



Stratix IV Device Handbook

Volume 2: Transceivers



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

SIV5V2-4.7

© 2015 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. 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. 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.



Chapter Revision Dates	ix
-------------------------------------	----

Section I. Transceiver Architecture

Chapter 1. Transceiver Architecture in Stratix IV Devices

Overview	1-1
Transceiver Channel Locations	1-4
Stratix IV GX Device Offerings	1-4
Stratix IV GT Device Offerings	1-8
Transceiver Block Architecture	1-16
Transceiver Channel Architecture	1-17
Transmitter Channel Datapath	1-19
Receiver Channel Datapath	1-40
CMU Channel Architecture	1-100
Configuring CMU Channels for Clock Generation	1-101
Configuring CMU Channels as Transceiver Channels	1-106
Other CMU Channel Features	1-109
Dynamic Reconfiguration of the CMU Channel Analog Controls	1-110
Functional Modes	1-110
Basic Functional Mode	1-111
Deterministic Latency Mode	1-122
PCIe Mode	1-127
XAUI Mode	1-153
GIGE Mode	1-164
SONET/SDH Mode	1-172
SDI Mode	1-178
(OIF) CEI PHY Interface Mode	1-181
Serial RapidIO Mode	1-182
Basic (PMA Direct) Functional Mode	1-187
Loopback Modes	1-190
Serial Loopback	1-190
Parallel Loopback	1-191
Reverse Serial Loopback	1-193
Reverse Serial Pre-CDR Loopback	1-194
PCIe Reverse Parallel Loopback	1-194
Auxiliary Transmit (ATX) PLL Block	1-195
6G ATX PLL Block	1-195
10G ATX PLL Block	1-196
Input Reference Clocks for the ATX PLL Block	1-198
Architecture of the ATX PLL Block	1-199
ATX Clock Divider	1-200
The Differences Between 10G ATX PLL, 6G ATX PLL, and CMU PLL	1-200
Calibration Blocks	1-201
Calibration Block Location	1-201
Calibration	1-205
Input Signals to the Calibration Block	1-205
Built-In Self Test Modes	1-207
BIST Mode Pattern Generators and Verifiers	1-207

PRBS in Single-Width Mode	1-208
PRBS in Double-Width Mode	1-209
Transceiver Port Lists	1-210
Reference Information	1-225

Chapter 2. Transceiver Clocking in Stratix IV Devices

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 a 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	2-21
Transmitter Channel Datapath Clocking Resources	2-22
Transmitter Channel Clocking Configurations	2-23
Non-Bonded Channel Configurations	2-24
Bonded Channel Configurations	2-27
Non-Bonded Basic (PMA Direct) Mode Channel Configurations	2-34
Bonded Basic (PMA Direct) $\times N$ Mode Channel Configurations	2-36
Receiver Channel Datapath Clocking	2-39
Non-Bonded Channel Configurations	2-39
Bonded Channel Configurations	2-43
Basic (PMA Direct) Mode Channel Configurations	2-49
FPGA Fabric-Transceiver Interface Clocking	2-51
FPGA Fabric-Transmitter Interface Clocking	2-52
Quartus II-Selected Transmitter Phase Compensation FIFO Write Clock	2-52
User-Selected Transmitter Phase Compensation FIFO Write Clock	2-58
FPGA Fabric-Receiver Interface Clocking	2-61
Quartus II Software-Selected Receiver Phase Compensation FIFO Read Clock	2-62
User-Selected Receiver Phase Compensation FIFO Read Clock	2-68
Using the CMU/ATX PLL for Clocking User Logic in the FPGA Fabric	2-71
Configuration Examples	2-72
Configuration Example 1: Configuring 24 Channels in Basic (PMA Direct) $\times N$ Mode in the EP4S100G5F45 Device	2-73
Configuration Example 2: Configuring Sixteen Identical Channels Across Four Transceiver Blocks	2-75
Configuration Example 3: Configuring Sixteen Channels Across Four Transceiver Blocks	2-76
Configuration Example 4: Configuring Left and Right, Left, or Right PLL in VCO Bypass Mode	2-78

Chapter 3. Configuring Multiple Protocols and Data Rates in Stratix IV Devices

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	3-9
Sharing ATX PLLs	3-10
Combining Receiver Only Channels	3-10
Combining Transmitter Channel and Receiver Channel Instances	3-11
Multiple Transmitter Channel and Receiver Channel Instances	3-11
Example 3	3-11
Combining Transceiver Instances in Multiple Transceiver Blocks	3-13
Example 4	3-14
Combining Transceiver Instances Using PLL Cascade Clocks	3-16
Combining Channels Configured in Protocol Functional Modes	3-17
Combining Channels in Bonded Functional Modes	3-17
Bonded $\times 4$ Functional Mode	3-17
Bonded $\times 8$ Functional Mode	3-20
Combining Channels Configured in Deterministic Latency Mode	3-24
Combining Channels Using the PCIe hard IP Block with Other Channels	3-24
Combining Transceiver Channels in Basic (PMA Direct) Configurations	3-25
Combining Multiple Channels Configured in Basic (PMA Direct) $\times 1$ Configurations	3-26
Multiple Basic (PMA Direct) $\times 1$ Configuration Instances with One Channel per Instance	3-26
One Instance in a Basic (PMA Direct) $\times 1$ Configuration with Multiple Transceiver Channels	3-26
Combining Multiple Instances of Transmitter Only and Receiver Only Configurations in Basic (PMA Direct) $\times 1$ Mode	3-29
Combining Channels Configured in Basic (PMA Direct) $\times 1$ with Non-Basic (PMA Direct) Modes	3-29
Basic (PMA Direct) $\times N$ Configurations	3-33
Channel Placement in a Basic (PMA Direct) $\times N$ Mode Instance	3-33
Examples of Combining Multiple Instances of Basic (PMA Direct) $\times N$ Modes	3-35
Combination Requirements When You Enable Channel Reconfiguration	3-42
Combination Requirements When You Enable the Use Alternate CMU PLL Option	3-42
Example 12	3-43
Combination Requirements When You Use Multiple TX PLLs	3-44
Example 13	3-45
Combining Transceiver Channels When You Enable the Adaptive Equalization (AEQ) Feature	3-47
Example 14	3-48
Combination Requirements for Stratix IV Devices	3-49
Placement Rules for Transceiver Channels at 9.9 Gbps to 11.3 Gbps	3-49
Summary	3-49

Chapter 4. Reset Control and Power Down in Stratix IV Devices

User Reset and Power-Down Signals	4-2
-----------------------------------	-----

Blocks Affected by the Reset and Power-Down Signals	4-3
Transceiver Reset Sequences	4-4
All Supported Functional Modes Except PCIe Functional Mode	4-6
Bonded Channel Configuration	4-6
Non-Bonded Channel Configuration	4-15
PCIe Functional Mode	4-22
PCIe Reset Sequence	4-22
PCIe Initialization/Compliance Phase	4-23
PCIe Normal Phase	4-23
PMA Direct Drive Mode Reset Sequences	4-24
Basic (PMA Direct) Drive $\times N$ Mode	4-25
Transmitter Only Channel with No PLL_L/R	4-25
Transmitter Only Channel with a PLL_L/R	4-26
Basic (PMA Direct) Drive $\times 1$ Mode	4-31
Receiver and Transmitter Channel Set-Up—Receiver CDR in Automatic Lock Mode	4-32
Receiver and Transmitter Channel Set-Up—Receiver CDR in Manual Lock Mode	4-34
Dynamic Reconfiguration Reset Sequences	4-36
Reset Sequence when Using Dynamic Reconfiguration with the ‘data rate division in TX’ Option	4-36
Reset Sequence when Using Dynamic Reconfiguration with the ‘Channel and TX PLL select/reconfig’ Option	4-37
Power Down	4-38
Simulation Requirements	4-39
Reference Information	4-39

Chapter 5. Dynamic Reconfiguration in Stratix IV Devices

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)	5-20
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	5-67
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	5-70
Connections Between the ALTGX and ALTGX_RECONFIG Instances	5-70

Controlling the EyeQ Hardware	5-71
Adaptive Equalization (AEQ)	5-75
Adaptive Equalization Limitations	5-75
Enabling the AEQ Control Logic and AEQ Hardware	5-75
Connections Between the ALTGX and ALTGX_RECONFIG Instances	5-76
One Time Mode for a Single Channel	5-78
Dynamic Reconfiguration Controller Port List	5-78
Error Indication During Dynamic Reconfiguration	5-90
Dynamic Reconfiguration Duration	5-91
PMA Controls Reconfiguration Duration	5-91
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
Example 2	5-100

Additional Information

How to Contact Altera	Info-1
Typographic Conventions	Info-1

The chapters in this document, *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. Transceiver Architecture in Stratix IV Devices
Revised: *September 2015*
Part Number: *SIV52001-4.7*

- Chapter 2. Transceiver Clocking in Stratix IV Devices
Revised: *September 2012*
Part Number: *SIV52002-3.4*

- Chapter 3. Configuring Multiple Protocols and Data Rates in Stratix IV Devices
Revised: *September 2012*
Part Number: *SIV52003-4.2*

- Chapter 4. Reset Control and Power Down in Stratix IV Devices
Revised: *January 2014*
Part Number: *SIV52004-4.4*

- Chapter 5. Dynamic Reconfiguration in Stratix IV Devices
Revised: *January 2014*
Part Number: *SIV52005-3.6*



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, Transceiver Architecture in Stratix IV Devices](#)
- [Chapter 2, Transceiver Clocking in Stratix IV Devices](#)
- [Chapter 3, Configuring Multiple Protocols and Data Rates in Stratix IV Devices](#)
- [Chapter 4, Reset Control and Power Down in Stratix IV Devices](#)
- [Chapter 5, Dynamic Reconfiguration 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.

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.

-  For information about upcoming Stratix IV device features, refer to the *Upcoming Stratix IV Device Features* document.
-  For information about changes to the currently published *Stratix IV Device Handbook*, refer to the *Addendum to the Stratix IV Device Handbook* chapter.

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](#) lists 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® (PCIe)	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 600 Mbps 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 protocols shown in [Table 1-2](#).

Table 1-2. Serial Protocols Supported by the Stratix IV GT Transceiver Channels

Protocol	Description
PCIe	Gen 1 at 2.5 Gbps and Gen 2 at 5.0 Gbps
XAUI	3.125 Gbps up to HiGig at 3.75 Gbps
GIGE	1.25 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 (600 Mbps to 3.75 Gbps) and Basic double-width (600 Mbps to 11.3 Gbps) functional modes.



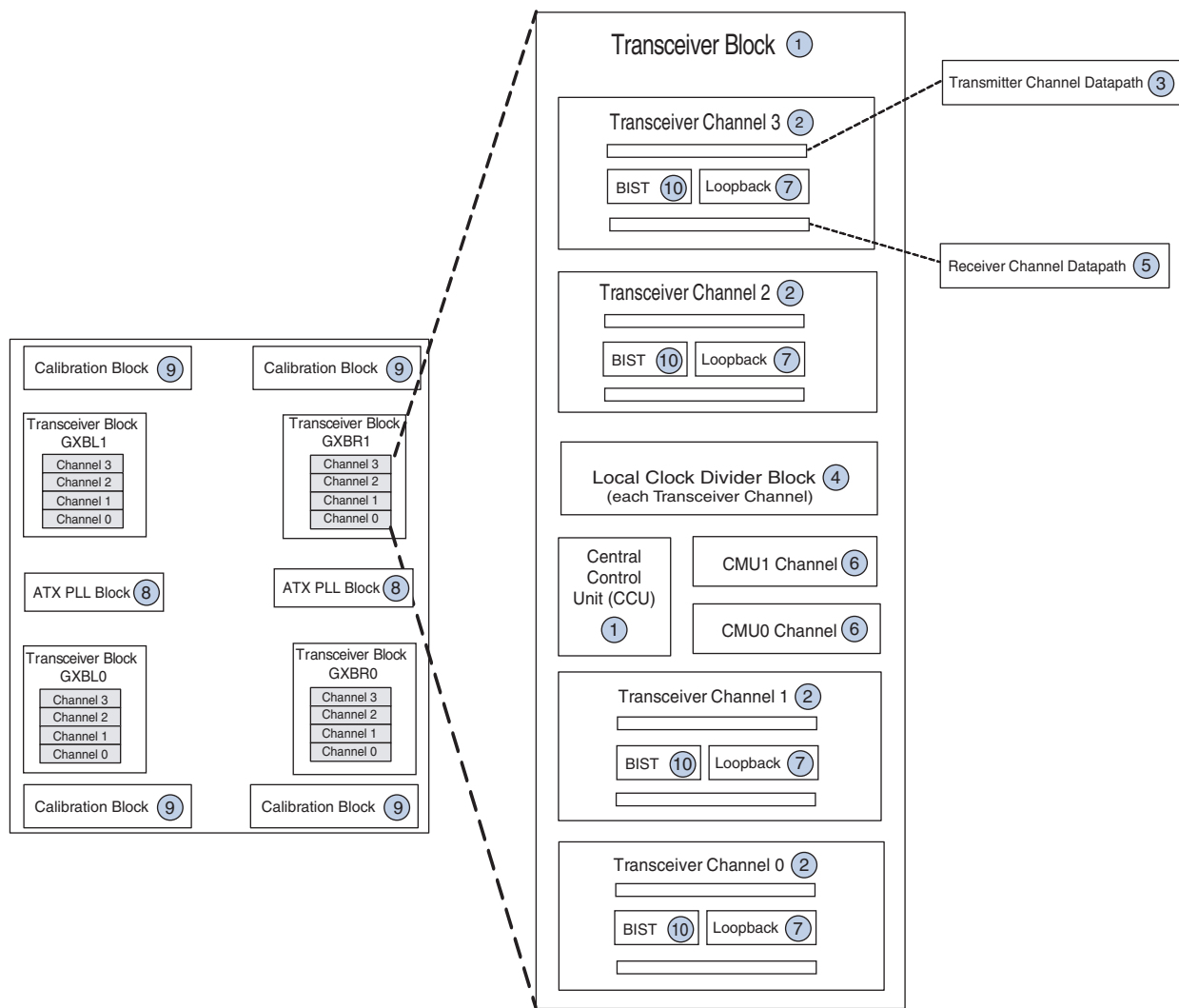
-  Stratix IV GX and GT devices have PCIe hard IP, PCS, and PMA blocks. For more information, refer to the [PCI Express Compiler User Guide](#).
-  For more information about Stratix IV GX and GT protocols, refer to the [Configuring Multiple Protocols and Data Rates in Stratix IV Devices](#) chapter.

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 1-16
2. [“Transceiver Channel Architecture”](#) on page 1-17
3. [“Transmitter Channel Datapath”](#) on page 1-19
4. [“Transmitter Local Clock Divider Block”](#) on page 1-39
5. [“Receiver Channel Datapath”](#) on page 1-40
6. [“CMU Channel Architecture”](#) on page 1-100
7. [“Loopback Modes”](#) on page 1-190
8. [“Auxiliary Transmit \(ATX\) PLL Block”](#) on page 1-195

9. “Calibration Blocks” on page 1–201
10. “Built-In Self Test Modes” on page 1–207

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 lists the total number of transceiver channels and transceiver block locations in each Stratix IV GX device member structured into full-duplex four- and six-channel groups called transceiver blocks.

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: <ul style="list-style-type: none"> ■ 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: <ul style="list-style-type: none"> ■ Right side—GXBR0 and GXBR1 ■ Left side—GXBL0 and GXBL1 Refer to Figure 1–2 on page 1–5.
EP4SGX180HF35 EP4SGX230HF35 EP4SGX290HF35 EP4SGX360HF35	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: <ul style="list-style-type: none"> ■ 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 EP4SGX290KF43 EP4SGX360KF40 EP4SGX360KF43 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: <ul style="list-style-type: none"> ■ Right side—GXBR0, GXBR1, and GXBR2 ■ Left side—GXBL0, GXBL1, and GXBL2 Refer to Figure 1-3 on page 1-6 .
EP4SGX290NF45 EP4SGX360NF45 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: <ul style="list-style-type: none"> ■ Right side—GXBR0, GXBR1, GXBR2, and GXBR3 ■ Left side—GXBL0, GXBL1, GXBL2, and GXBL3 Refer to Figure 1-4 on page 1-7 .

Figure 1-2 shows transceiver channel, PLL, and PCIe hard IP block locations in each Stratix IV GX device that has 8 or 16 transceiver channels.

Figure 1-2. Transceiver Channel, PLL, and PCIe Hard IP Block Locations with 8 and 16 Stratix IV GX Transceiver Channels

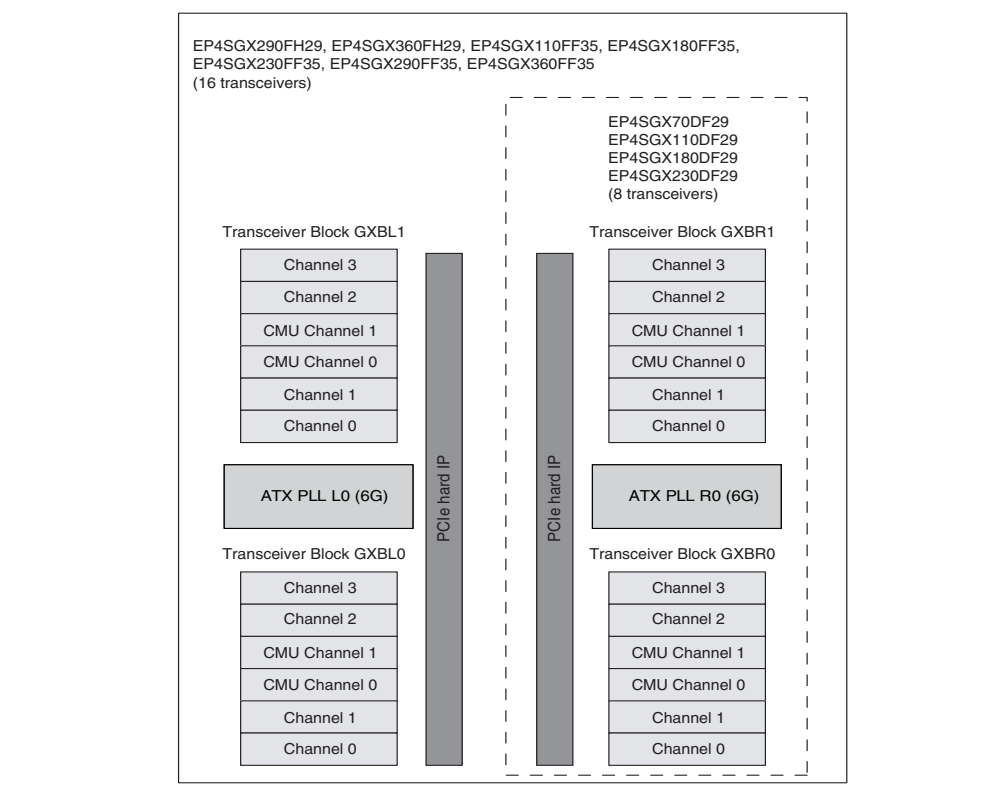
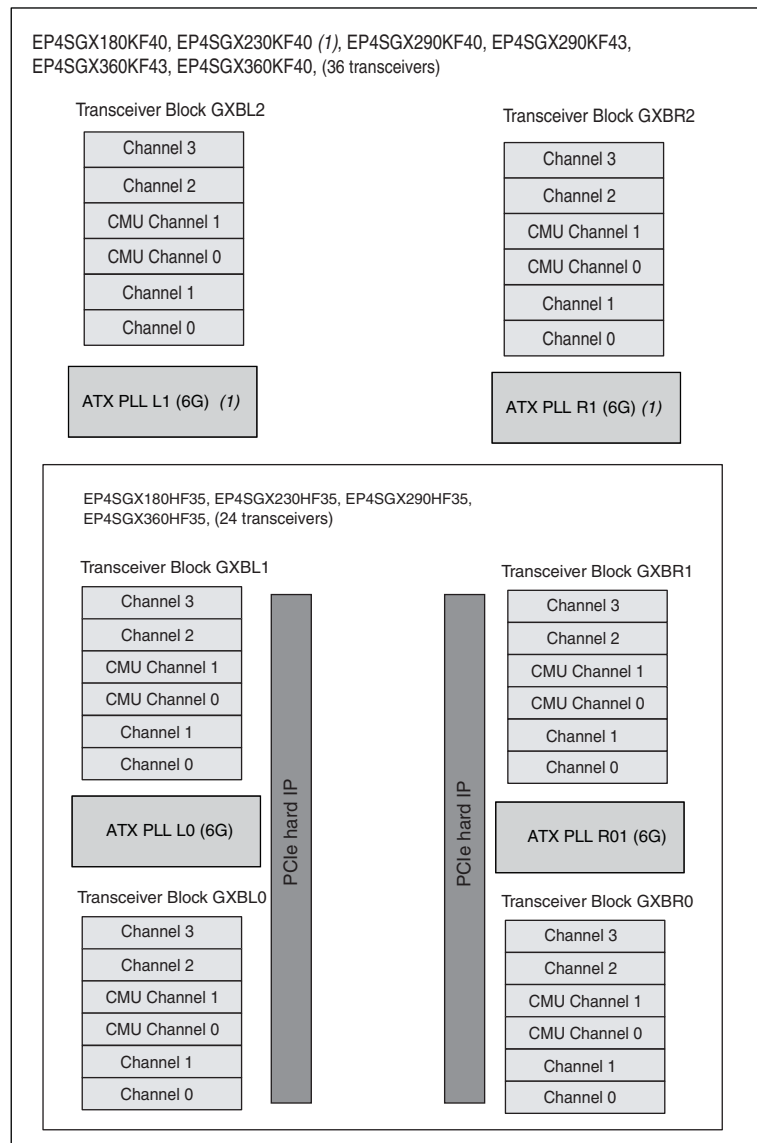


Figure 1-3 shows transceiver channel, PLL, and PCIe hard IP block locations in each Stratix IV GX device that has 24 or 36 transceiver channels (except for the EP4SGX530 device).

Figure 1-3. Transceiver Channel, PLL, and PCIe Hard IP Block Locations with 24 or 36 Stratix IV GX Transceiver Channels (Except the EP4SGX530 Device)

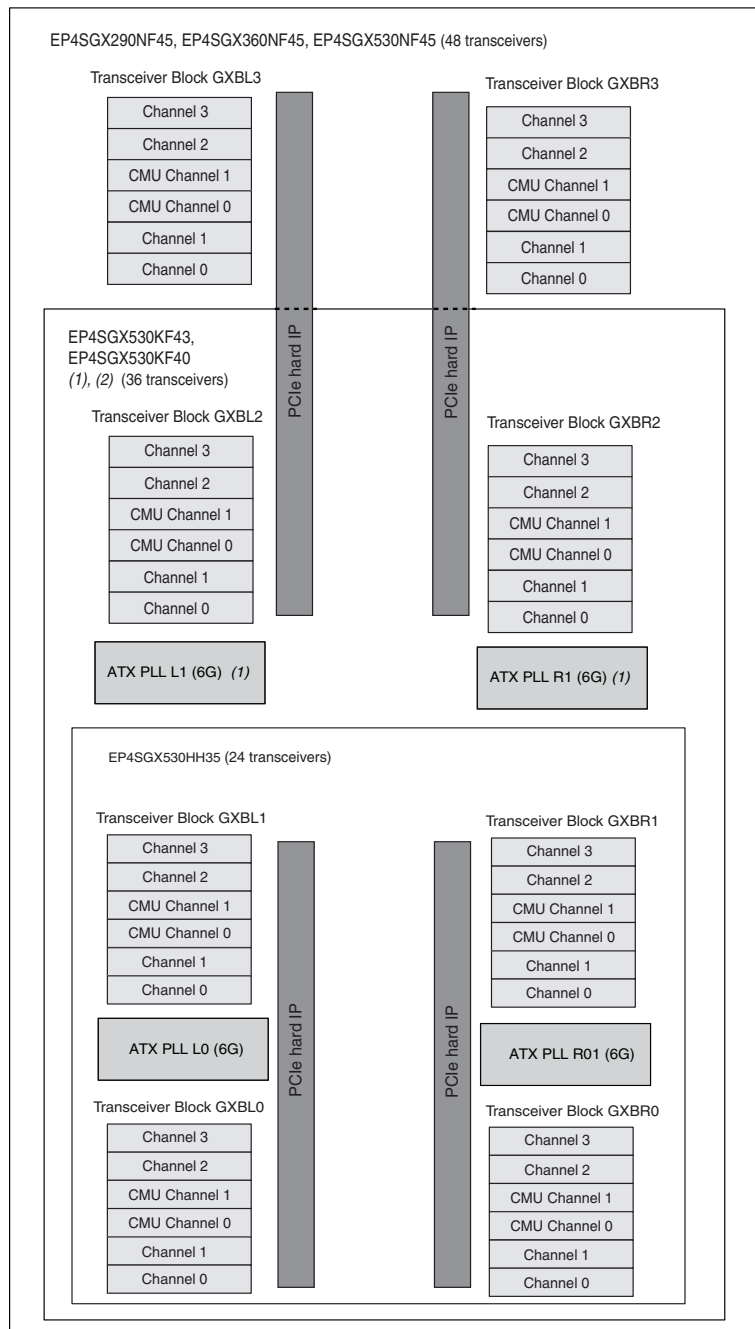


Note to Figure 1-3:

(1) The 6G ATX PLL R1 and L1 blocks are not available in the EP4SGX230KF40 devices.

Figure 1-4 shows transceiver channel, PLL, and PCIe hard IP block locations in each EP4SGX530 Stratix IV GX device that has 24 and 36 transceiver channels and the remaining Stratix IV GX devices that have 48 transceiver channels.

Figure 1-4. Transceiver Channel, PLL, and PCIe Hard IP Block Locations with 48 Stratix IV GX Transceiver Channels and EP4SGX530 with 24, 36, and 48 Stratix IV GX Transceiver Channels



Notes to Figure 1-4:

- (1) The 6G ATX PLL R1 and L1 blocks are not available in the EP4SGX530KF40 devices.
- (2) Only the EP4SGX530 device in the six transceiver block package has four PCI hard IP blocks.

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 600 Mbps and 11.3 Gbps.
- (2) 8G transceiver channels support data rates between 600 Mbps 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 600 Mbps and 6.5 Gbps are PMA-only channels that do not have PCS circuitry. For more information, refer to “CMU Channel Architecture” on page 1-100.
- (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)

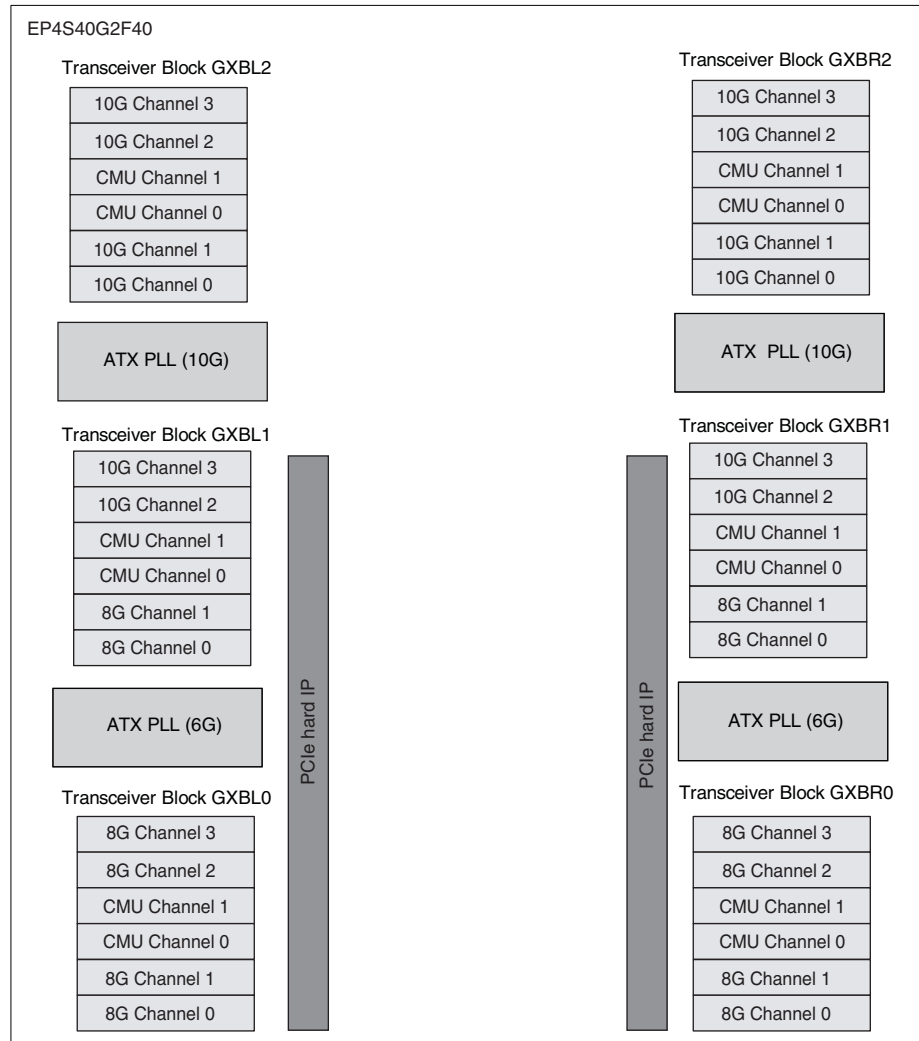
Device Member	Total number of Transceiver Channels	Transceiver Channel Location
EP4S40G2F40	36	12 regular transceiver channels (six 10G and six 8G), located in three transceiver blocks: Left Side—4 in GXBL2, 2 in GXBL1 Right Side—4 in GXBR2, 2 in GXBR1 Refer to Figure 1-5 on page 1-10.
EP4S40G5H40	36	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 1-6 on page 1-11.

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 1-7 on page 1-12 .
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 1-8 on page 1-13 .
EP4S100G3F45 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 (8G) Right Side—4 in GXBR3, 4 in GXBR2, 4 in GXBR1, 4 in GXBR0 (8G) Refer to Figure 1-9 on page 1-14 .
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 1-10 on page 1-15 .

Figure 1-5 shows the transceiver channel, PLL, and PCIe hard IP block locations for the EP4S40G2F40 Stratix IV GT devices.

Figure 1-5. Transceiver Channel, PLL, and PCIe Hard IP Block Locations in EP4S40G2F40 Stratix IV GT Devices ⁽¹⁾, ⁽²⁾

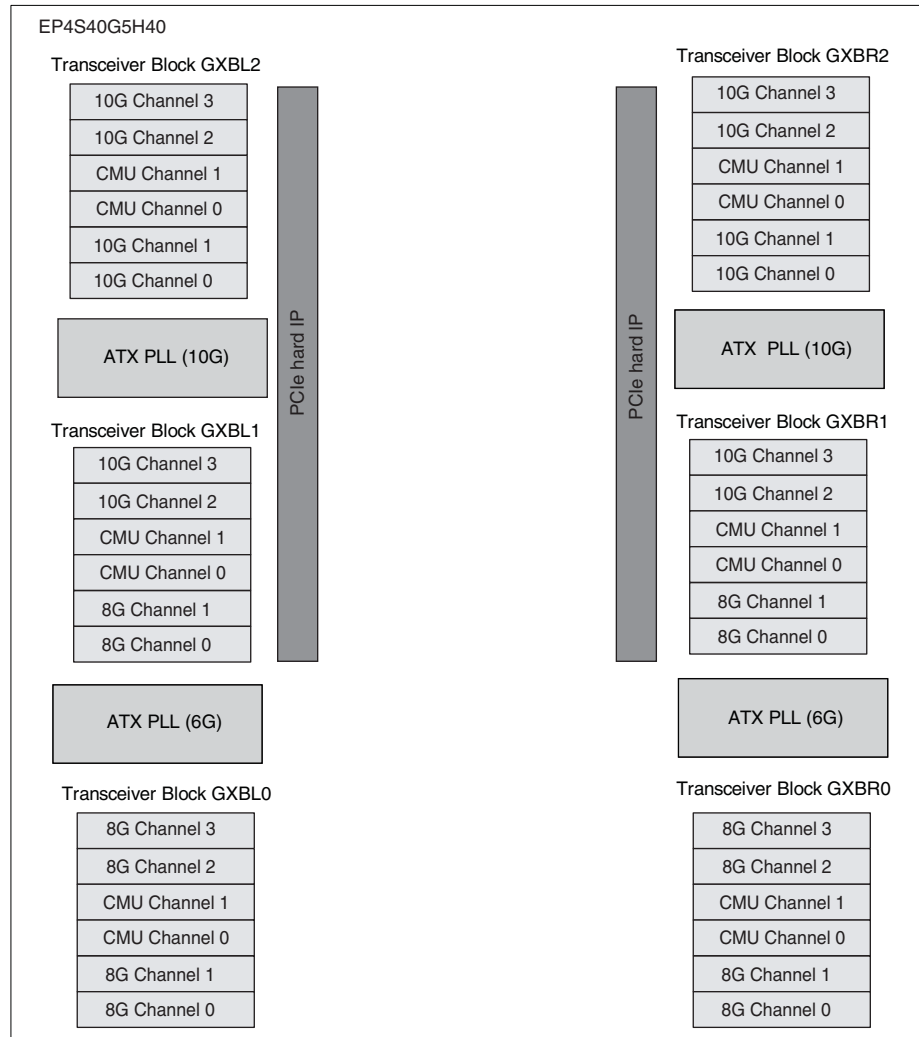


Notes to Figure 1-5:

- (1) EP4S40G2F40ES1 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.
- (2) If you are using the PCIe hard IP block, the EP4S40G2F40 device is not able to migrate to the EP4S40G5H40 device.

Figure 1-6 shows the transceiver channel, PLL, and PCIe hard IP block locations for the EP4S40G5H40 Stratix IV GT devices.

Figure 1-6. Transceiver Channel, PLL, and PCIe Hard IP Block Locations in EP4S40G5H40 Stratix IV GT Devices ⁽¹⁾

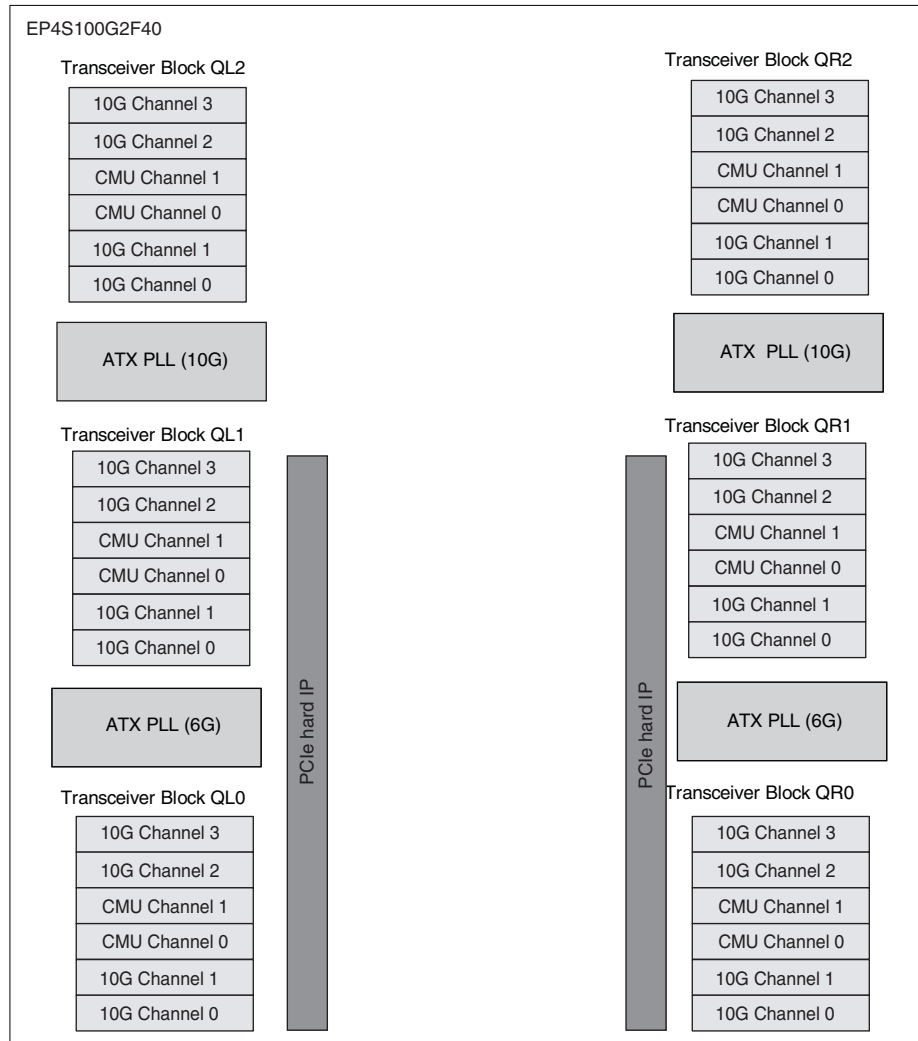


Note to Figure 1-6:

(1) If you are using the PCIe hard IP block, the EP4S40G2F40 device is not able to migrate to the EP4S40G5H40 device.

Figure 1-7 shows the transceiver channel, PLL, and PCIe hard IP block locations in EP4S100G2F40 Stratix IV GT devices.

Figure 1-7. Transceiver Channel, PLL, and PCIe Hard IP Block Locations in EP4S100G2F40 Stratix IV GT Devices

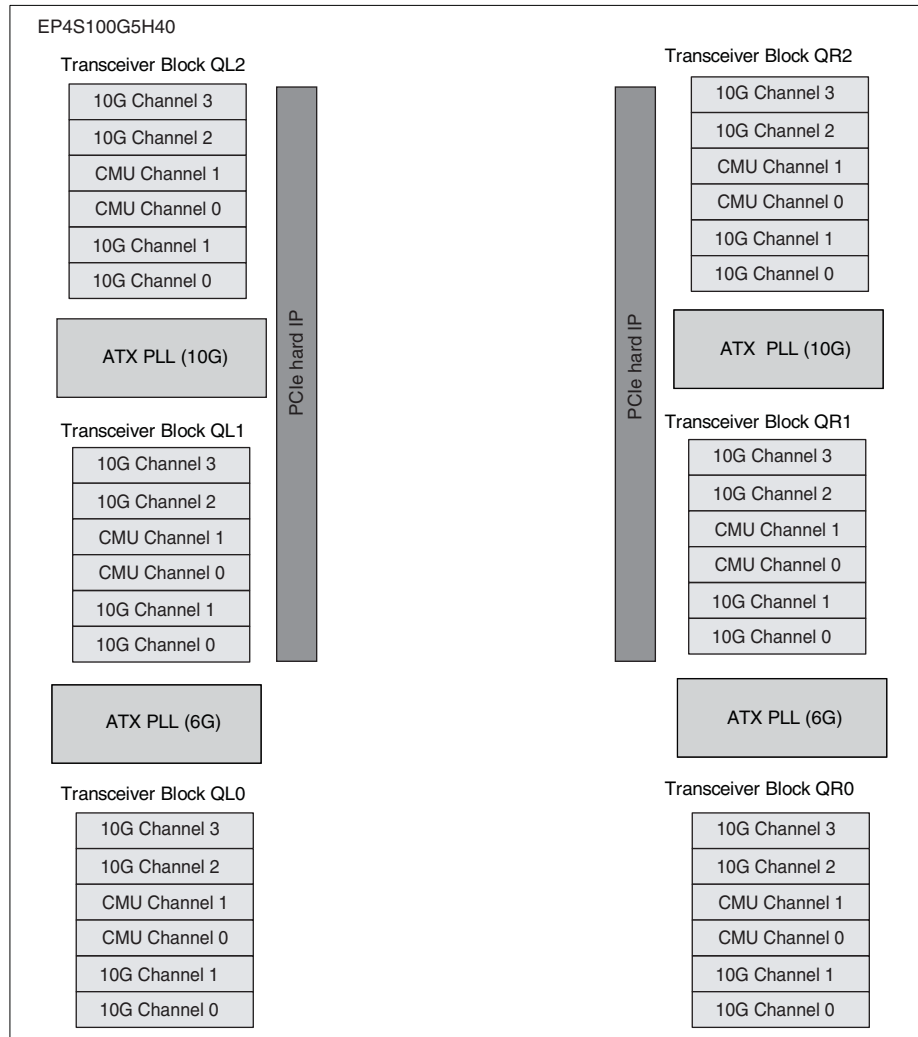


Notes to Figure 1-7:

- (1) EP4S100G2F40ES1 devices do not have 10G ATX PLL blocks. Use the CMU PLL to generate transceiver clocks for channels configured at 11.3 Gbps.
- (2) If you are using the PCIe hard IP block, the EP4S100G2F40 device is not able to migrate to the EP4S100G5H40 device.

Figure 1-8 shows the transceiver channel, PLL, and PCIe hard IP block locations for the EP4S100G5H40 Stratix IV GT devices.

Figure 1-8. Transceiver Channel, PLL, and Hard IP Block Locations in EP4S100G5H40 Stratix IV GT Devices

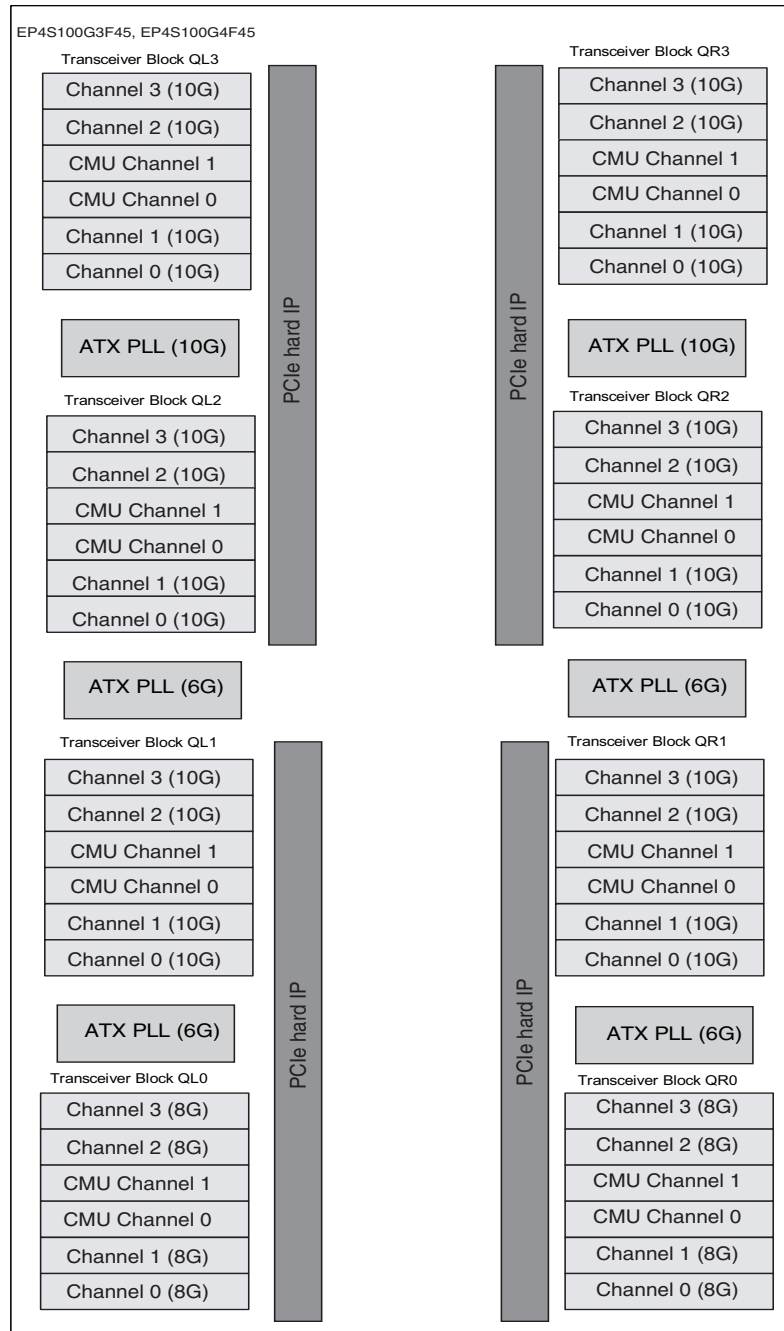


Note to Figure 1-8:

- (1) If you are using the PCIe hard IP block, the EP4S100G2F40 device is not able to migrate to the EP4S100G5H40 device.

Figure 1-9 shows the transceiver channel, PLL, and PCIe hard IP block locations for the EP4S100G3F45 and EP4S100G4F45 Stratix IV GT devices.

Figure 1-9. Transceiver Channel, PLL, and PCIe Hard IP Block Locations in EP4S100G3F45 and EP4S100G4F45 Stratix IV GT Devices ⁽¹⁾

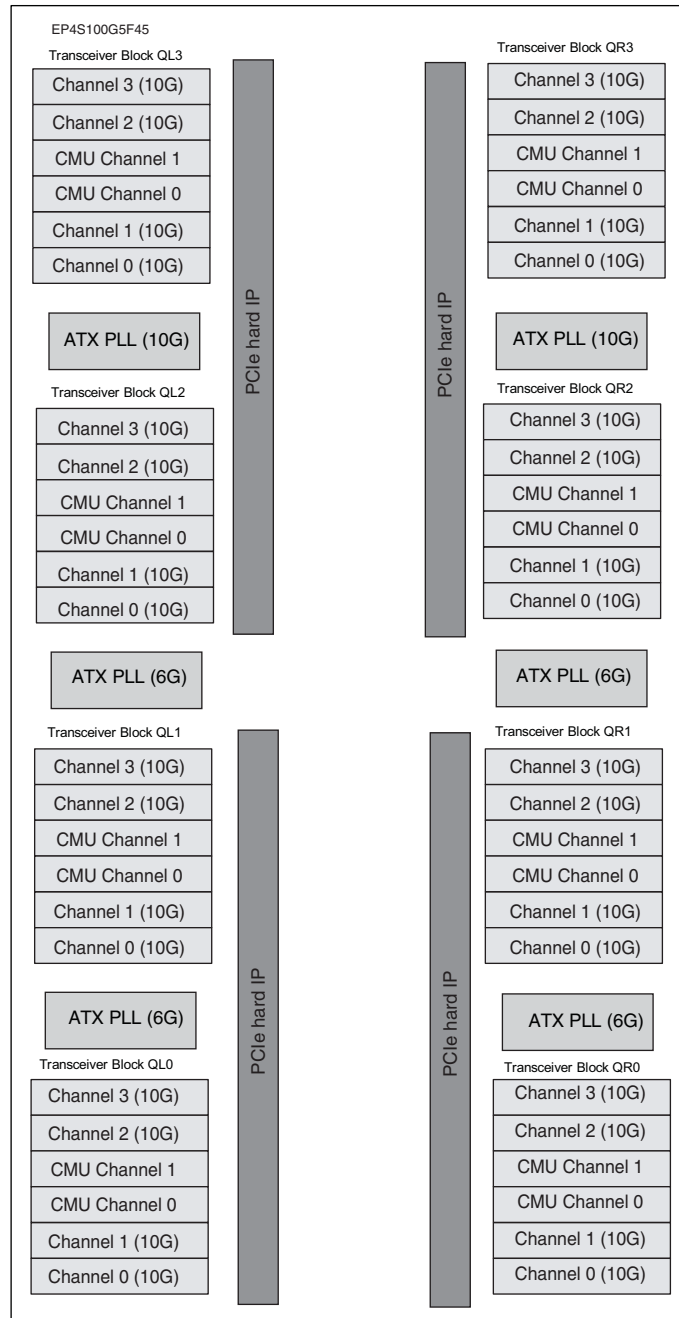


Note to Figure 1-9:

- (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.

Figure 1-10 shows the transceiver channel, PLL, and PCIe hard IP block locations for the EP4S100G5F45 Stratix IV GT devices.

Figure 1-10. Transceiver Channel, PLL, and PCIe Hard IP Block Locations in EP4S100G5F45 Stratix IV GT Devices ⁽¹⁾



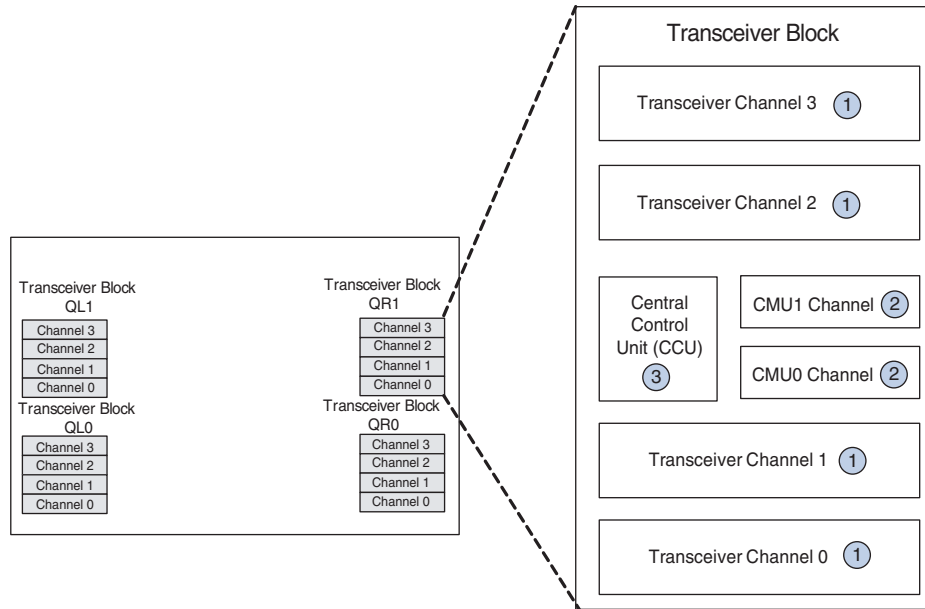
Note to Figure 1-10:

(1) 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 1-11 shows the transceiver block architecture of Stratix GX and GT devices.

Figure 1-11. 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 600 Mbps to 11.3 Gbps in Stratix IV GT devices. For more information, refer to [“Transceiver Channel Architecture”](#) on page 1-17.
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 1-100.
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, PCIe 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, PCIe, and Basic $\times 4$.
 - The PCIe rateswitch controller block controls the rateswitch circuit in the CMU0 channel in $\times 4$ configurations. In PCIe $\times 8$ configuration, the PCIe rateswitch controller block of the CCU in the master transceiver block is active. For more information, refer to [“PCIe Gen2 \(5 Gbps\) Support”](#) on page 1-140.

