F1152**	**F1517**	**F1760**	**F1932**
M9K Blocks (256 × 36 bits)	462		660			950				1,235				936						1,248						1,280			
M144K Blocks (2048 × 72 bits)	16		16			20				22				36						48						64			
Total Memory (MLAB+M9K+ M144K) Kbits	7,370		9,564			13,627				17,133				17,248						22,564						27,376			
Embedded Multipliers 18 × 18 *(2)*	384		512			920				1,288				832						1,040					1,024	1,024			
PLLs	3	4	3	4		3	6		8	3	6		8	4	6		8	12	12	4	6		8	12	12	6	8	12	12
User I/Os *(3)*	372	488	372	372	488	372	564	564	744	372	564	564	744	289	564	564	744	880	920	289	564	564	744	880	920	564	744	880	920
Speed Grade (fastest to slowest)	–2×, –3, –4	–2, –3, –4	–2×, –3, –4	–2×, –3, –4	–2, –3, –4	–2×, –3, –4	–2×, –3, –4	–2, –3, –4	–2, –3, –4	–2×, –3, –4	–2×, –3, –4	–2, –3, –4	–2, –3, –4	–2×, –3, –4	–2×, –3, –4	–2, –3, –4	–2, –3, –4	–2, –3, –4	–2, –3, –4	–2×, –3, –4	–2×, –3, –4	–2, –3, –4	–2, –3, –4	–2, –3, –4	–2, –3, –4	–2, –3, –4	–2, –3, –4	–2, –3, –4	–2, –3, –4

Notes to Table 1–1:

(1) The total number of transceivers is divided equally between the left and right side of each device, except for the devices in the F780 package. These devices have eight transceiver channels located only on the right side of the device.

(2) Four multiplier adder mode.

(3) The user I/Os count from pin-out files includes all general purpose I/O, dedicated clock pins, and dual purpose configuration pins. Transceiver pins and dedicated configuration pins are not included in the pin count.

(4) Total pairs of High-Speed LVDS SERDES take the lowest channel count of R_X/T_X.

Table 1–2 summarizes the Stratix IV GX device package options.

Table 1–2. Stratix IV GX Device Package Options *(Note 1)*

Device	F780 (29 mm × 29 mm) *(5)*	F1152 (35 mm × 35 mm) *(5)*	F1152 (35 mm × 35 mm) *(4)*, *(6)*	F1517 (40 mm × 40 mm) *(4)*, *(6)*	F1760 (42.5 mm × 42.5 mm) *(6)*	F1932 (45 mm × 45 mm) *(6)*
EP4SGX70	DF29	—	HF35	—	—	—
EP4SGX110	DF29	FF35	HF35	—	—	—
EP4SGX180	DF29	FF35	HF35	KF40	—	—
EP4SGX230	DF29	FF35	HF35	KF40	—	—
EP4SGX290	FH29 *(2)*	FF35	HF35	KF40	KF43	NF45
EP4SGX360	FH29 *(2)*	FF35	HF35	KF40	KF43	NF45
EP4SGX530	—	—	HH35 *(3)*	KH40 *(3)*	KF43	NF45

Notes to Table 1–2:

(1) Device packages in the same column and marked under the same arrow sign have vertical migration capability.

(2) The 780-pin EP4SGX290 and EP4SGX360 devices are available only in 33 mm × 33 mm Hybrid flip chip package.

(3) The 1152-pin and 1517-pin EP4SGX530 devices are available only in 42.5 mm × 42.5 mm Hybrid flip chip packages.

(4) When migrating between hybrid and flip chip packages, there is an additional keep-out area. For more information, refer to the *Altera Device Package Information Data Sheet*.

(5) Devices listed in this column are available in –2×, –3, and –4 speed grades. These devices do not have on-package decoupling capacitors.

(6) Devices listed in this column are available in –2, –3, and –4 speed grades. These devices have on-package decoupling capacitors. For more information about on-package decoupling capacitor value in each device, refer to Table 1–3.

Table 1–3 lists the Stratix IV GX device on-package decoupling information. On-package decoupling reduces the need for on-board or PCB decoupling capacitors by satisfying the transient current requirements at higher frequencies. The *Power Delivery Network* design tool for Stratix IV devices accounts for the on-package decoupling and reflects the reduced requirements for PCB decoupling capacitors.

Table 1–3. Stratix IV GX Device On-Package Decoupling Information *(Note 1)* (Part 1 of 2)

Ordering Information		V_{CC}	V_{CCIO}	V_{CCL_GXB}	$V_{CCA_L/R}$	V_{CCT} and V_{CCR} (Shared)
EP4SGX70	HF35	2× 1uF + 2×470nF	10nF per bank *(2)*	100nF per transceiver block	100nF	1×470nF + 1×47nF per side
EP4SGX110	HF35	2× 1uF + 2×470nF	10nF per bank *(2)*	100nF per transceiver block	100nF	1×470nF + 1×47nF per side
EP4SGX180	HF35 KF40	2× 1uF + 2×470nF	10nF per bank *(2)*	100nF per transceiver block	100nF	1×470nF + 1×47nF per side

Table 1–3. Stratix IV GX Device On-Package Decoupling Information *(Note 1)* (Part 2 of 2)

Ordering Information		V_{CC}	V_{CCIO}	V_{CCL_GXB}	$V_{CCA_L/R}$	V_{CCT} and V_{CCR} (Shared)
EP4SGX230	HF35 KF40	2×1 uF + 2×470 nF	10 nF per bank *(2)*	100 nF per transceiver block	100 nF	1×470 nF + 1×47 nF per side
EP4SGX290	HF35 KF40 KF43 NF45	4×1 uF + 4×470 nF	10 nF per bank *(2)*	100 nF per transceiver block	100nF	1×470 nF + 1×47 nF per side
EP4SGX360	HF35 KF40 KF43 NF45	4×1 uF + 4×470 nF	10 nF per bank *(2)*	100 nF per transceiver block	100 nF	1×470 nF + 1×47 nF per side
EP4SGX530	HH35 KH40 KF43 NF45	4×1 uF + 4×470 nF	10 nF per bank *(2)*	100 nF per transceiver block	100 nF	1×470 nF + 1×47 nF per side

Notes to Table 1–3:

(1) Table 1–3 is referring to production devices on-package decoupling. For more information about decoupling design of engineering sample (ES) devices, contact Altera Technical Support.

(2) For I/O banks 3(*), 4(*), 7(*), and 8(*) only. There is no OPD for I/O bank 1(*), 2(*), 5(*), and 6(*).

Table 1–4 lists the Stratix IV E device features.

Table 1–4. Stratix IV E Device Features

Feature	EP4SE230	EP4SE360		EP4SE530			EP4SE820		
Package Pin Count	780	780	1152	1152	1517	1760	1152	1517	1760
ALMs	91,200	141,440		212,480			325,220		
LEs	228,000	353,600		531,200			813,050		
High-Speed LVDS SERDES (up to 1.6 Gbps) *(3)*	56	56	88	88	112	112	88	112	132
SPI-4.2 Links	3	3	4	4	6		4	6	6
M9K Blocks (256 × 36 bits)	1,235	1,248		1,280			1610		
M144K Blocks (2048 × 72 bits)	22	48		64			60		
Total Memory (MLAB+M9K+ M144K) Kbits	17,133	22,564		27,376			33,294		
Embedded Multipliers (18 × 18) *(1)*	1,288	1,040		1,024			960		
PLLs	4	4	8	8	12	12	8	12	12
User I/Os *(2)*	488	488	744	744	976	976	744 *(4)*	976 *(4)*	1120 *(4)*
Speed Grade (fastest to slowest)	–2, –3, –4	–2, –3, –4	–2, –3, –4	–2, –3, –4	–2, –3, –4	–2, –3, –4	–3, –4	–3, –4	–3, –4

Notes to Table 1–4:

(1) Four multiplier adder mode.

(2) The user I/O count from the pin-out files include all general purpose I/Os, dedicated clock pins, and dual purpose configuration pins. Transceiver pins and dedicated configuration pins are not included in the pin count.

(3) Total pairs of high-speed LVDS SERDES take the lowest channel count of R_X/T_X.

(4) This data is preliminary.

Table 1–5 summarizes the Stratix IV E device package options.

Table 1–5. Stratix IV E Device Package Options *(Note 1)*

Device	F780 (29 mm × 29 mm) *(4)*, *(5)*	F1152 (35 mm × 35 mm) *(4)*, *(6)*	F1517 (40 mm × 40 mm) *(6)*	F1760 (42.5 mm × 42.5 mm) *(6)*
EP4SE230	F29	—	—	—
EP4SE360	H29 *(2)*	F35	—	—
EP4SE530	—	H35 *(3)*	H40 *(3)*	F43
EP4SE820	—	H35 *(3)*	H40 *(3)*	F43

Notes to Table 1–5:

(1) Device packages in the same column and marked under the same arrow sign have vertical migration capability.

(2) The 780-pin EP4SE360 device is available only in the 33 mm × 33 mm hybrid flip chip package.

(3) The 1152-pin and 1517-pin for EP4SE530 and EP4SE820 devices are available only in the 42.5 mm × 42.5 mm hybrid flip chip package.

(4) When migrating between hybrid and flip chip packages, there is an additional keep-out area. For more information, refer to the *Altera Device Package Information Data Sheet*.

(5) Devices listed in this column do not have on-package decoupling capacitors.

(6) Devices listed in this column have on-package decoupling capacitors. For more information about on-package decoupling capacitor value for each device, refer to Table 1–6.

Table 1–6 lists the Stratix IV E on-package decoupling information. On-package decoupling reduces the need for on-board or PCB decoupling capacitors by satisfying the transient current requirements at higher frequencies. The *Power Delivery Network* design tool for Stratix IV devices accounts for the on-package decoupling and reflects the reduced requirements for PCB decoupling capacitors.

Table 1–6. Stratix IV E Device On-Package Decoupling Information *(Note 1)*

Ordering Information		V_{CC}	V_{CCIO}
EP4SE360	F35	4×1 uF + 4×470 nF	10 nF per bank
EP4SE530	H35 H40 F43	4×1 uF + 4×470 nF	10 nF per bank
EP4SE820	H35 H40 F43	4×1 uF + 4×470 nF	10 nF per bank

Note to Table 1–6:

(1) Table 1–6 is referring to production devices on-package decoupling. For more information about decoupling design of engineering sample (ES) devices, contact Altera Technical Support.

Table 1–7 lists the Stratix IV GT device features.

Table 1–7. Stratix IV GT Device Features (Part 1 of 2)

Feature	EP4S40G2	EP4S40G5	EP4S100G2	EP4S100G3	EP4S100G4	EP4S100G5	
Package Pin Count	1517	1517	1517	1932	1932	1517	1932
ALMs	91,200	212,480	91,200	116,480	141,440	212,480	
LEs	228,000	531,200	228,000	291,200	353,600	531,200	
Total Transceiver Channels	36	36	36	48	48	36	48

Table 1–7. Stratix IV GT Device Features (Part 2 of 2)

Feature	EP4S40G2	EP4S40G5	EP4S100G2	EP4S100G3	EP4S100G4	EP4S100G5	
10G Transceiver Channels (2.488 Gbps-11.3 Gbps with PMA + PCS)	12	12	24	24	24	24	32
8G Transceiver Channels (2.488 Gbps- 8.5 Gbps with PMS + PCS) *(3)*	12	12	0	8	8	0	0
PMA-only CMU Channels (2.488 Gbps - 6.5 Gbps)	12	12	12	16	16	12	16
PCI Express hard IP Blocks	2	2	2	4	4	2	4
High-Speed LVDS SERDES (up to 1.6 Gbps) *(4)*	46	46	46	47	47	46	47
SP1-4.2 Links	2	2	2	2	2	2	2
M9K Blocks (256 × 72 bits)	1,235	1,280	1,235	936	1,248	1,280	
M144K Blocks (2048 × 72 bits)	22	64	22	36	48	64	
Total Memory (MLAB + M9K + M144K) Kbits	17,133	27,376	17,133	17,248	22,564	27,376	
Embedded Multipliers 18 × 18 *(1)*	1,288	1,024	1,288	832	1,024	1,024	
PLLs	8	8	8	12	12	8	12
User I/Os *(2)*, *(5)*	654	654	654	781	781	654	781
Speed Grade (fastest to slowest)	–1, –2, –3	–1, –2, –3	–1, –2, –3	–1, –2, –3	–1, –2, –3	–1, –2, –3	–1, –2, –3

Notes to Table 1–7:

(1) Four multiplier adder mode.

(2) The user I/O count from the pin-out files include all general purpose I/Os, dedicated clock pins, and dual purpose configuration pins. Transceiver pins and dedicated configuration pins are not included in the pin count.

(3) You can configure all 10G transceiver channels as 8G transceiver channels. For example, the EP4S40G2F40 device has twenty-four 8G transceiver channels and the EP4S100G5F45 device has thirty-two 8G transceiver channels.

(4) Total pairs of high-speed LVDS SERDES take the lowest channel count of R_X/T_X.

(5) This data is preliminary.

Table 1–8 summarizes the resource counts for the Stratix IV GT devices.

Table 1–8. Stratix IV GT Device Package Options *(Note 1)*

Device	1517 Pin (40 mm × 40 mm) *(3)*	1932 Pin (45 mm × 45 mm)
Stratix IV GT 40 G Devices		
EP4S40G2	F40	—
EP4S40G5	H40 *(2)*	—
Stratix IV GT 100 G Devices		
EP4S100G2	F40	—
EP4S100G3	—	F45
EP4S100G4	—	F45
EP4S100G5	H40 *(2)*	F45

Notes to Table 1–8:

(1) Devices under the same arrow sign have vertical migration capability.

(2) EP4S40G5 and EP4S100G5 devices with 1517 pin-count are only available in 42.5-mm x 42.5-mm hybrid flip chip packages.

(3) When migrating between hybrid and flip chip packages, there is an additional keep-out area. For more information, refer to the *Altera Device Package Information Data Sheet*.

Table 1–9 lists the Stratix IV GT on-package decoupling information. On-package decoupling reduces the need for on-board or PCB decoupling capacitors by satisfying the transient current requirements at higher frequencies. The *Power Delivery Network* design tool for Stratix IV devices accounts for the on-package decoupling and reflects the reduced requirements for PCB decoupling capacitors.

Table 1–9. Stratix IV GT Device On-Package Decoupling Information *(Note 1)*

Ordering Information	V_{CC}	V_{CCIO}	V_{CCL_GXB}	$V_{CCA_L/R}$	$V_{CCT_L/R}$	$V_{CCR_L/R}$
EP4S40G2F40 EP4S100G2F40	2× 1 uF + 2× 470 nF	10 nF per bank *(2)*	100 nF per transceiver block	100 nF	100 nF	100 nF
EP4S100G3F45	4× 1 uF + 4× 470 nF	10 nF per bank *(2)*	100 nF per transceiver block	100 nF	100 nF	100 nF
EP4S100G4F45						
EP4S40G5H40						
EP4S100G5H40						
EP4S100G5F45						

Notes to Table 1–9:

(1) Table 1–9 is referring to production devices on-package decoupling. For more information about decoupling design of engineering sample (ES) devices, contact Altera Technical Support.

(2) For I/O banks 3(*), 4(*), 7(*), and 8(*) only. There is no OPD for I/O bank 1(*), 2(*), 5(*), and 6(*).

Integrated Software Platform

The Quartus II software provides an integrated environment for HDL and schematic design entry, compilation and logic synthesis, full simulation and advanced timing analysis, SignalTap II logic analyzer, and device configuration of Stratix IV designs. The Quartus II software provides the MegaWizard™ Plug-In Manager user interface to generate different functional blocks, such as memory, PLL, and digital signal processing logic. For transceivers, the Quartus II software provides the ALTGX MegaWizard Plug-In Manager interface that guides you through configuration of the transceiver based on your application requirements.

The Stratix IV GX and GT transceivers allow you to implement low-power and reliable high-speed serial interface applications with its fully reconfigurable hardware, optimal signal integrity, and integrated Quartus II software platform.

For more information about the Quartus II software features, refer to the *Quartus II Handbook*.

Ordering Information

This section describes the Stratix IV E, GT, and GX devices ordering information. Figure 1–4 describes the ordering codes for Stratix IV GX and E devices.

Figure 1–4. Stratix IV GX and E Device Packaging Ordering Information

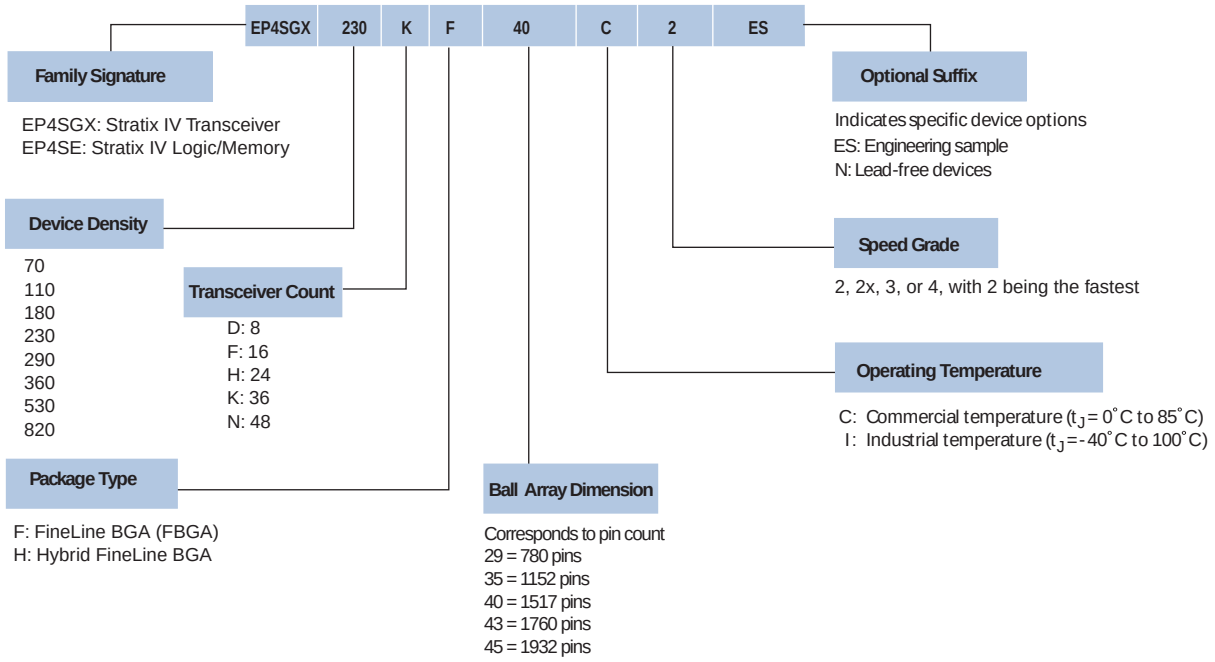
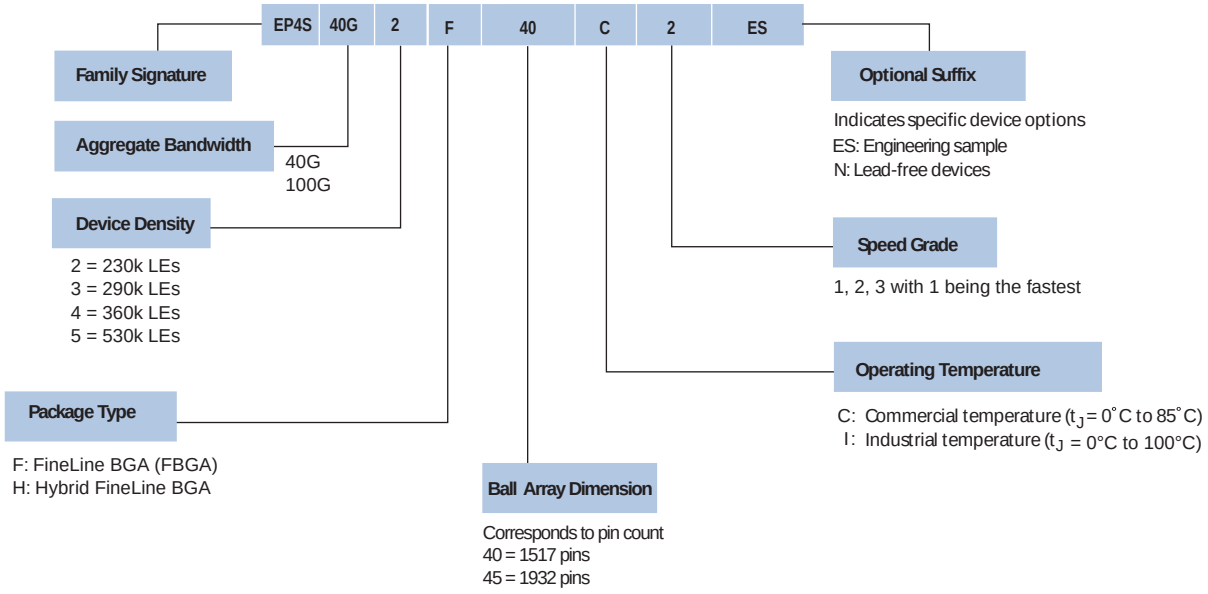


Figure 1–5 describes the ordering codes for Stratix IV GT devices.

Figure 1–5. Stratix IV GT Device Packaging Ordering Information



Document Revision History

Table 1–10 shows the revision history for this chapter.

Table 1–10. Document Revision History (Part 1 of 2)

Date and Document Version	Changes Made	Summary of Changes
November 2009 v3.0	■ Updated the "Stratix IV Device Family Overview", "Feature Summary", "Stratix IV GT Devices", "High-Speed Transceiver Features", "FPGA Fabric and I/O Features", "Highest Aggregate Data Bandwidth", "System Integration", and "Integrated Software Platform" sections. ■ Added Table 1–3, Table 1–6, and Table 1–9. ■ Updated Table 1–1, Table 1–2, Table 1–4, Table 1–5, Table 1–7, and Table 1–8. ■ Updated Figure 1–3, Figure 1–4, and Figure 1–5. ■ Minor text edits.	—
June 2009 v2.4	■ Updated Table 1–1. ■ Minor text edits.	—
April 2009 v2.3	■ Added Table 1–5, Table 1–6, and Figure 1–3. ■ Updated Figure 1–5. ■ Updated Table 1–1, Table 1–2, Table 1–3, and Table 1–4. ■ Updated "Introduction", "Feature Summary", "Stratix IV GX Devices", "Stratix IV GT Devices", "Architecture Features", and "FPGA Fabric and I/O Features"	—

Table 1–10. Document Revision History (Part 2 of 2)

Date and Document Version	Changes Made	Summary of Changes
March 2009 v2.2	■ Updated “Feature Summary”, “Stratix IV GX Devices”, “Stratix IV E Device”, “Stratix IV GT Devices”, “Signal Integrity” ■ Removed Tables 1-5 and 1-6 ■ Updated Figure 1–4	—
March 2009 v2.1	■ Updated “Introduction”, “Feature Summary”, “Stratix IV Device Diagnostic Features”, “Signal Integrity”, “Clock Networks”,“High-Speed Differential I/O with DPA and Soft-CDR”, “System Integration”, and “Ordering Information” sections. ■ Added “Stratix IV GT 100G Devices” and “Stratix IV GT 100G Transceiver Bandwidth” sections. ■ Updated Table 1–1, Table 1–2, Table 1–3, and Table 1–4. ■ Added Table 1–5 and Table 1–6. ■ Updated Figure 1–3 and Figure 1–4. ■ Added Figure 1–5. ■ Removed “Referenced Documents” section.	—
November 2008 v2.0	■ Updated “Feature Summary” on page 1–1. ■ Updated “Stratix IV Device Diagnostic Features” on page 1–7. ■ Updated “FPGA Fabric and I/O Features” on page 1–8. ■ Updated Table 1–1. ■ Updated Table 1–2. ■ Updated “Table 1–5 shows the total number of transceivers available in the Stratix IV GT Device.” on page 1–15.	—
July 2008 v1.1	Revised “Introduction”.	—
May 2008 v1.0	Initial Release.	—

2. Logic Array Blocks and Adaptive Logic Modules in Stratix IV Devices

SIV51002-3.0

This chapter describes the features of the LABs in the Stratix IV core fabric. LABs are made up of ALMs you can configure to implement logic functions, arithmetic functions, and register functions.

Logic array blocks (LABs) and adaptive logic modules (ALMs) are the basic building blocks of the Stratix® IV device. You can use these to configure logic functions, arithmetic functions, and register functions. The ALM provides advanced features with efficient logic utilization and is completely backward-compatible.

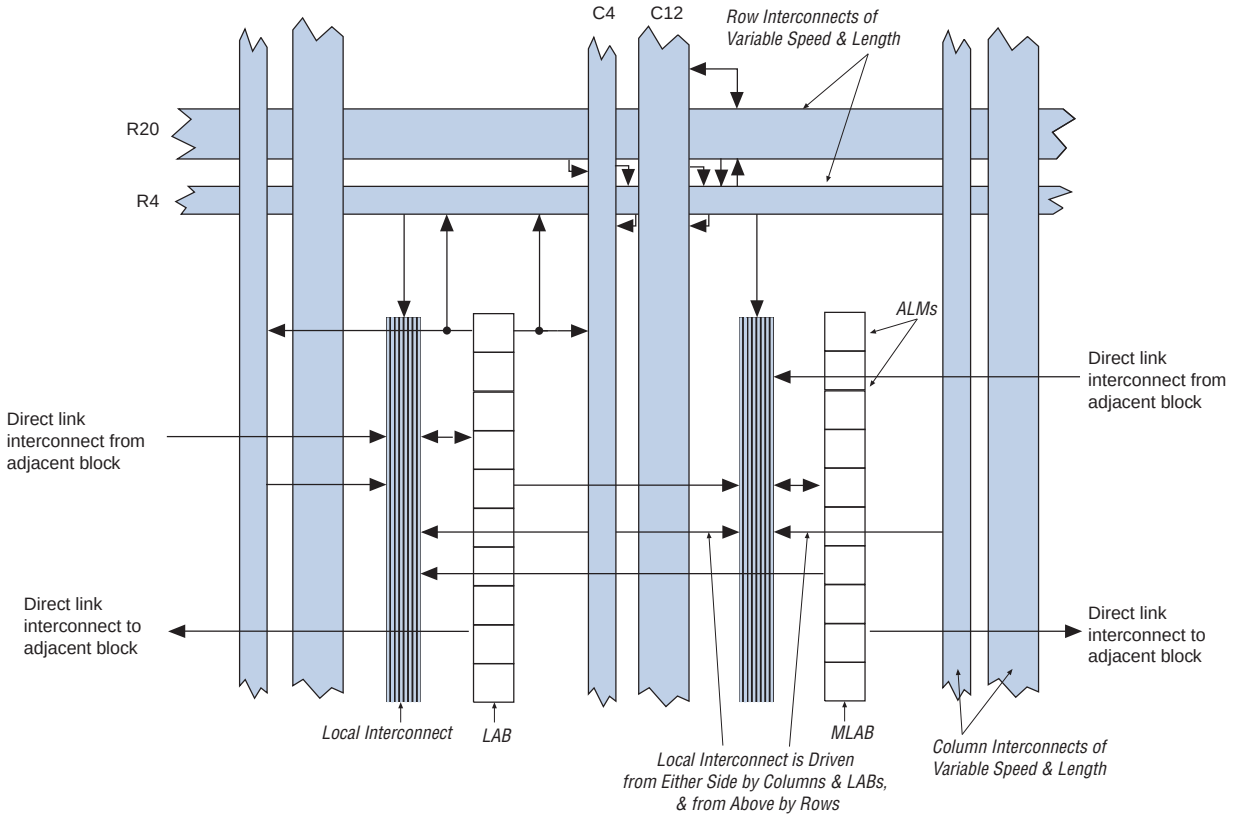
This chapter contains the following sections:

- "Logic Array Blocks" on page 2–1
- "Adaptive Logic Modules" on page 2–5

Logic Array Blocks

Each LAB consists of ten ALMs, various carry chains, shared arithmetic chains, LAB control signals, local interconnect, and register chain connection lines. The local interconnect transfers signals between ALMs in the same LAB. The direct link interconnect allows the LAB to drive into the local interconnect of its left and right neighbors. Register chain connections transfer the output of the ALM register to the adjacent ALM register in the LAB. The Quartus® II Compiler places associated logic in the LAB or adjacent LABs, allowing the use of local, shared arithmetic chain, and register chain connections for performance and area efficiency. Figure 2–1 shows the Stratix IV LAB structure and the LAB interconnects.

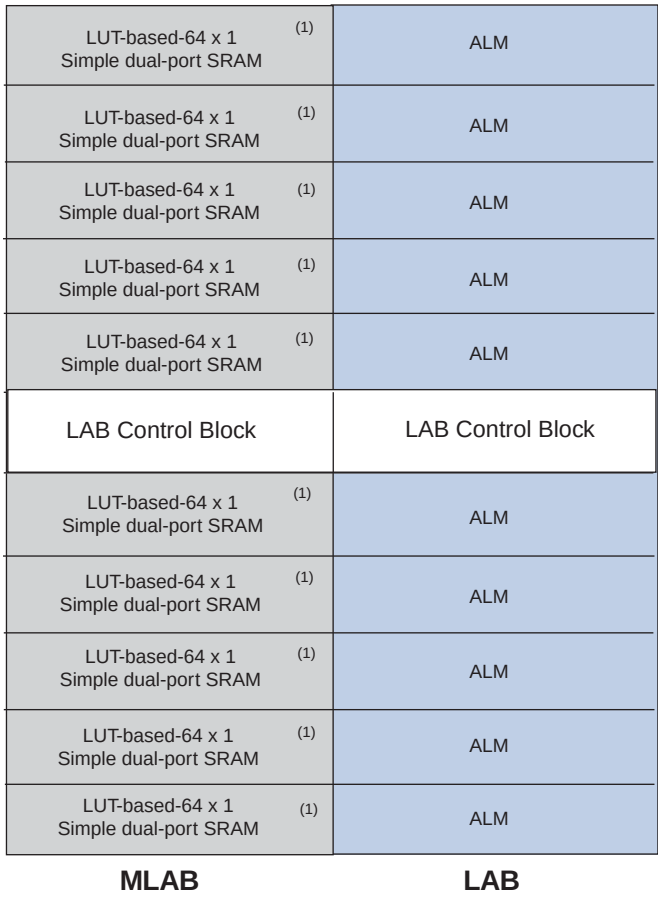
Figure 2–1. Stratix IV LAB Structure



The LAB of the Stratix IV device has a derivative called memory LAB (MLAB), which adds look-up table (LUT)-based SRAM capability to the LAB, as shown in Figure 2–2. The MLAB supports a maximum of 640 bits of simple dual-port static random access memory (SRAM). You can configure each ALM in an MLAB as either a 64 × 1 or a 32 × 2 block, resulting in a configuration of either a 64 × 10 or a 32 × 20 simple dual-port SRAM block. MLAB and LAB blocks always coexist as pairs in all Stratix IV families. MLAB is a superset of the LAB and includes all LAB features.

The MLAB is described in detail in the *TriMatrix Embedded Memory Blocks in Stratix IV Devices* chapter.

Figure 2–2. Stratix IV LAB and MLAB Structure



Note to Figure 2–2:

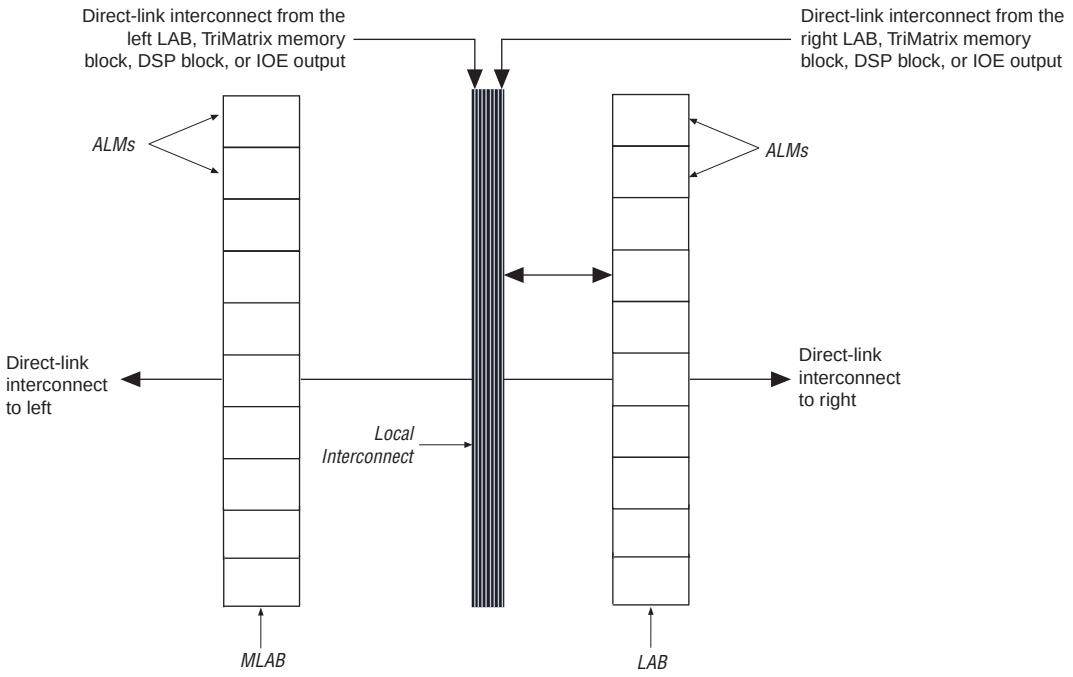
(1) You can use the MLAB ALM as a regular LAB ALM or configure it as a dual-port SRAM, as shown.

LAB Interconnects

The LAB local interconnect can drive ALMs in the same LAB. It is driven by column and row interconnects and ALM outputs in the same LAB. Neighboring LABs/MLABs, M9K RAM blocks, M144K blocks, or DSP blocks from the left or right can also drive the LAB's local interconnect through the direct link connection. The direct link connection feature minimizes the use of row and column interconnects, providing higher performance and flexibility. Each LAB can drive 30 ALMs through fast-local and direct-link interconnects.

Figure 2–3 shows the direct-link connection.

Figure 2–3. Direct-Link Connection



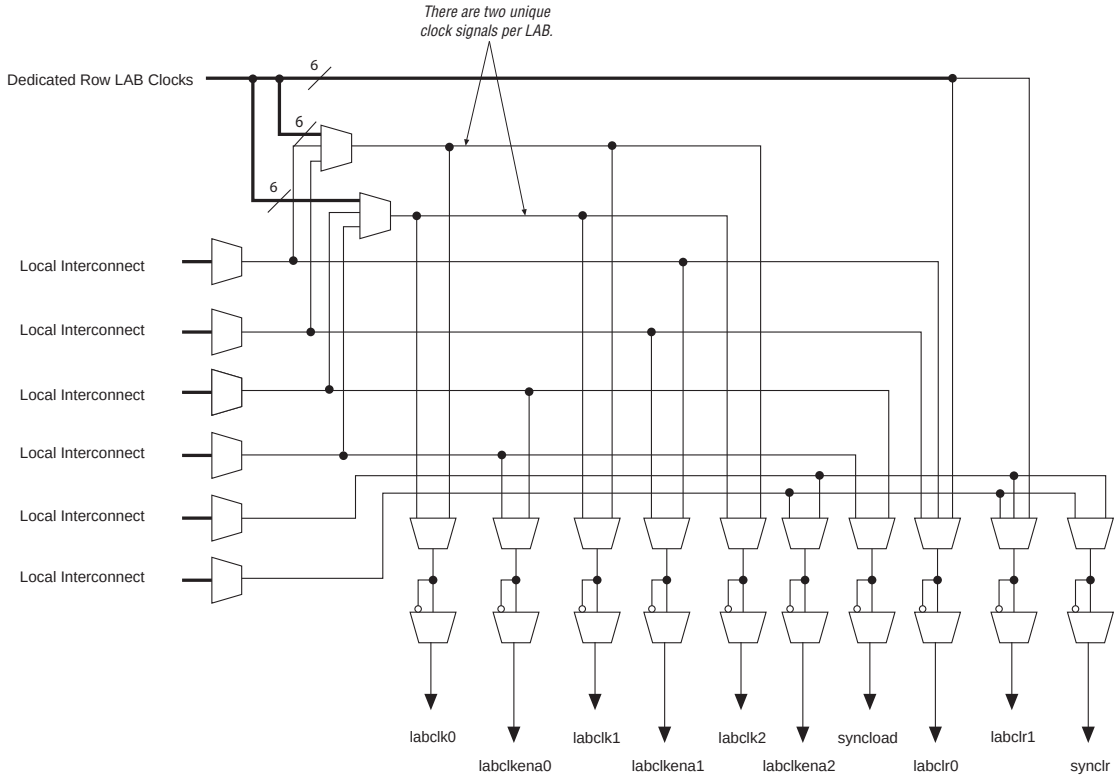
LAB Control Signals

Each LAB contains dedicated logic for driving control signals to its ALMs. Control signals include three clocks, three clock enables, two asynchronous clears, a synchronous clear, and synchronous load control signals. This gives a maximum of 10 control signals at a time. Although you generally use synchronous-load and clear signals when implementing counters, you can also use them with other functions.

Each LAB has two unique clock sources and three clock enable signals, as shown in Figure 2–4. The LAB control block can generate up to three clocks using two clock sources and three clock enable signals. Each LAB's clock and clock enable signals are linked. For example, any ALM in a particular LAB using the `labclk1` signal also uses the `labclkena1` signal. If the LAB uses both the rising and falling edges of a clock, it also uses two LAB-wide clock signals. De-asserting the clock enable signal turns off the corresponding LAB-wide clock.

The LAB row clocks [5..0] and LAB local interconnects generate the LAB-wide control signals. The MultiTrack interconnect's inherent low skew allows clock and control signal distribution in addition to data.

Figure 2–4. LAB-Wide Control Signals



Adaptive Logic Modules

The ALM is the basic building block of logic in the Stratix IV architecture. It provides advanced features with efficient logic utilization. Each ALM contains a variety of LUT-based resources that can be divided between two combinational adaptive LUTs (ALUTs) and two registers. With up to eight inputs for the two combinational ALUTs, one ALM can implement various combinations of two functions. This adaptability allows an ALM to be completely backward-compatible with four-input LUT architectures. One ALM can also implement any function with up to six inputs and certain seven-input functions.

In addition to the adaptive LUT-based resources, each ALM contains two programmable registers, two dedicated full adders, a carry chain, a shared arithmetic chain, and a register chain. Through these dedicated resources, an ALM can efficiently implement various arithmetic functions and shift registers. Each ALM drives all types of interconnects: local, row, column, carry chain, shared arithmetic chain, register chain, and direct link. Figure 2–5 shows a high-level block diagram of the Stratix IV ALM.

Figure 2–5. High-Level Block Diagram of the Stratix IV ALM

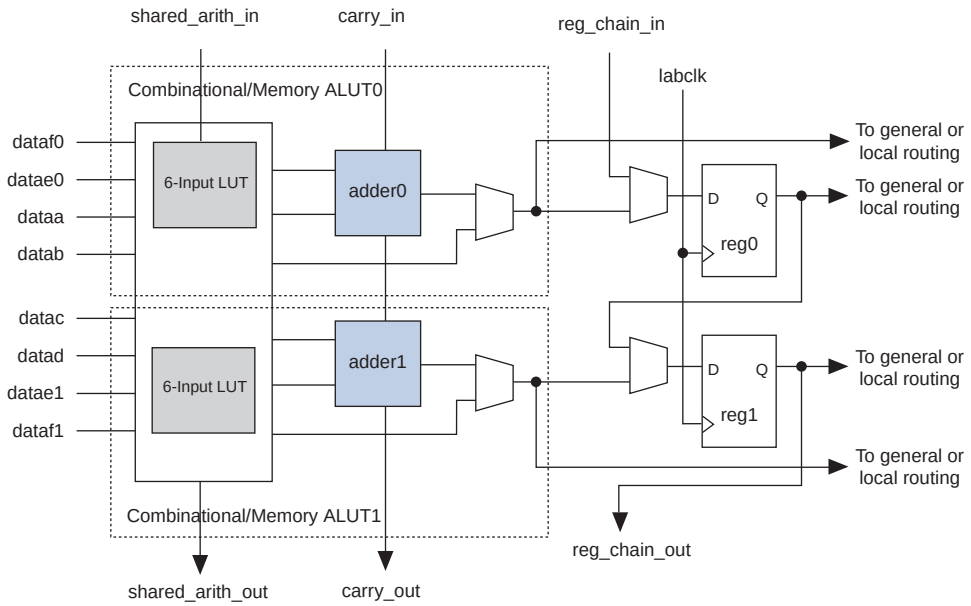
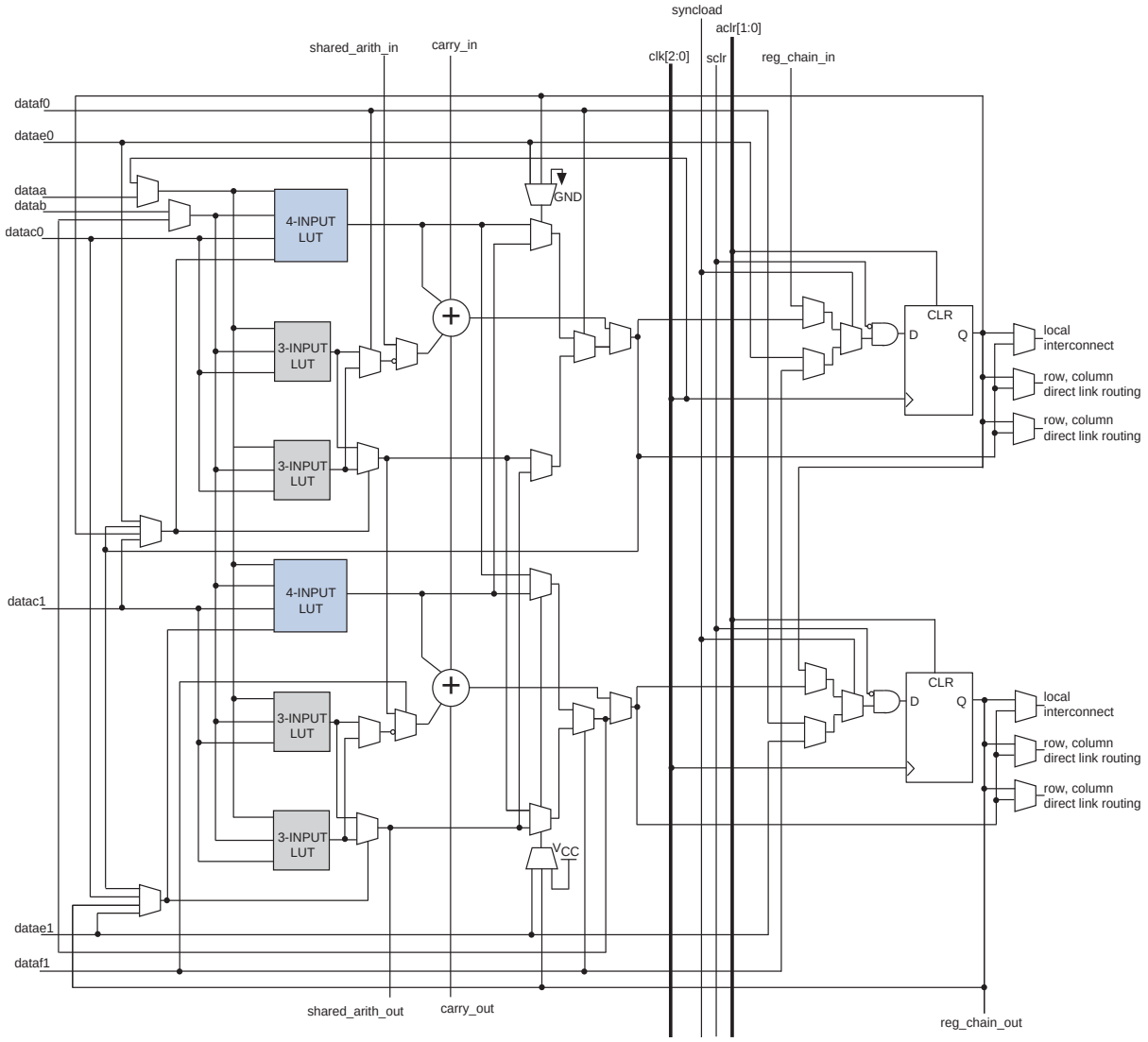


Figure 2–6 shows a detailed view of all the connections in an ALM.

Figure 2–6. Stratix IV ALM Connection Details



One ALM contains two programmable registers. Each register has data, clock, clock enable, synchronous and asynchronous clear, and synchronous load and clear inputs. Global signals, general-purpose I/O pins, or any internal logic can drive the register's clock and clear-control signals. Either general-purpose I/O pins or internal logic can drive the clock enable. For combinational functions, the register is bypassed and the output of the LUT drives directly to the outputs of an ALM.

Each ALM has two sets of outputs that drive the local, row, and column routing resources. The LUT, adder, or register outputs can drive these output drivers (refer to Figure 2–6). For each set of output drivers, two ALM outputs can drive column, row, or direct-link routing connections. One of these ALM outputs can also drive local interconnect resources. This allows the LUT or adder to drive one output while the register drives another output.

This feature, called register packing, improves device utilization because the device can use the register and the combinational logic for unrelated functions. Another special packing mode allows the register output to feed back into the LUT of the same ALM so that the register is packed with its own fan-out LUT. This provides another mechanism for improved fitting. The ALM can also drive out registered and unregistered versions of the LUT or adder output.

ALM Operating Modes

The Stratix IV ALM operates in one of the following modes:

- Normal
- Extended LUT
- Arithmetic
- Shared Arithmetic
- LUT-Register

Each mode uses ALM resources differently. In each mode, eleven available inputs to an ALM—the eight data inputs from the LAB local interconnect, carry-in from the previous ALM or LAB, the shared arithmetic chain connection from the previous ALM or LAB, and the register chain connection—are directed to different destinations to implement the desired logic function. LAB-wide signals provide clock, asynchronous clear, synchronous clear, synchronous load, and clock enable control for the register. These LAB-wide signals are available in all ALM modes.

For more information on the LAB-wide control signals, refer to "LAB Control Signals" on page 2–4.

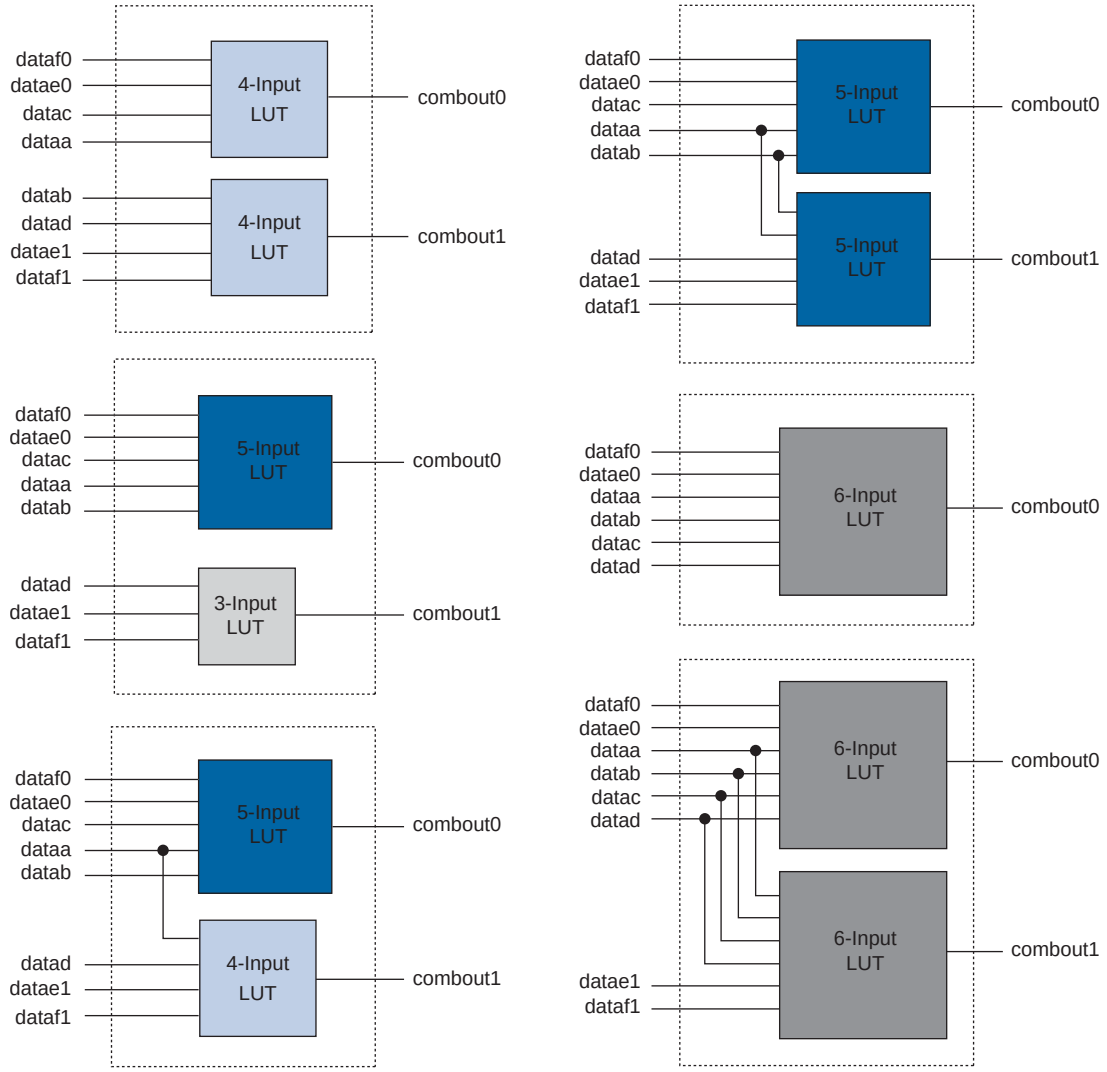
The Quartus II software and supported third-party synthesis tools, in conjunction with parameterized functions such as the library of parameterized modules (LPM) functions, automatically choose the appropriate mode for common functions such as counters, adders, subtractors, and arithmetic functions.

Normal Mode

Normal mode is suitable for general logic applications and combinational functions. In this mode, up to eight data inputs from the LAB local interconnect are inputs to the combinational logic. Normal mode allows two functions to be implemented in one Stratix IV ALM, or a single function of up to six inputs. The ALM can support certain combinations of completely independent functions and various combinations of functions that have common inputs.

Figure 2–7 shows the supported LUT combinations in normal mode.

Figure 2–7. ALM in Normal Mode *(Note 1)*



Note to Figure 2–7:

(1) Combinations of functions with fewer inputs than those shown are also supported. For example, combinations of functions with the following number of inputs are supported: 4 and 3, 3 and 3, 3 and 2, and 5 and 2.

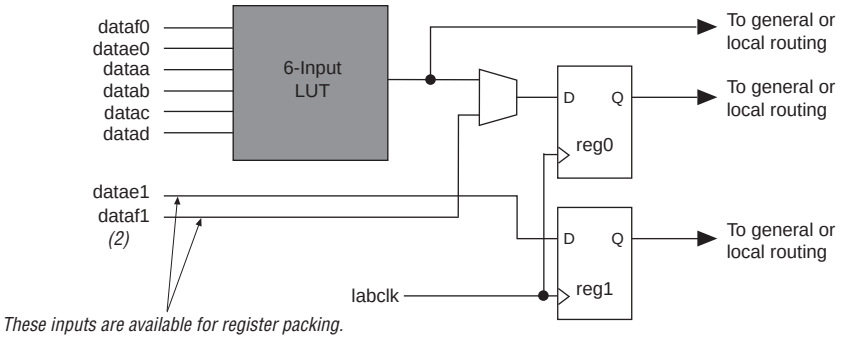
Normal mode provides complete backward-compatibility with four-input LUT architectures.

For the packing of 2 five-input functions into one ALM, the functions must have at least two common inputs. The common inputs are `dataa` and `datab`. The combination of a four-input function with a five-input function requires one common input (either `dataa` or `datab`).

In the case of implementing 2 six-input functions in one ALM, four inputs must be shared and the combinational function must be the same. In a sparsely used device, functions that could be placed in one ALM may be implemented in separate ALMs by the Quartus II software to achieve the best possible performance. As a device begins to fill up, the Quartus II software automatically utilizes the full potential of the Stratix IV ALM. The Quartus II Compiler automatically searches for functions using common inputs or completely independent functions to be placed in one ALM to make efficient use of device resources. In addition, you can manually control resource usage by setting location assignments.

Any six-input function can be implemented using inputs `dataa`, `datab`, `datac`, `datad`, and either `datae0` and `dataf0` or `datae1` and `dataf1`. If `datae0` and `dataf0` are utilized, the output is driven to `register0`, and/or `register0` is bypassed and the data drives out to the interconnect using the top set of output drivers (refer to Figure 2–8). If `datae1` and `dataf1` are used, the output either drives to `register1` or bypasses `register1` and drives to the interconnect using the bottom set of output drivers. The Quartus II Compiler automatically selects the inputs to the LUT. ALMs in normal mode support register packing.

Figure 2–8. Input Function in Normal Mode *(Note 1)*



Notes to Figure 2–8:

(1) If `datae1` and `dataf1` are used as inputs to a six-input function, `datae0` and `dataf0` are available for register packing.

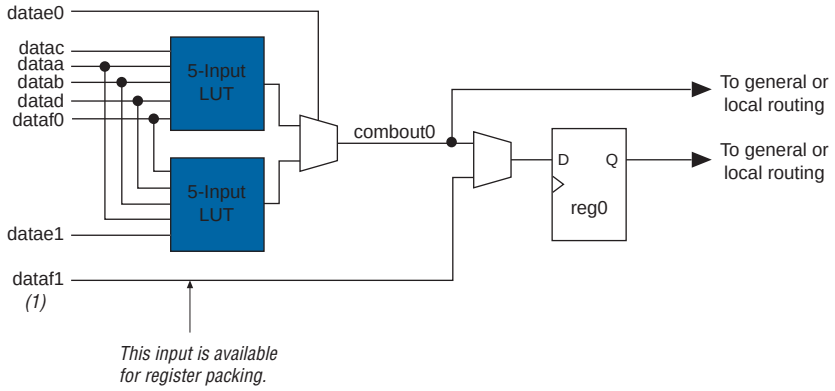
(2) The `dataf1` input is available for register packing only if the six-input function is unregistered.

Extended LUT Mode

Use extended LUT mode to implement a specific set of seven-input functions. The set must be a 2-to-1 multiplexer fed by two arbitrary five-input functions sharing four inputs. Figure 2–9 shows the template of supported seven-input functions using extended LUT mode. In this mode, if the seven-input function is unregistered, the unused eighth input is available for register packing.

Functions that fit into the template shown in Figure 2–9 occur naturally in designs. These functions often appear in designs as “if-else” statements in Verilog HDL or VHDL code.

Figure 2–9. Template for Supported Seven-Input Functions in Extended LUT Mode



Note to Figure 2–9:

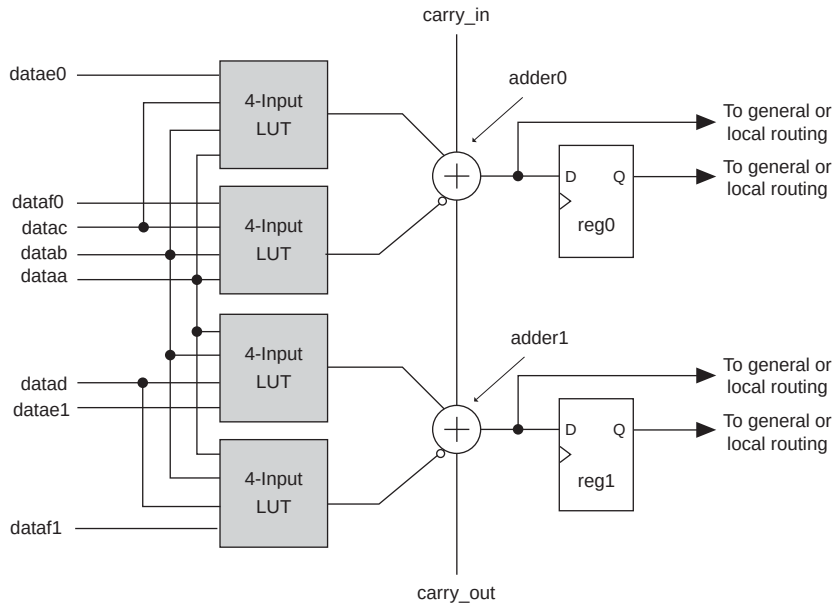
(1) If the seven-input function is unregistered, the unused eighth input is available for register packing. The second register, `reg1`, is not available.

Arithmetic Mode

Arithmetic mode is ideal for implementing adders, counters, accumulators, wide parity functions, and comparators. The ALM in arithmetic mode uses two sets of 2 four-input LUTs along with two dedicated full adders. The dedicated adders allow the LUTs to be available to perform pre-adder logic; therefore, each adder can add the output of 2 four-input functions.

The four LUTs share `dataa` and `datab` inputs. As shown in Figure 2–10, the carry-in signal feeds to `adder0` and the carry-out from `adder0` feeds to the carry-in of `adder1`. The carry-out from `adder1` drives to `adder0` of the next ALM in the LAB. ALMs in arithmetic mode can drive out registered and/or unregistered versions of the adder outputs.

Figure 2–10. ALM in Arithmetic Mode



While operating in arithmetic mode, the ALM can support simultaneous use of the adder's carry output along with combinational logic outputs. In this operation, adder output is ignored. Using the adder with combinational logic output provides resource savings of up to 50% for functions that can use this ability.

Arithmetic mode also offers clock enable, counter enable, synchronous up/down control, add/subtract control, synchronous clear, and synchronous load. The LAB local interconnect data inputs generate the clock enable, counter enable, synchronous up/down, and add/subtract control signals. These control signals are good candidates for the inputs that are shared between the four LUTs in the ALM. The synchronous clear and synchronous load options are LAB-wide signals that affect all registers in the LAB. These signals can also be individually disabled or enabled per register. The Quartus II software automatically places any registers that are not used by the counter into other LABs.

Carry Chain

The carry chain provides a fast carry function between the dedicated adders in arithmetic or shared-arithmetic mode. The two-bit carry select feature in Stratix IV devices halves the propagation delay of carry chains within the ALM. Carry chains can begin in either the first ALM or the fifth ALM in the LAB. The final carry-out signal is routed to the ALM, where it is fed to local, row, or column interconnects.

The Quartus II Compiler automatically creates carry-chain logic during design processing, or you can create it manually during design entry. Parameterized functions such as LPM functions automatically take advantage of carry chains for the appropriate functions.

The Quartus II Compiler creates carry chains longer than 20 (10 ALMs in arithmetic or shared arithmetic mode) by linking LABs together automatically. For enhanced fitting, a long carry chain runs vertically, allowing fast horizontal connections to TriMatrix memory and DSP blocks. A carry chain can continue as far as a full column.

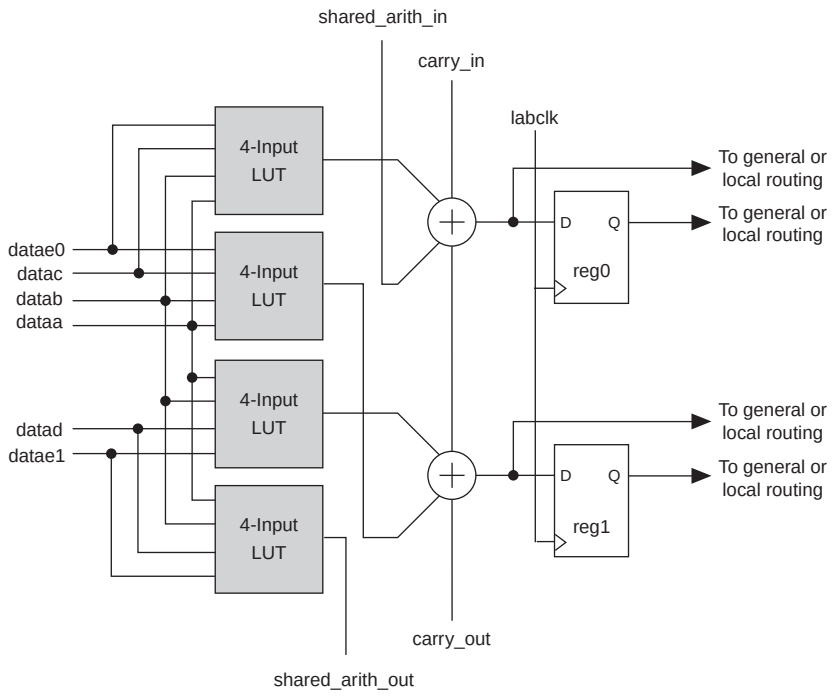
To avoid routing congestion in one small area of the device when a high fan-in arithmetic function is implemented, the LAB can support carry chains that only utilize either the top half or bottom half of the LAB before connecting to the next LAB. This leaves the other half of the ALMs in the LAB available for implementing narrower fan-in functions in normal mode. Carry chains that use the top five ALMs in the first LAB carry into the top half of the ALMs in the next LAB within the column. Carry chains that use the bottom five ALMs in the first LAB carry into the bottom half of the ALMs in the next LAB within the column. In every alternate LAB column, the top half can be bypassed; in the other MLAB columns, the bottom half can be bypassed.

For more information about carry-chain interconnects, refer to "ALM Interconnects" on page 2–17.

Shared Arithmetic Mode

In shared arithmetic mode, the ALM can implement a three-input add within the ALM. In this mode, the ALM is configured with 4 four-input LUTs. Each LUT either computes the sum of three inputs or the carry of three inputs. The output of the carry computation is fed to the next adder (either to `adder1` in the same ALM or to `adder0` of the next ALM in the LAB) using a dedicated connection called the shared arithmetic chain. This shared arithmetic chain can significantly improve the performance of an adder tree by reducing the number of summation stages required to implement an adder tree. Figure 2–11 shows the ALM using this feature.

Figure 2–11. ALM in Shared Arithmetic Mode



You can find adder trees in many different applications. For example, the summation of the partial products in a logic-based multiplier can be implemented in a tree structure. Another example is a correlator function that can use a large adder tree to sum filtered data samples in a given time frame to recover or de-spread data that was transmitted utilizing spread-spectrum technology.

Shared Arithmetic Chain

The shared arithmetic chain available in enhanced arithmetic mode allows the ALM to implement a three-input add. This significantly reduces the resources necessary to implement large adder trees or correlator functions.

The shared arithmetic chains can begin in either the first or sixth ALM in the LAB. The Quartus II Compiler creates shared arithmetic chains longer than 20 (10 ALMs in arithmetic or shared arithmetic mode) by linking LABs together automatically. For enhanced fitting, a long shared arithmetic chain runs vertically, allowing fast horizontal connections to the TriMatrix memory and DSP blocks. A shared arithmetic chain can continue as far as a full column.

Similar to the carry chains, the top and bottom halves of shared arithmetic chains in alternate LAB columns can be bypassed. This capability allows the shared arithmetic chain to cascade through half of the ALMs in an LAB while leaving the other half available for narrower fan-in functionality. Every other LAB column is top-half by-passable, while the other LAB columns are bottom-half by-passable.

For more information on shared arithmetic chain interconnect, refer to "ALM Interconnects" on page 2–17.

LUT-Register Mode

LUT-Register mode allows third-register capability within an ALM. Two internal feedback loops allow combinational `ALUT1` to implement the master latch and combinational `ALUT0` to implement the slave latch needed for the third register. The LUT register shares its clock, clock enable, and asynchronous clear sources with the top dedicated register. Figure 2–12 shows the register constructed using two combinational blocks within the ALM.

Figure 2–12. LUT Register from Two Combinational Blocks

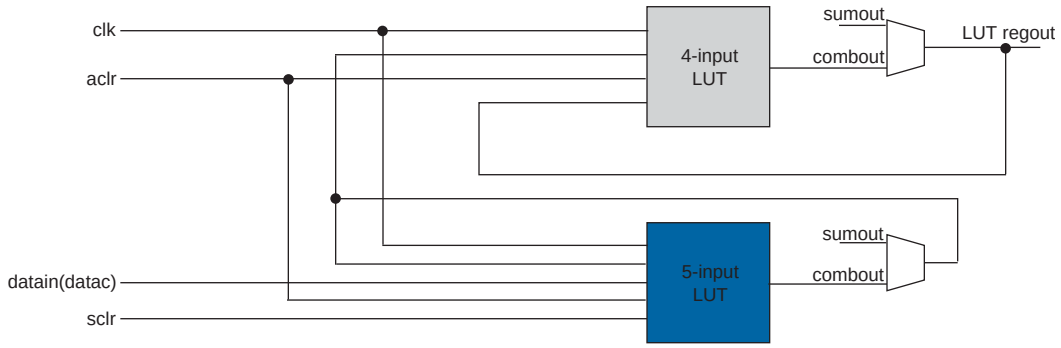
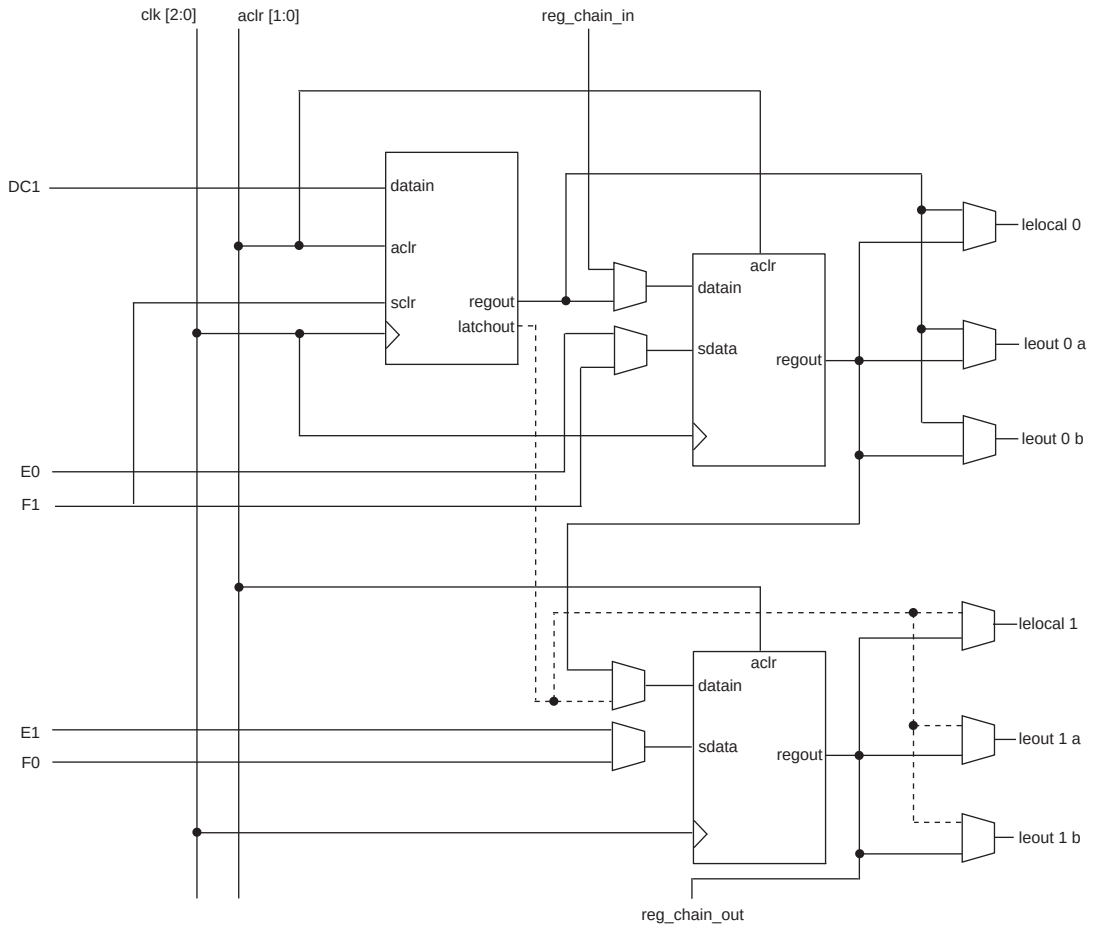


Figure 2–13 shows the ALM in LUT-register mode.

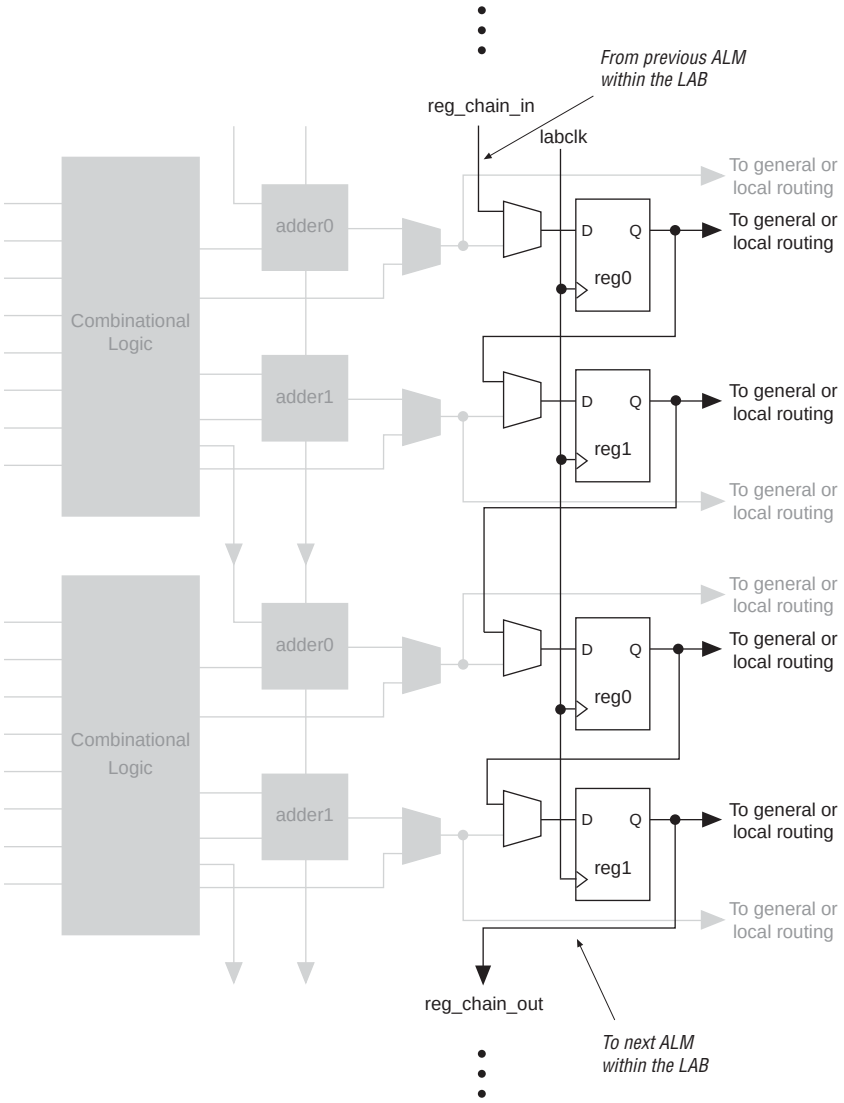
Figure 2–13. ALM in LUT-Register Mode with Three-Register Capability



Register Chain

In addition to general routing outputs, ALMs in the LAB have register-chain outputs. Register-chain routing allows registers in the same LAB to be cascaded together. The register-chain interconnect allows the LAB to use LUTs for a single combinational function and the registers to be used for an unrelated shift-register implementation. These resources speed up connections between ALMs while saving local interconnect resources (refer to Figure 2–14). The Quartus II Compiler automatically takes advantage of these resources to improve utilization and performance.

Figure 2–14. Register Chain within the LAB *(Note 1)*



Note to Figure 2–14:

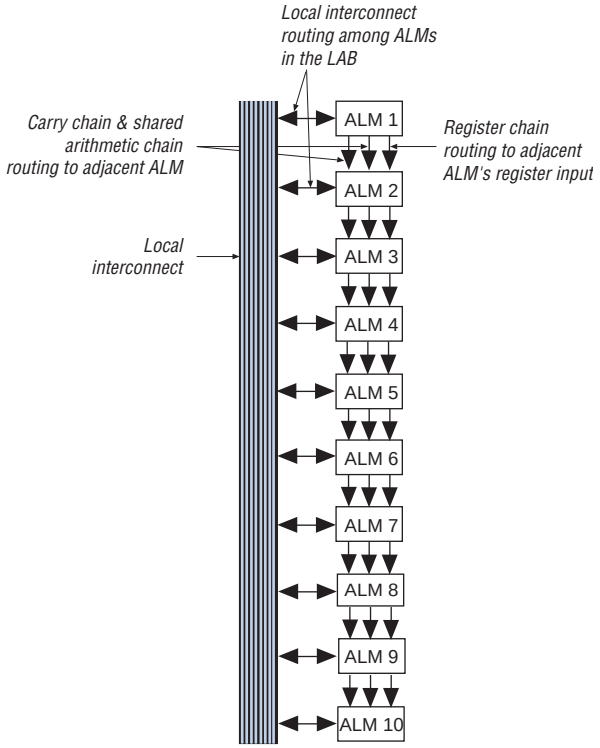
(1) You can use the combinational or adder logic to implement an unrelated, un-registered function.

For more information about register chain interconnect, refer to "ALM Interconnects" on page 2–17.

ALM Interconnects

There are three dedicated paths between ALMs: register cascade, carry chain, and shared arithmetic chain. Stratix IV devices include an enhanced interconnect structure in LABs for routing shared arithmetic chains and carry chains for efficient arithmetic functions. The register chain connection allows the register output of one ALM to connect directly to the register input of the next ALM in the LAB for fast shift registers. These ALM-to-ALM connections bypass the local interconnect. The Quartus II compiler automatically takes advantage of these resources to improve utilization and performance. Figure 2–15 shows the shared arithmetic chain, carry chain, and register chain interconnects.

Figure 2–15. Shared Arithmetic Chain, Carry Chain, and Register Chain Interconnects



Clear and Preset Logic Control

LAB-wide signals control the logic for the register's clear signal. The ALM directly supports an asynchronous clear function. You can achieve the register preset through the Quartus II software's **NOT-gate push-back logic** option. Each LAB supports up to two clears.

Stratix IV devices provide a device-wide reset pin (`DEV_CLRn`) that resets all the registers in the device. An option set before compilation in the Quartus II software controls this pin. This device-wide reset overrides all other control signals.

LAB Power Management Techniques

The following techniques are used to manage static and dynamic power consumption within the LAB:

- To save AC power, the Quartus II software forces all adder inputs low when ALM adders are not in use.
- Stratix IV LABs operate in high-performance mode or low-power mode. The Quartus II software automatically chooses the appropriate mode for the LAB, based on the design, to optimize speed versus leakage trade-offs.
- Clocks represent a significant portion of dynamic power consumption due to their high switching activity and long paths. The LAB clock that distributes a clock signal to registers within an LAB is a significant contributor to overall clock power consumption. Each LAB's clock and clock enable signal are linked. For example, a combinational ALUT or register in a particular LAB using the `labclk1` signal also uses the `labclkena1` signal. To disable an LAB-wide clock power consumption without disabling the entire clock tree, use the LAB-wide clock enable to gate the LAB-wide clock. The Quartus II software automatically promotes register-level clock enable signals to the LAB-level. All registers within the LAB that share a common clock and clock enable are controlled by a shared, gated clock. To take advantage of these clock enables, use a clock-enable construct in your HDL code for the registered logic.

For more information about implementing static and dynamic power consumption within the LAB, refer to the *Power Optimization* chapter in volume 2 of the *Quartus II Handbook*.

Document Revision History

Table 2–1 shows the revision history for this chapter. .

Table 2–1. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
November 2009 v3.0	■ Updated graphics. ■ Minor text edits.	—
June 2009 v2.2	■ Removed the Conclusion section. ■ Added introductory sentences to improve search ability. ■ Minor text edits.	—
March 2009 v2.1	Removed "Referenced Documents" section.	—
November 2008 v2.0	■ Updated Figure 2–6. ■ Made minor editorial changes.	—
May 2008 v1.0	Initial release.	—

3. TriMatrix Embedded Memory Blocks in Stratix IV Devices

SIV51003-3.0

This chapter describes the TriMatrix embedded memory blocks in Stratix® IV devices. TriMatrix embedded memory blocks provide three different sizes of embedded SRAM to efficiently address the needs of Stratix IV FPGA designs. TriMatrix memory includes 640-bit memory logic array blocks (MLABs), 9-Kbit M9K blocks, and 144-Kbit M144K blocks. MLABs have been optimized to implement filter delay lines, small FIFO buffers, and shift registers. You can use the M9K blocks for general purpose memory applications and the M144K blocks for processor code storage, packet buffering, and video frame buffering.

You can independently configure each embedded memory block to be a single- or dual-port RAM, FIFO buffer, ROM, or shift register using the Quartus® II MegaWizard™ Plug-In Manager. You can stitch together multiple blocks of the same type to produce larger memories with minimal timing penalty. TriMatrix memory provides up to 31,491 Kbits of embedded SRAM at up to 600 MHz operation.

This chapter contains the following sections:

- "Overview" on page 3–1
- "Memory Modes" on page 3–8
- "Clocking Modes" on page 3–17
- "Design Considerations" on page 3–18

Overview

Table 3–1 lists the features supported by the three sizes of TriMatrix memory.

Table 3–1. Summary of TriMatrix Memory Features (Part 1 of 2)

Feature	MLABs	M9K Blocks	M144K Blocks
Maximum performance	600 MHz	600 MHz	540 MHz
Total RAM bits (including parity bits)	640	9216	147,456
Configurations (depth × width)	64 × 8 64 × 9 64 × 10 32 × 16 32 × 18 32 × 20	8K × 1 4K × 2 2K × 4 1K × 8 1K × 9 512 × 16 512 × 18 256 × 32 256 × 36	16K × 8 16K × 9 8K × 16 8K × 18 4K × 32 4K × 36 2K × 64 2K × 72
Parity bits	✓	✓	✓
Byte enable	✓	✓	✓
Packed mode	—	✓	✓

Table 3–1. Summary of TriMatrix Memory Features (Part 2 of 2)

Feature	MLABs	M9K Blocks	M144K Blocks
Address clock enable	✓	✓	✓
Single-port memory	✓	✓	✓
Simple dual-port memory	✓	✓	✓
True dual-port memory	—	✓	✓
Embedded shift register	✓	✓	✓
ROM	✓	✓	✓
FIFO buffer	✓	✓	✓
Simple dual-port mixed width support	—	✓	✓
True dual-port mixed width support	—	✓	✓
Memory Initialization File (**.mif**)	✓	✓	✓
Mixed-clock mode	✓	✓	✓
Power-up condition	Outputs cleared if registered, otherwise reads memory contents	Outputs cleared	Outputs cleared
Register clears	Output registers	Output registers	Output registers
Write/Read operation triggering	Write: Falling clock edges Read: Rising clock edges	Write and Read: Rising clock edges	Write and Read: Rising clock edges
Same-port read-during-write	Outputs set to “don’t care”	Outputs set to “old data” or “new data”	Outputs set to “old data” or “new data”
Mixed-port read-during-write	Outputs set to “old data” or “don’t care”	Outputs set to “old data” or “don’t care”	Outputs set to “old data” or “don’t care”
ECC Support	Soft IP support using the Quartus II software	Soft IP support using the Quartus II software	Built-in support in ×64-wide SDP mode or soft IP support using the Quartus II software

Table 3–2 lists the capacity and distribution of the TriMatrix memory blocks in each Stratix IV family member.

Table 3–2. TriMatrix Memory Capacity and Distribution in Stratix IV Devices *(Part 1 of 2)*

Device	MLABs	M9K Blocks	M144K Blocks	Total Dedicated RAM Bits (Dedicated Memory Blocks Only) (Kb)	Total RAM Bits (Including MLABs) (Kb)
EP4SE230	4560	1235	22	14,283	17,133
EP4SE360	7072	1248	48	18,144	22,564
EP4SE530	10,624	1280	64	20,736	27,376
EP4SE820	16,261	1610	60	23,130	33,294
EP4SGX70	1452	462	16	6462	7370
EP4SGX110	2112	660	16	8244	9564
EP4SGX180	3515	950	20	11,430	13,627
EP4SGX230	4560	1235	22	14,283	17,133

Table 3–2. TriMatrix Memory Capacity and Distribution in Stratix IV Devices *(Part 2 of 2)*

Device	MLABs	M9K Blocks	M144K Blocks	Total Dedicated RAM Bits (Dedicated Memory Blocks Only) (Kb)	Total RAM Bits (Including MLABs) (Kb)
EP4SGX290	5824	936	36	13,608	17,248
EP4SGX360	7072	1248	48	18,144	22,564
EP4SGX530	10,624	1280	64	20,736	27,376
EP4S40G2	4560	1235	22	14,283	17,133
EP4S40G5	10,624	1280	64	20,736	27,376
EP4S100G2	4560	1235	22	14,283	17,133
EP4S100G3	5824	936	36	13,608	17,248
EP4S100G4	7072	1248	48	18,144	22,564
EP4S100G5	10624	1280	64	20,736	27,376

TriMatrix Memory Block Types

While the M9K and M144K memory blocks are dedicated resources, the MLABs are dual-purpose blocks. They can be configured as regular logic array blocks (LABs) or as MLABs. Ten adaptive logic modules (ALMs) make up one MLAB. Each ALM in an MLAB can be configured as either a 64 × 1 or a 32 × 2 block, resulting in a 64 × 10 or 32 × 20 simple dual-port SRAM block in a single MLAB.

Parity Bit Support

All TriMatrix memory blocks have built-in parity-bit support. The ninth bit associated with each byte can store a parity bit or serve as an additional data bit. No parity function is actually performed on the ninth bit.

Byte Enable Support

All TriMatrix memory blocks support byte enables that mask the input data so that only specific bytes of data are written. The unwritten bytes retain the previously written values. The write enable (`wren`) signals, along with the byte enable (`byteena`) signals, control the RAM blocks' write operations.

The default value for the byte enable signals is high (enabled), in which case writing is controlled only by the write enable signals. The byte enable registers have no clear port. When using parity bits on the M9K and M144K blocks, the byte enable controls all nine bits (eight bits of data plus one parity bit). When using parity bits on the MLAB, the byte-enable controls all 10 bits in the widest mode.

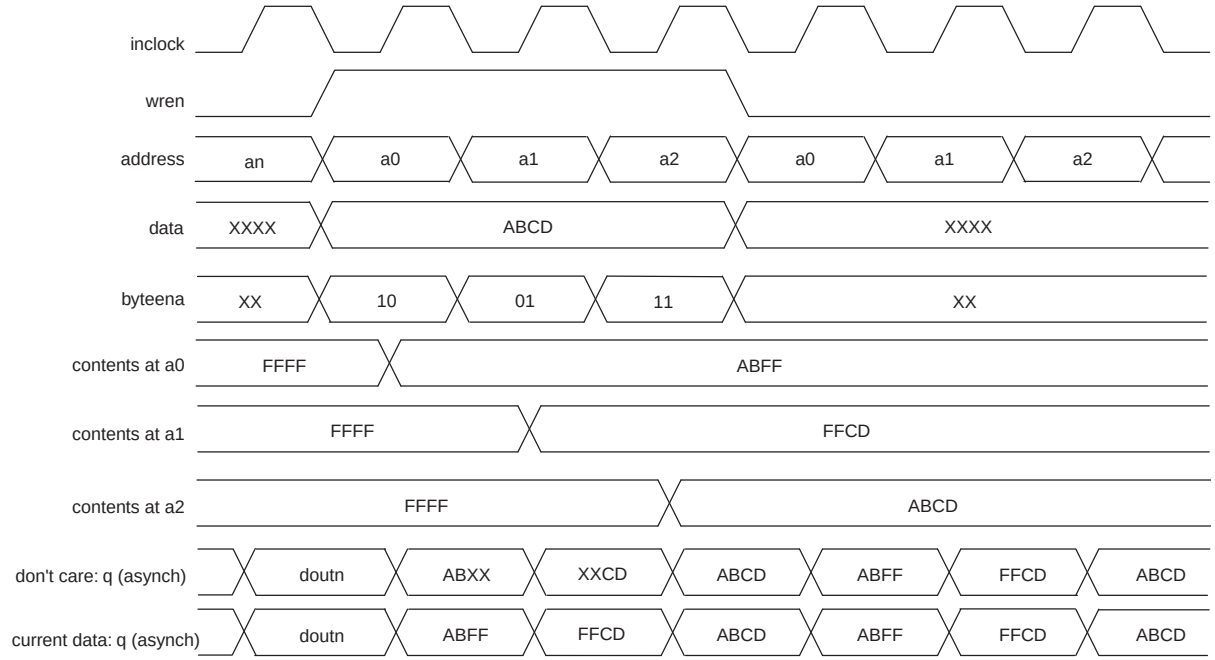
Byte enables operate in a one-hot fashion, with the LSB of the `byteena` signal corresponding to the LSB of the data bus. For example, if you use a RAM block in ×18 mode, with `byteena = 01`, `data[8..0]` is enabled, and `data[17..9]` is disabled. Similarly, if `byteena = 11`, both `data[8..0]` and `data[17..9]` are enabled. Byte enables are active high.

☞ You cannot use the byte enable feature when using the error correction coding (ECC) feature on M144K blocks.

Figure 3–1 shows how the write enable (`wren`) and byte enable (`byteena`) signals control the operations of the RAM blocks.

When a byte-enable bit is de-asserted during a write cycle, the corresponding data byte output can appear as either a "don't care" value or the current data at that location. The output value for the masked byte is controllable using the Quartus II software. When a byte-enable bit is asserted during a write cycle, the corresponding data byte output also depends on the setting chosen in the Quartus II software.

Figure 3–1. Byte Enable Functional Waveform



Packed Mode Support

Stratix IV M9K and M144K blocks support packed mode. The packed mode feature packs two independent single-port RAMs into one memory block. The Quartus II software automatically implements packed mode where appropriate by placing the physical RAM block into true dual-port mode and using the MSB of the address to distinguish between the two logical RAMs. The size of each independent single-port RAM must not exceed half of the target block size.

Address Clock Enable Support

All Stratix IV memory blocks support address clock enable, which holds the previous address value for as long as the signal is enabled (`addressstall` = `1`). When the memory blocks are configured in dual-port mode, each port has its own independent address clock enable. The default value for the address clock enable signals is low (disabled).

Figure 3–2 shows an address clock enable block diagram. The address clock enable is referred to by the port name `addressstall`.

Figure 3–2. Address Clock Enable

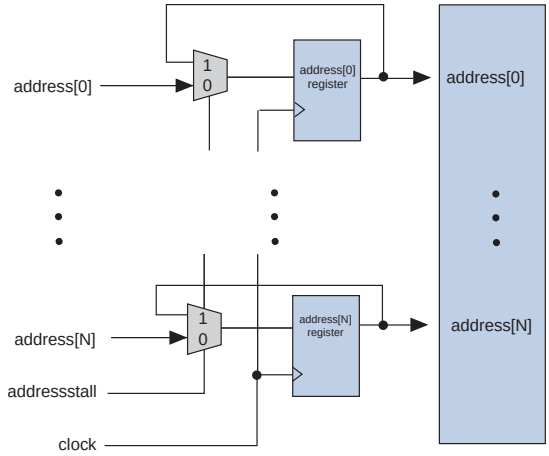


Figure 3–3 shows the address clock enable waveform during the read cycle.

Figure 3–3. Address Clock Enable During Read Cycle Waveform

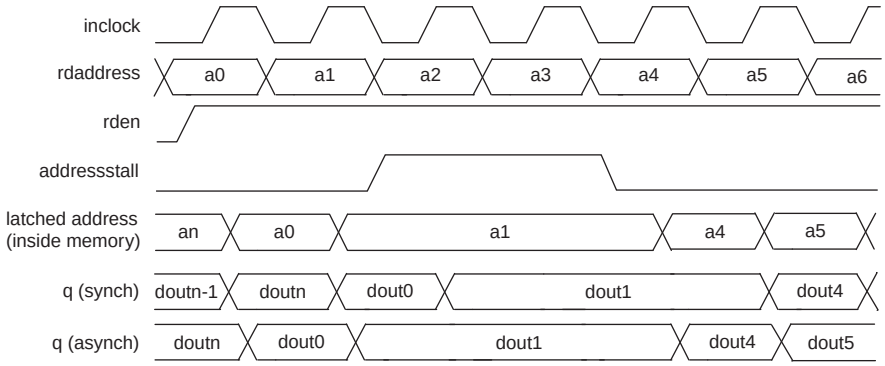
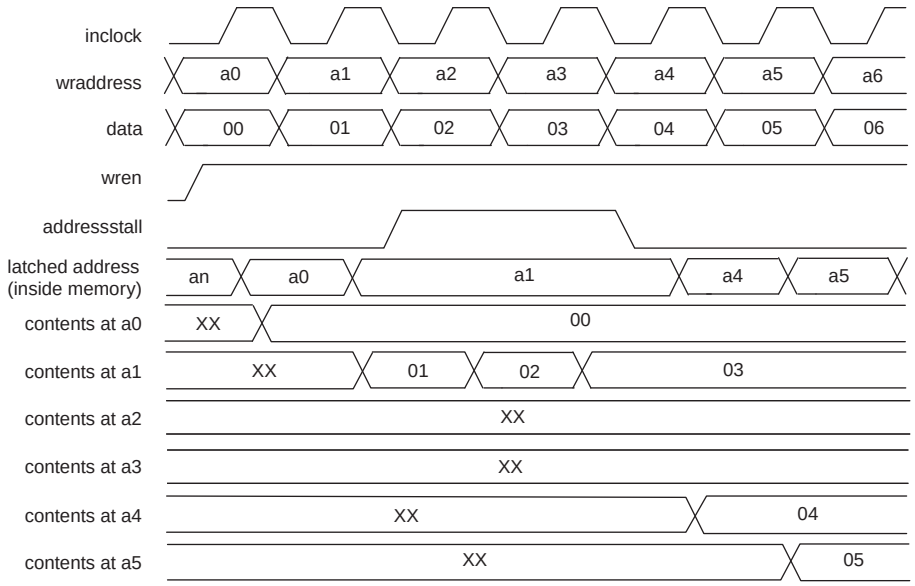


Figure 3–4 shows the address clock enable waveform during the write cycle.

Figure 3–4. Address Clock Enable During the Write Cycle Waveform



Mixed Width Support

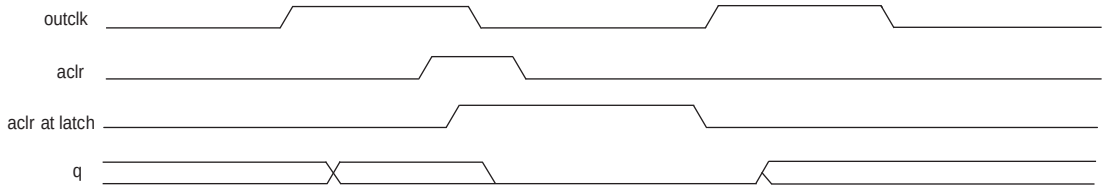
M9K and M144K memory blocks support mixed data widths inherently. MLABs can support mixed data widths through emulation using the Quartus II software. When using simple dual-port, true dual-port, or FIFO modes, mixed width support allows you to read and write different data widths to a memory block. For more information about the different widths supported per memory mode, refer to "Memory Modes" on page 3–8.

MLABs do not support mixed-width FIFO mode.

Asynchronous Clear

Stratix IV TriMatrix memory blocks support asynchronous clears on output latches and output registers. Therefore, if your RAM is not using output registers, you can still clear the RAM outputs using the output latch asynchronous clear. Figure 3–5 shows a waveform of the output latch asynchronous clear function.

Figure 3–5. Output Latch Asynchronous Clear Waveform



You can selectively enable asynchronous clears per logical memory using the Quartus II RAM MegaWizard Plug-In Manager.

For more information, refer to the *RAM Megafunction User Guide*.

Error Correction Code (ECC) Support

Stratix IV M144K blocks have built-in support for error correction code (ECC) when in ×64-wide simple dual-port mode. ECC allows you to detect and correct data errors in the memory array. The M144K blocks have a single-error-correction double-error-detection (SECDED) implementation. SECDED can detect and fix a single bit error in a 64-bit word, or detect two bit errors in a 64-bit word. It cannot detect three or more errors.

The M144K ECC status is communicated using a three-bit status flag `eccstatus[2..0]`. The status flag can be either registered or unregistered. When registered, it uses the same clock and asynchronous clear signals as the output registers. When unregistered, it cannot be asynchronously cleared.

Table 3–3 lists the truth table for the ECC status flags.

Table 3–3. Truth Table for ECC Status Flags

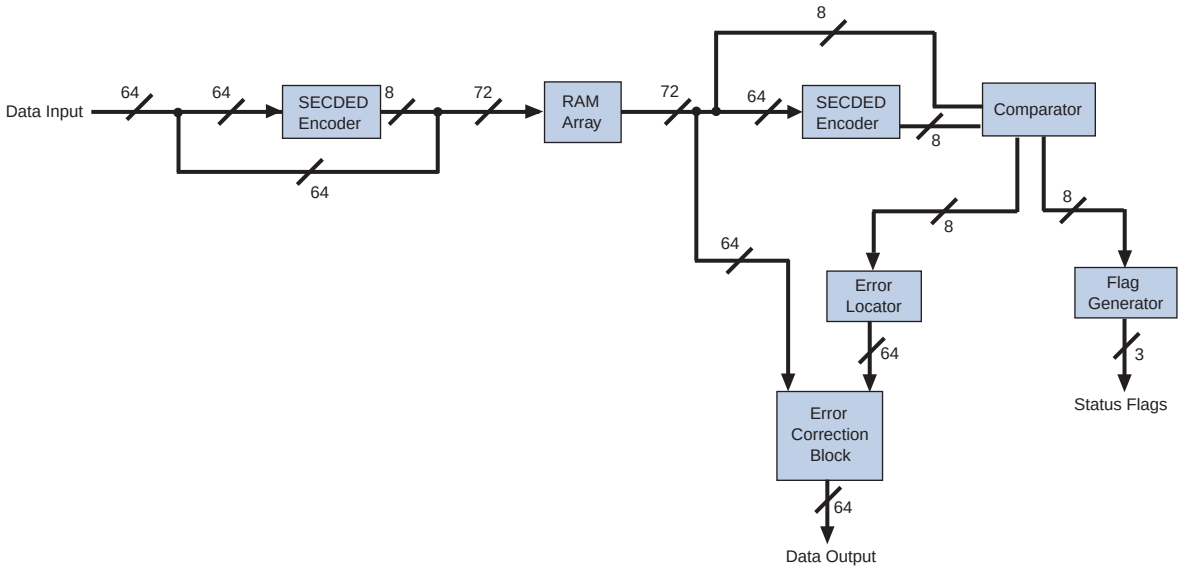
Status	eccstatus[2]	eccstatus[1]	eccstatus[0]
No error	0	0	0
Single error and fixed	0	1	1
Double error and no fix	1	0	1
Illegal	0	0	1
Illegal	0	1	0
Illegal	1	0	0
Illegal	1	1	X

You cannot use the byte enable feature when ECC is engaged.

Read during write “old data mode” is not supported when ECC is engaged.

Figure 3–6 shows a diagram of the ECC block of the M144K block.

Figure 3–6. ECC Block Diagram of the M144K Block



Memory Modes

Stratix IV TriMatrix memory blocks allow you to implement fully synchronous SRAM memory in multiple modes of operation. M9K and M144K blocks do not support asynchronous memory (unregistered inputs). MLABs support asynchronous (flow-through) read operations.

Depending on which TriMatrix memory block you target, you can use the following modes:

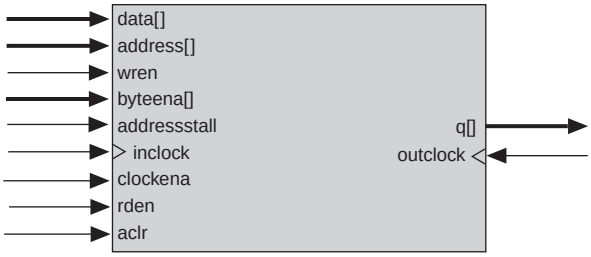
- Single-port
- Simple dual-port
- True dual-port
- Shift-register
- ROM
- FIFO

When using the memory blocks in ROM, single-port, simple dual-port, or true dual-port mode, you can corrupt the memory contents if you violate the setup or hold-time on any of the memory block input registers. This applies to both read and write operations.

Single-Port RAM

All TriMatrix memory blocks support single-port mode. Single-port mode allows you to do either one-read or one-write operation at a time. Simultaneous reads and writes are not supported in single-port mode. Figure 3–7 shows the single-port RAM configuration.

Figure 3–7. Single-Port RAM *(Note 1)*



Note to Figure 3–7:

(1) You can implement two single-port memory blocks in a single M9K or M144K block. For more information, refer to "Packed Mode Support" on page 3–5.

During a write operation, RAM output behavior is configurable. If you use the read-enable signal and perform a write operation with the read enable de-activated, the RAM outputs retain the values they held during the most recent active read enable. If you activate read enable during a write operation, or if you are not using the read-enable signal at all, the RAM outputs either show the "new data" being written, the "old data" at that address, or a "don't care" value. To choose the desired behavior, set the read-during-write behavior to either **new data**, **old data**, or **don't care** in the RAM MegaWizard Plug-In Manager in the Quartus II software. For more information, refer to "Read-During-Write Behavior" on page 3–19.

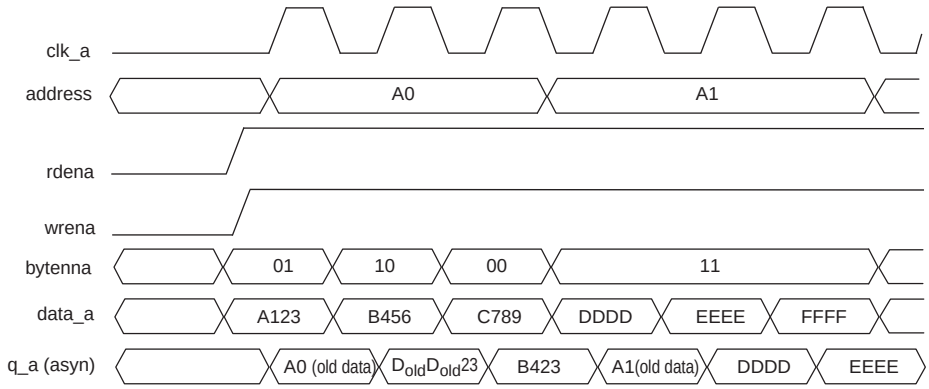
Table 3–4 lists the possible port width configurations for TriMatrix memory blocks in single-port mode.

Table 3–4. Port Width Configurations for MLABs, M9K, and M144K Blocks (Single-Port Mode)

	MLABs	M9K Blocks	M144K Blocks
Port Width Configurations	64 × 8 64 × 9 64 × 10 32 × 16 32 × 18 32 × 20	8K × 1 4K × 2 2K × 4 1K × 8 1K × 9 512 × 16 512 × 18 256 × 32 256 × 36	16K × 8 16K × 9 8K × 16 8K × 18 4K × 32 4K × 36 2K × 64 2K × 72

Figure 3–8 shows timing waveforms for read and write operations in single-port mode with unregistered outputs. Registering the RAM's outputs simply delays the q output by one clock cycle.

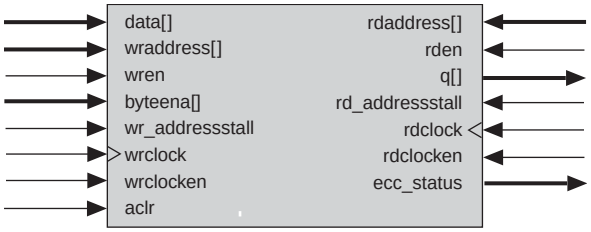
Figure 3–8. Timing Waveform for Read-Write Operations (Single-Port Mode)



Simple Dual-Port Mode

All TriMatrix memory blocks support simple dual-port mode. Simple dual-port mode allows you to perform one-read and one-write operation to different locations at the same time. Write operation happens on port A; read operation happens on port B. Figure 3–9 shows a simple dual-port configuration.

Figure 3–9. Stratix IV Simple Dual-Port Memory *(Note 1)*



Note to Figure 3–9:

(1) Simple dual-port RAM supports input/output clock mode in addition to read/write clock mode.

Simple dual-port mode supports different read and write data widths (mixed-width support). Table 3–5 lists the mixed width configurations for M9K blocks in simple dual-port mode. MLABs do not have native support for mixed-width operation. The Quartus II software implements mixed-width memories in MLABs by using more than one MLAB.

Table 3–5. M9K Block Mixed-Width Configurations (Simple Dual-Port Mode) (Part 1 of 2)

Read Port	Write Port								
	8K × 1	4K × 2	2K × 4	1K × 8	512 × 16	256 × 32	1K × 9	512 × 18	256 × 36
8K × 1	✓	✓	✓	✓	✓	✓	—	—	—
4K × 2	✓	✓	✓	✓	✓	✓	—	—	—
2K × 4	✓	✓	✓	✓	✓	✓	—	—	—

Table 3–5. M9K Block Mixed-Width Configurations (Simple Dual-Port Mode) (Part 2 of 2)

Read Port	Write Port								
	8K × 1	4K × 2	2K × 4	1K × 8	512 × 16	256 × 32	1K × 9	512 × 18	256 × 36
1K × 8	✓	✓	✓	✓	✓	✓	—	—	—
512 × 16	✓	✓	✓	✓	✓	✓	—	—	—
256 × 32	✓	✓	✓	✓	✓	✓	—	—	—
1K × 9	—	—	—	—	—	—	✓	✓	✓
512 × 18	—	—	—	—	—	—	✓	✓	✓
256 × 36	—	—	—	—	—	—	✓	✓	✓

Table 3–6 lists the mixed-width configurations for M144K blocks in simple dual-port mode.

Table 3–6. M144K Block Mixed-Width Configurations (Simple Dual-Port Mode)

Read Port	Write Port							
	16K × 8	8K × 16	4K × 32	2K × 64	16K × 9	8K × 18	4K × 36	2K × 72
16K × 8	✓	✓	✓	✓	—	—	—	—
8K × 16	✓	✓	✓	✓	—	—	—	—
4K × 32	✓	✓	✓	✓	—	—	—	—
2K × 64	✓	✓	✓	✓	—	—	—	—
16K × 9	—	—	—	—	✓	✓	✓	✓
8K × 18	—	—	—	—	✓	✓	✓	✓
4K × 36	—	—	—	—	✓	✓	✓	✓
2K × 72	—	—	—	—	✓	✓	✓	✓

In simple dual-port mode, M9K and M144K blocks support separate write-enable and read-enable signals. You can save power by keeping the read-enable signal low (inactive) when not reading. Read-during-write operations to the same address can either output a "don't care" value or "old data" value. To choose the desired behavior, set the read-during-write behavior to either **don't care** or **old data** in the RAM MegaWizard Plug-In Manager in the Quartus II software. For more information, refer to "Read-During-Write Behavior" on page 3–19.

MLABs only support a write-enable signal. Read-during-write behavior for the MLABs can be either "don't care" or "old data". The available choices depend on the configuration of the MLAB.

Figure 3–10 shows timing waveforms for read and write operations in simple dual-port mode with unregistered outputs. Registering the RAM outputs simply delays the q output by one clock cycle.

Figure 3–10. Simple Dual-Port Timing Waveforms

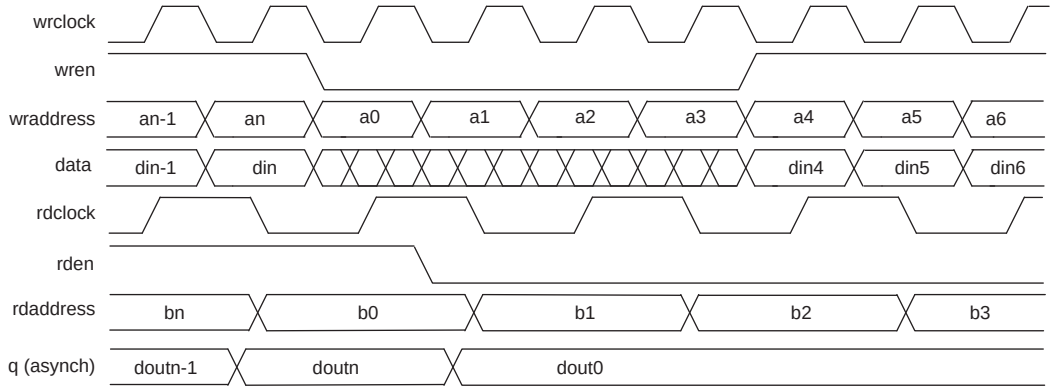
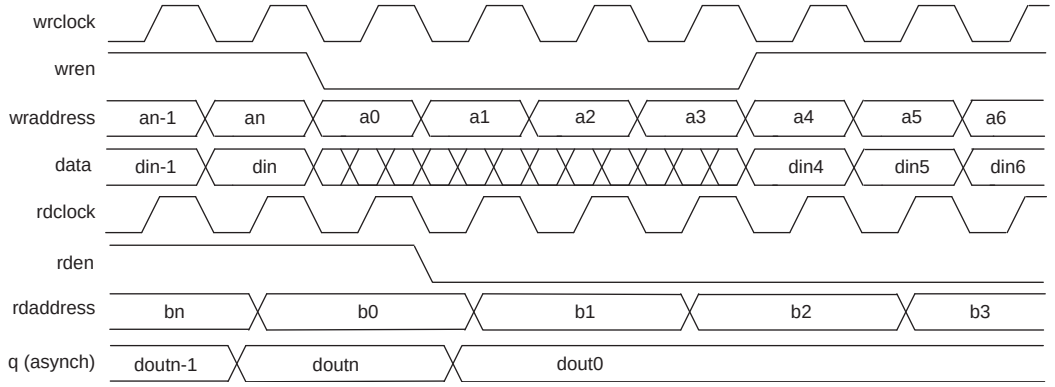


Figure 3–11 shows timing waveforms for read and write operations in mixed-port mode with unregistered outputs.

Figure 3–11. Mixed-Port Read-During-Write Timing Waveforms

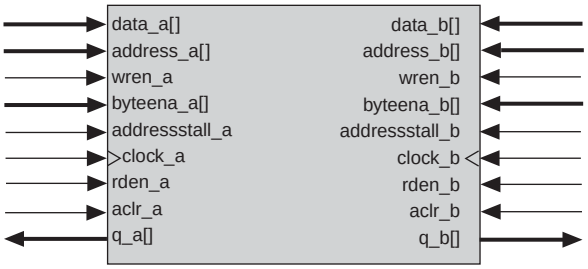


True Dual-Port Mode

Stratix IV M9K and M144K blocks support true dual-port mode. Sometimes called bi-directional dual-port, this mode allows you to perform any combination of two port operations: two reads, two writes, or one read and one write at two different clock frequencies.

Figure 3–12 shows the true dual-port RAM configuration.

Figure 3–12. Stratix IV True Dual-Port Memory *(Note 1)*



Note to Figure 3–12:

(1) True dual-port memory supports input/output clock mode in addition to independent clock mode.

The widest bit configuration of the M9K and M144K blocks in true dual-port mode is as follows:

- 512 × 16-bit (×18-bit with parity) (M9K)
- 4K × 32-bit (×36-bit with parity) (M144K)

Wider configurations are unavailable because the number of output drivers is equivalent to the maximum bit width of the respective memory block. Because true dual-port RAM has outputs on two ports, its maximum width equals half of the total number of output drivers. Table 3–7 lists the possible M9K block mixed-port width configurations in true dual-port mode.

Table 3–7. M9K Block Mixed-Width Configuration (True Dual-Port Mode)

Read Port	Write Port						
	8K × 1	4K × 2	2K × 4	1K × 8	512 × 16	1K × 9	512 × 18
8K × 1	✓	✓	✓	✓	✓	—	—
4K × 2	✓	✓	✓	✓	✓	—	—
2K × 4	✓	✓	✓	✓	✓	—	—
1K × 8	✓	✓	✓	✓	✓	—	—
512 × 16	✓	✓	✓	✓	✓	—	—
1K × 9	—	—	—	—	—	✓	✓
512 × 18	—	—	—	—	—	✓	✓

Table 3–8 lists the possible M144K block mixed-port width configurations in true dual-port mode.

Table 3–8. M144K Block Mixed-Width Configurations (True Dual-Port Mode) (Part 1 of 2)

Read Port	Write Port					
	16K × 8	8K × 16	4K × 32	16K × 9	8K × 18	4K × 36
16K × 8	✓	✓	✓	—	—	—
8K × 16	✓	✓	✓	—	—	—
4K × 32	✓	✓	✓	—	—	—

Table 3–8. M144K Block Mixed-Width Configurations (True Dual-Port Mode) (Part 2 of 2)

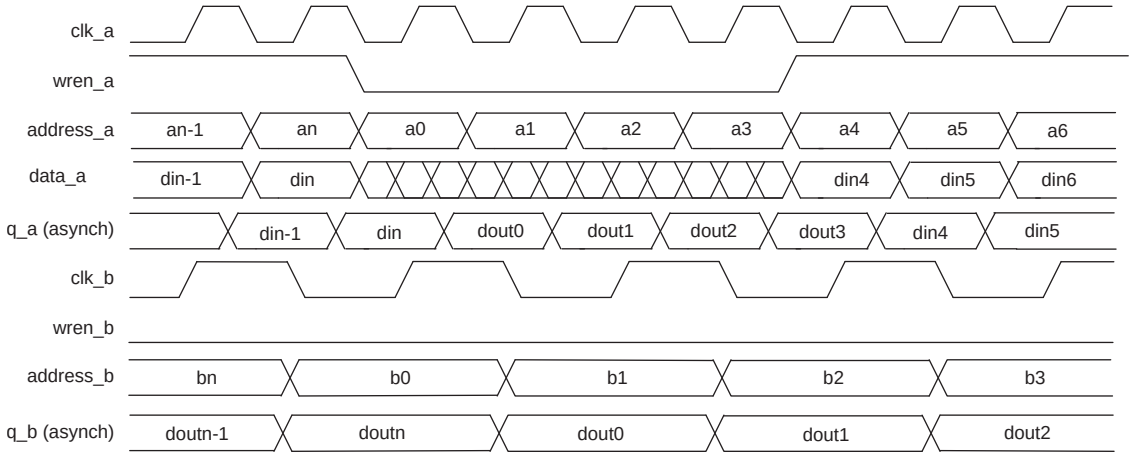
Read Port	Write Port					
	16K × 8	8K × 16	4K × 32	16K × 9	8K × 18	4K × 36
16K × 9	—	—	—	✓	✓	✓
8K × 18	—	—	—	✓	✓	✓
4K × 36	—	—	—	✓	✓	✓

In true dual-port mode, M9K and M144K blocks support separate write-enable and read-enable signals. You can save power by keeping the read-enable signal low (inactive) when not reading. Read-during-write operations to the same address can either output "new data" at that location or "old data". To choose the desired behavior, set the read-during-write behavior to either **new data** or **old data** in the RAM MegaWizard Plug-In Manager in the Quartus II software. For more information, refer to "Read-During-Write Behavior" on page 3–19.

In true dual-port mode, you can access any memory location at any time from either port. When accessing the same memory location from both ports, you must avoid possible write conflicts. A write conflict happens when you attempt to write to the same address location from both ports at the same time. This results in unknown data being stored to that address location. No conflict resolution circuitry is built into the Stratix IV TriMatrix memory blocks. You must handle address conflicts external to the RAM block.

Figure 3–13 shows true dual-port timing waveforms for the write operation at port A and read operation at port B, with the **Read-During-Write** behavior set to **new data**. Registering the RAM's outputs simply delays the q outputs by one clock cycle.

Figure 3–13. True Dual-Port Timing Waveform



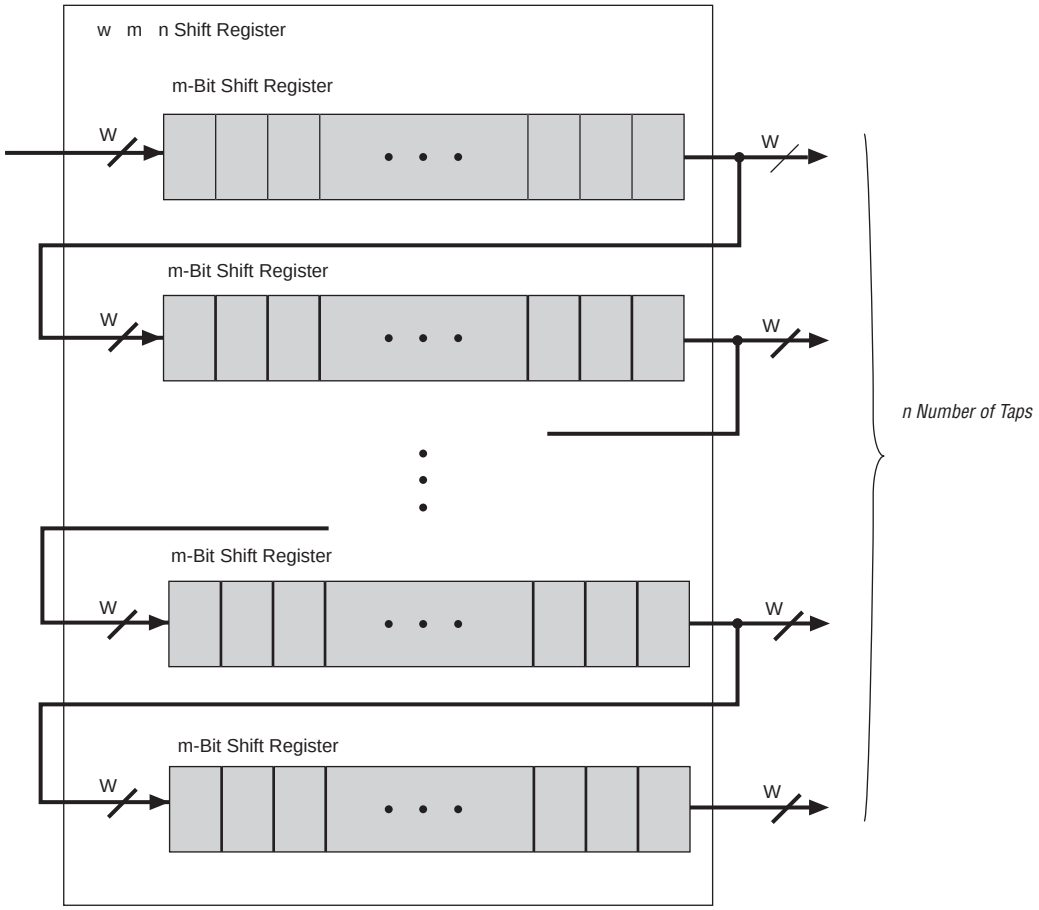
Shift-Register Mode

All Stratix IV memory blocks support shift register mode. Embedded memory block configurations can implement shift registers for digital signal processing (DSP) applications, such as finite impulse response (FIR) filters, pseudo-random number generators, multi-channel filtering, and auto- and cross-correlation functions. These and other DSP applications require local data storage, traditionally implemented with standard flipflops that quickly exhaust many logic cells for large shift registers. A more efficient alternative is to use embedded memory as a shift-register block, which saves logic cell and routing resources.

The size of a shift register ($w \times m \times n$) is determined by the input data width (w), the length of the taps (m), and the number of taps (n). You can cascade memory blocks to implement larger shift registers.

Figure 3–14 shows the TriMatrix memory block in shift-register mode.

Figure 3–14. Shift-Register Memory Configuration



ROM Mode

All Stratix IV TriMatrix memory blocks support ROM mode. A **.mif** file initializes the ROM contents of these blocks. The address lines of the ROM are registered on M9K and M144K blocks, but can be unregistered on MLABs. The outputs can be registered or unregistered. Output registers can be asynchronously cleared. The ROM read operation is identical to the read operation in the single-port RAM configuration.

FIFO Mode

All TriMatrix memory blocks support FIFO mode. MLABs are ideal for designs with many small, shallow FIFO buffers. To implement FIFO buffers in your design, use the Quartus II software FIFO MegaWizard Plug-In Manager. Both single- and dual-clock (asynchronous) FIFO buffers are supported.

For more information about implementing FIFO buffers, refer to the *Single- and Dual-Clock FIFO Megafunctions User Guide*.

MLABs do not support mixed-width FIFO mode.

Clocking Modes

Stratix IV TriMatrix memory blocks support the following clocking modes:

- Independent
- Input/output
- Read/write
- Single clock

Violating the setup or hold time on the memory block address registers could corrupt memory contents. This applies to both read and write operations.

Table 3–9 lists the clocking mode versus memory mode support matrix.

Table 3–9. TriMatrix Memory Clock Modes

Clocking Mode	True Dual-Port Mode	Simple Dual-Port Mode	Single-Port Mode	ROM Mode	FIFO Mode
Independent	✓	—	—	✓	—
Input/output	✓	✓	✓	✓	—
Read/write	—	✓	—	—	✓
Single clock	✓	✓	✓	✓	✓

Independent Clock Mode

Stratix IV TriMatrix memory blocks can implement independent clock mode for true dual-port memories. In this mode, a separate clock is available for each port (clock A and clock B). Clock A controls all registers on the port A side; clock B controls all registers on the port B side. Each port also supports independent clock enables for both port A and port B registers, respectively. Asynchronous clears are supported only for output latches and output registers on both ports.

Input/Output Clock Mode

Stratix IV TriMatrix memory blocks can implement input/output clock mode for true dual-port and simple dual-port memories. In this mode, an input clock controls all registers related to the data input to the memory block including data, address, byte enables, read enables, and write enables. An output clock controls the data output registers. Asynchronous clears are available on output latches and output registers only.

Read/Write Clock Mode

Stratix IV TriMatrix memory blocks can implement read/write clock mode for simple dual-port memories. In this mode, a write clock controls the data-input, write-address, and write-enable registers. Similarly, a read clock controls the data-output, read-address, and read-enable registers. The memory blocks support independent clock enables for both the read and write clocks. Asynchronous clears are available on data output latches and registers only.

When using read/write clock mode, if you perform a simultaneous read/write to the same address location, the output read data is unknown. If you require the output data to be a known value, use either single-clock mode or input/output clock mode and choose the appropriate read-during-write behavior in the Megawizard Plug-In Manager.

Single Clock Mode

Stratix IV TriMatrix memory blocks can implement single-clock mode for true dual-port, simple dual-port, and single-port memories. In this mode, a single clock, together with a clock enable, is used to control all registers of the memory block. Asynchronous clears are available on output latches and output registers only.

Design Considerations

This section describes guidelines for designing with TriMatrix memory blocks.

Selecting TriMatrix Memory Blocks

The Quartus II software automatically partitions user-defined memory into embedded memory blocks by taking into account both speed and size constraints placed on your design. For example, the Quartus II software may spread memory out across multiple memory blocks when resources are available to increase the performance of the design. You can manually assign memory to a specific block size using the RAM MegaWizard Plug-In Manager.

MLABs can implement single-port SRAM through emulation using the Quartus II software. Emulation results in minimal additional logic resources being used. Because of the dual-purpose architecture of the MLAB, it only has data input registers and output registers in the block. MLABs gain input address registers and additional data output registers from ALMs.

For more information about register packing, refer to the *Logic Array Blocks and Adaptive Logic Modules in Stratix IV Devices* chapter.

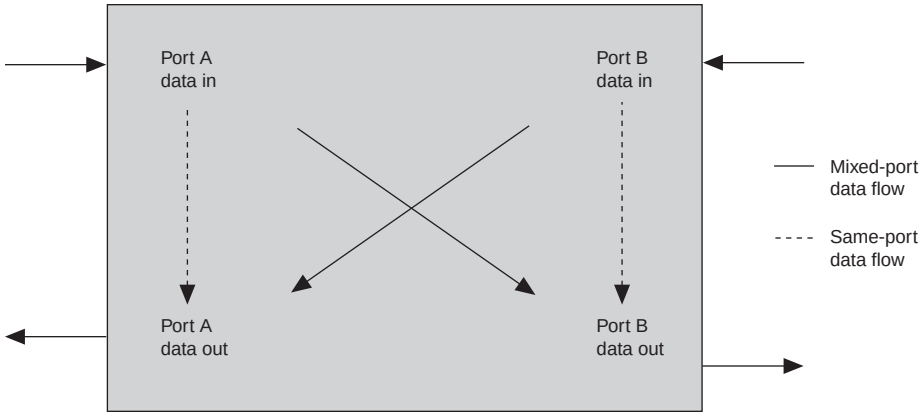
Conflict Resolution

When using memory blocks in true dual-port mode, it is possible to attempt two write operations to the same memory location (address). Because no conflict resolution circuitry is built into the memory blocks, this results in unknown data being written to that location. Therefore, you must implement conflict resolution logic external to the memory block to avoid address conflicts.

Read-During-Write Behavior

You can customize the read-during-write behavior of the Stratix IV TriMatrix memory blocks to suit your design needs. Two types of read-during-write operations are available: same port and mixed port. Figure 3–15 shows the difference between the two types.

Figure 3–15. Stratix IV Read-During-Write Data Flow



Same-Port Read-During-Write Mode

This mode applies to either a single-port RAM or the same port of a true dual-port RAM. In same-port read-during-write mode, three output choices are available: new data mode (or flow-through), old data mode, or don't care mode. In new data mode, the new data is available on the rising edge of the same clock cycle on which it was written. In old data mode, the RAM outputs reflect the old data at that address before the write operation proceeds. In don't care mode, the RAM outputs don't care values for a read-during-write operation.

Figure 3–16 shows sample functional waveforms of same-port read-during-write behavior in new data mode.

Figure 3–16. Same Port Read-During Write: New Data Mode

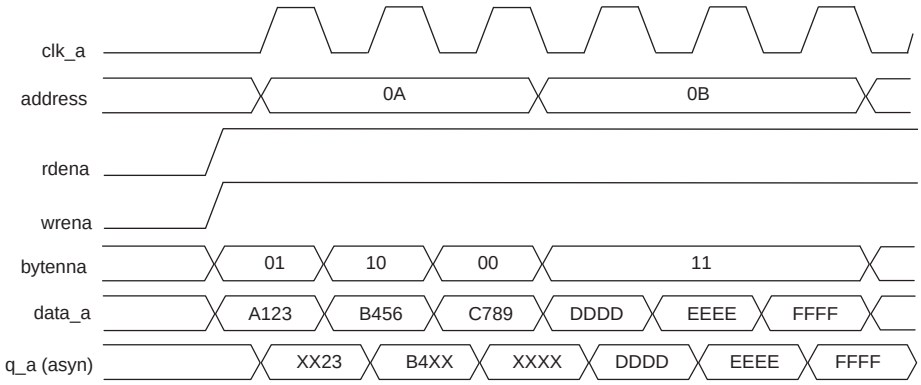
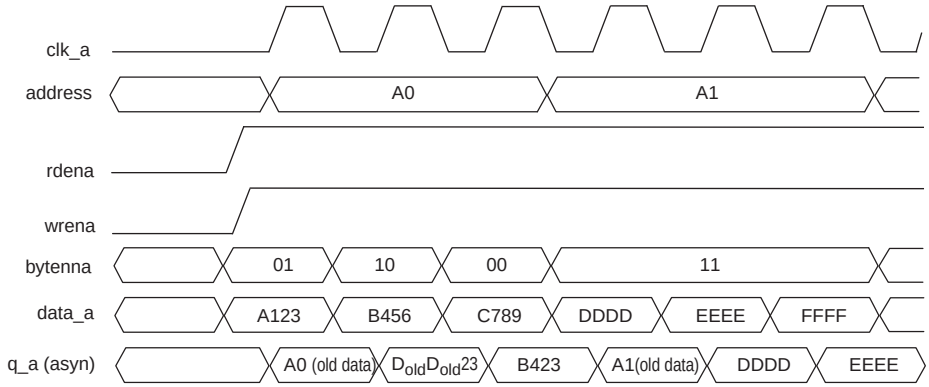


Figure 3–17 shows sample functional waveforms of same-port read-during-write behavior in old data mode.

Figure 3–17. Same Port Read-During-Write: Old Data Mode



Mixed-Port Read-During-Write Mode

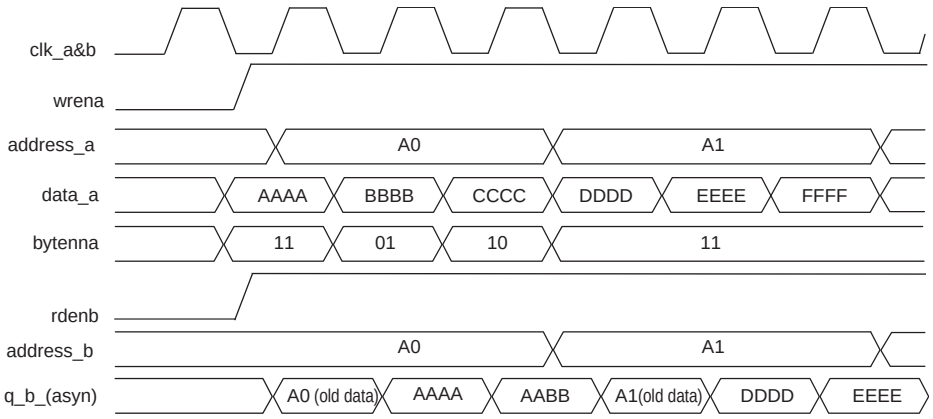
This mode applies to a RAM in simple or true dual-port mode that has one port reading from and the other port writing to the same address location with the same clock.

In this mode, you also have two output choices: "old data" or "don't care". In old data mode, a read-during-write operation to different ports causes the RAM outputs to reflect the "old data" at that address location. In don't care mode, the same operation results in a "don't care" or "unknown" value on the RAM outputs.

Read-during-write behavior is controlled using the RAM MegaWizard Plug-In Manager. For more information, refer to the *RAM Megafunction User Guide*.

Figure 3–18 shows a sample functional waveform of mixed-port read-during-write behavior for old data mode. In don't care mode, "old data" is simply replaced with "don't cares".

Figure 3–18. Mixed Port Read-During-Write: Old Data Mode



Mixed-port read-during-write is not supported when two different clocks are used in a dual-port RAM. The output value is unknown during a dual-clock mixed-port read-during-write operation.

Power-Up Conditions and Memory Initialization

M9K and M144K memory block outputs power up to zero (cleared), regardless of whether the output registers are used or bypassed. MLABs power up to zero if output registers are used and power up reading the memory contents if output registers are not used. You must take this into consideration when designing logic that might evaluate the initial power-up values of the MLAB memory block. For Stratix IV devices, the Quartus II software initializes the RAM cells to zero unless there is a **.mif** file specified.

All memory blocks support initialization using a **.mif** file. You can create **.mif** files in the Quartus II software and specify their use with the RAM MegaWizard Plug-In Manager when instantiating a memory in your design. Even if a memory is pre-initialized (for example, using a **.mif** file), it still powers up with its outputs cleared.

For more information about **.mif** files, refer to the *RAM Megafunction User Guide* and the *Quartus II Handbook*.

Power Management

Stratix IV memory block clock-enables allow you to control clocking of each memory block to reduce AC power consumption. Use the read-enable signal to ensure that read operations only occur when you need them to. If your design does not need read-during-write, you can reduce your power consumption by de-asserting the read-enable signal during write operations, or any period when no memory operations occur.

The Quartus II software automatically places any unused memory blocks in low-power mode to reduce static power.

Document Revision History

Table 3–10 lists the revision history for this chapter.

Table 3–10. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
November 2009 v3.0	■ Updated Table 3–2. ■ Updated the "Simple Dual-Port Mode" section. ■ Minor text edits. ■ Updated graphics.	—
June 2009 v2.3	■ Updated Table 3–1 and Figure 3–2. ■ Updated the "Introduction", "Byte Enable Support", "Mixed Width Support", "Asynchronous Clear", "Single-Port RAM", "Simple Dual-Port Mode", "True Dual-Port Mode", "FIFO Mode", and "Read/Write Clock Mode" sections. ■ Added introductory sentences to improve search ability. ■ Removed the Conclusion section. ■ Minor text edits.	—
April 2009 v2.2	■ Updated Table 3–2.	—
March 2009 v2.1	■ Updated Table 3–2. ■ Removed "Referenced Documents" section.	—
November 2008 v2.0	Updated "Power-Up Conditions and Memory Initialization" on page 3–20	—
May 2008 v1.0	Initial Release.	—

4. DSP Blocks in Stratix IV Devices

SIV51004-3.0

This chapter describes how the Stratix® IV device digital signal processing (DSP) blocks are optimized to support DSP applications requiring high data throughput, such as finite impulse response (FIR) filters, infinite impulse response (IIR) filters, fast Fourier transform (FFT) functions, and encoders. You can configure the DSP blocks to implement one of several operational modes to suit your application. The built-in shift register chain, multipliers, and adders/subtractors minimize the amount of external logic to implement these functions, resulting in efficient resource utilization and improved performance and data throughput for DSP applications.

Many complex systems, such as WiMAX, 3GPP WCDMA, high-performance computing (HPC), voice over Internet protocol (VoIP), H.264 video compression, medical imaging, and HDTV use sophisticated digital signal processing techniques, which typically require a large number of mathematical computations. Stratix IV devices are ideally suited for these tasks because the DSP blocks consist of a combination of dedicated elements that perform multiplication, addition, subtraction, accumulation, summation, and dynamic shift operations.

Along with the high-performance Stratix IV soft logic fabric and TriMatrix memory structures, you can configure DSP blocks to build sophisticated fixed-point and floating-point arithmetic functions. These can be manipulated easily to implement common, larger computationally intensive subsystems such as FIR filters, complex FIR filters, IIR filters, FFT functions, and discrete cosine transform (DCT) functions.

This chapter contains the following sections:

- "Stratix IV DSP Block Overview"
- "Stratix IV Simplified DSP Operation" on page 4–3
- "Stratix IV Operational Modes Overview" on page 4–8
- "Stratix IV DSP Block Resource Descriptions" on page 4–9
- "Stratix IV Operational Mode Descriptions" on page 4–15
- "Software Support" on page 4–34

Stratix IV DSP Block Overview

Each Stratix IV device has two to seven columns of DSP blocks that implement multiplication, multiply-add, multiply-accumulate (MAC), and dynamic shift functions efficiently. Architectural highlights of the Stratix IV DSP block include:

- High-performance, power optimized, fully registered, and pipelined multiplication operations
- Natively supported 9-bit, 12-bit, 18-bit, and 36-bit wordlengths
- Natively supported 18-bit complex multiplications
- Efficiently supported floating-point arithmetic formats (24-bit for single precision and 53-bit for double precision)
- Signed and unsigned input support
- Built-in addition, subtraction, and accumulation units to combine multiplication results efficiently
- Cascading 18-bit input bus to form tap-delay line for filtering applications
- Cascading 44-bit output bus to propagate output results from one block to the next block without external logic support
- Rich and flexible arithmetic rounding and saturation units
- Efficient barrel shifter support
- Loopback capability to support adaptive filtering

Table 4–1 shows the number of DSP blocks for the Stratix IV device family.

Table 4–1. Number of DSP Blocks in Stratix IV Devices (Part 1 of 2)

Family	Device	DSP Blocks	Independent Input and Output Multiplication Operators					High-Precision Multiplier Adder Mode	Four Multiplier Adder Mode
			9 × 9 Multipliers	12 × 12 Multipliers	18 × 18 Multipliers	18 × 18 Complex	36 × 36 Multipliers	18 × 36 Multipliers	18 × 18 Multipliers
Stratix IV E	EP4SE230	161	1288	966	644	322	322	644	1288
	EP4SE360	130	1040	780	520	260	260	520	1040
	EP4SE530	128	1024	768	512	256	256	512	1024
	EP4SE820	120	960	720	480	240	240	480	960
Stratix IV GX	EP4SGX70	48	384	288	192	96	96	192	384
	EP4SGX110	64	512	384	256	128	128	256	512
	EP4SGX180	115	920	690	460	230	230	460	920
	EP4SGX230	161	1288	966	644	322	322	644	1288
	EP4SGX290	104	832	624	416	208	208	416	832
	EP4SGX360 *(1)*	130	1040	780	520	260	260	520	1040
	EP4SGX360 *(2)*	128	1024	768	512	256	256	512	1024
	EP4SGX530	128	1024	768	512	256	256	512	1024

Table 4–1. Number of DSP Blocks in Stratix IV Devices (Part 2 of 2)

Family	Device	DSP Blocks	Independent Input and Output Multiplication Operators					High-Precision Multiplier Adder Mode	Four Multiplier Adder Mode
			9 × 9 Multipliers	12 × 12 Multipliers	18 × 18 Multipliers	18 × 18 Complex	36 × 36 Multipliers	18 × 36 Multipliers	18 × 18 Multipliers
Stratix IV GT	EP4S40G2	161	1288	966	644	322	322	644	1288
	EP4S40G5	128	1024	768	512	256	256	512	1024
	EP4S100G2	161	1288	966	644	322	322	644	1288
	EP4S100G3	104	832	624	416	208	208	416	832
	EP4S100G4	128	1024	768	512	256	256	512	1024
	EP4S100G5	128	1024	768	512	256	256	512	1024

Notes to Table 4–1:

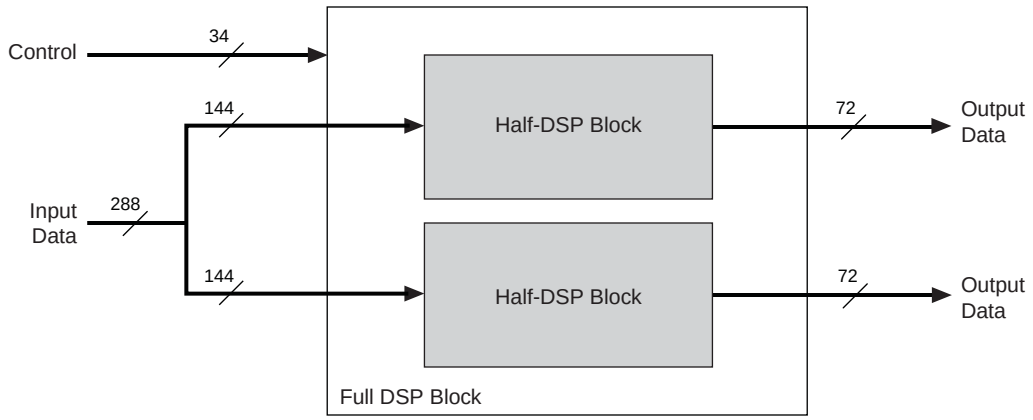
(1) This is applicable for all packages in EP4SGX360 except F1932.

(2) This is applicable for EP4SGX360F1932 only.

Table 4–1 shows that the largest Stratix IV DSP-centric device provides up to 1288 18 × 18 multiplier functionality in the 36 × 36, complex 18 × 18, and summation modes.

Each DSP block occupies four LABs in height and can be divided further into two half blocks that share some common clock signals, but are for all common purposes identical in functionality. Figure 4–1 shows the layout of each DSP block.

Figure 4–1. Overview of DSP Block Signals



Stratix IV Simplified DSP Operation

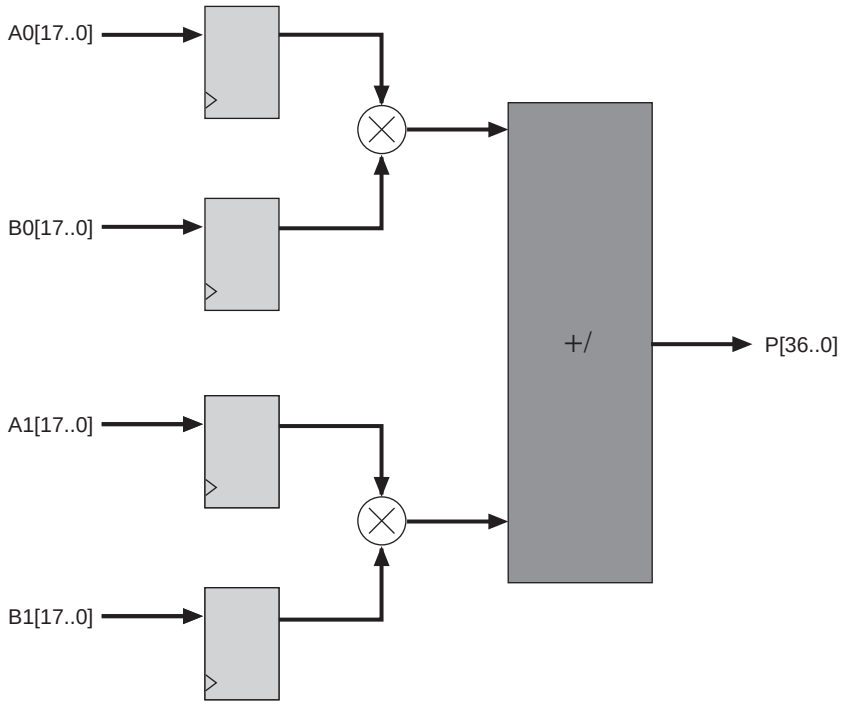
In Stratix IV devices, the fundamental building block is a pair of 18 × 18-bit multipliers followed by a first-stage 37-bit addition/subtraction unit, as shown in Equation 4–1 and Figure 4–2.

All signed numbers, input, and output data are represented in 2's-complement format only.

Equation 4–1. Multiplier Equation

$$P[36..0] = A_0[17..0] \times B_0[17..0] \pm A_1[17..0] \times B_1[17..0]$$

Figure 4–2. Basic Two-Multiplier Adder Building Block



The structure shown in Figure 4–2 is useful for building more complex structures, such as complex multipliers and 36 × 36 multipliers, as described in later sections.

Each Stratix IV DSP block contains four two-multiplier adder units (2 two-multiplier adder units per half block). Therefore, there are eight 18 × 18 multiplier functionalities per DSP block.

Following the two-multiplier adder units are the pipeline registers, the second-stage adders, and an output register stage. You can configure the second-stage adders to provide the alternative functions per half block, as shown in Equation 4–2 and Equation 4–3.

Equation 4–2. Four-Multiplier Adder Equation

$$Z[37..0] = P_0[36..0] + P_1[36..0]$$

Equation 4–3. Four-Multiplier Adder Equation (44-Bit Accumulation)

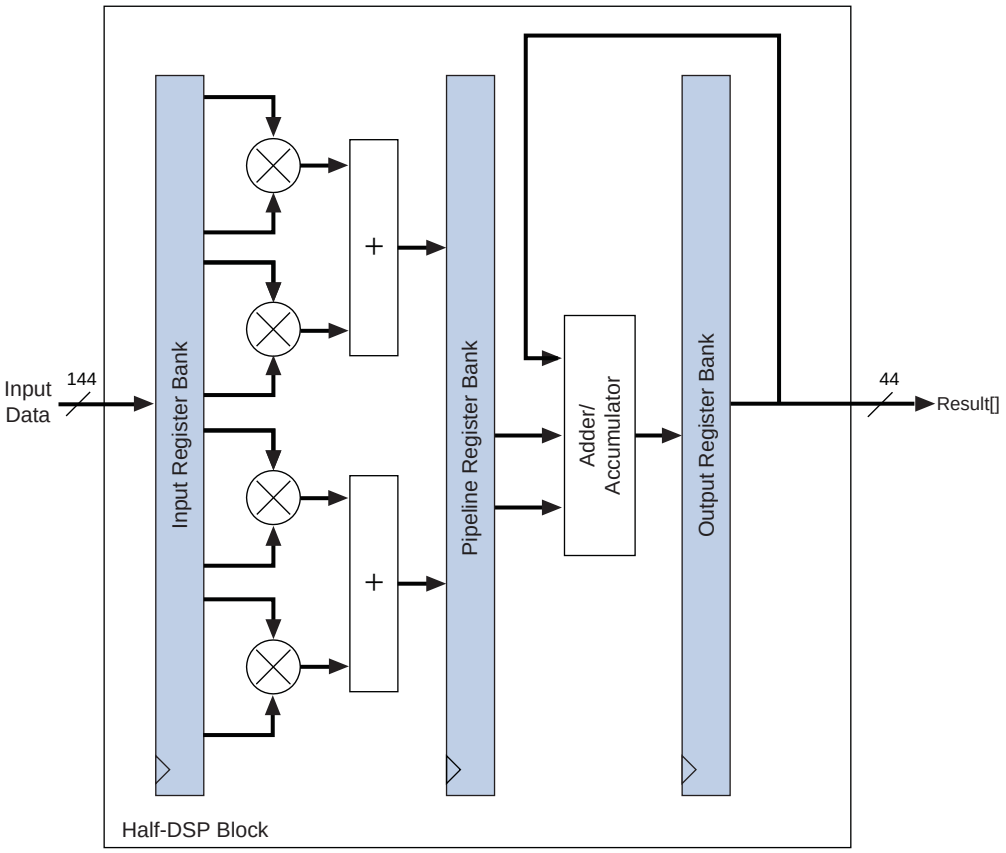
$$W_n[43..0] = W_{n-1}[43..0] \pm Z_n[37..0]$$

In these equations, *n* denotes sample time and *P[36..0]* denotes the result from the two-multiplier adder units.

Equation 4–2 provides a sum of four 18 × 18-bit multiplication operations (four-multiplier adder). Equation 4–3 provides a four 18 × 18-bit multiplication operation but with a maximum 44-bit accumulation capability by feeding the output of the unit back to itself, as shown in Figure 4–3.

Depending on the mode you select, you can bypass all register stages except accumulation and loopback mode. In these two modes, one set of register must be enabled. If the register is not enabled, an infinite loop occurs.

Figure 4–3. Four-Multiplier Adder and Accumulation Capability

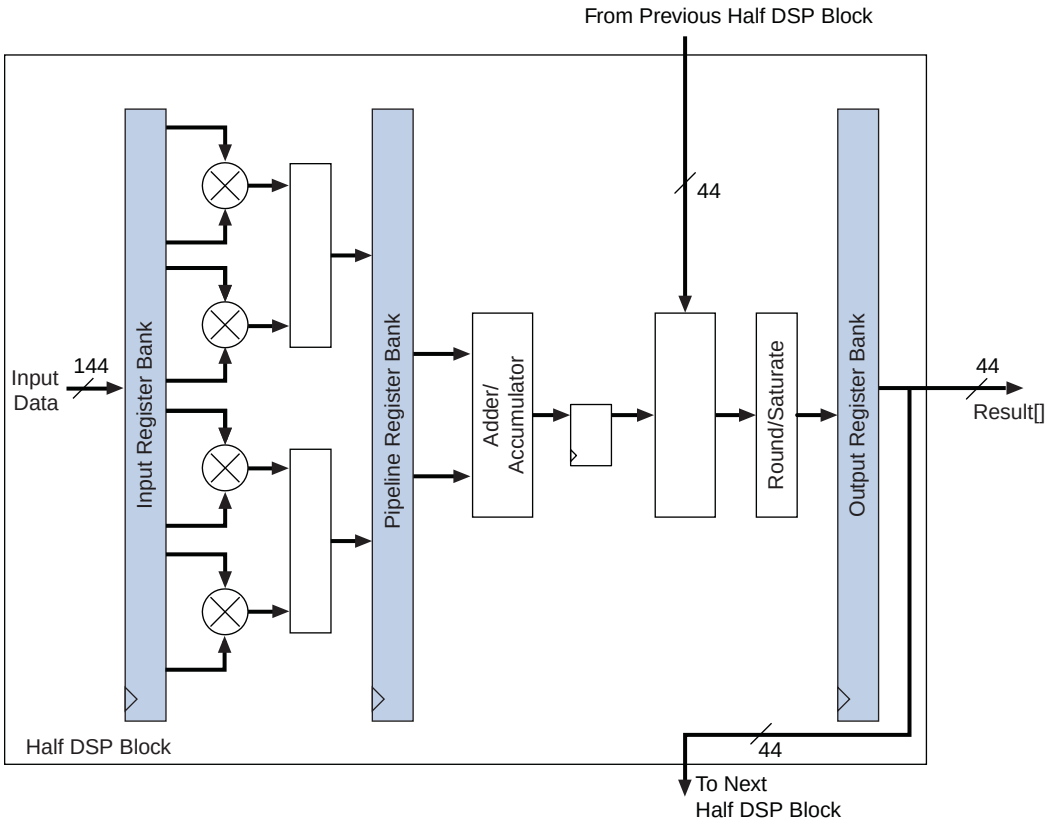


To support commonly found FIR-like structures efficiently, a major addition to the DSP block in Stratix IV devices is the ability to propagate the result of one half block to the next half block completely within the DSP block without additional soft logic overhead. This is achieved by the inclusion of a dedicated addition unit and routing that adds the 44-bit result of a previous half block with the 44-bit result of the current block. The 44-bit result is either fed to the next half block or out of the DSP block using the output register stage, as shown in Figure 4–4. Detailed examples are described in later sections.

The combination of a fast, low-latency four-multiplier adder unit and the "chained cascade" capability of the output chaining adder provides the optimal FIR and vector multiplication capability.

To support single-channel type FIR filters efficiently, you can configure one of the multiplier input's registers to form a tap delay line input, saving resources and providing higher system performance.

Figure 4–4. Output Cascading Feature for FIR Structures

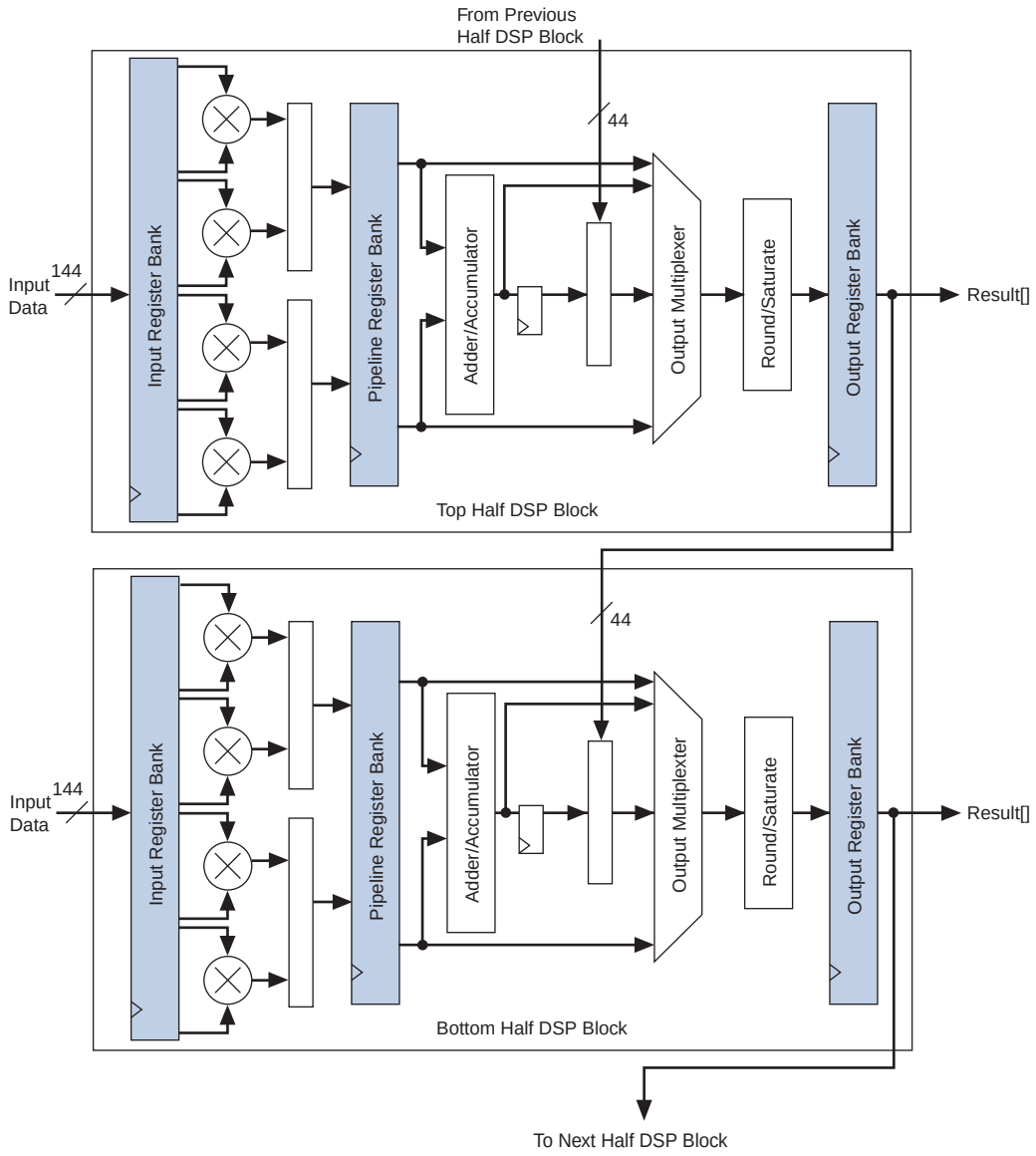


Also shown in Figure 4–4 is the optional rounding and saturation unit (RSU). This unit provides a rich set of commonly found arithmetic rounding and saturation functions used in signal processing.

In addition to the independent multipliers and sum modes, you can use DSP blocks to perform shift operations. DSP blocks can dynamically switch between logical shift left/right, arithmetic shift left/right, and rotation operation in one clock cycle.

Figure 4–5 shows a top-level view of the Stratix IV DSP block.
Figure 4–6 on page 4–9 shows a more detailed top-level view of the DSP block.

Figure 4–5. Stratix IV Full DSP Block



Stratix IV Operational Modes Overview

You can use each Stratix IV DSP block in one of five basic operational modes. Table 4–2 shows the five basic operational modes and the number of multipliers that you can implement within a single DSP block, depending on the mode.

Table 4–2. Stratix IV DSP Block Operation Modes

Mode	Multiplier in Width	# of Mults	# per Block	Signed or Unsigned	RND, SAT	In Shift Register	Chainout Adder	1st Stage Add/Sub	2nd Stage Add/Acc
Independent Multiplier	9 bits	1	8	Both	No	No	No	—	—
	12 bits	1	6	Both	No	No	No	—	—
	18 bits	1	4	Both	Yes	Yes	No	—	—
	36 bits	1	2	Both	No	No	No	—	—
	Double	1	2	Both	No	No	No	—	—
Two-Multiplier Adder *(1)*	18 bits	2	4	Signed *(4)*	Yes	No	No	Both	—
Four-Multiplier Adder	18 bits	4	2	Both	Yes	Yes	Yes	Both	Add Only
Multiply Accumulate	18 bits	4	2	Both	Yes	Yes	Yes	Both	Both
Shift *(2)*	36 bits *(3)*	1	2	Both	No	No	—	—	—
High Precision Multiplier Adder	18×36	2	2	Both	No	No	No	—	Add Only

Notes to Table 4–2:

(1) This mode also supports loopback mode. In loopback mode, the number of loopback multipliers per DSP block is two. You can use the remaining multipliers in regular two-multiplier adder mode.

(2) Dynamic shift mode supports arithmetic shift left, arithmetic shift right, logical shift left, logical shift right, and rotation operation.

(3) Dynamic shift mode operates on a 32-bit input vector but the multiplier width is configured as 36 bits.

(4) Unsigned value is also supported but you must ensure that the result can be contained within 36 bits.

The DSP block consists of two identical halves (the top half and bottom half). Each half has four 18 × 18 multipliers.

The Quartus® II software includes megafunctions used to control the mode of operation of the multipliers. After making the appropriate parameter settings using the megafunction's MegaWizard™ Plug-In Manager, the Quartus II software automatically configures the DSP block.

Stratix IV DSP blocks can operate in different modes simultaneously. Each half block is fully independent except for the sharing of the three `clock`, `ena`, and `aclr` signals. For example, you can break down a single DSP block to operate a 9 × 9 multiplier in one half block and an 18 × 18 two-multiplier adder in the other half block. This increases DSP block resource efficiency and allows you to implement more multipliers within a Stratix IV device. The Quartus II software automatically places multipliers that can share the same DSP block resources within the same block.

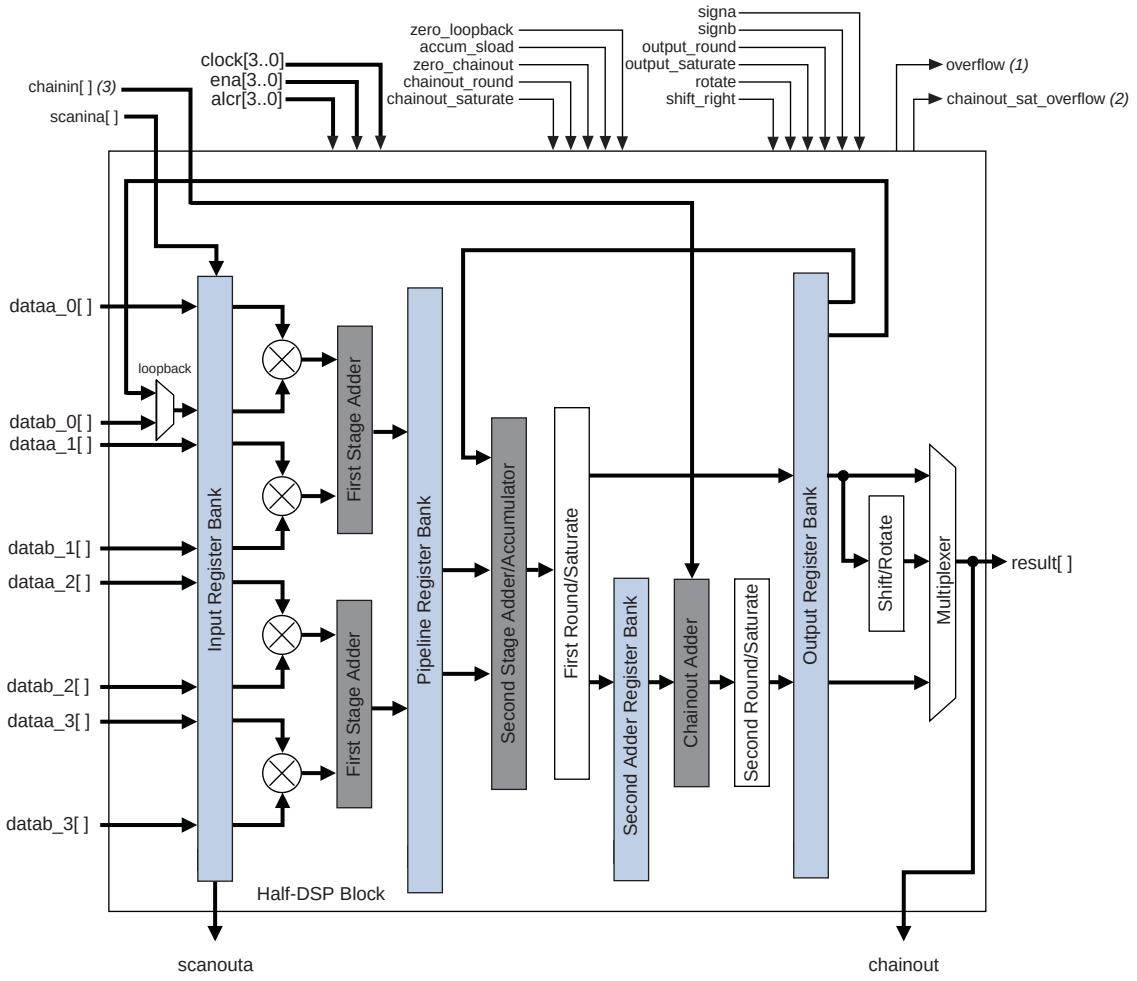
Stratix IV DSP Block Resource Descriptions

The DSP block consists of the following elements:

- Input register bank
- Four two-multiplier adders
- Pipeline register bank
- Two second-stage adders
- Four rounding and saturation logic units
- Second adder register and output register bank

Figure 4–6 shows a detailed overall architecture of the top half of the DSP block. Table 4–9 on page 4–33 shows a list of DSP block dynamic signals.

Figure 4–6. Half DSP Block Architecture



Notes to Figure 4–6:

(1) Block output for accumulator overflow and saturate overflow.

(2) Block output for saturation overflow of chainout.

(3) The chainin port must only be connected to chainout of the previous DSP blocks and must not be connected to general routings.

Input Registers

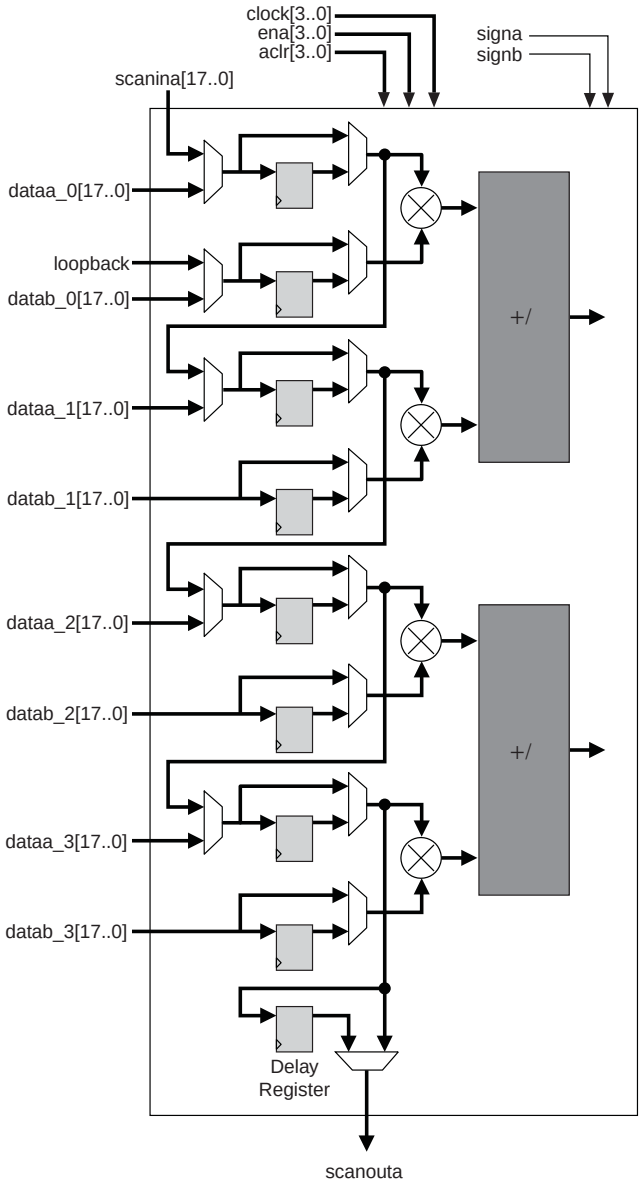
All of the DSP block registers are triggered by the positive edge of the clock signal and are cleared upon power up. Each multiplier operand can feed an input register or go directly to the multiplier, bypassing the input registers. The following DSP block signals control the input registers within the DSP block:

- `clock[3..0]`
- `ena[3..0]`
- `aclr[3..0]`

Every DSP block has nine 18-bit data input register banks per half DSP block. Every half DSP block has the option to use the eight data register banks as inputs to the four multipliers. The special ninth register bank is a delay register required by modes that use both the cascade and chainout features of the DSP block. Use the ninth register bank to balance the latency requirements when using the chained cascade feature.

A feature of the input register bank is to support a tap delay line. Therefore, the top leg of the multiplier input (A) can be driven from general routing or from the cascade chain, as shown in Figure 4–7. Table 4–9 on page 4–33 lists the DSP block dynamic signals.

Figure 4–7. Input Register of a Half DSP Block



At compile time, you must select whether the A-input comes from general routing or from the cascade chain. In cascade mode, the dedicated shift outputs from one multiplier block and directly feeds the input registers of the adjacent multiplier below it (within the same half DSP block) or the first multiplier in the next half DSP block, to form an 8-tap shift register chain per DSP Block. The DSP block can increase the length of the shift register chain by cascading to the lower DSP blocks. The dedicated shift register chain spans a single column, but you can implement longer shift register chains requiring multiple columns using the regular FPGA routing resources.

Shift registers are useful in DSP functions such as FIR filters. When implementing 18 × 18 or smaller width multipliers, you do not need external logic to create the shift register chain because the input shift registers are internal to the DSP block. This implementation significantly reduces the logical element (LE) resources required, avoids routing congestion, and results in predictable timing.

The first multiplier in every half DSP block (top- and bottom-half) in Stratix IV devices has a multiplexer for the first multiplier B-input (lower-leg input) register to select between general routing and loopback, as shown in Figure 4–6 on page 4–9. In loopback mode, the most significant 18-bit registered outputs are connected as feedback to the multiplier input of the first top multiplier in each half DSP block. Loopback modes are used by recursive filters where the previous output is needed to compute the current output.

Loopback mode is described in "Two-Multiplier Adder Sum Mode" on page 4–22.

Table 4–3 lists input register modes for the DSP block.

Table 4–3. Input Register Modes

Register Input Mode *(1)*	9 × 9	12 × 12	18 × 18	36 × 36	Double
Parallel input	✓	✓	✓	✓	✓
Shift register input *(2)*	—	—	✓	—	—
Loopback input *(3)*	—	—	✓	—	—

Notes to Table 4–3:

(1) Multiplier operand input wordlengths are statically configured at compile time.

(2) Available only on the A-operand.

(3) Only one loopback input is allowed per half block. For more information, refer to Figure 4–15 on page 4–23.

Multiplier and First-Stage Adder

The multiplier stage natively supports 9 × 9, 12 × 12, 18 × 18, or 36 × 36 multipliers. Other wordlengths are padded up to the nearest appropriate native wordlength; for example, 16 × 16 would be padded up to use 18 × 18. For more information, refer to "Independent Multiplier Modes" on page 4–15. Depending on the data width of the multiplier, a single DSP block can perform many multiplications in parallel.

Each multiplier operand can be a unique signed or unsigned number. Two dynamic signals, `signa` and `signb`, control the representation of each operand, respectively. A `logic 1` value on the `signa/signb` signal indicates that `data A/data B` is a signed number; a `logic 0` value indicates an unsigned number. Table 4–4 shows the sign of the multiplication result for the various operand sign representations. The result of the multiplication is signed if any one of the operands is a signed value.

Table 4–4. Multiplier Sign Representation

Data A (signa Value)	Data B (signb Value)	Result
Unsigned (logic 0)	Unsigned (logic 0)	Unsigned
Unsigned (logic 0)	Signed (logic 1)	Signed
Signed (logic 1)	Unsigned (logic 0)	Signed
Signed (logic 1)	Signed (logic 1)	Signed

Each half block has its own `signa` and `signb` signal. Therefore, all of the `data A` inputs feeding the same half DSP block must have the same sign representation. Similarly, all of the `data B` inputs feeding the same half DSP block must have the same sign representation. The multiplier offers full precision regardless of the sign representation in all operational modes except for full precision 18 × 18 loopback and two-multiplier adder modes. For more information, refer to "Two-Multiplier Adder Sum Mode" on page 4–22.

By default, when the `signa` and `signb` signals are unused, the Quartus II software sets the multiplier to perform unsigned multiplication.

Figure 4–6 on page 4–9 shows that the outputs of the multipliers are the only outputs that can feed into the first-stage adder. There are four first-stage adders in a DSP block (two adders per half DSP block). The first-stage adder block has the ability to perform addition and subtraction. The control signal for addition or subtraction is static and has to be configured upon compile time. The first-stage adders are used by the sum modes to compute the sum of two multipliers, 18 × 18-complex multipliers, and to perform the first stage of a 36 × 36 multiply and shift operations.

Depending on your specifications, the output of the first-stage adder has the option to feed into the pipeline registers, second-stage adder, rounding and saturation unit, or output registers.

Pipeline Register Stage

Figure 4–6 on page 4–9 shows that the output from the first-stage adder can either feed or bypass the pipeline registers. Pipeline registers increase the DSP block's maximum performance (at the expense of extra cycles of latency), especially when using the subsequent DSP block stages. Pipeline registers split up the long signal path between the input registers/multiplier/first-stage adder and the second-stage adder/round-and-saturation/output registers, creating two shorter paths.

Second-Stage Adder

There are four individual 44-bit second-stage adders per DSP block (two adders per half DSP block). You can configure the second-stage adders as follows:

- The final stage of a 36-bit multiplier
- A sum of four (18 × 18)
- An accumulator (44-bits maximum)
- A chained output summation (44-bits maximum)

You can use the chained-output adder at the same time as a second-level adder in chained output summation mode.

The output of the second-stage adder has the option to go into the rounding and saturation logic unit or the output register.

You cannot use the second-stage adder independently from the multiplier and first-stage adder.

Rounding and Saturation Stage

The rounding and saturation logic units are located at the output of the 44-bit second-stage adder (the rounding logic unit followed by the saturation logic unit). There are two rounding and saturation logic units per half DSP block. The input to the rounding and saturation logic unit can come from one of the following stages:

- Output of the multiplier (independent multiply mode in 18 × 18)
- Output of the first-stage adder (two-multiplier adder)
- Output of the pipeline registers
- Output of the second-stage adder (four-multiplier adder and multiply-accumulate mode in 18 × 18)

These stages are described in "Stratix IV Operational Mode Descriptions" on page 4–15.

The rounding and saturation logic unit is controlled by the dynamic rounding and saturate signals, respectively. A `logic 1` value on the rounding and/or saturate signals enables the rounding and/or saturate logic unit, respectively.

You can use the rounding and saturation logic units together or independently.

Second Adder and Output Registers

The second adder register and output register banks are two banks of 44-bit registers that you can combine to form larger 72-bit banks to support 36 × 36 output results.

The outputs of the different stages in the Stratix IV devices are routed to the output registers through an output selection unit. Depending on the operational mode of the DSP block, the output selection unit selects whether the outputs of the DSP blocks comes from the outputs of the multiplier block, first-stage adder, pipeline registers, second-stage adder, or the rounding and saturation logic unit. The output selection unit is set automatically by the software, based on the DSP block operational mode you specified, and has the option to either drive or bypass the output registers. The exception is when you use the block in shift mode, in which case you dynamically control the output-select multiplexer directly.

When the DSP block is configured in chained cascaded output mode, both of the second-stage adders are used. Use the first one for performing a four-multiplier adder; use the second for the chainout adder.

The outputs of the four-multiplier adder are routed to the second-stage adder registers before they enter the chainout adder. The output of the chainout adder goes to the regular output register bank. Depending on the configuration, you can route the chainout results to the input of the next half block's chainout adder input or to the general fabric (functioning as regular output registers). For more information, refer to "Stratix IV Operational Mode Descriptions" on page 4–15.

The second-stage and output registers are triggered by the positive edge of the clock signal and are cleared on power up. The following DSP block signals control the output registers within the DSP block:

- `clock[3..0]`
- `ena[3..0]`
- `aclr[3..0]`

Stratix IV Operational Mode Descriptions

This section contains an explanation of different operational modes in Stratix IV devices.

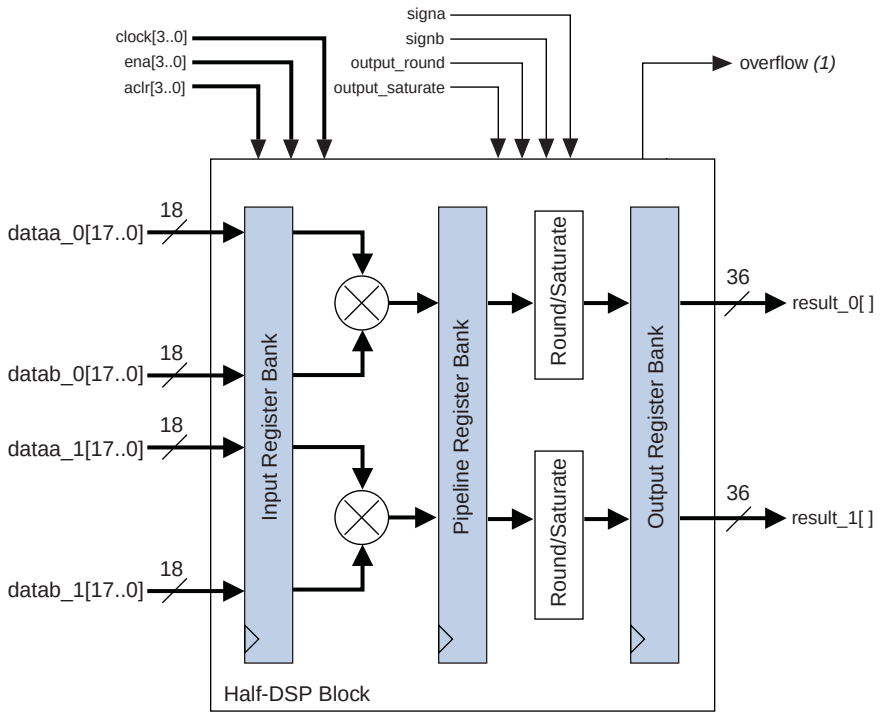
Independent Multiplier Modes

In independent input and output multiplier mode, the DSP block performs individual multiplication operations for general-purpose multipliers.

9-, 12-, and 18-Bit Multiplier

You can configure each DSP block multiplier for 9-, 12-, or 18-bit multiplication. A single DSP block can support up to eight individual 9 × 9 multipliers, six individual 12 × 12 multipliers, or four individual 18 × 18 multipliers. For operand widths up to 9 bits, a 9 × 9 multiplier is implemented. For operand widths from 10 to 12 bits, a 12 × 12 multiplier is implemented, and for operand widths from 13 to 18 bits, an 18 × 18 multiplier is implemented. This is done by the Quartus II software by zero-padding the LSBs. Figure 4–8, Figure 4–9, and Figure 4–10 show the DSP block in the independent multiplier operation. Table 4–9 on page 4–33 lists the dynamic signals for the DSP block.

Figure 4–8. 18-Bit Independent Multiplier Mode Shown for a Half DSP Block



Note to Figure 4–8:

(1) Block output for accumulator overflow and saturate overflow.

Figure 4–9. 12-Bit Independent Multiplier Mode Shown for a Half DSP Block

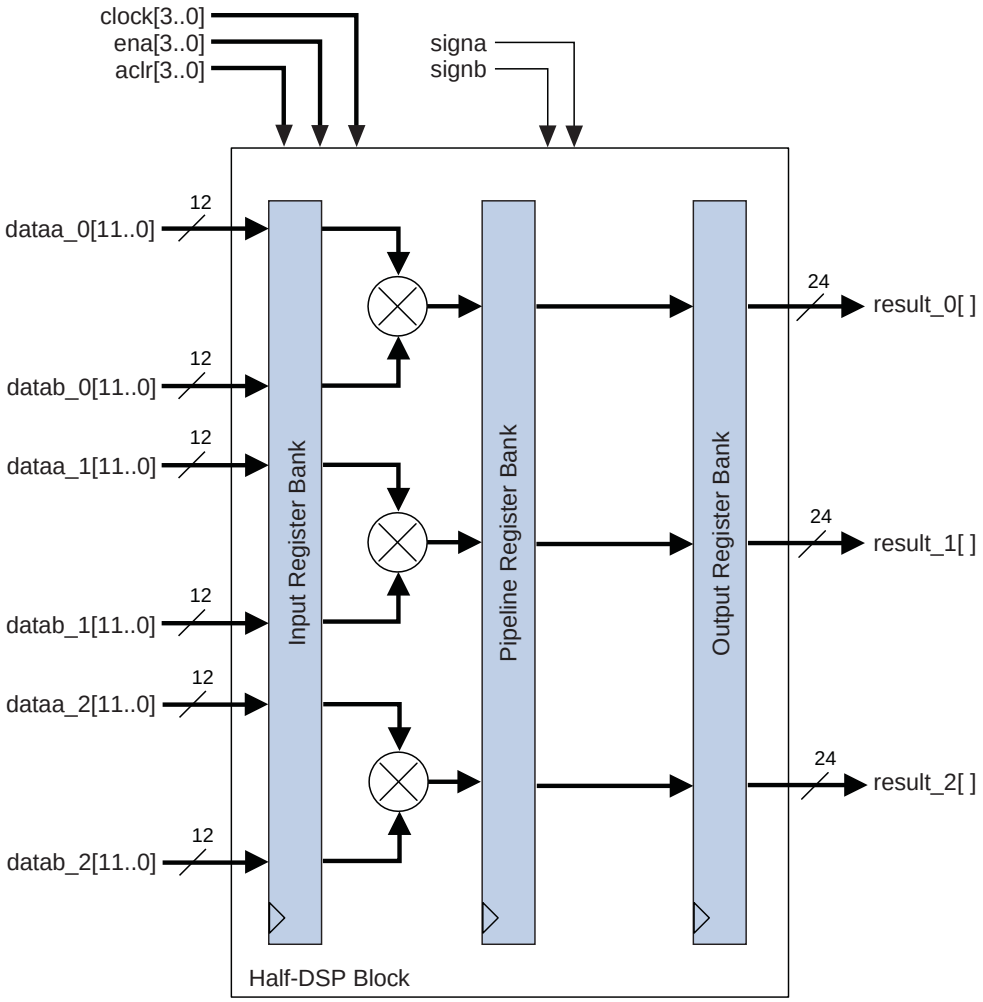
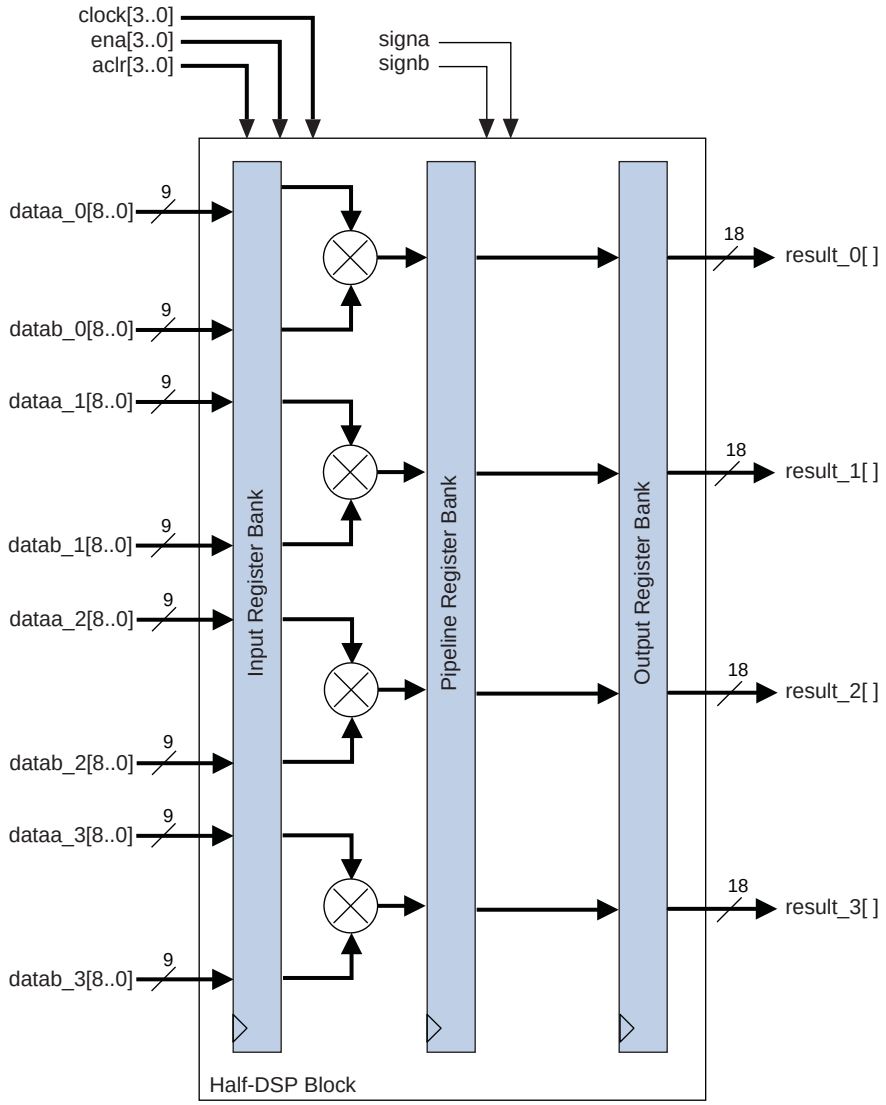


Figure 4–10. 9-Bit Independent Multiplier Mode Shown for a Half Block



The multiplier operands can accept signed integers, unsigned integers, or a combination of both. You can change the `signa` and `signb` signals dynamically and can register the signals in the DSP block. Additionally, the multiplier inputs and results can be registered independently. You can use the pipeline registers within the DSP block to pipeline the multiplier result, increasing the performance of the DSP block.

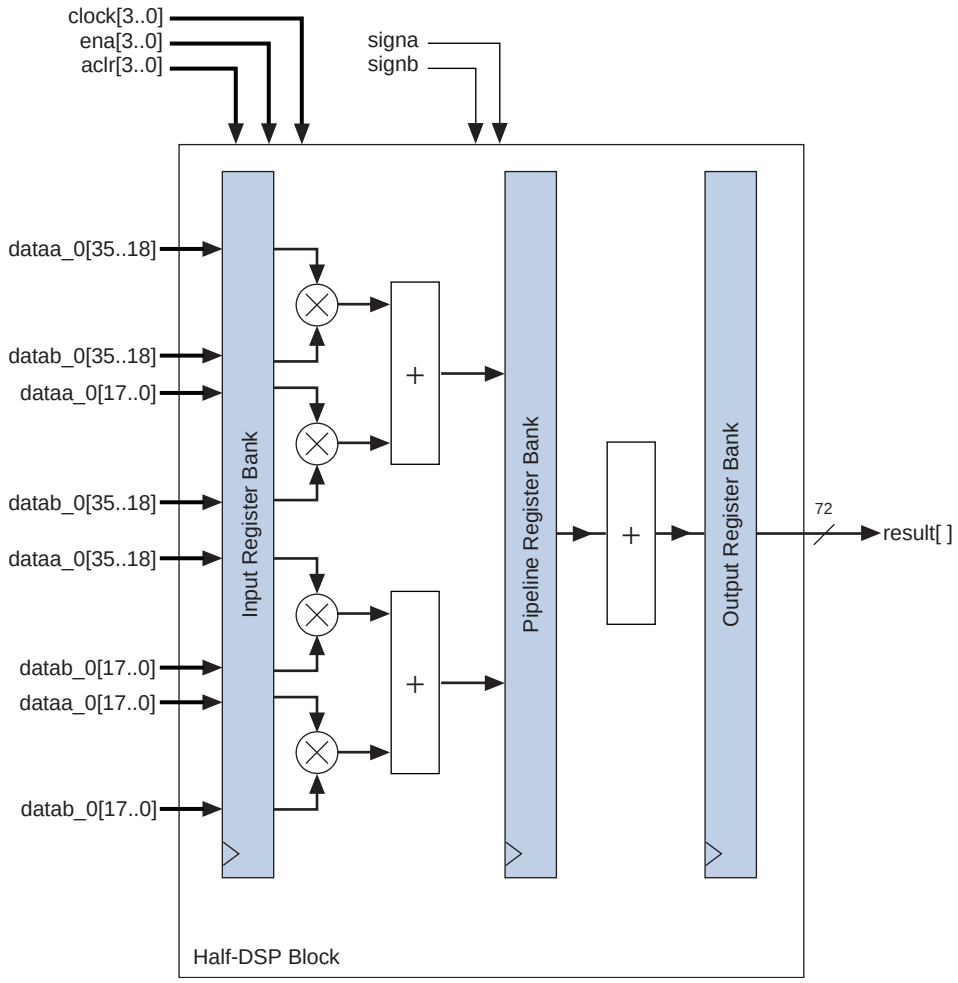
The rounding and saturation logic unit is supported for 18-bit independent multiplier mode only.

36-Bit Multiplier

You can efficiently construct a 36 × 36 multiplier using four 18 × 18 multipliers. This simplification fits conveniently into one half DSP block and is implemented in the DSP block automatically by selecting 36 × 36 mode. Stratix IV devices can have up to two 36-bit multipliers per DSP block (one 36-bit multiplier per half DSP block). The 36-bit multiplier is also under the independent multiplier mode but uses the entire half DSP block, including the dedicated hardware logic after the pipeline registers to implement the 36 × 36 bit multiplication operation, as shown in Figure 4–11.

The 36-bit multiplier is useful for applications requiring more than 18-bit precision; for example, for the mantissa multiplication portion of single precision and extended single precision floating-point arithmetic applications.

Figure 4–11. 36-Bit Independent Multiplier Mode Shown for a Half DSP Block



Double Multiplier

You can configure the Stratix IV DSP block to efficiently support a signed or unsigned 54 × 54-bit multiplier that is required to compute the mantissa portion of an IEEE double-precision floating point multiplication. You can build a 54 × 54-bit multiplier using basic 18 × 18 multipliers, shifters, and adders. In order to efficiently utilize the Stratix IV DSP block's built-in shifters and adders, a special double mode (partial 54 × 54 multiplier) is available that is a slight modification to the basic 36 × 36 multiplier mode, as shown in Figure 4–12 and Figure 4–13.

Figure 4–12. Double Mode Shown for a Half DSP Block

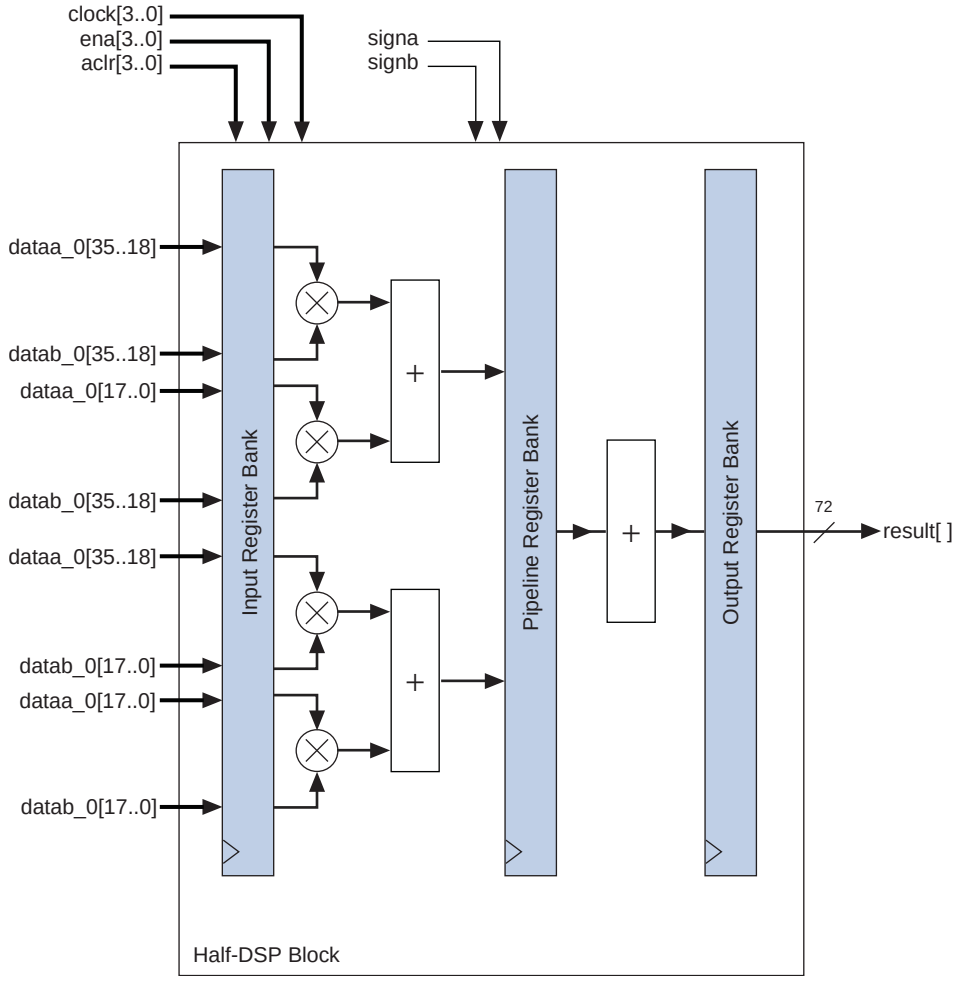
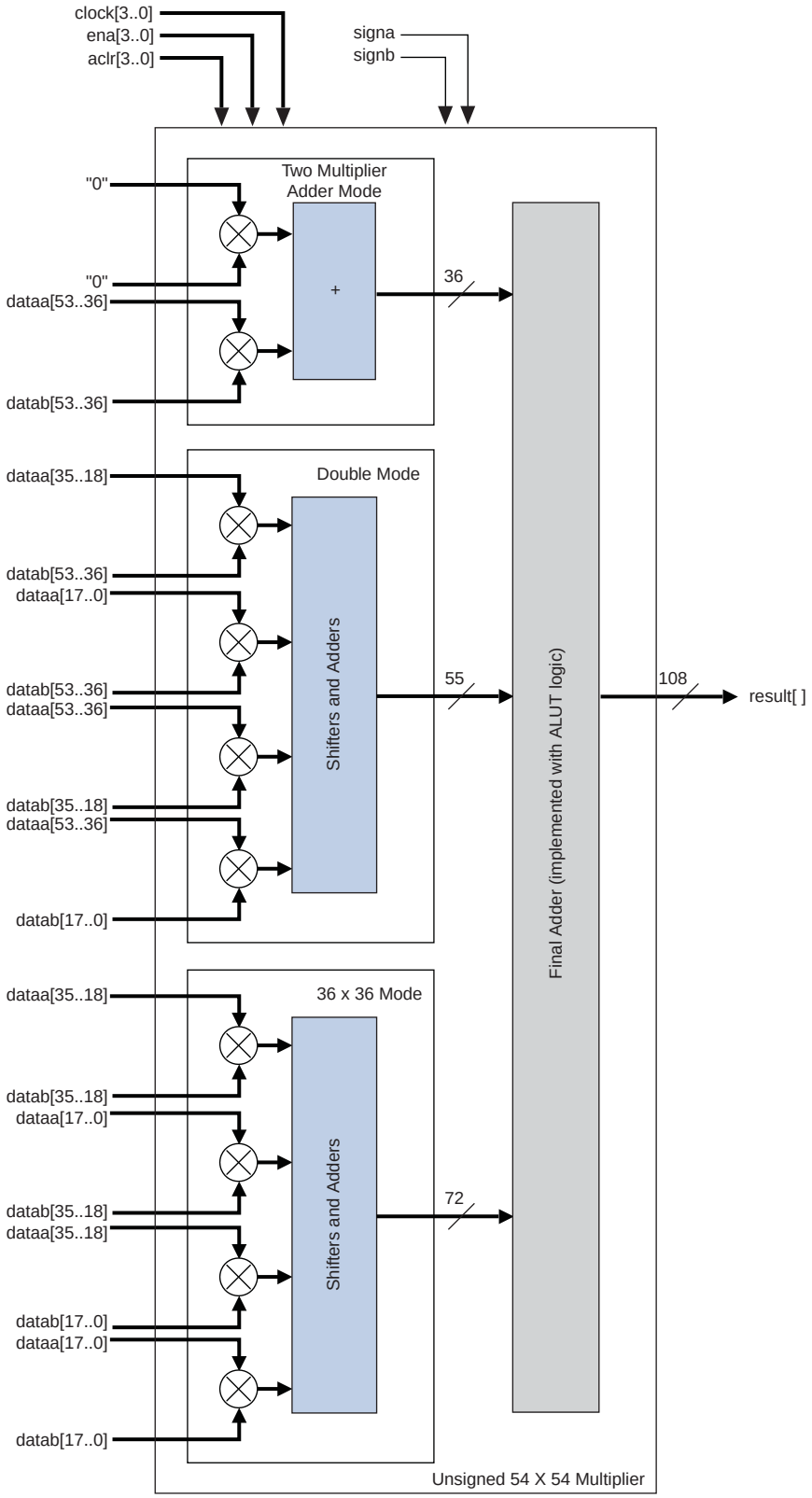


Figure 4–13. Unsigned 54 × 54 Multiplier for a Half-DSP Block



Two-Multiplier Adder Sum Mode

In the two-multiplier adder configuration, the DSP block can implement four 18-bit two-multiplier adders (2 two-multiplier adders per half DSP block). You can configure the adders to take the sum or difference of two multiplier outputs. You must select summation or subtraction at compile time. The two-multiplier adder function is useful for applications such as FFTs, complex FIR, and IIR filters. Figure 4–14 shows the DSP block configured in two-multiplier adder mode.

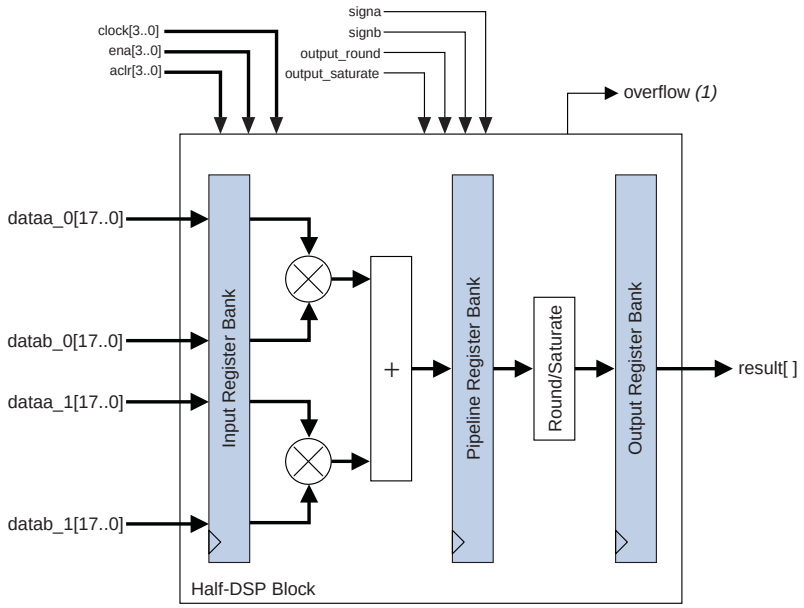
Loopback mode is the other sub-feature of the two-multiplier adder mode. Figure 4–15 shows the DSP block configured in the loopback mode. This mode takes the 36-bit summation result of the two multipliers and feeds back the most significant 18-bits to the input. The lower 18-bits are discarded. You have the option to disable or zero-out the loopback data by using the dynamic `zero_loopback` signal. A `logic 1` value on the `zero_loopback` signal selects the `zeroed` data or disables the looped back data, while a `logic 0` selects the looped back data.

You must select the option to use loopback mode or the general two-multiplier adder mode at compile time.

For two-multiplier adder mode, if all the inputs are full 18-bit and unsigned, the result requires 37 bits. As the output data width in two-multiplier adder mode is limited to 36 bits, this 37-bit output requirement is not allowed. Any other combination that does not violate the 36-bit maximum result is permitted; for example, two 16 × 16 signed two-multiplier adders is valid.

Two-multiplier adder mode supports the rounding and saturation logic unit. You can use the pipeline registers and output registers within the DSP block to pipeline the multiplier-adder result, increasing the performance of the DSP block.

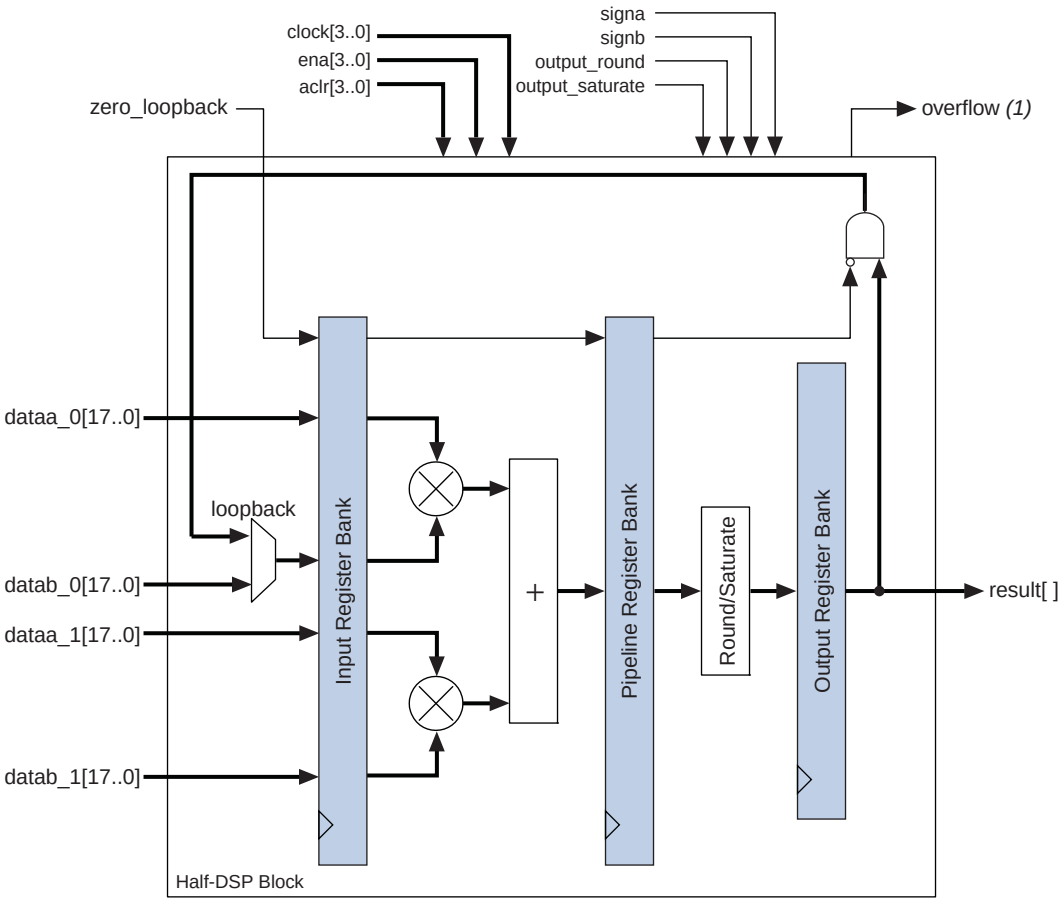
Figure 4–14. Two-Multiplier Adder Mode Shown for a Half DSP Block



Note to Figure 4–14:

(1) Block output for accumulator overflow and saturate overflow.

Figure 4–15. Loopback Mode for a Half DSP Block



Note to Figure 4–15:

(1) Block output for accumulator overflow and saturate overflow.

18 x 18 Complex Multiply

You can configure the DSP block to implement complex multipliers using two-multiplier adder mode. A single half DSP block can implement one 18-bit complex multiplier.

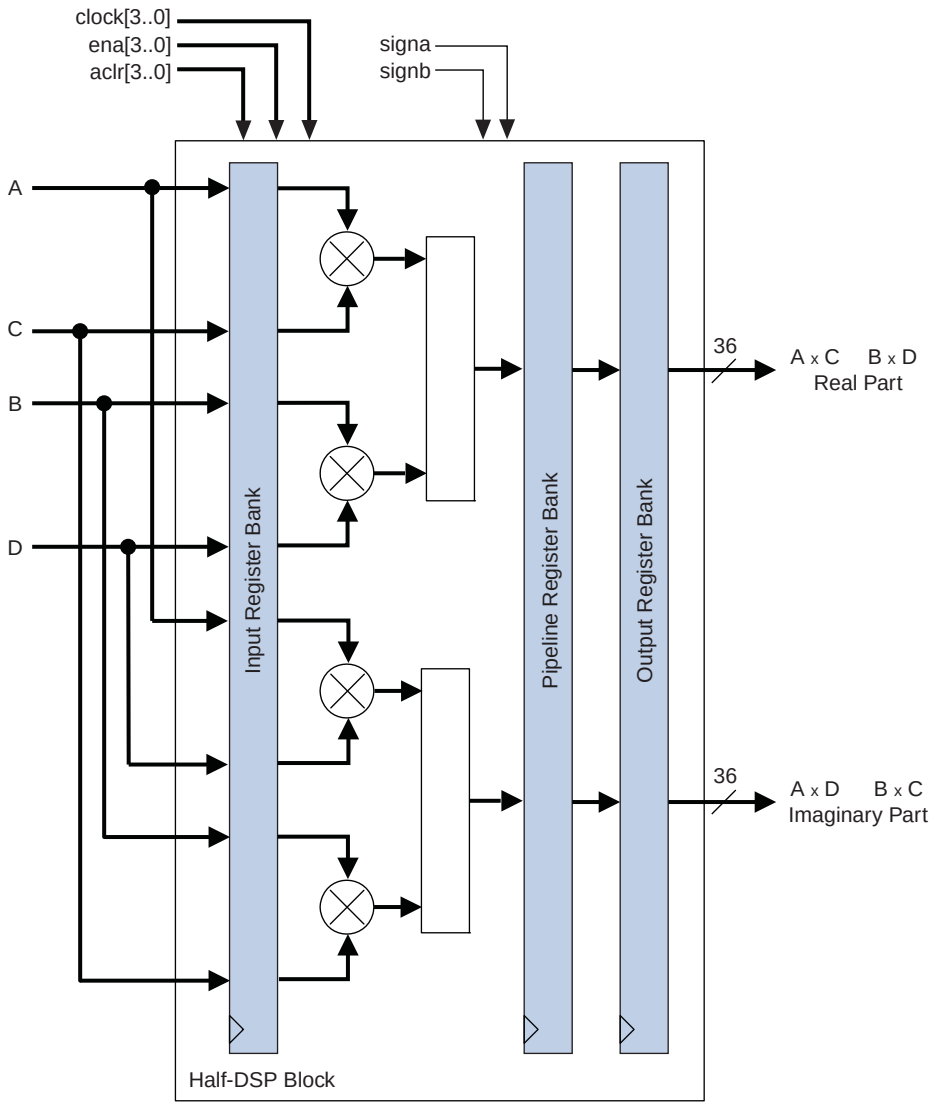
Equation 4–4 shows a complex multiplication.

Equation 4–4. Complex Multiplication Equation

$$(a + jb) \times (c + jd) = ((a \times c) - (b \times d)) + j((a \times d) + (b \times c))$$

To implement this complex multiplication within the DSP block, the real part $((a \times c) - (b \times d))$ is implemented using two multipliers feeding one subtractor block while the imaginary part $((a \times d) + (b \times c))$ is implemented using another two multipliers feeding an adder block. Figure 4–16 shows an 18-bit complex multiplication. This mode automatically assumes all inputs are using signed numbers.

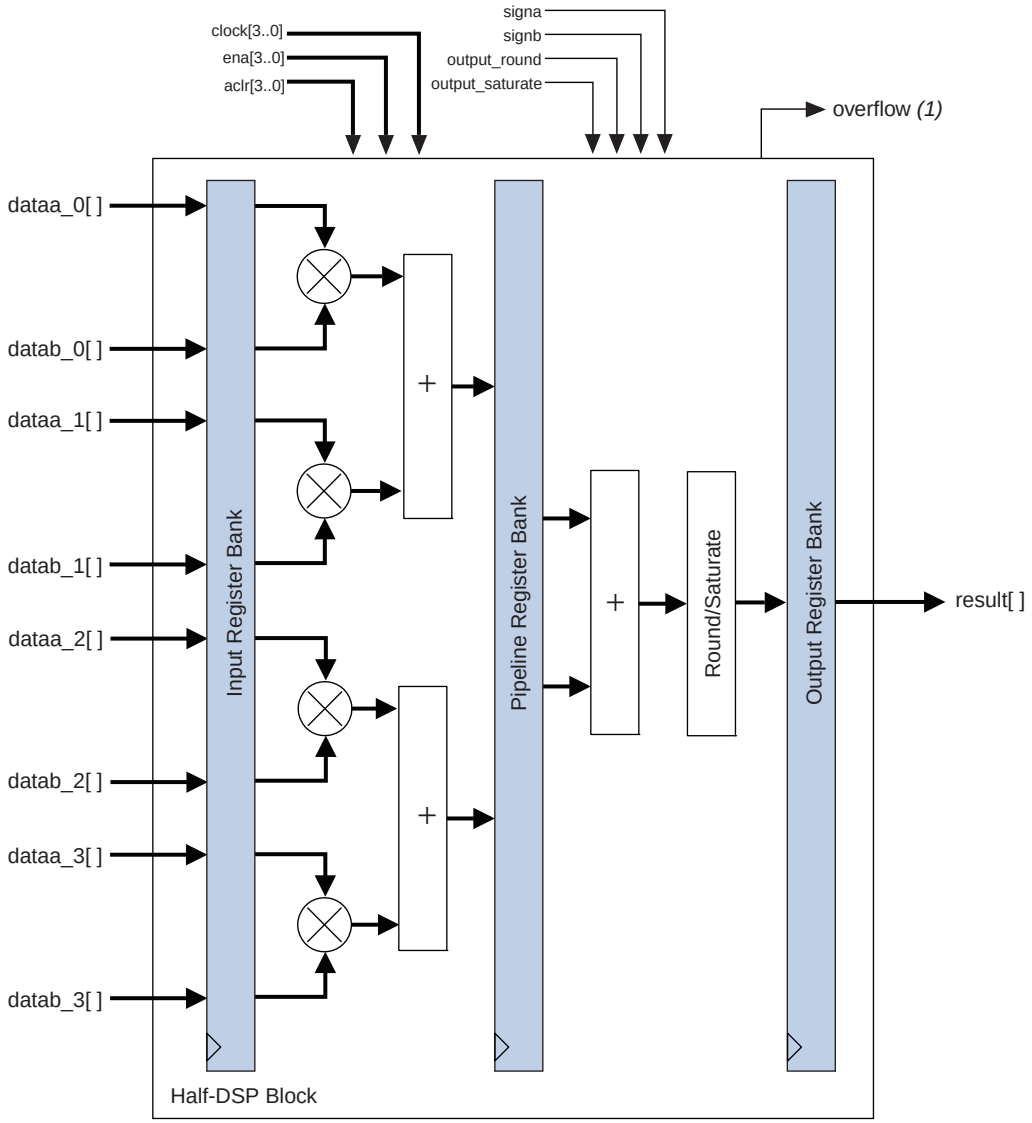
Figure 4–16. Complex Multiplier Using Two-Multiplier Adder Mode



Four-Multiplier Adder

In the four-multiplier adder configuration shown in Figure 4–17, the DSP block can implement two four-multiplier adders (one four-multiplier adder per half DSP block). These modes are useful for implementing one-dimensional and two-dimensional filtering applications. The four-multiplier adder is performed in two addition stages. The outputs of two of the four multipliers are initially summed in the two first-stage adder blocks. The results of these two adder blocks are then summed in the second-stage adder block to produce the final four-multiplier adder result, as shown by Equation 4–2 on page 4–4 and Equation 4–3 on page 4–4.

Figure 4–17. Four-Multiplier Adder Mode Shown for a Half DSP Block



Note to Figure 4–17:

(1) Block output for accumulator overflow and saturate overflow.

Four-multiplier adder mode supports the rounding and saturation logic unit. You can use the pipeline registers and output registers within the DSP block to pipeline the multiplier-adder result, increasing the performance of the DSP block.

High-Precision Multiplier Adder Mode

In the high-precision multiplier adder configuration, shown in Figure 4–18, the DSP block can implement 2 two-multiplier adders, with multiplier precision of 18 x 36 (one two-multiplier adder per half DSP block). This mode is useful in filtering or FFT applications where a data path greater than 18 bits is required, yet 18 bits is sufficient for the coefficient precision. This can occur where the data has a high dynamic range. If the coefficients are fixed, as in FFT and most filter applications, the precision of 18 bits provide a dynamic range over 100 dB, if the largest coefficient is normalized to the maximum 18-bit representation.

In these situations, the data path can be up to 36 bits, allowing sufficient capacity for bit growth or gain changes in the signal source without loss of precision. This mode is also extremely useful in single precision block floating point applications.

The high-precision multiplier adder is performed in two stages. The 18 × 36 multiply is divided into two 18 × 18 multipliers. The multiplier with the LSB of the data source is performed unsigned, while the multiplier with the MSB of the data source can be signed or unsigned. The latter multiplier has its result left shifted by 18 bits prior to the first adder stage, creating an effective 18 x 36 multiplier. The results of these two adder blocks are then summed in the second stage adder block to produce the final result:

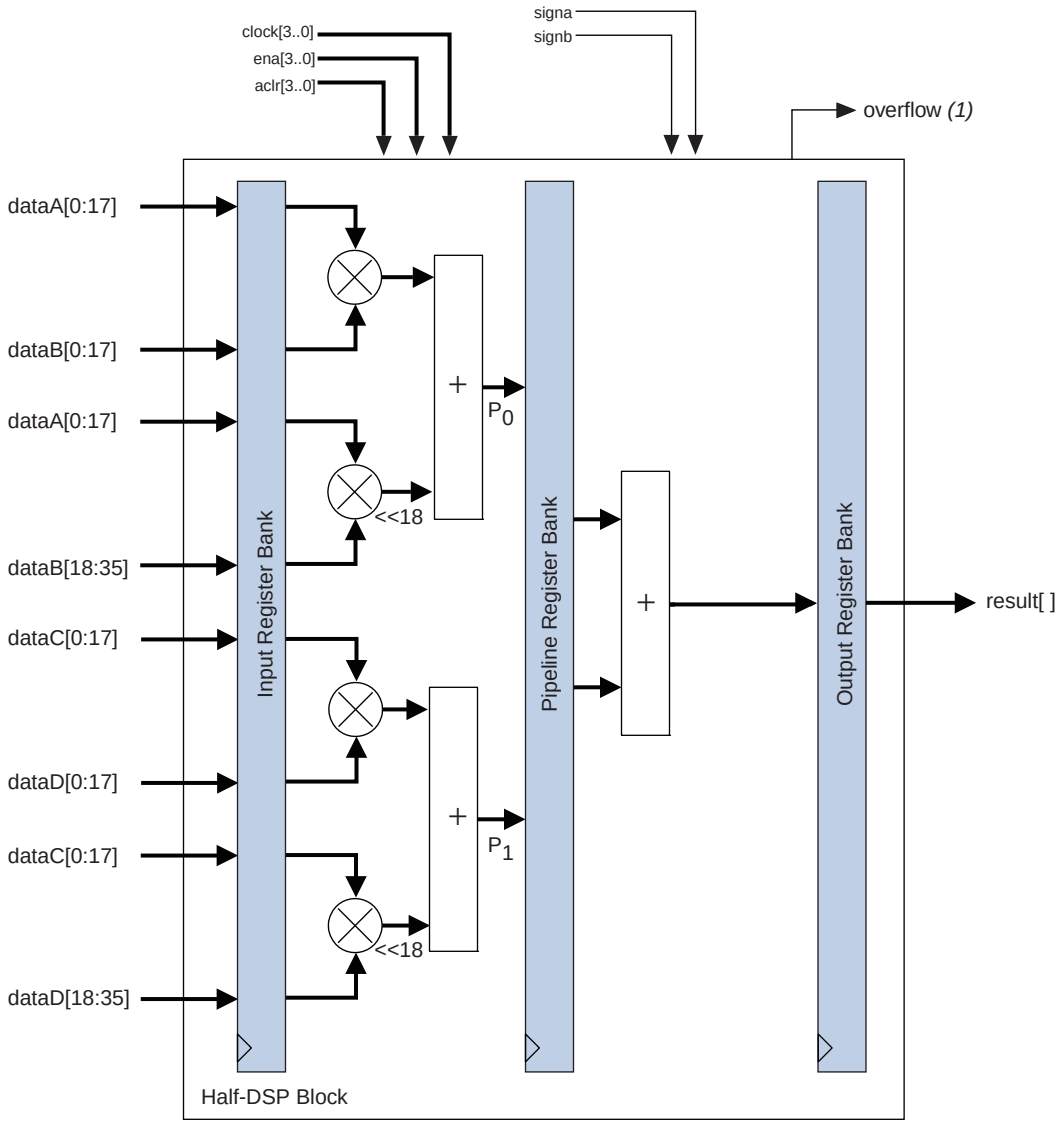
$Z[54..0] = P_0[53..0] + P_1[53..0]$

where:

$P_0 = A[17..0] \times B[35..0]$

$P_1 = C[17..0] \times D[35..0]$

Figure 4–18. High-Precision Multiplier Adder Configuration



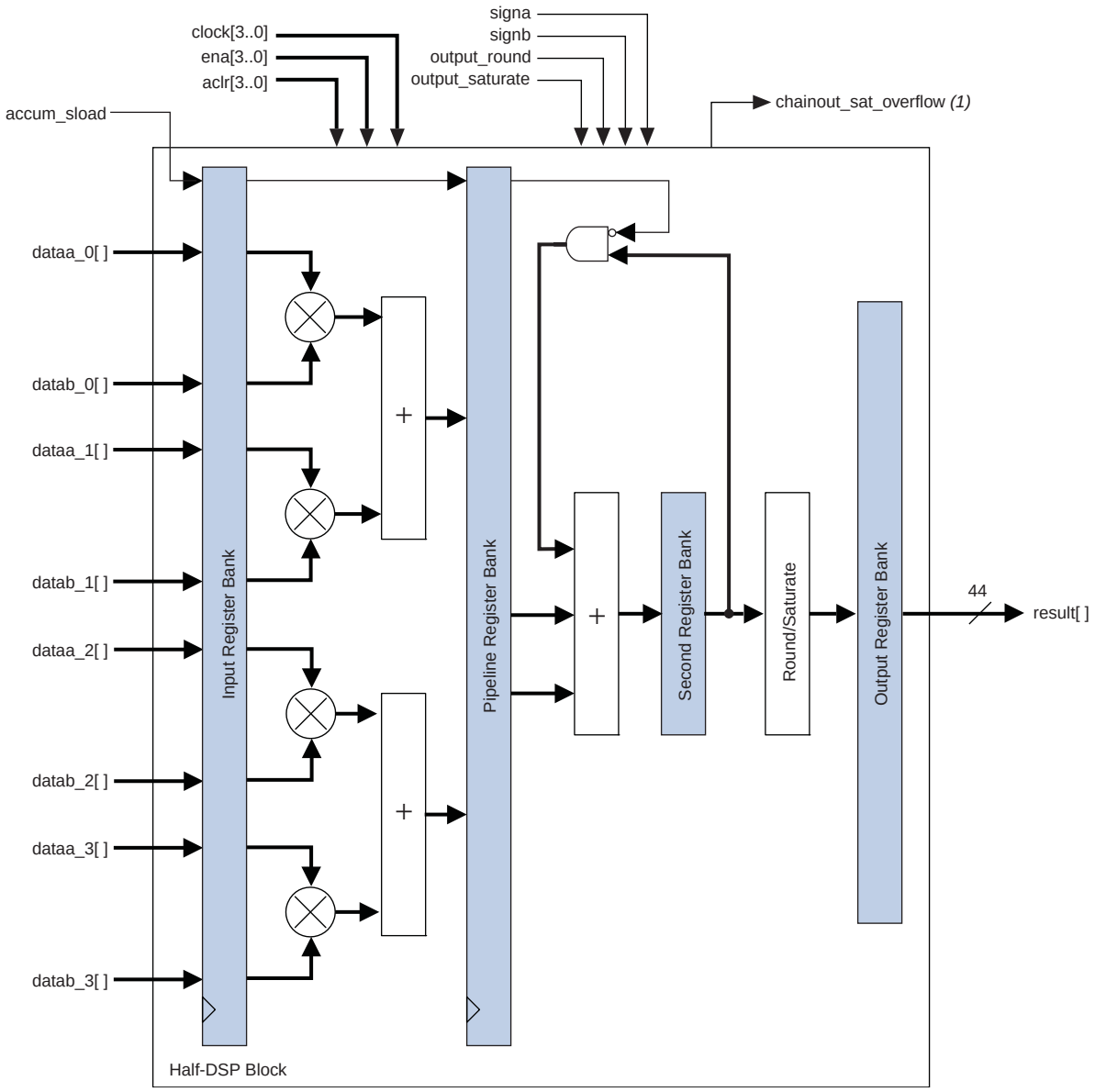
Note to Figure 4–18:

(1) Block output for accumulator overflow and saturate overflow.

Multiply Accumulate Mode

In multiply accumulate mode, the second-stage adder is configured as a 44-bit accumulator or subtractor. The output of the DSP block is looped back to the second-stage adder and added or subtracted with the two outputs of the first-stage adder block according to Equation 4–3 on page 4–4. Figure 4–19 shows the DSP block configured to operate in multiply accumulate mode.

Figure 4–19. Multiply Accumulate Mode Shown for a Half DSP Block



Note to Figure 4–19:

(1) Block output for saturation overflow of chainout.

A single DSP block can implement up to two independent 44-bit accumulators.

Use the dynamic `accum_sload` control signal to clear the accumulation. A `logic 1` value on the `accum_sload` signal synchronously loads the accumulator with the multiplier result only, while a `logic 0` enables accumulation by adding or subtracting the output of the DSP block (accumulator feedback) to the output of the multiplier and first-stage adder.

You must configure the control signal for the accumulator and subtractor is static at compile time.

This mode supports the rounding and saturation logic unit because it is configured as an 18-bit multiplier accumulator. You can use the pipeline registers and output registers within the DSP block to increase the performance of the DSP block.

Shift Modes

Stratix IV devices support the following shift modes for 32-bit input only:

- Arithmetic shift left, `ASL[N]`
- Arithmetic shift right, `ASR[32-N]`
- Logical shift left, `LSL[N]`
- Logical shift right, `LSR[32-N]`
- 32-bit rotator or barrel shifter, `ROT[N]`

You can switch shift mode between these modes using the dynamic rotate and shift control signals.

You can use shift mode in a Stratix IV device by using a soft embedded processor such as Nios® II to perform the dynamic shift and rotate operation. Figure 4–20 shows the shift mode configuration.

Shift mode makes use of the available multipliers to logically or arithmetically shift left, right, or rotate the desired 32-bit data. You can configure the DSP block similar to the independent 36-bit multiplier mode to perform the shift mode operations.

Arithmetic shift right requires a signed input vector. During an arithmetic shift right, the sign is extended to fill the MSB of the 32-bit vector. The logical shift right uses an unsigned input vector. During a logical shift right, zeros are padded in the MSBs, shifting the 32-bit vector to the right. The barrel shifter uses unsigned input vector and implements a rotation function on a 32-bit word length.

Two control signals, `rotate` and `shift_right`, together with the `signa` and `signb` signals, determine the shifting operation. Table 4–5 shows examples of shift operations.

Figure 4–20. Shift Operation Mode Shown for a Half DSP Block

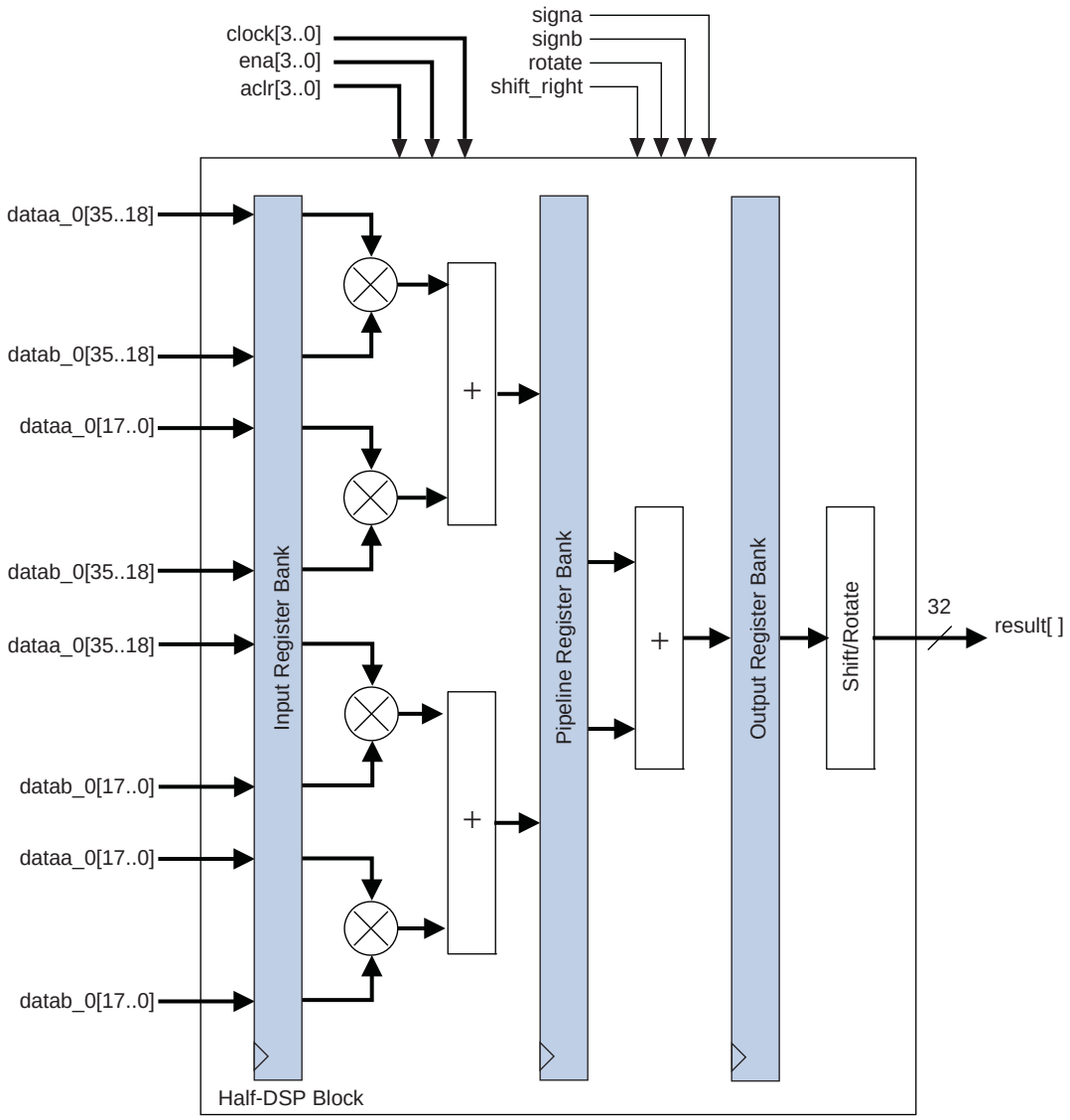


Table 4–5. Examples of Shift Operations

Example	Signa	Signb	Shift	Rotate	A-input	B-input	Result
Logical Shift Left `LSL[N]`	Unsigned	Unsigned	0	0	0xAABBCCDD	0x0000100	0xBBCCDD00
Logical Shift Right `LSR[32-N]`	Unsigned	Unsigned	1	0	0xAABBCCDD	0x0000100	0x000000AA
Arithmetic Shift Left `ASL[N]`	Signed	Unsigned	0	0	0xAABBCCDD	0x0000100	0xBBCCDD00
Arithmetic Shift Right `ASR[32-N]`	Signed	Unsigned	1	0	0xAABBCCDD	0x0000100	0xFFFFFFAA
Rotation `ROT[N]`	Unsigned	Unsigned	0	1	0xAABBCCDD	0x0000100	0xBBCCDDAA

Rounding and Saturation Mode

Rounding and saturation functions are often required in DSP arithmetic. Use rounding to limit bit growth and its side effects; use saturation to reduce overflow and underflow side effects.

Two rounding modes are supported in Stratix IV devices:

- Round-to-nearest-integer mode
- Round-to-nearest-even mode

☞ You must select one of these two options at compile time.

Round-to-nearest-integer provides the biased rounding support and is the simplest form of rounding commonly used in DSP arithmetic. The round-to-nearest-even method provides unbiased rounding support and is used where DC offsets are a concern. Table 4–6 shows how round-to-nearest-even works.

Table 4–7 shows examples of the difference between the two modes. In this example, a 6-bit input is rounded to 4 bits. Table 4–7 shows the main difference between the two rounding options is when the residue bits are exactly halfway between its nearest two integers and the LSB is zero (even).

Table 4–6. Example of Round-To-Nearest-Even Mode

6- to 4-bits Rounding	Odd/Even (Integer)	Fractional	Add to Integer	Result
010111	x	> 0.5 (11)	1	0110
001101	x	< 0.5 (01)	0	0011
001010	Even (0010)	= 0.5 (10)	0	0010
001110	Odd (0011)	= 0.5 (10)	1	0100
110111	x	> 0.5 (11)	1	1110
101101	x	< 0.5 (01)	0	1011
110110	Odd (1101)	= 0.5 (10)	1	1110
110010	Even (1100)	= 0.5 (10)	0	1100

Table 4–7. Comparison of Round-to-Nearest-Integer and Round-to-Nearest-Even

Round-To-Nearest-Integer	Round-To-Nearest-Even
010111 ⇨ 0110	010111 ⇨ 0110
001101 ⇨ 0011	001101 ⇨ 0011
001010 ⇨ 0011	001010 ⇨ 0010
001110 ⇨ 0100	001110 ⇨ 0100
110111 ⇨ 1110	110111 ⇨ 1110
101101 ⇨ 1011	101101 ⇨ 1011
110110 ⇨ 1110	110110 ⇨ 1110
110010 ⇨ 1101	110010 ⇨ 1100

Two saturation modes are supported in Stratix IV:

- Asymmetric saturation mode
- Symmetric saturation mode

You must select one of the two options at compile time.

In 2's-complement format, the maximum negative number that can be represented is $-2^{(n-1)}$, while the maximum positive number is $2^{(n-1)} - 1$. Symmetrical saturation limits the maximum negative number to $-2^{(n-1)} + 1$. For example, for 32 bits:

- Asymmetric 32-bit saturation: Max = 0x7FFFFFFF, Min = 0x80000000
- Symmetric 32-bit saturation: Max = 0x7FFFFFFF, Min = 0x80000001

Table 4–8 shows how saturation works. In this example, a 44-bit input is saturated to 36-bits.

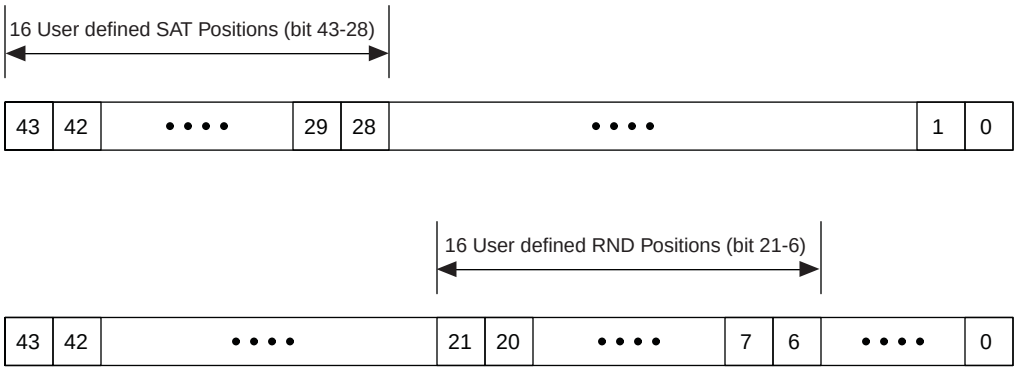
Table 4–8. Examples of Saturation

44- to 36-Bits Saturation	Symmetric SAT Result	Asymmetric SAT Result
5926AC01342h	7FFFFFFFh	7FFFFFFFh
ADA38D2210h	800000001h	800000000h

Stratix IV devices have up to 16 configurable bit positions out of the 44-bit bus (`[43:0]`) for the rounding and saturate logic unit providing higher flexibility. These 16-bit positions are located at bits `[21:6]` for rounding and `[43:28]` for saturation, as shown in Figure 4–21.

You must select the 16 configurable bit positions at compile time.

Figure 4–21. Rounding and Saturation Locations



For symmetric saturation, the RND bit position is also used to determine where the LSP for the saturated data is located.

You can use the rounding and saturation function described above in regular supported multiplication operations, as specified in Table 4–2 on page 4–8. However, for accumulation type operations, use the following convention:

The functionality of the round logic unit is in the format of:

Result = RND[S(A × B)], when used for an accumulation type of operation.

Likewise, the functionality of the saturation logic unit is in the format of:

Result = SAT[S(A × B)], when used for an accumulation type of operation.

If both the rounding and saturation logic units are used for an accumulation type of operation, the format is:

Result = SAT[RND[S(A × B)]]

DSP Block Control Signals

The Stratix IV DSP block is configured using a set of static and dynamic signals. You can configure the DSP block dynamic signals and can be set to toggle or not at run time. Table 4–9 lists the dynamic signals for the DSP block.

Table 4–9. DSP Block Dynamic Signals (Part 1 of 2)

Signal Name	Function	Count
■ `signa` ■ `signb`	Signed/unsigned control for all multipliers and adders. ■ `signa` for "multiplicand" input bus to `dataa[17:0]` each multiplier. ■ `signb` for "multiplier" input bus `datab[17:0]` to each multiplier. ■ `signa` = 1, `signb` = 1 for signed-signed multiplication ■ `signa` = 1, `signb` = 0 for signed-unsigned multiplication ■ `signa` = 0, `signb` = 1 for unsigned-signed multiplication ■ `signa` = 0, `signb` = 0 for unsigned-unsigned multiplication	2
`output_round`	Round control for first stage round and saturation block. ■ `output_round` = 1 for rounding on multiply output ■ `output_round` = 0 for normal multiply output	1
`chainout_round`	Round control for second stage round and saturation block. `chainout_round` = 1 for rounding multiply output `chainout_round` = 0 for normal multiply output	1
`output_saturate`	Saturation control for first stage round and saturation block for Q-format multiply. If both rounding and saturation is enabled, saturation is done on the rounded result. `output_saturate` = 1 for saturation support `output_saturate` = 0 for no saturation support	1
`chainout_saturate`	Saturation control for second stage round and saturation block for Q-format multiply. If both rounding and saturation is enabled, saturation is done on the rounded result. `chainout_saturate` = 1 for saturation support `chainout_saturate` = 0 for no saturation support	1

Table 4–9. DSP Block Dynamic Signals (Part 2 of 2)

Signal Name	Function	Count
`accum_sload`	Dynamically specifies whether the accumulator value is zero. `accum_sload` = 0, accumulation input is from the output registers `accum_sload` = 1, accumulation input is set to zero	1
`zero_chainout`	Dynamically specifies whether the chainout value is zero.	1
`zero_loopback`	Dynamically specifies whether the loopback value is zero.	1
`rotate`	`rotate` = 1, rotation feature is enabled	1
`shift_right`	`shift_right` = 1, shift right feature is enabled	1
Total Signals per Half Block		**11**

`clock0`
`clock1`
`clock2`
`clock3`
	DSP-block-wide clock signals.	4

`ena0`
`ena1`
`ena2`
`ena3`
	Input and Pipeline Register enable signals.	4

`aclr0`
`aclr1`
`aclr2`
`aclr3`
	DSP block-wide asynchronous clear signals (active low).	4

Total Count per Full Block		**34**

Software Support

Altera provides two distinct methods for implementing various modes of the DSP block in a design: instantiation and inference. Both methods use the following Quartus II megafunctions:

- `lpm_mult`
- `altmult_add`
- `altmult_accum`
- `altfp_mult`

To use the DSP block, instantiate the megafunctions in the Quartus II software. Alternatively, with inference, you can create an HDL design and synthesize it using a third-party synthesis tool (such as LeonardoSpectrum™, Synplify, or Quartus II Native Synthesis) that infers the appropriate megafunction by recognizing multipliers, multiplier adders, multiplier accumulators, and shift functions. Using either method, the Quartus II software maps the functionality to the DSP blocks during compilation.

For instructions about using these megafunctions and the MegaWizard Plug-In Manager, refer to the *Quartus II Software Help*.

For more information, refer to the "Synthesis" section in volume 1 of the *Quartus II Development Software Handbook*.

Document Revision History

Table 4–10 shows the revision history for this chapter.

Table 4–10. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
November 2009 v3.0	■ Updated Table 4–1. ■ Updated "Stratix IV Simplified DSP Operation" section. ■ Updated graphics. ■ Minor text edits.	—
June 2009 v2.3	■ Added an introductory paragraph to increase search ability. ■ Removed the Conclusion section.	—
April 2009 v2.2	■ Updated Table 4–1.	—
March 2009 v2.1	■ Updated Table 4–1. ■ Removed "Referenced Documents" section.	—
November 2008 v2.0	■ Updated Table 4–2. ■ Updated Figure 4–16. ■ Updated Figure 4–18.	—
May 2008 v1.0	Initial Release.	—

5. Clock Networks and PLLs in Stratix IV Devices

SIV51005-3.0

This chapter describes the hierarchical clock networks and phase-locked loops (PLLs) with advanced features in Stratix® IV devices. It includes details about the ability to reconfigure the PLL counter clock frequency and phase shift in real time, allowing you to sweep PLL output frequencies and dynamically adjust the output clock phase shift.

The Quartus® II software enables the PLLs and their features without external devices. The following sections describe the Stratix IV clock networks and PLLs in detail:

- "Clock Networks in Stratix IV Devices" on page 5–1
- "PLLs in Stratix IV Devices" on page 5–18

Clock Networks in Stratix IV Devices

The global clock networks (GCLKs), regional clock networks (RCLKs), and periphery clock networks (PCLKs) available in Stratix IV devices are organized into hierarchical clock structures that provide up to 236 unique clock domains (16 GCLKs + 88 RCLKs + 132 PCLKs) within the Stratix IV device and allow up to 71 unique GCLK, RCLK, and PCLK clock sources (16 GCLKs + 22 RCLKs + 33 PCLKs) per device quadrant. Table 5–1 lists the clock resources available in Stratix IV devices.

Table 5–1. Clock Resources in Stratix IV Devices

Clock Resource	Number of Resources Available	Source of Clock Resource
Clock input pins	32 Single-ended (16 Differential)	`CLK[0..15]p` and `CLK[0..15]n` pins
GCLK networks	16	`CLK[0..15]p` and `CLK[0..15]n` pins, PLL clock outputs, and logic array
RCLK networks	64/88 *(1)*	`CLK[0..15]p` and `CLK[0..15]n` pins, PLL clock outputs, and logic array
PCLK networks	56/88/112/132 (33 per device quadrant) *(2)*	DPA clock outputs, PLD-transceiver interface clocks, horizontal I/O pins, and logic array
GCLKs/RCLKs per quadrant	32/38 *(3)*	16 GCLKs + 16 RCLKs/ 16 GCLKs + 22 RCLKs
GCLKs/RCLKs per device	80/104 *(4)*	16 GCLKs + 64 RCLKs/ 16 GCLKs + 88 RCLKs

Notes to Table 5–1:

(1) There are 64 RCLKs in the EP4S40G2, EP4S100G2, EP4SE230, EP4SGX70, EP4SGX110, EP4SGX180, and EP4SGX230 devices. There are 88 RCLKs in the EP4S40G5, EP4S100G3, EP4S100G4, EP4S100G5, EP4SE360, EP4SE530, EP4SE820, EP4SGX290, EP4SGX360, and EP4SGX530 devices.

(2) There are 56 PCLKs in the EP4SGX70, and EP4SGX110 devices. There are 88 PCLKs in the EP4S40G2, EP4S100G2, EP4SE230, EP4SE360, EP4SGX180, EP4SGX230, EP4SGX290, and EP4SGX360 devices. There are 112 PCLKs in the EP4S40G5, EP4S100G3, EP4S100G4, EP4S100G5, EP4SE530 and EP4SGX530 devices. There are 132 PCLKs in the EP4SE820 device.

(3) There are 32 GCLKs/RCLKs per quadrant in the EP4S40G2, EP4S100G2, EP4SE230, EP4SGX70, EP4SGX110, EP4SGX180, and EP4SGX230 devices. There are 38 GCLKs/RCLKs per quadrant in the EP4S40G5, EP4S100G3, EP4S100G4, EP4S100G5, EP4SE360, EP4SE530, EP4SE820, EP4SGX290, EP4SGX360, and EP4SGX530 devices.

(4) There are 80 GCLKs/RCLKs per entire device in the EP4S40G2, EP4S100G2, EP4SE230, EP4SGX70, EP4SGX110, EP4SGX180, and EP4SGX230 devices. There are 104 GCLKs/RCLKS per entire device in the EP4S40G5, EP4S100G3, EP4S100G4, EP4S100G5, EP4SE360, EP4SE530, EP4SE820, EP4SGX290, EP4SGX360, and EP4SGX530 devices.

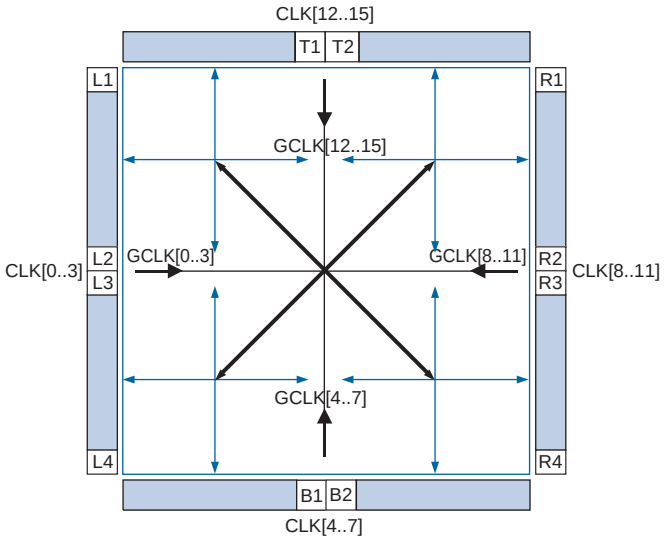
Stratix IV devices have up to 32 dedicated single-ended clock pins or 16 dedicated differential clock pins (`CLK[0..15]p` and `CLK[0..15]n`) that can drive either the GCLK or RCLK networks. These clock pins are arranged on the four sides of the Stratix IV device, as shown in Figure 5–1 through Figure 5–4 on page 5–4.

For more information about how to connect the clock input pins, refer to the *Stratix IV Device Family Pin Connection Guidelines*.

Global Clock Networks

Stratix IV devices provide up to 16 GCLKs that can drive throughout the device, serving as low-skew clock sources for functional blocks such as adaptive logic modules (ALMs), digital signal processing (DSP) blocks, TriMatrix memory blocks, and PLLs. Stratix IV device I/O elements (IOEs) and internal logic can also drive GCLKs to create internally generated global clocks and other high fan-out control signals; for example, synchronous or asynchronous clears and clock enables. Figure 5–1 shows the CLK pins and PLLs that can drive the GCLK networks in Stratix IV devices.

Figure 5–1. GCLK Networks



Regional Clock Networks

RCLK networks only pertain to the quadrant they drive into. RCLK networks provide the lowest clock delay and skew for logic contained within a single device quadrant. The Stratix IV device IOEs and internal logic within a given quadrant can also drive RCLKs to create internally generated regional clocks and other high fan-out control signals; for example, synchronous or asynchronous clears and clock enables.

Figure 5–2 through Figure 5–4 on page 5–4 show the CLK pins and PLLs that can drive the RCLK networks in Stratix IV devices.

Figure 5–2. RCLK Networks (EP4SE230, EP4SGX70, and EP4SGX110 Devices)

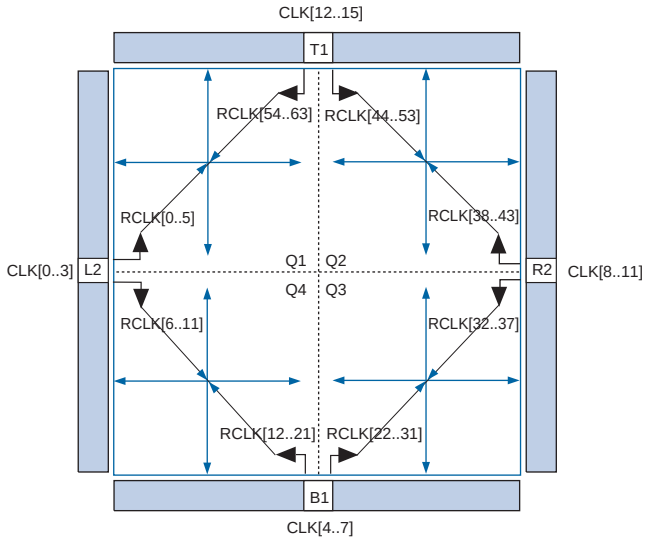


Figure 5–3. RCLK Networks (EP4S40G2, EP4S100G2, EP4SGX180, and EP4SGX230 Devices)

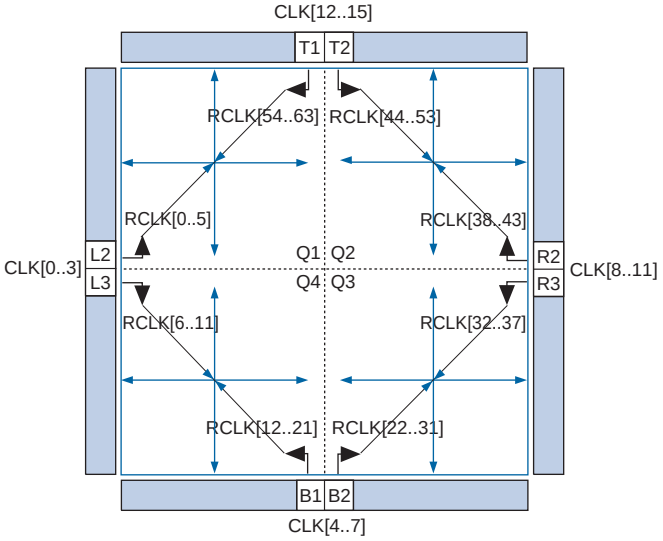
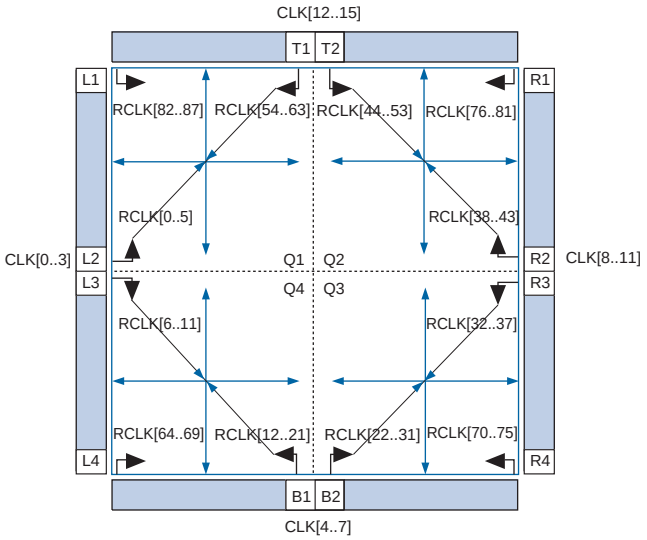


Figure 5–4. RCLK Networks (EP4S40G5, EP4S100G3, EP4S100G4, EP4S100G5, EP4SE360, EP4SE530, EP4SE820, EP4SGX290, EP4SGX360, and EP4SGX530 Devices) *(Note 1)*, *(2)*



Notes to Figure 5–4:

(1) The corner `RCLK[64..87]` can only be fed by their respective corner PLL outputs. For more details about connectivity, refer to Table 5–6 on page 5–12.

(2) EP4S40G5 and EP4SE360 devices have up to 8 PLLs. For more details about PLL availability, refer to Table 5–7 on page 5–18.

Periphery Clock Networks

PCLK networks shown in Figure 5–5 to Figure 5–8 on page 5–6 are collections of individual clock networks driven from the periphery of the Stratix IV device. Clock outputs from the dynamic phase aligner (DPA) block, programmable logic device (PLD)-transceiver interface clocks, horizontal I/O pins, and internal logic can drive the PCLK networks.

PCLKs have higher skew when compared with GCLK and RCLK networks. You can use PCLKs for general purpose routing to drive signals into and out of the Stratix IV device.

Legal clock sources for PCLK networks are clock outputs from the DPA block, PLD-transceiver interface clocks, horizontal I/O pins, and internal logic.

Figure 5–5. PCLK Networks (EP4SGX70 and EP4SGX110 Devices)

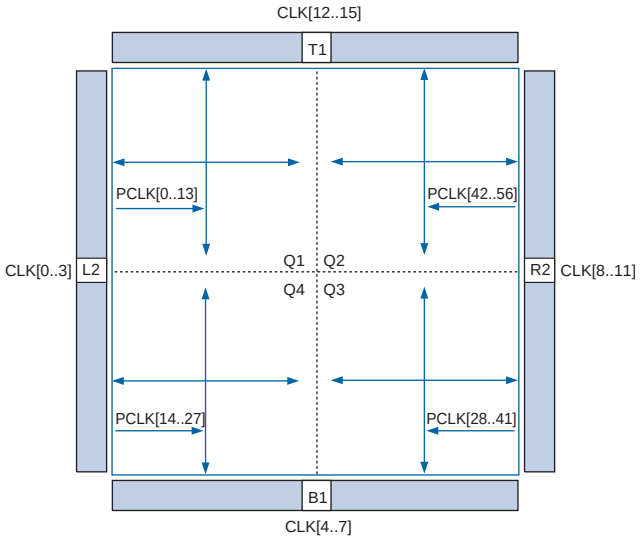
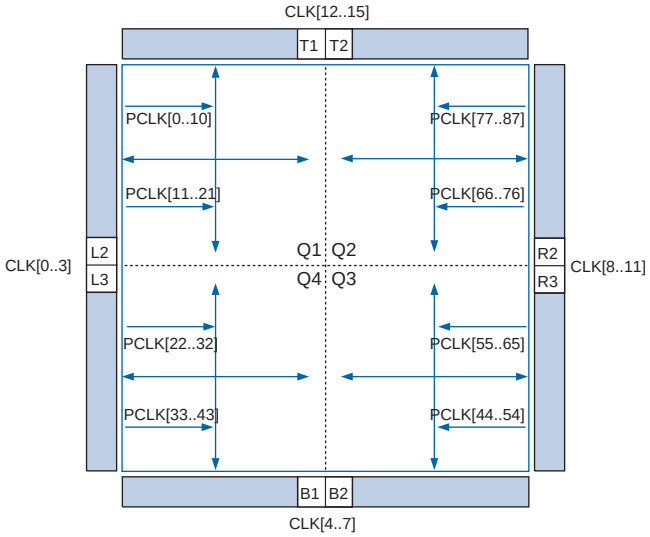


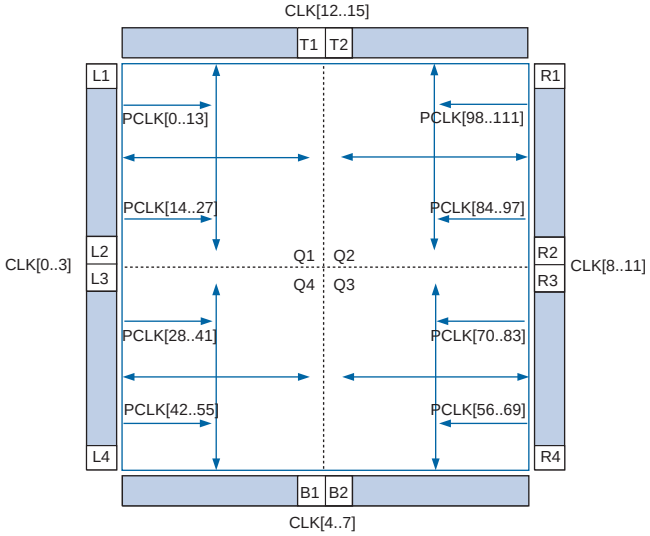
Figure 5–6. PCLK Networks (EP4S40G2, EP4S100G2, EP4SE230, EP4SE360, EP4SGX180, EP4SGX230, EP4SGX290, and EP4SGX360 Devices) *(Note 1)*



Note to Figure 5–6:

(1) EP4SE230 device has 4 PLLs. EP4SGX290 and EP4SGX360 devices have up to 12 PLLs. For more details about PLL availability, refer to Table 5–7 on page 5–18.

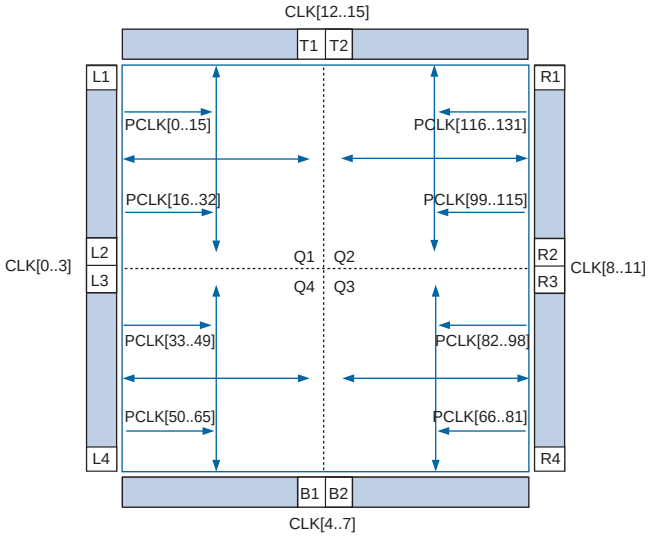
Figure 5–7. PCLK Networks (EP4S40G5, EP4S100G3, EP4S100G4, EP4S100G5, EP4SE530, and EP4SGX530 Devices) *(Note 1)*



Note to Figure 5–7:

(1) EP4S40G5 device has 8 PLLs. For more details about PLL availability, refer to Table 5–7 on page 5–18.

Figure 5–8. PCLK Networks (EP4SE820 Device)

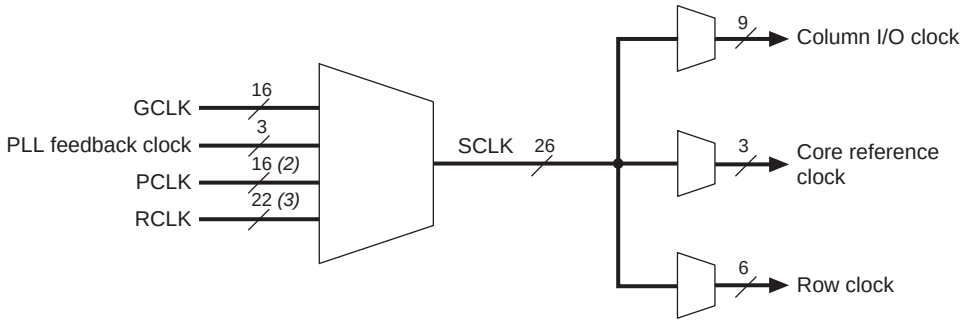


Clock Sources Per Region

There are 26 section clock (SCLK) networks available in each spine clock that can drive 6 row clocks in each logic array block (LAB) row, 9 column I/O clocks, and 3 core reference clocks. The SCLKs are the clock resources to the core functional blocks, PLLs, and I/O interfaces of the device. Figure 5–9 shows that the SCLKs can be driven by the GCLK, RCLK, PCLK, or the PLL feedback clock networks in each spine clock.

A spine clock is another layer of routing below the GCLKs, RCLKs, and PCLKs before each clock is connected to the clock routing for each LAB row. The settings for spine clocks are transparent to all users. The Quartus II software automatically routes the spine clock based on the GCLK, RCLK, and PCLKs.

Figure 5–9. Hierarchical Clock Networks per Spine Clock *(Note 1)*



Notes to Figure 5–9:

(1) The GCLK, RCLK, PCLK, and PLL feedback clocks share the same routing to the SCLKs. The total number of clock resources must not exceed the SCLK limits in each region to ensure successful design fitting in the Quartus II software.

(2) There are up to 16 PCLKs that can drive the SCLKs in each spine clock in the largest device.

(3) There are up to 22 RCLKs that can drive the SCLKs in each spine clock in the largest device.

Clock Regions

Stratix IV devices provide up to 104 distinct clock domains (16 GCLKs + 88 RCLKs) in the entire device. You can use these clock resources to form the following types of clock regions:

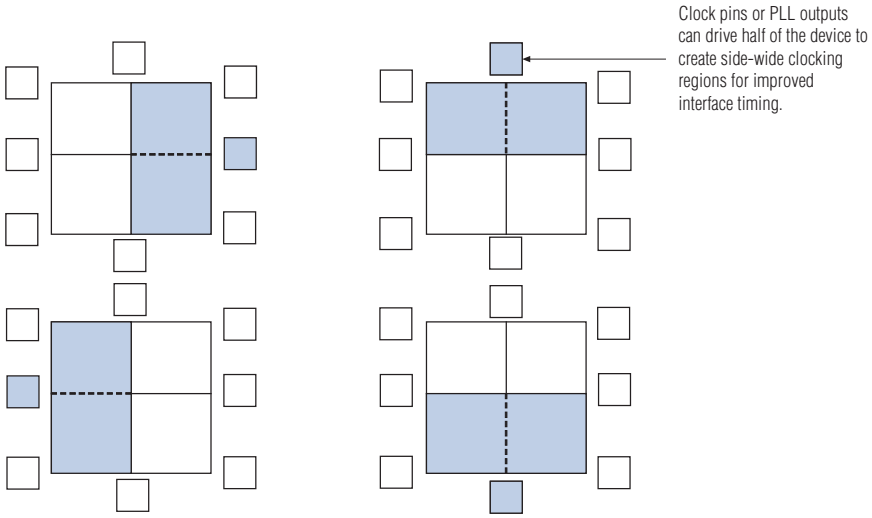
- Entire device
- Regional
- Dual-regional

To form the entire device clock region, a source (not necessarily a clock signal) drives a GCLK network that can be routed through the entire device. This clock region has the maximum delay when compared with other clock regions, but allows the signal to reach every destination within the device. This is a good option for routing global reset and clear signals or routing clocks throughout the device.

To form a RCLK region, a source drives a single quadrant of the device. This clock region provides the lowest skew within a quadrant and is a good option if all the destinations are within a single device quadrant.

To form a dual-regional clock region, a single source (a clock pin or PLL output) generates a dual-regional clock by driving two RCLK networks (one from each quadrant). This technique allows destinations across two device quadrants to use the same low-skew clock. The routing of this signal on an entire side has approximately the same delay as a RCLK region. Internal logic can also drive a dual-regional clock network. Corner PLL outputs only span one quadrant, they cannot generate a dual-regional clock network. Figure 5–10 shows the dual-regional clock region.

Figure 5–10. Stratix IV Dual-Regional Clock Region



Clock Network Sources

In Stratix IV devices, clock input pins, PLL outputs, and internal logic can drive the GCLK and RCLK networks. For the connectivity between dedicated pins `CLK[0..15]` and the GCLK and RCLK networks, refer to Table 5–2 and Table 5–3 on page 5–10.

Dedicated Clock Input Pins

CLK pins can be either differential clocks or single-ended clocks. Stratix IV devices support 16 differential clock inputs or 32 single-ended clock inputs. You can also use dedicated clock input pins `CLK[15..0]` for high fan-out control signals such as asynchronous clears, presets, and clock enables for protocol signals such as `TRDY` and `IRDY` for PCI through GCLK or RCLK networks.

LABs

You can drive each GCLK and RCLK network using LAB-routing to enable internal logic to drive a high fan-out, low-skew signal.

☞ Stratix IV PLLs cannot be driven by internally generated GCLKs or RCLKs. The input clock to the PLL has to come from dedicated clock input pins or pin/PLL-fed GCLKs or RCLKs.

PLL Clock Outputs

Stratix IV PLLs can drive both GCLK and RCLK networks, as described in Table 5–5 on page 5–11 and Table 5–6 on page 5–12.

Table 5–2 lists the connection between the dedicated clock input pins and GCLKs.

Table 5–2. Clock Input Pin Connectivity to the GCLK Networks (Part 1 of 2)

Clock Resources	CLK (p/n Pins)															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
GCLK0	✓	✓	✓	✓	—	—	—	—	—	—	—	—	—	—	—	—
GCLK1	✓	✓	✓	✓	—	—	—	—	—	—	—	—	—	—	—	—
GCLK2	✓	✓	✓	✓	—	—	—	—	—	—	—	—	—	—	—	—
GCLK3	✓	✓	✓	✓	—	—	—	—	—	—	—	—	—	—	—	—
GCLK4	—	—	—	—	✓	✓	✓	✓	—	—	—	—	—	—	—	—
GCLK5	—	—	—	—	✓	✓	✓	✓	—	—	—	—	—	—	—	—
GCLK6	—	—	—	—	✓	✓	✓	✓	—	—	—	—	—	—	—	—
GCLK7	—	—	—	—	✓	✓	✓	✓	—	—	—	—	—	—	—	—
GCLK8	—	—	—	—	—	—	—	—	✓	✓	✓	✓	—	—	—	—
GCLK9	—	—	—	—	—	—	—	—	✓	✓	✓	✓	—	—	—	—
GCLK10	—	—	—	—	—	—	—	—	✓	✓	✓	✓	—	—	—	—
GCLK11	—	—	—	—	—	—	—	—	✓	✓	✓	✓	—	—	—	—
GCLK12	—	—	—	—	—	—	—	—	—	—	—	—	✓	✓	✓	✓
GCLK13	—	—	—	—	—	—	—	—	—	—	—	—	✓	✓	✓	✓

Table 5–2. Clock Input Pin Connectivity to the GCLK Networks (Part 2 of 2)

Clock Resources	CLK (p/n Pins)															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
GCLK14	—	—	—	—	—	—	—	—	—	—	—	—	✓	✓	✓	✓
GCLK15	—	—	—	—	—	—	—	—	—	—	—	—	✓	✓	✓	✓

Table 5–3 lists the connectivity between the dedicated clock input pins and RCLKs in Stratix IV devices. A given clock input pin can drive two adjacent RCLK networks to create a dual-regional clock network.

Table 5–3. Clock Input Pin Connectivity to the RCLK Networks

Clock Resource	CLK (p/n Pins)															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RCLK [0, 4, 6, 10]	✓	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
RCLK [1, 5, 7, 11]	—	✓	—	—	—	—	—	—	—	—	—	—	—	—	—	—
RCLK [2, 8]	—	—	✓	—	—	—	—	—	—	—	—	—	—	—	—	—
RCLK [3, 9]	—	—	—	✓	—	—	—	—	—	—	—	—	—	—	—	—
RCLK [12, 16, 20, 22, 26, 30]	—	—	—	—	✓	—	—	—	—	—	—	—	—	—	—	—
RCLK [13, 17, 21, 23, 27, 31]	—	—	—	—	—	✓	—	—	—	—	—	—	—	—	—	—
RCLK [14, 18, 24, 28]	—	—	—	—	—	—	✓	—	—	—	—	—	—	—	—	—
RCLK [15, 19, 25, 29]	—	—	—	—	—	—	—	✓	—	—	—	—	—	—	—	—
RCLK [32, 36, 38, 42]	—	—	—	—	—	—	—	—	✓	—	—	—	—	—	—	—
RCLK [33, 37, 39, 43]	—	—	—	—	—	—	—	—	—	✓	—	—	—	—	—	—
RCLK [34, 40]	—	—	—	—	—	—	—	—	—	—	✓	—	—	—	—	—
RCLK [35, 41]	—	—	—	—	—	—	—	—	—	—	—	✓	—	—	—	—
RCLK [44, 48, 52, 54, 58, 62]	—	—	—	—	—	—	—	—	—	—	—	—	✓	—	—	—
RCLK [45, 49, 53, 55, 59, 63]	—	—	—	—	—	—	—	—	—	—	—	—	—	✓	—	—
RCLK [46, 50, 56, 60]	—	—	—	—	—	—	—	—	—	—	—	—	—	—	✓	—
RCLK [47, 51, 57, 61]	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	✓

Clock Input Connections to the PLLs

Table 5–4 lists the dedicated clock input pin connectivity to Stratix IV PLLs.

Table 5–4. Stratix IV Device PLLs and PLL Clock Pin Drivers

Dedicated Clock Input Pin CLK (p/n Pins)	PLL Number											
	L1	L2	L3	L4	B1	B2	R1	R2	R3	R4	T1	T2
CLK0	✓	✓	✓	✓	—	—	—	—	—	—	—	—
CLK1	✓	✓	✓	✓	—	—	—	—	—	—	—	—
CLK2	✓	✓	✓	✓	—	—	—	—	—	—	—	—
CLK3	✓	✓	✓	✓	—	—	—	—	—	—	—	—
CLK4	—	—	—	—	✓	✓	—	—	—	—	—	—
CLK5	—	—	—	—	✓	✓	—	—	—	—	—	—
CLK6	—	—	—	—	✓	✓	—	—	—	—	—	—
CLK7	—	—	—	—	✓	✓	—	—	—	—	—	—
CLK8	—	—	—	—	—	—	✓	✓	✓	✓	—	—
CLK9	—	—	—	—	—	—	✓	✓	✓	✓	—	—
CLK10	—	—	—	—	—	—	✓	✓	✓	✓	—	—
CLK11	—	—	—	—	—	—	✓	✓	✓	✓	—	—
CLK12	—	—	—	—	—	—	—	—	—	—	✓	✓
CLK13	—	—	—	—	—	—	—	—	—	—	✓	✓
CLK14	—	—	—	—	—	—	—	—	—	—	✓	✓
CLK15	—	—	—	—	—	—	—	—	—	—	✓	✓

Clock Output Connections

PLLs in Stratix IV devices can drive up to 20 RCLK networks and four GCLK networks. For Stratix IV PLL connectivity to GCLK networks, refer to Table 5–5. The Quartus II software automatically assigns PLL clock outputs to RCLK and GCLK networks.

Table 5–5 lists how the PLL clock outputs connect to the GCLK networks.

Table 5–5. Stratix IV PLL Connectivity to the GCLK Networks (Part 1 of 2)

Clock Network	PLL Number											
	L1	L2	L3	L4	B1	B2	R1	R2	R3	R4	T1	T2
GCLK0	✓	✓	✓	✓	—	—	—	—	—	—	—	—
GCLK1	✓	✓	✓	✓	—	—	—	—	—	—	—	—
GCLK2	✓	✓	✓	✓	—	—	—	—	—	—	—	—
GCLK3	✓	✓	✓	✓	—	—	—	—	—	—	—	—
GCLK4	—	—	—	—	✓	✓	—	—	—	—	—	—
GCLK5	—	—	—	—	✓	✓	—	—	—	—	—	—
GCLK6	—	—	—	—	✓	✓	—	—	—	—	—	—
GCLK7	—	—	—	—	✓	✓	—	—	—	—	—	—

Table 5–5. Stratix IV PLL Connectivity to the GCLK Networks (Part 2 of 2)

Clock Network	PLL Number											
	L1	L2	L3	L4	B1	B2	R1	R2	R3	R4	T1	T2
GCLK8	—	—	—	—	—	—	✓	✓	✓	✓	—	—
GCLK9	—	—	—	—	—	—	✓	✓	✓	✓	—	—
GCLK10	—	—	—	—	—	—	✓	✓	✓	✓	—	—
GCLK11	—	—	—	—	—	—	✓	✓	✓	✓	—	—
GCLK12	—	—	—	—	—	—	—	—	—	—	✓	✓
GCLK13	—	—	—	—	—	—	—	—	—	—	✓	✓
GCLK14	—	—	—	—	—	—	—	—	—	—	✓	✓
GCLK15	—	—	—	—	—	—	—	—	—	—	✓	✓

Table 5–6 lists how the PLL clock outputs connect to the RCLK networks.

Table 5–6. Stratix IV RCLK Outputs From the PLL Clock Outputs

Clock Resource	PLL Number											
	L1	L2	L3	L4	B1	B2	R1	R2	R3	R4	T1	T2
RCLK[0..11]	—	✓	✓	—	—	—	—	—	—	—	—	—
RCLK[12..31]	—	—	—	—	✓	✓	—	—	—	—	—	—
RCLK[32..43]	—	—	—	—	—	—	—	✓	✓	—	—	—
RCLK[44..63]	—	—	—	—	—	—	—	—	—	—	✓	✓
RCLK[64..69]	—	—	—	✓	—	—	—	—	—	—	—	—
RCLK[70..75]	—	—	—	—	—	—	—	—	—	✓	—	—
RCLK[76..81]	—	—	—	—	—	—	✓	—	—	—	—	—
RCLK[82..87]	✓	—	—	—	—	—	—	—	—	—	—	—

Clock Control Block

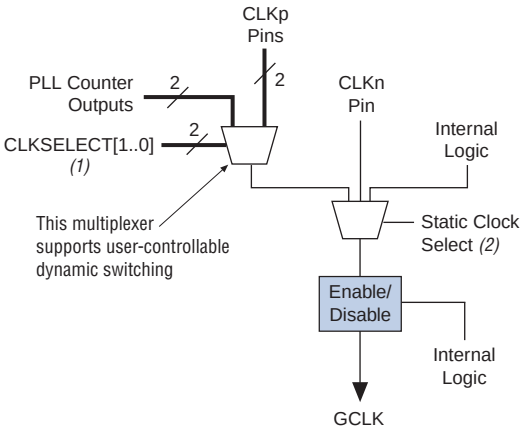
Every GCLK and RCLK network has its own clock control block. The control block provides the following features:

- Clock source selection (dynamic selection for GCLKs)
- Global clock multiplexing
- Clock power down (static or dynamic clock enable or disable)

Figure 5–11 and Figure 5–12 show the GCLK and RCLK select blocks, respectively.

You can select the clock source for the GCLK select block either statically or dynamically. You can statically select the clock source using a setting in the Quartus II software or you can dynamically select the clock source using internal logic to drive the multiplexer-select inputs. When selecting the clock source dynamically, you can select either PLL outputs (such as `C0` or `C1`) or a combination of clock pins or PLL outputs.

Figure 5–11. Stratix IV GCLK Control Block



Notes to Figure 5–11:

(1) When the device is operating in user mode, you can dynamically control the clock select signals through internal logic.

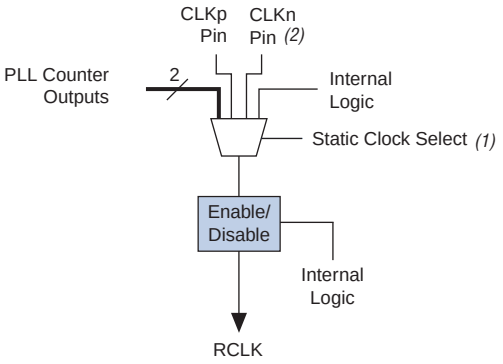
(2) When the device is operation in user mode, you can only set the clock select signals through a configuration file (SRAM object file [**.sof**] or programmer object file [**.pof**]) and cannot be dynamically controlled.

The mapping between the input clock pins, PLL counter outputs, and clock control block inputs is as follows:

- `inclk[0]` and `inclk[1]`—can be fed by any of the four dedicated clock pins on the same side of the Stratix IV device
- `inclk[2]`—can be fed by PLL counters C0 and C2 from the two center PLLs on the same side of the Stratix IV device
- `inclk[3]`—can be fed by PLL counters C1 and C3 from the two center PLLs on the same side of the Stratix IV device

The corner PLLs (L1, L4, R1, and R4) and the corresponding clock input pins (`PLL_L1_CLK` and so forth) do not support dynamic selection for the GCLK network. The clock source selection for the GCLK and RCLK networks from the corner PLLs (L1, L4, R1, and R4) and the corresponding clock input pins (`PLL_L1_CLK` and so forth) is controlled statically using configuration bit settings in the configuration file (**.sof** or **.pof**) generated by the Quartus II software.

Figure 5–12. RCLK Control Block



Notes to Figure 5–12:

(1) When the device is operation in user mode, you can only set the clock select signals through a configuration file (**.sof** or **.pof**) and cannot be dynamically controlled.

(2) The `CLKn` pin is not a dedicated clock input when used as a single-ended PLL clock input.

You can only control the clock source selection for the RCLK select block statically using configuration bit settings in the configuration file (**.sof** or **.pof**) generated by the Quartus II software.

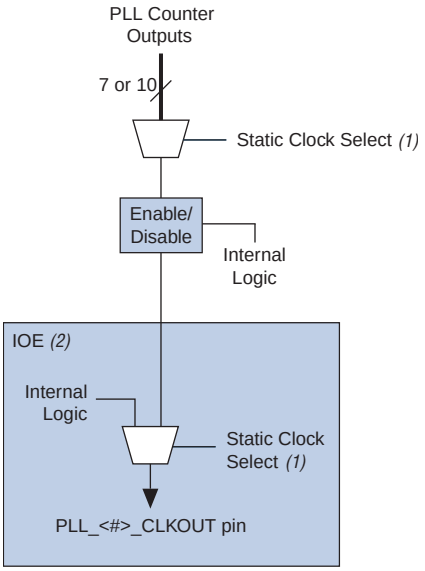
You can power down the Stratix IV clock networks using both static and dynamic approaches. When a clock network is powered down, all the logic fed by the clock network is in off-state, thereby reducing the overall power consumption of the device. The unused GCLK and RCLK networks are automatically powered down through configuration bit settings in the configuration file (**.sof** or **.pof**) generated by the Quartus II software. The dynamic clock enable or disable feature allows the internal logic to control power-up or power-down synchronously on the GCLK and RCLK networks, including dual-regional clock regions. This function is independent of the PLL and is applied directly on the clock network, as shown in Figure 5–11 and Figure 5–12.

You can set the input clock sources and the `clkena` signals for the GCLK and RCLK network multiplexers through the Quartus II software using the ALTCLKCTRL megafunction. You can also enable or disable the dedicated external clock output pins using the ALTCLKCTRL megafunction. Figure 5–13 shows the external PLL output clock control block.

When using the ALTCLKCTRL megafunction to implement dynamic clock source selection, the inputs from the clock pins feed the `inclk[0..1]` ports of the multiplexer, while the PLL outputs feed the `inclk[2..3]` ports. You can choose from among these inputs using the `CLKSELECT[1..0]` signal.

For more information, refer to the *ALTCLKCTRL Megafunction User Guide*.

Figure 5–13. Stratix IV External PLL Output Clock Control Block



Notes to Figure 5–13:

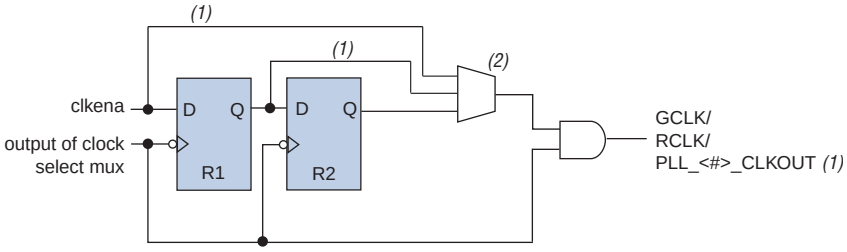
(1) When the device is operation in user mode, you can only set the clock select signals through a configuration file (**.sof** or **.pof**) and cannot be dynamically controlled.

(2) The clock control block feeds to a multiplexer within the `PLL_<#>_CLKOUT` pin's IOE. The `PLL_<#>_CLKOUT` pin is a dual-purpose pin. Therefore, this multiplexer selects either an internal signal or the output of the clock control block.

Clock Enable Signals

Figure 5–14 shows how the clock enable and disable circuit of the clock control block is implemented in Stratix IV devices.

Figure 5–14. clkena Implementation



Notes to Figure 5–14:

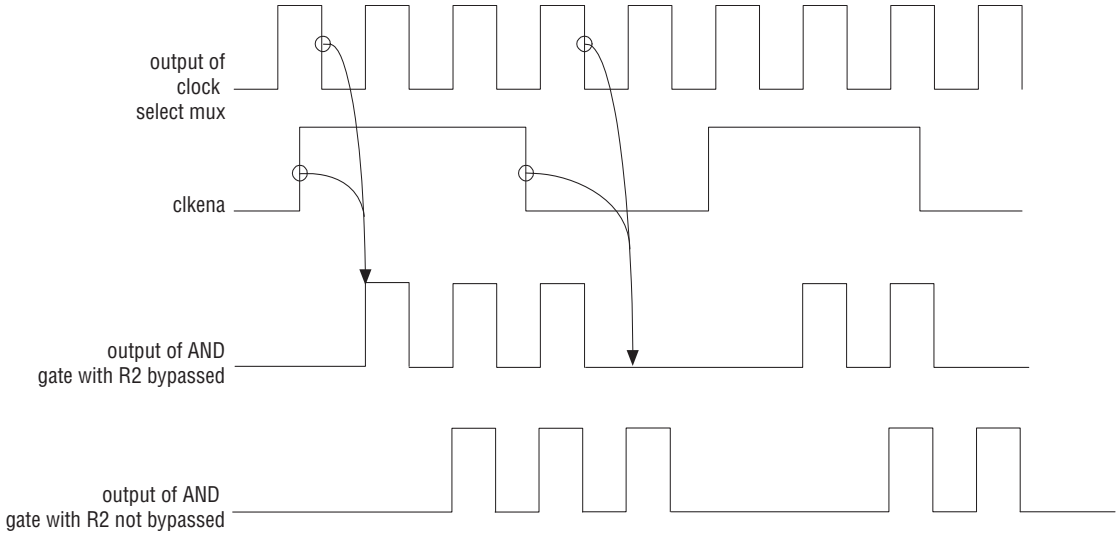
(1) The R1 and R2 bypass paths are not available for the PLL external clock outputs.

(2) The select line is statically controlled by a bit setting in the configuration file (**.sof** or **.pof**).

In Stratix IV devices, the `clkena` signals are supported at the clock network level instead of at the PLL output counter level. This allows you to gate off the clock even when you are not using a PLL. You can also use the `clkena` signals to control the dedicated external clocks from the PLLs. Figure 5–15 shows a waveform example for a clock output enable. `clkena` is synchronous to the falling edge of the clock output.

Stratix IV devices also have an additional metastability register that aids in asynchronous enable and disable of the GCLK and RCLK networks. You can optionally bypass this register in the Quartus II software.

Figure 5–15. clkena Signals *(Note 1)*



Note to Figure 5–15:

(1) You can use the `clkena` signals to enable or disable the GCLK and RCLK networks or the `PLL_<#>_CLKOUT` pins.

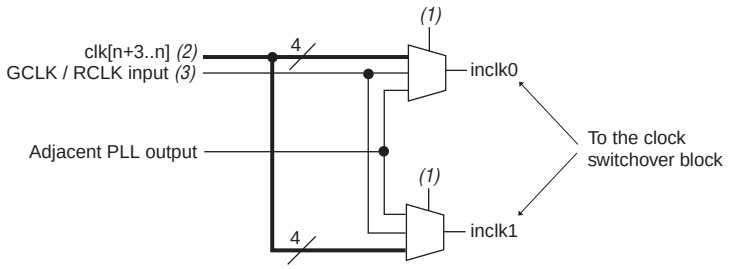
The PLL can remain locked independent of the `clkena` signals because the loop-related counters are not affected. This feature is useful for applications that require a low-power or sleep mode. The `clkena` signal can also disable clock outputs if the system is not tolerant of frequency over-shoot during resynchronization.

Clock Source Control for PLLs

The clock input to Stratix IV PLLs comes from clock input multiplexers. The clock multiplexer inputs come from dedicated clock input pins, PLLs through the GCLK and RCLK networks, or from dedicated connections between adjacent top/bottom and left/right PLLs. The clock input sources to top/bottom and left/right PLLs (L2, L3, T1, T2, B1, B2, R2, and R3) are shown in Figure 5–16; the corresponding clock input sources to left and right PLLs (L1, L4, R1, and R4) are shown in Figure 5–17.

The multiplexer select lines are only set in the configuration file (**.sof** or **.pof**). After programmed, this block cannot be changed without loading a new configuration file (**.sof** or **.pof**). The Quartus II software automatically sets the multiplexer select signals depending on the clock sources selected in the design.

Figure 5–16. Clock Input Multiplexer Logic for L2, L3, T1, T2, B1, B2, R2, and R3 PLLs



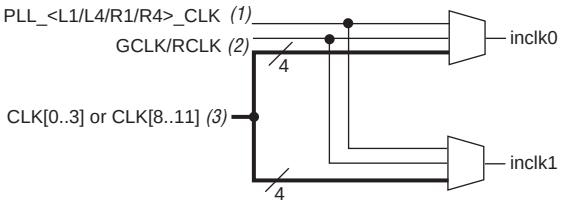
Notes to Figure 5–16:

(1) When the device is operating in user mode, input clock multiplexing is controlled through a configuration file (**.sof** or **.pof**) only and cannot be dynamically controlled.

(2) n=0 for L2 and L3 PLLs; n=4 for B1 and B2 PLLs; n=8 for R2 and R3 PLLs, and n=12 for T1 and T2 PLLs.

(3) You can drive the GCLK or RCLK input using an output from another PLL, a pin-driven GCLK or RCLK, or through a clock control block provided the clock control block is fed by an output from another PLL or a pin-driven dedicated GCLK or RCLK. An internally generated global signal or general purpose I/O pin cannot drive the PLL.

Figure 5–17. Clock Input Multiplexer Logic for L1, L4, R1, and R4 PLLs



Notes to Figure 5–17:

(1) Dedicated clock input pins to the PLLs are L1, L4, R1, and R4, respectively. For example, `PLL_L1_CLK` is the dedicated clock input for `PLL_L1`.

(2) You can drive the GCLK or RCLK input using an output from another PLL, a pin-driven GCLK or RCLK, or through a clock control block provided the clock control block is fed by an output from another PLL or a pin-driven dedicated GCLK or RCLK. An internally generated global signal or general purpose I/O pin cannot drive the PLL.

(3) The center clock pins can feed the corner PLLs on the same side directly through a dedicated path. However, these paths may not be fully compensated.

Cascading PLLs

You can cascade the left/right and top/bottom PLLs through the GCLK and RCLK networks. In addition, where two left/right or top/bottom PLLs exist next to each other, there is a direct connection between them that does not require the GCLK or RCLK network. Using this path reduces clock jitter when cascading PLLs.

Stratix IV GX devices allow cascading the left and right PLLs to transceiver PLLs (CMU PLLs and receiver CDRs).

For more information, refer to the "FPGA Fabric PLLs -Transceiver PLLs Cascading" section in the *Stratix IV Transceiver Clocking* chapter.

When cascading PLLs in Stratix IV devices, the source (upstream) PLL must have a low-bandwidth setting while the destination (downstream) PLL must have a high-bandwidth setting. Ensure that there is no overlap of the bandwidth ranges of the two PLLs.

For more information about PLL cascading in external memory interfaces designs, refer to the *External Memory PHY Interface Megafunction User Guide (ALTMEMPHY)*.

PLLs in Stratix IV Devices

Stratix IV devices offer up to 12 PLLs that provide robust clock management and synthesis for device clock management, external system clock management, and high-speed I/O interfaces. The nomenclature for the PLLs follows their geographical location in the device floor plan. The PLLs that reside on the top and bottom sides of the device are named `PLL_T1`, `PLL_T2`, `PLL_B1` and `PLL_B2`; the PLLs that reside on the left and right sides of the device are named `PLL_L1`, `PLL_L2`, `PLL_L3`, `PLL_L4`, `PLL_R1`, `PLL_R2`, `PLL_R3`, and `PLL_R4`.

Table 5–7 lists the number of PLLs available in the Stratix IV device family.

Table 5–7. Stratix IV Device PLL Availability (Part 1 of 2)

Device	Package	L1	L2	L3	L4	T1	T2	B1	B2	R1	R2	R3	R4
EP4S40G2	F1517	—	✓	✓	—	✓	✓	✓	✓	—	✓	✓	—
EP4S40G5	H1517	—	✓	✓	—	✓	✓	✓	✓	—	✓	✓	—
EP4S100G2	F1517	—	✓	✓	—	✓	✓	✓	✓	—	✓	✓	—
EP4S100G3	F1932	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
EP4S100G4	F1932	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
EP4S100G5	H1517	—	✓	✓	—	✓	✓	✓	✓	—	✓	✓	—
	F1932	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
EP4SE230	F780	—	✓	—	—	✓	—	✓	—	—	✓	—	—
EP4SE360	H780	—	✓	—	—	✓	—	✓	—	—	✓	—	—
	F1152	—	✓	✓	—	✓	✓	✓	✓	—	✓	✓	—
EP4SE530	H1152	—	✓	✓	—	✓	✓	✓	✓	—	✓	✓	—
	H1517	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	F1760	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
EP4SE820	H1152	—	✓	✓	—	✓	✓	✓	✓	—	✓	✓	—
	H1517	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	F1760	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
EP4SGX70	F780	—	✓	—	—	✓	—	✓	—	—	—	—	—
	F1152	—	✓	—	—	✓	—	✓	—	—	✓	—	—
EP4SGX110	F780	—	✓	—	—	✓	—	✓	—	—	—	—	—
	F1152	—	✓	—	—	✓	—	✓	—	—	✓	—	—
EP4SGX180	F780	—	✓	—	—	✓	—	✓	—	—	—	—	—
	F1152	—	✓	—	—	✓	✓	✓	✓	—	✓	—	—
	F1517	—	✓	✓	—	✓	✓	✓	✓	—	✓	✓	—
EP4SGX230	F780	—	✓	—	—	✓	—	✓	—	—	—	—	—
	F1152	—	✓	—	—	✓	✓	✓	✓	—	✓	—	—
	F1517	—	✓	✓	—	✓	✓	✓	✓	—	✓	✓	—

Table 5–7. Stratix IV Device PLL Availability (Part 2 of 2)

Device	Package	L1	L2	L3	L4	T1	T2	B1	B2	R1	R2	R3	R4
EP4SGX290	H780	—	—	—	—	✓	✓	✓	✓	—	—	—	—
	F1152	—	✓	—	—	✓	✓	✓	✓	—	✓	—	—
	F1517	—	✓	✓	—	✓	✓	✓	✓	—	✓	✓	—
	F1760	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	F1932	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
EP4SGX360	H780	—	—	—	—	✓	✓	✓	✓	—	—	—	—
	F1152	—	✓	—	—	✓	✓	✓	✓	—	✓	—	—
	F1517	—	✓	✓	—	✓	✓	✓	✓	—	✓	✓	—
	F1760	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	F1932	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
EP4SGX530	H1152	—	✓	—	—	✓	✓	✓	✓	—	✓	—	—
	H1517	—	✓	✓	—	✓	✓	✓	✓	—	✓	✓	—
	F1760	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	F1932	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

All Stratix IV PLLs have the same core analog structure with only minor differences in the features that are supported. Table 5–8 lists the features of top/bottom and left/right PLLs in Stratix IV devices.

Table 5–8. Stratix IV PLL Features (Part 1 of 2)

Feature	Stratix IV Top/Bottom PLLs	Stratix IV Left/Right PLLs
C (output) counters	10	7
M, N, C counter sizes	1 to 512	1 to 512
Dedicated clock outputs	6 single-ended or 4 single-ended and 1 differential pair	2 single-ended or 1 differential pair
Clock input pins	4 single-ended or 2 differential pin pairs	4 single-ended or 2 differential pin pairs
External feedback input pin	Single-ended or differential	Single-ended only
Spread-spectrum input clock tracking	Yes *(1)*	Yes *(1)*
PLL cascading	Through GCLK and RCLK and a dedicated path between adjacent PLLs	Through GCLK and RCLK and dedicated path between adjacent PLLs *(2)*
Compensation modes	All except LVDS clock network compensation	All except external feedback mode when using differential I/Os
PLL drives LVDSCLK and LOADEN	No	Yes
VCO output drives the DPA clock	No	Yes
Phase shift resolution	Down to 96.125 ps *(3)*	Down to 96.125 ps *(3)*
Programmable duty cycle	Yes	Yes
Output counter cascading	Yes	Yes

Table 5–8. Stratix IV PLL Features (Part 2 of 2)

Feature	Stratix IV Top/Bottom PLLs	Stratix IV Left/Right PLLs
Input clock switchover	Yes	Yes

Notes to Table 5–8:

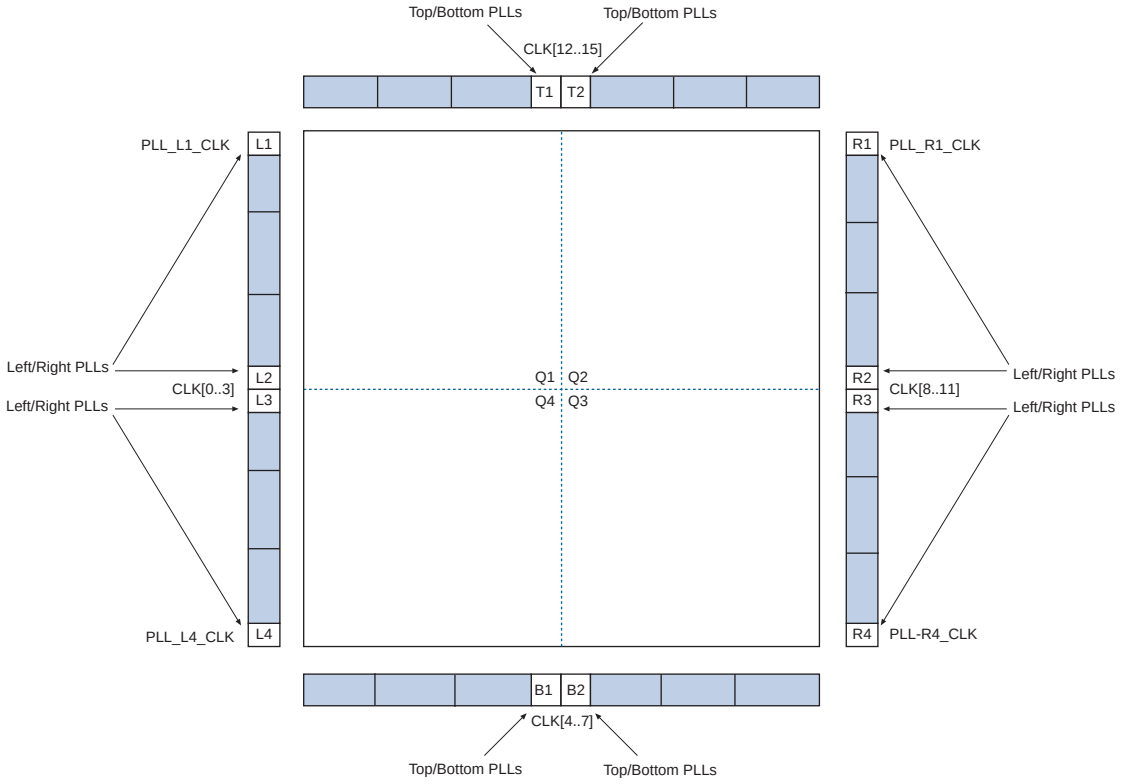
(1) Provided input clock jitter is within input jitter tolerance specifications.

(2) The dedicated path between adjacent PLLs is not available on L1, L4, R1, and R4 PLLs.

(3) The smallest phase shift is determined by the voltage-controlled oscillator (VCO) period divided by eight. For degree increments, the Stratix IV device can shift all output frequencies in increments of at least 45°. Smaller degree increments are possible depending on the frequency and divide parameters.

Figure 5–18 shows the location of PLLs in Stratix IV devices.

Figure 5–18. Stratix IV PLL Locations



Stratix IV PLL Hardware Overview

Stratix IV devices contain up to 12 PLLs with advanced clock management features. The goal of a PLL is to synchronize the phase and frequency of an internal or external clock to an input reference clock. There are a number of components that comprise a PLL to achieve this phase alignment.

Stratix IV PLLs align the rising edge of the input reference clock to a feedback clock using the phase-frequency detector (PFD). The falling edges are determined by the duty-cycle specifications. The PFD produces an up or down signal that determines whether the VCO needs to operate at a higher or lower frequency. The output of the PFD feeds the charge pump and loop filter, which produces a control voltage for

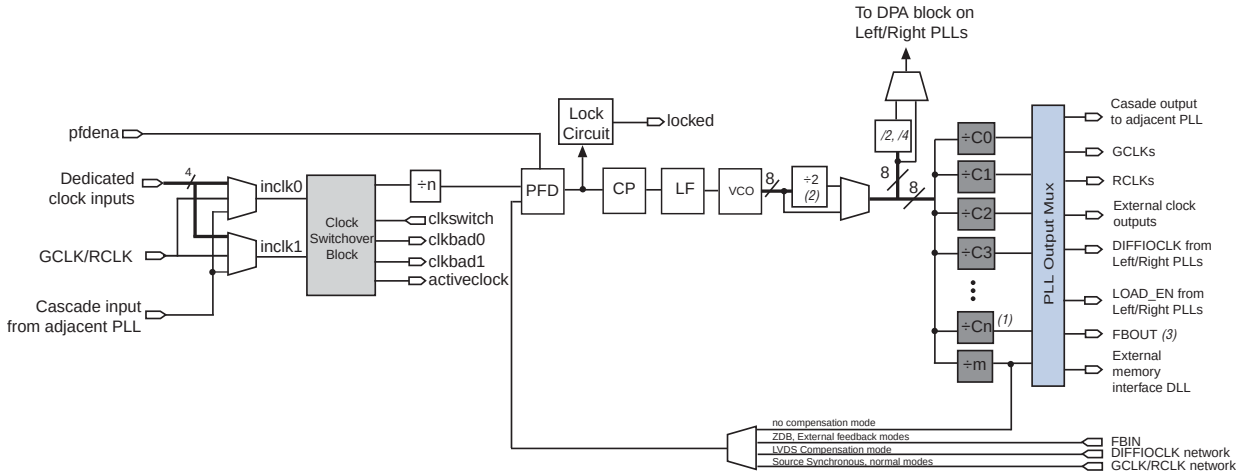
setting the VCO frequency. If the PFD produces an up signal, the VCO frequency increases. A down signal decreases the VCO frequency. The PFD outputs these up and down signals to a charge pump. If the charge pump receives an up signal, current is driven into the loop filter. Conversely, if the charge pump receives a down signal, current is drawn from the loop filter.

The loop filter converts these up and down signals to a voltage that is used to bias the VCO. The loop filter also removes glitches from the charge pump and prevents voltage over-shoot, which filters the jitter on the VCO. The voltage from the loop filter determines how fast the VCO operates. A divide counter (`m`) is inserted in the feedback loop to increase the VCO frequency above the input reference frequency. VCO frequency (f_{VCO}) is equal to (`m`) times the input reference clock (f_{REF}). The input reference clock (f_{REF}) to the PFD is equal to the input clock (f_{IN}) divided by the pre-scale counter (`N`). Therefore, the feedback clock (f_{FB}) applied to one input of the PFD is locked to the f_{REF} that is applied to the other input of the PFD.

The VCO output from the left and right PLLs can feed seven post-scale counters (`C[0..6]`), while the corresponding VCO output from the top and bottom PLLs can feed ten post-scale counters (`C[0..9]`). These post-scale counters allow a number of harmonically related frequencies to be produced by the PLL.

Figure 5–19 shows a simplified block diagram of the major components of the Stratix IV PLL.

Figure 5–19. Stratix IV PLL Block Diagram



Notes to Figure 5–19:

(1) The number of post-scale counters is seven for left and right PLLs and ten for top and bottom PLLs.

(2) This is the VCO post-scale counter `K`.

(3) The `FBOUT` port is fed by the `M` counter in Stratix IV PLLs.

You can drive the GCLK or RCLK inputs using an output from another PLL, a pin-driven GCLK or RCLK, or through a clock control block provided the clock control block is fed by an output from another PLL or a pin-driven dedicated GCLK or RCLK. An internally generated global signal or general purpose I/O pin cannot drive the PLL.

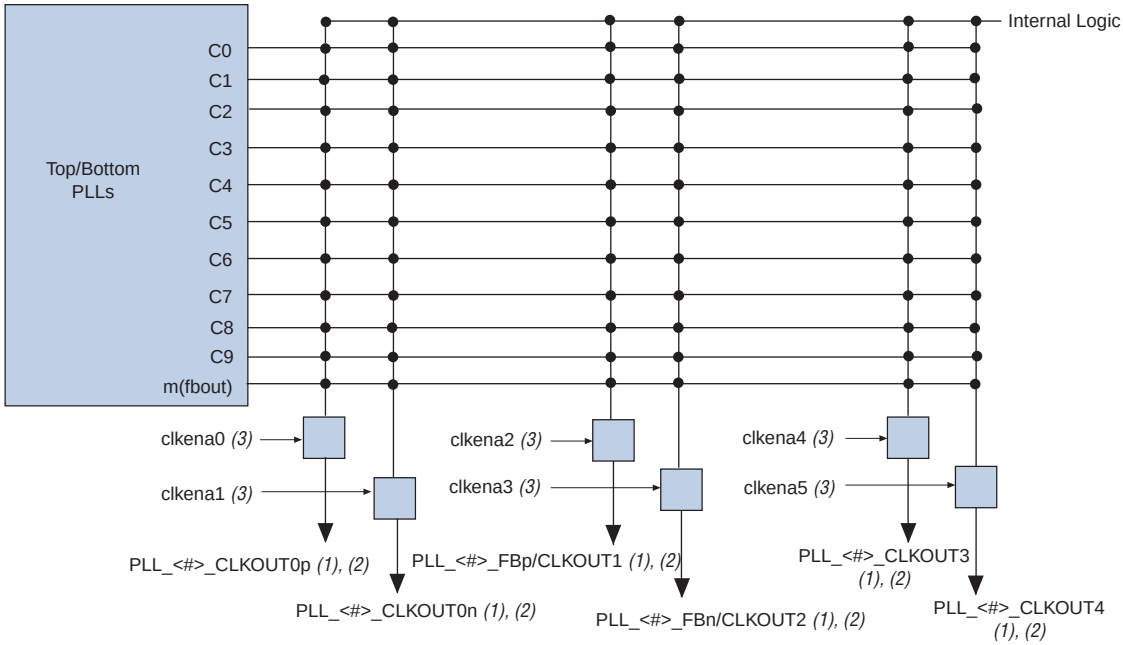
PLL Clock I/O Pins

Each top and bottom PLL supports six clock I/O pins, organized as three pairs of pins:

- 1st pair—two single-ended I/O or one differential I/O
- 2nd pair—two single-ended I/O or one differential external feedback input (FBp/FBn)
- 3rd pair—two single-ended I/O or one differential input

Figure 5–20 shows the clock I/O pins associated with the top and bottom PLLs.

Figure 5–20. External Clock Outputs for Top and Bottom PLLs



Notes to Figure 5–20:

(1) You can feed these clock output pins using any one of the `C[9..0]`, m counters.

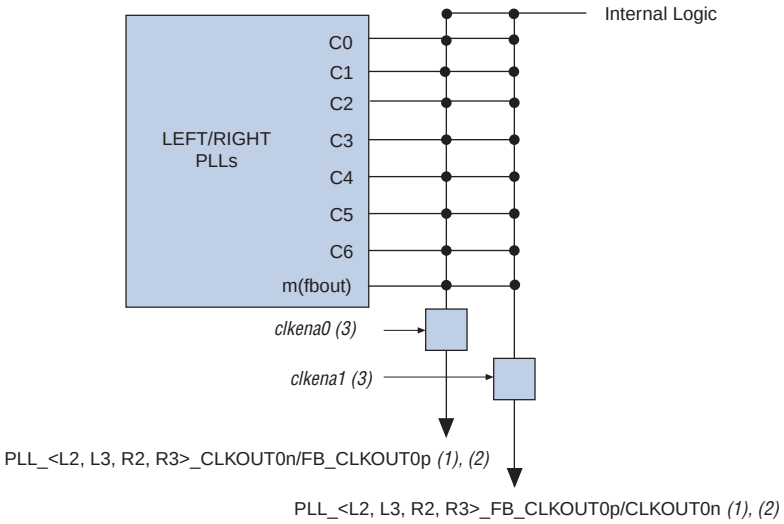
(2) The `CLKOUT0p` and `CLKOUT0n` pins can be either single-ended or differential clock outputs. The `CLKOUT1` and `CLKOUT2` pins are dual-purpose I/O pins that you can use as two single-ended outputs or one differential external feedback input pin. The `CLKOUT3` and `CLKOUT4` pins are two single-ended output pins.

(3) These external clock enable signals are available only when using the ALTCLKCTRL megafunction.

Any of the output counters (`C[9..0]` on the top and bottom PLLs and `C[6..0]` on the left and right PLLs) or the `M` counter can feed the dedicated external clock outputs, as shown in Figure 5–20 and Figure 5–21. Therefore, one counter or frequency can drive all output pins available from a given PLL.

Each left and right PLL supports two clock I/O pins, configured as either two single-ended I/Os or one differential I/O pair. When using both pins as single-ended I/Os, one of them can be the clock output while the other pin is the external feedback input (FB) pin. Therefore, for single-ended I/O standards, the left and right PLLs only support external feedback mode.

Figure 5–21. External Clock Outputs for Left and Right PLLs



Notes to Figure 5–21:

(1) You can feed these clock output pins using any one of the `C[6..0]`, m counters.

(2) The `CLKOUT0p` and `CLKOUT0n` pins are dual-purpose I/O pins that you can use as two single-ended outputs or one single-ended output and one external feedback input pin.

(3) These external clock enable signals are available only when using the ALTCLKCTRL megafunction.

Each pin of a single-ended output pair can either be in-phase or 180° out-of-phase. The Quartus II software places the NOT gate in the design into the IOE to implement the 180° phase with respect to the other pin in the pair. The clock output pin pairs support the same I/O standards as standard output pins (in the top and bottom banks) as well as LVDS, LVPECL, differential High-Speed Transceiver Logic (HSTL), and differential SSTL.

To determine which I/O standards are supported by the PLL clock input and output pins, refer to the *I/O Features in Stratix IV Devices* chapter.

Stratix IV PLLs can also drive out to any regular I/O pin through the GCLK or RCLK network. You can also use the external clock output pins as user I/O pins if you do not need external PLL clocking.

PLL Control Signals

You can use the three signals, `pfdena`, `areset`, and `locked`, to observe and control PLL operation and resynchronization.

pfdena

Use the `pfdena` signal to maintain the most recent locked frequency so your system has time to store its current settings before shutting down. The `pfdena` signal controls the PFD output with a programmable gate. If you disable PFD, the VCO operates at its most recent set value of control voltage and frequency, with some long-term drift to a lower frequency. The PLL continues running even if it goes out-of-lock or the input clock is disabled. You can use either your own control signal or the control signals available from the clock switchover circuit (`activeclock`, `clkbad[0]`, or `clkbad[1]`) to control `pfdena`.

areset

The `areset` signal is the reset or resynchronization input for each PLL. The device input pins or internal logic can drive these input signals. When `areset` is driven high, the PLL counters reset, clearing the PLL output and placing the PLL out-of-lock. The VCO is then set back to its nominal setting. When `areset` is driven low again, the PLL resynchronizes to its input as it re-locks.

You must assert the `areset` signal every time the PLL loses lock to guarantee the correct phase relationship between the PLL input and output clocks. You can set up the PLL to automatically reset (self reset) upon a loss-of-lock condition using the Quartus II MegaWizard™ Plug-In Manager. You must include the `areset` signal in designs if either of the following conditions is true:

- PLL reconfiguration or clock switchover is enabled in the design
- Phase relationships between the PLL input and output clocks need to be maintained after a loss-of-lock condition

☞ If the input clock to the PLL is not toggling or is unstable after power up, assert the `areset` signal after the input clock is stable and within specifications.

locked

The `locked` signal output of the PLL indicates that the PLL has locked onto the reference clock and the PLL clock outputs are operating at the desired phase and frequency set in the Quartus II MegaWizard Plug-In Manager. The lock detection circuit provides a signal to the core logic that gives an indication when the feedback clock has locked onto the reference clock both in phase and frequency.

☞ Altera recommends using the `areset` and `locked` signals in your designs to control and observe the status of your PLL.

Clock Feedback Modes

Stratix IV PLLs support up to six different clock feedback modes. Each mode allows clock multiplication and division, phase shifting, and programmable duty cycle. Table 5–9 lists the clock feedback modes supported by the Stratix IV device PLLs.

Table 5–9. Clock Feedback Mode Availability

Clock Feedback Mode	Availability	
	Top/Bottom PLLs	Left/Right PLLs
Source-synchronous	Yes	Yes
No-compensation	Yes	Yes
Normal	Yes	Yes
Zero-delay buffer (ZDB)	Yes	Yes
External feedback	Yes	Yes *(2)*
LVDS compensation	No	Yes

Notes to Table 5–9:

(1) The high-bandwidth PLL setting is not supported in the external feedback mode.

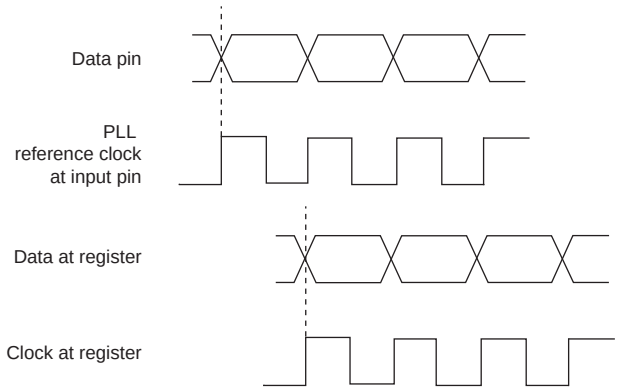
(2) External feedback mode is supported for single-ended inputs and outputs only on the left and right PLLs.

The input and output delays are fully compensated by a PLL only when using the dedicated clock input pins associated with a given PLL as the clock source. For example, when using `PLL_T1` in normal mode, the clock delays from the input pin to the PLL clock output-to-destination register are fully compensated, provided the clock input pin is one of the following four pins: `CLK12`, `CLK13`, `CLK14`, or `CLK15`. When an RCLK or GCLK network drives the PLL, the input and output delays may not be fully compensated in the Quartus II software. Another example is when you configure `PLL_T2` in zero-delay buffer mode and the PLL input is driven by a dedicated clock input pin, a fully compensated clock path results in zero-delay between the clock input and one of the output clocks from the PLL. If the PLL input is instead fed by a non-dedicated input (using the GCLK network), the output clock may not be perfectly aligned with the input clock.

Source Synchronous Mode

If data and clock arrive at the same time on the input pins, the same phase relationship is maintained at the clock and data ports of any IOE input register. Figure 5–22 shows an example waveform of the clock and data in this mode. Altera recommends source synchronous mode for source-synchronous data transfers. Data and clock signals at the IOE experience similar buffer delays as long as you use the same I/O standard.

Figure 5–22. Phase Relationship Between Clock and Data in Source-Synchronous Mode



Source-synchronous mode compensates for the delay of the clock network used plus any difference in the delay between these two paths:

- Data pin to the IOE register input
- Clock input pin to the PLL PFD input

The Stratix IV PLL can compensate multiple pad-to-input-register paths, such as a data bus when it is set to use source-synchronous compensation mode. You can use the "PLL Compensation" assignment in the Quartus II software Assignment Editor to select which input pins are used as the PLL compensation targets. You can include your entire data bus, provided the input registers are clocked by the same output of a source-synchronous-compensated PLL. In order for the clock delay to be properly compensated, all of the input pins must be on the same side of the device. The PLL compensates for the input pin with the longest pad-to-register delay among all input pins in the compensated bus.

If you do not make the "PLL Compensation" assignment, the Quartus II software automatically selects all of the pins driven by the compensated output of the PLL as the compensation target.

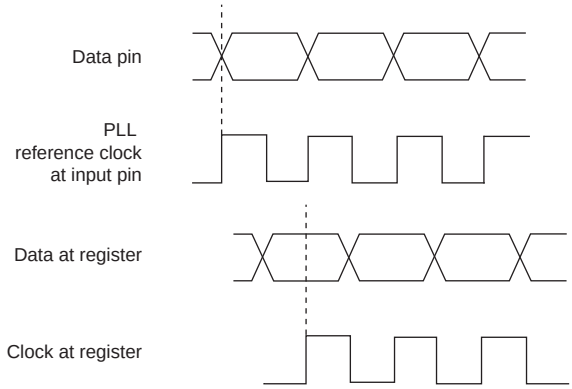
Source-Synchronous Mode for LVDS Compensation

The goal of source-synchronous mode is to maintain the same data and clock timing relationship seen at the pins of the internal serializer/deserializer (SERDES) capture register, except that the clock is inverted (180° phase shift). Thus, source-synchronous mode ideally compensates for the delay of the LVDS clock network plus any difference in delay between these two paths:

- Data pin-to-SERDES capture register
- Clock input pin-to-SERDES capture register. In addition, the output counter must provide the 180° phase shift

Figure 5–23 shows an example waveform of the clock and data in LVDS mode.

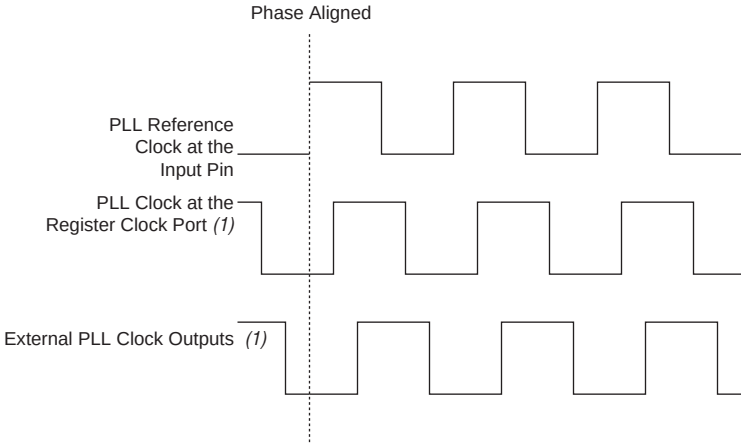
Figure 5–23. Phase Relationship Between the Clock and Data in LVDS Mode



No-Compensation Mode

In no-compensation mode, the PLL does not compensate for any clock networks. This mode provides better jitter performance because the clock feedback into the PFD passes through less circuitry. Both the PLL internal- and external-clock outputs are phase-shifted with respect to the PLL clock input. Figure 5–24 shows an example waveform of the PLL clocks' phase relationship in no-compensation mode.

Figure 5–24. Phase Relationship Between the PLL Clocks in No Compensation Mode



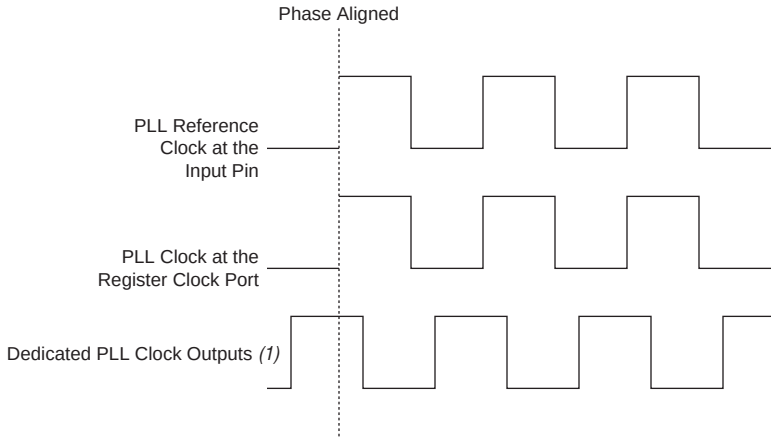
Note to Figure 5–24

(1) The PLL clock outputs lags the PLL input clocks depending on routine delays.

Normal Mode

An internal clock in normal mode is phase-aligned to the input clock pin. The external clock-output pin has a phase delay relative to the clock input pin if connected in this mode. The Quartus II software timing analyzer reports any phase difference between the two. In normal mode, the delay introduced by the GCLK or RCLK network is fully compensated. Figure 5–25 shows an example waveform of the PLL clocks' phase relationship in normal mode.

Figure 5–25. Phase Relationship Between the PLL Clocks in Normal Mode



Note to Figure 5–25:

(1) The external clock output can lead or lag the PLL internal clock signals.

Zero-Delay Buffer Mode

In ZDB mode, the external clock output pin is phase-aligned with the clock input pin for zero-delay through the device. When using this mode, you must use the same I/O standard on the input clocks and output clocks in order to guarantee clock alignment at the input and output pins. ZDB mode is supported on all Stratix IV PLLs.

When using Stratix IV PLLs in ZDB mode, along with single-ended I/O standards, to ensure phase alignment between the CLK pin and the external clock output (`CLKOUT`) pin, you must instantiate a bi-directional I/O pin in the design to serve as the feedback path connecting the `FBOUT` and `FBIN` ports of the PLL. The PLL uses this bi-directional I/O pin to mimic, and compensate for, the output delay from the clock output port of the PLL to the external clock output pin. Figure 5–26 shows ZDB mode in Stratix IV PLLs. When using ZDB mode, you cannot use differential I/O standards on the PLL clock input or output pins.

☞ The bi-directional I/O pin that you instantiate in your design must always be assigned a single-ended I/O standard.

☞ When using ZDB mode, to avoid signal reflection, do not place board traces on the bi-directional I/O pin.

Figure 5–26. ZDB Mode in Stratix IV PLLs

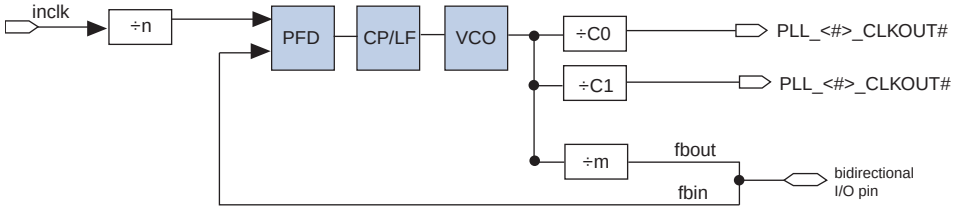
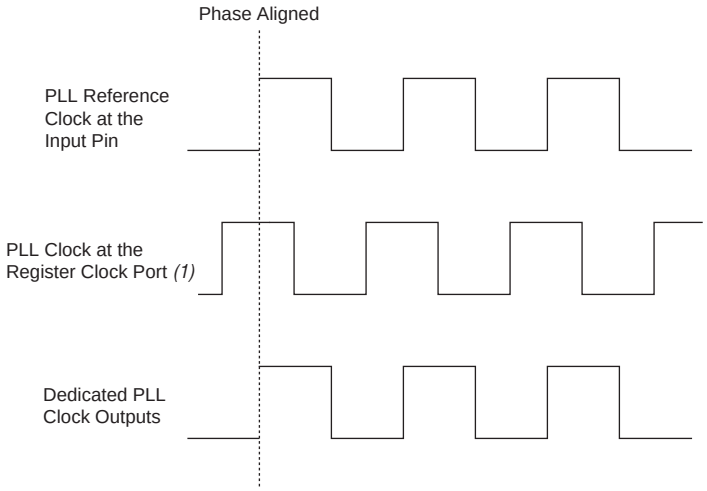


Figure 5–27 shows an example waveform of the PLL clocks' phase relationship in ZDB mode.

Figure 5–27. Phase Relationship Between the PLL Clocks in ZDB Mode



Note to Figure 5–27:

(1) The internal PLL clock output can lead or lag the external PLL clock outputs.

External Feedback Mode

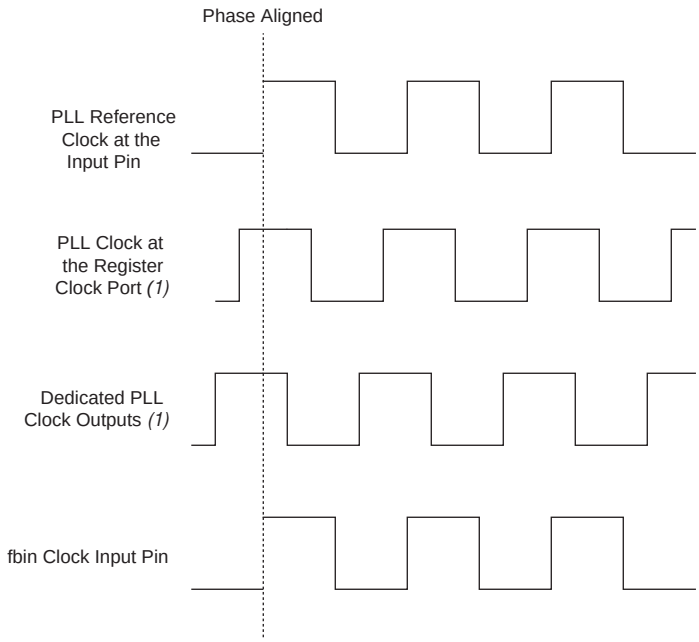
In external feedback mode, the external feedback input pin (`fbin`) is phase-aligned with the clock input pin, as shown in Figure 5–28. Aligning these clocks allows you to remove clock delay and skew between devices. This mode is supported on all Stratix IV PLLs.

In external feedback mode, the output of the `M` counter (`FBOUT`) feeds back to the PLL `fbin` input (using a trace on the board) becoming part of the feedback loop. Also, use one of the dual-purpose external clock outputs as the `fbin` input pin in this mode.

When using external feedback mode, you must use the same I/O standard on the input clock, feedback input, and output clocks. Left and right PLLs support this mode when using single-ended I/O standards only.

Figure 5–28 shows an example waveform of the phase relationship between the PLL clocks in external feedback mode.

Figure 5–28. Phase Relationship Between the PLL Clocks in External Feedback Mode

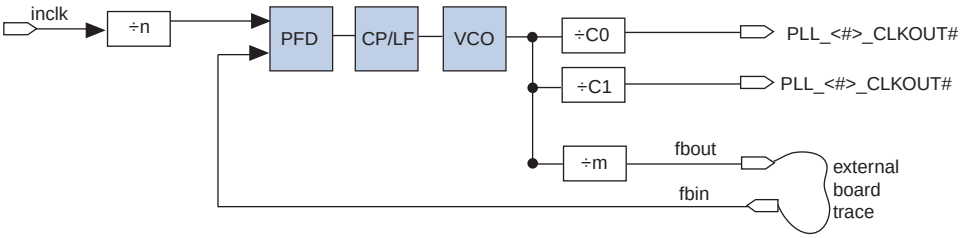


Note to Figure 5–28:

(1) The PLL clock outputs can lead or lag the `fbin` clock input.

Figure 5–29 shows external feedback mode implementation in Stratix IV devices.

Figure 5–29. External Feedback Mode in Stratix IV Devices



Clock Multiplication and Division

Each Stratix IV PLL provides clock synthesis for PLL output ports using $M/(N*$ post-scale counter) scaling factors. The input clock is divided by a pre-scale factor, n, and is then multiplied by the m feedback factor. The control loop drives the VCO to match f_{in} (M/N). Each output port has a unique post-scale counter that divides down the high-frequency VCO. For multiple PLL outputs with different frequencies, the VCO is set to the least common multiple of the output frequencies that meets its frequency specifications. For example, if the output frequencies required from one PLL are 33 and 66 MHz, the Quartus II software sets the VCO to 660 MHz (the least common multiple of 33 and 66 MHz within the VCO range). Then the post-scale counters scale down the VCO frequency for each output port.

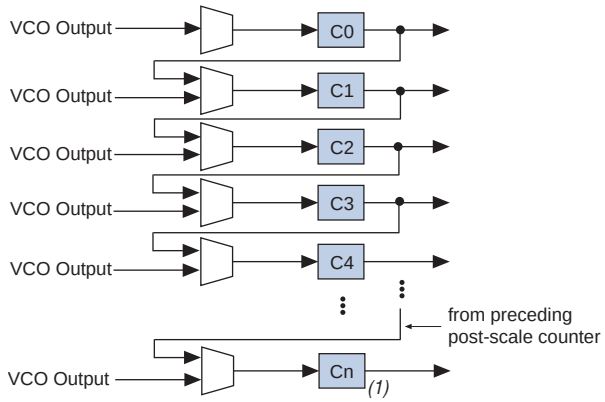
Each PLL has one pre-scale counter, n, and one multiply counter, m, with a range of 1 to 512 for both m and n. The n counter does not use duty-cycle control because the only purpose of this counter is to calculate frequency division. There are seven generic post-scale counters per left or right PLL and ten post-scale counters per top or bottom PLL that can feed the GCLKs, RCLKs, or external clock outputs. These post-scale counters range from 1 to 512 with a 50% duty cycle setting. The high- and low-count values for each counter range from 1 to 256. The sum of the high- and low-count values chosen for a design selects the divide value for a given counter.

The Quartus II software automatically chooses the appropriate scaling factors according to the input frequency, multiplication, and division values entered into the ALTPLL megafunction.

Post-Scale Counter Cascading

Stratix IV PLLs support post-scale counter cascading to create counters larger than 512. This is automatically implemented in the Quartus II software by feeding the output of one C counter into the input of the next C counter, as shown in Figure 5–30.

Figure 5–30. Counter Cascading



Note to Figure 5–30:

(1) N = 6 or N = 9

When cascading post-scale counters to implement a larger division of the high-frequency VCO clock, the cascaded counters behave as one counter with the product of the individual counter settings. For example, if C0 = 40 and C1 = 20, the cascaded value is C0 × C1 = 800.

Post-scale counter cascading is set in the configuration file. You cannot set this using PLL reconfiguration.

Programmable Duty Cycle

The programmable duty cycle allows PLLs to generate clock outputs with a variable duty cycle. This feature is supported on the PLL post-scale counters. The duty-cycle setting is achieved by a low and high time-count setting for the post-scale counters. To determine the duty cycle choices, the Quartus II software uses the frequency input and the required multiply or divide rate. The post-scale counter value determines the precision of the duty cycle. The precision is defined as 50% divided by the post-scale counter value. For example, if the `C0` counter is 10, steps of 5% are possible for duty-cycle choices from 5% to 90%.

If the PLL is in external feedback mode, set the duty cycle for the counter driving the `fbin` pin to 50%. Combining the programmable duty cycle with programmable phase shift allows the generation of precise non-overlapping clocks.

Programmable Phase Shift

Use phase shift to implement a robust solution for clock delays in Stratix IV devices. Implement phase shift by using a combination of the VCO phase output and the counter starting time. A combination of VCO phase output and counter starting time is the most accurate method of inserting delays because it is only based on counter settings, which are independent of process, voltage, and temperature (PVT).

You can phase-shift the output clocks from the Stratix IV PLLs in either of these two resolutions:

- Fine resolution using VCO phase taps
- Coarse resolution using counter starting time

Implement fine-resolution phase shifts by allowing any of the output counters (`C[n..0]`) or the `m` counter to use any of the eight phases of the VCO as the reference clock. This allows you to adjust the delay time with a fine resolution. Equation 5–1 shows the minimum delay time that you can insert using this method.

Equation 5–1. Fine-Resolution Phase Shift

$$\Phi_{fine} = \frac{1}{8} T_{VCO} = \frac{1}{8 f_{VCO}} = \frac{N}{8 M f_{REF}}$$

where f_{REF} is the input reference clock frequency.

For example, if f_{REF} is 100 MHz, N is 1, and M is 8, then f_{VCO} is 800 MHz and Φ_{fine} equals 156.25 ps. This phase shift is defined by the PLL operating frequency, which is governed by the reference clock frequency and the counter settings.

Equation 5–2 shows the coarse-resolution phase shifts are implemented by delaying the start of the counters for a predetermined number of counter clocks.

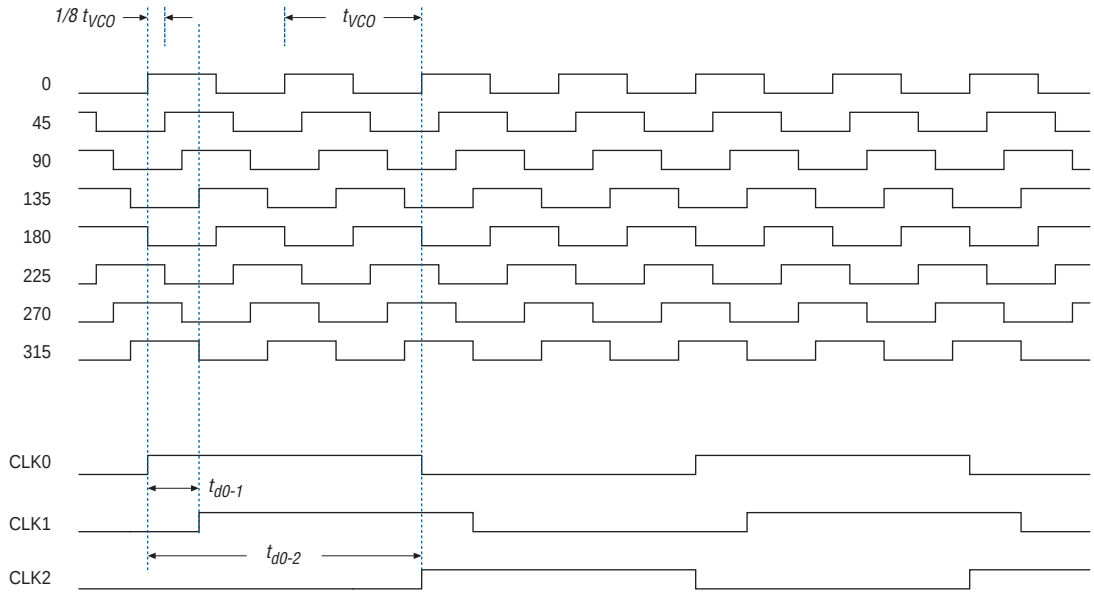
Equation 5–2. Coarse-Resolution Phase Shift

$$\Phi_{coarse} = \frac{C-1}{f_{VCO}} = \frac{(C-1)N}{Mf_{REF}}$$

where C is the count value set for the counter delay time (this is the initial setting in the "PLL usage" section of the compilation report in the Quartus II software). If the initial value is 1, C – 1 = 0° phase shift.

Figure 5–31 shows an example of phase-shift insertion with fine resolution using the VCO phase-taps method. The eight phases from the VCO are shown and labeled for reference. For this example, `CLK0` is based on the `0phase` from the VCO and has the C value for the counter set to one. The `CLK1` signal is divided by four, two VCO clocks for high time and two VCO clocks for low time. `CLK1` is based on the 135° phase tap from the VCO and also has the C value for the counter set to one. In this case, the two clocks are offset by 3 Φ_{FINE}. `CLK2` is based on the `0phase` from the VCO but has the C value for the counter set to **three**. This arrangement creates a delay of 2 Φ_{COARSE} (two complete VCO periods).

Figure 5–31. Delay Insertion Using VCO Phase Output and Counter Delay Time



You can use coarse- and fine-phase shifts to implement clock delays in Stratix IV devices.

Stratix IV devices support dynamic phase-shifting of VCO phase taps only. You can reconfigure the phase shift any number of times. Each phase shift takes about one `SCANCLK` cycle, allowing you to implement large phase shifts quickly.

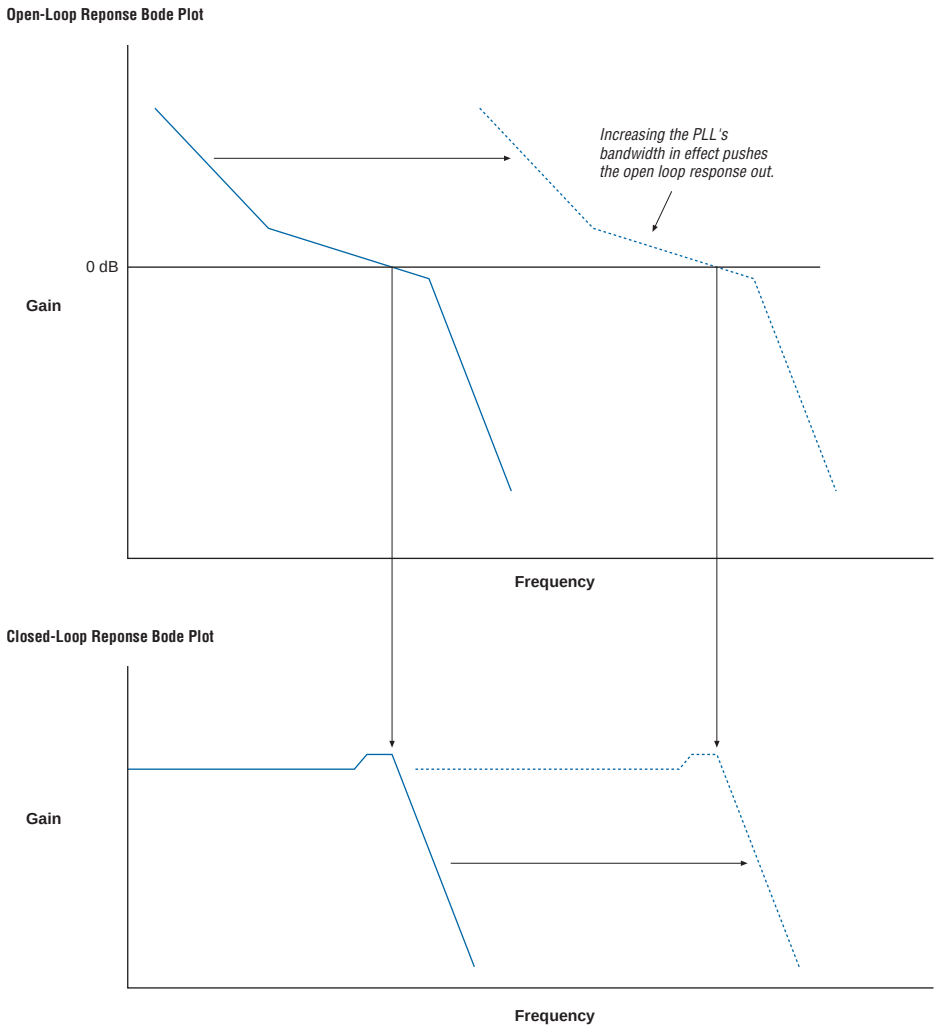
Programmable Bandwidth

Stratix IV PLLs provide advanced control of the PLL bandwidth using the PLL loop's programmable characteristics, including loop filter and charge pump.

Background

PLL bandwidth is the measure of the PLL's ability to track the input clock and its associated jitter. The closed-loop gain 3 dB frequency in the PLL determines PLL bandwidth. Bandwidth is approximately the unity gain point for open loop PLL response. As Figure 5–32 shows, these points correspond to approximately the same frequency. Stratix IV PLLs provide three bandwidth settings—low, medium (default), and high.

Figure 5–32. Open- and Closed-Loop Response Bode Plots



A high-bandwidth PLL provides a fast lock time and tracks jitter on the reference clock source, passing it through to the PLL output. A low-bandwidth PLL filters out reference clock jitter but increases lock time. Stratix IV PLLs allow you to control the bandwidth over a finite range to customize the PLL characteristics for a particular application. The programmable bandwidth feature in Stratix IV PLLs benefits applications requiring clock switchover.

A high-bandwidth PLL can benefit a system that must accept a spread-spectrum clock signal. Stratix IV PLLs can track a spread-spectrum clock by using a high-bandwidth setting. Using a low-bandwidth setting in this case could cause the PLL to filter out the jitter on the input clock.

A low-bandwidth PLL can benefit a system using clock switchover. When clock switchover occurs, the PLL input temporarily stops. A low-bandwidth PLL reacts more slowly to changes on its input clock and takes longer to drift to a lower frequency (caused by the input stopping) than a high-bandwidth PLL.

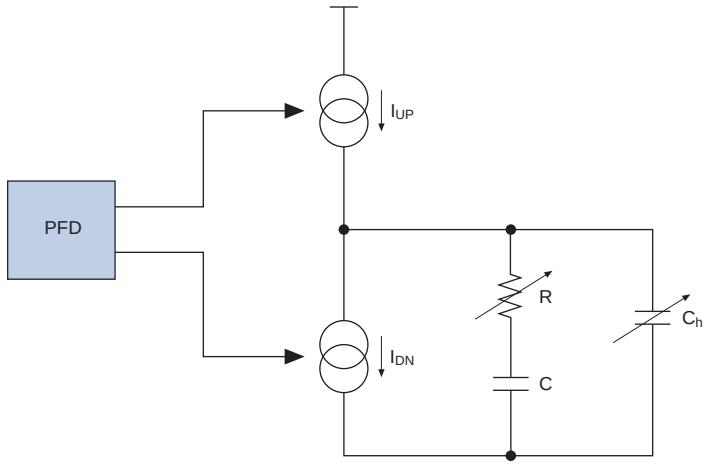
Implementation

Traditionally, external components such as the VCO or loop filter control a PLL's bandwidth. Most loop filters consist of passive components such as resistors and capacitors that take up unnecessary board space and increase cost. With Stratix IV PLLs, all the components are contained within the device to increase performance and decrease cost.

When you specify the bandwidth setting (low, medium, or high) in the ALTPLL MegaWizard Plug-in Manager, the Quartus II software automatically sets the corresponding charge pump and loop filter (`Icp`, `R`, `C`) values to achieve the desired bandwidth range.

Figure 5–33 shows the loop filter and components that you can set using the Quartus II software. The components are the loop filter resistor, R, the high frequency capacitor, C_h, and the charge pump current, I_{UP} or I_{DN}.

Figure 5–33. Loop Filter Programmable Components



Spread-Spectrum Tracking

Stratix IV devices can accept a spread-spectrum input with typical modulation frequencies. However, the device cannot automatically detect that the input is a spread-spectrum signal. Instead, the input signal looks like deterministic jitter at the input of the PLL. Stratix IV PLLs can track a spread-spectrum input clock as long as it is within the input-jitter tolerance specifications. Stratix IV devices cannot internally generate spread-spectrum clocks.

Clock Switchover

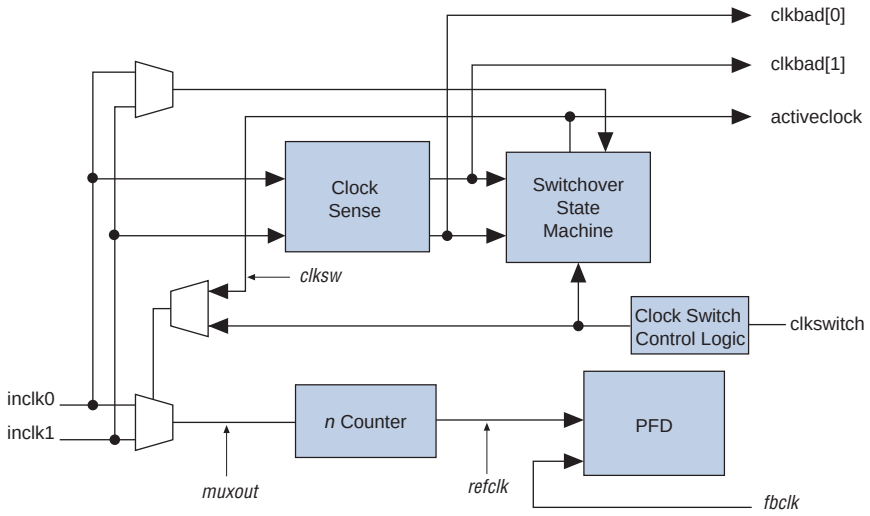
The clock switchover feature allows the PLL to switch between two reference input clocks. Use this feature for clock redundancy or for a dual-clock domain application such as in a system that turns on the redundant clock if the previous clock stops running. The design can perform clock switchover automatically when the clock is no longer toggling or based on a user control signal, `clkswitch`.

The following clock switchover modes are supported in Stratix IV PLLs:

- Automatic switchover—The clock sense circuit monitors the current reference clock and if it stops toggling, automatically switches to the other `inclk0` or `inclk1` clock.
- Manual clock switchover—Clock switchover is controlled using the `clkswitch` signal. When the `clkswitch` signal goes from logic low to logic high, and stays high for at least three clock cycles, the reference clock to the PLL is switched from `inclk0` to `inclk1`, or vice-versa.
- Automatic switchover with manual override—This mode combines automatic switchover and manual clock switchover. When the `clkswitch` signal goes high, it overrides the automatic clock switchover mode.

Stratix IV PLLs support a fully configurable clock switchover capability. Figure 5–34 shows a block diagram of the automatic switchover circuit built into the PLL. When the current reference clock is not present, the clock sense block automatically switches to the backup clock for PLL reference. The clock switchover circuit also sends out three status signals—`clkbad[0]`, `clkbad[1]`, and `activeclock`—from the PLL to implement a custom switchover circuit in the logic array. You can select a clock source as the backup clock by connecting it to the `inclk1` port of the PLL in your design.

Figure 5–34. Automatic Clock Switchover Circuit Block Diagram



Automatic Clock Switchover

Use the switchover circuitry to automatically switch between `inclk0` and `inclk1` when the current reference clock to the PLL stops toggling. For example, in applications that require a redundant clock with the same frequency as the reference clock, the switchover state machine generates a signal (`clksw`) that controls the multiplexer select input, as shown in Figure 5–34. In this case, `inclk1` becomes the reference clock for the PLL. When using automatic switchover mode, you can switch back and forth between `inclk0` and `inclk1` any number of times when one of the two clocks fails and the other clock is available.

When using automatic clock switchover mode, the following requirements must be satisfied:

- Both clock inputs must be running
- The period of the two clock inputs can differ by no more than 100% (2×)

If the current clock input stops toggling while the other clock is also not toggling, switchover is not initiated and the `clkbad[0..1]` signals are not valid. Also, if both clock inputs are not the same frequency, but their period difference is within 100%, the clock sense block detects when a clock stops toggling, but the PLL may lose lock after the switchover is completed and needs time to re-lock.

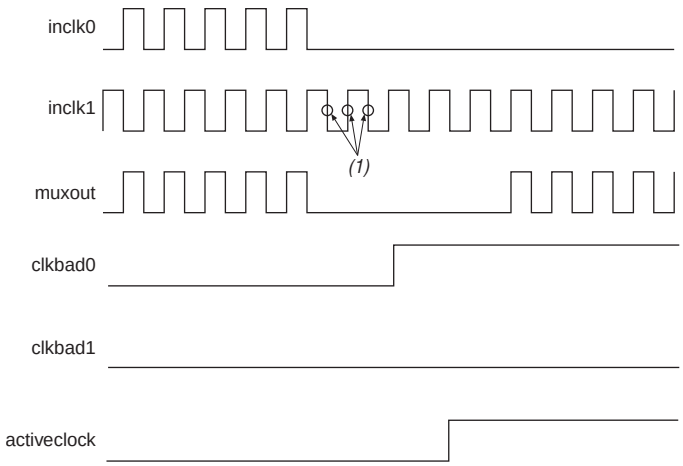
Altera recommends resetting the PLL using the `areset` signal to maintain the phase relationships between the PLL input and output clocks when using clock switchover.

In automatic switchover mode, the `clkbad[0]` and `clkbad[1]` signals indicate the status of the two clock inputs. When they are asserted, the clock sense block has detected that the corresponding clock input has stopped toggling. These two signals are not valid if the frequency difference between `inclk0` and `inclk1` is greater than 20%.

The `activeclock` signal indicates which of the two clock inputs (`inclk0` or `inclk1`) is being selected as the reference clock to the PLL. When the frequency difference between the two clock inputs is more than 20%, the `activeclock` signal is the only valid status signal.

Figure 5–35 shows an example waveform of the switchover feature when using automatic switchover mode. In this example, the `inclk0` signal is stuck low. After the `inclk0` signal is stuck at low for approximately two clock cycles, the clock sense circuitry drives the `clkbad[0]` signal high. Also, because the reference clock signal is not toggling, the switchover state machine controls the multiplexer through the `clkswitch` signal to switch to the backup clock, `inclk1`.

Figure 5–35. Automatic Switchover Upon Loss of Clock Detection



Note to Figure 5–35:

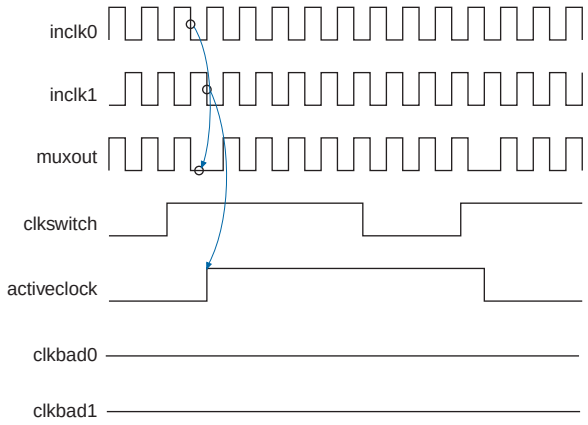
(1) Switchover is enabled on the falling edge of `inclk0` or `inclk1`, depending on which clock is available. In this figure, switchover is enabled on the falling edge of `inclk1`.

Manual Override

In automatic switchover with manual override mode, you can use the `clkswitch` input for user- or system-controlled switch conditions. You can use this mode for same-frequency switchover, or to switch between inputs of different frequencies. For example, if `inclk0` is 66 MHz and `inclk1` is 200 MHz, you must control switchover using `clkswitch` because the automatic clock-sense circuitry cannot monitor clock input (`inclk0` and `inclk1`) frequencies with a frequency difference of more than 100% (2×). This feature is useful when the clock sources originate from multiple cards on the backplane, requiring a system-controlled switchover between the frequencies of operation. You must choose the backup clock frequency and set the `m`, `n`, `c`, and `k` counters accordingly so the VCO operates within the recommended operating frequency range of 600 to 1,600 MHz. The ALTPLL MegaWizard Plug-in Manager notifies you if a given combination of `inclk0` and `inclk1` frequencies cannot meet this requirement.

Figure 5–36 shows an example of a waveform showing the switchover feature when controlled by `clkswitch`. In this case, both clock sources are functional and `inclk0` is selected as the reference clock; `clkswitch` goes high, which starts the switchover sequence. On the falling edge of `inclk0`, the counter's reference clock, `muxout`, is gated off to prevent clock glitching. On the falling edge of `inclk1`, the reference clock multiplexer switches from `inclk0` to `inclk1` as the PLL reference and the `activeclock` signal changes to indicate which clock is currently feeding the PLL.

Figure 5–36. Clock Switchover Using the clkswitch (Manual) Control *(Note 1)*



Note to Figure 5–36:

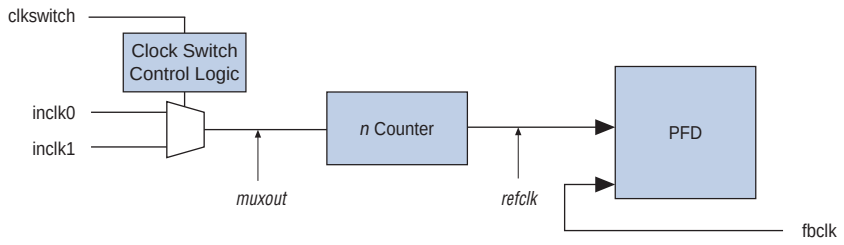
(1) To initiate a manual clock switchover event, both `inclk0` and `inclk1` must be running when the `clkswitch` signal goes high.

In automatic override with manual switchover mode, the `activeclock` signal mirrors the `clkswitch` signal. As both clocks are still functional during the manual switch, neither `clkbad` signal goes high. Because the switchover circuit is positive-edge sensitive, the falling edge of the `clkswitch` signal does not cause the circuit to switch back from `inclk1` to `inclk0`. When the `clkswitch` signal goes high again, the process repeats. `clkswitch` and automatic switch only work if the clock being switched to is available. If the clock is not available, the state machine waits until the clock is available.

Manual Clock Switchover

In manual clock switchover mode, the `clkswitch` signal controls whether `inclk0` or `inclk1` is selected as the input clock to the PLL. By default, `inclk0` is selected. A low-to-high transition on `clkswitch` and `clkswitch` being held high for at least three `inclk` cycles initiates a clock switchover event. You must bring `clkswitch` back low again in order to perform another switchover event in the future. If you do not require another switchover event in the future, you can leave `clkswitch` in a logic high state after the initial switch. Pulsing `clkswitch` high for at least three `inclk` cycles performs another switchover event. If `inclk0` and `inclk1` are different frequencies and are always running, the `clkswitch` minimum high time must be greater than or equal to three of the slower frequency `inclk0` or `inclk1` cycles. Figure 5–37 shows a block diagram of the manual switchover circuit.

Figure 5–37. Manual Clock Switchover Circuitry in Stratix IV PLLs



For more information about PLL software support in the Quartus II software, refer to the *ALTPLL Megafunction User Guide*.

Guidelines

When implementing clock switchover in Stratix IV PLLs, use the following guidelines:

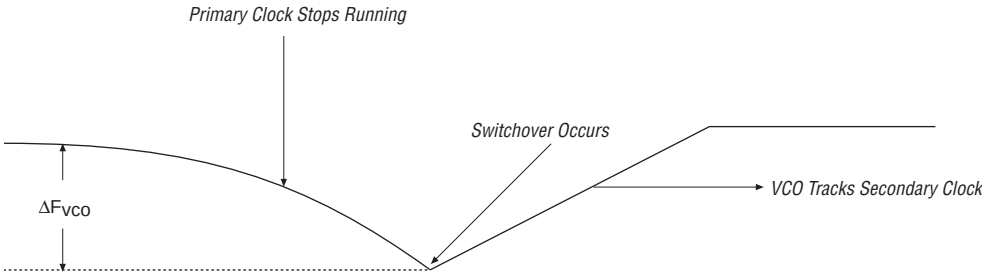
- Automatic clock switchover requires that the `inclk0` and `inclk1` frequencies be within 100% (2×) of each other. Failing to meet this requirement causes the `clkbad[0]` and `clkbad[1]` signals to not function properly.
- When using manual clock switchover, the difference between `inclk0` and `inclk1` can be more than 100% (2×). However, differences in frequency, phase, or both, of the two clock sources will likely cause the PLL to lose lock. Resetting the PLL ensures that the correct phase relationships are maintained between input and output clocks.

  Both `inclk0` and `inclk1` must be running when the `clkswitch` signal goes high to initiate the manual clock switchover event. Failing to meet this requirement causes the clock switchover to not function properly.

- Applications that require a clock switchover feature and a small frequency drift must use a low-bandwidth PLL. The low-bandwidth PLL reacts more slowly than a high-bandwidth PLL to reference input clock changes. When switchover happens, a low-bandwidth PLL propagates the stopping of the clock to the output more slowly than a high-bandwidth PLL. However, be aware that the low-bandwidth PLL also increases lock time.
- After a switchover occurs, there may be a finite resynchronization period for the PLL to lock onto a new clock. The exact amount of time it takes for the PLL to re-lock depends on the PLL configuration.
- The phase relationship between the input clock to the PLL and the output clock from the PLL is important in your design. Assert `areset` for at least 10 ns after performing a clock switchover. Wait for the locked signal to go high and be stable before re-enabling the output clocks from the PLL.

- Figure 5–38 shows how the VCO frequency gradually decreases when the current clock is lost and then increases as the VCO locks on to the backup clock.

Figure 5–38. VCO Switchover Operating Frequency



- Disable the system during clock switchover if it is not tolerant of frequency variations during the PLL resynchronization period. You can use the `clkbad[0]` and `clkbad[1]` status signals to turn off the PFD (`PFDENA` = `0`) so the VCO maintains its most recent frequency. You can also use the state machine to switch over to the secondary clock. When the PFD is re-enabled, output clock-enable signals (`clkena`) can disable clock outputs during the switchover and resynchronization period. When the lock indication is stable, the system can re-enable the output clocks.

PLL Reconfiguration

PLLs use several divide counters and different VCO phase taps to perform frequency synthesis and phase shifts. In Stratix IV PLLs, you can reconfigure both the counter settings and phase-shift the PLL output clock in real time. You can also change the charge pump and loop-filter components, which dynamically affects PLL bandwidth. You can use these PLL components to update the output-clock frequency and PLL bandwidth and to phase-shift in real time, without reconfiguring the entire Stratix IV device.

The ability to reconfigure the PLL in real time is useful in applications that operate at multiple frequencies. It is also useful in prototyping environments, allowing you to sweep PLL output frequencies and adjust the output-clock phase dynamically. For instance, a system generating test patterns is required to generate and transmit patterns at 75 or 150 MHz, depending on the requirements of the device under test. Reconfiguring the PLL components in real time allows you to switch between two such output frequencies within a few microseconds. You can also use this feature to adjust clock-to-out (`tCO`) delays in real time by changing the PLL output clock phase shift. This approach eliminates the need to regenerate a configuration file with the new PLL settings.

PLL Reconfiguration Hardware Implementation

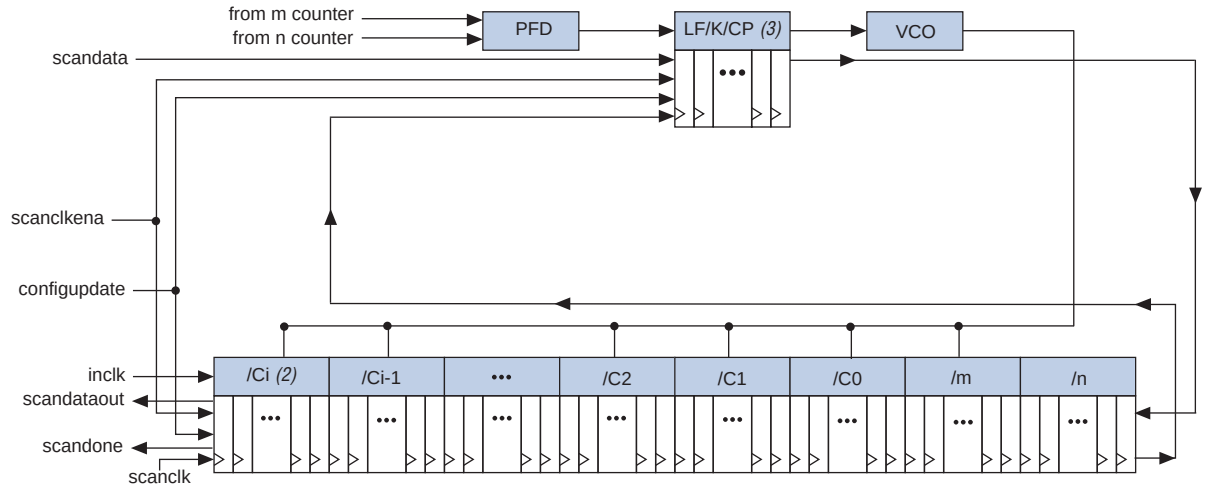
The following PLL components are reconfigurable in real time:

- Pre-scale counter (`n`)
- Feedback counter (`m`)

- Post-scale output counters (C0 - C9)
- Post VCO Divider (K)
- Dynamically adjust the charge-pump current (Icp) and loop-filter components (R, C) to facilitate reconfiguration of the PLL bandwidth

Figure 5–39 shows how you can dynamically adjust the PLL counter settings by shifting their new settings into a serial shift-register chain or scan chain. Serial data is input to the scan chain using the scandata port. Shift registers are clocked by scanclk. The maximum scanclk frequency is 100 MHz. Serial data is shifted through the scan chain as long as the scanclkena signal stays asserted. After the last bit of data is clocked, asserting the configupdate signal for at least one scanclk clock cycle causes the PLL configuration bits to be synchronously updated with the data in the scan registers.

Figure 5–39. PLL Reconfiguration Scan Chain *(Note 1)*



Notes to Figure 5–39:

(1) Stratix IV left and right PLLs support C0 - C6 counters.

(2) i = 6 or i = 9.

(3) This figure shows the corresponding scan register for the K counter in between the scan registers for the charge pump and loop filter. The K counter is physically located after the VCO.

The counter settings are updated synchronously to the clock frequency of the individual counters. Therefore, all counters are not updated simultaneously.

Table 5–10 lists how these signals can be driven by the PLD logic array or I/O pins.

Table 5–10. Real-Time PLL Reconfiguration Ports

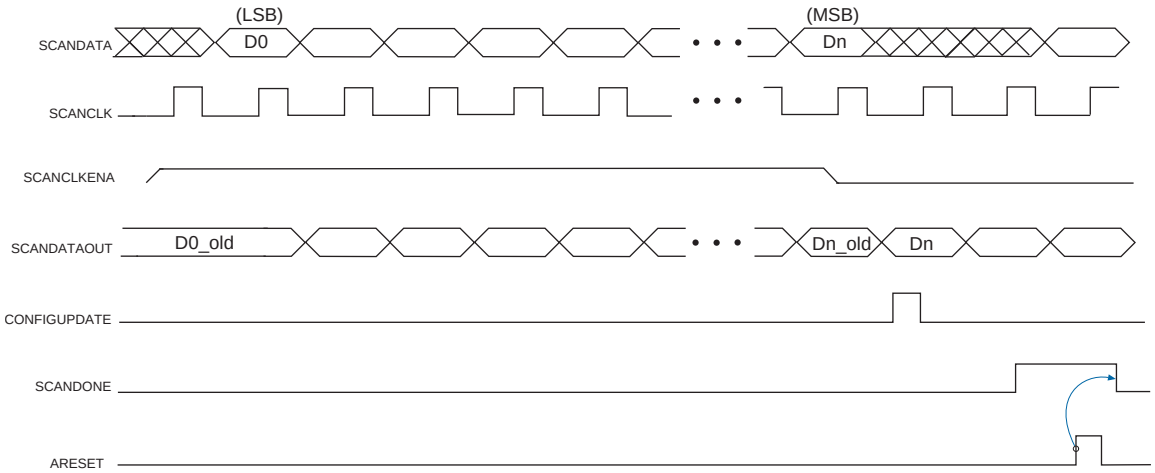
PLL Port Name	Description	Source	Destination
`scandata`	Serial input data stream to scan chain.	Logic array or I/O pin	PLL reconfiguration circuit
`scanclk`	Serial clock input signal. This clock can be free running.	GCLK, RCLK or I/O pins	PLL reconfiguration circuit
`scanclkena`	Enables `scanclk` and allows the `scandata` to be loaded in the scan chain. Active high.	Logic array or I/O pin	PLL reconfiguration circuit
`configupdate`	Writes the data in the scan chain to the PLL. Active high.	Logic array or I/O pin	PLL reconfiguration circuit
`scandone`	Indicates when the PLL has finished reprogramming. A rising edge indicates the PLL has begun reprogramming. A falling edge indicates the PLL has finished reprogramming.	PLL reconfiguration circuit	Logic array or I/O pins
`scandataout`	Used to output the contents of the scan chain.	PLL reconfiguration circuit	Logic array or I/O pins

To reconfigure the PLL counters, perform the following steps:

1. The `scanclkena` signal is asserted at least one `scanclk` cycle prior to shifting in the first bit of `scandata` (`D0`).
2. Serial data (`scandata`) is shifted into the scan chain on the second rising edge of `scanclk`.
3. After all 234 bits (top and bottom PLLs) or 180 bits (left and right PLLs) have been scanned into the scan chain, the `scanclkena` signal is de-asserted to prevent inadvertent shifting of bits in the scan chain.
4. The `configupdate` signal is asserted for one `scanclk` cycle to update the PLL counters with the contents of the scan chain.
5. The `scandone` signal goes high, indicating the PLL is being reconfigured. A falling edge indicates the PLL counters have been updated with new settings.
6. Reset the PLL using the `areset` signal if you make any changes to the `M`, `N`, or post-scale output C counters or to the `Icp`, `R`, or `C` settings.
7. You can repeat steps 1-5 to reconfigure the PLL any number of times.

Figure 5–40 shows a functional simulation of the PLL reconfiguration feature.

Figure 5–40. PLL Reconfiguration Waveform



When you reconfigure the counter clock frequency, you cannot reconfigure the corresponding counter phase shift settings using the same interface. Instead, reconfigure the phase shifts in real time using the dynamic phase shift reconfiguration interface. If you reconfigure the counter frequency, but wish to keep the same non-zero phase shift setting (for example, 90°) on the clock output, you must reconfigure the phase shift immediately after reconfiguring the counter clock frequency.

Post-Scale Counters (C0 to C9)

You can reconfigure the multiply or divide values and duty cycle of post-scale counters in real time. Each counter has an 8-bit high-time setting and an 8-bit low-time setting. The duty cycle is the ratio of output high- or low-time to the total cycle time, which is the sum of the two. Additionally, these counters have two control bits, `rbypass`, for bypassing the counter, and `rselodd`, to select the output clock duty cycle.

When the `rbypass` bit is set to 1, it bypasses the counter, resulting in a divide by 1. When the `rbypass` bit is set to 0, the high- and low-time counters are added to compute the effective division of the VCO output frequency. For example, if the post-scale divide factor is 10, the high- and low-count values can be set to 5 and 5, respectively, to achieve a 50% - 50% duty cycle. The PLL implements this duty cycle by transitioning the output clock from high to low on the rising edge of the VCO output clock. However, a 4 and 6 setting for the high- and low-count values, respectively, produces an output clock with a 40% - 60% duty cycle.

The `rselodd` bit indicates an odd divide factor for the VCO output frequency along with a 50% duty cycle. For example, if the post-scale divide factor is 3, the high- and low-time count values could be set to 2 and 1, respectively, to achieve this division. This implies a 67% - 33% duty cycle. If you need a 50% - 50% duty cycle, you can set the `rselodd` control bit to 1 to achieve this duty cycle despite an odd division factor. The PLL implements this duty cycle by transitioning the output clock from high to low on a falling edge of the VCO output clock. When you set `rselodd` = 1, you subtract 0.5 cycles from the high time and you add 0.5 cycles to the low time. For example:

- High-time count = 2 cycles
- Low-time count = 1 cycle
- `rselodd` = 1 effectively equals:
  - High-time count = 1.5 cycles
  - Low-time count = 1.5 cycles
  - Duty cycle = (1.5/3) % high-time count and (1.5/3) % low-time count

Scan Chain Description

The length of the scan chain varies for different Stratix IV PLLs. The top and bottom PLLs have ten post-scale counters and a 234-bit scan chain, while the left and right PLLs have seven post-scale counters and a 180-bit scan chain. Table 5–11 lists the number of bits for each component of a Stratix IV PLL.

Table 5–11. Top and Bottom PLL Reprogramming Bits (Part 1 of 2)

Block Name	Number of Bits		Total
	Counter	Other *(1)*	
C9 *(2)*	16	2	18
C8	16	2	18
C7	16	2	18
C6 *(3)*	16	2	18
C5	16	2	18
C4	16	2	18
C3	16	2	18
C2	16	2	18
C1	16	2	18
C0	16	2	18
M	16	2	18
N	16	2	18
Charge Pump Current	0	3	3
VCO Post-Scale divider (K)	1	0	1
Loop Filter Capacitor *(4)*	0	2	2
Loop Filter Resistor	0	5	5
Unused CP/LF	0	7	7

Table 5–11. Top and Bottom PLL Reprogramming Bits (Part 2 of 2)

Block Name	Number of Bits		Total
	Counter	Other *(1)*	
Total number of bits	—	—	234

Notes to Table 5–11:

(1) Includes two control bits, `rbypass`, for bypassing the counter, and `rselodd`, to select the output clock duty cycle.

(2) The LSB for the C9 low-count value is the first bit shifted into the scan chain for the top and bottom PLLs.

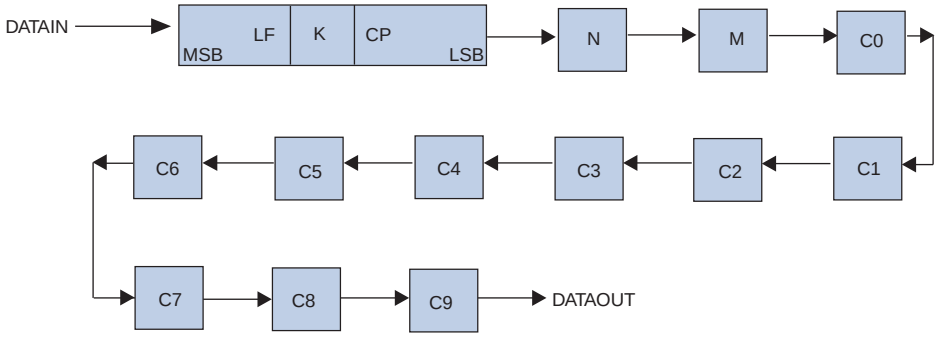
(3) The LSB for the C6 low-count value is the first bit shifted into the scan chain for the left and right PLLs.

(4) The MSB for the loop filter is the last bit shifted into the scan chain.

Table 5–11 lists the scan chain order of PLL components for the top and bottom PLLs, which have 10 post-scale counters. The order of bits is the same for the left and right PLLs, but the reconfiguration bits start with the C6 post-scale counter.

Figure 5–41 shows the scan-chain order of PLL components for the top and bottom PLLs.

Figure 5–41. Scan-Chain Order of PLL Components for Top and Bottom PLLs *(Note 1)*

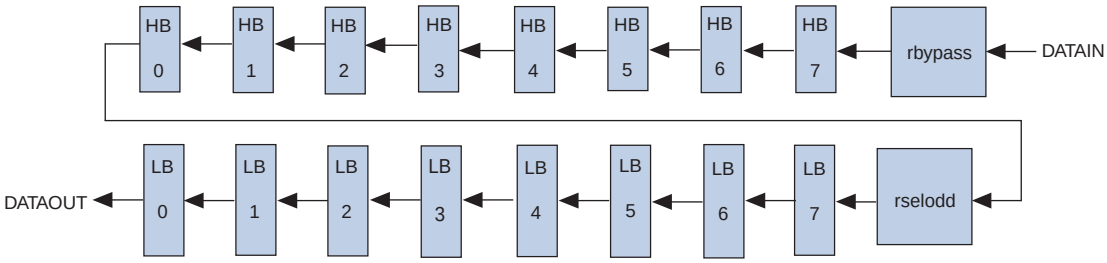


Note to Figure 5–41:

(1) Left and right PLLs have the same scan-chain order. The post-scale counters end at C6.

Figure 5–42 shows the scan-chain bit-order sequence for post-scale counters in all Stratix IV PLLs.

Figure 5–42. Scan-Chain Bit-Order Sequence for Post-Scale Counters in Stratix IV PLLs



Charge Pump and Loop Filter

You can reconfigure the charge-pump and loop-filter settings to update the PLL bandwidth in real time.

Table 5–12 lists the possible settings for charge pump current (`Icp`) values for Stratix IV PLLs.

Table 5–12. Charge Pump Current Bit Settings

CP[2]	CP[1]	CP[0]	Decimal Value for Setting
0	0	0	0
0	0	1	1
0	1	1	3
1	1	1	7

Table 5–13 lists the possible settings for loop-filter resistor (`R`) values for Stratix IV PLLs.

Table 5–13. Loop-Filter Resistor Bit Settings

LFR[4]	LFR[3]	LFR[2]	LFR[1]	LFR[0]	Decimal Value for Setting
0	0	0	0	0	0
0	0	0	1	1	3
0	0	1	0	0	4
0	1	0	0	0	8
1	0	0	0	0	16
1	0	0	1	1	19
1	0	1	0	0	20
1	1	0	0	0	24
1	1	0	1	1	27
1	1	1	0	0	28
1	1	1	1	0	30

Table 5–14 lists the possible settings for loop-filter capacitor (`C`) values for Stratix IV PLLs.

Table 5–14. Loop-Filter Capacitor Bit Settings

LFC[1]	LFC[0]	Decimal Value for Setting
0	0	0
0	1	1
1	1	3

Bypassing PLL

Bypassing a PLL counter results in a multiply (m counter) or a divide (n and C0 to C9 counters) factor of one.

Table 5–15 lists the settings for bypassing the counters in Stratix IV PLLs.

Table 5–15. PLL Counter Settings

PLL Scan Chain Bits [0..8] Settings									
LSB *(2)*								**MSB** *(1)*	**Description**
X	X	X	X	X	X	X	X	1 *(3)*	PLL counter bypassed
X	X	X	X	X	X	X	X	0 *(3)*	PLL counter not bypassed because bit 8 (MSB) is set to 0

Notes to Table 5–15:

(1) Most significant bit (MSB).
(2) Least significant bit (LSB).
(3) Counter-bypass bit.

To bypass any of the PLL counters, set the bypass bit to **1**. The values on the other bits are ignored. To bypass the VCO post-scale counter (K), set the corresponding bit to 0.

Dynamic Phase-Shifting

The dynamic phase-shifting feature allows the output phases of individual PLL outputs to be dynamically adjusted relative to each other and to the reference clock, without the need to send serial data through the scan chain of the corresponding PLL. This feature simplifies the interface and allows you to quickly adjust the clock-to-out (t_{co}) delays by changing the output clock phase-shift in real time. This adjustment is achieved by incrementing or decrementing the VCO phase-tap selection to a given C counter or to the M counter. The phase is shifted by 1/8 of the VCO frequency at a time. The output clocks are active during this phase-reconfiguration process.

Table 5–16 lists the control signals that are used for dynamic phase-shifting.

Table 5–16. Dynamic Phase-Shifting Control Signals (Part 1 of 2)

Signal Name	Description	Source	Destination
PHASECOUNTERSELECT[3..0]	Counter select. Four bits decoded to select either the M or one of the C counters for phase adjustment. One address maps to select all C counters. This signal is registered in the PLL on the rising edge of SCANCLK.	Logic array or I/O pins	PLL reconfiguration circuit
PHASEUPDOWN	Selects dynamic phase shift direction; 1 = UP; 0 = DOWN. Signal is registered in the PLL on the rising edge of SCANCLK.	Logic array or I/O pin	PLL reconfiguration circuit
PHASESTEP	Logic high enables dynamic phase shifting.	Logic array or I/O pin	PLL reconfiguration circuit

Table 5–16. Dynamic Phase-Shifting Control Signals (Part 2 of 2)

Signal Name	Description	Source	Destination
SCANCLK	Free running clock from the core used in combination with PHASESTEP to enable and disable dynamic phase shifting. Shared with SCANCLK for dynamic reconfiguration.	GCLK, RCLK or I/O pin	PLL reconfiguration circuit
PHASEDONE	When asserted, this indicates to core-logic that the phase adjustment is complete and the PLL is ready to act on a possible second adjustment pulse. Asserts based on internal PLL timing. De-asserts on the rising edge of SCANCLK.	PLL reconfiguration circuit	Logic array or I/O pins

Table 5–17 lists the PLL counter selection based on the corresponding PHASECOUNTERSELECT setting.

Table 5–17. Phase Counter Select Mapping

PHASECOUNTERSELECT[3]	[2]	[1]	[0]	Selects
0	0	0	0	All Output Counters
0	0	0	1	M Counter
0	0	1	0	C0 Counter
0	0	1	1	C1 Counter
0	1	0	0	C2 Counter
0	1	0	1	C3 Counter
0	1	1	0	C4 Counter
0	1	1	1	C5 Counter
1	0	0	0	C6 Counter
1	0	0	1	C7 Counter
1	0	1	0	C8 Counter
1	0	1	1	C9 Counter

To perform one dynamic phase-shift, follow these steps:

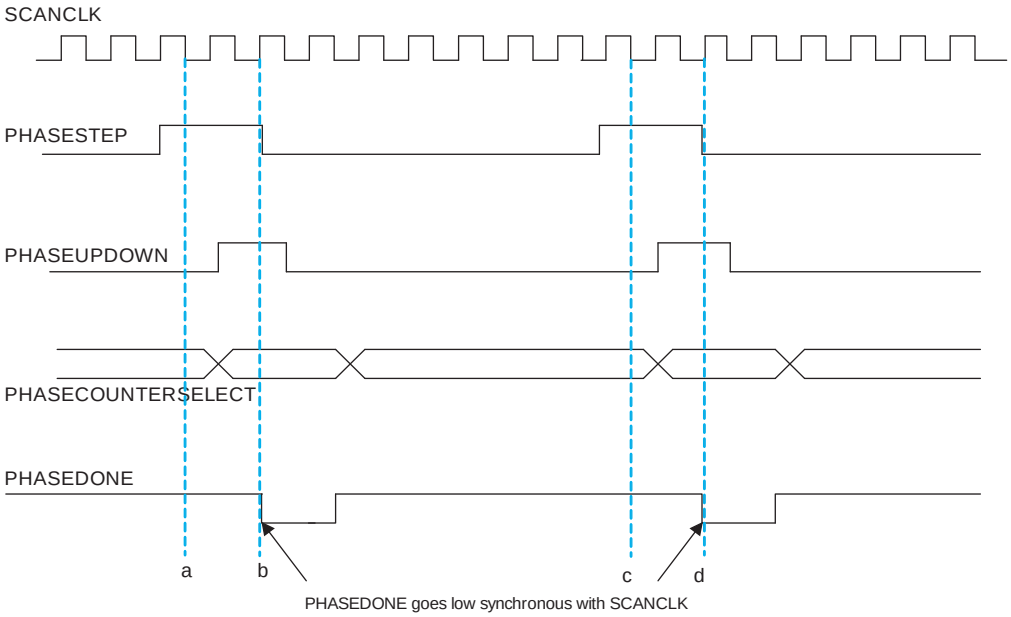
1. Set PHASEUPDOWN and PHASECOUNTERSELECT as required.
2. Assert PHASESTEP. Each PHASESTEP pulse enables one phase shift. The PHASESTEP pulses must be at least one scanclk cycle apart.
3. Wait for PHASEDONE to go low.
4. De-assert PHASESTEP.
5. Wait for PHASEDONE to go high.
6. Repeat steps 1-5 as many times as required to perform multiple phase-shifts.

All signals are synchronous to SCANCLK and are latched on the SCANCLK edges and must meet tsu/th requirements with respect to SCANCLK edges.

You can repeat dynamic phase-shifting indefinitely. For example, in a design where the VCO frequency is set to 1000 MHz and the output clock frequency is 100 MHz, performing 40 dynamic phase shifts (each one yields 125 ps phase shift) results in shifting the output clock by 180°, which is a phase shift of 5 ns.

The PHASESTEP signal is latched on the negative edge of SCANCLK. In Figure 5–43, this is shown by the second SCANCLK falling edge. PHASESTEP must stay high for at least two SCANCLK cycles. On the second SCANCLK rising edge after PHASESTEP is latched (the fourth SCANCLK rising edge in Figure 5–43), the values of PHASEUPDOWN and PHASECOUNTERSELECT are latched and the PLL starts dynamic phase-shifting for the specified counter(s) and in the indicated direction. On the fourth SCANCLK rising edge, PHASEDONE goes from high to low and remains low until the PLL finishes dynamic phase-shifting. You can perform another dynamic phase shift after the PHASEDONE signal goes from low to high.

Figure 5–43. Dynamic Phase Shifting Waveform



Depending on the VCO and SCANCLK frequencies, PHASEDONE low time may be greater than or less than one SCANCLK cycle. The maximum time for reconfiguring phase shift dynamically is to be determined based on device characterization.

After PHASEDONE goes from low to high, you can perform another dynamic phase shift.

For information about the ALTPLL_RECONFIG MegaWizard Plug-In Manager, refer to the *ALTPLL_RECONFIG Megafunction User Guide*.

PLL Specifications

For information about PLL timing specifications, refer to the *DC and Switching Characteristics of Stratix IV Devices* chapter.

Chapter Revision History

Table 5–18 lists the revision history for this chapter.

Table 5–18. Chapter Revision History (Part 1 of 2)

Date and Document Version	Changes Made	Summary of Changes
November 2009 v3.0	■ Updated Table 5–1 and Table 5–7. ■ Updated "Clock Networks in Stratix IV Devices", "Periphery Clock Networks", and "Cascading PLLs" sections. ■ Added Figure 5–5, Figure 5–6, Figure 5–7, Figure 5–8, and Figure 5–9. ■ Added "Clock Sources Per Region" section. ■ Updated Figure 5–40. ■ Removed EP4SE110, EP4SE290, and EP4SE680 devices. ■ Added EP4S40G2, EP4S100G2, EP4S40G5, EP4S100G3, EP4S100G4, EP4S100G5, and EP4SE820 devices.	—
June 2009 v2.3	■ Updated Table 5–7. ■ Updated the "PLL Reconfiguration Hardware Implementation" and "Zero-Delay Buffer Mode" sections. ■ Added introductory sentences to improve search ability. ■ Removed the Conclusion section. ■ Minor text edits.	—
April 2009 v2.2	■ Updated Table 5–1 and Table 5–7. ■ Updated Figure 5–3 and Figure 5–4. ■ Updated the "Periphery Clock Networks" section.	—
March 2009 v2.1	■ Updated Table 5–7. ■ Updated Figure 5–34. ■ Updated "Guidelines" section. ■ Removed "Referenced Documents" section.	—

Table 5–18. Chapter Revision History (Part 2 of 2)

Date and Document Version	Changes Made	Summary of Changes
November 2008 v2.0	■ Updated Table 5–7. ■ Updated Note 1 of Figure 5–10. ■ Updated Figure 5–15. ■ Updated Figure 5–20. ■ Added Figure 5–21. ■ Made minor editorial changes.	—
May 2008 v1.0	Initial Release.	—

Section II. I/O Interfaces

This section provides information on Stratix® IV device I/O features, external memory interfaces, and high-speed differential interfaces with DPA. This section includes the following chapters:

- Chapter 6, I/O Features in Stratix IV Devices
- Chapter 7, External Memory Interfaces in Stratix IV Devices
- Chapter 8, High-Speed Differential I/O Interfaces and DPA in Stratix IV Devices

Revision History

Refer to each chapter for its own specific revision history. For information on when each chapter was updated, refer to the Chapter Revision Dates section, which appears in the full handbook.

6. I/O Features in Stratix IV Devices

SIV51006-3.0

This chapter describes how Stratix® IV devices provide I/O capabilities that allow you to work in compliance with current and emerging I/O standards and requirements. With these device features, you can reduce board design interface costs and increase development flexibility.

Altera® Stratix IV FPGAs deliver a breakthrough level of system bandwidth and power efficiency for high-end applications, allowing you to innovate without compromise. Stratix IV I/Os are specifically designed for ease-of-use and rapid system integration while simultaneously providing the high bandwidth required to maximize internal logic capabilities and produce system-level performance.

Stratix IV device I/O capability far exceeds the I/O bandwidth available from previous generation FPGAs. Independent modular I/O banks with a common bank structure for vertical migration lend efficiency and flexibility to the high-speed I/O.

Package and die enhancements with dynamic termination and output control provide best-in-class signal integrity. Numerous I/O features assist high-speed data transfer into and out of the device, including:

- Up to 32 full-duplex clock data recovery (CDR)-based transceivers supporting data rates between 600 Mbps and 8.5 Gbps
- Dedicated circuitry to support physical layer functionality for popular serial protocols, such as PCI Express (PIPE) Gen1 and Gen2, Gigabit Ethernet, Serial RapidIO, SONET/SDH, XAUI/HiGig, (OIF) CEI-6G, SD/HD/3G-SDI, Fibre Channel, SFI-5, and Interlaken
- Complete PCI Express (PIPE) protocol solution with embedded PCI Express hard IP blocks that implement PHY-MAC layer, data link layer, and transaction layer functionality
- Single-ended, non-voltage-referenced, and voltage-referenced I/O standards
- Low-voltage differential signaling (LVDS), reduced swing differential signaling (RSDS), mini-LVDS, high-speed transceiver logic (HSTL), and SSTL
- Single data rate (SDR) and half data rate (HDR—half frequency and twice data width of SDR) input and output options
- Up to 132 full duplex 1.6 Gbps true LVDS channels (132 Tx + 132 Rx) on the row I/O banks
- Hard dynamic phase alignment (DPA) block with serializer/deserializer (SERDES)
- Deskew, read and write leveling, and clock-domain crossing functionality
- Programmable output current strength
- Programmable slew rate
- Programmable delay
- Programmable bus-hold circuit
- Programmable pull-up resistor

- Open-drain output
- Serial, parallel, and dynamic on-chip termination (OCT)
- Differential OCT
- Programmable pre-emphasis
- Programmable equalization
- Programmable differential output voltage (V_{OD})

This chapter contains the following sections:

- "I/O Standards Support"
- "I/O Banks" on page 6–5
- "I/O Structure" on page 6–17
- "On-Chip Termination Support and I/O Termination Schemes" on page 6–24
- "OCT Calibration" on page 6–31
- "Termination Schemes for I/O Standards" on page 6–37
- "Design Considerations" on page 6–44

I/O Standards Support

Stratix IV devices support a wide range of industry I/O standards. Table 6–1 lists the I/O standards Stratix IV devices support, as well as the typical applications. These devices support V_{CCIO} voltage levels of 3.0, 2.5, 1.8, 1.5, and 1.2 V.

Table 6–1. Stratix IV I/O Standards and Applications (Part 1 of 2)

I/O Standard	Application
3.3-V LVTTL/LVCMOS *(1)*, *(2)*	General purpose
2.5-V LVCMOS	General purpose
1.8-V LVCMOS	General purpose
1.5-V LVCMOS	General purpose
1.2-V LVCMOS	General purpose
3.0-V PCI/PCI-X	PC and embedded system
SSTL-2 Class I and II	DDR SDRAM
SSTL-18 Class I and II	DDR2 SDRAM
SSTL-15 Class I and II	DDR3 SDRAM
HSTL-18 Class I and II	QDRII/RLDRAM II
HSTL-15 Class I and II	QDRII/QDRII+/RLDRAM II
HSTL-12 Class I and II	General purpose
Differential SSTL-2 Class I and II	DDR SDRAM
Differential SSTL-18 Class I and II	DDR2 SDRAM
Differential SSTL-15 Class I and II	DDR3 SDRAM
Differential HSTL-18 Class I and II	Clock interfaces
Differential HSTL-15 Class I and II	Clock interfaces

Table 6–1. Stratix IV I/O Standards and Applications (Part 2 of 2)

I/O Standard	Application
Differential HSTL-12 Class I and II	Clock interfaces
LVDS	High-speed communications
RSDS	Flat panel display
mini-LVDS	Flat panel display
LVPECL	Video graphics and clock distribution

Notes to Table 6–1:

(1) The 3.3-V LVTTL/LVCMOS standard is supported using V_{CCIO} at 3.0 V.

(2) For more information about the 3.3-V LVTTL/LVCMOS standard supported in Stratix IV devices, refer to "3.3-V I/O Interface" on page 6–19.

For more information about transceiver supported I/O standards, refer to the *Stratix IV Transceiver Architecture* chapter.

I/O Standards and Voltage Levels

Stratix IV devices support a wide range of industry I/O standards, including single-ended, voltage-referenced single-ended, and differential I/O standards.

Table 6–2 lists the supported I/O standards and typical values for input and output V_{CCIO}, V_{CCPD}, V_{REF}, and board V_{TT}.

Table 6–2. Stratix IV I/O Standards and Voltage Levels *(Note 1), (2)* (Part 1 of 3)

I/O Standard	Standard Support	V_{CCIO} (V)				V_{CCPD} (V) (Pre-Driver Voltage)	V_{REF} (V) (Input Ref Voltage)	V_{TT} (V) (Board Termination Voltage)
		Input Operation		Output Operation				
		Column I/O Banks	Row I/O Banks	Column I/O Banks	Row I/O Banks			
3.3-V LVTTL	JESD8-B	3.0/2.5	3.0/2.5	3.0	3.0	3.0	—	—
3.3-V LVCMOS *(3)*	JESD8-B	3.0/2.5	3.0/2.5	3.0	3.0	3.0	—	—
2.5-V LVCMOS	JESD8-5	3.0/2.5	3.0/2.5	2.5	2.5	2.5	—	—
1.8-V LVCMOS	JESD8-7	1.8/1.5	1.8/1.5	1.8	1.8	2.5	—	—
1.5-V LVCMOS	JESD8-11	1.8/1.5	1.8/1.5	1.5	1.5	2.5	—	—
1.2-V LVCMOS	JESD8-12	1.2	1.2	1.2	1.2	2.5	—	—
3.0-V PCI	PCI Rev 2.1	3.0	3.0	3.0	3.0	3.0	—	—
3.0-V PCI-X	PCI-X Rev 1.0	3.0	3.0	3.0	3.0	3.0	—	—
SSTL-2 Class I	JESD8-9B	*(2)*	*(2)*	2.5	2.5	2.5	1.25	1.25
SSTL-2 Class II	JESD8-9B	*(2)*	*(2)*	2.5	2.5	2.5	1.25	1.25
SSTL-18 Class I	JESD8-15	*(2)*	*(2)*	1.8	1.8	2.5	0.90	0.90
SSTL-18 Class II	JESD8-15	*(2)*	*(2)*	1.8	1.8	2.5	0.90	0.90
SSTL-15 Class I	—	*(2)*	*(2)*	1.5	1.5	2.5	0.75	0.75
SSTL-15 Class II	—	*(2)*	*(2)*	1.5	—	2.5	0.75	0.75
HSTL-18 Class I	JESD8-6	*(2)*	*(2)*	1.8	1.8	2.5	0.90	0.90
HSTL-18 Class II	JESD8-6	*(2)*	*(2)*	1.8	1.8	2.5	0.90	0.90

Table 6–2. Stratix IV I/O Standards and Voltage Levels *(Note 1)*, *(2)* (Part 2 of 3)

I/O Standard	Standard Support	V_{CCIO} (V)				V_{CCPD} (V) (Pre-Driver Voltage)	V_{REF} (V) (Input Ref Voltage)	V_{TT} (V) (Board Termination Voltage)
		Input Operation		Output Operation				
		Column I/O Banks	Row I/O Banks	Column I/O Banks	Row I/O Banks			
HSTL-15 Class I	JESD8-6	*(2)*	*(2)*	1.5	1.5	2.5	0.75	0.75
HSTL-15 Class II	JESD8-6	*(2)*	*(2)*	1.5	—	2.5	0.75	0.75
HSTL-12 Class I	JESD8-16A	*(2)*	*(2)*	1.2	1.2	2.5	0.6	0.6
HSTL-12 Class II	JESD8-16A	*(2)*	*(2)*	1.2	—	2.5	0.6	0.6
Differential SSTL-2 Class I	JESD8-9B	*(2)*	*(2)*	2.5	2.5	2.5	—	1.25
Differential SSTL-2 Class II	JESD8-9B	*(2)*	*(2)*	2.5	2.5	2.5	—	1.25
Differential SSTL-18 Class I	JESD8-15	*(2)*	*(2)*	1.8	1.8	2.5	—	0.90
Differential SSTL-18 Class II	JESD8-15	*(2)*	*(2)*	1.8	1.8	2.5	—	0.90
Differential SSTL-15 Class I	—	*(2)*	*(2)*	1.5	1.5	2.5	—	0.75
Differential SSTL-15 Class II	—	*(2)*	*(2)*	1.5	—	2.5	—	0.75
Differential HSTL-18 Class I	JESD8-6	*(2)*	*(2)*	1.8	1.8	2.5	—	0.90
Differential HSTL-18 Class II	JESD8-6	*(2)*	*(2)*	1.8	1.8	2.5	—	0.90
Differential HSTL-15 Class I	JESD8-6	*(2)*	*(2)*	1.5	1.5	2.5	—	0.75
Differential HSTL-15 Class II	JESD8-6	*(2)*	*(2)*	1.5	—	2.5	—	0.75
Differential HSTL-12 Class I	JESD8-16A	*(2)*	*(2)*	1.2	1.2	2.5	—	0.60
Differential HSTL-12 Class II	JESD8-16A	*(2)*	*(2)*	1.2	—	2.5	—	0.60
LVDS *(4)*, *(5)*	ANSI/TIA/EIA-644	*(2)*	*(2)*	2.5	2.5	2.5	—	—
RSDS *(6)*, *(7)*	—	*(2)*	*(2)*	2.5	2.5	2.5	—	—
mini-LVDS *(6)*, *(7)*	—	*(2)*	*(2)*	2.5	2.5	2.5	—	—

Table 6–2. Stratix IV I/O Standards and Voltage Levels *(Note 1), (2)* (Part 3 of 3)

I/O Standard	Standard Support	V_{CCIO} (V)				V_{CCPD} (V) (Pre-Driver Voltage)	V_{REF} (V) (Input Ref Voltage)	V_{TT} (V) (Board Termination Voltage)
		Input Operation		Output Operation				
		Column I/O Banks	Row I/O Banks	Column I/O Banks	Row I/O Banks			
LVPECL	—	*(4)*	2.5	—	—	2.5	—	—

Notes to Table 6–2:

(1) V_{CCPD} is either 2.5 or 3.0 V. For V_{CCIO} = 3.0 V, V_{CCPD} = 3.0 V. For V_{CCIO} = 2.5 V or less, V_{CCPD} = 2.5 V.

(2) Single-ended HSTL/SSTL, differential SSTL/HSTL, and LVDS input buffers are powered by V_{CCPD}. Row I/O banks support both true differential input buffers and true differential output buffers. Column I/O banks support true differential input buffers, but not true differential output buffers. I/O pins are organized in pairs to support differential standards. Column I/O differential HSTL and SSTL inputs use LVDS differential input buffers without on-chip R_D support.

(3) For more information about the 3.3-V LVTTL/LVCMOS standard supported in Stratix IV devices, refer to "3.3-V I/O Interface" on page 6–19.

(4) Column I/O banks support LVPECL I/O standards for input clock operation. Clock inputs on column I/O are powered by $V_{CCCLKIN}$ when configured as differential clock inputs. They are powered by V_{CCIO} when configured as single-ended clock inputs. Differential clock inputs in row I/O are powered by V_{CCPD}.

(5) Column and row I/O banks support LVDS outputs using two single-ended output buffers, an external one-resistor (LVDS_E_1R), and a three-resistor (LVDS_E_3R) network.

(6) Row I/O banks support RSDS and mini-LVDS I/O standards using true LVDS output buffer without a resistor network.

(7) Column and row I/O banks support RSDS and mini-LVDS I/O standards using two single-ended output buffers with one-resistor (RSDS_E_1R and mini-LVDS_E_1R) and three-resistor (RSDS_E_3R and mini-LVDS_E_3R) networks.

For more information about electrical characteristics of each I/O standard, refer to the *DC and Switching Characteristics* chapter.

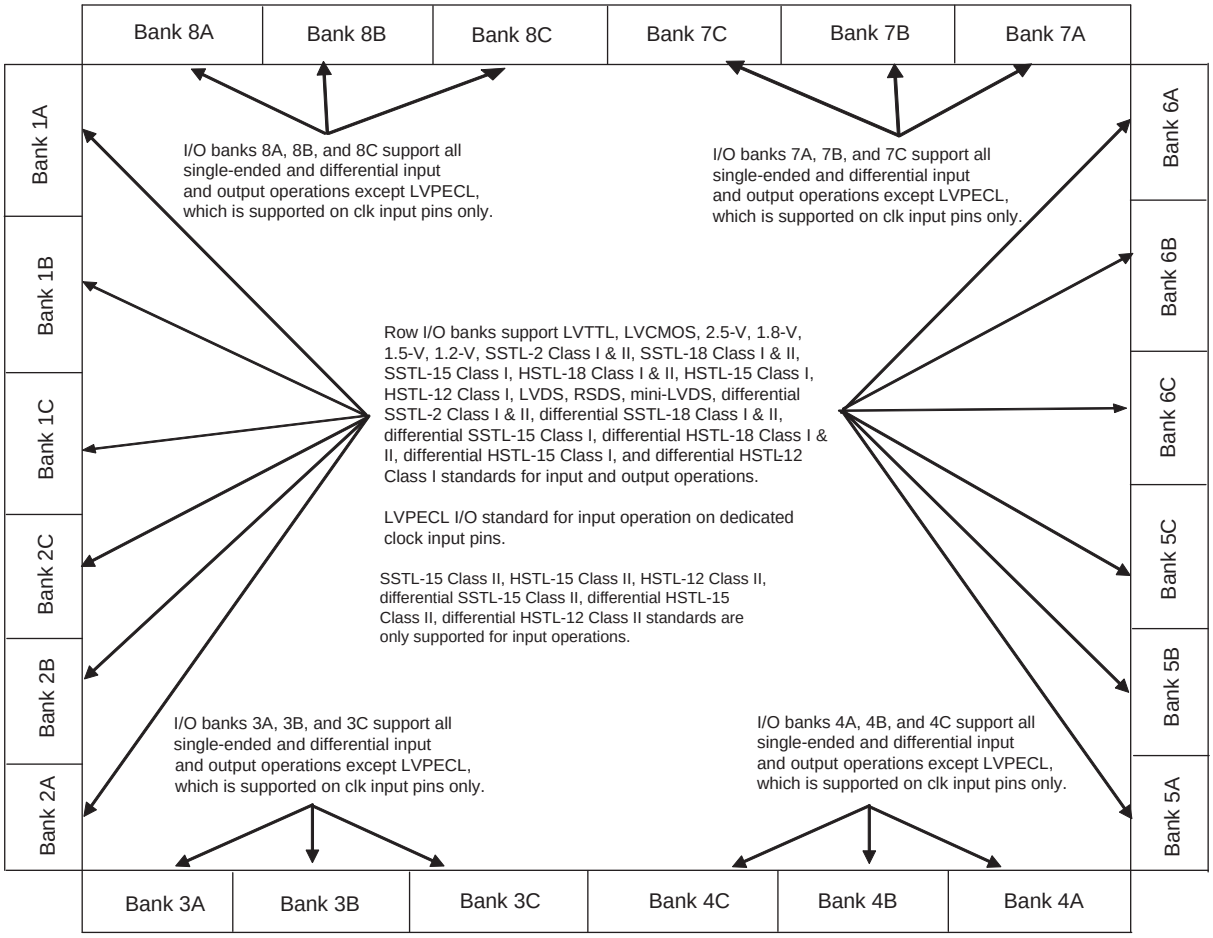
I/O Banks

Stratix IV devices contain up to 24 I/O banks, as shown in Figure 6–1 and Figure 6–2. The row I/O banks contain true differential input and output buffers and dedicated circuitry to support differential standards at speeds up to 1.6 Gbps.

Each I/O bank in Stratix IV devices can support high-performance external memory interfaces with dedicated circuitry. The I/O pins are organized in pairs to support differential standards. Each I/O pin pair can support both differential input and output buffers. The only exceptions are the `clk[1,3,8,10]`, `PLL_L[1,4]_clk`, and `PLL_R[1,4]_clk` pins, which support differential input operations only.

For the number of channels available for the LVDS I/O standard, refer to the *High-Speed Differential I/O Interface with DPA* chapter. For more information about transceiver-bank-related features, refer to the *Stratix IV Transceiver Architecture* chapter.

Figure 6–1. Stratix IV E Devices I/O Banks *(Note 1), (2), (3), (4), (5), (6), (7), (8)*



Notes to Figure 6–1:

(1) Differential HSTL and SSTL outputs are not true differential outputs. They use two single-ended outputs with the second output programmed as inverted.

(2) Column I/O differential HSTL and SSTL inputs use LVDS differential input buffers without differential OCT support.

(3) Column I/O supports LVDS outputs using single-ended buffers and external resistor networks.

(4) Column I/O supports PCI/PCI-X with on-chip clamp diode. Row I/O supports PCI/PCI-X with external clamp diode.

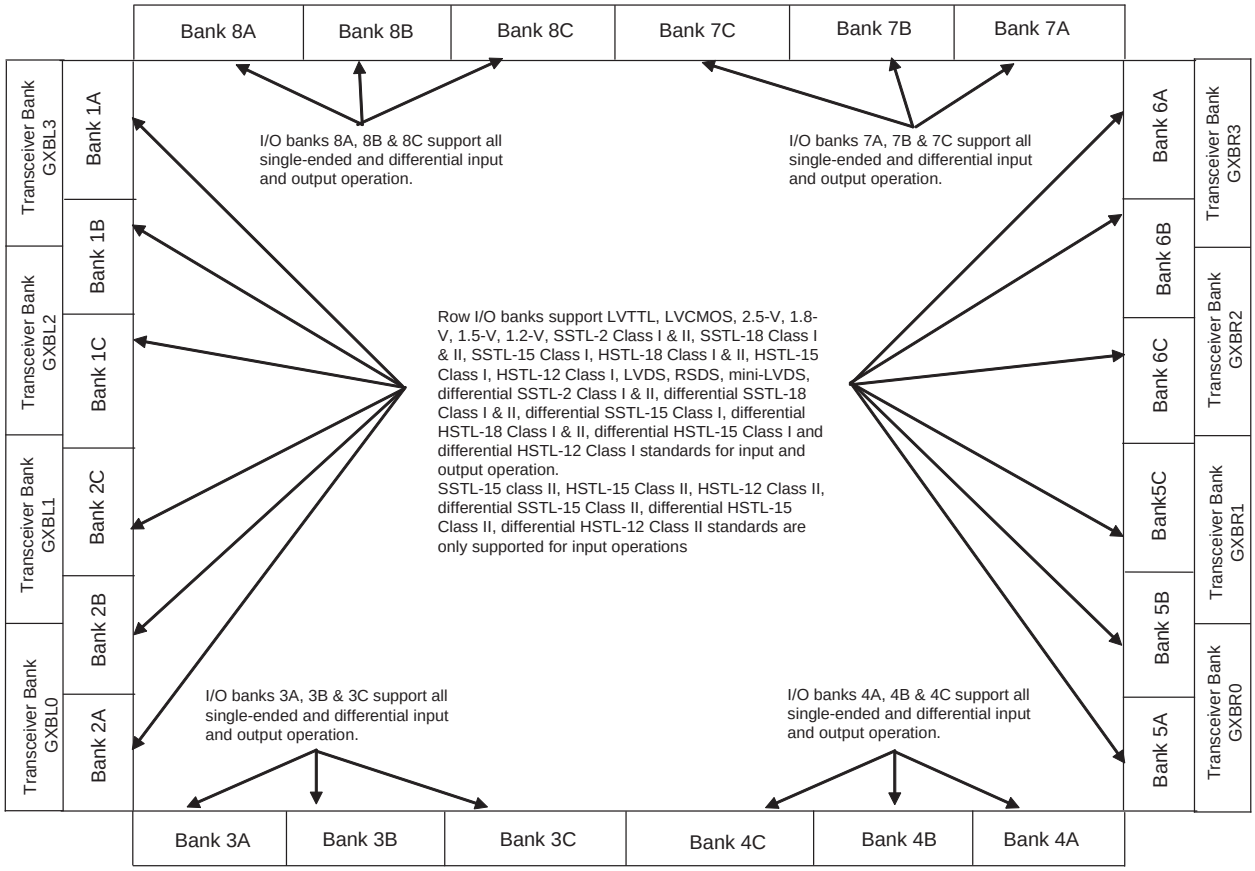
(5) Clock inputs on column I/O are powered by $V_{CCCLKIN}$ when configured as differential clock inputs. They are powered by V_{CCIO} when configured as single-ended clock inputs. All outputs use the corresponding bank V_{CCIO}.

(6) Row I/O supports the true LVDS output buffer.

(7) Column and row I/O banks support LVPECL standards for input clock operation.

(8) Figure 6–1 is a top view of the silicon die that corresponds to a reverse view for flip chip packages. It is a graphical representation only.

Figure 6–2. Stratix IV GX Devices I/O Banks *(Note 1), (2), (3), (4), (5), (6), (7), (8)*



Notes to Figure 6–2:

(1) Differential HSTL and SSTL outputs are not true differential outputs. They use two single-ended outputs with the second output programmed as inverted.

(2) Column I/O differential HSTL and SSTL inputs use LVDS differential input buffers without differential OCT support.

(3) Column I/O supports LVDS outputs using single-ended buffers and external resistor networks.

(4) Column I/O supports PCI/PCI-X with on-chip clamp diode. Row I/O supports PCI/PCI-X with external clamp diode.

(5) Clock inputs on column I/O are powered by $V_{CCCLKIN}$ when configured as differential clock inputs. They are powered by V_{CCIO} when configured as single-ended clock inputs. All outputs use the corresponding bank V_{CCIO}.

(6) Row I/O supports the true LVDS output buffer.

(7) Column and row I/O banks support LVPECL standards for input clock operation.

(8) Figure 6–2 is a top view of the silicon die that corresponds to a reverse view for flip chip packages. It is a graphical representation only.

Modular I/O Banks

The I/O pins in Stratix IV devices are arranged in groups called modular I/O banks. Depending on device densities, the number of Stratix IV device I/O banks range from 16 to 24. The number of I/O pins on each bank is 24, 32, 36, 40, or 48. Figure 6–4 through Figure 6–16 show the number of I/O pins available in each I/O bank.

In Stratix IV devices, the maximum number of I/O banks per side is either four or six, depending on the device density. When migrating between devices with a different number of I/O banks per side, it is the middle or "B" bank that is removed or inserted. For example, when moving from a 24-bank device to a 16-bank device, the banks that are dropped are "B" banks, namely: 1B, 2B, 3B, 4B, 5B, 6B, 7B, and 8B. Similarly, when moving from a 16-bank device to a 24-bank device, the banks that are added are the same "B" banks.

After migration from a smaller device to a larger device, the bank size increases or remains the same, but never decreases. For example, the number of I/O pins to a bank may increase from 24 to 26, 32, 36, 40, 42, or 48, but will never decrease. This is shown in Figure 6–3.

Figure 6–3. Bank Migration Path with Increasing Device Size

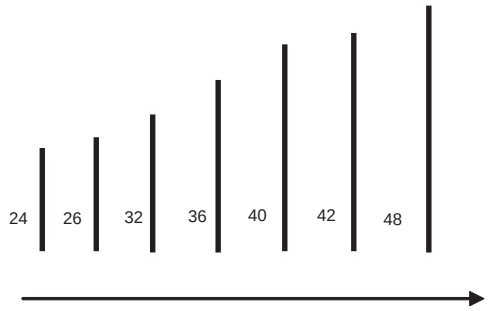


Figure 6–4 through Figure 6–16 show the number of I/O pins and packaging information for different sets of available devices. Figure 6–4 through Figure 6–16 shows the top view of the silicon die that corresponds to a reverse view for flip chip packages. They are graphical representations only.

For Figure 6–4 through Figure 6–16, the pin count includes all general purpose I/Os, dedicated clock pins, and dual purpose configuration pins. Transceiver pins and dedicated configuration pins are not included in the pin count.

Figure 6–4. Number of I/Os in Each Bank in EP4SE230 and EP4SE360 Devices in the 780-Pin FineLine BGA Package

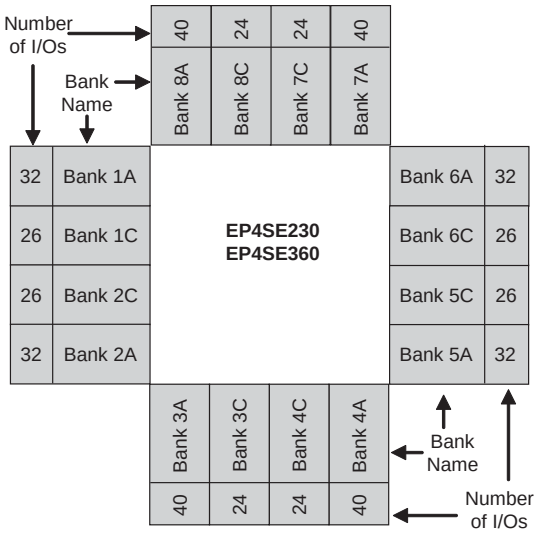


Figure 6–5. Number of I/Os in Each Bank in EP4SE360, EP4SE530, and EP4SE820 Devices in the 1152-Pin FineLine BGA Package

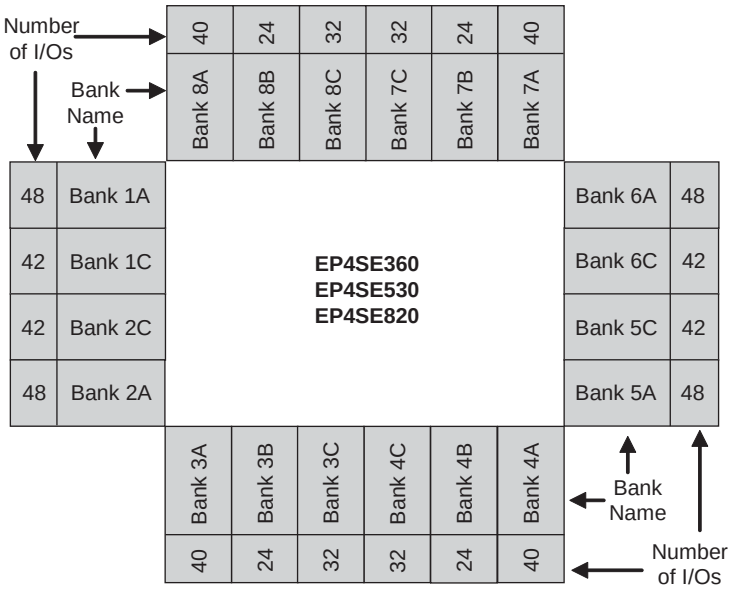


Figure 6–6. Number of I/Os in Each Bank in EP4SE530 and EP4SE820 Devices in the 1517-Pin FineLine BGA Package

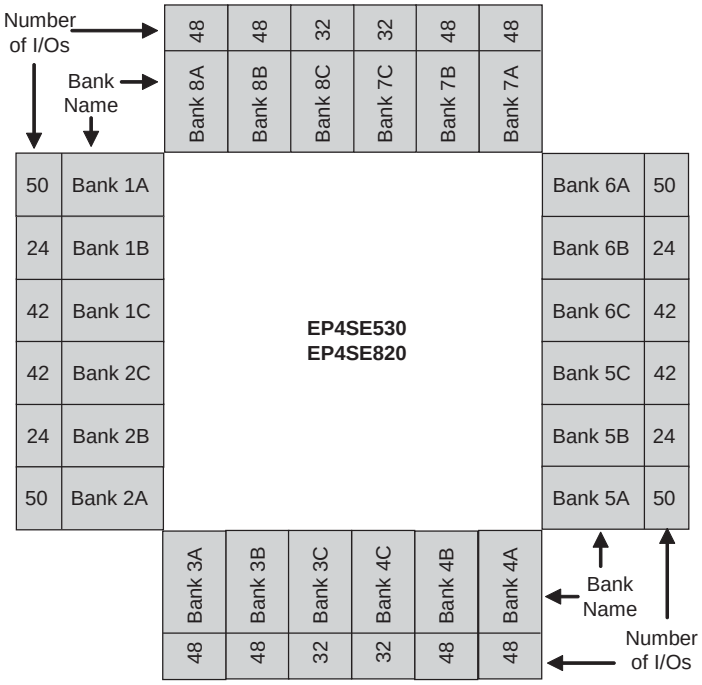


Figure 6–7. Number of I/Os in Each Bank in EP4SE530 and EP4SE820 Devices in the 1760-Pin Fineline BGA Package

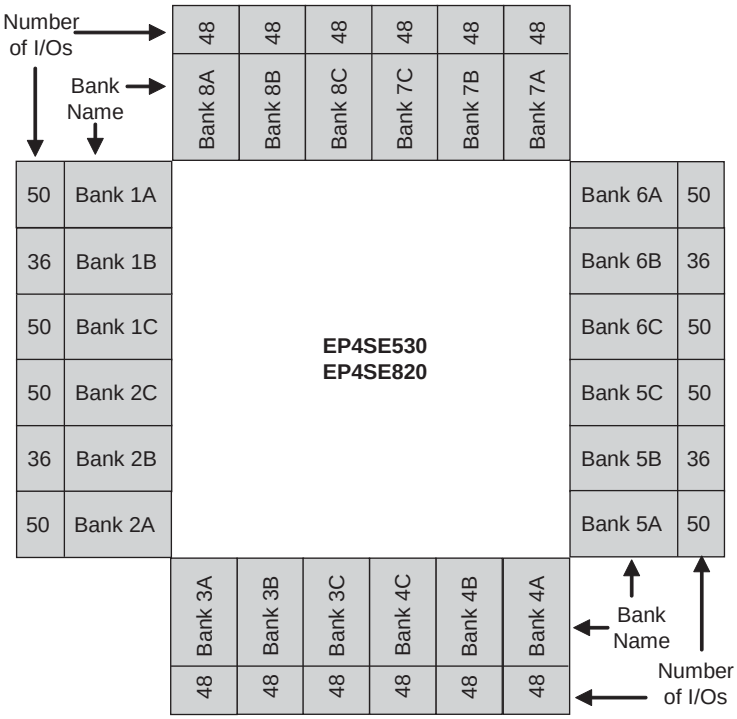


Figure 6–8. Number of I/Os in Each Bank in EP4SGX70, EP4SGX110, EP4SGX180, and EP4SGX230 Devices in the 780-Pin FineLine BGA Package

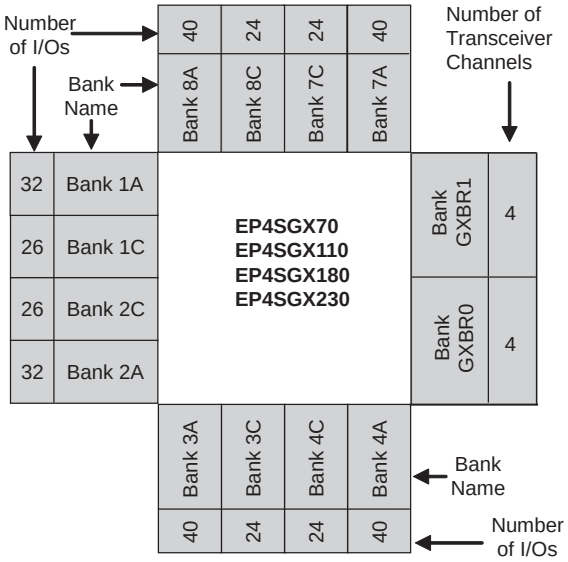


Figure 6–9. Number of I/Os in Each Bank in EP4SGX290 and EP4SGX360 Devices in the 780-Pin FineLine BGA Package

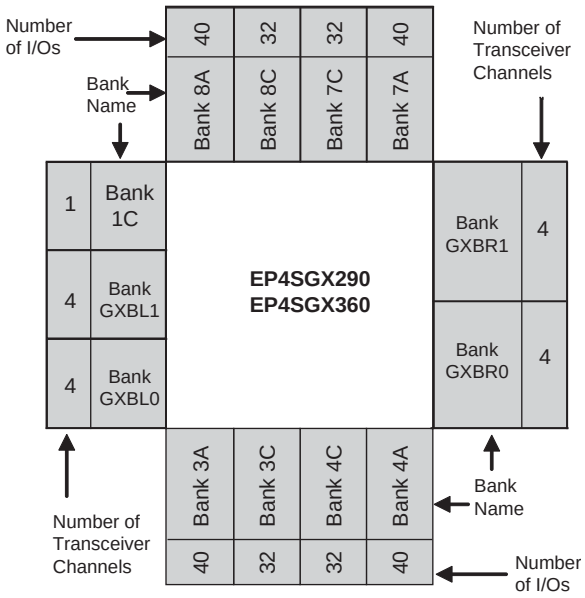


Figure 6–10. Number of I/Os in Each Bank in EP4SGX70 and EP4SGX110 Devices in the 1152-Pin FineLine BGA Package

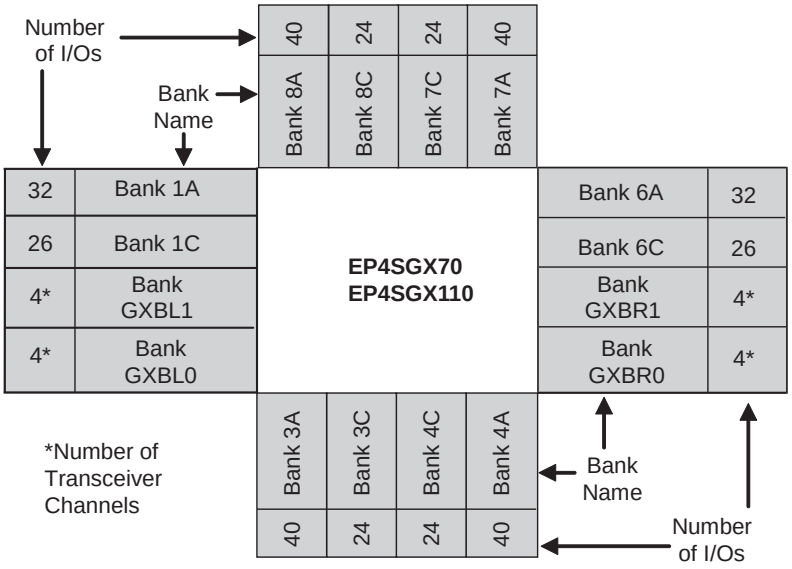
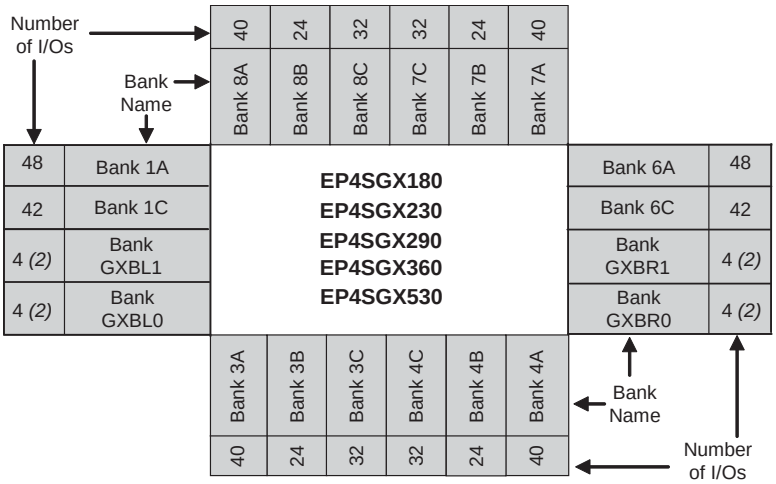


Figure 6–11. Number of I/Os in Each Bank in EP4SGX180, EP4SGX230, EP4SGX290, EP4SGX360, and EP4SGX530 Devices in the 1152-Pin FineLine BGA Package *(Note 1)*, *(2)*

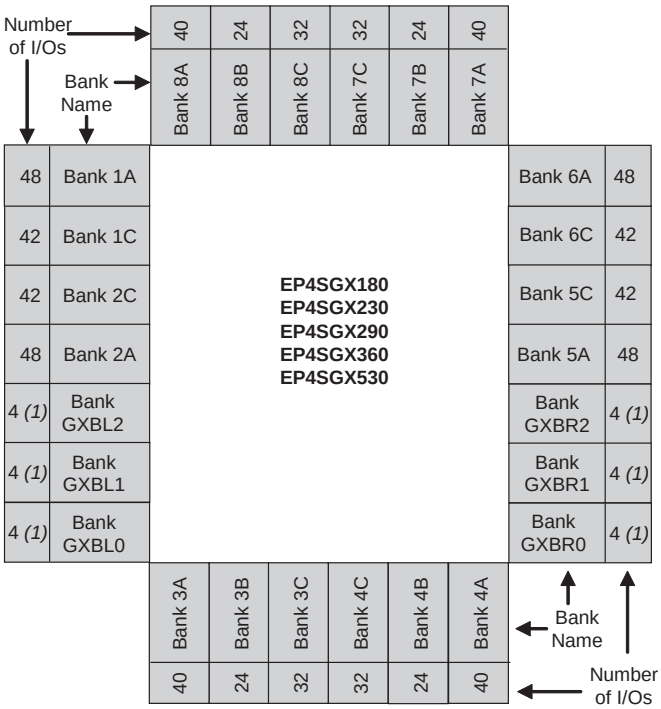


Notes to Figure 6–11:

(1) Except for the EP4SGX530 device, all listed devices have two variants in the F1152 package option—one with no PMA-only transceiver channels and the other with two PMA-only transceiver channels for each transceiver bank. The EP4SGX530 device is only offered with two PMA-only transceiver channels for each transceiver bank in the F1152 package option.

(2) There are two additional PMA-only transceiver channels in each transceiver bank for devices with the PMA-only transceiver package option.

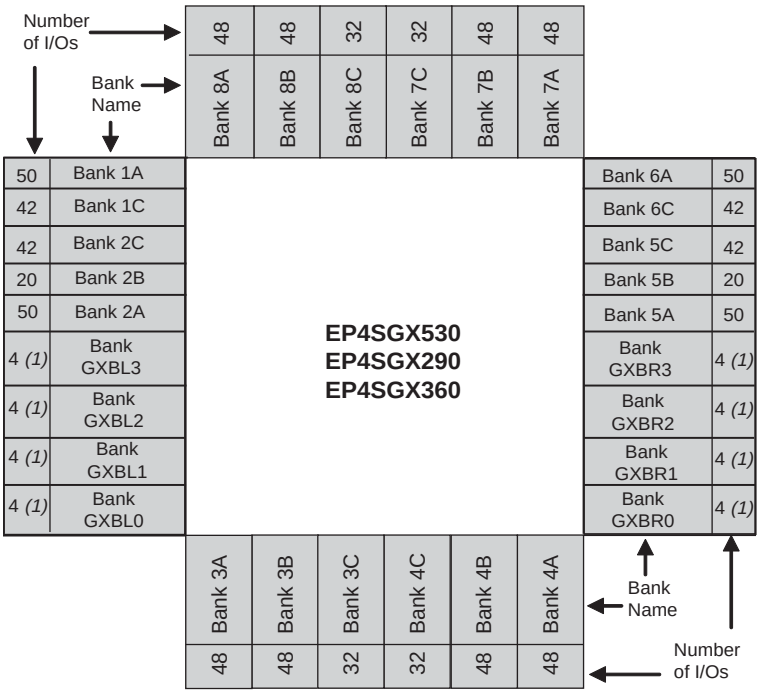
Figure 6–12. Number of I/Os in Each Bank in EP4SGX180, EP4SGX230, EP4SGX290, EP4SGX360, and EP4SGX530 Devices in the 1517-Pin FineLine BGA Package *(Note 1)*



Note to Figure 6–12:

(1) There are two additional PMA-only transceiver channels in each transceiver bank.

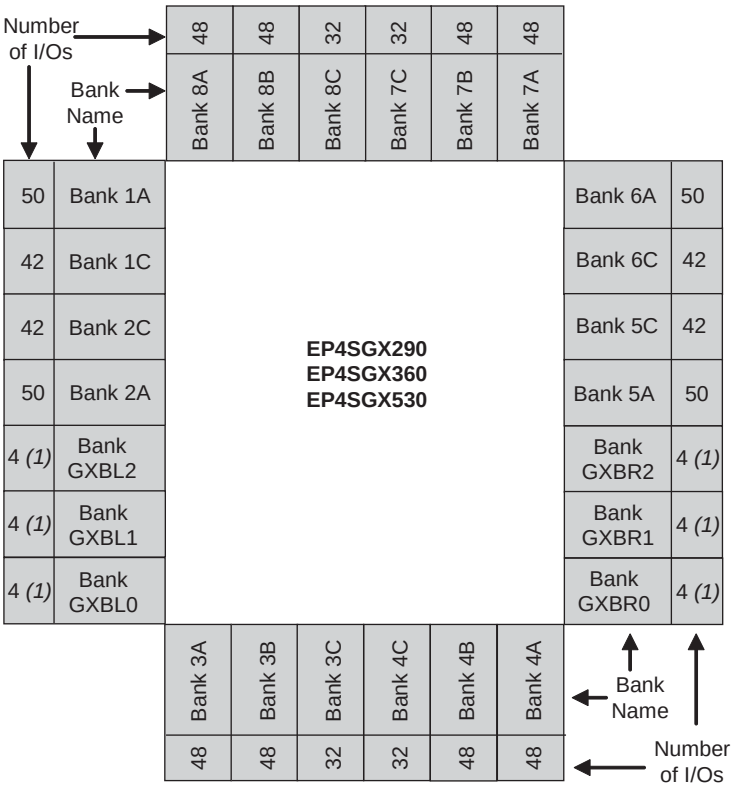
Figure 6–13. Number of I/Os in Each Bank in EP4SGX290, EP4SGX360, and EP4SGX530 Devices in the 1932-Pin FineLine BGA Package *(Note 1)*



Note to Figure 6–13:

(1) There are two additional PMA-only transceiver channels in each transceiver bank.

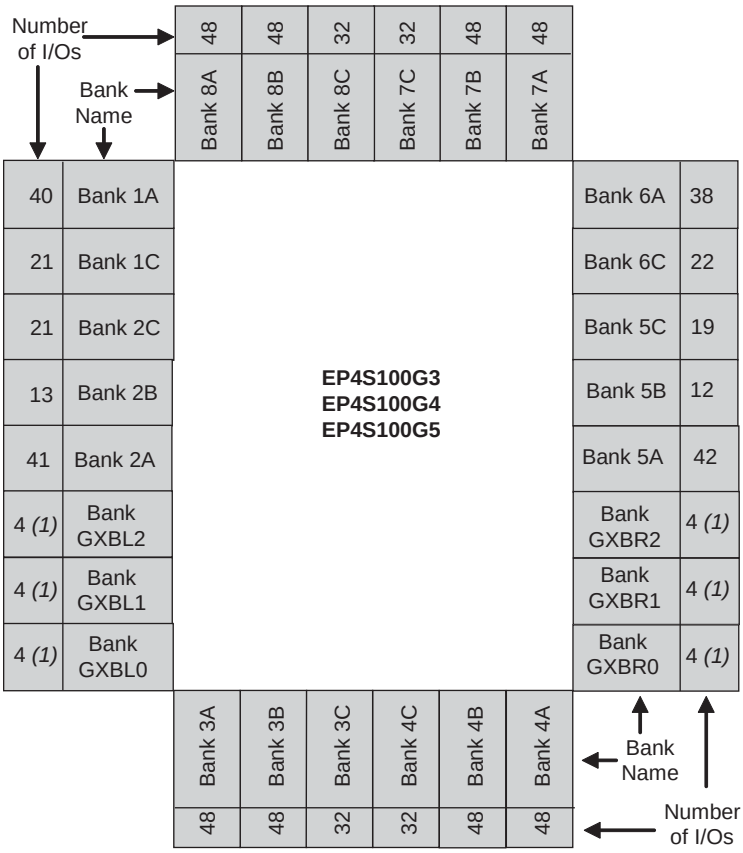
Figure 6–14. Number of I/Os in Each Bank in EP4SGX290, EP4SGX360, and EP4SGX530 Devices in the 1760-Pin FineLine BGA Package *(Note 1)*



Note to Figure 6–14:

(1) There are two additional PMA-only transceiver channels in each transceiver bank.

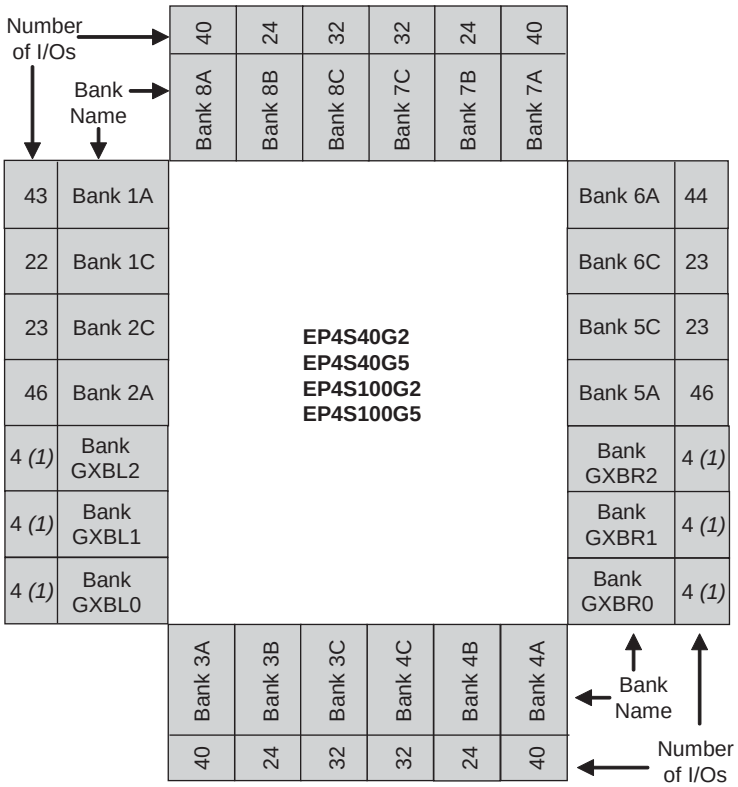
Figure 6–15. Number of I/Os in Each Bank in EP4S100G3, EP4S100G4, and EP4S100G5 Devices in the 1932-Pin FineLine BGA Package *(Note 1)*



Note to Figure 6–15:

(1) There are two additional PMA-only transceiver channels in each transceiver bank.

Figure 6–16. Number of I/Os in Each Bank in EP4S40G2, EP4S40G5, EP4S100G2, and EP4S100G5 Devices in the 1517-Pin FineLine BGA Package *(Note 1)*



Note to Figure 6–16:

(1) There are two additional PMA-only transceiver channels in each transceiver bank.

I/O Structure

The I/O element (IOE) in Stratix IV devices contain a bidirectional I/O buffer and I/O registers to support a complete embedded bidirectional single data rate or DDR transfer. The IOEs are located in I/O blocks around the periphery of the Stratix IV device. There are up to four IOEs per row I/O block and four IOEs per column I/O block. The row IOEs drive row, column, or direct link interconnects. The column IOEs drive column interconnects.

The Stratix IV bidirectional IOE also supports the following features:

- Programmable input delay
- Programmable output-current strength
- Programmable slew rate
- Programmable output delay
- Programmable bus-hold
- Programmable pull-up resistor
- Open-drain output
- On-chip series termination with calibration

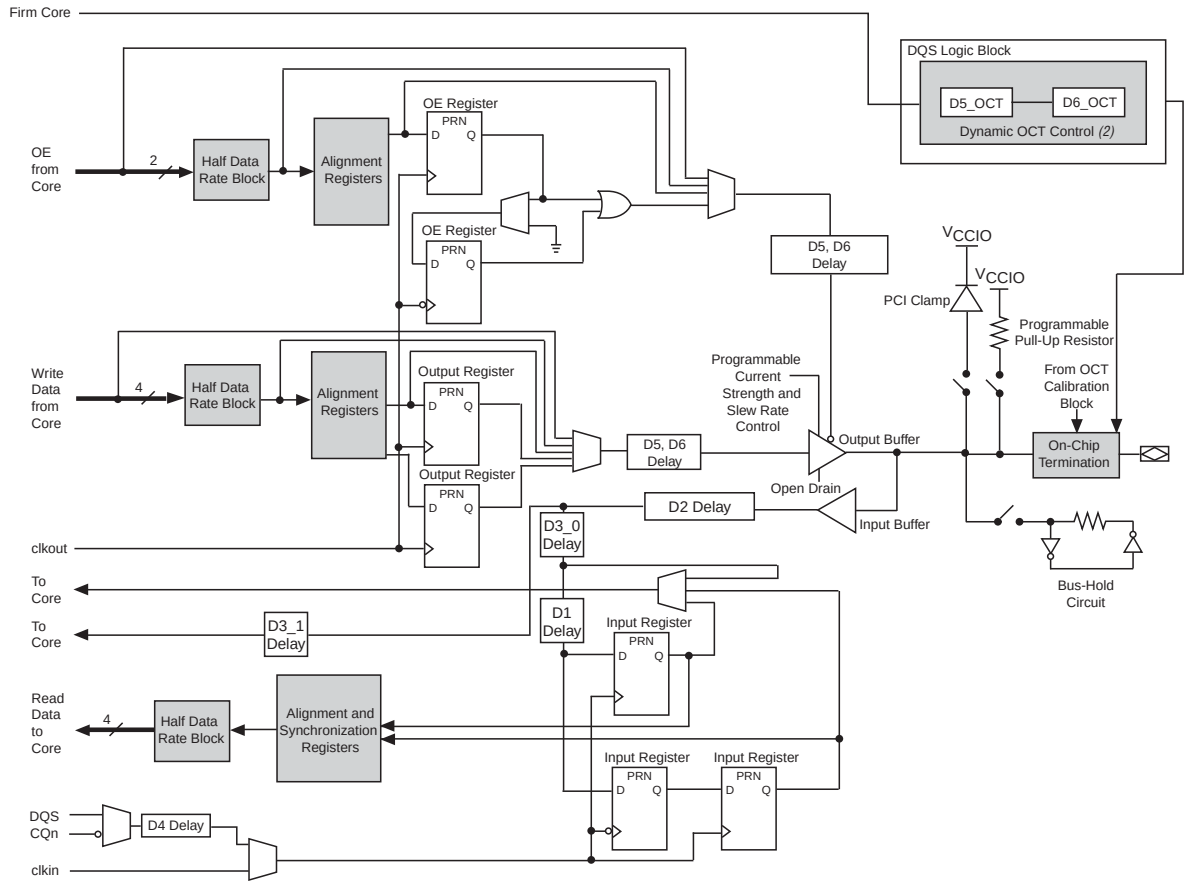
- On-chip series termination without calibration
- On-chip parallel termination with calibration
- On-chip differential termination
- PCI clamping diode

I/O registers are composed of the input path for handling data from the pin to the core, the output path for handling data from the core to the pin, and the output-enable (OE) path for handling the OE signal to the output buffer. These registers allow faster source-synchronous register-to-register transfers and resynchronization. The input path consists of the DDR input registers, alignment and synchronization registers, and HDR. You can bypass each block of the input path.

The output and OE paths are divided into output or OE registers, alignment registers, and HDR blocks. You can bypass each block of the output and OE paths.

Figure 6–17 shows the Stratix IV IOE structure.

Figure 6–17. Stratix IV IOE Structure *(Note 1)*, *(2)*



Notes to Figure 6–17:

(1) The `D3_0` and `D3_1` delays have the same available settings in the Quartus II software.

(2) One dynamic OCT control is available per DQ/DQS group.

For more information about I/O registers and how they are used for memory applications, refer to the *External Memory Interfaces* chapter.

3.3-V I/O Interface

Stratix IV I/O buffers support 3.3-V I/O standards. You can use them as transmitters or receivers in your system. The output high voltage (V_{OH}), output low voltage (V_{OL}), input high voltage (V_{IH}), and input low voltage (V_{IL}) levels meet the 3.3-V I/O standards specifications defined by EIA/JEDEC Standard JESD8-B with margin when the Stratix IV V_{CCIO} voltage is powered by 3.0 V.

To ensure device reliability and proper operation, when interfacing with a 3.3-V I/O system using Stratix IV devices, ensure that you do not violate the absolute maximum ratings of the devices. Altera recommends performing IBIS simulation to determine that the overshoot and undershoot voltages are within the guidelines.

When using the Stratix IV device as a transmitter, you can use slow slew rate and series termination to limit overshoot and undershoot at the I/O pins, but they are not required. Transmission line effects that cause large voltage deviations at the receiver are associated with an impedance mismatch between the driver and the transmission lines. By matching the impedance of the driver to the characteristic impedance of the transmission line, you can significantly reduce overshoot voltage. You can use a series termination resistor placed physically close to the driver to match the total driver impedance to the transmission line impedance. Stratix IV devices support series OCT for all LVTTL and LVCMOS I/O standards in all I/O banks.

When using the Stratix IV device as a receiver, you can use a clamping diode (on-chip or off-chip) to limit overshoot, though this is not required. Stratix IV devices provide an optional on-chip PCI-clamping diode for column I/O pins. You can use this diode to protect the I/O pins against overshoot voltage.

The 3.3-V I/O standard is supported using bank supply voltage (V_{CCIO}) at 3.0 V. In this method, the clamping diode (on-chip or off-chip), when enabled, can sufficiently clamp overshoot voltage to within the DC and AC input voltage specifications. The clamped voltage can be expressed as the sum of the supply voltage (V_{CCIO}) and the diode forward voltage.

For more information about the absolute maximum rating and maximum allowed overshoot during transitions, refer to the *DC and Switching Characteristics* chapter.

External Memory Interfaces

In addition to the I/O registers in each IOE, Stratix IV devices also have dedicated registers and phase-shift circuitry on all I/O banks for interfacing with external memory interfaces.

For more information about external memory interfaces, refer to the *External Memory Interfaces* chapter.

High-Speed Differential I/O with DPA Support

Stratix IV devices have the following dedicated circuitry for high-speed differential I/O support:

- Differential I/O buffer
- Transmitter serializer
- Receiver deserializer

- Data realignment
- Dynamic phase aligner (DPA)
- Synchronizer (FIFO buffer)
- Phase-locked loops (PLLs)

For more information about DPA support, refer to the *High-Speed Differential I/O Interfaces with DPA* chapter.

Current Strength

The output buffer for each Stratix IV device I/O pin has a programmable current strength control for certain I/O standards. You can use programmable current strength to mitigate the effects of high signal attenuation due to a long transmission line or a legacy backplane. The LVTTL, LVCMOS, SSTL, and HSTL standards have several levels of current strength that you can control. Table 6–3 lists the programmable current strength for Stratix IV devices.

Table 6–3. Programmable Current Strength *(Note 1), (2)*

I/O Standard	I_{OH} / I_{OL} Current Strength Setting (mA) for Column I/O Pins	I_{OH} / I_{OL} Current Strength Setting (mA) for Row I/O Pins
3.3-V LVTTL	16, 12, 8, 4	12, 8, 4
3.3-V LVCMOS	16, 12, 8, 4	8, 4
2.5-V LVCMOS	16, 12, 8, 4	12, 8, 4
1.8-V LVCMOS	12, 10, 8, 6, 4, 2	8, 6, 4, 2
1.5-V LVCMOS	12, 10, 8, 6, 4, 2	8, 6, 4, 2
1.2-V LVCMOS	8, 6, 4, 2	4, 2
SSTL-2 Class I	12, 10, 8	12, 8
SSTL-2 Class II	16	16
SSTL-18 Class I	12, 10, 8, 6, 4	12, 10, 8, 6, 4
SSTL-18 Class II	16, 8	16, 8
SSTL-15 Class I	12, 10, 8, 6, 4	8, 6, 4
SSTL-15 Class II	16, 8	—
HSTL-18 Class I	12, 10, 8, 6, 4	12, 10, 8, 6, 4
HSTL-18 Class II	16	16
HSTL-15 Class I	12, 10, 8, 6, 4	8, 6, 4
HSTL-15 Class II	16	—
HSTL-12 Class I	12, 10, 8, 6, 4	8, 6, 4
HSTL-12 Class II	16	—

Notes to Table 6–3:

(1) The default setting in the Quartus II software is 50-ΩOCT R_S without calibration for all non-voltage reference and HSTL and SSTL Class I I/O standards. The default setting is 25-ΩOCT R_S without calibration for HSTL and SSTL Class II I/O standards.

(2) The 3.3-V LVTTL and 3.3-V LVCMOS are supported using V_{CCIO} and V_{CCPD} at 3.0 V.

Altera recommends performing IBIS or SPICE simulations to determine the best current strength setting for your specific application.

Slew Rate Control

The output buffer for each Stratix IV device regular- and dual-function I/O pin has a programmable output slew-rate control that you can configure for low-noise or high-speed performance. A faster slew rate provides high-speed transitions for high-performance systems. A slower slew rate can help reduce system noise, but adds a nominal delay to the rising and falling edges. Each I/O pin has an individual slew-rate control, allowing you to specify the slew rate on a pin-by-pin basis.

You cannot use the programmable slew rate feature when using OCT R_S.

The Quartus II software allows four settings for programmable slew rate control—0, 1, 2, and 3—where 0 is slow slew rate and 3 is fast slew rate. Figure 6–4 lists the default slew rate settings from the Quartus II software.

Table 6–4. Default Slew Rate Settings

I/O Standard	Slew Rate Option	Default Slew Rate
1.2-V, 1.5-V, 1.8-V, 2.5-V LVCMOS, and 3.3-V LVTTL/LVCMOS	0, 1, 2, 3	3
SSTL-2, SSTL-18, SSTL-15, HSTL-18, HSTL-15, and HSTL-12	0, 1, 2, 3	3
3.0-V PCI/PCI-X	0, 1, 2, 3	3
LVDS_E_1R, mini-LVDS_E_1R, and RSDS_E_1R	0, 1, 2, 3	3
LVDS_E_3R, mini-LVDS_E_3R, and RSDS_E_3R	0, 1, 2, 3	3

You can use faster slew rates to improve the available timing margin in memory-interface applications or when the output pin has high-capacitive loading.

Altera recommends performing IBIS or SPICE simulations to determine the best slew rate setting for your specific application.

I/O Delay

The following sections describe programmable IOE delay and programmable output buffer delay.

Programmable IOE Delay

The Stratix IV device IOE includes programmable delays, shown in Figure 6–17 on page 6–18, that you can activate to ensure zero hold times, minimize setup times, or increase clock-to-output times. Each pin can have a different input delay from pin-to-input register or a delay from output register-to-output pin values to ensure that the bus has the same delay going into or out of the device. This feature helps read and time margins because it minimizes the uncertainties between signals in the bus.

For more information about programmable IOE delay specifications, refer to the *DC and Switching Characteristics* chapter.

Programmable Output Buffer Delay

Stratix IV devices support delay chains built inside the single-ended output buffer, as shown in Figure 6–17 on page 6–18. The delay chains can independently control the rising and falling edge delays of the output buffer, providing the ability to adjust the output-buffer duty cycle, compensate channel-to-channel skew, reduce simultaneous switching output (SSO) noise by deliberately introducing channel-to-channel skew, and improve high-speed memory-interface timing margins. Stratix IV devices support four levels of output buffer delay settings. The default setting is **No Delay**.

For more information about programmable output buffer delay specifications, refer to the *DC and Switching Characteristics* chapter.

Open-Drain Output

Stratix IV devices provide an optional open-drain output (equivalent to an open collector output) for each I/O pin. When configured as open drain, the logic value of the output is either high-Z or 0. Typically, an external pull-up resistor is required to provide logic high.

Bus Hold

Each Stratix IV device I/O pin provides an optional bus-hold feature. Bus-hold circuitry can weakly hold the signal on an I/O pin at its last-driven state. Because the bus-hold feature holds the last-driven state of the pin until the next input signal is present, you do not need an external pull-up or pull-down resistor to hold a signal level when the bus is tri-stated.

Bus-hold circuitry also pulls non-driven pins away from the input threshold voltage where noise can cause unintended high-frequency switching. You can select this feature individually for each I/O pin. The bus-hold output drives no higher than V_{CCIO} to prevent over-driving signals. If you enable the bus-hold feature, you cannot use the programmable pull-up option. Disable the bus-hold feature if the I/O pin is configured for differential signals.

Bus-hold circuitry uses a resistor with a nominal resistance (R_{BH}) of approximately 7 kΩ to weakly pull the signal level to the last-driven state.

For more information about the specific sustaining current driven through this resistor and the overdrive current used to identify the next-driven input level, refer to the *DC and Switching Characteristics* chapter.

Bus-hold circuitry is active only after configuration. When going into user mode, the bus-hold circuit captures the value on the pin present at the end of configuration.

Pull-Up Resistor

Each Stratix IV device I/O pin provides an optional programmable pull-up resistor during user mode. If you enable this feature for an I/O pin, the pull-up resistor (typically 25 KΩ) weakly holds the I/O to the V_{CCIO} level.

Programmable pull-up resistors are only supported on user I/O pins and are not supported on dedicated configuration pins, JTAG pins, or dedicated clock pins. If you enable the programmable pull-up option, you cannot use the bus-hold feature.

Pre-Emphasis

Stratix IV LVDS transmitters support programmable pre-emphasis to compensate for the frequency dependent attenuation of the transmission line. The Quartus II software allows four settings for programmable pre-emphasis.

For more information about programmable pre-emphasis, refer to the *High-Speed Differential I/O Interfaces with DPA in Stratix IV Devices* chapter.

Differential Output Voltage

Stratix IV LVDS transmitters support programmable V_{OD}. The programmable V_{OD} settings allow you to adjust output eye height to optimize trace length and power consumption. A higher V_{OD} swing improves voltage margins at the receiver end; a smaller V_{OD} swing reduces power consumption. The Quartus II software allows four settings for programmable V_{OD}.

For more information about programmable V_{OD}, refer to the *High-Speed Differential I/O Interfaces with DPA in Stratix IV Devices* chapter.

MultiVolt I/O Interface

The Stratix IV architecture supports the MultiVolt I/O interface feature that allows the Stratix IV devices in all packages to interface with systems of different supply voltages.

You can connect the `VCCIO` pins to a 1.2-, 1.5-, 1.8-, 2.5-, or 3.0-V power supply, depending on the output requirements. The output levels are compatible with systems of the same voltage as the power supply. (For example, when `VCCIO` pins are connected to a 1.5-V power supply, the output levels are compatible with 1.5-V systems.)

For more information about pin connection guidelines, refer to the *Stratix IV Device Family Pin Connection Guidelines*.

The Stratix IV VCCPD power pins must be connected to a 2.5- or 3.0-V power supply. Using these power pins to supply the pre-driver power to the output buffers increases the performance of the output pins. Table 6–5 lists Stratix IV MultiVolt I/O support.

Table 6–5. Stratix IV MultiVolt I/O Support *(Note 1)*

V_{CCIO} (V)	Input Signal (V)						Output Signal (V)					
	1.2	1.5	1.8	2.5	3.0	3.3	1.2	1.5	1.8	2.5	3.0	3.3
1.2	✓	—	—	—	—	—	✓	—	—	—	—	—
1.5	—	✓	✓	—	—	—	—	✓	—	—	—	—
1.8	—	✓	✓	—	—	—	—	—	✓	—	—	—
2.5	—	—	—	✓	✓*(2)*	✓*(2)*	—	—	—	✓	—	—
3.0	—	—	—	✓	✓	✓	—	—	—	—	✓	—

Notes to Table 6–5:

(1) The pin current may be slightly higher than the default value. You must verify that the driving device's V_{OL} maximum and V_{OH} minimum voltages do not violate the applicable Stratix IV V_{IL} maximum and V_{IH} minimum voltage specifications.

(2) Altera recommends that you use an external clamping diode on the I/O pins when the input signal is 3.0 V or 3.3 V. You have the option to use an internal clamping diode for column I/O pins.

On-Chip Termination Support and I/O Termination Schemes

Stratix IV devices feature dynamic series and parallel OCT to provide I/O impedance matching and termination capabilities. OCT maintains signal quality, saves board space, and reduces external component costs.

Stratix IV devices support:

- On-chip series (R_S) with calibration
- On-chip series (R_S) without calibration
- Parallel (R_T) with calibration
- Dynamic series termination for single-ended I/O standards
- Parallel termination for single-ended I/O standards
- On-chip differential termination (R_D) for differential LVDS I/O standards

Stratix IV devices support OCT in all I/O banks by selecting one of the OCT I/O standards.

These devices also support OCT R_S and R_T in the same I/O bank for different I/O standards if they use the same V_{CCIO} supply voltage. You can independently configure each I/O in an I/O bank to support OCT R_S, programmable current strength, or OCT R_T.

You cannot configure both OCT R_S and programmable current strength for the same I/O buffer.

A pair of RUP and RDN pins are available in a given I/O bank and are shared for series- and parallel-calibrated termination. The RUP and RDN pins share the same V_{CCIO} and GND, respectively, with the I/O bank where they are located. The RUP and RDN pins are dual-purpose I/Os and function as regular I/Os if you do not use the calibration circuit.

For calibration, connections are as follows:

- The RUP pin is connected to V_{CCIO} through an external 25-Ω ±1% or 50-Ω ±1% resistor for an on-chip series termination value of 25-Ω or 50-Ω, respectively.
- The RDN pin is connected to GND through an external 25-Ω ±1% or 50-Ω ±1% resistor for an on-chip series termination value of 25-Ω or 50-Ω, respectively.

For on-chip parallel termination, connections are as follows:

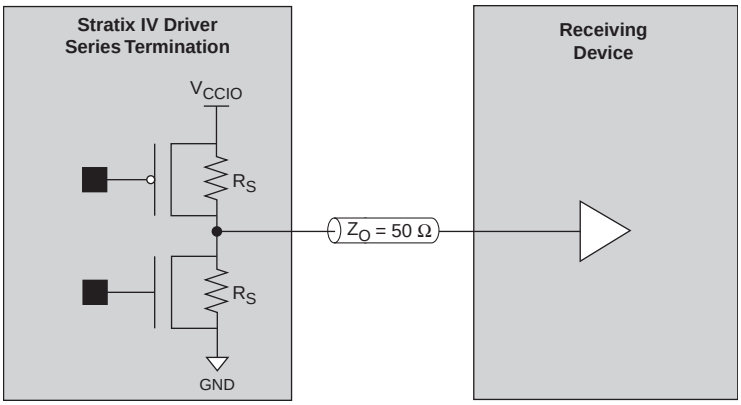
- The RUP pin is connected to V_{CCIO} through an external 50-Ω ±1% resistor.
- The RDN pin is connected to GND through an external 50-Ω ±1% resistor.

On-Chip Series (R_S) Termination Without Calibration

Stratix IV devices support driver-impedance matching to provide the I/O driver with controlled output impedance that closely matches the impedance of the transmission line. As a result, you can significantly reduce reflections. Stratix IV devices support on-chip series termination for single-ended I/O standards (Figure 6–18).

The R_S shown in Figure 6–18 is the intrinsic impedance of the output transistors. The typical R_S values are 25 Ω and 50 Ω. When you select matching impedance, current strength is no longer selectable.

Figure 6–18. On-Chip Series Termination Without Calibration



To use on-chip termination for the SSTL Class I standard, you must select the **50-Ω on-chip series termination** setting, thus eliminating the external 25-Ω R_S (to match the 50-Ω transmission line). For the SSTL Class II standard, you must select the **25-Ω on-chip series termination** setting (to match the 50-Ω transmission line and the near-end external 50-Ω pull-up to V_{TT}).

On-Chip Series Termination with Calibration

Stratix IV devices support on-chip series termination with calibration in all banks. The on-chip series termination calibration circuit compares the total impedance of the I/O buffer to the external 25-Ω ±1% or 50-Ω ±1% resistors connected to the RUP and RDN pins and dynamically enables or disables the transistors until they match.

The R_S shown in Figure 6–19 is the intrinsic impedance of the transistors. Calibration occurs at the end of device configuration. When the calibration circuit finds the correct impedance, it powers down and stops changing the characteristics of the drivers.

Figure 6–19. On-Chip Series Termination with Calibration

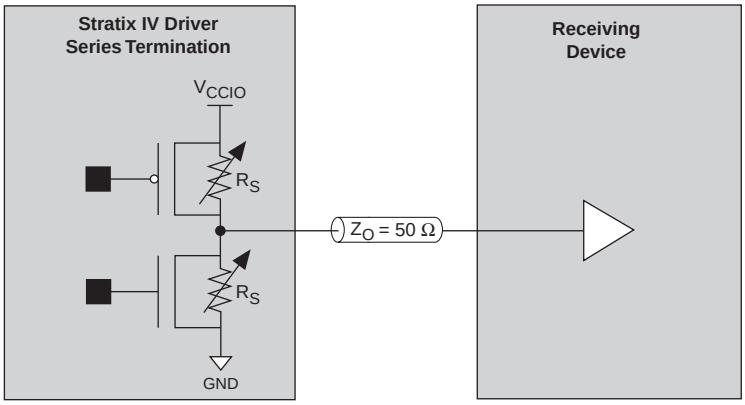


Table 6–6 lists the I/O standards that support on-chip series termination with calibration.

Table 6–6. Selectable I/O Standards for On-Chip Series Termination with and Without Calibration (Part 1 of 2)

I/O Standard	On-Chip Series Termination Setting	
	Row I/O (Ω)	Column I/O (Ω)
3.3-V LVTTL/LVCMOS	50	50
	25	25
2.5-V LVCMOS	50	50
	25	25
1.8-V LVCMOS	50	50
	25	25
1.5-V LVCMOS	50	50
		25
1.2-V LVCMOS	50	50
		25
SSTL-2 Class I	50	50
SSTL-2 Class II	25	25
SSTL-18 Class I	50	50
SSTL-18 Class II	25	25
SSTL-15 Class I	50	50

Table 6–6. Selectable I/O Standards for On-Chip Series Termination with and Without Calibration (Part 2 of 2)

I/O Standard	On-Chip Series Termination Setting	
	Row I/O (Ω)	Column I/O (Ω)
SSTL-15 Class II	—	25
HSTL-18 Class I	50	50
HSTL-18 Class II	25	25
HSTL-15 Class I	50	50
HSTL-15 Class II	—	25
HSTL-12 Class I	50	50
HSTL-12 Class II	—	25

Left-Shift Series Termination Control

Stratix IV devices support left-shift series termination control. You can use left-shift series termination control to get the calibrated OCT R_S with half of the impedance value of the external reference resistors connected to the RUP and RDN pins. This feature is useful in applications that require both 25-Ω and 50-Ω calibrated OCT R_S at the same V_{CCIO}. For example, if your application requires 25-Ω and 50-Ω calibrated OCT R_S for SSTL-2 Class I and Class II I/O standards, you only need one OCT calibration block with 50-Ω external reference resistors.

You can enable the left-shift series termination control feature in the ALTIOBUF megafunction in the Quartus II software. The Quartus II software only allows left-shift series termination control for 25-Ω calibrated OCT R_S with 50-Ω external reference resistors connected to the RUP and RDN pins. You can only use left-shift series termination control for the I/O standards that support 25-Ω calibrated OCT R_S .

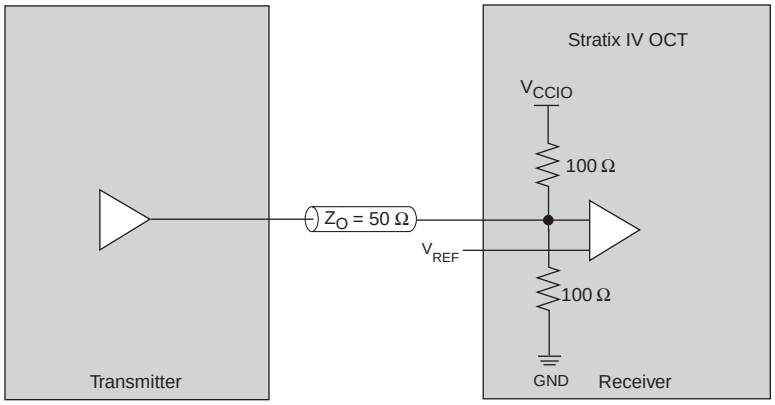
This feature is automatically enabled if you are using a bidirectional I/O with 25-Ω calibrated OCT R_S and 50-Ω parallel OCT.

For more information about how to enable the left-shift series termination feature in the ALTIOBUF megafunction, refer to the *ALTIOBUF Megafunction User Guide*.

On-Chip Parallel Termination with Calibration

Stratix IV devices support on-chip parallel termination with calibration in all banks. On-chip parallel termination with calibration is only supported for input configuration of input and bidirectional pins. Output pin configurations do not support on-chip parallel termination with calibration. Figure 6–20 shows on-chip parallel termination with calibration. When you use parallel OCT, the V_{CCIO} of the bank must match the I/O standard of the pin where the parallel OCT is enabled.

Figure 6–20. On-Chip Parallel Termination with Calibration



The on-chip parallel termination calibration circuit compares the total impedance of the I/O buffer to the external 50-Ω ±1% resistors connected to the RUP and RDN pins and dynamically enables or disables the transistors until they match. Calibration occurs at the end of device configuration. When the calibration circuit finds the correct impedance, it powers down and stops changing the characteristics of the drivers. Table 6–7 lists the I/O standards that support on-chip parallel termination with calibration.

Table 6–7. Selectable I/O Standards with On-Chip Parallel Termination with Calibration

I/O Standard	On-Chip Parallel Termination Setting (Column I/O) (Ω)	On-Chip Parallel Termination Setting (Row I/O) (Ω)
SSTL-2 Class I, II	50	50
SSTL-18 Class I, II	50	50
SSTL-15 Class I, II	50	50
HSTL-18 Class I, II	50	50
HSTL-15 Class I, II	50	50
HSTL-12 Class I, II	50	50
Differential SSTL-2 Class I, II	50	50
Differential SSTL-18 Class I, II	50	50
Differential SSTL-15 Class I, II	50	50
Differential HSTL-18 Class I, II	50	50
Differential HSTL-15 Class I, II	50	50
Differential HSTL-12 Class I, II	50	50

Expanded On-Chip Series Termination with Calibration

OCT calibration circuits always adjust OCT R_S to match the external resistors connected to the RUP and RDN pin; however, it is possible to achieve OCT R_S values other than the 25-Ω and 50-Ω resistors. Theoretically, if you need a different OCT R_S value, you can change the resistance connected to the RUP and RDN pins accordingly. Practically, the OCT R_S range that Stratix IV devices support is limited because of output buffer size and granularity limitations.

The Quartus II software only allows discrete OCT R_S calibration settings of 25, 40, 50, and 60 Ω. You can select the closest discrete value of OCT R_S with calibration settings in the Quartus II software to your system to achieve the closest timing. For example, if you are using 20-Ω OCT R_S with calibration in your system, you can select the **25-Ω OCT R_S with calibration** setting in the Quartus II software to achieve the closest timing.

Table 6–8 lists expanded OCT R_S with calibration supported in Stratix IV devices. Use expanded on-chip series termination with calibration of SSTL and HSTL for impedence matching to improve signal integrity but do not use it to meet the JEDEC standard.

Table 6–8. Selectable I/O Standards with Expanded On-Chip Series Termination with Calibration Range

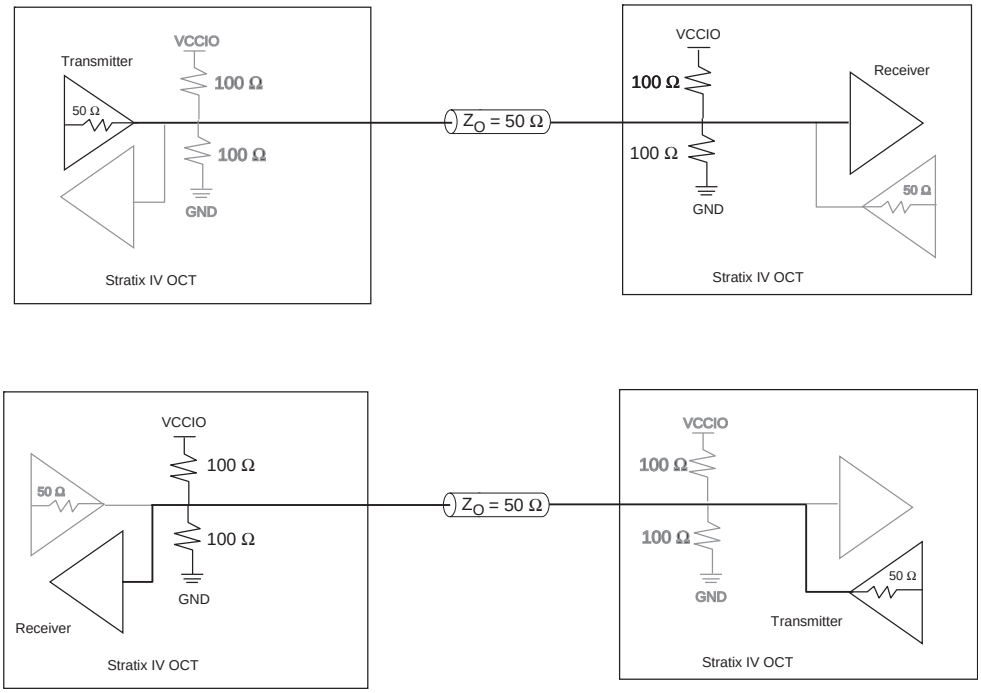
I/O Standard	Expanded OCT R_S Range	
	Row I/O (Ω)	Column I/O (Ω)
3.3-V LVTTL/LVCMOS	20–60	20–60
2.5-V LVTTL/LVCMOS	20–60	20–60
1.8-V LVTTL/LVCMOS	20–60	20–60
1.5-V LVTTL/LVCMOS	40–60	20–60
1.2-V LVTTL/LVCMOS	40–60	20–60
SSTL-2	20–60	20–60
SSTL-18	20–60	20–60
SSTL-15	40–60	20–60
HSTL-18	20–60	20–60
HSTL-15	40–60	20–60
HSTL-12	40–60	20–60

Dynamic On-Chip Termination

Stratix IV devices support on and off dynamic termination, both series and parallel, for a bidirectional I/O in all I/O banks. Figure 6–21 shows the termination schemes supported in Stratix IV devices. Dynamic parallel termination is enabled only when the bidirectional I/O acts as a receiver and is disabled when it acts as a driver. Similarly, dynamic series termination is enabled only when the bidirectional I/O acts as a driver and is disabled when it acts as a receiver. This feature is useful for terminating any high-performance bidirectional path because signal integrity is optimized depending on the direction of the data.

Using dynamic OCT helps save power because device termination is internal instead of external. Termination only switches on during input operation, thus drawing less static power.

Figure 6–21. Dynamic Parallel OCT in Stratix IV Devices

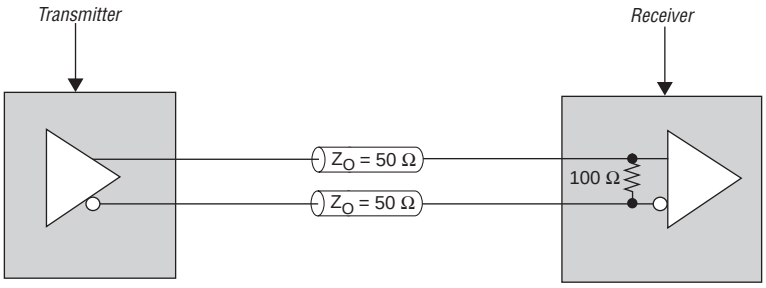


For more information about tolerance specifications for OCT with calibration, refer to the *DC and Switching Characteristics* chapter.

LVDS Input OCT (R_D)

Stratix IV devices support OCT for differential LVDS input buffers with a nominal resistance value of 100 Ω, as shown in Figure 6–22. Differential OCT R_D can be enabled in row I/O banks when both the V_{CCIO} and V_{CCPD} is set to 2.5 V. The column I/O banks do not support OCT R_D. The dedicated clock input pairs `CLK[1,3,8,10][p,n]`, `PLL_L[1,4]_CLK[p,n]`, and `PLL_R[1,4]_CLK[p,n]` on the row I/O banks of Stratix IV devices do not support R_D termination.

Figure 6–22. Differential Input OCT



For more information about differential on-chip termination, refer to the *High Speed Differential I/O Interfaces with DPA* chapter.

OCT Calibration

Stratix IV devices support calibrated on-chip series termination (R_S) and calibrated on-chip parallel termination (R_T) on all I/O pins. You can calibrate the device's I/O bank with any of the OCT calibration blocks available in the device.

OCT Calibration Block Location

Table 6–9 and Table 6–10 list the location of OCT calibration blocks in Stratix IV devices. Table 6–9 lists the OCT calibration blocks in Banks 1A through 4C. Table 6–10 lists the OCT calibration blocks in Banks 5A through 8C. For both tables, the following legend applies:

- "✓" indicates I/O banks with OCT calibration block
- "X" indicates I/O banks without OCT calibration block
- "—" indicates I/O banks that are not available in the device

Table 6–9 and Table 6–10 do not show transceiver banks and transceiver calibration blocks.

Table 6–9. OCT Calibration Block Counts and Placement in Stratix IV Devices (1A through 4C) (Part 1 of 2)

Device	Pin	Number of OCT Blocks	Bank											
			1A	1B	1C	2A	2B	2C	3A	3B	3C	4A	4B	4C
EP4SE230	780	8	✓	—	X	✓	—	X	✓	—	X	✓	—	X
EP4SE360	780	8	✓	—	X	✓	—	X	✓	—	X	✓	—	X
	1152	8	✓	—	X	✓	—	X	✓	X	X	✓	X	X
EP4SE530	1152	8	✓	—	X	✓	—	X	✓	X	X	✓	X	X
	1517	10	✓	X	X	✓	X	X	✓	X	✓	✓	X	X
	1760	10	✓	X	X	✓	X	X	✓	X	✓	✓	X	X
EP4SE820	1152	8	✓	—	X	✓	—	X	✓	X	X	✓	X	X
	1517	10	✓	X	X	✓	X	X	✓	X	✓	✓	X	X
	1760	10	✓	X	X	✓	X	X	✓	X	✓	✓	X	X
EP4SGX70	780	8	✓	—	X	✓	—	X	✓	—	X	✓	—	X
EP4SGX110	780	8	✓	—	X	✓	—	X	✓	—	X	✓	—	X
	1152	8	✓	—	X	—	—	—	✓	—	X	✓	—	X
EP4SGX180	780	8	✓	—	X	✓	—	X	✓	—	X	✓	—	X
	1152	8	✓	—	X	—	—	—	✓	X	X	✓	X	X
	1517	8	✓	—	X	✓	—	X	✓	X	X	✓	X	X
EP4SGX230	780	8	✓	—	X	✓	—	X	✓	—	X	✓	—	X
	1152	8	✓	—	X	—	—	—	✓	X	X	✓	X	X
	1517	8	✓	—	X	✓	—	X	✓	X	X	✓	X	X
EP4SGX290	780	8	—	—	—	—	—	—	✓	—	X	✓	—	X
	1152	8	✓	—	X	—	—	—	✓	X	X	✓	X	X
	1517	8	✓	—	X	✓	—	X	✓	X	X	✓	X	X
	1760	8	✓	—	X	✓	—	X	✓	X	X	✓	X	X
	1932	10	✓	X	X	✓	—	X	✓	X	✓	✓	X	X

Table 6–9. OCT Calibration Block Counts and Placement in Stratix IV Devices (1A through 4C) (Part 2 of 2)

Device	Pin	Number of OCT Blocks	Bank											
			1A	1B	1C	2A	2B	2C	3A	3B	3C	4A	4B	4C
EP4SGX360	780	8	—	—	—	—	—	—	✓	—	X	✓	—	X
	1152	8	✓	—	X	—	—	—	✓	X	X	✓	X	X
	1517	8	✓	—	X	✓	—	X	✓	X	X	✓	X	X
	1760	8	✓	—	X	✓	—	X	✓	X	X	✓	X	X
	1932	10	✓	X	X	✓	—	X	✓	X	✓	✓	X	X
EP4SGX530	1152	8	✓	—	X	—	—	—	✓	X	✓	✓	X	X
	1517	10	✓	—	X	✓	—	X	✓	X	✓	✓	X	X
	1760	10	✓	—	X	✓	—	X	✓	X	✓	✓	X	X
	1932	10	✓	—	X	✓	X	X	✓	X	✓	✓	X	X
EP4S40G2	1517	8	✓	—	X	✓	—	X	✓	X	X	✓	X	X
EP4S40G5	1517	10	✓	—	X	✓	—	X	✓	X	✓	✓	X	X
EP4S100G2	1517	8	✓	—	X	✓	—	X	✓	X	X	✓	X	X
EP4S100G3	1932	10	✓	—	X	✓	X	X	✓	X	✓	✓	X	X
EP4S100G4	1932	10	✓	—	X	✓	X	X	✓	X	✓	✓	X	X
EP4S100G5	1517	10	✓	—	X	✓	—	X	✓	X	✓	✓	X	X
	1932	10	✓	—	X	✓	X	X	✓	X	✓	✓	X	X

Table 6–10. OCT Calibration Block Counts and Placement in Stratix IV Devices (5A through 8C) (Part 1 of 2)

Device	Pin	Number of OCT Blocks	Bank											
			5A	5B	5C	6A	6B	6C	7A	7B	7C	8A	8B	8C
EP4SE230	780	8	✓	—	X	✓	—	X	✓	—	X	✓	—	X
EP4SE360	780	8	✓	—	X	✓	—	X	✓	—	X	✓	—	X
	1152	8	✓	—	X	✓	—	X	✓	X	X	✓	X	X
EP4SE530	1152	8	✓	—	X	✓	—	X	✓	X	X	✓	X	X
	1517	10	✓	X	X	✓	X	X	✓	X	X	✓	X	✓
	1760	10	✓	X	X	✓	X	X	✓	X	X	✓	X	✓
EP4SE820	1152	8	✓	—	X	✓	—	X	✓	X	X	✓	X	X
	1517	10	✓	X	X	✓	X	X	✓	X	X	✓	X	✓
	1760	10	✓	X	X	✓	X	X	✓	X	X	✓	X	✓
EP4SGX70	780	8	—	—	—	—	—	—	✓	—	X	✓	—	X
EP4SGX110	780	8	—	—	—	—	—	—	✓	—	X	✓	—	X
	1152	8	—	—	—	✓	—	X	✓	—	X	✓	—	X
EP4SGX180	780	8	—	—	—	—	—	—	✓	—	X	✓	—	X
	1152	8	—	—	—	✓	—	X	✓	X	X	✓	✓	X
	1517	8	✓	—	X	✓	—	X	✓	X	X	✓	X	X
EP4SGX230	780	8	—	—	—	—	—	—	✓	—	X	✓	—	X
	1152	8	—	—	—	✓	—	X	✓	X	X	✓	✓	X
	1517	8	✓	—	X	✓	—	X	✓	X	X	✓	X	X

Table 6–10. OCT Calibration Block Counts and Placement in Stratix IV Devices (5A through 8C) (Part 2 of 2)

Device	Pin	Number of OCT Blocks	Bank											
			5A	5B	5C	6A	6B	6C	7A	7B	7C	8A	8B	8C
EP4SGX290	780	8	—	—	—	—	—	—	✓	—	X	✓	—	X
	1152	8	—	—	—	✓	—	X	✓	X	X	✓	X	X
	1517	8	✓	—	X	✓	—	X	✓	X	X	✓	X	X
	1760	8	✓	—	X	✓	—	X	✓	X	X	✓	X	X
	1932	10	✓	—	X	✓	X	X	✓	X	X	✓	X	✓
EP4SGX360	780	8	—	—	—	—	—	—	✓	—	X	✓	—	X
	1152	8	—	—	—	✓	—	X	✓	X	X	✓	X	X
	1517	8	✓	—	X	✓	—	X	✓	X	X	✓	X	X
	1760	8	✓	—	X	✓	—	X	✓	X	X	✓	X	X
	1932	10	✓	—	X	✓	X	X	✓	X	X	✓	X	✓
EP4SGX530	1152	8	—	—	—	✓	—	X	✓	X	X	✓	X	✓
	1517	10	✓	—	X	✓	—	X	✓	X	X	✓	X	✓
	1760	10	✓	—	X	✓	—	X	✓	X	X	✓	X	✓
	1932	10	✓	X	X	✓	—	X	✓	X	X	✓	X	✓
EP4S40G2	1517	8	✓	—	X	✓	—	X	✓	X	X	✓	X	X
EP4S40G5	1517	10	✓	—	X	✓	—	X	✓	X	X	✓	X	✓
EP4S100G2	1517	8	✓	—	X	✓	—	X	✓	X	X	✓	X	X
EP4S100G3	1932	10	✓	X	X	✓	—	X	✓	X	X	✓	X	✓
EP4S100G4	1932	10	✓	X	X	✓	—	X	✓	X	X	✓	X	✓
EP4S100G5	1517	10	✓	—	X	✓	—	X	✓	X	X	✓	X	✓
	1932	10	✓	X	X	✓	—	X	✓	X	X	✓	X	✓

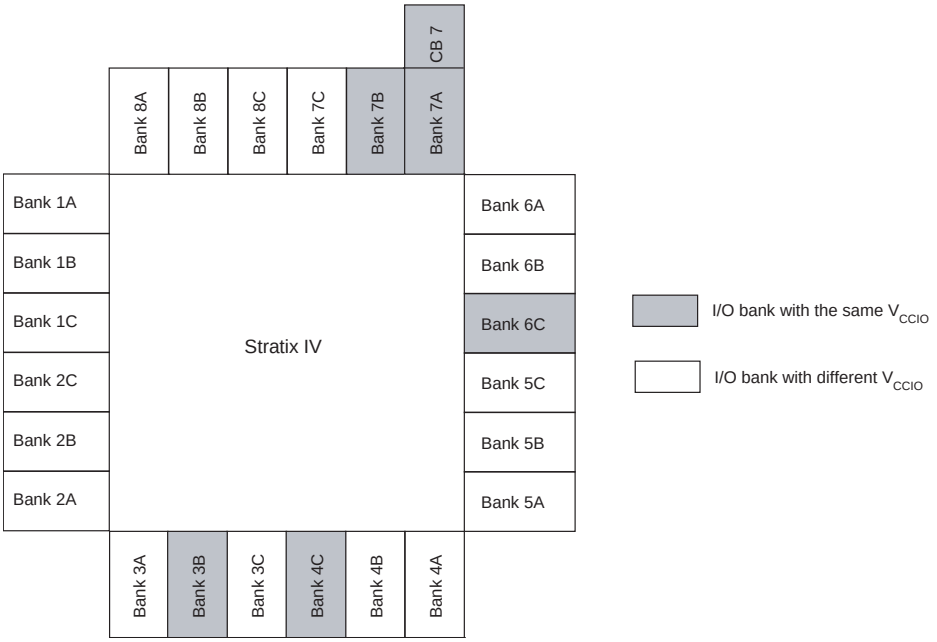
Sharing an OCT Calibration Block on Multiple I/O Banks

An OCT calibration block has the same V_{CCIO} as the I/O bank that contains the block. OCT R_S calibration is supported on all I/O banks with different V_{CCIO} voltage standards, up to the number of available OCT calibration blocks. You can configure the I/O banks to receive calibration codes from any OCT calibration block with the same V_{CCIO}. All I/O banks with the same V_{CCIO} can share one OCT calibration block, even if that particular I/O bank has an OCT calibration block.

For example, Figure 6–23 shows a group of I/O banks that has the same V_{CCIO} voltage. If a group of I/O banks has the same V_{CCIO} voltage, you can use one OCT calibration block to calibrate the group of I/O banks placed around the periphery. Because 3B, 4C, 6C, and 7B have the same V_{CCIO} as bank 7A, you can calibrate all four I/O banks (3B, 4C, 6C, and 7B) with the OCT calibration block located in bank 7A. You can enable this by serially shifting out OCT R_S calibration codes from the OCT calibration block located in bank 7A to the I/O banks located around the periphery.

Figure 6–23 is a top view of the silicon die that corresponds to a reverse view for flip chip packages. It is a graphical representation only. This figure does not show transceiver banks and transceiver calibration blocks.

Figure 6–23. Example of Calibrating Multiple I/O Banks with One Shared OCT Calibration Block



OCT Calibration Block Modes of Operation

Stratix IV devices support OCT R_S and OCT R_T on all I/O banks. The calibration can occur in either power-up or user mode.

Power-Up Mode

In power-up mode, OCT calibration is automatically performed at power up. Calibration codes are shifted to selected I/O buffers before transitioning to user mode.

User Mode

In user mode, the `OCTUSRCLK`, `ENAOCT`, `nCLRUSR`, and `ENASER[9..0]` signals are used to calibrate and serially transfer calibration codes from each OCT calibration block to any I/O. Table 6–11 lists the user-controlled calibration block signal names and their descriptions.

Table 6–11. OCT Calibration Block Ports for User Control

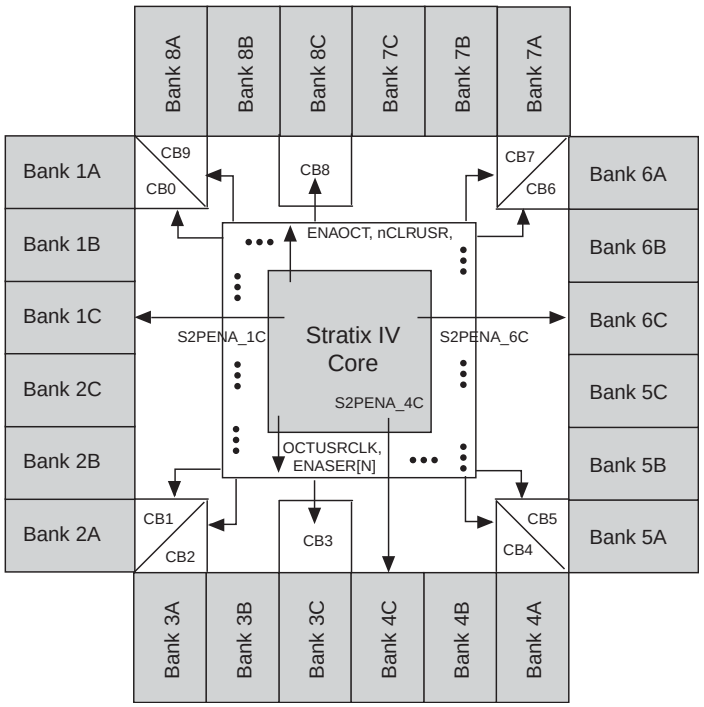
Signal Name	Description
OCTUSRCLK	Clock for OCT block.
ENAOCT	Enable OCT Termination (Generated by user IP).
ENASER[9..0]	When ENOCT = 0, each signal enables the OCT serializer for the corresponding OCT calibration block. When ENAOCT = 1, each signal enables OCT calibration for the corresponding OCT calibration block.
S2PENA_<*bank#*>	Serial-to-parallel load enable per I/O bank.
nCLRUSR	Clear user.

Figure 6–24 shows the flow of the user signal. When ENAOCT is 1, all OCT calibration blocks are in calibration mode; when ENAOCT is 0, all OCT calibration blocks are in serial data transfer mode. The OCTUSRCLK clock frequency must be 20 MHz or less.

You must generate all user signals on the rising edge of OCTUSRCLK.

Figure 6–24 does not show transceiver banks and transceiver calibration blocks.

Figure 6–24. Signals Used for User Mode Calibration



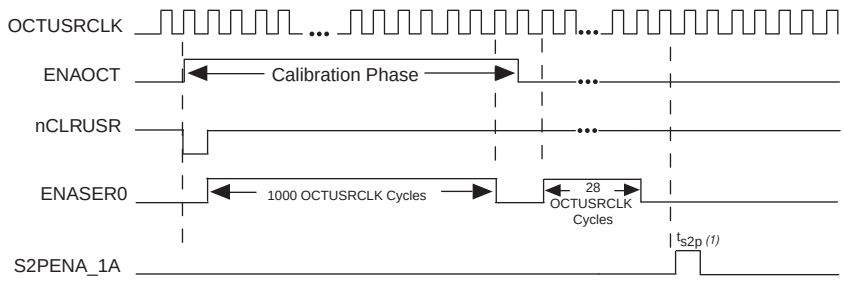
OCT Calibration

Figure 6–25 shows user mode signal-timing waveforms. To calibrate OCT block[N] (where N is a calibration block number), you must assert ENAOCT one cycle before asserting ENASER[N]. Also, nCLRUSR must be set to low for one OCTUSRCLK cycle before the ENASER[N] signal is asserted. Assert the ENASER[N] signals for 1000 OCTUSRCLK cycles to perform OCTRS and OCTRT calibration. You can de-assert ENAOCT one clock cycle after the last ENASER is de-asserted.

Serial Data Transfer

After you complete calibration, you must serially shift out the 28-bit OCT calibration codes (14-bit OCT RS code and 14-bit OCT RT) from each OCT calibration block to the corresponding I/O buffers. Only one OCT calibration block can send out the codes at any time by asserting only one ENASER[N] signal at a time. After you de-assert ENAOCT, wait at least one OCTUSRCLK cycle to enable any ENASER[N] signal to begin serial transfer. To shift the 28-bit code from the OCT calibration block[N], you must assert ENASER[N] for exactly 28 OCTUSRCLK cycles. Between two consecutive asserted ENASER signals, there must be at least one OCTUSRCLK cycle gap. (Figure 6–25).

Figure 6–25. OCT User Mode Signal—Timing Waveform for One OCT Block



Note to Figure 6–25:

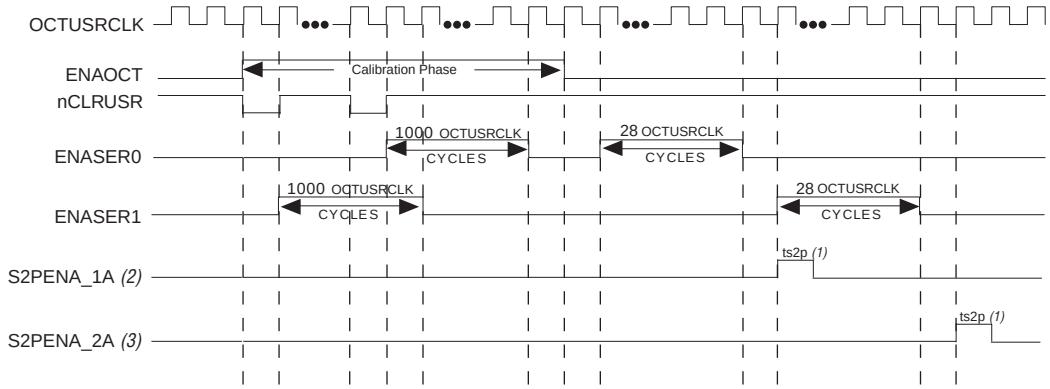
(1) $t_{s2p} \geq 25$ ns.

After calibrated codes are shifted in serially to each I/O bank, the calibrated codes must be converted from serial to parallel format before being used in the I/O buffers. Figure 6–25 shows the S2PENA signals that can be asserted at any time to update the calibration codes in each I/O bank. All I/O banks that received the codes from the same OCT calibration block can have S2PENA asserted at the same time, or at a different time, even while another OCT calibration block is calibrating and serially shifting codes. The S2PENA signal is asserted one OCTUSRCLK cycle after ENASER is de-asserted for at least 25 ns. You cannot use I/Os for transmitting or receiving data when their S2PENA is asserted for parallel codes transfer.

Example of Using Multiple OCT Calibration Blocks

Figure 6–26 shows a signal timing waveform for two OCT calibration blocks doing R_S and R_T calibration. Calibration blocks can start calibrating at different times by asserting the ENASER signals at different times. ENAOCT must remain asserted while any calibration is ongoing. You must set nCLRUSR low for one OCTUSRCLK cycle before each ENASER[N] signal is asserted. In Figure 6–26, when you set nCLRUSR to **0** for the second time to initialize OCT calibration block 0, this does not affect OCT calibration block 1, whose calibration is already in progress.

Figure 6–26. OCT User-Mode Signal Timing Waveform for Two OCT Blocks



Notes to Figure 6–26:

(1) `ts2p` ≥ 25 ns.

(2) `S2PENA_1A` is asserted in Bank 1A for calibration block 0.

(3) `S2PENA_2A` is asserted in Bank 2A for calibration block 1.

RS Calibration

If only R_S calibration is used for an OCT calibration block, its corresponding `ENASER` signal only needs to be asserted for 240 `OCTUSRCLK` cycles.

You must assert the `ENASER` signal for 28 `OCTUSRCLK` cycles for serial transfer.

Termination Schemes for I/O Standards

The following sections describe the different termination schemes for the I/O standards used in Stratix IV devices.

Single-Ended I/O Standards Termination

Voltage-referenced I/O standards require both an input reference voltage, V_{REF}, and a termination voltage, V_{TT}. The reference voltage of the receiving device tracks the termination voltage of the transmitting device.

Figure 6–27 and Figure 6–28 show the details of SSTL and HSTL I/O termination on Stratix IV devices.

In Stratix IV devices, you cannot use series and parallel OCT simultaneously. For more information, refer to "Dynamic On-Chip Termination" on page 6–29.

Figure 6–27. SSTL I/O Standard Termination

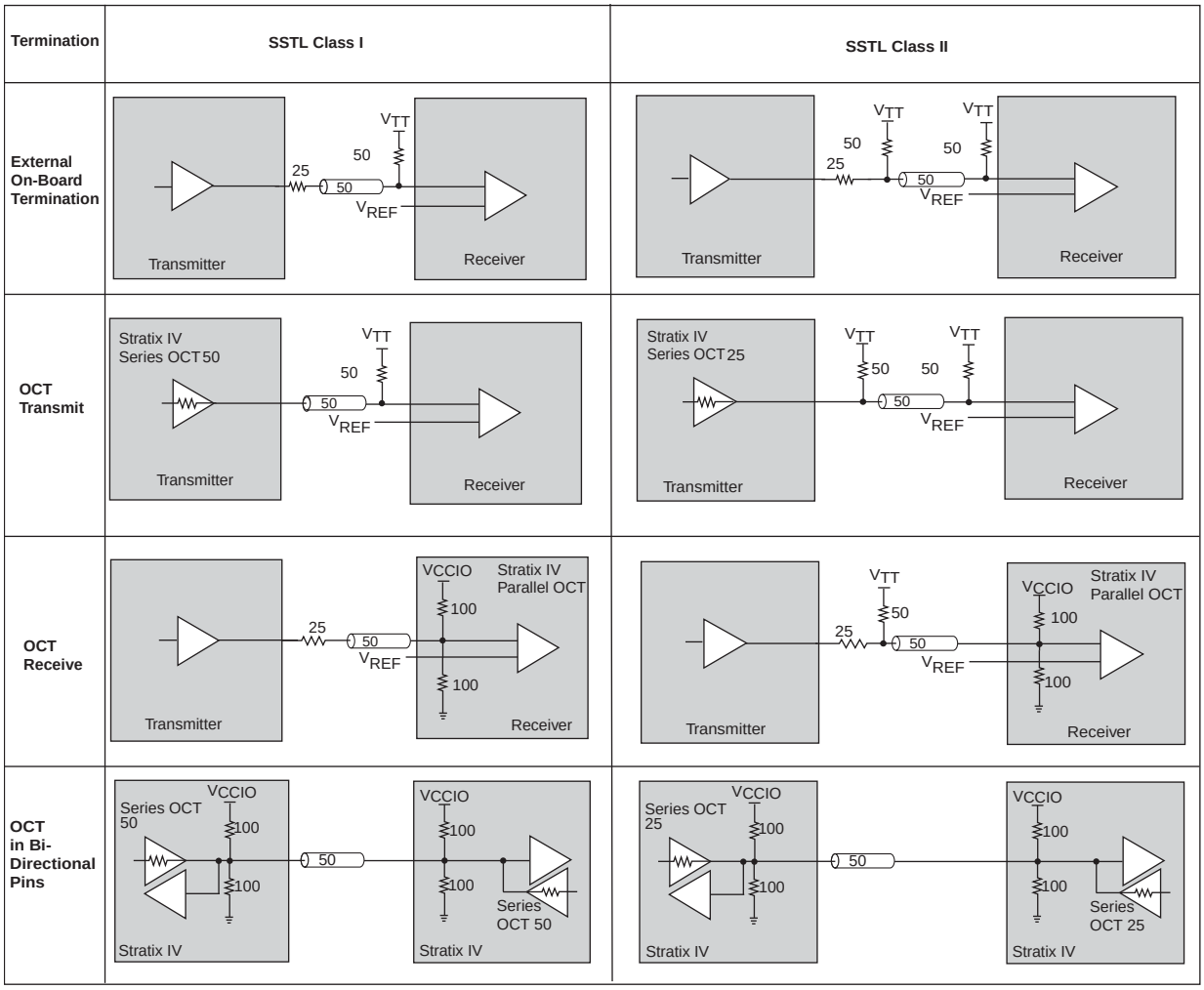
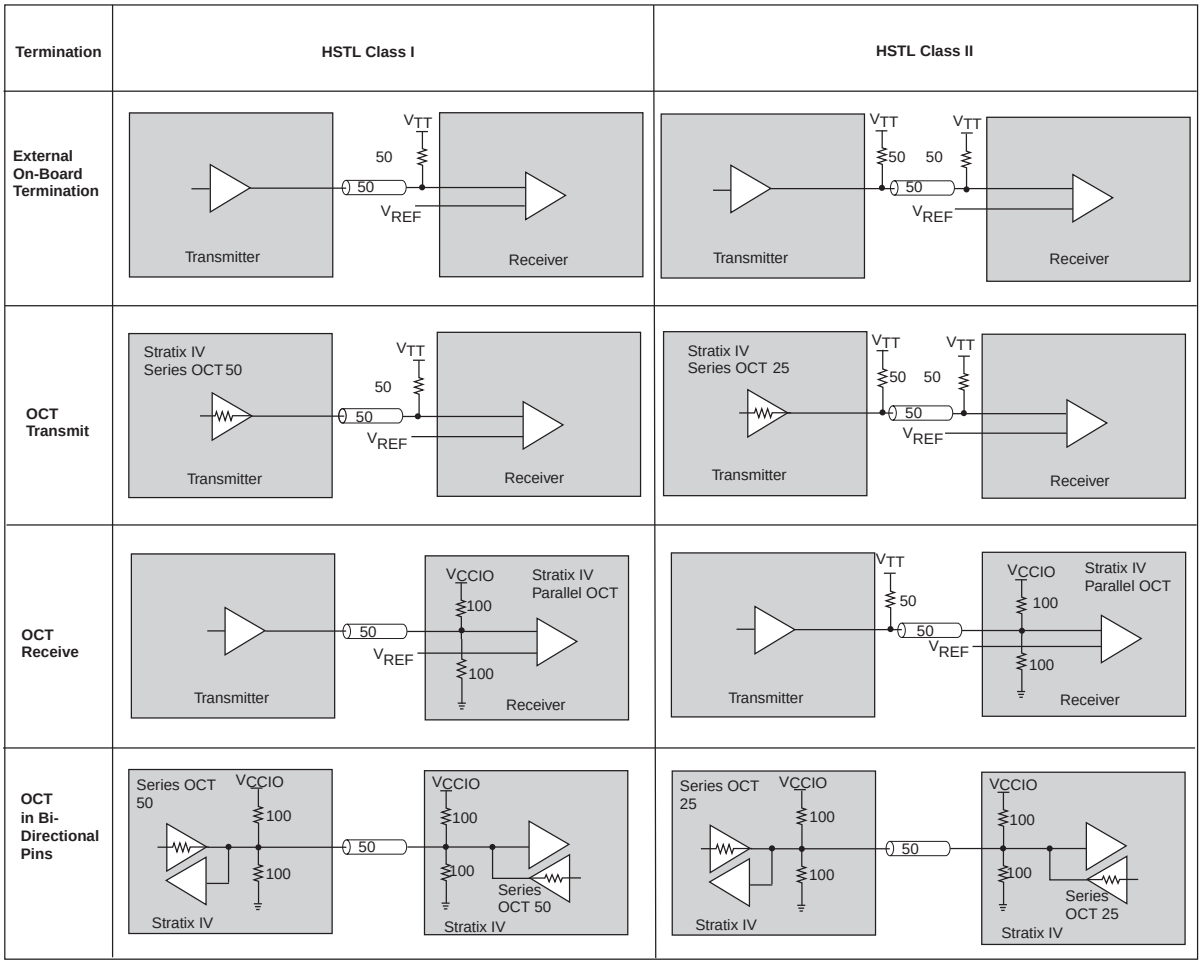


Figure 6–28. HSTL I/O Standard Termination



Differential I/O Standards Termination

Stratix IV devices support differential SSTL-18 and SSTL-2, differential HSTL-18, HSTL-15, HSTL-12, LVDS, LVPECL, RSDS, and mini-LVDS. Figure 6–29 through Figure 6–35 show the details of various differential I/O terminations on these devices.

☞ Differential HSTL and SSTL outputs are not true differential outputs. They use two single-ended outputs with the second output programmed as inverted.

Figure 6–29. Differential SSTL I/O Standard Termination

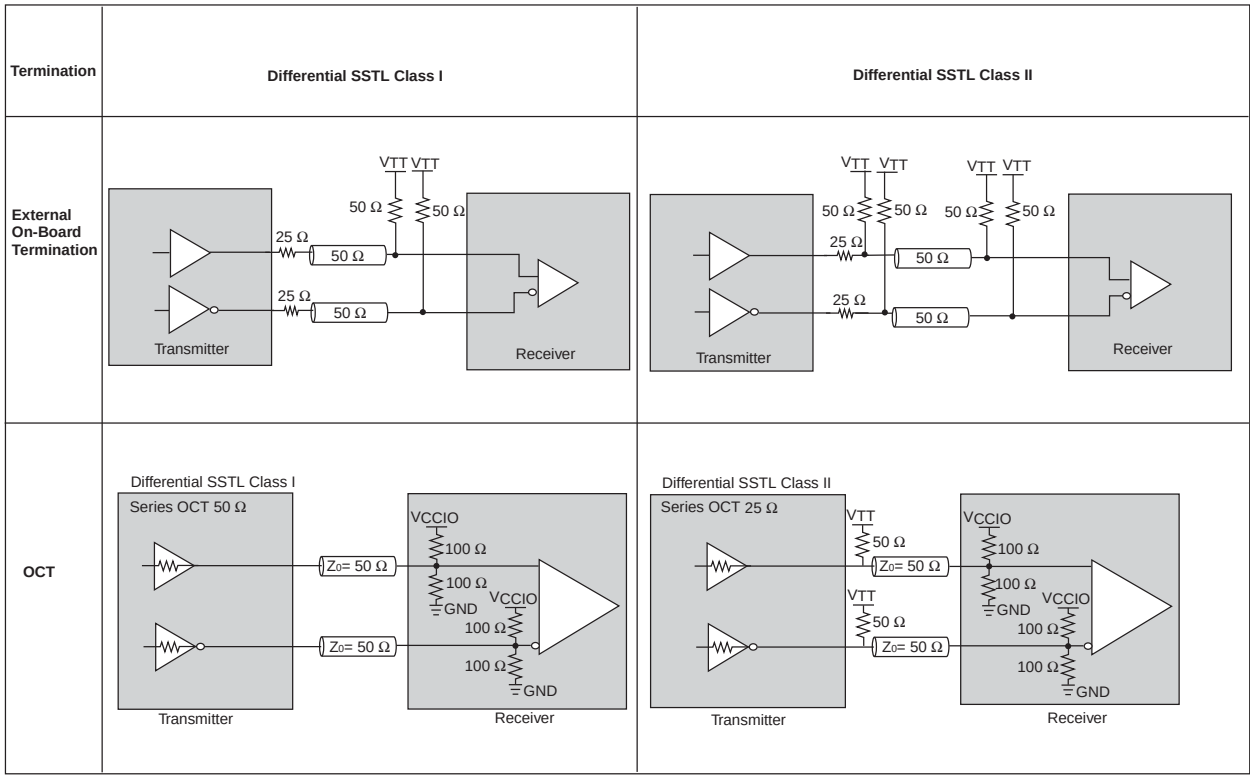
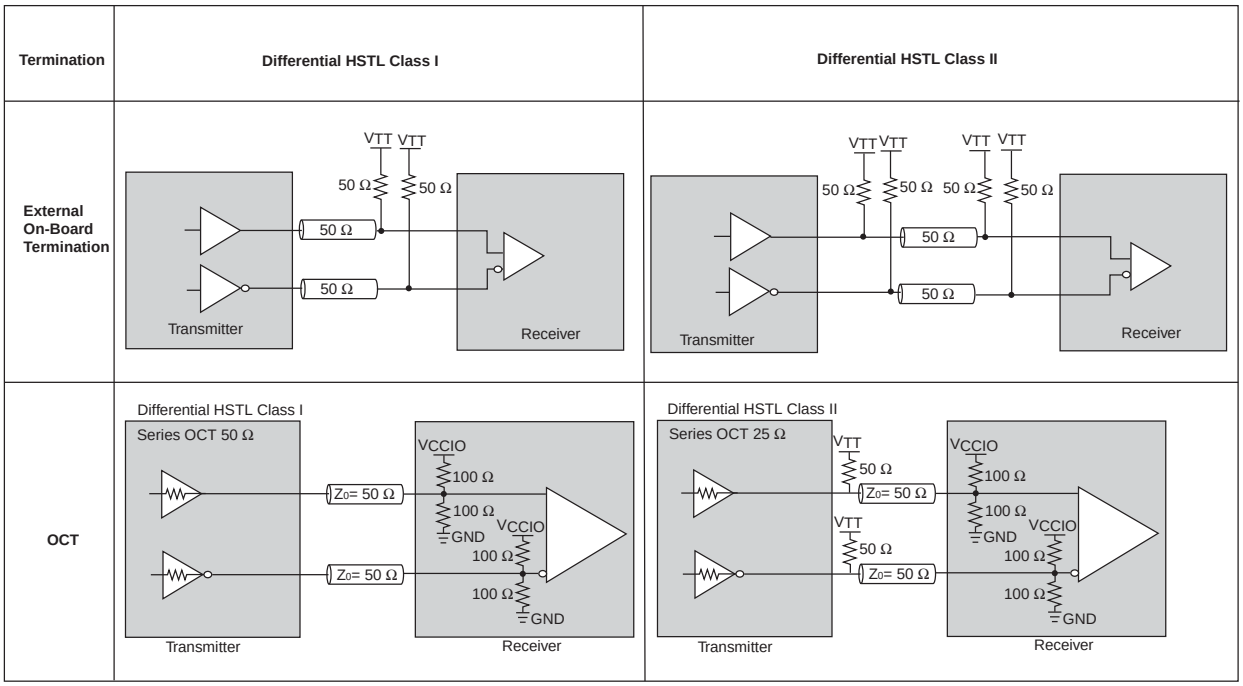


Figure 6–30. Differential HSTL I/O Standard Termination

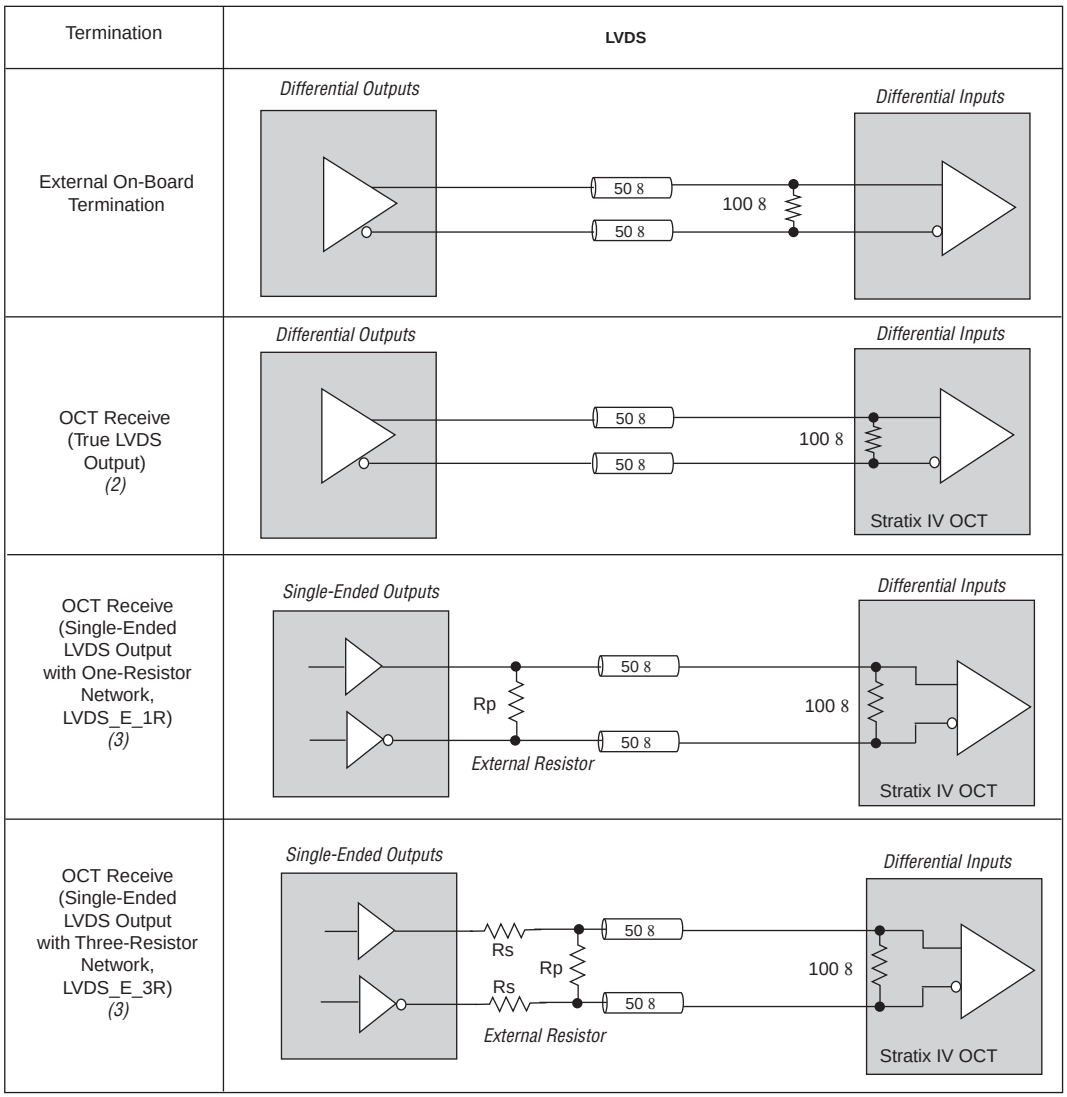


LVDS

The LVDS I/O standard is a differential high-speed, low-voltage swing, low-power, general-purpose I/O interface standard. In Stratix IV devices, the LVDS I/O standard requires a 2.5-V V_{CCIO} level. The LVDS input buffer requires 2.5-V V_{CCPD}. Use this standard in applications requiring high-bandwidth data transfer, such as backplane drivers and clock distribution. LVDS requires a 100-Ω termination resistor between the two signals at the input buffer. Stratix IV devices provide an optional 100-Ω differential termination resistor in the device using on-chip differential termination.

Figure 6–31 shows LVDS termination. The on-chip differential resistor is only available in the row I/O banks.

Figure 6–31. LVDS I/O Standard Termination *(Note 1)*



Notes to Figure 6–31:

(1) For LVDS output with a three-resistor network, the R_S and R_P values are 120 and 170 Ω respectively. For LVDS output with a one-resistor network, the R_P value is 120 Ω.

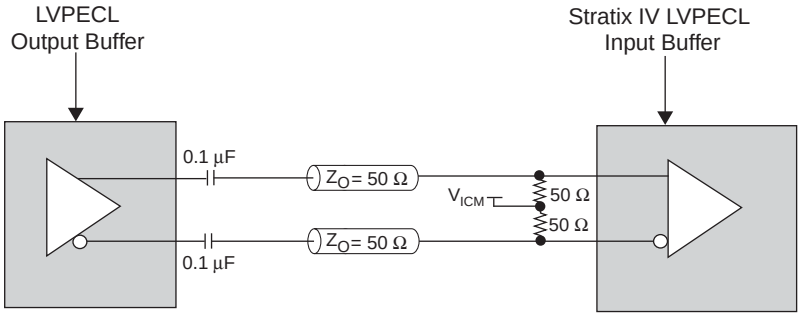
(2) Side I/O banks support true LVDS output buffers.

(3) Column and side I/O banks support LVDS_E_1R and LVDS_E_3R I/O standards using two single-ended output buffers.

Differential LVPECL

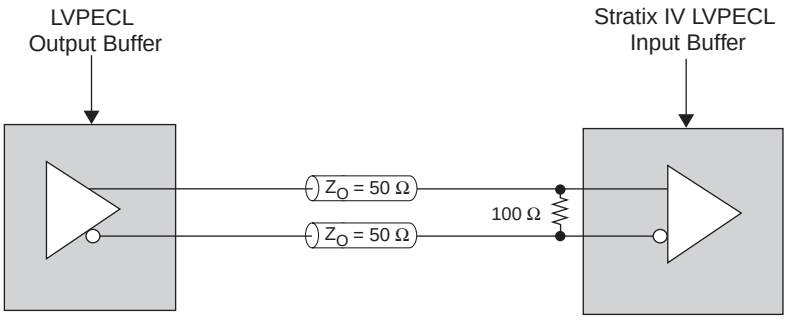
In Stratix IV devices, the LVPECL I/O standard is supported on input clock pins on column and row I/O banks. LVPECL output operation is not supported in Stratix IV devices. LVDS input buffers are used to support LVPECL input operation. AC coupling is required when the LVPECL common-mode voltage of the output buffer is higher than the LVPECL input common-mode voltage. Figure 6–32 shows the AC-coupled termination scheme. The 50-Ω resistors used at the receiver end are external to the device.

Figure 6–32. LVPECL AC-Coupled Termination



DC-coupled LVPECL is supported if the LVPECL output common mode voltage is within the Stratix IV LVPECL input buffer specification (Figure 6–33).

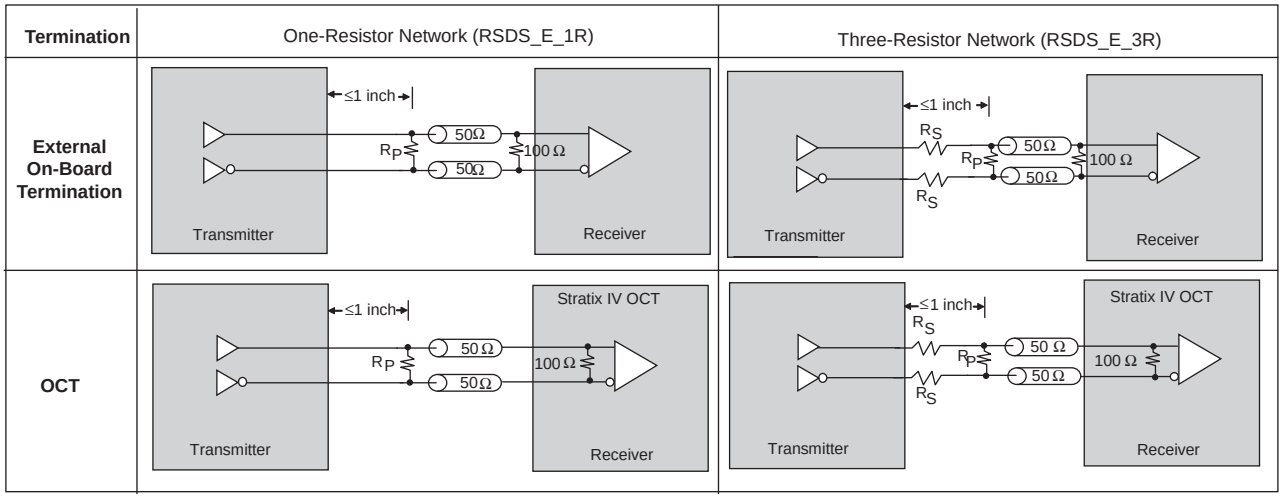
Figure 6–33. LVPECL DC-Coupled Termination



RSDS

Stratix IV devices support the RSDS output standard with data rates up to 230 Mbps using LVDS output buffer types. For transmitters, use two single-ended output buffers with the external one- or three-resistor networks in the column I/O bank, as shown in Figure 6–34. The one-resistor topology is for data rates up to 200 Mbps. The three-resistor topology is for data rates above 200 Mbps. The row I/O banks support RSDS output using true LVDS output buffers without an external resistor network.

Figure 6–34. RSDS I/O Standard Termination *(Note 1)*



Note to Figure 6–34:

(1) The R_S and R_P values are pending characterization.

A resistor network is required to attenuate the LVDS output-voltage swing to meet RSDS specifications. You can modify the three-resistor network values to reduce power or improve noise margin. The resistor values chosen must satisfy Equation 6–1.

Equation 6–1.

$$\frac{R_S \times \frac{R_p}{2}}{R_S + \frac{R_p}{2}} = 50\Omega$$

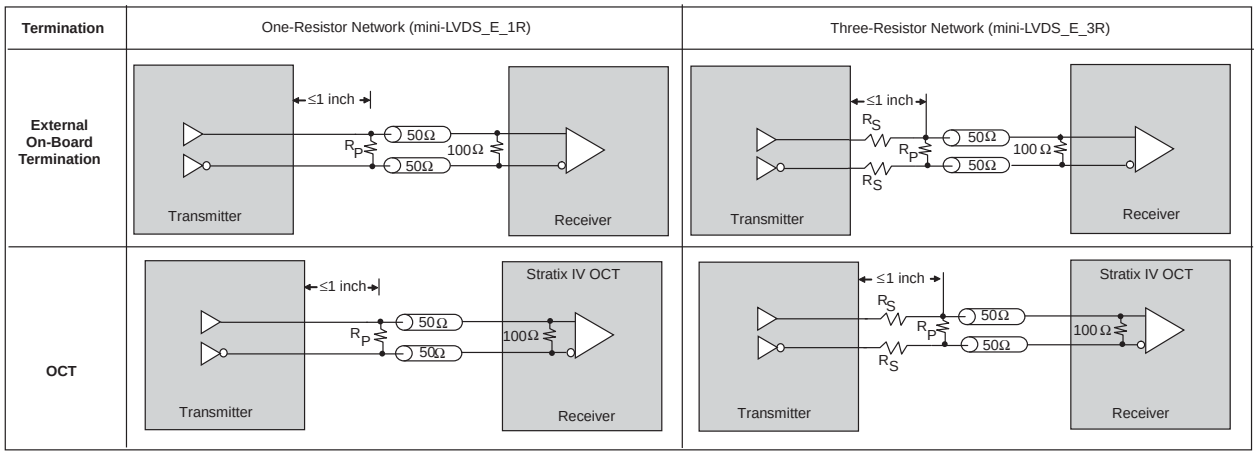
Altera recommends that you perform additional simulations using IBIS models to validate that custom resistor values meet the RSDS requirements.

For more information about the RSDS I/O standard, refer to the *RSDS Specification* from the National Semiconductor website at www.national.com.

Mini-LVDS

Stratix IV devices support the mini-LVDS output standard with data rates up to 340 Mbps using LVDS output buffer types. For transmitters, use two single-ended output buffers with external one- or three-resistor networks, as shown in Figure 6–35. The one-resistor topology is for data rates up to 200 Mbps. The three-resistor topology is for data rates above 200 Mbps. The row I/O banks support mini-LVDS output using true LVDS output buffers without an external resistor network.

Figure 6–35. Mini-LVDS I/O Standard Termination *(Note 1)*



Note to Figure 6–35:

(1) The R_S and R_P values are pending characterization.

A resistor network is required to attenuate the LVDS output voltage swing to meet the mini-LVDS specifications. You can modify the three-resistor network values to reduce power or improve noise margin. The resistor values chosen must satisfy Equation 6–1 on page 6–43.

Altera recommends that you perform additional simulations using IBIS models to validate that custom resistor values meet the RSDS requirements.

For more information about the mini-LVDS I/O standard, see the *mini-LVDS Specification* from the Texas Instruments website at www.ti.com.

Design Considerations

Although Stratix IV devices feature various I/O capabilities for high-performance and high-speed system designs, there are several other design considerations that require your attention to ensure the success of your designs.

I/O Bank Restrictions

Each I/O bank can simultaneously support multiple I/O standards. The following sections provide guidelines for mixing non-voltage-referenced and voltage-referenced I/O standards in Stratix IV devices.

Non-Voltage-Referenced Standards

Each I/O bank of a Stratix IV device has its own `VCCIO` pins and supports only one V_{CCIO}, either 1.2, 1.5, 1.8, 2.5, or 3.0 V. An I/O bank can simultaneously support any number of input signals with different I/O standard assignments if it meets the V_{CCIO} and V_{CCPD} requirement, as shown in Table 6–2 on page 6–3.

For output signals, a single I/O bank supports non-voltage-referenced output signals that are driving at the same voltage as V_{CCIO}. Because an I/O bank can only have one V_{CCIO} value, it can only drive out that one value for non-voltage-referenced signals. For example, an I/O bank with a 2.5-V V_{CCIO} setting can support 2.5-V standard inputs and outputs as well as 3.0-V LVCMOS inputs (but not output or bidirectional pins).

Voltage-Referenced Standards

To accommodate voltage-referenced I/O standards, each Stratix IV device's I/O bank supports multiple VREF pins feeding a common V_{REF} bus. The number of available VREF pins increases as device density increases. If these pins are not used as VREF pins, they cannot be used as generic I/O pins and must be tied to V_{CCIO} or GND. Each bank can only have a single V_{CCIO} voltage level and a single V_{REF} voltage level at a given time.

An I/O bank featuring single-ended or differential standards can support voltage-referenced standards if all voltage-referenced standards use the same V_{REF} setting.

For performance reasons, voltage-referenced input standards use their own V_{CCPD} level as the power source. This feature allows you to place voltage-referenced input signals in an I/O bank with a V_{CCIO} of 2.5 V or below. For example, you can place HSTL-15 input pins in an I/O bank with 2.5-V V_{CCIO}. However, the voltage-referenced input with parallel OCT enabled requires the V_{CCIO} of the I/O bank to match the voltage of the input standard.

Voltage-referenced bidirectional and output signals must be the same as the I/O bank's V_{CCIO} voltage. For example, you can only place SSTL-2 output pins in an I/O bank with a 2.5-V V_{CCIO}.

Mixing Voltage-Referenced and Non-Voltage-Referenced Standards

An I/O bank can support both voltage-referenced and non-voltage-referenced pins by applying each of the rule sets individually. For example, an I/O bank can support SSTL-18 inputs and 1.8-V inputs and outputs with a 1.8-V V_{CCIO} and a 0.9-V V_{REF}. Similarly, an I/O bank can support 1.5-V standards, 1.8-V inputs (but not outputs), and HSTL and HSTL-15 I/O standards with a 1.5-V V_{CCIO} and 0.75-V V_{REF}.

Document Revision History

Table 6–12 shows the revision history for this chapter.

Table 6–12. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
November 2009 v3.0	■ Updated Table 6–2, Table 6–4, Table 6–6, Table 6–9, and Table 6–10. ■ Updated Figure 6–1, Figure 6–2, Figure 6–4, Figure 6–5, Figure 6–6, Figure 6–8, Figure 6–9, Figure 6–10, Figure 6–11, Figure 6–12, Figure 6–13, and Figure 6–31. ■ Added Table 6–8. ■ Added Figure 6–7, Figure 6–14, Figure 6–15, and Figure 6–16. ■ Added "Left-Shift Series Termination Control" and "Expanded On-Chip Series Termination with Calibration" sections. ■ Updated "MultiVolt I/O Interface", "RSDS", "Mini-LVDS", and "Non-Voltage-Referenced Standards" sections. ■ Deleted Figure 6-5: Number of I/Os in Each Bank in EP4SE290 and EP4SE360 in the 1517-Pin FineLine BGA Package. ■ Minor text edits.	—
June 2009 v2.3	■ Added introductory sentences to improve search ability. ■ Removed the Conclusion section.	—
April 2009 v2.2	■ Updated Figure 6–2. ■ Updated Table 6–8 and Table 6–9. ■ Deleted Figure 6-14.	—
March 2009 v2.1	■ Updated Table 6–1, Table 6–2,Table 6–3, Table 6–4, Table 6–6, Table 6–8, and Table 6–9. ■ Updated Figure 6–2, Figure 6–7, Figure 6–8, Figure 6–9, Figure 6–10, Figure 6–11, and Figure 6–12. ■ Added Figure 6–14. ■ Removed Equation 6–2. ■ Removed "Referenced Documents" section.	—
November 2008 v2.0	■ Updated "Modular I/O Banks" on page 6–7. ■ Updated Figure 6–3. ■ Updated Figure 6–21. ■ Made minor editorial changes.	—
May 2008 v1.0	Initial release.	—

7. External Memory Interfaces in Stratix IV Devices

SIV51007-3.0

This chapter describes external memory interfaces available with the Stratix® IV device family and that family's silicon capability to support external memory interfaces. To support the level of system bandwidth achievable with Altera® Stratix IV FPGAs, the devices provide an efficient architecture to quickly and easily fit wide external memory interfaces within their small modular I/O bank structure. The I/Os are designed to provide high-performance support for existing and emerging external double data rate (DDR) memory standards, such as DDR3, DDR2, DDR SDRAM, QDRII+, QDRII SRAM, and RLDRAM II.

Stratix IV I/O elements provide easy-to-use built-in functionality required for a rapid and robust implementation with features such as dynamic calibrated on-chip termination (OCT), trace mismatch compensation, read- and write-leveling circuit for DDR3 SDRAM interfaces, half data rate (HDR) blocks, and 4- to 36-bit programmable DQ group widths.

The high-performance memory interface solution is backed-up by a self-calibrating megafunction (ALTMEMPHY), optimized to take advantage of the Stratix IV I/O structure and the TimeQuest Timing Analyzer, which completes the picture by providing the total solution for the highest reliable frequency of operation across process, voltage, and temperature (PVT) variations.

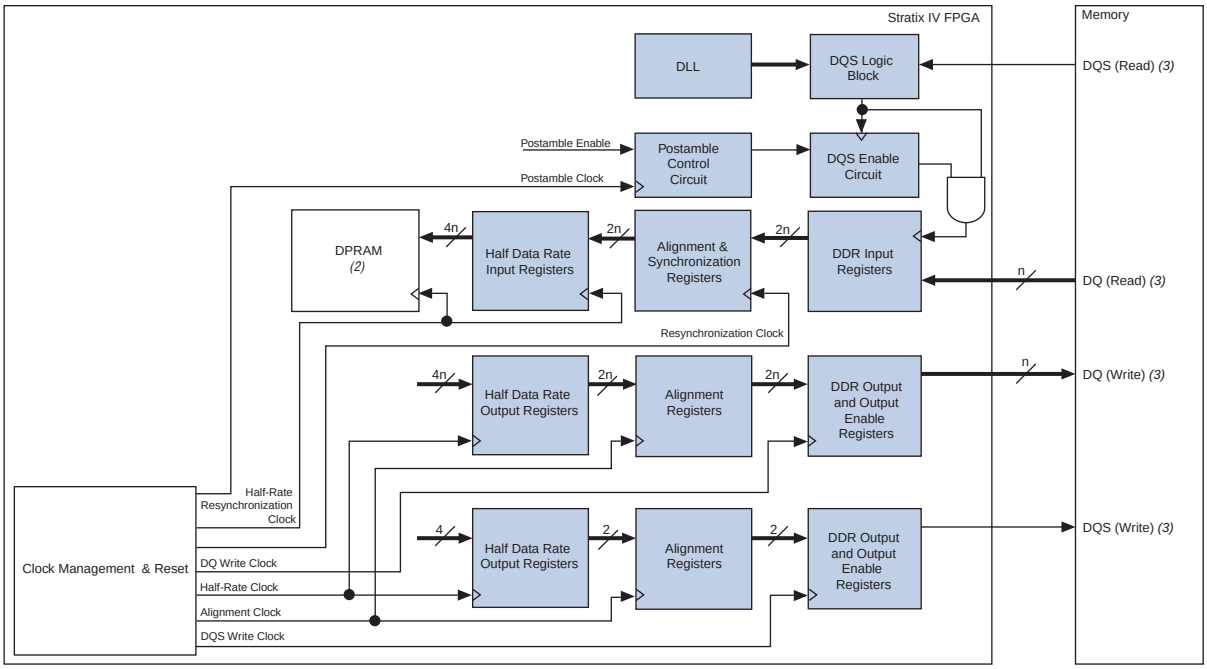
This chapter contains the following sections:

- "Memory Interfaces Pin Support" on page 7–3
- "Stratix IV External Memory Interface Features" on page 7–26

For more information about external memory system performance specifications, board design guidelines, timing analysis, simulation, and debugging information, refer to the *External Memory Interface Handbook*.

Figure 7–1 shows an overview of the memory interface data path that uses all the Stratix IV I/O element (IOE) features.

Figure 7–1. External Memory Interface Data Path Overview *(Note 1)*, *(2)*



Notes to Figure 7–1:

(1) You can bypass each register block.

(2) The blocks used for each memory interface may differ slightly. The shaded blocks are part of the Stratix IV IOE.

(3) These signals may be bidirectional or unidirectional, depending on the memory standard. When bidirectional, the signal is active during both read and write operations.

Memory interfaces use Stratix IV device features such as delay-locked loops (DLLs), dynamic OCT control, read- and write-leveling circuitry, and I/O features such as OCT, programmable input delay chains, programmable output delay, slew rate adjustment, and programmable drive strength.

For more information about I/O features, refer to the *I/O Features in Stratix IV Devices* chapter.

The ALTMEMPHY megafunction instantiates a phase-locked loop (PLL) and PLL reconfiguration logic to adjust the phase shift based on VT variation.

For more information about the Stratix IV PLL, refer to the *Clock Networks and PLLs in Stratix IV Devices* chapter. For more information about the ALTMEMPHY megafunction, refer to the *External Memory PHY Interface Megafunction User Guide (ALTMEMPHY)*.

Memory Interfaces Pin Support

A typical memory interface requires data (D, Q, or DQ), data strobe (DQS/CQ and DQSn/CQn), address, command, and clock pins. Some memory interfaces use data mask (DM, BWSn, or NWSn) pins to enable write masking and QVLD pins to indicate that the read data is ready to be captured. This section describes how Stratix IV devices support all these different pins.

For more information about pin connections, refer to the *Stratix IV Device Family Pin Connection Guidelines*.

Table 7–1 lists the pin connections between a Stratix IV device and an external memory device.

Table 7–1. Stratix IV Memory Interface Pin Utilization (Part 1 of 2)

Pin Description	Memory Standard	Stratix IV Pin Utilization
Read Data	All	DQ
Write Data	All	DQ *(1)*
Parity, DM, BWSn, NWSn, QVLD, ECC	All	DQ *(1)*, *(2)*
Read Data Strobes/Clocks	■ DDR3 SDRAM ■ DDR2 SDRAM (with differential DQS signaling) *(3)* ■ RLDRAM II	Differential DQS/DQSn (also used as a write data clock)
	■ DDR2 SDRAM (with single-ended DQS signaling) *(3)* ■ DDR SDRAM	Single-ended DQS (also used as a write data clock)
	■ QDRII+ SRAM ■ QDRII SRAM	Complementary CQ/CQn
Write Data Clocks	■ QDRII+ SRAM *(4)* ■ QDRII SRAM *(4)* ■ RLDRAM II Separate I/O (SIO)	Any DQS and DQSn pin pairs associated with the DQ groups used for the write data pins *(1)*
	■ RLDRAM II Common I/O (CIO) *(6)*	Any DQ pin with `DIFFOUT` capability within the same DQS/DQ group or adjacent group as the read data (Q) pins, or in the same bank as the address and command pins

Table 7–1. Stratix IV Memory Interface Pin Utilization (Part 2 of 2)

Pin Description	Memory Standard	Stratix IV Pin Utilization
Memory Clocks (for Address and Command) *(5)*	■ DDR3 SDRAM with leveling	Any unused DQ or DQS pins with `DIFFIO_RX` capability for the `mem_clk[0]` and `mem_clk_n[0]` signals
		Any unused DQ or DQS pins with `DIFFOUT` capability for the `mem_clk[n:1]` and `mem_clk_n[n:1]` signals (where n is greater than or equal to 1)
	■ DDR3 SDRAM without leveling ■ DDR2 SDRAM (with differential DQS signaling) *(3)*	Any unused pins with `DIFFIO_RX` capability for the `mem_clk[0]` and `mem_clk_n[0]` signals
		Any unused pins with `DIFFOUT` capability for the `mem_clk[n:1]` and `mem_clk_n[n:1]` signals (where n is greater than or equal to 1)
	■ DDR2 SDRAM (with single-ended DQS signaling) *(3)* ■ DDR SDRAM ■ RLDRAM II	Any `DIFFOUT` pins

Notes to Table 7–1:

(1) If the write data signals are unidirectional, connect them, including the data mask pins, to a separate DQS/DQ group other than the read DQS/DQ group. Connect the write clock to the DQS and DQSn pin-pair associated with that DQS/DQ group. Do not use the CQ and CQn pin-pair as write clocks.

(2) The BWSn, NWSn, and DM pins must be part of the write DQS/DQ group, while parity, QVLD, and ECC pins must be part of the read DQS/DQ group. The ALTMEMPHY megafunction does not support the QVLD pin. However, if your design supports the QVLD pin, the QVLD pin must be part of the read DQS/DQ group as well.

(3) DDR2 SDRAM supports either single-ended or differential DQS signaling.

(4) QDRII+/QDRII SRAM devices use the K/K# clock pin-pair to latch write data, address, and command signals. The clocks must be part of the DQS/DQ group and follow the write data clock rules.

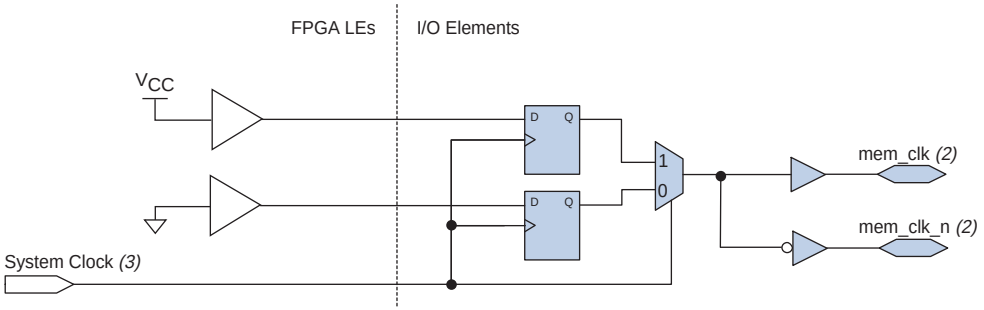
(5) ALTMEMPHY megafunction implementation for a DDR3, DDR2, or DDR SDRAM interface requires that you place all memory clock pin-pairs in a single DQ group of adequate width to minimize skew. For example, DIMMs requiring three memory clock pin-pairs must use a ×4 DQS/DQ group.

(6) When interfacing with RLDRAM II × 36 CIO devices, use two DQ pins in the ×16/×18 DQS/DQ groups, which are the DQS/DQSn pins in the × 4 or × 8/× 9 DQS/DQ groups for the write data clock.

DDR3, DDR2, DDR SDRAM, and RLDRAM II devices use the CK and CK# signals to capture the address and command signals. Generate these signals to mimic the write-data strobe using Stratix IV DDR I/O registers (DDIOs) to ensure that the timing relationships between the CK/CK# and DQS signals (t_{DQSS}, t_{DSS}, and t_{DSH} in DDR3, DDR2, and DDR SDRAM devices or t_{CKDK} in RLDRAM II devices) are met. QDRII+ and QDRII SRAM devices use the same clock (K/K#) to capture write data, address, and command signals.

Memory clock pins in Stratix IV devices are generated using a DDIO register going to differential output pins (refer to Figure 7–2), marked in the pin table with `DIFFOUT`, `DIFFIO_TX`, or `DIFFIO_RX` prefixes. For more information about which pins to use for memory clock pins, refer to Table 7–1.

Figure 7–2. Memory Clock Generation *(Note 1)*



Notes to Figure 7–2:

(1) For pin location requirements, refer to Table 7–1 on page 7–3.

(2) The `mem_clk[0]` and `mem_clk_n[0]` pins for DDR3, DDR2, and DDR SDRAM interfaces use the I/O input buffer for feedback required by the ALTMEMPHY megafunction for tracking; therefore, use bidirectional I/O buffers for these pins. For memory interfaces using a differential DQS input, the input feedback buffer is configured as differential input. For memory interfaces using a single-ended DQS input, the input buffer is configured as a single-ended input. Using a single-ended input feedback buffer requires that I/O standard's VREF voltage is provided to that I/O bank's VREF pins.

(3) To minimize jitter, regional clock networks are required for memory output clock generation.

Stratix IV devices offer differential input buffers for differential read-data strobe and clock operations. In addition, Stratix IV devices also provide an independent DQS logic block for each CQn pin for complementary read-data strobe and clock operations. In the Stratix IV pin tables, the differential DQS pin pairs are denoted as DQS and DQSn pins, while the complementary CQ signals are denoted as CQ and CQn pins. DQSn and CQn pins are marked separately in the pin table. Each CQn pin connects to a DQS logic block and the shifted CQn signals go to the negative-edge input registers in the DQ IOE registers.

☞ Use differential DQS signaling for DDR2 SDRAM interfaces running at or above 333 MHz.

DQ pins can be bidirectional signals, as in DDR3, DDR2, and DDR SDRAM, and RLDRAM II common I/O (CIO) interfaces, or unidirectional signals, as in QDRII+, QDRII SRAM, and RLDRAM II separate I/O (SIO) devices. Connect the unidirectional read-data signals to Stratix IV DQ pins and the unidirectional write-data signals to a different DQS/DQ group than the read DQS/DQ group. Furthermore, the write clocks must be assigned to the DQS/DQSn pins associated to this write DQS/DQ group. Do not use the CQ/CQn pin-pair for write clocks.

☞ Using a DQS/DQ group for the write-data signals minimizes output skew, allows access to the write-leveling circuitry (for DDR3 SDRAM interfaces), and allows vertical migration. These pins also have access to deskewing circuitry (using programmable delay chains) that can compensate for delay mismatch between signals on the bus.

The DQS and DQ pin locations are fixed in the pin table. Memory interface circuitry is available in every Stratix IV I/O bank that does not support transceivers. All the memory interface pins support the I/O standards required to support DDR3, DDR2, DDR SDRAM, QDRII+, QDRII SRAM, and RLDRAM II devices.

The Stratix IV device family supports DQS and DQ signals with DQ bus modes of ×4, ×8/×9, ×16/×18, or ×32/×36, although not all devices support DQS bus mode ×32/×36. When any of these pins are not used for memory interfacing, you can use them as user I/Os. In addition, you can use any DQSn or CQn pins not used for clocking as DQ (data) pins. Table 7–2 lists pin support per DQS/DQ bus mode, including the DQS/CQ and DQSn/CQn pin pair.

Table 7–2. Stratix IV DQS/DQ Bus Mode Pins

Mode	DQSn Support	CQn Support	Parity or DM (Optional)	QVLD (Optional) *(1)*	Typical Number of Data Pins per Group	Maximum Number of Data Pins per Group *(2)*
×4	Yes	No	No *(6)*	No	4	5
×8/×9 *(3)*	Yes	Yes	Yes	Yes	8 or 9	11
×16/×18 *(4)*	Yes	Yes	Yes	Yes	16 or 18	23
×32/×36 *(5)*	Yes	Yes	Yes	Yes	32 or 36	47

Notes to Table 7–2:

(1) The QVLD pin is not used in the ALTMEMPHY megafunction.

(2) This represents the maximum number of DQ pins (including parity, data mask, and QVLD pins) connected to the DQS bus network with single-ended DQS signaling. When you use differential or complementary DQS signaling, the maximum number of data per group decreases by one. This number may vary per DQS/DQ group in a particular device. Check the pin table for the accurate number per group. For DDR3, DDR2, and DDR interfaces, the number of pins is further reduced for an interface larger than ×8 due to the need of one DQS pin for each ×8/×9 group that is used to form the x16/×18 and ×32/×36 groups.

(3) Two ×4 DQS/DQ groups are stitched to make a ×8/×9 group so there are a total of 12 pins in this group.

(4) Four ×4 DQS/DQ groups are stitched to make a ×16/×18 group.

(5) Eight ×4 DQS/DQ groups are stitched to make a ×32/×36 group.

(6) The DM pin can be supported if differential DQS is not used and the group does not have additional signals.

Table 7–3 lists the number of DQS/DQ groups available per side in each Stratix IV device. For a more detailed listing of the number of DQS/DQ groups available per bank in each Stratix IV device, see Figure 7–3 through Figure 7–17. These figures represent the die-top view of the Stratix IV device.

Table 7–3. Number of DQS/DQ Groups in Stratix IV Devices per Side (Part 1 of 2) *(Note 1)*

Device	Package	Side	×4 *(2)*	×8/×9	×16/×18	×32/×36 *(3)*
EP4SGX70 EP4SGX110 EP4SGX180 EP4SGX230	780-pin FineLine BGA	Left	14	6	2	0
		Top / Bottom	17	8	2	0
		Right	0	0	0	0
EP4SGX290 EP4SGX360	780-pin FineLine BGA	Left / Right	0	0	0	0
		Top / Bottom	18	8	2	0
EP4SE230 EP4SE360	780-pin FineLine BGA	Left / Right	14	6	2	0
		Top / Bottom	17	8	2	0
EP4SGX110	1152-pin FineLine BGA (with 16 transceivers)	Right / Left	7	3	1	0
		Top / Bottom	17	8	2	0

Table 7–3. Number of DQS/DQ Groups in Stratix IV Devices per Side (Part 2 of 2) *(Note 1)*

Device	Package	Side	×4 *(2)*	×8/×9	×16/×18	×32/×36 *(3)*
EP4SGX70 EP4SGX110	1152-pin FineLine BGA (with 24 transceivers)	Right / Left	14	6	2	0
		Top / Bottom	17	8	2	0
EP4SGX180 EP4SGX230 EP4SGX290 EP4SGX360 EP4SGX530	1152-pin FineLine BGA	Right / Left	13	6	2	0
		Top / Bottom	26	12	4	0
EP4SE360 EP4SE530 EP4SE820	1152-pin FineLine BGA	All sides	26	12	4	0
EP4SGX180 EP4SGX230 EP4SGX290 EP4SGX360 EP4SGX530	1517-pin FineLine BGA	All sides	26	12	4	0
EP4SE530 EP4SE820	1517-pin FineLine BGA	Right / Left	34	16	6	0
		Top / Bottom	38	18	8	4
EP4S40G2 EP4S40G5 EP4S100G2 EP4S100G5	1517-pin FineLine BGA	Left	12	3	1	0
		Top / Bottom	26	12	4	0
		Right	11	4	1	0
EP4SGX290 EP4SGX360 EP4SGX530	1760-pin FineLine BGA	Right / Left	26	12	4	0
		Top / Bottom	38	18	8	4
EP4SE530	1760-pin FineLine BGA	Right / Left	34	16	6	0
		Top / Bottom	38	18	8	4
EP4SE820	1760-pin FineLine BGA	Right / Left	40	18	6	0
		Top / Bottom	44	22	10	4
EP4SGX290 EP4SGX360 EP4SGX530	1932-pin FineLine BGA	Right / Left	29	13	4	0
		Top / Bottom	38	18	8	4
EP4S100G3 EP4S100G4 EP4S100G5	1932-pin FineLine BGA	Left	8	2	0	0
		Top / Bottom	38	18	8	4
		Right	7	1	0	0

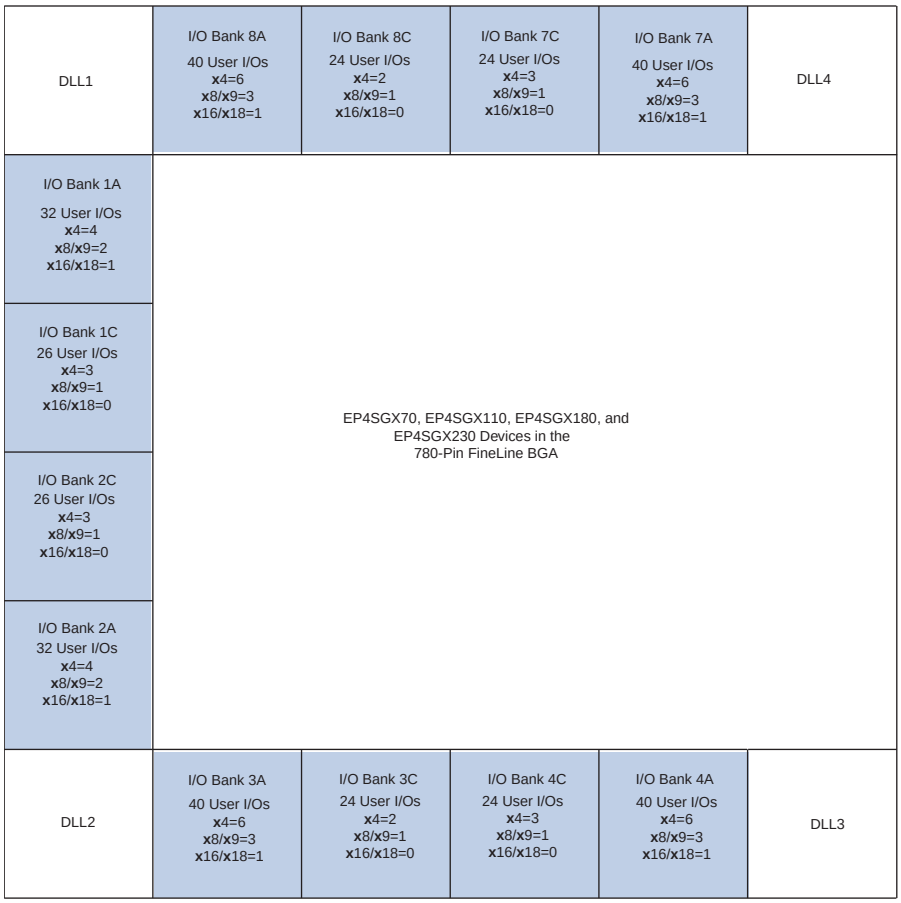
Notes to Table 7–3:

(1) These numbers are preliminary until the devices are available.

(2) Some of the ×4 groups may use R_{UP} and R_{DN} pins. You cannot use these groups if you use the Stratix IV calibrated OCT feature.

(3) To interface with a ×36 QDRII+/QDRII SRAM device in a Stratix IV FPGA that does not support the ×32/×36 DQS/DQ group, refer to "Combining ×16/×18 DQS/DQ Groups for a ×36 QDRII+/QDRII SRAM Interface" on page 7–24.

Figure 7–3. Number of DQS/DQ Groups per Bank in EP4SGX70, EP4SGX110, EP4SGX180, and EP4SGX230 Devices in the 780-Pin FineLine BGA Package *(Note 1)*, *(2)*, *(3)*, *(4)*. *(5)*



Notes to Figure 7–3:

(1) These numbers are preliminary until the devices are available.

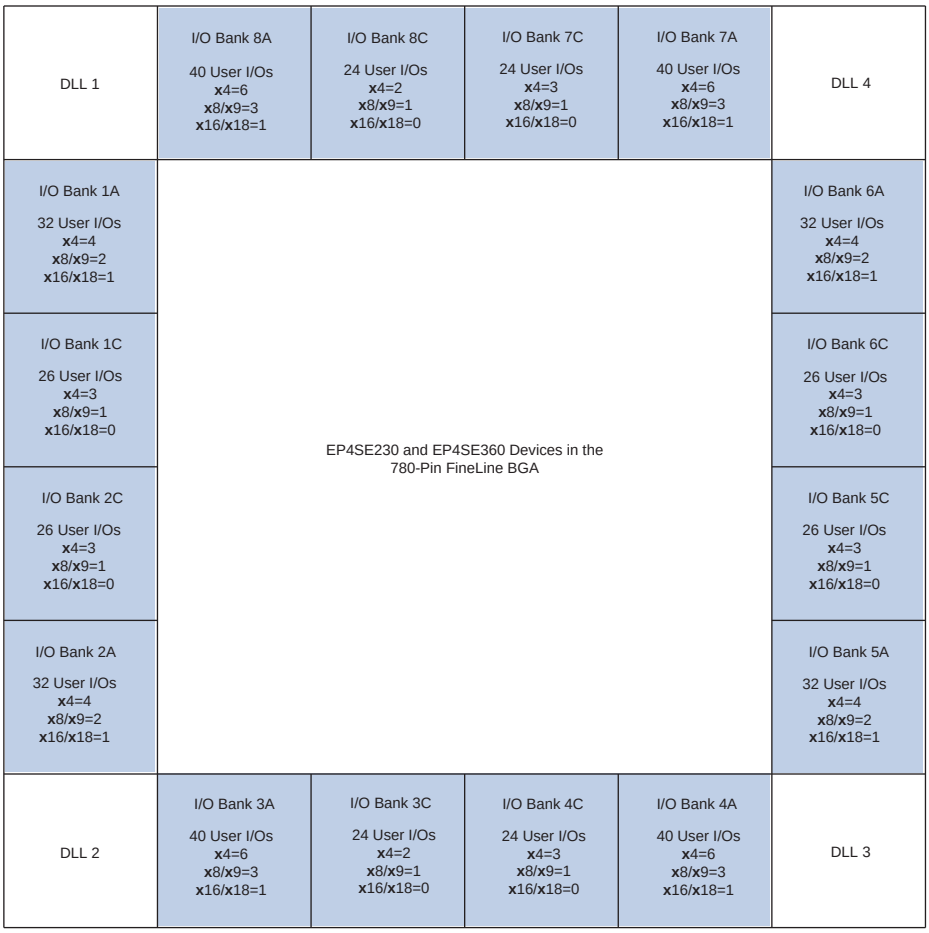
(2) EP4SGX70, EP4SGX110, EP4SGX180, and EP4SGX230 devices do not support ×32/×36 mode. To interface with a ×36 QDRII+/QDRII SRAM device, refer to "Combining ×16/×18 DQS/DQ Groups for a ×36 QDRII+/QDRII SRAM Interface" on page 7–24.

(3) You can also use DQS/DQSn pins in some of the ×4 groups as R_{UP} and R_{DN} pins, but you cannot use a ×4 group for memory interfaces if two pins of the ×4 group are used as R_{UP} and R_{DN} pins for OCT calibration. If two pins of a ×4 group are used as R_{UP} and R_{DN} pins for OCT calibration, you can use the ×16/×18 or ×32/×36 groups that include that ×4 group, however there are restrictions on using ×8/×9 groups that include that ×4 group.

(4) You can also use some of the DQS/DQ pins in I/O Bank 1C as configuration pins. You cannot use a ×4 DQS/DQ group with any of its pin members used for configuration purposes. Ensure that the DQS/DQ groups that you have chosen are not also used for configuration because you may lose up to four ×4 DQS/DQ groups, depending on your configuration scheme.

(5) All I/O pin counts include dedicated clock inputs that you can use for data inputs.

Figure 7–4. Number of DQS/DQ Groups per Bank in EP4SE230 and EP4SE360 Devices in the 780-Pin FineLine BGA Package *(Note 1)*, *(2)*, *(3)*, *(4)*, *(5)*



Notes to Figure 7–4:

(1) These numbers are preliminary until the devices are available.

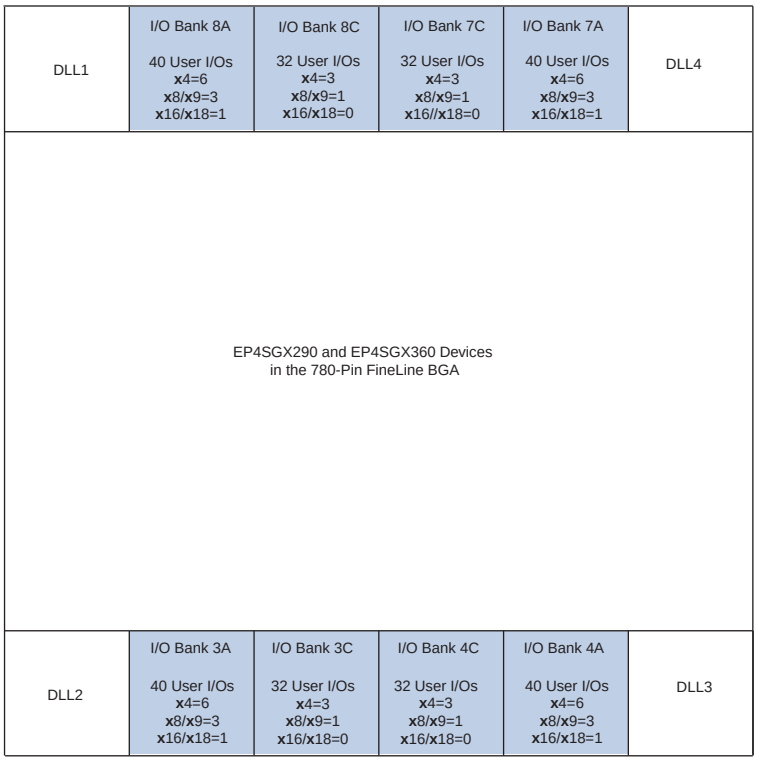
(2) EP4SE230 and EP4SE360 devices do not support ×32/×36 mode. To interface with a ×36 QDRII+/QDRII SRAM device, refer to "Combining ×16/×18 DQS/DQ Groups for a ×36 QDRII+/QDRII SRAM Interface" on page 7–24.

(3) You can also use DQS/DQSn pins in some of the ×4 groups as R_{UP} and R_{DN} pins, but you cannot use a ×4 group for memory interfaces if two pins of the ×4 group are used as R_{UP} and R_{DN} pins for OCT calibration. If two pins of a ×4 group are used as R_{UP} and R_{DN} pins for OCT calibration, you can use the ×16/×18 or ×32/×36 groups that include that ×4 group, however there are restrictions on using ×8/×9 groups that include that ×4 group.

(4) You can also use some of the DQS/DQ pins in I/O Bank 1C as configuration pins. You cannot use a ×4 DQS/DQ group with any of its pin members used for configuration purposes. Ensure that the DQS/DQ groups that you have chosen are not also used for configuration because you may lose up to four ×4 DQS/DQ groups, depending on your configuration scheme.

(5) All I/O pin counts include dedicated clock inputs that you can use for data inputs.

Figure 7–5. Number of DQS/DQ Groups per Bank in EP4SGX290 and EP4SGX360 Devices in the 780-Pin FineLine BGA Package *(Note 1)*, *(2)*

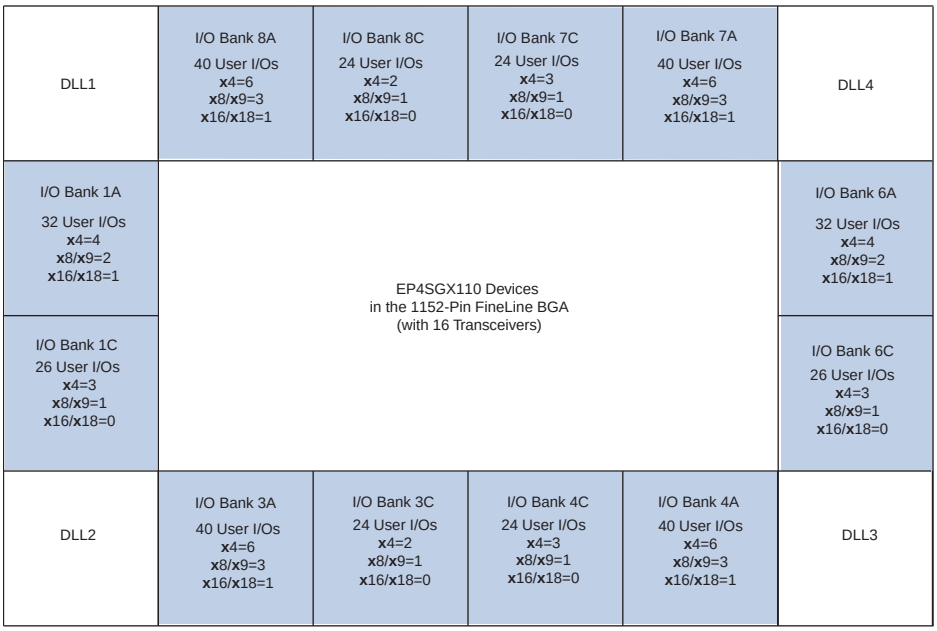


Notes to Figure 7–5:

(1) These numbers are preliminary until the devices are available.

(2) EP4SGX290 and EP4SGX360 devices do not support ×32/×36 mode. To interface with a ×36 QDRII+/QDRII SRAM device, refer to "Combining ×16/×18 DQS/DQ Groups for a ×36 QDRII+/QDRII SRAM Interface" on page 7–24.

Figure 7–6. Number of DQS/DQ Groups per Bank in EP4SGX110 Devices with 16 Transceivers in the 1152-Pin FineLine BGA Package *(Note 1)*, *(2)*, *(3)*, *(4)*, *(5)*



Notes to Figure 7–6:

(1) These numbers are preliminary until the devices are available.

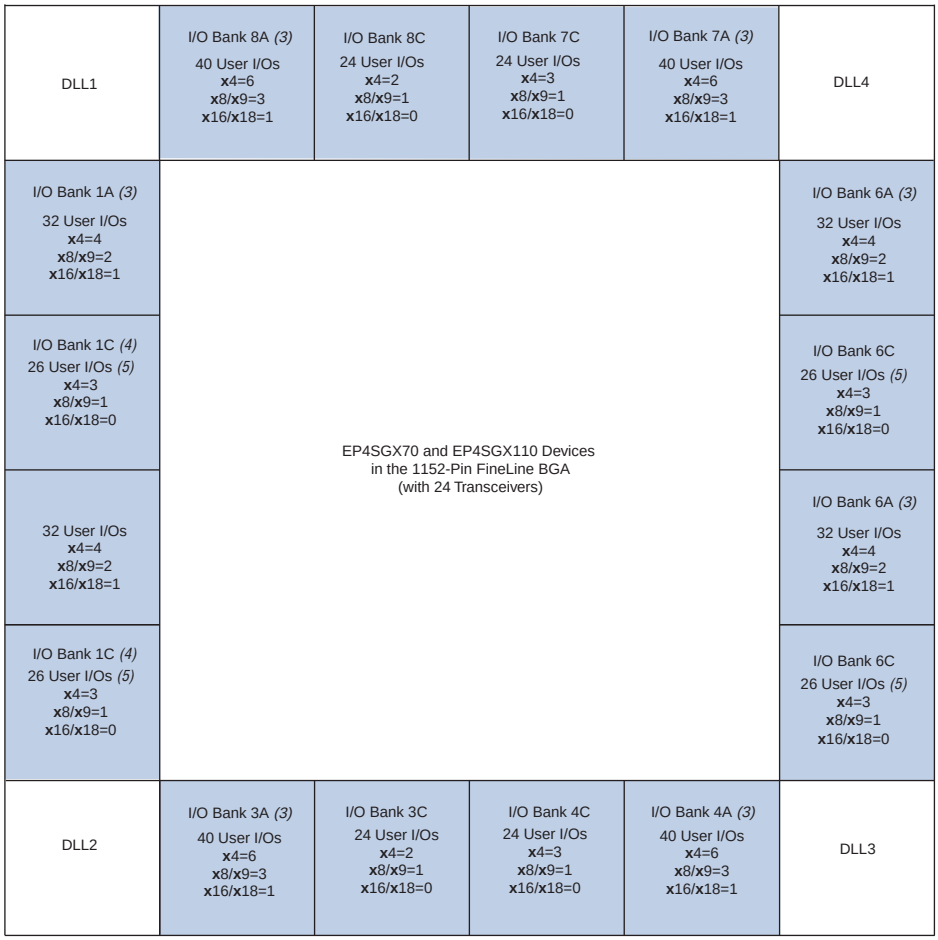
(2) EP4SGX110 devices do not support ×32/×36 mode. To interface with a ×36 QDRII+/QDRII SRAM device, refer to "Combining ×16/×18 DQS/DQ Groups for a ×36 QDRII+/QDRII SRAM Interface" on page 7–24.

(3) You can also use DQS/DQSn pins in some of the ×4 groups as R_{UP} and R_{DN} pins, but you cannot use a ×4 group for memory interfaces if two pins of the ×4 group are used as R_{UP} and R_{DN} pins for OCT calibration. If two pins of a ×4 group are used as R_{UP} and R_{DN} pins for OCT calibration, you can use the ×16/×18 or ×32/×36 groups that include that ×4 group, however there are restrictions on using ×8/×9 groups that include that ×4 group.

(4) You can also use some of the DQS/DQ pins in I/O Bank 1C as configuration pins. You cannot use a ×4 DQS/DQ group with any of its pin members used for configuration purposes. Ensure that the DQS/DQ groups that you have chosen are not also used for configuration because you may lose up to four ×4 DQS/DQ groups, depending on your configuration scheme.

(5) All I/O pin counts include dedicated clock inputs that you can use for data inputs.

Figure 7–7. Number of DQS/DQ Groups per Bank in EP4SGX70 and EP4SGX110 Devices with 24 Transceivers in the 1152-Pin FineLine BGA Package *(Note 1)*, *(2)*, *(3)*, *(4)*, *(5)*



Notes to Figure 7–7:

(1) These numbers are preliminary until the devices are available.

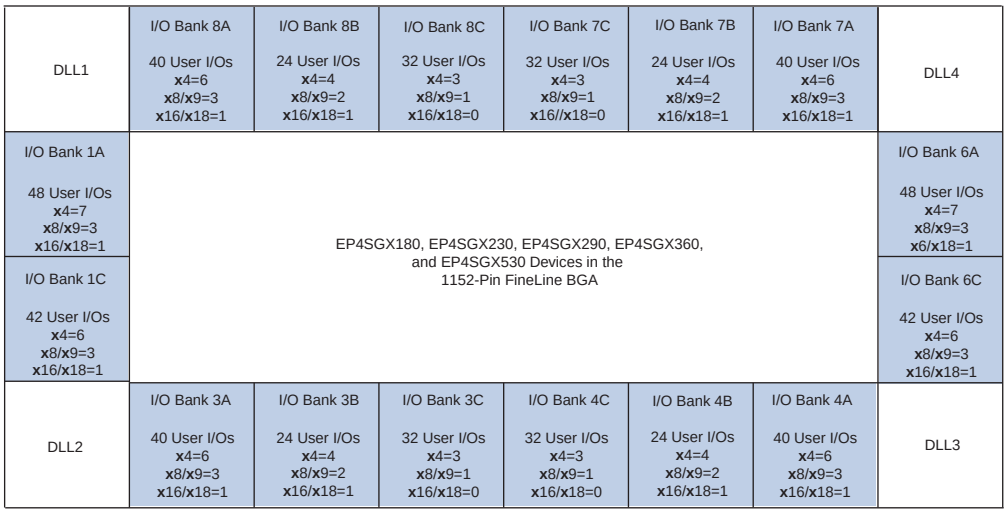
(2) EP4SGX70 and EP4SGX110 devices do not support ×32/×36 mode. To interface with a ×36 QDRII+/QDRII SRAM device, refer to "Combining ×16/×18 DQS/DQ Groups for a ×36 QDRII+/QDRII SRAM Interface" on page 7–24.

(3) You can also use DQS/DQSn pins in some of the ×4 groups as R_{UP} and R_{DN} pins, but you cannot use a ×4 group for memory interfaces if two pins of the ×4 group are used as R_{UP} and R_{DN} pins for OCT calibration. If two pins of a ×4 group are used as R_{UP} and R_{DN} pins for OCT calibration, you can use the ×16/×18 or ×32/×36 groups that include that ×4 group, however there are restrictions on using ×8/×9 groups that include that ×4 group.

(4) All I/O pin counts include dedicated clock inputs that you can use for data inputs.

(5) You can also use some of the DQS/DQ pins in I/O Bank 1C as configuration pins. You cannot use a ×4 DQS/DQ group with any of its pin members used for configuration purposes. Ensure that the DQS/DQ groups that you have chosen are not also used for configuration because you may lose up to four ×4 DQS/DQ groups, depending on your configuration scheme.

Figure 7–8. Number of DQS/DQ Groups per Bank in EP4SGX180, EP4SGX230, EP4SGX290, EP4SGX360, and EP4SGX530 Devices in the 1152-Pin FineLine BGA Package *(Note 1)*, *(2)*, *(3)*, *(4)*, *(5)*



Notes to Figure 7–8:

(1) These numbers are preliminary until the devices are available.

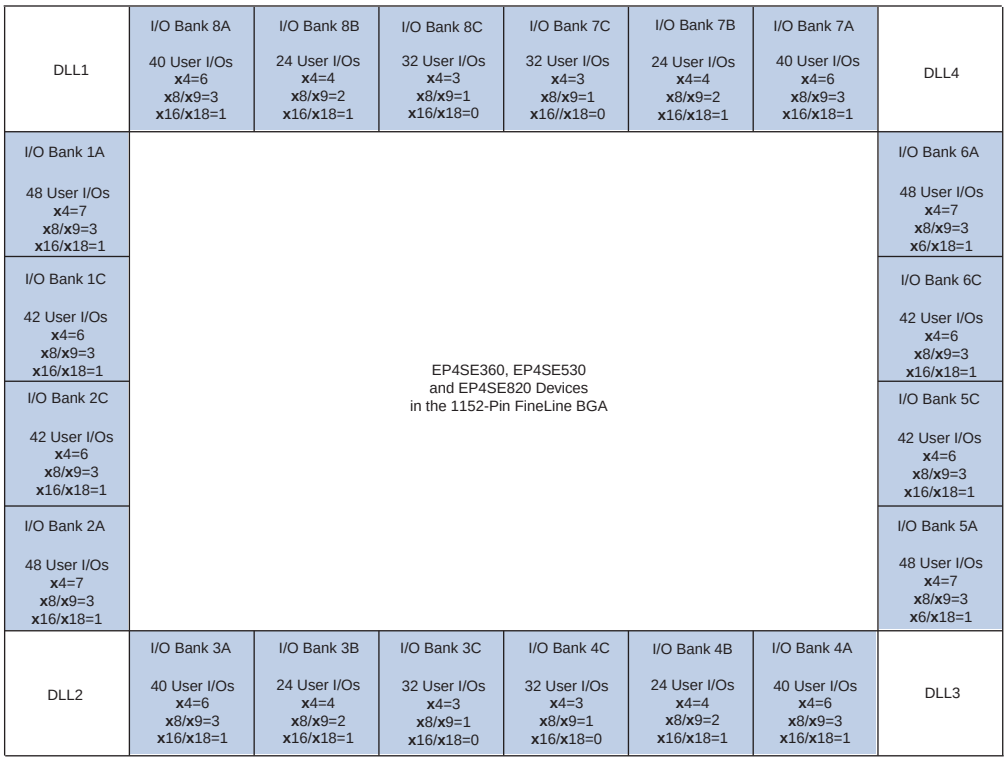
(2) EP4SGX180, EP4SGX230, EP4SGX290, EP4SGX360, and EP4SGX530 devices do not support ×32/×36 mode. To interface with a ×36 QDRII+/QDRII SRAM device, refer to "Combining ×16/×18 DQS/DQ Groups for a ×36 QDRII+/QDRII SRAM Interface" on page 7–24.

(3) You can also use DQS/DQSn pins in some of the ×4 groups as R_{UP} and R_{DN} pins, but you cannot use a ×4 group for memory interfaces if two pins of the ×4 group are used as R_{UP} and R_{DN} pins for OCT calibration. If two pins of a ×4 group are used as R_{UP} and R_{DN} pins for OCT calibration, you can use the ×16/×18 or ×32/×36 groups that include that ×4 group, however there are restrictions on using ×8/×9 groups that include that ×4 group.

(4) All I/O pin counts include dedicated clock inputs that you can use for data inputs.

(5) You can also use some of the DQS/DQ pins in I/O Bank 1C as configuration pins. You cannot use a ×4 DQS/DQ group with any of its pin members used for configuration purposes. Ensure that the DQS/DQ groups that you have chosen are not also used for configuration because you may lose up to four ×4 DQS/DQ groups, depending on your configuration scheme.

Figure 7–9. Number of DQS/DQ Groups per Bank in EP4SE360, EP4SE530, and EPSE820 Devices in the 1152-Pin FineLine BGA Package *(Note 1)*, *(2)*, *(3)*, *(4)*, *(5)*



Notes to Figure 7–9:

(1) These numbers are preliminary until the devices are available.

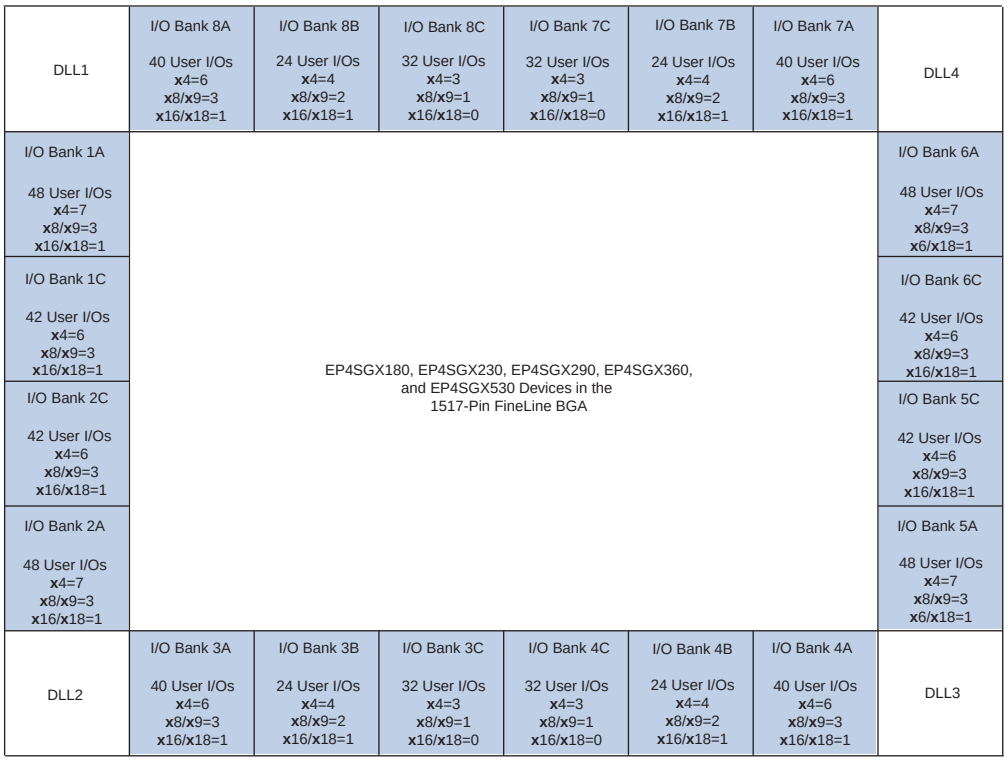
(2) EP4SE360, EP4SE530, and EPSE820 devices do not support ×32/×36 mode. To interface with a ×36 QDRII+/QDRII SRAM device, refer to "Combining ×16/×18 DQS/DQ Groups for a ×36 QDRII+/QDRII SRAM Interface" on page 7–24.

(3) You can also use DQS/DQSn pins in some of the ×4 groups as R_{UP} and R_{DN} pins, but you cannot use a ×4 group for memory interfaces if two pins of the ×4 group are used as R_{UP} and R_{DN} pins for OCT calibration. If two pins of a ×4 group are used as R_{UP} and R_{DN} pins for OCT calibration, you can use the ×16/×18 or ×32/×36 groups that include that ×4 group, however there are restrictions on using ×8/×9 groups that include that ×4 group.

(4) You can also use some of the DQS/DQ pins in I/O Bank 1C as configuration pins. You cannot use a ×4 DQS/DQ group with any of its pin members used for configuration purposes. Ensure that the DQS/DQ groups that you have chosen are not also used for configuration because you may lose up to four ×4 DQS/DQ groups, depending on your configuration scheme.

(5) All I/O pin counts include dedicated clock inputs that you can use for data inputs.

Figure 7–10. Number of DQS/DQ Groups per Bank in EP4SGX180, EP4SGX230, EP4SGX290, EP4SGX360, and EP4SGX530 Devices in the 1517-Pin FineLine BGA Package *(Note 1), (2), (3), (4), (5)*



Notes to Figure 7–10:

(1) These numbers are preliminary until the devices are available.

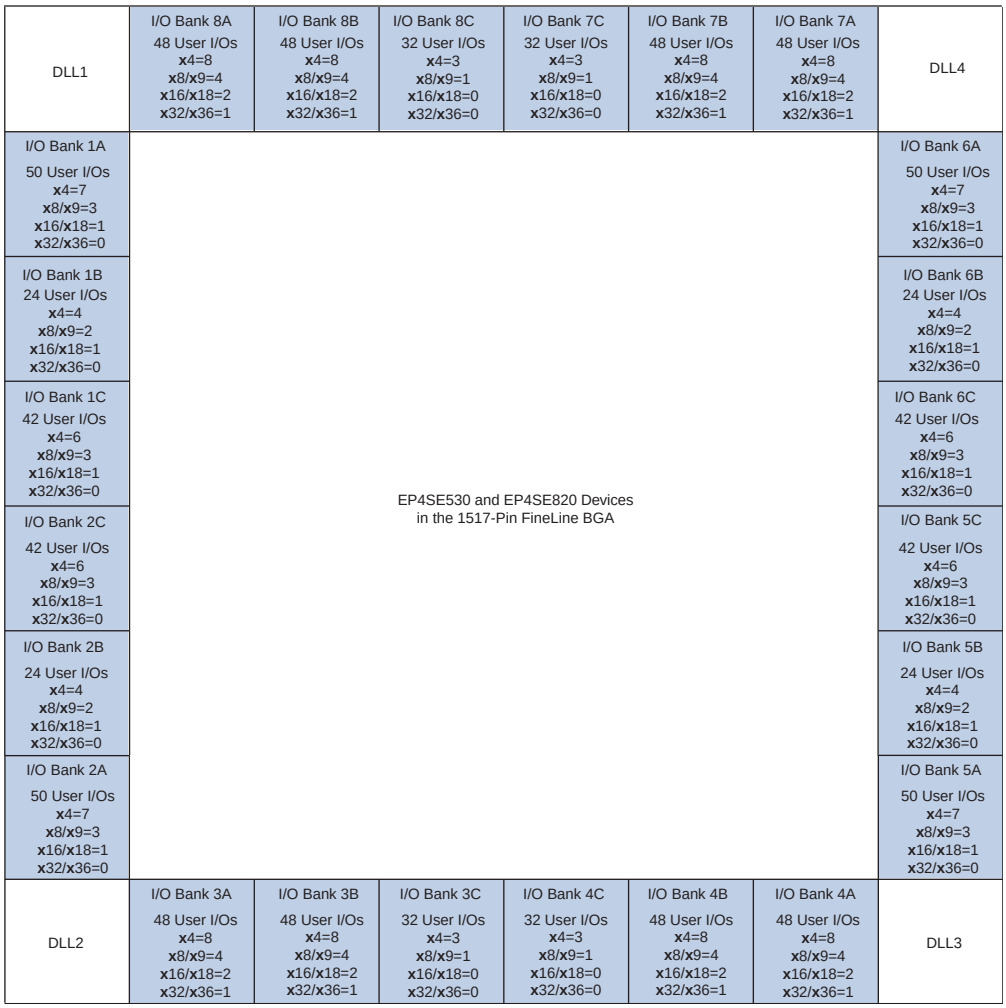
(2) EP4SGX180, EP4SGX230, EP4SGX290, EP4SGX360, and EP4SGX530 devices do not support ×32/×36 mode. To interface with a ×36 QDRII+/QDRII SRAM device, refer to "Combining ×16/×18 DQS/DQ Groups for a ×36 QDRII+/QDRII SRAM Interface" on page 7–24.

(3) You can also use DQS/DQSn pins in some of the ×4 groups as R_{UP} and R_{DN} pins, but you cannot use a ×4 group for memory interfaces if two pins of the ×4 group are used as R_{UP} and R_{DN} pins for OCT calibration. If two pins of a ×4 group are used as R_{UP} and R_{DN} pins for OCT calibration, you can use the ×16/×18 or ×32/×36 groups that include that ×4 group, however there are restrictions on using ×8/×9 groups that include that ×4 group.

(4) All I/O pin counts include dedicated clock inputs that you can use for data inputs.

(5) You can also use some of the DQS/DQ pins in I/O Bank 1C as configuration pins. You cannot use a ×4 DQS/DQ group with any of its pin members used for configuration purposes. Ensure that the DQS/DQ groups that you have chosen are not also used for configuration because you may lose up to four ×4 DQS/DQ groups, depending on your configuration scheme.

Figure 7–11. Number of DQS/DQ Groups per Bank in EP4SE530 and EP4SE820 Devices in the 1517-pin FineLine BGA Package *(Note 1)*, *(2)*, *(3)*, *(4)*



Notes to Figure 7–11:

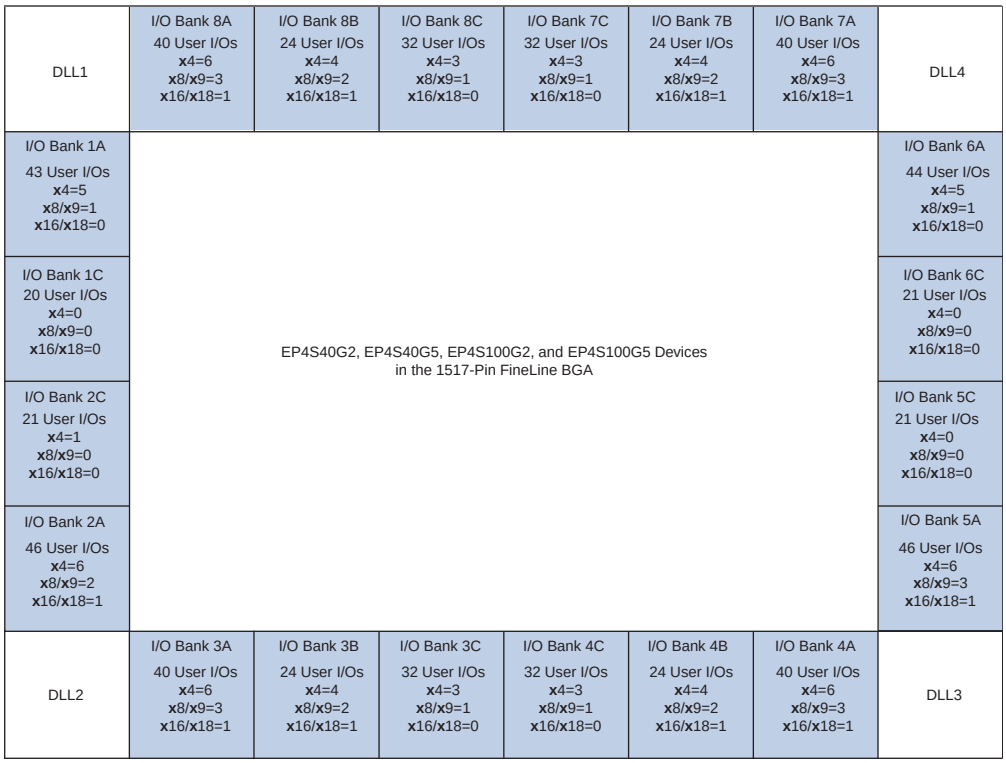
(1) These numbers are preliminary until the devices are available.

(2) You can also use DQS/DQSn pins in some of the ×4 groups as R_{UP} and R_{DN} pins, but you cannot use a ×4 group for memory interfaces if two pins of the ×4 group are used as R_{UP} and R_{DN} pins for OCT calibration. If two pins of a ×4 group are used as R_{UP} and R_{DN} pins for OCT calibration, you can use the ×16/×18 or ×32/×36 groups that include that ×4 group, however there are restrictions on using ×8/×9 groups that include that ×4 group.

(3) All I/O pin counts include dedicated clock inputs and dedicated corner PLL clock inputs that you can use for data inputs.

(4) You can also use some of the DQS/DQ pins in I/O Bank 1C as configuration pins. You cannot use a ×4 DQS/DQ group with any of its pin members used for configuration purposes. Ensure that the DQS/DQ groups that you have chosen are not also used for configuration because you may lose up to four ×4 DQS/DQ groups, depending on your configuration scheme.

Figure 7–12. Number of DQS/DQ Groups per Bank in EP4S40G2, EP4S40G5, EP4S100G2, and EP4S100G5 Devices in the 1517-Pin FineLine BGA Package *(Note 1)*, *(2)*, *(3)*, *(4)*, *(5)*



Notes to Figure 7–12:

(1) These numbers are preliminary until the devices are available.

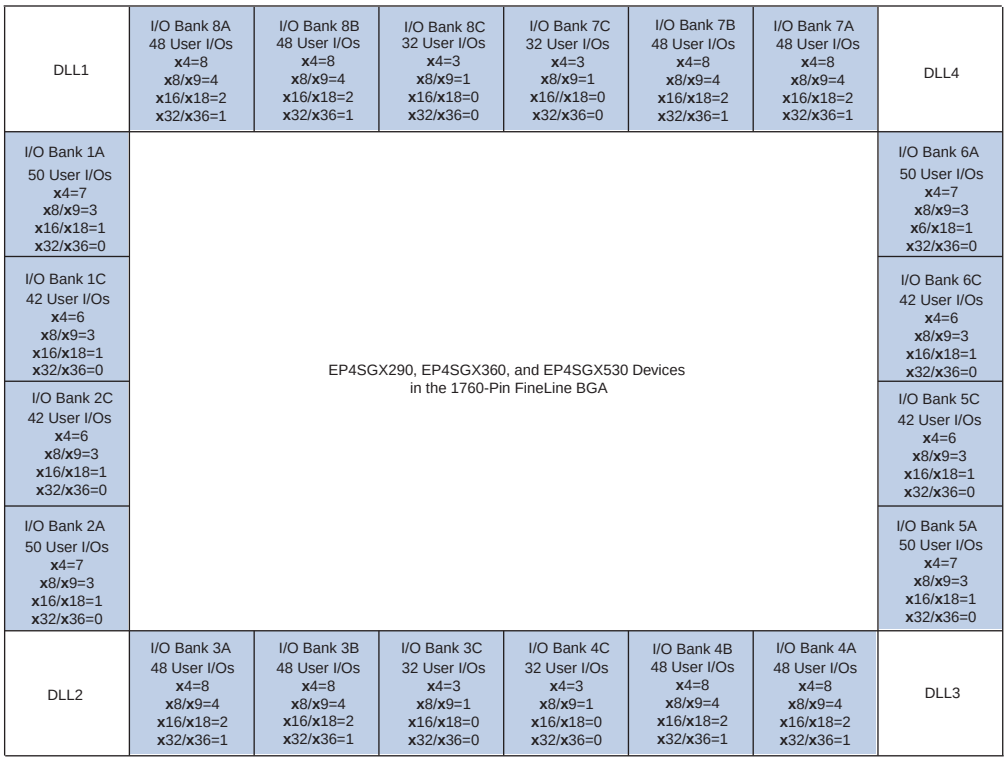
(2) EP4S40G2, EP4S40G5, EP4S100G2, and EP4S100G5 devices do not support ×32/× 36 mode. To interface with a × 36 QDRII+/QDRII SRAM device, refer to "Combining ×16/×18 DQS/DQ Groups for a ×36 QDRII+/QDRII SRAM Interface" on page 7–24.

(3) You can also use DQS/DQSn pins in some of the ×4 groups as R_{UP} and R_{DN} pins, but you cannot use a ×4 group for memory interfaces if two pins of the ×4 group are used as R_{UP} and R_{DN} pins for OCT calibration. If two pins of a ×4 group are used as R_{UP} and R_{DN} pins for OCT calibration, you can use the ×16/×18 or ×32/×36 groups that include that ×4 group, however there are restrictions on using ×8/×9 groups that include that ×4 group.

(4) All I/O pin counts include dedicated clock inputs that you can use for data inputs.

(5) You can also use some of the DQS/DQ pins in I/O Bank 1C as configuration pins. You cannot use a × 4 DQS/DQ group with any of its pin members used for configuration purposes. Make sure that the DQS/DQ groups that you have chosen are not used for configuration as you may lose up to four × 4 DQS/DQ groups, depending on your configuration scheme.

Figure 7–13. Number of DQS/DQ Groups per Bank in EP4SGX290, EP4SGX360, and EP4SGX530 Devices in the 1760-Pin FineLine BGA Package *(Note 1), (2), (3), (4)*



Notes to Figure 7–13:

(1) These numbers are preliminary until the devices are available.

(2) You can also use DQS/DQSn pins in some of the ×4 groups as R_{UP} and R_{DN} pins, but you cannot use a ×4 group for memory interfaces if two pins of the ×4 group are used as R_{UP} and R_{DN} pins for OCT calibration. If two pins of a ×4 group are used as R_{UP} and R_{DN} pins for OCT calibration, you can use the ×16/×18 or ×32/×36 groups that include that ×4 group, however there are restrictions on using ×8/×9 groups that include that ×4 group..

(3) All I/O pin counts include dedicated clock inputs and dedicated corner PLL clock inputs that you can use for data inputs.

(4) You can also use some of the DQS/DQ pins in I/O Bank 1C as configuration pins. You cannot use a ×4 DQS/DQ group with any of its pin members used for configuration purposes. Ensure that the DQS/DQ groups that you have chosen are not also used for configuration because you may lose up to four ×4 DQS/DQ groups, depending on your configuration scheme.

Figure 7–14. Number of DQS/DQ Groups per Bank in EP4SE530 Devices in the 1760-Pin FineLine BGA Package *(Note 1)*, *(2)*, *(3)*, *(4)*

DLL1	I/O Bank 8A 48 User I/Os x4=8 x8/x9=4 x16/x18=2 x32/x36=1	I/O Bank 8B 48 User I/Os x4=8 x8/x9=4 x16/x18=2 x32/x36=1	I/O Bank 8C 32 User I/Os x4=3 x8/x9=1 x16/x18=0 x32/x36=0	I/O Bank 7C 32 User I/Os x4=3 x8/x9=1 x16/x18=0 x32/x36=0	I/O Bank 7B 48 User I/Os x4=8 x8/x9=4 x16/x18=2 x32/x36=1	I/O Bank 7A 48 User I/Os x4=8 x8/x9=4 x16/x18=2 x32/x36=1	DLL4
I/O Bank 1A 50 User I/Os x4=7 x8/x9=3 x16/x18=1 x32/x36=0							I/O Bank 6A 50 User I/Os x4=7 x8/x9=3 x16/x18=1 x32/x36=0
I/O Bank 1B 24 User I/Os x4=4 x8/x9=2 x16/x18=1 x32/x36=0							I/O Bank 6B 24 User I/Os x4=4 x8/x9=2 x16/x18=1 x32/x36=0
I/O Bank 1C 42 User I/Os x4=6 x8/x9=3 x16/x18=1 x32/x36=0							I/O Bank 6C 42 User I/Os x4=6 x8/x9=3 x16/x18=1 x32/x36=0
I/O Bank 2C 42 User I/Os x4=6 x8/x9=3 x16/x18=1 x32/x36=0			EP4SE530 Devices in the 1760-Pin FineLine BGA				I/O Bank 5C 42 User I/Os x4=6 x8/x9=3 x16/x18=1 x32/x36=0
I/O Bank 2B 24 User I/Os x4=4 x8/x9=2 x16/x18=1 x32/x36=0							I/O Bank 5B 24 User I/Os x4=4 x8/x9=2 x16/x18=1 x32/x36=0
I/O Bank 2A 50 User I/Os x4=7 x8/x9=3 x16/x18=1 x32/x36=0							I/O Bank 5A 50 User I/Os x4=7 x8/x9=3 x16/x18=1 x32/x36=0
DLL2	I/O Bank 3A 48 User I/Os x4=8 x8/x9=4 x16/x18=2 x32/x36=1	I/O Bank 3B 48 User I/Os x4=8 x8/x9=4 x16/x18=2 x32/x36=1	I/O Bank 3C 32 User I/Os x4=3 x8/x9=1 x16/x18=0 x32/x36=0	I/O Bank 4C 32 User I/Os x4=3 x8/x9=1 x16/x18=0 x32/x36=0	I/O Bank 4B 48 User I/Os x4=8 x8/x9=4 x16/x18=2 x32/x36=1	I/O Bank 4A 48 User I/Os x4=8 x8/x9=4 x16/x18=2 x32/x36=1	DLL3

Notes to Figure 7–14:

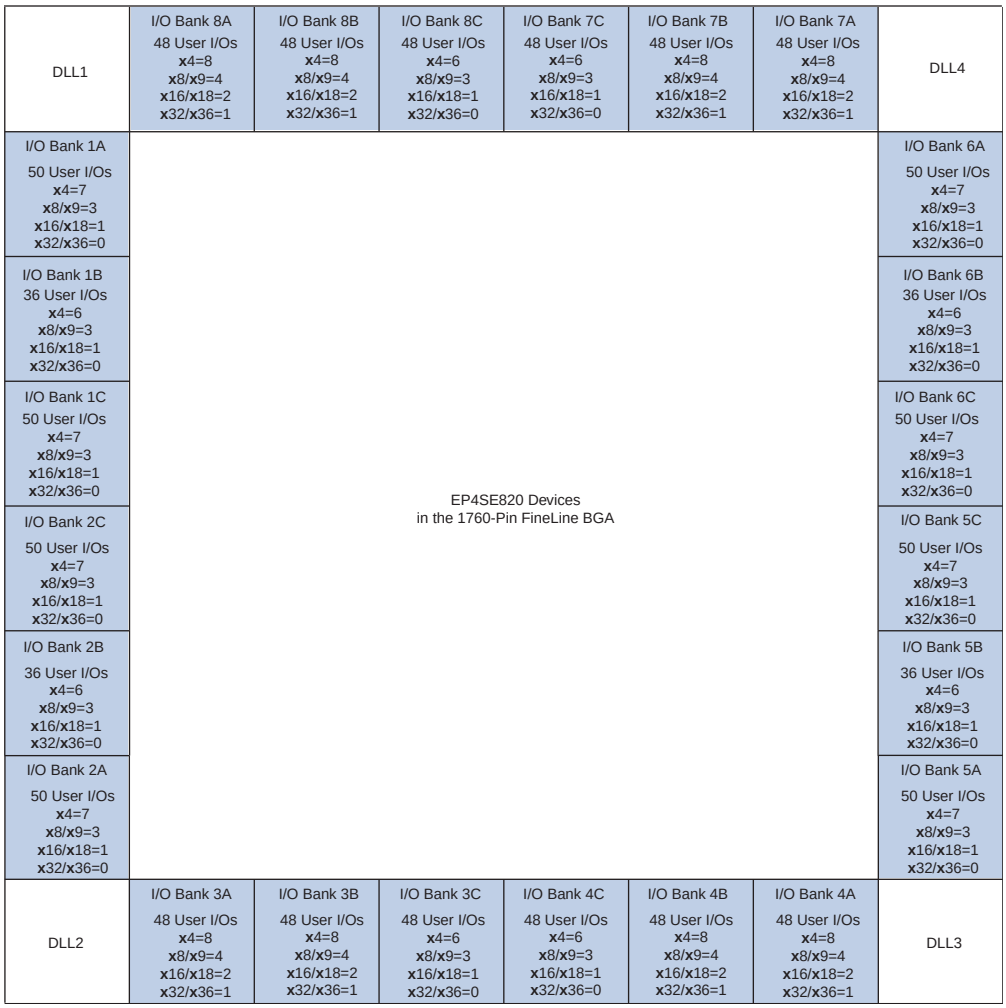
(1) These numbers are preliminary until the devices are available.

(2) You can also use DQS/DQSn pins in some of the ×4 groups as R_{UP} and R_{DN} pins, but you cannot use a ×4 group for memory interfaces if two pins of the ×4 group are used as R_{UP} and R_{DN} pins for OCT calibration. If two pins of a ×4 group are used as R_{UP} and R_{DN} pins for OCT calibration, you can use the ×16/×18 or ×32/×36 groups that include that ×4 group, however there are restrictions on using ×8/×9 groups that include that ×4 group.

(3) All I/O pin counts include dedicated clock inputs and dedicated corner PLL clock inputs that you can use for data inputs.

(4) You can also use some of the DQS/DQ pins in I/O Bank 1C as configuration pins. You cannot use a ×4 DQS/DQ group with any of its pin members used for configuration purposes. Ensure that the DQS/DQ groups that you have chosen are not also used for configuration because you may lose up to four ×4 DQS/DQ groups, depending on your configuration scheme.

Figure 7–15. Number of DQS/DQ Groups per Bank in EP4SE820 Devices in the 1760-pin FineLine BGA Package *(Note 1)*, *(2)*, *(3)*, *(4)*



Notes to Figure 7–15:

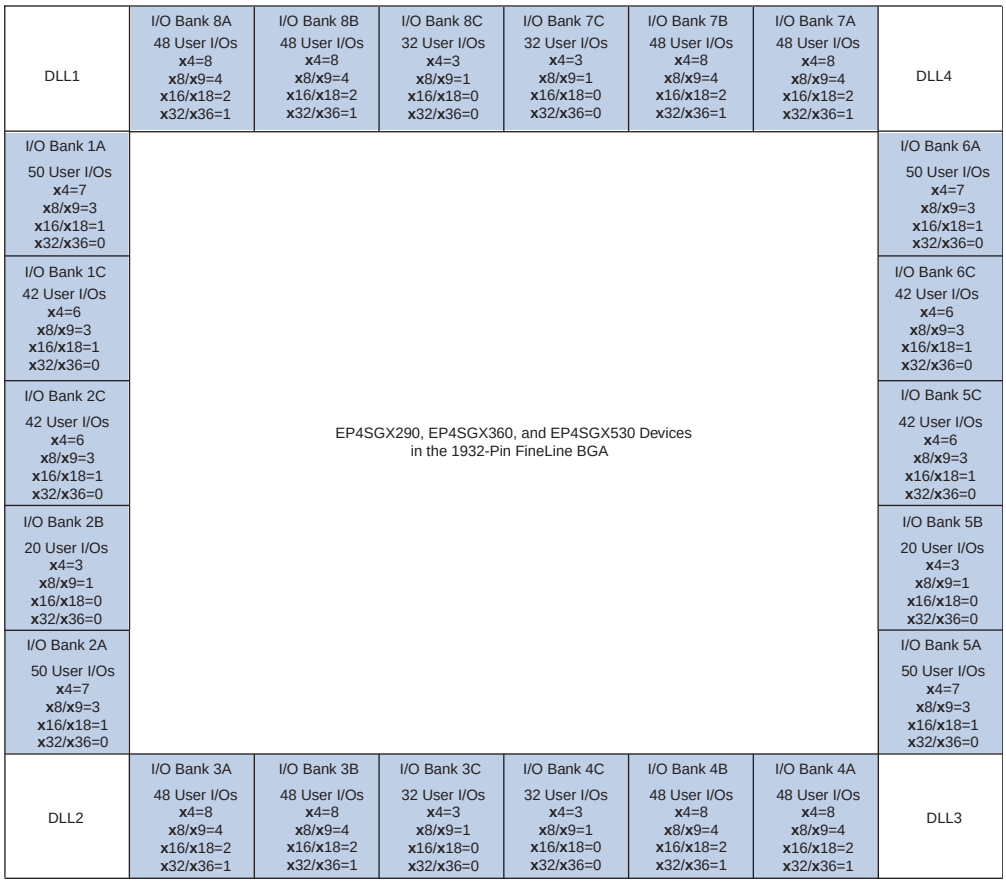
(1) These numbers are preliminary until the devices are available.

(2) You can also use DQS/DQSn pins in some of the ×4 groups as R_{UP} and R_{DN} pins, but you cannot use a ×4 group for memory interfaces if two pins of the ×4 group are used as R_{UP} and R_{DN} pins for OCT calibration. If two pins of a ×4 group are used as R_{UP} and R_{DN} pins for OCT calibration, you can use the ×16/×18 or ×32/×36 groups that include that ×4 group, however there are restrictions on using ×8/×9 groups that include that ×4 group.

(3) All I/O pin counts include dedicated clock inputs and dedicated corner PLL clock inputs that you can use for data inputs.

(4) You can also use some of the DQS/DQ pins in I/O Bank 1C as configuration pins. You cannot use a ×4 DQS/DQ group with any of its pin members used for configuration purposes. Ensure that the DQS/DQ groups that you have chosen are not also used for configuration because you may lose up to four ×4 DQS/DQ groups, depending on your configuration scheme.

Figure 7–16. Number of DQS/DQ Groups per Bank in EP4SGX290, EP4SGX360, and EP4SGX530 Devices in the 1932-Pin FineLine BGA Package *(Note 1), (2), (3), (4)*



Notes to Figure 7–16:

(1) These numbers are preliminary until the devices are available.

(2) You can also use DQS/DQSn pins in some of the ×4 groups as R_{UP} and R_{DN} pins, but you cannot use a ×4 group for memory interfaces if two pins of the ×4 group are used as R_{UP} and R_{DN} pins for OCT calibration. If two pins of a ×4 group are used as R_{UP} and R_{DN} pins for OCT calibration, you can use the ×16/×18 or ×32/×36 groups that include that ×4 group, however there are restrictions on using ×8/×9 groups that include that ×4 group.

(3) All I/O pin counts include dedicated clock inputs and dedicated corner PLL clock inputs that you can use for data inputs.

(4) You can also use some of the DQS/DQ pins in I/O Bank 1C as configuration pins. You cannot use a ×4 DQS/DQ group with any of its pin members used for configuration purposes. Ensure that the DQS/DQ groups that you have chosen are not also used for configuration because you may lose up to four ×4 DQS/DQ groups, depending on your configuration scheme.

Figure 7–17. Number of DQS/DQ Groups per Bank in EP4S100G3, EP4S100G4, and EP4S100G5 Devices in the 1932-Pin FineLine BGA Package *(Note 1)*, *(2)*, *(3)*, *(4)*

DLL1	I/O Bank 8A 48 User I/Os x4=8 x8/x9=4 x16/x18=2 x32/x36=1	I/O Bank 8B 48 User I/Os x4=8 x8/x9=4 x16/x18=2 x32/x36=1	I/O Bank 8C 32 User I/Os x4=3 x8/x9=1 x16/x18=0 x32/x36=0	I/O Bank 7C 32 User I/Os x4=3 x8/x9=1 x16/x18=0 x32/x36=0	I/O Bank 7B 48 User I/Os x4=8 x8/x9=4 x16/x18=2 x32/x36=1	I/O Bank 7A 48 User I/Os x4=8 x8/x9=4 x16/x18=2 x32/x36=1	DLL4
I/O Bank 1A 40 User I/Os x4=3 x8/x9=1 x16/x18=0 x32/x36=0							I/O Bank 6A 38 User I/Os x4=3 x8/x9=0 x16/x18=0 x32/x36=0
I/O Bank 1C 19 User I/Os x4=0 x8/x9=0 x16/x18=0 x32/x36=0							I/O Bank 6C 20 User I/Os x4=0 x8/x9=0 x16/x18=0 x32/x36=0
I/O Bank 2C 19 User I/Os x4=0 x8/x9=0 x16/x18=0 x32/x36=0			EP4S100G3, EP4S100G4, and EP4S100G5 Devices in the 1932-Pin FineLine BGA				I/O Bank 5C 17 User I/Os x4=0 x8/x9=0 x16/x18=0 x32/x36=0
I/O Bank 2B 13 User I/Os x4=1 x8/x9=0 x16/x18=0 x32/x36=0							I/O Bank 5B 12 User I/Os x4=0 x8/x9=0 x16/x18=0 x32/x36=0
I/O Bank 2A 39 User I/Os x4=4 x8/x9=1 x16/x18=0 x32/x36=0							I/O Bank 5A 40 User I/Os x4=4 x8/x9=1 x16/x18=0 x32/x36=0
DLL2	I/O Bank 3A 48 User I/Os x4=8 x8/x9=4 x16/x18=2 x32/x36=1	I/O Bank 3B 48 User I/Os x4=8 x8/x9=4 x16/x18=2 x32/x36=1	I/O Bank 3C 32 User I/Os x4=3 x8/x9=1 x16/x18=0 x32/x36=0	I/O Bank 4C 32 User I/Os x4=3 x8/x9=1 x16/x18=0 x32/x36=0	I/O Bank 4B 48 User I/Os x4=8 x8/x9=4 x16/x18=2 x32/x36=1	I/O Bank 4A 48 User I/Os x4=8 x8/x9=4 x16/x18=2 x32/x36=1	DLL3

Notes to Figure 7–17:

(1) These numbers are preliminary until the devices are available.

(2) You can also use DQS/DQSn pins in some of the ×4 groups as R_{UP} and R_{DN} pins, but you cannot use a ×4 group for memory interfaces if two pins of the ×4 group are used as R_{UP} and R_{DN} pins for OCT calibration. If two pins of a ×4 group are used as R_{UP} and R_{DN} pins for OCT calibration, you can use the ×16/×18 or ×32/×36 groups that include that ×4 group, however there are restrictions on using ×8/×9 groups that include that ×4 group.

(3) All I/O pin counts include dedicated clock inputs and dedicated corner PLL clock inputs that you can use for data inputs.

(4) You can also use some of the DQS/DQ pins in I/O Bank 1C as configuration pins. You cannot use a ×4 DQS/DQ group with any of its pin members used for configuration purposes. Ensure that the DQS/DQ groups that you have chosen are not also used for configuration because you may lose up to four ×4 DQS/DQ groups, depending on your configuration scheme.

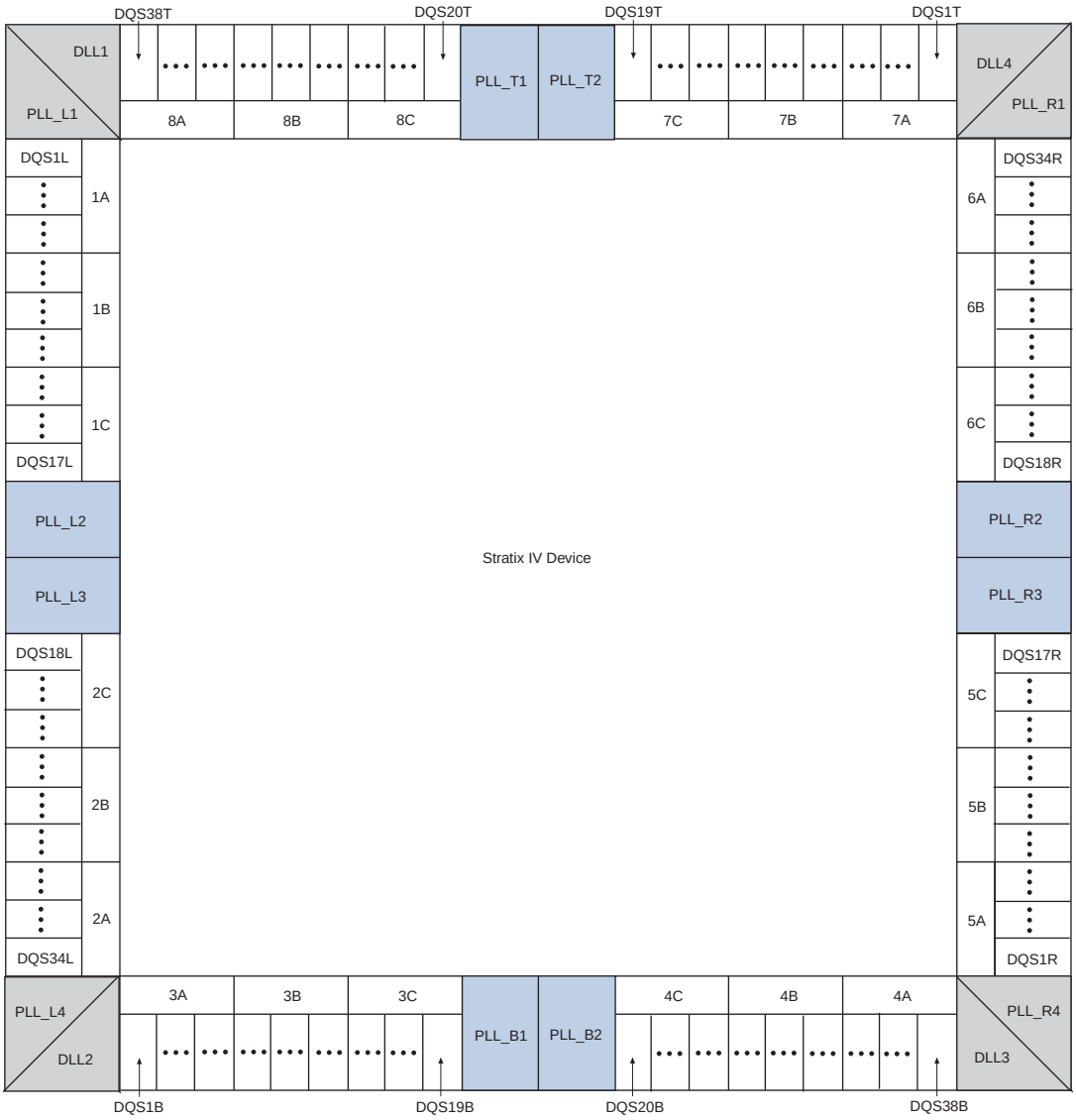
The DQS and DQSn pins are listed in the Stratix IV pin tables as `DQSXY` and `DQSnXY`, respectively, where `X` indicates the DQS/DQ grouping number and `Y` indicates whether the group is located on the top (T), bottom (B), left (L), or right (R) side of the device. The DQS/DQ pin numbering is based on ×4 mode.

The corresponding DQ pins are marked as `DQXY`, where `X` indicates which DQS group the pins belong to and `Y` indicates whether the group is located on the top (T), bottom (B), left (L), or right (R) side of the device. For example, `DQS1L` indicates a DQS pin located on the left side of the device. The DQ pins belonging to that group are shown as `DQ1L` in the pin table. For more information, refer to Figure 7–18.

The parity, DM, BWSn, NWSn, ECC, and QVLD pins are shown as DQ pins in the pin table.

The numbering scheme starts from the top-left side of the device going counter-clockwise in a die-top view. Figure 7–18 shows how the DQS/DQ groups are numbered in a die-top view of the device. The top and bottom sides of the device can contain up to 38 ×4 DQS/DQ groups. The left and right sides of the device can contain up to 34 ×4 DQS/DQ groups.

Figure 7–18. DQS Pins in Stratix IV I/O Banks



Using the R_{UP} and R_{DN} Pins in a DQS/DQ Group Used for Memory Interfaces

You can use the DQS/DQSn pins in some of the ×4 groups as R_{UP} and R_{DN} pins (listed in the pin table). You cannot use a ×4 DQS/DQ group for memory interfaces if any of its pin members are used as R_{UP} and R_{DN} pins for OCT calibration. You may be able to use the ×8/×9 group that includes this ×4 DQS/DQ group, if either of the following applies:

- You are not using DM pins with your differential DQS pins
- You are not using complementary or differential DQS pins

You can use the ×8/×9 group because a DQS/DQ ×8/×9 group actually comprises 12 pins, as the groups are formed by stitching two DQS/DQ groups in ×4 mode with six pins each (refer to Table 7–2 on page 7–6). A typical ×8 memory interface consists of one DQS, one DM, and eight DQ pins that add up to 10 pins. If you choose your pin assignment carefully, you can use the two extra pins for R_{UP} and R_{DN}. In a DDR3 SDRAM interface, you must use differential DQS, which means that you only have one extra pin. In this case, pick different pin locations for the R_{UP} and R_{DN} pins (for example, in the bank that contains the address and command pins).

You cannot use the R_{UP} and R_{DN} pins shared with DQS/DQ group pins when using ×9 QDRII+/QDRII SRAM devices, as the R_{UP} and R_{DN} pins are dual purpose with the CQn pins. In this case, pick different pin locations for R_{UP} and R_{DN} pins to avoid conflict with memory interface pin placement. In this case, you have the choice of placing the R_{UP} and R_{DN} pins in the data-write group or in the same bank as the address and command pins.

There is no restriction of using ×16/×18 or ×32/×36 DQS/DQ groups that include the ×4 groups whose pin members are being used as R_{UP} and R_{DN} pins as there are enough extra pins that can be used as DQS pins.

For ×8, ×16/×18, or ×32/×36 DQS/DQ groups whose members are used for R_{UP} and R_{DN}, you must assign DQS and DQ pins manually. The Quartus® II software might not be able to place DQS and DQ pins without manual pin assignments, resulting in a "no-fit".

Combining ×16/×18 DQS/DQ Groups for a ×36 QDRII+/QDRII SRAM Interface

This implementation combines ×16/×18 DQS/DQ groups to interface with a ×36 QDRII+/QDRII SRAM device. The ×36 read data bus uses two ×16/×18 groups while the ×36 write data uses another two ×16/×18 or four ×8/×9 groups. The CQ/CQn signal traces are split on the board trace to connect to two pairs of CQ/CQn pins in the FPGA. This is the only connection on the board that you need to change for this implementation. Other QDRII+/QDRII SRAM interface rules for Stratix IV devices also apply for this implementation.

The ALTMEMPHY megafunction and UniPHY-based external memory interface IPs do not use the QVLD signal, so you can leave the QVLD signal unconnected as in any QDRII+/QDRII SRAM interfaces.

For more information about the ALTMEMPHY megafunction or UniPHY-based IPs, refer to the *External Memory Interface Handbook*.

Rules to Combine Groups

In 780-, 1152-, and some 1517-pin package devices, there is at most one ×16/×18 group per I/O sub-bank. You can combine two ×16/×18 groups from a single side of the device for a ×36 interface.

For devices that do not have four ×16/×18 groups in a single side of the device to form two ×36 groups for read and write data, you can form one ×36 group on one side of the device and another ×36 group on the other side of the device.

For vertical migration with the ×36 emulation implementation, check if migration is possible by enabling device migration in the Quartus II project. The Quartus II software supports the use of four ×8/×9 DQ groups for write data pins and migration of these groups across device density. Table 7–4 lists the possible combinations to use two ×16/×18 DQS/DQ groups to form a ×32/×36 group on Stratix IV devices lacking a native ×32/×36 DQS/DQ group.

Table 7–4. Possible Group Combinations in Stratix IV Devices (Part 1 of 2)

Package	Device Density	I/O Sub-Bank Combinations
780-Pin FineLine BGA	■ EP4SGX70 ■ EP4SGX110 ■ EP4SGX180 ■ EP4SGX230 ■ EP4SGX290 ■ EP4SGX360	3A and 4A, 7A and 8A (bottom and top I/O banks) *(1)*
	■ EP4SE230 ■ EP4SE360	1A and 2A, 5A and 6A (left and right I/O banks) 3A and 4A, 7A and 8A (bottom and top I/O banks) *(1)*
1152-Pin FineLine BGA	■ EP4SGX70 ■ EP4SGX110	3A and 4A, 7A and 8A (bottom and top I/O banks) *(1)*
	■ EP4SGX180 ■ EP4SGX230 ■ EP4SGX290 ■ EP4SGX360 ■ EP4SGX530	1A and 1C, 6A and 6C (left and right I/O banks) 3A and 3B, 4A and 4B (bottom I/O banks) 7A and 7B, 8A and 8B (top I/O banks)
	■ EP4SE360 ■ EP4SE530 ■ EP4SE820	1A and 1C, 2A and 2C (left I/O banks) 3A and 3B, 4A and 4B (bottom I/O banks) 5A and 5C, 6A and 6C (right I/O banks) 7A and 7B, 8A and 8B (top I/O banks)

Table 7–4. Possible Group Combinations in Stratix IV Devices (Part 2 of 2)

Package	Device Density	I/O Sub-Bank Combinations
1517-Pin FineLine BGA	■ EP4SGX180 ■ EP4SGX230 ■ EP4SGX290 ■ EP4SGX360 ■ EP4SGX530	1A and 1C, 2A and 2C (left I/O banks) 3A and 3B, 4A and 4B (bottom I/O banks) 5A and 5C, 6A and 6C (right I/O banks) 7A and 7B, 8A and 8B (top I/O banks)
	■ EP4SE530 *(2)* ■ EP4SE820 *(2)*	1A and 1B, 2A and 2B or 1B and 1C, 2B and 2C (left I/O banks) *(3)* 5A and 5B, 6A and 6B or 5B and 5C, 6B and 6C (right I/O banks) *(3)*
	■ EP4S40G2 ■ EP4S40G5 ■ EP4S100G2 ■ EP4S100G5	3A and 3B, 4A and 4B (bottom I/O banks) 7A and 7B, 8A and 8B (top I/O banks)
1760-Pin FineLine BGA	■ EP4SGX290 ■ EP4SGX360 ■ EP4SGX530	1A and 1C, 2A and 2C (left I/O banks) 3A and 3B, 4A and 4B (bottom I/O banks) 5A and 5C, 6A and 6C (right I/O banks) 7A and 7B, 8A and 8B (top I/O banks)
	■ EP4SE530 *(2)* ■ EP4SE820 *(2)*	1A and 1B, 2A and 2B or 1B and 1C, 2B and 2C (left I/O banks) *(3)* 5A and 5B, 6A and 6B or 5B and 5C, 6B and 6C (right I/O banks) *(3)*
1932-Pin FineLine BGA	■ EP4SGX290 *(2)* ■ EP4SGX360 *(2)* ■ EP4SGX530 *(2)*	1A and 1C, 2A and 2C (left I/O banks) 5A and 5C, 6A and 6C (right I/O banks)

Notes to Table 7–4:

(1) Each side of the device in these packages has four remaining ×8/×9 groups. You can combine them for the write side (only) if you want to keep the ×36 QDRII+/QDRII SRAM interface on one side of the device. You must change the **Memory Interface Data Group** default assignment from the default **18** to **9** in this case.

(2) This device supports ×36 DQS/DQ groups on the top and bottom I/O banks natively.

(3) Although it is possible to combine the ×16/×18 DQS/DQ groups from I/O banks 1A and 1C, 2A and 2C, 5A and 5C, and 6A and 6C, Altera does not recommend this due to the size of the package. Similarly, crossing a bank number (for example, combining groups from I/O banks 6C and 5C) is not supported in this package.

Stratix IV External Memory Interface Features

Stratix IV devices are rich with features that allow robust high-performance external memory interfacing. The ALTMEMPHY megafunction allows you to use these external memory interface features and helps set up the physical interface (PHY) best suited for your system. This section describes each Stratix IV device feature that is used in external memory interfaces from the DQS phase-shift circuitry, DQS logic block, leveling multiplexers, and dynamic OCT control block.

The ALTMEMPHY megafunction and the Altera® memory controller MegaCore® functions can run at half the frequency of the I/O interface of the memory devices to allow better timing management in high-speed memory interfaces. Stratix IV devices have built-in registers in the IOE to convert data from full-rate (the I/O frequency) to half-rate (the controller frequency) and vice versa. You can bypass these registers if your memory controller is not running at half the rate of the I/O frequency. When using the Altera memory controller MegaCore functions, the ALTMEMPHY megafunction is instantiated for you.

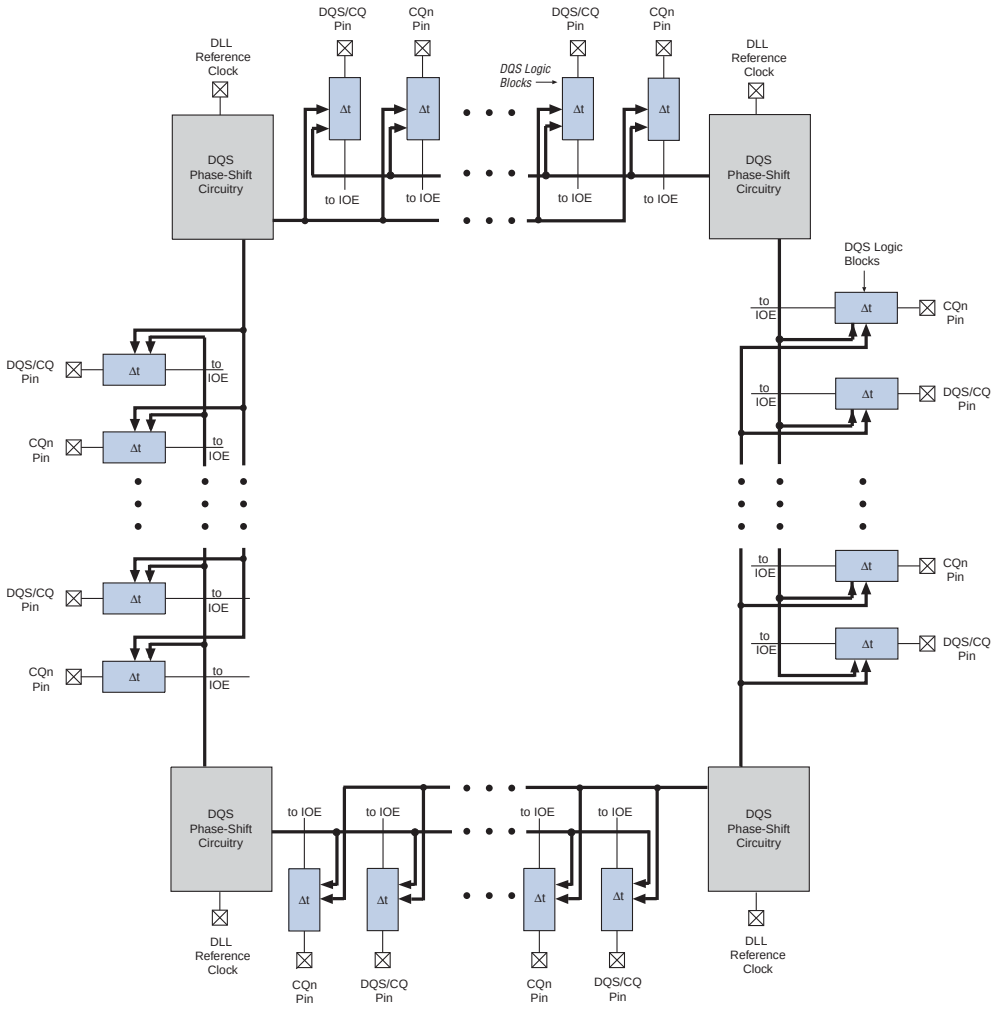
For more information about the ALTMEMPHY megafunction, refer to the *External Memory PHY Interface Megafunction User Guide (ALTMEMPHY)*.

DQS Phase-Shift Circuitry

Stratix IV phase-shift circuitry provides phase shift to the DQS/CQ and CQn pins on read transactions when the DQS/CQ and CQn pins are acting as input clocks or strobes to the FPGA. The DQS phase-shift circuitry consists of DLLs that are shared between multiple DQS pins and the phase-offset module to further fine-tune the DQS phase shift for different sides of the device.

Figure 7–19 shows how the DQS phase-shift circuitry is connected to the DQS/CQ and CQn pins in the device where memory interfaces are supported on all sides of the Stratix IV device.

Figure 7–19. DQS/CQ and CQn Pins and DQS Phase-Shift Circuitry *(Note 1)*, *(2)*



Notes to Figure 7–19:

(1) For possible reference input clock pins for each DLL, refer to "DLL" on page 7–28.

(2) You can configure each DQS/CQ and CQn pin with a phase shift based on one of two possible DLL output settings.

DQS phase-shift circuitry is connected to the DQS logic blocks that control each DQS/CQ or CQn pin. The DQS logic blocks allow the DQS delay settings to be updated concurrently at every DQS/CQ or CQn pin.

DLL

DQS phase-shift circuitry uses a DLL to dynamically control the clock delay needed by the DQS/CQ and CQn pin. The DLL, in turn, uses a frequency reference to dynamically generate control signals for the delay chains in each of the DQS/CQ and CQn pins, allowing it to compensate for PVT variations. The DQS delay settings are Gray-coded to reduce jitter when the DLL updates the settings. The phase-shift circuitry needs 1280 clock cycles to lock and calculate the correct input clock period

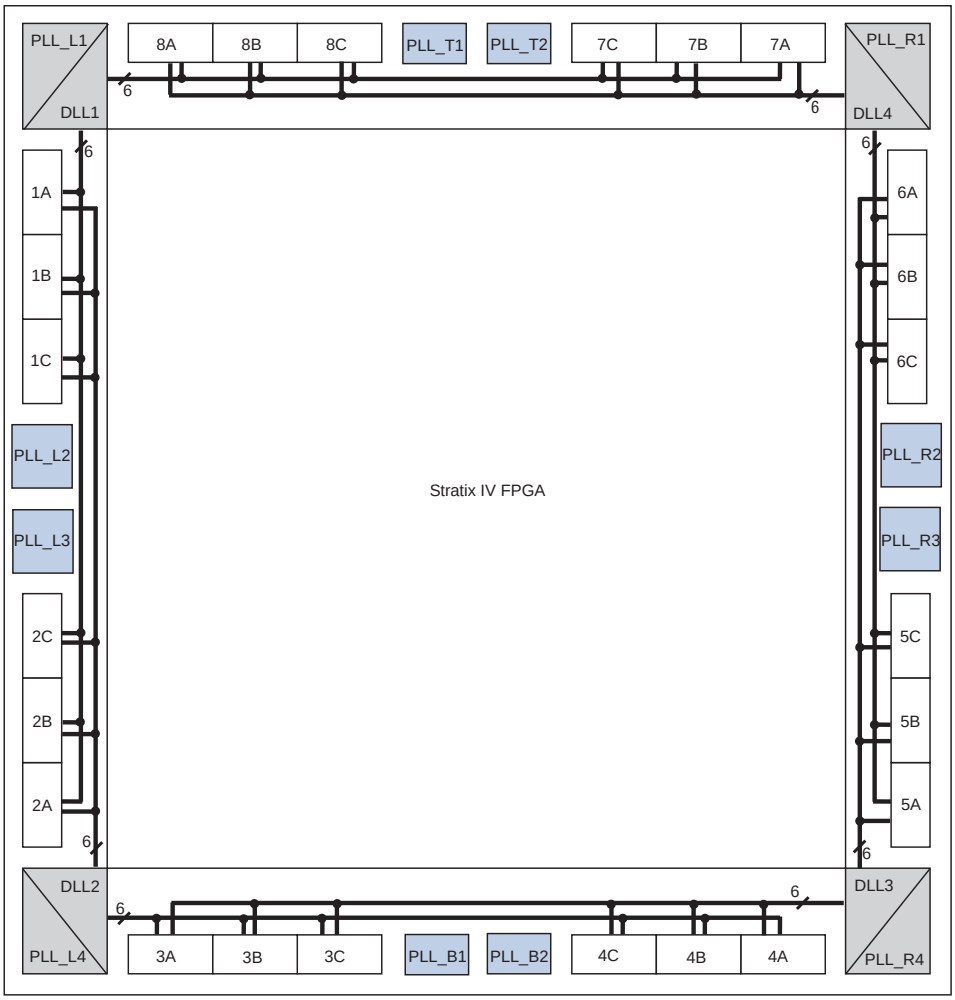
when the DLL is in low jitter mode. Otherwise, only 256 clock cycles are needed. Do not send data during these clock cycles because there is no guarantee that it will be captured properly. As the settings from the DLL may not be stable until this lock period has elapsed, be aware that anything using these settings (including the leveling delay system) may be unstable during this period.

You can still use the DQS phase-shift circuitry for any memory interfaces that are less than 100 MHz. However, the DQS signal may not shift over 2.5 ns. Even if the DQS signal is not shifted exactly to the middle of the DQ valid window, the I/O element should still be able to capture the data in low-frequency applications in which a large amount of timing margin is available.

There are a maximum of four DLLs in a Stratix IV device, located in each corner of the device. These four DLLs support a maximum of four unique frequencies, with each DLL running at one frequency. Each DLL can have two outputs with different phase offsets, which allows one Stratix IV device to have eight different DLL phase shift settings.

Figure 7–20 shows the DLL and I/O bank locations in Stratix IV devices from a die-top view if all sides of the device support external memory interfaces.

Figure 7–20. Stratix IV DLL and I/O Bank Locations (Die-Top View)



The DLL can access the two adjacent sides from its location within the device. For example, DLL1 on the top left of the device can access the top side (I/O banks 7A, 7B, 7C, 8A, 8B, and 8C) and the left side of the device (I/O banks 1A, 1B, 1C, 2A, 2B, and 2C). This means that each I/O bank is accessible by two DLLs, giving more flexibility to create multiple frequencies and multiple-type interfaces. You can have two different interfaces with the same frequency on the two sides adjacent to a DLL, where the DLL controls the DQS delay settings for both interfaces.

Each bank can use settings from either or both DLLs the bank is adjacent to. For example, `DQS1L` can get its phase-shift settings from DLL1, while `DQS2L` can get its phase-shift settings from DLL2. Table 7–5 lists the DLL location and supported I/O banks for Stratix IV devices.

☞ You can only have one memory interface in each I/O sub-bank (such as I/O sub-banks 1A, 1B, and 1C) when you use leveling delay chains. This is because there is only one leveling delay chain per I/O sub-bank.

Table 7–5. DLL Location and Supported I/O Banks

DLL	Location	Accessible I/O Banks *(1)*
DLL1	Top-left corner	1A, 1B, 1C, 2A, 2B, 2C, 7A, 7B, 7C, 8A, 8B, 8C
DLL2	Bottom-left corner	1A, 1B, 1C, 2A, 2B, 2C, 3A, 3B, 3C, 4A, 4B, 4C
DLL3	Bottom-right corner	3A, 3B, 3C, 4A, 4B, 4C, 5A, 5B, 5C, 6A, 6B, 6C
DLL4	Top-right corner	5A, 5B, 5C, 6A, 6B, 6C, 7A, 7B, 7C, 8A, 8B, 8C

Note to Table 7–5:

(1) The DLL can access these I/O banks if they are available for memory interfacing.

The reference clock for each DLL may come from PLL output clocks or any of the two dedicated clock input pins located in either side of the DLL. Table 7–6 through Table 7–17 show the available DLL reference clock input resources for the Stratix IV device family.

When you have a dedicated PLL that only generates the DLL input reference clock, set the PLL mode to **No Compensation** to achieve better performance or the Quartus II software changes it automatically. Because the PLL does not use any other outputs, it does not need to compensate for any clock paths.

Table 7–6. DLL Reference Clock Input for EP4SGX70, EP4SGX110, EP4SGX180, and EP4SGX230 Devices in the 780-Pin FineLine BGA Package

DLL	CLKIN (Top/Bottom)	CLKIN (Left/Right)	PLL (Top/Bottom)	PLL (Left/Right)	PLL (Corner)
DLL1	CLK12P CLK13P CLK14P CLK15P	CLK0P CLK1P CLK2P CLK3P	PLL_T1	PLL_L2	—
DLL2	CLK4P CLK5P CLK6P CLK7P	CLK0P CLK1P CLK2P CLK3P	PLL_B1	—	—
DLL3	CLK4P CLK5P CLK6P CLK7P	—	PLL_B1	—	—
DLL4	CLK12P CLK13P CLK14P CLK15P	—	PLL_T1	—	—

Table 7–7. DLL Reference Clock Input for EP4SE230 and EP4SE360 Devices in the 780-Pin FineLine BGA Package

DLL	CLKIN (Top/Bottom)	CLKIN (Left/Right)	PLL (Top/Bottom)	PLL (Left/Right)	PLL (Corner)
DLL1	CLK12P CLK13P CLK14P CLK15P	CLK0P CLK1P CLK2P CLK3P	PLL_T1	PLL_L2	—
DLL2	CLK4P CLK5P CLK6P CLK7P	CLK0P CLK1P CLK2P CLK3P	PLL_B1	PLL_L2	—
DLL3	CLK4P CLK5P CLK6P CLK7P	CLK8P CLK9P CLK10P CLK11P	PLL_B1	PLL_R2	—
DLL4	CLK12P CLK13P CLK14P CLK15P	CLK8P CLK9P CLK10P CLK11P	PLL_T1	PLL_R2	—

Table 7–8. DLL Reference Clock Input for EP4SGX290 and EP4SGX360 Devices in the 780-Pin FineLine BGA Package

DLL	CLKIN (Top/Bottom)	CLKIN (Left/Right)	PLL (Top/Bottom)	PLL (Left/Right)	PLL (Corner)
DLL1	CLK12P CLK13P CLK14P CLK15P	—	PLL_T1	—	—
DLL2	CLK4P CLK5P CLK6P CLK7P	—	PLL_B1	—	—
DLL3	CLK4P CLK5P CLK6P CLK7P	—	PLL_B2	—	—
DLL4	CLK12P CLK13P CLK14P CLK15P	—	PLL_T2	—	—

Table 7–9. DLL Reference Clock Input for EP4SGX70 and EP4SGX110 Devices in the 1152-Pin FineLine BGA Package (with 24 Transceivers)

DLL	CLKIN (Top/Bottom)	CLKIN (Left/Right)	PLL (Top/Bottom)	PLL (Left/Right)	PLL (Corner)
DLL1	CLK12P CLK13P CLK14P CLK15P	CLK0P CLK1P CLK2P CLK3P	PLL_T1	PLL_L2	—
DLL2	CLK4P CLK5P CLK6P CLK7P	CLK0P CLK1P CLK2P CLK3P	PLL_B1	PLL_L2	—
DLL3	CLK4P CLK5P CLK6P CLK7P	CLK8P CLK9P CLK10P CLK11P	PLL_B1	PLL_R2	—
DLL4	CLK12P CLK13P CLK14P CLK15P	CLK8P CLK9P CLK10P CLK11P	PLL_T1	PLL_R2	—

Table 7–10. DLL Reference Clock Input for EP4SGX110 Devices in the 1152-Pin FineLine BGA Package (with 16 Transceivers)

DLL	CLKIN (Top/Bottom)	CLKIN (Left/Right)	PLL (Top/Bottom)	PLL (Left/Right)	PLL (Corner)
DLL1	CLK12P CLK13P CLK14P CLK15P	CLK0P CLK1P	PLL_T1	PLL_L2	—
DLL2	CLK4P CLK5P CLK6P CLK7P	CLK0P CLK1P	PLL_B1	—	—
DLL3	CLK4P CLK5P CLK6P CLK7P	CLK10P CLK11P	PLL_B1	—	—
DLL4	CLK12P CLK13P CLK14P CLK15P	CLK10P CLK11P	PLL_T1	PLL_R2	—

Table 7–11. DLL Reference Clock Input for EP4SGX180, EP4SGX230, EP4SGX290, EP4SGX360, and EP4SGX530 Devices in the 1152-Pin FineLine BGA Package

DLL	CLKIN (Top/Bottom)	CLKIN (Left/Right)	PLL (Top/Bottom)	PLL (Left/Right)	PLL (Corner)
DLL1	CLK12P CLK13P CLK14P CLK15P	CLK0P CLK1P	PLL_T1	PLL_L2	—
DLL2	CLK4P CLK5P CLK6P CLK7P	CLK0P CLK1P	PLL_B1	—	—
DLL3	CLK4P CLK5P CLK6P CLK7P	CLK10P CLK11P	PLL_B2	—	—
DLL4	CLK12P CLK13P CLK14P CLK15P	CLK10P CLK11P	PLL_T2	PLL_R2	—

Table 7–12. DLL Reference Clock Input for EP4SE530 and EPSE820 Devices in the 1152-, 1517-, and 1760-Pin FineLine BGA Packages

DLL	CLKIN (Top/Bottom)	CLKIN (Left/Right)	PLL (Top/Bottom)	PLL (Left/Right)	PLL (Corner)
DLL1	CLK12P CLK13P CLK14P CLK15P	CLK0P CLK1P CLK2P CLK3P	PLL_T1	PLL_L2	—
DLL2	CLK4P CLK5P CLK6P CLK7P	CLK0P CLK1P CLK2P CLK3P	PLL_B1	PLL_L3	—
DLL3	CLK4P CLK5P CLK6P CLK7P	CLK8P CLK9P CLK10P CLK11P	PLL_B2	PLL_R3	—
DLL4	CLK12P CLK13P CLK14P CLK15P	CLK8P CLK9P CLK10P CLK11P	PLL_T2	PLL_R2	—

Table 7–13. DLL Reference Clock Input for EP4SGX180, EP4SGX230, EP4SGX290, EP4SGX360, and EP4SGX530 Devices in the 1517-Pin FineLine BGA Package

DLL	CLKIN (Top/Bottom)	CLKIN (Left/Right)	PLL (Top/Bottom)	PLL (Left/Right)	PLL (Corner)
DLL1	CLK12P CLK13P CLK14P CLK15P	CLK0P CLK1P CLK2P CLK3P	PLL_T1	PLL_L2	—
DLL2	CLK4P CLK5P CLK6P CLK7P	CLK0P CLK1P CLK2P CLK3P	PLL_B1	PLL_L3	—
DLL3	CLK4P CLK5P CLK6P CLK7P	CLK8P CLK9P CLK10P CLK11P	PLL_B2	PLL_R3	—
DLL4	CLK12P CLK13P CLK14P CLK15P	CLK8P CLK9P CLK10P CLK11P	PLL_T2	PLL_R2	—

Table 7–14. DLL Reference Clock Input for EP4S40G2, EP4S40G5, EP4S100G2, and EP4S100G5 Devices in the 1517-Pin FineLine BGA Package

DLL	CLKIN (Top/Bottom)	CLKIN (Left/Right)	PLL (Top/Bottom)	PLL (Left/Right)	PLL (Corner)
DLL1	CLK12P CLK13P CLK14P CLK15P	CLK1P CLK3P	PLL_T1	PLL_L2	—
DLL2	CLK4P CLK5P CLK6P CLK7P	CLK1P CLK3P	PLL_B1	PLL_L3	—
DLL3	CLK4P CLK5P CLK6P CLK7P	CLK8P CLK10P	PLL_B2	PLL_R3	—
DLL4	CLK12P CLK13P CLK14P CLK15P	CLK8P CLK10P	PLL_T2	PLL_R2	—

Table 7–15. DLL Reference Clock Input for EP4SGX290, EP4SGX360, and EP4SGX530 Devices in the 1760-Pin FineLine BGA Package

DLL	CLKIN (Top/Bottom)	CLKIN (Left/Right)	PLL (Top/Bottom)	PLL (Left/Right)	PLL (Corner)
DLL1	CLK12P CLK13P CLK14P CLK15P	CLK0P CLK1P CLK2P CLK3P	PLL_T1	PLL_L2	—
DLL2	CLK4P CLK5P CLK6P CLK7P	CLK0P CLK1P CLK2P CLK3P	PLL_B1	PLL_L3	—
DLL3	CLK4P CLK5P CLK6P CLK7P	CLK8P CLK9P CLK10P CLK11P	PLL_B2	PLL_R3	—
DLL4	CLK12P CLK13P CLK14P CLK15P	CLK8P CLK9P CLK10P CLK11P	PLL_T2	PLL_R2	—

Table 7–16. DLL Reference Clock Input for EP4SGX290, EP4SGX360, and EP4SGX530 Devices in the 1932-Pin FineLine BGA Package

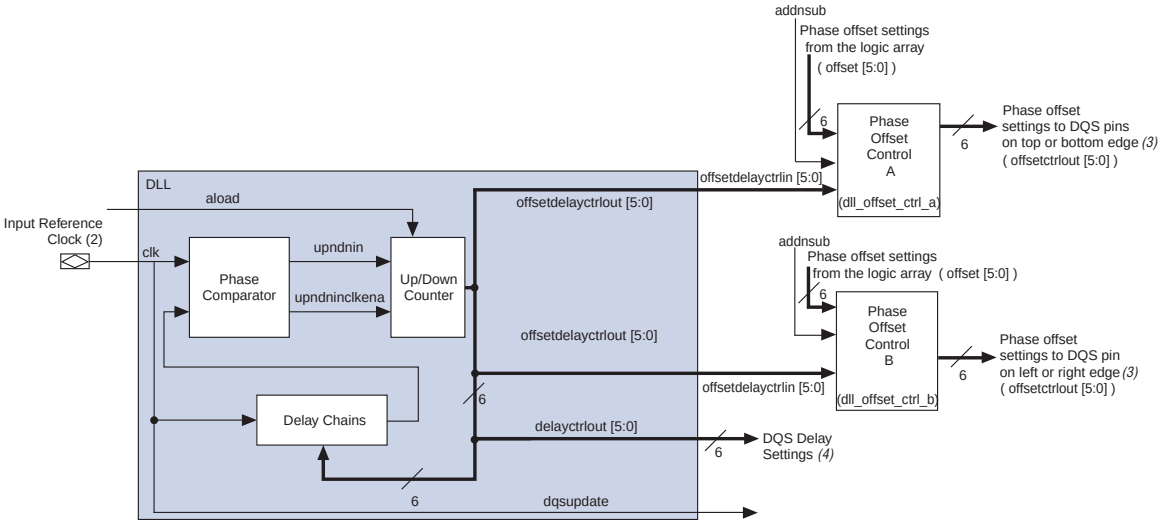
DLL	CLKIN (Top/Bottom)	CLKIN (Left/Right)	PLL (Top/Bottom)	PLL (Left/Right)	PLL (Corner)
DLL1	CLK12P CLK13P CLK14P CLK15P	CLK0P CLK1P CLK2P CLK3P	PLL_T1	PLL_L2	PLL_L1
DLL2	CLK4P CLK5P CLK6P CLK7P	CLK0P CLK1P CLK2P CLK3P	PLL_B1	PLL_L3	PLL_L4
DLL3	CLK4P CLK5P CLK6P CLK7P	CLK8P CLK9P CLK10P CLK11P	PLL_B2	PLL_R3	PLL_R4
DLL4	CLK12P CLK13P CLK14P CLK15P	CLK8P CLK9P CLK10P CLK11P	PLL_T2	PLL_R2	PLL_R1

Table 7–17. DLL Reference Clock Input for EP4S100G3, EP4S100G4, and EP4S100G5 Devices in the 1932-Pin FineLine BGA Package

DLL	CLKIN (Top/Bottom)	CLKIN (Left/Right)	PLL (Top/Bottom)	PLL (Left/Right)	PLL (Corner)
DLL1	CLK12P CLK13P CLK14P CLK15P	—	PLL_T1	PLL_L2	PLL_L1
DLL2	CLK4P CLK5P CLK6P CLK7P	—	PLL_B1	PLL_L3	PLL_L4
DLL3	CLK4P CLK5P CLK6P CLK7P	CLK9P CLK11P	PLL_B2	PLL_R3	PLL_R4
DLL4	CLK12P CLK13P CLK14P CLK15P	CLK9P CLK11P	PLL_T2	PLL_R2	PLL_R1

Figure 7–21 shows a simple block diagram of the DLL. The input reference clock goes into the DLL to a chain of up to 16 delay elements. The phase comparator compares the signal coming out of the end of the delay chain block to the input reference clock. The phase comparator then issues the upndn signal to the Gray-code counter. This signal increments or decrements a six-bit delay setting (DQS delay settings) that increases or decreases the delay through the delay element chain to bring the input reference clock and the signals coming out of the delay element chain in phase.

Figure 7–21. Simplified Diagram of the DQS Phase-Shift Circuitry *(Note 1)*



Notes to Figure 7–21:

(1) All features of the DQS phase-shift circuitry are accessible from the ALTMEMPHY megafunction in the Quartus II software.

(2) The input reference clock for the DQS phase-shift circuitry can come from a PLL output clock or an input clock pin. For more information, refer to Table 7–6 through Table 7–17.

(3) Phase offset settings can only go to the DQS logic blocks.

(4) DQS delay settings can go to the logic array, DQS logic block, and leveling circuitry.

The phase offset control block 'A' is designated as `DLLOFFSETCTRL_<coordinate x>_<coordinate y>_N1` and phase offset control block 'B' is designated as `DLLOFFSETCTRL_<coordinate x>_<coordinate y>_N2` in the Quartus II assignment.

You can reset the DLL from either the logic array or a user I/O pin. Each time the DLL is reset, you must wait for 1280 clock cycles for the DLL to lock before you can capture the data properly.

Depending on the DLL frequency mode, the DLL can shift the incoming DQS signals by 0°, 22.5°, 30°, 36°, 45°, 60°, 67.5°, 72°, 90°, 108°, 120°, 135°, 144°, or 180°. The shifted DQS signal is then used as the clock for the DQ IOE input registers.

All DQS/CQ and CQn pins, referenced to the same DLL, can have their input signal phase shifted by a different degree amount but all must be referenced at one particular frequency. For example, you can have a 90° phase shift on `DQS1T` and a 60° phase shift on `DQS2T`, referenced from a 200-MHz clock. Not all phase-shift combinations are supported. The phase shifts on the DQS pins referenced by the same DLL must all be a multiple of 22.5° (up to 90°), 30° (up to 120°), 36° (up to 144°), or 45° (up to 180°).

There are seven different frequency modes for the Stratix IV DLL, as shown in Table 7–18. Each frequency mode provides different phase shift selections. In frequency mode 0, 1, 2, and 3, the 6-bit DQS delay settings vary with PVT to implement the phase-shift delay. In frequency modes 4, 5, and 6, only 5 bits of the DQS delay settings vary with PVT to implement the phase-shift delay; the most significant bit of the DQS delay setting is set to 0.

Table 7–18. Stratix IV DLL Frequency Modes

Frequency Mode	Available Phase Shift	Number of Delay Chains
0	22.5, 45, 67.5, 90	16
1	30, 60, 90, 120	12
2	36, 72, 108, 144	10
3	45, 90, 135, 180	8
4	30, 60, 90, 120	12
5	36, 72, 108, 144	10
6	45, 90, 135, 180	8

For the frequency range of each mode, refer to the *DC and Switching Characteristics* chapter.

For 0° shift, the DQS/CQ signal bypasses both the DLL and DQS logic blocks. The Quartus II software automatically sets the DQ input delay chains so that the skew between the DQ and DQS/CQ pin at the DQ IOE registers is negligible when 0° shift is implemented. You can feed the DQS delay settings to the DQS logic block and logic array.

The shifted DQS/CQ signal goes to the DQS bus to clock the IOE input registers of the DQ pins. The signal can also go into the logic array for resynchronization if you are not using IOE resynchronization registers. The shifted CQn signal can only go to the negative-edge input register in the DQ IOE and is only used for QDRII+ and QDRII SRAM interfaces.

Phase Offset Control

Each DLL has two phase-offset modules and can provide two separate DQS delay settings with independent offset, one for the top and bottom I/O bank and one for the left and right I/O bank, so you can fine-tune the DQS phase-shift settings between two different sides of the device. Even though you have independent phase offset control, the frequency of the interface using the same DLL has to be the same. Use the

phase offset control module for making small shifts to the input signal and use the DQS phase-shift circuitry for larger signal shifts. For example, if the DLL only offers a multiple of 30° phase shift, but your interface needs a 67.5° phase shift on the DQS signal, you can use two delay chains in the DQS logic blocks to give you 60° phase shift and use the phase offset control feature to implement the extra 7.5° phase shift.

You can either use a static phase offset or a dynamic phase offset to implement the additional phase shift. The available additional phase shift is implemented in 2s: complement in Gray-code between settings –64 to +63 for frequency mode 0, 1, 2, and 3, and between settings –32 to +31 for frequency modes 4, 5, and 6. An additional bit indicates whether the setting has a positive or negative value. The settings are linear, each phase offset setting adds a delay amount specified in the *DC and Switching Characteristics* chapter. The DQS phase shift is the sum of the DLL delay settings and the user-selected phase offset settings whose top setting is 64 for frequency modes 0, 1, 2, and 3; and 32 for frequency modes 4, 5, and 6, so the actual physical offset setting range is 64 or 32 subtracted by the DQS delay settings from the DLL.

When using this feature, you need to monitor the DQS delay settings to know how many offsets you can add and subtract in the system. Note that the DQS delay settings output by the DLL are also Gray coded.

For example, if the DLL determines that DQS delay settings of 28 is needed to achieve a 30° phase shift in DLL frequency mode 1, you can subtract up to 28 phase offset settings and you can add up to 35 phase offset settings to achieve the optimal delay that you need. However, if the same DQS delay settings of 28 is needed to achieve 30° phase shift in DLL frequency mode 4, you can still subtract up to 28 phase offset settings, but you can only add up to 3 phase offset settings before the DQS delay settings reach their maximum settings because DLL frequency mode 4 only uses 5-bit DLL delay settings.

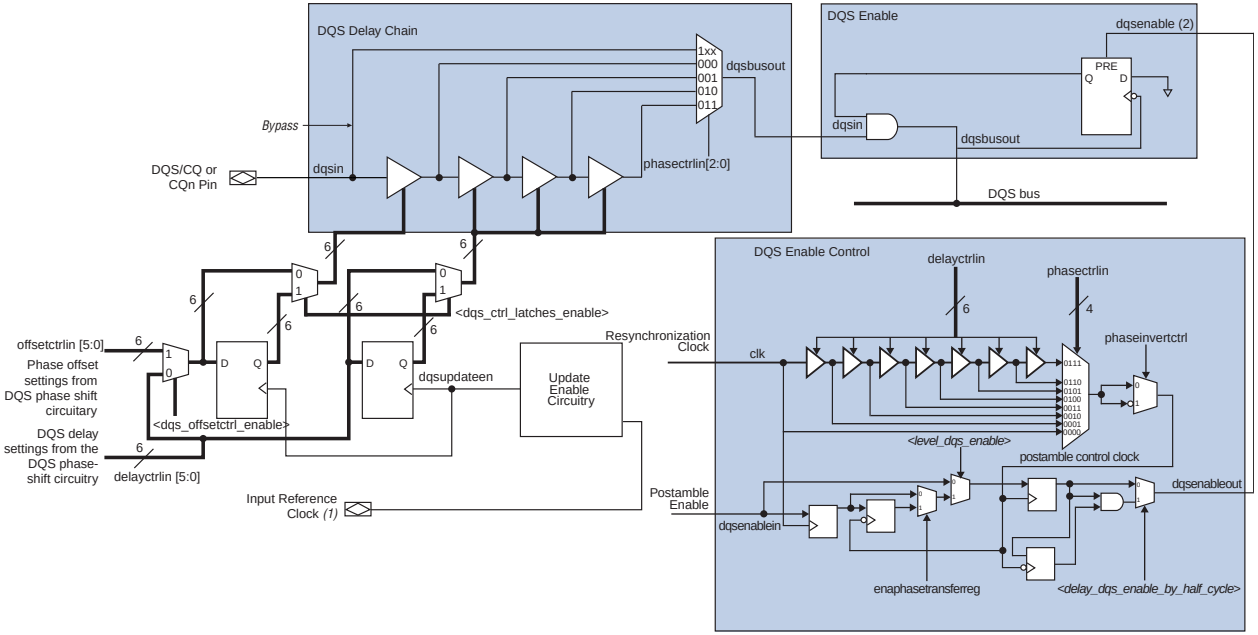
For more information about the value for each step, refer to the *DC and Switching Characteristics* chapter.

When using static phase offset, you can specify the phase offset amount in the ALTMEMPHY megafunction as a positive number for addition or a negative number for subtraction. You can also have a dynamic phase offset that is always added to, subtracted from, or both added to and subtracted from the DLL phase shift. When you always add or subtract, you can dynamically input the phase offset amount into the `dll_offset[5..0]` port. When you want to both add and subtract dynamically, you control the `addnsub` signal in addition to the `dll_offset[5..0]` signals.

DQS Logic Block

Each DQS/CQ and CQn pin is connected to a separate DQS logic block, which consists of the DQS delay chains, update enable circuitry, and DQS postamble circuitry (Figure 7–22).

Figure 7–22. Stratix IV DQS Logic Block



Notes to Figure 7–22:

(1) The input reference clock for the DQS phase-shift circuitry can come from a PLL output clock or an input clock pin. For more information, refer to Table 7–6 through Table 7–17.

(2) The `dqsenable` signal can also come from the Stratix IV FPGA fabric.

DQS Delay Chain

DQS delay chains consist of a set of variable delay elements to allow the input DQS/CQ and CQn signals to be shifted by the amount specified by the DQS phase-shift circuitry or the logic array. There are four delay elements in the DQS delay chain; the first delay chain closest to the DQS/CQ pin can either be shifted by the DQS delay settings or by the sum of the DQS delay setting and the phase-offset setting. The number of delay chains required is transparent because the ALTMEMPHY megafunction automatically sets it when you choose the operating frequency. The DQS delay settings can come from the DQS phase-shift circuitry on either end of the I/O banks or from the logic array.

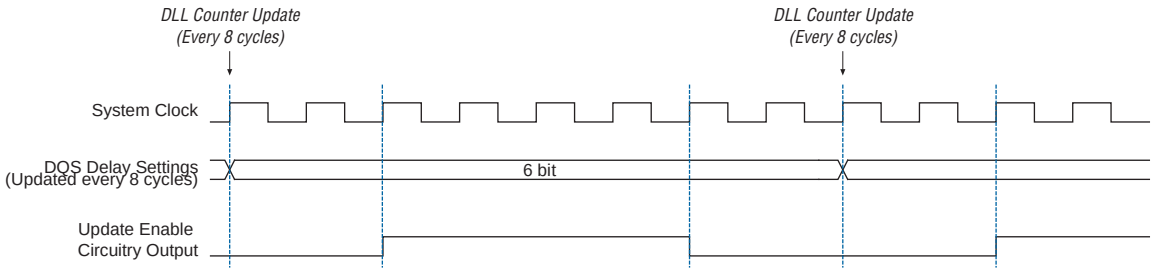
The delay elements in the DQS logic block have the same characteristics as the delay elements in the DLL. When the DLL is not used to control the DQS delay chains, you can input your own Gray-coded 6-bit or 5-bit settings using the `dqs_delayctrlin[5..0]` signals available in the ALTMEMPHY megafunction. These settings control 1, 2, 3, or all 4 delay elements in the DQS delay chains. The ALTMEMPHY megafunction can also dynamically choose the number of DQS delay chains needed for the system. The amount of delay is equal to the sum of the delay element's intrinsic delay and the product of the number of delay steps and the value of the delay steps.

You can also bypass the DQS delay chain to achieve 0° phase shift.

Update Enable Circuitry

Both the DQS delay settings and the phase-offset settings pass through a register before going into the DQS delay chains. The registers are controlled by the update enable circuitry to allow enough time for any changes in the DQS delay setting bits to arrive at all the delay elements. This allows them to be adjusted at the same time. The update enable circuitry enables the registers to allow enough time for the DQS delay settings to travel from the DQS phase-shift circuitry or core logic to all the DQS logic blocks before the next change. It uses the input reference clock or a user clock from the core to generate the update enable output. The ALTMEMPHY megafunction uses this circuit by default. Figure 7–23 shows an example waveform of the update enable circuitry output.

Figure 7–23. DQS Update Enable Waveform



DQS Postamble Circuitry

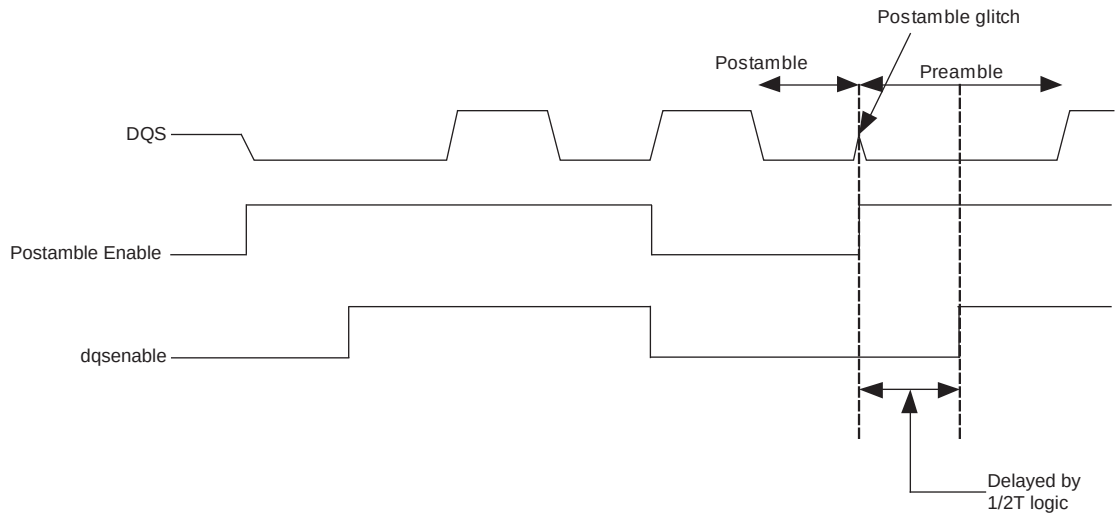
For external memory interfaces that use a bidirectional read strobe such as in DDR3, DDR2, and DDR SDRAM, the DQS signal is low before going to or coming from a high-impedance state. The state in which DQS is low, just after a high-impedance state, is called the preamble; the state in which DQS is low, just before it returns to a high-impedance state, is called the postamble. There are preamble and postamble specifications for both read and write operations in DDR3, DDR2, and DDR SDRAM. The DQS postamble circuitry ensures that data is not lost if there is noise on the DQS line during the end of a read operation that occurs while the DQS is in a postamble state.

Stratix IV devices have dedicated postamble registers that you can control to ground the shifted DQS signal used to clock the DQ input registers at the end of a read operation. This ensures that any glitches on the DQS input signals during the end of a read operation that occurs while the DQS is in a postamble state do not affect the DQ IOE registers.

In addition to the dedicated postamble register, Stratix IV devices also have an HDR block inside the postamble enable circuitry. Use these registers if the controller is running at half the frequency of the I/Os.

Using the HDR block as the first stage capture register in the postamble enable circuitry block is optional. The HDR block is clocked by the half-rate resynchronization clock, which is the output of the I/O clock divider circuit (shown in Figure 7–29 on page 7–47). There is an AND gate after the postamble register outputs that is used to avoid postamble glitches from a previous read burst on a non-consecutive read burst. This scheme allows a half-a-clock cycle latency for `dqsenable` assertion and zero latency for `dqsenable` de-assertion, as shown in Figure 7–24.

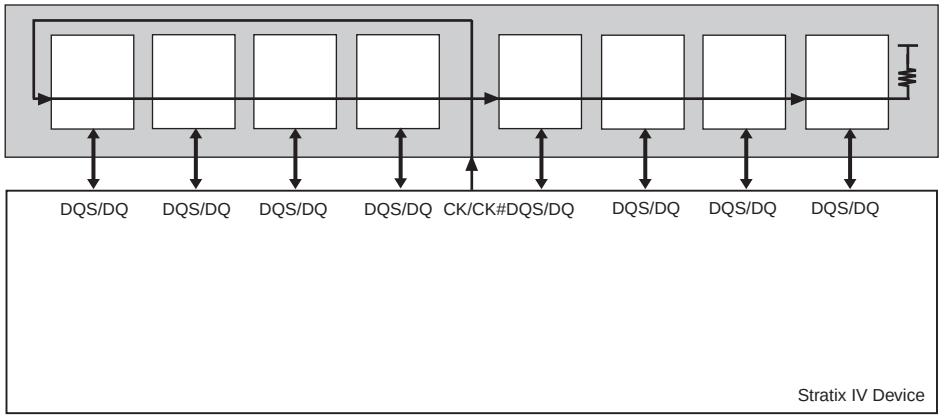
Figure 7–24. Avoiding Glitch on a Non-Consecutive Read Burst Waveform



Leveling Circuitry

DDR3 SDRAM unbuffered modules use a fly-by clock distribution topology for better signal integrity. This means that the CK/CK# signals arrive at each DDR3 SDRAM device in the module at different times. The difference in arrival time between the first DDR3 SDRAM device and the last device on the module can be as long as 1.6 ns. Figure 7–25 shows the clock topology in DDR3 SDRAM unbuffered modules.

Figure 7–25. DDR3 SDRAM Unbuffered Module Clock Topology



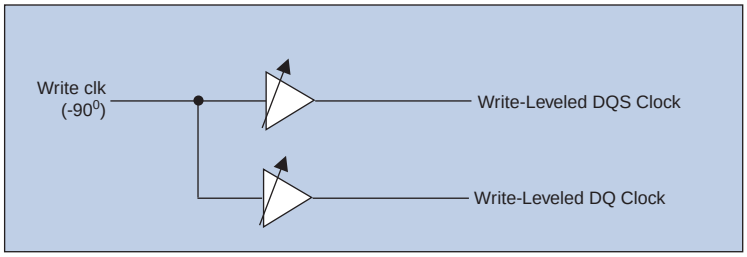
Because the data and read strobe signals are still point-to-point, take special care to ensure that the timing relationship between the CK/CK# and DQS signals (`tDQSS`, `tDSS`, and `tDSH`) during a write is met at every device on the modules. Furthermore, read data coming back into the FPGA from the memory is also staggered in a similar way.

Stratix IV FPGAs have leveling circuitry to address these two situations. There is one leveling circuitry per I/O sub-bank (for example, I/O sub-bank 1A, 1B, and 1C each has one leveling circuitry). These delay chains are PVT-compensated by the same DQS delay settings as the DLL and DQS delay chains.

For frequencies equal to and above 400 MHz, the DLL uses eight delay chains, such that each delay chain generates a 45° delay. The generated clock phases are distributed to every DQS logic block that is available in the I/O sub-bank. The delay chain taps then feeds a multiplexer controlled by the ALTMEMPHY megafunction to select which clock phases are to be used for that ×4 or × 8 DQS group. Each group can use a different tap output from the read-leveling and write-leveling delay chains to compensate for the different CK/CK# delay going into each device on the module.

Figure 7–26 and Figure 7–27 show the Stratix IV write- and read-leveling circuitry.

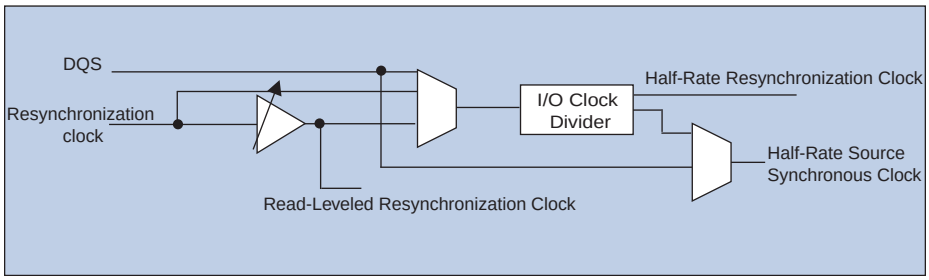
Figure 7–26. Stratix IV Write-Leveling Delay Chains and Multiplexers *(Note 1)*



Note to Figure 7–26:

(1) There is one leveling delay chain per I/O sub-bank (for example, I/O sub-banks 1A, 1B, and 1C). You can only have one memory interface in each I/O sub-bank when you use the leveling delay chain.

Figure 7–27. Stratix IV Read-Leveling Delay Chains and Multiplexers *(Note 1)*



Note to Figure 7–27:

(1) There is one leveling delay chain per I/O sub-bank (for example, I/O sub-banks 1A, 1B, and 1C). You can only have one memory interface in each I/O sub-bank when you use the leveling delay chain.

The –90° write clock of the ALTMEMPHY megafunction feeds the write-leveling circuitry to produce the clock to generate the DQS and DQ signals. During initialization, the ALTMEMPHY megafunction picks the correct write-leveled clock for the DQS and DQ clocks for each DQS/DQ group after sweeping all the available clocks in the write calibration process. The DQ clock output is –90° phase-shifted compared to the DQS clock output.

Similarly, the resynchronization clock feeds the read-leveling circuitry to produce the optimal resynchronization and postamble clock for each DQS/DQ group in the calibration process. The resynchronization and postamble clocks can use different clock outputs from the leveling circuitry. The output from the read-leveling circuitry can also generate the half-rate resynchronization clock that goes to the FPGA fabric.

The ALTMEMPHY megafunction dynamically calibrates the alignment for read and write-leveling during the initialization process.

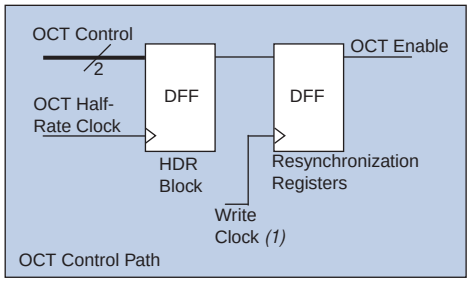
For more information about the ALTMEMPHY megafunction, refer to the *External Memory PHY Interface Megafunction User Guide (ALTMEMPHY)*.

Dynamic On-Chip Termination Control

Figure 7–28 shows the dynamic OCT control block. The block includes all the registers needed to dynamically turn on OCT RT during a read and turn OCT RT off during a write.

For more information about dynamic on-chip termination control, refer to the *I/O Features in Stratix IV Devices* chapter.

Figure 7–28. Stratix IV Dynamic OCT Control Block



Note to Figure 7–28:

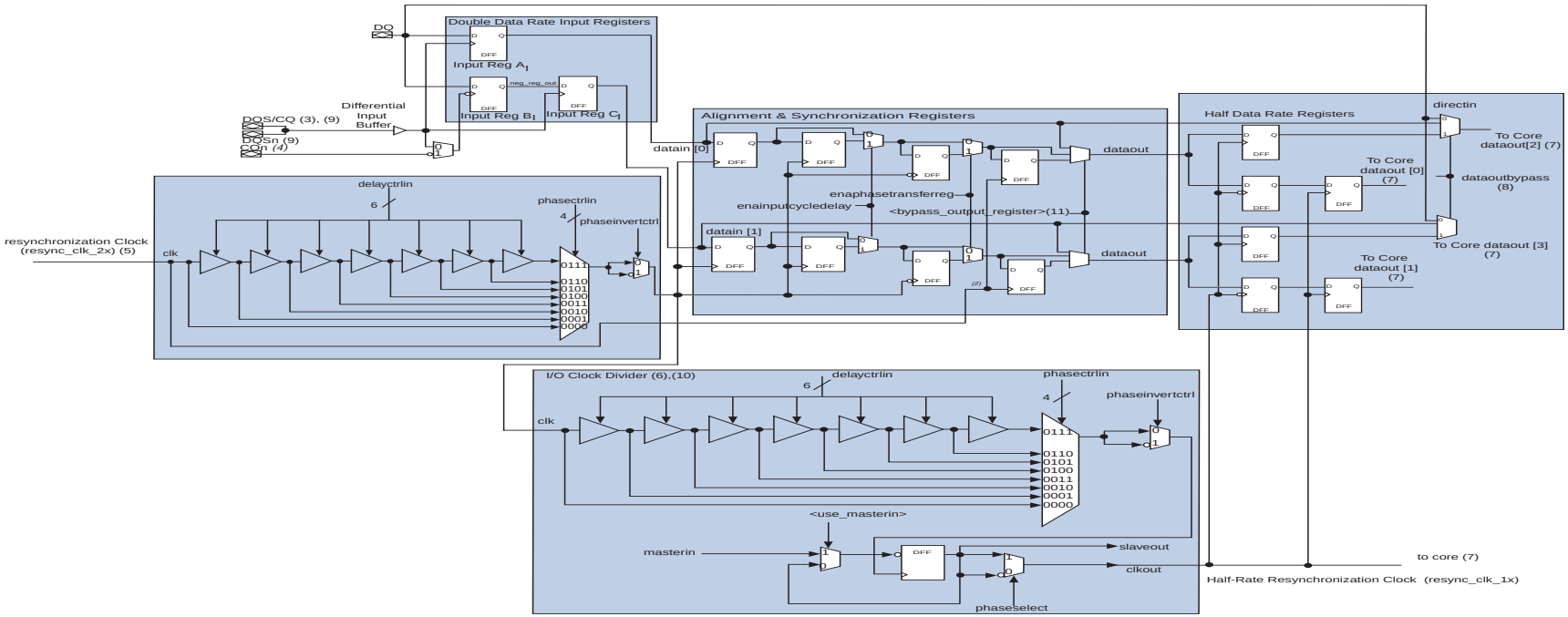
(1) The write clock comes from either the PLL or the write-leveling delay chain.

I/O Element Registers

The IOE registers are expanded to allow source-synchronous systems to have faster register-to-register transfers and resynchronization. Both top and bottom and left and right IOEs have the same capability. Left and right IOEs have extra features to support LVDS data transfer.

Figure 7–29 shows the registers available in the Stratix IV input path. The input path consists of the DDR input registers, resynchronization registers, and HDR block. You can bypass each block of the input path.

Figure 7–29. Stratix IV IOE Input Registers *(Note 1)*



Notes to Figure 7–29:

(1) You can bypass each register block in this path.

(2) This is the 0-phase resynchronization clock (from the read-leveling delay chain).

(3) The input clock can be from the DQS logic block (whether the postamble circuitry is bypassed or not) or from a global clock line.

(4) This input clock comes from the CQn logic block.

(5) This resynchronization clock can come either from the PLL or from the read-leveling delay chain.

(6) The I/O clock divider resides adjacent to the DQS logic block. In addition to the PLL and read-leveled resync clock, the I/O clock divider can also be fed by the DQS bus or CQn bus.

(7) The half-rate data and clock signals feed into a dual-port RAM in the FPGA core.

(8) You can dynamically change the `dataoutbypass` signal after configuration to select either the `directin` input or the output from the half data rate register to feed `dataout`.

(9) The DQS and DQSn signals must be inverted for DDR, DDR2, and DDR3 interfaces. When using Altera's memory interface IPs, the DQS and DQSn signals are automatically inverted.

(10) Each divider feeds up to six pins (from a × 4 DQS group) in the device. To feed wider DQS groups, you must chain multiple clock dividers together by feeding the `slaveout` output of one divider to the `masterin` input of the neighboring pins' divider.

(11) The **bypass_output_register** option allows you to select either the output from the second mux or the output of the fourth alignment/ synchronization register to feed `dataout`.

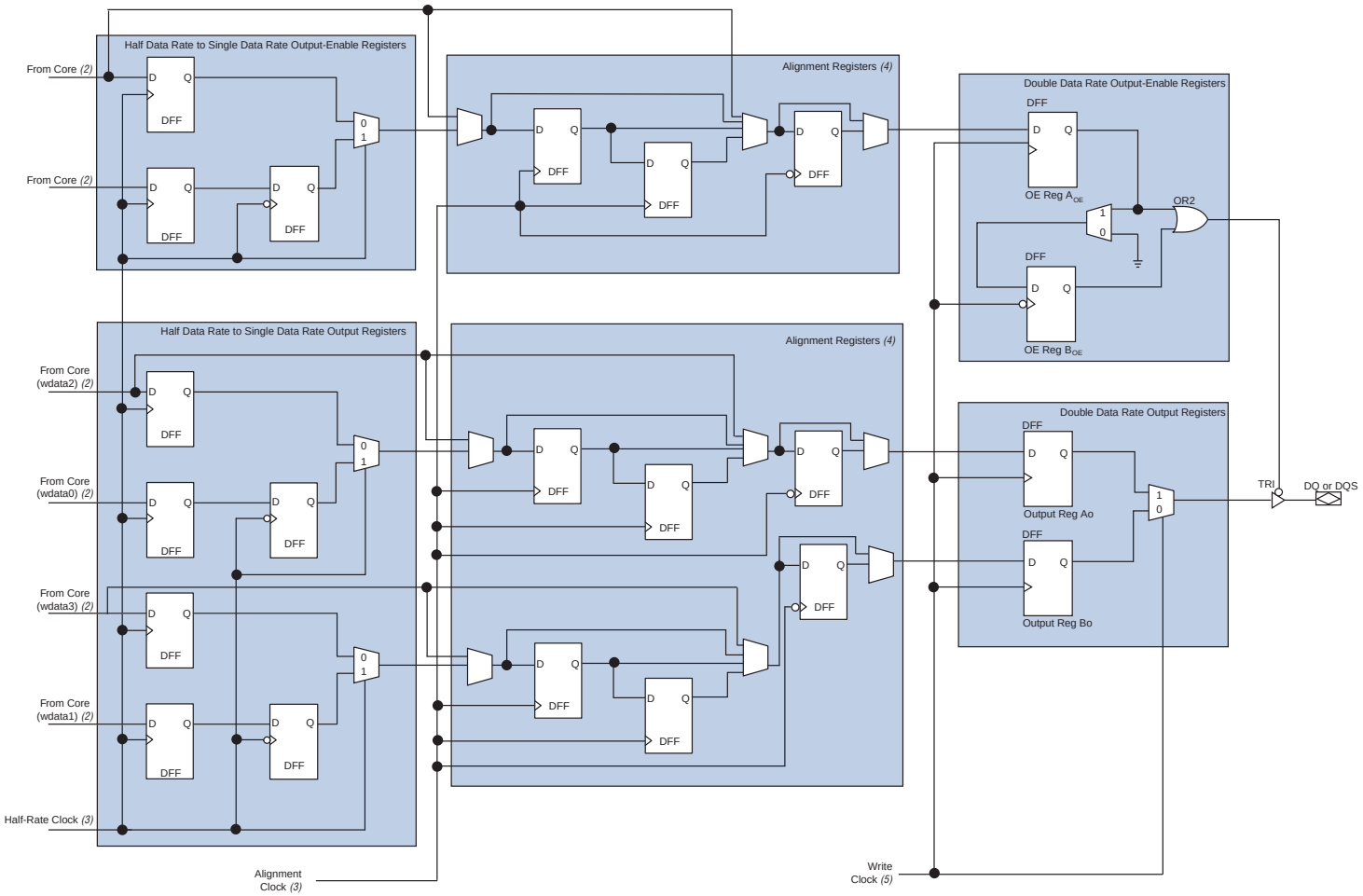
There are three registers in the DDR input registers block. Two registers capture data on the positive and negative edges of the clock, while the third register aligns the captured data. You can choose to use the same clock for the positive edge and negative edge registers, or two complementary clocks (DQS/CQ for the positive-edge register and DQSn/CQn for the negative-edge register). The third register that aligns the captured data uses the same clock as the positive edge registers.

The resynchronization registers consist of up to three levels of registers to resynchronize the data to the system clock domain. These registers are clocked by the resynchronization clock that is either generated by the PLL or the read-leveling delay chain. The outputs of the resynchronization registers can go straight to the core or to the HDR blocks, which are clocked by the divided-down resynchronization clock.

For more information about the read-leveling delay chain, refer to "Leveling Circuitry" on page 7–44.

Figure 7–30 shows the registers available in the Stratix IV output and output-enable paths. The path is divided into the HDR block, resynchronization registers, and output and output-enable registers. The device can bypass each block of the output and output-enable path.

Figure 7–30. Stratix IV IOE Output and Output-Enable Path Registers *(Note 1)*



Notes to Figure 7–30:

(1) You can bypass each register block of the output and output-enable paths.

(2) Data coming from the FPGA core are at half the frequency of the memory interface clock frequency in half-rate mode.

(3) The half-rate clock comes from the PLL, while the alignment clock comes from the write-leveling delay chains.

(4) These registers are only used in DDR3 SDRAM interfaces for write-leveling purposes.

(5) The write clock can come from either the PLL or from the write-leveling delay chain. The DQ write clock and DQS write clock have a 90° offset between them.

The output path is designed to route combinatorial or registered SDR outputs and full-rate or half-rate DDR outputs from the FPGA core. Half-rate data is converted to full-rate using the HDR block, clocked by the half-rate clock from the PLL. The resynchronization registers are also clocked by the same 0° system clock, except in the DDR3 SDRAM interface. In DDR3 SDRAM interfaces, the leveling registers are clocked by the write-leveling clock.

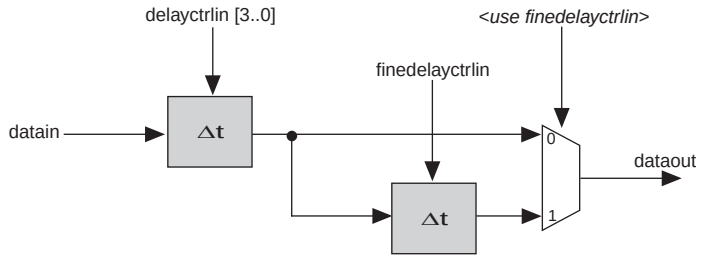
For more information about the write-leveling delay chain, refer to "Leveling Circuitry" on page 7–44.

The output-enable path has a structure similar to the output path. You can have a combinatorial or registered output in SDR applications and you can use half-rate or full-rate operation in DDR applications. Also, the ouput-enable path's resynchronization registers have a structure similar to the output path registers, ensuring that the output-enable path goes through the same delay and latency as the output path.

Delay Chain

Stratix IV devices have run-time adjustable delay chains in the I/O blocks and the DQS logic blocks. You can control the delay chain setting through the I/O or the DQS configuration block output. Figure 7–31 shows the delay chain ports.

Figure 7–31. Delay Chain

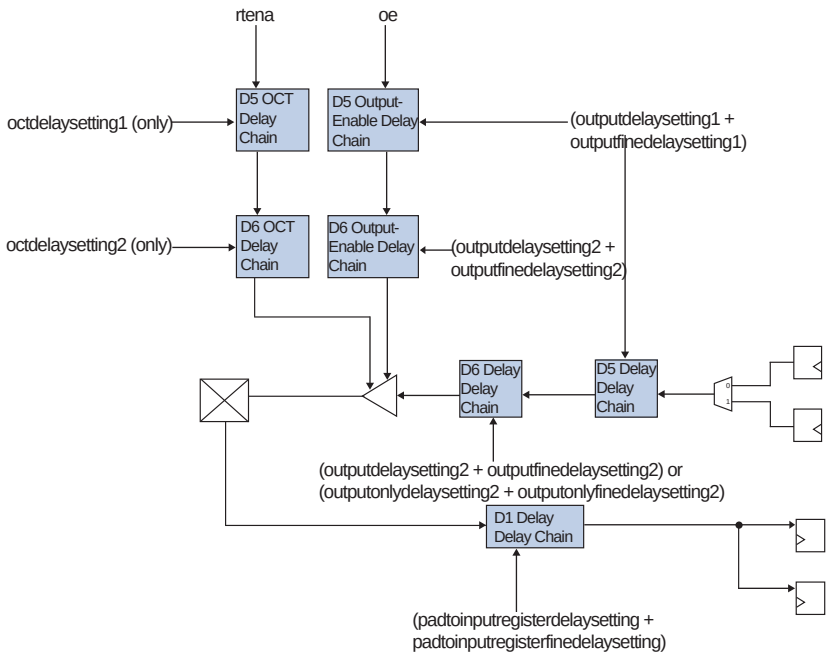


Every I/O block contains the following:

- Two delay chains in series between the output registers and output buffer
- One delay chain between the input buffer and input register
- Two delay chains between the output enable and output buffer
- Two delay chains between the OCT R_T enable control register and output buffer

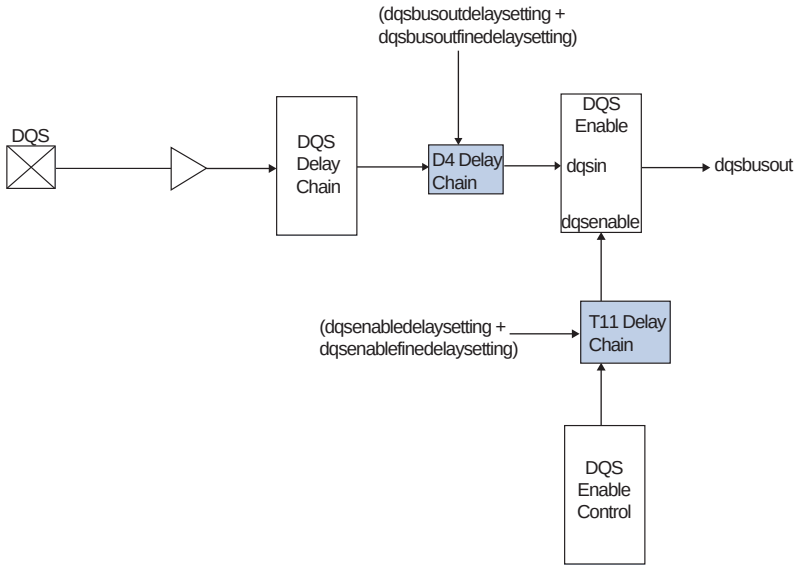
Figure 7–32 shows the delay chains in an I/O block.

Figure 7–32. Delay Chains in an I/O Block



Each DQS logic block contains a delay chain after the `dqsbusout` output and another delay chain before the `dqsenable` input. Figure 7–33 shows the delay chains in the DQS input path.

Figure 7–33. Delay Chains in the DQS Input Path



I/O Configuration Block and DQS Configuration Block

The I/O configuration block and the DQS configuration block are shift registers that you can use to dynamically change the settings of various device configuration bits. The shift registers power-up low. Every I/O pin contains one I/O configuration register, while every DQS pin contains one DQS configuration block in addition to the I/O configuration register. Figure 7–34 shows the I/O configuration block and the DQS configuration block circuitry.

Figure 7–34. I/O Configuration Block and DQS Configuration Block

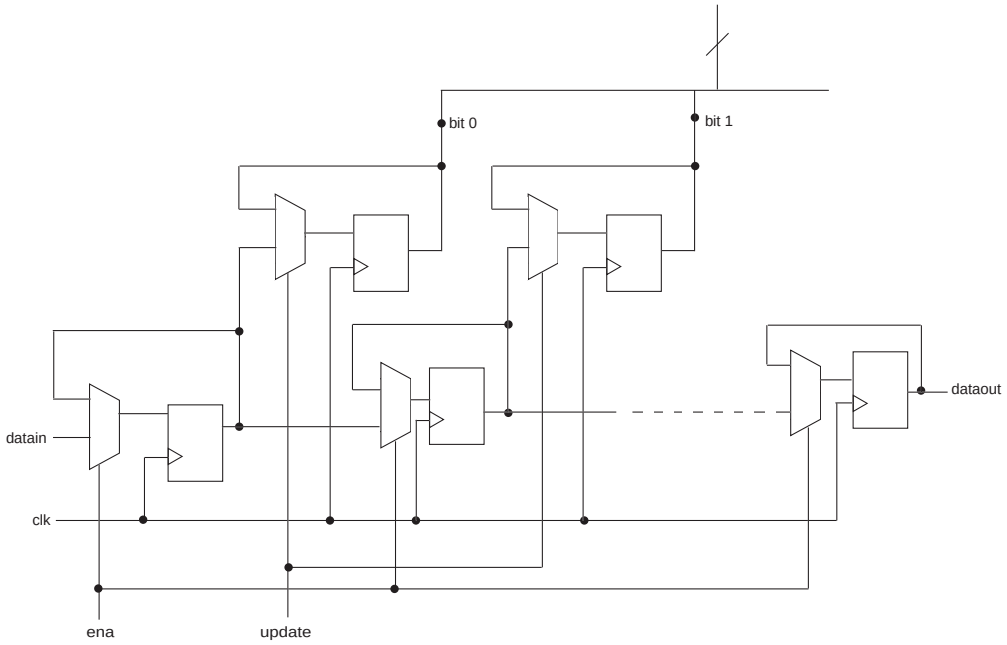


Table 7–19 lists the I/O configuration block bit sequence.

Table 7–19. I/O Configuration Block Bit Sequence

Bit	Bit Name
0..3	outputdelaysetting1[0..3]
4..6	outputdelaysetting2[0..2]
7..10	padtoinputregisterdelaysetting[0..3]
11	outputfinedelaysetting1
12	outputfinedelaysetting2
13	padtoinputregisterfinedelaysetting
14	outputonlyfinedelaysetting2
15..17	outputonlydelaysetting2[2..0]
18	dutycyclecorrectionmode
19..22	dutycyclecorrectionsetting[3..0]

Table 7–20 lists the DQS configuration block bit sequence.

Table 7–20. DQS Configuration Block Bit Sequence

Bit	Bit Name
0..3	dqsbusoutdelaysetting[0..3]
4..6	dqsinputphasesetting[0..2]
7..10	dqsenablectrlphasesetting[0..3]
11..14	dqsoutputphasesetting[0..3]
15..18	dqoutputphasesetting[0..3]
19..22	resyncinputphasesetting[0..3]
23	dividerphasesetting
24	enaoctcycledelaysetting
25	enainputcycledelaysetting
26	enaoutputcycledelaysetting
27..29	dqsenabledelaysetting[0..2]
30..33	octdelaysetting1[0..3]
34..36	octdelaysetting2[0..2]
37	enadataoutbypass
38	enadqsenablephasetransferreg
39	enaoctphasetransferreg
40	enaoutputphasetransferreg
41	enainputphasetransferreg
42	resyncinputphaseinvert
43	dqsenablectrlphaseinvert
44	dqoutputphaseinvert
45	dqsoutputphaseinvert
46	dqsbusoutfinedelaysetting
47	dqsenablefinedelaysetting

Document Revision History

Table 7–21 shows the revision history for this chapter.

Table 7–21. Document Revision History (Part 1 of 2)

Date and Document Version	Changes Made	Summary of Changes
November 2009 v3.0	■ Updated the "Memory Interfaces Pin Support" and "Combining ×16/×18 DQS/DQ Groups for a ×36 QDRII+/QDRII SRAM Interface" sections. ■ Updated Table 7–1, Table 7–2, Table 7–7, and Table 7–12. ■ Updated Figure 7–3, Figure 7–4, Figure 7–5, Figure 7–6, Figure 7–7, Figure 7–8, Figure 7–9, Figure 7–10, Figure 7–11, Figure 7–13, Figure 7–14, Figure 7–15, and Figure 7–16. ■ Added Figure 7–12 and Figure 7–17. ■ Added Table 7–14, Table 7–17, Table 7–19, and Table 7–20. ■ Added "Delay Chain" and "I/O Configuration Block and DQS Configuration Block" sections. ■ Removed Figure 7-8 and Figure 7-12. ■ Removed Table 7-1, Table 7-2, and Table 7-24. ■ Minor text edits.	—
June 2009 v2.3	■ Updated "Overview" and "Leveling Circuitry". ■ Updated Figure 7–26 and Figure 7–27. ■ Updated Table 7–3. ■ Added introductory sentences to improve search ability. ■ Removed the Conclusion section.	—
April 2009 v2.2	■ Updated Table 7–5, Table 7–6, Table 7–15, and Table 7–17 ■ Removed Figure 7-12, Figure 7-13, and Figure 7-20	—
March 2009 v2.1	■ Updated Table 7–1, Table 7–5, Table 7–8, Table 7–12, Table 7–13, Table 7–14, Table 7–15, and Table 7–17. ■ Replaced Table 7–6. ■ Added Table 7–11 and Table 7–16. ■ Updated Figure 7–3, Figure 7–6, Figure 7–8, Figure 7–9, and Figure 7–11. ■ Added Figure 7–7, Figure 7–11, Figure 7–12, Figure 7–13, and Figure 7–20. ■ Updated "Combining ×16/×18 DQS/DQ Groups for ×36 QDRII+/QDRII SRAM Interface" on page 7–26. ■ Updated "Rules to Combine Groups" on page 7–27. ■ Removed "Referenced Documents" section.	—

Table 7–21. Document Revision History (Part 2 of 2)

Date and Document Version	Changes Made	Summary of Changes
November 2008 v2.0	■ Updated Table 7–1, Table 7–2, Table 7–3, Table 7–4, Table 7–5, and Table 7–6. ■ Added Table 7–7. ■ Updated Figure 7–1. ■ Updated “Combining ×16/×18 DQS/DQ groups for ×36 QDRII+/QDRII SRAM Interface” on page 7–26. ■ Updated “Rules to Combine Groups” on page 7–27. ■ Updated “DQS Phase-Shift Circuitry” on page 7–29. ■ Updated Figure 7–19. ■ Updated Table 7–9, Table 7–10, Table 7–11, Table 7–13, Table 7–13, Table 7–14, Table 7–15, Table 7–15, Table 7–16, and Table 7–18. ■ Updated Figure 7–30. ■ Updated Figure 7–31. ■ Made minor editorial changes.	—
May 2008 v1.0	Initial release.	—

8. High-Speed Differential I/O Interfaces and DPA in Stratix IV Devices

SIV51008-3.0

This chapter describes the significant advantages of the high-speed differential I/O interfaces and the dynamic phase aligner (DPA) over single-ended I/Os and their contribution to the overall system bandwidth achievable with Stratix® IV FPGAs.

The Stratix IV device family consists of the Stratix IV E (Enhanced) devices without high-speed clock data recovery (CDR) based transceivers, Stratix IV GT devices with up to 48 CDR-based transceivers running up to 11.3 Gbps, and Stratix IV GX devices with up to 48 CDR-based transceivers running up to 8.5 Gbps.

The following sections describe the Stratix IV high-speed differential I/O interfaces and DPA in detail:

- "Locations of the I/O Banks" on page 8–3
- "LVDS Channels" on page 8–4
- "LVDS SERDES" on page 8–8
- "ALTLVDS Port List" on page 8–9
- "Differential Transmitter" on page 8–11
- "Differential Receiver" on page 8–16
- "LVDS Interface with the Use External PLL Option Enabled" on page 8–25
- "Left and Right PLLs (PLL_Lx and PLL_Rx)" on page 8–28
- "Stratix IV Clocking" on page 8–29
- "Source-Synchronous Timing Budget" on page 8–30
- "Differential Pin Placement Guidelines" on page 8–37

Overview

All Stratix IV E, GX, and GT devices have built-in serializer/deserializer (SERDES) circuitry that supports high-speed LVDS interfaces at data rates of up to 1.6 Gbps. SERDES circuitry is configurable to support source-synchronous communication protocols such as Utopia, Rapid I/O, XSBI, small form factor interface (SFI), serial peripheral interface (SPI), and asynchronous protocols such as SGMII and Gigabit Ethernet.

All references to Stratix IV devices in this chapter apply to Stratix IV E, GT, and GX devices.

The Stratix IV device family has the following dedicated circuitry for high-speed differential I/O support:

- Differential I/O buffer
- Transmitter serializer
- Receiver deserializer

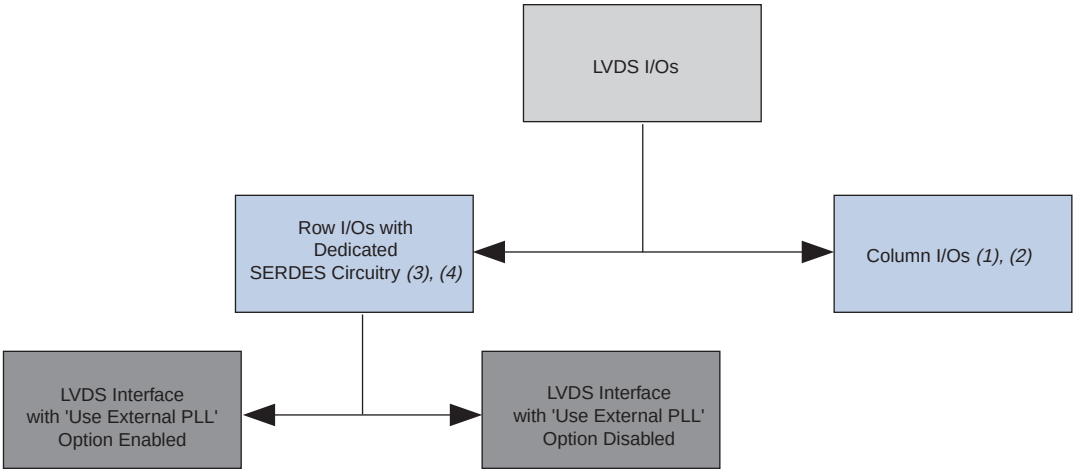
- Data realignment
- DPA
- Synchronizer (FIFO buffer)
- Phase-locked loops (PLLs) (located on left and right sides of the device)

For high-speed differential interfaces, the Stratix IV device family supports the following differential I/O standards:

- LVDS
- Mini-LVDS
- Reduced swing differential signaling (RSDS)

In the Stratix IV device family, I/Os are divided into row and column I/Os. Figure 8–1 shows I/O bank support for the Stratix IV device family. The row I/Os provide dedicated SERDES circuitry.

Figure 8–1. I/O Bank Support in the Stratix IV Device Family *(Note 1), (2), (3), (4)*



Notes to Figure 8–1:

(1) Column input buffers are true LVDS buffers, but do not support 100-Ω differential on-chip termination.

(2) Column output buffers are single ended and need external termination schemes to support LVDS, mini-LVDS, and RSDS standards. For more information, refer to the *I/O Features in Stratix IV Devices* chapter.

(3) Row input buffers are true LVDS buffers and support 100-Ω differential on-chip termination.

(4) Row output buffers are true LVDS buffers.

The ALTLVDS transmitter and receiver requires various clock and load enable signals from a left or right PLL. The Quartus® II software provides the following two choices when configuring the LVDS SERDES circuitry when using the PLL:

- LVDS interface with the **Use External PLL** option enabled—You control the PLL settings, such as dynamically reconfiguring the PLL to support different data rates, dynamic phase shift, and so on. You must enable the **Use External PLL** option in the ALTLVDS megafunction, using the ALTLVDS MegaWizard™ Plug-in Manager software. You also must instantiate an ALTPLL megafunction to generate the various clocks and load enable signals. For more information, refer to "LVDS Interface with the Use External PLL Option Enabled" on page 8–25.

- LVDS interface with the **Use External PLL** option disabled—The Quartus II software configures the PLL settings automatically. The software is also responsible for generating the various clock and load enable signals based on the input reference clock and data rate selected.

☞ Both choices target the same physical PLL; the only difference is the additional flexibility provided when an LVDS interface has the **Use External PLL** option enabled.

Locations of the I/O Banks

Stratix IV I/Os are divided into 16 to 24 I/O banks. The dedicated circuitry that supports high-speed differential I/Os is located in banks in the right and left side of the device. Figure 8–2 shows a high-level chip overview of the Stratix IV E device.

Figure 8–2. High-Speed Differential I/Os with DPA Locations in Stratix IV E Devices

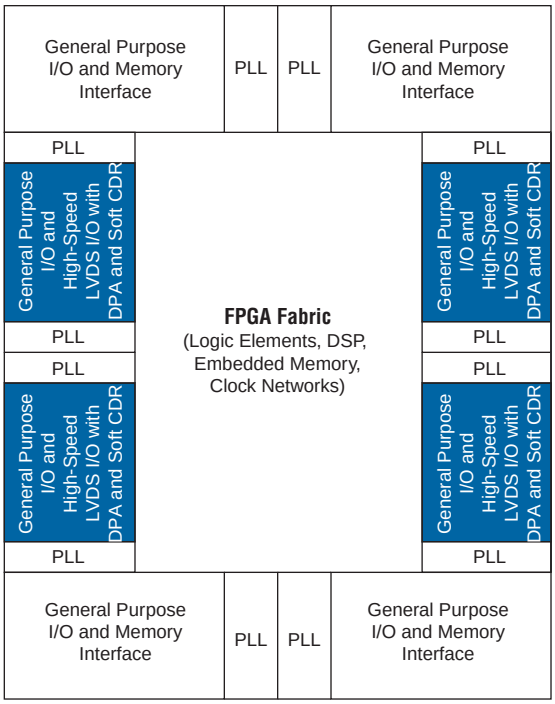
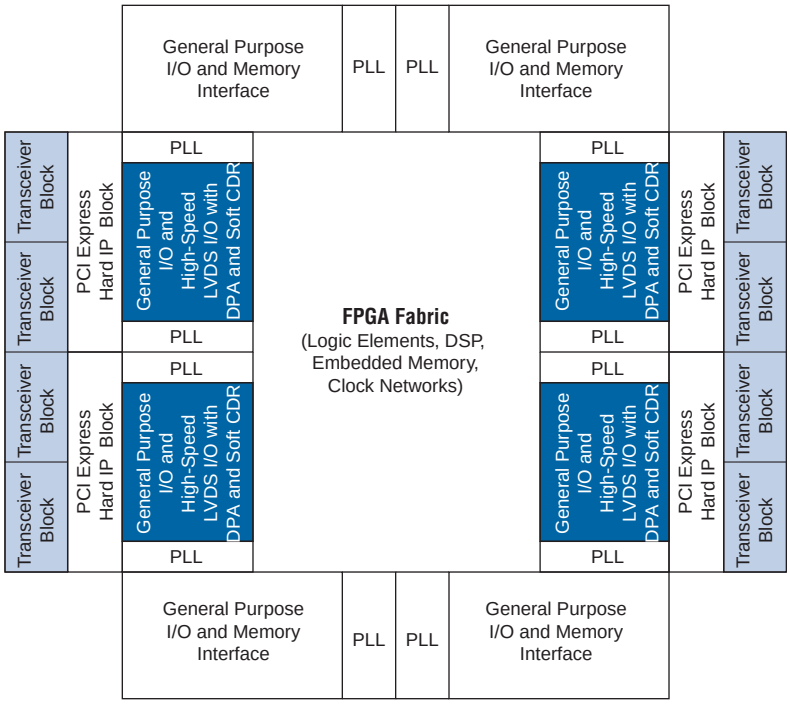


Figure 8–3 shows a high-level chip overview of the Stratix IV GT and GX devices.

Figure 8–3. High-Speed Differential I/Os with DPA Locations in Stratix IV GT and GX Devices



LVDS Channels

The Stratix IV device family supports LVDS on both row and column I/O banks. Row I/Os support true LVDS input with 100-Ω differential input termination (OCT R_D), and true LVDS output buffers. Column I/Os supports true LVDS input buffers without OCT R_D. Alternately, you can configure the row and column LVDS pins as emulated LVDS output buffers that use two single-ended output buffers with an external resistor network to support LVDS, mini-LVDS, and RSDS standards. Dedicated SERDES and DPA circuitries are implemented on the row I/O banks to further enhance LVDS interface performance in the device. For column I/O banks, SERDES is implemented in the core logic because there is no dedicated SERDES circuitry on column I/O banks.

☞ Emulated LVDS output buffers support tri-state capability starting with the Quartus II software version 9.1

Table 8–1 and Table 8–2 list the maximum number of row and column LVDS I/Os supported in Stratix IV E devices. You can design the LVDS I/Os as true LVDS buffers or emulated LVDS buffers, as long as the combination of the two do not exceed the maximum count.

For example, there are a total of 112 LVDS pairs on row I/Os in the 780-pin EP4SE230 device (refer to Table 8–1). You can design up to a maximum of 56 true LVDS input buffers and 56 true LVDS output buffers, or up to a maximum of 112 emulated LVDS output buffers. For the 780-pin EP4SE230 device (refer to Table 8–2), there are a total of 128 LVDS pairs on column I/Os. You can design up to a maximum of 64 true LVDS input buffers and 64 emulated LVDS output buffers, or up to a maximum of 128 emulated LVDS output buffers.

Table 8–1. LVDS Channels Supported in Stratix IV E Device Row I/O Banks *(Note 1)*, *(2)*, *(3)*

Device	780-Pin FineLine BGA	1152-Pin FineLine BGA	1517-Pin FineLine BGA	1760- Pin FineLine BGA
EP4SE230	56 Rx or eTx + 56 Tx or eTx	—	—	—
EP4SE360	56 Rx or eTx + 56 Tx or eTx *(4)*	88 Rx or eTx + 88 Tx or eTx	—	—
EP4SE530	—	88 Rx or eTx + 88 Tx or eTx *(5)*	112 Rx or eTx + 112 Tx or eTx *(6)*	112 Rx or eTx + 112 Tx or eTx
EP4SE820 *(7)*	—	88 Rx or eTx + 88 Tx or eTx	112 Rx or eTx + 112 Tx or eTx	132 Rx or eTx + 132 Tx or eTx

Notes to Table 8–1:

(1) Rx = true LVDS input buffers with OCT R_D, Tx = true LVDS output buffers, eTx = emulated LVDS output buffers (either `LVDS_E_1R` or `LVDS_E_3R`).
(2) The LVDS Rx and Tx channels are equally divided between the left and right sides of the device.
(3) The LVDS channel count does not include dedicated clock input pins.
(4) EP4SE360 devices are offered in the H780 package instead of the F780 package.
(5) EP4SE530 devices are offered in the H1152 package instead of the F1152 package.
(6) EP4SE530 devices are offered in the H1517 package instead of the F1517 package.
(7) The number of LVDS Rx and Tx channels are preliminary.

Table 8–2. LVDS Channels Supported in Stratix IV E Device Column I/O Banks *(Note 1)*, *(2)*, *(3)*

Device	780-Pin FineLine BGA	1152-Pin FineLine BGA	1517-Pin FineLine BGA	1760-Pin FineLine BGA
EP4SE230	64 Rx or eTx + 64 eTx	—	—	—
EP4SE360	64 Rx or eTx + 64 eTx *(4)*	96 Rx or eTx + 96 eTx	—	—
EP4SE530	—	96 Rx or eTx + 96 eTx *(5)*	128 Rx or eTx + 128 eTx *(6)*	128 Rx or eTx + 128 eTx
EP4SE820 *(7)*	—	96 Rx or eTx + 96 eTx	128 Rx or eTx + 128 eTx	144 Rx or eTx + 144 eTx

Notes to Table 8–2:

(1) Rx = true LVDS input buffers without OCT R_D, eTx = emulated LVDS output buffers (either `LVDS_E_1R` or `LVDS_E_3R`).
(2) The LVDS Rx and Tx channels are equally divided between the top and bottom sides of the device.
(3) The LVDS channel count does not include dedicated clock input pins.
(4) EP4SE360 devices are offered in the H780 package instead of the F780 package.
(5) EP4SE530 devices are offered in the H1152 package instead of the F1152 package.
(6) EP4SE530 devices are offered in the H1517 package instead of the F1517 package.
(7) The number of LVDS Rx and Tx channels are preliminary.

Table 8–3 and Table 8–4 list the maximum number of row and column LVDS I/Os supported in Stratix IV GT devices.

Table 8–3. LVDS Channels Supported in Stratix IV GT Device Row I/O Banks *(Note 1)*, *(2)*

Device	1517-pin FineLine BGA	1932-pin FineLine BGA
EP4S40G2	46 Rx or eTx + 73 Tx or eTx	—
EP4S40G5	46 Rx or eTx + 73 Tx or eTx	—
EP4S100G2	46 Rx or eTx + 73 Tx or eTx	—
EP4S100G3	—	47 Rx or eTx + 56 Tx or eTx
EP4S100G4	—	47 Rx or eTx + 56 Tx or eTx
EP4S100G5	46 Rx or eTx + 73 Tx or eTx	47 Rx or eTx + 56 Tx or eTx

Notes to Table 8–3:

(1) Rx = true LVDS input buffers with OCT R_D, eTx = emulated LVDS output buffers (either `LVDS_E_1R` or `LVDS_E_3R`).

(2) The LVDS Rx and Tx channel count does not include dedicated clock input pins.

Table 8–4. LVDS Channels Supported in Stratix IV GT Device Column I/O Banks *(Note 1)*, *(2)*

Device	1517-pin FineLine BGA	1932-pin FineLine BGA
EP4S40G2	96 Rx or eTx + 96 eTx	—
EP4S40G5	96 Rx or eTx + 96 eTx	—
EP4S100G2	96 Rx or eTx + 96 eTx	—
EP4S100G3	—	128 Rx or eTx + 128 eTx
EP4S100G4	—	128 Rx or eTx + 128 eTx
EP4S100G5	96 Rx or eTx + 96 eTx	128 Rx or eTx + 128 eTx

Notes to Table 8–4:

(1) Rx = true LVDS input buffers without OCT R_D, eTx = emulated LVDS output buffers (either `LVDS_E_1R` or `LVDS_E_3R`).

(2) The LVDS Rx and Tx channel count does not include dedicated clock input pins.

Table 8–5 and Table 8–6 list the maximum number of row and column LVDS I/Os supported in Stratix IV GX devices.

Table 8–5. LVDS Channels Supported in Stratix IV GX Device Row I/O Banks *(Note 1)*, *(2)*, *(3)* (Part 1 of 2)

Device	780-Pin FineLine BGA	1152-Pin FineLine BGA	1152-Pin FineLine BGA *(4)*	1517-Pin FineLine BGA	1760-Pin FineLine BGA	1932-Pin FineLine BGA
EP4SGX70	28 Rx or eTx + 28 Tx or eTx	—	56 Rx or eTx + 56 Tx or eTx	—	—	—
EP4SGX110	28 Rx or eTx + 28 Tx or eTx	28 Rx or eTx + 28 Tx or eTx	56 Rx or eTx + 56 Tx or eTx	—	—	—
EP4SGX180	28 Rx or eTx + 28 Tx or eTx	44 Rx or eTx + 44 Tx or eTx	44 Rx or eTx + 44 Tx or eTx	88 Rx or eTx + 88 Tx or eTx	—	—
EP4SGX230	28 Rx or eTx + 28 Tx or eTx	44 Rx or eTx + 44 Tx or eTx	44 Rx or eTx + 44 Tx or eTx	88 Rx or eTx + 88 Tx or eTx	—	—
EP4SGX290	— *(5)*	44 Rx or eTx + 44 Tx or eTx	44 Rx or eTx + 44 Tx or eTx	88 Rx or eTx + 88 Tx or eTx	88 Rx or eTx + 88 Tx or eTx	98 Rx or eTx + 98 Tx or eTx

Table 8–5. LVDS Channels Supported in Stratix IV GX Device Row I/O Banks *(Note 1)*, *(2)*, *(3)* (Part 2 of 2)

Device	780-Pin FineLine BGA	1152-Pin FineLine BGA	1152-Pin FineLine BGA *(4)*	1517-Pin FineLine BGA	1760-Pin FineLine BGA	1932-Pin FineLine BGA
EP4SGX360	— *(5)*	44 Rx or eTx + 44 Tx or eTx	44 Rx or eTx + 44 Tx or eTx	88 Rx or eTx + 88 Tx or eTx	88 Rx or eTx + 88 Tx or eTx	98 Rx or eTx + 98 Tx or eTx
EP4SGX530	—	—	44 Rx or eTx + 44 Tx or eTx *(6)*	88 Rx or eTx + 88 Tx or eTx *(7)*	88 Rx or eTx + 88 Tx or eTx	98 Rx or eTx + 98 Tx or eTx

Notes to Table 8–5:

(1) Rx = true LVDS input buffers with OCT R_D, Tx = true LVDS output buffers, eTx = emulated LVDS output buffers (either `LVDS_E_1R` or `LVDS_E_3R`).

(2) The LVDS Rx and Tx channels are equally divided between the left and right sides of the device, except for the devices in the 780-pin Fineline BGA. These devices have the LVDS Rx and Tx located on the left side of the device.

(3) The LVDS channel count does not include dedicated clock input pins.

(4) This package supports PMA-only transceiver channels.

(5) EP4SGX290 and EP4SGX360 devices are offered in the H780 package instead of the F780 package.

(6) EP4SGX530 devices are offered in the H1152 package instead of the F1152 package.

(7) EP4SGX530 devices are offered in the H1517 package instead of the F1517 package.

Table 8–6. LVDS Channels Supported in Stratix IV GX Device Column I/O Banks *(Note 1)*, *(2)*, *(3)*

Device	780-Pin FineLine BGA	1152-Pin FineLine BGA	1152-Pin FineLine BGA *(4)*	1517-Pin FineLine BGA	1760-Pin FineLine BGA	1932-Pin FineLine BGA
EP4SGX70	64 Rx or eTx + 64 eTx	—	64 Rx or eTx + 64 eTx	—	—	—
EP4SGX110	64 Rx or eTx + 64 eTx	64 Rx or eTx + 64 eTx	64 Rx or eTx + 64 eTx	—	—	—
EP4SGX180	64 Rx or eTx + 64 eTx	96 Rx or eTx + 96 eTx	96 Rx or eTx + 96 eTx	96 Rx or eTx + 96 eTx	—	—
EP4SGX230	64 Rx or eTx + 64 eTx	96 Rx or eTx + 96 eTx	96 Rx or eTx + 96 eTx	96 Rx or eTx + 96 eTx	—	—
EP4SGX290	72 Rx or eTx + 72 eTx *(5)*	96 Rx or eTx + 96 eTx	96 Rx or eTx + 96 eTx	96 Rx or eTx + 96 eTx	128 Rx or eTx + 128 eTx	128 Rx or eTx + 128 eTx *(8)*
EP4SGX360	72 Rx or eTx + 72 eTx *(5)*	96 Rx or eTx + 96 eTx	96 Rx or eTx + 96 eTx	96 Rx or eTx + 96 eTx	128 Rx or eTx + 128 eTx	128 Rx or eTx + 128 eTx *(8)*
EP4SGX530	—	—	96 Rx or eTx + 96 eTx *(6)*	96 Rx or eTx + 96 eTx *(7)*	128 Rx or eTx + 128 eTx	128 Rx or eTx + 128 eTx

Notes to Table 8–6:

(1) Rx = true LVDS input buffers without OCT R_D, eTx = emulated LVDS output buffers (either `LVDS_E_1R` or `LVDS_E_3R`).

(2) The LVDS Rx and Tx channels are equally divided between the left and right sides of the device.

(3) The LVDS channel count does not include dedicated clock input pins.

(4) This package supports PMA-only transceiver channels.

(5) EP4SGX290 and EP4SGX360 devices are offered in the H780 package instead of the F780 package.

(6) EP4SGX530 devices are offered in the H1152 package instead of the F1152 package.

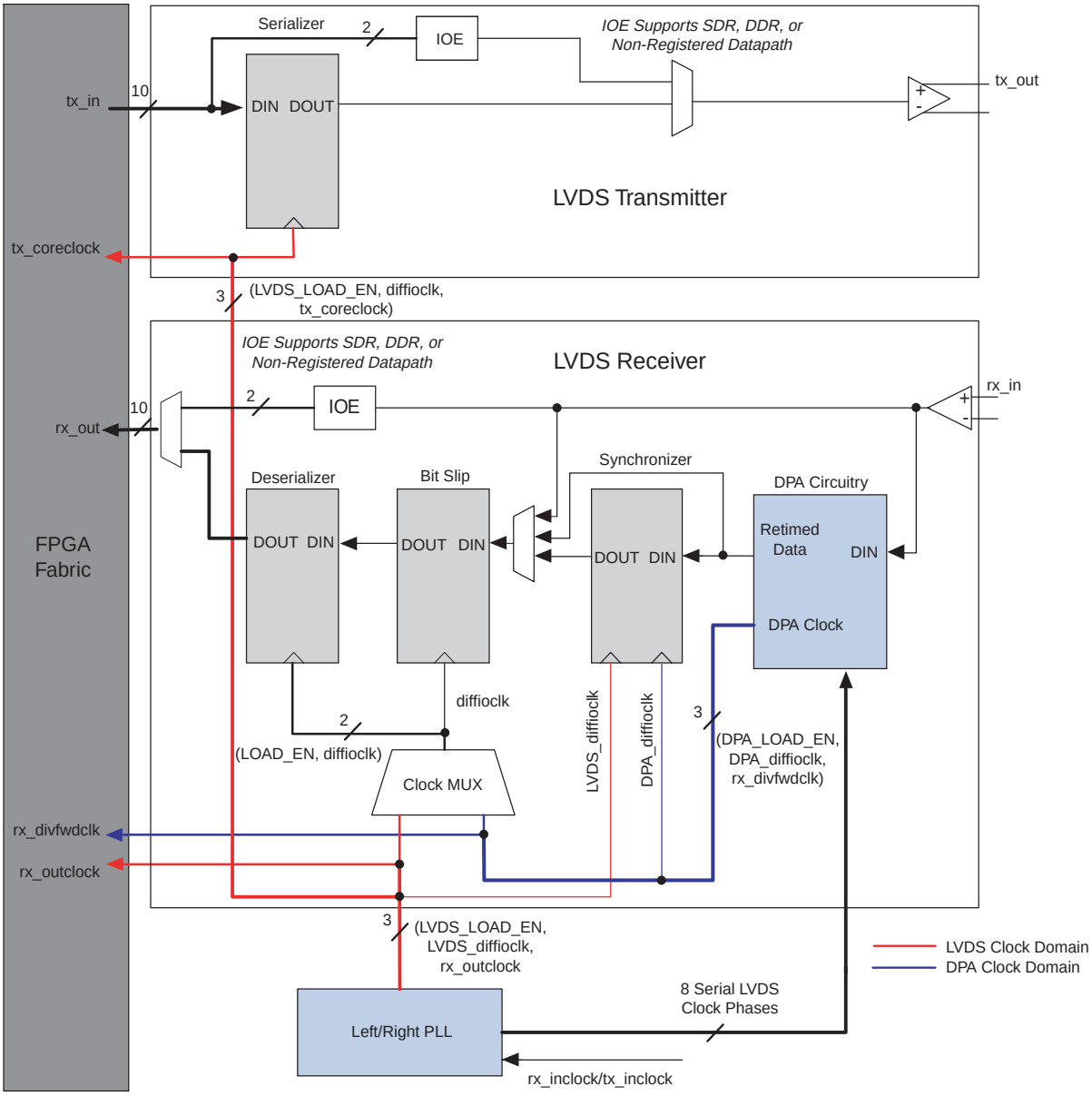
(7) EP4SGX530 devices are offered in the H1517 package instead of the F1517 package.

(8) The Quartus II software version 9.0 does not support EP4SGX290 and EP4SGX360 devices in the 1932-Pin FineLine BGA package. These devices will be supported in a future release of the Quartus II software.

LVDS SERDES

Figure 8–4 shows a transmitter and receiver block diagram for the LVDS SERDES circuitry in the left and right banks. This diagram shows the interface signals of the transmitter and receiver data path. For more information, refer to "Differential Transmitter" on page 8–11 and "Differential Receiver" on page 8–16.

Figure 8–4. LVDS SERDES *(Note 1), (2), (3)*



Notes to Figure 8–4:

(1) This diagram shows a shared PLL between the transmitter and receiver. If the transmitter and receiver are not sharing the same PLL, the two left and right PLLs are required.

(2) In SDR and DDR modes, the data width is 1 and 2 bits, respectively.

(3) The `tx_in` and `rx_out` ports have a maximum data width of 10 bits.

ALTLVDS Port List

Table 8–7 lists the interface signals for a LVDS transmitter and receiver.

Table 8–7. Port List of the LVDS Interface (ALTLVDS) *(Note 1)*, *(2)*, *(3)* (Part 1 of 3)

Port Name	Input / Output	Description
PLL Signals		
`pll_areset`	Input	Asynchronous reset to the LVDS transmitter and receiver PLL. The minimum pulse width requirement for this signal is 10 ns.
LVDS Transmitter Interface Signals		
`tx_in[ ]`	Input	The data bus width per channel is the same as the serialization factor (SF). Input data must be synchronous to the `tx_coreclock` signal.
`tx_inclock`	Input	Reference clock input for the transmitter PLL. The allowed frequency range for the reference clock is between 5 MHz and 625 MHz and must satisfy the following condition: 5 Mbps < ([Input Clock Frequency] × [PLL Multiplication Factor]) < 1600 Mbps. The ALTLVDS MegaWizard Plug-In Manager software automatically selects the appropriate PLL multiplication factor based on the data rate and reference clock frequency selection.
`tx_enable` *(3)*	Input	This port is instantiated only when you select the **Use External PLL** option in the MegaWizard Plug-In Manager software. This input port must be driven by the PLL instantiated though the ALTPLL MegaWizard Plug-In Manager software.
`tx_out`	Output	LVDS transmitter serial data output port. `tx_out` is clocked by a serial clock generated by the left and right PLL.
`tx_outclock`	Output	The frequency of this clock is programmable to be the same as the data rate (up to 717 MHz), half the data rate, or one-fourth the data rate. The phase offset of this clock, with respect to the serial data, is programmable in increments of 45°.
`tx_coreclock` *(3)*	Output	FPGA fabric-transmitter interface clock. The parallel transmitter data generated in the FPGA fabric must be clocked with this clock. This port is not available when you select the **Use External PLL** option in the MegaWizard Plug-In Manager software. The FPGA fabric-transmitter interface clock must be driven by the PLL instantiated through the ALTPLL MegaWizard Plug-In Manager software.
`tx_locked`	Output	When high, this signal indicates that the transmitter PLL is locked to the input reference clock.

Table 8–7. Port List of the LVDS Interface (ALTLVDS) *(Note 1)*, *(2)*, *(3)* (Part 2 of 3)

Port Name	Input / Output	Description
LVDS Receiver Interface Signals		
`rx_in`	Input	LVDS receiver serial data input port.
`rx_inclock`	Input	Reference clock input for the receiver PLL. The allowed frequency range for the reference clock is between 5 MHz and 625 MHz and must satisfy the following condition: 5 Mbps < ([Input Clock Frequency] × [PLL Multiplication Factor]) < 1600 Mbps. The ALTLVDS MegaWizard Plug-In Manager software automatically selects the appropriate PLL multiplication factor based on the data rate and reference clock frequency selection.
`rx_channel_data_align`	Input	Edge-sensitive bit-slip control signal. Each rising edge on this signal causes the data re-alignment circuitry to shift the word boundary by one bit. The minimum pulse width requirement is one parallel clock cycle. There is no maximum pulse width requirement.
`rx_dpll_hold`	Input	When low, the DPA tracks any dynamic phase variations between the clock and data. When high, the DPA holds the last locked phase and does not track any dynamic phase variations between the clock and data. This port is not available in non-DPA mode.
`rx_enable` *(3)*	Input	This port is instantiated only when you select the **Use External PLL** option in the MegaWizard Plug-In Manager software. This input port must be driven by the PLL instantiated though the ALTPLL MegaWizard Plug-In Manager software.
`rx_out[ ]`	Output	Receiver parallel data output. The data bus width per channel is the same as the deserialization factor (DF). The output data is synchronous to the `rx_outclock` signal in non-DPA and DPA modes. It is synchronous to the `rx_divfwdclk` signal in soft-CDR mode.
`rx_outclock`	Output	Parallel output clock from the receiver PLL. The parallel data output from the receiver is synchronous to this clock in non-DPA and DPA modes. This port is not available when you select the **Use External PLL** option in the MegaWizard Plug-In Manager software. The FPGA fabric-receiver interface clock must be driven by the PLL instantiated through the ALTPLL MegaWizard Plug-In Manager software.
`rx_locked`	Output	When high, this signal indicates that the receiver PLL is locked to `rx_inclock`.
`rx_dpa_locked`	Output	This signal only indicates an initial DPA lock condition to the optimum phase after power up or reset. This signal is not de-asserted if the DPA selects a new phase out of the eight clock phases to sample the received data. You must not use the `rx_dpa_locked` signal to determine a DPA loss-of-lock condition.
`rx_cda_max`	Output	Data re-alignment (bit slip) roll-over signal. When high for one parallel clock cycle, this signal indicates that the user-programmed number of bits for the word boundary to roll-over have been slipped.
`rx_divfwdclk`	Output	Parallel DPA clock to the FPGA fabric logic array. The parallel receiver output data to the FPGA fabric logic array is synchronous to this clock in soft-CDR mode. This signal is not available in non-DPA and DPA modes.

Table 8–7. Port List of the LVDS Interface (ALTLVDS) *(Note 1)*, *(2)*, *(3)* (Part 3 of 3)

Port Name	Input / Output	Description
`dpa_pll_recal`	Input	Enable PLL calibration dynamically without resetting the DPA circuitry or the PLL.
`dpa_pll_cal_busy`	Output	`Busy` signal that is asserted high when the PLL calibration occurs.
Reset Signals		
`rx_reset`	Input	Asynchronous reset to the DPA circuitry and FIFO. The minimum pulse width requirement for this reset is one parallel clock cycle. This signal resets DPA and FIFO blocks.
`rx_fifo_reset`	Input	Asynchronous reset to the FIFO between the DPA and the data realignment circuits. The synchronizer block must be reset after a DPA loses lock condition and the data checker shows corrupted received data. The minimum pulse width requirement for this reset is one parallel clock cycle. This signal resets the FIFO block.
`rx_cda_reset`	Input	Asynchronous reset to the data realignment circuitry. The minimum pulse width requirement for this reset is one parallel clock cycle. This signal resets the data realignment block.

Notes to Table 8–7:

(1) Unless stated, signals are valid in all three modes (non-DPA, DPA, and soft-CDR) for a single channel.

(2) All reset and control signals are active high.

(3) For more information, refer to "LVDS Interface with the Use External PLL Option Enabled" on page 8–25.

For more information about the LVDS transmitter and receiver settings using ALTLVDS, refer to the *SERDES Transmitter/Receiver (ALTLVDS) Megafunction User Guide*.

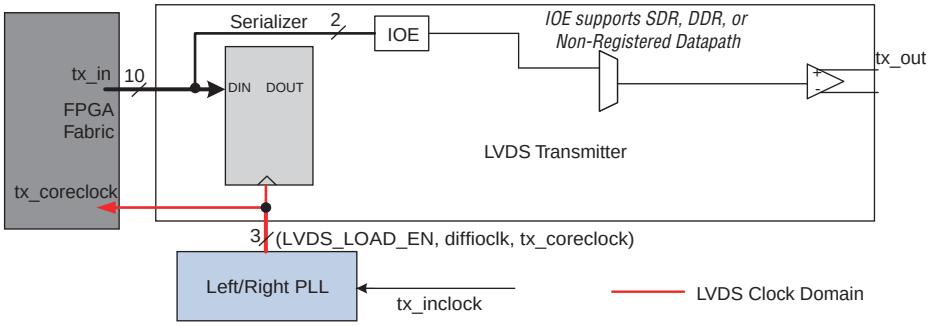
Differential Transmitter

The Stratix IV transmitter has a dedicated circuitry to provide support for LVDS signaling. The dedicated circuitry consists of a differential buffer, a serializer, and left and right PLLs, that can be shared between the transmitter and receiver. The differential buffer can drive out LVDS, mini-LVDS, and RSDS signaling levels. The serializer takes up to 10 bits wide parallel data from the FPGA fabric, clocks it into the load registers, and serializes it using shift registers clocked by the left and right PLL before sending the data to the differential buffer. The MSB of the parallel data is transmitted first.

When using emulated LVDS I/O standards at the differential transmitter, the SERDES circuitry must be implemented in logic cells but not the hard SERDES.

The load enable (`LVDS_LOAD_EN`) signal and the `diffioclk` signal (the clock running at serial data rate) generated from the `PLL_Lx` (left PLL) or `PLL_Rx` (right PLL) clocks the load and shift registers. You can statically set the serialization factor to ×4, ×6, ×7, ×8, or ×10 using the Quartus II software. The load enable signal is derived from the serialization factor setting. Figure 8–5 shows a block diagram of the Stratix IV transmitter.

Figure 8–5. Stratix IV Transmitter *(Note 1), (2)*



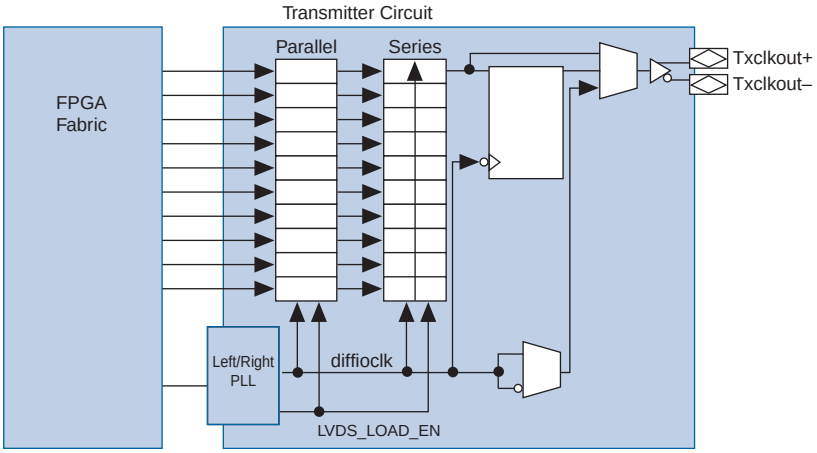
Notes to Figure 8–5:

(1) In SDR and DDR modes, the data width is 1 and 2 bits, respectively.

(2) The `tx_in` port has a maximum data width of 10 bits.

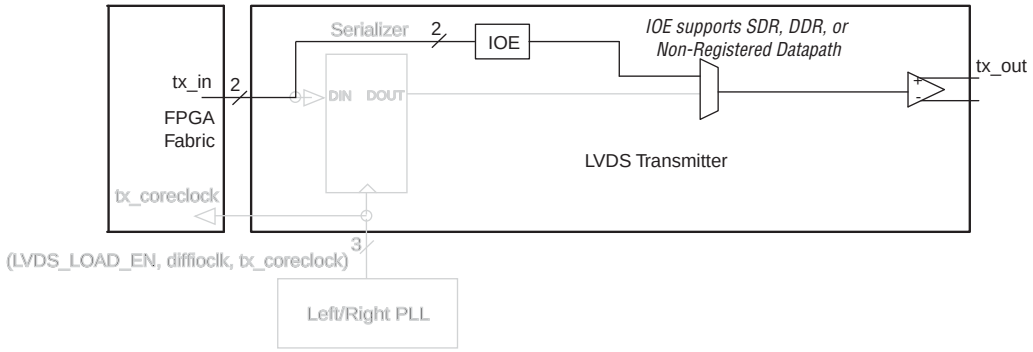
You can configure any Stratix IV transmitter data channel to generate a source-synchronous transmitter clock output. This flexibility allows the placement of the output clock near the data outputs to simplify board layout and reduce clock-to-data skew. Different applications often require specific clock-to-data alignments or specific data-rate-to-clock-rate factors. The transmitter can output a clock signal at the same rate as the data with a maximum frequency of 800 MHz. The output clock can also be divided by a factor of 1, 2, 4, 6, 8, or 10, depending on the serialization factor. You can set the phase of the clock in relation to the data at 0° or 180° (edge or center aligned). The left and right PLLs (`PLL_Lx` and `PLL_Rx`) provide additional support for other phase shifts in 45° increments. These settings are made statically in the Quartus II MegaWizard Plug-In Manager software. Figure 8–6 shows the Stratix IV transmitter in clock output mode. In clock output mode, you can use an LVDS channel as a clock output channel.

Figure 8–6. Stratix IV Transmitter in Clock Output Mode



You can bypass the Stratix IV serializer to support DDR (×2) and SDR (×1) operations to achieve a serialization factor of 2 and 1, respectively. The I/O element (IOE) contains two data output registers that can each operate in either DDR or SDR mode. Figure 8–7 shows the serializer bypass path.

Figure 8–7. Stratix IV Serializer Bypass *(Note 1), (2), (3)*



Notes to Figure 8–7:

(1) All disabled blocks and signals are grayed out.

(2) In DDR mode, `tx_inclock` clocks the IOE register. In SDR mode, data is directly passed through the IOE.

(3) In SDR and DDR modes, the data width to the IOE is 1 and 2 bits, respectively.

Programmable V_{OD} and Programmable Pre-Emphasis

Stratix IV LVDS transmitters support programmable pre-emphasis and programmable V_{OD}. Pre-emphasis increases the amplitude of the high-frequency component of the output signal, and thus helps to compensate for the frequency-dependent attenuation along the transmission line. Figure 8–8 shows the differential LVDS output.

Figure 8–8. Differential V_{OD}

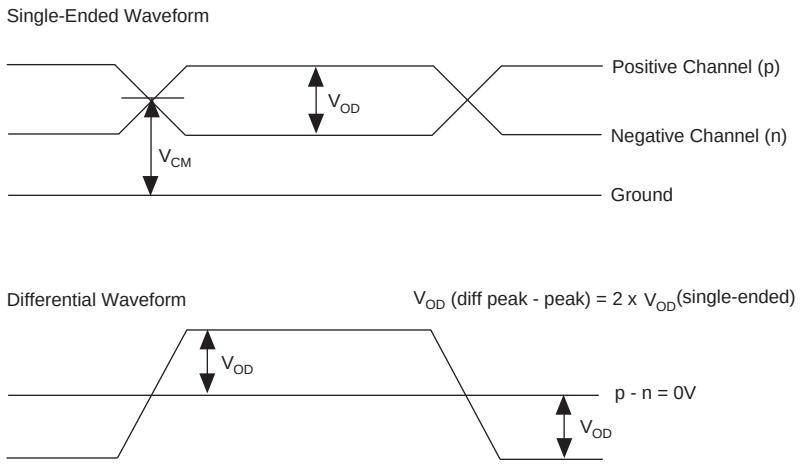
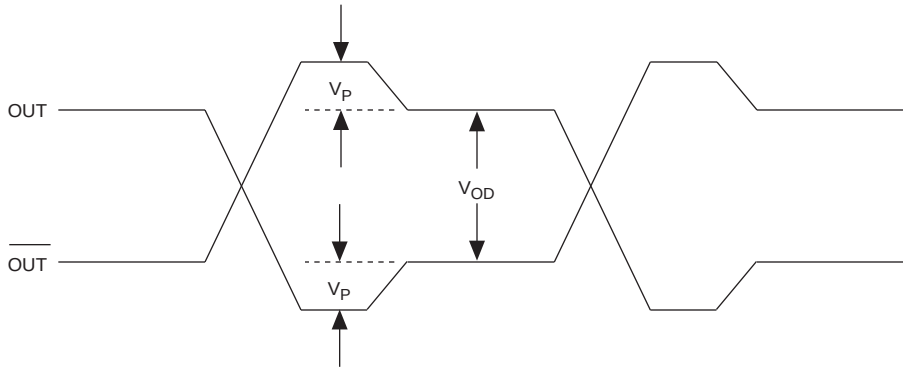


Figure 8–9 shows the LVDS output with pre-emphasis.

Figure 8–9. Programmable Pre-Emphasis *(Note 1)*



Note to Figure 8–9:

(1) V_P— voltage boost from pre-emphasis. V_{OD}— Differential output voltage (peak-peak).

Pre-emphasis is an important feature for high-speed transmission. Without pre-emphasis, the output current is limited by the V_{OD} setting and the output impedance of the driver. At high frequency, the slew rate may not be fast enough to reach full V_{OD} before the next edge, producing pattern-dependent jitter.

With pre-emphasis, the output current is boosted momentarily during switching to increase the output slew rate. The overshoot introduced by the extra current happens only during switching and does not ring, unlike the overshoot caused by signal reflection. The amount of pre-emphasis needed depends on the attenuation of the high-frequency component along the transmission line. The Quartus II software allows four settings for programmable pre-emphasis—zero (**0**), low (**1**), medium (**2**), and high (**3**). The default setting is low.

The V_{OD} is also programmable with four settings: low (**0**), medium low (**1**), medium high (**2**), and high (**3**). The default setting is medium low.

Programmable V_{OD}

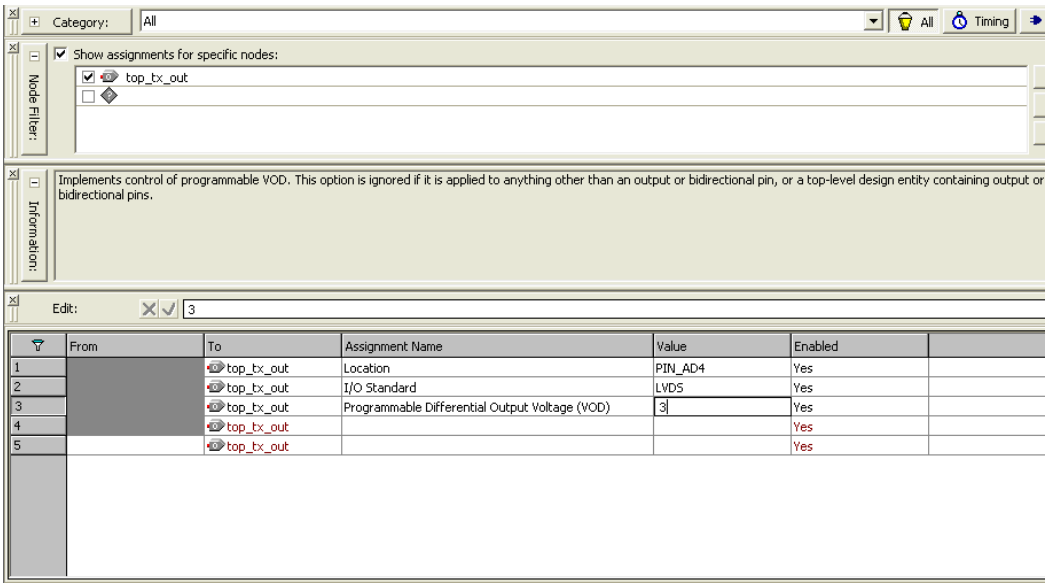
You can statically assign the V_{OD} settings from the Assignment Editor. Table 8–8 lists the assignment name for programmable V_{OD} and its possible values in the Quartus II software Assignment Editor.

Table 8–8. Quartus II Software Assignment Editor

To	`tx_out`
Assignment name	Programmable Differential Output Voltage (V_{OD})
Allowed values	0, 1, 2, 3

Figure 8–10 shows the assignment of programmable V_{OD} for a transmit data output from the Quartus II software Assignment Editor.

Figure 8–10. Quartus II Software Assignment Editor—Programmable V_{OD}



Programmable Pre-Emphasis

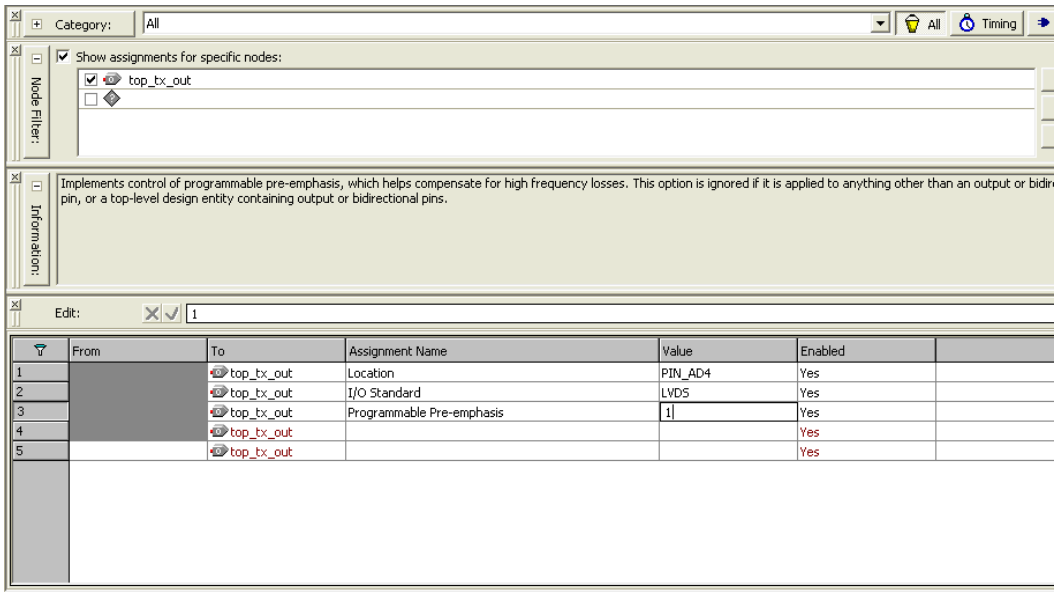
Four different settings are allowed for pre-emphasis from the Assignment Editor for each LVDS output channel. Table 8–9 lists the assignment name and its possible values for programmable pre-emphasis in the Quartus II software Assignment Editor.