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

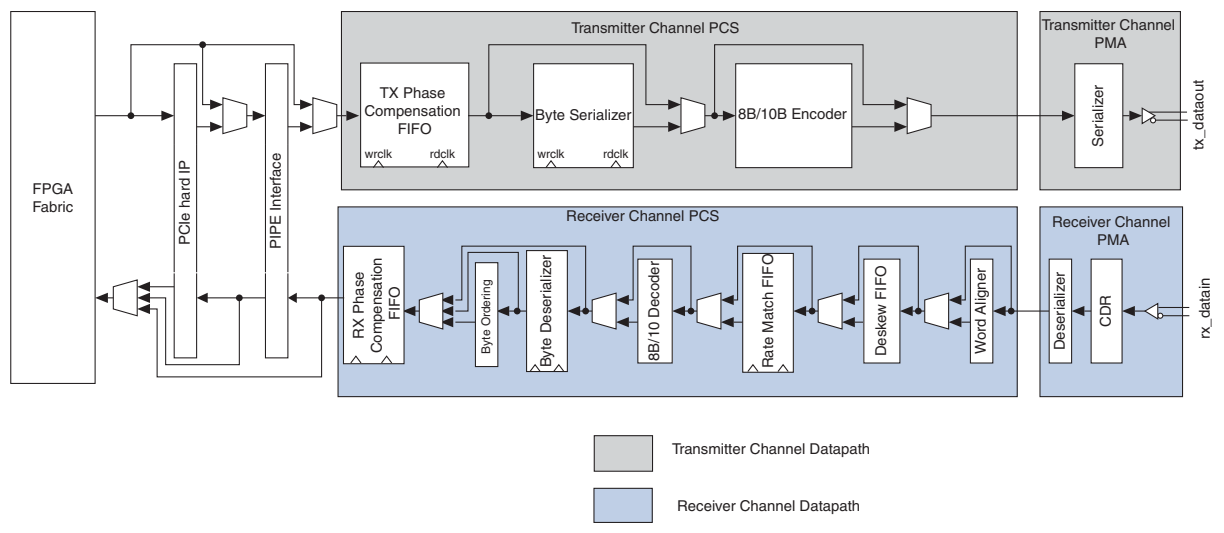
Four transceiver channels and two CMU channels are located in each transceiver block on the left and right sides of the device. Each Stratix IV GT device also has two 10G ATX PLLs that support data rates between 9.9 Gbps and 11.3 Gbps. Additionally, each Stratix IV GT device has two 6G ATX PLLs that support data rates between 600 Mbps 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 600 Mbps and 6.5 Gbps.

Transceiver Channel Architecture

Figure 1-12 shows the Stratix IV GX and GT transceiver channel datapath.

Figure 1-12. 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 PCIe hard IP block (PCIe hard IP-transceiver interface) or directly to the FPGA fabric (FPGA fabric-transceiver interface). The transceiver channel interfaces to the PCIe hard IP block if the hard IP block is used to implement the PCIe 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 600 Mbps and 8.5 Gbps
- 10G Transceiver Channel—supports data rates between 600 Mbps and 11.3 Gbps

 The PCIe hard IP-transceiver interface is beyond the scope of this chapter. This chapter describes the FPGA fabric-transceiver interface.


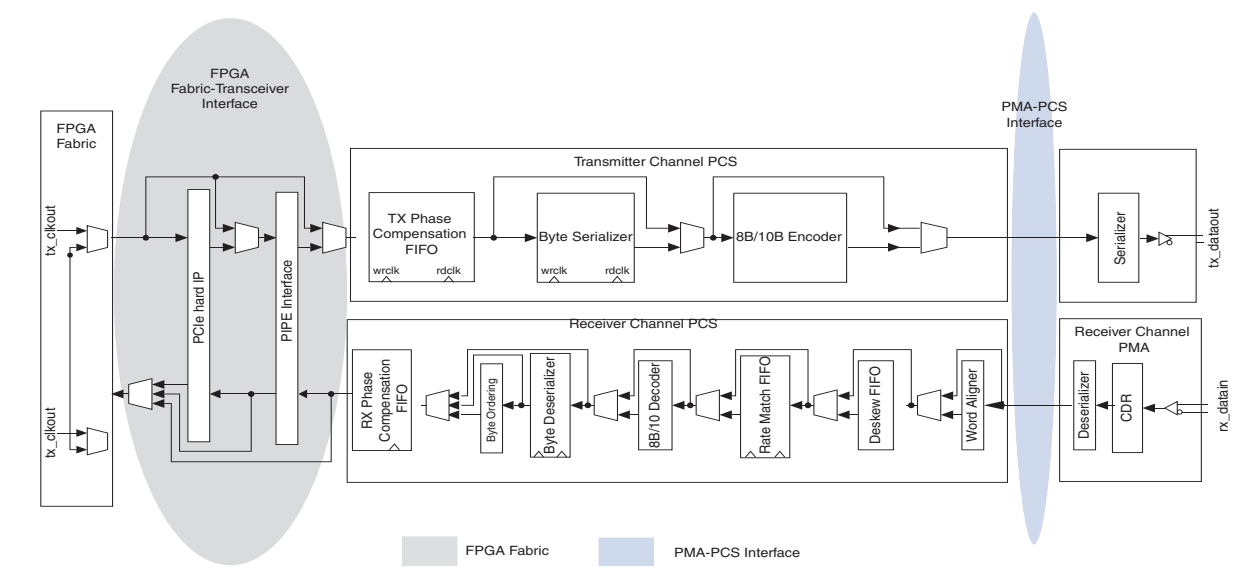
 For more information about the PCIe hard IP block, refer to the *PCI Express Compiler User Guide*.

Figure 1-13 shows the FPGA fabric-transceiver interface and transceiver PMA-PCS interface.

Figure 1-13. 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 lists 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 (Part 1 of 2)

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

Table 1-6. FPGA Fabric-Transceiver Interface Width and Transceiver PMA-PCS Widths (Part 2 of 2)

Name	Single-Width	Double-Width
Supported functional modes	<ul style="list-style-type: none"> ■ PCIe Gen1 and Gen2 ■ XAUI ■ GIGE ■ Serial RapidIO ■ SONET/SDH OC12 and OC48 ■ SDI ■ Basic single-width 	<ul style="list-style-type: none"> ■ (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 1-12 on page 1-17](#), consists of the following blocks:

- TX phase compensation FIFO
- Byte serializer
- 8B/10B encoder
- Transmitter output buffer

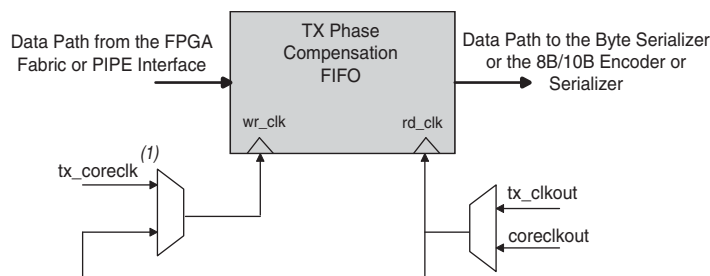
The Stratix IV GX and GT transceiver provides the **Enable low latency PCS mode** option in the ALTGX MegaWizard™ Plug-In Manager. If you select this option, the 8B/10B encoder in the datapath is disabled.

TX Phase Compensation FIFO

The TX phase compensation FIFO interfaces the transmitter channel PCS and the FPGA fabric PCIe 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 1-14](#) shows the datapath and clocking of the TX phase compensation FIFO.

Figure 1-14. TX Phase Compensation FIFO



Note to Figure 1-14:

(1) The clock used to clock the write side of the compensation FIFO is based on whether or not you enable the tx_coreclk option in the ALTGX MegaWizard Plug-In Manager.

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 <code>tx_clkout</code> port of the associated channel.
Bonded Functional	For example, in XAUI mode, the write clock of the FIFO is clocked by <code>coreclkout</code> provided by the <code>CMU0</code> clock divider block. You can clock the write side using <code>tx_coreclk</code> provided from the FPGA fabric by enabling the <code>tx_coreclk</code> 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 *Transceiver Clocking in Stratix IV Devices* chapter.

Input Data

In PCIe functional mode, the input data comes from the PCIe 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 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 for Stratix IV Devices* 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.

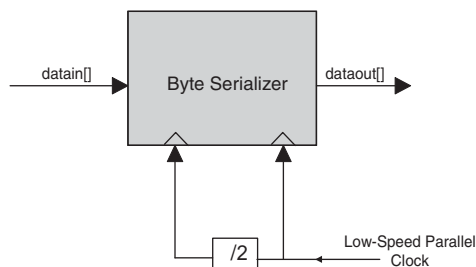


For more information about the maximum frequency limit for the transceiver interface, refer to the *DC and Switching Characteristics for Stratix IV Devices* chapter.

Single-Width Mode

Figure 1–15 shows the byte serializer datapath in single-width mode. For data port width, refer to Table 1–9.

Figure 1–15. Byte Serializer Datapath in Single-Width Mode (1), (2)



Notes to Figure 1–15:

- (1) For the `datain[]` and `dataout[]` port widths, refer to Table 1–9.
- (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 LSByte first, followed by the MSByte. 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 lists the input and output data widths of the byte serializer in single-width mode.

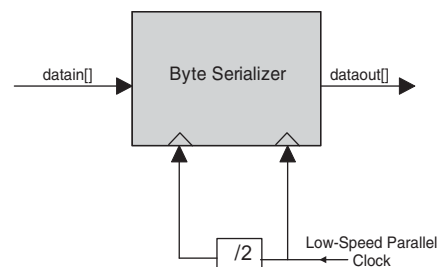
Table 1-9. Input and Output Data Width of the Byte Serializer in Single-Width Mode

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 1-16 shows the byte serializer datapath in double-width mode. For data port width, refer to Table 1-10.

Figure 1-16. Byte Serializer Datapath in Double-Width Mode (1), (2)



Notes to Figure 1-16:

- (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 lists 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

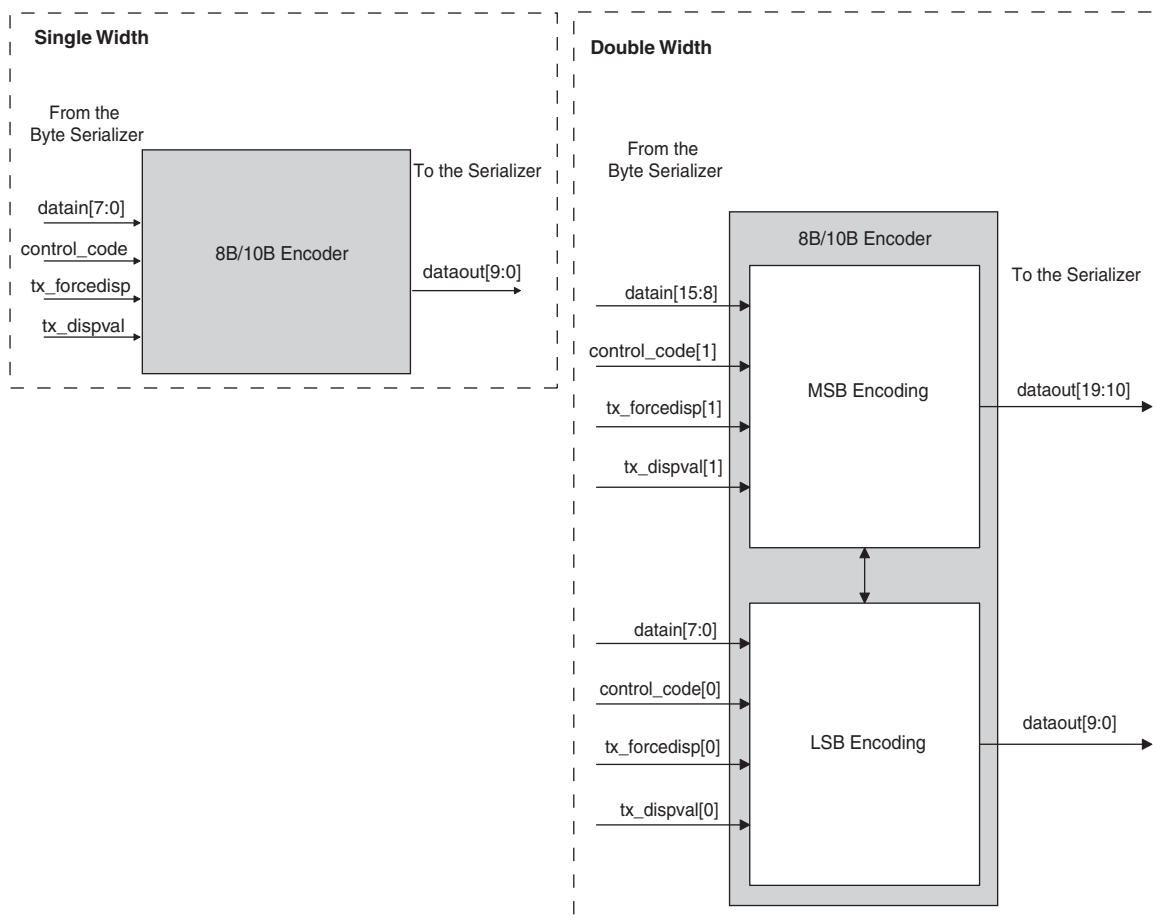
Asserting the `tx_digitalreset` signal resets the byte serializer block.

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 1-17 shows the 8B/10B encoder in single-width and double-width mode.

Figure 1-17. 8B/10B Encoder in Single-Width Mode



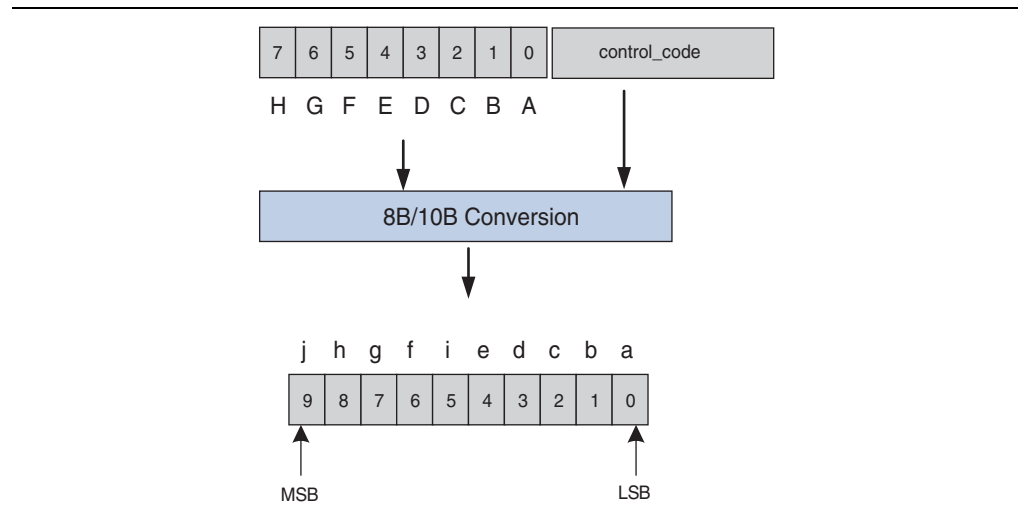
Single-Width Mode

The left side of Figure 1-17 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 1-27.

Figure 1-18 shows the conversion format. The LSB is transmitted first.

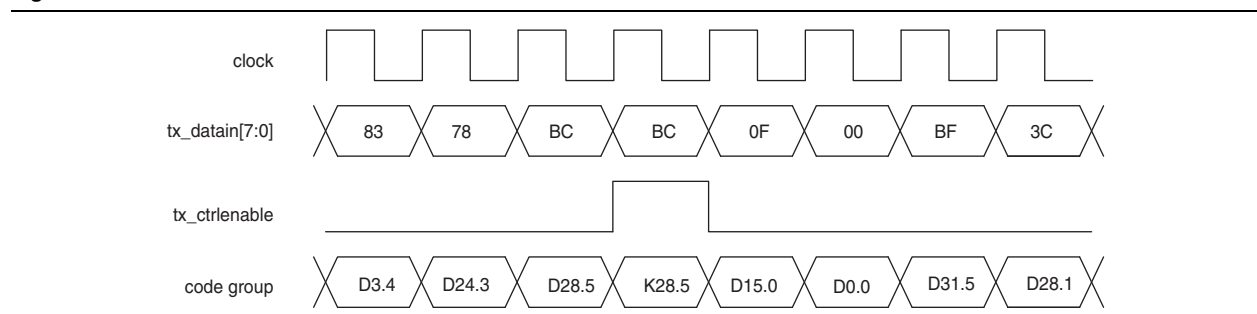
Figure 1-18. 8B/10B Conversion Format




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 ($K_{x.y}$). When `tx_ctrlenable` is low, the 8B/10B encoder block encodes the byte at the `tx_datain` port (the user-input port) as data ($D_{x.y}$). When `tx_ctrlenable` is high, the 8B/10B encoder encodes the input data as a $K_{x.y}$ code group. The waveform in Figure 1-19 shows the second $0 \times BC$ encoded as a control word ($K_{28.5}$). The rest of the `tx_datain` bytes are encoded as a data word ($D_{x.y}$).

Figure 1-19. 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 $D_{x.y}$ or $K_{x.y}$ code), or unintended valid $D_{x.y}$ code, depending on the value entered. It is possible for a downstream 8B/10B decoder to decode an invalid control word into a valid $D_{x.y}$ 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 $10'b0110001100$ ($0 \times 18C$), 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.


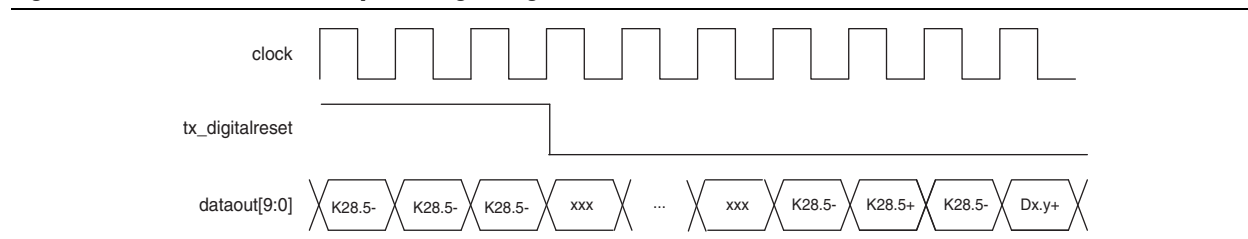
 While $tx_digitalreset$ is asserted, the downstream 8B/10B decoder that receives the data might observe synchronization or disparity errors.

Figure 1-20 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 1-20. 8B/10B Encoder Output During $tx_digitalreset$ Assertion

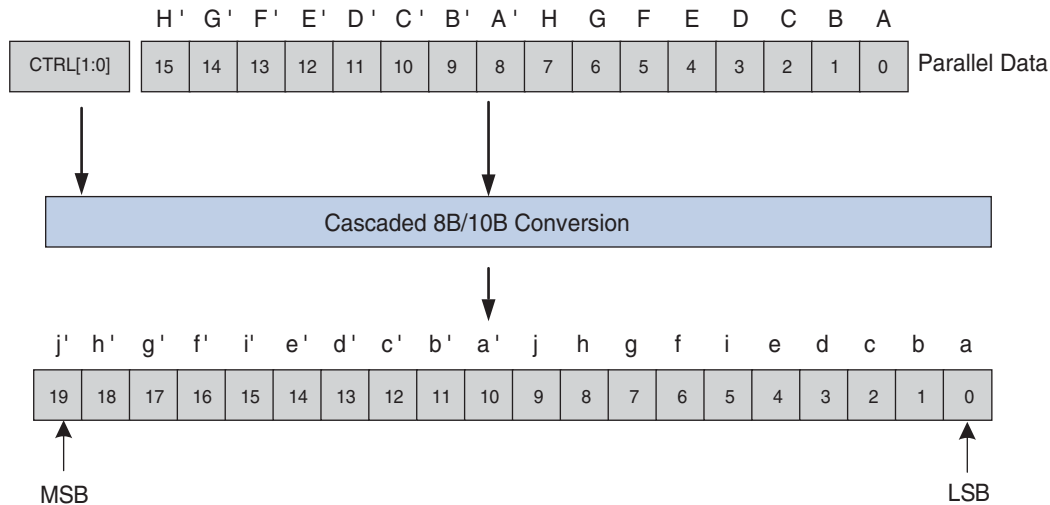


Double-Width Mode

In double-width mode, the 8B/10B encoder operates in a cascaded mode, as shown on the right side of Figure 1-20 on page 1-25. The LSBByte 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 1-21 shows the conversion format. The LSB shown in Figure 1-21 is transmitted first.

Figure 1-21. 8B/10B Conversion Format in Double-Width Mode

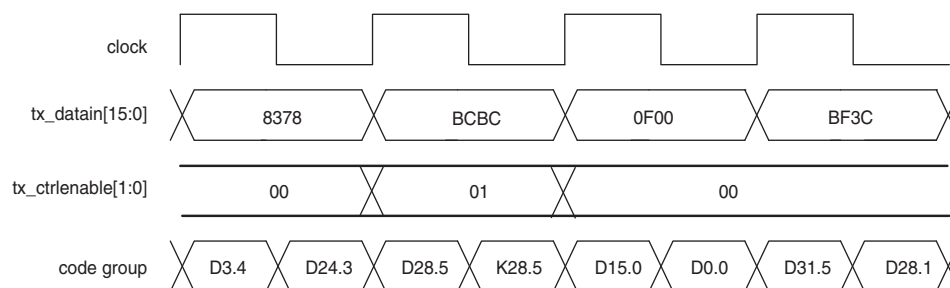


Control Code Encoding

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 1-22 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 1-22. 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 $10'b0110001100$ ($0 \times 18C$), 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.


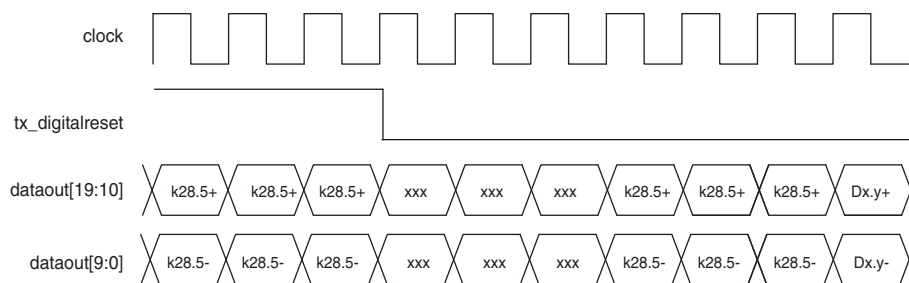
 If the `tx_digitalreset` signal is asserted, the downstream 8B/10B decoder receiving the data might get synchronization or disparity errors.

Figure 1-23 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 1-23 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 1-23. 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 lists the `tx_forcedisp` and `tx_dispval` port values.

Table 1-11. `tx_forcedisp` and `tx_dispval` Port Values

<code>tx_forcedisp</code>	<code>tx_dispval</code>	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 1-24 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 1-24. 8B/10B Encoder Force Running Disparity Operation in Single-Width Mode

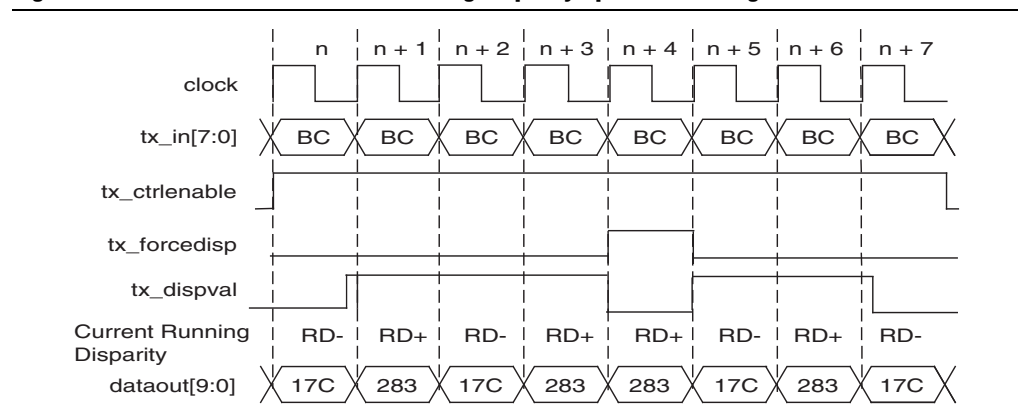
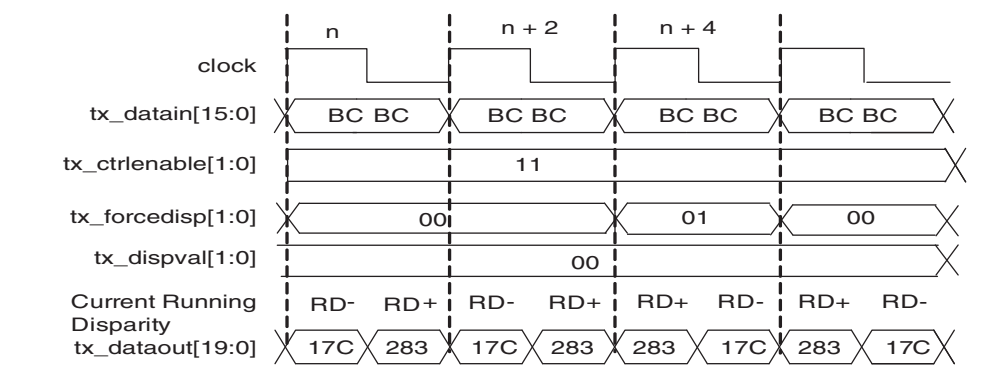


Figure 1-25 shows the current running disparity being altered in Basic double-width mode by forcing a positive disparity on a negative disparity K28.5. In this example, a series of K28.5 are continuously being sent. The stream alternates between a positive ending running disparity (RD+) K28.5 and a negative 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 1–25. 8B/10B Encoder Force Current Running Disparity in Double-Width Mode



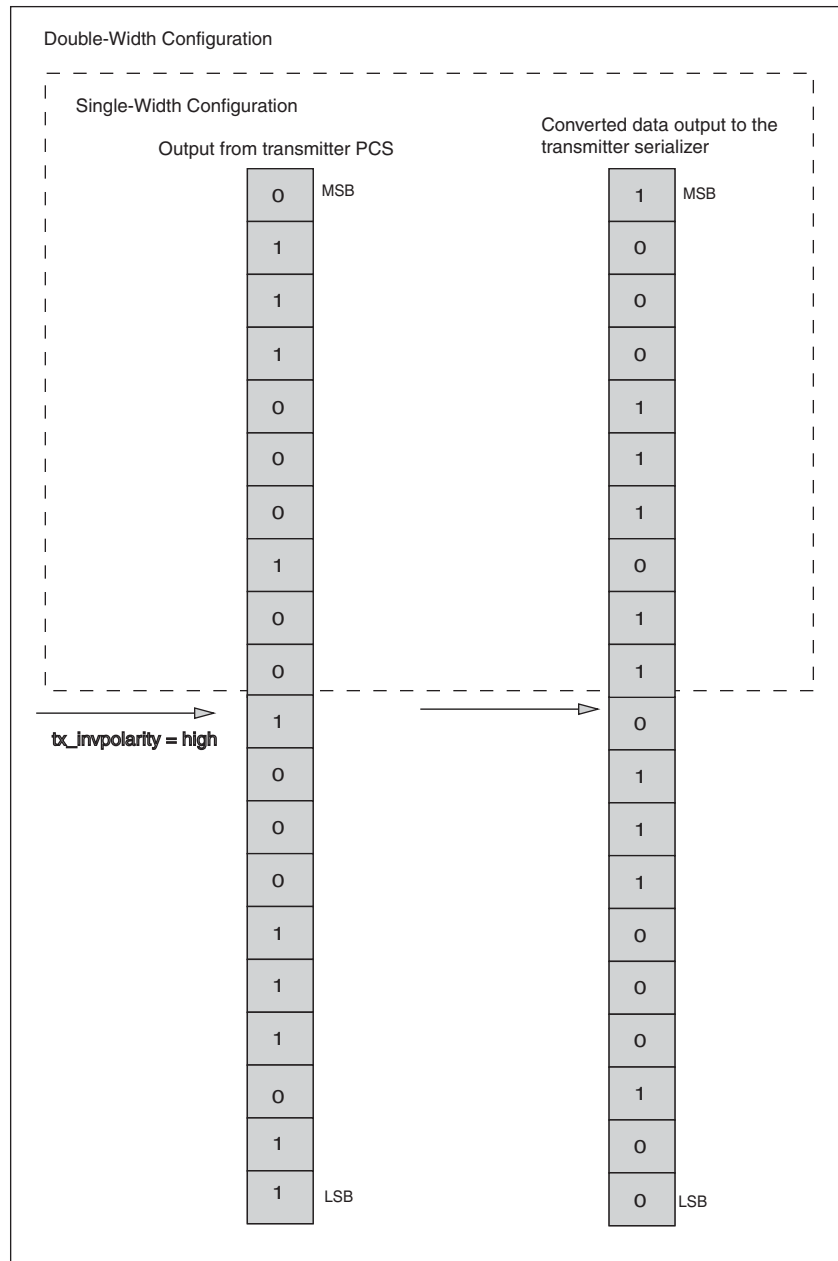
Transmitter Polarity Inversion

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 1-26 shows the transmitter polarity inversion feature in a single-width and double-width datapath configuration.

Figure 1-26. Transmitter Polarity Inversion in Single-Width and Double-Width Mode



Transmitter Bit Reversal

Table 1-12 lists 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: <ul style="list-style-type: none"> ■ 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: <ul style="list-style-type: none"> ■ 16-bit—D[15:0] rewired to D[0:15] ■ 20-bit—D[19:0] rewired to D[0:19]

Figure 1-27 shows the transmitter bit reversal feature in Basic single-width for a 10-bit wide datapath configuration.

Figure 1-27. Transmitter Bit Reversal Operation in Basic Single-Width Mode

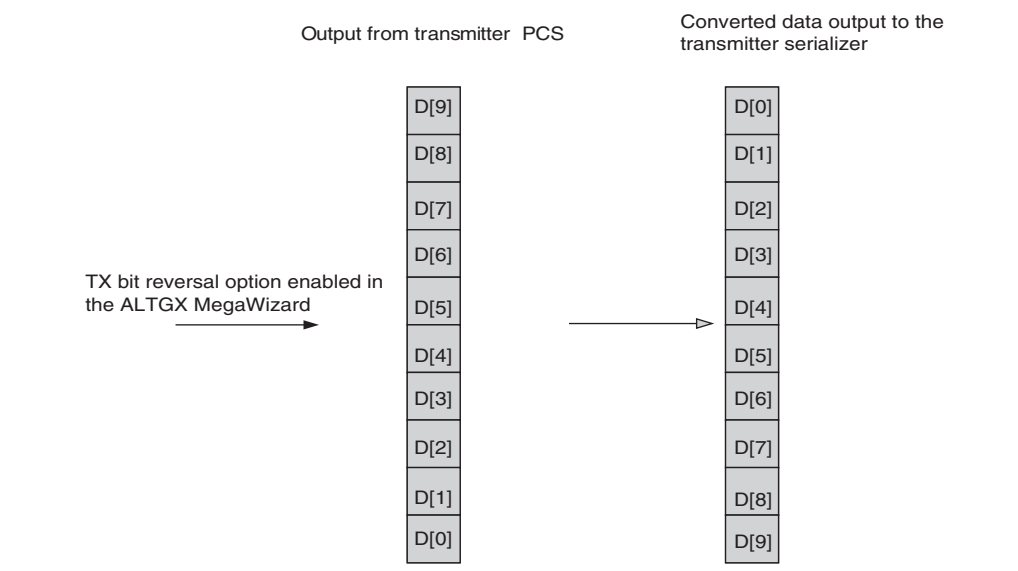
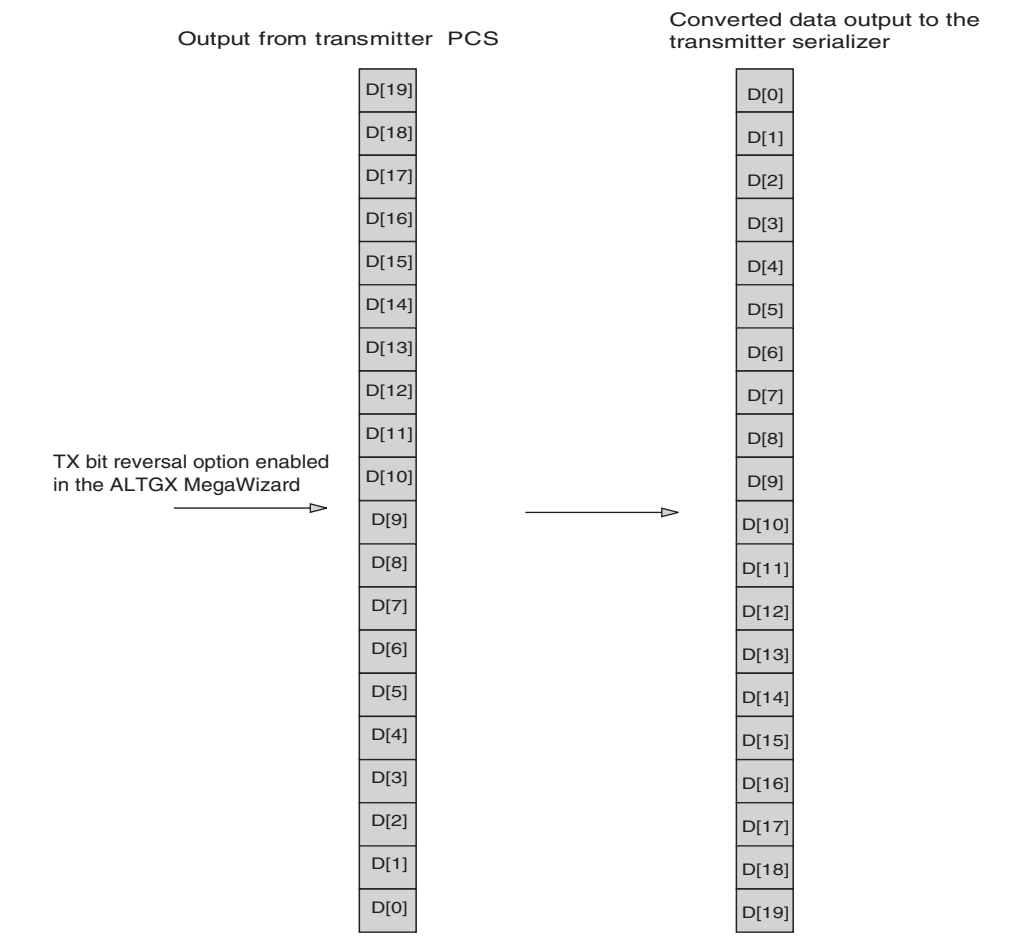


Figure 1-28 shows the transmitter bit reversal feature in Basic double-width mode for a 20-bit wide datapath configuration.

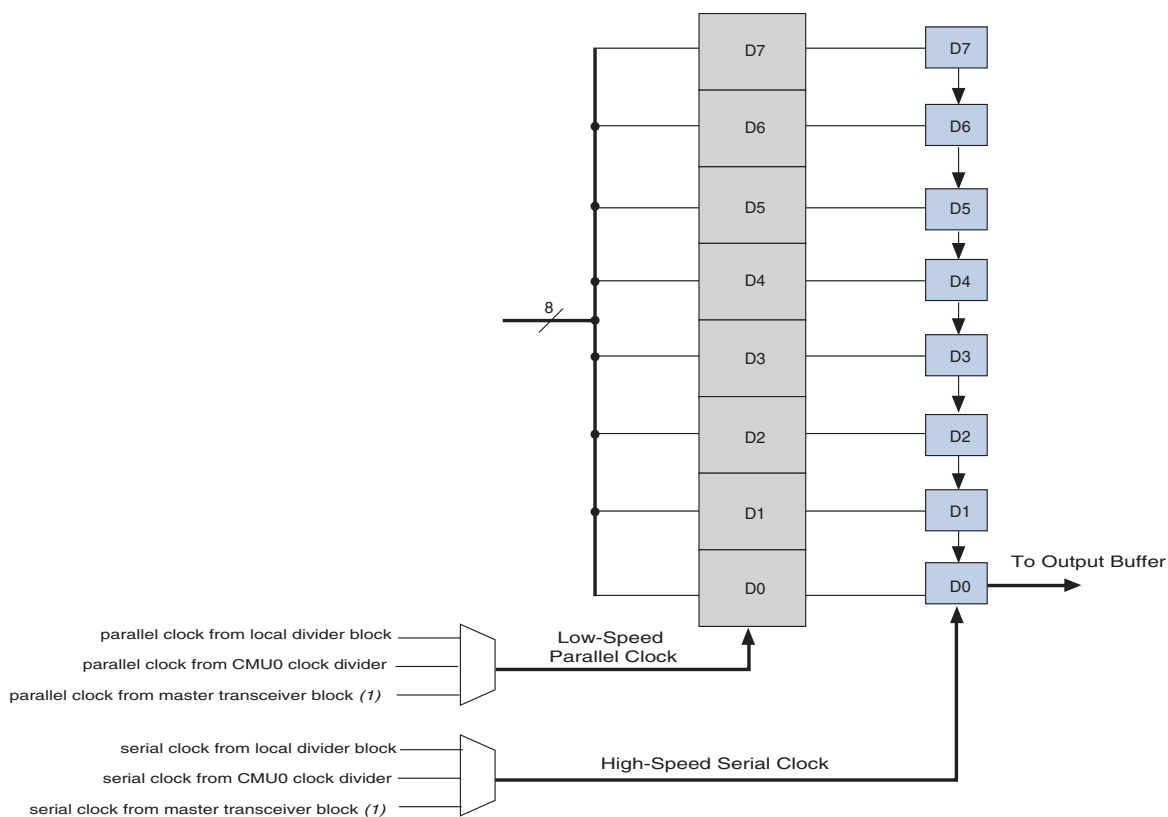
Figure 1-28. Transmitter Bit Reversal Operation in Basic Double-Width Mode



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 1-29. The serializer block sends out the LSB of the input data.

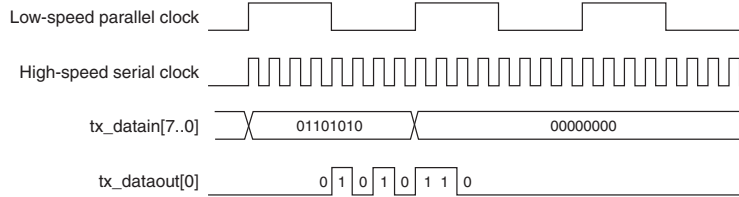
Figure 1-29. Serializer Block in 8-Bit PCS-PMA Interface



Note to Figure 1-29:

(1) The CMU0 clock divider of the master transceiver block provides the clocks. It is used only in bonded modes (for example, Basic x8, PCIe x8 mode).

Figure 1-30 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 1-30. Serializer Bit Order ⁽¹⁾**Note to Figure 1-30:**

(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 1-31, 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 PCIe functional mode.

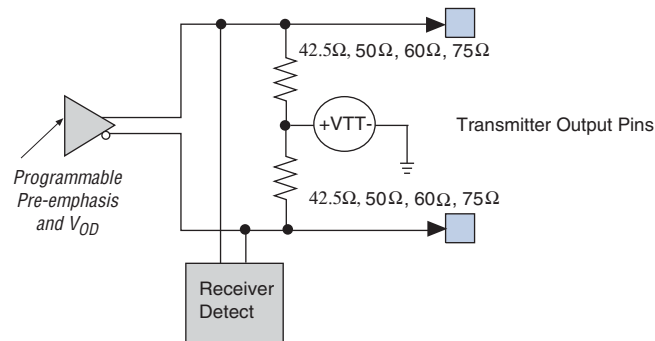
Figure 1-31. Transmitter Output Buffer

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 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	600 Mbps—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 1-201), 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 1-36.

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.

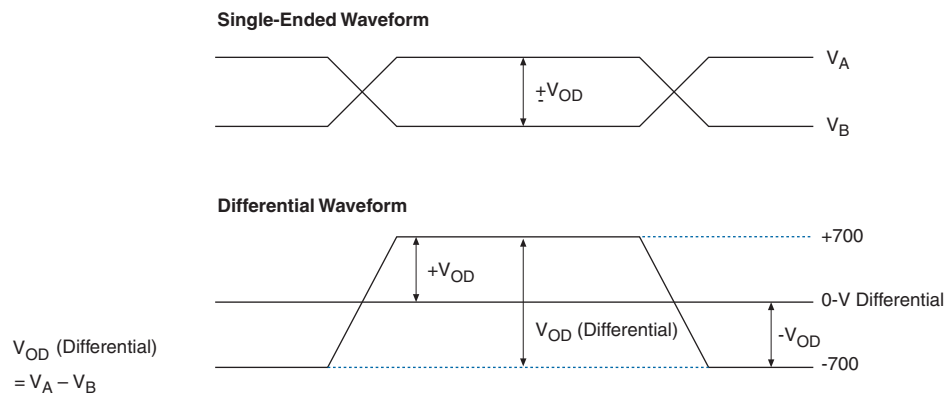
Table 1-15. Stratix IV GX and GT OCT Assignment Settings

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 1-32. 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 1-32. V_{OD} (Differential) Signal Level



 For more information about Stratix IV GX and GT V_{OD} values, refer to the *DC and Switching Characteristics for Stratix IV Devices* 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	600 Mbps to 11.3 Gbps
1.5 V	600 Mbps to 6.5 Gbps	—

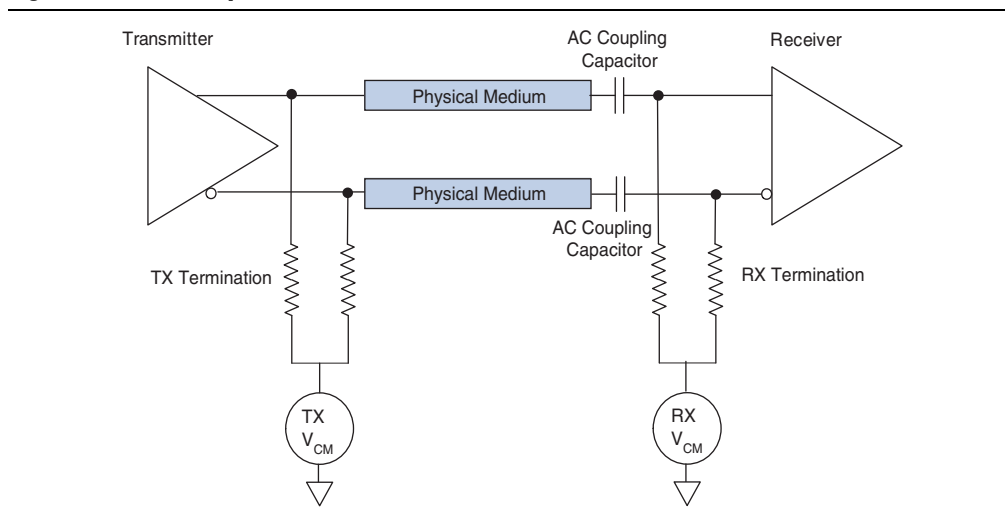
Link Coupling for Stratix IV GX and GT Devices

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 1-33 shows an AC-coupled link.

Figure 1-33. AC-Coupled Link



The following protocols supported by Stratix IV GX and GT devices mandate AC-coupled links:

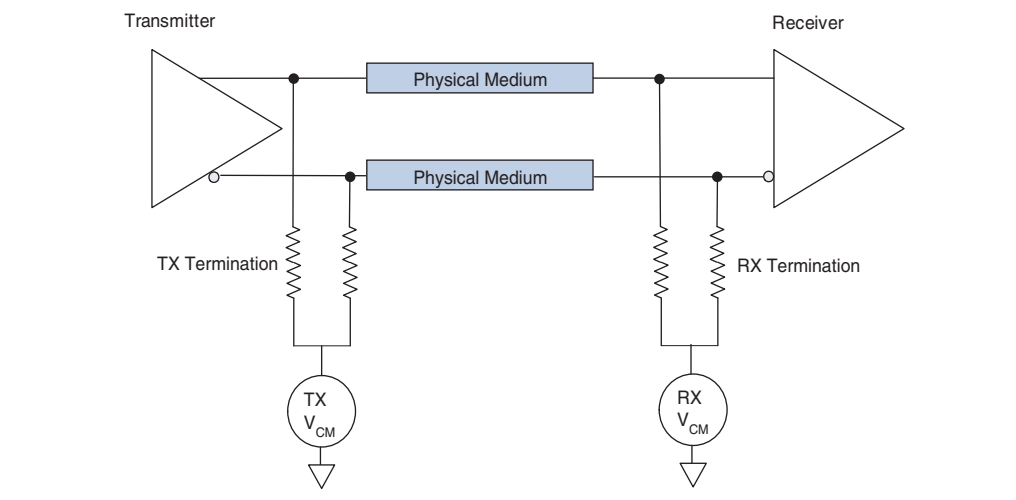
- PCIe
- Gigabit Ethernet
- Serial RapidIO
- XAUI
- SDI

Stratix IV GT devices allow the high-speed links to be AC-coupled for the entire data rate range between 600 Mbps 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 1-34 shows a DC-coupled link.

Figure 1-34. 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 600 Mbps to 11.3 Gbps with Tx $V_{cm} = 0.65$ V and Rx $V_{cm} = 0.82$ V.

For more information on the DC coupling capabilities of the Stratix IV GT device, refer to Table 1-23 on page 1-48.


PCIe Receiver Detect

The Stratix IV GX and GT transmitter buffers have a built-in receiver detection circuit for use in the PCIe 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 PCIe 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 PCIe 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 PCIe 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 `pestatus` 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 1-109 on page 1-134](#).

 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.

PCIe Electrical Idle

The Stratix IV GX and GT transmitter output buffers support transmission of PCIe 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 PCIe mode, refer to [Figure 1-108 on page 1-133](#).

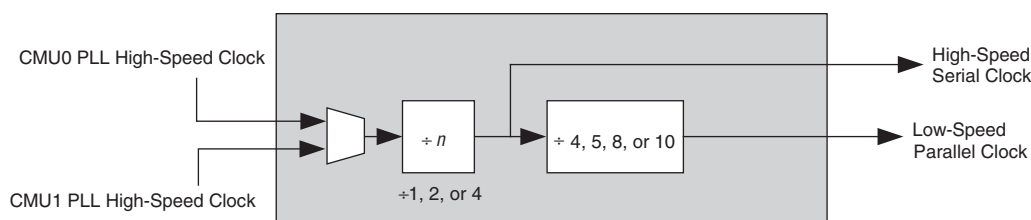
 For more information about using the `tx_forceelecidle` signal under different power states, refer to the PCIe specification 2.0.

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 1-35](#) shows the transmitter local clock divider block.

Figure 1-35. 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 1-12 on page 1-17](#) 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
- PCIe interface

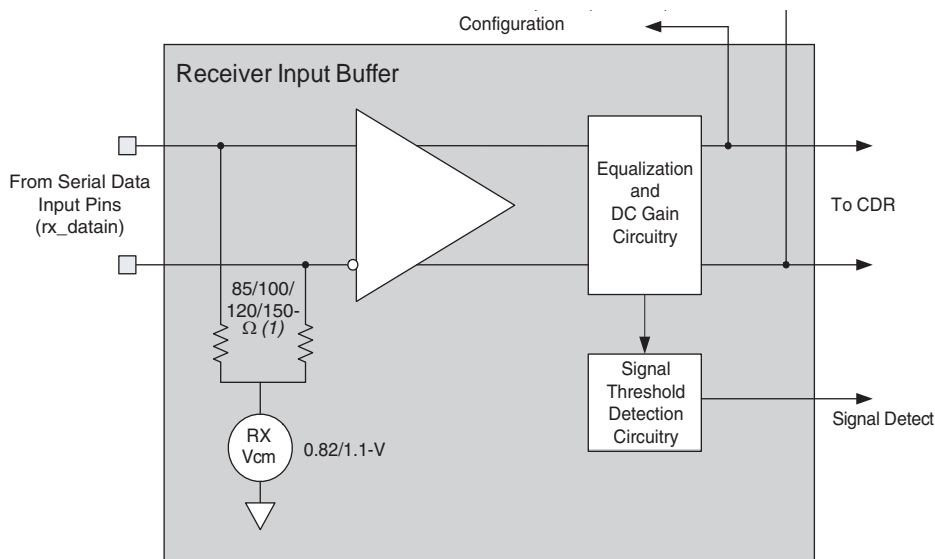
The receiver datapath is very flexible and allows multiple configurations, depending on the selected functional mode. You can configure the receiver datapath using the ALTGX MegaWizard Plug-In Manager.

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 ($Rx V_{CM}$), 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 1-36 shows the receiver input buffer.

Figure 1-36. Receiver Input Buffer



Note:
(1) These resistor values represent the combined total of the values in both resistors.

Table 1-17 lists 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 (1)

Data Rate	I/O Standard	Differential OCT with Calibration (Ω)	V_{CM} (V)	Coupling	Programmable DC Gain (dB)
Stratix IV GX 0.6 Gbps to 8.5 Gbps	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 0.6 Gbps to 11.3 Gbps	1.4 V PCML	85, 100, 120, 150	0.82	AC, DC	up to 16
	LVDS	85, 100, 120, 150	1.1	AC, DC	up to 16

Note to Table 1-17:

(1) Programmable equalization settings are 1 to 16 dB for Stratix IV GX and GT devices; for example, rx_eqctrl = 4'h0 maps to 1 dB gain, rx_eqctrl = 4'h1 maps to 2 dB gain, and so on.

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 , Off

 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 1-201](#).


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.

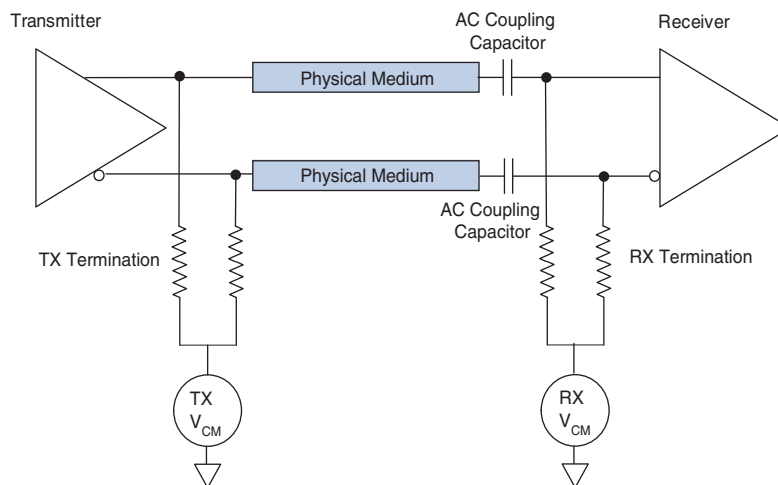
Link Coupling for Stratix IV GX Devices

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 1-37 shows an AC-coupled link.

Figure 1-37. AC-Coupled Link



Note to Figure 1-37:

(1) The receiver termination and biasing can be on-chip or off-chip.

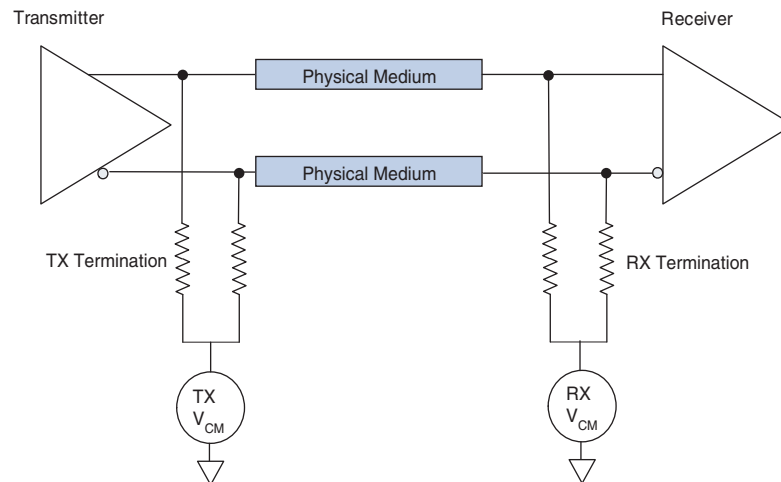
The following protocols supported by Stratix IV GX and GT devices mandate AC-coupled links:

- PCIe
- 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 1-38 shows a DC-coupled link.

Figure 1-38. DC-Coupled Link



Note to Figure 1-38:

(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:

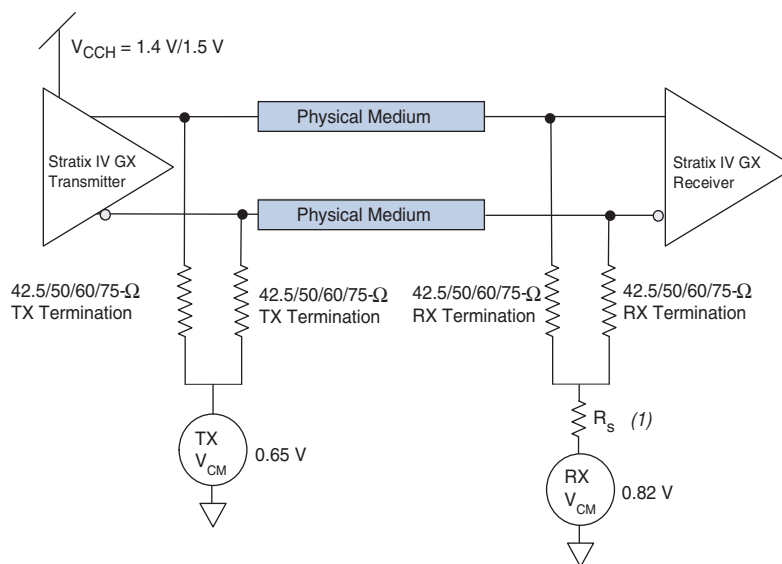
- Basic single- and double-width
- (OIF) CEI PHY interface

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 1-39 shows a typical Stratix IV GX transmitter (PCML) to Stratix IV GX Receiver (PCML) DC-coupled link.

Figure 1-39. Stratix IV GX Transmitter (PCML) to Stratix IV GX Receiver (PCML)



Note to Figure 1-39:

(1) R_s is the parasitic resistance present in the on-chip RX termination and biasing circuitry.

Table 1-19 lists 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

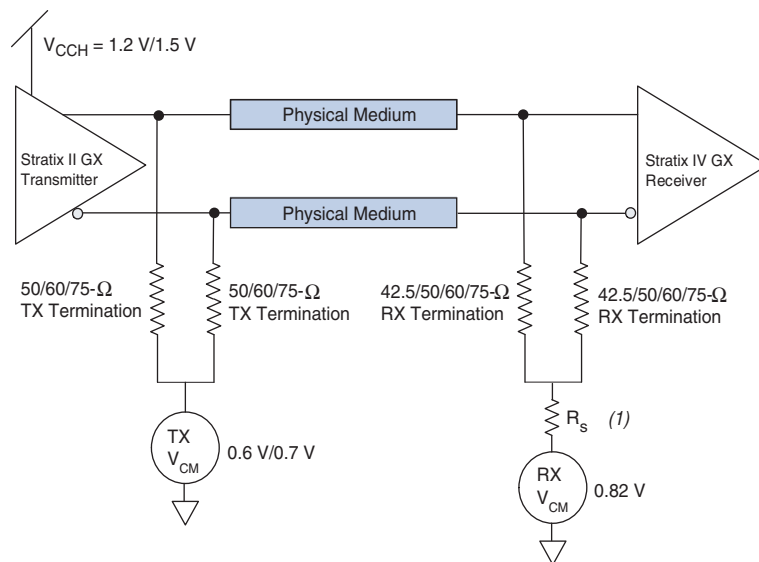
Transmitter (Stratix IV GX) Settings				Receiver (Stratix IV GX) Settings		
Data Rate	V_{CCH} (1)	TX V_{CM}	Differential Termination	Data Rate	RX V_{CM}	Differential Termination
600-8500 Mbps	1.4 V/1.5 V	0.65 V	85/100/120/150 Ω	600-8500 Mbps	0.82 V	85/100/120/150 Ω

Note to Table 1-19:

(1) $V_{CCH} = 1.5$ V can support data rates from 600 Mbps to 6500 Mbps. $V_{CCH} = 1.4$ V can support data rates from 600 Mbps to 8500 Mbps.

Figure 1-40 shows the Stratix II GX transmitter (PCML) to Stratix IV GX receiver (PCML) coupled link.

Figure 1-40. Stratix II GX Transmitter (PCML) to Stratix IV GX Receiver (PCML)



Note to Figure 1-40:

(1) R_S is the parasitic resistance present in the on-chip RX termination and biasing circuitry.

Table 1-20 lists 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

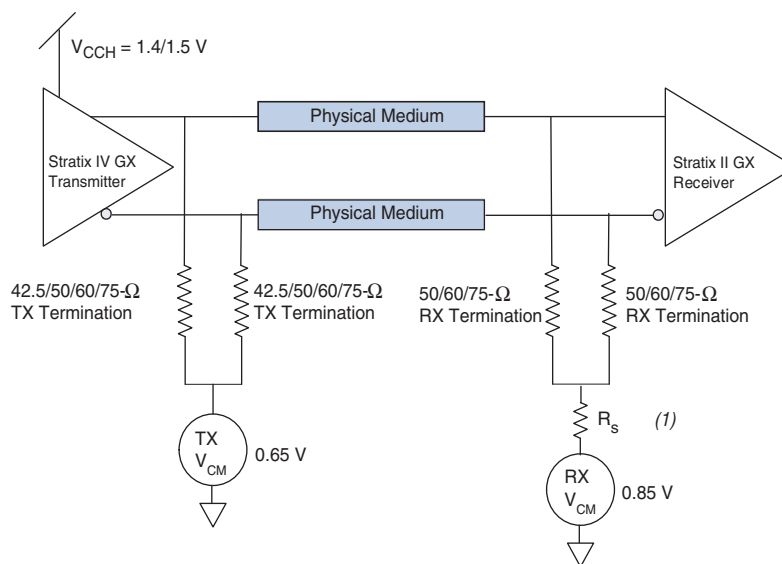
Transmitter (Stratix II GX) Settings				Receiver (Stratix IV GX) Settings		
Data Rate	V_{CCH} (1)	$\text{TX } V_{CM}$ (1)	Differential Termination	Data Rate	$\text{RX } V_{CM}$	Differential Termination
600-6375 Mbps	1.5 V (1.5 V PCML)	0.6 V/0.7 V	100/120/150 Ω	600-6375 Mbps	0.82 V	100/120/150 Ω

Note to Table 1-20:

(1) $V_{CCH} = 1.5\text{ V}$ with $\text{TX } V_{CM} = 0.7\text{ V}$ can support data rates from 600 Mbps to 3125 Mbps. $V_{CCH} = 1.5\text{ V}$ with $\text{TX } V_{CM} = 0.6\text{ V}$ can support data rates from 600 Mbps to 6375 Mbps.

Figure 1-41 shows the Stratix IV GX transmitter (PCML) to Stratix II GX receiver (PCML) DC-coupled link.

Figure 1-41. Stratix IV GX Transmitter (PCML) to Stratix II GX Receiver (PCML)



Note to Figure 1-41:

(1) R_s is the parasitic resistance present in the on-chip RX termination and biasing circuitry.

Table 1-21 lists 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

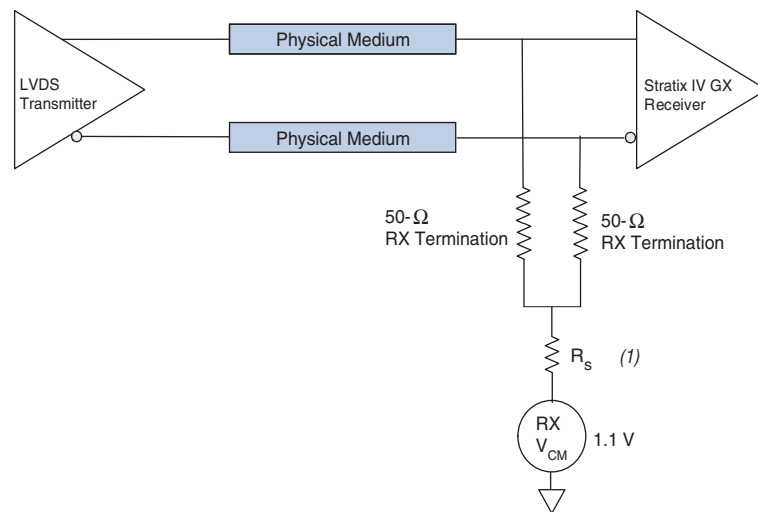
Transmitter (Stratix IV GX) Settings				Receiver (Stratix II GX) Settings			
Data Rate	V_{CCH} (1)	TX V_{CM}	Differential Termination	Data Rate	I/O Standard	RX V_{CM}	Differential Termination
600-6375 Mbps	1.4/1.5 V	0.65 V	100/120/150 Ω	600-6375 Mbps	1.4/1.5 V PCML	0.85 V	100/120/150 Ω

Note to Table 1-21:

(1) $V_{CCH} = 1.5$ V can support data rates from 600 Mbps to 6500 Mbps. $V_{CCH} = 1.4$ V can support data rates from 600 Mbps to 6375 Mbps.

Figure 1-42 shows the LVDS transmitter to Stratix IV GX receiver (PCML) DC-coupled link.

Figure 1-42. LVDS Transmitter to Stratix IV GX Receiver (PCML)



Note to Figure 1-42:

(1) R_s is the parasitic resistance present in the on-chip RX termination and biasing circuitry.

Table 1-22 lists 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 ⁽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 R_s 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 600 Mbps 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)	600 Mbps to 11.3 Gbps	<ul style="list-style-type: none"> ■ 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)	600 Mbps to 8.5 Gbps	<ul style="list-style-type: none"> ■ 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)	600 Mbps to 6.375 Gbps	<ul style="list-style-type: none"> ■ TX $V_{CM} = 0.7$ V (600 Mbps 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)	600 Mbps to 6.5 Gbps	<ul style="list-style-type: none"> ■ 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 PCIe 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 PCIe compliance parameter VRX-IDLE-DETDIFFp-p is available in the *DC and Switching Characteristics for Stratix IV Devices* 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 settings 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.

To enable the AEQ feature, in ALTGX and ALTGX_RECONFIG MegaWizard Plug-In Managers, select the **Enable adaptive equalizer control** option.

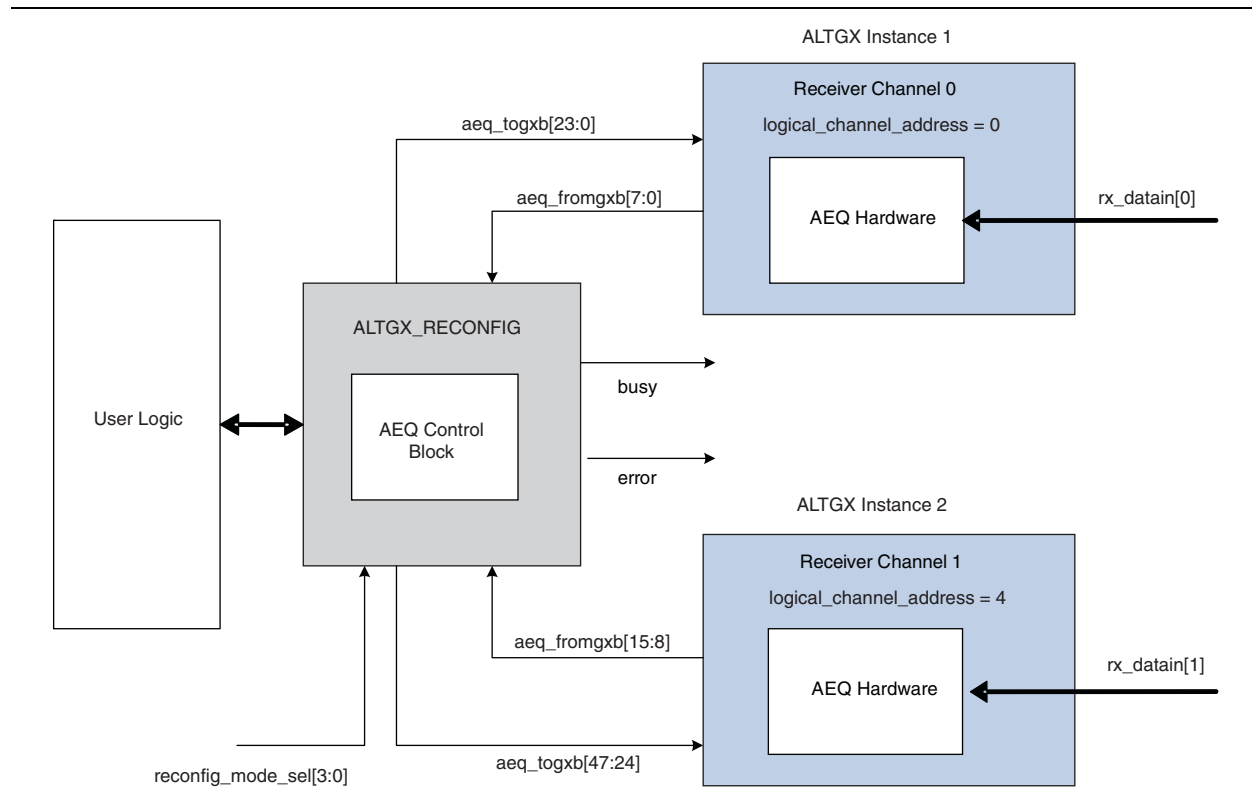
When you select AEQ, two ports, `aeq_fromgxb[]` and `aeq_togxb[]`, become available on the ALTGX and ALTGX_RECONFIG instances. These ports provide an interface between the PMA of the receiver channel and the AEQ control block in the ALTGX_RECONFIG MegaWizard Plug-In Manager.



AEQ hardware is not present in the CMU channels.

Figure 1-43 shows the receiver channel data path with the AEQ feature.

Figure 1-43. 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—This feature is not supported.
- One-time mode—The AEQ finds a stable setting of the receiver equalizer and locks that value. After it is 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—This feature is not supported.


 For more information about the AEQ port connections and various waveforms in all the above modes, refer to the *Dynamic Reconfiguration in Stratix IV Devices* 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.

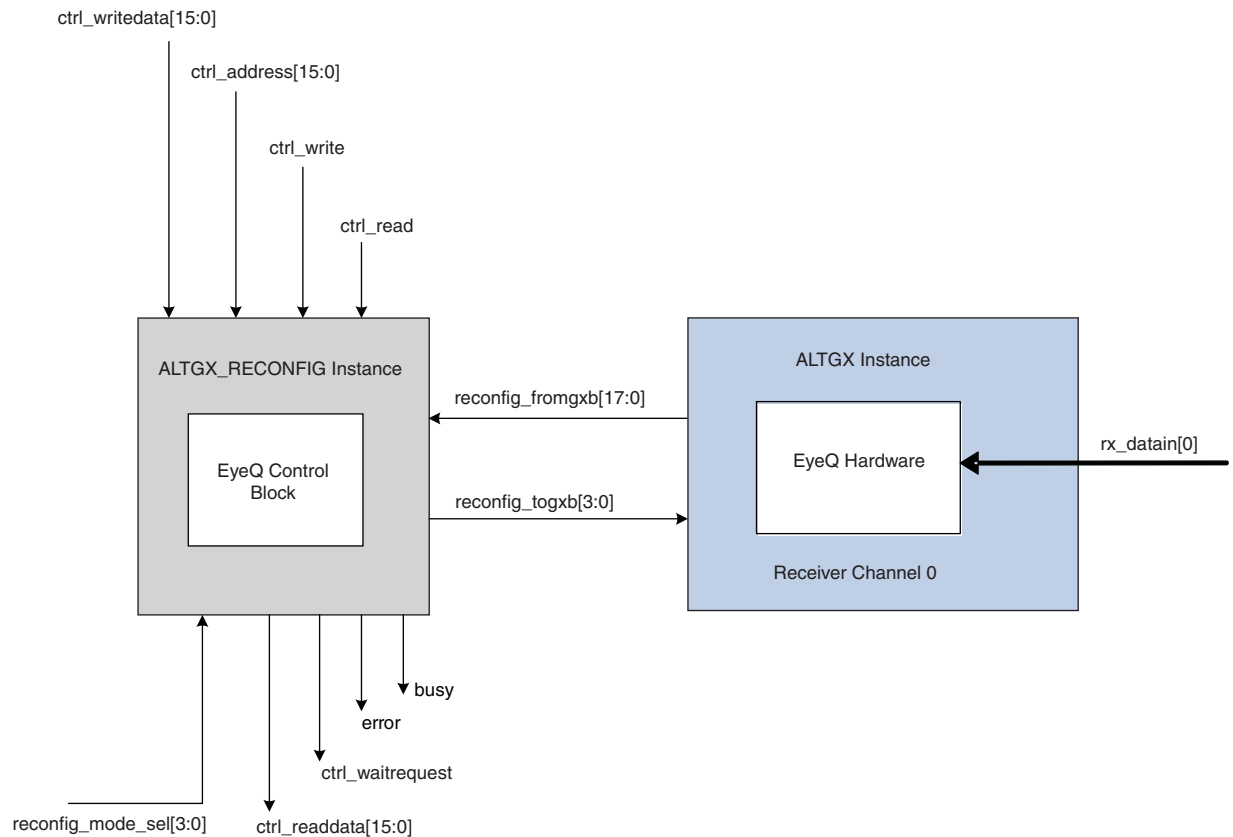
 The EyeQ hardware is available for both regular transceiver channels and CMU channels.

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 1-44](#) 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 1-44](#) shows the receiver channel data path using the EyeQ feature.

Figure 1-44. Receiver Channel Data Path showing the EyeQ Feature

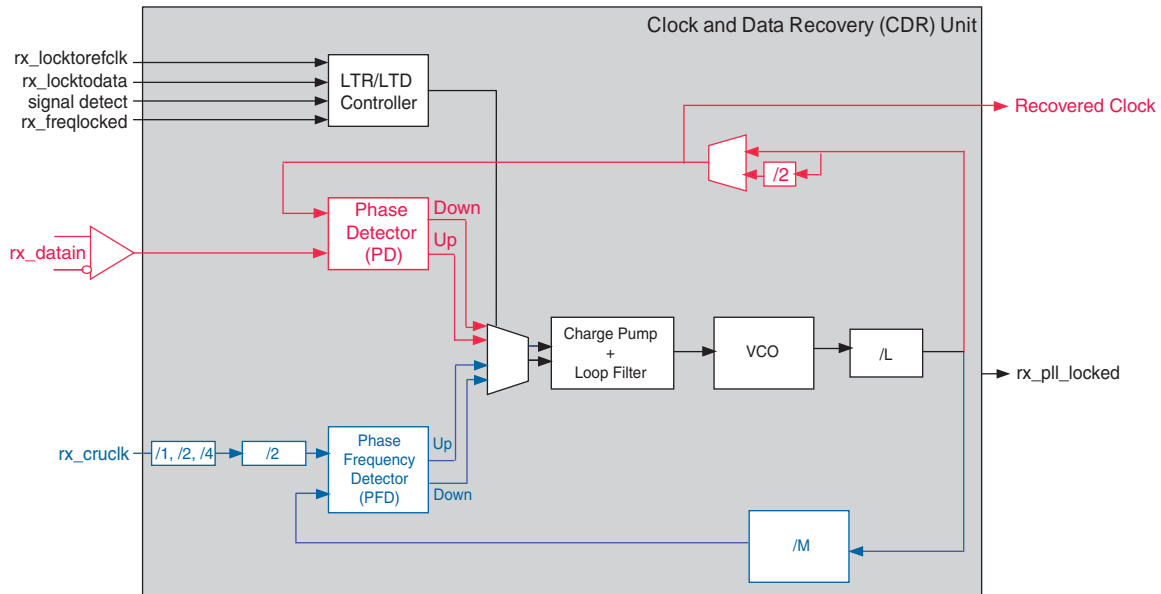


 For more information about using the EyeQ feature, refer to the *Dynamic Reconfiguration in Stratix IV Devices* chapter.

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 1-45 shows the CDR block diagram.

Figure 1-45. Clock and Data Recovery Unit (1)



Note to Figure 1-45:


(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. After it is 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_cruc1k. 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 the phase and frequency of the receiver input reference clock. Figure 1-45 on page 1-53 shows the active blocks (in blue) when the 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 lists CDR /M and /L divider values.

Table 1-24. CDR Divider Values

Parameter	Value
/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, or /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 1-45 on page 1-53 shows the 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 in Stratix IV Devices* chapter.

PCIe Clock Switch Circuitry

The feedback path from the CDR VCO to the PD has a /2 divider that is used in PCIe 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 PCIe signaling rateswitch, refer to “Dynamic Switch Between Gen1 (2.5 Gbps) and Gen2 (5 Gbps) Signaling Rate” on page 1-141.

 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_locktofreqclk` and `rx_locktodata`) allow you to configure the LTR/LTD controller in either automatic lock mode or manual lock mode. Table 1-25 lists 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

<code>rx_locktofreqclk</code>	<code>rx_locktodata</code>	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_locktofreqclk` 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 PCIe mode only. This condition is defaulted to **true** for all other 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 PCIe 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 in Stratix IV Devices](#) chapter.

Offset Cancellation in the Receiver Buffer and Receiver CDR

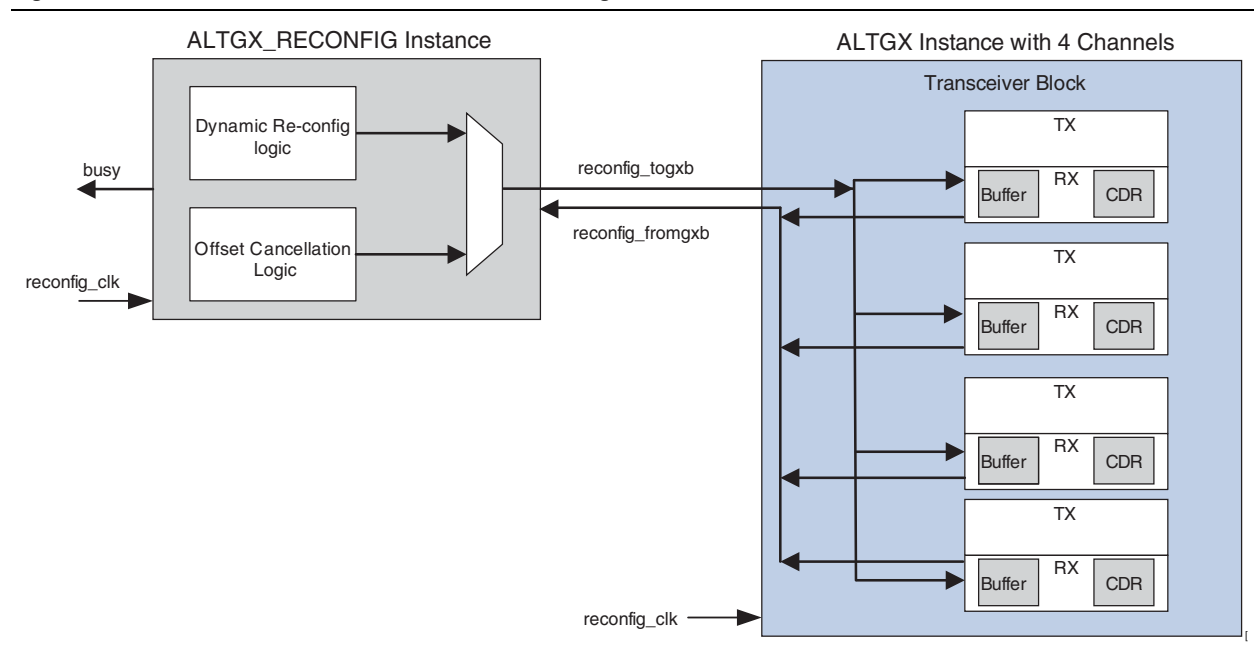
As silicon progresses towards smaller process nodes, the performance of circuits at these smaller nodes depends more on process variations. These process variations result in analog voltages that can be offset from the required ranges. Offset cancellation logic corrects these offsets. The receiver buffer and receiver CDR require offset cancellation.

Offset cancellation is executed automatically once each time a Stratix IV GX and GT device is powered on (after the device has finished programming and switches to user mode as indicated by `CONFIG_DONE=1`). 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.

-  You must reprogram your device to restart the Offset Cancellation process.
-  For more information about offset cancellation control logic connectivity, refer to the *Dynamic Reconfiguration in Stratix IV Devices* chapter.
-  During offset cancellation, signified by a high on the busy signal, `rx_analogreset` is not relevant until the busy signal goes low.

Offset cancellation logic requires a separate clock. In PCIe 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 1-46](#) shows the interface of the offset cancellation control logic (`ALTGX_RECONFIG` instance) and the `ALTGX` instance.

Figure 1-46. 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, the 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 about the offset cancellation process, refer to the *Reset Control and Power Down in Stratix IV Devices* 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 1-47 shows the deserializer operation in single-width mode with a 10-bit deserialization factor.

Figure 1-47. Deserializer Operation in Single-Width Mode

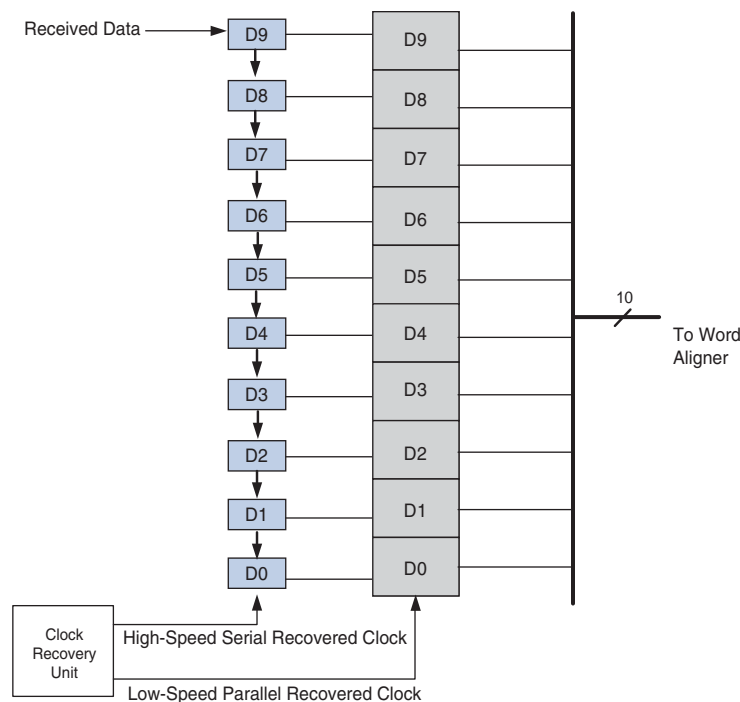
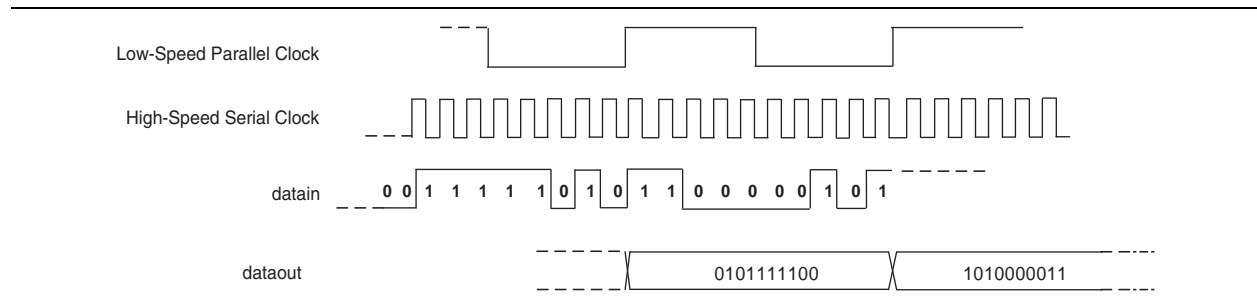


Figure 1-48 shows the serial bit order of the deserializer block input and the parallel data output of the deserializer block in single-width mode with a 10-bit deserialization factor. The serial stream (0101111100) is deserialized to a value 10'h17C. The serial data is assumed to be received LSB to MSB.

Figure 1-48. 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 PCIe, 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:

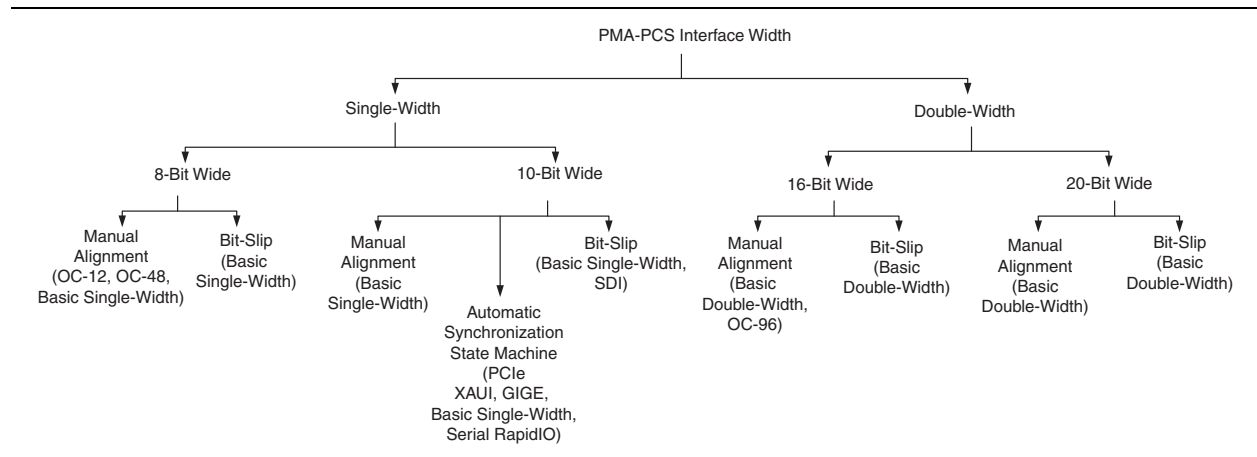
- Synchronization state machine in functional modes such as PCIe, XAUI, GIGE, Serial RapidIO, and Basic single-width
- Programmable run length violation detection in all functional modes
- Receiver polarity inversion in all functional modes except PCIe
- Receiver bit reversal in Basic single-width and 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 1-49 shows the word aligner operation in all supported configurations.

Figure 1-49. 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 lists 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_ala2size` 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_enaatternalign` 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_enaatternalign` 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.


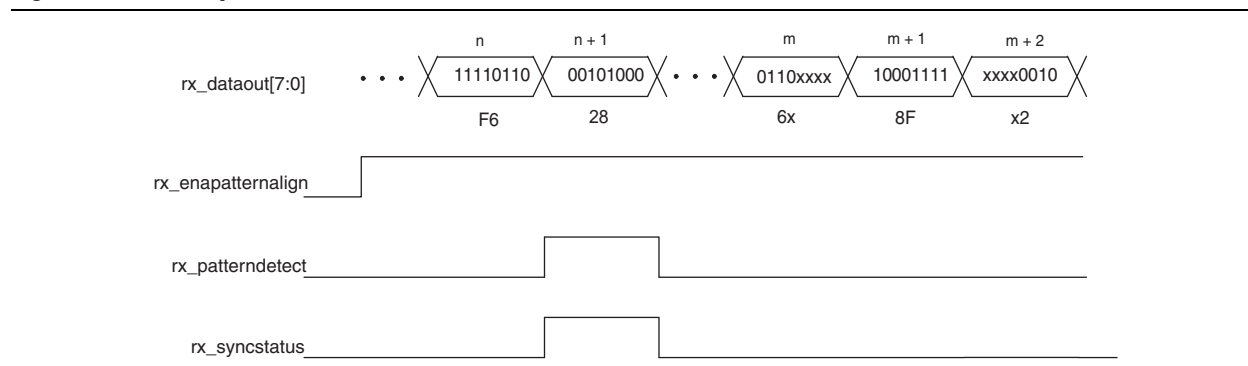
 For the word aligner to re-synchronize to a new word boundary, you must de-assert `rx_enaatternalign` and re-assert it again to create a rising edge. After a rising edge on the `rx_enaatternalign` signal, if the word alignment pattern is found in a different word boundary, the word aligner re-synchronizes to the new word boundary and asserts the `rx_syncstatus` and `rx_patterndetect` signals for one parallel clock cycle.

Figure 1-50 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_enaatternalign` signal. If you create a rising edge on the `rx_enaatternalign` 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 1-50. Bit-Slip Mode in 8-Bit PMA-PCS Interface Mode



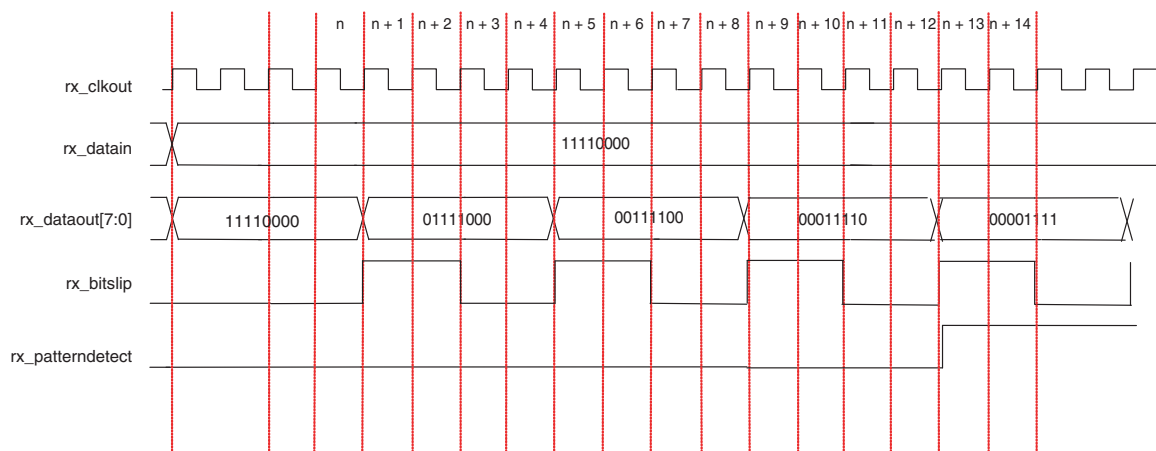
Bit-Slip Mode Word Aligner with 8-Bit PMA-PCS Interface Modes

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 1-51 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'b000011110` 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 1-51. 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:

- PCIe 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 lists 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
PCIe	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 PCIe, 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 PCIe, 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 lists the synchronization state machine parameters that the Quartus II software allows in supported functional modes. The synchronization state machine parameters are fixed for PCIe, 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	PCIe	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

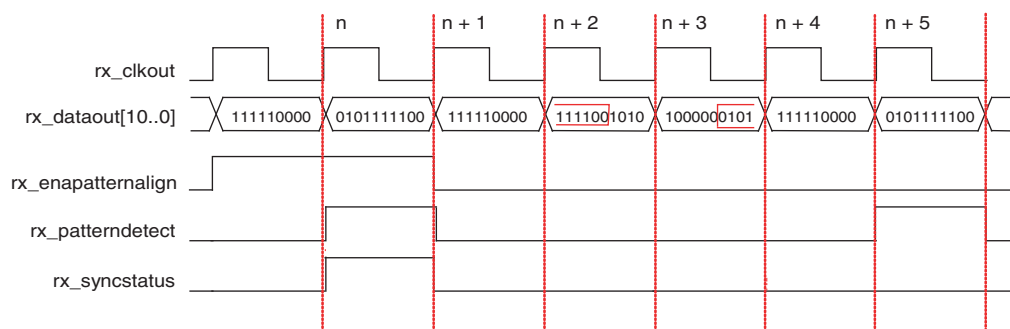
In Basic single-width mode with a 10-bit PMA-PCS interface, you can configure the word aligner in manual alignment mode by selecting the **Use manual word alignment mode** option in the ALTGX MegaWizard Plug-In Manager.


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 1-52 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 1-52. 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 1–60. 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](#) lists 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 ⁽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.

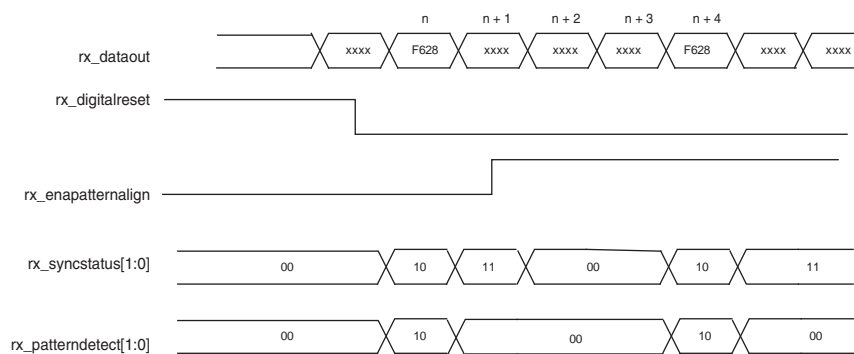
Two status signals, `rx_syncstatus` and `rx_patterndetect`, with the same latency as the datapath, are forwarded to the FPGA fabric to indicate word aligner status.

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 1-53 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 1-53. 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 1-60. 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](#) lists 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 1-66. 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 1-60. 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 lists 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 ⁽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 ⁽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 1–60 and “Word Aligner in Double-Width Mode” on page 1–66.

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 lists 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

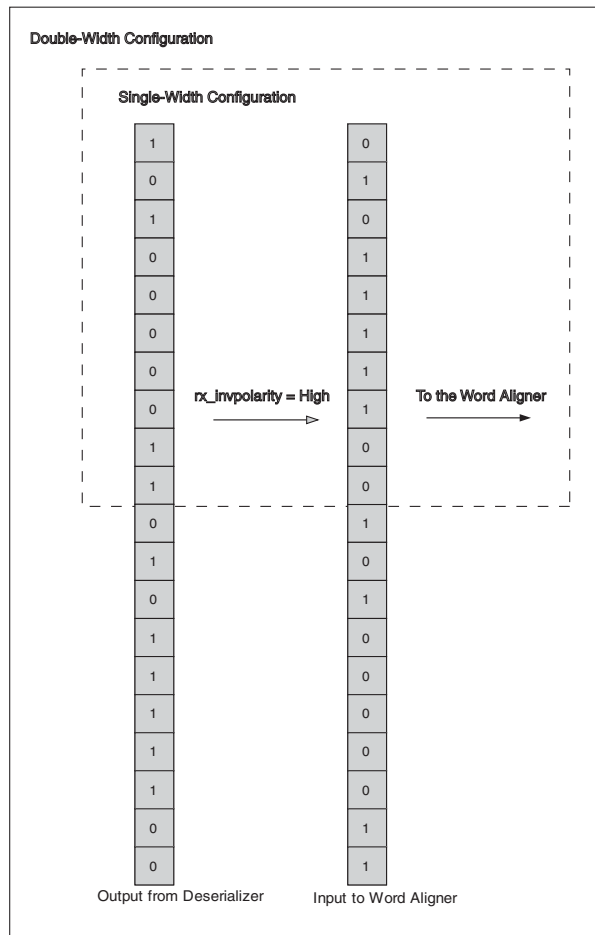
The positive and negative signals of a serial differential link are often erroneously swapped during board layout. Solutions like board re-spin or major updates to the PLD logic can be expensive. The receiver polarity inversion feature is provided to correct this situation.

An optional `rx_invpolarity` port is available in all single-width and double-width modes except (OIF) CEI PHY and PCIe 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 PCIe 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 PCIe mode. The PCIe 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 PCIe mode.

Figure 1-54 shows the receiver polarity inversion feature in single-width and double-width datapath configurations.

Figure 1-54. 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 1–55 shows the receiver bit reversal feature in Basic single-width 10-bit wide datapath configurations.

Figure 1–55. Receiver Bit Reversal in Single-Width Mode

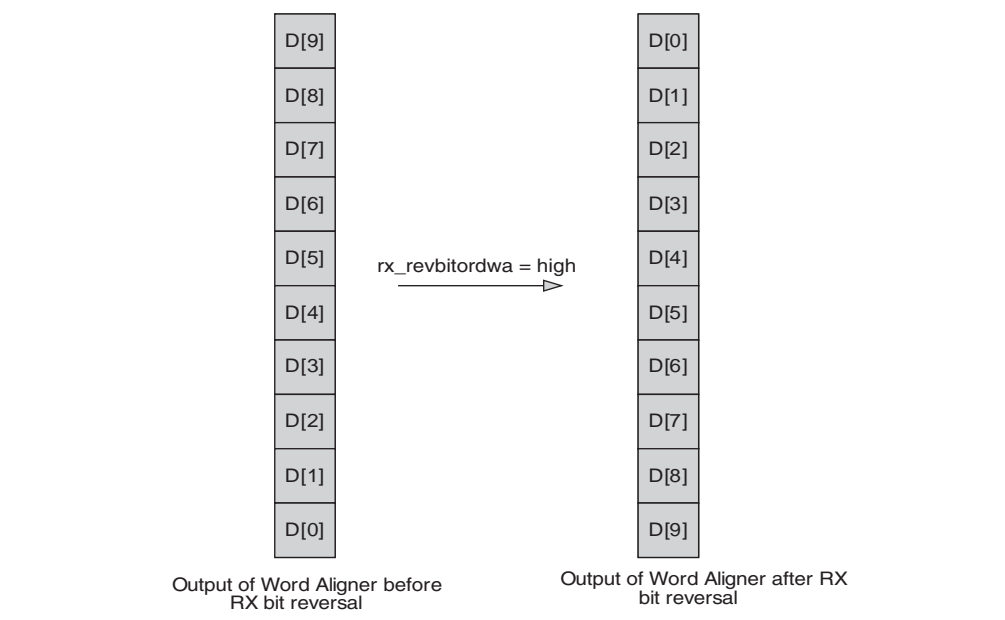
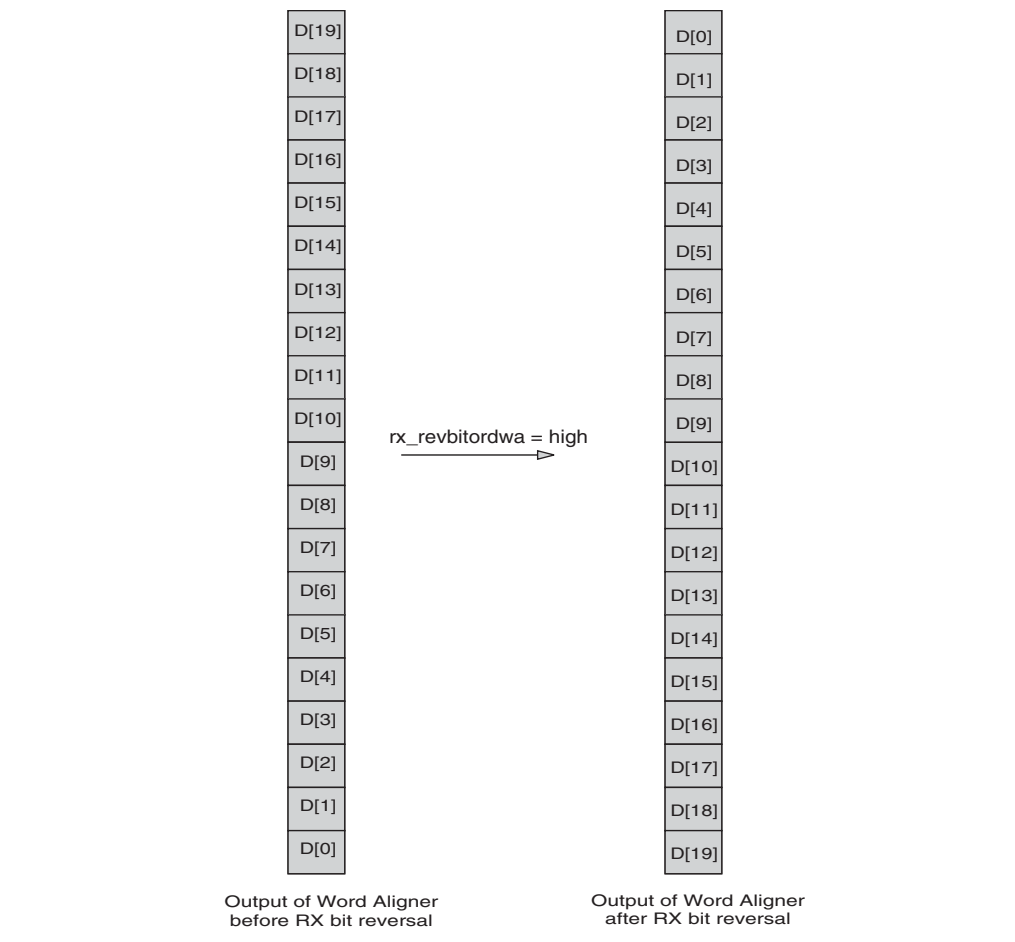


Figure 1–56 shows the receiver bit reversal feature in Basic double-width 20-bit wide datapath configurations.

Figure 1–56. 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.

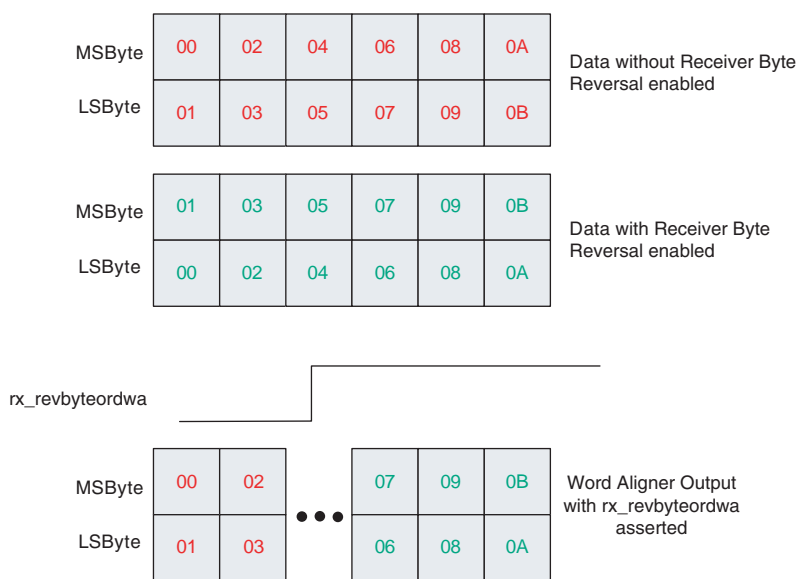
Receiver Byte Reversal in Basic Double-Width Modes

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 1-57 shows the receiver byte reversal feature.

Figure 1-57. 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

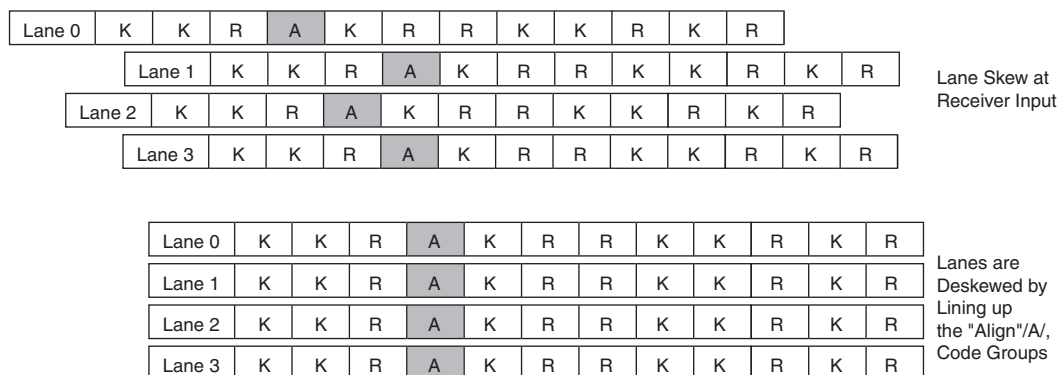


Deskew circuitry is only available in XAUI mode.

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 1-58 shows lane skew at the receiver input and how the deskew FIFO uses the /A/ code group to align the channels.

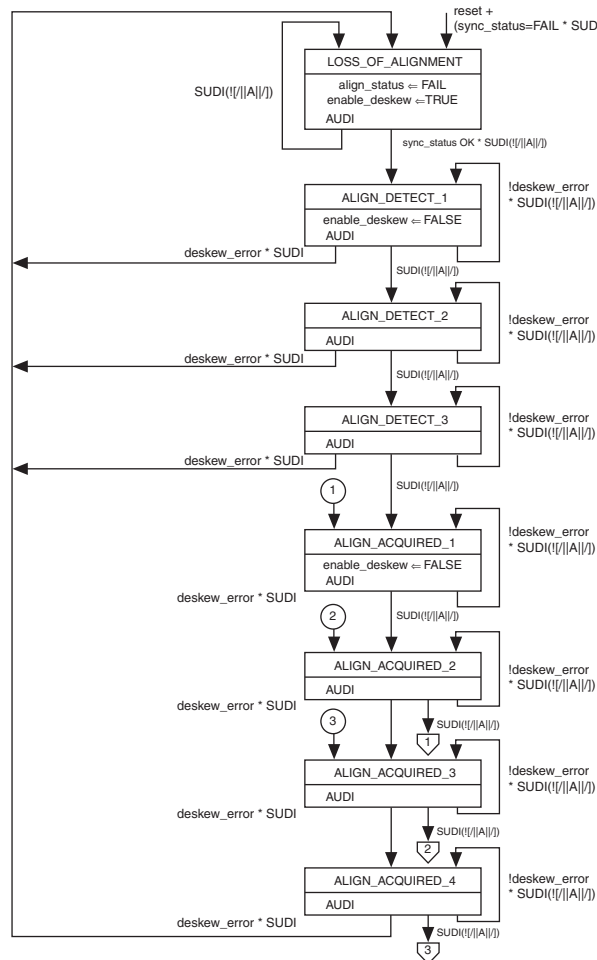
Figure 1-58. Deskew FIFO—Lane Skew at the Receiver Input



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 1-59.