Table 8–9. Quartus II Software Assignment Editor

To	`tx_out`
Assignment name	Programmable Pre-emphasis
Allowed values	0, 1, 2, 3

Figure 8–11 shows the assignment of programmable pre-emphasis for a transmit data output port from the Quartus II software Assignment Editor.

Figure 8–11. Quartus II Software Assignment Editor – Programmable Pre-Emphasis



Differential Receiver

The Stratix IV device family has a dedicated circuitry to receive high-speed differential signals in row I/Os. Figure 8–12 shows the hardware blocks of the Stratix IV receiver. The receiver has a differential buffer and left and right PLLs that can be shared between the transmitter and receiver, a DPA block, a synchronizer, a data realignment block, and a deserializer. The differential buffer can receive LVDS, mini-LVDS, and RSDS signal levels, which are statically set in the Quartus II software Assignment Editor.

The left and right PLL receives the external clock input and generates different phases of the same clock. The DPA block chooses one of the clocks from the left and right PLL and aligns the incoming data on each channel. The synchronizer circuit is a 1 bit wide by 6 bit deep FIFO buffer that compensates for any phase difference between the DPA clock and the data realignment block. If necessary, the user-controlled data realignment circuitry inserts a single bit of latency in the serial bit stream to align to the word boundary. The deserializer includes shift registers and parallel load registers, and sends a maximum of 10 bits to the internal logic.

The Stratix IV device family supports three different receiver modes:

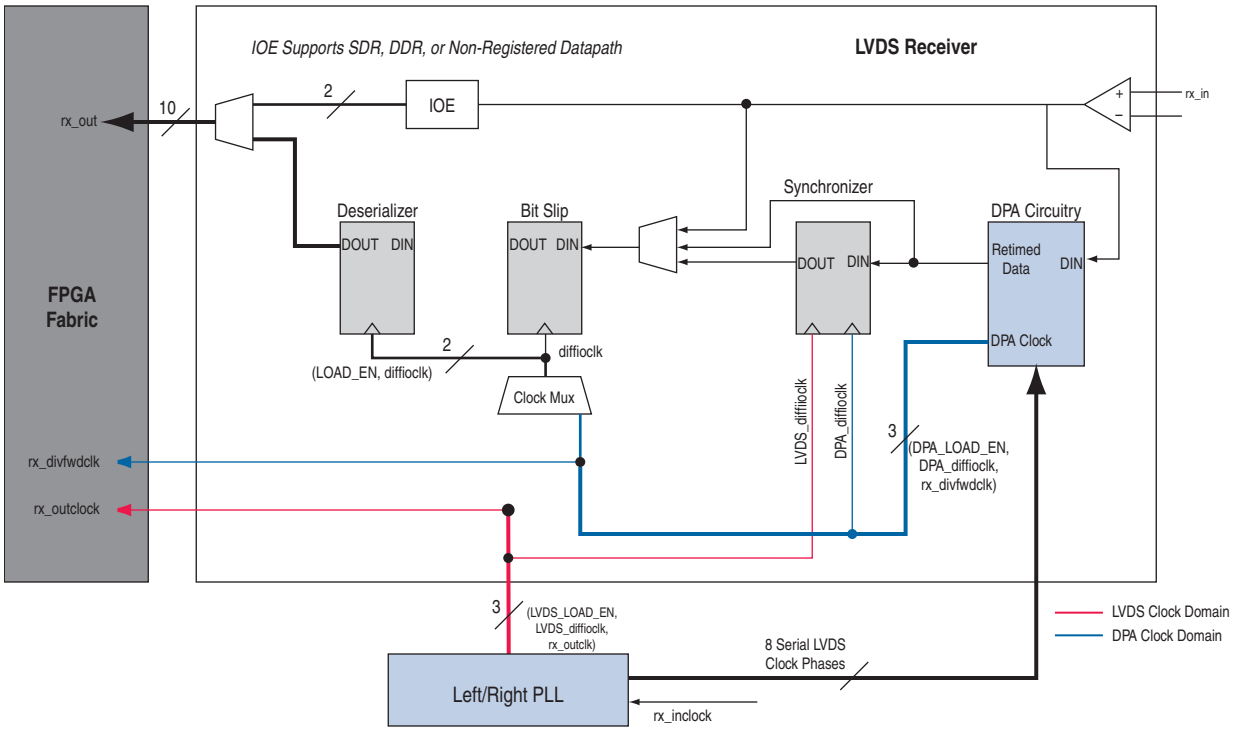
- Non-DPA mode
- DPA mode
- Soft-CDR mode

The physical medium connecting the transmitter and receiver LVDS channels may introduce skew between the serial data and the source-synchronous clock. The instantaneous skew between each LVDS channel and the clock also varies with the jitter on the data and clock signals as seen by the receiver. The three different modes—non-DPA, DPA, and soft-CDR—provide different options to overcome skew between the source synchronous clock (non-DPA, DPA) / reference clock (soft-CDR) and the serial data.

☞ Only non-DPA mode requires manual skew adjustment.

Non-DPA mode allows you to statically select the optimal phase between the source synchronous clock and the received serial data to compensate skew. In DPA mode, the DPA circuitry automatically chooses the best phase to compensate for the skew between the source synchronous clock and the received serial data. Soft-CDR mode provides opportunities for synchronous and asynchronous applications for chip-to-chip and short reach board-to-board applications for SGMII protocols.

Figure 8–12. Receiver Block Diagram *(Note 1)*, *(2)*



Notes to Figure 8–12:

(1) In SDR and DDR modes, the data width from the IOE is 1 and 2 bits, respectively.

(2) The `rx_out` port has a maximum data width of 10 bits.

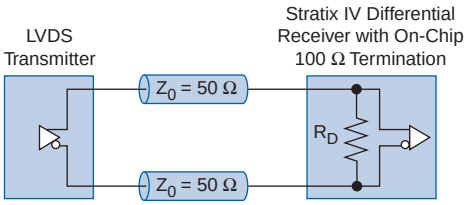
Differential I/O Termination

The Stratix IV device family provides a 100-Ω, on-chip differential termination option on each differential receiver channel for LVDS standards. On-chip termination saves board space by eliminating the need to add external resistors on the board. You can enable on-chip termination in the Quartus II software Assignment Editor.

On-chip differential termination is supported on all row I/O pins and dedicated clock input pins (`CLK[0,2,9,11]`). It is not supported for column I/O pins, dedicated clock input pins (`CLK[1,3,8,10]`), or the corner PLL clock inputs.

Figure 8–13 shows device on-chip termination.

Figure 8–13. On-Chip Differential I/O Termination



Receiver Hardware Blocks

The differential receiver has the following hardware blocks:

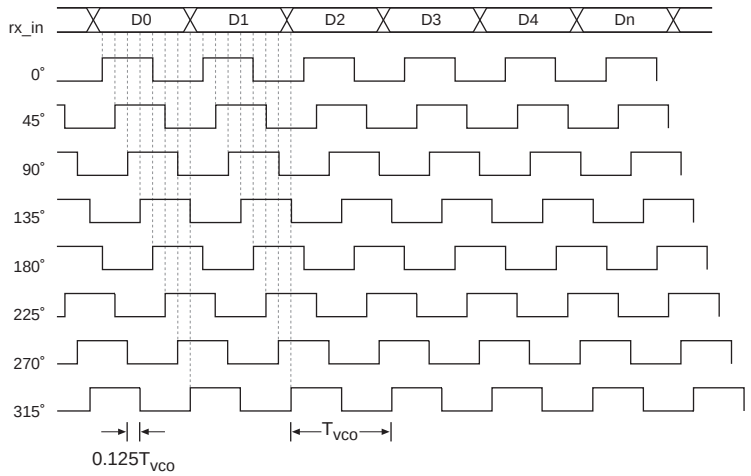
- DPA block
- Synchronizer
- Data realignment (bit slip)
- Deserializer

DPA Block

The DPA block takes in high-speed serial data from the differential input buffer and selects one of the eight phases generated by the left and right PLL to sample the data. The DPA chooses a phase closest to the phase of the serial data. The maximum phase offset between the received data and the selected phase is 1/8 UI, which is the maximum quantization error of the DPA. The eight phases of the clock are equally divided, offering a 45° resolution.

Figure 8–14 shows the possible phase relationships between the DPA clocks and the incoming serial data.

Figure 8–14. DPA Clock Phase to Serial Data Timing Relationship *(Note 1)*



Note to Figure 8–14:

(1) T_{VCO} is defined as the PLL serial clock period.

The DPA block continuously monitors the phase of the incoming serial data and selects a new clock phase if needed. You can prevent the DPA from selecting a new clock phase by asserting the optional `RX_DPLL_HOLD` port, which is available for each channel.

DPA circuitry does not require a fixed training pattern to lock to the optimum phase out of the eight phases. After reset or power up, DPA circuitry requires transitions on the received data to lock to the optimum phase. An optional output port, `RX_DPA_LOCKED`, is available to indicate an initial DPA lock condition to the optimum phase after power up or reset. This signal is not de-asserted if the DPA selects a new phase out of the eight clock phases to sample the received data. Do not use the `rx_dpa_locked` signal to determine a DPA loss-of-lock condition. Use data checkers such as a cyclic redundancy check (CRC) or diagonal interleaved parity (DIP-4) to validate the data.

An independent reset port, `RX_RESET`, is available to reset the DPA circuitry. DPA circuitry must be retrained after reset.

☞ The DPA block is bypassed in non-DPA mode.

Synchronizer

The synchronizer is a 1 bit wide and 6 bit deep FIFO buffer that compensates for the phase difference between `DPA_diffioclk`, which is the optimal clock selected by the DPA block, and `LVDS_diffioclk`, which is produced by the left and right PLL. The synchronizer can only compensate for phase differences, not frequency differences between the data and the receiver's input reference clock.

An optional port, RX_FIFO_RESET, is available to the internal logic to reset the synchronizer. The synchronizer is automatically reset when the DPA first locks to the incoming data. Altera recommends using RX_FIFO_RESET to reset the synchronizer when the DPA signals a loss-of-lock condition and the data checker indicates corrupted received data.

The synchronizer circuit is bypassed in non-DPA and soft-CDR mode.

Data Realignment Block (Bit Slip)

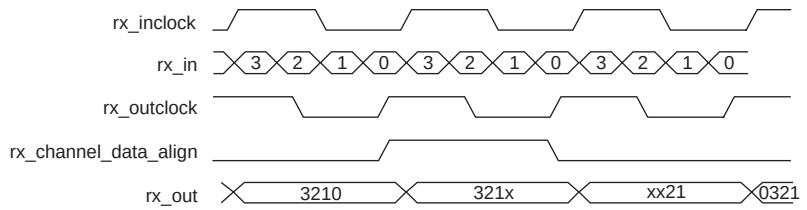
Skew in the transmitted data along with skew added by the link causes channel-to-channel skew on the received serial data streams. If the DPA is enabled, the received data is captured with different clock phases on each channel. This may cause the received data to be misaligned from channel-to-channel. To compensate for this channel-to-channel skew and establish the correct received word boundary at each channel, each receiver channel has a dedicated data realignment circuit that realigns the data by inserting bit latencies into the serial stream.

An optional RX_CHANNEL_DATA_ALIGN port controls the bit insertion of each receiver independently controlled from the internal logic. The data slips one bit on the rising edge of RX_CHANNEL_DATA_ALIGN. The requirements for the RX_CHANNEL_DATA_ALIGN signal include:

- The minimum pulse width is one period of the parallel clock in the logic array.
- The minimum low time between pulses is one period of the parallel clock.
- This is an edge-triggered signal.
- Valid data is available two parallel clock cycles after the rising edge of RX_CHANNEL_DATA_ALIGN.

Figure 8–15 shows receiver output (RX_OUT) after one bit slip pulse with the deserialization factor set to **4**.

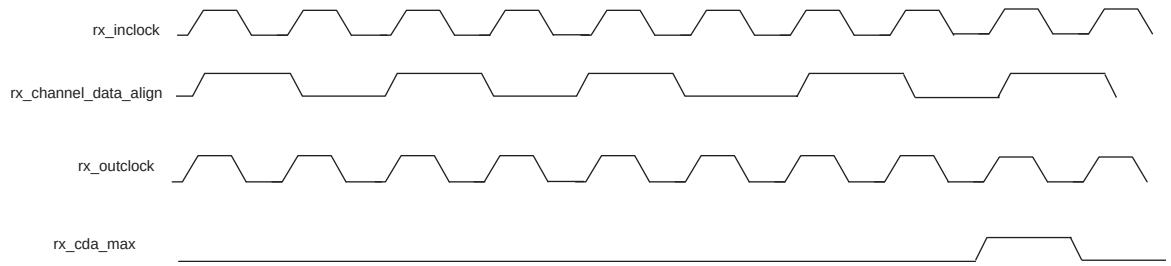
Figure 8–15. Data Realignment Timing



The data realignment circuit can have up to 11 bit-times of insertion before a rollover occurs. The programmable bit rollover point can be from 1 to 11 bit-times, independent of the deserialization factor. The programmable bit rollover point must be set equal to or greater than the deserialization factor, allowing enough depth in the word alignment circuit to slip through a full word. You can set the value of the bit rollover point using the MegaWizard Plug-In Manager software. An optional status port, RX_CDA_MAX, is available to the FPGA fabric from each channel to indicate when the preset rollover point is reached.

Figure 8–16 shows a preset value of four bit-times before rollover occurs. The `rx_cda_max` signal pulses for one `rx_outclock` cycle to indicate that rollover has occurred.

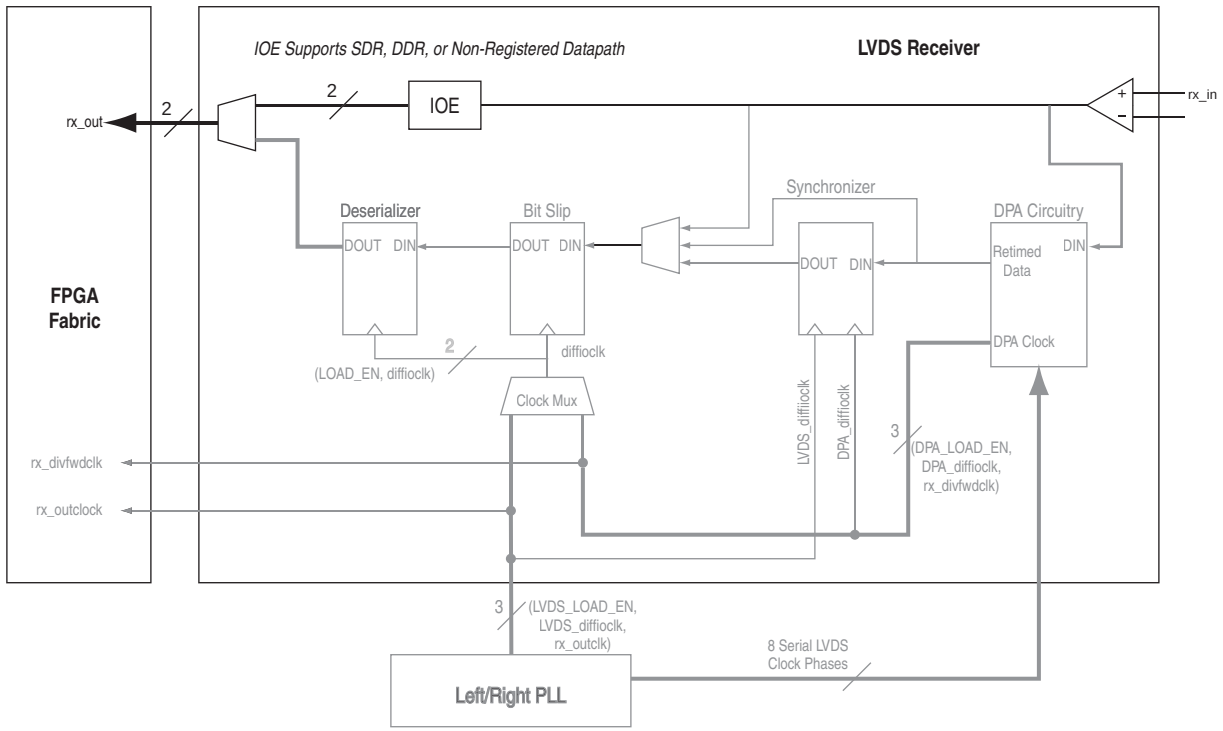
Figure 8–16. Receiver Data Re-alignment Rollover



Deserializer

You can statically set the deserialization factor to 4, 6, 7, 8, or 10 by using the Quartus II software. You can bypass the Stratix IV deserializer in the Quartus II MegaWizard Plug-In Manager software to support DDR (×2) or SDR (×1) operations, as shown Figure 8–17. The DPA and data realignment circuit cannot be used when the deserializer is bypassed. The IOE contains two data input registers that can operate in DDR or SDR mode.

Figure 8–17. Stratix IV Deserializer Bypass *(Note 1)*, *(2)*, *(3)*



Notes to Figure 8–17:

(1) All disabled blocks and signals are grayed out.

(2) In DDR mode, `rx_inclock` clocks the IOE register. In SDR mode, data is directly passed through the IOE.

(3) In SDR and DDR modes, the data width from the IOE is 1 and 2 bits, respectively.

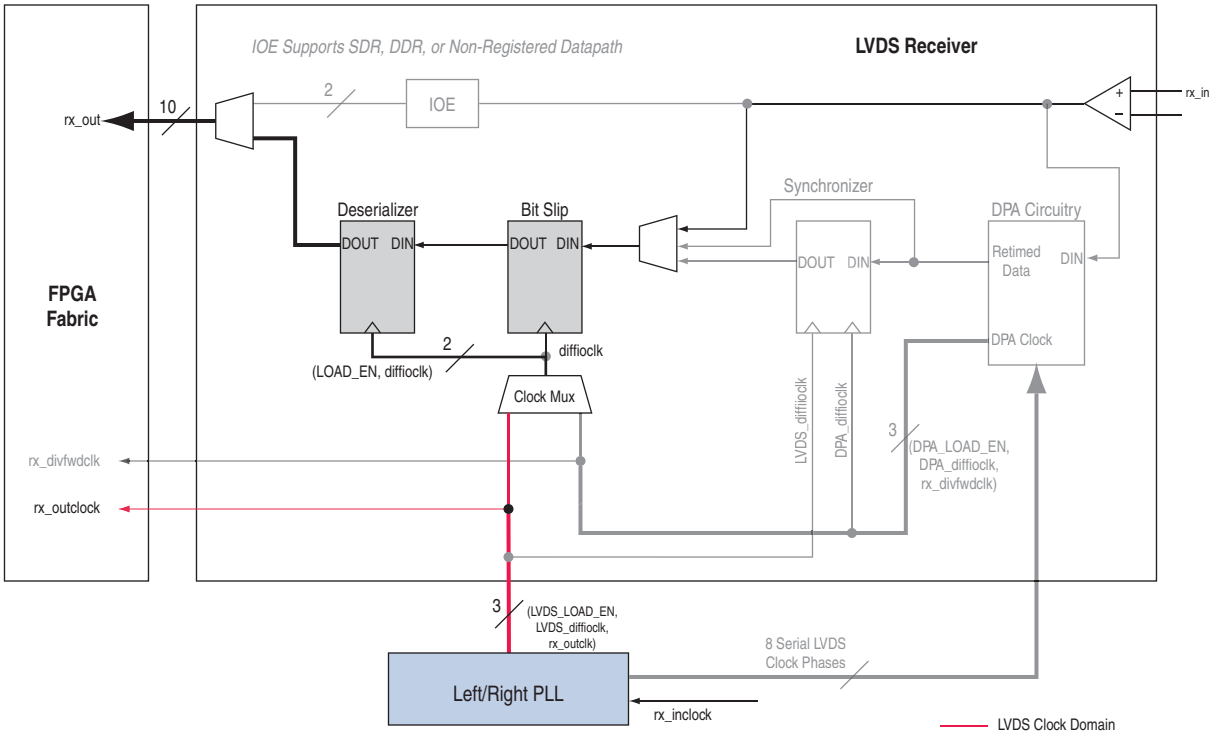
Receiver Data Path Modes

The Stratix IV device family supports three receiver datapath modes—non-DPA mode, DPA mode, and soft-CDR mode.

Non-DPA Mode

Figure 8–18 shows the non-DPA datapath block diagram. In non-DPA mode, the DPA and synchronizer blocks are disabled. Input serial data is registered at the rising or falling edge of the serial `LVDS_diffioclk` clock produced by the left and right PLL. You can select the rising/falling edge option using the ALTLDVS MegaWizard Plug-in Manager software. Both data realignment and deserializer blocks are clocked by the `LVDS_diffioclk` clock, which is generated by the left and right PLL.

Figure 8–18. Receiver Data Path in Non-DPA Mode *(Note 1)*, *(2)*



Notes to Figure 8–18:

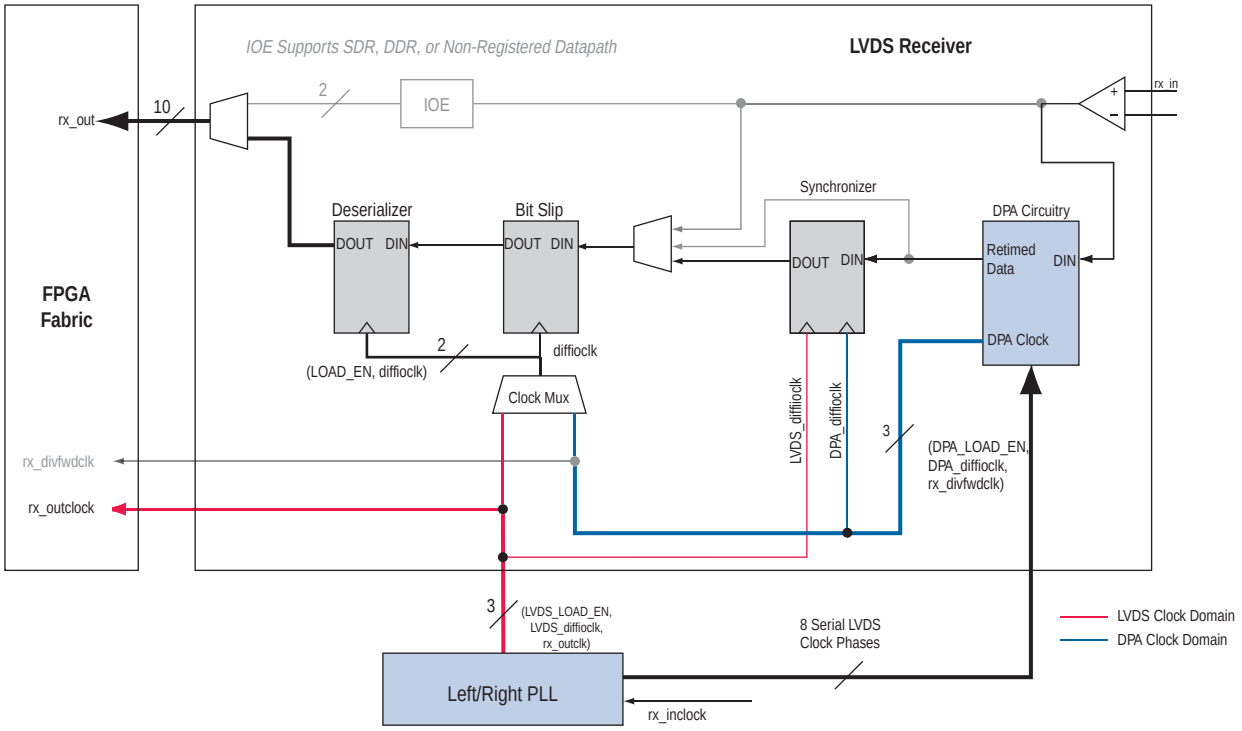
(1) In SDR and DDR modes, the data width from the IOE is 1 and 2 bits, respectively.

(2) The `rx_out` port has a maximum data width of 10 bits.

DPA Mode

Figure 8–19 shows the DPA mode datapath, where all the hardware blocks mentioned in "Receiver Hardware Blocks" on page 8–18 are active. The DPA block chooses the best possible clock (`DPA_diffioclk`) from the eight fast clocks sent by the left and right PLL. This serial `DPA_diffioclk` clock is used for writing the serial data into the synchronizer. A serial `LVDS_diffioclk` clock is used for reading the serial data from the synchronizer. The same `LVDS_diffioclk` clock is used in data realignment and deserializer blocks.

Figure 8–19. Receiver Datapath in DPA Mode *(Note 1), (2), (3)*



Notes to Figure 8–19:

(1) All disabled blocks and signals are grayed out.

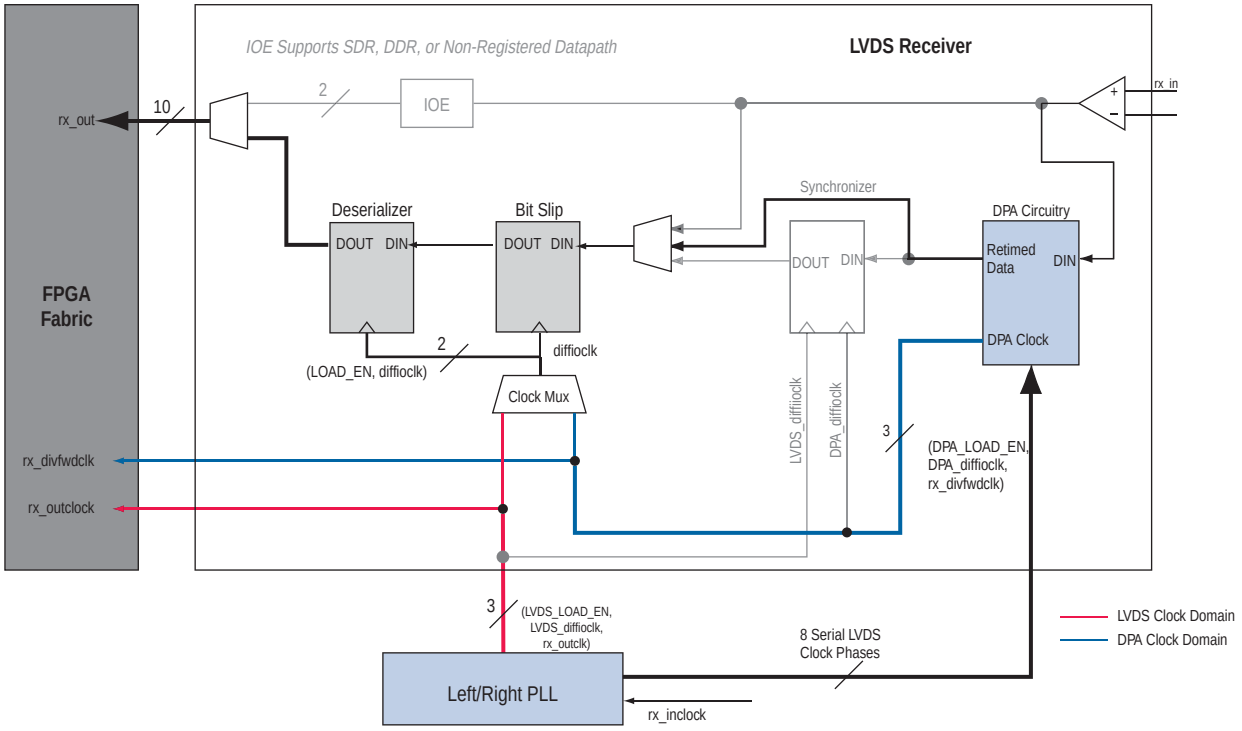
(2) In SDR and DDR modes, the data width from the IOE is 1 and 2 bits, respectively.

(3) The `rx_out` port has a maximum data width of 10 bits.

Soft-CDR Mode

The Stratix IV LVDS channel offers the soft-CDR mode to support the Gigabit Ethernet and SGMII protocols. A receiver PLL uses the local clock source for reference. Figure 8–20 shows the soft-CDR mode datapath.

Figure 8–20. Receiver Datapath in Soft-CDR Mode *(Note 1), (2), (3)*



Notes to Figure 8–20:

(1) All disabled blocks and signals are grayed out.

(2) In SDR and DDR modes, the data width from the IOE is 1 and 2 bits, respectively.

(3) The `rx_out` port has a maximum data width of 10 bits.

In soft-CDR mode, the synchronizer block is inactive. The DPA circuitry selects an optimal DPA clock phase to sample the data. Use the selected DPA clock for bit-slip operation and deserialization. The DPA block also forwards the selected DPA clock, divided by the deserialization factor called `rx_divfwdclk`, to the FPGA fabric, along with the deserialized data. This clock signal is put on the periphery clock (PCLK) network. When using soft-CDR mode, the `rx_reset` port must not be asserted after the `rx_dpa_lock` is asserted because the DPA will continuously choose new phase taps from the PLL to track parts per million (PPM) differences between the reference clock and incoming data.

For more information about PCLK networks, refer to the *Clock Networks and PLLs in Stratix IV Devices* chapter.

You can use every LVDS channel in soft-CDR mode and can drive the FPGA fabric using the PCLK network in the Stratix IV device family. The `rx_dpa_locked` signal is not valid in soft-CDR mode, as the DPA continuously changes its phase to track PPM differences between the upstream transmitter and the local receiver input reference clocks. The parallel clock `rx_outclock`, generated by the left and right PLL, is also forwarded to the FPGA fabric.

LVDS Interface with the Use External PLL Option Enabled

The ALTLVDS MegaWizard Plug-In Manager software provides an option for implementing the LVDS interface with the **Use External PLL** option. With this option enabled, you can control the PLL settings, such as dynamically reconfiguring the PLL to support different data rates, dynamic phase shift, and other settings. You also must instantiate an ALTPLL megafunction to generate the various clock and load enable signals.

When you enable the **Use External PLL** option with the ALTLVDS transmitter and receiver, the following signals are required from the ALTPLL megafunction:

- Serial clock input to the SERDES of the ALTLVDS transmitter and receiver
- Load enable to the SERDES of the ALTLVDS transmitter and receiver
- Parallel clock used to clock the transmitter FPGA fabric logic and parallel clock used for the receiver `rx_syncclock` port and receiver FPGA fabric logic
- Asynchronous PLL reset port of the ALTLVDS receiver

As an example, Table 8–10 describes the serial clock output, load enable output, and parallel clock output generated on ports c0, c1, and c2, respectively, along with the locked signal of the ALTPLL instance. You can choose any of the PLL output clock ports to generate the interface clocks.

Table 8–10 lists the signal interface between the output ports of the ALTPLL megafunction and input ports of the ALTLVDS transmitter and receiver.

Table 8–10. Signal Interface Between ALTPLL and ALTLVDS Megafunctions

From the ALTPLL Megafunction	To the ALTLVDS Transmitter	To the ALTLVDS Receiver
Serial clock output (c0)	`tx_inclock` (serial clock input to the transmitter)	`rx_inclock` (serial clock input)
Load enable output (c1)	`tx_enable` (load enable to the transmitter)	`rx_enable` (load enable for the deserializer)
Parallel clock output (c2)	Parallel clock used inside the transmitter core logic in the FPGA fabric	`rx_syncclock` (parallel clock input) and parallel clock used inside the receiver core logic in the FPGA fabric
~(locked)	—	`pll_areset` (asynchronous PLL reset port) *(1)*

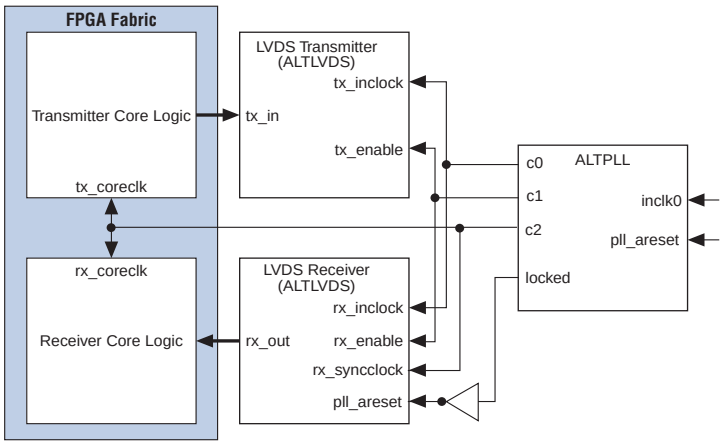
Note to Table 8–10:

(1) The `pll_areset` signal automatically is enabled for the LVDS receiver in external PLL mode. This signal does not exist for LVDS transmitter instantiation when the external PLL option is enabled.

The `rx_syncclock` port is enabled automatically in an LVDS receiver in external PLL mode. The Quartus II compiler errors out if this port is not connected as shown in Figure 8–21.

When generating the ALTPLL megafunction, the **Left/Right PLL** option is configured to set up the PLL in LVDS mode. Figure 8–21 shows the connection between the ALTPLL and ALTLVDS megafunctions.

Figure 8–21. LVDS Interface with the ALTPLL Megafunction *(Note 1)*



Note to Figure 8–21:

(1) Instantiation of `pll_areset` is optional for the ALTPLL instantiation.

Example 8–1 shows how to generate three output clocks using an ALTPLL megafunction.

Example 8–1. Generating Three Output Clocks Using an ALTPLL Megafunction

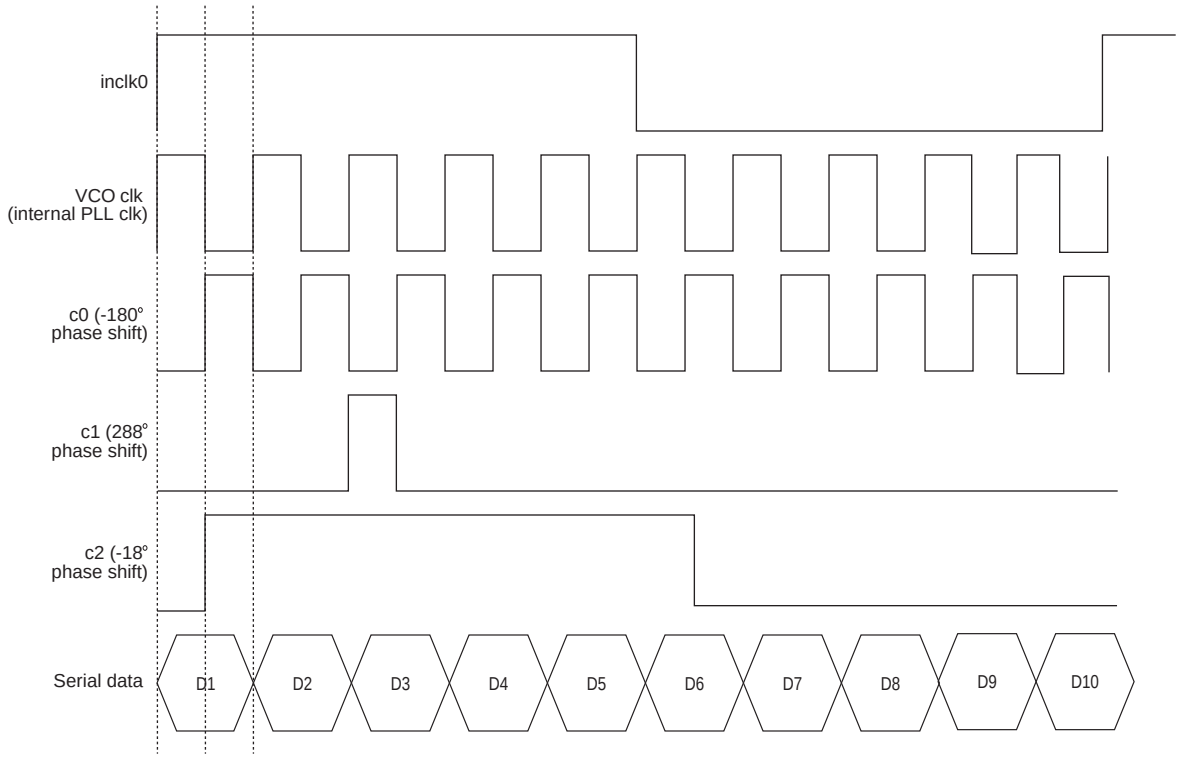
LVDS data rate = 1 Gbps; serialization factor = 10; input reference clock = 100 MHz

The following settings are used when generating the three output clocks using an ALTPLL megafunction. The serial clock must be **1000 MHz** and the parallel clock must be 100 MHz (serial clock divided by the serialization factor):

- c0
  - Frequency = **1000 MHz** (multiplication factor = 10 and division factor = 1)
  - Phase shift = **–180°** with respect to the voltage-controlled oscillator (VCO) clock
  - Duty cycle = **50%**
- c1
  - Frequency = (1000/10) = **100 MHz** (multiplication factor = 1 and division factor = 1)
  - Phase shift = (10 - 2) × 360/10 = **288°** [(deserialization factor - 2)/deserialization factor] × 360°
  - Duty cycle = (100/10) = **10%** (100 divided by the serialization factor)
- c2
  - Frequency = (1000/10) = **100 MHz** (multiplication factor = 1 and division factor = 1)
  - Phase shift = (–180/10) = **–18°** (c0 phase shift divided by the serialization factor)
  - Duty cycle = **50%**

The Equation 8–1 calculations for phase shift assume that the input clock and serial data are edge aligned. Introducing a phase shift of –180° to sampling clock (c0) ensures that the input data is center-aligned with respect to the c0, as shown in Figure 8–22.

Figure 8–22. Phase Relationship for External PLL Interface Signals



Left and Right PLLs (PLL_Lx and PLL_Rx)

The Stratix IV device family contains up to eight left and right PLLs with up to four PLLs located on the left side and four on the right side of the device. The left PLLs can support high-speed differential I/O banks on the left side; the right PLLs can support high-speed differential I/O banks on the right side of the device. The high-speed differential I/O receiver and transmitter channels use these left and right PLLs to generate the parallel clocks (`rx_outclock` and `tx_outclock`) and high-speed clocks (`diffioclk`).

Figure 8–2 on page 8–3 and Figure 8–3 on page 8–4 show the locations of the left and right PLLs for Stratix IV E, GT, and GX devices, respectively. The PLL VCO operates at the clock frequency of the data rate. Clock switchover and dynamic reconfiguration are allowed using the left and right PLL in high-speed differential I/O support mode.

For more information, refer to the *Clock Network and PLLs in Stratix IV Devices* chapter.

Stratix IV Clocking

The left and right PLLs feed into the differential transmitter and receive channels through the LVDS and DPA clock network. The center left and right PLLs can clock the transmitter and receive channels above and below them. The corner left and right PLLs can drive I/Os in the banks adjacent to them.

Figure 8–23 shows center PLL clocking in the Stratix IV device family. For more information about PLL clocking restrictions, refer to "Differential Pin Placement Guidelines" on page 8–37.

Figure 8–23. LVDS/DPA Clocks in the Stratix IV Device Family with Center PLLs

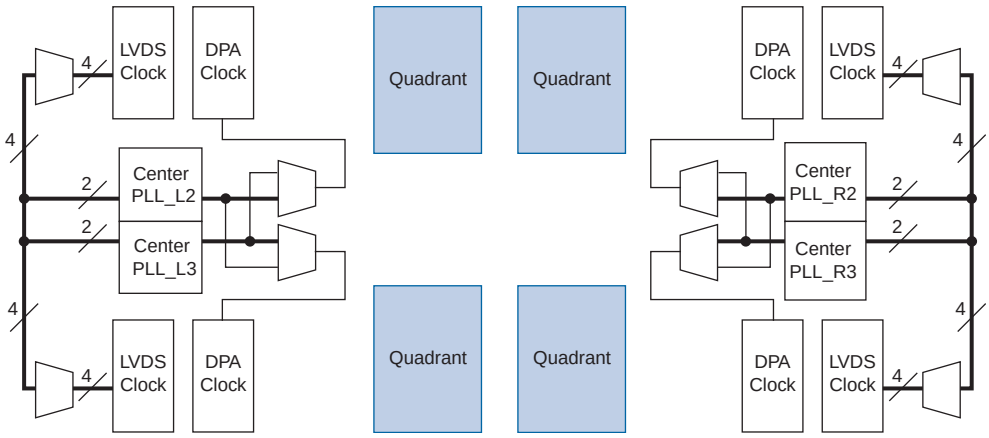
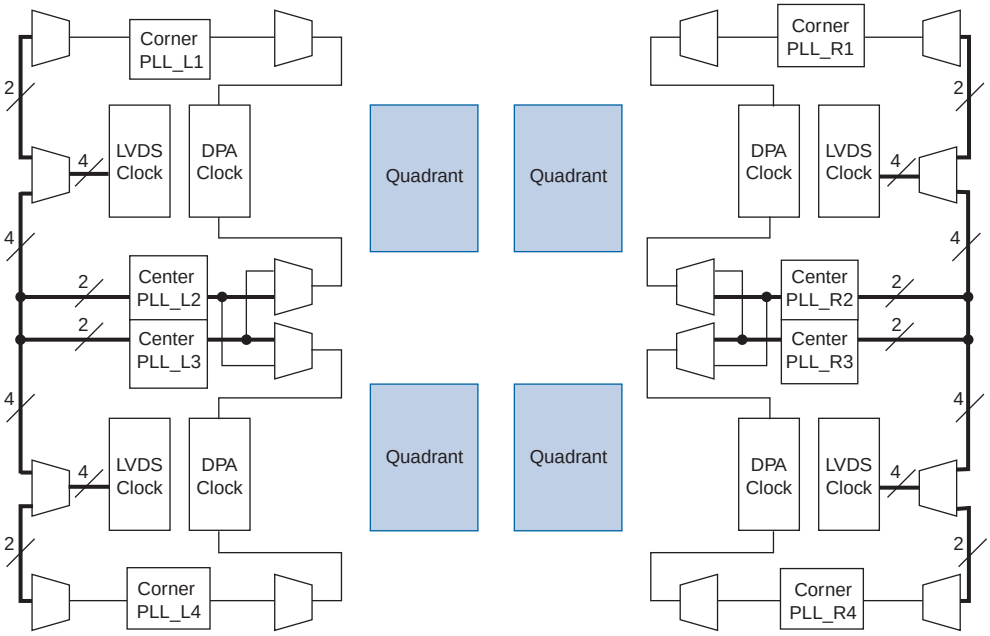


Figure 8–24 shows center and corner PLL clocking in the Stratix IV device family. For more information about PLL clocking restrictions, refer to "Differential Pin Placement Guidelines" on page 8–37.

Figure 8–24. LVDS/DPA Clocks in the Stratix IV Device Family with Center and Corner PLLs



Source-Synchronous Timing Budget

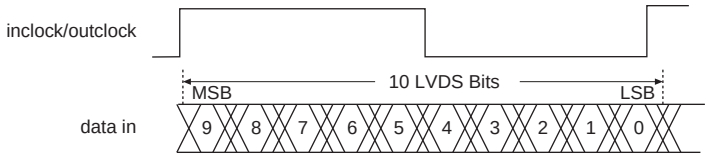
This section describes the timing budget, waveforms, and specifications for source-synchronous signaling in the Stratix IV device family. LVDS I/O standards enable high-speed data transmission. This high data transmission rate results in better overall system performance. To take advantage of fast system performance, it is important to understand how to analyze timing for these high-speed signals. Timing analysis for the differential block is different from traditional synchronous timing analysis techniques.

Instead of focusing on clock-to-output and setup times, source synchronous timing analysis is based on the skew between the data and the clock signals. High-speed differential data transmission requires the use of timing parameters provided by IC vendors and is strongly influenced by board skew, cable skew, and clock jitter. This section defines the source-synchronous differential data orientation timing parameters, the timing budget definitions for the Stratix IV device family, and how to use these timing parameters to determine a design's maximum performance.

Differential Data Orientation

There is a set relationship between an external clock and the incoming data. For operations at 1 Gbps and a serialization factor of 10, the external clock is multiplied by 10. You can set phase-alignment in the PLL to coincide with the sampling window of each data bit. The data is sampled on the falling edge of the multiplied clock. Figure 8–25 shows the data bit orientation of the ×10 mode.

Figure 8–25. Bit Orientation in the Quartus II Software



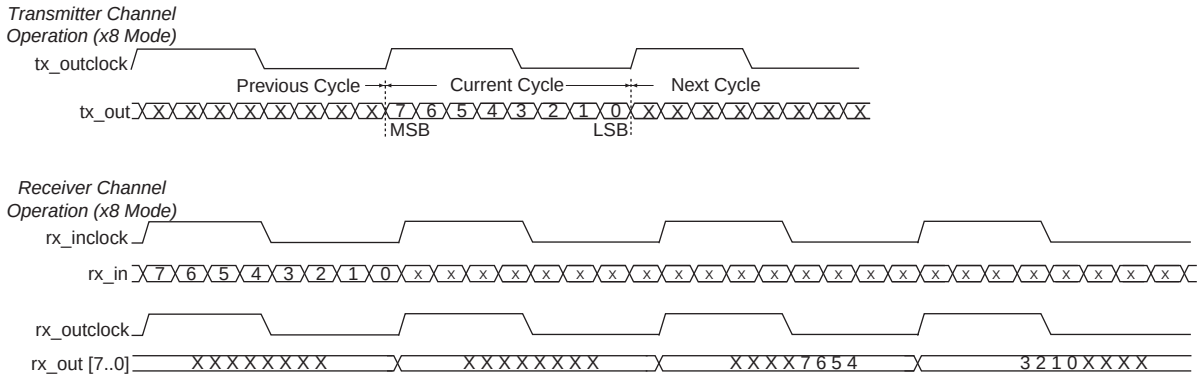
Differential I/O Bit Position

Data synchronization is necessary for successful data transmission at high frequencies. Figure 8–26 shows the data bit orientation for a channel operation. This figure is based on the following:

- Serialization factor equals the clock multiplication factor
- Edge alignment is selected for phase alignment
- Implemented in hard SERDES

For other serialization factors, use the Quartus II software tools to find the bit position within the word. The bit positions after deserialization are listed in Table 8–11.

Figure 8–26. Bit-Order and Word Boundary for One Differential Channel *(Note 1)*



Note to Figure 8–26:

(1) These are only functional waveforms and are not intended to convey timing information.

Table 8–11 lists the conventions for differential bit naming for 18 differential channels. The MSB and LSB positions increase with the number of channels used in a system.

Table 8–11. Differential Bit Naming

Receiver Channel Data Number	Internal 8-Bit Parallel Data	
	MSB Position	LSB Position
1	7	0
2	15	8
3	23	16
4	31	24
5	39	32
6	47	40
7	55	48
8	63	56
9	71	64
10	79	72
11	87	80
12	95	88
13	103	96
14	111	104
15	119	112
16	127	120
17	135	128
18	143	136

Transmitter-Channel-to-Channel Skew

Transmitter-channel-to-channel skew (TCCS) is an important parameter based on the Stratix IV transmitter in a source synchronous differential interface. This parameter is used in receiver skew margin calculation. For more information, refer to "Receiver Skew Margin for Non-DPA Mode" on page 8–32.

TCCS is the difference between the fastest and slowest data output transitions, including the TCO variation and clock skew. For LVDS transmitters, the TimeQuest Timing Analyzer provides a TCCS report, which shows TCCS values for serial output ports.

You can get the TCCS value from the TCCS report (`report_TCCS`) in the Quartus II compilation report under the TimeQuest Timing Analyzer, or from the *DC and Switching Characteristics* chapter.

Receiver Skew Margin for Non-DPA Mode

Changes in system environment, such as temperature, media (cable, connector, or PCB), and loading effect the receiver's setup and hold times; internal skew affects the sampling ability of the receiver.

Different modes of LVDS receivers use different specifications, that can help in deciding the ability to sample the received serial data correctly. In DPA mode, you must use DPA jitter tolerance instead of receiver input skew margin (RSKM).

In non-DPA mode, use TCCS, RSKM, and sampling window (SW) specifications for high-speed source-synchronous differential signals in the receiver data path. The relationship between RSKM, TCCS, and SW can be expressed by the RSKM equation shown in Equation 8–1.

Equation 8–1. RSKM

$$RSKM = \frac{TUI - SW - TCCS}{2}$$

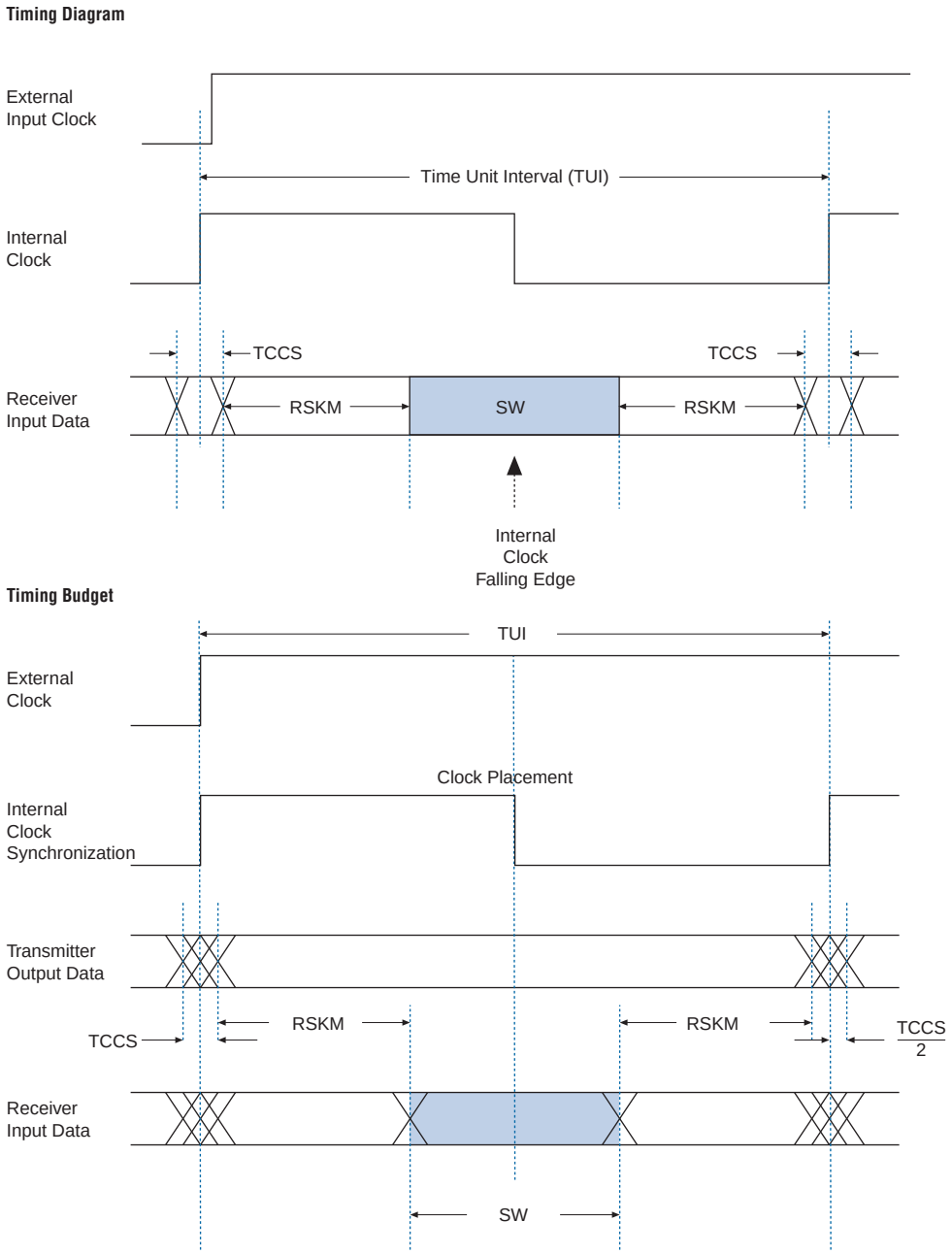
Conventions used for the equation:

- Time unit interval (TUI)—Time period of the serial data.
- RSKM—The timing margin between the receiver's clock input and the data input sampling window.
- SW—The period of time that the input data must be stable to ensure that data is successfully sampled by the LVDS receiver. The SW is a device property and varies with device speed grade.
- TCCS—The timing difference between the fastest and the slowest output edges, including t_{CO} variation and clock skew, across channels driven by the same PLL. The clock is included in the TCCS measurement.

Figure 8–27 shows the relationship between the RSKM, TCCS, and the receiver's SW.

You must calculate the RSKM value to decide whether or not data can be sampled properly by the LVDS receiver with the given data rate and device. A positive RSKM value indicates that the LVDS receiver can sample the data properly, whereas a negative RSKM indicates that it cannot.

Figure 8–27. Differential High-Speed Timing Diagram and Timing Budget for Non-DPA Mode



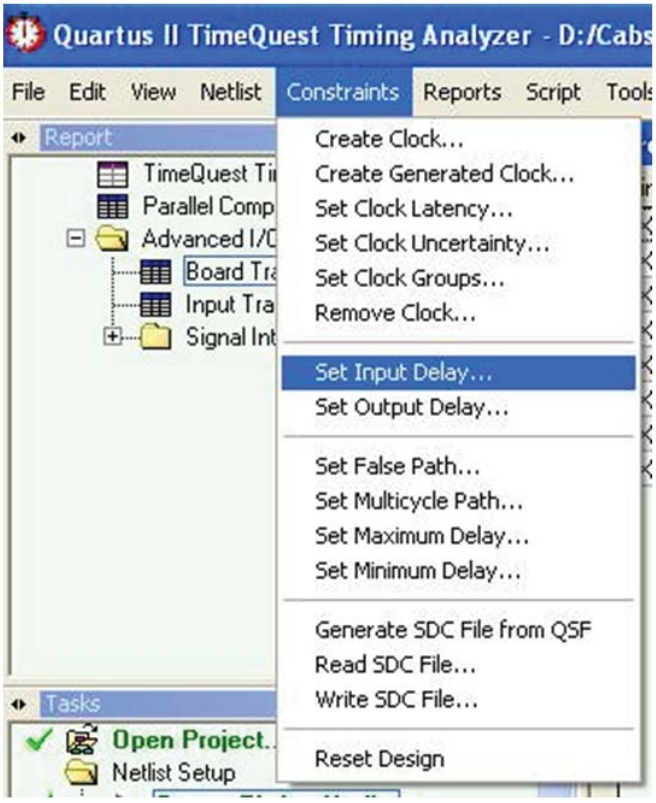
For LVDS receivers, the Quartus II software provides an RSKM report showing the SW, TUI, and RSKM values for non-DPA mode. You can generate the RSKM report by executing the `report_RSKM` command in the TimeQuest Timing Analyzer. You can find the RSKM report in the Quartus II compilation report under the TimeQuest Timing Analyzer section.

In order to obtain the RSKM value, you must assign an appropriate input delay to the LVDS receiver through the TimeQuest Timing Analyzer constraints menu.

For assigning input delay, follow these steps:

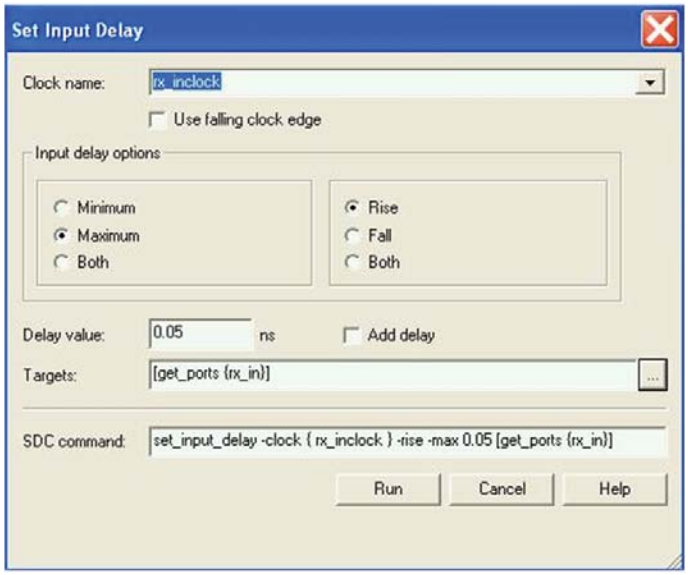
1. The Quartus II TimeQuest Timing Analyzer GUI has many options for setting the constraints and analyzing the design. Figure 8–28 shows various commands on the Constraints menu. For setting input delay, you must select the **Set Input Delay** option.

Figure 8–28. Selection of Constraint Menu in TimeQuest Timing Analyzer



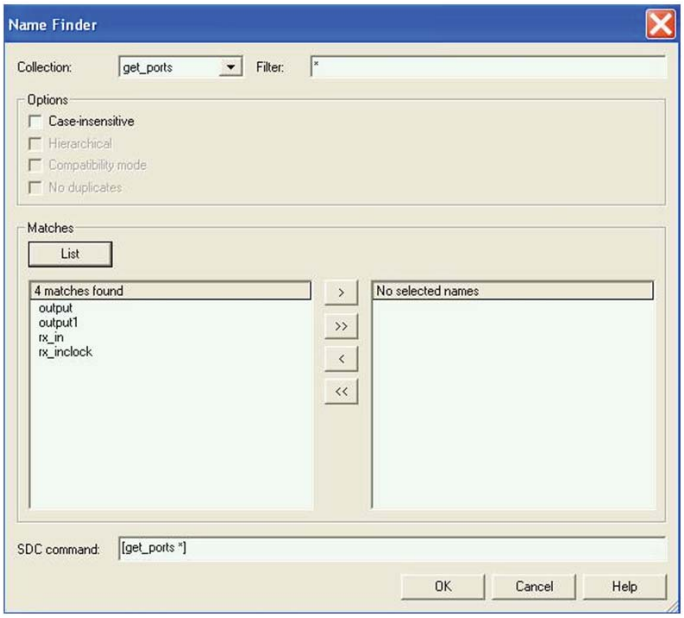
2. Figure 8–29 shows the setting parameters for the **Set Input Delay** option. The clock name must reference the source synchronous clock that feeds the LVDS receiver. Select the desired clock using the pull down menu.

Figure 8–29. Input Time Delay Assignment Through TimeQuest Timing Analyzer



3. Figure 8–30 shows the **Targets** option. You can view a list of all available ports using the **List** option in the **Name Finder** window.

Figure 8–30. Name Finder Window in Set Input Delay Option



4. Select the LVDS receiver serial input ports (from the list) according to the input delay you set. Click **OK**.

5. In the **Set Input Delay** window, set the appropriate values in the **Input Delay Options** section and **Delay** value.
6. Click **Run** to incorporate these values in the TimeQuest Timing Analyzer.
7. Assign the appropriate delay for all the LVDS receiver input ports following these steps. If you have already assigned **Input Delay** and you need to add more delay to that input port, use the **Add Delay** option in the **Set Input Delay** window.

If no input delay is set in the TimeQuest Timing Analyzer, the receiver-channel-to-channel skew (RCCS) defaults to zero. You can also directly set the input delay in a synopsys design constraint file (**.sdc**) using the `set_input_delay` command.

For more information about **.sdc** commands and the TimeQuest Timing Analyzer, refer to the *Quartus II TimeQuest Timing Analyzer* chapter in volume 3 of the *Quartus II Development Software Handbook*.

Example 8–2 shows the RSKM calculation.

Example 8–2. RSKM

Data Rate: 1 Gbps, Board channel-to-channel skew = 200 ps

For Stratix IV devices:

TCCS = **100 ps** (pending characterization)
SW = **300 ps** (pending characterization)

TUI = **1000 ps**
Total RCCS = TCCS + Board channel-to-channel skew= 100 ps + 200 ps
= **300 ps**

RSKM = **TUI - SW - RCCS**
= **1000 ps - 300 ps - 300 ps**
= **400 ps > 0**

Because the RSKM > 0 ps, receiver non-DPA mode must work correctly.

You can also calculate RSKM using the steps described in "Guidelines for DPA-Enabled Differential Channels" on page 8–37.

Differential Pin Placement Guidelines

To ensure proper high-speed operation, differential pin placement guidelines have been established. The Quartus II compiler automatically checks that these guidelines are followed and issues an error message if they are not met.

This section is divided into pin placement guidelines with and without DPA usage because DPA usage adds some constraints on the placement of high-speed differential channels.

DPA-enabled differential channels refer to DPA mode or soft-CDR mode; DPA disabled channels refer to non-DPA mode.

Guidelines for DPA-Enabled Differential Channels

The Stratix IV device family has differential receivers and transmitters in I/O banks on the left and right sides of the device. Each receiver has a dedicated DPA circuit to align the phase of the clock to the data phase of its associated channel. When you use DPA-enabled channels in differential banks, you must adhere to the guidelines listed in the following sections.

DPA-Enabled Channels and Single-Ended I/Os

When you enable a DPA channel in a bank, both single-ended I/Os and differential I/O standards are allowed in the bank.

- Single-ended I/Os are allowed in the same I/O bank, as long as the single-ended I/O standard uses the same V_{CCIO} as the DPA-enabled differential I/O bank.
- Single-ended inputs can be in the same logic array block (LAB) row as a differential channel using the SERDES circuitry.
- DDIO can be placed within the same LAB row as a SERDES differential channel but half rate DDIO (single data rate) output pins cannot be placed within the same LAB row as a receiver SERDES differential channel. The input register must be implemented within the FPGA fabric logic.

DPA-Enabled Channel Driving Distance

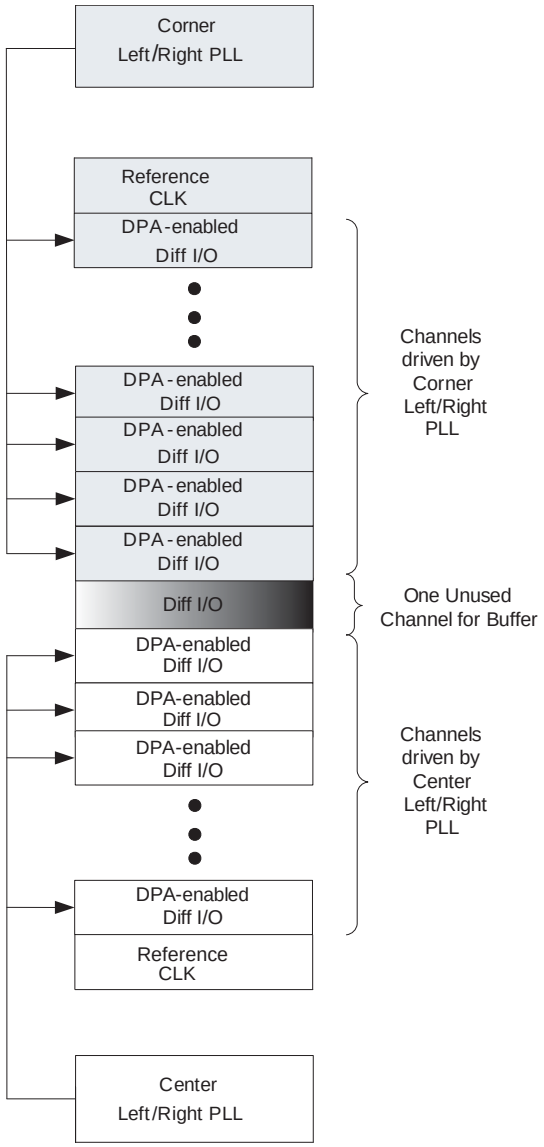
If the number of DPA channels driven by each left and right PLL exceeds 25 LAB rows, Altera recommends implementing data realignment (bit slip) circuitry for all the DPA channels.

Using Corner and Center Left and Right PLLs

If a differential bank is being driven by two left and right PLLs, where the corner left and right PLL is driving one group and the center left and right PLL is driving another group, there must be at least one row of separation between the two groups of DPA-enabled channels (refer to Figure 8–31). The two groups can operate at independent frequencies.

You do not need a separation if a single left and right PLL is driving the DPA-enabled channels as well as DPA-disabled channels.

Figure 8–31. Corner and Center Left and Right PLLs Driving DPA-Enabled Differential I/Os in the Same Bank



Using Both Center Left and Right PLLs

You can use both center left and right PLLs to drive DPA-enabled channels simultaneously, as long as they drive these channels in their adjacent banks only, as shown in Figure 8–32.

If one of the center left and right PLLs drives the top and bottom banks, you cannot use the other center left and right PLL to drive differential channels, as shown in Figure 8–32.

If the top PLL_L2 and PLL_R2 drives DPA-enabled channels in the lower differential bank, the PLL_L3 and PLL_R3 cannot drive DPA-enabled channels in the upper differential banks and vice versa. In other words, the center left and right PLLs cannot drive cross-banks simultaneously, as shown in Figure 8–33.

Figure 8–32. Center Left and Right PLLs Driving DPA-Enabled Differential I/Os

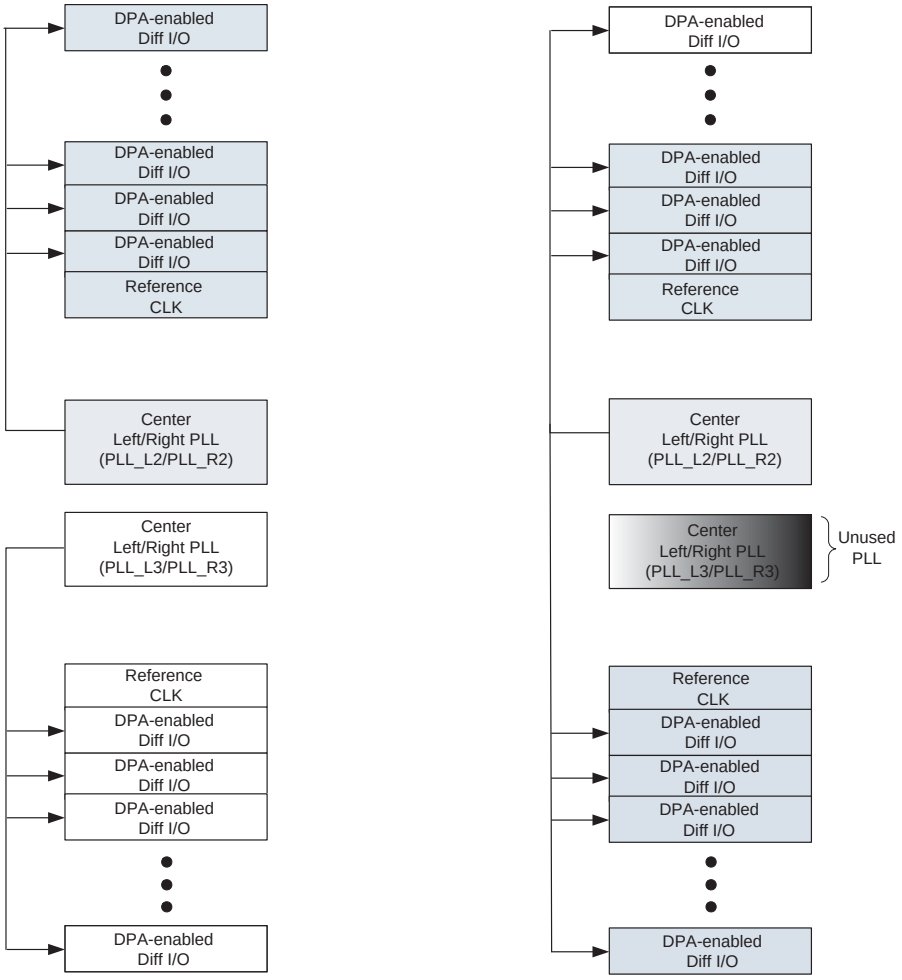
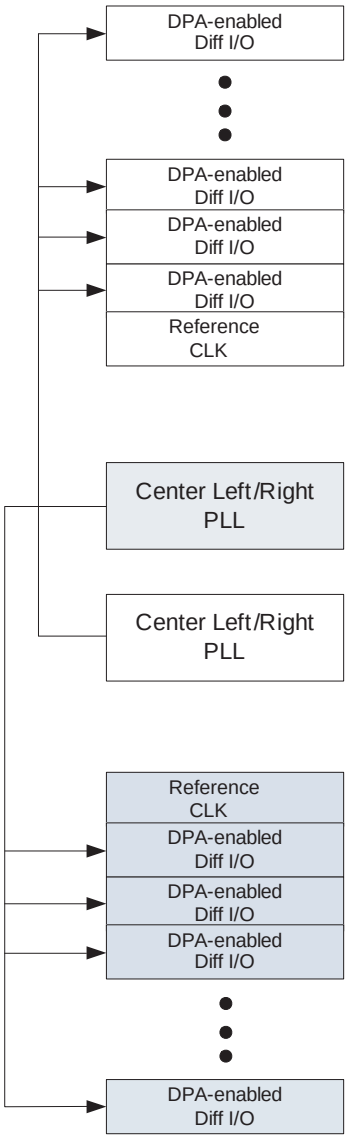


Figure 8–33. Invalid Placement of DPA-Enabled Differential I/Os Driven by Both Center Left and Right PLLs



Guidelines for DPA-Disabled Differential Channels

When you use DPA-disabled channels in the left and right banks of a Stratix IV device, you must adhere to the guidelines in the following sections.

DPA-Disabled Channels and Single-Ended I/Os

The placement rules for DPA-disabled channels and single-ended I/Os are the same as those for DPA-enabled channels and single-ended I/Os.

DPA-Disabled Channel Driving Distance

Each left and right PLL can drive all the DPA-disabled channels in the entire bank.

Using Corner and Center Left and Right PLLs

You can use a corner left and right PLL to drive all transmitter channels and a center left and right PLL to drive all DPA-disabled receiver channels within the same differential bank. In other words, a transmitter channel and a receiver channel in the same LAB row can be driven by two different PLLs, as shown in Figure 8–34.

A corner left and right PLL and a center left and right PLL can drive duplex channels in the same differential bank, as long as the channels driven by each PLL are not interleaved. Separation is not necessary between the group of channels driven by the corner and center left and right PLLs, as shown in Figure 8–34 and Figure 8–35.

Figure 8–34. Corner and Center Left and Right PLLs Driving DPA-Disabled Differential I/Os in the Same Bank

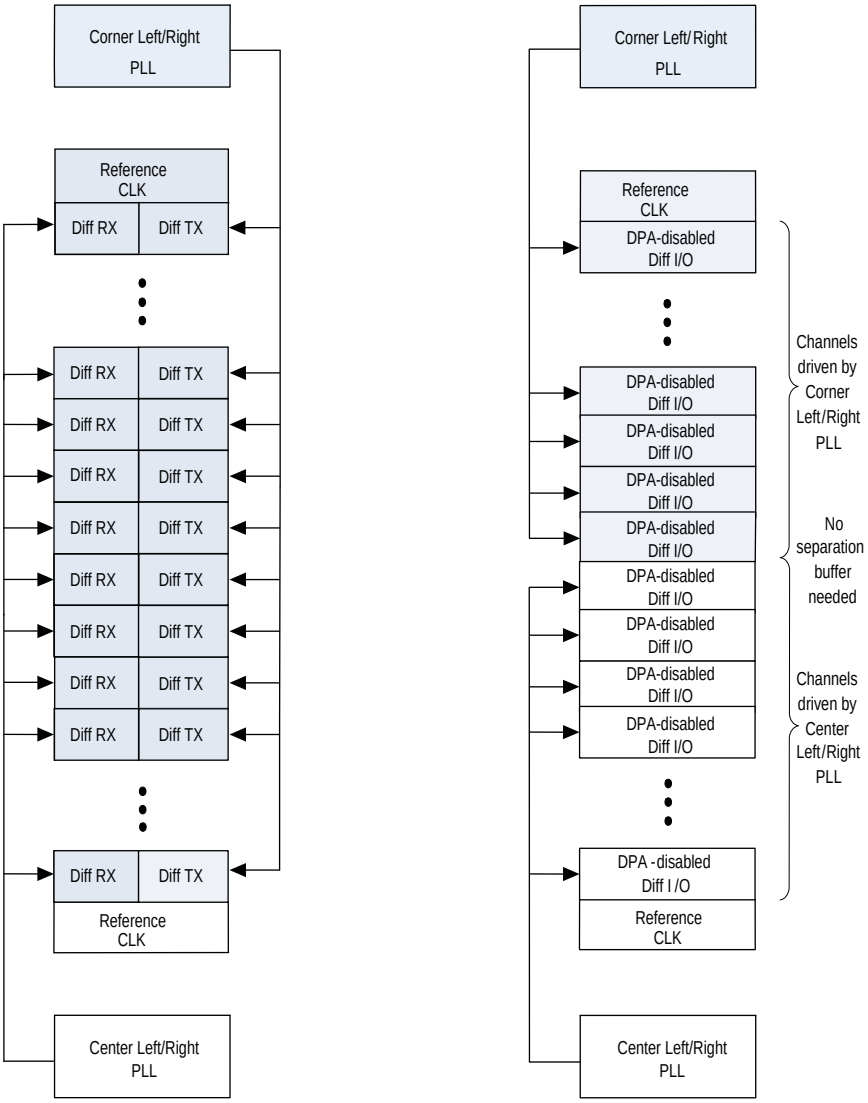
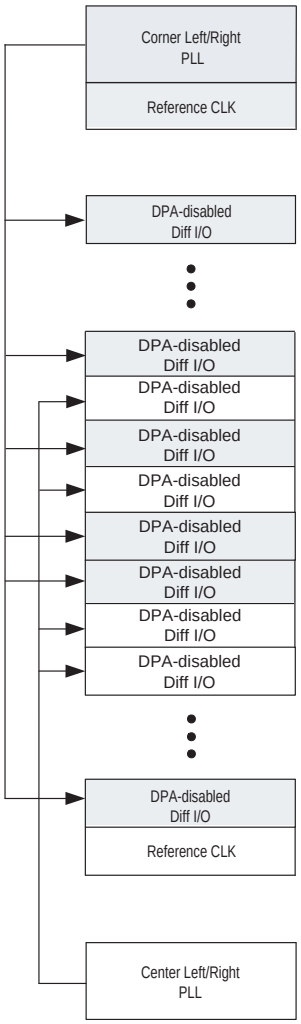


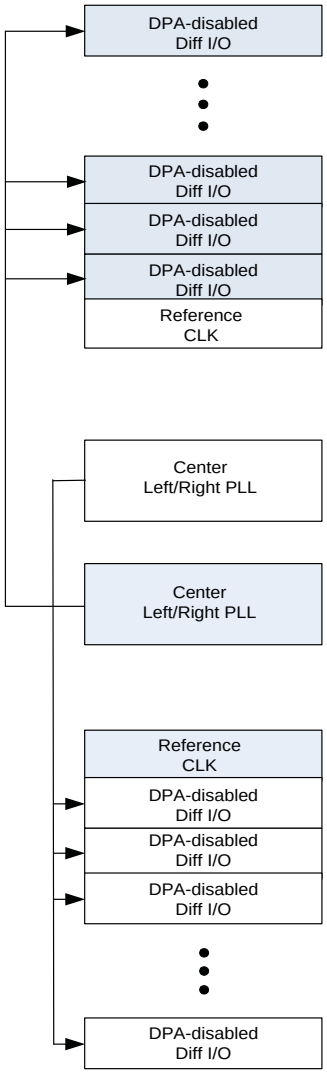
Figure 8–35. Invalid Placement of DPA-Disabled Differential I/Os Due to Interleaving of Channels Driven by the Corner and Center Left and Right PLLs



Using Both Center Left and Right PLLs

You can use both center left and right PLLs simultaneously to drive DPA-disabled channels on upper and lower differential banks. Unlike DPA-enabled channels, the center left and right PLLs can drive cross-banks. For example, the upper-center left and right PLL can drive the lower differential bank at the same time the lower center left and right PLL is driving the upper differential bank, and vice versa, as shown in Figure 8–36.

Figure 8–36. Both Center Left and Right PLLs Driving Cross-Bank DPA-Disabled Channels Simultaneously



Chapter Revision History

Table 8–12 lists the revision history for this chapter.

Table 8–12. Chapter Revision History

Date and Document Version	Changes Made	Changes Made
November 2009 v3.0	■ Changed "dedicated LVDS" to "true LVDS". ■ Removed EP4SE110, EP4SE290, and EP4SE680 devices. ■ Added EP4SE820 and Stratix IV GT devices. ■ Updated "LVDS Channels", "Differential Transmitter", "Soft-CDR Mode", and "DPA-Enabled Channels and Single-Ended I/Os" sections. ■ Updated Table 8–1, Table 8–2, Table 8–5, and Table 8–6. ■ Added Table 8–3 and Table 8–4. ■ Updated Example 8–1. ■ Updated Figure 8–22. ■ Minor text edits.	—
June 2009 v2.3	■ Added an introductory paragraph to increase search ability. ■ Minor text edits.	—
April 2009 v2.2	■ Updated "Introduction". ■ Updated Figure 8–3. ■ Removed Table 8-5 and Table 8-6.	—
March 2009 v2.1	■ Updated "Introduction", "Stratix IV LVDS Channels", "Stratix IV Differential Transmitter", "Differential I/O Termination", and "Dynamic Phase Alignment (DPA) Block" sections. ■ Updated Table 8–1, Table 8–2, Table 8–3, Table 8–4, and Table 8–7. ■ Added Table 8–5 and Table 8–6. ■ Updated Figure 8–2. ■ Removed "Referenced Documents" section.	—
November 2008 v2.0	■ Updated Figure 8–2, Figure 8–3, Figure 8–21, Figure 8–34. ■ Removed Figure 8–31. ■ Updated Table 8–1, Table 8–10. ■ Updated "Differential Pin Placement Guidelines" section.	—
May 2008 v1.0	Initial Release.	—

Section III. System Integration

This section includes the following chapters:

- Chapter 9, Hot Socketing and Power-On Reset in Stratix IV Devices
- Chapter 10, Configuration, Design Security, and Remote System Upgrades in Stratix IV Devices
- Chapter 11, SEU Mitigation in Stratix IV Devices
- Chapter 12, JTAG Boundary-Scan Testing in Stratix IV Devices
- Chapter 13, Power Management in Stratix IV Devices

Revision History

Refer to each chapter for its own specific revision history. For information on when each chapter was updated, refer to the Chapter Revision Dates section, which appears in the full handbook.

9. Hot Socketing and Power-On Reset in Stratix IV Devices

SIV51009-3.0

This chapter describes information about hot-socketing specifications, power-on reset (POR) requirements, and their implementation in Stratix® IV devices.

Stratix IV devices offer hot socketing, also known as hot plug-in or hot swap, and power sequencing support without the use of external devices. You can insert or remove a Stratix IV device or a board in a system during system operation without causing undesirable effects to the running system bus or board that is inserted into the system.

The hot-socketing feature also removes some of the difficulty when you use Stratix IV devices on PCBs that contain a mixture of 3.0-, 2.5-, 1.8-, 1.5-, and 1.2-V devices. With the Stratix IV hot-socketing feature, you no longer have to ensure a proper power-up sequence for each device on the board.

The Stratix IV hot-socketing feature provides:

- Board or device insertion and removal without external components or board manipulation
- Support for any power-up sequence
- I/O buffers non-intrusive to system buses during hot insertion

This section also describes POR circuitry in Stratix IV devices. POR circuitry keeps the devices in the reset state until the power supply outputs are within operating range.

This chapter contains the following sections:

- "Stratix IV Hot-Socketing Specifications"
- "Hot-Socketing Feature Implementation in Stratix IV Devices" on page 9–2
- "Power-On Reset Circuitry" on page 9–4
- "Power-On Reset Specifications" on page 9–5

Stratix IV Hot-Socketing Specifications

Stratix IV devices are hot-socketing compliant without the need for external components or special design requirements. Hot-socketing support in Stratix IV devices has the following advantages:

- You can drive the device before power-up without damaging it.
- I/O pins remain tri-stated during power-up. The device does not drive out before or during power-up, thereby not affecting other buses in operation.
- You can insert a Stratix IV device into or remove it from a powered-up system board without damaging the system board or interfering with its operation.

Stratix IV Devices can be Driven Before Power Up

You can drive signals into I/O pins, dedicated input pins, and dedicated clock pins of Stratix IV devices before or during power up or power down without damaging the device. Stratix IV devices support power up or power down of the V_{CCIO}, V_{CC}, V_{CCPGM}, and V_{CCPD} power supplies in any sequence in order to simplify system-level design.

I/O Pins Remain Tri-Stated During Power Up

A device that does not support hot socketing can interrupt system operation or cause contention by driving out before or during power up. In a hot-socketing situation, the Stratix IV device's output buffers are turned off during system power up or power down. Also, the Stratix IV device does not drive out until the device is configured and working within the recommended operating conditions.

Insertion or Removal of a Stratix IV Device from a Powered-Up System

Devices that do not support hot socketing can short power supplies when powered up through the device signal pins. This irregular power up can damage both the driving and driven devices and can disrupt card power up.

You can insert a Stratix IV device into or remove it from a powered-up system board without damaging the system board or interfering with its operation.

You can power up or power down the V_{CCIO}, V_{CC}, V_{CCPGM}, and V_{CCPD} supplies in any sequence. Individual power supply ramp-up and ramp-down rates range from 50 µs to 100 ms. During hot socketing, the I/O pin capacitance is less than 15 pF and the clock pin capacitance is less than 20 pF.

For more information about the hot-socketing specification, refer to the *DC and Switching Characteristics* and the *Hot-Socketing and Power Sequencing Feature and Testing for Altera Devices* white paper.

A possible concern regarding hot socketing is the potential for "latch-up." Nevertheless, Stratix IV devices are immune to latch-up when hot socketing. Latch-up can occur when electrical subsystems are hot socketed into an active system. During hot socketing, the signal pins can be connected and driven by the active system before the power supply can provide current to the device's power and ground planes. This condition can lead to latch-up and cause a low-impedance path from power to ground within the device. As a result, the device draws a large amount of current, possibly causing electrical damage.

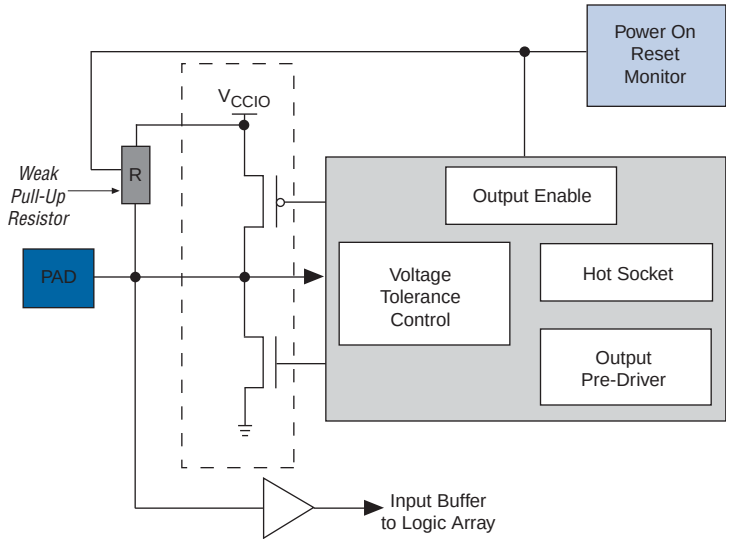
Hot-Socketing Feature Implementation in Stratix IV Devices

The hot-socketing feature turns off the output buffer during power up and power down of the V_{CC}, V_{CCIO}, V_{CCPGM}, or V_{CCPD} power supplies. The hot-socketing circuitry generates an internal `HOTSCKT` signal when the V_{CC}, V_{CCIO}, V_{CCPGM}, or V_{CCPD} power supplies are below the threshold voltage. Hot-socketing circuitry is designed to prevent excess I/O leakage during power up. When the voltage ramps up very slowly, it is still relatively low, even after the POR signal is released and the configuration is completed. The `CONF_DONE`, `nCEO`, and `nSTATUS` pins fail to

respond, as the output buffer cannot flip from the state set by the hot-socketing circuit at this low voltage. Therefore, the hot-socketing circuitry has been removed from these configuration pins to make sure that they are able to operate during configuration. Thus, it is expected behavior for these pins to drive out during power-up and power-down sequences.

Figure 9–1 shows the Stratix IV device's I/O pin circuitry.

Figure 9–1. Hot-Socketing Circuitry for Stratix IV Devices



The POR circuit monitors the voltage level of the power supplies (V_{CC}, V_{CCPT}, V_{CCPD}, and V_{CCPGM}) and keeps the I/O pins tri-stated until the device is in user mode. The weak pull-up resistor (R) in the Stratix IV input/output element (IOE) keeps the I/O pins from floating. The 3.0-V tolerance control circuit permits the I/O pins to be driven by 3.0 V before the V_{CCIO}, V_{CC}, V_{CCPD}, and/or V_{CCPGM} supplies are powered. It also prevents the I/O pins from driving out when the device is not in user mode.

Altera uses GND as a reference for hot-socketing operations and I/O buffer designs. To ensure proper operation, you must connect the GND between boards before connecting the power supplies. This prevents the GND on your board from being pulled up inadvertently by a path to power through other components on your board. A pulled up GND could otherwise cause an out-of-specification I/O voltage or current condition with the Altera device.

Power-On Reset Circuitry

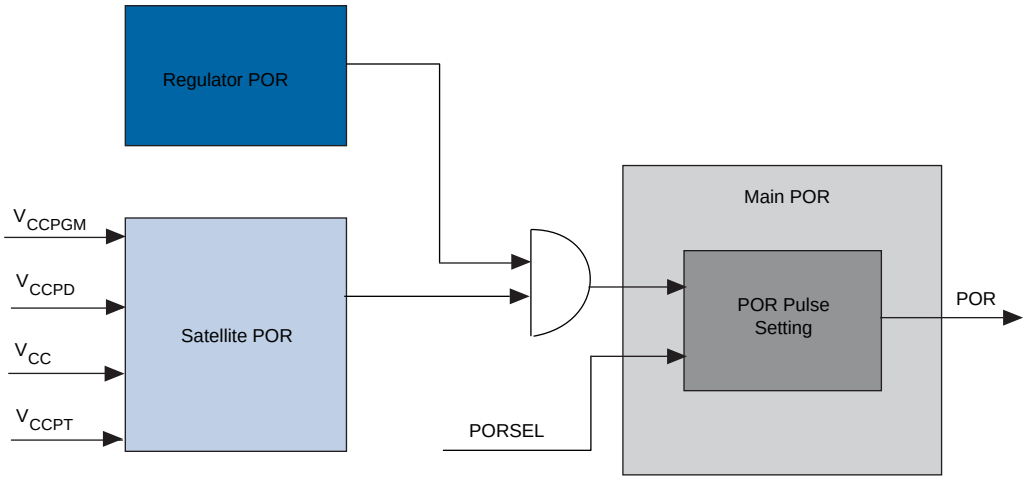
When power is applied to a Stratix IV device, a POR event occurs if the power supply reaches the recommended operating range within the maximum power supply ramp time (t_{RAMP}). If t_{RAMP} is not met, the device I/O pins and programming registers remain tri-stated, during which device configuration could fail. The maximum t_{RAMP} for Stratix IV devices is 100 ms; the minimum t_{RAMP} is 50 µs. Stratix IV devices provide a dedicated input pin (PORSEL) to select a POR delay time during power up. When the PORSEL pin is connected to GND, the POR delay time is 100 to 300 ms. When the PORSEL pin is set to high, the POR delay time is 4 to 12 ms.

The POR block consists of a regulator POR, satellite POR, and main POR to check the power supply levels for proper device configuration. The satellite POR monitors the V_{CCPD} and V_{CCPGM} power supplies that are used in the I/O buffers and for device programming. The satellite POR also monitors the V_{CC} and V_{CCPT} power supplies that are used in the device core. The POR block also checks for functionality of I/O level shifters powered by V_{CCPD} and V_{CCPGM} during power-up mode. The main POR waits for satellite POR and the regulator POR to release the POR signal. Until the release of the POR signal, the device configuration cannot start.

The internal configuration memory supply that is used during device configuration is checked by the regulator POR block and is gated in the main POR block for the final POR trip. Figure 9–2 shows a simplified diagram of the POR block.

All configuration-related dedicated and dual function I/O pins must be powered by V_{CCPGM}.

Figure 9–2. Simplified POR Diagram for Stratix IV Devices



Power-On Reset Specifications

The POR circuit monitors the power supplies listed in Table 9–1.

Table 9–1. Power Supplies Monitored by the POR Circuitry

Power Supply	Description	Setting (V)
V_{CC}	Core and periphery power supply	0.9
V_{CCPT}	Programmable power technology power supply	1.5
V_{CCPD}	I/O pre-driver power supply	2.5, 3.0
V_{CCPGM}	Configuration pins power supply	1.8, 2.5, 3.0

The POR circuit does not monitor the power supplies listed in Table 9–2.

Table 9–2. Power Supplies Not Monitored by the POR Circuitry

Power Supply	Description	Setting (V)
V_{CCIO}	I/O power supply	1.2, 1.5, 1.8, 2.5, 3.0
V_{CCA_PLL}	PLL analog global power supply	2.5
V_{CCD_PLL}	PLL digital power supply	0.9
V_{CC_CLKIN}	PLL differential clock input power supply (top and bottom I/O banks only)	2.5
V_{CCBAT}	Battery back-up power supply for design security volatile key storage	3.0

V_{CCIO}, V_{CCA_PLL}, V_{CCD_PLL}, V_{CC_CLKIN}, and V_{CCBAT} are not monitored by POR and have no affect on the device configuration.

The POR specification is designed to ensure that all the circuits in the Stratix IV device are at certain known states during power up.

The POR signal pulse width is programmable using the PORSEL input pin. When PORSEL is set to low, the POR signal pulse width is set to **100 ms**. When the PORSEL is set to high, the POR signal pulse width is set to **12 ms**.

For more information about the POR specification, refer to the *DC and Switching Characteristics*.

Document Revision History

Table 9–3 shows the revision history for this chapter.

Table 9–3. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
November 2009 v3.0	■ Updated graphics. ■ Minor text edits.	—
June 2009 v2.2	■ Updated Table 9–2. ■ Added introductory sentences to improve search ability. ■ Removed the Conclusion section. ■ Minor text edits.	—
March 2009 v2.1	■ Changed all “Stratix IV E” to “Stratix IV”. ■ Updated “Stratix IV Hot-Socketing Specifications” and “Hot-Socketing Feature Implementation in Stratix IV Devices” sections. ■ Updated Figure 9–2. ■ Removed “Referenced Documents” section.	—
November 2008 v2.0	■ Updated “Hot-Socketing Feature Implementation in Stratix IV Devices” on page 9–2. ■ Updated “Power-On Reset Circuitry” on page 9–4. ■ Updated Table 9–1. ■ Made minor editorial changes.	—
July 2008 v1.1	Revised “Introduction”.	—
May 2008 v1.0	Initial release.	—

10. Configuration, Design Security, and Remote System Upgrades in Stratix IV Devices

SIV51010-3.0

This chapter describes the configuration, design security, and remote system upgrades in Stratix® IV devices. Stratix IV devices provide configuration data decompression to save configuration memory space and time. They also provide a built-in design security feature that protects your designs against IP theft and tampering of your configuration files.

Stratix IV devices also offer remote system upgrade capability so that you can upgrade your system in real-time through any network. This helps to deliver feature enhancements and bug fixes and provides error detection, recovery, and status information to ensure reliable reconfiguration.

Overview

This chapter contains information about Stratix IV—supported configuration schemes, instructions about how to execute the required configuration schemes, and the necessary pin settings.

Stratix IV devices use SRAM cells to store configuration data. As SRAM is volatile, you must download configuration data to the Stratix IV device each time the device powers up. You can configure Stratix IV devices using one of four configuration schemes:

- Fast passive parallel (FPP)
- Fast active serial (AS)
- Passive serial (PS)
- Joint Test Action Group (JTAG)

All configuration schemes use either an external controller (for example, a MAX® II device or microprocessor), a configuration device, or a download cable. For more information, refer to "Configuration Features" on page 10–3.

This chapter includes the following sections:

- "Configuration Schemes" on page 10–2
- "Configuration Features" on page 10–3
- "Fast Passive Parallel Configuration" on page 10–5
- "Fast Active Serial Configuration (Serial Configuration Devices)" on page 10–14
- "Passive Serial Configuration" on page 10–23
- "JTAG Configuration" on page 10–32
- "Device Configuration Pins" on page 10–37
- "Configuration Data Decompression" on page 10–44
- "Remote System Upgrades" on page 10–46
- "Remote System Upgrade Mode" on page 10–50

- "Dedicated Remote System Upgrade Circuitry" on page 10–52
- "Quartus II Software Support" on page 10–57
- "Design Security" on page 10–58

Configuration Devices

Altera® serial configuration devices support a single-device and multi-device configuration solution for Stratix IV devices and are used in the fast AS configuration scheme. Serial configuration devices offer a low-cost, low pin-count configuration solution.

For information about serial configuration devices, refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* in volume 2 of the *Configuration Handbook*.

All minimum timing information in this chapter covers the entire Stratix IV family. Some devices may work at less than the minimum timing stated in this handbook due to process variation.

Configuration Schemes

Select the configuration scheme by driving the Stratix IV device `MSEL` pins either high or low, as shown in Table 10–1. The `MSEL` input buffers are powered by the V_{CC} power supply. Altera recommends you hardwire the `MSEL[]` pins to V_{CCPGM} and GND. The `MSEL[2..0]` pins have 5-kΩ internal pull-down resistors that are always active. During power-on reset (POR) and during reconfiguration, the `MSEL` pins must be at V_{IL} and V_{IH} levels of V_{CCPGM} voltage to be considered logic low and logic high.

To avoid problems with detecting an incorrect configuration scheme, hardwire the `MSEL[]` pins to V_{CCPGM} and GND without pull-up or pull-down resistors. Do not drive the `MSEL[]` pins by a microprocessor or another device.

Table 10–1. Stratix IV Configuration Schemes

Configuration Scheme	MSEL2	MSEL1	MSEL0
Fast passive parallel	0	0	0
Passive serial	0	1	0
Fast AS (40 MHz) *(1)*	0	1	1
Remote system upgrade fast AS (40 MHz) *(1)*	0	1	1
FPP with design security feature and/or decompression enabled *(2)*	0	0	1
JTAG-based configuration *(4)*	*(3)*	*(3)*	*(3)*

Notes to Table 10–1:

(1) Stratix IV devices only support fast AS configuration. You must use either EPCS64 or EPCS128 devices to configure a Stratix IV device in fast AS mode.

(2) These modes are only supported when using a MAX II device or a microprocessor with flash memory for configuration. In these modes, the host system must output a `DCLK` that is ×4 the data rate.

(3) Do not leave the `MSEL` pins floating, connect them to V_{CCPGM} or GND. These pins support the non-JTAG configuration scheme used in production. If you only use the JTAG configuration, connect the `MSEL` pins to GND.

(4) The JTAG-based configuration takes precedence over other configuration schemes, which means the `MSEL` pin settings are ignored. The JTAG-based configuration does not support the design security or decompression features.

Table 10–2 lists the uncompressed raw binary file (**.rbf**) configuration file sizes for Stratix IV devices.

Table 10–2. Stratix IV Uncompressed Raw Binary File (**.rbf**) Sizes *(Note 1)*

Device	Data Size (Bits)
EP4SE230	94, 600, 000
EP4SE360	128, 400, 000
EP4SE530	171, 800, 000
EP4SE820	241, 700, 000
EP4SGX70	47, 900, 000
EP4SGX110	47, 900, 000
EP4SGX180	94, 600, 000
EP4SGX230	94, 600, 000
EP4SGX290	128, 400, 000
EP4SGX360	128, 400, 000
EP4SGX530	171, 800, 000
EP4S40G2	94, 600, 000
EP4S40G5	171, 800, 000
EP4S100G2	94, 600, 000
EP4S100G3	171, 800, 000
EP4S100G4	171, 800, 000
EP4S100G5	171, 800, 000

Note to Table 10–2:

(1) These values are preliminary.

Use the data in Table 10–2 to estimate the file size before design compilation. Different configuration file formats; for example, a hexidecimal (**.hex**) or tabular text file (**.ttf**) format, have different file sizes. Refer to the Quartus® II software for the different types of configuration file and file sizes. However, for any specific version of the Quartus II software, any design targeted for the same device has the same uncompressed configuration file size. If you are using compression, the file size can vary after each compilation because the compression ratio is dependent on the design.

For more information about setting device configuration options or creating configuration files, refer to the *Device Configuration Options* and *Configuration File Formats* chapters in volume 2 of the *Configuration Handbook*.

Configuration Features

Stratix IV devices offer design security, decompression, and remote system upgrade features. Design security using configuration bitstream encryption is available in Stratix IV devices, which protects your designs. Stratix IV devices can receive a compressed configuration bitstream and decompress this data in real-time, reducing storage requirements and configuration time. You can make real-time system upgrades from remote locations of your Stratix IV designs with the remote system upgrade feature.

Table 10–3 summarizes which configuration features you can use in each configuration scheme.

Table 10–3. Stratix IV Configuration Features

Configuration Scheme	Configuration Method	Decompression	Design Security	Remote System Upgrade
FPP	MAX II device or a microprocessor with flash memory	✓ *(1)*	✓ *(1)*	—
Fast AS	Serial configuration device	✓	✓	✓
PS	MAX II device or a microprocessor with flash memory	✓	✓	—
	Download cable	✓	✓	—
JTAG	MAX II device or a microprocessor with flash memory	—	—	—
	Download cable	—	—	—

Note to Table 10–3:

(1) In these modes, the host system must send a DCLK that is ×4 the data rate.

You can also refer to the following:

- For more information about the configuration data decompression feature, refer to "Configuration Data Decompression" on page 10–44.
- For more information about the remote system upgrade feature, refer to "Remote System Upgrades" on page 10–46.
- For more information about the design security feature, refer to "Design Security" on page 10–58.

If your system already contains a common flash interface (CFI) flash memory, you can use it for Stratix IV device configuration storage as well. The MAX II parallel flash loader (PFL) feature in MAX II devices provides an efficient method to program CFI flash memory devices through the JTAG interface and provides the logic to control configuration from the flash memory device to the Stratix IV device. Both PS and FPP configuration modes are supported using this PFL feature.

For more information about PFL, refer to *AN 386: Using the MAX II Parallel Flash Loader with the Quartus II Software*.

For more information about programming Altera serial configuration devices, refer to "Programming Serial Configuration Devices" on page 10–21.

Power-On Reset Circuit

The POR circuit keeps the entire system in reset until the power supply voltage levels have stabilized on power-up. After power-up, the device does not release nSTATUS until V_{CCPT}, V_{CC}, V_{CCPD}, and V_{CCPGM} are above the device's POR trip point. On power down, brown-out occurs if the V_{CC} or V_{CCPT} ramps down below the POR trip point and if V_{CC}, V_{CCPD}, or V_{CCPGM} drops below the threshold level of the hot socket circuitry.

In Stratix IV devices, a pin-selectable option (PORSEL) is provided that allows you to select between the standard POR time or fast POR time. When PORSEL is driven low, the standard POR time is 100 ms < T_{POR} < 300 ms, which has a lower power-ramp rate. When PORSEL is driven high, the fast POR time is 4 ms < T_{POR} < 12 ms.

V_{CCPGM} Pins

Stratix IV devices have a power supply, V_{CCPGM}, for all the dedicated configuration pins and dual function pins. The supported configuration voltage is 1.8, 2.5, and 3.0 V. Stratix IV devices do not support 1.5 V configuration.

Use the V_{CCPGM} pin to power all dedicated configuration inputs, dedicated configuration outputs, dedicated configuration bidirectional pins, and some of the dual functional pins that you use for configuration. With V_{CCPGM}, the configuration input buffers do not have to share power lines with the regular I/O buffer in Stratix IV devices.

The operating voltage for the configuration input pin is independent of the I/O banks power supply V_{CCIO} during configuration. Therefore, Stratix IV devices do not need configuration voltage constraints on V_{CCIO}.

V_{CCPD} Pins

Stratix IV devices have a dedicated programming power supply, V_{CCPD}, which must be connected to 3.0 V/2.5 V to power the I/O pre-drivers and JTAG I/O pins (TCK, TMS, TDI, TDO, and TRST).

☞ V_{CCPGM} and V_{CCPD} must ramp up from 0 V to the desired voltage level within 100 ms when PORSEL is low or 4 ms when PORSEL is high. If these supplies are not ramped up within this specified time, your Stratix IV device will not configure successfully. If your system cannot ramp up the power supplies within 100 ms or 4 ms, you must hold nCONFIG low until all the power supplies are stable.

☞ V_{CCPD} must be greater than or equal to V_{CCIO} of the same bank. If V_{CCIO} of the bank is set to 3.0 V, V_{CCPD} must be powered up to 3.0 V. If the V_{CCIO} of the bank is powered to 2.5 V or lower, V_{CCPD} must be powered up to 2.5 V.

For more information about configuration pins power supply, refer to "Device Configuration Pins" on page 10–37.

Fast Passive Parallel Configuration

Fast passive parallel configuration in Stratix IV devices is designed to meet the continuously increasing demand for faster configuration times. Stratix IV devices are designed with the capability of receiving byte-wide configuration data per clock cycle.

You can perform FPP configuration of Stratix IV devices using an intelligent host, such as a MAX II device or a microprocessor.

FPP Configuration Using a MAX II Device as an External Host

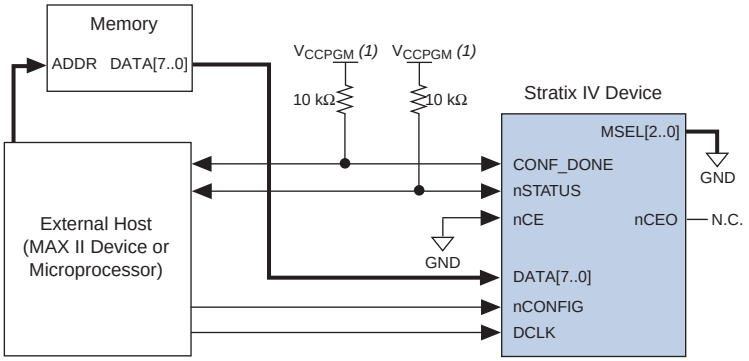
FPP configuration using compression and an external host provides the fastest method to configure Stratix IV devices. In this configuration scheme, you can use a MAX II device as an intelligent host that controls the transfer of configuration data from a storage device, such as flash memory, to the target Stratix IV device. You can store configuration data in **.rbf**, **.hex**, or **.ttf** format. When using the MAX II device as an intelligent host, a design that controls the configuration process, such as fetching the data from flash memory and sending it to the device, must be stored in the MAX II device.

If you are using the Stratix IV decompression and/or design security features, the external host must be able to send a DCLK frequency that is ×4 the data rate.

The ×4 DCLK signal does not require an additional pin and is sent on the DCLK pin. The maximum DCLK frequency is 125 MHz, which results in a maximum data rate of 250 Mbps. If you are not using the Stratix IV decompression or design security features, the data rate is ×8 the DCLK frequency.

Figure 10–1 shows the configuration interface connections between the Stratix IV device and a MAX II device for single device configuration.

Figure 10–1. Single Device FPP Configuration Using an External Host



Note to Figure 10–1:

(1) Connect the resistor to a supply that provides an acceptable input signal for the Stratix IV device. V_{CCPGM} must be high enough to meet the V_{IH} specification of the I/O on the device and the external host. Altera recommends that you power up all configuration system I/Os with V_{CCPGM}.

After power-up, the Stratix IV device goes through a POR. The POR delay depends on the PORSEL pin setting. When PORSEL is driven low, the standard POR time is 100 ms < T_{POR} < 300 ms. When PORSEL is driven high, the fast POR time is 4 ms < T_{POR} < 12 ms. During POR, the device resets, holds nSTATUS low, and tri-states all user I/O pins. After the device successfully exits POR, all user I/O pins continue to be tri-stated. If nIO_pullup is driven low during power up and configuration, the user I/O pins and dual-purpose I/O pins have weak pull-up resistors, which are on (after POR) before and during configuration. If nIO_pullup is driven high, the weak pull-up resistors are disabled.

The configuration cycle consists of three stages: reset, configuration, and initialization. While nCONFIG or nSTATUS are low, the device is in the reset stage. To initiate configuration, the MAX II device must drive the nCONFIG pin from low to high.

To begin the configuration process, you must fully power V_{CCPT}, V_{CC}, V_{CCPD}, and V_{CCPGM} of the banks where the configuration pins reside to the appropriate voltage levels.

When `nCONFIG` goes high, the device comes out of reset and releases the open-drain `nSTATUS` pin, which is then pulled high by an external 10-kΩ pull-up resistor. After `nSTATUS` is released, the device is ready to receive configuration data and the configuration stage begins. When `nSTATUS` is pulled high, the MAX II device places the configuration data one byte at a time on the `DATA[7..0]` pins.

Stratix IV devices receive configuration data on the `DATA[7..0]` pins and the clock is received on the `DCLK` pin. Data is latched into the device on the rising edge of `DCLK`. If you are using the Stratix IV decompression and/or design security features, configuration data is latched on the rising edge of every fourth `DCLK` cycle. After the configuration data is latched in, it is processed during the following three `DCLK` cycles. Therefore, you can only stop `DCLK` after three clock cycles after the last data is latched into the Stratix IV Devices.

Data is continuously clocked into the target device until `CONF_DONE` goes high. The `CONF_DONE` pin goes high one byte early in FPP modes. The last byte is required for AS and PS modes. After the device has received the next-to-last byte of the configuration data successfully, it releases the open-drain `CONF_DONE` pin, which is pulled high by an external 10-kΩ pull-up resistor. A low-to-high transition on `CONF_DONE` indicates configuration is complete and initialization of the device can begin. The `CONF_DONE` pin must have an external 10-kΩ pull-up resistor for the device to initialize.

In Stratix IV devices, the initialization clock source is either the internal oscillator or the optional `CLKUSR` pin. By default, the internal oscillator is the clock source for initialization. If you use the internal oscillator, the Stratix IV device provides itself with enough clock cycles for proper initialization. Therefore, if the internal oscillator is the initialization clock source, sending the entire configuration file to the device is sufficient to configure and initialize the device. Driving `DCLK` to the device after configuration is complete does not affect device operation.

You can also synchronize initialization of multiple devices or delay initialization with the `CLKUSR` option. You can turn on the **Enable user-supplied start-up clock** (**CLKUSR**) option in the Quartus II software from the **General** tab of the **Device and Pin Options** dialog box. Supplying a clock on `CLKUSR` does not affect the configuration process. The `CONF_DONE` pin goes high one byte early in FPP modes. The last byte is required for AS and PS modes. After the `CONF_DONE` pin transitions high, `CLKUSR` is enabled after the time specified at t_{CD2CU}. After this time period elapses, Stratix IV devices require 8,532 clock cycles to initialize properly and enter user mode. Stratix IV devices support a `CLKUSR` f_{MAX} of 125 MHz.

An optional `INIT_DONE` pin is available, which signals the end of initialization and the start of user-mode with a low-to-high transition. This **Enable INIT_DONE Output** option is available in the Quartus II software from the **General** tab of the **Device and Pin Options** dialog box. If you use the `INIT_DONE` pin, it is high because of an external 10-kΩ pull-up resistor when `nCONFIG` is low and during the beginning of configuration. After the option bit to enable `INIT_DONE` is programmed into the device (during the first frame of configuration data), the `INIT_DONE` pin goes low.

When initialization is complete, the INIT_DONE pin is released and pulled high. The MAX II device must be able to detect this low-to-high transition, which signals the device has entered user mode. When initialization is complete, the device enters user mode. In user-mode, the user I/O pins no longer have weak pull-up resistors and function as assigned in your design.

☞ Two DCLK falling edges are required after CONF_DONE goes high to begin the initialization of the device for both uncompressed and compressed bitstream in FPP.

To ensure DCLK and DATA[7..0] are not left floating at the end of configuration, the MAX II device must drive them either high or low, whichever is convenient on your board. The DATA[7..0] pins are available as user I/O pins after configuration. When you select the FPP scheme as a default in the Quartus II software, these I/O pins are tri-stated in user mode. To change this default option in the Quartus II software, select the **Dual-Purpose Pins** tab of the **Device and Pin Options** dialog box.

The configuration clock (DCLK) speed must be below the specified frequency to ensure correct configuration. No maximum DCLK period exists, which means you can pause configuration by halting DCLK for an indefinite amount of time.

☞ If you need to stop DCLK, it can only be stopped:

- three clock cycles after the last data byte was latched into the Stratix IV device when you use the decompression and/or design security features.
- two clock cycles after the last data byte was latched into the Stratix IV device when you do not use the Stratix IV decompression and/or design security features.

By stopping DCLK, the configuration circuit allows enough clock cycles to process the last byte of latched configuration data. When the clock restarts, the MAX II device must provide data on the DATA[7..0] pins prior to sending the first DCLK rising edge.

If an error occurs during configuration, the device drives its nSTATUS pin low, resetting itself internally. The low signal on the nSTATUS pin also alerts the MAX II device that there is an error. If the **Auto-restart configuration after error** option (available in the Quartus II software from the **General** tab of the **Device and Pin Options** dialog box) is turned on, the device releases nSTATUS after a reset time-out period (a maximum of 500 µs). After nSTATUS is released and pulled high by a pull-up resistor, the MAX II device can try to reconfigure the target device without needing to pulse nCONFIG low. If this option is turned off, the MAX II device must generate a low-to-high transition (with a low pulse of at least 2 µs) on nCONFIG to restart the configuration process.

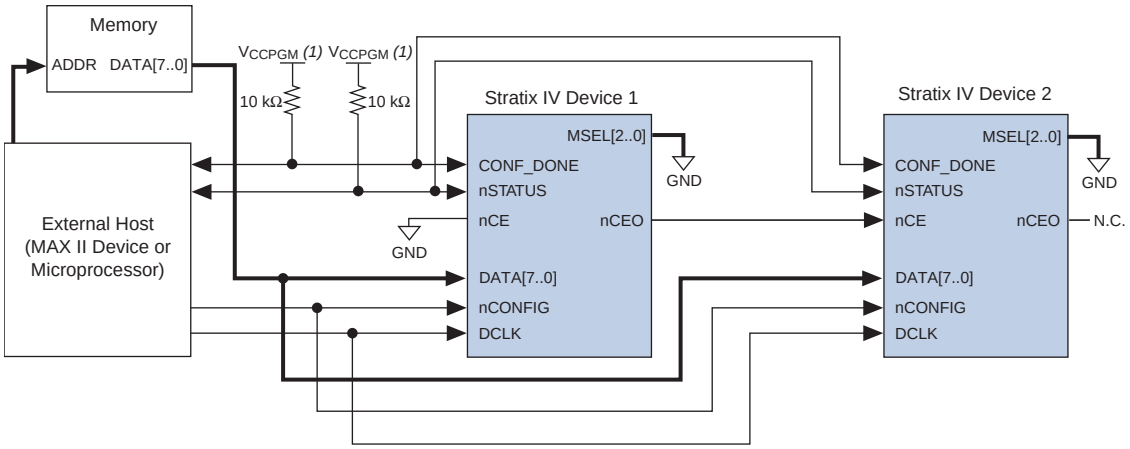
The MAX II device can also monitor the CONF_DONE and INIT_DONE pins to ensure successful configuration. The MAX II device must monitor the CONF_DONE pin to detect errors and determine when programming completes. If all the configuration data is sent, but the CONF_DONE or INIT_DONE signals have not gone high, the MAX II device reconfigures the target device.

☞ If you use the optional CLKUSR pin and nCONFIG is pulled low to restart the configuration during device initialization, ensure that CLKUSR continues toggling during the time nSTATUS is low (a maximum of 500 µs).

When the device is in user mode, initiating reconfiguration is done by transitioning the nCONFIG pin low-to-high. The nCONFIG pin must be low for at least 2 µs. When nCONFIG is pulled low, the device also pulls nSTATUS and CONF_DONE low and all I/O pins are tri-stated. After nCONFIG returns to a logic high level and nSTATUS is released by the device, reconfiguration begins.

Figure 10–2 shows how to configure multiple Stratix IV devices using a MAX II device. This circuit is similar to the FPP configuration circuit for a single device, except the devices are cascaded for multi-device configuration.

Figure 10–2. Multi-Device FPP Configuration Using an External Host



Note to Figure 10–2:

(1) Connect the pull-up resistor to a supply that provides an acceptable input signal for all Stratix IV devices in the chain. V_{CCPGM} must be high enough to meet the V_{IH} specification of the I/O standard on the device and the external host. Altera recommends you power up all configuration system I/Os with V_{CCPGM}.

In a multi-device FPP configuration, the first device's nCE pin is connected to GND while its nCEO pin is connected to nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. The second device in the chain begins configuration within one clock cycle; therefore, the transfer of data destinations is transparent to the MAX II device. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA[7..0], and CONF_DONE) are connected to every device in the chain. The configuration signals may require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered for every fourth device. Because all device CONF_DONE pins are tied together, all devices initialize and enter user mode at the same time.

All nSTATUS and CONF_DONE pins are tied together; if any device detects an error, configuration stops for the entire chain and you must reconfigure the entire chain. For example, if the first device flags an error on nSTATUS, it resets the chain by pulling its nSTATUS pin low. This behavior is similar to a single device detecting an error.

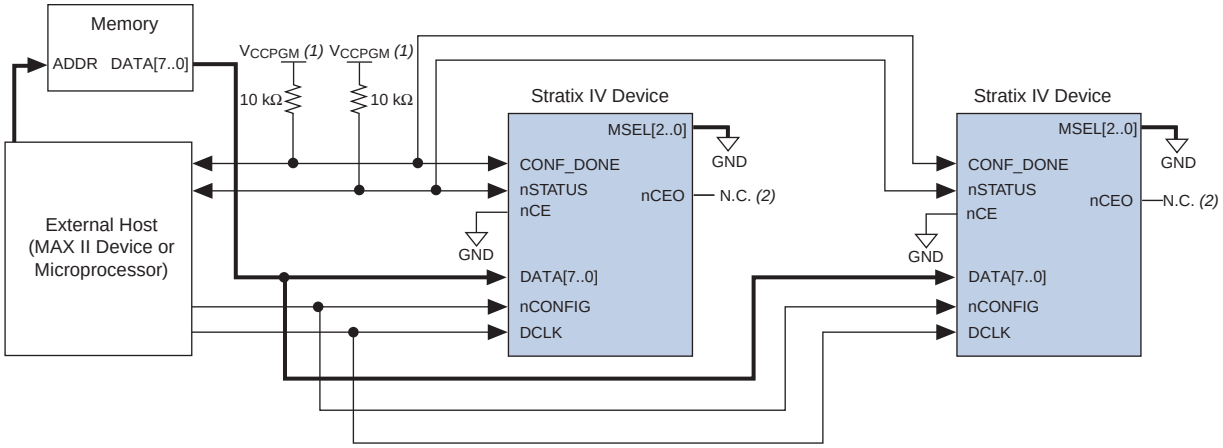
If the **Auto-restart configuration after error** option is turned on, the devices release their nSTATUS pins after a reset time-out period (a maximum of 500 µs). After all nSTATUS pins are released and pulled high, the MAX II device tries to reconfigure the chain without pulsing nCONFIG low. If this option is turned off, the MAX II device must generate a low-to-high transition (with a low pulse of at least 2 µs) on nCONFIG to restart the configuration process.

In a multi-device FPP configuration chain, all Stratix IV devices in the chain must either enable or disable the decompression and/or design security features. You cannot selectively enable the decompression and/or design security features for each device in the chain because of the DATA and DCLK relationship. If the chain contains devices that do not support design security, use a serial configuration scheme.

If a system has multiple devices that contain the same configuration data, tie all device nCE inputs to GND and leave the nCEO pins floating. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA[7..0], and CONF_DONE) are connected to every device in the chain. Configuration signals may require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered for every fourth device. Devices must be the same density and package. All devices start and complete configuration at the same time.

Figure 10–3 shows a multi-device FPP configuration when both Stratix IV devices are receiving the same configuration data.

Figure 10–3. Multiple-Device FPP Configuration Using an External Host When Both Devices Receive the Same Data



Notes to Figure 10–3:

(1) Connect the resistor to a supply that provides an acceptable input signal for all Stratix IV devices in the chain. V_{CCPGM} must be high enough to meet the V_{IH} specification of the I/O on the device and the external host. Altera recommends you power up all configuration system I/Os with V_{CCPGM}.

(2) The nCEO pins of both Stratix IV devices are left unconnected when configuring the same configuration data into multiple devices.

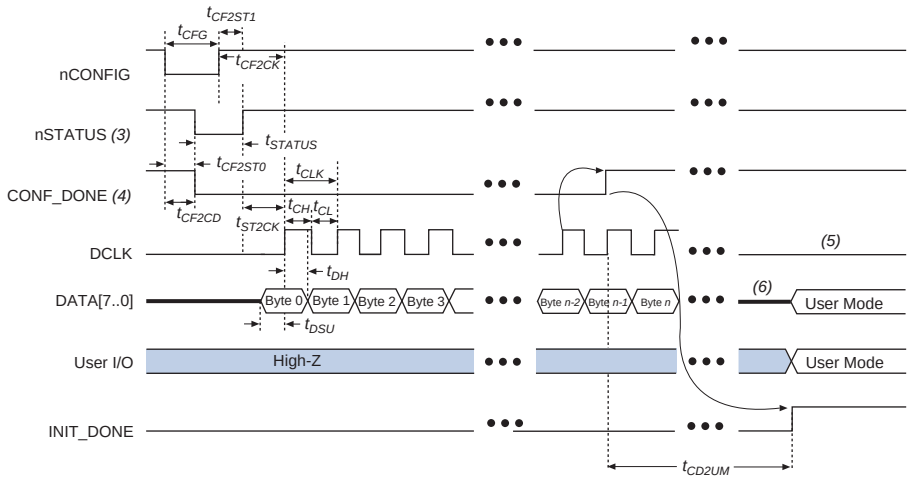
You can use a single configuration chain to configure Stratix IV devices with other Altera devices that support FPP configuration, such as other types of Stratix devices. To ensure that all devices in the chain complete configuration at the same time, or that an error flagged by one device initiates reconfiguration in all devices, tie all of the device CONF_DONE and nSTATUS pins together.

For more information about configuring multiple Altera devices in the same configuration chain, refer to the *Configuring Mixed Altera FPGA Chains* in volume 2 of the *Configuration Handbook*.

FPP Configuration Timing

Figure 10–4 shows the timing waveform for FPP configuration when using a MAX II device as an external host. This waveform shows the timing when you have not enabled the decompression and design security features.

Figure 10–4. FPP Configuration Timing Waveform *(Note 1)*, *(2)*



Notes to Figure 10–4:

(1) Use this timing waveform when you have not enabled the decompression and design security features.

(2) The beginning of this waveform shows the device in user mode. In user mode, `nCONFIG`, `nSTATUS`, and `CONF_DONE` are at logic high levels. When `nCONFIG` is pulled low, a reconfiguration cycle begins.

(3) After power-up, the Stratix IV device holds `nSTATUS` low for the time of the POR delay.

(4) After power-up, before and during configuration, `CONF_DONE` is low.

(5) Do not leave `DCLK` floating after configuration. You can drive it high or low, whichever is more convenient.

(6) `DATA[7..0]` are available as user I/O pins after configuration except for some exceptions on Stratix IV GT. The state of these pins depends on the dual-purpose pin settings.

Table 10–4 lists the timing parameters for Stratix IV devices for FPP configuration when you have not enabled the decompression and design security features.

Table 10–4. FPP Timing Parameters for Stratix IV Devices (Part 1 of 2) *(Note 1)*, *(2)*

Symbol	Parameter	Minimum	Maximum	Units
t_{CF2CD}	`nCONFIG` low to `CONF_DONE` low	—	800	ns
t_{CF2ST0}	`nCONFIG` low to `nSTATUS` low	—	800	ns
t_{CFG}	`nCONFIG` low pulse width	2	—	µs
t_{STATUS}	`nSTATUS` low pulse width	10	500 *(3)*	µs
t_{CF2ST1}	`nCONFIG` high to `nSTATUS` high	—	500 *(3)*	µs
t_{CF2CK}	`nCONFIG` high to first rising edge on `DCLK`	500	—	µs
t_{ST2CK}	`nSTATUS` high to first rising edge of `DCLK`	2	—	µs
t_{DSU}	Data setup time before rising edge on `DCLK`	4	—	ns

Table 10–4. FPP Timing Parameters for Stratix IV Devices (Part 2 of 2) *(Note 1), (2)*

Symbol	Parameter	Minimum	Maximum	Units
t_{DH}	Data hold time after rising edge on DCLK	0	—	ns
t_{CH}	DCLK high time *(5)*	3.2	—	ns
t_{CL}	DCLK low time *(5)*	3.2	—	ns
t_{CLK}	DCLK period *(5)*	8	—	ns
f_{MAX}	DCLK frequency	—	125	MHz
t_R	Input rise time	—	40	ns
t	Input fall time	—	40	ns
t_{CD2UM}	CONF_DONE high to user mode *(4)*	55	150	µs
t_{CD2CU}	CONF_DONE high to CLKUSR enabled	4 × maximum DCLK period	—	—
t_{CD2UMC}	CONF_DONE high to user mode with CLKUSR option on	t_{CD2CU} + (8532 × CLKUSR period)	—	—

Notes to Table 10–4:

(1) This information is preliminary.

(2) Use these timing parameters when you have not enabled the decompression and design security features.

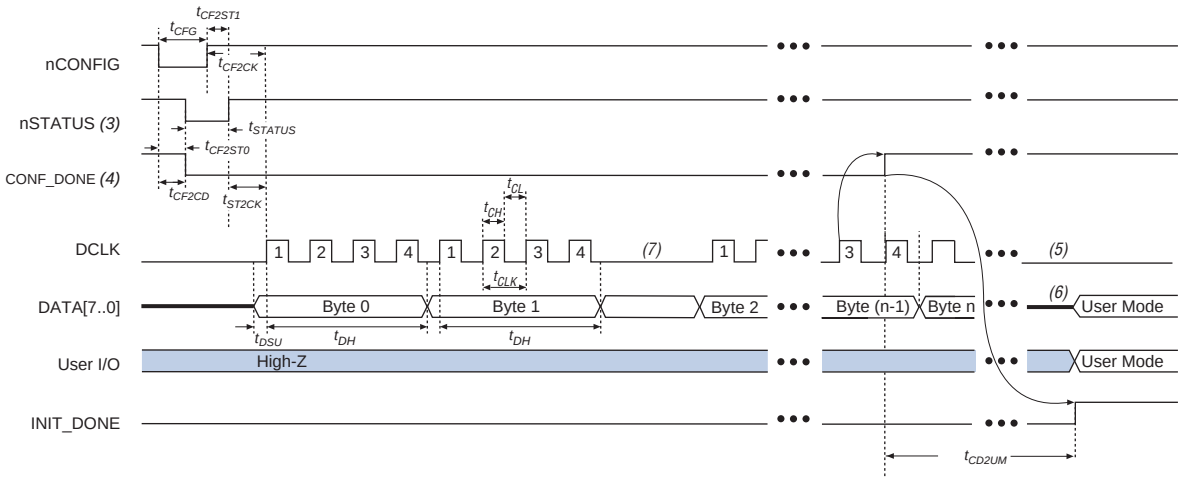
(3) You can obtain this value if you do not delay the configuration by extending the nCONFIG or nSTATUS low pulse width.

(4) The minimum and maximum numbers apply only if you chose the internal oscillator as the clock source for starting the device.

(5) Adding up t_{CH} and t_{CL} equals to t_{CLK}. When t_{CH} is 3.2 ns (min), t_{CL} must be 4.8 ns and vice versa.

Figure 10–5 shows the timing waveform for FPP configuration when using a MAX II device as an external host. This waveform shows the timing when you have enabled the decompression and/or design security features.

Figure 10–5. FPP Configuration Timing Waveform with Decompression or Design Security Feature Enabled *(Note 1)*, *(2)*



Notes to Figure 10–5:

(1) Use this timing waveform when you have enabled the decompression and/or design security features.

(2) The beginning of this waveform shows the device in user-mode. In user-mode, `nCONFIG`, `nSTATUS`, and `CONF_DONE` are at logic high levels. When `nCONFIG` is pulled low, a reconfiguration cycle begins.

(3) After power-up, the Stratix IV device holds `nSTATUS` low for the time of the POR delay.

(4) After power-up, before and during configuration, `CONF_DONE` is low.

(5) Do not leave `DCLK` floating after configuration. You can drive it high or low, whichever is more convenient.

(6) `DATA[7..0]` are available as user I/O pins after configuration except for some exceptions on Stratix IV GT. The state of these pins depends on the dual-purpose pin settings.

(7) If needed, you can pause `DCLK` by holding it low. When `DCLK` restarts, the external host must provide data on the `DATA[7..0]` pins prior to sending the first `DCLK` rising edge.

Table 10–5 lists the timing parameters for Stratix IV devices for FPP configuration when you enable the decompression and/or the design security features.

Table 10–5. FPP Timing Parameters for Stratix IV Devices with the Decompression and/or Design Security Features Enabled *(Note 1)*, *(2)* (Part 1 of 2)

Symbol	Parameter	Minimum	Maximum	Units
t_{CF2CD}	`nCONFIG` low to `CONF_DONE` low	—	800	ns
t_{CF2ST0}	`nCONFIG` low to `nSTATUS` low	—	800	ns
t_{CFG}	`nCONFIG` low pulse width	2	—	µs
t_{STATUS}	`nSTATUS` low pulse width	10	500 *(3)*	µs
t_{CF2ST1}	`nCONFIG` high to `nSTATUS` high	—	500 *(3)*	µs
t_{CF2CK}	`nCONFIG` high to first rising edge on `DCLK`	500	—	µs
t_{ST2CK}	`nSTATUS` high to first rising edge of `DCLK`	2	—	µs
t_{DSU}	Data setup time before rising edge on `DCLK`	4	—	ns
t_{DH}	Data hold time after rising edge on `DCLK`	24	—	ns
t_{CH}	`DCLK` high time *(5)*	3.2	—	ns

Table 10–5. FPP Timing Parameters for Stratix IV Devices with the Decompression and/or Design Security Features Enabled *(Note 1)*, *(2)* (Part 2 of 2)

Symbol	Parameter	Minimum	Maximum	Units
t_{CL}	DCLK low time *(5)*	3.2	—	ns
t_{CLK}	DCLK period *(5)*	8	—	ns
f_{MAX}	DCLK frequency	—	125	MHz
t_{DATA}	Data rate	—	250	Mbps
t_R	Input rise time	—	40	ns
t	Input fall time	—	40	ns
t_{CD2UM}	CONF_DONE high to user mode *(4)*	55	150	µs
t_{CD2CU}	CONF_DONE high to CLKUSR enabled	4 × maximum DCLK period	—	—
t_{CD2UMC}	CONF_DONE high to user mode with CLKUSR option on *(4)*	t_{CD2CU} + (8532 × CLKUSR period)	—	—

Notes to Table 10–5:

(1) This information is preliminary.
(2) Use these timing parameters when you use the decompression and/or design security features.
(3) You can obtain this value if you do not delay the configuration by extending the nCONFIG or nSTATUS low pulse width.
(4) The minimum and maximum numbers apply only if you chose the internal oscillator as the clock source for starting the device.
(5) Adding up t_{CH} and t_{CL} equals to t_{CLK}. When t_{CH} is 3.2 ns (min), t_{CL} must be 4.8 ns and vice versa.

For more information about device configuration options and how to create configuration files, refer to the *Device Configuration Options* and *Configuration File Formats* chapters in volume 2 of the *Configuration Handbook*.

FPP Configuration Using a Microprocessor

In this configuration scheme, a microprocessor can control the transfer of configuration data from a storage device, such as flash memory, to the target Stratix IV device.

All information in "FPP Configuration Using a MAX II Device as an External Host" on page 10–6 is also applicable when using a microprocessor as an external host. Refer to this section for all configuration and timing information.

Fast Active Serial Configuration (Serial Configuration Devices)

In the fast AS configuration scheme, Stratix IV devices are configured using a serial configuration device. These configuration devices are low-cost devices with non-volatile memory that feature a simple four-pin interface and a small form factor.

The largest serial configuration device currently supports 128 MBits of configuration bitstream. Use the Stratix IV decompression features or select an FPP or PS configuration scheme for EP4SE360, EP4SGX290, EP4S40G5, EP4S100G3 and larger devices.

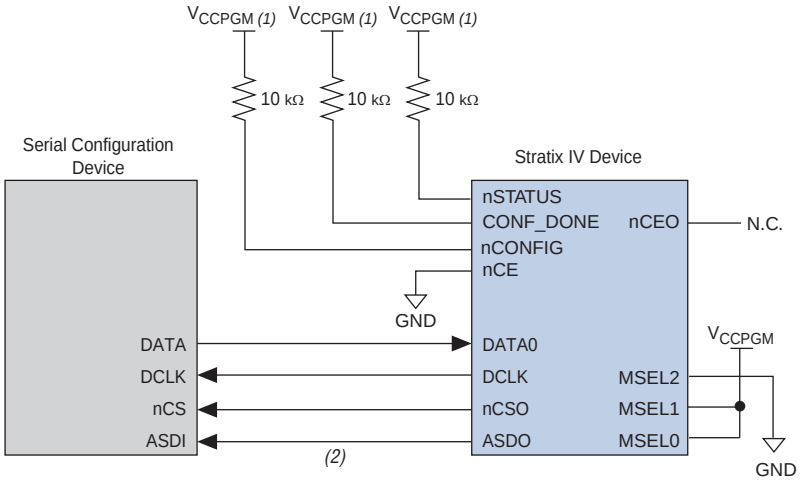
For more information about serial configuration devices, refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* chapter in volume 2 of the *Configuration Handbook*.

Serial configuration devices provide a serial interface to access configuration data. During device configuration, Stratix IV devices read configuration data using the serial interface, decompress data if necessary, and configure their SRAM cells. This scheme is referred to as the AS configuration scheme because the Stratix IV device controls the configuration interface. This scheme contrasts with the PS configuration scheme where the configuration device controls the interface.

The Stratix IV decompression and design security features are fully available when configuring your Stratix IV device using fast AS mode.

Serial configuration devices have a four-pin interface: serial clock input (`DCLK`), serial data output (`DATA`), AS data input (`ASDI`), and an active-low chip select (`nCS`). This four-pin interface connects to Stratix IV device pins, as shown in Figure 10–6.

Figure 10–6. Single Device Fast AS Configuration



Notes to Figure 10–6:

(1) Connect the pull-up resistors to V_{CCPGM} at a 3.0-V supply.

(2) Stratix IV devices use the `ASDO`-to-`ASDI` path to control the configuration device.

You can power the EPCS serial configuration device with 3.0 V when you configure the Stratix IV FPGA using Active Serial (AS) configuration mode. This is feasible because the power supply to the EPCS device ranges between 2.7 V and 3.6 V. You do not need a dedicated 3.3 V power supply to power the EPCS device. The EPCS device and the `VCCPGM` pins on the Stratix IV device may share the same 3.0 V power supply.

Upon power-up, the Stratix IV devices go through a POR. The POR delay depends on the `PORSEL` pin setting. When `PORSEL` is driven low, the standard POR time is 100 ms < T_{POR} < 300 ms. When `PORSEL` is driven high, the fast POR time is 4 ms < T_{POR} < 12 ms. During POR, the device resets, holds `nSTATUS` and `CONF_DONE` low, and tri-states all user I/O pins. After the device successfully exits POR, all the user I/O pins continue to be tri-stated. If `nIO_pullup` is driven low during power-up and configuration, the user I/O pins and dual-purpose I/O pins will have weak pull-up resistors, which are on (after POR) before and during configuration. If `nIO_pullup` is driven high, the weak pull-up resistors are disabled.

The configuration cycle consists of three stages: reset, configuration, and initialization. While nCONFIG or nSTATUS are low, the device is in reset. After POR, the Stratix IV device releases nSTATUS, which is pulled high by an external 10-kΩ pull-up resistor and enters configuration mode.

To begin configuration, power the V_{CC}, V_{CCIO}, V_{CCPGM}, and V_{CCPD} voltages (for the banks where the configuration pins reside) to the appropriate voltage levels.

The serial clock (DCLK) generated by the Stratix IV device controls the entire configuration cycle and provides timing for the serial interface. Stratix IV devices use an internal oscillator to generate DCLK. Using the MSEL[] pins, you can select to use a 40 MHz oscillator.

In fast AS configuration schemes, Stratix IV devices drive out control signals on the falling edge of DCLK. The serial configuration device responds to the instructions by driving out configuration data on the falling edge of DCLK. Then the data is latched into the Stratix IV device on the following falling edge of DCLK.

In configuration mode, Stratix IV devices enable the serial configuration device by driving the nCSO output pin low, which connects to the chip select (nCS) pin of the configuration device. The Stratix IV device uses the serial clock (DCLK) and serial data output (ASDO) pins to send operation commands and/or read address signals to the serial configuration device. The configuration device provides data on its serial data output (DATA) pin, which connects to the DATA0 input of the Stratix IV devices.

After all the configuration bits are received by the Stratix IV device, it releases the open-drain CONF_DONE pin, which is pulled high by an external 10-kΩ resistor. Initialization begins only after the CONF_DONE signal reaches a logic high level. All AS configuration pins (DATA0, DCLK, nCSO, and ASDO) have weak internal pull-up resistors that are always active. After configuration, these pins are set as input tri-stated and are driven high by the weak internal pull-up resistors. The CONF_DONE pin must have an external 10-kΩ pull-up resistor in order for the device to initialize.

In Stratix IV devices, the initialization clock source is either the internal oscillator or the optional CLKUSR pin. By default, the internal oscillator is the clock source for initialization. If you use the internal oscillator, the Stratix IV device provides itself with enough clock cycles for proper initialization. You also have the flexibility to synchronize initialization of multiple devices or to delay initialization with the CLKUSR option. You can turn on the **Enable user-supplied start-up clock** (**CLKUSR**) option in the Quartus II software from the **General** tab of the **Device and Pin Options** dialog box. When you select the **Enable user supplied start-up clock** option, the CLKUSR pin is the initialization clock source. Supplying a clock on CLKUSR does not affect the configuration process. After all configuration data is accepted and CONF_DONE goes high, CLKUSR is enabled after four clock cycles of DCLK. After this time period elapses, Stratix IV devices require 8,532 clock cycles to initialize properly and enter user mode. Stratix IV devices support a CLKUSR f_{MAX} of 125 MHz.

An optional INIT_DONE pin is available, which signals the end of initialization and the start of user-mode with a low-to-high transition. The **Enable INIT_DONE Output** option is available in the Quartus II software from the **General** tab of the **Device and Pin Options** dialog box. If you use the INIT_DONE pin, it is high due to an external 10-kΩ pull-up resistor when nCONFIG is low and during the beginning of configuration. After the option bit to enable INIT_DONE is programmed into the

device (during the first frame of configuration data), the INIT_DONE pin goes low. When initialization is complete, the INIT_DONE pin is released and pulled high. This low-to-high transition signals that the device has entered user mode. When initialization is complete, the device enters user mode. In user mode, the user I/O pins no longer have weak pull-up resistors and function as assigned in your design.

If an error occurs during configuration, Stratix IV devices assert the nSTATUS signal low, indicating a data frame error, and the CONF_DONE signal stays low. If the **Auto-restart configuration after error** option (available in the Quartus II software from the **General** tab of the **Device and Pin Options** dialog box) is turned on, the Stratix IV device resets the configuration device by pulsing nCSO, releases nSTATUS after a reset time-out period (a maximum of 500 µs), and retries configuration. If this option is turned off, the system must monitor nSTATUS for errors and then pulse nCONFIG low for at least 2 µs to restart configuration.

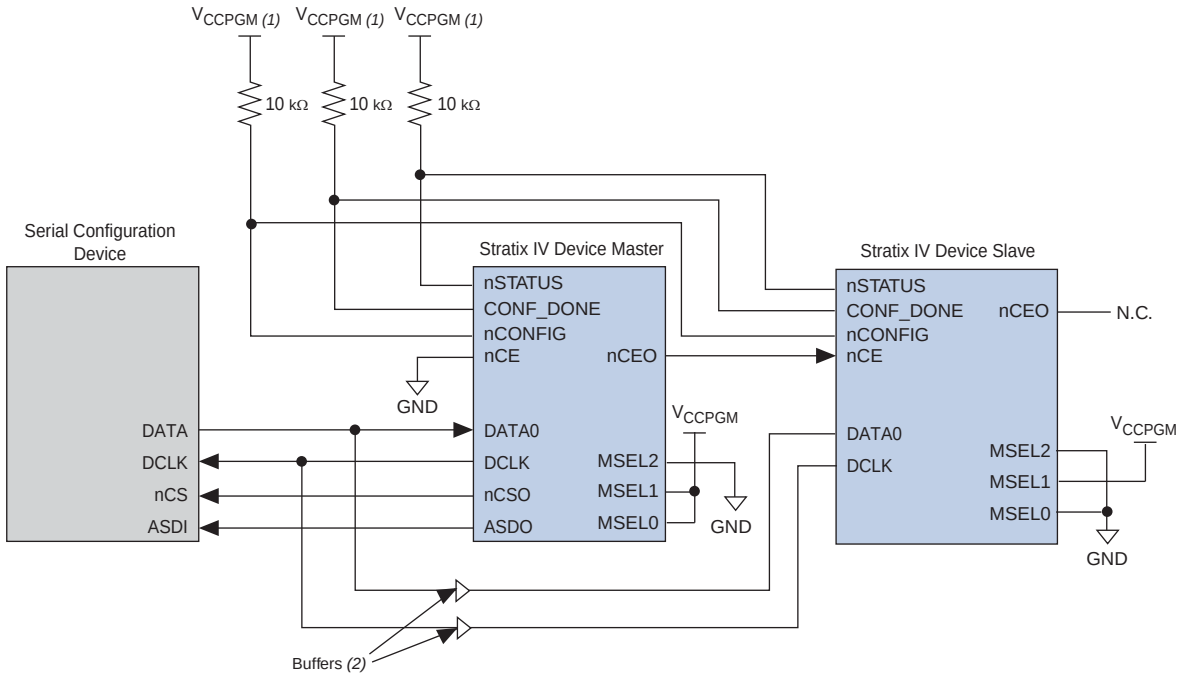
When the Stratix IV device is in user mode, you can initiate reconfiguration by pulling the nCONFIG pin low. The nCONFIG pin must be low for at least 2 µs. When nCONFIG is pulled low, the device also pulls nSTATUS and CONF_DONE low and all I/O pins are tri-stated. After nCONFIG returns to a logic high level and nSTATUS is released by the Stratix IV device, reconfiguration begins.

You can configure multiple Stratix IV devices using a single serial configuration device. You can cascade multiple Stratix IV devices using the chip-enable (nCE) and chip-enable-out (nCEO) pins. The first device in the chain must have its nCE pin connected to GND. You must connect its nCEO pin to the nCE pin of the next device in the chain. When the first device captures all of its configuration data from the bitstream, it drives the nCEO pin low, enabling the next device in the chain. You must leave the nCEO pin of the last device unconnected. The nCONFIG, nSTATUS, CONF_DONE, DCLK, and DATA0 pins of each device in the chain are connected (refer to Figure 10–7).

The first Stratix IV device in the chain is the configuration master and controls configuration of the entire chain. You must connect its MSEL pins to select the AS configuration scheme. The remaining Stratix IV devices are configuration slaves. You must connect their MSEL pins to select the PS configuration scheme. Any other Altera device that supports PS configuration can also be part of the chain as a configuration slave.

Figure 10–7 shows the pin connections for the multi-device fast AS configuration.

Figure 10–7. Multi-Device Fast AS Configuration



Notes to Figure 10–7:

(1) Connect the pull-up resistors to V_{CCPGM} at a 3.0-V supply.

(2) Connect the repeater buffers between the Stratix IV master and slave device(s) for `DATA[0]` and `DCLK`. This is to prevent potential signal integrity and clock skew problems.

As shown in Figure 10–7, the `nSTATUS` and `CONF_DONE` pins on all target devices are connected together with external pull-up resistors. These pins are open-drain bidirectional pins on the devices. When the first device asserts `nCEO` (after receiving all of its configuration data), it releases its `CONF_DONE` pin. But the subsequent devices in the chain keep this shared `CONF_DONE` line low until they have received their configuration data. When all target devices in the chain have received their configuration data and have released `CONF_DONE`, the pull-up resistor drives a high level on this line and all devices simultaneously enter initialization mode.

If an error occurs at any point during configuration, the `nSTATUS` line is driven low by the failing device. If you enable the **Auto-restart configuration after error** option, reconfiguration of the entire chain begins after a reset time-out period (a maximum of 500 µs). If you did not enable the **Auto-restart configuration after error** option, the external system must monitor `nSTATUS` for errors and then pulse `nCONFIG` low to restart configuration. The external system can pulse `nCONFIG` if it is under system control rather than tied to V_{CCGPM}.

While you can cascade Stratix IV devices, you cannot cascade or chain together serial configuration devices.

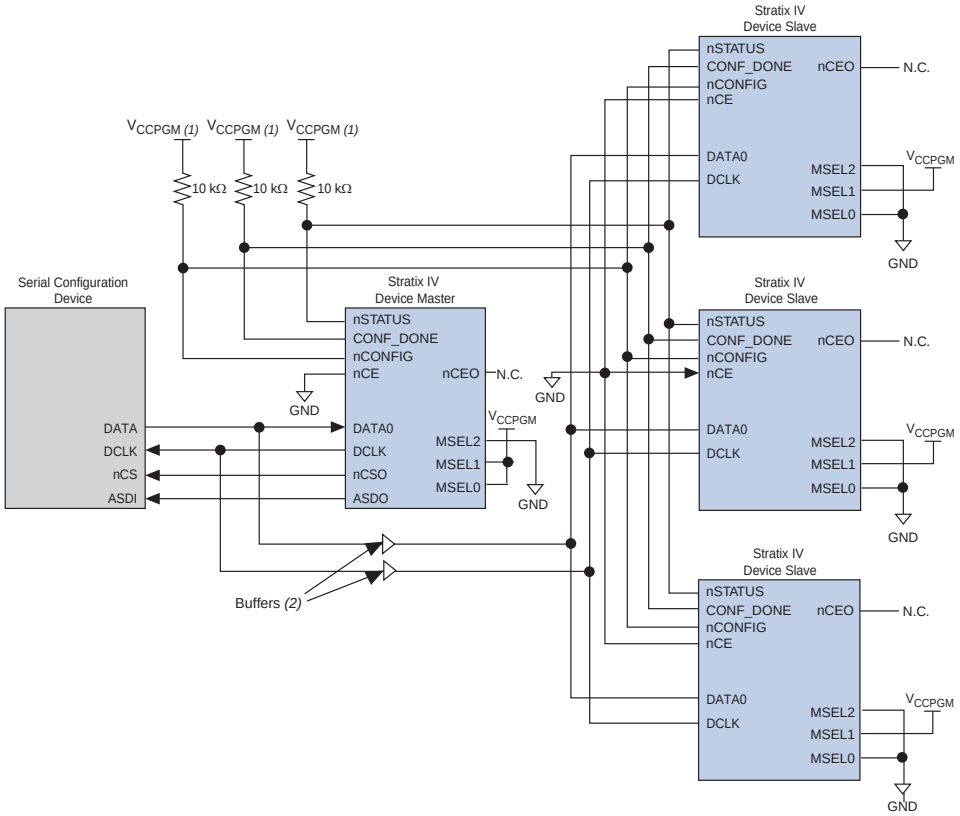
If the configuration bitstream size exceeds the capacity of a serial configuration device, you must select a larger configuration device and/or enable the compression feature. When configuring multiple devices, the size of the bitstream is the sum of the individual device configuration bitstreams.

A system may have multiple devices that contain the same configuration data. In active serial chains, you can implement this by storing one copy of the **.sof** in the serial configuration device. The same copy of the **.sof** configures the master Stratix IV device and all remaining slave devices concurrently. All Stratix IV devices must be the same density and package.

To configure four identical Stratix IV devices with the same **.sof**, you can set up the chain as shown in Figure 10–8. The first device is the master device and its `MSEL` pins must be set to select AS configuration. The other three slave devices are set up for concurrent configuration and their `MSEL` pins must be set to select PS configuration. The `nCE` input pins from the master and slave are connected to GND, and the `DATA` and `DCLK` pins connect in parallel to all four devices. During the configuration cycle, the master device reads its configuration data from the serial configuration device and transmits the configuration data to all three slave devices, configuring all of them simultaneously.

Figure 10–8 shows the multi-device fast AS configuration when the devices receive the same data using a single **.sof**.

Figure 10–8. Multi-Device Fast AS Configuration When the Devices Receive the Same Data Using a Single **.sof**



Notes to Figure 10–8:

(1) Connect the pull-up resistors to V_{CCPGM} at a 3.0-V supply.

(2) Connect the repeater buffers between the Stratix IV master and slave device(s) for `DATA[0]` and `DCLK`. This is to prevent potential signal integrity and clock skew problems.

Estimating Active Serial Configuration Time

Active serial configuration time is dominated by the time it takes to transfer data from the serial configuration device to the Stratix IV device. This serial interface is clocked by the Stratix IV `DCLK` output (generated from an internal oscillator) and must be set to **40 MHz (25 ns)**.Therefore, the minimum configuration time estimate for an EP4SE230 device (94, 600, 000 bits of uncompressed data) is:

RBF Size × (minimum `DCLK` period / 1 bit per `DCLK` cycle) = estimated minimum configuration time

94, 600, 000 bits × (25 ns / 1 bit) = 2365 ms

☞ The calculation above is based on a preliminary uncompressed **.rbf** size. The final **.rbf** size will be available after the Quartus II software is able to generate the **.rbf**.

Enabling compression reduces the amount of configuration data that is transmitted to the Stratix IV device, which also reduces configuration time. On average, compression reduces configuration time, depending on the design.

Programming Serial Configuration Devices

Serial configuration devices are non-volatile, flash-memory-based devices. You can program these devices in-system using the USB-Blaster™, EthernetBlaster™, or ByteBlaster™ II download cable. Alternatively, you can program them using the Altera programming unit (APU), supported third-party programmers, or a microprocessor with the SRunner software driver.

You can perform in-system programming of serial configuration devices using the conventional AS programming interface or the JTAG interface solution.

Because serial configuration devices do not support the JTAG interface, the conventional method to program them is using the AS programming interface. The configuration data used to program serial configuration devices is downloaded using programming hardware.

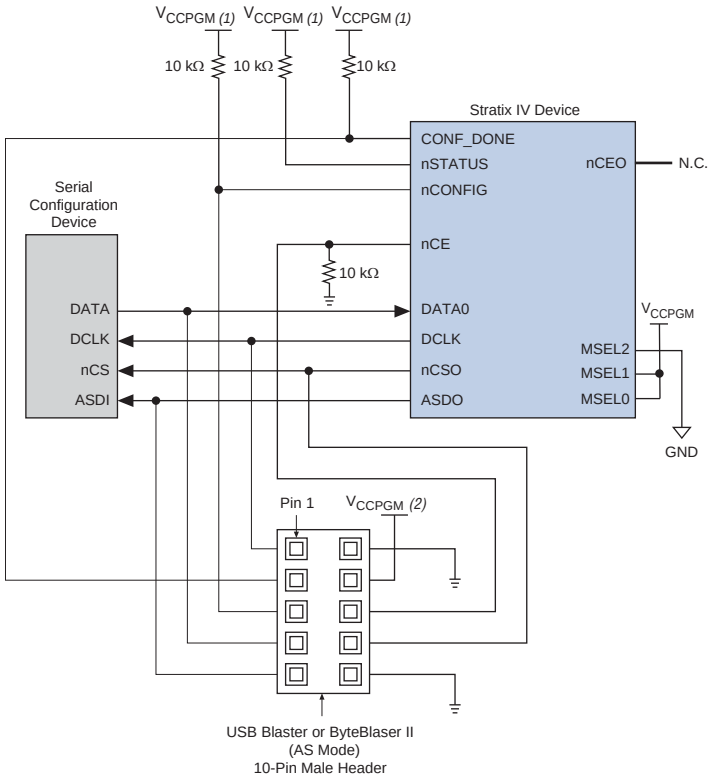
During in-system programming, the download cable disables device access to the AS interface by driving the `nCE` pin high. Stratix IV devices are also held in reset by a low level on `nCONFIG`. After programming is complete, the download cable releases `nCE` and `nCONFIG`, allowing the pull-down and pull-up resistors to drive GND and V_{CCPGM}, respectively. Figure 10–9 shows the download cable connections for the serial configuration device.

Altera has developed Serial FlashLoader (SFL), an in-system programming solution for serial configuration devices using the JTAG interface. This solution requires the Stratix IV device to be a bridge between the JTAG interface and the serial configuration device.

For more information about SFL, refer to *AN 370: Using the Serial FlashLoader with Quartus II Software*.

For more information about the USB-Blaster download cable, refer to the *USB-Blaster Download Cable User Guide*. For more information about the ByteBlaster II cable, refer to the *ByteBlaster II Download Cable User Guide*. For more information about the EthernetBlaster download cable, refer to the *EthernetBlaster Communications Cable User Guide*.

Figure 10–9. In-System Programming of Serial Configuration Devices



Notes to Figure 10–9:

(1) Connect these pull-up resistors to V_{CCPGM} at a 3.0-V supply.

(2) Power up the USB-ByteBlaster, ByteBlaster II, or EthernetBlaster cable's $V_{CC(TRGT)}$ with V_{CCPGM}.

You can program serial configuration devices with the Quartus II software using the Altera programming hardware and the appropriate configuration device programming adapter.

In production environments, you can program serial configuration devices using multiple methods. You can use Altera programming hardware or other third-party programming hardware to program blank serial configuration devices before they are mounted on PCBs. Alternatively, you can use an on-board microprocessor to program the serial configuration device in-system using C-based software drivers provided by Altera.

You can program a serial configuration device in-system by an external microprocessor using SRunner. SRunner is a software driver developed for embedded serial configuration device programming, which can be easily customized to fit in different embedded systems. SRunner is able to read raw programming data (.**rpd**) and write to serial configuration devices. The serial configuration device programming time using SRunner is comparable to the programming time with the Quartus II software.

For more information about SRunner, refer to *AN 418: SRunner: An Embedded Solution for EPCS Programming* and the source code on the Altera website at www.altera.com.

For more information about programming serial configuration devices, refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* chapter in volume 2 of the *Configuration Handbook*.

Passive Serial Configuration

You can program PS configuration for Stratix IV devices using an intelligent host, such as a MAX II device or microprocessor with flash memory, or a download cable. In the PS scheme, an external host (a MAX II device, embedded processor, or host PC) controls configuration. Configuration data is clocked into the target Stratix IV device using the `DATA0` pin at each rising edge of `DCLK`.

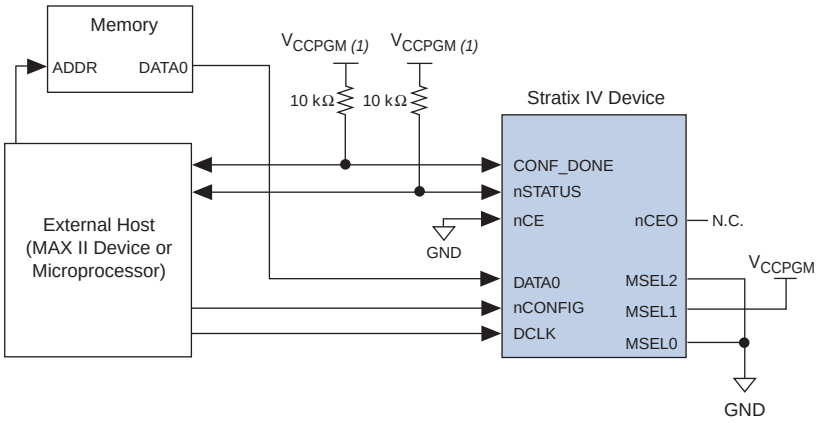
The Stratix IV decompression and design security features are fully available when configuring your Stratix IV device using PS mode.

PS Configuration Using a MAX II Device as an External Host

In this configuration scheme, you can use a MAX II device as an intelligent host that controls the transfer of configuration data from a storage device, such as flash memory, to the target Stratix IV device. You can store configuration data in **.rbf**, **.hex**, or **.ttf** format.

Figure 10–10 shows the configuration interface connections between a Stratix IV device and a MAX II device for single device configuration.

Figure 10–10. Single Device PS Configuration Using an External Host



Note to Figure 10–10:

(1) Connect the resistor to a supply that provides an acceptable input signal for the Stratix IV device. V_{CCPGM} must be high enough to meet the V_{IH} specification of the I/O on the device and the external host. Altera recommends you power up all configuration system I/Os with V_{CCPGM}.

After power-up, Stratix IV devices go through a POR. The POR delay depends on the PORSEL pin setting. When PORSEL is driven low, the standard POR time is 100 ms < T_{POR} < 300 ms. When PORSEL is driven high, the fast POR time is 4 ms < T_{POR} < 12 ms. During POR, the device resets, holds nSTATUS low, and tri-states all user I/O pins. After the device successfully exits POR, all user I/O pins continue to be tri-stated. If nIO_pullup is driven low during power-up and configuration, the user I/O pins and dual-purpose I/O pins will have weak pull-up resistors that are on (after POR) before and during configuration. If nIO_pullup is driven high, the weak pull-up resistors are disabled.

The configuration cycle consists of three stages: reset, configuration, and initialization. While nCONFIG or nSTATUS are low, the device is in reset. To initiate configuration, the MAX II device must generate a low-to-high transition on the nCONFIG pin.

☞ V_{CC}, V_{CCIO}, V_{CCPGM}, and V_{CCPD} of the banks where the configuration pins reside must be fully powered to the appropriate voltage levels to begin the configuration process.

When nCONFIG goes high, the device comes out of reset and releases the open-drain nSTATUS pin, which is then pulled high by an external 10-kΩ pull-up resistor. After nSTATUS is released, the device is ready to receive configuration data and the configuration stage begins. When nSTATUS is pulled high, the MAX II device places the configuration data one bit at a time on the DATA0 pin. If you are using configuration data in **.rbf**, **.hex**, or **.ttf** format, you must send the LSB of each data byte first. For example, if the **.rbf** contains the byte sequence 02 1B EE 01 FA, the serial bitstream you must transmit to the device is 0100-0000 1101-1000 0111-0111 1000-0000 0101-1111.

The Stratix IV device receives configuration data on the DATA0 pin and the clock is received on the DCLK pin. Data is latched into the device on the rising edge of DCLK. Data is continuously clocked into the target device until CONF_DONE goes high. After the device has received all configuration data successfully, it releases the open-drain CONF_DONE pin, which is pulled high by an external 10-kΩ pull-up resistor. A low-to-high transition on CONF_DONE indicates configuration is complete and initialization of the device can begin. The CONF_DONE pin must have an external 10-kΩ pull-up resistor for the device to initialize.

In Stratix IV devices, the initialization clock source is either the internal oscillator or the optional CLKUSR pin. By default, the internal oscillator is the clock source for initialization. If you use the internal oscillator, the Stratix IV device provides itself with enough clock cycles for proper initialization. Therefore, if the internal oscillator is the initialization clock source, sending the entire configuration file to the device is sufficient to configure and initialize the device. Driving DCLK to the device after configuration is complete does not affect device operation.

You also have the flexibility to synchronize initialization of multiple devices or to delay initialization with the CLKUSR option. You can turn on the **Enable user-supplied start-up clock** (**CLKUSR**) option in the Quartus II software from the **General** tab of the **Device and Pin Options** dialog box. If you supply a clock on CLKUSR, it will not affect the configuration process. After all configuration data has been accepted and CONF_DONE goes high, CLKUSR is enabled after the time specified at t_{CD2CU}. After this time period elapses, Stratix IV devices require 8,532 clock cycles to initialize properly and enter user mode. Stratix IV devices support a CLKUSR f_{MAX} of 125 MHz.

An optional INIT_DONE pin is available that signals the end of initialization and the start of user-mode with a low-to-high transition. The **Enable INIT_DONE Output** option is available in the Quartus II software from the **General** tab of the **Device and Pin Options** dialog box. If you use the INIT_DONE pin, it is high due to an external 10-kΩ pull-up resistor when nCONFIG is low and during the beginning of configuration. After the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin goes low. When initialization is complete, the INIT_DONE pin is released and pulled high. The MAX II device must be able to detect this low-to-high transition that signals the device has entered user mode. When initialization is complete, the device enters user mode. In user-mode, the user I/O pins no longer have weak pull-up resistors and function as assigned in your design.

☞ Two DCLK falling edges are required after CONF_DONE goes high to begin the initialization of the device for both uncompressed and compressed bitstream in PS.

To ensure DCLK and DATA0 are not left floating at the end of configuration, the MAX II device must drive them either high or low, whichever is convenient on your board. The DATA[0] pin is available as a user I/O pin after configuration. When you chose the PS scheme as a default in the Quartus II software, this I/O pin is tri-stated in user mode and must be driven by the MAX II device. To change this default option in the Quartus II software, select the **Dual-Purpose Pins** tab of the **Device and Pin Options** dialog box.

The configuration clock (DCLK) speed must be below the specified frequency to ensure correct configuration. No maximum DCLK period exists, which means you can pause the configuration by halting DCLK for an indefinite amount of time.

If an error occurs during configuration, the device drives its nSTATUS pin low, resetting itself internally. The low signal on the nSTATUS pin also alerts the MAX II device that there is an error. If the **Auto-restart configuration after error** option (available in the Quartus II software from the **General** tab of the **Device and Pin Options** dialog box) is turned on, the Stratix IV device releases nSTATUS after a reset time-out period (a maximum of 500 µs). After nSTATUS is released and pulled high by a pull-up resistor, the MAX II device can try to reconfigure the target device without needing to pulse nCONFIG low. If this option is turned off, the MAX II device must generate a low-to-high transition (with a low pulse of at least 2 µs) on nCONFIG to restart the configuration process.

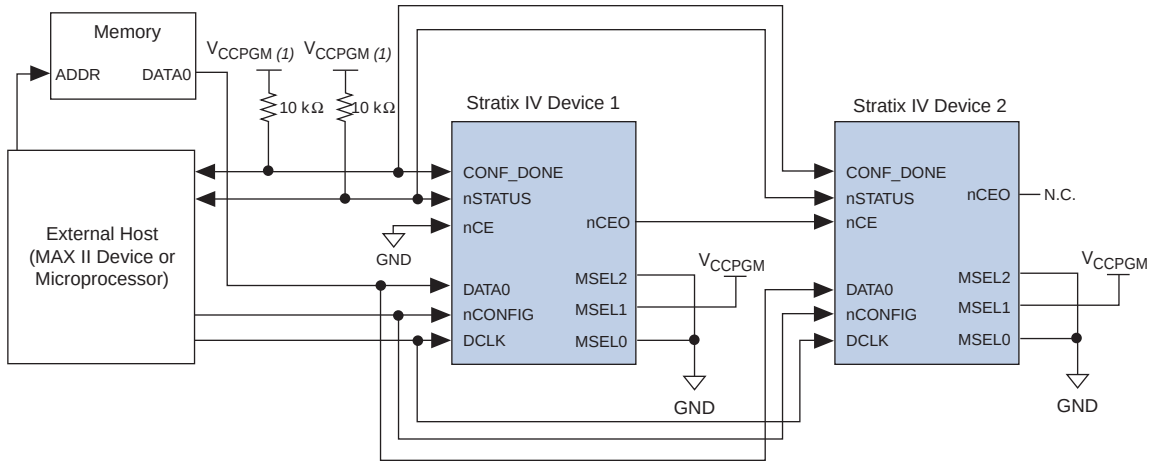
The MAX II device can also monitor the CONF_DONE and INIT_DONE pins to ensure successful configuration. The CONF_DONE pin must be monitored by the MAX II device to detect errors and determine when programming completes. If all configuration data is sent, but CONF_DONE or INIT_DONE have not gone high, the MAX II device must reconfigure the target device.

☞ If you use the optional CLKUSR pin and nCONFIG is pulled low to restart configuration during device initialization, you must ensure that CLKUSR continues toggling during the time nSTATUS is low (a maximum of 500 µs).

When the device is in user-mode, you can initiate a reconfiguration by transitioning the nCONFIG pin low-to-high. The nCONFIG pin must be low for at least 2 µs. When nCONFIG is pulled low, the device also pulls nSTATUS and CONF_DONE low and all I/O pins are tri-stated. Once nCONFIG returns to a logic high level and nSTATUS is released by the device, reconfiguration begins.

Figure 10–11 shows how to configure multiple devices using a MAX II device. This circuit is similar to the PS configuration circuit for a single device, except the Stratix IV devices are cascaded for multi-device configuration.

Figure 10–11. Multi-Device PS Configuration Using an External Host



Note to Figure 10–11:

(1) Connect the resistor to a supply that provides an acceptable input signal for all Stratix IV devices in the chain. V_{CCPGM} must be high enough to meet the V_{IH} specification of the I/O on the device and the external host. Altera recommends you power up all configuration system I/Os with V_{CCPGM}.

In multi-device PS configuration, the first device's nCE pin is connected to GND, while its nCEO pin is connected to nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. The second device in the chain begins configuration within one clock cycle. Therefore, the transfer of data destinations is transparent to the MAX II device. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE) are connected to every device in the chain. Configuration signals can require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered for every fourth device. Because all device CONF_DONE pins are tied together, all devices initialize and enter user mode at the same time.

Because all nSTATUS and CONF_DONE pins are tied together, if any device detects an error, configuration stops for the entire chain and you must reconfigure the entire chain. For example, if the first device flags an error on nSTATUS, it resets the chain by pulling its nSTATUS pin low. This behavior is similar to a single device detecting an error.

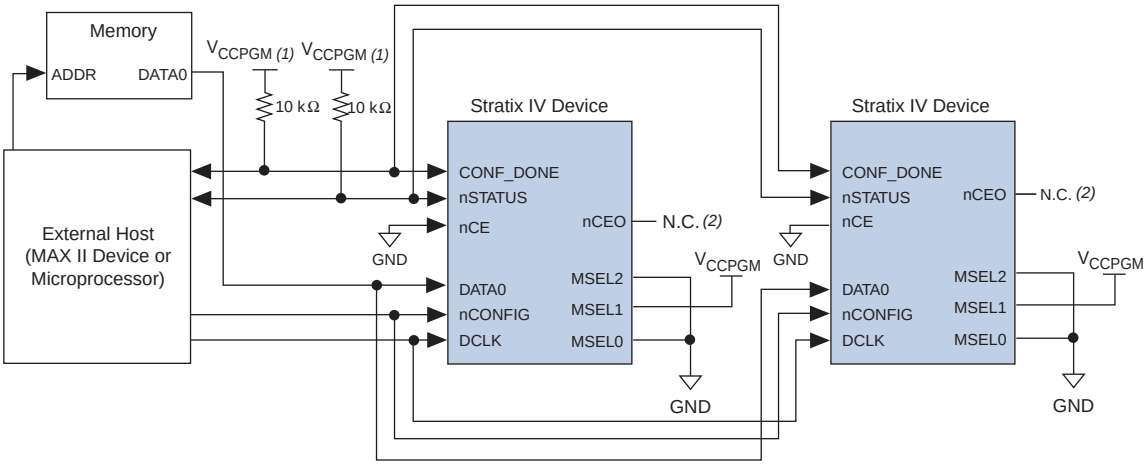
If the **Auto-restart configuration after error** option is turned on, the devices release their nSTATUS pins after a reset time-out period (a maximum of 500 µs). After all nSTATUS pins are released and pulled high, the MAX II device can try to reconfigure the chain without needing to pulse nCONFIG low. If this option is turned off, the MAX II device must generate a low-to-high transition (with a low pulse of at least 2 µs) on nCONFIG to restart the configuration process.

In your system, you can have multiple devices that contain the same configuration data. To support this configuration scheme, all device nCE inputs are tied to GND, while the nCEO pins are left floating. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE) are connected to every device in the chain.

Configuration signals can require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered for every fourth device. Devices must be the same density and package. All devices start and complete configuration at the same time.

Figure 10–12 shows multi-device PS configuration when both Stratix IV devices are receiving the same configuration data.

Figure 10–12. Multiple-Device PS Configuration When Both Devices Receive the Same Data



Notes to Figure 10–12:

(1) Connect the resistor to a supply that provides an acceptable input signal for all Stratix IV devices in the chain. V_{CCPGM} must be high enough to meet the V_{IH} specification of the I/O on the device and the external host. Altera recommends you power up all configuration system I/Os with V_{CCPGM}.

(2) The nCEO pins of both devices are left unconnected when configuring the same configuration data into multiple devices.

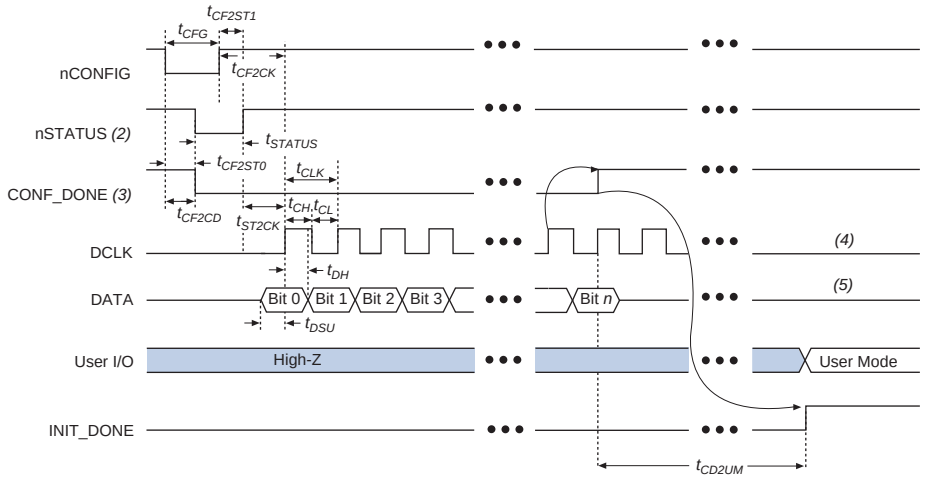
You can use a single configuration chain to configure Stratix IV devices with other Altera devices. To ensure that all devices in the chain complete configuration at the same time, or that an error flagged by one device initiates reconfiguration in all devices, all of the device CONF_DONE and nSTATUS pins must be tied together.

For more information about configuring multiple Altera devices in the same configuration chain, refer to the *Configuring Mixed Altera FPGA Chains* chapter in volume 2 of the *Configuration Handbook*.

PS Configuration Timing

Figure 10–13 shows the timing waveform for PS configuration when using a MAX II device as an external host.

Figure 10–13. PS Configuration Timing Waveform *(Note 1)*



Notes to Figure 10–13:

(1) The beginning of this waveform shows the device in user mode. In user mode, nCONFIG, nSTATUS, and CONF_DONE are at logic high levels. When nCONFIG is pulled low, a reconfiguration cycle begins.

(2) After power-up, the Stratix IV device holds nSTATUS low for the time of the POR delay.

(3) After power-up, before and during configuration, CONF_DONE is low.

(4) Do not leave DCLK floating after configuration. You can drive it high or low, whichever is more convenient.

(5) DATA[0] is available as a user I/O pin after configuration. The state of this pin depends on the dual-purpose pin settings.

Table 10–6 lists the timing parameters for Stratix IV devices for PS configuration.

Table 10–6. PS Timing Parameters for Stratix IV Devices (Part 1 of 2) *(Note 1)*

Symbol	Parameter	Minimum	Maximum	Units
t_{CF2CD}	nCONFIG low to CONF_DONE low	—	800	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low	—	800	ns
t_{CFG}	nCONFIG low pulse width	2	—	µs
t_{STATUS}	nSTATUS low pulse width	10	500 *(2)*	µs
t_{CF2ST1}	nCONFIG high to nSTATUS high	—	500 *(2)*	µs
t_{CF2CK}	nCONFIG high to first rising edge on DCLK	500	—	µs
t_{ST2CK}	nSTATUS high to first rising edge of DCLK	2	—	µs
t_{DSU}	Data setup time before rising edge on DCLK	4	—	ns
t_{DH}	Data hold time after rising edge on DCLK	0	—	ns
t_{CH}	DCLK high time *(4)*	3.2	—	ns
t_{CL}	DCLK low time *(4)*	3.2	—	ns
t_{CLK}	DCLK period *(4)*	8	—	ns
f_{MAX}	DCLK frequency	—	125	MHz
t_R	Input rise time	—	40	ns

Table 10–6. PS Timing Parameters for Stratix IV Devices (Part 2 of 2) *(Note 1)*

Symbol	Parameter	Minimum	Maximum	Units
t_F	Input fall time	—	40	ns
t_{CD2UM}	CONF_DONE high to user mode *(3)*	55	150	µs
t_{CD2CU}	CONF_DONE high to CLKUSR enabled	4 × maximum DCLK period	—	—
t_{CD2UMC}	CONF_DONE high to user mode with CLKUSR option on	t_{CD2CU} + (8532 CLKUSR period)	—	—

Notes to Table 10–6:

(1) This information is preliminary.
(2) This value is applicable if you do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.
(3) The minimum and maximum numbers apply only if you choose the internal oscillator as the clock source for starting the device.
(4) Adding up t_{CH} and t_{CL} equals to t_{CLK}. When t_{CH} is 3.2 ns (min), t_{CL} must be 4.8 ns and vice versa.

Device configuration options and how to create configuration files are described in the *Device Configuration Options* and *Configuration File Formats* chapters in volume 2 of the *Configuration Handbook*.

PS Configuration Using a Microprocessor

In this PS configuration scheme, a microprocessor controls the transfer of configuration data from a storage device, such as flash memory, to the target Stratix IV device.

For more information about configuration and timing information, refer to "PS Configuration Using a MAX II Device as an External Host" on page 10–23. This section is also applicable when using a microprocessor as an external host.

PS Configuration Using a Download Cable

In this section, the generic term "download cable" includes the Altera USB-Blaster universal serial bus (USB) port download cable, MasterBlaster serial/USB communications cable, ByteBlaster II parallel port download cable, ByteBlasterMV parallel port download cable, and EthernetBlaster download cable.

In a PS configuration with a download cable, an intelligent host (such as a PC) transfers data from a storage device to the device using the USB Blaster, MasterBlaster, ByteBlaster II, EthernetBlaster, or ByteBlasterMV cable.

After power-up, Stratix IV devices go through a POR. The POR delay depends on the PORSEL pin setting. When PORSEL is driven low, the standard POR time is 100 ms < T_{POR} < 300 ms. When PORSEL is driven high, the fast POR time is 4 ms < T_{POR} < 12 ms. During POR, the device resets, holds nSTATUS low, and tri-states all user I/O pins. After the device successfully exits POR, all user I/O pins continue to be tri-stated. If nIO_pullup is driven low during power-up and configuration, the user I/O pins and dual-purpose I/O pins will have weak pull-up resistors, which are on (after POR) before and during configuration. If nIO_pullup is driven high, the weak pull-up resistors are disabled.

The configuration cycle consists of three stages: reset, configuration, and initialization. While nCONFIG or nSTATUS are low, the device is in reset. To initiate configuration in this scheme, the download cable generates a low-to-high transition on the nCONFIG pin.

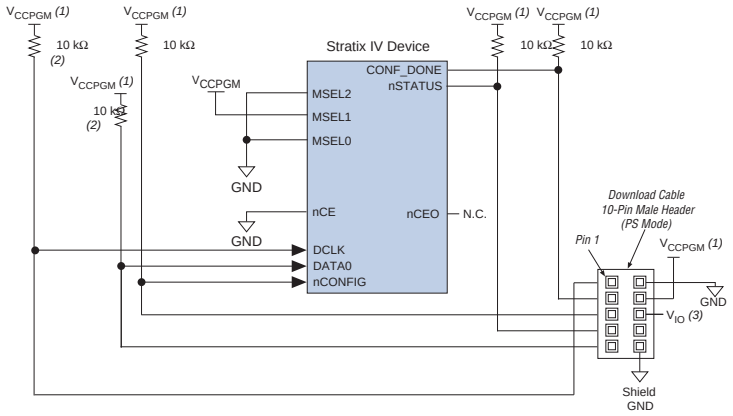
To begin configuration, power the V_{CC}, V_{CCIO}, V_{CCPGM}, and V_{CCPD} voltages (for the banks where the configuration pins reside) to the appropriate voltage levels.

When nCONFIG goes high, the device comes out of reset and releases the open-drain nSTATUS pin, which is then pulled high by an external 10-kΩ pull-up resistor. After nSTATUS is released, the device is ready to receive configuration data and the configuration stage begins. The programming hardware or download cable then places the configuration data one bit at a time on the device's DATA0 pin. The configuration data is clocked into the target device until CONF_DONE goes high. The CONF_DONE pin must have an external 10-kΩ pull-up resistor for the device to initialize.

When using a download cable, setting the **Auto-restart configuration after error** option does not affect the configuration cycle because you must manually restart configuration in the Quartus II software when an error occurs. Additionally, the **Enable user-supplied start-up clock** (**CLKUSR**) option has no affect on the device initialization because this option is disabled in the **.sof** when programming the device using the Quartus II programmer and download cable. Therefore, if you turn on the CLKUSR option, you do not need to provide a clock on CLKUSR when you are configuring the device with the Quartus II programmer and a download cable.

Figure 10–14 shows PS configuration for Stratix IV devices using a USB Blaster, EthernetBlaster, MasterBlaster, ByteBlaster II, or ByteBlasterMV cable.

Figure 10–14. PS Configuration Using a USB Blaster, EthernetBlaster, MasterBlaster, ByteBlaster II, or ByteBlasterMV Cable



Notes to Figure 10–14:

(1) Connect the pull-up resistor to the same supply voltage (V_{CCPGM}) as the USB Blaster, MasterBlaster (VIO pin), ByteBlaster II, ByteBlasterMV, or EthernetBlaster cable.

(2) You only need the pull-up resistors on DATA0 and DCLK if the download cable is the only configuration scheme used on your board. This ensures that DATA0 and DCLK are not left floating after configuration. For example, if you are also using a configuration device, you do not need the pull-up resistors on DATA0 and DCLK.

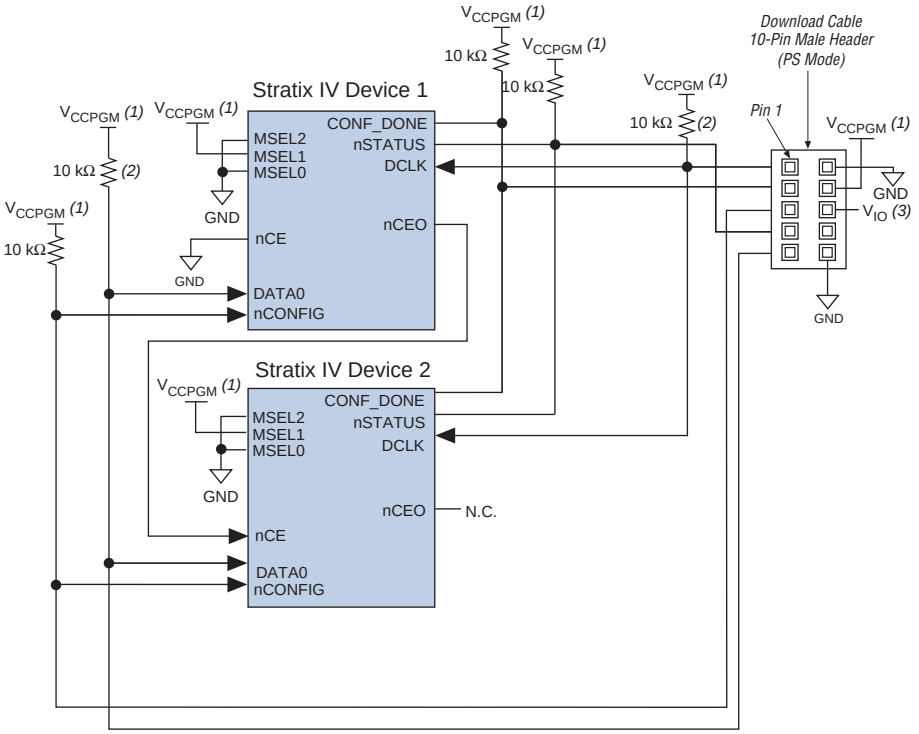
(3) Pin 6 of the header is a V_{IO} reference voltage for the MasterBlaster output driver. V_{IO} must match the device's V_{CCPGM}. For more information about this value, refer to the *MasterBlaster Serial/USB Communications Cable User Guide*. In the USB-Blaster, ByteBlaster II, and ByteBlasterMV cable, this pin is a no connect.

You can use a download cable to configure multiple Stratix IV devices by connecting each device's nCEO pin to the subsequent device's nCE pin. The first device's nCE pin is connected to GND, while its nCEO pin is connected to the nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE) are connected to every device in the chain. Because all CONF_DONE pins are tied together, all devices in the chain initialize and enter user mode at the same time.

In addition, because the nSTATUS pins are tied together, the entire chain halts configuration if any device detects an error. The **Auto-restart configuration after error** option does not affect the configuration cycle because you must manually restart the configuration in the Quartus II software when an error occurs.

Figure 10–15 shows how to configure multiple Stratix IV devices with a download cable.

Figure 10–15. Multi-Device PS Configuration Using a USB Blaster, EthernetBlaster, MasterBlaster, ByteBlaster II, or ByteBlasterMV Cable



Notes to Figure 10–15:

(1) Connect the pull-up resistor to the same supply voltage (V_{CCPGM}) as the USB Blaster, MasterBlaster (VIO pin), ByteBlaster II, ByteBlasterMV, or EthernetBlaster cable.

(2) You only need the pull-up resistors on DATA0 and DCLK if the download cable is the only configuration scheme used on your board. This is to ensure that DATA0 and DCLK are not left floating after configuration. For example, if you are also using a configuration device, you do not need the pull-up resistors on DATA0 and DCLK.

(3) Pin 6 of the header is a V_{IO} reference voltage for the MasterBlaster output driver. V_{IO} must match the device's V_{CCPGM}. For more information about this value, refer to the *MasterBlaster Serial/USB Communications Cable User Guide*. In the USB-Blaster, ByteBlaster II, and ByteBlasterMV cables, this pin is a no connect.

For more information about how to use the USB Blaster, MasterBlaster, ByteBlaster II, or ByteBlasterMV cables, refer to the following user guides:

- *USB-Blaster Download Cable User Guide*
- *MasterBlaster Serial/USB Communications Cable User Guide*
- *ByteBlaster II Download Cable User Guide*
- *ByteBlasterMV Download Cable User Guide*
- *EthernetBlaster Communications Cable User Guide*

JTAG Configuration

JTAG has developed a specification for boundary-scan testing. This boundary-scan test (BST) architecture offers the capability to efficiently test components on PCBs with tight lead spacing. The BST architecture can test pin connections without using physical test probes and capture functional data while a device is operating normally. You can also use JTAG circuitry to shift configuration data into the device. The Quartus II software automatically generates **.sof**s that you can use for JTAG configuration with a download cable in the Quartus II software programmer.

For more information about JTAG boundary-scan testing and commands available using Stratix IV devices, refer to the following documents:

- *JTAG Boundary Scan Testing* chapter
- *Programming Support for Jam STAPL Language*

Stratix IV devices are designed such that JTAG instructions have precedence over any device configuration modes. Therefore, JTAG configuration can take place without waiting for other configuration modes to complete. For example, if you attempt JTAG configuration of Stratix IV devices during PS configuration, PS configuration is terminated and JTAG configuration begins.

You cannot use the Stratix IV decompression or design security features if you are configuring your Stratix IV device when using JTAG-based configuration.

A device operating in JTAG mode uses four required pins, `TDI`, `TDO`, `TMS`, and `TCK`, and one optional pin, `TRST`. The `TCK` pin has an internal weak pull-down resistor, while the `TDI`, `TMS`, and `TRST` pins have weak internal pull-up resistors (typically 25 kΩ). The JTAG output pin `TDO` and all JTAG input pins are powered by 2.5-V/3.0-V V_{CCPD}. All the JTAG pins only support the LVTTL I/O standard.

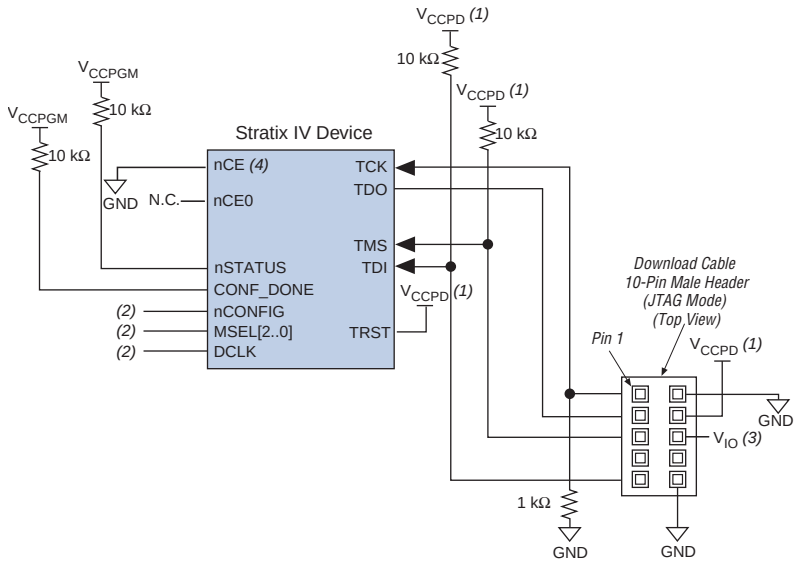
All user I/O pins are tri-stated during JTAG configuration.

The `TDO` output is powered by the V_{CCPD} power supply of I/O bank 1A. For more information about how to connect a JTAG chain with multiple voltages across the devices in the chain, refer to the *JTAG Boundary Scan Testing* chapter.

During JTAG configuration, you can download data to the device on the PCB through the USB Blaster, MasterBlaster, ByteBlaster II, EthernetBlaster, or ByteBlasterMV download cable. Configuring devices through a cable is similar to programming devices in-system, except you must connect the TRST pin to V_{CCPD}. This ensures that the TAP controller is not reset.

Figure 10–16 shows JTAG configuration of a single Stratix IV device when using a download cable.

Figure 10–16. JTAG Configuration of a Single Device Using a Download Cable



Notes to Figure 10–16:

(1) Connect the pull-up resistor to the same supply voltage as the USB Blaster, MasterBlaster (VIO pin), ByteBlaster II, ByteBlasterMV, or EthernetBlaster cable. The voltage supply can be connected to the V_{CCPD} of the device.

(2) Connect the nCONFIG and MSEL[2..0] pins to support a non-JTAG configuration scheme. If you only use the JTAG configuration, connect nCONFIG to V_{CCPGM} and MSEL[2..0] to GND. Pull DCLK either high or low, whichever is convenient on your board.

(3) Pin 6 of the header is a V_{IO} reference voltage for the MasterBlaster output driver. V_{IO} must match the device's V_{CCPD}. For more information about this value, refer to the *MasterBlaster Serial/USB Communications Cable User Guide*. In the USB-Blaster, ByteBlaster II, and ByteBlasterMV cable, this pin is a no connect.

(4) You must connect nCE to GND or driven low for successful JTAG configuration.

To configure a single device in a JTAG chain, the programming software places all other devices in bypass mode. In bypass mode, devices pass programming data from the TDI pin to the TDO pin through a single bypass register without being affected internally. This scheme enables the programming software to program or verify the target device. Configuration data driven into the device appears on the TDO pin one clock cycle later.

The Quartus II software verifies successful JTAG configuration upon completion. At the end of configuration, the software checks the state of CONF_DONE through the JTAG port. When the Quartus II software generates a JAM file (**.jam**) for a multi-device chain, it contains instructions so that all the devices in the chain are initialized at the same time. If CONF_DONE is not high, the Quartus II software indicates that configuration has failed. If CONF_DONE is high, the software indicates that configuration was successful. After the configuration bitstream is transmitted serially using the JTAG TDI port, the TCK port is clocked an additional 1,094 cycles to perform device initialization.

Stratix IV devices have dedicated JTAG pins that always function as JTAG pins. Not only can you perform JTAG testing on Stratix IV devices before and after, but also during configuration. While other device families do not support JTAG testing during configuration, Stratix IV devices support the bypass, ID code, and sample instructions during configuration without interrupting configuration. All other JTAG instructions may only be issued by first interrupting configuration and reprogramming the I/O pins using the CONFIG_IO instruction.

The CONFIG_IO instruction allows I/O buffers to be configured using the JTAG port and when issued, interrupts configuration. This instruction allows you to perform board-level testing prior to configuring the Stratix IV device or waiting for a configuration device to complete configuration. After configuration has been interrupted and JTAG testing is complete, you must reconfigure the part using JTAG (PULSE_CONFIG instruction) or by pulsing nCONFIG low.

The chip-wide reset (DEV_CLRn) and chip-wide output enable (DEV_OE) pins on Stratix IV devices do not affect JTAG boundary-scan or programming operations. Toggling these pins does not affect JTAG operations (other than the usual boundary-scan operation).

When designing a board for JTAG configuration for Stratix IV devices, consider the dedicated configuration pins. Table 10–7 lists how these pins are connected during JTAG configuration.

Table 10–7. Dedicated Configuration Pin Connections During JTAG Configuration

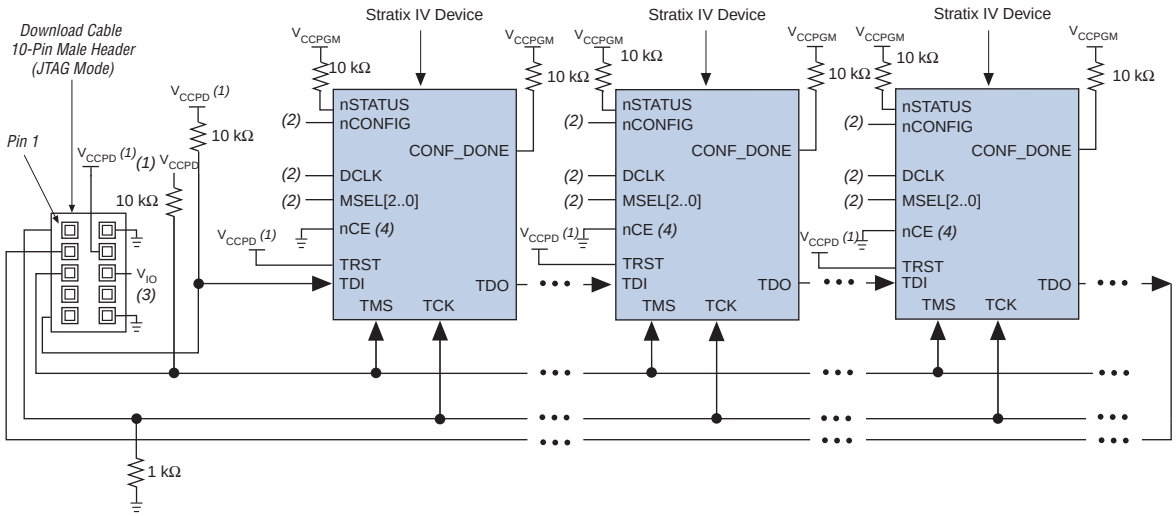
Signal	Description
nCE	On all Stratix IV devices in the chain, nCE must be driven low by connecting it to ground, pulling it low using a resistor, or driving it by some control circuitry. For devices that are also in multi-device FPP, AS, or PS configuration chains, the nCE pins must be connected to GND during JTAG configuration or JTAG must be configured in the same order as the configuration chain.
nCEO	On all Stratix IV devices in the chain, you can leave nCEO floating or connected to the nCE of the next device.
MSEL	Do not leave these pins floating. These pins support whichever non-JTAG configuration is used in production. If you only use JTAG configuration, tie these pins to GND.
nCONFIG	Driven high by connecting to V_{CCPGM}, pulling up using a resistor, or driven high by some control circuitry.
nSTATUS	Pull to V_{CCPGM} using a 10-kΩ resistor. When configuring multiple devices in the same JTAG chain, each nSTATUS pin must be pulled up to V_{CCPGM} individually.
CONF_DONE	Pull to V_{CCPGM} using a 10-kΩ resistor. When configuring multiple devices in the same JTAG chain, each CONF_DONE pin must be pulled up to V_{CCPGM} individually. CONF_DONE going high at the end of JTAG configuration indicates successful configuration.
DCLK	Do not leave DCLK floating. Drive low or high, whichever is more convenient on your board.

When programming a JTAG device chain, one JTAG-compatible header is connected to several devices. The number of devices in the JTAG chain is limited only by the drive capability of the download cable. When four or more devices are connected in a JTAG chain, Altera recommends buffering the TCK, TDI, and TMS pins with an on-board buffer.

JTAG-chain device programming is ideal when the system contains multiple devices, or when testing your system using JTAG BST circuitry.

Figure 10–17 shows a multi-device JTAG configuration when using a download cable.

Figure 10–17. JTAG Configuration of Multiple Devices Using a Download Cable



Notes to Figure 10–17:

(1) Connect the pull-up resistor to the same supply voltage as the USB Blaster, MasterBlaster (V_{IO} pin), ByteBlaster II, ByteBlasterMV, or EthernetBlaster cable. Connect the voltage supply to V_{CCPD} of the device.

(2) Connect the `nCONFIG` and `MSEL[2..0]` pins to support a non-JTAG configuration scheme. If you only use JTAG configuration, connect `nCONFIG` to V_{CCPGM} and `MSEL[2..0]` to GND. Pull `DCLK` either high or low, whichever is convenient on your board.

(3) Pin 6 of the header is a V_{IO} reference voltage for the MasterBlaster output driver. V_{IO} must match the device's V_{CCPD}. For more information about this value, refer to the *MasterBlaster Serial/USB Communications Cable User Guide*. In the USB-Blaster, ByteBlaster II, and ByteBlasterMV cables, this pin is a no connect.

(4) You must connect `nCE` to GND or drive it low for successful JTAG configuration.

You must connect the `nCE` pin to GND or drive it low during JTAG configuration. In multi-device FPP, AS, and PS configuration chains, the first device's `nCE` pin is connected to GND, while its `nCEO` pin is connected to `nCE` of the next device in the chain. The last device's `nCE` input comes from the previous device, while its `nCEO` pin is left floating. In addition, the `CONF_DONE` and `nSTATUS` signals are all shared in multi-device FPP, AS, or PS configuration chains so the devices can enter user mode at the same time after configuration is complete. When the `CONF_DONE` and `nSTATUS` signals are shared among all the devices, you must configure every device when JTAG configuration is performed.

If you only use JTAG configuration, Altera recommends that you connect the circuitry as shown in Figure 10–17, where each of the `CONF_DONE` and `nSTATUS` signals are isolated, so that each device can enter user mode individually.

After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. Therefore, if these devices are also in a JTAG chain, ensure the nCE pins are connected to GND during JTAG configuration or that the devices are JTAG configured in the same order as the configuration chain. As long as the devices are JTAG configured in the same order as the multi-device configuration chain, the nCEO of the previous device drives the nCE of the next device low when it has successfully been JTAG configured.

You can place other Altera devices that have JTAG support in the same JTAG chain for device programming and configuration.

JTAG configuration support is enhanced and allows more than 17 Stratix IV devices to be cascaded in a JTAG chain.

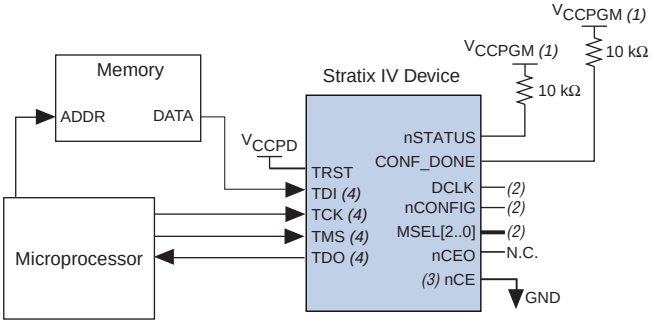
For more information about configuring multiple Altera devices in the same configuration chain, refer to the *Configuring Mixed Altera Device Chains* chapter in volume 2 of the *Configuration Handbook*.

You can configure Stratix IV devices using multiple configuration schemes on the same board. Combining JTAG configuration with AS configuration on your board is useful in the prototyping environment because it allows multiple methods to configure your FPGA.

For more information about combining JTAG configuration with other configuration schemes, refer to the *Combining Different Configuration Schemes* chapter in volume 2 of the *Configuration Handbook*.

Figure 10–18 shows JTAG configuration of a Stratix IV device using a microprocessor.

Figure 10–18. JTAG Configuration of a Single Device Using a Microprocessor



Notes to Figure 10–18:

(1) Connect the pull-up resistor to a supply that provides an acceptable input signal for all Stratix IV devices in the chain. V_{CCPGM} must be high enough to meet the V_{IH} specification of the I/O on the device.

(2) Connect the nCONFIG and MSEL[2..0] pins to support a non-JTAG configuration scheme. If you use only the JTAG configuration, connect nCONFIG to V_{CCGPM} and MSEL[2..0] to GND. Pull DCLK either high or low, whichever is convenient on your board.

(3) Connect nCE to GND or drive it low for successful JTAG configuration.

(4) The microprocessor must use the same I/O standard as V_{CCPD} to drive the JTAG pins.

Jam STAPL

Jam™ STAPL, JEDEC standard JESD-71, is a standard file format for in-system programmability (ISP) purposes. Jam STAPL supports programming or configuration of programmable devices and testing of electronic systems, using the IEEE 1149.1 JTAG interface. Jam STAPL is a freely licensed open standard.

The Jam Player provides an interface for manipulating the IEEE Std. 1149.1 JTAG TAP state machine.

For more information about JTAG and Jam STAPL in embedded environments, refer to *AN 122: Using Jam STAPL for ISP and ICR via an Embedded Processor*. To download the Jam Player, visit the Altera website at www.altera.com.

Device Configuration Pins

The following tables list the connections and functionality of all the configuration-related pins on Stratix IV devices. Table 10–8 lists the Stratix IV configuration pins and their power supply.

Table 10–8. Stratix IV Configuration Pin Summary (Part 1 of 2) *(Note 1)*

Description	Input/Output	Dedicated	Powered By	Configuration Mode
TDI	Input	Yes	V_{CCPD}	JTAG
TMS	Input	Yes	V_{CCPD}	JTAG
TCK	Input	Yes	V_{CCPD}	JTAG
TRST	Input	Yes	V_{CCPD}	JTAG
TDO	Output	Yes	V_{CCPD}	JTAG
CRC_ERROR	Output	—	Pull-up	Optional, all modes
DATA0	Input	—	V_{CCPGM}/V_{CCIO} *(3)*	All modes except JTAG
DATA[7..1]	Input	—	V_{CCPGM}/V_{CCIO} *(3)*	FPP
INIT_DONE	Output	—	Pull-up	Optional, all modes
CLKUSR	Input	—	V_{CCPGM}/V_{CCIO} *(3)*	Optional
nSTATUS	Bidirectional	Yes	V_{CCPGM}/Pull-up	All modes
nCE	Input	Yes	V_{CCPGM}	All modes
CONF_DONE	Bidirectional	Yes	V_{CCPGM}/Pull-up	All modes
nCONFIG	Input	Yes	V_{CCPGM}	All modes
PORSEL	Input	Yes	V_{CC} *(2)*	All modes
ASDO	Output	Yes	V_{CCPGM}	AS
nCSO	Output	Yes	V_{CCPGM}	AS
DCLK	Input	Yes	V_{CCPGM}	PS, FPP
	Output	Yes	V_{CCPGM}	AS
nIO_PULLUP	Input	Yes	V_{CC} *(2)*	All modes
nCEO	Output	Yes	V_{CCPGM}	All modes

Table 10–8. Stratix IV Configuration Pin Summary (Part 2 of 2) *(Note 1)*

Description	Input/Output	Dedicated	Powered By	Configuration Mode
MSEL[2..0]	Input	Yes	V_{CC} *(2)*	All modes

Notes to Table 10–8:

(1) The total number of pins is 29. The total number of dedicated pins is 18.

(2) Although MSEL[2..0], PORSEL, and nIO_PULLUP are powered up by V_{CC}, Altera recommends you connect these pins to V_{CCPGM} or GND directly without using a pull-up or pull-down resistor.

(3) These pins are powered up by V_{CCPGM} during configuration. These pins are powered up by V_{CCIO} if they are used as regular I/O in user mode.

Table 10–9 lists the dedicated configuration pins. You must connect these pins properly on your board for successful configuration. Some of these pins may not be required for your configuration schemes.

Table 10–9. Dedicated Configuration Pins on the Stratix IV Device (Part 1 of 4)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
VCCPGM	N/A	All	Power	Dedicated power pin. Use this pin to power all dedicated configuration inputs, dedicated configuration outputs, dedicated configuration bidirectional pins, and some of the dual functional pins that are used for configuration. You must connect this pin to 1.8, 2.5, or 3.0 V. V_{CCPGM} must ramp-up from 0 V to V_{CCPGM} within 100 ms when PORSEL is low or 4 ms when PORSEL is high. If V_{CCPGM} is not ramped up within this specified time, your Stratix IV device will not configure successfully. If your system does not allow a V_{CCPGM} ramp-up within 100 ms or 4 ms, you must hold nCONFIG low until all power supplies are stable.
VCCPD	N/A	All	Power	Dedicated power pin. Use this pin to power the I/O pre-drivers, JTAG input and output pins, and design security circuitry. You must connect this pin to 2.5 V or 3.0 V, depending on the I/O standards selected. For the 3.0-V I/O standard, V_{CCPD} = 3.0 V. For the 2.5 V or below I/O standards, V_{CCPD} = 2.5 V. V_{CCPD} must ramp-up from 0 V to 2.5 V / 3.0 V within 100 ms when PORSEL is low or 4 ms when PORSEL is high. If V_{CCPD} is not ramped up within this specified time, your Stratix IV device will not configure successfully. If your system does not allow a V_{CCPD} to ramp-up time within 100 ms or 4 ms, you must hold nCONFIG low until all power supplies are stable.
PORSEL	N/A	All	Input	Dedicated input that selects between a standard POR time or a fast POR time. A logic low selects a standard POR time setting of 100 ms < T_{POR} < 300 ms and a logic high selects a fast POR time setting of 4 ms < T_{POR} < 12 ms. The PORSEL input buffer is powered by V_{CC} and has an internal 5-kΩ pull-down resistor that is always active. Tie the PORSEL pin directly to V_{CCPGM} or GND.

Table 10–9. Dedicated Configuration Pins on the Stratix IV Device (Part 2 of 4)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nIO_PULLUP	N/A	All	Input	Dedicated input that chooses whether the internal pull-up resistors on the user I/O pins and dual-purpose I/O pins (nCSO, nASDO, DATA[7..0], CLKUSR, and INIT_DONE) are on or off before and during configuration. A logic high turns off the weak internal pull-up resistors; a logic low turns them on. The nIO-PULLUP input buffer is powered by V_{CC} and has an internal 5-kΩ pull-down resistor that is always active. Tie the nIO-PULLUP directly to V_{CCPGM} or GND.
MSEL[2..0]	N/A	All	Input	Three-bit configuration input that sets the Stratix IV device configuration scheme. For the appropriate connections, refer to Table 10–1 on page 10–2 . You must hardwire these pins to V_{CCPGM} or GND. The MSEL[2..0] pins have internal 5-kΩ pull-down resistors that are always active.
nCONFIG	N/A	All	Input	Configuration control input. Pulling this pin low during user-mode causes the device to lose its configuration data, enter a reset state, and tri-state all I/O pins. Returning this pin to a logic high level initiates a reconfiguration. Configuration is possible only if this pin is high, except in JTAG programming mode, when nCONFIG is ignored.
nSTATUS	N/A	All	Bidirectional open-drain	The device drives nSTATUS low immediately after power-up and releases it after the POR time. During user mode and regular configuration, this pin is pulled high by an external 10-kΩ resistor. This pin, when driven low by the Stratix IV device, indicates that the device has encountered an error during configuration. ■ Status output—If an error occurs during configuration, nSTATUS is pulled low by the target device. ■ Status input—If an external source drives the nSTATUS pin low during configuration or initialization, the target device enters an error state. Driving nSTATUS low after configuration and initialization does not affect the configured device. If you use a configuration device, driving nSTATUS low causes the configuration device to attempt to configure the device, but because the device ignores transitions on nSTATUS in user mode, the device does not reconfigure. To initiate a reconfiguration, nCONFIG must be pulled low.

Table 10–9. Dedicated Configuration Pins on the Stratix IV Device (Part 3 of 4)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nSTATUS (continued)	—	—	—	If V_{CCPGM} is not fully powered up, the following could occur: ■ V_{CCPGM} is powered high enough for the nSTATUS buffer to function properly and nSTATUS is driven low. When V_{CCPGM} is ramped up, POR trips and nSTATUS is released after POR expires. ■ V_{CCPGM} is not powered high enough for the nSTATUS buffer to function properly. In this situation, nSTATUS might appear logic high, triggering a configuration attempt that would fail because POR did not yet trip. When V_{CCPD} is powered up, nSTATUS is pulled low because POR did not yet trip. When POR trips after V_{CCPGM} is powered up, nSTATUS is released and pulled high. At that point, reconfiguration is triggered and the device is configured.
CONF_DONE	N/A	All	Bidirectional open-drain	Status output. The target device drives the CONF_DONE pin low before and during configuration. After all the configuration data is received without error and the initialization cycle starts, the target device releases CONF_DONE. Status input. After all the data is received and CONF_DONE goes high, the target device initializes and enters user mode. The CONF_DONE pin must have an external 10-kΩ pull-up resistor for the device to initialize. Driving CONF_DONE low after configuration and initialization does not affect the configured device.
nCE	N/A	All	Input	Active-low chip enable. The nCE pin activates the device with a low signal to allow configuration. The nCE pin must be held low during configuration, initialization, and user mode. In single device configuration, it must be tied low. In multi-device configuration, nCE of the first device is tied low, while its nCEO pin is connected to nCE of the next device in the chain. The nCE pin must also be held low for successful JTAG programming of the device.
nCEO	N/A	All	Output	Output that drives low when device configuration is complete. In single device configuration, this pin is left floating. In multi-device configuration, this pin feeds the next device's nCE pin. The nCEO of the last device in the chain is left floating. The nCEO pin is powered by V_{CCPGM}.
ASDO	N/A	AS	Output	Control signal from the Stratix IV device to the serial configuration device in AS mode used to read out configuration data. In AS mode, ASDO has an internal pull-up resistor that is always active.
nCSO	N/A	AS	Output	Output control signal from the Stratix IV device to the serial configuration device in AS mode that enables the configuration device. In AS mode, nCSO has an internal pull-up resistor that is always active.

Table 10–9. Dedicated Configuration Pins on the Stratix IV Device (Part 4 of 4)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
DCLK	N/A	Synchronous configuration schemes (PS, FPP, AS)	Input (PS, FPP) Output (AS)	In PS and FPP configuration, DCLK is the clock input used to clock data from an external source into the target device. Data is latched into the device on the rising edge of DCLK. In AS mode, DCLK is an output from the Stratix IV device that provides timing for the configuration interface. In AS mode, DCLK has an internal pull-up resistor (typically 25 kΩ) that is always active. After configuration, this pin is tri-stated. In schemes that use a configuration device, DCLK is driven low after configuration is done. In schemes that use a control host, DCLK must be driven either high or low, whichever is more convenient. Toggling this pin after configuration does not affect the configured device.
DATA0	N/A in AS mode. I/O in PS or FPP mode.	PS, FPP, AS	Input	Data input. In serial configuration modes, bit-wide configuration data is presented to the target device on the DATA0 pin. In AS mode, DATA0 has an internal pull-up resistor that is always active. After PS or FPP configuration, DATA0 is available as a user I/O pin. The state of this pin depends on the **Dual-Purpose Pin** settings.
DATA[7..1]	I/O	Parallel configuration schemes (FPP)	Inputs	Data inputs. Byte-wide configuration data is presented to the target device on DATA[7..0]. In serial configuration schemes, they function as user I/O pins during configuration, which means they are tri-stated. After FPP configuration, DATA[7..1] are available as user I/O pins. The state of these pins depends on the **Dual-Purpose Pin** settings.

Table 10–10 lists the optional configuration pins. If these optional configuration pins are not enabled in the Quartus II software, they are available as general-purpose user I/O pins. Therefore, during configuration, these pins function as user I/O pins and are tri-stated with weak pull-up resistors.

Table 10–10. Optional Configuration Pins

Pin Name	User Mode	Pin Type	Description
CLKUSR	N/A if option is on. I/O if option is off.	Input	Optional user-supplied clock input synchronizes the initialization of one or more devices. Enable this pin by turning on the **Enable user-supplied start-up clock** (**CLKUSR**) option in the Quartus II software.
INIT_DONE	N/A if option is on. I/O if option is off.	Output open-drain	Use as a status pin to indicate when the device has initialized and is in user mode. When nCONFIG is low and during the beginning of configuration, the INIT_DONE pin is tri-stated and pulled high due to an external 10-kΩ pull-up resistor. After the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin goes low. When initialization is complete, the INIT_DONE pin is released and pulled high and the device enters user mode. Thus, the monitoring circuitry must be able to detect a low-to-high transition. Enable this pin by turning on the **Enable INIT_DONE output** option in the Quartus II software.
DEV_OE	N/A if option is on. I/O if option is off.	Input	Optional pin that allows you to override all tri-states on the device. When this pin is driven low, all I/O pins are tri-stated. When this pin is driven high, all I/O pins behave as programmed. Enable this pin by turning on the **Enable device-wide output enable** (**DEV_OE**) option in the Quartus II software.
DEV_CLRn	N/A if option is on. I/O if option is off.	Input	Optional pin that allows you to override all clears on all device registers. When this pin is driven low, all registers are cleared. When this pin is driven high, all registers behave as programmed. Enable this pin by turning on the **Enable device-wide reset** (**DEV_CLRn**) option in the Quartus II software.

Table 10–11 lists the dedicated JTAG pins. JTAG pins must be kept stable before and during configuration to prevent accidental loading of JTAG instructions. The TDI, TMS, and TRST pins have weak internal pull-up resistors, while TCK has a weak internal pull-down resistor (typically 25 kΩ). If you plan to use the SignalTap® embedded logic array analyzer, you must connect the JTAG pins of the Stratix IV device to a JTAG header on your board.

Table 10–11. Dedicated JTAG Pins

Pin Name	User Mode	Pin Type	Description
TDI	N/A	Test data input	Serial input pin for instructions as well as test and programming data. Data is shifted on the rising edge of TCK. The TDI pin is powered by the 2.5-V/3.0-V V_{CCPD} supply. If the JTAG interface is not required on your board, you can disable the JTAG circuitry by connecting this pin to logic high using a 1-kΩ resistor.
TDO	N/A	Test data output	Serial data output pin for instructions as well as test and programming data. Data is shifted out on the falling edge of TCK. The pin is tri-stated if data is not being shifted out of the device. The TDO pin is powered by V_{CCPD}. For recommendations about connecting a JTAG chain with multiple voltages across the devices in the chain, refer to the *JTAG Boundary Scan Testing* chapter. If the JTAG interface is not required on your board, you can disable the JTAG circuitry by leaving this pin unconnected.
TMS	N/A	Test mode select	Input pin that provides the control signal to determine the transitions of the TAP controller state machine. TMS is evaluated on the rising edge of TCK. Therefore, you must set up TMS before the rising edge of TCK. Transitions within the state machine occur on the falling edge of TCK after the signal is applied to TMS. The TMS pin is powered by 2.5-V/3.0-V V_{CCPD}. If the JTAG interface is not required on your board, you can disable the JTAG circuitry by connecting this pin to logic high using a 1-kΩ resistor.
TCK	N/A	Test clock input	Clock input to the BST circuitry. Some operations occur at the rising edge, while others occur at the falling edge. The TCK pin is powered by the 2.5-V/3.0-V V_{CCPD} supply. It is expected that the clock input waveform have a nominal 50% duty cycle. If the JTAG interface is not required on your board, you can disable the JTAG circuitry by connecting TCK to GND.
TRST	N/A	Test reset input (optional)	Active-low input to asynchronously reset the boundary-scan circuit. The TRST pin is optional according to IEEE Std. 1149.1. The TRST pin is powered by the 2.5-V/3.0-V V_{CCPD} supply. Hold TMS at 1 or keep TCK static while TRST is changed from 0 to 1. If the JTAG interface is not required on your board, you can disable the JTAG circuitry by connecting the TRST pin to GND.

For more information about the pin connection recommendations, refer to the *Stratix IV GX Device Family Pin Connection Guidelines*.

Configuration Data Decompression

Stratix IV devices support configuration data decompression, which saves configuration memory space and time. This feature allows you to store compressed configuration data in configuration devices or other memory and transmit this compressed bitstream to Stratix IV devices. During configuration, the Stratix IV device decompresses the bitstream in real time and programs its SRAM cells.

Preliminary data indicates that compression typically reduces the configuration bitstream size by 35 to 55% based on the designs used.

Stratix IV devices support decompression in the FPP (when using a MAX II device or microprocessor + flash), fast AS, and PS configuration schemes. The Stratix IV decompression feature is not available in the JTAG configuration scheme.

In PS mode, use the Stratix IV decompression feature because sending compressed configuration data reduces configuration time.

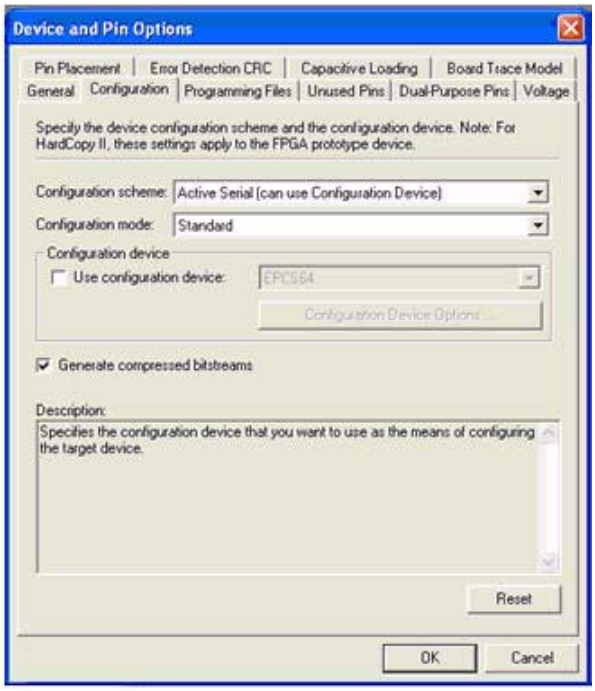
When you enable compression, the Quartus II software generates configuration files with compressed configuration data. This compressed file reduces the storage requirements in the configuration device or flash memory, and decreases the time needed to transmit the bitstream to the Stratix IV device. The time required by a Stratix IV device to decompress a configuration file is less than the time needed to transmit the configuration data to the device.

There are two ways to enable compression for Stratix IV bitstreams: before design compilation (in the Compiler Settings menu) and after design compilation (in the **Convert Programming Files** window).

To enable compression in the project's Compiler Settings menu, perform the following steps:

1. On the Assignments menu, click **Device** to bring up the **Settings** dialog box.
2. After selecting your Stratix IV device, open the **Device and Pin Options** window.
3. In the **Configuration** settings tab, turn on **Generate compressed bitstreams** (as shown in Figure 10–19).

Figure 10–19. Enabling Compression for Stratix IV Bitstreams in Compiler Settings



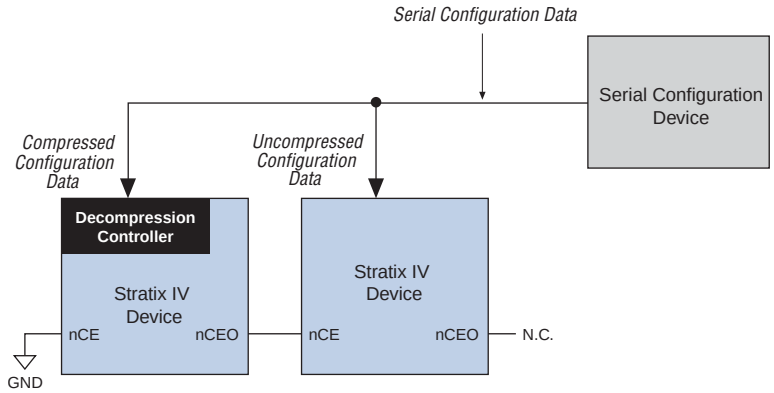
You can also enable compression when creating programming files from the **Convert Programming Files** window. To do this, perform the following steps:

1. On the File menu, click **Convert Programming Files**.
2. Select the programming file type (**.pof**, **.sram**, **.hex**, **.rbf**, or **.ttf**).
3. For **.pof** output files, select a configuration device.
4. In the **Input files to convert** box, select **SOF Data**.
5. Select **Add File** and add a Stratix IV device **.sof** file.
6. Select the name of the file you added to the **SOF Data** area and click **Properties**.
7. Check the **Compression** check box.

When multiple Stratix IV devices are cascaded, you can selectively enable the compression feature for each device in the chain if you are using a serial configuration scheme. Figure 10–20 shows a chain of two Stratix IV devices. The first Stratix IV device has compression enabled; therefore, receives a compressed bitstream from the configuration device. The second Stratix IV device has the compression feature disabled and receives uncompressed data.

In a multi-device FPP configuration chain (with a MAX II device or microprocessor + flash), all Stratix IV devices in the chain must either enable or disable the decompression feature. You cannot selectively enable the compression feature for each device in the chain because of the DATA and DCLK relationship.

Figure 10–20. Compressed and Uncompressed Configuration Data in the Same Configuration File



You can generate programming files for this setup by clicking **Convert Programming Files** on the File menu in the Quartus II software.

Remote System Upgrades

This section describes the functionality and implementation of the dedicated remote system upgrade circuitry. It also defines several concepts related to remote system upgrade, including factory configuration, application configuration, remote update mode, and user watchdog timer. Additionally, this section provides design guidelines for implementing remote system upgrades with the supported configuration schemes.

System designers sometimes face challenges such as shortened design cycles, evolving standards, and system deployments in remote locations. Stratix IV devices help overcome these challenges with their inherent reprogrammability and dedicated circuitry to perform remote system upgrades. Remote system upgrades help deliver feature enhancements and bug fixes without costly recalls, reduce time-to-market, extend product life, and avoid system downtime.

Stratix IV devices feature dedicated remote system upgrade circuitry. Soft logic (either the Nios® II embedded processor or user logic) implemented in a Stratix IV device can download a new configuration image from a remote location, store it in configuration memory, and direct the dedicated remote system upgrade circuitry to initiate a reconfiguration cycle. The dedicated circuitry performs error detection during and after the configuration process, recovers from any error condition by reverting back to a safe configuration image, and provides error status information.

Remote system upgrade is supported in fast AS Stratix IV configuration schemes. You can also implement remote system upgrade in conjunction with advanced Stratix IV features such as real-time decompression of configuration data and design security using the advanced encryption standard (AES) for secure and efficient field upgrades. The largest serial configuration device currently supports 128 Mbits of configuration bitstream.

Stratix IV devices only support remote system upgrade in the single device fast AS configuration scheme. Because the largest serial configuration device currently supports 128 Mbits of configuration bitstream, the remote system upgrade feature is not supported in EP4SGX290, EP4SE360, and larger devices.

The remote system upgrade feature is not supported in a multi-device chain.

Functional Description

The dedicated remote system upgrade circuitry in Stratix IV devices manages remote configuration and provides error detection, recovery, and status information. User logic or a Nios II processor implemented in the Stratix IV device logic array provides access to the remote configuration data source and an interface to the system's configuration memory.

Stratix IV devices have remote system upgrade processes that involve the following steps:

1. A Nios II processor (or user logic) implemented in the Stratix IV device logic array receives new configuration data from a remote location. The connection to the remote source uses a communication protocol such as the transmission control protocol/Internet protocol (TCP/IP), peripheral component interconnect (PCI), user datagram protocol (UDP), universal asynchronous receiver/transmitter (UART), or a proprietary interface.
2. The Nios II processor (or user logic) stores this new configuration data in non-volatile configuration memory.
3. The Nios II processor (or user logic) initiates a reconfiguration cycle with the new or updated configuration data.
4. The dedicated remote system upgrade circuitry detects and recovers from any error(s) that might occur during or after the reconfiguration cycle and provides error status information to the user design.

Figure 10–21 shows the steps required for performing remote configuration updates. (The numbers in Figure 10–21 coincide with the steps just mentioned.)

Figure 10–21. Functional Diagram of Stratix IV Remote System Upgrade

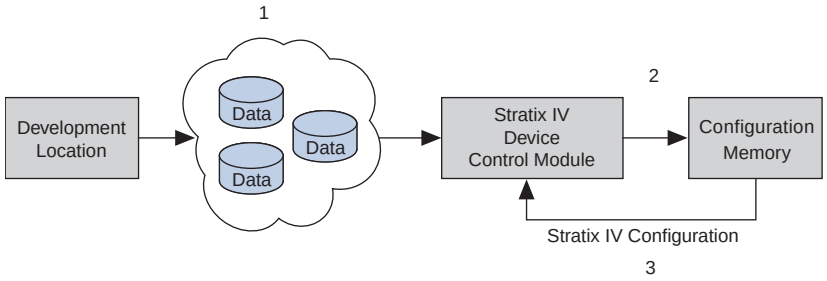
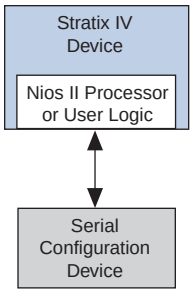


Figure 10–22 shows a block diagram for implementing a remote system upgrade with the Stratix IV fast AS configuration scheme.

Figure 10–22. Remote System Upgrade Block Diagram for Stratix IV Fast AS Configuration Scheme



You must set the mode select pins (`MSEL[2..0]`) to fast AS mode to use remote system upgrade in your system. Table 10–12 lists the `MSEL` pin settings for Stratix IV devices in standard configuration mode and remote system upgrade mode. The following sections describe remote update of the remote system upgrade mode.

For more information about standard configuration schemes supported in Stratix IV devices, refer to "Configuration Schemes" on page 10–2.

Table 10–12. Stratix IV Remote System Upgrade Modes

Configuration Scheme	MSEL[2..0]	Remote System Upgrade Mode
Fast AS (40 MHz)	011	Standard
	011	Remote update *(1)*

Note to Table 10–12:

(1) EPCS64 and EPCS128 serial configuration devices support a `DCLK` up to 40 MHz. For more information, refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* chapter in volume 2 of the *Configuration Handbook*.

When using fast AS mode, you must select remote update mode in the Quartus II software and insert the ALTREMOTE_UPDATE megafunction to access the circuitry. For more information, refer to "ALTREMOTE_UPDATE Megafunction" on page 10–58.

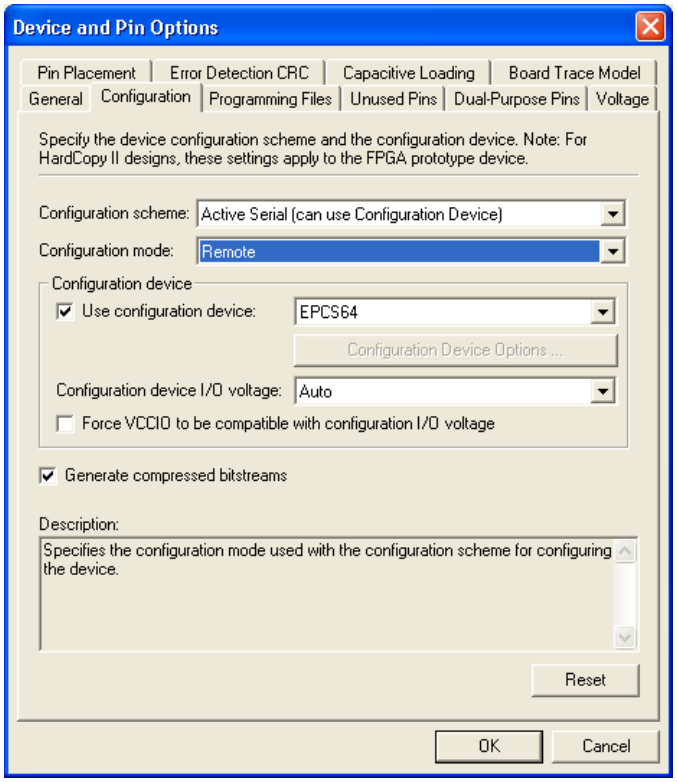
Enabling Remote Update

You can enable remote update for Stratix IV devices in the Quartus II software before design compilation (in the Compiler Settings menu). In remote update mode, the **auto-restart configuration after error** option is always enabled. To enable remote update in the project's compiler settings, perform the following steps in the Quartus II software:

1. On the Assignment menu, click **Device**. The **Settings** dialog box appears.
2. Click **Device and Pin Options**. The **Device and Pin Options** dialog box appears.
3. Click the **Configuration** tab.
4. From the **Configuration scheme** list, select **Active Serial** (you can also use **Configuration Device**) (Figure 10–23).

5. From the **Configuration Mode** list, select **Remote** (Figure 10–23).
6. Click **OK**.
7. In the **Settings** dialog box, click **OK**.

Figure 10–23. Enabling Remote Update for Stratix IV Devices in the Compiler Settings Menu



Configuration Image Types

When performing a remote system upgrade, Stratix IV device configuration bitstreams are classified as factory configuration images or application configuration images. An image, also referred to as a configuration, is a design loaded into the Stratix IV device that performs certain user-defined functions.

Each Stratix IV device in your system requires one factory image or the addition of one or more application images. The factory image is a user-defined fall-back, or safe configuration, and is responsible for administering remote updates in conjunction with the dedicated circuitry. Application images implement user-defined functionality in the target Stratix IV device. You may include the default application image functionality in the factory image.

A remote system upgrade involves storing a new application configuration image or updating an existing one using the remote communication interface. After an application configuration image is stored or updated remotely, the user design in the Stratix IV device initiates a reconfiguration cycle with the new image. Any errors during or after this cycle are detected by the dedicated remote system upgrade

circuitry and cause the device to automatically revert to the factory image. The factory image then performs error processing and recovery. The factory configuration is written to the serial configuration device only once by the system manufacturer and must not be remotely updated. On the other hand, application configurations may be remotely updated in the system. Both images can initiate system reconfiguration.

Remote System Upgrade Mode

Remote system upgrade has one mode of operation—remote update mode. Remote update mode allows you to determine the functionality of your system upon power-up and offers several features.

Remote Update Mode

In remote update mode, Stratix IV devices load the factory configuration image after power up. The user-defined factory configuration determines which application configuration is to be loaded and triggers a reconfiguration cycle. The factory configuration may also contain application logic.

When used with serial configuration devices, remote update mode allows an application configuration to start at any flash sector boundary. For example, this translates to a maximum of 128 sectors in the EPCS64 device and 32 sectors in the EPCS16 device, where the minimum size of each page is 512 KBits. Altera recommends not using the same page in the serial configuration devices for two images. Additionally, remote update mode features a user watchdog timer that determines the validity of an application configuration.

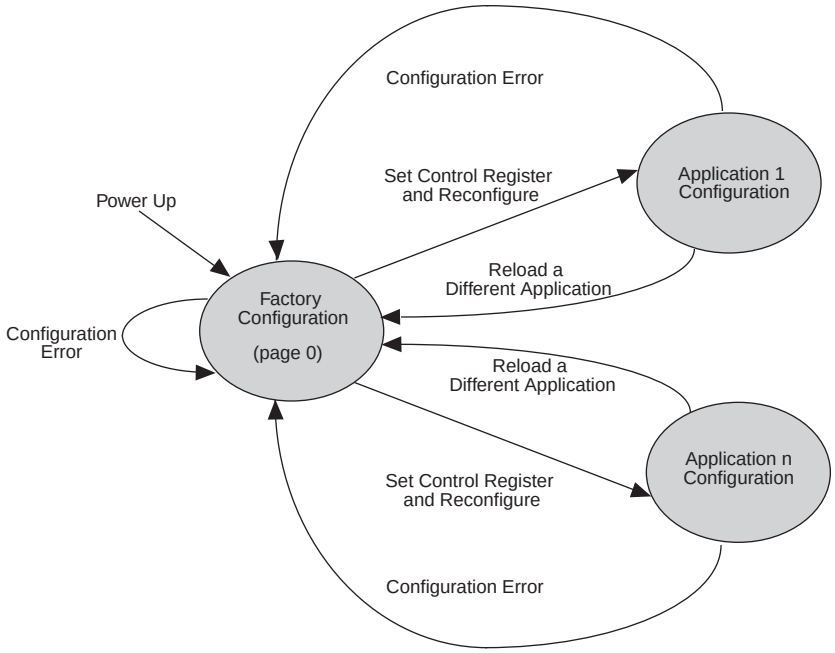
When a Stratix IV device is first powered up in remote update mode, it loads the factory configuration located at page zero (page registers `PGM[23:0] = 24'b0`). Always store the factory configuration image for your system at page address zero. This corresponds to the start address location 0×000000 in the serial configuration device.

The factory image is user-designed and contains soft logic to:

- Process any errors based on status information from the dedicated remote system upgrade circuitry
- Communicate with the remote host and receive new application configurations and store this new configuration data in the local non-volatile memory device
- Determine which application configuration is to be loaded into the Stratix IV device
- Enable or disable the user watchdog timer and load its time-out value (optional)
- Instruct the dedicated remote system upgrade circuitry to initiate a reconfiguration cycle

Figure 10–24 shows the transitions between the factory and application configurations in remote update mode.

Figure 10–24. Transitions between Configurations in Remote Update Mode



After power up or a configuration error, the factory configuration logic is loaded automatically. The factory configuration also must specify whether to enable the user watchdog timer for the application configuration and if enabled, to include the timer setting information.

The user watchdog timer ensures that the application configuration is valid and functional. The timer must be continually reset within a specific amount of time during user mode operation of an application configuration. Only valid application configurations contain the logic to reset the timer in user mode. This timer reset logic must be part of a user-designed hardware and/or software health monitoring signal that indicates error-free system operation. If the timer is not reset in a specific amount of time; for example, the user application configuration detects a functional problem or if the system hangs, the dedicated circuitry updates the remote system upgrade status register, triggering the loading of the factory configuration.

☞ The user watchdog timer is automatically disabled for factory configurations. For more information about the user watchdog timer, refer to "User Watchdog Timer" on page 10–57.

If there is an error while loading the application configuration, the cause of the reconfiguration is written by the dedicated circuitry to the remote system upgrade status register. Actions that cause the remote system upgrade status register to be written are:

- nSTATUS driven low externally
- Internal CRC error

- User watchdog timer time-out
- A configuration reset (logic array `nCONFIG` signal or external `nCONFIG` pin assertion to low)

Stratix IV devices automatically load the factory configuration located at page address zero. This user-designed factory configuration can read the remote system upgrade status register to determine the reason for the reconfiguration. The factory configuration then takes appropriate error recovery steps and writes to the remote system upgrade control register to determine the next application configuration to be loaded.

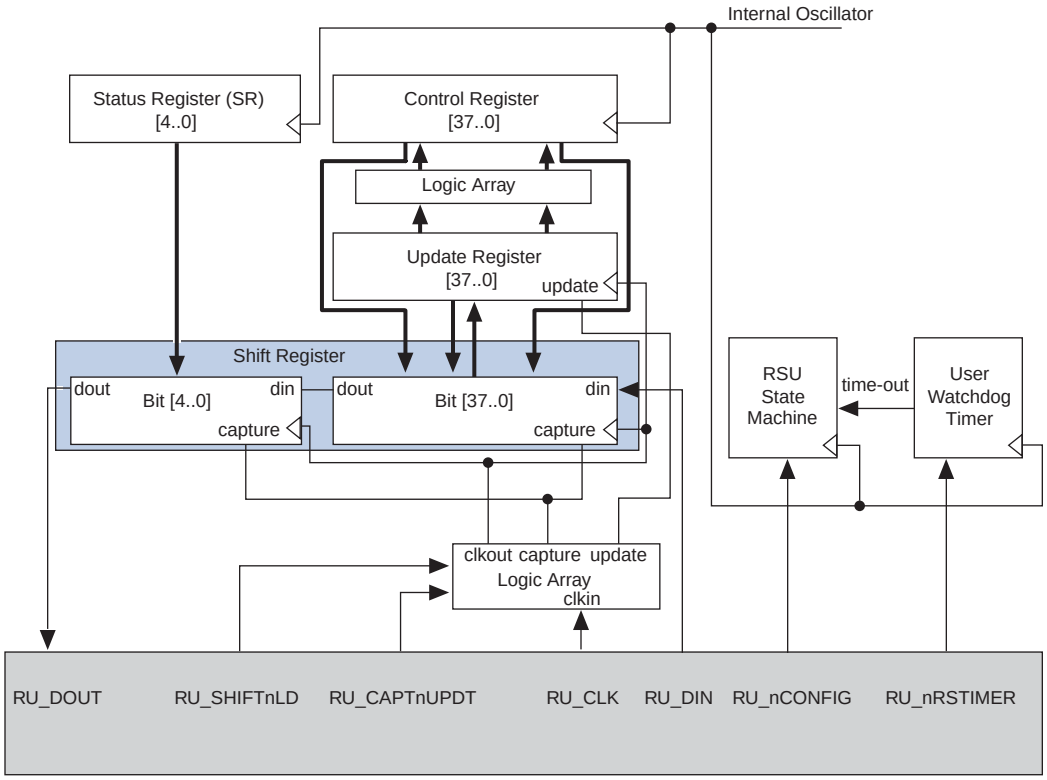
When Stratix IV devices successfully load the application configuration, they enter into user mode. In user mode, the soft logic (Nios II processor or state machine and the remote communication interface) assists the Stratix IV device in determining when a remote system update is arriving. When a remote system update arrives, the soft logic receives the incoming data, writes it to the configuration memory device, and triggers the device to load the factory configuration. The factory configuration reads the remote system upgrade status register and control register, determines the valid application configuration to load, writes the remote system upgrade control register accordingly, and initiates system reconfiguration.

Dedicated Remote System Upgrade Circuitry

This section describes the implementation of the Stratix IV remote system upgrade dedicated circuitry. The remote system upgrade circuitry is implemented in hard logic. This dedicated circuitry interfaces to the user-defined factory and application configurations implemented in the Stratix IV device logic array to provide the complete remote configuration solution. The remote system upgrade circuitry contains the remote system upgrade registers, a watchdog timer, and a state machine that controls those components.

Figure 10–25 shows the data path for the remote system upgrade block.

Figure 10–25. Remote System Upgrade Circuit Data Path *(Note 1)*



Note to Figure 10–25:

(1) The `RU_DOUT`, `RU_SHIFTnLD`, `RU_CAPTnUPDT`, `RU_CLK`, `RU_DIN`, `RU_nCONFIG`, and `RU_nRSTIMER` signals are internally controlled by the ALTREMOTE_UPDATE megafunction.

Remote System Upgrade Registers

The remote system upgrade block contains a series of registers that store the page addresses, watchdog timer settings, and status information. Table 10–13 lists these registers.

Table 10–13. Remote System Upgrade Registers (Part 1 of 2)

Register	Description
Shift register	This register is accessible by the logic array and allows the update, status, and control registers to be written and sampled by user logic.
Control register	This register contains the current page address, user watchdog timer settings, and one bit specifying whether the current configuration is a factory configuration or an application configuration. During a read operation in an application configuration, this register is read into the shift register. When a reconfiguration cycle is initiated, the contents of the update register are written into the control register.

Table 10–13. Remote System Upgrade Registers (Part 2 of 2)

Register	Description
Update register	This register contains data similar to that in the control register. However, it can only be updated by the factory configuration by shifting data into the shift register and issuing an update operation. When a reconfiguration cycle is triggered by the factory configuration, the control register is updated with the contents of the update register. During a capture in a factory configuration, this register is read into the shift register.
Status register	This register is written to by the remote system upgrade circuitry on every reconfiguration to record the cause of the reconfiguration. This information is used by the factory configuration to determine the appropriate action following a reconfiguration. During a capture cycle, this register is read into the shift register.

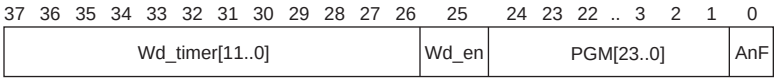
The remote system upgrade control and status registers are clocked by the 10-MHz internal oscillator (the same oscillator that controls the user watchdog timer). However, the remote system upgrade shift and update registers are clocked by the user clock input (`RU_CLK`).

Remote System Upgrade Control Register

The remote system upgrade control register stores the application configuration page address and user watchdog timer settings. The control register functionality depends on the remote system upgrade mode selection. In remote update mode, the control register page address bits are set to all zeros (`24'b0 = 0×000000`) at power up to load the factory configuration. A factory configuration in remote update mode has write access to this register.

Figure 10–26 and Table 10–14 specify the control register bit positions. In the figure, the numbers show the bit position of a setting within a register. For example, bit number 25 is the enable bit for the watchdog timer.

Figure 10–26. Remote System Upgrade Control Register



The application-not-factory (`AnF`) bit indicates whether the current configuration loaded in the Stratix IV device is the factory configuration or an application configuration. This bit is set low by the remote system upgrade circuitry when an error condition causes a fall-back to the factory configuration. When the `AnF` bit is high, the control register access is limited to read operations. When the `AnF` bit is low, the register allows write operations and disables the watchdog timer.

In remote update mode, the factory configuration design sets this bit high (1'b1) when updating the contents of the update register with the application page address and watchdog timer settings.

Table 10–14 lists the remote system upgrade control register contents.

Table 10–14. Remote System Upgrade Control Register Contents

Control Register Bit	Remote System Upgrade Mode	Value *(2)*	Definition
AnF *(1)*	Remote update	1'b0	Application not factory
PGM[23..0]	Remote update	24'b0×000000	AS configuration start address (StAdd[23..0])
Wd_en	Remote update	1'b0	User watchdog timer enable bit
Wd_timer[11..0]	Remote update	12'b000000000000	User watchdog time-out value (most significant 12 bits of 29-bit count value: {Wd_timer[11..0], 17'b0})

Notes to Table 10–14:

(1) In remote update mode, the remote configuration block does not update the AnF bit automatically (you can update it manually).

(2) This is the default value of the control register bit.

Remote System Upgrade Status Register

The remote system upgrade status register specifies the reconfiguration trigger condition. The various trigger and error conditions include:

- Cyclic redundancy check (CRC) error during application configuration
- nSTATUS assertion by an external device due to an error
- Stratix IV device logic array triggered a reconfiguration cycle, possibly after downloading a new application configuration image
- External configuration reset (nCONFIG) assertion
- User watchdog timer time-out

Figure 10–27 and Table 10–15 specify the contents of the status register. The numbers in the figure show the bit positions within a 5-bit register.

Figure 10–27. Remote System Upgrade Status Register

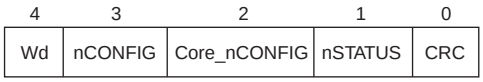


Table 10–15. Remote System Upgrade Status Register Contents (Part 1 of 2)

Status Register Bit	Definition	POR Reset Value
CRC (from the configuration)	CRC error caused reconfiguration	1 bit '0'
nSTATUS	nSTATUS caused reconfiguration	1 bit '0'
CORE_nCONFIG *(1)*	Device logic array caused reconfiguration	1 bit '0'
nCONFIG	nCONFIG caused reconfiguration	1 bit '0'

Table 10–15. Remote System Upgrade Status Register Contents (Part 2 of 2)

Status Register Bit	Definition	POR Reset Value
wd	Watchdog timer caused reconfiguration	1 bit '0'

Note to Table 10–15:

(1) Logic array reconfiguration forces the system to load the application configuration data into the Stratix IV device. This occurs after the factory configuration specifies the appropriate application configuration page address by updating the update register.

Remote System Upgrade State Machine

The remote system upgrade control and update registers have identical bit definitions, but serve different roles (refer to Table 10–13 on page 10–53). While both registers can only be updated when the device is loaded with a factory configuration image, the update register writes are controlled by the user logic; the control register writes are controlled by the remote system upgrade state machine.

In factory configurations, the user logic sends the AnF bit (set high), the page address, and the watchdog timer settings for the next application configuration bit to the update register. When the logic array configuration reset (RU_nCONFIG) goes low, the remote system upgrade state machine updates the control register with the contents of the update register and initiates system reconfiguration from the new application page.

To ensure successful reconfiguration between the pages, assert the RU_nCONFIG signal for a minimum of 250 ns. This is equivalent to strobing the reconfiguration input of the ALTREMOTE_UPDATE megafunction high for a minimum of 250 ns.

In the event of an error or reconfiguration trigger condition, the remote system upgrade state machine directs the system to load a factory or application configuration (page zero or page one, based on the mode and error condition) by setting the control register accordingly. Table 10–16 lists the contents of the control register after such an event occurs for all possible error or trigger conditions.

The remote system upgrade status register is updated by the dedicated error monitoring circuitry after an error condition but before the factory configuration is loaded.

Table 10–16. Control Register Contents after an Error or Reconfiguration Trigger Condition

Reconfiguration Error/Trigger	Control Register Setting Remote Update
nCONFIG reset	All bits are 0
nSTATUS error	All bits are 0
CORE triggered reconfiguration	Update register
CRC error	All bits are 0
wd time out	All bits are 0

Capture operations during factory configuration access the contents of the update register. This feature is used by the user logic to verify that the page address and watchdog timer settings were written correctly. Read operations in application configurations access the contents of the control register. This information is used by the user logic in the application configuration.

User Watchdog Timer

The user watchdog timer prevents a faulty application configuration from stalling the device indefinitely. The system uses the timer to detect functional errors after an application configuration is successfully loaded into the Stratix IV device.

The user watchdog timer is a counter that counts down from the initial value loaded into the remote system upgrade control register by the factory configuration. The counter is 29 bits wide and has a maximum count value of 2^{29}. When specifying the user watchdog timer value, specify only the most significant 12 bits. The granularity of the timer setting is 2^{15} cycles. The cycle time is based on the frequency of the 10-MHz internal oscillator. Table 10–17 lists the operating range of the 10-MHz internal oscillator.

Table 10–17. 10-MHz Internal Oscillator Specifications *(Note 1)*

Minimum	Typical	Maximum	Units
4.3	5.3	10	MHz

Note to Table 10–17:

(1) These values are preliminary.

The user watchdog timer begins counting once the application configuration enters device user mode. This timer must be periodically reloaded or reset by the application configuration before the timer expires by asserting `RU_nRSTIMER`. If the application configuration does not reload the user watchdog timer before the count expires, a time-out signal is generated by the remote system upgrade dedicated circuitry. The time-out signal tells the remote system upgrade circuitry to set the user watchdog timer status bit (`Wd`) in the remote system upgrade status register and reconfigures the device by loading the factory configuration.

To allow remote system upgrade dedicated circuitry to reset the watchdog timer, you must assert the `RU_nRSTIMER` signal active for a minimum of 250 ns. This is equivalent to strobing the `reset_timer` input of the ALTREMOTE_UPDATE megafunction high for a minimum of 250 ns.

The user watchdog timer is not enabled during the configuration cycle of the device. Errors during configuration are detected by the CRC engine. Also, the timer is disabled for factory configurations. Functional errors should not exist in the factory configuration because it is stored and validated during production and is never updated remotely.

The user watchdog timer is disabled in factory configurations and during the configuration cycle of the application configuration. It is enabled after the application configuration enters user mode.

Quartus II Software Support

The Quartus II software provides the flexibility to include the remote system upgrade interface between the Stratix IV device logic array and the dedicated circuitry, generate configuration files for production, and allows remote programming of the system configuration memory.

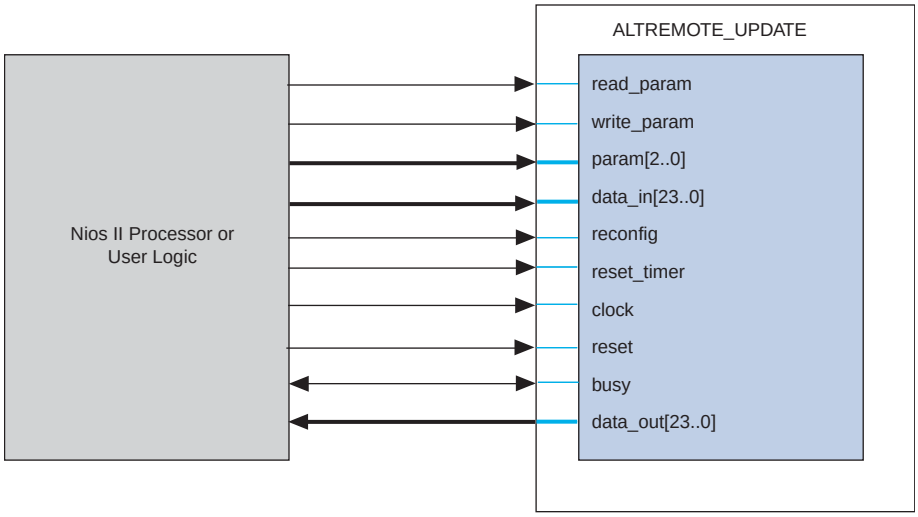
The ALTREMOTE_UPDATE megafunction is the implementation option in the Quartus II software that you use for the interface between the remote system upgrade circuitry and the device logic array interface. Using the megafunction block instead of creating your own logic saves design time and offers more efficient logic synthesis and device implementation.

ALTREMOTE_UPDATE Megafunction

The ALTREMOTE_UPDATE megafunction provides a memory-like interface to the remote system upgrade circuitry and handles the shift register read and write protocol in the Stratix IV device logic. This implementation is suitable for designs that implement the factory configuration functions using a Nios II processor or user logic in the device.

Figure 10–28 shows the interface signals between the ALTREMOTE_UPDATE megafunction and Nios II processor or user logic.

Figure 10–28. Interface Signals between the ALTREMOTE_UPDATE Megafunction and the Nios II Processor



For more information about the ALTREMOTE_UPDATE megafunction and the description of ports listed in Figure 10–28, refer to the *ALTREMOTE_UPDATE Megafunction User Guide*.

Design Security

This section provides an overview of the design security feature and its implementation on Stratix IV devices using the advanced encryption standard (AES). It also covers the new security modes available in Stratix IV devices.

As Stratix IV devices continue play a role in larger and more critical designs in competitive commercial and military environments, it is increasingly important to protect the designs from copying, reverse engineering, and tampering.

Stratix IV devices address these concerns with both volatile and non-volatile security feature support. Stratix IV devices have the ability to decrypt configuration bitstreams using the AES algorithm, an industry-standard encryption algorithm that is FIPS-197 certified. Stratix IV devices have a design security feature that utilizes a 256-bit security key.

Stratix IV devices store configuration data in SRAM configuration cells during device operation. Because SRAM is volatile, the SRAM cells must be loaded with configuration data each time the device powers up. It is possible to intercept configuration data when it is being transmitted from the memory source (flash memory or a configuration device) to the device. The intercepted configuration data could then be used to configure another device.

When using the Stratix IV design security feature, the security key is stored in the Stratix IV device. Depending on the security mode, you can configure the Stratix IV device using a configuration file that is encrypted with the same key, or for board testing, configured with a normal configuration file.

The design security feature is available when configuring Stratix IV devices using FPP configuration mode with an external host (such as a MAX II device or microprocessor), or when using fast AS or PS configuration schemes. The design security feature is also available in remote update with fast AS configuration mode. The design security feature is not available when you are configuring your Stratix IV device using JTAG-based configuration. For more information, refer to "Supported Configuration Schemes" on page 10–63.

When using a serial configuration scheme such as PS or fast AS, configuration time is the same whether or not you enable the design security feature. If the FPP scheme is used with the design security or decompression feature, a ×4 `DCLK` is required. This results in a slower configuration time when compared with the configuration time of a Stratix IV device that has neither the design security nor the decompression feature enabled.

Stratix IV Security Protection

Stratix IV device designs are protected from copying, reverse engineering, and tampering using configuration bitstream encryption.

Security Against Copying

The security key is securely stored in the Stratix IV device and cannot be read out through any interfaces. In addition, as configuration file read-back is not supported in Stratix IV devices, the design information cannot be copied.

Security Against Reverse Engineering

Reverse engineering from an encrypted configuration file is very difficult and time consuming because the Stratix IV configuration file formats are proprietary and the file contains millions of bits which require specific decryption. Reverse engineering the Stratix IV device is just as difficult because the device is manufactured on the most advanced 40-nm process technology.

Security Against Tampering

The non-volatile keys are one-time programmable. After the Tamper Protection bit is set in the key programming file generated by the Quartus II software, the Stratix IV device can only be configured with configuration files encrypted with the same key.

AES Decryption Block

The main purpose of the AES decryption block is to decrypt the configuration bitstream prior to entering data decompression or configuration.

Prior to receiving encrypted data, you must enter and store the 256-bit security key in the device. You can choose between a non-volatile security key and a volatile security key with battery backup.

The security key is scrambled prior to storing it in the key storage to make it more difficult for anyone to retrieve the stored key using de-capsulation of the device.

Flexible Security Key Storage

Stratix IV devices support two types of security key programming—volatile and non-volatile keys. Table 10–18 lists the differences between volatile keys and non-volatile keys.

Table 10–18. Security Key Options

Options	Volatile Key	Non-Volatile Key
Key programmability	Reprogrammable and erasable	One-time programmable
External battery	Required	Not required
Key programming method *(1)*	On-board	On and off board
Design protection	Secure against copying and reverse engineering	Secure against copying and reverse engineering. Tamper resistant if tamper protection bit is set.

Note to Table 10–18:

(1) Key programming is carried out using the JTAG interface.

You can program the non-volatile key to the Stratix IV device without an external battery. Also, there are no additional requirements to any of the Stratix IV power supply inputs.

V_{CCBAT} is a dedicated power supply for volatile key storage and not shared with other on-chip power supplies, such as V_{CCIO} or V_{CC}. V_{CCBAT} continuously supplies power to the volatile register regardless of the on-chip supply condition.

After power-up, you must wait 300 ms (PORSEL = 0) or 12 ms (PORSEL = 1) before beginning key programming to ensure that V_{CCBAT} is at full rail.

For more information about how to calculate the key retention time of the battery used for volatile key storage, refer to the *Stratix IV PowerPlay Early Power Estimator*.

For more information about battery specifications, refer to the *DC and Switching Characteristics* chapter.

For more information about the V_{CCBAT} pin connection recommendations, refer to the *Stratix IV GX Device Family Pin Connection Guidelines*.

Stratix IV Design Security Solution

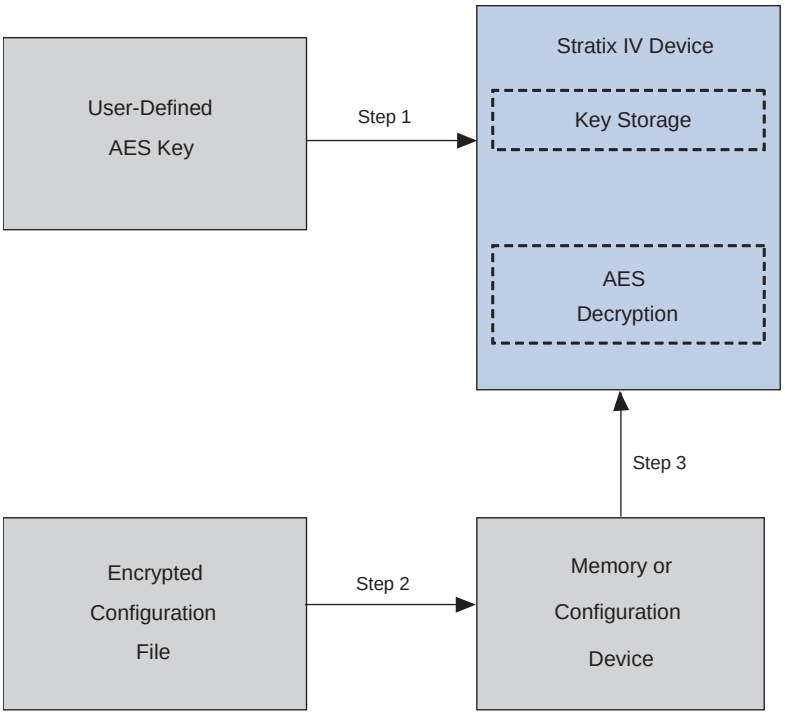
Stratix IV devices are SRAM-based devices. To provide design security, Stratix IV devices require a 256-bit security key for configuration bitstream encryption.

You can carry out secure configuration in the following steps, as shown in Figure 10–29:

1. Program the security key into the Stratix IV device.
2. Program the user-defined 256-bit AES keys to the Stratix IV device through the JTAG interface.
3. Encrypt the configuration file and store it in the external memory.
4. Encrypt the configuration file with the same 256-bit keys used to program the Stratix IV device. Encryption of the configuration file is done using the Quartus II software. The encrypted configuration file is then loaded into the external memory, such as a configuration or flash device.
5. Configure the Stratix IV device.

At system power-up, the external memory device sends the encrypted configuration file to the Stratix IV device.

Figure 10–29. Design Security *(Note 1)*



Note to Figure 10–29:

(1) Step 1, Step 2, and Step 3 correspond to the procedure described in "Design Security" on page 10–58.

Security Modes Available

The following security modes are available on the Stratix IV device:

Volatile Key

Secure operation with volatile key programmed and required external battery: this mode accepts both encrypted and unencrypted configuration bitstreams. Use the unencrypted configuration bitstream support for board-level testing only.

Non-Volatile Key

Secure operation with one time programmable (OTP) security key programmed: this mode accepts both encrypted and unencrypted configuration bitstreams. Use the unencrypted configuration bitstream support for board level testing only.

Non-Volatile Key with Tamper Protection Bit Set

Secure operation in tamper resistant mode with OTP security key programmed: only encrypted configuration bitstreams are allowed to configure the device. Tamper protection disables JTAG configuration with unencrypted configuration bitstream.

☞ Enabling the tamper protection bit disables test mode in Stratix IV devices. This process is irreversible and prevents Altera from conducting carry-out failure analysis if test mode is disabled. Contact Altera Technical Support to enable the tamper protection bit.

No Key Operation

Only unencrypted configuration bitstreams are allowed to configure the device.

Table 10–19 lists the different security modes and configuration bitstream supported for each mode.

Table 10–19. Security Modes Supported

Mode *(1)*	Function	Configuration File
Volatile key	Secure	Encrypted
	Board-level testing	Unencrypted
Non-volatile key	Secure	Encrypted
	Board-level testing	Unencrypted
Non-volatile key with tamper protection bit set	Secure (tamper resistant) *(2)*	Encrypted

Notes to Table 10–19:

(1) In No key operation, only the unencrypted configuration file is supported.

(2) The tamper protection bit setting does not prevent the device from being reconfigured.

Supported Configuration Schemes

The Stratix IV device supports only selected configuration schemes, depending on the security mode you select when you encrypt the Stratix IV device.

Figure 10–30 shows the restrictions of each security mode when encrypting Stratix IV devices.

Figure 10–30. Stratix IV Security Modes—Sequence and Restrictions

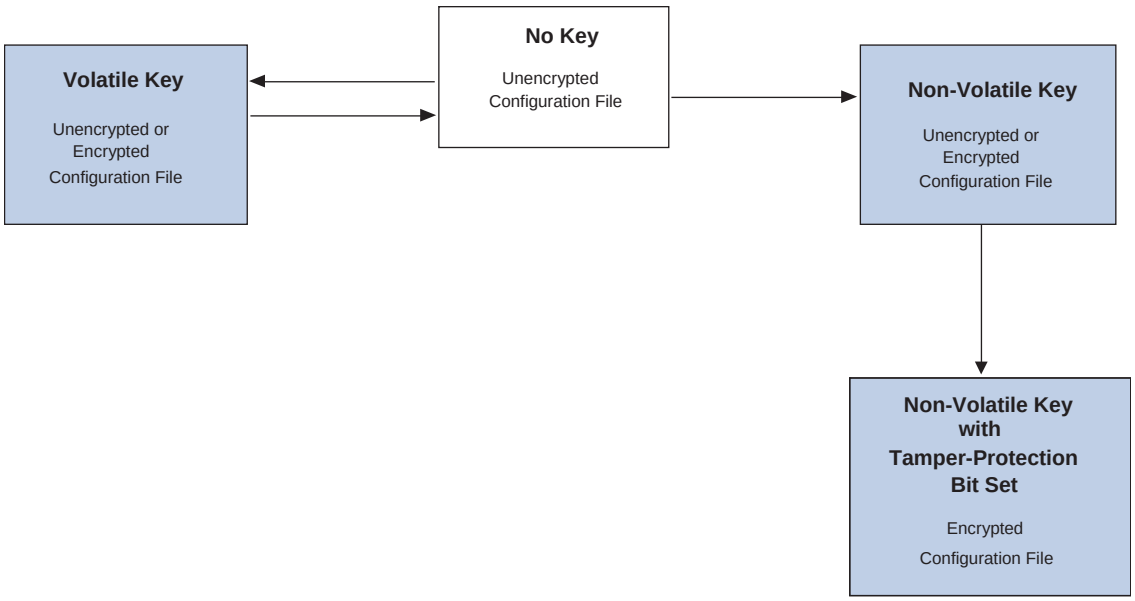


Table 10–20 lists the configuration modes allowed in each of the security modes.

Table 10–20. Allowed Configuration Modes for Various Security Modes (Part 1 of 2) *(Note 1)*

Security Mode	Configuration File	Allowed Configuration Modes
No key	Unencrypted	All configuration modes that do not engage the design security feature.
Secure with volatile key	Encrypted	■ Passive serial with AES (and/or with decompression) ■ Fast passive parallel with AES (and/or with decompression) ■ Remote update fast AS with AES (and/or with decompression) ■ Fast AS (and/or with decompression)
Board-level testing with volatile key	Unencrypted	All configuration modes that do not engage the design security feature.
Secure with non-volatile key	Encrypted	■ Passive serial with AES (and/or with decompression) ■ Fast passive parallel with AES (and/or with decompression) ■ Remote update fast AS with AES (and/or with decompression) ■ Fast AS (and/or with decompression)
Board-level testing with non-volatile key	Unencrypted	All configuration modes that do not engage the design security feature.

Table 10–20. Allowed Configuration Modes for Various Security Modes (Part 2 of 2) *(Note 1)*

Security Mode	Configuration File	Allowed Configuration Modes
Secure in tamper resistant mode using non-volatile key with tamper protection set	Encrypted	■ Passive serial with AES (and/or with decompression) ■ Fast passive parallel with AES (and/or with decompression) ■ Remote update fast AS with AES (and/or with decompression) ■ Fast AS (and/or with decompression)

Note to Table 10–20:

(1) There is no impact to the configuration time required when compared with unencrypted configuration modes except FPP with AES (and/or decompression), which requires a DCLK that is ×4 the data rate.

You can use the design security feature with other configuration features, such as compression and remote system upgrade features. When you use compression with the design security feature, the configuration file is first compressed and then encrypted using the Quartus II software. During configuration, the Stratix IV device first decrypts and then decompresses the configuration file.

Document Revision History

Table 10–21 shows the revision history for this chapter.

Table 10–21. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
November 2009 v3.0	■ Updated Table 10–1 and Table 10–2. ■ Updated the "FPP Configuration Using a MAX II Device as an External Host","Fast Active Serial Configuration (Serial Configuration Devices)", "Device Configuration Pins", "Remote System Upgrades", "Remote System Upgrade Mode", "Estimating Active Serial Configuration Time", "Remote System Upgrade State Machine", and "User Watchdog Timer" sections. ■ Removed Table 10-4, Table 10-7, Table 10-8, and Table 10-25. ■ Minor text edits.	—
June 2009 v2.3	■ Updated the "VCCPD Pins", "FPP Configuration Using a MAX II Device as an External Host", "Estimating Active Serial Configuration Time", "Fast Active Serial Configuration (Serial Configuration Devices)", "Remote System Upgrades", "PS Configuration Using a MAX II Device as an External Host", and "PS Configuration Using a Download Cable" sections. ■ Updated Table 10–3, Table 10–13 and Table 10–2. ■ Added introductory sentences to improve search ability. ■ Removed the Conclusion section. ■ Minor text edits.	—
April 2009 v2.2	■ Updated Table 10–2.	—
March 2009 v2.1	■ Updated Table 10–1, Table 10–2, and Table 10–9. ■ Removed "Referenced Documents" section.	—

Table 10–21. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
November 2008 v2.0	■ Updated “Fast Active Serial Configuration (Serial Configuration Devices)” and “JTAG Configuration” sections. ■ Updated Figure 10–4, Figure 10–5, Figure 10–6, and Figure 10–13. ■ Updated Table 10–2 and Table 10–13.	Medium update.
May 2008 v1.0	Initial release.	—

11. SEU Mitigation in Stratix IV Devices

SIV51011-3.0

This chapter describes how to use the error detection cyclical redundancy check (CRC) feature when a Stratix® IV device is in user mode and recovers from CRC errors. The purpose of the error detection CRC feature in the Stratix IV device is to detect a flip in any of the configuration random access memory (CRAM) bits in Stratix IV devices due to a soft error. With the error detection circuitry, you can continuously verify the integrity of the configuration CRAM bits.

In critical applications such as avionics, telecommunications, system control, and military applications, it is important to be able to do the following:

- Confirm that the configuration data stored in a Stratix IV device is correct
- Alert the system to the occurrence of a configuration error

The error detection feature is enhanced in the Stratix IV device family. Similar to Stratix III devices, the error detection and recovery time for single-event upset (SEU) in Stratix IV devices is reduced when compared with Stratix II devices.

For more information about test methodology for enhanced error detection in Stratix IV devices, refer to *AN 539: Test Methodology of Error Detection and Recovery using CRC in Altera FPGA Devices*.

Dedicated circuitry is built into Stratix IV devices and consists of a CRC error detection feature that optionally checks for SEUs continuously and automatically.

For Stratix IV devices, the error detection CRC feature is provided in the Quartus® II software version 8.0 and onwards.

Using error detection CRC for the Stratix IV device family has no impact on fitting or performance of your device.

This chapter contains the following sections:

- "Error Detection Fundamentals" on page 11–2
- "Configuration Error Detection" on page 11–2
- "User Mode Error Detection" on page 11–2
- "Error Detection Pin Description" on page 11–5
- "Error Detection Block" on page 11–6
- "Error Detection Timing" on page 11–8
- "Recovering From CRC Errors" on page 11–11

Error Detection Fundamentals

Error detection determines whether the data received is corrupted during transmission. To accomplish this, the transmitter uses a function to calculate a checksum value for the data and appends the checksum to the original data frame. The receiver uses the same calculation methodology to generate a checksum for the received data frame and compares the received checksum to the transmitted checksum. If the two checksum values are equal, the received data frame is correct and no data corruption occurred during transmission or storage.

The error detection CRC feature uses the same concept. When Stratix IV devices are configured successfully and are in user mode, the error detection CRC feature ensures the integrity of the configuration data.

There are two CRC error checks. One CRC error check always runs during configuration and a second optional CRC error check runs in the background in user mode. Both CRC error checks use the same CRC polynomial but different error detection implementations. For more information, refer to "Configuration Error Detection" on page 11–2 and "User Mode Error Detection" on page 11–2.

Configuration Error Detection

In configuration mode, a frame-based CRC is stored within the configuration data and contains the CRC value for each data frame.

During configuration, the Stratix IV device calculates the CRC value based on the frame of data that is received and compares it against the frame CRC value in the data stream. Configuration continues until either the device detects an error or configuration is completed.

In Stratix IV devices, the CRC value is calculated during the configuration stage. A parallel CRC engine generates 16 CRC check bits per frame and then stores them in CRAM. The CRAM chain used for storing the CRC check bits is 16 bits wide and its length is equal to the number of frames in the device.

User Mode Error Detection

Stratix IV devices have built-in error detection circuitry to detect data corruption by soft errors in the CRAM cells. This feature allows all CRAM contents to be read and verified to match a configuration-computed CRC value. Soft errors are changes in a CRAM bit state due to an ionizing particle.

The error detection capability continuously computes the CRC of the configured CRAM bits and compares it with the pre-calculated CRC. If the CRCs match, there is no error in the current configuration CRAM bits. The process of error detection continues until the device is reset (by setting `nCONFIG` low).

If you enable the **CRC error detection** option in the Quartus II software, after the device transitions into user mode, the error detection process is enabled. The internal 100 MHz configuration oscillator is divided down by a factor of two to 256 (at powers of two) to be used as the clock source during the error detection process. You must set the clock divide factor in the Quartus II software.

A single 16-bit CRC calculation is done on a per-frame basis. After it has finished the CRC calculation for a frame, the resulting 16-bit signature is `hex 0000` if there are no CRAM bit errors detected in a frame by the error detection circuitry and the output signal `CRC_ERROR` is `0`. If a CRAM bit error is detected by the circuitry within a frame in the device, the resulting signature is non-zero. This causes the CRC engine to start searching for the error bit location.

Error detection in Stratix IV devices calculates CRC check bits for each frame and pulls the `CRC_ERROR` pin high when it detects bit errors in the chip. Within a frame, it can detect all single-bit, double-bit, and three-bit errors. The probability of more than three CRAM bits being flipped by an SEU event is very low. In general, for all error patterns the probability of detection is 99.998%.

The CRC engine reports the bit location and determines the type of error for all single-bit errors and over 99.641% of double-adjacent errors. The probability of other error patterns is very low and report of the location of bit flips is not guaranteed by the CRC engine.

You can also read-out the error bit location through the JTAG and the core interface. Shift these bits out through either the `SHIFT_EDERROR_REG` JTAG instruction or the core interface, before the CRC detects the next error in another frame. If the next frame also has an error, you must shift these bits out within the amount of time of one frame CRC verification. You can choose to extend this time interval by slowing down the error detection clock frequency, but this slows down the error recovery time for the SEU event. For the minimum update interval for Stratix IV devices, refer to Table 11–6 on page 11–9. If these bits are not shifted out before the next error location is found, the previous error location and error message is overwritten by the new information. The CRC circuit continues to run, and if an error is detected, you must decide whether to complete a reconfiguration or to ignore the CRC error.

The error detection logic continues to calculate the `CRC_ERROR` and 16-bit signatures for the next frame of data regardless if any error has occurred in the current frame or not. You need to monitor these signals and take the appropriate actions if a soft error occurs.

The error detection circuitry in Stratix IV devices uses a 16-bit CRC-ANSI standard (16-bit polynomial) as the CRC generator.

The computed 16-bit CRC signature for each frame is stored in the registers within the core. The total storage register size is 16 (the number of bits per frame) × the number of frames.

The Stratix IV device error detection feature does not check memory blocks and I/O buffers. Thus, the `CRC_ERROR` signal might stay solid high or low depending on the error status of the previously checked CRAM frame. The I/O buffers are not verified during error detection because these bits use flipflops as storage elements that are more resistant to soft errors when compared with CRAM cells. The support parity bits of MLAB, M9K, and M144K are used to check the contents of the memory blocks for any errors. The M144K TriMatrix memory block has a built-in error correction code block that is able to check and correct the errors in the block.

For more information, refer to the *TriMatrix Embedded Memory Blocks in Stratix IV Devices* chapter.

A JTAG instruction, EDERROR_INJECT, is provided to test the capability of the error detection block. This instruction is able to change the content of the 21-bit JTAG fault injection register that is used for error injection in Stratix IV devices, enabling the testing of the error detection block.

You can only execute the EDERROR_INJECT JTAG instruction when the device is in user mode.

Table 11–1 lists the description of the EDERROR_INJECT JTAG instruction.

Table 11–1. EDERROR_INJECT JTAG Instruction

JTAG Instruction	Instruction Code	Description
EDERROR_INJECT	00 0001 0101	This instruction controls the 21-bit JTAG fault injection register, which is used for error injection.

You can create a Jam™ file (**.jam**) to automate the testing and verification process. This allows you to verify the CRC functionality in-system, on-the-fly, without having to reconfigure the device. You can then switch to the CRC circuit to check for real errors induced by an SEU.

You can introduce a single-error or double-errors adjacent to each other to the configuration memory. This provides an extra way to facilitate design verification and system fault tolerance characterization. Use the JTAG fault injection register with the EDERROR_INJECT instruction to flip the readback bits. The Stratix IV device is then forced into error test mode.

The content of the JTAG fault injection register is not loaded into the fault injection register during the processing of the last and first frame. It is only loaded at the end of this period.

You can only introduce error injection in the first data frame, but you can monitor the error information at any time. For more information about the JTAG fault injection register and fault injection register, refer to "Error Detection Registers" on page 11–6.

Table 11–2 lists how the fault injection register is implemented and describes error injection.

Table 11–2. Fault Injection Register

Bit	Bit[20..19]			Bit[18..8]	Bit[7..0]
Description	**Error Type**			**Byte Location of the Injected Error**	**Error Byte Value**
Content	Error Type *(1)*		Error injection type	Depicts the location of the injected error in the first data frame.	Depicts the location of the bit error and corresponds to the error injection type selection.
	Bit[20]	Bit[19]			
	0	1	Single-byte error injection		
	1	0	Double-adjacent byte error injection		
	0	0	No error injection		

Note to Table 11–2:

(1) Bit[20] and Bit[19] cannot both be set to 1 as this is not a valid selection. The error detection circuitry decodes this as no error injection.

After the test completes, Altera recommends that you reconfigure the device.

Automated Single-Event Upset Detection

Stratix IV devices offer on-chip circuitry for automated checking of SEU detection. Some applications that require the device to operate error-free in high-neutron flux environments require periodic checks to ensure continued data integrity. The error detection CRC feature ensures data reliability and is one of the best options for mitigating SEU.

You can implement the error detection CRC feature with existing circuitry in Stratix IV devices, eliminating the need for external logic. The CRC_ERROR pin reports a soft error when the configuration CRAM data is corrupted. You must decide whether to reconfigure the device or to ignore the error.

Error Detection Pin Description

Depending on the type of error detection feature you choose, you must use different error detection pins to monitor the data during user mode.

CRC_ERROR Pin

Table 11–3 describes the CRC_ERROR pin.

Table 11–3. CRC_ERROR Pin Description

Pin Name	Pin Type	Description
CRC_ERROR	I/O, output open-drain	Active-high signal indicates that the error detection circuit has detected errors in the configuration CRAM bits. This pin is optional and is used when the error detection CRC circuit is enabled. When the error detection CRC circuit is disabled, it is a user I/O pin. The CRC error output, when using the `stratixiv_crcblock` WYSIWYG function, is a dedicated path to the CRC_ERROR pin. To use the CRC_ERROR pin, you can either tie this pin to V_{CCPGM} through a 10k Ω resistor or, depending on the input voltage specification of the system receiving the signal, you can tie this pin to a different pull-up voltage.

The WYSIWYG function performs optimization on the Verilog Quartus Mapping (VQM) netlist within the Quartus II software.

For more information about the `stratixiv_crcblock` WYSIWYG function, refer to the *AN 539: Test Methodology of Error Detection and Recovery using CRC in Altera FPGA Devices*.

For more information about the CRC_ERROR pin for Stratix IV devices, refer to *Device Pin-Outs* on the Altera website.

Error Detection Block

You can enable the Stratix IV device error detection block in the Quartus II software (refer to "Software Support" on page 11–10). This block contains the logic necessary to calculate the 16-bit CRC signature for the configuration CRAM bits in the device.

The CRC circuit continues running even if an error occurs. When a soft error occurs, the device sets the `CRC_ERROR` pin high. Two types of CRC detection checks the configuration bits:

- CRAM error checking ability (16-bit CRC), which occurs during user mode to be used by the `CRC_ERROR` pin.
  - For each frame of data, the pre-calculated 16-bit CRC enters the CRC circuit at the end of the frame data and determines whether there is an error or not.
  - If an error occurs, the search engine starts to find the location of the error.
  - The error messages are shifted out through the JTAG instruction or core interface logics while the error detection block continues running.
  - The JTAG interface reads out the 16-bit CRC result for the first frame and also shifts the 16-bit CRC bits to the 16-bit CRC storage registers for test purposes.
  - Single error, double errors, or double-errors adjacent to each other are deliberately introduced to configuration memory for testing and design verification.
- 16-bit CRC that is embedded in every configuration data frame.
  - During configuration, after a frame of data is loaded into the Stratix IV device, the pre-computed CRC is shifted into the CRC circuitry.
  - At the same time, the CRC value for the data frame shifted-in is calculated. If the pre-computed CRC and calculated CRC values do not match, `nSTATUS` is set low. Every data frame has a 16-bit CRC; therefore, there are many 16-bit CRC values for the whole configuration bitstream. Every device has different lengths of configuration data frame.

The "Error Detection Block" section describes the 16-bit CRC only when the device is in user mode.

Error Detection Registers

There is one set of 16-bit registers in the error detection circuitry that stores the computed CRC signature. A non-zero value on the syndrome register causes the `CRC_ERROR` pin to be set high.

Figure 11–1 shows the error detection circuitry, syndrome registers, and error injection block.

Figure 11–1. Error Detection Block Diagram

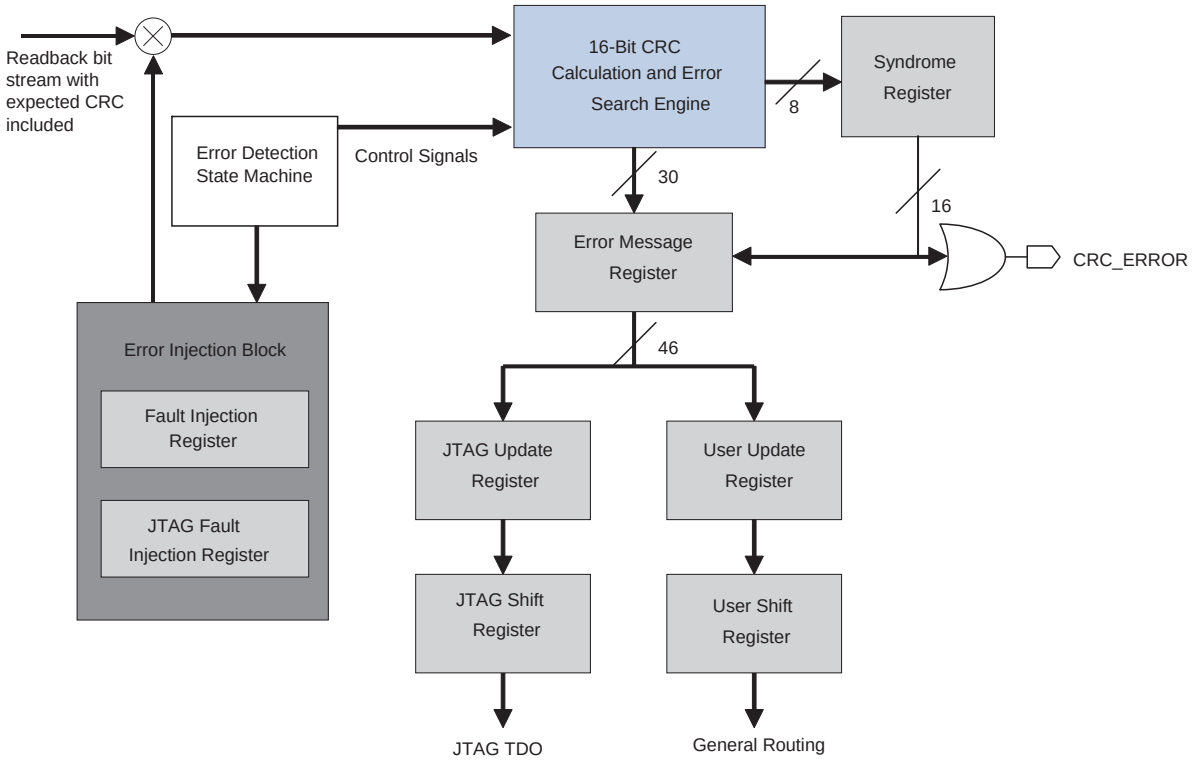


Table 11–4 defines the registers shown in Figure 11–1.

Table 11–4. Error Detection Registers (Part 1 of 2)

Register	Description
Syndrome Register	This register contains the CRC signature of the current frame through the error detection verification cycle. The `CRC_ERROR` signal is derived from the contents of this register.
Error Message Register	This 46-bit register contains information on the error type, location of the error, and the actual syndrome. The types of errors and location reported are single- and double-adjacent bit errors. The location bits for other types of errors are not identified by the error message register. The content of the register can be shifted out through the `SHIFT_EDERROR_REG` JTAG instruction or to the core through the core interface.
JTAG Update Register	This register is automatically updated with the contents of the error message register one cycle after the 46-bit register content is validated. It includes a clock enable that must be asserted prior to being sampled into the JTAG shift register. This requirement ensures that the JTAG update register is not being written into by the contents of the error message register at exactly the same time that the JTAG shift register is reading its contents.

Table 11–4. Error Detection Registers (Part 2 of 2)

Register	Description
User Update Register	This register is automatically updated with the contents of the Error Message Register, one cycle after the 46-bit register content is validated. It includes a clock enable that must be asserted prior to being sampled into the User Shift Register. This requirement ensures that the User Update Register is not being written into by the contents of the Error Message Register at exactly the same time that the User Shift Register is reading its contents.
JTAG Shift Register	This register is accessible by the JTAG interface and allows the contents of the JTAG Update Register to be sampled and read by the JTAG instruction `SHIFT_EDERROR_REG`.
User Shift Register	This register is accessible by the core logic and allows the contents of the User Update Register to be sampled and read by user logic.
JTAG Fault Injection Register	This 21-bit register is fully controlled by the JTAG instruction `EDERROR_INJECT`. This register holds the information of the error injection that you want in the bitstream.
Fault Injection Register	The content of the JTAG Fault Injection Register is loaded into this 21-bit register when it is being updated.

Error Detection Timing

When you enable the CRC feature through the Quartus II software, the device automatically activates the CRC process upon entering user mode, after configuration, and after initialization is complete.

If an error is detected within a frame, `CRC_ERROR` is driven high at the end of the error location search, after the error message register is updated. At the end of this cycle, the `CRC_ERROR` pin is pulled low for a minimum of 32 clock cycles. If the next frame contains an error, the `CRC_ERROR` is driven high again after the error message register is overwritten by the new value. You can start to unload the error message on each rising edge of the `CRC_ERROR` pin. The error detection runs until the device is reset.

The error detection circuitry runs off an internal configuration oscillator with a divisor that sets the maximum frequency. Table 11–5 lists the minimum and maximum error detection frequencies based on the best performance of the internal configuration oscillator.

Table 11–5. Minimum and Maximum Error Detection Frequencies

Device Type	Error Detection Frequency	Maximum Error Detection Frequency	Minimum Error Detection Frequency	Valid Divisors (n)
Stratix IV	100 MHz / 2^n	50 MHz	390 kHz	1, 2, 3, 4, 5, 6, 7, 8

You can set a lower clock frequency by specifying a division factor in the Quartus II software (refer to "Software Support" on page 11–10). The divisor is a power of two, in which n is between 1 and 8. The divisor ranges from 2 through 256. Refer to Equation 11–1.

Equation 11–1.

$$\text{error detection frequency} = \frac{100\text{MHz}}{2^n}$$

The error detection frequency reflects the frequency of the error detection process for a frame because the CRC calculation in the Stratix IV device is done on a per-frame basis.

You must monitor the error message to avoid missing information in the error message register. The error message register is updated whenever an error or errors occur. The minimum interval time between each update for the error message register depends on the device and the error detection clock frequency.

Table 11–6 lists the estimated minimum interval time between each update for the error message register for Stratix IV devices.

Table 11–6. Minimum Update Interval for Error Message Register *(Note 1)*

Device	Timing Interval (µs)
EP4SGX70	15
EP4SGX110	15
EP4SGX180	21
EP4SGX230	21
EP4SGX290	23
EP4SGX360	23
EP4SGX530	28
EP4SE230	21
EP4SE360	23
EP4SE530	28
EP4SE820	35
EP4S40G2	21
EP4S40G5	28
EP4S100G2	21
EP4S100G3	28
EP4S100G4	28
EP4S100G5	28

Note to Table 11–6:

(1) These timing numbers are preliminary.

CRC calculation time for the error detection circuitry to check from the first until the last frame depends on the device and the error detection clock frequency.

Table 11–7 lists the estimated time for each CRC calculation with minimum and maximum clock frequencies for Stratix IV devices. The minimum CRC calculation time is calculated by using the maximum error detection frequency with a divisor factor of one, while the maximum CRC calculation time is calculated by using the minimum error detection frequency with a divisor factor of eight.

Table 11–7. CRC Calculation Time *(Note 1)*

Device	Minimum Time (ms)	Maximum Time (s)
EP4SGX70	111	30.90
EP4SGX110	111	30.90
EP4SGX180	225	62.44
EP4SGX230	225	62.44
EP4SGX290	296	82.05
EP4SGX360	296	82.05
EP4SGX530	398	110.38
EP4SE230	225	62.44
EP4SE360	296	82.05
EP4SE530	398	110.38
EP4SE820	577	160.00
EP4S40G2	225	62.44
EP4S40G5	398	110.38
EP4S100G2	225	62.44
EP4S100G3	398	110.38
EP4S100G4	398	110.38
EP4S100G5	398	110.38

Note to Table 11–7:

(1) These timing numbers are preliminary.

Software Support

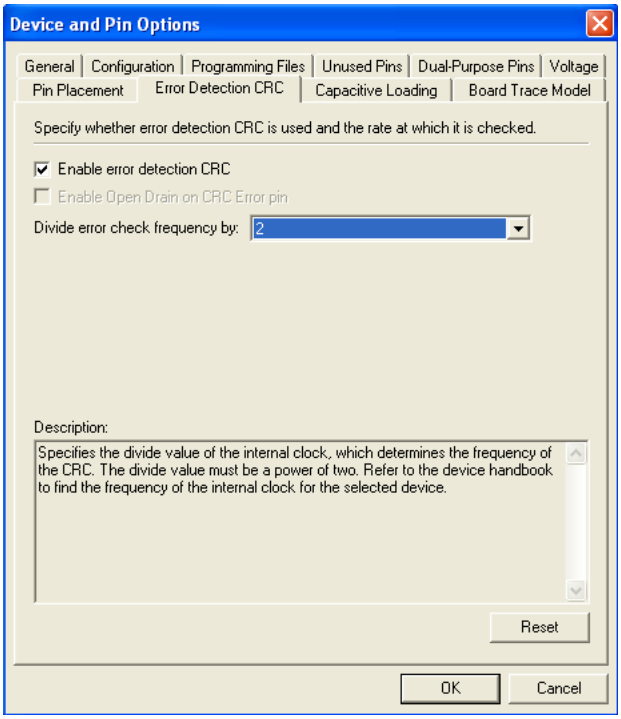
The Quartus II software version 8.0 and onwards supports the error detection CRC feature for Stratix IV devices. Enabling this feature generates the `CRC_ERROR` output to the optional dual purpose `CRC_ERROR` pin.

The error detection CRC feature is controlled by the **Device and Pin Options** dialog box in the Quartus II software.

To enable the error detection feature using CRC, perform the following steps:

1. Open the Quartus II software and load a project using a Stratix IV device.
2. On the Assignments menu, click **Settings**. The **Settings** dialog box is shown.
3. In the **Category** list, select **Device**. The **Device** page is shown.
4. Click **Device and Pin Options**. The **Device and Pin Options** dialog box is shown (refer to Figure 11–2).
5. In the **Device and Pin Options** dialog box, click the **Error Detection CRC** tab.
6. Turn on **Enable error detection CRC** (Figure 11–2).

Figure 11–2. Enabling the Error Detection CRC Feature in the Quartus II Software



7. In the **Divide error check frequency by** pull-down list, enter a valid divisor as listed in Table 11–5 on page 11–8.

☞ The divide value divides the frequency of the configuration oscillator output clock that clocks the CRC circuitry.

8. Click **OK**.

Recovering From CRC Errors

The system that the Stratix IV device resides in must control the device reconfiguration. After detecting an error on the `CRC_ERROR` pin, strobing the `nCONFIG` signal low directs the system to perform the reconfiguration at a time when it is safe for the system to reconfigure the device.

When the data bit is rewritten with the correct value by reconfiguring the device, the device functions correctly.

While soft errors are uncommon in Altera devices, certain high-reliability applications require a design to account for these errors.

Document Revision History

Table 11–8 lists the revision history for this chapter.

Table 11–8. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
November 2009 v3.0	■ Updated Table 11–3, Table 11–5, Table 11–6, and Table 11–7. ■ Updated the "CRC_ERROR Pin" section. ■ Minor text edits.	—
June 2009 v2.3	■ Added an introductory paragraph to increase search ability. ■ Removed the Conclusion section. ■ Minor text edits.	—
April 2009 v2.2	■ Updated Table 11–6 and Table 11–7.	—
March 2009 v2.1	■ Updated "Error Detection Timing" section. ■ Updated Table 11–6. ■ Added Table 11–7. ■ Removed "Critical Error Detection", "Critical Error Pin", and "Referenced Documents" sections.	—
November 2008 v2.0	Minor text edits.	—
May 2008 | v1.0	Initial Release.	—

12. JTAG Boundary-Scan Testing in Stratix IV Devices

SIV51012-3.0

The IEEE Std. 1149.1 boundary-scan test (BST) circuitry available in Stratix® IV devices provides a cost-effective and efficient way to test systems that contain devices with tight lead spacing. Circuit boards with Altera and other IEEE Std. 1149.1-compliant devices can use `EXTEST`, `SAMPLE/PRELOAD`, and `BYPASS` modes to create serial patterns that internally test the pin connections between devices and check device operation.

This chapter describes how to use the IEEE Std. 1149.1 BST circuitry in Stratix IV devices. The features are similar to Stratix III devices, unless stated otherwise in this document.

This chapter contains the following sections:

- "BST Architecture" on page 12–1
- "BST Operation Control" on page 12–1
- "I/O Voltage Support in a JTAG Chain" on page 12–3
- "BST Circuitry" on page 12–3
- "BSDL Support" on page 12–4

BST Architecture

A device operating in IEEE Std. 1149.1 BST mode uses four required pins, `TDI`, `TDO`, `TMS`, `TCK`, and one optional pin, `TRST`. The `TCK` pin has an internal weak pull-down resistor, while the `TDI`, `TMS`, and `TRST` pins have internal weak pull-up resistors. The `TDO` output pin and all the JTAG input pins are powered by the 2.5-V/3.0-V V_{CCPD} supply of I/O bank 1A. All user I/O pins are tri-stated during JTAG configuration.

For more information about the description and functionality of all JTAG pins, registers used by the IEEE Std. 1149.1 BST circuitry, and the test access port (TAP) controller, refer to the *IEEE 1149.1 (JTAG) Boundary-Scan Testing in Stratix III Devices* chapter in volume 1 of the *Stratix III Device Handbook*.

BST Operation Control

Table 12–1 lists the boundary-scan register length for Stratix IV devices.

Table 12–1. Stratix IV Devices Boundary-Scan Register Length (Part 1 of 2)

Device	Boundary-Scan Register Length
EP4SGX70	1506
EP4SGX110	1506
EP4SGX180	2274
EP4SGX230	2274
EP4SGX290 *(1)*	2682
EP4SGX360 *(1)*	2682

Table 12–1. Stratix IV Devices Boundary-Scan Register Length (Part 2 of 2)

Device	Boundary-Scan Register Length
EP4SGX530	2970
EP4SE230	2274
EP4SE360	2682
EP4SE530	2970
EP4SE820	3402
EP4S40G2	2274
EP4S40G5	2970
EP4S100G2	2274
EP4S100G3	2970
EP4S100G4	2970
EP4S100G5	2970

Note to Table 12–1:

(1) For the F1932 package of EP4SGX290 and EP4SGX360 devices, the boundary-scan register length is 2970.

Table 12–2 lists the `IDCODE` JTAG instruction information for Stratix IV devices.

Table 12–2. Stratix IV Devices IDCODE JTAG Instruction Information (Part 1 of 2)

Device	IDCODE (32 Bits) *(1)*			
	Version (4 Bits)	Part Number (16 Bits)	Manufacturer Identity (11 Bits)	LSB (1 Bit) *(2)*
EP4SGX70	0000	0010 0100 0010 0000	000 0110 1110	1
EP4SGX110	0000	0010 0100 0000 0000	000 0110 1110	1
EP4SGX180	0000	0010 0100 0010 0001	000 0110 1110	1
EP4SGX230	0000	0010 0100 0000 1001	000 0110 1110	1
EP4SGX290 *(3)*	0000	0010 0100 0010 0010	000 0110 1110	1
EP4SGX290 *(4)*	0000	0010 0100 0100 0011	000 0110 1110	1
EP4SGX360 *(3)*	0000	0010 0100 0000 0010	000 0110 1110	1
EP4SGX360 *(4)*	0000	0010 0100 1000 0011	000 0110 1110	1
EP4SGX530	0000	0010 0100 0000 0011	000 0110 1110	1
EP4SE230	0000	0010 0100 0001 0001	000 0110 1110	1
EP4SE360	0000	0010 0100 0001 0010	000 0110 1110	1
EP4SE530	0000	0010 0100 0001 0011	000 0110 1110	1
EP4SE820	0000	0010 0100 0000 0100	000 0110 1110	1
EP4S40G2 *(5)*	0000	0010 0100 0100 0001	000 0110 1110	1
EP4S40G5 *(6)*	0000	0010 0100 0010 0011	000 0110 1110	1
EP4S100G2 *(5)*	0000	0010 0100 0100 0001	000 0110 1110	1
EP4S100G3	0000	0010 0100 1010 0011	000 0110 1110	1
EP4S100G4	0000	0010 0100 0110 0011	000 0110 1110	1

Table 12–2. Stratix IV Devices IDCODE JTAG Instruction Information (Part 2 of 2)

Device	IDCODE (32 Bits) *(1)*			
	Version (4 Bits)	Part Number (16 Bits)	Manufacturer Identity (11 Bits)	LSB (1 Bit) *(2)*
EP4S100G5 *(6)*	0000	0010 0100 0010 0011	000 0110 1110	1

Notes to Table 12–2:

(1) The MSB is on the left.
(2) The LSB of the `IDCODE` JTAG instruction is always 1.
(3) The `IDCODE` JTAG instruction is applicable for all packages except F1932.
(4) The `IDCODE` JTAG instruction is applicable for package F1932 only.
(5) For the ES1 device, the `IDCODE` JTAG instruction is the same as the `IDCODE` JTAG instruction of EP4SGX230.
(6) For the ES1 device, the `IDCODE` JTAG instruction is the same as the `IDCODE` JTAG instruction of EP4SGX530.

To correctly read the `IDCODE` JTAG instruction, you must issue the `IDCODE` JTAG instruction after initialization, which is signaled by `nSTATUS` going high.

For more information about the following topics, refer to the *IEEE 1149.1 (JTAG) Boundary-Scan Testing in Stratix III Devices* chapter in volume 1 of the *Stratix III Device Handbook*:

- JTAG instruction codes with descriptions
- TAP controller state-machine
- Timing requirements for IEEE Std. 1149.1 signals
- Instruction mode
- Mandatory JTAG instructions (`SAMPLE/PRELOAD`, `EXTEST`, and `BYPASS`)
- Optional JTAG instructions (`IDCODE`, `USERCODE`, `CLAMP`, and `HIGHZ`)

I/O Voltage Support in a JTAG Chain

The JTAG chain supports several devices. However, you must use caution if the chain contains devices that have different V_{CCIO} levels.

For more information, refer to the *IEEE 1149.1 (JTAG) Boundary-Scan Testing in Stratix III Devices* chapter in volume 1 of the *Stratix III Device Handbook*.

BST Circuitry

The IEEE Std. 1149.1 BST circuitry is enabled upon device power-up. You can perform BST on Stratix IV devices before, during, and after configuration. Stratix IV devices support `BYPASS`, `IDCODE`, and `SAMPLE` JTAG instructions during configuration without interrupting configuration. To send all other JTAG instructions, you must interrupt configuration using the `CONFIG_IO` JTAG instruction.

For more information, refer to *AN 39: IEEE Std. 1149.1 (JTAG) Boundary-Scan Testing in Altera Devices*.

For more information about using the `CONFIG_IO` JTAG instruction for dynamic I/O buffer configuration, considerations when performing BST for configured devices, and JTAG pin connections to mask-out the BST circuitry, refer to the *IEEE 1149.1 (JTAG) Boundary-Scan Testing in Stratix III Devices* chapter in volume 1 of the *Stratix III Device Handbook*.

For more information about using the IEEE Std.1149.1 circuitry for device configuration, refer to the *Configuration, Design Security, Remote System Upgrades* chapter.

If you must perform BST for configured devices, you must use the Quartus II software version 8.1 and onwards to generate the design-specific boundary-scan description language (BSDL) files. For the procedure to generate post-configured BSDL files using the Quartus II software, refer to the *BSDL Files Generation in Quartus II* on the Altera website.

BSDL Support

BSDL, a subset of VHDL, provides a syntax that allows you to describe the features of an IEEE Std. 1149.1 BST-capable device that can be tested.

For more information about BSDL files for IEEE Std. 1149.1-compliant Stratix IV devices, refer to the *Stratix IV BSDL Files* on the Altera website.

BSDL files for IEEE std. 1149.1-compliant Stratix IV devices can also be generated using the Quartus II software version 8.1 and onwards. For more information about the procedure to generate BSDL files using the Quartus II software, refer to the *BSDL Files Generation in Quartus II* on the Altera website.

Document Revision History

Table 12–3 shows the revision history for this chapter.

Table 12–3. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
November 2009, v3.0	■ Updated Table 12–1 and Table 12–2. ■ Minor text edits.	—
June 2009, v2.3	■ Added an introductory paragraph to increase search ability. ■ Removed the Conclusion section. ■ Minor text edits.	—
April 2009 v2.2	■ Updated Table 12–1.	—
March 2009 v2.1	■ Updated Table 12–1 and Table 12–2. ■ Removed “Referenced Documents” section.	—
November 2008 v2.0	Minor text edits.	—
May 2008 v1.0	Initial Release.	—

13. Power Management in Stratix IV Devices

SIV51013-3.0

This chapter describes power management in Stratix® IV devices. Stratix IV devices offer programmable power technology options for low-power operation. You can use these features, along with speed grade choices, in different permutations to give the best power and performance combination. For thermal management, use the Stratix IV internal temperature sensing device (TSD) with built-in analog-to-digital converter (ADC) circuitry or external TSD with an external temperature sensor to easily incorporate this feature in your designs. Being able to monitor the junction temperature of the device at any time also allows you the ability to control air flow to the device and save power for the whole system.

Overview

This chapter describes how Stratix IV FPGAs deliver a breakthrough level of system bandwidth and power efficiency for high-end applications, allowing you to innovate without compromise. Advanced power management techniques are used in Stratix IV devices to enable both density and performance increases, while simultaneously reducing power dissipation.

The total power of an FPGA includes static power and dynamic power. Static power is the power consumed by the FPGA when it is configured but no clocks are operating. Dynamic power is comprised of switching power when the device is configured and running. Dynamic power is calculated using the equation shown in Equation 13–1.

Equation 13–1. Dynamic Power Equation

$$P = \frac{1}{2} C V^2 \times \text{frequency}$$

Note to Equation 13–1:

(1) P=power; C=load capacitance; V=supply voltage level.

Equation 13–1 shows that frequency is design-dependent. However, voltage can be varied to lower dynamic power consumption by the square value of the voltage difference. Stratix IV devices minimize static and dynamic power with advanced process optimizations and programmable power technology. These technologies enable Stratix IV designs to optimally meet design-specific performance requirements with the lowest possible power.

The Quartus® II software optimizes all designs with Stratix IV power technology to ensure performance is met at the lowest power consumption. This automatic process allows you to concentrate on the functionality of the design instead of the power consumption of the design.

Power consumption also affects thermal management. Stratix IV devices offer a TSD feature, which can self-monitor the device junction temperature and can be used with external circuitry for activities such as controlling air flow to the Stratix IV FPGA.

This chapter contains the following sections:

- "Stratix IV Power Technology"
- "Stratix IV External Power Supply Requirements"
- "Temperature Sensing Diode"

Stratix IV Power Technology

The following sections describe Stratix IV programmable power technology.

Programmable Power Technology

Stratix IV devices offer the ability to configure portions of the core, called tiles, for high-speed or low-power mode of operation performed by the Quartus II software without user intervention. Setting a tile to high-speed or low-power mode is accomplished with on-chip circuitry and does not require extra power supplies brought into the Stratix IV device. In a design compilation, the Quartus II software determines whether a tile must be in high-speed or low-power mode based on the timing constraints of the design.

For more information about how the Quartus II software uses programmable power technology when compiling a design, refer to *AN 514: Power Optimization in Stratix IV Devices*.

A Stratix IV tile can consist of the following:

- Memory logic array block (MLAB)/logic array block (LAB) pairs with routing to the pair
- MLAB/LAB pairs with routing to the pair and to adjacent digital signal processing (DSP)/memory block routing
- TriMatrix memory blocks
- DSP blocks

All blocks and routing associated with the tile share the same setting of either high speed or low power. By default, tiles that include DSP blocks or memory blocks are set to high-speed mode for optimum performance. Unused DSP blocks and memory blocks are set to low-power mode to minimize static power. Clock networks do not support programmable power technology.

With programmable power technology, faster speed grade FPGAs may require less power because there are fewer high-speed MLAB and LAB pairs, when compared with slower speed grade FPGAs. The slower speed grade device may need to use more high-speed MLAB and LAB pairs to meet performance requirements, while the faster speed grade device can meet performance requirements with MLAB and LAB pairs in low-power mode.

The Quartus II software can set unused device resources in the design to low-power mode to reduce static and dynamic power. The Quartus II software can set the following resources to low power when they are not used in the design:

- LABs and MLABs
- TriMatrix memory blocks
- DSP blocks

If the PLL is instantiated in the design, asserting the `areset` pin high keeps the PLL in low-power mode.

Table 13–1 lists the available Stratix IV programmable power capabilities. Speed grade considerations can add to the permutations to give you flexibility in designing your system.

Table 13–1. Stratix IV Programmable Power Capabilities

Feature	Programmable Power Technology
LAB	Yes
Routing	Yes
Memory Blocks	Fixed setting *(1)*
DSP Blocks	Fixed setting *(1)*
Global Clock Networks	No

Note to Table 13–1:

(1) Tiles with DSP blocks and memory blocks that are used in the design are always set to high-speed mode. By default, unused DSP blocks and memory blocks are set to low-power mode.

Stratix IV External Power Supply Requirements

This section describes the different external power supplies needed to power Stratix IV devices. You can supply some of the power supply pins with the same external power supply, provided they need the same voltage level.

For power supply pin connection guidelines and power regulator sharing, refer to the *Stratix IV GX Device Family Pin Connection Guidelines*.

For each Altera recommended power supply's operating conditions, refer to the *DC and Switching Characteristics* chapter.

Temperature Sensing Diode

The Stratix IV TSD uses the characteristics of a PN junction diode to determine die temperature. Knowing the junction temperature is crucial for thermal management. Historically, junction temperature is calculated using ambient or case temperature, junction-to-ambient (ja) or junction to-case (jc) thermal resistance, and device power consumption. Stratix IV devices can either monitor its die temperature with the internal TSD with built-in ADC circuitry or the external TSD with an external temperature sensor. This allows you to control the air flow to the device.

You can use the Stratix IV internal TSD in two different modes of operations—power-up mode and user mode. For power-up mode, the internal TSD reads the die's temperature during configuration if the ALTTEMP_SENSE megafunction is enabled in your design. The ALTTEMP_SENSE megafunction allows temperature sensing during device user mode by asserting the `clken` signal to the internal TSD circuitry. To reduce device static power, you can disable the internal TSD with built-in ADC circuitry when not in use.

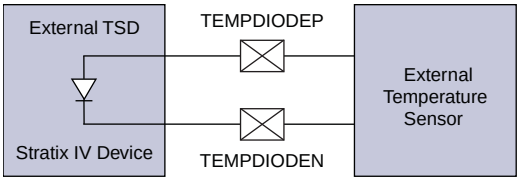
For more information about using the ALTTEMP_SENSE megafunction, refer to the *ALTTEMP_SENSE Megafunction User Guide*.

The external temperature sensor steers bias current through the Stratix IV external TSD, measuring forward voltage and converting this reading to temperature in the form of an 8-bit signed number (7 bits plus sign). The 8-bit output represents the junction temperature of the Stratix IV device and can be used for intelligent power management.

External Pin Connections

Stratix IV external TSD requires two pins for voltage reference. Figure 13–1 shows how you can connect the external TSD with an external temperature sensor device. As an example, external temperature sensing devices, such as MAX1619, MAX1617A, MAX6627, and ADT 7411, can be connected to the two external TSD pins for temperature reading.

Figure 13–1. Stratix IV TSD External Pin Connections



For more information about the external TSD specification, refer to the *DC and Switching Characteristics* chapter.

The TSD is a very sensitive circuit that can be influenced by noise coupled from other traces on the board and possibly within the device package itself, depending on your device usage. The interfacing device registers' temperature is based on millivolts (mV) of difference, as seen at the external TSD pins. Switching the I/O near the TSD pins can affect the temperature reading. Altera recommends taking temperature readings during periods of inactivity in the device or use the internal TSD with built-in ADC circuitry.

The following are board connection guidelines for the TSD external pin connections:

- The maximum trace lengths for the TEMPDIODEP/TEMPDIODEN traces must be less than eight inches.
- Route both traces in parallel and place them close to each other with grounded guard tracks on each side.
- Altera recommends 10-mils width and space for both traces.

- Route traces through a minimum number of vias and crossunders to minimize the thermocouple effects.
- Ensure that the number of vias are the same on both traces.
- Ensure both traces are approximately the same length.
- Avoid coupling with toggling signals (for example, clocks and I/O) by having the GND plane between the diode traces and the high frequency signals.
- For high-frequency noise filtering, place an external capacitor (close to the external chip) between the TEMPDIODEP/TEMPDIODEN trace.
- For Maxim devices, use an external capacitor between 2200 pF to 3300 pF.
- Place a 0.1 uF bypass capacitor close to the external device.

For device specification and connection guidelines, refer to the external temperature sensor device data sheet from the device manufacturer.

Document Revision History

Table 13–2 shows the revision history for this chapter.

Table 13–2. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
November 2009 v3.0	■ Updated the "Temperature Sensing Diode" and "External Pin Connections" sections. ■ Updated Equation 13–1. ■ Removed Table 13-2: Stratix IV External Power Supply Pins. ■ Minor text edits.	—
June 2009 v2.2	■ Updated the "External Pin Connections" section. ■ Added an introductory paragraph to increase search ability. ■ Removed the Conclusion section.	—
March 2009 v2.1	■ Updated "Temperature Sensing Diode" and "External Pin Connections" sections. ■ Updated Figure 13–1. ■ Removed "Referenced Documents" section.	—
November 2008 v2.0	Minor text edits.	—
May 2008 v1.0	Initial Release.	—

Stratix IV Device Handbook

Volume 2

101 Innovation Drive
San Jose, CA 95134
www.altera.com

SIV5V2-4.0

Copyright © 2009 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

I.S. EN ISO 9001

Contents

Chapter Revision Dates ix

Additional Information
About this Handbook . . . . . Info-xi
How to Contact Altera . . . . . Info-xi
Typographic Conventions . . . . . Info-xi

Section I. Transceiver Architecture
Revision History I-1

Chapter 1. Stratix IV Transceiver Architecture
Overview 1-1
Transceiver Channel Locations 1-4
Stratix IV GX Device Offerings 1-4
Stratix IV GT Device Offerings 1-7
Transceiver Block Architecture 1-11
Transceiver Channel Architecture 1-12
Transmitter Channel Datapath 1-14
Receiver Channel Datapath 1-36
CMU Channel Architecture 1-97
Configuring CMU Channels for Clock Generation 1-98
Configuring CMU Channels as Transceiver Channels 1-103
Other CMU Channel Features 1-106
Dynamic Reconfiguration of the CMU Channel Analog Controls 1-107
Functional Modes 1-107
Basic Functional Mode 1-108
Deterministic Latency Mode 1-119
PCI Express (PIPE) Mode 1-124
XAUI Mode 1-150
GIGE Mode 1-161
SONET/SDH Mode 1-169
SDI Mode 1-175
(OIF) CEI PHY Interface Mode 1-179
Serial RapidIO Mode 1-180
Basic (PMA Direct) Functional Mode 1-185
Loopback Modes 1-188
Serial Loopback 1-188
Parallel Loopback 1-189
Reverse Serial Loopback 1-191
Reverse Serial Pre-CDR Loopback 1-191
PCI Express (PIPE) Reverse Parallel Loopback 1-192
Auxiliary Transmit (ATX) PLL Block 1-193
6G ATX PLL Block 1-193
10G ATX PLL Block 1-194
Input Reference Clocks for the ATX PLL Block 1-196
Architecture of the ATX PLL Block 1-197
ATX Clock Divider 1-198
The Differences Between 10G ATX PLL, 6G ATX PLL, and CMU PLL 1-198

Calibration Blocks 1-199
Calibration Block Location 1-199
Calibration 1-203
Input Signals to the Calibration Block 1-203
Built-In Self Test Modes 1-204
BIST Mode Pattern Generators and Verifiers 1-204
PRBS in Single-Width Mode 1-206
PRBS in Double-Width Mode 1-206
Transceiver Port Lists 1-207
Reference Information 1-223
Document Revision History 1-225

Chapter 2. Stratix IV Transceiver Clocking

Glossary of Terms 2-2
Input Reference Clocking 2-2
Input Reference Clock Source 2-3
refclk0 and refclk1 Pins 2-7
Inter-Transceiver Block (ITB) Clock Lines 2-8
Dedicated CLK Input Pins on the FPGA Global Clock Network 2-8
Clock Output from Left and Right PLLs in the FPGA Fabric 2-9
FPGA Fabric PLLs-Transceiver PLLs Cascading 2-9
Example 1: Channel Configuration with 4-Gbps Data Rate 2-9
Dedicated Left and Right PLL Cascade Network 2-10
FPGA Fabric PLLs-Transceiver PLLs Cascading in the 780-Pin Package 2-11
FPGA Fabric PLLs-Transceiver PLLs Cascading in the 1152-Pin Package 2-11
FPGA Fabric PLLs-Transceiver PLLs Cascading in the 1517-Pin Package 2-12
FPGA Fabric PLLs-Transceiver PLLs Cascading in the 1932-Pin Package 2-13
FPGA Fabric PLLs-Transceiver PLLs Cascading Rules 2-14
Example 2: Design Target—EP4SGX530NF45 Device 2-14
Left and Right, Left, or Right PLL in VCO Bypass Mode 2-17
Transceiver Channel Datapath Clocking 2-20
Transmitter Channel Datapath Clocking 2-20
Transmitter Channel-to-Channel Skew Optimization for Modes Other than Basic (PMA Direct) Mode 2-20
Transmitter Channel Datapath Clocking Resources 2-21
Transmitter Channel Clocking Configurations 2-22
Non-Bonded Channel Configurations 2-23
Bonded Channel Configurations 2-26
Non-Bonded Basic (PMA Direct) Mode Channel Configurations 2-32
Bonded Basic (PMA Direct) ×N Mode Channel Configurations 2-34
Receiver Channel Datapath Clocking 2-37
Non-Bonded Channel Configurations 2-37
Bonded Channel Configurations 2-41
Basic (PMA Direct) Mode Channel Configurations 2-47
FPGA Fabric-Transceiver Interface Clocking 2-49
FPGA Fabric-Transmitter Interface Clocking 2-50
Quartus II-Selected Transmitter Phase Compensation FIFO Write Clock 2-50
User-Selected Transmitter Phase Compensation FIFO Write Clock 2-56
FPGA Fabric-Receiver Interface Clocking 2-59
Quartus II Software-Selected Receiver Phase Compensation FIFO Read Clock 2-60
User-Selected Receiver Phase Compensation FIFO Read Clock 2-67
Using the CMU/ATX PLL for Clocking User Logic in the FPGA Fabric 2-70

Configuration Examples 2-72
Configuration Example 1: Configuring 24 Channels in Basic (PMA Direct) ×N Mode in the EP4S100G5F45 Device 2-72
Configuration Example 2: Configuring Sixteen Identical Channels Across Four Transceiver Blocks . 2-74
Configuration Example 3: Configuring Sixteen Channels Across Four Transceiver Blocks 2-75
Configuration Example 4: Configuring Left and Right, Left, or Right PLL in VCO Bypass Mode 2-77
Document Revision History 2-80

Chapter 3. Configuring Multiple Protocols and Data Rates in a Transceiver Block

Overview 3-1
Glossary of Terms 3-2
Creating Transceiver Channel Instances 3-3
General Requirements to Combine Channels 3-3
Transmitter Buffer Voltage (V_{CCH}) 3-3
Transceiver Analog Power ($V_{CCA_L/R}$) 3-3
Control Signals 3-4
gxb_powerdown Port 3-4
reconfig_fromgxb and reconfig_togxb Ports 3-4
Calibration Clock and Power Down 3-5
Sharing CMU PLLs 3-5
Multiple Channels Sharing a CMU PLL 3-5
Example 1 3-6
Example 2 3-8
Sharing ATX PLLs 3-9
Combining Receiver Only Channels 3-9
Combining Transmitter Channel and Receiver Channel Instances 3-10
Multiple Transmitter Channel and Receiver Channel Instances 3-10
Example 3 3-10
Combining Transceiver Instances in Multiple Transceiver Blocks 3-12
Example 4 3-13
Combining Transceiver Instances Using PLL Cascade Clocks 3-15
Combining Channels Configured in Protocol Functional Modes 3-16
Combining Channels in Bonded Functional Modes 3-16
Bonded ×4 Functional Mode 3-16
Bonded x8 Functional Mode 3-19
Combining Channels Configured in Deterministic Latency Mode 3-23
Combining Channels Using the PCI Express hard IP Block with Other Channels 3-23
Combining Transceiver Channels with Basic (PMA Direct) Configuration 3-24
Combining Multiple Channels Configured in Basic (PMA Direct) ×1 Configurations 3-25
Multiple Basic (PMA Direct) ×1 Configuration Instances with One Channel per Instance 3-25
One Instance in Basic (PMA Direct) ×1 Configuration with Multiple Transceiver Channels . . 3-25
Combining Multiple Instances of TX Only and RX Only Configurations in Basic (PMA Direct) ×1 Mode 3-28
Combining Channels Configured in Basic (PMA Direct) ×1 with Non-Basic (PMA Direct) Modes 3-28
Basic (PMA Direct) ×N Configuration 3-32
Channel Placement in a Basic (PMA Direct) ×N Mode Instance 3-32
Examples of Combining Multiple Instances of Basic (PMA Direct) ×N Modes 3-34

Combination Requirements When Channel Reconfiguration is Enabled 3-41
Combination Requirements When the Use Alternate CMU PLL Option is Selected 3-41
Example 12 3-42
Key Observations 3-43
Combination Requirements When Multiple TX PLLs are Used 3-43
Example 13 3-44
Combining Transceiver Channels When the Adaptive Equalization (AEQ) is Enabled 3-46
Example 14 3-47
Combination Requirements for Stratix IV GT Devices 3-48
Placement Rules for Transceiver Channels at 9.9 Gbps to 10.3125 Gbps 3-48
Summary 3-49
Document Revision History 3-49

Chapter 4. Reset Control and Power Down

User Reset and Power-Down Signals 4-1
Blocks Affected by the Reset and Power-Down Signals 4-3
Transceiver Reset Sequences 4-4
All Supported Functional Modes Except the PCI Express (PIPE) Functional Mode 4-5
Bonded Channel Configuration 4-6
Non-Bonded Channel Configuration 4-14
PCI Express (PIPE) Functional Mode 4-20
PCI Express (PIPE) Reset Sequence 4-21
PCI Express (PIPE) Initialization/Compliance Phase 4-21
PCI Express (PIPE) Normal Phase 4-22
PMA Direct Drive Mode Reset Sequences 4-23
Basic (PMA Direct) Drive ×N Mode 4-24
Transmitter Only Channel with No PLL_L/R 4-24
Transmitter Only Channel with a PLL_L/R 4-25
Basic (PMA Direct) Drive x1 Mode 4-30
Receiver and Transmitter Channel Set-Up—Receiver CDR in Automatic Lock Mode 4-31
Receiver and Transmitter Channel Set-up—Receiver CDR in Manual Lock Mode 4-33
Dynamic Reconfiguration Reset Sequences 4-34
Reset Sequence when Using Dynamic Reconfiguration with the ‘data rate division in TX’ Option . . . 4-34
Reset Sequence when Using Dynamic Reconfiguration with the ‘Channel and TX PLL select/reconfig’ Option 4-36
Power Down 4-37
Simulation Requirements 4-38
Reference Information 4-39
Document Revision History 4-40

Chapter 5. Stratix IV Dynamic Reconfiguration

Glossary of Terms 5-1
Dynamic Reconfiguration Controller Architecture 5-3
Quartus II MegaWizard Plug-In Manager Interfaces to Support Dynamic Reconfiguration 5-4
ALTGX MegaWizard Plug-In Manager 5-4
The reconfig_clk Clock Requirements for the ALTGX Instance 5-4
ALTGX_RECONFIG MegaWizard Plug-In Manager 5-5
The reconfig_clk Clock Requirements for the ALTGX_RECONFIG Instance 5-5
Interfacing ALTGX and ALTGX_RECONFIG Instances 5-5
Logical Channel Addressing 5-5
Total Number of Channels Option in the ALTGX_RECONFIG Instance 5-10
Connecting the ALTGX and ALTGX_RECONFIG Instances 5-11

Dynamic Reconfiguration Modes Implementation 5-12
PMA Controls Reconfiguration Mode Details 5-12
Dynamically Reconfiguring PMA Controls 5-13
Transceiver Channel Reconfiguration Mode Details 5-19
Memory Initialization File (.mif) 5-19
Channel and CMU PLL Reconfiguration Mode Details 5-24
Channel Reconfiguration with Transmitter PLL Select Mode Details 5-48
CMU PLL Reconfiguration Mode Details 5-54
Central Control Unit Reconfiguration Mode Details 5-57
Special Guidelines 5-57
Data Rate Division in Transmitter Mode Details 5-63
Offset Cancellation Feature 5-66
Operation 5-67
ALTGX_RECONFIG Instance Signals Transition during Offset Cancellation 5-68
EyeQ 5-69
Enabling the EyeQ Control Logic and the EyeQ Hardware 5-69
Connections Between the ALTGX and ALTGX_RECONFIG Instances 5-70
Controlling the EyeQ Hardware 5-71
Adaptive Equalization (AEQ) 5-74
Adaptive Equalization Limitations 5-74
Enabling the AEQ Control Logic and AEQ Hardware 5-75
Connections Between the ALTGX and ALTGX_RECONFIG Instances 5-75
Controlling the AEQ Hardware 5-77
Dynamic Reconfiguration Controller Port List 5-78
Error Indication During Dynamic Reconfiguration 5-90
Dynamic Reconfiguration Duration 5-92
PMA Controls Reconfiguration Duration 5-92
Offset Cancellation Duration 5-93
Dynamic Reconfiguration Duration for Channel and Transmitter PLL Select/Reconfig Modes . . . 5-94
Dynamic Reconfiguration (ALTGX_RECONFIG Instance) Resource Utilization 5-94
Functional Simulation of the Dynamic Reconfiguration Process 5-95
Dynamic Reconfiguration Examples 5-96
Example 1 5-96
Section 1 5-97
Section 2 5-99
Example 2 5-100
Section 1 5-100
Section 2 5-102
Document Revision History 5-102

Chapter Revision Dates

The chapters in this book, *Stratix IV Device Handbook Volume 2*, were revised on the following dates. Where chapters or groups of chapters are available separately, part numbers are listed.

Chapter 1 Stratix IV Transceiver Architecture
Revised: *November 2009*
Part Number: *SIV52001-4.0*

Chapter 2 Stratix IV Transceiver Clocking
Revised: *November 2009*
Part Number: *SIV52002-3.0*

Chapter 3 Configuring Multiple Protocols and Data Rates in a Transceiver Block
Revised: *November 2009*
Part Number: *SIV52003-4.0*

Chapter 4 Reset Control and Power Down
Revised: *November 2009*
Part Number: *SIV52004-4.0*

Chapter 5 Stratix IV Dynamic Reconfiguration
Revised: *November 2009*
Part Number: *SIV52005-3.0*

About this Handbook

This handbook provides comprehensive information about the Altera® Stratix® IV family of devices.

How to Contact Altera

For the most up-to-date information about Altera products, see the following table.

Contact *(Note 1)*	Contact Method	Address
Technical support	Website	www.altera.com/support
Technical training	Website	www.altera.com/training
	Email	custrain@altera.com
Product literature	Website	www.altera.com/literature
Non-technical support (General)	Email	nacomp@altera.com
(Software Licensing)	Email	authorization@altera.com

Note:

(1) You can also contact your local Altera sales office or sales representative.

Typographic Conventions

The following table shows the typographic conventions that this document uses.

Visual Cue	Meaning
Bold Type with Initial Capital Letters	Indicates command names, dialog box titles, dialog box options, and other GUI labels. For example, **Save As** dialog box. For GUI elements, capitalization matches the GUI.
bold type	Indicates directory names, project names, disk drive names, file names, file name extensions, dialog box options, software utility names, and other GUI labels. For example, **\qdesigns** directory, **d:** drive, and **chiptrip.gdf** file.
Italic Type with Initial Capital Letters	Indicates document titles. For example, *AN 519: Stratix IV Design Guidelines.*
Italic type	Indicates variables. For example, $n + 1$. Variable names are enclosed in angle brackets (< >). For example, *<file name>* and *<project name>***.pof** file.
Initial Capital Letters	Indicates keyboard keys and menu names. For example, Delete key and the Options menu.
“Subheading Title”	Quotation marks indicate references to sections within a document and titles of Quartus II Help topics. For example, “Typographic Conventions.”

Visual Cue	Meaning
`Courier type`	Indicates signal, port, register, bit, block, and primitive names. For example, `data1`, `tdi`, and `input`. Active-low signals are denoted by suffix n. For example, `resetn`. Indicates command line commands and anything that must be typed exactly as it appears. For example, `c:\qdesigns\tutorial\chiptrip.gdf`. Also indicates sections of an actual file, such as a Report File, references to parts of files (for example, the AHDL keyword `SUBDESIGN`), and logic function names (for example, `TRI`).
1., 2., 3., and a., b., c., and so on.	Numbered steps indicate a list of items when the sequence of the items is important, such as the steps listed in a procedure.
■ ■	Bullets indicate a list of items when the sequence of the items is not important.
☞	The hand points to information that requires special attention.
CAUTION	A caution calls attention to a condition or possible situation that can damage or destroy the product or your work.
WARNING	A warning calls attention to a condition or possible situation that can cause you injury.
↵	The angled arrow instructs you to press **Enter**.
	The feet direct you to more information about a particular topic.

Section I. Transceiver Architecture

This section provides a description of transceiver architecture and transceiver clocking for the Stratix® IV device family. It also describes configuring for multiple protocols and data rates, reset control and power down, and dynamic reconfiguration for Stratix IV devices. This section includes the following chapters:

- Chapter 1, Stratix IV Transceiver Architecture
- Chapter 2, Stratix IV Transceiver Clocking
- Chapter 3, Configuring Multiple Protocols and Data Rates in a Transceiver Block
- Chapter 4, Reset Control and Power Down
- Chapter 5, Stratix IV Dynamic Reconfiguration

Revision History

Refer to each chapter for its own specific revision history. For information on when each chapter was updated, refer to the Chapter Revision Dates section, which appears in the full handbook.

1. Stratix IV Transceiver Architecture

SIV52001-4.0

This chapter provides details about Stratix® IV GX and GT transceiver architecture, transceiver channels, available modes, and a description of transmitter and receiver channel datapaths.

Overview

Stratix IV FPGAs deliver a breakthrough level of system bandwidth and power efficiency for high-end applications, allowing you to innovate without compromise with two offerings: Stratix IV GX and Stratix IV GT.

Stratix IV GX devices are part of the Altera® 40 nm Stratix IV device family. Stratix IV GX devices provide up to 32 full-duplex CDR-based transceivers with physical coding sublayer (PCS) and physical medium attachment (PMA), at serial data rates between 600 Mbps and 8.5 Gbps. Also, Stratix IV GX provides up to 16 additional full-duplex CDR-based transceivers with PMA supporting serial data rates between 600 Mbps and 6.5 Gbps. Table 1–1 shows the Stratix IV GX serial protocols the transceiver channels support.

Table 1–1. Serial Protocols Supported by the Stratix IV GX Transceiver Channels

Protocol	Description
PCI Express (PIPE)	Gen 1 at 2.5 Gbps and Gen 2 at 5.0 Gbps
XAUI	3.125 Gbps to 3.75 Gbps for HiGig support
GIGE	1.25 Gbps
Serial RapidIO	1.25 Gbps, 2.5 Gbps, and 3.125 Gbps
SONET/SDH	OC-12 at 622 Mbps, OC-48 at 2.488 Gbps, and OC-96 at 4.976 Gbps
(OIF) CEI PHY Interface	4.976 Gbps to 6.375 Gbps for Interlaken support
Serial Digital Interface (SDI)	HD-SDI at 1.485 Gbps and 1.4835 Gbps 3G-SDI at 2.97 Gbps and 2.967 Gbps

To implement proprietary protocols, the transceiver channels in the Stratix IV GX device supports the highly flexible Basic single-width (600 Mbps to 3.75 Gbps) and Basic double-width (1 Gbps to 8.5 Gbps) functional modes.

Stratix IV GT devices are also part of Altera's 40 nm Stratix IV device family and contain serial transceivers that support data rates between 2.488 Gbps and 11.3 Gbps. Stratix IV GT devices are targeted towards implementing 40 Gbps/100 Gbps transceiver links. Example applications include 40G/100G Ethernet and SFI-S. Stratix IV GT devices can be broadly classified into the following:

- Stratix IV GT devices targeted to achieve 100 Gbps ingress/egress data rates—48 full duplex clock and clock data recovery (CDR)-based transceivers, 32 of which support data rates up to 11.3 Gbps
- Stratix IV GT devices targeted to achieve 40 Gbps ingress/egress data rates—36 full duplex CDR-based transceivers, 12 of which support data rates up to 11.3 Gbps

Though optimized for 40 Gbps/100 Gbps systems, Stratix IV GT transceivers also provide PMA and PCS support for the following protocols as shown in Table 1–2.

Table 1–2. Serial Protocols Supported by the Stratix IV GT Transceiver Channels

Protocol	Description
PCI Express (PIPE)	Gen 1 at 2.5 Gbps and Gen 2 at 5.0 Gbps
XAUI	3.125 Gbps up to HiGig at 3.75 Gbps
Serial RapidIO	2.5 Gbps and 3.125 Gbps
SONET/SDH	OC-48 and OC-96
(OIF) CEI PHY Interface	4.976 Gbps to 6.375 Gbps
Serial Digital Interface (SDI)	3G-SDI at 2.97Gbps and 2.967 Gbps

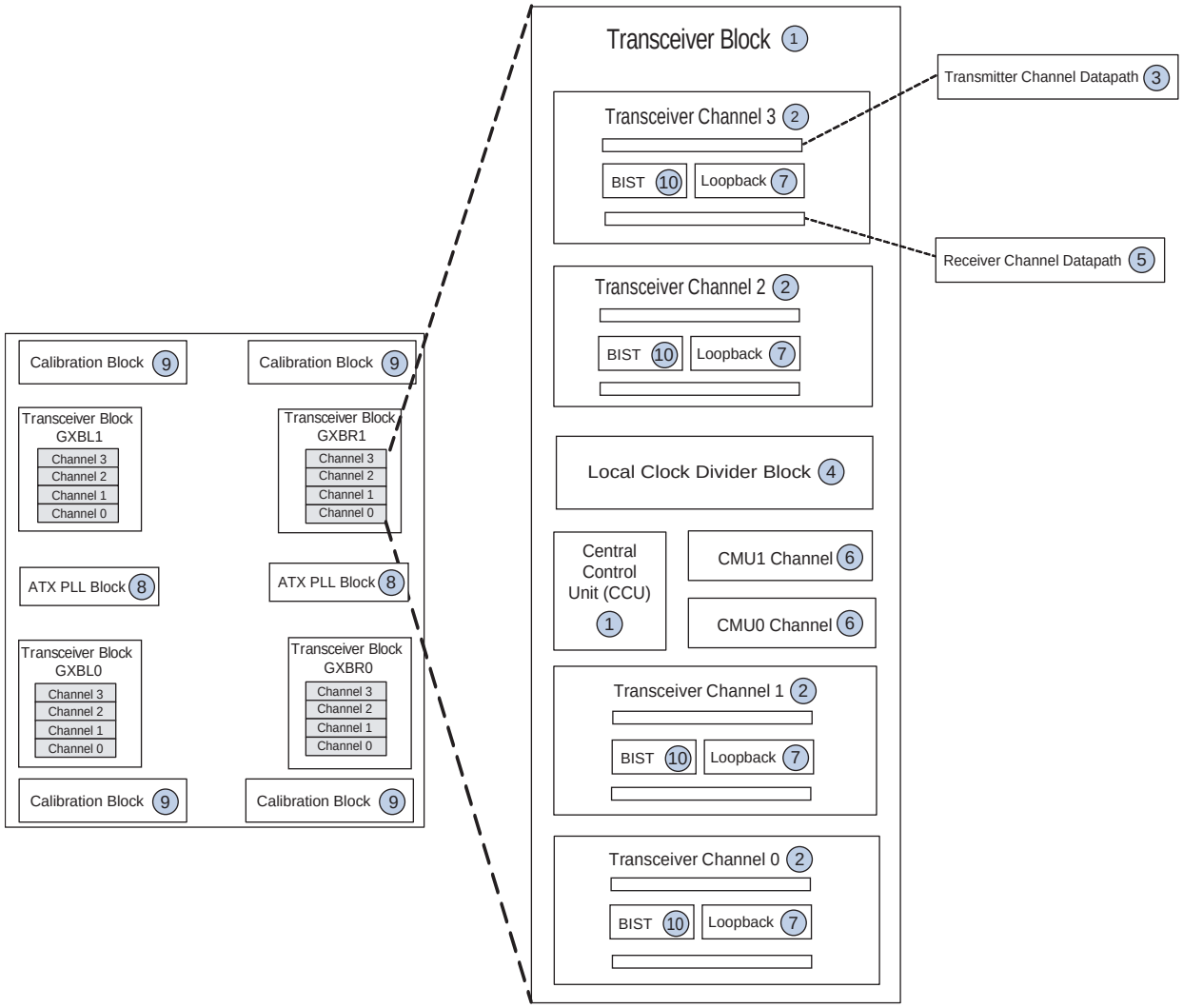
To implement proprietary protocols, the transceiver channels in the Stratix IV GT device support the highly flexible Basic single-width (2.488 Gbps to 3.75 Gbps) and Basic double-width (2.488 Gbps to 11.3 Gbps) functional modes.

Stratix IV GX and GT devices have PCI Express hard IP, PCS, and PMA blocks. For more information, refer to the *PCI Express Compiler Guide*.

For more information about Stratix IV GX and GT protocols, refer to the *Configuring Multiple Protocols and Data Rates in a Transceiver Block* chapter.

Figure 1–1 shows an example of the Stratix IV GX and GT transceiver architecture. Links to the corresponding transceiver architecture descriptions are listed below. This is an elementary diagram and does not represent an actual transceiver block.

Figure 1–1. Example of a Transceiver Block



Descriptions for the example transceiver architecture are as follows:

1. "Transceiver Block Architecture" on page 1–11
2. "Transceiver Channel Architecture" on page 1–12
3. "Transmitter Channel Datapath" on page 1–14
4. "Transmitter Local Clock Divider Block" on page 1–35
5. "Receiver Channel Datapath" on page 1–36
6. "CMU Channel Architecture" on page 1–97
7. "Loopback Modes" on page 1–188

8. "Auxiliary Transmit (ATX) PLL Block" on page 1–193
9. "Calibration Blocks" on page 1–199
10. "Built-In Self Test Modes" on page 1–204

Transceiver Channel Locations

Stratix IV GX and GT transceivers are structured into full-duplex (Transmitter and Receiver) four-channel groups called transceiver blocks. The total number of transceiver channels and the location of transceiver blocks varies from device to device.

Stratix IV GX Device Offerings

Table 1–3 summarizes the total number of transceiver channels and transceiver block locations in each Stratix IV GX device member.

Table 1–3. Number of Transceiver Channels and Transceiver Block Locations in Stratix IV GX Devices (Part 1 of 2)

Device Member	Total Number of Transceiver Channels	Transceiver Channel Location
EP4SGX70DF29 EP4SGX110DF29 EP4SGX180DF29 EP4SGX230DF29	8	Eight transceiver channels located in two transceiver blocks: ■ Right side—GXBR0 and GXBR1 Refer to Figure 1–2 on page 1–5.
EP4SGX290FH29 EP4SGX360FH29 EP4SGX110FF35 EP4SGX180FF35 EP4SGX230FF35 EP4SGX290FF35 EP4SGX360FF35	16	Eight transceiver channels located in two transceiver blocks: ■ Right side—GXBR0 and GXBR1 ■ Left side—GXBL0 and GXBL1 Refer to Figure 1–2 on page 1–5.
EP4SGX180HF35 EP4SGX230HF35 EP4SGX290HF35 EP4SGX360HF35 EP4SGX530HH35	24	Eight regular transceiver channels supporting data rates between 600 Mbps and 8.5 Gbps and four clock multiplier unit (CMU) channels supporting data rates between 600 Mbps and 6.5 Gbps located in two transceiver blocks: ■ Right side—GXBR0 and GXBR1 ■ Left side—GXBL0 and GXBL1 Refer to Figure 1–3 on page 1–6.

Table 1–3. Number of Transceiver Channels and Transceiver Block Locations in Stratix IV GX Devices (Part 2 of 2)

Device Member	Total Number of Transceiver Channels	Transceiver Channel Location
EP4SGX180KF40 EP4SGX230KF40 EP4SGX290KF40 EP4SGX360KF40 EP4SGX530KF40	36	Twelve regular transceiver channels supporting data rates between 600 Mbps and 8.5 Gbps and six CMU channels supporting data rates between 600 Mbps and 6.5 Gbps located in three transceiver blocks: ■ Right side—`GXBR0`, `GXBR1`, and `GXBR2` ■ Left side—`GXBL0`, `GXBL1`, and `GXBL2` Refer to Figure 1–3 on page 1–6.
EP4SGX530NF45	48	Sixteen regular transceiver channels supporting data rates between 600 Mbps and 8.5 Gbps and eight CMU channels supporting data rates between 600 Mbps and 6.5 Gbps located in four transceiver blocks: ■ Right side—`GXBR0`, `GXBR1`, `GXBR2`, and `GXBR3` ■ Left side—`GXBL0`, `GXBL1`, `GXBL2`, and `GXBL3` Refer to Figure 1–3 on page 1–6.

Figure 1–2 shows transceiver channel locations in each Stratix IV GX device member that have 8 or 16 transceiver channels.

Figure 1–2. Stratix IV GX Devices with 8 and 16 Transceiver Channels

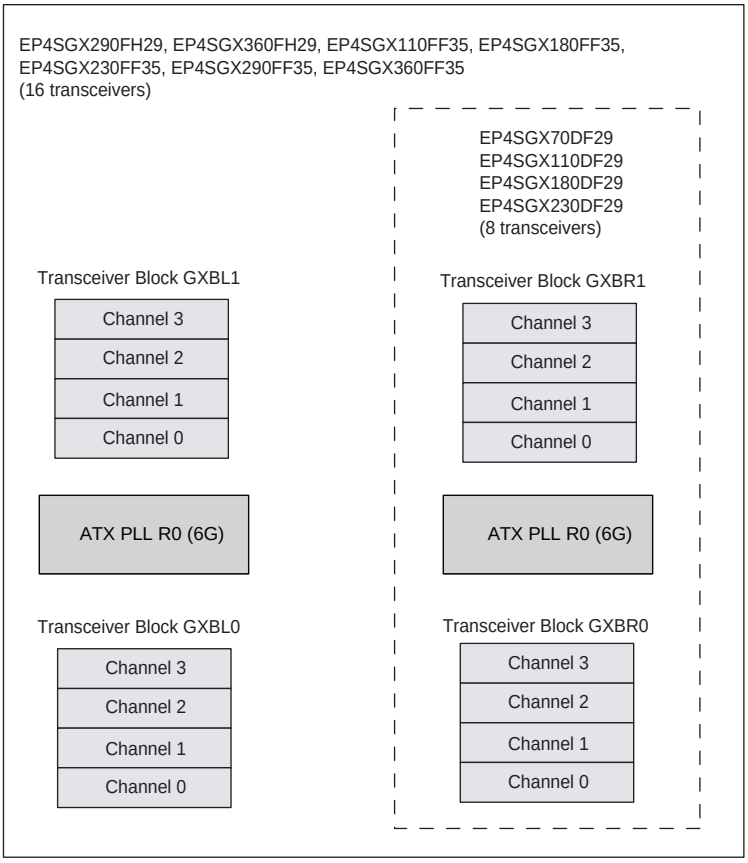
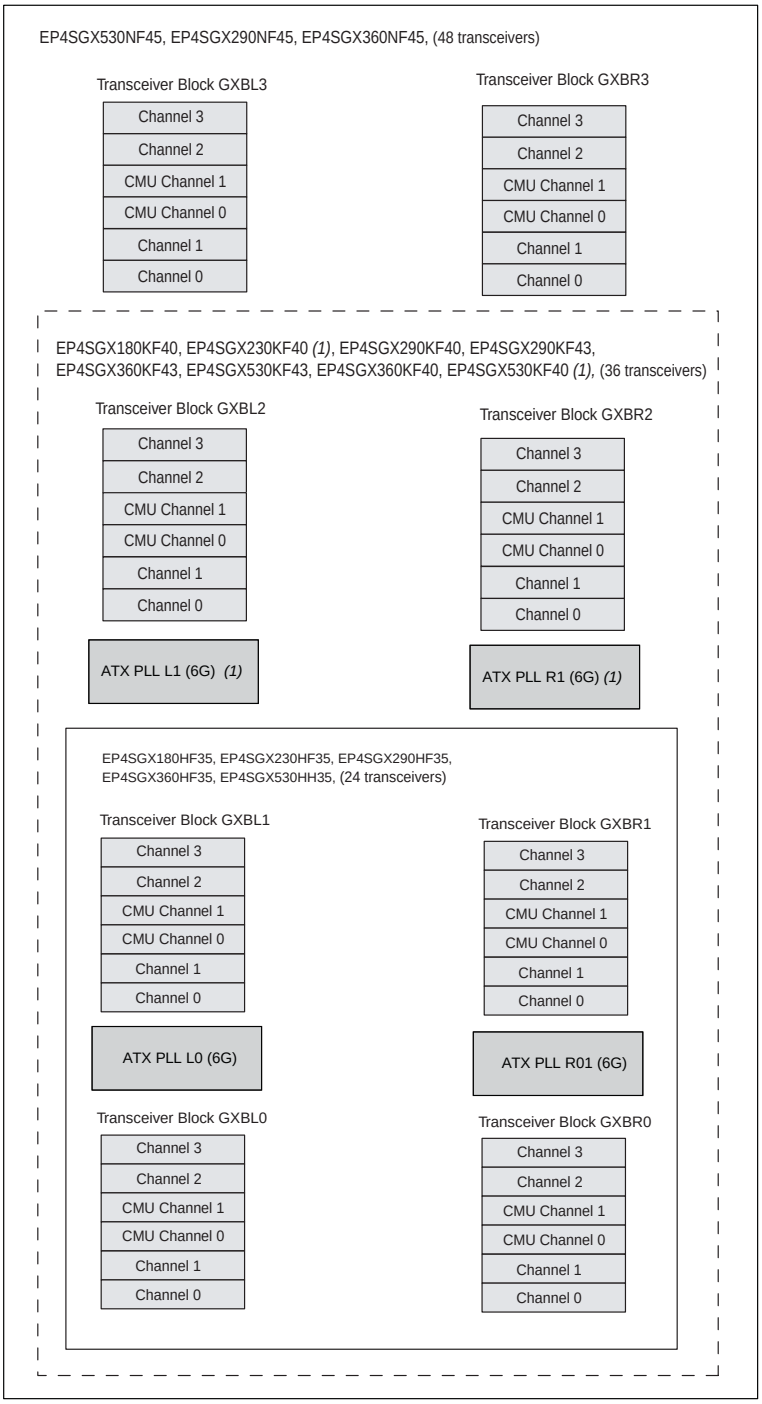


Figure 1–3 shows transceiver channel locations in each Stratix IV GX device member that have 24, 36, or 48 transceiver channels.

Figure 1–3. Stratix IV GX Device with 24, 36, or 48 Transceiver Channels



Note to Figure 1–3:

(1) The 6G ATX PLL R1 and L1 blocks are not available in the EP4SGX230KF40 and EP4SGX530KF40 devices.

Stratix IV GT Device Offerings

Table 1–4 lists the Stratix IV GT device offerings along with the number of transceiver channels available in each device.

Table 1–4. Stratix IV GT Device Offerings and Transceiver Channels Available in Each Device

Transceiver Channels	EP4S40G2F40 *(4)*	EP4S40G5H40	EP4S100G2F40 *(4)*	EP4S100G5H40	EP4S100G3F45, EP4S100G4F45	EP4S100G5F45
Total Transceiver Channels	36	36	36	36	48	48
10G Transceiver Channels *(1)*	12	12	24	24	24	32
8G Transceiver Channels *(2)*	12	12	0	0	8	0
CMU Channels (PMA-only) *(3)*	12	12	12	12	16	16

Notes to Table 1–4:

(1) 10G transceiver channels support data rates between 2.488 Gbps and 11.3 Gbps.

(2) 8G transceiver channels support data rates between 2.488 Gbps and 8.5 Gbps. All 10G transceiver channels can also be configured as 8G transceiver channels. For example, the EP4S40G2F40 device has twenty-four 8G transceiver channels and the EP4S100G5F45 device has thirty-two 8G transceiver channels.

(3) CMU channels that support data rates between 2.488 Gbps and 6.5 Gbps are PMA-only channels that do not have PCS circuitry. For more information, refer to "CMU Channel Architecture" on page 1–97.

(4) F40 devices use 1517-pin flip chip packages. H40 devices use 1517-pin hybrid flip chip packages. F45 devices use 1932-pin flip chip packages.

Table 1–5 lists the transceiver blocks in each Stratix IV GT device that support transceiver channels up to 11.3 Gbps.

Table 1–5. Transceiver Blocks in Stratix IV GT Devices Supporting Transceiver Channels up to 11.3 Gbps (Part 1 of 2)

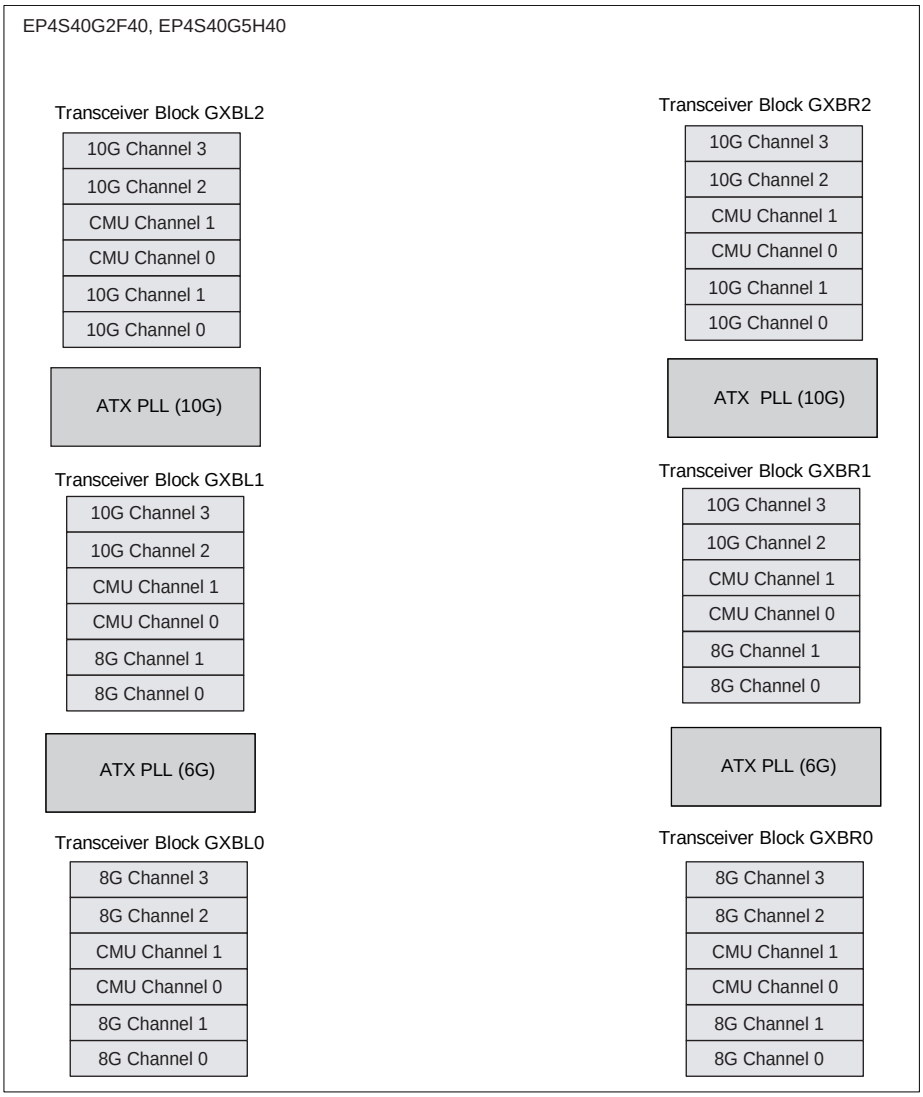
Device Member	Total number of Transceiver Channels	Transceiver Channel Location
EP4S40G2F40	36	12 regular transceiver channels (six 10G and six 8G), located in three transceiver blocks: Left Side—4 in GXBL2, 2 in GXBL1 Right Side—4 in GXBR2, 2 in GXBR1 Refer to Figure 1–4 on page 1–9.
EP4S40G5H40	36	12 regular transceiver channels (six 10G and six 8G), located in three transceiver blocks: Left Side—4 in GXBL2, 4 in GXBL1, 4 in GXBL0 Right Side—4 in GXBR2, 4 in GXBR1, 4 in GXBR0 Refer to Figure 1–4 on page 1–9.

Table 1–5. Transceiver Blocks in Stratix IV GT Devices Supporting Transceiver Channels up to 11.3 Gbps (Part 2 of 2)

Device Member	Total number of Transceiver Channels	Transceiver Channel Location
EP4S100G2F40	36	12 regular transceiver channels, capable of 10G each, located in three transceiver blocks: Left Side—4 in GXBL2, 4 in GXBL1, 4 in GXBL0 Right Side—4 in GXBR2, 4 in GXBR1, 4 in GXBR0 Refer to Figure 1–5 on page 1–10.
EP4S100G5H40	36	12 regular transceiver channels, capable of 10G each, located in three transceiver blocks: Left Side—4 in GXBL2, 4 in GXBL1, 4 in GXBL0 Right Side—4 in GXBR2, 4 in GXBR1, 4 in GXBR0 Refer to Figure 1–5 on page 1–10.
EP4S100G3F45	48	16 regular transceiver channels (twelve 10G and four 8G) located in four transceiver blocks: Left Side—4 in GXBL3, 4 in GXBL2, 4 in GXBL1, 4 in GXBL0 Right Side—4 in GXBR3, 4 in GXBR2, 4 in GXBR1, 4 in GXBR0 Refer to Figure 1–5 on page 1–10.
EP4S100G4F45	48	16 regular transceiver channels (twelve 10G and four 8G) located in four transceiver blocks: Left Side—4 in GXBL3, 4 in GXBL2, 4 in GXBL1, 4 in GXBL0 Right Side—4 in GXBR3, 4 in GXBR2, 4 in GXBR1, 4 in GXBR0 Refer to Figure 1–5 on page 1–10.
EP4S100G5F45	48	16 regular transceiver channels, all capable of 10G each, located in four transceiver blocks: Left Side—4 in GXBL3, 4 in GXBL2, 4 in GXBL1, 4 in GXBL0 Right Side—4 in GXBR3, 4 in GXBR2, 4 in GXBR1, 4 in GXBR0 Refer to Figure 1–5 on page 1–10.

Figure 1–4 and Figure 1–5 show transceiver channel and PLL locations for all Stratix IV GT devices offered.

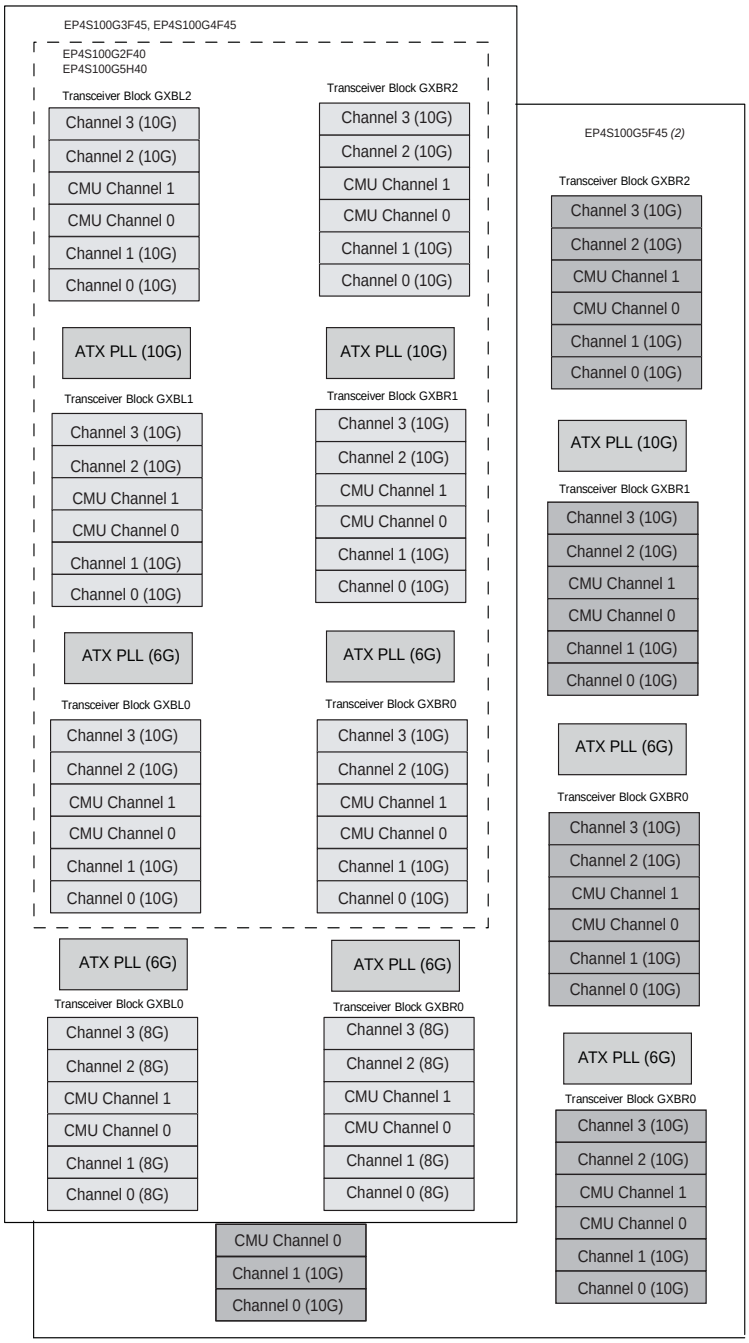
Figure 1–4. Transceiver Channel and PLL Locations in EP4S40G2F40 and EP4S40G5H40 Devices *(Note 1)*



Note to Figure 1–4:

(1) EP4S40G2F40C2ES1 devices do not have 10G auxiliary transmit (ATX) PLL blocks. Use the CMU PLL to generate transceiver clocks for channels configured at 11.3 Gbps.

Figure 1–5. Transceiver Channel and PLL Locations in EP4S100G2F40, EP4S100G5H40, EP4S100G3F45, EP4S100G4F45, and EP4S100G5F45 Devices



Note to Figure 1–5:

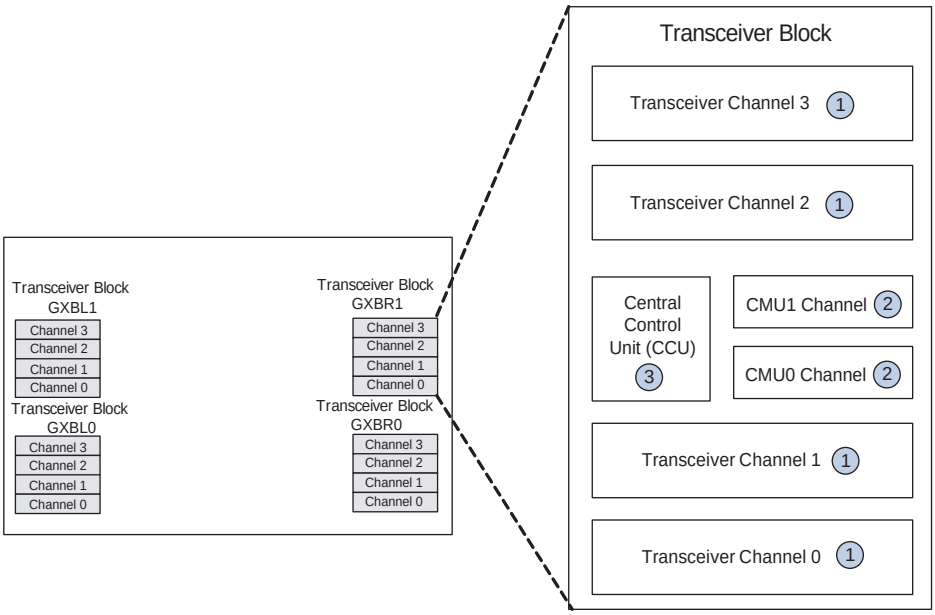
(1) EP4S100G2F40C2ES1 devices do not have 10G ATX PLL blocks. Use the CMU PLL to generate transceiver clocks for channels configured at 11.3 Gbps.

(2) EP4S100G5F45 devices are the same as EP4S100G3F45 and EP4S100G4F45 devices except that the GXBR0 transceiver block is 10G instead of 8G.

Transceiver Block Architecture

Figure 1–6 shows the transceiver block architecture of Stratix GX and GT devices.

Figure 1–6. Top-Level View of a Transceiver Block



Each transceiver block has the following components:

1. Four full-duplex (transmitter and receiver) transceiver channels that support serial data rates from 600 Mbps to 8.5 Gbps in Stratix IV GX devices and 2.488 Gbps to 11.3 Gbps in Stratix IV GT devices. For more information, refer to "Transceiver Channel Architecture" on page 1–12.
2. Two CMU channels—`CMU0` and `CMU1` channels—that provide the high-speed serial and low-speed parallel clock to the transceiver channels. For more information, refer to "CMU Channel Architecture" on page 1–97.
3. Central control unit (CCU) that implements the XAUI state machine for XGMII-to-PCS code group conversion, XAUI deskew state machine, shared control signal generation block, PCI Express (PIPE) rateswitch controller block, and reset control logic
   - The shared control signal generation block provides control signals to the transceiver channels in bonded functional modes, such as XAUI, PCI Express (PIPE), and Basic ×4.
   - The PCI Express (PIPE) rateswitch controller block controls the rateswitch circuit in the `CMU0` channel in ×4 configurations. In PCI Express (PIPE) ×8 configuration, the PCI Express (PIPE) rateswitch controller block of the CCU in the master transceiver block is active. For more information, refer to "PCI Express (PIPE) Gen2 (5 Gbps) Support" on page 1–138.

The Stratix IV GT transceiver architecture has the following components:

- Regular transceiver channels with PMA and PCS support
- CMU channels with PMA-only support
- ATX PLL blocks

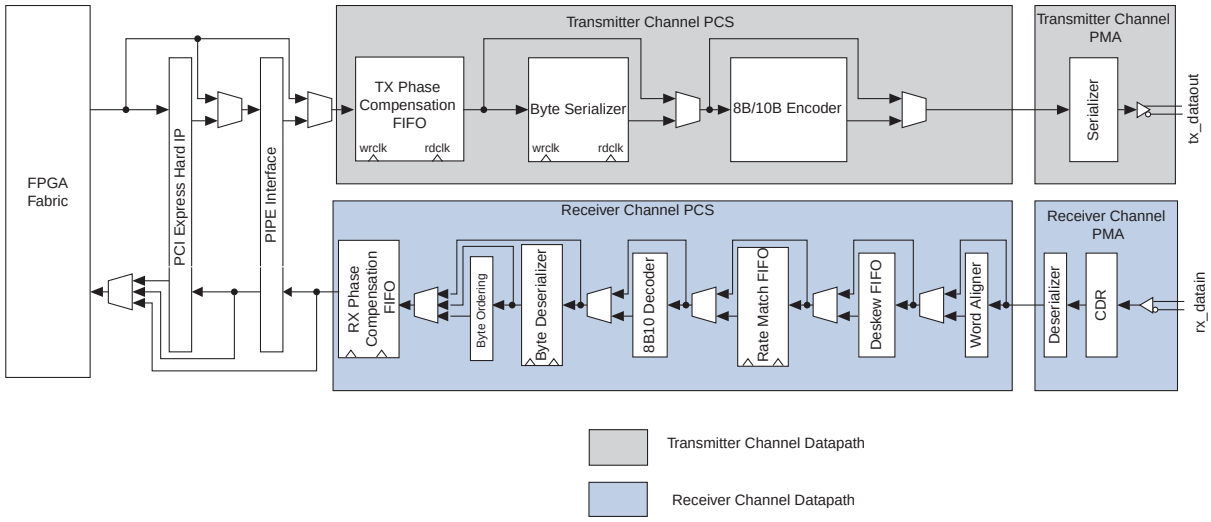
Four transceiver channels and two CMU channels are located in each transceiver block on the left and right sides of the device. Each Stratix IV GT device also has two 10G ATX PLLs that support data rates between 9.9 Gbps and 11.3 Gbps. Additionally, each Stratix IV GT device has two 6G ATX PLLs that support data rates between 2.488 Gbps and 6.5 Gbps, except the EP4S100G5F45 device that has four 6G ATX PLLs.

The 6G ATX PLL does not support all data rates between 2.488 Gbps and 6.5 Gbps.

Transceiver Channel Architecture

Figure 1–7 shows the Stratix IV GX and GT transceiver channel datapath.

Figure 1–7. Stratix IV GX and GT Transceiver Datapath



Each transceiver channel consists of the:

- Transmitter channel, further divided into:
  - Transmitter channel PCS
  - Transmitter channel PMA
- Receiver channel, further divided into:
  - Receiver channel PCS
  - Receiver channel PMA

Each transceiver channel interfaces to either the PCI Express (PIPE) hard IP block (PCI Express [PIPE] hard IP-transceiver interface) or directly to the FPGA fabric (FPGA fabric-transceiver interface). The transceiver channel interfaces to the PCI Express (PIPE) hard IP block if the hard IP block is used to implement the PCI Express (PIPE) PHY MAC, data link layer, and transaction layer. Otherwise, the transceiver channel interfaces directly to the FPGA fabric.