Figure 1-59. Deskew FIFO Operation in XAUI Functional Mode (1)



Note to Figure 1-59:

(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:

- PCIe
- 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_rmifodatainserted`—indicates insertion of a skip character or ordered set
- `rx_rmifodatadeleted`—indicates deletion of a skip character or ordered set
- `rx_rmifofull`—indicates rate match FIFO full condition
- `rx_rmifoempty`—indicates rate match FIFO empty condition



The rate match FIFO status signals are not available in PCIe mode. These signals are encoded on the `pipestatus[2:0]` signal in PCIe mode as specified in the PCIe specification.

Rate Match FIFO in PCIe Mode

In PCIe 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 PCIe 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 PCIe 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 PCIe 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. 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 1-60 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 1-60. Rate Match Deletion in PCIe Mode

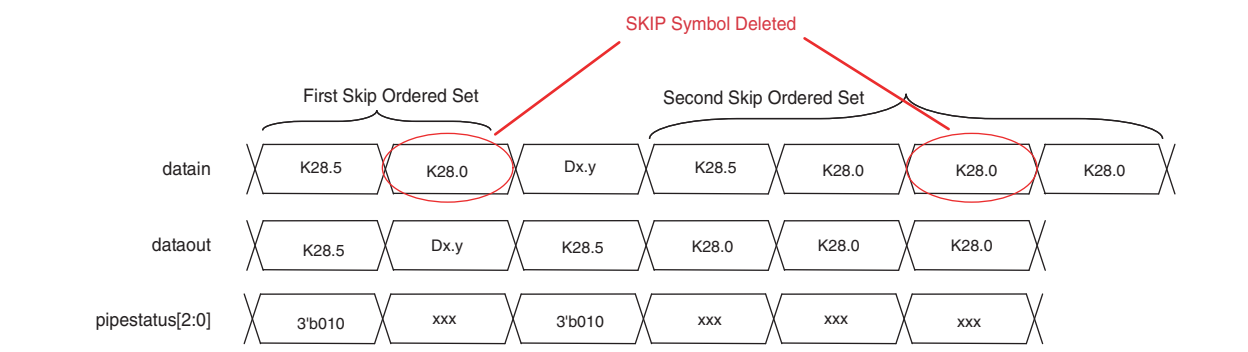
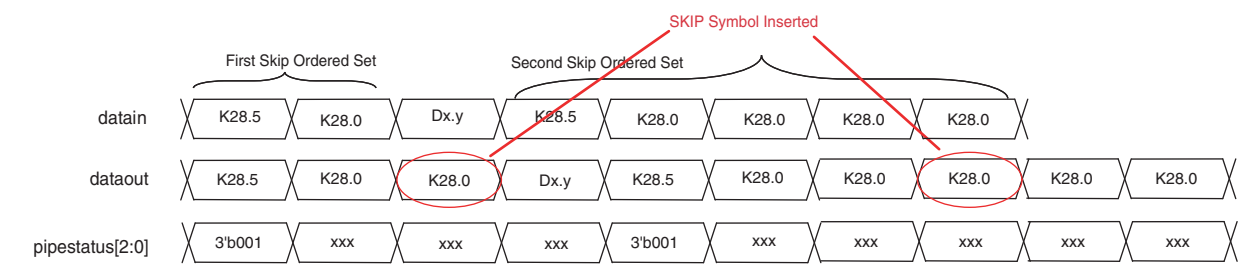


Figure 1-61 shows an example of rate match insertion in the case where two SKP symbols are required to be inserted. Only one `/K28.0/ SKP` symbol is inserted per SKP ordered set received.

Figure 1-61. Rate Match Insertion in PCIe Mode

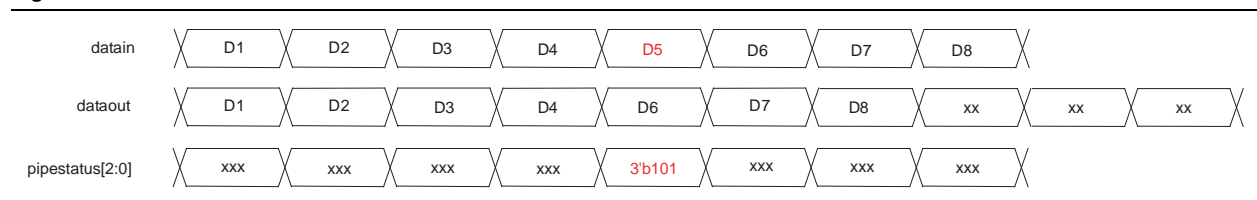


The rate match FIFO full and empty conditions are communicated to the FPGA fabric on the `pipestatus[2:0]` port from each channel.

The rate match FIFO in PCIe 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 1-62 shows the rate match FIFO full condition in PCIe mode. The rate match FIFO becomes full after receiving data byte D4.

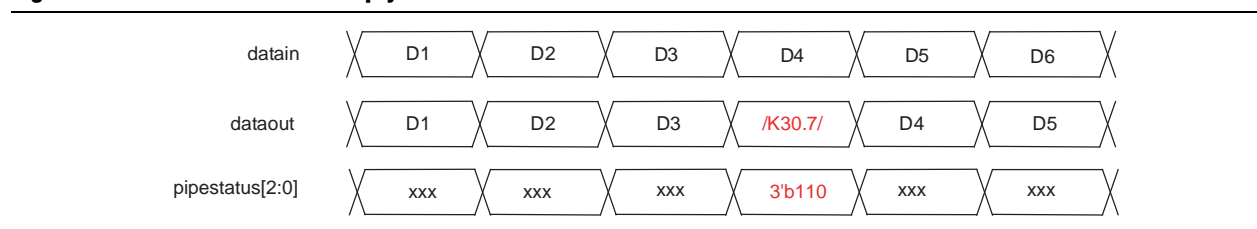
Figure 1-62. Rate Match FIFO Full Condition in PCIe Mode



The rate match FIFO automatically inserts `/K30.7/` (`9'h1FE`) after the data byte that causes the FIFO to go empty and drives `PCIestatus[2:0] = 3'b110` flag synchronous to the inserted `/K30.7/` (`9'h1FE`).

Figure 1-63 shows rate match FIFO empty condition in PCIe mode. The rate match FIFO becomes empty after reading out data byte D3.

Figure 1-63. Rate Match FIFO Empty Condition in PCIe 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_rmfifoatadeleted` and `rx_rmfifoatainserted`, that indicate rate match FIFO deletion and insertion events, respectively, are forwarded to the FPGA fabric. If an `||R||` column is deleted, the `rx_rmfiodeleted` 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_rmfiointerposed` flag from each of the four channels goes high for one clock cycle per inserted `||R||` column.

Figure 1-64 shows an example of rate match deletion in the case where three `||R||` columns must be deleted.

Figure 1-64. Rate Match Deletion in XAUI Mode

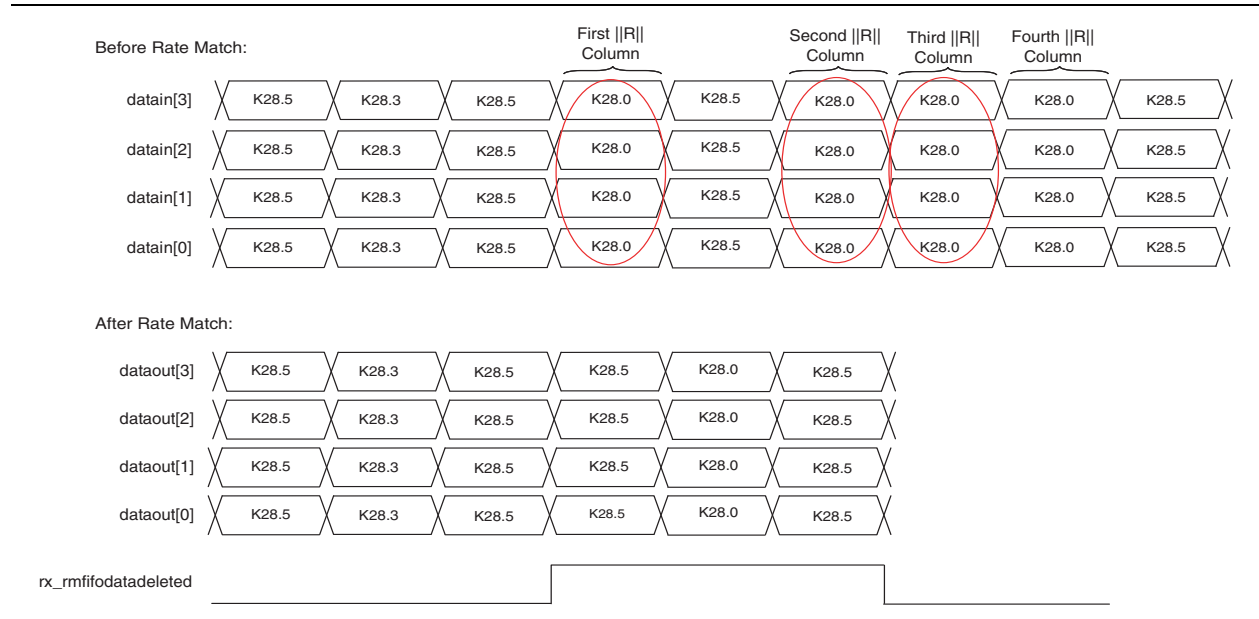
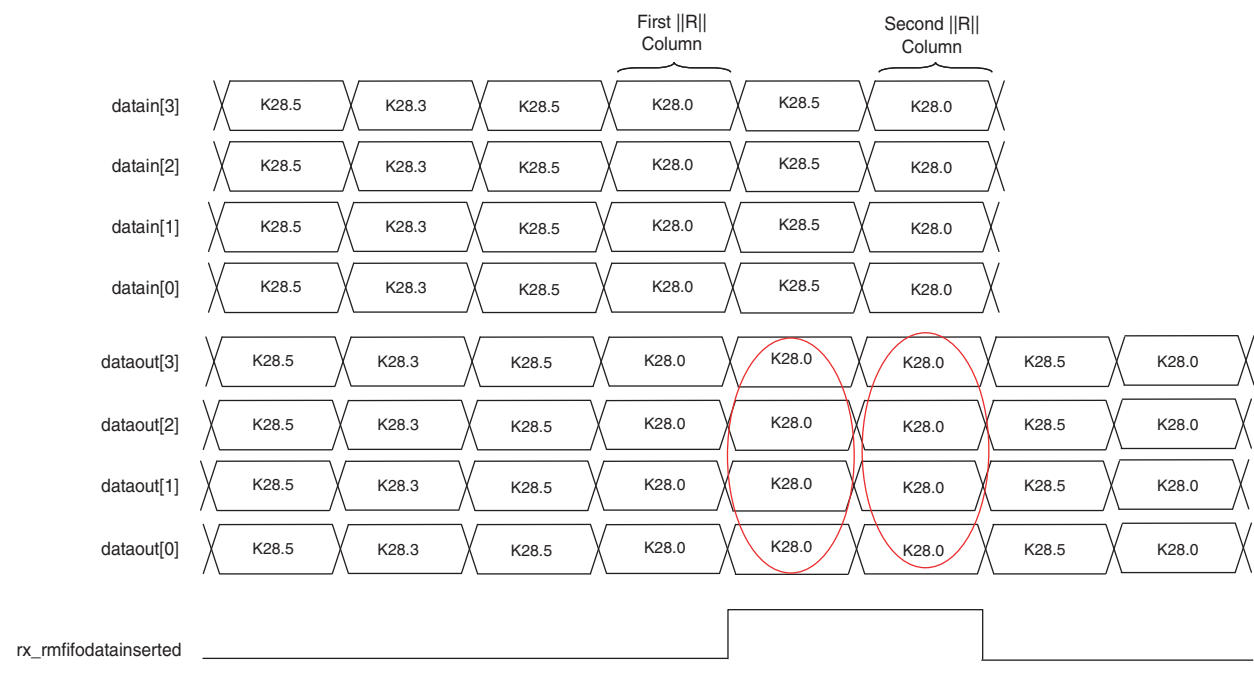


Figure 1-65 shows an example of rate match insertion in the case where two $||R||$ columns are required to be inserted.

Figure 1-65. Rate Match Insertion in XAUI Mode



Two flags, `rx_rmfifoempty` and `rx_rmfifooverflow`, 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_rmfifoempty` and `rx_rmfifooverflow` 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_rmifodatadeleted` and `rx_rmifodatainserted`, that indicate rate match FIFO deletion and insertion events, respectively, are forwarded to the FPGA fabric. Both the `rx_rmifodatadeleted` and `rx_rmifodatainserted` flags are asserted for two clock cycles for each deleted and inserted /I2/ ordered set, respectively.

Figure 1-66 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 1-66. Rate Match Deletion in GIGE Mode

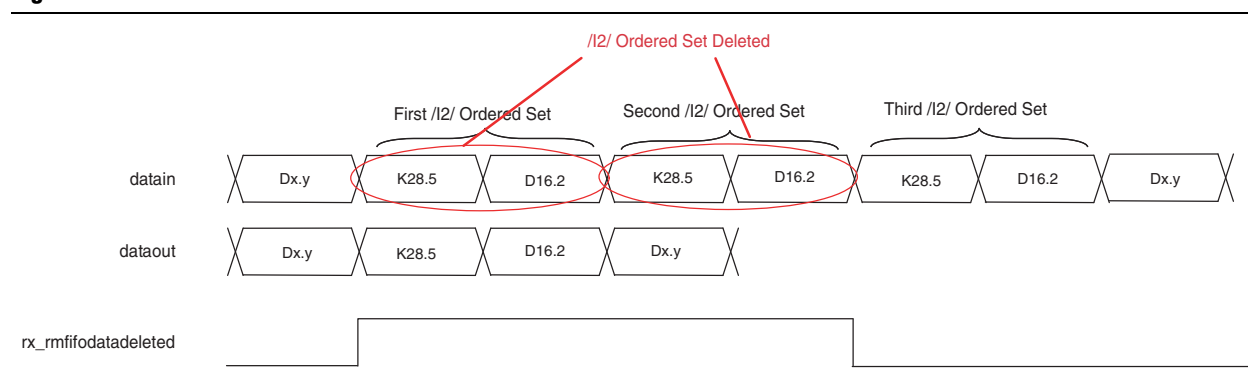
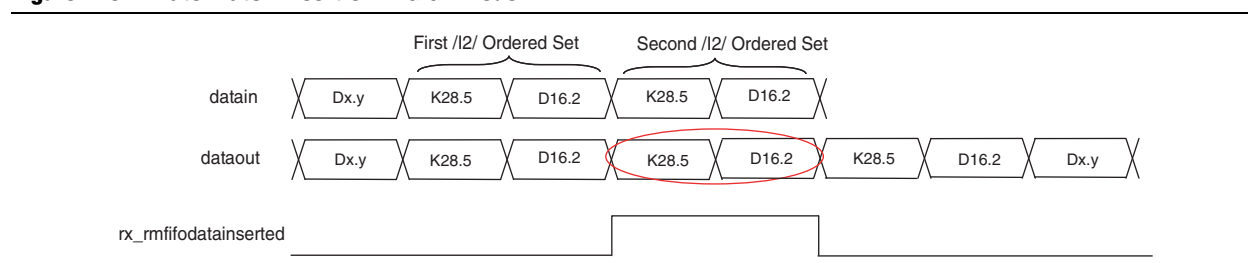



Figure 1-67 shows an example of rate match FIFO insertion in the case where one symbol is required to be inserted. Because the rate match FIFO can only insert a /I2/ ordered set, it inserts one /I2/ ordered set (two symbols inserted).

Figure 1-67. Rate Match Insertion in GIGE Mode




Two flags, `rx_rmifofull` and `rx_rmifoempty`, 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_rmifofull` and `rx_rmifoempty` 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_rmifodatadeleted` and `rx_rmifodatainserted`, 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 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 1-68. Rate Match Deletion in Basic Single-Width Mode

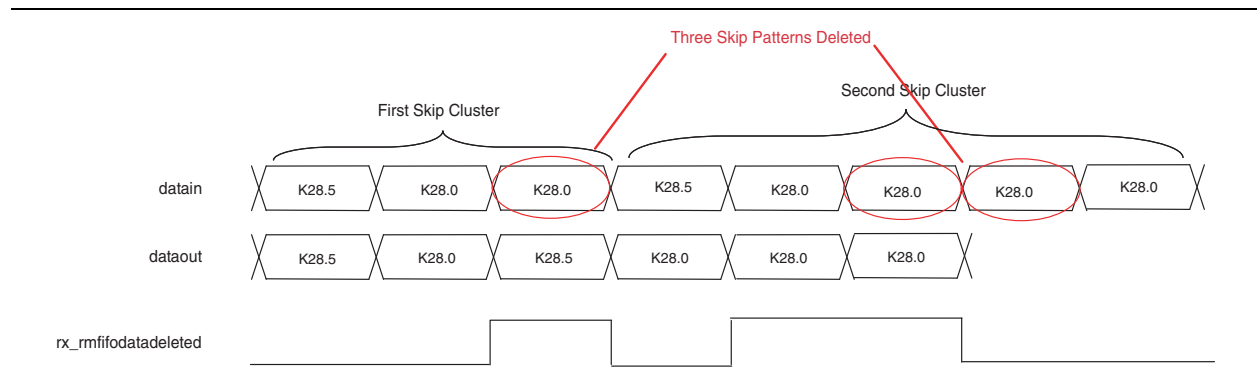
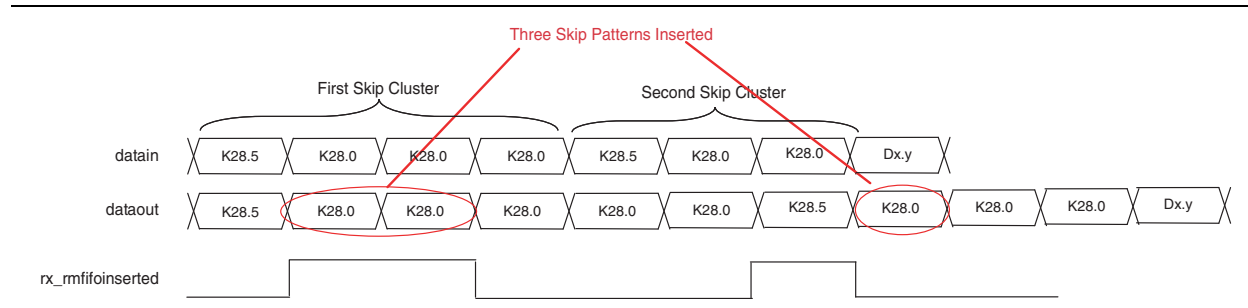


Figure 1-69 shows an example of rate match FIFO insertion in the case where three skip patterns are required to be inserted. In this example, /K28.5/ is the control pattern and neutral disparity /K28.0/ is the skip pattern. The first skip cluster has a /K28.5/ control pattern followed by three /K28.0/ 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 1-69. Rate Match Insertion in Basic Single-Width Mode

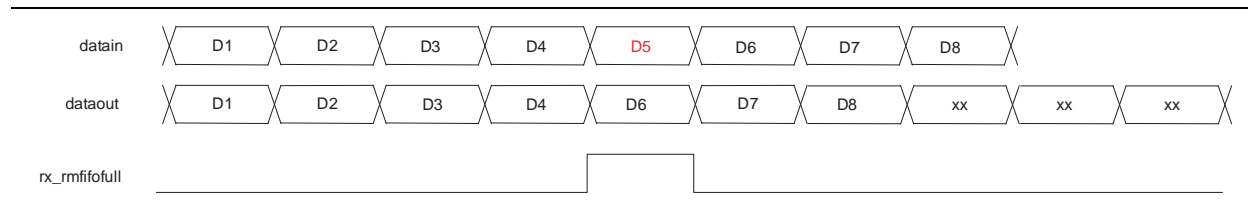


Two flags, `rx_rmffifo`full and `rx_rmffifo`empty, are forwarded to the FPGA fabric to indicate rate match FIFO full and empty conditions.

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_rmffifo`full flag synchronous to the subsequent data byte.

Figure 1-70 shows the rate match FIFO full condition in Basic single-width mode. The rate match FIFO becomes full after receiving data byte D4.

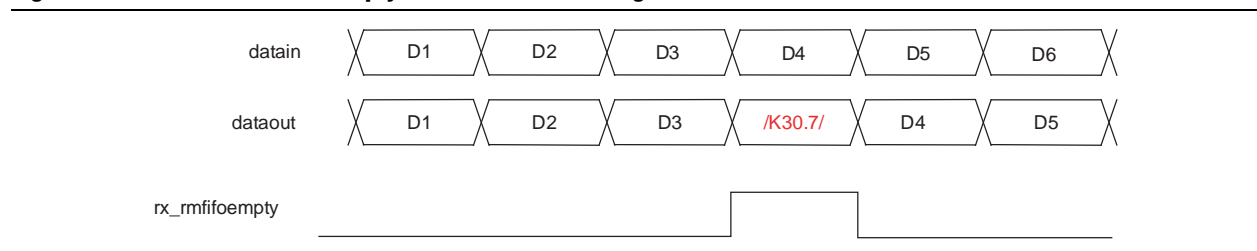
Figure 1-70. Rate Match FIFO Full Condition in Basic Single-Width Mode



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_fifo`empty flag synchronous to the inserted /K30.7/ (9'h1FE).


Figure 1-71 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 1-71. 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_rmfiifodatadeleted` and `rx_rmfiifodatainserted`, indicating rate match FIFO deletion and insertion events, respectively, are forwarded to the FPGA fabric.

Figure 1-72 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 1-72. Rate Match Deletion in Basic Double-Width Mode

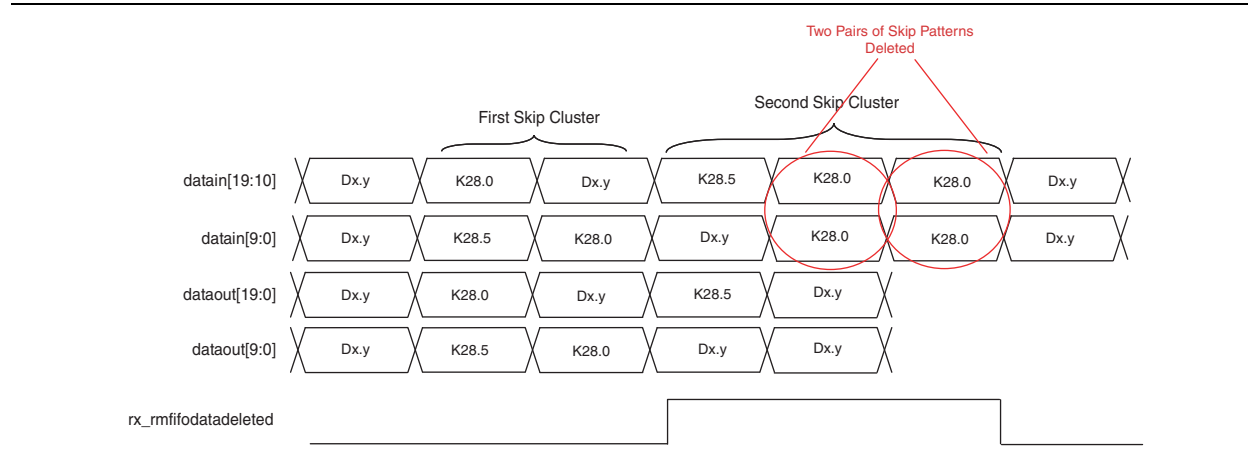
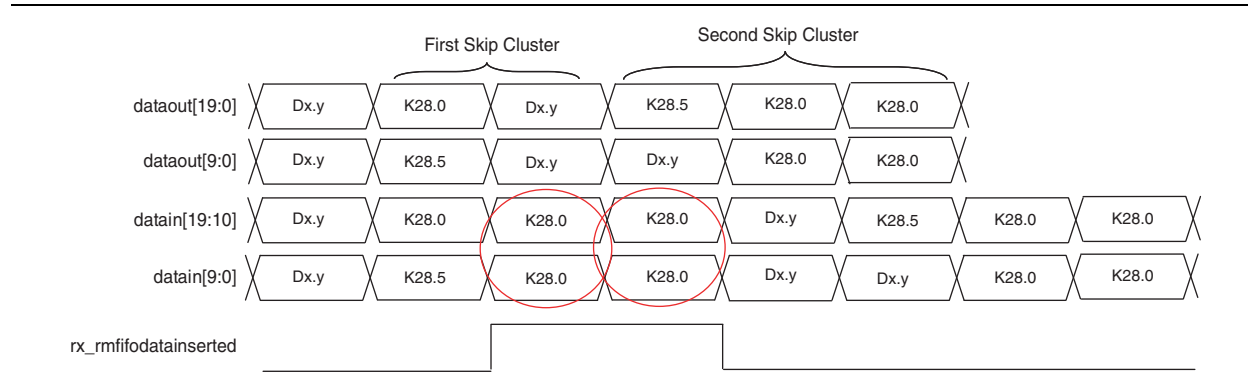


Figure 1-73 shows an example of rate match FIFO insertion in the case where three skip patterns are required to be inserted. In this example, /K28.5/ is the control pattern and neutral disparity /K28.0/ is the skip pattern. The first skip cluster has a /K28.5/ control pattern in the LSByte and /K28.0/ 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 1-73. Rate Match Insertion in Basic Double-Width Mode

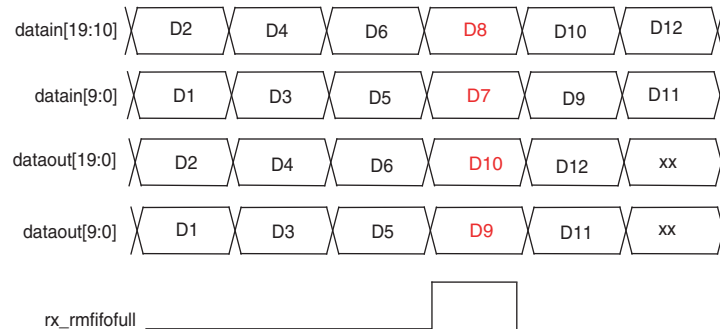


Two flags, `rx_rmfifoempty` and `rx_rmfifoempty`, are forwarded to the FPGA fabric to indicate rate match FIFO full and empty conditions.

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_rmfifoempty` flag synchronous to the subsequent pair of data bytes.

Figure 1-74 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.

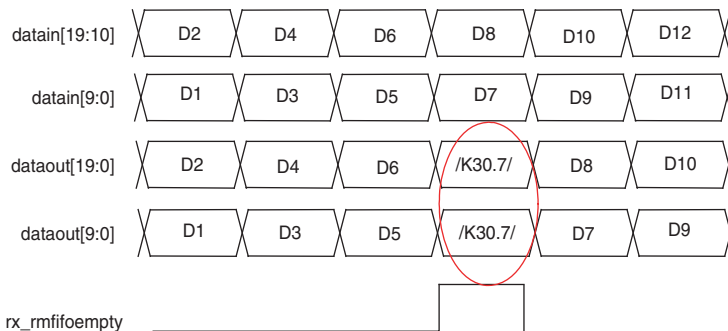
Figure 1-74. Rate Match FIFO Full Condition in Basic Double-Width Mode



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_rmfifoempty` flag synchronous to the inserted pair of `/K30.7/` (`{9'h1FE,9'h1FE}`).

Figure 1-75 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 1-75. Rate Match FIFO Empty Condition in Basic Double-Width Mode



If you enabled the auto-negotiation state machine in the FPGA core with the rate match FIFO, refer to “Rate Match FIFO” on page 1-171.

8B/10B Decoder

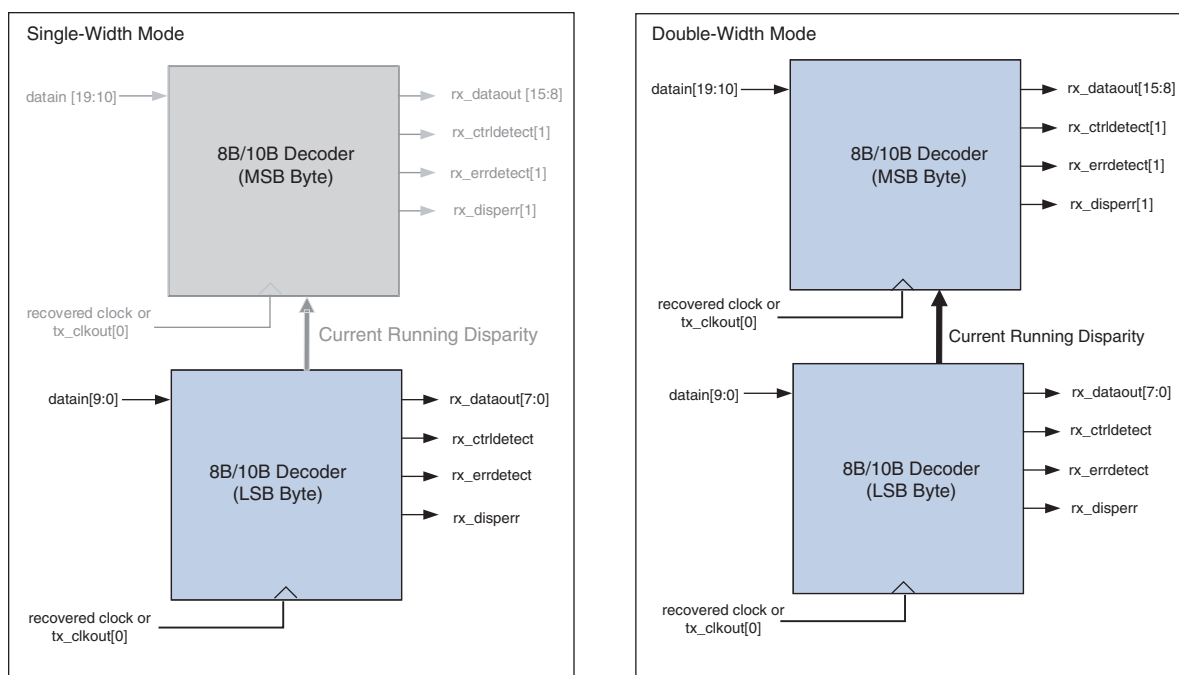
Protocols such as PCIe, XAUI, GIGE, and Serial RapidIO require the serial data sent over the link to be 8B/10B encoded to maintain the DC balance in the serial data transmitted. These protocols require the receiver PCS logic to implement an 8B/10B decoder to decode the data before forwarding it to the 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 1-76):


- Single-width mode
- Double-width mode

Figure 1-76. 8B/10B Decoder in Single-Width and Double-Width Mode



8B/10B Decoder in Single-Width Mode

The left side of Figure 1-76 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.

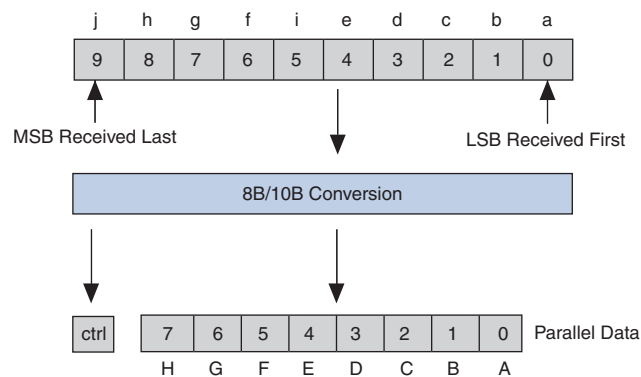
The 8B/10B decoder operates in single-width mode in the following functional modes:

- PCIe
- XAUI
- GIGE
- Serial RapidIO
- Basic single-width

For PCIe, 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 1-77 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 1-77. 8B/10B Decoder in Single-Width Mode

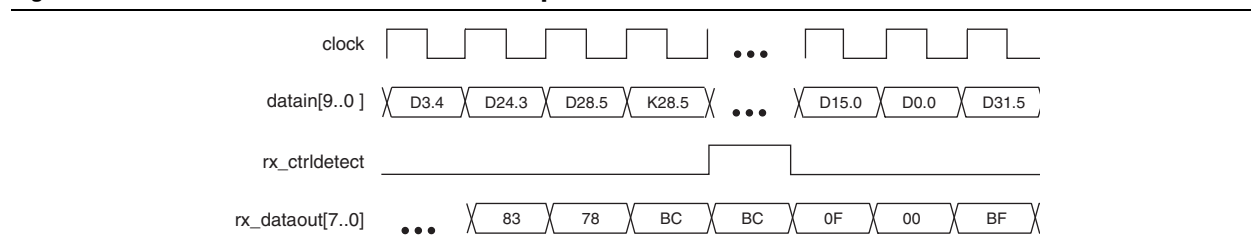


Control Code Group Detection

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 1-78 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_ctrlldetect 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 1-78. 8B/10B Decoder in Control Code Group Detection



8B/10B Decoder in Double-Width Mode

The left side of Figure 1-76 on page 1-89 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 1-79. The 10-bit LSB 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 LSB 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).

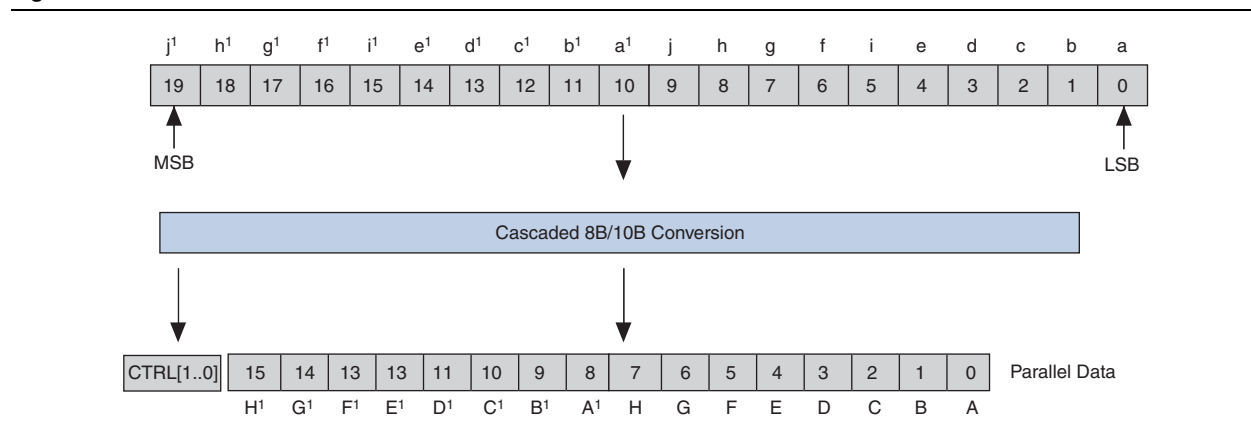


Each of the two cascaded 8B/10B decoders is compliant to Clause 36 in the IEEE802.3 specification.

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 1-79 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 1-79. 8B/10 Decoder in 20-Bit Double-Width Mode

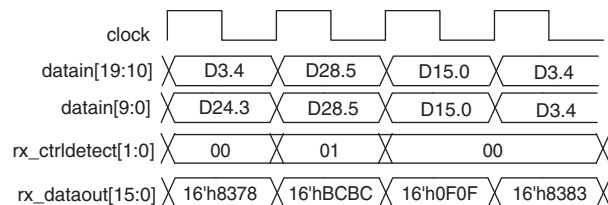


Control Code Group Detection

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 1–80 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 1–80. 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:

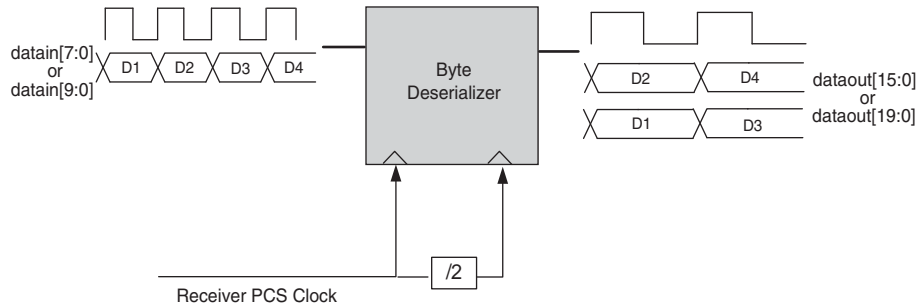
- Single-width mode
- Double-width mode

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 1-81 shows the byte deserializer in single-width mode.

Figure 1-81. Byte Deserializer in Single-Width Mode

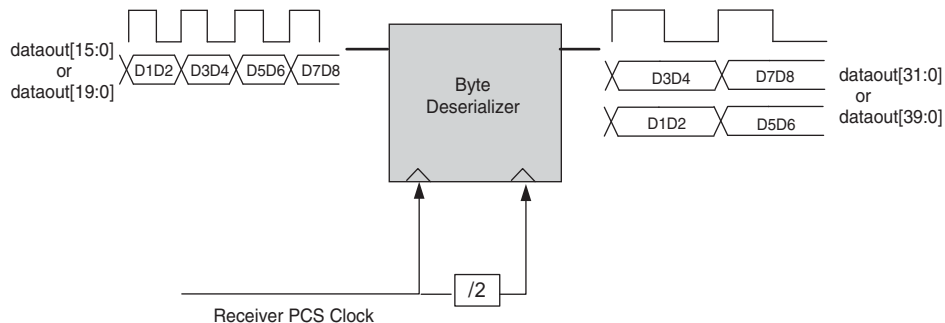


Byte Deserializer in Double-Width Mode

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 1-82 shows the byte deserializer in double-width mode.

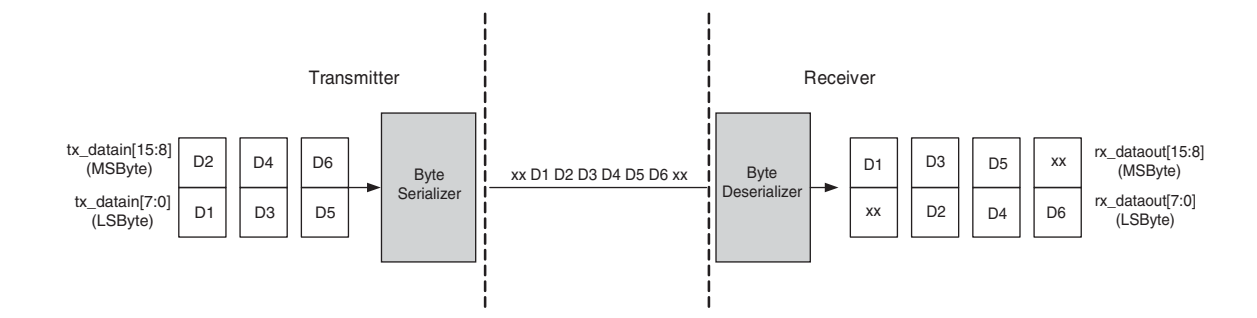
Figure 1-82. Byte Deserializer in Double-Width Mode



Byte Ordering Block

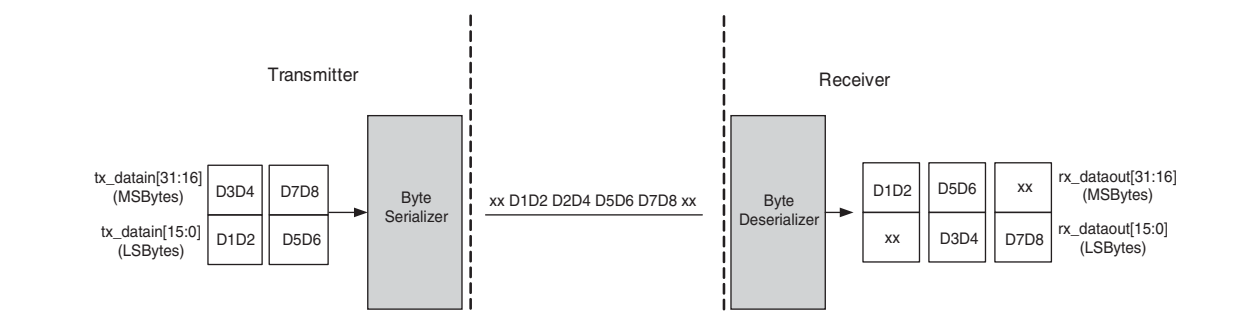
In single-width modes with the 16-bit or 20-bit FPGA fabric-transceiver interface, the byte deserializer receives one data byte (8 or 10 bits) and deserializes it into two data bytes (16 or 20 bits). Depending on when the receiver PCS logic comes out of reset, the byte ordering at the output of the byte 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 1-83 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 1-83. MSByte and LSByte of the Two-Byte 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 1-84 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 1-84. MSByte and LSByte of the Four-Byte 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:	<ul style="list-style-type: none"> ■ 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:	<ul style="list-style-type: none"> ■ 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 1-113.

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 1-177.

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 lists 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: <ul style="list-style-type: none"> ■ 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: <ul style="list-style-type: none"> ■ 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:	<ul style="list-style-type: none"> ■ 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:	<ul style="list-style-type: none"> ■ 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:	<ul style="list-style-type: none"> ■ 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 1-117.

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 lists 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: <ul style="list-style-type: none"> ■ 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: <ul style="list-style-type: none"> ■ 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: <ul style="list-style-type: none"> ■ 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 $rx_ctrlrdetect[1]$ and $D[16:9]$ corresponds to $rx_dataout[15:8]$. Similarly, $D[8]$ corresponds to $rx_ctrlrdetect[0]$ 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.


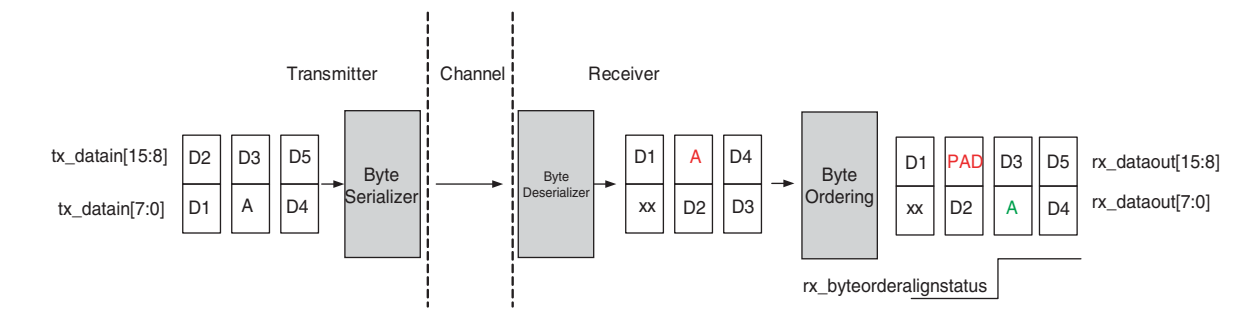
 You can choose word-alignment-based byte ordering in the **Rate match/Byte order** tab of the ALTGX MegaWizard Plug-In Manager. For the **What do you want the byte ordering to be based on?** question, select the **The sync status signal from the word aligner** option.

Figure 1-85 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 1-85. Byte Ordering in Single-Width Modes

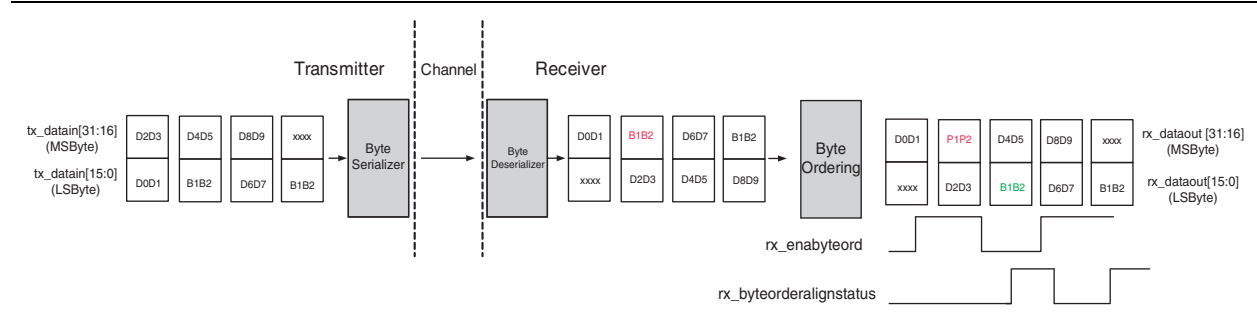


User-Controlled Byte Ordering

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 1-86 shows user-controlled byte ordering in Basic double-width Mode.

Figure 1-86. 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 lists 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 (<code>tx_clkout</code>)	Divide-by-two version of the parallel transmitter PCS clock from the local clock divider in the associated channel (<code>tx_clkout</code>)
Non-bonded channel configuration without rate matcher	Parallel recovered clock from the receiver PMA in the associated channel (<code>rx_clkout</code>)	Divide-by-two version of the parallel recovered clock from the receiver PMA in the associated channel (<code>rx_clkout</code>)
×4 bonded channel configuration	Parallel transmitter PCS clock from the central clock divider in the <code>CMU0</code> of the associated transceiver block (<code>coreclkout</code>)	Divide-by-two version of the parallel transmitter PCS clock from the central clock divider in <code>CMU0</code> of the associated transceiver block (<code>coreclkout</code>)
×8 bonded channel configuration	Parallel transmitter PCS clock from the central clock divider in <code>CMU0</code> of the master transceiver block (<code>coreclkout</code> from master transceiver block)	Divide-by-two version of the parallel transmitter PCS clock from the central clock divider in <code>CMU0</code> of the master transceiver block (<code>coreclkout</code> 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 lists 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	
	<code>rx_coreclk</code> Port Not Instantiated	<code>rx_coreclk</code> Port Instantiated ⁽¹⁾
Non-bonded channel configuration with rate matcher	FPGA fabric clock driven by the clock signal on the <code>tx_clkout</code> port	FPGA fabric clock driven by the clock signal on the <code>rx_coreclk</code> port
Non-bonded channel configuration without rate matcher	FPGA fabric clock driven by the clock signal on the <code>rx_clkout</code> port	FPGA fabric clock driven by the clock signal on the <code>rx_coreclk</code> port
×4 bonded channel configuration	FPGA fabric clock driven by the clock signal on the <code>coreclkout</code> port	FPGA fabric clock driven by the clock signal on the <code>rx_coreclk</code> port
×8 bonded channel configuration	FPGA fabric clock driven by the clock signal on the <code>coreclkout</code> port	FPGA fabric clock driven by the clock signal on the <code>rx_coreclk</code> 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

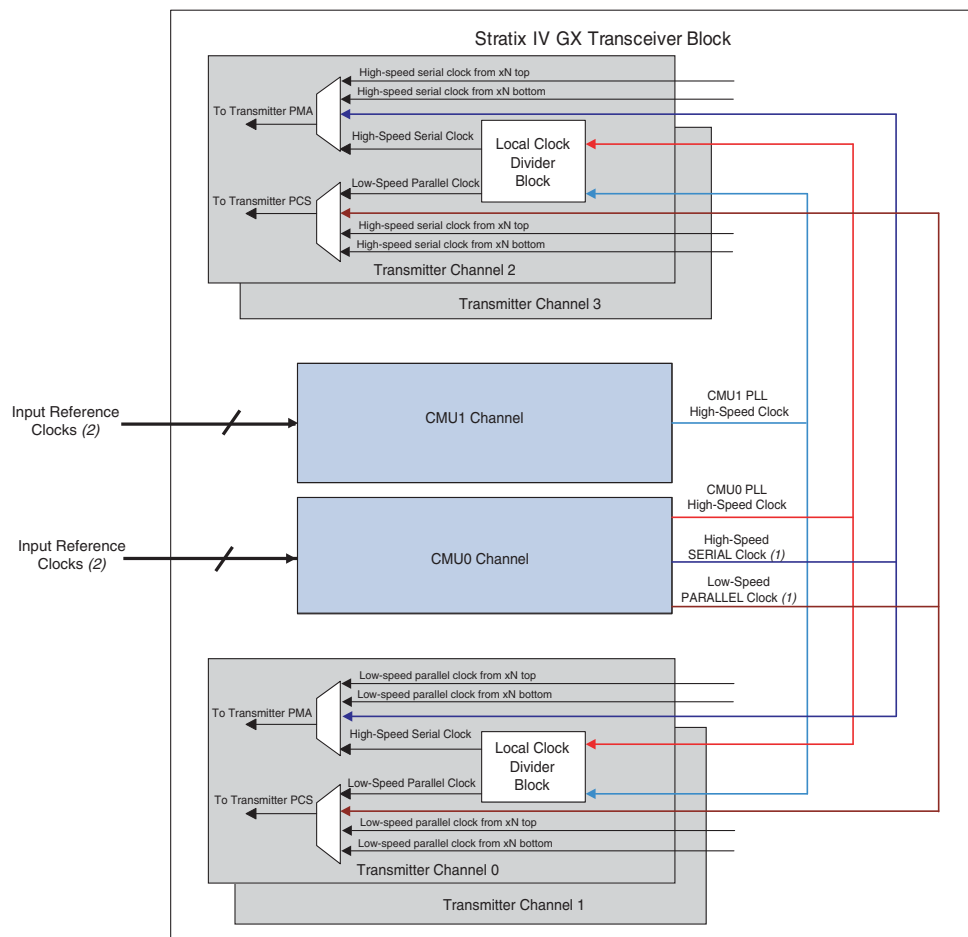
An optional `rx_phase_comp_fifo_error` port is available in all functional modes to indicate a receiver phase compensation FIFO under-run or overflow condition. The `rx_phase_comp_fifo_error` signal is asserted high when the phase compensation FIFO gets either full or empty. This feature is useful to 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 1-87 shows the two CMU channels in a transceiver block.

Figure 1-87. CMU Channels in a Transceiver Block



Notes to Figure 1-87:


- (1) Clocks are provided to support bonded channel functional mode.
- (2) For more information, refer to the *Transceiver Clocking in Stratix IV Devices* 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 $\times 4$, $\times 8$, and $\times N$ bonded mode configurations such as XAUI, Basic $\times 4$, Basic $\times 8$, and Basic (PMA-Direct) $\times N$.

The CMU0 channel has additional capabilities to support bonded protocol functional modes such as Basic $\times 4$, XAUI, and PCIe. Use the ALTGX MegaWizard Plug-In Manager to select these functional modes (to enable Basic $\times 4$ functional mode, select the $\times 4$ option in Basic mode). For more information, refer to “Functional Modes” on page 1-110.

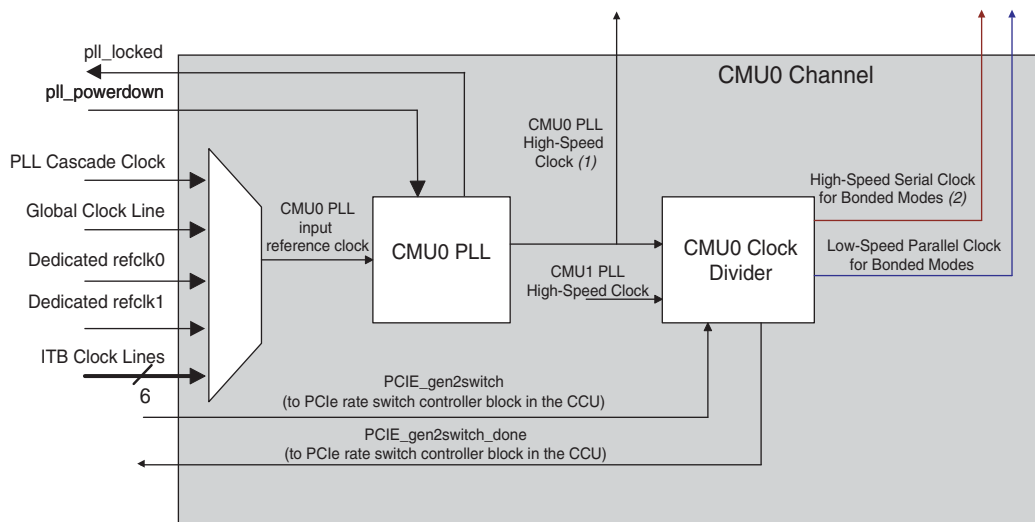
 For Stratix IV GT devices, you can use the CMU PLL to generate transceiver clocks at data rates between 600 Mbps and 11.3 Gbps.

CMU0 Channel

The CMU0 channel, shown in Figure 1-88, contains the following blocks:

- CMU0 PLL (refer to “CMU0 Clock Divider” on page 1-103)
- CMU0 clock divider (refer to “CMU Clock Divider” on page 1-108)

Figure 1-88. CMU0 Channel with the CMU0 PLL and CMU0 Clock Divider



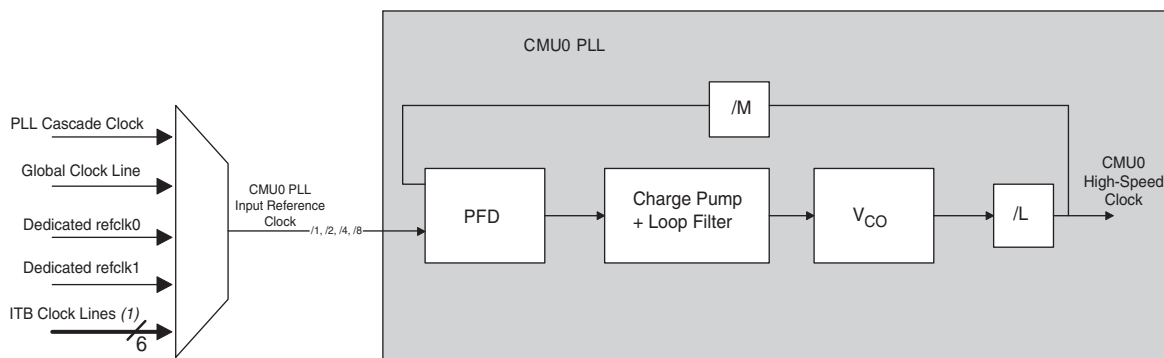
Notes to Figure 1-88:

- (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 $\times 4$, and PCIe $\times 4$ functional modes. In PCIe $\times 8$ functional mode, only the CMU0 channel of the master transceiver block provides clock output to all eight transceiver channels configured in PCIe functional mode.

CMU0 PLL

Figure 1-89 shows the CMU0 PLL.

Figure 1-89. CMU0 PLL




Note to Figure 1-89:

- (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 *Transceiver Clocking in Stratix IV Devices* 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 1-89).

 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 `p11_locked` signal that is asserted to indicate that the CMU PLL is locked to the input reference clock. You can use the `p11_locked` signal in your transceiver reset sequence, as described in the *Reset Control and Power Down in Stratix IV Devices* 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 `p11_powerdown` signal.

- For more information, refer to the *Reset Control and Power Down in Stratix IV Devices* chapter.

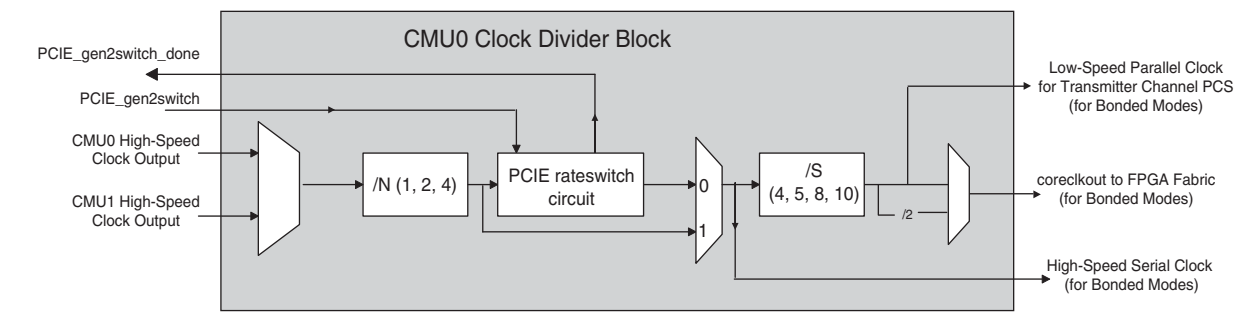
CMU0 Clock Divider

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 *Transceiver Clocking in Stratix IV Devices* chapter.

You can configure the CMU0 clock divider shown in Figure 1-90 to select the high-speed clock output from the CMU0 or CMU1 PLLs. The CMU1 PLL is present in the CMU1 channel.

Figure 1-90. 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 $\times 4/\times 8$, XAUI, and PCIe $\times 4/\times 8$ configurations. In XAUI and Basic $\times 4/\times 8$ 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 PCIe $\times 4$ mode, the clock path through the PCIe rateswitch circuit block is selected. This high-speed serial clock is provided to all the transmitter channels.
- In PCIe $\times 8$ mode and Basic $\times 8$ mode, only the CMU0 clock divider of the master transceiver block provides the high-speed serial clock to all eight channels.
- In PCIe $\times 1$ 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 *Transceiver Clocking in Stratix IV Devices* chapter.

PCIe Rateswitch Circuit

The PCIe rateswitch circuit is enabled only in PCIe $\times 4$ mode. In PCIe $\times 8$ 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.

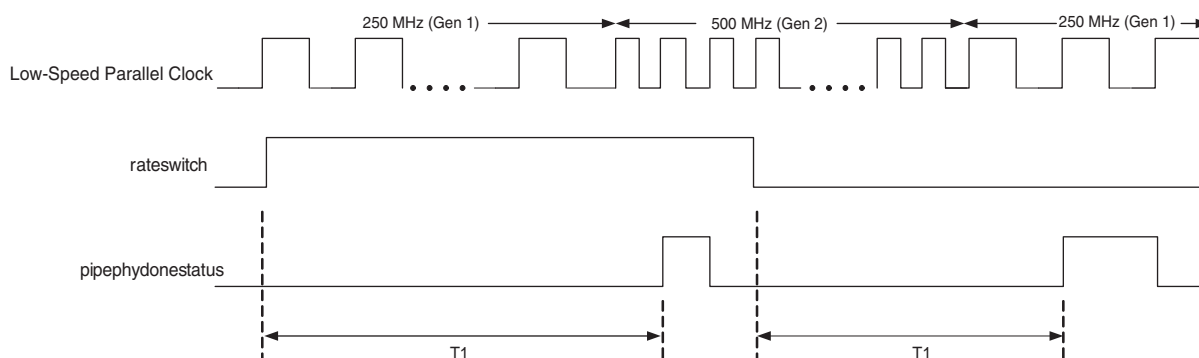
- When you set the `rateswitch` port to 0, the PCIe rateswitch controller (in the CCU) signals the PCIe rateswitch circuit to select the divide by /2 to provide a high-speed serial clock for the Gen1 (2.5 Gbps) data rate.
- When you set the `rateswitch` port to 1, the /N divider output is forwarded, 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 PCIe rateswitch controller in the CCU. The PCIe rateswitch controller asserts the `pipephydonestatus` signal for one clock cycle after the rateswitch operation is completed for both the transmit and receive channels. [Figure 1-91](#) shows the timing diagram for the rateswitch operation.


For more information about PCIe functional mode rate switching, refer to “[PCIe Gen2 \(5 Gbps\) Support](#)” on page 1-140.

Figure 1-91. Rateswitch in PCIe Mode ⁽¹⁾




Note to Figure 1-91:

(1) Time T1 is pending characterization.

 When creating a PCIe 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

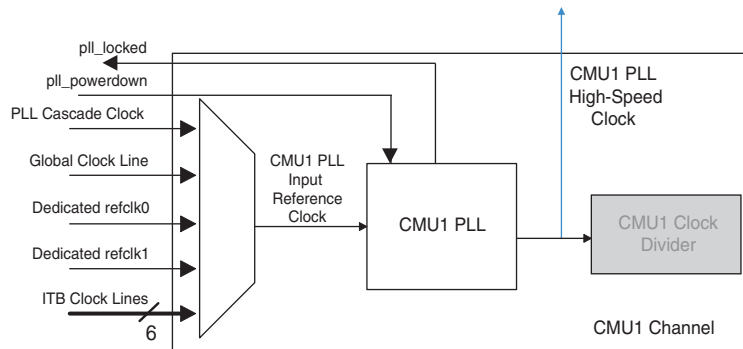
The /S divider receives the clock output from the /N divider or PCIE rateswitch circuit (only in PCIe mode) and generates the low-speed parallel clock for the PCS block of all transmitter channels and `coreclkout` for the FPGA fabric. If the byte serializer block is enabled in bonded channel modes, the /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 1-17.

 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 1-92](#), 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 1-102).

Figure 1-92. 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 in Stratix IV Devices](#) chapter.

Power Down CMU1 PLL

You can power down the CMU1 PLL by asserting the `pll_powerdown` signal.

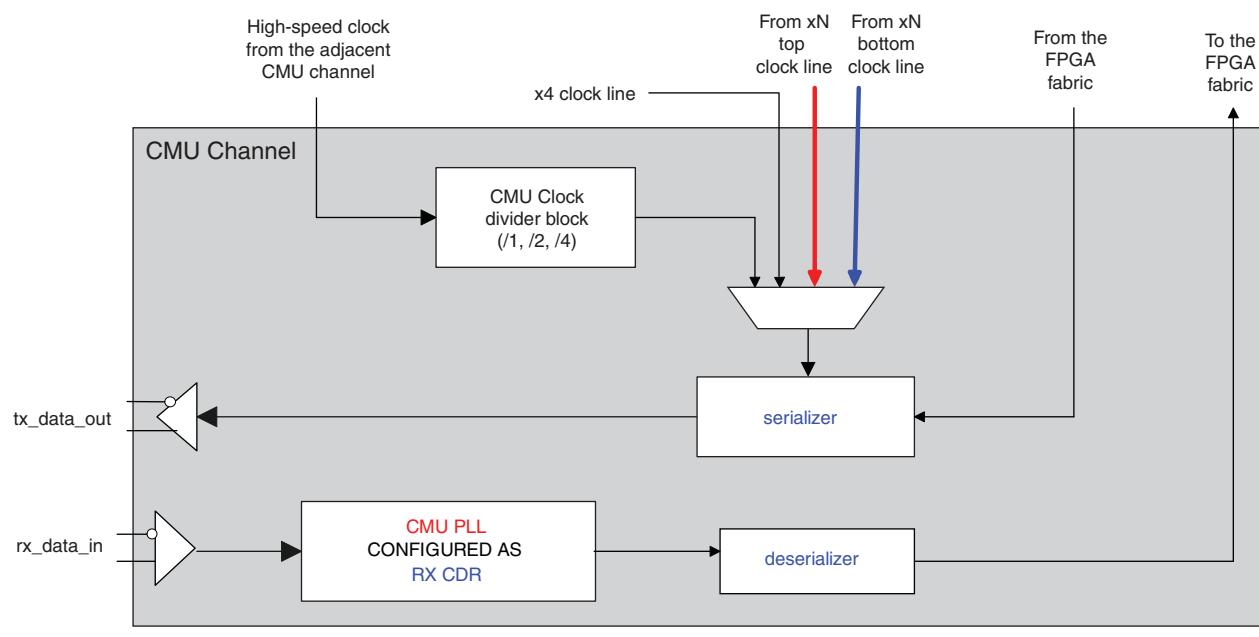
- For more information, refer to the [Reset Control and Power Down in Stratix IV Devices](#) chapter.


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.

Figure 1-93 shows the functional blocks that are enabled to support the transceiver channel functionality.

Figure 1-93. 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 lists 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 *Overview for the Stratix IV Device Family* 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 $\times N$ 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 *Transceiver Clocking in Stratix IV Devices* chapter.

CMU Clock Divider

When you configure a CMU channel in Basic (PMA Direct) $\times 1$ 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 $\times N$ clock line (the $\times N_{Top}$ or $\times N_{Bottom}$ clock line). If you configure a CMU channel in Basic (PMA Direct) $\times N$ mode, you can use this clocking option
- From one of the ATX PLL blocks on the same side of the device through the $\times N$ clock line (the $\times N_{Top}$ or $\times N_{Bottom}$ 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 1-18](#).



For more information, refer to the “Input Reference Clocking” section of the [Transceiver Clocking in Stratix IV Devices](#) 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 1-34](#) and [“Receiver Input Buffer” on page 1-40](#).

For information about loopback, refer to [“Loopback Modes” on page 1-190](#).

Dynamic Reconfiguration of the CMU Channel Analog Controls

 For the dynamic reconfiguration capabilities of the CMU channels in Basic (PMA Direct) $\times 1/\times N$ configurations, refer to the *Dynamic Reconfiguration in Stratix IV Devices* chapter.

Functional Modes

Table 1-40 lists 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 1-113
Basic Double Width	1 Gbps to 8.5 Gbps	"Basic Double-Width Mode Configurations" on page 1-117
PCIe	<ul style="list-style-type: none"> ■ Gen1 at 2.5 Gbps ■ Gen2 at 5 Gbps 	"PCIe Mode Configurations" on page 1-128
XAUI	3.125 Gbps up to HiGig at 3.75 Gbps	"XAUI Mode Datapath" on page 1-157
GIGE	1.25 Gbps	"GIGE Mode Datapath" on page 1-167
Serial RapidIO	<ul style="list-style-type: none"> ■ 1.25 Gbps ■ 2.5 Gbps ■ 3.125 Gbps 	"Serial RapidIO Mode" on page 1-182
SONET/SDH	<ul style="list-style-type: none"> ■ OC-12 ■ OC-48 	"SONET/SDH Frame Structure" on page 1-172
SDI	<ul style="list-style-type: none"> ■ HD at 1.485/1.4835 Gbps ■ 3G at 2.97/2.967 Gbps 	"SDI Mode Datapath" on page 1-180
(OIF) CEI PHY Interface	>4.976 Gbps to 6.375 Gbps	"(OIF) CEI PHY Interface Mode Datapath" on page 1-182

Table 1-41 lists 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	600 Mbps to 3.75 Gbps	"Basic Single-Width Mode Configurations" on page 1-113
Basic Double Width	1.0 to 11.3 Gbps	"Basic Double-Width Mode Configurations" on page 1-117
Basic (PMA-Direct) single-width	600 Mbps to 3.25 Gbps	"Basic (PMA Direct) $\times 1$ Configuration" on page 1-189
Basic (PMA-Direct) double-width	1.0 to 6.5 Gbps	"Basic (PMA Direct) $\times N$ Configuration" on page 1-189
XAUI	3.125 Gbps up to HiGig at 3.75 Gbps	"XAUI Mode Datapath" on page 1-157
GIGE	1.25 Gbps	"GIGE Mode Datapath" on page 1-167

Table 1–41. Functional Modes for the Stratix IV GT Devices (Part 2 of 2)

Functional Mode	Data Rate	Refer To
Serial RapidIO	<ul style="list-style-type: none"> ■ 2.5 Gbps ■ 3.125 Gbps 	“Serial RapidIO Mode” on page 1–182
SONET/SDH	<ul style="list-style-type: none"> ■ OC-48 ■ OC-96 	“SONET/SDH Frame Structure” on page 1–172
SDI	<ul style="list-style-type: none"> ■ 3G at 2.97/2.967 Gbps 	“SDI Mode Datapath” on page 1–180
(OIF) CEI PHY Interface	> 4.976 Gbps to 6.375 Gbps	“(OIF) CEI PHY Interface Mode Datapath” on page 1–182

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 lists 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 1–113 and “Basic Double-Width Mode Configurations” on page 1–117.

Table 1-43 lists 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 ⁽¹⁾

Basic Functional Mode	Supported Data Rate Range	PMA-PCS Interface Width
Basic single-width mode	600 Mbps to 3.75 Gbps	8-bit, 10-bit
Basic double-width mode	1.0 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 1-113 and “Basic Double-Width Mode Configurations” on page 1-117.

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 1-113 and “Basic Double-Width Mode Configurations” on page 1-117.

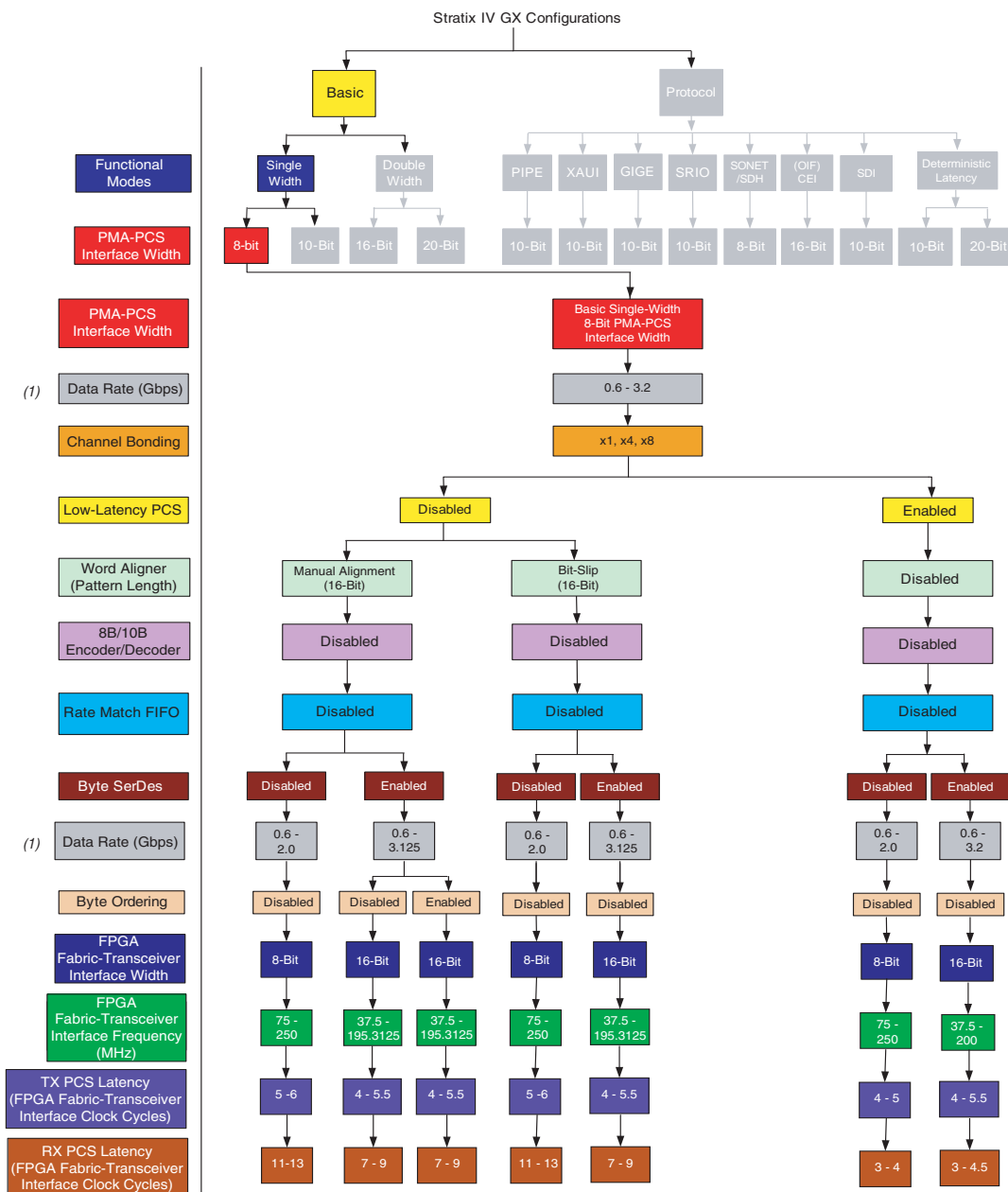
-  The PCS latency in Basic single-width and Basic double-width modes with and without the low latency PCS mode option is pending characterization.
-  Basic double-width mode configurations at data rates of > 6.5 Gbps are only allowed in low-latency PCS bypass mode.

Basic Single-Width Mode Configurations

Figure 1-94 shows Stratix IV GX transceiver configurations allowed in Basic single-width functional mode with an 8-bit PMA-PCS interface.

Figure 1-95 shows Stratix IV GT transceiver configurations allowed in Basic single-width functional mode with an 8-bit PMA-PCS interface.

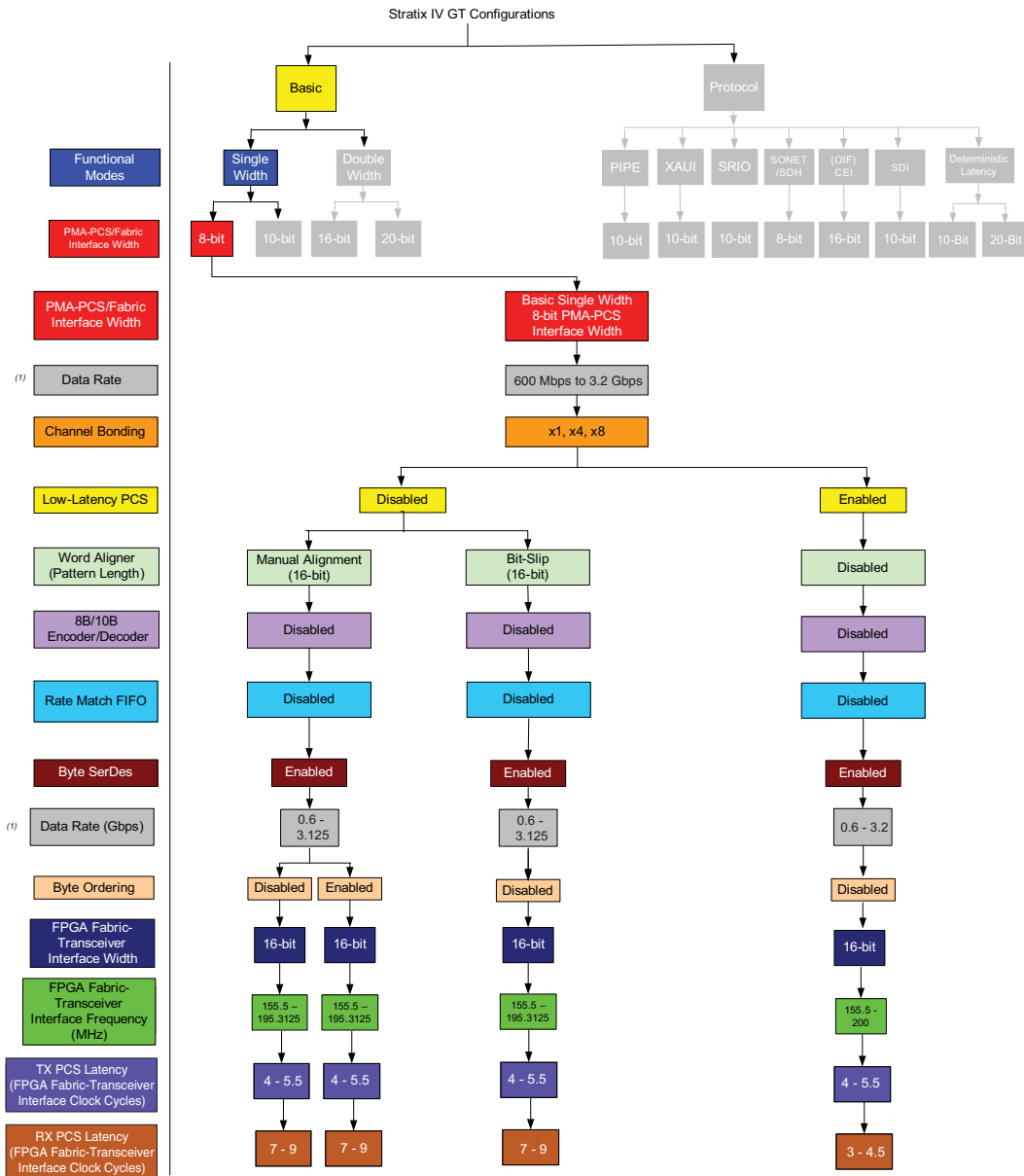
Figure 1-94. Transceiver Configurations in Basic Single-Width Mode with an 8-Bit PMA-PCS Interface for Stratix IV GX Devices



Note 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 for Stratix IV Devices* chapter.

Figure 1-95. Transceiver Configurations in Basic Single-Width Mode with an 8-Bit PMA-PCS Interface for Stratix IV GT Devices



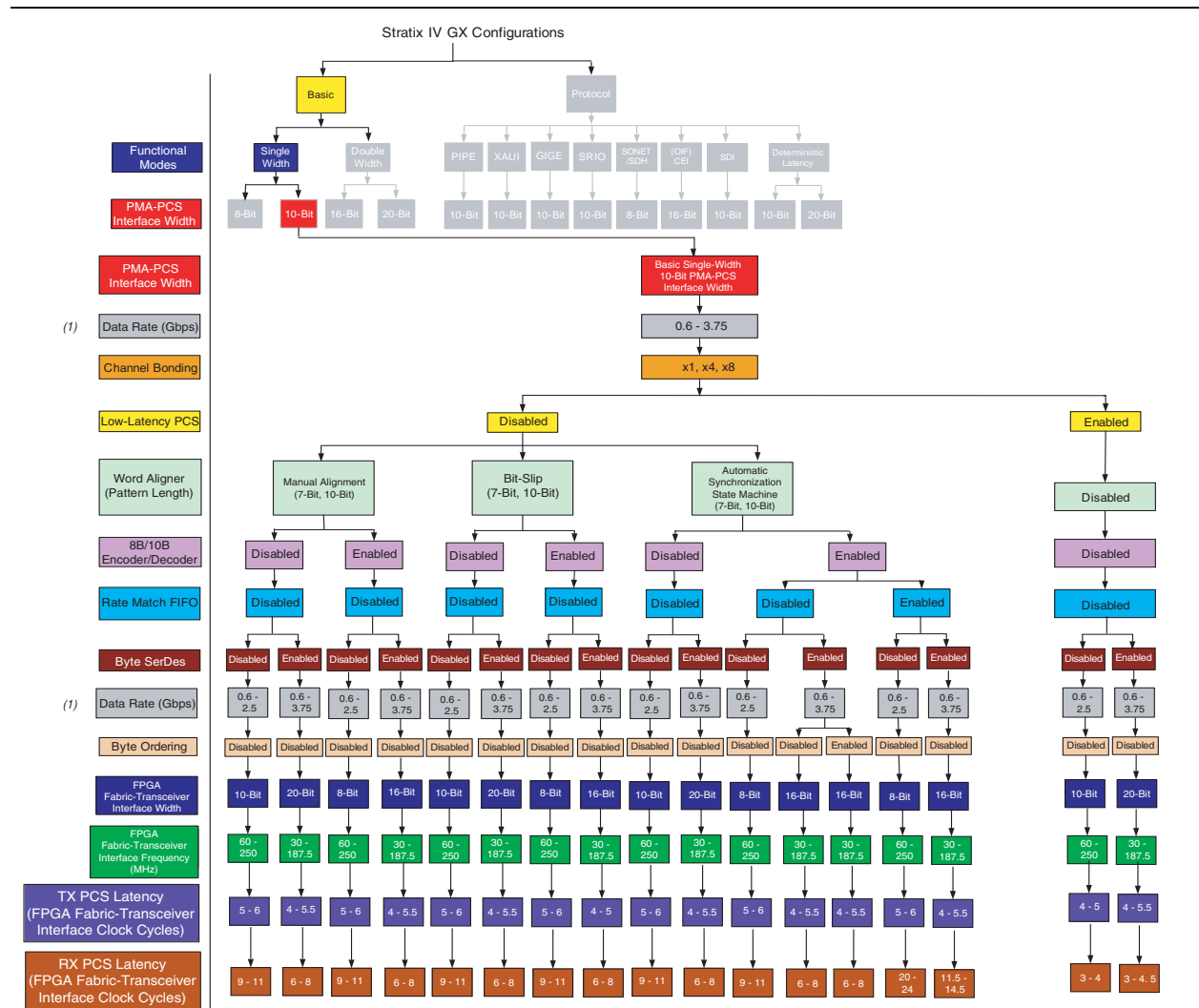
Note to Figure 1-95:

- (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 for Stratix IV Devices* chapter.

Figure 1-96 shows Stratix IV GX transceiver configurations allowed in Basic single-width functional mode with a 10-bit PMA-PCS interface.

Figure 1-97 shows Stratix IV GT transceiver configurations allowed in Basic single-width functional mode with a 10-bit PMA-PCS interface.

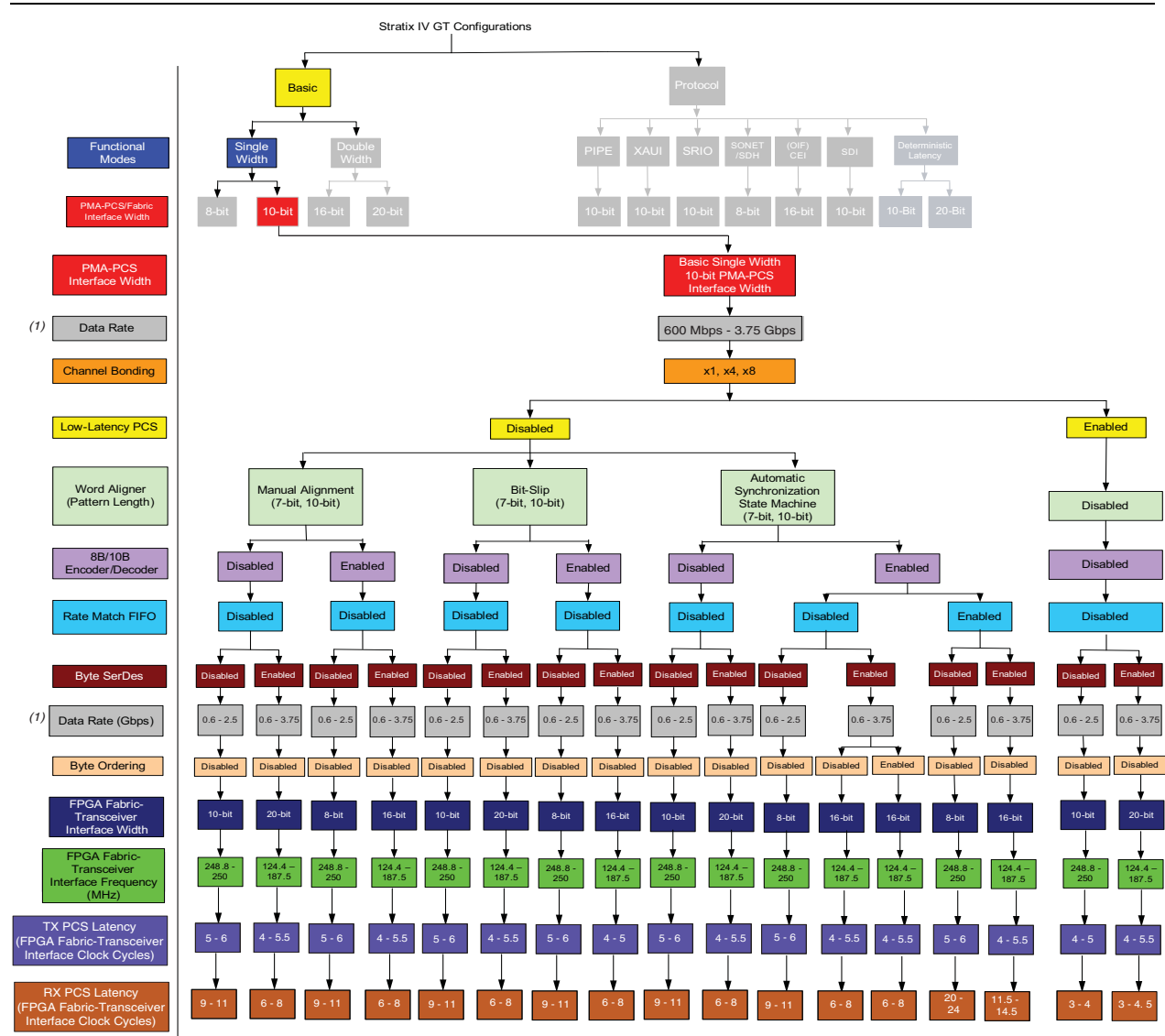
Figure 1-96. Transceiver Configurations in Basic Single-Width Mode with a 10-Bit PMA-PCS Interface for Stratix IV GX Devices



Note 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 for Stratix IV Devices* chapter.

Figure 1-97. Transceiver Configurations in Basic Single-Width Mode with a 10-Bit PMA-PCS Interface for Stratix IV GT Devices



Note to Figure 1-97:

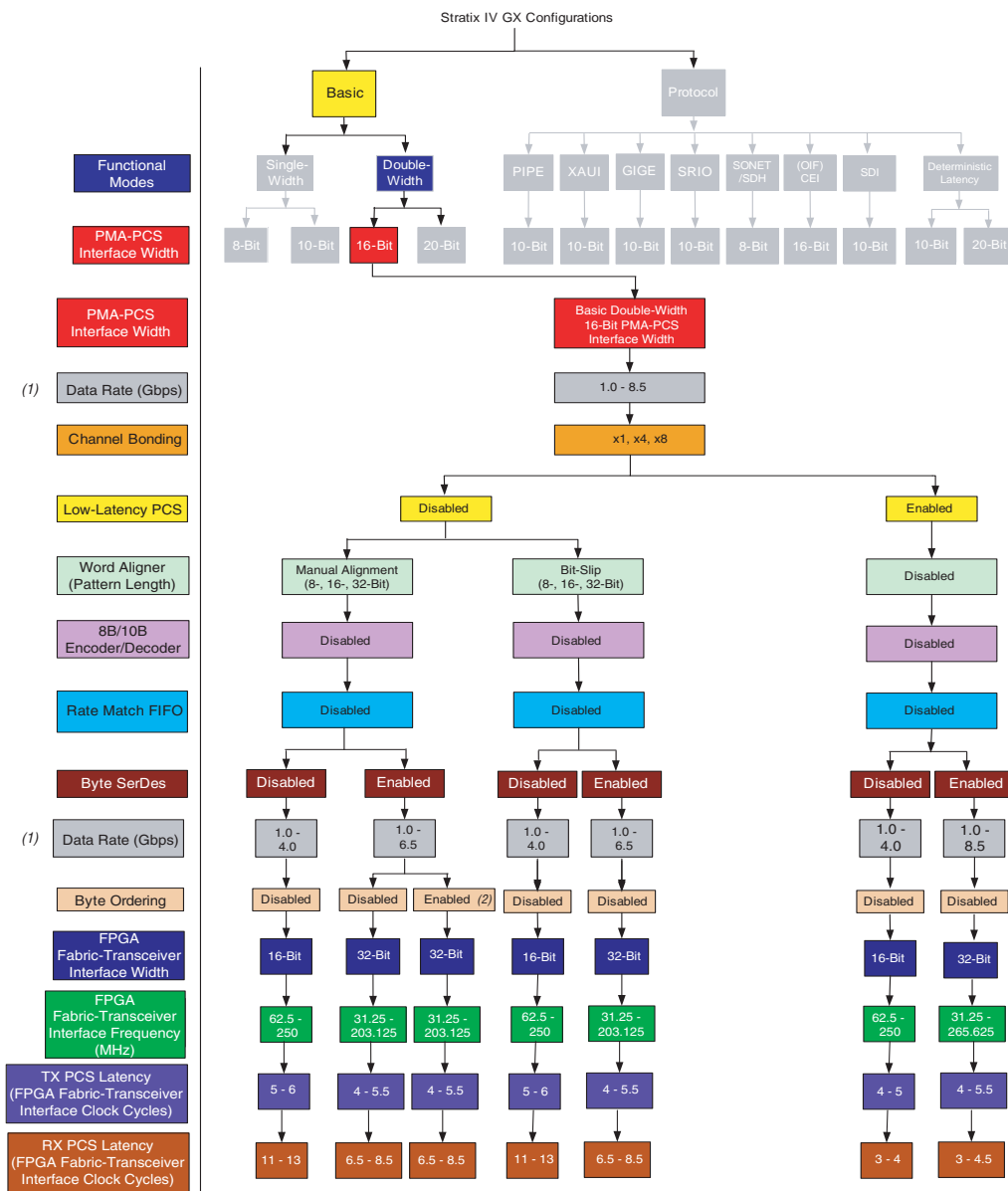
(1) The maximum data rate specification shown in Figure 1-97 is valid only for the -2 (fastest) speed grade devices. For data rate specifications for other speed grades offered, refer to the *DC and Switching Characteristics for Stratix IV Devices* chapter.

Basic Double-Width Mode Configurations

Figure 1-98 shows Stratix IV GX transceiver configurations allowed in Basic double-width functional mode with a 16-bit PMA-PCS interface.

Figure 1-99 shows Stratix IV GT transceiver configurations allowed in Basic double-width functional mode with a 16-bit PMA-PCS interface.

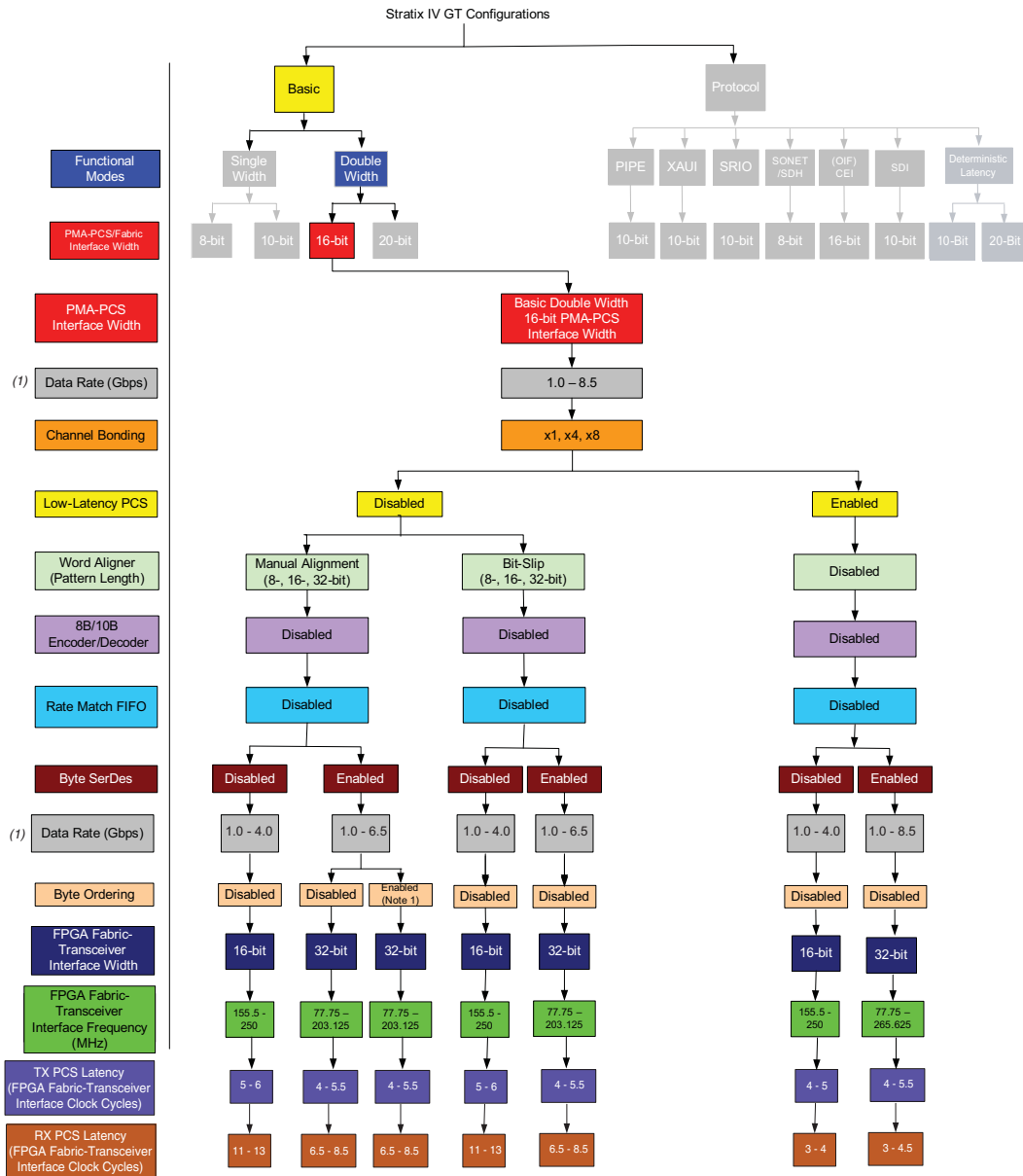
Figure 1-98. Transceiver Configurations in Basic Double-Width Mode with a 16-Bit PMA-PCS Interface for Stratix IV GX Devices



Notes to Figure 1-98:

- (1) The maximum data rate specification shown in Figure 1-98 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 for Stratix IV Devices* chapter.
- (2) The byte ordering block is available only if you select the word alignment pattern length of 16 or 32 bits.

Figure 1-99. Transceiver Configurations in Basic Double-Width Mode with a 16-Bit PMA-PCS Interface for Stratix IV GT Devices



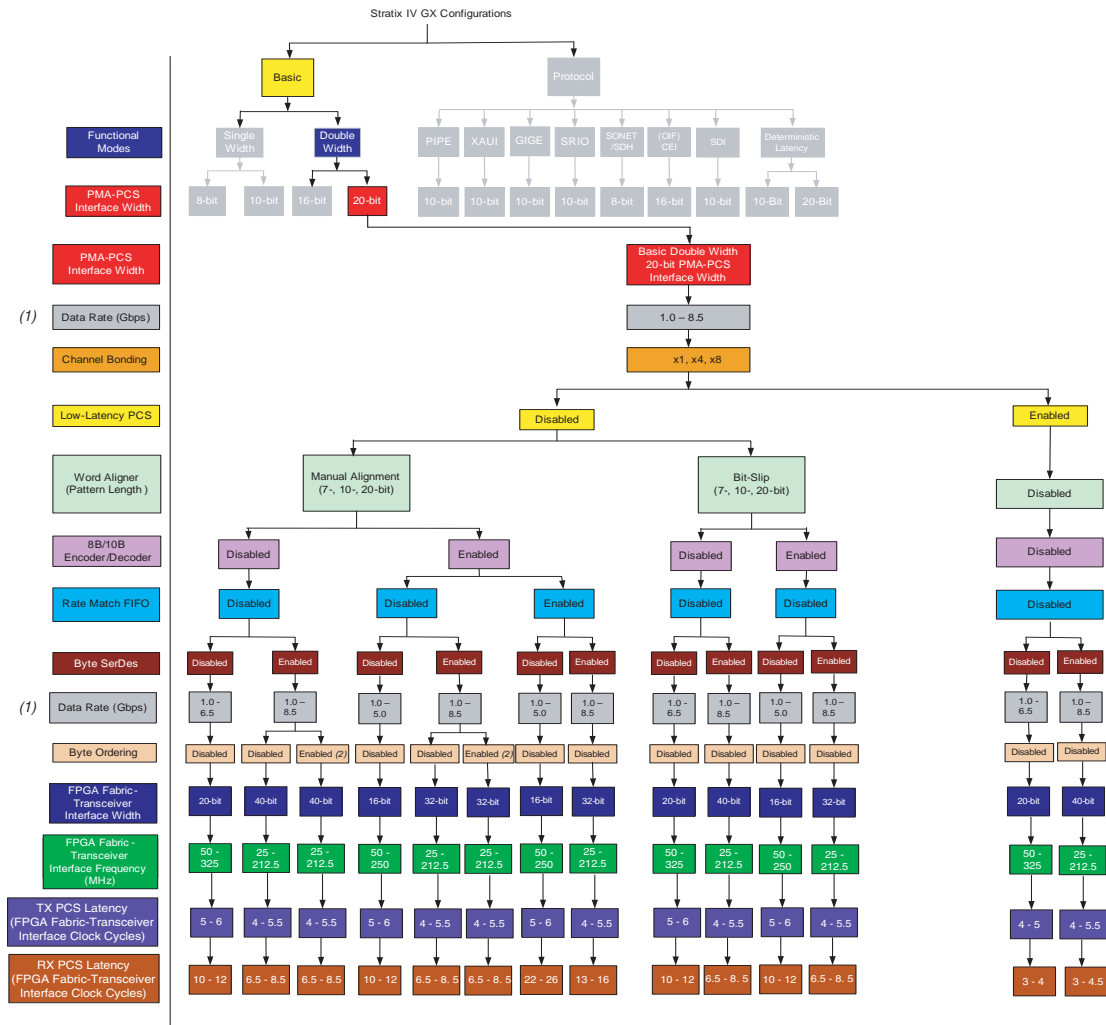
Note to Figure 1-99:

(1) The maximum data rate specification shown in Figure 1-99 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 for Stratix IV Devices* chapter.

Figure 1-100 shows Stratix IV GX transceiver configurations allowed in Basic double-width functional mode with a 20-bit PMA-PCS interface.

Figure 1-101 shows Stratix IV GT transceiver configurations allowed in Basic double-width functional mode with a 20-bit PMA-PCS interface.

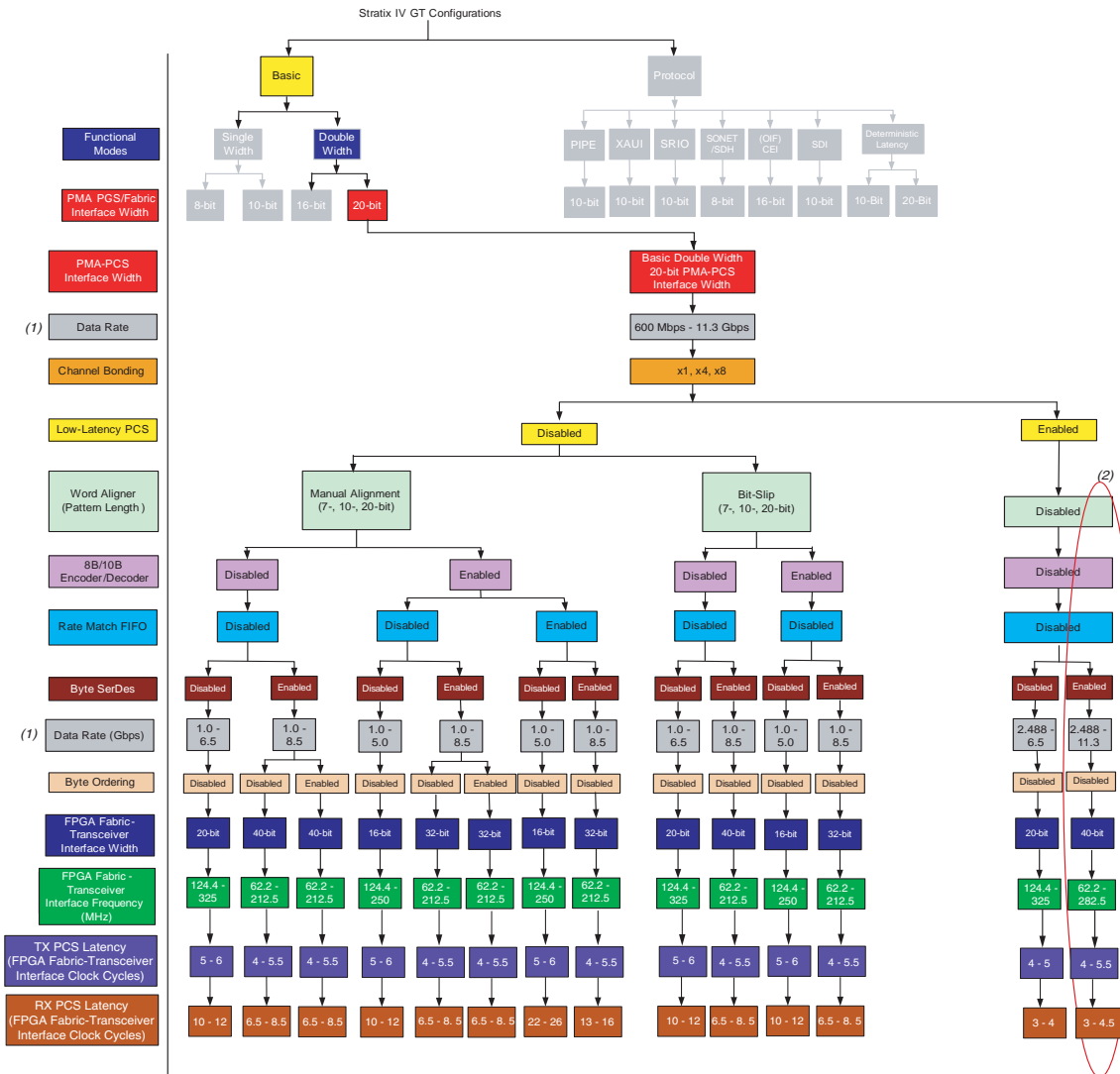
Figure 1-100. Transceiver Configurations in Basic Double-Width Mode with a 20-Bit PMA-PCS Interface for Stratix IV GX Devices



Notes to Figure 1-100:

- (1) The maximum data rate specification shown in Figure 1-100 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 for Stratix IV Devices* chapter.
- (2) The byte ordering block is available only if you select the word alignment pattern length of 20 bits.

Figure 1-101. Transceiver Configurations in Basic Double-Width Mode with a 20-Bit PMA-PCS Interface for Stratix IV GT Devices



Notes to Figure 1-101:

- (1) The maximum data rate specification shown in Figure 1-101 is valid only for the -1 (fastest) speed grade devices. For data rate specifications for other speed grades offered, refer to the *DC and Switching Characteristics for Stratix IV Devices* 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 such as 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.

-  Manual CDR lock mode is required because you must be in lock-to-reference mode during OOB signaling.
-  For more information about the transmitter buffer in the Electrical Idle state, refer to the “Transmitter Buffer Electrical Idle” section in “PCIe Mode” on page 1-127.

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 OOB signals.