Each regular Stratix IV GT transceiver channel can be categorized into:

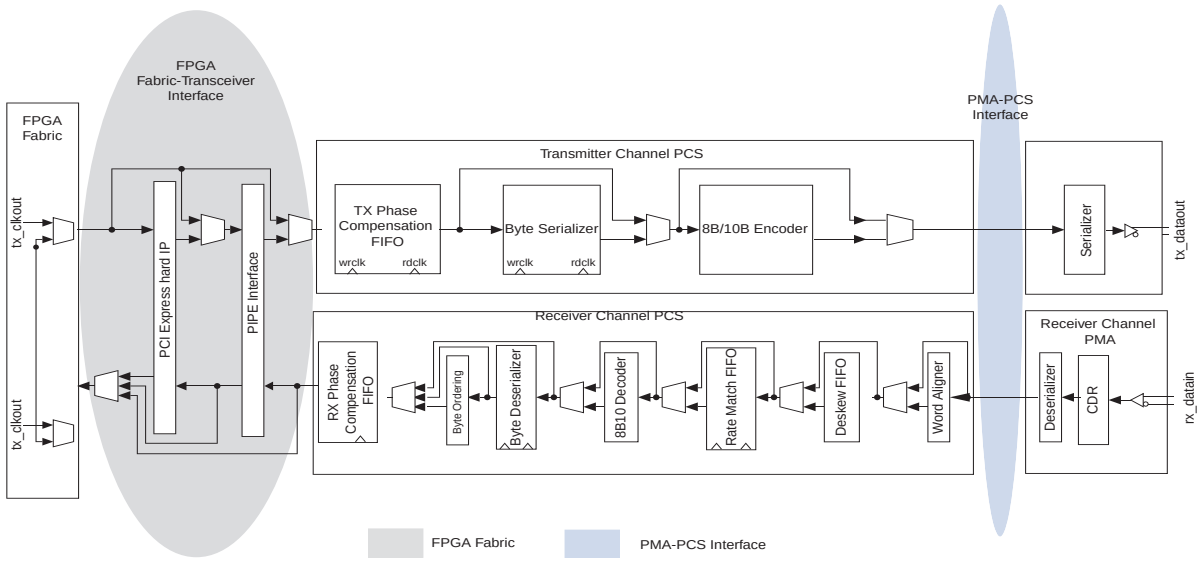
- 8G transceiver channel—supports data rates between 2.488 Gbps and 8.5 Gbps
- 10G Transceiver Channel—supports data rates between 2.488 Gbps and 11.3 Gbps

The PCI Express (PIPE) hard IP-transceiver interface is beyond the scope of this chapter. This chapter describes the FPGA fabric-transceiver interface.

For more information about the PCI Express (PIPE) hard IP block, refer to the *PCI Express Compiler User Guide*.

Figure 1–8 shows the FPGA fabric-transceiver interface and transceiver PMA-PCS interface.

Figure 1–8. FPGA Fabric-Transceiver Interface and Transceiver PMA-PCS Interface



The transceiver channel datapath can be divided into the following two modes based on the FPGA fabric-transceiver interface width (channel width) and the transceiver channel PMA-PCS width (serialization factor):

- Single-width mode
- Double-width mode

Table 1–6 shows the FPGA fabric-transceiver interface widths (channel width) and transceiver PMA-PCS widths (serialization factor) allowed in single-width and double-width modes.

Table 1–6. FPGA Fabric-Transceiver Interface Width and Transceiver PMA-PCS Widths

Name	Single-Width	Double-Width
PMA-PCS interface widths	8/10 bit	16/20 bit
FPGA fabric-transceiver interface width	8/10 bit 16/20 bit	16/20 bit 32/40 bit
Supported functional modes	■ PCI Express (PIPE) Gen1 and Gen2 ■ XAUI ■ GIGE ■ Serial RapidIO ■ SONET/SDH OC12 and OC48 ■ SDI ■ Basic single-width	■ (OIF) CEI PHY Interface ■ SONET/SDH OC96 ■ Basic double-width
Data rate range in Basic functional mode	0.6 Gbps to 3.75 Gbps	1 Gbps to 8.5 Gbps

Transmitter Channel Datapath

The transmitter channel datapath, shown in Figure 1–7 on page 1–12, consists of the following blocks:

- TX phase compensation FIFO
- Byte serializer
- 8B/10B encoder
- Transmitter output buffer

The Stratix IV GX and GT transceiver provides the **Enable low latency PCS mode** option in the ALTGX MegaWizard™ Plug-In Manager. If you select this option, the 8B/10B encoder in the datapath is disabled.

TX Phase Compensation FIFO

The TX phase compensation FIFO interfaces the transmitter channel PCS and the FPGA fabric PCI Express (PIPE) interface. It compensates for the phase difference between the low-speed parallel clock and the FPGA fabric interface clock. The TX phase compensation FIFO operates in low-latency and high-latency modes. Figure 1–9 shows the datapath and clocking of the TX phase compensation FIFO.

Figure 1–9. TX Phase Compensation FIFO

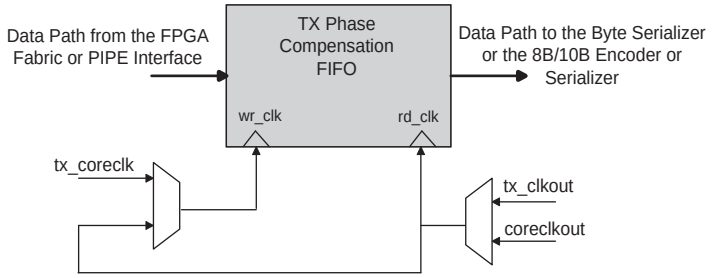


Table 1–7 lists the TX phase compensation FIFO modes.

Table 1–7. TX Phase Compensation FIFO Modes

Modes	Description
Low-Latency	The FIFO is four words deep. Latency through the FIFO is two to three FPGA fabric parallel clock cycles (pending characterization). The default setting for every mode.
High-Latency	The FIFO is eight words deep. The latency through the FIFO is four to five FPGA parallel cycles (pending characterization).
Non-Bonded Functional	For example, in GIGE mode, the read port of the phase compensation FIFO is clocked by the low-speed parallel clock. The write clock is fed by the `tx_clkout` port of the associated channel.
Bonded Functional	For example, in XAUI mode, the write clock of the FIFO is clocked by `coreclkout` provided by the `CMU0` clock divider block. You can clock the write side using `tx_coreclk` provided from the FPGA fabric by enabling the `tx_coreclk` port in the ALTGX MegaWizard Plug-In Manager. If you use this port, ensure that there is 0 parts-per-million (PPM) difference in frequency between the write and read side. The Quartus® II software requires that you provide a 0 PPM assignment in the Assignment Editor.

For more information about the TX phase compensation FIFO, refer to the "Limitations of the Quartus II Software-Selected Transmitter Phase Compensation FIFO Write Clock" section of the *Stratix IV Transceiver Clocking* chapter.

Input Data

In PCI Express (PIPE) functional mode, the input data comes from the PCI Express (PIPE) interface. In all other functional modes, the input data comes directly from the FPGA fabric.

Output Data Destination Block

The output from the TX phase compensation FIFO is used by the byte serializer block, 8B/10B encoder, or serializer block. Table 1–8 lists the conditions under which the TX phase compensation FIFO outputs are provided to these blocks.

Table 1–8. Output Data Destination Block for the TX Phase Compensation FIFO Output Data

Byte Serializer	8B/10B Encoder	Serializer
If you select: single-width mode and channel width = 16 or 20	If you select: single-width mode and channel width = 8 and 8B/10B encoder enabled	If you select: low-latency PCS bypass mode enabled or single-width mode and channel width = 8 or 10
If you select: double-width mode and channel width = 32 or 40	If you select: double-width mode and channel width = 16 and 8B/10B encoder enabled	If you select: low-latency PCS bypass mode enabled or double-width mode and channel width = 16 or 20

TX Phase Compensation FIFO Status Signal

An optional `tx_phase_comp_fifo_error` port is available in all functional modes to indicate a receiver phase compensation FIFO overflow or under-run condition. The `tx_phase_comp_fifo_error` signal is asserted high when the TX phase compensation FIFO either overflows or under-runs due to any frequency PPM difference between the FIFO read and write clocks. If the `tx_phase_comp_fifo_error` flag is asserted, verify the FPGA fabric-transceiver interface clocking to ensure that there is 0 PPM difference between the TX phase compensation FIFO read and write clocks.

Byte Serializer

The byte serializer divides the input datapath by two. This allows you to run the transceiver channel to run at higher data rates while keeping the FPGA fabric interface frequency within the maximum limit stated in the "Interface Frequency" section in the *DC and Switching Characteristics* chapter. In single-width mode, it converts the two byte wide datapath to a one byte wide datapath. In double-width mode, it converts the four-byte wide datapath to a two byte wide datapath. It is optional in configurations that do not exceed the FPGA fabric-transceiver interface maximum frequency limit.

For example, if you want to run the transceiver channel at 6.25 Gbps, without the byte serializer in double-width mode, the FPGA fabric interface clock frequency must be 312.5 MHz (6.25/20). This violates the FPGA fabric interface frequency limit. When you use the byte serializer, the FPGA fabric interface frequency is 156.25 MHz (6.25G/40). You can enable the byte serializer in single-width or double-width mode.

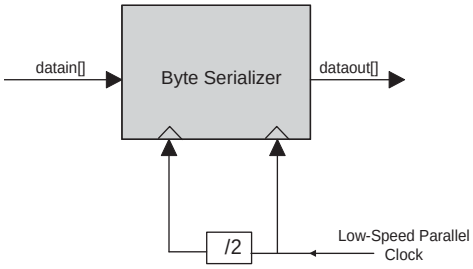
The byte deserializer is required in configurations that exceed the FPGA fabric-transceiver interface maximum frequency limit.

For more information about the maximum frequency limit for the transceiver interface, refer to the *Stratix IV Device Datasheet* section.

Single-Width Mode

Figure 1–10 shows the byte serializer datapath in single-width mode. For data port width, refer to Table 1–9.

Figure 1–10. Byte Serializer Datapath in Single-Width Mode *(Note 1)*, *(2)*



Notes to Figure 1–10:

(1) For the `datain[]` and `dataout[]` port widths, refer to Table 1–9 on page 1–17.

(2) The `datain` signal is the input from the FPGA fabric that has already passed through the TX phase compensation FIFO.

The byte serializer forwards the least significant byte first, followed by the most significant byte. The input data width to the byte serializer depends on the channel width option that you selected in the ALTGX MegaWizard Plug-In Manager. For example, in single-width mode, assuming a channel width of 20, the byte serializer sends out the least significant word `datain[9:0]` of the parallel data from the FPGA fabric, followed by `datain[19:10]`. Table 1–9 shows the input and output data widths of the byte serializer in single-width mode.

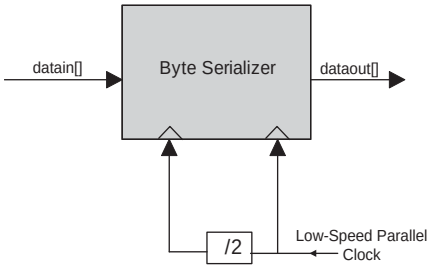
Table 1–9. Input and Output Data Width of the Byte Serializer in Single-Width Mode

Deserialization Width	Input Data Width to the Byte Serializer	Output Data Width from the Byte Serializer
Single-width mode	16	8
	20	10

Double-Width Mode

Figure 1–11 shows the byte serializer datapath in double-width mode. For data port width, refer to Table 1–10.

Figure 1–11. Byte Serializer Datapath in Double-Width Mode *(Note 1)*, *(2)*



Notes to Figure 1–11:

(1) For the `datain[]` and `dataout[]` port width, refer to Table 1–10.

(2) The `datain` signal is the input from the FPGA fabric that has already passed through the TX phase compensation FIFO.

The operation in double-width mode is similar to that of single-width mode. For example, assuming a channel width of 40, the byte serializer forwards `datain[19:0]` first, followed by `datain[39:20]`. Table 1–10 shows the input and output data widths of the byte serializer in double-width mode.

Table 1–10. Input and Output Data Width of the Byte Serializer in Double-Width Mode

Deserialization Width	Input Data Width to the Byte Serializer	Output Data Width from the Byte Serializer
Double-width mode	32	16
	40	20

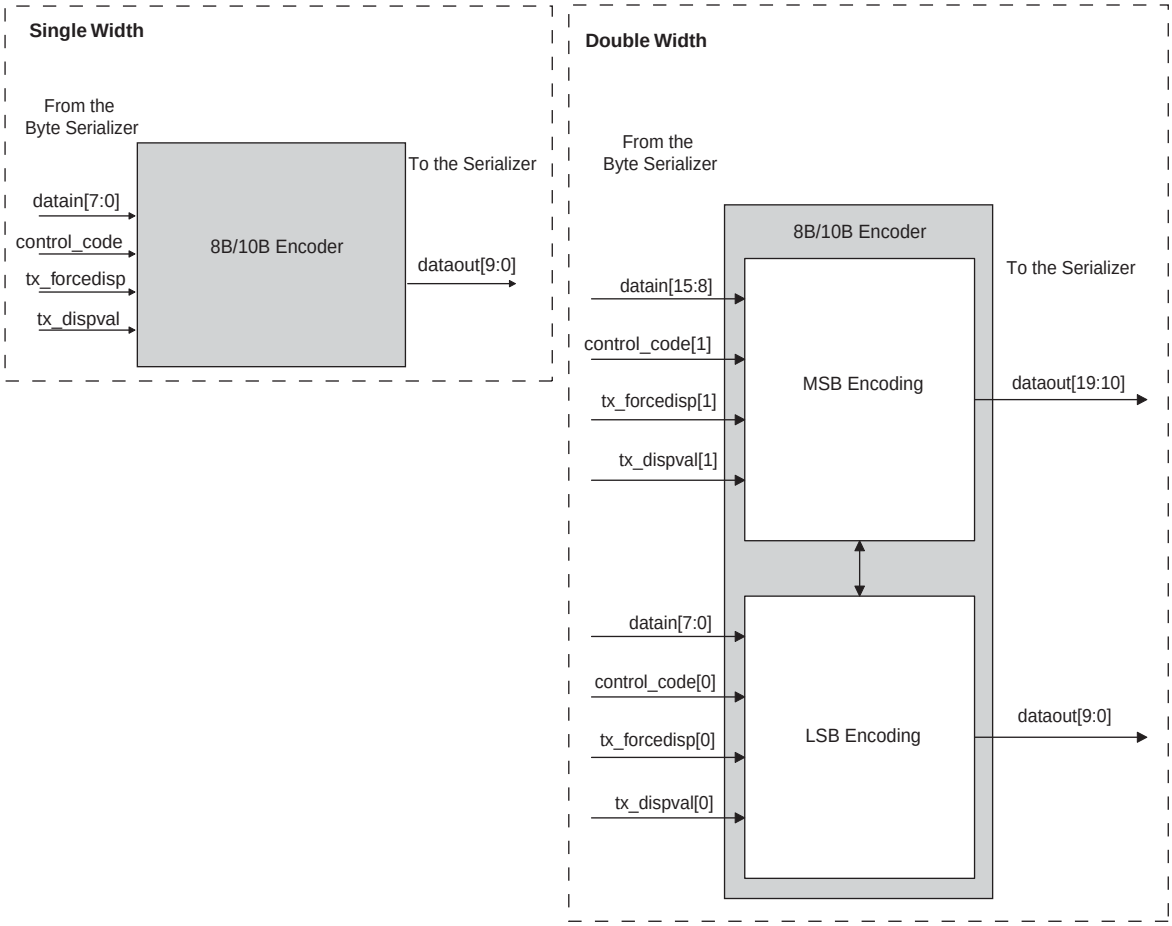
Asserting the `tx_digitalreset` signal resets the byte serializer block.

If you select the **8B/10B Encoder** option in the ALTGX MegaWizard Plug-In Manager, the 8B/10B encoder uses the output from the byte serializer. Otherwise, the byte serializer output is forwarded to the serializer.

8B/10B Encoder

The 8B/10B encoder generates 10-bit code groups from the 8-bit data and 1-bit control identifier. The 8B/10B encoder operates in two modes: single-width and double-width. Figure 1–12 shows the 8B/10B encoder in single-width and double-width mode.

Figure 1–12. 8B/10B Encoder in Single-Width Mode



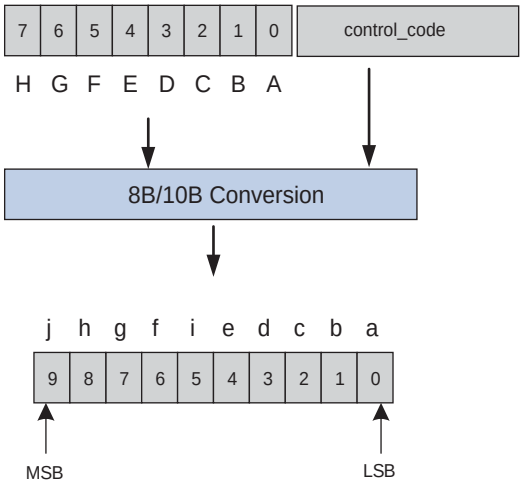
Single-Width Mode

The left side of Figure 1–12 shows the 8B/10B encoder in single-width mode. In this mode, the 8B/10B encoder translates the 8-bit data to a 10-bit code group (control word or data word) with proper disparity. If the `control_code` input is high, the 8B/10B encoder translates the input `data[7:0]` to a 10-bit control word. If the `control_code` input is low, the 8B/10B encoder translates the input `data[7:0]` to a 10-bit data word.

You can use the `tx_forcedisp` and `tx_dispval` ports to control the running disparity of the generated output data. For more information, refer to "Controlling Running Disparity" on page 1–23.

Figure 1–13 shows the conversion format. The LSB is transmitted first.

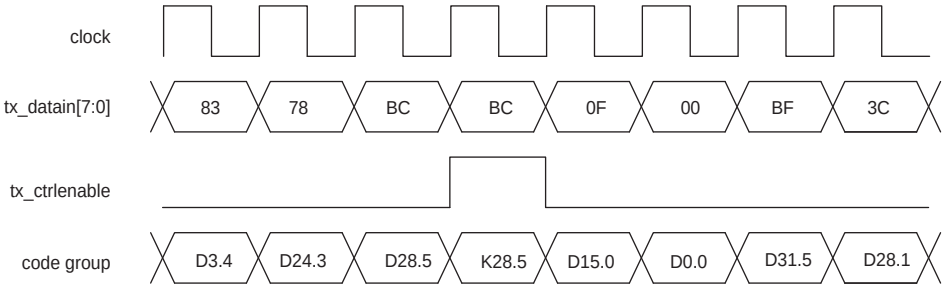
Figure 1–13. 8B/10B Conversion Format



Control Code Encoding

The ALTGX MegaWizard Plug-In Manager provides the `tx_ctrlenable` port to indicate whether the 8-bit data at the `tx_datain` port should be encoded as a control word (Kx.y). When `tx_ctrlenable` is low, the 8B/10B encoder block encodes the byte at the `tx_datain` port (the user-input port) as data (Dx.y). When `tx_ctrlenable` is high, the 8B/10B encoder encodes the input data as a Kx.y code group. The waveform in Figure 1–14 shows the second 0 × BC encoded as a control word (K28.5). The rest of the `tx_datain` bytes are encoded as a data word (Dx.y).

Figure 1–14. Control Word and Data Word Transmission



The IEEE 802.3 8B/10B encoder specification identifies only a set of 8-bit characters for which `tx_ctrlenable` should be asserted. If you assert `tx_ctrlenable` for any other set of bytes, the 8B/10B encoder might encode the output 10-bit code as an invalid code (it does not map to a valid Dx.y or Kx.y code), or unintended valid Dx.y code, depending on the value entered. It is possible for a downstream 8B/10B decoder to decode an invalid control word into a valid Dx.y code without asserting code error flags.

For example, depending on the current running disparity, the invalid code K24.1 (`tx_datain` = 8'h38 + `tx_ctrl` = 1'b1) can be encoded to 10'b0110001100 (0 × 18C), which is equivalent to a D24.6+ (8'hD8 from the RD+ column). Altera recommends that you do not assert `tx_ctrlenable` for unsupported 8-bit characters.

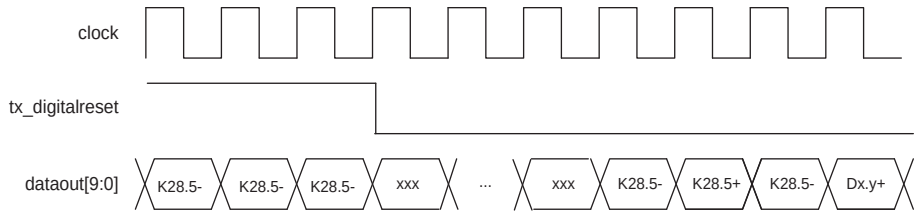
Reset Condition

The `tx_digitalreset` signal resets the 8B/10B encoder. During reset, running disparity and data registers are cleared. Also, the 8B/10B encoder outputs a K28.5 pattern from the RD- column continuously until `tx_digitalreset` is de-asserted. The input data and control code from the FPGA fabric is ignored during the reset state. After reset, the 8B/10B encoder starts with a negative disparity (RD-) and transmits three K28.5 code groups for synchronization before it starts encoding and transmitting the data on its output.

While `tx_digitalreset` is asserted, the downstream 8B/10B decoder that receives the data might observe synchronization or disparity errors.

Figure 1–15 shows the reset behavior of the 8B/10B encoder. When in reset (`tx_digitalreset` is high), a K28.5- (K28.5 10-bit code group from the RD- column) is sent continuously until `tx_digitalreset` is low. Due to some pipelining of the transmitter channel PCS, some "don't cares" (10'hxxx) are sent before the three synchronizing K28.5 code groups. User data follows the third K28.5 code group.

Figure 1–15. 8B/10B Encoder Output During tx_digitalreset Assertion

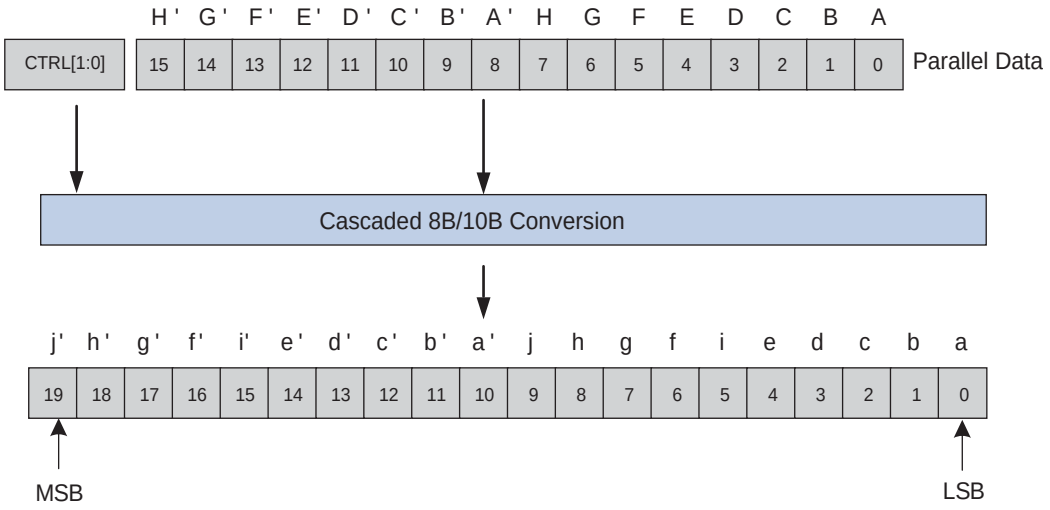


Double-Width Mode

In double-width mode, the 8B/10B encoder operates in a cascaded mode, as shown on the right side of Figure 1–15 on page 1–21. The LSByte of the input data is encoded and transmitted prior to the MSByte.

In double-width mode, the cascaded 8B/10B encoder generates two 10-bit code groups from two 8-bit data and their respective control code identifiers. Figure 1–16 shows the conversion format. The LSB shown in Figure 1–16 is transmitted first.

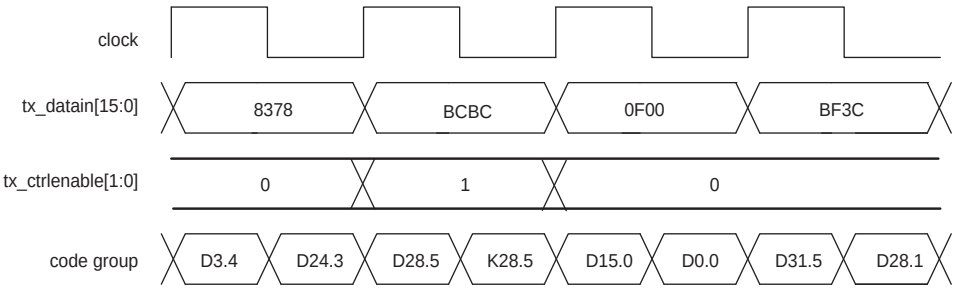
Figure 1–16. 8B/10B Conversion Format in Double-Width Mode



Control Code Encoding

In double-width mode, the `tx_ctrlenable[1:0]` port is used to identify which 8-bit data is to be encoded as a control word. The lower bit, `tx_ctrlenable[0]`, is associated with the LSByte; the upper bit, `tx_ctrlenable[1]`, is associated with the MSByte. When `tx_ctrlenable` is low, the byte at the `tx_datain` port of the transceiver is encoded as data (Dx.y); otherwise, it is encoded as a control code (Kx.y). Figure 1–17 shows that only the lower byte of the `tx_datain[15:0]` port is encoded as a control code because `tx_ctrlenable[0]` is high in the second clock cycle.

Figure 1–17. Encoded Control Word and Data Word Transmission



The 8B/10B encoder does not check to see if the code word entered is one of the 12 valid control code groups specified in the IEEE 802.3 8B/10B encoder specification. If an invalid control code is entered, the resulting 10-bit code may be encoded as an invalid code (it does not map to a valid Dx.y or Kx.y code), or unintended valid Dx.y code, depending on the value entered.

The following is an example of an invalid control word encoded into a valid Dx.y code. With an encoding invalid code K24.1 (`tx_datain` = 8'h38 + `tx_ctrl` = 1'b1), depending on the current running disparity, the K24.1 can be encoded as 10'b0110001100 (0 × 18C), which is equivalent to a D24.6+ (8'hD8 from the RD+ column). An 8B/10B decoder can decode this and not assert a code error flag.

Altera does not recommend sending invalid control words to the 8B/10B encoder.

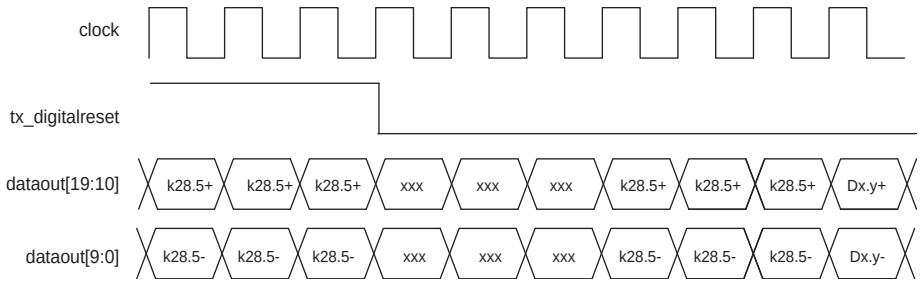
Reset Condition

The `tx_digitalreset` signal resets the 8B/10B encoder. During reset, the running disparity and data registers are cleared. Also, the 8B/10B encoder outputs a K28.5 pattern with proper disparity continuously until `tx_digitalreset` goes low. The inputs from the `tx_datain` and `tx_ctrlenable` ports are ignored during the reset state. After reset, the 8B/10B encoder starts the LSByte with a negative disparity (RD-) and the MSByte with a positive disparity (RD+) and transmits six K28.5 code groups (three on the LSByte and three on the MSByte encoder) for synchronizing before it starts encoding and transmitting data.

If the `tx_digitalreset` signal is asserted, the downstream 8B/10B decoder receiving the data might get synchronization or disparity errors.

Figure 1–18 shows the reset behavior of the 8B/10B encoder. When in reset (`tx_digitalreset` is high), a K28.5- on LSB and K28.5+ on MSB is sent continuously until `tx_digitalreset` is low. Due to pipelining of the TX channel, there will be some “don’t cares” (10'hxxx) until the first K28.5 is sent (Figure 1–18 shows six “don’t cares”, but the number of “don’t cares” can vary). Both the LSByte and MSByte transmit three K28.5s before the data at the `tx_datain` port is encoded and sent out.

Figure 1–18. Transmitted Output Data When tx_digitalreset is Asserted



Controlling Running Disparity

After power on or reset, the 8B/10B encoder has a negative disparity and chooses the 10-bit code from the RD- column (refer to the 8B/10B encoder specification for the RD+ and RD- column values). The ALTGX MegaWizard Plug-In Manager provides the `tx_forcedisp` and `tx_dispval` ports to control the running disparity of the output from the 8B/10B encoder. These ports are available only in Basic single-width and Basic double-width modes.

A high value on the `tx_forcedisp` port is the control signal to the disparity value of the output data. The disparity value (RD+ or RD-) is indicated by the value on the `tx_dispval` port. If the `tx_forcedisp` port is low, `tx_dispval` is ignored and the current running disparity is not altered. Forcing disparity can either maintain the current running disparity calculations if the forced disparity value (on the `tx_dispval` bit) matches the current running disparity, or flip the current running disparity calculations if it does not. If the forced disparity flips the current running disparity, the downstream 8B/10B decoder might detect a disparity error. Table 1–11 shows the `tx_forcedisp` and `tx_dispval` port values.

Table 1–11. tx_forcedisp and tx_dispval Port Values

tx_forcedisp	tx_dispval	Disparity Value
0	X	Current running disparity has no change
1	0	Encoded data has positive disparity
1	1	Encoded data has negative disparity

Figure 1–19 shows the current running disparity being altered in Basic single-width mode by forcing a positive disparity K28.5 when it was supposed to be a negative disparity K28.5. In this example, a series of K28.5 code groups are continuously being sent. The stream alternates between a positive running disparity (RD+) K28.5 and a negative running disparity (RD-) K28.5 to maintain a neutral overall disparity. The current running disparity at time n + 3 indicates that the K28.5 in time n + 4 should be encoded with a negative disparity. Because `tx_forcedisp` is high at time n + 4, and `tx_dispval` is low, the K28.5 at time n + 4 is encoded as a positive disparity code group.

Figure 1–19. 8B/10B Encoder Force Running Disparity Operation in Single-Width Mode

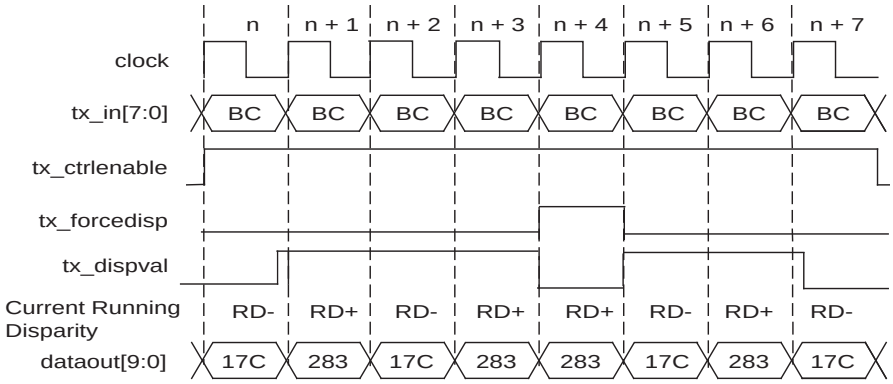
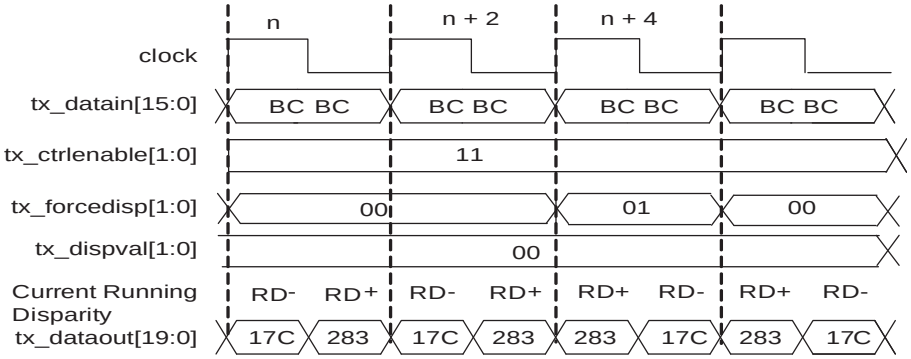


Figure 1–20 shows the current running disparity being altered in Basic double-width mode by forcing a positive disparity on a negative disparity K28.5. In this example, a series of K28.5 are continuously being sent. The stream alternates between a positive ending running disparity (RD+) K28.5 and a negative ending running disparity (RD-) K28.5 as governed by the 8B/10B encoder specification to maintain a neutral overall disparity. The current running disparity at the end of time n + 2 indicates that the K28.5 at the low byte position in time n + 4 should be encoded with a positive disparity. Because `tx_forcedisp` is high at time n + 4, the low signal level of `tx_dispval` is used to convert the lower byte K28.5 to be encoded as a positive disparity code word. As the upper bit of `tx_forcedisp` is low at n + 4, the high byte K28.5 takes the current running disparity from the low byte.

Figure 1–20. 8B/10B Encoder Force Current Running Disparity in Double-Width Mode



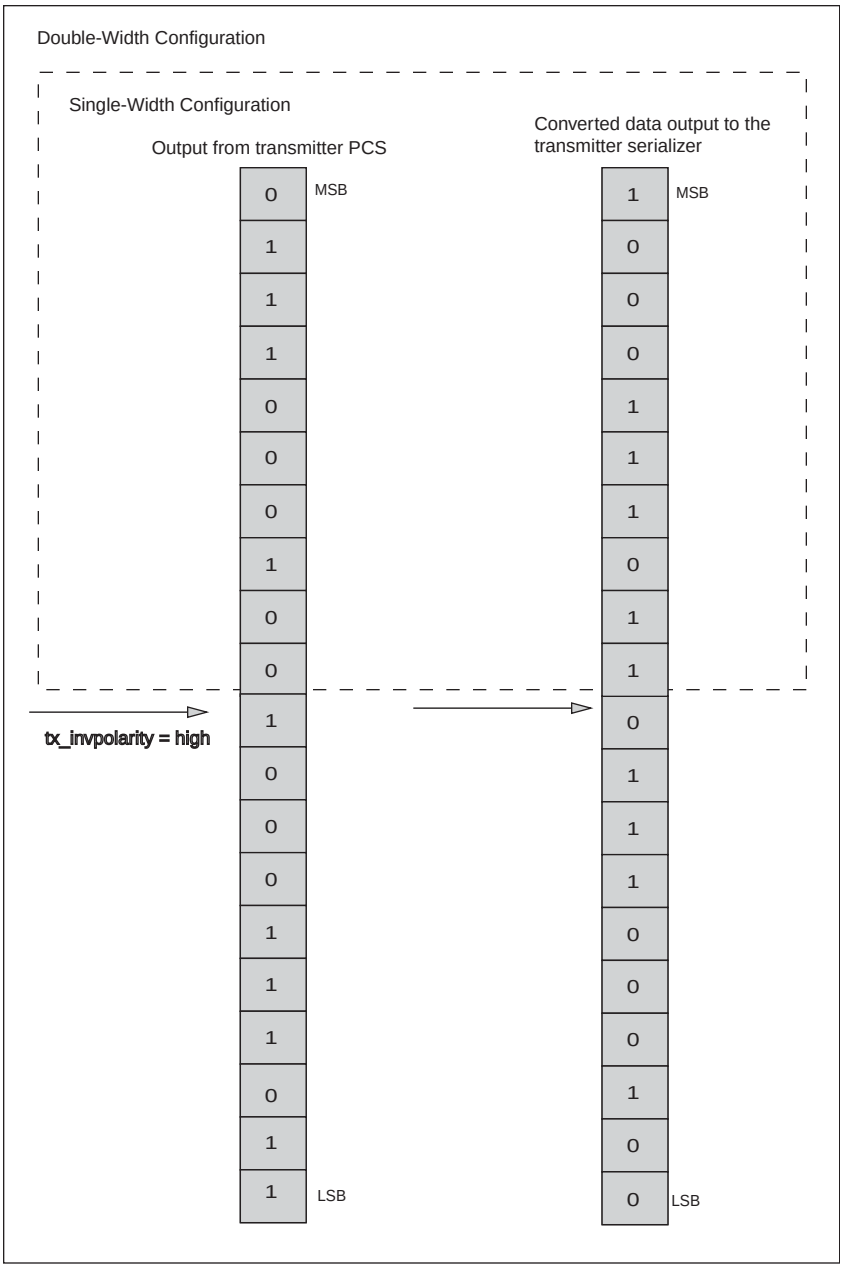
Transmitter Polarity Inversion

The positive and negative signals of a serial differential link might accidentally be swapped during board layout. Solutions like a board re-spin or major updates to the logic in the FPGA fabric can be expensive. The transmitter polarity inversion feature is provided to correct this situation. An optional `tx_invpolarity` port is available in all functional modes except (OIF) CEI PHY to dynamically enable the transmitter polarity inversion feature.

In single-width mode, a high value on the `tx_invpolarity` port inverts the polarity of every bit of the 8-bit or 10-bit input data word to the serializer in the transmitter datapath. In double-width mode, a high value on the `tx_invpolarity` port inverts the polarity of every bit of the 16-bit or 20-bit input data word to the serializer in the transmitter datapath. Because inverting the polarity of each bit has the same effect as swapping the positive and negative signals of the differential link, correct data is seen by the receiver. `tx_invpolarity` is a dynamic signal and might cause initial disparity errors at the receiver of an 8B/10B encoded link. The downstream system must be able to tolerate these disparity errors.

Figure 1–21 shows the transmitter polarity inversion feature in a single-width and double-width datapath configuration.

Figure 1–21. Transmitter Polarity Inversion in Single-Width and Double-Width Mode



Transmitter Bit Reversal

Table 1–12 shows the transmission bit order with and without the transmitter bit reversal enabled.

Table 1–12. Transmission Bit Order for the Bit Reversal Feature

Transmitter Bit Reversal Feature	Single-Width Mode (8- or 10-Bit)	Double-Width Mode (16- or 20-Bit)
Not enabled (default)	LSB to MSB	LSB to MSB
Enabled	MSB to LSB For example: ■ 8-bit—`D[7:0]` rewired to `D[0:7]` ■ 10-bit— `D[9:0]` rewired to `D[0:9]`	MSB to LSB For example: ■ 16-bit—`D[15:0]` rewired to `D[0:15]` ■ 20-bit—`D[19:0]` rewired to `D[0:19]`

Figure 1–22 shows the transmitter bit reversal feature in Basic single-width for a 10-bit wide datapath configuration.

Figure 1–22. Transmitter Bit Reversal Operation in Basic Single-Width Mode

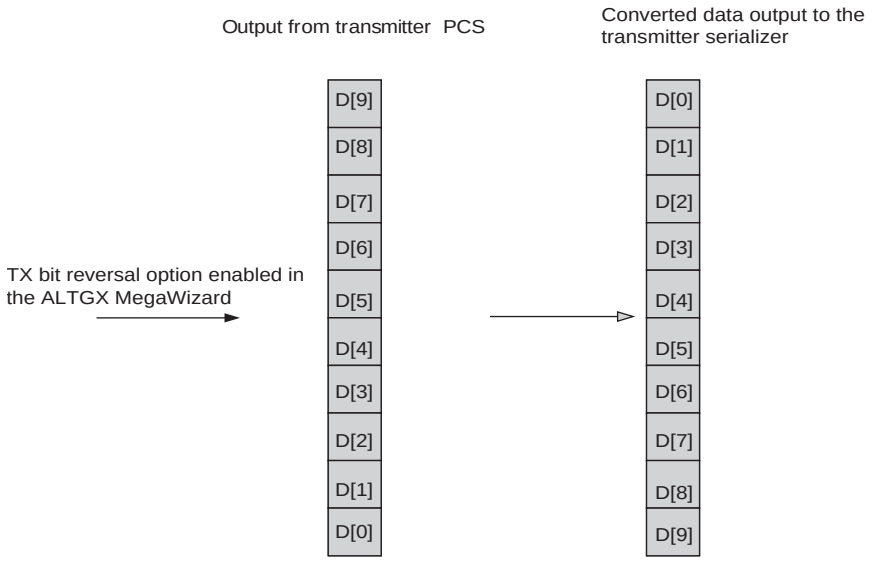
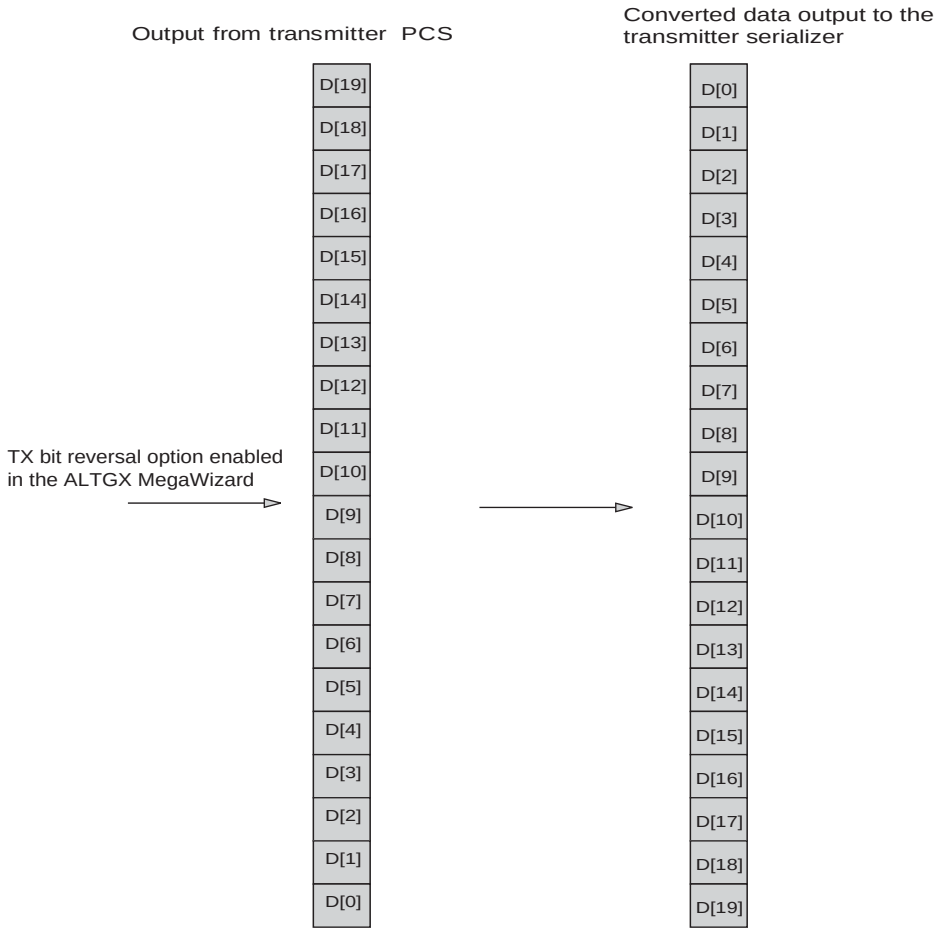


Figure 1–23 shows the transmitter bit reversal feature in Basic double-width mode for a 20-bit wide datapath configuration.

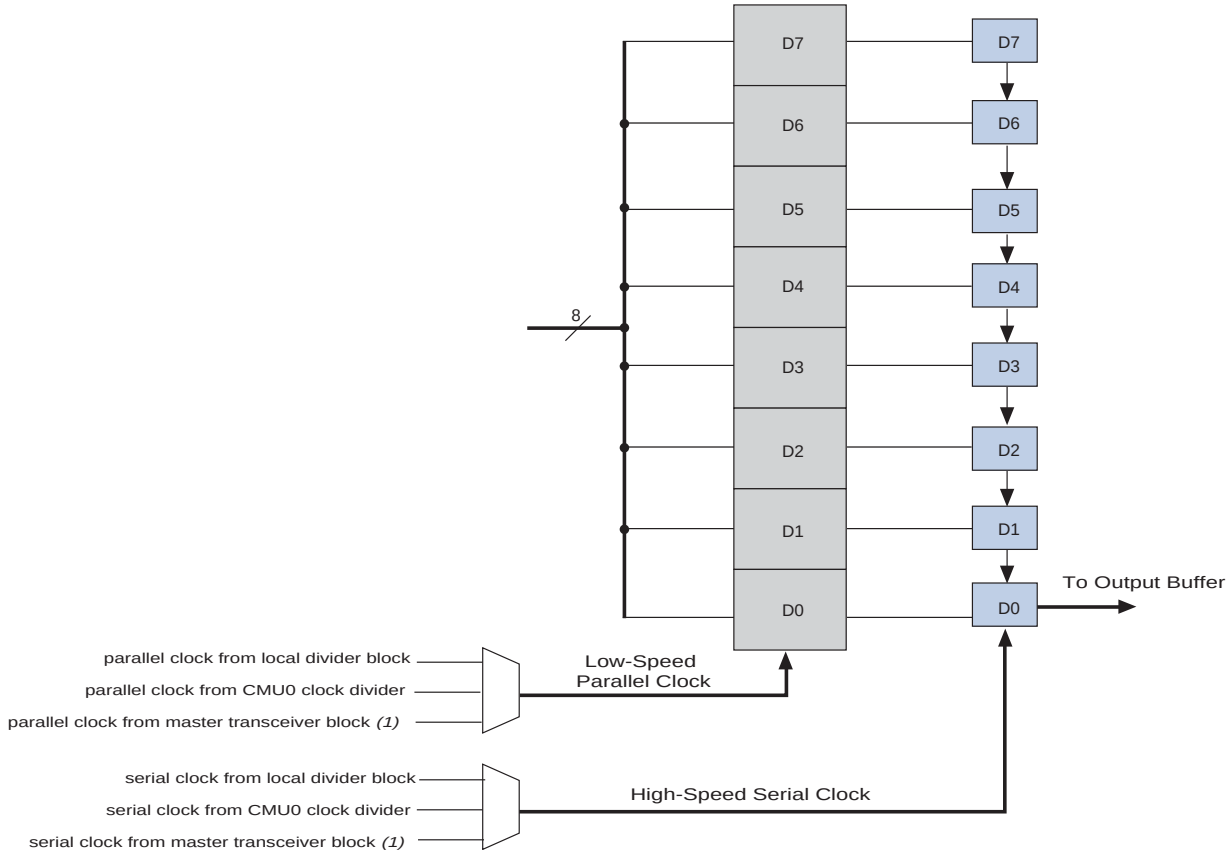
Figure 1–23. Transmitter Bit Reversal Operation in Basic Double-Width Mode



Serializer

The serializer converts the incoming low-speed parallel signal from the transceiver PCS to high-speed serial data and sends it to the transmitter buffer. The serializer supports an 8-bit or 10-bit serialization factor in single-width mode and a 16-bit or 20-bit serialization factor in double-width mode. The serializer block drives the serial data to the output buffer, as shown in Figure 1–24. The serializer block sends out the LSB of the input data.

Figure 1–24. Serializer Block in 8-Bit PCS-PMA Interface

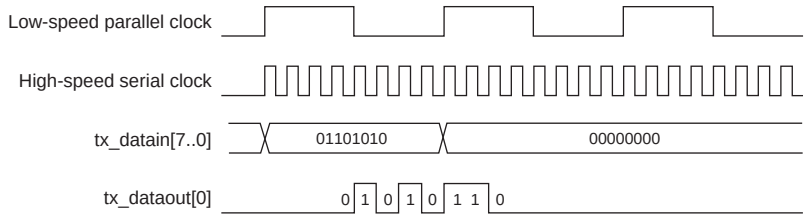


Note to Figure 1–24:

(1) The CMU0 clock divider of the master transceiver block provides the clocks. It is used only in bonded modes (for example, Basic ×8, PCI Express [PIPE] ×8 mode).

Figure 1–25 shows the serial bit order of the serializer block output. In this example, a constant 8'h6A (01101010) value is serialized and the serial data is transmitted from LSB to MSB.

Figure 1–25. Serializer Bit Order *(Note 1)*



Note to Figure 1–25:

(1) It is assumed that the input data to the serializer is 8 bits (channel width = 8 bits or 16 bits with the 8B/10B encoder disabled).

Transmitter Output Buffer

The Stratix IV GX and GT transmitter buffers are architecturally similar to each other. They both support programmable output differential voltage (V_{OD}), pre-emphasis, and on-chip termination (OCT) settings.

The transmitter buffer power supply only provides voltage to the transmitter output buffers in the transceiver channels. The transmitter output buffer, shown in Figure 1–26, has additional circuitry to improve signal integrity, such as V_{OD}, programmable three-tap pre-emphasis circuit, internal termination circuitry, and receiver detect capability to support PCI Express (PIPE) functional mode.

Figure 1–26. Transmitter Output Buffer

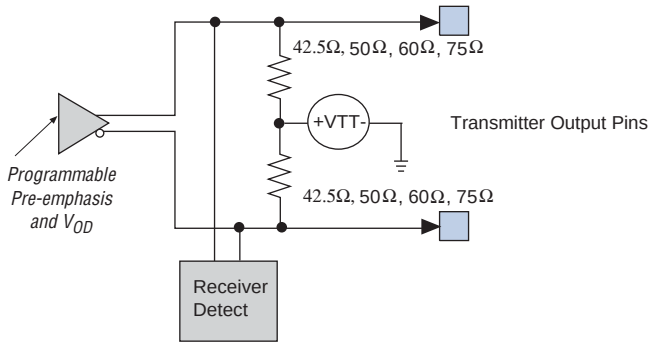


Table 1–13 and Table 1–14 list the supported settings of the transmitter buffers in the Stratix IV GX and GT devices, respectively.

Table 1–13. Supported Settings for the Stratix IV GX and GT Transmitter Buffer

Parameter	Setting
Data rate	600 Mbps to 8.5 Gbps (1.4 V) 600 Mbps to 6.5 Gbps (1.5 V)
Transmitter buffer power ($V_{CCH_GXBL/Rn}$)	1.4 V or 1.5 V
Transmitter buffer I/O standard	1.4-V and 1.5-V pseudo current mode logic (PCML)
Transmitter buffer V_{CM}	0.65 V

Table 1–14. Supported Settings for the Stratix IV GT Transmitter Buffer

Parameter	Setting
Data rate	2.488 Gbps—11.3 Gbps
Transmitter buffer power ($V_{CCH_GXBL/Rn}$)	1.4 V
Transmitter buffer I/O standard	1.4-V PCML
Transmitter buffer V_{CM}	0.65 V

Programmable Transmitter Termination

The Stratix IV GX and GT transmitter buffers includes programmable on-chip differential termination of 85, 100, 120, or 150 Ω. The resistance is adjusted by the on-chip calibration circuit in the calibration block (for more information, refer to "Calibration Blocks" on page 1–199), which compensates for temperature, voltage, and process changes. The Stratix IV GX and GT transmitter buffers in the transceiver are current mode drivers. Therefore, the resultant V_{OD} is a function of the transmitter termination value. For more information about resultant V_{OD} values, refer to "Programmable Output Differential Voltage" on page 1–31.

You can disable OCT and use external termination. If you select external termination, the transmitter common mode is tri-stated. You can set the transmitter termination in the ALTGX MegaWizard Plug-In Manager.

You can also set the OCT through the Assignment Editor. Set the assignment shown in Table 1–15 to the transmitter serial output pin.

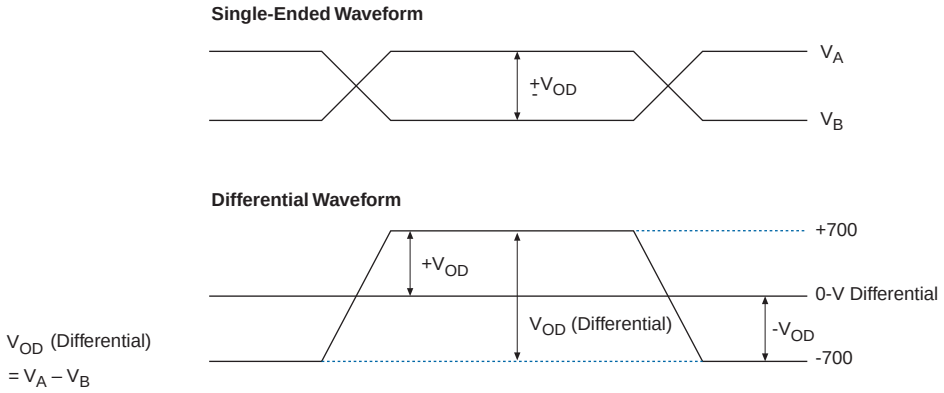
Table 1–15. Stratix IV GX and GT OCT Assignment Settings

Assign To	Transmitter Serial Output Data Pin
Assignment Name	Output termination
Available Values	OCT 85 Ω, OCT 100 Ω, OCT 120 Ω, OCT 150 Ω

Programmable Output Differential Voltage

The Stratix IV GX and GT devices allow you to customize the differential output voltage to handle different trace lengths, various backplanes, and receiver requirements, as shown in Figure 1–27. You can change the V_{OD} values using the dynamic reconfiguration controller. Set the V_{OD} value through the `tx_vodctrl[2:0]` port of the dynamic reconfiguration controller. For example, to set V_{OD} to a value of 3, set the `tx_vodctrl[2:0]` to **011**.

Figure 1–27. V_{OD} (Differential) Signal Level



For more information about Stratix IV GX and GT V_{OD} values, refer to the *DC and Switching Characteristics* chapter.

Programmable Pre-Emphasis

The programmable pre-emphasis module in each transmit buffer boosts high frequencies in the transmit data signal, which might be attenuated in the transmission media. Using pre-emphasis can maximize the data

opening at the far-end receiver. The transmission line's transfer function can be represented in the frequency domain as a low-pass filter. Any frequency components below –3dB can pass through with minimal loss. Frequency components greater than –3dB are attenuated. This variation in frequency response yields data-dependent jitter and other inter-symbol interference (ISI) effects. By applying pre-emphasis, the high-frequency components are boosted; that is, pre-emphasized. Pre-emphasis equalizes the frequency response at the receiver so the difference between the low-frequency and high-frequency components is reduced, which minimizes the ISI effects from the transmission medium. Pre-emphasis requirements increase as data rates through legacy backplanes increase. You set the pre-emphasis settings in the ALTGX MegaWizard Plug-In Manager. The Stratix IV GX and GT transceivers provide three pre-emphasis taps—pre tap, first post tap, and second post tap. The ALTGX MegaWizard Plug-In Manager provides options to select the different values on these three taps. The pre tap sets the pre-emphasis on the data bit before the transition. The first post tap and second post tap set the pre-emphasis on the transition bit and the successive bit, respectively. The pre tap and second post tap also provide inversion control, shown by negative values on the corresponding tap settings in the ALTGX MegaWizard Plug-In Manager. The ALTGX MegaWizard Plug-In Manager only shows the valid pre-emphasis tap values for a selected V_{OD} and transmitter termination resistance setting.

Programmable Transmitter Output Buffer Power (V_{CCH})

The ALTGX MegaWizard Plug-In Manager provides an option to select V_{CCH}. Table 1–16 lists the data rates for the two V_{CCH} options.

Table 1–16. V_{CCH} Option Data Rates

V_{CCH} Options	Stratix IV GX Data Rate	Stratix IV GT Data Rate
1.4 V	600 Mbps to 8.5 Gbps	2.488 Gbps to 11.3 Gbps
1.5 V	600 Mbps to 6.5 Gbps	—

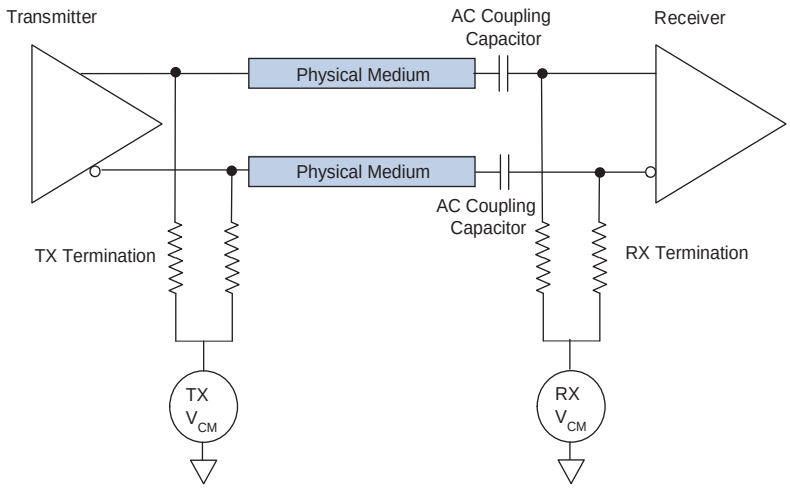
Link Coupling for Stratix IV GX and GT Devices

A high-speed serial link can be AC-coupled or DC-coupled, depending on the serial protocol being implemented.

AC-Coupled Links

In an AC-coupled link, the AC-coupling capacitor blocks the transmitter DC V_{CM}. The on-chip or off-chip receiver termination and biasing circuitry automatically restores the selected V_{CM}. Figure 1–28 shows an AC-coupled link.

Figure 1–28. AC-Coupled Link



The following protocols supported by Stratix IV GX and GT devices mandate AC-coupled links:

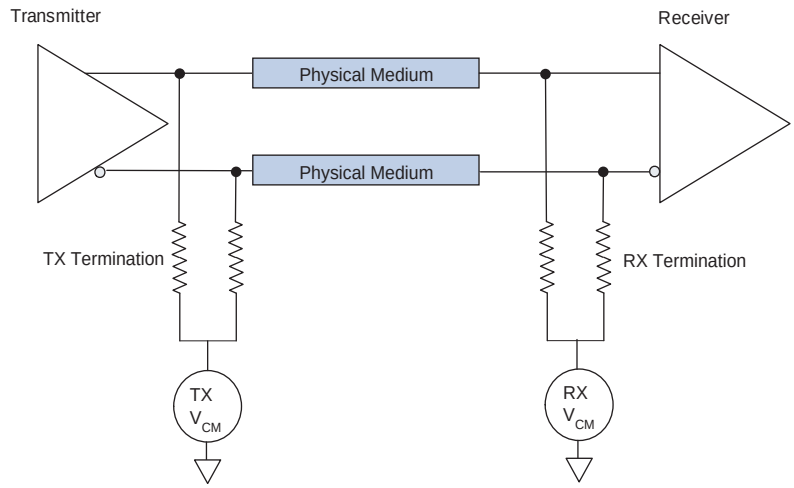
- PCI Express (PIPE)
- Gigabit Ethernet
- Serial RapidIO
- XAUI
- SDI

Stratix IV GT devices allow the high-speed links to be AC-coupled for the entire data rate range between 2.488 Gbps and 11.3 Gbps.

DC-Coupled Links

In a DC-coupled link, the transmitter DC V_{CM} is seen unblocked at the receiver buffer. The link V_{CM} depends on the transmitter V_{CM} and the receiver V_{CM}. The on-chip or off-chip receiver termination and biasing circuitry must ensure compatibility between the transmitter and receiver V_{CM}. Figure 1–29 shows a DC-coupled link.

Figure 1–29. DC-Coupled Link



The Stratix IV GX and GT transmitter can be DC-coupled to a Stratix IV GX and GT receiver for the entire operating data rate range of Stratix IV GX, from 600 Mbps to 8.5 Gbps.

The Stratix IV GT transmitter can be DC-coupled to the Stratix IV GT receiver for the entire data rate range of 2.488 Gbps to 11.3 Gbps with Tx Vcm = 0.65 V and Rx Vcm = 0.82 V.

For more information on the DC coupling capabilities of the Stratix IV GT device, refer to Table 1–23 on page 1–44.

PCI Express (PIPE) Receiver Detect

The Stratix IV GX and GT transmitter buffers have a built-in receiver detection circuit for use in the PCI Express (PIPE) mode for Gen1 and Gen2 data rates. This circuit detects if there is a receiver downstream by sending out a pulse on the common mode of the transmitter and monitoring the reflection. This mode requires the transmitter buffer to be tri-stated (in Electrical Idle mode), OCT utilization, and a 125 MHz `fixedclk` signal. You can enable this feature in PCI Express (PIPE) mode by setting the `tx_forceelecidle` and `tx_detectrxloopback` ports to **1'b1**. Receiver detect circuitry is active only in the P1 power state.

For more information about power states, refer to the PCI Express (PIPE) 2.0 specification.

In the P1 power state, the transmitter output buffer is tri-stated because the transmitter output buffer is in electrical idle. A high on the `tx_detectrxloopback` port triggers the receiver detect circuitry to alter the transmitter output buffer V_{CM}. The sudden change in V_{CM} effectively appears as a step voltage at the tri-stated transmitter buffer output. If a receiver (that complies with PCI Express [PIPE] input impedance requirements) is present at the far end, the time constant of the step voltage is higher. If a receiver is not present or is powered down, the time constant of the step voltage is lower. The receiver detect circuitry snoops the transmitter buffer output for the time constant of the step voltage to detect the presence of the receiver at

the far end. A high pulse is driven on the `pipephydonestatus` port and 3'b011 is driven on the `pipestatus` port to indicate that a receiver has been detected. There is some latency after asserting the `tx_detectrxloopback` signal, before the receiver detection is indicated on the `pipephydonestatus` port. For signal timing to perform the receiver detect operation, refer to Figure 1–105 on page 1–131.

The `tx_forceelecidle` port must be asserted at least 10 parallel clock cycles prior to the `tx_detectrxloopback` port to ensure that the transmitter buffer is tri-stated.

PCI Express (PIPE) Electrical Idle

The Stratix IV GX and GT transmitter output buffers support transmission of PCI Express (PIPE) Electrical Idle (or individual transmitter tri-state). The `tx_forceelecidle` port puts the transmitter buffer in Electrical Idle mode. This port has a specific functionality in each power state. For the signal timing to perform the electrical idle transmission in PCI Express (PIPE) mode, refer to Figure 1–104 on page 1–130.

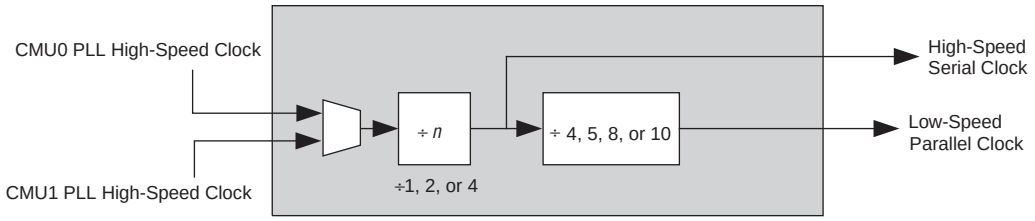
For use of the `tx_forceelecidle` signal under different power states, refer to the PCI Express (PIPE) specification 2.0.

Transmitter Local Clock Divider Block

Each transmitter channel contains a local clock divider block. It receives the high-speed clock from the `CMU0` PLL or `CMU1` PLL and generates the high-speed serial clock for the serializer and the low-speed parallel clock for the transmitter PCS datapath. The low-speed parallel clock is also forwarded to the FPGA fabric (`tx_clkout`). The local clock divider block allows each transmitter channel to run at /1, /2, or /4 of the CMU PLL data rate. The local clock divider block is used only in non-bonded functional modes (for example, GIGE, SONET/SDH, and SDI mode).

Figure 1–30 shows the transmitter local clock divider block.

Figure 1–30. Transmitter Local Clock Divider Block



Receiver Channel Datapath

This section describes the Stratix IV GX and GT receiver channel datapath architecture. The sub-blocks in the receiver datapath are described in order from the serial receiver input buffer to the receiver phase compensation FIFO buffer at the FPGA fabric-transceiver interface. Figure 1–7 on page 1–12 shows the receiver channel datapath in Stratix IV GX and GT devices.

The receiver channel PMA datapath consists of the following blocks:

- Receiver input buffer
- Clock and data recovery (CDR) unit
- Deserializer

The receiver channel PCS datapath consists of the following blocks:

- Word aligner
- Deskew FIFO
- Rate match (clock rate compensation) FIFO
- 8B/10B decoder
- Byte deserializer
- Byte ordering
- Receiver phase compensation FIFO
- PCI Express (PIPE) interface

The receiver datapath is very flexible and allows multiple configurations, depending on the selected functional mode. You can configure the receiver datapath using the ALTGX MegaWizard Plug-In Manager.

Receiver Input Buffer

The Stratix IV GX and GT receiver input buffers are architecturally similar to each other. They both support programmable common mode voltage (Rx VCM), equalization, DC gain, and on-chip termination (OCT) settings. Table 1–17 lists the supported settings of the receiver input buffers in Stratix IV GX and GT devices.

The receiver input buffer receives serial data from the `rx_datain` port and feeds it to the CDR unit. In the reverse serial loopback (pre-CDR) configuration, it also feeds the received serial data to the transmitter output buffer. Figure 1–31 shows the receiver input buffer.

Figure 1–31. Receiver Input Buffer

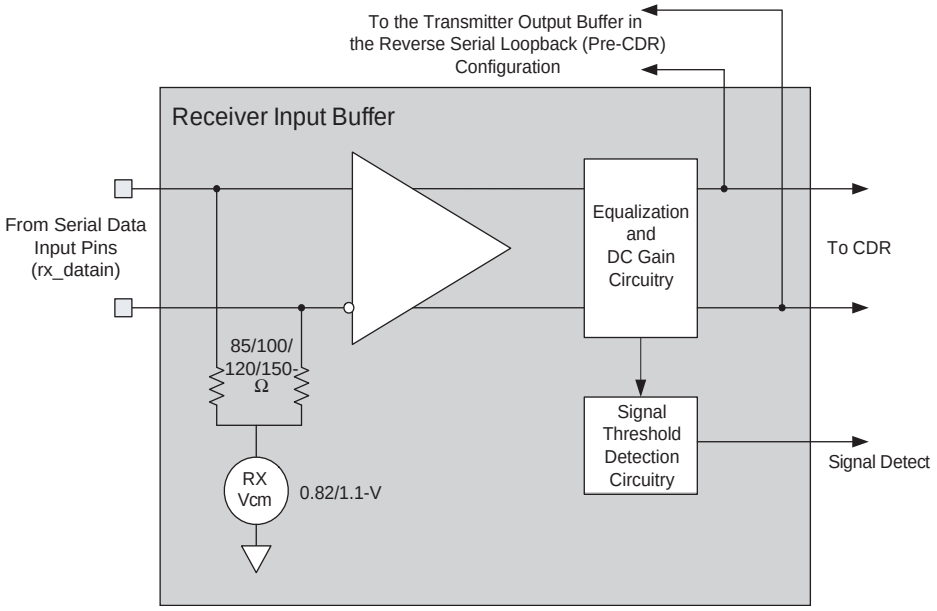


Table 1–17 shows the electrical features supported by the Stratix IV GX and GT receiver input buffer.

Table 1–17. Electrical Features Supported by the Receiver Input Buffer for Stratix IV GX and GT Devices *(Note 1)*

Data Rate (Gbps)	I/O Standard	Differential OCT with Calibration (Ω)	V_{CM} (V)	Coupling	Programmable DC Gain (dB)
Stratix IV GX 0.6 to 8.5	1.4 V PCML	85, 100, 120, 150	0.82	AC, DC	up to 16
	1.5 V PCML	85, 100, 120, 150	0.82	AC, DC	up to 16
	2.5 V PCML	85, 100, 120, 150	0.82	AC	up to 16
	LVPECL	85, 100, 120, 150	0.82	AC	up to 16
	LVDS	85, 100, 120, 150	1.1	AC, DC	up to 16
Stratix IV GT 2.488 to 11.3	1.4 V PCML	85, 100, 120, 150	0.82	AC, DC	0, 3, 6, 9, and 12
	LVDS	85, 100, 120, 150	1.1	AC, DC	0, 3, 6, 9, and 12

Note to Table 1–17:

(1) Programmable equalization settings are 0 to 16 dB for Stratix IV GX and GT devices.

The Stratix IV GX and GT receiver buffers support the following features:

- Programmable differential OCT
- Programmable V_{CM}
- AC and DC coupling
- Programmable equalization and DC gain
- Signal threshold detection circuitry

Programmable Differential On-Chip Termination

The Stratix IV GX and GT receiver buffers support optional differential OCT resistors of 85, 100, 120, and 150 Ω. To select the desired receiver OCT resistor, make the assignments shown in Table 1–18 in the Quartus II software Assignment Editor.

Table 1–18. Stratix IV GX and GT Receiver On-Chip Termination Assignment Settings

Assign To	rx_datain (Receiver Input Data Pins)
Assignment Name	Input Termination
Stratix IV GX Available Values	OCT 85 Ω, OCT 100 Ω, OCT 120 Ω, OCT 150 Ω, Off
Stratix IV GT Available Values	OCT 85 Ω ,OCT 100 Ω, OCT 120 Ω, OCT 150 Ω

The Stratix IV GX and GT receiver OCT resistors have calibration support to compensate for process, voltage, and temperature variations. For more information about OCT calibration support, refer to "Calibration Blocks" on page 1–199.

Programmable V_{CM}

The Stratix IV GX and GT receiver buffers have on-chip biasing circuitry to establish the required V_{CM} at the receiver input. It supports V_{CM} settings of 0.82 V and 1.1 V that you can select in the ALTGX MegaWizard Plug-In Manager.

You must select **0.82 V** as the receiver buffer V_{CM} for the following receiver input buffer I/O standards:

- 1.4-V PCML
- 1.5-V PCML
- 2.5-V PCML
- LVPECL

You must select **1.1 V** as the receiver buffer V_{CM} for the LVDS receiver input buffer I/O standard.

On-chip biasing circuitry is effective only if you select **on-chip receiver termination**. If you select **external termination**, you must implement off-chip biasing circuitry to establish the V_{CM} at the receiver input buffer.

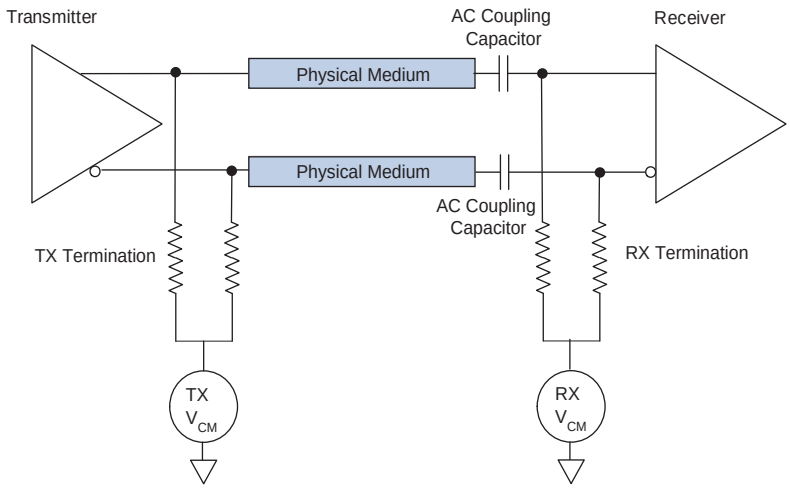
Link Coupling for Stratix IV GX Devices

A high-speed serial link can either be AC-coupled or DC-coupled, depending on the serial protocol being implemented. Most of the serial protocols require links to be AC-coupled, but protocols such as Common Electrical I/O (CEI) optionally allow DC coupling.

AC-Coupled Links

In an AC-coupled link, the AC coupling capacitor blocks the transmitter DC V_{CM}. The on-chip or off-chip receiver termination and biasing circuitry automatically restores the selected V_{CM}. Figure 1–32 shows an AC-coupled link.

Figure 1–32. AC-Coupled Link



Note to Figure 1–32:

(1) The receiver termination and biasing can be on-chip or off-chip.

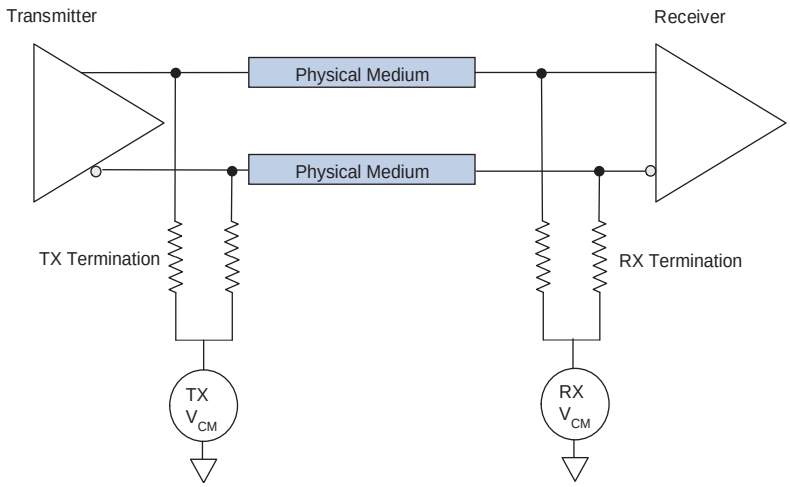
The following protocols supported by Stratix IV GX and GT devices mandate AC-coupled links:

- PCI Express (PIPE)
- Gigabit Ethernet
- Serial RapidIO
- XAUI
- SDI

DC-Coupled Links

In a DC-coupled link, the transmitter DC V_{CM} is seen unblocked at the receiver buffer. Link V_{CM} depends on the transmitter V_{CM} and the receiver V_{CM}. The on-chip or off-chip receiver termination and biasing circuitry must ensure compatibility between the transmitter and the receiver V_{CM}. Figure 1–33 shows a DC-coupled link.

Figure 1–33. DC-Coupled Link



Note to Figure 1–33:

(1) The receiver termination and biasing can be on-chip or off-chip.

You might choose to use the DC-coupled high-speed link for these functional modes only:

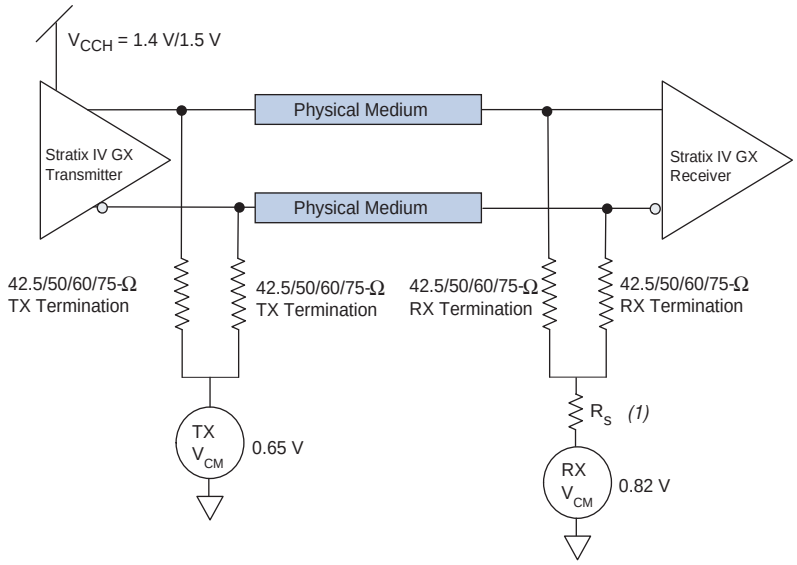
- Basic single- and double-width
- (OIF) CEI PHY interface

The following sections describe DC-coupling requirements for a high-speed link with a Stratix IV GX device used as the transmitter, receiver, or both. Specifically, the following link configurations are described:

- Stratix IV GX Transmitter (PCML) to Stratix IV GX Receiver (PCML)
- Stratix II GX Transmitter (PCML) to Stratix IV GX Receiver (PCML)
- Stratix IV GX Transmitter (PCML) to Stratix II GX Receiver (PCML)
- LVDS Transmitter to Stratix IV GX Receiver (PCML)

Figure 1–34 shows a typical Stratix IV GX transmitter (PCML) to Stratix IV GX Receiver (PCML) DC-coupled link.

Figure 1–34. Stratix IV GX Transmitter (PCML) to Stratix IV GX Receiver (PCML)



Note to Figure 1–34:

(1) R_S is the parasitic resistance present in the on-chip RX termination and biasing circuitry.

Table 1–19 shows the allowed transmitter and receiver settings in a Stratix IV GX transmitter (PCML) to Stratix IV GX receiver (PCML) DC-coupled link.

Table 1–19. Settings for a Stratix IV GX Transmitter (PCML) to Stratix IV GX Receiver (PCML) DC-Coupled Link

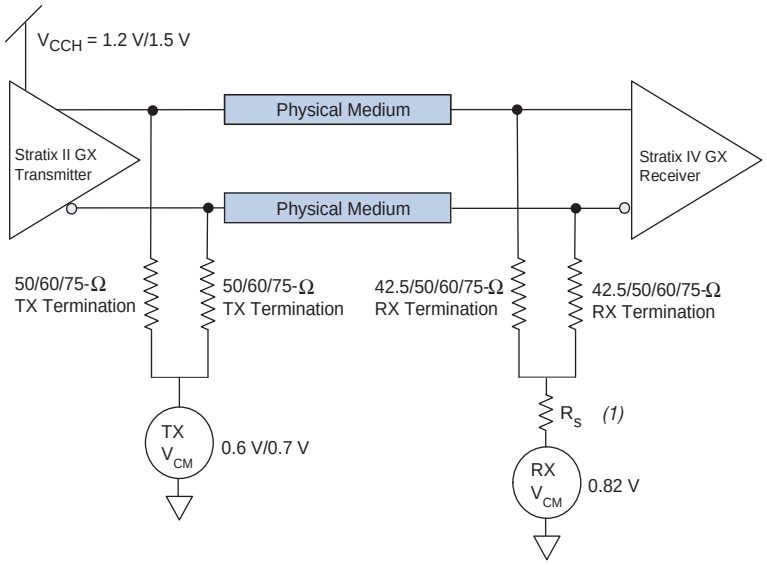
Transmitter (Stratix IV GX) Settings				Receiver (Stratix IV GX) Settings		
Data Rate	**V_{CCH} *(1)***	**TX V_{CM}**	**Differential Termination**	**Data Rate**	**RX V_{CM}**	**Differential Termination**
600-8500 Mbps	1.4 V/1.5 V	0.65 V	85/100/120/150 Ω	600-8500 Mbps	0.82 V	85/100/120/150 Ω

Note to Table 1–19:

(1) V_{CCH} = 1.5 V can support data rates from 600 Mbps to 6.5G Mbps. V_{CCH} = 1.4 V can support data rates from 600 Mbps to 8500 Mbps.

Figure 1–35 shows the Stratix II GX transmitter (PCML) to Stratix IV GX receiver (PCML) coupled link.

Figure 1–35. Stratix II GX Transmitter (PCML) to Stratix IV GX Receiver (PCML)



Note to Figure 1–35:

(1) R_S is the parasitic resistance present in the on-chip RX termination and biasing circuitry.

Table 1–20 shows the allowed transmitter and receiver settings in a Stratix II GX to Stratix IV GX DC-coupled link.

Table 1–20. Settings for a Stratix II GX to Stratix IV GX DC-Coupled Link

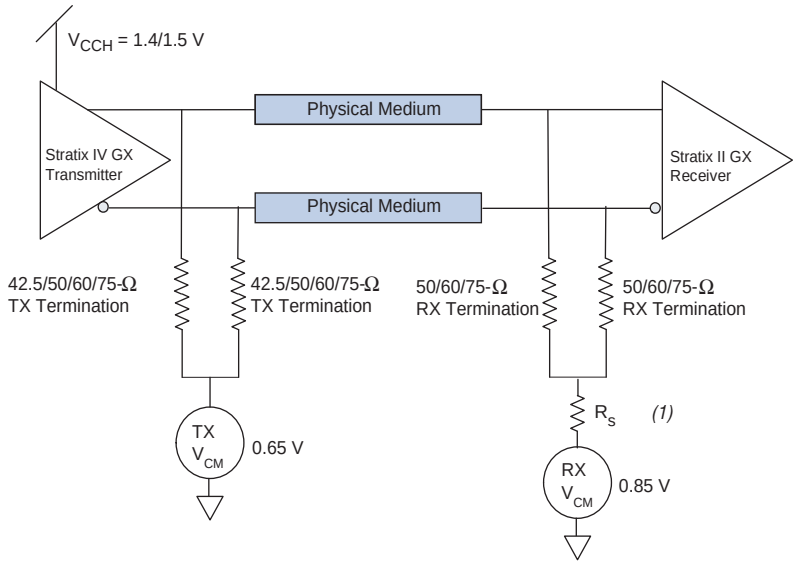
Transmitter (Stratix II GX) Settings				Receiver (Stratix IV GX) Settings		
Data Rate	**V_{CCH} *(1)***	**TX V_{CM} *(1)***	**Differential Termination**	**Data Rate**	**RX V_{CM}**	**Differential Termination**
600-6375 Mbps	1.5 V (1.5 V PCML)	0.6 V/0.7 V	100/120/150 Ω	600-6375 Mbps	0.82 V	100/120/150 Ω

Note to Table 1–20:

(1) V_{CCH} = 1.5 V with TX V_{CM} = 0.7 V can support data rates from 600 Mbps to 3125 Mbps. V_{CCH} = 1.5 V with TX V_{CM} = 0.6 V can support data rates from 600 Mbps to 6375 Mbps.

Figure 1–36 shows the Stratix IV GX transmitter (PCML) to Stratix II GX receiver (PCML) DC-coupled link.

Figure 1–36. Stratix IV GX Transmitter (PCML) to Stratix II GX Receiver (PCML)



Note to Figure 1–36:

(1) R_S is the parasitic resistance present in the on-chip RX termination and biasing circuitry.

Table 1–21 shows the allowed transmitter and receiver settings in a Stratix IV GX transmitter (PCML) to Stratix II GX receiver (PCML) DC-coupled link.

Table 1–21. Settings for a Stratix IV GX to Stratix II GX DC-Coupled Link

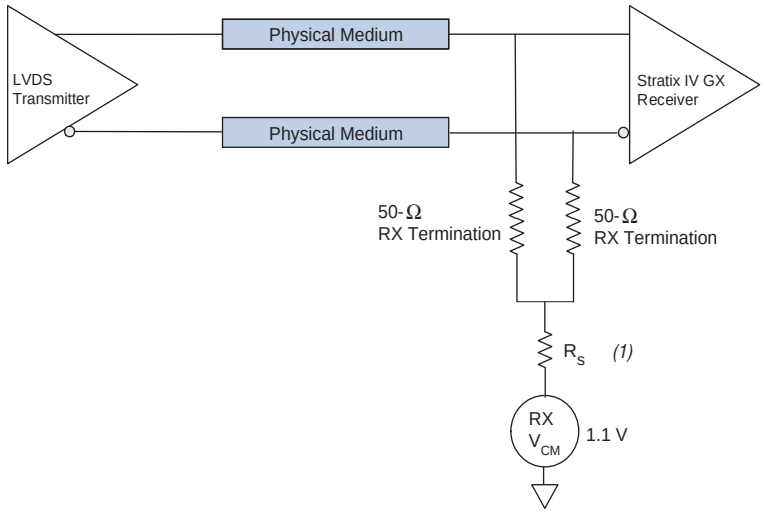
Transmitter (Stratix IV GX) Settings				Receiver (Stratix II GX) Settings			
Data Rate	**V_{CCH} *(1)***	**TX V_{CM}**	**Differential Termination**	**Data Rate**	**I/O Standard**	**RX V_{CM}**	**Differential Termination**
600-6375 Mbps	1.4/1.5 V	0.65 V	100/120/150 Ω	600-6375 Mbps	1.4/1.5 V PCML	0.85 V	100/120/150 Ω

Note to Table 1–21:

(1) V_{CCH} = 1.5 V can support data rates from 600 Mbps to 6.5G Mbps. V_{CCH} = 1.4 V can support data rates from 600 Mbps to 6375 Mbps.

Figure 1–37 shows the LVDS transmitter to Stratix IV GX receiver (PCML) DC-coupled link.

Figure 1–37. LVDS Transmitter to Stratix IV GX Receiver (PCML)



Note to Figure 1–37:

(1) R_S is the parasitic resistance present in the on-chip RX termination and biasing circuitry.

Table 1–22 shows the allowed transmitter and receiver settings in a LVDS transmitter to Stratix IV GX receiver DC-coupled link.

Table 1–22. Settings for a LVDS transmitter to Stratix IV GX Receiver DC-Coupled Link *(Note 1)*

Receiver (Stratix IV GX) Settings		
RX V_{CM}	**Differential Termination**	**R_S**
1.1 V	100 Ω	*(2)*

Notes to Table 1–22:

(1) When DC-coupling an LVDS transmitter to the Stratix IV GX receiver, use RX V_{CM} = 1.1 V and series resistance value RS to verify compliance with the LVDS specification.

(2) Pending characterization.

Link Coupling for Stratix IV GT Devices

Stratix IV GT devices allow the high-speed links to be AC- or DC-coupled links (AC-coupling allowed for the entire data rate range between 2.488 Gbps and 11.3 Gbps).

Table 1–23 lists the allowed DC-coupling scenarios for Stratix IV GT devices.

Table 1–23. Allowed DC-Coupling Scenarios for Stratix IV GT Devices (Part 1 of 2)

From (Transmitter I/O Standard)	To (Receiver I/O Standard)	Data Rate Range	Conditions
Stratix IV GT Transmitter (1.4-V PCML)	Stratix IV GT Receiver (1.4-V PCML)	2.488 Gbps to 11.3 Gbps	■ TX V_{CM} = 0.65 V ■ RX V_{CM} = 0.82 V
Stratix IV GX Transmitter (1.4-V PCML)	Stratix IV GT Receiver (1.4-V PCML)	2.488 Gbps to 8.5 Gbps	■ TX V_{CM} = 0.65 V ■ RX V_{CM} = 0.82 V

Table 1–23. Allowed DC-Coupling Scenarios for Stratix IV GT Devices (Part 2 of 2)

From (Transmitter I/O Standard)	To (Receiver I/O Standard)	Data Rate Range	Conditions
Stratix II GX Transmitter (1.5-V PCML)	Stratix IV GT Receiver (1.4-V PCML)	2.488 Gbps to 6.375 Gbps	■ TX V_{CM} = 0.7 V (2.488 Gbps to 3.125 Gbps) ■ TX V_{CM} = 0.6 V (3.125 Gbps to 6.375 Gbps) ■ RX V_{CM} = 0.82 V
Third-Party LVDS Transmitter	Stratix IV GT Receiver (LVDS)	2.488 Gbps to 6.5 Gbps	■ RX V_{CM} = 1.1 V

Programmable Equalization and DC Gain

The transfer function of the physical medium can be represented as a low-pass filter in the frequency domain. Frequency components below –3 dB frequency pass through with minimal loss. Frequency components greater than –3 dB frequency are attenuated as a function of frequency due to skin-effect and dielectric losses. This variation in frequency response yields data-dependent jitter and other ISI effects, which can cause incorrect sampling of the input data.

Each Stratix IV GX and GT receiver buffer has independently programmable equalization circuitry that boosts the high-frequency gain of the incoming signal, thereby compensating for the low-pass filter effects of the physical medium. The amount of high-frequency gain required depends on the loss characteristics of the physical medium. Stratix IV GX and GT equalization circuitry supports 16 equalization settings that provide up to 16 dB of high-frequency boost. You can select the appropriate equalization setting in the ALTGX MegaWizard Plug-In Manager.

Stratix IV GX and GT receiver buffers also support programmable DC gain circuitry. Unlike equalization circuitry, DC gain circuitry provides equal boost to the incoming signal across the frequency spectrum. The receiver buffer supports DC gain settings of 0, 3, 6, 9, and 12 dB. You can select the appropriate DC gain setting in the ALTGX MegaWizard Plug-In Manager.

Signal Threshold Detection Circuitry

In PCI Express (PIPE) mode, you can enable the optional signal threshold detection circuitry by not selecting the **Force signal detection** option in the ALTGX MegaWizard Plug-In Manager. If enabled, this option senses whether the signal level present at the receiver input buffer is above the signal detect threshold voltage that you specified in the **What is the signal detect and signal loss threshold?** option in the ALTGX MegaWizard Plug-In Manager.

☞ The appropriate signal detect threshold level that complies with the PCI Express (PIPE) compliance parameter VRX-IDLE-DETDIFFp-p is available in the *DC and Switching Characteristics* chapter.

Signal threshold detection circuitry has a hysteresis response that filters out any high-frequency ringing caused by inter-symbol interference or high-frequency losses in the transmission medium. If the signal threshold detection circuitry senses the signal level present at the receiver input buffer to be higher than the signal detect threshold, it asserts the `rx_signaldetect` signal high. Otherwise, the signal threshold detection circuitry de-asserts the `rx_signaldetect` signal low.

Adaptive Equalization (AEQ)

Stratix IV GX and GT receivers offer an Adaptive Equalization feature that automatically compensates for losses on the receiver channels. High-speed interface systems are used at different data rates with multiple backplane environments. These systems require different equalization setting to compensate for changing data rates and back plane characteristics. Manually selecting optimal equalization settings is cumbersome under these changing system characteristics. The Adaptive Equalization feature solves this problem by enabling the Stratix IV device to continuously tune the receiver equalization settings based on the frequency content of the incoming signal and comparing it with internally generated reference signals.

Without this feature, you would have to tune the receiver channel's equalization stages manually, finding the optimal settings through trial and error, then locking in those values at compile time.

The AEQ block resides within the PMA of the receiver channel and is available on the four regular channels of a transceiver block. To use AEQ, you must first enable the AEQ hardware in the ALTGX MegaWizard Plug-In Manager and the AEQ control block in the ALTGX_RECONFIG MegaWizard Plug-In Manager.

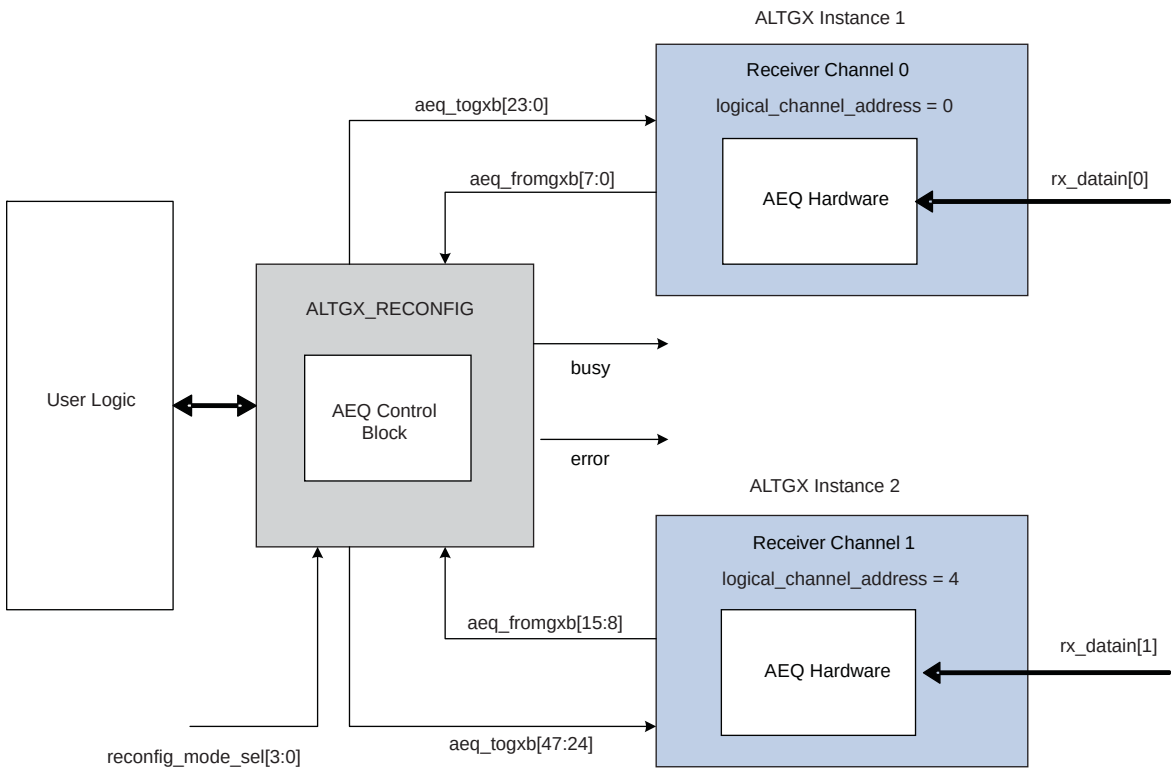
In ALTGX and ALTGX_RECONFIG MegaWizard Plug-In Managers, select the **Enable adaptive equalizer control** option to enable the AEQ.

When you select AEQ, two ports, `aeq_fromgxb[]` and `aeq_togxb[]` become available on the ALTGX and ALTGX_RECONFIG instances. These ports provide an interface between the PMA of the receiver channel and the AEQ control block in the ALTGX_RECONFIG.

☞ AEQ hardware is not present in the CMU channels.

Figure 1–38 shows the receiver channel data path with the AEQ feature.

Figure 1–38. Receiver Channel Data Path Showing AEQ



Modes of Operation of the AEQ

Depending on the value you set for `reconfig_mode_sel[3:0]`, the AEQ has three modes of operation:

- Continuous mode—The AEQ continuously monitors the frequency content of the received signal and adapts to it by providing dynamically changing equalizer settings to the Stratix IV GX and GT receiver. This mode is available on one channel or all channels of the receiver. The `reconfig_mode_sel[3:0]` = 1000 in this mode.
- One-time mode—The AEQ finds a stable setting of the receiver equalizer and locks that value. Once locked, the equalizer values are no longer changed. This mode is available in one channel or all channels of the receiver. The `reconfig_mode_sel[3:0]` = 1001 in this mode.
- Powerdown mode—In this mode, the AEQ of the specific channel is placed in standby mode. This mode is available in one channel or all channels of the receiver. The `reconfig_mode_sel[3:0]` = 1010 in this mode.

You are allowed to switch between these modes dynamically.

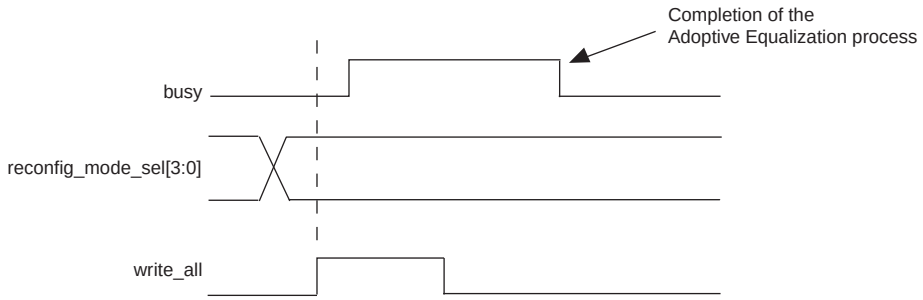
The AEQ comes out of standby mode as soon as the value on `reconfig_mode_sel` is changed to another mode of the AEQ control modes. Once the AEQ goes into powerdown mode and comes out, it does not remember the converged equalization value. By design, the AEQ starts at the maximum equalization value after powering up again.

To control the AEQ hardware in any of the above modes, follow these steps:

1. Watch for `busy` to be low, then write the appropriate value on `reconfig_mode_sel[3:0]`.
2. Assert `write_all`, then watch for `busy` to be asserted.
3. De-assert `write_all` and watch for the de-assertion of `busy`. De-assertion of `busy` indicates that Adaptive Equalization has ended.

Figure 1–39 shows a waveform the AEQ process.

Figure 1–39. AEQ Process Waveform



For more information about the AEQ port connections and various waveforms in all the above modes, refer to the *Stratix IV Dynamic Reconfiguration* chapter.

EyeQ

The EyeQ hardware is available in Stratix IV GX and GT transceivers to analyze the receiver data recovery path, including receiver gain, clock jitter and noise level. You can use EyeQ to monitor the width of the incoming data eye and assess the quality of the incoming signal.

Normally, the receiver CDR samples the incoming signal at the center of the eye. When you enable the EyeQ hardware, it allows the CDR to sample across 32 different positions within one unit interval (UI) of a data eye. You can manually control the sampling points and check the bit-error rate (BER) at each of these 32 sampling points.

At the center of the eye, the BER is 0. As the sampling point is moved away from the center of the eye towards an edge, the BER increases. By observing sampling points with 0 BER and sampling points with higher BER, you can determine the eye width.

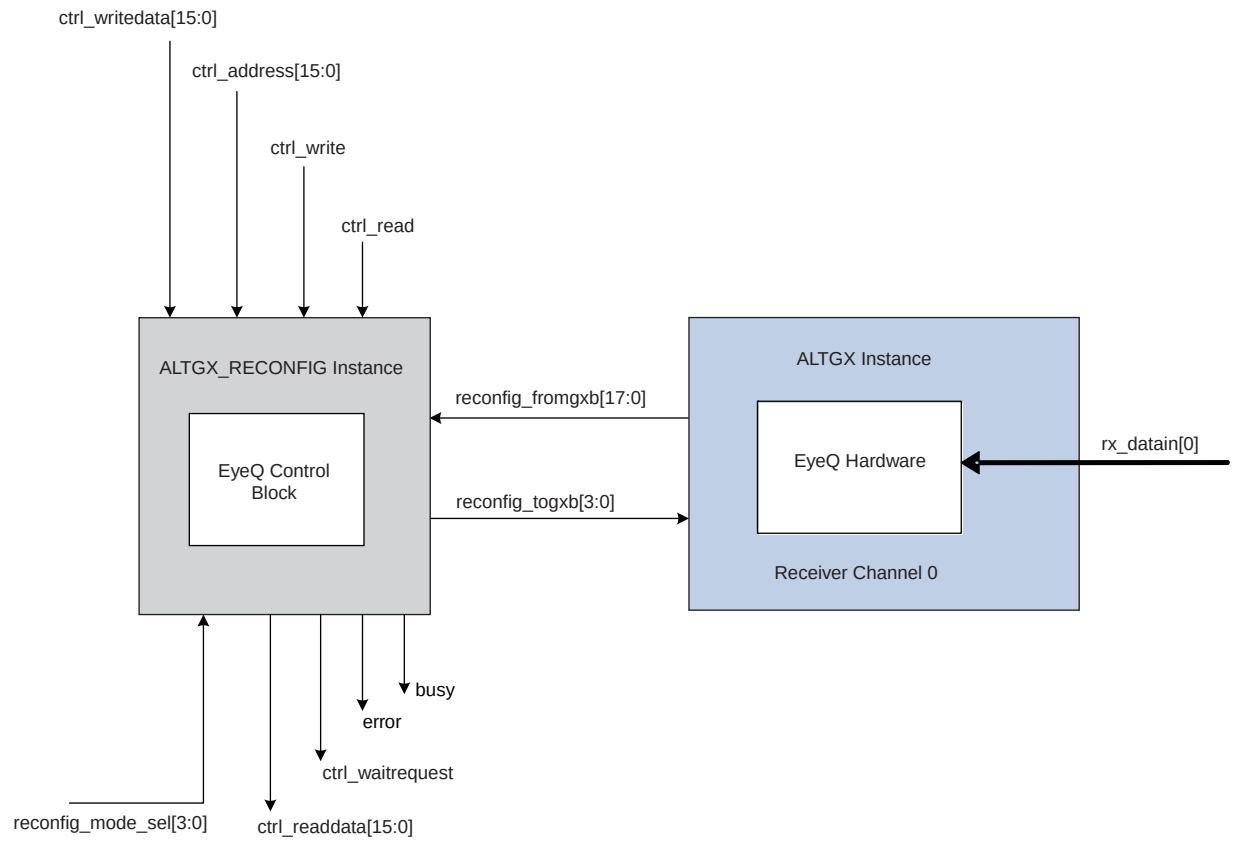
The EyeQ hardware is available for both regular transceiver channels and CMU channels.

The EyeQ block resides within the PMA of the receiver channel and is available for both the transceiver channels and CMU channels of a transceiver block. Figure 1–40 shows the EyeQ feature within a receiver channel datapath.

You must implement logic to check the bit error rate (BER). This includes a pattern generator and checker.

Figure 1–40 shows the receiver channel data path using the EyeQ feature.

Figure 1–40. Receiver Channel Data Path showing the EyeQ Feature

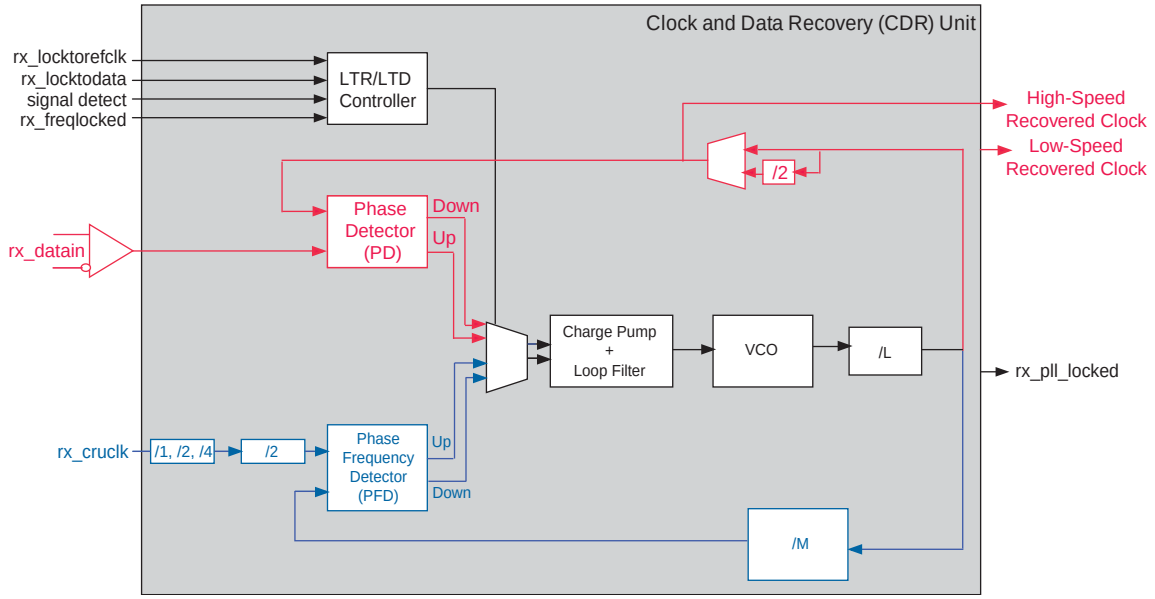


For more information about using the EyeQ feature, refer to the *Stratix IV Dynamic Reconfiguration* chapter.

Clock and Data Recovery Unit

Each Stratix IV GX and GT receiver channel has an independent CDR unit to recover the clock from the incoming serial data stream. The high-speed and low-speed recovered clocks are used to clock the receiver PMA and PCS blocks. Figure 1–41 shows the CDR block diagram.

Figure 1–41. Clock and Data Recovery Unit *(Note 1)*



Note to Figure 1–41:

(1) The blue colored path is active in lock-to-reference mode; the red colored path is active in lock-to-data mode.

The CDR operates either in LTR mode or LTD mode. In LTR mode, the CDR tracks the input reference clock. In LTD mode, the CDR tracks the incoming serial data.

After the receiver power up and reset cycle, the CDR must be kept in LTR mode until it locks to the input reference clock. Once locked to the input reference clock, the CDR output clock is trained to the configured data rate. The CDR can now switch to LTD mode to recover the clock from incoming data. The LTR/LTD controller controls the switch between LTR and LTD modes.

Lock-to-Reference (LTR) Mode

In LTR mode, the phase frequency detector in the CDR tracks the receiver input reference clock, `rx_cruclk`. The PFD controls the charge pump that tunes the VCO in the CDR. Depending on the data rate and the selected input reference clock frequency, the Quartus II software automatically selects the appropriate /M and /L divider values such that the CDR output clock frequency is half the data rate. An active high, the `rx_pll_locked` status signal is asserted to indicate that the CDR has locked to phase and frequency of the receiver input reference clock. Figure 1–41 on page 1–50 shows active blocks (in blue) when CDR is in LTR mode.

The phase detector (PD) is inactive in LTR mode.

You can drive the receiver input reference clock with the following clock sources:

- Dedicated REFCLK pins (`refclk0` and `refclk1`) of the associated transceiver block
- Inter-transceiver block (ITB) clock lines from other transceiver blocks on the same side of the device (up to six ITB clock lines, two from each transceiver block)
- Global PLD clock driven by a dedicated clock input pin
- Clock output from the left and right PLLs in the FPGA fabric

Table 1–24 shows CDR specifications in LTR mode.

Table 1–24. CDR Specifications in Lock-To-Reference Mode

Parameter	Value
Input Reference Clock Frequency	50 MHz to 672 MHz *(1)*
PFD Input Frequency	50 MHz to 325 MHz
/M Divider	4, 5, 8, 10, 16, 20, 25
/L Divider	1, 2, 4, 8

Note to Table 1–24:

(1) The maximum reference clock frequency of 672 MHz is only applicable to speed grades -2 and -3. For speed grade -4, the maximum reference clock frequency is 637.5 MHz

For input reference clock frequencies greater than 325 MHz, the Quartus II software automatically selects the appropriate /1, /2, /4 pre-divider to meet the PFD input frequency limitation of 325 MHz.

Lock-to-Data (LTD) Mode

The CDR must be in LTD mode to recover the clock from the incoming serial data during normal operation. In LTD mode, the phase detector (PD) in the CDR tracks the incoming serial data at the receiver buffer. Depending on the phase difference between the incoming data and the CDR output clock, the PD controls the CDR charge pump that tunes the VCO. Figure 1–41 on page 1–50 shows active blocks (in red) when the CDR is in LTD mode.

The PFD is inactive in LTD mode. The `rx_pll_locked` signal toggles randomly and has no significance in LTD mode.

After switching to LTD mode, it can take a maximum of 1 ms for the CDR to get locked to the incoming data and produce a stable recovered clock. The actual lock time depends on the transition density of the incoming data and the PPM difference between the receiver input reference clock and the upstream transmitter reference clock. The receiver PCS logic must be held in reset until the CDR produces a stable recovered clock.

For more information about receiver reset recommendations, refer to the *Reset Control and Power Down* chapter.

PCI Express (PIPE) Clock Switch Circuitry

The feedback path from the CDR VCO to the PD has a /2 divider that is used in PCI Express (PIPE) mode configured at Gen2 (5 Gbps) data rate for the dynamic switch between Gen1 (2.5 Gbps) and Gen2 (5 Gbps) signaling rates. When the PHY-MAC layer instructs a Gen2-to-Gen1 signaling rateswitch, the /2 divider is enabled. When the PHY-MAC layer instructs a Gen1-to-Gen2 signaling rateswitch, the /2 divider is disabled. For more information about the PCI Express (PIPE) signaling rateswitch, refer to "Dynamic Switch Between Gen1 (2.5 Gbps) and Gen2 (5 Gbps) Signaling Rate" on page 1–138.

The /2 divider in the receiver CDR between the VCO and the PD is disabled in all other functional modes.

LTR/LTD Controller

The LTR/LTD controller controls whether the CDR is in LTR or LTD mode. You can configure the LTR/LTD controller either in automatic lock mode or manual lock mode.

Two optional input ports (`rx_locktorefclk` and `rx_locktodata`) allow you to configure the LTR/LTD controller in either automatic lock mode or manual lock mode. Table 1–25 shows the relationship between these optional input ports and the LTR/LTD controller lock mode.

Table 1–25. Optional Input Ports and LTR/LTD Controller Lock Mode

rx_locktorefclk	rx_locktodata	LTR/LTD Controller Lock Mode
1	0	Manual – LTR Mode
x	1	Manual – LTD Mode
0	0	Automatic Lock Mode

If you do not instantiate the optional `rx_locktorefclk` and `rx_locktodata` signals, the Quartus II software automatically configures the LTR/LTD controller in automatic lock mode.

Automatic Lock Mode

In automatic lock mode, the LTR/LTD controller initially sets the CDR to lock to the input reference clock (LTR mode). After the CDR locks to the input reference clock, the LTR/LTD controller automatically sets it to lock to the incoming serial data (LTD mode) when the following three conditions are met:

- Signal threshold detection circuitry indicates the presence of valid signal levels at the receiver input buffer
  - Valid for PCI Express (PIPE) mode only. This condition is defaulted to **true** for all modes.
- The CDR output clock is within the configured PPM frequency threshold setting with respect to the input reference clock (frequency locked)
- The CDR output clock and the input reference clock are phase matched within approximately 0.08 UI (phase locked)

The switch from LTR to LTD mode is indicated by the assertion of the `rx_freqlocked` signal.

In LTD mode, the CDR uses a phase detector to keep the recovered clock phase-matched to the data. If the CDR does not stay locked to data due to frequency drift or severe amplitude attenuation, the LTR/LTD controller switches the CDR back to LTR mode to lock to the input reference clock. In automatic lock mode, the LTR/LTD controller switches the CDR from LTD to LTR mode when the following conditions are met:

- Signal threshold detection circuitry indicates the absence of valid signal levels at the receiver input buffer
  - Valid for PCI Express (PIPE) mode only. This condition is defaulted to **true** for all other modes.
- The CDR output clock is not within the configured PPM frequency threshold setting with respect to the input reference clock

The switch from LTD to LTR mode is indicated by the de-assertion of the `rx_freqlocked` signal.

Manual Lock Mode

In automatic lock mode, the LTR/LTD controller relies on the PPM detector and the phase relationship detector to set the CDR in LTR or LTD mode. The PPM detector and phase relationship detector reaction times can be too long for some applications that require faster CDR lock time. You can manually control the CDR to reduce its lock time using the `rx_locktorefclk` and `rx_locktodata` ports. In manual lock mode, the LTR/LTD controller sets the CDR in LTR or LTD mode depending on the logic level on the `rx_locktorefclk` and `rx_locktodata` signals.

When the `rx_locktorefclk` signal is asserted high, the LTR/LTD controller forces the CDR to lock to the reference clock. When the `rx_locktodata` signal is asserted high, it forces the CDR to lock to data. When both signals are asserted, the `rx_locktodata` signal takes precedence over the `rx_locktorefclk` signal, forcing the CDR to lock to data.

When the `rx_locktorefclk` signal is asserted high, the `rx_freqlocked` signal does not have any significance and is always driven low, indicating that the CDR is in LTR mode. When the `rx_locktodata` signal is asserted high, the `rx_freqlocked` signal is always driven high, indicating that the CDR is in LTD mode. If both signals are de-asserted, the CDR is in automatic lock mode.

The Altera-recommended transceiver reset sequence varies depending on the CDR lock mode.

For more information about reset sequence recommendations, refer to the *Reset Control and Power Down* chapter.

Offset Cancellation in the Receiver Buffer and Receiver CDR

As silicon progresses towards smaller process nodes, the performance of circuits at these smaller nodes depends more on process variations. These process variations result in analog voltages that can be offset from the required ranges. Offset cancellation logic corrects these offsets. The receiver buffer and receiver CDR require offset cancellation.

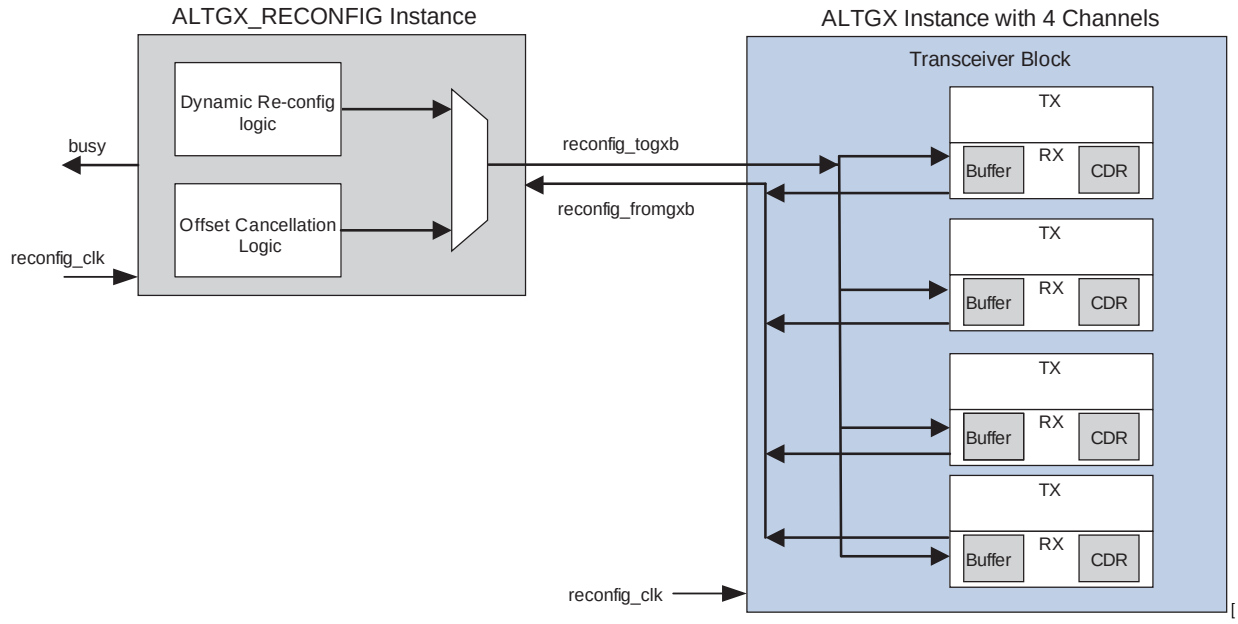
Offset cancellation is executed automatically once each time a Stratix IV GX and GT device is powered on. The control logic for offset cancellation is integrated into the ALTGX_RECONFIG megafunction. The `reconfig_fromgxb` and `reconfig_togxb` buses and the necessary clocks must be connected between the ALTGX instance and the ALTGX_RECONFIG instance.

For more information about offset cancellation control logic connectivity, refer to the *Stratix IV Dynamic Reconfiguration* chapter.

During offset cancellation, signified by high on the `busy` signal, the `rx_analogreset` is not relevant until the `busy` signal goes low.

Offset cancellation logic requires a separate clock. In PCI Express (PIPE) mode, you must connect the clock input to the `fixedclk` port provided by the ALTGX MegaWizard Plug-In Manager. The frequency of this clock input must be 125 MHz. For all other functional modes, connect the clock input to the `reconfig_clk` port provided by the ALTGX MegaWizard Plug-In Manager. The frequency of the clock connected to the `reconfig_clk` port must be within the range of 37.5 to 50 MHz. Figure 1–42 shows the interface of the offset cancellation control logic (ALTGX_RECONFIG instance) and the ALTGX instance.

Figure 1–42. Interface of Offset Cancellation Control Logic to the ALTGX Instance



The offset cancellation process begins by disconnecting the path from the receiver input buffer to the receiver CDR. It then sets the receiver CDR into a fixed set of dividers to guarantee a VCO clock rate that is within the range necessary to provide proper offset cancellation. Subsequently, the offset cancellation process goes through various states and culminates in the offset cancellation of the receiver buffer and the receiver CDR.

After offset cancellation is complete, your divider settings are restored. Then the reconfiguration block sends and receives data to the ALTGX instance using the `reconfig_togxb` and `reconfig_fromgxb` buses. Connect the buses between the ALTGX_RECONFIG and ALTGX instances. The de-assertion of the `busy` signal from the offset cancellation control logic indicates the offset cancellation process is complete.

Due to the offset cancellation process, the transceiver reset sequence has changed. For more information on the offset cancellation process, refer to the *Reset Control and Power Down* chapter.

Deserializer

The deserializer block clocks in serial input data from the receiver buffer using the high-speed serial recovered clock and deserializes it using the low-speed parallel recovered clock. It forwards the deserialized data to the receiver PCS channel.

In single-width mode, the deserializer supports 8-bit and 10-bit deserialization factors. In double-width mode, the deserializer supports 16-bit and 20-bit deserialization factors.

Figure 1–43 shows the deserializer operation in single-width mode with a 10-bit deserialization factor.

Figure 1–43. Deserializer Operation in Single-Width Mode

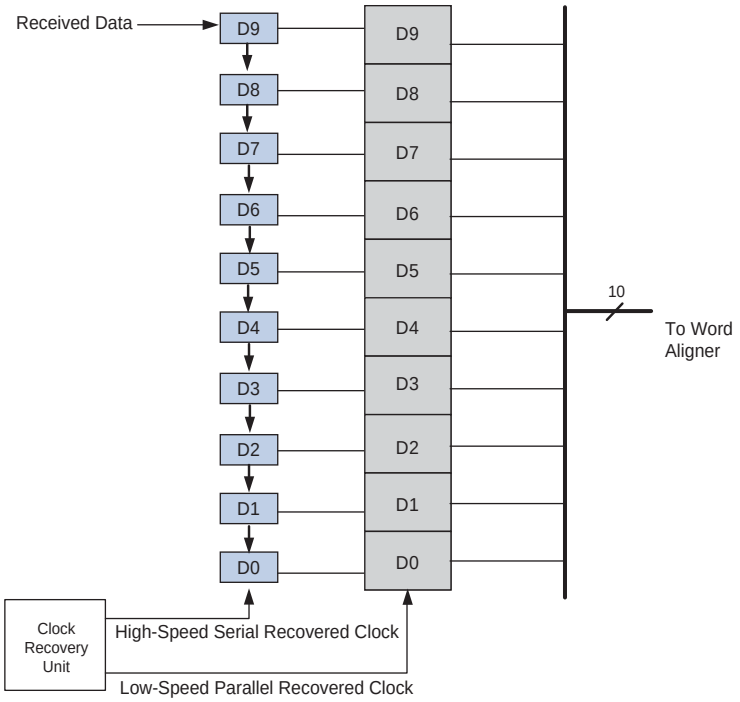
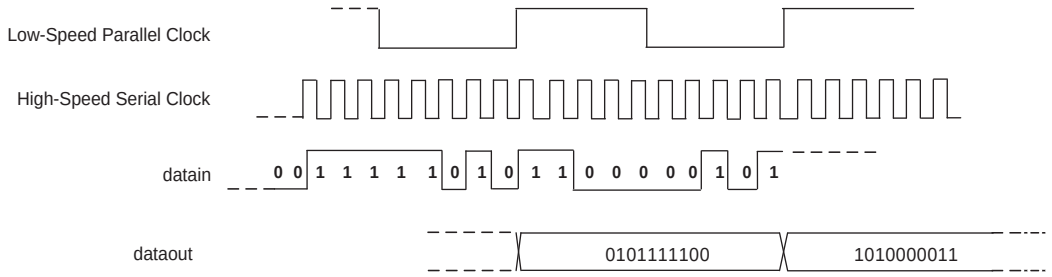


Figure 1–44 shows the serial bit order of the deserializer block input and the parallel data output of the deserializer block in single-width mode with a 10-bit deserialization factor. The serial stream (0101111100) is deserialized to a value 10'h17C. The serial data is assumed to be received LSB to MSB.

Figure 1–44. Deserializer Bit Order in Single-Width Mode



Word Aligner

Because the data is serialized before transmission and then deserialized at the receiver, it loses the word boundary of the upstream transmitter upon deserialization. The word aligner receives parallel data from the deserializer and restores the word boundary based on a pre-defined alignment pattern that must be received during link synchronization.

Serial protocols such as PCI Express (PIPE), XAUI, Gigabit Ethernet, Serial RapidIO, and SONET/SDH, specify a standard word alignment pattern. For proprietary protocols, the Stratix IV GX and GT transceiver architecture allows you to select a custom word alignment pattern specific to your implementation.

In addition to restoring the word boundary, the word aligner also implements the following features:

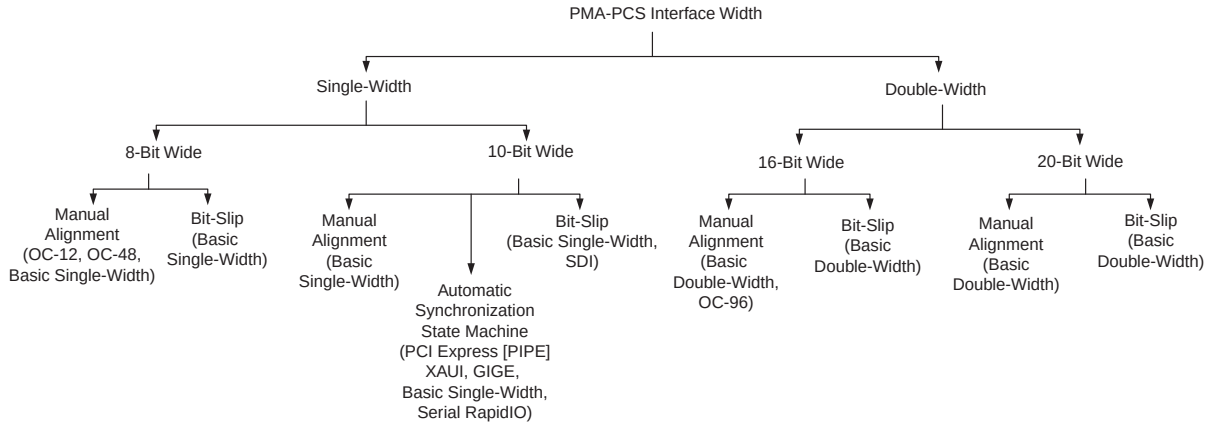
- Synchronization state machine in functional modes such as PCI Express (PIPE), XAUI, GIGE, Serial RapidIO, and Basic single-width
- Programmable run length violation detection in all functional modes
- Receiver polarity inversion in all functional modes except PCI Express (PIPE)
- Receiver bit reversal in Basic single-width and Basic double-width modes
- Receiver byte reversal in Basic double-width modes

Depending on the configured functional mode, the word aligner operates in one of the following three modes:

- Manual alignment mode
- Automatic synchronization state machine mode
- Bit-slip mode

Figure 1–45 shows the word aligner operation in all supported configurations.

Figure 1–45. Word Aligner in All Supported Configurations



Word Aligner in Single-Width Mode

In single-width mode, the PMA-PCS interface is either 8 or 10 bits wide. In 8-bit wide PMA-PCS interface modes, the word aligner receives 8-bit wide data from the deserializer. In 10-bit wide PMA-PCS interface modes, the word aligner receives 10-bit wide data from the deserializer. Depending on the configured functional mode, you can configure the word aligner in manual alignment mode, automatic synchronization state machine mode, or bit-slip mode.

Word Aligner in Single-Width Mode with 8-Bit PMA-PCS Interface Modes

The following functional modes support the 8-bit PMA-PCS interface:

- SONET/SDH OC-12
- SONET/SDH OC-48
- Basic single-width

Table 1–26 shows the word aligner configurations allowed in functional modes with an 8-bit PMA-PCS interface.

Table 1–26. Word Aligner Configurations with an 8-Bit PMA-PCS Interface

Functional Mode	Allowed Word Configurations	Allowed Word Alignment Pattern Length
SONET/SDH OC-12	Manual Alignment	16 bits
SONET/SDH OC-48	Manual Alignment	16 bits
Basic single-width	Manual Alignment, Bit-Slip	16 bits

Manual Alignment Mode Word Aligner with 8-Bit PMA-PCS Interface Modes

In manual alignment mode, the word aligner operation is controlled by the input signal `rx_enapatternalign`. The word aligner operation is edge-sensitive to the `rx_enapatternalign` signal. After de-assertion of `rx_digitalreset`, a rising edge on the `rx_enapatternalign` signal triggers the word aligner to look for the word alignment pattern in the received data stream. In SONET/SDH OC-12 and OC-48 modes, the word aligner looks for 16'hF628 (A1A2) or 32'hF6F62828

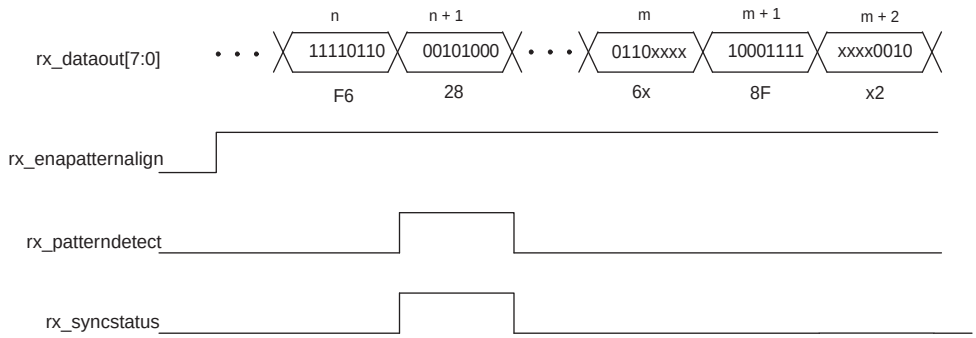
(A1A1A2A2), depending on whether the input signal `rx_a1a2size` is driven low or high, respectively. In Basic single-width mode, the word aligner looks for the 16-bit word alignment pattern programmed in the ALTGX MegaWizard Plug-In Manager. The word aligner aligns the 8-bit word boundary to the first word alignment pattern received after the rising edge on the `rx_enapatternalign` signal.

Two status signals, `rx_syncstatus` and `rx_patterndetect`, with the same latency as the datapath, are forwarded to the FPGA fabric to indicate word aligner status. On receiving the first word alignment pattern after the rising edge on the `rx_enapatternalign` signal, both the `rx_syncstatus` and `rx_patterndetect` signals are driven high for one parallel clock cycle synchronous to the MSByte of the word alignment pattern. Any word alignment pattern received thereafter in the same word boundary causes only the `rx_patterndetect` signal to go high for one clock cycle.

For the word aligner to re-synchronize to a new word boundary, you must de-assert `rx_enapatternalign` and re-assert it again to create a rising edge. After a rising edge on the `rx_enapatternalign` signal, if the word alignment pattern is found in a different word boundary, the word aligner re-synchronizes to the new word boundary and asserts the `rx_syncstatus` and `rx_patterndetect` signals for one parallel clock cycle.

Figure 1–46 shows word aligner behavior in SONET/SDH OC-12 functional mode. The LSByte (8'hF6) and the MSByte (8'h28) of the 16-bit word alignment pattern are received in parallel clock cycles n and n + 1, respectively. The `rx_syncstatus` and `rx_patterndetect` signals are both driven high for one parallel clock cycle synchronous to the MSByte (8'h28) of the word alignment pattern. After initial word alignment, the 16-bit word alignment pattern is again received across the word boundary in clock cycles m, m + 1, and m + 2. The word aligner does not re-align to the new word boundary because of the lack of a preceding rising edge on the `rx_enapatternalign` signal. If you create a rising edge on the `rx_enapatternalign` signal before the word alignment pattern is received across clock cycles m, m + 1, and m + 2, the word aligner re-aligns to the new word boundary, causing both the `rx_syncstatus` and `rx_patterndetect` signals to go high for one parallel clock cycle.

Figure 1–46. Bit-Slip Mode in 8-Bit PMA-PCS Interface Mode



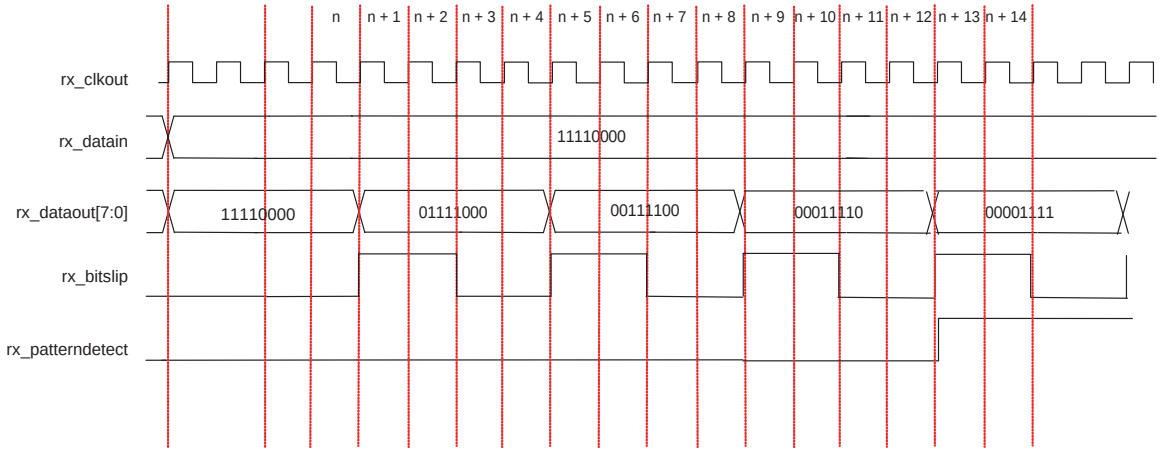
Bit-Slip Mode Word Aligner with 8-Bit PMA-PCS Interface Modes

Basic single-width mode with 8-bit PMA-PCS interface width allows the word aligner to be configured in bit-slip mode. The word aligner operation is controlled by the input signal `rx_bitslip` in bit-slip mode. At every rising edge of the `rx_bitslip` signal, the bit-slip circuitry slips one bit into the received data stream, effectively shifting the word boundary by one bit. In bit-slip mode, the word aligner status signal `rx_patterndetect` is driven high for one parallel clock cycle when the received data after bit-slipping matches the 16-bit word alignment pattern programmed in the ALTGX MegaWizard Plug-In Manager.

You can implement a bit-slip controller in the FPGA fabric that monitors either the `rx_dataout` signal and/or the `rx_patterndetect` signal and controls the `rx_bitslip` signal to achieve word alignment.

Figure 1–47 shows an example of the word aligner configured in bit-slip mode. For this example, consider that 8'b11110000 is received back-to-back and 16'b0000111100011110 is specified as the word alignment pattern. A rising edge on the `rx_bitslip` signal at time n + 1 slips a single bit 0 at the MSB position, forcing the `rx_dataout` to 8'b01111000. Another rising edge on the `rx_bitslip` signal at time n + 5 forces `rx_dataout` to 8'b00111100. Another rising edge on the `rx_bitslip` signal at time n + 9 forces `rx_dataout` to 8'b00011110. Another rising edge on the `rx_bitslip` signal at time n + 13 forces the `rx_dataout` to 8'b00001111. At this instance, `rx_dataout` in cycles n + 12 and n + 13 is 8'b00011110 and 8'b00001111, respectively, which matches the specified 16-bit alignment pattern 16'b0000111100011110. This results in the assertion of the `rx_patterndetect` signal.

Figure 1–47. Word Aligner Configured in Bit-Slip Mode



Word Aligner in Single-Width Mode with 10-Bit PMA-PCS Interface Modes

The following functional modes support the 10-bit PMA-PCS interface:

- PCI Express (PIPE) Gen1 and Gen2
- Serial RapidIO
- XAUI
- GIGE
- SDI
- Basic single-width mode

This section describes the following word aligner 10-bit PMA-PCS interface modes:

- Automatic synchronization state machine mode with 10-bit PMA-PCS interface mode
- Manual alignment mode with 10-bit PMA-PCS interface mode
- Bit-slip mode with 10-bit PMA-PCS interface mode

Table 1–27 shows the word aligner configurations allowed in functional modes with a 10-bit PMA-PCS interface.

Table 1–27. Word Aligner Configurations with a 10-Bit PMA-PCS Interface

Functional Mode	Allowed Word Aligner Configurations	Allowed Word Alignment Pattern Length
PCI Express (PIPE)	Automatic synchronization state machine	10 bits
Serial RapidIO	Automatic synchronization state machine	10 bits
XAUI	Automatic synchronization state machine	7 bits, 10 bits
GIGE	Automatic synchronization state machine	7 bits, 10 bits
SDI	Bit-slip	N/A
Basic single-width mode	Manual alignment, Automatic synchronization state machine, Bit-slip	7 bits, 10 bits

Automatic Synchronization State Machine Mode Word Aligner with 10-Bit PMA-PCS Interface Mode

Protocols such as PCI Express (PIPE), XAUI, Gigabit Ethernet, and Serial RapidIO require the receiver PCS logic to implement a synchronization state machine to provide hysteresis during link synchronization. Each of these protocols defines a specific number of synchronization code groups that the link must receive to acquire synchronization and a specific number of erroneous code groups that it must receive to fall out of synchronization.

In PCI Express (PIPE), XAUI, Gigabit Ethernet, and Serial RapidIO functional modes, the Quartus II software configures the word aligner in automatic synchronization state machine mode. It automatically selects the word alignment pattern length and pattern as specified by each protocol. In each of these functional modes, the protocol-compliant synchronization state machine is implemented in the word aligner.

In Basic single-width functional mode with a 10-bit PMA-PCS interface, you can configure the word aligner in automatic synchronization state machine mode by selecting the **Use the built-in synchronization state machine** option in the ALTGX MegaWizard Plug-In Manager. It also allows you to program a custom 7-bit or 10-bit word alignment pattern that the word aligner uses for synchronization.

The 10-bit input data to the word aligner configured in automatic synchronization state machine mode must be 8B/10B encoded.

Table 1–28 shows the synchronization state machine parameters that the Quartus II software allows in supported functional modes. The synchronization state machine parameters are fixed for PCI Express (PIPE), XAUI, GIGE, and Serial RapidIO modes as specified by the respective protocol. For Basic single-width mode, you can program these parameters as suited to your proprietary protocol implementation.

Table 1–28. Synchronization State Machine Functional Modes

Functional Mode	PCI Express (PIPE)	XAUI	GIGE	Serial RapidIO	Basic Single-Width Mode
Number of valid synchronization code groups or ordered sets received to achieve synchronization	4	4	3	127	1 to 256
Number of erroneous code groups received to lose synchronization	17	4	4	3	1 to 64
Number of continuous good code groups received to reduce the error count by one	16	4	4	255	1 to 256

After de-assertion of the `rx_digitalreset` signal in automatic synchronization state machine mode, the word aligner starts looking for the word alignment pattern or synchronization code groups in the received data stream. When the programmed number of valid synchronization code groups or ordered sets is received, the `rx_syncstatus` signal is driven high to indicate that synchronization is acquired. The `rx_syncstatus` signal is constantly driven high until the programmed number of erroneous code groups is received without receiving intermediate good groups; after which the `rx_syncstatus` is driven low. The word aligner indicates loss of synchronization (`rx_syncstatus` remains low) until the programmed number of valid synchronization code groups are received again.

Manual Alignment Mode Word Aligner with 10-Bit PMA-PCS Interface Mode

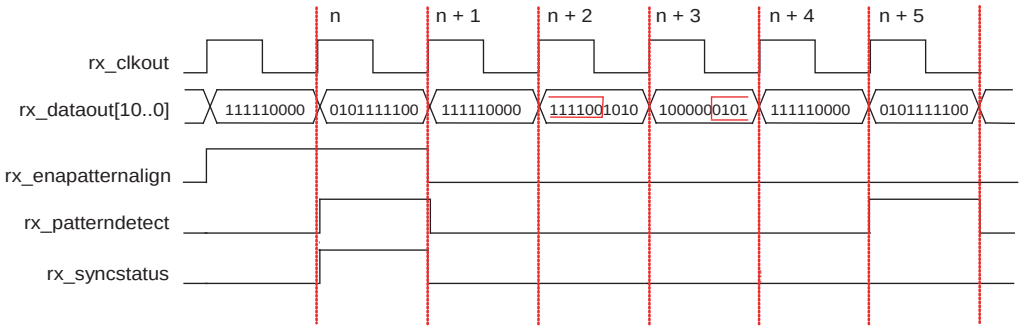
In Basic single-width mode with a 10-bit PMA-PCS interface, you can configure the word aligner in manual alignment mode by selecting the **Use manual word alignment mode** option in the ALTGX MegaWizard Plug-In Manager.

In manual alignment mode, the word aligner operation is controlled by the input signal `rx_enapatternalign`. The word aligner operation is level-sensitive to the `rx_enapatternalign` signal. If the `rx_enapatternalign` signal is held high, the word aligner looks for the programmed 7-bit or 10-bit word alignment pattern in the received data stream. It updates the word boundary if it finds the word alignment pattern in a new word boundary. If the `rx_enapatternalign` signal is de-asserted low, the word aligner maintains the current word boundary even when it sees the word alignment pattern in a new word boundary.

Two status signals, `rx_syncstatus` and `rx_patterndetect`, with the same latency as the datapath, are forwarded to the FPGA fabric to indicate the word aligner status. After receiving the first word alignment pattern after the `rx_enapatternalign` signal is asserted high, both the `rx_syncstatus` and `rx_patterndetect` signals are driven high for one parallel clock cycle. Any word alignment pattern received thereafter in the same word boundary causes only the `rx_patterndetect` signal to go high for one clock cycle. Any word alignment pattern received thereafter in a different word boundary causes the word aligner to re-align to the new word boundary only if the `rx_enapatternalign` signal is held high. The word aligner asserts the `rx_syncstatus` signal for one parallel clock cycle whenever it re-aligns to the new word boundary.

Figure 1–48 shows the manual alignment mode word aligner operation with 10-bit PMA-PCS interface mode. In this example, a /K28.5/ (10'b0101111100) is specified as the word alignment pattern. The word aligner aligns to the /K28.5/ alignment pattern in cycle n because the `rx_enapatternalign` signal is asserted high. The `rx_syncstatus` signal goes high for one clock cycle, indicating alignment to a new word boundary. The `rx_patterndetect` signal also goes high for one clock cycle to indicate initial word alignment. At time n + 1, the `rx_enapatternalign` signal is de-asserted to instruct the word aligner to lock the current word boundary. The alignment pattern is detected again in a new word boundary across cycles n + 2 and n + 3. The word aligner does not align to this new word boundary because the `rx_enapatternalign` signal is held low. The /K28.5/ word alignment pattern is detected again in the current word boundary during cycle n + 5, causing the `rx_patterndetect` signal to go high for one parallel clock cycle.

Figure 1–48. Word Aligner with 10-Bit PMA-PCS Manual Alignment Mode



If the word alignment pattern is known to be unique and does not appear between word boundaries, you can constantly hold the `rx_enapatternalign` signal high because there is no possibility of false word alignment. If there is a possibility of the word alignment pattern occurring across word boundaries, you must control the `rx_enapatternalign` signal to lock the word boundary after the desired word alignment is achieved to avoid re-alignment to an incorrect word boundary.

Bit-Slip Mode Word Aligner with 10-Bit PMA-PCS Interface Mode

In some Basic single-width configurations with a 10-bit PMA-PCS interface, you can configure the word aligner in bit-slip mode by selecting the **Use manual bit slipping mode** option in the ALTGX MegaWizard Plug-In Manager.

The word aligner operation for Basic single-width with a 10-bit PMA-PCS interface is similar to the word aligner operation in Basic single-width mode with an 8-bit PMA-PCS interface. For word aligner operation in bit-slip mode, refer to "Manual Alignment Mode Word Aligner with 8-Bit PMA-PCS Interface Modes" on page 1–57. The only difference is that the bit-slip word aligner with 10-bit PMA-PCS interface modes allows 7-bit and 10-bit word alignment patterns, whereas the one with 8-bit PMA-PCS interface modes allows only 16-bit word alignment patterns.

Word Aligner in Double-Width Mode

In double-width mode, the PMA-PCS interface is either 16 or 20 bits wide. In 16-bit PMA-PCS interface modes, the word aligner receives 16 bit wide data from the deserializer. In 20-bit PMA-PCS interface modes, the word aligner receives 10-bit wide data from the deserializer. Depending on the configured functional mode, you can configure the word aligner in manual alignment mode or bit-slip mode. Automatic synchronization state machine mode is not supported for word aligner in double-width mode.

Word Aligner in Double-Width Mode with 16-Bit PMA-PCS Interface Modes

The following functional modes support the 16-bit PMA-PCS interface:

- SONET/SDH OC-96
- (OIF) CEI PHY interface
- Basic double-width

Table 1–29 shows the word aligner configurations allowed in functional modes with a 16-bit PMA-PCS interface.

Table 1–29. Word Aligner Configurations with 16-Bit PMA-PCS Interface *(Note 1)*

Functional Mode	Allowed Word Aligner Configurations	Allowed Word Alignment Pattern Length
SONET/SDH OC-96	Manual alignment	16 bits, 32 bits
Basic double-width	Manual alignment, Bit-slip	8 bits, 16 bits, 32 bits

Note to Table 1–29:

(1) The word aligner is bypassed in (OIF) CEI PHY interface mode.

Manual Alignment Mode Word Aligner with 16-Bit PMA-PCS Interface Modes

In manual alignment mode, the word aligner starts looking for the programmed 8-bit, 16-bit, or 32-bit word alignment pattern in the received data stream as soon as `rx_digitalreset` is de-asserted low. It aligns to the first word alignment pattern received regardless of the logic level driven on the `rx_enapatternalign` signal. Any word alignment pattern received thereafter in a different word boundary does not cause the word aligner to re-align to this new word boundary. After the initial word alignment following de-assertion of the `rx_digitalreset` signal, if a word re-alignment is required, you must use the `rx_enapatternalign` signal.

Word aligner operation is controlled by the input signal `rx_enapatternalign` and is edge-sensitive to the `rx_enapatternalign` signal. A rising edge on the `rx_enapatternalign` signal triggers the word aligner to look for the word alignment pattern in the received data stream. The word aligner aligns the 16-bit word boundary to the first word alignment pattern received after the rising edge on the `rx_enapatternalign` signal. Any word alignment pattern received thereafter in a different word boundary does not cause the word aligner to re-align to this new word boundary. If another word re-alignment is required, you must de-assert and re-assert the `rx_enapatternalign` signal to create a rising edge on this signal.

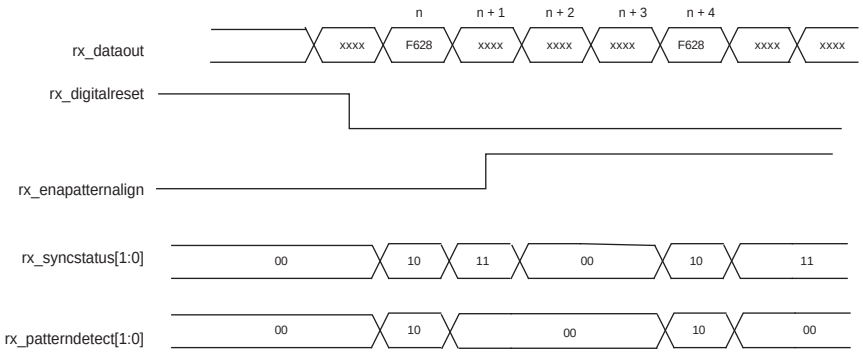
Two status signals, `rx_syncstatus` and `rx_patterndetect`, with the same latency as the datapath, are forwarded to the FPGA fabric to indicate word aligner status.

After receiving the first word alignment pattern, the `rx_patterndetect` signal is driven high for one parallel clock cycle synchronous to the data that matches the MSByte of the word alignment pattern. Any word alignment pattern received thereafter in the same word boundary causes `rx_patterndetect` to go high for one parallel clock cycle.

After receiving the first word alignment pattern, the `rx_syncstatus` signal is constantly driven high until the word aligner sees another rising edge on the `rx_enapatternalign` signal. The rising edge on the `rx_enapatternalign` signal re-triggers the word alignment operation.

Figure 1–49 shows the manual alignment mode word aligner operation in 16-bit PMA-PCS interface mode. In this example, a 16'hF628 is specified as the word alignment pattern. The word aligner aligns to the 16'hF628 pattern received in cycle n after de-assertion of `rx_digitalreset`. The `rx_patterndetect[1]` signal is driven high for one parallel clock cycle. The `rx_syncstatus[1]` signal is driven high constantly until cycle n + 2, after which it is driven low because of the rising edge on the `rx_enapatternalign` signal that re-triggers the word aligner operation. The word aligner receives the word alignment pattern again in cycle n + 4, causing the `rx_patterndetect[1]` signal to be driven high for one parallel clock cycle and the `rx_syncstatus[1]` signal to be driven high constantly.

Figure 1–49. Manual Alignment Mode Word Aligner in 16-Bit PMA-PCS Interface Modes



Bit-Slip Mode Word Aligner with 16-Bit PMA-PCS Interface Modes

In some Basic double-width configurations with 16-bit PMA-PCS interface, you can configure the word aligner in bit-slip mode by selecting the **Use manual bit slipping mode** option in the ALTGX MegaWizard Plug-In Manager.

The word aligner operation for Basic double-width with 16-bit PMA-PCS interface is similar to the word aligner operation in Basic single-width mode with 8-bit PMA-PCS interface. For word aligner operation in bit-slip mode, refer to "Word Aligner in Single-Width Mode with 8-Bit PMA-PCS Interface Modes" on page 1–57. The only difference is that the bit-slip word aligner in 16-bit PMA-PCS interface modes allows 8-bit and 16-bit word alignment patterns, whereas the bit-slip word aligner in 8-bit PMA-PCS interface modes allows only a 16-bit word alignment pattern.

Word Aligner in Double-Width Mode with 20-Bit PMA-PCS Interface Modes

A 20-bit PMA-PCS interface is supported only in Basic double-width mode.

Table 1–30 shows the word aligner configurations allowed in functional modes with a 20-bit PMA-PCS interface.

Table 1–30. Word Aligner in 20-Bit PMA-PCS Interface Modes

Functional Mode	Allowed Word Aligner Configurations	Allowed Word Alignment Pattern Length
Basic double-width	Manual alignment, Bit-slip	7 bits, 10 bits, 20 bits

Manual Alignment Mode Word Aligner with 20-Bit PMA-PCS Interface Modes

The word aligner operation in Basic double-width mode with 20-bit PMA-PCS interface is similar to the word aligner operation in Basic double-width mode with a 16-bit PMA-PCS interface. For word aligner operation in manual alignment mode, refer to "Word Aligner in Double-Width Mode with 16-Bit PMA-PCS Interface Modes" on page 1–63. The only difference is that the manual alignment mode word aligner in 20-bit PMA-PCS interface modes allows 7-, 10-, and 20-bit word alignment patterns, whereas the manual alignment mode word aligner in 16-bit PMA-PCS interface modes allows only 8-, 16-, and 32-bit word alignment patterns.

Bit-Slip Mode Word Aligner with 20-Bit PMA-PCS Interface Modes

In some Basic single-width configurations with 20-bit PMA-PCS interface, you can configure the word aligner in bit-slip mode by selecting the **Use manual bit slipping mode** option in the ALTGX MegaWizard Plug-In Manager.

The word aligner operation for Basic double-width with 20-bit PMA-PCS interface is similar to the word aligner operation in Basic single-width mode with an 8-bit PMA-PCS interface. For word aligner operation in bit-slip mode, refer to "Word Aligner in Single-Width Mode with 8-Bit PMA-PCS Interface Modes" on page 1–57. The difference is that the bit-slip word aligner in 20-bit PMA-PCS interface modes allows only 7-, 10-, and 20-bit word alignment patterns, whereas the bit-slip word aligner in 8-bit PMA-PCS interface modes allows only a 16-bit word alignment pattern.

Table 1–31 summarizes the word aligner options available in Basic single-width and double-width modes.

Table 1–31. Word Aligner Options Available in Basic Single-Width and Double-Width Modes *(Note 1)* (Part 1 of 2)

Functional Mode	PMA-PCS Interface Width	Word Alignment Mode	Word Alignment Pattern Length	rx_enapatternalign Sensitivity	rx_syncstatus Behavior	rx_patterndetect Behavior
Basic Single-Width	8-bit	Manual Alignment	16-bit	Rising Edge Sensitive	Asserted high for one parallel clock cycle when the word aligner aligns to a new word boundary.	Asserted high for one parallel clock cycle when the word alignment pattern appears in the current word boundary.
		Bit-Slip	16-bit	N/A	N/A	Asserted high for one parallel clock cycle when the word alignment pattern appears in the current word boundary.
	10-bit	Manual Alignment	7- and 10-bit	Level Sensitive	Asserted high for one parallel clock cycle when the word aligner aligns to a new word boundary.	Asserted high for one parallel clock cycle when the word alignment pattern appears in the current word boundary.
		Bit-Slip	7- and 10-bit	N/A	N/A	Asserted high for one parallel clock cycle when the word alignment pattern appears in the current word boundary.
		Automatic Synchronization State Machine	7- and 10-bit	N/A	Stays high as long as the synchronization conditions are satisfied.	Asserted high for one parallel clock cycle when the word alignment pattern appears in the current word boundary.

Table 1–31. Word Aligner Options Available in Basic Single-Width and Double-Width Modes *(Note 1)* (Part 2 of 2)

Functional Mode	PMA-PCS Interface Width	Word Alignment Mode	Word Alignment Pattern Length	rx_enapatternalign Sensitivity	rx_syncstatus Behavior	rx_patterndetect Behavior
Basic Double-Width	16-bit	Manual Alignment	8-, 16-, and 32-bit	Rising Edge Sensitive	Stays high after the word aligner aligns to the word alignment pattern. Goes low on receiving a rising edge on `rx_enapatternalign` until a new word alignment pattern is received.	Asserted high for one parallel clock cycle when the word alignment pattern appears in the current word boundary.
		Bit-Slip	8-, 16-, and 32-bit	N/A	N/A	Asserted high for one parallel clock cycle when the word alignment pattern appears in the current word boundary.
	20-bit	Manual Alignment	7-, 10-, and 20-bit	Rising Edge Sensitive	Stays high after the word aligner aligns to the word alignment pattern. Goes low on receiving a rising edge on `rx_enapatternalign` until a new word alignment pattern is received.	Asserted high for one parallel clock cycle when the word alignment pattern appears in the current word boundary.
		Bit-Slip	7-, 10-, and 20-bit	N/A	N/A	Asserted high for one parallel clock cycle when the word alignment pattern appears in the current word boundary.

Note to Table 1–31:

(1) For more information about word aligner operation, refer to "Word Aligner in Single-Width Mode" on page 1–57 and "Word Aligner in Double-Width Mode" on page 1–63.

Programmable Run Length Violation Detection

The programmable run length violation circuit resides in the word aligner block and detects consecutive 1s or 0s in the data. If the data stream exceeds the preset maximum number of consecutive 1s or 0s, the violation is signified by the assertion of the `rx_rlv` signal.

The run length violation status signal on the `rx_rlv` port has lower latency when compared with the parallel data on the `rx_dataout` port. The `rx_rlv` signal in each channel is clocked by its parallel recovered clock. The FPGA fabric clock might have phase difference and/or PPM difference (in asynchronous systems) with respect to the recovered clock. To ensure that the FPGA fabric clock can latch the `rx_rlv` signal reliably, the run length violation circuitry asserts the `rx_rlv` signal for a minimum of two recovered clock cycles in single-width modes and a minimum of three recovered clock cycles in double-width modes. The `rx_rlv` signal can be asserted longer, depending on the run length of the received data.

In single-width mode, the run length violation circuit detects up to a run length of 128 (for an 8-bit deserialization factor) or 160 (for a 10-bit deserialization factor). The settings are in increments of four or five for the 8-bit or 10-bit deserialization factors, respectively.

In double-width mode, the run length violation circuit maximum run length detection is 512 (with a run length increment of eight) and 640 (with a run length increment of 10) for the 16-bit and 20-bit deserialization factors, respectively.

Table 1–32 summarizes the detection capabilities of the run length violation circuit.

Table 1–32. Detection Capabilities of the Run Length Violation Circuit

Mode	PMA-PCS Interface Width	Run Length Violation Detector Range	
		Minimum	Maximum
Single-width mode	8-bit	4	128
	10-bit	5	160
Double-width mode	16-bit	8	512
	20-bit	10	640

Receiver Polarity Inversion

The positive and negative signals of a serial differential link are often erroneously swapped during board layout. Solutions like board re-spin or major updates to the PLD logic can be expensive. The receiver polarity inversion feature is provided to correct this situation.

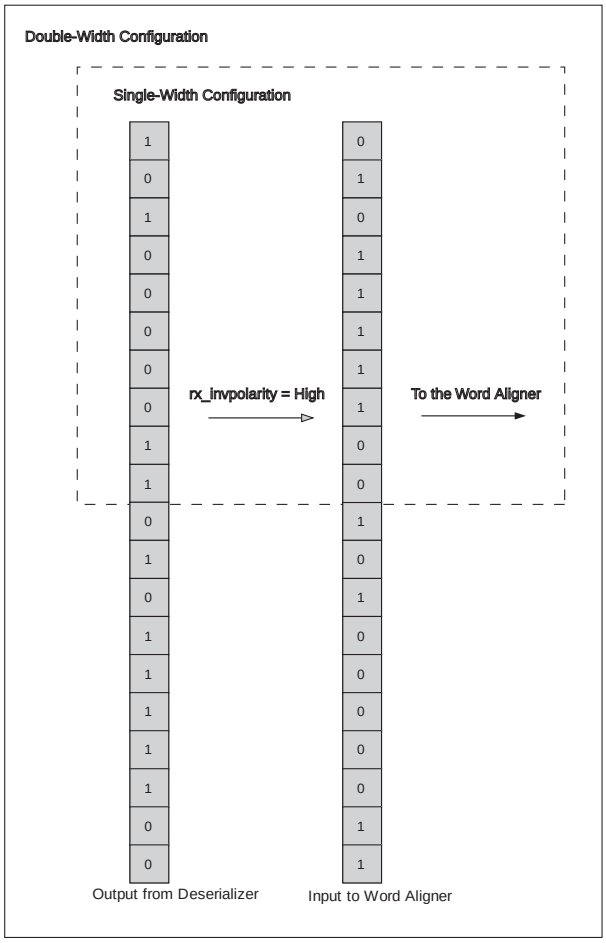
An optional `rx_invpolarity` port is available in all single-width and double-width modes except (OIF) CEI PHY and PCI Express (PIPE) modes to dynamically enable the receiver polarity inversion feature. In single-width modes, a high value on the `rx_invpolarity` port inverts the polarity of every bit of the 8-bit or 10-bit input data word to the word aligner in the receiver datapath. In double-width modes, a high value on the `rx_invpolarity` port inverts the polarity of every bit of the 16-bit or 20-bit input data word to the word aligner in the receiver datapath. Because

inverting the polarity of each bit has the same effect as swapping the positive and negative signals of the differential link, correct data is seen by the receiver. `rx_invpolarity` is a dynamic signal and can cause initial disparity errors in an 8B/10B encoded link. The downstream system must be able to tolerate these disparity errors.

The generic receiver polarity inversion feature is different from the PCI Express (PIPE) 8B/10B polarity inversion feature. The generic receiver polarity inversion feature inverts the polarity of the data bits at the input of the word aligner and is not available in PCI Express (PIPE) mode. The PCI Express (PIPE) 8B/10B polarity inversion feature inverts the polarity of the data bits at the input of the 8B/10B decoder and is available only in PCI Express (PIPE) mode.

Figure 1–50 shows the receiver polarity inversion feature in single-width and double-width datapath configurations.

Figure 1–50. Receiver Polarity Inversion in Single-Width and Double Width Mode



Receiver Bit Reversal

By default, the Stratix IV GX and GT receiver assumes a LSB-to-MSB transmission. If the transmission order is MSB-to-LSB, the receiver forwards the bit-flipped version of the parallel data to the FPGA fabric on the `rx_dataout` port. The receiver bit reversal feature is available to correct this situation.

The receiver bit reversal feature is available through the `rx_revbitordwa` port only in Basic single-width and double-width modes with the word aligner configured in bit-slip mode. When the `rx_revbitordwa` signal is driven high in Basic single-width mode, the 8-bit or 10-bit data `D[7:0]` or `D[9:0]` at the output of the word aligner gets rewired to `D[0:7]` or `D[0:9]`, respectively. When the `rx_revbitordwa` signal is driven high in Basic double-width mode, the 16-bit or 20-bit data `D[15:0]` or `D[19:0]` at the output of the word aligner gets rewired to `D[0:15]` or `D[0:19]`, respectively.

Flipping the parallel data using this feature allows the receiver to forward the correct bit-ordered data to the FPGA fabric on the `rx_dataout` port in the case of MSB-to-LSB transmission.

Figure 1–51 shows the receiver bit reversal feature in Basic single-width 10-bit wide datapath configurations.

Figure 1–51. Receiver Bit Reversal in Single-Width Mode

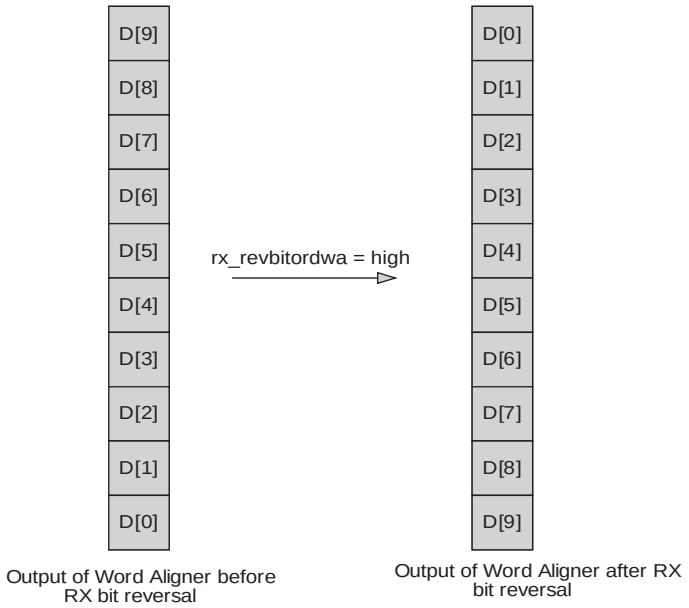
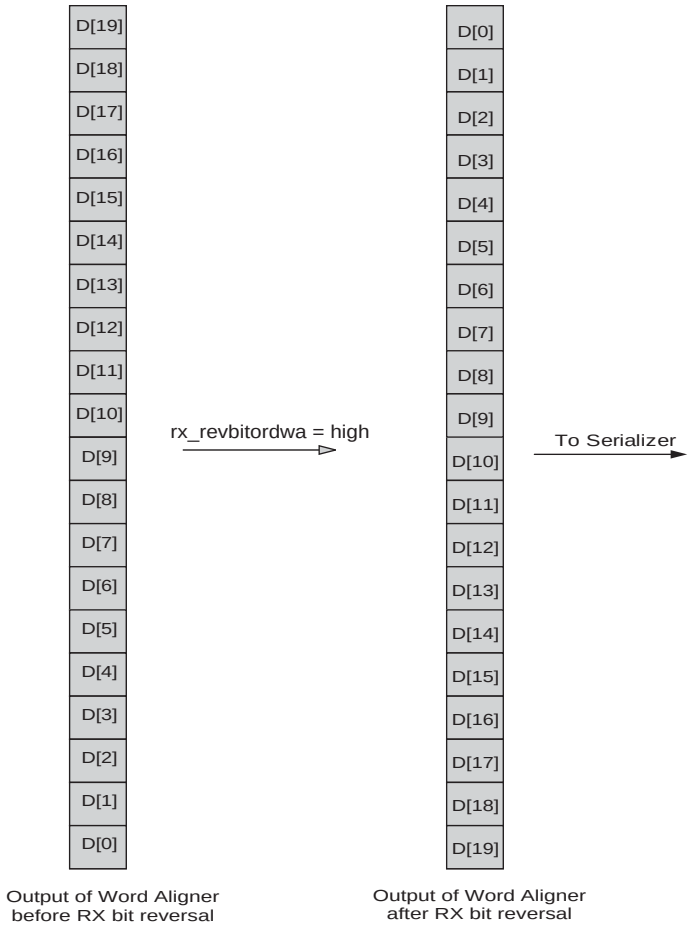


Figure 1–52 shows the receiver bit reversal feature in Basic double-width 20-bit wide datapath configurations.

Figure 1–52. Receiver Bit Reversal in Double-Width Mode



Because receiver bit reversal is done at the output of the word aligner, a dynamic bit reversal also requires a reversal of the word alignment pattern. As a result, the Receiver Bit Reversal feature is dynamic only if the receiver is dynamically reconfigurable (it allows changing the word alignment pattern dynamically) or uses manual bit slip alignment mode (no word alignment pattern). The Receiver Bit Reversal feature is static in all other Basic mode configurations. You can enable this feature using the MegaWizard Plug-In Manager. In configurations where the Receiver Bit Reversal feature is dynamic, an `rx_revbitordwa` port is available to control the bit reversal dynamically. A high on the `rx_revbitordwa` port reverses the bit order at the input of the word aligner.

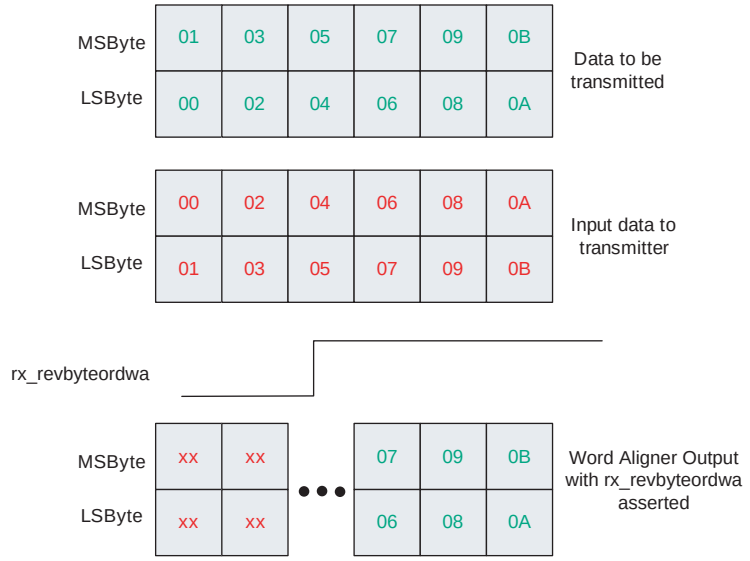
Receiver Byte Reversal in Basic Double-Width Modes

The MSByte and LSByte of the input data to the transmitter may be erroneously swapped. The receiver byte reversal feature is available to correct this situation.

An optional port, `rx_revbyteordwa`, is available only in Basic double-width mode to enable receiver byte reversal. In 8B/10B enabled mode, a high value on `rx_revbyteordwa` exchanges the 10-bit MSByte for the LSByte of the 20-bit word at the output of the word aligner in the receiver datapath. In non-8B/10B enabled mode, a high value on `rx_revbyteordwa` exchanges the 8-bit MSByte for the LSByte of the 16-bit word at the output of the word aligner in the receiver datapath. This compensates for the erroneous exchanging at the transmitter and corrects the data received by the downstream systems. `rx_revbyteorderwa` is a dynamic signal and can cause an initial disparity error at the receiver of an 8B/10B encoded link. The downstream system must be able to tolerate this disparity error.

Figure 1–53 shows the receiver byte reversal feature.

Figure 1–53. Receiver Byte Reversal Feature



Deskew FIFO

Code groups received across four lanes in a XAUI link can be misaligned with respect to one another because of skew in the physical medium or differences between the independent clock recoveries per lane. The XAUI protocol allows a maximum skew of 40 UI (12.8 ns) as seen at the receiver of the four lanes.

XAUI protocol requires the physical layer device to implement a deskew circuitry to align all four channels. To enable the deskew circuitry at the receiver to align the four channels, the transmitter sends a /A/ (/K28.3/) code group simultaneously on all four channels during inter-packet gap (IPG). The skew introduced in the physical medium and the receiver channels can cause the /A/ code groups to be received misaligned.

Deskew circuitry performs the deskew operation by the XAUI functional mode. Deskew circuitry consists of:

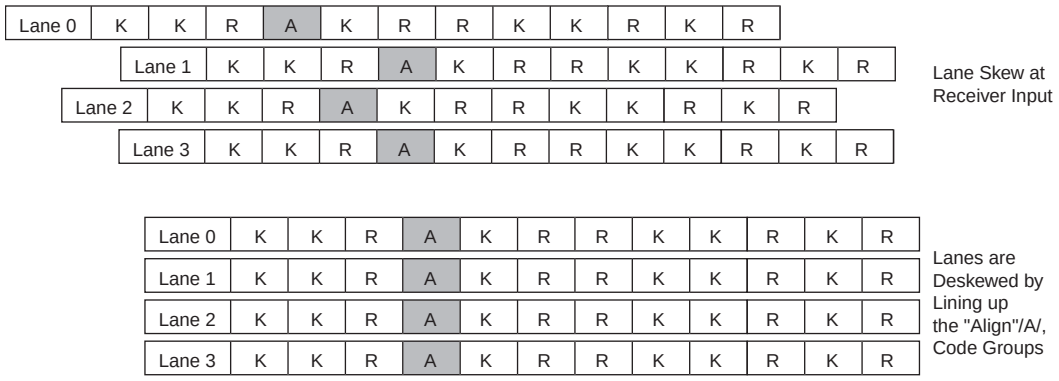
- A 16-word deep deskew FIFO in each of the four channels
- Control logic in the `CMU0` channel of the transceiver block that controls the deskew FIFO write and read operations in each channel

Deskew circuitry is only available in XAUI mode.

The deskew FIFO in each channel receives data from its word aligner. The deskew operation begins only after link synchronization is achieved on all four channels as indicated by a high level on the `rx_syncstatus` signal from the word aligner in each channel. Until the first /A/ code group is received, the deskew FIFO read and write pointers in each channel are not incremented. After the first /A/ code group is received, the write pointer starts incrementing for each word received but the read pointer is frozen. If the /A/ code group is received on each of the four channels within 10 recovered clock cycles of each other, the read pointer for all four deskew FIFOs is released simultaneously, aligning all four channels.

Figure 1–54 shows lane skew at the receiver input and how the deskew FIFO uses the /A/ code group to align the channels.

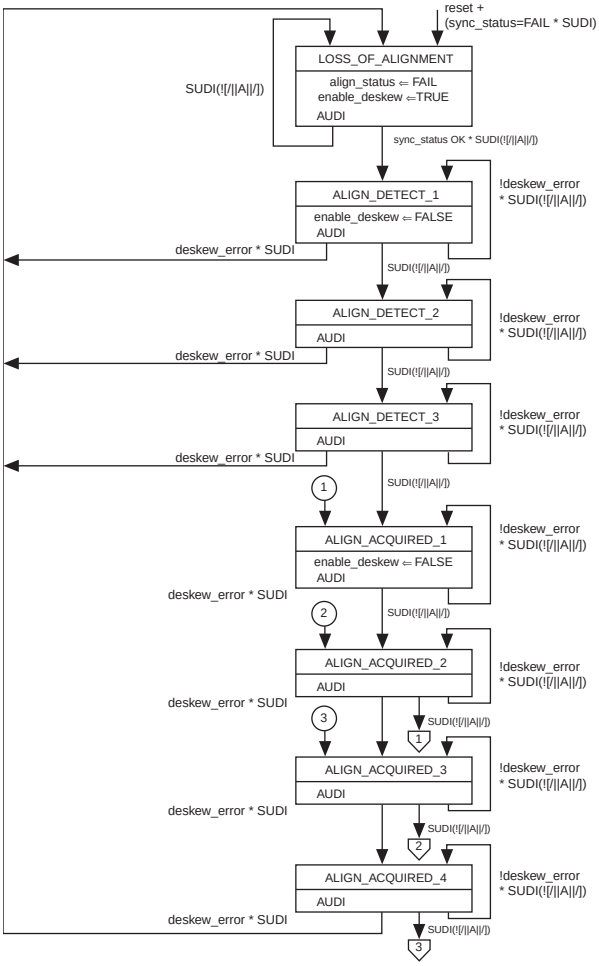
Figure 1–54. Deskew FIFO—Lane Skew at the Receiver Input



After alignment of the first ||A|| column, if three additional aligned ||A|| columns are observed at the output of the deskew FIFOs of the four channels, the `rx_channelaligned` signal is asserted high, indicating channel alignment is acquired. After acquiring channel alignment, if four misaligned ||A|| columns are seen at the output of the deskew FIFOs in all four channels with no aligned ||A|| columns in between, the `rx_channelaligned` signal is de-asserted low, indicating loss-of-channel alignment.

The deskew operation in XAUI functional mode is compliant to the PCS deskew state machine diagram specified in clause 48 of IEEE P802.3ae, as shown in Figure 1–55.

Figure 1–55. Deskew FIFO Operation in XAUI Functional Mode *(Note 1)*



Note to Figure 1–55:

(1) This figure is from IEEE P802.3ae.

Rate Match (Clock Rate Compensation) FIFO

In asynchronous systems, the upstream transmitter and local receiver can be clocked with independent reference clocks. Frequency differences in the order of a few hundred PPM can corrupt the data when latching from the recovered clock domain (the same clock domain as the upstream transmitter reference clock) to the local receiver reference clock domain.

The rate match FIFO compensates for small clock frequency differences between the upstream transmitter and the local receiver clocks by inserting or removing SKP symbols or ordered sets from the IPG or idle streams. It deletes SKP symbols or ordered sets when the upstream transmitter reference clock frequency is higher than the local receiver reference clock frequency. It inserts SKP symbols or ordered-sets when the local receiver reference clock frequency is higher than the upstream transmitter reference clock frequency.

The rate match FIFO consists of a 20-word deep FIFO and necessary logic that controls insertion and deletion of a skip character or ordered set, depending on the PPM difference.

The rate match FIFO is mandatory and cannot be bypassed in the following functional modes:

- PCI Express (PIPE)
- XAUI
- GIGE

The rate match FIFO is optional in the following functional modes:

- Basic single-width
- Basic double-width
- SRIO

The rate match FIFO receives data from the word aligner (non-XAUI functional modes) or deskew FIFO (XAUI functional mode) in the receiver datapath. It provides the following status signals forwarded to the FPGA fabric:

- `rx_rmfifodatainserted`—indicates insertion of a skip character or ordered set
- `rx_rmfifodatadeleted`—indicates deletion of a skip character or ordered set
- `rx_rmfifofull`—indicates rate match FIFO full condition
- `rx_rmfifoempty`—indicates rate match FIFO empty condition

The rate match FIFO status signals are not available in PCI Express (PIPE) mode. These signals are encoded on the `pipestatus[2:0]` signal in PCI Express (PIPE) mode as specified in the PCI Express (PIPE) specification.

Rate Match FIFO in PCI Express (PIPE) Mode

In PCI Express (PIPE) mode, the rate match FIFO is capable of compensating up to ± 300 PPM (total 600 PPM) difference between the upstream transmitter and the local receiver. The PCI Express (PIPE) protocol requires the transmitter to send SKP ordered sets during IPGs, adhering to rules listed in the base specification. The SKP ordered set is defined as a /K28.5/ COM symbol followed by three consecutive /K28.0/ SKP symbol groups. The PCI Express (PIPE) protocol requires the receiver to recognize a SKP ordered set as a /K28.5/ COM symbol followed by one to five consecutive /K28.0/ SKP symbols.

The rate match FIFO operation is compliant to PCI Express (PIPE) Base Specification 2.0. The rate match operation begins after the synchronization state machine in the word aligner indicates synchronization is acquired by driving the `rx_syncstatus` signal high. The rate match FIFO looks for the SKP ordered set and deletes or inserts SKP symbols as necessary to prevent the rate match FIFO from overflowing or under-running.

The rate match FIFO inserts or deletes only one SKP symbol per SKP ordered set received. The rate match FIFO insertion and deletion events are communicated to the FPGA fabric on the `pipestatus[2:0]` port from each channel. The `pipestatus[2:0]` signal is driven to 3'b001 for one clock cycle synchronous to the /K28.5/ COM symbol of the SKP ordered set in which the /K28.0/ SKP symbol is inserted. The `pipestatus[2:0]` signal is driven to 3'b010 for one clock cycle synchronous to the /K28.5/ COM symbol of the SKP ordered set from which the /K28.0/ SKP symbol is deleted.

Figure 1–56 shows an example of rate match deletion in the case where two /K28.0/ SKP symbols are required to be deleted. Only one /K28.0/ SKP symbol is deleted per SKP ordered set received.

Figure 1–56. Rate Match Deletion in PCI Express (PIPE) Mode

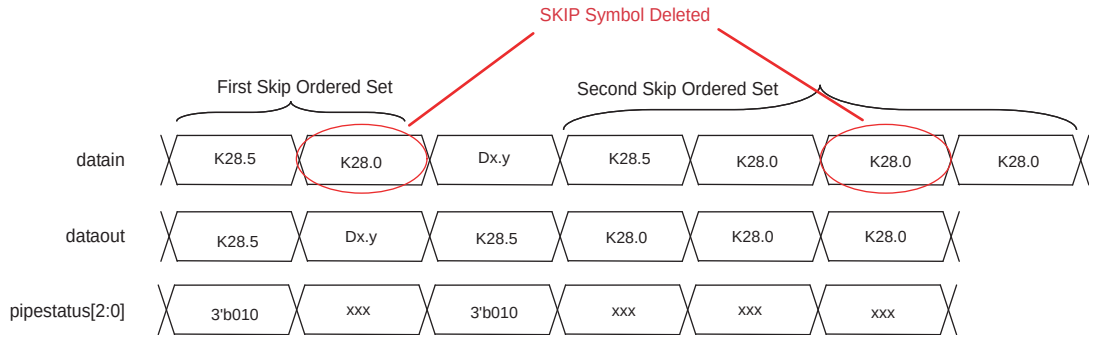
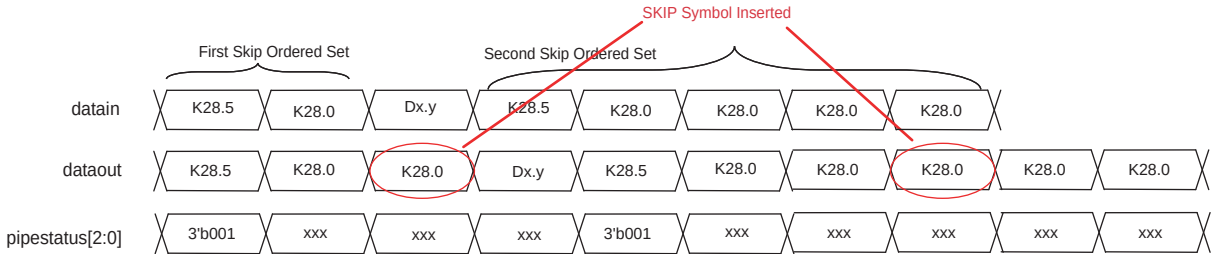


Figure 1–57 shows an example of rate match insertion in the case where two SKP symbols are required to be inserted. Only one /K28.0/ SKP symbol is inserted per SKP ordered set received.

Figure 1–57. Rate Match Insertion in PCI Express (PIPE) Mode

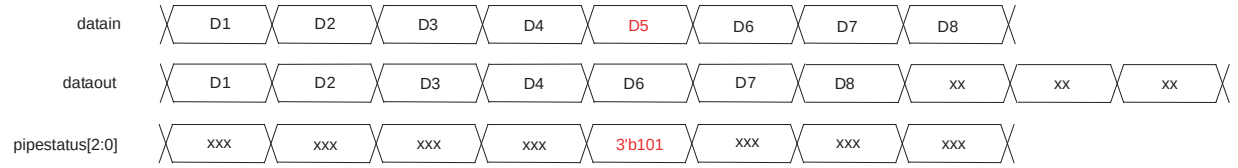


The rate match FIFO full and empty conditions are communicated to the FPGA fabric on the `pipestatus[2:0]` port from each channel.

The rate match FIFO in PCI Express (PIPE) mode automatically deletes the data byte that causes the FIFO to go full and drives `pipestatus[2:0] = 3'b101` synchronous to the subsequent data byte.

Figure 1–58 shows the rate match FIFO full condition in PCI Express (PIPE) mode. The rate match FIFO becomes full after receiving data byte D4.

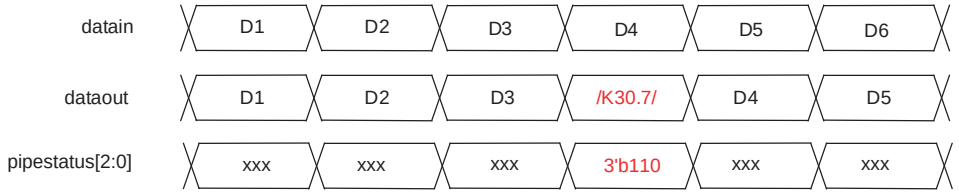
Figure 1–58. Rate Match FIFO Full Condition in PCI Express (PIPE) Mode



The rate match FIFO automatically inserts /K30.7/ (9'h1FE) after the data byte that causes the FIFO to go empty and drives `PCI Express (PIPE)status[2:0] = 3'b110` flag synchronous to the inserted /K30.7/ (9'h1FE).

Figure 1–59 shows rate match FIFO empty condition in PCI Express (PIPE) mode. The rate match FIFO becomes empty after reading out data byte D3.

Figure 1–59. Rate Match FIFO Empty Condition in PCI Express (PIPE) Mode



You can configure the rate match FIFO in low latency mode by turning off the **Enable Rate Match FIFO** option in the ALTGX MegaWizard Plug-In Manager.

Rate Match FIFO in XAUI Mode

In XAUI mode, the rate match FIFO is capable of compensating for up to ±100 PPM (200 PPM total) difference between the upstream transmitter and the local receiver reference clock. The XAUI protocol requires the transmitter to send /R/ (/K28.0/) code groups simultaneously on all four lanes (denoted as ||R|| column) during inter-packet gaps, adhering to rules listed in the IEEE P802.3ae specification. The rate match FIFO operation in XAUI mode is compliant to the IEEE P802.3ae specification.

The rate match operation begins after:

- The synchronization state machine in the word aligner of all four channels indicates synchronization was acquired by driving its `rx_syncstatus` signal high
- The deskew FIFO block indicates alignment was acquired by driving the `rx_channelaligned` signal high

The rate match FIFO looks for the ||R|| column (simultaneous /R/ code group on all four channels) and deletes or inserts the ||R|| column to prevent the rate match FIFO from overflowing or under-running. It can insert or delete as many ||R|| columns as necessary to perform the rate match operation.

Two flags, `rx_rmfifodatadeleted` and `rx_rmfifodatainserted`, that indicate rate match FIFO deletion and insertion events, respectively, are forwarded to the FPGA fabric. If an ||R|| column is deleted, the `rx_rmfifodeleted` flag from each of the four channels goes high for one clock cycle per deleted ||R|| column. If an ||R|| column is inserted, the `rx_rmfifoinserted` flag from each of the four channels goes high for one clock cycle per inserted ||R|| column.

Figure 1–60 shows an example of rate match deletion in the case where three ||R|| columns must be deleted.

Figure 1–60. Rate Match Deletion in XAUI Mode

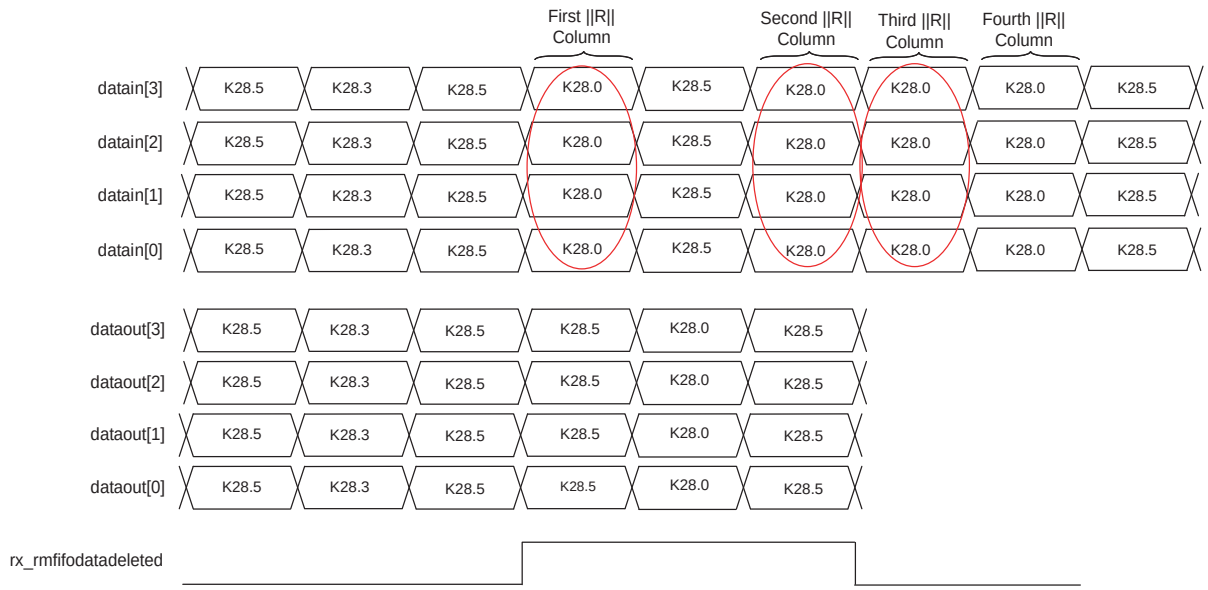
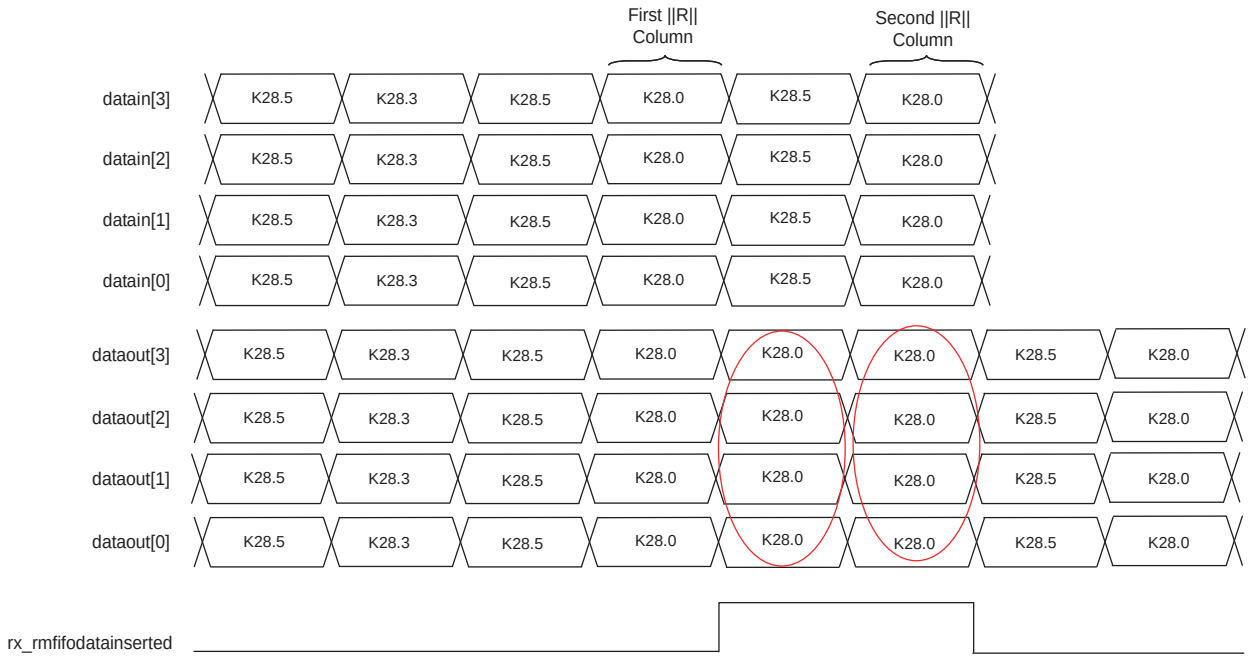


Figure 1–61 shows an example of rate match insertion in the case where two ||R|| columns are required to be inserted.

Figure 1–61. Rate Match Insertion in XAUI Mode



Two flags, `rx_rmfifofull` and `rx_rmfifoempty`, are forwarded to the FPGA fabric to indicate rate match FIFO full and empty conditions.

In XAUI mode, the rate match FIFO does not automatically insert or delete code groups to overcome FIFO empty and full conditions, respectively. It asserts the `rx_rmfifofull` and `rx_rmfifoempty` flags for at least three recovered clock cycles to indicate rate match FIFO full and empty conditions, respectively.

In the case of rate match FIFO full and empty conditions, you must assert the `rx_digitalreset` signal to reset the receiver PCS blocks.

Rate Match FIFO in GIGE Mode

In GIGE mode, the rate match FIFO is capable of compensating for up to ±100 PPM (200 PPM total) difference between the upstream transmitter and the local receiver reference clock. The GIGE protocol requires the transmitter to send idle ordered sets /I1/ (/K28.5/D5.6/) and /I2/ (/K28.5/D16.2/) during inter-packet gaps, adhering to rules listed in the IEEE 802.3 specification.

The rate match operation begins after the synchronization state machine in the word aligner indicates synchronization is acquired by driving the `rx_syncstatus` signal high. The rate match FIFO is capable of deleting or inserting the /I2/ (/K28.5/D16.2/) ordered set to prevent the rate match FIFO from overflowing or under-running during normal packet transmission. The rate match FIFO is also capable of deleting or inserting the first two bytes of the /C2/ ordered set (/K28.5/D2.2/Dx.y/Dx.y/) to prevent the rate match FIFO from overflowing or under-running during the auto negotiation phase.

The rate match FIFO can insert or delete as many /I2/ or /C2/ (first two bytes) as necessary to perform the rate match operation.

Two flags, `rx_rmfifodatadeleted` and `rx_rmfifodatainserted`, that indicate rate match FIFO deletion and insertion events, respectively, are forwarded to the FPGA fabric. Both the `rx_rmfifodatadeleted` and `rx_rmfifodatainserted` flags are asserted for two clock cycles for each deleted and inserted /I2/ ordered set, respectively.

Figure 1–62 shows an example of rate match FIFO deletion in the case where three symbols are required to be deleted. Because the rate match FIFO can only delete /I2/ ordered set, it deletes two /I2/ ordered sets (four symbols deleted).

Figure 1–62. Rate Match Deletion in GIGE Mode

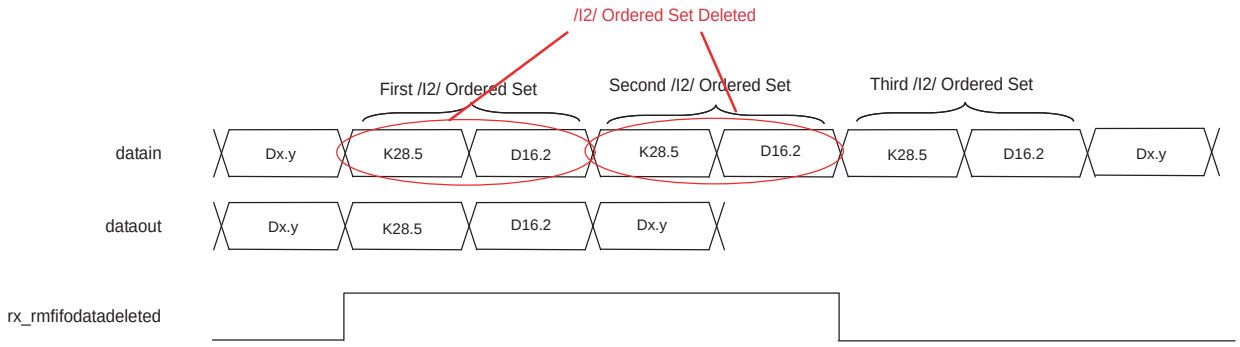
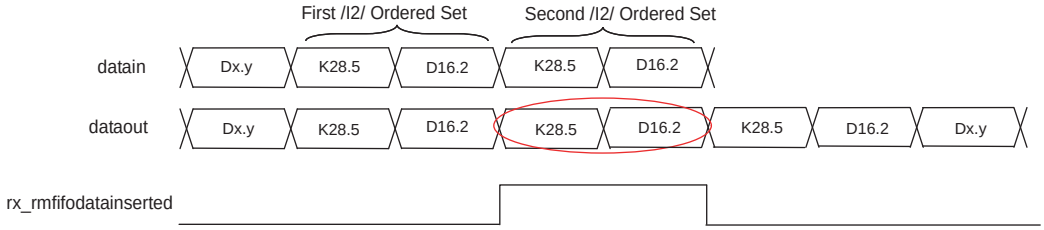


Figure 1–63 shows an example of rate match FIFO insertion in the case where one symbol is required to be inserted. Because the rate match FIFO can only insert a /I2/ ordered set, it inserts one /I2/ ordered set (two symbols inserted).

Figure 1–63. Rate Match Insertion in GIGE Mode



Two flags, `rx_rmfifofull` and `rx_rmfifoempty`, are forwarded to the FPGA fabric to indicate rate match FIFO full and empty conditions.

In GIGE mode, the rate match FIFO does not insert or delete code groups automatically to overcome FIFO empty and full conditions, respectively. It asserts the `rx_rmfifofull` and `rx_rmfifoempty` flags for at least two recovered clock cycles to indicate rate match FIFO full and empty conditions, respectively.

In the case of rate match FIFO full and empty conditions, you must assert the `rx_digitalreset` signal to reset the receiver PCS blocks.

Rate Match FIFO in Basic Single-Width Mode

In Basic single-width mode, the rate match FIFO is capable of compensating for up to ±300 PPM (600 PPM total) difference between the upstream transmitter and the local receiver reference clock.

To enable the rate match FIFO in Basic single-width mode, the transceiver channel must have both the transmitter and receiver channel instantiated. You must select the **Receiver and Transmitter** option in the **What is the operation mode?** field in the ALTGX MegaWizard Plug-In Manager. You must also enable the 8B/10B encoder/decoder in Basic single-width mode with rate match FIFO enabled.

Depending on your proprietary protocol implementation, you can select two 20-bit rate match patterns in the ALTGX MegaWizard Plug-In Manager under the **What is the rate match pattern1** and **What is the rate match pattern2** fields. Each of the two programmed 20-bit rate match patterns consists of a 10-bit skip pattern and a 10-bit control pattern. You must choose 10-bit code groups that have a neutral disparity as the skip patterns. The rate match FIFO operation begins after the word aligner synchronization status `rx_syncstatus` goes high. When the rate matcher receives either of the two 10-bit control patterns followed by the respective 10-bit skip pattern, it inserts or deletes the 10-bit skip pattern as necessary to avoid the rate match FIFO from overflowing or under-running.

The rate match FIFO can delete a maximum of four skip patterns from a cluster, if there is one skip pattern left in the cluster after deletion. The rate match FIFO can insert a maximum of four skip patterns in a cluster, if there are no more than five skip patterns in the cluster after insertion. Two flags, `rx_rmfifodatadeleted` and `rx_rmfifodatainserted`, indicating rate match FIFO deletion and insertion events, respectively, are forwarded to the FPGA fabric.

Figure 1–64 shows an example of rate match FIFO deletion in the case where three skip patterns are required to be deleted. In this example, /K28.5/ is the control pattern and neutral disparity /K28.0/ is the skip pattern. The first skip cluster has a /K28.5/ control pattern followed by two /K28.0/ skip patterns. The second skip cluster has a /K28.5/ control pattern followed by four /K28.0/ skip patterns. The rate match FIFO deletes only one /K28.0/ skip pattern from the first skip cluster to maintain at least one skip pattern in the cluster after deletion. Two /K28.0/ skip patterns are deleted from the second cluster for a total of three skip patterns deletion requirement.

Figure 1–64. Rate Match Deletion in Basic Single-Width Mode

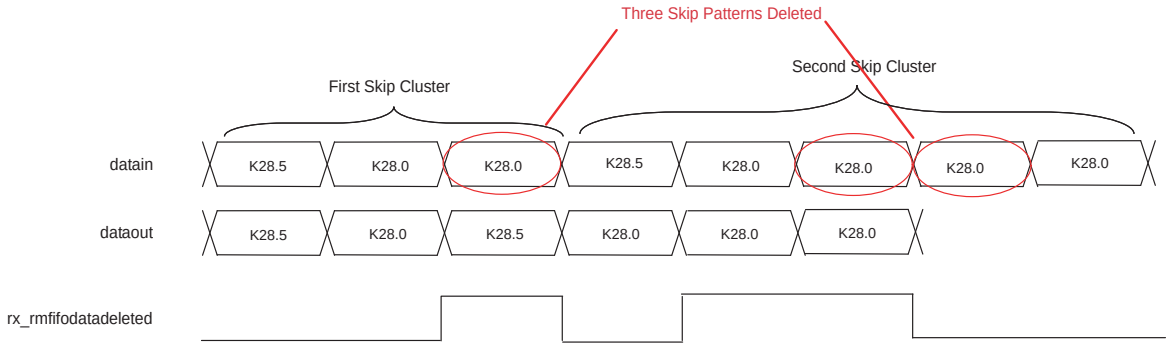
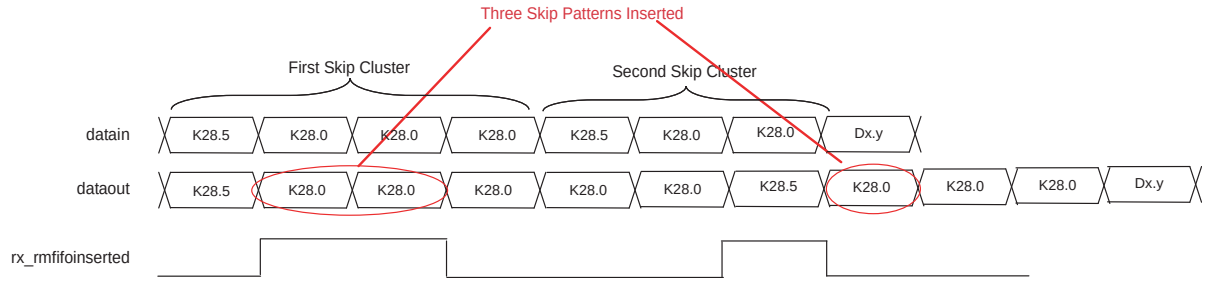


Figure 1–65 shows an example of rate match FIFO insertion in the case where three skip patterns are required to be inserted. In this example, /K28.5/ is the control pattern and neutral disparity /K28.0/ is the skip pattern. The first skip cluster has a /K28.5/ control pattern followed by three /K28.0/ skip patterns. The second skip cluster has a /K28.5/ control pattern followed by one /K28.0/ skip pattern. The rate match FIFO inserts only two /K28.0/ skip patterns into the first skip cluster to maintain a maximum of five skip patterns in the cluster after insertion. One /K28.0/ skip pattern is inserted into the second cluster for a total of three skip patterns to meet the insertion requirement.

Figure 1–65. Rate Match Insertion in Basic Single-Width Mode

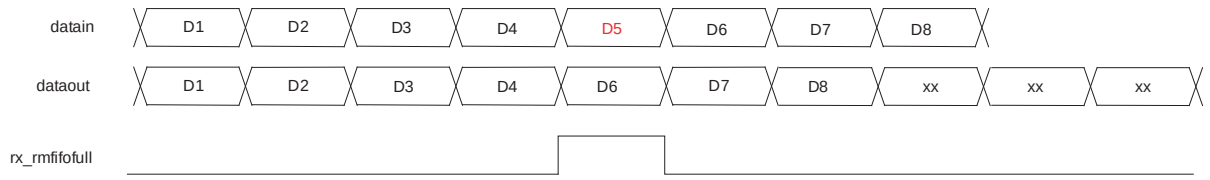


Two flags, `rx_rmfifofull` and `rx_rmfifoempty`, are forwarded to the FPGA fabric to indicate rate match FIFO full and empty conditions.

The rate match FIFO in Basic single-width mode automatically deletes the data byte that causes the FIFO to go full and asserts the `rx_rmfifofull` flag synchronous to the subsequent data byte.

Figure 1–66 shows the rate match FIFO full condition in Basic single-width mode. The rate match FIFO becomes full after receiving data byte D4.

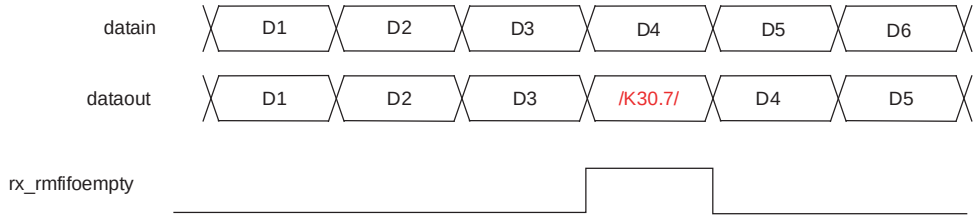
Figure 1–66. Rate Match FIFO Full Condition in Basic Single-Width Mode



The rate match FIFO automatically inserts /K30.7/ (9'h1FE) after the data byte that causes the FIFO to go empty and asserts the `rx_fifoempty` flag synchronous to the inserted /K30.7/ (9'h1FE).

Figure 1–67 shows the rate match FIFO empty condition in Basic single-width mode. The rate match FIFO becomes empty after reading out data byte D3.

Figure 1–67. Rate Match FIFO Empty Condition in Basic Single-Width Mode



Rate Match FIFO in Basic Double-Width Mode

In Basic double-width mode, the rate match FIFO is capable of compensating up to ±300 PPM (total 600 PPM total) difference between the upstream transmitter and the local receiver reference clock.

To enable the rate match FIFO in Basic double-width mode, the transceiver channel must have both the transmitter and receiver channel instantiated. You must select the **Receiver and Transmitter** option in the **What is the operation mode?** field in the ALTGX MegaWizard Plug-In Manager. You must also enable the 8B/10B encoder/decoder in Basic double-width mode with rate match FIFO enabled.

Depending on your proprietary protocol implementation, you can select two 20-bit rate match patterns in the ALTGX MegaWizard Plug-In Manager under the **What is the rate match pattern1** and **What is the rate match pattern2** fields. Each of the two programmed 20-bit rate match patterns consists of a 10-bit skip pattern and a 10-bit control pattern. You must choose 10-bit code groups that have a neutral disparity as the skip patterns. The rate match FIFO operation begins after the word aligner synchronization status `rx_syncstatus` goes high. When the rate matcher receives either of the two 10-bit control patterns followed by the respective 10-bit skip pattern, it inserts or deletes a pair of 10-bit skip patterns as necessary to avoid the rate match FIFO from overflowing or under-running.

The rate match FIFO can delete as many pairs of skip patterns from a cluster necessary to avoid the rate match FIFO from overflowing. The rate match FIFO can delete a pair of skip patterns only if the two 10-bit skip patterns appear in the same clock cycle on the LSByte and MSByte of the 20-bit word. If the two skip patterns appear straddled on the MSByte of a clock cycle and the LSByte of the next clock cycle, the rate match FIFO cannot delete the pair of skip patterns. The rate match FIFO can insert as many pairs of skip patterns into a cluster necessary to avoid the rate match FIFO from under-running. The 10-bit skip pattern can appear on MSByte or LSByte, or both, of the 20-bit word.

Two flags, `rx_rmfifodatadeleted` and `rx_rmfifodatainserted`, indicating rate match FIFO deletion and insertion events, respectively, are forwarded to the FPGA fabric.

Figure 1–68 shows an example of rate match FIFO deletion in the case where three skip patterns are required to be deleted. In this example, /K28.5/ is the control pattern and neutral disparity /K28.0/ is the skip pattern. The first skip cluster has a /K28.5/ control pattern in the LSByte and /K28.0/ skip pattern in the MSByte of a clock cycle followed by one /K28.0/ skip pattern in the LSByte of the next clock cycle. The rate match FIFO cannot delete the two skip patterns in this skip cluster because they do not appear in the same clock cycle. The second skip cluster has a /K28.5/ control pattern in the MSByte of a clock cycle followed by two pairs of /K28.0/ skip patterns in the next two cycles. The rate match FIFO deletes both pairs of /K28.0/ skip patterns (for a total of four skip patterns deleted) from the second skip cluster to meet the three skip pattern deletion requirement.

Figure 1–68. Rate Match Deletion in Basic Double-Width Mode

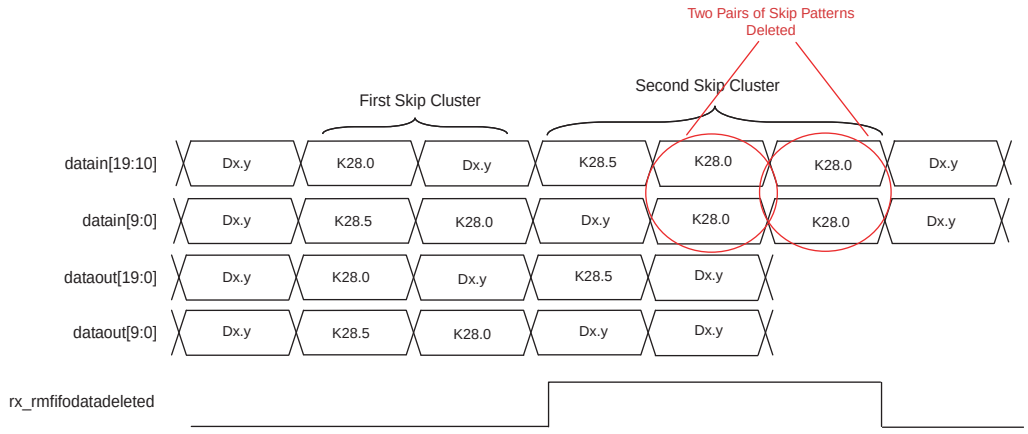
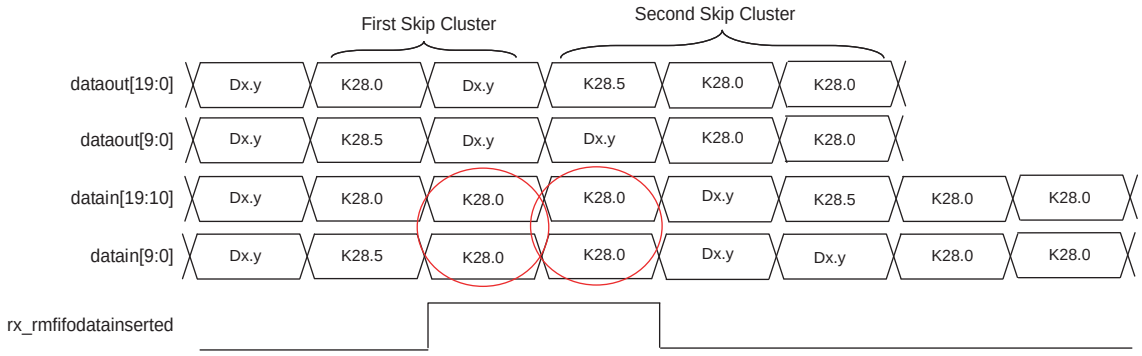


Figure 1–69 shows an example of rate match FIFO insertion in the case where three skip patterns are required to be inserted. In this example, /K28.5/ is the control pattern and neutral disparity /K28.0/ is the skip pattern. The first skip cluster has a /K28.5/ control pattern in the LSByte and /K28.0/ skip pattern in the MSByte of a clock cycle followed by one /K28.0/ skip pattern in the LSByte of the next clock cycle. The rate match FIFO inserts pairs of skip patterns in this skip cluster to meet the three skip pattern insertion requirement.

Figure 1–69. Rate Match Insertion in Basic Double-Width Mode

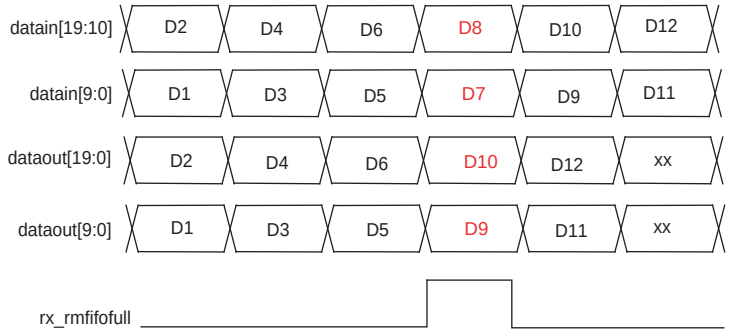


Two flags, `rx_rmfifofull` and `rx_rmfifoempty`, are forwarded to the FPGA fabric to indicate rate match FIFO full and empty conditions.

The rate match FIFO in Basic double-width mode automatically deletes the pair of data byte that causes the FIFO to go full and asserts the `rx_rmfifofull` flag synchronous to the subsequent pair of data bytes.

Figure 1–70 shows the rate match FIFO full condition in Basic double-width mode. The rate match FIFO becomes full after receiving the 20-bit word D5D6.

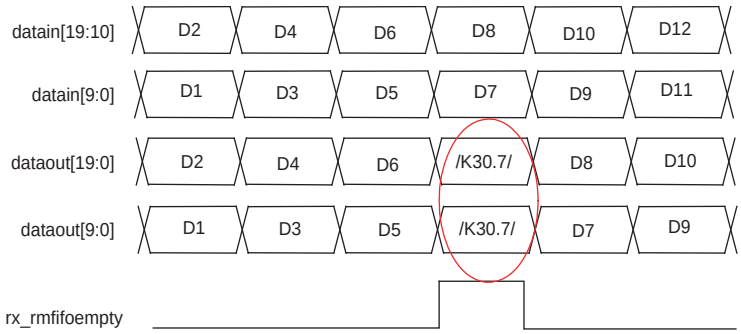
Figure 1–70. Rate Match FIFO Full Condition in Basic Double-Width Mode



The rate match FIFO automatically inserts a pair of /K30.7/ ({9'h1FE,9'h1FE}) after the data byte that causes the FIFO to go empty and asserts the `rx_fifoempty` flag synchronous to the inserted pair of /K30.7/ ({9'h1FE,9'h1FE}).

Figure 1–71 shows the rate match FIFO empty condition in Basic double-width mode. The rate match FIFO becomes empty after reading out the 20-bit word D5D6.

Figure 1–71. Rate Match FIFO Empty Condition in Basic Double-Width Mode



8B/10B Decoder

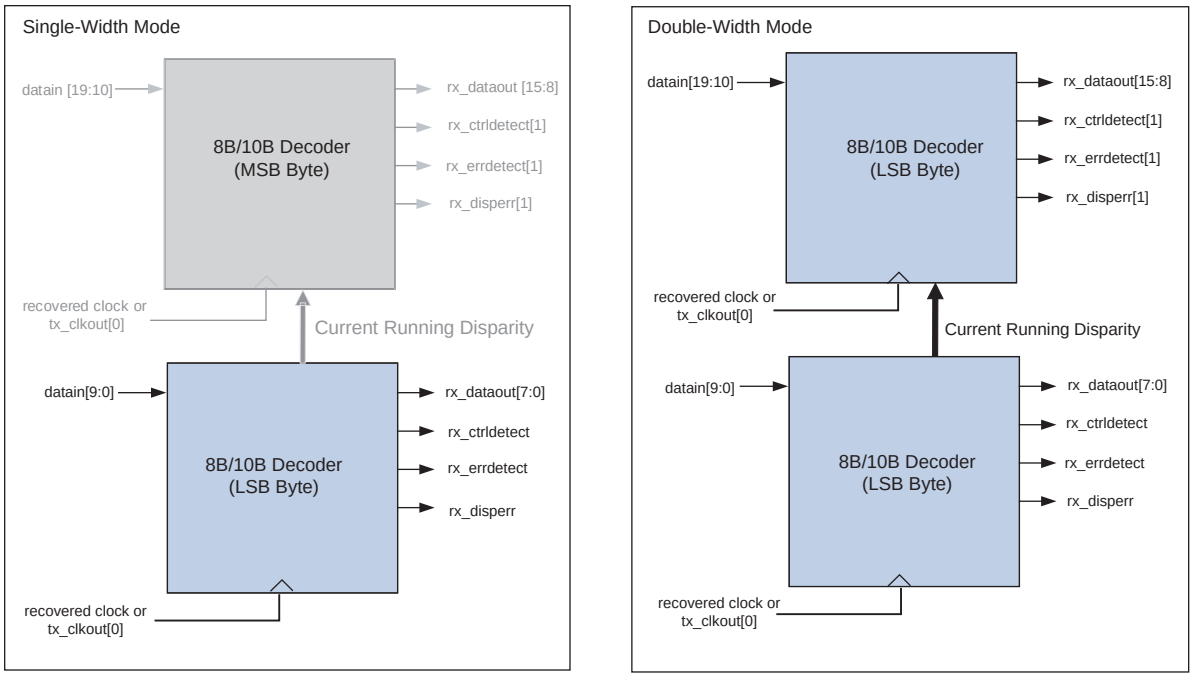
Protocols such as PCI Express (PIPE), XAUI, GIGE, and Serial RapidIO require the serial data sent over the link to be 8B/10B encoded to maintain the DC balance in the serial data transmitted. These protocols require the receiver PCS logic to implement an 8B/10B decoder to decode the data before forwarding it to the upper layers for packet processing.

The Stratix IV GX and GT receiver channel PCS datapaths implement the 8B/10B decoder after the rate matcher. In functional modes with rate matcher enabled, the 8B/10B decoder receives data from the rate matcher. In functional modes with rate matcher disabled, the 8B/10B decoder receives data from the word aligner.

The 8B/10B decoder operates in two modes (Figure 1–72):

- Single-width mode
- Double-width mode

Figure 1–72. 8B/10B Decoder in Single-Width and Double-Width Mode



8B/10B Decoder in Single-Width Mode

The left side of Figure 1–72 shows the 8B/10B decoder in single-width mode. In this mode, the 8B/10B decoder receives 10-bit data from the rate matcher or word aligner (when rate matcher is disabled) and decodes it into an 8-bit data + 1-bit control identifier. The decoded data is fed to the byte deserializer or the receiver phase compensation FIFO (if byte deserializer is disabled).

The 8B/10B decoder is compliant to Clause 36 in the IEEE802.3 specification.

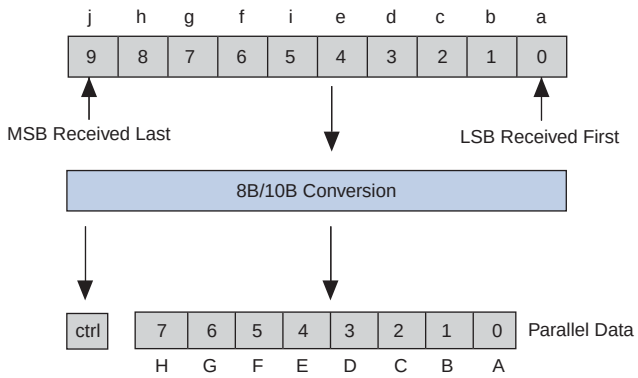
The 8B/10B decoder operates in single-width mode in the following functional modes:

- PCI Express (PIPE)
- XAUI
- GIGE
- Serial RapidIO
- Basic single-width

For PCI Express (PIPE), XAUI, GIGE, and Serial RapidIO functional modes, the ALTGX MegaWizard Plug-In Manager forces selection of the 8B/10B decoder in the receiver datapath. In Basic single-width mode, it allows you to enable or disable the 8B/10B decoder depending on your proprietary protocol implementation.

Figure 1–73 shows a 10-bit code group decoded into an 8-bit data and a 1-bit control identifier by the 8B/10B decoder in single-width mode.

Figure 1–73. 8B/10B Decoder in Single-Width Mode

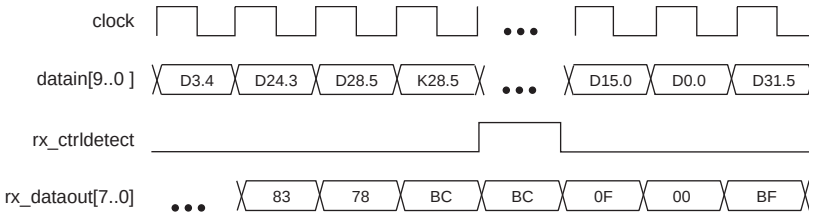


Control Code Group Detection

The 8B/10B decoder indicates whether the decoded 8-bit code group is a data or control code group on the `rx_ctrldetect` port. If the received 10-bit code group is one of the 12 control code groups (/Kx.y/) specified in the IEEE802.3 specification, the `rx_ctrldetect` signal is driven high. If the received 10-bit code group is a data code group (/Dx.y/), the `rx_ctrldetect` signal is driven low.

Figure 1–74 shows the 8B/10B decoder decoding the received 10-bit /K28.5/ control code group into an 8-bit data code group (8'hBC) driven on the `rx_dataout` port. The `rx_ctrldetect` signal is asserted high synchronous with 8'hBC on the `rx_dataout` port, indicating that it is a control code group. The rest of the codes received are data code groups /Dx.y/.

Figure 1–74. 8B/10B Decoder in Control Code Group Detection



8B/10B Decoder in Double-Width Mode

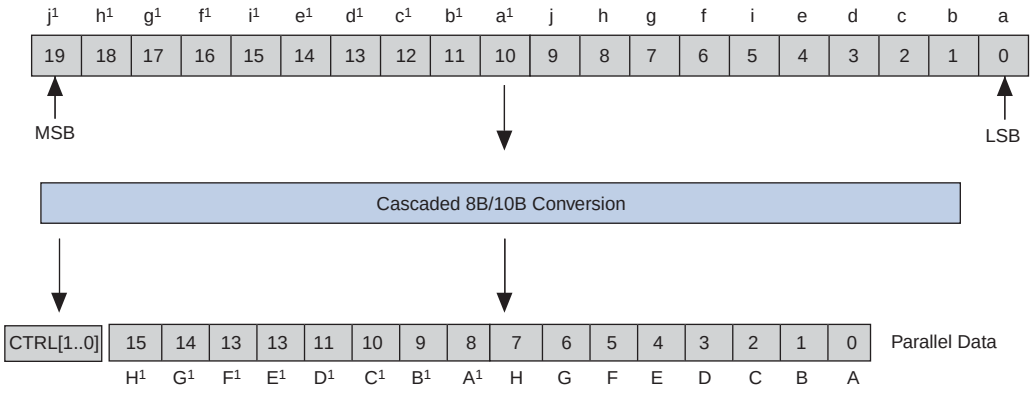
The left side of Figure 1–72 on page 1–86 shows the 8B/10B decoder in double-width mode. In this mode, two 8B/10B decoders are cascaded for decoding the 20-bit encoded data, as shown in Figure 1–75. The 10-bit LSByte of the received 20-bit encoded data is decoded first and the ending running disparity is forwarded to the 8B/10B decoder responsible for decoding the 10-bit MSByte. The cascaded 8B/10B decoder decodes the 20-bit encoded data into 16-bit data + 2-bit control identifier. The MSB and LSB of the 2-bit control identifier corresponds to the MSByte and LSByte of the 16-bit decoded data code group. The decoded data is fed to the byte deserializer or the receiver phase compensation FIFO (if byte deserializer is disabled).

Each of the two cascaded 8B/10B decoders is compliant to Clause 36 in the IEEE802.3 specification.

The 8B/10B decoder operates in double-width mode only in Basic double-width functional mode. You can enable or disable the 8B/10B decoder depending on your proprietary protocol implementation.

Figure 1–75 shows a 20-bit code group decoded into 16-bit data and 2-bit control identifier by the 8B/10B decoder in double-width mode.

Figure 1–75. 8B/10 Decoder in 20-Bit Double-Width Mode

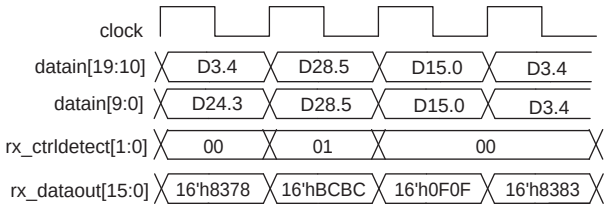


Control Code Group Detection

The cascaded 8B/10B decoder indicates whether the decoded 16-bit code group is a data or control code group on the 2-bit `rx_ctrldetect[1:0]` port. The `rx_ctrldetect[0]` signal is driven high or low depending on whether decoded data on the `rx_dataout[7:0]` port (LSByte) is a control or data code group, respectively. The `rx_ctrldetect[1]` signals are driven high or low depending on whether decoded data on the `rx_dataout[15:8]` port (MSByte) is a control or data code group, respectively.

Figure 1–76 shows the 8B/10B decoding of the received 10-bit /K28.5/ control code group into 8-bit data code group (8'hBC) driven on the `rx_dataout` port. The `rx_ctrldetect` signal is asserted high synchronous with 8'hBC on the `rx_dataout` port, indicating that it is a control code group. The rest of the codes received are data code groups /Dx.y/.

Figure 1–76. 8B/10B Decoder 10-Bit Control Code Group



Byte Deserializer

The FPGA fabric-transceiver interface frequency has an upper limit that is stated in the "Interface Frequency" section in the *DC and Switching Characteristics* chapter. In functional modes that have a receiver PCS frequency greater than the upper limit stated in the *DC and Switching Characteristics* chapter, the parallel received data and status signals cannot be forwarded directly to the FPGA fabric because it violates this upper limit for the FPGA fabric-transceiver interface frequency. In such configurations, the byte deserializer is required to reduce the FPGA fabric-transceiver interface frequency to half while doubling the parallel data width. For example, at 3.2 Gbps data rate with a deserialization factor of 10, the receiver PCS datapath runs at 320 MHz. The 10-bit parallel received data and status signals at 320 MHz cannot be forwarded to the FPGA fabric because it violates the upper limit for the FPGA fabric-transceiver interface frequency. The byte serializer converts the 10-bit parallel received data at 320 MHz into 20-bit parallel data at 160 MHz before forwarding to the FPGA fabric.

☞ The byte deserializer is required in configurations that exceed the FPGA fabric-transceiver interface clock upper frequency limit. It is optional in configurations that do not exceed the FPGA fabric-transceiver interface clock upper frequency limit.

The byte deserializer operates in two modes:

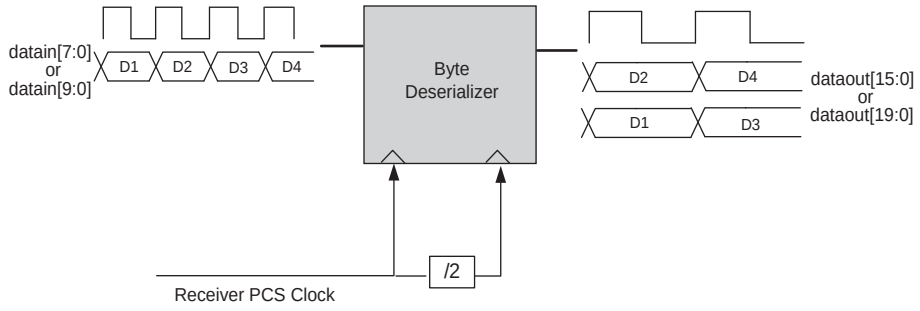
- Single-width mode
- Double-width mode

Byte Deserializer in Single-Width Mode

In single-width mode, the byte deserializer receives 8-bit wide data from the 8B/10B decoder or 10-bit wide data from the word aligner (if the 8B/10B decoder is disabled) and deserializes it into 16-bit or 20-bit wide data at half the speed.

Figure 1–77 shows the byte deserializer in single-width mode.

Figure 1–77. Byte Deserializer in Single-Width Mode

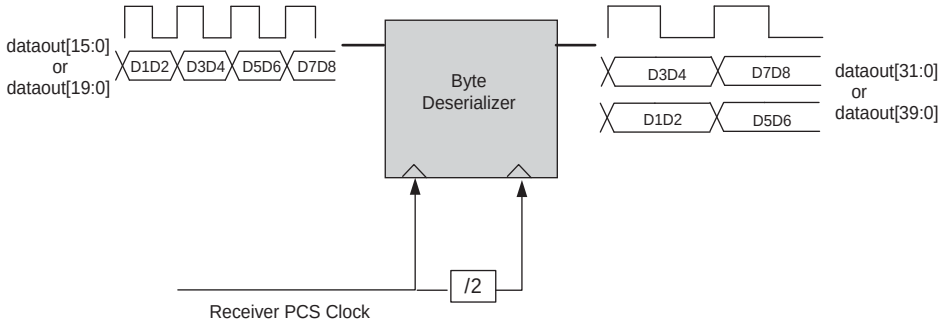


Byte Deserializer in Double-Width Mode

In double-width mode, the byte deserializer receives 16-bit wide data from the 8B/10B decoder or 20-bit wide data from the word aligner (if the 8B/10B decoder is disabled) and deserializes it into 32-bit or 40-bit wide data at half the speed.

Figure 1–78 shows the byte deserializer in double-width mode.

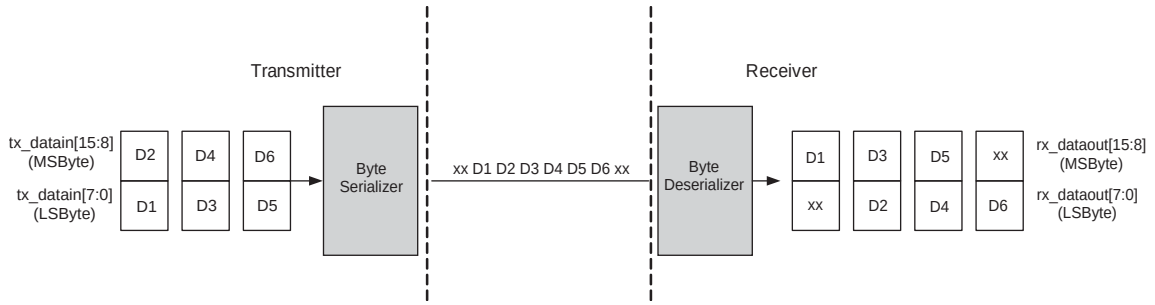
Figure 1–78. Byte Deserializer in Double-Width Mode



Byte Ordering Block

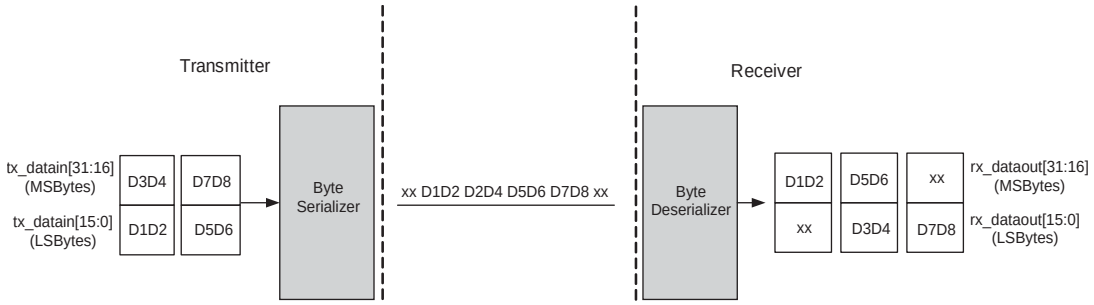
In single-width modes with the 16-bit or 20-bit FPGA fabric-transceiver interface, the byte deserializer receives one data byte (8 or 10 bits) and deserializes it into two data bytes (16 or 20 bits). Depending on when the receiver PCS logic comes out of reset, the byte ordering at the output of the byte deserializer may or may not match the original byte ordering of the transmitted data. The byte misalignment resulting from byte deserialization is unpredictable because it depends on which byte is being received by the byte deserializer when it comes out of reset. Figure 1–79 shows a scenario in which the MSByte and LSByte of the two-byte transmitter data appears straddled across two word boundaries after getting byte deserialized at the receiver.

Figure 1–79. MSByte and LSByte of the Two-Bit Transmitter Data Straddled Across Two Word Boundaries



In double-width modes with the 32-bit or 40-bit FPGA fabric-transceiver interface, the byte deserializer receives two data bytes (16 or 20 bits) and deserializes it into four data bytes (32 or 40 bits). Figure 1–80 shows a scenario in which the two MSBytes and LSBytes of the four-byte transmitter data appears straddled across two word boundaries after getting byte deserialized at the receiver.

Figure 1–80. MSByte and LSByte of the Four-Bit Transmitter Data Straddled Across Two Word Boundaries



Stratix IV GX and GT transceivers have an optional byte ordering block in the receiver datapath that you can use to restore proper byte ordering before forwarding the data to the FPGA fabric. The byte ordering block looks for the user-programmed byte ordering pattern in the byte-deserialized data. You must select a byte ordering pattern that you know appears at the LSByte(s) position of the parallel transmitter data. If the byte ordering block finds the programmed byte ordering pattern in the MSByte(s) position of the byte-deserialized data, it inserts the appropriate number of user-programmed PAD bytes to push the byte ordering pattern to the LSByte(s) position, thereby restoring proper byte ordering.

Byte Ordering Block in Single-Width Modes

Table 1–33 lists the single-width byte ordering block functional modes.

Table 1–33. Single Width Functional Modes for the Byte Ordering Block

Functional Modes	Descriptions
SONET/SDH OC-48	—
Basic single-width mode with:	■ 16-bit FPGA fabric-transceiver interface ■ No 8B/10B decoder (8-bit PMA-PCS interface) ■ Word aligner in manual alignment mode
Basic single-width mode with:	■ 16-bit FPGA fabric-transceiver interface ■ 8B/10B decoder ■ Word aligner in automatic synchronization state machine mode

For more information about configurations that allow the byte ordering block in the receiver datapath, refer to "Basic Single-Width Mode Configurations" on page 1–110.

The Quartus II software automatically configures the byte ordering pattern and byte ordering PAD pattern for SONET/SDH OC-48 functional mode. For more information, refer to "OC-48 Byte Ordering" on page 1–174.

In Basic single-width mode, you can program a custom byte ordering pattern and byte ordering PAD pattern in the ALTGX MegaWizard Plug-In Manager. Table 1–34 shows the byte ordering pattern length allowed in Basic single-width mode.

Table 1–34. Byte Ordering Pattern Length in Basic Single-Width Mode

Functional Mode	Byte Ordering Pattern Length	Byte Ordering PAD Pattern Length
Basic single-width mode with: ■ 16-bit FPGA fabric-transceiver interface ■ No 8B/10B decoder ■ Word aligner in manual alignment mode	8 bits	8 bits
Basic single-width mode with: ■ 16-bit FPGA fabric-transceiver interface ■ 8B/10B decoder ■ Word aligner in automatic synchronization state machine mode	9 bits *(1)*	9 bits

Note to Table 1–34:

(1) If a /Kx.y/ control code group is selected as the byte ordering pattern, the MSB of the 9-bit byte ordering pattern must be 1'b1. If a /Dx.y/ data code group is selected as the byte ordering pattern, the MSB of the 9-bit byte ordering pattern must be 1'b0. The least significant 8 bits must be the 8B/10B decoded version of the code group used for byte ordering.

Byte Ordering Block in Double-Width Modes

Table 1–35 lists the double-width byte ordering block functional modes.

Table 1–35. Double Width Functional Modes for the Byte Ordering Block

Functional Modes	Descriptions
Basic double-width mode with:	■ 32-bit FPGA fabric-transceiver interface ■ No 8B/10B decoder (16-bit PMA-PCS interface) ■ Word aligner in manual alignment mode
Basic double-width mode with:	■ 32-bit FPGA fabric-transceiver interface ■ 8B/10B decoder (20-bit PMA-PCS interface) ■ Word aligner in manual alignment mode
Basic double-width mode with:	■ 40-bit FPGA fabric-transceiver interface ■ No 8B/10B decoder (20-bit PMA-PCS interface) ■ Word aligner in manual alignment mode

For more information about configurations that allow the byte ordering block in the receiver datapath, refer to "Basic Double-Width Mode Configurations" on page 1–114.

In Basic double-width modes, you can program a custom byte ordering pattern and byte ordering PAD pattern in the ALTGX MegaWizard Plug-In Manager. Table 1–36 shows the byte ordering pattern length allowed in Basic double-width mode.

Table 1–36. Byte Ordering Pattern Length in Basic Double-Width Mode

Functional Mode	Byte Ordering Pattern Length	Byte Ordering PAD Pattern Length
Basic double-width mode with: ■ 32-bit FPGA fabric-transceiver interface ■ No 8B/10B decoder (16-bit PMA-PCS interface) ■ Word aligner in manual alignment mode	16 bits, 8 bits	8 bits
Basic double-width mode with: ■ 32-bit FPGA fabric-transceiver interface ■ 8B/10B decoder (20-bit PMA-PCS interface) ■ Word aligner in manual alignment mode	18 bits, 9 bits *(1)*	9 bits
Basic double-width mode with: ■ 40-bit FPGA fabric-transceiver interface ■ No 8B/10B decoder (20-bit PMA-PCS interface) ■ Word aligner in manual alignment mode	20 bits, 10 bits	10 bits

Note to Table 1–36:

(1) The 18-bit byte ordering pattern `D[17:0]` consists of MSByte `D[17:9]` and LSByte `D[8:0]`; `D[17]` corresponds to `rx_ctrldetect[1]` and `D[16:9]` corresponds to `rx_dataout[15:8]`. Similarly, `D[9]` corresponds to `rx_ctrldetect[0]` and `D[7:0]` corresponds to `rx_dataout[7:0]`.

The byte ordering block modes of operation in both single-width and double-width modes are:

- Word-alignment-based byte ordering
- User-controlled byte ordering

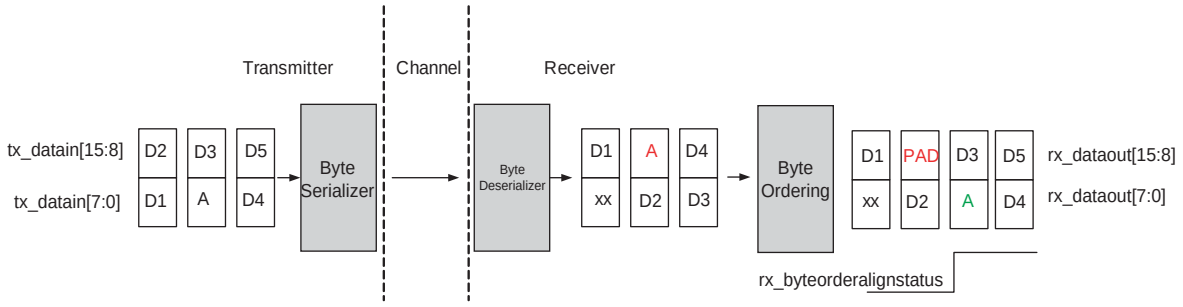
Word-Alignment-Based Byte Ordering

In word-alignment-based byte ordering, the byte ordering block starts looking for the byte ordering pattern in the byte-deserialized data every time it sees a rising edge on the `rx_syncstatus` signal. After a rising edge on the `rx_syncstatus` signal, if the byte ordering block finds the first data byte that matches the programmed byte ordering pattern in the MSByte position of the byte-deserialized data, it inserts one programmed PAD pattern to push the byte ordering pattern in the LSByte position. If the byte ordering block finds the first data byte that matches the programmed byte ordering pattern in the LSByte position of the byte-deserialized data, it considers the data to be byte ordered and does not insert any PAD pattern. In either case, the byte ordering block asserts the `rx_byteorderalignstatus` signal.

You can choose word-alignment-based byte ordering by selecting the `sync status` signal from the word aligner tab in the **What do you want the byte ordering to be based on?** field in the ALTGX MegaWizard Plug-In Manager.

Figure 1–81 shows an example of the byte ordering operation in single-width modes. In this example, A is the programmed byte ordering pattern and PAD is the programmed PAD pattern. The byte deserialized data places the byte ordering pattern A in the MSByte position, resulting in incorrect byte ordering. Assuming that a rising edge on the `rx_syncstatus` signal had occurred before the byte ordering block sees the byte ordering pattern A in the MSByte position, the byte ordering block inserts a PAD byte and pushes the byte ordering pattern A in the LSByte position. The data at the output of the byte ordering block has correct byte ordering as reflected on the `rx_byteorderalignstatus` signal.

Figure 1–81. Byte Ordering in Single-Width Modes



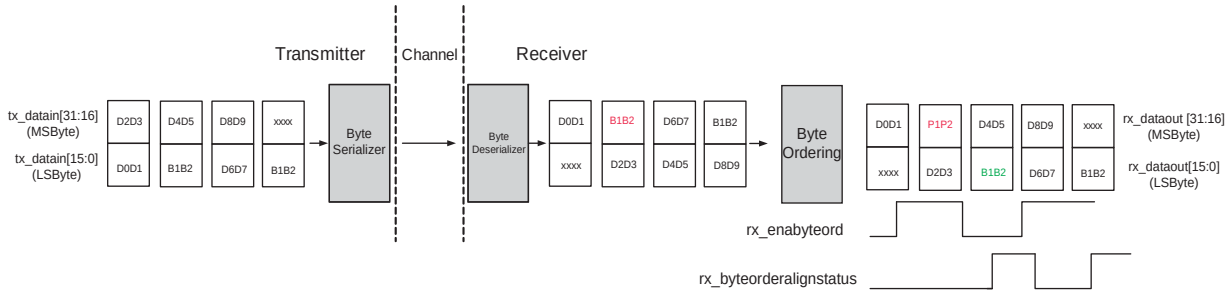
If the byte ordering block sees another rising edge on the `rx_syncstatus` signal from the word aligner, it de-asserts the `rx_byteorderalignstatus` signal and repeats the byte ordering operation as previously described.

User-Controlled Byte Ordering

Unlike word-alignment-based byte ordering, user-controlled byte ordering provides control to the user logic to restore correct byte ordering at the receiver. When enabled, an `rx_enabyteord` port is available that you can use to trigger the byte ordering operation. A rising edge on the `rx_enabyteord` port triggers the byte ordering block. After a rising edge on the `rx_enabyteord` signal, if the byte ordering block finds the first data byte that matches the programmed byte ordering pattern in the MSByte position of the byte-deserialized data, it inserts one programmed PAD pattern to push the byte ordering pattern in the LSByte position. If the byte ordering block finds the first data byte that matches the programmed byte ordering pattern in the LSByte position of the byte-deserialized data, it considers the data to be byte ordered and does not insert any PAD byte. In either case, the byte ordering block asserts the `rx_byteorderalignstatus` signal.

Figure 1–82 shows user-controlled byte ordering in Basic double-width Mode.

Figure 1–82. User-Controlled Byte Ordering in Basic Double-Width Mode



Receiver Phase Compensation FIFO

The receiver phase compensation FIFO in each channel ensures reliable transfer of data and status signals between the receiver channel and the FPGA fabric. The receiver phase compensation FIFO compensates for the phase difference between the parallel receiver PCS clock (FIFO write clock) and the FPGA fabric clock (FIFO read clock).

The receiver phase compensation FIFO operates in one of the following two modes:

- Low latency mode—The Quartus II software automatically configures the receiver phase compensation FIFO in low latency mode in all functional modes. In this mode, the FIFO is four words deep and the latency through the FIFO is two to three parallel clock cycles (pending characterization).
- High latency mode—In this mode, the FIFO is eight words deep and the latency through the FIFO is four to five parallel clock cycles (pending characterization).

The receiver phase compensation FIFO write clock source varies with the receiver channel configuration. Table 1–37 shows the receiver phase compensation FIFO write clock source in different configurations.

Table 1–37. Receiver Phase Compensation FIFO Write Clock Source

Configuration	Receiver Phase Compensation FIFO Write Clock	
	Without Byte Serializer	**With Byte Serializer**
Non-bonded channel configuration with rate matcher	Parallel transmitter PCS clock from the local clock divider in the associated channel (`tx_clkout`)	Divide-by-two version of the parallel transmitter PCS clock from the local clock divider in the associated channel (`tx_clkout`)
Non-bonded channel configuration without rate matcher	Parallel recovered clock from the receiver PMA in the associated channel (`rx_clkout`)	Divide-by-two version of the parallel recovered clock from the receiver PMA in the associated channel (`rx_clkout`)
×4 bonded channel configuration	Parallel transmitter PCS clock from the central clock divider in the `CMU0` of the associated transceiver block (`coreclkout`)	Divide-by-two version of the parallel transmitter PCS clock from the central clock divider in `CMU0` of the associated transceiver block (`coreclkout`)
×8 bonded channel configuration	Parallel transmitter PCS clock from the central clock divider in `CMU0` of the master transceiver block (`coreclkout` from master transceiver block)	Divide-by-two version of the parallel transmitter PCS clock from the central clock divider in `CMU0` of the master transceiver block (`coreclkout` from master transceiver block)

The receiver phase compensation FIFO read clock source varies depending on whether or not you instantiate the `rx_coreclk` port in the ALTGX MegaWizard Plug-In Manager. Table 1–38 shows the receiver phase compensation FIFO read clock source in different configurations.

Table 1–38. Receiver Phase Compensation FIFO Read Clock Source

Configuration	Receiver Phase Compensation FIFO Read Clock	
	rx_coreclk Port Not Instantiated	**rx_coreclk Port Instantiated *(1)***
Non-bonded channel configuration with rate matcher	FPGA fabric clock driven by the clock signal on the `tx_clkout` port	FPGA fabric clock driven by the clock signal on the `rx_coreclk` port
Non-bonded channel configuration without rate matcher	FPGA fabric clock driven by the clock signal on the `rx_clkout` port	FPGA fabric clock driven by the clock signal on the `rx_coreclk` port
×4 bonded channel configuration	FPGA fabric clock driven by the clock signal on the `coreclkout` port	FPGA fabric clock driven by the clock signal on the `rx_coreclk` port
×8 bonded channel configuration	FPGA fabric clock driven by the clock signal on the `coreclkout` port	FPGA fabric clock driven by the clock signal on the `rx_coreclk` port

Note to Table 1–38:

(1) The clock signal driven on the `rx_coreclk` port must have 0 PPM frequency difference with respect to the receiver phase compensation FIFO write clock.

Receiver Phase Compensation FIFO Error Flag

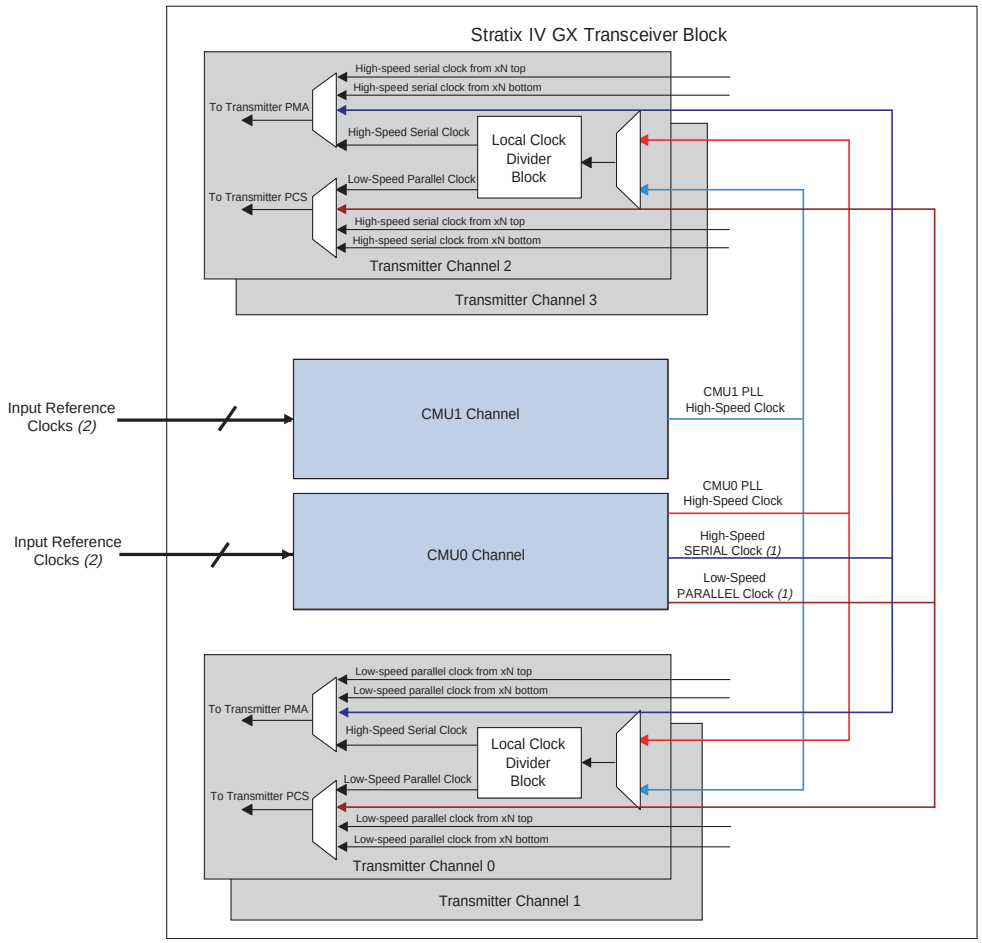
An optional `rx_phase_comp_fifo_error` port is available in all functional modes to indicate a receiver phase compensation FIFO under-run or overflow condition. The `rx_phase_comp_fifo_error` signal is asserted high when the phase compensation FIFO gets either full or empty. This feature is useful to verify a phase compensation FIFO under-run or overflow condition as a probable cause of link errors.

CMU Channel Architecture

Stratix IV GX and GT devices contain two CMU channels—`CMU0` and `CMU1`—within each transceiver block that you can configure as a transceiver channel or as a clock generation block. In addition, each CMU channel contains a CMU PLL that provides clocks to the transmitter channels within the same transceiver block.

Figure 1–83 shows the two CMU channels in a transceiver block.

Figure 1–83. CMU Channels in a Transceiver Block



Notes to Figure 1–83:

(1) Clocks are provided to support bonded channel functional mode.

(2) For more information, refer to the *Stratix IV Transceiver Clocking* chapter.

The following sections describe the CMU channel building blocks.

Configuring CMU Channels for Clock Generation

Each CMU channel has a CMU PLL that generates high-speed serial transceiver clocks when the CMU channel is configured as a CMU. The `CMU0` clock divider block also generates the low-speed parallel transceiver clock for ×4, ×8, and ×N bonded mode configurations such as XAUI, Basic ×4, Basic ×8, and Basic (PMA-Direct) ×N.

The `CMU0` channel has additional capabilities to support bonded protocol functional modes such as Basic ×4, XAUI, and PCI Express (PIPE). Use the ALTGX MegaWizard Plug-In Manager to select these functional modes (to enable Basic ×4 functional mode, select the **×4** option in Basic mode). For more information, refer to "Functional Modes" on page 1–107.

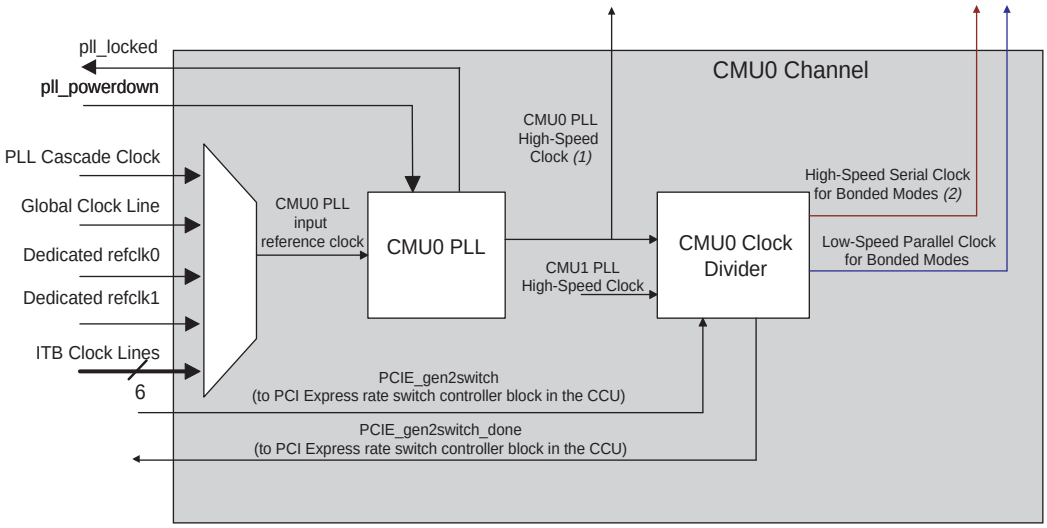
For Stratix IV GT devices, you can use the CMU PLL to generate transceiver clocks at data rates between 2.488 Gbps and 11.3 Gbps. Altera strongly recommends using 10G ATX PLLs to generate transceiver clocks at data rates between 9.9 Gbps and 11.3 Gbps.

CMU0 Channel

The `CMU0` channel, shown in Figure 1–84, contains the following blocks:

- `CMU0` PLL (refer to "CMU0 Clock Divider" on page 1–100)
- `CMU0` clock divider (refer to "CMU Clock Divider" on page 1–105)

Figure 1–84. CMU0 Channel with the CMU0 PLL and CMU0 Clock Divider



Notes to Figure 1–84:

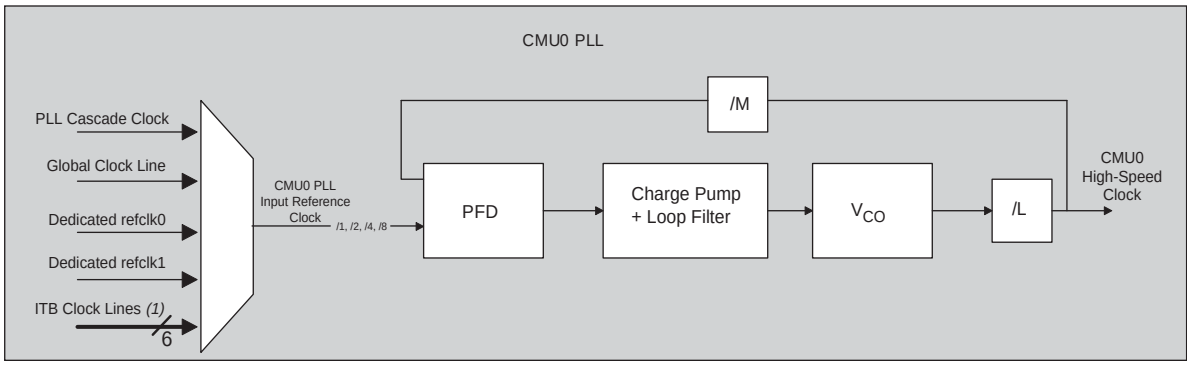
(1) To provide clocks for its PMA and PCS blocks in non-bonded functional modes (for example, GIGE functional mode), the transmitter channel uses the transmitter local clock divider to divide this high-speed clock output.

(2) Used in XAUI, Basic ×4, and PCI Express (PIPE) ×4 functional modes. In PCI Express (PIPE) ×8 functional mode, only the `CMU0` channel of the master transceiver block provides clock output to all eight transceiver channels configured in PCI Express (PIPE) functional mode.

CMU0 PLL

Figure 1–85 shows the `CMU0` PLL.

Figure 1–85. CMU0 PLL



Note to Figure 1–85:

(1) The inter transceiver block (ITB) clock lines are the maximum value. The actual number of ITB lines in your device depends on the number of transceiver blocks on one side of your device.

You can select the input reference clock to the `CMU0` PLL from multiple clock sources:

- PLL cascade clock—the output from the general purpose PLLs in the FPGA fabric
- Global clock line—the input reference clock from the dedicated CLK pins are connected to the global clock line
- `refclk0`—dedicated REFCLK in the transceiver block
- `refclk1`—dedicated REFCLK in the transceiver block
- Inter transceiver block lines—the ITB lines connect `refclk0` and `refclk1` of all other transceiver blocks on the same side of the device

The `CMU0` PLL generates the high-speed clock from the input reference clock. The PFD tracks the VCO output with the input reference clock.

For more information about transceiver input reference clocks, refer to the *Stratix IV Transceiver Clocking* chapter.

The VCO in the `CMU0` PLL is half rate and runs at half the serial data rate. To generate the high-speed clock required to support a native data rate range of 600 Mbps to 8.5 Gbps, the `CMU0` PLL uses two multiplier blocks (/M and /L) in the feedback path (shown in Figure 1–85).

The ALTGX MegaWizard Plug-In Manager provides the list of input reference clock frequencies based on the data rate you select. The Quartus II software automatically selects the /M and /L settings based on the input reference clock frequency and serial data rate.

The `CMU0` and `CMU1` PLLs have a dedicated `pll_locked` signal that is asserted to indicate that the CMU PLL is locked to the input reference clock. You can use the `pll_locked` signal in your transceiver reset sequence, as described in the *Reset Control and Power Down* chapter.

PLL Bandwidth Setting

The bandwidth of a PLL is the measure of its ability to track input clock and jitter. It is determined by the –3 dB frequency of the closed-loop gain of the PLL. There are three bandwidth settings: high, medium, and low. You can program the PLL bandwidth setting using the ALTGX MegaWizard Plug-In Manager.

- The high bandwidth setting filters out internal noise from the VCO because it tracks the input clock above the frequency of the internal VCO noise.
- With the low bandwidth setting, if the noise on the input reference clock is greater than the internal noise of the VCO, the PLL filters out the noise above the –3 dB frequency of the closed-loop gain of the PLL.
- The medium bandwidth setting is a compromise between the high and low bandwidth settings.

The –3 dB frequencies for these settings can vary because of the non-linear nature and frequency dependencies of the circuit.

Power Down CMU0 PLL

You can power down the `CMU0` PLL by asserting the `pll_powerdown` signal.

For more information, refer to the *Reset Control and Power Down* chapter.

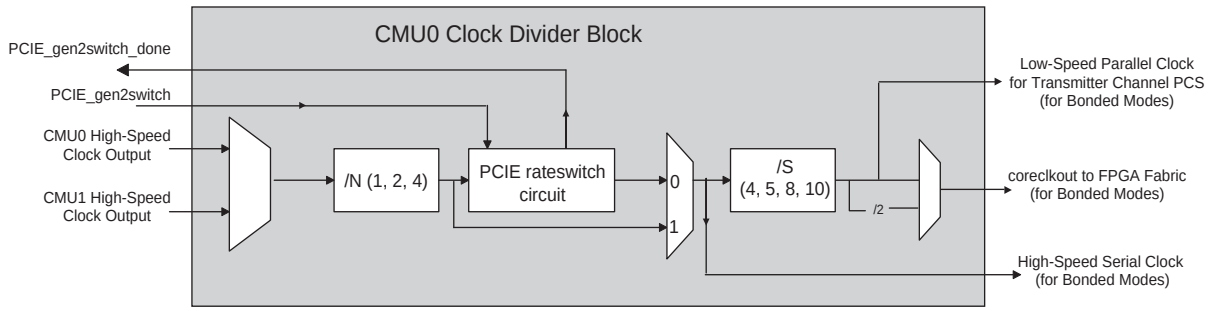
CMU0 Clock Divider

The high-speed clock output from the `CMU0` PLL is forwarded to two clock dividers: the `CMU0` clock divider and the transmitter channel local clock divider. Use the clock divider only in bonded channel functional modes. In non-bonded functional modes (such as GIGE functional mode), the local clock divider divides the high-speed clock to provide clocks for its PCS and PMA blocks. This section only describes the `CMU0` clock divider.

For more information about the local clock divider, refer to the "Transceiver Channel Datapath Clocking" section in the *Stratix IV Transceiver Clocking* chapter.

You can configure the `CMU0` clock divider shown in Figure 1–86 to select the high-speed clock output from the `CMU0` or `CMU1` PLLs. The `CMU1` PLL is present in the `CMU1` channel.

Figure 1–86. CMU0 Clock Divider



High-Speed Serial Clock Generation

The /N divider receives the high-speed clock output from one of the CMU PLLs and produces a high-speed serial clock. Use this clock for bonded functional modes such as Basic ×4/×8, XAUI, and PCI Express (PIPE) ×4/×8 configurations. In XAUI and Basic ×4/×8 modes, the Quartus II software chooses the path (shown by "1" in the MUX) and provides the high-speed serial clock to all the transmitter channels within the transceiver block.

- In PCI Express (PIPE) ×4 mode, the clock path through the PCI Express (PIPE) rateswitch circuit block is selected. This high-speed serial clock is provided to all the transmitter channels.
- In PCI Express (PIPE) ×8 mode and Basic ×8 mode, only the `CMU0` clock divider of the master transceiver block provides the high-speed serial clock to all eight channels.
- In PCI Express (PIPE) ×1 mode, the `CMU0` clock divider does not provide a high-speed serial clock. Instead, the local clock divider block in the transmitter channel receives the `CMU0` or `CMU1` PLL high-speed clock output and generates the high-speed serial clock to its serializer.

For more information about the clock from the master transceiver block, refer to the *Stratix IV Transceiver Clocking* chapter.

PCIE Rateswitch Circuit

The PCIE rateswitch circuit is enabled only in PCI Express (PIPE) ×4 mode. In PCI Express (PIPE) ×8 mode, the PCIE rateswitch circuit of the `CMU0` clock divider of the master transceiver block is active.

There are two paths in the PCIE rateswitch circuit. One path divides the /N output by two. The other path forwards the /N divider output.

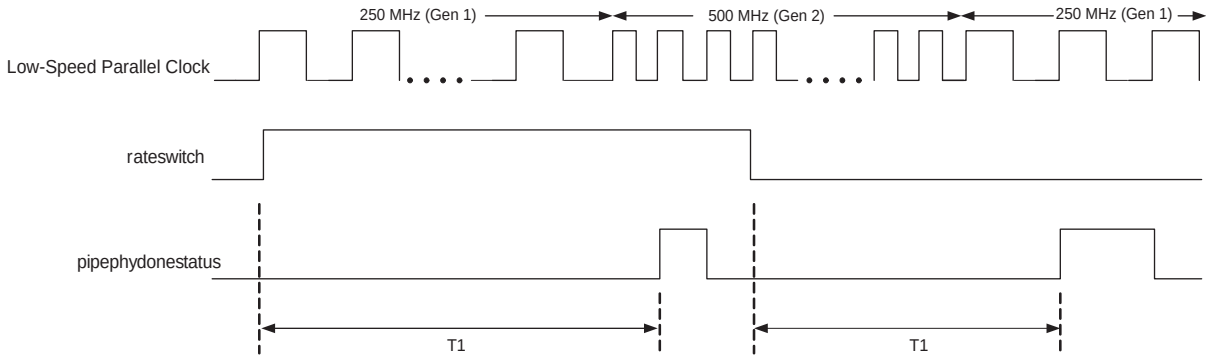
- When you set the `rateswitch` port to **0**, the PCI Express (PIPE) rateswitch controller (in the CCU) signals the PCIE rateswitch circuit to select the divide by /2 to provide a high-speed serial clock for the Gen1 (2.5 Gbps) data rate.
- When you set the `rateswitch` port to **1**, the /N divider output is forwarded, providing a high-speed serial clock for the Gen2 (5 Gbps) data rate to the transmitter channels.

The PCIE rateswitch circuit performs the rateswitch operation only for the transmitter channels. For the receiver channels, the rateswitch circuit within the receiver CDR performs the rateswitch operation.

The PCIE rateswitch circuit is controlled by the PCI Express (PIPE) rateswitch controller in the CCU. The PCI Express (PIPE) rateswitch controller asserts the `pipephydonestatus` signal for one clock cycle after the rateswitch operation is completed for both the transmit and receive channels. Figure 1–87 shows the timing diagram for the rateswitch operation.

For more information about PCI Express (PIPE) functional mode rate switching, refer to "PCI Express (PIPE) Gen2 (5 Gbps) Support" on page 1–138.

Figure 1–87. Rateswitch in PCI Express (PIPE) Mode *(Note 1)*



Note to Figure 1–87:

(1) Time T1 is pending characterization.

When creating a PCI Express (PIPE) Gen2 configuration, configure the CMU PLL to 5 Gbps. This helps to generate the 2.5 Gbps and 5 Gbps high-speed serial clock using the rateswitch circuit.

Low-Speed Parallel Clock Generation

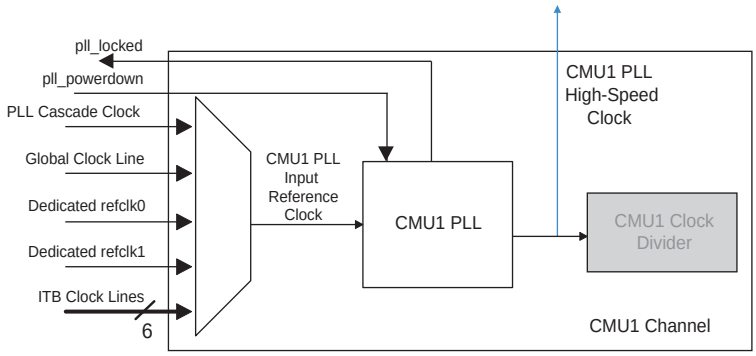
The /S divider receives the clock output from the /N divider or PCIE rateswitch circuit (only in PCI Express [PIPE] mode) and generates the low-speed parallel clock for the PCS block of all transmitter channels and `coreclkout` for the FPGA fabric. If the byte serializer block is enabled in bonded channel modes, the /S divider output is divided by the /2 divider and sent out as `coreclkout` to the FPGA fabric. The Quartus II software automatically selects the /S values based on the deserialization width setting (single-width or double-width mode) that you select in the ALTGX MegaWizard Plug-In Manager. For more information about single-width or double-width mode, refer to "Transceiver Channel Architecture" on page 1–12.

The Quartus II software automatically selects all the divider settings based on the input clock frequency, data rate, deserialization width, and channel width settings.

CMU1 Channel

The CMU1 channel, shown in Figure 1–88, contains the CMU1 PLL that provides the high-speed clock to the transmitter channels within the transceiver block. The CMU1 PLL is similar to the CMU0 PLL (refer to "CMU0 PLL" on page 1–99).

Figure 1–88. CMU1 Channel (Grayed Area Shows the Inactive Block)



The CMU1 PLL generates the high-speed clock that is only used in non-bonded functional modes. The transmitter channels within the transceiver block can receive a high-speed clock from either of the two CMU PLLs and uses local dividers to provide clocks to its PCS and PMA blocks.

For more information about using two CMU PLLs to configure transmitter channels, refer to the *Configuring Multiple Protocols and Data Rates* chapter.

Power Down CMU1 PLL

You can power down the CMU1 PLL by asserting the pll_powerdown signal.

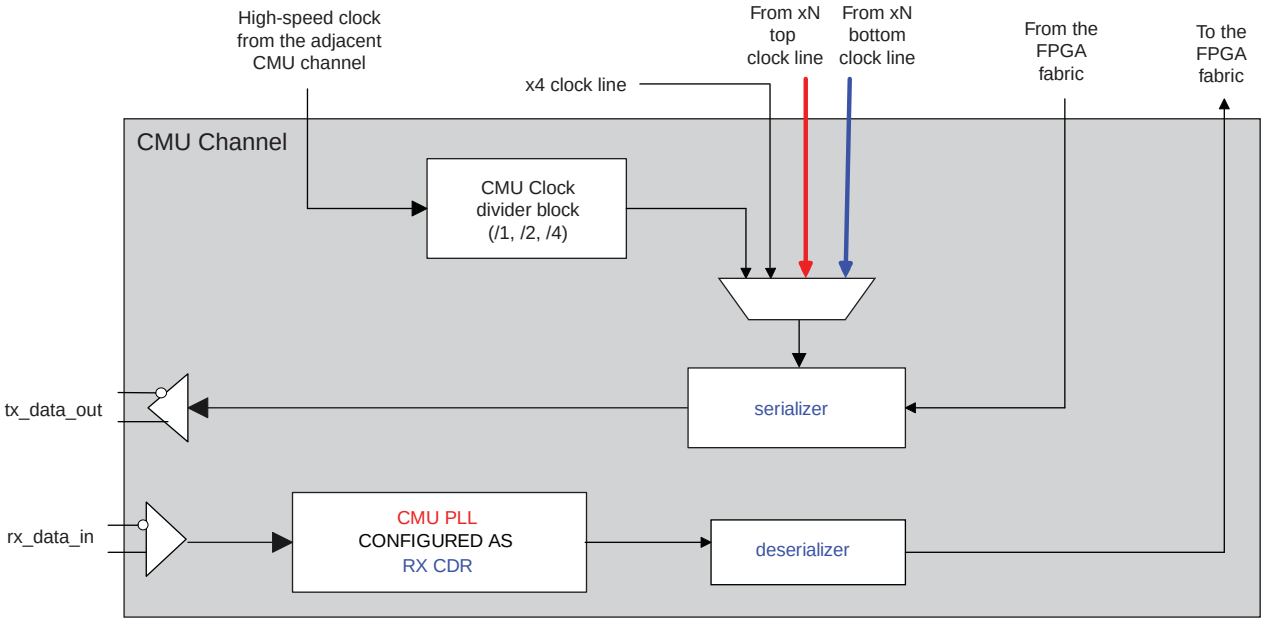
For more information, refer to the *Reset Control and Power Down* chapter.

Configuring CMU Channels as Transceiver Channels

You can configure the two CMU channels in the transceiver block of Stratix IV GX and GT devices as full-duplex PMA-only channels to run between 600 Mbps and 6.5 Gbps. For Stratix IV GT devices, you can configure both CMU0 and CMU1 channels in each transceiver block as full-duplex PMA-only channels supporting data rates between 2.488 Gbps and 6.5 Gbps.

Figure 1–89 shows the functional blocks that are enabled to support the transceiver channel functionality.

Figure 1–89. Functional Blocks Enabled to Support Transceiver Channel Functionality



The CMU PLL is configured as a CDR to recover data. The dedicated input reference clock pin is configured to receive serial data.

When configured as a full-duplex or receiver-only channel, the CMU PLL performs the functionality of the receiver CDR and recovers clock from the incoming serial data. The high-speed serial and low-speed parallel recovered clocks are used by the deserializer in the CMU channel and the deserialized data is forwarded directly to the FPGA fabric.

When configured as a full-duplex or transmitter-only channel, the serializer in the CMU channel serializes the parallel data from the FPGA fabric and drives the serial data to the transmitter buffer.

Table 1–39 shows the pins that are used as transmit and receive serial pins.

Table 1–39. Transmit and Receive Serial Pins (Part 1 of 2)

Pins *(1)*	When a CMU Channel is Configured as a Transceiver Channel	When a CMU Channel is Configured for Clock Generation
`REFCLK_[L,R][0,2,4,6]P`, `GXB_CMURX_[L_R][0,2,4,6]P` *(2)*	Receive serial input for CMU `Channel0`	Input reference clocks
`GXB_TX_[L,R][0,2,4,6]` *(2)*	Transmit serial output for CMU `Channel0`	Not available for use
`REFCLK_[L,R][1,3,5,7]P`, `GXB_CMURX_[L_R][1,3,5,7]P` *(3)*	Receive serial input for CMU `Channel1`	Input reference clocks

Table 1–39. Transmit and Receive Serial Pins (Part 2 of 2)

Pins *(1)*	When a CMU Channel is Configured as a Transceiver Channel	When a CMU Channel is Configured for Clock Generation
`GXB_TX_[L,R][1,3,5,7]P` *(3)*	Transmit serial output for CMU `Channel1`	Not available for use

Notes to Table 1–39:

(1) These indexes are for the Stratix IV GX and GT device with the maximum number of transceiver blocks. For exact information about how many of these pins are available for a specific device family, refer to the *Stratix IV Device Family Overview* chapter.

(2) Pins 0,2,4,6 are hardwired to CMU `channel0` in the corresponding transceiver blocks.

(3) Pins 1,3,5,7 are hardwired to CMU `channel1` in the corresponding transceiver blocks.

Interpret the pins column as follows:

For pins `REFCLK_[L,R][0,2,4,6]P` and `GXB_CMURX_[L_R][0,2,4,6]`, the "L, R" indicates the left and right side and the "0, 2, 4, 6" indicates the different pins. For example, a pin on the left side with index 0 is named: `REFCLK_L0P, GXB_CMURX_L0P.`

The receiver serial input pins are hardwired to their corresponding CMU channels. For more information, refer to the notes to Table 1–39.

Serializer and Deserializer

The serializer and deserializer convert the serial-to-parallel data on the transmitter and receiver side, respectively. The ALTGX MegaWizard Plug-In Manager provides the Basic (PMA Direct) functional mode (with a none and ×N option) to configure a transceiver channel to enable the transmitter serializer and receiver deserializer. To configure a CMU channel as a transceiver channel, you must use this functional mode.

The input data width options to serializer/from deserializer for a channel configured in this mode are 8, 10, 16, and 20.

To understand the maximum FPGA fabric-transceiver interface clock frequency limits, refer to the "Transmitter Channel Datapath Clocking" section in the *Stratix IV Transceiver Clocking* chapter.

CMU Clock Divider

When you configure a CMU channel in Basic (PMA Direct) ×1 mode, the CMU clock divider divides the high-speed clock from the other CMU channel (used as a clock generation unit) within the same transceiver block and provides the high-speed serial clock and low-speed parallel clocks to the transmitter side of the CMU channel. The CMU clock divider can divide the high-speed clock by /1, /2, and /4.

Clocks for the Transmitter Serializer

When you configure the CMU channel as a transceiver channel, the clocks for the transmitter side is provided by one of these sources:

- The other CMU channel in the same transceiver block that is configured as a clock multiplication unit
- From CMU `channel0` on the other transceiver block on the same side of the device through the ×N clock line (the `×N_Top` or `×N_Bottom` clock line). If you configure a CMU channel in Basic (PMA Direct) ×N mode, you can use this clocking option
- From one of the ATX PLL blocks on the same side of the device through the ×N clock line (the `×N_Top` or `×N_Bottom` clock line)

Input Reference Clocks for the Receiver CDR

When you configure a CMU Channel as a transceiver channel, there are multiple sources of input reference clocks for the receiver CDR:

- From adjacent REFCLKs within the same transceiver block, if the adjacent CMU Channel is not used as a transceiver channel
- From the REFCLK of adjacent transceiver blocks on the same side of the device, if the corresponding CMU channels are not used as transceiver channels. For REFCLK connections to the CMU channel from the global clock lines and PLL cascade network, refer to Table 1–6 on page 1–14.

For more information, refer to the "Input Reference Clocking" section of the *Stratix IV Transceiver Clocking* chapter.

Clocks for the Receiver Deserializer

The CDR provides high-speed serial and low-speed parallel clocks to the receiver deserializer from the recovered data.

Other CMU Channel Features

The CMU channels provide the following features:

- Analog control options—Differential output voltage (V_{OD}), pre-emphasis, equalization, and DC gain settings present in the regular channels are also available in the CMU channels.
- OCT—CMU channels can have an OCT feature. The allowed termination values are the same as regular channels (85, 100, 120, and 150 Ω).
- Loopback—The available loopback options are serial, reverse serial (pre-CDR), and reverse serial (CDR) loopback.

For more information about analog controls and OCT, refer to "Transmitter Output Buffer" on page 1–30 and "Receiver Input Buffer" on page 1–36.

For information about loopback, refer to "Loopback Modes" on page 1–188.

Dynamic Reconfiguration of the CMU Channel Analog Controls

For the dynamic reconfiguration capabilities of the CMU channels in Basic (PMA Direct) ×1/×N configurations, refer to the *Stratix IV Dynamic Reconfiguration* chapter.

Functional Modes

Table 1–40 list the transceiver functional modes you can use to configure the Stratix IV GX and GT devices using the ALTGX MegaWizard Plug-In Manager.

Table 1–40. Functional Modes for the Stratix IV GX and GT Devices

Functional Mode	Data Rate	Refer To
Basic Single Width	600 Mbps to 3.75 Gbps	"Basic Single-Width Mode Configurations" on page 1–110
Basic Double Width	1 Gbps to 8.5 Gbps	"Basic Double-Width Mode Configurations" on page 1–114
PCI Express (PIPE)	■ Gen1 at 2.5 Gbps ■ Gen2 at 5 Gbps	"PCI Express (PIPE) Mode Configurations" on page 1–125
XAUI	3.125 Gbps up to HiGig at 3.75 Gbps	"XAUI Mode Datapath" on page 1–154
GIGE	1.25 Gbps	"GIGE Mode Datapath" on page 1–164
Serial RapidIO	■ 1.25 Gbps ■ 2.5 Gbps ■ 3.125 Gbps	"Serial RapidIO Mode" on page 1–180
SONET/SDH	■ OC-12 ■ OC-48	"SONET/SDH Frame Structure" on page 1–169
SDI	■ HD at 1.485/1.4835 Gbps ■ 3G at 2.97/2.967 Gbps	"SDI Mode Datapath" on page 1–178
(OIF) CEI PHY Interface	>4.976 Gbps to 6.375 Gbps	"(OIF) CEI PHY Interface Mode Datapath" on page 1–180

Table 1–41 list the transceiver functional modes you can use to configure the Stratix IV GT devices using the ALTGX MegaWizard Plug-In Manager.

Table 1–41. Functional Modes for the Stratix IV GT Devices (Part 1 of 2)

Functional Mode	Data Rate	Refer To
Basic Single Width	2.488 Gbps to 3.75 Gbps	"Basic Single-Width Mode Configurations" on page 1–110
Basic Double Width	2.488 Gbps to 11.3 Gbps	"Basic Double-Width Mode Configurations" on page 1–114
Basic (PMA-Direct) single-width	2.488 Gbps to 3.25 Gbps	"Basic (PMA Direct) ×1 Configuration" on page 1–187
Basic (PMA-Direct) double-width	2.488 Gbps to 6.5 Gbps	"Basic (PMA Direct) ×N Configuration" on page 1–187
XAUI	3.125 Gbps up to HiGig at 3.75 Gbps	"XAUI Mode Datapath" on page 1–154

Table 1–41. Functional Modes for the Stratix IV GT Devices (Part 2 of 2)

Functional Mode	Data Rate	Refer To
Serial RapidIO	■ 2.5 Gbps ■ 3.125 Gbps	"Serial RapidIO Mode" on page 1–180
SONET/SDH	■ OC-48 ■ OC-96	"SONET/SDH Frame Structure" on page 1–169
SDI	■ 3G at 2.97/2.967 Gbps	"SDI Mode Datapath" on page 1–178
(OIF) CEI PHY Interface	>4.976 Gbps to 6.375 Gbps	"(OIF) CEI PHY Interface Mode Datapath" on page 1–180

Basic Functional Mode

The Stratix IV GX and GT transceiver datapaths are extremely flexible in Basic functional mode. To configure the transceivers in Basic functional mode, you must select **Basic** in the **Which protocol will you be using?** option of the ALTGX MegaWizard Plug-In Manager.

Basic functional mode can be further sub-divided into the following two functional modes:

- Basic single-width mode
- Basic double-width mode

You can configure the transceiver in Basic single-width mode by selecting **Single** in the **What is the deserializer block width?** option in the ALTGX MegaWizard Plug-In Manager. You can configure the transceiver in Basic double-width mode by selecting **Double** in the **What is the deserializer block width?** option in the ALTGX MegaWizard Plug-In Manager.

Table 1–42 shows the Stratix IV GX and GT PCS-PMA interface widths and data rates supported in Basic single-width and double-width modes.

Table 1–42. PCS-PMA Interface Widths and Data Rates in Basic Single-Width and Double-Width Modes for Stratix IV GX and GT Devices

Basic Functional Mode	Supported Data Rate Range *(1)*	PMA-PCS Interface Width
Basic single-width mode	600 Mbps to 3.75 Gbps	8 bit, 10 bit
Basic double-width mode	1 Gbps to 8.5 Gbps	16 bit, 20 bit

Note to Table 1–42:

(1) The data rate range supported in Basic single-width and double-width modes varies depending on whether or not you use the byte serializer/deserializer. For more information, refer to "Basic Single-Width Mode Configurations" on page 1–110 and "Basic Double-Width Mode Configurations" on page 1–114.

Table 1–43 shows the Stratix IV GT PCS-PMA interface widths and data rates supported in Basic single-width and double-width modes.

Table 1–43. PCS-PMA Interface Widths and Data Rates Supported in Basic Single-Width and Double-Width Modes for Stratix IV GT Devices *(Note 1)*

Basic Functional Mode	Supported Data Rate Range	PMA-PCS Interface Width
Basic single-width mode	2.488 Gbps to 3.75 Gbps	8-bit, 10-bit
Basic double-width mode	2.488 Gbps to 11.3 Gbps	16-bit, 20-bit

Note to Table 1–43:

(1) The data rate range supported in Basic single-width and double-width modes varies depending on whether or not you use the byte serializer/deserializer. For more information, refer to "Basic Single-Width Mode Configurations" on page 1–110 and "Basic Double-Width Mode Configurations" on page 1–114.

Low Latency PCS Datapath

The ALTGX MegaWizard Plug-In Manager provides an **Enable low latency PCS mode** option when configured in Basic single-width or Basic double-width mode. If you select this option, the following transmitter and receiver channel PCS blocks are bypassed to yield a low latency PCS datapath:

- 8B/10B encoder and decoder
- Word aligner
- Deskew FIFO
- Rate match (clock rate compensation) FIFO
- Byte ordering

In low latency PCS modes, the transmitter and receiver phase compensation FIFOs are always enabled. Depending on the targeted data rate, you can optionally bypass the byte serializer and deserializer blocks. For more information, refer to "Basic Single-Width Mode Configurations" on page 1–110 and "Basic Double-Width Mode Configurations" on page 1–114.

PCS latency in Basic single-width and Basic double-width modes with and without the low latency PCS mode option is pending characterization.

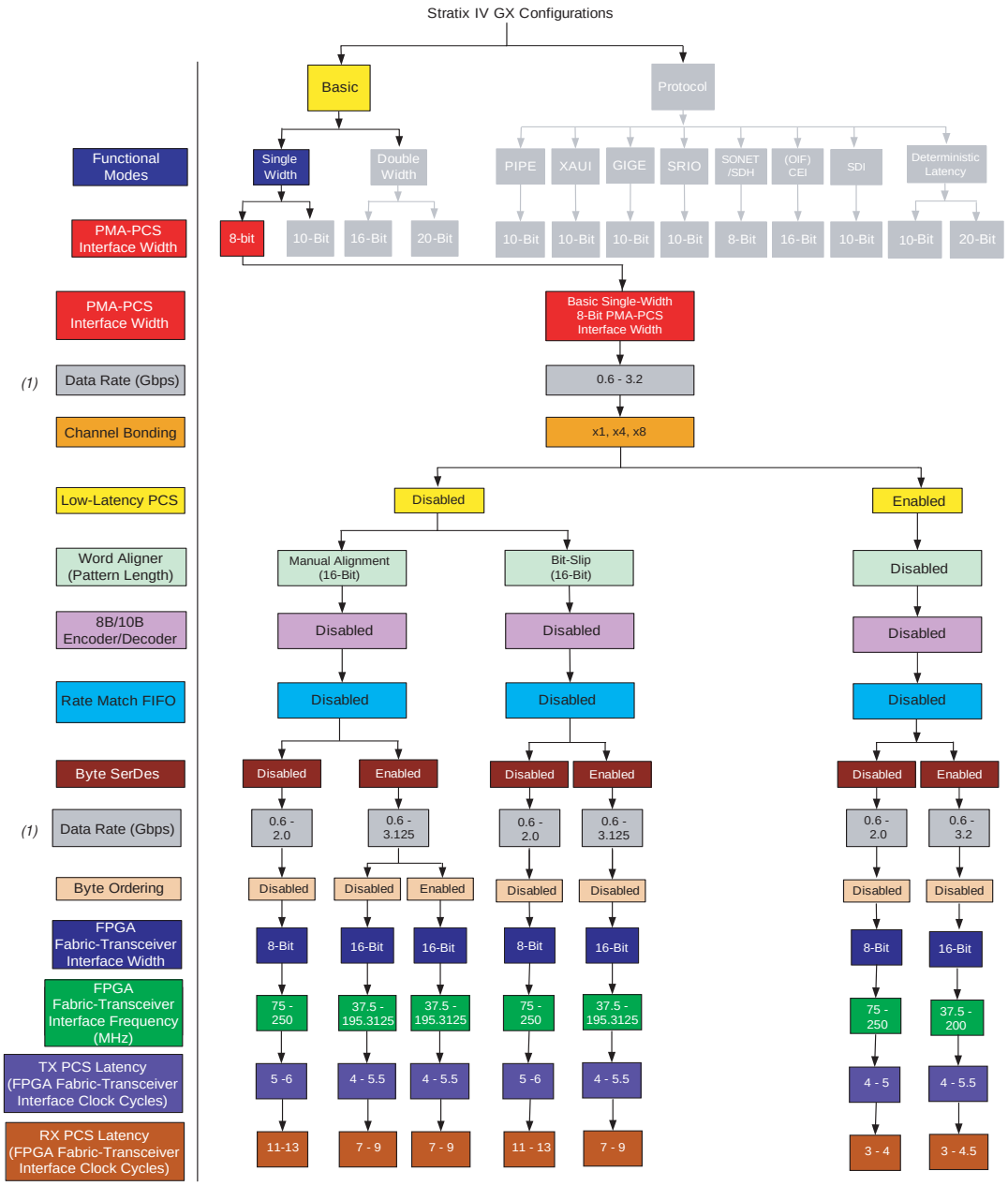
Basic double-width mode configurations at data rates of > 6.5 Gbps are only allowed in low-latency PCS bypass mode.

Basic Single-Width Mode Configurations

Figure 1–90 shows Stratix IV GX transceiver configurations allowed in Basic single-width functional mode with an 8-bit PMA-PCS interface.

Figure 1–91 shows Stratix IV GT transceiver configurations allowed in Basic single-width functional mode with an 8-bit PMA-PCS interface.

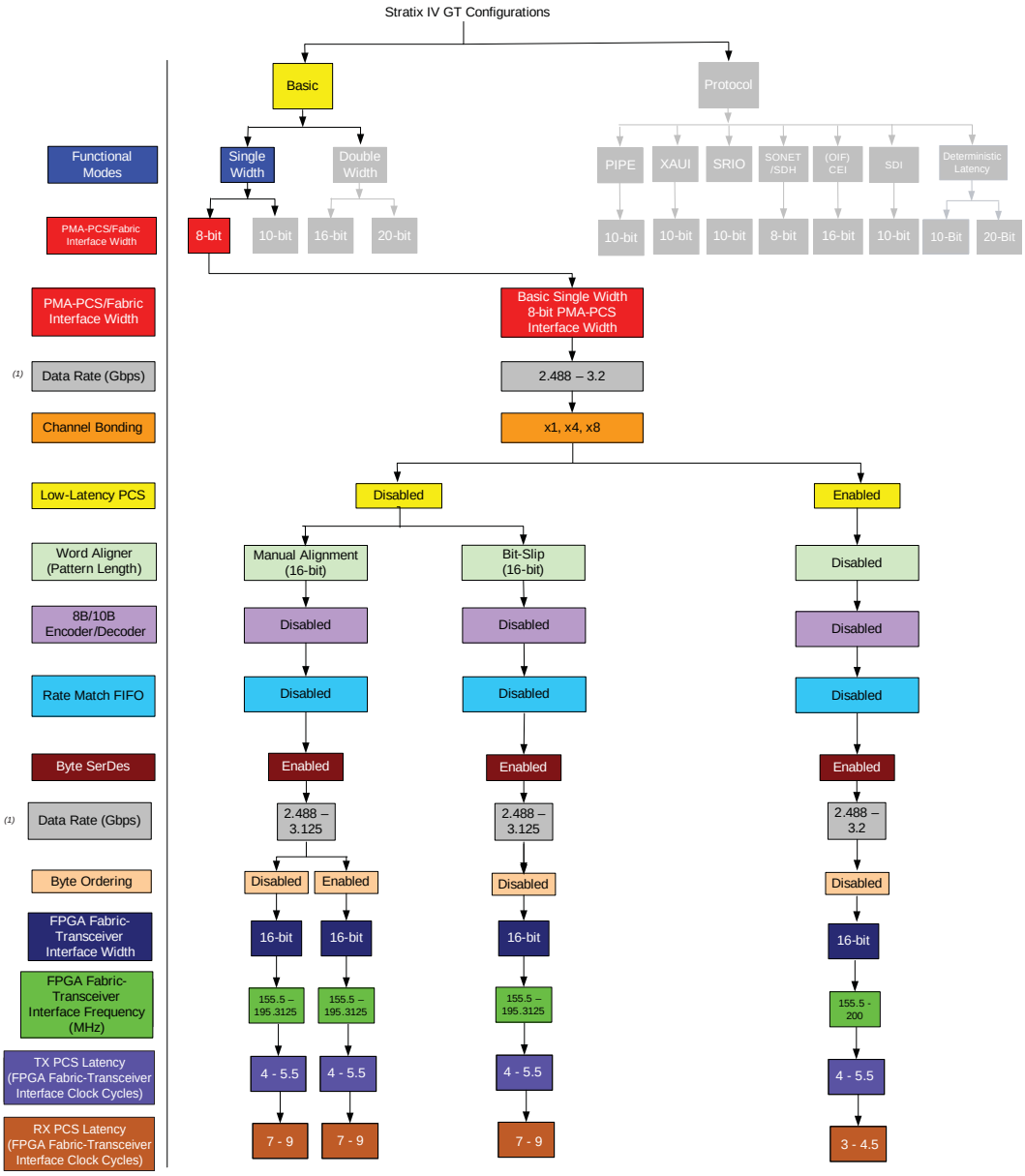
Figure 1–90. Transceiver Configurations in Basic Single-Width Mode with an 8-Bit PMA-PCS Interface for Stratix IV GX Devices



Note to Figure 1–90:

(1) The maximum data rate specification shown in Figure 1–90 is valid only for the -2 (fastest) speed grade devices. For data rate specifications for other speed grades offered, refer to the *DC and Switching Characteristics* chapter.

Figure 1–91. Transceiver Configurations in Basic Single-Width Mode with an 8-Bit PMA-PCS Interface for Stratix IV GT Devices



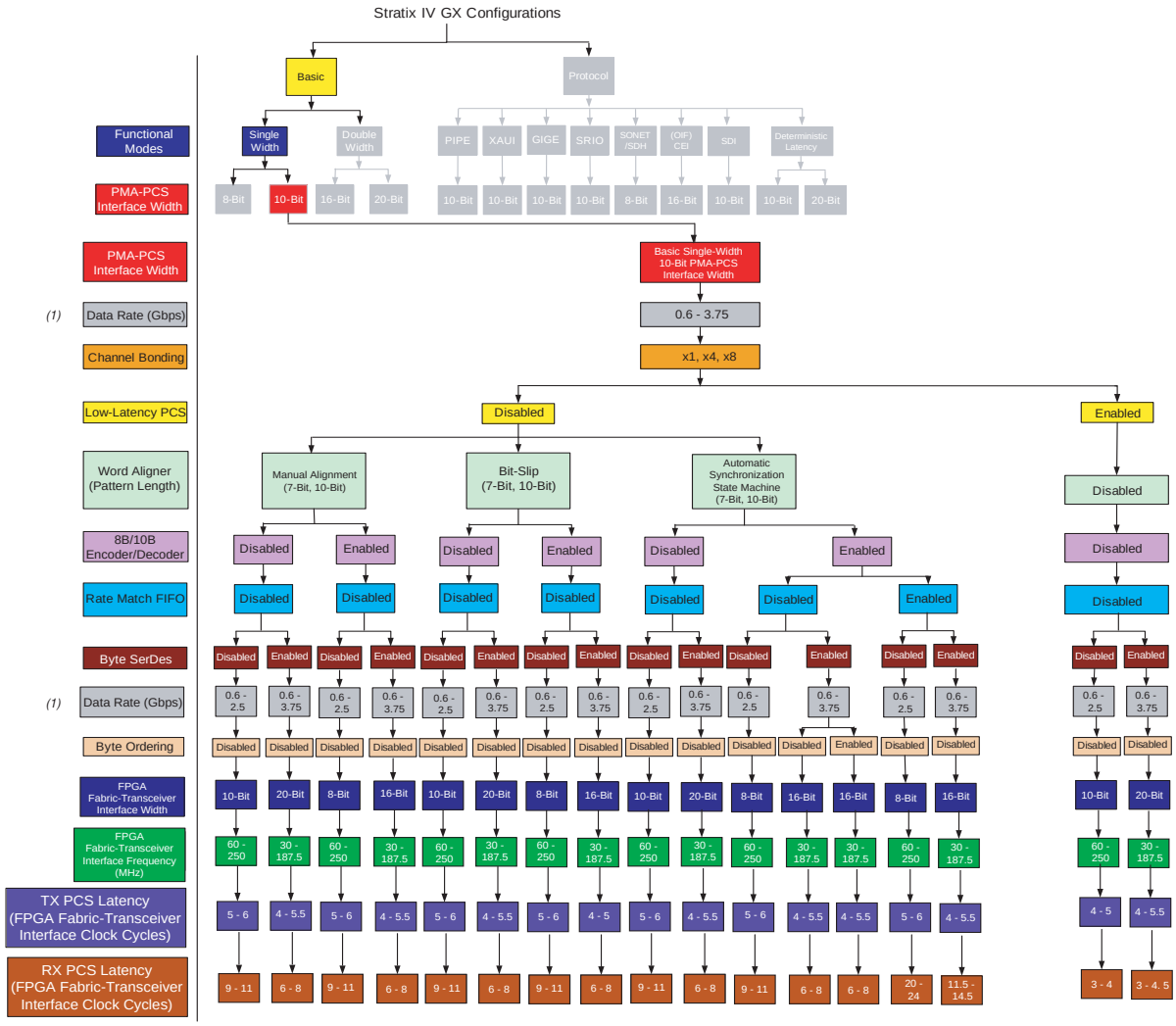
Note to Figure 1–91:

(1) The maximum data rate specification shown in Figure 1–91 is valid only for the -2 (fastest) speed grade devices. For data rate specifications for other speed grades offered, refer to the *DC and Switching Characteristics* chapter.

Figure 1–92 shows Stratix IV GX transceiver configurations allowed in Basic single-width functional mode with a 10-bit PMA-PCS interface.

Figure 1–93 shows Stratix IV GT transceiver configurations allowed in Basic single-width functional mode with a 10-bit PMA-PCS interface.

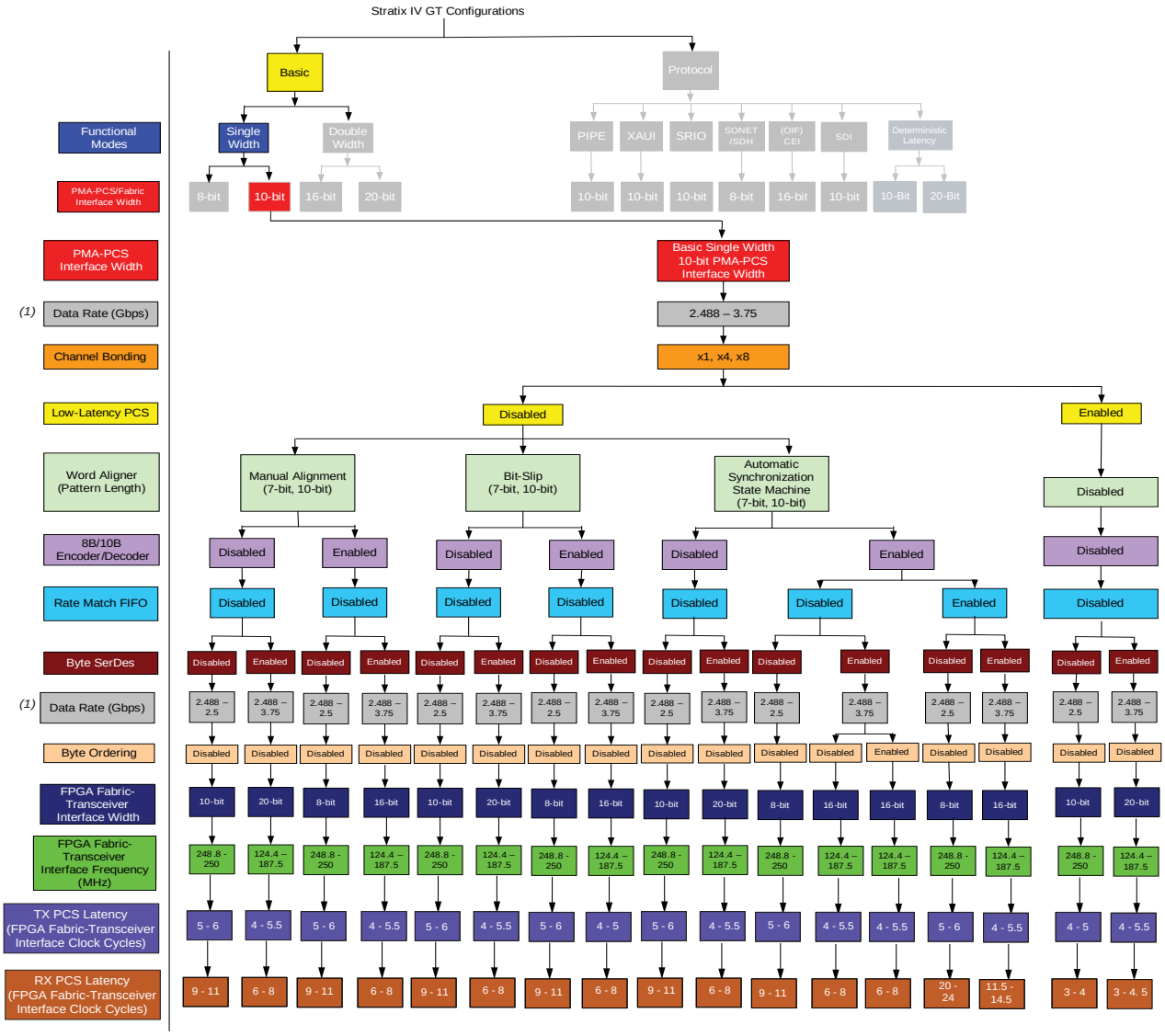
Figure 1–92. Transceiver Configurations in Basic Single-Width Mode with a 10-Bit PMA-PCS Interface for Stratix IV GX Devices



Note to Figure 1–92:

(1) The maximum data rate specification shown in Figure 1–92 is valid only for the -2 (fastest) speed grade devices. For data rate specifications for other speed grades offered, refer to the *DC and Switching Characteristics* chapter.

Figure 1–93. Transceiver Configurations in Basic Single-Width Mode with a 10-Bit PMA-PCS Interface for Stratix IV GT Devices



Note to Figure 1–93:

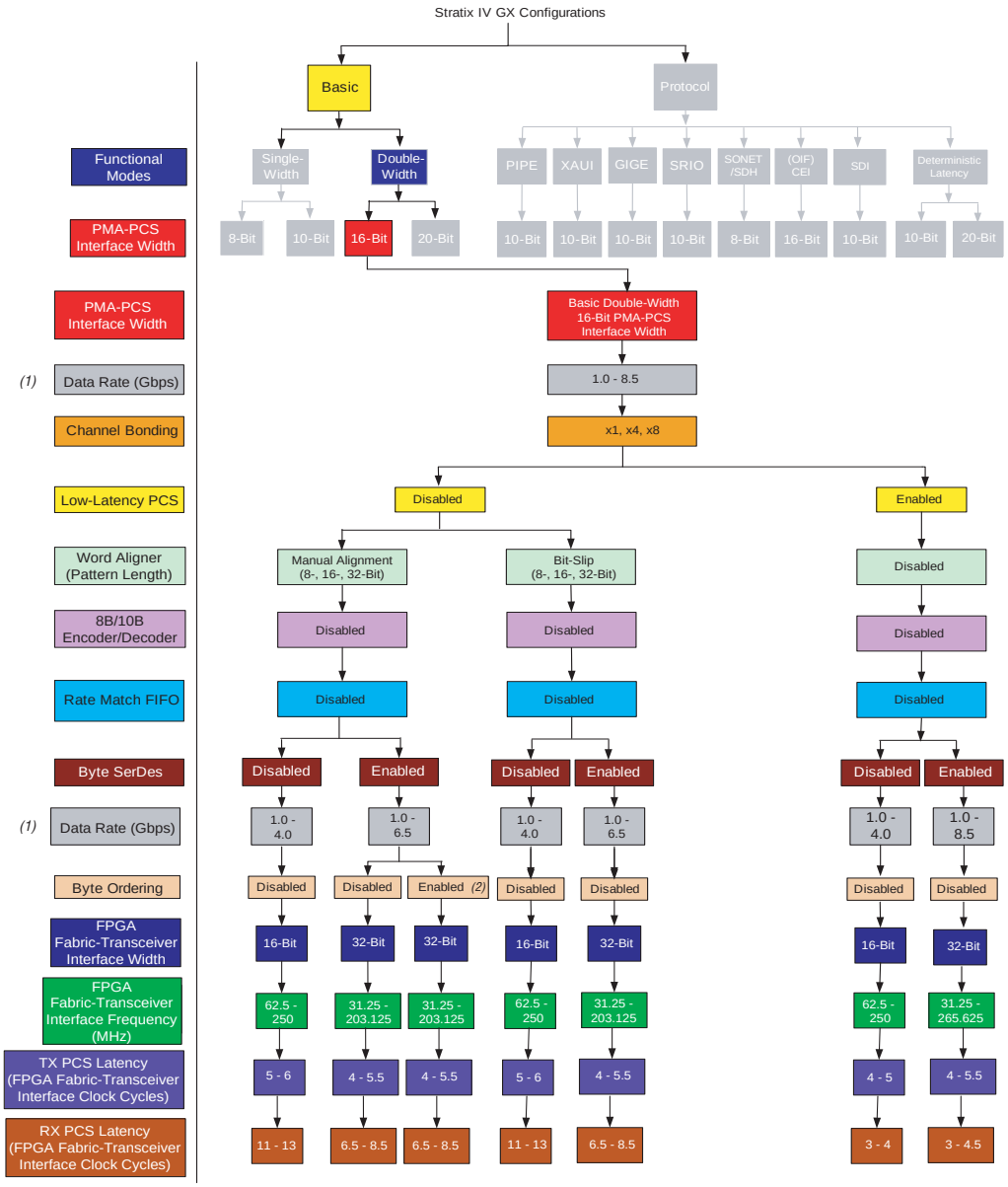
(1) The maximum data rate specification shown in Figure 1–93 is valid only for the -2 (fastest) speed grade devices. For data rate specifications for other speed grades offered, refer to the *DC and Switching Characteristics* chapter.

Basic Double-Width Mode Configurations

Figure 1–94 shows Stratix IV GX transceiver configurations allowed in Basic double-width functional mode with a 16-bit PMA-PCS interface.

Figure 1–95 shows Stratix IV GT transceiver configurations allowed in Basic double-width functional mode with a 16-bit PMA-PCS interface.

Figure 1–94. Transceiver Configurations in Basic Double-Width Mode with a 16-Bit PMA-PCS Interface for Stratix IV GX Devices

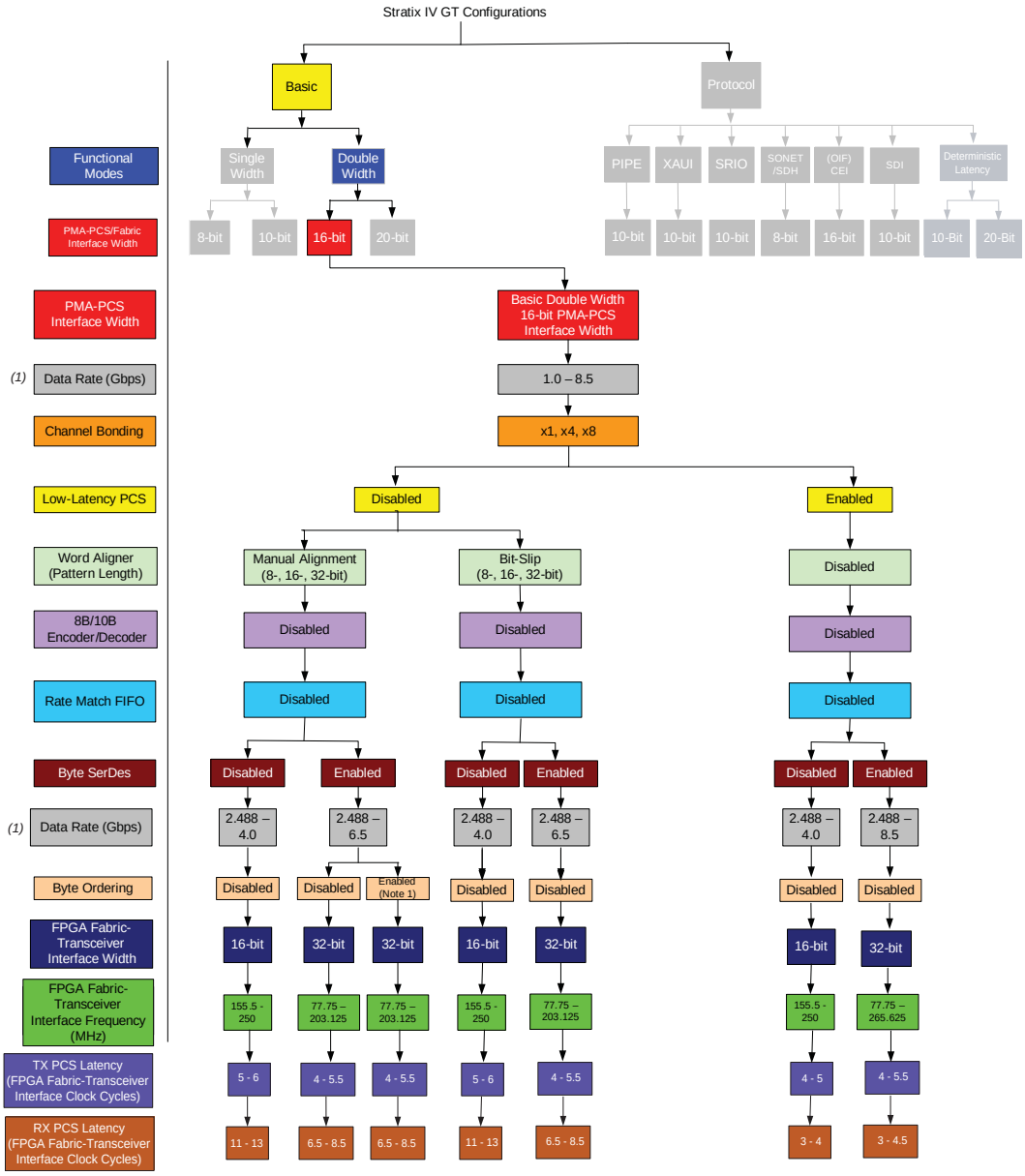


Notes to Figure 1–94:

(1) The maximum data rate specification shown in Figure 1–94 is valid only for the -2 (fastest) speed grade devices. For data rate specifications for other speed grades offered, refer to the *DC and Switching Characteristics* chapter.

(2) The byte ordering block is available only if you select the word alignment pattern length of 16 or 32 bits.

Figure 1–95. Transceiver Configurations in Basic Double-Width Mode with a 16-Bit PMA-PCS Interface for Stratix IV GT Devices



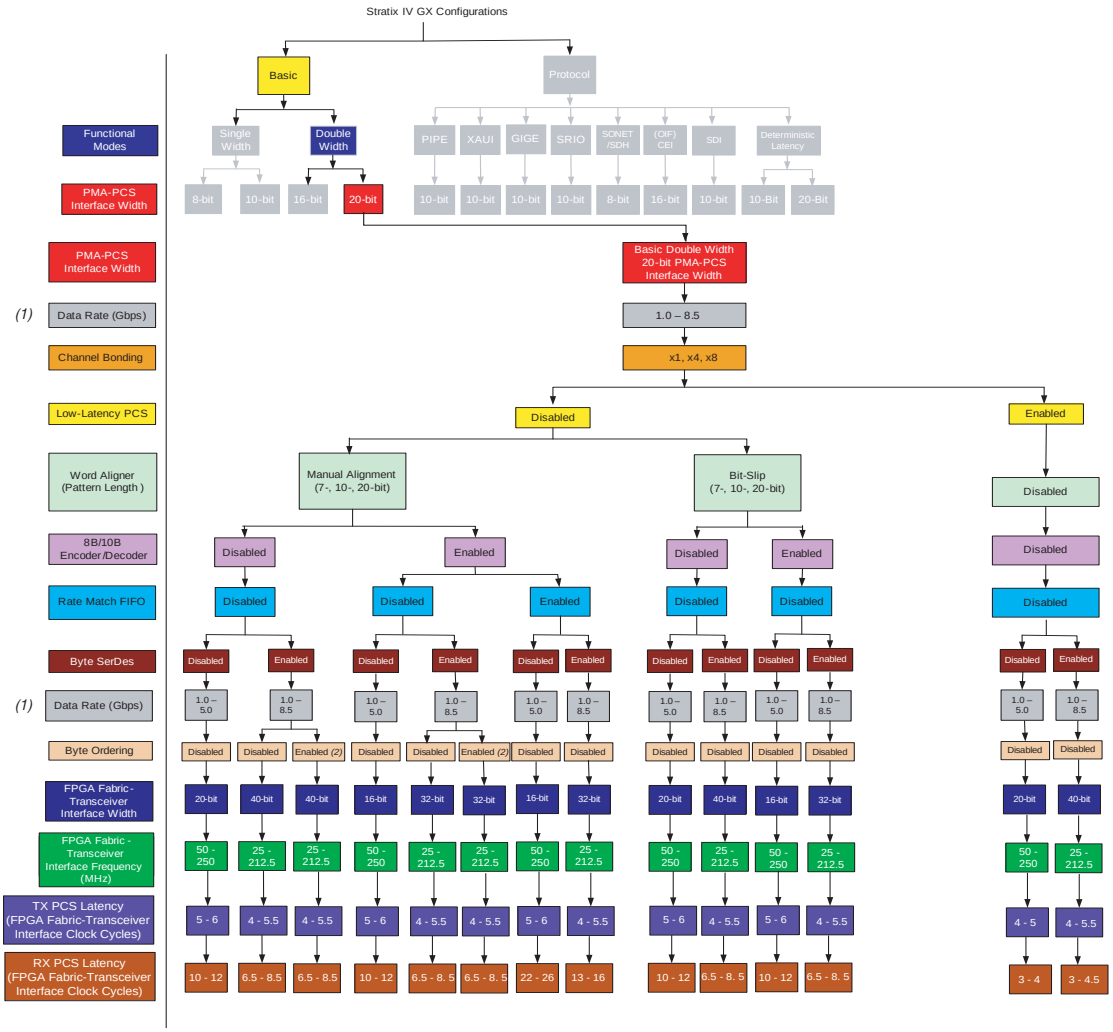
Note to Figure 1–95:

(1) The maximum data rate specification shown in Figure 1–95 is valid only for the -2 (fastest) speed grade devices. For data rate specifications for other speed grades offered, refer to the *DC and Switching Characteristics* chapter.

Figure 1–96 shows Stratix IV GX transceiver configurations allowed in Basic double-width functional mode with a 20-bit PMA-PCS interface.

Figure 1–97 shows Stratix IV GT transceiver configurations allowed in Basic double-width functional mode with a 20-bit PMA-PCS interface.

Figure 1–96. Transceiver Configurations in Basic Double-Width Mode with a 20-Bit PMA-PCS Interface for Stratix IV GX Devices

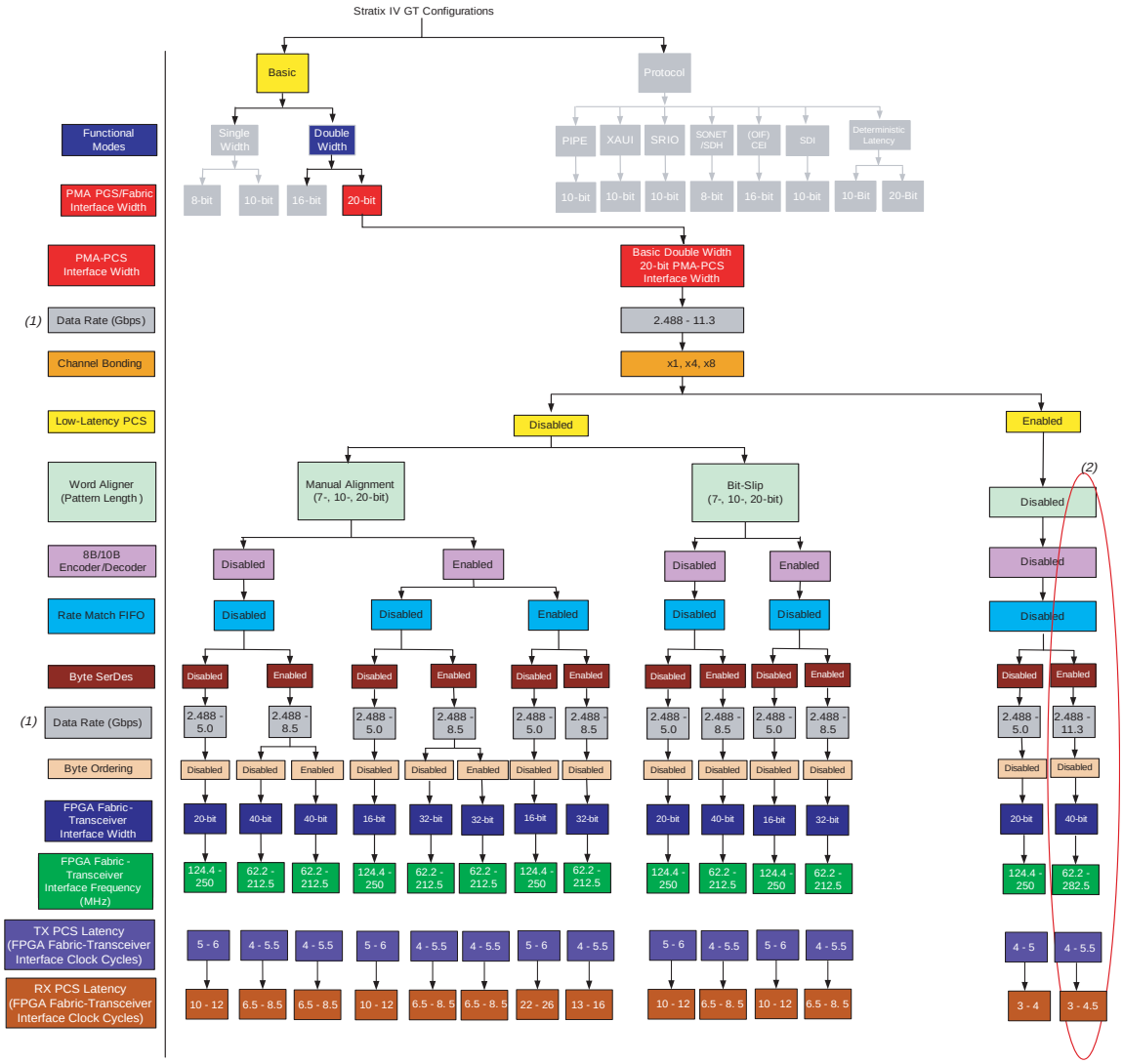


Notes to Figure 1–96:

(1) The maximum data rate specification shown in Figure 1–96 is valid only for the -2 (fastest) speed grade devices. For data rate specifications for other speed grades offered, refer to the *DC and Switching Characteristics* chapter.

(2) The byte ordering block is available only if you select the word alignment pattern length of 20 bits.

Figure 1–97. Transceiver Configurations in Basic Double-Width Mode with a 20-Bit PMA-PCS Interface for Stratix IV GT Devices



Notes to Figure 1–97:

(1) The maximum data rate specification shown in Figure 1–97 is valid only for the -2 (fastest) speed grade devices. For data rate specifications for other speed grades offered, refer to the *DC and Switching Characteristics* chapter.

(2) The circled configuration supports data rates up to 11.3 Gbps per channel to implement 40G/100G links.

For more information about 40G/100G transceivers, refer to:

- Enabling 40G/100G Solutions with FPGAs with 11.3-Gbps Transceivers web cast
- Stratix IV FPGA 40G/100G IP Solutions website
- *AN 570: Implementing the 40G/100G Ethernet Protocol in Stratix IV Devices*

SATA and SAS Options

Serial advanced technology attachment (SATA) and serial attached SCSI (SAS) are computer bus standards used in computers to transfer data between a mother board and mass storage devices. Stratix IV GX and GT devices offer options to implement a transceiver that satisfies SATA and SAS protocols. These options are:

- Transmitter in electrical idle mode
- Receiver signal detect functionality

These options and their selections are described in the following sections.

Transmitter Buffer Electrical Idle

In Basic functional mode, you can enable the optional input signal `tx_forceelecidle`. When this input signal of a channel is asserted high, the transmitter buffer in that channel is placed in the electrical idle state. During electrical idle, the output of the transmitter buffer is tri-stated.

This signal is used in applications like SATA and SAS for generating out of band (OOB) signals. An OOB signal is a pattern of idle times and burst times. Different OOB signals are distinguished by their different idle times.

For more information about the transmitter buffer in the Electrical Idle state, refer to the "Transmitter Buffer Electrical Idle" section in "PCI Express (PIPE) Mode" on page 1–124.

Receiver Input Signal Detect

In Basic functional mode, you can enable the optional `rx_signaldetect` signal (used for protocols such as SATA and SAS) only if you select the 8B/10B block. When you select the optional `rx_signaldetect` signal, an option is available to set the desired threshold level of the signal being received at the receiver's input buffer. If the signal threshold detection circuitry senses the signal level present at the receiver input buffer to be higher than the chosen signal detect threshold, it asserts the `rx_signaldetect` signal high. Otherwise, the signal threshold detection circuitry de-asserts the `rx_signaldetect` signal low. This signal is useful in applications such as SATA and SAS for detecting out of band (OOB) signals.

For more information on the signal threshold detection circuitry, refer to the "Signal Threshold Detection Circuitry" section in *Stratix IV Transceiver Architecture* chapter.

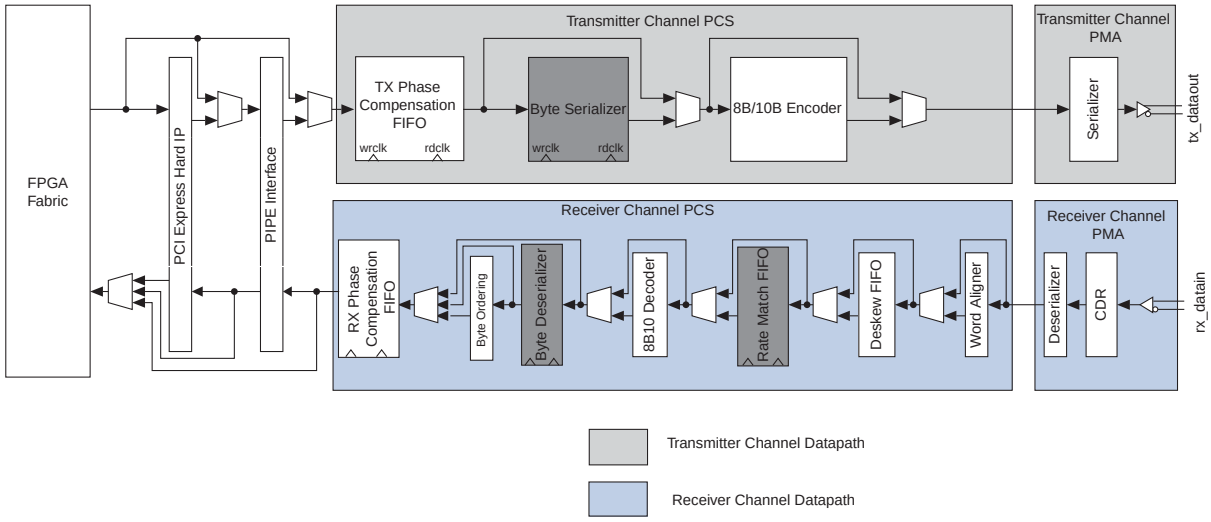
For information about other protocols supported using Basic functional mode, refer to *AN 577: Recommended Protocol Configurations for Stratix IV FPGAs*.

Deterministic Latency Mode

Stratix IV GX and GT devices have a deterministic latency option available for use in high-speed serial interfaces such as CPRI (Common Public Radio Interface) and Open Base Station Architecture Initiative Reference Point3 (OBSAI RP3). Both CPRI and OBSAI RP3 protocols place stringent requirements on the amount of latency variation that is permissible through a link implementing these protocols.

Figure 1–98 shows the transceiver datapath when using deterministic latency mode.

Figure 1–98. Transceiver Datapath When in Deterministic Latency Mode



To implement this mode, select the **Deterministic Latency** option under the **Which Protocol will you be using?** section in the ALTGX MegaWizard Plug-In Manager. When you select this option, the transmitter channel is automatically placed in bit-slip mode and **Enable TX Phase Comp FIFO in register mode** is automatically selected as well. The receiver's phase compensation FIFO is automatically placed in the register mode. In addition, an output port (`rx_bitslipboundaryselectout[4:0]`) from the receiver's word aligner and an input port (`tx_bitslipboundaryselect[4:0]`) for the transmitter bit-slip circuitry are instantiated. The option for placing the transmitter phase compensation FIFO in register mode is also available.

Transmitter Bit Slipping

The transmitter is bit slipped to achieve deterministic latency. Use the `tx_bitslipboundaryselect[4:0]` port to set the number of bits that the transmitter block needs to slip. Table 1–44 lists the number of bits that are allowed to be slipped under different channel widths.

Table 1–44. Number of Transmitter Bits Allowed to be Slipped in Deterministic Latency Mode

Channel Width	Slip Zero
8/10 bit	9 bits
16/20 bit	19 bits

Receiver Bit Slipping

The number of bits slipped in the receiver's word aligner is given out on the `rx_bitslipboundaryselectout[4:0]` output port. The information on this output depends on your deserializer block width.

In single-width mode with 8/10-bit channel width, the number of bits slipped in the receiver path is given out sequentially on this output. For example, if zero bits are slipped, the output on `rx_bitslipboundaryselectout[4:0]` shows a value of 0(00000); if two bits are slipped, the output on `rx_bitslipboundaryselectout[4:0]` shows a value of 2 (00010).

In double-width mode with 16/20-bit channel width, the output is 19 minus the number of bits slipped. For example, if zero bits are slipped, the output on `rx_bitslipboundaryselectout[4:0]` shows a value of 19 (10011); if two bits are slipped, the output on `rx_bitslipboundaryselectout[4:0]` shows a value of 17 (10001).

The information on `rx_bitslipboundaryselectout[4:0]` output port helps in calculating the latency through the receiver datapath. You can use the information on `rx_bitslipboundaryselectout[4:0]` to set up the `tx_bitslipboundaryselect[4:0]` appropriately to cancel out the latency uncertainty.

Receiver Phase Comp FIFO in Register Mode

To remove the latency uncertainty through the receiver's phase compensation FIFO, select the **Enable the RX phase comp FIFO in register mode** option in the ALTGX MegaWizard Plug-In Manager. In register mode, the phase compensation FIFO acts as a register and thereby removes the uncertainty in latency. The latency through the phase compensation FIFO in register mode is one clock cycle.

This mode is available in:

- Basic single-width mode with 8-bit channel width and 8B/10B Encoder enabled or 10-bit channel width with 8B/10B disabled.
- Basic double-width mode with 16-bit channel width and 8B/10B encoder enabled or 20-bit channel width with 8B/10B disabled.

Transmitter Phase Compensation FIFO in Register Mode

In register mode, the phase compensation FIFO acts as a register and thereby removes the uncertainty in latency. The latency through the transmitter and receiver phase compensation FIFO in register mode is one clock cycle.

CMU PLL Feedback

To implement deterministic latency functional mode, the phase relationship between the low-speed parallel clock and CMU PLL input reference clock must be deterministic. You can achieve this by selecting the **Enable PLL phase frequency detector (PFD) feedback to compensate latency uncertainty in Tx dataout and Tx clkout paths relative to the reference clock** option in the ALTGX MegaWizard Plug-In Manager. By selecting this option, a feedback path is enabled that ensures a deterministic relationship between the low-speed parallel clock and CMU PLL input reference clock.

In order to achieve deterministic latency through the transceiver, the reference clock to the CMU PLL must be the same as the low-speed parallel clock. For example, if you need a data rate of 1.2288 Gbps to be implemented for the CPRI protocol that places stringent requirements on the amount of latency variation, you must choose a reference clock of 122.88 MHz to allow for a feedback path from the CMU PLL to be used. This feedback path reduces the variations in latency.

When selecting this option, you must provide an input reference clock to the CMU PLL that is of the same frequency as the low-speed parallel clock.

In a CPRI implementation, the input reference clock to the CMU PLL must be the same as the low-speed parallel clock. Each CPRI channel uses one CMU PLL; therefore, each transceiver block can implement two CPRI ×1 channels only. ATX PLLs do not have the feedback path enabled; therefore, they cannot be used for implementing the CPRI configuration.

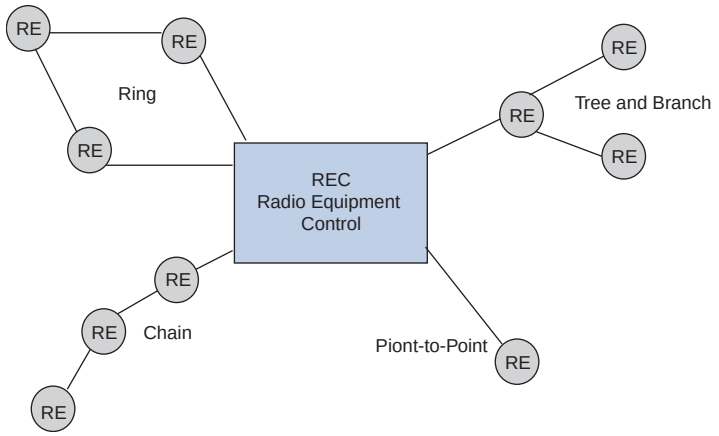
In the deterministic latency ×4 option, up to four CPRI TX channels can be bundled in an ×4 group so that they all have the same TX uncertainty and just require one TX PLL to compensate for it. This is allowed in cases where the data rates are multiples of a single PLL output frequency; for example, 0.6144 Gbps, 1.228 Gbps, 2.4576 Gbps, and 4.9152 Gbps. For ×4 bundled channels to maintain PLL lock during auto-negotiation, the IP must use over-sampling (sending the same bit multiple times) to output lower auto-negotiated line rates. Do not use the hard 8B/10B for oversampled channels.

CPRI and OBSAI

You can use deterministic latency functional mode to implement protocols such as CPRI and OBSAI.

The CPRI interface defines a digital point-to-point interface between the Radio Equipment Control (REC) and the Radio Equipment (RE) allowing flexibility in either co-locating the REC and the RE or remote location of the RE. Figure 1–99 shows various CPRI topologies. In most cases, CPRI links are between REC and RE modules or between two RE modules in a chain configuration.

Figure 1–99. CPRI Topologies

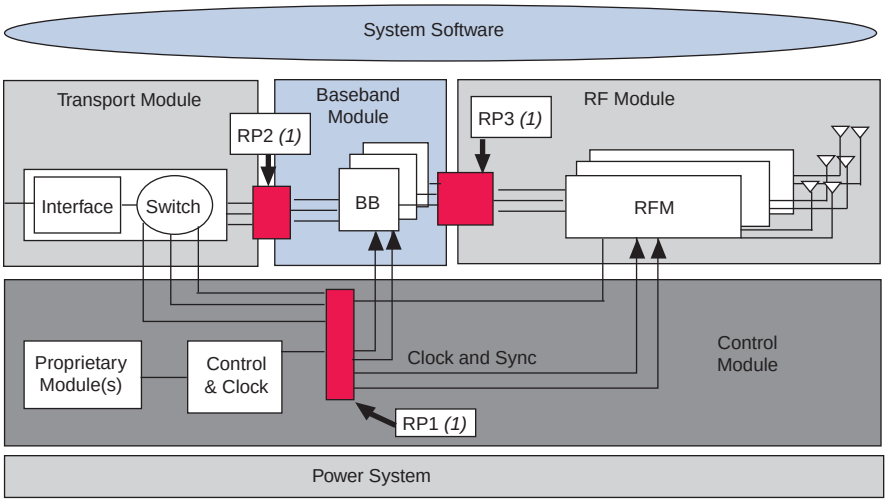


If the destination for high-speed serial data leaving the REC is the first RE, it is a single-hop connection. If serial data from the REC has to traverse through multiple REs before reaching the destination RE, it is a multi-hop connection. Remotely locating the RF transceiver from the main base station introduces a complexity with overall system delay. CPRI specification requires that the accuracy of measurement of round-trip delay on single-hop and multi-hop connections be within ±16.276 ns in order to properly estimate the cable delay. For a single-hop system, this allows a variation in round-trip delay of up to ±16.276 ns. For multi-hop systems however, the allowed delay variation is divided among number of hops in the connection—typically equal to ±16.276 ns/ (# of hops), but not always equally divided among the hops. Deterministic latency on a CPRI link also enables highly accurate triangulation of a caller's location.

The OBSAI was established by several OEM's for developing a set of specifications that can be used for configuring and connecting common modules into base transceiver stations (BTS). The BTS has four main modules—radio frequency (RF), baseband, control and transport.

Figure 1–100 show a typical BTS. The radio frequency module (RFM) receives signals using portable devices and converts them to digital data. The baseband module processes the encoded signal and brings it back to baseband before transmitting it to the terrestrial network using the transport module. Coordination between these three functions is maintained by a control module.

Figure 1–100. BTS in OSBAL

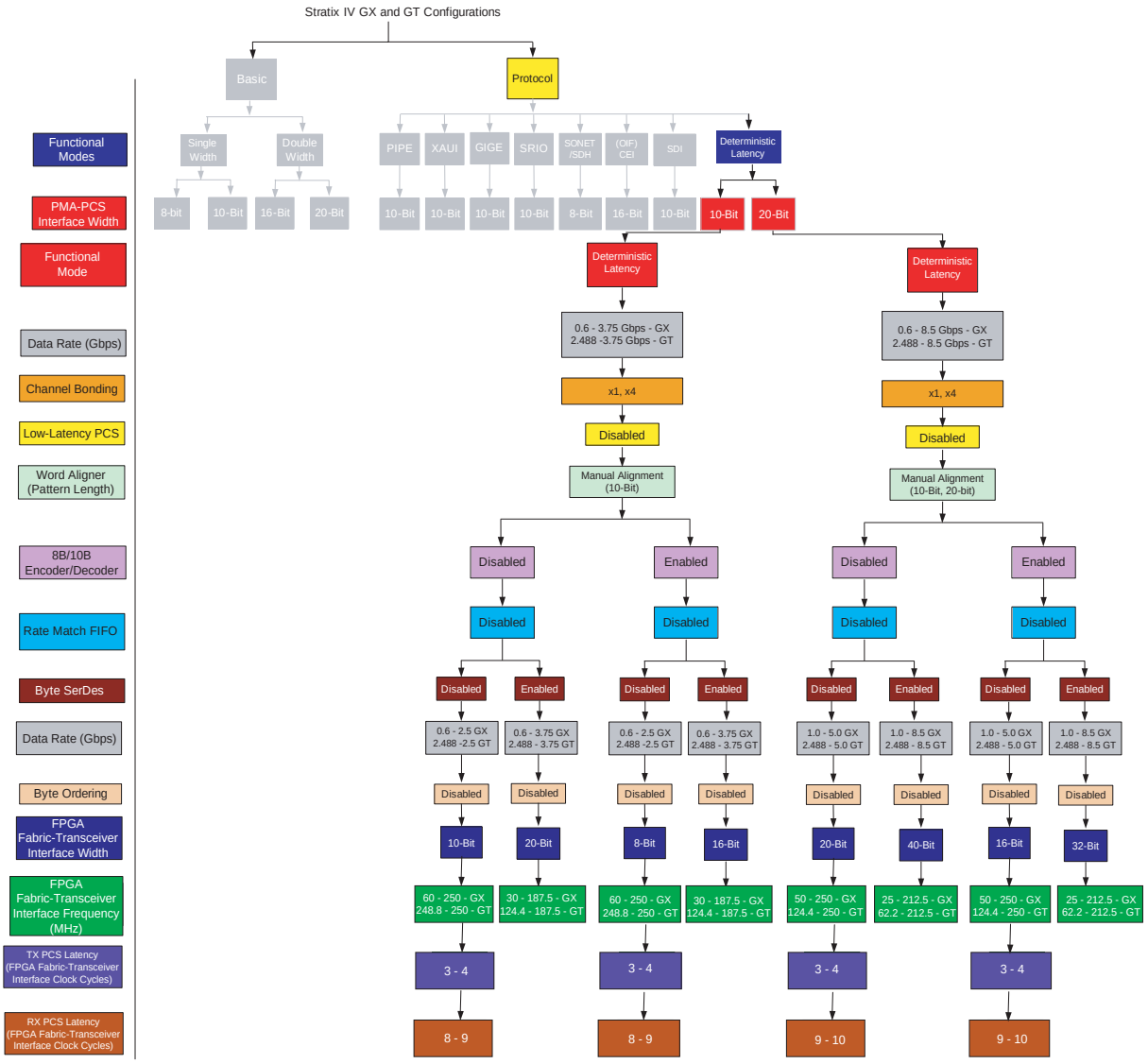


Note to Figure 1–100:

(1) "RP" means Reference Point.

Under the deterministic latency option, CPRI data rates can be implemented in single-width mode with 8/10-bit channel width and double-width mode with 16/20-bit channel width options only. Figure 1–101 shows the block diagram of the deterministic latency option.

Figure 1–101. Block Diagram of the Deterministic Latency Option



To implement CPRI/OBSAI using deterministic latency mode, Altera recommends using configurations with the byte serializer/deserializer disabled.

Table 1–45 lists the PMA-PCS interface widths, CPRI and OSBAI data rates in deterministic latency mode.

Table 1–45. PMA-PCS Interface Widths, CPRI and OSBAI Data Rates in Deterministic latency Mode

Deterministic Latency Mode	Supported Data Rate Range	PMA-PCS Interface Width for CPRI & OBSAI	CPRI Data Rate (GBPS)	PCS Clock Frequency (MHz)	OBSAI Data Rate (Gbps)	PCS Clock Frequency (MHz)
Single-width mode	600 Mbps to 3.75 Gbps	8 bit/10 bit	0.6144	61.44	768	76.8
			1.2288	122.88	1.536	153.6
			2.4576 *(1)*	245.76	—	—
Double-width mode	> 1 Gbps	16 bit/20 bit	3072	153.6	1.536	76.8
		16 bit/20 bit	4915.2 *(2)*, *(3)*	245.76	3072 *(3)*	153.6
		32 bit/40 bit	6144 *(2)*, *(3)*, *(4)*	307.2	6144 *(3)*, *(4)*	307.2

Notes to Table 1–45:

(1) When configured in double-width mode for the same data rate, the core clock frequency is halved.

(2) Requires double-width mode.

(3) When configured for 32/40-bit channel width requiring byte serializer/deserializer, the core clock is halved.

(4) Requires the byte serializer/deserializer.

PCI Express (PIPE) Mode

Intel Corporation has developed a PHY interface for the PCI Express (PIPE) Architecture specification to enable implementation of a PCI Express (PIPE)-compliant physical layer device. The PCI Express (PIPE) specification also defines a standard interface between the physical layer device and the media access control layer (MAC). Version 2.0 of the PCI Express (PIPE) specification provides implementation details for a PCI Express (PIPE)-compliant physical layer device at both Gen1 (2.5 GT/s) and Gen2 (5 GT/s) signaling rates.

To implement a Version 2.0 PCI Express (PIPE)-compliant PHY, you must configure the Stratix IV GX and GT transceivers in PCI Express (PIPE) functional mode. Stratix IV GX and GT devices have built-in PCI Express (PIPE) hard IP blocks that you can use to implement the PHY-MAC layer, data link layer, and transaction layer of the PCI Express (PIPE) protocol stack. You can also bypass the PCI Express (PIPE) hard IP blocks and implement the PHY-MAC layer, data link layer, and transaction layer in the FGPA fabric using a soft IP. If you enable the PCI Express (PIPE) hard IP blocks, the Stratix IV transceivers interface with these hard IP blocks. Otherwise, the Stratix IV transceivers interface with the FPGA fabric.

You can configure the Stratix IV GX and GT transceivers in PCI Express (PIPE) functional mode using one of the following two methods:

- ALTGX MegaWizard Plug-In Manager—if you do not use the PCI Express (PIPE) hard IP block
- PCI Express (PIPE) Compiler—if you use the PCI Express (PIPE) hard IP block

☞ Description of PCI Express (PIPE) hard IP architecture and PCI Express (PIPE) mode configurations allowed when using the PCI Express (PIPE) hard IP block are beyond the scope of this chapter. For more information about the PCI Express (PIPE) hard IP block, refer to the *PCI Express Compiler User Guide.*

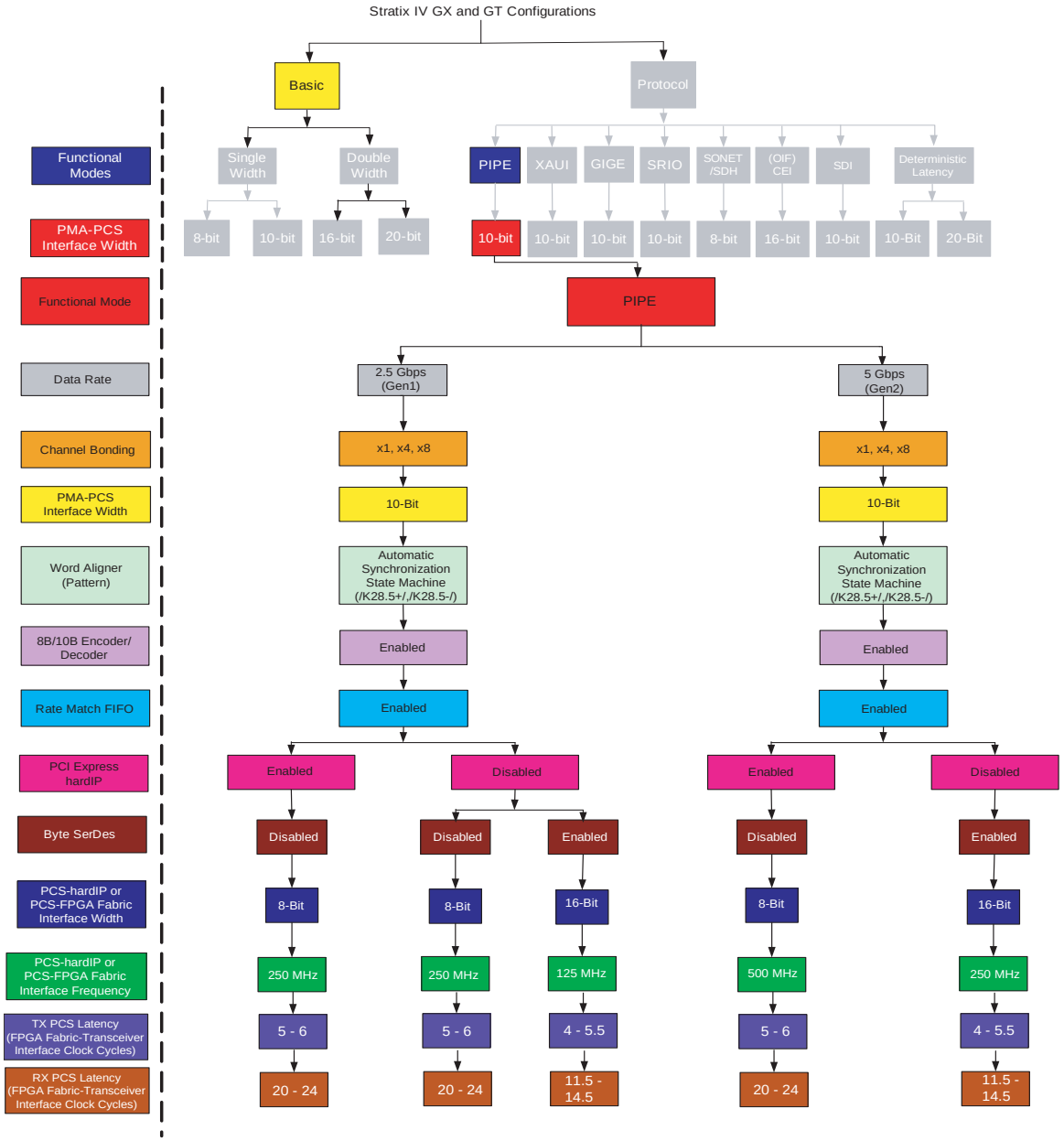
PCI Express (PIPE) Mode Configurations

Stratix IV GX and GT transceivers support both Gen1 (2.5 Gbps) and Gen2 (5 Gbps) data rates in PCI Express (PIPE) functional mode. When configured for the Gen2 (5 Gbps) data rate, the Stratix IV GX and GT transceivers allow dynamic switching between Gen2 (5 Gbps) and Gen1 (2.5 Gbps) signaling rates. Dynamic switch capability between the two PCI Express (PIPE) signaling rates is critical for speed negotiation during link training.

Stratix IV GX and GT transceivers support ×1, ×4, and ×8 lane configurations in PCI Express (PIPE) functional mode at both 2.5 Gbps and 5 Gbps data rates. In PCI Express (PIPE) ×1 configuration, the PCS and PMA blocks of each channel are clocked and reset independently. PCI Express (PIPE) ×4 and ×8 configurations support channel bonding for four-lane and eight-lane PCI Express (PIPE) links. In these bonded channel configurations, the PCS and PMA blocks of all bonded channels share common clock and reset signals.

Figure 1–102 shows the Stratix IV GX and GT transceiver configurations allowed in PCI Express (PIPE) functional mode.

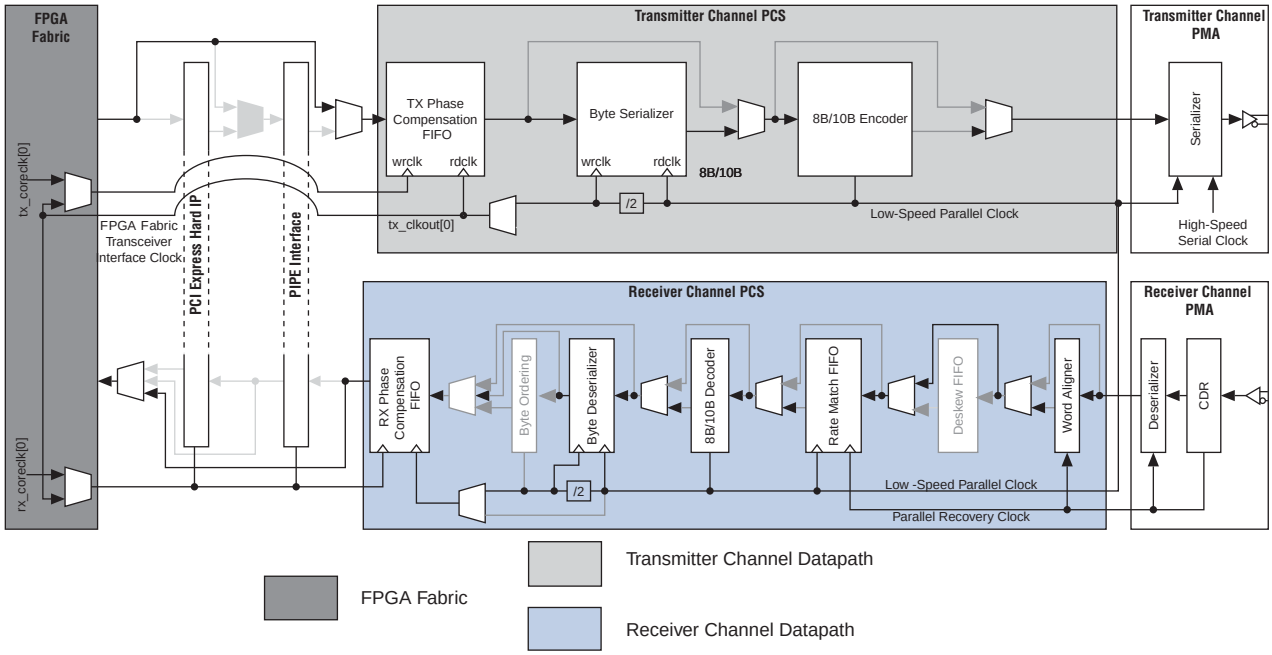
Figure 1–102. Stratix IV GX and GT Transceivers in PCI Express (PIPE) Functional Mode



PCI Express (PIPE) Mode Datapath

Figure 1–103 shows the Stratix IV GX and GT transceiver datapath when configured in PCI Express (PIPE) functional mode.

Figure 1–103. Stratix IV GX and GT Transceiver Datapath in PCI Express (PIPE) ×1 Mode



For more information, refer to "Rate Match (Clock Rate Compensation) FIFO" on page 1–74.

Table 1–46 shows the transceiver datapath clock frequencies in PCI Express (PIPE) functional mode configured using the ALTGX MegaWizard Plug-In Manager.

Table 1–46. Stratix IV GX and GT Transceiver Datapath Clock Frequencies in PCI Express (PIPE) Mode

Functional Mode	Data Rate	High-Speed Serial Clock Frequency	Parallel Recovered Clock and Low-Speed Parallel Clock Frequency	FPGA Fabric-Transceiver Interface Clock Frequency	
				Without Byte Serializer/ Deserializer (8 Bit Wide)	With Byte Serializer/ Deserializer (16 Bit Wide)
PCI Express (PIPE) ×1, ×4, and ×8 (Gen1)	2.5 Gbps	1.25 GHz	250 MHz	250 MHz	125 MHz
PCI Express (PIPE) ×1, ×4, and ×8 (Gen2)	5 Gbps	2.5 GHz	500 MHz	N/A *(1)*	250 MHz

Note to Table 1–46:

(1) In PCI Express (PIPE) functional mode at Gen2 (5 Gbps) data rate, the byte serializer/deserializer cannot be bypassed.

Transceiver datapath clocking varies between non-bonded (×1) and bonded (×4 and ×8) configurations in PCI Express (PIPE) mode.

For more information about transceiver datapath clocking in different PCI Express (PIPE) configurations, refer to the *Stratix IV Transceiver Clocking* chapter.

Table 1–47 lists the transmitter and receiver datapaths in PCI Express (PIPE) mode.

Table 1–47. Datapaths in PCI Express (PIPE) Mode

	Transmitter Datapath	Receiver Datapath
PCI Express (PIPE) interface	✓	✓
Transmitter phase compensation FIFO	✓	—
Optional byte serializer (enabled for 16-bit and disabled for 8-bit FPGA fabric-transceiver interface	✓	—
8B/10B encoder	✓	—
10:1 serializer	✓	—
Transmitter buffer with receiver detect circuitry	✓	—
Receiver buffer with signal detect circuitry	—	✓
1:10 deserializer	—	✓
Word aligner that implements PCI Express (PIPE)-compliant synchronization state machine	—	✓
Optional rate match FIFO (clock rate compensation) that can tolerate up to 600 PPM frequency difference	—	✓
8B/10B decoder	—	✓
Optional byte deserializer (enabled for 16-bit and disabled for 8-bit FPGA fabric-transceiver interface)	—	✓
Receiver phase compensation FIFO	—	✓

Table 1–48 shows features supported in PCI Express (PIPE) functional mode for 2.5 Gbps and 5 Gbps data rate configurations.

For more information, refer to "Rate Match FIFO in PCI Express (PIPE) Mode" on page 1–75.

Table 1–48. Supported Features in PCI Express (PIPE) Mode (Part 1 of 2)

Feature	2.5 Gbps (Gen1)	5 Gbps (Gen2)
×1, ×4, ×8 link configurations	✓	✓
PCI Express (PIPE)-compliant synchronization state machine	✓	✓
±300 PPM (total 600 PPM) clock rate compensation	✓	✓
8-bit FPGA fabric-transceiver interface	✓	—
16-bit FPGA fabric-transceiver interface	✓	✓
Transmitter buffer electrical idle	✓	✓
Receiver Detection	✓	✓
8B/10B encoder disparity control when transmitting compliance pattern	✓	✓

Table 1–48. Supported Features in PCI Express (PIPE) Mode (Part 2 of 2)

Feature	2.5 Gbps (Gen1)	5 Gbps (Gen2)
Power state management	✓	✓
Receiver status encoding	✓	✓
Dynamic switch between 2.5 Gbps and 5 Gbps signaling rate	—	✓
Dynamically selectable transmitter margining for differential output voltage control	—	✓
Dynamically selectable transmitter buffer de-emphasis of -3.5 db and -6 dB	—	✓

PCI Express (PIPE) Interface

In PCI Express (PIPE) mode, each channel has a PCI Express (PIPE) interface block that transfers data, control, and status signals between the PHY-MAC layer and the transceiver channel PCS and PMA blocks. The PCI Express (PIPE) interface block is compliant to version 2.0 of the PCI Express (PIPE) specification. If you use the PCI Express (PIPE) hard IP block, the PHY-MAC layer is implemented in the hard IP block. Otherwise, the PHY-MAC layer can be implemented using soft IP in the FPGA fabric.

☞ The PCI Express (PIPE) interface block is only used in PCI Express (PIPE) mode and cannot be bypassed.

Besides transferring data, control, and status signals between the PHY-MAC layer and the transceiver, the PCI Express (PIPE) interface block implements the following functions required in a PCI Express (PIPE)-compliant physical layer device:

- Forces the transmitter buffer in electrical idle state
- Initiates the receiver detect sequence
- 8B/10B encoder disparity control when transmitting compliance pattern
- Manages the PCI Express (PIPE) power states
- Indicates the completion of various PHY functions; for example, receiver detection and power state transitions on the `pipephydonestatus` signal
- Encodes the receiver status and error conditions on the `pipestatus[2:0]` signal as specified in the PCI Express (PIPE) specification

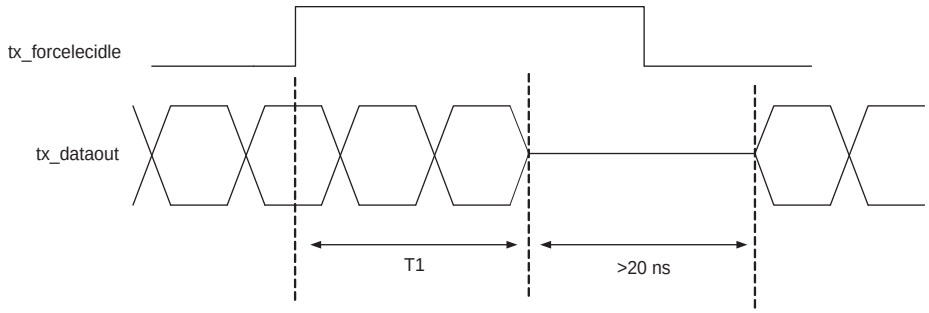
Transmitter Buffer Electrical Idle

When the input signal `tx_forceelecidle` is asserted high, the PCI Express (PIPE) interface block puts the transmitter buffer in that channel in the electrical idle state. During electrical idle, the transmitter buffer differential and common mode output voltage levels are compliant to the PCI Express (PIPE) Base Specification 2.0 for both PCI Express (PIPE) Gen1 and Gen2 data rates.

Figure 1–104 shows the relationship between the assertion of the `tx_forceelecidle` signal and the transmitter buffer output on the `tx_dataout` port. Time T1 taken from the assertion of the `tx_forceelecidle` signal to the transmitter buffer reaching electrical idle voltage levels is pending characterization. Once in the electrical idle state, the PCI Express (PIPE) protocol requires the transmitter buffer to stay in electrical idle for a minimum of 20 ns for both Gen1 and Gen2 data rates.

The minimum period of time for which the `tx_forceelecidle` signal must be asserted high such that the transmitter buffer stays in electrical idle state for at least 20 ns is pending characterization.

Figure 1–104. Transmitter Buffer Electrical Idle State



The PCI Express (PIPE) specification requires the transmitter buffer to be in electrical idle in certain power states. For more information about the `tx_forceelecidle` signal levels required in different PCI Express (PIPE) power states, refer to Table 1–50 on page 1–134.

Receiver Detection

During the detect substate of the link training and status state machine (LTSSM), the PCI Express (PIPE) protocol requires the transmitter channel to perform a receiver detect sequence to detect if a receiver is present at the far end of each lane. The PCI Express (PIPE) specification requires the receiver detect operation to be performed during the P1 power state.

The PCI Express (PIPE) interface block in Stratix IV GX and GT transceivers provide an input signal `tx_detectrxloopback` for the receiver detect operation. When the input signal `tx_detectrxloopback` is asserted high in the P1 power state, the PCI Express (PIPE) interface block sends a command signal to the transmitter buffer in that channel to initiate a receiver detect sequence. In the P1 power state, the transmitter buffer must always be in the electrical idle state. After receiving this command signal, the receiver detect circuitry creates a step voltage at the output of the transmitter buffer. If an active receiver (that complies with the PCI Express [PIPE] input impedance requirements) is present at the far end, the time constant of the step voltage on the trace is higher when compared with the time constant of the step voltage when the receiver is not present. The receiver detect circuitry monitors the time constant of the step signal seen on the trace to determine if a receiver was detected. Receiver detect circuitry monitor needs a 125-MHz clock for operation that you must drive on the `fixedclk` port.

For the receiver detect circuitry to function reliably, the AC-coupling capacitor on the serial link and the receiver termination values used in your system must be compliant to the PCI Express (PIPE) Base Specification 2.0.

Receiver detect circuitry communicates the status of the receiver detect operation to the PCI Express (PIPE) interface block. If a far-end receiver is successfully detected, the PCI Express (PIPE) interface block asserts `pipephydonestatus` for one clock cycle and synchronously drives the `pipestatus[2:0]` signal to 3'b011. If a far-end receiver is not detected, the PCI Express (PIPE) interface block asserts `pipephydonestatus` for one clock cycle and synchronously drives the `pipestatus[2:0]` signal to 3'b000.

Figure 1–105 and Figure 1–106 show the receiver detect operation where a receiver was successfully detected and where a receiver was not detected, respectively.

Figure 1–105. Receiver Detect, Successfully Detected

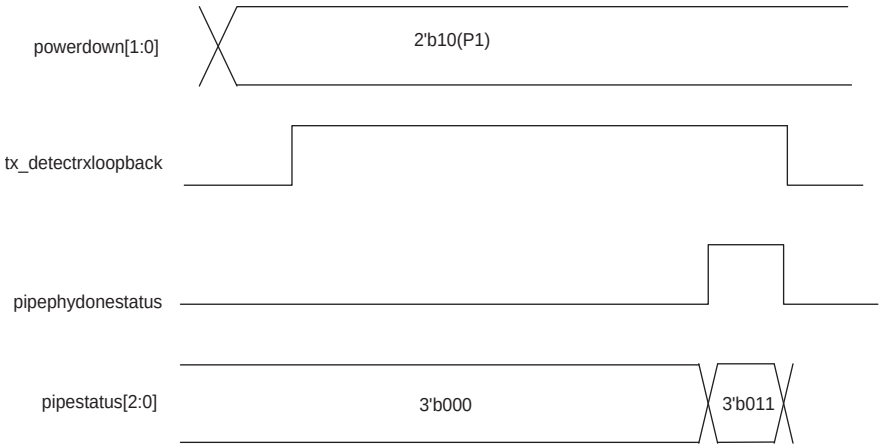
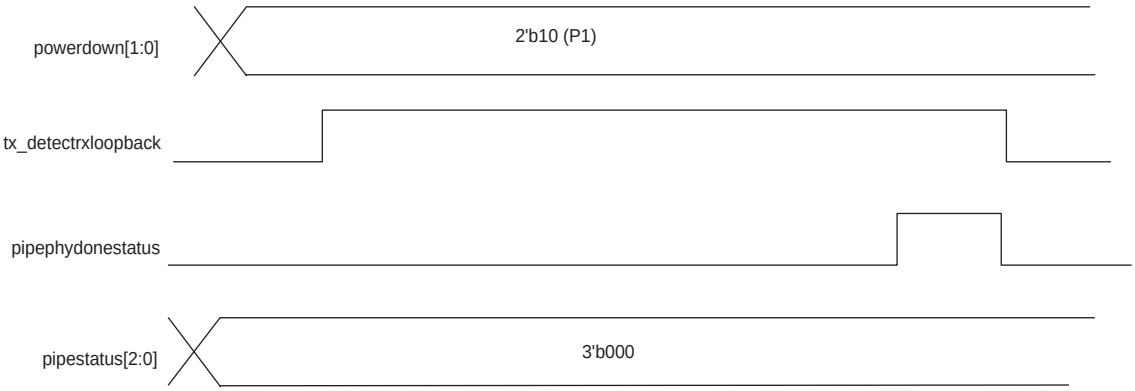


Figure 1–106. Receiver Detect, Unsuccessfully Detected



Compliance Pattern Transmission Support

The LTSSM state machine can enter the polling.compliance substate where the transmitter is required to transmit a compliance pattern as specified in the PCI Express (PIPE) Base Specification 2.0. The polling.compliance substate is intended to assess if the transmitter is electrically compliant with the PCI Express (PIPE) voltage and timing specifications.

The compliance pattern is a repeating sequence of the following four code groups:

- /K28.5/
- /D21.5/
- /K28.5/
- /D10.2/

The PCI Express (PIPE) protocol requires the first /K28.5/ code group of the compliance pattern to be encoded with negative current disparity. To satisfy this requirement, the PCI Express (PIPE) interface block provides the input signal `tx_forcedispcompliance`. A high level on `tx_forcedispcompliance` forces the associated parallel transmitter data on the `tx_datain` port to transmit with negative current running disparity.

- For 8-bit transceiver channel width configurations, you must drive `tx_forcedispcompliance` high in the same parallel clock cycle as the first /K28.5/ of the compliance pattern on the `tx_datain` port.
- For 16-bit transceiver channel width configurations, you must drive only the LSB of `tx_forcedispcompliance[1:0]` high in the same parallel clock cycle as /K28.5/D21.5/ of the compliance pattern on the `tx_datain` port.

Figure 1–107 and Figure 1–108 show the required level on the `tx_forcedispcompliance` signal while transmitting the compliance pattern in 8-bit and 16-bit channel width configurations, respectively.

Figure 1–107. Compliance Pattern Transmission Support, 8-Bit Channel Width Configurations

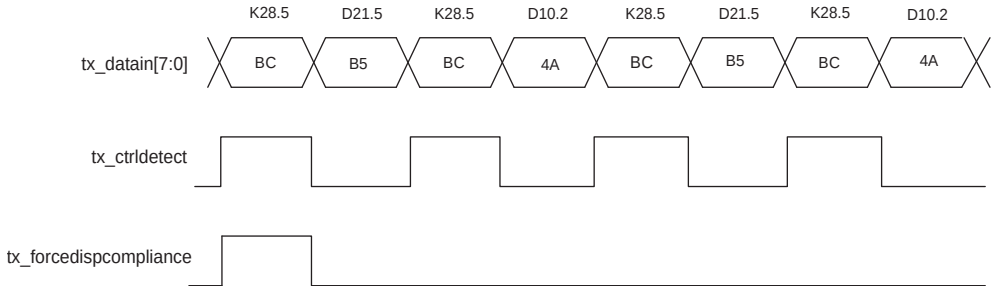
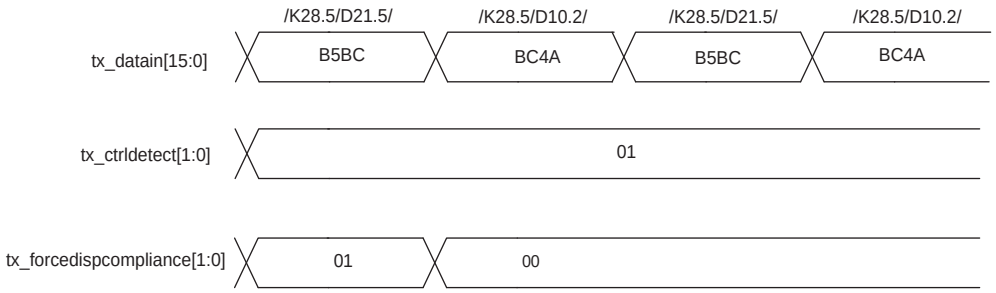


Figure 1–108. Compliance Pattern Transmission Support, 16-Bit Wide Channel Configurations



Power State Management

The PCI Express (PIPE) specification defines four power states—P0, P0s, P1, and P2—that the physical layer device must support to minimize power consumption.

- P0 is the normal operating state during which packet data is transferred on the PCI Express (PIPE) link.
- P0s, P1, and P2 are low-power states into which the physical layer must transition as directed by the PHY-MAC layer to minimize power consumption.

The PCI Express (PIPE) specification provides the mapping of these power states to the LTSSM states specified in the PCI Express (PIPE) Base Specification 2.0. The PHY-MAC layer is responsible for implementing the mapping logic between the LTSSM states and the four power states in the PCI Express (PIPE)-compliant PHY.