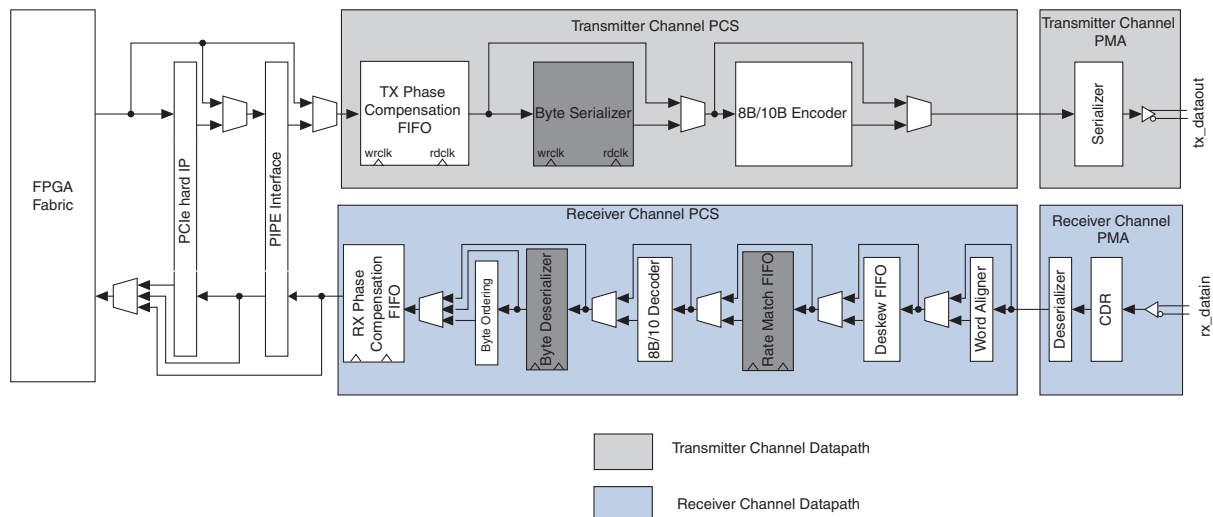
-  For more information on the signal threshold detection circuitry, refer to the “Signal Threshold Detection Circuitry” section.
-  For information about other protocols supported using Basic functional mode, refer to *AN 577: Recommended Protocol Configurations for Stratix IV FPGAs*.

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 1–102 shows the transceiver datapath when using deterministic latency mode.

Figure 1–102. 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 about the `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 $\times 1$ channels only. ATX PLLs do not have the feedback path enabled; therefore, they cannot be used for implementing the CPRI configuration.

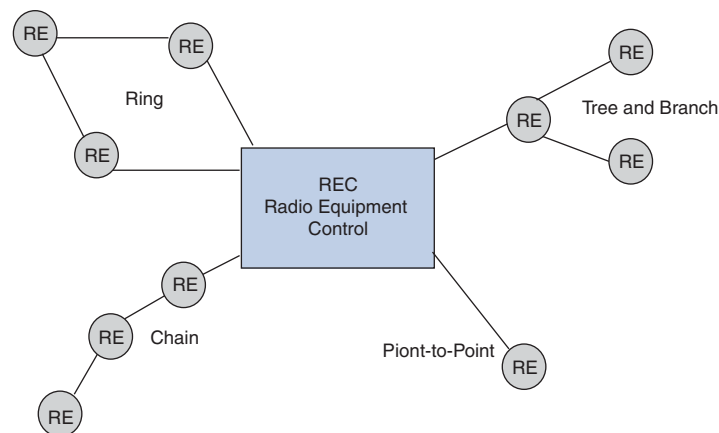
In the deterministic latency $\times 4$ option, up to four CPRI TX channels can be bundled in an $\times 4$ group so that they all have the same TX uncertainty and just require one TX PLL to compensate for it. This is allowed in cases where the data rates are multiples of a single PLL output frequency; for example, 0.6144 Gbps, 1.228 Gbps, 2.4576 Gbps, and 4.9152 Gbps. For $\times 4$ 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 1-103](#) 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 1-103. CPRI Topologies

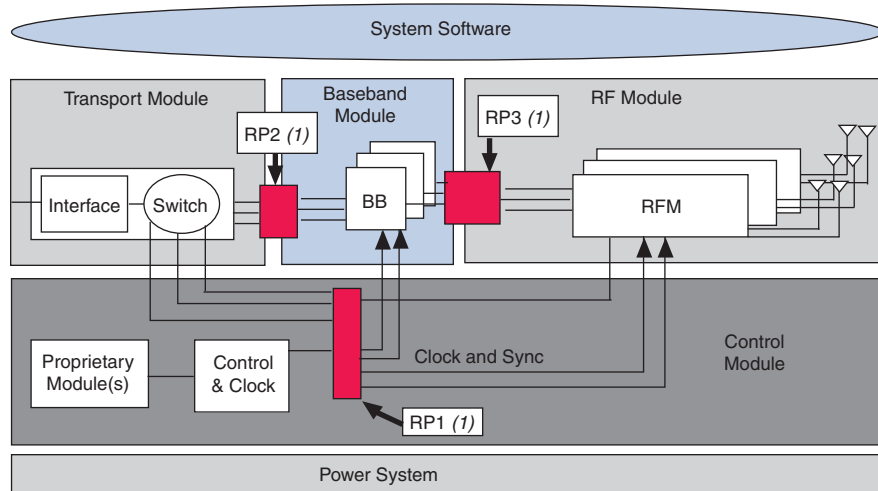


If the destination for high-speed serial data leaving the REC is the first RE, it is a single-hop connection. If serial data from the REC has to traverse through multiple REs before reaching the destination RE, it is a multi-hop connection. Remotely locating the RF transceiver from the main base station 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 1-104 shows 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 1-104. BTS in OSBAL

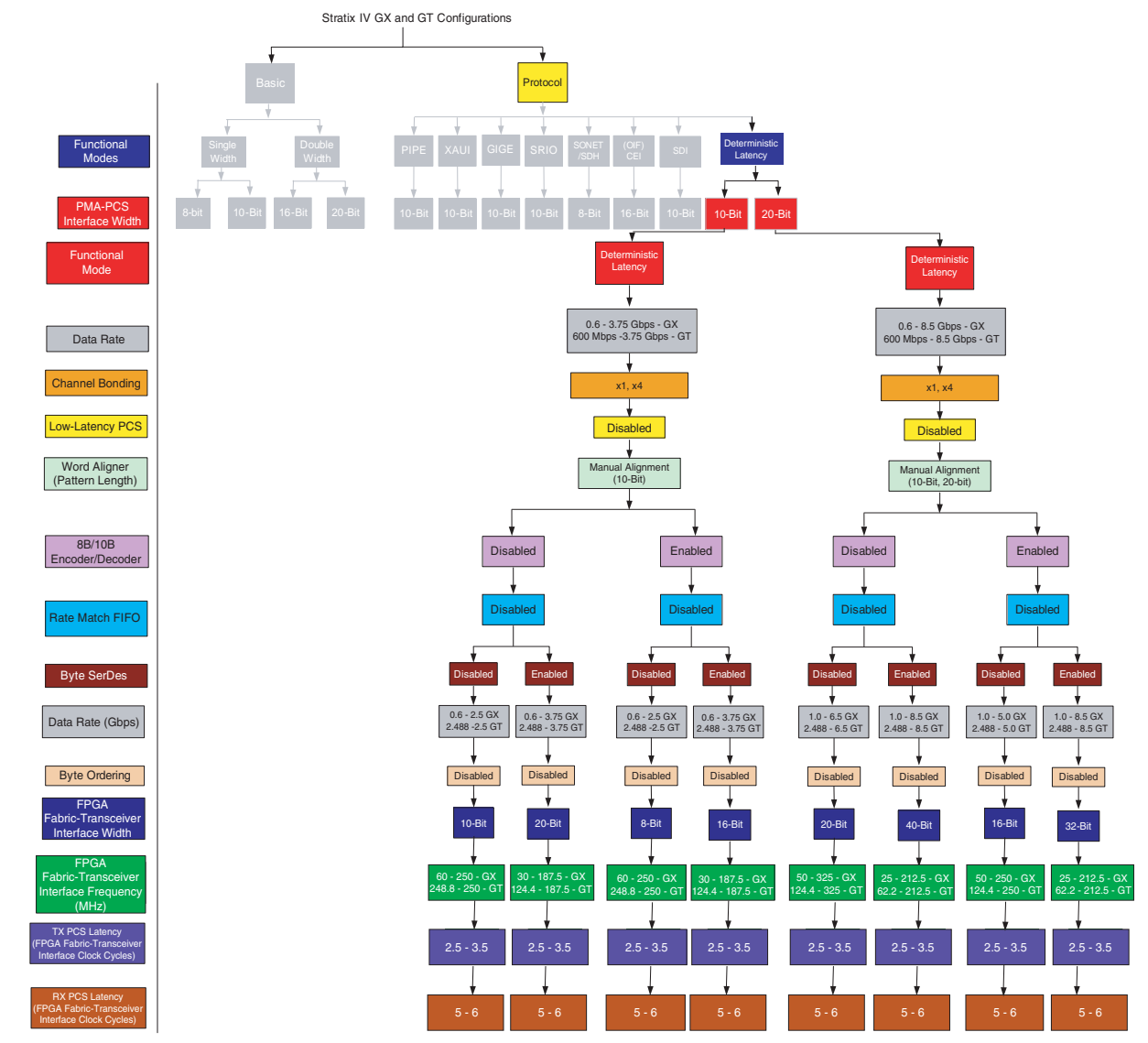


Note to Figure 1-104:

(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 1-105 shows the block diagram of the deterministic latency option.

Figure 1-105. 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.
-  When the byte serializer is enabled, divide the latency of the byte serializer block by 2. Therefore, the byte serializer/deserializer latency is 0.5–1 when enabled. When the byte serializer is disabled, the latency value is 1.
-  Dividing the latency also applies to the sub-blocks after the byte serializer (8B/10B encoder) and before the byte deserializer (word aligner, 8B/10B decoder, etc.). The latency value of each sub-block should also be divided by 2 if they are enabled.

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 & OSBAI	CPRI Data Rate (GBPS)	PCS Clock Frequency (MHz)	OSBAI 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	3.072	153.6	1.536	76.8
		16 bit/20 bit	4.915 ^{(2), (3)}	245.76	3.072 ⁽³⁾	153.6
		32 bit/40 bit	6.144 ^{(2), (3), (4)}	307.2	6.144 ^{(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.

PCIe Mode

Intel Corporation has developed a PHY interface for the PCIe Architecture specification to enable implementation of a PCIe-compliant physical layer device. The PCIe specification also defines a standard interface between the physical layer device and the media access control layer (MAC). Version 2.0 of the PCIe specification provides implementation details for a PCIe-compliant physical layer device at both Gen1 (2.5 GT/s) and Gen2 (5 GT/s) signaling rates.

To implement a Version 2.0 PCIe-compliant PHY, you must configure the Stratix IV GX and GT transceivers in PCIe functional mode. Stratix IV GX and GT devices have built-in PCIe hard IP blocks that you can use to implement the PHY-MAC layer, data link layer, and transaction layer of the PCIe protocol stack. You can also bypass the PCIe hard IP blocks and implement the PHY-MAC layer, data link layer, and transaction layer in the FPGA fabric using a soft IP. If you enable the PCIe 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 PCIe functional mode using one of the following two methods:

- ALTGX MegaWizard Plug-In Manager—if you do not use the PCIe hard IP block
- PCIe Compiler—if you use the PCIe hard IP block



Description of PCIe hard IP architecture and PCIe mode configurations allowed when using the PCIe hard IP block are beyond the scope of this chapter. For more information about the PCIe hard IP block, refer to the *PCI Express Compiler User Guide*.

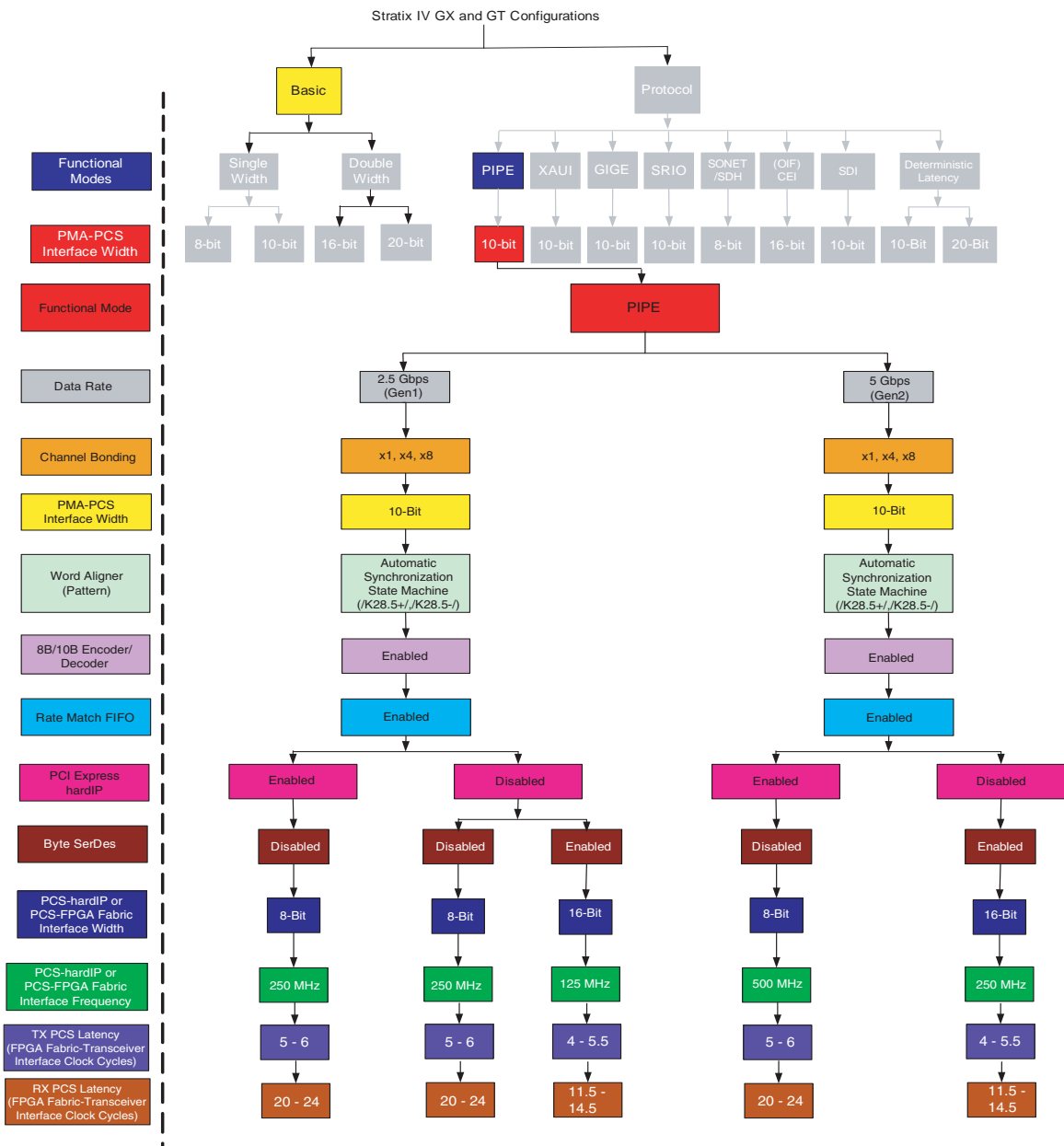
PCIe Mode Configurations

Stratix IV GX and GT transceivers support both Gen1 (2.5 Gbps) and Gen2 (5 Gbps) data rates in PCIe 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 PCIe signaling rates is critical for speed negotiation during link training.

Stratix IV GX and GT transceivers support $\times 1$, $\times 4$, and $\times 8$ lane configurations in PCIe functional mode at both 2.5 Gbps and 5 Gbps data rates. In PCIe $\times 1$ configuration, the PCS and PMA blocks of each channel are clocked and reset independently. PCIe $\times 4$ and $\times 8$ configurations support channel bonding for four-lane and eight-lane PCIe links. In these bonded channel configurations, the PCS and PMA blocks of all bonded channels share common clock and reset signals.

Figure 1-106 shows the Stratix IV GX and GT transceiver configurations allowed in PCIe functional mode.

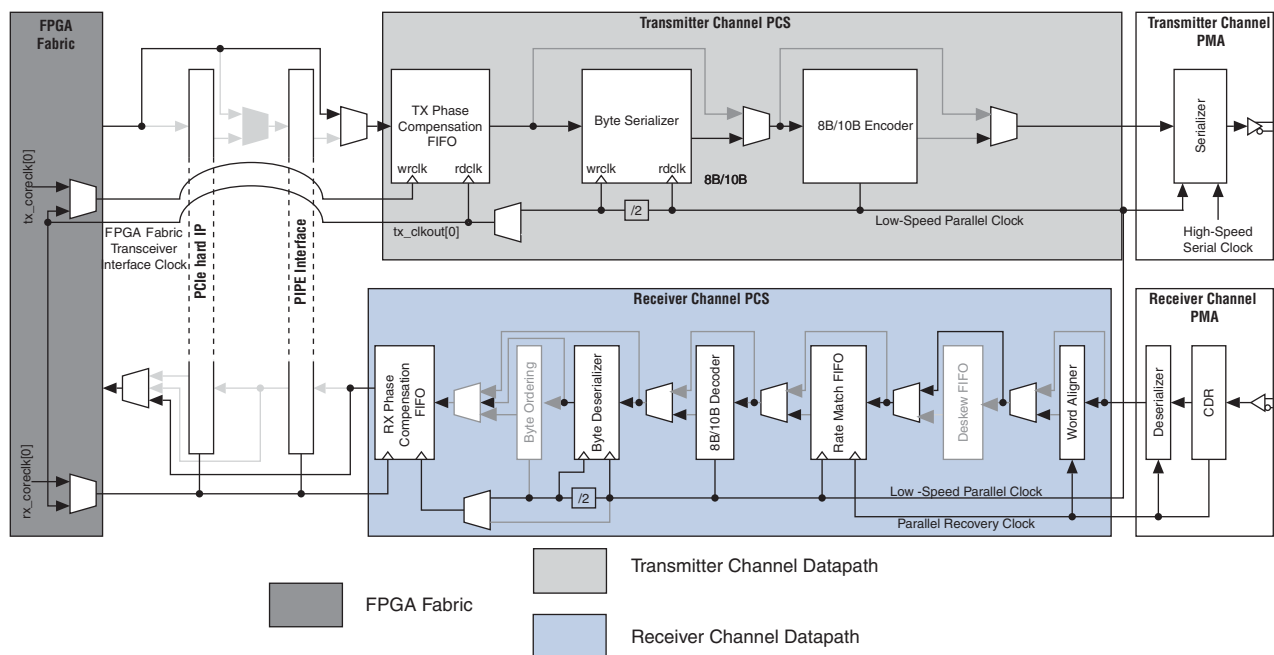
Figure 1-106. Stratix IV GX and GT Transceivers in PCIe Functional Mode



PCIe Mode Datapath

Figure 1-107 shows the Stratix IV GX and GT transceiver datapath when configured in PCIe functional mode.

Figure 1-107. Stratix IV GX and GT Transceiver Datapath in PCIe ×1 Mode



For more information, refer to “Rate Match (Clock Rate Compensation) FIFO” on page 1-77.

Table 1-46 lists the transceiver datapath clock frequencies in PCIe functional mode configured using the ALTGX MegaWizard Plug-In Manager.

Table 1-46. Stratix IV GX and GT Transceiver Datapath Clock Frequencies in PCIe 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)
PCIe ×1, ×4, and ×8 (Gen1)	2.5 Gbps	1.25 GHz	250 MHz	250 MHz	125 MHz
PCIe ×1, ×4, and ×8 (Gen2)	5 Gbps	2.5 GHz	500 MHz	N/A ⁽¹⁾	250 MHz

Note to Table 1-46:

(1) In PCIe 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 PCIe mode.

 For more information about transceiver datapath clocking in different PCIe configurations, refer to the *Transceiver Clocking in Stratix IV Devices* chapter.

Table 1-47 lists the transmitter and receiver datapaths in PCIe mode.

Table 1-47. Datapaths in PCIe Mode

	Transmitter Datapath	Receiver Datapath
PCIe interface	Y	Y
Transmitter phase compensation FIFO	Y	—
Optional byte serializer (enabled for 16-bit and disabled for 8-bit FPGA fabric-transceiver interface)	Y	—
8B/10B encoder	Y	—
10:1 serializer	Y	—
Transmitter buffer with receiver detect circuitry	Y	—
Receiver buffer with signal detect circuitry	—	Y
1:10 deserializer	—	Y
Word aligner that implements PCIe-compliant synchronization state machine	—	Y
Optional rate match FIFO (clock rate compensation) that can tolerate up to 600 PPM frequency difference	—	Y
8B/10B decoder	—	Y
Optional byte deserializer (enabled for 16-bit and disabled for 8-bit FPGA fabric-transceiver interface)	—	Y
Receiver phase compensation FIFO	—	Y

Table 1-48 lists the features supported in PCIe functional mode for 2.5 Gbps and 5 Gbps data rate configurations.

For more information, refer to “Rate Match FIFO in PCIe Mode” on page 1-78.

Table 1-48. Supported Features in PCIe Mode (Part 1 of 2)

Feature	2.5 Gbps (Gen1)	5 Gbps (Gen2)
×1, ×4, ×8 link configurations	Y	Y
PCIe-compliant synchronization state machine	Y	Y
±300 PPM (total 600 PPM) clock rate compensation	Y	Y
8-bit FPGA fabric-transceiver interface	Y	—
16-bit FPGA fabric-transceiver interface	Y	Y
Transmitter buffer electrical idle	Y	Y
Receiver Detection	Y	Y
8B/10B encoder disparity control when transmitting compliance pattern	Y	Y
Power state management	Y	Y

Table 1-48. Supported Features in PCIe Mode (Part 2 of 2)

Feature	2.5 Gbps (Gen1)	5 Gbps (Gen2)
Receiver status encoding	Y	Y
Dynamic switch between 2.5 Gbps and 5 Gbps signaling rate	—	Y
Dynamically selectable transmitter margining for differential output voltage control	—	Y
Dynamically selectable transmitter buffer de-emphasis of -3.5 db and -6 dB	—	Y

PCIe Interface

In PCIe mode, each channel has a PCIe interface block that transfers data, control, and status signals between the PHY-MAC layer and the transceiver channel PCS and PMA blocks. The PCIe interface block is compliant to version 2.0 of the PCIe specification. If you use the PCIe 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 PCIe interface block is only used in PCIe mode and cannot be bypassed.

Besides transferring data, control, and status signals between the PHY-MAC layer and the transceiver, the PCIe interface block implements the following functions required in a PCIe-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 PCIe 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 PCIe specification

Transmitter Buffer Electrical Idle

When the input signal `tx_forceelecidle` is asserted high, the PCIe 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 PCIe Base Specification 2.0 for both PCIe Gen1 and Gen2 data rates.

Figure 1-108 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 PCIe protocol requires the transmitter buffer to stay in electrical idle for a minimum of 20 ns for both Gen1 and Gen2 data rates.


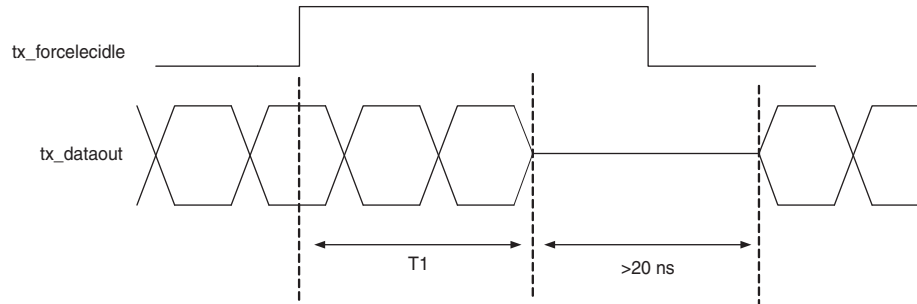
 The minimum period of time for which the `tx_forceidle` signal must be asserted high such that the transmitter buffer stays in electrical idle state for at least 20 ns is pending characterization.

Figure 1-108. Transmitter Buffer Electrical Idle State




The PCIe specification requires the transmitter buffer to be in electrical idle in certain power states. For more information about the `tx_forceidle` signal levels required in different PCIe power states, refer to [Table 1-50 on page 1-137](#).

Receiver Detection

During the detect substate of the link training and status state machine (LTSSM), the PCIe 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 PCIe specification requires the receiver detect operation to be performed during the P1 power state.

The PCIe 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 PCIe 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 PCIe 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. The receiver detect circuitry monitor requires 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 PCIe Base Specification 2.0.

Receiver detect circuitry communicates the status of the receiver detect operation to the PCIe interface block. If a far-end receiver is successfully detected, the PCIe 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 PCIe interface block asserts `pipephydonestatus` for one clock cycle and synchronously drives the `pipestatus[2:0]` signal to `3'b000`.

Figure 1–109 and Figure 1–110 show the receiver detect operation where a receiver was successfully detected and where a receiver was not detected, respectively.

Figure 1–109. Receiver Detect, Successfully Detected

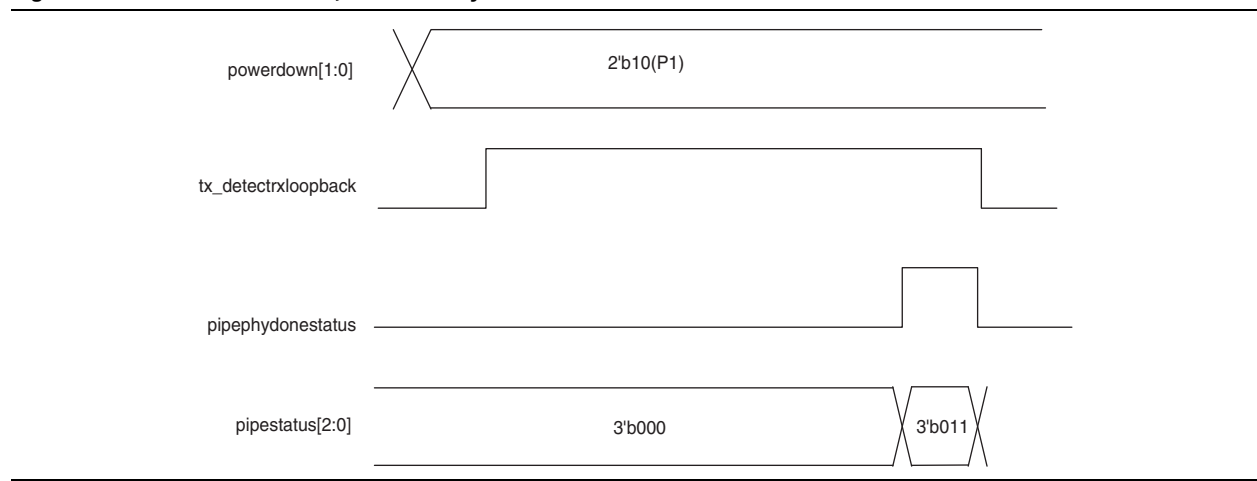
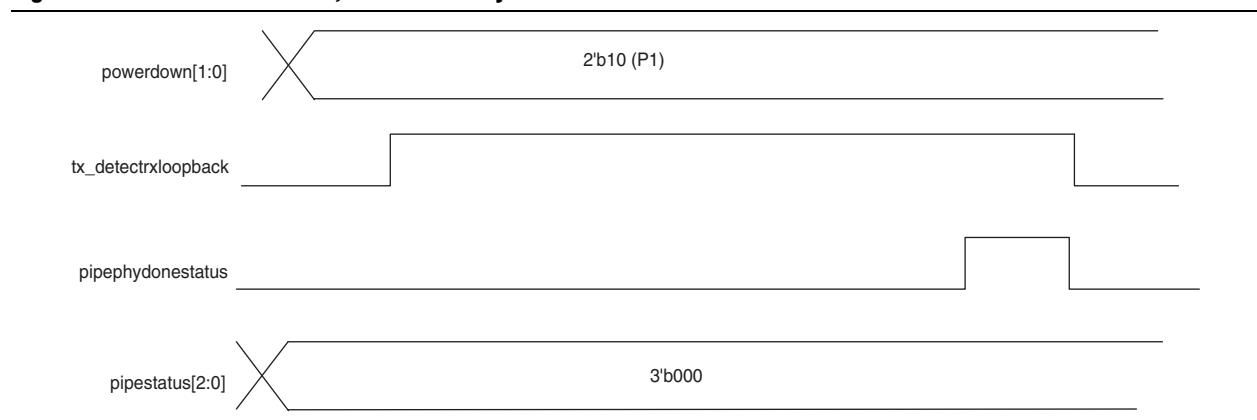


Figure 1–110. Receiver Detect, Unsuccessfully Detected



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 PCIe Base Specification 2.0. The polling.compliance substate is intended to assess if the transmitter is electrically compliant with the PCIe 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 PCIe protocol requires the first /K28.5/ code group of the compliance pattern to be encoded with negative current disparity. To satisfy this requirement, the PCIe 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 the `tx_forcedispcompliance` high in the same parallel clock cycle as /K28.5/D21.5/ of the compliance pattern on the `tx_datain` port.

Figure 1-111 and Figure 1-112 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 1-111. Compliance Pattern Transmission Support, 8-Bit Channel Width Configurations

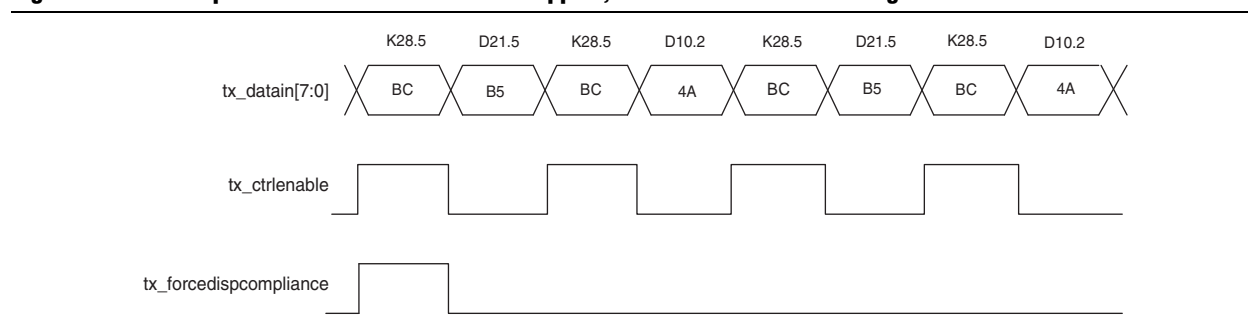
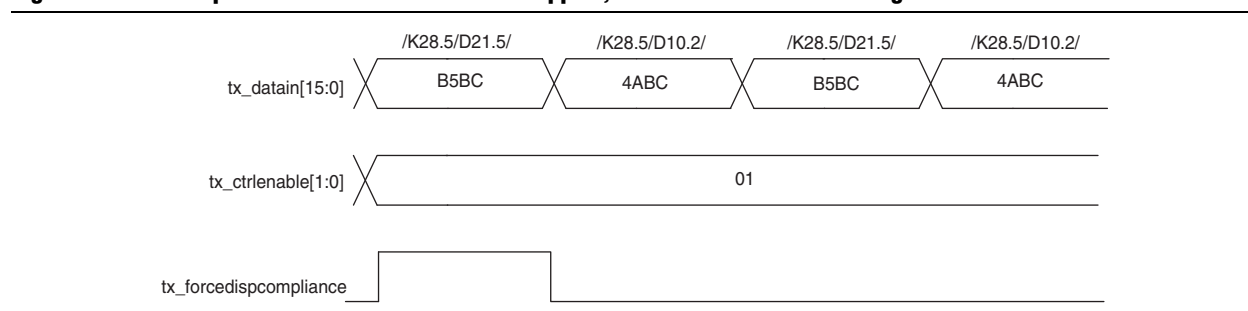


Figure 1-112. Compliance Pattern Transmission Support, 16-Bit Wide Channel Configurations



Power State Management

The PCIe 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 PCIe 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 PCIe specification provides the mapping of these power states to the LTSSM states specified in the PCIe 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 PCIe-compliant PHY.

The PCIe interface in Stratix IV GX and GT transceivers provides an input port, `powerdn[1:0]`, for each transceiver channel configured in PCIe mode. Table 1-49 lists mapping between the logic levels driven on the `powerdn[1:0]` port and the resulting power state that the PCIe 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

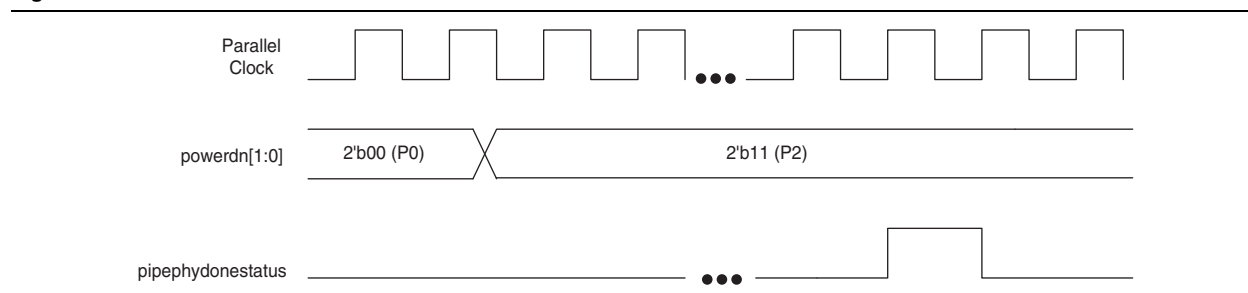


When transitioning from the P0 power state to lower power states (P0s, P1, and P2), the PCIe specification requires the physical layer device to implement power saving measures. Stratix IV GX and GT transceivers do not implement these power saving measures except putting the transmitter buffer in electrical idle in the lower power states.

The PCIe interface block indicates successful power state transition by asserting the `pipephydonestatus` signal for one parallel clock cycle as specified in the PCIe 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 1-113 shows an example waveform for a transition from the P0 to P2 power state.

Figure 1-113. Power State Transition from the P0 to P2 Power State



The PCIe specification allows the PCIe 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_forceidle` signals appropriately in each power state to perform these functions. Table 1-50 lists the logic levels that the PHY-MAC layer must drive on the `tx_detectrxloopback` and `tx_forceidle` signals in each power state.

Table 1-50. Logic Levels for `tx_detectrxloopback` and `tx_forceidle` in Different Power States

Power State	<code>tx_detectrxloopback</code>	<code>tx_forceidle</code>
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 PCIe 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 PCIe 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 PCIe specification and is listed in Table 1-51.

Table 1-51. Encoding of the Status Signals on `pipestatus[2:0]`


<code>pipestatus[2:0]</code>	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 PCIe interface follows the priority listed in Table 1-51 while encoding the receiver status on the `pipestatus[2:0]` port. For example, if the PCIe 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 PCIe Base specification fast training sequences (FTS) are used for bit and byte synchronization to transition from L0s to L0 (PCIe P0s to P0) power states. When transitioning from the L0s to L0 power state, the PCIe Base Specification requires the physical layer device to acquire bit and byte synchronization after receiving a maximum of 255 FTS (~4 μ s at Gen1 data rate and ~2 μ s 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 PCIe specification of acquiring bit and byte synchronization within 4 μ s (Gen1 data rate) or 2 μ s (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 PCIe 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 PCIe Base Specification 2.0 specifies conditions to infer electrical idle at the receiver in various substates of the LTSSM state machine.

In all PCIe modes ($\times 1$, $\times 4$, and $\times 8$), each receiver channel PCS has an optional Electrical Idle Inference module designed to implement the electrical idle inference conditions specified in the PCIe 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_elecidleinferred[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 lists electrical idle inference conditions specified in the PCIe 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_elecidleinferse1[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)	<code>rx_elecidleinferse1[2:0]</code>
L0	Absence of skip ordered set in 128 μ s window	Absence of skip ordered set in 128 μ s 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 μ s window	N/A	3'b111

In the Recovery.Speed substate of the LTSSM state machine with unsuccessful speed negotiation (`rx_elecidleinferse1[2:0] = 3'b110`), the PCIe 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_elecidleinferse1[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_elecidleinferse1[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 μ s interval.

When configured for Gen2 data rate and `rx_elecidleinferse1[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 PCIe Base Specification.

If you select the **Enable Electrical Idle Inference Functionality** option in the ALTGX MegaWizard Plug-In Manager and drive `rx_elecidleinferse1[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 PCIe link, when operating at Gen2 data rate, the downstream device can go into the Disable state after instruction from the upper layer. Once 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 PCIe 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 PCIe Base Specification Rev 2.0.

When the Stratix IV GX and GT device is operating as a downstream device at PCIe 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.

PCIe Gen2 (5 Gbps) Support

The PCIe 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 PCIe ports negotiate the speed (2.5 Gbps or 5 Gbps) at which the link operates. Because the upstream and downstream PCIe ports do not know the speed capabilities of their link partner, the PCIe 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 PCIe 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 PCIe specification requires a PCIe-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 PCIe 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 PCIe 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 PCIe 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 PCIe 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 PCIe rateswitch circuitry performs the dynamic switch between the Gen1 (2.5 Gbps) and Gen2 (5 Gbps) signaling rate. The PCIe rateswitch circuitry consists of:

- PCIe rateswitch controller
- PCIe clock switch circuitry

PCIe Rateswitch Controller

The rateswitch signal serves as the input signal to the PCIe rateswitch controller. After seeing a transition on the rateswitch signal from the PHY-MAC layer, the PCIe rateswitch controller performs the following operations:

- Controls the PCIe 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 PCIe clock switchover circuitry indicates successful rateswitch completion
- Communicates completion of rateswitch to the PCIe interface module, which in turn communicates completion of the rateswitch to the PHY-MAC layer on the pipephydonestatus signal

PCIe rateswitch controller location:

- In PCIe ×1 mode, the PCIe rateswitch controller is located in the transceiver PCS of each channel.
- In PCIe ×4 mode, the PCIe rateswitch controller is located in CMU0_Channel within the transceiver block.
- In PCIe ×8 mode, the PCIe 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 PCIe 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.

PCIe 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 PCIe 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

PCIe transmitter high-speed serial and low-speed parallel clock switch occurs:

- In PCIe ×1 mode, the `CMU_PLL` clock switch occurs in the local clock divider in each transceiver channel.
- In PCIe ×4 mode, the `CMU_PLL` clock switch occurs in the `CMU0` clock divider in the `CMU0_Channel` within the transceiver block.
- In PCIe ×8 mode, the `CMU_PLL` clock switch occurs in the `CMU0` clock divider in the `CMU0_Channel` within the master transceiver block.

In PCIe ×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 PCIe rateswitch controller and the PCIe clock switch circuitry in PCIe ×1, ×4, and ×8 modes.

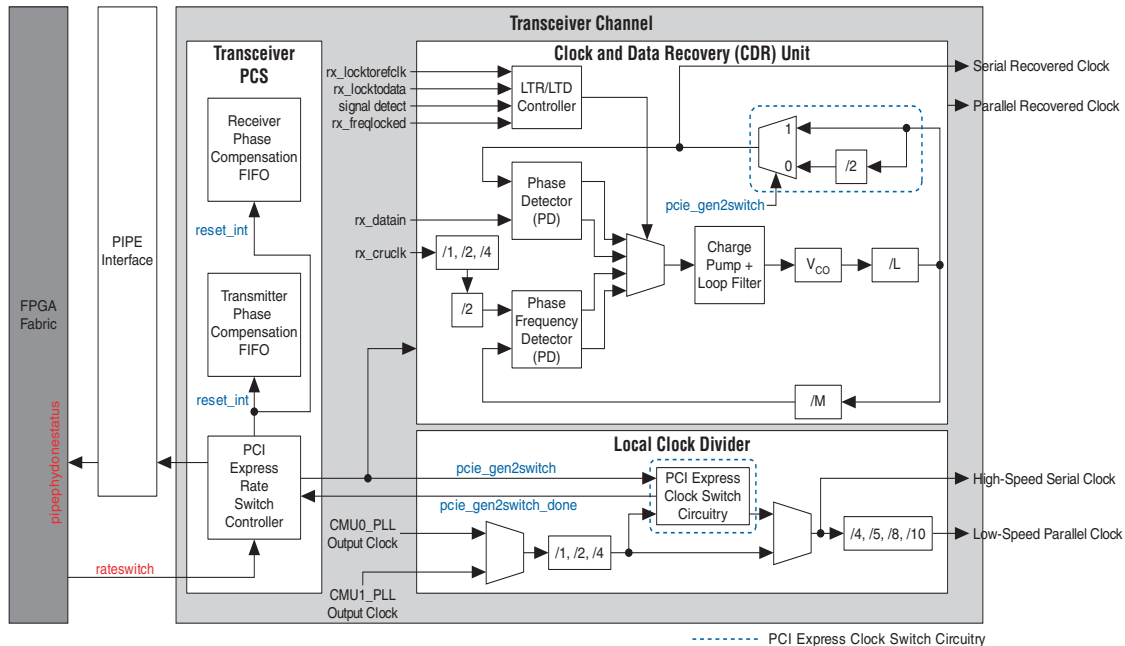
Table 1-53. PCIe Rateswitch Controller and Clock Switch Circuitry

Channel Bonding Option	Location of PCIe Rateswitch Controller Module	Location of PCIe 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	<code>CMU0_Channel</code>	<code>CMU0</code> clock divider in <code>CMU0_Channel</code>	CDR block in receiver PMA of each channel
×8	<code>CMU0_Channel</code> of the master transceiver block	<code>CMU0</code> clock divider in <code>CMU0_Channel</code> 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 PCIe x1 Mode

Figure 1-114 shows the PCIe rateswitch circuitry in PCIe x1 mode configured at Gen2 (5 Gbps) data rate.

Figure 1-114. Dynamic Switch Signaling in PIPE x1 Mode



In PCIe x1 mode configured at Gen2 (5 Gbps) data rate, when the PCIe rateswitch controller sees a transition on the rateswitch signal, it sends control signal `pcie_gen2switch` to the PCIe 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 lists the transceiver clock frequencies when switching between 2.5 Gbps and 5 Gbps signaling rates.

Table 1-54. Transceiver Clock Frequencies Signaling Rates in PCIe x1 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

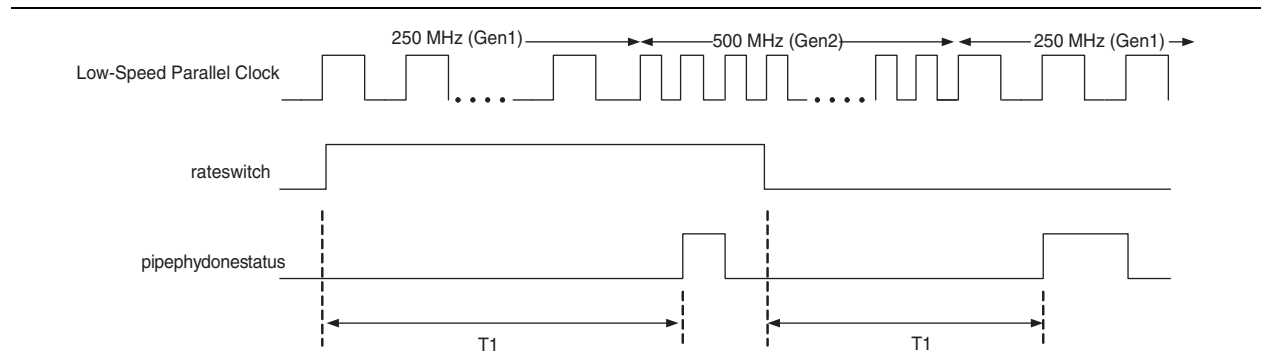
The PCIe clock switch circuitry in the local clock divider block performs the clock switch between 250 MHz and 500 MHz on the low-speed parallel clock when switching between Gen1 (2.5 Gbps) and Gen2 (5 Gbps) signaling rates. It indicates successful completion of clock switch on the `pcie_gen2switchdone` signal to the PCIe rateswitch controller. The PCIe rateswitch controller forwards the clock switch completion status to the PCIe interface block. The PCIe interface block communicates the clock switch completion status to the PHY-MAC layer by asserting the `pipephydonestatus` signal for one parallel clock cycle.

Figure 1-115 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.



Time T1 from a transition on the `rateswitch` signal to the assertion of `pipephydonestatus` is pending characterization.

Figure 1-115. Low-Speed Parallel Clock Switching in PCIe x1 Mode

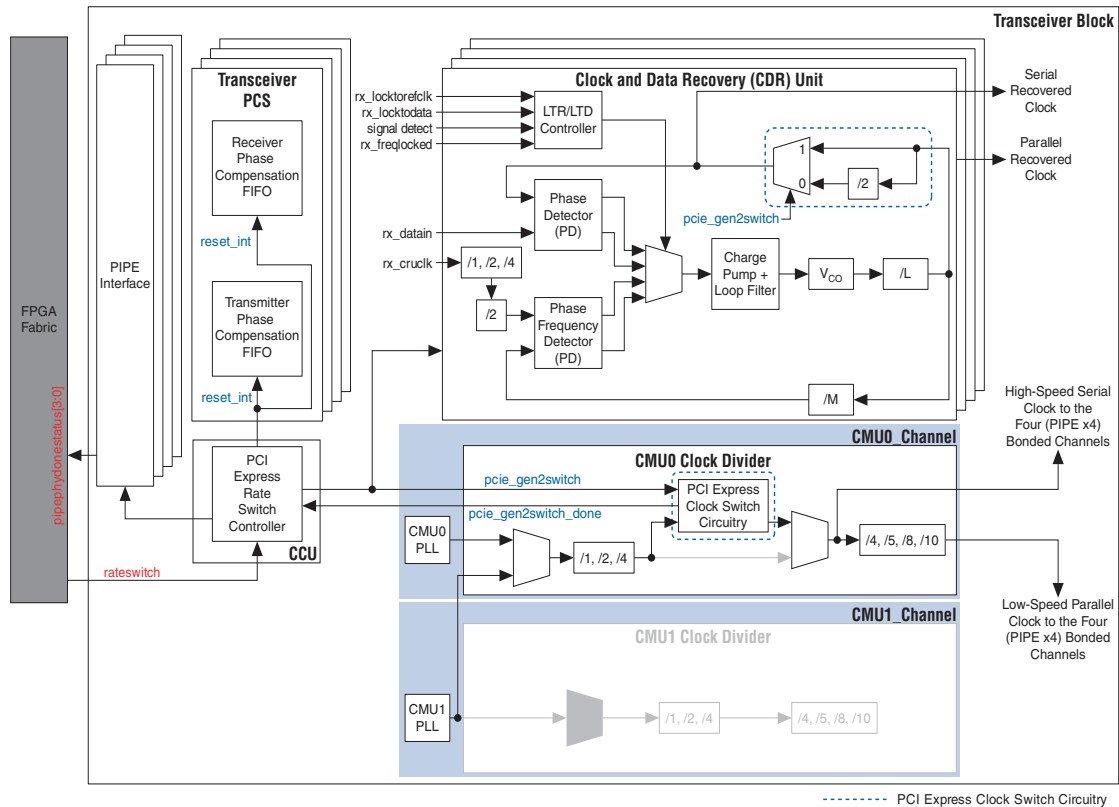


As a result of the signaling rateswitch between Gen1 (2.5 Gbps) and Gen2 (5 Gbps), the FPGA fabric-transceiver interface clock switches between 125 MHz and 250 MHz. The FPGA fabric-transceiver interface clock clocks the read side and write side of the transmitter phase compensation FIFO and the receiver 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 PCIe rateswitch controller automatically disables and resets the pointers during clock switch. When the PCIe clock switch circuitry in the local clock divider indicates successful clock switch completion, the PCIe rateswitch controller releases the phase compensation FIFO pointer resets.

Dynamic Switch Between Gen1 (2.5 Gbps) and Gen2 (5 Gbps) Signaling Rates in PCIe ×4 Mode

Figure 1-116 shows the PCIe rateswitch circuitry in PCIe ×4 mode configured at Gen2 (5 Gbps) data rate.

Figure 1-116. Dynamic Switch Signaling in PCIe ×4 Mode



In PCIe ×4 mode configured at Gen2 (5 Gbps) data rate, when the PCIe rateswitch controller sees a transition on the rateswitch signal, it sends the pcie_gen2switch control signal to the PCIe 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 lists the transceiver clock frequencies when switching between the 2.5 Gbps and 5 Gbps signaling rates.

Table 1-55. Transceiver Clock Frequencies Signaling Rates in PCIe ×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 PCIe x4 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 PCIe 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 PCIe rateswitch controller. The PCIe rateswitch controller forwards the clock switch completion status to the PCIe interface block. The PCIe 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 1-117 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.


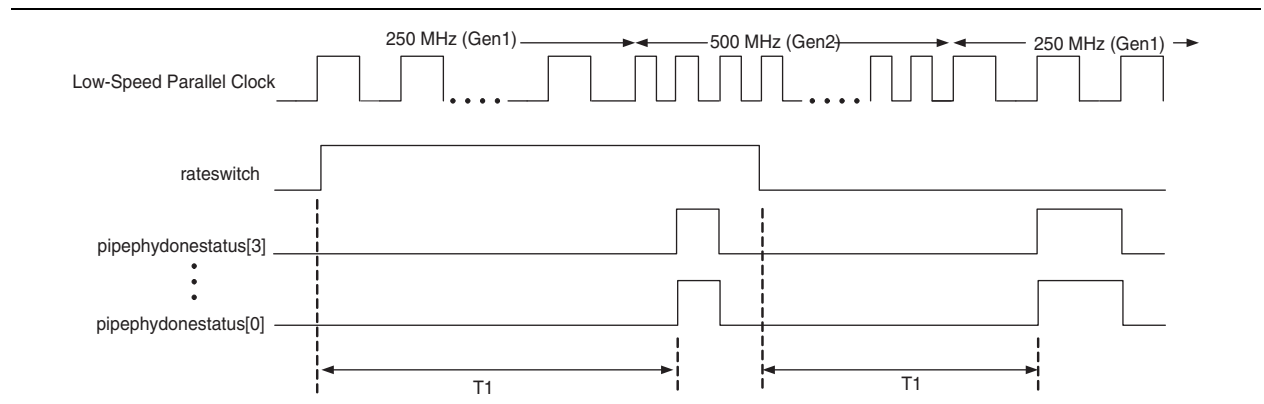
 Time T1 from a transition on the `rateswitch` signal to the assertion of `pipephydonestatus` is pending characterization.