The PCI Express (PIPE) interface in Stratix IV GX and GT transceivers provides an input port, `powerdn[1:0]`, for each transceiver channel configured in PCI Express (PIPE) mode. Table 1–49 shows mapping between the logic levels driven on the `powerdn[1:0]` port and the resulting power state that the PCI Express (PIPE) interface block puts the transceiver channel into.

Table 1–49. Power State Functions and Descriptions

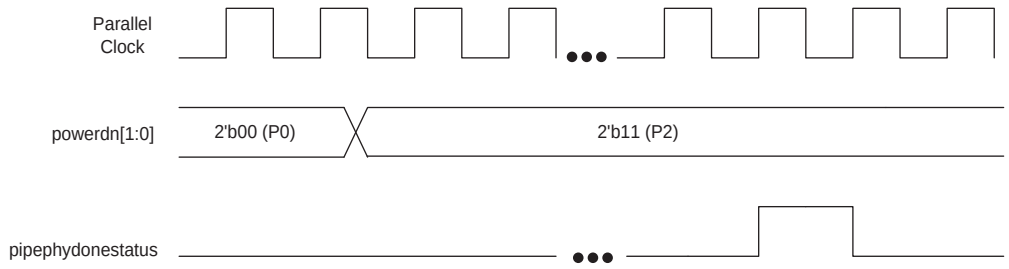
Power State	powerdn	Function	Description
P0	2'b00	Transmits normal data, transmits electrical idle, or enters into loopback mode	Normal operation mode
P0s	2'b01	Only transmits electrical idle	Low recovery time saving state
P1	2'b10	Transmitter buffer is powered down and can do a receiver detect while in this state	High recovery time power saving state
P2	2'b11	Transmits electrical idle or a beacon to wake up the downstream receiver	Lowest power saving state

When transitioning from the P0 power state to lower power states (P0s, P1, and P2), the PCI Express (PIPE) specification requires the physical layer device to implement power saving measures. Stratix IV GX and GT transceivers do not implement these power saving measures except putting the transmitter buffer in electrical idle in the lower power states.

The PCI Express (PIPE) interface block indicates successful power state transition by asserting the `pipephydonestatus` signal for one parallel clock cycle as specified in the PCI Express (PIPE) specification. The PHY-MAC layer must not request any further power state transition until the `pipephydonestatus` signal has indicated the completion of the current power state transition request.

Figure 1–109 shows an example waveform for a transition from the P0 to P2 power state.

Figure 1–109. Power State Transition from the P0 to P2 Power State



The PCI Express (PIPE) specification allows the PCI Express (PIPE) interface to perform protocol functions; for example, receiver detect, loopback, and beacon transmission, in specified power states only. This requires the PHY-MAC layer to drive the `tx_detectrxloopback` and `tx_forceelecidle` signals appropriately in each power state to perform these functions. Table 1–50 summarizes the logic levels that the PHY-MAC layer must drive on the `tx_detectrxloopback` and `tx_forceelecidle` signals in each power state.

Table 1–50. Logic Levels for tx_detectrxloopback and tx_forceelecidle in Different Power States

Power State	tx_detectrxloopback	tx_forceelecidle
P0	0: normal mode 1: datapath in loopback mode	0: Must be de-asserted 1: Illegal mode
P0s	Don't care	0: Illegal mode 1: Must be asserted in this state
P1	0: Electrical Idle 1: receiver detect	0: Illegal mode 1: Must be asserted in this state
P2	Don't care	De-asserted in this state for sending beacon. Otherwise asserted.

Receiver Status

The PCI Express (PIPE) specification requires the PHY to encode the receiver status on a 3-bit `RxStatus[2:0]` signal. This status signal is used by the PHY-MAC layer for its operation.

The PCI Express (PIPE) interface block receives status signals from the transceiver channel PCS and PMA blocks and encodes the status on the 3-bit output signal `pipestatus[2:0]` to the FPGA fabric. The encoding of the status signals on `pipestatus[2:0]` is compliant with the PCI Express (PIPE) specification and is listed in Table 1–51.

Table 1–51. Encoding of the Status Signals on pipestatus[2:0]

pipestatus[2:0]	Description	Error Condition Priority
3'b000	Received data OK	N/A
3'b001	One SKP symbol added	5
3'b010	One SKP symbol deleted	6
3'b011	Receiver detected	N/A
3'b100	8B/10B decode error	1
3'b101	Elastic buffer (rate match FIFO) overflow	2
3'b110	Elastic buffer (rate match FIFO) underflow	3
3'b111	Received disparity error	4

Two or more of the error conditions (for example, 8B/10B decode error [code group violation], rate match FIFO overflow or underflow, and receiver disparity error), can occur simultaneously. The PCI Express (PIPE) interface follows the priority listed in Table 1–51 while encoding the receiver status on the `pipestatus[2:0]` port. For example, if the PCI Express (PIPE) interface receives an 8B/10B decode error and disparity error for the same symbol, it drives 3'b100 on the `pipestatus[2:0]` signal.

Fast Recovery Mode

The PCI Express (PIPE) Base specification fast training sequences (FTS) are used for bit and byte synchronization to transition from L0s to L0 (PCI Express [PIPE] P0s to P0) power states. When transitioning from the L0s to L0 power state, the PCI Express (PIPE) Base Specification requires the physical layer device to acquire bit and byte synchronization after receiving a maximum of 255 FTS (~4 us at Gen1 data rate and ~2 us at Gen2 data rate).

If you have configured the Stratix IV GX and GT receiver CDR in Automatic Lock mode, the receiver cannot meet the PCI Express (PIPE) specification of acquiring bit and byte synchronization within 4 µs (Gen1 data rate) or 2 µs (Gen2 data rate) due to the signal detect and PPM detector time. To meet this specification, each Stratix IV GX and GT transceiver has a built-in Fast Recovery circuitry that you can optionally enable.

To enable the Fast Recovery circuitry, select the **Enable fast recovery mode** option in the ALTGX MegaWizard Plug-In Manager.

If you enable the **Fast Recovery mode** option, the Fast Recovery circuitry controls the receiver CDR `rx_locktorefclk` and `rx_locktodata` signals to force the receiver CDR in LTR or LTD mode. It relies on the Electrical Idle Ordered Sets (EIOS), N_FTS sequences received in the L0 power state, and the signal detect signal from the receiver input buffer to control the receiver CDR lock mode.

The Fast Recovery circuitry is self-operational and does not require control inputs from you. When enabled, the `rx_locktorefclk` and `rx_locktodata` ports are not available in the ALTGX MegaWizard Plug-In Manager.

Electrical Idle Inference

The PCI Express (PIPE) protocol allows inferring the electrical idle condition at the receiver instead of detecting the electrical idle condition using analog circuitry. Clause 4.2.4.3 in the PCI Express (PIPE) Base Specification 2.0 specifies conditions to infer electrical idle at the receiver in various substates of the LTSSM state machine.

In all PCI Express (PIPE) modes (×1, ×4, and ×8), each receiver channel PCS has an optional Electrical Idle Inference module designed to implement the electrical idle inference conditions specified in the PCI Express (PIPE) Base Specification 2.0. You can enable the Electrical Idle Inference module by selecting the **Enable electrical idle inference functionality** option in the ALTGX MegaWizard Plug-In manager.

If enabled, this module infers electrical idle depending on the logic level driven on the `rx_elecidleinfersel[2:0]` input signal. The Electrical Idle Inference module in each receiver channel indicates whether the electrical idle condition is inferred or not on the `pipeelecidle` signal of that channel. The Electrical Idle Interface module drives the `pipeelecidle` signal high if it infers an electrical idle condition; otherwise, it drives it low.

Table 1–52 shows electrical idle inference conditions specified in the PCI Express (PIPE) Base Specification 2.0 and implemented in the Electrical Idle Inference module to infer electrical idle in various substates of the LTSSM state machine. For the Electrical Idle Inference Module to correctly infer an electrical idle condition in each LTSSM substate, you must drive the `rx_elecidleinfersel[2:0]` signal appropriately, as shown in Table 1–52.

Table 1–52. Electrical Idle Inference Conditions

LTSSM State	Gen1 (2.5 Gbps)	Gen2 (5 Gbps)	rx_elecidleinfersel[2:0]
L0	Absence of skip ordered set in 128 µs window	Absence of skip ordered set in 128 µs window	3'b100
Recovery.RcvrCfg	Absence of TS1 or TS2 ordered set in 1280 UI interval	Absence of TS1 or TS2 ordered set in 1280 UI interval	3'b101
Recovery.Speed when successful speed negotiation = 1'b1	Absence of TS1 or TS2 ordered set in 1280 UI interval	Absence of TS1 or TS2 ordered set in 1280 UI window	3'b101
Recovery.Speed when successful speed negotiation = 1'b0	Absence of an exit from Electrical Idle in 2000 UI interval	Absence of an exit from Electrical Idle in 16000 UI interval	3'b110
Loopback.Active (as slave)	Absence of an exit from Electrical Idle in 128 µs window	N/A	3'b111

In the Recovery.Speed substate of the LTSSM state machine with unsuccessful speed negotiation (`rx_elecidleinfersel[2:0]` = `3'b110`), the PCI Express (PIPE) Base Specification requires the receiver to infer an electrical idle condition (`pipeelecidle` = high) if absence of an exit from Electrical Idle is detected in a 2000 UI interval for Gen1 data rate and 16000 UI interval for Gen2 data rate. The electrical idle inference module detects an absence of exit from Electrical Idle if four /K28.5/ COM code groups are not received in the specified interval.

In other words, when configured for Gen1 data rate and `rx_elecidleinfersel[2:0]` = 3'b110, the Electrical Idle Inference module asserts `pipeelecidle` high if it does not receive four /K28.5/ COM code groups in a 2000 UI interval. When configured for Gen1 data rate and `rx_elecidleinfersel[2:0]` = 3'b111 in the Loopback.Active substate of the LTSSM state machine, the Electrical Idle Inference module asserts `pipeelecidle` high if it does not receive four /K28.5/ COM code groups in a 128 µs interval.

When configured for Gen2 data rate and `rx_elecidleinfersel[2:0]` = 3'b110, the Electrical Idle Inference module asserts `pipeelecidle` high if it does not receive four /K28.5/ COM code groups in a 16000 UI interval.

☞ The Electrical Idle Inference module does not have the capability to detect the electrical idle exit condition based on reception of the electrical idle exit ordered set (EIEOS), as specified in the PCI Express (PIPE) Base Specification.

If you select the **Enable Electrical Idle Inference Functionality** option in the ALTGX MegaWizard Plug-In Manager and drive `rx_elecidleinfersel[2:0] = 3'b0xx`, the Electrical Idle Inference block uses the EIOS detection from the Fast Recovery circuitry to drive the `pipeelecidle` signal.

If you do not select the **Enable electrical idle inference functionality** option in the ALTGX MegaWizard Plug-In Manager, the Electrical Idle Inference module is disabled. In this case, the `rx_signaldetect` signal from the signal detect circuitry in the receiver buffer is inverted and driven as the `pipeelecidle` signal.

Recommendation When Using the Electrical Idle Inference Block

In a PCI Express (PIPE) link, when operating at Gen2 data rate, the downstream device can go into the Disable state upon instruction from the upper layer. After in the Disable state, the downstream device must detect an Electrical Idle Exit condition to go into the Detect state. At this same time, the upstream device can be directed by the upper layer to go into the Detect state and start transmitting the COM symbols to the downstream device at Gen 1 data rate.

☞ The Disable and Detect states are different states of the Link Training and Status State Machine as described by the PCI Express (PIPE) Base Specification Rev 2.0.

☞ The COM symbol is an 8B/10B encoded value of K28.5 and is part of the training sequences TS1 and TS2 as described by the PCI Express (PIPE) Base Specification Rev 2.0.

When the Stratix IV GX and GT device is operating as a downstream device at PCI Express (PIPE) Gen 2 data rates and if it goes into the Disable State, the Stratix IV GX and GT receiver must receive an Electrical Idle Exit condition in order to move out of the Disable state.

For the Stratix IV GX and GT receiver, the Electrical Idle Exit condition is achieved when COM symbols are received from the upstream device. However, after the Disable state is achieved by the Stratix IV GX and GT receiver (the downstream device) during Gen 2 data rate operation, and if at the same time the upstream device is directed to transition to the Detect state, the upstream device starts to send COM symbols at Gen 1 data rate. Consequently, the Stratix IV GX and GT receiver (the downstream device) does not recognize the COM symbols as it is operating at Gen 2

data rate. To avoid this scenario, the Link Training Status State Machine (LTSSM) in the FPGA fabric of Stratix IV GX and GT receiver (the downstream device) must be implemented in such a way that whenever the downstream device goes into the Disable state and the upstream device is directed to go into the Detect state, the rateswitch signal must be transitioned from high to low. This allows the Stratix IV GX and GT receiver (the downstream device) to move from Gen 2 to Gen 1 data rate. Subsequently, the Stratix IV GX and GT receiver (the downstream device) recognizes the COM symbols being sent by the upstream device at Gen 1 data rates and moves from the Disable state to the Detect state.

PCI Express (PIPE) Gen2 (5 Gbps) Support

The PCI Express (PIPE) functional mode supports the following additional features when configured for 5 Gbps data rate:

- Dynamic switch between 2.5 Gbps and 5 Gbps signaling rate
- Dynamically selectable transmitter margining for differential output voltage control
- Dynamically selectable transmitter buffer de-emphasis of -3.5 db and -6 dB

Dynamic Switch Between Gen1 (2.5 Gbps) and Gen2 (5 Gbps) Signaling Rate

During link training, the upstream and downstream PCI Express (PIPE) ports negotiate the speed (2.5 Gbps or 5 Gbps) at which the link operates. Because the upstream and downstream PCI Express (PIPE) ports do not know the speed capabilities of their link partner, the PCI Express (PIPE) protocol requires each port to start with a Gen1 (2.5 Gbps) signaling rate. One of the ports capable of supporting the Gen2 (5 Gbps) signaling rate might initiate a speed change request by entering the Recovery state of the LTSSM. In the Recovery state, each port advertises its speed capabilities by transmitting training sequences as specified in the PCI Express (PIPE) Base Specification 2.0. If both ports are capable of operating at the Gen2 (5 Gbps) signaling rate, the PHY-MAC layer instructs the physical layer device to operate at the Gen2 (5 Gbps) signaling rate.

To support speed negotiation during link training, the PCI Express (PIPE) specification requires a PCI Express (PIPE)-compliant physical layer device to provide an input signal (`Rate`) to the PHY-MAC layer. When this input signal is driven low, the physical layer device must operate at the Gen1 (2.5 Gbps) signaling rate; when driven high, this input signal must operate at the Gen2 (5 Gbps) signaling rate. The PCI Express (PIPE) specification allows the PHY-MAC layer to initiate a signaling rateswitch only in power states P0 and P1 with the transmitter buffer in the Electrical Idle state. The PCI Express (PIPE) specification allows the physical layer device to implement the signaling rateswitch using either of the following approaches:

- Change the transceiver datapath clock frequency, keeping the transceiver interface width constant
- Change the transceiver interface width between 8 bit and 16 bit, keeping the transceiver clock frequency constant

When configured in PCI Express (PIPE) functional mode at Gen2 (5 Gbps) data rate, the ALTGX MegaWizard Plug-In Manager provides the input signal `rateswitch`. The `rateswitch` signal is functionally equivalent to the `Rate` signal specified in the PCI Express (PIPE) specification. The PHY-MAC layer can use the `rateswitch` signal to instruct the Stratix IV GX and GT device to operate at either Gen1 (2.5 Gbps)

or Gen2 (5 Gbps) data rate, depending on the negotiated speed between the upstream and downstream ports. A low-to-high transition on the `rateswitch` signal initiates a data rateswitch from Gen1 (2.5 Gbps) to Gen2 (5 Gbps). A high-to-low transition on the `rateswitch` signal initiates a data rateswitch from Gen2 (5 Gbps) to Gen1 (2.5 Gbps). The signaling rateswitch between Gen1 (2.5 Gbps) and Gen2 (5 Gbps) is achieved by changing the transceiver datapath clock frequency between 250 MHz and 500 MHz, while maintaining a constant transceiver interface width of 16-bit.

The dedicated PCI Express (PIPE) rateswitch circuitry performs the dynamic switch between the Gen1 (2.5 Gbps) and Gen2 (5 Gbps) signaling rate. The PCI Express (PIPE) rateswitch circuitry consists of:

- PCI Express (PIPE) rateswitch controller
- PCI Express (PIPE) clock switch circuitry

PCI Express (PIPE) Rateswitch Controller

The `rateswitch` signal serves as the input signal to the PCI Express (PIPE) rateswitch controller. After seeing a transition on the `rateswitch` signal from the PHY-MAC layer, the PCI Express (PIPE) rateswitch controller performs the following operations:

- Controls the PCI Express (PIPE) clock switch circuitry to switch between Gen1 (2.5 Gbps) and Gen2 (5 Gbps) signaling rate depending on the `rateswitch` signal level
- Disables and resets the transmitter and receiver phase compensation FIFO pointers until the PCI Express (PIPE) clock switchover circuitry indicates successful rateswitch completion
- Communicates completion of rateswitch to the PCI Express (PIPE) interface module, which in turn communicates completion of the rateswitch to the PHY-MAC layer on the `pipephydonestatus` signal

PCI Express (PIPE) rateswitch controller location:

- In PCI Express (PIPE) ×1 mode, the PCI Express (PIPE) rateswitch controller is located in the transceiver PCS of each channel.
- In PCI Express (PIPE) ×4 mode, the PCI Express (PIPE) rateswitch controller is located in `CMU0_Channel` within the transceiver block.
- In PCI Express (PIPE) ×8 mode, the PCI Express (PIPE) rateswitch controller is located in `CMU0_Channel` within the master transceiver block.

When operating at the Gen 2 data rate, asserting the `rx_digitalreset` signal causes the PCI Express (PIPE) rateswitch circuitry to switch the transceiver to Gen 1 data rate.

When switching from Gen1 to Gen2 using the dynamic reconfiguration controller, you must set the two ports of the dynamic reconfiguration controller, `tx_preemp_0t` and `tx_preemp_2t`, to zero to meet the Gen2 de-emphasis specifications. When switching from Gen2 to Gen1, if your system requires specific settings on `tx_preemp_01` and `tx_preemp_2t`, those values must be set at the respective two ports of the dynamic reconfiguration controller to meet your system requirements.

PCI Express (PIPE) Clock Switch Circuitry

When the PHY-MAC layer instructs a rateswitch between the Gen1 (2.5 Gbps) and Gen2 (5 Gbps) signaling rates, both the transmitter high-speed serial and low-speed parallel clock and the CDR recovered clock must switch to support the instructed data rate. Stratix IV GX and GT transceivers have dedicated PCI Express (PIPE) clock switch circuitry located in the following blocks:

- Local clock divider in transmitter PMA of each transceiver channel
- `CMU0` clock divider in `CMU0_Channel` of each transceiver block
- Receiver CDR in receiver PMA of each transceiver channel

PCI Express (PIPE) transmitter high-speed serial and low-speed parallel clock switch occurs:

- In PCI Express (PIPE) ×1 mode, the `CMU_PLL` clock switch occurs in the local clock divider in each transceiver channel.
- In PCI Express (PIPE) ×4 mode, the `CMU_PLL` clock switch occurs in the `CMU0` clock divider in the `CMU0_Channel` within the transceiver block.
- In PCI Express (PIPE) ×8 mode, the `CMU_PLL` clock switch occurs in the `CMU0` clock divider in the `CMU0_Channel` within the master transceiver block.

In PCI Express (PIPE) ×1, ×4, and ×8 modes, the recovered clock switch happens in the receiver CDR of each transceiver channel.

Table 1–53 lists the locations of the PCI Express (PIPE) rateswitch controller and the PCI Express (PIPE) clock switch circuitry in PCI Express (PIPE) ×1, ×2, ×4, and ×8 modes.

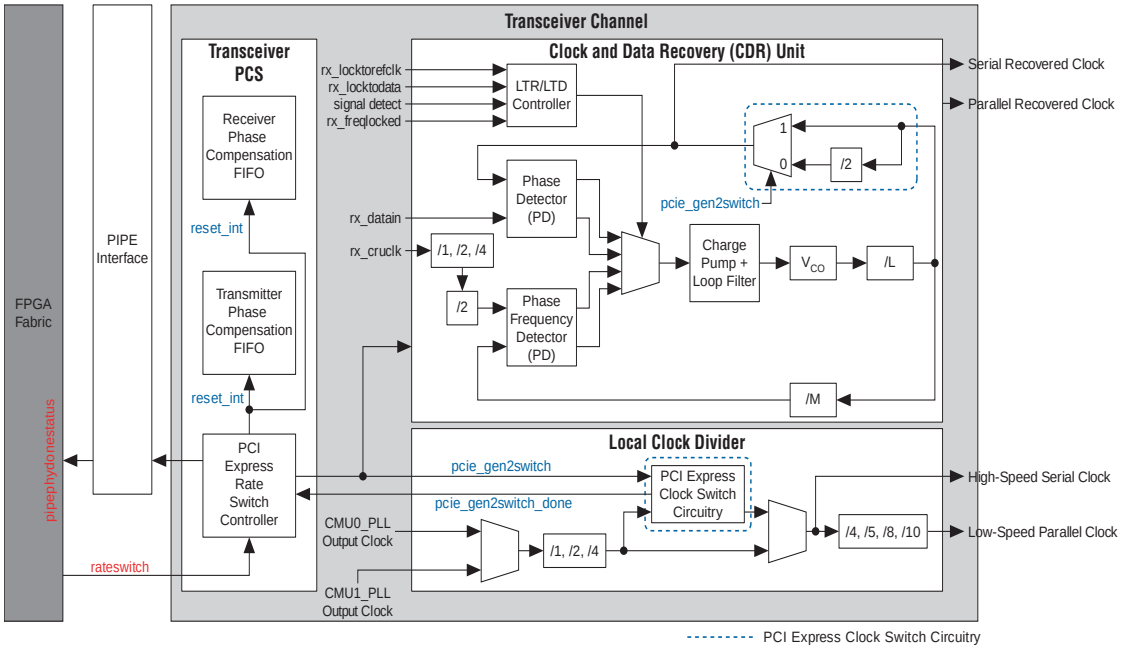
Table 1–53. PCI Express (PIPE) Rateswitch Controller and Clock Switch Circuitry

Channel Bonding Option	Location of PCI Express (PIPE) Rateswitch Controller Module	Location of PCI Express (PIPE) Clock Switch Circuitry	
		Transmitter High-Speed Serial and Low-Speed Parallel Clock Switch Circuitry	Recovered Clock Switch Circuitry
×1	Individual channel PCS block	Local clock divider in transmitter PMA of each channel	CDR block in receiver PMA of each channel
×4	`CMU0_Channel`	CMU0 clock divider in `CMU0_Channel`	CDR block in receiver PMA of each channel
×8	`CMU0_Channel` of the master transceiver block	CMU0 clock divider in `CMU0_Channel` of the master transceiver block	CDR block in receiver PMA of each channel

Dynamic Switch Between Gen1 (2.5 Gbps) and Gen2 (5 Gbps) Signaling Rates in PCI Express (PIPE) x1 Mode

Figure 1–110 shows the PCI Express (PIPE) rateswitch circuitry in PCI Express (PIPE) ×1 mode configured at Gen2 (5 Gbps) data rate.

Figure 1–110. Dynamic Switch Signaling in PIPE ×1 Mode



In PCI Express (PIPE) ×1 mode configured at Gen2 (5 Gbps) data rate, when the PCI Express (PIPE) rateswitch controller sees a transition on the `rateswitch` signal, it sends control signal `pcie_gen2switch` to the PCI Express (PIPE) clock switch circuitry in the local clock divider block and the receiver CDR to switch to the instructed signaling rate. A low-to-high transition on the `rateswitch` signal initiates a Gen1 (2.5 Gbps) to Gen2 (5 Gbps) signaling rateswitch. A high-to-low transition on the `rateswitch` signal initiates a Gen2 (5 Gbps) to Gen1 (2.5 Gbps) signaling rateswitch.

Table 1–54 shows transceiver clock frequencies when switching between 2.5 Gbps and 5 Gbps signaling rates.

Table 1–54. Transceiver Clock Frequencies Signaling Rates in PCI Express (PIPE) ×1 Mode

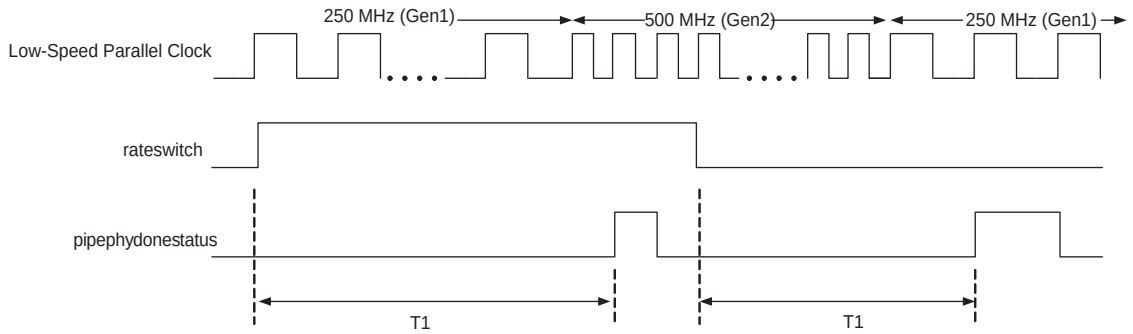
Transceiver Clocks	Gen1 (2.5 Gbps) to Gen2 (5 Gbps) Switch (Low-to-High Transition on the rateswitch Signal)	Gen2 (5 Gbps) to Gen1 (2.5 Gbps) Switch (High-to-Low Transition on the rateswitch Signal)
High-Speed Serial Clock	1.25 GHz to 2.5 GHz	2.5 GHz to 1.25 GHz
Low-Speed Parallel Clock	250 MHz to 500 MHz	500 MHz to 250 MHz
Serial Recovered Clock	1.25 GHz to 2.5 GHz	2.5 GHz to 1.25 GHz
Parallel Recovered Clock	250 MHz to 500 MHz	500 MHz to 250 MHz
FPGA Fabric-Transceiver Interface Clock	125 MHz to 250 MHz	250 MHz to 125 MHz

The PCI Express (PIPE) clock switch circuitry in the local clock divider block performs the clock switch between 250 MHz and 500 MHz on the low-speed parallel clock when switching between Gen1 (2.5 Gbps) and Gen2 (5 Gbps) signaling rates. It indicates successful completion of clock switch on the `pcie_gen2switchdone` signal to the PCI Express (PIPE) rateswitch controller. The PCI Express (PIPE) rateswitch controller forwards the clock switch completion status to the PCI Express (PIPE) interface block. The PCI Express (PIPE) interface block communicates the clock switch completion status to the PHY-MAC layer by asserting the `pipephydonestatus` signal for one parallel clock cycle.

Figure 1–111 shows low-speed parallel clock switch between Gen1 (250 MHz) and Gen2 (500 MHz) in response to the change in the logic level on the `rateswitch` signal. The rateswitch completion is shown marked with a one clock cycle assertion of the `pipephydonestatus` signal.

Time T1 from a transition on the `rateswitch` signal to the assertion of `pipephydonestatus` is pending characterization.

Figure 1–111. Low-Speed Parallel Clock Switching in PCI Express (PIPE) ×1 Mode

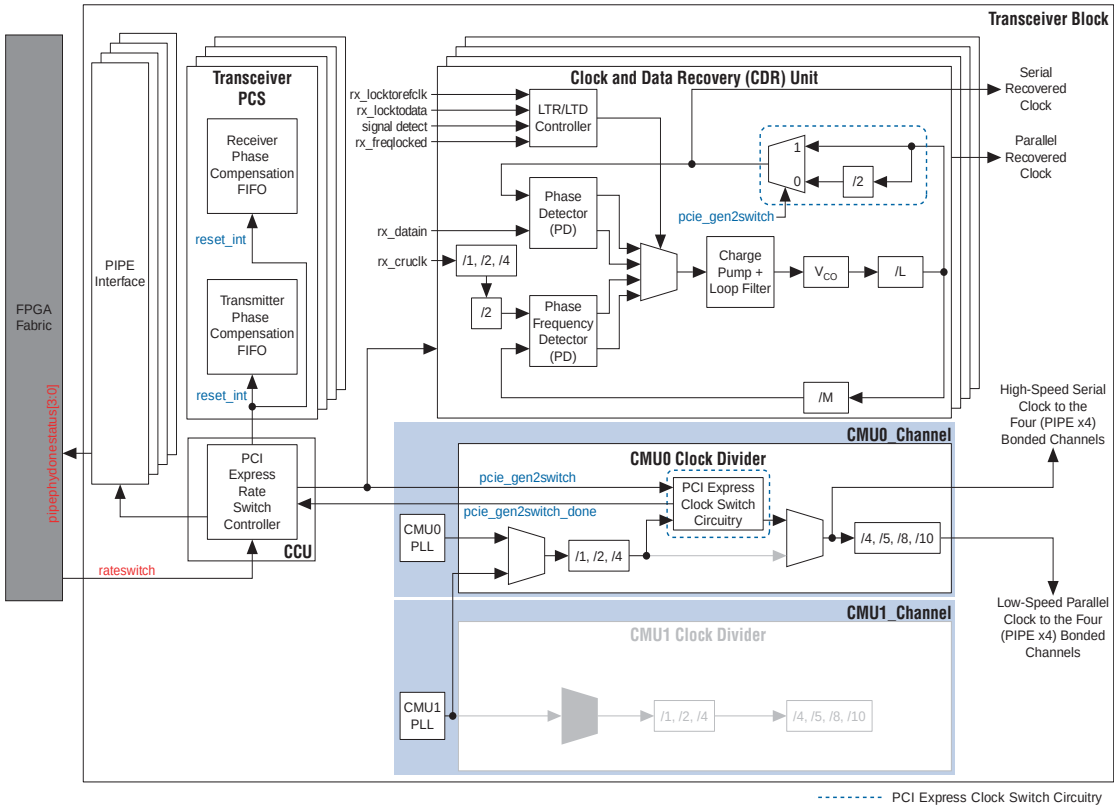


As a result of the signaling rateswitch between Gen1 (2.5 Gbps) and Gen2 (5 Gbps), the FPGA fabric-transceiver interface clock switches between 125 MHz and 250 MHz. The FPGA fabric-transceiver interface clock clocks the read side and write side of the transmitter phase compensation FIFO and the receiver phase compensation FIFO, respectively. It is also routed to the FPGA fabric on a global or regional clock resource and looped back to clock the write port and read port of the transmitter phase compensation FIFO and the receiver phase compensation FIFO, respectively. Due to the routing delay between the write and read clock of the transmitter and receiver phase compensation FIFOs, the write pointers and read pointers might collide during a rateswitch between 125 MHz and 250 MHz. To avoid collision of the phase compensation FIFO pointers, the PCI Express (PIPE) rateswitch controller automatically disables and resets the pointers during clock switch. When the PCI Express (PIPE) clock switch circuitry in the local clock divider indicates successful clock switch completion, the PCI Express (PIPE) rateswitch controller releases the phase compensation FIFO pointer resets.

Dynamic Switch Between Gen1 (2.5 Gbps) and Gen2 (5 Gbps) Signaling Rates in PCI Express (PIPE) ×4 Mode

Figure 1–112 shows the PCI Express (PIPE) rateswitch circuitry in PCI Express (PIPE) ×4 mode configured at Gen2 (5 Gbps) data rate.

Figure 1–112. Dynamic Switch Signaling in PCI Express (PIPE) ×4 Mode



In PCI Express (PIPE) ×4 mode configured at Gen2 (5 Gbps) data rate, when the PCI Express (PIPE) rateswitch controller sees a transition on the `rateswitch` signal, it sends the `pcie_gen2switch` control signal to the PCI Express (PIPE) clock switch circuitry in the `CMU0` clock divider block and the receiver CDR to switch to the instructed signaling rate. A low-to-high transition on the `rateswitch` signal initiates a Gen1 (2.5 Gbps) to Gen2 (5 Gbps) signaling rateswitch. A high-to-low transition on the `rateswitch` signal initiates a Gen2 (5 Gbps) to Gen1 (2.5 Gbps) signaling rateswitch.

Table 1–55 shows the transceiver clock frequencies when switching between 2.5 Gbps and 5 Gbps signaling rates.

Table 1–55. Transceiver Clock Frequencies Signaling Rates in PCI Express (PIPE) ×4 Mode (Part 1 of 2)

Transceiver Clocks	Gen1 (2.5 Gbps) to Gen2 (5 Gbps) Switch (Low-to-High Transition on the rateswitch Signal)	Gen2 (5 Gbps) to Gen1 (2.5 Gbps) Switch (High-to-Low Transition on the rateswitch Signal)
High-Speed Serial Clock	1.25 GHz to 2.5 GHz	2.5 GHz to 1.25 GHz
Low-Speed Parallel Clock	250 MHz to 500 MHz	500 MHz to 250 MHz
Serial Recovered Clock	1.25 GHz to 2.5 GHz	2.5 GHz to 1.25 GHz

Table 1–55. Transceiver Clock Frequencies Signaling Rates in PCI Express (PIPE) ×4 Mode (Part 2 of 2)

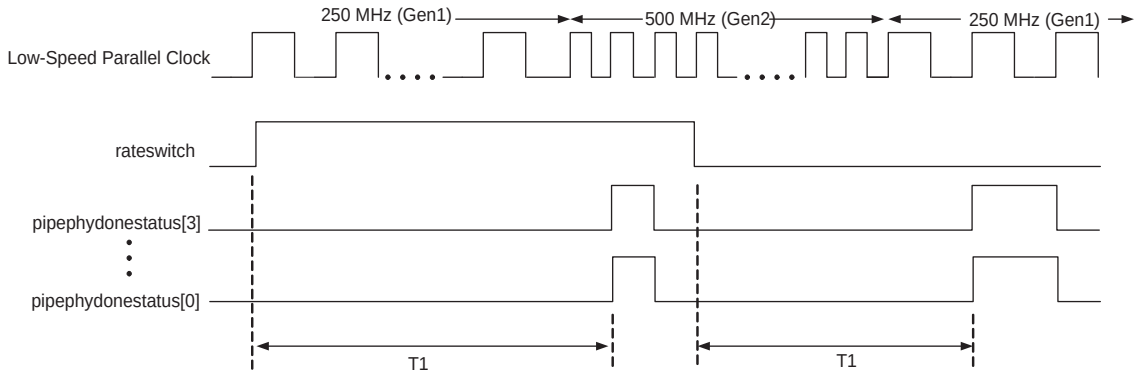
Transceiver Clocks	Gen1 (2.5 Gbps) to Gen2 (5 Gbps) Switch (Low-to-High Transition on the rateswitch Signal)	Gen2 (5 Gbps) to Gen1 (2.5 Gbps) Switch (High-to-Low Transition on the rateswitch Signal)
Parallel Recovered Clock	250 MHz to 500 MHz	500 MHz to 250 MHz
FPGA Fabric-Transceiver Interface Clock	125 MHz to 250 MHz	250 MHz to 125 MHz

The PCI Express (PIPE) clock switch circuitry in the `CMU0` clock divider block performs the clock switch between 250 MHz and 500 MHz on the low-speed parallel clock when switching between Gen1 (2.5 Gbps) and Gen2 (5 Gbps) signaling rates. It indicates successful completion of clock switch on the `pcie_gen2switchdone` signal to the PCI Express (PIPE) rateswitch controller. The PCI Express (PIPE) rateswitch controller forwards the clock switch completion status to the PCI Express (PIPE) interface block. The PCI Express (PIPE) interface block communicates the clock switch completion status to the PHY-MAC layer by asserting the `pipephydonestatus` signal of all bonded channels for one parallel clock cycle.

Figure 1–113 shows the low-speed parallel clock switch between Gen1 (250 MHz) and Gen2 (500 MHz) in response to the change in the logic level on the `rateswitch` signal. The rateswitch completion is shown marked with a one clock cycle assertion of the `pipephydonestatus` signal of all bonded channels.

Time T1 from a transition on the `rateswitch` signal to the assertion of `pipephydonestatus` is pending characterization.

Figure 1–113. Low-Speed Parallel Clock Switching in PCI Express (PIPE) ×4 Mode



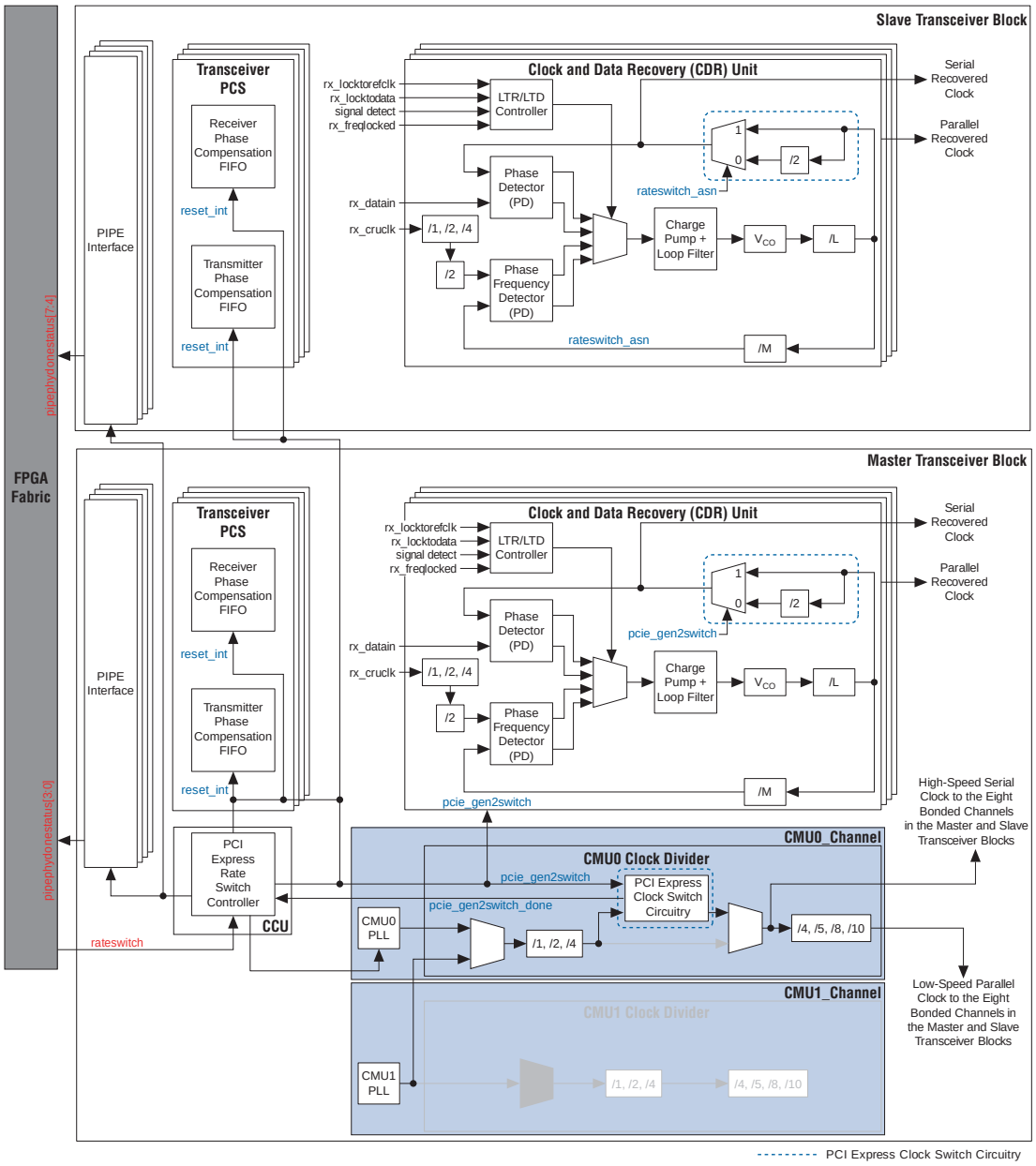
As a result of the signaling rateswitch between Gen1 (2.5 Gbps) and Gen2 (5 Gbps), the FPGA fabric-transceiver interface clock switches between 125 MHz and 250 MHz. The FPGA fabric-transceiver interface clock clocks the read side and write side of the transmitter phase compensation FIFO and the receiver phase compensation FIFO of all bonded channels, respectively. It is also routed to the FPGA fabric on a global or regional clock resource and looped back to clock the write port and read port of the transmitter phase compensation FIFO and the receiver phase compensation FIFO, respectively. Due to the routing delay between the write and read clock of the transmitter and receiver phase compensation FIFOs, the write pointers and read pointers might collide during a rateswitch between 125 MHz and 250 MHz. To avoid collision of the phase compensation FIFO pointers, the PCI Express (PIPE) rateswitch

controller automatically disables and resets the phase compensation FIFO pointers of all bonded channels during clock switch. When the PCI Express (PIPE) clock switch circuitry in the local clock divider indicates successful clock switch completion, the PCI Express (PIPE) rateswitch controller releases the phase compensation FIFO pointer resets.

Dynamic Switch Between Gen1 (2.5 Gbps) and Gen2 (5 Gbps) Signaling Rates in PCI Express (PIPE) ×8 Mode

Figure 1–114 shows the PCI Express (PIPE) rateswitch circuitry in PCI Express (PIPE) ×8 mode configured at Gen2 (5 Gbps) data rate.

Figure 1–114. Dynamic Switch Signaling in PCI Express (PIPE) ×8 Mode



In PCI Express (PIPE) ×8 mode configured at 5 Gbps data rate, when the PCI Express (PIPE) rateswitch controller sees a transition on the `rateswitch` signal, it sends the `pcie_gen2switch` control signal to the PCI Express (PIPE) clock switch circuitry in the `CMU0` clock divider of the master transceiver block and the receiver CDR in all eight bonded channels to switch to the instructed signaling rate. A low-to-high transition on the `rateswitch` signal initiates a Gen1 (2.5 Gbps) to Gen2 (5 Gbps) signaling rateswitch. A high-to-low transition on the `rateswitch` signal initiates a Gen2 (5 Gbps) to Gen1 (2.5 Gbps) signaling rateswitch.

Table 1–56 shows the transceiver clock frequencies when switching between 2.5 Gbps and 5 Gbps signaling rates.

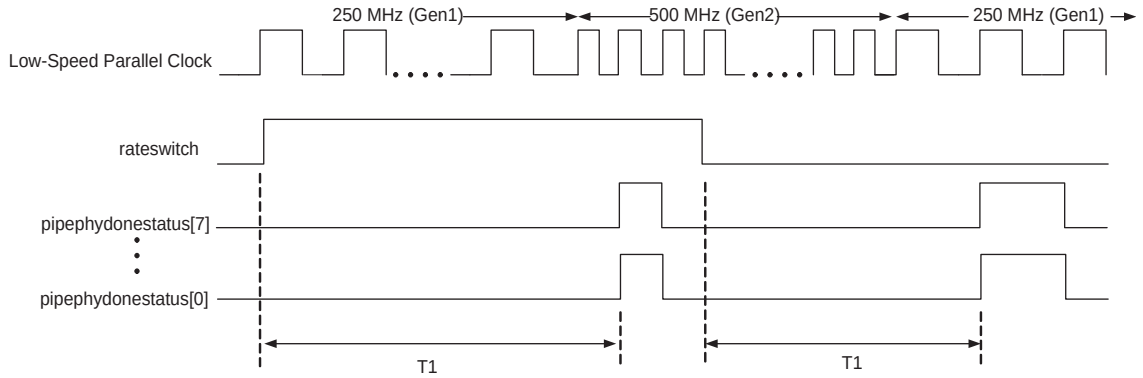
Table 1–56. Transceiver Clock Frequencies Signaling Rates in PCI Express (PIPE) ×8 Mode

Transceiver Clocks	Gen1 (2.5 Gbps) to Gen 2 (5 Gbps) Switch (Low-to-High Transition on the rateswitch Signal)	Gen2 (5 Gbps) to Gen1 (2.5 Gbps) Switch (High-to-Low Transition on the rateswitch Signal)
High-Speed Serial Clock	1.25 GHz to 2.5 GHz	2.5 GHz to 1.25 GHz
Low-Speed Parallel Clock	250 MHz to 500 MHz	500 MHz to 250 MHz
Serial Recovered Clock	1.25 GHz to 2.5 GHz	2.5 GHz to 1.25 GHz
Parallel Recovered Clock	250 MHz to 500 MHz	500 MHz to 250 MHz
FPGA Fabric-Transceiver Interface Clock	125 MHz to 250 MHz	250 MHz to 125 MHz

The PCI Express (PIPE) clock switch circuitry in the `CMU0` clock divider of the master transceiver block performs the clock switch between 250 MHz and 500 MHz on the low-speed parallel clock when switching between Gen1 (2.5 Gbps) and Gen2 (5 Gbps) signaling rates. It indicates successful completion of clock switch on the `pcie_gen2switchdone` signal to the PCI Express (PIPE) rateswitch controller. The PCI Express (PIPE) rateswitch controller forwards the clock switch completion status to the PCI Express (PIPE) interface block. The PCI Express (PIPE) interface block communicates the clock switch completion status to the PHY-MAC layer by asserting the `pipephydonestatus` signal of all eight bonded channels for one parallel clock cycle.

Figure 1–115 shows the low-speed parallel clock switch between Gen1 (250 MHz) and Gen2 (500 MHz) in response to the change in the logic level on the `rateswitch` signal. The rateswitch completion is shown marked with a one clock cycle assertion of the `pipephydonestatus` signal of all eight bonded channels.

Figure 1–115. Low-Speed Parallel Clock Switching in PCI Express (PIPE) ×8 Mode *(Note 1)*



Note to Figure 1–115:

(1) Time T1 from a transition on the `rateswitch` signal to the assertion of `pipephydonestatus` is pending characterization.

As a result of the signaling rateswitch between Gen1 (2.5 Gbps) and Gen2 (5 Gbps), the FPGA fabric-transceiver interface clock switches between 125 MHz and 250 MHz. The FPGA fabric-transceiver interface clock clocks the read side and write side of the transmitter phase compensation FIFO and the receiver phase compensation FIFO of all eight bonded channels, respectively. It is also routed to the FPGA fabric on a global or regional clock resource and looped back to clock the write port and read port of the transmitter phase compensation FIFO and the receiver phase compensation FIFO, respectively. Due to the routing delay between the write and read clock of the transmitter and receiver phase compensation FIFOs, the write pointers and read pointers might collide during a rateswitch between 125 MHz and 250 MHz. To avoid collision of the phase compensation FIFO pointers, the PCI Express (PIPE) rateswitch controller automatically disables and resets the phase compensation FIFO pointers of all eight bonded channels during clock switch. When the PCI Express (PIPE) clock switch circuitry in the local clock divider indicates successful clock switch completion, the PCI Express (PIPE) rateswitch controller releases the phase compensation FIFO pointer resets.

PCI Express (PIPE) Cold Reset Requirements

The PCI Express (PIPE) Base Specification 2.0 defines the following three types of conventional resets to the PCI Express (PIPE) system components:

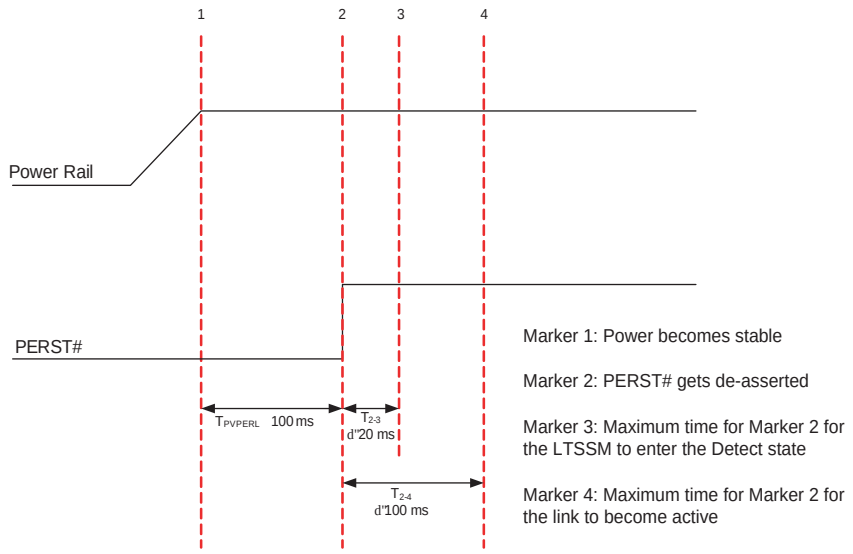
- Cold reset—fundamental reset after power up
- Warm reset—fundamental reset without removal and re-application of power
- Hot reset—In-band conventional reset initiated by the higher layer by setting the Hot Reset bit in the TS1 or TS2 training sequences

Fundamental reset is provided by the system to the component or adapter card using the auxiliary signal PERST#. The PCI Express (PIPE) Base Specification 2.0 specifies that PERST# must be kept asserted for a minimum of 100 ms (TPVPERL) after the system power becomes stable in a cold reset situation. Additionally, all system components must enter the LTSSM Detect state within 20 ms and the link must become active within 100 ms after de-assertion of the PERST# signal. This implies that each PCI Express (PIPE) system component must become active within 200 ms after the power becomes stable.

The link being active is interpreted as the physical layer device coming out of electrical idle in the L0 state of the LTSSM state machine.

Figure 1–116 shows the PCI Express (PIPE) cold reset timing requirements.

Figure 1–116. PCI Express (PIPE) Cold Reset Requirements



The time taken by a PCI Express (PIPE) port implemented using the Stratix IV GX and GT device to go from power up to link active state is described below:

- Power on reset (POR)—begins after power rails become stable. Typically takes 12 ms
- FPGA configuration/programming—begins after POR. Configuration time depends on the FPGA density
- Time taken from de-assertion of PERST# to link active—typically takes 40 ms (pending characterization and verification of PCI Express [PIPE] soft IP and hard IP)

To meet the PCI Express (PIPE) specification of 200 ms from power on to link active, the Stratix IV GX and GT device configuration time must be less than 148 ms (200 ms –12 ms for power on reset and -40 ms for the link to become active after PERST# de-assertion).

Table 1–57 shows the typical configuration times for Stratix IV GX devices when configured using the Fast Passive Parallel (FPP) configuration scheme at 125 MHz.

Table 1–57. Typical Configuration Times for Stratix IV GX Devices Configured with Fast Passive Parallel

Stratix IV GX	Stratix IV GT	Configuration Time (ms)
EP4SGX70	—	48
EP4SGX110	—	48
EP4SGX230	EP4S(40/100)G2	95
EP4SGX290	EP4S100G3	128
EP4SGX360	EP4S100G4	128
EP4SGX530	EP4S(40/100)G5	172

For more information about the FPP configuration scheme, refer to the *Configuration, Design Security, Remote System Upgrades with Stratix IV Devices* chapter.

Most flash memories available can run up to 100 MHz. To configure the Stratix IV GX and GT device at 125 MHz, Altera recommends using a MAX II device to convert the 16-bit flash memory output at 62.5 MHz to 8-bit configuration data input to the Stratix IV GX and GT device at 125 MHz.

PCI Express Electrical Gold Test with Compliance Base Board (CBB)

The PCI Express Electrical Gold Test requires the v2.0 CBB to be connected to the Device Under Test (DUT). The CBB sends out a 100 MHz signal for 1 ms to indicate the Link Training and Status State Machine (LTSSM) of the downstream device Under Test (DUT) to transition to several polling compliance states. Under these states, the DUT sends out data at Gen1, Gen2 (with -3.5db de-emphasis), and Gen2 (with -6 db de-emphasis) rates, which can be observed in the scope to confirm electrical signal compliance. The CBB is DC-coupled to the downstream receiver.

When you use the Stratix IV GX and GT device as DUT, because of being DC-coupled to CBB with a different common mode level, the Stratix IV GX and GT receiver does not receive the required V_{CM} (0.85 V) to detect the signal. The logic in the FPGA fabric that implements LTSSM cannot transition to the multiple polling compliance states to complete the test. Therefore, when testing with the CBB, force the LTSSM implemented in the FPGA fabric to transfer to different polling compliance states using an external push button or user logic.

If you use the Stratix IV GX and GT PCI Express hard IP block, assert the `testin[5]` port of the PCI Express Compiler-generated wrapper file in your design. Asserting this port forces the LTSSM within the hard IP block to transition to these states. The `testin[5]` port must be asserted for a minimum of 16 ns and less than 24 ms.

For more information about the PCI Express (PIPE) hard IP block, refer to the *PCI Express Compiler User Guide*.

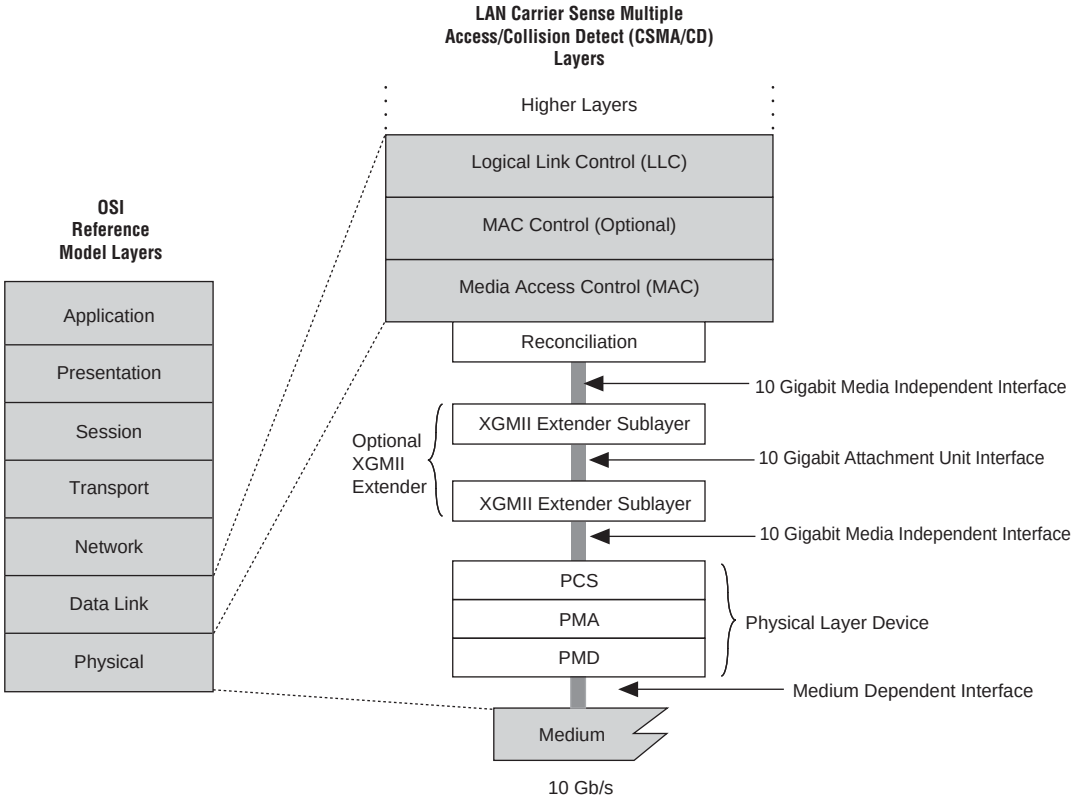
XAUI Mode

XAUI is an optional, self-managed interface that you can insert between the reconciliation sublayer and the PHY layer to transparently extend the physical reach of the XGMII.

XAUI addresses several physical limitations of the XGMII. XGMII signaling is based on the HSTL Class 1 single-ended I/O standard, which has an electrical distance limitation of approximately 7 cm. Because XAUI uses a low-voltage differential signaling method, the electrical limitation is increased to approximately 50 cm. Another advantage of XAUI is simplification of backplane and board trace routing. XGMII is composed of 32 transmit channels, 32 receive channels, 1 transmit clock, 1 receive clock, 4 transmitter control characters, and 4 receive control characters for a 74-pin wide interface. XAUI, on the other hand, only consists of 4 differential transmitter channels and 4 differential receiver channels for a 16-pin wide interface. This reduction in pin count significantly simplifies the routing process in the layout design.

Figure 1–117 shows the relationships between the XGMII and XAUI layers.

Figure 1–117. XAUI and XGMII Layers



The XGMII interface consists of four lanes of 8 bits. At the transmit side of the XAUI interface, the data and control characters are converted within the XGXS into an 8B/10B encoded data stream. Each data stream is then transmitted across a single differential pair running at 3.125 Gbps (3.75 Gbps for HiGig). At the XAUI receiver, the incoming data is decoded and mapped back to the 32-bit XGMII format. This provides a transparent extension of the physical reach of the XGMII and also reduces the interface pin count.

In Stratix IV GX and GT XAUI functional mode, the interface between the transceiver and FPGA fabric is 64 bits wide (four channels of 16 bits each) at single data rate.

XAUI functions as a self-managed interface because code group synchronization, channel deskew, and clock domain decoupling is handled with no upper layer support requirements. This functionality is based on the PCS code groups that are used during the IPG time and idle periods. PCS code groups are mapped by the XGXS to XGMII characters, as specified in Table 1–58.

Table 1–58. XGMII Character to PCS Code-Group Mapping

XGMII TXC	XGMII TXD *(1)*	PCD Code Group	Description
0	00 through FF	Dxx,y	Normal data transmission
1	07	K28.0 or K28.3 or K28.5	Idle in ||I||
1	07	K28.5	Idle in ||T||
1	9C	K28.4	Sequence
1	FB	K27.7	Start
1	FD	K29.7	Terminate
1	FE	K30.7	Error
1	Any other value	K30.7	Invalid XGMII character

Note to Table 1–58:

(1) The values in the XGMII TXD column are in hexadecimal.

Figure 1–118 shows an example of mapping between XGMII characters and the PCS code groups that are used in XAUI. The idle characters are mapped to a pseudo-random sequence of /A/, /R/, and /K/ code groups.

Figure 1–118. Example of Mapping XGMII Characters to PCS Code Groups

XGMII

T/RxD<7..0>	I	I	S	Dp	D	D	D	- - -	D	D	D	D	I	I	I	I	I	I
T/RxD<15..8>	I	I	Dp	Dp	D	D	D	- - -	D	D	D	T	I	I	I	I	I	I
T/RxD<23..16>	I	I	Dp	Dp	D	D	D	- - -	D	D	D	I	I	I	I	I	I	I
T/RxD<31..24>	I	I	Dp	Dp	D	D	D	- - -	D	D	D	I	I	I	I	I	I	I

PCS

Lane 0	K	R	S	Dp	D	D	D	- - -	D	D	D	D	A	R	R	K	K	R
Lane 1	K	R	Dp	Dp	D	D	D	- - -	D	D	D	T	A	R	R	K	K	R
Lane 2	K	R	Dp	Dp	D	D	D	- - -	D	D	D	K	A	R	R	K	K	R
Lane 3	K	R	Dp	Dp	D	D	D	- - -	D	D	D	K	A	R	R	K	K	R

PCS code groups are sent via PCS ordered sets. PCS ordered sets consist of combinations of special and data code groups defined as a column of code groups. These ordered sets are composed of four code groups beginning in lane 0. Table 1–59 lists the defined idle ordered sets (||I||) that are used for the self-managed properties of XAUI.

Table 1–59. Defined Idle Ordered Set

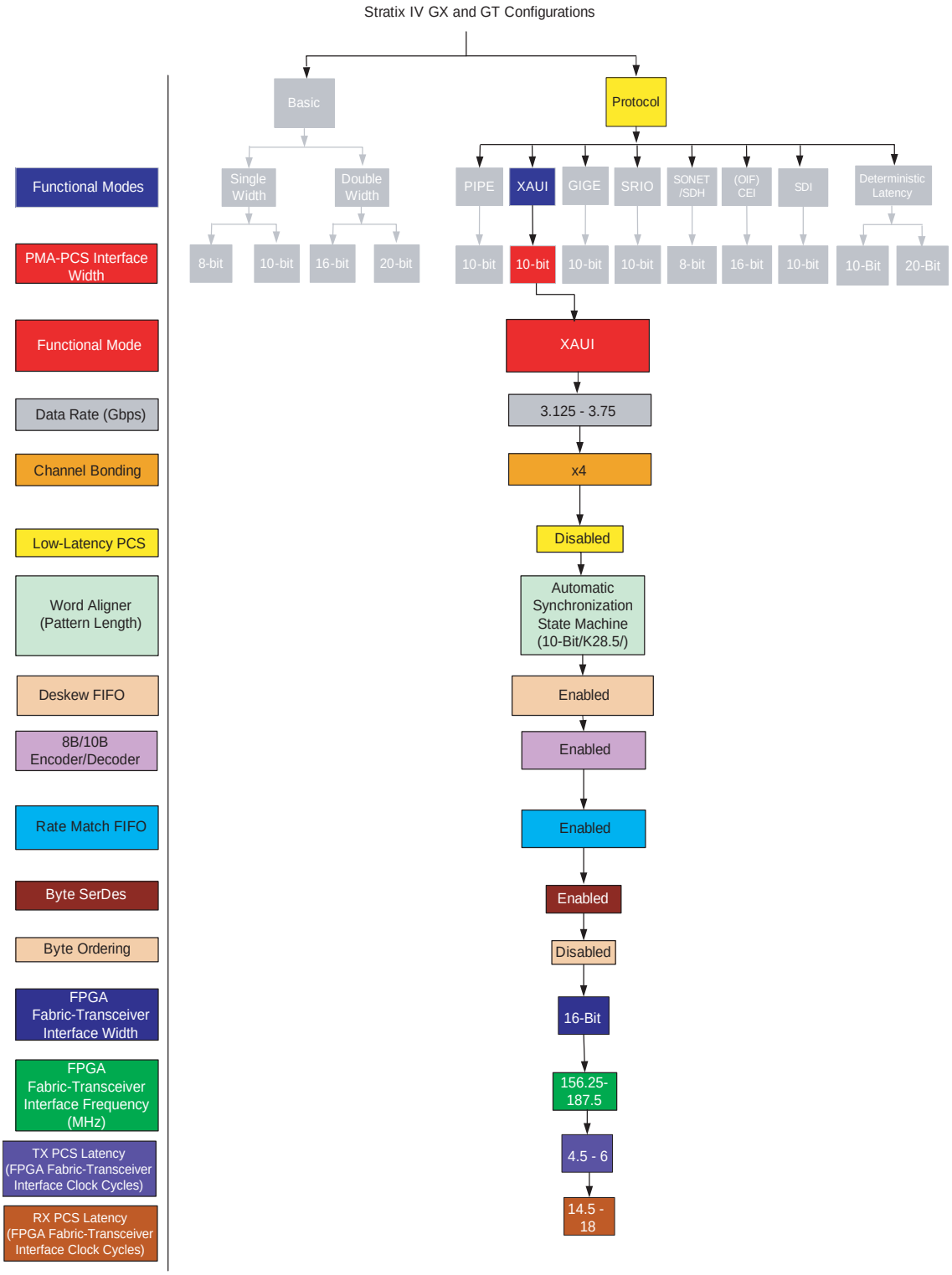
Code	Ordered Set	Number of Code Groups	Encoding
||I||	Idle		Substitute for XGMII Idle
||K||	Synchronization column	4	/K28.5/K28.5/K28.5/K28.5/
||R||	Skip column	4	/K28.0/K28.0/K28.0/K28.0/
||A||	Align column	4	/K28.3/K28.3/K28.3/K28.3/

Stratix IV GX and GT transceivers configured in XAUI mode provide the following protocol features:

- XGMII-to-PCS code conversion at the transmitter
- PCS-to-XGMII code conversion at the receiver
- 8B/10B encoding and decoding
- IEEE P802.3ae-compliant synchronization state machine
- ±100 PPM clock rate compensation
- Channel deskew of four lanes of the XAUI link

Figure 1–119 shows the XAUI mode configuration supported in Stratix IV GX and GT devices.

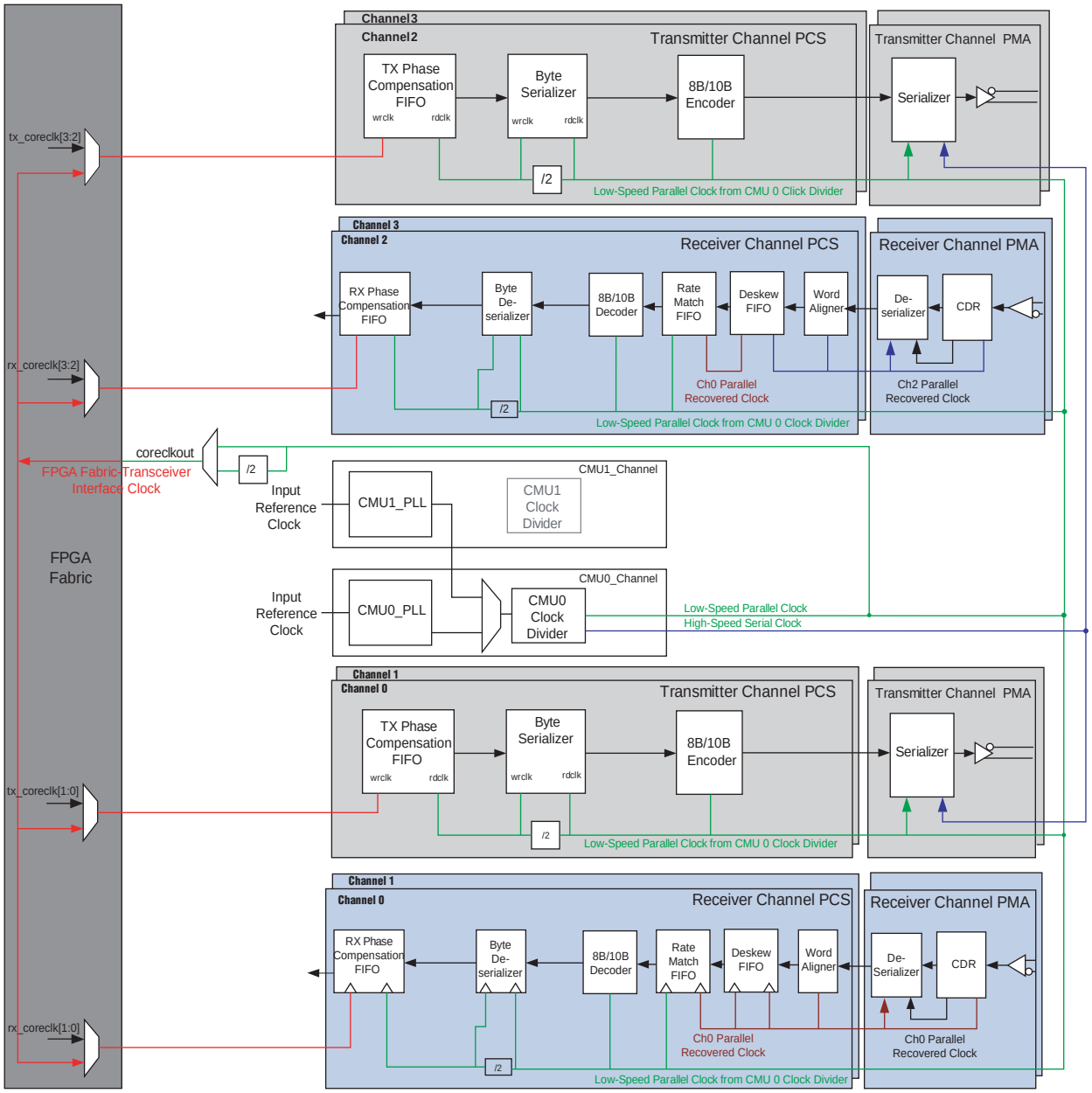
Figure 1–119. Stratix IV GX and GT XAUI Mode Configuration



XAUI Mode Datapath

Figure 1–120 shows the ALTGX megafunction transceiver datapath when configured in XAUI mode.

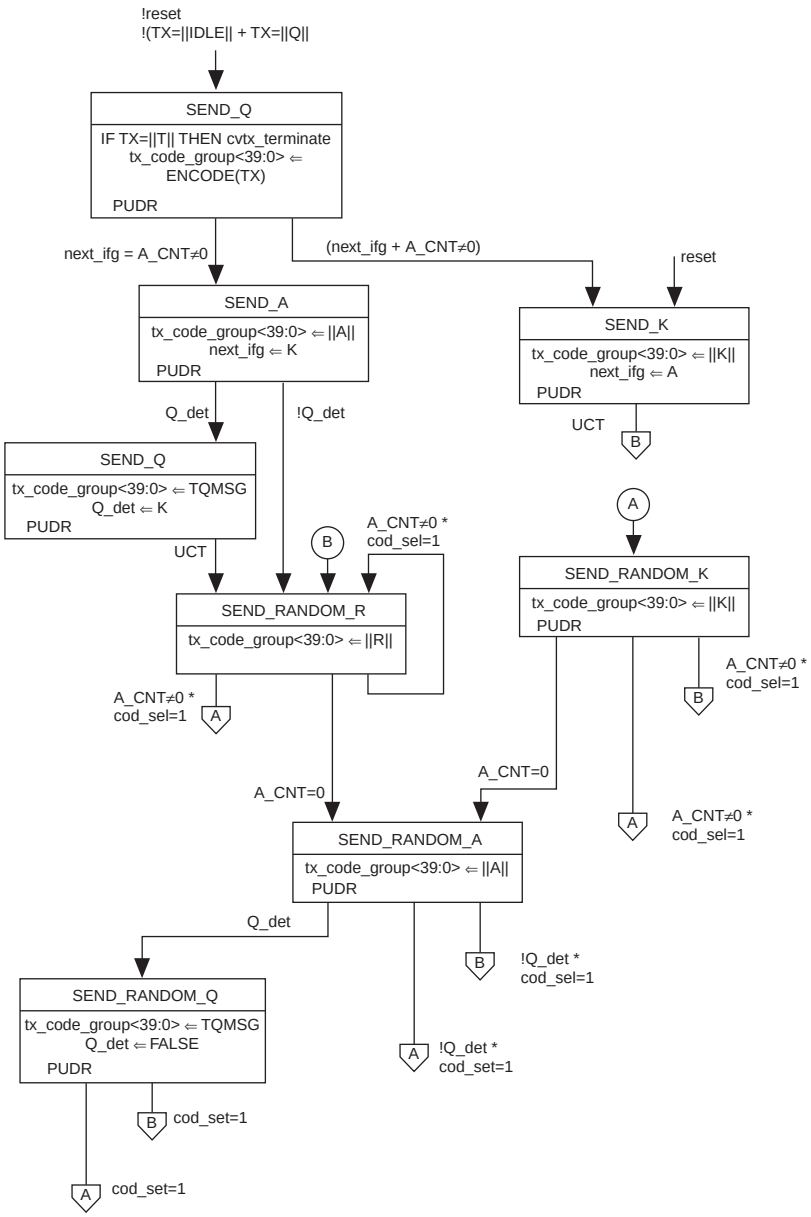
Figure 1–120. Transceiver Datapath in XAUI Mode



XGMII-To-PCS Code Conversion at the Transmitter

In XAUI mode, the 8B/10B encoder in the Stratix IV GX and GT transmitter datapath is controlled by a transmitter state machine that maps various 8-bit XGMII codes to 10-bit PCS code groups. This state machine complies with the IEEE P802.3ae PCS transmit source state diagram shown in Figure 1–121.

Figure 1–121. XGMII-To-PCS Code Conversion in XAUI Mode *(Note 1)*



Note to Figure 1–121:

(1) This figure is from IEEE P802.3ae.

Table 1–60 lists the XGMII-to-PCS code group conversion in XAUI functional mode. The XGMII TXC control signal is equivalent to the `tx_ctrlenable` signal; the XGMII TXD control signal is equivalent to the `tx_datain[7:0]` signal.

Table 1–60. XGMII Character to PCS Code-Group Mapping

XGMII TXC	XGMII TXD *(1)*	PCD Code Group	Description
0	00 through FF	Dxx,y	Normal data transmission
1	07	K28.0 or K28.3 or K28.5	Idle in ||I||
1	07	K28.5	Idle in ||T||
1	9C	K28.4	Sequence
1	FB	K27.7	Start
1	FD	K29.7	Terminate
1	FE	K30.7	Error
1	Any other value	K30.7	Invalid XGMII character

Note to Table 1–58:

(1) The values in the XGMII TXD column are in hexadecimal.

PCS-To-XGMII Code Conversion at the Receiver

In XAUI mode, the 8B/10B decoder in the Stratix IV GX and GT receiver datapath is controlled by a XAUI receiver state machine that converts received PCS code groups into specific 8-bit XGMII codes. This state machine complies with the IEEE P802.3ae specifications.

Table 1–61 lists the PCS-to-XGMII code group conversion in XAUI functional mode. The XGMII RXC control signal is equivalent to the `rx_ctrldetect` signal; the XGMII RXD control signal is equivalent to the `rx_dataout[7:0]` signal.

Table 1–61. PCS Code Group to XGMII Character Mapping

XGMII RXC	XGMII RXD *(1)*	PCD Code Group	Description
0	00 through FF	Dxx,y	Normal data transmission
1	07	K28.0 or K28.3 or K28.5	Idle in ||I||
1	07	K28.5	Idle in ||T||
1	9C	K28.4	Sequence
1	FB	K27.7	Start
1	FD	K29.7	Terminate
1	FE	K30.7	Error
1	FE	Invalid code group	Received code group

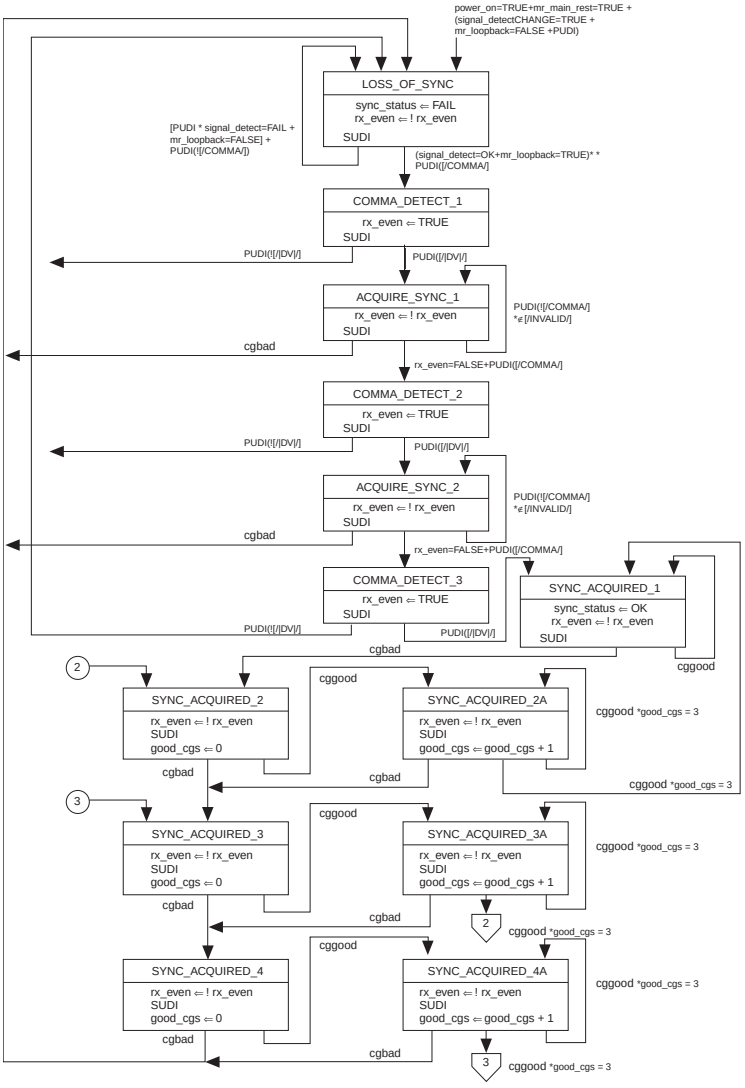
Note to Table 1–58:

(1) The values in the XGMII RXD column are in hexadecimal.

Word Aligner

The word aligner in XAUI functional mode is configured in automatic synchronization state machine mode. The Quartus II software automatically configures the synchronization state machine to indicate synchronization when the receiver receives four /K28.5/ comma code groups without intermediate invalid code groups. The synchronization state machine implemented in XAUI mode is compliant to the PCS synchronization state diagram specified in Clause 48 of the IEEE P802.3ae specification and is shown in Figure 1–122.

Figure 1–122. IEEE 802.3ae PCS Synchronization State Diagram *(Note 1)*



Note to Figure 1–122:

(1) This figure is from IEEE P802.3ae.

Receiver synchronization is indicated on the `rx_syncstatus` port of each channel. A high on the `rx_syncstatus` port indicates that the lane is synchronized; a low on the `rx_syncstatus` port indicates that it has fallen out of synchronization. The receiver loses synchronization when it detects four invalid code groups separated by less than four valid code groups or when it is reset.

Deskew FIFO

Code groups received across four lanes in a XAUI link can be misaligned with respect to one another because of skew in the physical medium or differences between the independent clock recoveries per lane. The XAUI protocol allows a maximum skew of 40 UI (12.8 ns) as seen at the receiver of the four lanes.

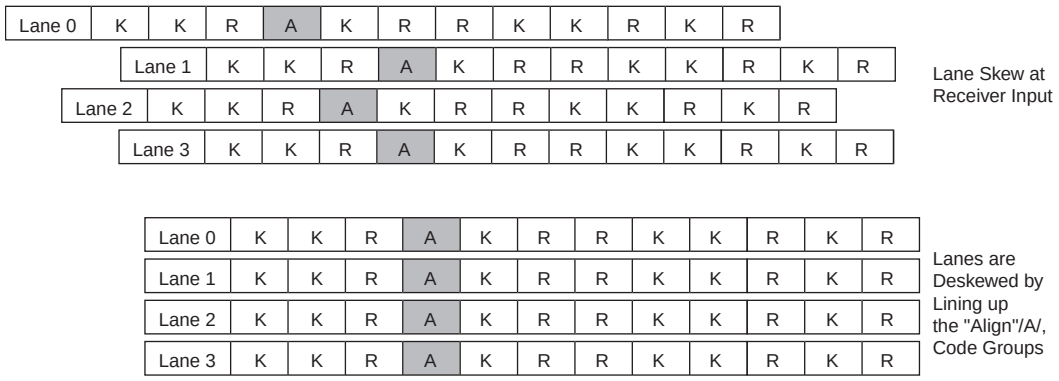
The XAUI protocol requires the physical layer device to implement a deskew circuitry to align all four channels. To enable the deskew circuitry at the receiver to align the four channels, the transmitter sends a /A/ (/K28.3/) code group simultaneously on all four channels during inter-packet gap. The skew introduced in the physical medium and the receiver channels can be /A/ code groups to be received misaligned with respect to each other.

The deskew operation is performed by the deskew FIFO in XAUI functional mode.

The deskew FIFO in each channel receives data from its word aligner. The deskew operation begins only after link synchronization is achieved on all four channels as indicated by a high on the `rx_syncstatus` signal from the word aligner in each channel. Until the first /A/ code group is received, the deskew FIFO read and write pointers in each channel are not incremented. After the first /A/ code group is received, the write pointer starts incrementing for each word received but the read pointer is frozen. If the /A/ code group is received on each of the four channels within 10 recovered clock cycles of each other, the read pointer of all four deskew FIFOs is released simultaneously, aligning all four channels.

Figure 1–123 shows lane skew at the receiver input and how the deskew FIFO uses the /A/ code group to align the channels.

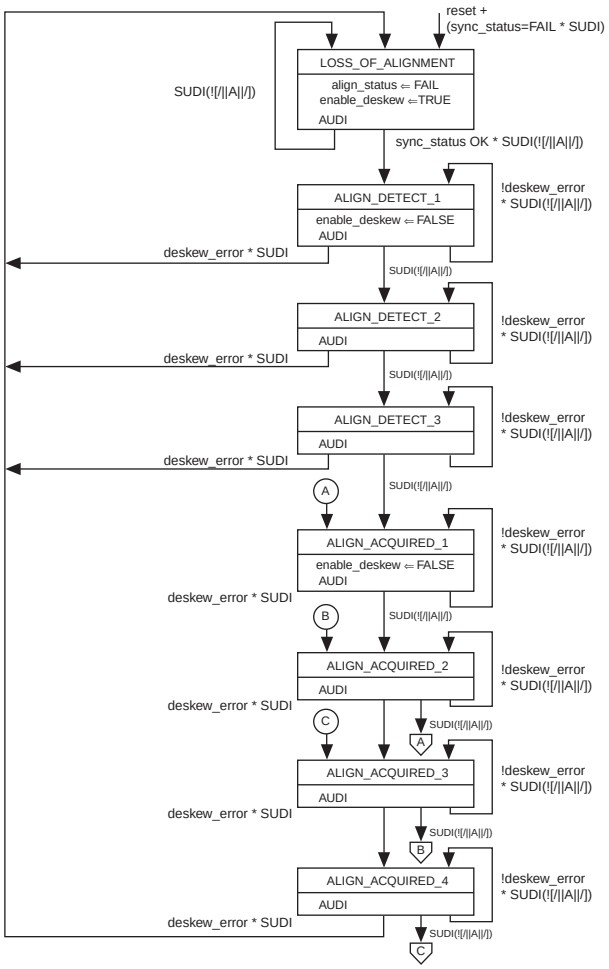
Figure 1–123. Receiver Input Lane Skew in XAUI Mode



After alignment of the first ||A|| column, if three additional aligned ||A|| columns are observed at the output of the deskew FIFOs of the four channels, the `rx_channelaligned` signal is asserted high, indicating channel alignment is acquired. After acquiring channel alignment, if four misaligned ||A|| columns are seen at the output of the deskew FIFOs in all four channels with no aligned ||A|| columns in between, the `rx_channelaligned` signal is de-asserted low, indicating loss-of-channel alignment.

The deskew FIFO operation in XAUI functional mode is compliant with the PCS deskew state machine diagram specified in clause 48 of the IEEE P802.3ae, as shown in Figure 1–124.

Figure 1–124. Deskew FIFO in XAUI Mode *(Note 1)*



Note to Figure 1–124:

(1) This figure is from IEEE P802.3ae.

Rate Match FIFO

In XAUI mode, the rate match FIFO is capable of compensating for up to ±100 PPM (200 PPM total) difference between the upstream transmitter and the local receiver reference clock. The XAUI protocol requires the transmitter to send /R/ (/K28.0/) code groups simultaneously on all four lanes (denoted as ||R|| column) during inter-packet gaps, adhering to rules listed in the IEEE P802.3ae specification. The rate match FIFO operation in XAUI mode is compliant to the IEEE P802.3ae specification.

The rate match operation begins after:

- The synchronization state machine in the word aligner of all four channels indicates synchronization has been acquired by driving the `rx_syncstatus` signal high
- The deskew FIFO indicates alignment has been acquired by driving the `rx_channelaligned` signal high

The rate match FIFO looks for the ||R|| column (simultaneous /R/ code group on all four channels) and deletes or inserts ||R|| column to prevent the rate match FIFO from overflowing or under-running. The rate match FIFO can insert or delete as many ||R|| columns as necessary to perform the rate match operation.

Two flags, `rx_rmfifodatadeleted` and `rx_rmfifodatainserted`, indicating rate match FIFO deletion and insertion events, respectively, are forwarded to the FPGA fabric. If an ||R|| column is deleted, the `rx_rmfifodeleted` flag from each of the four channels goes high for one clock cycle per deleted ||R|| column. If an ||R|| column is inserted, the `rx_rmfifoinserted` flag from each of the four channels goes high for one clock cycle per inserted ||R|| column.

Figure 1–125 shows an example of rate match deletion in the case where three ||R|| columns are required to be deleted.

For more information, refer to "Rate Match FIFO in XAUI Mode" on page 1–77.

Figure 1–125. Rate Match Deletion in XAUI Mode

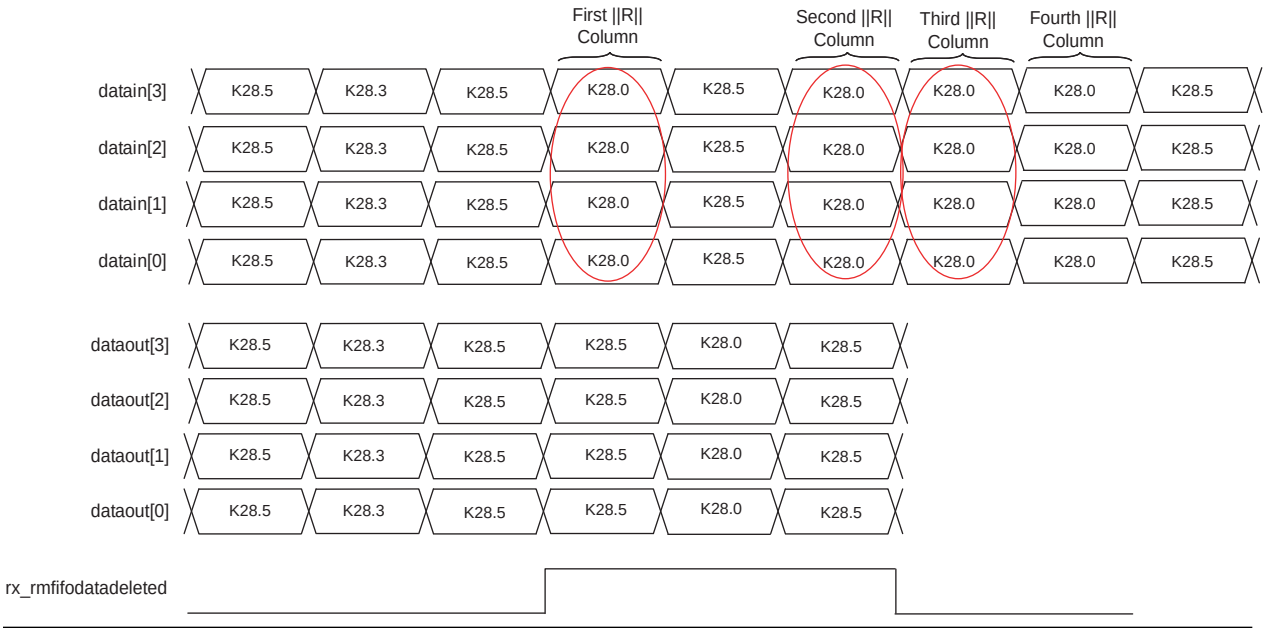
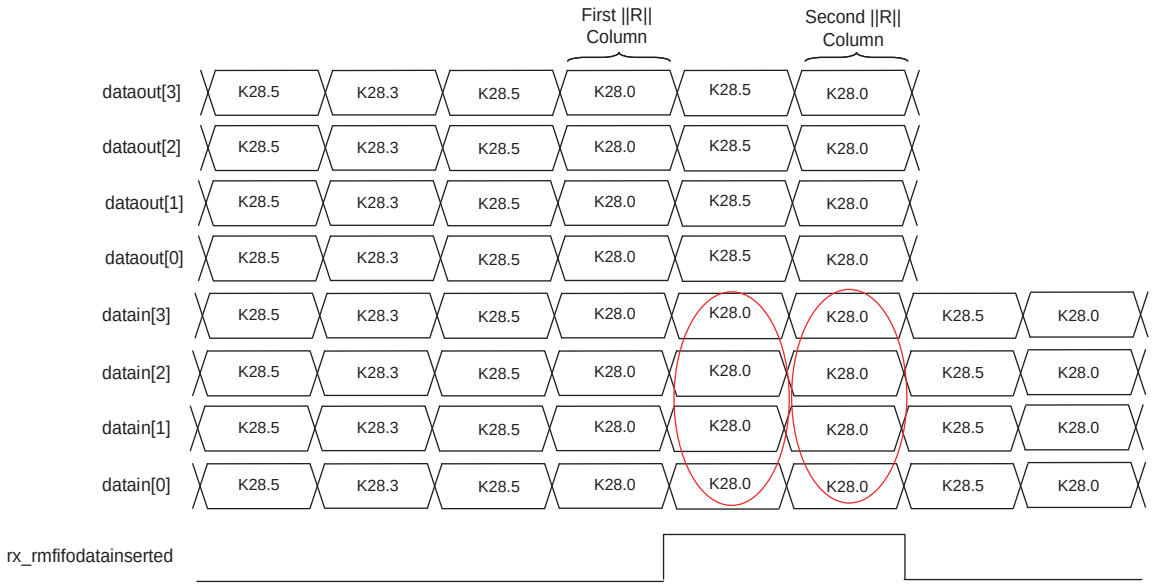


Figure 1–126 shows an example of rate match insertion in the case where two ||R|| columns are required to be inserted.

Figure 1–126. Rate Match Insertion in XAUI Mode



For more information, refer to "Rate Match (Clock Rate Compensation) FIFO" on page 1–74.

GIGE Mode

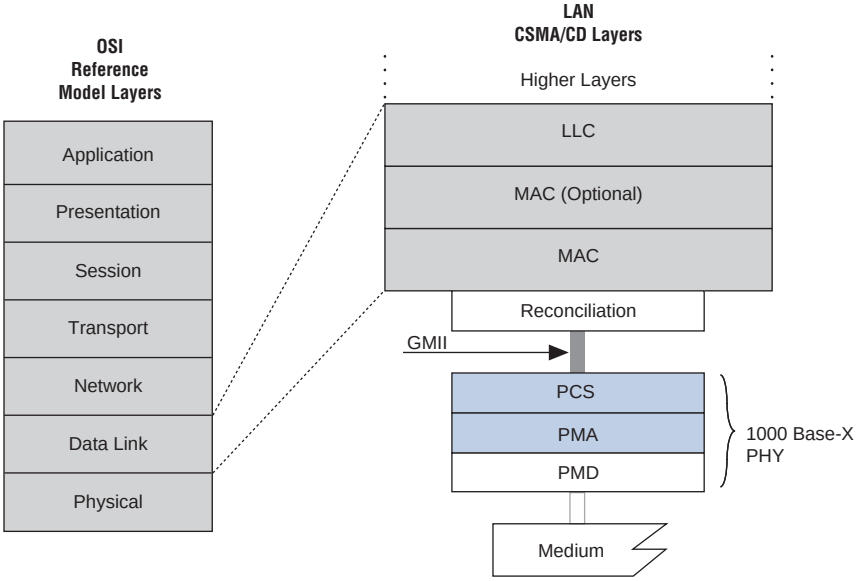
IEEE 802.3 defines the 1000 Base-X PHY as an intermediate, or transition, layer that interfaces various physical media with the media access control (MAC) in a gigabit ethernet system. It shields the MAC layer from the specific nature of the underlying medium. The 1000 Base-X PHY is divided into three sub-layers:

- Physical coding sublayer
- Physical media attachment
- Physical medium dependent (PMD)

The PCS sublayer interfaces with the MAC through the gigabit medium independent interface (GMII). The 1000 Base-X PHY defines a physical interface data rate of 1 Gbps.

Figure 1–127 shows the 1000 Base-X PHY position in a Gigabit Ethernet OSI reference model.

Figure 1–127. 1000 Base-X PHY in a Gigabit Ethernet OSI Reference Model



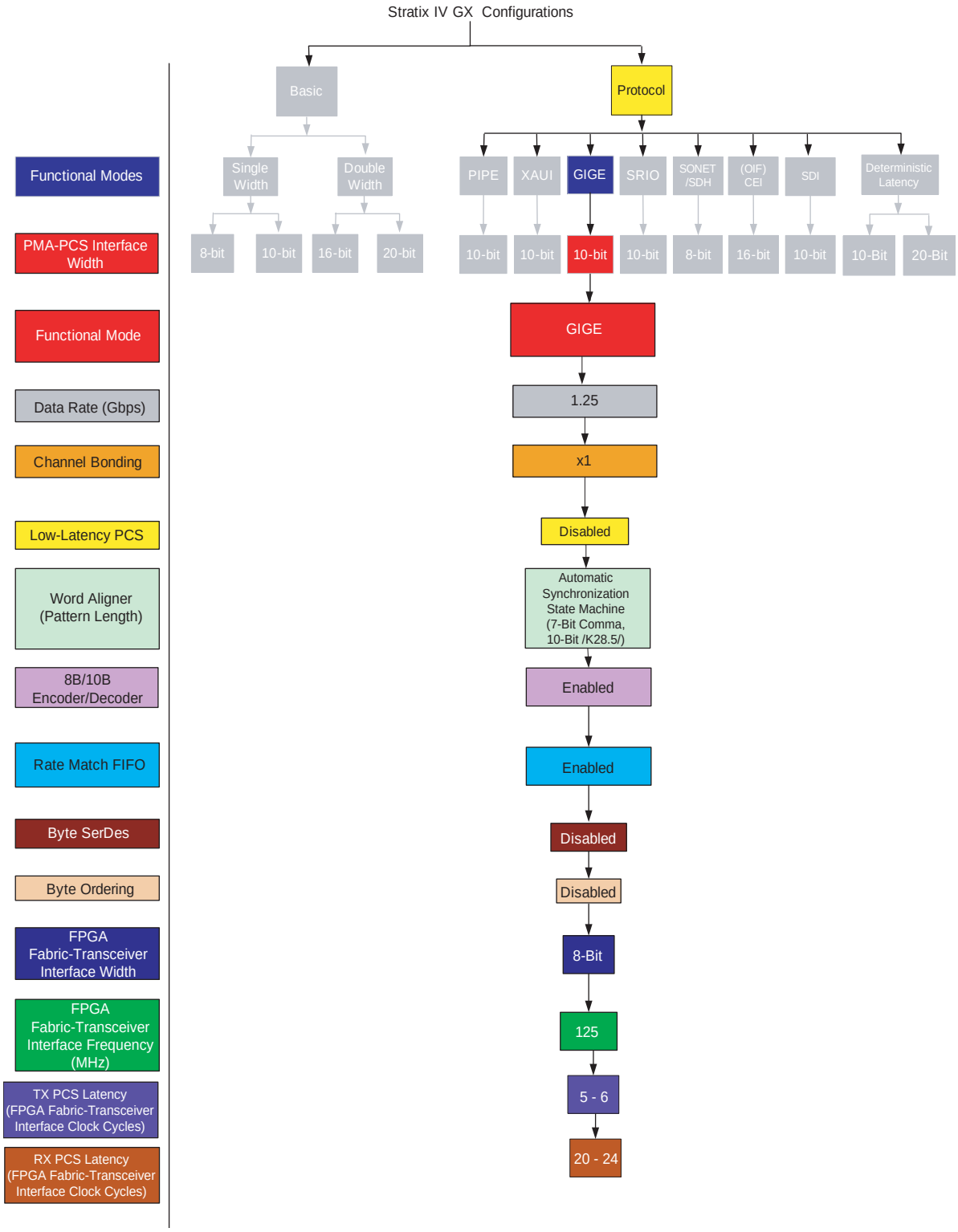
Stratix IV GX and GT transceivers, when configured in GIGE functional mode, have built-in circuitry to support the following PCS and PMA functions defined in the IEEE 802.3 specification:

- 8B/10B encoding and decoding
- Synchronization
- Upstream transmitter and local receiver clock frequency compensation (rate matching)
- Clock recovery from the encoded data forwarded by the receiver PMD
- Serialization and deserialization

Stratix IV GX and GT transceivers do not have built-in support for other PCS functions; for example, auto-negotiation state machine, collision-detect, and carrier-sense. If required, you must implement these functions in a PLD logic array or external circuits.

Figure 1–128 shows the GIGE mode configuration supported in Stratix IV GX devices.

Figure 1–128. GIGE Mode for Stratix IV GX Devices



GIGE Mode Datapath

Figure 1–129 shows the transceiver datapath when configured in GIGE functional mode.

Figure 1–129. GIGE Mode Datapath

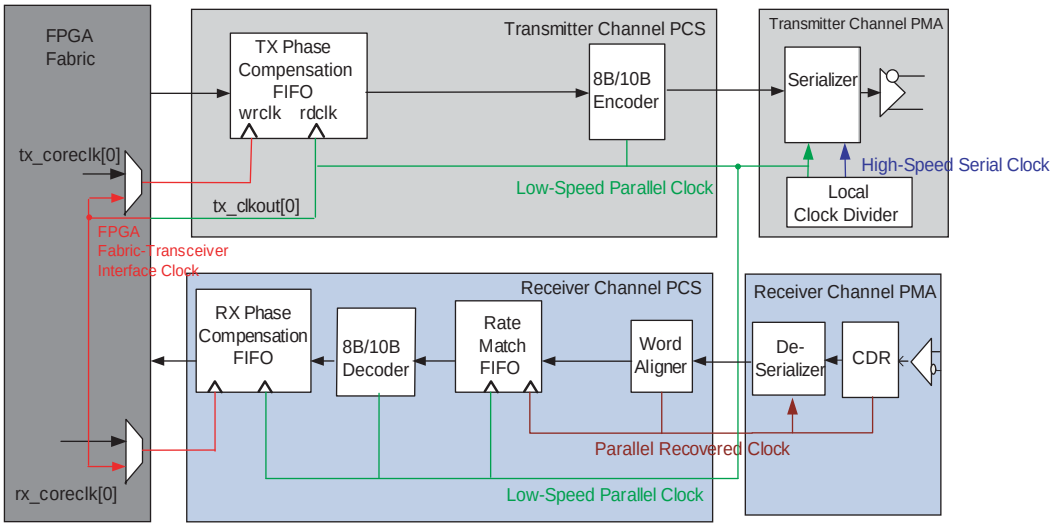


Table 1–62 shows the transceiver datapath clock frequencies in GIGE functional mode.

Table 1–62. Transceiver Datapath Clock Frequencies in GIGE Mode

Functional Mode	Data Rate	High-Speed Serial Clock Frequency	Parallel Recovered Clock and Low-Speed Parallel Clock Frequency	FPGA Fabric-Transceiver Interface Clock Frequency
GIGE	1.25 Gbps	625 MHz	125 MHz	125 MHz

8B/10B Encoder

In GIGE mode, the 8B/10B encoder clocks in 8-bit data and 1-bit control identifiers from the transmitter phase compensation FIFO and generates 10-bit encoded data. The 10-bit encoded data is fed to the serializer. For more information about 8B/10B encoder functionality, refer to "8B/10B Encoder" on page 1–19.

GIGE Protocol—Ordered Sets and Special Code Groups

Table 1–63 lists ordered sets and special code groups specified in the IEEE 802.3 specification.

Table 1–63. GIGE Ordered Sets (Part 1 of 2)

Code	Ordered Set	Number of Code Groups	Encoding
/C/	Configuration	—	Alternating /C1/ and /C2/
/C1/	Configuration 1	4	/K28.5/D21.5/`Config_Reg` *(1)*
/C2/	Configuration 2	4	/K28.5/D2.2/`Config_Reg` *(1)*

Table 1–63. GIGE Ordered Sets (Part 2 of 2)

Code	Ordered Set	Number of Code Groups	Encoding
/I/	IDLE	—	Correcting /I1/, Preserving /I2/
/I1/	IDLE 1	2	/K28.5/D5.6
/I2/	IDLE 2	2	/K28.5/D16.2
	Encapsulation	—	—
/R/	`Carrier_Extend`	1	/K23.7/
/S/	`Start_of_Packet`	1	/K27.7/
/T/	`End_of_Packet`	1	/K29.7/
/V/	`Error_Propagation`	1	/K30.7/

Note to Table 1–63:

(1) Two data code groups representing the `Config_Reg` value.

Idle Ordered-Set Generation

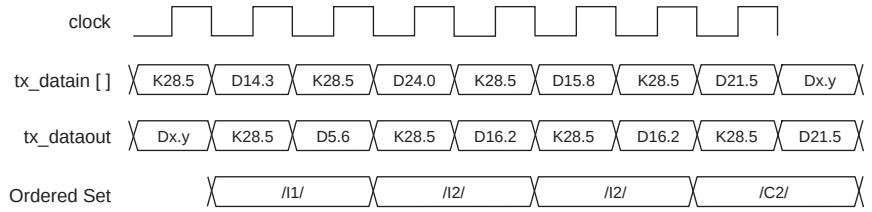
The IEEE 802.3 specification requires the GIGE PHY to transmit idle ordered sets (/I/) continuously and repetitively whenever the GMII is idle. This ensures that the receiver maintains bit and word synchronization whenever there is no active data to be transmitted.

In GIGE functional mode, any /Dx.y/ following a /K28.5/ comma is replaced by the transmitter with either a /D5.6/ (/I1/ ordered set) or a /D16.2/ (/I2/ ordered set), depending on the current running disparity. The exception is when the data following the /K28.5/ is /D21.5/ (/C1/ ordered set) or /D2.2/ (/C2/) ordered set. If the running disparity before the /K28.5/ is positive, an /I1/ ordered set is generated. If the running disparity is negative, a /I2/ ordered set is generated. The disparity at the end of a /I1/ is the opposite of that at the beginning of the /I1/. The disparity at the end of a /I2/ is the same as the beginning running disparity (right before the idle code). This ensures a negative running disparity at the end of an idle ordered set. A /Kx.y/ following a /K28.5/ is not replaced.

Note that /D14.3/, /D24.0/, and /D15.8/ are replaced by /D5.6/ or /D16.2/ (for /I1/, /I2/ ordered sets). /D21.5/ (part of the /C1/ order set) is not replaced.

Figure 1–130 shows the automatic idle ordered set generation.

Figure 1–130. Automatic Ordered Set Generation



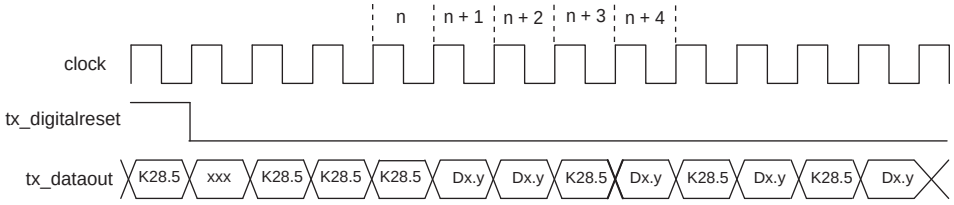
Reset Condition

After de-assertion of `tx_digitalreset`, the GIGE transmitter automatically transmits three /K28.5/ comma code groups before transmitting user data on the `tx_datain` port. This could affect the synchronization state machine behavior at the receiver.

Depending on when you start transmitting the synchronization sequence, there could be an even or odd number of /Dx.y/ code groups transmitted between the last of the three automatically sent /K28.5/ code groups and the first /K28.5/ code group of the synchronization sequence. If there is an even number of /Dx.y/ code groups received between these two /K28.5/ code groups, the first /K28.5/ code group of the synchronization sequence begins at an odd code group boundary (`rx_even` = FALSE). An IEEE802.3-compliant GIGE synchronization state machine treats this as an error condition and goes into the loss of sync state.

Figure 1–131 shows an example of even numbers of /Dx.y/ between the last automatically sent /K28.5/ and the first user-sent /K28.5/. The first user-sent /K28.5/ code group received at an odd code group boundary in cycle n + 3 takes the receiver synchronization state machine in the loss of sync state. The first synchronization ordered set /K28.5/Dx.y/ in cycles n + 3 and n + 4 is discounted and three additional ordered sets are required for successful synchronization.

Figure 1–131. Reset Condition in GIGE Mode



Word Aligner

The word aligner in GIGE functional mode is configured in automatic synchronization state machine mode. The Quartus II software automatically configures the synchronization state machine to indicate synchronization when the receiver receives three consecutive synchronization ordered sets. A synchronization ordered set is a /K28.5/ code group followed by an odd number of valid /Dx.y/ code groups. The fastest way for the receiver to achieve synchronization is to receive three continuous {/K28.5/, /Dx.y/} ordered sets.

Receiver synchronization is indicated on the `rx_syncstatus` port of each channel. A high on the `rx_syncstatus` port indicates that the lane is synchronized; a low on the `rx_syncstatus` port indicates that the lane has fallen out of synchronization. The receiver loses synchronization when it detects four invalid code groups separated by less than three valid code groups or when it is reset.

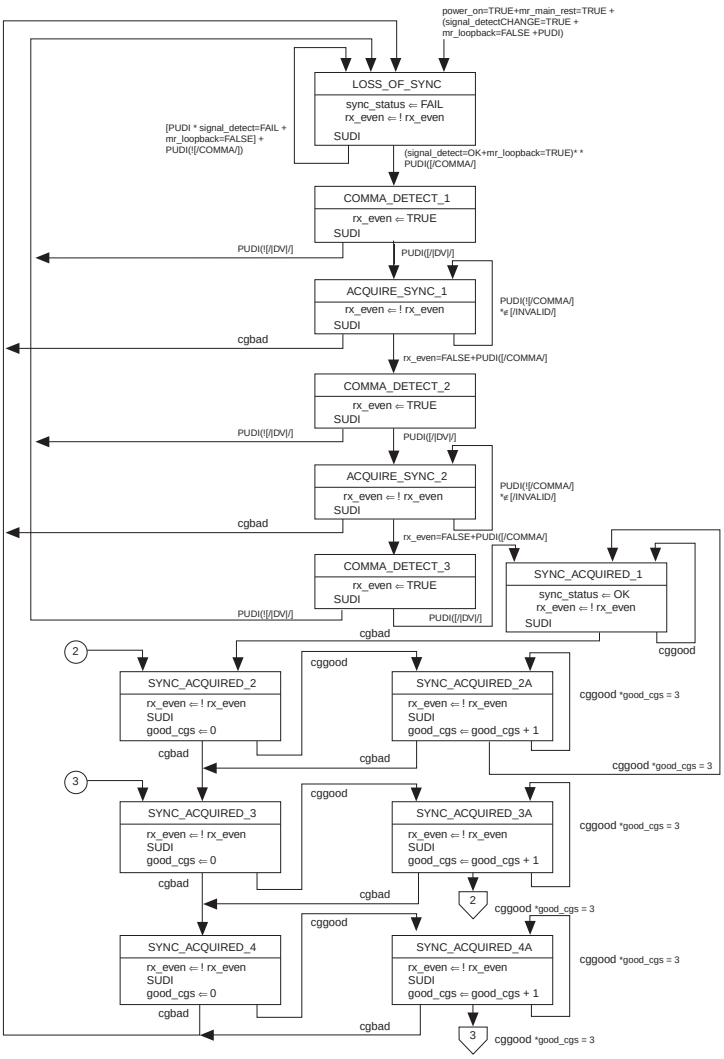
Table 1–64 lists the synchronization state machine parameters when configured in GIGE mode.

Table 1–64. Synchronization State Machine Parameters in GIGE Functional Mode

Synchronization State Machine Parameters	Setting
Number of valid {/K28.5/, /Dx,y/} ordered sets received to achieve synchronization	3
Number of errors received to lose synchronization	4
Number of continuous good code groups received to reduce the error count by 1	4

Figure 1–132 shows the synchronization state machine implemented in GIGE mode.

Figure 1–132. Synchronization State Machine in GIGE Mode *(Note 1)*



Note to Figure 1–132:

(1) This figure is from IEEE P802.3ae.

Rate Match FIFO

In GIGE mode, the rate match FIFO is capable of compensating for up to ±100 PPM (200 PPM total) difference between the upstream transmitter and the local receiver reference clock. The GIGE protocol requires the transmitter to send idle ordered sets /I1/ (/K28.5/D5.6/) and /I2/ (/K28.5/D16.2/) during inter-packet gaps adhering to the rules listed in the IEEE 802.3 specification.

The rate match operation begins after the synchronization state machine in the word aligner indicates synchronization is acquired by driving the `rx_syncstatus` signal high. The rate matcher deletes or inserts both symbols (/K28.5/ and /D16.2/) of the /I2/ ordered sets even if it requires deleting only one symbol to prevent the rate match FIFO from overflowing or under-running. It can insert or delete as many /I2/ ordered sets as necessary to perform the rate match operation.

Two flags, `rx_rmfifodatadeleted` and `rx_rmfifodatainserted`, indicating rate match FIFO deletion and insertion events, respectively, are forwarded to the FPGA fabric. Both the `rx_rmfifodatadeleted` and `rx_rmfifodatainserted` flags are asserted for two clock cycles for each deleted and inserted /I2/ ordered set, respectively.

Figure 1–133 shows an example of rate match FIFO deletion where three symbols are required to be deleted. Because the rate match FIFO can only delete /I2/ ordered set, it deletes two /I2/ ordered sets (four symbols deleted).

Figure 1–133. Rate Match Deletion in GIGE Mode

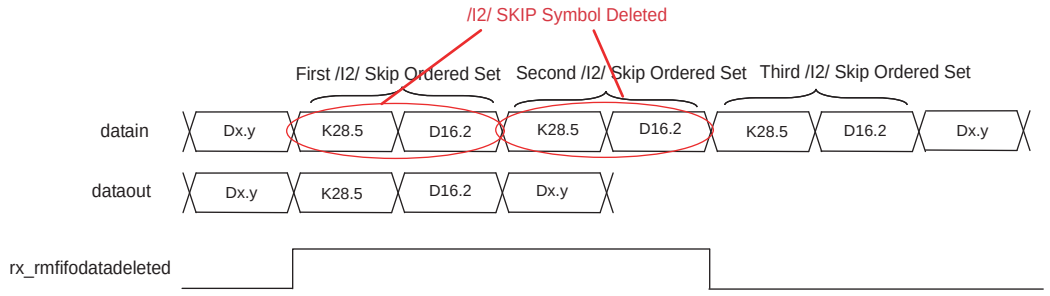
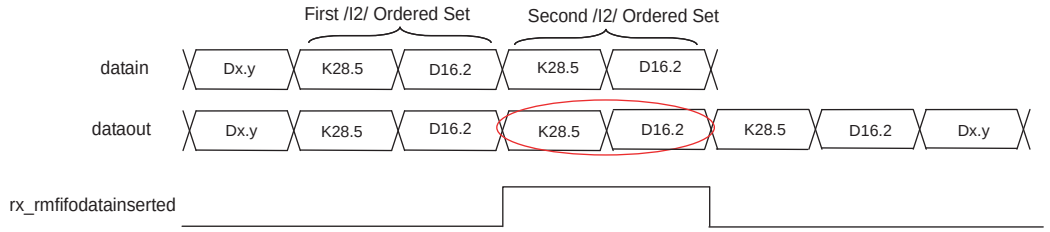


Figure 1–134 shows an example of rate match FIFO insertion in the case where one symbol is required to be inserted. Because the rate match FIFO can only delete /I2/ ordered set, it inserts one /I2/ ordered set (two symbols inserted).

Figure 1–134. Rate Match Insertion in GIGE Mode



For more information, refer to "Rate Match (Clock Rate Compensation) FIFO" on page 1–74.

SONET/SDH Mode

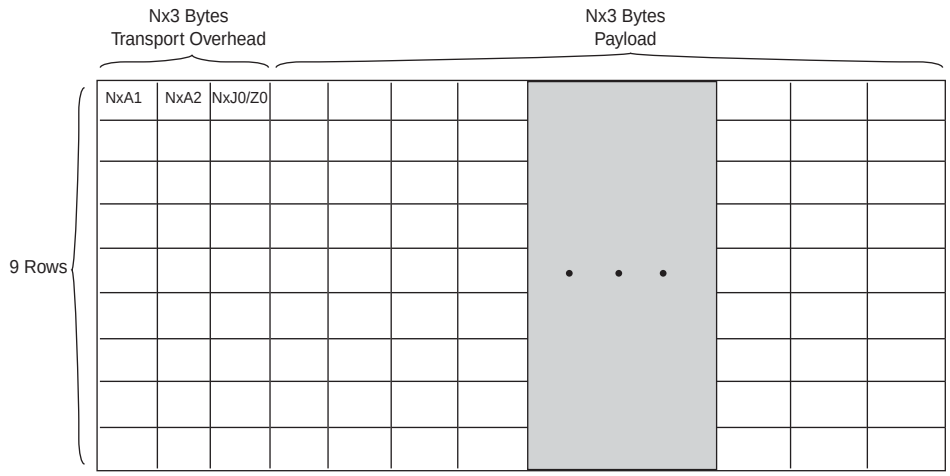
SONET/SDH is one of the most common serial-interconnect protocols used in backplanes deployed in communications and telecom applications. SONET/SDH defines various optical carrier (OC) sub-protocols for carrying signals of different capacities through a synchronous optical hierarchy.

SONET/SDH Frame Structure

Base OC-1 frames are byte-interleaved to form SONET/SDH frames. For example, 12 OC-1 frames are byte-interleaved to form one OC-12 frame; 48 OC-1 frames are byte-interleaved to form one OC-48 frame, and so on. SONET/SDH frame sizes are constant, with a frame transfer rate of 125 µs.

Figure 1–135 shows the SONET/SDH frame structure.

Figure 1–135. SONET/SDH Mode



Transport overhead bytes A1 and A2 are used for restoring frame boundary from the serial data stream. Frame sizes are fixed, so the A1 and A2 bytes appear within the serial data stream every 125 µs. In an OC-12 system, 12 A1 bytes are followed by 12 A2 bytes. Similarly, in an OC-48 system, 48 A1 bytes are followed by 48 A2 bytes.

In SONET/SDH systems, byte values of A1 and A2 are fixed as follows:

- A1 = 11110110 or 8'hF6
- A2 = 00101000 or 8'h28

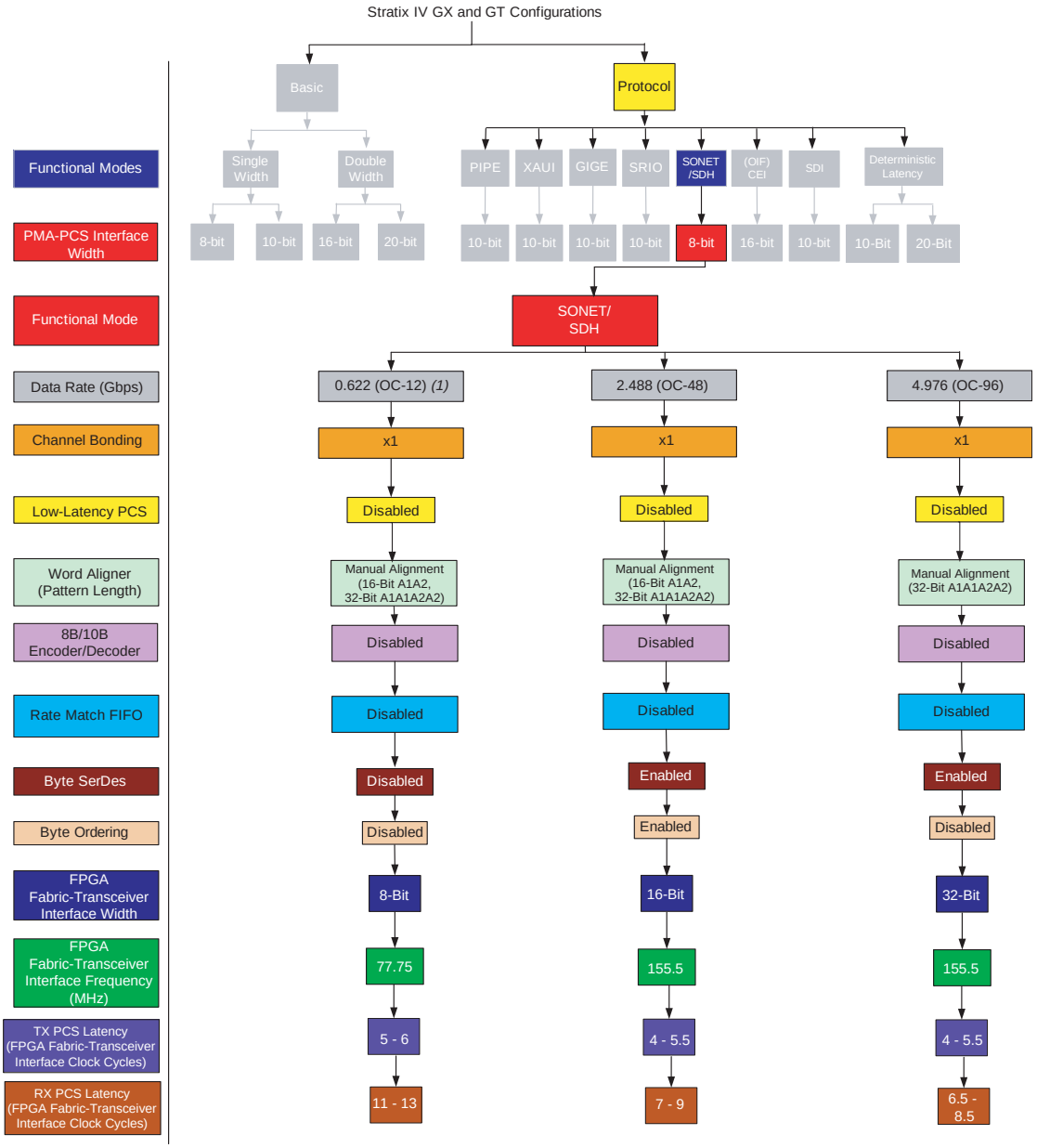
You can employ Stratix IV GX and GT transceivers as physical layer devices in a SONET/SDH system. These transceivers provide support for SONET/SDH protocol-specific functions and electrical features; for example, alignment to A1A2 or A1A1A2A2 pattern.

Stratix IV transceivers are designed to support the following three SONET/SDH sub-protocols:

- OC-12 at 622 Mbps with 8-bit channel width (not supported in Stratix IV GT devices)
- OC-48 at 2488.32 Mbps with 16-bit channel width
- OC-96 at 4976 Mbps with 32-bit channel width

Figure 1–136 shows SONET/SDH mode configurations supported in Stratix IV GX and GT devices.

Figure 1–136. SONET/SDH Mode Configurations in Stratix IV GX and GT Devices



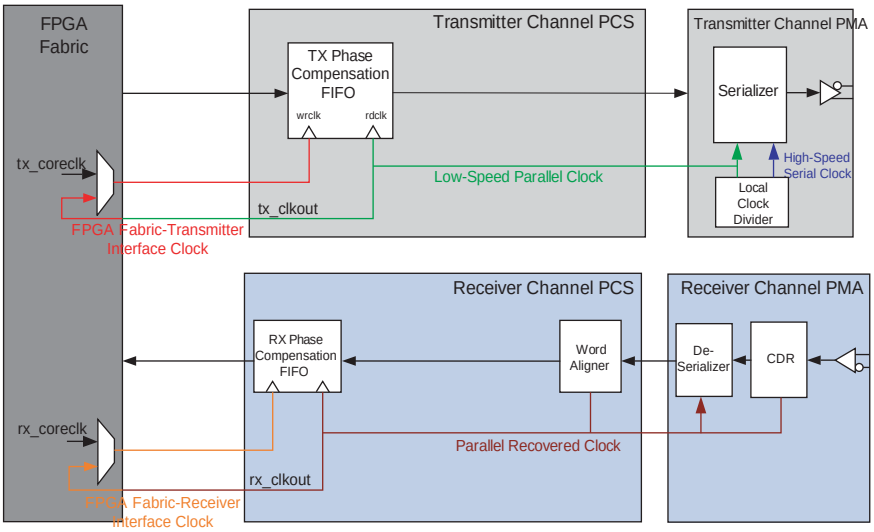
Note to Figure 1–36:

(1) This is not supported in Stratix IV GT devices.

SONET/SDH OC-12 Datapath

Figure 1–137 shows the transceiver datapath when configured in SONET/SDH OC-12 mode.

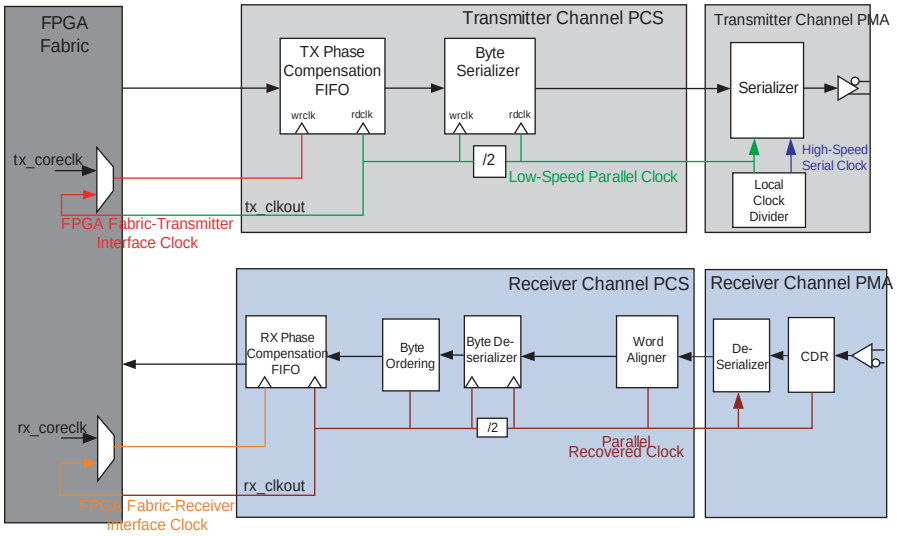
Figure 1–137. SONET/SDH OC-12 Datapath



SONET/SDH OC-48 Datapath

Figure 1–138 shows the transceiver datapath when configured in SONET/SDH OC-48 mode.

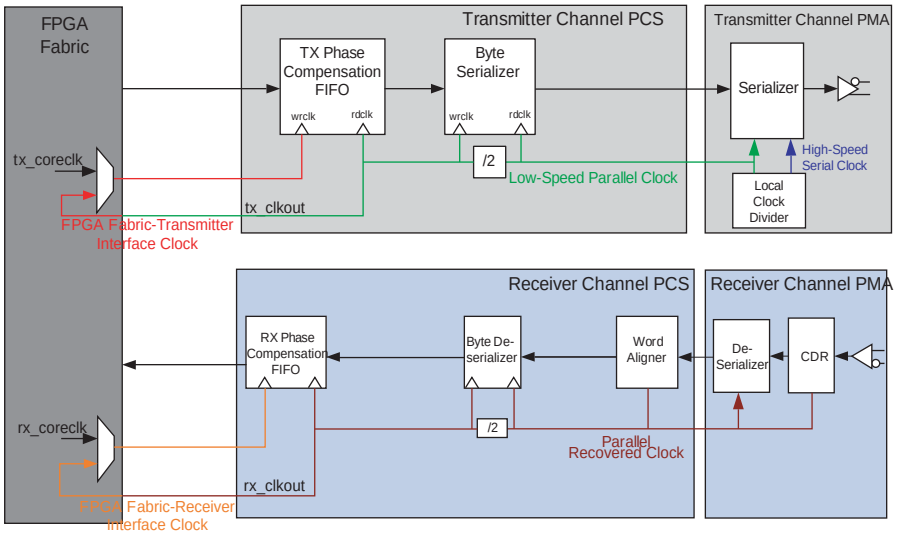
Figure 1–138. SONET/SDH OC-48 Datapath



SONET/SDH OC-96 Datapath

Figure 1–139 shows the transceiver datapath when configured in SONET/SDH OC-96 mode.

Figure 1–139. SONET/SDH OC-96 Datapath



SONET/SDH Transmission Bit Order

Unlike Ethernet, where the LSB of the parallel data byte is transferred first, SONET/SDH requires the MSB to be transferred first and the LSB to be transferred last. To facilitate the MSB-to-LSB transfer, you must enable the following options in the ALTGX MegaWizard Plug-In Manager:

- **Flip transmitter input data bits**
- **Flip receiver output data bits**

Depending on whether data bytes are transferred MSB-to-LSB or LSB-to-MSB, you must select the appropriate word aligner settings in the ALTGX MegaWizard Plug-In Manager. Table 1–65 on page 1–174 lists the correct word aligner settings for each bit transmission order.

Word Alignment

The word aligner in SONET/SDH OC-12, OC-48, and OC-96 modes is configured in manual alignment mode, as described in "Word Aligner in Single-Width Mode with 8-Bit PMA-PCS Interface Modes" on page 1–57.

In OC-12 and OC-48 configurations, you can configure the word aligner to either align to a 16-bit A1A2 pattern or a 32-bit A1A1A2A2 pattern. This is controlled by the `rx_a1a2size` input port to the transceiver. A low level on the `rx_a1a2size` port configures the word aligner to align to a 16-bit A1A2 pattern; a high level on the `rx_a1a2size` port configures the word aligner to align to a 32-bit A1A1A2A2 pattern.

In OC-96 configuration, the word aligner is only allowed to align to a A1A1A2A2 pattern, so the input port `rx_ala2size` is unavailable. Barring this difference, the OC-96 word alignment operation is similar to that of the OC-12 and OC-48 configurations.

You can configure the word aligner to flip the alignment pattern bits programmed in the MegaWizard and compare them with the incoming data for alignment. This feature offers flexibility to the SONET backplane system for either a MSB-to-LSB or LSB-to-MSB data transfer. Table 1–65 lists word alignment patterns that you must program in the ALTGX MegaWizard Plug-In Manager based on the bit-transmission order and the word aligner bit-flip option.

Table 1–65. Word Aligner Settings

Serial Bit Transmission Order	Word Alignment Bit Flip	Word Alignment Pattern
MSB-to-LSB	On	1111011000101000 (16'hF628)
MSB-to-LSB	Off	0001010001101111 (16'h146F)
LSB-to-MSB	Off	0010100011110110 (16'h28F6)

The behavior of the SONET/SDH word aligner control and status signals, along with an operational timing diagram, are explained in "Word Aligner in Single-Width Mode with 8-Bit PMA-PCS Interface Modes" on page 1–57.

OC-48 and OC-96 Byte Serializer and Deserializer

The OC-48 and OC-96 transceiver datapath includes the byte serializer and deserializer to allow the PLD interface to run at a lower speed. The OC-12 configuration does not use the byte serializer and deserializer blocks.

The byte serializer and deserializer blocks are explained in "Byte Serializer" on page 1–16 and "Byte Deserializer" on page 1–89, respectively.

The OC-48 byte serializer converts 16-bit data words from the FPGA fabric and translates the 16-bit data words into two 8-bit data bytes at twice the rate. The OC-48 byte deserializer takes in two consecutive 8-bit data bytes and translates them into a 16-bit data word to the FPGA fabric at half the rate.

The OC-96 byte serializer converts 32-bit data words from the FPGA fabric and translates them into two 16-bit data words at twice the rate. The OC-96 byte deserializer takes in two consecutive 16-bit data words and translates them into a 32-bit data word to the FPGA fabric at half the rate.

OC-48 Byte Ordering

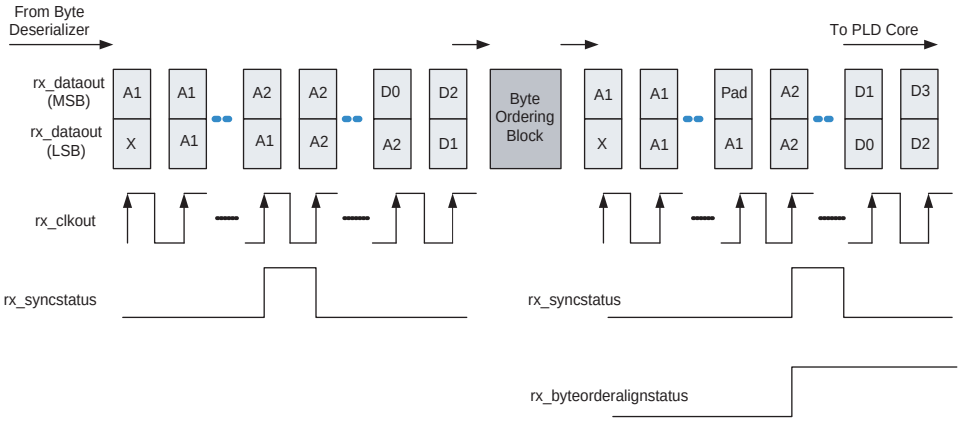
Because of byte deserialization, the MSByte of a word might appear at the `rx_dataout` port along with the LSByte of the next word.

In an OC-48 configuration, the byte ordering block is built into the datapath and can be leveraged to perform byte ordering. Byte ordering in an OC-48 configuration is automatic, as explained in "Word-Alignment-Based Byte Ordering" on page 1–94.

In automatic mode, the byte ordering block is triggered by the rising edge of the `rx_syncstatus` signal. As soon as the byte ordering block sees the rising edge of the `rx_syncstatus` signal, it compares the LSByte coming out of the byte deserializer with the A2 byte of the A1A2 alignment pattern. If the LSByte coming out of the byte deserializer does not match the A2 byte set in the ALTGX MegaWizard Plug-In Manager, the byte ordering block inserts a PAD character, as seen in Figure 1–140. Insertion of this PAD character enables the byte ordering block to restore the correct byte order.

The PAD character is defaulted to the A1 byte of the A1A2 alignment pattern.

Figure 1–140. OC-48 Byte Ordering in Automatic Mode



SDI Mode

The Society of Motion Picture and Television Engineers (SMPTE) defines various SDI standards for transmission of uncompressed video.

The following three SMPTE standards are popular in video broadcasting applications:

- SMPTE 259M standard—more popularly known as the standard-definition (SD) SDI, is defined to carry video data at 270 Mbps
- SMPTE 292M standard—more popularly known as the high-definition (HD) SDI, is defined to carry video data at either 1485 Mbps or 1483.5 Mbps
- SMPTE 424M standard—more popularly known as the third-generation (3G) SDI, is defined to carry video data at either 2970 Mbps or 2967 Mbps

You can configure Stratix IV GX and GT transceivers in HD-SDI or 3G-SDI configuration using the ALTGX MegaWizard Plug-In Manager.

Table 1–66 shows ALTGX configurations supported by Stratix IV transceivers in SDI mode.

Table 1–66. ALTGX Configurations in SDI Mode

Configuration	Data Rate (Mbps)	REFCLK Frequencies (MHz)	FPGA Fabric-Transceiver Interface Width
HD *(2)*	1485	74.25, 148.5	10 bit and 20 bit
	1483.5	74.175, 148.35	10 bit and 20 bit
3G *(1)*	2970	148.5, 297	Only 20-bit interface allowed in 3G
	2967	148.35, 296.7	Only 20-bit interface allowed in 3G

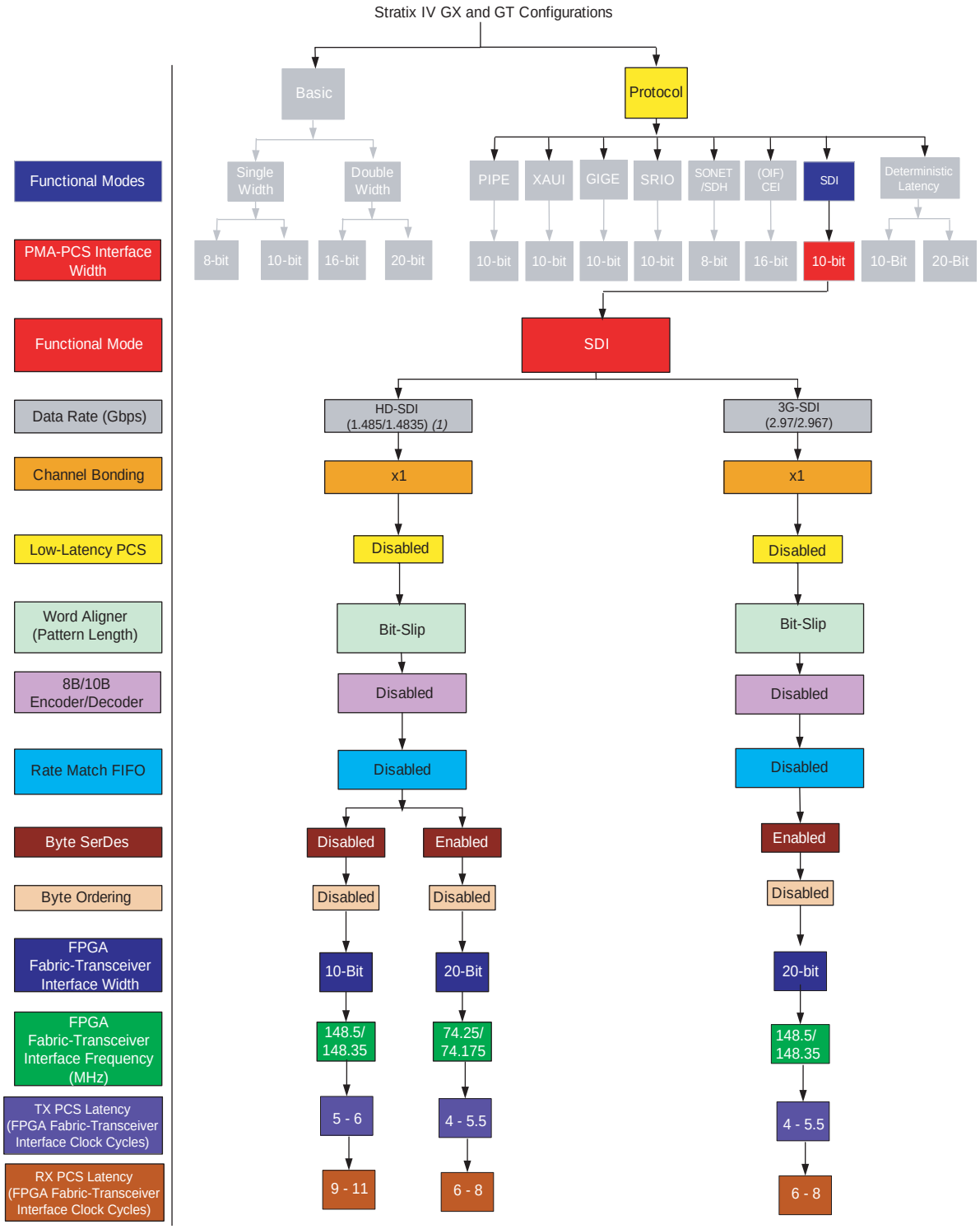
Notes to Table 1–66:

(1) Stratix IV GT devices only support the 3G configuration.

(2) Not supported by Stratix IV GT devices.

Figure 1–141 shows SDI mode configurations supported in Stratix IV GX and GT devices.

Figure 1–141. SDI Mode



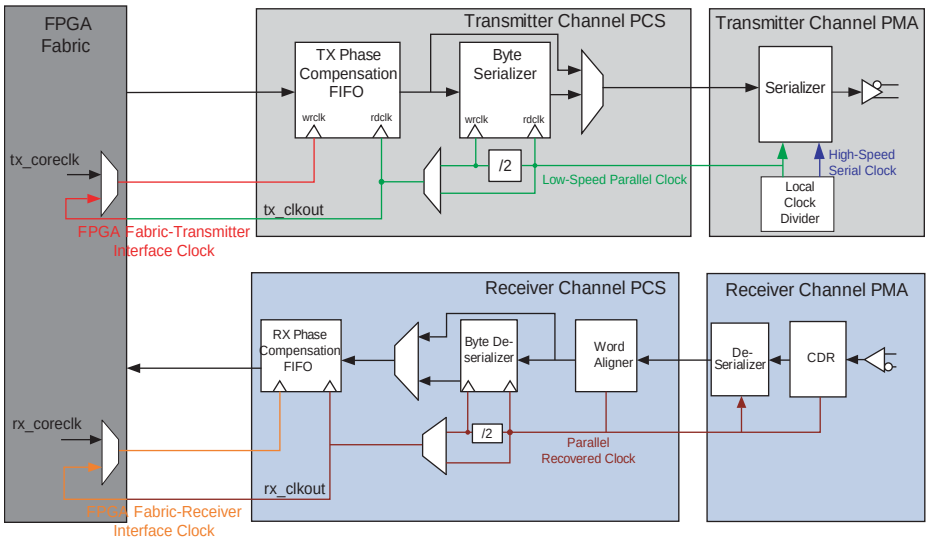
Note to Figure 1–41:

(1) Not supported in Stratix IV GT devices.

SDI Mode Datapath

Figure 1–142 shows the transceiver datapath when configured in SDI mode.

Figure 1–142. SDI Mode Datapath



Transmitter Datapath

The transmitter datapath, in HD-SDI configuration with 10-bit wide FPGA fabric-transceiver interface, consists of the transmitter phase compensation FIFO and the 10:1 serializer. The transmitter datapath, in HD-SDI and 3G-SDI configurations with 20-bit wide FPGA fabric-transceiver interface, also includes the byte serializer.

In SDI mode, the transmitter is purely a parallel-to-serial converter. SDI transmitter functions, such as scrambling and cyclic redundancy check (CRC) code generation, must be implemented in the FPGA logic array.

Receiver Datapath

In the 10-bit channel width SDI configuration, the receiver datapath is comprised of the clock recovery unit (CRU), 1:10 deserializer, word aligner in bit-slip mode, and receiver phase compensation FIFO. In the 20-bit channel width SDI configuration, the receiver datapath also includes the byte deserializer.

SDI receiver functions, such as de-scrambling, framing, and CRC checker, must be implemented in the FPGA logic array.

Receiver Word Alignment and Framing

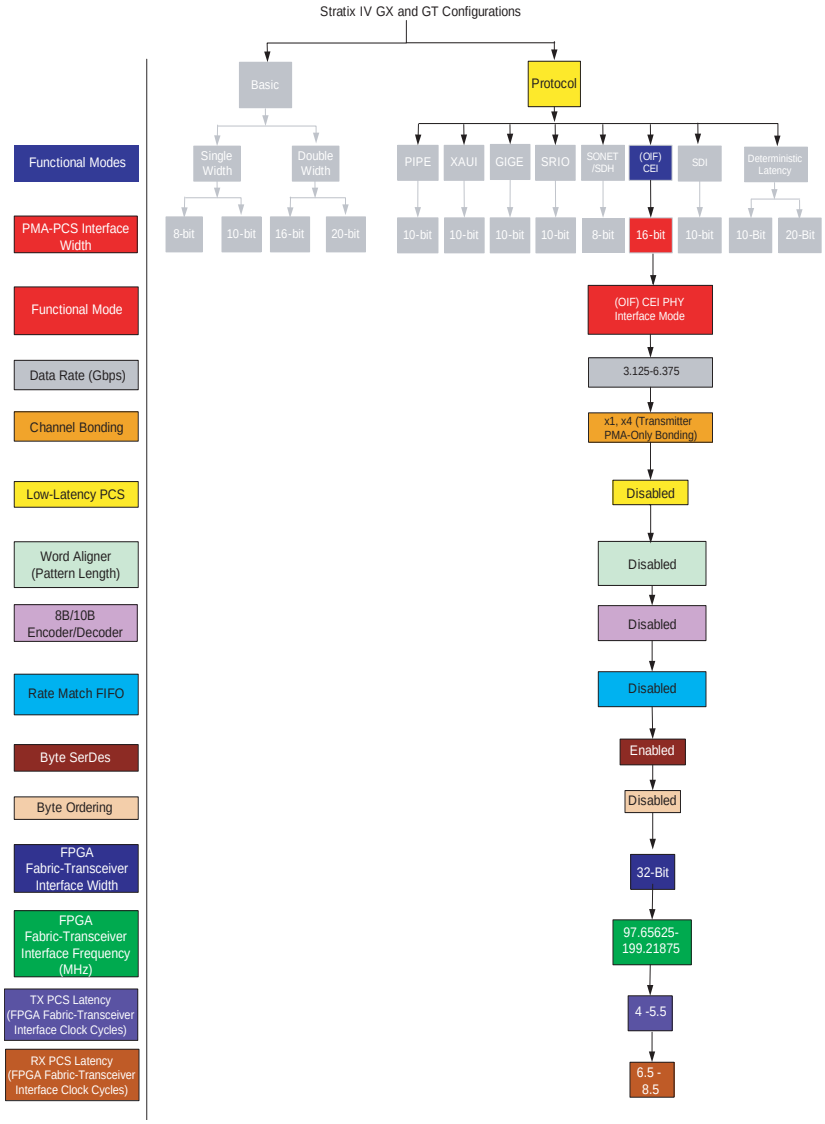
In SDI systems, the word aligner in the receiver datapath is not useful because word alignment and framing happens after de-scrambling. Altera recommends driving the ALTGX megafunction `rx_bitslip` signal low to avoid having the word aligner insert bits in the received data stream.

(OIF) CEI PHY Interface Mode

Stratix IV GX and GT transceivers support a data rate between 4.976 Gbps and 6.375 Gbps in (OIF) CEI PHY interface mode.

Figure 1–143 shows (OIF) CEI PHY interface mode configurations supported in Stratix IV GX and GT devices.

Figure 1–143. (OIF) CEI PHY Interface Mode for Stratix IV GX and GT Devices



(OIF) CEI PHY Interface Mode Datapath

Figure 1–144 shows the ALTGX megafunction transceiver datapath when configured in (OIF) CEI PHY interface mode.

Figure 1–144. (OIF) CEI PHY Interface Mode Datapath

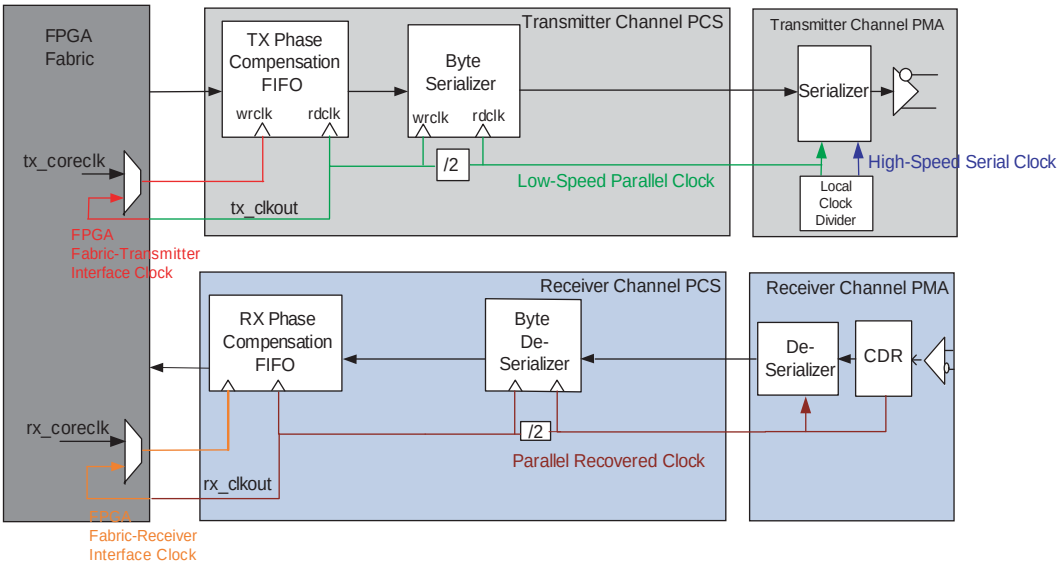
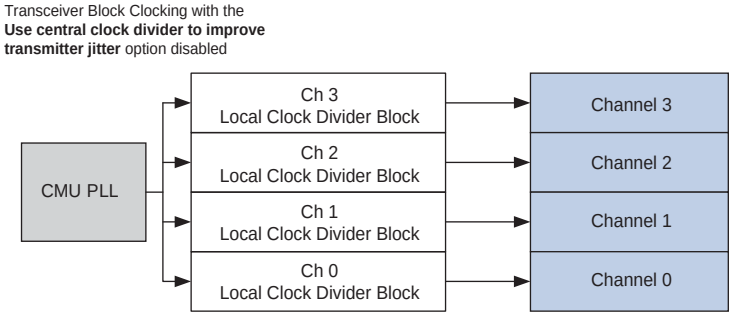


Figure 1–145 shows transceiver clocking in (OIF) CEI PHY interface mode.

Figure 1–145. Transceiver Clocking in (OIF) CEI PHY Interface Mode



Serial RapidIO Mode

The RapidIO Trade Association defines a high-performance, packet-switched interconnect standard to pass data and control information between microprocessors, digital signal, communications, and network processors, system memories, and peripheral devices.

Serial RapidIO physical layer specification defines three line rates:

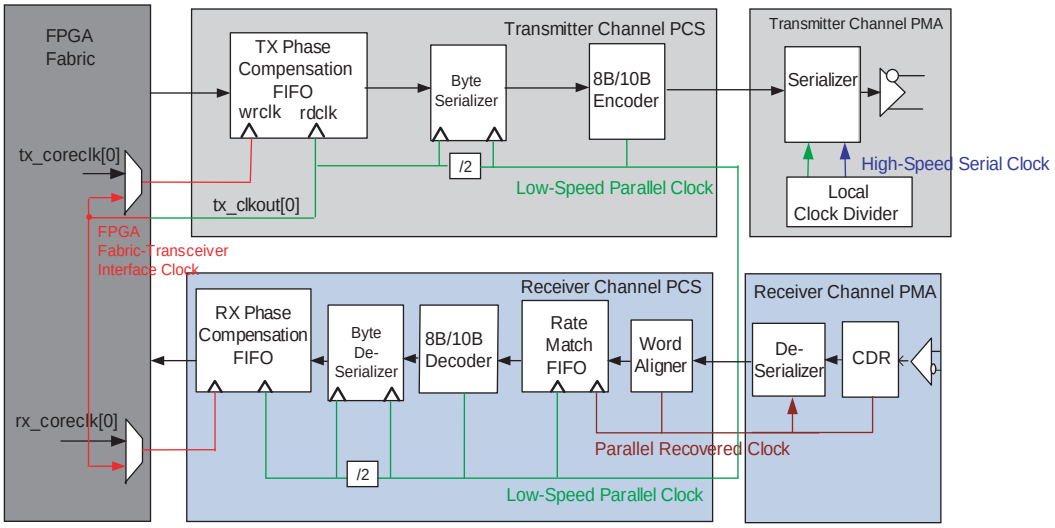
- 1.25 Gbps
- 2.5 Gbps
- 3.125 Gbps

It also defines two link widths—single-lane (1×) and bonded four-lane (4×) at each line rate.

Stratix IV GX and GT transceivers support only single-lane (1×) configuration at all three line rates. Four 1× channels configured in Serial RapidIO mode can be instantiated to achieve a 4× Serial RapidIO link. The four transmitter channels in this 4× Serial RapidIO link are not bonded. The four receiver channels in this 4× Serial RapidIO link do not have lane alignment or deskew capability.

Figure 1–146 shows the ALTGX transceiver datapath when configured in Serial RapidIO mode.

Figure 1–146. Serial RapidIO Mode Datapath



Stratix IV GX and GT transceivers, when configured in Serial RapidIO functional mode, provide the following PCS and PMA functions:

- 8B/10B encoding/decoding
- Word alignment
- Lane synchronization state machine
- Clock recovery from the encoded data
- Serialization/deserialization

Stratix IV GX and GT transceivers do not have built-in support for other PCS functions; for example, pseudo-random idle sequence generation and lane alignment in 4× mode. Depending on your system requirements, you must implement these functions in the logic array or external circuits.

Synchronization State Machine

In Serial RapidIO mode, the ALTGX MegaWizard Plug-In Manager defaults the word alignment pattern to K28.5. The word aligner has a synchronization state machine that handles the receiver lane synchronization.

The ALTGX MegaWizard Plug-In Manager automatically defaults the synchronization state machine to indicate synchronization when the receiver receives 127 K28.5 (10'b0101111100 or 10'b1010000011) synchronization code groups without receiving an intermediate invalid code group. After synchronization, the state machine indicates loss of synchronization when it detects three invalid code groups separated by less than 255 valid code groups or when it is reset.

Receiver synchronization is indicated on the `rx_syncstatus` port of each channel. A high on the `rx_syncstatus` port indicates that the lane is synchronized and a low indicates that it has fallen out of synchronization.

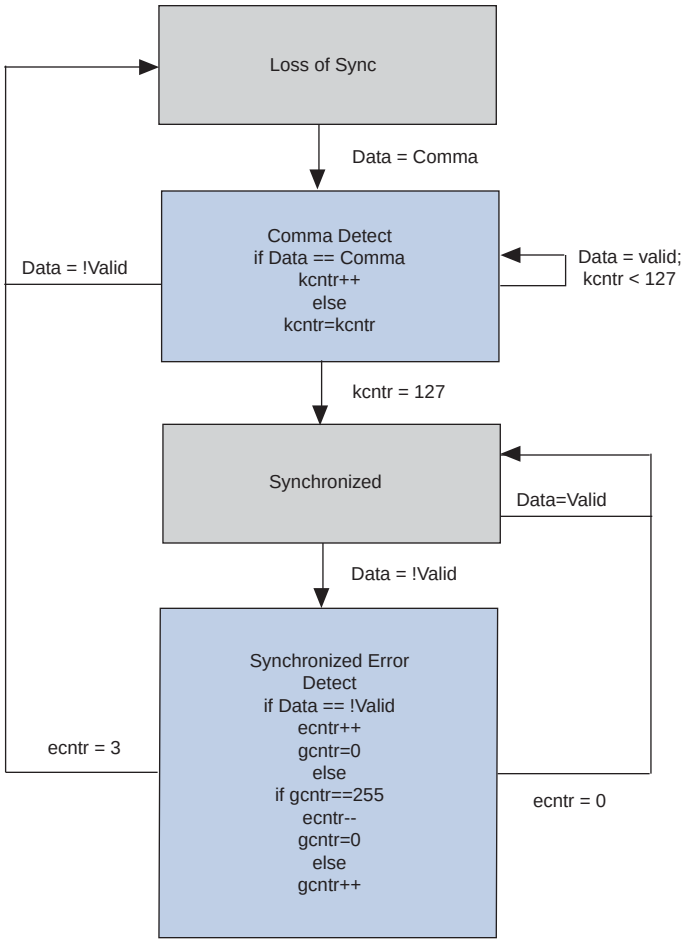
Table 1–67 lists the ALTGX megafunction synchronization state machine parameters when configured in Serial RapidIO mode.

Table 1–67. Synchronization State Machine Parameters in Serial RapidIO Mode

Parameters	Number
Number of valid K28.5 code groups received to achieve synchronization.	127
Number of errors received to lose synchronization.	3
Number of continuous good code groups received to reduce the error count by one.	255

Figure 1–147 shows a conceptual view of the synchronization state machine implemented in Serial RapidIO functional mode.

Figure 1–147. Synchronization State Machine in Serial RapidIO Mode



Rate Match FIFO in Serial RapidIO Mode

In Serial RapidIO mode, the rate match FIFO is capable of compensating for up to ±100 PPM (200 PPM total) difference between the upstream transmitter and the local receiver reference clock.

To enable the rate match FIFO in Serial RapidIO mode, the transceiver channel must have both the transmitter and receiver channel instantiated. You must select the **Receiver and Transmitter** option in the **What is the operation mode?** field in the ALTGX MegaWizard Plug-In Manager. The 8B/10B encoder/decoder is always enabled in Serial RapidIO mode.

Rate matcher is an optional block available for selection in SRIO functional mode. However, this block is not fully compliant to the SRIO specification.

Depending on your implementation, you can select two 20-bit rate match patterns in the ALTGX MegaWizard Plug-In Manager under the **What is the rate match pattern1** and **What is the rate match pattern2** fields. Each of the two programmed 20-bit rate match patterns consists of a 10-bit skip pattern and a 10-bit control pattern.

For Serial RapidIO mode in the ALTGX MegaWizard Plug-In Manager, the control pattern1 defaults to K28.5 with positive disparity and the skip pattern1 defaults to K29.7 with positive disparity. The control pattern2 defaults to K28.5 with negative disparity and the skip pattern2 defaults to K29.7 with negative disparity.

The rate match FIFO operation begins after the word aligner synchronization status `rx_syncstatus` goes high. When the rate matcher receives either of the two 10-bit control patterns followed by the respective 10-bit skip pattern, it inserts or deletes the 10-bit skip pattern as necessary to avoid the rate match FIFO from overflowing or under-running.

In Serial RapidIO mode, the rate match FIFO can delete/insert a maximum of one skip pattern from a cluster.

Two flags, `rx_rmfifodatadeleted` and `rx_rmfifodatainserted`, indicate that rate match FIFO deletion and insertion events, respectively, are forwarded to the FPGA fabric.

Figure 1–148 shows an example of rate match FIFO deletion in the case where one skip pattern is required to be deleted. In this example, the first skip cluster has a /K28.5/ control pattern followed by two /K29.7/ skip patterns. The second skip cluster has a /K28.5/ control pattern followed by four /K29.7/ skip patterns. The rate match FIFO deletes only one /K29.7/ skip pattern from the first skip cluster. One /K29.7/ skip pattern is deleted from the second cluster.

Figure 1–148. Rate Match FIFO Deletion with One Skip Pattern Deleted

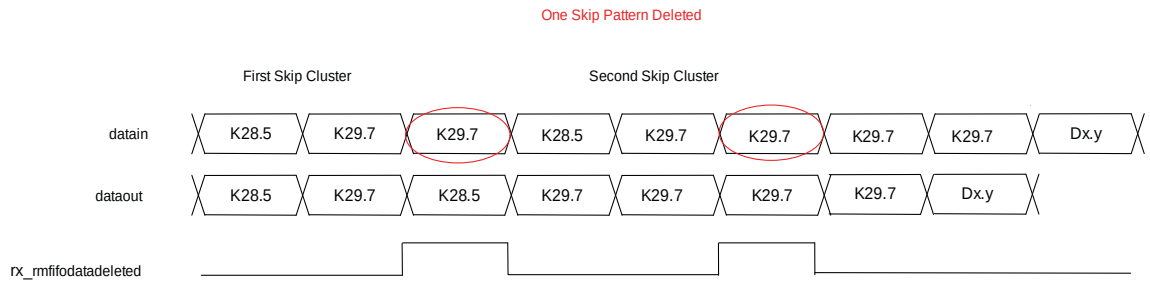
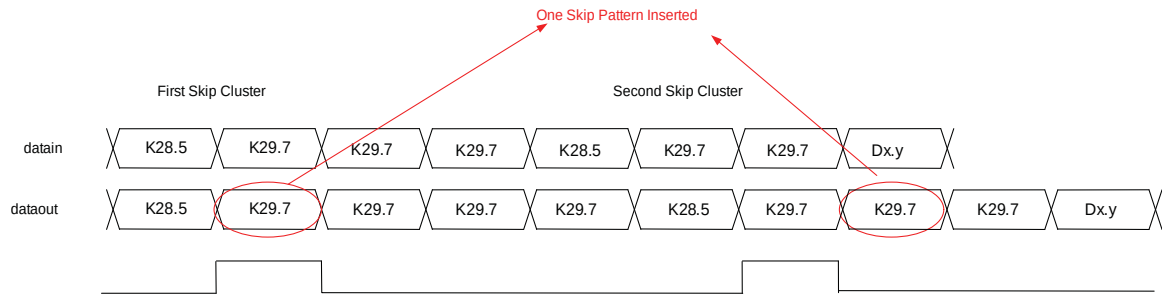


Figure 1–149 shows an example of rate match FIFO insertion in the case where one skip pattern is required to be inserted. In this example, the first skip cluster has a /K28.5/ control pattern followed by three /K29.7/ skip patterns. The second skip cluster has a /K28.5/ control pattern followed by two /K29.7/ skip patterns. The rate match FIFO inserts only one /K29.7/ skip pattern into the first skip cluster. One /K29.7/ skip pattern is inserted into the second cluster.

Figure 1–149. Rate Match FIFO Deletion with One Skip Pattern Inserted



Two flags, `rx_rmfifofull` and `rx_rmfifoempty`, are forwarded to the FPGA fabric to indicate rate match FIFO full and empty conditions. For more information on the behavior of these two signals, refer to "Rate Match FIFO in Basic Single-Width Mode" on page 1–81.

Basic (PMA Direct) Functional Mode

In Basic (PMA Direct) functional mode, the Stratix IV GX and GT transceiver datapath contains only PMA blocks. Parallel data is transferred directly between the FPGA fabric and the serializer/deserializer inside the transmitter/receiver PMA. Because all PCS blocks are bypassed in Basic (PMA Direct) mode, you must implement the required PCS logic in the FPGA fabric.

You can configure four regular transceiver channels inside each transceiver block in Basic (PMA Direct) functional mode. You can configure two CMU channels inside each transceiver block only in Basic (PMA Direct) functional mode, as they do not support PCS circuitry.

In PMA Direct mode, you must create your own logic to support PCS functionality. There are specific reset sequences to be followed in this mode.

Use dynamic reconfiguration to dynamically reconfigure the various PMA controls to tailor the transceivers in PMA direct drive mode for a particular application.

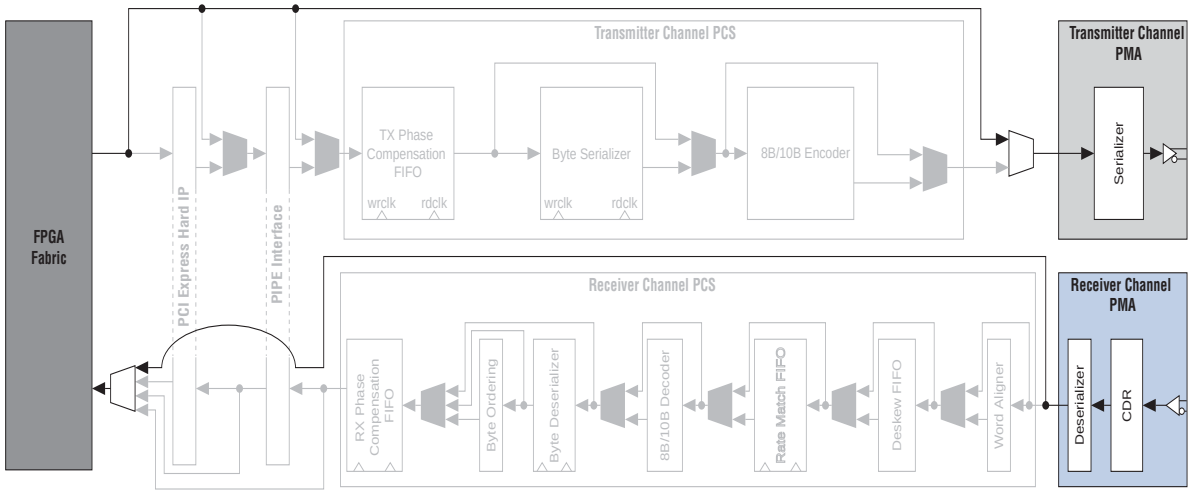
For more information, refer to the *Stratix IV Dynamic Reconfiguration* chapter. For more information about the reset sequence to follow in PMA-Direct mode, refer to the *Stratix IV Reset Control and Power Down* chapter.

The term 'PMA-Direct' is used to describe various configurations in this mode.

In Basic (PMA Direct) mode, all the PCS blocks are bypassed; therefore, any PCS-type features (for example, phase compensation FIFOs, byte serializer, 8B/10B encoder/decoder, word aligner, deskew FIFO, rate match FIFO, byte deserializer, and byte ordering), must be implemented in the FPGA fabric. In Basic (PMA Direct) mode, you must create your own logic to support PCS functionality.

Figure 1–150 shows the Stratix IV GX and GT transceiver configured in Basic (PMA Direct) functional mode. The grayed out blocks indicate areas that are not active in this mode.

Figure 1–150. Stratix IV GX and GT Transceiver Configured in Basic (PMA Direct) Mode



Note to Figure 1–150:

(1) The grayed out blocks shown in Figure 1–150 are not available in the CMU channels. Therefore, the CMU channels can be configured to operate as transceiver channels in PMA direct mode only.

The grayed out blocks shown in Figure 1–150 are not available in the CMU channels. Therefore, the CMU channels can be configured to operate as transceiver channels in PMA Direct mode only.

In Basic (PMA Direct) Mode, you can configure the transceiver channel in two main configurations:

- Basic (PMA Direct) ×1 configuration
- Basic (PMA Direct) ×N configuration

You can configure the transceiver in Basic (PMA Direct) ×1/ ×N mode by setting the appropriate sub-protocol in the **Which sub protocol will you be using**? field. You can select single-width or double-width by selecting **Single/Double** in the **What is the deserializer block width?** field in the ALTGX MegaWizard Plug-In Manager.

In single-width mode, the PMA-PLD interface is 8 bit/10 bit wide; whereas in double-width mode, the PMA-PLD interface is 16 bit/20 bit wide.

Table 1–68 shows the Stratix IV GX and GT PLD-PMA interface widths and data rates supported in Basic (PMA Direct) ×1/×N single-width and double-width modes.

Table 1–68. FPGA Fabric-PMA Interface Widths and Data Rates Supported in Basic (PMA Direct) ×1/×N Single-Width and Double-Width Modes for Stratix IV GX and GT Devices

Basic (PMA Direct) Functional Mode	FPGA Fabric-PMA Interface Width	Supported Data Rate Range			
		Stratix IV GX			Stratix IV GT
		C2 Speed Grade	C3/I3 Speed Grade	C4 Speed Grade	I1, I2, I3
×1/×N Single-width mode	8 bit	0.6Gbps to 2.6Gbps	0.6Gbps to 2.6Gbps	0.6Gbps to 2.6Gbps	2.488 Gbps to 2.6 Gbps
	10 bit	0.6Gbps to 3.25Gbps	0.6Gbps to 3.25Gbps	0.6Gbps to 3.25Gbps	2.488 Gbps to 3.25 Gbps
×1/×N Double-width mode	16 bit	1.0Gbps to 5.2Gbps	1.0Gbps to 5.2Gbps	1.0Gbps to 5.0Gbps	2.488 Gbps to 5.2 Gbps
	20 bit	1.0Gbps to 6.5Gbps	1.0Gbps to 6.5Gbps	1.0Gbps to 5.0Gbps	2.488 Gbps to 6.5 Gbps

Basic (PMA Direct) ×1 Configuration

You can configure a transceiver channel in this mode by setting the **which protocol will you be using?** field to **Basic (PMA Direct)** and the **which sub protocol will you be using?** field to **none**. In this configuration, the Quartus II software requires one of the two CMU PLLs within the same transceiver block to provide high-speed clocks to the transmitter side of the channel.

If the `CMU0` or `CMU1` channel is configured in Basic (PMA Direct) ×1 configuration, use their local clock dividers to provide clock to their respective transmitter channels.

For information about clocking restrictions in Basic (PMA Direct) ×1 mode, refer to the "Non-Bonded Basic (PMA Direct) Mode Channel Configurations" section in the *Stratix IV Transceiver Clocking* chapter.

For information about routing the clocks to transceiver channels in Basic (PMA Direct) ×1 mode, refer to the *Stratix IV Transceiver Clocking* chapter.

Basic (PMA Direct) ×N Configuration

You can configure a transceiver channel in this mode by setting the **which protocol will you be using** field to **Basic (PMA Direct)** and the **which sub protocol will you be using** field to **×N**. In this mode, all the transmitter channels can receive their high-speed clock from the `CMU0` PLL from the transceiver blocks or the ATX PLL present on the same side of the device. These clocks are provided through the `×N_Top` or `×N_Bottom` clock line.

In this mode, if you use a CMU PLL to generate the transceiver channel datapath clocks, only the `CMU0` central clock divider of the transceiver block containing the CMU PLL is used.

For information about clocking restrictions in Basic (PMA Direct) ×N mode, refer to the "Non-Bonded Basic (PMA Direct) Mode Channel Configurations" section in the *Stratix IV Transceiver Clocking* chapter.

For more information about combining multiple transceiver channels, refer to the *Configuring Multiple Protocols and Data Rates in a Transceiver Block* chapter.

Each receiver in a receiver channel has a dedicated CDR that provides a high-speed clock.

For more information about timing closure in Basic (PMA Direct) mode, refer to *AN 580: Achieving Timing Closure in Basic (PMA Direct) Functional Mode*.

Loopback Modes

Stratix IV GX and GT devices provide various loopback options that allow you to verify how different functional blocks work in the transceiver channel. The available loopback options are:

- "Serial Loopback" on page 1–188—available in all functional modes except PCI Express (PIPE) mode
- "Parallel Loopback" on page 1–189—available in either single-width or double-width modes.
- "Reverse Serial Loopback" on page 1–191—available in Basic mode only
- "Reverse Serial Pre-CDR Loopback" on page 1–191—available in Basic mode only
- "PCI Express (PIPE) Reverse Parallel Loopback" on page 1–192—supported in PCI Express (PIPE) protocol only (this loopback mode does not support Stratix IV GT devices)

Serial Loopback

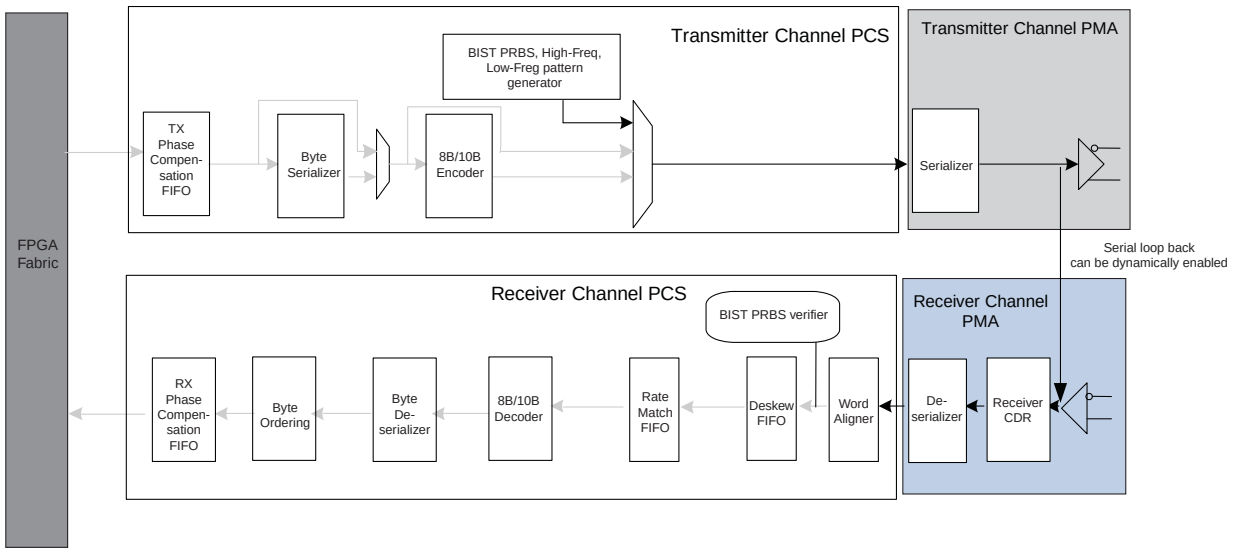
The **serial loopback** option is available for all functional modes except PCI Express (PIPE) mode. Figure 1–151 shows the datapath for serial loopback. The data from the FPGA fabric passes through the transmitter channel and gets looped back to the receiver channel, bypassing the receiver buffer. The received data is available to the FPGA logic for verification. Using this option, you can check the working for all enabled PCS and PMA functional blocks in the transmitter and receiver channel. When you enable the **serial loopback** option, the ALTGX MegaWizard Plug-In Manager provides the `rx_seriallpbken` port to dynamically enable serial loopback on a channel-by-channel basis. Set the `rx_seriallpbken` signal to logic high to enable serial loopback.

When serial loopback is enabled, the transmitter channel sends the data to both the `tx_dataout` output port and to the receiver channel. The differential output voltage on the `tx_dataout` ports is based on the selected V_{OD} settings. The looped back data is received by the receiver CDR and is retimed through different clock domains. You must provide an alignment pattern for the word aligner to enable the receiver channel to retrieve the byte boundary.

Suppose the device is not in serial loopback mode and is receiving data from a remote device. At this point, the receiver CDR's recovered clock is locked to the data from that source. If the device is placed in serial loopback mode, the data source to the receiver changes from the remote device to local transmitter channel. This prompts the receiver CDR to start tracking the phase of the new data source. During this time, the receiver CDR's recovered clock may be unstable. As the receiver PCS is running off of this recovered clock, you must place the receiver PCS under reset by asserting the `rx_digitalreset` signal during this time period.

When moving into or out of serial loopback, you must assert `rx_digitalreset` for a minimum of two parallel clock cycles.

Figure 1–151. Serial Loopback Datapath



Parallel Loopback

You can configure a transceiver channel in this mode by setting the **which protocol will you be using**? field to **Basic** and the **which sub protocol will you be using?** field to **BIST**. You can only configure a **Receiver and Transmitter** transceiver channel in this functional mode. You can configure a transceiver channel in this mode in either a single-width or double-width configuration.

The BIST pattern generator and pattern verifier are located near the FPGA fabric in the PCS block of the transceiver channel. This placement allows for testing the complete transmitter PCS and receiver PCS datapaths for bit errors. This mode is primarily used for transceiver channel debugging, if needed.

The parallel loopback mode is available only with a built-in 16-bit incremental pattern generator and verifier. The channel width is fixed to 16 bits in this mode. Also in this mode, the incremental pattern 00-FF is looped back to the receiver channel at the PCS functional block boundary before the PMA and is sent to the `tx_dataout` port. The received data is verified by the verifier. This loopback allows you to verify the complete PCS block. The differential output voltage of the transmitted serial data on the `tx_dataout` port is based on the selected V_{OD} settings. The datapath for parallel loopback is shown in Figure 1–152. The incremental data pattern is not available to the FPGA logic for verification.

Figure 1–152. Enabled PCS Functional Blocks in Parallel Loopback

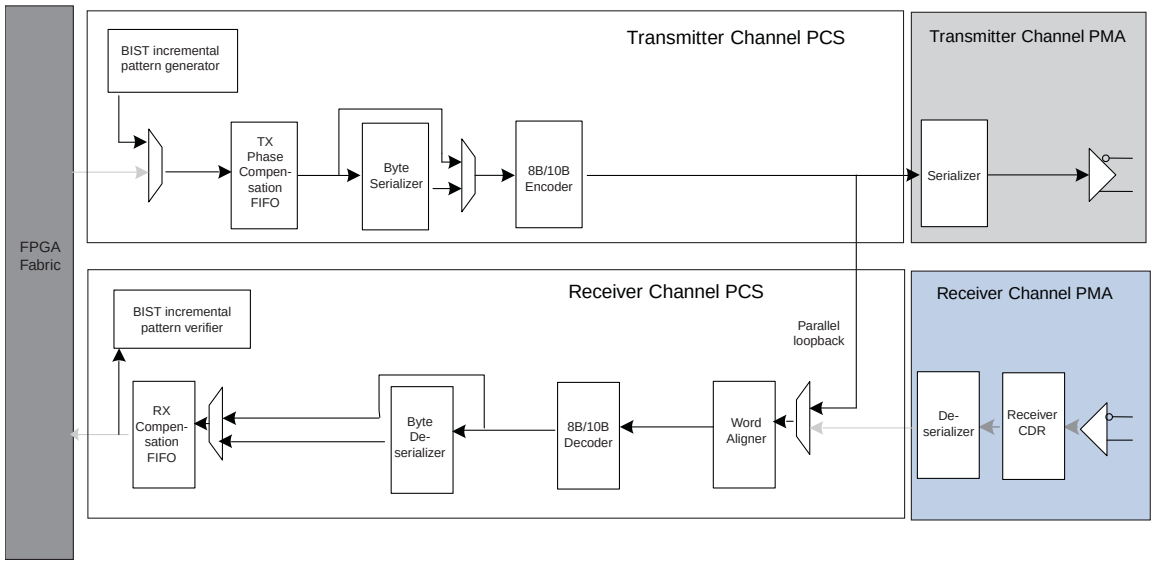


Table 1–69 shows the enabled PCS functional blocks for single-width and double-width mode. The last column in Table 1–69 shows the supported channel width setting for parallel loopback.

Table 1–69. Enabled PCS Functional Blocks for Parallel Loopback

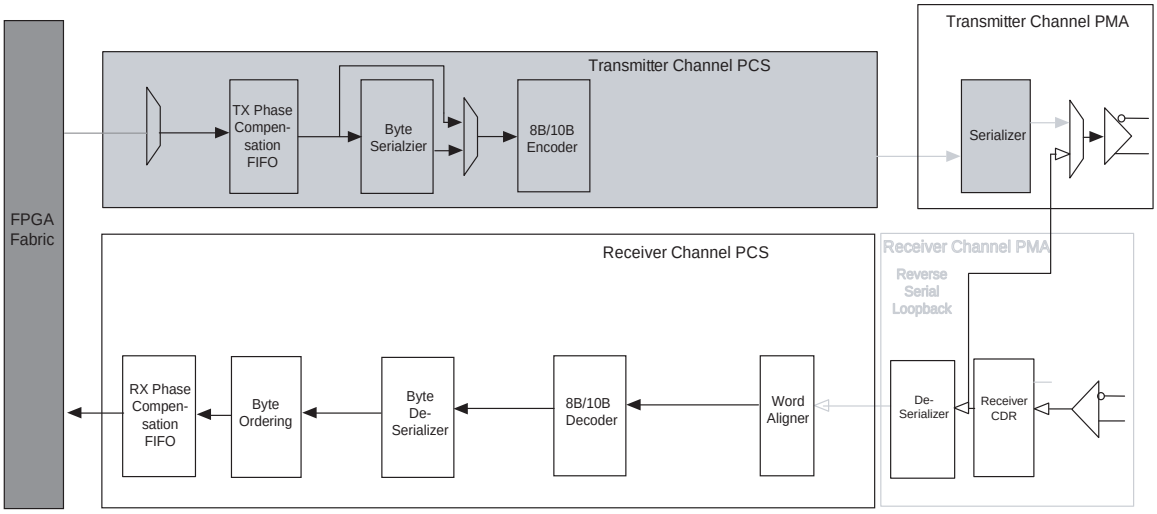
Configuration	8B/10B Encoder	Byte Serializer	Data Rate Range	Supported Channel Width Setting in the ALTGX MegaWizard Plug-In Manager for Parallel Loopback
Single-width mode	Enabled	Enabled	600 Mbps to 3.125Gbps	16
Double-width mode	Enabled	Disabled	1Gbps to 5Gbps	16

The status signals `rx_bistdone` and `rx_bisterr` indicate the status of the verifier. The `rx_bistdone` port is asserted and stays high when the verifier either receives one full cycle of incremental pattern or it detects an error in the receiver data. The `rx_bisterr` signal is asserted and stays high when the verifier detects an error. You can reset the incremental pattern generator and verifier by asserting the `tx_digitalreset` and `rx_digitalreset` signals, respectively.

Reverse Serial Loopback

Reverse serial loopback is available as a subprotocol under Basic functional mode. In reverse serial loopback mode, the data is received through the `rx_datain` port, retimed through the receiver CDR and sent out to the `tx_dataout` port. The received data is also available to the FPGA logic. Figure 1–153 shows the transceiver channel datapath for reverse serial loopback mode. The active block of the transmitter channel is only the transmitter buffer. You can change the output differential voltage and the pre-emphasis first post tap values on the transmitter buffer through the ALTGX MegaWizard Plug-In Manager or through the dynamic reconfiguration controller. Reverse serial loopback is often implemented when using a bit error rate tester (BERT) on the upstream transmitter.

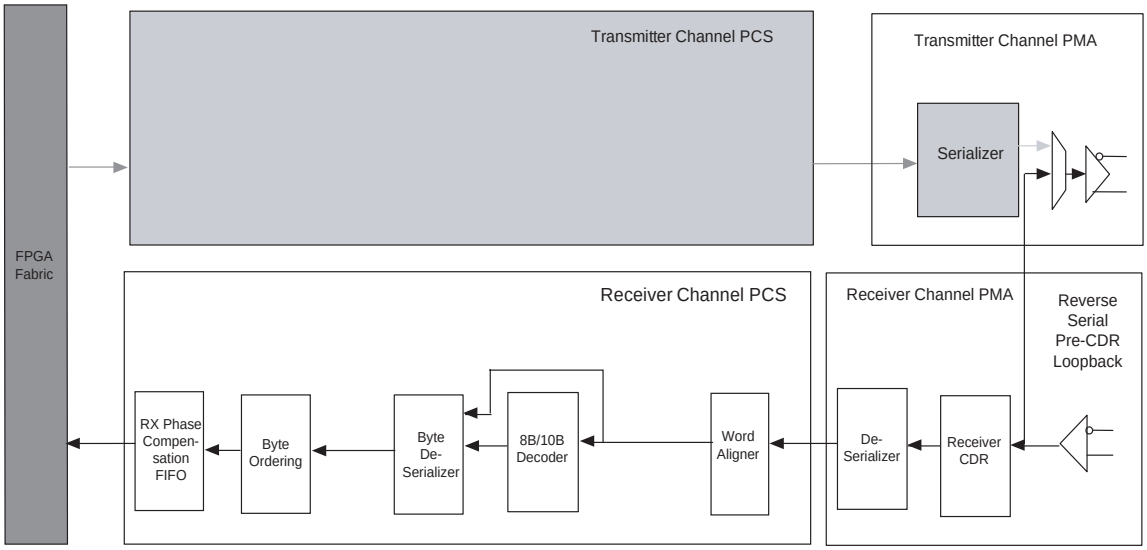
Figure 1–153. Reverse Serial Loopback Datapath (Grayed-Out Blocks are Not Active in this Mode)



Reverse Serial Pre-CDR Loopback

The reverse serial pre-CDR loopback is available as a subprotocol under Basic functional mode. In reverse serial pre-CDR loopback, the data received through the `rx_datain` port is looped back to the `tx_dataout` port *before* the receiver CDR. The received data is also available to the FPGA logic. Figure 1–154 shows the transceiver channel datapath for reverse serial pre-CDR loopback mode. The active block of the transmitter channel is only the transmitter buffer. You can change the output differential voltage on the transmitter buffer through the ALTGX MegaWizard Plug-In Manager. The pre-emphasis settings for the transmitter buffer cannot be changed in this configuration.

Figure 1–154. Reverse Serial Pre-CDR Loopback Datapath



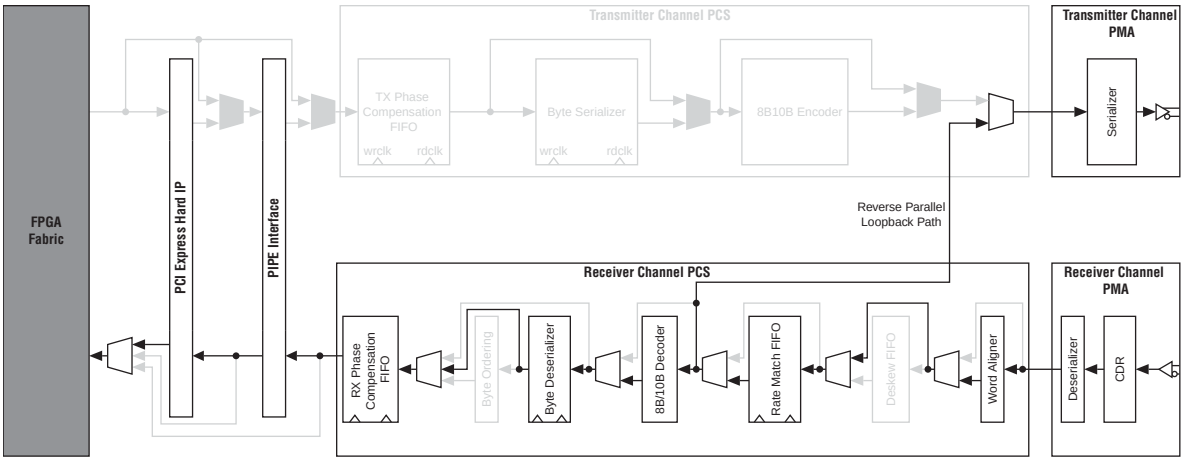
PCI Express (PIPE) Reverse Parallel Loopback

PCI Express (PIPE) reverse parallel loopback is only available in PCI Express (PIPE) functional mode for Gen1 and Gen2 data rates. As shown in Figure 1–155, the received serial data passes through the receiver CDR, deserializer, word aligner, and rate matching FIFO buffer. It is then looped back to the transmitter serializer and transmitted out through the `tx_dataout` port. The received data is also available to the FPGA fabric through the `rx_dataout` port. This loopback mode is compliant with the PCI Express (PIPE) specification 2.0. To enable this loopback mode, assert the `tx_detectrxloopback` port.

This is the only loopback option supported in PCI Express (PIPE) functional mode.

In Figure 1–155, the grayed areas show the inactive paths when the PCI Express (PIPE) reverse parallel loopback mode is enabled.

Figure 1–155. PCI Express (PIPE) Reverse Parallel Loopback Mode Datapath (Grayed-Out Blocks are Not Active in this Mode)



Auxiliary Transmit (ATX) PLL Block

Stratix IV GX and GT transceivers contain the ATX PLL block that you can use to generate high-speed clocks for the transmitter channels on the same side of the device. Each:

- Stratix IV GX device has 6G ATX PLL
- Stratix IV GT device has 6G ATX PLL and 10G ATX PLLs

For data rates supported by these ATX PLLs, refer to the *Stratix IV Device Datasheet* section.

6G ATX PLL Block

Stratix IV GX can have either two (one on each side of the device) or four (two on each side of the device) 6G ATX PLLs, depending on the specific devices.

For data rates supported by 6G ATX PLLs, refer to the *Stratix IV Device Datasheet* section.

10G ATX PLL Block

Each Stratix IV GT device has two 10G ATX PLL blocks, one located on each side of the device. The 10G ATX PLLs provide low-jitter transceiver clocks to implement 40G/100G Ethernet and SFI-S links specified by IEEE802.3ba and OIF specifications.

In EP4S40G2F40 and EP4S40G5H40 devices, you can use each 10G ATX PLL to generate transceiver clocks for up to six channels at data rates of up to 11.3 Gbps each.

In EP4S100G2F40, EP4S100G5H40, and EP4S100G5F45 devices, you can use each 10G ATX PLL to generate transceiver clocks for up to 12 channels at data rates of up to 11.3 Gbps each.

Figure 1–159 and Figure 1–160 show transceiver channels that support data rates up to 11.3 Gbps in each Stratix IV GT device.

The 10G ATX PLL block consists of:

- 10G ATX PLL—Synthesizes the input reference clock to generate the high-speed serial transceiver clock at frequency of half the configured data rate
- ATX clock divider block—Divides the high-speed serial clock from the 10G ATX PLL to generate the low-speed parallel transceiver clock

The 10G ATX PLL architecture is functionally similar to the 6G ATX PLL architecture, except that it is optimized for the 10 Gbps data rate range.

Figure 1–156 shows the location of the ATX PLL blocks in two transceiver block device families.

Figure 1–156. Location of ATX PLL Blocks in a Four-Transceiver Block Stratix IV GX Device (Two on Each Side)

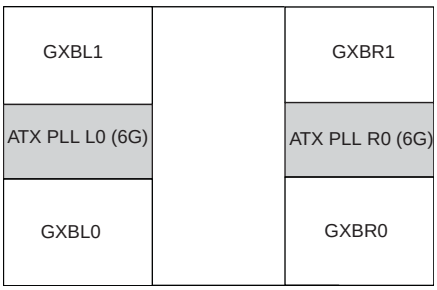


Figure 1–157 shows the location of the ATX PLL blocks in three transceiver block device families (for 230K and 530K devices and all other devices).

Figure 1–157. Location of ATX PLL Blocks with a Six Transceiver Block Stratix IV GX Device (Three on Each side)

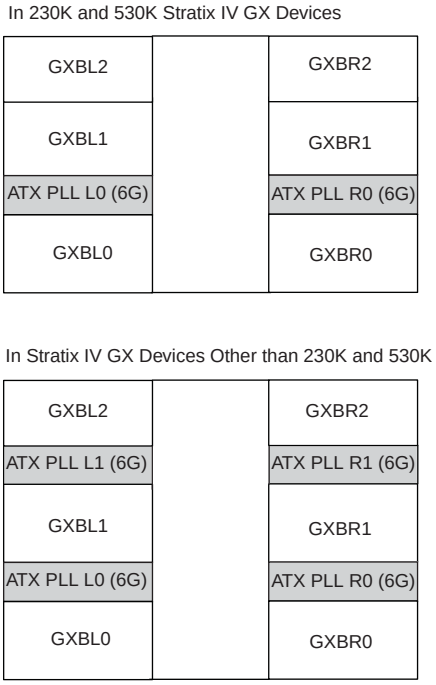


Figure 1–158 shows the location of the ATX PLL blocks in four transceiver block device families.

Figure 1–158. Location of ATX PLL Blocks in an Eight-Transceiver Block Stratix IV GX Device (Four on Each Side)

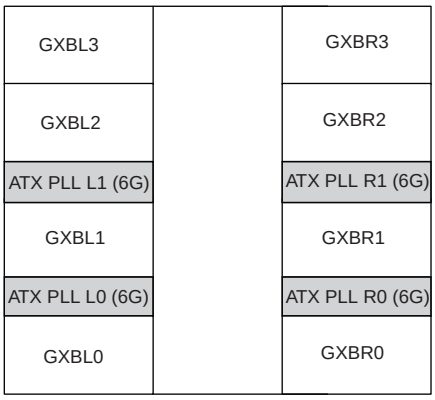


Figure 1–159 and Figure 1–160 show the locations of the 6G and 10G ATX PLLs in each Stratix IV GT device.

Figure 1–159. Location of Transceiver Channel and PLL in Stratix IV GT Devices (EP4S40G2F40, EP4S40G5H40, EP4S100G2F40 and EP4S100G5H40)

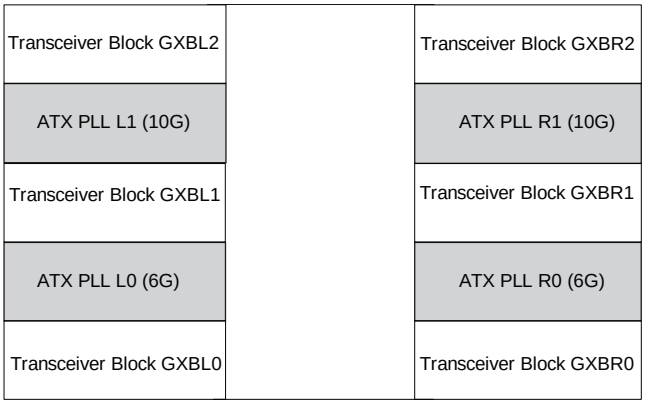
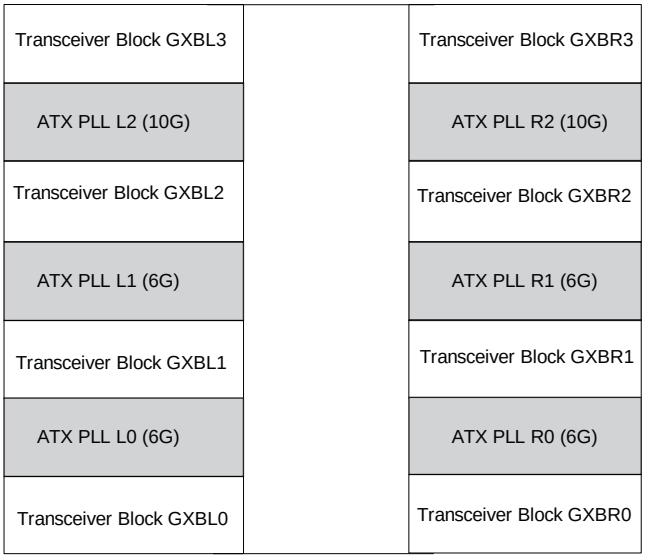


Figure 1–160. Location of Transceiver Channel and PLL in Stratix IV GT Devices (EP4S100G5F45)



Input Reference Clocks for the ATX PLL Block

The 6G ATX PLL block does not have a dedicated reference clock pin. The following are the possible input reference clock sources:

- REFCLKs from the transceiver blocks on the same side of the device if the corresponding CMU channels are not used as transceiver channels
- Input reference clock provided through the PLL cascade clock network
- Clock inputs connected through the global clock lines

Altera recommends that you use the REFCLKs pins from the adjacent transceiver block below the ATX PLL block to improve performance.

For the 10G ATX PLL, Stratix IV GT devices only allow driving the reference clock source from one of the dedicated `reflck` pins on the same side of the device.

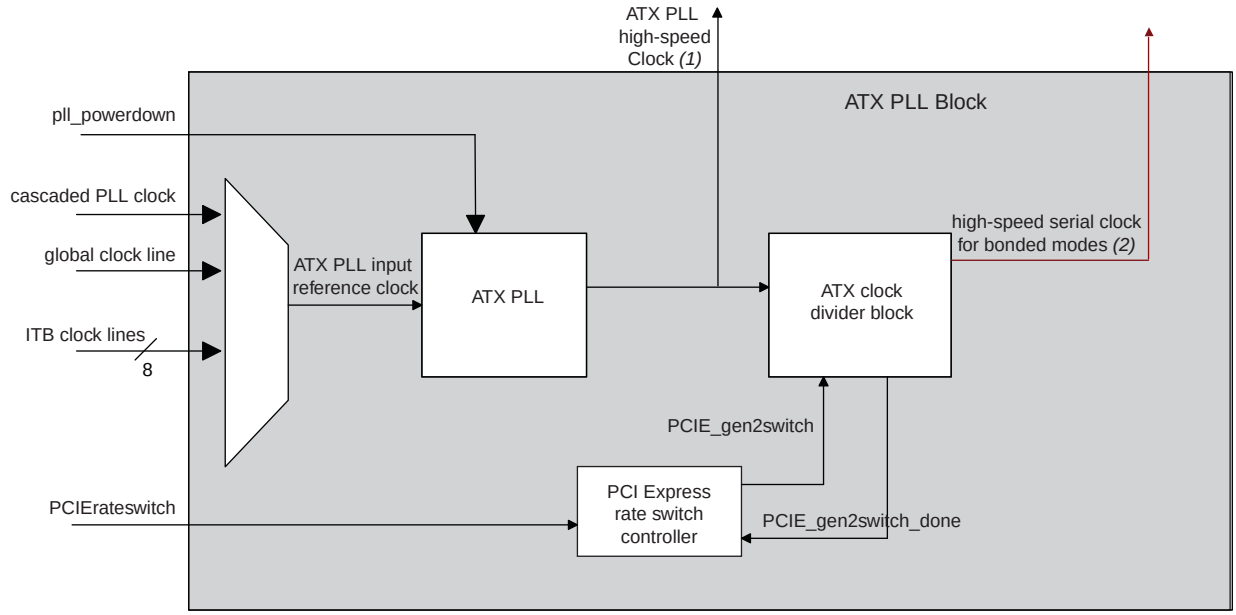
For improved jitter performance, Altera strongly recommends using the REFCLK pins of the transceiver block located immediately below the 10G ATX PLL block to drive the input reference clock.

For more information about the input reference clocks for ATX PLLs, refer to the *Stratix IV Transceiver Clocking* chapter.

Architecture of the ATX PLL Block

Figure 1–161 shows the ATX PLL block components (the ATX PLL, ATX clock divider, and a shared control signal generation block).

Figure 1–161. ATX PLL Block



Notes to Figure 1–161:

(1) In non-bonded functional modes (for example, CEI functional mode), the transmitter channel uses the transmitter local clock divider to divide this high-speed clock output to provide clocks for its PMA and PCS blocks.

(2) This is used in Basic ×4, ×8, and PCI Express (PIPE) ×4 and ×8 functional modes.

The functional blocks on the ATX PLL are similar to the blocks explained in "CMU0 PLL" on page 1–99. The values of the /M and /L divider settings in the ATX PLL are automatically selected by the Quartus II software based on the transceiver channel configuration.

The ATX PLL high-speed clock output provides high-speed serial clocks for non-bonded functional modes such as CEI (with the "none" subprotocol).

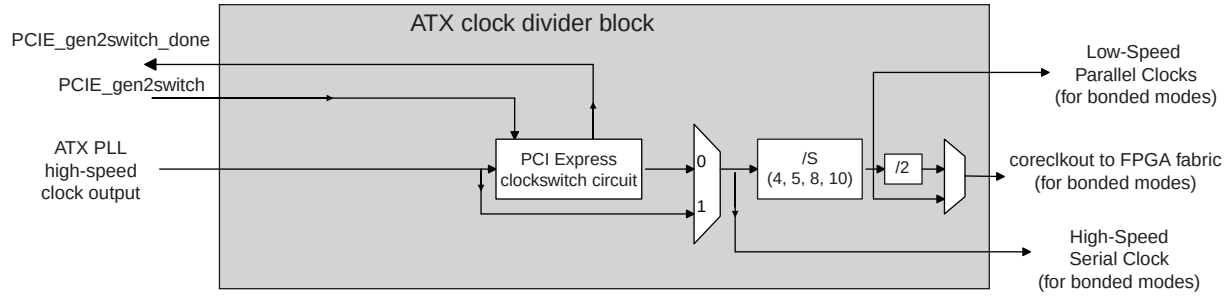
ATX Clock Divider

The ATX clock divider divides the ATX PLL high-speed clock and provides high-speed serial and low-speed parallel clock for bonded functional modes such as PCI Express (PIPE) (×4 and ×8), Basic ×4 and ×8, and PMA-Direct mode with ×N configuration. For PCI Express (PIPE) functional mode support, the ATX clock divider consists of the PCI Express (PIPE) rateswitch circuit to enable dynamic rateswitch between PCI Express (PIPE) Gen1 and Gen2 data rates. For more information on this circuit, refer to "CMU0 Channel" on page 1–98.

The clock outputs from the ATX PLL block are provided to the transmitter channels through the `×N_Top` or `×N_bottom` clock lines, as shown in Figure 1–162.

For more information, refer to the *Stratix IV Transceiver Clocking* chapter.

Figure 1–162. ATX Clock Divider



The Differences Between 10G ATX PLL, 6G ATX PLL, and CMU PLL

Table 1–70 lists the differences between the 10G ATX PLL, 6G ATX PLL, and CMU PLL.

Table 1–70. Differences Between the 10G ATX PLL, 6G ATX PLL, and CMU PLL (Part 1 of 2)

Difference Category/PLLs	10G ATX PLL	6G ATX PLL	CMU PLL
Available in	Stratix IV GT device	Stratix IV GX and GT devices	Stratix IV GX and GT devices
Data rates (Gbps)	9.9 to 11.3	4.8 to 5.4 and 6.0 and 6.5 2.4 to 2.7 and 3.0 and 3.25 *(1)* 1.2 to 1.35 and 1.5 to 1.625 *(1)*	0.6 to 8.5
Input reference clock options	Only dedicated `refclk` pins on the same side of the device *(2)*, *(3)*	■ Clock inputs connected through the inter transceiver block (ITB) lines. ■ Clock inputs connected through the PLL cascade clock network. ■ Clock inputs connected through the global clock lines. *(3)*	■ Clock inputs connected through the inter transceiver block (ITB) lines. ■ clock inputs connected through the PLL cascade clock network. ■ Clock inputs connected through the global clock lines, `refclk0` and `refclk1` clock input, dedicated `refclks` in the transceiver block. *(3)*

Table 1–70. Differences Between the 10G ATX PLL, 6G ATX PLL, and CMU PLL (Part 2 of 2)

Difference Category/PLLs	10G ATX PLL	6G ATX PLL	CMU PLL
Power Supply—$V_{CCA_L/R}$ (V) options for PLLs	3.3	3.0 or 3.3 *(4)*	2.5 or 3.0 or 3.3 *(4)*
Phase noise	Lower when compared with the CMU PLL *(5)*	Lower when compared with the CMU PLL *(5)*	Higher when compared with the ATX PLLs *(5)*

Notes to Table 1–70:

(1) Using the L dividers available in ATX PLLs.

(2) For improved jitter performance, Altera strongly recommends using the `refclk` pins of the transceiver block located immediately below the 10G ATX PLL block to drive the input reference clock.

(3) For more information, refer to the Input Reference Clock Source table in the *Stratix IV Transceiver Clocking* chapter.

(4) Option in Stratix IV GT devices.

(5) For more information on phase noise and PLL bandwidths of ATX and CMU PLLs, refer to the characterization reports.

Calibration Blocks

Stratix IV GX and GT devices contain calibration circuits that calibrate the OCT resistors and the analog portions of the transceiver blocks to ensure that the functionality is independent of process, voltage, or temperature variations.

Calibration Block Location

Figure 1–163 shows the location and number of calibration blocks available for different Stratix IV GX and GT devices. In Figure 1–163 through Figure 1–168, the calibration block R0 and L0 refer to the calibration blocks on the right and left side of the devices, respectively.

Figure 1–163. Calibration Block Locations in Stratix IV GX and GT Device with Two Transceiver Blocks (on Each Side)

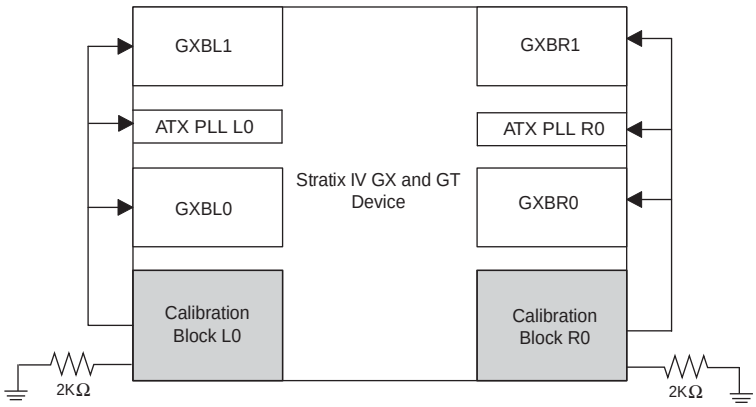


Figure 1–164 shows Stratix IV GX 230K and 530K devices that have three transceiver blocks each on the left and right side and one ATX PLL block on each side.

Figure 1–164. Calibration Block Locations in Stratix IV GX 230K and 530K Devices with Three Transceiver Blocks (on Each Side)

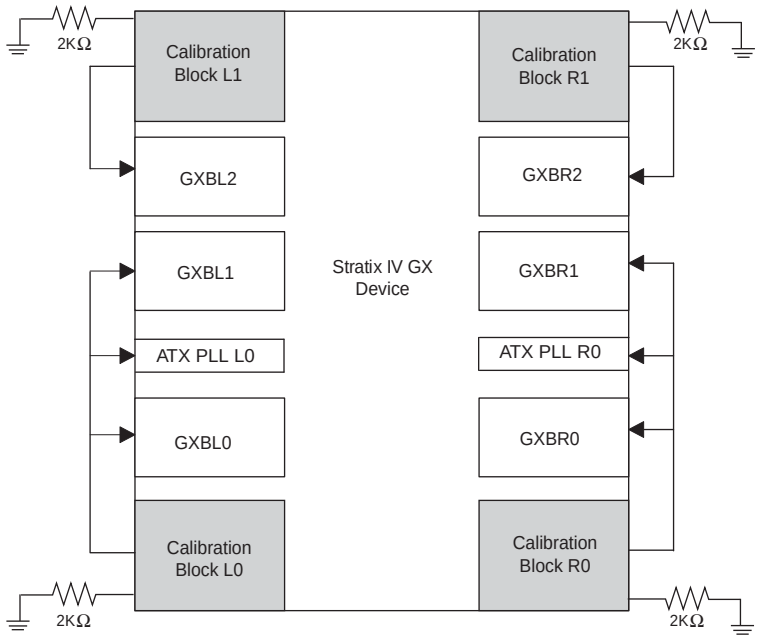


Figure 1–165 shows Stratix IV GX devices other than 230K and 530K that have three transceiver blocks each on the left and right side and two ATX PLL blocks on each side.

Figure 1–165. Calibration Block Locations in Stratix IV GX Devices Other than 230K and 530K with Three Transceiver Blocks (on Each Side)

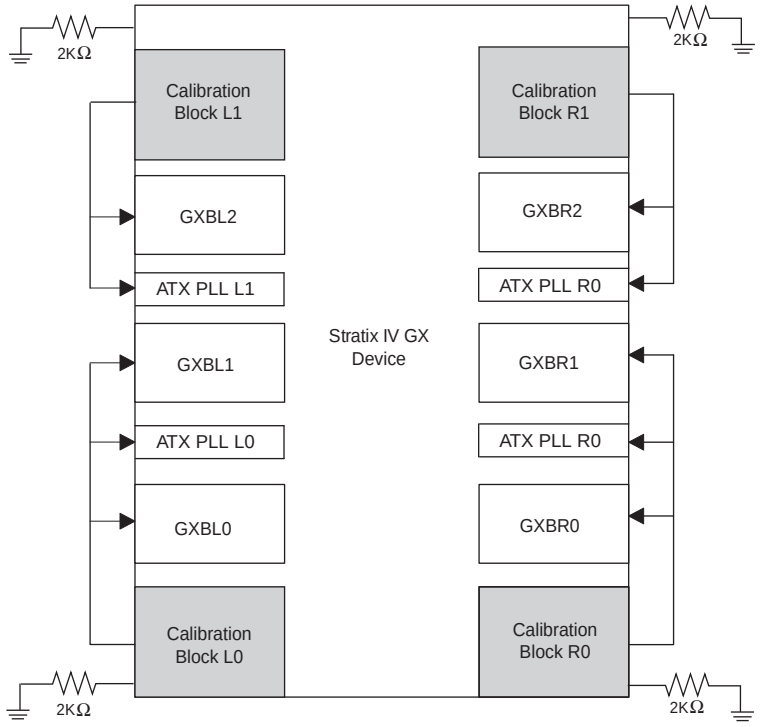


Figure 1–166 shows Stratix IV GX devices that have four transceiver blocks each on the left and right side and two ATX PLL blocks on each side.

Figure 1–166. Calibration Block Locations in Stratix IV GX Devices with Four Transceiver Blocks (on Each Side)

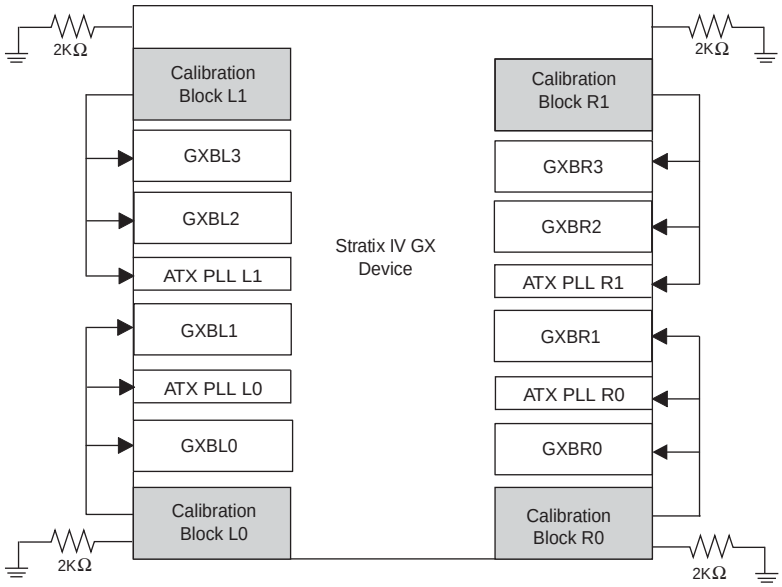
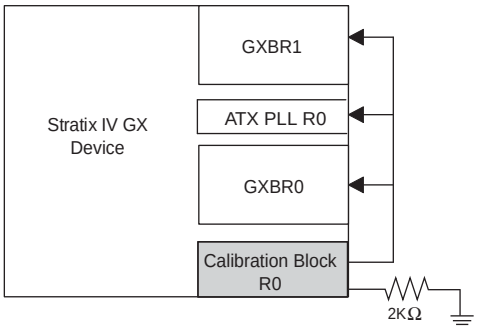


Figure 1–167 shows Stratix IV GX devices that have two transceiver blocks only on the right side of the device.

Figure 1–167. Calibration Block Locations in Stratix IV GX Devices with Two Transceiver Blocks (Right Side Only)



The Quartus II software automatically selects the appropriate calibration block based on the assignment of the transceiver `tx_dataout` and `rx_datain` pins.

Calibration

The calibration block internally generates a constant internal reference voltage, independent of process, voltage, or temperature variations. It uses the internal reference voltage and external reference resistor (you must connect the resistor to the RREF pin) to generate constant reference currents. These reference currents are used by the analog block calibration circuit to calibrate the transceiver blocks.

The OCT calibration circuit calibrates the OCT resistors present in the transceiver channels. You can enable the OCT resistors in the transceiver channels through the ALTGX MegaWizard Plug-In Manager.

You must connect a separate 2 kΩ (tolerance max ± 1%) external resistor on each RREF pin in the Stratix IV GX and GT device to ground. To ensure proper operation of the calibration block, the RREF resistor connection in the board must be free from external noise.

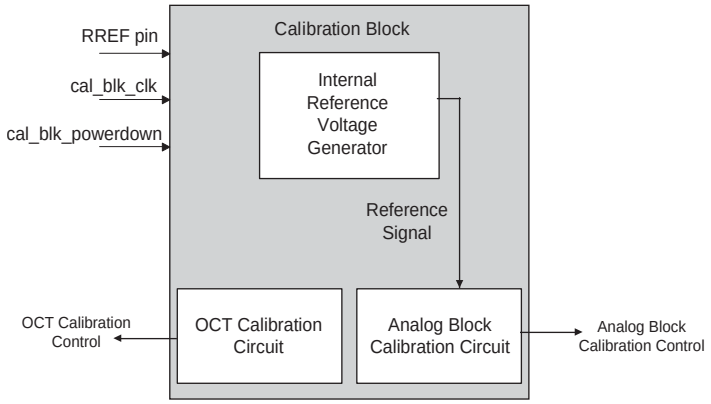
Input Signals to the Calibration Block

The ALTGX MegaWizard Plug-In Manager provides the `cal_blk_clk` and `cal_blk_powerdown` ports to control the calibration block:

- `cal_blk_clk`—you must use the `cal_blk_clk` port to provide input clock to the calibration clock. The frequency of `cal_blk_clk` must be within 10 MHz to 125 MHz (this range is preliminary. Final values will be available upon characterization). You can use dedicated clock routes such as the global or regional clock. If you do not have suitable input reference clock or dedicated clock routing resources available, use divide-down logic from the FPGA fabric to generate a slow clock and use local clocking routing. Drive the `cal_blk_clk` port of all ALTGX instances that are associated with the same calibration block from the same input pin or logic.
- `cal_blk_powerdown`—you can perform calibration multiple times by using the `cal_blk_powerdown` port available through the ALTGX MegaWizard Plug-In Manager. Assert this signal for approximately 500 ns (this is preliminary. Final values will be available upon characterization). Following de-assertion of `cal_blk_powerdown`, the calibration block restarts the calibration process. Drive the `cal_blk_powerdown` port of all ALTGX instances that are associated with the same calibration block from the same input pin or logic.

Figure 1–168 shows the required inputs to the calibration block.

Figure 1–168. Input Signals to the Calibration Blocks



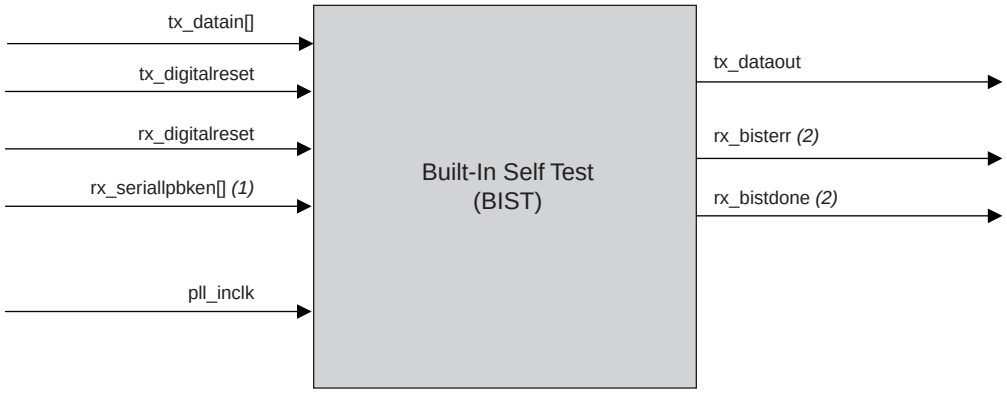
Built-In Self Test Modes

This section describes Built-In Self Test (BIST) modes.

BIST Mode Pattern Generators and Verifiers

Each transceiver channel in the Stratix IV GX and GT devices contain a different BIST pattern generator and verifier. Using these BIST patterns, you can verify the functionality of the functional blocks in the transceiver channel without requiring user logic. The BIST functionality is provided as an optional mechanism for debugging transceiver channels. Figure 1–169 shows the enabled input and output ports when you select BIST mode (except incremental patterns).

Figure 1–169. Input and Output Ports for BIST Modes



Notes to Figure 1–169:

(1) `rx_serialpbken` is required in PRBS.

(2) `rx_bisterr` and `rx_bistdone` are only available in PRBS and BIST modes.

Three types of pattern generators and verifiers are available:

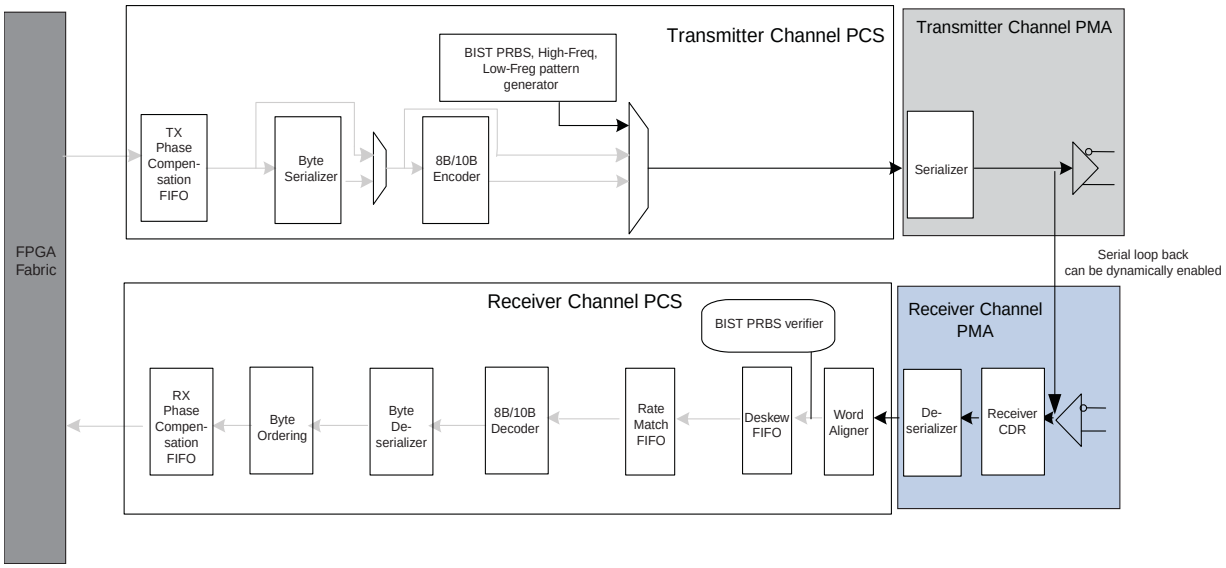
- BIST incremental data generator and verifier—This is only available in parallel loopback mode. For more information, refer to "Serial Loopback" on page 1–188.
- High frequency and low frequency pattern generator—The high frequency patterns generate alternate ones and zeros and the low frequency patterns generate five ones and five zeroes in single-width mode and ten ones and ten zeroes in double-width mode. These patterns do not have a corresponding verifier. You can enable the **serial loopback** option to dynamically loop the generated pattern to the receiver channel using the `rx_seriallpbken` port. Therefore, the 8B/10B encoder/decoder blocks are bypassed in the Basic PRBS mode.
- Pseudo Random Binary Sequence (PRBS) generator and verifier—The PRBS generator and verifier interface with the serializer and deserializer in the PMA blocks. The advantage of using a PRBS data stream is that the randomness yields an environment that stresses the transmission medium. In the data stream, you can observe both random jitter and deterministic jitter using a time interval analyzer, bit error rate tester, or oscilloscope. The PRBS repeats after completing an iteration. The number of bits the PRBSx pattern sends before repeating the pattern is $(2^{x} - 1)$ bits.

Different PRBS patterns are available as a subprotocol under Basic functional mode for single-width and double-width mode, as shown in the following sections.

You can enable the **serial loopback** option in Basic PRBS mode to loop the generated pattern to the receiver channel. This creates a `rx_seriallpbken` port that you can use to dynamically control the serial loopback. The 8B/10B encoder/decoder blocks are bypassed in Basic PRBS mode.

Figure 1–170 shows the datapath for the PRBS patterns. The generated PRBS pattern is sent to the transmitter serializer. The verifier checks the data from the word aligner.

Figure 1–170. BIST PRBS, High Frequency, and Low Frequency Pattern Datapath



PRBS in Single-Width Mode

Table 1–71 shows various PRBS patterns and corresponding word alignment patterns for PRBS in single-width mode configuration.

Table 1–71. Available PRBS, High Frequency, and Low Frequency Patterns in Single-Width Mode

Patterns	Polynomial	Channel Width of 8 Bit *(1)*	Word Alignment Pattern with Channel Width 8 Bit	Maximum Data Rate With Channel Width 8 Bit (Gbps)	Channel Width of 10 Bit *(1)*	Word Alignment Pattern	Maximum Data Rate with Channel Width 10 Bit (Gbps)
PRBS 7	$X^7 + X^6 + 1$	Y	16'h3040	2.5	N	NA	N/A
PRBS 8	$X^8 + X^7 + 1$	Y	16'hFF5A	2.5	N	NA	N/A
PRBS 10	$X^{10} + X^7 + 1$	N	NA	N/A	Y	10'h3FF	3.125
PRBS 23	$X^{23} + X^{18} + 1$	Y	16'hFFFF	2.5	N	NA	N/A
High frequency *(2)*	1010101010	Y	NA	2.5	Y	NA	3.125
Low Frequency *(2)*	0000011111	N	NA	N/A	Y	NA	3.125

Notes to Table 1–71:

(1) Channel width refers to the **What is the channel width?** option in the **General** screen of the ALTGX MegaWizard Plug-In Manager. Based on the selection, an 8 or 10 bits wide pattern is generated as indicated by a **Yes (Y)** or **No (N)**.

(2) A verifier is not available for the specified patterns.

The status signals `rx_bistdone` and `rx_bisterr` indicate the status of the verifier. The `rx_bistdone` port gets asserted and stays high when the verifier either receives one full cycle of incremental pattern or it detects an error in the receiver data. The `rx_bisterr` signal gets asserted and stays high when the verifier detects an error. You can reset the PRBS pattern generator and verifier by asserting the `tx_digitalreset` and `rx_digitalreset` signals, respectively.

PRBS in Double-Width Mode

Table 1–72 shows various PRBS patterns and corresponding word alignment patterns for PRBS in double-width mode configuration.

Table 1–72. Available PRBS, High Frequency, and Low Frequency Patterns in Double-Width Mode (Part 1 of 2)

Patterns	Polynomial	Channel Width of 16-Bit *(1)*	Word Alignment Pattern with Channel Width of 16-Bit	Maximum Data Rate with Channel Width of 16-Bit (Gbps)	Channel Width of 20-Bit *(1)*	Word Alignment Pattern	Maximum Data Rate with Channel Width of 20-Bit (Gbps)
PRBS 7	$X^7 + X^6 + 1$	Y	16'h3040	5	Y	20'h43040	6.375
PRBS 23	$X^{23} + X^{18} + 1$	Y	32'h007FFFFF	5	Y	40'h00007FFFFF	6.375
High frequency *(2)*	1010101010	Y	NA	5	Y	N/A	6.375

Table 1–72. Available PRBS, High Frequency, and Low Frequency Patterns in Double-Width Mode (Part 2 of 2)

Patterns	Polynomial	Channel Width of 16-Bit *(1)*	Word Alignment Pattern with Channel Width of 16-Bit	Maximum Data Rate with Channel Width of 16-Bit (Gbps)	Channel Width of 20-Bit *(1)*	Word Alignment Pattern	Maximum Data Rate with Channel Width of 20-Bit (Gbps)
Low Frequency *(2)*	0000011111	N	NA	N/A	Y	N/A	6.375

Notes to Table 1–72:

(1) Channel width refers to the **what is the channel width?** option in the **General** screen of the ALTGX MegaWizard Plug-In Manager. Based on the selection, A 16 or 20 bits wide pattern is generated as indicated by a **Yes (Y)** or **No (N)**.

(2) Verifier is not available for the specified patterns.

The status signals `rx_bisterr` and `rx_bistdone` are available to indicate the status of the verifier. For more information about the behavior of these status signals, refer to "Single-Width Mode" on page 1–17.

Transceiver Port Lists

Instantiate the Stratix IV GX and GT transceivers using the ALTGX megafunction instance in the Quartus II MegaWizard Plug-In Manager. The ALTGX megafunction instance allows you to configure transceivers for your intended protocol and select optional control and status ports to and from the instantiated transceiver channels.

Table 1–73 through Table 1–79 provide a brief description of the ALTGX megafunction ports.

Table 1–73 lists the ALTGX megafunction transmitter ports.

Table 1–73. Stratix IV GX and GT ALTGX Megafunction Ports: Transmitter Ports (Part 1 of 4)

Port Name	Input/ Output	Clock Domain	Description	Scope
Transmitter Phase Compensation FIFO				
`tx_datain`	Input	Synchronous to `tx_clkout` or `coreclkout`. `tx_clkout` for non-bonded modes. `coreclk` for bonded modes.	Parallel data input from the FPGA fabric to the transmitter. ■ Bus width—depends on the channel width multiplied by the number of channels per instance.	Channel

Table 1–73. Stratix IV GX and GT ALTGX Megafunction Ports: Transmitter Ports (Part 2 of 4)

Port Name	Input/ Output	Clock Domain	Description	Scope
`tx_clkout`	Output	Clock signal	FPGA fabric-transceiver interface clock. ■ Bonded channel configurations—not available. ■ Non-bonded channel configurations—each channel has a `tx_clkout` signal. ■ Use this clock signal to clock the parallel data `tx_datain` from the FPGA fabric into the transmitter.	Channel
`tx_coreclk`	Input	Clock signal	Optional write clock port for the transmitter phase compensation FIFO. ■ If not selected—the Quartus II software automatically selects `tx_clkout/coreclkout` as the write clock for transmitter phase compensation FIFO. ■ If selected—you must drive this port with a clock that is frequency locked to `tx_clkout/coreclkout`.	Channel
`tx_phase_comp_fifo_error`	Output	Synchronous to `tx_clkout/coreclkout` clock signal.	Transmitter phase compensation FIFO full or empty indicator. ■ A high level—the transmitter phase compensation FIFO is either full or empty.	Channel
8B/10B Encoder				
`tx_ctrlenable`	Input	Synchronous to `tx_clkout/coreclkout` clock signal.	8B/10B encoder /Kx.y/ or /Dx.y/ control. ■ When asserted high—the 8B/10B encoder encodes the data on the `tx_datain` port as a /Kx.y/ control code group. ■ When de-asserted low—it encodes the data on the `tx_datain` port as a /Dx.y/ data code group. ■ Channel Width: 8—`tx_ctrlenable` = 1 16—`tx_ctrlenable` = 2 32—`tx_ctrlenable` = 4	Channel

Table 1–73. Stratix IV GX and GT ALTGX Megafunction Ports: Transmitter Ports (Part 3 of 4)

Port Name	Input/ Output	Clock Domain	Description	Scope
`tx_forcedisp`	Input	Asynchronous signal. Minimum pulse width is two parallel clock cycles.	8B/10B encoder force disparity control. ■ When asserted high—forces the 8B/10B encoder to encode the data on the `tx_datain` port with a positive or negative disparity depending on the `tx_dispval` signal level. ■ When de-asserted low—the 8B/10B encoder encodes the data on the `tx_datain` port according to the 8B/10B running disparity rules. ■ Channel Width: 8—`tx_forcedisp` = 1 16—`tx_forcedisp` = 2 32—`tx_forcedisp` = 4	Channel
`tx_dispval`	Input	Asynchronous signal. Minimum pulse width is two parallel clock cycles.	8B/10B encoder force disparity value. ■ A high level—when the `tx_forcedisp` signal is asserted high, it forces the 8B/10B encoder to encode the data on the `tx_datain` port with a negative starting running disparity. ■ A low level—when the `tx_forcedisp` signal is asserted high, it forces the 8B/10B encoder to encode the data on the `tx_datain` port with a positive starting running disparity. ■ Channel Width: 8—`tx_dispval` = 1 16—`tx_dispval` = 2 32—`tx_dispval` = 4	Channel
`tx_invpolarity`	Input	Asynchronous signal. Minimum pulse width is two parallel clock cycles.	Transmitter polarity inversion control. This feature is useful for correcting situations in which the positive and negative signals of the differential serial link are accidentally swapped during board layout. ■ When asserted high in single-width modes—the polarity of every bit of the 8-bit or 10-bit input data to the serializer gets inverted. ■ When asserted high in double-width modes—the polarity of every bit of the 16-bit or 20-bit input data to the serializer gets inverted.	Channel

Table 1–73. Stratix IV GX and GT ALTGX Megafunction Ports: Transmitter Ports (Part 4 of 4)

Port Name	Input/ Output	Clock Domain	Description	Scope
Transmitter Physical Media Attachment				
`tx_dataout`	Output	N/A	Transmitter serial data output port.	Channel
`fixedclk`	Input	Clock signal	125-MHz clock for receiver detect and offset cancellation in PCI Express (PIPE) mode.	Channel

Table 1–74 lists the ALTGX megafunction receiver ports.

Table 1–74. Stratix IV GX and GT ALTGX Megafunction Ports: Receiver Ports (Part 1 of 8)

Port Name	Input/ Output	Clock Domain	Description	Scope
`rx_syncstatus`	Output	Synchronous to `coreclkout` clock signal	Word alignment synchronization status indicator. ■ Automatic synchronization state machine mode—this signal is driven high if the conditions required to remain in synchronization are met. Driven low if the conditions required to lose synchronization are met. ■ Manual alignment mode—the behavior of this signal depends on whether the transceiver is configured in single-width or double-width mode. ■ Bit-Slip mode—not available. For more information, refer to "Word Aligner in Single-Width Mode" on page 1–57 and "Word Aligner in Double-Width Mode" on page 1–63. ■ Channel width: 8/10—`rx_syncstatus` = 1 16/20—`rx_syncstatus` = 2 32/40—`rx_syncstatus` = 4	Channel
`rx_bitslip`	Input	Asynchronous signal. Minimum pulse width is two parallel clock cycles.	Bit-slip control for the word aligner configured in bit-slip mode. At every rising edge, word aligner slips one bit into the received data stream, effectively shifting the word boundary by one bit.	Channel
`rx_ala2size`	Input	Asynchronous Signal. Minimum pulse width is two parallel clock cycles.	Available only in SONET OC-12 and OC-48 modes. Select between these options: ■ 0 = 16-bit A1A2 ■ 1 = 32-bit A1A1A2A2	Channel

Table 1–74. Stratix IV GX and GT ALTGX Megafunction Ports: Receiver Ports (Part 2 of 8)

Port Name	Input/ Output	Clock Domain	Description	Scope
`rx_rlv`	Output	Asynchronous signal. Driven for a minimum of two recovered clock cycles in configurations without byte serializer and a minimum of three recovered clock cycles in configurations with byte serializer.	Run-length violation indicator. A high pulse is driven when the number of consecutive 1s or 0s in the received data stream exceeds the programmed run length violation threshold.	Channel
`rx_invpolarity`	Input	Asynchronous Signal. Minimum pulse width is two parallel clock cycles.	Generic receiver polarity inversion control. Useful feature for correcting situations where the positive and negative signals of the differential serial link are accidentally swapped during board layout. ■ When asserted high in single-width modes—the polarity of every bit of the 8-bit or 10-bit input data word to the word aligner gets inverted. ■ When asserted high in double-width modes—the polarity of every bit of the 16-bit or 20-bit input data to the word aligner gets inverted.	Channel
`rx_revbitorderwa`	Input	Asynchronous Signal. Minimum pulse width is two parallel clock cycles.	Receiver bit reversal control. This is a useful feature where the link transmission order is MSB to LSB. ■ Available only in Basic single-width and double-width modes with the word aligner configured in bit-slip mode. ■ When asserted high in Basic single-width modes—the 8-bit or 10-bit data `D[7:0]` or `D[9:0]` at the output of the word aligner gets rewired to `D[0:7]` or `D[0:9]`, respectively. ■ When asserted high in Basic double-width modes—the 16-bit or 20-bit data `D[15:0]` or `D[19:0]` at the output of the word aligner gets rewired to `D[0:15]` or `D[0:19]`, respectively.	Channel

Table 1–74. Stratix IV GX and GT ALTGX Megafunction Ports: Receiver Ports (Part 3 of 8)

Port Name	Input/ Output	Clock Domain	Description	Scope
rx_revbyteorderwa	Input	Asynchronous Signal. Minimum pulse width is two parallel clock cycles.	Receiver byte reversal control. This is a useful feature in situations where the MSByte and LSByte of the transmitted data are erroneously swapped. ■ Available only in Basic double-width mode. ■ When asserted high, the MSByte and LSByte of the 16- and 20-bit data at the output of the word aligner get swapped.	Channel
Deskew FIFO				
rx_channelaligned	Output	Synchronous to coreclkout clock signal	XAUI deskew FIFO channel aligned indicator. ■ Available only in XAUI mode. ■ A high level—the XAUI deskew state machine is either in ALIGN_ACQUIRED_1, ALIGN_ACQUIRED_2, ALIGN_ACQUIRED_3, or ALIGN_ACQUIRED_4 state, as specified in the PCS deskew state diagram in the IEEE P802.3ae specification. ■ A low level—the XAUI deskew state machine is either in LOSS_OF_ALIGNMENT, ALIGN_DETECT_1, ALIGN_DETECT_2, or ALIGN_DETECT_3 state, as specified in the PCS deskew state diagram in the IEEE P802.3ae specification.	Transceiver block
Rate Match (Clock Rate Compensation) FIFO				
rx_rmfifodatainserted	Output	Synchronous to tx_clkout or coreclkout. tx_clkout for non-bonded modes. coreclkout for bonded modes.	Rate match FIFO insertion status indicator. ■ A high level—the rate match pattern byte has inserted to compensate for the PPM difference in reference clock frequencies between the upstream transmitter and the local receiver.	Channel
rx_rmfifodatadeleted	Output	Synchronous to tx_clkout or coreclkout. tx_clkout for non-bonded modes. coreclkout for bonded modes.	Rate match FIFO deletion status indicator. ■ A high leve—the rate match pattern byte got deleted to compensate for the PPM difference in reference clock frequencies between the upstream transmitter and the local receiver.	Channel

Table 1–74. Stratix IV GX and GT ALTGX Megafunction Ports: Receiver Ports (Part 4 of 8)

Port Name	Input/ Output	Clock Domain	Description	Scope
`rx_rmfifofull`	Output	Synchronous to `tx_clkout` or `coreclkout`. `tx_clkout` for non-bonded modes. `coreclkout` for bonded modes.	Rate match FIFO full status indicator. ■ A high level indicates that the rate match FIFO is full. ■ Without byte serializer —driven a minimum of two recovered clock cycles. ■ With byte serializer—driven a minimum of three recovered clock cycles.	Channel
`rx_rmfifoempty`	Output	Synchronous to `tx_clkout` or `coreclkout`. `tx_clkout` for non-bonded modes. `coreclkout` for bonded modes.	Rate match FIFO empty status indicator. ■ A high level—the rate match FIFO is empty. ■ Without byte serializer—driven a minimum of two recovered clock cycles. ■ With byte serializer—driven a minimum of three recovered clock cycles.	Channel
8B/10B Decoder				
`rx_ctrldetect`	Output	Synchronous to `coreclkout` clock signal	Receiver control code indicator. ■ Available in configurations with 8B/10B decoder. ■ A high level—the associated received code group is a control (/Kx.y/) code group. ■ A low level—the associated received code group is a data (/Dx.y/) code group. ■ The width of this signal depends on the following channel width: Channel Width: 8—`rx_ctrldetect` = 1 16—`rx_ctrldetect` = 2 32—`rx_ctrldetect` = 4	Channel

Table 1–74. Stratix IV GX and GT ALTGX Megafunction Ports: Receiver Ports (Part 5 of 8)

Port Name	Input/ Output	Clock Domain	Description	Scope
rx_errdetect	Output	Synchronous to tx_clkout or coreclkout. tx_clkout for non-bonded modes. coreclkout for bonded modes.	8B/10B code group violation or disparity error indicator. ■ Available in configurations with 8B/10B decoder. ■ A high level—a code group violation or disparity error was detected on the associated received code group. Use with the rx_disperr signal to differentiate between a code group violation and/or a disparity error as follows: → [rx_errdetect: rx_disperr] → 2'b00—no error → 2'b10—code group violation → 2'b11—disparity error or both ■ Channel Width: 8—rx_errdetect = 1 16—rx_errdetect = 2 32—rx_errdetect = 4	Channel
rx_disperr	Output	Synchronous to tx_clkout or coreclkout. tx_clkout for non-bonded modes. coreclkout for bonded modes.	8B/10B disparity error indicator port. ■ Available in configurations with 8B/10B decoder. ■ A high level—a disparity error was detected on the associated received code group. ■ Channel Width: 8—rx_disperr = 1 16—rx_disperr = 2 32—rx_disperr = 4	Channel
rx_runningdisp	Output	Synchronous to tx_clkout or coreclkout. tx_clkout for non-bonded modes. coreclkout for bonded modes.	8B/10B running disparity indicator. ■ Available in configurations with the 8B/10B decoder. ■ A high level—the data on the rx_dataout port was received with a negative running disparity. ■ A low level—the data on the rx_dataout port was received with a positive running disparity. ■ Channel Width: 8—rx_runningdisp = 1 16—rx_runningdisp = 2 32—rx_runningdisp = 4	Channel

Table 1–74. Stratix IV GX and GT ALTGX Megafunction Ports: Receiver Ports (Part 6 of 8)

Port Name	Input/ Output	Clock Domain	Description	Scope
Byte Ordering Block				
`rx_enabyteord`	Input	Asynchronous signal	Enable byte ordering control. ■ Available in configurations with the byte ordering block enabled. The byte ordering block is rising-edge sensitive to this signal. ■ A low-to-high transition triggers the byte ordering block to restart the byte ordering operation.	Channel
`rx_ byteorderalignstatus`	Output	Synchronous to `tx_clkout` or `coreclkout`. `tx_clkout` for non-bonded modes. `coreclkout` for bonded modes.	Byte ordering status indicator. ■ Available in configurations with the byte ordering block enabled. ■ A high level—the byte ordering block has detected the programmed byte ordering pattern in the LSByte of the received data from the byte deserializer.	Channel
Receiver Phase Compensation FIFO				
`rx_dataout`	Output	Synchronous to `tx_clkout` or `coreclkout`. `tx_clkout` for non-bonded modes. `coreclkout` for bonded modes.	Parallel data output from the receiver to the FPGA fabric. ■ The bus width depends on the channel width multiplied by the number of channels per instance.	Channel
`rx_clkout`	Output	Clock signal	Recovered clock from the receiver channel. ■ Available only when the rate match FIFO is not used in the receiver datapath.	Channel
`rx_coreclk`	Input	Clock signal	Optional read clock port for the receiver phase compensation FIFO. ■ If not selected—the Quartus II software automatically selects `rx_clkout/tx_clkout/ coreclkout` as the read clock for the receiver phase compensation FIFO. ■ If selected—drive this port with a clock that has 0 PPM difference with respect to `rx_clkout/tx_clkout/ coreclkout`.	Channel

Table 1–74. Stratix IV GX and GT ALTGX Megafunction Ports: Receiver Ports (Part 7 of 8)

Port Name	Input/ Output	Clock Domain	Description	Scope
`rx_phase_comp_fifo_error`	Output	Synchronous to `tx_clkout` or `coreclkout`. `tx_clkout` for non-bonded modes. `coreclkout` for bonded modes.	Receiver phase compensation FIFO full or empty indicator. ■ A high level—the receiver phase compensation FIFO is either full or empty.	Channel
Receiver Physical Media Attachment (PMA)				
`rx_datain`	Input	N/A	Receiver serial data input port.	Channel
`rx_cruclk`	Input	Clock signal	Input reference clock for the receiver clock and data recovery.	Channel
`rx_pll_locked`	Output	Asynchronous signal	Receiver CDR lock-to-reference indicator. ■ A high level—the receiver CDR is locked to the input reference clock. ■ A low level—the receiver CDR is not locked to the input reference clock.	Channel
`rx_freqlocked`	Output	Asynchronous signal	Receiver CDR lock mode indicator. ■ A high level—the receiver CDR is in lock-to-data mode. ■ A low level—the receiver CDR is in lock-to-reference mode.	Channel
`rx_locktodata`	Input	Asynchronous signal	Receiver CDR lock-to-data mode control signal. ■ When asserted high—the receiver CDR is forced to lock-to-data mode. ■ When de-asserted low—the receiver CDR lock mode depends on the `rx_locktorefclk` signal level.	Channel
`rx_locktorefclk`	Input	Asynchronous signal	Receiver CDR lock-to-reference mode control signal. The `rx_locktorefclk` signal, along with the `rx_locktodata` signal, controls whether the receiver CDR is in automatic (0/0), lock-to-reference (0/1), or lock-to-data (1/x) mode.	Channel

Table 1–74. Stratix IV GX and GT ALTGX Megafunction Ports: Receiver Ports (Part 8 of 8)

Port Name	Input/ Output	Clock Domain	Description	Scope
`rx_signaldetect`	Output	Asynchronous signal	Signal threshold detect indicator. ■ Available in Basic functional mode when the 8B/10B Encoder/Decoder is selected. ■ Available in PCI Express (PIPE) mode. ■ A high level indicates that the signal present at the receiver input buffer is above the programmed signal detection threshold value. ■ If the electrical idle inference block is disabled in PCI Express (PIPE) mode, the `rx_signaldetect` signal is inverted and driven on the `pipeelecidle` port.	Channel
`rx_seriallpbken`	Input	Asynchronous signal	Serial loopback control port. ■ 0–normal datapath, no serial loopback ■ 1–serial loopback	Channel

Table 1–75 lists the ALTGX megafunction CMU ports.

Table 1–75. Stratix IV GX and GT ALTGX Megafunction Ports: CMU (Part 1 of 2)

Port Name	Input/ Output	Clock Domain	Description	Scope
`pll_inclk`	Input	Clock signal	Input reference clock for the CMU phase-locked loop.	Transceiver block
`pll_locked`	Output	Asynchronous signal	CMU PLL lock indicator. ■ A high level—the CMU PLL is locked to the input reference clock. ■ A low level—The CMU PLL is not locked to the input reference clock.	Transceiver block

Table 1–75. Stratix IV GX and GT ALTGX Megafunction Ports: CMU (Part 2 of 2)

Port Name	Input/ Output	Clock Domain	Description	Scope
`pll_powerdown`	Input	Asynchronous signal. For minimum pulse width requirements, refer the device *DC and Switching Characteristics* chapter.	CMU PLL power down. ■ Asserted high—the CMU PLL is powered down. ■ De-asserted low—the CMU PLL is active and locks to the input reference clock. Note: Asserting the `pll_powerdown` signal does not power down the REFCLK buffers.	Transceiver block
`coreclkout`	Output	Clock signal	FPGA fabric-transceiver interface clock. ■ Generated by the `CMU0` clock divider in the transceiver block in ×4 bonded channel configurations. ■ Generated by the `CMU0` clock divider in the master transceiver block in ×8 bonded channel configurations. ■ Not available in non-bonded channel configurations. ■ Use to clock the write port of the transmitter phase compensation FIFOs in all bonded channels and to clock parallel data `tx_datain` from the FPGA fabric into the transmitter phase compensation FIFO of all bonded channels. ■ Use to clock the read port of the receiver phase compensation FIFOs in all bonded channels with rate match FIFO enabled and to clock parallel data `rx_dataout` from the receiver phase compensation FIFOs of all bonded channels (with rate match FIFO enabled) into the FPGA fabric.	Transceiver block

Table 1–76 lists the ALTGX megafunction dynamic reconfiguration ports.

Table 1–76. Stratix IV GX and GT ALTGX Megafunction Ports: Dynamic Reconfiguration (Part 1 of 2)

Port Name	Input/ Output	Clock Domain	Description	Scope
`reconfig_clk`	Input	Clock signal	Dynamic reconfiguration clock. ■ Also used for offset cancellation in all modes except PCI Express (PIPE) mode. ■ If configured in **Transmitter only** mode—the frequency range is 2.5 MHz to 50 MHz. ■ If configured in **Receiver only** or **Receiver and Transceiver** mode—the frequency range of this clock is 37.5 MHz to 50 MHz.	

Table 1–76. Stratix IV GX and GT ALTGX Megafunction Ports: Dynamic Reconfiguration (Part 2 of 2)

Port Name	Input/ Output	Clock Domain	Description	Scope
`reconfig_togxb`	Input	Asynchronous signal	From the dynamic reconfiguration controller.	
`reconfig_fromgxb`	Output	Asynchronous signal	To the dynamic reconfiguration controller.	

Table 1–77 lists the ALTGX megafunction PCI Express (PIPE) interface ports.

Table 1–77. Stratix IV GX and GT ALTGX Megafunction Ports: PCI Express (PIPE) Interface (Part 1 of 4)

Port Name	Input/ Output	Clock Domain	Description	Scope
PCI Express (PIPE) Interface (Available only in PCI Express [PIPE] functional Mode)				
`powerdn`	Input	Asynchronous signal	PCI Express (PIPE) power state control. ■ Functionally equivalent to the `powerdown[1:0]` signal defined in the PCI Express (PIPE) specification revision 2.0. ■ The width of this signal is 2 bits and is encoded as follows: → 2'b00: P0—Normal Operation → 2'b01: P0s—Low Recovery Time Latency, Low Power State → 2'b10: P1—Longer Recovery Time Latency, Lower Power State → 2'b11: P2—Lowest Power State	Channel
`tx_forcedispcompliance`	Input	Asynchronous signal	Force 8B/10B encoder to encode with a negative running disparity. ■ Functionally equivalent to the `txcompliance` signal defined in PCI Express (PIPE) specification revision 2.0. ■ Must be asserted high only when transmitting the first byte of the PCI Express (PIPE) compliance pattern to force the 8B/10B encode with a negative running disparity as required by the PCI Express (PIPE) protocol.	Channel
`tx_forceelecidle`	Input	Asynchronous signal	Force transmitter buffer to PCI Express (PIPE) electrical idle signal levels. ■ Functionally equivalent to the `txelecidle` signal defined in the PCI Express (PIPE) specification revision 2.0. ■ Available in the Basic mode.	Channel
`rateswitch`	Input	Asynchronous signal	PCI Express (PIPE) rateswitch control. ■ 1'b0—Gen1 (2.5 Gbps) ■ 1'b1—Gen2 (5 Gbps)	

Table 1–77. Stratix IV GX and GT ALTGX Megafunction Ports: PCI Express (PIPE) Interface (Part 2 of 4)

Port Name	Input/ Output	Clock Domain	Description	Scope
`tx_pipemargin`	Input	Asynchronous signal	Transmitter differential output voltage level control. ■ Functionally equivalent to the `txmargin` signal defined in the PCI Express (PIPE) specification revision 2.0. ■ Available only in PCI Express (PIPE) Gen2 configuration. ■ The width of this signal is 3 bits and is decoded as follows: → 3'b000—Normal Operating Range → 3'b001—Full Swing = 800 - 1200 mV → 3'b010—TBD → 3'b011—TBD → 3'b100—If last value, full Swing = 200 to 400 mV → 3'b101—If last value, full Swing = 200 to 400 mV → 3'b110—If last value, full Swing = 200 to 400 mV → 3'b111—If last value, full Swing = 200 to 400 mV	
`tx_pipedeemph`	Input	Asynchronous signal	Transmitter buffer de-emphasis level control. ■ Functionally equivalent to the `txdeemph` signal defined in the PCI Express (PIPE) specification revision 2.0. ■ Available only in PCI Express (PIPE) Gen2 configuration. ■ 1'b0: -6 dB de-emphasis ■ 1'b1:-3.5 dB de-emphasis	
`pipe8b10binvpolarity`	Input	Asynchronous signal	PCI Express (PIPE) polarity inversion control. ■ Functionally equivalent to the `rxpolarity` signal defined in the PCI Express (PIPE) specification revision 2.0. ■ Available only in PCI Express (PIPE) mode. ■ When asserted high—the polarity of every bit of the 10-bit input data to the 8B/10B decoder gets inverted.	Channel

Table 1–77. Stratix IV GX and GT ALTGX Megafunction Ports: PCI Express (PIPE) Interface (Part 3 of 4)

Port Name	Input/ Output	Clock Domain	Description	Scope
`tx_detectrxloopback`	Input	Asynchronous signal	Receiver detect or PCI Express (PIPE) loopback control. ■ Functionally equivalent to the `txdetectrx/loopback` signal defined in the PCI Express (PIPE) specification revision 2.0. ■ When asserted high in the P1 power state with the `tx_forceelecidle` signal asserted—the transmitter buffer begins the receiver detection operation. After the receiver detect completion is indicated on the `pipephydonestatus` port, this signal must be de-asserted. ■ When asserted high in the P0 power state with the `tx_forceelecidle` signal de-asserted—the transceiver datapath gets dynamically configured to support parallel loopback, as described in "PCI Express (PIPE) Reverse Parallel Loopback" on page 1–192.	Channel
`pipestatus`	Output	N/A	PCI Express (PIPE) receiver status port. ■ Functionally equivalent to the `rxstatus[2:0]` signal defined in the PCI Express (PIPE) specification revision 2.0. ■ The width of this signal is 3 bits per channel. The encoding of receiver status on the `pipestatus` port is as follows: → 000—Received data OK → 001—1 skip added → 010—1 skip removed → 011—Receiver detected → 100—8B/10B decoder error → 101—Elastic buffer overflow → 110—Elastic buffer underflow → 111—Received disparity error	Channel
`pipephydonestatus`	Output	N/A	PHY function completion indicator. ■ Functionally equivalent to the `phystatus` signal defined in the PCI Express (PIPE) specification revision 2.0. ■ Assert this signal high for one parallel clock cycle to communicate completion of several PHY functions, such as power state transition, receiver detection, and signaling rate change between Gen1 (2.5 Gbps) and Gen2 (5 Gbps).	Channel

Table 1–77. Stratix IV GX and GT ALTGX Megafunction Ports: PCI Express (PIPE) Interface (Part 4 of 4)

Port Name	Input/ Output	Clock Domain	Description	Scope
`rx_pipedatavalid`	Output	N/A	Valid data and control on the `rx_dataout` and `rx_ctrldetect` ports indicator. ■ Functionally equivalent to the `rxvalid` signal defined in the PCI Express (PIPE) specification revision 2.0.	Channel
`pipeelecidle`	Output	Asynchronous signal	Electrical idle detected or inferred at the receiver indicator. ■ Functionally equivalent to the `rxelecidle` signal defined in the PCI Express (PIPE) specification revision 2.0. ■ If the electrical idle inference block is enabled—it drives this signal high when it infers an electrical idle condition, as described in "Electrical Idle Inference" on page 1–136. Otherwise, it drives this signal low. ■ If the electrical idle inference block is disabled—the `rx_signaldetect` signal from the signal detect circuitry in the receiver buffer is inverted and driven on this port.	Channel

Table 1–78 lists the ALTGX megafunction reset and power down ports.

Table 1–78. Stratix IV GX and GT ALTGX Megafunction Ports: Reset and Power Down (Part 1 of 2)

Port Name	Input/ Output	Clock Domain	Description	Scope
`gxb_powerdown`	Input	Asynchronous signal. For minimum pulse width requirements, refer the device *DC and Switching Characteristics* chapter.	Transceiver block power down. ■ When asserted high—all digital and analog circuitry within the PCS, PMA, CMU channels, and the CCU of the transceiver block, is powered down. ■ Asserting the `gxb_powerdown` signal does not power down the REFCLK buffers.	Transceiver block
`rx_digitalreset`	Input	Asynchronous signal. Minimum pulse width is two parallel clock cycles.	Receiver PCS reset. ■ When asserted high—the receiver PCS blocks are reset. Refer to *Reset Control and Power Down.*	Channel

Table 1–78. Stratix IV GX and GT ALTGX Megafunction Ports: Reset and Power Down (Part 2 of 2)

Port Name	Input/ Output	Clock Domain	Description	Scope
`rx_analogreset`	Input	Asynchronous signal. Minimum pulse width is two parallel clock cycles.	Receiver PMA reset. ■ When asserted high—analog circuitry within the receiver PMA gets reset. Refer to *Reset Control and Power Down.*	Channel
`tx_digitalreset`	Input	Asynchronous signal. Minimum pulse width is two parallel clock cycles.	Transmitter PCS reset. ■ When asserted high, the transmitter PCS blocks are reset. Refer to *Reset Control and Power Down.*	Channel

Table 1–79 lists the ALTGX megafunction calibration block ports.

Table 1–79. Stratix IV GX and GT ALTGX Megafunction Ports: Calibration Block

Port Name	Input/ Output	Clock Domain	Description	Scope
`cal_blk_clk`	Input	Clock signal	Clock for transceiver calibration blocks.	Device
`cal_blk_powerdown`	Input	Clock signal	Calibration block power down control.	Device

Reference Information

Use the links listed in Table 1–80 for more information about some useful reference terms used in this chapter.

Table 1–80. Reference Information (Part 1 of 3)

Terms Used in this Chapter	Useful Reference Points
(OIF) CEI PHY Interface Mode	page 1–179
8B/10B Decoder	page 1–86
8B/10B Encoder	page 1–19
AEQ	page 1–46
Auxiliary Transmit (ATX) PLL Block	page 1–193
Basic (PMA Direct) Functional Mode	page 1–185
Basic Functional Mode	page 1–108
Built-In Self Test Modes	page 1–204
Byte Ordering Block	page 1–92
Byte Serializer	page 1–90
Calibration Blocks	page 1–199
Clock and Data Recovery Unit (CDR)	page 1–50
CMU Channel Architecture	page 1–97
CMU0 PLL	page 1–99
CMU1 PLL	page 1–99

Table 1–80. Reference Information (Part 2 of 3)

Terms Used in this Chapter	Useful Reference Points
CPRI and OBSAI	page 1–121
Deserializer	page 1–55
Deskew FIFO	page 1–72
Deterministic Latency Mode	page 1–119
EyeQ	page 1–48
GIGE Mode	page 1–161
Lock-to-Data (LTD)	page 1–51
Lock-to-Reference (LTR)	page 1–50
Low Latency PCS Datapath	page 1–109
Offset Cancellation in the Receiver Buffer and Receiver CDR	page 1–54
Parallel loopback	page 1–189
PCI Express (PIPE) Clock Switch Circuitry	page 1–52
PCI Express (PIPE) Mode	page 1–124
PCI EXpress (PIPE) Reverse Parallel Loopback	page 1–192
Programmable Common Mode Voltage	page 1–36
Programmable Equalization and DC Gain	page 1–45
Programmable Pre-Emphasis	page 1–32
Programmable Differential On-Chip Termination	page 1–38
Rate Match (Clock Rate Compensation) FIFO	page 1–74
Receiver Bit Reversal	page 1–70
Receiver Input Buffer	page 1–36
Receiver Phase Compensation FIFO	page 1–95
Receiver Polarity Inversion	page 1–68
Reverse Serial Loopback	page 1–191
Reverse serial Pre-CDR Loopback	page 1–191
SATA and SAS options	page 1–118
SDI Mode	page 1–175
Serial Loopback	page 1–188
Serial RapidIO Mode	page 1–180
Signal Threshold Detection Circuitry	page 1–45
SONET/SDH Mode	page 1–169
Transceiver Block Architecture	page 1–11
Transceiver Channel Locations	page 1–4
Transceiver Port Lists	page 1–207
Transmitter Bit Reversal	page 1–27
Transmitter Local Clock Divider Block	page 1–35
Transmitter Output Buffer	page 1–30
Transmitter Polarity Inversion	page 1–25
TX Phase Compensation FIFO	page 1–14

Table 1–80. Reference Information (Part 3 of 3)

Terms Used in this Chapter	Useful Reference Points
Word Aligner	page 1–56
XAUI Mode	page 1–150

Document Revision History

Table 1–81 shows the revision history for this chapter.

Table 1–81. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
November 2009, v4.0	■ Added "Adaptive Equalization (AEQ)", "EyeQ", "SATA and SAS Options", "Deterministic Latency Mode", "CPRI and OBSAI", and "Reference Information" sections. ■ Added Figure 1–91, Figure 1–93, Figure 1–95, and Figure 1–97. ■ Added Stratix IV GT device information. ■ Updated Figures. ■ Updated Tables. ■ Re-organized chapter information. ■ Minor text edits.	—
June 2009, v3.1	■ Updated the "Introduction", "Auxiliary Transmit (ATX) PLL Block", "Rate Match (Clock Rate Compensation) FIFO", "Transmitter Buffer Electrical Idle", "PCI Express (PIPE) Gen2 (5 Gbps) Support", "Reverse Serial Loopback", and "Reverse Serial Pre-CDR Loopback" sections. ■ Added new "PCI Express Electrical Gold Test with Compliance Base Board (CBB)", "Recommendation When Using the Electrical Idle Inference Block". and "Rate Match FIFO in Serial RapidIO Mode" sections. ■ Added new Figure 1–165. ■ Updated Table 1–2, Table 1–17, Table 1–32, Table 1–34, and Table 1–52. ■ Updated Figure 1–7, and Figure 1–165 through Figure 1–168. ■ Minor text edits.	—
March 2009, v3.0	■ Reorganized sections. ■ Added the section "Link Coupling". ■ Updated the section "DC-Coupled Links".	—
November 2008 v2.0	Added Offset Cancellation in the Receiver Buffer and Receiver CDR to the Receiver Channel Datapath section	—
May 2008 v1.0	Initial release.	—

2. Stratix IV Transceiver Clocking

SIV52002-3.0

This chapter provides detailed information about the Stratix® IV transceiver clocking architecture. For this chapter, the term "Stratix IV devices" includes both Stratix IV GX and GT devices. Similarly, the term "Stratix IV transceivers" includes both Stratix IV GX and GT transceivers.

The clocking architecture chapter is divided into three main sections:

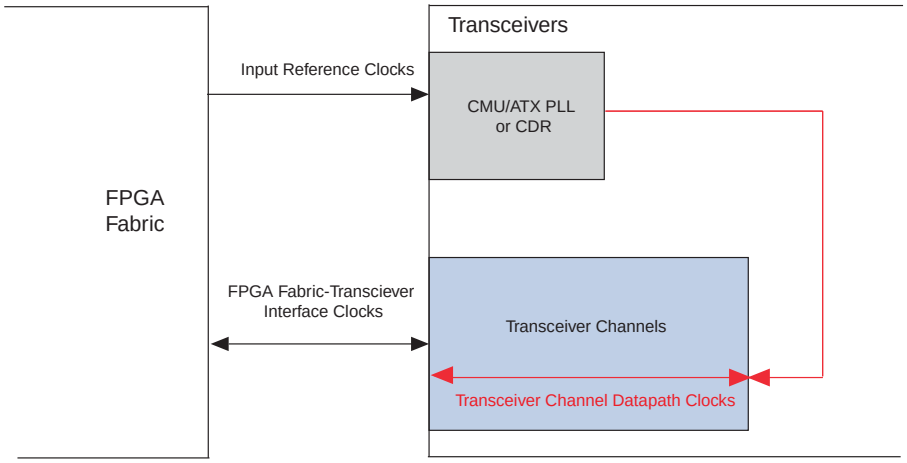
- "Input Reference Clocking" on page 2–2—describes how the reference clock is provided to the clock multiplier unit (CMU)/auxiliary transmit phase-locked loop (ATX PLL) to generate the clocks required for transceiver operation.
- "Transceiver Channel Datapath Clocking" on page 2–20—describes the clocking architecture internal to the transceiver block.
- "FPGA Fabric-Transceiver Interface Clocking" on page 2–49—describes the clocking options available when interfacing the transceiver with the FPGA fabric.

Other sections in this chapter include:

- "FPGA Fabric PLLs-Transceiver PLLs Cascading" on page 2–9
- "Using the CMU/ATX PLL for Clocking User Logic in the FPGA Fabric" on page 2–70
- "Configuration Examples" on page 2–72

Figure 2–1 shows an overview of the clocking architecture.

Figure 2–1. Clocking Architecture Overview



Glossary of Terms

Table 2–1 lists the terms used in the chapter.

Table 2–1. Glossary of Terms Used in this chapter

Convention	Description
ATX PLL	Auxiliary transmit PLL block. For more information, refer to the "Auxiliary Transmit (ATX) PLL Block" section in the *Stratix IV Transceiver Architecture* chapter.
CDR	Clock data recovery block. For more information, refer to the "Clock and Data Recovery Unit" section in the *Stratix IV Transceiver Architecture* chapter.
CMU	Clock multiplier unit. For more information, refer to "CMU Channel Architecture" section in the *Stratix IV Transceiver Architecture* chapter.
ITB lines	The Inter-Transceiver block (ITB) clock lines provide an input reference clock path from the `refclk` pins of one transceiver block CMU PLLs and receiver CDRs of other transceiver blocks. They also provide input reference clock to ATX PLLs. For more information, refer to "Inter-Transceiver Block (ITB) Clock Lines" on page 2–8.

Input Reference Clocking

Each transceiver block has:

- Two clock multiplier unit channels (`CMU0_Channel` and `CMU1_Channel`)

  You can configure each as either a CMU to generate transceiver clocks or as a PMA-Only channel.

  For more information, refer to the "CMU Channel Architecture" section in the *Stratix IV Transceiver Architecture* chapter.

- Four regular channels

  When the CMU channel is configured as a CMU, the CMU PLL synthesizes the input reference clock to generate the high-speed serial transceiver clock. When the CMU channel is configured as a **Receiver Only** or **Receiver and Transmitter** channel, the CMU PLL acts as a CDR and uses the input reference clock as a training clock when it is in lock-to-reference (LTR) mode. Each of the four regular channels also has a receiver CDR that uses the input reference clock as a training clock when it is in LTR mode.

  Each Stratix IV device also has ATX PLLs that you can use in addition to the CMU PLLs to generate the high-speed serial transceiver clock. The ATX PLLs also need an input reference clock for operation. 6G ATX PLLs are available in both Stratix IV GX and Stratix IV GT devices. 10G ATX PLLs are available only in Stratix IV GT devices.

  For more information, refer to the "Auxiliary Transmit (ATX) PLL Block" and the "Transmitter Channel Datapath" sections in the *Stratix IV Transceiver Architecture* chapter.

Input Reference Clock Source

Receiver clock data recoveries (CDRs), CMU PLLs (when the CMU channel is configured as a CMU) and ATX PLLs can derive the input reference clock from one of the sources shown in Table 2–2.

Table 2–2. Input Reference Clock Source

Index	Clock Source	CMU PLL	6G ATX PLL	10G ATX PLL	CDR	Jitter Performance *(5)*
1	`refclk0` and `refclk1` pins of the same transceiver block	Yes	No *(1)*	No *(1)*	Yes	1
2	`refclk0` and `refclk1` pins of other transceiver blocks on the same side of the device using the inter-transceiver block (ITB) clock lines *(2)*	Yes	Yes	Yes	Yes	2 *(4)*
3	Clock output from the left and right PLLs in the FPGA fabric with voltage controlled oscillator (VCO) bypass mode *(3)*	Yes	Yes	No	Yes	3
4	Clock output from the left and right PLLs in the FPGA fabric	Yes	Yes	No	Yes	4
5	Dedicated CLK input pins on the FPGA global clock network	Yes	Yes	No	Yes	4

Notes to Table 2–2:

(1) ATX PLLs do not have dedicated `refclk` pins.

(2) For more information, refer to "Inter-Transceiver Block (ITB) Clock Lines" on page 2–8.

(3) For more information, refer to "Configuration Examples" on page 2–72.

(4) For better jitter performance, Altera strongly recommends using the `refclk0` and `refclk1` pins of the transceiver block located immediately below the ATX PLL.

(5) Lowest number indicates best jitter performance.

When a CMU channel is configured as a channel, its CMU PLL acts as a receiver CDR and can derive the input reference clock sources 2 through 5 listed in the Table 2–2. You can also use the `refclk` pin of the other CMU channel within the transceiver block as a clock source as long as the other CMU channel is not configured as a **Receiver only** or **Receiver and Transmitter** channel. For example, the `CMU0` PLL can derive its input reference clock from the `refclk1` pin if the `CMU1` channel is not configured as a **Receiver only** or **Receiver and Transmitter** channel.

☞ When a CMU channel is configured as a channel, its `refclk` pin is used to receive serial input data. As a result, the `refclk` pin is not available to provide the input reference clock.

Table 2–3 lists the input reference clock frequencies allowed for the 10G ATX PLL.

Table 2–3. Input Reference Clock Frequencies for the 10G ATX PLL Clock

Data Rate	Allowed Divider Values	Reference Clock Frequency
9.9 Gbps to 11.3 Gbps	M = 16, N = 1	281.25 to 322 MHz
	M = 16, N = 2	562.5 to 706.25 MHz

Figure 2–2 shows the input reference clock sources for CMU PLLs and receiver CDRs within a transceiver block. One global clock line is available for each CMU PLL and receiver CDR in a transceiver block. This allows each CMU PLL and receiver CDR to derive its input reference clock from a separate FPGA CLK input pin.

Figure 2–2. Input Reference Clock Sources in a Transceiver Block

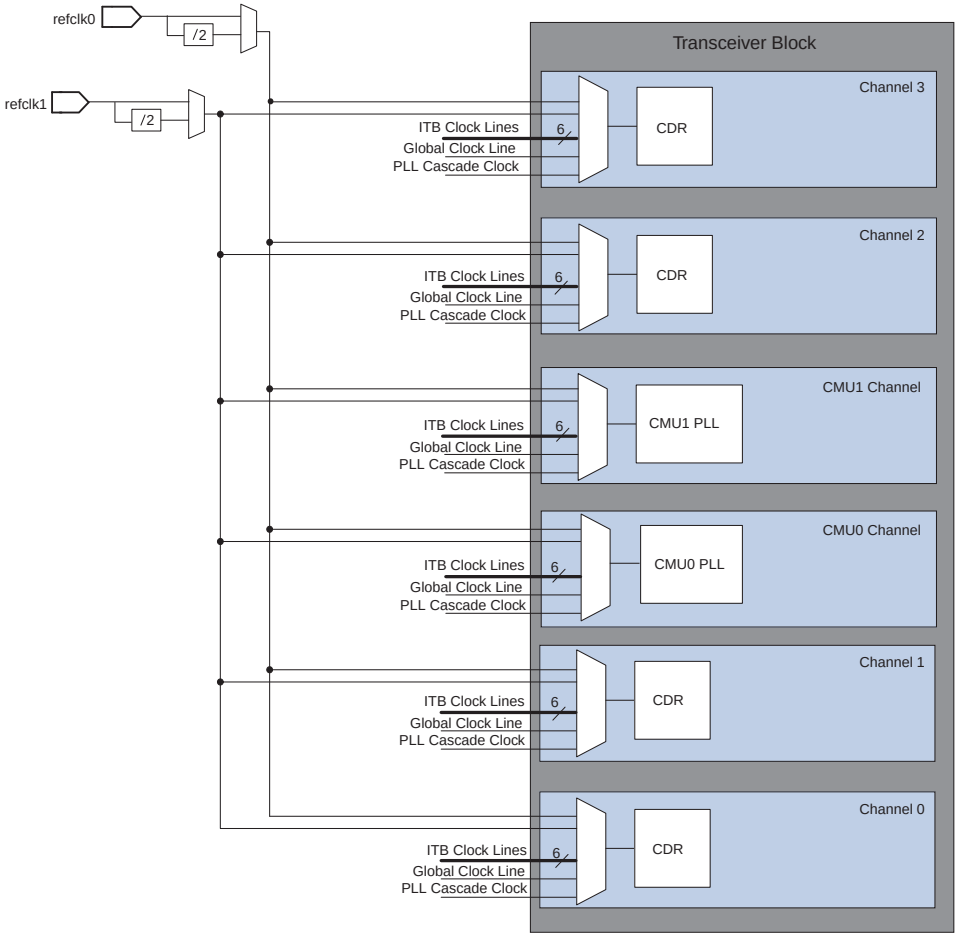


Figure 2–3 shows the input reference clock sources for CMU PLLs, ATX PLLs, and receiver CDRs in four transceiver blocks on the right side of the EP4SGX530F45 device. In this figure, the input reference clock sources for four transceiver blocks are located only on the right side of the device but the EP4SGX530NF45 device has similar input reference clock resources available for the four transceiver blocks located on the left side of the device as well.

Figure 2–3 also shows the ITB clock lines on the right side of the device. The number of ITB clock lines available in any Stratix IV GX device is equal to the number of `refclk` pins available in that device.

Figure 2–3. Input Reference Clock Sources Across Transceiver Blocks

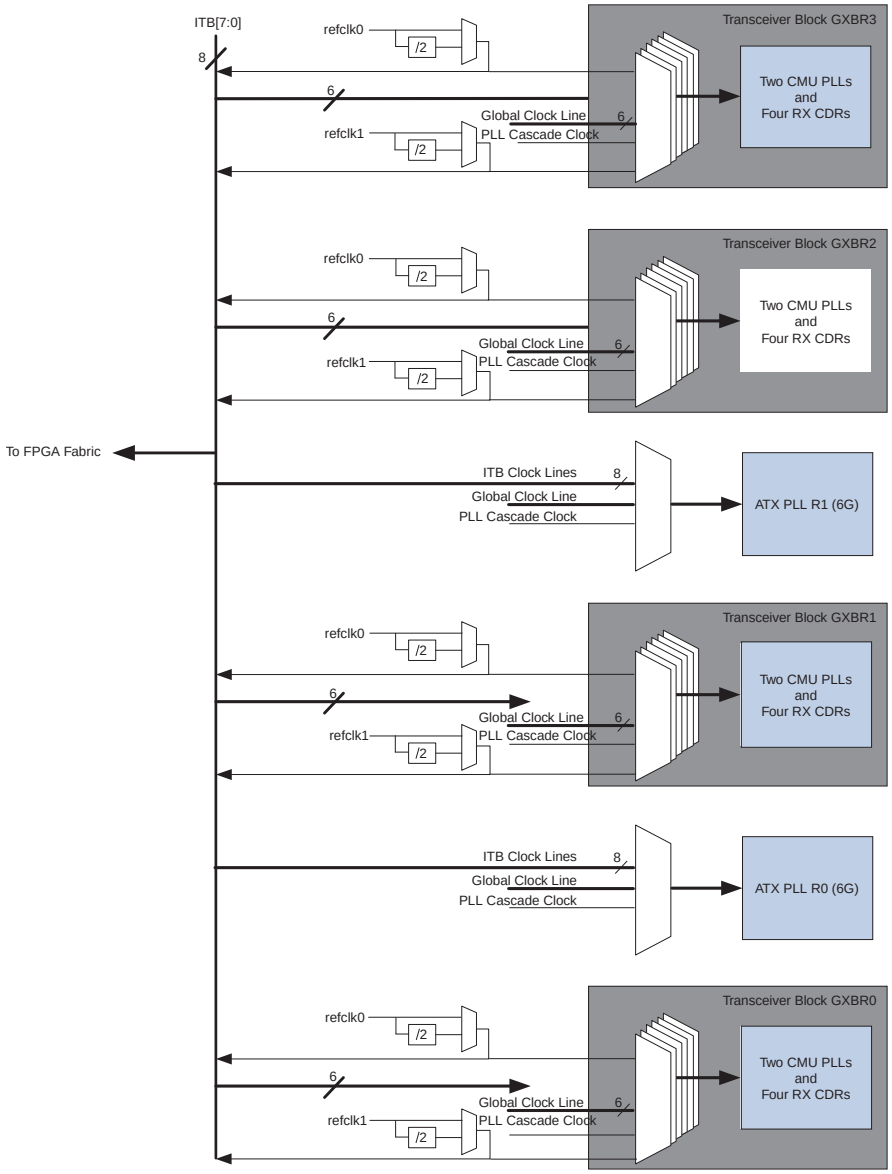
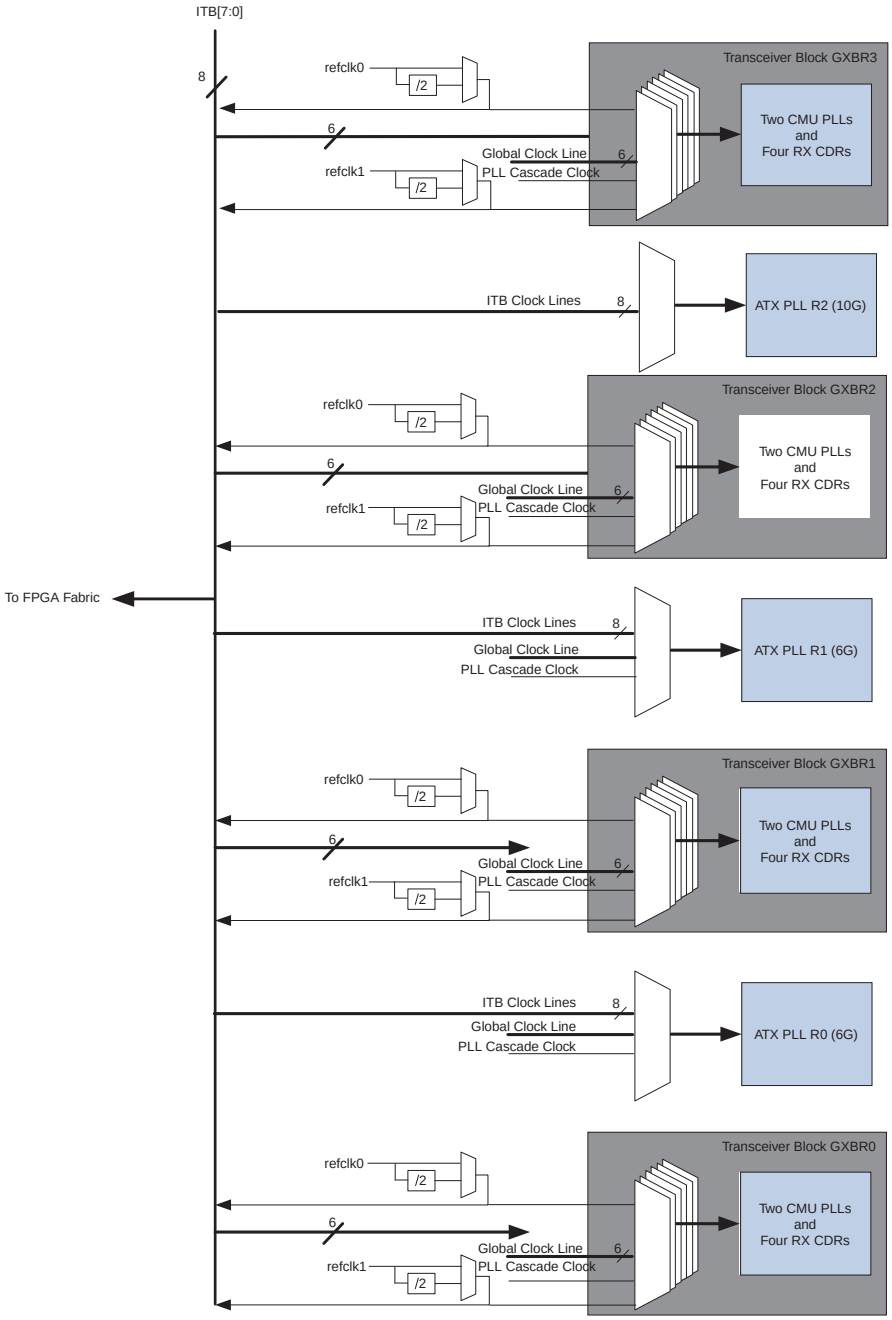


Figure 2–4 shows the input reference clock sources for CMU PLLs, ATX PLLs, and receiver CDRs in four transceiver blocks on the right side of the EP4S100G5F45 device. In this figure, the input reference clock sources for four transceiver blocks are located only on the right side of the EP4S100G5F45 device but the device has similar input reference clock resources available for the four transceiver blocks located on the left side of the device as well.

Figure 2–4 also shows the ITB clock lines on the right side of the EP4S100G5F45 device. The number of ITB clock lines available in any Stratix IV GT device is equal to the number of `refclk` pins available in that device.

Figure 2–4. Input Reference Clock Sources Across Transceiver Blocks for Stratix IV GT Devices



refclk0 and refclk1 Pins

Each transceiver block has two dedicated `refclk` pins that you can use to drive the CMU PLL, receiver CDR, or both, input reference clocks. Each of the two CMU PLLs and four receiver CDRs within a transceiver block can derive its input reference clock from either the `refclk0` or `refclk1` pin.

The `refclk` pins provide the cleanest input reference clock path to the CMU/ATX PLLs when compared with other input reference clock sources. Altera recommends using the `refclk` pins to drive the CMU PLL input reference clock for improved transmitter output jitter performance.

Table 2–4 lists the electrical specifications for the input reference clock signal driven on the `refclk` pins.

For specifications regarding the input frequency supported by the `refclk` pins, refer to the *DC and Switching Characteristics* chapter.

Table 2–4. Electrical Specifications for the Input Reference Clock

Protocol	I/O Standard	Coupling	Termination
■ GIGE ■ XAUI ■ Serial RapidIO ■ SONET/SDH ■ SDI ■ (OIF) CEI PHY Interface ■ Basic	■ 1.2-V PCML ■ 1.5-V PCML ■ 2.5-V PCML ■ Differential LVPECL ■ LVDS	AC	On-chip
PCI Express (PIPE)	■ 1.2-V PCML ■ 1.5-V PCML ■ 2.5-V PCML ■ Differential LVPECL ■ LVDS	AC	On-chip
	■ HCSL *(1)*	DC	Off-chip *(2)*

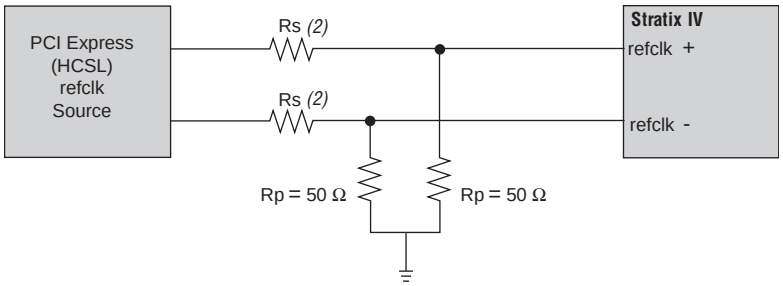
Notes to Table 2–4:

(1) In PCI Express (PIPE) mode, you have the option of selecting the HCSL standard for the reference clock if compliance to the PCI Express protocol is required. When configured as HCSL, the Quartus® II software automatically selects the **DC coupling with external termination** option for the `refclk` pins signal.

(2) For an example termination scheme, refer to Figure 2–5 on page 2–8.

Figure 2–5 shows an example termination scheme for a reference clock signal when configured as HCSL.

Figure 2–5. Termination Scheme for a Reference Clock Signal When Configured as HCSL *(Note 1)*



Notes to Figure 2–5:

(1) No biasing is required if the reference clock signals are generated from a clock source that conforms to the PCI Express (PIPE) specification.

(2) Select resistor values as recommended by the PCI Express (PIPE) clock source vendor.

Inter-Transceiver Block (ITB) Clock Lines

The `refclk0` and `refclk1` pins of other transceiver blocks using the ITB clock lines provide an input reference clock path from the `refclk` pins of one transceiver block to the CMU PLLs and receiver CDRs of the other transceiver blocks. In designs that have channels located in different transceiver blocks, the ITB clock lines eliminate the need to connect the on-board reference clock crystal oscillator to the `refclk` pin of each transceiver block. The ITB clock lines also drive the clock signal on the `refclk` pins to the clock logic in the FPGA fabric.

The ITB clock lines also provide an input reference clock path from the `refclk` pins of any transceiver block to the ATX PLLs located on the same side of the device.

Each `refclk` pin drives one ITB clock line for a total of up to eight ITB clock lines on each of the right and left sides of the device, as shown in Figure 2–3 on page 2–5.

The ITB clock lines provide input reference clock paths from the `refclk` pins of one transceiver block to the CMU PLLs and receiver CDRs of other transceiver blocks located on the same side of the device.

Dedicated CLK Input Pins on the FPGA Global Clock Network

Stratix IV devices provide up to 8 differential clock input pins located in non-transceiver I/O banks that you can use to provide up to eight input reference clocks to the transceiver blocks. The Quartus II software automatically chooses the global clock (GCLK) network to route the input reference clock signal from the CLK pins to the transceiver blocks.

For more information, refer to the "Dedicated Clock Input Pins" section in the *Clock Networks and PLLs in Stratix IV Devices* chapter.

One GCLK resource is available for each CMU PLL, 6G ATX PLL, and receiver CDR. This allows each CMU PLL, 6G ATX PLL, and receiver CDR to derive its input reference clock from a separate FPGA CLK input pin.

Clock Output from Left and Right PLLs in the FPGA Fabric

You can use the synthesized clock output from one of the left or right PLLs to provide the input reference clock to the CMU PLLs, 6G ATX PLLs, and receiver CDRs. Stratix IV devices provide a dedicated clock path from the left PLLs (`PLL_L1`, `PLL_L2`, `PLL_L3`, and `PLL_L4`) in the FPGA fabric to the PLL cascade network located on the left side of the device. Stratix IV devices also provide a dedicated clock path from the right PLLs (`PLL_R1`, `PLL_R2`, `PLL_R3`, and `PLL_R4`) in the FPGA fabric to the PLL cascade network located on the right side of the device. The additional clock multiplication factors available in the left and right PLLs allow more options for on-board crystal oscillator frequencies.

FPGA Fabric PLLs-Transceiver PLLs Cascading

The CMU PLL synthesizes the input reference clock to generate the high-speed serial clock used in the transmitter PMA. The receiver CDR synthesizes the input reference clock in lock-to-reference (LTR) mode to generate the high-speed serial clock.

This high-speed serial clock output from the CMU PLL and the receiver CDR runs at a frequency that is half the configured data rate. The CMU PLLs and receiver CDRs only support multiplication factors (M) of 2, 4, 5, 8, 10, 16, 20, and 25. If you use an on-board crystal oscillator to provide the input reference clock through the dedicated `refclk` pins or ITB lines, the allowed crystal frequencies are limited by the CMU PLL and the receiver CDR multiplication factors. The input reference clock frequencies are also limited by the allowed phase frequency detector (PFD) frequency range.

Example 1: Channel Configuration with 4-Gbps Data Rate

Consider a channel configured for 4-Gbps data rate. The high-speed serial clock output from the CMU PLL and the receiver CDR must run at 2 Gbps. Table 2–5 lists the allowed input reference clock frequencies for Example 1.

Table 2–5. Allowed Input Reference Clock Frequency for Example 1

Multiplication Factor (M)	On-Board Crystal Reference Clock Frequency (MHz)		Allowed
	With /N = 1	With /N = 2	
2	1000	2000	No. Violates the PFD frequency limit.
4	500	1000	No. Violates the PFD frequency limit.
5	400	800	Yes but only for /N = 1.
8	250	500	Yes
10	200	400	Yes
16	125	250	Yes
20	100	200	Yes
25	80	160	Yes

For a 4-Gbps data rate, the Quartus II software only allows an input reference clock frequency of 80, 100, 125, 160, 200, 250, 400, and 500 MHz. To overcome this limitation, Stratix IV devices allow the synthesized clock output from the left and right PLLs in the FPGA fabric to drive the CMU PLL and receiver CDR input reference clock. The additional clock multiplication factors available in the left and right PLLs allow more options for on-board crystal oscillator frequencies.

Dedicated Left and Right PLL Cascade Network

Stratix IV devices have a dedicated PLL cascade network on the left and right side of the device that connects to the input reference clock selection multiplexer of the CMU PLLs, 6G ATX PLLs, and receiver CDRs on the left and right side of the device, respectively.

The dedicated PLL cascade networks are segmented by bidirectional tri-state buffers located along the clock line. Segmentation of the dedicated PLL cascade network allows two or more left and right PLLs to drive the cascade clock line simultaneously.

Because the number of left and right PLLs and transceiver blocks vary from device to device, the capability of cascading a left and right PLL to the CMU PLLs, 6G ATX PLLs, and receiver CDRs also varies from device to device.

The following sections describe the Stratix IV GX and GT FPGA fabric-Transceiver PLLs cascading for the various device packages.

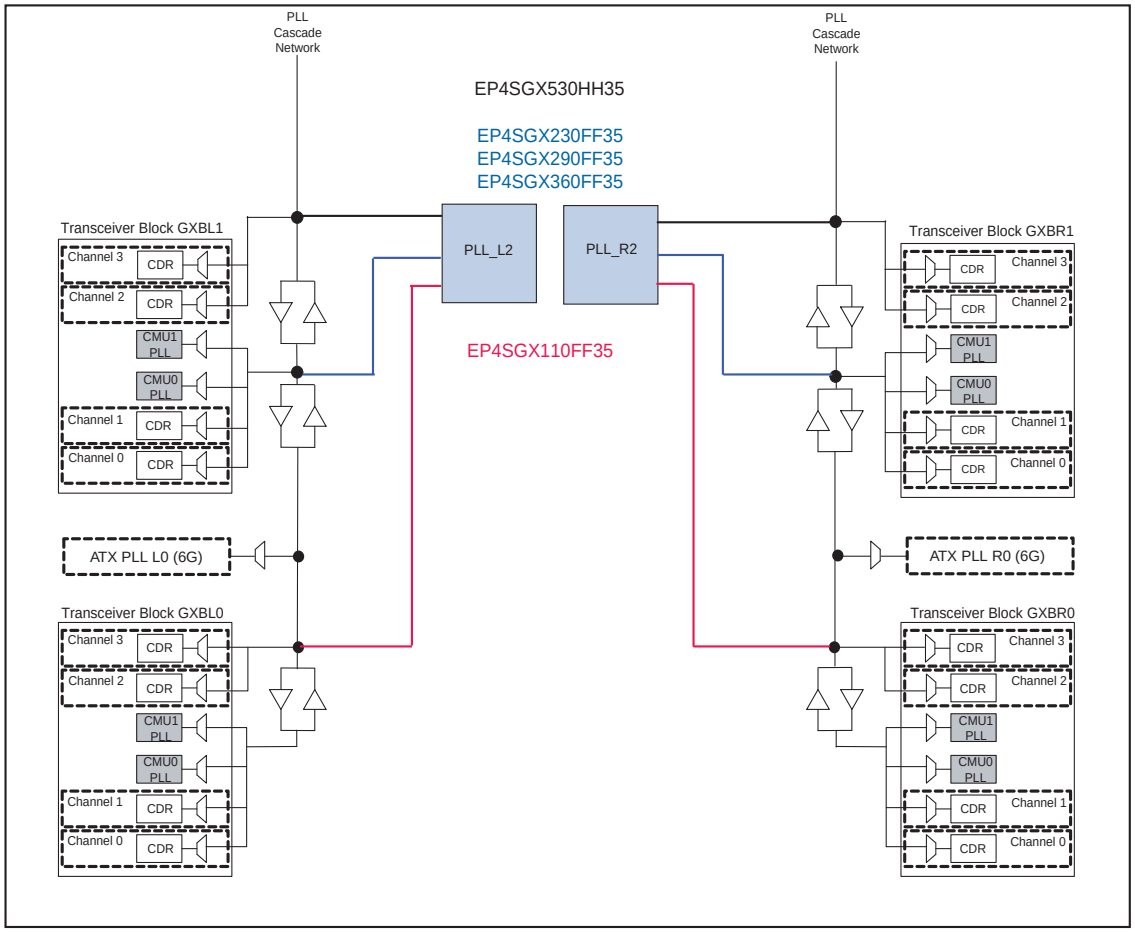
FPGA Fabric PLLs-Transceiver PLLs Cascading in the 780-Pin Package

Stratix IV GX devices in 780-pin packages do not support FPGA fabric PLLs-transceiver PLLs cascading.

FPGA Fabric PLLs-Transceiver PLLs Cascading in the 1152-Pin Package

Figure 2–6 shows the FPGA fabric PLLs-Transceiver PLLs cascading options allowed in the EP4SGX110FF35 device (red), the EP4SGX230FF35, EP4SGX290FF35 and EP4SGX360FF35 devices (blue), and the EP4SGX530HH35 device (black).

Figure 2–6. FPGA Fabric PLLs-Transceiver PLLs Cascading Options Allowed for 1152-Pin Package Devices

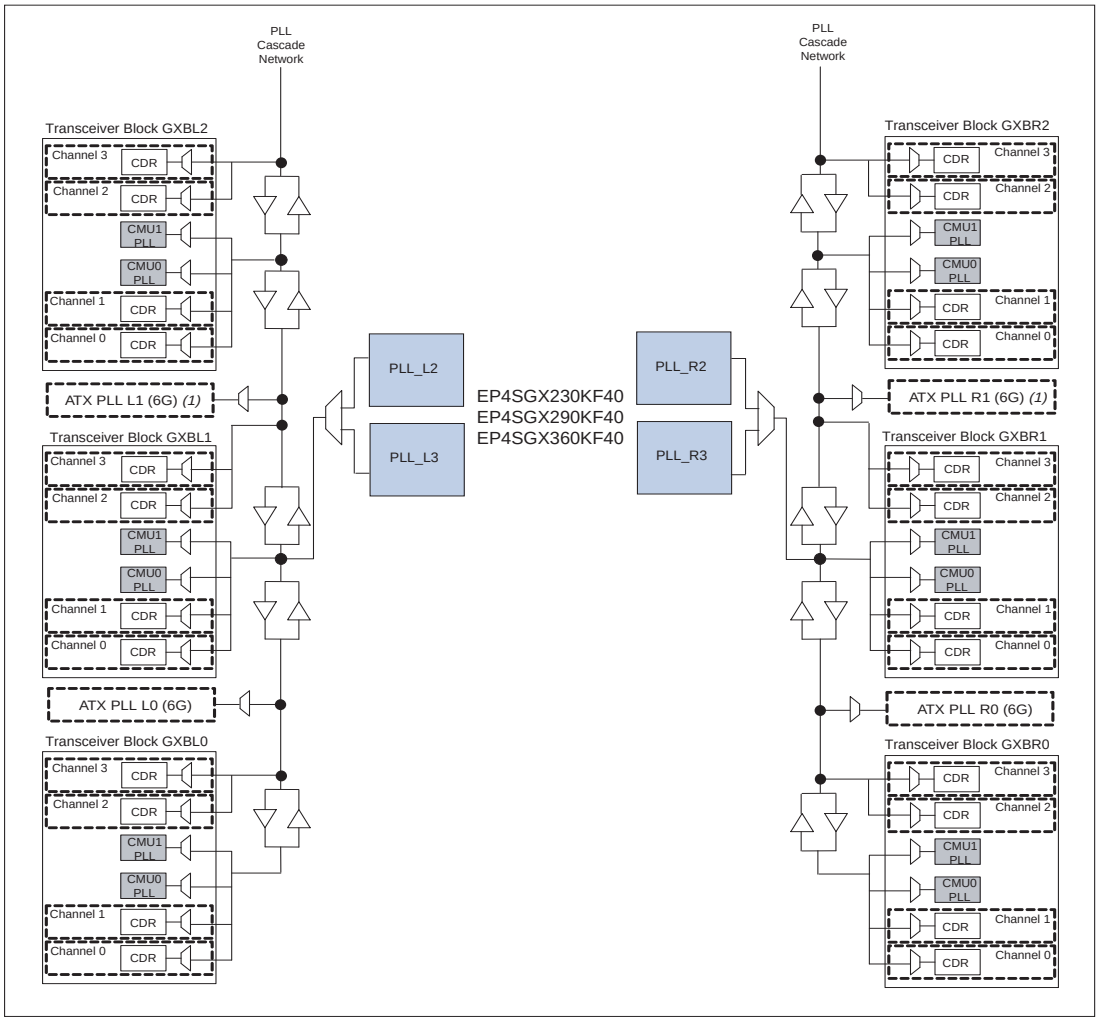


FPGA Fabric PLLs-Transceiver PLLs Cascading in the 1517-Pin Package

Figure 2–7 shows the FPGA fabric PLLs-Transceiver PLLs cascading options allowed in the EP4SGX230KF40, EP4SGX290KF40, EP4SGX360KF40, and EP4SGX530KF40 devices.

For Stratix IV GT devices, FPGA fabric PLLs-Transceiver PLLs cascading is not supported for the 10G ATX PLLs. For the EP4S40G2KF40, EP4S40G5KF40, EP4S100G2KF40, and EP4S100G5KF40 devices, FPGA fabric PLLs-Transceiver PLLs cascading for the 6G ATX PLLs and CMU PLLs is the same as the Stratix IV GX devices in the 1517-pin package.

Figure 2–7. FPGA Fabric PLLs-Transceiver PLLs Cascading Options Allowed in the 1517-Pin Package Devices



Note to Figure 2–7:

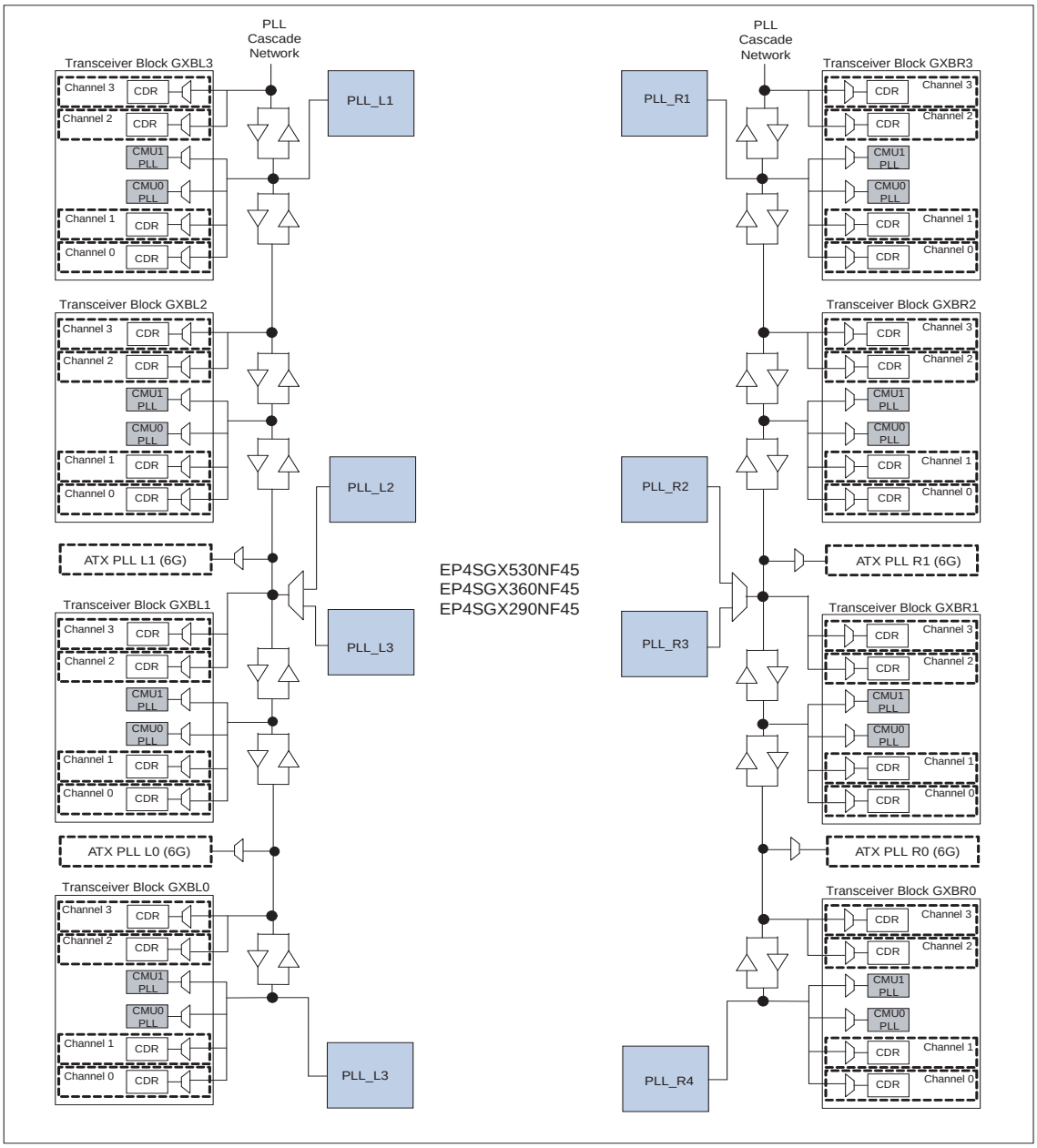
(1) ATX PLL L1 and ATX PLL R1 are not present in the EP4SGX530KF40 device.

FPGA Fabric PLLs-Transceiver PLLs Cascading in the 1932-Pin Package

Figure 2–8 shows the FPGA fabric PLLs-Transceiver PLLs cascading options allowed in the EP4SGX530NF45, EP4SGX360NF45, and EP4SGX290NF45 devices.

For Stratix IV GT devices, FPGA fabric PLLs-Transceiver PLLs cascading is not supported for the 10G ATX PLLs. For the EP4S100G3NF45, EP4S100G4N45, and EP4S100G5NF45 devices, FPGA fabric PLLs-Transceiver PLLs cascading for the 6G ATX PLLs and CMU PLLs is the same as the Stratix IV GX devices in the 1932-pin package.

Figure 2–8. FPGA Fabric PLLs-Transceiver PLLs Cascading Options Allowed in the 1932-Pin Package Device



FPGA Fabric PLLs-Transceiver PLLs Cascading Rules

You can only cascade the left PLLs (PLL_L1, PLL_L2, PLL_L3, and PLL_L4) to the transceiver blocks located on the left side of the device. Similarly, you can only cascade the right PLLs (PLL_R1, PLL_R2, PLL_R3, and PLL_R4) to the transceiver blocks located on the right side of the device.

The PLL cascade networks are single clock lines segmented by bidirectional tri-state buffers located along the clock line. Segmentation of the PLL cascade network allows two left and right PLLs to drive the cascade clock line simultaneously and provides the input reference clock to the CMU PLLs and receiver CDRs in different transceiver blocks. When cascading two or more FPGA fabric PLLs to the CMU PLLs and receiver CDRs, there must be no crossover in the cascaded clock paths on the PLL cascade network (Figure 2–9).

For better noise rejection, ensure the bandwidth setting of the FPGA fabric PLL (the upstream PLL) is lower than the transceiver PLL (the downstream PLL).

Example 2: Design Target—EP4SGX530NF45 Device

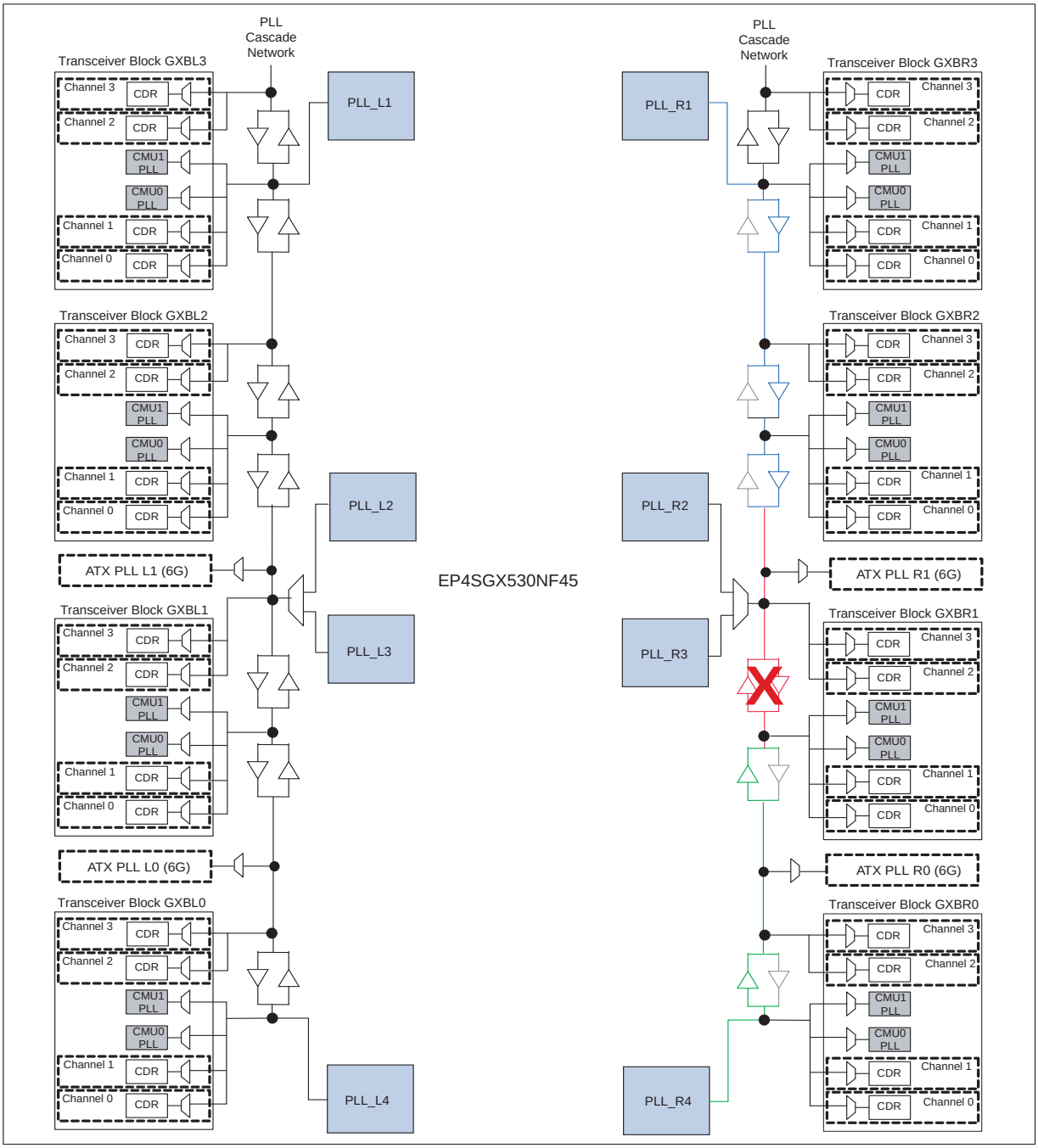
If your design is targeted for a EP4SGX530NF45 device, it requires providing input reference clocks to the following CMU PLLs and receiver CDRs from two right PLLs in the FPGA fabric:

- CMU0 PLL in Transceiver Block GXBR1
- Receiver CDRs in channel 2 and channel 3 in Transceiver Block GXBR1

Case 1: use PLL_R4 to provide the input reference clock to the receiver CDRs in channel 2 and channel 3 (shown in GREEN) and use PLL_R1 to provide the input reference clock to the CMU0 PLL (shown in BLUE) in transceiver block GXBR1.

Figure 2–9 shows that this FPGA fabric-Transceiver PLL cascading configuration is illegal due to crossover (shown in RED) of the cascade clock paths on the PLL cascade network.

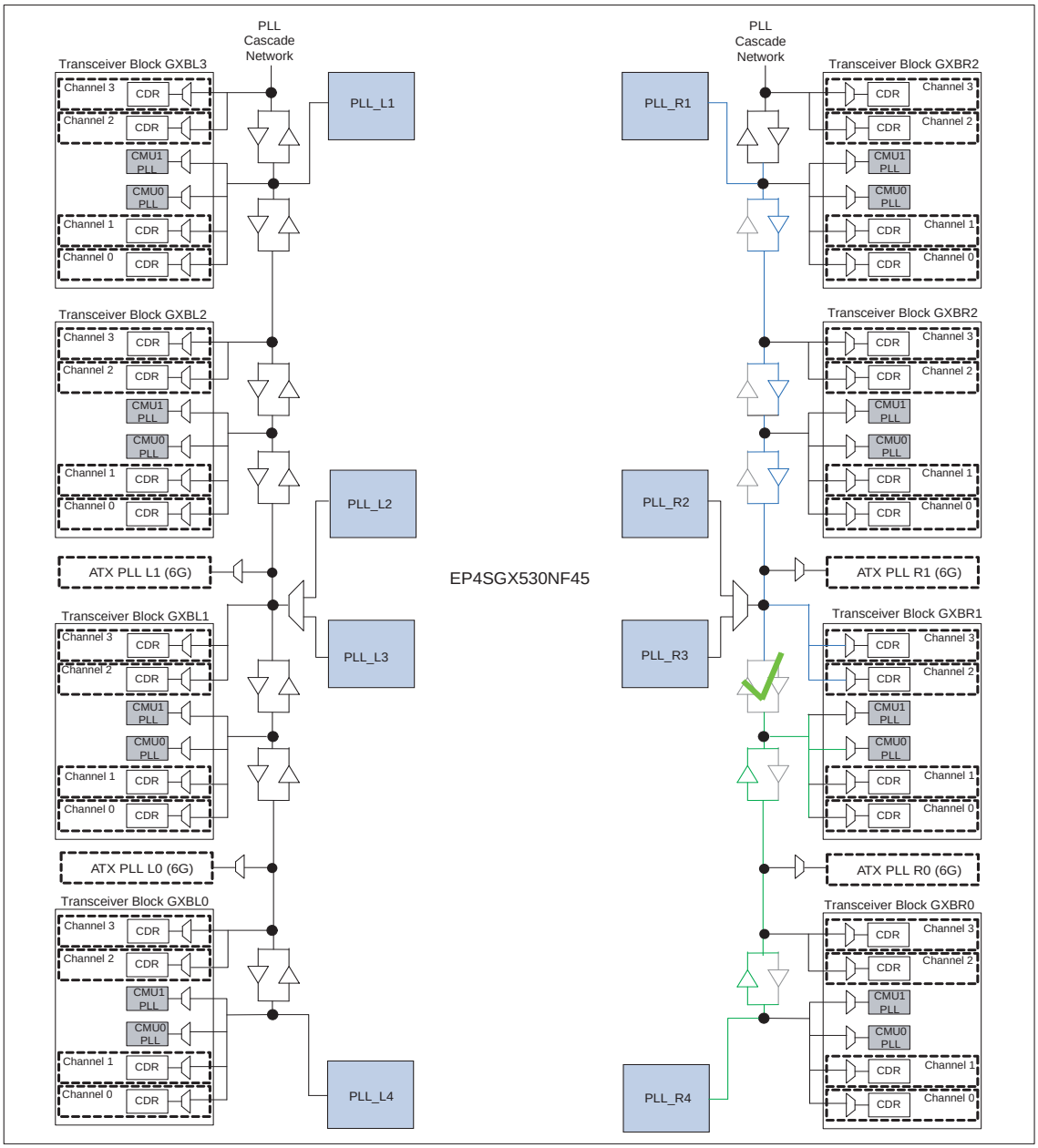
Figure 2–9. Illegal FPGA Fabric-Transceiver PLL Cascading Configuration



Case 2: use `PLL_R1` to provide the input reference clock to the receiver CDRs in channel 2 and channel 3 (shown in BLUE) and use `PLL_R4` to provide the input reference clock to the `CMU0` PLL (shown in GREEN) in transceiver block `GXBR1`.

Figure 2–10 shows that this FPGA fabric-Transceiver PLL cascading configuration is legal as there is no crossover of the cascade clock paths on the PLL cascade network.

Figure 2–10. Legal FPGA Fabric-Transceiver PLL Cascading Configuration



Left and Right, Left, or Right PLL in VCO Bypass Mode

If all CMU channels on the same side of the device are configured as channels, all `refclk` pins are used as receiver serial input data pins. All CMU PLLs are also used as receiver CDRs. In such designs, you must use the 6G ATX PLLs to generate the high-speed serial and low-speed parallel transceiver clocks provided that the configured data rate is supported by the 6G ATX PLLs. Additionally, Altera recommends providing the input reference clock to the 6G ATX PLL using the left or right PLL cascade clock line because none of the `refclk` pins are available. To avoid jitter amplification because of cascading of the left or right PLL to the 6G ATX PLL, you must place the left or right PLL in VCO bypass mode.

For more information about CMU PLLs, refer to "Configuring CMU Channels as Transceiver Channels" in the *Stratix IV Transceiver Architecture* chapter.

Figure 2–11 shows that in VCO bypass mode, the input reference clock from the dedicated FPGA CLK pins to the `inclk` port of the left and right, left, or right PLL bypasses the PLL loop and is driven directly on the PLL output clock port.

Figure 2–11. Left and Right, Left, or Right PLL in VCO Bypass Mode

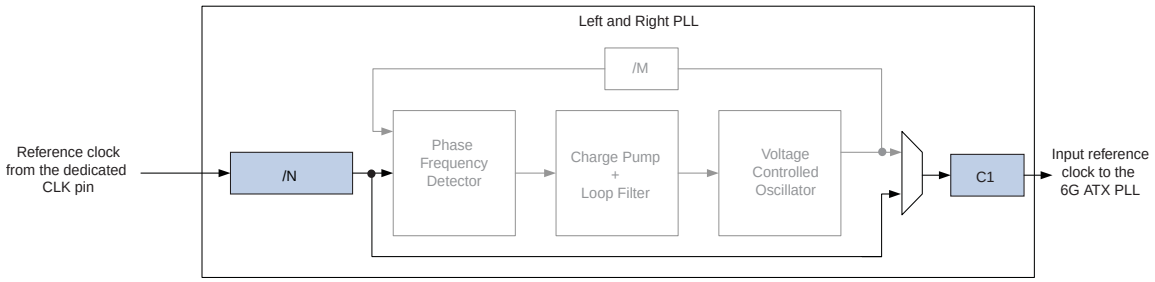
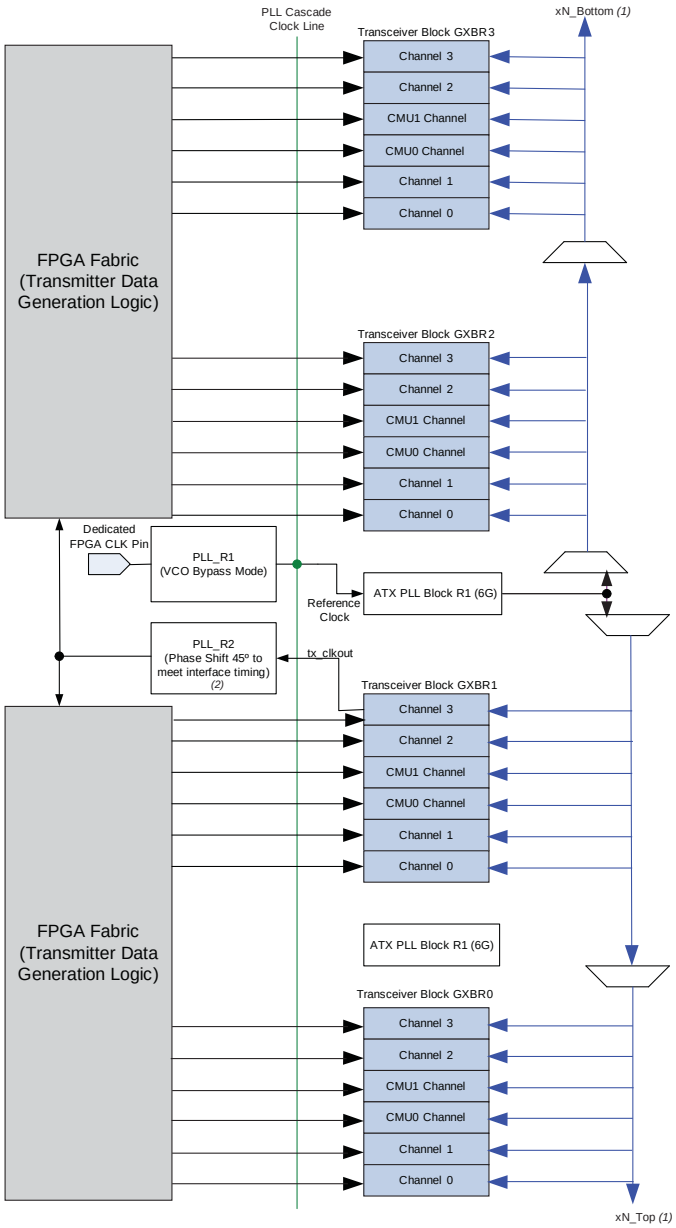


Figure 2–12 shows 24 channels on the right side of the EP4SGX530NF45 device configured in Basic (PMA Direct) ×N mode running at 6.5 Gbps with a 20-bit FPGA fabric-PMA interface width. Because all 24 channels on the right side of the device are configured in Basic (PMA Direct) ×N mode, use the right `PLL_R1` configured in VCO bypass mode to provide the input reference clock to the 6G ATX PLL.

Because the data rate of 6.5 Gbps requires a left and right, left, or right PLL to meet FPGA fabric-Transmitter PMA interface timing, the `tx_clkout` from one of the 24 channels is phase shifted using `PLL_R2`. Use the phase-shifted output clock from `PLL_R2` to clock the FPGA fabric logic that generates the transmitter parallel data and control signals.

Figure 2–12. Input Reference Clocking Using Left and Right, Left, or Right PLL in VCO Bypass Mode *(Note 3)*



Notes to Figure 2–12:

(1) For more information, refer to "Transceiver Channel Datapath Clocking" on page 2–20.

(2) For more information, refer to *AN 580: Achieving Timing Closure in Basic (PMA Direct) Functional Mode*.

(3) The green line represents the PLL cascade clock line and the blue lines represent 6G ATX PLL R1.

For more information about configuring left or right PLLs in VCO bypass mode, refer to "Configuration Example 4: Configuring Left and Right, Left, or Right PLL in VCO Bypass Mode" on page 2–77.

Transceiver Channel Datapath Clocking

This section describes the transmitter channel and receiver channel datapath clocking in various configurations. Datapath clocking varies with physical coding sublayer (PCS) configurations in different functional modes as well as channel bonding options. This section contains:

- "Transmitter Channel Datapath Clocking" on page 2–20
- "Receiver Channel Datapath Clocking" on page 2–37

Clocking described in this section is internal to the transceiver and clock routing is primarily performed by the Quartus II software.

Transmitter Channel Datapath Clocking

This section describes the transmitter channel PMA and PCS datapath clocking in non-bonded and bonded channel configurations:

- "Non-Bonded Channel Configurations" on page 2–23
- "Bonded Channel Configurations" on page 2–26
- "Non-Bonded Basic (PMA Direct) Mode Channel Configurations" on page 2–32
- "Bonded Basic (PMA Direct) ×N Mode Channel Configurations" on page 2–34

Transmitter Channel-to-Channel Skew Optimization for Modes Other than Basic (PMA Direct) Mode

High-speed serial clock and low-speed parallel clock skew between channels and unequal latency in the transmitter phase compensation FIFO contribute to transmitter channel-to-channel skew. Transmitter datapath clocking is set up to provide low channel-to-channel skew when compared with non-bonded channel configurations.

- In bonded channel configurations—the high-speed serial clock and low-speed parallel clock for all bonded channels are generated by the `CMU0` clock divider or the ATX clock divider block, resulting in lower channel-to-channel clock skew.

  The transmitter phase compensation FIFO in all bonded channels (except in Basic [PMA Direct] ×N mode) share common pointers and control logic generated in the central control unit (CCU), resulting in equal latency in the transmitter phase compensation FIFO of all bonded channels. The lower transceiver clock skew and equal latency in the transmitter phase compensation FIFOs in all channels provides lower channel-to-channel skew in bonded channel configurations.

- In non-bonded channel configurations—the high-speed serial clock and low-speed parallel clock in each channel are generated independently by its local clock divider. This results in higher channel-to-channel clock skew.

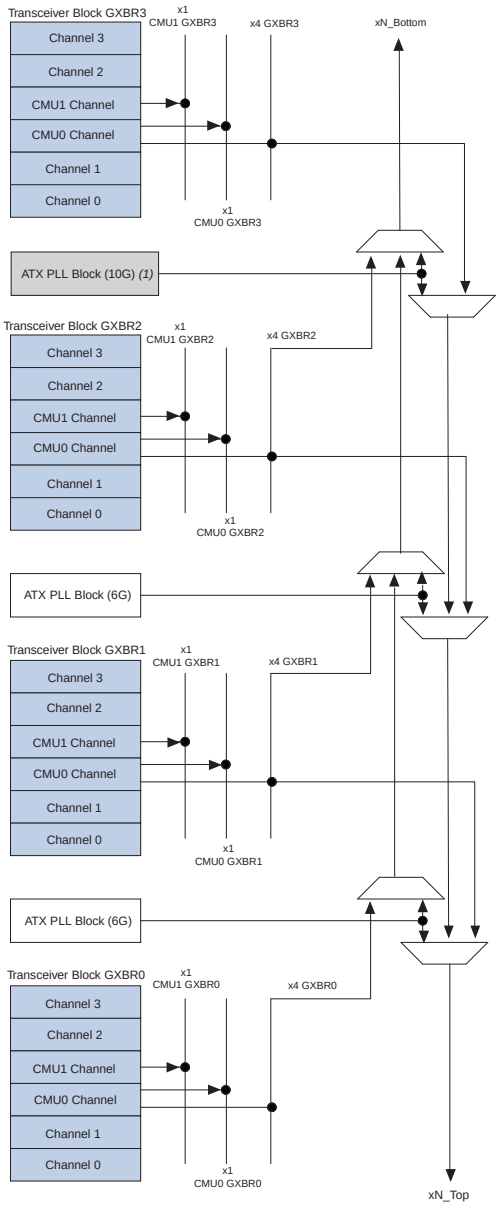
  The transmitter phase compensation FIFO in each non-bonded channel (except in Basic [PMA Direct] mode) has its own pointers and control logic that can result in unequal latency in the transmitter phase compensation FIFO of each channel. The higher transceiver clock skew and unequal latency in the transmitter phase compensation FIFO in each channel can result in higher channel-to-channel skew.

Transmitter Channel Datapath Clocking Resources

The Stratix IV transceivers support various non-bonded and bonded transceiver clocking configurations through the dedicated ×1, ×4, and ×N high-speed serial and low-speed parallel clock lines.

Figure 2–13 shows the transceiver clock distribution in ×1, ×4, ×8, and ×N bonded modes.

Figure 2–13. Transceiver Clock Distribution in the Stratix IV GT EP4S100G5F45 and Stratix IV GX EP4SGX530KF40 Devices



Note to Table 2–14:

(1) The 10G ATX PLL block is not available for the EP4SGX530KF40 device.

Non-bonded and bonded configurations use the following:

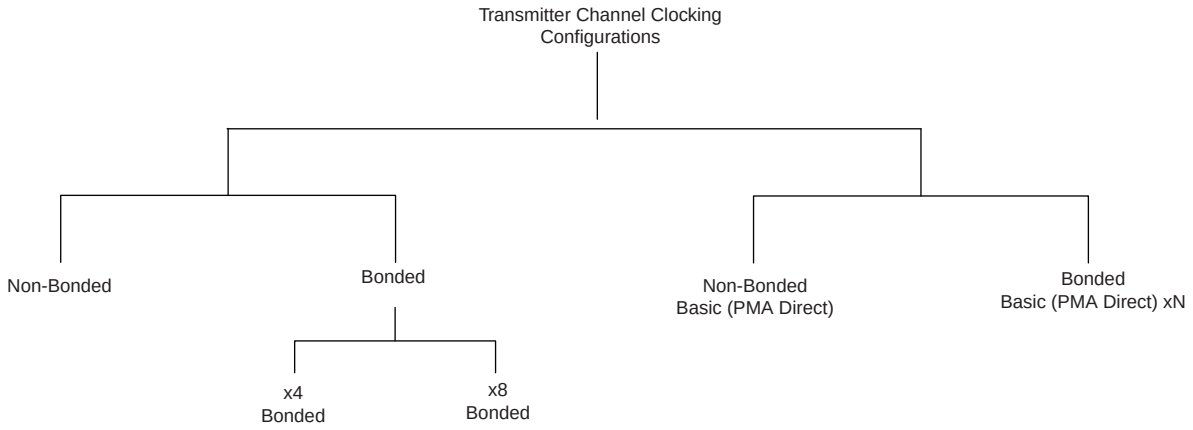
- ×1 non-bonded configurations use the ×1 clock lines to distribute only the high-speed serial transceiver clock synthesized by the `CMU0` PLL or `CMU1` PLL to the clock transmitter channels located in the same transceiver block. The low-speed parallel transceiver clock is generated in the transceiver channels using the local clock dividers.
- ×4 bonded configurations use the `×4_GXB` clock lines to distribute both the high-speed serial and low-speed parallel transceiver clocks generated by the `CMU0_Channel` to clock the bonded transmitter channels located in the same transceiver block.
- ×8 and ×N bonded configurations use the `×N_Top` or `×N_Bottom` clock lines to distribute both the high-speed serial and low-speed parallel transceiver clocks generated by the `CMU0` channel block to all bonded transmitter channels located across transceiver blocks.

ATX PLLs always use ×N lines to distribute the high-speed serial and low-speed parallel transceiver clocks. Use the `×N_Top` line if the `CMU0` PLL or ATX PLL that generates the transceiver clocks is located at the top of the transmitter channel. Use the `×N_Bottom` line if the `CMU0` PLL or ATX PLL is located at the bottom of the transmitter channel. Because there is only one `×N_Top` and `×N_Bottom` line on each side of the device, using an ATX PLL limits the use of the ×N clock lines to distribute the transceiver clocks to other transmitter channels in the design.

Transmitter Channel Clocking Configurations

Figure 2–14 shows various transmitter channel clocking configurations.

Figure 2–14. Transmitter Channel Clocking Configurations



Transmitter channels configured in modes other than Basic (PMA Direct) mode use both the transmitter channel PCS and PMA blocks. As a result, Stratix IV devices allow placing these transmitter channels only in the four regular channels of a transceiver block. Stratix IV devices do not allow configuring the CMU channels in any mode other than Basic (PMA Direct) mode because of the absence of PCS blocks in the CMU channels.

The transmitter channel datapath clocking in modes other than Basic (PMA Direct) mode depends on whether the transmitter channel is configured in non-bonded or bonded mode.

Non-Bonded Channel Configurations

The following modes other than Basic (PMA Direct) functional mode have a non-bonded transmitter channel configuration:

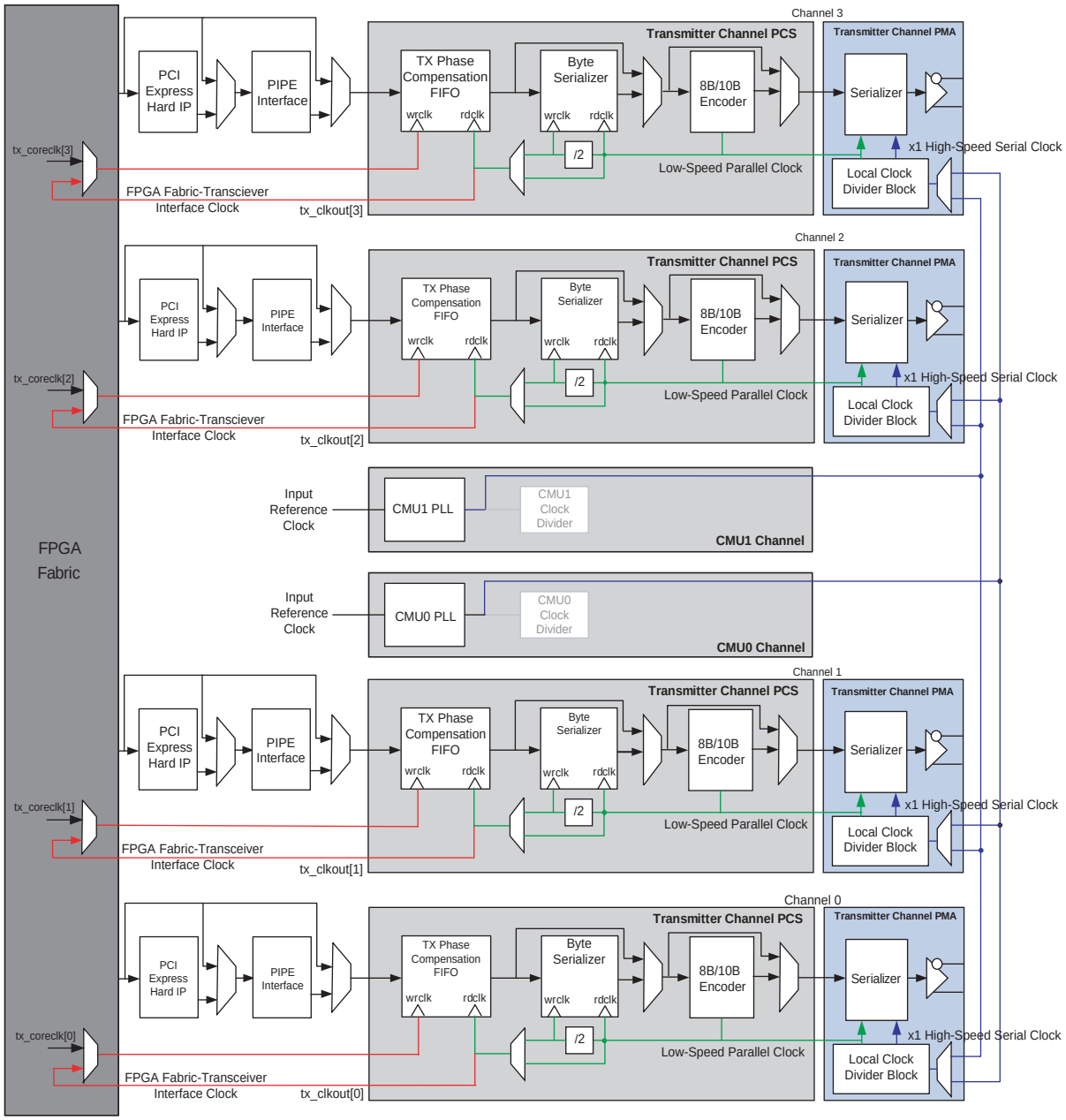
- PCI Express (PIPE) ×1—Gen1 and Gen2
- Gigabit Ethernet (GIGE)
- Serial RapidIO
- SONET/SDH
- SDI
- (OIF) CEI PHY Interface
- Basic (except Basic ×4 and Basic ×8 modes)
- Deterministic Latency

Use the CMU channels to generate transceiver clocks for all the non-bonded functional modes listed above. Additionally, use the ATX PLLs to generate transceiver clocks for PCI Express (PIPE) ×1 Gen 2, (OIF) CEI PHY, and Basic functional mode.

For data rates supported by ATX PLLs, refer to the "Auxiliary Transmit (ATX) PLL Block" section in the *Stratix IV Transceiver Architecture* chapter.

Figure 2–15 shows transmitter channel datapath clocking in non-bonded channel configurations when clocked using the CMU PLLs.

Figure 2–15. Transmitter Datapath Clocking in a Non-Bonded Configuration Clocked by CMU PLL *(Note 1)*



Note to Figure 2–15:

(1) The red lines represent the FPGA fabric-Transceiver interface clock, the green lines represent the low-speed parallel clock, and the blue lines represent the ×1 high-speed serial clock.

In non-bonded channel configurations clocked by CMU PLL, each channel can derive its clock independently from either `CMU0` PLL or `CMU1` PLL within the same transceiver block. The CMU PLL synthesizes the input reference clock to generate a clock that is distributed to the local clock divider block in each channel using the ×1 high-speed serial clock line. Depending on the configured functional mode, the local clock divider block in each channel generates the low-speed parallel clock and high-speed serial clock. The serializer in the transmitter channel PMA uses both the low-speed parallel clock and high-speed serial clock for its parallel-in-serial-out operation. The low-speed parallel clock clocks the 8B/10B encoder (if enabled) and the write port of the byte serializer (if enabled) in the transmitter channel PCS.

Depending on whether you use the byte serializer or not, the low-speed parallel clock (when you do not use the byte serializer) or a divide-by-two version of the low-speed parallel clock (when you use the byte serializer) from the `CMU0` clock divider block clocks the read port of the transmitter phase compensation FIFO in all four bonded channels. This clock is driven directly on the `tx_clkout` port as the FPGA fabric-Transceiver interface clock. You can use the `coreclkout` signal to clock the transmitter data and control logic in the FPGA fabric for all four bonded channels.

If you configure the ATX PLL to clock the transmitter channel, the ATX PLL block drives the high-speed serial clock and low-speed parallel clock to the transmitter channel on the `×N_Top` or `×N_Bottom` lines.

For more information, refer to the *Configuring Multiple Protocols and Data Rates* chapter.

Table 2–2 lists the transmitter channel datapath clock frequencies in non-bonded functional modes that have a fixed data rate.

Table 2–6. Transmitter Channel Datapath Clock Frequencies in Non-Bonded Functional Modes

Functional Mode	Data Rate	High-Speed Serial Clock Frequency	Low-Speed Parallel Clock Frequency (MHz)	FPGA Fabric-Transceiver Interface Clock Frequency	
				Without Byte Serializer (MHz)	With Byte Serializer (MHz)
PCI Express (PIPE) ×1 (Gen 1)	2.5 Gbps	1.25 GHz	250	250	125
PCI Express (PIPE) ×1 (Gen 2)	5 Gbps	2.5 GHz	500	N/A	250
GIGE	1.25 Gbps	625 MHz	125	125	N/A
Serial RapidIO	1.25 Gbps	625 MHz	125	N/A	62.5
	2.5 Gbps	1.25 GHz	250	N/A	125
	3.125 Gbps	1.5625 GHz	312.5	N/A	156.25
SONET/SDH OC12	622 Mbps	311 MHz	77.75	77.75	N/A
SONET/SDH OC48	2.488 Gbps	1.244 GHz	311	N/A	155.5
HD-SDI	1.485 Gbps	742.5 MHz	148.5	148.5	74.25
	1.4835 Gbps	741.75 MHz	148.35	148.35	74.175
3G-SDI	2.97 Gbps	1.485 GHz	297	N/A	148.5
	2.967 Gbps	1.4835 GHz	296.7	N/A	148.35

Bonded Channel Configurations

In PCS and PMA bonded channel configurations, the PCS and PMA blocks of all bonded channels are clocked by the same low-speed parallel clock and high-speed serial clock from the `CMU0` clock divider or the ATX PLL block. The phase compensation FIFOs of all bonded channels also share common read and write pointers and enable signals generated in the CCU.

Stratix IV devices support ×4 PCS and PMA channel bonding that allows bonding of four channels within the same transceiver block. Stratix IV devices also support ×8 channel bonding that allows bonding of eight PCS and PMA channels across two transceiver blocks on the same side of the device.

×4 PCS and PMA Bonded Channel Configuration

The following functional modes support ×4 PCS and PMA bonded transmitter channel configuration:

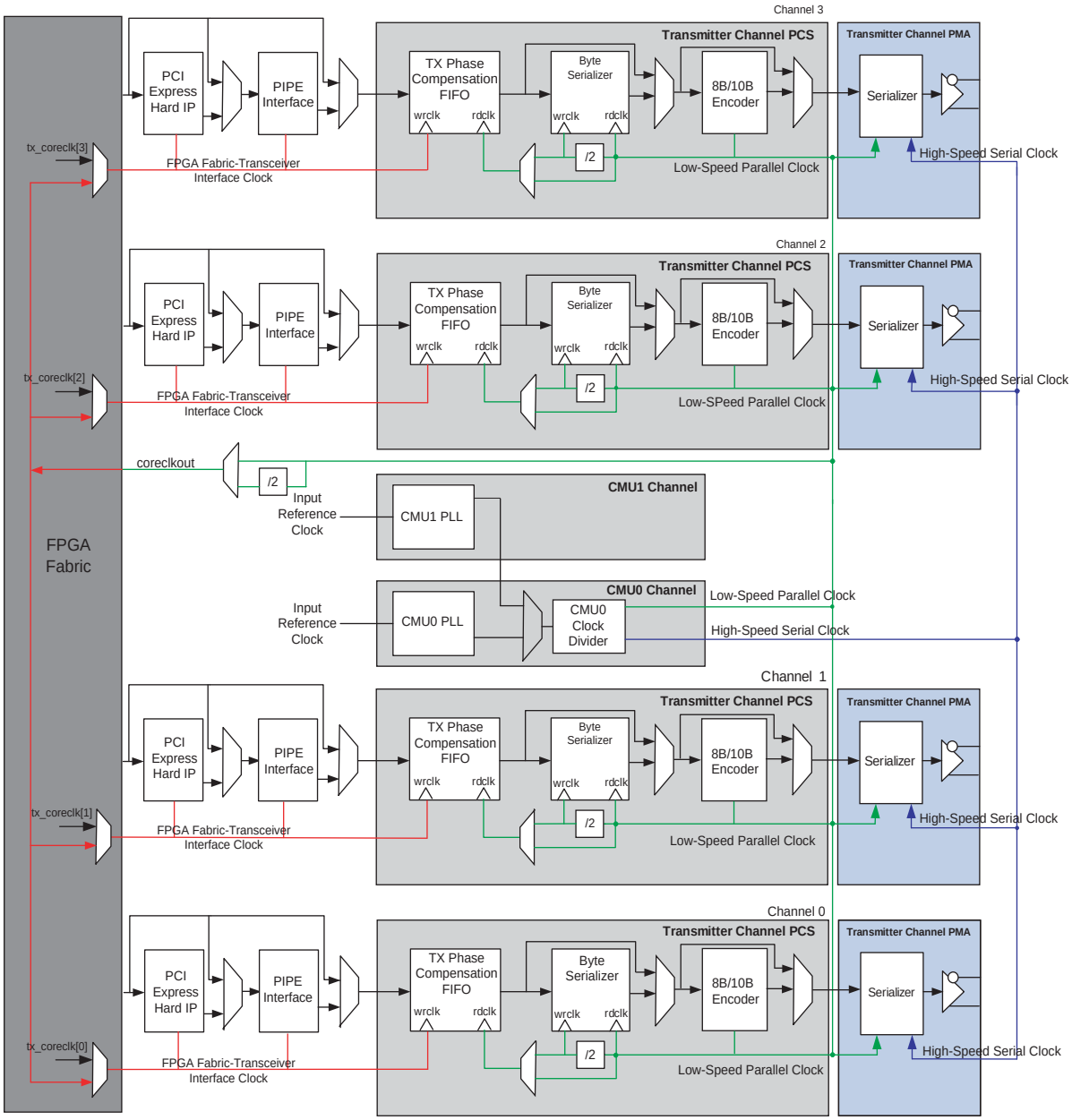
- PCI Express (PIPE) ×4—Gen1 and Gen2
- XAUI
- Basic ×4

Use the CMU channels to generate the transceiver clocks for all ×4 bonded functional modes listed above. Additionally, use the ATX PLLs to generate the transceiver clocks for PCI Express (PIPE) ×4 Gen 2 and Basic ×4 functional mode.

You must assign `tx_dataout[0]` of the ×4 bonded link (XAUI or PCI Express [PIPE] ×4) to physical channel 0 of the transceiver block, `tx_dataout[1]` to physical channel 1 of the transceiver block, `tx_dataout[2]` to physical channel 2 of the transceiver block, and `tx_dataout[3]` to physical channel 3 of the transceiver block. Otherwise, the Quartus II compilation errors out.

Figure 2–16 shows the transmitter channel datapath clocking in ×4 channel bonding configurations when clocked using the `CMU0` channel.

Figure 2–16. Transmitter Datapath Clocking in x4 Bonded Configurations *(Note 1)*



Note to Figure 2–16:

(1) The red lines represent the FPGA fabric-Transceiver interface clock, the green lines represent the low-speed parallel clock, and the blue lines represent the high-speed serial clock.

The transceiver clocks are distributed to the four bonded channels on the ×4 high-speed serial and ×4 low-speed parallel clock lines. The serializer in the transmitter channel PMA of the four bonded channels uses the same low-speed parallel clock and high-speed serial clock from `CMU0 Channel` for their parallel-in-serial-out operation. The low-speed parallel clock clocks the 8B/10B encoder and the write port of the byte serializer (if enabled) in the transmitter channel PCS.

Depending on whether the you use the byte serializer or not, the low-speed parallel clock (when you do not use the byte serializer) or a divide-by-two version of the low-speed parallel clock (when you use the byte serializer) from the `CMU0` clock divider block clocks the read port of the transmitter phase compensation FIFO in all four bonded channels. This clock is driven directly on the `coreclkout` port as the FPGA fabric-Transceiver interface clock. You can use the `coreclkout` signal to clock the transmitter data and control logic in the FPGA fabric for all four bonded channels.

The ATX PLL block drives the high-speed serial clock and low-speed parallel clock to the transmitter channels on the `×N_Top` or `×N_Bottom` lines.

For more information, refer to the *Configuring Multiple Protocols and Data Rates for Stratix IV* chapter.

In ×4 PCS and PMA bonded channel configurations, the transmitter phase compensation FIFOs in all four bonded channels share common read and write pointers and enable signals generated in the `CMU0` channel of the transceiver block. This ensures equal transmitter phase compensation FIFO latency across all four bonded channels, resulting in low transmitter channel-to-channel skew.

Table 2–3 lists the transmitter datapath clock frequencies in ×4 bonded functional modes that have a fixed data rate.

Table 2–7. Transmitter Datapath Clock Frequencies in ×4 Bonded Functional Modes

Functional Mode	Data Rate	High-Speed Serial Clock Frequency	Low-Speed Parallel Clock Frequency (MHz)	FPGA Fabric-Transceiver Interface Clock Frequency	
				Without Byte Serializer (MHz)	With Byte Serializer (MHz)
PCI Express (PIPE) ×4 (Gen 1)	2.5 Gbps	1.25 GHz	250	250	125
PCI Express (PIPE) ×4 (Gen 2)	5 Gbps	2.5 GHz	500	N/A	250
XAUI	3.125 Gbps	1.5625 GHz	312.5	N/A	156.25

×8 PCS and PMA Bonded Channel Configuration

The following functional modes support ×8 PCS and PMA bonded transmitter channel configuration:

- PCI Express (PIPE) ×8—Gen1 and Gen2
- Basic ×8

Use the CMU channels to generate the transceiver clocks for all ×8 PCS and PMA bonded functional modes listed above. Additionally, use the ATX PLLs to generate the transceiver clocks for PCI Express (PIPE) ×8 Gen 2 and Basic ×8 functional modes.

The eight bonded channels are located in two transceiver blocks, referred to as the master transceiver block and the slave transceiver block, with four channels each. When clocked using a CMU PLL, the `CMU0` clock divider in `CMU0` channel of the master transceiver block drives the high-speed serial clock and low-speed parallel clock on the `xN_Top` clock line. The serializer in the transmitter channel PMA of all eight bonded channels uses the same low-speed parallel clock and high-speed serial clock driven by the `CMU0` channel of the master transceiver block on the `xN_Top` clock line. The low-speed parallel clock from `CMU0` channel of the master transceiver block clocks the 8B/10B encoder and the write port of the byte serializer (if enabled) in the transmitter channel PCS of all eight channels.

Depending on whether you use the byte serializer or not, the low-speed parallel clock (when you do not use the byte serializer) or a divide-by-two version of the low-speed parallel clock (when you use the byte serializer) from the `CMU0` clock divider block clocks the read port of the transmitter phase compensation FIFO in all eight bonded channels. This clock is driven directly on the `coreclkout` port as the FPGA fabric-Transceiver interface clock. You can use the `coreclkout` signal to clock the transmitter data and control logic in the FPGA fabric for all eight bonded channels.

If you choose the ATX PLL to generate the transceiver clocks for the ×8 bonded channels, Altera recommends placing the ATX PLL between the master and slave transceiver block to minimize transmitter channel-to-channel skew. In this configuration, the ATX PLL block drives the high-speed serial clock and low-speed parallel clock to the master transceiver block on the `×N_Bottom` lines. It drives the high-speed serial clock and low-speed parallel clock to the slave transceiver block on the `×N_Top` lines.

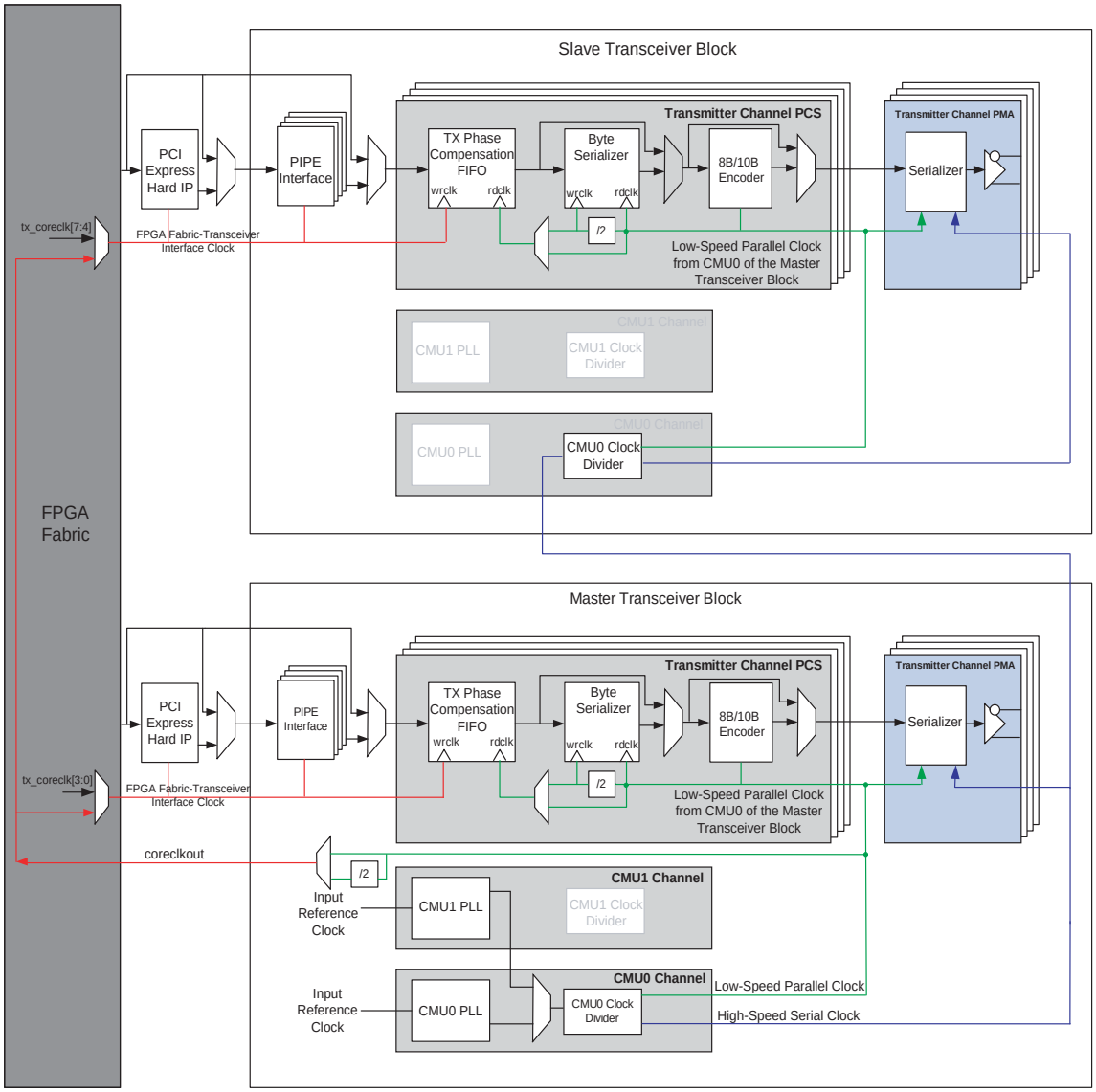
For more information, refer to the *Configuring Multiple Protocols and Data Rates for Stratix IV* chapter.

In PCI Express (PIPE) ×8 and Basic ×8 bonded channel configurations, the transmitter phase compensation FIFOs in all eight bonded channels share common read and write pointers and enable signals generated in the central control unit of the master transceiver block. This ensures equal transmitter phase compensation FIFO latency across all eight bonded channels, resulting in low transmitter channel-to-channel skew.

The difference in clock routing delays between the ×4 clock lines and the ×N clock lines can result in higher transmitter channel-to-channel skew. To compensate for this difference in clock routing delays between the ×4 and the ×N clock lines, the Stratix IV transceivers introduce a fixed amount of delay in the ×4 clock lines of the transceiver block whose `CMU0` channel generates the transceiver clocks in Basic ×8 bonded channel configuration.

Figure 2–17 shows the transmitter datapath clocking in PCI Express (PIPE) ×8 channel bonding configurations when clocked using the CMU channel in the master transceiver block.

Figure 2–17. Transmitter Datapath Clocking in x8 Bonded Configuration *(Note 1)*



Note to Figure 2–17:

(1) The red lines represent the FPGA fabric-Transceiver interface clock, the green lines represent the low-speed parallel clock, and the blue lines represent the high-speed serial clock.

Figure 2–18 through Figure 2–20 show the allowed master and slave transceiver block locations and PCI Express (PIPE) logical lane to physical transceiver channel mapping in all Stratix IV devices.

☞ The Quartus II compilation errors out if you do not map the PCI Express (PIPE) logical lanes to the physical transceiver channels, as shown in Figure 2–18 through Figure 2–20.

Figure 2–18 shows one PCI Express (PIPE) ×8 link in two transceiver block devices and two PCI Express (PIPE) ×8 links in four transceiver block devices.

Figure 2–18. One PCI Express (PIPE) x8 Link in Two Transceiver Block Devices and Two PCI Express (PIPE) x8 Links in Four Transceiver Block Devices

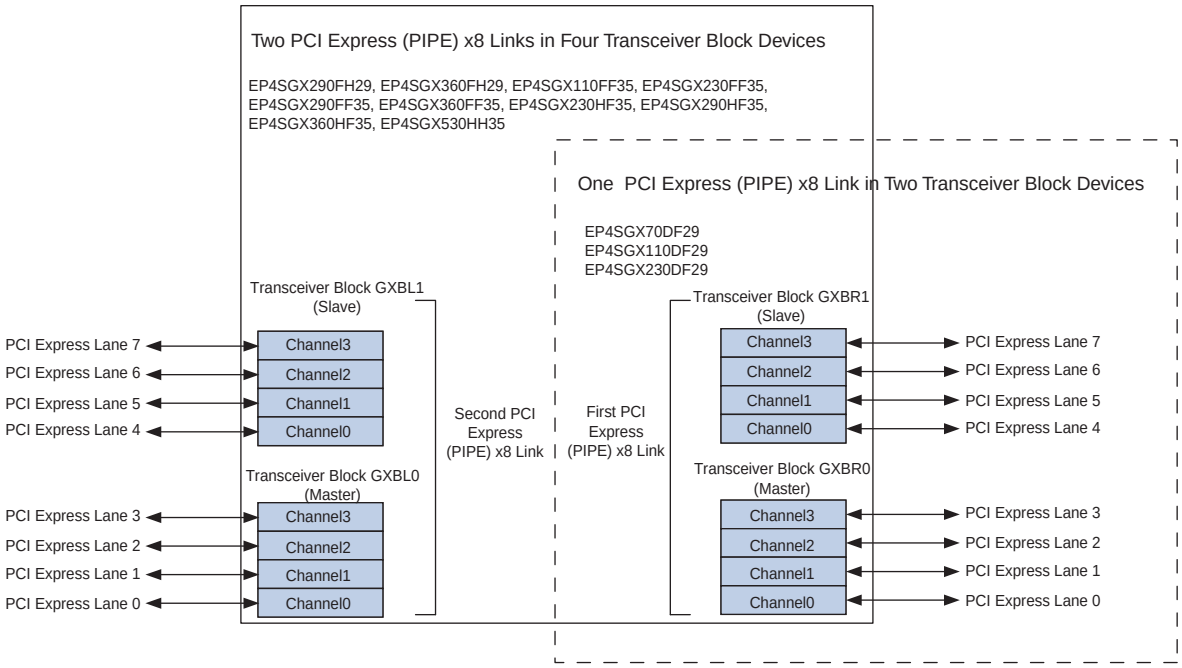
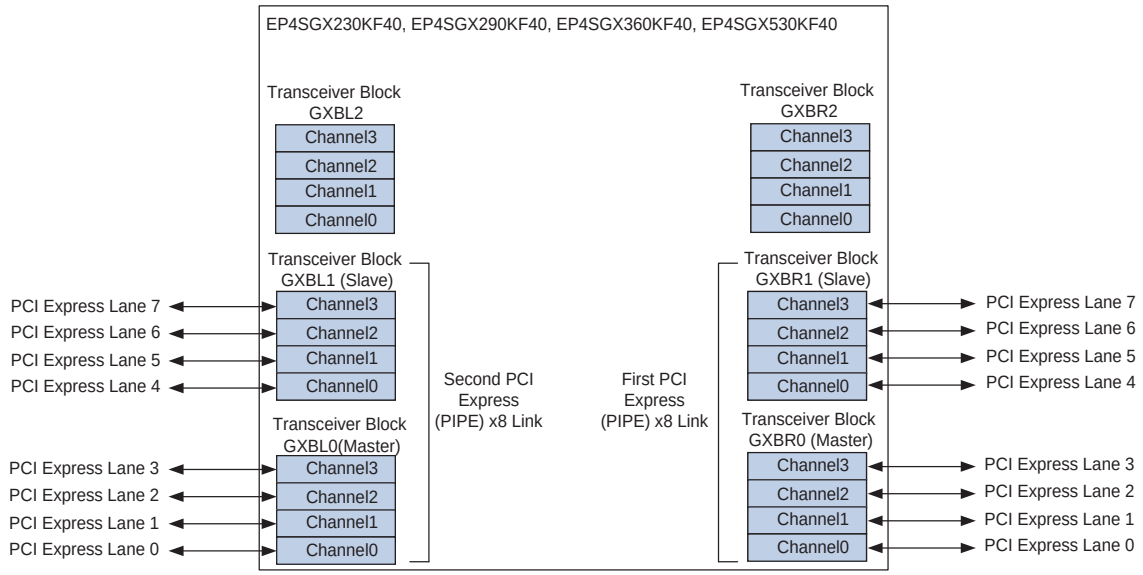


Figure 2–19 shows two PCI Express (PIPE) ×8 links in six transceiver block devices.

Figure 2–19. Two PCI Express (PIPE) ×8 Links in Six Transceiver Block Devices *(Note 1)*

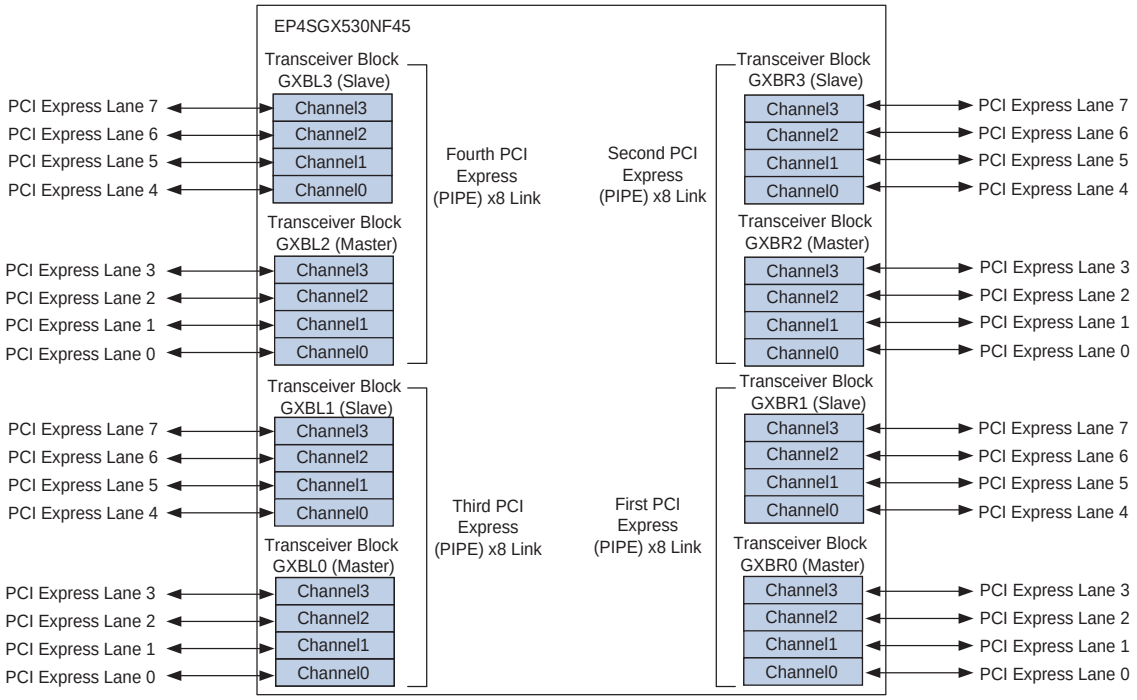


Note to Figure 2–19:

(1) Stratix IV devices with six transceiver blocks allow a maximum of two PCI Express (PIPE) ×8 links occupying four transceiver blocks. You can configure the other two transceiver blocks to implement other functional modes.

Figure 2–20 shows four PCI Express (PIPE) ×8 links in eight transceiver block devices.

Figure 2–20. Four PCI Express (PIPE) ×8 Links in Eight Transceiver Block Devices

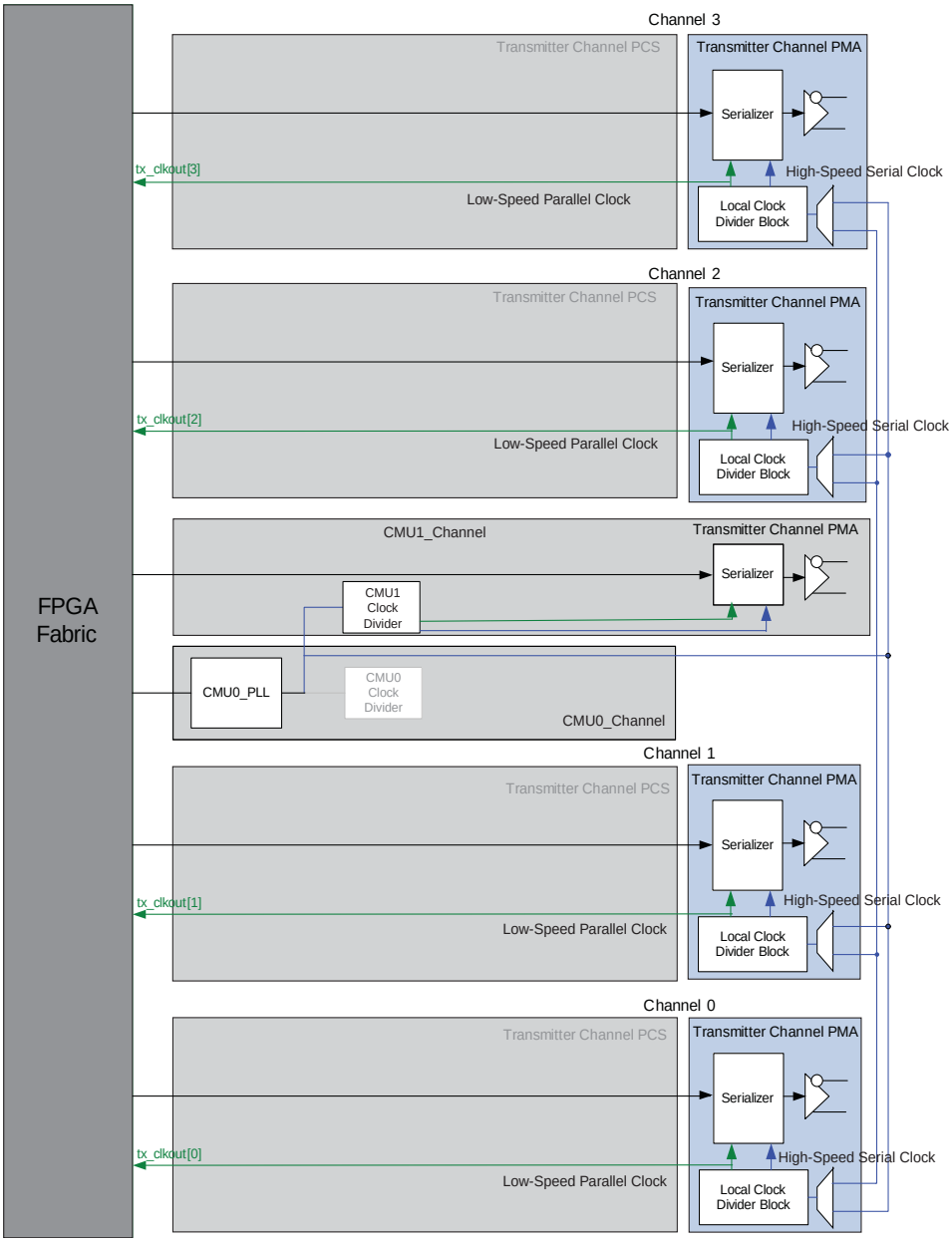


Non-Bonded Basic (PMA Direct) Mode Channel Configurations

Figure 2–21 shows four regular channels and the `CMU1` channel in a transceiver block configured in non-bonded Basic (PMA Direct) mode. Each channel derives its clock independently from either the `CMU0` PLL or `CMU1` PLL within the same transceiver block if the CMU channel is configured as a CMU PLL.

For more information about Basic (PMA Direct) mode, refer to the *Stratix IV Transceiver Architecture* chapter.

Figure 2–21. Transmitter Channel PMA Directly Interfacing to the User Logic in the FPGA Fabric *(Note 1)*



Note to Figure 2–21:

(1) The green lines represent the low-speed parallel clock and the blue lines represent the high-speed serial clock.

☞ Stratix IV devices do not allow the 6G ATX PLL to generate transceiver clocks in non-bonded Basic (PMA Direct) mode. The transmitter clock for channels configured in non-bonded Basic (PMA Direct) mode must be generated by one of the CMU PLLs in the transceiver block containing the channels.

The CMU0 PLL synthesizes the input reference clock to generate a clock that is distributed to the local clock divider block in each of the four regular channels using the ×1 high-speed serial clock line. It is also forwarded to the CMU1 clock divider in the CMU1 channel configured as a non-bonded Basic (PMA-Direct) channel. The local clock divider block in each regular channel and the CMU1 clock divider in the CMU1 channel generate the low-speed parallel clock and high-speed serial clock. The serializer in the transmitter channel PMA of each channel uses both the low-speed parallel clock and high-speed serial clock for its parallel-in-serial-out operation.

The low-speed parallel clock is also driven directly on the tx_clkout port as the FPGA fabric-Transceiver interface clock. You can use the tx_clkout port to clock transmitter data and control logic in the FPGA fabric.

Bonded Basic (PMA Direct) ×N Mode Channel Configurations

Bonded Basic (PMA Direct) ×N mode offers low transmitter channel-to-channel skew in addition to the flexibility of implementing custom PCS logic in the FPGA fabric. Stratix IV devices allow bonding all regular channels and CMU channels on one side of the device in Basic (PMA Direct) ×N mode. For example, devices such as EP4SGX530NF45 or EP4S100G5F45 allow bonding of up to 24 channels placed in four transceiver blocks on each side of the device.

☞ The coreclkout port is not available in Basic (PMA Direct) ×N mode.

In bonded channel configurations, the CMU0 clock divider of all the transceiver blocks is used, as shown in Figure 2–17. Unlike bonded channel configurations, in Basic (PMA Direct) ×N configuration:

- If you use the ATX PLL to generate the transceiver datapath clocks, only the clock divider of the ATX PLL is used.
- If you use the CMU PLL to generate the transceiver datapath clocks, only the CMU0 clock divider block of the transceiver block containing the CMU PLL is used.

Figure 2–22 shows transmitter channel clocking for 17 channels configured in Basic (PMA Direct) ×N mode.

Figure 2–22. Transmitter Channel Clocking for 17 Channels Configured in Basic (PMA Direct) ×N Mode

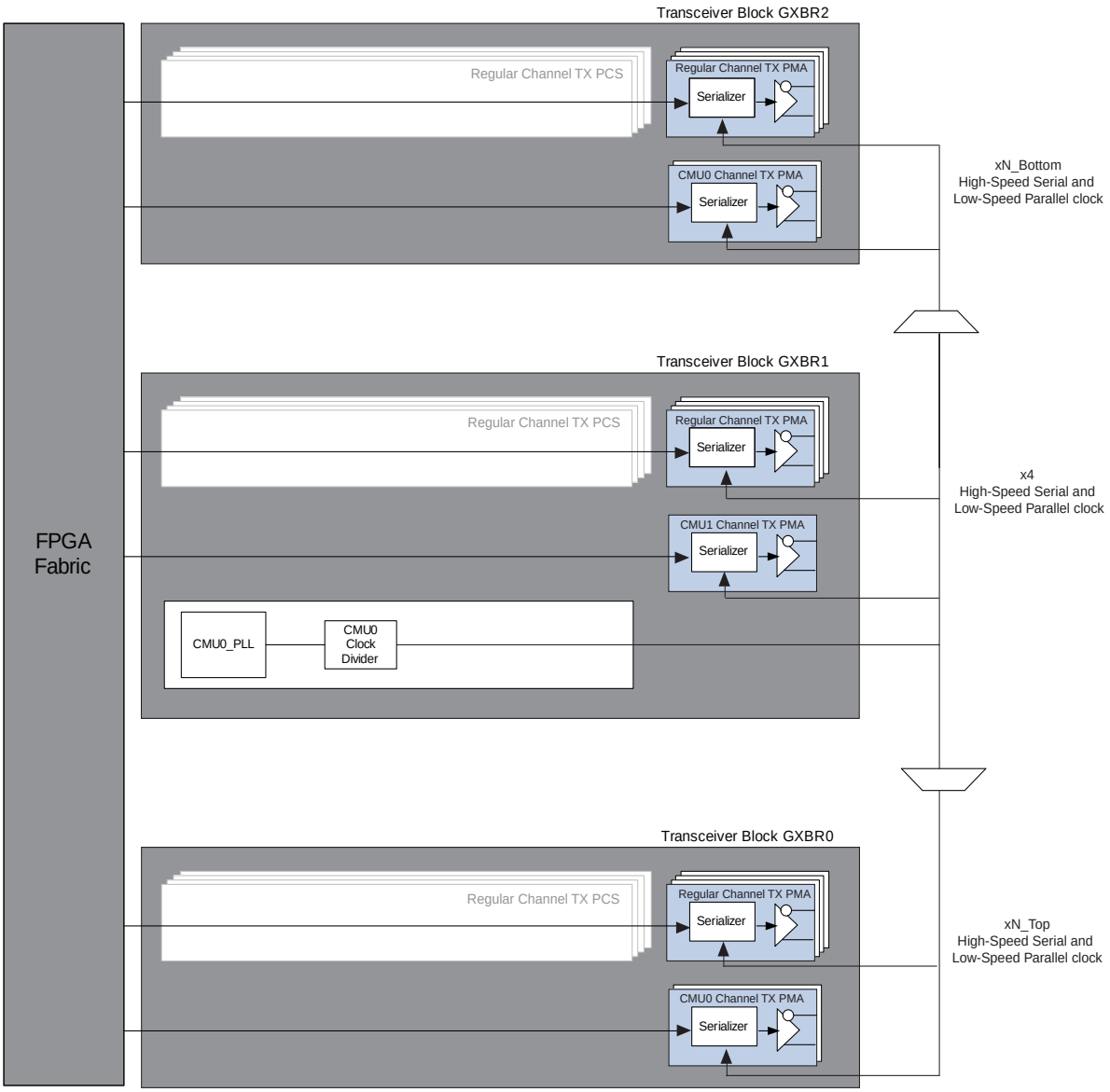


Figure 2–22 shows 17 channels configured in Basic (PMA Direct) ×N mode and located across three transceiver blocks on the right side of the Stratix IV device. Each of the two transceiver blocks, `GXBR0` and `GXBR2`, contain six of the 17 ×N bonded channels located in four regular channels and two CMU channels. The remaining five of the 17 ×N bonded channels are located in four regular channels and the `CMU1` channel of the transceiver block `GXBR1`.

☞ Stratix IV devices allow both CMU channels and 6G ATX PLL blocks to generate the high-speed serial and low-speed parallel transceiver clocks when configured in Basic (PMA Direct) ×N mode.

For more examples regarding this clocking scheme, refer to:

- "Example 1: Channel Configuration with 4-Gbps Data Rate" on page 2–9
- *AN 571: Implementing the SERDES Framer Interface Level 5 (SFI-5.1) Protocol in Stratix IV Devices*
- *AN 572: Implementing the Scalable SERDES Framer Interface (SFI-S) Protocol in Stratix IV GT Devices*

Transmitter Channel-to-Channel Skew Optimization in Basic (PMA Direct) ×N Mode

In Basic (PMA-Direct) ×N mode, the `CMU0` channel distributes the transceiver clocks to the channels placed in the same transceiver block using the ×4 clock lines. The ×4 clock lines drive the `×N_Top` and `×N_Bottom` clock lines to distribute the transceiver clocks to the transmitter channels located in transceiver blocks on the bottom and top.

The difference in clock routing delays between the ×4 clock lines and the ×N clock lines can result in higher transmitter channel-to-channel skew. To compensate for this difference in clock routing delays between the ×4 and the ×N clock lines, the Stratix IV transceivers introduce a fixed amount of delay in the ×4 clock lines of the transceiver block whose `CMU0` channel generates the transceiver clocks.

The delay compensation mechanism engaged in Basic (PMA Direct) mode only compensates for the clock routing delays between the transceiver block whose `CMU0` channel generates the transceiver clocks and its adjacent transceiver block located above and below.

To minimize transmitter channel-to-channel skew in ×N bonded channels, use the recommended placement shown in Table 2–8.

Table 2–8. Recommended Placement of Channels and CMU in Bonded Modes

Channel Placement	CMU Placement
2 adjacent transceiver blocks	In either of the two transceiver blocks.
3 adjacent transceiver blocks	In the middle transceiver block.
4 adjacent transceiver blocks	In either of the middle transceiver blocks.

If you use the ATX PLL to generate the transceiver clocks, Altera recommends that you place the channels in the transceiver blocks adjacent to the ATX PLL on both sides of the ATX PLL.

If the Quartus II software does not automatically pick the most optimal location for skew, refer to *AN 578: Manual Placement of CMU PLLs and ATX PLLs in Stratix IV GX and GT Devices* for manual placement of the CMU and ATX PLLs.

Meeting Timing in Basic (PMA Direct) Mode

Timing may not be met for higher data rates when transceiver channels are configured in Basic (PMA Direct) functional mode. To meet FPGA fabric-Transmitter PMA interface timing above certain data rates, you may need to phase shift the interface clock `tx_clkout` used to clock the transmitter user logic. To meet FPGA fabric-Receiver hold time violations, you may have to modify the way data is captured in the FPGA fabric.

For more information, refer to *AN 580: Achieving Timing Closure in Basic (PMA Direct) Functional Mode*.

Receiver Channel Datapath Clocking

This section describes the receiver PMA and PCS datapath clocking in supported configurations. The receiver datapath clocking varies between non-bonded and bonded channel configurations. It also varies with the use of PCS blocks, such as deskew FIFO and rate matcher. This section describes the following:

- "Non-Bonded Channel Configurations" on page 2–37
- "Bonded Channel Configurations" on page 2–41
- "Basic (PMA Direct) Mode Channel Configurations" on page 2–47

Non-Bonded Channel Configurations

In non-bonded channel configurations, receiver PCS blocks of each channel are clocked independently. Each non-bonded channel also has separate `rx_analogreset` and `rx_digitalreset` signals that allow independent reset of the receiver PCS logic in each channel.

For more information about transceiver reset and power down signals, refer to the *Reset Control and Power Down* chapter.

In non-bonded channel configurations, receiver channel datapath clocking has two scenarios:

- "Non-Bonded Receiver Clocking Without Rate Matcher" on page 2–37
- "Non-Bonded Receiver Clocking with Rate Matcher" on page 2–39

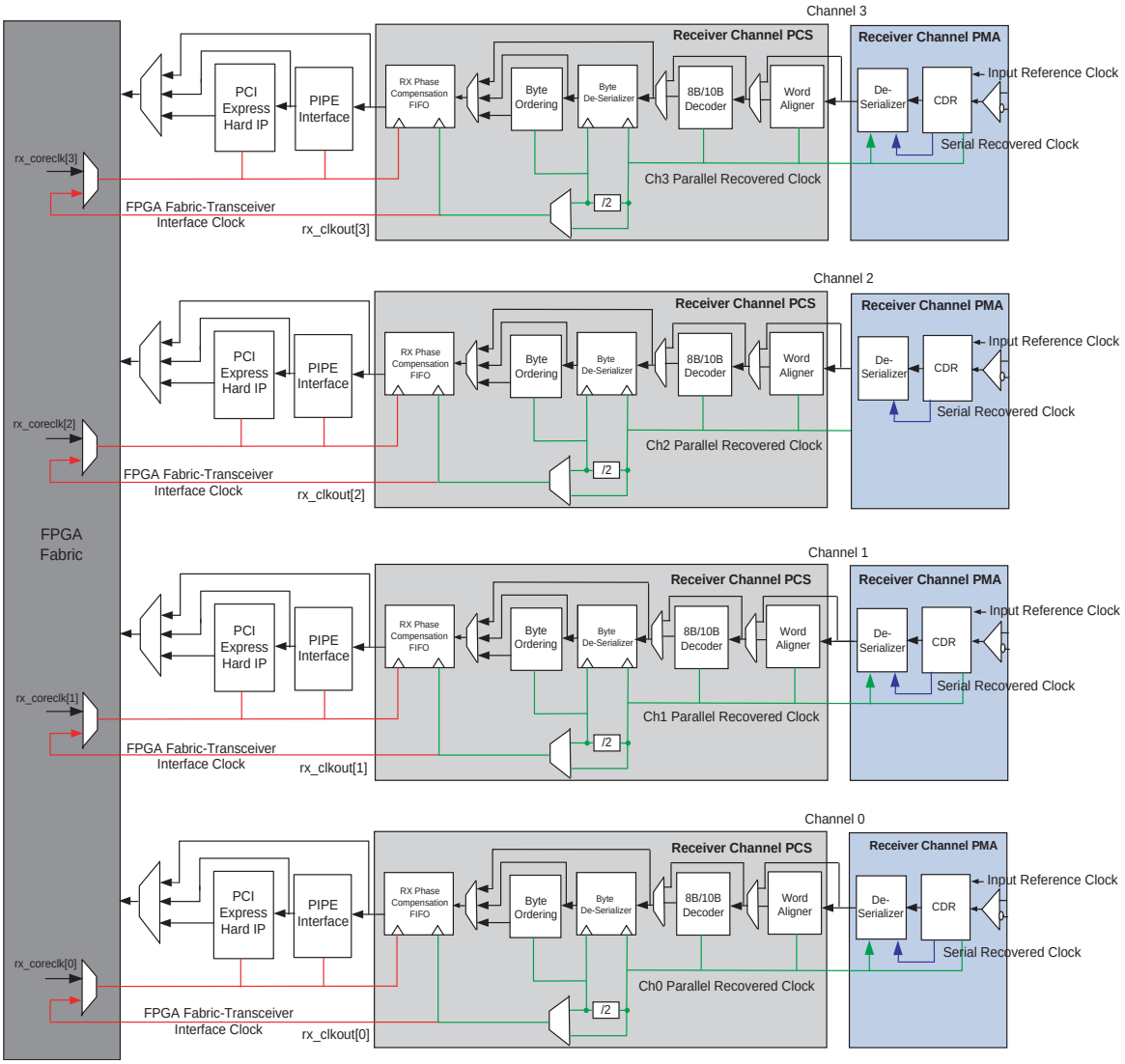
Non-Bonded Receiver Clocking Without Rate Matcher

The following functional modes have non-bonded receiver channel configuration without rate matcher:

- SONET/SDH
- SDI
- (OIF) CEI PHY Interface
- Basic without rate matcher

Figure 2–23 shows receiver datapath clocking in non-bonded channel configurations without rate matcher.

Figure 2–23. Receiver Datapath Clocking in Non-Bonded Configurations Without Rate Matcher *(Note 1)*



Note to Figure 2–23:

(1) The red lines represent the FPGA fabric-Transceiver interface clock, the green lines represent the parallel recovered clock, and the blue lines represent the serial recovered clock.

In non-bonded configurations without rate matcher, the CDR in each receiver channel recovers the serial clock from the received data. The serial recovered clock is divided within the receiver PMA to generate the parallel recovered clock. The deserializer uses the serial recovered clock in the receiver PMA. The parallel recovered clock and deserialized data is forwarded to the receiver PCS. The parallel recovered clock in each channel clocks the word aligner and 8B/10B decoder (if enabled).

Depending on whether you use the byte deserializer or not, the parallel recovered clock (when you do not use the byte deserializer) or a divide-by-two version of the parallel recovered clock (when you use the byte deserializer) clocks the write port of the receiver phase compensation FIFO. This clock is driven directly on the `rx_clkout` port as the FPGA fabric-Transceiver interface clock. You can use the `rx_clkout` signal to capture the receiver data and status signals in the FPGA fabric.

Table 2–9 lists the receiver datapath clock frequencies in non-bonded functional modes without rate matcher.

Table 2–9. Receiver Datapath Clock Frequencies in Non-Bonded Functional Modes Without Rate Matcher

Functional Mode	Data Rate	Serial Recovered Clock Frequency	Parallel Recovered Clock Frequency (MHz)	FPGA Fabric-Transceiver Interface Clock Frequency	
				Without Byte Deserializer (MHz)	With Byte Deserializer (MHz)
SONET/SDH OC12	622 Mbps	311 MHz	77.75	77.75	N/A
SONET/SDH OC48	2.488 Gbps	1.244 GHz	311	N/A	155.5
HD-SDI	1.485 Gbps	742.5 MHz	148.5	148.5	74.25
	1.4835 Gbps	741.75 MHz	148.35	148.35	74.175
3G-SDI	2.97 Gbps	1.485 GHz	297	N/A	148.5
	2.967 Gbps	1.4835 GHz	296.7	N/A	148.35

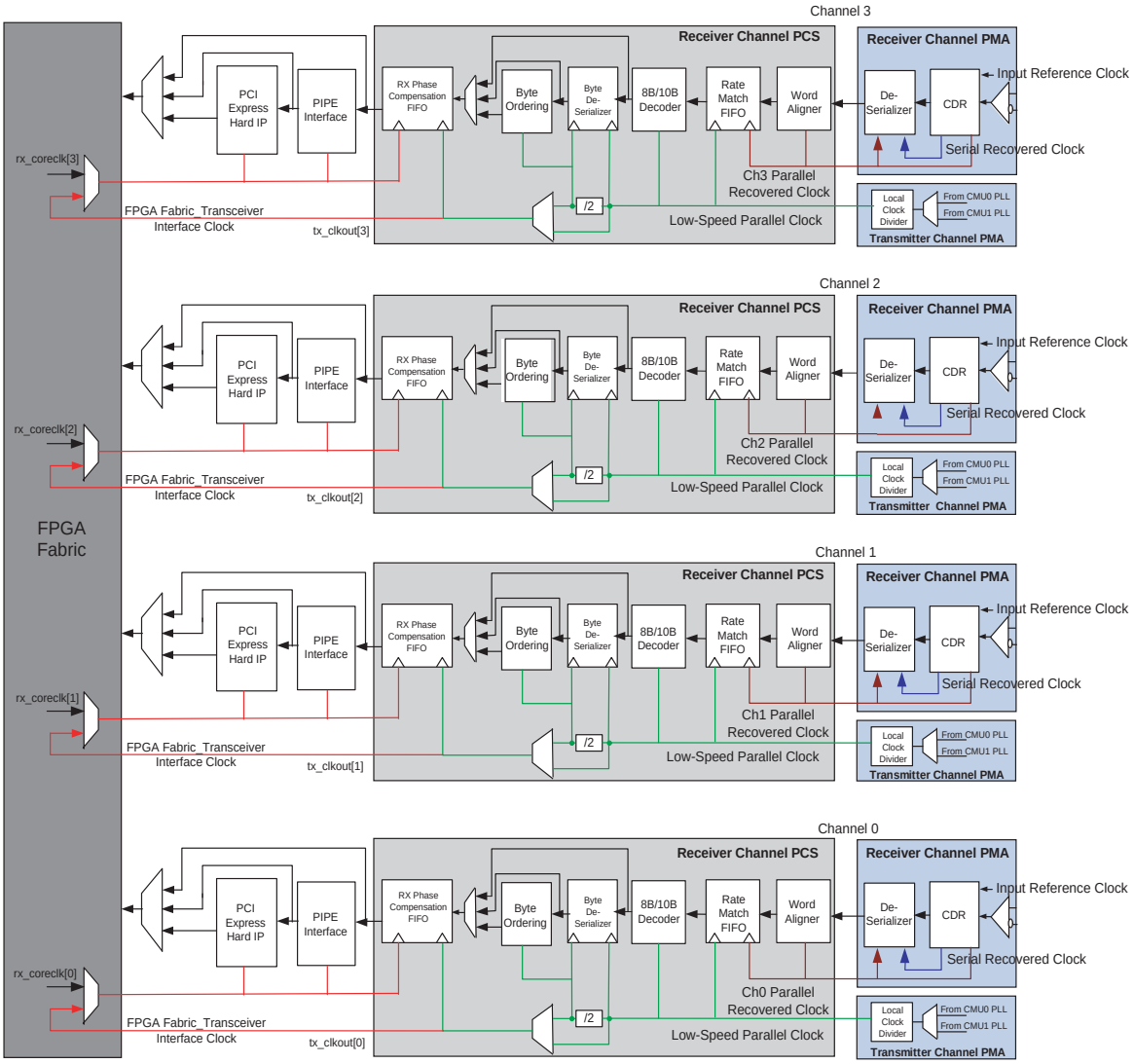
Non-Bonded Receiver Clocking with Rate Matcher

The following functional modes have non-bonded receiver channel configuration with rate-matcher:

- PCI Express (PIPE) ×1
- GIGE
- Serial RapidIO
- Basic with rate matcher

Figure 2–24 shows the receiver datapath clocking in non-bonded channel configurations with rate matcher.

Figure 2–24. Receiver Datapath Clocking in Non-Bonded Configurations with Rate Matcher *(Note 1)*



Note to Figure 2–24:

(1) The red lines represent the FPGA fabric-Transceiver interface clock, the green lines represent the low-speed parallel clock, the dark red lines represent the parallel recovered clock, and the blue lines represent the serial recovered clock.

In non-bonded configurations with rate matcher, the CDR in each receiver channel recovers the serial clock from the received data. The serial recovered clock is divided within the receiver PMA to generate the parallel recovered clock. The deserializer uses the serial recovered clock in the receiver PMA. The parallel recovered clock and deserialized data is forwarded to the receiver PCS.

The parallel recovered clock from the receiver PMA in each channel clocks the word aligner and the write port of the rate match FIFO. The low-speed parallel clock from the transmitter local clock divider block in each channel clocks the read port of the rate match FIFO, the 8B/10B decoder, and the write port of the byte deserializer (if enabled). The parallel transmitter PCS clock or its divide-by-two version (if byte deserializer is enabled) clocks the write port of the receiver phase compensation FIFO. It is also driven on the `tx_clkout` port as the FPGA fabric-Transceiver interface clock. You can use the `tx_clkout` signal to latch the receiver data and status signals in the FPGA fabric.

Table 2–10 lists the receiver datapath clock frequencies in non-bonded functional modes with rate matcher.

Table 2–10. Receiver Datapath Clock Frequencies in Non-Bonded Functional Modes with Rate Matcher

Functional Mode	Data Rate	Serial Recovered Clock Frequency	Parallel Recovered Clock and Parallel Transmitter PCS Clock Frequency (MHz)	FPGA Fabric-Transceiver Interface Clock Frequency	
				Without Byte Deserializer (MHz)	With Byte Deserializer (MHz)
PCI Express (PIPE) ×1 (Gen 1)	2.5 Gbps	1.25 GHz	250	250	125
PCI Express (PIPE) ×1 (Gen 2)	5 Gbps	2.5 GHz	500	N/A	250
GIGE	1.25 Gbps	625 MHz	125	125	N/A
Serial RapidIO	1.25 Gbps	625 MHz	125	N/A	62.5
	2.5 Gbps	1.25 GHz	250	N/A	125
	3.125 Gbps	1.5625 GHz	312.5	N/A	156.25

Bonded Channel Configurations

The Stratix IV device supports ×4 channel bonding that allows bonding of four channels within the same transceiver block. It also supports ×8 channel bonding that allows bonding of eight channels across two transceiver blocks on the same side of the device.

In bonded channel configurations, the low-speed parallel clock for all bonded channels are generated by the same `CMU0` clock divider or the ATX clock divider block, resulting in lower channel-to-channel clock skew. The receiver phase compensation FIFO in all bonded channels (except in Basic [PMA Direct] ×N mode) share common pointers and control logic generated in the central control unit, resulting in equal latency in the receiver phase compensation FIFO of all bonded channels.

☞ Bonding is not supported on the receive side for Basic ×4 and Basic ×8 functional modes. If you use rate matcher, the clocking scheme for Basic ×4 and Basic ×8 functional modes, the clocking is similar to PCI Express (PIPE) ×4 mode, as shown in Figure 2–26 on page 2–44 and PCI Express (PIPE) ×8 mode, as shown in Figure 2–27 on page 2–46.

×4 Bonded Channel Configuration

The following functional modes support ×4 receiver channel bonded configuration:

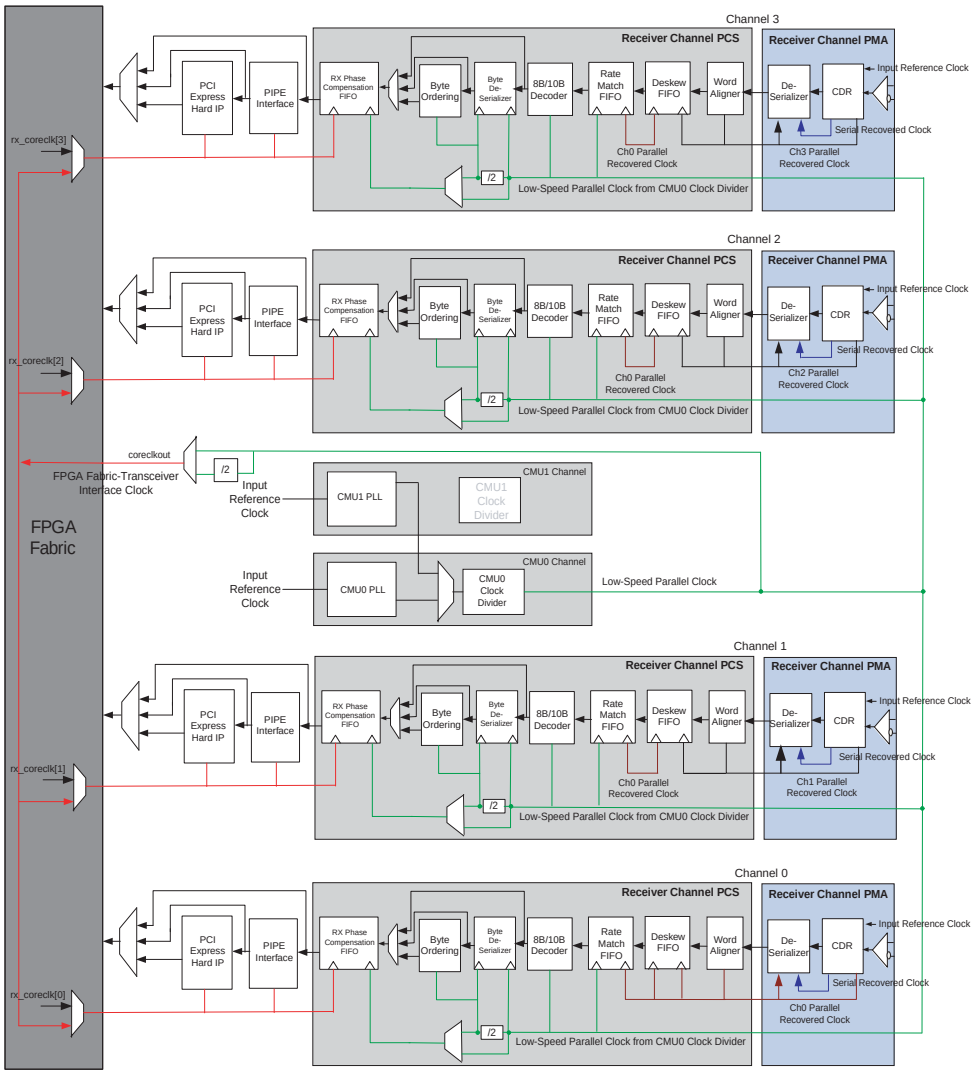
- XAUI ("x4 Bonded Channel Configuration with Deskew FIFO" on page 2–42)
- PCI Express (PIPE) ("x4 Bonded Channel Configuration Without Deskew FIFO" on page 2–44)

x4 Bonded Channel Configuration with Deskew FIFO

XAUI functional mode has ×4 bonded channel configuration with deskew FIFO.

Figure 2–25 shows the receiver datapath clocking in ×4 channel bonding configurations with deskew FIFO.

Figure 2–25. Receiver Datapath Clocking in x4 Bonded Channel Configuration with Deskew FIFO *(Note 1)*



Note to Figure 2–25:

(1) The red lines represent the FPGA fabric-Transceiver interface clock, the green lines represent the low-speed parallel clock, the dark red lines represent the Ch0 parallel recovered clock, and the blue lines represent the serial recovered clock.

In ×4 bonded channel configurations with deskew FIFO, the CDR in each receiver channel recovers the serial clock from the received data. The serial recovered clock is divided within each channel's receiver PMA to generate the parallel recovered clock. The deserializer uses the serial recovered clock in the receiver PMA. The parallel recovered clock and deserialized data is forwarded to the receiver PCS in each channel.

The parallel recovered clock from the receiver PMA in each channel clocks the word aligner in that channel. The parallel recovered clock from `Channel 0` clocks the deskew FIFO and the write port of the rate match FIFO in all four bonded channels. The low-speed parallel clock from the `CMU0` clock divider block in `CMU0_Channel` clocks the read port of the rate match FIFO, the 8B/10B decoder, and the write port of the byte deserializer (if enabled) in all four bonded channels. The low-speed parallel clock or its divide-by-two version (if byte deserializer is enabled) clocks the write port of the receiver phase compensation FIFO. It is also driven on the `coreclkout` port as the FPGA fabric-Transceiver interface clock. You can use the `coreclkout` signal to latch the receiver data and status signals in the FPGA fabric for all four bonded channels.

Table 2–11 lists the receiver datapath clock frequencies in ×4 bonded functional modes with deskew FIFO.

Table 2–11. Receiver Datapath Clock Frequencies in x4 Bonded Functional Modes with Deskew FIFO

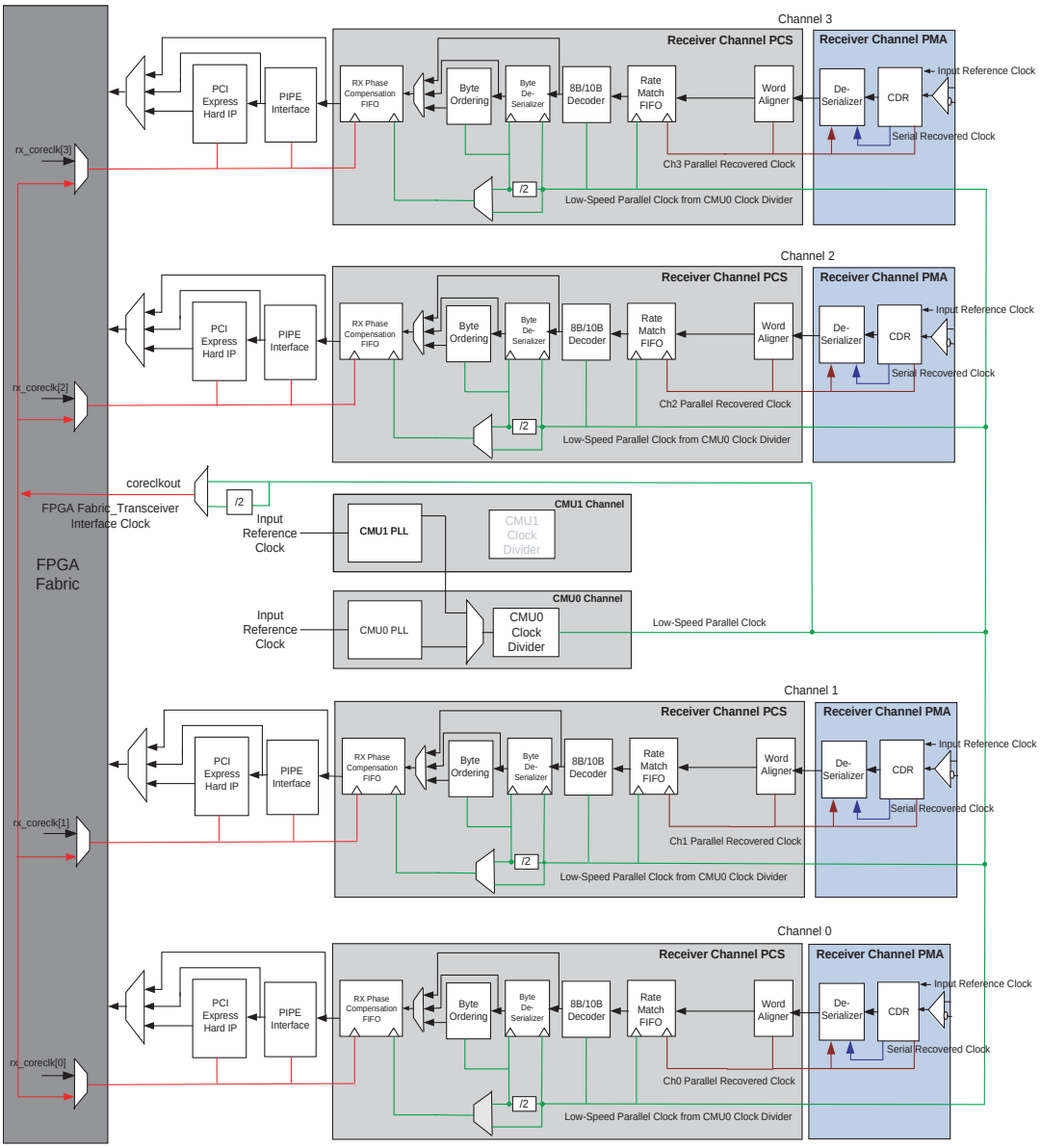
Functional Mode	Data Rate	Serial Recovered Clock Frequency	Parallel Recovered Clock and Parallel Transmitter PCS Clock Frequency (MHz)	FPGA-Fabric Transceiver Interface Clock Frequency	
				Without Byte Deserializer (MHz)	With Byte Deserializer (MHz)
PCI Express (PIPE) ×4 (Gen 1)	2.5 Gbps	1.25 GHz	250	250	125
PCI Express (PIPE) ×4 (Gen 2)	5 Gbps	2.5 GHz	500	N/A	250
XAUI	3.125 Gbps	1.5625 MHz	312.5	N/A	156.25

x4 Bonded Channel Configuration Without Deskew FIFO

PCI Express (PIPE) ×4 functional modes have ×4 bonded channel configuration without deskew FIFO.

Figure 2–26 shows the receiver datapath clocking in ×4 channel bonding configurations without deskew FIFO.

Figure 2–26. Receiver Datapath Clocking in x4 Bonded Channel Configuration Without Deskew FIFO *(Note 1)*



Note to Figure 2–26:

(1) The red lines represent the FPGA fabric-Transceiver interface clock, the green lines represent the low-speed parallel clock, the dark red lines represent the parallel recovered clock, and the blue lines represent the serial recovered clock.

In ×4 bonded channel configurations without deskew FIFO, the CDR in each receiver channel recovers the serial clock from the received data. The serial recovered clock is divided within each channel's receiver PMA to generate the parallel recovered clock. The deserializer uses the serial recovered clock in the receiver PMA. The parallel recovered clock and deserialized data is forwarded to the receiver PCS in each channel.

The parallel recovered clock from the receiver PMA in each channel clocks the word aligner and the write side of the rate matcher FIFO in that channel. The low-speed parallel clock from the `CMU0` clock divider block in `CMU0_Channel` clocks the read port of the rate match FIFO, the 8B/10B decoder, and the write port of the byte deserializer (if enabled). The low-speed parallel clock or its divide-by-two version (if byte deserializer is enabled) clocks the receiver phase compensation FIFO. It is also driven on the `coreclkout` port as the FPGA fabric-Transceiver interface clock. You can use the `coreclkout` signal to latch the receiver data and status signals in the FPGA fabric for all four bonded channels.

Table 2–12 lists the receiver datapath clock frequencies in ×4 bonded functional modes without deskew FIFO.

Table 2–12. Receiver Datapath Clock Frequencies in x4 Bonded Functional Modes without Deskew FIFO

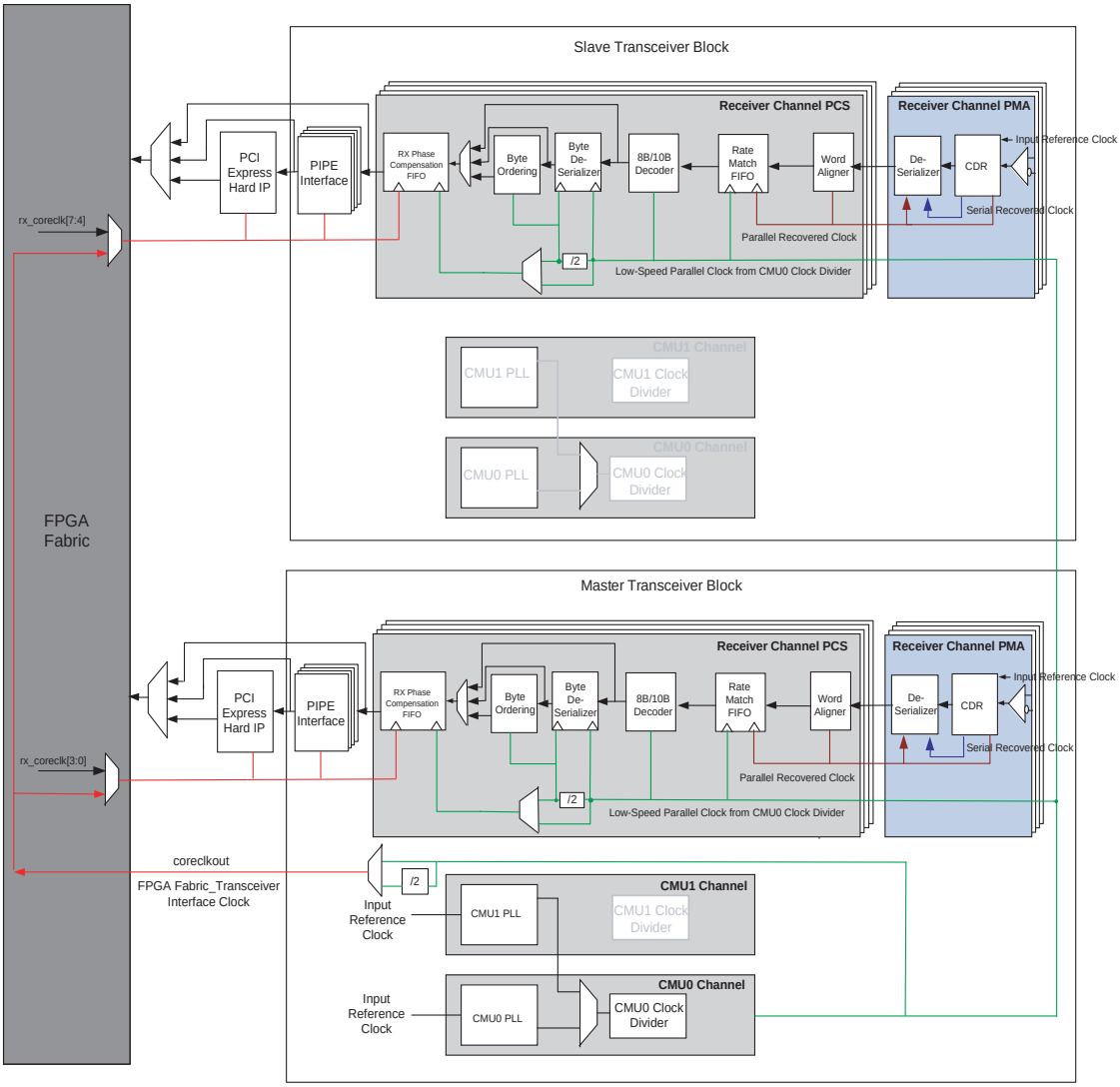
Functional Mode	Data Rate	Serial Recovered Clock Frequency	Parallel Recovered Clock and Parallel Transmitter PCS Clock Frequency (MHz)	FPGA Fabric-Transceiver Interface Clock Frequency	
				Without Byte Deserializer (MHz)	With Byte Deserializer (MHz)
PCI Express (PIPE) ×4 (Gen 1)	2.5 Gbps	1.25 GHz	250	250	125
PCI Express (PIPE) ×4 (Gen 2)	5 Gbps	2.5 GHz	500	N/A	250

x8 Bonded Channel Configuration

PCI Express (PIPE) ×8 functional mode supports the ×8 receiver channel bonding configuration. The eight bonded channels are located in two transceiver blocks, referred to as the master transceiver block and slave transceiver block, with four channels each.

Figure 2–27 shows the receiver datapath clocking in PCI Express (PIPE) ×8 bonded channel configuration.

Figure 2–27. Receiver Datapath Clocking in x8 Bonded Channel Configuration *(Note 1)*



Note to Figure 2–27:

(1) The red lines represent the FPGA fabric-Transceiver interface clock, the green lines represent the low-speed parallel clock, the dark red lines represent the parallel recovered clock, and the blue lines represent the serial recovered clock.

The CDR in each of the eight receiver channels recovers the serial clock from the received data on that channel. The serial recovered clock is divided within each channel's receiver PMA to generate the parallel recovered clock. The deserializer uses the serial recovered clock in the receiver PMA. The parallel recovered clock and deserialized data from the receiver PMA in each channel is forwarded to the receiver PCS in that channel.

The parallel recovered clock from the receiver PMA in each channel clocks the word aligner and the write side of the rate matcher FIFO in that channel. The low-speed parallel clock from the `CMU0` clock divider of the master transceiver block clocks the read port of the rate match FIFO, the 8B/10B decoder, and the write port of the byte deserializer (if enabled) in all eight channels. The low-speed parallel clock or its divide-by-two version (if byte deserializer is enabled) clocks the write port of the receiver phase compensation FIFO in all eight channels. It is also driven on the `coreclkout` port as the FPGA fabric-Transceiver interface clock. You can use the `coreclkout` signal to latch the receiver data and status signals in the FPGA fabric for all eight bonded channels.

Table 2–13 lists the receiver datapath clock frequencies in PCI Express (PIPE) ×8 functional mode.

Table 2–13. Receiver Datapath Clock Frequencies PCI Express (PIPE) x8 Functional Mode

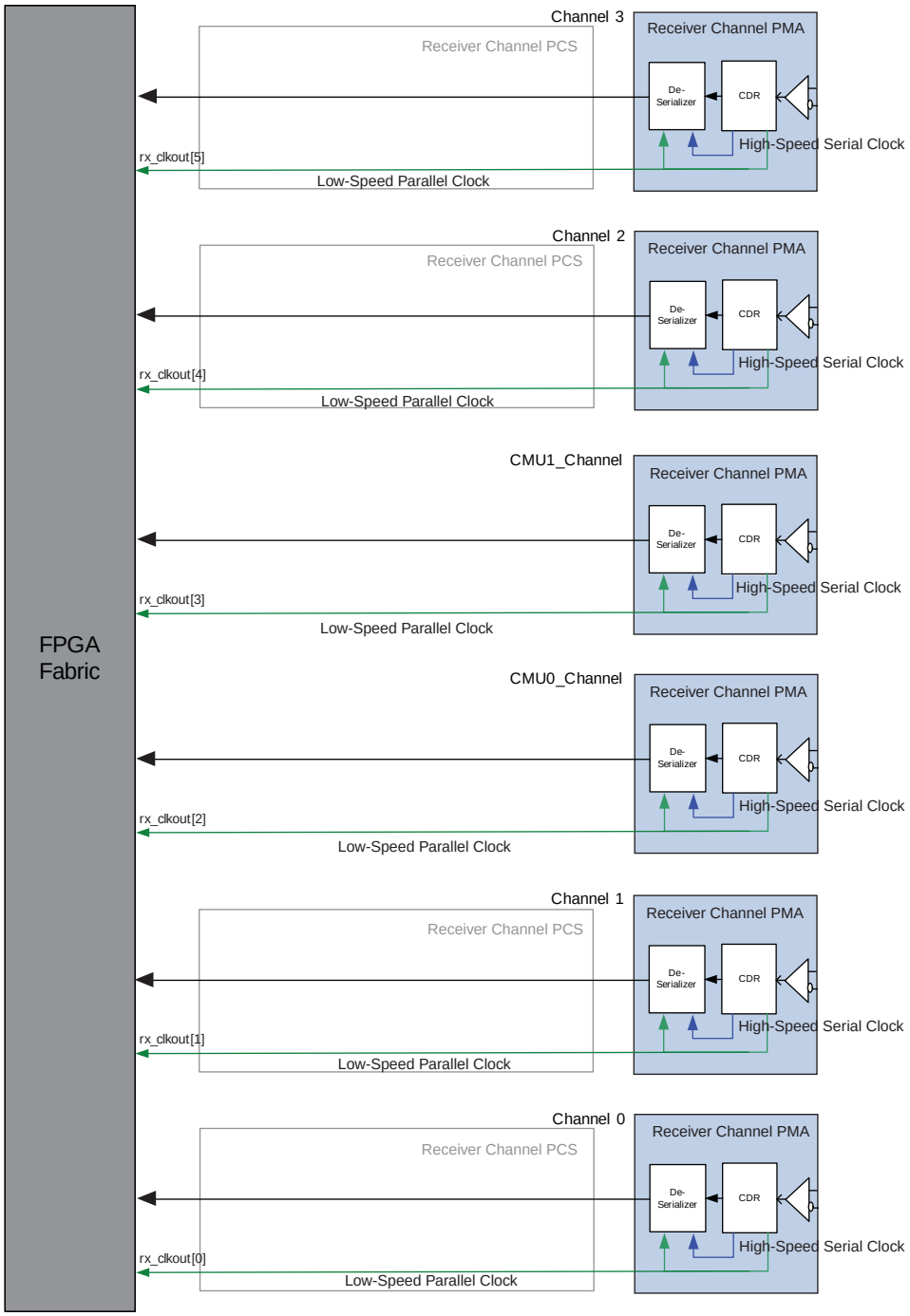
Functional Mode	Data Rate	Serial Recovered Clock Frequency	Parallel Recovered Clock and Parallel Transmitter PCS Clock Frequency (MHz)	FPGA Fabric-Transceiver Interface Clock Frequency	
				Without Byte Deserializer (MHz)	With Byte Deserializer (MHz)
PCI Express (PIPE) ×8 (Gen 1)	2.5 Gbps	1.25 GHz	250	250	125
PCI Express (PIPE) ×8 (Gen 2)	5 Gbps	2.5 GHz	500	N/A	250

Basic (PMA Direct) Mode Channel Configurations

Figure 2–28 shows six channels in a transceiver block configured in Basic (PMA Direct) functional mode with two of the channels being CMU channels. The receiver channel PMA directly interfaces to the user logic in the FPGA fabric. The CDR recovers the high-speed serial clock and low-speed parallel clock for the deserializer. The low-speed parallel clock is forwarded to the FPGA fabric as `rx_clkout`.

☞ Bonded mode is not available for the receivers configured in Basic (PMA Direct) functional mode. Data registers to capture the receiver data in the FPGA fabric for each channel must be clocked by `rx_clkout` forwarded by that channel's CDR.

Figure 2–28. Receiver Channel PMA Directly Interfacing to the User Logic in the FPGA Fabric *(Note 1)*



Note to Figure 2–28:

(1) The green lines represent the low-speed parallel clock and the blue lines represent the serial recovered clock.

FPGA Fabric-Transceiver Interface Clocking

The FPGA fabric-transceiver interface clocks consist of clock signals from the FPGA fabric to the transceiver blocks and clock signals from the transceiver blocks to the FPGA fabric. These clock resources use the clock networks in the FPGA core that include the global, regional, and periphery clock networks.

The FPGA fabric-transceiver interface clocks can be subdivided into the following three categories:

- **Input Reference Clocks**—Refer to "Input Reference Clock Source" on page 2–3.
- **Transceiver Datapath Clocks**—are used to transfer data, control, and status signals between the FPGA fabric and the transceiver channels. The transceiver channel forwards the `tx_clkout` signal (in non-bonded modes) or the `coreclkout` signal (in bonded channel modes) to the FPGA fabric to clock the data and control signals into the transmitter. The transceiver channel also forwards the recovered `rx_clkout` clock (in configurations without rate matcher) or `tx_clkout/coreclkout` (in configurations with rate matcher) to the FPGA fabric to clock the data and status signals from the receiver into the FPGA fabric.
- **Other Transceiver Clocks**—The following transceiver clocks form a part of the FPGA fabric-Transceiver interface clocks:
  - `cal_blk_clk`—calibration block clock
  - `fixed_clk`—125 MHz fixed-rate clock used in the PCI Express (PIPE) receiver detect circuitry and for the adaptive equalization (AEQ) block
  - `reconfig_clk`—clock used for transceiver dynamic reconfiguration (for more information, refer to Table 2–5 on page 2–9)

☞ In Basic (PMA Direct) functional mode, only `tx_clkout` and `rx_clkout` are available to clock the logic in the core. In bonded mode, you may use `tx_clkout` of one of the channels to clock all of the channels. For receivers in bonded mode, you must use separate `rx_clkout` for each channel.

Table 2–14 lists the FPGA fabric-Transceiver interface clocks.

Table 2–14. FPGA Fabric-Transceiver Interface Clocks *(Note 1)* (Part 1 of 2)

Clock Name	Clock Description	Interface Direction	FPGA Fabric Clock Resource Utilization *(1)*
`pll_inclk`	CMU PLL input reference clock when driven from an FPGA CLK input pin	FPGA fabric-to-transceiver	GCLK
`rx_cruclk`	Receiver CDR input reference clock when driven from an FPGA CLK input pin	FPGA fabric-to-transceiver	GCLK, RCLK
`tx_clkout`	Phase compensation FIFO clock	Transceiver-to-FPGA fabric	GCLK, RCLK, PCLK
`coreclkout`	Phase compensation FIFO clock	Transceiver-to-FPGA fabric	GCLK, RCLK, PCLK
`rx_clkout`	Phase compensation FIFO clock	Transceiver-to-FPGA fabric	GCLK, RCLK, PCLK
`fixed_clk`	PCI Express (PIPE) receiver detect clock	FPGA fabric-to-transceiver	GCLK, RCLK
`reconfig_clk` *(2)*	Transceiver dynamic reconfiguration clock	FPGA fabric-to-transceiver	GCLK

Table 2–14. FPGA Fabric-Transceiver Interface Clocks *(Note 1)* (Part 2 of 2)

Clock Name	Clock Description	Interface Direction	FPGA Fabric Clock Resource Utilization *(1)*
`cal_blk_clk`	Transceiver calibration block clock	FPGA fabric-to-transceiver	GCLK, RCLK

Note to Table 2–11:

(1) For more information about GCLK, RCLK, and PCLK resources available in each device, refer to the *Clock Networks and PLLs in Stratix IV Devices* chapter.

(2) Ensure that the `reconfig_clk` is a free-running clock that is not derived from the transceiver blocks.

"FPGA Fabric-Transmitter Interface Clocking" on page 2–50 and "FPGA Fabric-Receiver Interface Clocking" on page 2–59 describe the criteria and methodology to share transmitter and receiver phase compensation FIFO clocks in order to reduce the GCLK, RCLK, and PCLK resource utilization in your design.

FPGA Fabric-Transmitter Interface Clocking

The transmitter phase compensation FIFO compensates for the phase difference between the FPGA fabric clock (phase compensation FIFO write clock) and the parallel transmitter PCS clock (phase compensation FIFO read clock). The transmitter phase compensation FIFO write clock forms the FPGA fabric-Transmitter interface clock. The phase compensation FIFO write clock and read clocks must have exactly the same frequency (0 parts-per-million [PPM] frequency difference).

Stratix IV transceivers provide the following two options for selecting the transmitter phase compensation FIFO write clock:

- "Quartus II-Selected Transmitter Phase Compensation FIFO Write Clock" on page 2–50
- "User-Selected Transmitter Phase Compensation FIFO Write Clock" on page 2–56

☞ User-selection is provided to share transceiver datapath clocks in order to reduce the GCLK, RCLK, and PCLK resource utilization in your design.

Quartus II-Selected Transmitter Phase Compensation FIFO Write Clock

If you do not select the `tx_coreclk` port in the ALTGX MegaWizard™ Plug-In Manager, the Quartus II software automatically selects the transmitter phase compensation FIFO write clock for each channel in that ALTGX instance. The Quartus II software selects the FIFO write clock depending on the channel configuration.

Non-Bonded Channel Configuration

In a non-bonded channel configuration, the transmitter channels may or may not be identical. Identical transmitter channels are defined as channels that have exactly the same CMU PLL input reference clock source, exactly the same CMU PLL configuration, and exactly the same transmitter PMA and PCS configuration.

☞ Identical transmitter channels may have different transmitter voltage output differential (V_{OD}), transmitter common mode voltage (V_{CM}), or pre-emphasis setting.

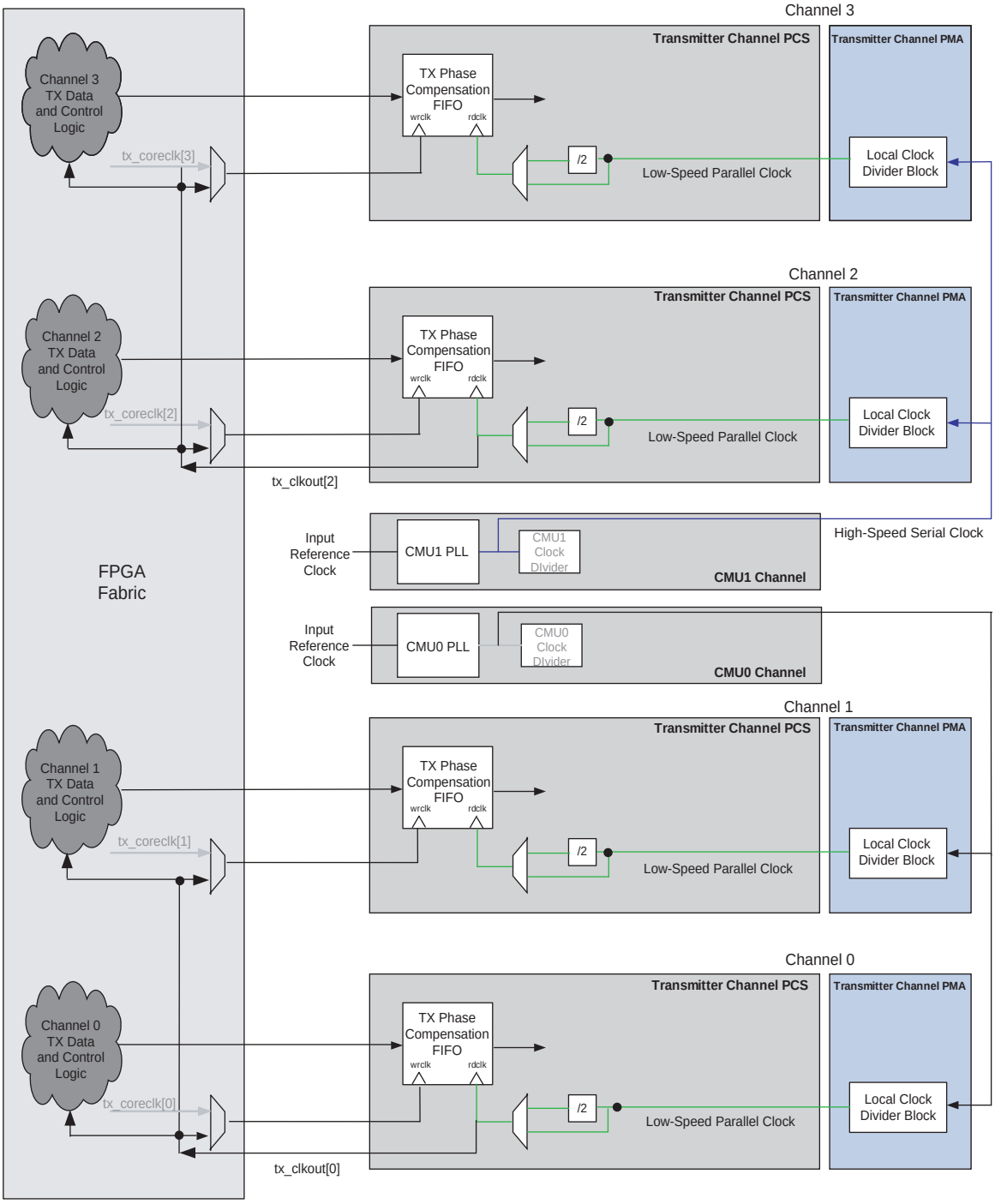
Example 3: Two Groups of Two Identical Channels in a Transceiver Block

Example 3 assumes channels 0 and 1, driven by `CMU0_PLL` in a transceiver block, are identical. Also, channels 2 and 3, driven by `CMU1_PLL` in the same transceiver block, are identical. In this case, the Quartus II software automatically drives the write port of the transmitter phase compensation FIFO in channels 0 and 1 with the `tx_clkout[0]` signal. It also drives the write port of the transmitter phase compensation FIFO in channels 2 and 3 with the `tx_clkout[2]` signal. Use the `tx_clkout[0]` signal to clock the transmitter data and control logic for channels 0 and 1 in the FPGA fabric. Use the `tx_clkout[2]` signal to clock the transmitter data and control logic for channels 2 and 3 in the FPGA fabric.

☞ This configuration uses two FPGA global and/or regional clock resources, one for the `tx_clkout[0]` signal and the other for the `tx_clkout[2]` signal.

Figure 2–29 shows the FPGA fabric-Transmitter interface clocking for Example 3.

Figure 2–29. FPGA Fabric-Transmitter Interface Clocking for Example 3 *(Note 1)*



Note to Figure 2–29:

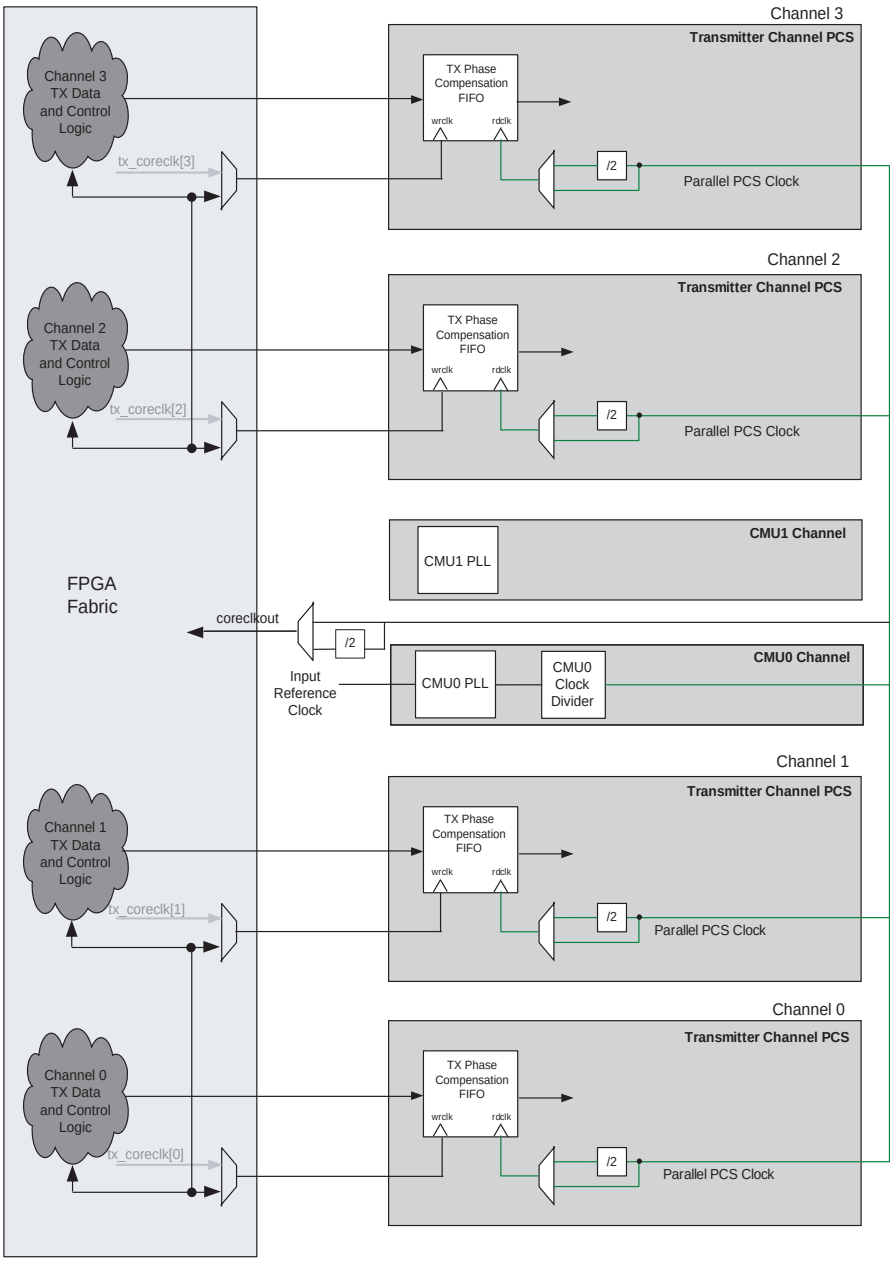
(1) The green lines represent the low-speed parallel clock and the blue lines represent the high-speed serial clock.

Bonded Channel Configuration

In ×4 and ×8 bonded channel configurations, all channels within the transceiver block are identical. The Quartus II software automatically drives the write port of the transmitter phase compensation FIFO in all channels with the `coreclkout` signal. Use the `coreclkout` signal to clock the transmitter data and control logic for all four channels in the FPGA fabric.

Figure 2–30 shows the FPGA fabric-Transmitter interface clocking in a ×4 bonded channel configuration.

Figure 2–30. FPGA Fabric-Transmitter Interface Clocking in a x4 Bonded Channel Configuration *(Note 1)*



Note to Figure 2–30:

(1) The green lines represent the parallel PCS clock.

Limitations of the Quartus II Software-Selected Transmitter Phase Compensation FIFO Write Clock

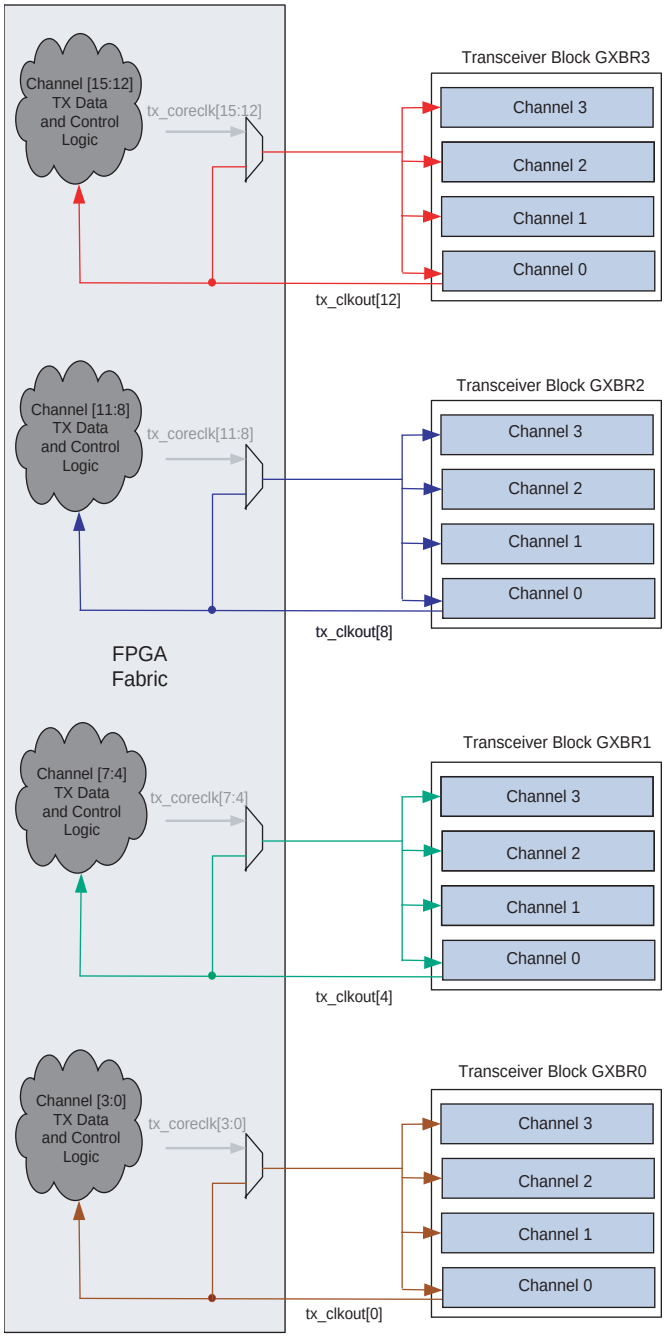
The Quartus II software uses a single `tx_clkout` signal to clock the transmitter phase compensation FIFO write port of all identical channels within a transceiver block. This results in one global and/or regional clock resource being used for each group of identical channels within a transceiver block.

For identical channels located across the transceiver blocks, the Quartus II software does not use a single `tx_clkout` signal to clock the write port of the transmitter phase compensation FIFOs for all channels. It uses one `tx_clkout` signal for each group of identical channels per transceiver block. This results in higher global and regional clock resource utilization.

Example 4: Sixteen Identical Channels Across Four Transceiver Blocks

Figure 2–31 shows 16 identical transmitter channels located across four transceiver blocks. The Quartus II software uses `tx_clkout` from Channel 0 in each transceiver block to clock the write port of the transmitter phase compensation FIFO in all four channels in that transceiver block. This results in four global and/or regional clock resources being used, one for each transceiver block.

Figure 2–31. Sixteen Identical Channels Across Four Transceiver Blocks for Example 4 *(Note 1)*



Note to Figure 2–31:

(1) The red lines represent `tx_clkout[12]`, the blue lines represent `tx_clkout[8]`, the green lines represent `tx_clkout[4]`, and the brown lines represent `tx_clkout[0]`.

Because all 16 channels are identical, using a single `tx_clkout` to clock the transmitter phase compensation FIFO in all 16 channels results in only one global or regional clock resource being used instead of four. To achieve this, you must choose the transmitter phase compensation FIFO write clocks instead of the Quartus II software automatic selection, as described in "User-Selected Transmitter Phase Compensation FIFO Write Clock" on page 2–56.

User-Selected Transmitter Phase Compensation FIFO Write Clock

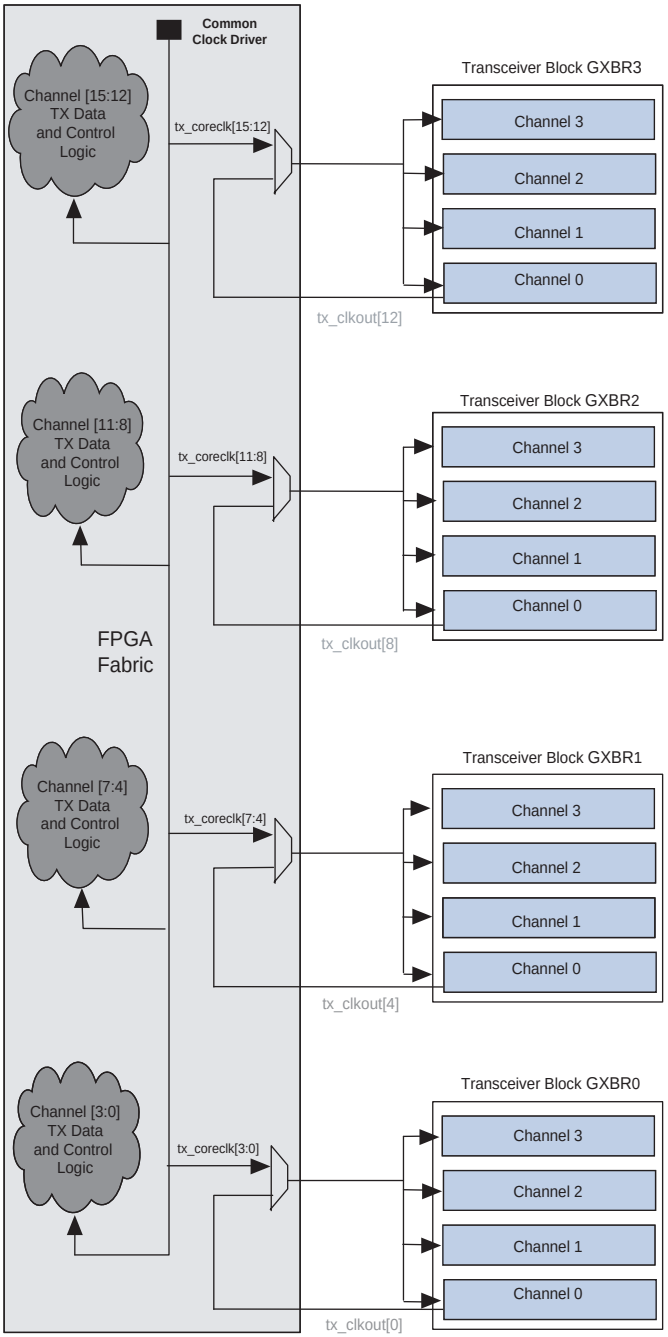
The ALTGX MegaWizard Plug-In Manager provides an optional port named `tx_coreclk` for each instantiated transmitter channel. If you enable this port, the Quartus II software does not automatically select the transmitter phase compensation FIFO write clock source. Instead, the signal that you drive on the `tx_coreclk` port of the channel clocks the write side of its transmitter phase compensation FIFO.

Use the flexibility of selecting the transmitter phase compensation FIFO write clock to reduce global and regional clock resource utilization. You can connect the `tx_coreclk` ports of all identical channels in your design and drive them using a common clock driver that has 0 PPM frequency difference with respect to the FIFO read clocks of these channels. Use the common clock driver to clock the transmitter data and control logic in the FPGA fabric for all identical channels. This FPGA fabric-Transceiver interface clocking scheme uses only one global or regional clock resource for all identical channels in your design.

Example 5: Sixteen Identical Channels Across Four Transceiver Blocks

Figure 2–32 shows 16 identical transmitter channels located across four transceiver blocks. The `tx_coreclk` ports of all 16 transmitter channels are connected together and driven by a common clock driver. This common clock driver also drives the transmitter data and control logic of all 16 transmitter channels in the FPGA fabric. You use only one global or regional clock resource with this clocking scheme, compared to four global and regional clock resources needed without the `tx_coreclk` ports (the Quartus II software-selected transmitter phase compensation FIFO write clock).

Figure 2–32. Sixteen Identical Channels Across Four Transceiver Blocks for Example 5



Common Clock Driver Selection Rules

The common clock driver driving the `tx_coreclk` ports of all identical channels must have 0 PPM frequency difference with respect to the transmitter phase compensation FIFO read clocks of these channels. If there is any frequency difference between the FIFO write clock (`tx_coreclk`) and the FIFO read clock, the FIFO overflows or under-runs, resulting in corrupted data transfer between the FPGA fabric and the transmitter.

Table 2–15 lists the transmitter phase compensation FIFO read clocks that the Quartus II software selects in various configurations.

Table 2–15. Transmitter Phase Compensation FIFO Read Clocks

Configuration	Transmitter Phase Compensation FIFO Read Clock	
	Without Byte Serializer	**With Byte Serializer**
Non-Bonded Channel Configuration	Parallel transmitter PCS clock from the local clock divider in the associated channel (`tx_clkout`)	Divide-by-two version of the parallel transmitter PCS clock from the local clock divider in the associated channel (`tx_clkout`)
×4 Bonded Channel Configuration	Low-speed parallel clock from the `CMU0` clock divider of the associated transceiver block (`coreclkout`)	Divide-by-two version of the low-speed parallel clock from the `CMU0` clock divider of the associated transceiver block (`coreclkout`)
×8 Bonded Channel Configuration	Low-speed parallel clock from the `CMU0` clock divider of the master transceiver block (`coreclkout` from master transceiver block)	Divide-by-two version of the low-speed parallel clock from the `CMU0` clock divider of the master transceiver block (`coreclkout` from master transceiver block)

To ensure that you understand the 0 PPM clock driver rule, the Quartus II software expects the following set of user assignments whenever you use the `tx_coreclk` port to drive the transmitter phase compensation FIFO write clock:

- **GXB 0 PPM Core Clock Setting**

Failing to make this assignment correctly when using the `tx_coreclk` port results in a Quartus II compilation error.

The **GXB 0 PPM core clock** setting allows the following clock drivers to drive the `tx_coreclk` ports:

- `tx_clkout` in non-bonded channel configurations
- `coreclkout` in bonded channel configurations
- `FPGA_CLK` input pins
- Transceiver `refclk` pins
- Clock output from left and right and top and bottom PLLs (`PLL_L`, `PLL_R`, and `PLL_T`, `PLL_B`)

The Quartus II software does not allow gated clocks or clocks generated in FPGA logic to drive the `tx_coreclk` ports.

Because the **GXB 0 PPM core clock** setting allows the FPGA CLK input pins and transceiver `refclk` pins as the clock driver, the Quartus II compiler cannot determine if there is a 0 PPM difference between the FIFO write clock and read clock for each channel.

You must ensure that the clock driver for all connected `tx_coreclk` ports has a 0 PPM difference with respect to the FIFO read clock in those channels.

Table 2–16 lists the Quartus II assignments that you must make in the assignment editor.

Table 2–16. Quartus II Assignments

From	Full design hierarchy name of one of the following clock drivers that you choose to drive the `tx_coreclk` ports of all identical channels *(1)*: ■ `tx_clkout` ■ `coreclkout` ■ FPGA CLK input pins ■ Transceiver `refclk` pins ■ Clock output from the left and right or top and bottom PLLs ■ `tx_dataout` port of one of the identical channels
To	`tx_dataout` pins of all identical channels whose `tx_coreclk` ports are connected together and driven by the 0 PPM clock driver.
Assignment Name	GXB 0 PPM Core Clock Setting
Value	ON

Note to Table 2–13:

(1) You can find the full hierarchy name of the 0 PPM clock driver using the **Node Finder** feature in the Quartus II Assignment Editor.

For more implementation information, refer to "Configuration Example 2: Configuring Sixteen Identical Channels Across Four Transceiver Blocks" on page 2–74.

Basic (PMA Direct) mode

In Basic (PMA Direct) mode, each channel must be clocked by its own `tx_clkout`. As a result, the number of global and/or regional clock resources required is significantly higher. In Basic (PMA Direct) ×N mode, to save on global and/or regional clock resources, you may use `tx_clkout` from centrally located channels to clock all the channels. The `coreclkout` port is not available in Basic (PMA Direct) ×N mode.

FPGA Fabric-Receiver Interface Clocking

The receiver phase compensation FIFO compensates for the phase difference between the parallel receiver PCS clock (FIFO write clock) and the FPGA fabric clock (FIFO read clock). The receiver phase compensation FIFO read clock forms the FPGA fabric-Receiver interface clock. The FIFO write clock and read clock must have exactly the same frequency (0 PPM frequency difference).

Stratix IV transceivers provide the following two options for selecting the receiver phase compensation FIFO read clock:

- "Quartus II Software-Selected Receiver Phase Compensation FIFO Read Clock" on page 2–60
- "User-Selected Receiver Phase Compensation FIFO Read Clock" on page 2–67

User-selection is provided to share transceiver datapath clocks in order to reduce the GCLK, RCLK, and PCLK resource utilization in your design.

Quartus II Software-Selected Receiver Phase Compensation FIFO Read Clock

If you do not select the `rx_coreclk` port in the ALTGX MegaWizard Plug-In Manager, the Quartus II software automatically selects the receiver phase compensation FIFO read clock for each channel in that ALTGX instance. The Quartus II software selects the FIFO read clock depending on the channel configuration. In non-bonded channel configurations, the FPGA fabric-receiver interface clocking has two scenarios:

- Receivers the do not use a rate matcher block (refer to "Non-Bonded Receiver Clocking Without Rate Matcher" on page 2–37
- Receivers that use a rate matcher block (refer to "Non-Bonded Receiver Clocking with Rate Matcher" on page 2–39

Non-Bonded Channel Configuration with Rate Matcher

In non-bonded channel configuration, the transceiver channels may or may not be identical. Identical transceiver channels are defined as channels that have exactly the same CMU PLL and receiver CDR input reference clock sources, exactly the same CMU PLL and receiver CDR configuration, and exactly the same PMA and PCS configuration.

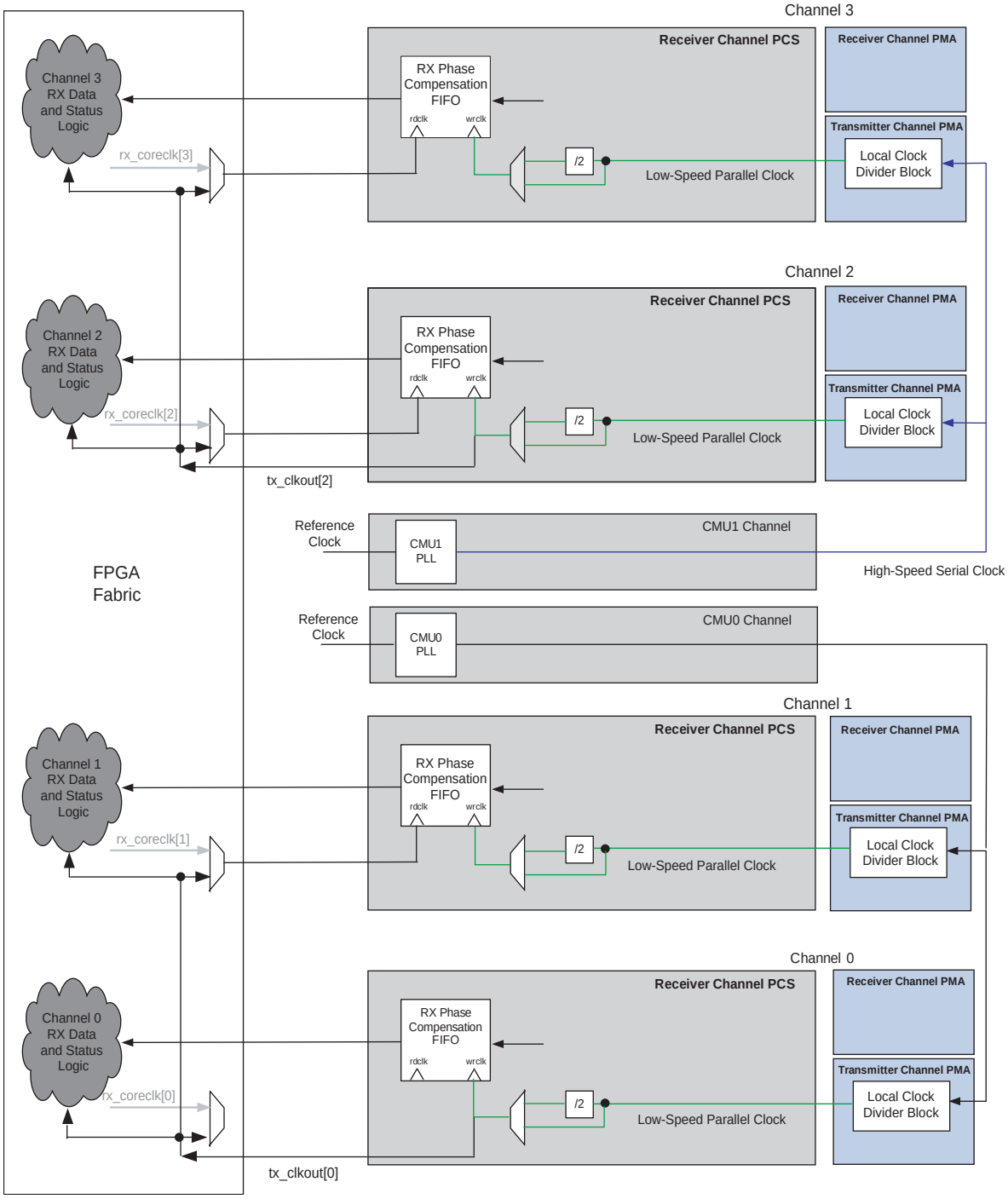
Example 6: Two Groups of Two Identical Channels in a Transceiver Block

Example 6 assumes channels 0 and 1, driven by the `CMU0` PLL in a transceiver block, are identical. Also, channels 2 and 3, driven by the `CMU1` PLL in the same transceiver block, are identical. In this case, the Quartus II software automatically drives the read port of the receiver phase compensation FIFO in channels 0 and 1 with the `tx_clkout[0]` signal. It also drives the read port of the receiver phase compensation FIFO in channels 2 and 3 with the `tx_clkout[2]` signal. Use the `tx_clkout[0]` signal to latch the receiver data and status signals from channels 0 and 1 in the FPGA fabric. Use the `tx_clkout[2]` signal to latch the receiver data and status signals from channels 2 and 3 in the FPGA fabric.

This configuration uses two FPGA global and/or regional clock resources, one for the `tx_clkout[0]` signal and the other for the `tx_clkout[2]` signal.

Figure 2–33 shows the FPGA fabric-Receiver interface clocking for Example 6.

Figure 2–33. FPGA Fabric-Receiver Interface Clocking for Example 6 *(Note 1)*



Note to Figure 2–33:

(1) The green lines represent the low-speed parallel clock and the blue lines represent the high-speed serial clock.

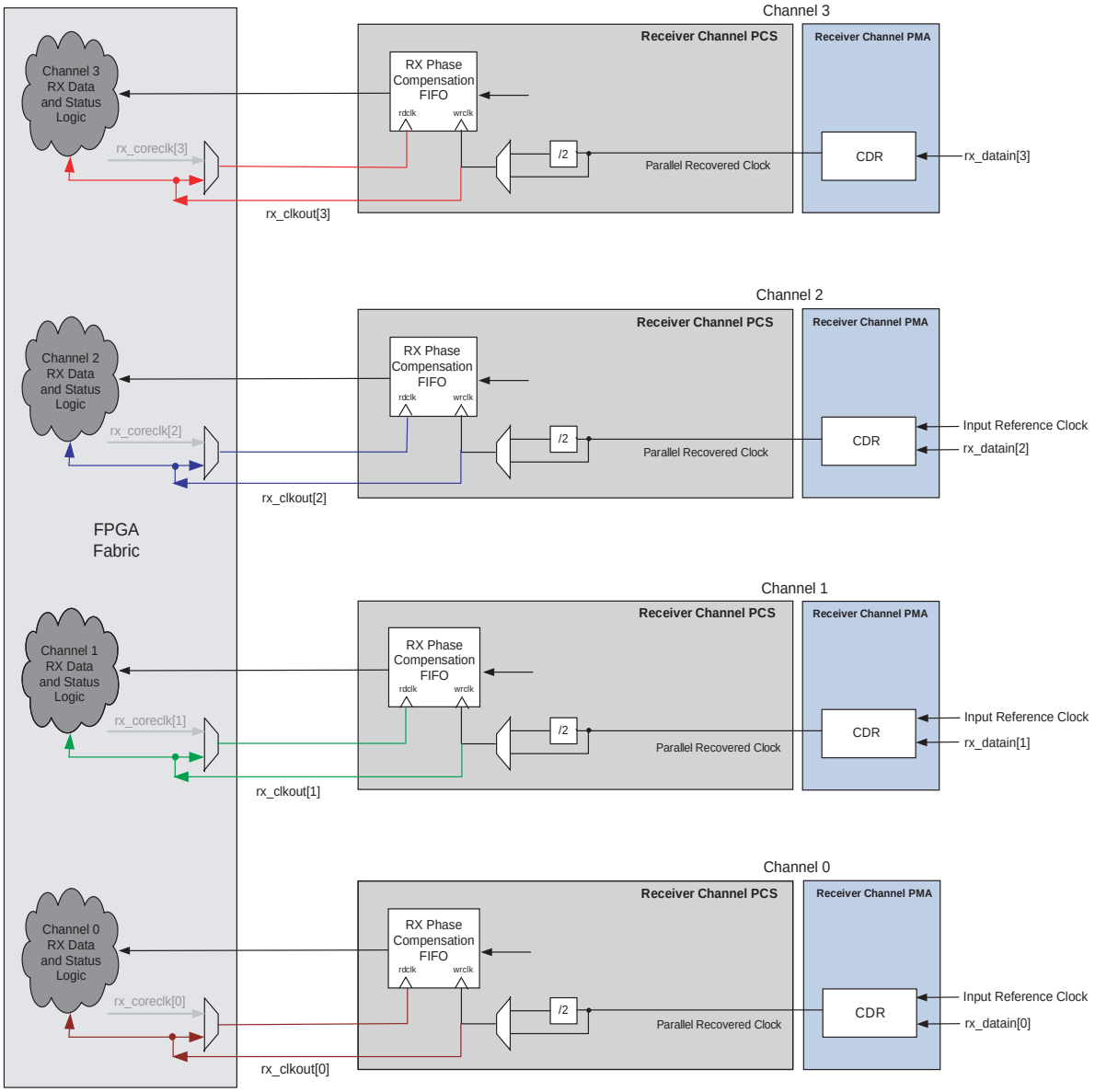
Non-Bonded Channel Configuration Without Rate Matcher

In non-bonded channel configuration without rate matcher, the Quartus II software cannot determine if the incoming serial data in all channels have a 0 PPM frequency difference. The Quartus II software automatically drives the read port of the receiver phase compensation FIFO in each channel with the recovered clock driven on the `rx_clkout` port of that channel. Use the `rx_clkout` signal from each channel to latch its receiver data and status signals in the FPGA fabric.

☞ This configuration uses one FPGA global, regional clock, or both, resource per channel for the `rx_clkout` signal.

Figure 2–34 shows the FPGA fabric-Receiver interface clocking for non-bonded channel configurations without rate matcher.

Figure 2–34. FPGA Fabric-Receiver Interface Clocking for Non-Bonded Channel Configurations Without Rate Matcher *(Note 1)*



Note to Figure 2–34:

(1) The red lines represent `rx_clkout[3]`, the blue lines represent `rx_clkout[2]`, the green lines represent `rx_clkout[1]`, and the brown lines represent `rx_clkout[0]`.

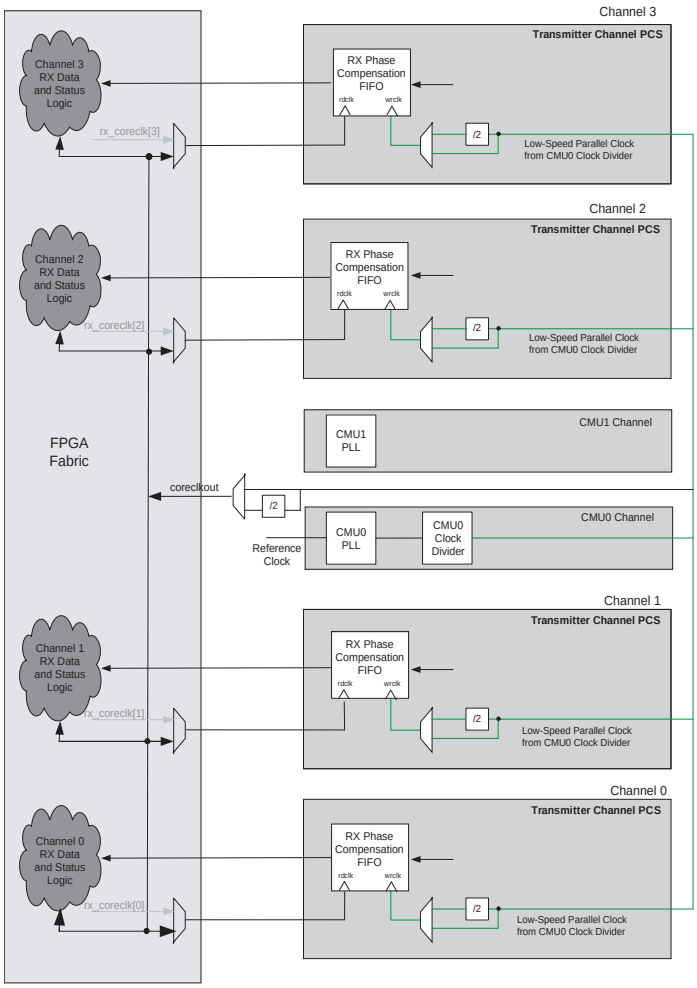
Bonded Channel Configuration

All bonded transceiver channel configurations have rate matcher in the receiver data path. In ×4 and ×8 bonded channel configurations, the Quartus II software automatically drives the read port of the receiver phase compensation FIFO in all channels with the `coreclkout` signal (from the master transceiver block in the case of ×8 bonded mode). Use the `coreclkout` signal to latch the receiver data and status signals from all channels in the FPGA fabric.

This configuration uses one FPGA global and/or regional clock resource per bonded link for the `coreclkout` signal.

Figure 2–35 shows the FPGA fabric-Receiver interface clocking in ×4 bonded channel configuration.

Figure 2–35. FPGA Fabric-Receiver Interface Clocking in a x4 Bonded Channel Configuration *(Note 1)*



Note to Figure 2–35:

(1) The green lines represent low-speed parallel clock from the `CMU0` clock divider.

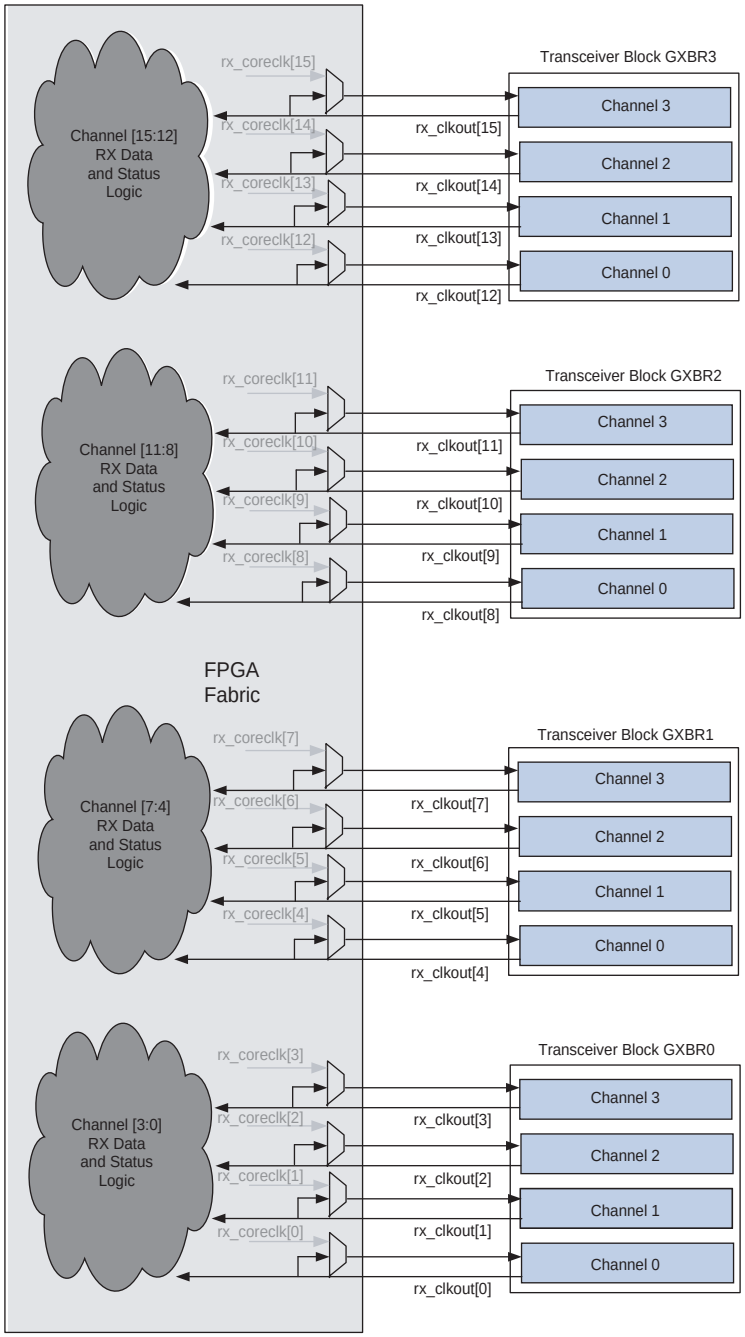
Limitations of the Quartus II Software-Selected Receiver Phase Compensation FIFO Read Clock

In non-bonded channel configurations without rate matcher, the Quartus II software cannot determine if the incoming serial data in all channels has a 0 PPM frequency difference. The Quartus II software uses the recovered clock `rx_clkout` signal from each channel to clock the read port of its receiver phase compensation FIFO. This results in one global, regional, or global and regional clock resource being used per channel for the `rx_clkout` signal.

Example 7: Sixteen Channels Across Four Transceiver Blocks

Figure 2–36 shows 16 non-bonded receiver channels without rate matcher, located across four transceiver blocks. The incoming serial data to all 16 channels have a 0 PPM frequency difference with respect to each other. The Quartus II software uses `rx_clkout` from each channel to clock the read port of its receiver phase compensation FIFO. This results in 16 global, regional, or global and regional clock resources being used, one for each channel.

Figure 2–36. Sixteen Non-Bonded Receiver Channels without Rate Match for Example 7



Because the recovered clock `rx_clkout` signals from all 16 channels have a 0 PPM frequency difference, you can use a single `rx_clkout` to clock the receiver phase compensation FIFO in all 16 channels. This results in only one global, regional, or global and regional clock resource being used instead of 16. To achieve this, you must select the receiver phase compensation FIFO read clocks instead of the Quartus II software default selection, as described in "User-Selected Receiver Phase Compensation FIFO Read Clock" on page 2–67.

User-Selected Receiver Phase Compensation FIFO Read Clock

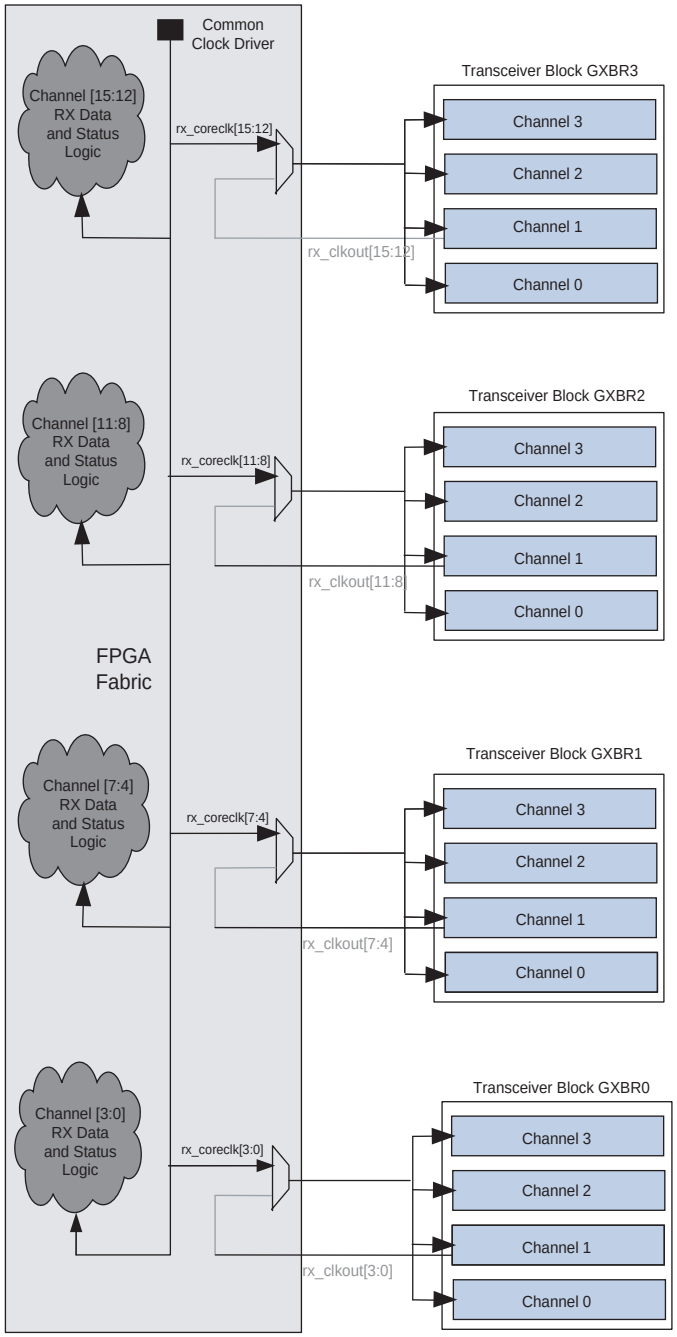
The ALTGX MegaWizard Plug-In Manager provides an optional port named `rx_coreclk` for each instantiated receiver channel. If you enable this port, the Quartus II software does not automatically select the receiver phase compensation FIFO read clock source. Instead, the signal that you drive on the `rx_coreclk` port of the channel clocks the read side of its receiver phase compensation FIFO.

You can use the flexibility of selecting the receiver phase compensation FIFO read clock to reduce the global, regional, or global and regional clock resource utilization. You can connect the `rx_coreclk` ports of all receiver channels in your design and drive them using a common clock driver that has a 0 PPM frequency difference with respect to the FIFO write clocks of these channels. Use this common clock driver to latch the receiver data and status signals in the FPGA fabric for these channels. This FPGA fabric-Transceiver interface clocking scheme uses only one global, regional, or global and regional clock resource for all channels.

Example 8: Sixteen Identical Channels Across Four Transceiver Blocks

Figure 2–37 shows 16 channels located across four transceiver blocks. The incoming serial data to all 16 channels has a 0 PPM frequency difference with respect to each other. The `rx_coreclk` ports of all 16 channels are connected together and driven by a common clock driver. This common clock driver also latches the receiver data and status logic of all 16 receiver channels in the FPGA fabric. Only one global, regional, or global and regional clock resource is used with this clocking scheme, compared to 16 global, regional, or global and regional clock resources needed without the `rx_coreclk` ports (the Quartus II software-selected receiver phase compensation FIFO read clock).

Figure 2–37. Sixteen Identical Channels Across Four Transceiver Blocks for Example 8



Common Clock Driver Selection Rules

The common clock driver driving the `rx_coreclk` ports of all channels must have a 0 PPM frequency difference with respect to the receiver phase compensation FIFO write clocks of these channels. If there is any frequency difference between the FIFO read clock (`rx_coreclk`) and the FIFO write clock, the FIFO overflows or under-runs, resulting in corrupted data transfer between the FPGA fabric and the receiver.

Table 2–17 lists the receiver phase compensation FIFO write clocks that the Quartus II software selects in various configurations.

Table 2–17. Receiver Phase Compensation FIFO Write Clocks

Configuration	Receiver Phase Compensation FIFO Write Clock	
	Without Byte Serializer	**With Byte Serializer**
Non-Bonded Channel Configuration with rate matcher	Low-speed parallel clock from the local clock divider in the associated channel (`tx_clkout`)	Divide-by-two version of the low-speed parallel clock from the local clock divider in the associated channel (`tx_clkout`)
Non-Bonded Channel Configuration without rate matcher	Parallel recovered clock from the receiver PMA in the associated channel (`rx_clkout`)	Divide-by-two version of the parallel recovered clock from the receiver PMA in the associated channel (`rx_clkout`)
×4-Bonded Channel Configuration	Low-speed parallel clock from the `CMU0` clock divider of the associated transceiver block (`coreclkout`)	Divide-by-two version of the low-speed parallel clock from the `CMU0` clock divider of the associated transceiver block (`coreclkout`)
×8-Bonded Channel Configuration	Low-speed parallel clock from the `CMU0` clock divider of the master transceiver block (`coreclkout` from the master transceiver block)	Divide-by-two version of the low-speed parallel clock from the `CMU0` clock divider of the master transceiver block (`coreclkout` from the master transceiver block)

To ensure that you understand the 0 PPM clock driver rule, the Quartus II software expects the following set of user assignments whenever you use the `rx_coreclk` port to drive the receiver phase compensation FIFO read clock:

- **GXB 0 PPM Core Clock Setting**

Failing to make this assignment correctly when using the `rx_coreclk` port results in a Quartus II compilation error.

The **GXB 0 PPM core clock setting** allows the following clock drivers to drive the `rx_coreclk` ports:

- `tx_clkout` in non-bonded channel configurations with rate matcher
- `tx_clkout` and `rx_clkout` in non-bonded configurations without rate matcher
- `coreclkout` in bonded channel configurations
- FPGA CLK input pins
- Transceiver `refclk` pins
- Clock output from left and right and top and bottom PLLs (`PLL_L`, `PLL_R`, and `PLL_T`, `PLL_B`)

The Quartus II software does not allow gated clocks or clocks generated in FPGA logic to drive the `tx_coreclk` ports.

Because the 0 PPM clock group assignment allows the FPGA CLK input pins and transceiver `refclk` pins as the clock driver, the Quartus II compiler cannot determine if there is a 0 PPM difference between the FIFO write clock and read clock for each channel.

You must ensure that the clock driver for all connected `rx_coreclk` ports has a 0 PPM difference with respect to the FIFO write clock in those channels.

Table 2–18 lists the Quartus II assignments that you must make.

Table 2–18. Quartus II Assignments

From	Full design hierarchy name of one of the following clock drivers that you choose to drive the `rx_coreclk` ports of all identical channels *(1)*: ■ `tx_clkout` ■ `rx_clkout` ■ `coreclkout` ■ FPGA CLK input pins ■ Transceiver `refclk` pins ■ Clock output from the left and right or top and bottom PLLs ■ `tx_dataout` port of one of the identical channels
To	`rx_datain` pins of all channels whose `rx_coreclk` ports are connected together and driven by the 0 PPM clock driver.
Assignment Name	GXB 0 PPM Core Clock Setting
Value	ON

Note to Table 2–18:

(1) You can find the full hierarchy name of the 0 PPM clock driver using the **Node Finder** feature in the Quartus II Assignment Editor.

For more implementation details, refer to "Configuration Example 3: Configuring Sixteen Channels Across Four Transceiver Blocks" on page 2–75.

Basic (PMA Direct) mode

In Basic (PMA Direct) mode, each channel must be clocked by its own `rx_clkout`. As a result, the number of global and/or regional clock resources required is significantly higher. Bonding is not supported for receivers configured in Basic (PMA Direct) functional mode.

Using the CMU/ATX PLL for Clocking User Logic in the FPGA Fabric

Some designs that use multiple clock domains may run out of PLLs in the FPGA fabric. In such a scenario, if your design has CMU or ATX PLLs that are not being used, it may be possible to use them for clocking user logic in the FPGA fabric. However, the CMU PLLs and ATX PLLs do not have many features that are supported by the PLLs in the FPGA fabric.

The following are the supported features on CMU PLLs and ATX PLLs used as PLLs for clocking user logic in the FPGA fabric:

- Single clock output
- Programmable PLL bandwidth
- PLL PFD power down control
- Lock status signal

To use this feature, you must create an ALTGX instance with a single channel in **Transmitter Only** mode that uses the required CMU PLL or ATX PLL. Follow these steps to create the ALTGX instance:

1. Choose **Basic (PMA Direct) ×N mode** as the protocol.
2. Select **Transmitter Only** operation mode.
3. Select the input clock frequency.
4. Select the appropriate values of data rate and channel width based on the desired output clock frequency. To generate a 250 MHz clock using an input clock frequency of 50 MHz, select a channel width of **10** and a data rate of **2500 Mbps** (Equation 2–1).

Equation 2–1.

$$f_{out} = \frac{\text{data rate}}{\text{channel width}}$$

5. You can select the PLL bandwidth by choosing **Tx PLL bandwidth mode**.
6. You can instantiate the `pll_locked` port to indicate the PLL lock status.
7. You can instantiate `pll_powerdown` or `gxb_powerdown` to enable the PLL PFD power down control.
8. Use `tx_clkout` of the ALTGX instance as the clock source for clocking user logic in the FPGA fabric.

Configuration Examples

This section describes the following examples:

- "Configuration Example 1: Configuring 24 Channels in Basic (PMA Direct) ×N Mode in the EP4S100G5F45 Device" on page 2–72
- "Configuration Example 2: Configuring Sixteen Identical Channels Across Four Transceiver Blocks" on page 2–74
- "Configuration Example 3: Configuring Sixteen Channels Across Four Transceiver Blocks" on page 2–75
- "Configuration Example 4: Configuring Left and Right, Left, or Right PLL in VCO Bypass Mode" on page 2–77

Configuration Example 1: Configuring 24 Channels in Basic (PMA Direct) ×N Mode in the EP4S100G5F45 Device

Each transceiver block has four regular channels and two CMU channels that you can configure in Basic (PMA Direct) ×N mode. The EP4S100G5F45 device has four transceiver blocks located on each side of the device allowing configuration of up to 24 channels in Basic (PMA Direct) ×N mode.

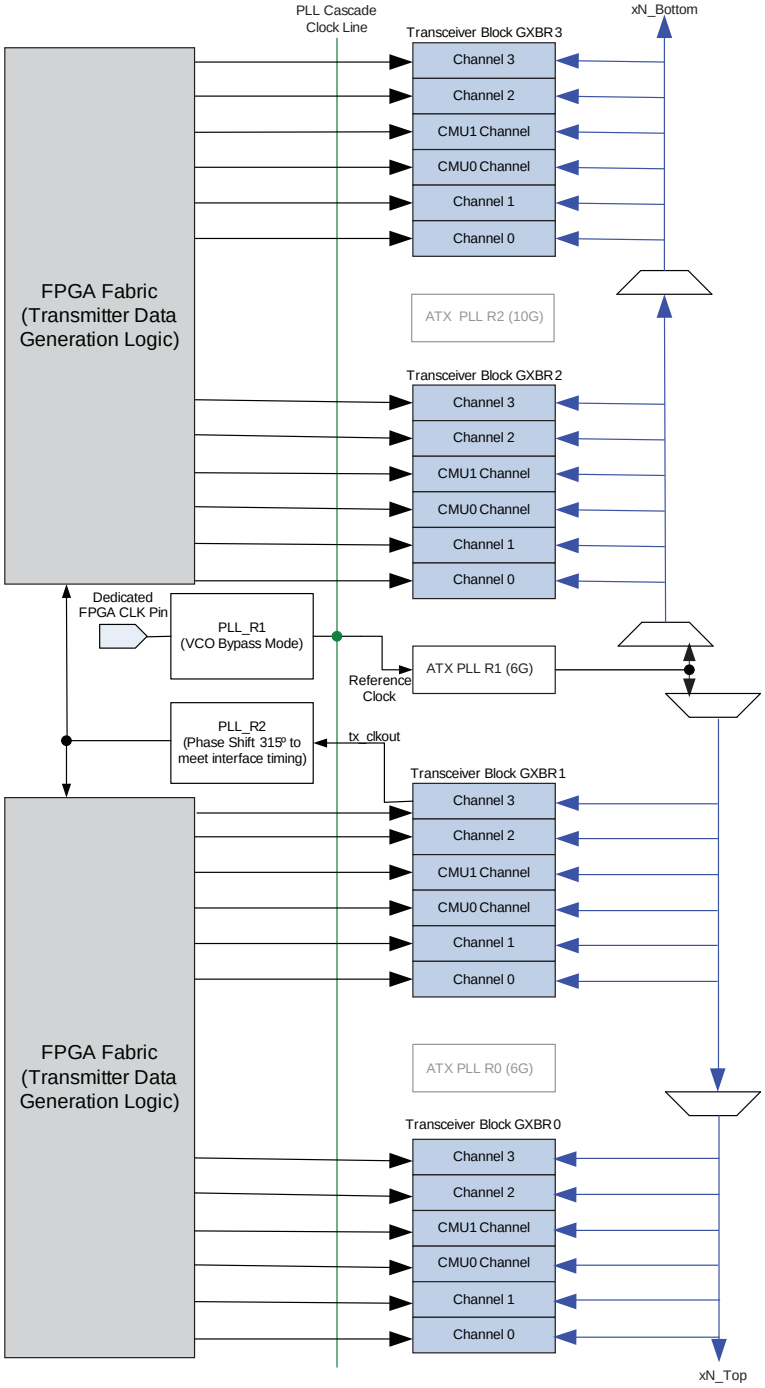
When all 24 channels on one side of the device are configured in Basic (PMA Direct) ×N mode, all eight CMU channels (two in each transceiver block) are configured as PMA-Only channels.

Use the `refclk` pins in each of the four transceiver blocks as receiver serial data input pins and configure the CMU PLLs as receiver CDRs when the CMU channel is configured as a PMA-Only channel. Due to the non-availability of CMU PLLs, you must use the 6G ATX PLL to generate the high-speed serial and low-speed parallel transceiver clocks for all 24 channels. Due to the non-availability of a `refclk` pin, you must use the left and right, or left or right PLL in VCO bypass mode to provide the reference clock through the PLL cascade clock line.

For more information about left and right PLL VCO bypass mode, refer to the *Transceiver Clocking for Stratix IV* chapter.

Figure 2–38 shows 24 channels on the right side of the EP4S100G5F45 device configured in Basic (PMA Direct) ×N mode running at 6.5 Gbps with a 20-bit FPGA fabric-PMA interface width. Because all 24 channels on the right side of the device are configured in Basic (PMA Direct) ×N mode, the right `PLL_R1` configured in VCO bypass mode is used to provide the input reference clock to the 6G ATX PLL. The 6G ATX PLL generates the high-speed serial and low-speed parallel transceiver clocks that are distributed to the 24 channels though the `×N_Top` and `×N_Bottom` clock network. Because the data rate of 6.5 Gbps requires a left and right, or left or right PLL to meet FPGA fabric-Transmitter PMA interface timing, `tx_clkout` from one of the 24 channels is phase shifted by 315° using `PLL_R2`. The phase shifted output clock from `PLL_R2` is used to clock the FPGA fabric logic that generates the transmitter parallel data and control signals.

Figure 2–38. Twenty-Four Channels on the Right Side of the EP4S100G5F45 Device Configured in Basic (PMA Direct) ×N Mode for Configuration Example 1 *(Note 1)*



Note to Figure 2–38:

(1) The green line represents the PLL cascade clock line and the blue lines represent the 6G ATX PLL block.

Configuration Example 2: Configuring Sixteen Identical Channels Across Four Transceiver Blocks

This example relates to "User-Selected Receiver Phase Compensation FIFO Read Clock" on page 2–67.

Figure 2–39 shows 16 identical transmitter channels located across four transceiver blocks. The `tx_coreclk` ports of all 16 transmitter channels are connected together and driven by the `tx_clkout[4]` signal from channel 0 in transceiver block `GXBR1`. The `tx_clkout[4]` signal also drives the transmitter data and control logic of all 16 transmitter channels in the FPGA fabric. With this clocking scheme, only one global clock resource is used by the `tx_clkout[4]` signal.

Figure 2–39. Sixteen Identical Channels Across Four Transceiver Blocks for Configuration Example 2

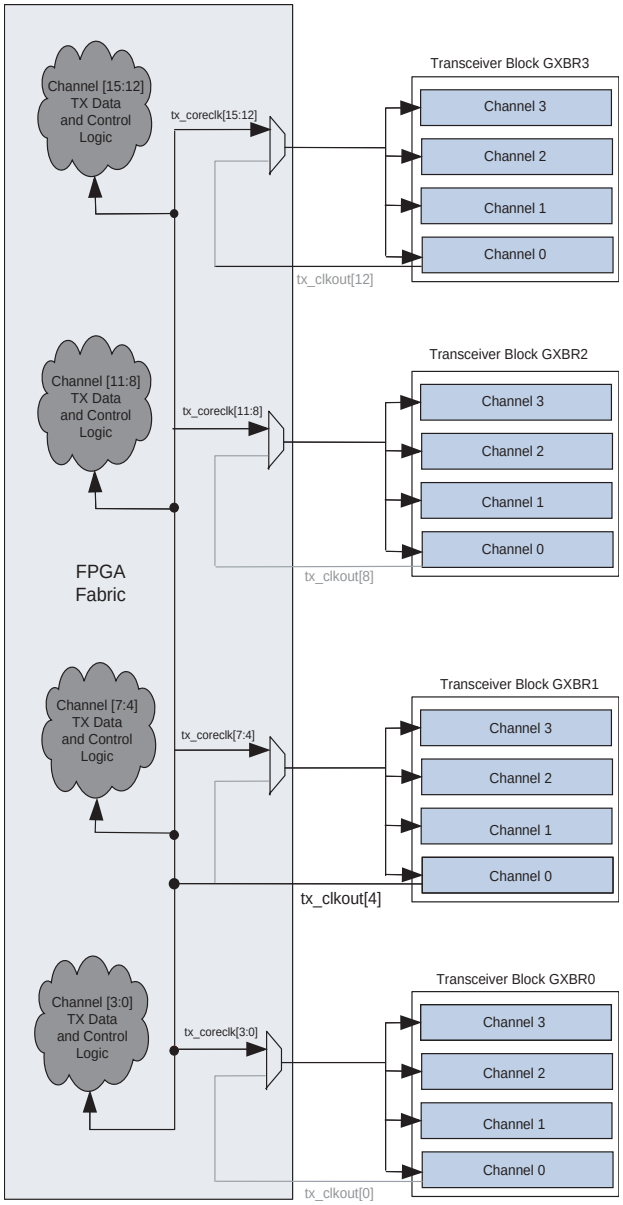


Table 2–19 lists the Quartus II assignments that you must make for the clocking schem shown in Figure 2–38.

Table 2–19. Quartus II Assignments

From	`top_level/top_xcvr_instance1/altgx_component/tx_clkout[4]` *(1)*
To	`tx_dataout[15..0]`
Assignment Name	GXB 0 PPM Core Clock Setting
Value	ON

Note to Table 2–19:

(1) This is an example design hierarchy path for the `tx_clkout[4]` signal.

Configuration Example 3: Configuring Sixteen Channels Across Four Transceiver Blocks

This example relates to "User-Selected Receiver Phase Compensation FIFO Read Clock" on page 2–67.

Figure 2–40 shows 16 non-bonded channels without rate matcher located across four transceiver blocks. The incoming serial data to all 16 channels has a 0 PPM frequency difference with respect to each other. The `rx_coreclk` ports of all 16 channels are connected together and driven by `rx_clkout[9]` in transceiver block `GXBR2`. `rx_clkout[9]` also clocks the receiver data and status signals of all 16 channels in the FPGA fabric. With this clocking scheme, only one global, regional, or global and regional clock resource is used by `rx_clkout[9]`.

Figure 2–40. Sixteen Channels Across Four Transceiver Blocks for Configuration Example 3

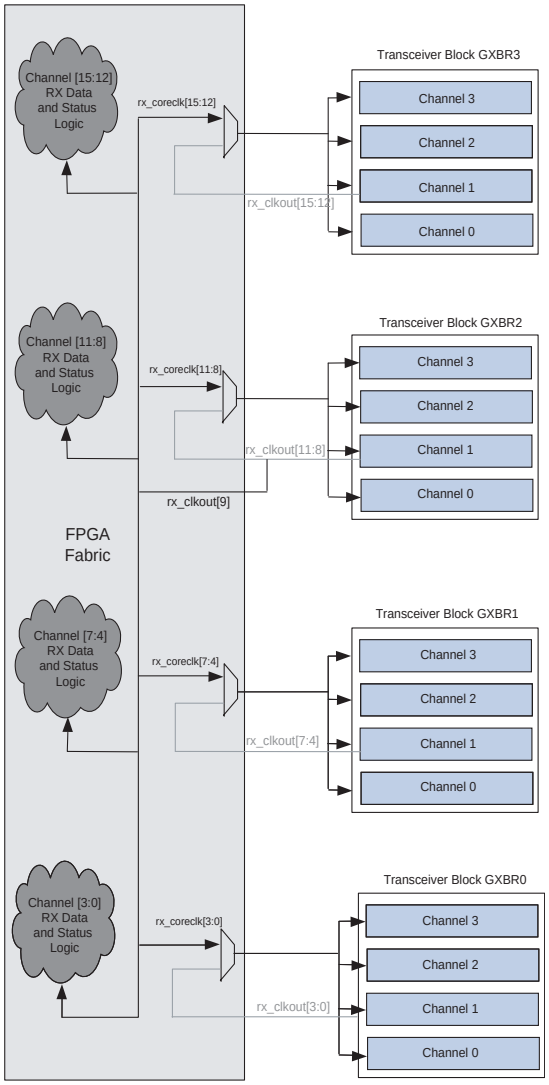


Table 2–20 lists the Quartus II assignments that you must make for the clocking scheme shown in Figure 2–40.

Table 2–20. Quartus II Assignments for Appendix Example 4

From	`top_level/top_xcvr_instance1/altgx_component/rx_clkout[9]` *(1)*
To	`rx_datain[15..0]`
Assignment Name	GXB 0 PPM Core Clock Setting
Value	ON

Note to Table 2–20:

(1) This is an example design hierarchy path for the `rx_clkout[9]` signal.

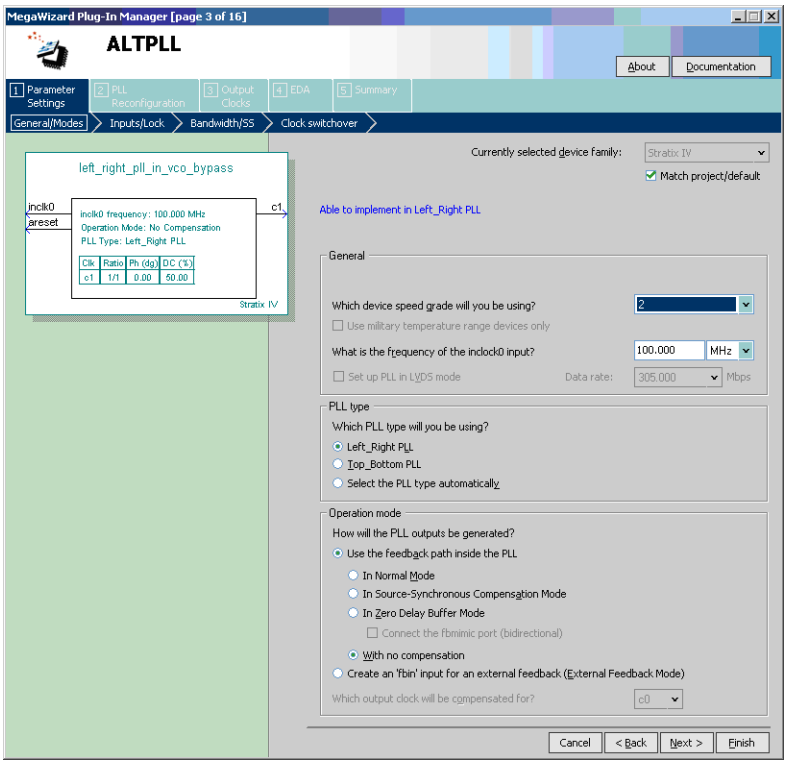
Configuration Example 4: Configuring Left and Right, Left, or Right PLL in VCO Bypass Mode

This example relates to "Left and Right, Left, or Right PLL in VCO Bypass Mode" on page 2–17.

To configure the left and right, left, or right PLL in VCO bypass mode, use the following steps:

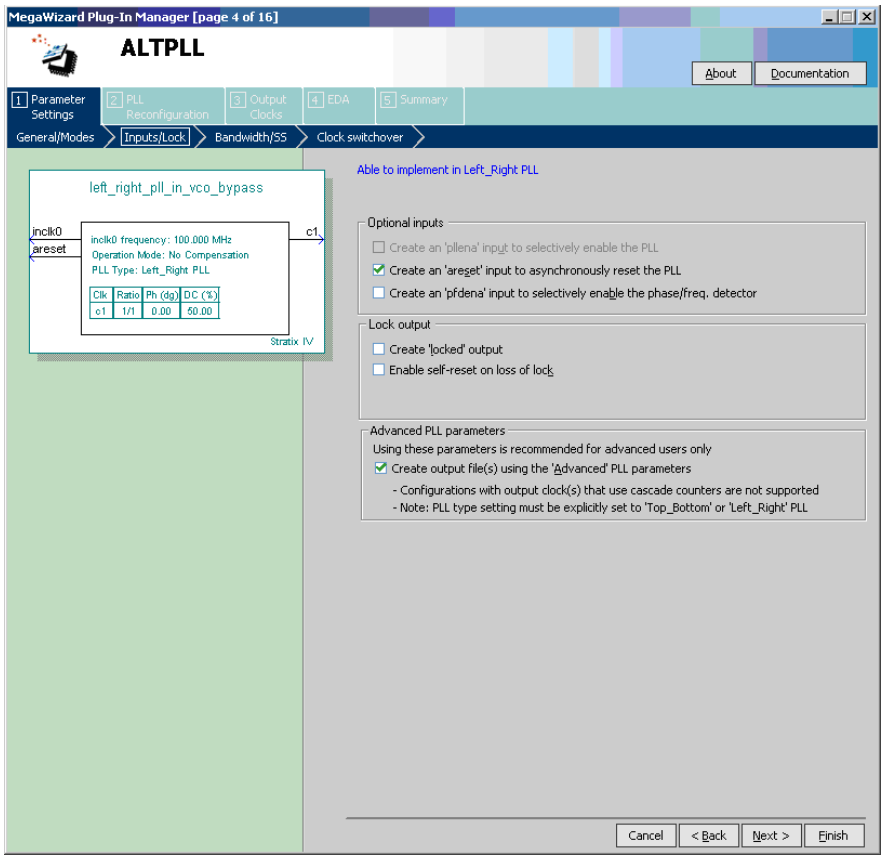
1. Under the **General/Modes** tab, enter the desired input reference clock frequency.
   a. Under **PLL Type**, select **Left_Right_PLL.**
   b. Under **Operation** mode, select the **With no compensation** option (Figure 2–41).

Figure 2–41. No Compensation Option Used for Configuration Example 4



2. Under the **Inputs/Lock** tab, select **Create output file(s) using the 'Advanced' PLL parameters** (Figure 2–42).

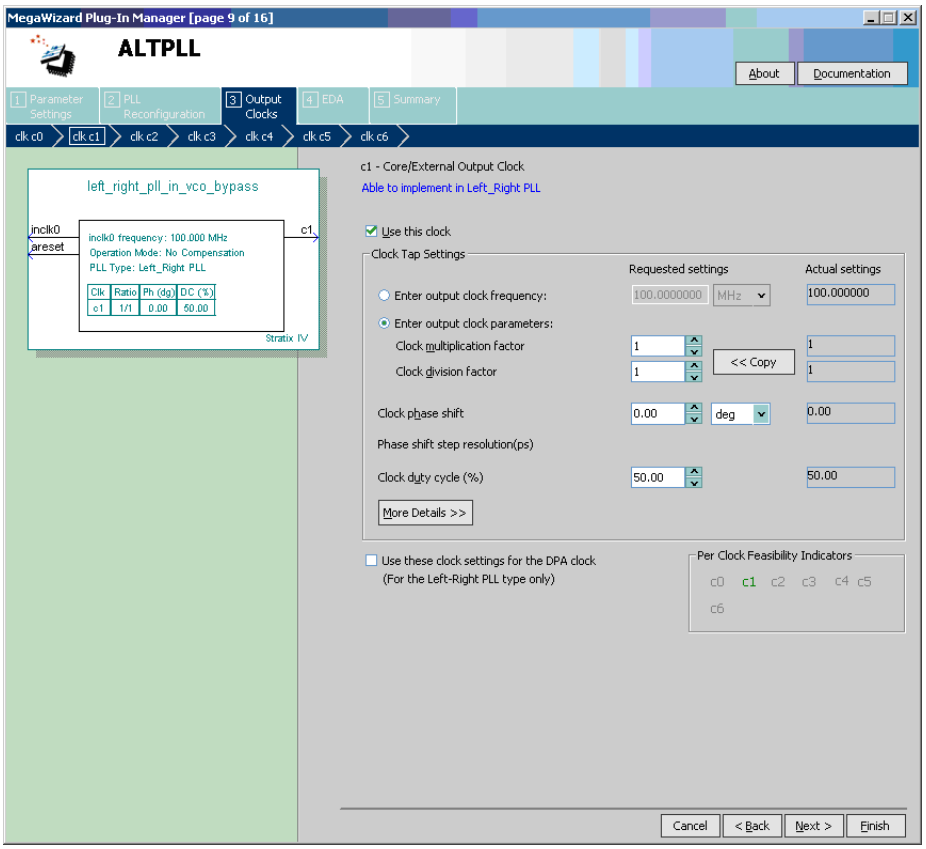
Figure 2–42. Create Output File(s) Using the 'Advanced' PLL Parameters Option Use for Configuration Example 4



3. Under the **Output Clocks** tab turn off **Use this clock** for clk c0.
4. Turn on **Use this clock** for clk c1 (Figure 2–43).

The VCO bypass option is only enabled for clock output c1.

Figure 2–43. Use This Clock Option Used for Configuration Example 4



5. Click **Finish** for the MegaWizard Plug-In Manager to generate the verilog **.v** file for the ALTPLL instantiation.

Next, from the command line, go to the directory where you have the ALTPLL instance files (**.v** or **.vhd**) and type the following command:

```
>> qmegawiz -silent -wiz_override =
c1_test_source=1,c1_mode=BYPASS,clk1_counter=C1" <altpll design file>
```

This command places your ALTPLL instance in VCO bypass mode. Connect the clk c1 output of the left and right, left, or right PLL to the input reference clock port of the 6G ATX PLL used to generate the transceiver clocks.

Document Revision History

Table 2–21 shows the revision history for this chapter.

Table 2–21. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
November 2009, v3.0	■ Added Figure 2–1, Figure 2–12,and Figure 2–13. ■ Added Table 2–1, Table 2–2, Table 2–8, and Table 2–2. ■ Updated Table 2–5 and Table 2–14. ■ Updated all graphics. ■ Updated all sections. ■ Added Stratix IV GT information. ■ Re-organized information. ■ Minor text edits.	—
June 2009, v2.2	■ Updated Figure 2–5 and Figure 2–7. ■ Updated the “Transceiver Data Rates Supported in Basic (PMA Direct) Mode”, ■ , “FPGA Fabric PLLs-Transceiver PLLs Cascading in the 780-Pin Package”, “FPGA Fabric PLLs-Transceiver PLLs Cascading in the 1152-Pin Package”, sections. ■ Removed Table 2-5, Table 2-6, Table 2-7 ■ Removed Figure 2-17 and Figure 2-18. ■ Minor text edits.	—
March 2009, v2.1	Minor updates.	—
November 2008, v2.0	Update to chapter.	—
May 2008 v1.0	Initial release.	—

3. Configuring Multiple Protocols and Data Rates in a Transceiver Block

SIV52003-4.0

Use this chapter to create transceiver instances and understand how to combine transceiver channels. The instances you can combine include **Receiver Only** and **Transmitter and Receiver** channels as well as channels configured in Protocol Functional modes, channels using PLL cascade clocks, channels in multiple transceiver blocks, and channels with a Basic (PMA Direct) configuration. This chapter also offers several examples of sharing the clock multiplier unit phase-locked loops (CMU PLLs).

For information about the supported data rate range for the ATX PLL, refer to the "Transceiver Performance Specifications" section in the *DC and Switching Characteristics* chapter.

Overview

Each transceiver channel in a Stratix® IV GX device can run at an independent data rate or in an independent protocol mode. Within each transceiver channel, the transmitter and receiver channels can run at different data rates. Each transceiver block consists of two CMU PLLs that provide clocks to all the transmitter channels within the transceiver block. Each receiver channel contains a dedicated clock data recovery (CDR) unit.

In addition to the CMU PLLs, the auxiliary transmit (ATX) PLLs are available to provide clocks to the transmitter channels that are configured for a specific data rate range.

This chapter includes the following sections:

- "Glossary of Terms" on page 3–2
- "Creating Transceiver Channel Instances" on page 3–3
- "General Requirements to Combine Channels" on page 3–3
- "Sharing CMU PLLs" on page 3–5
- "Sharing ATX PLLs" on page 3–9
- "Combining Receiver Only Channels" on page 3–9
- "Combining Transmitter Channel and Receiver Channel Instances" on page 3–10
- "Combining Transceiver Instances in Multiple Transceiver Blocks" on page 3–12
- "Combining Transceiver Instances Using PLL Cascade Clocks" on page 3–15
- "Combining Channels Configured in Protocol Functional Modes" on page 3–16
- "Combining Transceiver Channels with Basic (PMA Direct) Configuration" on page 3–24
- "Combination Requirements When Channel Reconfiguration is Enabled" on page 3–41

- "Combining Transceiver Channels When the Adaptive Equalization (AEQ) is Enabled" on page 3–46
- "Combination Requirements for Stratix IV GT Devices" on page 3–48
- "Summary" on page 3–49

Each transmitter channel has a local divider (/1, /2, or /4) that divides the high clock output of the CMU PLL to provide high-speed serial and low-speed parallel clocks for its physical coding sublayer (PCS) and physical medium attachment (PMA) functional blocks.

You can configure the RX CDR present in the receiver channel to a distinct data rate and provide separate input reference clocks. Each receiver channel also contains a local divider that divides the high-speed clock output of the RX CDR and provides clocks for its PCS and PMA functional blocks. To enable transceiver channel settings, the Quartus® II software provides the ALTGX MegaWizard™ Plug-In Manager interface. The ALTGX MegaWizard Plug-In Manager allows you to instantiate a single transceiver channel or multiple transceiver channels in **Receiver and Transmitter**, **Receiver only**, and **Transmitter only** configurations.

Glossary of Terms

Table 3–1 lists the terms used in the chapter.

Table 3–1. Glossary of Terms Used in this Chapter

Configuration	Description
Regular Channels	This refers to the four transceiver channels in each transceiver block that contain PCS.
Basic (PMA Direct)	This refers to the Basic (PMA Direct) configuration that you can use for both regular and CMU channels. Basic (PMA Direct) mode has two variations, ×1 and ×N. The term "Basic (PMA Direct)" used in this chapter refers to both ×1 and ×N and to regular/CMU Channels. Any specific reference to ×1 and ×N or regular/CMU channels is stated explicitly.
Non-Basic (PMA Direct)	This term refers to all single channel non-bonded configurations (for example, GIGE, PCI Express [PIPE] ×1) or bonded channel configurations that have PCS enabled (for example, Basic ×4 and ×8, XAUI, PCI Express [PIPE] ×4 and ×8). Also, any reference to a channel in non-Basic (PMA Direct) mode indicates that the channel is a regular transceiver channel.
Basic (PMA Direct) ×1	A transceiver channel set up in this configuration uses the high-speed serial clock from the CMU PLL that is present within the same transceiver block. You can select this configuration by setting the **Which protocol you will be using?** option to **Basic (PMA Direct)** and the **Which sub protocol you will be using?** option to **none**.
Basic (PMA Direct) ×N	A transceiver channel set up in this configuration uses the ×N high-speed clock lines. You can select this configuration by setting the **Which protocol you will be using?** option to **Basic (PMA direct)** and the **Which sub protocol you will be using?** option to **×N**.

For more information about transceiver channel set up using a Basic (PMA Direct) ×N configuration, refer to the *Stratix IV GX Transceiver Clocking* chapter.

Creating Transceiver Channel Instances

The two ways you can instantiate multiple transceiver channels in the **General** screen of the ALTGX MegaWizard Plug-In Manager are:

- In the **What is the number of channels?** option, select the required value. This method creates all the transceiver channels with identical configurations. For examples, refer to "Combining Transceiver Instances in Multiple Transceiver Blocks" on page 3–12.
- In the **What is the number of channels?** option, select **1** and create a single channel transceiver instance. To instantiate additional transceiver channels with an identical configuration, select the created ALTGX instance multiple times. If you need additional transceiver channels with different configurations, create separate ALTGX megafunction instances with different settings and use them in your design.

When you create instances using the above methods, you can force the placement of up to four transceiver channels within the same transceiver block. Do this by assigning the `tx_dataout` and `rx_datain` ports of the channel instances to a single transceiver bank. If you do not assign pins to the `tx_dataout` and `rx_datain` ports, the Quartus II software chooses default pin assignments. When you compile the design, the Quartus II software combines multiple channel instances within the same transceiver block if the instances meet specific requirements. The following sections explain these requirements for different transceiver configurations.

General Requirements to Combine Channels

When you create multiple ALTGX instances, the Quartus II software requires you to set identical values for the following parameters and signals to combine the ALTGX instances within the same transceiver block or in transceiver blocks on the same side of the device. The following sections describe these requirements.

Transmitter Buffer Voltage (V_{CCH})

The Stratix IV GX device provides you the option to select 1.4 V or 1.5 V for the V_{CCH} supply through the ALTGX MegaWizard Plug-In Manager. To combine the channel instances within the same transceiver block, the Quartus II software requires you to set the same V_{CCH} value in all the channel instances.

The data rate of the transmitter channel is limited based on the V_{CCH} value selected.

For the data rate restrictions, refer to the *DC and Switching characteristics* chapter.

Transceiver Analog Power ($V_{CCA_L/R}$)

The Stratix IV GX device contains two different power supply pins, `VCCA_L` and `VCCA_R`, that provide power to the PMA blocks in all the transceiver channels on the left and right sides of the device, respectively.

The Stratix IV GX device provides you the option to select 2.5 V or 3.0 V for the `VCCA_L/R` supply through the ALTGX MegaWizard Plug-In Manager. You must set the same `VCCA_L/R` value for all the transceiver channel instances to enable the Quartus II software to place them in the transceiver blocks on the same side of the device. For example, if you have two ALTGX instances that you would like to place on the left side transceiver banks `GXBL0` and `GXBL1`, the `VCCA_L/R` values in the two ALTGX instances must be the same.

The data rate of the transceiver channel is limited based on the $V_{CCA_L/R}$ value selected.

For the data rate restrictions, refer to the *DC and Switching characteristics* chapter.

Control Signals

This section contains information about the `gxb_powerdown`, `reconfig_fromgxb`, and `reconfig_togxb` ports.

gxb_powerdown Port

The `gxb_powerdown` port is an optional port that you can enable in the ALTGX MegaWizard Plug-In Manager. If enabled, you must drive the `gxb_powerdown` port in the ALTGX instances from the same logic or the same input pin to enable the Quartus II software to assign them in the same transceiver block.

reconfig_fromgxb and reconfig_togxb Ports

In the ALTGX MegaWizard Plug-In Manager, the `reconfig_fromgxb` and `reconfig_togxb` ports are enabled if you select one of the following options in the **Reconfig** screen:

- Analog Controls (VOD, Pre-emphasis, Manual Equalization, EyeQ)
- Enable Channel and Transmitter PLL reconfiguration
- Offset cancellation for receiver channels (always enabled if the configuration is **Transmitter and Receiver** or **Receiver only**)

To combine multiple instances within the same transceiver block:

- The `reconfig_fromgxb` ports must be enabled in each instance AND
- These ports must be connected to the same reconfig controller

For example, consider that you want to place a **Receiver only** and **Transmitter only** instance in the same transceiver block. For the **Receiver only** instance, the Quartus II software automatically enables the `reconfig_fromgxb` port. For the **Transmitter only** instance, you must select the options in the **reconfig screen** (mentioned above) to enable the `reconfig_fromgxb` port. In the design, connect these ports from the **Transmitter only** and **Receiver only** instance to the same reconfig controller.

For more information about connecting these ports to the dynamic reconfiguration controller, refer to the "Connecting the ALTGX and ALTGX_RECONFIG Instances" section of the *Stratix IV Dynamic Reconfiguration* chapter.

Calibration Clock and Power Down

Each calibration block in a Stratix IV GX device is shared by multiple transceiver blocks.

If your design uses multiple transceiver blocks, depending on the transceiver banks selected, you must connect the `cal_blk_clk` and `cal_blk_powerdown` ports of all channel instances to the same input pin or logic.

For more information about the calibration block and transceiver banks that are connected to a specific calibration block, refer to the "Calibration Blocks" section in the *Stratix IV Transceiver Architecture* chapter.

Asserting the `cal_blk_powerdown` port affects calibration on all transceiver channels connected to the calibration block.

Sharing CMU PLLs

When you create multiple transceiver channel instances using CMU PLLs and intend to combine these instances in the same transceiver block, the Quartus II software checks whether a single CMU PLL can be used to provide clock outputs for the transmitter side of the channel instances. If a single CMU PLL is not sufficient, the Quartus II software attempts to combine the channel instances using two CMU PLLs. Otherwise, the Quartus II software issues a Fitter error.

The following two sections describes the ALTGX instance requirements to enable the Quartus II software to share the CMU PLL.

Multiple Channels Sharing a CMU PLL

To enable the Quartus II software to share the same CMU PLL for multiple channels, the following parameters in the channel instantiations must be identical:

- "Base data rate" (the CMU PLL is configured for this data rate)
- CMU PLL bandwidth setting
- Reference clock frequency
- Input reference clock pin
- `pll_powerdown` port of the ALTGX instances must be driven from the same logic
- If the selected functional mode in one instance is (OIF) CEI Phy Interface or PCI Express (PIPE), the other instance must have the same functional mode to share the CMU PLL. For example, if you have two channels, one configured in Basic mode and the other configured in (OIF) CEI Phy Interface mode at the same data rate, the Quartus II software does not share the same PLL because the internal parameters for these two functional modes are different.

Each channel instance can have a different local divider setting. This is a useful option when you intend to run each channel within the transceiver block at different data rates that are derived from the same base data rate using the local divider values /1, /2, and /4. Example 1 shows this design configuration.

Example 1

Consider an example design with four instances of **Receiver and Transmitter** configuration in the same transceiver block at various serial data rates. Assume that each instance contains a channel and is driven from the same clock source and has the same CMU PLL bandwidth settings. Table 3–2 shows the configuration for Example 1.

Table 3–2. Configuration for Example 1

User-Created Instance Name	ALTGX MegaWizard Plug-In Manager Settings		
	Number of Channels	Configuration	Effective Data Rate
`inst0`	1	Receiver and Transmitter	4.25 Gbps
`inst1`	1	Receiver and Transmitter	2.125 Gbps
`inst2`	1	Receiver and Transmitter	1.0625 Gbps
`inst3`	1	Receiver and Transmitter	4.25 Gbps

For Example 1, you can share a single CMU PLL for all four channels because:

- One CMU PLL can be configured to run at 4.25 Gbps.
- Each channel can divide the CMU PLL clock output using the local divider and achieve the required data rates of 4.25 Gbps, 2.125 Gbps, and 1.0625 Gbps. Because each receiver channel has a dedicated CDR, the receiver side in each instance can be set up for these three data rates without any restrictions.

To enable the Quartus II software to share a single CMU PLL for all four channels, set the values shown in Table 3–3 in the **General** screen of the ALTGX MegaWizard Plug-In Manager.

Table 3–3. ALTGX MegaWizard Plug-In Manager Settings for Example 1

Instance	General Screen Option	Setting
`inst0`	**What is the effective data rate?**	4.25 Gbps
	Specify base data rate	4.25 Gbps *(1)*
`inst1`	**What is the effective data rate?**	2.125 Gbps
	Specify base data rate	4.25 Gbps *(1)*
`inst2`	**What is the effective data rate?**	1.0625 Gbps
	Specify base data rate	4.25 Gbps *(1)*
`inst3`	**What is the effective data rate?**	4.25 Gbps
	Specify base data rate	4.25 Gbps *(1)*

Note to Table 3–3:

(1) The **Specify base data rate** option is 4.25 Gbps for all four instances. Given that the CMU PLL bandwidth setting and input reference clock are the same and that the `pll_powerdown` ports are driven from the same logic or pin, the Quartus II software shares a single CMU PLL that runs at 4.25 Gbps.

You can force the placement of transceiver channels to a specific transceiver block by assigning pins to `tx_dataout` and `rx_datain`. Otherwise, the Quartus II software selects a transceiver bank.

Figure 3–1 and Figure 3–2 show the scenario before and after the Quartus II software combines the transceiver channel instances. Because the RX CDR is not shared between channels, only the CMU PLL is shown.

Each of the ALTGX instances has a `pll_powerdown` port. You must drive the `pll_powerdown` ports for all the instances from the same logic to enable the Quartus II software to share the same CMU PLL.

Figure 3–1. ALTGX Instances Before Compilation for Example 1

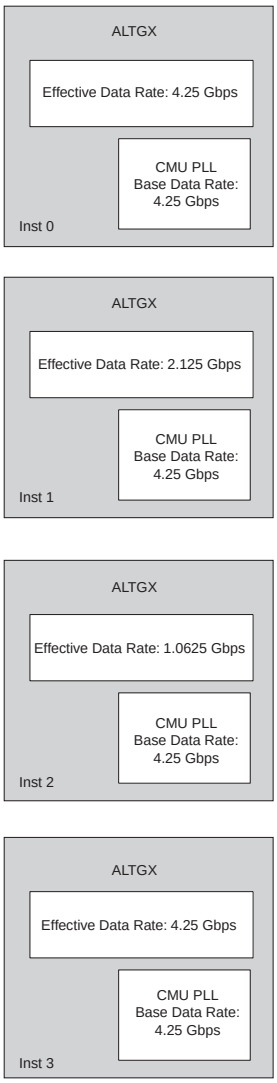
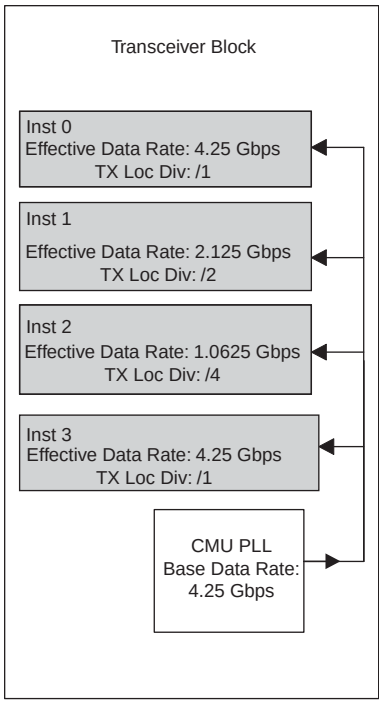


Figure 3–2 show the scenario after the Quartus II software combines the transceiver channel instances.

Figure 3–2. Combined Instances after Compilation for Example 1



Example 2

Consider the example design shown in Table 3–4. When you have two instances with the same serial data rate but with different CMU PLL data rates, the Quartus II software creates a separate CMU PLL for the two instances.

Table 3–4. Configuration for Example 2

User-Created Instance Name	ALTGX MegaWizard Plug-In Manager Settings			
	Number of Channels	Configuration	Effective Data Rate	Base Data Rate
`inst0`	1	Receiver and Transmitter	2.5 Gbps	2.5 Gbps
`inst1`	1	Receiver and Transmitter	2.5 Gbps	5 Gbps
`inst2`	1	Receiver and Transmitter	1 Gbps	1 Gbps

Even though the effective data rate of `inst1` is 2.5 Gbps (5 Gbps/2 = 2.5 Gbps), the same as `inst0`, when you compile the design, the Quartus II software requires two CMU PLLs to provide clocks for the transmitter side of the two instances because their base data rates are different. In this example, you have the third instance, `inst2`, that requires a third CMU PLL. Therefore, the Quartus II software cannot combine the above three instances within the same transceiver block.

Sharing ATX PLLs

The Quartus II software allows you to share the same ATX PLL for multiple transceiver instances if the following requirements are met:

- The ATX PLL bandwidth in both instances are the same
- If the selected functional mode in one instance is (OIF) CEI Phy Interface or PCI Express (PIPE), the other functional modes must be the same to share the ATX PLL. For example, if you have two channels, one configured in Basic mode and the other configured in (OIF) CEI Phy Interface mode at the same data rate, the Quartus II software does not share the same PLL because the internal parameters for these two functional modes are different.
- The base data rate and effective data rate values are the same.
- The `pll_powerdown` port in the instances are connected to the same logic.
- The instances are placed on the same side of the device.
- There is no contention on the ×N clock lines from the ATX PLL and the two instances.

For more information about ×N clocking, refer to the "Transmitter Channel Data Path Clocking" section in the *Stratix IV Transceiver Clocking* chapter.

Combining Receiver Only Channels

You can selectively use the receiver in the transceiver channel by selecting the **Receiver only** configuration in the **What is the Operating Mode?** option on the **General** screen of the ALTGX MegaWizard Plug-In Manager.

You can combine **Receiver only** channel instances of different configurations and data rates into the same transceiver block. Because each receiver channel contains its own dedicated CDR, each **Receiver only** instance (assuming one receiver channel per instance) can have a different data rate.

For the Quartus II software to combine the **Receiver only** instances within the same transceiver block, you must connect `gxb_powerdown` (if used) for all the channel instances to the same logic or input pin. For more information, refer to "General Requirements to Combine Channels" on page 3–3.

If your design contains a **Receiver only** instance, the Quartus II software disables all the settings for the unused transmitter channel present in the same physical transceiver channel. Therefore, the unused transmitter channel is always powered down in the hardware.

Combining Transmitter Channel and Receiver Channel Instances

You can create separate transmitter and receiver channel instances and assign the `tx_dataout` and `rx_datain` pins of the transmitter and receiver instances, respectively, to the same physical transceiver channel. This configuration is useful when you intend to run the transmitter and receiver channel at different serial data rates. To create separate transmitter and receiver channel instances, select the **Transmitter only** and **Receiver only** options in the operating mode (**General** screen) of the ALTGX MegaWizard Plug-In Manager.

Multiple Transmitter Channel and Receiver Channel Instances

The Quartus II software allows you to combine multiple **Transmitter only** and **Receiver only** channel instances within the same transceiver block. Based on the pin assignments, the Quartus II software combines the corresponding **Transmitter only** and **Receiver only** channels in the same physical channel. To enable the Quartus II software to combine the transmitter channel and receiver channel instances in the same transceiver block, follow the rules and requirements outlined in:

- "General Requirements to Combine Channels" on page 3–3
- "Multiple Channels Sharing a CMU PLL" on page 3–5
- "Combining Receiver Only Channels" on page 3–9

Example 3

Consider the example design shown in Table 3–5 with four ALTGX instances.

Table 3–5. Four ALTGX Instances for Example 3

Instance Name	Configuration	Serial Data Rate	Input Reference Clock Frequency
`inst0`	Transmitter only	3.125 bps	156.25 MHz
`inst1`	Receiver only	2.5 Gbps	156.25 MHz
`inst2`	Transmitter only	1.25 Gbps	125 MHz
`inst3`	Receiver only	2 Gbps	125 MHz

After you create the above instances, if you force the placement of the instances, as shown in Table 3–6, the Quartus II software combines `inst0` and `inst1` to physical channel 0, and `inst2` and `inst3` to physical channel 1.

Table 3–6. Forced Placement of the Instances for Example 3

Instance Name	Physical Channel Pin Assignments in the Same Transceiver Block
`inst0`	TX pin of channel 0
`inst1`	RX pin of channel 0
`inst2`	TX pin of channel 1
`inst3`	RX pin of channel 1

Figure 3–3 and Figure 3–4 show the transceiver channel instances before and after compilation.

Figure 3–3. ALTGX Transceiver Channel Instances Before Compilation for Example 3

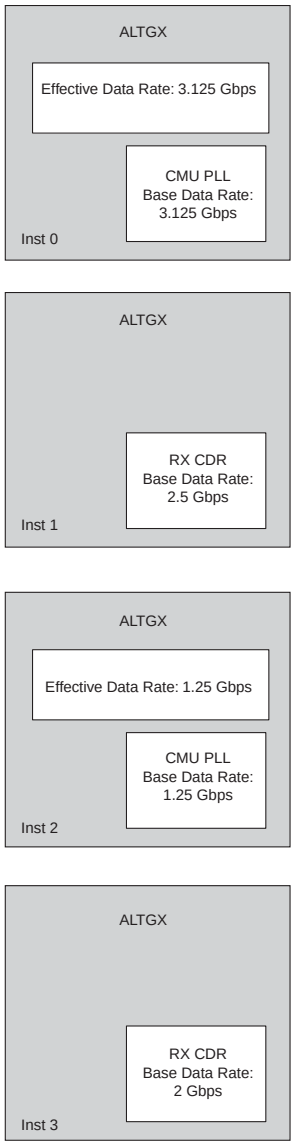
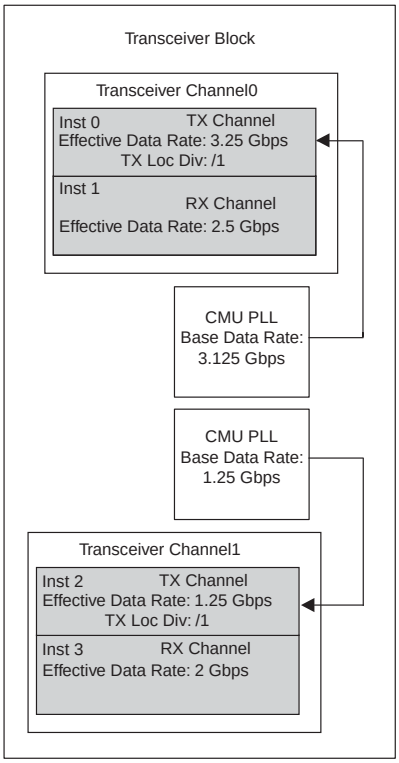


Figure 3–4 shows the transceiver channel instances after compilation.

Figure 3–4. Combined Transceiver Instances After Compilation for Example 3



Combining Transceiver Instances in Multiple Transceiver Blocks

The method to instantiate multiple transceiver channels using a single ALTGX instance was described in "Creating Transceiver Channel Instances" on page 3–3. The following section describes the method to instantiate multiple transceiver channels using multiple transceiver blocks.

When you create a transceiver instance that has more than four transceiver channels (assuming that the instance is created in non-Basic (PMA Direct) functional mode which requires regular channels), the Quartus II software attempts to combine the transceiver channels in multiple transceiver blocks. This is shown in the following examples.

Example 4

Consider the design example configuration shown in Table 3–7 with two ALTGX instances.

Table 3–7. Two ALTGX Instances for Example 4

Instance Name	Number of Transceiver Channels	Configuration	Serial Data Rate	Input Reference Clock
`inst0`	7	Receiver and Transmitter	4.25 Gbps	125 MHz from `refclk0`
`inst1`	1	Receiver and Transmitter	4.25 Gbps	125 MHz from `refclk0` (same as `inst0`)

In this case, assuming that all the required parameters specified in "Multiple Channels Sharing a CMU PLL" on page 3–5 are identical for `inst0` and `inst1`, the Quartus II software fits `inst0` and `inst1` in two transceiver blocks.

Figure 3–5 and Figure 3–6 show the transceiver instances before and after compilation.

Figure 3–5. Transceiver Channel Instances Before Compilation for Example 4

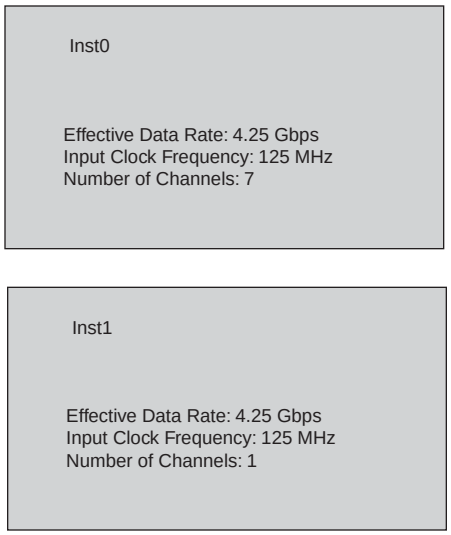
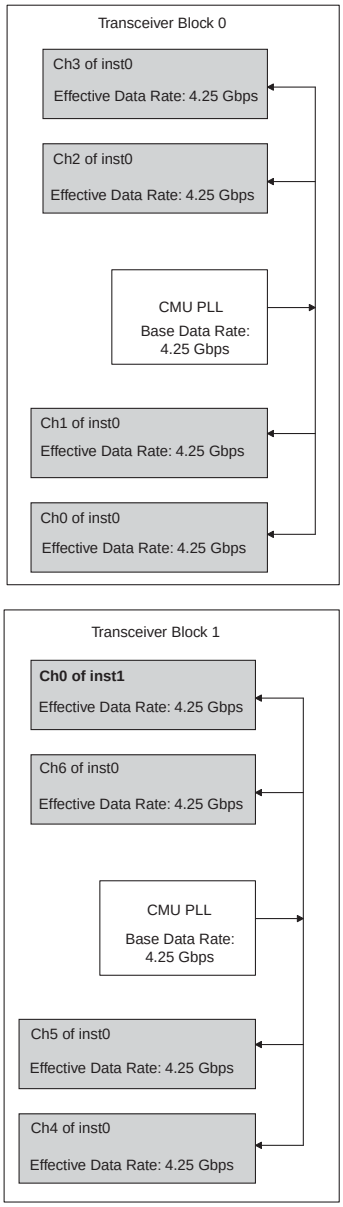


Figure 3–6 shows the transceiver instances after compilation.

Figure 3–6. Combined Transceiver Instances After Compilation for Example 4



You can force the placement of the transceiver channels in specific transceiver banks by assigning pins to the `tx_dataout` and `rx_datain` ports of `inst0` and `inst1`.

Even though `inst0` instantiates seven transceiver channels, the ALTGX MegaWizard Plug-In Manager provides only a one-bit wide `pll_inclk` port for `inst0`. In your design, provide only one clock input for the `pll_inclk` port. The Quartus II software uses two transceiver blocks to fit the seven channels and internally connects the input reference clock (connected to the `pll_inclk` port in your design) to the CMU PLLs of two transceiver blocks.

For `inst1`, the ALTGX MegaWizard Plug-In Manager provides a `pll_inclk` port. In this example, it is assumed that a single reference clock is provided for `inst0` and `inst1`. Therefore, connect the `pll_inclk` port of `inst0` and `inst1` to the same input reference clock pin. This enables the Quartus II software to share a single CMU PLL in transceiver block 1 that has three channels of `inst0` and one channel of `inst1` (shown as ch4, ch5, and ch6 in transceiver block 1 in Figure 3–6).

For the RX CDRs in `inst0`, the ALTGX MegaWizard Plug-In Manager provides seven bits for the `rx_cruclk` port (if you do not select the **Train Receiver CDR from pll_inclk** option in the **PLL/Ports** screen). This allows separate input reference clocks to the RX CDRs of each channel.

Combining Transceiver Instances Using PLL Cascade Clocks

The Stratix IV GX transceiver has the ability to cascade the output of the general purpose PLLs (`PLL_L` and `PLL_R`) to the CMU PLLs, ATX PLLs, and receiver CDRs. The left side PLLs can only be cascaded with the transceivers on the left side of the device. Similarly, the right side PLLs can only be cascaded with the transceivers on the right side of the device. Each side of the Stratix IV GX device contains a PLL cascade clock network; a single line network that connects the PLL cascade clock to the transceiver block. This clock line is segmented to allow different PLL cascade clocks to drive the transceiver CMU PLLs, ATX PLLs, and RX CDRs. Within the same segment, only a single `PLL_L/PLL_R` can drive these transceiver PLLs/CDRs. Therefore, if you create two instances that use different PLLs for cascading, you cannot place these instances within the transceiver block.

The segmentation locations differ based on the device family.

For more information about using the PLL cascade clock and segmentation, refer to the "Dedicated Left and Right PLL Cascade Lines Network" section in the *Stratix IV Transceiver Clocking* chapter.

Combining Channels Configured in Protocol Functional Modes

This section describes how to combine channels for various protocol functional modes.

Combining Channels in Bonded Functional Modes

This section describes the combination requirements in the two variations of bonded functional modes using transceiver PCS blocks. The two bonded functional modes are:

- "Bonded ×4 Functional Mode" on page 3–16—Examples of bonded ×4 mode:
  - Basic mode with sub protocol set to ×4
  - XAUI
  - PCI Express (PIPE) mode with sub protocol set to Gen1 ×4 or Gen2 ×4.
- "Bonded x8 Functional Mode" on page 3–19—Examples of bonded ×8 mode:
  - Basic mode with sub protocol ×8
  - PCI Express (PIPE) mode with sub protocol ×8

Bonded ×4 Functional Mode

The combination requirements for Basic x4, Deterministic Latency ×4, and PCI Express (PIPE) x4 functional modes (if PCI Express hard IP block is not used) are similar.

In this mode, the transmitter channels are synchronized to reduce skew. The Quartus II software shares the control from physical transmitter channel 0 with the other transmitter channels in the transceiver block. Therefore, when you an create an instance in this mode, the logical transmit channel 0 (`tx_dataout[0]` in the instance) must be assigned by the physical channel location 0 in the transceiver block.

The central clock divider block in the `CMU0` channel forwards the high-speed serial and low-speed parallel clocks to the transmitter channels.

This clocking scheme is described in the "Bonded Channel Configurations" section of the *Stratix IV Transceiver Clocking* chapter.

Because you used the central clock divider, the are two restrictions on the channel combinations:

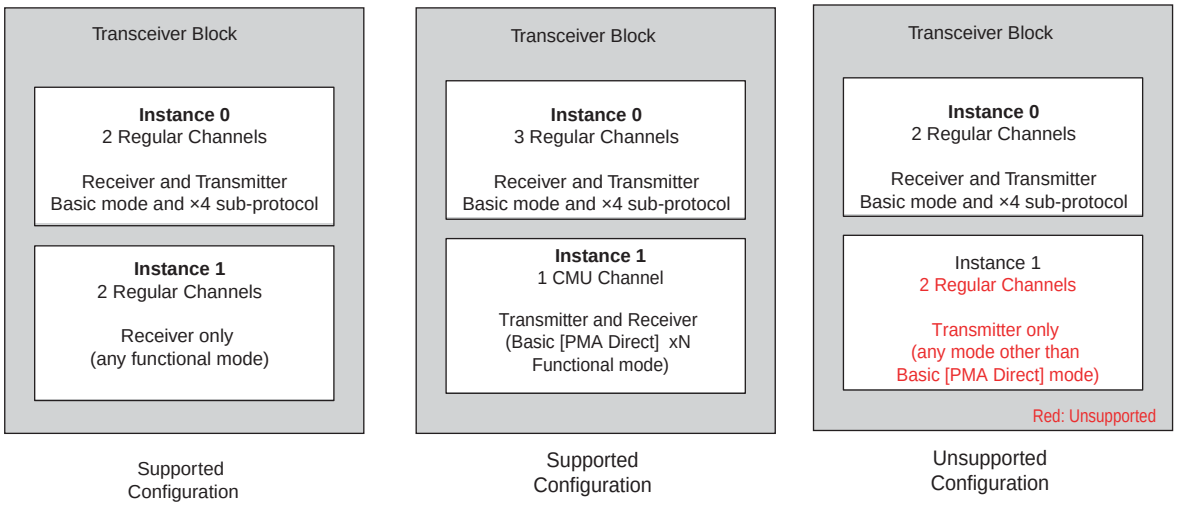
1. If you configure channels in bonded ×4 functional mode, the remaining transmitter channels (regular or CMU channels) within the transceiver block can be used only in Basic (PMA Direct) ×1 or ×N mode.

If PCI Express (PIPE) functional mode uses the PCI Express hard IP block, the combination requirements are different. For more information, refer to "Combining Channels Using the PCI Express hard IP Block with Other Channels" on page 3–23.

The receiver channels are clocked independently. Therefore, you can configure the unused receiver channels within a transceiver block in any allowed configuration.

Examples of supported and unsupported combinations are shown in Figure 3–7.

Figure 3–7. Examples of Supported and Unsupported Configurations to Combine Instances in Basic ×4 Mode



The `CMU0` PLL or `CMU1` PLL can drive the central clock divider block in the `CMU0` channel. In cases where you use `CMU1` PLL for bonded ×4 mode, the Quartus II software does not allow you to use `CMU0` PLL for any other configuration because part of the `CMU0` channel (the central clock divider) is already used by the bonded ×4 functional mode.

Using the remaining channels in Basic (PMA Direct) ×1 or ×N mode depends on the following conditions.

1. If `CMU1` PLL is available for clock generation, you can use the remaining transmitter channels in the transceiver block in Basic (PMA Direct) ×1 configuration.
2. If you want to configure the remaining transmitter channels at the same data rate as the bonded ×4 functional mode, you can configure the remaining transmitter channels in Basic (PMA Direct) ×1 mode. The requirements are specified in "Sharing CMU PLLs" on page 3–5 and "General Requirements to Combine Channels" on page 3–3.

3. If all the regular channels are configured in bonded ×4 functional mode, you can configure the transmitter side of the `CMU0` channel in Basic (PMA Direct) ×N mode in single-width configuration only (double-width configuration is not supported). You can use the `CMU1` channel in Basic (PMA Direct) ×N single-width or double-width configuration.

   ☞ This only applies to the transmitter side and not the receiver side of the CMU channel.

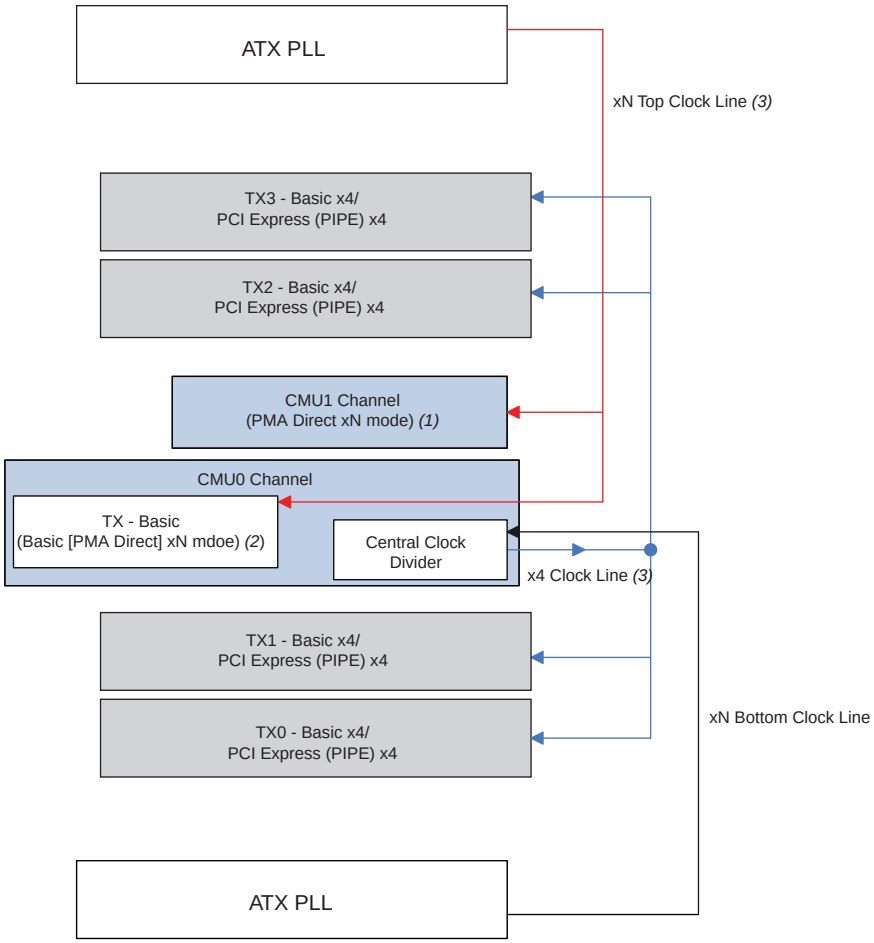
4. Using the receiver side of the CMU channels depends on whether you use `CMU1` PLL or `CMU0` PLL to generate clocks for the bonded ×4 functional mode. If the CMU PLL within the corresponding CMU channel is not available to perform CDR functionality, you cannot configure it as a receiver.
5. If you use ATX PLL to generate clocks for the ×4 bonded functional mode, you can use both the Transmitter and Receiver side of the `CMU0` and `CMU1` channels. You must satisfy the requirements specified in number 3.

Figure 3–8 shows a configuration in which all the transmitter channels in the transceiver block are used.

☞ For XAUI, the option to select ATX PLL is not available.

Figure 3–8 shows the combination of Basic/PCI Express (PIPE) x4 functional mode with Basic (PMA Direct) xN mode within the same transceiver block.

Figure 3–8. Basic ×4 Functional Mode Configuration when Combining Channels *(Note 4)*



Notes to Figure 3–8:

(1) You can configure this channel in Basic (PMA Direct) single-width or double-width mode.

(2) You can configure this channel only in Basic (PMA Direct) single-width mode

(3) The red lines represent the ×N top clock line, the blue lines represent the ×4 clock line, and the black line represents the ×N bottom clock line.

(4) To simplify the illustration, only the transmitter side is shown. PCI Express (PIPE) ×4 refers to PCI Express (PIPE) with the sub protocol set to **Gen1 x4** and **Gen2 x4**.

Bonded x8 Functional Mode

Bonded ×8 functional mode is similar to bonded ×4 functional mode except that the controls are shared from the physical channel 0 of the master transceiver block. The master is the lower of the two adjacent transceiver blocks selected for the ×8 configuration. Therefore, when you an create an instance in this mode, you must assign the logical transmit channel 0 (`tx_dataout[0]` in the instance to the physical channel location 0 in the master transceiver block.

There are specific transceiver blocks that can be paired as master-slave in the ×8 configuration.

The master is the adjacent lower transceiver block. For more information about location requirements, refer to the "Bonded Channel Configurations" section of the *Stratix IV Transceiver Clocking* chapter.

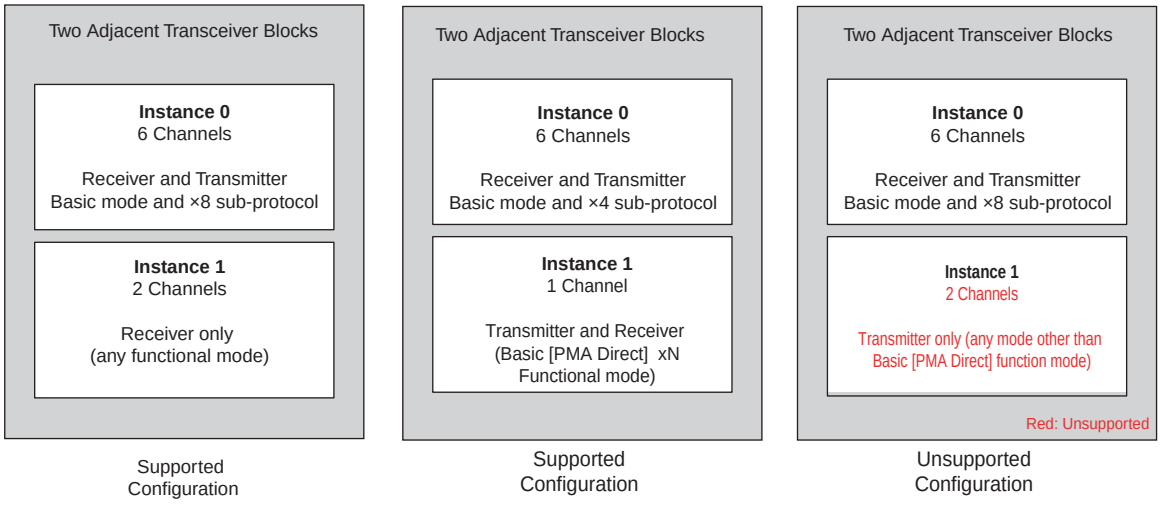
In Basic ×8 functional mode, you can select the number of channels to be less than 8 by setting the **What is the number of channels?** option on the **General** screen. In this instance, you can use the remaining transmitter channels only in Basic (PMA Direct) ×1 or ×N mode. In PCI Express (PIPE) Gen1 ×8 and Gen2 ×8 functional modes, the number of regular channels used is always 8.

The number of remaining transmitter channels (CMU channels or regular channels) in the two transceiver blocks available for use in Basic (PMA Direct) ×1 or ×N mode depends on whether the ×8 functional mode uses CMU PLL or ATX PLL, as described below.

If PCI Express (PIPE) functional mode uses the PCI Express hard IP block, the combination requirements are different. For more information, refer to "Combining Channels Using the PCI Express hard IP Block with Other Channels" on page 3–23.

Each receiver channel configured in Basic ×8 functional mode is clocked independently by the recovered clock from its receiver CDR. You can use the available receiver channels in any configuration. Figure 3–9 shows examples of supported and unsupported configuration in Basic ×8 mode.

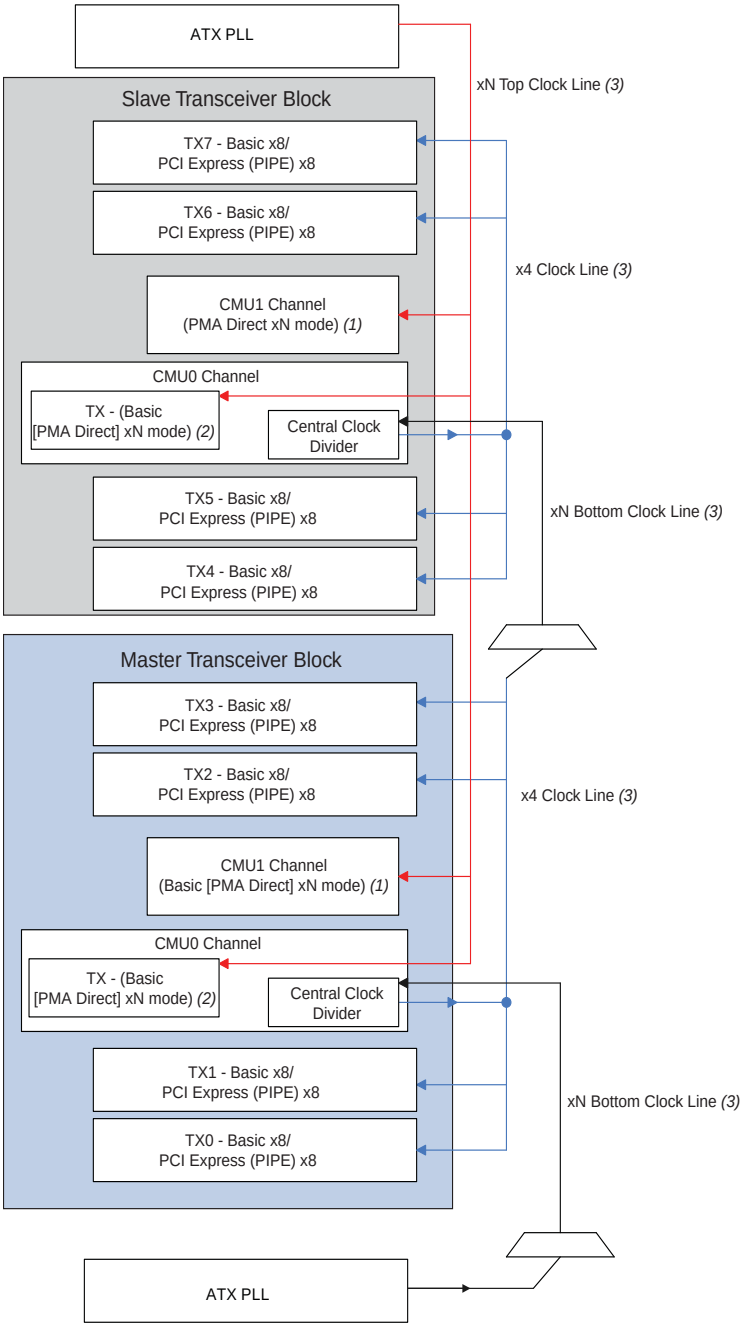
Figure 3–9. Examples of Supported and Unsupported Configurations to Combine Instances in Basic ×8 Mode



When the eight regular channels are used up in bonded ×8 functional mode:

- If ATX PLL is used to generate clocks for the ×8 functional mode shown in Figure 3–10, you can use the four CMU channels (two from the master and slave transceiver block) in Basic (PMA Direct) ×N mode. Within Basic (PMA Direct) ×N mode, you can configure the `CMU0` channels in the master and slave transceiver block only in single-width mode (use the **single-width mode** option in the **General** screen). If a `CMU1` channel or regular channels are available for use, you can use them in Basic (PMA Direct) ×N mode in single-width or double-width configuration.

Figure 3–10. Basic ×8/PCI Express (PIPE) ×8 Functional Mode Configuration when Combining Channels (ATX PLL) *(Note 4)*



Notes to Figure 3–10:

(1) You can configure this channel in Basic (PMA Direct) single-width or double-width mode.

(2) You can configure this channel only in Basic (PMA Direct) single-width mode.

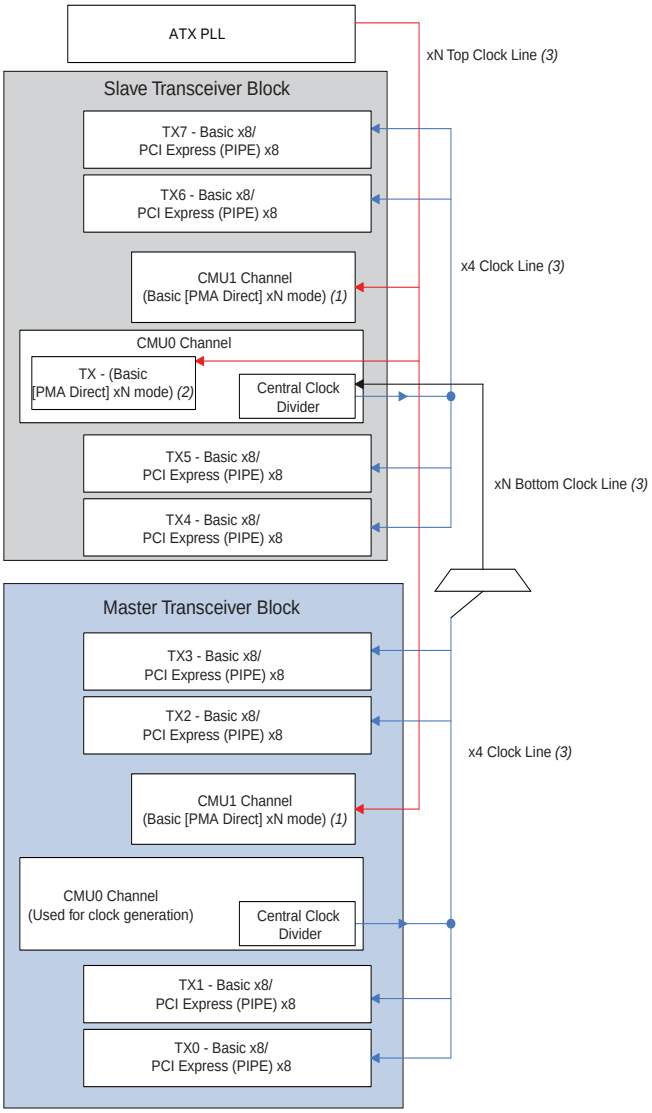
(3) The red lines represent the ×N top clock line, the blue lines represent the ×4 clock line, and the black line represents the ×N bottom clock line.

(4) To simplify the illustration, only the transmitter side is shown. PCI Express (PIPE) ×8 refers to PCI Express (PIPE) with the sub protocol set to **Gen1 ×8** and **Gen2 ×8**.

- If CMU PLL is used to generate clocks for the ×8 bonded functional mode, you can use the `CMU0` channel in the slave transceiver block only in Basic (PMA Direct) ×N mode in the single-width configuration. You can use the `CMU1` channels in both the master and slave transceiver blocks in Basic (PMA Direct) ×N mode in single-width or double-width configuration.

Figure 3–11 shows the Basic ×8 functional mode configuration for these combination restrictions when using CMU PLL.

Figure 3–11. Basic ×8/PCI Express ×8 Functional Mode Configuration when Combining Channels (CMU PLL) *(Note 4)*



Notes to Figure 3–11:

(1) You can configure this channel in Basic (PMA Direct) single-width or double-width mode.

(2) You can configure this channel only in Basic (PMA Direct) single-width mode.

(3) The red lines represent the ×N top clock line, the blue lines represent the ×4 clock line, and the black line represents the ×N bottom clock line.

(4) To simplify the illustration, only the transmitter side is shown. PCI Express (PIPE) ×8 refers to PCI Express (PIPE) with the sub protocol set to **Gen1 ×8** and **Gen2 ×8**.

Combining Channels Configured in Deterministic Latency Mode

The ALTGX MegaWizard Plug-In Manager provides Deterministic Latency mode with two variations (×1 and ×4) to eliminate uncertainty in the transceiver data path. This functional mode provides the **Enable Phase Frequency Detector (PFD) feed back …** option in the **PLL/Ports** screen. If you select this option for ×1, the low-speed parallel clock from the transmitter serializer is fed back to the PFD input of the CMU PLL; for ×4, the output of the low-speed parallel clock from the central clock divider is provided as feed back.

For the ×1 variation, one CMU PLL is required for each transmitter channel in the instance. As a result, in ×1 mode, you can configure only two channels within the transceiver block in this mode. The restrictions for Deterministic Latency mode in ×4 mode are the same as that of the bonded x4 functional mode. For more information, refer to "Bonded ×4 Functional Mode" on page 3–16.

Combining Channels Using the PCI Express hard IP Block with Other Channels

The Stratix IV GX device contains an embedded PCI Express hard IP block that performs the phyMAC, datalink, and transaction layer functionality specified by PCI Express (PIPE) base specification 2.0. Each PCI Express hard IP block is shared by two transceiver blocks. The PCI Express (PIPE) Compiler Wizard provides you the options to configure the PCI Express hard IP block. When enabled, the transceiver channels associated with this block are also enabled.

There are restrictions on combining transceiver channels with different functional and/or protocol modes (for example, Basic mode) within two contiguous transceiver blocks with the channels that use the PCI Express hard IP block. The restrictions depend on the number of channels used (×1 or ×4) and the number of virtual channels (VCs) selected in the PCI Express (PIPE) compiler MegaWizard Plug-In Manager. Table 3–8 shows the restrictions.

When you use the PCI Express hard IP block, there are placement restrictions on the locations of the transceiver channels.

For these channel placement restrictions, refer to the *PCI Express Compiler User Guide*.

Table 3–8. PCI Express hard IP Block Restrictions When Combining Transceiver Channels with Different Functional and/or Protocol Modes (Part 1 of 2) *(Note 1), (2), (7)*

PCI Express (PIPE) Configuration (PCI Express hard IP Options Enabled in the PCIE Compiler Wizard) *(3)*			Transceiver Block 0 *(4)*				Transceiver Block 1 *(5)*			
Link Width	**Lane (Data Interface Width)**	**Virtual Channel (VC)**	**Ch0** *(6)*	**Ch1**	**Ch2**	**Ch3**	**Ch4**	**Ch5**	**Ch6**	**Ch7**
×1	64-bit	1	PCIe ×1	Avail.	Avail.	Avail.	Avail.	Avail.	Avail.	Avail.
		2	PCIe ×1	—	—	—	Avail.	Avail.	Avail.	Avail.
×4	64-bit	1	PCIe ×4				Avail.	Avail.	Avail.	Avail.
		2	PCIe ×4				Avail.	Avail.	Avail.	Avail.
×4	128-bit	1	PCIe ×4				Avail.	Avail.	Avail.	Avail.
		2	PCIe ×4				—	—	Avail.	Avail.

Table 3–8. PCI Express hard IP Block Restrictions When Combining Transceiver Channels with Different Functional and/or Protocol Modes (Part 2 of 2) *(Note 1), (2), (7)*

PCI Express (PIPE) Configuration (PCI Express hard IP Options Enabled in the PCIE Compiler Wizard) *(3)*			Transceiver Block 0 *(4)*				Transceiver Block 1 *(5)*			
Link Width	**Lane (Data Interface Width)**	**Virtual Channel (VC)**	**Ch0** *(6)*	**Ch1**	**Ch2**	**Ch3**	**Ch4**	**Ch5**	**Ch6**	**Ch7**
×8	—	—	PCIe ×8							

Notes to Table 3–8:

(1) Avail. indicates that the channels can be used in other configurations.

(2) An em-dash (—) indicates that the channels are NOT available for use.

(3) The CMU PLL is used for the transmitter side of the channels in this table.

(4) Transceiver block 0—the master transceiver block that provides high-speed serial and low-speed parallel clocks in a PCI Express (PIPE) ×4 or ×8 configuration.

(5) Transceiver block 1—the adjacent transceiver block that shares the same PCI Express hard IP block with transceiver block 0.

(6) The physical channel 0 in the transceiver block. For more information about physical-to-logical channel mapping in PCI Express (PIPE) functional mode, refer to the "×8 Channel Configuration" section in the *Stratix IV Transceiver Clocking* chapter in volume 2 of the *Stratix IV Device Handbook*.

(7) When you the use PCI Express (PIPE) hard IP Block, you cannot configure the CMU channels within the transceiver block as transceiver channels.

For more information about the PCI Express (PIPE) compiler MegaCore functions and hard IP implementation, refer to the *PCI Express Compiler User Guide.*

If you configure a transceiver channel in PCI Express (PIPE) configuration and if an ATX PLL is used to provide clocks for the transmitter side of the channel, you can use the remaining transmitter channels within the same transceiver block only in Basic (PMA Direct) ×1 or ×N mode.

Combining Transceiver Channels with Basic (PMA Direct) Configuration

In this configuration, the transmitter and receiver PCS blocks of a transceiver channel are bypassed and the transceiver channel can run at a maximum of 6.5 Gbps.

For the data rate restrictions in Basic (PMA Direct) mode, refer to the "Transceiver Performance Specifications" section in the *DC and Switching Characteristics* chapter.

Using the Quartus II software, you can configure the two CMU channels and regular transceiver channels in Basic (PMA Direct) mode. The following sections describes the different scenarios for combining Basic (PMA Direct) mode with other transceiver configurations.

For information about the FPGA fabric-transceiver interface, refer to the "Non-Bonded Basic (PMA Direct) Mode Channel Configurations" section in the *Stratix IV Transceiver Clocking* chapter.

Combining Multiple Channels Configured in Basic (PMA Direct) ×1 Configurations

When you configure a transceiver channel in Basic (PMA Direct) ×1 configuration, the Quartus II software requires one of the two CMU PLLs within the same transceiver block to provide high-speed clocks to the transmitter side of the channel (you cannot use the ATX PLL). Therefore, within a transceiver block, you can only combine a maximum of five transceiver channels (using both Transmitter and Receiver) configured in Basic (PMA Direct) ×1 mode (one CMU channel to perform the clock multiplication unit functionality).

You can configure the transmitter side of the CMU channel that uses its CMU PLL for clock generation in Basic (PMA Direct) x1 in single-width or double-width configuration, as shown in Figure 3–17 on page 3–31.

There are multiple ways you can combine channels in Basic (PMA Direct) ×1 mode within the same transceiver block:

- "Multiple Basic (PMA Direct) ×1 Configuration Instances with One Channel per Instance" on page 3–25
- "One Instance in Basic (PMA Direct) ×1 Configuration with Multiple Transceiver Channels" on page 3–25
- "Combining Multiple Instances of TX Only and RX Only Configurations in Basic (PMA Direct) ×1 Mode" on page 3–28
- "Combining Channels Configured in Basic (PMA Direct) ×1 with Non-Basic (PMA Direct) Modes" on page 3–28

Multiple Basic (PMA Direct) ×1 Configuration Instances with One Channel per Instance

If you create multiple instances of Basic (PMA Direct) ×1 with one channel per instance, you can combine them within the same transceiver block. To achieve this combination, refer to the requirements specified in "Multiple Channels Sharing a CMU PLL" on page 3–5 and "General Requirements to Combine Channels" on page 3–3. Note that one CMU PLL within the transceiver block must provide a high-speed clock for the transmitter side of the channels. Therefore, at least one CMU channel (that contains the CMU PLL) within the transceiver block must be available to generate high-speed serial and low-speed parallel clocks for the channels configured in this mode. You can also place the individual instances in this configuration in separate transceiver blocks. For this placement, the Quartus II software enables one CMU PLL per instance.

One Instance in Basic (PMA Direct) ×1 Configuration with Multiple Transceiver Channels

In this case, if the number of channels selected in the instance is less than six, the Quartus II software, by default, combines these channels within the same transceiver block and uses one CMU PLL to provides high-speed clocks. If the number of channels is six or more, the Quartus II software requires two transceiver blocks and two CMU PLLs, one from each transceiver block.

The following two examples show the combinations of channels under two different conditions.

Example 5

Consider a design example configuration with a Basic (PMA Direct) ×1 instance with the number of channels set to **7** in the ALTGX MegaWizard Plug-In Manager. With this setting, the ALTGX MegaWizard Plug-In Manager provides 7 bits of `gxb_powerdown`, `rx_analogreset`, and `pll_powerdown` ports.

In this case, the Quartus II software attempts to combine the five channels in the instance to one transceiver block and the remaining two channels to the second transceiver block, assuming that the `gxb_powerdown` and `pll_powerdown` ports for the five channels are driven from the same logic. Figure 3–12 and Figure 3–13 show the conditions before and after compilation.

Figure 3–12. Logical View of the Instance with Seven Channels Before Compilation for Example 5

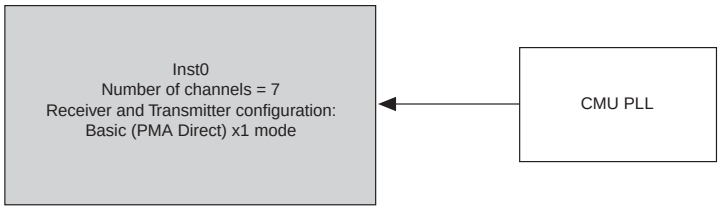
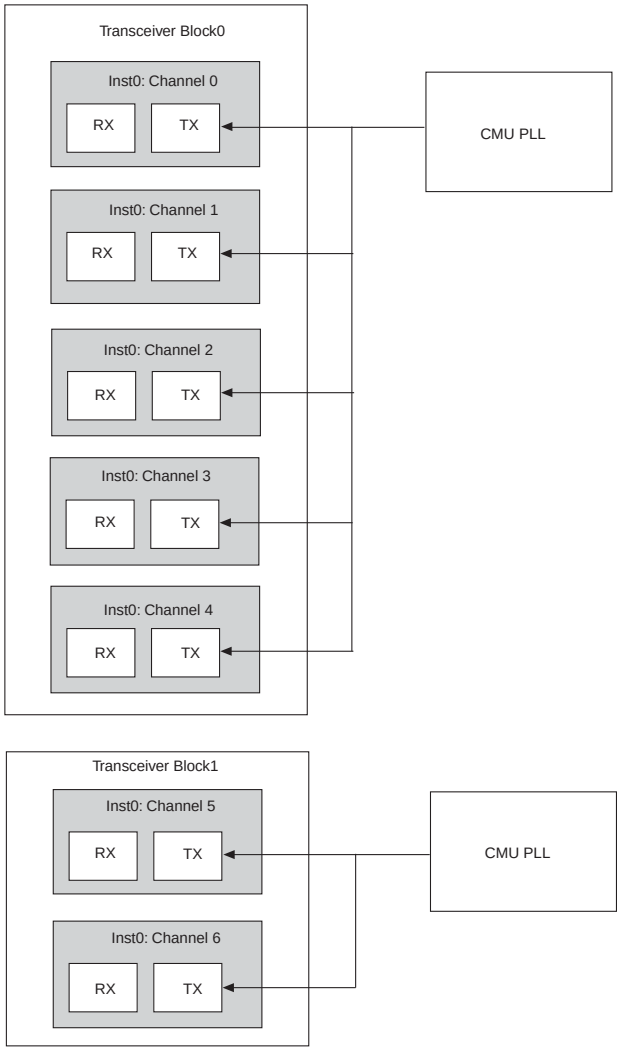


Figure 3–13 shows the conditions after compilation. In this example, the `gxb_powerdown` and `pll_powerdown` ports for channels 0 to 4 and channels 5 to 6 are driven from the same logic.

Figure 3–13. Combined Channels After Compilation for Example 5



If you connect each of the seven bits of the `gxb_powerdown` and `pll_powerdown` ports to different reset control logic, the Quartus II software requires seven transceiver blocks to combine the seven channels in the instance.

Combining Multiple Instances of TX Only and RX Only Configurations in Basic (PMA Direct) ×1 Mode

The Quartus II software allows you to combine instances of TX only and RX only configurations in Basic (PMA Direct) ×1 mode.

You can also combine TX only instances in non-Basic (PMA Direct) ×1 configuration (non-bonded only) and the RX only instance in Basic (PMA Direct) configurations (and vice versa) in the same physical channel. The combination requirements of the instances in Basic (PMA Direct) ×1 configuration are similar to that of non-Basic (PMA Direct) configuration. For more information, refer to the "Combining Transmitter Channel and Receiver Channel Instances" on page 3–10.

Combining Channels Configured in Basic (PMA Direct) ×1 with Non-Basic (PMA Direct) Modes

You can combine a transceiver channel instance configured in Basic (PMA Direct) ×1 configuration with instances set up in non-Basic (PMA Direct) configurations (for example, GIGE and SDI within the same transceiver block).

If the CMU PLL configuration for the Basic (PMA Direct) ×1 configuration and the non-Basic (PMA Direct) configuration instances meet the requirements specified in "Multiple Channels Sharing a CMU PLL" on page 3–5 and "General Requirements to Combine Channels" on page 3–3, the Quartus II software uses a single CMU PLL for these two instances. In addition, to share the same CMU PLL between the two instances, the channel reconfiguration option must not be enabled in the instance setup in non-Basic (PMA Direct) configuration.

Example 6

Consider the example design shown in Table 3–9 for Basic (PMA Direct) ×1 and non-Basic (PMA Direct) configurations at the same data rate.

Table 3–9. Basic (PMA Direct) ×1 and Non-Basic (PMA Direct) Configurations at the Same Data Rate for Example 6

Instances	Configuration	CMU PLL Base Data Rate	Transmitter Channel Effective Data Rate	Input Reference Clock Frequency
`inst0`	GIGE	1.25 Gbps	1.25 G	125 MHz (assume `refclk0`)
`inst1`	Basic (PMA Direct) ×1 (four channels)	1.25 Gbps	1.25 G	Same as `inst0`

Figure 3–14 shows Basic (PMA Direct) ×1 and non-Basic (PMA Direct) configurations before compilation.

Figure 3–14. Logical View of the Instances in Basic (PMA Direct) ×1 and Non-Basic (PMA Direct) Configurations Before Compilation for Example 6

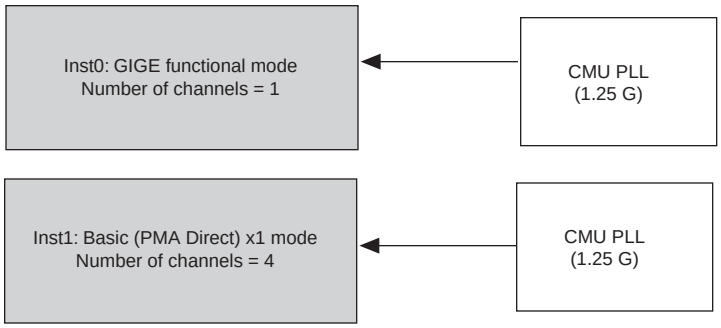
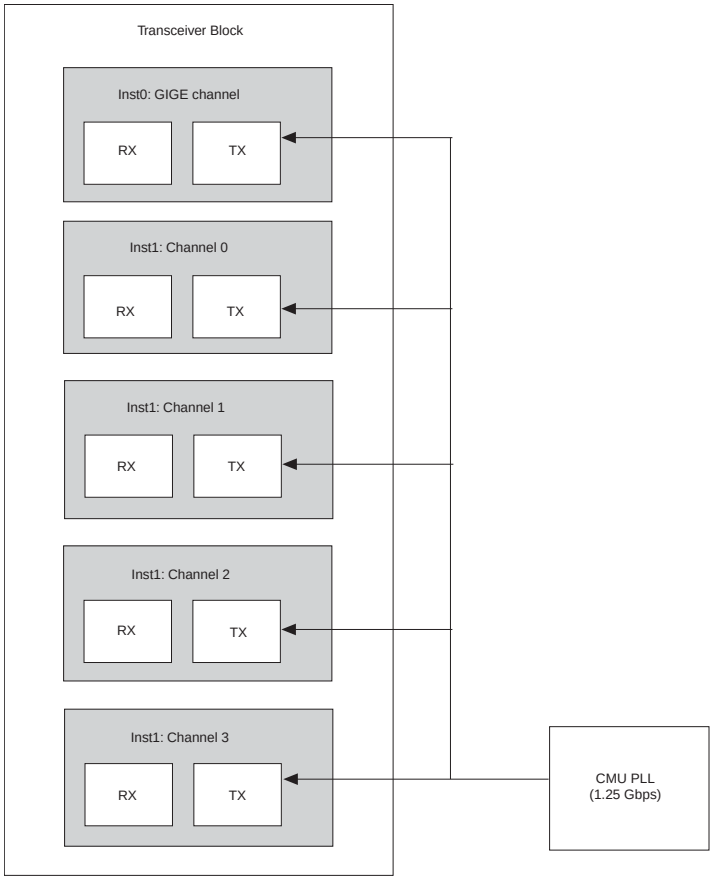


Figure 3–15 shows Basic (PMA Direct) ×1 and non-Basic (PMA Direct) configurations after compilation.

Figure 3–15. Combining Basic (PMA-Direct) ×1 and Non-Basic (PMA Direct) Configurations After Compilation for Example 6



Example 7

Consider the example design shown in Table 3–10 for Basic (PMA Direct) ×1 and non-Basic (PMA Direct) configurations at different data rates.

Table 3–10. Basic (PMA Direct) ×1 and Non-Basic (PMA Direct) Configurations at Different Data Rates for Example 7

Instances	Configuration	CMU PLL Base Data Rate	Transmitter Channel Effective Data Rate	Input Reference Clock Frequency
`inst0`	GIGE	1.25 Gbps	1.25 Gbps	125 MHz (assume `refclk0`)
`inst1`	Basic (PMA Direct) ×1 (three channels in **Receiver and Transmitter** configuration)	2 Gbps	2 Gbps	`refclk1`
`inst2`	Basic (PMA Direct) ×1 (one channel in **Transmitter Only** configuration)	2 Gbps	2 Gbps	`refclk1`

Figure 3–16 shows Basic (PMA Direct) ×1 and non-Basic (PMA Direct) configurations before compilation.

The data rate configurations of the two CMU PLLs are different.

Figure 3–16. Logical View of the Basic (PMA Direct) ×1 and Non-Basic (PMA Direct) Configurations Before Compilation for Example 7

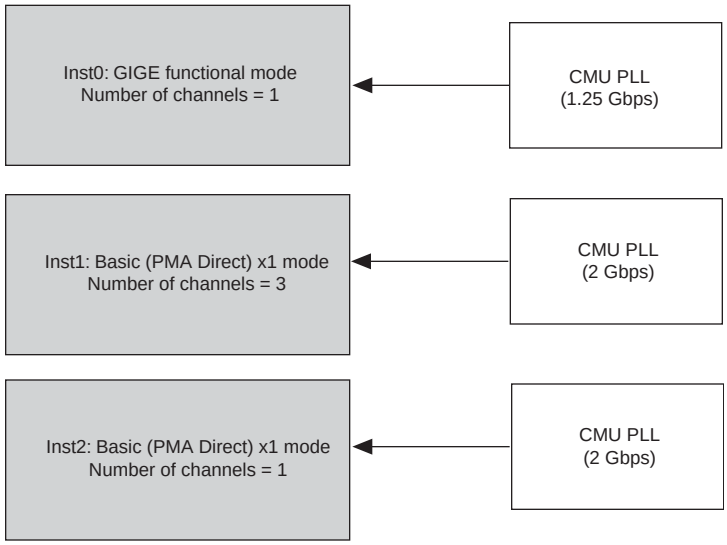
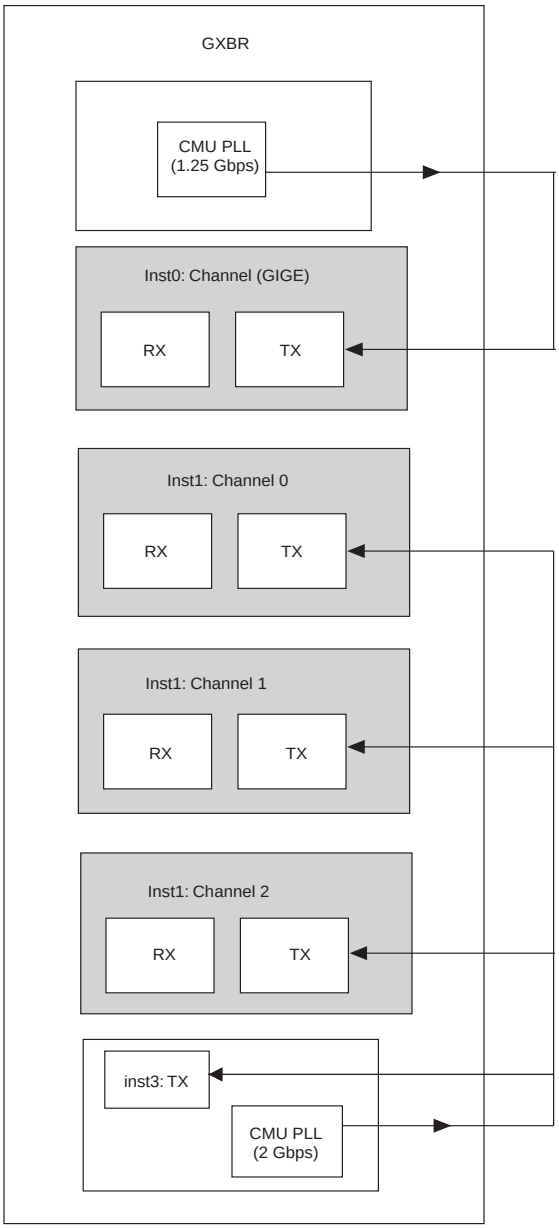


Figure 3–17 shows Basic (PMA Direct) ×1 and non-Basic (PMA Direct) configurations after compilation.

Figure 3–17. Combining Basic (PMA Direct) ×1 and Non-Basic (PMA Direct) Instances in a Transceiver Block After Compilation for Example 7



Key Observations

To combine the these instances, two CMU PLLs are required due to the different data rates. Therefore, two CMU channels must be available to enable their respective CMU PLLs. Note that inst3 uses the transmit side of the CMU channel that uses the CMU PLL for clock generation.

Basic (PMA Direct) ×N Configuration

When you configure a transceiver channel in Basic (PMA Direct) ×N configuration, you can enable the Quartus II software to use the ×N lines to provide clocks to the transmitter channels, as shown in Figure 3–18.

The following are the possible sources driving the ×N clock lines:

- The `CMU0` central divider within the `CMU0` channel. Only the `CMU0` clock divider block can drive the ×N clock lines. Either the `CMU0` PLL or `CMU1` PLL can drive the central clock divider block.

  To understand the input clock connections to the central clock divider block, refer to the "CMU0 Channel" section in the *Stratix IV Transceiver Architecture* chapter.

- The ATX PLL block

Channel Placement in a Basic (PMA Direct) ×N Mode Instance

If you compile a design with a transceiver instance configured in Basic (PMA Direct) xN mode, the Quartus II software, by default, places these channels contiguously.

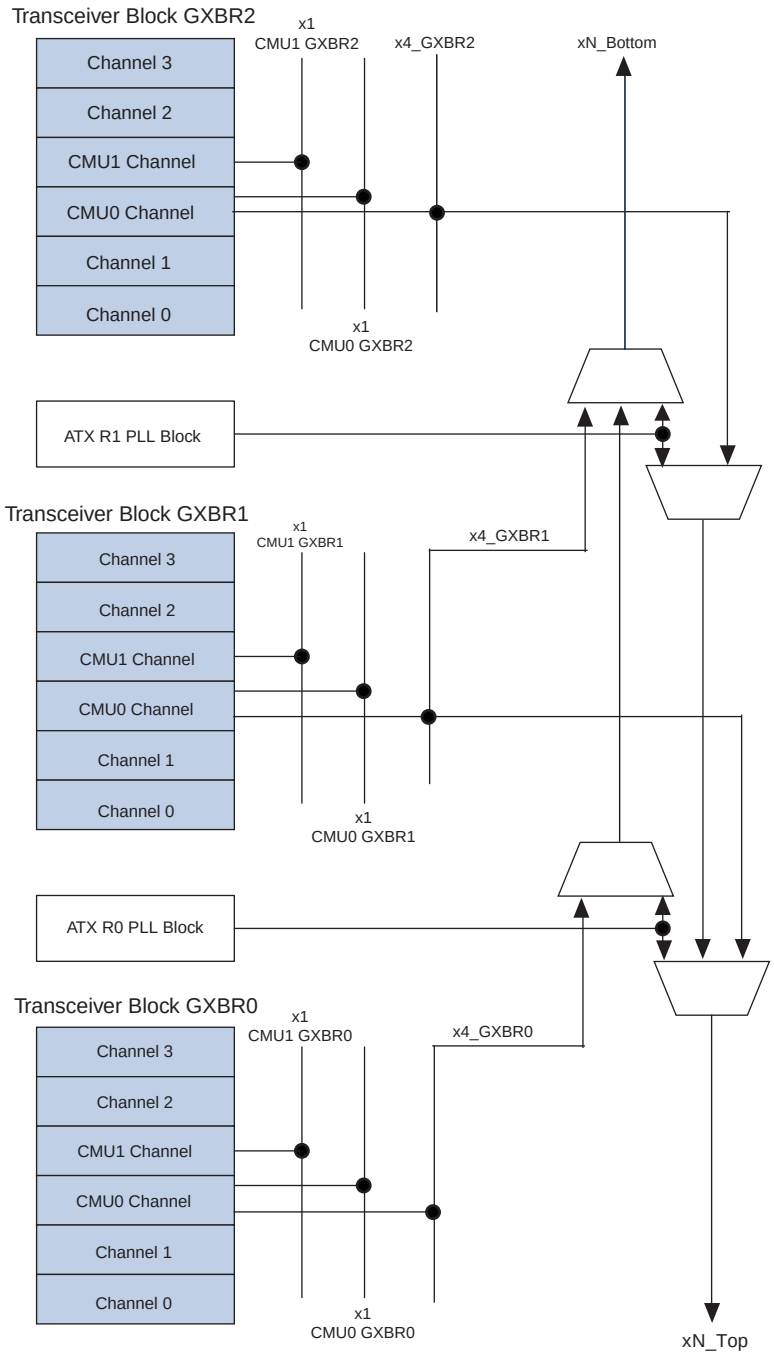
You can force the placement of the transceiver channels across multiple transceiver blocks on the same side of the device by assigning pins to the transmitter and receiver serial ports.

The logical channel 0 of the Basic (PMA Direct) ×N mode instance does not have to be assigned to the physical channel 0 of a transceiver block. The logical channel 0 of an instance with multiple channels is `tx_dataout[0]` or `rx_datain[0]`, which are the serial transmit and receive ports provided by the ALTGX MegaWizard Plug-In Manager. When you assign pins, you are not required to assign `tx_dataout[0]` to the location of physical channel 0 in the transceiver block to compile your design.

This is not the case if you have a PCI Express (PIPE) ×4 configuration where `tx_dataout[0]`and `rx_datain[0]` must be assigned to physical channel 0 of the transceiver block.

Figure 3–18 shows the different drivers of the ×N_Top and ×N_Bottom clock lines.

Figure 3–18. The ×N_Top and ×N_Bottom Clock Line Connections



Examples of Combining Multiple Instances of Basic (PMA Direct) ×N Modes

The following section describes combining multiple transceiver channel instances in Basic (PMA Direct) ×N mode. Configuration examples include transceiver channels with different data rates, configurations in Basic (PMA Direct) ×N mode with non-Basic (PMA Direct) and ATX PLL, and unsupported configurations.

Example 8

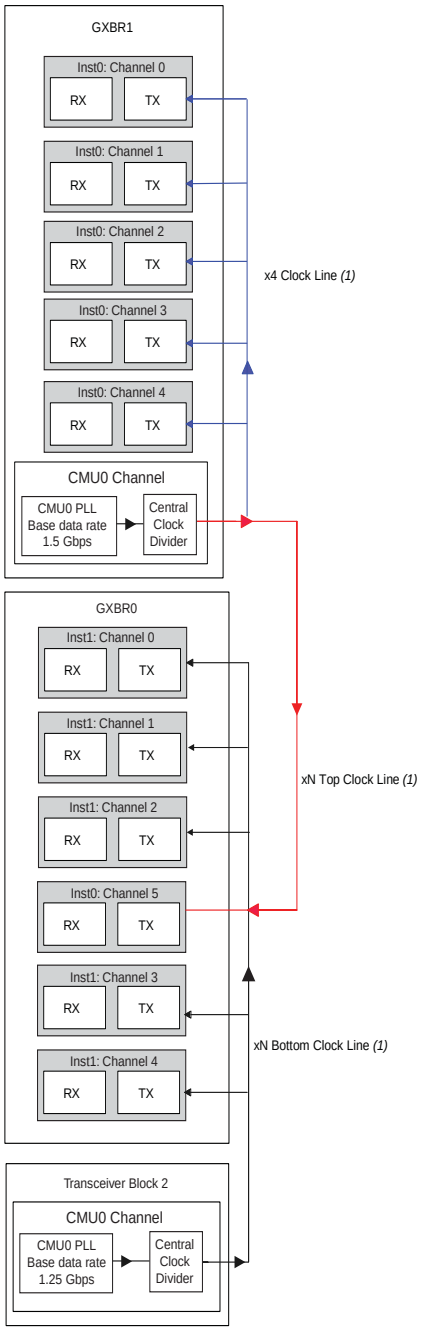
Consider the configuration for the two instances shown in Table 3–11 when combining transceiver channels in Basic (PMA Direct) ×N mode with different data rates.

Table 3–11. Combining Transceiver Channels in Basic (PMA Direct) ×N Configuration with Different Data Rates for Example 8

User Defined Instance Name	Number of Channels	Effective Data Rate	Configuration
`inst 0`	6	1.5 Gbps	Receiver and transmitter
`inst 1`	5	1.25 Gbps	Receiver and transmitter

You can place channels within a given instance non contiguously as shown in Figure 3–19.

Figure 3–19. Non-Contiguous Placements of Channels Using Different CMU PLLs for Example 8



Note to Figure 3–19:

(1) The red lines represent the ×N top clock line, the blue lines represent the ×4 clock line, and the black lines represent the ×N bottom clock line.

Key Observations:

- Note that channel 5 in `inst0` is placed in transceiver block 1 and receives the high-speed clock through the `×N_Top` clock line.
- Some of the channels in transceiver block 1 receive their high-speed clock from the `×N_Bottom` clock line. Because the `×N_Top` and `×N_Bottom` lines are separate, this scenario is allowed. To understand the clock multiplexer on the ×N clock lines, refer to Figure 3–18 on page 3–33.

Combining Channels Configured in Basic (PMA Direct) ×N Configuration with Non-Basic (PMA Direct) Configurations

The Quartus II software only allows a combination of a transceiver channel instances configured in Basic (PMA Direct) ×N mode with instances in non-Basic (PMA Direct) configurations; for example, GIGE and SDI.

Example 9

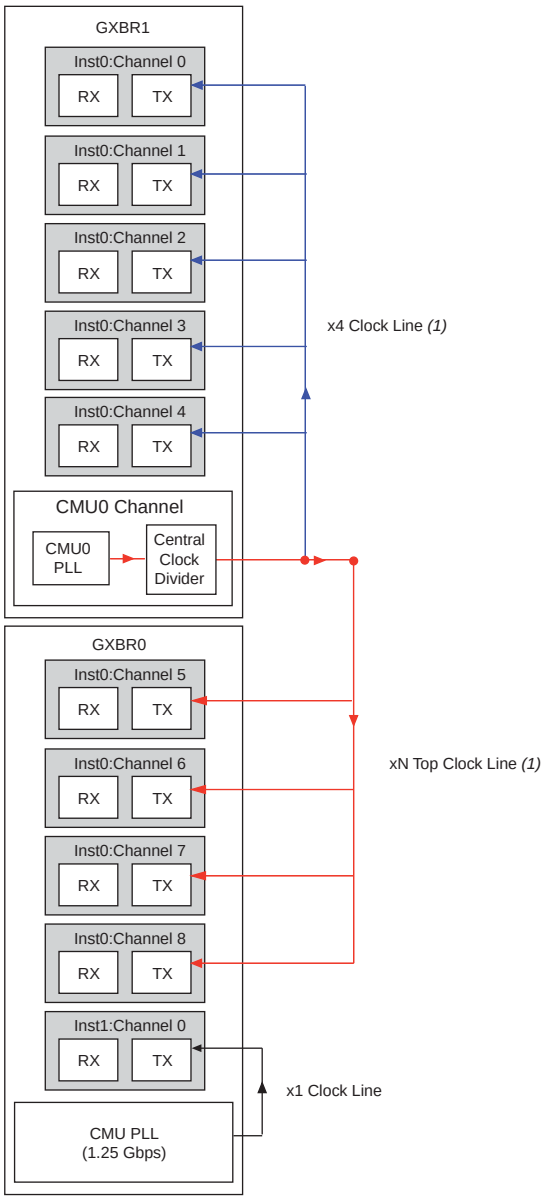
Consider the example design shown in Table 3–12 for the two instances when combining a Basic (PMA Direct) ×N configuration with a non-Basic (PMA Direct) configuration using a CMU PLL.

Table 3–12. Combining Basic (PMA Direct) ×N Configuration with Non-Basic (PMA Direct) Configuration Using CMU PLL for Example 9

User Defined Instance Name	Number of Channels	Effective Data Rate	Configuration	Functional Mode
`inst 0`	9	1.5 Gbps	Receiver and transmitter	Basic (PMA Direct) ×N
`inst 1`	1	1.25 Gbps	Receiver and transmitter	GIGE

You can place these two instances in two transceiver blocks, as shown in Figure 3–20.

Figure 3–20. Combining Basic (PMA Direct) ×N Configuration with Non-Basic (PMA Direct) Configuration Using CMU PLL in Two Transceiver Blocks For Example 9



Note to Figure 3–20:

(1) The red lines represent the ×N top clock line, the blue lines represent the ×4 clock line, and the black line represents the ×N bottom clock line.

Example 10

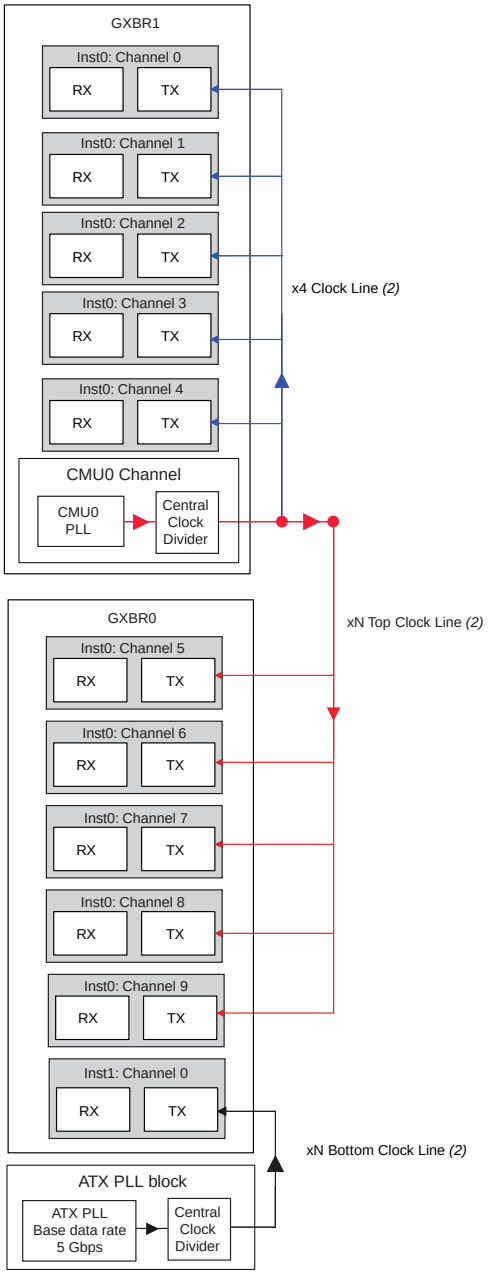
Consider the example design shown in Table 3–13 when combining a Basic (PMA Direct) ×N configuration with a non-Basic (PMA Direct) configuration using an ATX PLL.

Table 3–13. Combining Basic (PMA Direct) ×N Configuration with Non-Basic (PMA Direct) Configuration Using ATX PLL for Example 10

User Defined Instance Name	Number of Channels	Effective Data Rate	Configuration	Functional Mode
`inst 0`	10	1.5 Gbps	Receiver and transmitter	Basic (PMA Direct) ×N configuration
`inst 1`	1	5 Gbps	Receiver and transmitter	Basic mode (PCS+PMA) using ATX PLL

In this case, the ATX PLL provides the high-speed clock to the transmitter channel of `inst 1`. Therefore, you can combine 10 channels of `inst 0` and one channel of `inst 1` in two transceiver blocks, as shown in Figure 3–21

Figure 3–21. Combining Basic (PMA Direct) ×N Configuration with Non-Basic (PMA Direct) Configuration Using an ATX PLL for Example 10 *(Note 1)*



Notes to Figure 3–21:

(1) The ATX PLL provides the high-speed clock to channel 0 of `inst1`.

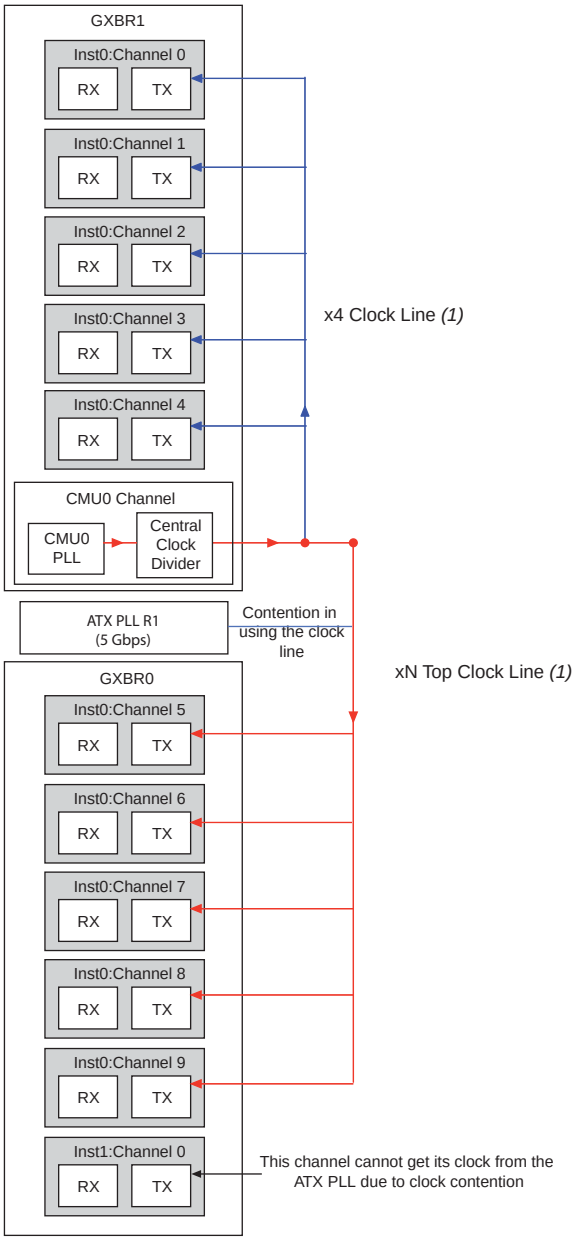
(2) The red lines represent the ×N top clock line, the blue lines represent the ×4 clock line, and the black line represents the ×N bottom clock line.

You can also combine channels configured in Basic (PMA Direct) ×N mode with bonded ×4 and ×8 functional modes. For example scenarios, refer to Figure 3–8 on page 3–19 and Figure 3–10 on page 3–21.

Example 11

Consider the unsupported placement design example shown Figure 3–22. The placement is unsupported because of the ×N_Top clock line contention between the ATX PLL and the CMU0 PLL in transceiver block 0.

Figure 3–22. Unsupported Placement Due to ×N Clock Line Contention for Example 11



Note to Figure 3–22:

(1) The red lines represent the ×N top clock line and the blue lines represent the ×4 clock line.

Combination Requirements When Channel Reconfiguration is Enabled

You can configure a transmitter channel to:

- Switch to an alternate CMU PLL present within the same transceiver block.
- Switch to multiple TX PLLs (CMU or ATX PLLs) that are present outside the transceiver block.

For more information about setting up the transceiver to enable one of these three options, refer to the *Stratix IV Dynamic Reconfiguration* chapter.

The section describes the combination requirements for an instance that is configured in one of the three options mentioned above with other instances.

Combination Requirements When the Use Alternate CMU PLL Option is Selected

If you create a transceiver instance by selecting the **use alternate CMU PLL** option, the Quartus II software uses two CMU PLLs. If you intend to combine other transmitter instances within the same transceiver block, the CMU PLLs must be shared between multiple instances. To enable the Quartus II software to share the CMU PLLs between these instances, you must:

1. Select the **user alternate CMU PLL** option in all the instances.
2. Set the same **PLL logical reference index** value for the similar PLLs between the two instances (similar PLLs are the ones that have the same data rate, input clock frequency, and bandwidth setting).
3. Create a **GXB TX PLL Reconfiguration** group setting in the assignment editor and assign the same value for both instances.

Table 3–14 lists the assignment for the first instance (Instance 1).

Table 3–14. Assignment for the First Instance—Instance 1

Assignment	Setting
To	`<provide the tx_dataout port name of first instance>`
Assignment Name	GXB TX PLL Reconfiguration group setting
Value	<any number>

Table 3–15 lists the assignment for the second instance (Instance 2).

Table 3–15. Assignment for the Second Instance—Instance 2

Assignment	Setting
To	`<provide the tx_dataout port name of first instance>`
Assignment Name	GXB TX PLL Reconfiguration group setting
Value	<same number as that of the first instance>

Ensure that the requirements specified in the "General Requirements to Combine Channels" on page 3–3 and "Sharing CMU PLLs" on page 3–5 sections are met.

Example 12 shows the requirements.

Example 12

Consider that you intend to run four channels within the transceiver block to switch between GIGE and SONET OC48 data rates. Assume that by default two channels run at GIGE data rates and the other two channels run at SONET OC48 data rates.

Assume that Instance1 with two channels running at GIGE data rate is created with the configuration, as shown in Table 3–16.

Table 3–16. Combining Requirements with the 'Use Alternate CMU PLL' Option Enabled—Instance 1 for Example 12

PLL	Data Rate	Input Reference Clock	PLL Logical Reference Index
Main PLL	1.25 Gbps	125 MHz	0
Alternate PLL	2.488 Gbps	155.5 MHz	1

Create Instance 2 with the following parameters to enable the Quartus II software to share CMU PLLs between the two instances.

Table 3–17 lists the required parameters to be set for Instance 2

Table 3–17. Combining Requirements with the 'Use Alternate CMU PLL' Option Enabled—Instance 2 for Example 12

PLL	Data Rate	Input Reference Clock	PLL Logical Reference Index
Main PLL	2.488 Gbps	155.5 MHz	1
Alternate PLL	1.25 Gbps	125 MHz	0

Table 3–18 lists the assignment for the **GXB TX PLL Reconfiguration** group for Instance 1 when you compile the design.

Table 3–18. Assignment for the GXB TX PLL Reconfiguration Group for Instance 1

Assignment	Setting
To	`tx_dataout_instance1[0]` Note that the number of channels in this instance is 2. You can use any one of the channel port names within this instance for this assignment.
Assignment Name	GXB TX PLL Reconfiguration group setting
Value	6

Table 3–19 lists the assignment for the **GXB TX PLL Reconfiguration** group for Instance 2 when you compile the design.

Table 3–19. Assignment for the GXB TX PLL Reconfiguration Group for Instance 2

Assignment	Setting
To	`tx_dataout_instance2[1]` You can use any one of the channel port names within this Instance for this assignment.
Assignment Name	GXB TX PLL Reconfiguration group setting
Value	6

Key Observations

- The Main PLL in Instance 1 is configured for GIGE data rates because this is the data rate that you intend to run the first instance.
- The main PLL in Instance 2 is configured for SONET OC48 data rates because this is the data rate that you intend to run the second channel.
- Note that the PLL logical reference index values for similar PLLs in Instance 1 and Instance 2 are the same.
- The **GXB TX PLL Reconfiguration** group setting value for `tx_dataout` of instance 1 and instance 2 are the same.

Combination Requirements When Multiple TX PLLs are Used

This scenario describes transceiver configurations that have the **use additional CMU/ATX Transmitter PLLs from outside the transceiver block** option in the **reconfig screen** enabled.

If you create a transceiver instance using the above option and would like to share the CMU PLLs or ATX PLL with other instances, ensure that you have met the following requirements:

- Select the **use additional CMU/ATX Transmitter PLLs from outside the transceiver block** option in other instances.

  ☞ The number of additional PLLs selected (in the **How many additional PLLs are used** option) can be different between instances.

- The PLL logical reference index value of the similar PLLs that you intend to combine must be the same in all the instances considered.

  ☞ Similar PLLs are the ones that have the same data rate, input clock frequency, and bandwidth setting.

- Assign the same **GXB TX PLL Reconfiguration** group setting value for the `tx_dataout` ports of all the instances. This is explained in "Combination Requirements When the Use Alternate CMU PLL Option is Selected" on page 3–41.
- Ensure that the requirements specified in "General Requirements to Combine Channels" on page 3–3, "Sharing CMU PLLs" on page 3–5, and "Sharing ATX PLLs" on page 3–9 are met.

If you create an instance using the **use additional CMU/ATX Transmitter PLLs from outside the transceiver block** option and place your transmitter channels in one transceiver block (for example, QL1) and you use a CMU/ATX PLL from another transceiver block (for example, QL0), the channels (if used) in QL0 must be connected to the same reconfiguration controller as that of QL1. Example 13 shows an instance using multiple PLLs.

Example 13

Consider that the following 12-channel design is targeted for a THREE transceiver block per side device. The requirements for this design are:

1. Four transceiver channels to switch independently between four protocol data rates (SONET OC48, FC 2G, GIGE, and OTU1).
2. Four transceiver channels to operate in SONET OC48 data rate.
3. Four transceiver channels to operate in GIGE data rate.

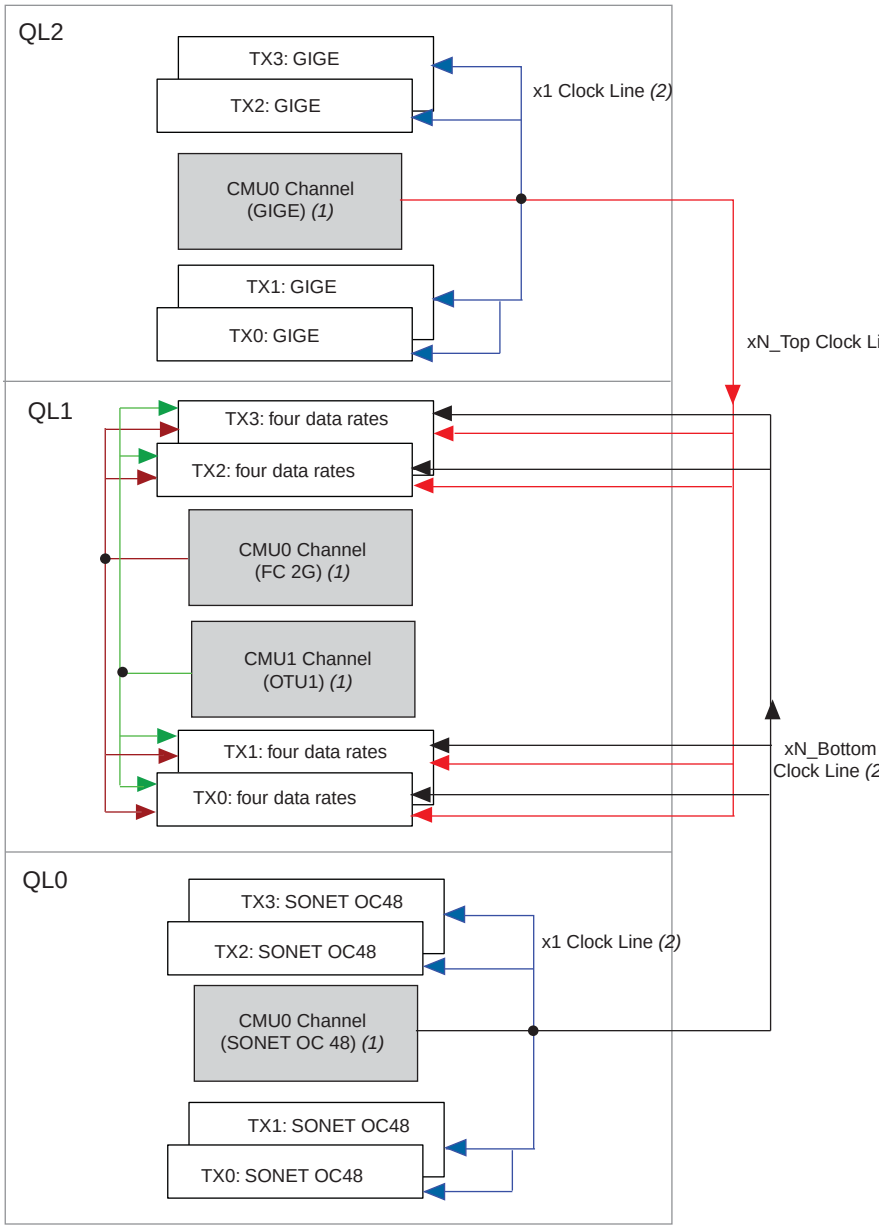
To implement step 1, you need four TX PLLs. Place the four channels in the middle transceiver block (QL1—the left side was chosen for illustration purposes), and provide one CMU PLL from QL0 for the SONET OC48 data rate and one CMU PLL from QL2 for the GIGE data rate. Use the two CMU PLLs from QL1 for the FC 2G and OTU1 data rates.

To implement step 2, note that the CMU PLL in QL0 already provides the SONET OC48 data rate. Therefore, use the four channels in QL0 to run the SONET OC48 data rate.

To implement step 3, note that the CMU PLL in QL2 already provides the GIGE data rate. Therefore, use the four channels in QL2 to run the GIGE data rate.

Figure 3–23 shows the configuration for Example 13.

Figure 3–23. Three Transceiver Block Configuration for Example 13



Notes to Figure 3–23:

(1) CMU channels are used for clock generation.

(2) The red lines represent the ×N top clock line, the blue lines represent the ×1 clock line, the black lines represent ×N bottom clock, the green lines represents the `CMU1` channel, and the brown lines represent the `CMU0` channel.

Create three Instances for steps 1, 2, and 3 with the following parameters:

Instance 1

- Select the **use additional CMU/ATX Transmitter PLLs from outside the transceiver block** option.
- Number of additional PLLs: 3 (Table 3–20)

Table 3–20. Instance 1 for Example 13

PLL	Data Rate	PLL Logical Reference Index
Main PLL	FC 2G	0
PLL1	OTU1	1
PLL2	GIGE	2
PLL3	SONET OC48	3

Instance 2

- Select the **use additional CMU/ATX Transmitter PLLs from outside the transceiver block** option.
- Number of additional PLLs: 0 (Table 3–21)

Table 3–21. Instance 2 for Example 13

PLL	Data Rate	PLL Logical Reference Index
Main PLL	SONET OC48	3

Instance 3

- Select the **use additional CMU/ATX Transmitter PLLs from outside the transceiver block** option.
- Number of additional PLLs: 0 (Table 3–22)

Table 3–22. Instance 3 for Example 13

PLL	Data Rate	PLL Logical Reference Index
Main PLL	GIGE	2

For more information, refer to "Combination Requirements When the Use Alternate CMU PLL Option is Selected" on page 3–41.

Combining Transceiver Channels When the Adaptive Equalization (AEQ) is Enabled

To enable the AEQ feature in a transceiver channel, select the **Enable Adaptive Equalization** option in the **Reconfig** screen of the ALTGX MegaWizard Plug-In Manager. When you select this option, the `aeq_fromgxb` and `aeq_togxb` ports are enabled.

For more information about initiating the AEQ feature, refer to the "Adaptive Equalization (AEQ)" section in the *Stratix IV Dynamic Reconfiguration* chapter.

This section describes the requirements to combine transceiver channels when you enable the AEQ feature.

You are not required to enable AEQ in all instances to combine them within the same transceiver block. When you instantiate the reconfiguration controller (`ALTGX_Reconfig`), the `aeq_fromgxb` and `aeq_togxb` ports available depend on the setting in the **what is the number of channels controlled by the reconfig controller** option. In configurations where AEQ is enabled on some of the transceiver channels that are connected to the same reconfiguration controller, the reconfiguration controller instance has additional `aeq_fromgxb` ports. To compile the design successfully, connect the unused `aeq_fromgxb` ports to **0**. "Example 14" shows the configuration.

Example 14

Consider that you have two ALTGX instances, Instance 1 and Instance 2 with one channel each. Assume that only Instance 1 has the **Enable adaptive equalization** option enabled.

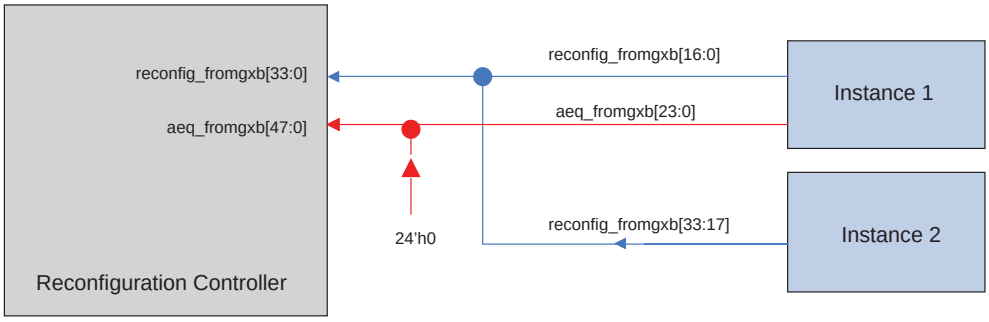
Because there are two instances, the starting channel numbers of Instance 1 and Instance 2 are spaced four apart (0 and 4, respectively).

In the ALTGX_Reconfig Instance, set the **what is the number of channels controlled by the reconfig controller** option to **8**. The ALTGX_Reconfig Instance has 48 bits for the `aeq_fromgxb` port (24 bits per 4 channels). Instance 1 has the `aeq_fromgxb[23:0]` port because AEQ is enabled. Instance 2 does not have this port. Because Instance 1 has the starting channel number of 0, connect `aeq_fromgxb` of instance 1 to `aeq_fromgxb[23:0]` of the ALTGX_Reconfig Instance and tie `aeq_fromgxb[47:24]` to 0.

For more information about setting this parameter, refer to the *Stratix IV Dynamic Reconfiguration* chapter.

Figure 3–24 shows the required connection for the `aeq_fromgxb` port.

Figure 3–24. Required Connection for the aeq_fromgxb Port



The top 24 bits of `aeq_fromgxb` are tied to 0. This is because the logical channel address of Instance 1 starts at 4. Therefore, the top 24 bits of `aeq_fromgxb` corresponds to Instance 2.

Combination Requirements for Stratix IV GT Devices

Stratix IV GT devices allow configuring multiple protocols or data rates in the same transceiver block. For common protocols supported by both Stratix IV GX and Stratix IV GT devices, as well as for Basic functional mode at data rates between 2.488 Gbps and 8.5 Gbps, Stratix IV GT devices follow the same transceiver channel placement rules as Stratix IV GX devices.

Placement Rules for Transceiver Channels at 9.9 Gbps to 10.3125 Gbps

You can use either the CMU PLL or the 10G ATX PLL to generate transceiver clocks for channels configured at data rates between 9.9 Gbps and 10.3125 Gbps.

If you use a 10G ATX PLL to generate transceiver clocks for any channel configured between 9.9 Gbps and 10.3125 Gbps within a transceiver block, the remaining channels in the same transceiver block must either be unused or must be configured at the same data rate and clocked by the same 10G ATX PLL.

If you use a CMU PLL to generate transceiver clocks for any channel configured between 8.5 Gbps and 10.3125 Gbps within a transceiver block, the remaining channels in the same transceiver block may be configured at a different data rate and clocked by another CMU PLL or 6G ATX PLL. In this case, Stratix IV GT devices follow the same transceiver channel placement rules as Stratix IV GX devices.

Placing transceiver channels clocked by another PLL in the same transceiver block as a 10G channel can result in higher transmitter output jitter on the 10G channel. The amount of additional jitter is pending characterization.

Summary

The following is a summary for configuring multiple protocols and data rates in a transceiver block:

- You can run each transceiver channel at independent data rates or in independent protocol functional modes.
- Each transceiver block consists of two CMU PLLs that provide clocks to run the transmitter channels within the transceiver block.
- To enable the Quartus II software to combine multiple instances of transceiver channels within a transceiver block, follow the rules specified in "General Requirements to Combine Channels" on page 3–3 and "Sharing CMU PLLs" on page 3–5.
- You can reset each CMU PLL within a transceiver block using a `pll_powerdown` signal. For each transceiver instance, the ALTGX MegaWizard Plug-In Manager provides an option to select the `pll_powerdown` port. If you want to share the same CMU PLL between multiple transceiver channels, connect the `pll_powerdown` ports of the instances and drive the signal from the same logic.
- If you enable the PCI Express (PIPE) hard IP block using the PCI Express (PIPE) compiler, the Quartus II software has certain requirements for using the remaining transceiver channels within the transceiver block in the other configurations. For more information, refer to "Combining Channels Using the PCI Express hard IP Block with Other Channels" on page 3–23.
- The Quartus II software supports two kinds of Basic (PMA Direct) configurations (×1 and ×N).
- If you use Basic (PMA Direct) ×1 configuration, you must use the CMU PLL within the same transceiver block.

Document Revision History

Table 3–23 shows the revision history for this chapter.

Table 3–23. Document Revision History (Part 1 of 2)

Date and Document Version	Changes Made	Summary of Changes
November 2009, v4.0	■ Added "Sharing ATX PLLs" on page 3–9, "Combination Requirements When Channel Reconfiguration is Enabled" on page 3–41, "Combining Transceiver Channels When the Adaptive Equalization (AEQ) is Enabled" on page 3–46, and "Combination Requirements for Stratix IV GT Devices" on page 3–48. ■ Added Figure 3–8, Figure 3–10, Figure 3–11, Figure 3–23, and Figure 3–24. ■ Updated all other sections. ■ Added Stratix IV GT information. ■ Updated graphics. ■ Minor text edits.	—

Table 3–23. Document Revision History (Part 2 of 2)

June 2009, v3.1	■ Updated Table 3–7. ■ Minor text edits.	—
March 2009, v3.0	■ Updated sections "Combining Channels Using the PCI Express Hard IP Block with Other Channels" on page 3–17, "Convention Used" on page 3–21, "PMA Direct Mode Restrictions" on page 3–22, "Multiple 'PMA Direct x1' Configuration Instances with One Channel per Instance" on page 3–22, "Combining Multiple Instances of TX Only and RX Only PMA-Direct x1 Configurations" on page 3–26, "Combining Transceiver Channels with PMA Direct Configuration" on page 3–21. ■ Updated Table 3–7. ■ Updated Figure 3–19.	—
November 2008 v2.0	■ Updated "Transmitter Buffer Voltage (VCCH)" on page 3–2 ■ Added "reconfig_fromgxb and reconfig_togxb Ports" on page 3–3 ■ Updated Figure 3–7 ■ Added "Basic x8 Mode" on page 3–15 ■ Added Figure 3–8 ■ Updated Table 3–7	—
May 2008 v1.0	Initial release.	—

4. Reset Control and Power Down

SIV52004-4.0

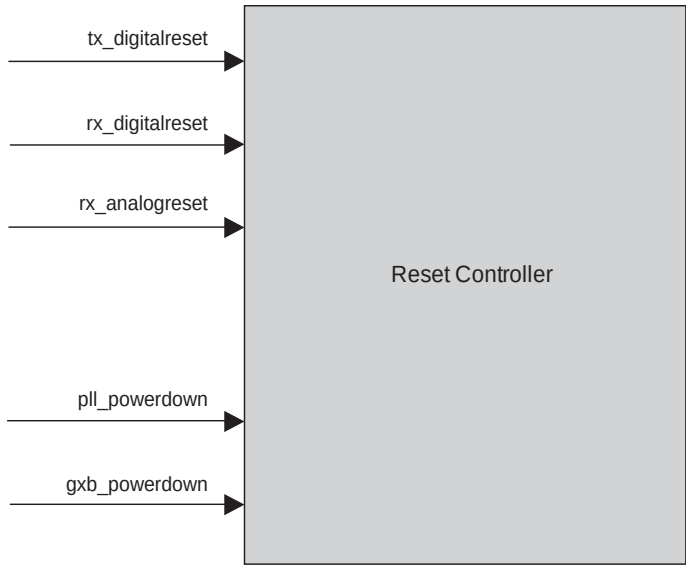
Stratix® IV GX devices offer multiple reset signals to control transceiver channels and clock multiplier unit (CMU) phase-locked loops (PLLs) independently. The ALTGX Transceiver MegaWizard™ Plug-In Manager provides individual reset signals for each channel instantiated in your design. It also provides one power-down signal for each transceiver block.

This chapter includes the following sections:

- "User Reset and Power-Down Signals" on page 4–1
- "Transceiver Reset Sequences" on page 4–4
- "PMA Direct Drive Mode Reset Sequences" on page 4–23
- "Dynamic Reconfiguration Reset Sequences" on page 4–34
- "Power Down" on page 4–37
- "Simulation Requirements" on page 4–38
- "Reference Information" on page 4–39

Figure 4–1 shows the reset control and power-down block for a Stratix IV GX device.

Figure 4–1. Reset Control and Power-Down Block



User Reset and Power-Down Signals

Each transceiver channel in the Stratix IV GX device has individual reset signals to reset its physical coding sublayer (PCS) and physical medium attachment (PMA) blocks. Each CMU PLL in the transceiver block has a dedicated reset signal. The transceiver block also has a power-down signal that affects all the channels and CMU PLLs in the transceiver block.

All reset and power-down signals are asynchronous.

Table 4–1 lists the reset signals available for each transceiver channel.

Table 4–1. Transceiver Channel Reset Signals

Signal	ALTGX MegaWizard Plug-In Manager Configurations	Description
`tx_digitalreset` *(1)*	■ Transmitter Only ■ Receiver and Transmitter	Provides asynchronous reset to all digital logic in the transmitter PCS, including the XAUI transmit state machine. The minimum pulse width for this signal is two parallel clock cycles.
`rx_digitalreset` *(1)*	■ Receiver Only ■ Receiver and Transmitter	Resets all digital logic in the receiver PCS, including: ■ XAUI receiver state machines ■ GIGE receiver state machines ■ XAUI channel alignment state machine ■ BIST-PRBS verifier ■ BIST-incremental verifier The minimum pulse width for this signal is two parallel clock cycles.
`rx_analogreset`	■ Receiver Only ■ Receiver and Transmitter	Resets the receiver CDR present in the receiver channel. The minimum pulse width is two parallel clock cycles.

Note to Table 4–1:

(1) Assert this signal until the clocks coming out of the transmitter PLL and receiver CDR are stabilized. Stable parallel clocks are essential for proper operation of transmitter and receiver phase-compensation FIFOs in the PCS.

Table 4–2 lists the power-down signals available for each CMU PLL transceiver block.

Table 4–2. Transceiver Block Power-Down Signals (Part 1 of 2)

Signal	Description
`pll_powerdown` *(1)*	Each transceiver block has two CMU PLLs. Each CMU PLL has this dedicated power-down signal. This signal powers down the CMU PLLs that provide high-speed serial and low-speed parallel clocks to the transceiver channels.
`gxb_powerdown` *(1)*	Powers down the entire transceiver block. When this signal is asserted, this signal powers down: ■ the PCS and PMA in all the transceiver channels ■ the CMU PLLs This signal operates independently from the other reset signals. This signal is common to the transceiver block.
`pll_locked`	A status signal. Indicates the status of the transmitter PLL. ■ A high level—the transmitter PLL is locked to the incoming reference clock frequency.
`rx_pll_locked`	A status signal. ■ A high level—the receiver CDR is locked to the incoming reference clock frequency.

Table 4–2. Transceiver Block Power-Down Signals (Part 2 of 2)

Signal	Description
`rx_freqlocked`	A status signal. Indicates the status of the receiver CDR lock mode. ■ A high level—the receiver is in lock-to-data mode. ■ A low level—the receiver CDR is in lock-to-reference mode.
`busy`	A status signal. An output from the ALTGX_RECONFIG block indicates the status of the dynamic reconfiguration controller. This signal remains low for the first `reconfig_clk` clock cycle after power up. It then gets asserted from the second `reconfig_clk` clock cycle. Assertion on this signal indicates that the offset cancellation process is being executed on the receiver buffer as well as the receiver CDR. When this signal is de-asserted, it indicates that offset cancellation is complete.

Note to Table 4–2:

(1) The refclk (`refclk0` or `refclk1`) buffer is not powered down by this signal.

For more information about offset cancellation, refer to the *Stratix IV Dynamic Reconfiguration* chapter.

If none of the channels is instantiated in a transceiver block, the Quartus® II software automatically powers down the entire transceiver block.

Blocks Affected by the Reset and Power-Down Signals

Table 4–3 lists the blocks that are affected by specific reset and power-down signals.

Table 4–3. Blocks Affected by Reset and Power-Down Signals (Part 1 of 2)

Transceiver Block	rx_digitalreset	rx_analogreset	tx_digitalreset	pll_powerdown	gxb_powerdown
CMU PLLs	—	—	—	✓	✓
Transmitter Phase Compensation FIFO	—	—	✓	—	✓
Byte Serializer	—	—	✓	—	✓
8B/10B Encoder	—	—	✓	—	✓
Serializer	—	—	✓	—	✓
Transmitter Buffer	—	—	—	—	✓
Transmitter XAUI State Machine	—	—	✓	—	✓
Receiver Buffer	—	—	—	—	✓
Receiver CDR	—	✓	—	—	✓
Receiver Deserializer	—	—	—	—	✓
Receiver Word Aligner	✓	—	—	—	✓
Receiver Deskew FIFO	✓	—	—	—	✓
Receiver Clock Rate Compensation FIFO	✓	—	—	—	✓
Receiver 8B/10B Decoder	✓	—	—	—	✓
Receiver Byte Deserializer	✓	—	—	—	✓

Table 4–3. Blocks Affected by Reset and Power-Down Signals (Part 2 of 2)

Transceiver Block	rx_digitalreset	rx_analogreset	tx_digitalreset	pll_powerdown	gxb_powerdown
Receiver Byte Ordering	✓	—	—	—	✓
Receiver Phase Compensation FIFO	✓	—	—	—	✓
Receiver XAUI State Machine	✓	—	—	—	✓
BIST Verifiers	✓	—	—	—	✓

Transceiver Reset Sequences

You can configure transceiver channels in Stratix IV GX devices in various configurations. In all functional modes except XAUI functional mode, transceiver channels can be either bonded or non-bonded. In XAUI functional mode, transceiver channels must be bonded. In PCI Express (PIPE) functional mode, transceiver channels can be either bonded or non-bonded and need to follow a specific reset sequence.

The two categories of reset sequences for Stratix IV GX devices described in this chapter are:

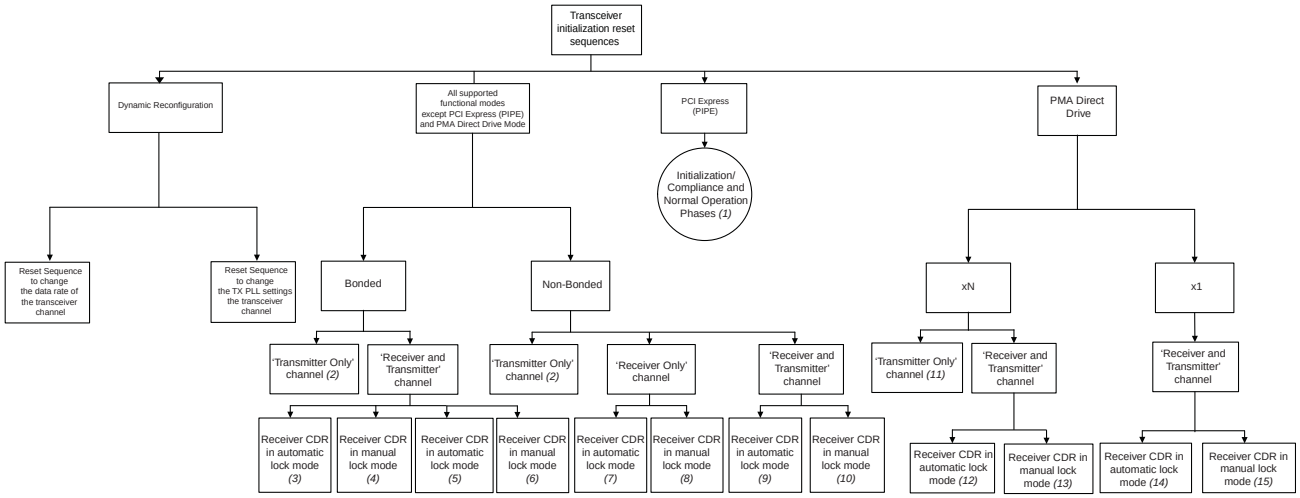
- "All Supported Functional Modes Except the PCI Express (PIPE) Functional Mode" on page 4–5—describes the reset sequences in bonded and non-bonded configurations.
- "PCI Express (PIPE) Functional Mode" on page 4–20—describes the reset sequence for the initialization/compliance phase and the normal operation phase in PCI Express (PIPE) functional modes.

The `busy` signal remains low for the first `reconfig_clk` clock cycle. It then gets asserted from the second `reconfig_clk` clock cycle. Subsequent de-assertion of the `busy` signal indicates the completion of the offset cancellation process. This `busy` signal is required in transceiver reset sequences except for transmitter only channel configurations. Refer to the reset sequences shown in Figure 4–2 and the associated references listed in the notes for the figure.

Altera strongly recommends adhering to these reset sequences for proper operation of the Stratix IV GX transceiver.

Figure 4–2 shows the transceiver reset sequences for Stratix IV GX devices.

Figure 4–2. Transceiver Reset Sequences Chart



Notes to Figure 4–2:

(1) Refer to the Timing Diagram in Figure 4–12.
(2) Refer to the Timing Diagram in Figure 4–3.
(3) Refer to the Timing Diagram in Figure 4–4.
(4) Refer to the Timing Diagram in Figure 4–5.
(5) Refer to the Timing Diagram in Figure 4–6.
(6) Refer to the Timing Diagram in Figure 4–7.
(7) Refer to the Timing Diagram in Figure 4–8.
(8) Refer to the Timing Diagram in Figure 4–9.
(9) Refer to the Timing Diagram in Figure 4–10.
(10) Refer to the Timing Diagram in Figure 4–11.
(11) Refer to the Timing Diagram in Figure 4–13.
(12) Refer to the Timing Diagram in Figure 4–16.
(13) Refer to the Timing Diagram in Figure 4–17.
(14) Refer to the Timing Diagram in Figure 4–18.
(15) Refer to the Timing Diagram in Figure 4–19.

All Supported Functional Modes Except the PCI Express (PIPE) Functional Mode

This section describes reset sequences for transceiver channels in bonded and non-bonded configurations. Timing diagrams of some typical configurations are shown to facilitate proper reset sequence implementation. In these functional modes, you can set the receiver CDR either in automatic lock or manual lock mode.

In manual lock mode, the receiver CDR locks to the reference clock (lock-to-reference) or the incoming serial data (lock-to-data), depending on the logic levels on the `rx_locktorefclk` and `rx_locktodata` signals. With the receiver CDR in manual lock mode, you can either configure the transceiver channels in the Stratix IV GX device in a non-bonded configuration or a bonded configuration. In a bonded configuration, for example in XAUI mode, four channels are bonded together.

Table 4–4 lists the lock-to-reference (LTR) and lock-to-data (LTD) controller lock modes for the `rx_locktorefclk` and `rx_locktodata` signals.

Table 4–4. Lock-To-Reference and Lock-To-Data Modes

rx_locktorefclk	rx_locktodata	LTR/LTD Controller Lock Mode
1	0	Manual, LTR Mode
—	1	Manual, LTD Mode
0	0	Automatic Lock Mode

Bonded Channel Configuration

In a bonded channel configuration, you can reset all the bonded channels simultaneously. Examples of bonded channel configurations are the XAUI, PCI Express (PIPE), and Basic ×4 functional modes. In Basic ×4 functional mode, you can bond **Transmitter Only** channels together.

In XAUI mode, the receiver and transmitter channels are bonded. Each of the receiver channels in this mode has its own output status signals, `rx_pll_locked` and `rx_freqlocked`. You must consider the timing of these signals in the reset sequence.

Table 4–5 lists the reset and power-down sequences for bonded configurations under the stated functional modes.

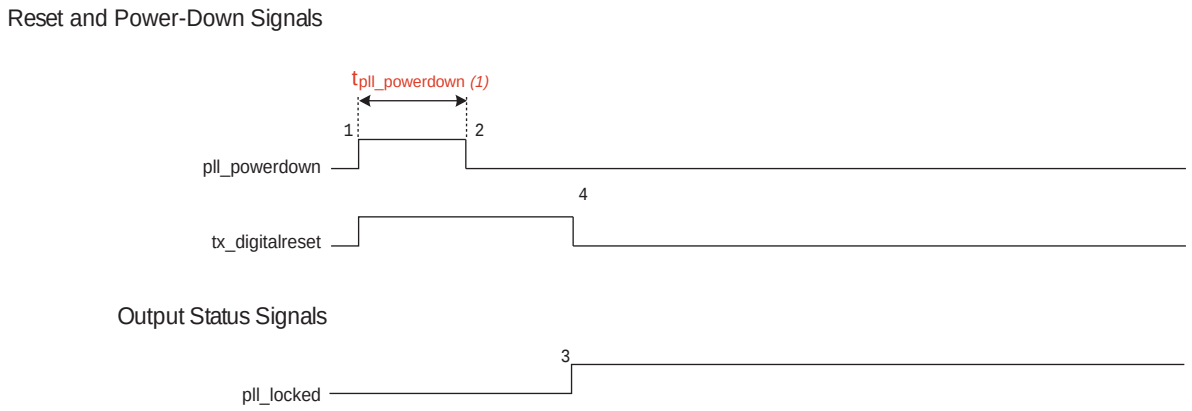
Table 4–5. Reset and Power-Down Sequences for Bonded Channel Configurations

Channel Set Up	Receiver CDR Mode	Refer to
Transmitter Only	Basic ×4	"Transmitter Only Channel" on page 4–7
Receiver and Transmitter	Automatic lock mode for XAUI functional mode	"Receiver and Transmitter Channel—Receiver CDR in Automatic Lock Mode" on page 4–8
Receiver and Transmitter	Manual lock mode for XAUI functional mode	"Receiver and Transmitter Channel—Receiver CDR in Manual Lock Mode" on page 4–10
Receiver and Transmitter	Automatic lock mode for Basic ×8 functional mode	"Receiver and Transmitter Channel—Receiver CDR in Automatic Lock Mode" on page 4–12
Receiver and Transmitter	Manual lock mode for Basic ×8 functional mode	"Receiver and Transmitter Channel—Receiver CDR in Manual Lock Mode" on page 4–13

Transmitter Only Channel

This configuration contains only a transmitter channel. If you create a **Transmitter Only** instance in the ALTGX MegaWizard Plug-In Manager in Basic ×4 functional mode, use the reset sequence shown in Figure 4–3.

Figure 4–3. Sample Reset Sequence for Four Transmitter Only Channels



Note to Figure 4–3:

(1) For $t_{pll_powerdown}$ duration, refer to the *DC and Switching Characteristics* chapter in the "Stratix IV Device Datasheet" section.

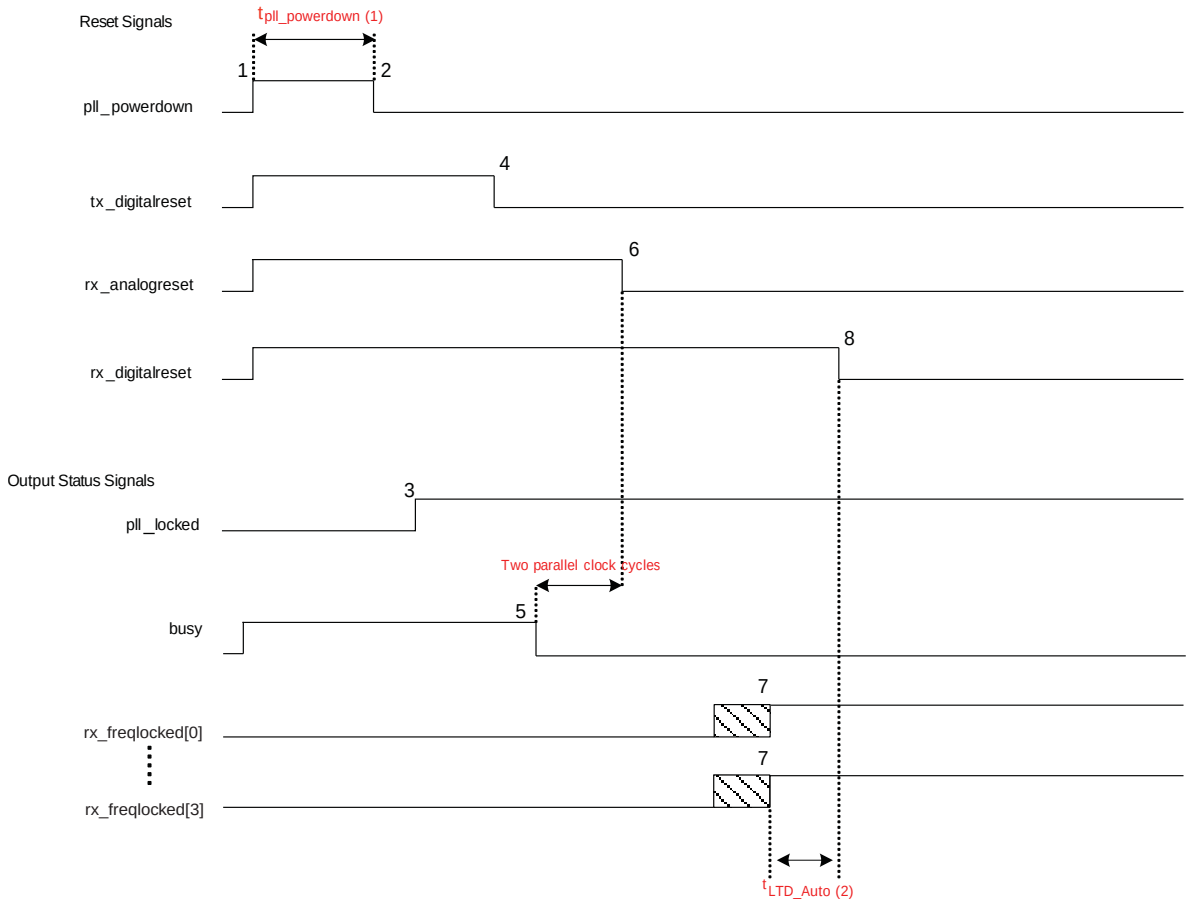
As shown in Figure 4–3, perform the following reset procedure for the **Transmitter Only** channel configuration:

1. After power up, assert `pll_powerdown` for a minimum period of $t_{pll_powerdown}$ (the time between markers 1 and 2).
2. Keep the `tx_digitalreset` signal asserted during this time period. After you de-assert the `pll_powerdown` signal, the transmitter PLL starts locking to the transmitter input reference clock.
3. When the transmitter PLL locks, as indicated by the `pll_locked` signal going high (marker 3), de-assert the `tx_digitalreset` signal (marker 4). At this point, the transmitter is ready for transmitting data.

Receiver and Transmitter Channel—Receiver CDR in Automatic Lock Mode

This configuration contains both a transmitter and receiver channel. For XAUI functional mode, with the receiver CDR in automatic lock mode, use the reset sequence shown in Figure 4–4.

Figure 4–4. Sample Reset Sequence for Four Receiver and Transmitter Channels—Receiver CDR in Automatic Lock Mode



Notes to Figure 4–4:

(1) For $t_{pll_powerdown}$ duration, refer to the *DC and Switching Characteristics* chapter in the "Stratix IV Device Datasheet" section.

(2) For t_{LTD_Auto} duration, refer to the *DC and Switching Characteristics* chapter in the "Stratix IV Device Datasheet" section.

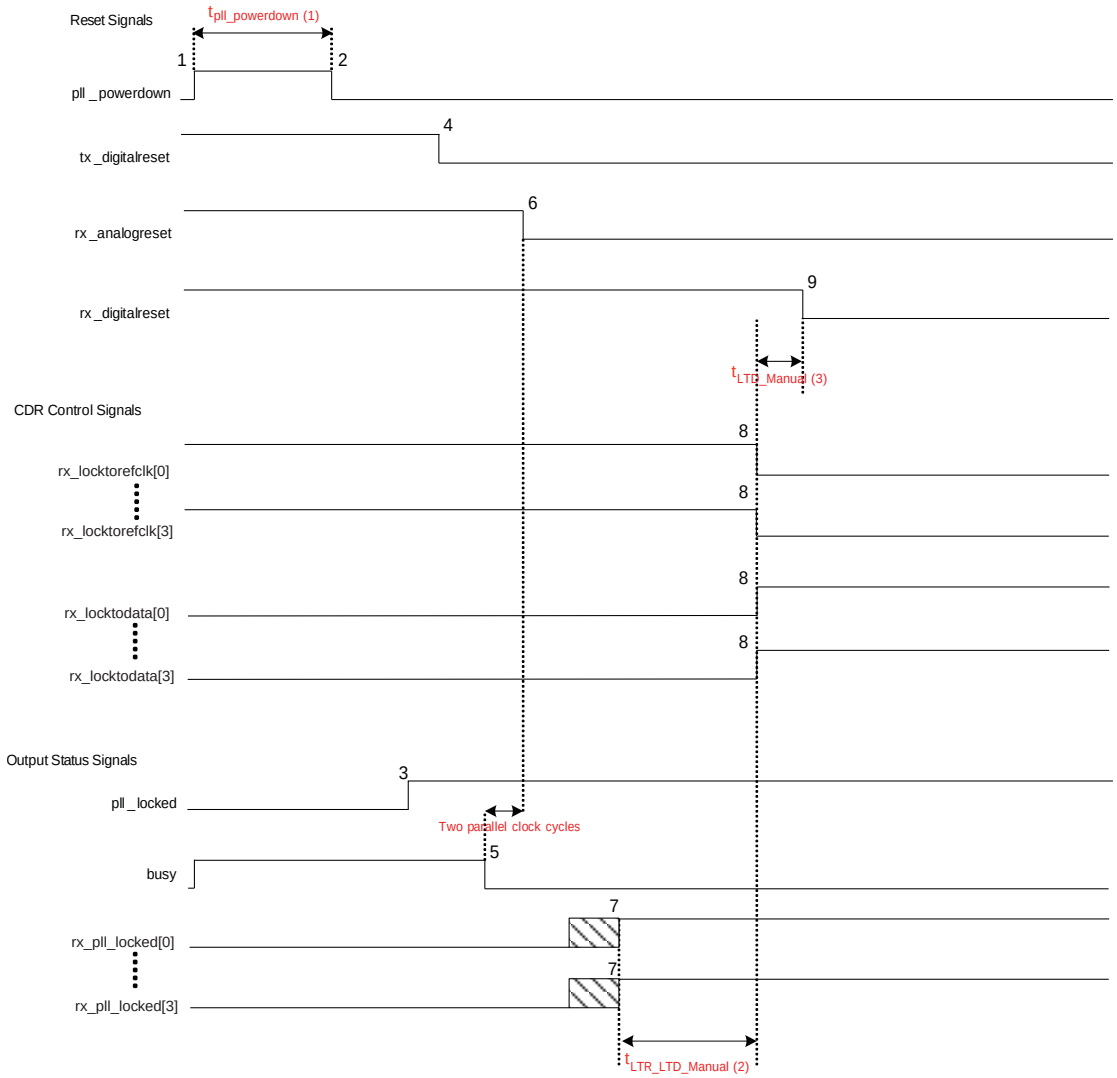
As shown in Figure 4–4, perform the following reset procedure for the receiver CDR in automatic lock mode configuration:

1. After power up, assert `pll_powerdown` for a minimum period of $t_{pll_powerdown}$ (the time between markers 1 and 2).
2. Keep the `tx_digitalreset`, `rx_analogreset`, and `rx_digitalreset` signals asserted during this time period. After you de-assert the `pll_powerdown` signal, the transmitter PLL starts locking to the transmitter input reference clock.
3. After the transmitter PLL locks, as indicated by the `pll_locked` signal going high, de-assert the `tx_digitalreset` signal. At this point, the transmitter is ready for data traffic.
4. For the receiver operation, after de-assertion of `busy` signal, wait for two parallel clock cycles to de-assert the `rx_analogreset` signal. After `rx_analogreset` is de-asserted, the receiver CDR of each channel starts locking to the receiver input reference clock.
5. Wait for the `rx_freqlocked` signal from each channel to go high. The `rx_freqlocked` signal of each channel may go high at different times (indicated by the slashed pattern at marker 7).
6. In a bonded channel group, when the `rx_freqlocked` signals of all the channels has gone high, from that point onwards, wait for at least t_{LTD_Auto} time for the receiver parallel clock to be stable, then de-assert the `rx_digitalreset` signal (marker 8). At this point, all the receivers are ready for data traffic.

Receiver and Transmitter Channel—Receiver CDR in Manual Lock Mode

This configuration contains both a transmitter and receiver channel. For XAUI functional mode, with the receiver CDR in manual lock mode, use the reset sequence shown in Figure 4–5.

Figure 4–5. Sample Reset Sequence for Four Receiver and Transmitter Channels—Receiver CDR in Manual Lock Mode



Notes to Figure 4–5:

(1) For $t_{pll_powerdown}$ duration, refer to the *DC and Switching Characteristics* chapter in the "Stratix IV Device Datasheet" section.

(2) For $t_{LTR_LTD_Manual}$ duration, refer to the *DC and Switching Characteristics* chapter in the "Stratix IV Device Datasheet" section.

(3) For t_{LTD_Manual} duration, refer to the *DC and Switching Characteristics* chapter in the "Stratix IV Device Datasheet" section.

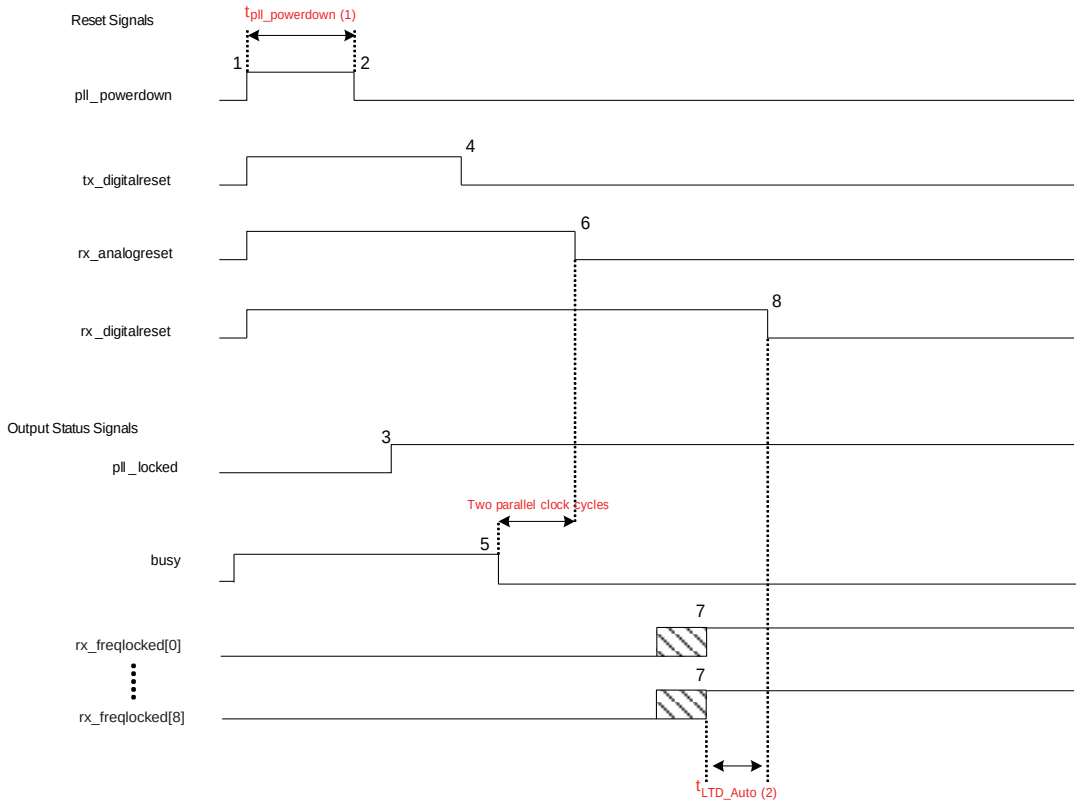
As shown in Figure 4–5, perform the following reset procedure for the receiver CDR in manual lock mode configuration:

1. After power up, assert `pll_powerdown` for a minimum period of $t_{pll_powerdown}$ (the time between markers 1 and 2).
2. Keep the `tx_digitalreset`, `rx_analogreset`, `rx_digitalreset`, and `rx_locktorefclk` signals asserted and the `rx_locktodata` signal de-asserted during this time period. After you de-assert the `pll_powerdown` signal, the transmitter PLL starts locking to the transmitter input reference clock.
3. After the transmitter PLL locks, as indicated by the `pll_locked` signal going high (marker 3), de-assert the `tx_digitalreset` signal (marker 4). For the receiver operation, after de-assertion of the `busy` signal, wait for two parallel clock cycles to de-assert the `rx_analogreset` signal. After the `rx_analogreset` signal is de-asserted, the receiver CDR of each channel starts locking to the receiver input reference clock because `rx_locktorefclk` is asserted.
4. Wait for the `rx_pll_locked` signal from each channel to go high. The `rx_pll_locked` signal of each channel may go high at different times with respect to each other (indicated by the slashed pattern at marker 7).
5. In a bonded channel group, when the `rx_pll_locked` signal of all the channels have gone high, from that point onwards, wait for at least $t_{LTR_LTD_Manual}$ time, then de-assert `rx_locktorefclk` and assert `rx_locktodata` (marker 8). At this point, the receiver CDR of all the channels enters into lock-to-data mode and starts locking to the received data.
6. After asserting the `rx_locktodata` signal, wait for at least t_{LTD_Manual} time before de-asserting `rx_digitalreset` (the time between markers 8 and 9).

Receiver and Transmitter Channel—Receiver CDR in Automatic Lock Mode

This configuration contains both a transmitter and a receiver channel. For Basic ×8 functional mode, with the receiver CDR in automatic lock mode, use the reset sequence shown in Figure 4–6.

Figure 4–6. Sample Reset Sequence for Eight Receiver and Transmitter Channels—Receiver CDR in Automatic Lock Mode



Notes to Figure 4–6:

(1) For $t_{pll_powerdown}$ duration, refer to the *DC and Switching Characteristics* chapter in the "Stratix IV Device Datasheet" section.

(2) For t_{LTD_Auto} duration, refer to the *DC and Switching Characteristics* chapter in the "Stratix IV Device Datasheet" section.

As shown in Figure 4–6, perform the following reset procedure for the receiver CDR in automatic lock mode:

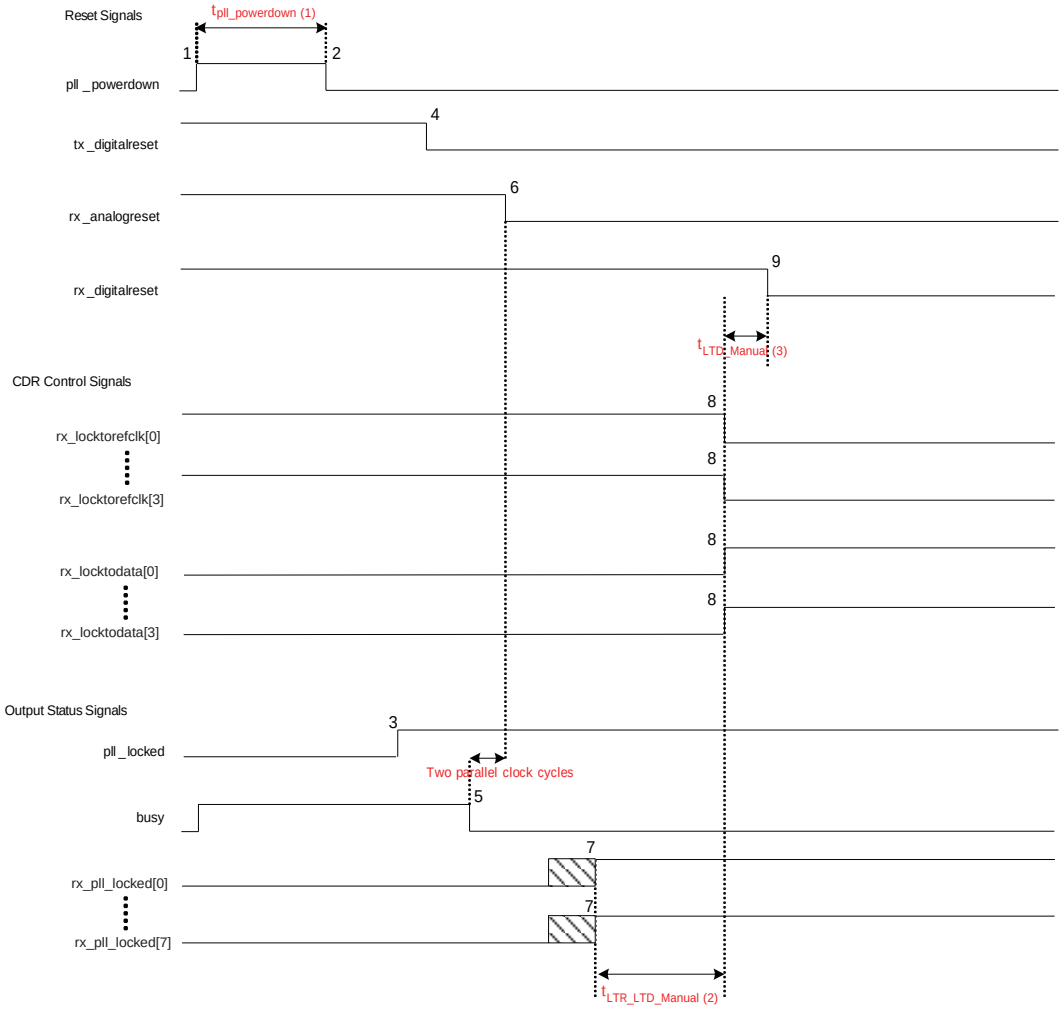
1. After power up, assert `pll_powerdown` for a minimum period of $t_{pll_powerdown}$ (the time between markers 1 and 2).
2. Keep the `tx_digitalreset`, `rx_analogreset`, and `rx_digitalreset` signals asserted during this time period. After you de-assert the `pll_powerdown` signal, the transmitter PLL starts locking to the transmitter input reference clock.
3. After the transmitter PLL locks, as indicated by the `pll_locked` signal going high, de-assert the `tx_digitalreset` signal. At this point, the transmitter is ready for data traffic.
4. For the receiver operation, after de-assertion of the `busy` signal, wait for two parallel clock cycles to de-assert the `rx_analogreset` signal. After `rx_analogreset` is de-asserted, the receiver CDR of each channel starts locking to the receiver input reference clock.

5. Wait for the `rx_freqlocked` signal from each channel to go high. The `rx_freqlocked` signal of each channel may go high at different times (indicated by the slashed pattern at marker 7).
6. In a bonded channel group, when the `rx_freqlocked` signals of all the channels have gone high, from that point onwards, wait for at least t_{LTD_Auto} time for the receiver parallel clock to stabilize, then de-assert the `rx_digitalreset` signal (marker 8). At this point, all the receivers are ready for data traffic.

Receiver and Transmitter Channel—Receiver CDR in Manual Lock Mode

This configuration contains both a transmitter and receiver channel. For Basic ×8 functional mode, with the receiver CDR in manual lock mode, use the reset sequence shown in Figure 4–7.

Figure 4–7. Sample Reset Sequence for Eight Receiver and Transmitter Channels—Receiver CDR in Manual Lock Mode



Notes to Figure 4–7:

(1) For $t_{pll_powerdown}$ duration, refer to the *DC and Switching Characteristics* chapter in the "Stratix IV Device Datasheet" section.

(2) For $t_{LTR_LTD_Manual}$ duration, refer to the *DC and Switching Characteristics* chapter in the "Stratix IV Device Datasheet" section.

(3) For t_{LTD_Manual} duration, refer to the *DC and Switching Characteristics* chapter in the "Stratix IV Device Datasheet" section.

As shown in Figure 4–7, perform the following reset procedure for the receiver CDR in manual lock mode:

1. After power up, assert `pll_powerdown` for a minimum period of $t_{pll_powerdown}$ (the time between markers 1 and 2).
2. Keep the `tx_digitalreset`, `rx_analogreset`, `rx_digitalreset`, and `rx_locktorefclk` signals asserted and the `rx_locktodata` signal de-asserted during this time period. After you de-assert the `pll_powerdown` signal, the transmitter PLL starts locking to the transmitter input reference clock.
3. After the transmitter PLL locks, as indicated by the `pll_locked` signal going high (marker 3), de-assert the `tx_digitalreset` signal (marker 4). For the receiver operation, after de-assertion of the `busy` signal, wait for two parallel clock cycles to de-assert the `rx_analogreset` signal. After the `rx_analogreset` signal is de-asserted, the receiver CDR of each channel starts locking to the receiver input reference clock because `rx_locktorefclk` is asserted.
4. Wait for the `rx_pll_locked` signal from each channel to go high. The `rx_pll_locked` signal of each channel may go high at different times with respect to each other (indicated by the slashed pattern at marker 7).
5. In a bonded channel group, when the `rx_pll_locked` signal of all the channels has gone high, from that point onwards, wait for at least $t_{LTR_LTD_Manual}$ time, then de-assert `rx_locktorefclk` and assert `rx_locktodata` (marker 8). At this point, the receiver CDR of all the channels enters into lock-to-data mode and starts locking to the received data.
6. De-assert `rx_digitalreset` at least t_{LTD_Manual} time (the time between markers 8 and 9) after asserting the `rx_locktodata` signal.

Non-Bonded Channel Configuration

In non-bonded channels, each channel in the ALTGX MegaWizard Plug-In Manager instance contains its own `tx_digitalreset`, `rx_analogreset`, `rx_digitalreset`, `rx_pll_locked`, and `rx_freqlocked` signals.

You can reset each channel independently. For example, if there are four non-bonded channels, the ALTGX MegaWizard Plug-In Manager provides four each of the following signals: `tx_digitalreset`, `rx_analogreset`, `rx_digitalreset`, `rx_pll_locked`, and `rx_freqlocked`.

Table 4–6 lists the reset and power-down sequences for one channel in a non-bonded configuration under the stated functional modes.

Table 4–6. Reset and Power-Down Sequences for Bonded Channel Configurations (Part 1 of 2)

Channel Set Up	Receiver CDR Mode	Refer to
Transmitter Only	Basic ×4	"Transmitter Only Channel" on page 4–15
Receiver Only	Automatic lock mode	"Receiver Only Channel—Receiver CDR in Automatic Lock Mode" on page 4–15
Receiver Only	Manual lock mode	"Receiver Only Channel—Receiver CDR in Manual Lock Mode" on page 4–16

Table 4–6. Reset and Power-Down Sequences for Bonded Channel Configurations (Part 2 of 2)

Channel Set Up	Receiver CDR Mode	Refer to
Receiver and Transmitter	Automatic lock mode	"Receiver and Transmitter Channel—Receiver CDR in Automatic Lock Mode" on page 4–17
Receiver and Transmitter	Manual lock mode	"Receiver and Transmitter Channel—Receiver CDR in Manual Lock Mode" on page 4–19

Follow the same reset sequence for all the other channels in the non-bonded configuration.

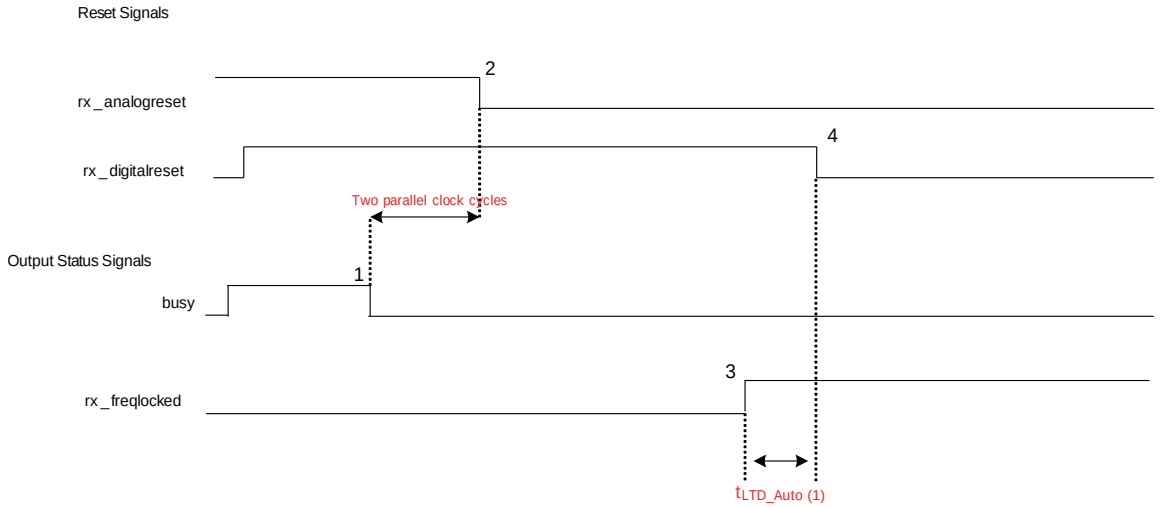
Transmitter Only Channel

This configuration contains only a transmitter channel. If you create a **Transmitter Only** instance in the ALTGX MegaWizard Plug-In Manager, use the same reset sequence shown in Figure 4–3 on page 4–7.

Receiver Only Channel—Receiver CDR in Automatic Lock Mode

This configuration contains only a receiver channel. If you create a **Receiver Only** instance in the ALTGX MegaWizard Plug-In Manager with the receiver CDR in automatic lock mode, use the reset sequence shown in Figure 4–8.

Figure 4–8. Sample Reset Sequence of Receiver Only Channel—Receiver CDR in Automatic Lock Mode



Note to Figure 4–8:

(1) For t_{LTD_Auto} duration, refer to the *DC and Switching Characteristics* chapter in the "Stratix IV Device Datasheet" section.

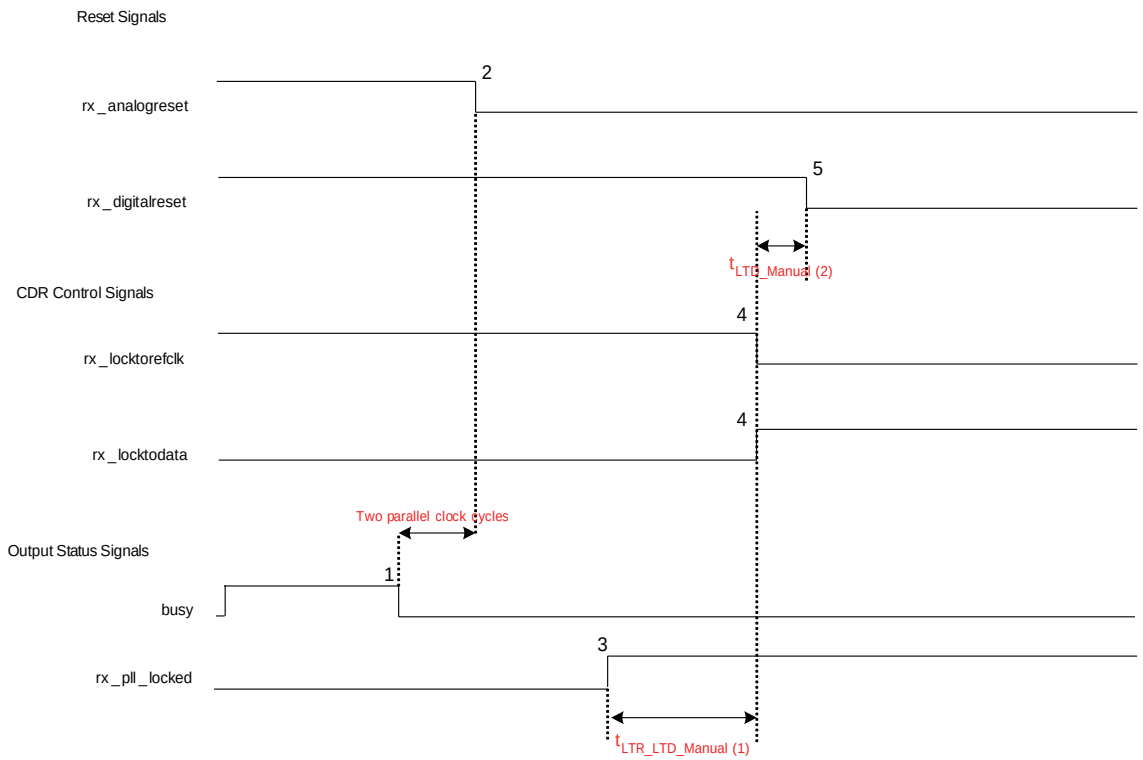
As shown in Figure 4–8, perform the following reset procedure for the receiver in CDR automatic lock mode:

1. After power up, wait for the `busy` signal to be de-asserted.
2. De-assert the `rx_analogreset` signal.
3. Keep the `rx_digitalreset` signal asserted during this time period. After you de-assert the `rx_analogreset` signal, the receiver CDR starts locking to the receiver input reference clock.
4. Wait for the `rx_freqlocked` signal to go high.
5. When `rx_freqlocked` goes high (marker 3), from that point onwards, wait for at least t_{LTD_Auto}, then de-assert the `rx_digitalreset` signal (marker 4). At this point, the receiver is ready to receive data.

Receiver Only Channel—Receiver CDR in Manual Lock Mode

This configuration contains only a receiver channel. If you create a **Receiver Only** instance in the ALTGX MegaWizard Plug-In Manager with receiver CDR in manual lock mode, use the reset sequence shown in Figure 4–9.

Figure 4–9. Sample Reset Sequence of Receiver Only Channel—Receiver CDR in Manual Lock Mode



Notes to Figure 4–9:

(1) For t_{LTR_LTD} duration, refer to the *DC and Switching Characteristics* chapter in the "Stratix IV Device Datasheet" section.

(2) For t_{LTD_Manual} duration, refer to the *DC and Switching Characteristics* chapter in the "Stratix IV Device Datasheet" section.

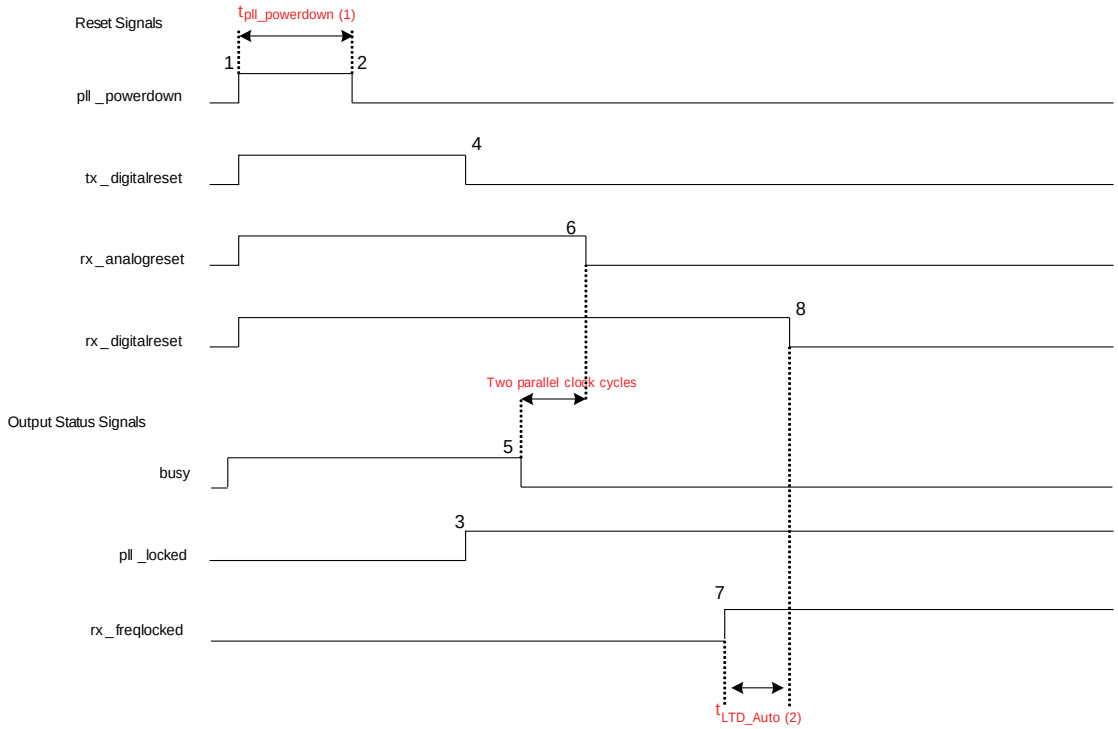
As shown in Figure 4–9, perform the following reset procedure for the receiver CDR in manual lock mode:

1. After power up, wait for the `busy` signal to be asserted.
2. Keep the `rx_digitalreset` and `rx_locktorefclk` signals asserted and the `rx_locktodata` signal de-asserted during this time period.
3. After de-assertion of the `busy` signal, de-assert the `rx_analogreset` signal. The receiver CDR then starts locking to the receiver input reference clock because the `rx_locktorefclk` signal is asserted.
4. Wait for at least $t_{LTR_LTD_Manual}$ time (the time between markers 3 and 4) after the `rx_pll_locked` signal goes high and then de-assert the `rx_locktorefclk` signal. At the same time, assert the `rx_locktodata` signal (marker 4). At this point, the receiver CDR enters lock-to-data mode and the receiver PLL starts locking to the received data.
5. De-assert `rx_digitalreset` at least t_{LTD_Manual} (the time between markers 4 and 5) after asserting the `rx_locktodata` signal.

Receiver and Transmitter Channel—Receiver CDR in Automatic Lock Mode

This configuration contains both a transmitter and a receiver channel. If you create a **Receiver and Transmitter** instance in the ALTGX MegaWizard Plug-In Manager with the receiver CDR in automatic lock mode, use the reset sequence shown in Figure 4–10.

Figure 4–10. Sample Reset Sequence of Receiver and Transmitter Channel—Receiver CDR in Automatic Lock Mode



Notes to Figure 4–10:

(1) For $t_{pll_powerdown}$ duration, refer to the *DC and Switching Characteristics* chapter in the "Stratix IV Device Datasheet" section.

(2) For t_{LTD_Auto} duration, refer to the *DC and Switching Characteristics* chapter in the "Stratix IV Device Datasheet" section.

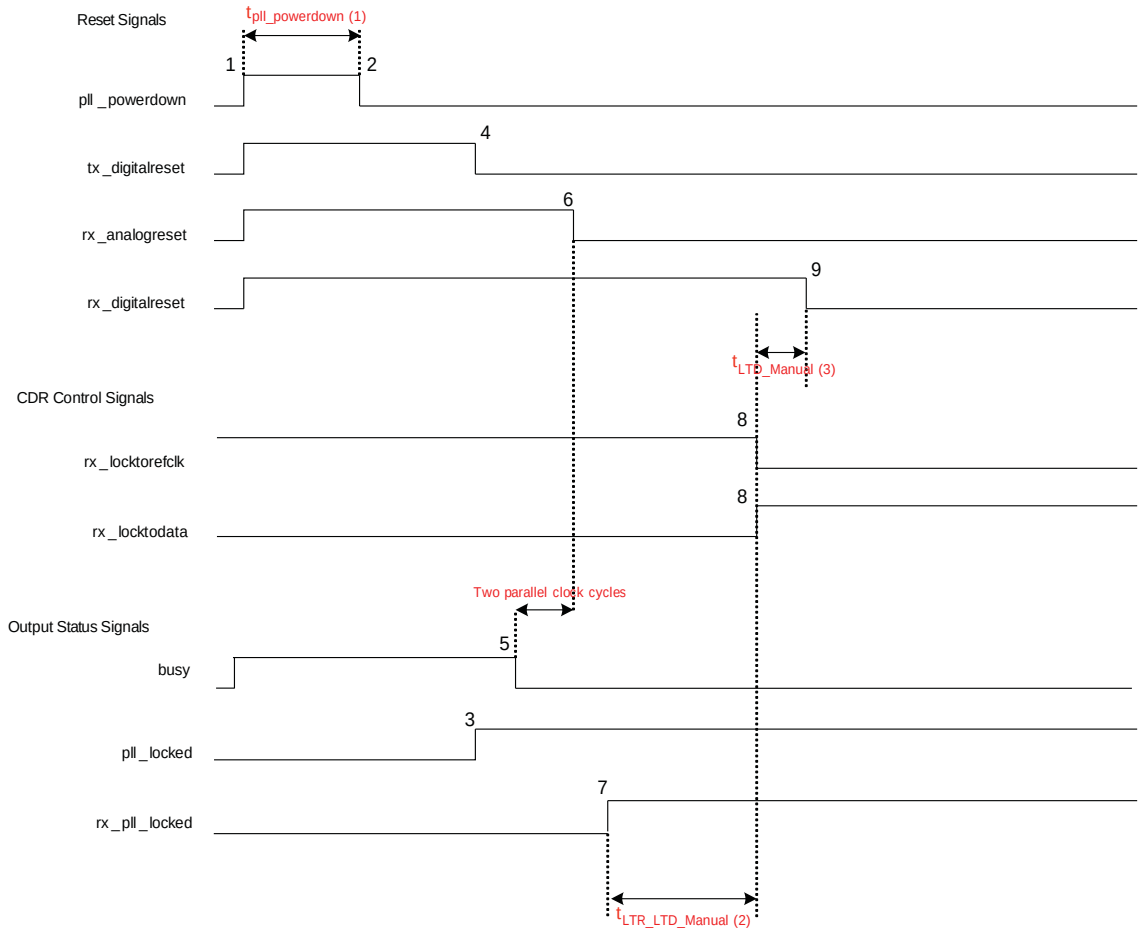
As shown in Figure 4–10, perform the following reset procedure for the receiver in CDR automatic lock mode:

1. After power up, assert `pll_powerdown` for a minimum period of $t_{pll_powerdown}$ (the time between markers 1 and 2).
2. Keep the `tx_digitalreset`, `rx_analogreset`, and `rx_digitalreset` signals asserted during this time period. After you de-assert the `pll_powerdown` signal, the transmitter PLL starts locking to the transmitter input reference clock.
3. After the transmitter PLL locks, as indicated by the `pll_locked` signal going high (marker 3), de-assert `tx_digitalreset`. For receiver operation, wait for the `busy` signal to be de-asserted, after which `rx_analogreset` is de-asserted. After you de-assert `rx_analogreset`, the receiver CDR starts locking to the receiver input reference clock.
4. Wait for the `rx_freqlocked` signal to go high (marker 7).
5. After the `rx_freqlocked` signal goes high, wait for at least t_{LTD_Auto}, then de-assert the `rx_digitalreset` signal (marker 8). At this point, the transmitter and receiver are ready for data traffic.

Receiver and Transmitter Channel—Receiver CDR in Manual Lock Mode

This configuration contains both a transmitter and receiver channel. If you create a **Receiver and Transmitter** instance in the ALTGX MegaWizard Plug-In Manager with the receiver CDR in manual lock mode, use the reset sequence shown in Figure 4–11.

Figure 4–11. Sample Reset Sequence of Receiver and Transmitter Channel—Receiver CDR in Manual Lock Mode



Notes to Figure 4–11:

(1) For $t_{pll_powerdown}$ duration, refer to the *DC and Switching Characteristics* chapter in the "Stratix IV Device Datasheet" section.

(2) For $t_{LTR_LTD_Manual}$ duration, refer to the *DC and Switching Characteristics* chapter in the "Stratix IV Device Datasheet" section.

(3) For t_{LTD_Manual} duration, refer to the *DC and Switching Characteristics* chapter in the "Stratix IV Device Datasheet" section.

As shown in Figure 4–11, perform the following reset procedure for the receiver in manual lock mode:

1. After power up, assert `pll_powerdown` for a minimum period of $t_{pll_powerdown}$ (the time between markers 1 and 2).
2. Keep the `tx_digitalreset`, `rx_analogreset`, `rx_digitalreset`, and `rx_locktorefclk` signals asserted and the `rx_locktodata` signal de-asserted during this time period. After you de-assert the `pll_powerdown` signal, the transmitter PLL starts locking to the transmitter input reference clock.
3. After the transmitter PLL locks, as indicated by the `pll_locked` signal going high (marker 3), de-assert `tx_digitalreset`. For receiver operation, wait for the `busy` signal to be de-asserted. At this point `rx_analogreset` is de-asserted. When `rx_analogreset` is de-asserted, the receiver CDR starts locking to the receiver input reference clock because `rx_locktorefclk` is asserted.
4. Wait for at least $t_{LTR_LTD_Manual}$ (the time between markers 7 and 8) after the `rx_pll_locked` signal goes high, then de-assert the `rx_locktorefclk` signal. At the same time, assert the `rx_locktodata` signal (marker 8). At this point, the receiver CDR enters lock-to-data mode and the receiver CDR starts locking to the received data.
5. De-assert `rx_digitalreset` at least t_{LTD_Manual} (the time between markers 8 and 9) after asserting the `rx_locktodata` signal.

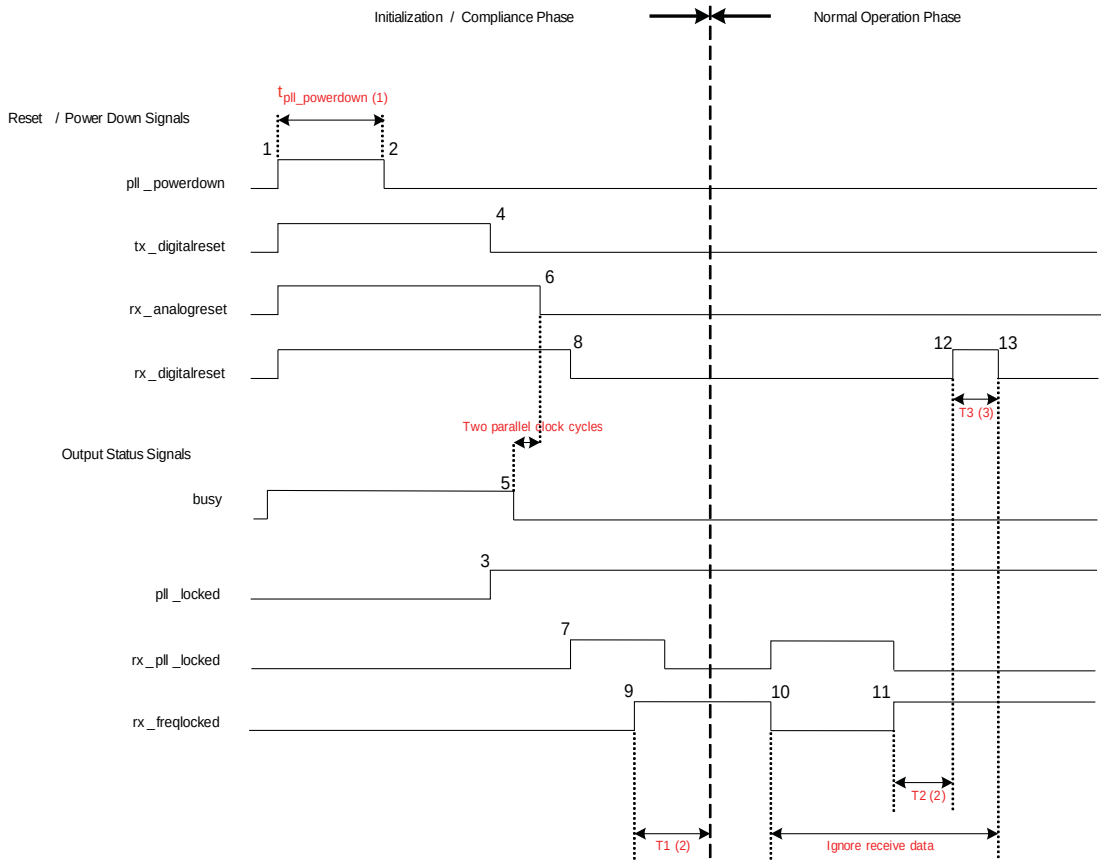
PCI Express (PIPE) Functional Mode

You can configure PCI Express (PIPE) functional mode with or without the receiver clock rate compensation FIFO in the Stratix IV GX device. The reset sequence remains the same whether or not you use the receiver clock rate compensation FIFO.

PCI Express (PIPE) Reset Sequence

PCI Express (PIPE) protocol consists of an initialization/compliance phase and a normal operation phase. The reset sequences for these two phases are described based on the timing diagram in Figure 4–12.

Figure 4–12. Reset Sequence of PCI Express (PIPE) Functional Mode



Notes to Figure 4–12:

(1) For $t_{pll_powerdown}$ duration, refer to the *DC and Switching Characteristics* chapter in the "Stratix IV Device Datasheet" section.
(2) The minimum T1 and T2 period is 4 µ s.
(3) The minimum T3 period is two parallel clock cycles.

PCI Express (PIPE) Initialization/Compliance Phase

After the device is powered up, a PCI Express (PIPE)-compliant device goes through the compliance phase during initialization. In this phase, the PCI Express (PIPE) protocol requires the system to be operating at Gen1 data rate. The `rx_digitalreset` signal must be de-asserted during this compliance phase to achieve transitions on the `pipephydonestatus` signal, as expected by the link layer. The `rx_digitalreset` signal is de-asserted based on the assertion of the `rx_freqlocked` signal.

During the initialization/compliance phase, do not use the `rx_freqlocked` signal to trigger a de-assertion of the `rx_digitalreset` signal. Instead, perform the following reset sequence:

1. After power up, assert `pll_powerdown` for a minimum period of $t_{pll_powerdown}$ (the time between markers 1 and 2). Keep the `tx_digitalreset`, `rx_analogreset`, and `rx_digitalreset` signals asserted during this time period. After you de-assert the `pll_powerdown` signal, the transmitter PLL starts locking to the transmitter input reference clock.
2. When the transmitter PLL locks, as indicated by the `pll_locked` signal going high (marker 3), de-assert `tx_digitalreset`. For a receiver operation, wait for the `busy` signal to be de-asserted. `rx_analogreset` is then de-asserted. After `rx_analogreset` is de-asserted, the receiver CDR starts locking to the receiver input reference clock.
3. When the receiver CDR locks to the input reference clock, as indicated by the `rx_pll_locked` signal going high at marker 7 in Figure 4–12, de-assert the `rx_digitalreset` signal (marker 8). After de-asserting `rx_digitalreset`, the `pipephydonestatus` signal transitions from the transceiver channel to indicate the status to the link layer. Depending on its status, `pipephydonestatus` helps with the continuation of the compliance phase. After successful completion of this phase, the device enters into the normal operation phase.

PCI Express (PIPE) Normal Phase

For the normal PCI Express (PIPE) phase:

1. After completion of the Initialization/Compliance phase, during the normal operation phase at the Gen1 data rate, when the `rx_freqlocked` signal is de-asserted (marker 10 in Figure 4–12), wait for the `rx_pll_locked` signal assertion signifying the lock-to-reference clock.
2. Wait for the `rx_freqlocked` signal to go high again. In this phase, the received data is valid (not electrical idle) and the receiver CDR locks to the incoming data. Proceed with the reset sequence after assertion of the `rx_freqlocked` signal.
3. After the `rx_freqlocked` signal goes high, wait for at least 4 µs before asserting `rx_digitalreset` (marker 12 in Figure 4–12) for two parallel receive clock cycles so that the receiver phase compensation FIFO is initialized.
4. During normal operation, after you speed-negotiate to the Gen2 data rate, asserting the `rx_digitalreset` signal causes the PCI Express (PIPE) rate switch circuitry to switch the transceiver to the Gen1 data rate.

   Data from the transceiver block is not valid from the time the `rx_freqlocked` signal goes low (marker 10 in Figure 4–12) to the time `rx_digitalreset` is de-asserted (marker 13 in Figure 4–12). The PLD logic ignores the data during this period (between markers 10 and 13 in Figure 4–12).

You can configure the Stratix IV GX device in ×1, ×4, and ×8 PCI Express (PIPE) configurations. The reset sequence described in "PCI Express (PIPE) Reset Sequence" on page 4–21 applies to all these multi-lane configurations.

PMA Direct Drive Mode Reset Sequences

Stratix IV GX devices provide a PMA Direct mode in which all PCS blocks, including the phase compensation FIFOs, are bypassed in both the transmitter and receiver channel data paths. In this mode, the PMA block in the transmitter and receiver channels directly interface with the FPGA fabric.

In PMA Direct drive mode, you can configure the transceiver channels as a single channel or in bonded configurations. Basic single- and double-width functional modes support bonding of PMA functional blocks across all transceiver channels on the same side of the device.

☞ The `tx_digitalreset` and `rx_digitalreset` signals are not available because there are no PCS blocks available in this mode.

Table 4–7 lists the reset and power-down sequences for PMA Direct drive ×N functional mode.

Table 4–7. Reset and Power-Down Sequences for PMA Direct Drive ×N Configurations

Channel Set Up	Functional Mode	Refer to
Transmitter Only with no PLL_L/R	Basic (PMA Direct) drive ×4	"Transmitter Only Channel with No PLL_L/R" on page 4–24
Transmitter Only with a PLL_L/R	Manual lock mode	"Transmitter Only Channel with a PLL_L/R" on page 4–25
Receiver and Transmitter	Automatic lock mode for Basic (PMA Direct) drive ×N mode	"Receiver and Transmitter Channel Set-up—Receiver CDR in Automatic Lock Mode" on page 4–27
Receiver and Transmitter	Manual lock mode for Basic (PMA Direct) drive ×N mode	"Receiver and Transmitter Channel Set-up—Receiver CDR in Manual Lock Mode" on page 4–29

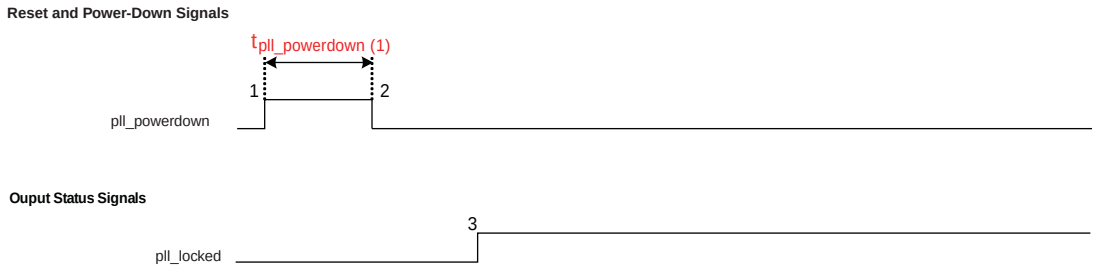
Basic (PMA Direct) Drive ×N Mode

When bonding ×N channels in a Basic (PMA Direct) drive mode configuration, you can reset all bonded channels simultaneously.

Transmitter Only Channel with No PLL_L/R

An example reset sequence timing diagram of four **Transmitter Only** channels in Basic (PMA Direct) drive ×4 functional mode with no PLL_L/R is shown in Figure 4–13.

Figure 4–13. Reset Sequence Timing in Basic (PMA Direct) Drive ×4 Mode



Note to Figure 4–13:

(1) For $t_{pll_powerdown}$ duration, refer to the *DC and Switching Characteristics* chapter in the "Stratix IV Device Datasheet" section.

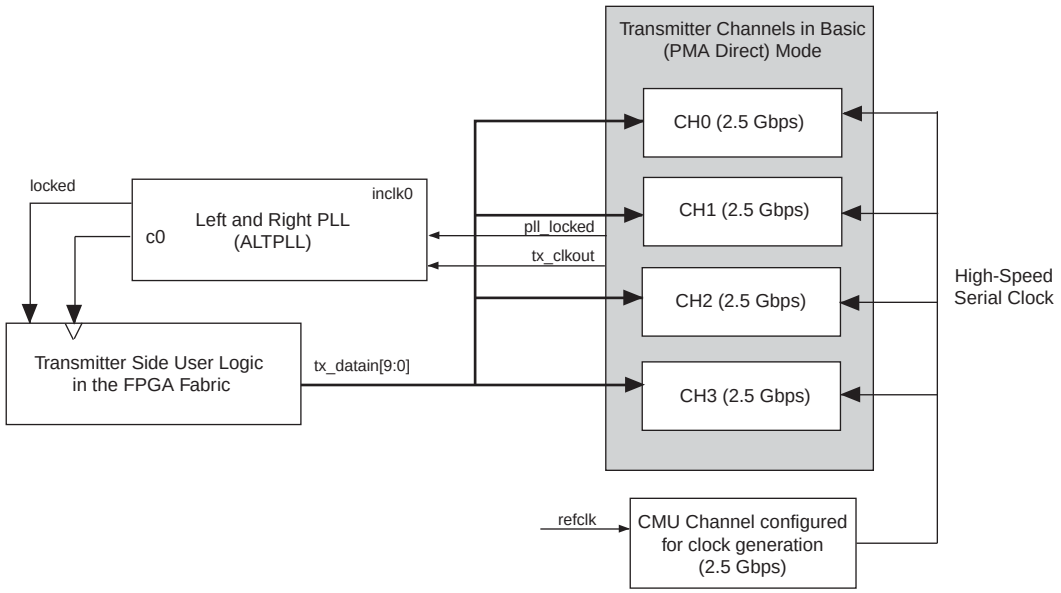
As shown in Figure 4–13, perform the following reset procedure for the **Transmitter Only** channel in Basic (PMA Direct) drive functional ×4 mode with no PLL_L/R:

1. After power up, assert `pll_powerdown` for a minimum of $t_{pll_powerdown}$ (the time between markers 1 and 2).
2. When the transmitter PLL locks, as indicated by the `pll_locked` signal going high (marker 3), the transmitter is ready to accept parallel data from the FPGA fabric and subsequently transmitting serial data reliably.

Transmitter Only Channel with a PLL_L/R

The Basic (PMA Direct) mode configuration that requires a PLL_L/R is one where each channel in PMA-Direct mode is identical. Figure 4–14 shows a simple set up of identical channels.

Figure 4–14. Identical Channels



Identical channels have the following same configuration:

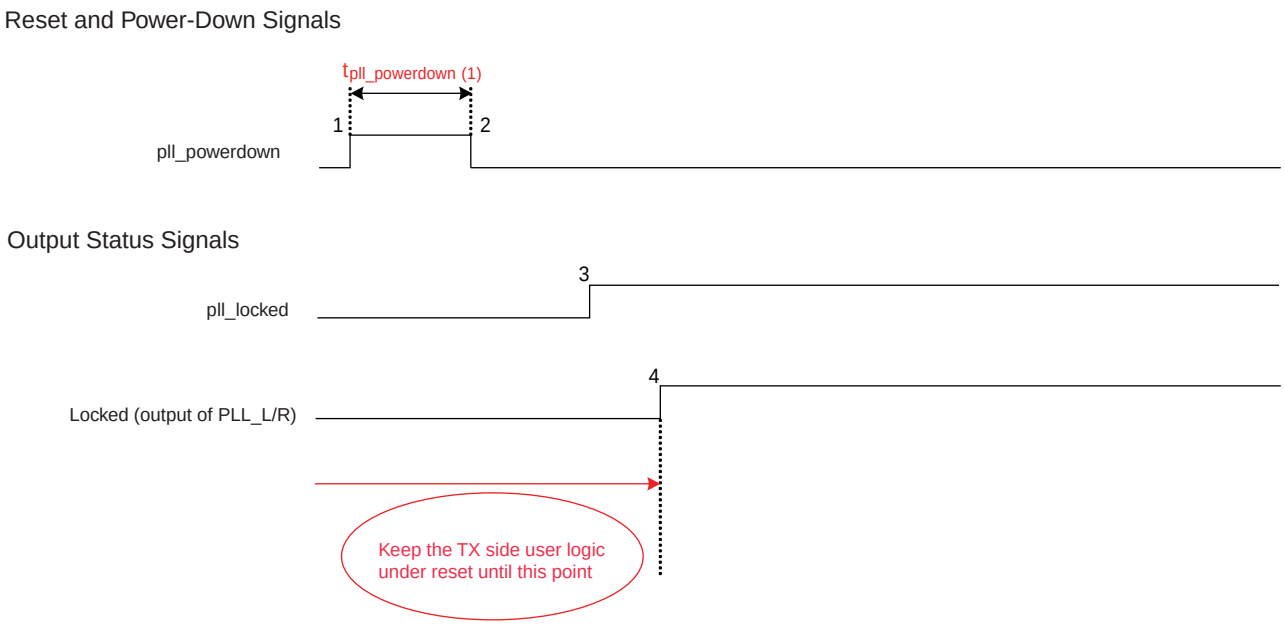
- Same effective data rate. Having the same transmitter local clock divider settings in each channel
- Same FPGA Fabric-to-transceiver interface data path width.
- The transmitter channels must receive the high-speed clock from the same PLL (either CMU PLL or ATX PLL).

Figure 4–15 shows an example reset sequence timing diagram of four **Transmitter Only** channels in Basic (PMA Direct) Drive -x4 functional mode with a PLL_L/R.

As shown in Figure 4–15, perform the following reset procedure for the **Transmitter Only** channel in Basic (PMA Direct) Drive functional mode with a PLL_L/R configuration:

1. After power up, assert `pll_powerdown` for a minimum of $t_{pll_powerdown}$ (the time between markers 1 and 2).
2. After the transmitter PLL locks, as indicated by the `pll_locked` signal going high (marker 3), wait for the locked signal to be asserted. The locked signal is an output of the PLL_L/R.
3. After the PLL_L/R locks, as indicated by the locked signal going high (marker 4), the transmitter is ready to accept parallel data from the FPGA fabric and subsequently transmitting serial data reliably.

Figure 4–15. Reset Sequence Timing Diagram of Four Transmitter-Only Channels in Basic (PMA Direct) Drive ×4 Functional Mode



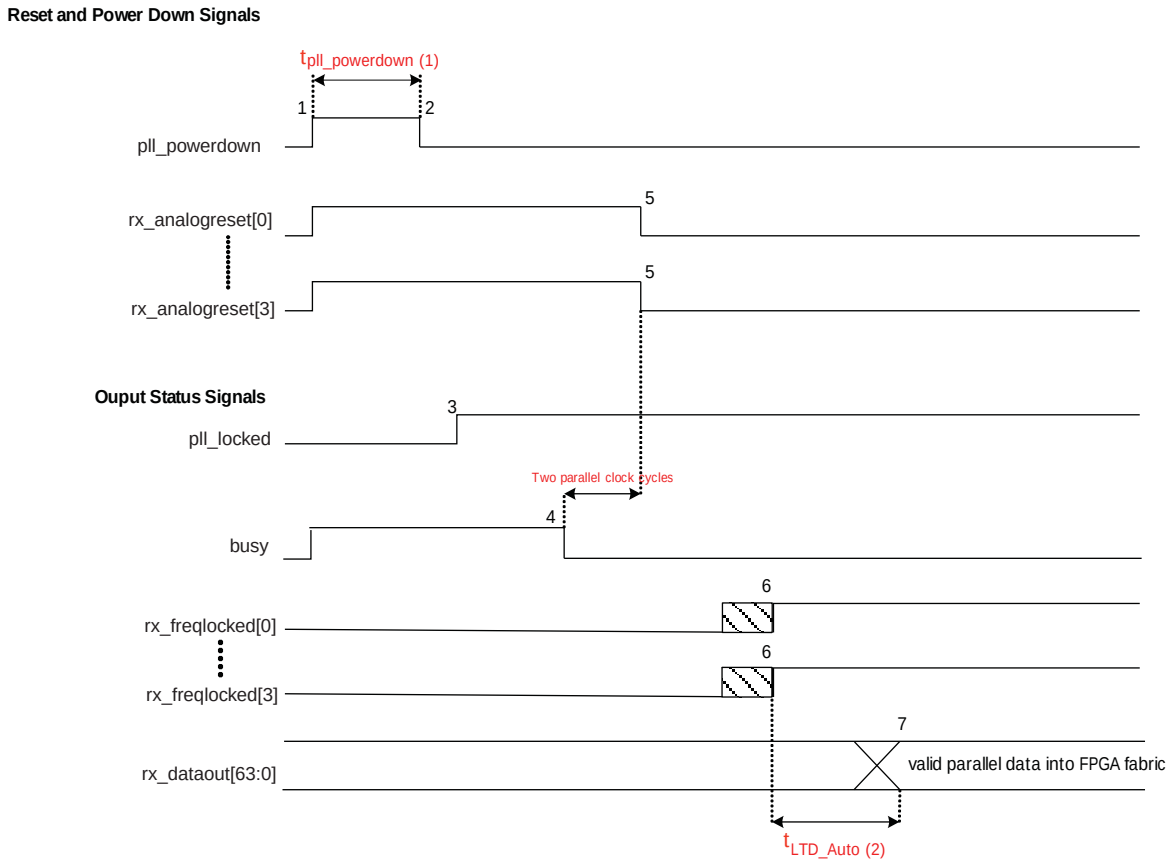
Note to Figure 4–15:

(1) For $t_{pll_powerdown}$ duration, refer to the *DC and Switching Characteristics* chapter in the "Stratix IV Device Datasheet" section.

Receiver and Transmitter Channel Set-up—Receiver CDR in Automatic Lock Mode

This configuration contains both a transmitter and receiver channel. For PMA Direct drive ×N mode, with the receiver CDR in automatic lock mode, use the reset sequence shown in Figure 4–16. In this example, N = 4.

Figure 4–16. Reset Sequence with CDR in Automatic Lock Mode



Notes to Figure 4–16:

(1) For $t_{pll_powerdown}$ duration, refer to the *DC and Switching Characteristics* chapter in the "Stratix IV Device Datasheet" section.

(2) For t_{LTD_Auto} duration, refer to the *DC and Switching Characteristics* chapter in the "Stratix IV Device Datasheet" section.

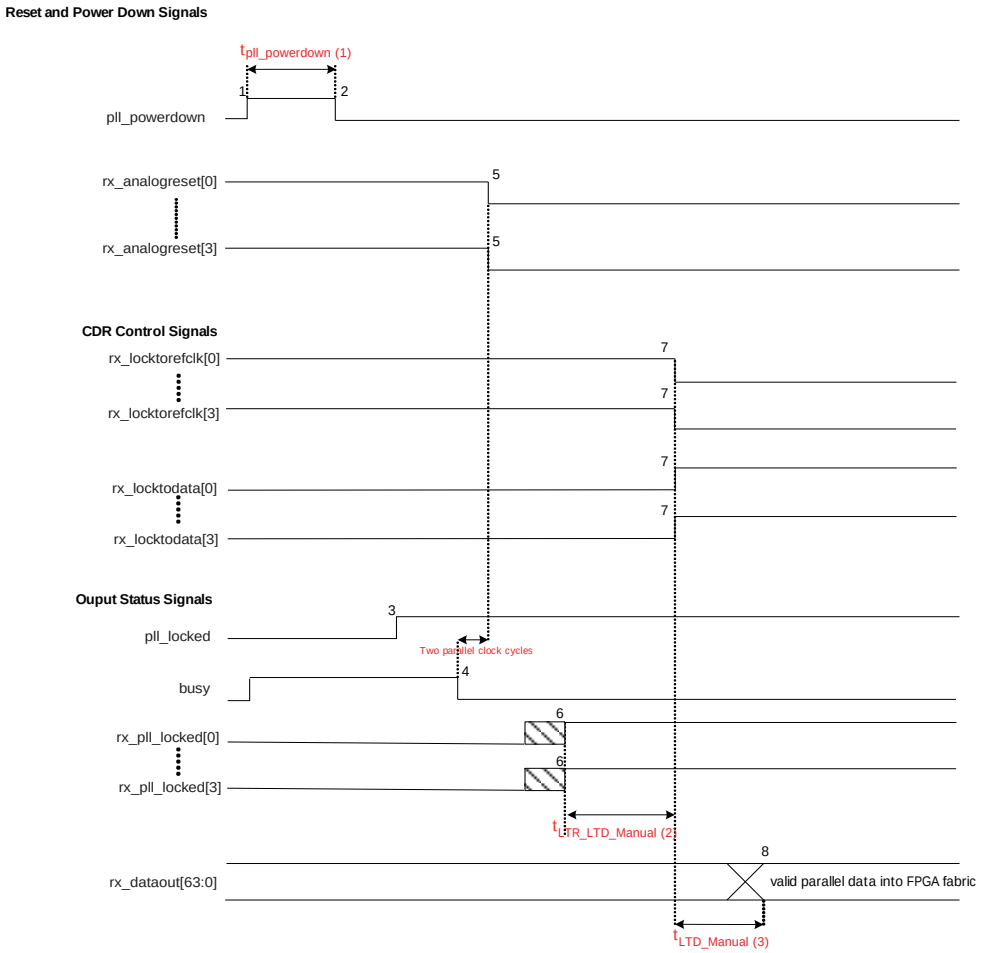
As shown in Figure 4–16, perform the following reset procedure for the receiver and transmitter channel in PMA Direct drive ×4 double-width configuration with CDR in automatic lock mode:

1. After power up, assert `pll_powerdown` for a minimum period of 1 $_{pll_powerdown}$ (the time between markers 1 and 2).
2. Keep the `rx_analogreset` signal asserted during this time period. After you de-assert the `pll_powerdown` signal, the transmitter PLL starts locking to the transmitter input reference clock.
3. When the transmitter PLL locks, as indicated by the `pll_locked` signal going high (marker 3), the transmitter is ready to accept parallel data from the FPGA fabric and transmitting serial data reliably.
4. For the receiver operation, after de-assertion of the `busy` signal, wait for two parallel clock cycles to de-assert the `rx_analogreset` signals of each channel. After `rx_analogreset` is de-asserted, the receiver CDR of each channel starts locking to the receiver input reference clock.
5. Wait for the `rx_freqlocked` signal from each channel to go high. The `rx_freqlocked` signal of each channel may go high at different times (as indicated by the slashed pattern at marker 6).
6. In a PMA Direct drive ×4 double-width configuration, when the `rx_freqlocked` signals of all the channels has gone high (marker 6), from that point onwards, wait for at least t_{LTD_Auto} (marker 7) for the receiver parallel clock to become stable. At this point, all the receivers are ready for transferring valid parallel data into the FPGA fabric (until this time, Altera recommends that the user logic that processes this data be under reset).

Receiver and Transmitter Channel Set-up—Receiver CDR in Manual Lock Mode

This configuration contains both a transmitter and receiver channel. For PMA Direct drive ×N mode, with receiver CDR in manual lock mode, use the reset sequence shown in Figure 4–17. In this example, N = 4.

Figure 4–17. Reset Sequence with CDR in Manual Lock Mode



Notes to Figure 4–17:

(1) For $t_{pll_powerdown}$ duration, refer to the *DC and Switching Characteristics* chapter in the "Stratix IV Device Datasheet" section.

(2) For $t_{LTR_LTD_Manual}$ duration, refer to the *DC and Switching Characteristics* chapter in the "Stratix IV Device Datasheet" section.

(3) For t_{LTD_Manual} duration, refer to the *DC and Switching Characteristics* chapter in the "Stratix IV Device Datasheet" section.

As shown in Figure 4–17, perform the following reset procedure for the receiver and transmitter channel in PMA Direct drive ×4 double-width configuration with CDR in manual lock mode:

1. After power up, assert `pll_powerdown` for a minimum period of $t_{pll_powerdown}$ (the time between markers 1 and 2).
2. Keep the `rx_analogreset` and `rx_locktorefclk` signals asserted and the `rx_locktodata` signal de-asserted during this time period. After you de-assert the `pll_powerdown` signal, the transmitter PLL starts locking to the transmitter input reference clock.
3. When the transmitter PLL locks, as indicated by the `pll_locked` signal going high (marker 3), the transmitter is ready to accept parallel data from the FPGA fabric and transmitting serial data reliably.
4. For the receiver operation, after de-assertion of the `busy` signal (marker 4), wait for two parallel clock cycles to de-assert the `rx_analogreset` signal. After the `rx_analogreset` signal is de-asserted, the receiver CDR of each channel starts locking to the receiver input reference clock because `rx_locktorefclk` is asserted.
5. Wait for the `rx_pll_locked` signal from each channel to go high. The `rx_pll_locked` signal of each channel may go high at different times with respect to each other (indicated by the slashed pattern at marker 6).
6. In a PMA Direct drive ×4 double-width configuration, when the `rx_pll_locked` signal of all the channels has gone high, from that point onwards, wait for at least $t_{LTR_LTD_Manual}$, then de-assert `rx_locktorefclk` and assert `rx_locktodata` (marker 7). At this point, the receiver CDR of all the channels enters into lock-to-data mode and starts locking to the received data.
7. After assertion of the `rx_locktodata` signal, from that point onwards, wait for at least t_{LTD_Manual} (marker 8) for the receiver parallel clock to become stable. At this point, all the receivers are ready for transferring valid parallel data into the FPGA fabric (until this time, Altera recommends that the user logic that processes this data be under reset).

Basic (PMA Direct) Drive x1 Mode

The following timing diagram examples are used to describe the reset and power down sequences for Basic (PMA Direct) drive mode without bonding between the transceiver channels.

Table 4–8 lists the reset and power-down sequences for Basic (PMA Direct) drive ×1 functional mode.

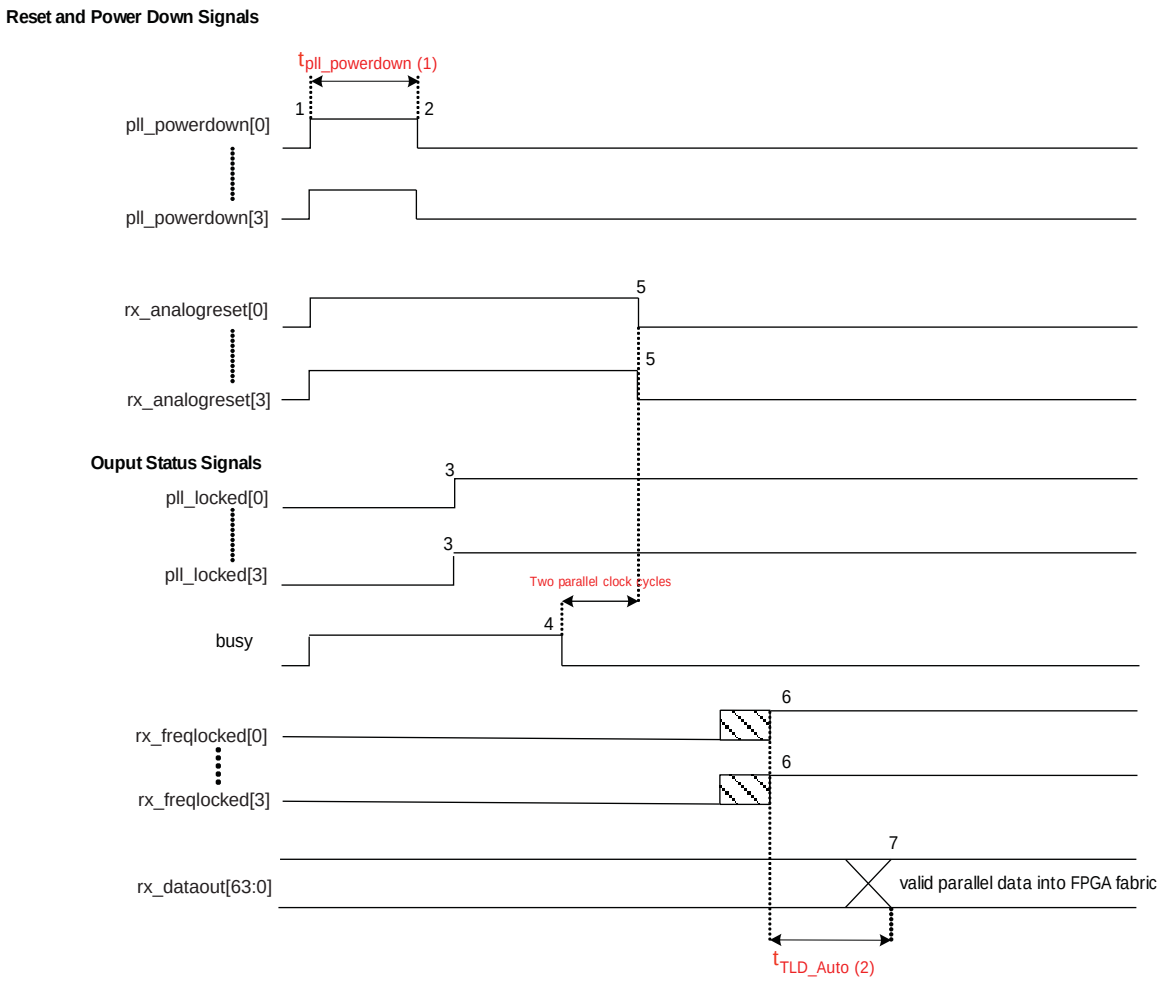
Table 4–8. Reset and Power-Down Sequences for Basic (PMA Direct) Drive ×1 Configurations

Channel Set Up	Functional Mode	Refer to
Receiver and Transmitter	Automatic lock mode for Basic (PMA Direct) drive ×1 mode	"Receiver and Transmitter Channel Set-Up—Receiver CDR in Automatic Lock Mode" on page 4–31
Receiver and Transmitter	Manual lock mode for Basic (PMA Direct) drive ×1 mode	"Receiver and Transmitter Channel Set-up—Receiver CDR in Manual Lock Mode" on page 4–33

Receiver and Transmitter Channel Set-Up—Receiver CDR in Automatic Lock Mode

This configuration contains both a transmitter and receiver channel. For Basic (PMA Direct) drive ×1 mode, with receiver CDR in automatic lock mode, use the reset sequence shown in Figure 4–18. In this example, four channels are configured in this mode.

Figure 4–18. Reset Sequence with CDR in Automatic Lock Mode



Notes to Figure 4–18:

(1) For $t_{pll_powerdown}$ duration, refer to the *DC and Switching Characteristics* chapter in the "Stratix IV Device Datasheet" section.

(2) For t_{LTD_Auto} duration, refer to the *DC and Switching Characteristics* chapter in the "Stratix IV Device Datasheet" section.

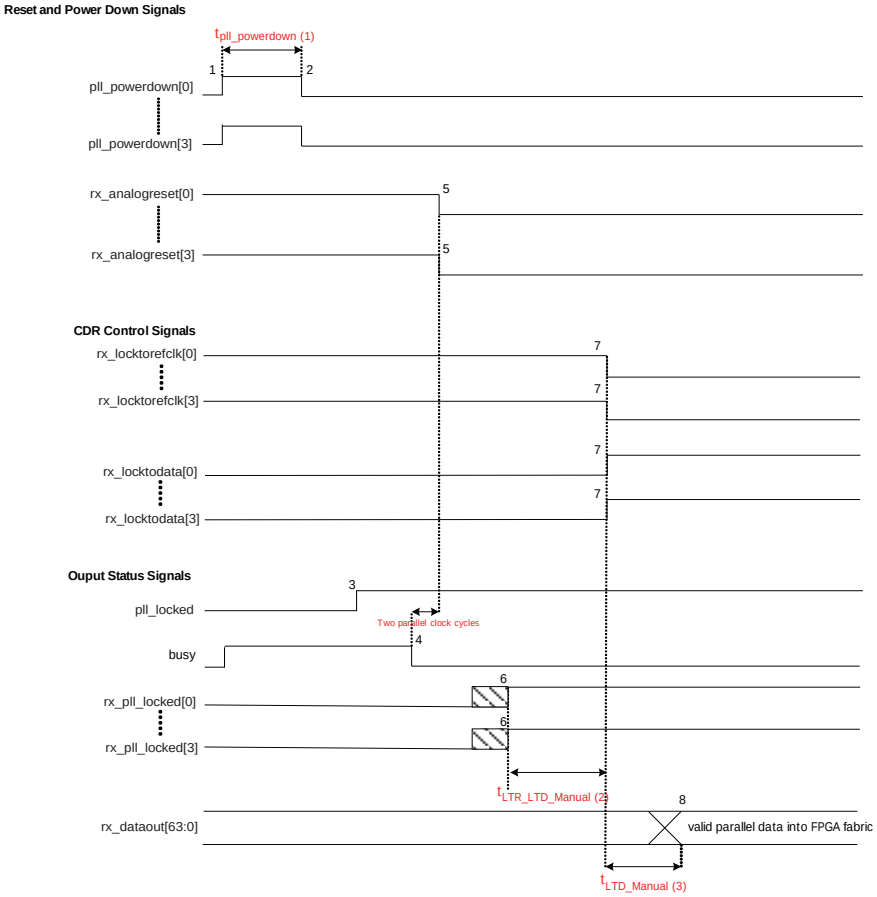
As shown in Figure 4–18, perform the following reset procedure for the receiver and transmitter channel in Basic (PMA Direct) drive double-width configuration, non-bonded with CDR in automatic lock mode:

1. After power up, assert `pll_powerdown` of each channel for a minimum period of $t_{pll_powerdown}$ (the time between markers 1 and 2).
2. Keep the `rx_analogreset` signal of each channel asserted during this time period. After you de-assert the `pll_powerdown` signal on all channels, the transmitter PLL of each channel starts locking to the transmitter input reference clock.
3. When the transmitter PLL locks, as indicated by the `pll_locked` signal going high (marker 3), the transmitters are ready for accepting parallel data from the FPGA fabric and subsequently transmitting serial data reliably.
4. For the receiver operation, after de-assertion of the `busy` signal, wait for two parallel clock cycles to de-assert the `rx_analogreset` signals of each channel. After `rx_analogreset` is de-asserted, the receiver CDR of each channel starts locking to the receiver input reference clock.
5. Wait for the `rx_freqlocked` signal from each channel to go high. The `rx_freqlocked` signal of each channel may go high at different times (indicated by the slashed pattern at marker 6).
6. In a Basic (PMA Direct) drive double-width configuration without bonding between channels, when the `rx_freqlocked` signals of all the channels have gone high (marker 6), from that point onwards, wait for at least t_{LTD_Auto} (marker 7) for the receiver parallel clock to become stable. At this point, all the receivers are ready for transferring valid parallel data into the FPGA fabric (until this time, Altera recommends that the user logic that processes this data be under reset).

Receiver and Transmitter Channel Set-up—Receiver CDR in Manual Lock Mode

This configuration contains both a transmitter and receiver channel. For Basic (PMA Direct) drive ×1 mode, with receiver CDR in manual lock mode, use the reset sequence shown in Figure 4–19. In this example, four channels are configured in this mode.

Figure 4–19. Reset Sequence with CDR in Manual Lock Mode



Notes to Figure 4–19

(1) For $t_{pll_powerdown}$ duration, refer to the *DC and Switching Characteristics* chapter in the "Stratix IV Device Datasheet" section.

(2) For $t_{LTR_LTD_Manual}$ duration, refer to the *DC and Switching Characteristics* chapter in the "Stratix IV Device Datasheet" section.

(3) For t_{LTD_Manual} duration, refer to the *DC and Switching Characteristics* chapter in the "Stratix IV Device Datasheet" section.

As shown in Figure 4–19, perform the following reset procedure for the receiver and transmitter channel in Basic (PMA Direct) drive double-width configuration, non-bonded with CDR in manual lock mode:

1. After power up, assert `pll_powerdown` of each channel for a minimum period of $t_{pll_powerdown}$ (the time between markers 1 and 2).
2. Keep the `rx_analogreset` and `rx_locktorefclk` signals of each channel asserted and the `rx_locktodata` signals de-asserted during this time period. After you de-assert the `pll_powerdown` signal, the transmitter PLL starts locking to the transmitter input reference clock.
3. When the transmitter PLL locks, as indicated by the `pll_locked` signal going high (marker 3), the transmitters are ready to accept parallel data from the FPGA fabric and subsequently transmitting serial data reliably.
4. For the receiver operation, after de-assertion of the `busy` signal (marker 4), wait for two parallel clock cycles to de-assert the `rx_analogreset` signal of each channel. After the `rx_analogreset` signal is de-asserted, the receiver CDR of each channel starts locking to the receiver input reference clock because `rx_locktorefclk` is asserted.
5. Wait for the `rx_pll_locked` signal from each channel to go high. The `rx_pll_locked` signal of each channel may go high at different times with respect to each other (indicated by the slashed pattern at marker 6).
6. In a Basic (PMA Direct) drive double-width configuration without bonding between channels, when the `rx_pll_locked` signal of all the channels has gone high, from that point onwards, wait for at least $t_{LTR_LTD_Manual}$, then de-assert `rx_locktorefclk` and assert `rx_locktodata` (marker 7). At this point, the receiver CDR of all the channels enters into lock-to-data mode and starts locking to the received data.
7. After assertion of the `rx_locktodata` signal, from that point onwards, wait for at least t_{LTD_Manual} (marker 8) for the receiver parallel clock to be stable. At this point, all the receivers are ready for transferring valid parallel data into the FPGA fabric (until this time, Altera recommends that the user logic that processes this data be reset).

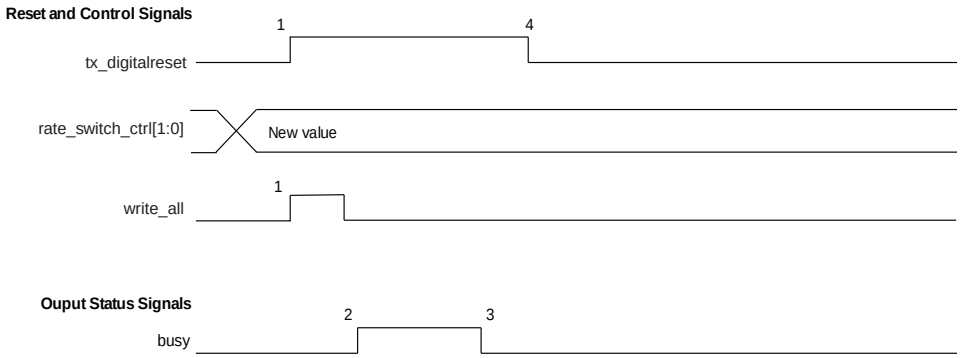
Dynamic Reconfiguration Reset Sequences

When using dynamic reconfiguration in data rate divisions in TX or channel and TX CMU PLL select/reconfig modes, use the following reset sequences.

Reset Sequence when Using Dynamic Reconfiguration with the 'data rate division in TX' Option

Use the example reset sequence shown in Figure 4–20 when you use the dynamic reconfiguration controller to change the data rate of the transceiver channel. In this example, dynamic reconfiguration is used to dynamically reconfigure the data rate of the transceiver channel configured in Basic ×1 mode with the receiver CDR in automatic lock mode.

Figure 4–20. Reset Sequence When Using the Dynamic Reconfiguration Controller to Change the Data Rate of the Transceiver Channel



As shown in Figure 4–20, perform the following reset procedure when using the dynamic reconfiguration controller to change the configuration of the transmitter channel:

1. After power up and properly establishing that the transmitter is operating as desired, write the desired new value for the data rate in the appropriate register (in this example, `rate_switch_ctrl[1:0]`) and subsequently assert the `write_all` signal (marker 1) to initiate the dynamic reconfiguration.

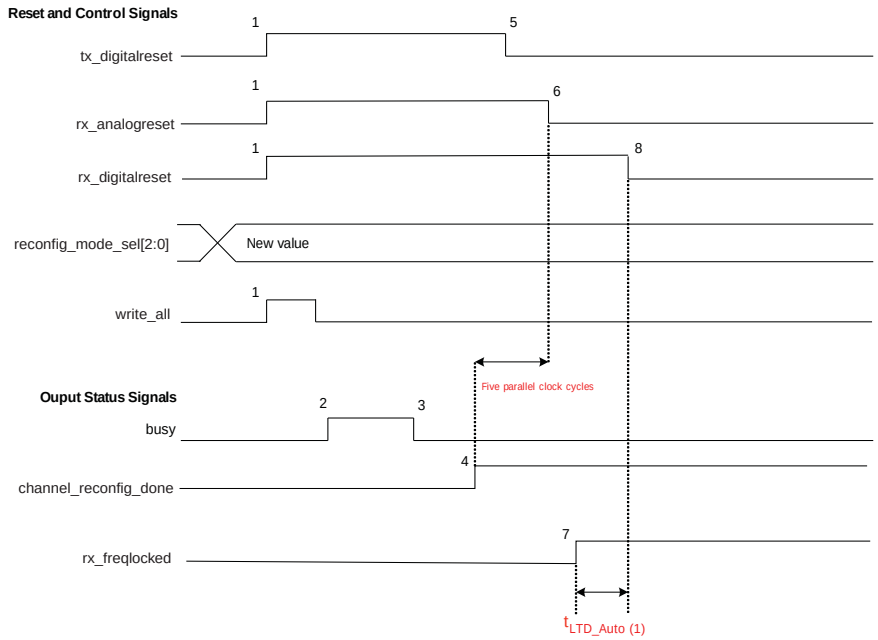
   For more information, refer to the *Stratix IV Dynamic Reconfiguration* chapter.

2. Assert the `tx_digitalreset` signal.
3. As soon as `write_all` is asserted, the dynamic reconfiguration controller starts to execute its operation. This is indicated by the assertion of the `busy` signal (marker 2).
4. After the completion of dynamic reconfiguration, the `busy` signal is de-asserted (marker 3).
5. Lastly, `tx_digitalreset` can be de-asserted to continue with the transmitter operation (marker 4).

Reset Sequence when Using Dynamic Reconfiguration with the 'Channel and TX PLL select/reconfig' Option

Use the example reset sequence shown in Figure 4–21 when you are using the dynamic reconfiguration controller to change the TX PLL settings of the transceiver channel. In this example, the dynamic reconfiguration is used to dynamically reconfigure the data rate of the transceiver channel configured in Basic ×1 mode with receiver CDR in automatic lock mode.

Figure 4–21. Reset Sequence When Using the Dynamic Reconfiguration Controller to Change the TX PLL Settings of the Transceiver Channel



Note to Figure 4–21:

(1) For t_{LTD_Auto} duration, refer to the *DC and Switching Characteristics* chapter in the "Stratix IV Device Datasheet" section.

As shown in Figure 4–21, perform the following reset procedure when using the dynamic reconfiguration controller to change the configuration of the transceiver channel:

1. After power up and establishing that the transceiver is operating as desired, write the desired new value in the appropriate registers (including `reconfig_mode_sel[2:0]`) and subsequently assert the `write_all` signal (marker 1) to initiate the dynamic reconfiguration.

   For more information, refer to the *Stratix IV Dynamic Reconfiguration* chapter.

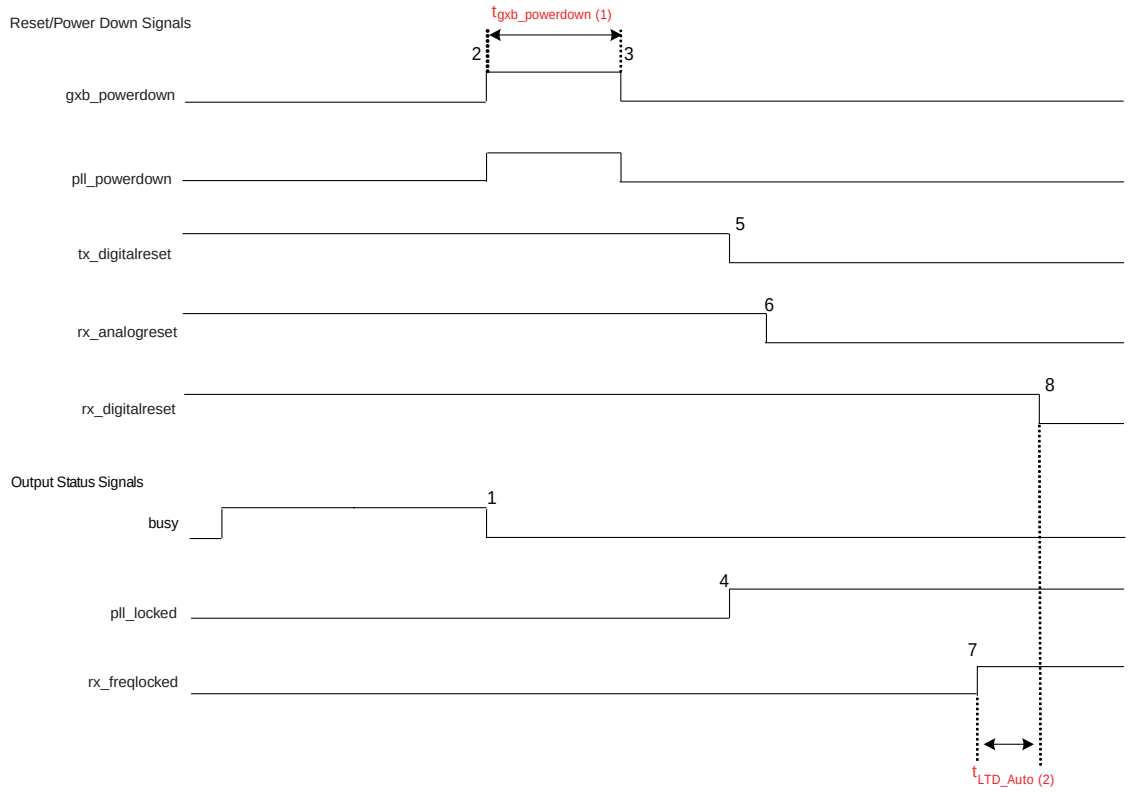
2. Assert the `tx_digitalreset`, `rx_analogreset`, and `rx_digitalreset` signals.
3. As soon as `write_all` is asserted, the dynamic reconfiguration controller starts to execute its operation. This is indicated by the assertion of the `busy` signal (marker 2).
4. Wait for the assertion of the `channel_reconfig_done` signal (marker 4) that indicates the completion of dynamic reconfiguration in this mode.
5. After assertion of the `channel_reconfig_done` signal, de-assert `tx_digitalreset` (marker 5) and wait for at least five parallel clock cycles to de-assert the `rx_analogreset` signal (marker 6).
6. Lastly, wait for the `rx_freqlocked` signal to go high. After `rx_freqlocked` goes high (marker 7), wait for t_{LTD_Auto} to de-assert the `rx_digitalreset` signal (marker 8). At this point, the receiver is ready for data traffic.

Power Down

The Quartus II software automatically selects the power-down channel feature, which takes effect when you configure the Stratix IV GX device. All unused transceiver channels and blocks are powered down to reduce overall power consumption. The `gxb_powerdown` signal is an optional transceiver block signal. It powers down all transceiver channels and all functional blocks in the transceiver block. The minimum pulse width for this signal is 1 µs. After power up, if you use the `gxb_powerdown` signal, wait for de-assertion of the `busy` signal, then assert the `gxb_powerdown` signal for a minimum of 1 µs. Lastly, follow the sequence shown in Figure 4–22.

The de-assertion of the `busy` signal indicates proper completion of the offset cancellation process on the receiver channel.

Figure 4–22. Sample Reset Sequence of Four Receiver and Transmitter Channels-Receiver CDR in Automatic Lock Mode with the Optional gxb_powerdown Signal



Notes to Figure 4–22:

(1) For $t_{gxb_powerdown}$ duration, refer to the *DC and Switching Characteristics* chapter in the "Stratix IV Device Datasheet" section.

(2) For t_{LTD_Auto} duration, refer to the *DC and Switching Characteristics* chapter in the "Stratix IV Device Datasheet" section.

Simulation Requirements

The following are simulation requirements:

- The `gxb_powerdown` port is optional. In simulation, if the `gxb_powerdown` port is not instantiated, you must assert the `tx_digitalreset`, `rx_digitalreset`, and `rx_analogreset` signals appropriately for correct simulation behavior.
- If the `gxb_powerdown` port is instantiated, and the other reset signals are not used, you must assert the `gxb_powerdown` signal for at least one parallel clock cycle for correct simulation behavior.
- You can de-assert the `rx_digitalreset` signal immediately after the `rx_freqlocked` signal goes high to reduce the simulation run time. It is not necessary to wait for t_{LTD_Auto} (as suggested in the actual reset sequence).
- The `busy` signal is de-asserted after about 20 parallel `reconfig_clk` clock cycles in order to reduce simulation run time. For silicon behavior in hardware, you can follow the reset sequences described in the previous pages.
- In PCI Express (PIPE) mode simulation, you must assert the `tx_forceelecidle` signal for at least one parallel clock cycle before transmitting normal data for correct simulation behavior.

Reference Information

For more information about some useful reference terms used in this chapter, refer to the links listed in Table 4–9.

Table 4–9. Reference Information

Terms Used in this Chapter	Useful Reference Points
Automatic Lock Mode	page 4–8
Basic (PMA Direct) Drive x1 Mode	page 4–30
Basic (PMA Direct) Drive xN Mode	page 4–24
Bonded channel configuration	page 4–6
`busy`	page 4–3
Dynamic Reconfiguration Reset Sequences	page 4–34
`gxb_powerdown`	page 4–2
LTD	page 4–6
LTR	page 4–6
Manual Lock Mode	page 4–10
Non-Bonded channel configuration	page 4–14
PCI Express (PIPE)	page 4–21
`pll_locked`	page 4–2
`pll_powerdown`	page 4–2
`rx_analogreset`	page 4–2
`rx_digitalreset`	page 4–2
`rx_freqlocked`	page 4–3
`rx_pll_locked`	page 4–2
`tx_digitalreset`	page 4–2

Document Revision History

Table 4–10 shows the revision history for this chapter.

Table 4–10. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
November 2009, v4.0	■ Added Table 4–1, Table 4–2, Table 4–5, Table 4–6, Table 4–7, and Table 4–8. ■ Added the "Reference Information" section. ■ Updated all figures (except Figure 1). ■ Changed "PLL_powerdown" to "pll_powerdown" throughout. ■ Minor text edits.	Updated all figures (except Figure 1, Figure 2, and Figure 14) and all sections so they use the same terms that are found in the *DC and Switching Characteristics* chapter in the *Stratix IV Device Datasheet* section.
June 2009, v3.1	■ Added new "Transmitter Only Channel with a PLL_L/R" section. ■ Updated the "Transmitter Only Channel with no PLL_L/R" and "Transmitter Only Channel" sections. ■ Minor text edits.	—
March 2009, v3.0	Added: ■ "PMA Direct Drive Mode Reset Sequences" ■ "Dynamic Reconfiguration Reset Sequences"	—
November 2008, v2.0	Added chapter to the *Stratix IV Device Handbook*.	—

5. Stratix IV Dynamic Reconfiguration

SIV52005-3.0

Stratix® IV GX and GT transceivers allow you to dynamically reconfigure different portions of the transceivers without powering down any part of the device. This chapter describes and provides examples about the different modes available for dynamic reconfiguration.

You can use the ALTGX_RECONFIG instance to reconfigure the physical medium attachment (PMA) controls, functional blocks, clock multiplier unit (CMU) phase-locked loops (PLLs), receiver clock data recovery (CDR), and input reference clocks of a transceiver channel.

Additionally, you can monitor the receiver eye width, implement decision feedback control, and achieve adaptive equalization (AEQ) control with dynamic reconfiguration.

This chapter contains the following sections:

- "Glossary of Terms" on page 5–1
- "Dynamic Reconfiguration Controller Architecture" on page 5–3
- "Quartus II MegaWizard Plug-In Manager Interfaces to Support Dynamic Reconfiguration" on page 5–4
- "Dynamic Reconfiguration Modes Implementation" on page 5–12
- "Dynamic Reconfiguration Controller Port List" on page 5–78
- "Error Indication During Dynamic Reconfiguration" on page 5–90
- "Dynamic Reconfiguration Duration" on page 5–92
- "Dynamic Reconfiguration (ALTGX_RECONFIG Instance) Resource Utilization" on page 5–94
- "Functional Simulation of the Dynamic Reconfiguration Process" on page 5–95
- "Dynamic Reconfiguration Examples" on page 5–96

Glossary of Terms

Table 5–1 lists the terms used in this chapter:

Table 5–1. Glossary of Terms Used in this Chapter (Part 1 of 2)

Term	Description
AEQ Control Logic	Adaptive equalization control logic is soft IP that you can enable in the dynamic reconfiguration controller.
AEQ Hardware	Adaptive equalization hardware is circuitry that you can enable in the receiver portion of the transceivers.
ALTGX_RECONFIG Instance	Dynamic reconfiguration controller instance generated by the ALTGX_RECONFIG MegaWizard™ Plug-In Manager.
ALTGX Instance	Transceiver instance generated by the ALTGX MegaWizard Plug-In Manager.

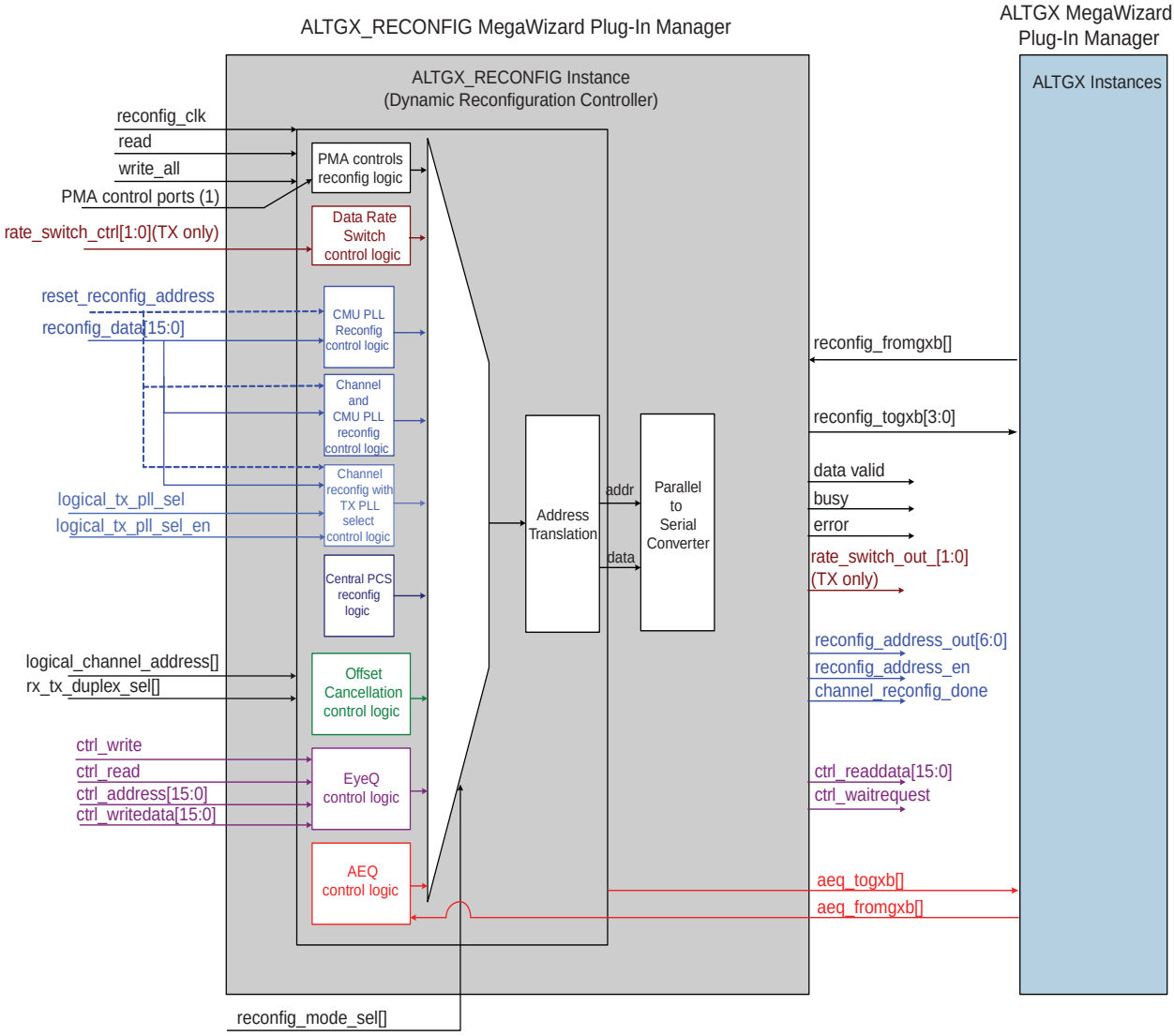
Table 5–1. Glossary of Terms Used in this Chapter (Part 2 of 2)

Term	Description
Alternate CMU transmitter PLL	Refers to one of the two CMU PLLs within one transceiver block.
Channel and transmitter PLL select/reconfig mode	Refers to the following dynamic reconfiguration modes: ■ CMU PLL reconfiguration ■ Channel and CMU PLL reconfiguration ■ Channel reconfiguration with transmitter PLL select ■ Central control unit reconfiguration
Logical Channel Addressing	Used whenever the concept of logical channel addressing is explained. This term does not refer to the `logical_channel_address` port available in the ALTGX_RECONFIG MegaWizard Plug-In Manager.
Logical reference index	Refers to the logical identification value that you must set up for the transmitter PLLs used in the design. You can use a set up value of 0, 1, 2 or 3 in the **Reconfiguration Settings** screen of the ALTGX MegaWizard Plug-In Manager.
logical tx pll	Refers to the logical reference index value of the transmitter PLLs stored in the memory initialization file (**.mif**).
Main PLL	Refers to the transmitter PLL configured in the **General** screen of the ALTGX MegaWizard Plug-In Manager.
Memory Initialization File, also known as **.mif**	When you enable **.mif** generation in your design, a file with the **.mif** extension is generated. This file contains information about the various ALTGX MegaWizard Plug-In Manager options that you set. Each word in the **.mif** is 16 bits wide. The dynamic reconfiguration controller writes information from the **.mif** into the transceiver channel, but only when you use a reconfiguration mode that supports **.mif**-based reconfiguration.
PMA controls	Represents **analog controls** (**Voltage Output Differential** [V_{OD}], **Pre-emphasis**, and **Manual Equalization**) as displayed in both the ALTGX and ALTGX_RECONFIG MegaWizard Plug-In Managers.
PMA-Only channels	Channels configured in Basic (PMA Direct) functional mode.
Regular transceiver channel	Refers to a transmitter channel, a receiver channel, or a duplex channel that has both PMA and physical coding sublayer (PCS) blocks.