Figure 1-117. Low-Speed Parallel Clock Switching in PCIe x4 Mode



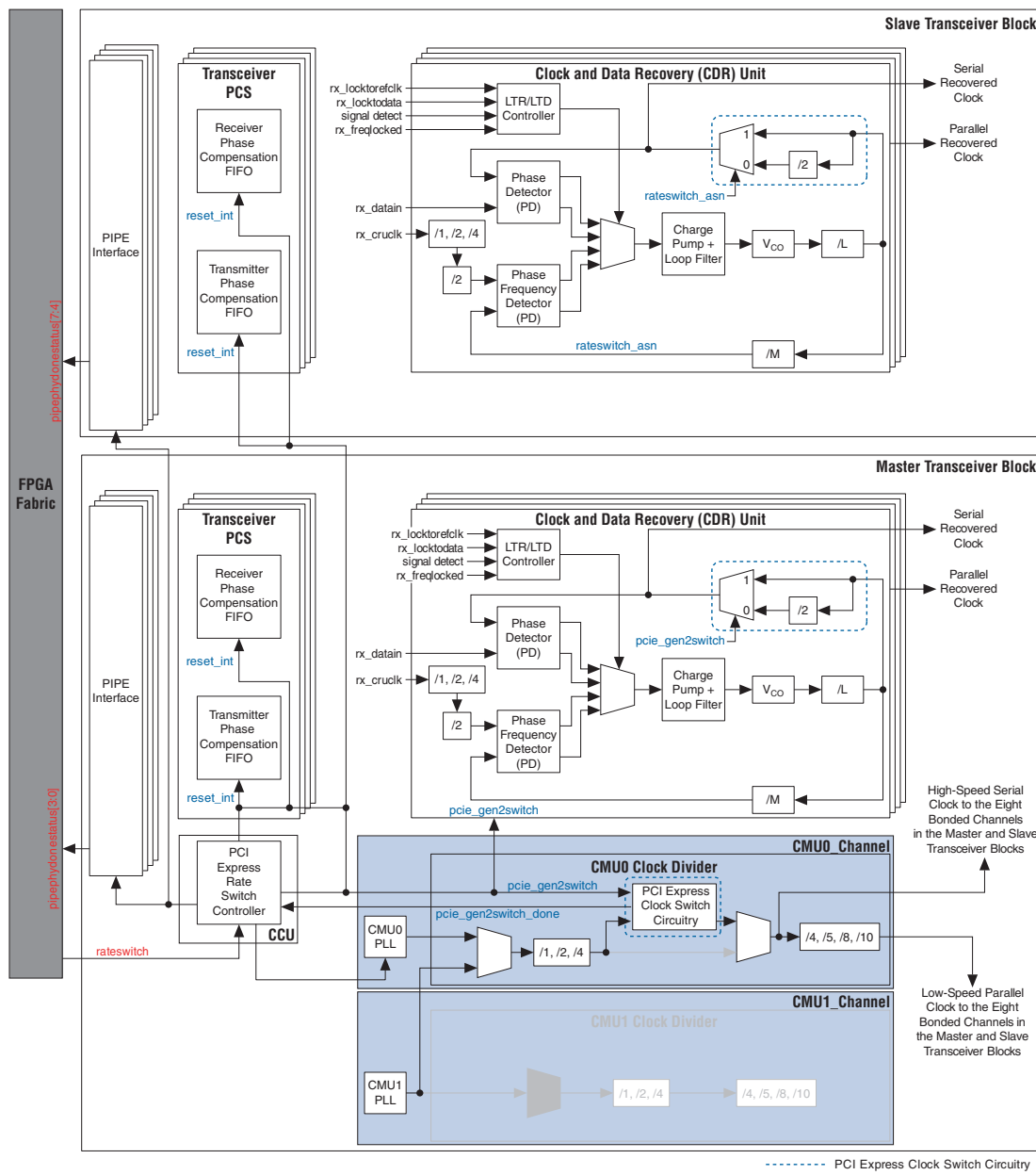
As a result of the signaling rateswitch between Gen1 (2.5 Gbps) and Gen2 (5 Gbps), the FPGA fabric-transceiver interface clock switches between 125 MHz and 250 MHz. The FPGA fabric-transceiver interface clock clocks the read side and write side of the transmitter phase compensation FIFO and the receiver 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 PCIe rateswitch controller automatically disables and resets the phase compensation FIFO pointers of all bonded channels during clock switch. When the PCIe clock switch circuitry in the local clock divider indicates successful clock switch completion, the PCIe rateswitch controller releases the phase compensation FIFO pointer resets.

Dynamic Switch Between Gen1 (2.5 Gbps) and Gen2 (5 Gbps) Signaling Rates in PCIe ×8 Mode

Figure 1-118 shows the PCIe rateswitch circuitry in PCIe ×8 mode configured at Gen2 (5 Gbps) data rate.

Figure 1-118. Dynamic Switch Signaling in PCIe ×8 Mode



In PCIe ×8 mode configured at 5 Gbps data rate, when the PCIe rateswitch controller sees a transition on the `rateswitch` signal, it sends the `pcie_gen2switch` control signal to the PCIe 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 lists the transceiver clock frequencies when switching between the 2.5 Gbps and 5 Gbps signaling rates.

Table 1-56. Transceiver Clock Frequencies Signaling Rates in PCIe ×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

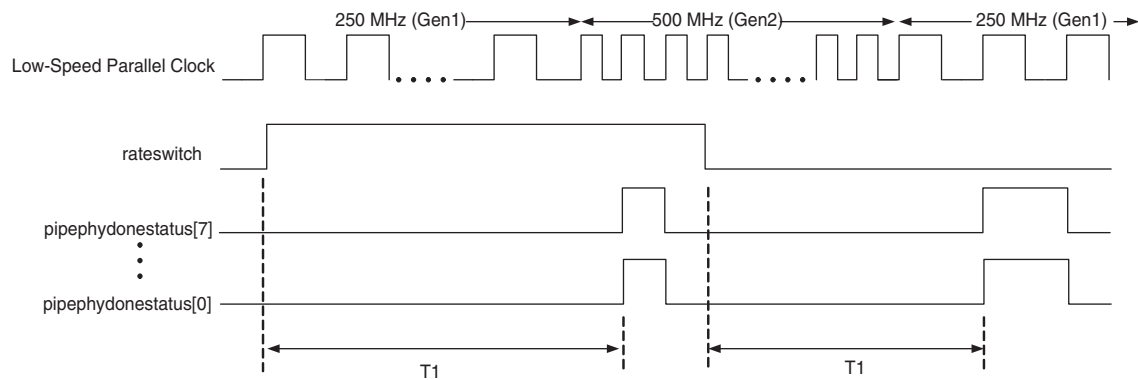
The PCIe clock switch circuitry in the `CMU0` clock divider of the master transceiver block performs the clock switch between 250 MHz and 500 MHz on the low-speed parallel clock when switching between Gen1 (2.5 Gbps) and Gen2 (5 Gbps) signaling rates. It indicates successful completion of clock switch on the `pcie_gen2switchdone` signal to the PCIe rateswitch controller. The PCIe rateswitch controller forwards the clock switch completion status to the PCIe interface block. The PCIe 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 1–119 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.



Time T1 from a transition on the rateswitch signal to the assertion of pipephydonestatus is pending characterization.

Figure 1–119. Low-Speed Parallel Clock Switching in PCIe x8 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 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 PCIe rateswitch controller automatically disables and resets the phase compensation FIFO pointers of all eight bonded channels during clock switch. When the PCIe clock switch circuitry in the local clock divider indicates successful clock switch completion, the PCIe rateswitch controller releases the phase compensation FIFO pointer resets.

PCIe Cold Reset Requirements

The PCIe Base Specification 2.0 defines the following three types of conventional resets to the PCIe 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 PCIe Base Specification 2.0 specifies that `PERST#` must be kept asserted for a minimum of 100 ms (T_{PVPERL}) 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 PCIe system component must become active within 100 ms after `PERST#` is deasserted.


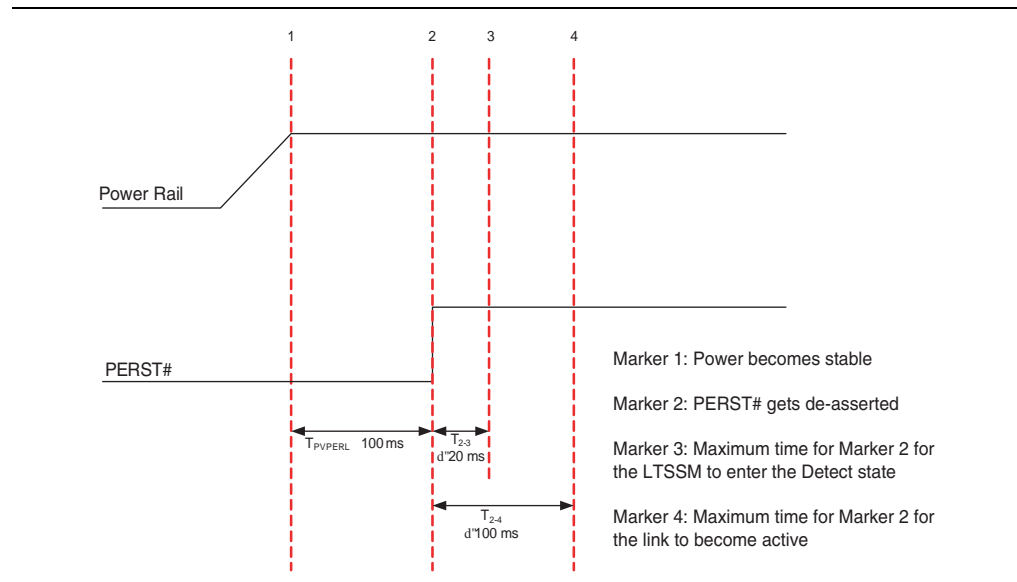
 The link being active is interpreted as the physical layer device coming out of electrical idle in the L0 state of the LTSSM state machine.

Figure 1-120 lists the PCIe cold reset timing requirements.

Figure 1-120. PCIe Cold Reset Requirements



The time taken by a PCIe 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 PCIe soft IP and hard IP)


To meet the PCIe 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 lists the typical configuration times for Stratix IV GX devices when configured using the Fast Passive Parallel (FPP) configuration scheme at the maximum frequency.

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 in 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 PCIe hard IP block, assert the `test_in[6]` port of the PCIe Compiler-generated wrapper file in your design. Asserting this port forces the LTSSM within the hard IP block to transition to these states. The `test_in[6]` port must be asserted for a minimum of 16 ns and less than 24 ms.

 For more information about the PCIe hard IP block, refer to the *PCI Express Compiler User Guide*.

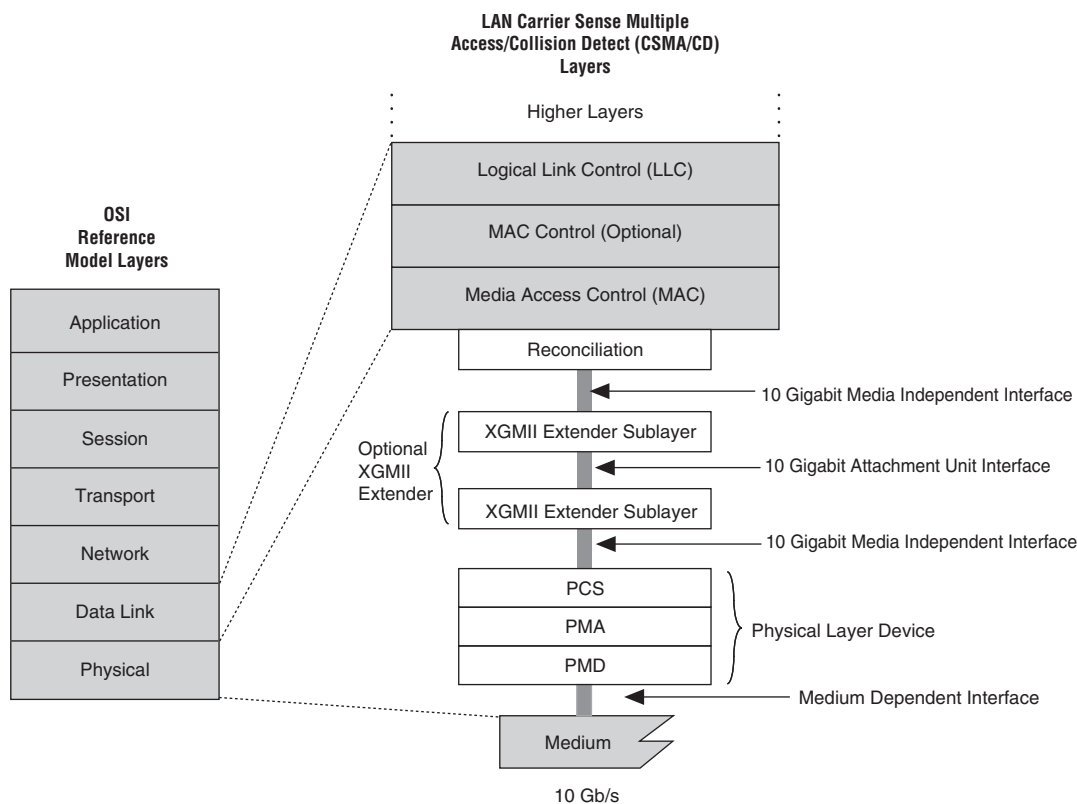
XAUI Mode

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 1-121 shows the relationships between the XGMII and XAUI layers.

Figure 1-121. 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 listed 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
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 1-122](#) 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 1-122. Example of Mapping XGMII Characters to PCS Code Groups

XGMII																		
T/RxD<7..0>			S	Dp	D	D	D	---	D	D	D	D						
T/RxD<15..8>			Dp	Dp	D	D	D	---	D	D	D	T						
T/RxD<23..16>			Dp	Dp	D	D	D	---	D	D	D							
T/RxD<31..24>			Dp	Dp	D	D	D	---	D	D	D							
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

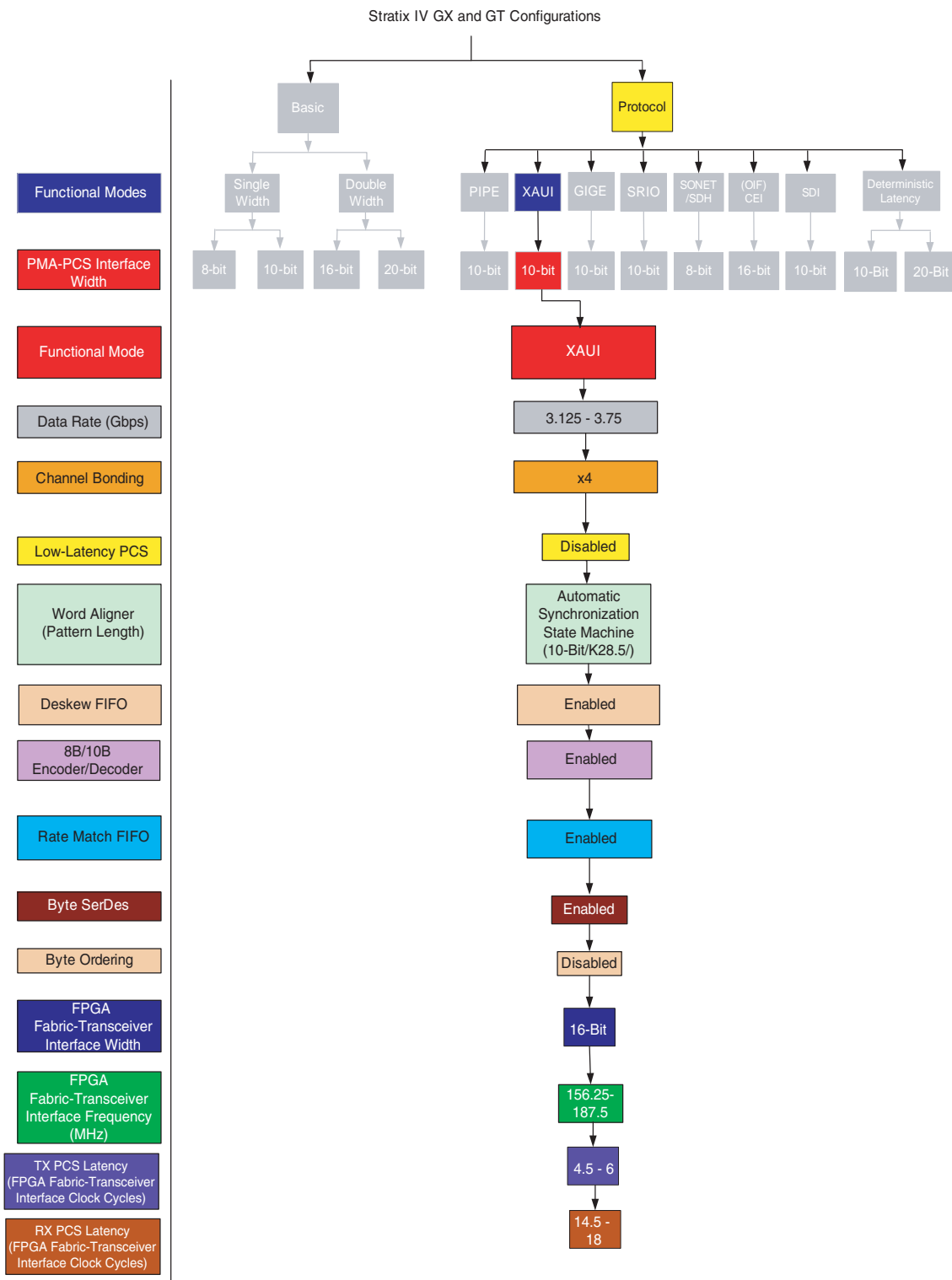
Code	Ordered Set	Number of Code Groups	Encoding
I	Idle		Substitute for XGMII Idle
K	Synchronization column	4	/K28.5/K28.5/K28.5/K28.5/
R	Skip column	4	/K28.0/K28.0/K28.0/K28.0/
A	Align column	4	/K28.3/K28.3/K28.3/K28.3/

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 1-123 shows the XAUI mode configuration supported in Stratix IV GX and GT devices.

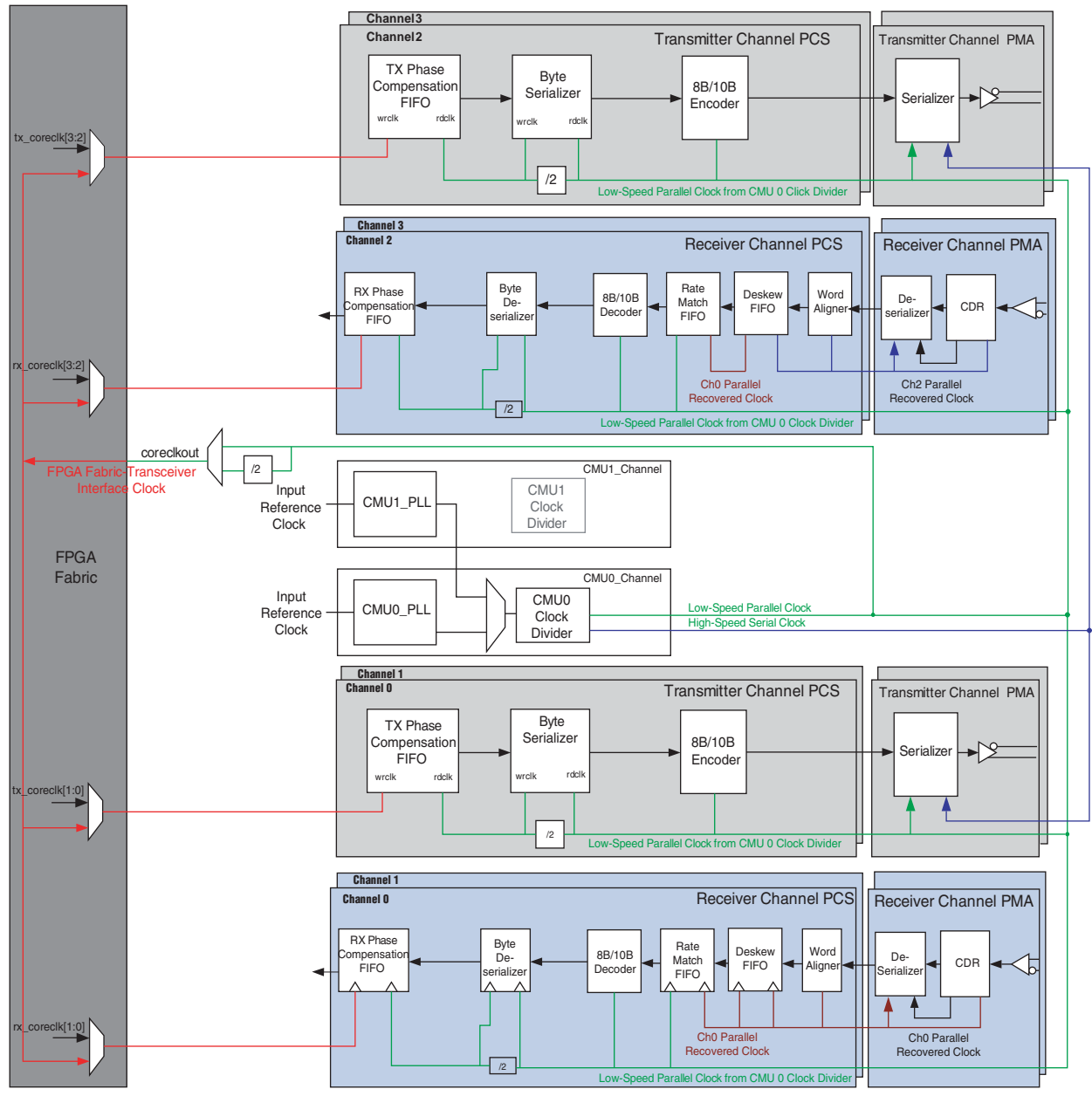
Figure 1-123. Stratix IV GX and GT XAUI Mode Configuration



XAUI Mode Datapath

Figure 1-124 shows the ALTGX megafunction transceiver datapath when configured in XAUI mode.

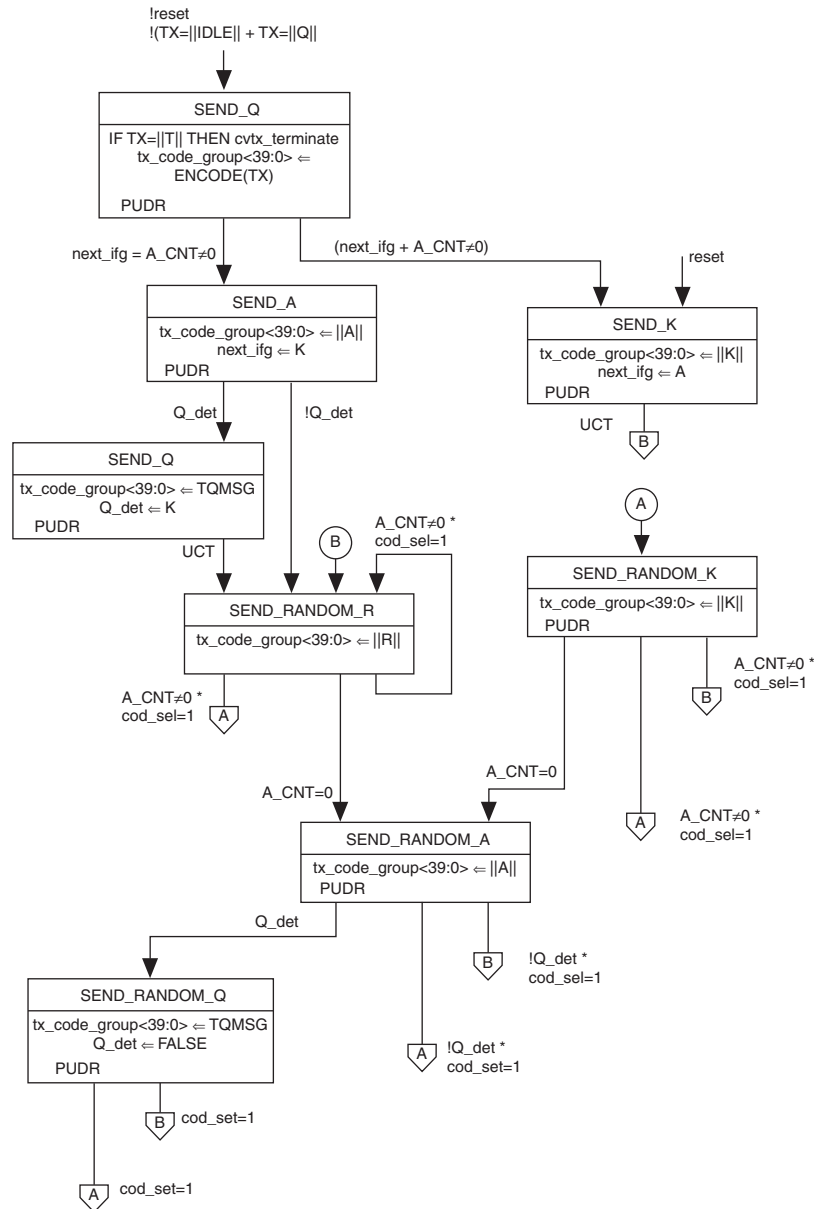
Figure 1-124. Transceiver Datapath in XAUI Mode



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 1-125.

Figure 1-125. XGMII-To-PCS Code Conversion in XAUI Mode (1)



Note to Figure 1-125:

(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
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
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

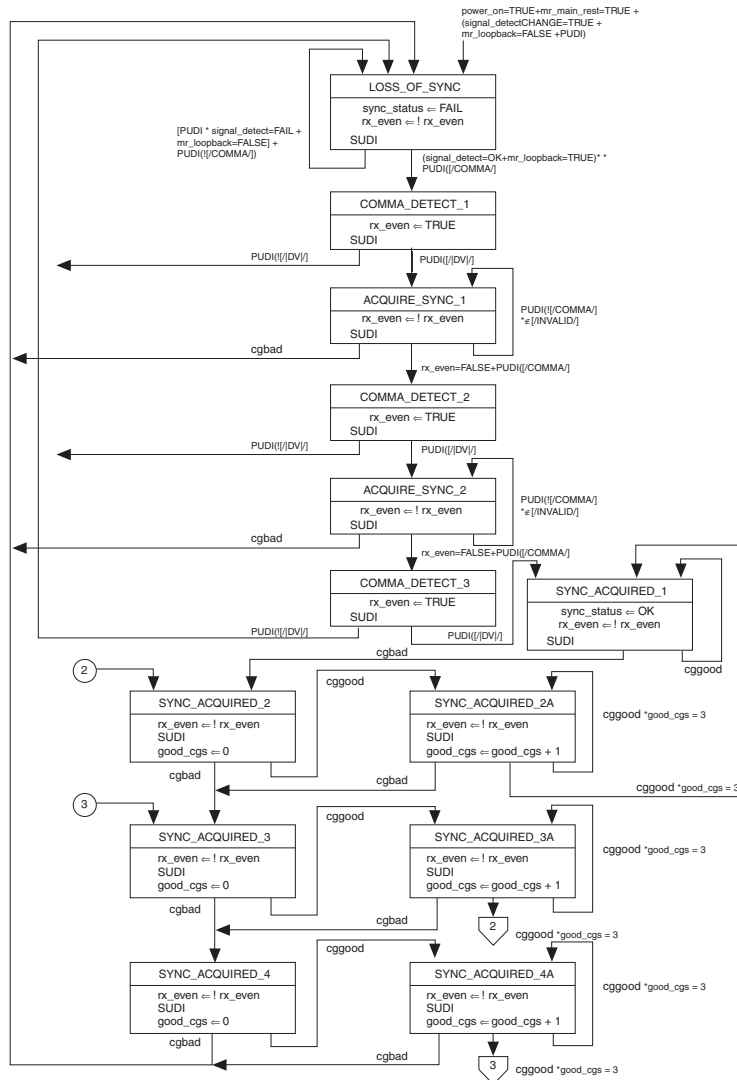
Note to Table 1-58:

(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 1-126.

Figure 1-126. IEEE 802.3ae PCS Synchronization State Diagram ⁽¹⁾



Note to Figure 1-126:

(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.

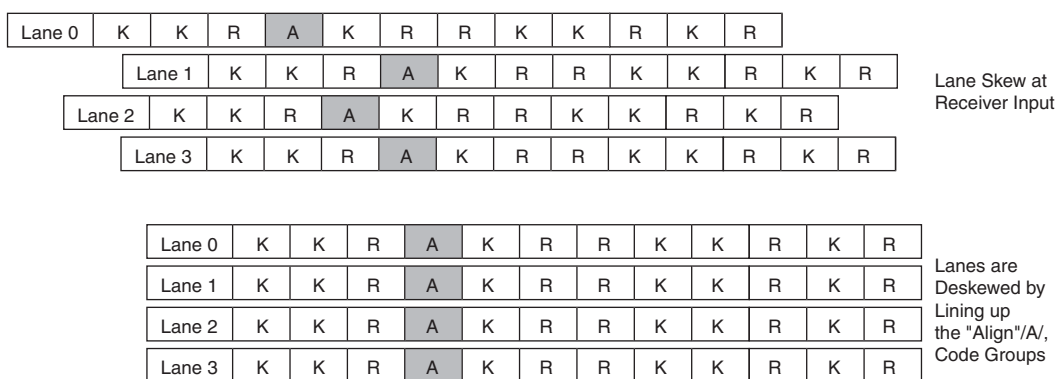
The XAUI protocol requires the physical layer device to implement a deskew circuitry to align all four channels. To enable the deskew circuitry at the receiver to align the four channels, the transmitter sends a /A/ (/K28.3/) code group simultaneously on all four channels during inter-packet gap. The 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 1-127 shows lane skew at the receiver input and how the deskew FIFO uses the /A/ code group to align the channels.

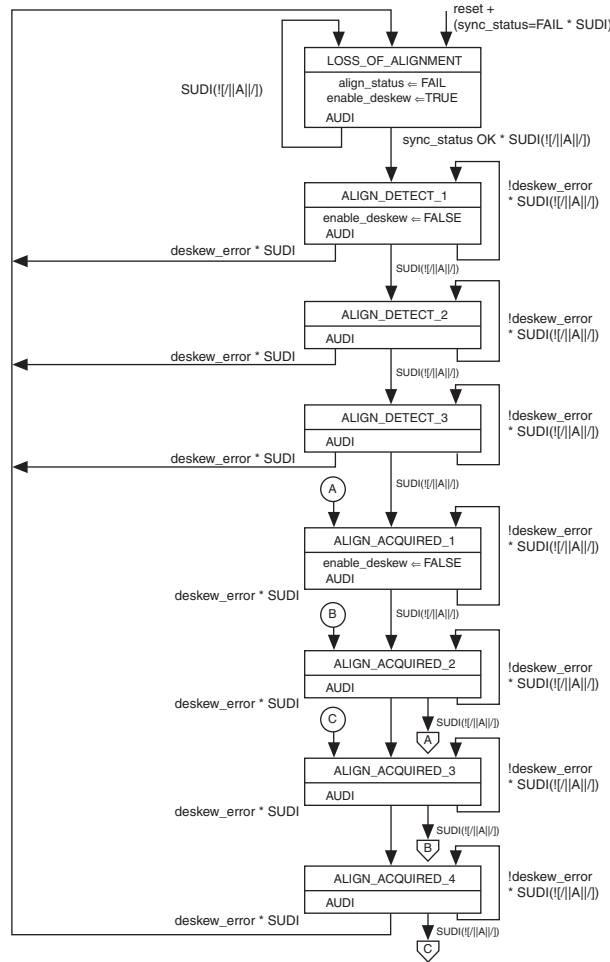
Figure 1-127. Receiver Input Lane Skew in XAUI Mode



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 1-128.

Figure 1-128. Deskew FIFO in XAUI Mode ⁽¹⁾



Note to Figure 1-128:

(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_rmfiodeleted and rx_rmfifoinserted, indicating rate match FIFO deletion and insertion events, respectively, are forwarded to the FPGA fabric. If an ||R|| column is deleted, the rx_rmfiodeleted 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 1-129 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 1-80.

Figure 1-129. Rate Match Deletion in XAUI Mode

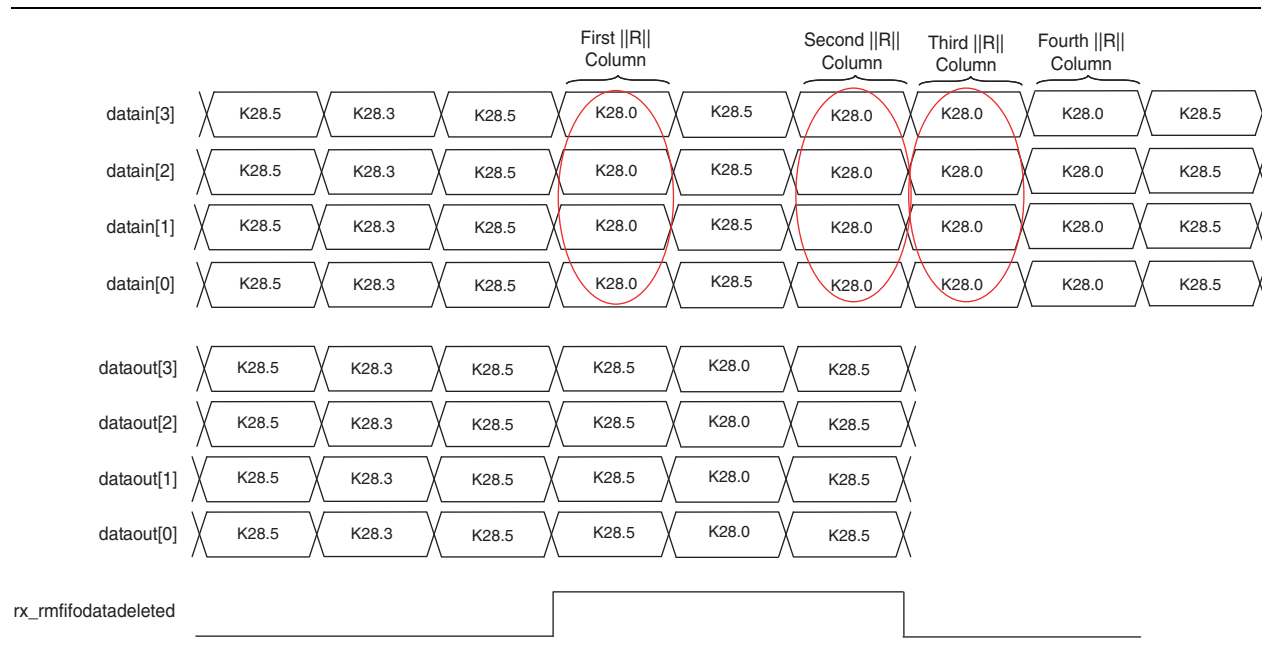
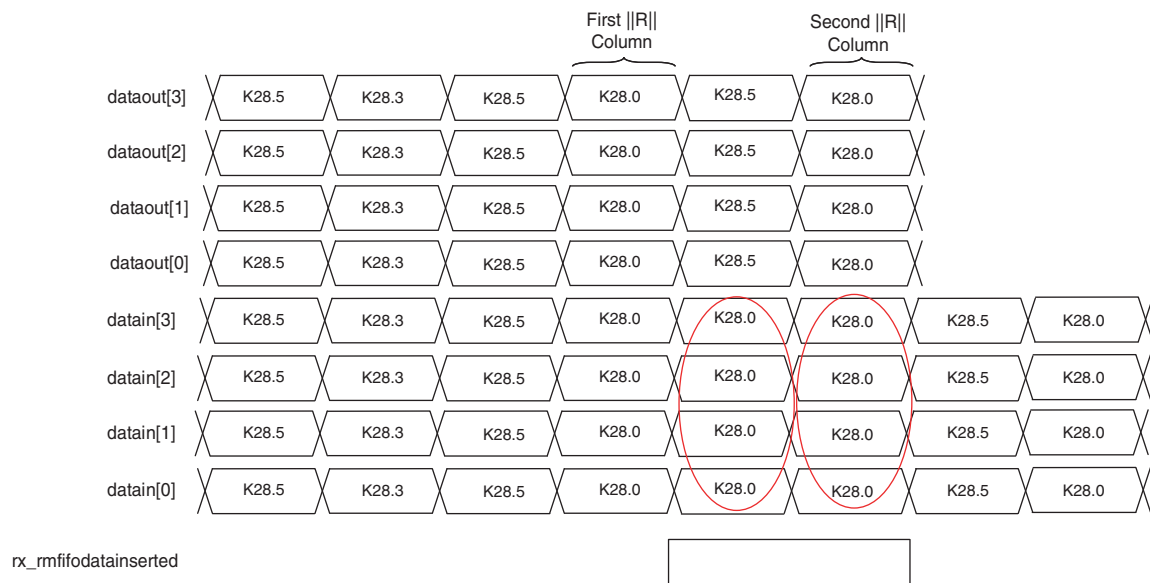


Figure 1-130 shows an example of rate match insertion in the case where two $||R||$ columns are required to be inserted.

Figure 1-130. Rate Match Insertion in XAUI Mode



For more information, refer to “Rate Match (Clock Rate Compensation) FIFO” on page 1-77.

GIGE Mode

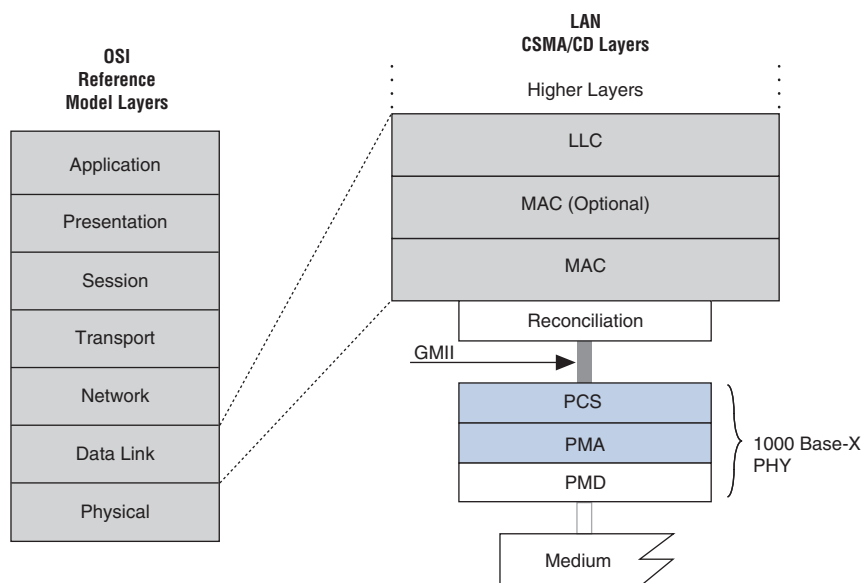
IEEE 802.3 defines the 1000 Base-X PHY as an intermediate, or transition, layer that interfaces various physical media with the media access control (MAC) in a gigabit ethernet system. It shields the MAC layer from the specific nature of the underlying medium. The 1000 Base-X PHY is divided into three 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 1-131 shows the 1000 Base-X PHY position in a Gigabit Ethernet OSI reference model.

Figure 1-131. 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:

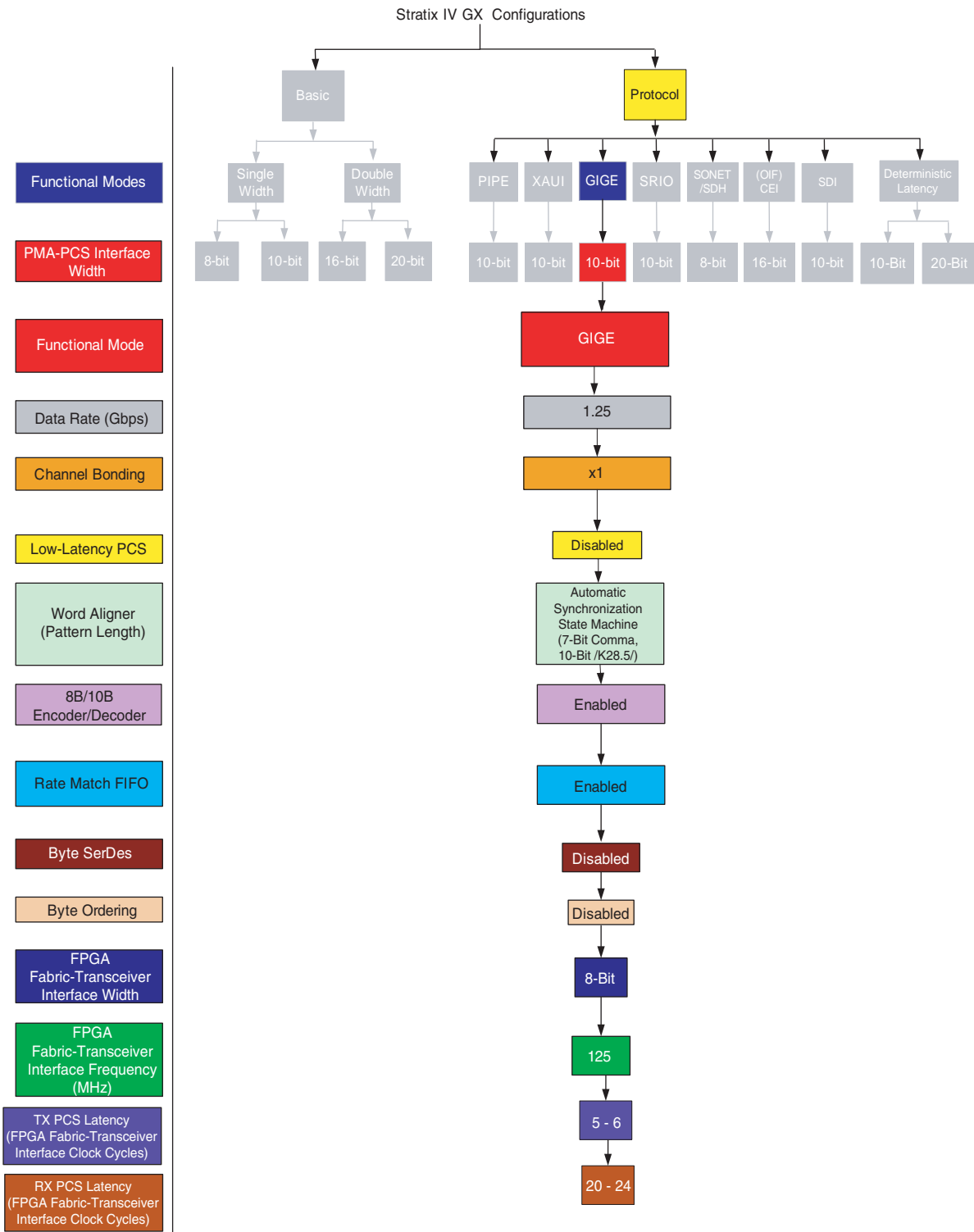
- 8B/10B encoding and decoding
- Synchronization
- Clock recovery from the encoded data forwarded by the receiver PMD
- Optional `rx_recovclkout` port enables recovered clock at the pin level (use with VCXO)
- Serialization and deserialization

 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.

 If you enabled the auto-negotiation state machine in the FPGA core with the rate match FIFO, refer to “Rate Match FIFO” on page 1-171.

Figure 1-132 shows the GIGE mode configuration supported in Stratix IV GX devices.

Figure 1-132. GIGE Mode for Stratix IV GX Devices



GIGE Mode Datapath

Figure 1-133 shows the transceiver datapath when configured in GIGE functional mode.

Figure 1-133. GIGE Mode Datapath

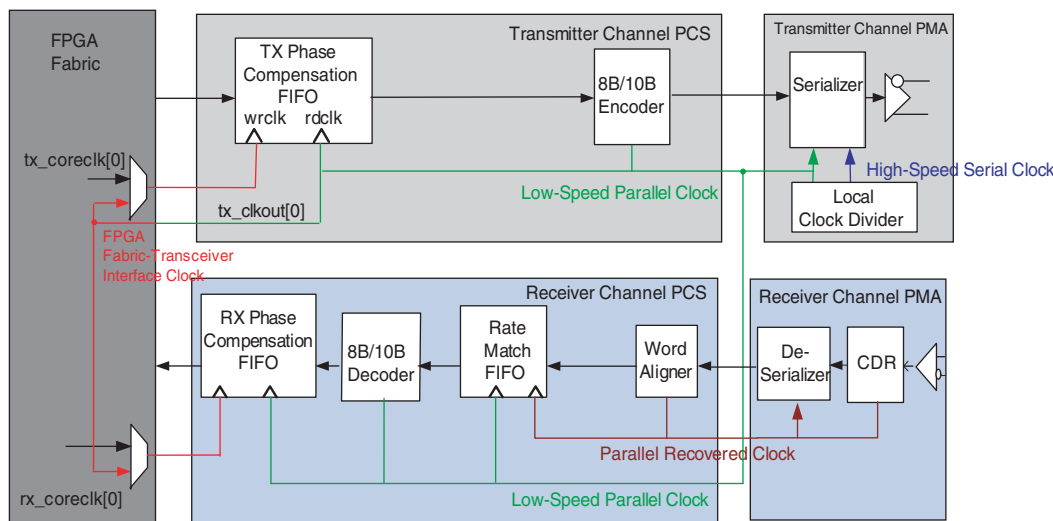


Table 1-62 lists the transceiver datapath clock frequencies in GIGE functional mode.

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 1-23.

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.


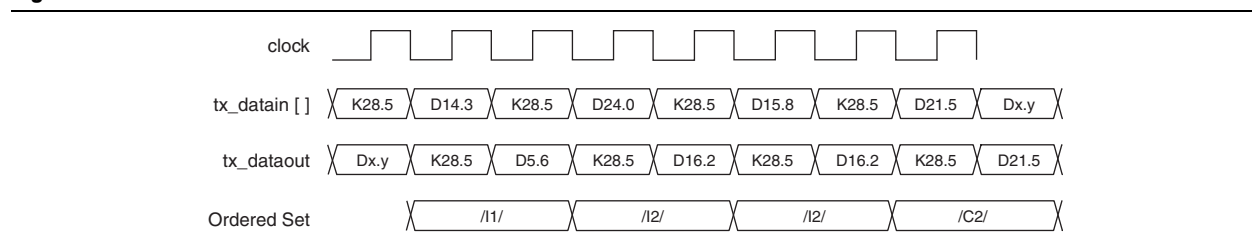
 Note that /D14.3/, /D24.0/, and /D15.8/ are replaced by /D5.6/ or /D16.2/ (for /I1/, /I2/ ordered sets). /D21.5/ (part of the /C1/ order set) is not replaced.

Figure 1-134 shows the automatic idle ordered set generation.

Figure 1-134. Automatic Ordered Set Generation

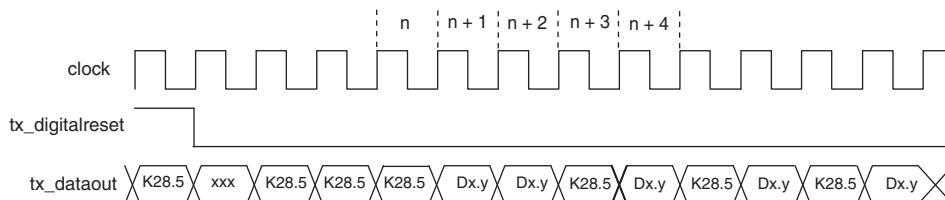
Reset Condition

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 1-135 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 1-135. 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.

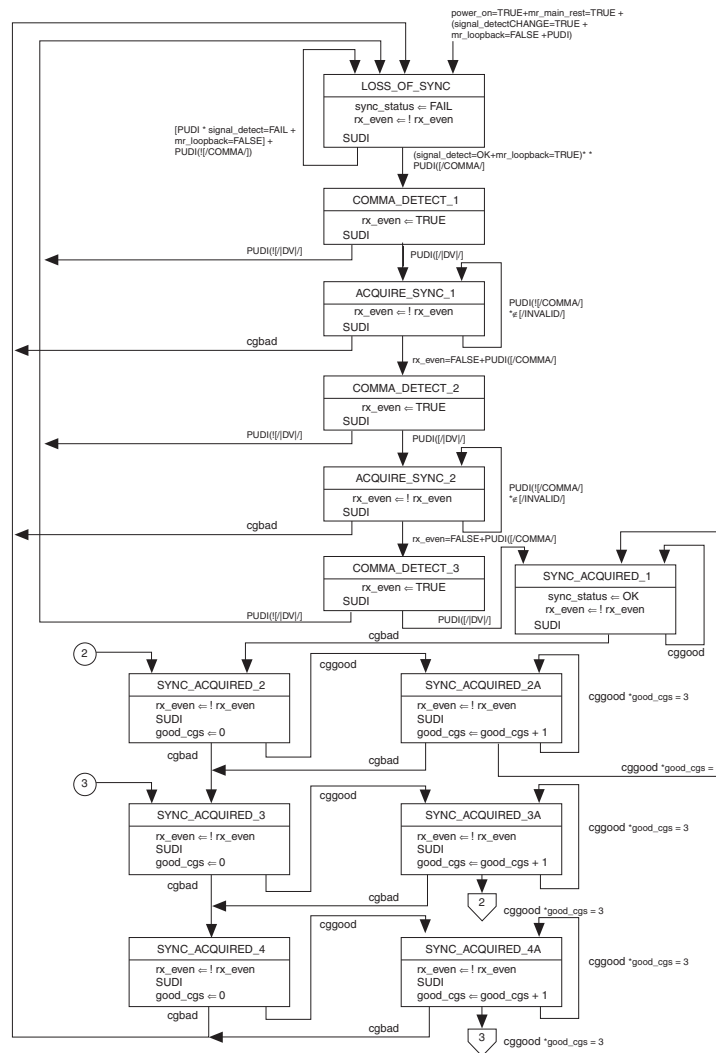
Table 1-64 lists the synchronization state machine parameters when configured in GIGE mode.

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 1-136 shows the synchronization state machine implemented in GIGE mode.

Figure 1-136. Synchronization State Machine in GIGE Mode (1)



Note to Figure 1-136:

(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_rmifodatadeleted and rx_rmifodatainserted, indicating rate match FIFO deletion and insertion events, respectively, are forwarded to the FPGA fabric. Both the rx_rmifodatadeleted and rx_rmifodatainserted flags are asserted for two clock cycles for each deleted and inserted /I2/ ordered set, respectively.