Dynamic Reconfiguration Controller Architecture

The dynamic reconfiguration controller is a soft IP that utilizes FPGA-fabric resources. You can use only one controller per transceiver block. You cannot use the dynamic reconfiguration controller to control multiple Stratix IV devices or any off-chip interfaces. Figure 5–1 shows a conceptual view of the dynamic reconfiguration controller architecture. For a detailed description of the inputs and outputs of the ALTGX_RECONFIG instance, refer to "Dynamic Reconfiguration Controller Port List" on page 5–78.

Figure 5–1. Dynamic Reconfiguration Controller



Note to Figure 5–1:

(1) The PMA control ports consist of the V_{OD}, pre-emphasis, DC gain, and manual equalization controls.

You can use only one ALTGX_RECONFIG instance per transceiver block. You may use a single ALTGX_RECONFIG instance with multiple transceiver blocks.

Quartus II MegaWizard Plug-In Manager Interfaces to Support Dynamic Reconfiguration

Stratix IV GX devices provide two MegaWizard Plug-In Manager interfaces to support dynamic reconfiguration—ALTGX and ALTGX_RECONFIG.

ALTGX MegaWizard Plug-In Manager

Use the ALTGX MegaWizard Plug-In manager to enable dynamic reconfiguration settings for the transceiver instances.

For more information, refer to the "Reconfiguration Settings" section of the *ALTGX Transceiver Setup Guide* chapter.

The reconfig_clk Clock Requirements for the ALTGX Instance

You must connect the `reconfig_clk` port to the ALTGX instance in all the configurations using the dynamic reconfiguration feature.

Table 5–2 lists the source clock for the offset cancellation circuit in the ALTGX instance, based on its configuration.

Table 5–2. Source Clock for the Offset Cancellation Circuit in the ALTGX Instance

Source Clock for the Offset Cancellation Circuit *(1)*	ALTGX Configurations
`reconfig_clk`	**Receiver only** and **Transmitter only**
`reconfig_clk`	**Receiver and Transmitter**
`fixedclk`	PCI Express (PIPE)

Note to Figure 5–2:

(1) The clock source used for offset cancellation must be a free running clock that is not derived from the PLL as this clock is required for offset cancellation at power up.

Select the `reconfig_clk` frequency based on the ALTGX configuration shown in Table 5–3. This clock must be a free-running clock sourced from an I/O clock pin. Do not use dedicated transceiver REFCLK pins or any clocks generated by transceivers.

Altera recommends that you drive the `reconfig_clk` signal on a global clock resource.

Table 5–3. reconfig_clk Settings for the ALTGX Instance

ALTGX Instance Configuration	reconfig_clk Frequency Range (MHz)
Receiver and Transmitter	37.5 to 50
Receiver only	37.5 to 50
Transmitter only and PCI Express (PIPE)	2.5 *(1)* to 50

Note to Figure 5–3:

(1) The source clock for the offset cancellation circuit in the ALTGX instance must be faster than 37.5 MHz. Offset cancellation is not required for transmitters and is accomplished using a fixed clock in PCI Express (PIPE) mode.

ALTGX_RECONFIG MegaWizard Plug-In Manager

Use the ALTGX_RECONFIG MegaWizard Plug-In Manager to instantiate the dynamic reconfiguration controller.

For more information, refer to the *ALTGX_RECONFIG Megafunction User Guide*.

The reconfig_clk Clock Requirements for the ALTGX_RECONFIG Instance

You must connect the `reconfig_clk` input port of the ALTGX_RECONFIG instance to the same clock that is connected to the `reconfig_clk` input port of the ALTGX instance.

Table 5–3 on page 5–4 lists the range of frequency values allowed for the `reconfig_clk` input port for **Receiver only**, **Receiver and Transmitter**, and **Transmitter only** configuration modes of the ALTGX instance.

Based on the ALTGX configurations (**Receiver only**, **Transmitter only**, and **Receiver and Transmitter**) controlled by the ALTGX_RECONFIG instance, select the fastest `reconfig_clk` frequency value. This satisfies both the offset cancellation control for the receiver channels and the dynamic reconfiguration of the transmitter and receiver channels.

Interfacing ALTGX and ALTGX_RECONFIG Instances

To dynamically reconfigure the transceiver channel, you must understand the concepts related to interfacing the transceivers with the dynamic reconfiguration controller. These concepts are:

- "Logical Channel Addressing" on page 5–5
- "Total Number of Channels Option in the ALTGX_RECONFIG Instance" on page 5–10
- "Connecting the ALTGX and ALTGX_RECONFIG Instances" on page 5–11

Logical Channel Addressing

The dynamic reconfiguration controller identifies a transceiver channel by using the logical channel address. The **What is the starting channel number?** option in the ALTGX MegaWizard Plug-In Manager allows you to set the logical channel address of all the channels within the ALTGX instance.

For channel reconfiguration with transmitter PLL select mode, the logical channel addressing concept extends to transmitter PLLs. For more information, refer to "Logical Channel Addressing When Using Additional PLLs" on page 5–52.

The following sections describe the concept of logical channel addressing for ALTGX instances configured with:

- Regular transceiver channels (PCS and PMA channels)
- PMA-Only channels
- A combination of PMA-Only channels and regular transceiver channels

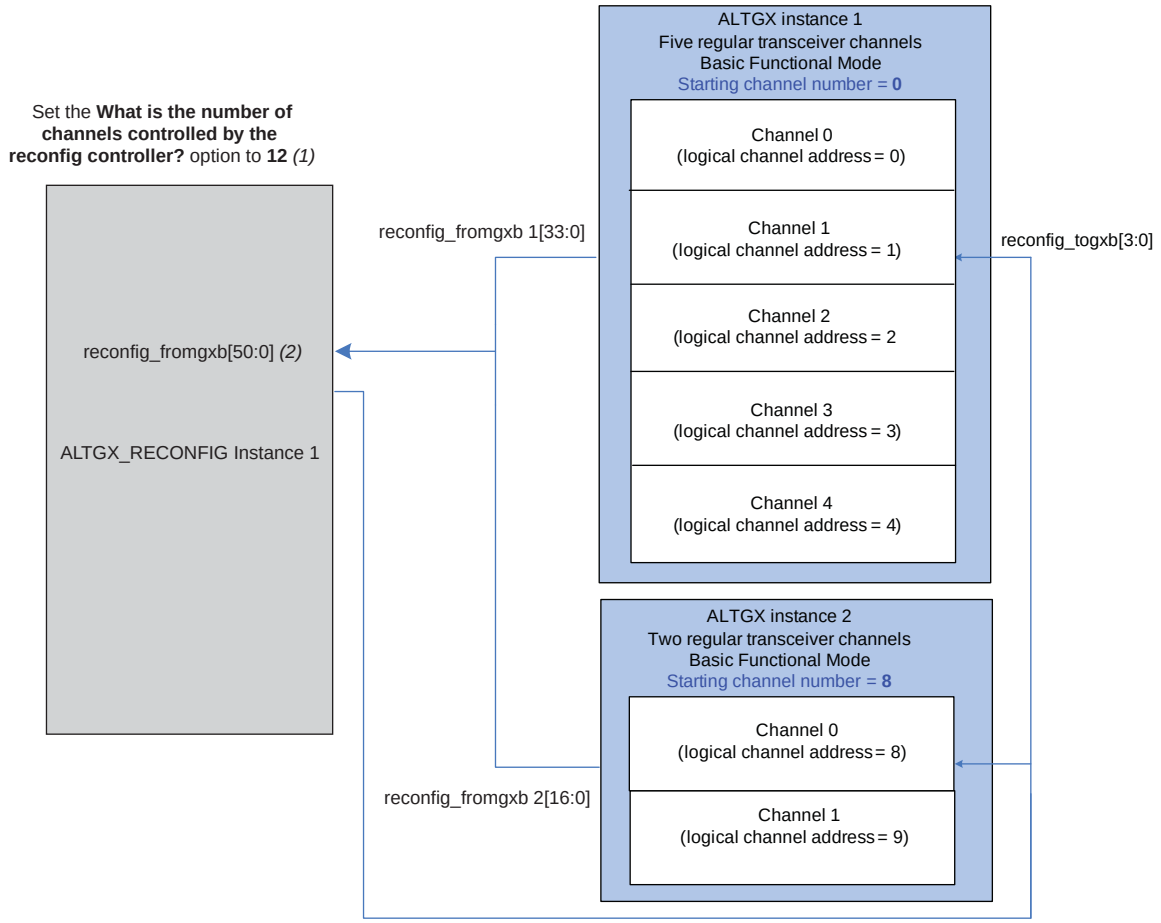
Logical Channel Addressing of Regular Transceiver Channels

For a single ALTGX instance connected to the dynamic reconfiguration controller, set the starting channel number to **0**. The logical channel addresses of the first channel within the ALTGX instance is 0. The logical channel addresses of the remaining channels increment by one.

For multiple ALTGX instances connected to the dynamic reconfiguration controller, set the starting channel number of the first instance to **0**. For the starting channel number for the following ALTGX instances, you must set the next multiple of four. The logical channel address of channels within each ALTGX instance increment by one.

Figure 5–2 shows how to set the starting channel number for multiple ALTGX instances controlled by a single dynamic reconfiguration controller, where both ALTGX instances have regular transceiver channels.

Figure 5–2. Logical Channel Addressing of Regular Transceiver Channels



Notes to Figure 5–2:

(1) For more information, refer to "Total Number of Channels Option in the ALTGX_RECONFIG Instance" on page 5–10.

(2) reconfig_fromgxb[50:0] = { reconfig_fromgxb 2[16:0], reconfig_fromgxb 1[33:0]}.

Logical Channel Addressing of PMA-Only Channels

CMU channels are always PMA-Only channels. The regular transceiver channels can be optionally configured as PMA-Only channels.

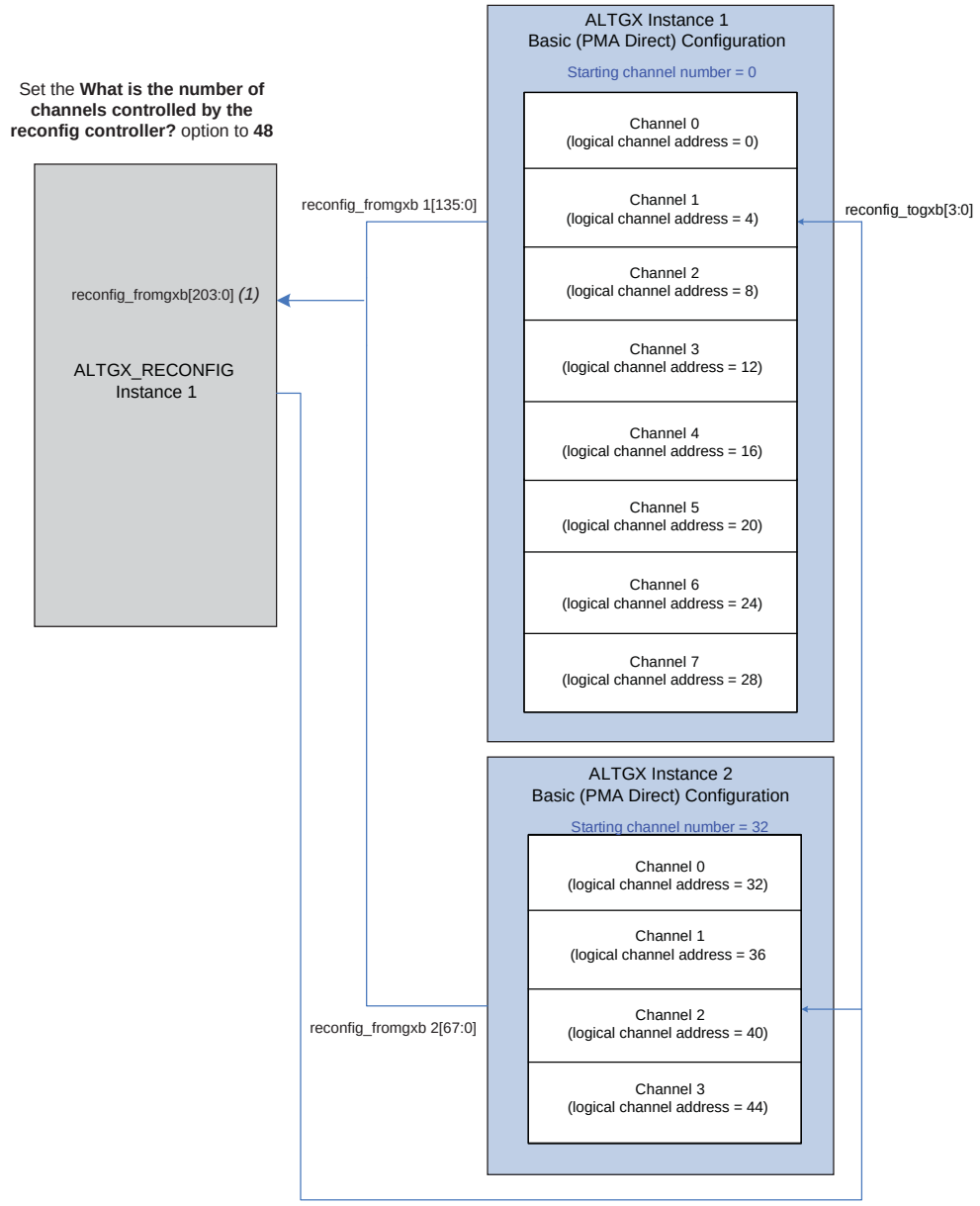
Set the starting channel number for the PMA-Only channels in the **What is the starting channel number?** option in the ALTGX MegaWizard Plug-In Manager.

For a single ALTGX instance connected to the dynamic reconfiguration controller, set the starting channel number to **0**. The logical channel address of the first channel in the ALTGX instance is 0. The logical channel addresses of the PMA-Only channels within the same ALTGX instance increment in multiples of four (unlike the logical channel addressing of regular transceiver channels that are not configured in Basic [PMA Direct] functional mode, where the logical channel address increments in steps of one within the same ALTGX instance).

For multiple ALTGX instances connected to the dynamic reconfiguration controller, set the starting channel number of the first instance to **0**. You must set the next multiple of four as the starting channel number for the remaining ALTGX instances.

Figure 5–3 shows how to set the starting channel number for multiple ALTGX instances controlled by a single dynamic reconfiguration controller, where both ALTGX instances have PMA-Only channels. For more information about the **What is the number of channels controlled by the reconfig controller?** option, refer to "Total Number of Channels Option in the ALTGX_RECONFIG Instance" on page 5–10.

Figure 5–3. Logical Channel Addressing of PMA-Only Channels



Note to Figure 5–3:

(1) reconfig_fromgxb[203:0] = { reconfig_fromgxb 2[67:0], reconfig_fromgxb 1[135:0]}.

Logical Channel Addressing—Combination of Regular Transceiver Channels and PMA-Only Channels

For a combination of regular transceiver channels and PMA-Only channels, there must be at least two different ALTGX instances connected to the same dynamic reconfiguration controller. This is because you cannot have a combination of regular transceiver channels and PMA-Only channels within the same ALTGX instance.

Set the starting channel number in the first ALTGX Instance 1 to **0**. If you have configured ALTGX Instance 1 with regular transceiver channels, the logical channel addresses of the remaining channels increment in steps of one.

Set the starting channel number of the following ALTGX Instance 2 as the next multiple of four. If you have configured ALTGX Instance 2 with PMA-Only channels, the logical channel addresses of the remaining channels increment in steps of four.

Figure 5–42 in "Example 1" on page 5–96 shows how to set the starting channel number for multiple ALTGX instances controlled by a single dynamic reconfiguration controller, where one ALTGX instance has PMA-Only channels and the other ALTGX instance has regular transceiver channels.

Table 5–18 in "Example 1" on page 5–96 lists an example scenario where the logical channel address of both the PMA-Only channels and regular transceiver channels is set based on the starting channel number.

For more information, refer to "Section 1" on page 5–97 of "Example 1" on page 5–96.

Highest Possible Logical Channel Address

Table 5–4 lists the highest possible logical channel address assigned to a transceiver channel in a Stratix IV device.

The maximum number of transceiver channels in the largest Stratix IV device is 48 (24 transceiver channels located in four transceiver blocks on the right side of the device and 24 transceiver channels located in four transceiver blocks on the left side of the device).

You can individually configure these 48 transceiver channels as 48 **Transmitter only** and 48 **Receiver only** channels. You achieve this by using 48 **Transmitter only** ALTGX instances and 48 **Receiver only** ALTGX instances in your design.

Table 5–4. Highest Possible Logical Channel Address

96 ALTGX Instances			ALTGX_RECONFIG Instance
ALTGX MegaWizard Plug-In Manager Setting	ALTGX instance 1	ALTGX instance 2	ALTGX_RECONFIG instance 1: Controls all 96 ALTGX instances.
What is the number of channels? in the **General** screen	**48**	**48**	
What is the starting channel number? in the **Reconfig** screen	TX instance 1: **0** TX instance 2: **4** . . . . . . TX instance 48: **188**	RX instance 1: **192** RX instance 2: **196** . . . . . . RX instance 48: **380**	

The highest logical channel address is assigned to the **Receiver only** channel in the 96th ALTGX instance; therefore, the setting is **380**.

The highest possible logical channel address assigned to a transceiver channel in a Stratix IV device is the same whether the channel is a regular transceiver channel or a PMA-Only channel.

Total Number of Channels Option in the ALTGX_RECONFIG Instance

You can connect every dynamic reconfiguration controller in a design to either a single ALTGX instance or to multiple ALTGX instances. Depending on the number of channels within each of these ALTGX instances, you must set the total number of channels controlled by the dynamic reconfiguration controller in the ALTGX_RECONFIG MegaWizard Plug-In Manager. Based on this information, the `reconfig_fromgxb` and `logical_channel_address` input ports vary in width.

Use the following steps to determine the number of channels:

1. Determine the highest logical channel address among all the transceiver instances connected to the same dynamic reconfiguration controller. For more information, refer to "Logical Channel Addressing" on page 5–5.
2. Round the logical channel address value to the next higher multiple of four.
3. Use this value to set the **What is the number of channels controlled by the reconfig controller?** option.

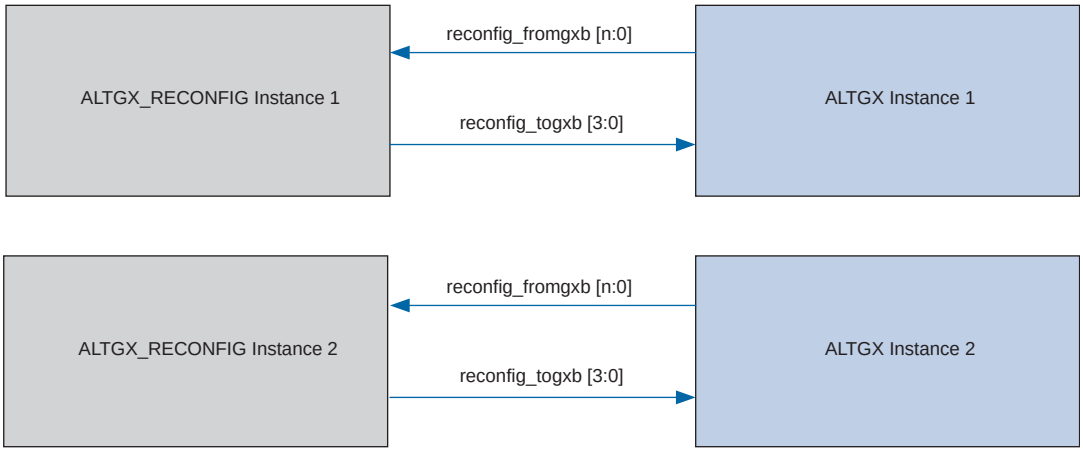
For more information, refer to "Section 1" on page 5–97 of "Example 1" on page 5–96.

Connecting the ALTGX and ALTGX_RECONFIG Instances

There are two ways to connect the ALTGX_RECONFIG instance to the ALTGX instance in your design:

- Single dynamic reconfiguration controller—You can use a single ALTGX_RECONFIG instance to control all the ALTGX instances in your design. Figure 5–2 on page 5–6 shows a block diagram of a single dynamic reconfiguration controller in a design.
- Multiple dynamic reconfiguration controllers—Your design can have multiple ALTGX_RECONFIG instances but you can use only one ALTGX_RECONFIG instance per transceiver block, as shown in Figure 5–4.

Figure 5–4. Multiple Dynamic Reconfiguration Controllers in a Design



In the dynamic reconfiguration interface, you must connect the `reconfig_fromgxb` and `reconfig_togxb` signals between the ALTGX_RECONFIG instance and the ALTGX instance to successfully complete the dynamic reconfiguration process. Make the following connections:

- Connect the `reconfig_fromgxb` input port of the ALTGX_RECONFIG instance to the `reconfig_fromgxb` output ports of all the ALTGX instances controlled by the ALTGX_RECONFIG instance.
- Connect the `reconfig_fromgxb` port of the ALTGX instance whose starting channel number is 0, to the lowest significant bit of the `reconfig_fromgxb` input port of the ALTGX_RECONFIG instance.
- Connect the `reconfig_fromgxb` port of the ALTGX instance with the next highest starting channel number to the following bits of the `reconfig_fromgxb` of the ALTGX_RECONFIG instance, and so on.
- Connect the same `reconfig_togxb` ports of all the ALTGX instances controlled by the ALTGX_RECONFIG instance to the `reconfig_togxb` output port of the ALTGX_RECONFIG instance. The `reconfig_togxb` output port is fixed to 3 bits.

Connecting reconfig_fromgxb for the Regular Transceiver Channels

Figure 5–3 on page 5–8 shows how to connect the `reconfig_fromgxb` output port of the ALTGX instance to the `reconfig_fromgxb` input port of the ALTGX_RECONFIG instance for regular transceiver channels.

Table 5–18 in "Example 1" on page 5–96 describes how to connect the `reconfig_fromgxb` port for regular transceiver channels.

Connecting reconfig_fromgxb for the PMA-Only Channels

Figure 5–3 on page 5–8 shows how to connect the `reconfig_fromgxb` output port of the ALTGX instance to the `reconfig_fromgxb` input port of the ALTGX_RECONFIG instance for PMA-Only channels.

Table 5–18 in "Example 1" on page 5–96 describes how to connect the `reconfig_fromgxb` port for PMA-Only channels.

Dynamic Reconfiguration Modes Implementation

The modes available for dynamically reconfiguring the Stratix IV transceivers are:

- "PMA Controls Reconfiguration Mode Details" on page 5–12
- "Transceiver Channel Reconfiguration Mode Details" on page 5–19
  - Channel and CMU PLL reconfiguration (**.mif** based)
  - Channel reconfiguration with transmitter PLL select (**.mif** based)
  - CMU PLL reconfiguration (**.mif** based)
  - Central control unit reconfiguration (**.mif** based)
  - Data rate division in transmitter
- "Offset Cancellation Feature" on page 5–66
- "EyeQ" on page 5–69
- "Adaptive Equalization (AEQ)" on page 5–74
- "Dynamic Reconfiguration Controller Port List" on page 5–78

The following sections describe each of these modes in detail.

PMA Controls Reconfiguration Mode Details

You can dynamically reconfigure the following PMA controls for both regular transceiver channels and PMA-Only channels:

- Pre-emphasis settings
- Equalization settings
- DC gain settings
- Voltage output differential (V_{OD}) settings

PMA controls reconfiguration is available for all supported transceiver configurations (ALTGX configurations).

The following section describes how to connect the transceiver channels (the ALTGX instance) to the dynamic reconfiguration controller (the ALTGX_RECONFIG instance) to dynamically reconfigure the PMA controls.

The PMA control ports for the ALTGX_RECONFIG MegaWizard Plug-In Manager are available in the **Analog controls** screen. You can select the PMA control ports you want to reconfigure. For example, to use `tx_vodctrl` to write new V_{OD} settings or to use `tx_vodctrl_out` to read the existing V_{OD} settings.

Dynamically Reconfiguring PMA Controls

You can dynamically reconfigure the PMA controls of a transceiver channel using three methods:

- Reconfiguring the PMA controls of a specific transceiver channel. For more information, refer to "Method 1—Using the logical_channel_address Port" on page 5–13.
- Dynamically reconfiguring the PMA controls of the transceiver channels without using the `logical_channel_address` port (where all transceiver channels are reconfigured). If you use this method, the PMA controls of all the transceiver channels connected to the dynamic reconfiguration controller are reconfigured. For more information, refer to "Method 2—Using the Same Control Signals for All Channels" on page 5–15.
- Dynamically reconfiguring the PMA controls of the transceiver channels without using the `logical_channel_address` port (where only the PMA controls of the transceiver channels are reconfigured). If you use this method, each channel has its own PMA control port. Based on the value set at the ports, the PMA controls of the corresponding transceiver channels are reconfigured. For more information, refer to "Method 3—Using Individual Control Signals for Each Channel" on page 5–17.

For the above three methods, you can additionally use the `rx_tx_duplex_sel[1:0]` port transmitter and receiver parameters. For more information, refer to "Dynamic Reconfiguration Controller Port List" on page 5–78.

Method 1—Using the logical_channel_address Port

Using Method 1, you can dynamically reconfigure the PMA controls of a transceiver channel by using the `logical_channel_address` port without affecting the remaining active channels. Enable the `logical_channel_address` port by selecting the **Use 'logical_channel_address' port for Analog controls reconfiguration** option in the **Analog controls** screen of the ALTGX_RECONFIG MegaWizard Plug-In Manager.

This method is applicable only for a design where the dynamic reconfiguration controller controls more than one channel.

When using Method 1, the selected PMA control write and read ports remain fixed in width, regardless of the number of channels controlled by the ALTGX_RECONFIG instance.

To observe the width of the PMA control ports, refer to the ALTGX_RECONFIG MegaWizard Plug-In Manager.

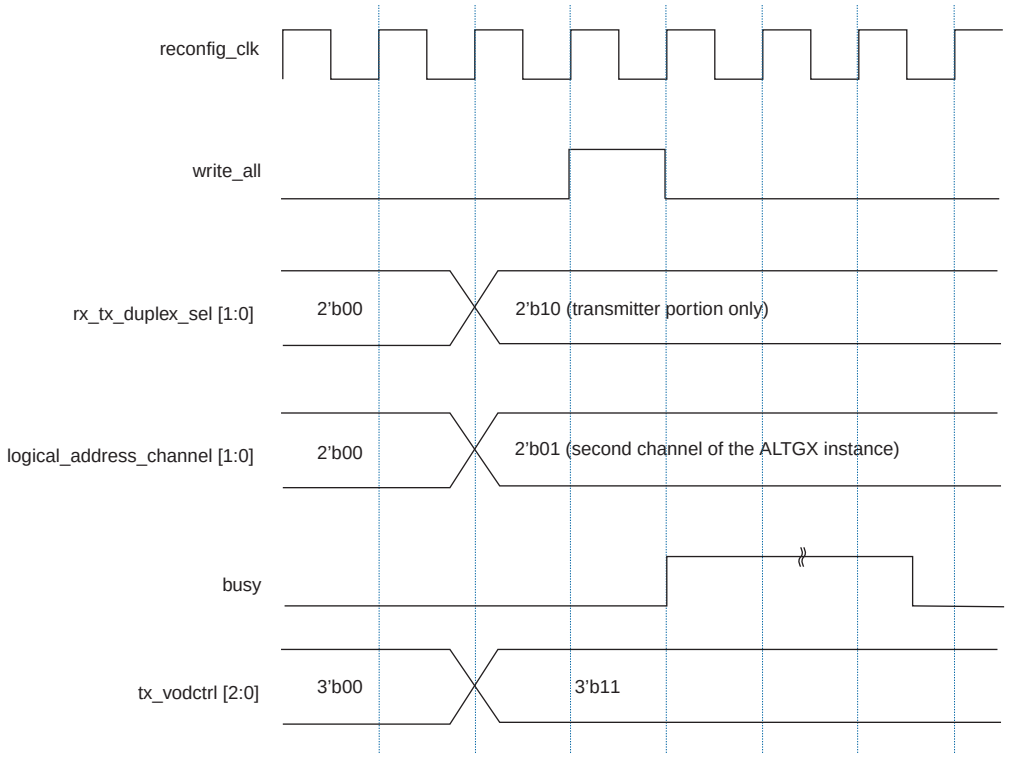
The value you set at the PMA control ports is only written into the specified transceiver channel.

Ensure that the `busy` signal is low before you start a write or read transaction. The `busy` output status signal is asserted high when the dynamic reconfiguration controller is occupied writing or reading the PMA control values. When the write or read transaction has completed, the `busy` signal goes low.

Write Transaction

Figure 5–5 shows the write transaction waveform when using Method 1. In this example, the number of channels connected to the dynamic reconfiguration controller is four. Therefore, the `logical_channel_address` port is 2 bits wide. Also, to initiate the write transaction, you must assert the `write_all` signal for one `reconfig_clk` cycle.

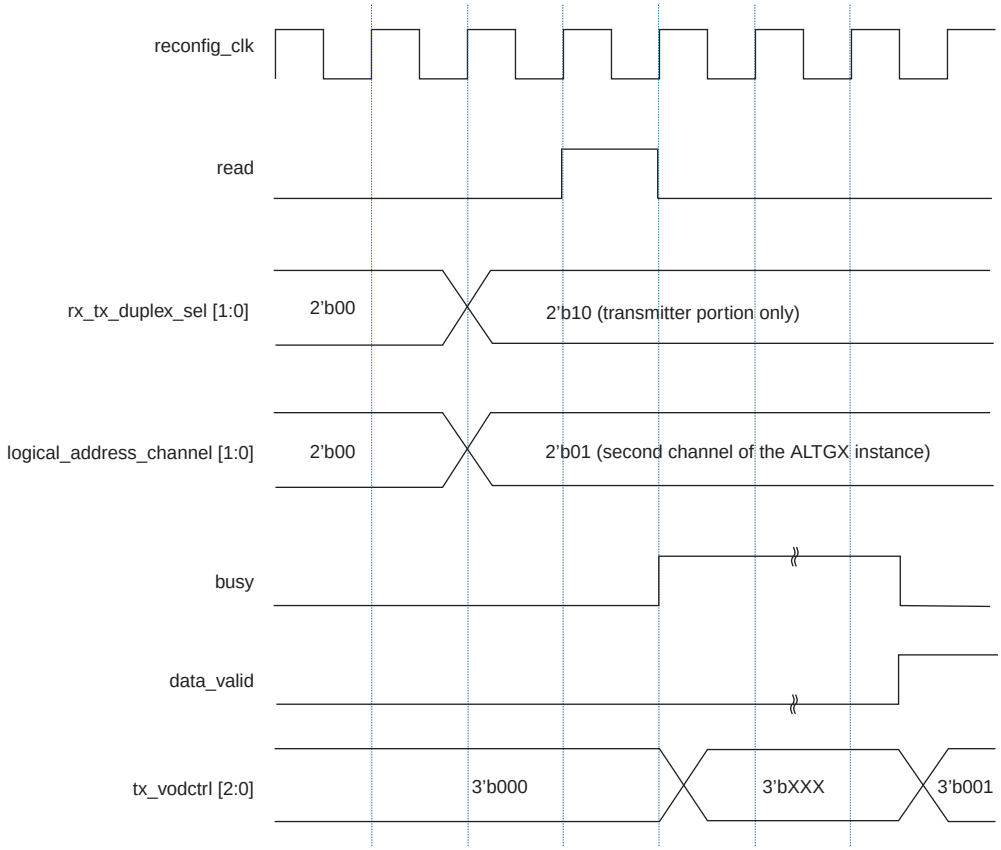
Figure 5–5. Method 1—Write Transaction Waveform



Read Transaction

In this example, you want to read the existing V_{OD} values from the transmit V_{OD} control registers of the transmitter portion of a specific channel controlled by the ALTGX_RECONFIG instance. For this example, the number of channels connected to the dynamic reconfiguration controller is four. Therefore, the `logical_channel_address` port is 2 bits wide. Also, to initiate the read transaction, assert the read signal for one `reconfig_clk` clock cycle. After the read transaction has completed, the `data_valid` signal is asserted. Figure 5–6 shows the read transaction waveform.

Figure 5–6. Method 1—Read Transaction Waveform



Simultaneous write and read transactions are not allowed.

Method 2—Using the Same Control Signals for All Channels

To use Method 2, enable the **Use the same control signal for all channels** option in the **Analog controls** screen of the ALTGX_RECONFIG MegaWizard Plug-In Manager.

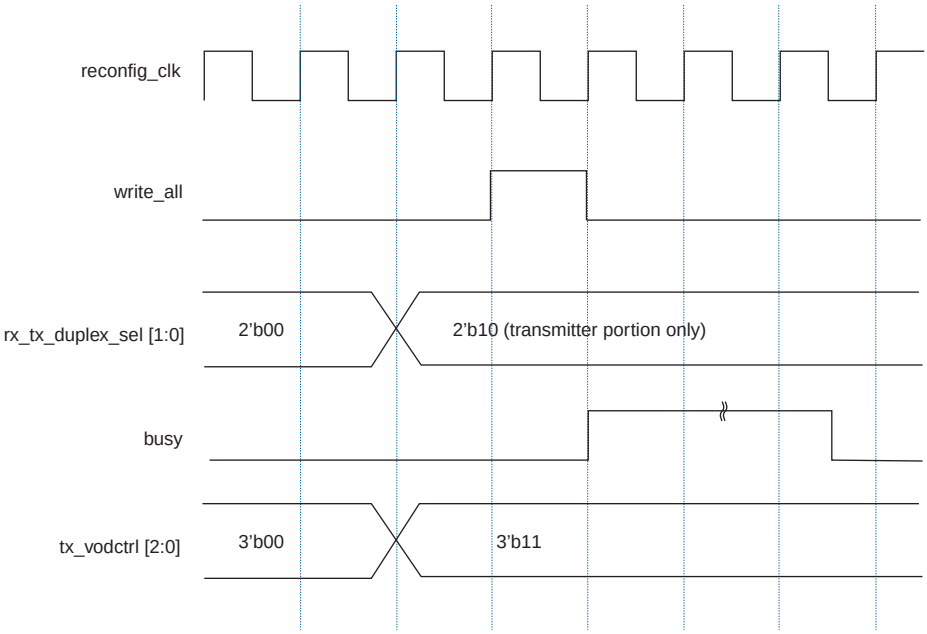
Using Method 2, you can write the same PMA control value into all the transceiver channels connected to the dynamic reconfiguration controller.

The PMA control `write` ports remain fixed in width irrespective of the number of channels controlled by the ALTGX_RECONFIG instance. The PMA control `read` ports increase in width based on the number of channels controlled by the ALTGX_RECONFIG instance.

Write Transaction

Assume that you have enabled `tx_vodctrl` in the ALTGX_RECONFIG MegaWizard Plug-In Manager to reconfigure the V_{OD} of the transceiver channels. Figure 5–7 shows the write transaction to reconfigure the V_{OD}.

Figure 5–7. Method 2—Write Transaction Waveform



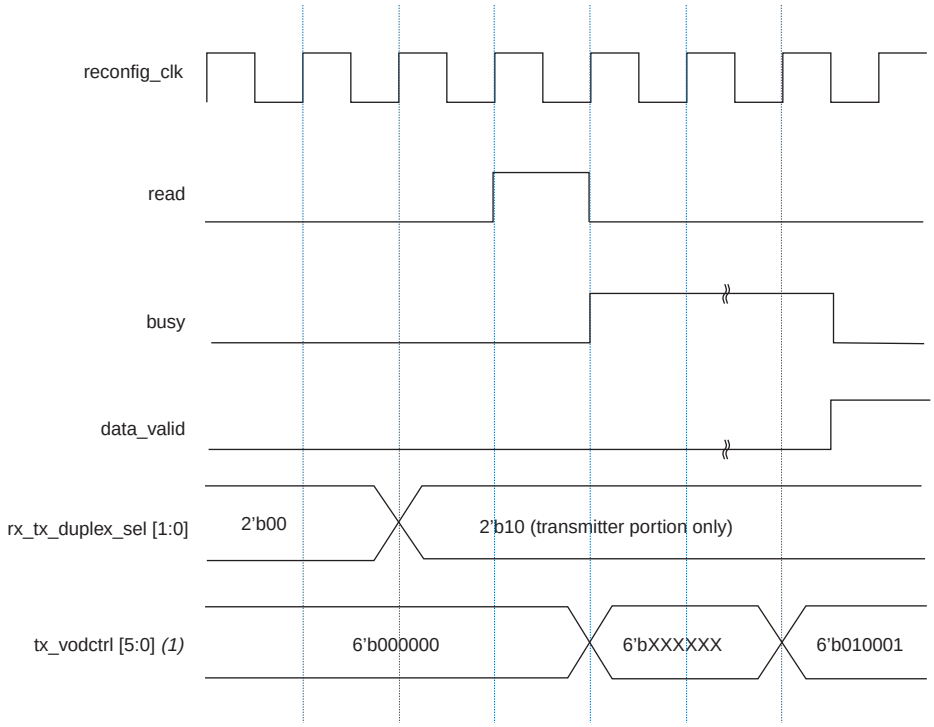
Read Transaction

If you want to read the existing values from a specific channel connected to the ALTGX_RECONFIG instance, observe the corresponding byte positions of the PMA control output port after the read transaction is complete.

For example, if the number of channels controlled by the ALTGX_RECONFIG instance is two, `tx_vodctrl_out` is 6 bits wide (`tx_vodctrl_out[2:0]` corresponds to channel 1 and `tx_vodctrl_out[5:3]` corresponds to channel 2). Figure 5–8 shows how to read the V_{OD} values of the second channel.

Figure 5–8 shows the read transaction waveform. The transmit V_{OD} settings written in channels 1 and 2 prior to the read transaction are 3'b001 and 3'b010, respectively.

Figure 5–8. Method 2—Read Transaction Waveform



Note to Figure 5–8:

(1) To read the current V_{OD} values in channel 2, observe the values in `tx_vodctrl_out[5:3]`.

Simultaneous write and read transactions are not allowed.

Method 3—Using Individual Control Signals for Each Channel

You can optionally used Method 3 to individually reconfigure the PMA controls of each transceiver channel.

When you disable the **Use the same control signal for all channels** option, the PMA control ports for the write transaction are also separate for each channel. For example, if you have two channels, `tx_vodctrl` is 6 bits wide (`tx_vodctrl[2:0]` corresponds to channel 1 and `tx_vodctrl[5:3]` corresponds to channel 2).

The width of the PMA control ports for a read transaction are always separate for each channel (the same as the PMA control ports, as explained in "Method 2—Using the Same Control Signals for All Channels" on page 5–15.)

Write Transaction

In this method, the PMA controls are written into all the channels connected to the dynamic reconfiguration controller. Therefore, to write to a specific channel:

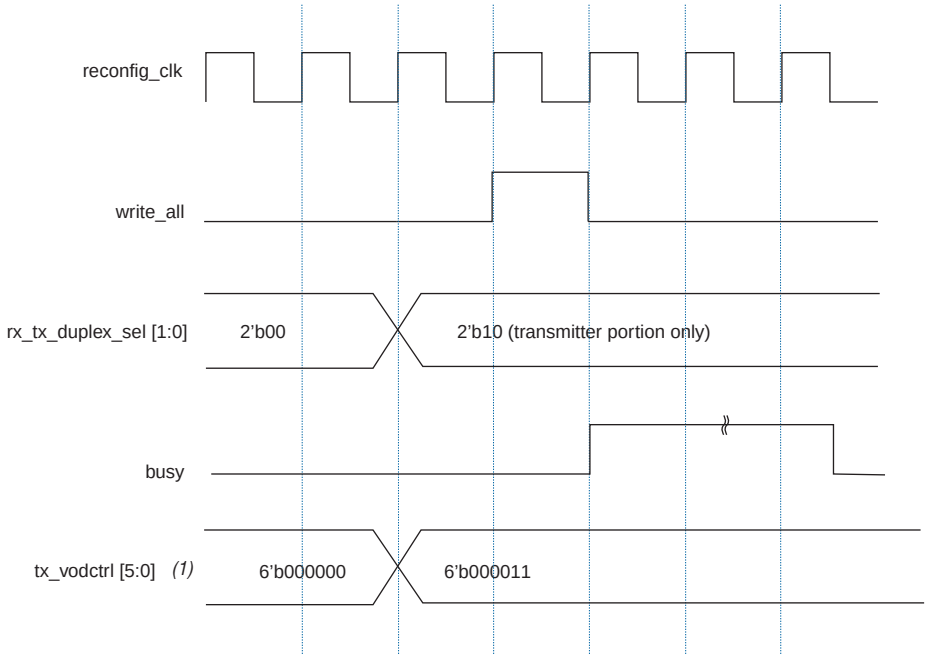
1. Retain the stored values of the other active channels using a read transaction.
2. Set the new value at the bits corresponding to the specific channel.
3. Perform a write transaction.

For example, assume that the number of channels controlled by the ALTGX_RECONFIG instance is two, `tx_vodctrl` in this case is 6 bits wide (`tx_vodctrl[2:0]` corresponds to channel 1 and `tx_vodctrl[5:3]` corresponds to channel 2). Follow these steps:

1. If you want to dynamically reconfigure the PMA controls of only channel 2 with a new value, first perform a read transaction to retrieve the existing PMA control values from `tx_vodctrl_out[5:0]`. Take `tx_vodctrl_out[2:0]` and provide this value in `tx_vodctrl[2:0]` to the write in channel 1. By doing so, channel 1 is overwritten with the same value.
2. Perform a write transaction. This ensures that the new values are written only to channel 2, while channel 1 remains unchanged.

Figure 5–9 shows a write transaction waveform using Method 3.

Figure 5–9. Method 3—Write Transaction Waveform



Note to Figure 5–9:

(1) For this example, the number of channels controlled by the dynamic reconfiguration controller (ALTGX_RECONFIG instance) is two and that the `tx_vodctrl` control port is enabled.

Simultaneous write and read transactions are not allowed.

Read Transaction

The read transaction in Method 3 is identical to that in Method 2. Refer to "Read Transaction" on page 5–16.

Transceiver Channel Reconfiguration Mode Details

Table 5–5 lists the supported configurations for the various transceiver channel reconfiguration modes available in the ALTGX_RECONFIG MegaWizard Plug-In Manager.

Table 5–5. Transceiver Channel Reconfiguration Modes and .mif Requirements

Dynamic Reconfiguration Mode	Supported Configurations		.mif Requirements
	To	From	
Channel and CMU PLL reconfiguration	All configurations of regular transceiver channels	All configurations of regular transceiver channels	✓
	Basic (PMA Direct) ×1 configuration	Basic (PMA Direct) ×1 configuration	✓
	Basic (PMA Direct) ×N configuration	Basic (PMA Direct) ×N configuration	✓
Channel reconfiguration with transmitter PLL select	Non-bonded configurations of regular transceiver channels	Non-bonded configurations of regular transceiver channels	✓
	Basic (PMA Direct) ×1 configuration	Basic (PMA Direct) ×1 configuration	✓
	Basic (PMA Direct) ×N configuration	Basic (PMA Direct) ×N configuration	✓
Central control unit reconfiguration	×4 bonded mode	×4 bonded mode	✓
	×8 bonded mode	×8 bonded mode	✓
Data rate division in transmitter	All **Transmitter only** configurations of regular transceiver channels	All **Transmitter only** configurations of regular transceiver channels	—

Memory Initialization File (.mif)

As listed in Table 5–5, all the dynamic reconfiguration modes with a check mark in the "**.mif** Requirement" column use memory initialization files to reconfigure the transceivers. These **.mifs** contain the valid settings, in the form of words, required to reconfigure the transceivers. To understand using **.mifs**, it is helpful to understand these two concepts:

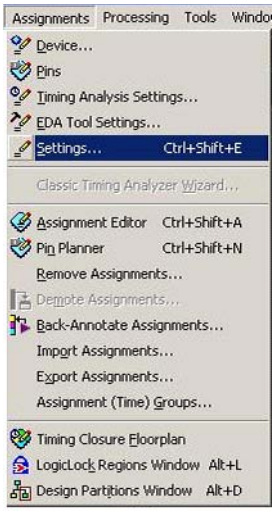
- How to generate a **.mif**?—The Quartus II software generates **.mifs** when you provide the appropriate project settings and then compiles an ALTGX instance. For more information, refer to "Quartus II Settings to Enable .mif Generation" on page 5–20.
- How is a **.mif** used between the ALTGX_RECONFIG instance and the ALTGX instance?—The Quartus II software provides a design flow called the user memory initialization file flow. For more information, refer to ".mif-Based Design Flow" on page 5–22.

Quartus II Settings to Enable .mif Generation

The **.mif** is not generated by default in a Quartus II compilation. To generate a **.mif**, you must enable the following Quartus II software settings:

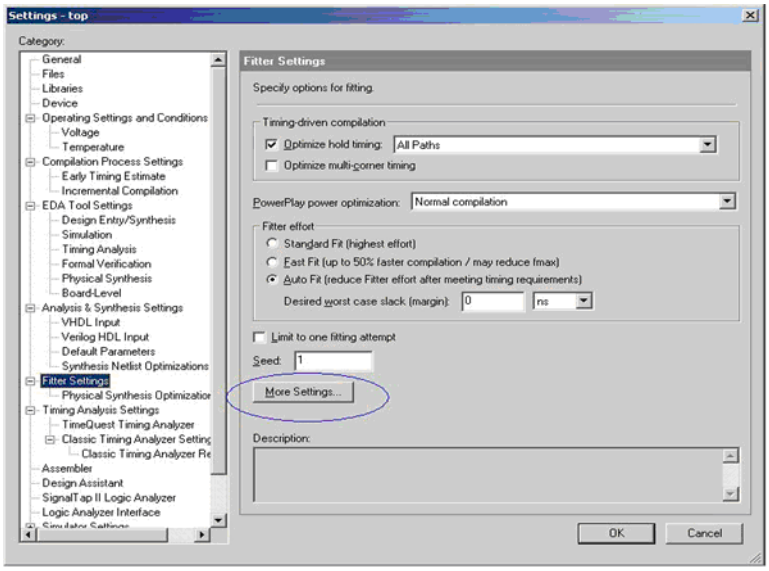
1. On the Assignments menu, select **Settings** (Figure 5–10).

Figure 5–10. Step 1 to Enable .mif Generation



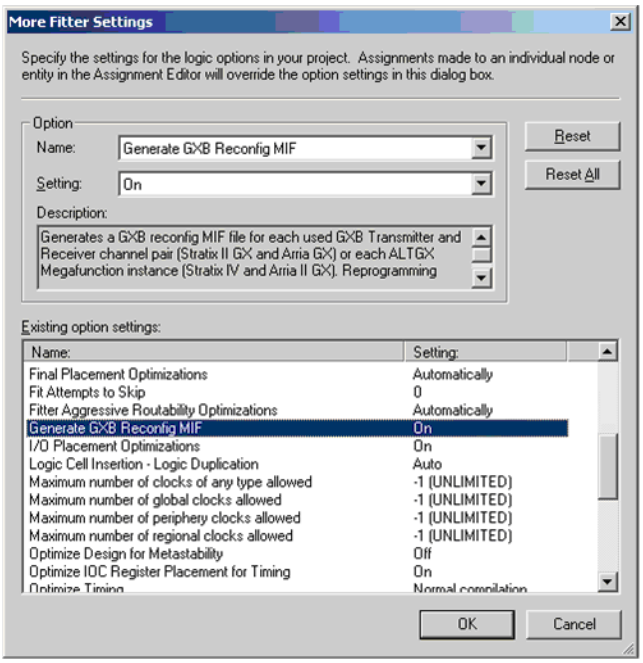
2. Select **Fitter settings**, then choose **More Settings** (Figure 5–11).

Figure 5–11. Step 2 to Enable .mif Generation



3. In the **Option** box of the **More Fitter Settings** page, set the **Generate GXB Reconfig MIF** option to **On** (Figure 5–12).

Figure 5–12. Step 3 to Enable .mif Generation



The **.mif** is generated in the Assembler stage of the compilation process. However, for any change in the design or the above settings, the Quartus II software runs through the fitter stage before starting the assembler stage.

A **.mif** is generated for every ALTGX instance defined in the top-level RTL file.

The Quartus II software creates the **.mif** under the *Project_DIR>/reconfig_mif* folder. The file name is based on the ALTGX instance name (*<instance name>***.mif**); for example, **basic_gxb.mif**. One design can have multiple **.mifs** (there is no limit) and you can use one **.mif** to reconfigure multiple channels.

To generate a **.mif**, create a top-level design and connect the clock inputs in the RTL/schematic. specifically, for the transceiver clock inputs `pll_inclk_cruclk`.

If you do not specify pins for `tx_dataout` and `rx_datain` for the transceiver channel, the Quartus II software selects a channel and generates a **.mif** for that channel. However, the **.mif** can still be used for any transceiver channel.

You can generate multiple **.mifs** in the following two ways:

Method 1:

1. Compile the design created and generate the first **.mif**.
2. Update the ALTGX instance with the alternate configuration.
3. Compile the design to get the second **.mif**.

If you have to generate **.mifs** for many configurations, Method 1 takes more time to complete.

Method 2:

1. In the top-level design, instantiate all the different configurations of the ALTGX instantiation for which the **.mif** is required.
2. Connect the appropriate clock inputs of all the ALTGX instantiations.
3. Generate the **.mif**. The **.mifs** are generated for all the ALTGX configurations.

This method requires special attention when generating the **.mif**. Refer to the following:

- The different ALTGX instantiations must have the appropriate **logical reference clock index** option values.
- The clock inputs for each instance must be connected to the appropriate clock source.
- When you generate the **.mif**, use the proper naming convention for the files so you know the configuration supported by the **.mif**.

.mif-Based Design Flow

The **.mif**-based design flow involves writing the contents of the **.mif** to the transceiver channel or CMU PLL.

To reconfigure the transceiver channel or CMU PLL, you must configure the required settings for the transceiver channel or CMU PLL in the ALTGX MegaWizard Plug-In Manager and compile the ALTGX instance. The dynamic reconfiguration controller requires that you write these configured settings through the **.mif** into the transceiver channel or CMU PLL (using the `write_all` and `reconfig_data[15:0]` signals). The maximum possible size of the **.mif** is 59 words. Each word contains legal register settings of the transceiver channel stored in 16 bits. `reconfig_address_out[5:0]` provides the address (location) of the 16-bit word in the **.mif**.

Table 5–6 lists the **.mif** size depending on the ALTGX configuration.

Table 5–6. .mif Size for the ALTGX Configuration

ALTGX Configuration	.mif Size in Words *(1)*
Duplex (**Receiver and Transmitter**) + Central control unit	60
Duplex (**Receiver and Transmitter**)	55
Receiver only	37
Transmitter only	19

Note to Table 5–6:

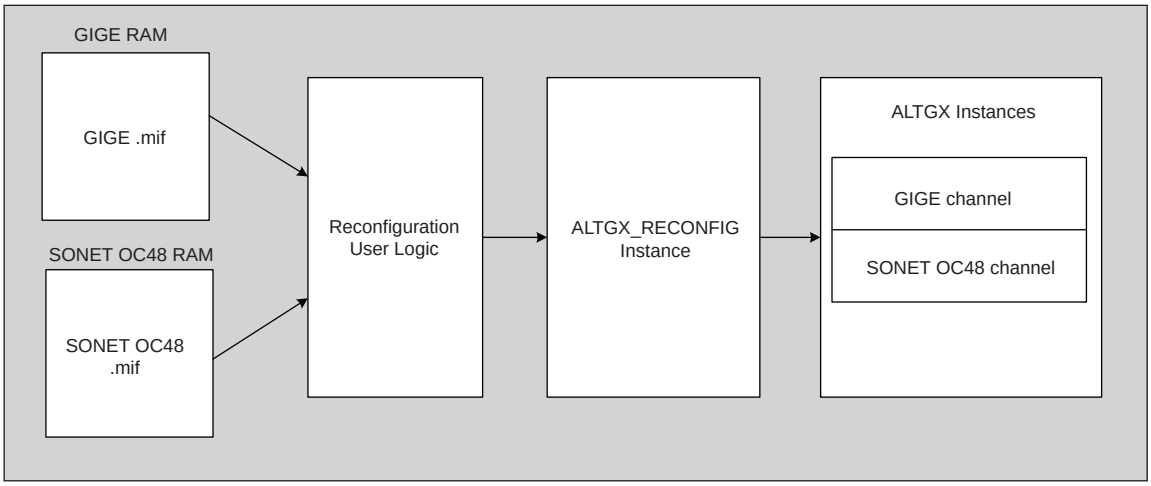
(1) Each word in the **.mif** is 16 bits wide.

You can store these **.mifs** in the on-chip or off-chip memory.

Applying a .mif in the User Design

Store the **.mif** in on-chip or off-chip memory and connect it to the dynamic reconfiguration controller, as shown in Figure 5–13.

Figure 5–13. .mif Instantiation in the User Design



When applying a **.mif** in the user design, be sure to:

- Use the RAM: 1-PORT megafunction to instantiate a memory block.
- Choose the size of the memory block based on the size of the **.mif** generated.
- Instantiate the **.mif** in the memory block.

Whenever a **.mif** is applied to a channel, the PMA controls for that channel is set to the default settings chosen in the ALTGX instance used for **.mif** generation.

Reduced .mif Reconfiguration

This mode is available only for the **.mif**-based transceiver channel reconfiguration modes.

This is an optional feature that allows faster reconfiguration and faster simulation time. For example, if you intend to make minor changes to the transceiver channel, this might involve a change of only a few words in the **.mif**.

Here is an example of changing only the termination setting:

- Assume that the only word difference is word address 32.
- Instead of loading the entire **.mif**, you can use **altgx_diffmifgen.exe** to generate a new **.mif**. This new **.mif** only has the modified words.
- The new **.mif** is 22 bits wide, compared with the 16 bits wide in the regular **.mif**. There are 6 bits of address in addition to 16 bits of data.

  ```
  <addr 6 bits> <data 16 bits>
  ```

- Enable the **Use 'reconfig_address' to input address from the MIF in reduced MIF reconfiguration** option in the **Channel and TX PLL Reconfiguration** screen of the ALTGX_RECONFIG MegaWizard Plug-In Manager.
- Use the `reconfig_data[15:0]` port to connect the 16 bits of data from the new **.mif**.
- Use the `reconfig_address[5:0]` port to connect the 6 bits of address from the new **.mif**.

Using altgx_diffmifgen.exe

Browse to the project directory where you have the Quartus II software installed. For example, **altgx_diffmifgen.exe** is available in the following path:

\altera\91\quartus\bin

The syntax for using this **.exe** is as follows:

\altera\91\quartus\bin\altgx_diffmifgen.exe <a.mif> <b.mif>

That is executed in the project directory with the **.mifs**. The **altgx_diffmifgen.exe** requires two or more ALTGX **.mifs**.

Channel and CMU PLL Reconfiguration Mode Details

Use this dynamic reconfiguration mode to reconfigure a transceiver channel to a different functional mode and data rate. To reconfigure a channel successfully, select the appropriate options in the ALTGX MegaWizard Plug-In Manager (described in the following sections) and generate a **.mif**. Connect the ALTGX_RECONFIG instance to the ALTGX instance. The dynamic reconfiguration controller reconfigures the transceiver channel by writing the **.mif** contents into the channel.

☞ Channel and CMU PLL reconfiguration mode only affects the channel involved in the reconfiguration (the transceiver channel specified by the `logical_channel_address` port), without affecting the remaining transceiver channels controlled by the dynamic reconfiguration controller.

☞ You cannot reconfigure the ATX PLLs in Stratix IV transceivers.

Channel Reconfiguration Classifications

Table 5–7 lists the classification for channel and CMU PLL reconfiguration mode.

Table 5–7. Channel Reconfiguration Classifications

Data Rate Reconfiguration	Functional Mode Reconfiguration
■ By reconfiguring the CMU PLL connected to the transceiver channel. ■ By selecting the alternate CMU PLL in the transceiver block to supply clocks to the transceiver channel. ■ Every transmitter channel has one local clock divider. Similarly, every receiver channel has one local clock divider. You can reconfigure the data rate of a transceiver channel by reconfiguring these local clock dividers to 1, 2, or 4. When you reconfigure these local clock dividers, ensure that the functional mode of the transceiver channel supports the reconfigured data rate.	■ Use this feature to reconfigure the existing functional mode of the transceiver channel to a totally different functional mode. ■ There is no limit to the number of functional modes you can reconfigure the transceiver channel to if the various clocks involved support the transition. For more information about core clocks, refer to "Clocking/Interface Options" on page 5–30.

In addition to the categories mentioned, you can also choose to reconfigure both the data rate and functional mode of a transceiver channel.

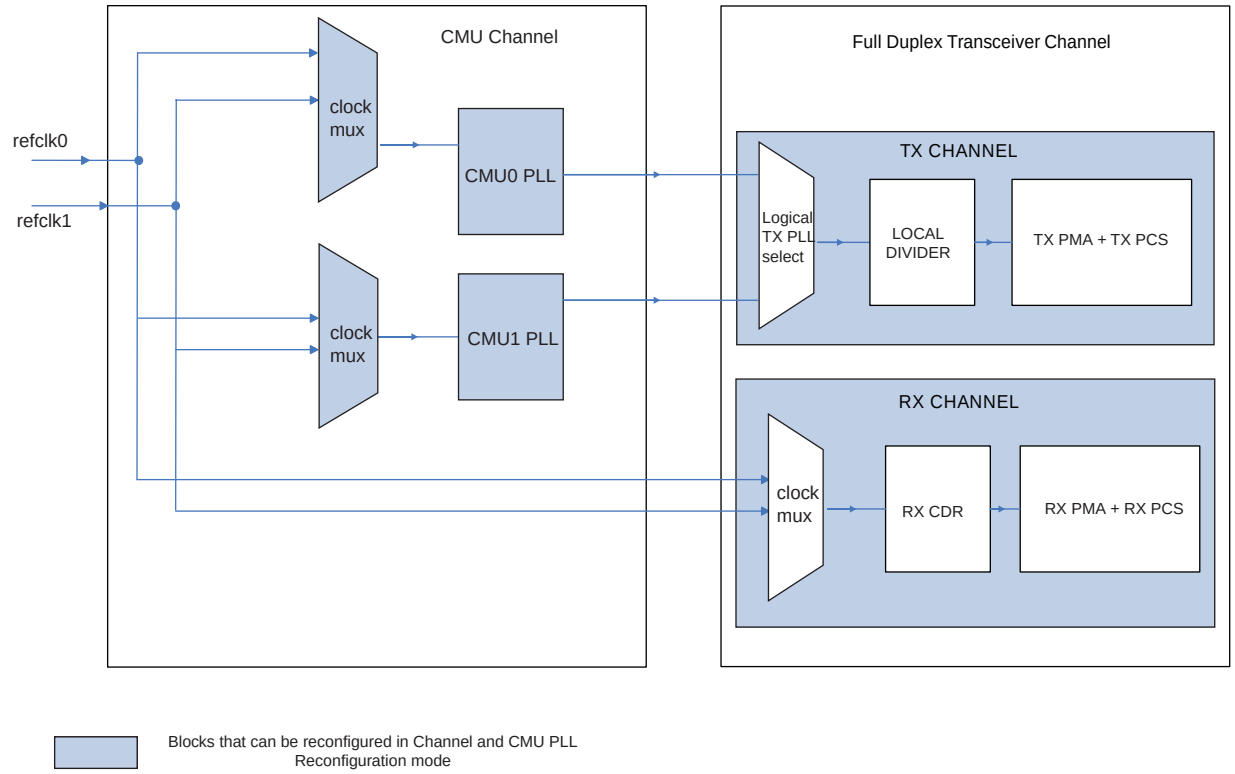
For the following sections, assume that the transceiver channel has the **Receiver and Transmitter** configuration in the ALTGX MegaWizard Plug-In Manager, unless specified as **Transmitter only** or **Receiver only**.

Blocks Reconfigured in Channel and CMU PLL Reconfiguration Mode

The blocks that are reconfigured by this dynamic reconfiguration mode are the PCS and PMA blocks of a transceiver channel, the local divider settings of the transmitter and receiver channel, and the CMU PLL.

Figure 5–14 shows the functional blocks that you can dynamically reconfigure using the channel and CMU PLL reconfiguration mode.

Figure 5–14. Channel and CMU PLL Reconfiguration in a Transceiver Block



Channel reconfiguration from either a **Transmitter only** configuration to a **Receiver only** configuration or vice versa is not allowed.

ALTGX MegaWizard Plug-In Manager Setup for Channel and CMU PLL Reconfiguration Mode

To reconfigure the transceiver channel and CMU PLL, set up the ALTGX MegaWizard Plug-In Manager using the following steps:

1. Select the **Channel and Transmitter PLL reconfiguration** option in the **Modes** screen under the **Reconfiguration Settings** tab.
2. If you want to reconfigure the data rate of the transceiver channel by reconfiguring the CMU PLL, provide the new data rate you want the CMU PLL to run at in the **General** screen.
3. If you want to reconfigure the data rate of the transceiver channel by switching to the alternate CMU PLL within the same transceiver block, select the **Use alternate CMU transmitter PLL** option in the **Modes** screen. For more information, refer to the "Using the Alternate CMU Transmitter PLL" on page 5–27.
4. Provide the number of input reference clocks available for the CMU PLL in the **How many input clocks?** option of the corresponding PLL screen. The maximum number of input reference clocks allowed is 10. For more information, refer to "Guidelines for Specifying the Input Reference Clocks" on page 5–61.

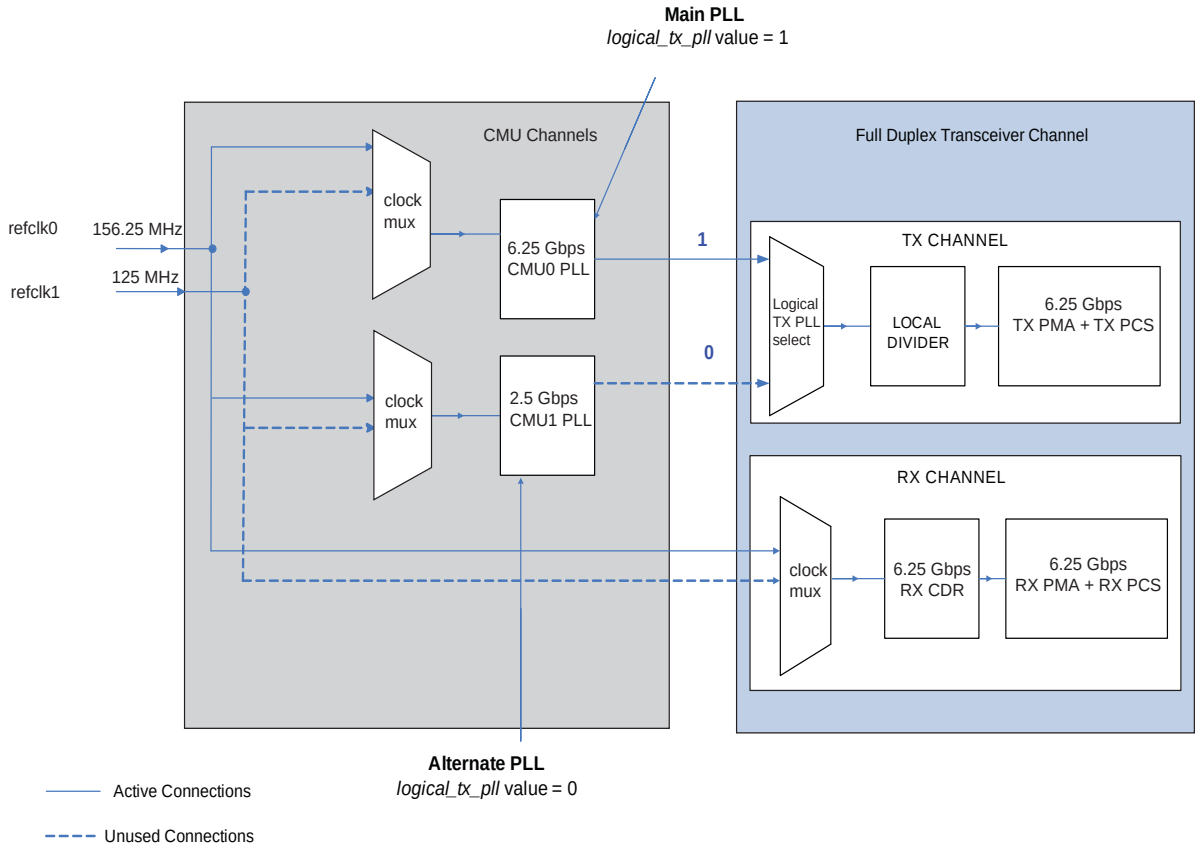
5. Provide the starting channel number in the **Modes** screen. For more information, refer to "Logical Channel Addressing" on page 5–5.
6. Provide the logical reference index of the CMU PLL in the **What is the PLL logical reference index?** option in the corresponding PLL screen. For more information, refer to "Selecting the Logical Reference Index of the CMU PLL" on page 5–29.
7. Provide the identification of the input reference clock used by the CMU PLL in the corresponding PLL screens.
8. Set up the **Clocking/Interface** options. For more information, refer to "Clocking/Interface Options" on page 5–30.
9. Set up the **Channel Interface** options. For more information, refer to "FPGA Fabric-Transceiver Channel Interface Selection" on page 5–36.

Using the Alternate CMU Transmitter PLL

To reconfigure the CMU PLL during run time, you need the flexibility to select one of the two CMU PLLs of a transceiver block.

Consider that the transceiver channel is listening to `CMU0` PLL and that you want to the reconfigure `CMU0` PLL, as shown in Figure 5–15.

Figure 5–15. Reconfiguring the CMU0 PLL



You can select `CMU0` PLL by specifying its identity in the ALTGX MegaWizard Plug-In Manager. This identification is referred to as the `logical tx pll` value. This value provides a logical identification to `CMU0` PLL and associates it with a transceiver channel without requiring the knowledge of its physical location.

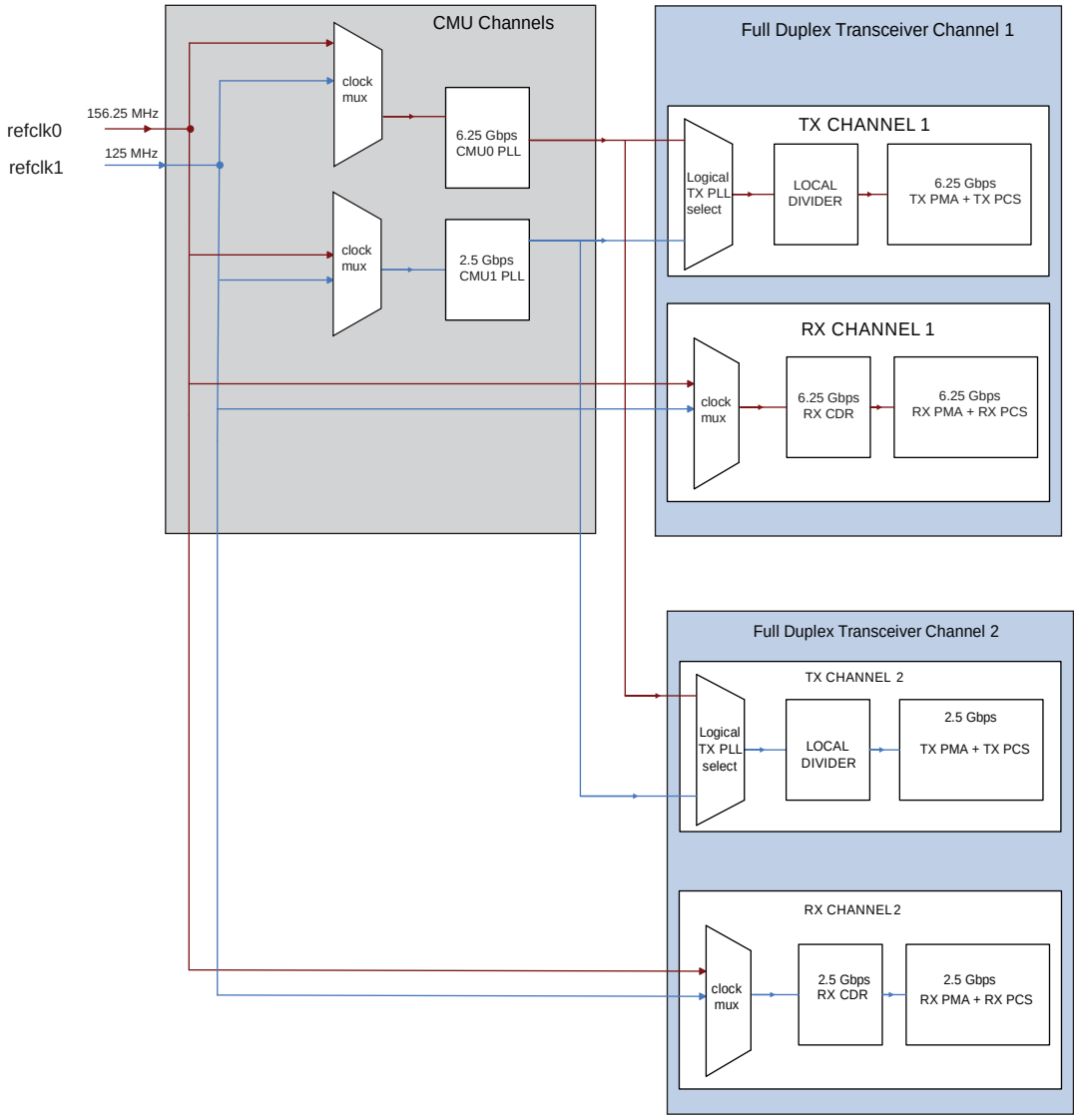
In the ALTGX MegaWizard Plug-In Manager, the transmitter PLL configuration set in the **General** screen is called the main PLL. When you provide the alternate PLL with a `logical tx pll` value (for example, 0), the main PLL automatically takes the complement value 1. The `logical tx pll` value for the main PLL is stored along with the other transceiver channel information in the generated **.mif**.

The main PLL corresponds to the CMU PLL configuration set in the **General** screen of the ALTGX MegaWizard Plug-In Manager. The alternate PLL corresponds to the CMU PLL configuration set in the **Alt PLL** screen.

Selecting the Logical Reference Index of the CMU PLL

In Figure 5–16, transceiver channel 1 listens to `CMU0` PLL of the transceiver block. Similarly, transceiver channel 2 listens to `CMU1` PLL of the transceiver block.

Figure 5–16. Logical Reference Index of CMU PLLs in a Transceiver Block



To direct the ALTGX_RECONFIG instance to dynamically reconfigure `CMU0` PLL, specify its logical reference index (the identity of a transmitter PLL). Similarly, to direct the ALTGX_RECONFIG instance to dynamically reconfigure `CMU1` PLL instead, provide the logical reference index of `CMU1` PLL. The allowed values for the logical reference index of the CMU PLLs within a transceiver block are 0 or 1. Similarly, the transmitter PLLs outside the transceiver block can also be assigned a logical reference index value. For more information, refer to "Selecting the PLL Logical Reference Index for Additional PLLs" on page 5–53.

The logical reference index of `CMU0` PLL within a transceiver block is always the complement of the logical reference index of `CMU1` PLL within the same transceiver block.

This logical reference index value is stored as `logical tx pll`, along with the other transceiver channel settings in the **.mif**.

Clocking/Interface Options

The following describes the **Clocking/Interface** options. The core clocking setup describes the transceiver core clocks that are the write and read clocks of the Transmit Phase Compensation FIFO and the Receive Phase Compensation FIFO, respectively. Core clocking is classified as transmitter core clocking and receiver core clocking.

Transmitter core clocking refers to the clock that is used to write the parallel data from the FPGA fabric into the Transmit Phase Compensation FIFO. You can use one of the following clocks to write into the Transmit Phase Compensation FIFO:

- `tx_coreclk`—You can use a clock of the same frequency as `tx_clkout` from the FPGA fabric to provide the write clock to the Transmit Phase Compensation FIFO. If you use `tx_coreclk`, it overrides the `tx_clkout` options in the ALTGX MegaWizard Plug-In Manager.
- `tx_clkout`—The Quartus II software automatically routes `tx_clkout` to the FPGA fabric and back into the Transmit Phase Compensation FIFO.

The **Clocking/Interface** screen is not available for PMA-Only channels.

Option 1: Share a Single Transmitter Core Clock Between Transmitters

- Enable this option if you want `tx_clkout` of the first channel (channel 0) of the transceiver block to provide the write clock to the Transmitter Phase Compensation FIFOs of the remaining channels in the transceiver block.
- This option is typically enabled when all the channels of a transceiver block are of the same functional mode and data rate and are reconfigured to the identical functional mode and data rate.

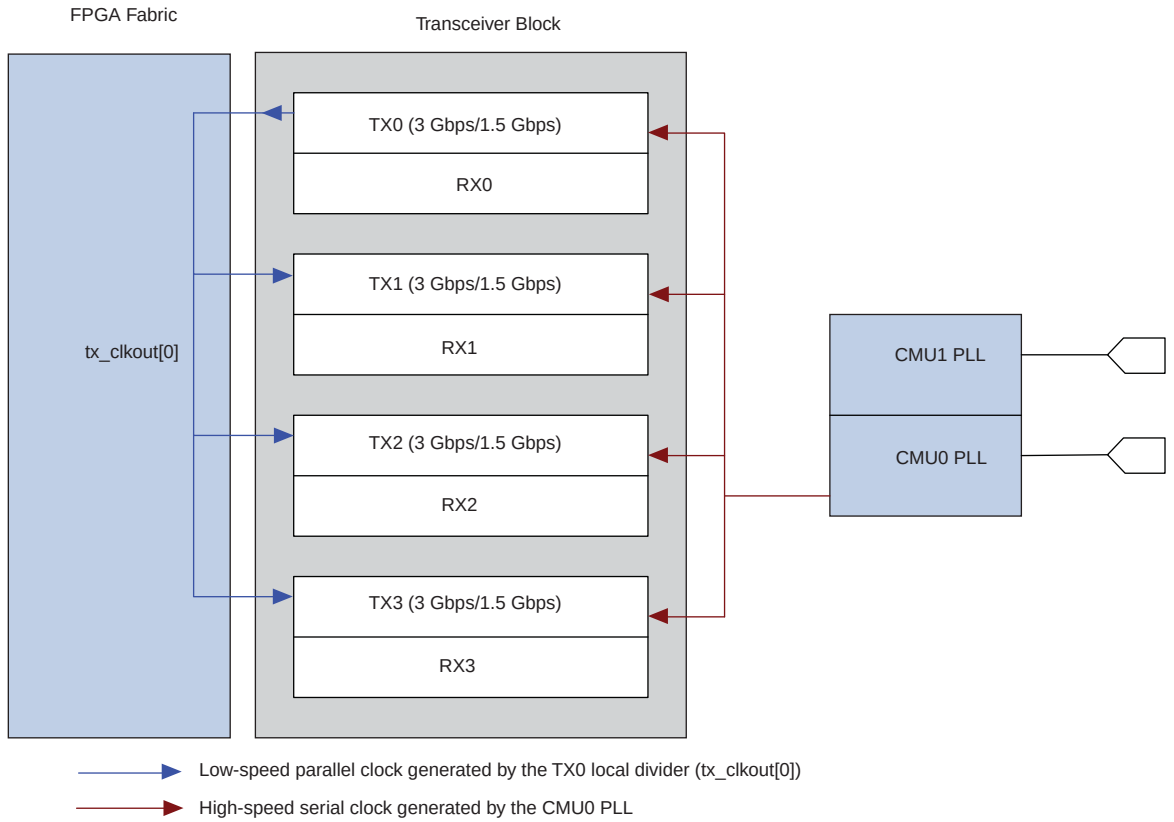
Consider the following scenario:

- Four regular transceiver channels configured at 3 Gbps and in the same functional mode.
- Channel and CMU PLL reconfiguration mode is enabled in the ALTGX_RECONFIG MegaWizard Plug-In Manager.
- You want to reconfigure all four regular transceiver channels to 1.5 Gbps and vice versa.

Option 1 is applicable in this scenario because it saves clock resources.

Figure 5–17 shows the sharing of channel 0's `tx_clkout` between all four regular channels of a transceiver block.

Figure 5–17. Option 1 for Transmitter Core Clocking (Channel and CMU PLL Reconfiguration Mode)



Option 2: Use the Respective Channel Transmitter Core Clocks

- Enable this option if you want the individual transmitter channel `tx_clkout` signals to provide the write clock to their respective Transmit Phase Compensation FIFOs.
- This option is typically enabled when each transceiver channel is reconfigured to a different functional mode using channel reconfiguration.

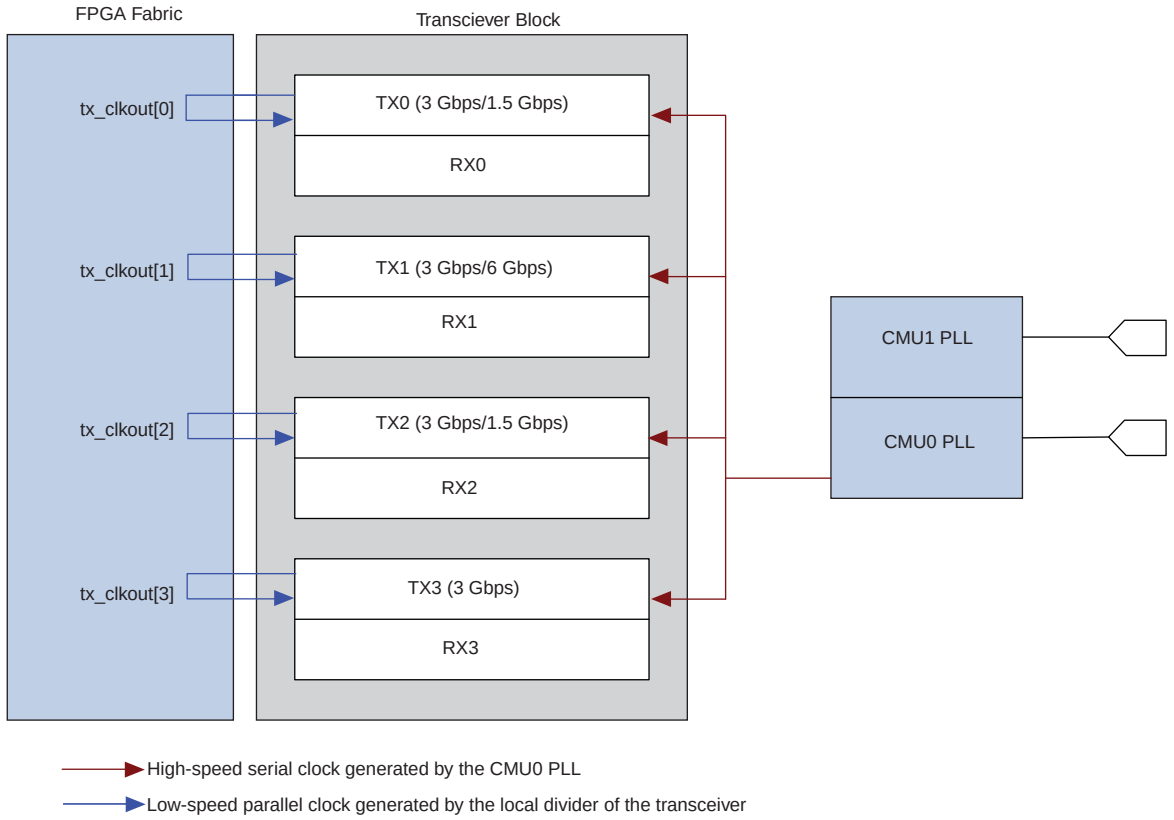
Consider the following scenario:

- Four regular transceiver channels configured at 3 Gbps and different functional modes.
- Channel and CMU PLL reconfiguration mode is enabled in the ALTGX_RECONFIG MegaWizard Plug-In Manager.
- You want to reconfigure each of the four regular transceiver channels to different data rates and different functional modes.

Option 2 is applicable in this scenario because the design requires all four regular transceiver channels to be reconfigured to different data rates and functional modes. Each channel can be reconfigured to a different functional mode using the channel and CMU PLL reconfiguration mode.

Figure 5–18 shows how each transmitter channel's `tx_clkout` signal provides a clock to the Transmit Phase Compensation FIFOs of the respective transceiver channels.

Figure 5–18. Option 2 for Transmitter Core Clocking (Channel and CMU PLL Reconfiguration Mode)



Receiver core clocking refers to the clock that is used to read the parallel data from the Receiver Phase Compensation FIFO into the FPGA fabric. You can use one of the following clocks to read from the Receive Phase Compensation FIFO:

- `rx_coreclk`—You can use a clock of the same frequency as `rx_clkout` from the FPGA fabric to provide the read clock to the Receive Phase Compensation FIFO. If you use `rx_coreclk`, it overrides the `rx_clkout` options in the ALTGX MegaWizard Plug-In Manager.
- `rx_clkout`—The Quartus II software automatically routes `rx_clkout` to the FPGA fabric and back into the Receive Phase Compensation FIFO.

The **Clocking/Interface** screen is not available for PMA-Only channels.

Option 1: Share a Single Transmitter Core Clock Between Receivers

- Enable this option if you want `tx_clkout` of the first channel (channel 0) of the transceiver block to provide the read clock to the Receive Phase Compensation FIFOs of the remaining receiver channels in the transceiver block.
- This option is typically enabled when all the channels of a transceiver block are in a Basic or Protocol configuration with rate matching enabled and are reconfigured to another Basic or Protocol configuration with rate matching enabled.

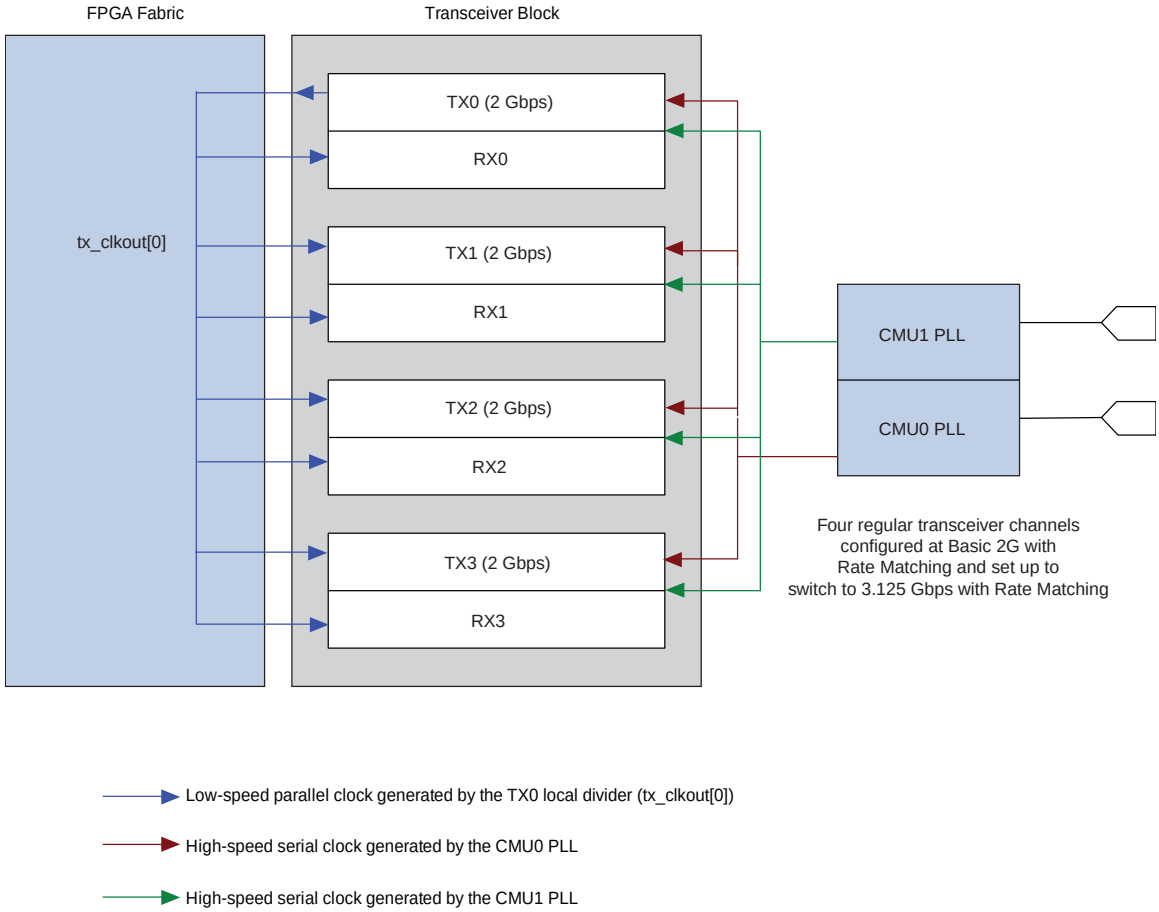
Consider the following scenario:

- Four regular transceiver channels configured to Basic 2 Gbps functional mode with rate matching enabled.
- Channel and CMU PLL reconfiguration mode is enabled in the ALTGX_RECONFIG MegaWizard Plug-In Manager.
- You want to reconfigure all four regular transceiver channels to 3.125 Gbps configuration with rate matching enabled.

Option 1 is applicable in this scenario.

Figure 5–19 shows the sharing of channel 0's `tx_clkout` between all four channels of a transceiver block.

Figure 5–19. Option 1 for Receiver Core Clocking (Channel and CMU PLL Reconfiguration Mode)



Option 2: Use the Respective Channel Transmitter Core Clocks

- Enable this option if you want the individual transmitter channel's `tx_clkout` signal to provide the read clock to its respective Receive Phase Compensation FIFO.
- This option is typically enabled when all the transceiver channels have rate matching enabled with different data rates and are reconfigured to another Basic or Protocol functional mode with rate matching enabled.

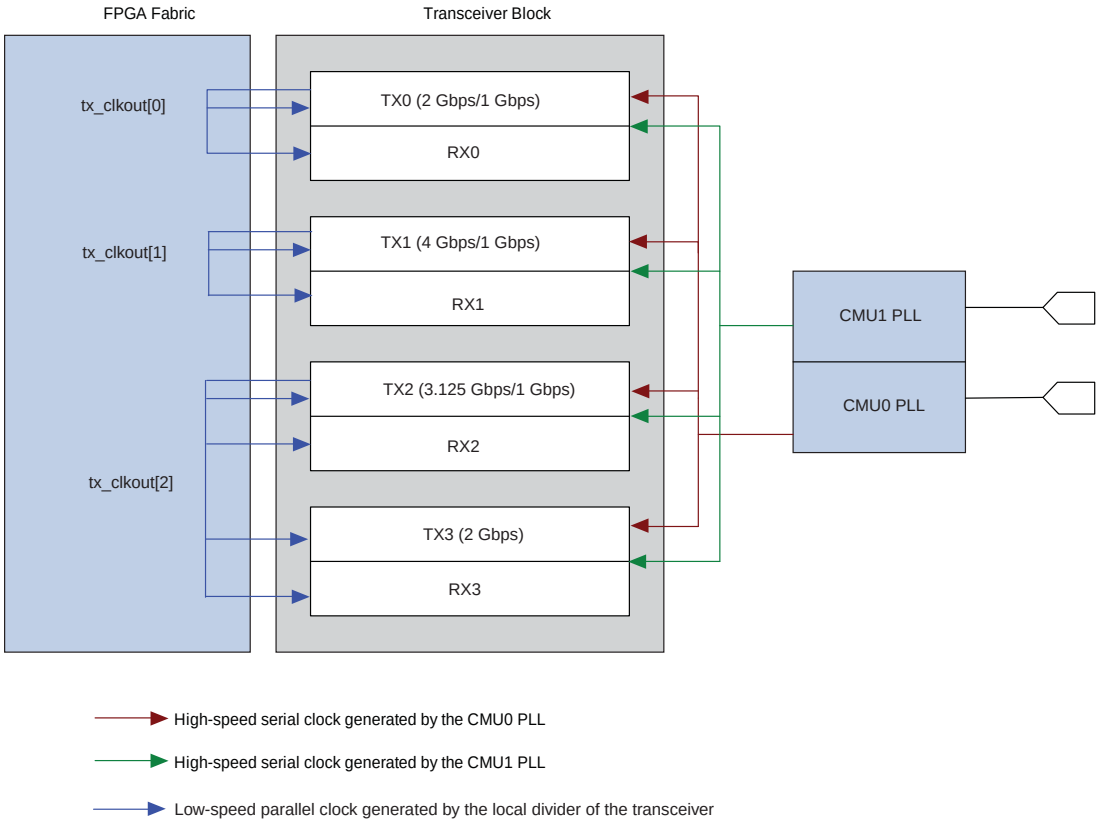
Consider the following scenario:

- TX0/RX0: You want to dynamically reconfigure Basic 1 Gbps configuration with rate matching enabled to Basic 2 Gbps configuration with rate matching enabled.
- TX1/RX1: You want to dynamically reconfigure Basic 4 Gbps configuration with rate matching enabled to Basic 1 Gbps configuration with rate matching enabled.
- TX2/RX2 and TX3/RX3: You want to dynamically reconfigure Basic 3.125 Gbps configuration with rate matching enabled to 1 Gbps configuration with rate matching and vice versa.
- Channel and CMU PLL reconfiguration mode is enabled in the ALTGX_RECONFIG MegaWizard Plug-In Manager.

Option 2 is applicable because the design requires the individual transceiver channels to be reconfigured with different data rates to another Basic or Protocol functional mode with rate matching. Therefore, each channel can be reconfigured to another Basic or Protocol functional mode with rate matching enabled and a different data rate.

Figure 5–20 shows the respective `tx_clkout` of each channel clocking the respective channels of a transceiver block.

Figure 5–20. Option 2 for Receiver Core Clocking (Channel and CMU PLL Reconfiguration Mode)



Option 3: Use the Respective Channel Receiver Core Clocks

- Enable this option if you want the individual channel's `rx_clkout` signal to provide the read clock to its respective Receive Phase Compensation FIFO.
- This option is typically enabled when the channel is reconfigured from a Basic or Protocol configuration with or without rate matching to another Basic or Protocol configuration with or without rate matching.

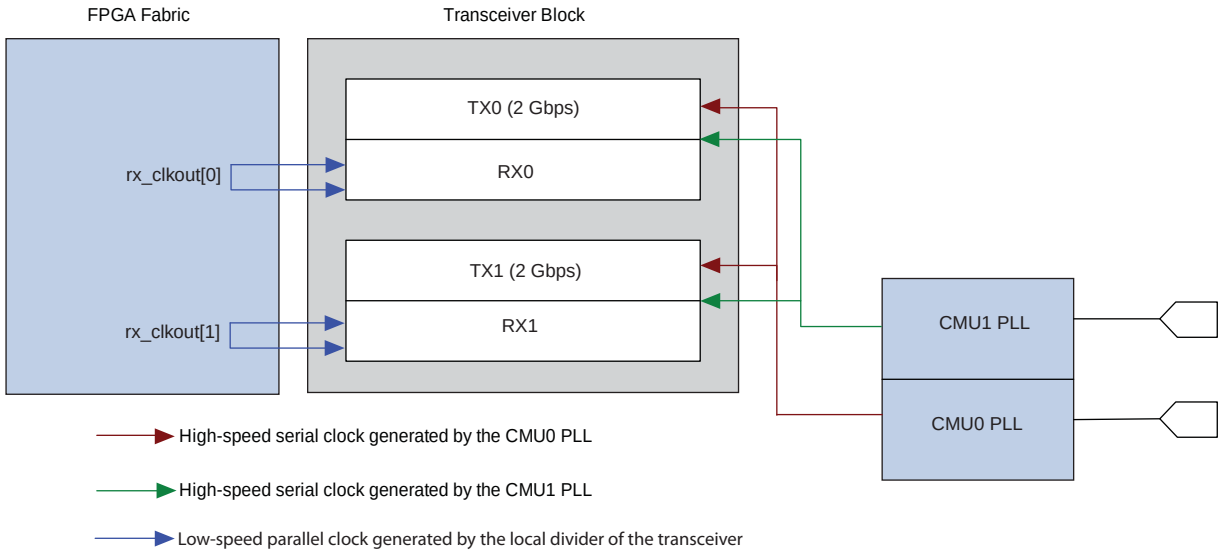
Consider the following scenario:

- TX1/RX1: GIGE configuration to SONET/SDH OC48 configuration.
- TX2/RX2: Basic 2.5 Gbps configuration with rate matching disabled to Basic 1.244 Gbps configuration with rate matching disabled.
- Channel and CMU PLL reconfiguration mode is enabled in the ALTGX_RECONFIG MegaWizard Plug-In Manager

Option 3 is applicable in this scenario.

Figure 5–21 shows the respective `rx_clkout` of each channel clocking the respective receiver channels of a transceiver block.

Figure 5–21. Option 3 for Receiver Core Clocking (Channel and CMU PLL Reconfiguration Mode)



FPGA Fabric-Transceiver Channel Interface Selection

This section describes the ALTGX MegaWizard Plug-In Manager settings related to the FPGA fabric-Transceiver channel interface data width when you select and activate channel and CMU PLL reconfiguration mode. You need to set up the FPGA fabric-Transceiver channel interface data width when functional mode reconfiguration involves:

- changes in the FPGA fabric-Transceiver channel data width

  OR

- enabling and disabling the static PCS blocks of the transceiver channel

You can set up the FPGA fabric-Transceiver channel interface data width by enabling the **Channel Interface** option in the **Modes** screen.

Enable the **Channel Interface** option if the reconfiguration if the reconfigured channel has:

- changed FPGA fabric-Transceiver channel interface data width

  OR

- changed input control signals and output status signals

There are two signals available when you enable the **Channel Interface** option:

- `tx_datainfull`—The width of this input signal depends on the number of channels you set up in the **General** screen. It is 44 bits wide per channel. This signal is available only for **Transmitter only** and **Receiver and Transmitter** configurations. This port replaces the existing `tx_datain` port.
- `rx_dataoutfull`—The width of this output signal depends on the number of channels you set up in the **General** screen. It is 64 bits wide per channel. This signal is available only for **Receiver only** and **Receiver and Transmitter** configurations. This port replaces the existing `rx_dataout` port.

In addition to these two ports, you can select the necessary control and status signals for the reconfigured channel in the **Clocking/Interface** screen.

For more information about control and status signals, refer to the "Transceiver Port Lists" section in the *Stratix IV GX Transceiver Architecture* chapter.

These control and status signals are not applicable in Basic (PMA Direct) functional mode. Table 5–8 lists the signals not available when you enable the **Channel Interface** option.

Table 5–8. Control and Status Signals Not Applicable in Basic (PMA Direct) Mode with the Channel Interface Option Enabled

FPGA Fabric-Receiver Interface	FPGA Fabric-Transmitter Interface
`rx_dataout`	`tx_datain`
`rx_syncstatus`	`tx_ctrlenable`
`rx_patterndetect`	`tx_forcedisp`
`rx_a1a2sizeout`	`tx_dispval`
`rx_ctrldetect`	
`rx_errdetect`	
`rx_disperr`	

The Quartus II software has legal checks for the connectivity of `tx_datainfull` and `rx_dataoutfull` and the various control and status signals you enable in the **Clocking/Interface** screen.

For example, the Quartus II software allows you to select and connect the `pipestatus` and `powerdn` signals. It assumes that you are planning to switch to and from PCI Express (PIPE) functional mode. Table 5–9 describes the `tx_datainfull[43:0]` FPGA fabric-Transceiver channel interface signals.

Table 5–9. tx_datainfull[43:0] FPGA Fabric-Transceiver Channel Interface Signal Descriptions (Part 1 of 2) *(Note 1)*

FPGA Fabric-Transceiver Channel Interface Description	Transmit Signal Description (Based on Stratix IV GX Supported FPGA Fabric-Transceiver Channel Interface Widths)
8-bit FPGA fabric-Transceiver Channel Interface	`tx_datainfull[7:0]`: 8-bit data (`tx_datain`)
	The following signals are used only in 8B/10B modes:
	`tx_datainfull[8]`: Control bit (`tx_ctrlenable`)
	`tx_datainfull[9]` Transmitter force disparity Compliance (PCI Express [PIPE]) (`tx_forcedisp`) in all modes except PCI Express (PIPE). For PCI Express (PIPE) mode, (`tx_forcedispcompliance`) is used.
	`tx_datainfull[10]`: Forced disparity value (`tx_dispval`)
10-bit FPGA fabric-Transceiver Channel Interface	`tx_datainfull[9:0]`: 10-bit data (`tx_datain`)
16-bit FPGA fabric-Transceiver Channel Interface with PCS-PMA set to 16/20 bits	Two 8-bit Data (`tx_datain`) `tx_datainfull[7:0]` - `tx_datain` (LSByte) and `tx_datainfull[18:11]` - `tx_datain` (MSByte)
	The following signals are used only in 8B/10B modes:
	`tx_datainfull[8]` - `tx_ctrlenable` (LSB) and `tx_datainfull[19]` - `tx_ctrlenable` (MSB)
	Force Disparity Enable `tx_datainfull[9]` - `tx_forcedisp` (LSB) and `tx_datainfull[20]` - `tx_forcedisp` (MSB)
	Force Disparity Value `tx_datainfull[10]` - `tx_dispval` (LSB) and `tx_datainfull[21]` - `tx_dispval` (MSB)
16-bit FPGA fabric-Transceiver Channel Interface with PCS-PMA set to 8/10 bits	Two 8-bit Data (`tx_datain`) `tx_datainfull[7:0]` - `tx_datain` (LSByte) and `tx_datainfull[29:22]` - `tx_datain` (MSByte)
	The following signals are used only in 8B/10B modes:
	Two Control Bits (`tx_ctrlenable`) `tx_datainfull[8]` - `tx_ctrlenable` (LSB) and `tx_datainfull[30]` - `tx_ctrlenable` (MSB)
	Force Disparity Enable For non-PIPE: `tx_datainfull[9]` - `tx_forcedisp` (LSB) and `tx_datainfull[31]` - `tx_forcedisp` (MSB) For PCI Express (PIPE): `tx_datainfull[9]` - `tx_forcedispcompliance` (LSB) and `tx_datainfull[31]` - `tx_forcedispcompliance` (MSB)
	Force Disparity Value `tx_datainfull[10]` - `tx_dispval` (LSB) and `tx_datainfull[32]` - `tx_dispval` (MSB)

Table 5–9. tx_datainfull[43:0] FPGA Fabric-Transceiver Channel Interface Signal Descriptions (Part 2 of 2) *(Note 1)*

FPGA Fabric-Transceiver Channel Interface Description	Transmit Signal Description (Based on Stratix IV GX Supported FPGA Fabric-Transceiver Channel Interface Widths)
20-bit FPGA fabric-Transceiver Channel Interface with PCS-PMA set to 20 bits	Two 10-bit Data (`tx_datain`) `tx_datainfull[9:0]` - `tx_datain` (LSByte) and `tx_datainfull[20:11]` - `tx_datain` (MSByte)
20-bit FPGA fabric-Transceiver Channel Interface with PCS-PMA set to 10 bits	Two 10-bit Data (`tx_datain`) `tx_datainfull[9:0]` - `tx_datain` (LSByte) and `tx_datainfull[31:22]` - `tx_datain` (MSByte)

32-bit FPGA fabric-Transceiver Channel Interface with PCS-PMA set to 16/20 bits	Four 8-bit Data (`tx_datain`)
	`tx_datainfull[7:0]`- `tx_datain` (LSByte) and
	`tx_datainfull[18:11]`
	`tx_datainfull[29:22]`
	`tx_datainfull[40:33]` - `tx_datain` (MSByte)

	The following signals are used only in 8B/10B modes:

	Four Control Bits (`tx_ctrlenable`)
	`tx_datainfull[8]` - `tx_ctrlenable` (LSB) and
	`tx_datainfull[19]`
	`tx_datainfull[30]`
	`tx_datainfull[41]`- `tx_ctrlenable` (MSB)

	Force Disparity Enable (`tx_forcedisp`)
	`tx_datainfull`[9]- `tx_forcedisp` (LSB) and
	`tx_datainfull[20]`
	`tx_datainfull[31]`
	`tx_datainfull[42]`- `tx_forcedisp` (MSB)

	Force Disparity Value (`tx_dispval`)
	`tx_datainfull[10]`- `tx_dispval` (LSB) and
	`tx_datainfull[21]`
	`tx_datainfull[32]`
	`tx_datainfull[43]`- `tx_dispval` (MSB)

40-bit FPGA fabric-Transceiver Channel Interface with PCS-PMA set to 20 bits	Four 10-bit Data (`tx_datain`)
	`tx_datainfull[9:0]` - `tx_datain` (LSByte) and
	`tx_datainfull[20:11]`
	`tx_datainfull[31:22]`
	`tx_datainfull[42:33]`- `tx_datain` (MSByte)

Note to Table 5–9:

(1) For all transceiver-related ports, refer to the "Transceiver Port Lists" section in the *Stratix IV Transceiver Architecture* chapter.

Table 5–10 describes the `tx_dataoutfull[63:0]` FPGA fabric-Transceiver channel interface signals.

Table 5–10. rx_dataoutfull[63:0] FPGA Fabric-Transceiver Channel Interface Signal Descriptions (Part 1 of 7)

FPGA Fabric-Transceiver Channel Interface Description	Receive Signal Description (Based on Stratix IV GX Supported FPGA Fabric-Transceiver Channel Interface Widths)
8-bit FPGA fabric-Transceiver Channel Interface	**The following signals are used in 8-bit 8B/10B modes:**
	`rx_dataoutfull[7:0]`: 8-bit decoded data (`rx_dataout`)
	`rx_dataoutfull[8]`: Control bit (`rx_ctrldetect`)
	`rx_dataoutfull[9]`: Code violation status signal (`rx_errdetect`)
	`rx_dataoutfull`[10]: `rx_syncstatus`
	`rx_dataoutfull[11]`: Disparity error status signal (`rx_disperr`)
	`rx_dataoutfull[12]`: Pattern detect status signal (`rx_patterndetect`)
	`rx_dataoutfull[13]`: Rate Match FIFO deletion status indicator (`rx_rmfifodatadeleted`) in non-PCI Express (PIPE)/PCIe modes.
	`rx_dataoutfull[14]`: Rate Match FIFO insertion status indicator (`rx_rmfifodatainserted`) in non-PCI Express (PIPE)/PCIe modes.
	`rx_dataoutfull[14:13]`: PCI Express (PIPE)/PCIe mode (`rx_pipestatus`)
	`rx_dataoutfull[15]`: 8B/10B running disparity indicator (`rx_runningdisp`)
	The following signals are used in 8-bit SONET/SDH mode:
	`rx_dataoutfull[7:0]`: 8-bit un-encoded data (`rx_dataout`)
	`rx_dataoutfull[8]`: `rx_a1a2sizeout`
	`rx_dataoutfull[10]`: `rx_syncstatus`
	`rx_dataoutfull[11]`: Reserved
	`rx_dataoutfull[12]`: `rx_patterndetect`
10-bit FPGA fabric-Transceiver Channel Interface	`rx_dataoutfull[9:0]`: 10-bit un-encoded data (`rx_dataout`)
	`rx_dataoutfull[10]`: `rx_syncstatus`
	`rx_dataoutfull[11]`: 8B/10B disparity error indicator (`rx_disperr`)
	`rx_dataoutfull[12]`: `rx_patterndetect`
	`rx_dataoutfull[13]`: Rate Match FIFO deletion status indicator (`rx_rmfifodatadeleted`) in non-PCI Express (PIPE)/PCIe modes
	`rx_dataoutfull[14]`: Rate Match FIFO insertion status indicator (rx_rmfifodatainserted) in non-PCI Express (PIPE)/PCIe modes
	`rx_dataoutfull[15]`: 8B/10B running disparity indicator (`rx_runningdisp`)

Table 5–10. rx_dataoutfull[63:0] FPGA Fabric-Transceiver Channel Interface Signal Descriptions (Part 2 of 7)

FPGA Fabric-Transceiver Channel Interface Description	Receive Signal Description (Based on Stratix IV GX Supported FPGA Fabric-Transceiver Channel Interface Widths)
16-bit FPGA fabric-Transceiver Channel Interface with PCS-PMA set to 16/20 bits	Two 8-bit unencoded Data (`rx_dataout`) `rx_dataoutfull[7:0]` - `rx_dataout` (LSByte) and `rx_dataoutfull[23:16]`- `rx_dataout` (MSByte)
	The following signals are used in 16-bit 8B/10B modes:
	Two Control Bits `rx_dataoutfull[8]` - `rx_ctrldetect` (LSB) and `rx_dataoutfull[24]`- `rx_ctrldetect` (MSB)
	Two Receiver Error Detect Bits `rx_dataoutfull[9]` - `rx_errdetect` (LSB) and `rx_dataoutfull[25]`- `rx_errdetect` (MSB)
	Two Receiver Sync Status Bits `rx_dataoutfull [10]` - `rx_syncstatus` (LSB) and `rx_dataoutfull[26] -` `rx_syncstatus` (MSB)
	Two Receiver Disparity Error Bits `rx_dataoutfull [11]` - `rx_disperr` (LSB) and `rx_dataoutfull[27]` - `rx_disperr` (MSB)
	Two Receiver Pattern Detect Bits `rx_dataoutfull[12]` - `rx_patterndetect` (LSB) and `rx_dataoutfull[28]`- `rx_patterndetect` (MSB)
	`rx_dataoutfull[13]` and `rx_dataoutfull`[45]: Rate Match FIFO deletion status indicator (`rx_rmfifodatadeleted`) in non-PCI Express (PIPE)/PCIe modes
	`rx_dataoutfull[14]` and `rx_dataoutfull`[46]: Rate Match FIFO insertion status indicator (`rx_rmfifodatainserted`) in non-PCI Express (PIPE)/PCIe modes
	Two 2-bit PCI Express (PIPE) Status Bits `rx_dataoutfull[14:13]` - `rx_pipestatus` (LSB) and `rx_dataoutfull[30:29]` - `rx_pipestatus` (MSB)
	`rx_dataoutfull[15]` and `rx_dataoutfull[47]`: 8B/10B running disparity indicator (`rx_runningdisp`)

Table 5–10. rx_dataoutfull[63:0] FPGA Fabric-Transceiver Channel Interface Signal Descriptions (Part 3 of 7)

FPGA Fabric-Transceiver Channel Interface Description	Receive Signal Description (Based on Stratix IV GX Supported FPGA Fabric-Transceiver Channel Interface Widths)
16-bit FPGA fabric-Transceiver Channel Interface with PCS-PMA set to 8/10 bits	Two 8-bit Data `rx_dataoutfull[7:0]` - `rx_dataout` (LSByte) and `rx_dataoutfull[39:32]` - `rx_dataout` (MSByte)
	The following signals are used in 16-bit 8B/10B mode:
	Two Control Bits `rx_dataoutfull[8]` - `rx_ctrldetect` (LSB) and `rx_dataoutfull[40]` - `rx_ctrldetect` (MSB)
	Two Receiver Error Detect Bits `rx_dataoutfull[9]` - `rx_errdetect` (LSB) and `rx_dataoutfull[41]`- `rx_errdetect` (MSB)
	Two Receiver Sync Status Bits `rx_dataoutfull[10]` - `rx_syncstatus` (LSB) and `rx_dataoutfull[42]`- `rx_syncstatus` (MSB)
	Two Receiver Disparity Error Bits `rx_dataoutfull[11]` - `rx_disperr` (LSB) and `rx_dataoutfull[43]` - `rx_disperr` (MSB)
	Two Receiver Pattern Detect Bits `rx_dataoutfull[12]` - `rx_patterndetect` (LSB) and `rx_dataoutfull[44]` - `rx_patterndetect` (MSB)
	`rx_dataoutfull[13]` and `rx_dataoutfull[45]`: Rate Match FIFO deletion status indicator (`rx_rmfifodatadeleted`) in non-PCI Express (PIPE)/PCIe modes
	`rx_dataoutfull[14]` and `rx_dataoutfull[46]`: Rate Match FIFO insertion status indicator (`rx_rmfifodatainserted`) in non-PCI Express (PIPE)/PCIe modes
	Two 2-bit PCI Express (PIPE) Status Bits `rx_dataoutfull[14:13]` - `rx_pipestatus` (LSB) and `rx_dataoutfull[46:45]`- `rx_pipestatus` (MSB)
	`rx_dataoutfull`[15] and `rx_dataoutfull[47]`: 8B/10B running disparity indicator (`rx_runningdisp`)

Table 5–10. rx_dataoutfull[63:0] FPGA Fabric-Transceiver Channel Interface Signal Descriptions (Part 4 of 7)

FPGA Fabric-Transceiver Channel Interface Description	Receive Signal Description (Based on Stratix IV GX Supported FPGA Fabric-Transceiver Channel Interface Widths)
16-bit FPGA fabric-Transceiver Channel Interface with PCS-PMA set to 8/10 bits (continued)	**The following signals are used in 16-bit SONET/SDH mode:**
	Two 8-bit Data `rx_dataoutfull[7:0]` - `rx_dataout` (LSByte) and `rx_dataoutfull[39:32]` - `rx_dataout` (MSByte)
	Two Receiver Alignment Pattern Length Bits `rx_dataoutfull[8]` - `rx_a1a2sizeout` (LSB) and `rx_dataoutfull[40]`- `rx_a1a2sizeout` (MSB)
	Two Receiver Sync Status Bits `rx_dataoutfull[10]` - `rx_syncstatus` (LSB) and `rx_dataoutfull[42]` - `rx_syncstatus` (MSB)
	Two Receiver Pattern Detect Bits `rx_dataoutfull[12]` - `rx_patterndetect` (LSB) and `rx_dataoutfull[44]` - `rx_patterndetect` (MSB)
20-bit FPGA fabric-Transceiver Channel Interface with PCS-PMA set to 20 bits	Two 10-bit Data (`rx_dataout`) `rx_dataoutfull[9:0]` - `rx_dataout` (LSByte) and `rx_dataoutfull[25:16]` - `rx_dataout` (MSByte)
	wo Receiver Sync Status Bits `rx_dataoutfull[10]` - `rx_syncstatus` (LSB) and `rx_dataoutfull[26]` - `rx_syncstatus` (MSB)
	`rx_dataoutfull[11]` and `rx_dataoutfull`[27]: 8B/10B disparity error indicator (`rx_disperr`)
	Two Receiver Pattern Detect Bits `rx_dataoutfull[12]` - `rx_patterndetect` (LSB) and `rx_dataoutfull[28]` - `rx_patterndetect` (MSB)
	`rx_dataoutfull[13]` and `rx_dataoutfull[29]`: Rate Match FIFO deletion status indicator (`rx_rmfifodatadeleted`) in non-PCI Express (PIPE)/PCIe modes
	`rx_dataoutfull[14]` and `rx_dataoutfull[30]`: Rate Match FIFO insertion status indicator (`rx_rmfifodatainserted`) in non-PCI Express (PIPE)/PCIe modes
	`rx_dataoutfull[15]` and `rx_dataoutfull[31]`: 8B/10B running disparity indicator (`rx_runningdisp`)

Table 5–10. rx_dataoutfull[63:0] FPGA Fabric-Transceiver Channel Interface Signal Descriptions (Part 5 of 7)

FPGA Fabric-Transceiver Channel Interface Description	Receive Signal Description (Based on Stratix IV GX Supported FPGA Fabric-Transceiver Channel Interface Widths)
20-bit FPGA fabric-Transceiver Channel Interface with PCS-PMA set to 10 bits	Two 10-bit Data `rx_dataoutfull[9:0]` - `rx_dataout` (LSByte) and `rx_dataoutfull[41:32]` - `rx_dataout` (MSByte)
	Two Receiver Sync Status Bits `rx_dataoutfull[10]` - `rx_syncstatus` (LSB) and `rx_dataoutfull[42]` - `rx_syncstatus` (MSB)
	`rx_dataoutfull[11]` and `rx_dataoutfull[43]`: 8B/10B disparity error indicator (`rx_disperr`)
	Two Receiver Pattern Detect Bits `rx_dataoutfull[12]` - `rx_patterndetect` (LSB) and `rx_dataoutfull[44]` - `rx_patterndetect` (MSB)
	`rx_dataoutfull[13]` and `rx_dataoutfull[45]`: Rate Match FIFO deletion status indicator (`rx_rmfifodatadeleted`) in non-PCI Express (PIPE)/PCIe modes
	`rx_dataoutfull[14]` and `rx_dataoutfull[46]`: Rate Match FIFO insertion status indicator (`rx_rmfifodatainserted`) in non-PCI Express (PIPE)/PCIe modes
	`rx_dataoutfull[15]` and `rx_dataoutfull[47]`: 8B/10B running disparity indicator (`rx_runningdisp`)

Table 5–10. rx_dataoutfull[63:0] FPGA Fabric-Transceiver Channel Interface Signal Descriptions (Part 6 of 7)

FPGA Fabric-Transceiver Channel Interface Description	Receive Signal Description (Based on Stratix IV GX Supported FPGA Fabric-Transceiver Channel Interface Widths)

32-bit mode	Four 8-bit un-encoded Data (`rx_dataout`)
	`rx_dataoutfull[7:0]`- `rx_dataout` (LSByte)
	`rx_dataoutfull[23:16]`
	`rx_dataoutfull[39:32]`
	`rx_dataoutfull[55:48]` - `rx_dataout` (MSByte)

	The following signals are used in 32-bit 8B/10B mode:

	Four Control Data Bits (`rx_dataout`)
	`rx_dataoutfull[8]` - `rx_ctrldetect` (LSB)
	`rx_dataoutfull[24]`
	`rx_dataoutfull[40]`
	`rx_dataoutfull[56]` - `rx_ctrldetect` (MSB)

	Four Receiver Error Detect Bits
	`rx_dataoutfull[9]`- `rx_errdetect` (LSB)
	`rx_dataoutfull[25]`
	`rx_dataoutfull[41]`
	`rx_dataoutfull[57]` - `rx_errdetect` (MSB)

	Four Receiver Pattern Detect Bits
	`rx_dataoutfull`[10]- `rx_syncstatus` (LSB) and
	`rx_dataoutfull[26]`
	`rx_dataoutfull[42]`
	`rx_dataoutfull[58]` - `rx_syncstatus` (MSB)

	Four Receiver Disparity Error Bits
	`rx_dataoutfull[11]`- `rx_disperr` (LSB)
	`rx_dataoutfull[27]`
	`rx_dataoutfull[43]`
	`rx_dataoutfull[59]` - `rx_disperr` (MSB)

	Four Receiver Pattern Detect Bits
	`rx_dataoutfull[12]`- `rx_patterndetect` (LSB)
	`rx_dataoutfull[28]`
	`rx_dataoutfull[44]`
	`rx_dataoutfull[60]` - `rx_patterndetect` (MSB)

	`rx_dataoutfull[13]`, `rx_dataoutfull[29]`, `rx_dataoutfull[45]` and `rx_dataoutfull[61]`: Rate Match FIFO deletion status indicator (`rx_rmfifodatadeleted`) in non-PCI Express (PIPE)/PCIe modes
	`rx_dataoutfull[14]`, `rx_dataoutfull[30]`, `rx_dataoutfull[46]`, and `rx_dataoutfull[62]`: Rate Match FIFO insertion status indicator (`rx_rmfifodatainserted`) in non-PCI Express (PIPE)/PCIe modes

Table 5–10. rx_dataoutfull[63:0] FPGA Fabric-Transceiver Channel Interface Signal Descriptions (Part 7 of 7)

FPGA Fabric-Transceiver Channel Interface Description	Receive Signal Description (Based on Stratix IV GX Supported FPGA Fabric-Transceiver Channel Interface Widths)
32-bit mode (continued)	`rx_dataoutfull[15]`, `rx_dataoutfull[31]`, `rx_dataoutfull[47]`, and `rx_dataoutfull[63]`: 8B/10B running disparity indicator (`rx_runningdisp`)
	The following signals are used in 32-bit SONET/SDH scrambled backplane mode:

	Four Control Data Bits (`rx_dataout`)
	`rx_dataoutfull[7:0]`- `rx_dataout` (LSByte)
	`rx_dataoutfull[23:16]`
	`rx_dataoutfull[39:32]`
	`rx_dataoutfull[55:48]` - `rx_dataout` (MSByte)

	`rx_dataoutfull[8]`, `rx_dataoutfull[24]`, `rx_dataoutfull[40]`, and `rx_dataoutfull[56]`: four `rx_a1a2sizeout`

	Four Receiver Sync Status Bits
	`rx_dataoutfull[10]`- `rx_syncstatus` (LSB)
	`rx_dataoutfull[26]`
	`rx_dataoutfull[42]`
	`rx_dataoutfull[58]` - `rx_syncstatus` (MSB)

	Four Receiver Pattern Detect Bits
	`rx_dataoutfull[12]`- `rx_patterndetect` (LSB)
	`rx_dataoutfull[28]`
	`rx_dataoutfull[44]`
	`rx_dataoutfull[60]` - `rx_patterndetect` (MSB)

40-bit mode	Four 10-bit Control Data Bits (`rx_dataout`) `rx_dataoutfull[9:0]`- `rx_dataout` (LSByte) rx_dataoutfull[25:16] `rx_dataoutfull[41:32]` `rx_dataoutfull[57:48]` - `rx_dataout` (MSByte)

	Four Receiver Sync Status Bits
	`rx_dataoutfull[10]`- `rx_syncstatus` (LSB)
	`rx_dataoutfull[26]`
	`rx_dataoutfull[42]`
	`rx_dataoutfull[58]` - `rx_syncstatus` (MSB)

	Four Receiver Pattern Detect Bits
	`rx_dataoutfull[12]`- `rx_patterndetect` (LSB)
	`rx_dataoutfull[28]`
	`rx_dataoutfull[44]`
	`rx_dataoutfull[60]` - `rx_patterndetect` (MSB)

ALTGX_RECONFIG MegaWizard Plug-In Manager Setup for Channel and CMU PLL Reconfiguration Mode

To setup channel and CMU PLL reconfiguration mode in the ALTGX_RECONFIG MegaWizard Plug-In Manager, use the following steps:

1. In the **Reconfiguration settings** screen, set the **What is the number of channels controlled by the reconfig controller?** option. For more information, refer to "Total Number of Channels Option in the ALTGX_RECONFIG Instance" on page 5–10.
2. In the **Reconfiguration settings** screen, select the **Channel and TX PLL select/reconfig** option.

The following control signals are always available when you enable the **Channel and TX PLL select/reconfig** option:

- `channel_reconfig_done`
- `reconfig_address_out[5:0]`

The following ports are optional and available for selection in the **Channel and TX PLL Reconfiguration** screen:

- `reset_reconfig_address`
- `reconfig_address_en`
- `logical_tx_pll_sel` and `logical_tx_pll_sel_en`—For more information about these two ports, refer to "Guidelines for logical_tx_pll_sel and logical_tx_pll_sel_en Ports" on page 5–59.
- `rx_tx_duplex_sel[1:0]`

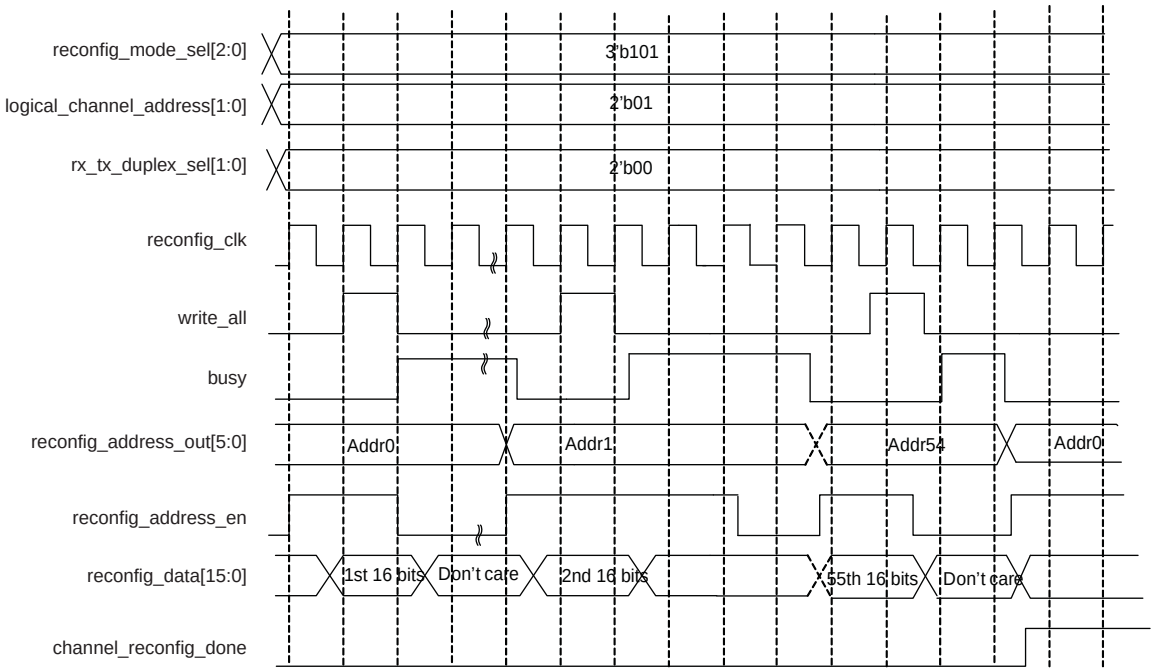
Channel and CMU PLL Reconfiguration Operation

In channel reconfiguration, only a write transaction can occur; no read transactions are allowed. In the example shown in Figure 5–22, the ALTGX_RECONFIG controls two channels. Therefore, the `logical_channel_address` signal is 2 bits wide. Also, the transceiver channel is configured in Basic mode with the **Receiver and Transmitter** configuration.

You can optionally choose to trigger `write_all` once by selecting the continuous write operation in the ALTGX_RECONFIG MegaWizard Plug-In Manager. The Quartus II software then continuously writes all the words required for reconfiguration.

Figure 5–22 shows a **.mif** write transaction when using channel and CMU PLL reconfiguration mode.

Figure 5–22. .mif Write Transaction in Channel and CMU PLL Reconfiguration Mode



Notes to Figure 5–22:

(1) The `logical_channel_address` port is set to **2'b01** to reconfigure the second transceiver channel.

(2) The `rx_tx_duplex_sel[1:0]` port is set to **2'b00** to match the **Receiver and Transmitter** configuration of the specified transceiver channel.

For guidelines regarding re-using **.mifs**, specifying input reference clocks, or using `logical_tx_pll_sel` ports, refer to "Special Guidelines" on page 5–57.

For more information about reset, refer to the "Reset Sequence when Using Dynamic Reconfiguration with the Channel and TX PLL select/reconfig Option" section in the *Reset Control and Power Down* chapter.

Channel Reconfiguration with Transmitter PLL Select Mode Details

You can reconfigure the data rate of a transceiver channel by switching between a maximum of four transmitter PLLs.

You can select between the following transmitter PLLs

- CMU PLLs present in a transceiver block
- CMU PLLs present in other transceiver blocks
- ATX PLLs outside the transceiver block

You can use the channel reconfiguration with transmitter PLL select mode along with the CMU PLL reconfiguration mode, only if it is a CMU PLL and not an ATX PLL. You can first reconfigure the second CMU PLL to the desired data rate using CMU PLL reconfiguration mode. Then use channel reconfiguration with transmitter PLL select mode to reconfigure the transceiver channel to listen to the second CMU PLL.

For more information about supported configurations, refer to "Transceiver Channel Reconfiguration Mode Details" on page 5–19 and "Memory Initialization File (.mif)" on page 5–19.

Channel reconfiguration with transmitter PLL select mode is not applicable to regular transceiver channels in bonded mode configurations (×4 and ×8).

For guidelines regarding re-using **.mifs**, specifying input reference clocks, or using `logical_tx_pll_sel` ports, refer to "Special Guidelines" on page 5–57.

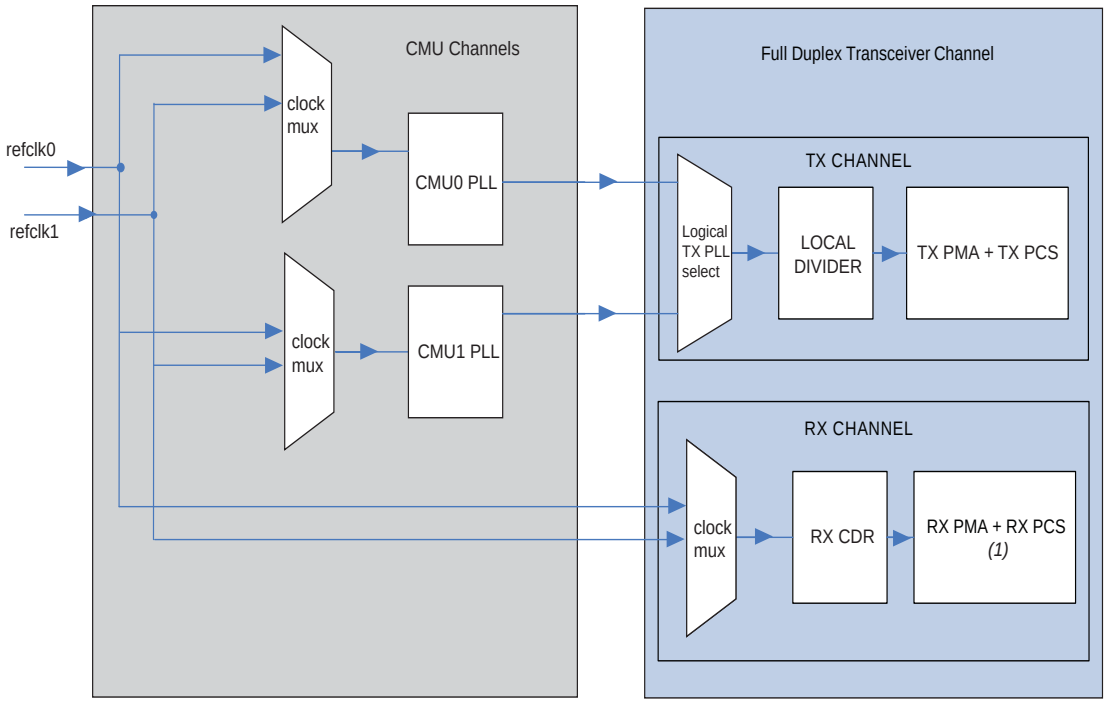
For more information about reset, refer to the "Reset Sequence when Using Dynamic Reconfiguration with the Channel and TX PLL select/reconfig Option" section in the *Reset Control and Power Down* chapter.

Blocks Reconfigured in the Channel Reconfiguration with Transmitter PLL Select Mode

The blocks reconfigured in this mode have two types of multiplexers. When you switch between the CMU PLLs within the same transceiver block, the multiplexer that is reconfigured is within the transceiver block. It is located in the transmitter channel path.

Figure 5–23 shows the multiplexers that you can dynamically reconfigure using channel reconfiguration with transmitter PLL select mode.

Figure 5–23. Channel Reconfiguration with Transmitter PLL Select in a Transceiver Block

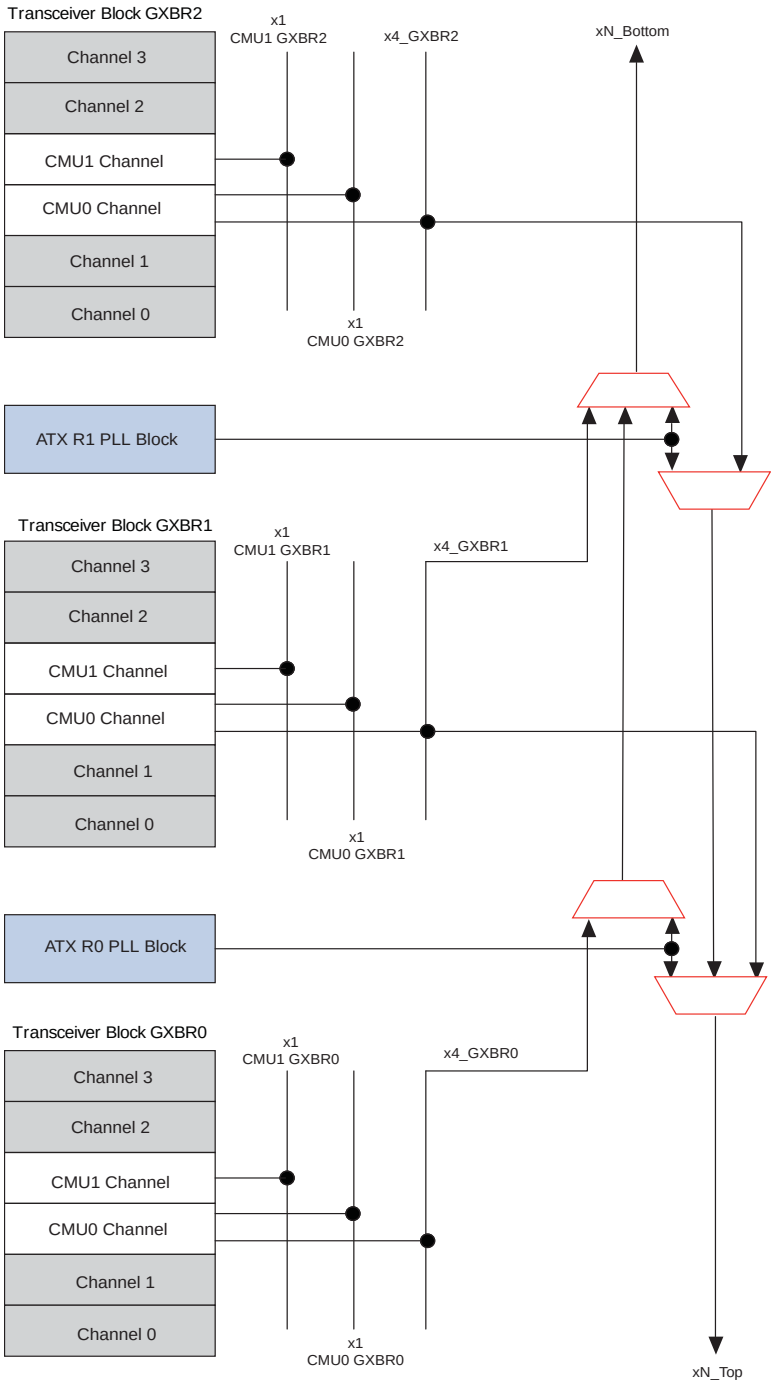


Note to Figure 5–23:

(1) Depending on the mode you select, PCS may or may not be present.

Figure 5–24 shows the multiplexers that are reconfigured when you switch to an additional PLL that is outside the transceiver block.

Figure 5–24. Multiplexers that are Reconfigured When you Switch to an Additional PLL



ALTGX MegaWizard Plug-In Manager Setup for Channel Reconfiguration with Transmitter PLL Select Mode

Follow steps 1, 2, 4, 7, 8, and 9 described in "ALTGX MegaWizard Plug-In Manager Setup for Channel and CMU PLL Reconfiguration Mode" on page 5–26. In addition to these steps, you must also set up the following:

Multi-PLL Settings

The **Use additional CMU/ATX Transmitter PLLs from outside the transceiver block** option allows you select a maximum of four transmitter PLLs.

Specify the number of additional PLLs required for the ALTGX instance in the **Modes** screen. Based on this number, the Quartus II software opens up the corresponding PLL screens (for example, **PLL 1** and **PLL 2**).

The PLL set up in the **General** screen is always the Main PLL and the settings are available in the **Main PLL** screen. Similarly, the PLL settings for the additional PLLs are available in the corresponding **PLL1** screen, **PLL 2** screen, and so on.

Additional PLLs also include the CMU PLLs within the same transceiver block.

For example, you can select the ATX PLL as the main PLL, and three additional PLLs as follows:

- PLL 1—`CMU0` PLL of the same transceiver block
- PLL 2—`CMU1` PLL of the same transceiver block
- PLL 3—`CMU0` PLL/`CMU1` PLL of another transceiver block.

The Quartus II software differentiates between the CMU PLLs of the same transceiver block and the transmitter PLLs outside the transceiver block based on the **Use central clock divider to drive the transmitter channels using ×4/×N lines** option.

If this option is enabled, the transmitter PLL is outside the transceiver block. Similarly, if this option is disabled, the transmitter PLL is one of the CMU PLLs within the same transceiver block.

Logical Channel Addressing When Using Additional PLLs

The logical channel addressing of the transceiver channel is the same as described in "Logical Channel Addressing" on page 5–5 so long as you are ONLY using the CMU PLLs within the same transceiver block.

In the case of additional PLLs (when transmitter PLLs are outside the transceiver block), the additional PLLs also have their own logical channel address. This affects the starting channel number of the following ALTGX instances connected to the dynamic reconfiguration controller, if any. Therefore, you must take into account the logical channel address of transmitter PLLs outside the transceiver block when setting the **Total number of channels controlled by the reconfig controller** option in the ALTGX_RECONFIG instance.

When you select the **Use central clock divider to drive the transmitter channels using ×4/×N lines** option for an additional PLL, you can see its logical channel address value at the bottom of the corresponding PLL screen.

Selecting the PLL Logical Reference Index for Additional PLLs

The PLL logical reference index of additional PLLs outside the transceiver block can only be 2 or 3. When you

- enable the **Use central clock divider to drive the transmitter channels using ×4/×N lines** option for an additional PLL, you can only select between 2 or 3 as the PLL logical reference index.
- disable the **Use central clock divider to drive the transmitter channels using ×4/×N lines** option for an additional PLL, the additional PLL is one of the CMU PLLs within the same transceiver block. Therefore, the PLL logical reference index is either 0 or 1.

For more information about the PLL logical reference index of CMU PLLs within the same transceiver block, refer to "Selecting the Logical Reference Index of the CMU PLL" on page 5–29.

ALTGX_RECONFIG MegaWizard Plug-In Manager Setup for Channel Reconfiguration with Transmitter PLL Select Mode

For more information, refer to the "ALTGX_RECONFIG MegaWizard Plug-In Manager Setup for Channel and CMU PLL Reconfiguration Mode" on page 5–47.

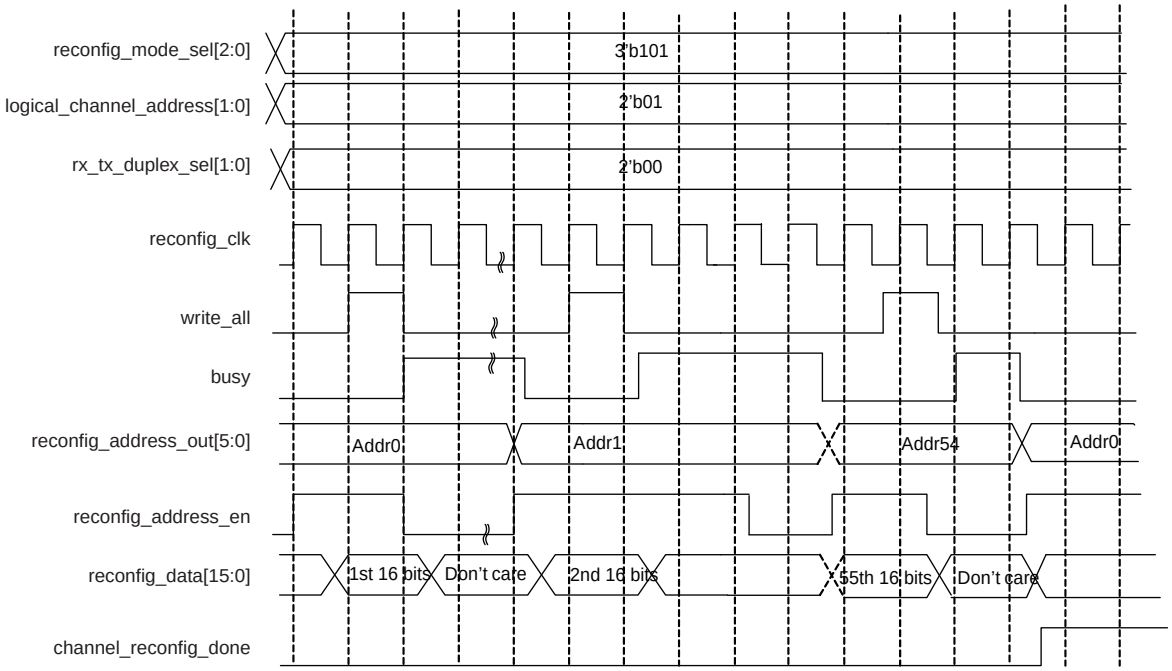
Channel Reconfiguration with Transmitter PLL Select Operation

Read transactions are not allowed in this mode.

Figure 5–25 shows a **.mif** write transaction when dynamically reconfiguring a transceiver channel. The **.mif** write transaction in channel reconfiguration with transmitter PLL select mode remains the same except the `reconfig_mode_sel[2:0]` value and the difference in the number of **.mif** words used. In this example, the transceiver channel is configured in **Receiver and Transmitter** configuration. Therefore, the **.mif** size is 8.

You can optionally choose to trigger `write_all` once by selecting the continuous write operation in the ALTGX_RECONFIG MegaWizard Plug-In Manager. The Quartus II software then continuously writes all the words required for reconfiguration.

Figure 5–25. .mif write transaction in Channel and CMU PLL Reconfiguration Mode



For guidelines regarding re-using **.mifs**, specifying input reference clocks, or using `logical_tx_pll_sel` ports, refer to "Special Guidelines" on page 5–57.

For more information about reset, refer to the "Reset Sequence when Using Dynamic Reconfiguration with the Channel and TX PLL select/reconfig Option" section in the *Reset Control and Power Down* chapter.

CMU PLL Reconfiguration Mode Details

Use this mode to reconfigure only the CMU PLL without affecting the remaining blocks of the transceiver channel. When you reconfigure the CMU PLL of a transceiver block to run at a different data rate, all the transceiver channels listening to this CMU PLL also are reconfigured to the new data rate.

You cannot dynamically reconfigure a CMU PLL into a CMU channel and vice versa.

For more information about the supported configurations in CMU PLL reconfiguration mode, refer to Table 5–5 on page 5–19.

Transmitter PLL Powerdown

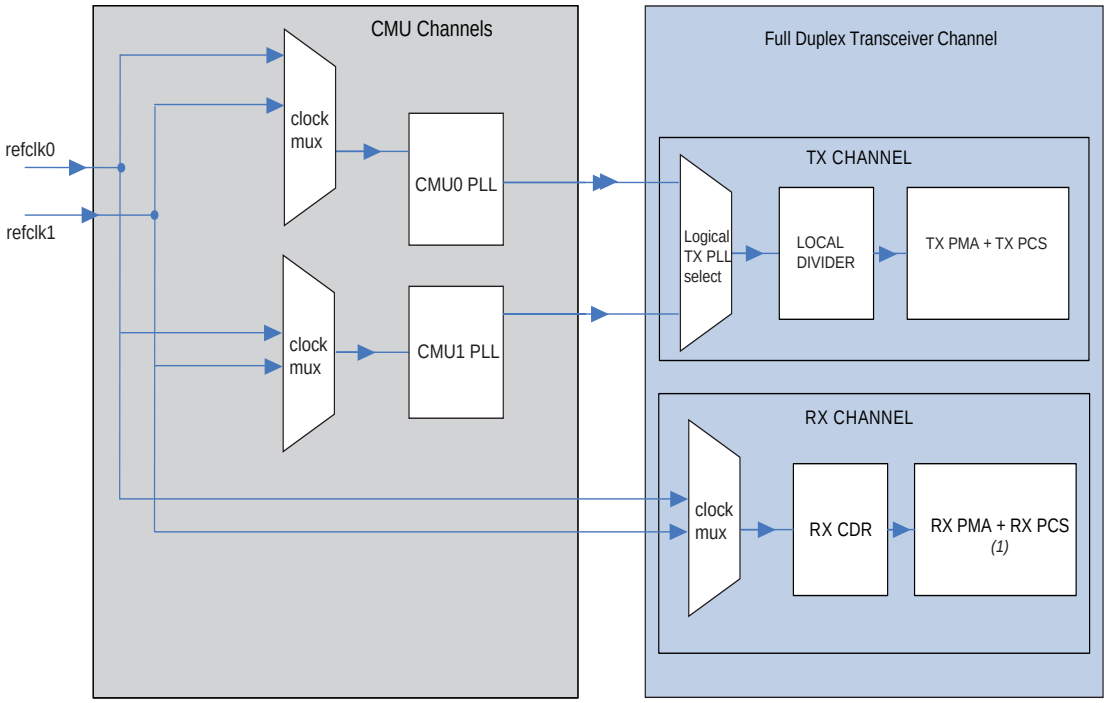
During CMU PLL reconfiguration mode, the dynamic reconfiguration controller automatically powers down the selected CMU PLL until it completes reconfiguration. The ALTGX_RECONFIG instance does not provide external ports to control the CMU PLL power down. When you reconfigure the CMU PLL, the `pll_locked` signal goes low. Therefore, after reconfiguring the transceiver, wait for the `pll_locked` signal from the ALTGX instance before continuing normal operation.

The dynamic reconfiguration controller powers down only the selected CMU PLL. The other CMU PLL is not affected.

Blocks Reconfigured in CMU PLL Reconfiguration Mode

Each transceiver block has two CMU PLLs—`CMU0` PLL and `CMU1` PLL.You can reconfigure each of these CMU PLLs to a different data rate in this mode. Figure 5–26 shows a view of the reconfigurable blocks using CMU PLL reconfiguration mode.

Figure 5–26. CMU PLLs in a Transceiver Block in CMU PLL Reconfiguration Mode



Note to Figure 5–26:

(1) Depending on the mode you select, PCS may or may not be present.

ALTGX MegaWizard Plug-In Manager Setup for CMU PLL Reconfiguration Mode

When you want to reconfigure the CMU PLL to another data rate, enable **.mif** generation and set up the ALTGX MegaWizard Plug-In Manager, as described in the following steps. The dynamic reconfiguration controller reconfigures the CMU PLL with the new information stored in the **.mif**.

1. Select the **Channel and Transmitter PLL reconfiguration** option in the **Modes** screen.
2. Provide the new data rate you want the CMU PLL to run at in the **General** screen.

☞ The logical reference index of `CMU0` PLL within a transceiver block is always the complement of the logical reference index of `CMU1` PLL.

ALTGX_RECONFIG Plug-In Manager Setup for CMU PLL Reconfiguration Mode

For more information, refer to "ALTGX_RECONFIG MegaWizard Plug-In Manager Setup for Channel and CMU PLL Reconfiguration Mode" on page 5–47.

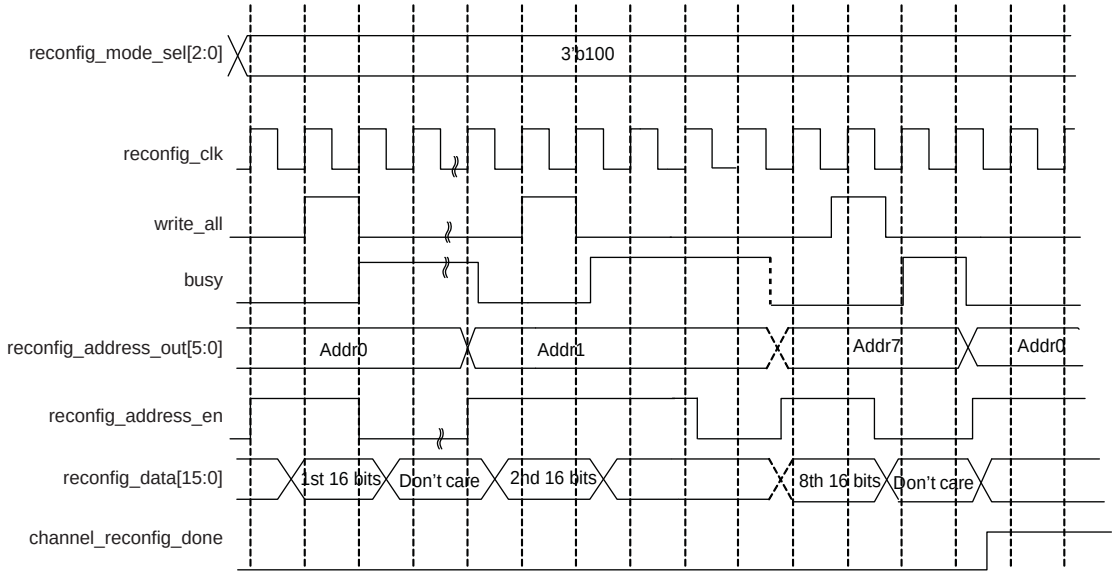
CMU PLL Reconfiguration Operation

Set the `reconfig_mode_sel[2:0]` signal to **3' b100** to activate this mode.

Figure 5–27 shows a **.mif** write transaction in CMU PLL reconfiguration mode. The dynamic reconfiguration controller asserts the `channel_reconfig_done` signal to indicate that the CMU PLL reconfiguration is complete. In this example, the transceiver channel is configured in **Receiver and Transmitter** configuration. Therefore, the **.mif** size is 8.

You can optionally choose to trigger `write_all` once by selecting the continuous write operation in the ALTGX_RECONFIG MegaWizard Plug-In Manager. The Quartus II software then continuously writes all the words required for reconfiguration.

Figure 5–27. CMU PLL Reconfiguration .mif Write Transaction



For guidelines regarding re-using **.mifs**, specifying input reference clocks, or using `logical_tx_pll_sel` ports, refer to "Special Guidelines" on page 5–57.

For more information about reset, refer to the "Reset Sequence when Using Dynamic Reconfiguration with the Channel and TX PLL select/reconfig Option" section in the *Reset Control and Power Down* chapter.

Central Control Unit Reconfiguration Mode Details

Central control unit reconfiguration mode is a **.mif**-based mode used to reconfigure the central control unit (CCU) of the transceiver. Use `reconfig_mode_sel[]` to activate this mode. Central control unit reconfiguration mode is applicable for bonded PCS configurations such as Basic ×4/×8, XAUI, PCI Express (PIPE) ×4/×8, refer to Table 5–5 on page 5–19 for the allowed configurations.

For instance, to dynamically reconfigure an ALTGX instance in Basic ×4 configuration to XAUI configuration, you must first configure:

1. The transceiver channel and CMU PLL to run at the XAUI data rate and functional mode (use channel and CMU PLL reconfiguration mode).
2. Reconfigure the central control unit portion of the transceiver from Basic to XAUI functional mode (use central control unit reconfiguration mode). For more information about the central control unit reconfiguration mode, refer to "Example 2" on page 5–100.

Dynamic reconfiguration is not available if hard IP is used in PCI Express mode

Special Guidelines

The following section describes the special guidelines required for the transceiver channel reconfiguration modes previously described. This section includes the following:

- "Guidelines for Re-Using .mifs" on page 5–57
- "Guidelines for logical_tx_pll_sel and logical_tx_pll_sel_en Ports" on page 5–59
- "Guidelines for Specifying the Input Reference Clocks" on page 5–61

Guidelines for Re-Using .mifs

To configure the transceiver PLLs and receiver CDRs for multiple data rates, it is important to understand the input reference clock requirements. This helps you to efficiently create the clocking scheme for reconfiguration and to reuse the **.mifs** across all channels in the device. This section describes the clocking enhancements and the implications of using input clocks from various clock sources.

The available clock inputs appear as a `pll_inclk_rx_cruclk[]` port and can be provided from the inter-transceiver block lines (also known as ITB lines), from the global clock networks that are driven by an input pin or by a PLL cascade clock.

For more information about input reference clocking, refer to the "Input Reference Clocking" section of the *Stratix IV Transceiver Clocking* chapter.

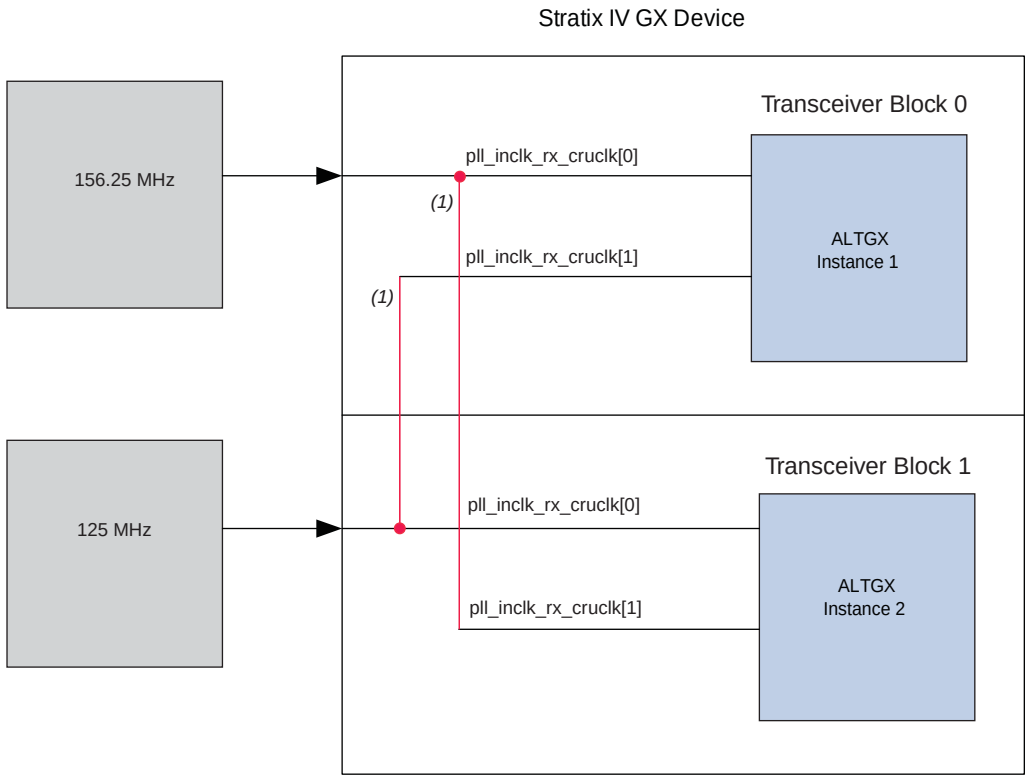
The following section describes the clocking requirements to re-use **.mifs**.

The **.mif** contains information about the input clock multiplexer settings and the functional blocks that you selected during the ALTGX MegaWizard Plug-In Manager instantiation. You can use a **.mif** to dynamically reconfigure any of the other transceiver channels in the device as long as the order of the clock inputs is consistent. For example, assume that a **.mif** is generated for a transceiver channel in transceiver

block 0 and the input clock source is connected to the `pll_inclk_rx_cruclk[0]` port. When you use the generated **.mif** for a channel in other transceiver blocks (for example, transceiver block 1), the same clock source must be connected to the `pll_inclk_rx_cruclk[0]` port. Figure 5–28 and Figure 5–29 show the incorrect and correct order of input reference clocks, respectively.

In Figure 5–28, the clocking is incorrect when re-using the **.mif** because the input reference clock is not connected to the corresponding `pll_inclk_rx_cruclk[]` ports in the two instances.

Figure 5–28. Incorrect Input Reference Clock Connections When Reusing a .mif

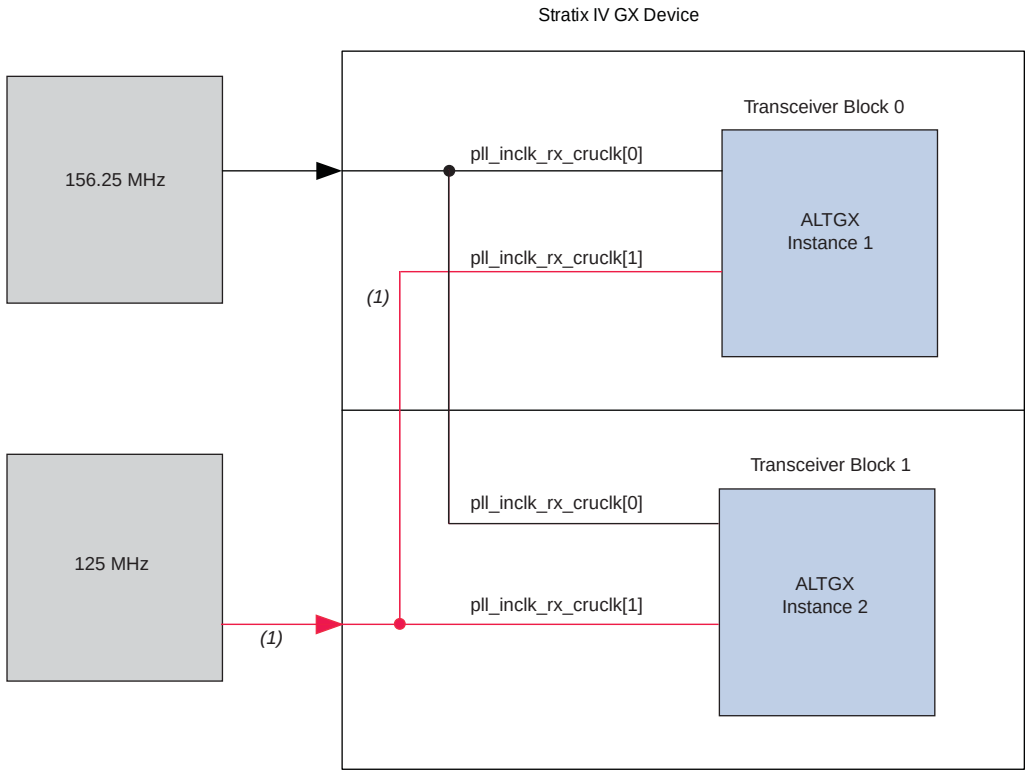


Note to Figure 5–28:

(1) The red lines represent the alternate source of REFCLK.

Figure 5–29 shows the correct input reference clock connections when re-using a **.mif**.

Figure 5–29. Correct Input Reference Clock Connections When Reusing a .mif



Note to Figure 5–29:

(1) The red lines represent the alternate source of REFCLK.

You can re-use the **.mif** generated for a transceiver channel on one side of the device for a transceiver channel on the other side of device, only if the input reference clock frequencies and order of the `pll_inclk_rx_cruclk[]` ports in the ALTGX instances on both sides are identical.

In addition to the input reference clock requirements when re-using a **.mif**, refer to "Guidelines for logical_tx_pll_sel and logical_tx_pll_sel_en Ports" on page 5–59 for additional ways to re-use a **.mif**

Guidelines for logical_tx_pll_sel and logical_tx_pll_sel_en Ports

This section describes when to enable the `logical_tx_pll_sel` and `logical_tx_pll_sel_en` ports and how to use them in the following dynamic reconfiguration modes:

- Channel and CMU PLL reconfiguration mode
- Channel reconfiguration with transmitter PLL select mode
- CMU PLL reconfiguration mode

These are optional input ports to the ALTGX_RECONFIG instance.

Table 5–11 shows the conditions under which the dynamic reconfiguration controller uses either the `logical_tx_pll_sel` port value or the logical reference index value stored in the **.mif**.

Figure 5–30 shows the `logical_tx_pll_sel` and `logical_tx_pll_sel_en` ports.

Figure 5–30. Using logical_tx_pll_sel and logical_tx_pll_sel_en Ports

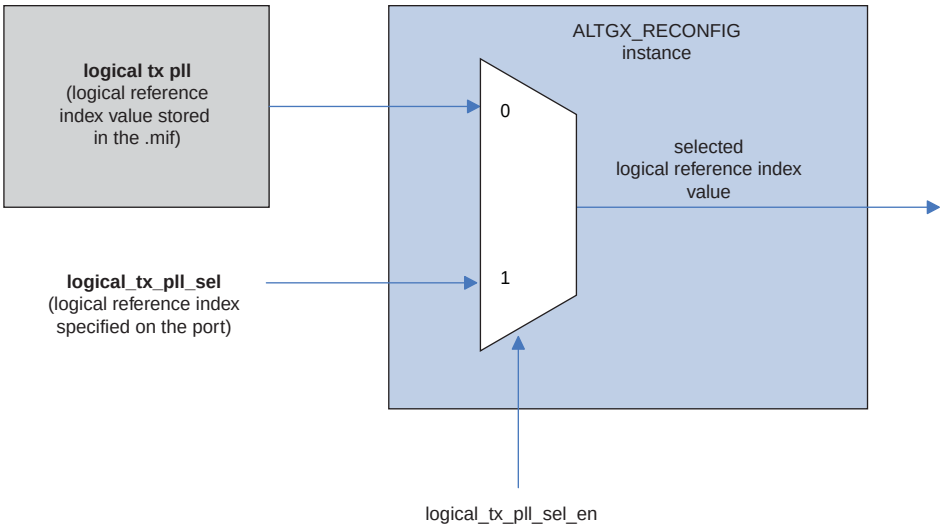


Table 5–11 lists how the dynamic reconfiguration controller selects between the logical reference index stored in the **.mif** (logical tx pll) and the logical reference index specified at the `logical_tx_pll_sel` port.

Table 5–11. Various Combinations of the logical_tx_pll_sel and logical_tx_pll_sel_en Ports

logical_tx_pll_sel	logical_tx_pll_sel_en	Logical Reference Index Value Selected by the ALTGX_RECONFIG Instance
enabled	enabled and value is 1	Value on the `logical_tx_pll_sel` port
enabled	enabled and value is 0	logical reference index value stored in the **.mif** (logical tx pll)
enabled	disabled	Value on the `logical_tx_pll_sel` port
disabled	disabled	logical reference index value stored in the **.mif** (logical tx pll)

When you configure a transceiver channel in the ALTGX MegaWizard Plug-In Manager, Altera recommends that you keep track of the transmitter PLL that drives the channel.

The `logical_tx_pll_sel` port does not modify transceiver settings on the receiver side.

If both the `logical_tx_pll_sel` and `logical_tx_pll_sel_en` ports are enabled, reconfigure the transmitter PLL. Keep the `logical_tx_pll_sel` and `logical_tx_pll_sel_en` signals at a constant logic level until the dynamic reconfiguration controller asserts the `channel_reconfig_done` signal.

Table 5–12 lists the two conditions under which you can re-use **.mifs** when using the `logical_tx_pll_sel` and `logical_tx_pll_sel_en` ports.

Table 5–12. Two Conditions Under Which You can Re-Use .mifs (logical_tx_pll_sel and logical_tx_pll_sel_en)

Condition 1: Re-use the **.mif** created for one CMU PLL on the other CMU PLL of the same transceiver block.		**Condition 2:** Re-use the **.mif** created for one transmitter PLL on the transmitter PLL of another transceiver block.	
Channel and CMU PLL Reconfiguration and CMU PLL Reconfiguration	**Channel Reconfiguration with Transmitter PLL Select**	**Channel and CMU PLL Reconfiguration and CMU PLL Reconfiguration**	**Channel Reconfiguration with Transmitter PLL Select**
Consider that you create a **.mif** containing the desired ALTGX settings to reconfigure the `CMU0` PLL. Assume that the logical reference index you assigned to `CMU0` PLL is **0**. ■ You can re-use this **.mif** created for `CMU0` PLL on `CMU1` PLL of the same transceiver block if you want to reconfigure `CMU1` PLL to the new data rate information stored in the **.mif**. ■ You must set `logical_tx_pll_sel` to the logical reference index of `CMU1` PLL (**1'b1**) and `logical_tx_pll_sel_en` to **1'b1** and then write this **.mif** into the transceiver channel. By doing so, the dynamic reconfiguration controller overwrites the logical tx pll value stored in the **.mif** with the logical reference index of `CMU1` PLL.	Assume that the transceiver channel listens to `CMU1` PLL and the logical reference index assigned to it is **0**. ■ Generate a **.mif** for these settings. ■ When you use channel reconfiguration with transmitter PLL select mode and reconfigure the transceiver channel with this **.mif**, the transceiver channel is reconfigured to listen to `CMU1` PLL. ■ If you want to reconfigure the transceiver channel to listen to `CMU0` PLL instead, you can re-use this **.mif**. ■ You must set `logical_tx_pll_sel` to the logical reference index of `CMU0` PLL (**1'b1**) and `logical_tx_pll_sel_en` to **1'b1** and then write this **.mif** into the transceiver channel.	Consider that you create a **.mif** containing the desired ALTGX settings to reconfigure the transmitter PLL of a transceiver block. Assume that the logical reference of the transmitter PLL is **1**. ■ You can re-use this **.mif** created to reconfigure the transmitter PLL of another transceiver block under the following condition: → You want to reconfigure the transmitter PLL of the other transceiver block to exactly the same data rate information stored in the **.mif**. ■ You must set `logical_channel_address` to the logical channel address of the transmitter PLL you intend to reconfigure.	Consider that you create a **.mif** containing the logical reference index of the transmitter PLL that the reconfigured transceiver channel needs to listen to. ■ Assume that the transmitter PLL used is `CMU0` PLL and the logical reference index assigned is **0**. ■ When you use channel reconfiguration with transmitter PLL select mode and reconfigure the transceiver channel with this **.mif**, the transceiver channel is reconfigured to listen to `CMU0` PLL. ■ If you want to reconfigure this transceiver channel to listen to another transmitter PLL outside the transceiver block, you can reuse this **.mif**, provided the intended data rate is the same.

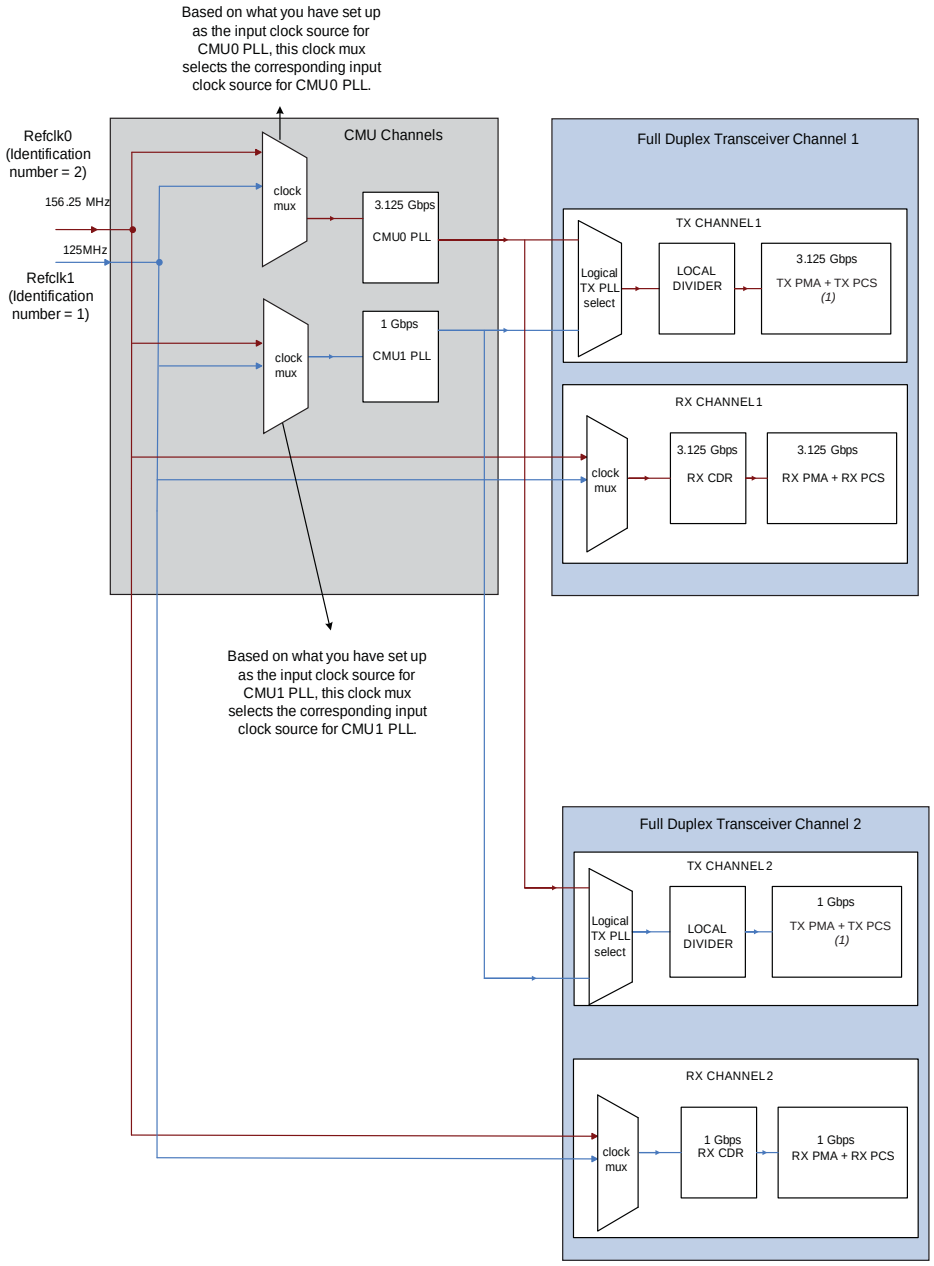
Guidelines for Specifying the Input Reference Clocks

The following are guidelines for setting up the input reference clocks in the **Reconfiguration Settings** screen of the ALTGX MegaWizard Plug-In Manager.

- Assign the identification numbers to all input reference clocks that are used by the transmitter PLLs in their corresponding PLL screens. You can set up a maximum of 10 input reference clocks and assign identification numbers from 1 to 10 (1, 2, 3, 4, 5, 6, 7, 8, 9, and 10).
- Keep the identification numbers consistent for all the **.mifs** generated in the design.
- Maintain the input reference clock frequencies settings for all the **.mifs**.

Figure 5–31 shows an example scenario where the input reference clock connections to the transceiver channels are based on what you set as the input clock source for each of the CMU transmitter PLLs within a transceiver block.

Figure 5–31. Input Reference Clocks Connections to the Transceiver Channels



Note to Figure 5–31:

(1) Depending on the mode you select, PCS unit may or may not be present.

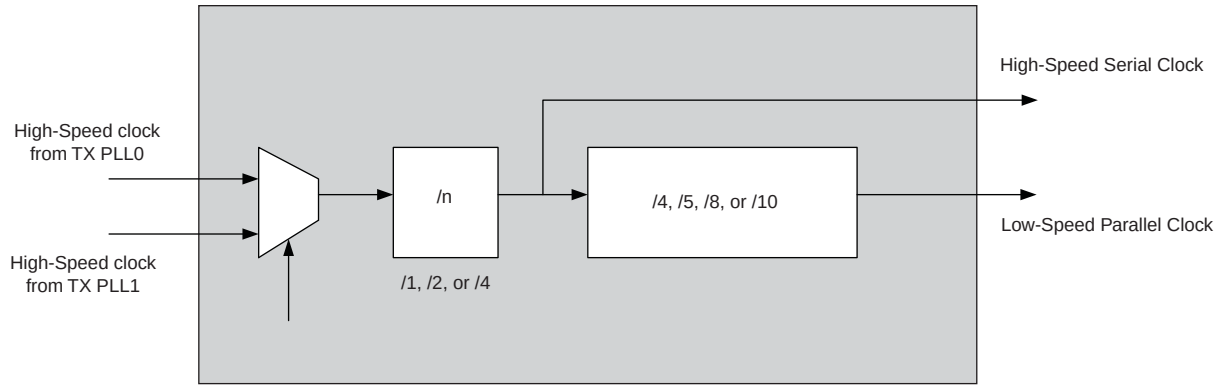
Data Rate Division in Transmitter Mode Details

You can use date rate division in transmitter mode to modify the data rate of the transmitter channel in multiples of 1, 2, and 4. This dynamic reconfiguration mode is available only for the transmit side and not for the receive side.

Blocks Reconfigured in the Data Rate Division in Transmitter Mode

The only block that is reconfigured by the data rate division in transmitter mode is the transmitter local divider block of a transmitter channel. You can set the transmitter local divider to a divide by value of /1, /2, or /4, as shown in Figure 5–32.

Figure 5–32. Local Divider of a Transmitter Channel



You must be aware of the device operating range before you enable and use this feature. There are no legal checks that are imposed by the Quartus II software because it is an on-the-fly control feature. You also need to ensure that a specific functional mode supports the data rate range before dividing the clock when using this rate switch option.

The date rate division in transmitter mode is applicable only to regular transceiver channels and not the PMA-Only channels.

ALTGX MegaWizard Plug-In Manager Setup for Data Rate Division in Transmitter Mode

Enable the following settings in the ALTGX MegaWizard Plug-In Manager:

1. Select the **Channel and Transmitter PLL Reconfiguration** option in the **Reconfig** screen to enable the ALTGX_RECONFIG instance to modify the transmitter channel local divider values dynamically.
2. Set the **What is the starting channel number?** option in the **Reconfig** screen. For more information, refer to "Logical Channel Addressing" on page 5–5.

The alternate reference clock is not required because a single clock source is used. The /1, /2, or /4 data rates can be derived from the single input reference clock.

ALTGX_RECONFIG MegaWizard Plug-In Manager Setup for Data Rate Division in Transmitter Mode

Enable the following settings in the ALTGX_RECONFIG MegaWizard Plug-In Manager for data rate division in transmitter mode:

1. In the **Reconfiguration settings** screen, set the **What is the number of channels controlled by the reconfig controller?** option. For more information, refer to "Total Number of Channels Option in the ALTGX_RECONFIG Instance" on page 5–10.
2. Specify the logical channel address of the transmitter channel at the `logical_channel_address` input port.
3. In the **Reconfiguration settings** screen, select the **Data rate division in TX** option.

The `rate_switch_ctrl[1:0]` input port is available when you enable the **Data rate division in TX** option. The value you set at the `rate_switch_ctrl[1:0]` signal determines the transmitter local divider settings, as explained in "Dynamic Reconfiguration Controller Port List" on page 5–78.

To read the existing local divider settings of the transmitter channel, select the **Use 'rate_switch_out' port to read out the current data rate division** option in the **Error checks/Data rate switch** screen.

Decoding for the `rate_switch_out[1:0]` output signal is the same as the `rate_switch_ctrl[1:0]` input signal.

☞ Dynamic rate switch has no effect on the dividers on the receive side of the transceiver channel. It can be used only for the transmitter.

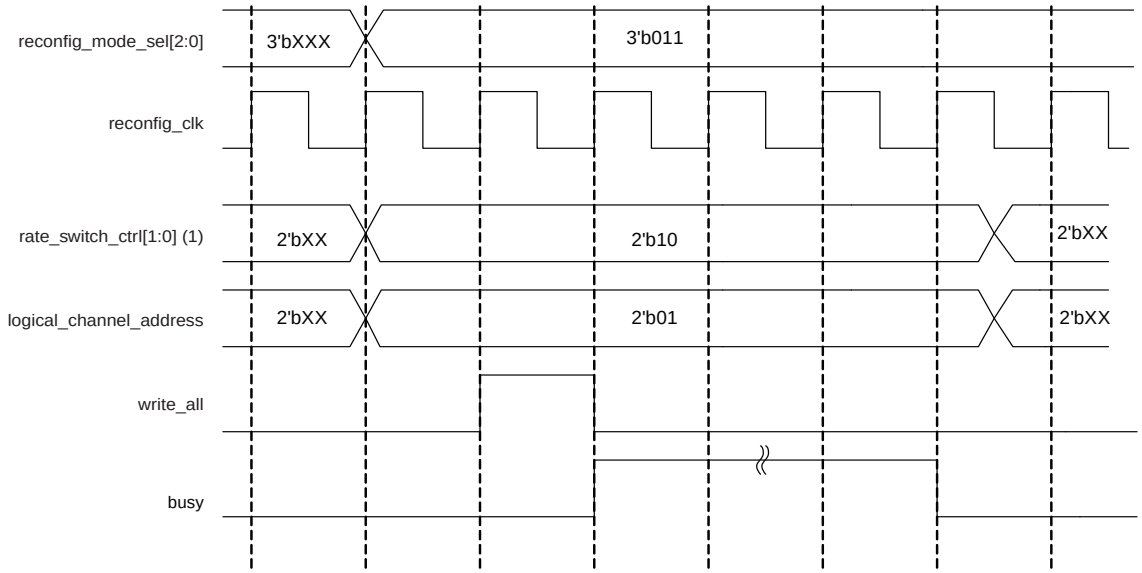
☞ Data rate division in transmitter mode does not require a **.mif**.

Data Rate Division in Transmitter Operation

The following sections describe the steps involved in write and read transactions for the data rate division in transmitter mode.

For this example, the value set in the **What is the number of channels controlled by the reconfig controller?** option of the ALTGX_RECONFIG MegaWizard Plug-In Manager is 4. Therefore, the `logical_channel_address` input is 2 bits wide. Also, you must reconfigure the local divider settings of the transmitter channel whose logical channel address is 2'b01. Figure 5–33 shows a write transaction in data rate division in transmitter mode.

Figure 5–33. Write Transaction in Data Rate Division in Transmitter Mode

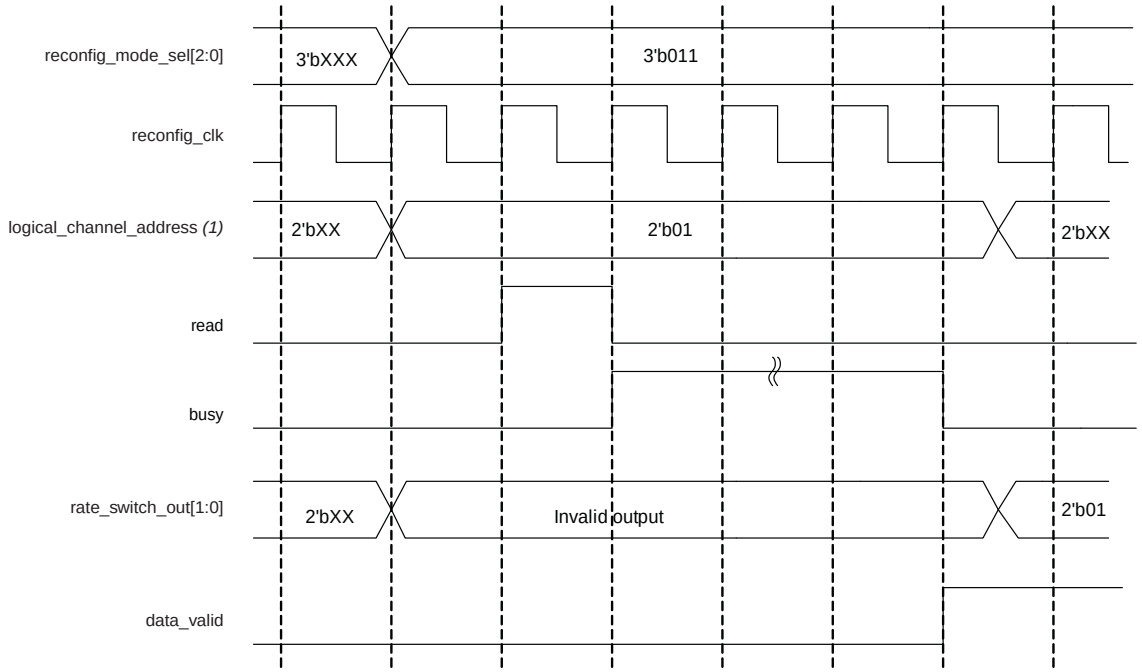


Note to Figure 5–33:

(1) For this example, you want to reconfigure the local divider settings of the transmitter channel to "Divide by 4". Therefore, the value set at `rate_switch_ctrl[1:0]` is **2'b10**.

For this example, the value set in the **What is the number of channels controlled by the reconfig controller?** option of the ALTGX_RECONFIG MegaWizard Plug-In Manager is 4. Therefore, the `logical_channel_address` input is 2 bits wide. Also, you must read the existing local divider settings of the transmitter channel whose logical channel address is 2'b01. Figure 5–34 shows a read transaction waveform in data rate division in transmitter mode.

Figure 5–34. Read Transaction in Data Rate Division in Transmitter Mode



Note to Figure 5–34:

(1) For this example, the existing local divider settings of the transmitter channel are "Divide by 2". Therefore, the value read out at `rate_switch_out[1:0]` is **2'b01**.

Do not perform a read transaction in date rate division in transmitter mode if `rate_switch_out[1:0]` is not selected in the ALTGX_RECONFIG MegaWizard Plug-In Manager.

For more information about reset, refer to the "Reset Sequence when Using Dynamic Reconfiguration with the Channel and TX PLL select/reconfig Option" section in the *Reset Control and Power Down* chapter.

Offset Cancellation Feature

The Stratix IV GX and GT devices provide an offset cancellation circuit per receiver channel to counter the offset variations due to process, voltage, and temperature (PVT). These variations create an offset in the analog circuit voltages, pushing them out of the expected range. In addition to reconfiguring the transceiver channel, the dynamic reconfiguration controller performs offset cancellation on all receiver channels connected to it on power up.

The **Offset cancellation for Receiver channels** option is automatically enabled in both the ALTGX and ALTGX_RECONFIG MegaWizard Plug-In Managers for **Receiver and Transmitter** and **Receiver only** configurations. It is not available for **Transmitter only** configurations. For **Receiver and Transmitter** and **Receiver only** configurations, you must connect the necessary interface signals between the ALTGX_RECONFIG and ALTGX (with receiver channels) instances.

Offset cancellation is automatically executed once every time the device is powered on. The control logic for offset cancellation is integrated into the dynamic reconfiguration controller. You must connect the ALTGX_RECONFIG instance to the ALTGX instances (with receiver channels) in your design. You must connect the `reconfig_fromgxb`, `reconfig_togxb`, and necessary clock signals to both the ALTGX_RECONFIG and ALTGX (with receiver channels) instances.

☞ The offset cancellation control functionality remains the same for both regular transceiver channels and PMA-Only channels.

Operation

Every ALTGX instance for **Receiver and Transmitter** or **Receiver only** configurations require that the **Offset cancellation for Receiver channels** option is enabled in the **Reconfig** screen of the ALTGX MegaWizard Plug-In Manager. This option is enabled by default for the above two configurations. It is disabled for the **Transmitter only** configuration.

Because this option is enabled by default, the ALTGX instance must be connected to an ALTGX_RECONFIG instance (dynamic reconfiguration controller). The offset cancellation controls are also enabled by default in the **Reconfiguration settings** screen of the ALTGX_RECONFIG instance.

You must also set the starting channel number in the **What is the starting channel number?** option for every ALTGX instance connected to the ALTGX_RECONFIG instance. For more information, refer to:

- "Logical Channel Addressing of Regular Transceiver Channels" on page 5–6
- "Logical Channel Addressing of PMA-Only Channels" on page 5–7
- "Logical Channel Addressing—Combination of Regular Transceiver Channels and PMA-Only Channels" on page 5–9.

When the device powers up, the dynamic reconfiguration controller initiates offset cancellation on the receiver channel by disconnecting the receiver input pins from the receiver data path. It also sets the receiver CDR into a fixed set of dividers to guarantee a voltage controlled oscillator (VCO) clock rate within the range necessary to provide proper offset cancellation. Subsequently, the offset cancellation process goes through different states and culminates in the offset cancellation of the receiver buffer and receiver CDR. After offset cancellation is complete, the user divider settings are restored.

The dynamic reconfiguration controller sends and receives data to the transceiver channel through the `reconfig_togxb` and `reconfig_fromgxb` signals. You must connect these signals between the ALTGX_RECONFIG instance and the ALTGX instance. You must also set the **What is the number of channels controlled by the reconfig controller?** option in the **Reconfiguration settings** screen of the ALTGX_RECONFIG MegaWizard Plug-In Manager. For more information, refer to "Total Number of Channels Option in the ALTGX_RECONFIG Instance" on page 5–10.

The **Use 'logical_channel_address' port for Analog controls reconfiguration** option in the **Analog controls** screen of the ALTGX_RECONFIG MegaWizard Plug-In Manager is not applicable for the receiver offset cancellation process.

If the design does not require PMA controls reconfiguration and uses optimum LE resources, you can connect all the ALTGX instances in the design to a single dynamic reconfiguration controller (ALTGX_RECONFIG instance).

The `gxb_powerdown` signal must not be asserted during the offset cancellation sequence.

To understand the impact on system start-up when you control all the transceiver channels using a single dynamic reconfiguration controller, refer to "PMA Controls Reconfiguration Duration" on page 5–92.

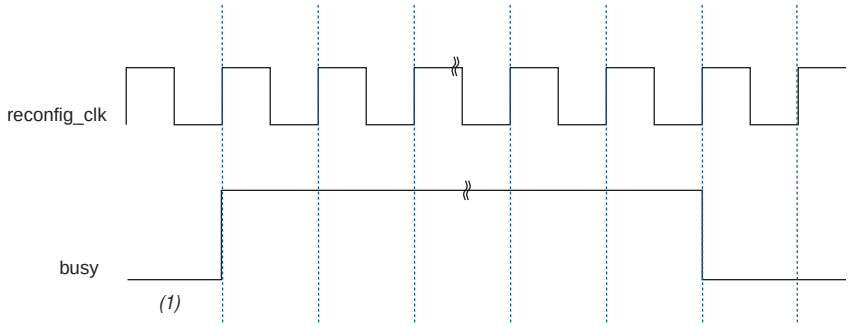
ALTGX_RECONFIG Instance Signals Transition during Offset Cancellation

Consider that the design has ALTGX instances with channels of both **Transmitter only** and **Receiver only** configurations. You must include the **Transmitter only** channels while setting the **What is the starting channel number?** option in the ALTGX instance and setting the **What is the number of channels controlled by the reconfig controller?** option in the ALTGX_RECONFIG instance for receiver offset cancellation.

- After the device powers up, the `busy` signal remains low for the first `reconfig_clk` clock cycle.
- The `busy` signal then gets asserted for the second `reconfig_clk` clock cycle, when the dynamic reconfiguration controller initiates the offset cancellation process.
- The de-assertion of the `busy` signal indicates the successful completion of the offset cancellation process.

Figure 5–35 shows the dynamic reconfiguration signals transition during offset cancellation on receiver channels.

Figure 5–35. Dynamic Reconfiguration Signals Transition during Offset Cancellation on Receiver Channels



Note to Figure 5–35:

(1) After device power up, the `busy` signal remains low for the first `reconfig_clk` cycle.

Due to the offset cancellation process, the transceiver reset sequence has changed. For more information, refer to the *Reset Control and Power Down* chapter.

EyeQ

EyeQ hardware is available in Stratix IV transceivers to analyze and debug the receiver data recovery path (receiver gain, clock jitter, and noise level). You can use it to monitor the eye width and assess the quality of the incoming signal.

Normally, the receiver CDR samples the incoming signal at the center of the eye. When you enable the EyeQ hardware, it allows the CDR to sample across 64 different positions within two unit intervals (UIs) of the incoming data. You can manually control the sampling points and check the bit-error rate (BER) at each of these 64 sampling points. These sampling points are also known as phase steps.

The BER increases at the edge of the eye-opening. By observing the number of sampling points results in a desired BER value, you can determine the eye width.

The EyeQ hardware is available for both regular transceiver channels and CMU channels.

For more information about the supported data rates, phase step translation, and other specifications, refer to the *DC and Switching Characterization* chapter.

Enabling the EyeQ Control Logic and the EyeQ Hardware

You must enable the EyeQ hardware in the ALTGX MegaWizard Plug-In Manager and the EyeQ control block in the ALTGX_RECONIG MegaWizard Plug-In Manager.

- EyeQ hardware is available for each transceiver channel in the receiver data path. Select the **Analog Controls** option in the **Reconfiguration Settings** screen of the ALTGX MegaWizard Plug-In Manager to enable the EyeQ hardware.
- EyeQ control logic is available in the dynamic reconfiguration controller. Select the **EyeQ control** option in the ALTGX_RECONFIG MegaWizard Plug-In Manager to enable the EyeQ control logic.

EyeQ uses an Avalon Memory Mapped interface. For more information about this interface, refer to the *Avalon specification*.

Connections Between the ALTGX and ALTGX_RECONFIG Instances

To enable the EyeQ options, use the following steps:

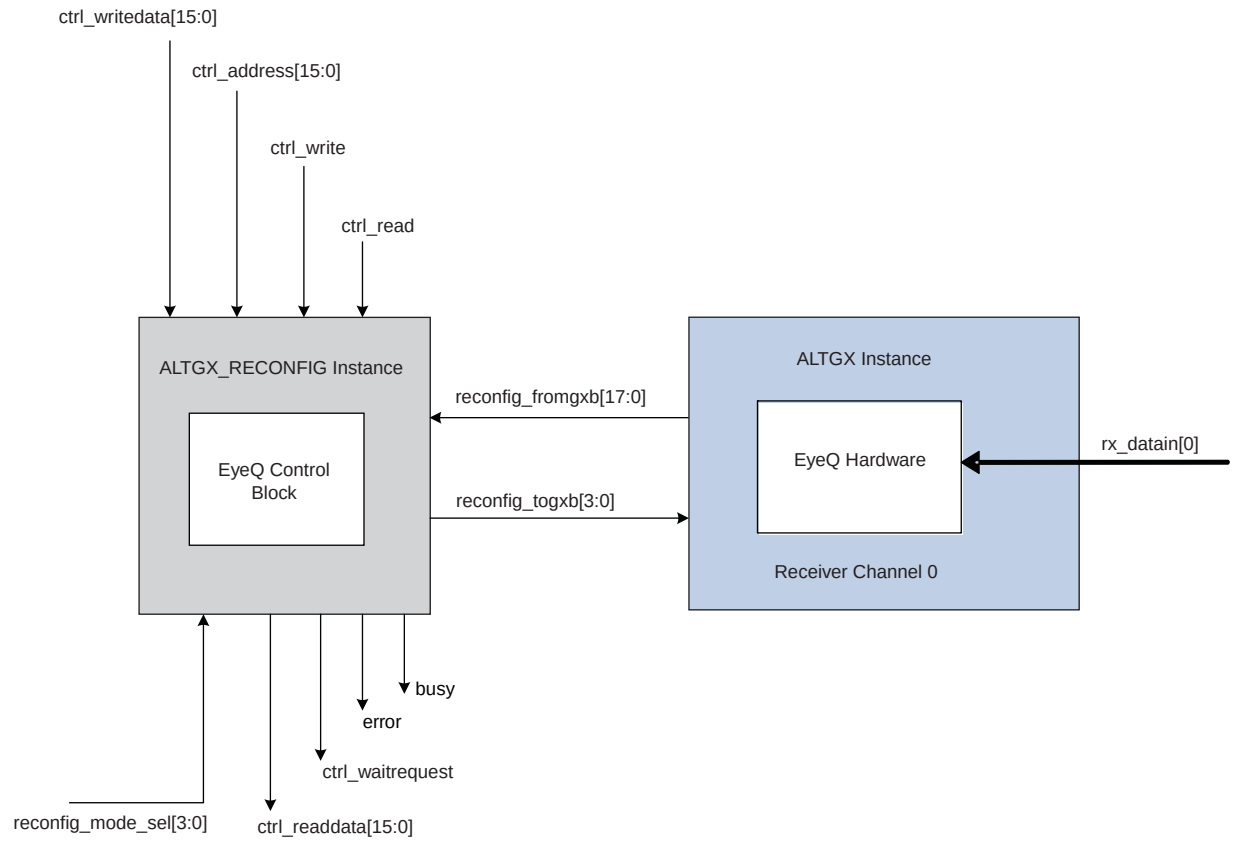
1. Enable the **EyeQ** options in the ALTGX and ALTGX_RECONFIG MegaWizard Plug-In Managers, as explained in "Enabling the EyeQ Control Logic and the EyeQ Hardware" on page 5–69.
2. Connect the `reconfig_{to/from}gxb` ports between the ALTGX and ALTGX_RECONFIG instances.

EyeQ control logic in the dynamic reconfiguration controller allows you to write to the registers in the EyeQ hardware. Therefore, you must have a state machine in the user design that communicates to the EyeQ control block of the ALTGX_RECONFIG instance. You can then access the internal registers of the EyeQ hardware indirectly through the EyeQ control logic.

Altera recommends having an input pattern generator and checker to monitor the BER of the received data.

Figure 5–36 shows the connections between the EyeQ hardware in the ALTGX instances and the EyeQ control logic in the dynamic reconfiguration controller.

Figure 5–36. Connecting ALTGX and ALTGX_RECONFIG Instances with EyeQ Enabled



Controlling the EyeQ Hardware

The EyeQ hardware is controlled by writing to the EyeQ registers using EyeQ interface registers in the ALTGX_RECONFIG instance. Table 5–13 shows the register memory of the 16-bit EyeQ registers.

Table 5–13. EyeQ Register Mapping

Address	Description
0×0	■ Bit[0]—0/1: Disable/Enable EyeQ feature ■ Bit [15:1]—15'b000000000000000
0×1	■ Bits [5:0]—EyeQ phase step value. Refer to Table 5–14 for the EyeQ phase step encoding. ■ Bits [15:6]—10'b0000000000

Table 5–14 shows the EyeQ phase step encoding for the 32 phase steps spanning one unit interval (UI).

Table 5–14. EyeQ Phase Step Encoding (Part 1 of 2)

Desired Phase Step Setting	EyeQ Phase Step Encoding
0	6'b111111
1	6'b111110
2	6'b111101
3	6'b111100
4	6'b111011
5	6'b111010
6	6'b111001
7	6'b111000
8	6'b110111
9	6'b110110
10	6'b110101
11	6'b110100
12	6'b110011
13	6'b110010
14	6'b110001
15	6'b110000
16	6'b010000
17	6'b010001
18	6'b010010
19	6'b010011
20	6'b010100
21	6'b010101
22	6'b010110
23	6'b010111

Table 5–14. EyeQ Phase Step Encoding (Part 2 of 2)

Desired Phase Step Setting	EyeQ Phase Step Encoding
24	6'b011000
25	6'b011001
26	6'b011010
27	6'b011011
28	6'b011100
29	6'b011101
30	6'b011110
31	6'b011111

Table 5–15 shows the register memory of the 16-bit EyeQ interface registers.

Table 5–15. EyeQ Interface Register Mapping

Address	Description
0×0	Control/Status register (EyeQ CSR) ■ Bit [0]—Start: Writing a 1 to this bit instructs the ALTGX_RECONFIG instance to program the EyeQ hardware. Writing to this bit automatically clears any error bits. ■ Bit [1]—Read/Write: Writing a 1 to this bit writes the contents of the data register to one of the EyeQ registers depending on the address stored in the EyeQ register address register. Writing a 0 reads the contents of the EyeQ register. ■ Bit [12:2]—11'b00000000000 ■ Bit [13] —Channel address error: This bit is set to 1 if the programmed channel address is invalid. Writing a 1 to this bit clears the error. ■ Bit [14]—EyeQ register address error: this bit is set to 1 if the programmed word address is invalid. Writing a 1 to this bit clears the error. ■ Bit [15]—Busy status: The value of this bit can be polled to determine if the ALTGX_RECONFIG read/write request has completed. When this active-high bit is asserted, all registers become read only until this bit is de-asserted.
0×1	Channel address [15:0]—Specifies the transceiver channel for the desired EyeQ operation. This must match the `logical_channel_address` input port.
0×2	EyeQ register address [15:0]—Specifies the address EyeQ register to be read from or written to. The values supported are 0×0 or 0×1.
0×3	Data [15:0]— ■ For a write operation, the data in this register is written to the EyeQ register selected. ■ For a read operation, this register contains the contents of the EyeQ register selected. The data in this register is only valid when the busy status is low. A read operation overwrites the current contents of this register.

To control the EyeQ hardware, follow these steps:

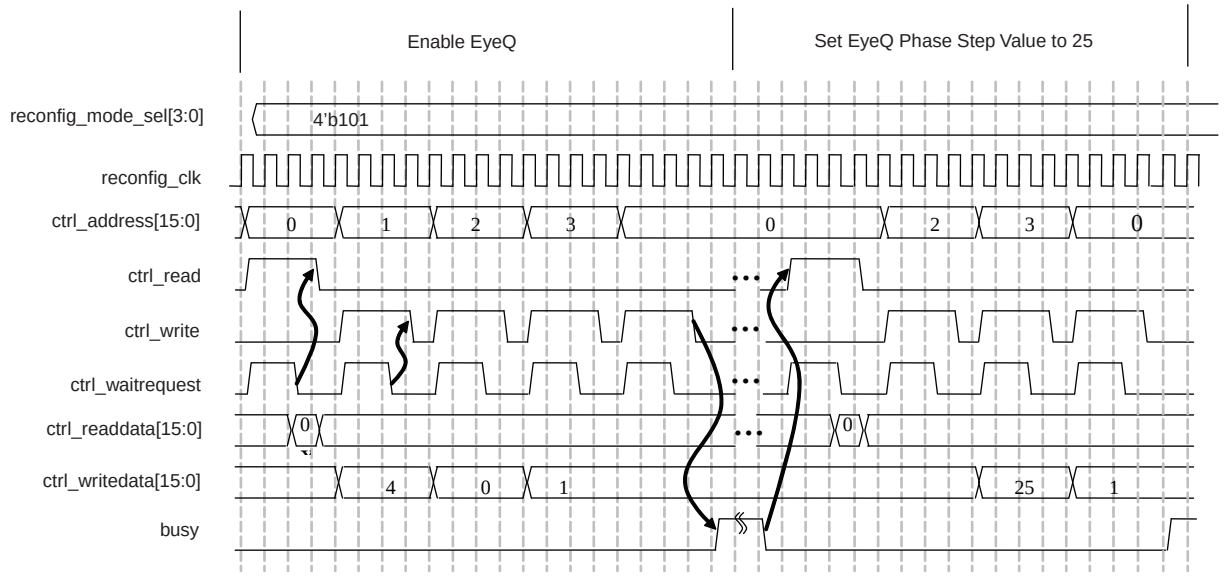
1. Read the EyeQ interface register 0×0 (the control and status register) to check the busy status. The clear status bit indicates an idle status.
2. Issue a `write` to the EyeQ interface register 0×1 (the channel address register) to select the desired channel.
3. Issue a `write` to the EyeQ interface register 0×2 (the eye monitor register address) to select the desired EyeQ register.
4. Issue a `write` to the EyeQ interface register 0×3 (the data register) to provide the data to be written to the target EyeQ register.
5. Issue a `write` to EyeQ interface register 0×0 (the control and status register) to specify read/~write and to issue the start command.
6. Poll the EyeQ interface register 0×0 (the control and status register) and wait for the busy status to be de-asserted. Once the status is no longer busy, the data is considered successfully written for write transactions. For read transactions, this indicates that the contents of the data register has been updated and can be read out. Note that all `writes` that occur when the busy status is asserted are ignored; all registers become read only.
7. If the next operation is to the same EyeQ register and same channel, you do not need to repeat steps 2 and 3.

Example of Using the EyeQ Feature

Consider a design with one regular transceiver channel configured in Basic functional mode. The channel has a data rate of 2.5 Gbps with the EyeQ feature enabled in both the ALTGX and ALTGX_RECONFIG instances. Figure 5–37 shows how the EyeQ mode is first enabled by writing into the EyeQ registers using the EyeQ interface registers. A phase step value of 25 is written to the EyeQ register. Before performing any operation, the following conditions must be met:

- The `busy` bit is **0** in the EyeQ CSR
- The `ctrl_waitrequest` is low

Figure 5–37. Enabling EyeQ Mode



Adaptive Equalization (AEQ)

High-speed interface systems require different equalization settings to compensate for changing data rates and backplane losses. Manual tuning of the receiver channel's equalization stages involves finding the optimal settings through trial and error, and then locking in those values at compile time. This manual method is cumbersome under varying system characteristics. The AEQ feature solves this problem by automatically tuning an active receiver channel's equalization filters based on a frequency content comparison between the incoming signal and internally generated reference signals.

User logic can dynamically control the AEQ hardware in the receiver through the dynamic reconfiguration controller. This section describes how to enable different options and use them to control the AEQ hardware.

Adaptive Equalization Limitations

The following are the AEQ feature requirements and limitations:

- The receive data must be 8B/10B encoded
- Not available in PCI-Express (PIPE) functional mode (because the AEQ hardware cannot perform the equalization process when the receive link is under the electrical idle condition)
- The receiver input signal must have a minimum envelope of 400 mv (differential peak-to-peak). The Quartus II software does not check for this requirement
- AEQ hardware is not present in the CMU channels

For more information about speed grade, data rates, receiver input signal level, and other specifications that support the AEQ feature, refer to the *DC and Switching Characterization* chapter.

Enabling the AEQ Control Logic and AEQ Hardware

To use the AEQ feature, enable the AEQ hardware in the ALTGX MegaWizard Plug-In Manager and the AEQ control block in the ALTGX_RECONIG MegaWizard Plug-In Manager. To enable the AEQ hardware and the AEQ control logic:

- Select the **Enable adaptive equalizer control** option in the **Reconfiguration Settings** screen of the ALTGX MegaWizard Plug-In Manager. The AEQ hardware is available for each transceiver channel in the receiver data path.
- Select the **Enable adaptive equalizer control** option in the ALTGX_RECONFIG MegaWizard Plug-In Manager. The AEQ control logic is available in the dynamic reconfiguration controller.

When you select the above two options, the ALTGX and ALTGX_RECONFIG MegaWizard Plug-In Managers provide the following additional ports:

- `aeq_fromgxb[]`
- `aeq_togxb[]`

The `aeq_fromgxb[]` and `aeq_togxb[]` ports provide the interface between the receiver channel and the dynamic reconfiguration controller.

The following section describes the connections between the AEQ control block of the ALTGX_RECONFIG instance and the AEQ hardware of the ALTGX instance.

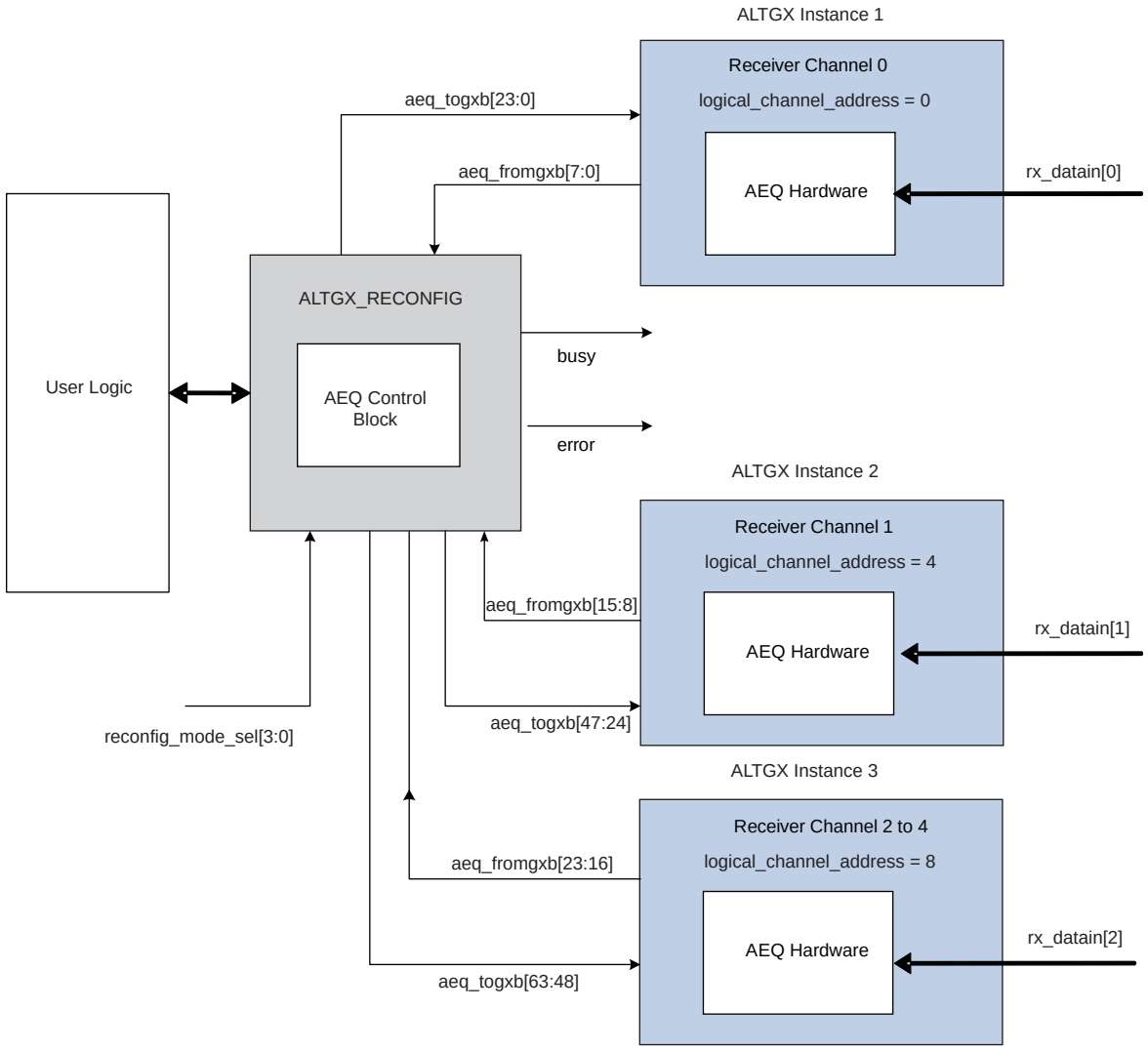
Connections Between the ALTGX and ALTGX_RECONFIG Instances

Enable the **adaptive equalization** options in the ALTGX and ALTGX_RECONFIG MegaWizard Plug-In Managers, as explained in the previous section. To use the AEQ control block and AEQ hardware, you must connect the ALTGX receivers to the ALTGX_RECONFIG instance using the `reconfig_{to/from}gxb` and `aeq_{to/from}gxb` ports. You must also connect the ALTGX_RECONFIG instance to your design.

If you have multiple transceiver instances and a single ALTGX_RECONFIG instance, connect the LSB of the `aeq_togxb[]` and `aeq_fromgxb[]` ports of the ALTGX_RECONFIG instance to the transceiver channel with a `logical_channel_address` value of 0.

Figure 5–38 shows the `aeq_fromgxb[]` and `aeq_togxb[]` connections between multiple ALTGX instances and the dynamic reconfiguration controller.

Figure 5–38. Connecting the ALTGX and ALTGX_RECONFIG Instances with AEQ Enabled



You have three options to control the AEQ hardware using the ALTGX_RECONFIG instance. The following section explains the three user modes.

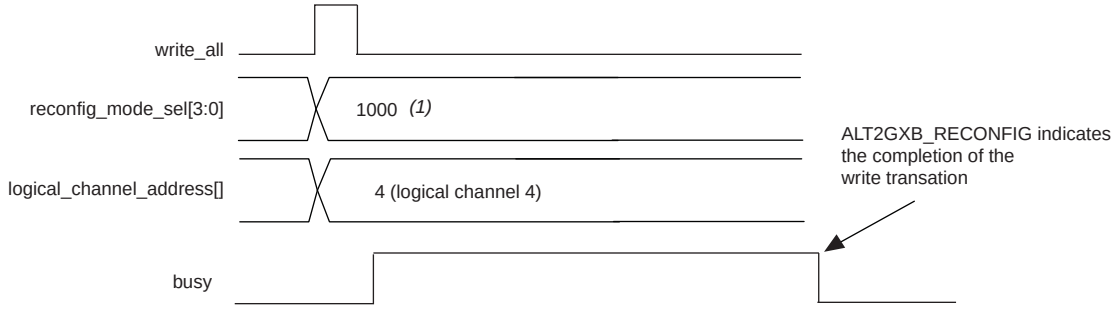
Controlling the AEQ Hardware

Use `reconfig_mode_sel[3:0]`to select one of the following three modes.

Continuous Mode for a Single Channel

All functionalities of the AEQ hardware are active and the equalization stages are continually being feedback-optimized (Figure 5–39).

Figure 5–39. Timing Diagram for Enabling AEQ in Continuous Mode for a Single Channel (Modes 1 and 2)



Note to Figure 5–39:

(1) **1000** for Mode 1 and **1001** for Mode 2.

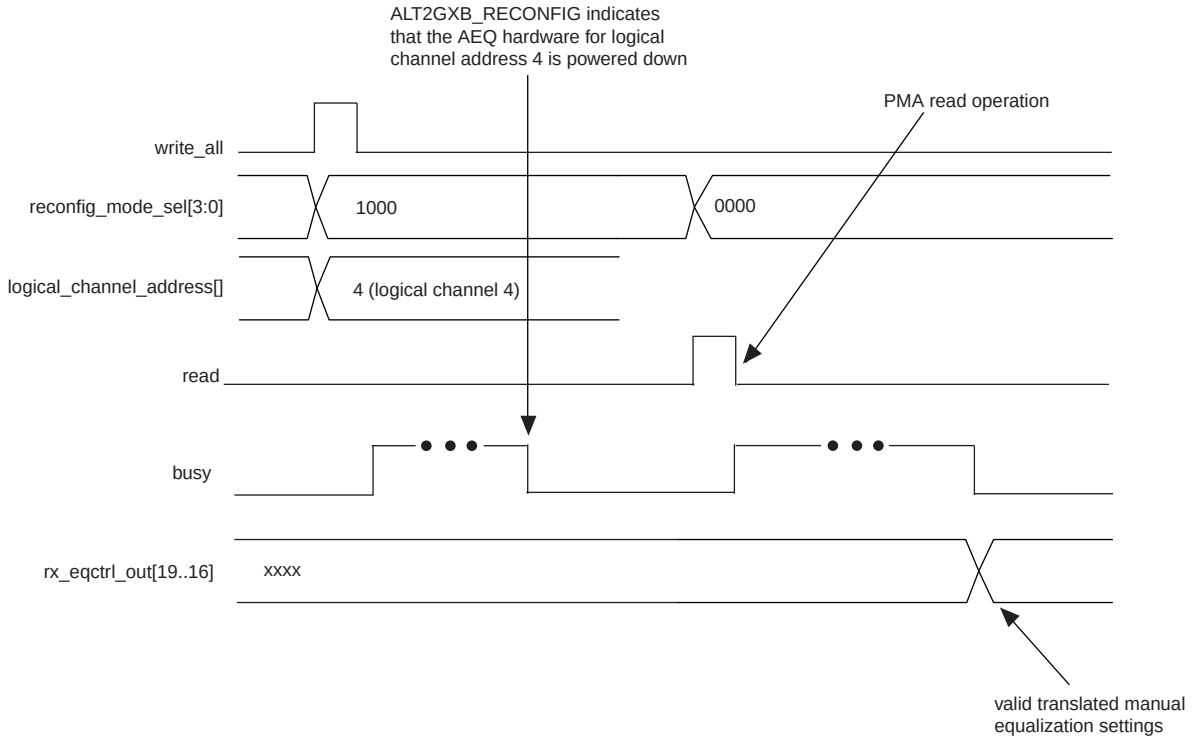
One Time Mode for a Single Channel

The AEQ hardware attempts to find a stable equalization and then locks to that value. Once locked, the equalization values are held and are no longer updated. The timing diagram for this mode is similar to mode 1 shown in Figure 5–39 with the exception that the `reconfig_mode_sel[3:0]` value used is **1001**.

Powerdown for a Single Channel

The AEQ hardware of the specified receiver channel is put in standby mode. The AEQ hardware comes out of the standby mode as soon as you change the value at `reconfig_mode_sel[3:0]` to one of the other two AEQ control modes. The AEQ hardware of the powered down receiver channel does not remember the converged equalization value once it comes out of the standby mode. It instead starts at the maximum equalization value after powering up again (Figure 5–40).

Figure 5–40. Timing Diagram for Powering Down the AEQ for a Single Channel (Mode 3)



Dynamic Reconfiguration Controller Port List

Table 5–16 lists the input control ports and output status ports of the dynamic reconfiguration controller.

Table 5–16. Dynamic Reconfiguration Controller Port List (ALTGX_RECONFIG Instance) (Part 1 of 13) *(Note 3)*, *(4)*

Port Name	Input/ Output	Description
Clock Inputs to ALTGX_RECONFIG Instance		
`reconfig_clk`	Input	The frequency range of this clock depends on the following transceiver channel configuration modes: ■ **Receiver only** (37.5 MHz to 50 MHz) ■ **Receiver and Transmitter** (37.5 MHz to 50 MHz) ■ **Transmitter only** (2.5 MHz to 50 MHz) By default, the Quartus II software assigns a global clock resource to this port. This clock must be a free-running clock sourced from an I/O clock pin. Do not use dedicated transceiver REFCLK pins or any clocks generated by transceivers.

Table 5–16. Dynamic Reconfiguration Controller Port List (ALTGX_RECONFIG Instance) (Part 2 of 13) *(Note 3)*, *(4)*

Port Name	Input/ Output	Description
ALTGX and ALTGX_RECONFIG Interface Signals		
`reconfig_fromgxb`	Input	An output port in the ALTGX instance and an input port in the ALTGX_RECONFIG instance. This signal is transceiver-block based. Therefore, the width of this signal increases in steps of 17 bits per transceiver block. In the ALTGX MegaWizard Plug-In Manager, the width of this signal depends on the following: ■ Whether the channels configured in the ALTGX instance are regular transceiver channels or PMA-Only channels. ■ The number of channels you select in the **What is the number of channels?** option in the **General** screen. For example, if the channels in the ALTGX instance are regular transceiver channels and if you select the number of channels as follows: 1 ≤Channels ≤4, then the output port `reconfig_fromgxb` = 17 bits 5 ≤Channels ≤8, then the output port `reconfig_fromgxb` = 34 bits 9 ≤Channels ≤12, then the output port `reconfig_fromgxb` = 51 bits However, if the channels in the ALTGX instance are PMA-Only channels and if you select the number of channels as follows: Number of PMA-Only channels = n, then the output port `reconfig_fromgxb` = n*17 bits For example, `reconfig_fromgxb` = 6 * 17 bits for 6 PMA-Only channels. In the ALTGX_RECONFIG MegaWizard Plug-In Manager, the width of this signal depends on the value you select in the **What is the number of channels controlled by the reconfig controller?** option in the **Reconfiguration settings** screen. For example, if you select the total number of channels controlled by ALTGX_RECONFIG instance as follows: 1 ≤ Channels ≤ 4, then the input port `reconfig_fromgxb` = 17 bits 5 ≤ Channels ≤ 8, then the input port `reconfig_fromgxb` = 34 bits 9 ≤ Channels ≤12, then the input port `reconfig_fromgxb` = 51 bits

Table 5–16. Dynamic Reconfiguration Controller Port List (ALTGX_RECONFIG Instance) (Part 3 of 13) *(Note 3)*, *(4)*

Port Name	Input/ Output	Description
`reconfig_fromgxb` (continued)	Input	To connect the `reconfig_fromgxb` port between the ALTGX_RECONFIG instance and multiple ALTGX instances, follow these rules: ■ Connect the `reconfig_fromgxb[16:0]` of ALTGX Instance 1 to the `reconfig_fromgxb[16:0]` of the ALTGX_RECONFIG instance. Connect the `reconfig_fromgxb[]` port of the next ALTGX instance to the next available bits of the ALTGX_RECONFIG instance, and so on. ■ Connect the `reconfig_fromgxb` port of the ALTGX instance, which has the highest **What is the starting channel number?** option, to the MSB of the `reconfig_fromgxb` port of the ALTGX_RECONFIG instance. The Quartus II Fitter produces an error if the dynamic reconfiguration option is enabled in the ALTGX instance but the `reconfig_fromgxb` and `reconfig_togxb` ports are not connected to the ALTGX_RECONFIG instance. For more information, refer to "Connecting the ALTGX and ALTGX_RECONFIG Instances" on page 5–11.
`reconfig_togxb[3:0]`	Output	An input port of the ALTGX instance and an output port of the ALTGX_RECONFIG instance. You must connect the `reconfig_togxb[3:0]` input port of every ALTGX instance controlled by the dynamic reconfiguration controller to the `reconfig_togxb[3:0]` output port of the ALTGX_RECONFIG instance. The width of this port is always fixed to 3 bits. For more information, refer to "Connecting the ALTGX and ALTGX_RECONFIG Instances" on page 5–11.
FPGA Fabric and ALTGX_RECONFIG Interface Signals		
`write_all`	Input	Assert this signal for one `reconfig_clk` clock cycle to initiate a write transaction from the ALTGX_RECONFIG instance to the ALTGX instance. You can use this signal in two ways for **.mif**-based modes: ■ Continuous write operation—Select the **Enable continuous write of all the words needed for reconfiguration** option to pulse the `write_all` signal only once for writing a whole **.mif**. The **What is the read latency of the MIF contents** option is available for selection in this case only. Enter the desired latency in terms of the `reconfig_clk` cycles. ■ Regular write operation—When the **Enable continuous write of all the words needed for reconfiguration** option is disabled, every word of the **.mif** requires its own write cycle.

Table 5–16. Dynamic Reconfiguration Controller Port List (ALTGX_RECONFIG Instance) (Part 4 of 13) *(Note 3)*, *(4)*

Port Name	Input/ Output	Description
`busy`	Output	This signal is used to indicate the busy status of the dynamic reconfiguration controller during offset cancellation. After the device powers up, this signal remains low for the first `reconfig_clk` clock cycle. It then is asserted and remains high when the dynamic reconfiguration controller performs offset cancellation on all the receiver channels connected to the ALTGX_RECONFIG instance. De-assertion of the `busy` signal indicates the successful completion of the offset cancellation process. For more information, refer to "Operation" on page 5–67. ■ PMA controls reconfiguration mode—This signal is high when the dynamic reconfiguration controller performs a read or write transaction. ■ All other dynamic reconfiguration modes—This signal is high when the dynamic reconfiguration controller writes the **.mif** into the transceiver channel.
`read`	Input	Assert this signal for one `reconfig_clk` clock cycle to initiate a read transaction. The `read` port is applicable only to the PMA controls reconfiguration mode. The `read` port is available when you select **Analog controls** in the **Reconfiguration settings** screen and select at least one of the PMA control ports in the **Analog controls** screen. For more information, refer to "Dynamically Reconfiguring PMA Controls" on page 5–13.
`data_valid`	Output	Applicable only to PMA controls reconfiguration mode. This port indicates the validity of the data read from the transceiver by the dynamic reconfiguration controller. The current data on the output read ports is the valid data ONLY if `data_valid` is high. This signal is enabled when you enable at least one PMA control port used in read transactions, for example `tx_vodctrl_out`.
`error`	Output	This indicates that an unsupported operation is attempted. You can select this in the **Error checks/Data rate switch** screen. The dynamic reconfiguration controller de-asserts the `busy` signal and asserts the `error` signal for two `reconfig_clk` cycles when you attempt an unsupported operation. For more information, refer to the "Error Indication During Dynamic Reconfiguration" on page 5–90.

Table 5–16. Dynamic Reconfiguration Controller Port List (ALTGX_RECONFIG Instance) (Part 5 of 13) *(Note 3)*, *(4)*

Port Name	Input/ Output	Description
`logical_channel_address [8:0]`	Input	Enabled by the ALTGX_RECONFIG MegaWizard Plug-In Manager when you enable the **Use 'logical_channel_address' port for Analog controls reconfiguration** option in the **Analog controls** screen. The width of the `logical_channel_address` port depends on the value you set in the **What is the number of channels controlled by the reconfig controller?** option in the **Reconfiguration settings** screen. This port can be enabled only when the number of channels controlled by the dynamic reconfiguration controller is more than one. For more information, refer to "Logical Channel Addressing of Regular Transceiver Channels" on page 5–6 and "Logical Channel Addressing of PMA-Only Channels" on page 5–7.
`rx_tx_duplex_sel[1:0]`	Input	This is a 2-bit wide signal. You can select this in the **Error checks/Data rate switch** screen. The advantage of using this optional port is that it allows you to reconfigure only the transmitter portion of a channel, even if the channel configuration is duplex. For a setting of: ■ `rx_tx_duplex_sel[1:0]` = 2'b00—the transmitter and receiver portion of the channel is reconfigured. ■ `rx_tx_duplex_sel[1:0]` = 2'b01—the receiver portion of the channel is reconfigured. ■ `rx_tx_duplex_sel[1:0]` = 2'b10—the transmitter portion of the channel is reconfigured.

Table 5–16. Dynamic Reconfiguration Controller Port List (ALTGX_RECONFIG Instance) (Part 6 of 13) *(Note 3)*, *(4)*

Port Name	Input/ Output	Description
Analog Settings Control/Status Signals		
`tx_vodctrl[2:0]` *(1)*	Input	This is an optional transmit buffer V_{OD} control signal. It is 3 bits per transmitter channel. The number of settings varies based on the transmit buffer supply setting and the termination resistor setting on the **TX Analog** screen of the ALTGX MegaWizard Plug-In Manager. The width of this signal is fixed to 3 bits if you enable either the **Use 'logical_channel_address' port for Analog controls reconfiguration** option or the **Use same control signal for all the channels** option in the **Analog controls** screen. Otherwise, the width of this signal is 3 bits per channel. For more information, refer to "Dynamically Reconfiguring PMA Controls" on page 5–13. The following shows the V_{OD} values corresponding to the `tx_vodctrl` settings for 100-Ω termination. For more information, refer to the "Programmable Output Differential Voltage" section of the *Stratix IV Transceiver Architecture* chapter. `tx_vodctrl[2:0]` — V_{OD} (mV) for 1.4 V V_{CCH} 3'b000 — 200 3'b001 — 400 3'b010 — 600 3'b011 — 700 3'b100 — 800 3'b101 — 900 3'b110 — 1000 3'b111 — 1200

Table 5–16. Dynamic Reconfiguration Controller Port List (ALTGX_RECONFIG Instance) (Part 7 of 13) *(Note 3)*, *(4)*

Port Name	Input/ Output	Description
`tx_preemp_0t[4:0]` *(1)*	Input	This is an optional pre-emphasis control for pre-tap for the transmit buffer. Depending on what value you set at this input, the controller dynamically writes the value to the pre-emphasis control register of the transmit buffer. This signal controls both pre-emphasis positive and its inversion. The width of this signal is fixed to 5 bits if you enable either the **Use 'logical_channel_address' port for Analog controls reconfiguration** option or the **Use same control signal for all the channels** option in the **Analog controls** screen. Otherwise, the width of this signal is 5 bits per channel. For more information, refer to "Dynamically Reconfiguring PMA Controls" on page 5–13. The following values are the legal settings allowed for this signal: 0 represents 0 1-15 represents -15 to -1 16 represents 0 17 - 31 represents 1 to 15 In PCI Express (PIPE) configuration, set `tx_preemp_0t[4:0]` to **5'b00000** when you do a rate switch from Gen 1 to Gen 2 mode. This is to ensure that `tx_preemp_0t[4:0]` does not add to the signal boost, when `tx_pipemargin` and `tx_pipedeemph` take affect in PCI Express (PIPE) Gen 2 mode. For more information, refer to the "Programmable Pre-Emphasis" section of the *Stratix IV Transceiver Architecture* chapter.
`tx_preemp_1t[4:0]` *(1)*	Input	This is an optional pre-emphasis write control for the first post-tap for the transmit buffer. Depending on what value you set at this input, the controller dynamically writes the value to the first post-tap control register of the transmit buffer. The width of this signal is fixed to 5 bits if you enable either the **Use 'logical_channel_address' port for Analog controls reconfiguration** option or the **Use same control signal for all the channels** option in the **Analog controls** screen. Otherwise, the width of this signal is 5 bits per channel. For more information, refer to "Dynamically Reconfiguring PMA Controls" on page 5–13 and the "Programmable Pre-Emphasis" section of the *Stratix IV Transceiver Architecture* chapter.

Table 5–16. Dynamic Reconfiguration Controller Port List (ALTGX_RECONFIG Instance) (Part 8 of 13) *(Note 3)*, *(4)*

Port Name	Input/ Output	Description
`tx_preemp_2t[4:0]` *(1)*	Input	This is an optional pre-emphasis write control for the second post-tap for the transmit buffer. This signal controls both pre-emphasis positive and its inversion. Depending on what value you set at this input, the controller dynamically writes the value to the pre-emphasis control register of the transmit buffer. The width of this signal is fixed to 5 bits if you enable either the **Use 'logical_channel_address' port for Analog controls reconfiguration** option or the **Use same control signal for all the channels** option in the **Analog controls** screen. Otherwise, the width of this signal is 5 bits per channel. For more information, refer to "Dynamically Reconfiguring PMA Controls" on page 5–13. The following values are the legal settings allowed for this signal: 0 represents 0 1-15 represents -15 to -1 16 represents 0 17-31 represents 1 to 15 In PCI Express (PIPE) configuration, set `tx_preemp_2t[4:0]` to **5'b00000** when you do a rate switch from Gen 1 to Gen 2 mode. This is to ensure that `tx_preemp_2t[4:0]` does not add to the signal boost when `tx_pipemargin` and `tx_pipedeemph` take affect in PCI Express (PIPE) Gen 2 mode. For more information, refer to the "Programmable Pre-Emphasis" section of the *Stratix IV Transceiver Architecture* chapter.
`rx_eqctrl[3:0]` *(1)*	Input	This is an optional write control to write an equalization control value for the receive side of the PMA. The width of this signal is fixed to 4 bits if you enable either the **Use 'logical_channel_address' port for Analog controls reconfiguration** option or the **Use same control signal for all the channels** option in the **Analog controls** screen. Otherwise, the width of this signal is 4 bits per channel. For more information, refer to "Dynamically Reconfiguring PMA Controls" on page 5–13 and the "Programmable Equalization and DC Gain" section of the *Stratix IV Transceiver Architecture* chapter.

Table 5–16. Dynamic Reconfiguration Controller Port List (ALTGX_RECONFIG Instance) (Part 9 of 13) *(Note 3)*, *(4)*

Port Name	Input/ Output	Description
`rx_eqdcgain[2:0]` *(1), (2)*	Input	This is an optional equalizer DC gain write control. The width of this signal is fixed to 3 bits if you enable either the **Use 'logical_channel_address' port for Analog controls reconfiguration** option or the **Use same control signal for all the channels** option in the **Analog controls** screen. Otherwise, the width of this signal is 3 bits per channel. For more information, refer to "Dynamically Reconfiguring PMA Controls" on page 5–13. The following values are the legal settings allowed for this signal: 3'b000 => 0 dB 3'b001 => 3 dB 3'b010 => 6 dB 3'b011 => 9 dB 3'b100 => 12 dB All other values => N/A For more information, refer to the "Programmable Equalization and DC Gain" section of the *Stratix IV Transceiver Architecture* chapter.
`tx_vodctrl_out[2:0]`	Output	This is an optional transmit V_{OD} read control signal. This signal reads out the value written into the V_{OD} control register. The width of this output signal depends on the number of channels controlled by the dynamic reconfiguration controller.
`tx_preemp_0t_out[4:0]`	Output	This is an optional pre-tap, pre-emphasis read control signal. This signal reads out the value written by its input control signal. The width of this output signal depends on the number of channels controlled by the dynamic reconfiguration controller.
`tx_preemp_1t_out[4:0]`	Output	This is an optional first post-tap, pre-emphasis read control signal. This signal reads out the value written by its input control signal. The width of this output signal depends on the number of channels controlled by the dynamic reconfiguration controller.
`tx_preemp_2t_out[4:0]`	Output	This is an optional second post-tap pre-emphasis read control signal. This signal reads out the value written by its input control signal. The width of this output signal depends on the number of channels controlled by the dynamic reconfiguration controller.
`rx_eqctrl_out[3:0]`	Output	This is an optional read control signal to read the setting of equalization setting of the ALTGX instance. The width of this output signal depends on the number of channels controlled by the dynamic reconfiguration controller.
`rx_eqdcgain_out[2:0]`	Output	This is an optional equalizer DC gain read control signal. This signal reads out the settings of the ALTGX instance DC gain. The width of this output signal depends on the number of channels controlled by the dynamic reconfiguration controller.

Table 5–16. Dynamic Reconfiguration Controller Port List (ALTGX_RECONFIG Instance) (Part 10 of 13) *(Note 3)*, *(4)*

Port Name	Input/ Output	Description
Transceiver Channel Reconfiguration Control/Status Signals		
`reconfig_mode_sel[3:0]`	Input	Set the following values at this signal to activate the appropriate dynamic reconfiguration mode: 3'b000 = PMA controls reconfiguration mode. This is the default value. 3'b011 = data rate division in transmitter mode 3'b100 = CMU PLL reconfiguration mode 3'b101 = channel and CMU PLL reconfiguration mode 3'b110 = channel reconfiguration with transmitter PLL select mode 3'b111 = central control unit reconfiguration mode The `reconfig_mode_sel` signal is 4 bits wide when you enable Adaptive Equalization control or EyeQ control: 4'b1000 = AEQ control (continuous mode for a single channel) 4'b1001 = AEQ control (one time mode for a single channel) 4'b1010 = AEQ control (power down for a single channel) 4/b1011 = EyeQ control `reconfig_mode_sel[]` is available as an input only when you enable more than one dynamic reconfiguration mode.
`reconfig_address_out[5:0]`	Output	This signal is always available for you to select in the **Channel and TX PLL reconfiguration** screen. This signal is applicable only in the dynamic reconfiguration modes grouped under **Channel and TX PLL select/reconfig** option. This signal represents the current address used by the ALTGX_RECONFIG instance when writing the **.mif** into the transceiver channel. This signal increments by 1, from 0 to the last address, then starts at 0 again. You can use this signal to indicate the end of all the **.mif** write transactions (`reconfig_address_out[5:0]` changes from the last address to 0 at the end of all the **.mif** write transactions).
`reconfig_address_en`	Output	This is an optional signal you can select in the **Channel and TX PLL reconfiguration** screen. This signal is applicable only in dynamic reconfiguration modes grouped under the **Channel and TX PLL select/reconfig** option. The dynamic reconfiguration controller asserts `reconfig_address_en` to indicate that `reconfig_address_out[5:0]` has changed. This signal is asserted only after the dynamic reconfiguration controller completes writing one 16-bit word of the **.mif**.
`reset_reconfig_address`	Input	This is an optional signal you can select in the **Channel and TX PLL reconfiguration** screen. This signal is applicable only in dynamic reconfiguration modes grouped under the **Channel and TX PLL select/reconfig** option. Enable this signal and assert it for one `reconfig_clk` clock cycle if you want to reset the reconfiguration address used by the ALTGX_RECONFIG instance during reconfiguration.

Table 5–16. Dynamic Reconfiguration Controller Port List (ALTGX_RECONFIG Instance) (Part 11 of 13) *(Note 3)*, *(4)*

Port Name	Input/ Output	Description
`reconfig_data[15:0]`	Input	This signal is applicable only in the dynamic reconfiguration modes grouped under the **Channel and TX PLL select/reconfig** option. This is a 16-bit word carrying the reconfiguration information. It is stored in a **.mif** that you must generate. The ALTGX_RECONFIG instance requires that you provide `reconfig_data [15:0]`on every **.mif** write transaction using the `write_all` signal.
`reconfig_address[5:0]`	Input	This port is available for selection only in the **.mif**-based transceiver channel reconfiguration modes. For more information, refer to "Reduced .mif Reconfiguration" on page 5–23.
`rate_switch_ctrl[1:0]`	Input	This signal is available when you select data rate division in transmitter mode. Based on the value you set here, the divide-by setting of the local divider in the transmitter channel gets modified. The legal values for this port are: 2'b00 = Divide by 1 2'b01 = Divide by 2 2'b10 = Divide by 4 2'b11 = Not supported
`rate_switch_out[1:0]`	Input	This signal is available when you select data rate division in transmitter mode. You can read the existing local divider settings of a transmitter channel at this port. The decoding for this signal is listed below: 2'b00 = Division of 1 2'b01 = Division of 2 2'b10 = Division of 4 2'b11= Not supported
`logical_tx_pll_sel`	Input	At this port you specify the identity of the transmitter PLL you want to reconfigure. You can also specify the identity of the transmitter PLL that you want the transceiver channel to listen to. When you enable this signal, the value set at this signal overwrites the `logical_tx_pll` value contained in the **.mif**. The value at this port must be held at a constant logic level until reconfiguration is done.
`logical_tx_pll_sel_en`	Input	If you want to use the `logical_tx_pll_sel` port only under some conditions and use the `logical_tx_pll` value contained in the **.mif** otherwise, enable this optional `logical_tx_pll_sel_en` port. Only when `logical_tx_pll_sel_en` is enabled and set to **1** does the dynamic reconfiguration controller use `logical_tx_pll_sel` to identify the transmitter PLL. The value at this port must be held at a constant logic level until reconfiguration is done.
`channel_reconfig_done`	Output	This signal goes high to indicate that the dynamic reconfiguration controller has finished writing all the words of the **.mif**. The `channel_reconfig_done` signal is automatically de-asserted at the start of a new dynamic reconfiguration write sequence. This signal is applicable only in channel and CMU PLL reconfiguration and channel reconfiguration with transmitter PLL select modes.

Table 5–16. Dynamic Reconfiguration Controller Port List (ALTGX_RECONFIG Instance) (Part 12 of 13) *(Note 3)*, *(4)*

Port Name	Input/ Output	Description
`aeq_fromgxb[7:0]`	Input	The width of this signal depends on the number of channels controlled by the ALTGX_RECONFIG instance. For example, if you select the total number of channels controlled by the ALTGX_RECONFIG instance as follows: 1 ≤Channels ≤4, then the input port `reconfig_fromgxb` = 8 bits 5 ≤Channels ≤8, then the input port `reconfig_fromgxb` = 16 bits 9 ≤Channels ≤12, then the input port `reconfig_fromgxb` = 24 bits This signal is available only when you enable the **AEQ control** option. You must connect this signal between the ALTGX_RECONFIG and ALTGX instances when using AEQ control.
`aeq_togxb`	Output	The width of this signal depends on the number of channels controlled by the ALTGX_RECONFIG instance. For example, if you select the total number of channels controlled by ALTGX_RECONFIG instance as follows: 1 ≤Channels ≤4, then the input port `reconfig_fromgxb` = 24 bits 5 ≤Channels ≤8, then the input port `reconfig_fromgxb` = 48 bits 9 ≤Channels ≤12, then the input port `reconfig_fromgxb` = 64 bits This signal is available only when you enable the **AEQ control** option. You must connect this signal between the ALTGX_RECONFIG and ALTGX instances when using AEQ control.
`ctrl_address[15:0]`	Input	Used for EyeQ control. This port is used to specify the address of the EyeQ interface register for read and write operations.
`ctrl_writedata[15:0]`	Input	Used for EyeQ control. Data present on this port is written to the EyeQ interface register selected using the `ctrl_address` port.
`ctrl_readdata[15:0]`	Output	Used for EyeQ control. Contents of the EyeQ interface register selected using the `ctrl_address` port are available on this port after a read operation.
`ctrl_write`	Input	Used for EyeQ control. Assert this signal high to write the data present on the `ctrl_writedata` port to the EyeQ interface registers.
`ctrl_read`	Input	Used for EyeQ control. Assert this signal high to read the contents of the EyeQ interface registers to the `ctrl_readdata` port.

Table 5–16. Dynamic Reconfiguration Controller Port List (ALTGX_RECONFIG Instance) (Part 13 of 13) *(Note 3)*, *(4)*

Port Name	Input/ Output	Description
`ctrl_waitrequest`	Output	Used for EyeQ control. If asserted, this port indicates that the EyeQ controller is busy with a read or write operation. You must wait until this signal goes low before you perform the next operation. Ensure that the values on the `ctrl_read`, `ctrl_write`, `ctrl_readdata`, and `ctrl_writedata` ports are constant when `ctrl_waitrequest` is asserted.

Notes to Table 5–16:

(1) Not all combinations of the input bits are legal values.

(2) In PCI Express (PIPE) mode, this input must be tied to 001 to be PCI E-compliant.

(3) For the various dynamic reconfiguration controller input and output ports and the software settings, refer to the *ALTGX_RECONFIG Megafunction User Guide* chapter.

(4) For the various transceiver input and output ports and the software settings, refer to the *ALTGX Transceiver Setup Guide* chapter.

Error Indication During Dynamic Reconfiguration

The ALTGX_RECONFIG MegaWizard Plug-In Manager provides an error status signal when you select the **Enable illegal mode checking** option or the **Enable self recovery** option in the **Error checks/data rate switch** screen. The conditions under which the error signal is asserted are:

- **Enable illegal mode checking** option—When you select this option, the dynamic reconfiguration controller checks whether an attempted operation falls under one of the conditions listed below. The dynamic reconfiguration controller detects these conditions within two `reconfig_clk` cycles, de-asserts the `busy` signal, and asserts the error signal for two `reconfig_clk` cycles.
  - PMA controls, read operation—None of the output ports (`rx_eqctrl_out`, `rx_eqdcgain_out`, `tx_vodctrl_out`, `tx_preemp_0t_out`, `tx_preemp_1t_out`, and `tx_preemp_2t_out`) are selected in the ALTGX_RECONFIG instance and the `read` signal is asserted.

- PMA controls, write operation—None of the input ports (`rx_eqctrl`, `rx_eqdcgain`, `tx_vodctrl`, `tx_preemp_0t`, `tx_preemp_1t`, and `tx_preemp_2t`) are selected in the ALTGX_RECONFIG instance and the `write_all` signal is asserted.
- **TX Data Rate Switch using Local Divider-read operation** option—The read transaction is valid only for data rate division in transmitter mode
- **TX Data Rate Switch using Local Divider-write operation with unsupported value** option:
  - The `rate_switch_ctrl` input port is set to **11**
  - The `reconfig_mode_sel` input port is set to **3** (if other reconfiguration mode options are selected in the **Reconfiguration settings** screen)
  - The `write_all` is asserted
- **TX Data Rate Switch using Local Divider-write operation without input port** option:
  - The `rate_switch_ctrl` input port is not used
  - The `reconfig_mode_sel` port is set to **3** (if other reconfiguration mode options are selected in the **Reconfiguration settings** screen)
  - The `write_all` is asserted
- **TX Data Rate Switch using Local Divider- read operation without output port** option:
  - The `rate_switch_out` output port is not used
  - The `reconfig_mode_sel` port is set to **3** (if other reconfiguration mode options are selected in the **Reconfiguration settings** screen)
  - The `read` is asserted
- **Channel and/or TX PLL reconfig/select-read operation** option:
  - The `reconfig_mode_sel` input port is set to **4**, **5**, **6, or 7**
  - The `read` signal is asserted

- **Enable self recovery** option—When you select this option, the controller automatically recovers if the operation did not complete within the expected time. The error signal is driven high whenever the controller performs a self recovery.
- **Adaptive Equalization** option—read operation:
  - `reconfig_mode_sel` input port is et to 7, 8, 9, or 10
  - read signal is asserted
- **EyeQ** option—read operation:
  - `reconfig_mode_sel` input port is et to 11
  - read signal is asserted

Dynamic Reconfiguration Duration

Dynamic reconfiguration duration is the number of cycles the `busy` signal is asserted when the dynamic reconfiguration controller performs write transactions, read transactions, or offset cancellation of the receiver channels.

PMA Controls Reconfiguration Duration

The following section contains an estimate of the number of `reconfig_clk` clock cycles the `busy` signal is asserted during PMA controls reconfiguration using Method 1, Method 2, or Method 3. For more information, refer to "Dynamically Reconfiguring PMA Controls" on page 5–13.

PMA Controls Reconfiguration Duration When Using Method 1

The `logical_channel_address` port is used in Method 1. The write transaction and read transaction duration is as follows:

Write Transaction Duration

For writing values to the following PMA controls, the `busy` signal is asserted for 260 `reconfig_clk` clock cycles for each of these controls:

- `tx_preemp_1t` (pre-emphasis control first post-tap)
- `tx_vodctrl` (voltage output differential)
- `rx_eqctrl` (equalizer control)
- `rx_eqdcgain` (equalizer DC gain)

For writing values to the following PMA controls, the `busy` signal is asserted for 520 `reconfig_clk` clock cycles for each of these controls:

- `tx_preemp_0t` (pre-emphasis control pre-tap)
- `tx_preemp_2t` (pre-emphasis control second post-tap)

Read Transaction Duration

For reading the existing values of the following PMA controls, the `busy` signal is asserted for 130 `reconfig_clk` clock cycles for each of these controls. The `data_valid` signal is then asserted after the `busy` signal goes low.

- `tx_preemp_1t_out` (pre-emphasis control first post-tap)
- `tx_vodctrl_out` (voltage output differential)
- `rx_eqctrl_out` (equalizer control)
- `rx_eqdcgain_out` (equalizer DC gain)

For reading the existing values of the following PMA controls, the `busy` signal is asserted for 260 `reconfig_clk` clock cycles for each of these controls. The `data_valid` signal is then asserted once the `busy` signal goes low.

- `tx_preemp_0t_out` (pre-emphasis control pre-tap)
- `tx_preemp_2t_out` (pre-emphasis control second post-tap)

PMA Controls Reconfiguration Duration When Using Method 2 or Method 3

The `logical_channel_address` port is not used in Method 2 and Method 3. The write transaction duration and read transaction duration are as follows:

Write Transaction Duration

For writing values to the following PMA controls, the `busy` signal is asserted for 260 `reconfig_clk` clock cycles per channel for each of these controls:

- `tx_preemp_1t` (pre-emphasis control first post-tap)
- `tx_vodctrl` (voltage output differential)
- `rx_eqctrl` (equalizer control)
- `rx_eqdcgain` (equalizer DC gain)

For writing values to the following PMA controls, the `busy` signal is asserted for 520 `reconfig_clk` clock cycles per channel for each of these controls:

- `tx_preemp_0t` (pre-emphasis control pre-tap)
- `tx_preemp_2t` (pre-emphasis control second post-tap)

Read Transaction Duration

For reading the existing values of the following PMA controls, the `busy` signal is asserted for 130 `reconfig_clk` clock cycles per channel for each of these controls. The `data_valid` signal is then asserted after the `busy` signal goes low.

- `tx_preemp_1t_out` (pre-emphasis control first post-tap)
- `tx_vodctrl_out` (voltage output differential)
- `rx_eqctrl_out` (equalizer control)
- `rx_eqdcgain_out` (equalizer DC gain)

For reading the existing values of the following PMA controls, the `busy` signal is asserted for 260 `reconfig_clk` clock cycles per channel for each of these controls. The `data_valid` signal is then asserted after the `busy` signal goes low.

- `tx_preemp_0t_out` (pre-emphasis control pre-tap)
- `tx_preemp_2t_out` (pre-emphasis control second post-tap)

Offset Cancellation Duration

When the device powers up, the `busy` signal remains low for the first `reconfig_clk` clock cycle. Offset cancellation control is only for the receiver channels. The ALTGX_RECONFIG instance takes approximately 18307 `reconfig_clk` clock cycles per channel for **Receiver only** and **Receiver and Transmitter** channels. It takes approximately 877 `reconfig_clk` clock cycles per channel for **Transmitter only** channels to determine if the channel under reconfiguration is a receiver channel or not. The ATLGX_RECONFIG requires an add it on al 130,000 clock cycles for these values to take effect. The ALTGX_RECONFIG instance takes approximately two `reconfig_clk` clock cycles per channel for the unused logical channels.

To demonstrate offset cancellation duration, consider the following example:

- One ALTGX_RECONFIG instance is connected to two ALTGX instances.
- ALTGX Instance 1 has one **Transmitter only** channel (logical_channel_address = 0)
- ALTGX Instance 2 has one **Receiver only** channel (logical_channel_address = 4)

For this example, the ALTGX_RECONFIG instance consumes the following number of `reconfig_clk` clock cycles for offset cancellation:

- 877 cycles for the **Transmitter only** channel
- 18307 cycles for the **Receiver only** channel
- 2 cycles each for non-existent channels with logical_channel_addresses = 1, 2, and 3. 130000 cycles as a baseline for the values to take affect.

The offset cancellation duration for the ALTGX_RECONFIG instance to reconfigure the **Transmitter only** channel, **Receiver only** channel, non-existent logical channels 1, 2, and 3 = 149190 cycles (877 +18307 +6 + 130000).

Dynamic Reconfiguration Duration for Channel and Transmitter PLL Select/Reconfig Modes

Table 5–17 lists the number of `reconfig_clk` clock cycles it takes for the dynamic reconfiguration controller to reconfigure various parts of the transceiver channel and CMU PLL.

Table 5–17. Dynamic Reconfiguration Duration for Transceiver Channel and CMU PLL Reconfiguration

Transceiver Portion Under Reconfiguration	Number of reconfig_clk Clock Cycles
Transmitter channel reconfiguration	1518 clock cycles
Receiver channel reconfiguration	5255 clock cycles
Transmitter and receiver channel reconfiguration	6762 clock cycles
CMU PLL only reconfiguration	863 clock cycles
Transmitter channel and CMU PLL reconfiguration	2370 clock cycles
Transceiver channel and CMU PLL reconfiguration	7614 clock cycles

Dynamic Reconfiguration (ALTGX_RECONFIG Instance) Resource Utilization

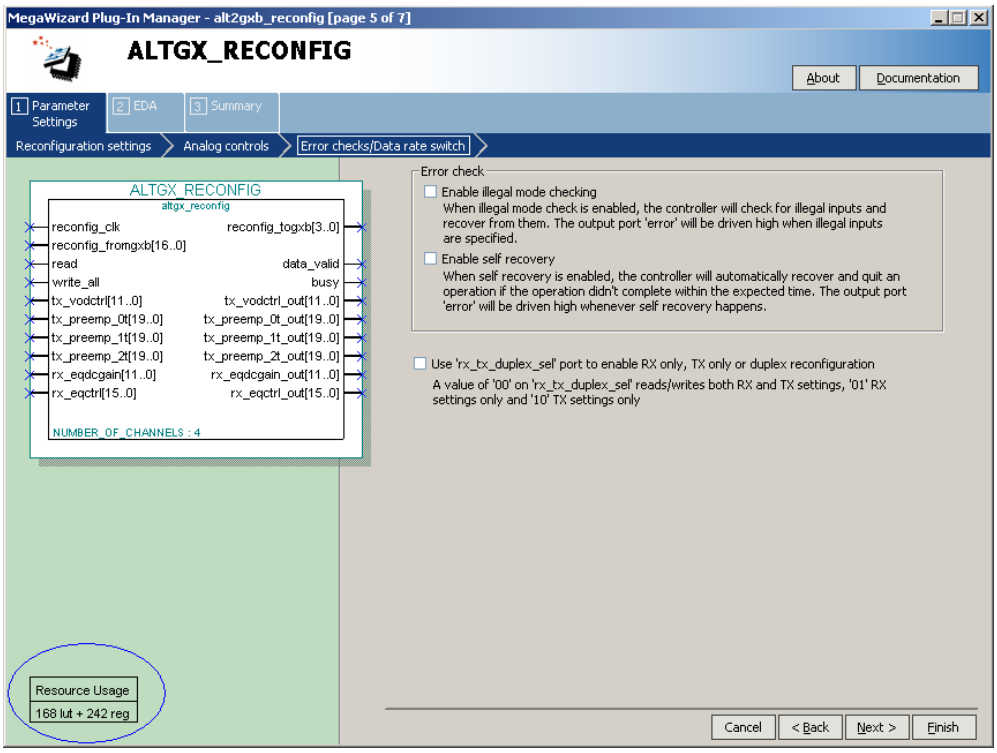
You can observe the resources used during dynamic reconfiguration in the ALTGX_RECONFIG MegaWizard Plug-In Manager. This section contains an estimate of the LE resources used during dynamic reconfiguration.

You can obtain resource utilization for all other PMA controls from the ALTGX_RECONFIG MegaWizard Plug-In Manager.

For example, the number of LEs used by one dynamic reconfiguration controller is 43 with only `tx_vodctrl` selected and the number of registers is 130.

Figure 5–41 shows resource utilization in the ALTGX_RECONFIG MegaWizard Plug-In Manager.

Figure 5–41. Resource Utilization in the ALTGX_RECONFIG MegaWizard Plug-In Manager



Functional Simulation of the Dynamic Reconfiguration Process

This section describes the points to be considered during functional simulation of the dynamic reconfiguration process.

- You must connect the ALTGX_RECONFIG instance to the ALTGX_instance/ALTGX instances in your design for functional simulation.
- The functional simulation uses a reduced timing model of the dynamic reconfiguration controller. Therefore, the duration of the dynamic reconfiguration process is 16 `reconfig_clk` clock cycles for functional simulation only.
- The `gxb_powerdown` signal must not be asserted during the offset cancellation sequence (for functional simulation and silicon).

Dynamic Reconfiguration Examples

The following examples help to describe the dynamic reconfiguration feature.

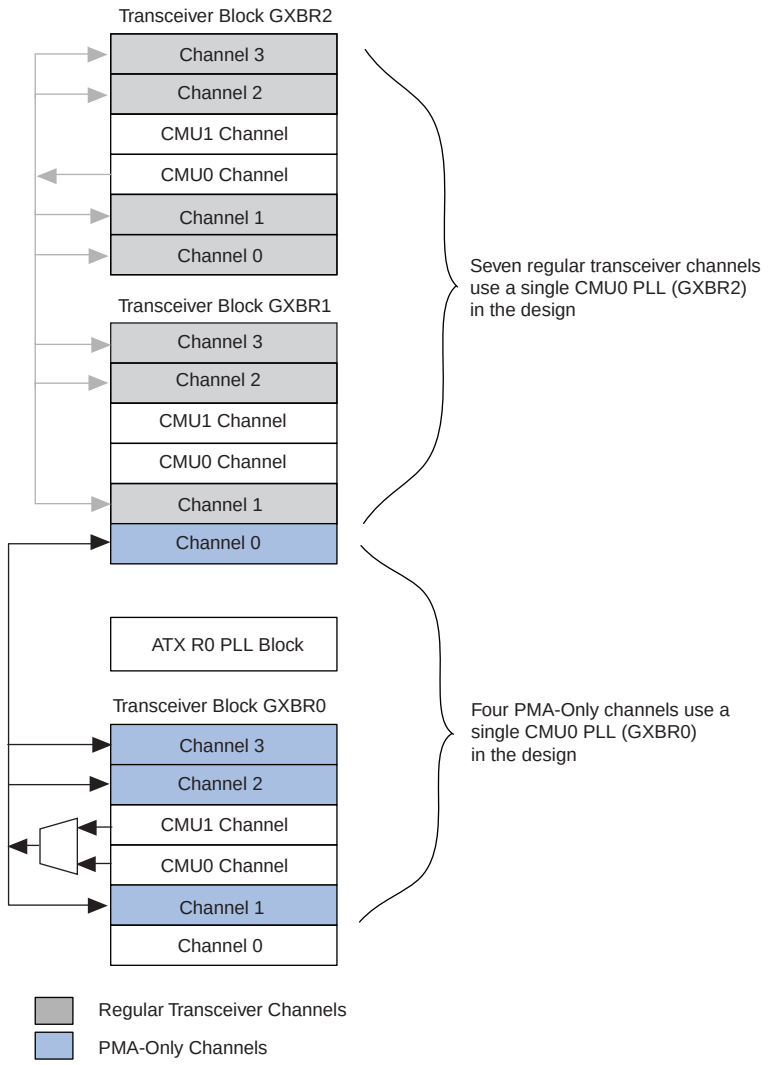
Example 1

Consider a design with the following configuration:

- Seven regular transceiver channels in Basic functional mode. You can configure the seven regular transceiver channels from 2.5 Gbps to 5 Gbps and vice versa using a single CMU.
- Four channels in Basic (PMA Direct) functional mode. You can reconfigure the four PMA-only channels from 3.125 Gbps to 5 Gbps and vice versa.
- You can reconfigure the PMA controls for any one of these channels.

Figure 5–42 shows the arrangement of these channels in the S4GX230 device.

Figure 5–42. Dynamic Reconfiguration Configuration for the S4GX230 Device (Example 1)



Section 1

Because this example does not require the use of the alternate CMU transmitter PLL or additional transmitter PLLs, the logical channel addressing remains the same as explained in "Logical Channel Addressing" on page 5–5.

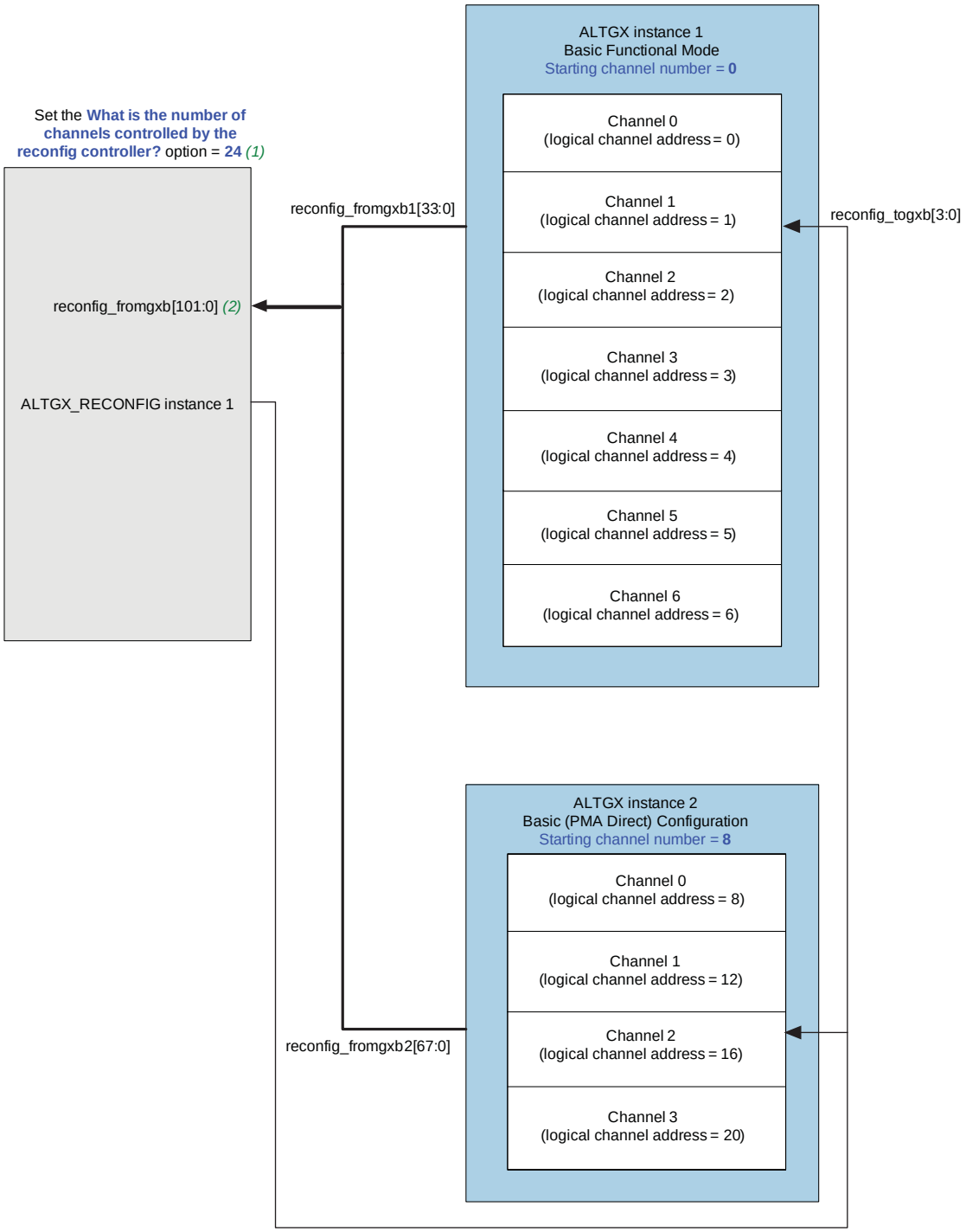
Table 5–18 lists how to set the starting channel number in the ALTGX MegaWizard Plug-In Manager, the total number of channels in the ALTGX_RECONFIG MegaWizard Plug-In Manager, and how to connect the ALTGX instances to the ALTGX_RECONFIG instance.

Table 5–18. Logical Channel Addressing combination of Regular Transceiver Channels and PMA-Only Channels (Example 1)

ALTGX Settings and Instances			ALTGX_RECONFIG Setting and Instance	
ALTGX Setting	**ALTGX Instance 1 (Basic Functional Mode)**	**ALTGX Instance 2 (Basic [PMA Direct] Functional Mode)**	**ALTGX_RECONFIG Setting**	**ALTGX_RECONFIG Instance 1**
What is the number of channels? option in the **General** screen	**7** (Regular Transceiver Channels)	**4** (PMA-Only Channels)	**What is the number of channels controlled by the reconfig controller?** option in the **Reconfiguration settings** screen.	■ Determine the highest logical channel address (20). ■ Round it up to the next multiple of 4. ■ Set this option to **24**.
What is the starting channel number? option in the **Reconfig** screen	■ Set this option to **0**. ■ The logical channel addresses of the first to sixth channels are **0**, **1**, **2**, **3**, **4**, **5**, and **6**, respectively.	■ Set this option to **8**. This is because the starting channel numbers 0 and 4 have already been used in ALTGX instance 1. ■ The logical channel addresses of the first to fourth channels are **8**, **12**, **16**, and **20**, respectively.	—	—
`reconfig_fromgxb1` and `reconfig_fromgxb2` outputs	`reconfig_fromgxb1` is 34 bits wide (**2** * 17)	`reconfig_fromgxb2` is 68 bits wide (**4** * 17)	`reconfig_fromgxb` input	`reconfig_fromgxb` is 102 bits wide (**24** regular transceiver channels can logically fit into 6 transceiver blocks; 6 * 17 = 102)

Figure 5–43 shows how the logical channel addresses of all the channels are set, based on what you set as the starting channel number.

Figure 5–43. Logical Channel Addresses for Example 1



Notes to Figure 5–43:

(1) For more information, refer to "Total Number of Channels Option in the ALTGX_RECONFIG Instance" on page 5–10.

(2) `reconfig_fromgxb[101:0] = { reconfig_fromgxb2[67:0], reconfig_fromgxb1[33:0] }`

Section 2

This section describes the details related to design implementation.

Different Dynamic Reconfiguration Modes Involved

1. Channel and CMU PLL reconfiguration mode:
   - is used for reconfiguring the seven regular transceiver channels from one data rate to another using the same `CMU0` PLL (in GXBR2)

     ☞ This mode is chosen because both the receiver and transmitter of the regular channels must be re-configured using a single CMU.

2. Channel and CMU PLL select reconfiguration mode:
   - is used for reconfiguring the four PMA-Only channels from one data rate to another using `CMU0` PLL (in GXBR0) and `CMU1` PLL (GXBR0)

     ☞ This mode is chosen because both the receiver and transmitter of the regular channels must be re-configured and more than one CMU can be used.

3. The `rx_tx_duplex_sel[1:0]` port allows you to reconfigure the transmitter and receiver channels to operate at the different data rates.
4. PMA controls reconfiguration mode used to configure the PMA settings for all the channels.

For more information, refer to "Transceiver Channel Reconfiguration Mode Details" on page 5–19.

.mif Generation

The following **.mifs** are required for this example:

- For the seven regular transceiver channels, you will need to generate two **.mifs**. One to move from a data rate of 2.5 Gbps to 5 Gbps and the other revert back to 2.5 Gbps.
- For the for PMA-Only channels, you will need to generate two **.mifs**. One to move from a data rate of 3.125 Gbps to 5 Gbps and the other revert back to 3.125 Gbps.

For more information, refer to "Memory Initialization File (.mif)" on page 5–19.

Various Dynamic Reconfiguration Transactions

The following dynamic reconfiguration transactions are required "Example 1" on page 5–96:

- **.mif** write transaction—for more information, refer to "Channel and CMU PLL Reconfiguration Mode Details" on page 5–24 and "Channel Reconfiguration with Transmitter PLL Select Mode Details" on page 5–48.
- Reconfiguring PMA controls—for more information, refer to "Dynamically Reconfiguring PMA Controls" on page 5–13.

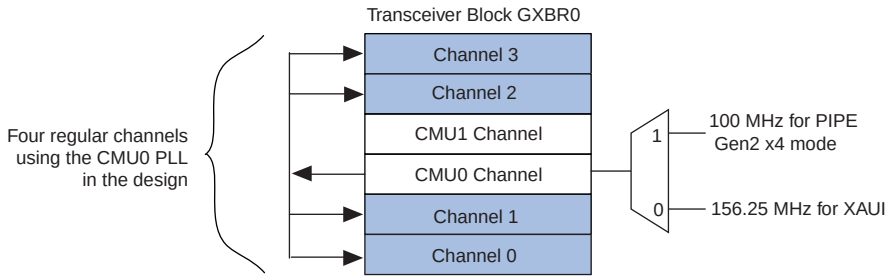
Example 2

Consider a design with the following configuration:

- Four regular transceiver channels in XAUI configuration.
- You can configure these channels from the XAUI configuration (the primary configuration) to the PCI Express (PIPE) Gen2 ×4 configuration (the secondary configuration) and vice versa.

Figure 5–44 shows the arrangement of these channels in the S4GX230 device.

Figure 5–44. Dynamic Reconfiguration Configuration for the S4GX230 Device (Example 2)



Section 1

Because this example does not require the use of the alternate CMU transmitter PLL or additional transmitter PLLs, the logical channel addressing remains the same as explained in "Logical Channel Addressing" on page 5–5.

Table 5–19 lists how to set the starting channel number in the ALTGX MegaWizard Plug-In Manager, the total number of channels in the ALTGX_RECONFIG MegaWizard Plug-In Manager, and how to connect the ALTGX instances to the ALTGX_RECONFIG instance.

Table 5–19. Logical Channel Addressing combination ×4 Bonded Channels (Example 2) (Part 1 of 2)

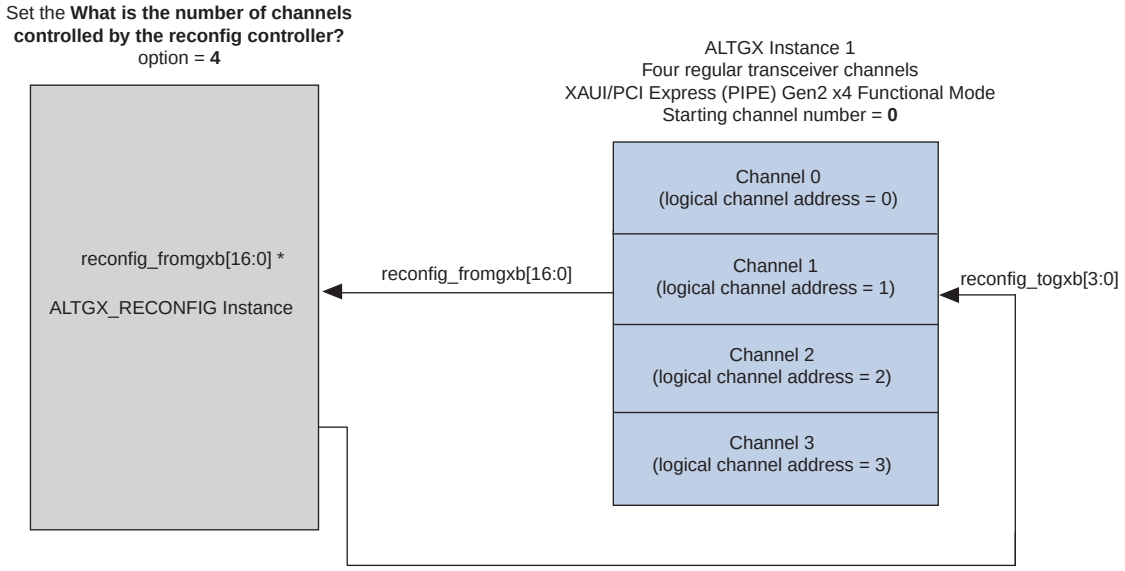
ALTGX Settings and Instances		ALTGX_RECONFIG Setting and Instance	
ALTGX Setting	**ALTGX Instance 1 (XAUI Mode)**	**ALTGX_RECONFIG Setting**	**ALTGX_RECONFIG Instance 1**
What is the number of channels? option in the **General** screen	**4** (Regular transceiver channels)	**What is the number of channels controlled by the reconfig controller?** option in the **Reconfiguration settings** screen.	■ Determine the highest logical channel address (3). ■ Round it up to the next multiple of 4. ■ Set this option to **4**.
What is the starting channel number? option in the **Reconfig** screen	■ Set this option to **0**. ■ The logical channel addresses of the first to sixth channels are **0**, **1**, **2**, and **3**, respectively.	—	—

Table 5–19. Logical Channel Addressing combination ×4 Bonded Channels (Example 2) (Part 2 of 2)

ALTGX Settings and Instances		ALTGX_RECONFIG Setting and Instance	
ALTGX Setting	**ALTGX Instance 1 (XAUI Mode)**	**ALTGX_RECONFIG Setting**	**ALTGX_RECONFIG Instance 1**
Settings in **Reconfiguration settings** page	■ Enable **Channel and Transmitter PLL reconfiguration** ■ Enable **Channel interface** ■ Set **2** for the **How many clock inputs are used?** option. ■ Set **0** for the XAUI ALTGX instance and ■ Set **1** for the PCI Express (PIPE) Gen2 x4 ALTGX instance for the **What is the selected input clock source for Tx/Rx PLLs?** option.	—	—
`reconfig_fromgxb1` and `reconfig_fromgxb2` outputs	`reconfig_fromgxb1` is 17 bits wide	`reconfig_fromgxb` input	`reconfig_fromgxb` is 17 bits wide (**4** regular transceiver channels can logically fit into 1 transceiver blocks; 1 * 17 = 17)

Figure 5–45 shows how the logical channel addresses of all the channels are set, based on what you set as the starting channel number.

Figure 5–45. Logical Channel Addresses for Example 2



Section 2

This section describes the details related to design implementation.

Different Dynamic Reconfiguration Modes Involved

1. Channel and CMU PLL reconfiguration mode—used for reconfiguring four regular transceiver channels and `CMU0` PLL (in GXBR0) from XAUI mode to PCI Express (PIPE) ×4 mode and vice versa.

   ☞ Use this mode instead of channel reconfiguration with transmitter PLL select mode because the central clock divider used for bonded modes is only available in `CMU0`; therefore, you cannot use `CMU1` as an alternate TX PLL.

2. Central control unit reconfiguration mode—used for reconfiguring central control unit logic used in bonded modes from XAUI mode to PCI Express (PIPE) ×4 mode.

For more information, refer to "Transceiver Channel Reconfiguration Mode Details" on page 5–19.

.mif Generation

The following **.mifs** are required for this example:

- One **.mif** is needed to move from XAUI mode to PCI Express (PIPE) ×4 mode
- Another **.mif** is needed to revert back to XAUI mode from PCI Express (PIPE) ×4 mode

For more information, refer to "Memory Initialization File (.mif)" on page 5–19.

Various Dynamic Reconfiguration Transactions

The following dynamic reconfiguration transactions are required for this example:

- **.mif** write transaction—for more information, refer to "Channel and CMU PLL Reconfiguration Mode Details" on page 5–24.
- Alternatively, you may use reduced **.mif** reconfiguration. Reduced **.mifs** are generated using the **altgx_diffmifgen.exe** command. For more information, refer to "Reduced .mif Reconfiguration" on page 5–23.

Document Revision History

Table 5–20 shows the revision history for this chapter.

Table 5–20. Document Revision History (Part 1 of 2)

Date and Document Version	Changes Made	Summary of Changes
November 2009, v3.0	■ Completely re-wrote and re-organized chapter. ■ Updated all graphics and tables.	—

Table 5–20. Document Revision History (Part 2 of 2)

June 2009, v2.1	■ Updated Figure 5–4, Figure 5–8, Figure 5–9, Figure 5–10, Figure 5–11, Figure 5–15, Figure 5–22, Table 5–37, Table 5–38, Figure 5–44, Figure 5–47, Figure 5–48, Figure 5–49, Figure 5–50, Figure 5–51, Figure 5–52, Figure 5–53, and Figure 5–54 ■ Updated Table 5–2 and Table 5–31 ■ Changed “logical_tx_pll_sel[1:0]” to “logical_tx_pll_sel” throughout ■ Updated “The reconfig_clk Clock Requirements for the ALTGX Instance and ALTGX_RECONFIG Instance”, “The logical_tx_pll_sel and logical_tx_pll_sel_en Ports”, “How to Use the logical_tx_pll_sel Port?”, and “When Can the logical_tx_pll_sel and logical_tx_pll_sel_en Ports be Used?” ■ Minor text edits	—
March 2009, v2.0	Complete re-write and re-organization of the chapter. Added or revised: ■ Offset Cancellation Control for Receiver Channels ■ PMA Controls Reconfiguration ■ Channel and CMU PLL Reconfiguration Mode ■ Data Rate Division in Transmitter: Operation ■ Channel Reconfiguration with Transmitter PLL Select Mode ■ CMU PLL Reconfiguration Mode	—
November 2008, v1.0	Initial release	—

Stratix IV Device Handbook

Volume 3

101 Innovation Drive
San Jose, CA 95134
www.altera.com

SIV5V3-4.0

Copyright © 2009 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

I.S. EN ISO 9001

Contents

Chapter Revision Dates v

Additional Information
About this Handbook . . . . . Info-vii
How to Contact Altera . . . . . Info-vii
Typographic Conventions . . . . . Info-vii

Section I. Transceiver Configuration Guide
Revision History I-1

Chapter 1. ALTGX Transceiver Setup Guide
Parameter Settings 1-3
General Screen for the Parameter Settings 1-3
PLL/Ports Screen for the Parameter Settings 1-15
Ports/Calibration Screen for the Parameter Settings 1-19
Loopback Screen for the Parameter Settings 1-22
RX Analog Screen for the Parameter Settings 1-24
TX Analog Screen for the Parameter Settings 1-26
Reconfiguration Settings 1-28
Modes Screen for the Reconfiguration Settings 1-28
Transmitter PLL Settings 1-31
Clocking/Interface Screen for the Reconfiguration Settings 1-34
Protocol Settings 1-35
8B10B Screen for the Protocol Settings 1-36
Word Aligner Screen for the Protocol Settings 1-39
Rate Match/Byte Order Screen for the Protocol Settings 1-44
Protocol Settings Screen for GIGE and XAUI 1-47
Protocol Settings Screen for the (OIF) CEI Phy Interface 1-50
Protocol Settings Screen for PCI Express (PIPE) 1-51
Protocol Settings Screen for SONET/SDH 1-57
EDA Screen 1-61
Summary Screen 1-62
Document Revision History 1-63

Chapter 2. Transceiver Design Flow Guide
Architecture 2-3
Device Specification 2-3
Transceiver Configuration 2-3
Dynamic Reconfiguration 2-4
Clocking 2-5
Power Supplies 2-6
Board Design Requirements 2-6

Implementation and Integration 2-7
Create Transceiver Instances 2-7
Create Dynamic Reconfiguration Controller Instances 2-8
Create Reset and Control Logic 2-8
Create Data Processing and Other User Logic 2-8
PPM Detector When the Receiver CDR Is Used in Manual Lock Mode 2-8
Synchronization State Machine in Manual Word Alignment Mode 2-9
Gear Boxing Logic 2-9
Integrate the Design 2-10
Compilation 2-10
Report Files 2-11
Fitter Summary 2-11
Pin-Out File 2-11
Resource Section 2-11
Verification 2-12
Functional Simulation 2-12
Guidelines to Debug Transceiver-Based Designs 2-15
Guidelines to Debug the FPGA Logic and the Transceiver Interface 2-15
Guidelines to Debug System Level Issues 2-16
Example 1: Fibre Channel Protocol Application 2-17
Phase 1—Architecture 2-18
Device Specification 2-18
Transceiver Configuration 2-18
Dynamic Reconfiguration 2-19
Clocking 2-19
Phase 2—Implementation 2-21
Create the Transceiver Instance for an FC4G Configuration (Channel 0) 2-21
Create the Transceiver Instance for an FC1G Configuration (Channel 1) 2-30
Create the Instance for an FC4G Configuration—Transmitter Only Mode (Channel 2) 2-31
Create the Dynamic Reconfiguration Controller (ALTGX_Reconfig) Instance 2-33
Create Reset Logic to Control the FPGA Fabric and Transceivers 2-34
Create Data Processing and Other User Logic 2-36
Phase 3—Compilation 2-36
Phase 4—Simulating the Design 2-36
Document Revision History 2-36

Chapter 3. Stratix IV ALTGX_RECONFIG Megafunction User Guide

Dynamic Reconfiguration 3-1
Document Revision History 3-14

Chapter Revision Dates

The chapters in this book, *Stratix IV Device Handbook Volume 3*, were revised on the following dates. Where chapters or groups of chapters are available separately, part numbers are listed.

Chapter 1 ALTGX Transceiver Setup Guide
Revised: *November 2009*
Part Number: *SIV53001-4.0*

Chapter 2 Transceiver Design Flow Guide
Revised: *November 2009*
Part Number: *SIV53002-4.0*

Chapter 3 Stratix IV ALTGX_RECONFIG Megafunction User Guide
Revised: *November 2009*
Part Number: *SIV53004-3.0*

About this Handbook

This handbook provides comprehensive information about the Altera® Stratix® IV family of devices.

How to Contact Altera

For the most up-to-date information about Altera products, see the following table.

Contact *(Note 1)*	Contact Method	Address
Technical support	Website	www.altera.com/support
Technical training	Website	www.altera.com/training
	Email	custrain@altera.com
Product literature	Website	www.altera.com/literature
Non-technical support (General)	Email	nacomp@altera.com
(Software Licensing)	Email	authorization@altera.com

Note:

(1) You can also contact your local Altera sales office or sales representative.

Typographic Conventions

The following table shows the typographic conventions that this document uses.

Visual Cue	Meaning
Bold Type with Initial Capital Letters	Indicates command names, dialog box titles, dialog box options, and other GUI labels. For example, **Save As** dialog box. For GUI elements, capitalization matches the GUI.
bold type	Indicates directory names, project names, disk drive names, file names, file name extensions, dialog box options, software utility names, and other GUI labels. For example, **\qdesigns** directory, **d:** drive, and **chiptrip.gdf** file.
Italic Type with Initial Capital Letters	Indicates document titles. For example, *AN 519: Stratix IV Design Guidelines.*
Italic type	Indicates variables. For example, $n + 1$. Variable names are enclosed in angle brackets (< >). For example, *<file name>* and *<project name>***.pof** file.
Initial Capital Letters	Indicates keyboard keys and menu names. For example, Delete key and the Options menu.
“Subheading Title”	Quotation marks indicate references to sections within a document and titles of Quartus II Help topics. For example, “Typographic Conventions.”

Visual Cue	Meaning
`Courier type`	Indicates signal, port, register, bit, block, and primitive names. For example, `data1`, `tdi`, and `input`. Active-low signals are denoted by suffix n. For example, `resetn`. Indicates command line commands and anything that must be typed exactly as it appears. For example, `c:\qdesigns\tutorial\chiptrip.gdf`. Also indicates sections of an actual file, such as a Report File, references to parts of files (for example, the AHDL keyword `SUBDESIGN`), and logic function names (for example, `TRI`).
1., 2., 3., and a., b., c., and so on.	Numbered steps indicate a list of items when the sequence of the items is important, such as the steps listed in a procedure.
■ ■	Bullets indicate a list of items when the sequence of the items is not important.
☞	The hand points to information that requires special attention.
CAUTION	A caution calls attention to a condition or possible situation that can damage or destroy the product or your work.
WARNING	A warning calls attention to a condition or possible situation that can cause you injury.
↵	The angled arrow instructs you to press **Enter**.
👣	The feet direct you to more information about a particular topic.

Section I. Transceiver Configuration Guide

This section includes the following chapters:

- Chapter 1, ALTGX Transceiver Setup Guide
- Chapter 2, Transceiver Design Flow Guide
- Chapter 3, Stratix IV ALTGX_RECONFIG Megafunction User Guide

Revision History

Refer to each chapter for its own specific revision history. For information on when each chapter was updated, refer to the Chapter Revision Dates section, which appears in the full handbook.

1. ALTGX Transceiver Setup Guide

SIV53001-4.0

This chapter describes the options you can choose in the ALTGX MegaWizard™ Plug-In Manager in the Quartus® II software to configure Stratix® IV GX and GT devices in different functional modes.

The MegaWizard Plug-In Manager in the Quartus II software creates or modifies design files that contain custom megafunction variations that can then be instantiated in a design file. The MegaWizard Plug-In Manager provides a MegaWizard that allows you to specify options for the ALTGX megafunction. You can use the MegaWizard Plug-In Manager to set the ALTGX megafunction features in the design. The ALTGX megafunction allows you to configure one or more transceiver channels. You can select the physical coding sublayer (PCS) and physical medium attachment (PMA) functional blocks depending on your transceiver configuration.

This chapter contains the following sections:

- "Parameter Settings" on page 1–3
- "Reconfiguration Settings" on page 1–28
- "Protocol Settings" on page 1–35

Start the MegaWizard Plug-In Manager using one of the following methods:

- From the Tools menu, select **MegaWizard Plug-In Manager**.
- When working in the Block Editor, click **MegaWizard Plug-In Manager** in the **Symbol** dialog box (Edit menu).
- Start the stand-alone version of the MegaWizard Plug-In Manager by typing the following command at the command prompt: `qmegawiz`.

Figure 1–1 shows the first page of the MegaWizard Plug-In Manager. To generate an ALTGX custom megafunction variation, select **Create a new custom megafunction variation**.

Figure 1–1. MegaWizard Plug-In Manager (Page 1)

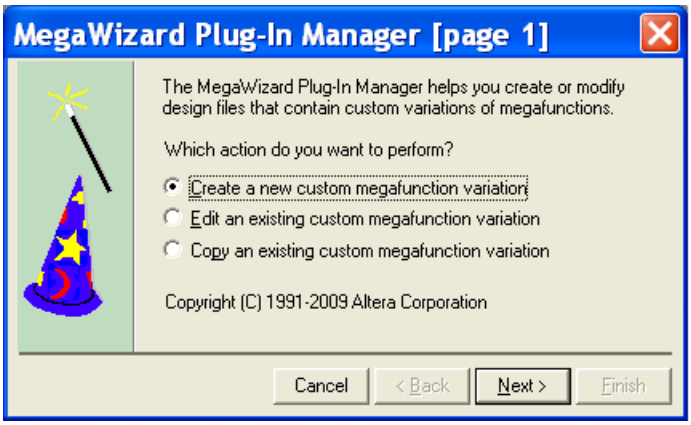
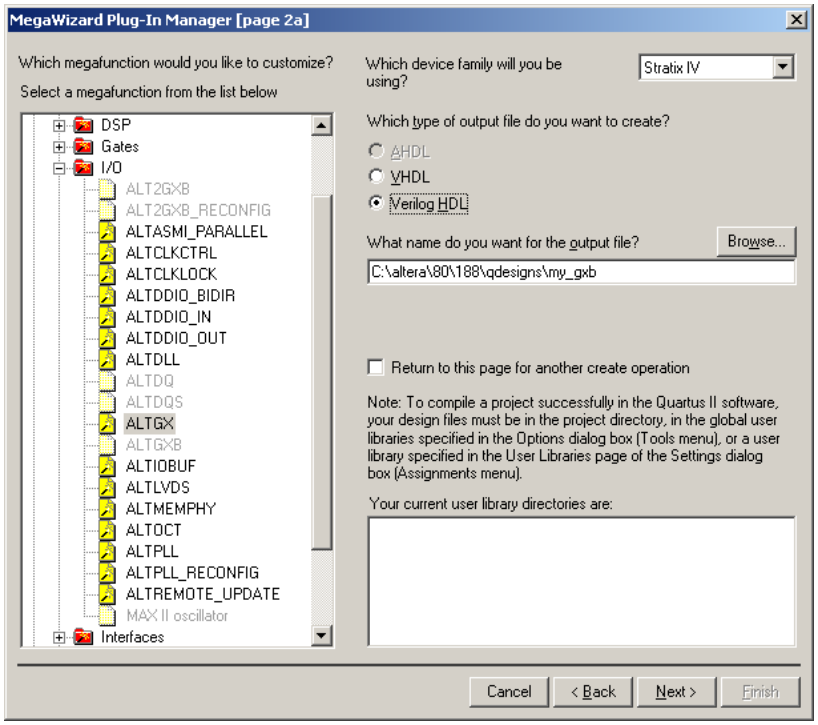


Figure 1–2 shows the second page of the MegaWizard Plug-In Manager.

To use the MegaWizard Plug-In Manager to configure a Stratix IV device:

1. Select **Stratix IV** as the device family.
2. Select either **VHDL** or **Verilog HDL** depending on the type of output files you want to create.
3. Select the **ALTGX** megafunction under the I/O section of the available megafunctions.
4. Name the output file, and then **Browse** to the folder you want to save your file in and click **Next**. The **General** screen of the ALTGX MegaWizard Plug-In Manager opens (Figure 1–3).

Figure 1–2. MegaWizard Plug-In Manager (Page 2)



All reset and control signals are active high unless otherwise mentioned.

All output ports are synchronous to the data path unless otherwise specified.

Throughout this chapter, the various functional modes and their settings are explained for Stratix IV GX and GT devices.

Parameter Settings

This section describes the options available on the individual pages of the ALTGX MegaWizard Plug-In Manager for the Parameter Settings. The MegaWizard Plug-In Manager provides a warning if any of the settings you choose are illegal.

General Screen for the Parameter Settings

Figure 1–3 shows the **General** screen of the ALTGX MegaWizard Plug-In Manager for the Parameter Settings.

Figure 1–3. MegaWizard Plug-In Manager—ALTGX (General Screen for the Parameter Settings)

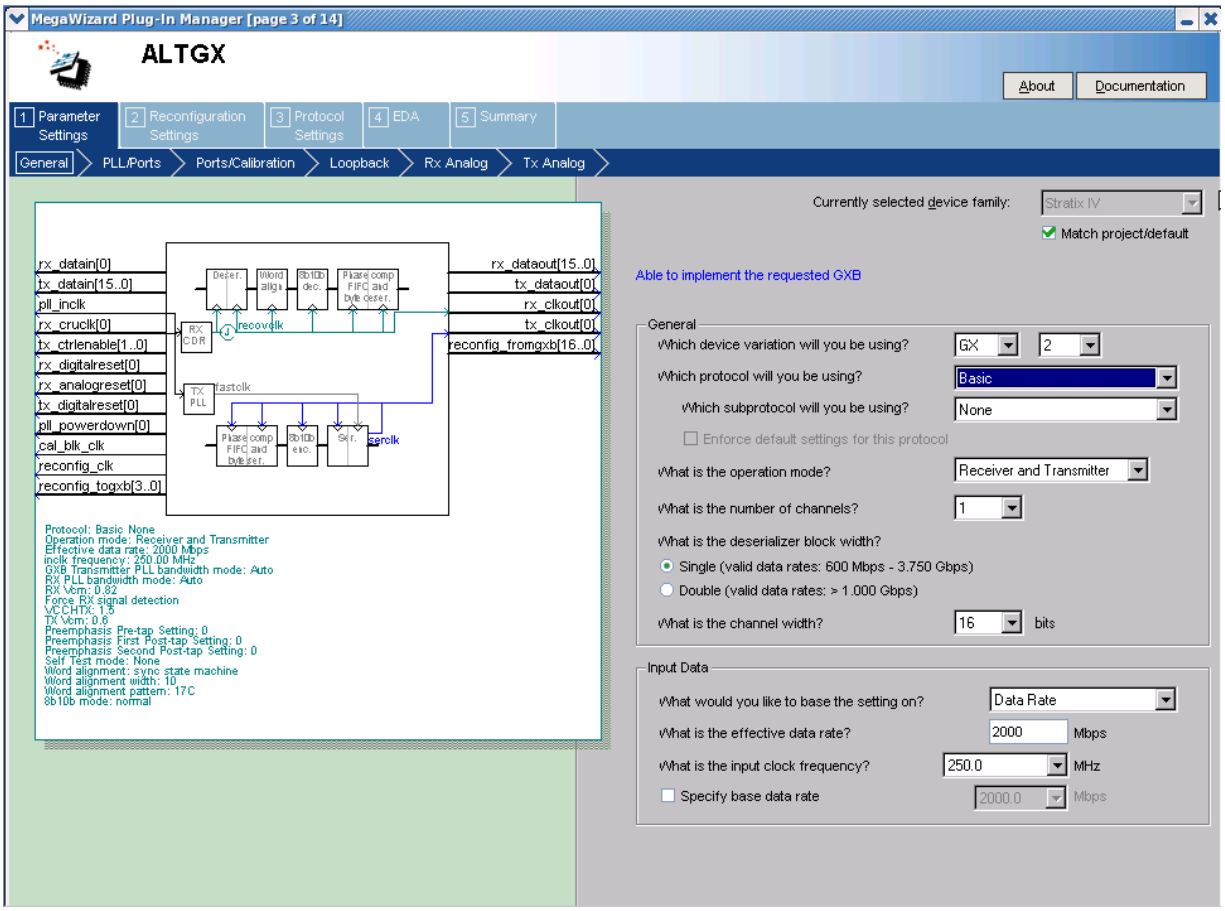


Table 1–1 describes the available functional modes and their options on the **General** screen of the MegaWizard Plug-In Manager. Depending on your configuration, you will select one of the following functional modes:

- **Basic**
- **Basic (PMA Direct)**
- **Deterministic Latency**
- **GIGE**
- **(OIF) CEI Phy Interface**
- **PCI Express (PIPE)**
- **SDI**
- **Serial RapidIO**
- **SONET/SDH**
- **XAUI**

If you select Basic (PMA Direct) mode, all the channels are configured with only the PMA blocks. These channels are called PMA-only channels throughout this chapter. The PMA-only channels include:

- Regular transceiver channels with PMA blocks only
- CMU channels (clock multiplier unit phase-locked loops [CMU PLLs] configured as additional transceiver channels with PMA blocks only)

Table 1–1. MegaWizard Plug-In Manager Options (General Screen for Basic Mode) (Part 1 of 10)

ALTGX Setting	Description	Reference
Which device variation will you be using?	Select **GX** or **GT** based on the Stratix IV device used in your design. Select the speed grade of your device. The available speed grades for the Stratix IV GX device are 2, 2×, 3, and 4. Based on the speed grade you select, the corresponding Stratix IV device can operate at the following maximum speeds: ■ -2 => 8.5 Gbps ■ -2×, 3 => 6.5 Gbps ■ -4 => 5 Gbps The available speed grades for the Stratix IV GT device are 1, 2 and 3. Refer to Datasheet for the supported data rates.	Table 1-22 and Table 1-23 in the *DC and Switching Characteristics* chapter. *Stratix IV Device Datasheet* section.
Which protocol will you be using?	Determines the specific protocol under which the transceiver operates. For a specific mode, you must select the desired protocol from the following list: ■ **Basic** ■ **Basic (PMA Direct)** ■ **Deterministic Latency** ■ **GIGE** ■ **(OIF) CEI PHY Interface** ■ **PCI Express (PIPE)** ■ **Serial RapidIO**	Table 1-1 in the *Stratix IV Transceiver Architecture* chapter in volume 2 of the *Stratix IV Device Handbook.*
Which protocol will you be using?	Determines the specific protocol under which the transceiver operates. For a specific mode, you must select the desired protocol from the following list: ■ **(OIF) CEI PHY Interface** ■ **SDI** ■ **SONET/SDH** ■ **XAUI**	Table 1-2 in the *Stratix IV Transceiver Architecture* chapter in volume 2 of the *Stratix IV Device Handbook.*

Table 1–1. MegaWizard Plug-In Manager Options (General Screen for Basic Mode) (Part 2 of 10)

ALTGX Setting	Description	Reference
Which subprotocol will you be using?	**Basic** In Basic mode, the subprotocols are diagnostic modes. The available options are as follows: ■ None—This is the normal operation of the transceiver. ■ ×4—In this mode, all four channels within the transceiver block are clocked from its central clock divider block to minimize transmitter channel-to-channel skew. ■ ×8—In this mode, all eight channels in two transceiver blocks are clocked from the central clock divider of the master transceiver block to minimize transmitter channel-to-channel skew. ■ BIST—This subprotocol is applicable only for **Receiver and Transmitter** operation mode. This mode loops the parallel data from the built-in self test (BIST) (non-PRBS) back to the BIST verifier in the receiver path. Parallel loopback is allowed only in Basic double-width mode. ■ PRBS—This subprotocol is applicable only for **Receiver and Transmitter** operation mode.This is another Serial Loopback mode but with the pseudo-random binary sequence (PRBS) BIST block active. The PRBS pattern depends on the serializer/deserializer (SERDES) factor.	"Basic Functional Mode" section in the *Stratix IV Transceiver Architecture* chapter.
	Basic (PMA Direct) ■ None—This is the normal mode of operation in which each channel is treated independently. ■ XN—In this mode, the "N" in XN represents the number of channels in the bonded configuration. All N channels are clocked by the same transmit clock from the central clock divider block to minimize transmitter channel-to-channel skew.	"Basic PMA Direct Functional Mode" section in the *Stratix IV Transceiver Architecture* chapter.
	Deterministic Latency ■ ×1—In this mode, you can have up to two configured channels per transceiver block. Each channel uses one CMU PLL and its feedback path to compensate for the uncertain latency. ■ ×4—In this mode, you can have up to four configured channels per transceiver block. All channels use one CMU PLL per block and its feedback path to compensate for the uncertain latency.	"Deterministic Latency Mode" section in the *Stratix IV Transceiver Architecture* chapter.

Table 1–1. MegaWizard Plug-In Manager Options (General Screen for Basic Mode) (Part 3 of 10)

ALTGX Setting	Description	Reference
Which subprotocol will you be using?	**PCI Express (PIPE)** In PCI Express (PIPE) mode, there are six subprotocols: ■ Gen1 ×1—The transceiver is configured as a single-lane PCI Express (PIPE) link for a 2.5 Gbps data rate. ■ Gen1 ×4—The transceiver is configured as a four-lane PCI Express (PIPE) link for a data rate of 2.5 Gbps. ■ Gen1 ×8—The transceiver is configured as an eight-lane PCI Express (PIPE) link for a data rate of 2.5 Gbps. ■ Gen2 ×1—The transceiver is configured as a single-lane PCI Express (PIPE) link for a 5.0 Gbps data rate. ■ Gen2 ×4—The transceiver is configured as a four-lane PCI Express link for a data rate of 5.0 Gbps. ■ Gen2 ×8—The transceiver is configured as an eight-lane PCI Express (PIPE) link for a data rate of 5.0 Gbps.	"PCI Express (PIPE) Mode" in the *Stratix IV Transceiver Architecture* chapter.
	SDI In SDI mode, the two available subprotocols are: ■ 3G—third-generation (3 Gbps) SDI at 2967 Mbps or 2970 Mbps. ■ HD—high-definition SDI at 1483.5 Mbps or 1485 Mbps.	"SDI Mode" in the *Stratix IV Transceiver Architecture* chapter.
	SONET/SDH In SONET/SDH mode, the three available subprotocols and their data rates are: ■ OC-12—622 Mbps ■ OC-48—2488.32 Mbps ■ OC-96—4976.64 Mbps	"SONET/SDH Mode" in the *Stratix IV Transceiver Architecture* chapter.
Enforce default settings for this protocol.	**Deterministic Latency** **GIGE** **(OIF) CEI PHY Interface** **PCI Express (PIPE)** **SONET/SDH** **XAUI** If you select this option, all mode-specific ports and settings are used.	—

Table 1–1. MegaWizard Plug-In Manager Options (General Screen for Basic Mode) (Part 4 of 10)

ALTGX Setting	Description	Reference
What is the operation mode?	**Basic** **Basic (PMA Direct)** **Deterministic Latency** **SDI** **Serial RapidIO** **SONET/SDH** The available operation modes are **Receiver only**, **Transmitter only**, and **Receiver and Transmitter**.	—
	GIGE The available operation modes are **Transmitter only**, and **Receiver and Transmitter**.	—
	PCI Express (PIPE) **XAUI** Only **Receiver and Transmitter** mode is allowed.	—
What is the number of channels?	**Basic** **Basic (PMA Direct)** **Deterministic Latency** **SDI** **Serial RapidIO** The number of channels required with the same configuration. This option determines how many identical channels this ALTGX instance contains.	—
	GIGE **(OIF) CEI PHY Interface** **SONET/SDH** This option allows you to select how many channels this ALTGX instance contains. In these modes, the number of channels increments by one.	—
	PCI Express (PIPE) This is the number of channels required with the same configuration. ■ In a ×4 subprotocol, the number of channels increments by 4. ■ In a ×8 subprotocol, the number of channels increment by 8.	—
	XAUI This option allows you to select how many identical channels this ALTGX instance contains. In XAUI mode, the number of channels increments by 4.	—

Table 1–1. MegaWizard Plug-In Manager Options (General Screen for Basic Mode) (Part 5 of 10)

ALTGX Setting	Description	Reference
What is the deserializer block width?	**Basic** **Basic (PMA Direct)** **Deterministic Latency** This option sets the transceiver data path width. ■ **Single-width**—This mode operates from 600 Mbps to 3.75 Gbps. ■ **Double-width**—This mode operates from 1 Gbps to 8.5 Gbps.	"Basic Single-Width Mode Configurations" and "Basic Double-Width Mode Configurations "sections in the *Stratix IV Transceiver Architecture* chapter.
	GIGE **PCI Express (PIPE)** **SDI** **Serial RapidIO** **XAUI** These modes only operate in single-width mode. Double-width mode is not allowed.	—
	(OIF) CEI PHY Interface The (OIF) CEI PHY Interface mode only operates in double-width mode. Single-width mode is not allowed.	—
	SONET/SDH This option allows you to set the transceiver data path width. ■ **Single-width**—Selected automatically in OC-12 and OC-48 configurations. The transceiver data path width is 8 bits. ■ **Double-width**—Selected automatically in OC-96 configurations. The transceiver data path width is 16 bits.	—

Table 1–1. MegaWizard Plug-In Manager Options (General Screen for Basic Mode) (Part 6 of 10)

ALTGX Setting	Description	Reference
What is the channel width?	**Basic** **Deterministic Latency** This option determines the FPGA fabric-Transceiver interface width. ■ Single-width mode—Selecting 8 or 10 bits bypasses the byte serializer/deserializer. Selecting 16 or 20 bits uses the byte serializer/deserializer. ■ Double-width mode—Selecting 16 or 20 bits bypasses the byte serializer/deserializer. Selecting 32 or 40 bits uses the byte serializer/deserializer.	"Byte Serializer" and "Byte Deserializer" sections in the *Stratix IV Transceiver Architecture* chapter.
	Basic (PMA Direct) This option determines the FPGA fabric-Transceiver interface width. ■ Single-width mode—You can select 8 or 10 bits. ■ Double-width mode— You can select 16 or 20 bits.	
	GIGE This option determines the FPGA fabric-Transceiver interface width. In GIGE mode, only 8 bits are allowed.	
	(OIF) CEI PHY Interface This option selects the FPGA fabric-Transceiver width. In (OIF) CEI PHY Interface mode, only 32 bits are allowed.	
	PCI Express (PIPE) This option determines the FPGA fabric-Transceiver interface width. ■ In PCI Express (PIPE) Gen1 (2.5 Gbps) mode, 8 and 16 bits are allowed. ■ In PCI Express (PIPE) Gen2 (5 Gbps) mode, only 16 bits are allowed.	
	SDI This option determines the FPGA fabric-Transceiver interface width: ■ HD mode—10-bit and 20-bit channel widths are allowed. ■ 3G mode—only 20-bit channel width is allowed. ■ 10-bit configuration—the byte serializer is not used. ■ 20-bit configuration—the byte serializer is used.	
	Serial RapidIO The channel width is fixed to 16 in Serial RapidIO mode.	

Table 1–1. MegaWizard Plug-In Manager Options (General Screen for Basic Mode) (Part 7 of 10)

ALTGX Setting	Description	Reference
What is the channel width?	**SONET/SDH** This option selects the FPGA fabric-Transceiver interface width. Depending on your subprotocol selection, choose one of the following: ■ **8 bits** for OC-12 mode ■ **16 bits** for OC-48 mode ■ **32 bits** for OC-96 mode	"Byte Serializer" and "Byte Deserializer" sections in the *Stratix IV Transceiver Architecture* chapter.
	XAUI XAUI mode only operates in single-width mode.	
What would you like to base the setting on?	**Basic** **Basic (PMA Direct)** You can select one of the following options: ■ **Data rate**—Selecting this option allows you to enter the transceiver channel serial data rate. Based on the value you enter, the ALTGX MegaWizard Plug-In Manager populates the input reference clock frequency options in the **What is the input clock frequency?** field. The ALTGX MegaWizard Plug-In Manager determines these input reference clock frequencies depending on the available multiplier settings. ■ **Input clock frequency**—Selecting this option allows you to enter your input clock frequency. Based on the value you enter, the ALTGX MegaWizard Plug-In Manager populates the data rate options in the **What is the effective data rate?** field. The ALTGX MegaWizard Plug-In Manager determines these data rate options depending on the available multipler settings.	—

Table 1–1. MegaWizard Plug-In Manager Options (General Screen for Basic Mode) (Part 8 of 10)

ALTGX Setting	Description	Reference
What is the effective data rate?	**Basic** **Basic (PMA Direct)** **Deterministic Latency** ■ If you select the **Data Rate** option in the **What would you like to base the setting on?** field, the ALTGX MegaWizard Plug-In Manager allows you to specify the effective serial data rate value in this field. ■ If you select the **Input Clock Frequency** option in the **What would you like to base the setting on?** field, the ALTGX MegaWizard Plug-In Manager displays the list of effective serial data rates in this field.	—
	GIGE This option is not available in **GIGE** mode. The transceiver channel serial data rate is fixed to 1250 Mbps in this mode.	—
	(OIF) CEI PHY Interface The allowed effective data rate is between 3125 Mbps and 6500 Mbps. Enter the transceiver channel's serial data rate in this field.	—
	PCI Express (PIPE) This option is not available in PCI Express (PIPE) mode. The defaults are: ■ 2500 Mbps for PCI Express (PIPE) Gen1 mode. ■ 5000 Mbps for PCI Express (PIPE) Gen 2 mode.	—
	SDI The effective data rate is fixed at: ■ 2967 Mbps or 2970 Mbps in 3G mode. ■ 1483.5 Mbps or 1485 Mbps in HD mode.	—
	Serial RapidIO Enter one of these three data rates in this option: ■ 1250 Mbps. ■ 2500 Mbps. ■ 3125 Mbps.	—
	SONET/SDH The effective data rate is fixed at: ■ 622 Mbps in OC-12 mode. ■ 2488.32 Mbps in OC-48 mode. ■ 4976 Mbps in OC-96 mode.	—
	XAUI The effective data rate can be from 3125 Mbps to 3750 Mbps.	—

Table 1–1. MegaWizard Plug-In Manager Options (General Screen for Basic Mode) (Part 9 of 10)

ALTGX Setting	Description	Reference
What is the input clock frequency?	**Basic** **Basic (PMA Direct)** ■ If you select the **Input Clock Frequency** option in the **What would you like to base the setting on?** field, the ALTGX MegaWizard Plug-In Manager allows you to specify the input reference clock frequency in this field. ■ If you select the **Data Rate** option in the **What would you like to base the setting on?** field, the ALTGX MegaWizard Plug-In Manager displays the list of input reference clock frequencies in this field.	"Input Reference Clocking" section in the *Stratix IV Transceiver Clocking* chapter.
	Deterministic Latency **GIGE** **(OIF) CEI PHY Interface** **SDI** **SONET/SDH** Based on the effective data rate value in the **What is the effective data rate?** field, the ALTGX MegaWizard Plug-In Manager determines the input reference clock frequencies depending on the available multiplier settings.	
	PCI Express (PIPE) This option is not available in PCI Express (PIPE) mode. The input reference clock frequency is fixed to 100 MHz in PCI Express (PIPE) mode.	
	Serial RapidIO **XAUI** This option provides the available input reference clock frequencies depending on whether your effective serial data rate is 1250 Mbps, 2500 Mbps, or 3125 Mbps and the available multiplier settings.	

Table 1–1. MegaWizard Plug-In Manager Options (General Screen for Basic Mode) (Part 10 of 10)

ALTGX Setting	Description	Reference
Specify base data rate.	**Basic** **Basic (PMA Direct)** The ALTGX MegaWizard Plug-In Manager provides you the **base data rate** options for the CMU/ATX PLL and receiver clock data recovery (CDR). If you select a value in this field that is greater than the value in the **What is the effective data rate?** field, the ALTGX MegaWizard Plug-In Manager enables the appropriate local clock divider values. The local divider is present in the and receiver channels.	—
	GIGE This option is not available in this mode because the data rate is fixed. The ALTGX MegaWizard Plug-In Manager provides you the base data rate options for the CMU PLL and receiver CDR.	—
	(OIF) CEI PHY Interface **Serial RapidIO** **XAUI** This option is not available in these modes. The ALTGX MegaWizard Plug-In Manager provides you the base data rate options for the CMU PLL and receiver CDR.	—
	PCI Express (PIPE) For Gen1 ×1, an optional base data rate of either 2500 or 5000 Mbps is available.	—
	SDI This option is not available this mode as the data rate is fixed in 3G and HD modes. The ALTGX MegaWizard Plug-In Manager provides you the base data rate options for the CMU PLL and receiver CDR.	—
	SONET/SDH This option is not available in this mode as the data rates are fixed in OC-12, OC-48, and OC-96 modes. The ALTGX MegaWizard Plug-In Manager provides you the base data rate options for the CMU PLL and receiver CDR in this option.	—

PLL/Ports Screen for the Parameter Settings

Figure 1–4 shows the **PLL/Ports** screen of the ALTGX MegaWizard Plug-In Manager for the Parameter Settings.

Figure 1–4. MegaWizard Plug-In Manager—ALTGX (PLL/Ports Screen)

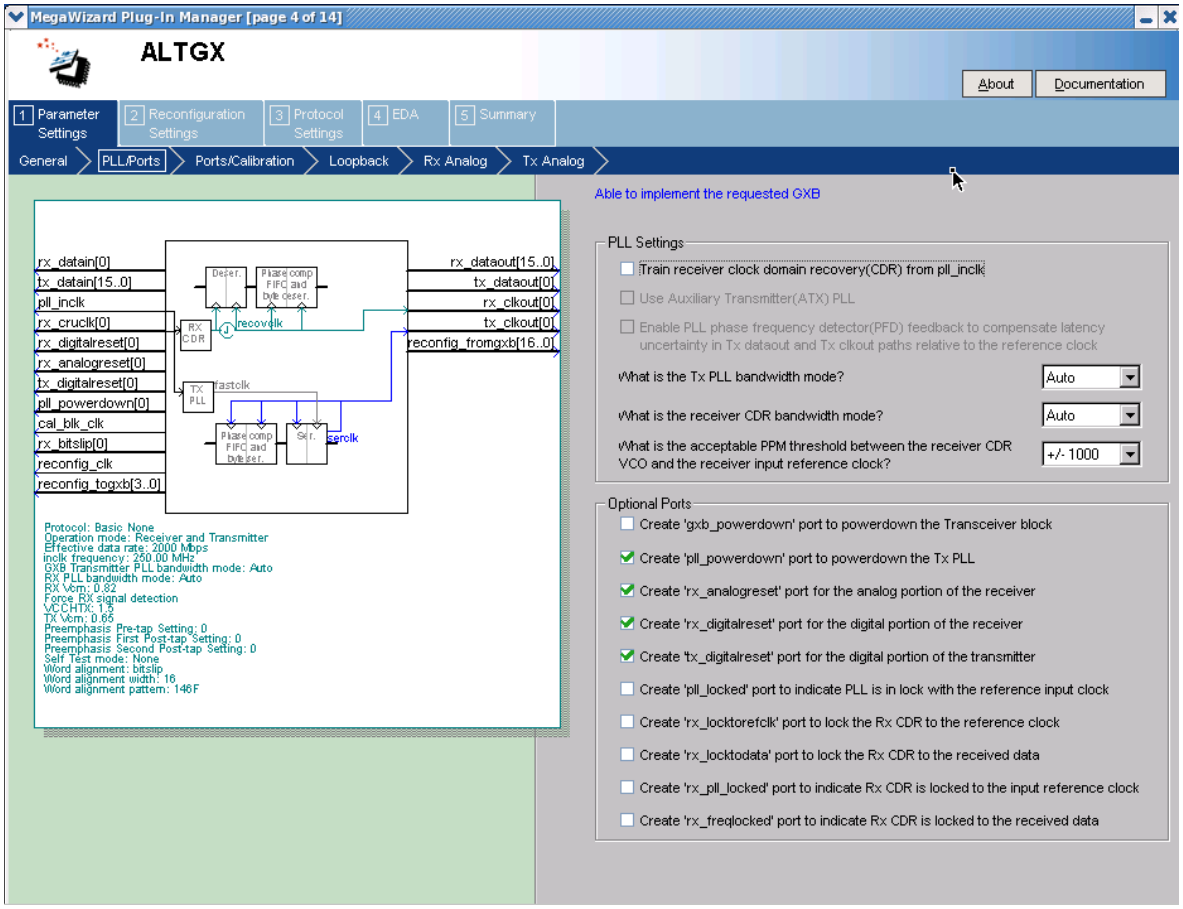


Table 1–2 describes the available options on the **PLL/ports** screen of the MegaWizard Plug-In Manager for your ALTGX custom megafunction variation.

Table 1–2. MegaWizard Plug-In Manager Options (PLL/Ports Screen) (Part 1 of 3)

ALTGX Setting	Description	Reference
Train receiver clock and data recovery (CDR) from `pll_inclk`.	If you select this option, the input reference clock to the CMU PLL trains the receiver CDR.	Table 1-77 in the *Stratix IV Transceiver Architecture* chapter.
Use ATX Transmitter PLL	This option is only available for certain data rates. Refer to Datasheet for the supported data rates. This option enables the auxiliary transmitter PLL. This is a low-jitter PLL that resides between the transceiver blocks and can be used as a transmitter PLL.	"Auxiliary Transmit (ATX) PLL Block" section in the *Stratix IV Transceiver Architecture*, the *Stratix IV Transceiver Clocking* chapter, and the *Stratix IV Device Datasheet* section.
Enable PLL phase frequency detector (PFD) feedback to compensate latency uncertainty in `tx_dataout` and `tx_clkout` paths relative to the reference clock.	This option applies only when you select **Deterministic Latency** functional mode.	"CMU PLL Feedback" section in the *Stratix IV Transceiver Architecture* chapter.
What is the TX PLL bandwidth mode?	The available options are **Auto, Low**, **Medium**, and **High**. Select the appropriate option based on your system requirements.	"PLL Bandwidth Setting" section in the *Stratix IV Transceiver Architecture* chapter and the *Stratix IV Device Datasheet* section.
What is the receiver CDR bandwidth mode?	The available options are **Auto, Low**, **Medium**, and **High**. Select the appropriate option based on your system requirements.	"Clock and Data Recovery Unit" section in the *Stratix IV Transceiver Architecture* chapter and the *Stratix IV Device Datasheet* section.
What is the acceptable PPM threshold between the receiver CDR VCO and the receiver input reference clock?	In Automatic Lock mode, the CDR remains in Lock-to-Data (LTD) mode as long as the parts per million (PPM) difference between the CDR VCO output clock and the input reference clock is less than the PPM value that you set in this option. If the PPM difference is greater than the PPM value that you set in this option, the CDR switches to Lock-to-Reference (LTR) mode. The range of values available in this option is ±62.5 ppm to ±1000 ppm. *(1)*	"Automatic Lock Mode" section in the *Stratix IV Transceiver Architecture* chapter.
Optional Ports		
Create a `gxb_powerdown` port to power down the transceiver block.	When asserted, this signal powers down the entire transceiver block. If none of the channels are instantiated in a transceiver block, the Quartus II software automatically powers down the entire transceiver block.	"User Reset and Power Down Signals" section in the *Reset Control and Power Down* chapter.
Create a `pll_powerdown` port to power down the TX PLL.	Each transceiver block has two CMU PLLs. Each CMU/ATX PLL has a dedicated power down signal called `pll_powerdown`. This signal powers down the CMU/ATX PLL.	"User Reset and Power Down Signals" section in the *Reset Control and Power Down* chapter.

Table 1–2. MegaWizard Plug-In Manager Options (PLL/Ports Screen) (Part 2 of 3)

ALTGX Setting	Description	Reference
Create a `rx_analogreset` port for the analog portion of the receiver.	The receiver analog reset port is available in **Receiver only** and **Receiver and Transmitter** operation modes. This resets part of the analog portion of the receiver CDR in the receiver channel. Altera recommends using this port to implement the recommended reset sequence. The minimum pulse width is two parallel clock cycles.	"User Reset and Power Down Signals" in the *Reset Control and Power Down* chapter.
Create a `rx_digitalreset` port for the digital portion of the receiver.	The receiver digital reset port is available in **Receiver only** and **Receiver and Transmitter** operation modes. This resets the PCS portion of the receiver channel. Altera recommends using this port to implement the recommended reset sequence. The minimum pulse width is two parallel clock cycles.	"User Reset and Power Down Signals" section in the *Reset Control and Power Down* chapter.
Create a `tx_digitalreset` port for the digital portion of the transmitter.	The transmitter digital reset port is available in **Transmitter only** and **Receiver and Transmitter** operation modes. This resets the PCS portion of the transmitter channel. Altera recommends using this port to implement the recommended reset sequence. The minimum pulse width is two parallel clock cycles.	"User Reset and Power Down Signals" section in the *Reset Control and Power Down* chapter.
Create a `pll_locked` port to indicate PLL is in lock with the reference input clock.	Each CMU/ATX PLL has a dedicated `pll_locked` signal that is fed to the FPGA fabric to indicate when the PLL is locked to the input reference clock.	"Transceiver Reset Sequences" section in the *Reset Control and Power Down* chapter.
Create an `rx_locktorefclk` port to lock the RX CDR to the reference clock.	When this signal is asserted high, the LTR/LTD controller forces the receiver CDR to lock to the phase and frequency of the input reference clock. *(1), (2)*	"LTR/LTD Controller" section in the *Stratix IV Transceiver Architecture* chapter.
Create an `rx_locktodata` port to lock the RX CDR to the received data.	When this signal is asserted high, the LTR/LTD controller forces the receiver CDR to lock to the received data. *(1), (2)*	"LTR/LTD Controller" section in the *Stratix IV Transceiver Architecture* chapter.
Create an `rx_pll_locked` port to indicate RX CDR is locked to the input reference clock.	■ In LTR mode, this signal is asserted high to indicate that the receiver CDR has locked to the phase and frequency of the input reference clock. ■ In LTD mode, this signal has no significance. *(1)*	"Lock-to-Reference (LTR) Mode" section in the *Stratix IV Transceiver Architecture* chapter.

Table 1–2. MegaWizard Plug-In Manager Options (PLL/Ports Screen) (Part 3 of 3)

ALTGX Setting	Description	Reference
Create an `rx_freqlocked` port to indicate RX CDR is locked to the received data.	This signal is asserted high to indicate that the receiver CDR has switched from LTR to LTD mode. This signal has relevance only in Automatic Lock mode and may be required to control the transceiver resets, as described in the *User Reset and Power Down Signals* section in the *Reset Control and Power Down* chapter in volume 2 of the *Stratix IV Device Handbook*. *(1)*	"LTR/LTD Controller" section in the *Stratix IV Transceiver Architecture* chapter.

Notes to Table 1–2:

(1) LTR mode is Lock-to-Reference mode and LTD mode is Lock-to-Data mode.

(2) When `rx_locktorefclk` and `rx_locktodata` are both asserted high, `rx_locktodata` takes precedence over `rx_locktorefclk`, forcing the CDR to lock to the received data. When both these signals are de-asserted, the LTR/LTD controller is configured in Automatic Lock mode.

Ports/Calibration Screen for the Parameter Settings

Figure 1–5 shows the **Ports/Calibration** screen of the ALTGX MegaWizard Plug-In Manager for the Parameter Settings.

Figure 1–5. MegaWizard Plug-In Manager—ALTGX (Ports/Calibration Screen)

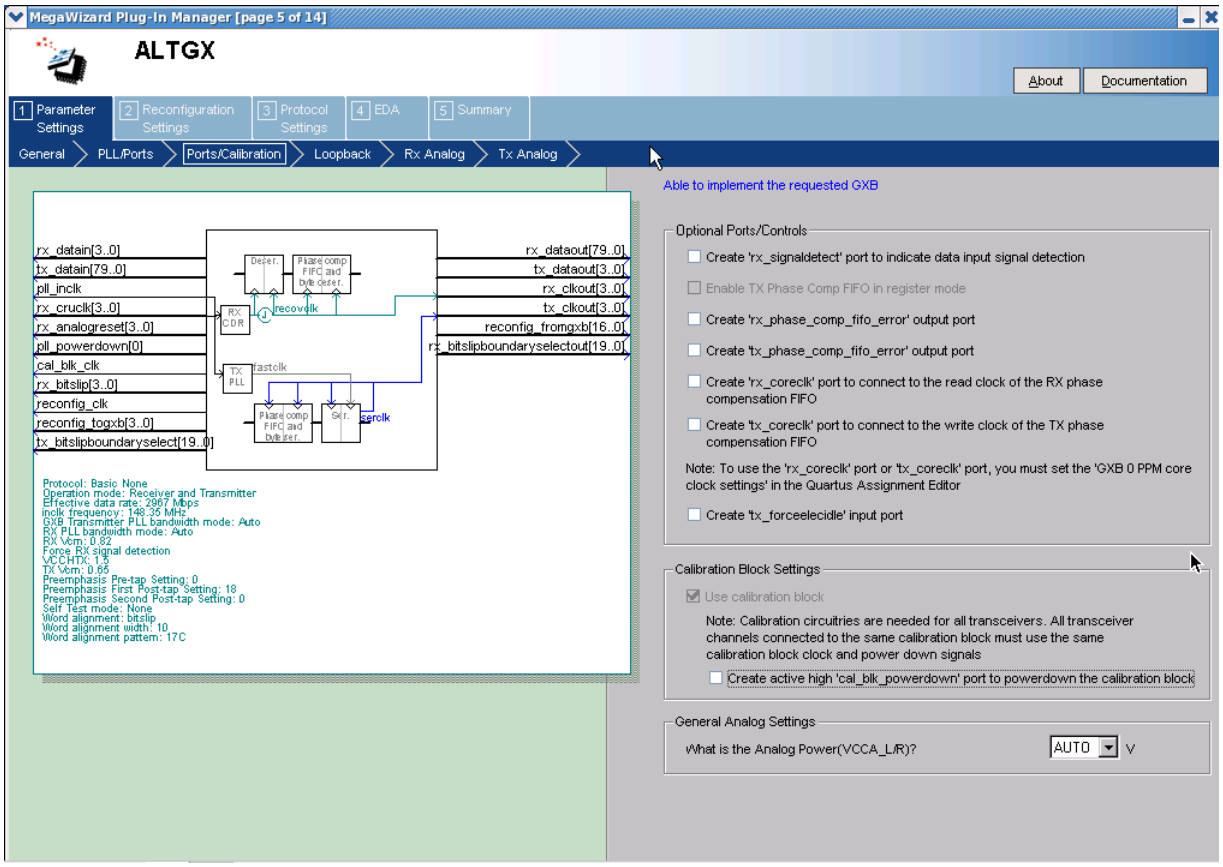


Table 1–3 describes the available options on the **Ports/Calibration** screen of the MegaWizard Plug-In Manager for your ALTGX custom megafunction variation. Unless indicated otherwise, the options apply to all functional modes.

Table 1–3. MegaWizard Plug-In Manager Options (Ports/Calibration Screen) (Part 1 of 3)

ALTGX Setting	Description	Reference
Optional Ports/Controls		
Create an `rx_signaldetect` port to indicate data input signal detection.	This port is only available in **Basic** and **PCI Express (PIPE)** mode.	"Signal Threshold Detection Circuitry" section in the *Stratix IV Transceiver Architecture* chapter.
Enable TX Phase Comp FIFO in register mode.	This option is only available in **Deterministic Latency** mode.	"TX Phase Compensation FIFO Status Signal" section in the *Stratix IV Transceiver Architecture* chapter.
Create an `rx_phase_comp_fifo_error` output port.	This output port indicates a Receiver Phase Compensation FIFO overflow or under-run condition.	"Receiver Phase Compensation FIFO Error Flag" section in the *Stratix IV Transceiver Architecture* chapter.

Table 1–3. MegaWizard Plug-In Manager Options (Ports/Calibration Screen) (Part 2 of 3)

ALTGX Setting	Description	Reference
Create a `tx_phase_comp_fifo_error` output port.	This output port indicates a Transmitter Phase Compensation FIFO overflow or under-run condition.	"TX Phase Compensation FIFO Status Signal" section in the *Stratix IV Transceiver Architecture* chapter.
Create an `rx_coreclk` port to connect to the read clock of the RX phase compensation FIFO.	You can clock the parallel output data from the receiver using this optional input port. This port allows you to clock the read side of the Receiver Phase Compensation FIFO with a user-provided clock (FPGA fabric clock, FPGA fabric-Transceiver interface clock, or input reference clock).	"FPGA Fabric-Transceiver Interface Clocking" section in the *Stratix IV Transceiver Clocking* chapter.
Create a `tx_coreclk` port to connect to the write clock of the TX phase compensation FIFO.	You can clock the parallel transmitter data generated in the FPGA fabric using this optional input port. This port allows you to clock the write side of the Transmitter Phase Compensation FIFO with a user-provided clock (FPGA fabric clock, FPGA fabric-Transceiver interface clock, or input reference clock).	"FPGA Fabric-Transceiver Interface Clocking" section in the *Stratix IV Transceiver Clocking* chapter.
Create a `tx_forceelecidle` input port	In Basic and PCI Express (PIPE) modes, this optional input signal places the transmitter buffer in the electrical idle state.	"Transceiver Channel Architecture" section in the *Stratix IV Transceiver Architecture* chapter.
Use calibration block.	The calibration block is always enabled.	"Calibration Blocks" section in the *Stratix IV Transceiver Architecture* chapter.

Table 1–3. MegaWizard Plug-In Manager Options (Ports/Calibration Screen) (Part 3 of 3)

ALTGX Setting	Description	Reference
Create an active high `cal_blk_powerdown` to power down the calibration block.	Asserting this signal high powers down the calibration block. A high-to-low transition on this signal restarts calibration.	"Input Signals to the Calibration Block" section in the *Stratix IV Transceiver Architecture* chapter.
What is the Analog Power ($V_{CCA_L/R}$)?	The options available for selection are based on what you specify in the **Specify base data rate** option: ■ **3.3 V**—Available up to 11.3 Gbps for Stratix IV GT devices only. ■ **2.5 V**—Available up to 4.25 Gbps. ■ **AUTO**—The ALTGX MegaWizard Plug-In Manager automatically sets $V_{CCA_L/R}$ to **2.5 V** for the VCO data rates less than 4.25 Gbps. or $V_{CCA_L/R}$ to **3.0 V** for the VCO data rates greater than 4.25 Gbps. It is up to you to connect the correct voltage supply to the $V_{CCA_L/R}$ pins on the board.	"General Requirements to Combine Channels" section in the *Configuring Multiple Protocols and Data Rates* chapter.

Loopback Screen for the Parameter Settings

Figure 1–6 shows the **Loopback** screen of the ALTGX MegaWizard Plug-In Manager for the Parameter Settings.

Figure 1–6. MegaWizard Plug-In Manager—ALTGX (Loopback Screen)

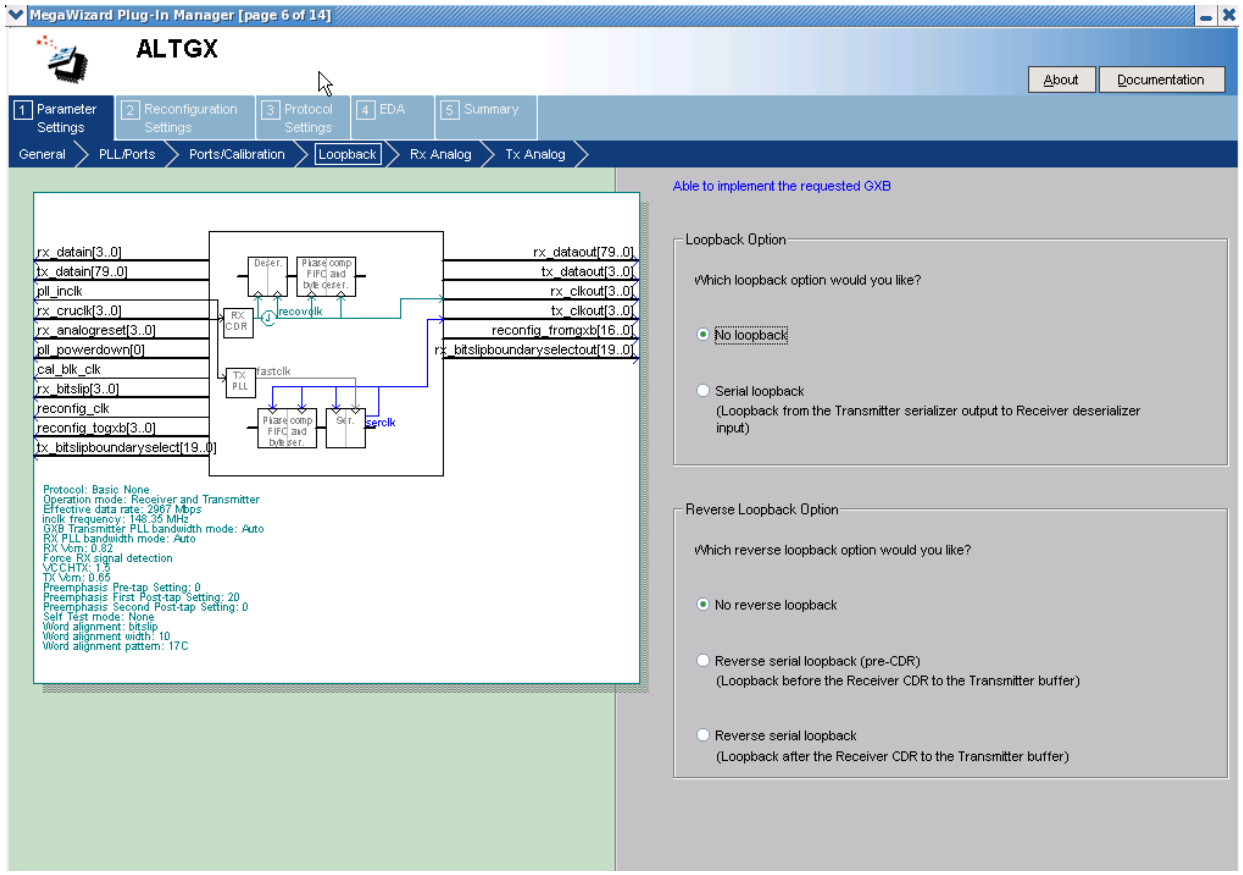


Table 1–4 describes the available options on the **Loopback** screen of the MegaWizard Plug-In Manager for your ALTGX custom megafunction variation.

Table 1–4. MegaWizard Plug-In Manager Options (Lpbk Screen)

ALTGX Setting	Description	Reference
Which loopback option would you like?	There are two options available: ■ **No loopback**—This is the default mode. ■ **Serial loopback**—If you select serial loopback, the `rx_serialllpbken` port is available to control the serial loopback feature dynamically. →1'b1—enables serial loopback →1'b0—disables serial loopback This signal is asynchronous to the receiver datapath.	"Serial Loopback" section in the *Stratix IV Transceiver Architecture* chapter.
Which reverse loopback option would you like?	There are three options available: ■ **No reverse loopback**—This is the default mode. ■ **Reverse Serial loopback (pre-CDR)**—This is the loopback before the receiver's CDR block to the transmitter buffer. The receiver path in PCS is active but the transmitter side is not. ■ **Reverse Serial loopback**—This is a loopback after the receiver's CDR block to the transmitter buffer. The receiver path in PCS is active but the transmitter side is not.	"Loopback Modes" section in the *Stratix IV Transceiver Architecture* chapter.

RX Analog Screen for the Parameter Settings

Figure 1–7 shows the **RX Analog** screen of the ALTGX MegaWizard Plug-In Manager for the Parameter Settings.

Figure 1–7. MegaWizard Plug-In Manager—ALTGX (RX Analog Screen)

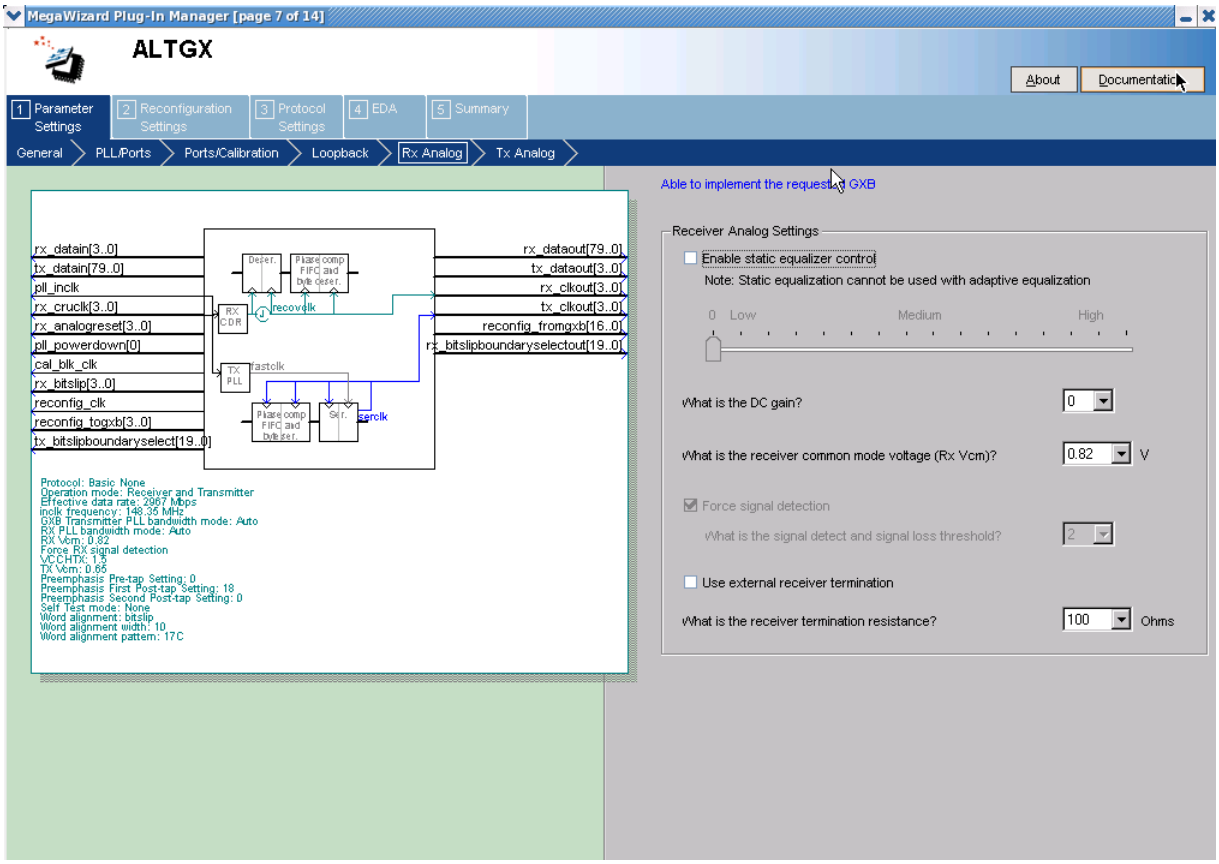


Table 1–5 describes the available options on the **RX Analog** screen of the MegaWizard Plug-In Manager for your ALTGX custom megafunction variation.

Table 1–5. MegaWizard Plug-In Manager Options (RX Analog Screen) (Part 1 of 2)

ALTGX Setting	Description	Reference
Enable static equalizer control.	This option enables the static equalizer settings.	"Programmable Equalization and DC Gain" section in the *Stratix IV Transceiver Architecture* chapter and the *Stratix IV Device Datasheet* section.
What is the DC gain?	This DC gain option has five settings: ■ 0 – 0 dB ■ 1 – 3 dB ■ 2 – 6 dB ■ 3 – 9 dB ■ 4 – 12 dB	"Programmable Equalization and DC Gain" section in the *Stratix IV Transceiver Architecture* chapter.

Table 1–5. MegaWizard Plug-In Manager Options (RX Analog Screen) (Part 2 of 2)

ALTGX Setting	Description	Reference
What is the receiver common mode voltage (RX V_{CM})?	The receiver common mode voltage is programmable to 0.82 V or 1.1 V.	"Receiver Channel Datapath" section in the *Stratix IV Transceiver Architecture* chapter.
Force signal detection.	In **PCI Express (PIPE)** mode, this option disables the signal threshold detect circuit for the receiver CDR. The receiver CDR no longer depends on the signal detect criterion to switch from LTR to LTD mode.	"Signal Threshold Detection Circuitry" section in the *Stratix IV Transceiver Architecture* chapter.
What is the signal detect threshold?	Use this option in **PCI Express (PIPE)** or **Basic** mode with the 8B10B block enabled and the `rx_signaldetect` port selected to determine the threshold level for the signal detect circuit. ■ **PIPE** mode—The levels are fixed. ■ **Basic** mode—A range of values depending on the data rate are available. The levels will be determined after characterization.	"Signal Threshold Detection Circuitry" section in the *Stratix IV Transceiver Architecture* chapter.
Use external receiver termination.	Select this option if you want to use an external termination resistor instead of differential on-chip termination (OCT). If checked, this option turns off the receiver OCT.	"Programmable Differential On-Chip Termination" section in the *Stratix IV Transceiver Architecture* chapter.
What is the receiver termination resistance?	This option allows you to select the receiver differential termination value. The settings allowed are: ■ 85 Ω ■ 100 Ω ■ 120 Ω ■ 150 Ω	"Programmable Differential On-Chip Termination" section in the *Stratix IV Transceiver Architecture* chapter, and the *Stratix IV Device Datasheet* section.

TX Analog Screen for the Parameter Settings

Figure 1–8 shows the **TX Analog** screen of the ALTGX MegaWizard Plug-In Manager for the Parameter Settings.

Figure 1–8. MegaWizard Plug-In Manager—ALTGX (TX Analog Screen)

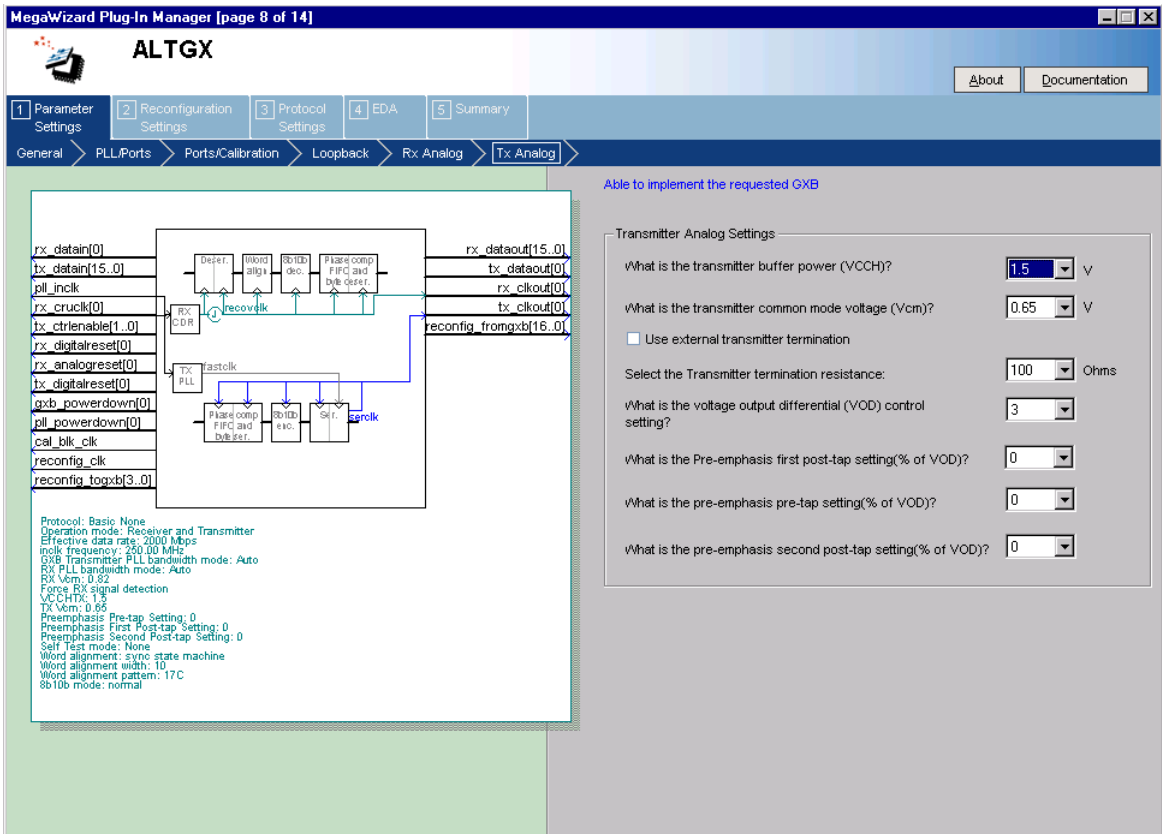


Table 1–6 describes the available options on the **TX Analog** screen of the MegaWizard Plug-In Manager for your ALTGX custom megafunction variation.

Table 1–6. MegaWizard Plug-In Manager Options (TX Analog Screen)

ALTGX Setting	Description	Reference
What is the transmitter buffer power (V_{CCH})?	The options available for selection are based on what you enter in the **What is the effective data rate?** option. ■ **1.4 V**—Available up to 8.5 Gbps. ■ **1.5 V** is available up to 6.5 Gbps (not available for Stratix IV GT). ■ **AUTO**—The ALTGX MegaWizard Plug-In Manager automatically sets V_{CCH} to **1.5 V** for the effective data rates less than 6.5 Gbps or V_{CCH} to **1.4 V** for effective data rates greater than 6.5 Gbps. It is up to you to connect the correct voltage supply to the V_{CCH} pins on the board.	"Programmable Transmit Output Buffer Power (V_{CCH})" section in the *Stratix IV Transceiver Architecture* chapter.
What is the transmitter common mode voltage (V_{CM})?	The transmitter common mode voltage is fixed to 0.65 V.	"Transmitter Output Buffer" in the *Stratix IV Transceiver Architecture* chapter and the *Stratix IV Device Datasheet* section.
Use external transmitter termination.	This option is available if you want to use an external termination resistor instead of the differential on-chip termination. Checking this option turns off the transmitter differential OCT.	"Programmable Transmitter Termination" section in the *Stratix IV Transceiver Architecture* chapter and the *Stratix IV Device Datasheet* section.
Select the transmitter termination resistance.	This option selects the transmitter differential termination value. The settings allowed are 85 Ω, 100 Ω, 120 Ω, and 150 Ω	"Programmable Transmitter Termination" section in the *Stratix IV Transceiver Architecture* chapter and the *Stratix IV Device Datasheet* section.
What is the voltage output differential (VOD) control setting?	This option selects the V_{OD} of the transmitter buffer. The available V_{OD} settings change based on the transmitter termination resistance value.	"Programmable Output Differential Voltage" section in the *Stratix IV Transceiver Architecture* chapter and and the *Stratix IV Device Datasheet* section.
What is the pre-emphasis first post-tap setting (% of VOD)?	This option sets the amount of pre-emphasis on the transmitter buffer using first post-tap.	"Programmable Pre-Emphasis" section in the *Stratix IV Transceiver Architecture* chapter.
What is the pre-emphasis pre-tap setting (% of VOD)?	This option sets the amount of pre-emphasis on the transmitter buffer using pre-tap.	"Programmable Pre-Emphasis" section in the *Stratix IV Transceiver Architecture* chapter.
What is the pre-emphasis second post-tap setting (% of VOD)?	This option sets the amount of pre-emphasis on the transmitter buffer using second post-tap.	"Programmable Pre-Emphasis" section in the *Stratix IV Transceiver Architecture* chapter.

Reconfiguration Settings

This section describes the various dynamic reconfiguration modes and settings for Stratix IV GX and GT transceivers.

In Reconfiguration Settings, when you enable the **Enable Channel and Transmitter PLL reconfiguration** option, the following screens become available:

- Modes
- Transmitter PLLs
- Clocking/Interface

The following sections describe these screens and their corresponding settings.

Modes Screen for the Reconfiguration Settings

Figure 1–9 shows the **Modes** screen, listing the various dynamic reconfiguration modes available.

Figure 1–9. MegaWizard Plug-In Manager—Reconfiguration Settings

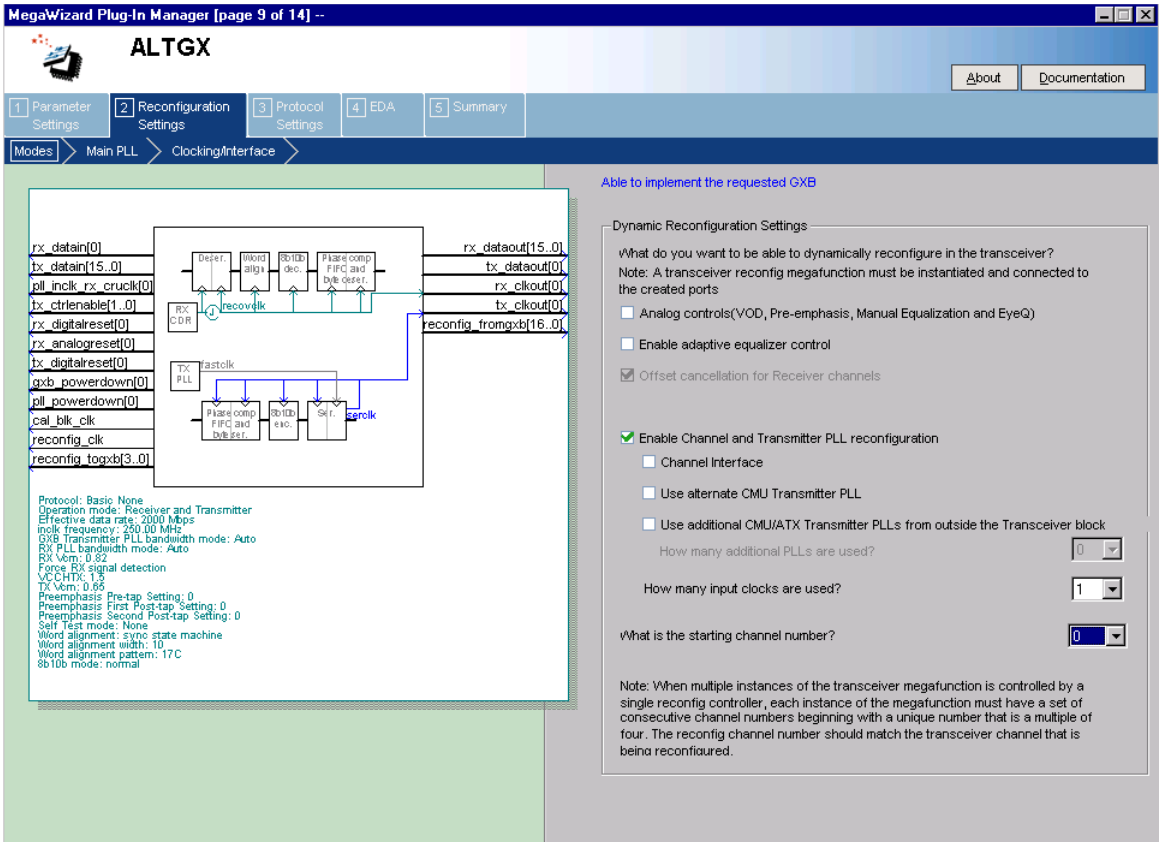


Table 1–7 describes the different options available in the **Modes** screen of the MegaWizard Plug-In Manager for your ALTGX custom megafunction variation.

Table 1–7. MegaWizard Plug-In Manager Options (Modes Screen) (Part 1 of 2)

ALTGX Setting	Description	Reference
Dynamic Reconfiguration Settings		
What do you want to be able to dynamically reconfigure in the transceiver?	The different dynamic reconfiguration modes available are listed in the **Reconfiguration Settings** screen. Based on which portion of the transceiver you want to reconfigure, select the corresponding options, and connect the ALTGX_RECONFIG instance to the ALTGX instance.	"Dynamic Reconfiguration Modes Implementation" section in the *Stratix IV Dynamic Reconfiguration* chapter.
	■ Analog controls (VOD, Pre-emphasis, and Manual Equalization and EyeQ)—Enable this option to dynamically reconfigure the PMA control settings similar to VOD, pre-emphasis, manual equalization, DC gain, and EyeQ.	"PMA Controls Reconfiguration Mode Details" section in the *Stratix IV Dynamic Reconfiguration* chapter.
	■ Enable adaptive equalizer control—Selecting this option enables the Adaptive Equalization (AEQ) hardware and provides the following additional ports: → `aeq_togxb[]` → `aeq_fromgxb[]` These ports provide the interface between the receiver channel and the dynamic reconfiguration controller.	"Enabling the AEQ Control Logic and AEQ Hardware" section in the *Stratix IV Dynamic Reconfiguration* chapter.
	■ Offset cancellation for receiver channels—This option is enabled by default for **Receiver only** and **Receiver and Transmitter** configurations. It is not available for **Transmitter only** configurations. Ensure that you connect a dynamic reconfiguration controller to all the transceiver channels in the design.	"Offset Cancellation Feature" section in the *Stratix IV Dynamic Reconfiguration* chapter.

Table 1–7. MegaWizard Plug-In Manager Options (Modes Screen) (Part 2 of 2)

ALTGX Setting	Description	Reference
Enable Channel and Transmitter PLL Reconfiguration	You must enable this option to reconfigure one of the following: Transmitter local divider block, CMU PLL, Transceiver channel, or Both the CMU PLL and transceiver channel. ■ Channel Interface—This option enables memory initialization file (**.mif**)-based reconfiguration among functional modes that have different FPGA fabric-Transceiver interface signals. This option also allows channel interface reconfiguration. ■ Use alternate CMU Transmitter PLL—This option sets up the alternate PLL so that the transceiver channel can optionally select between the output of the main and alternate transmitter PLL. ■ Use additional CMU/ATX Transmitter PLLs from outside the Transceiver Block—This option allows you to select a maximum of four transmitter PLLs. For example, you can select the ATX PLL as the main PLL and three additional PLLs. → How many additional PLLs are used?—You can have a maximum of two PLLs outside the transceiver block.	"Transceiver Channel Reconfiguration Modes Details" section in the *Stratix IV Dynamic Reconfiguration* chapter. "FPGA Fabric-Transceiver Channel Interface Selection" section in the *Stratix IV Dynamic Reconfiguration* chapter. "Transceiver Channel Reconfiguration Modes Details" section in the *Stratix IV Dynamic Reconfiguration* chapter in volume 2 of the *Stratix IV Device Handbook.* "Multi-PLL Settings" section in the *Stratix IV Dynamic Reconfiguration* chapter.
How many input clocks are used?	Enter the number of input clocks available for selection for the transmitter PLLs and receiver PLL. You have a choice of up to 10 input clock sources (clock 1, clock 2, and so on).	"Guidelines for Specifying the Input Reference Clocks" section in the *Stratix IV Dynamic Reconfiguration* chapter.
What is the starting channel number?	You must set the starting channel number of the first ALTGX instance controlled by the dynamic reconfiguration controller to **0**. Set the starting channel number of the consecutive ALTGX instances controlled by the same dynamic reconfiguration controller, if any, in the next available multiples of 4.	"Logical Channel Addressing while Reconfiguring the PMA Controls" section in the *Stratix IV Dynamic Reconfiguration* chapter.

Transmitter PLL Settings

Depending on the number of additional PLLs you select in the **How many additional PLLs are used?** option in Reconfiguration Settings, the corresponding PLL screens become available.

Each of these PLL screens have the same settings available for selection. Table 1–8 describes each of these settings in detail.

The Main PLL is the PLL you configure in the **General** screen. Therefore, some of the options are already enabled or disabled for this PLL. Some of the options differ when compared with the additional transmitter PLLs.

Figure 1–10 shows the options available on the **Main PLL** screen of the ALTGX MegaWizard Plug-In Manager.

Figure 1–10. MegaWizard Plug-In Manager Options—Main PLL Screen

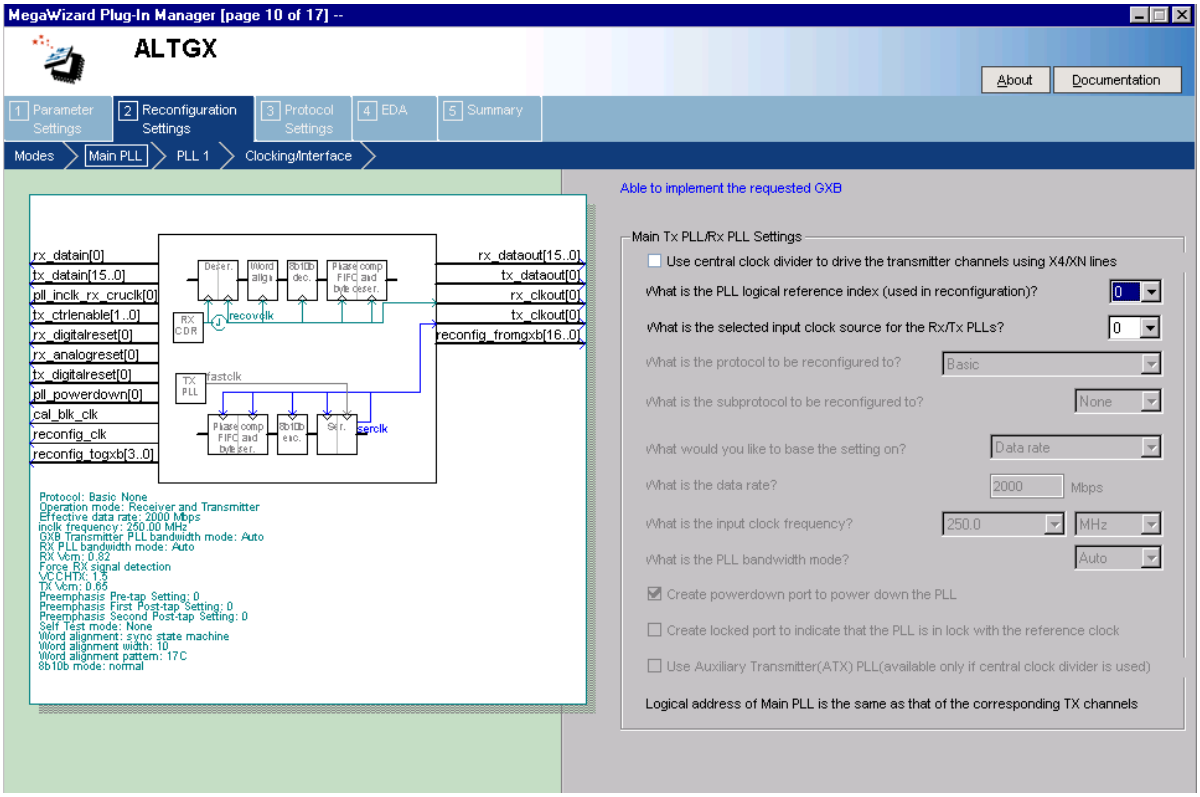


Table 1–8 describes the available options on the **Main PLL** screen of the MegaWizard Plug-In Manager for your ALTGX custom megafunction variation.

Table 1–8. MegaWizard Plug-In Manager Options (Main PLL Screen) (Part 1 of 3)

ALTGX Setting	Description	Reference
Main Tx PLL/Rx PLL Settings		
Use central clock divider to drive the transmitter channels using ×4/×N lines	If this option is enabled, the transmitter PLL is outside the transceiver block. If this option is disabled, the transmitter PLL is one of the CMU PLLs within the same transceiver block.	"Selecting the PLL Logical Reference Index for Additional PLLs" and the "Multi-PLL Settings" sections in the *Stratix IV Dynamic Reconfiguration* chapter.
What is the PLL logical reference index (used in reconfiguration)?	The PLL logical reference index is selected based on the location of the alternate PLL. If the **Use central clock divider to drive the transmitter channels using ×4/×N lines** option is unchecked this must be 0 or 1, otherwise this must be 2 or 3.	"Selecting the PLL Logical Reference Index for Additional PLLs" and "Selecting the Logical Reference Index of the CMU PLL" sections in the *Stratix IV Dynamic Reconfiguration* chapter.
What is the selected input clock source for the Rx/Tx PLLs?	Assign identification numbers to all input reference clocks that are used by the transmitter PLLs in their corresponding PLL screens. You can set up a maximum of 10 input reference clocks and assign identification numbers from 1 to 10.	"Guidelines for Specifying the Input Reference Clocks" section in the *Stratix IV Dynamic Reconfiguration* chapter.
What is the protocol to be reconfigured to?	Select the desired functional mode here, if you intend to dynamically reconfigure the transceiver channel to a different functional mode using the alternate transmitter PLL.	"Channel Reconfiguration with Transmitter PLL Select Mode Details" in the *Stratix IV Dynamic Reconfiguration* chapter.
What is the subprotocol to be reconfigured to?	This option is not available for the **Basic**, **(OIF) CEI PHY Interface**, **Serial RapidIO**, **GIGE**, and **XAUI** functional modes. This option is available for the following protocols and subprotocols: ■ Protocol = **PCI Express (PIPE)**; Subprotocols = Gen 1 and Gen 2 ■ Protocol = **SDI**; Subprotocols = 3G and HD ■ Protocol = **SONET/SDH**; Subprotocols = OC12, OC48, and OC96	—

Table 1–8. MegaWizard Plug-In Manager Options (Main PLL Screen) (Part 2 of 3)

ALTGX Setting	Description	Reference
What would you like to base the setting on?	This option is available only for **Basic** mode.You can select one of the following options for the alternate transmitter PLL: ■ **Input clock frequency**—Selecting this option allows you to enter your input clock frequency. Based on the value you enter, the ALTGX MegaWizard Plug-In Manager populates the data rate options in the **What is the effective data rate?** field. The ALTGX MegaWizard Plug-In Manager determines these data rate options depending on the available multiplier settings. ■ **Data rate**—Selecting this option allows you to enter the transceiver channel serial data rate. Based on the value you enter, the ALTGX MegaWizard Plug-In Manager populates the input reference clock frequency options in the **What is the input clock frequency?** field. The ALTGX MegaWizard Plug-In Manager determines these input reference clock frequencies depending on the available multiplier settings.	—
What is the data rate?	These settings are to dynamically reconfigure the transceiver channel to listen to the alternate transmitter PLL. ■ If you select the data rate option in the **What would you like to base the setting on?** field, the ALTGX MegaWizard Plug-In Manager allows you to specify the effective serial data rate value in this field. ■ If you select the **input clock frequency** option in the **What would you like to base the setting on?** field, the ALTGX MegaWizard Plug-In Manager displays the list of effective serial data rates in this field.	—
What is the input clock frequency?	These settings are to dynamically reconfigure the transceiver channel to listen to the alternate transmitter PLL. ■ If you select the **input clock frequency** option in the **What would you like to base the setting on?** field, the ALTGX MegaWizard Plug-In Manager displays the list of effective serial data rates in this field. ■ If you select the data rate option in the **What would you like to base the setting on?** field, the ALTGX MegaWizard Plug-In Manager allows you to specify the effective serial data rate value in this field.	"CMU PLL Reconfiguration Mode Details" section in the *Stratix IV Dynamic Reconfiguration* chapter.
What is the PLL bandwith mode?	The available options are **Auto**, **Low**, **Medium**, and **High**. Select the appropriate option based on your system requirements.	"PLL Bandwidth Setting" section in the *Stratix IV Transceiver Architecture* chapter.
Create powerdown port to power down the PLL.	Each transceiver block has two CMU PLLs. Each CMU/ATX PLL has a dedicated power down signal called `pll_powerdown`. This signal powers down the CMU PLL.	"User Reset and Power-Down Signals" section in the *Reset Control and Power Down* chapter.

Table 1–8. MegaWizard Plug-In Manager Options (Main PLL Screen) (Part 3 of 3)

ALTGX Setting	Description	Reference
Create locked port to indicate that the PLL is in lock with the reference clock.	Each CMU/ATX PLL has a dedicated `pll_locked` signal that is fed to the FPGA fabric to indicate when the PLL is locked to the input reference clock.	"User Reset and Power-Down Signals" section in the *Reset Control and Power Down* chapter.
Use Auxiliary Transmitter (ATX) PLL (available only if central clock divider is used)	This option is only available for certain data rates. Refer to Datasheet for the supported data rates. This option enables the auxiliary transmitter PLL. This is a low-jitter PLL that resides between the transceiver blocks and can be used as a transmitter PLL.	"Auxiliary Transmit (ATX) PLL Block" section in the *Stratix IV Transceiver Architecture* chapter and the *Stratix IV Device Datasheet* section.

Clocking/Interface Screen for the Reconfiguration Settings

Figure 1–11 shows the **Clocking/Interface** screen of the ALTGX MegaWizard Plug-In Manager for the Reconfiguration settings.

Figure 1–11. MegaWizard Plug-In Manager Options (Clocking/Interface Screen)

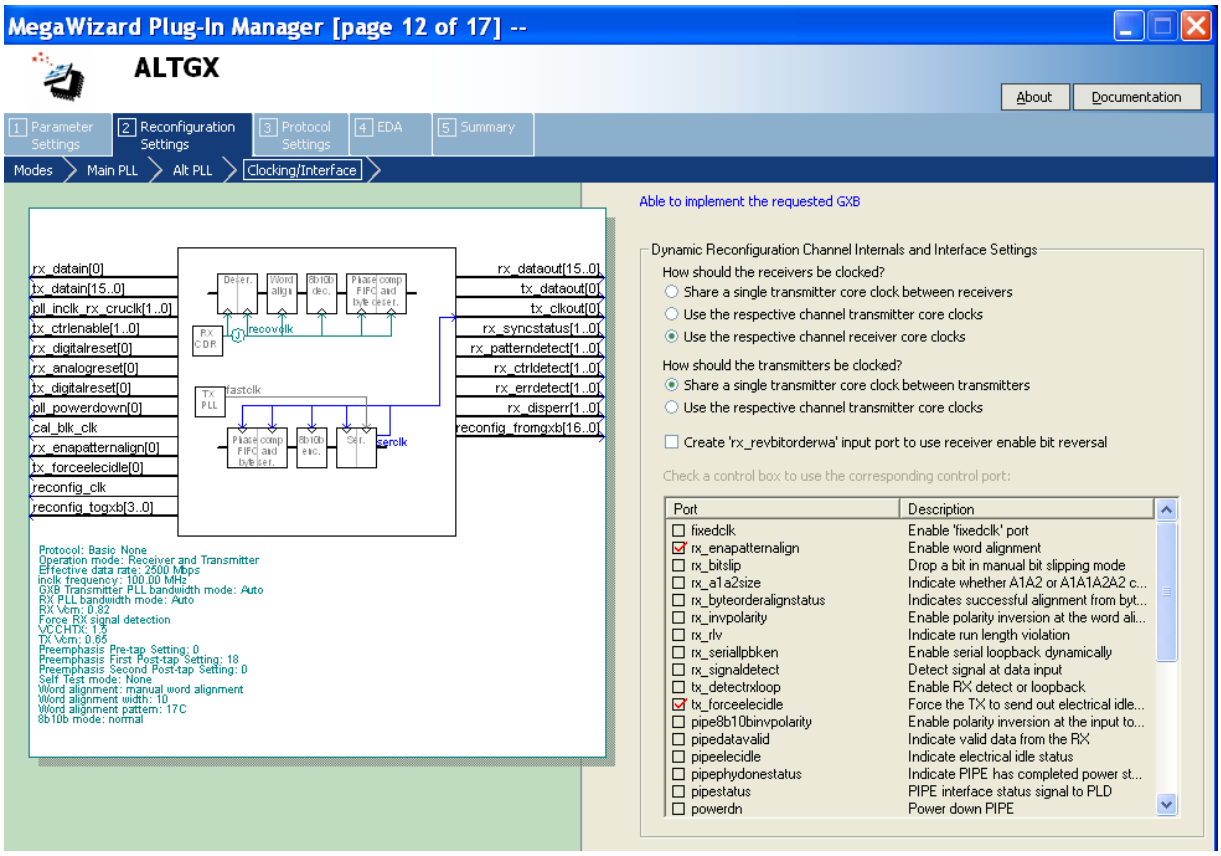


Table 1–9 describes the available options on the **Clocking/Interface** screen of the MegaWizard Plug-In Manager for your ALTGX custom megafunction variation.

This screen is not available for Basic (PMA Direct) ×1 and xN configurations.

Table 1–9. MegaWizard Plug-In Manager Options (Clocking/Interface Screen)

ALTGX Setting	Description	Reference
Dynamic Reconfiguration Channel Internal and Interface Settings		
How should the receivers be clocked?	Select one of the following available options: ■ **Share a single transmitter core clock between receivers** ■ **Use the respective channel transmitter core clocks** ■ **Use the respective channel receiver core clocks**	"Clocking/Interface Options" section in the *Stratix IV Dynamic Reconfiguration* chapter.
How should the transmitters be clocked?	Select one of the following available options: ■ **Share a single transmitter core clock between transmitters** ■ **Use the respective channel transmitter core clocks**	"Clocking/Interface Options" section in the *Stratix IV Dynamic Reconfiguration* chapter.
Create an 'rx_revbitorderwa' input port to use receiver enable bit reversal	This optional input port allows you to dynamically reverse the bit order at the output of the receiver word aligner.	"Word Aligner" section in the *Stratix IV Transceiver Architecture* chapter.
Check a control box to use the corresponding control port.	You can select various control and status signals depending on what protocol(s) you intend to dynamically reconfigure the transceiver channel to.	"FPGA Fabric-Transceiver Channel Interface Selection" section in the *Stratix IV Dynamic Reconfiguration* chapter.

Protocol Settings

This section describes the various screens available to set up the PCS blocks of the Stratix IV transceiver.

Protocol Settings are not available for Basic (PMA Direct) functional mode.

Based on the protocol you select in the **General** screen of Parameter Settings, the screens listed in Table 1–10 become available.

Table 1–10. Protocol Settings

Protocols	Protocol Settings Screens		
	8B/10B	**Word Aligner**	**Rate match/Byte order**
Basic	√(Basic/8B10B)	√	√
Deterministic Latency	√(Det. Latency/8B10B)	√	—
SDI	√(SDI/8B10B)	√	—
Serial RapidIO	√(Serial RapidIO/8B10B)	√	√

The following sections describe these screens and the available settings for each of them.

8B10B Screen for the Protocol Settings

Figure 1–12 shows the **8B10B** screen of the MegaWizard Plug-In Manager for the Protocol Settings.

Figure 1–12. MegaWizard Plug-In Manager—ALTGX (8B10B Screen)

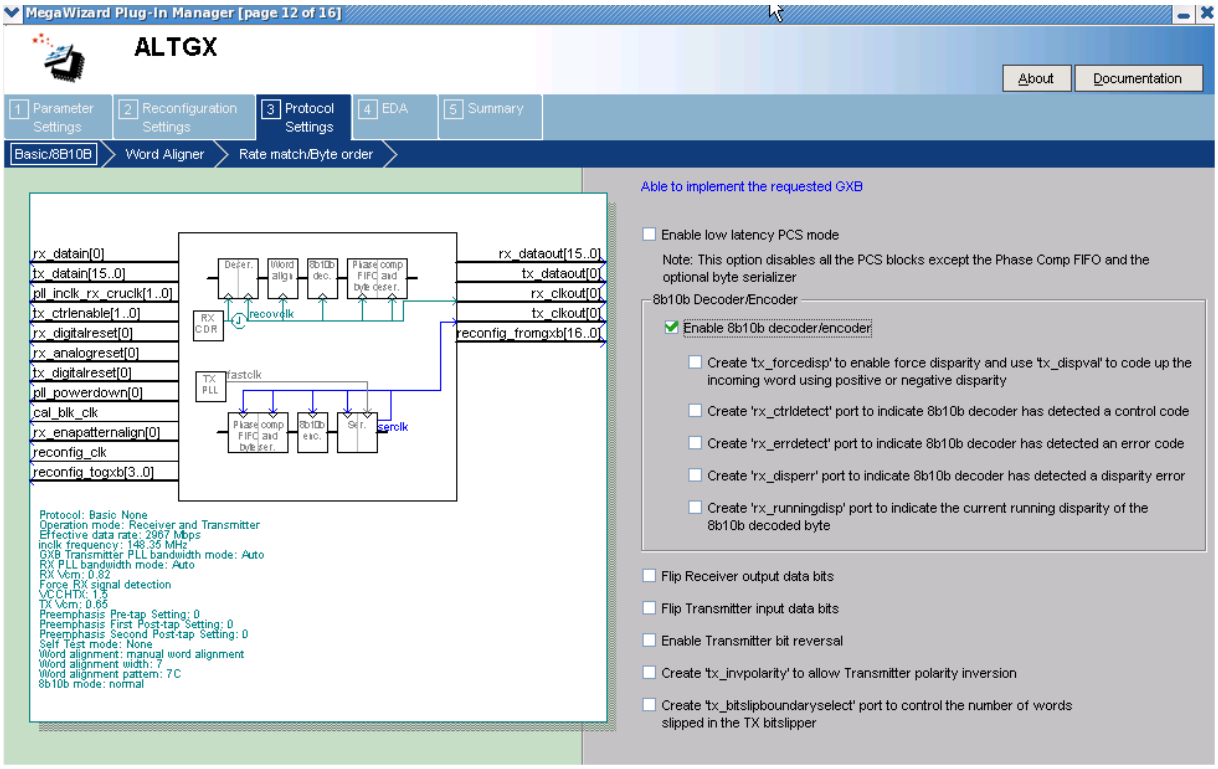


Table 1–11 describes the available options on the **8B10B** screen of the MegaWizard Plug-In Manager for your ALTGX custom megafunction variation.

Table 1–11. MegaWizard Plug-In Manager Options (8B10B Screen) (Part 1 of 3)

ALTGX Setting	Description	Reference
Enable low latency PCS mode.	This option disables all the PCS blocks except the TX/RX Phase Comp FIFO and optional byte serializer/de-serializer.	"Low Latency PCS Datapath" section in the *Stratix IV Transceiver Architecture* chapter.
Enable 8B/10B decoder/encoder.	This option is available if the channel width is 8-bits, 16-bits, or 32-bits.	"8B/10B Decoder" section in the *Stratix IV Transceiver Architecture* chapter.
Create a `tx_forcedisp` to enable Force disparity and use `tx_dispval` to code up the incoming word using positive or negative disparity.	8B/10B encoder force disparity control: ■ When asserted high—forces the 8B/10B encoder to encode the data on the `tx_datain` port with a positive or negative disparity depending on the `tx_dispval` signal level. ■ When de-asserted low—the 8B/10B encoder encodes the data on the `tx_datain` port according to the 8B/10B running disparity rules.	"8B/10B Encoder" and "Transceiver Port Lists" sections in the *Stratix IV Transceiver Architecture* chapter.

Table 1–11. MegaWizard Plug-In Manager Options (8B10B Screen) (Part 2 of 3)

ALTGX Setting	Description	Reference
Create an `rx_ctrldetect` port to indicate 8B/10B decoder has detected a control code.	This is an output status signal that the 8B/10B decoder forwards to the FPGA fabric. This signal indicates whether the decoded 8-bit code group is a data or control code group on this port. If the received 10-bit code group is one of the 12 control code groups (/Kx.y/) specified in the IEEE802.3 specification, this signal is driven high. If the received 10-bit code group is a data code group (/Dx.y/), this signal is driven low. The signal width is 1, 2, and 4 bits for a channel width of 8 bits, 16 bits, and 32 bits, respectively.	"8B/10B Decoder" section in the *Stratix IV Transceiver Architecture* chapter.
Create an `rx_errdetect` port to indicate 8B/10B decoder has detected an error code.	This is an output status signal that the 8B/10B decoder forwards to the FPGA fabric, and indicates an 8B/10B code group violation. This signal is asserted high if the received 10-bit code group has a code violation or disparity error. It is used along with the `rx_disperr` signal to differentiate between a code violation error and/or a disparity error. The signal width is 1, 2 and 4 bits for a channel width of 8 bits, 16 bits, and 32 bits, respectively.	"8B/10B Decoder" section in the *Stratix IV Transceiver Architecture* chapter.
Create an `rx_disperr` port to indicate 8B/10B decoder has detected a disparity error.	This is an output status signal that the 8B/10B decoder forwards to the FPGA fabric. This signal is asserted high if the received 10-bit code or data group has a disparity error. When this signal goes high, `rx_errdetect` is also asserted high. The signal width is 1, 2, and 4 bits for a channel width of 8 bits, 16 bits, and 32 bits, respectively.	"8B/10B Decoder" section in the *Stratix IV Transceiver Architecture* chapter.
Create an `rx_runningdisp` port to indicate the current running disparity of the 8B10B decoded byte.	This is an output status signal that the 8B/10B decoder forwards to the FPGA fabric to indicate the current running disparity of the 8B/10B decoded byte.	"8B/10B Decoder" section of Table 1-77 in the *Stratix IV Transceiver Architecture* chapter.
Flip receiver output data bits.	This option reverses the bit order of the parallel receiver data at a byte level at the output of the receiver phase compensation FIFO. For example, if the 16-bit parallel receiver data at the output of the receiver phase compensation FIFO is '10111100 10101101' (16'hBCAD), enabling this option reverses the data on `rx_dataout` port to '00111101 10110101' (16'h3DB5).	—
Flip transmitter input data bits.	This option reverses the bit order of the parallel transmitter data at a byte level at the input of the transmitter phase compensation FIFO. For example, if the 16-bit parallel transmitter data at the `tx_datain` port is '10111100 10101101' (16'hBCAD), enabling this option reverses the input data to the transmitter phase compensation FIFO to '00111101 10110101' (16'h3DB5).	—

Table 1–11. MegaWizard Plug-In Manager Options (8B10B Screen) (Part 3 of 3)

ALTGX Setting	Description	Reference
Enable transmitter bit reversal.	Enabling this option in: ■ Single-width mode—the 8-bit `D[7:0]` or 10-bit `D[9:0]` data at the input of the serializer gets rewired to `D[0:7]` or `D[0:9]`, respectively. ■ Double-width mode—the 16-bit `D[15:0]` or 20-bit `D[19:0]` data at the input of the serializer gets rewired to `D[0:15]` or `D[0:19]`, respectively. For example, if the 8-bit parallel data at the input of the serializer is '00111101', enabling this option reverses this serializer input data to '10111100.'	"Transmitter Bit Reversal" section in the *Stratix IV Transceiver Architecture* chapter.
Create a `tx_invpolarity` port to allow Transmitter polarity inversion.	This optional port allows you to dynamically reverse the polarity of every bit of the data word fed to the serializer in the transmitter data path. Use this option when the positive and negative signals of the differential output from the transmitter (`tx_dataout`) are erroneously swapped on the board.	"Transmitter Polarity Inversion" section in the *Stratix IV Transceiver Architecture* chapter.
Create `tx_bitslipboundary select` port to control the number of words slipped in the TX bitslipper.	You can only select this option when you use the **Transmitter only** or **Receiver and Transmitter** operation mode. This option enables the `tx_bitslipboundaryselect` input to control the number of bits slipped in the TX bitslipper.	—

Word Aligner Screen for the Protocol Settings

Figure 1–13 shows the **Word Aligner** screen of the MegaWizard Plug-In Manager for the Protocol Settings.

Figure 1–13. MegaWizard Plug-In Manager—ALTGX (Word Aligner Screen)

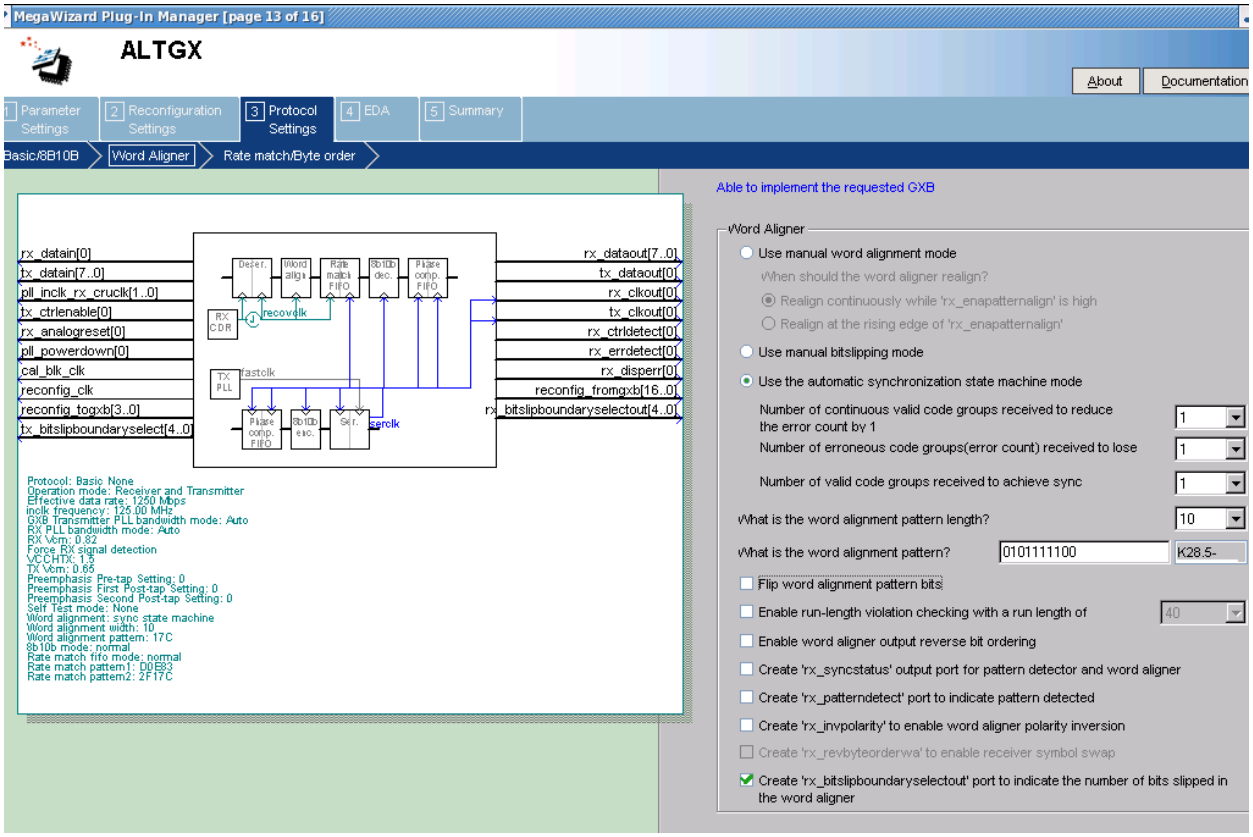


Table 1–12 describes the available options on the **Word Aligner** screen of the MegaWizard Plug-In Manager for your ALTGX custom megafunction variation.

The word aligner and rate matcher operations and patterns are pre-configured for PCI Express (PIPE), GIGE, and XAUI modes, and cannot be altered.

Table 1–12. MegaWizard Plug-In Manager Options (Word Aligner Screen) (Part 1 of 4)

ALTGX Setting	Description	Reference
Use manual word alignment mode.	Enabling this option sets the word aligner in Manual Alignment mode. In Manual Alignment mode, the word aligner operation is controlled by the input signal `rx_enapatternalign`.	"Manual Alignment Mode Word Aligner with 8-bit PMA-PCS Interface Modes" and "Manual Alignment Mode Word Aligner with 10-bit PMA-PCS Interface Modes" sections in the *Stratix IV Transceiver Architecture* chapter.
When should the word aligner realign?	Two options are available in manual mode: ■ Realign continuously while the `rx_enapatternalign` signal is high. ■ Realign at the rising edge of the `rx_enapatternalign` signal.	"Manual Alignment Mode Word Aligner with 8-bit PMA-PCS Interface Modes" and "Manual Alignment Mode Word Aligner with 10-bit PMA-PCS Interface Modes" sections in the *Stratix IV Transceiver Architecture* chapter.
Use manual bitslipping mode.	This option sets the word aligner in Bit-Slip mode. Enabling this option creates an input signal `rx_bitslip` to control the word aligner. At every rising edge of the `rx_bitslip` signal, the bit slip circuitry slips one bit into the received data stream, effectively shifting the word boundary by one bit. **SDI** Because word alignment and framing occur after de-scrambling, the word aligner in the receiver data path is not useful in SDI systems. Altera recommends driving the ALTGX `rx_bitslip` signal low to prevent the word aligner from inserting bits in the received data stream.	"Word Aligner" section in the *Stratix IV Transceiver Architecture* chapter.
Use the Automatic synchronization state machine mode.	This option sets the word aligner in Automatic Synchronization State Machine mode. This mode is available only in Single-width mode for 8B/10B encoded data: ■ 10-bit PCS-PMA Interface where the 8B/10B encoder is enabled or ■ 10-bit PCS-PMA Interface where the 8B/10B is disabled but the data is already 8B/10B encoded	"Automatic Synchronization State Machine Mode Word Aligner with 10-bit PMA-PCS Interface Mode" section in the *Stratix IV Transceiver Architecture* chapter.
Number of continuous valid code groups received to reduce the error count by 1.	Use this option in Automatic Synchronization State Machine mode to indicate the number of continuous valid code groups that it must receive between erroneous code groups to reduce the error count by one. The `rx_syncstatus` stays high as long as the error count is less than the programmed error count.	"Automatic Synchronization State Machine Mode Word Aligner with 10-bit PMA-PCS Interface Mode" section in the *Stratix IV Transceiver Architecture* chapter.

Table 1–12. MegaWizard Plug-In Manager Options (Word Aligner Screen) (Part 2 of 4)

ALTGX Setting	Description	Reference
Number of erroneous code groups (error count) received to lose synchronization.	Use this option in Automatic Synchronization State Machine mode to indicate the number of erroneous code groups (error count) that it must receive to lose synchronization. The loss-of-synch is indicated by the `rx_syncstatus` signal going low.	"Automatic Synchronization State Machine Mode Word Aligner with 10-bit PMA-PCS Interface Mode" section in the *Stratix IV Transceiver Architecture* chapter.
Number of valid code groups received to achieve synchronization.	Use this option in Automatic Synchronization State Machine mode to indicate the number of word alignment patterns that it must receive without intermediate erroneous code groups to achieve synchronization. The `rx_syncstatus` signal is driven high to indicate that synchronization has been achieved.	"Automatic Synchronization State Machine Mode Word Aligner with 10-bit PMA-PCS Interface Mode" section in the *Stratix IV Transceiver Architecture* chapter.
What is the word alignment pattern length?	This option sets the word alignment pattern length. The available choices depend on the following conditions: ■ Whether the data is 8B/10B encoded or not ■ Which mode is used in Single-width mode: → for 8-bit PCS-PMA Interface (8B/10B encoder disabled), only 16 bits are allowed. → for 10-bit PCS-PMA, 7 and 10 bits are allowed. ■ Which mode is used in Double-width mode: → for 16-bit PCS-PMA Interface (8B/10B encoder disabled), 8, 16, and 32 bits are allowed. → for 20-bit PCS-PMA Interface, 7, 10, and 20 bits are allowed.	"Word Aligner in Single-Width Mode" and "Word Aligner in Double-Width Mode" sections in the *Stratix IV Transceiver Architecture* chapter.
What is the word alignment pattern?	Enter the word alignment pattern in MSB to LSB order with MSB at the left most bit position. The length of the alignment pattern is based on the **What is the word alignment pattern length?** option. The word aligner restores the word boundary by looking for the pattern that you enter here. For example, if you want to set the word alignment pattern to /K28.5/: ■ You must enter the word alignment pattern length: **10**. ■ You must enter the word alignment pattern: **0101111100** (17C).	"Word Aligner in Single-Width Mode" and "Word Aligner in Double-Width Mode" sections in the *Stratix IV Transceiver Architecture* chapter.
Flip word alignment pattern bits.	When this option is enabled, the ALTGX MegaWizard Plug-In Manager flips the bit order of the pattern that you enter in the **What is the word alignment pattern?** option and uses the flipped version as the word alignment pattern. For example, if you enter '0101111100' (17C) as the word alignment pattern and enable this option, the word aligner uses '0011111010' as the word alignment pattern.	—

Table 1–12. MegaWizard Plug-In Manager Options (Word Aligner Screen) (Part 3 of 4)

ALTGX Setting	Description	Reference
Enable run-length violation checking with a run length of:	This option creates the output signal `rx_rlv`. Enabling this option also activates the run-length violation circuit. If the number of continuous 1s and 0s exceeds the number that you set in this option, the run-length violation circuit asserts the `rx_rlv` signal. The `rx_rlv` signal is asynchronous to the receiver data path and is asserted for a minimum of two recovered clock cycles in Single-width mode. Similarly, it is asserted for a minimum of three recovered clock cycles in Double-width mode. The run length limits are as follows: ■ Single-width mode: → 8-bit and 16-bit channel width: 4 to 128 in increments of four → 10-bit and 20-bit channel width: 5 to 160 in increments of five ■ Double-width mode: → 16-bit and 32-bit channel width: 8 to 512 in increments of eight → 20-bit and 40-bit channel width: 10 to 640 in increments of 10	"Programmable Run Length Violation Detection" section in the *Stratix IV Transceiver Architecture* chapter.
Enable word aligner output reverse bit ordering.	In manual bit-slip mode, this option creates an input port `rx_revbitorderwa` to dynamically reverse the bit order at the output of the receiver word aligner.	"Receiver Bit Reversal" section in the *Stratix IV Transceiver Architecture* chapter.
Create an `rx_syncstatus` output port for pattern detector and word aligner.	This is an output status signal that the word aligner forwards to the FPGA fabric to indicate that synchronization has been achieved. This signal is synchronous with the parallel receiver data on the `rx_dataout` port. This signal is not available in bit-slip mode. Signal width is 1, 2, and 4 bits for a channel width of 8-bits/10-bits, 16-bits/20-bits, and 32-bits/40-bits, respectively.	Table 1-77, "Word Aligner in Single-Width Mode" and "Word Aligner in Double-Width Mode" sections in the *Stratix IV Transceiver Architecture* chapter.
Create an `rx_patterndetect` port to indicate pattern detected.	This is an output status signal that the word aligner forwards to the FPGA fabric to indicate that the word alignment pattern programmed has been detected in the current word boundary. Signal width is 1, 2, and 4 bits for a channel width of 8-bits/10-bits, 16-bits/20-bits, and 32-bits/40-bits, respectively.	Table 1-77 *and* "Word Aligner in Single-Width Mode" and "Word Aligner in Double-Width Mode" sections in the *Stratix IV Transceiver Architecture* chapter.
Create an `rx_invpolarity` port to enable word aligner polarity inversion.	This optional port allows you to dynamically reverse the polarity of every bit of the received data at the input of the word aligner. Use this option when the positive and negative signals of the differential input to the receiver (`rx_datain`) are erroneously swapped on the board.	"Receiver Polarity Inversion" section in the *Stratix IV Transceiver Architecture* chapter.

Table 1–12. MegaWizard Plug-In Manager Options (Word Aligner Screen) (Part 4 of 4)

ALTGX Setting	Description	Reference
Create an `rx_revbyteorderwa` to enable Receiver symbol swap.	This is an optional input port that is available only in the Double-width mode. It creates an `rx_revbyteorderwa` port to dynamically swap the MSByte and LSByte of the data at the output of the word aligner in the receiver data path. Enabling this option compensates for the erroneous swapping of bytes at the upstream transmitter and corrects the data received by the downstream systems. For example, if the 16-bit output of the word aligner is 0B0A, asserting the `rx_revbyteorderwa` signal swaps the two bytes so the output becomes 0A0B.	"Receiver Byte Reversal in Basic Double-Width Modes" section in the *Stratix IV Transceiver Architecture* chapter.
Create `rx_bitslipboundaryselectout` port to indicate the number of bits slipped in the word aligner.	This option is available for selection only when you are in **Receiver only** or **Receiver and Transmitter** operation mode. This option enables the `rx_bitslipboundaryselectout` output to indicate the number of bits slipped in the word aligner.	—

Rate Match/Byte Order Screen for the Protocol Settings

Figure 1–14 shows the **Rate Match/Byte Order** screen of the MegaWizard Plug-In Manager for the Protocol Settings.

Figure 1–14. MegaWizard Plug-In Manager—ALTGX (Rate Match/Byte Order Screen)

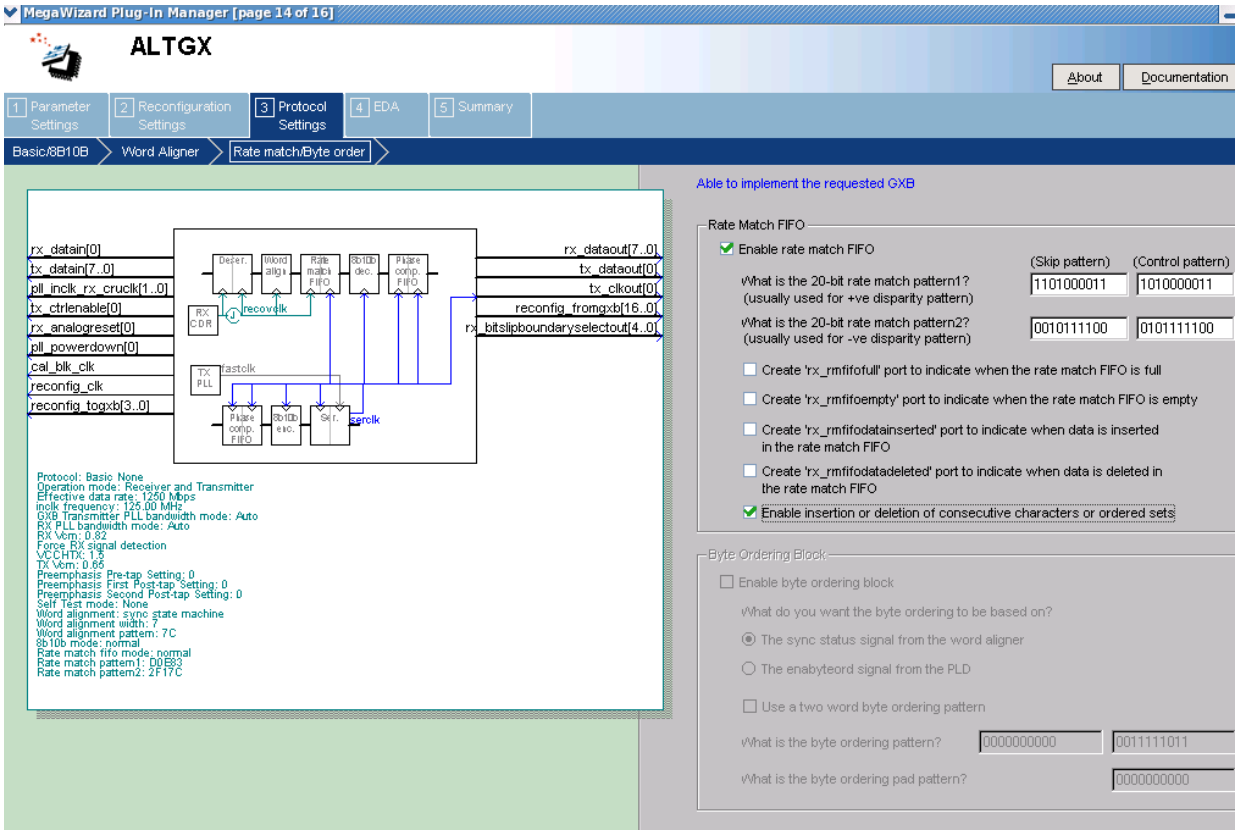


Table 1–13 describes the available options on the **Rate Match/Byte Order** screen of the MegaWizard Plug-In Manager for your ALTGX custom megafunction variation.

Table 1–13. MegaWizard Plug-In Manager Options (Rate Match/Byte Order Screen) (Part 1 of 3)

ALTGX Setting	Description	Reference
Enable rate match FIFO.	This option enables the rate match (clock rate compensation) FIFO. The rate match block consists of a 20-word deep FIFO. Depending on the PPM difference, the rate match FIFO controls insertion and deletion of skip characters based on the 20-bit rate match pattern you enter in the **What is the 20-bit rate match pattern1?** and **What is the 20-bit rate match pattern2?** options. To enable this block: ■ The transceiver channel must have both the transmitter and the receiver channels instantiated. You must select the **Receiver and Transmitter** option in the **What is the operation mode?** field in the **General** screen. ■ You must also enable the 8B/10B encoder/decoder in the **8B10B** screen. The rate match block is capable of compensating up to ±300 PPM difference between the upstream transmitter clock and the local receiver's input reference clock.	"Rate Match FIFO in Basic Single-Width Mode" and "Rate Match FIFO in Basic Double-Width Mode" sections in the *Stratix IV Transceiver Architecture* chapter.
What is the 20-bit rate match pattern1? (usually used for +ve disparity pattern)	Enter a 10-bit skip pattern and a 10-bit control pattern. In the **skip pattern** field, you must choose a 10-bit code group that has neutral disparity. When the rate matcher receives the 10-bit control pattern followed by the 10-bit skip pattern, it inserts or deletes the 10-bit skip pattern as necessary to avoid rate match FIFO overflow or underflow conditions. *(1)*	"Rate Match FIFO in Basic Single-Width Mode" and "Rate Match FIFO in Basic Double-Width Mode" sections in the *Stratix IV Transceiver Architecture* chapter.
What is the 20-bit rate match pattern2? (usually used for -ve disparity pattern)	Enter a 10-bit skip pattern and a 10-bit control pattern. In the **skip pattern** field, you must choose a 10-bit code group that has neutral disparity. When the rate matcher receives the 10-bit control pattern followed by the 10-bit skip pattern, it inserts or deletes the 10-bit skip pattern as necessary to avoid rate match FIFO overflow or underflow conditions. *(1)*	"Rate Match FIFO in Basic Single-Width Mode" and "Rate Match FIFO in Basic Double-Width Mode" sections in the *Stratix IV Transceiver Architecture* chapter.
Create the `rx_rmfifofull` port to indicate when the rate match FIFO is full.	This option creates the output port `rx_rmfifofull` when you enable the **Enable Rate Match FIFO** option. It is a status flag that the rate match block forwards to the FPGA fabric. It indicates when the rate match FIFO block is full (20 words). This signal remains high as long as the FIFO is full. It is asynchronous to the receiver data path.	"Rate Match FIFO in Basic Single-Width Mode" and "Rate Match FIFO in Basic Double-Width Mode" sections in the *Stratix IV Transceiver Architecture* chapter.

Table 1–13. MegaWizard Plug-In Manager Options (Rate Match/Byte Order Screen) (Part 2 of 3)

ALTGX Setting	Description	Reference
Create the `rx_rmfifoempty` port to indicate when the rate match FIFO is empty.	This option creates the output port `rx_rmfifoempty` when you enable the **Enable Rate Match FIFO** option. It is a status flag that the rate match block forwards to the FPGA fabric. It indicates when the rate match FIFO block is empty (5 words full). This signal remains high as long as the FIFO is empty. It is asynchronous to the receiver data path.	"Rate Match FIFO in Basic Single-Width Mode" and "Rate Match FIFO in Basic Double-Width Mode" sections in the *Stratix IV Transceiver Architecture* chapter.
Create the `rx_rmfifodatainserted` port to indicate when data is inserted in the rate match FIFO.	This option creates the output port `rx_rmfifodatainserted` flag when you enable the **Enable Rate Match FIFO** option. It is a status flag that the rate match block forwards to the FPGA fabric. This indicates the insertion of skip patterns. For every deletion, this signal is high for one parallel clock cycle.	"Rate Match FIFO in Basic Single-Width Mode" and "Rate Match FIFO in Basic Double-Width Mode" sections in the *Stratix IV Transceiver Architecture* chapter.
Create the `rx_rmfifodatadeleted` port to indicate when data is deleted in the rate match FIFO.	This option creates the output port `rx_rmfifodatadeleted` flag when you enable the **Enable Rate Match FIFO** option. It is a status flag that the rate match block forwards to the FPGA fabric. This indicates the deletion of skip patterns. For every insertion, this signal is high for one parallel clock cycle.	"Rate Match FIFO in Basic Single-Width Mode" and "Rate Match FIFO in Basic Double-Width Mode" sections in the *Stratix IV Transceiver Architecture* chapter.
Enable insertion or deletion of consecutive characters or ordered sets	This option enables the back-to-back insertion or deletion of skip characters in the rate match FIFO. This option is available for selection in Single-width mode. It is enabled by default in Double-width mode.	—
Enable byte ordering block.	This option enables the byte ordering block. It is available in both Single-width and Double-width modes. It is available only when the channel width is: ■ 16-bits/20-bits in Single-width mode ■ 32-bits/40-bits in Double-width mode As soon as the byte ordering block sees the rising edge of the appropriate signal, it compares the LSByte coming out of the byte deserializer with the byte ordering pattern. If they do not match, the byte ordering block inserts the pad character that you enter in the **What is the byte ordering pad pattern?** option such that the byte ordering pattern is seen in the LSByte position. Inserting this pad character enables the byte ordering block to restore the correct byte order.	"Byte Ordering Block" section in the *Stratix IV Transceiver Architecture* chapter.
What do you want the byte ordering to be based on?	This option is available only when the byte ordering block is enabled. This option allows you to trigger the byte ordering block on the rising edge of either the `rx_syncstatus` signal or the user-controlled `rx_enabyteord` signal from the FPGA fabric.	"Byte Ordering Block" section in the *Stratix IV Transceiver Architecture* chapter.
What is the byte ordering pattern?	This option is available only when the byte ordering block is enabled. Enter the 10-bit pattern that the byte ordering block must place in the LSByte position of the receiver parallel data on the `rx_dataout` port.	"Byte Ordering Block" section in the *Stratix IV Transceiver Architecture* chapter.

Table 1–13. MegaWizard Plug-In Manager Options (Rate Match/Byte Order Screen) (Part 3 of 3)

ALTGX Setting	Description	Reference
What is the byte ordering pad pattern?	When the byte ordering block does not find the byte ordering pattern in the LSByte position of the data coming out of the byte deseriazlier, it inserts this byte ordering pad pattern such that the byte ordering pattern is seen in the LSByte position of the receiver parallel data on the `rx_dataout` port. Inserting this pad character enables the byte ordering block to restore the correct byte order.	"Byte Ordering Block" section in the *Stratix IV Transceiver Architecture* chapter.

Note to Table 1–13:

(1) If you want the rate matcher to insert or delete both the positive and negative disparities of the 20-bit rate matching pattern, enter the positive disparity as pattern1 and negative disparity as pattern2.

Protocol Settings Screen for GIGE and XAUI

Figure 1–15 shows the Protocol Settings screen for the **GIGE** and **XAUI** modes of the MegaWizard Plug-In Manager.

Figure 1–15. MegaWizard Plug-In Manager—ALTGX (Protocol Settings Screen—GIGE and XAUI)

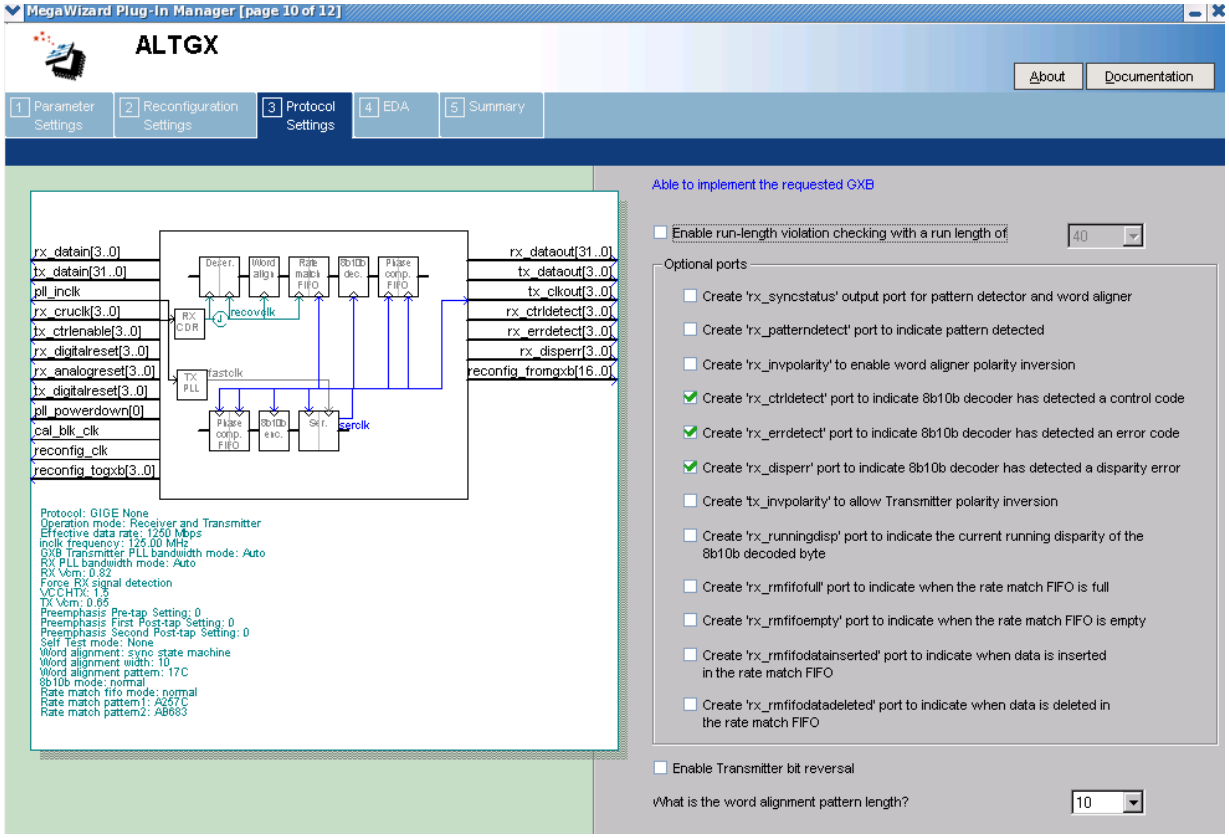


Table 1–14 describes the available options for the **GIGE** and **XAUI** modes in the Protocol Settings screen of the MegaWizard Plug-In Manager for your ALTGX custom megafunction variation.

Table 1–14. MegaWizard Plug-In Manager Options (Protocol Settings —GIGE and XAUI) (Part 1 of 3)

ALTGX Setting	Description	Reference
Enable run-length violation checking with a run length of __.	This option creates the output signal `rx_rlv`. Enabling this option also activates the run-length violation circuit. If the number of continuous 1s and 0s exceeds the number that you set in this option, the run-length violation circuit asserts the `rx_rlv` signal. The `rx_rlv` signal is asynchronous to the receiver data path and is asserted for a minimum of two recovered clock cycles. The run length limits are 5 to 160 in increments of five.	"Programmable Run Length Violation Detection" section in the *Stratix IV Transceiver Architecture* chapter.
Create an `rx_syncstatus` output port for pattern detector and word aligner.	This is an output status signal that the word aligner forwards to the FPGA fabric to indicate that synchronization has been achieved. This signal is synchronous with the parallel receiver data on the `rx_dataout` port. Receiver synchronization is indicated on the `rx_syncstatus` port of each channel.	Table 1-33 and the "Word Aligner" section in the *Stratix IV Transceiver Architecture* chapter.
Create an `rx_patterndetect` port to indicate pattern detected.	This is an output status signal that the word aligner forwards to the FPGA fabric to indicate that the word alignment pattern programmed has been detected in the current word boundary.	Table 1-33 and the "Word Aligner" section in the *Stratix IV Transceiver Architecture* chapter.
Create an `rx_invpolarity` port to enable word aligner polarity inversion.	This optional port allows you to dynamically reverse the polarity of every bit of the received data at the input of the word aligner. Use this option when the positive and negative signals of the differential input to the receiver (`rx_datain`) are erroneously swapped on the board.	"Receiver Polarity Inversion" section in the *Stratix IV Transceiver Architecture* chapter.
Create an `rx_ctrldetect` port to indicate 8B/10B decoder has detected a control code.	This is an output status signal that the 8B/10B decoder forwards to the FPGA fabric. This signal indicates whether the decoded 8-bit code group is a data or control code group on this port. If the received 10-bit code group is one of the 12 control code groups (/Kx.y/) specified in IEEE802.3 specification, this signal is driven high. If the received 10-bit code group is a data code group (/Dx.y/), this signal is driven low.	"8B/10B Decoder" section in the *Stratix IV Transceiver Architecture* chapter.
Create an `rx_errdetect` port to indicate 8B/10B decoder has detected an error code.	This is an output status signal that the 8B/10B decoder forwards to the FPGA fabric. This signal indicates an 8B/10B code group violation. It is asserted high if the received 10-bit code group has a code violation or disparity error. It is used along with the `rx_disperr` signal to differentiate between a code violation error and/or a disparity error.	"8B/10B Decoder" section in the *Stratix IV Transceiver Architecture* chapter.

Table 1–14. MegaWizard Plug-In Manager Options (Protocol Settings —GIGE and XAUI) (Part 2 of 3)

ALTGX Setting	Description	Reference
Create an `rx_disperr` port to indicate 8B/10B decoder has detected a disparity error.	This is an output status signal that the 8B/10B decoder forwards to the FPGA fabric.This signal is asserted high if the received 10-bit code or data group has a disparity error. When this signal goes high, `rx_errdetect` also is asserted high.	"8B/10B Decoder" section in the *Stratix IV Transceiver Architecture* chapter.
Create a `tx_invpolarity` port to allow Transmitter polarity inversion.	This optional port allows you to dynamically reverse the polarity of every bit of the data word fed to the serializer in the transmitter data path. Use this option when the positive and negative signals of the differential output from the transmitter (`tx_dataout`) are erroneously swapped on the board.	"Transmitter Polarity Inversion" section in the *Stratix IV Transceiver Architecture* chapter.
Create an `rx_runningdisp` port to indicate the current running disparity of the 8B/10B decoded byte.	This is an output status signal that the 8B/10B decoder forwards to the FPGA fabric. This signal is asserted high when the current running disparity of the 8B/10B decoded byte is negative. This signal is low when the current running disparity of the 8B/10B decoded byte is positive.	—
Create an `rx_rmfifofull` port to indicate when the rate match FIFO is full.	This option creates the output port `rx_rmfifofull`. It is a status flag that the rate match block forwards to the FPGA fabric. This indicates when the rate match FIFO block is full (20 words). This signal remains high as long as the FIFO is full and is asynchronous to the receiver data path.	"Rate Match (Clock Rate Compensation) FIFO" section in the *Stratix IV Transceiver Architecture* chapter.
Create an `rx_rmfifoempty` port to indicate when the rate match FIFO is empty.	This option creates the output port `rx_rmfifoempty`. It is a status flag that the rate match block forwards to the FPGA fabric. This indicates when the rate match FIFO block is empty (5 words). This signal remains high as long as the FIFO is empty and is asynchronous to the receiver data path.	"Rate Match (Clock Rate Compensation) FIFO" section in the *Stratix IV Transceiver Architecture* chapter.
Create an `rx_rmfifodatainserted` port to indicate when data is inserted in the rate match FIFO.	This option creates the output port `rx_rmfifodatainserted` flag. It is a status flag that the rate match block forwards to the FPGA fabric. The `rx_rmfifodatainserted` flag is asserted when a rate match pattern byte is inserted to compensate for the PPM difference in reference clock frequencies between the upstream transmitter and the local receiver.	"Rate Match (Clock Rate Compensation) FIFO" section in the *Stratix IV Transceiver Architecture* chapter.
Create an `rx_rmfifodatadeleted` port to indicate when data is deleted in the rate match FIFO.	This option creates the output port `rx_rmfifodatadeleted`. It is a status flag that the rate match block forwards to the FPGA fabric. The `rx_rmfifodatadeleted` flag is asserted when a rate match pattern byte is inserted to compensate for the PPM difference in reference clock frequencies between the upstream transmitter and the local receiver.	"Rate Match (Clock Rate Compensation) FIFO" section in the *Stratix IV Transceiver Architecture* chapter.

Table 1–14. MegaWizard Plug-In Manager Options (Protocol Settings —GIGE and XAUI) (Part 3 of 3)

ALTGX Setting	Description	Reference
Enable transmitter bit reversal.	Enabling this option reverses every bit of the 10-bit parallel data at the input of the serializer. The 10-bit input to the serializer `D[9:0]` is reversed to `D[0:9]`.	"8B/10B Encoder" section in the *Stratix IV Transceiver Architecture* chapter.
What is the word alignment pattern length?	This option sets the word alignment pattern length. The available choices are **7** and **10** for the GIGE and XAUI modes. The default setting for this option is **10**.	"Rate Match (Clock Rate Compensation) FIFO" section in the *Stratix IV Transceiver Architecture* chapter.

Protocol Settings Screen for the (OIF) CEI Phy Interface

Table 1–15 describes the available options for the **(OIF) CEI Phy Interface** mode in the Protocol Settings screen of the MegaWizard Plug-In Manager for your ALTGX custom megafunction variation.

Table 1–15. MegaWizard Plug-In Manager Options (Protocol Settings - [OIF] CEI PHY Interface)

ALTGX Setting	Description	Reference
Enable run-length violation checking with a run length of __.	This option creates the output signal `rx_rlv`. Enabling this option also activates the run-length violation circuit. If the number of continuous 1s and 0s exceeds the number that you set in this option, the run-length violation circuit asserts the `rx_rlv` signal. The `rx_rlv` signal is asynchronous to the receiver data path and is asserted for a minimum of two recovered clock cycles. For a 32-bit channel width, the run length limits are 8 to 512 in increments of eight.	"Programmable Run Length Violation Detection" section in the *Stratix IV Transceiver Architecture* chapter.

Protocol Settings Screen for PCI Express (PIPE)

Figure 1–16 shows the **PCI Express (PIPE) 1** screen for Protocol Settings of the MegaWizard Plug-In Manager.

Figure 1–16. MegaWizard Plug-In Manager—ALTGX (PCI Express [PIPE] 1 Screen)

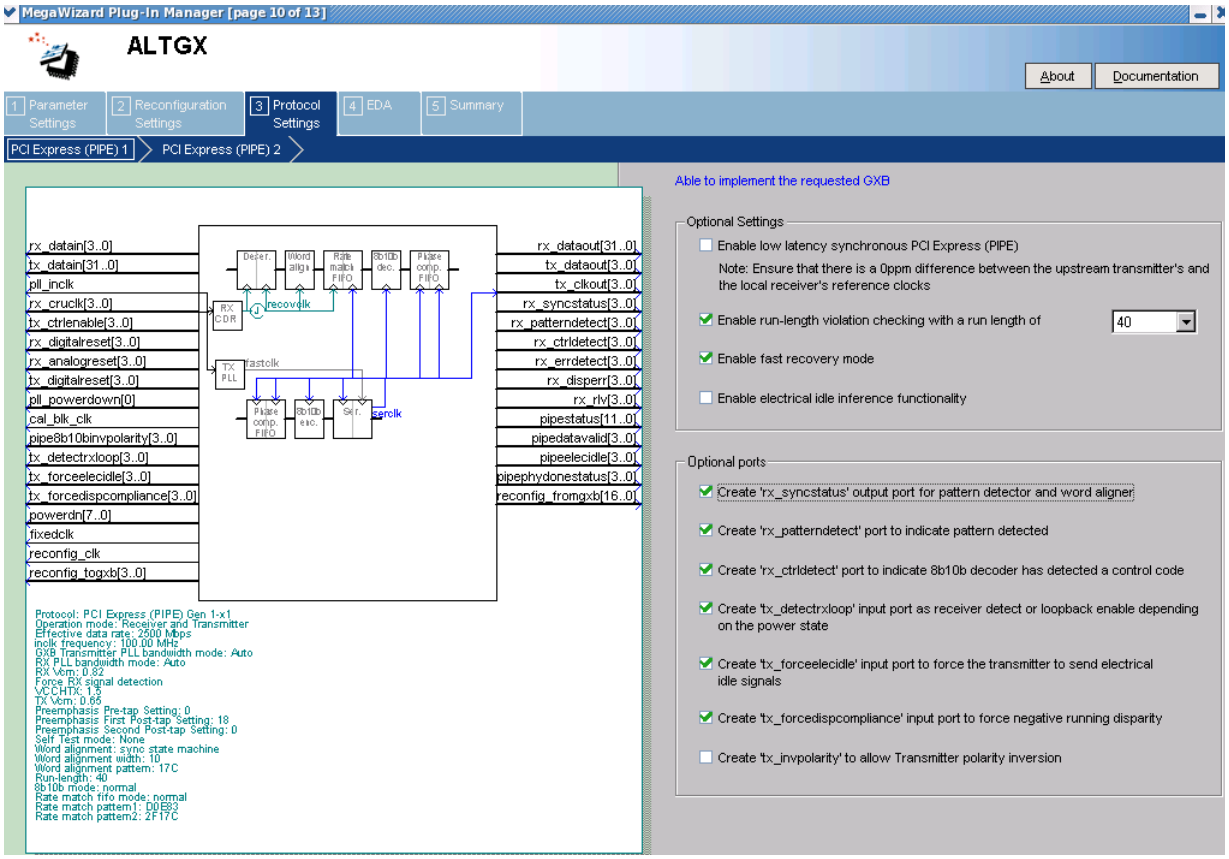


Table 1–16 describes the available options on the **PCI Express (PIPE) 1** screen of the MegaWizard Plug-In Manager for your ALTGX custom megafunction variation.

Table 1–16. MegaWizard Plug-In Manager Options (PCI Express [PIPE] 1) (Part 1 of 3)

ALTGX Setting	Description	Reference
Enable low latency synchronous PCI Express (PIPE).	This option puts the rate match FIFO into low latency mode, which forces the system into a 0 ppm mode. Ensure that there is a 0 ppm difference between the upstream transmitter's and the local receiver's input reference clocks.	"Rate Match (Clock Rate Compensation) FIFO" section in the *Stratix IV Transceiver Architecture* chapter.
Enable run-length violation checking with a run length of __.	This option creates the output signal `rx_rlv`. Enabling this option also activates the run-length violation circuit. If the number of continuous 1s and 0s exceeds the number that you set in this option, the run-length violation circuit asserts the `rx_rlv` signal. The `rx_rlv` signal is asynchronous to the receiver data path. For both 8-bit and 16-bit channel widths, the run length limits are 5 to 160 in increments of five.	"Programmable Run Length Violation Detection" section in the *Stratix IV Transceiver Architecture* chapter.
Enable fast recovery mode.	This option enables the CDR control block. When this block is enabled, the `rx_locktodata` and `rx_locktorefclk` signals are disabled.	"Fast Recovery Mode" section in the *Stratix IV Transceiver Architecture* chapter.
Enable electrical idle inference functionality.	Enable the electrical idle inference module by selecting this option. In PCI Express (PIPE) mode, the PCS has an optional electrical idle inference module designed to implement the electrical idle inference conditions specified in PCI Express (PIPE) base specification 2.0. Enabling this option creates the `rx_elecidleinfersel[2:0]` input signal. The electrical idle Inference module infers electrical idle depending on the logic level driven on `rx_elecidleinfersel[2:0]` input signal. For the electrical idle Inference module to correctly infer an electrical idle condition in each LTSSM sub-state, you must drive the `rx_elecidleinfersel[2:0]` signal appropriately.	"Electrical Idle Inference" section in the *Stratix IV Transceiver Architecture* chapter.
Create an `rx_syncstatus` output port for pattern detector and word aligner.	The ALTGX MegaWizard Plug-In Manager automatically configures the word aligner in Automatic Synchronization State Machine mode for PCI Express (PIPE) mode. This is an output status signal that the word aligner forwards to the FPGA fabric to indicate that synchronization has been achieved. This signal is synchronous with the parallel receiver data on the `rx_dataout` port. The signal width is 1 and 2 bits for a channel width of 8 bits and 16 bits, respectively.	Table 1-29 and "Automatic Synchronization State Machine Mode Word Aligner with 10-bit PMA-PCS Interface Mode" section in the *Stratix IV Transceiver Architecture* chapter.

Table 1–16. MegaWizard Plug-In Manager Options (PCI Express [PIPE] 1) (Part 2 of 3)

ALTGX Setting	Description	Reference
Create an `rx_patterndetect` output port to indicate pattern detected.	This is an output status signal that the word aligner forwards to the FPGA fabric to indicate that the word alignment pattern programmed has been detected in the current word boundary. The signal width is 1 and 2 bits for a channel width of 8 bits and 16 bits, respectively.	"Automatic Synchronization State Machine Mode Word Aligner with 10-bit PMA-PCS Interface Mode" section in the *Stratix IV Transceiver Architecture* chapter.
Create an `rx_ctrldetect` port to indicate 8B/10B decoder has detected a control code.	This is an output status signal that the 8B/10B decoder forwards to the FPGA fabric. This signal indicates whether the decoded 8-bit code group is a data or control code group on this port. If the received 10-bit code group is one of the 12 control code groups (/Kx.y/) specified in the IEEE802.3 specification, this signal is driven high. If the received 10-bit code group is a data code group (/Dx.y/), this signal is driven low. The signal width is 1 and 2 bits for a channel width of 8 bits and 16 bits, respectively.	"8B/10B Decoder" section in the *Stratix IV Transceiver Architecture* chapter.
Create a `tx_detectrxloop` input port as Receiver detect or loopback enable, depending on the power state.	Depending on the power-down mode, asserting this signal enables either the receiver detect operation or Loopback mode. *(1)*	"Receiver Detection" and "PCI Express (PIPE) Reverse Parallel Loopback" section in the *Stratix IV Transceiver Architecture* chapter.
Create a `tx_forceelecidle` input port to force the Transmitter to send Electrical Idle signals.	Enabling this port sets the transmitter buffer in electrical idle mode. This port is available in all PCI Express (PIPE) power-down modes and has a specific use in each mode. *(1)*	"Transmitter Buffer Electrical Idle" section in the *Stratix IV Transceiver Architecture* chapter.
Create a `tx_forcedispcompliance` input port to force negative running disparity.	A high level on this port forces the associated parallel transmitter data on the `tx_datain` port to be transmitted with negative current running disparity. ■ For 8-bit transceiver channel width configurations, you must drive `tx_forcedispcompliance[1:0]` high in the same parallel clock cycle as the first /K28.5/ of the compliance pattern on the `tx_datain` port. ■ For 16-bit transceiver channel width configurations, you must drive only the LSB of `tx_forcedispcompliance[1:0]`high in the same parallel clock cycle as /K28.5/D21.5/ of the compliance pattern on the `tx_datain` port.	"Compliance Pattern Transmission Support" section in the *Stratix IV Transceiver Architecture* chapter.

Table 1–16. MegaWizard Plug-In Manager Options (PCI Express [PIPE] 1) (Part 3 of 3)

ALTGX Setting	Description	Reference
Create a `tx_invpolarity` port to allow Transmitter polarity inversion.	This optional port allows you to dynamically reverse the polarity of every bit of the data word fed to the serializer in the transmitter data path. Use this option when the positive and negative signals of the differential output from the transmitter (`tx_dataout`) are erroneously swapped on the board.	"Transmitter Polarity Inversion" section in the *Stratix IV Transceiver Architecture* chapter.

Note to Table 1–16:

(1) Refer to the table '*Power States and Functions Allowed in Each Power State*' in the *PIPE Interface* section in the *Stratix IV Transceiver Architecture* chapter in volume 2 of the *Stratix IV Device Handbook*.

Figure 1–17 shows the **PCI Express (PIPE) 2** screen of Protocol Settings for the MegaWizard Plug-In Manager.

Figure 1–17. MegaWizard Plug-In Manager—ALTGX (PCI Express [PIPE] 2 Screen)

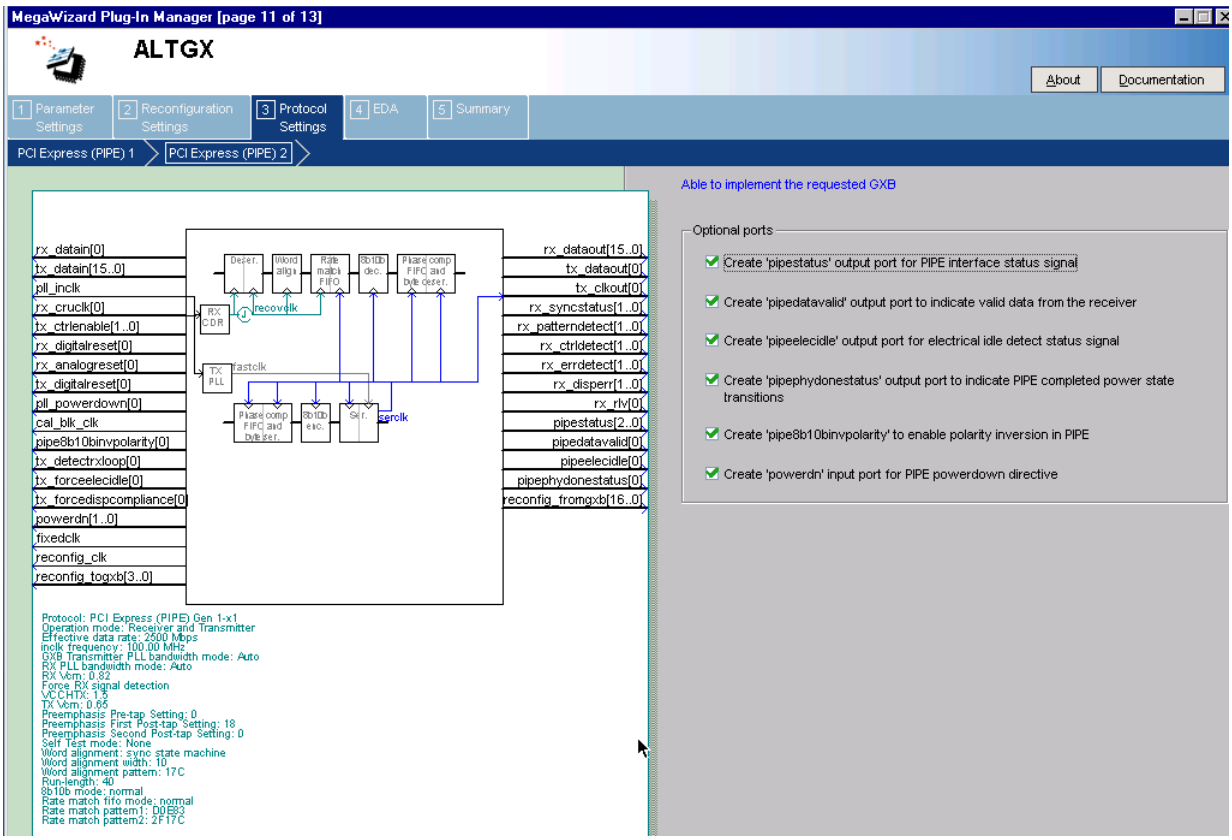


Table 1–17 describes the available options on the **PCI Express (PIPE) 2** screen of the MegaWizard Plug-In Manager for your ALTGX custom megafunction variation.

Table 1–17. MegaWizard Plug-In Manager Options (PCI Express [PIPE] 2 Screen) (Part 1 of 2)

ALTGX Setting	Description	Reference
Create a `pipestatus` output port for PIPE interface status signal.	The PCI Express (PIPE) interface block receives status signals from the transceiver channel PCS and PMA blocks and encodes the status on a 3-bit output signal (`pipestatus[2:0]`) that is forwarded to the FPGA fabric. The encoding of the status signals on `pipestatus[2:0]` is compliant to the PCI Express (PIPE) v2.00 specification: ■ 3'b000—Received data OK ■ 3'b001—One SKP symbol added ■ 3'b010—One SKP symbol deleted ■ 3'b011—Receiver detected ■ 3'b100—8B/10B decode error ■ 3'b101—Elastic buffer (rate match FIFO) overflow ■ 3'b110—Elastic buffer (rate match FIFO) underflow ■ 3'b111—Received disparity error	"Receiver Status" section and Table 1-53 in the *Stratix IV Transceiver Architecture* chapter.
Create a `pipedatavalid` output port to indicate valid data from the receiver.	This is an output status port that indicates the receiver parallel data on the `rx_dataout` port is valid.	—
Create a `pipeelecidle` output port for Electrical Idle detect status signal.	Enabling this option creates the `pipeelecidle` output status port that is forwarded to the FPGA fabric. ■ If you select **Enable Electrical Idle Inference Module,** the `pipeelecidle` signal is driven high when the electrical idle inference module infers an electrical idle condition depending on the logic driven on the `rx_elecidleinfersel[2:0]` port. Otherwise, it is driven low. ■ If you do not select **Enable Electrical Idle Inference Module**, the `rx_signaldetect` output signal from the signal threshold detection circuitry is inverted and driven on the `pipeelecidle` port. The `pipeelecidle` signal is asynchronous to the receiver data path.	"Electrical Idle Inference" section in the *Stratix IV Transceiver Architecture* chapter.
Create a `pipephydonestatus` output port to indicate PIPE completed power state transitions.	This is an output status signal forwarded to the FPGA fabric. The completion of various PHY functions; for example, receiver detection, power state transition, clock switch, and rate switch, are indicated on this `pipephydonestatus` signal by driving this signal high for one parallel clock cycle.	"PCI Express (PIPE) Mode" section in the *Stratix IV Transceiver Architecture* chapter.

Table 1–17. MegaWizard Plug-In Manager Options (PCI Express [PIPE] 2 Screen) (Part 2 of 2)

ALTGX Setting	Description	Reference
Create a `pipe8b10binvpolarity` port to enable polarity inversion in PIPE.	This optional port allows you to dynamically reverse every bit of the received data at the input of the 8B/10B decoder.	"PCI Express (PIPE) Mode" section in the *Stratix IV Transceiver Architecture* chapter.
Create a `powerdn` input port for PIPE powerdown directive.	Enabling this option creates an input control port `powerdn[1:0]` for each transceiver channel. Depending on the logic levels driven on this port, the PCI Express (PIPE) Interface block drives the transceiver channel into one of the following power states: 2'b00—P0 (Normal operation) 2'b01—P0s (low recovery time power saving 2'b10—P1 (high recovery time power saving) 2'b11—P2 (lowest power saving state)	"Power State Management" section and Table 1-51 in the *Stratix IV Transceiver Architecture* chapter.

Protocol Settings Screen for SONET/SDH

Figure 1–18 shows the **SONET/SDH** screen for Protocol Settings of the MegaWizard Plug-In Manager.

Figure 1–18. MegaWizard Plug-In Manager—ALTGX (Protocol Settings—SONET/SDH)

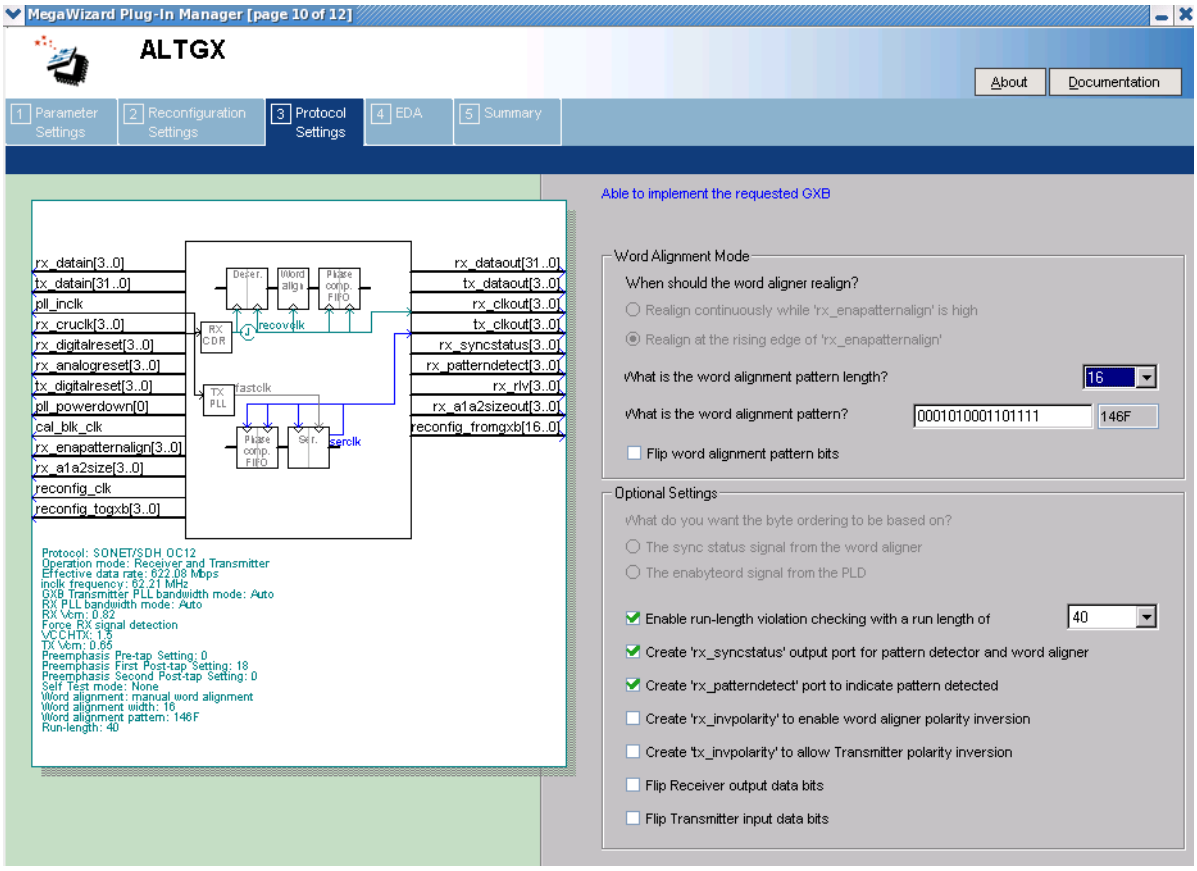


Table 1–18 describes the available options on the **SONET/SDH** screen for Protocol Settings of the MegaWizard Plug-In Manager for your ALTGX custom megafunction variation.

Table 1–18. MegaWizard Plug-In Manager Options (SONET/SDH Screen) (Part 1 of 3)

ALTGX Setting	Description	Reference
When should the word aligner realign?	This option is not available in SONET/SDH mode. In SONET/SDH mode, the word aligner operates in Manual Alignment mode. By default, the ALTGX MegaWizard Plug-In Manager sets the behavior of the word aligner such that re-alignment occurs when there is a rising edge of the `rx_enapatternalign` input signal in this mode.	"Word Aligner" section in the *Stratix IV Transceiver Architecture* chapter.
What is the word alignment pattern length?	This option sets the length of the word alignment pattern. The following options are available: ■ **OC-12**—only 16-bit pattern is allowed. ■ **OC-48**—only 16-bit pattern is allowed. ■ **OC-96**—16-bit and 32-bit patterns are allowed.	"SONET/SDH Mode" (OC-12, OC-48, and OC-96) section in the *Stratix IV Transceiver Architecture* chapter.
What is the word alignment pattern?	Enter the word alignment pattern. By default, the pattern that appears in the MegaWizard Plug-In Manager is '0001010001101111' (16'h146F).	"SONET/SDH Mode" (OC-12, OC-48, and OC-96) section in the *Stratix IV Transceiver Architecture* chapter.
Flip word alignment pattern bits.	This option is enabled in the MegaWizard Plug-In Manager by default. This option reverses the order of the alignment pattern at a bit level to support MSB-to-LSB transmission in SONET/SDH mode. The ALTGX MegaWizard Plug-In Manager flips the bit order of the default word alignment pattern '0001010001101111 '(16'h146F) and uses the flipped version '1111011000101000' (16'hF628) as the word alignment pattern.	—
What do you want the byte ordering to be based on?	This option allows you to trigger the byte ordering block either on the rising edge of the `rx_syncstatus` signal or the user-controlled `rx_enabyteord` signal from the FPGA fabric. The byte ordering block is enabled only in OC-48 mode.	"Byte Ordering Block" section in the *Stratix IV Transceiver Architecture* chapter.

Table 1–18. MegaWizard Plug-In Manager Options (SONET/SDH Screen) (Part 2 of 3)

ALTGX Setting	Description	Reference
Enable run-length violation checking with a run length of.	This option creates the output signal `rx_rlv`. Enabling this option also activates the run-length violation circuit. If the number of continuous 1s and 0s exceeds the number that you set in this option, the run-length violation circuit asserts the `rx_rlv` signal. The `rx_rlv` signal is asynchronous to the receiver data path and is asserted for a minimum of two recovered clock cycles in OC-12 and OC-48 modes. Similarly, it is asserted for a minimum of three recovered clock cycles in the OC-96 mode. For the OC-12 and OC-48 modes, the run length limits are 4 to 128 in increments of four. For the OC-96 mode, the run length limits are 5 to 160 in increments of five.	"Programmable Run Length Violation Detection" section in the *Stratix IV Transceiver Architecture* chapter.
Create an `rx_syncstatus` output port for pattern detector and word aligner.	This is an output status signal that the word aligner forwards to the FPGA fabric to indicate that synchronization has been achieved. This signal is synchronous with the parallel receiver data on the `rx_dataout` port. The signal width is 1 bit, 2 bits, and 4 bits for a channel width of 8 bits, 16 bits, and 32 bits, respectively.	Table 1-77 and "Word Aligner" section in the *Stratix IV Transceiver Architecture* chapter.
Create an `rx_patterndetect` port to indicate pattern detected.	This is an output status signal that the word aligner forwards to the FPGA fabric to indicate that the word alignment pattern programmed has been detected in the current word boundary. The signal width is 1 bit, 2 bits, and 4 bits for a channel width of 8 bits, 16 bits, and 32 bits, respectively.	Table 1-33 and "Word Aligner" section in the *Stratix IV Transceiver Architecture* chapter.
Create a `rx_invpolarity` port to enable word aligner polarity inversion.	This optional port allows you to dynamically reverse the polarity of every bit of the received data at the input of the word aligner. Use this option when the positive and negative signals of the differential input to the receiver (`rx_datain`) are erroneously swapped on the board.	"Receiver Polarity Inversion" section in the *Stratix IV Transceiver Architecture* chapter.
Create a `tx_invpolarity` port to allow Transmitter polarity inversion.	This optional port allows you to dynamically reverse the polarity of every bit of the data word fed to the serializer in the transmitter data path. Use this option when the positive and negative signals of the differential output from the transmitter (`tx_dataout`) are erroneously swapped on the board.	"Transmitter Polarity Inversion" section in the *Stratix IV Transceiver Architecture* chapter.

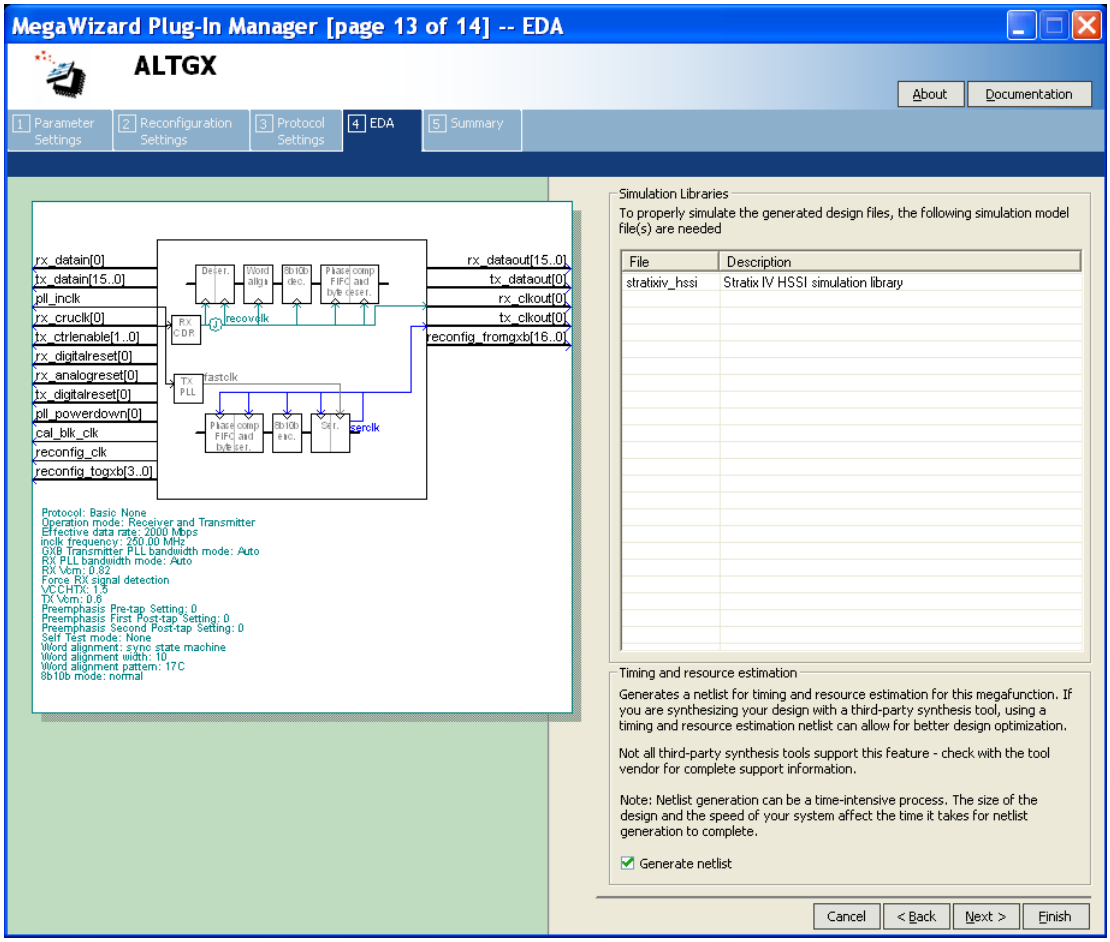
Table 1–18. MegaWizard Plug-In Manager Options (SONET/SDH Screen) (Part 3 of 3)

ALTGX Setting	Description	Reference
Flip receiver output data bits.	This option reverses the bit order of the parallel receiver data at a byte level at the output of the receiver phase compensation FIFO to support MSB-to-LSB transmission in SONET/SDH mode. For example, if the 16-bit parallel receiver data at the output of the receiver phase compensation FIFO is '10111100 10101101' (16'hBCAD), enabling this option reverses the data on the `rx_dataout` port to '00111101 10110101' (16'h3DB5).	"SONET/SDH Mode" (OC-12, OC-48, and OC-96) section in the *Stratix IV Transceiver Architecture* chapter.
Flip transmitter input data bits.	This option reverses the bit order of the parallel transmitter data at a byte level at the input of the transmitter phase compensation FIFO to support MSB-to-LSB transmission protocols in SONET/SDH mode. For example, if the 16-bit parallel transmitter data at the `tx_datain` port is '10111100 10101101' (16'hBCAD), enabling this option reverses the input data to the transmitter phase compensation FIFO to '00111101 10110101' (16'h3DB5).	"SONET/SDH Mode" (OC-12, OC-48, and OC-96) section in the *Stratix IV Transceiver Architecture* chapter.

EDA Screen

Figure 1–19 shows the EDA screen of the MegaWizard Plug-In Manager. The **Generate Netlist** option generates a netlist for the third party EDA synthesis tool to estimate timing and resource utilization for the ALTGX instance.

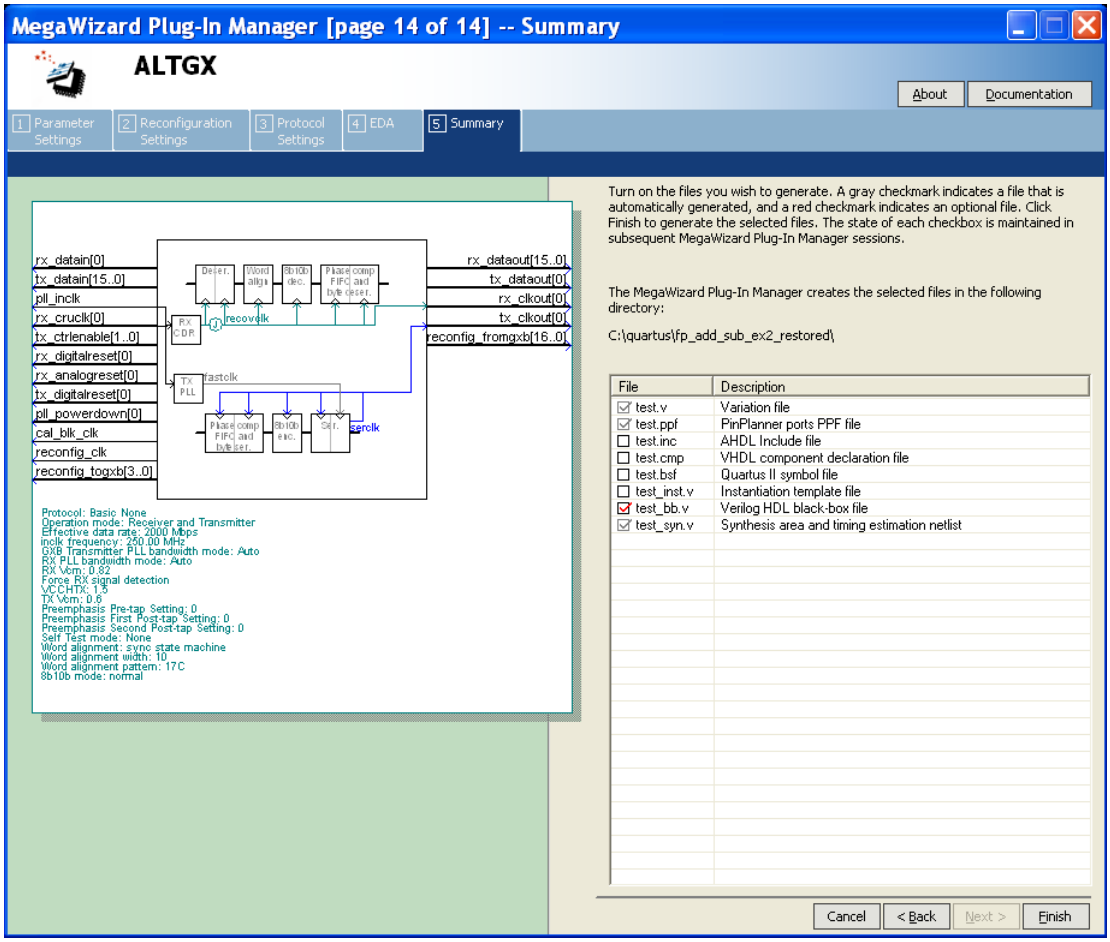
Figure 1–19. MegaWizard Plug-In Manager—ALTGX (EDA Screen)



Summary Screen

Figure 1–20 shows the Summary screen of the MegaWizard Plug-In Manager. You can select optional files on this page. After you make your selections, click **Finish** to generate the files.

Figure 1–20. MegaWizard Plug-In Manager—ALTGX (Summary Screen)



Document Revision History

Table 1–19 lists the revision history for this chapter.

Table 1–19. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
November 2009, v4.0	■ Added Deterministic Latency protocol information. ■ Added AEQ information. ■ Updated PLL setting information. ■ Consolidated Parameter Settings information (Table 1–1 to Table 1–6). ■ Consolidated Reconfiguration Settings information (Table 1–7 to Table 1–9). ■ Consolidated Protocol Settings information (Table 1–10 to Table 1–18). ■ Minor text edits.	—
June 20009, v3.1	■ Updated Table 1–9, Table 1–29 and Table 1–35. ■ Updated Figure 1–10. ■ Added introductory sentences to improve search ability. ■ Minor text edits.	—
March 2009, v3.0	■ Updated the figures to match the software changes. ■ Removed the 'Deterministic Latency' subprotocol from Basic functional mode. ■ Removed the various clock frequencies from the **Reconfig Clks** screen for all the applicable functional modes.	—
November 2008, v2.0	■ Updated Table 1–1, Table 1–6, and Table 1–11. ■ Updated Figure 1–8. ■ Added **Reconfig Clks** and **Reconfig 2** sections. ■ Added the “Use ATX Transmitter PLL” setting. ■ Changed the “Which device speed grade will you be using?” setting to the “Which device variation will you be using” setting.	—
June 2008, v1.1	Minor text edit.	—
May 2008, v1.0	Initial Release.	—

2. Transceiver Design Flow Guide

SIV53002-4.0

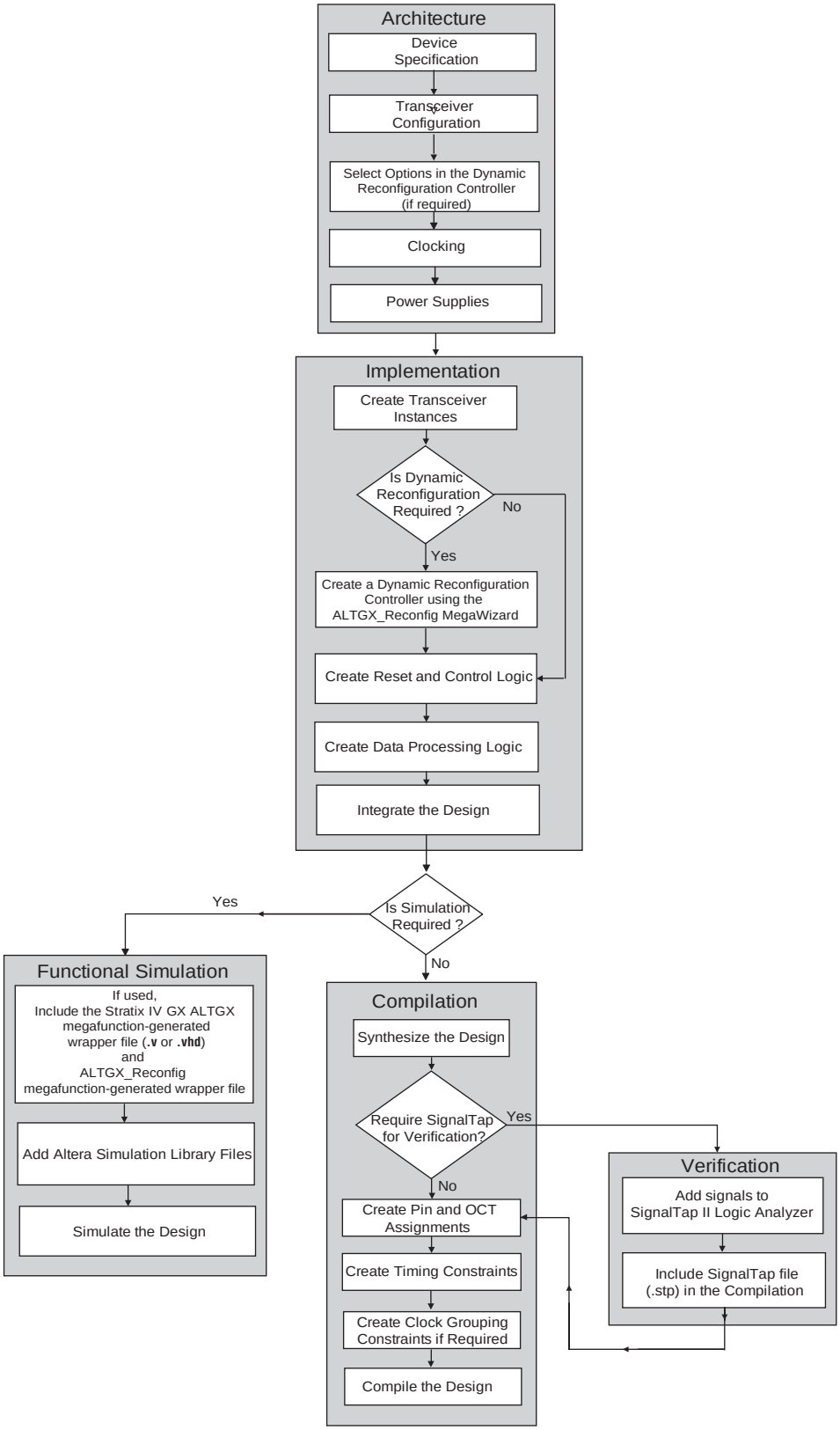
This chapter describes the Altera-recommended basic design flow that simplifies Stratix® IV GX transceiver-based designs.

Use the following design flow techniques to simplify transceiver implementation. The "Guidelines to Debug Transceiver-Based Designs" on page 2–15 provides guidelines to trouble-shoot transceiver-based designs. An example of a fibre channel protocol application is also described in this chapter. The transceiver-based design is divided into phases and are detailed in the following sections:

- "Architecture" on page 2–3
- "Implementation and Integration" on page 2–7
- "Compilation" on page 2–10
- "Verification" on page 2–12
- "Functional Simulation" on page 2–12
- "Example 1: Fibre Channel Protocol Application" on page 2–17

Figure 2–1 shows the design flow chart of the different stages of the design flow. The design flow stages include architecture, functional simulation, compilation, and verification. Each stage of the design flow are explained in the sections that follow.

Figure 2–1. Flow Chart of the Different Stages in a Transceiver-Based Design



Architecture

The first step in creating a transceiver-based design is to map your system requirements with the Stratix IV GX device supported features. The Stratix IV GX device contains multiple transceiver channels that you can configure in multiple data rates and protocols. It also provides multiple transceiver clocking options. For your design, identify the transceiver capabilities and clocking options to ensure that the transceiver meets your system requirements.

This section describes the critical parameters that you need to identify as part of this architecture phase.

Device Specification

The following device specifications must meet your requirements:

- Refer to the device data sheet to ensure that the transceivers meet the data rate and electrical requirements for your target high-speed interface application; for example, the jitter specification and voltage output differential (V_{OD}) range.
- Check whether the device family that you select supports your design requirements; for example, the number of transceiver channels, FPGA logic density, memory elements, and DSP blocks.
- If you intend to migrate to a higher logic density or higher transceiver count device in the future, ensure that the migration device is available.

For information about device characteristics, refer to the "Transceiver Performance Specifications" section in the *DC and Switching Characteristics of Stratix IV Devices* chapter. For information about transceiver resources, refer to the *Stratix IV Device Family Overview* chapter.

Transceiver Configuration

Use the ALTGX MegaWizard™ Plug-In Manager interface to configure the Stratix IV transceiver channel's features and options.

When selecting a transceiver configuration, check for the following parameters:

- Check whether the transceiver physical coding sublayer (PCS) and physical medium attachment (PMA) functional blocks comply with your system requirements. For example, check whether the rate match (clock rate compensation) FIFO in the receiver channel PCS meets the parts per million (PPM) specifications required for your application.

For more information about transceiver specifications, refer to the "Transceiver Performance Specifications" section of the *DC and Switching Characteristics of Stratix IV Devices* chapter.

- Select a configuration that meets your latency requirements. If your system has maximum latency requirements through the transceiver data path, consider the appropriate functional configuration. The Stratix IV GX transceiver supports various configurations that differ in latency (for example, low latency PCS mode and Basic [PMA direct] mode).

- In some configurations, specific functional blocks in the transceiver are disabled or bypassed. Before you select a transceiver configuration, understand the functional blocks that must be implemented in the FPGA fabric. For example, Basic (PMA direct) mode provides reduced latency but does not have PCS functional blocks enabled (for example, word aligner and 8B/10B encoder). Therefore, implement these functional blocks in the FPGA fabric if you need them in your application. Some examples of functional blocks that you may need to implement in the FPGA fabric are shown in "Create Data Processing and Other User Logic" on page 2–8.

For more information about the ALTGX MegaWizard Plug-In Manager, refer to the *ALTGX Megafunction User Guide* chapter.

- Check whether the loop-back features are available for your selected functional mode. The Stratix IV GX transceiver provides diagnostic loop-back features between the transmitter channel and the receiver channel at the transceiver PCS and PMA interfaces. These loop-back features help in debugging your design.
- If your design uses multiple transceiver channels within the same transceiver block, based on the transceiver channel configurations, the Quartus® II software might impose restrictions on combining these channels.

For more information about these restrictions, refer to the *Configuring Multiple Protocols and Data Rates in a Transceiver Block* chapter.

Dynamic Reconfiguration

You can to use the Stratix IV transceivers in multiple-link interconnect environments by allowing you to dynamically reconfigure the PMA controls (for example, V_{OD}, Pre-emphasis, Equalization, DC gain, and the transceiver channel configuration). You can reconfigure the PMA controls without affecting any other transceiver channel or the logic in the FPGA fabric.

Use the transceiver channel reconfiguration to dynamically switch a transceiver channel to multiple protocols and data rates. The Quartus II software allows you to generate a memory initialization file (**.mif**) that stores unique transceiver settings and provides a dynamic reconfiguration controller, which is soft logic that controls the transceiver reconfiguration with minimal user interface logic. You can generate this soft logic using the ALTGX_RECONFIG MegaWizard interface.

For more information about the ALTGX_RECONFIG interface, refer to the *ALTGX_RECONFIG MegaWizard Plug-In Manager* chapter.

All receiver channels in the Stratix IV GX device require offset cancellation to counter offset variations in process, voltage, and temperature (PVT) on the receiver. The dynamic reconfiguration controller initiates the sequence to perform offset cancellation on the receiver channels. Therefore, if you configure the Stratix IV GX transceiver channel in **Receiver only** or **Transmitter and Receiver** configuration, you must instantiate a dynamic reconfiguration controller.

For more information about offset cancellation or dynamic reconfiguration of PMA controls or channel configuration, refer to the "Offset Cancellation Feature" section in the *Stratix IV Dynamic Reconfiguration* chapter.

Clocking

The Stratix IV GX transceiver is clocked by various input reference clocks, for example:

- Dedicated transceiver reference clock (`refclk`) pins. Altera recommends using `refclk` pins whenever possible because the `refclk` pins yield reduced jitter on the transmitted data.
- Clock sources connected to global clock lines.
- Clock outputs from the phase-locked loops (PLLs) in the FPGA fabric.

Identify the transceiver channels input reference clock sources, for example:

- Ensure that your selected device has the required number of input reference clock resources to implement your design.
- Ensure that the transceiver clock input supports the required I/O standards.
- Ensure that the clocking restrictions work with your selected device:
  - Check whether the allowed frequencies for the transceiver input reference clocks meet your system requirements.
  - If you use the PLL cascade clock, understand its restrictions.
  - If you are using the auxiliary transmit (ATX) PLL, understand the recommendations for the input reference clock sources and the restrictions on data rate ranges supported by the ATX PLL.

For transceiver-FPGA interface clocking:

- Ensure that the transceiver-FPGA interface clock frequency limits meet your system requirements.

For information about transceiver specifications, refer to the *DC and Switching Characteristics of Stratix IV Devices* chapter.

- Identify the clocking scheme to clock the transceiver data to the logic in the FPGA fabric. For example, if your design has multiple transceiver channels that run at the same data rate and are connected to the one upstream link, you might be able to use a single transceiver-FPGA clock to provide clocks to the transceiver data path, which can conserve clock routing resources.
- If you are using Basic (PMA direct) mode, determine whether you require a left/right PLL to provide phase shifted clocks to the FPGA fabric. The left/right PLL clocks the data received and transmitted between the transceiver and the FPGA fabric interface and may be required to meet the timing requirements of the data transfer.

For information about transceiver clocking, refer to the *Stratix IV Transceiver Clocking* chapter.

After you identify the required transceiver parameters, start the implementation and integration phase.

Power Supplies

The Stratix IV GX device requires multiple power supplies. The pin connection guidelines provide specific recommendations about the type of power supply regulator (linear or switching) and the voltage supply options and restrictions. For example, the transmitter buffer supply VCCHTx has two options -1.5 V and 1.4 V. There are specific data rate restrictions when using 1.5 V. You must understand these restrictions when you select a power supply value.

For more information, refer to the *Stratix IV Pin Connection Guidelines*.

Estimate the power required to run your design. This estimation allows you to select the appropriate power supply modules and to design the power distribution network on your board.

Use the Early Power Estimator tool to estimate the transient current requirements.

For more information about the Early Power Estimation tool, refer to the *Stratix III and Stratix IV PowerPlay Early Power Estimator*.

If your design is already complete, use the power optimization features available in the Stratix IV Devices.

For more information about optimizing power in Stratix IV FPGA devices, refer to *AN 514: Power Optimization in Stratix IV FPGAs*.

Board Design Requirements

For improved signal integrity on the high-speed serial interface, follow the best design practices for your power distribution network, PCB design, and stack up.

For detailed guidelines and recommendations about your power distribution network, PCB design, and stack up, refer to the Board Design Resource Center web site.

For more information about the Stratix IV GX design process, refer to *AN 519: Stratix IV Design Guidelines*.

Implementation and Integration

There are three steps to the implementation and integration phase:

- "Create Transceiver Instances" on page 2–7
- "Create Reset Logic to Control the FPGA Fabric and Transceivers" on page 2–34
- "Create Data Processing and Other User Logic" on page 2–36

Create Transceiver Instances

The ALTGX MegaWizard Plug-In Manager to creates the transceiver instance. In the architecture phase, you identified the transceiver configuration for your design. Using the ALTGX MegaWizard Plug-In Manager, select the appropriate parameters that apply to your architecture requirements.

- Reset signals:

The ALTGX MegaWizard Plug-In Manger provides various reset and status signals:

- Reset signals—`tx_digitalreset`, `rx_digitalreset`, `rx_analogreset`, and `pll_powerdown` are required to reset the transceiver PCS and PMA functional blocks.
- Status signals—`rx_freqlocked` and `pll_locked` indicate the state of the receiver CDR and transmitter PLL, respectively. Use these reset and status signals to implement the transceiver reset control logic in the FPGA fabric. For more information, refer to "Create Reset and Control Logic" on page 2–8.

If you determine that your application requires dynamic reconfiguration, select the options in the **Reconfig** screen of the ALTGX MegaWizard interface.

If you intend to dynamically reconfigure the channel into other protocol modes or data rates, the **Reconfig** screen provides multiple options (for example, the **channel interface** and **Use alternate PLL** options) to enable this feature.

To understand the **logical channel addressing**, **logical PLL index**, and **type of reconfiguration to select** options in the **Reconfig** screen, refer to the "Channel and CMU PLL Reconfiguration Mode Details" section in the *Stratix IV Dynamic Reconfiguration* chapter.

Depending on your system, when you use multiple transceiver channels, you might be able to share the transmitter and receiver parallel clocks of one channel with the other channels. If your design requires sharing a clock resource, select the `tx_coreclk` and `rx_coreclk` ports.

Transceiver-FPGA fabric interface clock sharing conditions are provided in the *Stratix IV Transceiver Clocking* chapter.

For more information about using the ALTGX MegaWizard Plug-In Manager and the functionality of the different options and signals available, refer to the *ALTGX Megafunction User Guide* chapter.

Create Dynamic Reconfiguration Controller Instances

Use the ALTGX_RECONFIG MegaWizard interface to create the dynamic reconfiguration controller instance. If you intend to use the channel and CMU PLL reconfiguration feature, select the relevant options in the ALTGX_RECONFIG Megawizard Plug-In Manager.

For descriptions of the options in the ALTGX_RECONFIG megafunction, refer to the *Stratix IV ALTGX_RECONFIG Megafunction User Guide* chapter.

For more information about using the signals, refer to the *Stratix IV Dynamic Reconfiguration* chapter.

Create Reset and Control Logic

The reset sequence is important for initializing the transceiver functional blocks to proper operating condition. Altera recommends a reset sequence for different transceiver configurations and protocol functional modes. The ALTGX MegaWizard Plug-In Manager provides the `tx_digitalreset`, `rx_analogreset`, `rx_digitalreset`, and `pll_powerdown` signals to reset the different functional blocks of the transceiver. You can reset the CMU PLL or the ATX PLL (based on your selection) using the `pll_powerdown` signal. For transceiver instances that share the same CMU PLL or ATX PLL, the `pll_powerdown` port of these instances must be driven by the same logic.

For more information about reset sequences, refer to the *Reset Control and Powerdown* chapter.

Create Data Processing and Other User Logic

A typical transceiver-based design consists of custom data processing and other user logic that must be implemented in the FPGA fabric based on your application requirements. In addition to application-specific logic, for specific transceiver configurations, you may need additional logic to interface with the transceivers. This section provides examples of such logic.

PPM Detector When the Receiver CDR Is Used in Manual Lock Mode

Each receiver channel contains a clock data recovery (CDR) that you can use in automatic or manual lock mode.

If you use receiver CDR in manual lock mode, you can control the timing of the CDR to lock to the input reference clock using the `rx_locktorefclk` port or lock to the recovered data using the `rx_locktodata` port.

When you use the receiver CDR in manual lock mode, you may need to implement the PPM detector in the FPGA fabric to determine the PPM difference between the upstream transmitter and the Stratix IV GX receiver.

Synchronization State Machine in Manual Word Alignment Mode

Each receiver channel contains a synchronization state machine in the PCS that you can enable in certain functional modes. The synchronization state machine triggers the loss of synchronization status to the FPGA fabric based on invalid 8B/10B code groups.

However, the synchronization state machine in the PCS is not available in some functional modes. You may need to implement custom logic in the FPGA fabric to indicate the loss-of-synchronization status of the received data.

Gear Boxing Logic

Some protocols require a wider data path than provided by the transceiver interface; for example, the Interlaken Protocol requires 64/67-bit encoding and decoding, but the maximum data path interface in the Stratix IV GX transceiver is 40 bits. Therefore, you must implement gear boxing logic to interface the 64/67-bit encoder-decoder with the transceiver interface.

Functional Blocks to Interface with the Transceiver Configured in Basic (PMA Direct) Mode

In Basic (PMA direct) mode, all the PCS functional blocks in the transceiver channel are disabled. Therefore, you may need to implement the following blocks in the FPGA fabric:

- Word Alignment—To align the byte boundary on the received data.
- Byte Deserializer—To increase the data path width to the rest of the user logic and to reduce the clock frequency of the data path by two.
- Phase Compensation FIFO (for bonded channel applications)—In bonded channel applications in which multiple transceiver channels are connected to the same upstream system (for example, one Interlaken Protocol link using 24 transceiver channels). To minimize the global clock routing resources you use, implement a phase compensation FIFO to interface the receiver side of the transceiver interface with the logic in the FPGA Fabric.
  - Use the recovered clock from each channel to clock the write side of the phase compensation FIFO.
  - Use the recovered clock from any of the channels to clock the read side of the phase compensation FIFO.

  With this method, you only use one clock resource and the subsequent receive-side logic in the FPGA fabric can operate in this single clock domain.
- Deskew Logic (for bonded channel applications)—In bonded channel applications in which multiple transceiver channels are connected to the same upstream system, the data received between multiple channels are not aligned due to potential skew in the interconnect and the upstream transmitter system. To compensate for the skew, use deskew logic in the FPGA fabric.
- Encoding/Decoding or Scrambling/Descrambling—Many protocols require the transmitter data to be encoded or scrambled to maintain signal integrity. This logic may be required in the FPGA fabric based on your application requirements.

Integrate the Design

After you implement all of the required logic, integrate the transceiver instances with the remaining logic and provide the appropriate transceiver-FPGA fabric interface clocking. Synthesize the design using third-party synthesis tools, such as Synopsys Synplicity or the Quartus II software synthesis tool. This allows you to detect the syntax errors in your design.

If you are using the transceiver in Basic (PMA direct) mode, you must develop all the PCS functionality in the FPGA fabric.

Compilation

When you compile your design, the Quartus II software generates an SRAM Object File (**.sof)** or programmer object file (**.pof**) that you can download to the Stratix IV GX hardware. Typically, the first step in compiling the design is assigning pin locations for the I/Os and clocks. Use the pin planner tool in the Quartus II software to assign pins.

For a basic tutorial on the Quartus II software, open the Quartus II software, click the **Help** menu and select **Tutorial**.

- Stratix IV GX transceivers support a variety of I/O standards for the input reference clocks and serial data pins. Assign pins and the logic level standard (for example, 1.5-V PCML and LVDS) for the input and output pins.

  For more information, refer to the *I/O Features in Stratix IV Devices* chapter.

- If you share the same transceiver-FPGA fabric interface clocks for multiple transceiver channels (`tx_coreclk` and `rx_coreclk`) in your design, set the **0 ppm** constraints. These constraints enable the Quartus II software to relax the legality check restrictions on clocking.

For more information, refer to the "Common Clock Driver Selection Rules" section of the *Stratix IV Transceiver Clocking* chapter.

- For transceiver serial pins and `refclk` pins, set the on-chip termination (OCT) resistor settings.

For more information about supported OCT settings, refer to "Transmitter Output Buffer" section of the *Stratix IV Transceiver Architecture* chapter.

- Create timing constraints for the clocks and data paths. Use the TimeQuest Timing Analyzer to set timing constraints.

For more information about the TimeQuest Timing Analyzer, refer to the *Quartus II Development Software Handbook*.

- Compile the design. This generates a **.sof** that can be downloaded in the FPGA.

The Quartus II software generates multiple report files that contain information such as transceiver configuration and clock resource utilization. The following section describes the report files relevant to using transceivers and clock resource.

Report Files

The Quartus II software provides a report file in the synthesis, fitter, map, placement, and assembler stages. The report file provides useful information on the device and transceiver configuration generated by the Quartus II software. This section only describes the reports provided in the fitter stage. To access the report, click on the **Processing** menu, select the **Compilation Report** option and expand the **Fitter** tab.

Fitter Summary

The fitter summary provides high-level information on FPGA fabric resources and transceiver channels used by your design. For example, to ensure that the Quartus II software has created the number of transceiver channels as specified in your design, refer to the GXB Receiver channels and GXB Transmitter channels field at the bottom of the report. For detailed information on resource utilization, expand the **Fitter** tab.

Pin-Out File

Select the **Pin-Out file** option under the **Fitter** tab. The Quartus II software displays the I/O standards and bank numbers of all the pins (used and unused) needed to connect to the board. The Quartus II software also generates a PIN file (**.pin**) with the above information. Altera recommends that you use the **.pin** as a guideline. Use the pin connection guidelines for board layout.

For more information about pin connection guidelines for board layout, refer to *Stratix IV GX Device Family Pin Connection Guidelines*.

Resource Section

Expand the **Resource Section** option under the **Fitter** tab to view the following tabs:

- The **GXB Transmitter channel** tab—Provides generated settings for all the transmitter channels instantiated in your design.
- The **GXB Transmitter PLL** tab—Provides generated settings for all the transmitter PLLs instantiated in your design.
- The **GXB Receiver channel** tab—Provides generated settings for all the receiver channels instantiated in your design.
- The **Global and other fast signals** tab—Displays the list of clock and other signals in your design that are assigned to the global and regional clock resources.

You can use the report file to verify whether the transceiver settings (for example, data rate), are generated per your settings in the ALTGX MegaWizard Plug-In Manager.

Verification

The SignalTap® Logic Analyzer allows you to verify design functionality using the on-chip logic analyzer. SignalTap provides options to create multiple sets of signals that can be sampled using different trigger clocks. You can add the signals to the SignalTap Logic Analyzer and save the file as an STP file (**.stp**). When you include this **.stp** along with the design files and compile the design, the Quartus II software creates an **.sof** that allows you to verify the functionality of the signals that you added in the SignalTap Logic Analyzer file.

You can run the **.stp** that connects to the device through the JTAG port and displays the signal transitions using the Quartus II software. Because the JTAG port is required to run SignalTap, consider designing the board with the JTAG interface for debugging your system.

For more information about using SignalTap, refer to the *In-System Design Debugging* section in volume 3 of the *Quartus II Development Software Handbook*.

To verify the functionality of the PCS and PMA blocks, the Stratix IV GX transceiver provides diagnostic loop-back features between the transmitter and receiver channels.

For more information, refer to the "Loopback Modes" section in the *Stratix IV Transceiver Architecture* chapter.

Functional Simulation

Use the ALTGX MegaWizard-generated wrapper file to simulate the instantiated transceiver configuration in third-party simulation software such as ModelSim. For simulation, specific Altera® simulation library files are required (listed in Table 2–1). The following library files are available in VHDL and Verilog versions:

- 220pack
- 220model
- altera_mf_components
- altera_mf
- sgate_pack
- sgate
- stratixiv_hssi_component
- stratixiv_hssi_atoms

These simulation files are available under the following folder in the Quartus II installation directory: <*Quartus II installation folder*>`/eda/sim_lib`

The stratixiv_hssi_component library file is only applicable if the transceiver instance is created using VHDL.

For VHDL simulation using ModelSim, create the following libraries in your ModelSim project:

- lpm
- sgate
- altera_mf
- stratixiv_hssi

These simulation files are available under <*Quartus II installation folder\quartus\eda\sim_lib*>.

Compile the simulation files into the libraries specified in Table 2–1.

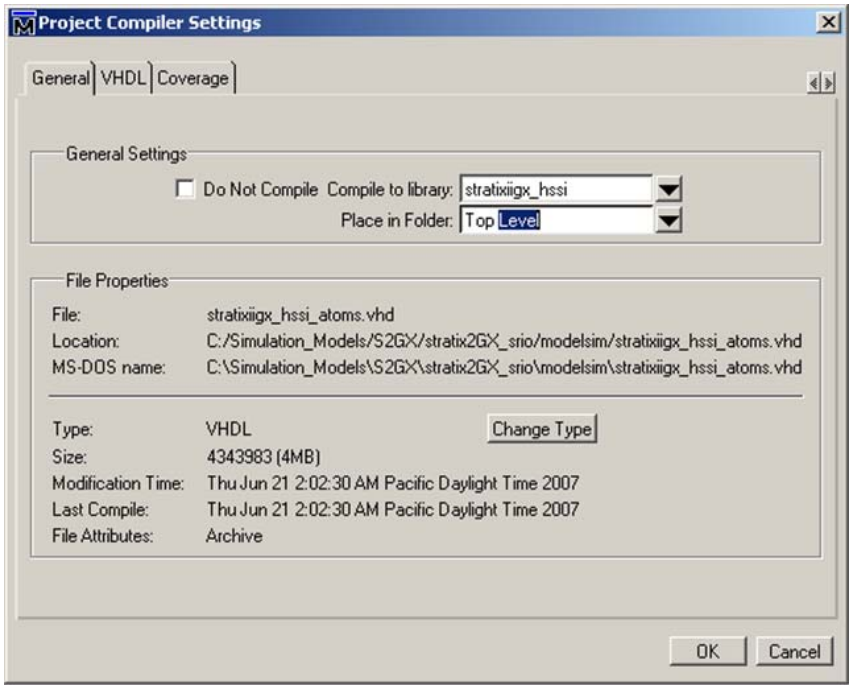
Table 2–1. Library to Compile Simulation Files

Altera Simulation Files	Library
220pack	lpm
220model	lpm
sgate pack	sgate
sgate	sgate
altera_mf_components	altera_mf
altera_mf	altera_mf
stratixiv_hssi_component	stratixiv_hssi
stratixiv_hssi_atoms	stratixiv_hssi
user design files	work

For example, to compile a file into a specific library using ModelSim, right click on the file, select **Properties**, then click the **General** tab.

In the **Compile to library** option, select the corresponding library for the file selected. Figure 2–2 shows the ModelSim window compilation of files in a specific library for the Stratix II GX device.

Figure 2–2. ModelSim Option to Compile Files in a Specific Library



Include all the libraries in the search path. Add the ALTGX and ALTGX_RECONFIG MegaWizard Plug-In Manager-generated wrapper files (**.v** or **.vhd**) and all of the design files to the library. Compile all the library files first, then the design files, and lastly run the simulation.

For Verilog simulation, add the ALTGX and ALTGX_RECONFIG MegaWizard Plug-In Manager-generated Verilog wrapper files (**.v**), the Altera library files, and all of the design files. Compile all the library files first, then the simulation model file, followed by the design files. Lastly, run the simulation.

These guidelines are further described in "Example 1: Fibre Channel Protocol Application" below.

For more information about functional RTL simulation or post-fit simulation, refer to the *Simulation* chapter in volume 3 of the *Quartus II Handbook*.

Guidelines to Debug Transceiver-Based Designs

This section provides guidelines to debug transceiver-based designs. If a system failure occurs, the first step is to ensure the functionality of the logic within the FPGA. Use the following information when you observe a system failure.

Guidelines to Debug the FPGA Logic and the Transceiver Interface

Before checking the functionality in silicon, perform functional simulation to ensure the basic functionality of the RTL and the transceiver-FPGA fabric interface.

- Understand the limitations of functional simulation. If you intend to simulate timing parameters, consider post-fit simulation. The functional simulation model for transceivers does not model timing-related parameters or uncertainties in the transceiver data path. For example, the PPM difference in the rate matcher clocks (clock rate compensation) or the phase differences between the read and write side of the phase compensation FIFO are not modeled.

  For information about functional RTL simulation or post-fit simulation, refer to the *Simulation* chapter in volume 3 of the *Quartus II Handbook*.

- Check whether the compiled design has timing violations in the TimeQuest Timing Analyzer report. Set the appropriate timing constraints on the failing paths.

  For information about using the TimeQuest Timing Analyzer, refer to the *Timing Analysis* chapter in volume 3 of the *Quartus II Handbook*.

- Verify the functionality of the transmitter and receiver data path with serial loopback. Dynamically control the serial loopback through the `rx_serialpbken` port. When this signal is asserted, data from the transmitter serializer is looped back to the receiver CDR of the channel.
- Use SignalTap to verify the behavior of the user logic and the transceiver interface signals. If you have FPGA I/O pins available for debug, you can also use the external logic analyzer to debug the functionality of the device.

  For more information, refer to the *In-System Debugging Using External Logic Analyzers* chapter in volume 3 of the *Quartus II Handbook*.

  To use these features, you must connect the JTAG configuration pins in the FPGA.

- Verify the interconnect on the receive side by configuring the transceiver in reverse serial loopback mode. In this case, the recovered data from the receiver channel is sent to the transmitter buffer. To configure a transceiver channel operating in a different configuration to reverse serial loopback mode, use the dynamic reconfiguration controller.
- Check whether the transceiver FPGA fabric interface clocking schemes follow the recommendations provided in the "FPGA Fabric-Transceiver Interface Clocking" section in the *Stratix IV Transceiver Clocking* chapter.
- Ensure that you have used the recommended transceiver reset sequence.

Guidelines to Debug System Level Issues

If you have determined that the logic in the FPGA fabric is functionally correct, check for system level issues:

- Check the voltage ripple across the 2 kΩ resistor that is connected to the RREF pin. The voltage ripple must be less than 60 mv.
- Measure the eye on the near-end and far-end of the transmitter to understand the jitter added by the transmitter and interconnect.
  - Ensure that the high-speed scopes you use for measurement have sufficient bandwidth (bandwidth rating on the scope and cables must be at least three times the serial data rate).
  - Check whether the eye meets the eye-mask requirements if specified by the protocol application.
  - Use scopes that provide information on the different jitter components to understand the possible source of the increased jitter. For example, increased intersymbol interface (ISI) indicates potential bandwidth limitations on the interconnect.

    ☞ Some scopes, such as Agilent 86100C DCA, require pre-defined patterns (for example, PRBS7 or PRBS23) to provide jitter components.
- Measure signals on the traces (no connector) using high-impedance differential probe with short leads.
- Ensure that characteristic impedance on the interconnect matches the source and load systems.
  - Check for impedance discontinuities on the trace by Time Domain Reflectometry (TDR).
  - Revisit the board design, layout, and routing for any inconsistencies that can cause impedance discontinuities.
  - Check whether the termination schemes on the Stratix IV GX device and on the upstream system are matched. Altera recommends using OCT in the Stratix IV GX device instead of external termination to improve signal integrity.
  - Change the transmit output differential voltage to improve eye amplitude.
- Compensate for high frequency losses in the interconnect by changing the equalization settings of the Stratix IV GX device and check for improvement of the bit error rate. If the upstream system does not have an equalization feature, increase the pre-emphasis (1st post tap) of the Stratix IV GX transmitter. In cases where there are multiple interconnects between the Stratix IV GX device and upstream system, use the pre-tap and 2nd post tap. Altera provides tools to select the pre-emphasis.

- Measure the increase in jitter at the near end and far end with one channel turned on at a time if you have multiple transceiver channels connected to the upstream system. This helps to observe the effect of cross talk from adjacent channels on the victim channel.
  - Check the board layout and routing to ensure that you have implemented the design practices to mitigate cross talk.
- Ensure that the input voltage and duty cycle of the input reference clock source provided to the transmitter PLLs meet the input reference clock requirements.
- Check whether the voltage drop on the power supplies is within the specified tolerance range.
  - Measure the voltage at the via beneath the power supply pin using a high-impedance probe.
  - Check whether the voltage regulator specifications meet the Stratix IV GX power supply requirements.
  - Revisit the power distribution scheme for the supply voltage to ensure that it is designed to handle the transient current requirements of the transceiver.

    For the tolerance values of the different power supplies, refer to the *Stratix IV DC and Switching Characteristics* chapter.
- Check for periodic modulation of other frequency components on the transmit data. Send a high-frequency pattern (1010) from the transmitter side and connect the transmitter serial output to a spectrum analyzer.

For more information about Stratix IV GX transceivers, refer to *AN 553: Debugging Transceivers*.

Example 1: Fibre Channel Protocol Application

Assume that you want to implement a fibre channel protocol application using three transceiver channels. Consider the following system requirements:

- You need three transceiver channels
- All the channels need to be placed in the same transceiver block
- All the channels need to have independent control to reset their PCS and PMA functional blocks

Table 2–2 shows the transceiver channel configuration for Example 1.

Table 2–2. Transceiver Channel Configuration for Example 1

Channels	Mode of Operation	Data Rate	Input Reference Clock Frequency
0	Receiver and Transmitter	FC4G (4.25 Gbps)	106.25 MHz
1	Receiver and Transmitter	FC1G (1.0625 Gbps)	53.125 MHz
2	Transmitter Only	FC4G (4.25 Gbps)	106.25 MHz

Phase 1—Architecture

In this phase, check whether the Stratix IV GX device supports or meets your design requirements.

Device Specification

Consider the questions listed in Table 2–3 before setting device-specific parameters.

Table 2–3. Device Specific Parameters

Questions	Answer
Do the parameters meet the fibre channel protocol electrical requirements?	Yes For more information, refer to the "Transceiver Performance Characteristics" section in the *DC and Switching Characteristics of Stratix IV Devices* chapter
Are three transceiver channels available?	Yes
Is there support for 4.25 Gbps and 1.0625 Gbps data rates?	Yes Two CMU PLLs are available within each transceiver block to support two different transmitter data rates. Each receiver channel contains a dedicated receiver CDR that supports 4.25 Gbps and 1.0625 Gbps data rates.

For the maximum data rates supported, refer to the "Transceiver Performance Specifications" section in the *DC and Switching Characteristics of Stratix IV Devices* chapter.

Transceiver Configuration

The fibre channel protocol uses an 8B/10B encoder and requires clock rate compensation.

Functional Blocks

Consider the questions listed in Table 2–4 before configuring the transceiver.

Table 2–4. Configuring the Transceiver

Questions	Answer
Is the 8B/10B encoder in the PCS block fibre channel compliant?	No The fibre channel protocol consists of two different End-of-Frame (EOFt) ordered sets. The correct EOFt ordered set sent by user logic depends on the ending disparity of the word preceeding the EOFt. The Stratix IV GX transceiver does not provide running disparity flags to the user logic. Therefore, the user logic might not be able to select the correct EOFt ordered set.
Is there a workaround?	Yes Implement the 8B/10B encoder in the FPGA fabric.
Is the clock rate compensation block in the PCS available without an 8B/10B encoder?	No You can implement this in the FPGA fabric.

The design requires a **Transmitter and Receiver** configuration for two channels and a **Transmitter Only** configuration for one channel (Table 2–5).

Table 2–5. Multiple Channels

Questions	Answer
Does the Stratix IV GX transceiver support these two configurations and allow you to combine them within the same transceiver block	Yes The available FPGA fabric interface width is 20 or 40 bits to support 4.25 Gbps and 1.0625 Gbps data rates, respectively. This FPGA fabric interface facilitates 8B/10B encoding and decoding in the FPGA fabric without additional re-arrangement of the received parallel data to a 10-bit boundary.

Dynamic Reconfiguration

If your application requires you to dynamically reconfigure the transceiver PMA controls, ensure that you understand the settings, options, and user logic required to enable this feature.

For more information, refer to the "Interfacing ALTGX and ALTGX_RECONFIG Instances" section in the *Stratix IV Dynamic Reconfiguration* chapter.

For more information about initiating read and write transactions, refer to the "Dynamically Reconfiguring PMA Controls" section in the *Stratix IV Dynamic Reconfiguration* chapter.

If you are using the channel reconfiguration feature, enable the appropriate options in the ALTGX and ALTGX_RECONFIG MegaWizards.

You can dynamically use the reconfiguration modes to reconfigure different functional blocks in a transceiver channel using **.mifs**. For information about generating **.mifs**, refer to the "Channel and CMU PLL Reconfiguration Mode Details" section in the *Stratix IV Dynamic Reconfiguration* chapter.

Clocking

Consider the questions listed in Table 2–6 before configuring clocking.

Table 2–6. Configuring Clocking (Part 1 of 2)

Questions	Answer
Is there support for two different input reference clocks?	Yes The Stratix IV GX transceiver has two refclk pins for each transceiver block.
Do the `refclk` pins support the required frequency range?	Yes The minimum frequency range of refclk is 50 MHz; the maximum frequency range is 622.08 MHz.

Table 2–6. Configuring Clocking (Part 2 of 2)

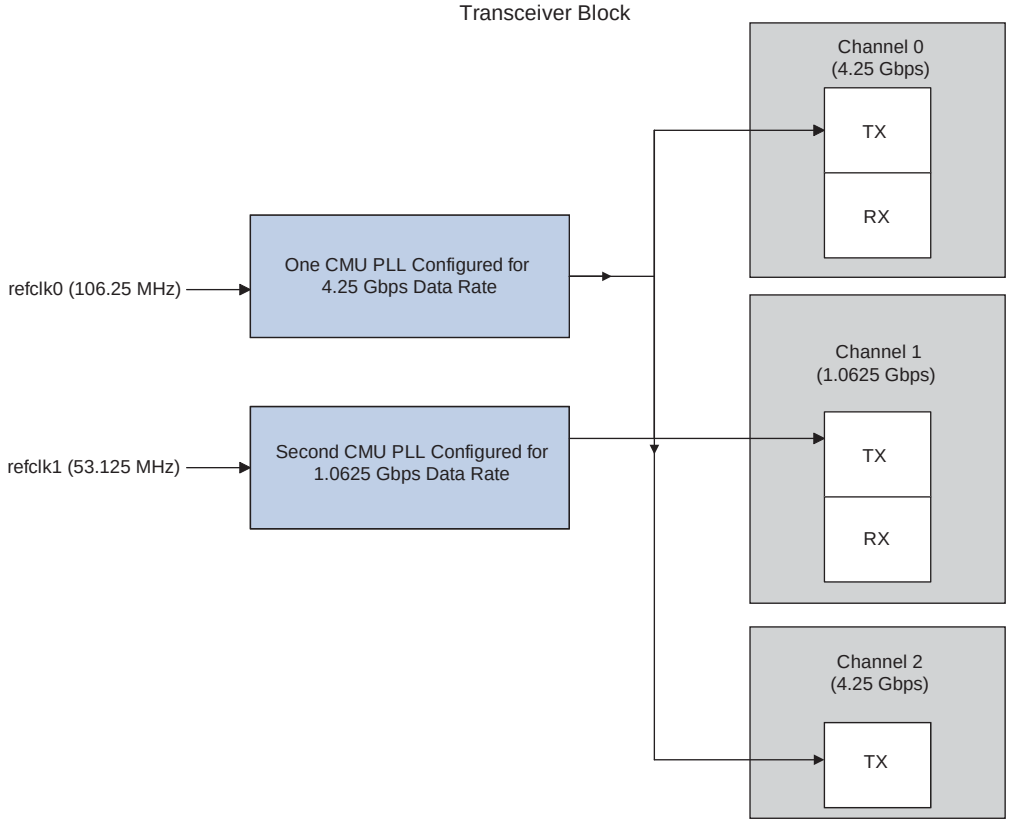
Questions	Answer
Can transceiver-FPGA fabric interface clocking be shared?	No The design requires independent control on all channels, so you must not share the transceiver-FPGA fabric interface clock of one channel with another channel. Each of the channels must use its own tx_clkout and rx_clkout signals to clock the data between the transceiver channels and the FPGA fabric.
Does the Stratix IV GX transceiver support this feature?	Yes

For more information about clocking the transmitter and receiver channel data path for this type of configuration, refer to the "Transmitter Channel Datapath Clocking" section of the *Stratix IV Transceiver Clocking* chapter.

Figure 2–3 shows the transmitter side of the transceiver setup for Example 1.

The transmitter side receives its clocks from the clock multiplier unit (CMU) PLLs. The receiver side contains its dedicated CDR that provides the high-speed serial and low-speed parallel clocks to its PMA and PCS blocks, respectively.

Figure 2–3. Top-Level Transceiver Setup—Transmitter-Side Only



Phase 2—Implementation

Create the transceiver instance using the ALTGX MegaWizard Plug-In Manager.

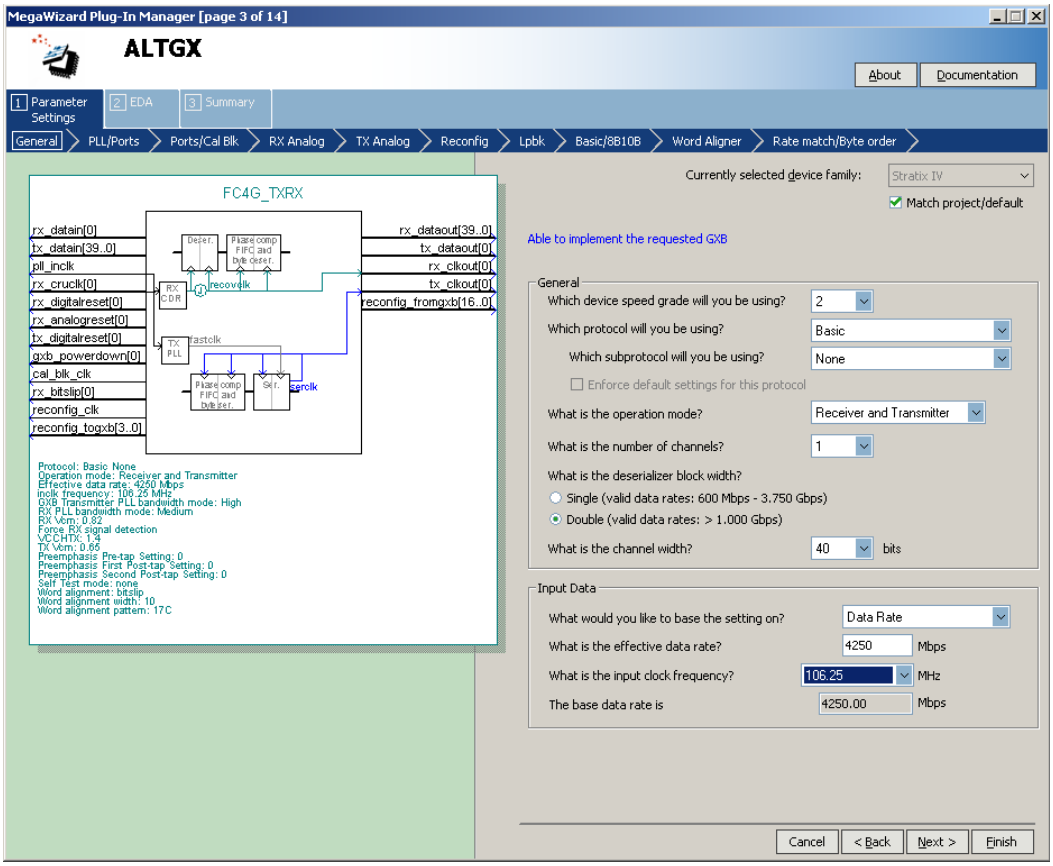
For a description of the individual options, refer to the *ALTGX Megafunction User Guide* chapter.

Create the Transceiver Instance for an FC4G Configuration (Channel 0)

Figure 2–4 through Figure 2–14 show the different options available in the ALTGX MegaWizard Plug-In Manager to create the transceiver channel instance for the FC4G data rate. Use this instance for channel 0, with the following settings:

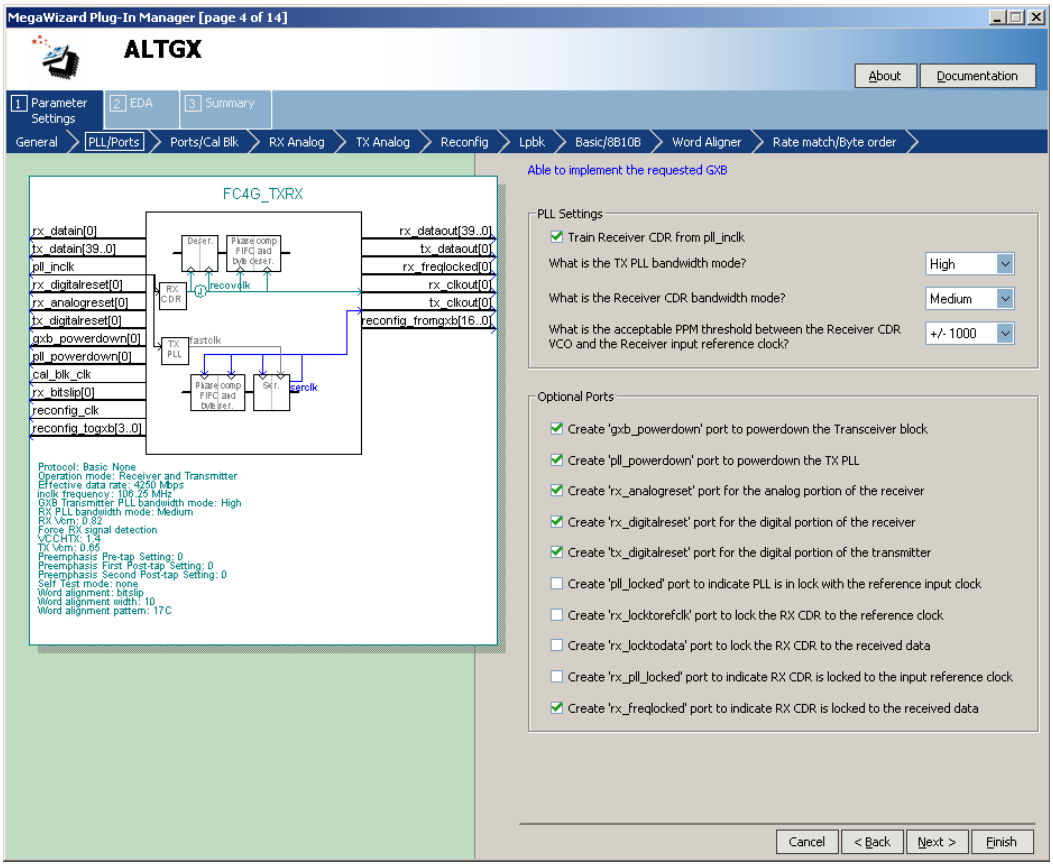
- **General** screen—You can configure the Stratix IV GX transceiver for fibre channel protocol using Basic mode. Set the options with the values shown in Figure 2–4.

Figure 2–4. FC4G Instance Settings (General Screen)



- **PLL/Ports** screen—Check the **Train Receiver CDR from PLL inclk** option, as shown in Figure 2–5. When you select this option, the same input reference clock used for the CMU PLL is provided as a training clock to the receiver CDR.

Figure 2–5. FC4G Instance Settings (PLL/Ports Screen)

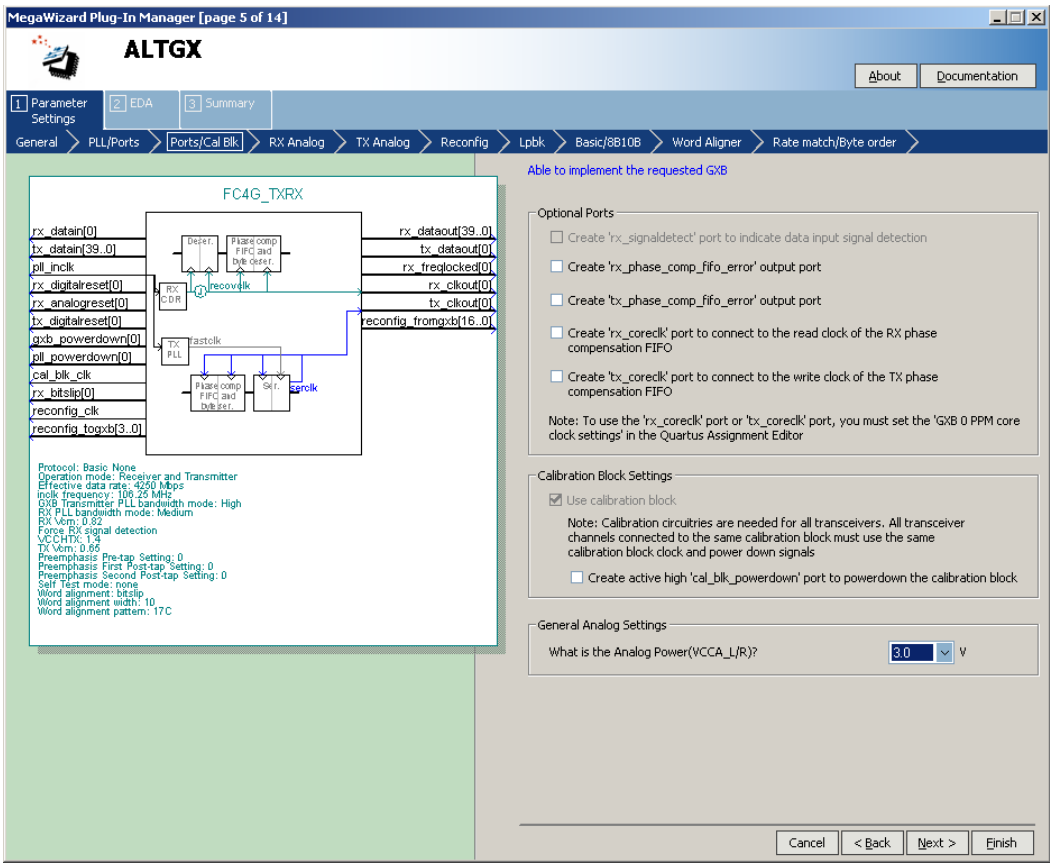


- Check the `pll_powerdown` signal. This signal allows you to power down the CMU PLL. Use this signal as part of your reset sequence.
- Check the `pll_locked` signal. This signal indicates whether the CMU PLL is locked to the input reference clock. The user logic waits until the `pll_locked` signal goes high before transmitting data.
- Check the `rx_freqlocked` signal. This signal indicates whether the receiver CDR is locked to data. When the receiver CDR is configured in automatic lock mode, assert the `rx_digitalreset` signal if the `rx_freqlocked` signal goes low to keep the receiver PCS under reset. Altera recommends specific transceiver reset sequences to ensure proper device operation.

For more information about receiver CDR and lock modes, refer to the "Receiver Channel Datapath" section of *Stratix IV Transceiver Architecture* chapter.

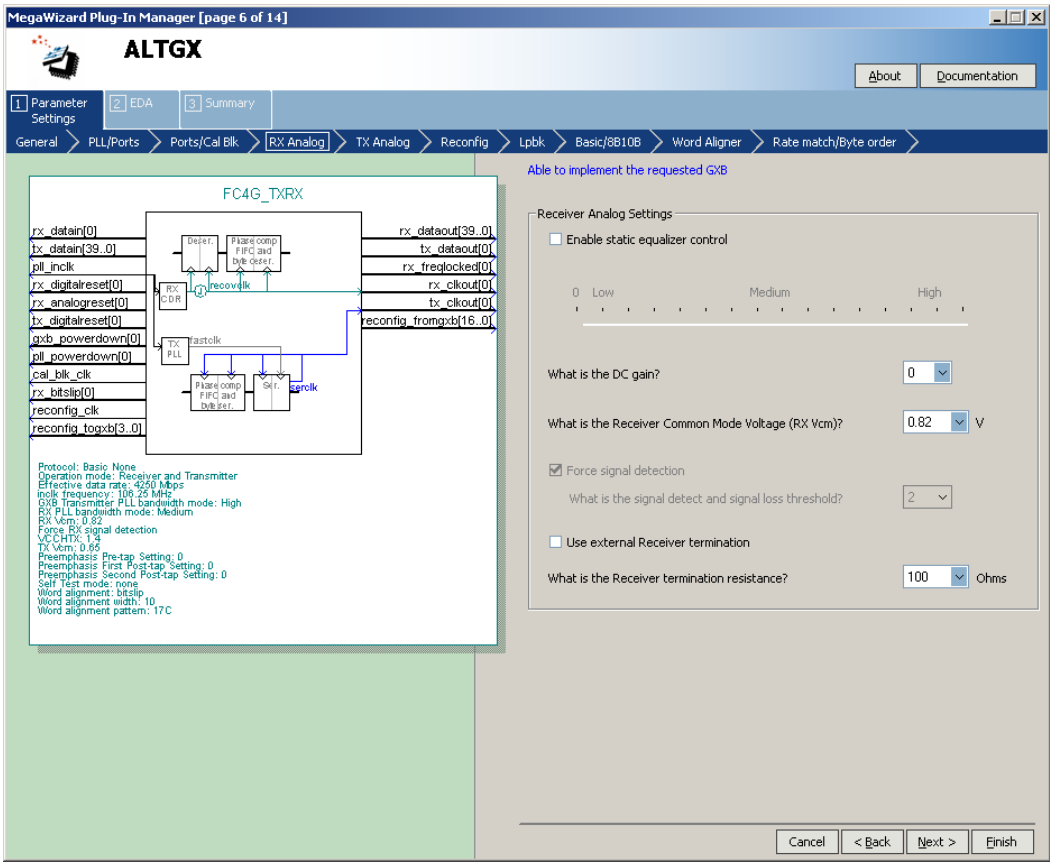
- **Ports /Cal Blk** screen—The calibration block is required so it is always enabled. Select the options shown in Figure 2–6.

Figure 2–6. FC4G Instance Settings (Ports/Cal Blk Screen)



- **RX Analog** screen—Select the options shown in Figure 2–7.

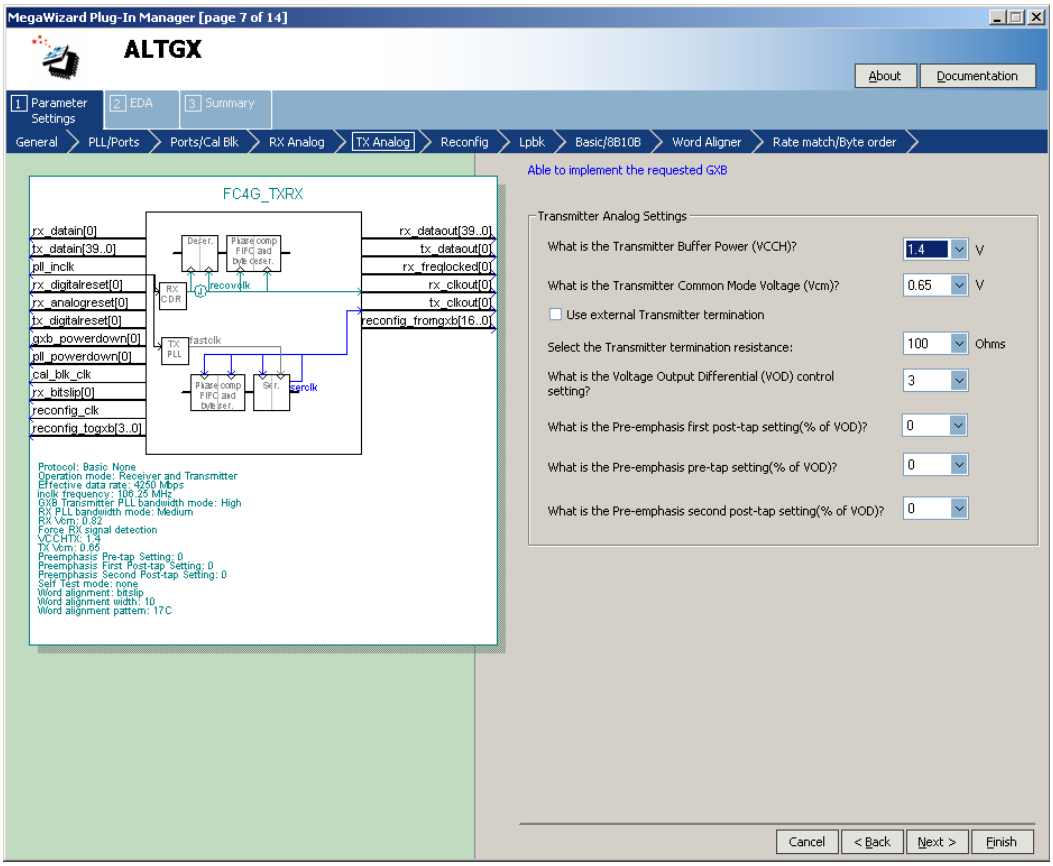
Figure 2–7. FC4G Instance Settings (RxAnalog Screen)



For a description of the individual options, refer to the *ALTGX Megafunction User Guide* chapter.

- **TX Analog** screen—Select the output differential voltage and common mode voltage values that meet the fibre channel protocol specification. If you intend to transmit data through faulty interconnects, select the pre-emphasis settings shown in Figure 2–8.

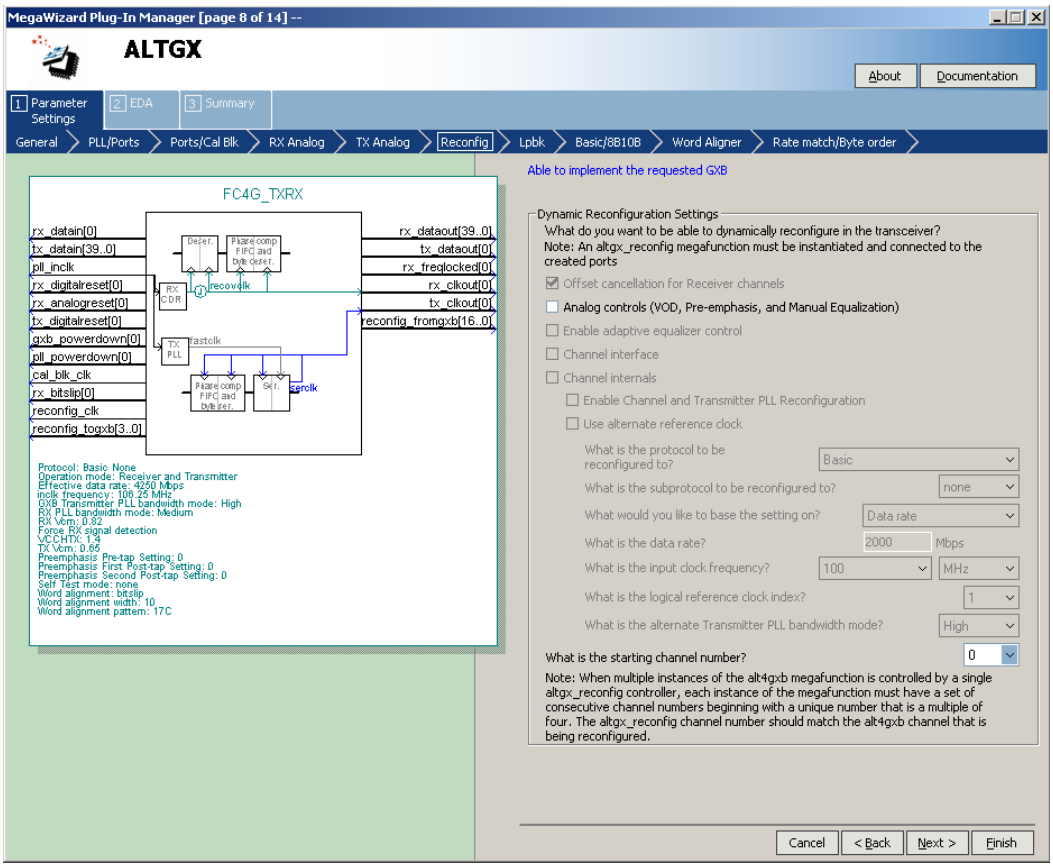
Figure 2–8. FC4G Instance Settings (TX Analog Screen)



For more information about pre-emphasis settings, refer to the *DC and Switching Characteristics of Stratix IV Devices* chapter.

- **Reconfig** screen—Set the starting channel number to **0**. Because offset cancellation is required for receiver channels, the **Offset Cancellation for Receiver Channels** option is automatically enabled. Ensure that you connect the `reconfig_fromgxb` and `reconfig_togxb` ports with the dynamic reconfiguration controller (Figure 2–9).

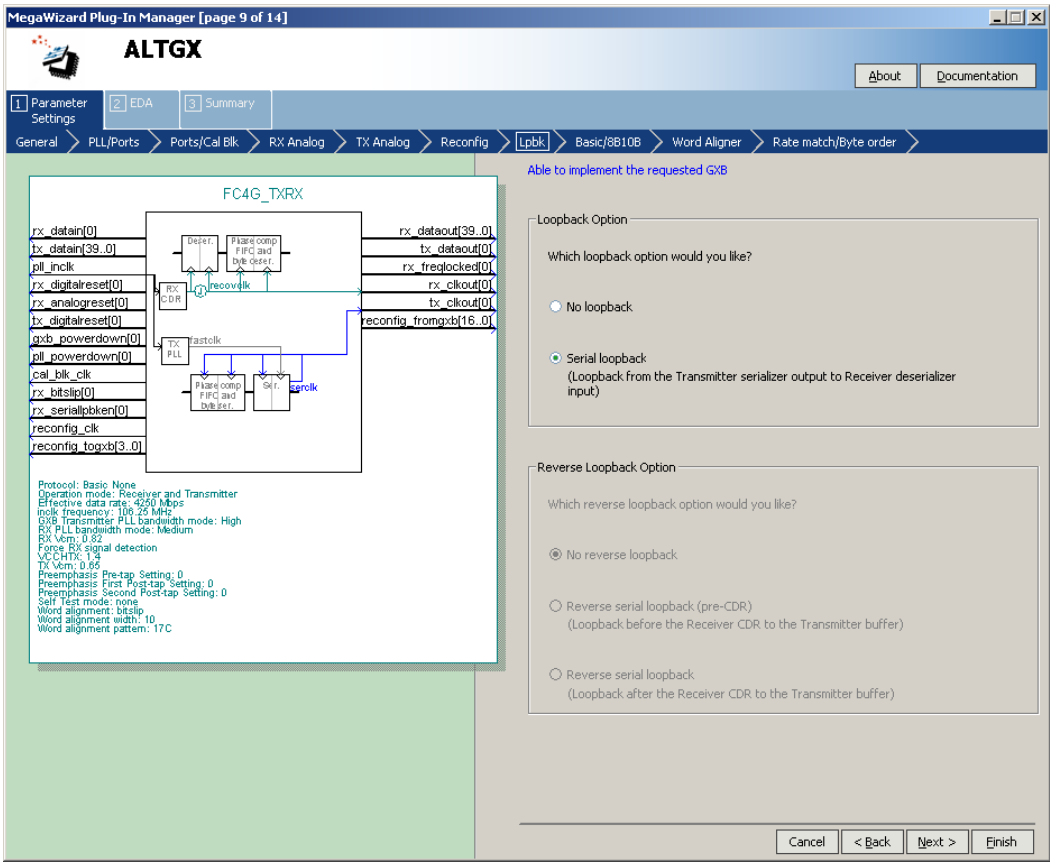
Figure 2–9. FC4G Instance Settings (Reconfig Screen)



For more information about the starting channel numbers, refer to the "Logical Channel Addressing While Reconfiguring the PMA Controls" section of the *Stratix IV Dynamic Reconfiguration* chapter.

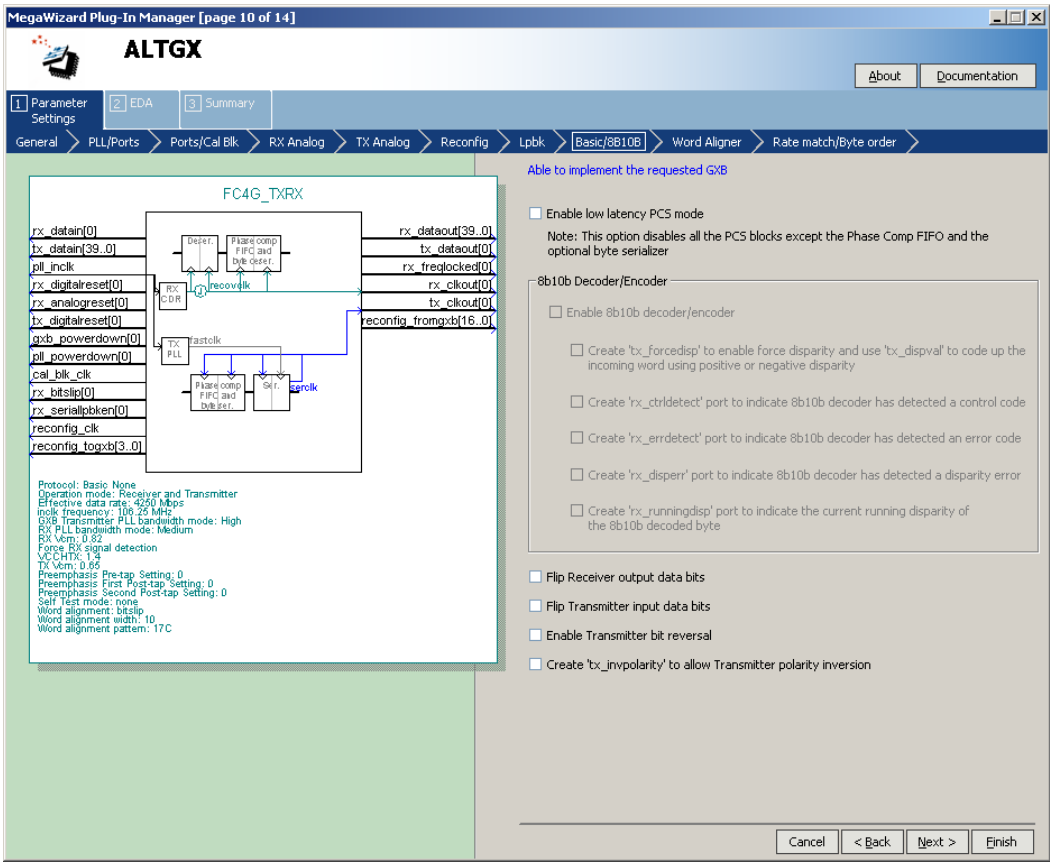
- **Lpbk** screen—The serial loopback option is enabled, as shown in Figure 2–10.

Figure 2–10. FC4G Instance Settings (Lpbk Screen)



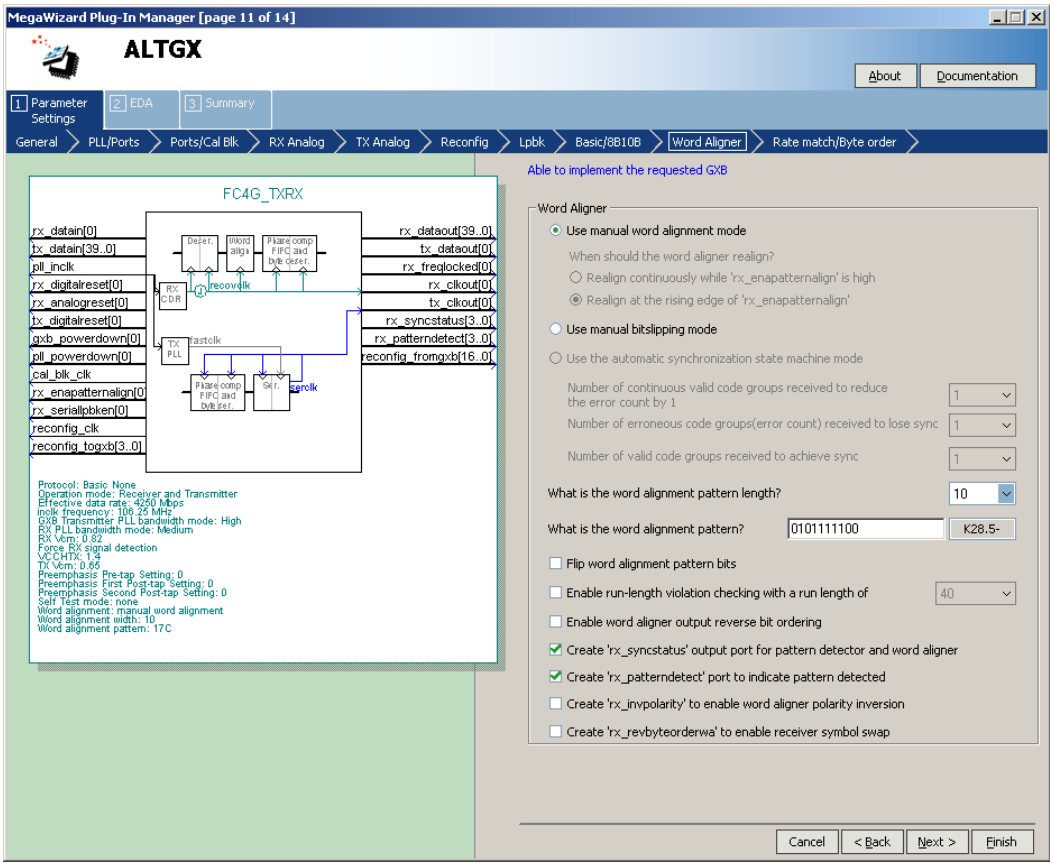
- **Basic/8B10B** screen—The Basic/8B10B screen is shown in Figure 2–11. The 8B/10B encoder is not compatible with the fibre channel protocol application; therefore, this option is unchecked.

Figure 2–11. FC4G Instance Settings (Basic 8B/10B)



- **Word Aligner** screen—The fibre channel protocol requires that you use K28.5 to align the byte boundary. In the **What is the word alignment pattern?** option, set one of the 10-bit disparity values to **K28.5**. The word aligner automatically detects when the other disparity value is received.

Figure 2–12. FC4G Instance Settings (Word Aligner Screen)



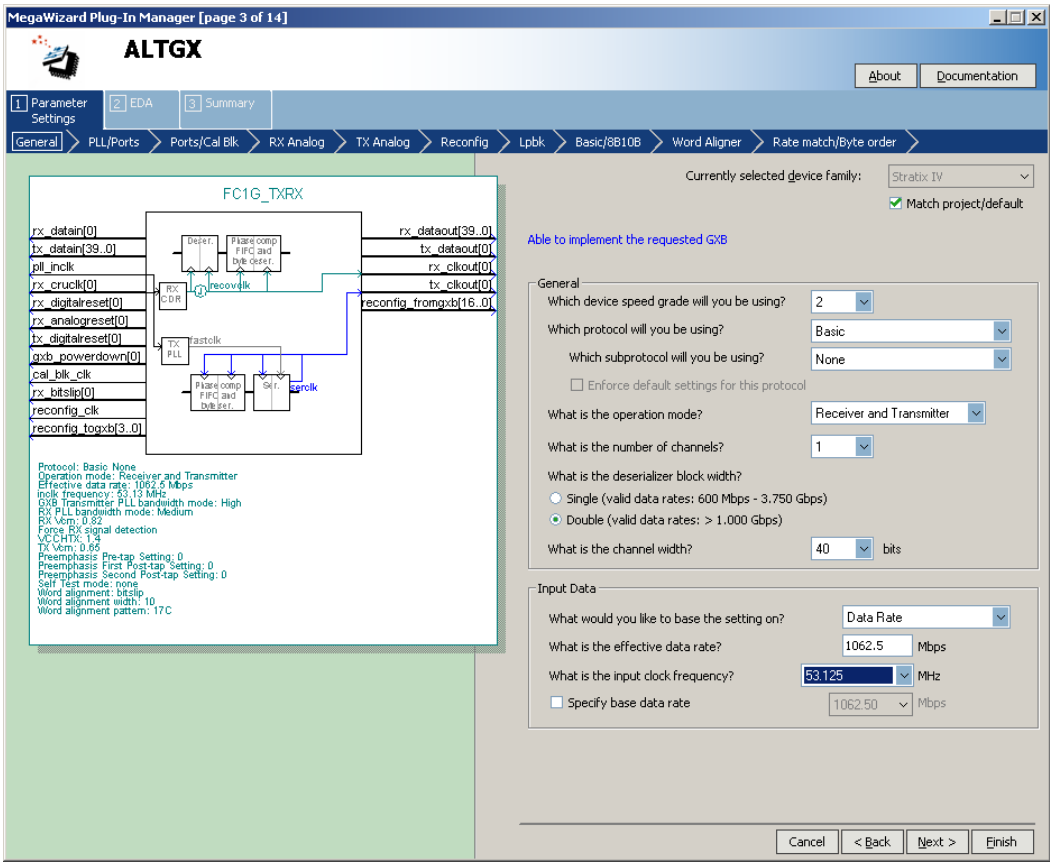
- Select the `rx_patterndetect` and `rx_syncstatus` signals. The `rx_patterndetect` signal indicates whenever the word alignment pattern is detected in the word boundary.
- Click **Finish** to exit the ALTGX MegaWizard Plug-In Manager.

Create the Transceiver Instance for an FC1G Configuration (Channel 1)

Creating the instance for FC1G is very similar to that of the FC4G configuration, with the following changes:

- **General** screen—Set the values shown in Figure 2–13.

Figure 2–13. FC1G Instance (Channel 1) Settings (General Screen)



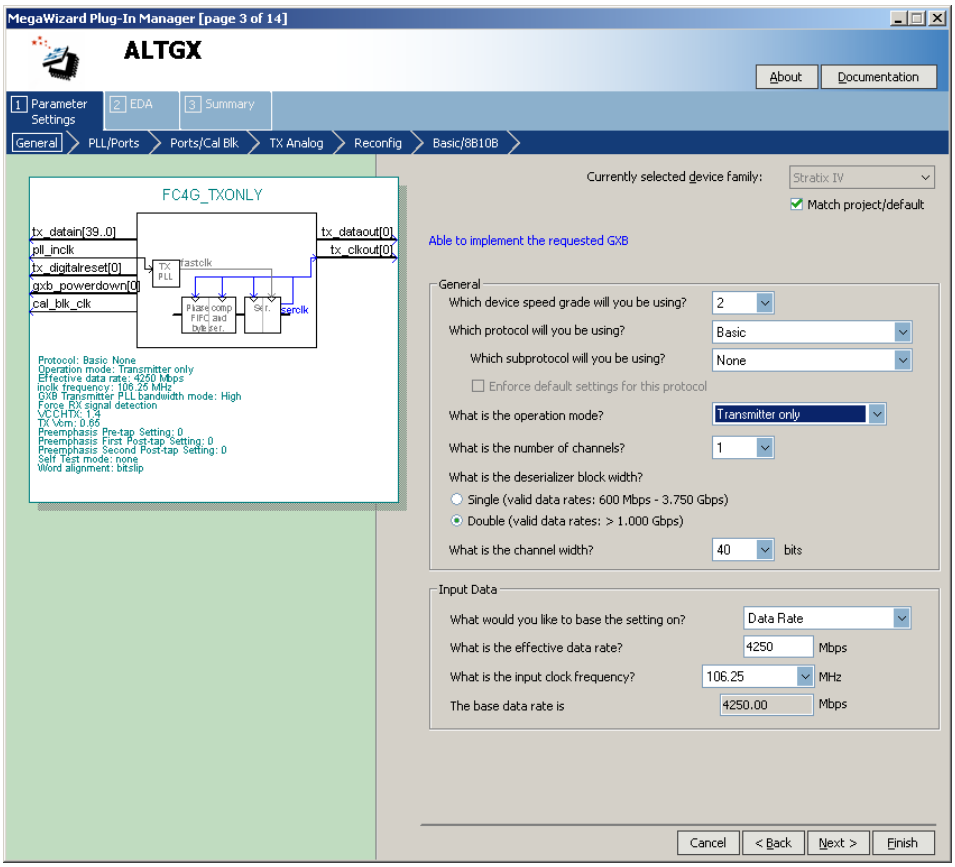
- **Reconfig** screen—Set the starting channel number to **4**.

Create the Instance for an FC4G Configuration—Transmitter Only Mode (Channel 2)

This configuration is similar to the channel 0 configuration, with the following changes:

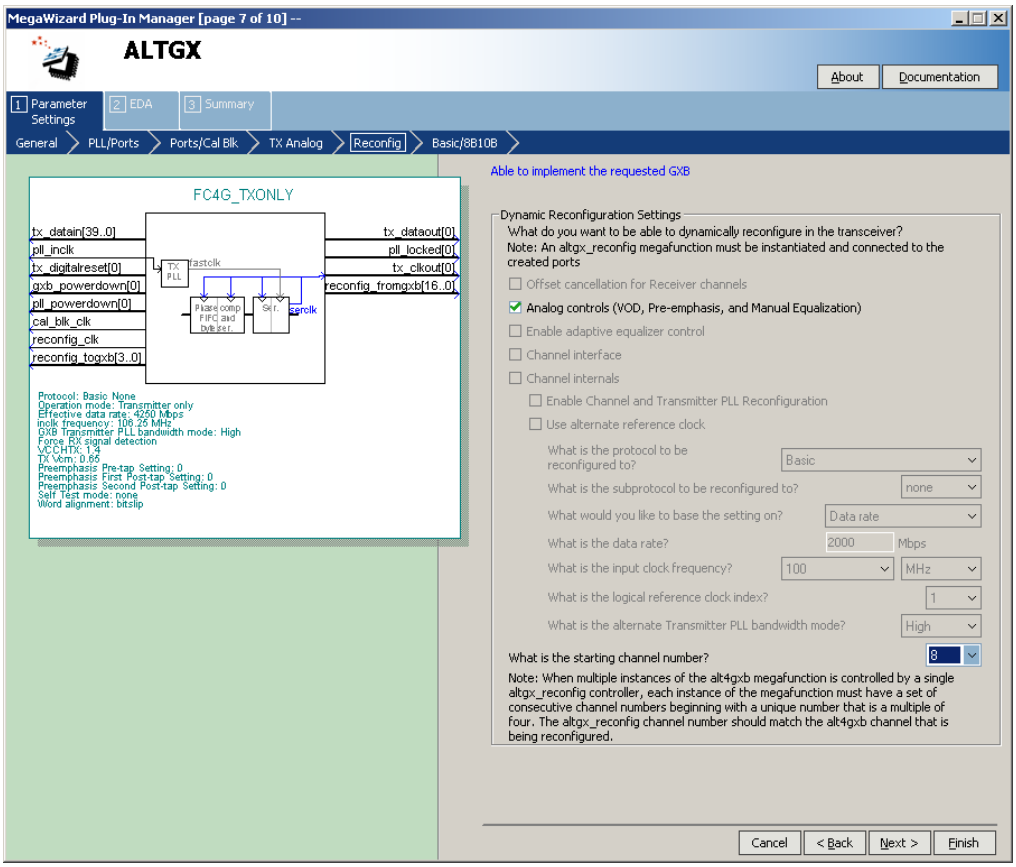
- Set the operation mode to **Transmitter Only**, as shown in Figure 2–14. Because this is a **Transmitter Only** instance, all the options relevant to the receiver are not available in the ALTGX MegaWizard Plug-In Manager.

Figure 2–14. FC4G_TXONLY Instance (Channel 1) Settings (General Screen)



- **Reconfig** screen—Set the starting channel number to **8**. Select the **Analog controls** option even if you do not intend to dynamically reconfigure the PMA controls, as shown in Figure 2–15. Selecting this option is required for this example scenario because:
  - For a **Transmitter Only** instance, offset cancellation is not available; therefore, the `reconfig_fromgxb` and `reconfig_togxb` ports are not available.
  - The other two instances (containing a receiver channel) have these ports available because offset cancellation is automatically enabled.
  - If one transceiver instance has the `reconfig_fromgxb` and `reconfig_togxb` ports enabled, the Quartus II software requires the other transceiver instances to have these ports enabled to combine them in the same transceiver block. Therefore, for this **Transmitter Only** instance, the **Analog options...** must be selected.

Figure 2–15. FC4G_TXONLY Instance (Reconfig) Screen

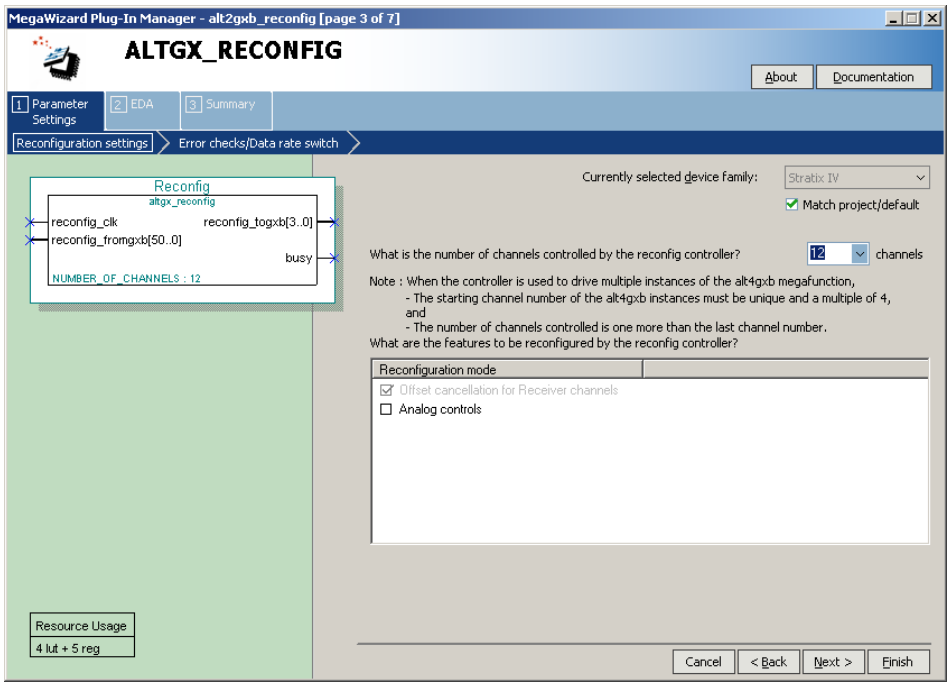


For more information about the requirements to combine multiple transceiver instances, refer to the "Combining Transceiver Instances in Multiple Transceiver Blocks" section in the *Configuring Multiple Protocols and Data Rates in a Transceiver Block* chapter.

Create the Dynamic Reconfiguration Controller (ALTGX_Reconfig) Instance

This section only describes the relevant options that must be set to implement the application.

Figure 2–16. ALTGX_Reconfig Settings (Reconfiguration Settings Screen)



For more information, refer to the *Stratix IV Dynamic Reconfiguration* chapter.

Figure 2–16 shows the options that you must set (assuming that you do not require dynamic reconfiguration of the PMA controls in the transceiver channels).

For more information about selecting the **Number of Channels** option, refer to the "Total Number of Channels Option in the ALTGX_RECONFIG Instance" section in the *Stratix IV Dynamic Reconfiguration* chapter.

Connect the following:

- `reconfig_fromgxb[16:0]` of the ALTGX_RECONFIG instance to the FC4G instance (channel0)
- `reconfig_fromgxb[33:17]` to the FC1G instance (channel1)
- `reconfig_fromgxb[50:34]` to the FC4G **Transmitter Only** instance (channel2)
- `reconfig_togxb[3:0]` of the ALTGX_RECONFIG instance to all three transceiver instances

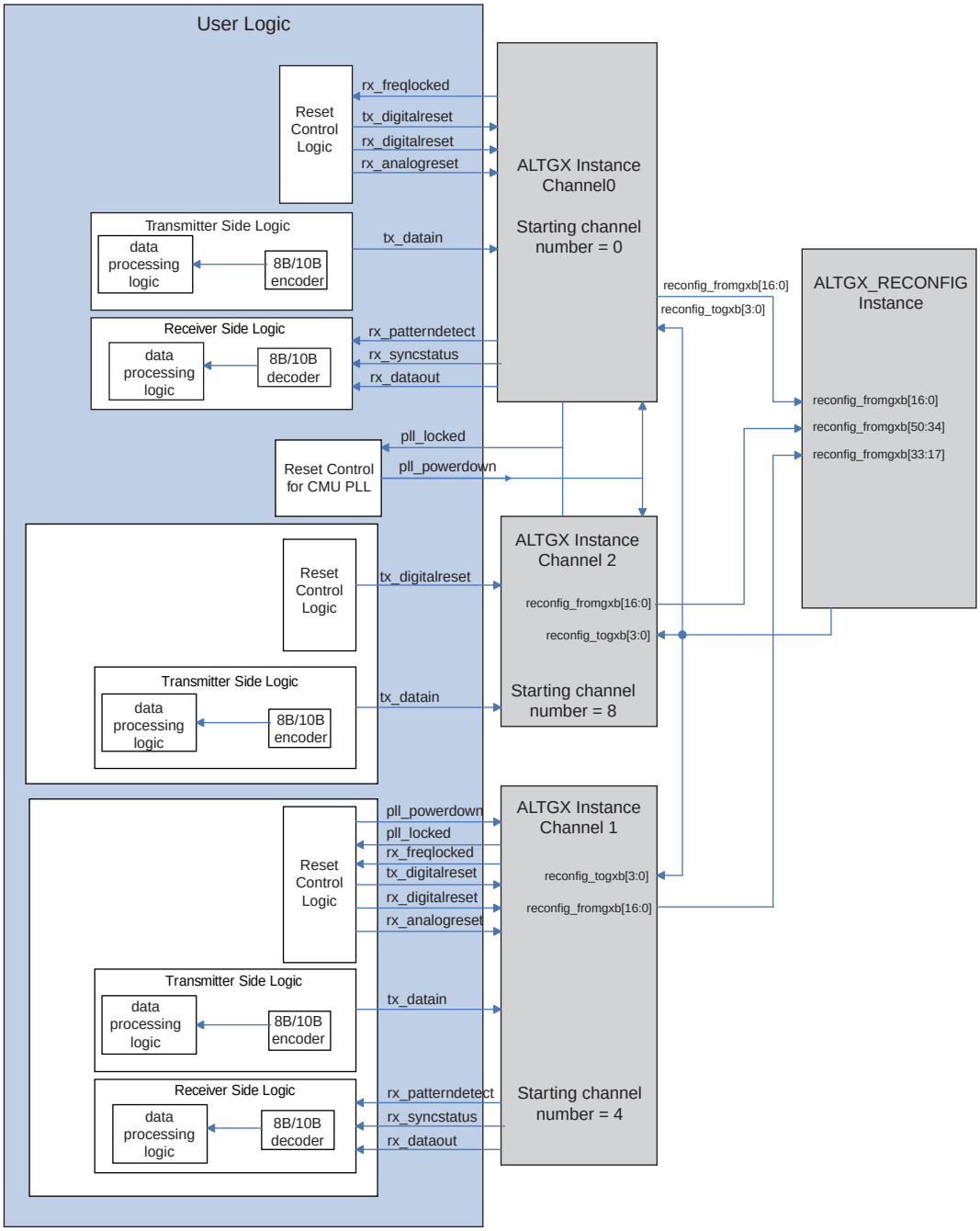
Create Reset Logic to Control the FPGA Fabric and Transceivers

The design requires independent control on each channel. Altera recommends creating independent reset control logic for each channel.

In this design, channel 0 and channel 2 share the same CMU PLL (because they are configured at the same data rate) and channel 1 uses the second CMU PLL. When you create a **Transmitter Only** or **Receiver and Transmitter** instance, the ALTGX MegaWizard Plug-In Manager provides a `pll_powerdown` signal to reset the CMU PLL that provides clocks to the transmitter channel. In this design example, because channels 0 and 2 share the same CMU PLL, drive the `pll_powerdown` port of channel 0 and channel 2 in the ALTGX instance from the same logic.

Channels 0, 1, and 2 have separate `rx_digitalreset`, `rx_analogreset`, and `tx_digitalreset` signals. Figure 2–17 shows the interface between the three transceiver instances and the FPGA fabric.

Figure 2–17. Transceiver—FPGA Fabric Interface



Create Data Processing and Other User Logic

For this example, you must implement the 8B/10B encoder and decoder in the FPGA fabric. Figure 2–17 on page 2–35 shows the logic on the transmitter and receiver side and the system logic controls for all channels in the FPGA fabric. This block diagram is a representation of a typical system and may not exactly show the different blocks in a practical application. Interface all the logic blocks with the transceiver.

If you would like to add SignalTap for verification, first complete synthesis, then add the transceiver-FPGA fabric or other user logic signals in SignalTap. Lastly, compile the design to generate the **.sof**.

Phase 3—Compilation

Assign pins for the input and output signals in your design. The Quartus II software versions 8.1 and earlier do not allow pin assignments for the Stratix IV GX device.

Set the OCT values for the transceiver serial pins, add timing constraints for the clocks and data paths in your logic, then compile the design.

Phase 4—Simulating the Design

To simulate the design, follow the steps outlined in "Functional Simulation" on page 2–12.

Document Revision History

Table 2–7 shows the revision history for this chapter.

Table 2–7. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
November 2009, v4.0	■ Added Table 2–3, Table 2–4, Table 2–5, and Table 2–6. ■ Minor text edits.	—
June 2009, v3.1	■ Updated the "Introduction", "Power Supplies", "Transceiver Configuration", "Clocking", "Create Transceiver Instances", "Create Dynamic Reconfiguration Controller Instances", "Create Data Processing and Other User Logic", "Functional Simulation" sections. ■ Added the "Board Design Requirements", "Gear Boxing Logic", "Guidelines to Debug the FPGA Logic and the Transceiver Interface", and "Guidelines to Debug System Level Issues" sections. ■ Added introductory sentences to improve search ability.	—
March 2009, v3.0	■ Add "Power Supplies" on page 2–6 ■ Updated "Dynamic Reconfiguration" on page 2–4 ■ Text edits	—

Table 2–7. Document Revision History

November 2008, v2.0	■ Added “Transceiver Configuration” on page 2–3 ■ Added “Create Dynamic Reconfiguration Controller Instances” on page 2–8 ■ “Dynamic Reconfiguration” on page 2–15 ■ Updated “Create the Instance for an FC4G Configuration—Transmitter Only Mode (Channel 2)” on page 2–28 ■ Added “Create the Dynamic Reconfiguration Controller (ALTGX_Reconfig) Instance” on page 2–30 ■ Updated Figure 2–1, Figure 2–4, Figure 2–5, Figure 2–6, Figure 2–7, Figure 2–8, Figure 2–10, Figure 2–11, Figure 2–12, Figure 2–13, and Figure 2–14 ■ Added Figure 2–9, Figure 2–15, and Figure 2–16	—
May 2008, v1.0	Initial release	—

3. Stratix IV ALTGX_RECONFIG Megafunction User Guide

SIV53004-3.0

You can use the ALTGX_RECONFIG MegaWizard™ Plug-In Manager in the Quartus® II software to create and modify design files for the Stratix® IV device family. This chapter describes the different Quartus II settings for dynamic reconfiguration in the ALTGX_RECONFIG MegaWizard Plug-In Manager.

The MegaWizard Plug-In Manager helps you create or modify design files that contain custom megafunction variations. These auto-generated MegaWizard files can then be instantiated in a design file. The MegaWizard Plug-In Manager allows you to specify options for the ALTGX_RECONFIG megafunction.

Start the MegaWizard Plug-In Manager using one of the following methods:

- Choose the **MegaWizard Plug-In Manager** command (Tools menu).
- When working in the **Block Editor** (schematic symbol), open the Edit menu and choose **Insert Symbol**. The **Symbol** dialog box appears. In the **Symbol** dialog box, click **MegaWizard Plug-In Manager**.
- Start the stand-alone version of the MegaWizard Plug-In Manager by typing the following command at the command prompt: `qmegawiz`.

Dynamic Reconfiguration

This section describes the options available on the individual pages of the ALTGX_RECONFIG MegaWizard Plug-In Manager.

The MegaWizard Plug-In Manager provides a warning if any of the settings you choose are illegal.

Figure 3–1 shows the first page of the MegaWizard Plug-In Manager. To generate an ALTGX_RECONFIG custom megafunction variation, select **Create a new custom megafunction variation**. Click **Next**.

Figure 3–1. MegaWizard Plug-In Manager (Page 1)

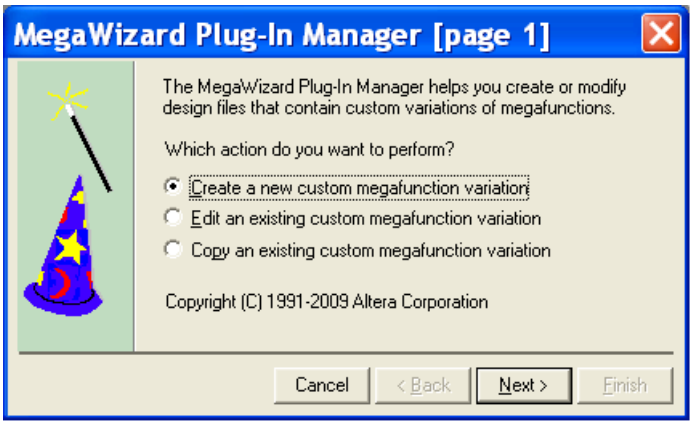


Figure 3–2 shows the second page of the MegaWizard Plug-In Manager. Select the following options (click **Next** when you are done):

1. In the list of megafunctions on the left, click the "+" icon beside the I/O item. From the options presented, choose **ALTGX_RECONFIG megafunction**.
2. From the drop-down menu beside **Which device family will you be using?**, select **Stratix IV**.
3. From the radio buttons under **Which type of output file do you want to create?**, choose your output file format (**AHDL**, **VHDL**, or **Verilog HDL**).
4. In the box beneath **What name do you want for the output file?**, enter the file name or click the **Browse** button to search for it.

For the design to compile successfully, always enable the dynamic reconfiguration controller for all the ALTGX instances in the design.

Figure 3–2. MegaWizard Plug-In Manager—ALTGX_RECONFIG (Page 2)

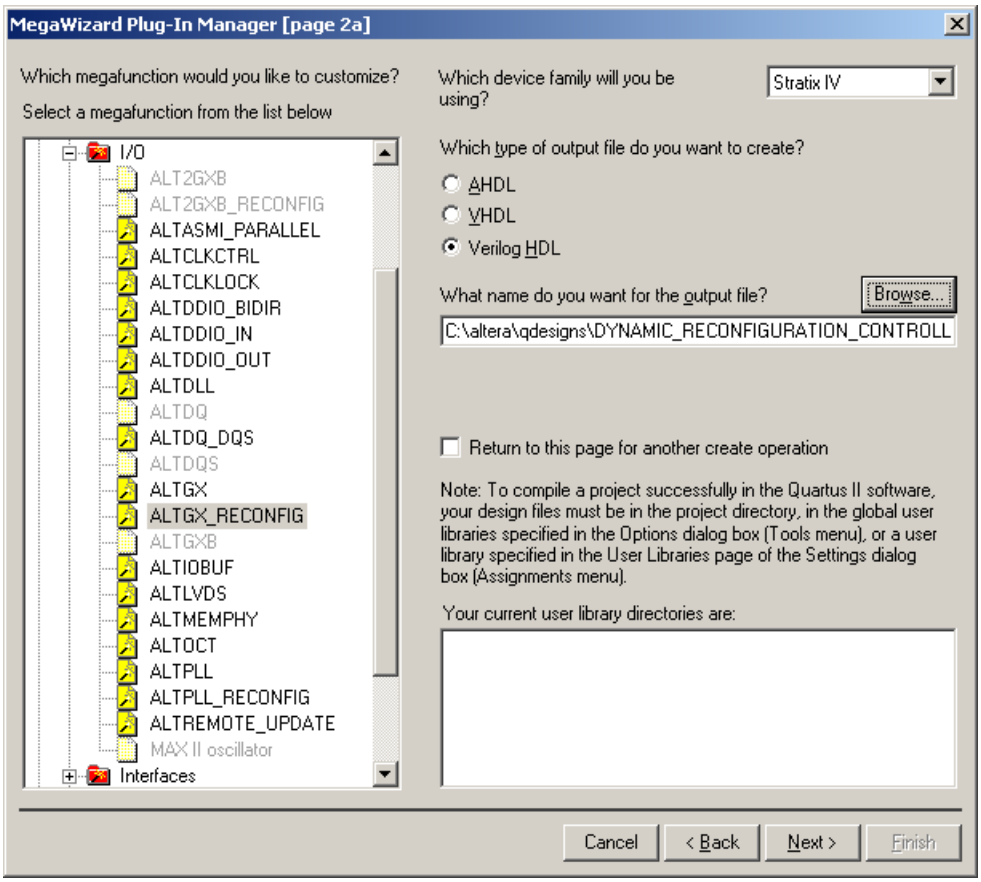


Figure 3–3 shows page 3 of the ALTGX_RECONFIG MegaWizard Plug-In Manager. From the drop-down menu, select the number of channels controlled by the dynamic reconfiguration controller.

Figure 3–3. MegaWizard Plug-In Manager—ALTGX_RECONFIG (Reconfiguration Settings)

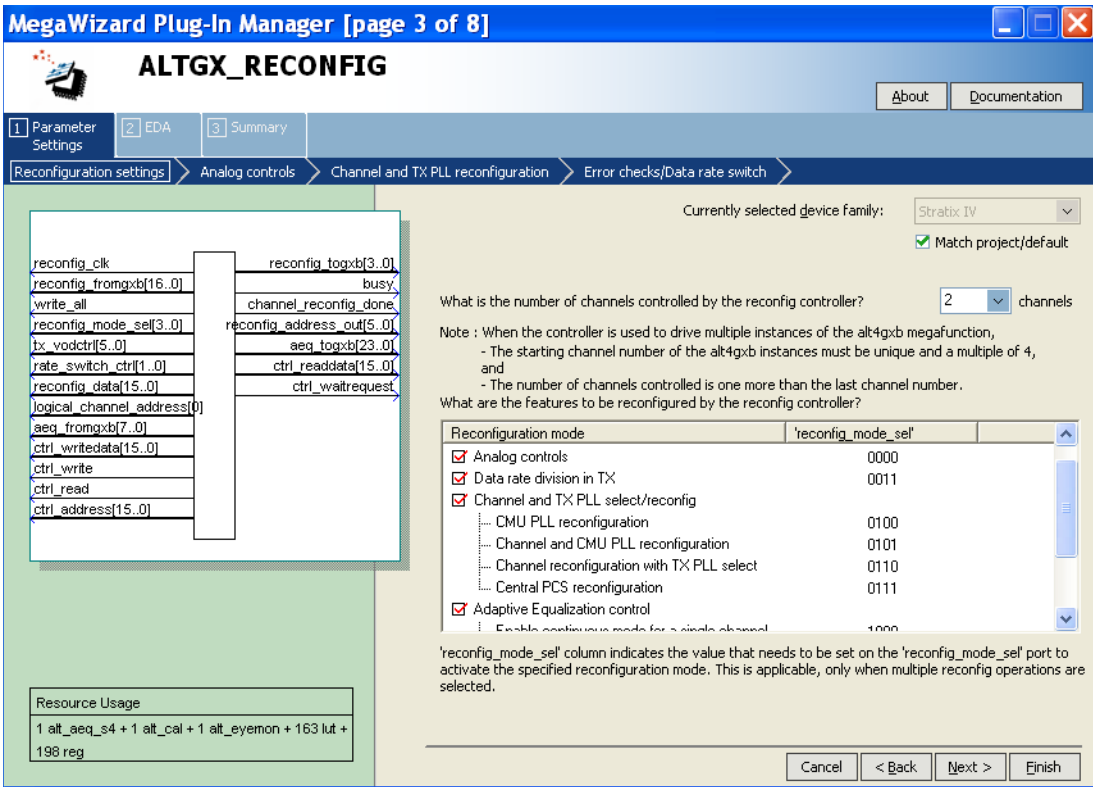


Table 3–1 describes the available options on page 3 of the MegaWizard Plug-In Manager for your ALTGX_RECONFIG custom megafunction variation. Select the **Match project/default** option if you want to change the device **Currently selected device family** options.

Make your selections on page 3, then click **Next**.

Table 3–1. MegaWizard Plug-In Manager Options (Page 3) (Part 1 of 2)

ALTGX_RECONFIG Setting	Description	Reference
What is the number of channels controlled by the reconfig controller?	Determine the highest logical channel address among all the ALTGX instances connected to the ALTGX_RECONFIG instance. Round it up to the next multiple of four and set that number in this option. Depending on this setting, the ALTGX_RECONFIG MegaWizard Plug-in Manager generates the appropriate signal width for the interface signal (`reconfig_fromgxb`) between the ALTGX_RECONFIG and the ALTGX instances. It also gives the necessary bus width for all the selected physical media attachment (PMA) signals. Depending on the number of channels set, the resource estimate changes because this is a soft implementation that uses fabric logic resources. The resource estimate is shown in the bottom left of Page 3 of the MegaWizard Plug-in Manager.	"Total Number of Channels Controlled by the ALTGX_RECONFIG Instance" section of the *Stratix IV Dynamic Reconfiguration* chapter.

Table 3–1. MegaWizard Plug-In Manager Options (Page 3) (Part 2 of 2)

ALTGX_RECONFIG Setting	Description	Reference
What are the features to be reconfigured by the reconfig controller?	This feature is always enabled by default: ■ **Offset Cancellation for Receiver Channels**—After the device powers up, the dynamic reconfiguration controller performs offset cancellation on the receiver portion of all the transceiver channels controlled by it.	"Offset Cancellation" section of the *Stratix IV Dynamic Reconfiguration* chapter.
	These features are available for selection: ■ **Analog Controls**—Allows dynamic reconfiguration of PMA controls such as Equalization, Pre-emphasis, DC Gain, and voltage offset differential (VOD). ■ **Data rate division in TX**—Allows dynamic reconfiguration of the transmitter local divider settings to 1, 2, or 4. The transmitter channel data rate is reconfigured based on the local divider settings. ■ **Channel and TX PLL select/reconfig**—The following features are available under this option: → **CMU PLL Reconfiguration**—Allows you to dynamically reconfigure the clock multiplier unit (CMU) phase-locked loop (PLL) to a different data rate. → **Channel and CMU PLL reconfiguration**—Allows the dynamic reconfiguration of the transceiver channel from one functional mode to another and also the CMU PLL reconfiguration. → **Channel reconfiguration with TX PLL select**—Allows you to select additional transmitter PLLs for the transceiver channel and reconfigure the functional mode of the channel. → **Central Control Unit reconfiguration**—Allows you to reconfigure bonded mode configurations from one to another. ■ **Adaptive Equalization Control** —Allows you to reconfigure the adaptive equalization hardware (AEQ) in the receiver portion of the transceivers. There are three modes available: → **Enable continuous mode for a single channel**—The equalization settings for the specified channel are continuously optimized by the AEQ hardware. → **Enable one time mode for a single channel**—A single stable equalization value is set up and locked for the specified channel by the AEQ hardware. → **Powerdown for a single channel**—The AEQ hardware of the specified channel is put into standby mode by the AEQ control logic (soft-IP) in the dynamic reconfiguration controller.	"PMA Controls Reconfiguration Mode Details" section of the *Stratix IV Dynamic Reconfiguration* chapter. "Data Rate Division in Transmitter Mode Details" section of the *Stratix IV Dynamic Reconfiguration* chapter. "CMU PLL Reconfiguration Mode Details'" section of the *Stratix IV Dynamic Reconfiguration* chapter. "Channel and CMU PLL Reconfiguration Mode Details" section of the *Stratix IV Dynamic Reconfiguration* chapter. "Channel reconfiguration with TX PLL Select Mode Details" section of the *Stratix IV Dynamic Reconfiguration* chapter. "Adaptive Equalization (AEQ)" section in the *Stratix IV Dynamic Reconfiguration* chapter.
What are the features to be reconfigured by the reconfig controller?	■ **EyeQ control**—Allows you to reconfigure the EyeQ hardware in the receiver portion of the transceivers.	"EyeQ" section in the *Stratix IV Dynamic Reconfiguration* chapter.

Figure 3–4 shows page 4 of the ALTGX_RECONFIG MegaWizard Plug-In Manager.

Figure 3–4. MegaWizard Plug-In Manager—ALTGX_RECONFIG (Analog Controls)

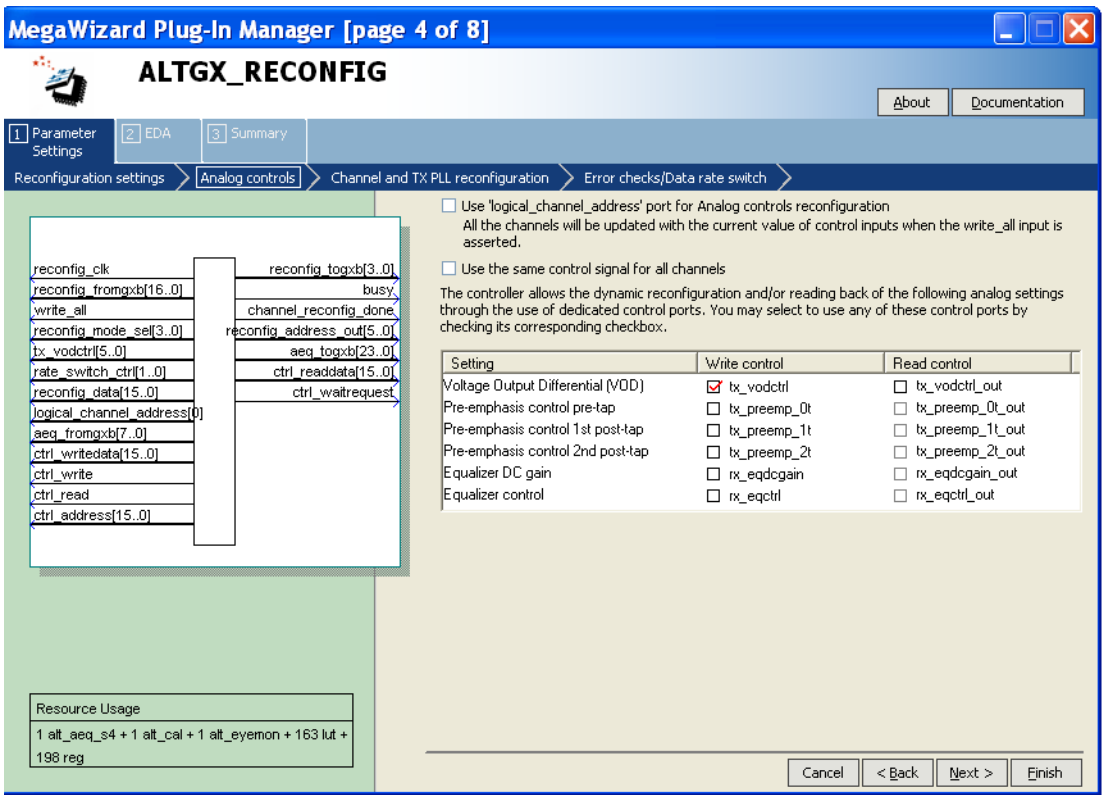


Table 3–2 describes the available options on page 4 of the MegaWizard Plug-In Manager for your ALTGX_RECONFIG custom megafunction variation.

Make your selections on page 4, then click **Next**.

Table 3–2. MegaWizard Plug-In Manager Options (Page 4) (Part 1 of 2)

ALTGX_RECONFIG Setting	Description	Reference
Use '`logical_channel_address`' port for Analog controls reconfiguration	This option is applicable only for Analog controls reconfiguration and is available for selection when the number of channels controlled by the ALTGX_RECONFIG instance is more than one. The dynamic reconfiguration controller reconfigures only the channel whose logical channel address is specified at the `logical_channel_address` port. The width of this port is selected by the ALTGX_RECONFIG MegaWizard Plug-In Manager depending on the number of channels controlled by the dynamic reconfiguration controller. The maximum width of the `logical_channel_address` port is 9 bits.	"Dynamic Reconfiguration Controller Port List" and "Method 1—Using the logical_channel_address Port" sections of the *Stratix IV Dynamic Reconfiguration* chapter.
Use the same control signal for all channels	This option is available for selection when the number of channels controlled by the ALTGX_RECONFIG instance is more than one. When you enable this option, the dynamic reconfiguration controller writes the same control signals to all the channels connected to it. You cannot select this option if you enable the **Use 'logical_channel_address' port for Analog controls reconfiguration** option.	Method 2 and Method 3 of the "PMA Controls Reconfiguration Mode Details" section of the *Stratix IV Dynamic Reconfiguration* chapter.

Table 3–2. MegaWizard Plug-In Manager Options (Page 4) (Part 2 of 2)

ALTGX_RECONFIG Setting	Description	Reference
Write Control	The PMA control ports available to write various analog settings to the transceiver channels controlled by the dynamic reconfiguration controller are as follows: ■ `tx_vodctrl`—VOD; 3 bits per channel ■ `tx_preemp_0t`—Pre-emphasis control pre-tap; 5 bits per channel ■ `tx_preemp_1t`—Pre-emphasis control 1st post-tap; 5 bits per channel ■ `tx_preemp_2t`—Pre-emphasis control 2nd post-tap; 5 bits per channel ■ `rx_eqdcgain`—Equalizer DC gain; 3 bits per channel ■ `rx_eqctrl`—Equalizer control; 4 bits per channel These are optional signals. The signal widths are based on the setting you entered for the **What is the number of channels controlled by the reconfig controller?** option and whether you enabled the **Use 'logical_channel_address' port for Analog controls reconfiguration** option. The port width is also determined by the **Use the same control signal for all channels** option. At least one of these PMA control ports must be enabled to configure and use the dynamic reconfiguration controller.	"Dynamically Reconfiguring PMA Controls" section of the *Stratix IV Dynamic Reconfiguration* chapter.
Read Control	The PMA control ports available to read the existing values from the transceiver channels controlled by the dynamic reconfiguration controller are as follows: ■ `tx_vodctrl_out`—VOD; 3 bits per channel ■ `tx_preemp_0t_out`—Pre-emphasis control pre-tap; 5 bits per channel ■ `tx_preemp1t_out`—Pre-emphasis control 1st post-tap; 5 bits per channel ■ `tx_preemp_2t_out`—Pre-emphasis control 2nd post-tap; 5 bits per channel ■ `rx_eqdcgain_out`—Equalizer DC gain; 3 bits per channel ■ `rx_eqctrl_out`—Equalizer control; 4 bits per channel These are optional signals. The signal widths are based on the setting you entered for the **What is the number of channels controlled by the reconfig controller?** option and whether you enabled the **Use 'logical_channel_address' port for Analog controls reconfiguration** option. The PMA controls are available for selection only if you select the corresponding write control. Read and write transactions cannot be performed simultaneously.	

Figure 3–5 shows page 5 of the ALTGX_RECONFIG MegaWizard Plug-In Manager.

Figure 3–5. MegaWizard Plug-In Manager—ALTGX_RECONFIG (Channel and TX/PLL Reconfiguration)

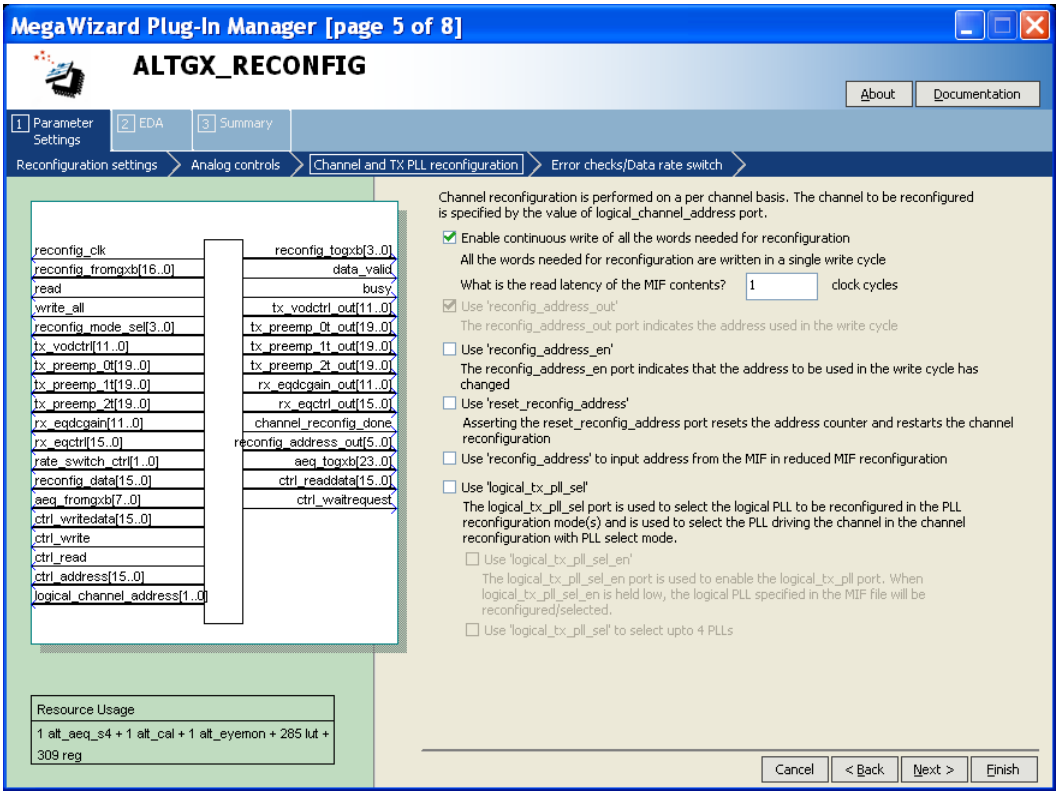


Table 3–4 describes the available options on page 5 of the MegaWizard Plug-In Manager for your ALTGX_RECONFIG custom megafunction variation.

Table 3–3. MegaWizard Plug-In Manager Options (Page 5) (Part 1 of 2)

ALTGX_RECONFIG Setting	Description	Reference
Enable continuous write of all the words needed for reconfiguration.	For a continuous write operation, select the **Enable continuous write of all the words needed for reconfiguration** option to pulse the `write_all signal` once to write an entire memory initialization file (**.mif**).	"Dynamic Reconfiguration Controller Port List" section in the *Stratix IV Dynamic Reconfiguration* chapter.
What is the read latency of the MIF contents?	This option is available only if you have selected the **Enable continuous write of all the words needed for reconfiguration** option. Enter the desired latency in terms of the `reconfig_clk` cycles it takes for each **.mif** word to be present at the `reconfig_data` port. For more information, refer to Figure 3–6.	"Dynamic Reconfiguration Controller Port List" section in the *Stratix IV Dynamic Reconfiguration* chapter.

Table 3–3. MegaWizard Plug-In Manager Options (Page 5) (Part 2 of 2)

ALTGX_RECONFIG Setting	Description	Reference
Use 'reconfig_address_out'	This option is enabled by default when you select the **Channel and TX PLL select/reconfig** option. The value on `reconfig_address_out[5:0]` indicates the address associated with the words in the **.mif**, which contains the dynamic reconfiguration instructions. The dynamic reconfiguration controller automatically increments the address at the end of each **.mif** write transaction.	"Dynamic Reconfiguration Controller Port List" section in the *Stratix IV Dynamic Reconfiguration* chapter.
Use 'reconfig_address_en'	When high, this optional output status signal indicates that the address used in the **.mif** write transaction cycle has changed. This signal is asserted when the **.mif** write transaction is completed (when the `busy` signal is de-asserted).	"Dynamic Reconfiguration Controller Port List" section in the *Stratix IV Dynamic Reconfiguration* chapter.
Use 'reset_reconfig_address'	When asserted, this optional control signal resets `reconfig_address_out` (the current reconfiguration address) to **0**.	"Dynamic Reconfiguration Controller Port List" section in the *Stratix IV Dynamic Reconfiguration* chapter.
Use 'logical_tx_pll_sel'	This is an optional control signal. The `logical_tx_pll_sel[1:0]` signal refers to the logical reference index of the CMU PLL. The functionality of the signal depends on the feature activated, as shown below: ■ **CMU PLL reconfiguration**—The corresponding CMU PLL is reconfigured based on the value at `logical_tx_pll_sel[1:0]`. ■ **Channel and CMU PLL reconfiguration**—The corresponding CMU PLL is reconfigured based on the value at this signal. The transceiver channel listens to the CMU PLL selected by `logical_tx_pll_sel[1:0]`. ■ **Channel reconfiguration with TX PLL select**—The transceiver channel listens to the TX PLL selected by `logical_tx_pll_sel[1:0]`.	"The logical_tx_pll_sel and logical_tx_pll_sel_en Ports" section in the *Stratix IV Dynamic Reconfiguration* chapter.
Use 'logical_tx_pll_sel_en'	This is an optional control signal. When you enable this signal, the value set on the `logical_tx_pll_sel[1:0]` signal is valid only if the `logical_tx_pll_sel_en` is set to **1**.	"The logical_tx_pll_sel and logical_tx_pll_sel_en Ports" section in the *Stratix IV Dynamic Reconfiguration* chapter.

Figure 3–6 shows that the read latency of the **.mif** contents is 2, as it takes two `reconfig_clk` cycles for the **.mif** data to become available on the `reconfig_data` port after providing address on the `reconfig_address_out` port.

Figure 3–6. Read Latency

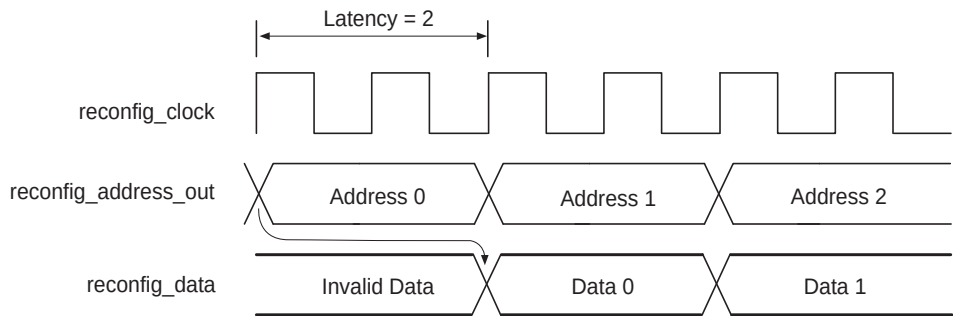


Figure 3–7 shows page 6 of the ALTGX_RECONFIG MegaWizard Plug-In Manager.

Figure 3–7. MegaWizard Plug-In Manager—ALTGX_RECONFIG (Error Checks/Data Rate Switch)

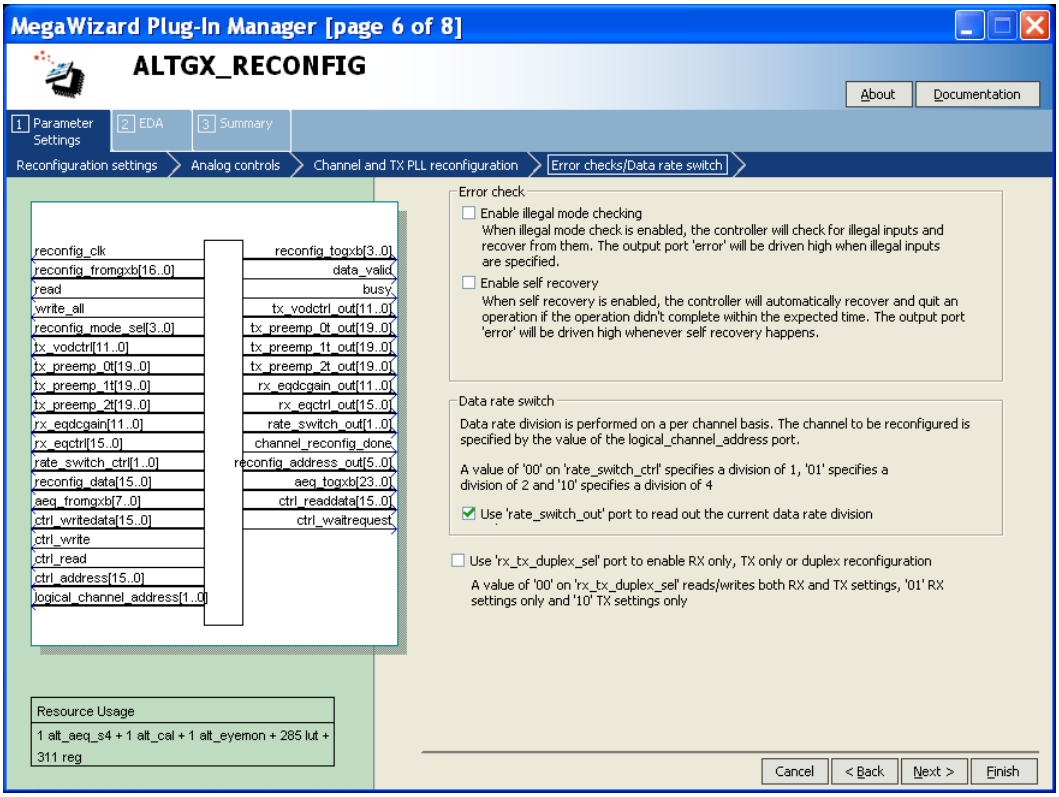


Table 3–4 describes the available options on page 6 of the MegaWizard Plug-In Manager for your ALTGX_RECONFIG custom megafunction variation.

Make your selections on page 6, then click **Next**.

Table 3–4. MegaWizard Plug-In Manager Options (Page 6)

ALTGX_RECONFIG Setting	Description	Reference
Enable illegal mode checking	When you select this option, the ALTGX_RECONFIG MegaWizard Plug-In Manager provides the `error` output port. The dynamic reconfiguration controller detects the error conditions within two `reconfig_clk cycles`, de-asserts the `busy` signal, and asserts the error signal for two `reconfig_clk cycles`.	"Error Indication in the ALTGX_RECONFIG MegaWizard Plug-In Manager" section of the *Stratix IV Dynamic Reconfiguration* chapter.
Enable self recovery	When you select this option, the controller automatically recovers if the operation did not complete within the expected time. The error signal is driven high whenever the controller performs a self recovery.	"Error Indication in the ALTGX_RECONFIG MegaWizard Plug-In Manager" section of the *Stratix IV Dynamic Reconfiguration* chapter.
Use `rate_switch_out` port to read out the current data rate division	The `rate_switch_out[1:0]` signal is available when you select Data Rate Division in TX mode. You can read the existing local divider settings of a transmitter channel at this port. The decoding for this signal is listed below: 2'b00—Division of 1 2'b01—Division of 2 2'b10—Division of 4 2'b11—Not supported	"Data Rate Division in TX' mode section in the *Stratix IV Dynamic Reconfiguration* chapter.
Use the `rx_tx_duplex_sel` port to enable RX only, TX only or duplex configuration	You can read or write the receiver and transmitter settings, only the receiver settings, or only the transmitter settings, based on the value you set at the `rx_tx_duplex_sel[1:0]` port; ■ 2'b00—Duplex mode ■ 2'b01—RX only mode ■ 2'b10—TX only mode ■ 2'b11—unsupported value (do not use this value) If you disable the `rx_tx_duplex_sel[1:0]` port, the dynamic reconfiguration controller reads or writes both the receiver and transmitter settings.	"Dynamically Reconfiguring PMA Controls" section of the *Stratix IV Dynamic Reconfiguration* chapter.

Figure 3–8 shows page 7 (the Simulation Libraries page) of the MegaWizard Plug-In Manager, which is used for dynamic reconfiguration selection.

Make your selections, then click **Next**.

Figure 3–8. MegaWizard Plug-In Manager—ALTGX_RECONFIG (Simulation Libraries)

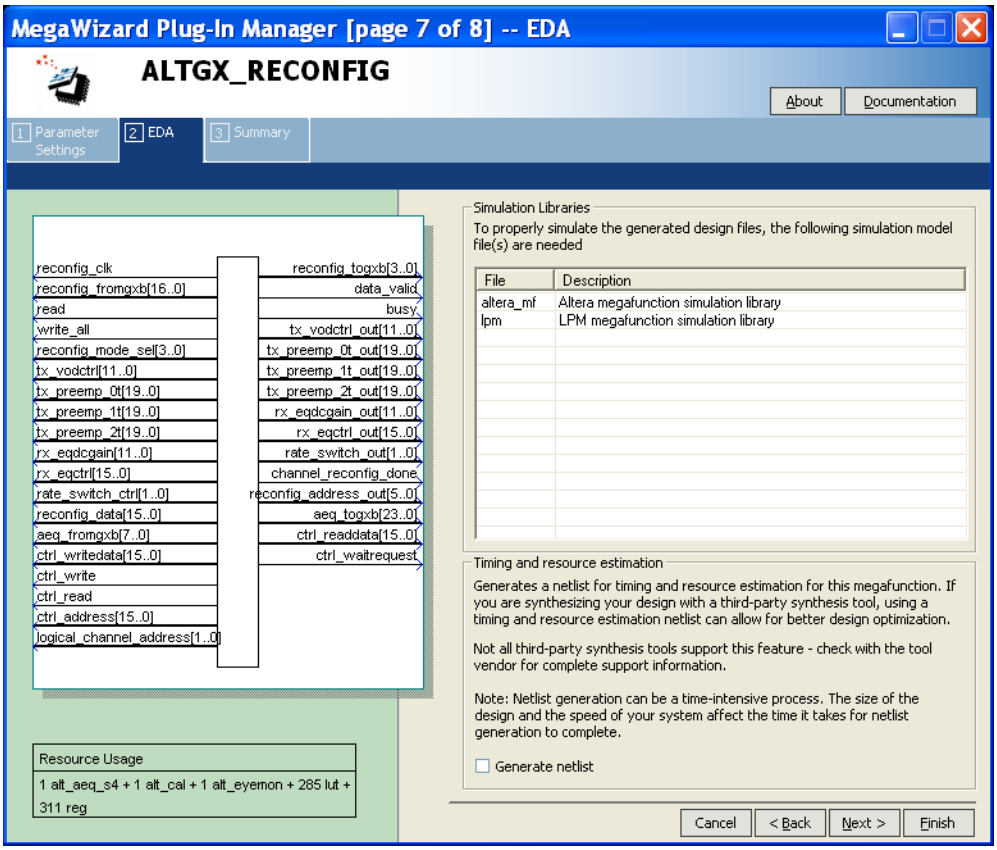


Table 3–5 describes the available option on page 7 of the MegaWizard Plug-In Manager for your ALTGX_RECONFIG custom Megafunction variation.

Make your selections on page 7, then click **Next**.

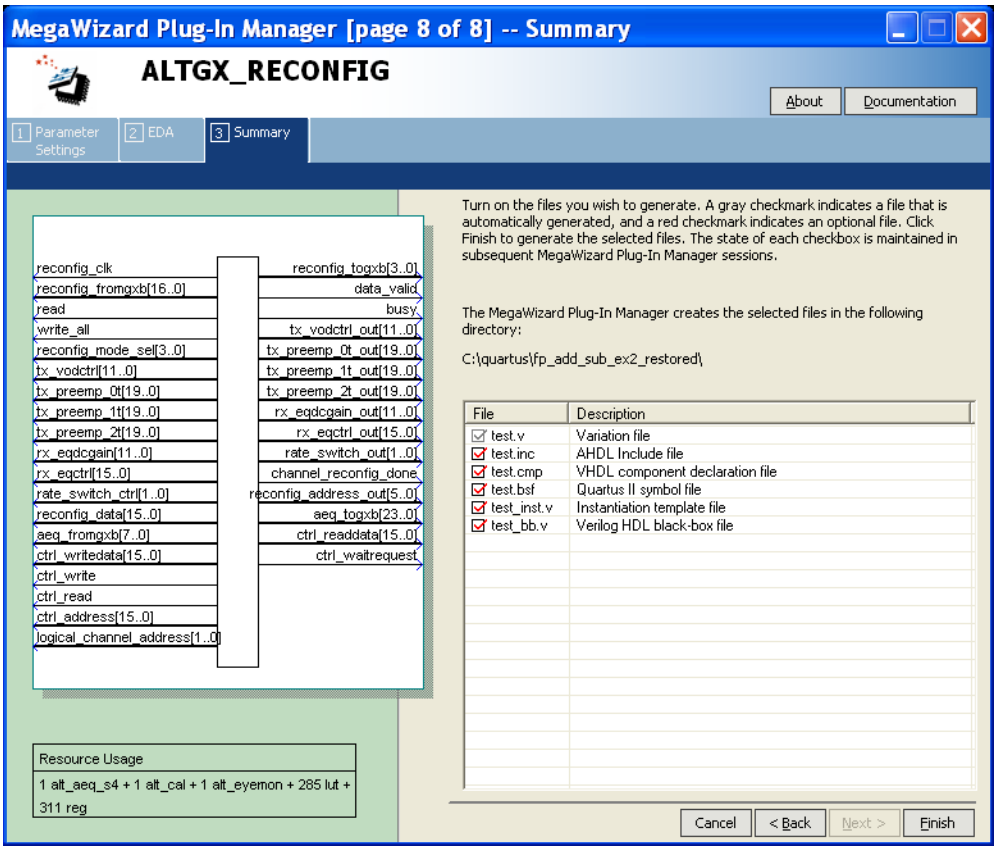
Table 3–5. MegaWizard Plug-In Manager Options (Page 7)

ALTGX_RECONFIG Setting	Description	Reference
Generate a netlist for synthesis area and timing estimation	Selecting this option generates a netlist file that third-party synthesis tools can use to estimate the timing and resource usage.	—

Figure 3–9 shows page 8 (the last page) of the MegaWizard Plug-In Manager for the dynamic reconfiguration protocol set up. You can select optional files on this page.

After you make your selections, click **Finish** to generate the files.

Figure 3–9. MegaWizard Plug-In Manager—ALTGX_RECONFIG (Summary)



Document Revision History

Table 3–6 shows the revision history for this chapter.

Table 3–6. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
November 2009, v3.0	■ Updated Table 3–1. ■ Updated Table 3–3. ■ Added Figure 3–6. ■ Made minor text edits.	—
June 2009, v2.1	■ Updated Table 3–3. ■ Added introductory sentences to improve search ability. ■ Minor text edits.	—
March 2009, v2.0	Updated screen shots.	—
November 2008, v1.0	Added chapter to the Stratix IV Device Handbook	—

Stratix IV Device Handbook

Volume 4

101 Innovation Drive
San Jose, CA 95134
www.altera.com

SIV5V4-4.0

Copyright © 2009 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

I.S. EN ISO 9001

Contents

Chapter Revision Dates v

Additional Information
About this Handbook . . . . . Info-vii
How to Contact Altera . . . . . Info-vii
Typographic Conventions . . . . . Info-vii

Section I. Stratix IV Device Datasheet
Revision History I-1

Chapter 1. DC and Switching Characteristics
Electrical Characteristics 1-1
Operating Conditions 1-1
Absolute Maximum Ratings 1-1
Recommended Operating Conditions 1-3
DC Characteristics 1-6
Internal Weak Pull-Up Resistor 1-9
I/O Standard Specifications 1-10
Power Consumption 1-13
Switching Characteristics 1-13
Transceiver Performance Specifications 1-13
Transceiver Datapath PCS Latency 1-34
Core Performance Specifications 1-35
Clock Tree Specifications 1-35
PLL Specifications 1-35
DSP Block Specifications 1-37
TriMatrix Memory Block Specifications 1-38
Configuration and JTAG Specifications 1-39
Temperature Sensing Diode Specifications 1-40
Periphery Performance 1-40
High-Speed I/O Specification 1-41
OCT Calibration Block Specifications 1-47
Duty Cycle Distortion (DCD) Specifications 1-47
I/O Timing 1-47
Programmable IOE Delay 1-48
Programmable Output Buffer Delay 1-48
Glossary 1-49
Document Revision History 1-53

Chapter Revision Dates

The chapters in this book, *Stratix IV Device Handbook Volume 4*, were revised on the following dates. Where chapters or groups of chapters are available separately, part numbers are listed.

Chapter 1 DC and Switching Characteristics
Revised: *November 2009*
Part Number: *SIV54001-4.0*

About this Handbook

This handbook provides comprehensive information about the Altera® Stratix® IV family of devices.

How to Contact Altera

For the most up-to-date information about Altera products, see the following table.

Contact *(Note 1)*	Contact Method	Address
Technical support	Website	www.altera.com/support
Technical training	Website	www.altera.com/training
	Email	custrain@altera.com
Altera literature services	Email	literature@altera.com
Non-technical support (General)	Email	nacomp@altera.com
(Software Licensing)	Email	authorization@altera.com

Note:

(1) You can also contact your local Altera sales office or sales representative.

Typographic Conventions

The following table shows the typographic conventions that this document uses.

Visual Cue	Meaning
Bold Type with Initial Capital Letters	Indicates command names, dialog box titles, dialog box options, and other GUI labels. For example, **Save As** dialog box. For GUI elements, capitalization matches the GUI.
bold type	Indicates directory names, project names, disk drive names, file names, file name extensions, dialog box options, software utility names, and other GUI labels. For example, **\qdesigns** directory, **d:** drive, and **chiptrip.gdf** file.
Italic Type with Initial Capital Letters	Indicates document titles. For example, *AN 519: Stratix IV Design Guidelines.*
Italic type	Indicates variables. For example, $n + 1$. Variable names are enclosed in angle brackets (< >). For example, *<file name>* and *<project name>***.pof** file.
Initial Capital Letters	Indicates keyboard keys and menu names. For example, Delete key and the Options menu.
“Subheading Title”	Quotation marks indicate references to sections within a document and titles of Quartus II Help topics. For example, “Typographic Conventions.”

Visual Cue	Meaning
`Courier type`	Indicates signal, port, register, bit, block, and primitive names. For example, `data1`, `tdi`, and `input`. Active-low signals are denoted by suffix n. For example, `resetn`. Indicates command line commands and anything that must be typed exactly as it appears. For example, `c:\qdesigns\tutorial\chiptrip.gdf`. Also indicates sections of an actual file, such as a Report File, references to parts of files (for example, the AHDL keyword `SUBDESIGN`), and logic function names (for example, `TRI`).
1., 2., 3., and a., b., c., and so on.	Numbered steps indicate a list of items when the sequence of the items is important, such as the steps listed in a procedure.
■ ■	Bullets indicate a list of items when the sequence of the items is not important.
☞	The hand points to information that requires special attention.
CAUTION	A caution calls attention to a condition or possible situation that can damage or destroy the product or your work.
WARNING	A warning calls attention to a condition or possible situation that can cause you injury.
↵	The angled arrow instructs you to press **Enter**.
👣	The feet direct you to more information about a particular topic.

Section I. Stratix IV Device Datasheet

This section includes the following chapters:

- Chapter 1, DC and Switching Characteristics

Revision History

Refer to each chapter for its own specific revision history. For information on when each chapter was updated, refer to the Chapter Revision Dates section, which appears in the full handbook.

SIV54001-4.0

Electrical Characteristics

This chapter covers the electrical and switching characteristics for Stratix® IV devices. Electrical characteristics include operating conditions and power consumption. Switching characteristics include transceiver specifications, core, and periphery performance. This chapter also describes I/O timing, including programmable I/O element (IOE) delay and programmable output buffer delay.

For information regarding the densities and packages of devices in the Stratix IV family, refer to Table 1-1 and Table 1-2 of the *Stratix IV Device Family Overview* chapter.

Operating Conditions

When you use Stratix IV devices, they are rated according to a set of defined parameters. To maintain the highest possible performance and reliability of the Stratix IV devices, you must consider the operating requirements described in this chapter.

Stratix IV devices are offered in both commercial and industrial grades. Commercial devices are offered in –2 (fastest), –2×, –3, and –4 speed grades. Industrial devices are offered in –1, –2, –3, and –4 speed grades.

Absolute Maximum Ratings

Absolute maximum ratings define the maximum operating conditions for Stratix IV devices. The values are based on experiments conducted with the devices and theoretical modeling of breakdown and damage mechanisms. The functional operation of the device is not implied for these conditions.

CAUTION

Conditions other than those listed in Table 1–1, Table 1–2, and Table 1–3 may cause permanent damage to the device. Additionally, device operation at the absolute maximum ratings for extended periods of time may have adverse effects on the device.

Table 1–1. Absolute Maximum Ratings for Stratix IV Devices (Part 1 of 2)

Symbol	Description	Minimum	Maximum	Unit
V_{CC}	Core voltage and periphery circuitry power supply	-0.5	1.35	V
V_{CCPT}	Power supply for programmable power technology	-0.5	1.8	V
V_{CCPGM}	Configuration pins power supply	-0.5	3.75	V
V_{CCAUX}	Auxiliary supply for the programmable power technology	-0.5	3.75	V
V_{CCBAT}	Battery back-up power supply for design security volatile key register	-0.5	3.75	V
V_{CCPD}	I/O pre-driver power supply	-0.5	3.75	V
V_{CCIO}	I/O power supply	-0.5	3.9	V
V_{CC_CLKIN}	Differential clock input power supply	-0.5	3.75	V
V_{CCD_PLL}	PLL digital power supply	-0.5	1.35	V

Table 1–1. Absolute Maximum Ratings for Stratix IV Devices (Part 2 of 2)

Symbol	Description	Minimum	Maximum	Unit
V_{CCA_PLL}	PLL analog power supply	-0.5	3.75	V
V_I	DC input voltage	-0.5	4.0	V
I_{OUT}	DC output current per pin	-25	40	mA
T_J	Operating junction temperature	-55	125	°C
T_{STG}	Storage temperature (No bias)	-65	150	°C

Table 1–2. Transceiver Power Supply Absolute Maximum Ratings for Stratix IV GX Devices

Symbol	Description	Minimum	Maximum	Unit
V_{CCA_L}	Transceiver high voltage power (left side)	-0.5	3.75	V
V_{CCA_R}	Transceiver high voltage power (right side)	-0.5	3.75	V
V_{CCHIP_L}	Transceiver HIP digital power (left side)	-0.5	1.35	V
V_{CCHIP_R}	Transceiver HIP digital power (right side)	-0.5	1.35	V
V_{CCR_L}	Receiver power (left side)	-0.5	1.35	V
V_{CCR_R}	Receiver power (right side)	-0.5	1.35	V
V_{CCT_L}	Transmitter power (left side)	-0.5	1.35	V
V_{CCT_R}	Transmitter power (right side)	-0.5	1.35	V
V_{CCL_GXBLn} *(1)*	Transceiver clock power (left side)	-0.5	1.35	V
V_{CCL_GXBRn} *(1)*	Transceiver clock power (right side)	-0.5	1.35	V
V_{CCH_GXBLn} *(1)*	Transmitter output buffer power (left side)	-0.5	1.8	V
V_{CCH_GXBRn} *(1)*	Transmitter output buffer power (right side)	-0.5	1.8	V

Note to Table 1–2:

(1) n = 0, 1, 2, or 3.

Table 1–3. Transceiver Power Supply Absolute Maximum Ratings for Stratix IV GT Devices *(Note 1)* (Part 1 of 2)

Symbol	Description	Minimum	Maximum	Unit
V_{CCA_L}	Transceiver high voltage power (left side)	-0.5	3.75	V
V_{CCA_R}	Transceiver high voltage power (right side)	-0.5	3.75	V
V_{CCHIP_L}	Transceiver HIP digital power (left side)	-0.5	1.35	V
V_{CCHIP_R}	Transceiver HIP digital power (right side)	-0.5	1.35	V
V_{CCR_L}	Receiver power (left side)	-0.5	1.35	V
V_{CCR_R}	Receiver power (right side)	-0.5	1.35	V
V_{CCT_L}	Transmitter power (left side)	-0.5	1.35	V
V_{CCT_R}	Transmitter power (right side)	-0.5	1.35	V
V_{CCL_GXBLn} *(2)*	Transceiver clock power (left side)	-0.5	1.35	V
V_{CCL_GXBRn} *(2)*	Transceiver clock power (right side)	-0.5	1.35	V
V_{CCH_GXBLn} *(2)*	Transmitter output buffer power (left side)	-0.5	1.8	V

Table 1–3. Transceiver Power Supply Absolute Maximum Ratings for Stratix IV GT Devices *(Note 1)* (Part 2 of 2)

Symbol	Description	Minimum	Maximum	Unit
V_{CCH_GXBRn} *(2)*	Transmitter output buffer power (right side)	-0.5	1.8	V

Notes to Table 1–3:

(1) For absolute maximum ratings for Stratix IV GT engineering sample (ES1) devices, contact your local Altera sales representative.

(2) n = 0, 1, 2, or 3.

Maximum Allowed Overshoot and Undershoot Voltage

During transitions, input signals may overshoot to the voltage shown in Table 1–4 and undershoot to -2.0 V for input currents less than 100 mA and periods shorter than 20 ns.

Table 1–4 lists the maximum allowed input overshoot voltage and the duration of the overshoot voltage as a percentage of device lifetime. The maximum allowed overshoot duration is specified as a percentage of high time over the lifetime of the device. A DC signal is equivalent to 100% duty cycle. For example, a signal that overshoots to 4.3 V can only be at 4.3 V for ~5% over the lifetime of the device; for a device lifetime of 10 years, this amounts to half of a year.

Table 1–4. Maximum Allowed Overshoot During Transitions

Symbol	Description	Condition	Overshoot Duration as % of High Time	Unit
Vi (AC)	AC input voltage	4.0 V	100.000	%
		4.05 V	79.330	%
		4.1 V	46.270	%
		4.15 V	27.030	%
		4.2 V	15.800	%
		4.25 V	9.240	%
		4.3 V	5.410	%
		4.35 V	3.160	%
		4.4 V	1.850	%
		4.45 V	1.080	%
		4.5 V	0.630	%
		4.55 V	0.370	%
		4.6 V	0.220	%

Recommended Operating Conditions

This section lists the functional operation limits for AC and DC parameters for Stratix IV devices. Table 1–5 shows the steady-state voltage and current values expected from Stratix IV devices. All supplies must be strictly monotonic, without plateaus.

Table 1–5. Recommended Operating Conditions for Stratix IV Device (Part 1 of 2)

Symbol	Description	Condition	Minimum	Typical	Maximum	Unit
V_{CC} (Stratix IV GX and Stratix IV E)	Core voltage and periphery circuitry power supply	—	0.87	0.90	0.93	V
V_{CC} (Stratix IV GT)	Core voltage and periphery circuitry power supply	—	0.92	0.95	0.98	V

Table 1–5. Recommended Operating Conditions for Stratix IV Device (Part 2 of 2)

Symbol	Description	Condition	Minimum	Typical	Maximum	Unit
V_{CCPT}	Power supply for programmable power technology	—	1.45	1.50	1.55	V
V_{CCAUX}	Auxiliary supply for the programmable power technology	—	2.375	2.5	2.625	V
V_{CCPD} *(2)*	I/O pre-driver (3.0 V) power supply	—	2.85	3.0	3.15	V
	I/O pre-driver (2.5 V) power supply	—	2.375	2.5	2.625	V
V_{CCIO}	I/O buffers (3.0 V) power supply	—	2.85	3.0	3.15	V
	I/O buffers (2.5 V) power supply	—	2.375	2.5	2.625	V
	I/O buffers (1.8 V) power supply	—	1.71	1.8	1.89	V
	I/O buffers (1.5 V) power supply	—	1.425	1.5	1.575	V
	I/O buffers (1.2 V) power supply	—	1.14	1.2	1.26	V
V_{CCPGM}	Configuration pins (3.0 V) power supply	—	2.85	3.0	3.15	V
	Configuration pins (2.5 V) power supply	—	2.375	2.5	2.625	V
	Configuration pins (1.8 V) power supply	—	1.71	1.8	1.89	V
V_{CCA_PLL}	PLL analog voltage regulator power supply	—	2.375	2.5	2.625	V
V_{CCD_PLL} (Stratix IV GX and Stratix IV E)	PLL digital voltage regulator power supply	—	0.87	0.90	0.93	V
V_{CCD_PLL} (Stratix IV GT)	PLL digital voltage regulator power supply	—	0.92	0.95	0.98	V
V_{CC_CLKIN}	Differential clock input power supply	—	2.375	2.5	2.625	V
V_{CCBAT} *(1)*	Battery back-up power supply (For design security volatile key register)	—	1.2	—	3.3	V
V_I	DC input voltage	—	–0.5	—	3.6	V
V_O	Output voltage	—	0	—	V_{CCIO}	V
T_J (Stratix IV GX and Stratix IV E)	Operating junction temperature	Commercial	0	—	85	°C
		Industrial	–40	—	100	°C
T_J (Stratix IV GT)	Operating junction temperature	Industrial	0	—	100	°C
t_{RAMP}	Power supply ramp time	Normal POR (PORSEL=0)	0.05	—	100	ms
		Fast POR (PORSEL=1)	0.05	—	4	ms

Notes to Table 1–5:

(1) Altera recommends a 3.0-V nominal battery voltage when connecting V_{CCBAT} to a battery for volatile key backup. If you do not use the volatile security key, you may connect the V_{CCBAT} to either GND or a 3.0-V power supply.

(2) V_{CCPD} must be 2.5 V when V_{CCIO} is 2.5, 1.8, 1.5, or 1.2 V. V_{CCPD} must be 3.0 V when V_{CCIO} is 3.0 V.

Table 1–6 shows the transceiver power supply recommended operating conditions.

Table 1–6. Transceiver Power Supply Operating Conditions for Stratix IV GX Devices (Part 1 of 2) *(Note 4)*

Symbol	Description	Minimum	Typical	Maximum	Unit
V_{CCA_L}	Transceiver high voltage power (left side)	2.85/2.375	3.0/2.5 *(2)*	3.15/2.625	V
V_{CCA_R}	Transceiver high voltage power (right side)				V
V_{CCHIP_L}	Transceiver HIP digital power (left side)	0.87	0.9	0.93	V

Table 1–6. Transceiver Power Supply Operating Conditions for Stratix IV GX Devices (Part 2 of 2) *(Note 4)*

Symbol	Description	Minimum	Typical	Maximum	Unit
V_{CCHIP_R}	Transceiver HIP digital power (right side)	0.87	0.9	0.93	V
V_{CCR_L}	Receiver power (left side)	1.05	1.1	1.15	V
V_{CCR_R}	Receiver power (right side)	1.05	1.1	1.15	V
V_{CCT_L}	Transmitter power (left side)	1.05	1.1	1.15	V
V_{CCT_R}	Transmitter power (right side)	1.05	1.1	1.15	V
V_{CCL_GXBLn} *(1)*	Transceiver clock power (left side)	1.05	1.1	1.15	V
V_{CCL_GXBRn} *(1)*	Transceiver clock power (right side)	1.05	1.1	1.15	V
V_{CCH_GXBLn} *(1)*	Transmitter output buffer power (left side)	1.33/1.425	1.4/1.5 *(3)*	1.47/1.575	V
V_{CCH_GXBRn} *(1)*	Transmitter output buffer power (right side)				V

Notes to Table 1–6:

(1) n = 0, 1, 2, or 3.

(2) $V_{CCA_L/R}$ must be connected to a 3.0-V supply if the CMU PLL, receiver CDR, or both, are configured at a base data rate > 4.25 Gbps. For data rates up to 4.25 Gbps, you can connect $V_{CCA_L/R}$ to either 3.0 V or 2.5 V.

(3) $V_{CCH_GXBL/R}$ must be connected to a 1.4-V supply if the transmitter channel data rate is > 6.25 Gbps. For data rates up to 6.25 Gbps, you can connect $V_{CCH_GXBL/R}$ to either 1.4 V or 1.5 V.

(4) Transceiver power supplies do not have power-on-reset (POR) circuitry. After initial power-up, violating the transceiver power supply operating conditions could lead to unpredictable link behavior.

Table 1–7 shows the recommended operating conditions for the Stratix IV GT transceiver power supply.

Table 1–7. Transceiver Power Supply Operating Conditions for Stratix IV GT Devices *(Note 1)*, *(3)*

Symbol	Description	Minimum	Typical	Maximum	Unit
V_{CCA_L}	Transceiver high voltage power (left side)	3.25	3.3	3.35	V
V_{CCA_R}	Transceiver high voltage power (right side)	3.25	3.3	3.35	V
V_{CCHIP_L}	Transceiver HIP digital power (left side)	0.92	0.95	0.98	V
V_{CCHIP_R}	Transceiver HIP digital power (right side)	0.92	0.95	0.98	V
V_{CCR_L}	Receiver power (left side)	1.15	1.2	1.25	V
V_{CCR_R}	Receiver power (right side)	1.15	1.2	1.25	V
V_{CCT_L}	Transmitter power (left side)	1.15	1.2	1.25	V
V_{CCT_R}	Transmitter power (right side)	1.15	1.2	1.25	V
V_{CCL_GXBLn} *(2)*	Transceiver clock power (left side)	1.15	1.2	1.25	V
V_{CCL_GXBRn} *(2)*	Transceiver clock power (right side)	1.15	1.2	1.25	V
V_{CCH_GXBLn} *(2)*	Transmitter output buffer power (left side)	1.33	1.4	1.47	V
V_{CCH_GXBRn} *(2)*	Transmitter output buffer power (right side)	1.33	1.4	1.47	V

Notes to Table 1–7:

(1) For recommended operating conditions for Stratix IV GT engineering sample (ES1) devices, contact your local Altera sales representative.

(2) n = 0, 1, 2, or 3.

(3) Transceiver power supplies do not have power-on-reset (POR) circuitry. After initial power-up, violating the transceiver power supply operating conditions could lead to unpredictable link behavior.

DC Characteristics

This section lists the supply current, I/O pin leakage current, input pin capacitance, on-chip termination tolerance, and hot socketing specifications.

Supply Current

Standby current is the current drawn from the respective power rails used for power budgeting. Use the Excel-based Early Power Estimator (EPE) to get supply current estimates for your design because these currents vary largely with the resources you use.

For more information about power estimation tools, refer to the *PowerPlay Early Power Estimator User Guide* and the *PowerPlay Power Analysis* chapter in the *Quartus II Handbook*.

I/O Pin Leakage Current

Table 1–8 defines the Stratix IV I/O pin leakage current specifications.

Table 1–8. I/O Pin Leakage Current for Stratix IV Devices

Symbol	Description	Conditions	Min	Typ	Max	Unit
I_I	Input pin	V_I = 0V to $V_{CCIOMAX}$	-20	—	20	µA
I_{OZ}	Tri-stated I/O pin	V_O = 0V to $V_{CCIOMAX}$	-20	—	20	µA

Bus Hold specifications

Table 1–9 shows the Stratix IV device family bus hold specifications.

Table 1–9. Bus Hold Parameters

Parameter	Symbol	Conditions	V_{CCIO}										Unit
			1.2 V		1.5 V		1.8 V		2.5 V		3.0 V		
			Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	
Low sustaining current	I_{SUSL}	$V_{IN} > V_{IL}$ (maximum)	22.5	—	25.0	—	30.0	—	50.0	—	70.0	—	µA
High sustaining current	I_{SUSH}	$V_{IN} < V_{IH}$ (minimum)	-22.5	—	-25.0	—	-30.0	—	-50.0	—	-70.0	—	µA
Low overdrive current	I_{ODL}	$0V < V_{IN} < V_{CCIO}$	—	120	—	160	—	200	—	300	—	500	µA
High overdrive current	I_{ODH}	$0V < V_{IN} < V_{CCIO}$	—	-120	—	-160	—	-200	—	-300	—	-500	µA
Bus-hold trip point	V_{TRIP}	—	0.45	0.95	0.50	1.00	0.68	1.07	0.70	1.70	0.80	2.00	V

On-Chip Termination (OCT) Specifications

If you enable OCT calibration, calibration is automatically performed at power-up for I/Os connected to the calibration block. Table 1–10 lists the Stratix IV OCT termination calibration accuracy specifications.

Table 1–10. Stratix IV On-Chip Termination Calibration Accuracy Specifications *(Note 1)*

Symbol	Description	Conditions	Calibration Accuracy			Unit
			C2	C3,I3	C4,I4	
25-Ω R_S *(2)* 3.0, 2.5, 1.8, 1.5, 1.2	Internal series termination with calibration (25-Ω setting)	V_{CCIO} = 3.0, 2.5, 1.8, 1.5, 1.2 V	± 8	± 8	± 8	%
50-Ω R_S 3.0, 2.5, 1.8, 1.5, 1.2	Internal series termination with calibration (50-Ω setting)	V_{CCIO} = 3.0, 2.5, 1.8, 1.5, 1.2 V	± 8	± 8	± 8	%
50-Ω R_T 2.5, 1.8, 1.5, 1.2	Internal parallel termination with calibration (50-Ω setting)	V_{CCIO} = 2.5, 1.8, 1.5, 1.2 V	± 10	± 10	± 10	%
20-Ω, 40-Ω, and 60-Ω R_S *(3)* 3.0, 2.5, 1.8, 1.5, 1.2	Expanded range for internal series termination with calibration (20-Ω, 40-Ω , and 60-Ω R_S setting)	V_{CCIO} = 3.0, 2.5, 1.8, 1.5, 1.2 V	± 10	± 10	± 10	%
25-Ω $R_{S_left_shift}$	Internal left shift series termination with calibration (25-Ω $R_{S_left_shift}$ setting)	V_{CCIO} = 3.0, 2.5, 1.8, 1.5, 1.2 V	± 10	± 10	± 10	%

Notes to Table 1–10:

(1) OCT calibration accuracy is valid at the time of calibration only.
(2) 25-Ω R_S is not supported for 1.5 V and 1.2 V in Row I/O.
(3) 20-Ω R_S is not supported for 1.5 V and 1.2 V in Row I/O.

The calibration accuracy for calibrated series and parallel OCTs are applicable at the moment of calibration. When process, voltage, and temperature (PVT) conditions change after calibration, the tolerance may change. Table 1–11 lists the Stratix IV OCT without calibration resistance tolerance to PVT changes.

Table 1–11. Stratix IV On-Chip Termination Without Calibration Resistance Tolerance Specifications (Part 1 of 2)

Symbol	Description	Conditions	Resistance Tolerance			Unit
			C2	C3,I3	C4,I4	
25-Ω R_S 3.0 and 2.5	Internal series termination without calibration (25-Ω setting)	V_{CCIO} = 3.0 and 2.5 V	± 30	± 40	± 40	%
25-Ω R_S 1.8 and 1.5	Internal series termination without calibration (25-Ω setting)	V_{CCIO} = 1.8 and 1.5 V	± 30	± 40	± 40	%
25-Ω R_S 1.2	Internal series termination without calibration (25-Ω setting)	V_{CCIO} = 1.2 V	± 35	± 50	± 50	%
50-Ω R_S 3.0 and 2.5	Internal series termination without calibration (50-Ω setting)	V_{CCIO} = 3.0 and 2.5 V	± 30	± 40	± 40	%

Table 1–11. Stratix IV On-Chip Termination Without Calibration Resistance Tolerance Specifications (Part 2 of 2)

Symbol	Description	Conditions	Resistance Tolerance			Unit
			C2	C3,I3	C4,I4	
50-Ω R_S 1.8 and 1.5	Internal series termination without calibration (50-Ω setting)	V_{CCIO} = 1.8 and 1.5 V	± 30	± 40	± 40	%
50-Ω R_S 1.2	Internal series termination without calibration (50-Ω setting)	V_{CCIO} = 1.2 V	± 35	± 50	± 50	%

OCT calibration is automatically performed at power-up for OCT-enabled I/Os. Table 1–12 lists OCT variation with temperature and voltage after power-up calibration. Use Table 1–12 to determine the OCT variation after power-up calibration and Equation 1–1 to determine the OCT variation without re-calibration.

Equation 1–1. OCT Variation Without Re-Calibration *(Note 1), (2), (3), (4), (5), (6)*

$$R_{OCT} = R_{SCAL}\left(1 + \langle \frac{dR}{dT} \times \Delta T \rangle \pm \langle \frac{dR}{dV} \times \Delta V \rangle\right)$$

Notes to Equation 1–1:

(1) The R_{OCT} value calculated from Equation 1–1 shows the range of OCT resistance with the variation of temperature and V_{CCIO}.

(2) R_{SCAL} is the OCT resistance value at power-up.

(3) ΔT is the variation of temperature with respect to the temperature at power-up.

(4) ΔV is the variation of voltage with respect to the V_{CCIO} at power-up.

(5) dR/dT is the percentage change of R_{SCAL} with temperature.

(6) dR/dV is the percentage change of R_{SCAL} with voltage.

Table 1–12 shows the on-chip termination variation after the power-up calibration.

Table 1–12. On-Chip Termination Variation after Power-Up Calibration *(Note 1)*

Symbol	Description	V_{CCIO}(V)	Typical	Unit
dR/dV	OCT variation with voltage without re-calibration	3.0	0.0297	%/mV
		2.5	0.0344	
		1.8	0.0499	
		1.5	0.0744	
		1.2	0.1241	
dR/dT	OCT variation with temperature without re-calibration	3.0	0.189	%/°C
		2.5	0.208	
		1.8	0.266	
		1.5	0.273	
		1.2	0.317	

Note to Table 1–12:

(1) Valid for V_{CCIO} range of ±5% and temperature range of 0° to 85°C.

Pin Capacitance

Table 1–13 shows the Stratix IV device family pin capacitance.

Table 1–13. Pin Capacitance for Stratix IV Devices

Symbol	Description	Typical	Unit
C_{IOTB}	Input capacitance on top/bottom I/O pins	4	pF
C_{IOLR}	Input capacitance on left/right I/O pins	4	pF
C_{CLKTB}	Input capacitance on top/bottom non-dedicated clock input pins	4	pF
C_{CLKLR}	Input capacitance on left/right non-dedicated clock input pins	4	pF
C_{OUTFB}	Input capacitance on dual-purpose clock output/feedback pins	5	pF
C_{CLK1}, C_{CLK3}, C_{CLK8}, and C_{CLK10}	Input capacitance for dedicated clock input pins	2	pF

Hot Socketing

Table 1–14 shows the hot socketing specification for Stratix IV devices.

Table 1–14. Hot Socketing Specifications for Stratix IV Devices

Symbol	Description	Maximum
$I_{IOPIN\ (DC)}$	DC current per I/O pin	300 µA
$I_{IOPIN\ (AC)}$	AC current per I/O pin	8 mA *(1)*
$I_{XCVR\text{-}TX\ (DC)}$ *(2)*	DC current per transceiver TX pin	100 mA
$I_{XCVR\text{-}RX\ (DC)}$ *(2)*	DC current per transceiver RX pin	50 mA

Notes to Table 1–14:

(1) The I/O ramp rate is 10 ns or more. For ramp rates faster than 10 ns, $|I_{IOPIN}|$ = C dv/dt, in which C is the I/O pin capacitance and dv/dt is the slew rate.

(2) These specifications are preliminary.

Internal Weak Pull-Up Resistor

Table 1–15 lists the weak pull-up resistor values for Stratix IV devices.

Table 1–15. Stratix IV Internal Weak Pull-Up Resistor *(Note 1)*, *(3)*

Symbol	Description	Conditions	Min	Typ	Max	Unit
R_{PU}	Value of the I/O pin pull-up resistor before and during configuration, as well as user mode if the programmable pull-up resistor option is enabled.	V_{CCIO} = 3.0 V ±5% *(2)*	—	25	—	kΩ
		V_{CCIO} = 2.5 V ±5% *(2)*	—	25	—	kΩ
		V_{CCIO} = 1.8 V ±5% *(2)*	—	25	—	kΩ
		V_{CCIO} = 1.5 V ±5% *(2)*	—	25	—	kΩ
		V_{CCIO} = 1.2 V ±5% *(2)*	—	25	—	kΩ

Notes to Table 1–15:

(1) All I/O pins have an option to enable weak pull-up except configuration, test, and JTAG pins.

(2) Pin pull-up resistance values may be lower if an external source drives the pin higher than V_{CCIO}.

(3) The internal weak pull-down feature is only available for the JTAG TCK pin. The typical value for this internal weak pull-down resistor is approximately 25 kΩ .

I/O Standard Specifications

Table 1–16 through Table 1–21 list input voltage (V_{IH} and V_{IL}), output voltage (V_{OH} and V_{OL}), and current drive characteristics (I_{OH} and I_{OL}) for various I/O standards supported by Stratix IV devices. These tables also show the Stratix IV device family I/O standard specifications. V_{OL} and V_{OH} values are valid at the corresponding I_{OH} and I_{OL}, respectively.

For an explanation of terms used in Table 1–16 through Table 1–21, refer to "Glossary" on page 1–49.

Table 1–16. Single-Ended I/O Standards

I/O Standard	V_{CCIO} (V)			V_{IL} (V)		V_{IH} (V)		V_{OL} (V)	V_{OH} (V)	I_{OL} (mA)	I_{OH} (mA)
	Min	Typ	Max	Min	Max	Min	Max	Max	Min		
LVTTL	2.85	3	3.15	-0.3	0.8	1.7	3.6	0.4	2.4	2	-2
LVCMOS	2.85	3	3.15	-0.3	0.8	1.7	3.6	0.2	V_{CCIO} - 0.2	0.1	-0.1
2.5 V	2.375	2.5	2.625	-0.3	0.7	1.7	3.6	0.4	2	1	-1
1.8 V	1.71	1.8	1.89	-0.3	0.35 * V_{CCIO}	0.65 * V_{CCIO}	V_{CCIO} + 0.3	0.45	V_{CCIO} -0.45	2	-2
1.5 V	1.425	1.5	1.575	-0.3	0.35 * V_{CCIO}	0.65 * V_{CCIO}	V_{CCIO} + 0.3	0.25 * V_{CCIO}	0.75 * V_{CCIO}	2	-2
1.2 V	1.14	1.2	1.26	-0.3	0.35 * V_{CCIO}	0.65 * V_{CCIO}	V_{CCIO} + 0.3	0.25 * V_{CCIO}	0.75 * V_{CCIO}	2	-2
3.0-V PCI	2.85	3	3.15	—	0.3 * V_{CCIO}	0.5 * V_{CCIO}	3.6	0.1 * V_{CCIO}	0.9 * V_{CCIO}	1.5	-0.5
3.0-V PCI-X	2.85	3	3.15	—	0.35 * V_{CCIO}	0.5 * V_{CCIO}	—	0.1 * V_{CCIO}	0.9 * V_{CCIO}	1.5	-0.5

Table 1–17. Single-Ended SSTL and HSTL I/O Reference Voltage Specifications

I/O Standard	V_{CCIO} (V)			V_{REF} (V)			V_{TT} (V)		
	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max
SSTL-2 Class I, II	2.375	2.5	2.625	0.49 * V_{CCIO}	0.5 * V_{CCIO}	0.51 * V_{CCIO}	V_{REF} - 0.04	V_{REF}	V_{REF} + 0.04
SSTL-18 Class I, II	1.71	1.8	1.89	0.833	0.9	0.969	V_{REF} - 0.04	V_{REF}	V_{REF} + 0.04
SSTL-15 Class I, II	1.425	1.5	1.575	0.47 * V_{CCIO}	0.5 * V_{CCIO}	0.53 * V_{CCIO}	0.47 * V_{CCIO}	V_{REF}	0.53 * V_{CCIO}
HSTL-18 Class I, II	1.71	1.8	1.89	0.85	0.9	0.95	—	V_{CCIO}/2	—
HSTL-15 Class I, II	1.425	1.5	1.575	0.68	0.75	0.9	—	V_{CCIO}/2	—
HSTL-12 Class I, II	1.14	1.2	1.26	0.47 * V_{CCIO}	0.5 * V_{CCIO}	0.53 * V_{CCIO}	—	V_{CCIO}/2	—

Table 1–18. Single-Ended SSTL and HSTL I/O Standards Signal Specifications

I/O Standard	$V_{IL(DC)}$(V)		$V_{IH(DC)}$(V)		$V_{IL(AC)}$(V)	$V_{IH(AC)}$(V)	V_{OL}(V)	V_{OH}(V)	I_{ol} (mA)	I_{oh} (mA)
	Min	Max	Min	Max	Max	Min	Max	Min		
SSTL-2 Class I	-0.3	V_{REF} - 0.15	V_{REF} + 0.15	V_{CCIO} + 0.3	V_{REF} - 0.31	V_{REF} + 0.31	V_{TT} - 0.57	V_{TT} + 0.57	8.1	-8.1
SSTL-2 Class II	-0.3	V_{REF} - 0.15	V_{REF} + 0.15	V_{CCIO} + 0.3	V_{REF} - 0.31	V_{REF} + 0.31	V_{TT} - 0.76	V_{TT} + 0.76	16.2	-16.2
SSTL-18 Class I	-0.3	V_{REF} - 0.125	V_{REF} + 0.125	V_{CCIO} + 0.3	V_{REF} -0.25	V_{REF} + 0.25	V_{TT} - 0.475	V_{TT} + 0.475	6.7	-6.7
SSTL-18 Class II	-0.3	V_{REF} - 0.125	V_{REF} + 0.125	V_{CCIO} + 0.3	V_{REF} -0.25	V_{REF} + 0.25	0.28	V_{CCIO} - 0.28	13.4	-13.4
SSTL-15 Class I	—	V_{REF} -0.1	V_{REF} + 0.1	—	V_{REF} -0.175	V_{REF} + 0.175	0.2 * V_{CCIO}	0.8 * V_{CCIO}	8	-8
SSTL-15 Class II	—	V_{REF} -0.1	V_{REF} + 0.1	—	V_{REF} -0.175	V_{REF} + 0.175	0.2 * V_{CCIO}	0.8 * V_{CCIO}	16	-16
HSTL-18 Class I	—	V_{REF} -0.1	V_{REF} + 0.1	—	V_{REF} -0.2	V_{REF} + 0.2	0.4	V_{CCIO} - 0.4	8	-8
HSTL-18 Class II	—	V_{REF} -0.1	V_{REF} + 0.1	—	V_{REF} -0.2	V_{REF} + 0.2	0.4	V_{CCIO} - 0.4	16	-16
HSTL-15 Class I	—	V_{REF} -0.1	V_{REF} + 0.1	—	V_{REF} -0.2	V_{REF} + 0.2	0.4	V_{CCIO} - 0.4	8	-8
HSTL-15 Class II	—	V_{REF} -0.1	V_{REF} + 0.1	—	V_{REF} -0.2	V_{REF} + 0.2	0.4	V_{CCIO} - 0.4	16	-16
HSTL-12 Class I	-0.15	V_{REF} - 0.08	V_{REF} + 0.08	V_{CCIO} + 0.15	V_{REF} -0.15	V_{REF} + 0.15	0.25* V_{CCIO}	0.75* V_{CCIO}	8	-8
HSTL-12 Class II	-0.15	V_{REF} - 0.08	V_{REF} + 0.08	V_{CCIO} + 0.15	V_{REF} -0.15	V_{REF} + 0.15	0.25* V_{CCIO}	0.75* V_{CCIO}	16	-16

Table 1–19. Differential SSTL I/O Standards

I/O Standard	V_{CCIO}(V)			$V_{SWING(DC)}$(V)		$V_{X(AC)}$(V)			$V_{SWING(AC)}$(V)		$V_{OX(AC)}$(V)		
	Min	Typ	Max	Min	Max	Min	Typ	Max	Min	Max	Min	Typ	Max
SSTL-2 Class I, II	2.375	2.5	2.625	0.3	V_{CCIO} + 0.6	V_{CCIO}/2 - 0.2	—	V_{CCIO}/2 + 0.2	0.62	V_{CCIO} + 0.6	V_{CCIO}/2 - 0.15	—	V_{CCIO}/2 + 0.15
SSTL-18 Class I, II	1.71	1.8	1.89	0.25	V_{CCIO} + 0.6	V_{CCIO}/2 - 0.175	—	V_{CCIO}/2 + 0.175	0.5	V_{CCIO} + 0.6	V_{CCIO}/2 - 0.125	—	V_{CCIO}/2 + 0.125
SSTL-15 Class I, II	1.425	1.5	1.575	0.2	—	—	V_{CCIO}/2	—	0.35	—	—	V_{CCIO}/2	—

Table 1–20. Differential HSTL I/O Standards (Part 1 of 2)

I/O Standard	V_{CCIO}(V)			$V_{DIF(DC)}$(V)		$V_{X(AC)}$(V)			$V_{CM(DC)}$(V)			$V_{DIF(AC)}$(V)	
	Min	Typ	Max	Min	Max	Min	Typ	Max	Min	Typ	Max	Min	Max
HSTL-18 Class I	1.71	1.8	1.89	0.2	—	0.78	—	1.12	0.78	—	1.12	0.4	—

Table 1–20. Differential HSTL I/O Standards (Part 2 of 2)

I/O Standard	V_{CCIO}(V)			$V_{DIF(DC)}$(V)		$V_{X(AC)}$(V)			$V_{CM(DC)}$(V)			$V_{DIF(AC)}$(V)	
	Min	Typ	Max	Min	Max	Min	Typ	Max	Min	Typ	Max	Min	Max
HSTL-15 Class I, II	1.425	1.5	1.575	0.2	—	0.68	—	0.9	0.68	—	0.9	0.4	—
HSTL-12 Class I, II	1.14	1.2	1.26	0.16	V_{CCIO} + 0.3	—	0.5* V_{CCIO}	—	0.4* V_{CCIO}	0.5* V_{CCIO}	0.6* V_{CCIO}	0.3	V_{CCIO} + 0.48

Table 1–21. Differential I/O Standard Specifications *(Note 1)*, *(2)*

I/O Standard	V_{CCIO}(V)			V_{ID}(mV)			$V_{ICM(DC)}$(V)			V_{OD}(V) *(3)*			V_{OCM}(V) *(3)*		
	Min	Typ	Max	Min	Condition	Max	Min	Condition	Max	Min	Typ	Max	Min	Typ	Max
PCML	Transmitter, receiver, and input reference clock pins of high-speed transceivers use PCML I/O standard. For transmitter, receiver, and reference clock I/O pin specifications, refer to Table 1–22 on page 1–13 and Table 1–23 on page 1–20.														
2.5 V LVDS (HIO)	2.375	2.5	2.625	100	V_{CM} = 1.25 V	—	0.05	D_{MAX} ≤ 700 Mbps	1.8	0.247	—	0.6	1.125	1.25	1.375
						—	1.05	D_{MAX} > 700 Mbps	1.55	0.247	—	0.6	1.125	1.25	1.375
2.5 V LVDS (VIO)	2.375	2.5	2.625	100	V_{CM} = 1.25 V	—	0.05	D_{MAX} ≤ 700 Mbps	1.8	0.247	—	0.6	1	1.25	1.5
						—	1.05	D_{MAX} > 700 Mbps	1.55	0.247	—	0.6	1	1.25	1.5
RSDS (HIO)	2.375	2.5	2.625	100	V_{CM} = 1.25 V	—	0.3	—	1.4	0.1	0.2	0.6	0.5	1.2	1.4
RSDS (VIO)	2.375	2.5	2.625	100	V_{CM} = 1.25 V	—	0.3	—	1.4	0.1	0.2	0.6	0.5	1.2	1.5
Mini-LVDS (HIO)	2.375	2.5	2.625	200	—	600	0.4	—	1.325	0.25	—	0.6	1	1.2	1.4
Mini-LVDS (VIO)	2.375	2.5	2.625	200	—	600	0.4	—	1.325	0.25	—	0.6	1	1.2	1.5
LVPECL	2.375	2.5	2.625	300	—	—	0.6	D_{MAX} ≤ 700 Mbps	1.8 *(4)*	—	—	—	—	—	—
	2.375	2.5	2.625	300	—	—	1	D_{MAX} > 700 Mbps	1.6 *(4)*	—	—	—	—	—	—

Notes to Table 1–21:

(1) Vertical (VIO) is top and bottom I/Os; horizontal I/O (HIO) is left and right I/Os.

(2) 1.4-V/1.5-V PCML transceiver I/O standard specifications are described in "Transceiver Performance Specifications" on page 1–13.

(3) RL range: 90 ≤ RL ≤ 110 Ω.

(4) For D_{MAX} > 700 Mbps, the minimum input voltage is 0.85 V; the maximum input voltage is 1.75 V. For F_{MAX} ≤ 700 Mbps, the minimum input voltage is 0.45 V; the maximum input voltage is 1.95 V.

Power Consumption

Altera offers two ways to estimate power consumption for a design: the Excel-based Early Power Estimator and the Quartus® II PowerPlay Power Analyzer feature.

You typically use the interactive Excel-based Early Power Estimator before designing the FPGA to get a magnitude estimate of the device power. The Quartus II PowerPlay Power Analyzer provides better quality estimates based on the specifics of the design after you complete place-and-route. The PowerPlay Power Analyzer can apply a combination of user-entered, simulation-derived, and estimated signal activities that, combined with detailed circuit models, yields very accurate power estimates.

For more information about power estimation tools, refer to the *PowerPlay Early Power Estimator User Guide for Stratix III and Stratix IV FPGAs* and the *PowerPlay Power Analysis* chapter in the *Quartus II Handbook*.

Switching Characteristics

This section provides performance characteristics of Stratix IV core and periphery blocks for commercial grade devices.

These characteristics can be designated as Preliminary or Final.

- Preliminary characteristics are created using simulation results, process data, and other known parameters. The title of these tables show the designation as "Preliminary".
- Final numbers are based on actual silicon characterization and testing. The numbers reflect the actual performance of the device under worst-case silicon process, voltage, and junction temperature conditions. There are no designations on finalized tables.

Transceiver Performance Specifications

This section describes transceiver performance specifications.

Table 1–22 lists the Stratix IV GX transceiver specifications.

Table 1–22. Transceiver Specification for Stratix IV GX Devices (Part 1 of 7)

Symbol/ Description	Conditions	–2 Commercial Speed Grade			–3 Commercial/Industrial and –2× Commercial Speed Grade *(1)*			–4 Commercial/Industrial Speed Grade			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Reference Clock											
Supported I/O Standards	1.2 V PCML, 1.5 V PCML, 2.5 V PCML, Differential LVPECL, LVDS, HCSL										
Input frequency from REFCLK input pins	—	50	—	697	50	—	697	50	—	637.5	MHz
Phase frequency detector (CMU PLL and receiver CDR)	—	50	—	425	50	—	325	50	—	325	MHz
Absolute V_{MAX} for a REFCLK pin	—	—	—	1.6	—	—	1.6	—	—	1.6	V

Table 1–22. Transceiver Specification for Stratix IV GX Devices (Part 2 of 7)

Symbol/ Description	Conditions	–2 Commercial Speed Grade			–3 Commercial/Industrial and –2× Commercial Speed Grade *(1)*			–4 Commercial/Industrial Speed Grade			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Operational V_{MAX} for a REFCLK pin	—	—	—	1.5	—	—	1.5	—	—	1.5	V
Absolute V_{MIN} for a REFCLK pin	—	-0.4	—	—	-0.4	—	—	-0.4	—	—	V
Rise/fall time	—	—	—	0.2	—	—	0.2	—	—	0.2	UI
Duty cycle	—	45	—	55	45	—	55	45	—	55	%
Peak-to-peak differential input voltage	—	200	—	1600	200	—	1600	200	—	1600	mV
Spread-spectrum modulating clock frequency	PCI Express	30	—	33	30	—	33	30	—	33	kHz
Spread-spectrum downspread	PCI Express	—	0 to -0.5%	—	—	0 to -0.5%	—	—	0 to -0.5%	—	—
On-chip termination resistors	—	—	100	—	—	100	—	—	100	—	Ω
V_{ICM} (AC coupled)	—	—	1100	—	—	1100	—	—	1100	—	mV
V_{ICM} (DC coupled)	HCSL I/O standard for PCI Express reference clock	250	—	550	250	—	550	250	—	550	mV
R_{REF}	—	—	2000± 1%	—	—	2000 ± 1%	—	—	2000 ± 1%	—	Ω
Transceiver Clocks											
Calibration block clock frequency	—	10	—	125	10	—	125	10	—	125	MHz
fixedclk clock frequency	PCI Express Receiver Detect	—	125	—	—	125	—	—	125	—	MHz
reconfig_clk clock frequency	Dynamic reconfiguration clock frequency	2.5/ 37.5 *(2)*	—	50	2.5/ 37.5 *(2)*	—	50	2.5/ 37.5 *(2)*	—	50	—
Delta time between reconfig_clks *(17)*	—	—	—	2	—	—	2	—	—	2	ms
Transceiver block minimum power-down (gxb_powerdown) pulse width	—	—	1	—	—	1	—	—	1	—	µs
Receiver											
Supported I/O Standards	1.4 V PCML, 1.5 V PCML, 2.5 V PCML, LVPECL, LVDS										
Data rate (Single width, non-PMA Direct)	—	600	—	3750	600	—	3750	600	—	3750	Mbps
Data rate (Double width, non-PMA Direct)	—	600	—	8500	600	—	6500	600	—	5000	Mbps
Data rate (Single width, PMA Direct)	—	600	—	3250	600	—	3250	600	—	3250	Mbps

Table 1–22. Transceiver Specification for Stratix IV GX Devices (Part 3 of 7)

Symbol/ Description	Conditions	–2 Commercial Speed Grade			–3 Commercial/Industrial and –2× Commercial Speed Grade *(1)*			–4 Commercial/Industrial Speed Grade			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Data rate (Double width, PMA Direct)	—	1000	—	6500	1000	—	6500	1000	—	5000	Mbps
Absolute V_{MAX} for a receiver pin *(3)*	—	—	—	1.6	—	—	1.6	—	—	1.6	V
Operational V_{MAX} for a receiver pin	—	—	—	1.5	—	—	1.5	—	—	1.5	V
Absolute V_{MIN} for a receiver pin	—	-0.4	—	—	-0.4	—	—	-0.4	—	—	V
Maximum peak-to-peak differential input voltage V_{ID} (diff p-p) before device configuration	—	—	—	1.6	—	—	1.6	—	—	1.6	V
Maximum peak-to-peak differential input voltage V_{ID} (diff p-p) after device configuration	V_{ICM} = 0.82 V setting	—	—	2.7	—	—	2.7	—	—	2.7	V
	V_{ICM} =1.1 V setting *(4)*	—	—	1.6	—	—	1.6	—	—	1.6	V
Minimum peak-to-peak differential input voltage V_{ID} (diff p-p) *(18)*	Data Rate = 600 Mbps to 5 Gbps	100	—	—	100	—	—	165	—	—	mV
	Data Rate > 5 Gbps	165	—	—	165	—	—	—	—	—	mV
V_{ICM}	V_{ICM} = 0.82 V setting	—	820	—	—	820	—	—	820	—	mV
	V_{ICM} = 1.1 V setting *(4)*	—	1100	—	—	1100	—	—	1100	—	mV
Receiver DC Coupling Support	—	For more information about receiver DC coupling support, refer to the "DC-Coupled Links" section in the *Stratix IV Transceiver Architecture* chapter.									
Differential on-chip termination resistors	85-Ω setting	—	85	—	—	85	—	—	85	—	Ω
	100-Ω setting	—	100	—	—	100	—	—	100	—	Ω
	120-Ω setting	—	120	—	—	120	—	—	120	—	Ω
	150-Ω setting	—	150	—	—	150	—	—	150	—	Ω
Differential and common mode return loss	PCI Express (PIPE) (Gen 1 and Gen 2), XAUI, HiGig+, CEI SR/LR, Serial RapidIO SR/LR, CPRI LV/HV, OBSAI, SATA	Compliant									—
Programmable PPM detector *(5)*	—	± 62.5, 100, 125, 200, 250, 300, 500, 1000									ppm
Run length	—	—	200	—	—	200	—	—	200	—	UI
Programmable equalization *(16)*	—	—	—	16	—	—	16	—	—	16	dB

Table 1–22. Transceiver Specification for Stratix IV GX Devices (Part 4 of 7)

Symbol/ Description	Conditions	–2 Commercial Speed Grade			–3 Commercial/Industrial and –2× Commercial Speed Grade *(1)*			–4 Commercial/Industrial Speed Grade			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
t_{LTR} *(6)*	—	—	—	75	—	—	75	—	—	75	µs
$t_{LTR_LTD_Manual}$ *(7)*	—	15	—	—	15	—	—	15	—	—	µs
t_{LTD_Manual} *(8)*	—	—	—	4000	—	—	4000	—	—	4000	ns
t_{LTD_Auto} *(9)*	—	—	—	4000	—	—	4000	—	—	4000	ns
Receiver CDR 3 dB Bandwidth in lock-to-data (LTD) mode	PCI Express (PIPE) Gen1	23									MHz
	PCI Express (PIPE) Gen2	47									MHz
	(OIF) CEI PHY at 6.375 Gbps	22.5									MHz
	XAUI	12									MHz
	Serial RapidIO 1.25 Gbps	56									MHz
	Serial RapidIO 2.5 Gbps	26									MHz
	Serial RapidIO 3.125 Gbps	76									MHz
	GIGE	20									MHz
	SONET OC12	31									MHz
	SONET OC48	18									MHz
Receiver buffer and CDR offset cancellation time (per channel)	—	—	—	17000	—	—	17000	—	—	17000	`reconfig_clk` cycles
Programmable DC gain	DC Gain Setting = 0	—	0	—	—	0	—	—	0	—	dB
	DC Gain Setting = 1	—	3	—	—	3	—	—	3	—	dB
	DC Gain Setting = 2	—	6	—	—	6	—	—	6	—	dB
	DC Gain Setting = 3	—	9	—	—	9	—	—	9	—	dB
	DC Gain Setting = 4	—	12	—	—	12	—	—	12	—	dB
EyeQ Max Data Rate	—	—	6	—	—	6	—	—	5	—	Gbps
Transmitter											
Supported I/O Standards	1.4 V PCML, 1.5 V PCML										
Data rate (Single width, non-PMA Direct)	—	600	—	3750	600	—	3750	600	—	3750	Mbps
Data rate (Double width, non-PMA Direct)	—	1000	—	8500	1000	—	6500	1000	—	5000	Mbps
Data rate (Single width, PMA Direct)	—	600	—	3250	600	—	3250	600	—	3250	Mbps
Data rate (Double width, PMA Direct) *(10)*	—	1000	—	6500	1000	—	6500	1000	—	5000	Mbps

Table 1–22. Transceiver Specification for Stratix IV GX Devices (Part 5 of 7)

Symbol/ Description	Conditions	–2 Commercial Speed Grade			–3 Commercial/Industrial and –2× Commercial Speed Grade *(1)*			–4 Commercial/Industrial Speed Grade			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
V_{OCM}	0.65 V setting	—	650	—	—	650	—	—	650	—	mV
Differential on-chip termination resistors	85-Ω setting	—	85	—	—	85	—	—	85	—	Ω
	100-Ω setting	—	100	—	—	100	—	—	100	—	Ω
	120-Ω setting	—	120	—	—	120	—	—	120	—	Ω
	150-Ω setting	—	150	—	—	150	—	—	150	—	Ω
Differential and common mode return loss	PCI Express (PIPE) Gen1 and Gen2 (TX V_{OD}=4), XAUI (TX V_{OD}=6), HiGig+ (TX V_{OD}=6), CEI SR/LR (TX V_{OD}=8), Serial RapidIO SR (V_{OD}=6), Serial RapidIO LR (V_{OD}=8), CPRI LV (V_{OD}=6), CPRI HV (V_{OD}=2), OBSAI (V_{OD}=6), SATA (V_{OD}=4),	Compliant									—
Rise time *(11)*	—	50	—	200	50	—	200	50	—	200	ps
Fall time *(11)*	—	50	—	200	50	—	200	50	—	200	ps
Intra differential pair skew	—	—	—	15	—	—	15	—	—	15	ps
Intra-transceiver block transmitter channel-to-channel skew	×4 PMA and PCS bonded mode Example: XAUI, PCI EXpress (PIPE), ×4, Basic ×4	—	—	120	—	—	120	—	—	120	ps
Inter-transceiver block transmitter channel-to-channel skew	×8 PMA and PCS bonded mode Example: PCI Express (PIPE) ×8, Basic ×8	—	—	500	—	—	500	—	—	500	ps
Inter-transceiver block skew xN PMA-Only bonded *(12)*	Basic (PMA Direct) xN (For N < 18 channels located across three transceiver blocks)	—	—	*(13)*	—	—	*(13)*	—	—	*(13)*	ps
Inter-transceiver block skew xN PMA-Only bonded *(12)*	Basic (PMA Direct) xN (For N ≥ 18 channels located across four transceiver blocks)	*(13)*									ps
CMU PLL0 and CMU PLL1											
Supported Data Range	—	600	—	8500	600	—	6500	600	—	5000	Mbps

Table 1–22. Transceiver Specification for Stratix IV GX Devices (Part 6 of 7)

Symbol/ Description	Conditions	–2 Commercial Speed Grade			–3 Commercial/Industrial and –2× Commercial Speed Grade *(1)*			–4 Commercial/Industrial Speed Grade			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
CMU PLL lock time from `pll_powerdown` de-assertion	—	—	—	100	—	—	100	—	—	100	µs
-3 dB Bandwidth	PCI Express (PIPE) Gen1	1.6									MHz
	PCI Express (PIPE) Gen2	5.6									MHz
	(OIF) CEI PHY at 4.976 Gbps	4.5									MHz
	(OIF) CEI PHY at 6.375 Gbps	4.85									MHz
	XAUI	1.1									MHz
	Serial RapidIO 1.25 Gbps	2.5									MHz
	Serial RapidIO 2.5 Gbps	2.25									MHz
	Serial RapidIO 3.125 Gbps	2.75									MHz
	GIGE	2.3									MHz
	SONET OC12	2									MHz
	SONET OC48	2.75									MHz
ATX PLL (6G)											
Supported Data Range *(14)*	/L = 1	4800-5400 and 6000-6500			4800-5400 and 6000-6500			—			Mbps
	/L = 2	2400-2700 and 3000-3250			2400-2700 and 3000-3250			—			Mbps
	/L = 4	1200-1350 and 1500-1625			1200-1350 and 1500-1625			—			Mbps
-3 dB Bandwidth	PCI Express (PIPE) Gen 2	—	1.5	—	—	1.5	—	—	—	—	MHz
	(OIF) CEI PHY at 6.375 Gbps	—	3.4	—	—	3.4	—	—	—	—	
Transceiver-FPGA Fabric Interface											
Interface speed (non-PMA Direct)	—	25	—	265.625	25	—	250	25	—	250	MHz
Interface speed (PMA Direct)	—	50	—	325	50	—	325	50	—	325	MHz

Table 1–22. Transceiver Specification for Stratix IV GX Devices (Part 7 of 7)

Symbol/ Description	Conditions	–2 Commercial Speed Grade			–3 Commercial/Industrial and –2× Commercial Speed Grade *(1)*			–4 Commercial/Industrial Speed Grade			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Digital reset pulse width	—	Minimum is two parallel clock cycles									—

Notes to Table 1–22:

(1) The –2× speed grade is the fastest speed grade offered in the following Stratix IV GX devices: EP4SGX70DF29, EP4SGX110DF29, EP4SGX110FF35, EP4SGX230DF29, EP4SGX110FF35, EP4SGX230DF29, EP4SGX230FF35, EP4SGX290FF35, EP4SGX290FH29, EP4SGX360FF35, and EPSGX360FH29.

(2) The minimum `reconfig_clk` frequency is 2.5 MHz if the transceiver channel is configured in transmitter only mode. The minimum `reconfig_clk` frequency is 37.5 MHz if the transceiver channel is configured in receiver only or receiver and transmitter mode. For more information, refer to the *Stratix IV Dynamic Reconfiguration* chapter in volume 2 of the *Stratix IV Device Handbook*.

(3) The device cannot tolerate prolonged operation at this absolute maximum.

(4) You must use the 1.1-V RX V_{ICM} setting if the input serial data standard is LVDS.

(5) The rate matcher supports only up to ±300 ppm.

(6) Time taken to `rx_pll_locked` goes high from `rx_analogreset` de-assertion. Refer to Figure 1–2 on page 1–27.

(7) Time for which the CDR must be kept in lock-to-reference mode after `rx_pll_locked` goes high and before `rx_locktodata` is asserted in manual mode. Refer to Figure 1–2 on page 1–27.

(8) Time taken to recover valid data after the `rx_locktodata` signal is asserted in manual mode. Refer to Figure 1–2 on page 1–27.

(9) Time taken to recover valid data after the `rx_freqlocked` signal goes high in automatic mode. Refer to Figure 1–3 on page 1–27.

(10) A GPLL may be required to meet PMA-FPGA fabric interface timing above certain data rates. Refer to the "Left/Right PLL Requirements in Basic (PMA Direct) Mode" section in the *Stratix IV GX Transceiver Clocking* chapter in volume 2 of the *Stratix IV Device Handbook*.

(11) The Quartus II software automatically selects the appropriate slew rate depending on the configured data rate or functional mode.

(12) For applications that require low transmit lane-to-lane skew, use Basic (PMA Direct) xN to achieve PMA-Only bonding across all channels in the link. You can bond all channels on one side of the device by configuring them in Basic (PMA Direct) xN mode. For more information about clocking requirements in this mode, refer to the "Basic (PMA Direct) Mode Clocking" section in the *Stratix IV GX Transceiver Clocking* chapter in volume 2 of the *Stratix IV Device Handbook*.

(13) Pending Characterization.

(14) The Quartus II software automatically selects the appropriate /L divider depending upon the configured data.

(15) The maximum Transceiver-FPGA Fabric interface speed of 265.625 MHz is allowed only in Basic Low-Latency PCS mode with a 32-bit interface width. For more information, refer to the "Basic Double-Width Mode Configurations" section in the *Stratix IV Transceiver Architecture* chapter in volume 2 of the *Stratix IV Device Handbook*.

(16) Figure 1–1 shows the AC gain curves for each of the 16 available equalization settings.

(17) If your design uses more than one dynamic reconfiguration controller (`altgx_reconfig`) instances to control the transceiver (`altgx`) channels physically located on the same side of the device AND if you use different `reconfig_clk` sources for these `altgx_reconfig` instances, the delta time between any two of these `reconfig_clk` sources becoming stable must not exceed the maximum specification listed.

(18) The outer envelope of the eye diagram measured at the receiver input pins must meet the minimum V_{ID} specifications if **RX equalization** is enabled. The inner envelope of the eye diagram measured at the receiver input pins must meet the minimum V_{ID} specifications if **RX equalization** is not enabled.

Figure 1–1 shows the top-to-bottom AC gain curve for equalization settings 0 to 15.

Figure 1–1. AC Gain Curves for Equalization Settings 0 to 15 (Bottom to Top)

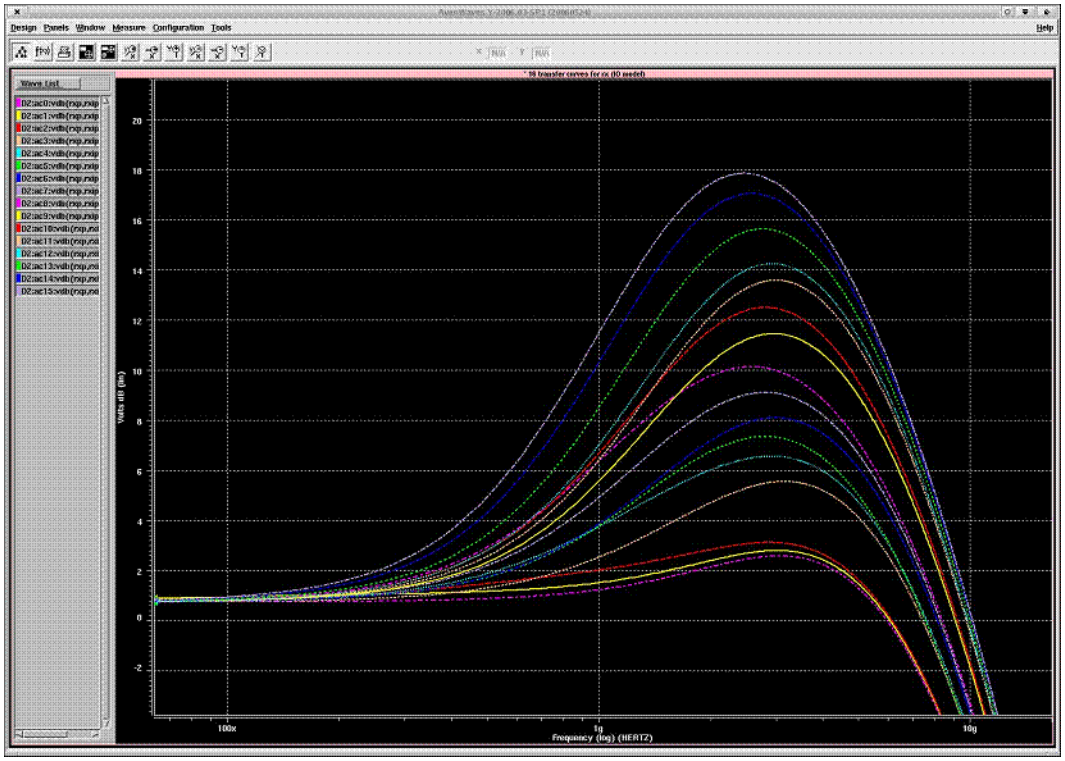


Table 1–23 lists Stratix IV GT transceiver specifications.

Table 1–23. Transceiver Specification for Stratix IV GT Devices (Part 1 of 7)

Symbol/ Description	Conditions	–1 Industrial Speed Grade			–2 Industrial Speed Grade			–3 Industrial Speed Grade			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Reference Clock											
Supported I/O Standards	1.2 V PCML, 1.5 V PCML, 2.5 V PCML, Differential LVPECL, LVDS										
Input frequency from REFCLK input pins	—	50	—	706.25	50	—	706.25	50	—	706.25	MHz
Phase frequency detector (CMU PLL and receiver CDR)	—	50	—	425	50	—	425	50	—	425	MHz
Absolute V_{MAX} for a REFCLK pin	—	—	—	1.6	—	—	1.6	—	—	1.6	V
Operational V_{MAX} for a REFCLK pin	—	—	—	1.5	—	—	1.5	—	—	1.5	V
Absolute V_{MIN} for a REFCLK pin	—	-0.3	—	—	-0.3	—	—	-0.3	—	—	V
Rise/fall time	—	—	0.2	—	—	—	0.2	—	—	0.2	UI
Duty cycle	—	Min = 45 Max = 55	—	—	45	—	55	45	—	55	%

Table 1–23. Transceiver Specification for Stratix IV GT Devices (Part 2 of 7)

Symbol/ Description	Conditions	–1 Industrial Speed Grade			–2 Industrial Speed Grade			–3 Industrial Speed Grade			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Peak-to-peak differential input voltage after device configuration	—	200	—	—	200	—	1600	200	—	1600	mV
Max peak-to-peak differential input voltage before device configuration	—	—	—	900	—	—	900	—	—	900	mV
On-chip termination resistors	—	—	—	100	—	100	—	—	100	—	Ω
V_{ICM}	—	—	—	1200	—	1200	—	—	1200	—	mV
R_{REF}	—	—	—	2000 ± 1%	—	2000 ± 1%	—	—	2000 ± 1%	—	Ω
Transceiver Clocks											
Calibration block clock frequency	—	10	—	125	10	—	125	10	—	125	MHz
`reconfig_clk` clock frequency	Dynamic reconfiguration clock frequency	2.5/ 37.5 *(1)*	—	—	2.5/ 37.5 *(1)*	—	50	2.5/ 37.5 *(1)*	—	50	MHz
Transceiver block minimum (`gxb_powerdown`) power-down pulse width	—	—	1	—	—	1	—	—	1	—	µs
Receiver											
Supported I/O Standards	1.4 V PCML, 1.5 V PCML, 2.5 V PCML, LVPECL, LVDS										
Data rate (Single width, non-PMA Direct)	—	2488	—	3750	2488	—	3750	2488	—	3750	Mbps
Data rate (Double width, non-PMA Direct)	—	2488	—	11300	2488	-	10312.5	2488	—	8500	Mbps
Data rate (Single width, PMA Direct)	—	2488	-	3250	2488	-	3250	2488	—	3250	Mbps
Data rate (Double width, PMA Direct)	—	2488	-	6500	2488	-	6500	2488	—	6500	Mbps
Absolute V_{MAX} for a receiver pin *(2)*	—	—	—	1.6	—	—	1.6	—	—	1.6	V
Operational V_{MAX} for a receiver pin	—	—	—	1.5	—	—	1.5	—	—	1.5	V
Absolute V_{MIN} for a receiver pin	—	—	-0.4	—	-0.4	—	—	-0.4	—	—	V
Maximum peak-to-peak differential input voltage V_{ID} (diff p-p) before device configuration	—	1.6	—	1.6	—	—	1.6	—	—	1.6	V

Table 1–23. Transceiver Specification for Stratix IV GT Devices (Part 3 of 7)

Symbol/ Description	Conditions	–1 Industrial Speed Grade			–2 Industrial Speed Grade			–3 Industrial Speed Grade			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Maximum peak-to-peak differential input voltage V_{ID} (diff p-p) after device configuration	V_{ICM} = 0.82 V setting	—	—	2.7	—	—	2.7	—	—	2.7	V
	V_{ICM} = 1.2 V setting *(3)*	—	—	1.2	—	—	1.2	—	—	1.2	V
Minimum peak-to-peak differential input voltage V_{ID} (diff p-p)	Data Rate = 2488 Mbps to 5 Gbps	100	—	—	100	—	—	100	—	—	mV
	Data Rate > 5 Gbps	165	—	—	165	—	—	165	—	—	mV
V_{ICM}	V_{ICM} = 0.82 V setting	—	—	—	—	820	—	—	820	—	mV
	V_{ICM} = 1.2 V setting *(3)*	—	—	—	—	1200	—	—	1200	—	mV
Differential on-chip termination resistors	85-Ω setting	—	85	—	—	85	—	—	85	—	Ω
	100-Ω setting	—	100	—	—	100	—	—	100	—	Ω
	120-Ω setting	—	120	—	—	120	—	—	120	—	Ω
	150-Ω setting	—	150	—	—	150	—	—	150	—	Ω
Differential and common mode return loss	PCI Express (PIPE) (Gen 1 and Gen 2), XAUI, HiGig+, CEI SR/LR, Serial RapidIO SR/LR, CPRI LV/HV, OBSAI, SATA	Compliant									—
Programmable PPM detector *(4)*	—	—	± 62.5, 100, 125, 200, 250, 300, 500, 1000								ppm
Run length	—	—	200	—	—	200	—	—	200	—	UI
Programmable equalization	—	—	—	16	—	—	16	—	—	16	dB
t_{LTR} *(5)*	—	—	—	75	—	—	75	—	—	75	µs
$t_{LTR_LTD_Manual}$ *(6)*	—	15	—	—	15	—	—	15	—	—	µs
t_{LTD_Manual} *(7)*	—	—	—	4000	—	—	4000	—	—	4000	ns
t_{LTD_Auto} *(8)*	—	—	—	4000	—	—	4000	—	—	4000	ns
Receiver buffer and CDR offset cancellation time (per channel)	—	—	—	17000	—	—	17000	—	—	17000	`reconfig_clk` cycles

Table 1–23. Transceiver Specification for Stratix IV GT Devices (Part 4 of 7)

Symbol/ Description	Conditions	–1 Industrial Speed Grade			–2 Industrial Speed Grade			–3 Industrial Speed Grade			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Programmable DC gain	DC Gain Setting = 0	—	0	—	—	0	—	—	0	—	dB
	DC Gain Setting = 1	—	3	—	—	3	—	—	3	—	dB
	DC Gain Setting = 2	—	6	—	—	6	—	—	6	—	dB
	DC Gain Setting = 3	—	9	—	—	9	—	—	9	—	dB
	DC Gain Setting = 4	—	12	—	—	12	—	—	12	—	dB
Transmitter											
Supported I/O Standards	1.4 V PCML										
Data rate (Single width, non-PMA Direct)	—	2488	—	3750	2488	—	3750	2488	—	3750	Mbps
Data rate (Double width, non-PMA Direct)	—	2488	—	11300	2488	—	10312.5	2488	—	8500	Mbps
Data rate (Single width, PMA Direct)	—	2488	—	3250	2488	—	3250	2488	—	3250	Mbps
Data rate (Double width, PMA Direct) *(9)*	—	2488	—	6500	2488	—	6500	2488	—	6500	Mbps
V_{OCM}	0.65 V setting	—	650	—	—	650	—	—	650	—	mV
Differential on-chip termination resistors	85-Ω setting	—	85	—	—	85	—	—	85	—	Ω
	100-Ω setting	—	100	—	—	100	—	—	100	—	Ω
	120-Ω setting	—	120	—	—	120	—	—	120	—	Ω
	150-Ω setting	—	150	—	—	150	—	—	150	—	Ω

Table 1–23. Transceiver Specification for Stratix IV GT Devices (Part 5 of 7)

Symbol/ Description	Conditions	–1 Industrial Speed Grade			–2 Industrial Speed Grade			–3 Industrial Speed Grade			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Differential and common mode return loss	PCI Express (PIPE) Gen1 and Gen2 (TX V_{OD}=4), XAUI (TX V_{OD}=6), HiGig+ (TX V_{OD}=6), CEI SR/LR (TX V_{OD}=8), Serial RapidIO SR (V_{OD}=6), Serial RapidIO LR (V_{OD}=8), CPRI LV (V_{OD}=6), CPRI HV (V_{OD}=2), OBSAI (V_{OD}=6), SATA (V_{OD}=4),	Compliant									—
Rise time *(10)*	—	50	—	200	50	—	200	50	—	200	ps
Fall time *(10)*	—	50	—	200	50	—	200	50	—	200	ps
Intra-differential pair skew	—	—	—	15	—	—	15	—	—	15	ps
Intra-transceiver block transmitter channel-to-channel skew	×4 PMA and PCS bonded mode Example: XAUI, PCI Express (PIPE) ×4, Basic ×4	—	—	120	—	—	120	—	—	120	ps
Inter-transceiver block transmitter channel-to-channel skew	×8 PMA and PCS bonded mode Example: PCI Express (PIPE) ×8 Basic ×8	—	—	300	—	—	300	—	—	300	ps
Inter-transceiver block skew xN PMA-Only bonded *(11)*	Basic (PMA Direct) xN (For N < 18 channels located across three transceiver blocks)	*(12)*									ps

Table 1–23. Transceiver Specification for Stratix IV GT Devices (Part 6 of 7)

Symbol/ Description	Conditions	–1 Industrial Speed Grade			–2 Industrial Speed Grade			–3 Industrial Speed Grade			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Inter-transceiver block skew xN PMA-Only bonded *(11)*	Basic (PMA Direct) xN (For N ≥ 18 channels located across four transceiver blocks)	*(12)*									ps
CMU PLL0 and CMU PLL1											
Supported data range	—	2488		11300	2488		10312.5	2488		8500	Mbps
CMU PLL lock time from `pll_powerdown` de-assertion	—	—	—	100	—	—	100	—	—	100	µs
ATX PLL (6G)											
Supported Data Range	/L = 1	4800-5400 and 6000-6500			4800-5400 and 6000-6500			—			Mbps
	/L = 2	2400-2700 and 3000-3250			2400-2700 and 3000-3250			—			Mbps
	/L = 4	1200-1350 and 1500-1625			1200-1350 and 1500-1625			—			Mbps
ATX PLL (10G)											
Supported Data Range	—	9900	—	11300	Min 9900 Max 10312.5			—			Mbps
Transceiver-FPGA Fabric Interface											
Interface speed (non-PMA Direct)	—	62.2	—	—	62.2	—	265.625	62.2	—	265.625	MHz
Interface speed (PMA Direct)	—	124.4	—	—	124.4	—	325	124.4	—	325	MHz

Table 1–23. Transceiver Specification for Stratix IV GT Devices (Part 7 of 7)

Symbol/ Description	Conditions	–1 Industrial Speed Grade			–2 Industrial Speed Grade			–3 Industrial Speed Grade			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Digital reset pulse width	—	—	—	—	Minimum is two parallel clock cycles						—

Notes to Table 1–23:

(1) The minimum `reconfig_clk` frequency is 2.5 MHz if the transceiver channel is configured in transmitter only mode. The minimum `reconfig_clk` frequency is 37.5 MHz if the transceiver channel is configured in receiver only or receiver and transmitter mode. For more information, refer to the *Stratix IV Dynamic Reconfiguration* chapter in volume 1 of the *Stratix IV Device Handbook*.

(2) The device cannot tolerate prolonged operation at this absolute maximum.

(3) You must use the 1.2-V RXV_{ICM} setting if the input serial data standard is LVDS.

(4) The rate matcher supports only up to ±300 ppm.

(5) Time taken to `rx_pll_locked` goes high from `rx_analogreset` de-assertion. Refer to Figure 1–2 on page 1–27.

(6) Time for which the CDR must be kept in lock-to-reference mode after `rx_pll_locked` goes high and before `rx_locktodata` is asserted in manual mode. Refer to Figure 1–2 on page 1–27.

(7) Time taken to recover valid data after the `rx_locktodata` signal is asserted in manual mode. Refer to Figure 1–2 on page 1–27.

(8) Time taken to recover valid data after the `rx_freqlocked` signal goes high in automatic mode. Refer to Figure 1–3 on page 1–27.

(9) A GPLL may be required to meet PMA-FPGA fabric interface timing above certain data rates. Refer to the "Left/Right PLL Requirements in Basic (PMA Direct) Mode" section in the *Stratix IV GX Transceiver Clocking* chapter in volume 2 of the *Stratix IV Device Handbook*.

(10) The Quartus II software automatically selects the appropriate slew rate depending on the configured data rate or functional mode.

(11) For applications that require low transmit lane-to-lane skew, use Basic (PMA Direct) xN to achieve PMA-Only bonding across all channels in the link. You can bond all channels on one side of the device by configuring them in Basic (PMA Direct) xN mode. For more information about clocking requirements in this mode, refer to the “Basic (PMA Direct) Mode Clocking” section in the *Stratix IV GX Transceiver Clocking* chapter in volume 2 of the *Stratix IV Device Handbook*.

(12) Pending Characterization.

Figure 1–2 shows the lock time parameters in manual mode.

LTD = Lock-To-Data; LTR = Lock-To-Reference

Figure 1–2. Lock Time Parameters for Manual Mode

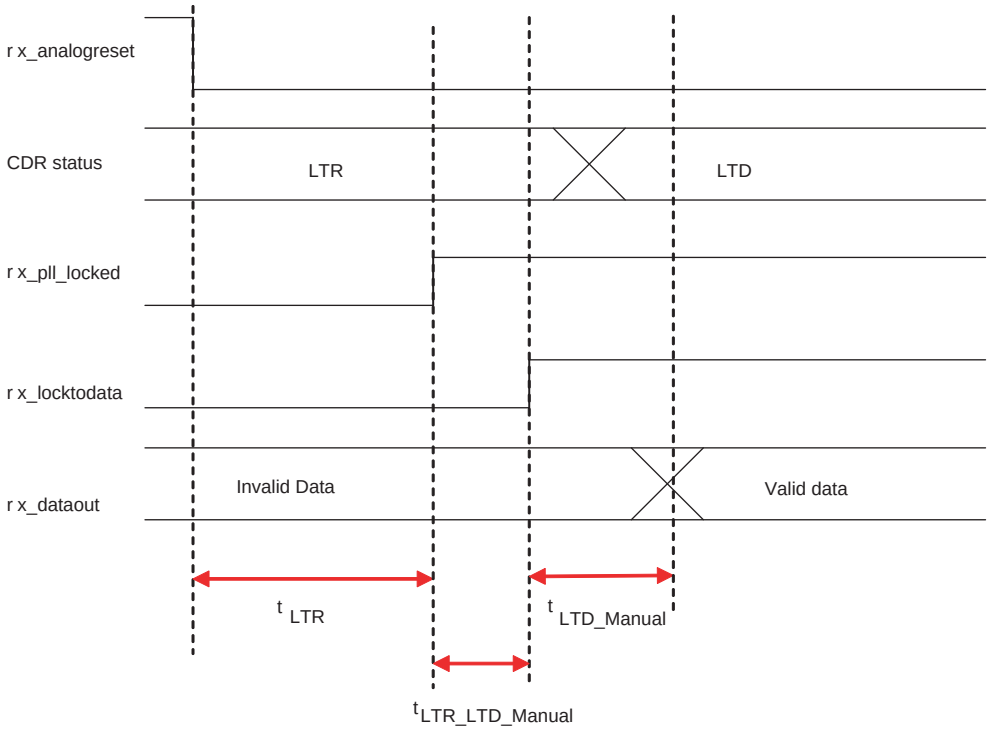


Figure 1–3 shows the lock time parameters in automatic mode.

Figure 1–3. Lock Time Parameters for Automatic Mode

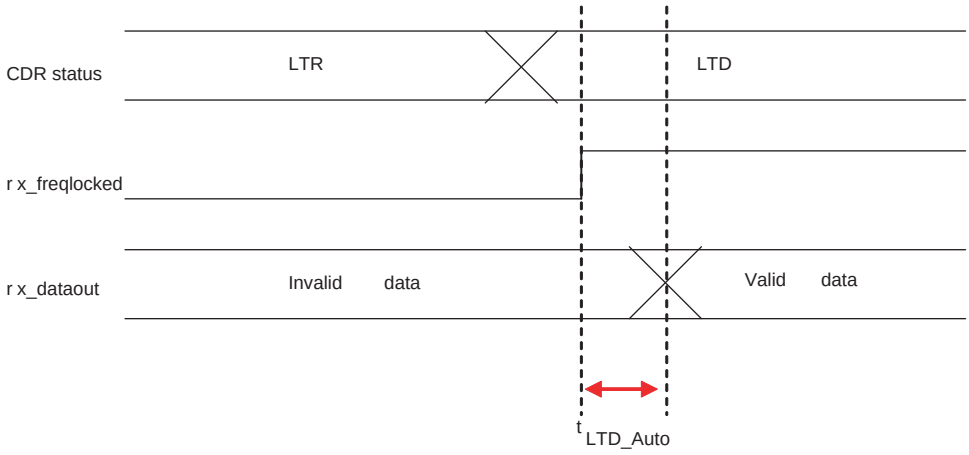


Table 1–24 through Table 1–27 show the typical differential V_{OD} termination settings for Stratix IV GX and GT devices.

Table 1–24. Typical V_{OD} Setting, TX Term = 85 Ω

Symbol	V_{OD} Setting (mV)							
	0	1	2	3	4	5	6	7
V_{OD} differential peak-peak Typical (mV)	170	340	510	595	680	765	850	1020

Table 1–25. Typical V_{OD} Setting, TX Term = 100 Ω

Symbol	V_{OD} Setting (mV)							
	0	1	2	3	4	5	6	7
V_{OD} differential peak-peak Typical (mV)	200	400	600	700	800	900	1000	1200

Table 1–26. Typical V_{OD} Setting, TX Term = 120 Ω

Symbol	V_{OD} Setting (mV)						
	0	1	2	3	4	5	6
V_{OD} differential peak-peak Typical (mV)	240	480	720	840	960	1080	1200

Table 1–27. Typical V_{OD} Setting, TX Term = 150 Ω

Symbol	V_{OD} Setting (mV)					
	0	1	2	3	4	5
V_{OD} differential peak-peak Typical (mV)	300	600	900	1050	1200	1350

Table 1–28 shows Stratix IV GX transceiver jitter specifications for all supported protocols. For protocols supported by Stratix IV GT –2 and –3 industrial speed grade devices, refer to the Stratix IV GX –2 commercial speed grade column in Table 1–28.

Table 1–28. Transceiver Block Jitter Specification for Stratix IV GX Devices *(Note 1)*, *(2)* (Part 1 of 7)

Symbol/ Description	Conditions	–2 Commercial Speed Grade			–3 Commercial/ Industrial and –2× Commercial Speed Grade			–4 Commercial/ Industrial Speed Grade			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
SONET/SDH Transmit Jitter Generation *(3)*											
Peak-to-peak jitter at 622.08 Mbps	Pattern = PRBS23	—	—	0.1	—	—	0.1	—	—	0.1	UI
RMS jitter at 622.08 Mbps	Pattern = PRBS23	—	—	0.01	—	—	0.01	—	—	0.01	UI
Peak-to-peak jitter at 2488.32 Mbps	Pattern = PRBS23	—	—	0.1	—	—	0.1	—	—	0.1	UI
RMS jitter at 2488.32 Mbps	Pattern = PRBS23	—	—	0.01	—	—	0.01	—	—	0.01	UI

Table 1–28. Transceiver Block Jitter Specification for Stratix IV GX Devices *(Note 1)*, *(2)* (Part 2 of 7)

Symbol/ Description	Conditions	–2 Commercial Speed Grade			–3 Commercial/ Industrial and –2× Commercial Speed Grade			–4 Commercial/ Industrial Speed Grade			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
SONET/SDH Receiver Jitter Tolerance *(3)*											
Jitter tolerance at 622.08 Mbps	Jitter frequency = 0.03 KHz Pattern = PRBS23	> 15			> 15			> 15			UI
	Jitter frequency = 25 KHZ Pattern = PRBS23	> 1.5			> 1.5			> 1.5			UI
	Jitter frequency = 250 KHz Pattern = PRBS23	> 0.15			> 0.15			> 0.15			UI
Jitter tolerance at 2488.32 Mbps	Jitter frequency = 0.06 KHz Pattern = PRBS23	> 15			> 15			> 15			UI
	Jitter frequency = 100 KHZ Pattern = PRBS23	> 1.5			> 1.5			> 1.5			UI
	Jitter frequency = 1 MHz Pattern = PRBS23	> 0.15			> 0.15			> 0.15			UI
	Jitter frequency = 10 MHz Pattern = PRBS23	> 0.15			> 0.15			> 0.15			UI
Fibre Channel Transmit Jitter Generation *(4)*, *(12)*											
Total jitter FC-1	Pattern = CRPAT	—	—	0.23	—	—	0.23	—	—	0.23	UI
Deterministic jitter FC-1	Pattern = CRPAT	—	—	0.11	—	—	0.11	—	—	0.11	UI
Total jitter FC-2	Pattern = CRPAT	—	—	0.33	—	—	0.33	—	—	0.33	UI
Deterministic jitter FC-2	Pattern = CRPAT	—	—	0.2	—	—	0.2	—	—	0.2	UI
Total jitter FC-4	Pattern = CRPAT	—	—	0.52	—	—	0.52	—	—	0.52	UI
Deterministic jitter FC-4	Pattern = CRPAT	—	—	0.33	—	—	0.33	—	—	0.33	UI
Fibre Channel Receiver Jitter Tolerance *(4)*, *(13)*											
Deterministic jitter FC-1	Pattern = CJTPAT	> 0.37			> 0.37			> 0.37			UI
Random jitter FC-1	Pattern = CJTPAT	> 0.31			> 0.31			> 0.31			UI
Sinusoidal jitter FC-1	Fc/25000	> 1.5			> 1.5			> 1.5			UI
	Fc/1667	> 0.1			> 0.1			> 0.1			UI
Deterministic jitter FC-2	Pattern = CJTPAT	> 0.33			> 0.33			> 0.33			UI
Random jitter FC-2	Pattern = CJTPAT	> 0.29			> 0.29			> 0.29			UI
Sinusoidal jitter FC-2	Fc/25000	> 1.5			> 1.5			> 1.5			UI
	Fc/1667	> 0.1			> 0.1			> 0.1			UI
Deterministic jitter FC-4	Pattern = CJTPAT	> 0.33			> 0.33			> 0.33			UI
Random jitter FC-4	Pattern = CJTPAT	> 0.29			> 0.29			> 0.29			UI
Sinusoidal jitter FC-4	Fc/25000	> 1.5			> 1.5			> 1.5			UI
	Fc/1667	> 0.1			> 0.1			> 0.1			UI

Table 1–28. Transceiver Block Jitter Specification for Stratix IV GX Devices *(Note 1)*, *(2)* (Part 3 of 7)

Symbol/ Description	Conditions	–2 Commercial Speed Grade			–3 Commercial/ Industrial and –2× Commercial Speed Grade			–4 Commercial/ Industrial Speed Grade			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
XAUI Transmit Jitter Generation *(5)*											
Total jitter at 3.125 Gbps	Pattern = CJPAT	—	—	0.3	—	—	0.3	—	—	0.3	UI
Deterministic jitter at 3.125 Gbps	Pattern = CJPAT	—	—	0.17	—	—	0.17	—	—	0.17	UI
XAUI Receiver Jitter Tolerance *(5)*											
Total jitter	—	> 0.65			> 0.65			> 0.65			UI
Deterministic jitter	—	> 0.37			> 0.37			> 0.37			UI
Peak-to-peak jitter	Jitter frequency = 22.1 KHz	> 8.5			> 8.5			> 8.5			UI
Peak-to-peak jitter	Jitter frequency = 1.875 MHz	> 0.1			> 0.1			> 0.1			UI
Peak-to-peak jitter	Jitter frequency = 20 MHz	> 0.1			> 0.1			> 0.1			UI
PCI Express Transmit Jitter Generation *(6)*											
Total jitter at 2.5 Gbps (Gen1)	Compliance pattern	—	—	0.25	—	—	0.25	—	—	0.25	UI
Total jitter at 5 Gbps (Gen2)	Compliance pattern	—	—	0.25	—	—	0.25	—	—	0.25	UI
PCI Express Receiver Jitter Tolerance *(6)*											
Total jitter at 2.5 Gbps (Gen1)	Compliance pattern	> 0.6			> 0.6			> 0.6			UI
Total jitter at 5 Gbps (Gen2)	Compliance pattern	Compliant			Compliant			Compliant			UI
Serial RapidIO Transmit Jitter Generation *(7)*											
Deterministic jitter (peak-to-peak)	Data Rate = 1.25, 2.5, 3.125 Gbps Pattern = CJPAT	—	—	0.17	—	—	0.17	—	—	0.17	UI
Total jitter (peak-to-peak)	Data Rate = 1.25, 2.5, 3.125 Gbps Pattern = CJPAT	—	—	0.35	—	—	0.35	—	—	0.35	UI
Serial RapidIO Receiver Jitter Tolerance *(7)*											
Deterministic jitter tolerance (peak-to-peak)	Data Rate = 1.25, 2.5, 3.125 Gbps Pattern = CJPAT	> 0.37			> 0.37			> 0.37			UI
Combined deterministic and random jitter tolerance (peak-to-peak)	Data Rate = 1.25, 2.5, 3.125 Gbps Pattern = CJPAT	> 0.55			> 0.55			> 0.55			UI

Table 1–28. Transceiver Block Jitter Specification for Stratix IV GX Devices *(Note 1)*, *(2)* (Part 4 of 7)

Symbol/ Description	Conditions	–2 Commercial Speed Grade			–3 Commercial/ Industrial and –2× Commercial Speed Grade			–4 Commercial/ Industrial Speed Grade			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Sinusoidal jitter tolerance (peak-to-peak)	Jitter Frequency = 22.1 KHz Data Rate = 1.25, 2.5, 3.125 Gbps Pattern = CJPAT	> 8.5			> 8.5			> 8.5			UI
	Jitter Frequency = 1.875 MHz Data Rate = 1.25, 2.5, 3.125 Gbps Pattern = CJPAT	> 0.1			> 0.1			> 0.1			UI
	Jitter Frequency = 20 MHz Data Rate = 1.25, 2.5, 3.125 Gbps Pattern = CJPAT	> 0.1			> 0.1			> 0.1			UI
GIGE Transmit Jitter Generation *(8)*											
Deterministic jitter (peak-to-peak)	Pattern = CRPAT	—	—	0.14	—	—	0.14	—	—	0.14	UI
Total jitter (peak-to-peak)	Pattern = CRPAT	—	—	0.279	—	—	0.279	—	—	0.279	UI
GIGE Receiver Jitter Tolerance *(8)*											
Deterministic jitter tolerance (peak-to-peak)	Pattern = CJPAT	> 0.4			> 0.4			> 0.4			UI
Combined deterministic and random jitter tolerance (peak-to-peak)	Pattern = CJPAT	> 0.66			> 0.66			> 0.66			UI
HiGig Transmit Jitter Generation *(9)*											
Deterministic jitter (peak-to-peak)	Data Rate = 3.75 Gbps Pattern = CJPAT	—	—	0.17	—	—	—	—	—	—	UI
Total jitter (peak-to-peak)	Data Rate = 3.75 Gbps Pattern = CJPAT	—	—	0.35	—	—	—	—	—	—	UI
HiGig Receiver Jitter Tolerance *(9)*											
Deterministic jitter tolerance (peak-to-peak)	Data Rate = 3.75 Gbps Pattern = CJPAT	> 0.37			—	—	—	—	—	—	UI
Combined deterministic and random jitter tolerance (peak-to-peak)	Data Rate = 3.75 Gbps Pattern = CJPAT	> 0.65			—	—	—	—	—	—	UI

Table 1–28. Transceiver Block Jitter Specification for Stratix IV GX Devices *(Note 1)*, *(2)* (Part 5 of 7)

Symbol/ Description	Conditions	–2 Commercial Speed Grade			–3 Commercial/ Industrial and –2× Commercial Speed Grade			–4 Commercial/ Industrial Speed Grade			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Sinusoidal jitter tolerance (peak-to-peak)	Jitter Frequency = 22.1 KHz Data Rate = 3.75 Gbps Pattern = CJPAT	> 8.5			—	—	—	—	—	—	UI
	Jitter Frequency = 1.875MHz Data Rate = 3.75 Gbps Pattern = CJPAT	> 0.1			—	—	—	—	—	—	UI
	Jitter Frequency = 20 MHz Data Rate = 3.75 Gbps Pattern = CJPAT	> 0.1			—	—	—	—	—	—	UI
(OIF) CEI Transmitter Jitter Generation *(10)*											
Total jitter (peak-to-peak)	Data Rate = 6.375 Gbps Pattern = PRBS15 BER = 10^{-12}	—	—	0.3	—	—	N/A	—	—	N/A	UI
(OIF) CEI Receiver Jitter Tolerance *(10)*											
Deterministic jitter tolerance (peak-to-peak)	Data Rate = 6.375 Gbps Pattern = PRBS31 BER = 10^{-12}	> 0.675			N/A	—	—	N/A	—	—	UI
Combined deterministic and random jitter tolerance (peak-to-peak)	Data Rate = 6.375 Gbps Pattern=PRBS31 BER = 10^{-12}	> 0.988			N/A	—	—	N/A	—	—	UI
Sinusoidal jitter tolerance (peak-to-peak)	Jitter Frequency = 38.2 KHz Data Rate = 6.375 Gbps Pattern = PRBS31 BER = 10^{-12}	> 5			N/A	—	—	N/A	—	—	UI
	Jitter Frequency = 3.82 MHz Data Rate = 6.375 Gbps Pattern = PRBS31 BER = 10^{-12}	> 0.05			N/A	—	—	N/A	—	—	UI
	Jitter Frequency = 20 MHz Data Rate= 6.375 Gbps Pattern = PRBS31 BER = 10^{-12}	> 0.05			N/A	—	—	N/A	—	—	UI

Table 1–28. Transceiver Block Jitter Specification for Stratix IV GX Devices *(Note 1)*, *(2)* (Part 6 of 7)

Symbol/ Description	Conditions	–2 Commercial Speed Grade			–3 Commercial/ Industrial and –2× Commercial Speed Grade			–4 Commercial/ Industrial Speed Grade			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
SDI Transmitter Jitter Generation *(11)*											
Alignment jitter (peak-to-peak)	Data Rate = 1.485 Gbps (HD) Pattern = Color Bar Low-Frequency Roll-Off = 100 KHz	0.2	—	—	0.2	—	—	0.2	—	—	UI
	Data Rate = 2.97 Gbps (3G) Pattern = Color Bar Low-Frequency Roll-Off = 100 KHz	0.3	—	—	0.3	—	—	0.3	—	—	UI
SDI Receiver Jitter Tolerance *(11)*											
Sinusoidal jitter tolerance (peak-to-peak)	Jitter Frequency = 15 KHz Data Rate = 2.97 Gbps (3G) Pattern = Single Line Scramble Color Bar	> 2			> 2			> 2			UI
	Jitter Frequency = 100 KHz Data Rate = 2.97 Gbps (3G) Pattern = Single Line Scramble Color Bar	> 0.3			> 0.3			> 0.3			UI
	Jitter Frequency = 148.5 MHz Data Rate = 2.97 Gbps (3G) Pattern = Single Line Scramble Color Bar	> 0.3			> 0.3			> 0.3			UI
Sinusoidal jitter tolerance (peak-to-peak)	Jitter Frequency = 20 KHz Data Rate = 1.485 Gbps (HD) Pattern = 75% Color Bar	> 1			> 1			> 1			UI
	Jitter Frequency = 100 KHz Data Rate = 1.485 Gbps (HD) Pattern = 75% Color Bar	> 0.2			> 0.2			> 0.2			UI
	Jitter Frequency = 148.5 MHz Data Rate = 1.485 Gbps (HD) Pattern =75% Color Bar	> 0.2			> 0.2			> 0.2			UI

Table 1–28. Transceiver Block Jitter Specification for Stratix IV GX Devices *(Note 1)*, *(2)* (Part 7 of 7)

Symbol/ Description	Conditions	–2 Commercial Speed Grade			–3 Commercial/ Industrial and –2× Commercial Speed Grade			–4 Commercial/ Industrial Speed Grade			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	

Notes to Table 1–28:

(1) Dedicated `refclk` pins were used to drive the input reference clocks.
(2) The Jitter numbers are valid for the stated conditions only.
(3) The jitter numbers for SONET/SDH are compliant to the GR-253-CORE Issue 3 Specification.
(4) The jitter numbers for Fibre Channel are compliant to the FC-PI-4 Specification revision 6.10.
(5) The jitter numbers for XAUI are compliant to the IEEE802.3ae-2002 Specification.
(6) The jitter numbers for PCI Express are compliant to the PCIe Base Specification 2.0.
(7) The jitter numbers for Serial RapidIO are compliant to the RapidIO Specification 1.3.
(8) The jitter numbers for GIGE are compliant to the IEEE802.3-2002 Specification.
(9) The jitter numbers for HiGig are compliant to the IEEE802.3ae-2002 Specification.
(10) The jitter numbers for (OIF) CEI are compliant to the OIF-CEI-02.0 Specification.
(11) The HD-SDI and 3G-SDI jitter numbers are compliant to the SMPTE292M and SMPTE424M Specifications.
(12) The fibre channel transmitter jitter generation numbers are compliant to the specification at δ_T interoperability point.
(13) The fibre channel receiver jitter tolerance numbers are compliant to the specification at δ_R interoperability point.

Transceiver Datapath PCS Latency

For more information about:

- Basic mode PCS latency, refer to Figure 1-90 through Figure 1-97 in the *Stratix IV Transceiver Architecture* chapter.
- PCI Express (PIPE) mode PCS latency, refer to Figure 1-102 in the *Stratix IV Transceiver Architecture* chapter.
- XAUI mode PCS latency, refer to Figure 1-119 in the *Stratix IV Transceiver Architecture* chapter.
- GIGE mode PCS latency, refer to Figure 1-128 in the *Stratix IV Transceiver Architecture* chapter.
- SONET/SDH mode PCS latency, refer to Figure 1-136 in the *Stratix IV Transceiver Architecture* chapter.
- SDI mode PCS latency, refer to Figure 1-141 in the *Stratix IV Transceiver Architecture* chapter.
- (OIF) CEI PHY mode PCS latency, refer to Figure 1-143 in the *Stratix IV Transceiver Architecture* chapter.

Core Performance Specifications

This sections describes the clock tree, phase-locked loop (PLL), digital signal processing (DSP), TriMatrix, configuration and JTAG specifications.

Clock Tree Specifications

Table 1–29 lists the clock tree specifications for Stratix IV devices.

Table 1–29. Clock Tree Performance for Stratix IV Devices—Preliminary

Performance			Unit
–2/–2× Speed Grade	–3 Speed Grade	–4 Speed Grade	
800	700	500	MHz

PLL Specifications

Table 1–30 describes the Stratix IV PLL specifications when operating in both the commercial junction temperature range (0° to 85°C) and the industrial junction temperature range (-40° to 100°C).

Table 1–30. PLL Specifications for Stratix IV Devices (Part 1 of 2)—Preliminary

Symbol	Parameter	Min	Typ	Max	Unit
f_{IN}	Input clock frequency (–2/–2x Speed Grade)	5	—	800 *(1)*	MHz
	Input clock frequency (–3 Speed Grade)	5	—	717 *(1)*	MHz
	Input clock frequency (–4 Speed Grade)	5	—	717 *(1)*	MHz
f_{INPFD}	Input frequency to the PFD	5	—	325	MHz
f_{VCO}	PLL VCO operating range (–2 Speed Grade)	600	—	1600	MHz
	PLL VCO operating range (–3 Speed Grade)	600	—	1300	MHz
	PLL VCO operating range (–4 Speed Grade)	600	—	1300	MHz
$t_{EINDUTY}$	Input clock or external feedback clock input duty cycle	40	—	60	%
f_{OUT}	Output frequency for internal global or regional clock (–2/–2x Speed Grade)	—	—	800 *(2)*	MHz
	Output frequency for internal global or regional clock (–3 Speed Grade)	—	—	717 *(2)*	MHz
	Output frequency for internal global or regional clock (–4 Speed Grade)	—	—	717 *(2)*	MHz
f_{OUT_EXT}	Output frequency for external clock input (–2 Speed Grade)	—	—	800 *(2)*	MHz
	Output frequency for external clock input (–3 Speed Grade)	—	—	717 *(2)*	MHz
	Output frequency for external clock input (–4 Speed Grade)	—	—	717 *(2)*	MHz
$t_{OUTDUTY}$	Duty cycle for external clock output (when set to 50%)	45	50	55	%
t_{FCOMP}	External feedback clock compensation time	—	—	10	ns
$t_{CONFIGPLL}$	Time required to reconfigure scan chain	—	3.5	—	scanclk cycles
$t_{CONFIGPHASE}$	Time required to reconfigure phase shift	—	1	—	scanclk cycles
$f_{SCANCLK}$	scanclk frequency	—	—	100	MHz
t_{LOCK}	Time required to lock from end-of-device configuration	—	—	1	ms

Table 1–30. PLL Specifications for Stratix IV Devices (Part 2 of 2)—Preliminary

Symbol	Parameter	Min	Typ	Max	Unit
t_{DLOCK}	Time required to lock dynamically (after switchover or reconfiguring any non-post-scale counters/delays)	—	—	1	ms
f_{CLBW}	PLL closed-loop low bandwidth	—	0.3	—	MHz
	PLL closed-loop medium bandwidth	—	1.5	—	MHz
	PLL closed-loop high bandwidth *(6)*	—	4	—	MHz
t_{PLL_PSERR}	Accuracy of PLL phase shift	—	—	±50	ps
t_{ARESET}	Minimum pulse width on the `areset` signal	10	—	—	ns
t_{INCCJ} *(3)*	Input clock cycle to cycle jitter ($F_{REF} \geq 100$ MHz)	—	—	0.15	UI (p-p)
	Input clock cycle to cycle jitter ($F_{REF} < 100$ MHz)	—	—	±750	ps (p-p)
t_{OUTPJ_DC} *(4)*	Period Jitter for dedicated clock output ($F_{OUT} \geq 100$ MHz)	—	—	175	ps (p-p)
	Period Jitter for dedicated clock output ($F_{OUT} < 100$ MHz)	—	—	17.5	mUI (p-p)
t_{OUTCCJ_DC} *(4)*	Cycle to Cycle Jitter for dedicated clock output ($F_{OUT} \geq 100$ MHz)	—	—	175	ps (p-p)
	Cycle to Cycle Jitter for dedicated clock output ($F_{OUT} < 100$ MHz)	—	—	17.5	mUI (p-p)
t_{OUTPJ_IO} *(4)*, *(7)*	Period Jitter for clock output on regular I/O ($F_{OUT} \geq 100$ MHz)	—	—	600	ps (p-p)
	Period Jitter for clock output on regular I/O ($F_{OUT} < 100$ MHz)	—	—	60	mUI (p-p)
t_{OUTCCJ_IO} *(4)*, *(7)*	Cycle to Cycle Jitter for clock output on regular I/O ($F_{OUT} \geq 100$ MHz)	—	—	600	ps (p-p)
	Cycle to Cycle Jitter for clock output on regular I/O ($F_{OUT} < 100$ MHz)	—	—	60	mUI (p-p)
$t_{CASC_OUTPJ_DC}$ *(4)*, *(5)*	Period Jitter for dedicated clock output in cascaded PLLs ($F_{OUT} \geq$100MHz)	—	—	250	ps (p-p)
	Period Jitter for dedicated clock output in cascaded PLLs ($F_{OUT} <$ 100MHz)	—	—	25	mUI (p-p)
f_{DRIFT}	Frequency drift after PFDENA is disabled for duration of 100 us	—	—	±10	%

Notes to Table 1–30:

(1) This specification is limited in the Quartus II software by the I/O maximum frequency. The maximum I/O frequency is different for each I/O standard.

(2) This specification is limited by the lower of the two: I/O F_{MAX} or F_{OUT} of the PLL.

(3) A high input jitter directly affects the PLL output jitter. To have low PLL output clock jitter, you need to provide a clean clock source, which is less than 120 ps.

(4) Peak-to-peak jitter with a probability level of 10^{-12} (14 sigma, 99.99999999974404 % confidence level). The output jitter specification applies to the intrinsic jitter of the PLL, when an input jitter of 30 ps is applied. The external memory interface clock output jitter specifications use a different measurement method and are available in **Table 1–45 on page 1–47**.

(5) The cascaded PLL specification is only applicable with the following condition:

a. Upstream PLL: 0.59Mhz ≤Upstream PLL BW < 1 MHz

b. Downstream PLL: Downstream PLL BW > 2 MHz

(6) High bandwidth PLL settings are not supported in external feedback mode.

(7) External memory interface clock output jitter specifications use a different measurement method, which is available in Table 1–43 on page 1–46.

DSP Block Specifications

Table 1–31 shows the Stratix IV DSP block performance specifications.

Table 1–31. Block Performance Specifications for Stratix IV DSP Devices *(Note 1)*—Preliminary

Mode	Resources Used	Performance			Unit
	Number of Multipliers	–2/–2× Speed Grade	–3 Speed Grade	–4 Speed Grade	
9×9-bit multiplier	1	490	405	375	MHz
12×12-bit multiplier	1	490	405	375	MHz
18×18-bit multiplier	1	600	550	490	MHz
36×36-bit multiplier	1	440	365	335	MHz
18×18-bit multiply accumulator	4	490	405	375	MHz
18×18-bit multiply adder	4	490	405	375	MHz
18×18-bit multiply adder-signed full precision	2	490	405	375	MHz
18×18-bit multiply adder with loopback *(2)*	2	390	320	300	MHz
36-bit shift (32-bit data)	1	440	365	335	MHz
Double mode	1	440	365	335	MHz

Notes to Table 1–31:

(1) Maximum is for fully pipelined block with **Round** and **Saturation** disabled.

(2) Maximum for loopback input registers disabled, **Round** and **Saturation** disabled, and pipeline and output registers enabled.

TriMatrix Memory Block Specifications

Table 1–32 lists the Stratix IV TriMatrix memory block specifications.

Table 1–32. TriMatrix Memory Block Performance Specification for Stratix IV Devices—Preliminary *(Note 1)* (Part 1 of 2)

Memory	Mode	Resources Used		Performance				
		ALUTs	TriMatrix Memory	–2 /–2× Commercial/ Industrial Speed Grade (MHz)	–3 Commercial/ Industrial Speed Grade (MHz)	–4 Commercial/ Industrial Speed Grade (MHz)	–3 Industrial Speed Grade (MHz) *(2)*	–4 Industrial Speed Grade (MHz) *(2)*
MLAB *(3)*	Single port 64×10	0	1	600	500	450	500	450
	Simple dual-port 32×20	0	1	600	500	450	500	450
	Simple dual-port 64×10	0	1	600	500	450	500	450
	ROM 64×10	0	1	600	500	450	500	450
	ROM 32×20	0	1	600	500	450	500	450
M9K Block *(3)*	Single-port 256×36	0	1	600	540	475	540	475
	Simple dual-port 256×36	0	1	560	490	420	490	420
	Simple dual-port 256×36, with the read-during-write option set to **Old Data**	0	1	380	340	300	340	300
	True dual port 512×18	0	1	500	440	380	440	380
	True dual-port 512×18, with the read-during-write option set to **Old Data**	0	1	380	340	300	340	300
	ROM 1 Port	0	1	600	540	475	540	475
	ROM 2 Port	0	1	600	540	475	540	475
	Min Pulse Width (clock high time)	—	—	750	800	850	800	850
	Min Pulse Width (clock low time)	—	—	500	625	690	625	690

Table 1–32. TriMatrix Memory Block Performance Specification for Stratix IV Devices—Preliminary *(Note 1)* (Part 2 of 2)

Memory	Mode	Resources Used		Performance				
		ALUTs	TriMatrix Memory	–2 /–2× Commercial/ Industrial Speed Grade (MHz)	–3 Commercial/ Industrial Speed Grade (MHz)	–4 Commercial/ Industrial Speed Grade (MHz)	–3 Industrial Speed Grade (MHz) *(2)*	–4 Industrial Speed Grade (MHz) *(2)*
M144K *(3)*	Single-port 2K×72	0	1	475	440	380	395	345
	Simple dual-port 2K×72	0	1	435	400	345	360	315
	Simple dual-port 2K×72, with the read-during-write option set to **Old Data**	0	1	260	240	205	225	195
	Simple dual-port 2K×64 (with ECC)	0	1	275	255	220	255	220
	True dual-port 4K×36	0	1	435	400	345	360	315
	True dual-port 4K×36, with the read-during-write option set to **Old Data**	0	1	260	240	205	225	195
	ROM 1 Port	0	1	540	500	435	440	390
	ROM 2 Port	0	1	500	465	400	415	360
	Min Pulse Width (clock high time)	—	—	700	755	860	860	950
	Min Pulse Width (clock low time)	—	—	500	625	690	690	690

Notes to Table 1–32:

(1) To achieve the maximum memory block performance, use a memory block clock that comes through global clock routing from an on-chip PLL set to 50% output duty cycle. Use the Quartus II software to report timing for this and other memory block clocking schemes.

(2) This is only applicable to the Stratix IV E and GX devices.

(3) When you use the error detection CRC feature, there is no degradation in F_{MAX}.

Configuration and JTAG Specifications

Table 1–33 lists the Stratix IV configuration mode specifications.

Table 1–33. Configuration Mode Specifications for Stratix IV Devices—Preliminary

Programming Mode	DCLK F_{MAX}	Unit
Passive serial	125	MHz
Fast passive parallel	125	MHz
Fast active serial	40	MHz
Remote update only in fast AS mode	10	MHz

Table 1–34 shows the JTAG timing parameters and values for Stratix IV devices.

Table 1–34. JTAG Timing Parameters and Values for Stratix IV Devices —Preliminary

Symbol	Description	Min	Max	Unit
t_{JCP}	TCK clock period	30	—	ns
t_{JCH}	TCK clock high time	14	—	ns
t_{JCL}	TCK clock low time	14	—	ns
$t_{JPSU\ (TDI)}$	TDI JTAG port setup time	1	—	ns
$t_{JPSU\ (TMS)}$	TMS JTAG port setup time	3	—	ns
t_{JPH}	JTAG port hold time	5	—	ns
t_{JPCO}	JTAG port clock to output	—	11 *(1)*	ns
t_{JPZX}	JTAG port high impedance to valid output	—	14 *(1)*	ns
t_{JPXZ}	JTAG port valid output to high impedance	—	14 *(1)*	ns

Note to Table 1–34:

(1) A 1 ns adder is required for each V_{CCIO} voltage step down from 3.0 V. For example, t_{JPCO} = 12 ns if V_{CCIO} of the TDO I/O bank = 2.5 V, or 13 ns if it equals 1.8 V.

Temperature Sensing Diode Specifications

Table 1–35 lists the specifications for the Stratix IV temperature sensing diode.

Table 1–35. External Temperature Sensing Diode Specification—Preliminary

Description	Min	Typ	Max	Unit
I_{bias}, diode source current	8	—	500	µA
$V_{bias,}$ voltage across diode	0.3	—	0.9	V
Series resistance	—	—	< 5	Ω
Diode ideality factor	—	—	1.030	—

Periphery Performance

This section describes periphery performance, including high-speed I/O and external memory interface.

I/O performance supports several system interfaces, such as the LVDS high-speed I/O interface, external memory interface, and the PCI/PCI-X bus interface. General-purpose I/O standards such as 3.3-, 2.5-, 1.8-, and 1.5-LVTTL/LVCMOS are capable of typical 167 MHz and 1.2 LVCMOS at 100 MHz interfacing frequency with 10pF load.

Actual achievable frequency depends on design- and system-specific factors. You need to perform HSPICE/IBIS simulations based on your specific design and system setup to determine the maximum achievable frequency in your system.

High-Speed I/O Specification

Table 1–36 shows high-speed I/O timing for Stratix IV devices.

Table 1–36. High-Speed I/O Specifications *(Note 1), (2),* (Part 1 of 2)—Preliminary

Symbol	Conditions	–2/–2× Speed Grade			–3			–4			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
f_{HSCLK_in} (input clock frequency) True Differential I/O Standards	Clock boost factor W = 1 to 40 *(3)*	5	—	800	5	—	717	5	—	717	MHz
f_{HSCLK_in} (input clock frequency) Single Ended I/O Standards *(10)*	Clock boost factor W = 1 to 40 *(3)*	5	—	800	5	—	717	5	—	717	MHz
f_{HSCLK_in} (input clock frequency) Single Ended I/O Standards *(11)*	Clock boost factor W = 1 to 40 *(3)*	5	—	520	5	—	420	5	—	420	MHz
f_{HSCLK_OUT} (output clock frequency)	—	5	—	800 *(7)*	5	—	717 *(7)*	5	—	717 *(7)*	MHz
Transmitter											
True Differential I/O Standards - f_{HSDR} (data rate)	SERDES factor J = 3 to 10 *(5)*	*(4)*	—	1600	*(4)*	—	1250	*(4)*	—	1250	Mbps
	SERDES factor J = 2, Uses DDR Registers	*(4)*	—	*(4)*	*(4)*	—	*(4)*	*(4)*	—	*(4)*	Mbps
	SERDES factor J = 1, Uses SDR Register	*(4)*	—	*(4)*	*(4)*	—	*(4)*	*(4)*	—	*(4)*	Mbps
Emulated Differential I/O Standards with Three External Output Resistor Networks - f_{HSDR} (data rate)	SERDES factor J =4 to 10	*(4)*	—	1100	*(4)*	—	1100	*(4)*	—	800	Mbps
Emulated Differential I/O Standards with One External Output Resistor - f_{HSDR} (data rate)		*(4)*	—	311	*(4)*	—	200	*(4)*	—	200	Mbps
$t_{x\ Jitter}$ *(5)*	Total Jitter for Data Rate, 600 Mbps - 1.6 Gbps	—	—	160	—	—	160	—	—	160	ps
	Total Jitter for Data Rate, < 600 Mbps	—	—	0.1	—	—	0.1	—	—	0.1	UI
t_{DUTY}	Tx output clock duty cycle for both True and Emulated Differential I/O Standards	45	50	55	45	50	55	45	50	55	%

Table 1–36. High-Speed I/O Specifications *(Note 1), (2),* (Part 2 of 2)—Preliminary

Symbol	Conditions	–2/–2× Speed Grade			–3			–4			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
t_{RISE} & t_{FALL}	True Differential I/O Standards	—	—	160	—	—	200	—	—	200	ps
	Emulated Differential I/O Standards with Three External Output Resistor Networks	—	—	310	—	—	350	—	—	350	ps
	Emulated Differential I/O Standards with One External Output Resistor	—	—	460	—	—	500	—	—	500	ps
TCCS	True Differential I/O Standards	—	—	100	—	—	100	—	—	100	ps
	Emulated Differential I/o Standards	—	—	250	—	—	250	—	—	250	ps
Receiver *(9)*											
$f_{HSDRDPA}$ (data rate) *(7)*	SERDES factor J = 3 to 10	150	—	1600	150	—	1250	150	—	1250	Mbps
f_{HSDR} (data rate)	SERDES factor J = 3 to 10	*(4)*		*(6)*	*(4)*		*(6)*	*(4)*		*(6)*	Mbps
DPA Mode											
DPA run length *(8)*	—	—	—	10000	—	—	10000	—	—	10000	UI
Soft CDR mode											
Soft-CDR PPM tolerance	—	—	—	300	—	—	300	—	—	300	± PPM
Non DPA Mode											
Sampling Window	All differential I/O standards	—	—	300	—	—	300	—	—	300	ps

Notes to Table 1–36:

(1) When J = 3 to 10, use the SERDES block.

(2) When J = 1 or 2, bypass the SERDES block.

(3) Clock Boost Factor (Ω) is the ratio between input data rate to the input clock rate.

(4) The minimum specification depends on the clock source (for example, PLL and clock pin) and the clock routing resource (global, regional, or local) that you use.

(5) The TX jitter specification is for the true differential I/O standards only.

(6) You can estimate the achievable maximum data rate for non-DPA mode by performing link timing closure analysis. You should consider the board skew margin, transmitter delay margin, and the receiver sampling margin to determine the maximum data rate supported.

(7) This is achieved by using the LVDS and DPA clock network.

(8) If the receiver with DPA enabled and transmitter are using shared PLLs, the minimum data rate is 150 Mbps.

(9) When J = 1 or 2, the receiver data rate is limited by the clock tree performance, as shown in Table 1–29 on page 1–35.

(10) This only applies to DPA and soft-CDR modes.

(11) This only applies to LVDS source synchronous mode (non-DPA).

Table 1–37 shows the DPA lock time specifications for Stratix IV ES devices.

Table 1–37. DPA Lock Time Specifications—Stratix IV ES Devices Only *(Note 1)*, *(2)*, *(3)*

Standard	Training Pattern	Number of Data Transitions in one repetition of training pattern	Number of repetitions per 256 data transitions *(4)*	Condition	Maximum
SPI-4	00000000001111111111	2	128	without DPA PLL calibration	256 data transitions
				with DPA PLL calibration	3x256 data transitions + 2x96 slow clock cycles *(5)*
Parallel Rapid I/O	00001111	2	128	without DPA PLL calibration	256 data transitions
				with DPA PLL calibration	3x256 data transitions + 2x96 slow clock cycles *(5)*
	10010000	4	64	without DPA PLL calibration	256 data transitions
				with DPA PLL calibration	3x256 data transitions + 2x96 slow clock cycles *(5)*
Miscellaneous	10101010	8	32	without DPA PLL calibration	256 data transitions
				with DPA PLL calibration	3x256 data transitions + 2x96 slow clock cycles *(5)*
	01010101	8	32	without DPA PLL calibration	256 data transitions
				with DPA PLL calibration	3x256 data transitions + 2x96 slow clock cycles *(5)*

Notes to Table 1–37:

(1) The DPA lock time is for one channel.
(2) One data transition is defined as a 0-to-1 or 1-to-0 transition.
(3) The DPA lock time applies to both commercial and industrial grade.
(4) This is the number of repetition for the stated training pattern to achieve 256 data transitions.
(5) Slow clock = Data rate (MHz)/Deserialization factor.

Figure 1–4 shows the DPA lock time specifications with DPA PLL calibration enabled.

Figure 1–4. DPA Lock Time Specification with DPA PLL Calibration Enabled

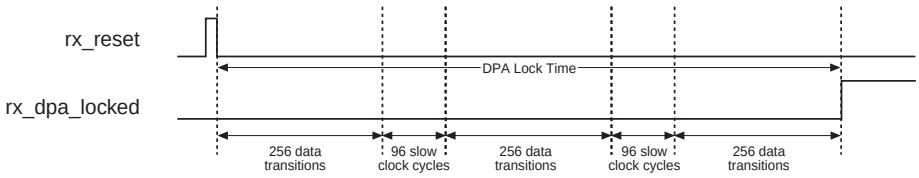


Table 1–38 lists the DPA lock time specifications for Stratix IV devices.

Table 1–38. DPA Lock Time Specifications—Stratix IV Non-ES Devices Only *(Note 1)*, *(2)*, *(3)*

Standard	Training Pattern	Number of Data Transitions in One Repetition of the Training Pattern	Number of Repetitions per 256 Data Transitions *(4)*	Maximum
SPI-4	00000000001111111111	2	128	640 data transitions
Parallel Rapid I/O	00001111	2	128	640 data transitions
	10010000	4	64	640 data transitions
Miscellaneous	10101010	8	32	640 data transitions
	01010101	8	32	640 data transitions

Notes to Table 1–38:

(1) The DPA lock time is for one channel.
(2) One data transition is defined as a 0-to-1 or 1-to-0 transition.
(3) The DPA lock time stated in the table applies to both commercial and industrial grade.
(4) This is the number of repetitions for the stated training pattern to achieve the 256 data transitions.
(5)

Figure 1–5 shows the LVDS soft-CDR/DPA sinusoidal jitter tolerance specification for a data rate equal to or higher than 1.25 Gbps. Table 1–39 lists this information in table form.

Figure 1–5. LVDS Soft-CDR/DPA Sinusoidal Jitter Tolerance Specification for a Data Rate Equal to or Higher Than 1.25 Gbps

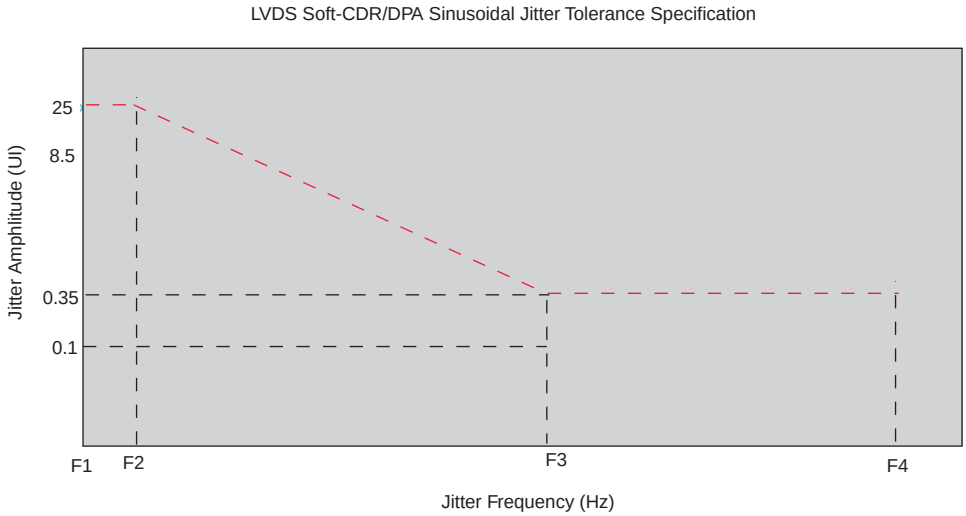


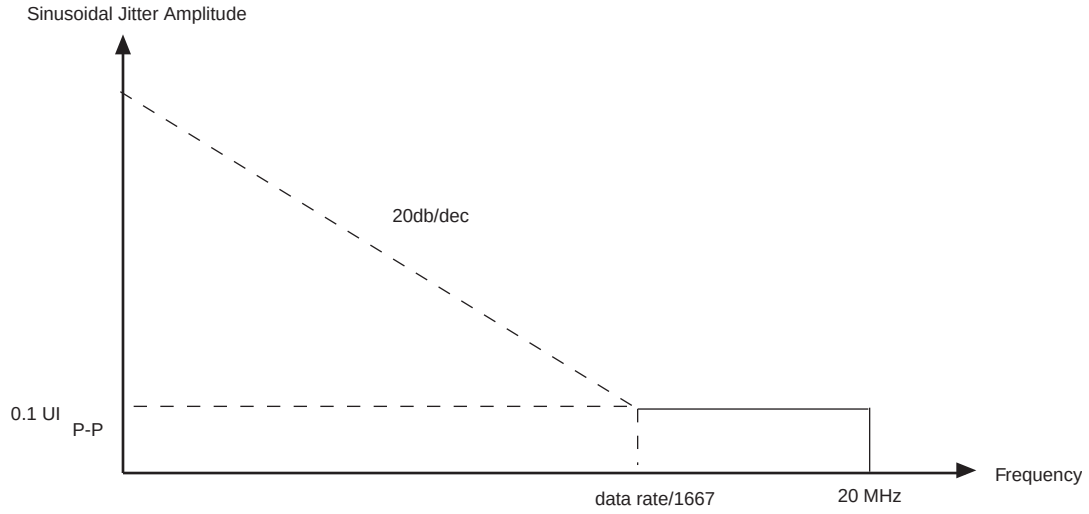
Table 1–39 lists the LVDS soft-CDR/DPA sinusoidal jitter tolerance specification for a data rate equal to or higher than 1.25 Gbps.

Table 1–39. LVDS Soft-CDR/DPA Sinusoidal Jitter Mask Values for a Data Rate Equal to or Higher than 1.25 Gbps

Jitter Frequency (Hz)		Sinusoidal Jitter (UI)
F1	10,000	25.000
F2	17,565	25.000
F3	1,493,000	0.350
F4	50,000,000	0.350

Figure 1–6 shows the LVDS soft-CDR/DPA sinusoidal jitter tolerance specification for a data rate less than 1.25 Gbps.

Figure 1–6. LVDS Soft-CDR/DPA Sinusoidal Jitter Tolerance Specification for a Data Rate Less than 1.25 Gbps



DLL and DQS Logic Block Specifications

Table 1–40 lists the DLL frequency range specifications for Stratix IV devices.

Table 1–40. DLL Frequency Range Specifications for Stratix IV Devices—Preliminary (Part 1 of 2)

Frequency Mode	Frequency Range (MHz)			Available Phase Shift	DQS Delay Buffer Mode *(1)*	Number of Delay Chains
	–2/–2× Speed Grade	–3 Speed Grade	–4 Speed Grade			
0	90-150	90-140	90-120	22.5°, 45°, 67.5°, 90°	Low	16
1	120-200	120-190	120-170	30°, 60°, 90°, 120°	Low	12
2	150-240	150-230	150-200	36°, 72°, 108°, 144°	Low	10
3	180-300	180-290	180-250	45°, 90°,135°, 180°	Low	8
4	240-370	240-350	240-310	30°, 60°, 90°, 120°	High	12
5	290-450	290-420	290-370	36°, 72°, 108°, 144°	High	10
6	360-560	360-530	360-460	45°, 90°, 135°, 180°	High	8

Table 1–40. DLL Frequency Range Specifications for Stratix IV Devices—Preliminary (Part 2 of 2)

Frequency Mode	Frequency Range (MHz)			Available Phase Shift	DQS Delay Buffer Mode *(1)*	Number of Delay Chains
	–2/–2× Speed Grade	–3 Speed Grade	–4 Speed Grade			
7	470-740	470-700	470-610	60°, 120°, 180°, 240°	High	6

Note to Table 1–40:

(1) Low indicates a 6-bit DQS delay setting; high indicates a 5-bit DQS delay setting.

Table 1–41 describes the DQS phase offset delay per stage for Stratix IV devices.

Table 1–41. DQS Phase Offset Delay Per Setting for Stratix IV Devices *(Note 1)*, *(2)*, *(3)*

Speed Grade	Min	Max	Unit
–2/–2×	7	13	ps
–3	7	15	ps
–4	7	16	ps

Notes to Table 1–41:

(1) The valid settings for phase offset are -64 to +63 for frequency modes 0 to 3 and -32 to +31 for frequency modes 4 to 6.

(2) The typical value equals the average of the minimum and maximum values.

(3) The delay settings are linear, with a cumulative delay variation of 40 ps for all speed grades. For example, when using a –2 speed grade and applying a 10 phase offset settings to a 90° phase shift at 400 MHz, the expected average cumulative delay is [625 ps + (10 × 10.5 ps) ± 20 ps] = 730 ps ± 20 ps.

Table 1–42 lists the DQS phase shift error for Stratix IV devices.

Table 1–42. DQS Phase Shift Error Specification for DLL-Delayed Clock (t_{DQS_PSERR}) for Stratix IV Devices *(Note 1)*

Number of DQS Delay Buffer	–2/–2X Speed Grade	–3 Speed Grade	–4 Speed Grade	Unit
1	26	28	30	ps
2	52	56	60	ps
3	78	84	90	ps
4	104	112	120	ps

Note to Table 1–42:

(1) This error specification is the absolute maximum and minimum error. For example, skew on three DQS delay buffers in a –2/–2x speed grade is ± 78 ps or ± 39 ps.

Table 1–43 lists the DQS phase shift error for Stratix IV devices.

Table 1–43. Memory Output Clock Jitter Specification for Stratix IV Devices *(Note 1)*, *(2)* (Part 1 of 2)

Parameter	Clock Network	Symbol	–2/–2X Speed Grade		–3 Speed Grade		–4 Speed Grade		Unit
			Min	Max	Min	Max	Min	Max	
Clock period jitter	Regional	$t_{JIT(per)}$	-75	75	-85	85	-100	100	ps
Cycle-to-cycle period jitter	Regional	$t_{JIT(cc)}$	-150	150	-170	170	-190	190	ps
Duty cycle jitter	Regional	$t_{JIT(duty)}$	-80	80	-90	90	-100	100	ps
Clock period jitter	Global	$t_{JIT(per)}$	-113	113	-128	128	-150	150	ps

Table 1–43. Memory Output Clock Jitter Specification for Stratix IV Devices *(Note 1), (2)* (Part 2 of 2)

Parameter	Clock Network	Symbol	–2/–2X Speed Grade		–3 Speed Grade		–4 Speed Grade		Unit
			Min	Max	Min	Max	Min	Max	
Cycle-to-cycle period jitter	Global	$t_{JIT(cc)}$	-225	225	-255	255	-285	285	ps
Duty cycle jitter	Global	$t_{JIT(duty)}$	-120	120	-135	135	-150	150	ps

Notes to Table 1–43:

(1) The memory output clock jitter measurements are for 200 consecutive clock cycles, as specified in the JEDEC DDR2/DDR3 SDRAM standard.

(2) The clock jitter specification applies to memory output clock pins generated using differential signal-splitter and DDIO circuits clocked by a PLL output routed on a regional or global clock network as specified. Altera recommends using regional clock networks whenever possible.

OCT Calibration Block Specifications

Table 1–44 lists the OCT calibration block specifications for Stratix IV devices.

Table 1–44. OCT Calibration Block Specifications

Symbol	Description	Min	Typ	Max	Unit
OCTUSRCLK	Clock required by OCT calibration blocks	—	—	20	MHz
T_{OCTCAL}	Number of OCTUSRCLK clock cycles required for OCT R_S/R_T calibration	—	1000	—	Cycles
$T_{OCTSHIFT}$	Number of OCTUSRCLK clock cycles required for OCT code to shift out	—	28	—	Cycles
T_{RS_RT}	Time required between the `dyn_term_ctrl` and `oe` signal transitions in a bidirectional I/O buffer to dynamically switch between OCT R_S and R_T	—	2.5	—	ns

Duty Cycle Distortion (DCD) Specifications

Table 1–45 lists the worst-case DCD for Stratix IV devices.

Table 1–45. Worst-Case DCD on Stratix IV I/O Pins

Symbol	–2/–2× Speed Grade		–3 Speed Grade		–4 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
Output Duty Cycle	45	55	45	55	45	55	%

I/O Timing

Altera offers two ways to determine I/O timing: the Excel-based I/O Timing and the Quartus II Timing Analyzer.

Excel-based I/O Timing provides pin timing performance for each device density and speed grade. The data is typically used prior to designing the FPGA to get an estimate of the timing budget as part of the link timing analysis. The Quartus II Timing Analyzer provides a more accurate and precise I/O timing data based on the specifics of the design after you complete place-and-route.

The Excel-based I/O Timing spreadsheet is downloadable from the Stratix IV Devices Literature webpage.

Programmable IOE Delay

Table 1–46 shows Stratix IV IOE programmable delay settings.

Table 1–46. IOE Programmable Delay for Stratix IV Devices

Parameter *(1)*	Available Settings	Min Offset *(2)*	Fast Model		Slow Model				
			Industrial	Commercial *(3)*	C2 *(3)*	C3	C4	I3	I4
D1	15	0	462	505	732	796	857	801	863
D2	7	0	214	232	337	368	397	370	405
D3	7	0	1700	1769	2695	2927	3157	2948	3178
D4	15	0	462	504	802	802	865	808	871
D5	15	0	472	500	746	811	875	817	868
D6	6	0	180	195	294	310	334	311	347

Notes to Table 1–46:

(1) You can set this value in the Quartus II software by selecting **D1**, **D2**, **D3**, **D4**, **D5**, and **D6** in the **Assignment Name** column.

(2) Minimum offset does not include the intrinsic delay.

(3) For the EP4SGX530 device density, the IOE programmable delays have an additional 5% maximum offset.

Programmable Output Buffer Delay

Table 1–47 lists the delay chain settings that control the rising and falling edge delays of the output buffer. The default delay is 0 ps.

Table 1–47. Programmable Output Buffer Delay *(Note 1)*

Symbol	Parameter	Typical	Unit
D_{OUTBUF}	Rising and/or falling edge delay	0 (default)	ps
		50	ps
		100	ps
		150	ps

Note to Table 1–47:

(1) You can set the programmable output buffer delay in the Quartus II software by setting the **Output Buffer Delay Control** assignment to either positive, negative, or both edges, with the specific values stated here (in ps) for the **Output Buffer Delay** assignment.

Glossary

Table 1–48 shows the glossary for this chapter.

Table 1–48. Glossary Table (Part 1 of 4)

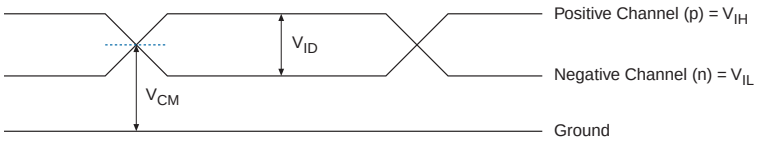
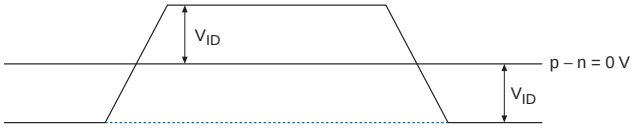
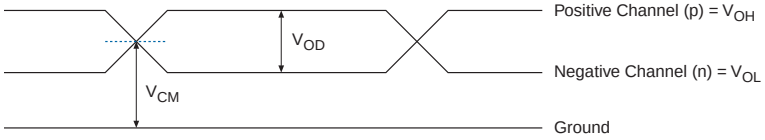
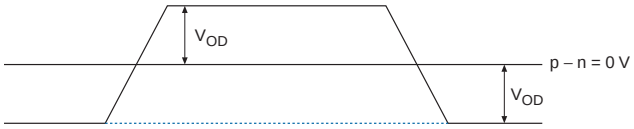
Letter	Subject	Definitions
A	—	—
B	—	—
C	—	—
D	Differential I/O Standards	*Receiver Input Waveforms* **Single-Ended Waveform**  **Differential Waveform**  *Transmitter Output Waveforms* **Single-Ended Waveform**  **Differential Waveform** 
E	—	—
F	f_{HSCLK}	Left/right PLL input clock frequency.
	f_{HSDR}	High-speed I/O block: Maximum/minimum LVDS data transfer rate (f_{HSDR} = 1/TUI), non-DPA.
	$f_{HSDRDPA}$	High-speed I/O block: Maximum/minimum LVDS data transfer rate ($f_{HSDRDPA}$ = 1/TUI), DPA.
G	—	—
H	—	—
I	—	—

Table 1–48. Glossary Table (Part 2 of 4)

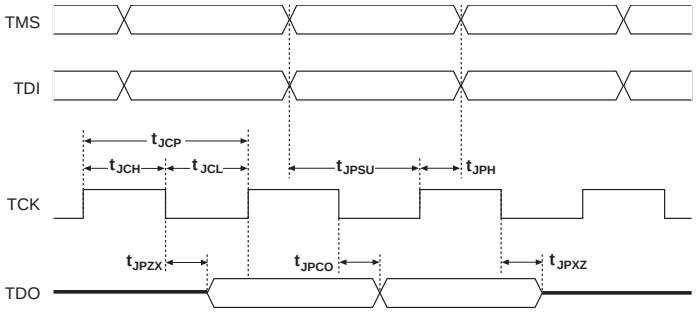
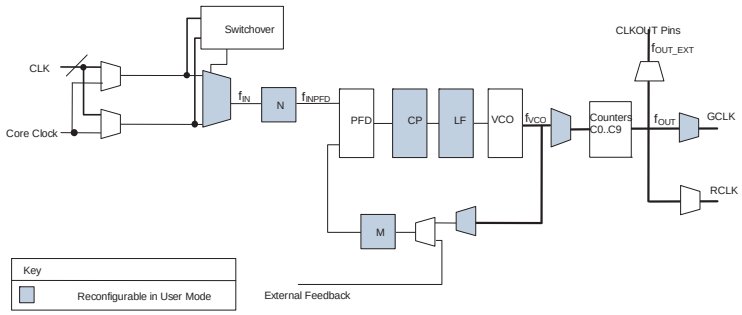
Letter	Subject	Definitions
J	J	High-speed I/O block: Deserialization factor (width of parallel data bus).
J	JTAG Timing Specifications	JTAG Timing Specifications: 
K	—	—
L	—	—
M	—	—
N	—	—
O	—	—
P	**PLL Specifications**	**Diagram of PLL Specifications** *(1)*  **Note:** (1) `Core Clock` can only be fed by dedicated clock input pins or PLL outputs.
Q	—	—
R	R_L	Receiver differential input discrete resistor (external to Stratix IV device).

Table 1–48. Glossary Table (Part 3 of 4)

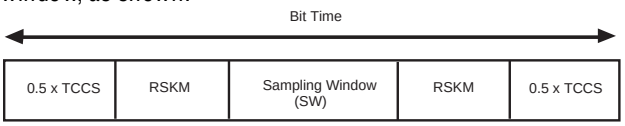
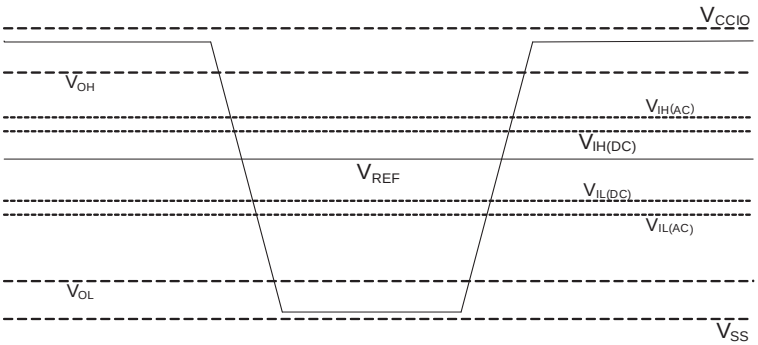
Letter	Subject	Definitions
S	SW (sampling window)	Timing Diagram—the period of time during which the data must be valid in order to capture it correctly. The setup and hold times determine the ideal strobe position within the sampling window, as shown: 
	Single-ended voltage referenced I/O standard	The JEDEC standard for SSTI and HSTL I/O defines both the AC and DC input signal values. The AC values indicate the voltage levels at which the receiver must meet its timing specifications. The DC values indicate the voltage levels at which the final logic state of the receiver is unambiguously defined. After the receiver input has crossed the AC value, the receiver changes to the new logic state. The new logic state is then maintained as long as the input stays beyond the AC threshold. This approach is intended to provide predictable receiver timing in the presence of input waveform ringing, as shown: *Single-Ended Voltage Referenced I/O Standard* 
T	t_C	High-speed receiver/transmitter input and output clock period.
	TCCS (channel-to-channel-skew)	The timing difference between the fastest and slowest output edges, including t_{CO} variation and clock skew, across channels driven by the same PLL. The clock is included in the TCCS measurement (refer to the *Timing Diagram* figure under **SW** in this table).
	t_{DUTY}	High-speed I/O block: Duty cycle on high-speed transmitter output clock. **Timing Unit Interval (TUI)** The timing budget allowed for skew, propagation delays, and data sampling window. (TUI = 1/(Receiver Input Clock Frequency Multiplication Factor) = t_C/w)
	t_{FALL}	Signal high-to-low transition time (80-20%)
	t_{INCCJ}	Cycle-to-cycle jitter tolerance on the PLL clock input
	t_{OUTPJ_IO}	Period jitter on the general purpose I/O driven by a PLL
	t_{OUTPJ_DC}	Period jitter on the dedicated clock output driven by a PLL
	t_{RISE}	Signal low-to-high transition time (20-80%)
U	—	—

Table 1–48. Glossary Table (Part 4 of 4)

Letter	Subject	Definitions
V	$V_{CM(DC)}$	DC Common mode input voltage.
	V_{ICM}	Input Common mode voltage—The common mode of the differential signal at the receiver.
	V_{ID}	Input differential voltage swing—The difference in voltage between the positive and complementary conductors of a differential transmission at the receiver.
	$V_{DIF(AC)}$	AC differential input voltage—Minimum AC input differential voltage required for switching.
	$V_{DIF(DC)}$	DC differential input voltage— Minimum DC input differential voltage required for switching.
	V_{IH}	Voltage input high—The minimum positive voltage applied to the input which is accepted by the device as a logic high.
	$V_{IH(AC)}$	High-level AC input voltage
	$V_{IH(DC)}$	High-level DC input voltage
	V_{IL}	Voltage input low—The maximum positive voltage applied to the input which is accepted by the device as a logic low.
	$V_{IL(AC)}$	Low-level AC input voltage
	$V_{IL(DC)}$	Low-level DC input voltage
	V_{OCM}	Output Common mode voltage—The common mode of the differential signal at the transmitter.
	V_{OD}	Output differential voltage swing—The difference in voltage between the positive and complementary conductors of a differential transmission at the transmitter.
	V_{SWING}	Differential input voltage
	V_X	Input differential cross point voltage
	V_{OX}	Output differential cross point voltage
W	W	High-speed I/O block: Clock Boost Factor
X	—	—
Y	—	—
Z	—	—

Document Revision History

Table 1–49 shows the revision history for this chapter.

Table 1–49. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
November 2009 v4.0	■ Added Table 1–9, Table 1–15, Table 1–38, and Table 1–39. ■ Added Figure 1–5 and Figure 1–6. ■ Added the "Transceiver Datapath PCS Latency" section. ■ Updated the "Electrical Characteristics", "Operating Conditions", and "I/O Timing" sections. ■ All tables updated except Table 1–16, Table 1–24, Table 1–25, Table 1–26, Table 1–27, Table 1–33, Table 1–34, and Table 1–45. ■ Updated Figure 1–2 and Figure 1–3. ■ Updated Equation 1–1. ■ Deleted Table 1-28, Table 1-29, Table 1-30, Table 1-42, Table 1-43, and Table 1-44. ■ Minor text edits.	—
June 2009 v3.1	■ Added "Preliminary Specifications" to the footer of each page. ■ Updated Table 1–1, Table 1–2, Table 1–7, Table 1–10, Table 1–11, Table 1–12, Table 1–21, Table 1–22, Table 1–23, Table 1–25, Table 1–37, Table 1–38, Table 1–39, Table 1–40, and Table 1–44. ■ Minor text edits.	—
March 2009 v3.0	■ Replaced Table 1–31 and Table 1–37. ■ Updated Table 1–1, Table 1–2, Table 1–5, Table 1–19, Table 1–41, Table 1–44, Table 1–45, Table 1–49, and Table 1–51. ■ Added Table 1–21, Table 1–46, and Table 1–47 ■ Added Figure 1–3. ■ Removed "Timing Model", "Preliminary and Final Timing", "I/O Timing Measurement Methodology", "I/O Default Capacitive Loading", and "Referenced Documents" sections.	—
December 2009 v2.1	Minor changes.	—
November 2008 v2.0	■ Minor text edits. ■ Updated Table 1–19, Table 1–32, Table 1–34 - Table 1–39.	Minor text edits.
August 2008 v1.1	■ Updated Table 1–1, Table 1–2, Table 1–4, Table 1–5, and Table 1–26. ■ Removed figures from "Transceiver Performance Specifications" on page 1–10 that are repeated in the glossary.	Minor text edits and an additional note to Table 1–26.
May 2008 v1.0	Initial release.	—