Figure 1-137 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 1-137. Rate Match Deletion in GIGE Mode

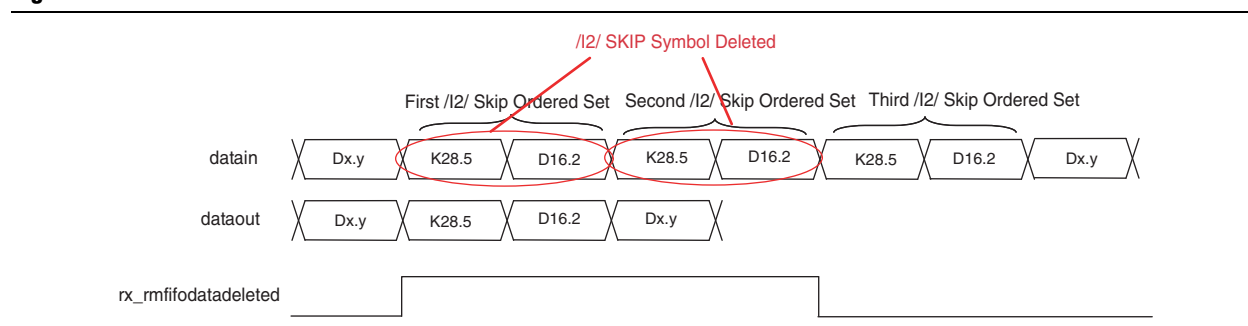
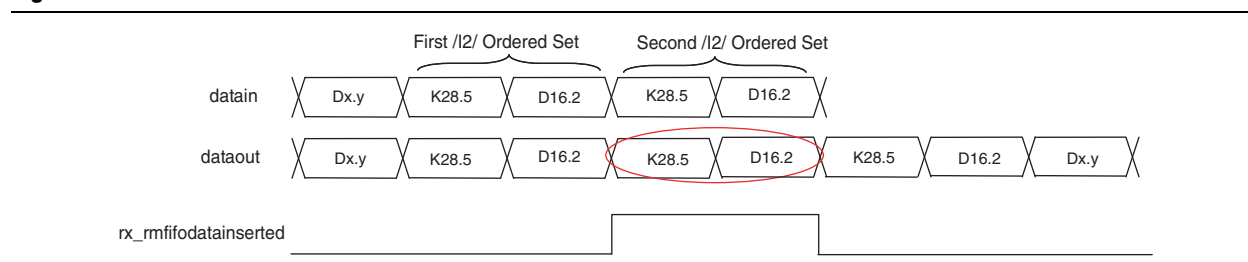



Figure 1-138 shows an example of rate match FIFO insertion in the case where one symbol is required to be inserted. Because the rate match FIFO can only delete /I2/ ordered set, it inserts one /I2/ ordered set (two symbols inserted).

Figure 1-138. Rate Match Insertion in GIGE Mode



For more information, refer to “Rate Match (Clock Rate Compensation) FIFO” on page 1-77.

 If you have the auto-negotiation state machine in the FPGA, note that the rate match FIFO is capable of inserting or deleting the first two bytes (/K28.5//D2.2/) of /C2/ ordered sets during auto-negotiation. However, the insertion or deletion of the first two bytes of /C2/ ordered sets can cause the auto-negotiation link to fail. For more information, refer to the [Altera Knowledge Base Support Solution](#).

SONET/SDH Mode

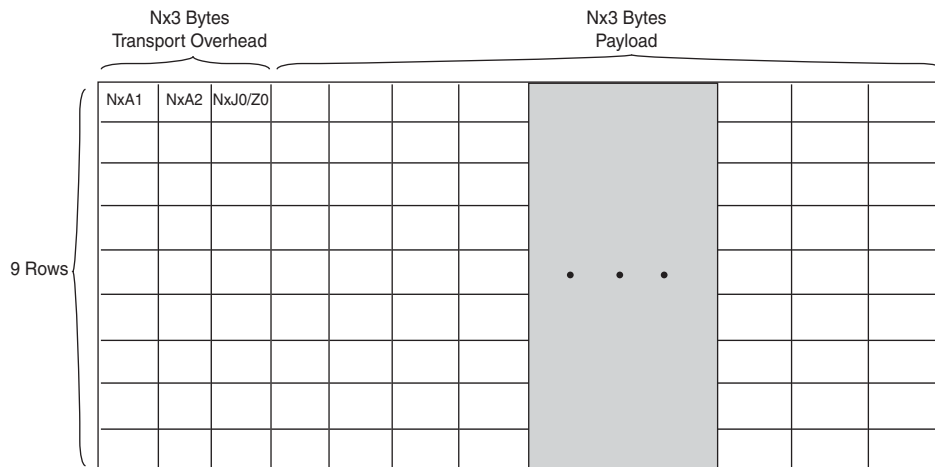
SONET/SDH is one of the most common serial-interconnect protocols used in backplanes deployed in communications and telecom applications. SONET/SDH defines various optical carrier (OC) sub-protocols for carrying signals of different capacities through a synchronous optical hierarchy.

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 μ s.

Figure 1–139 shows the SONET/SDH frame structure.

Figure 1–139. 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 μ s. 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

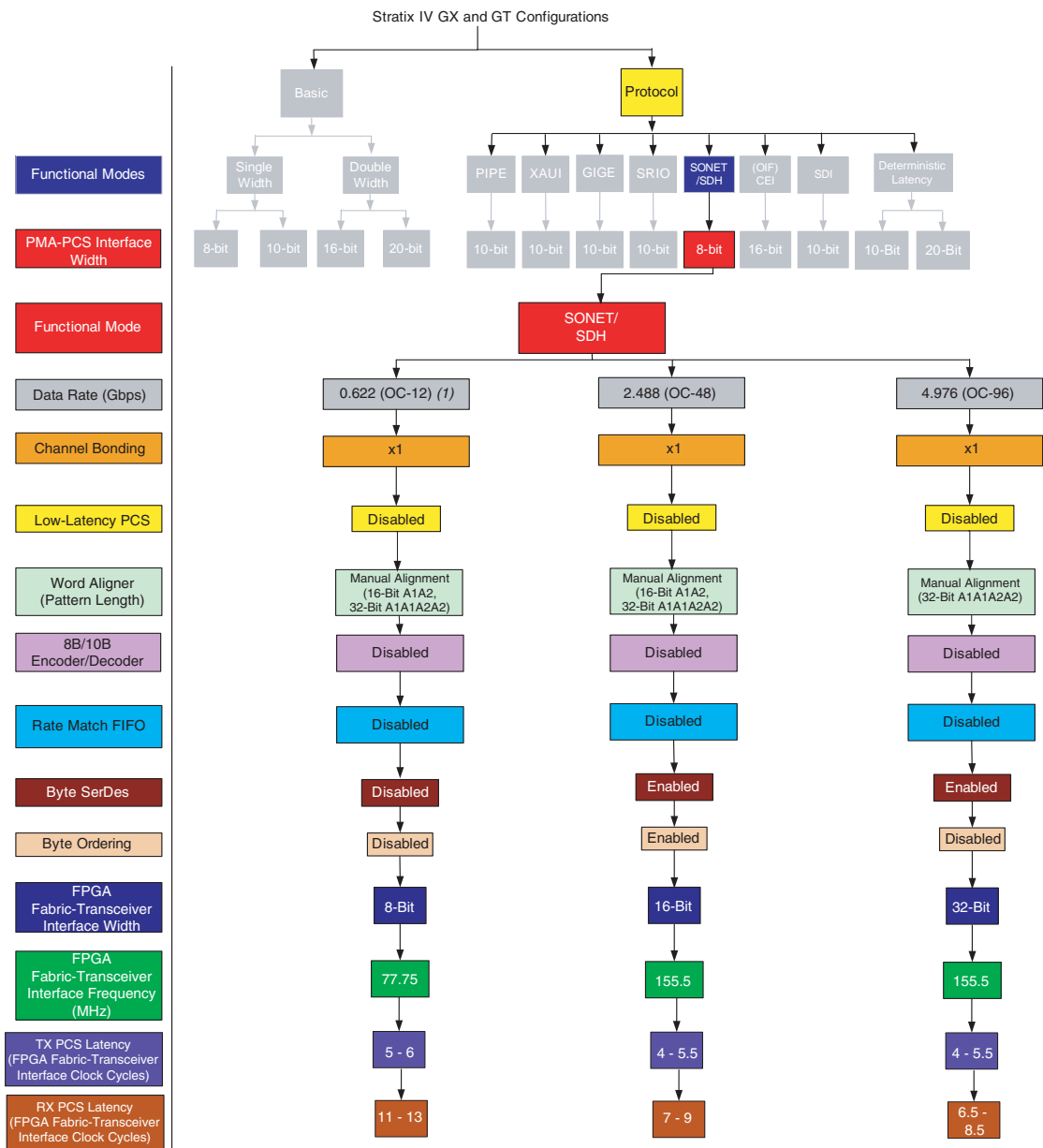
You can employ Stratix IV GX and GT transceivers as physical layer devices in a SONET/SDH system. These transceivers provide support for SONET/SDH protocol-specific functions and electrical features; for example, alignment to A1A2 or A1A1A2A2 pattern.

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 1-140 shows SONET/SDH mode configurations supported in Stratix IV GX and GT devices.

Figure 1-140. SONET/SDH Mode Configurations in Stratix IV GX and GT Devices



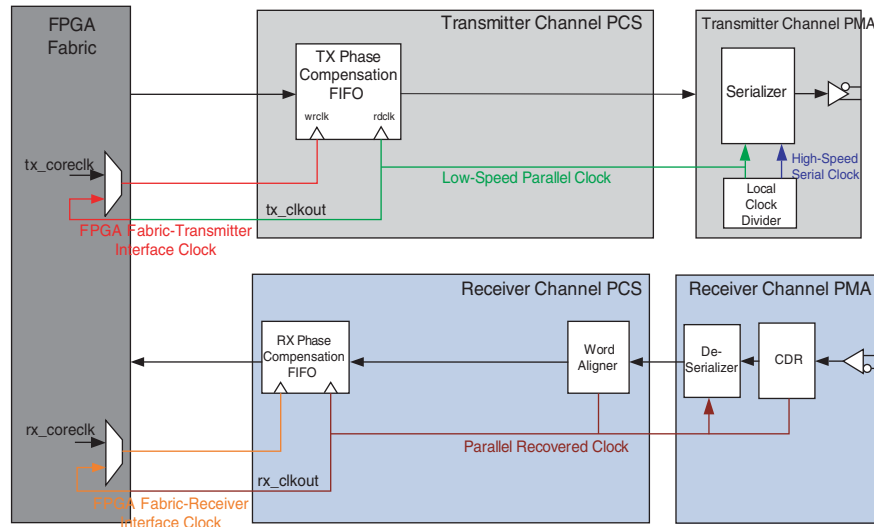
Note to Figure 1-41:

(1) This is not supported in Stratix IV GT devices.

SONET/SDH OC-12 Datapath

Figure 1-141 shows the transceiver datapath when configured in SONET/SDH OC-12 mode.

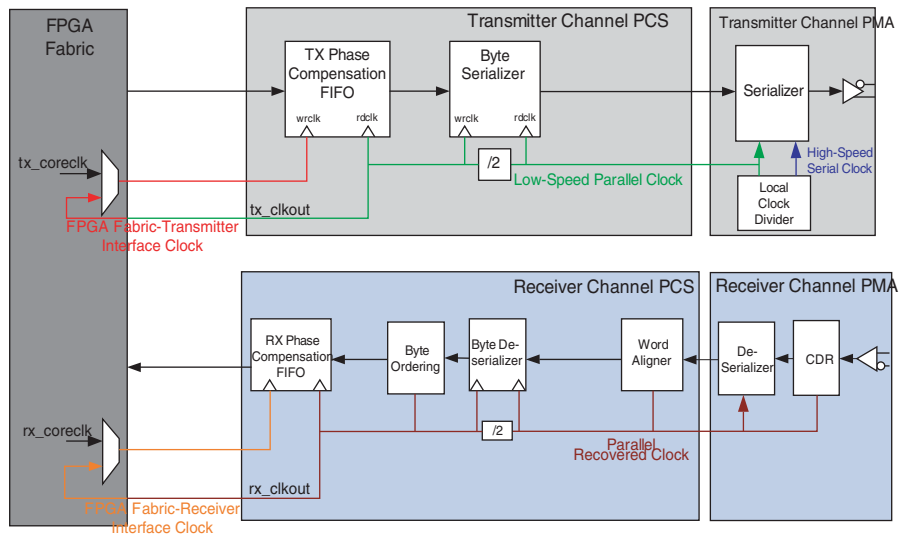
Figure 1-141. SONET/SDH OC-12 Datapath



SONET/SDH OC-48 Datapath

Figure 1-142 shows the transceiver datapath when configured in SONET/SDH OC-48 mode.

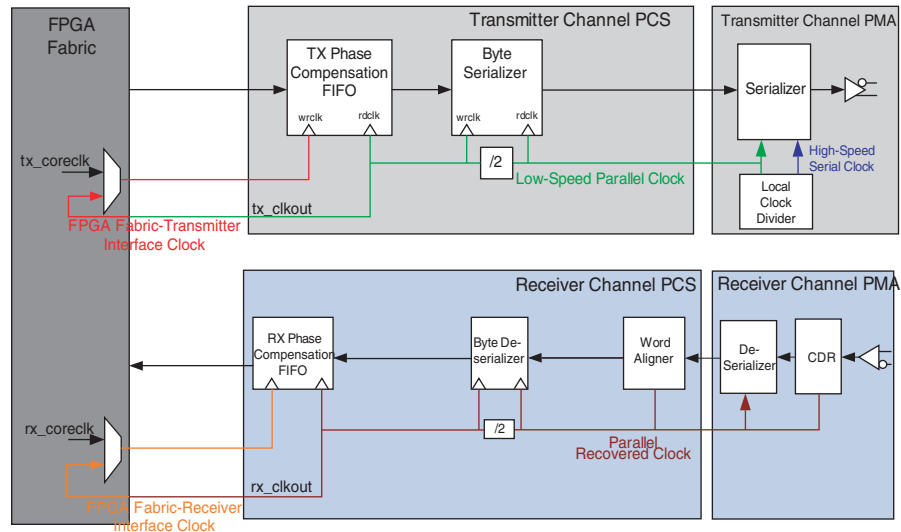
Figure 1-142. SONET/SDH OC-48 Datapath



SONET/SDH OC-96 Datapath

Figure 1-143 shows the transceiver datapath when configured in SONET/SDH OC-96 mode.

Figure 1-143. 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 1-177 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 1-60.

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_ala2size input port to the transceiver. A low level on the rx_ala2size port configures the word aligner to align to a 16-bit A1A2 pattern; a high level on the rx_ala2size 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 Plug-In Manager 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 1-60.

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 1-21 and [“Byte Deserializer”](#) on page 1-92, 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 1-97.

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 1-144](#). Insertion of this PAD character enables the byte ordering block to restore the correct byte order.


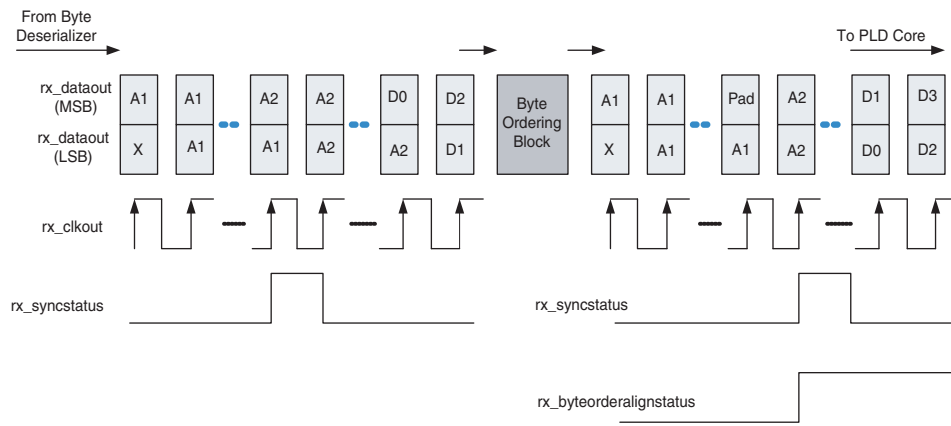
 The PAD character is defaulted to the A1 byte of the A1A2 alignment pattern.

Figure 1-144. 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 lists the ALTGX configurations supported by Stratix IV transceivers in SDI mode.

Table 1-66. ALTGX Configurations in SDI Mode

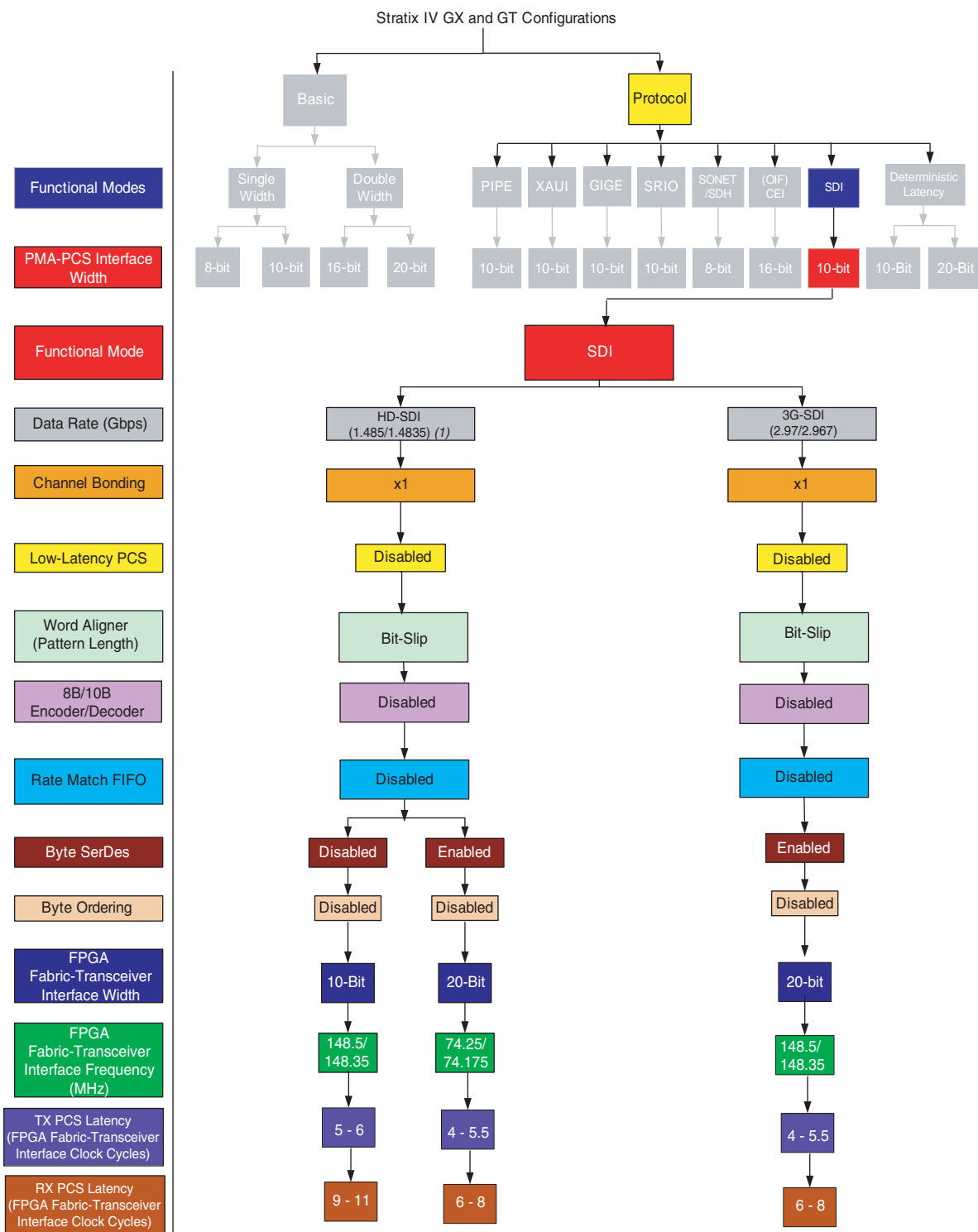
Configuration	Data Rate (Mbps)	REFCLK Frequencies (MHz)	FPGA Fabric-Transceiver Interface Width
HD ⁽¹⁾	1485	74.25, 148.5	10 bit and 20 bit
	1483.5	74.175, 148.35	10 bit and 20 bit
3G ⁽²⁾	2970	148.5, 297	Only 20-bit interface allowed in 3G
	2967	148.35, 296.7	Only 20-bit interface allowed in 3G

Notes to Table 1-66:

- (1) Not supported by Stratix IV GT devices.
- (2) Stratix IV GT devices only support the 3G configuration.

Figure 1-145 shows SDI mode configurations supported in Stratix IV GX and GT devices.

Figure 1-145. SDI Mode



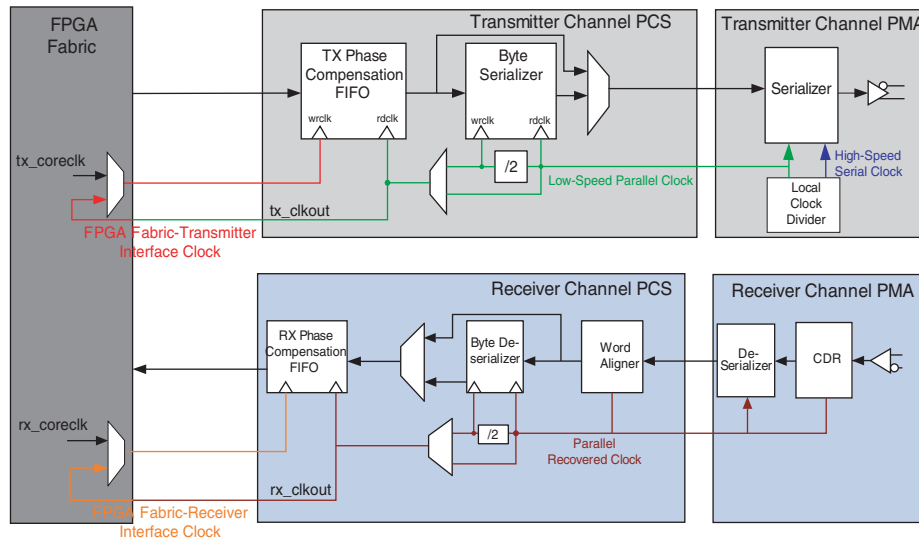
Note to Figure 1-45:

(1) Not supported in Stratix IV GT devices.

SDI Mode Datapath


Figure 1-146 shows the transceiver datapath when configured in SDI mode.

Figure 1-146. 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

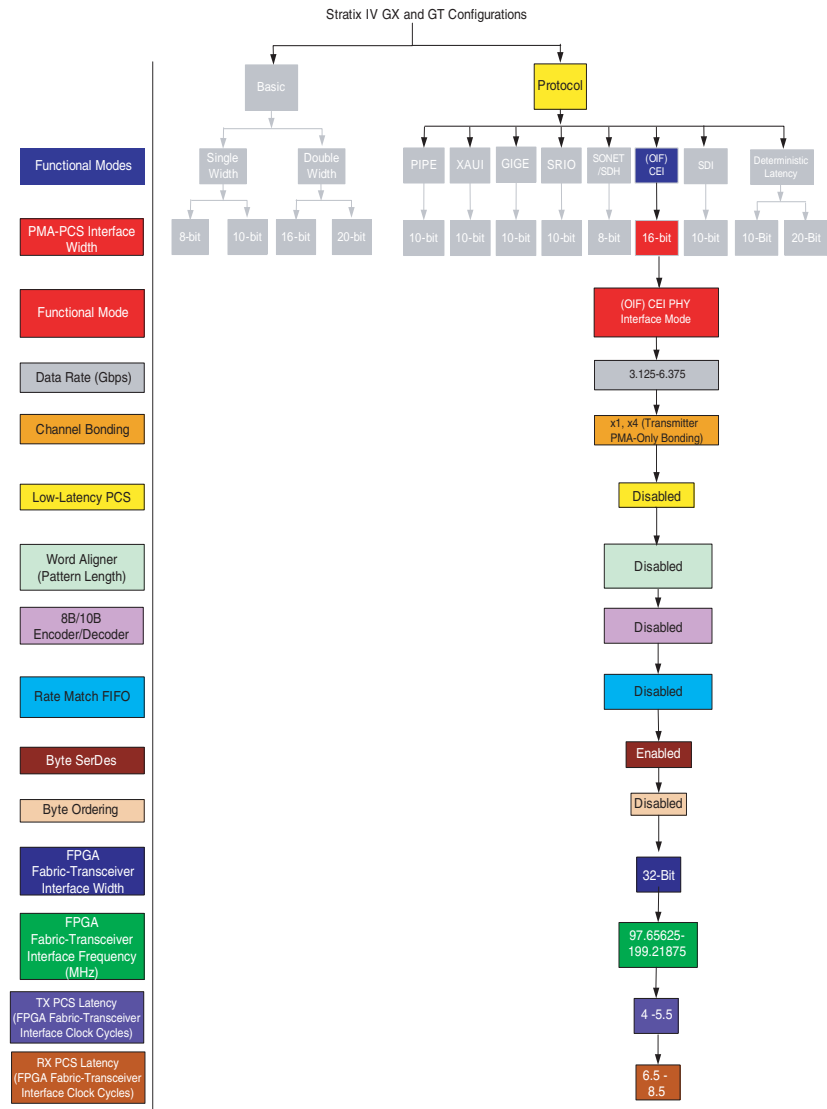
In SDI systems, the word aligner in the receiver datapath is not useful because word alignment and framing happens after de-scrambling. Altera recommends driving the ALTGX megafunction `rx_bitslip` signal low to avoid having the word aligner insert bits in the received data stream.

(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 1-147 shows (OIF) CEI PHY interface mode configurations supported in Stratix IV GX and GT devices.

Figure 1-147. (OIF) CEI PHY Interface Mode for Stratix IV GX and GT Devices



(OIF) CEI PHY Interface Mode Datapath

Figure 1-148 shows the ALTGX megafunction transceiver datapath when configured in (OIF) CEI PHY interface mode.

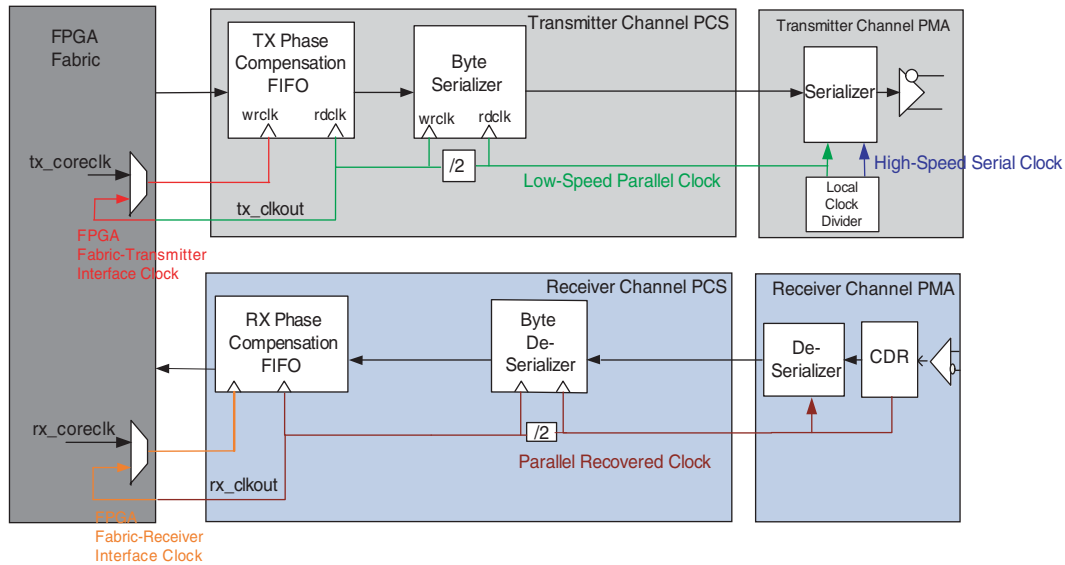
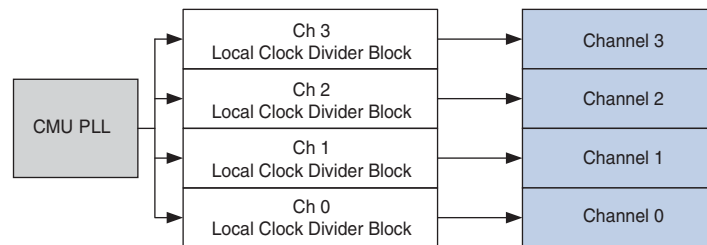
Figure 1-148. (OIF) CEI PHY Interface Mode Datapath

Figure 1-149 shows transceiver clocking in (OIF) CEI PHY interface mode.

Figure 1-149. Transceiver Clocking in (OIF) CEI PHY Interface Mode

Transceiver Block Clocking with the
Use central clock divider to improve
transmitter jitter option disabled

**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:

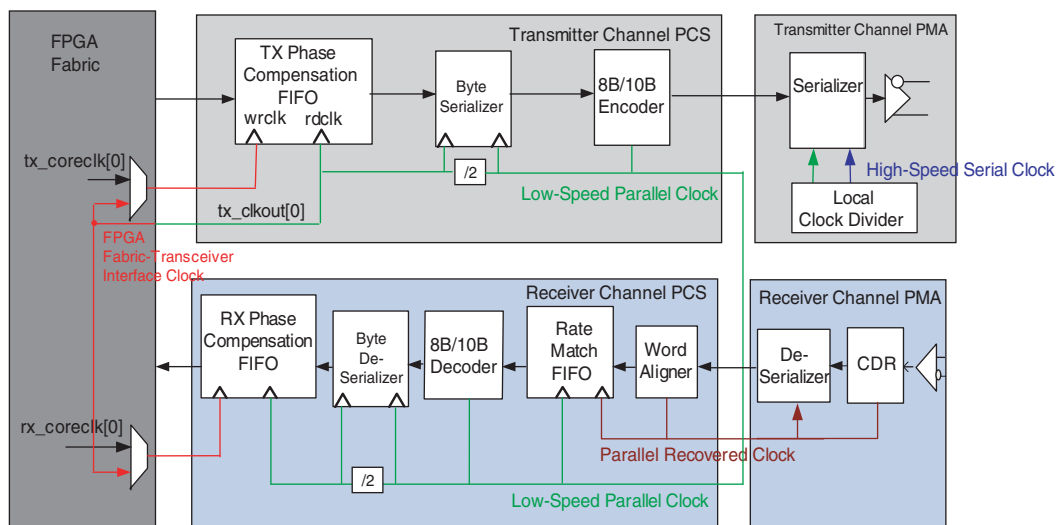
- 1.25 Gbps
- 2.5 Gbps
- 3.125 Gbps

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 1-150 shows the ALTGX transceiver datapath when configured in Serial RapidIO mode.

Figure 1-150. 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.

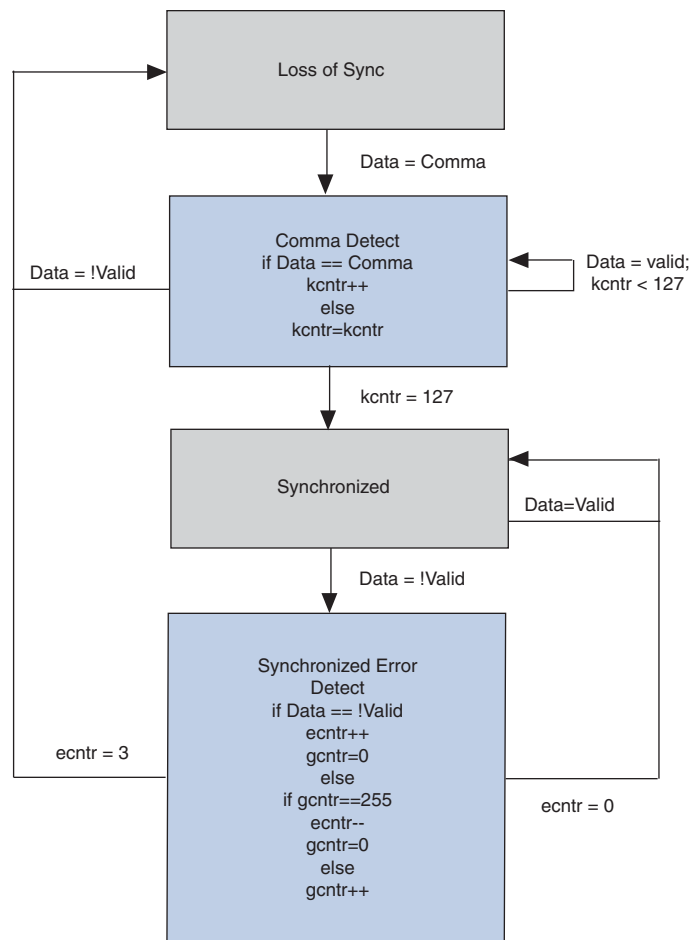
Table 1-67 lists the ALTGX megafunction synchronization state machine parameters when configured in Serial RapidIO mode.

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 1-151 shows a conceptual view of the synchronization state machine implemented in Serial RapidIO functional mode.


Figure 1-151. 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_rmifodatadeleted` and `rx_rmifodatainserted`, indicate that rate match FIFO deletion and insertion events, respectively, are forwarded to the FPGA fabric.

Figure 1-152 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 1-152. Rate Match FIFO Deletion with One Skip Pattern Deleted

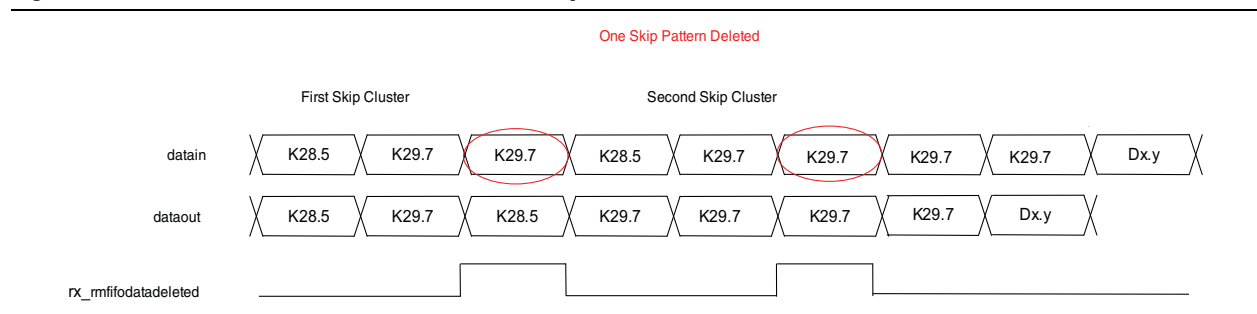
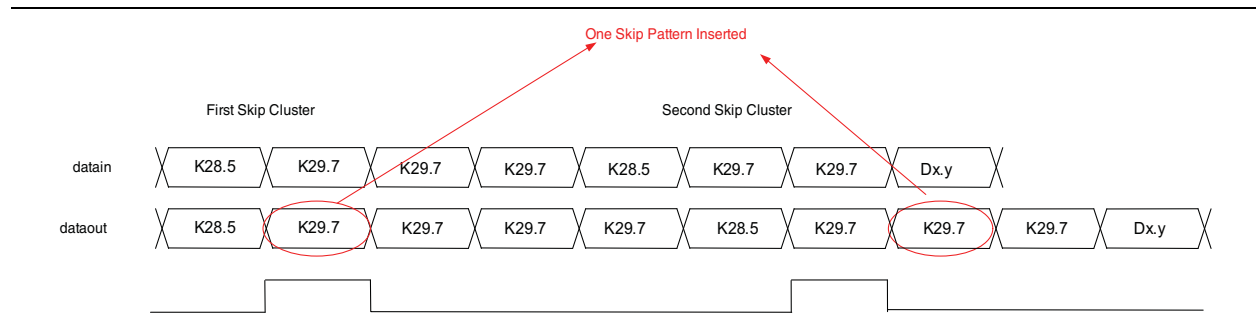


Figure 1-153 shows an example of rate match FIFO insertion in the case where one skip pattern is required to be inserted. In this example, the first skip cluster has a /K28.5/ control pattern followed by three /K29.7/ skip patterns. The second skip cluster has a /K28.5/ control pattern followed by two /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 1-153. Rate Match FIFO Deletion with One Skip Pattern Inserted



Two flags, `rx_rmfifoempty` and `rx_rmfiifull`, are forwarded to the FPGA fabric to indicate rate match FIFO full and empty conditions. For more information about the behavior of these two signals, refer to “Rate Match FIFO in Basic Single-Width Mode” on page 1-84.


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 *Dynamic Reconfiguration in Stratix IV Devices* chapter. For more information about the reset sequence to follow in PMA-Direct mode, refer to the *Reset Control and Power Down in Stratix IV Devices* chapter.

The term ‘PMA-Direct’ is used to describe various configurations in this mode.


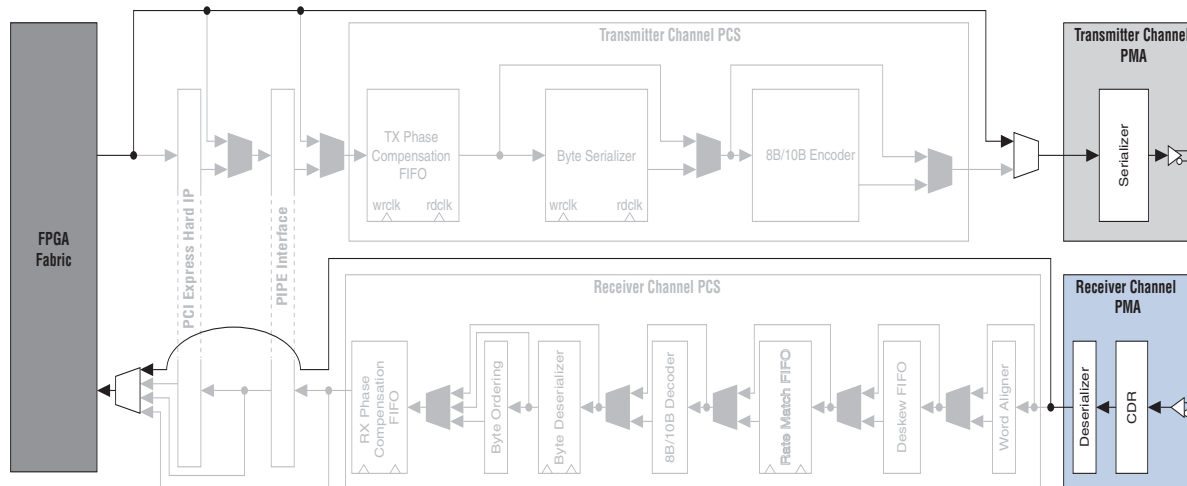
 In Basic (PMA Direct) mode, all the PCS blocks are bypassed; therefore, any PCS-type features (for example, phase compensation FIFOs, byte serializer, 8B/10B encoder/decoder, word aligner, deskew FIFO, rate match FIFO, byte deserializer, and byte ordering), must be implemented in the FPGA fabric. In Basic (PMA Direct) mode, you must create your own logic to support PCS functionality.

Figure 1-154 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 1-154. Stratix IV GX and GT Transceiver Configured in Basic (PMA Direct) Mode



Note to Figure 1-154:

- (1) The grayed out blocks shown in Figure 1-154 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 1-154 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 lists the Stratix IV GX and GT PLD-PMA interface widths and data rates supported in Basic (PMA Direct) $\times 1/\times N$ single-width and double-width modes.


Table 1-68. FPGA Fabric-PMA Interface Widths and Data Rates Supported in Basic (PMA Direct) $\times 1/\times N$ 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, and M3 Speed Grades	C4 Speed Grade	I1, I2, and I3 Speed Grades
$\times 1/\times N$ Single-width mode	8 bit	0.6 Gbps to 2.6 Gbps	0.6 Gbps to 2.6 Gbps	0.6 Gbps to 2.6 Gbps	600 Mbps to 2.6 Gbps
	10 bit	0.6 Gbps to 3.25 Gbps	0.6 Gbps to 3.25 Gbps	0.6 Gbps to 3.25 Gbps	600 Mbps to 3.25 Gbps
$\times 1/\times N$ Double-width mode	16 bit	1.0 Gbps to 5.2 Gbps	1.0 Gbps to 5.2 Gbps	1.0 Gbps to 5.0 Gbps	1.0 Gbps to 5.2 Gbps
	20 bit	1.0 Gbps to 6.5 Gbps	1.0 Gbps to 6.5 Gbps	1.0 Gbps to 5.0 Gbps	1.0 Gbps to 6.5 Gbps

Basic (PMA Direct) $\times 1$ 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) $\times 1$ configuration, use their local clock dividers to provide clock to their respective transmitter channels.


 For information about clocking restrictions in Basic (PMA Direct) $\times 1$ mode, refer to the “Non-Bonded Basic (PMA Direct) Mode Channel Configurations” section in the *Transceiver Clocking in Stratix IV Devices* chapter.

 For information about routing the clocks to transceiver channels in Basic (PMA Direct) $\times 1$ mode, refer to the *Transceiver Clocking in Stratix IV Devices* chapter.

Basic (PMA Direct) $\times N$ 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 **$\times N$** . 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 $\times N_Top$ or $\times N_Bottom$ clock line.

In this mode, if you use a CMU PLL to generate the transceiver channel datapath interface 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) $\times N$ mode, refer to the “Non-Bonded Basic (PMA Direct) Mode Channel Configurations” section in the *Transceiver Clocking in Stratix IV Devices* chapter.

- For more information about combining multiple transceiver channels, refer to the *Configuring Multiple Protocols and Data Rates in Stratix IV Devices* 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 1-190—available in all functional modes except PCIe mode
- “Parallel Loopback” on page 1-191—available in either single-width or double-width modes.
- “Reverse Serial Loopback” on page 1-193—available in Basic mode only
- “Reverse Serial Pre-CDR Loopback” on page 1-194—available in Basic mode only
- “PCIe Reverse Parallel Loopback” on page 1-194—available in PCIe mode

Serial Loopback

The **serial loopback** option is available for all functional modes except PCIe mode. [Figure 1-155](#) 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_serialpbken` port to dynamically enable serial loopback on a channel-by-channel basis. Set the `rx_serialpbken` 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.


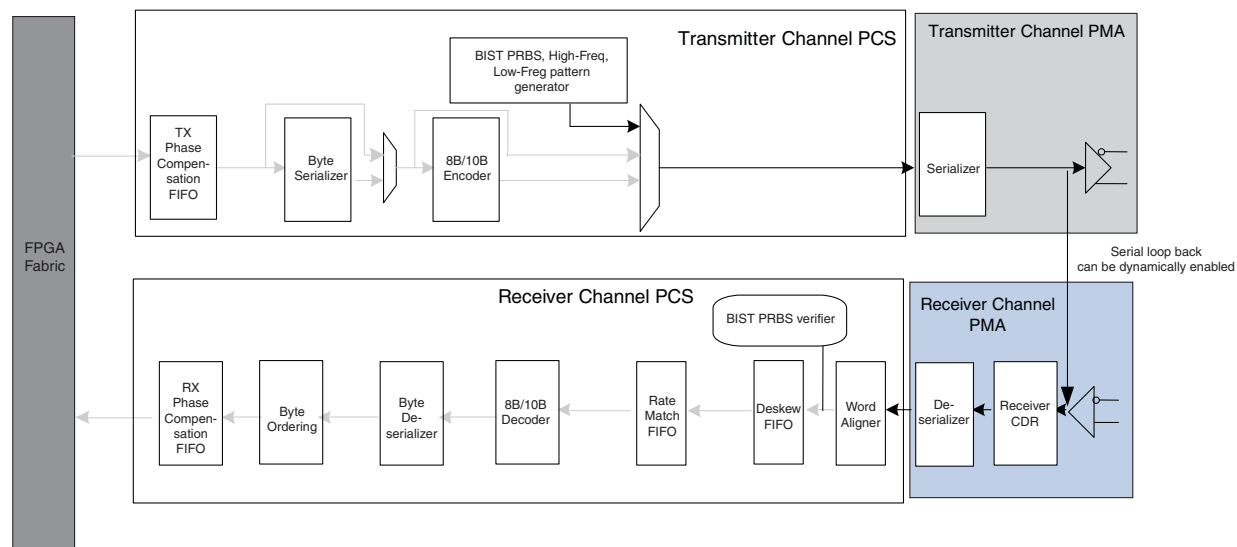
 When moving into or out of serial loopback, you must assert `rx_digitalreset` for a minimum of two parallel clock cycles.

Figure 1-155. 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 1-156. The incremental data pattern is not available to the FPGA logic for verification.

Figure 1-156. Enabled PCS Functional Blocks in Parallel Loopback

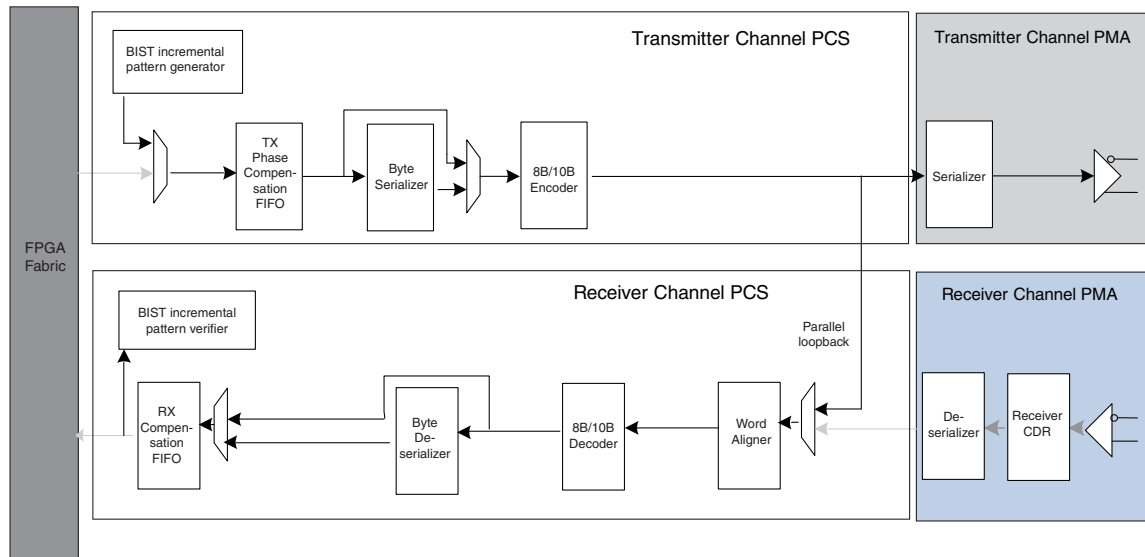


Table 1-69 lists the enabled PCS functional blocks for single-width and double-width mode. The last column in Table 1-69 lists the supported channel width setting for parallel loopback.

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 3.125 Gbps	16
Double-width mode	Enabled	Disabled	1 Gbps to 5 Gbps	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.

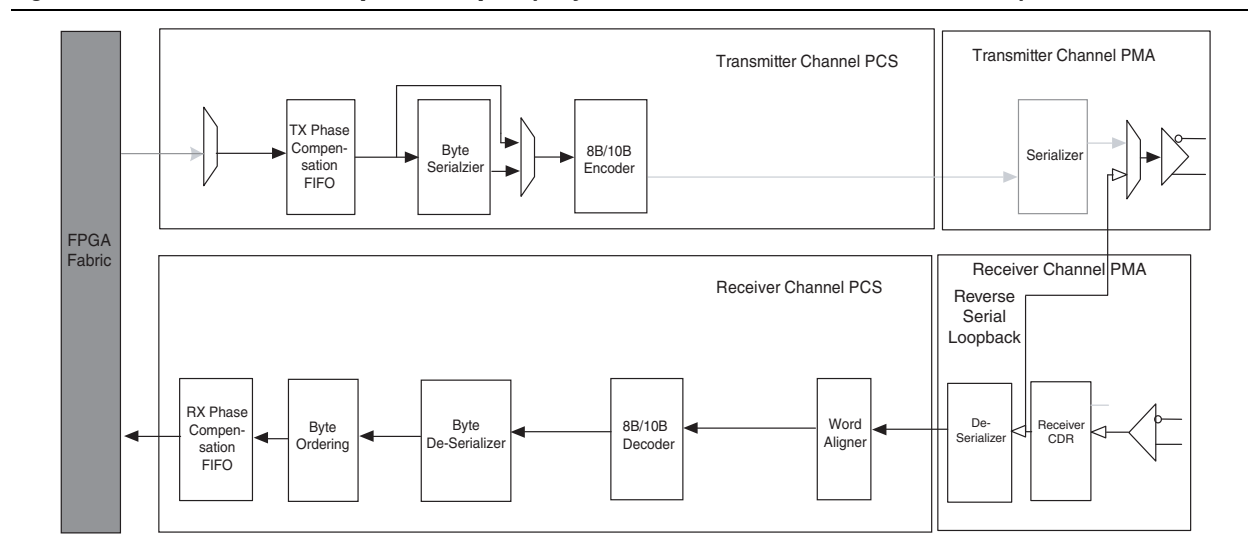
Reverse Serial Loopback

The Reverse Serial Loopback can be set by selecting the radial button under the Loopback tab in the ALTGX MegaWizard. 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. No dynamic pin control is available to select or deselect reverse serial loopback.

Figure 1-157 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.

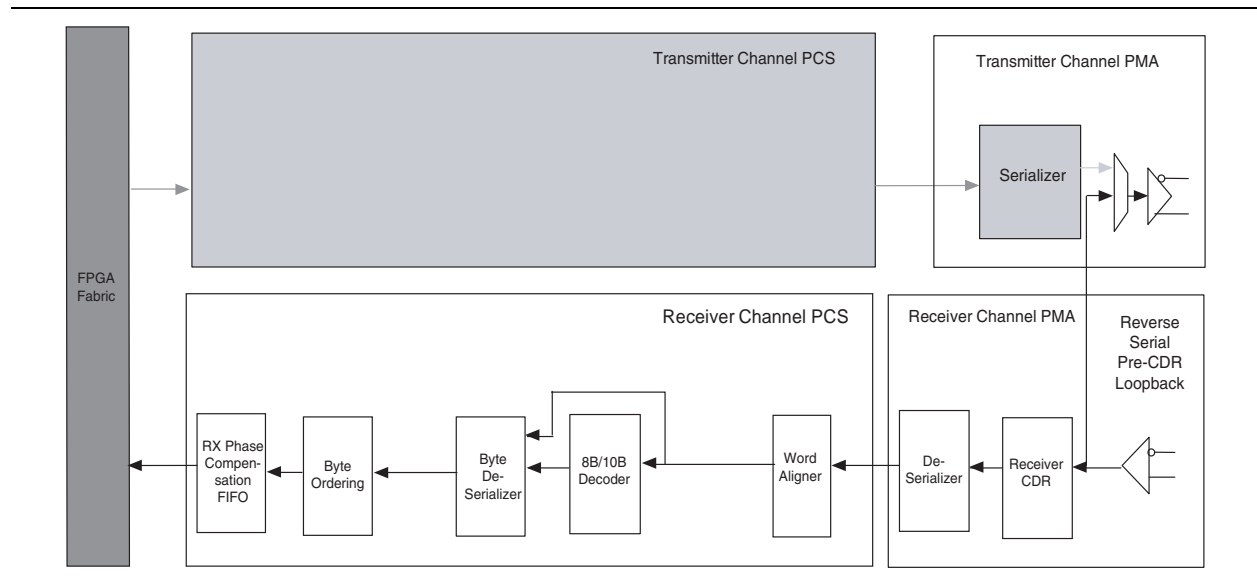
Figure 1-157. Reverse Serial Loopback Datapath (Grayed-Out Blocks are Not Active in this Mode)



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 1-158 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 1-158. Reverse Serial Pre-CDR Loopback Datapath

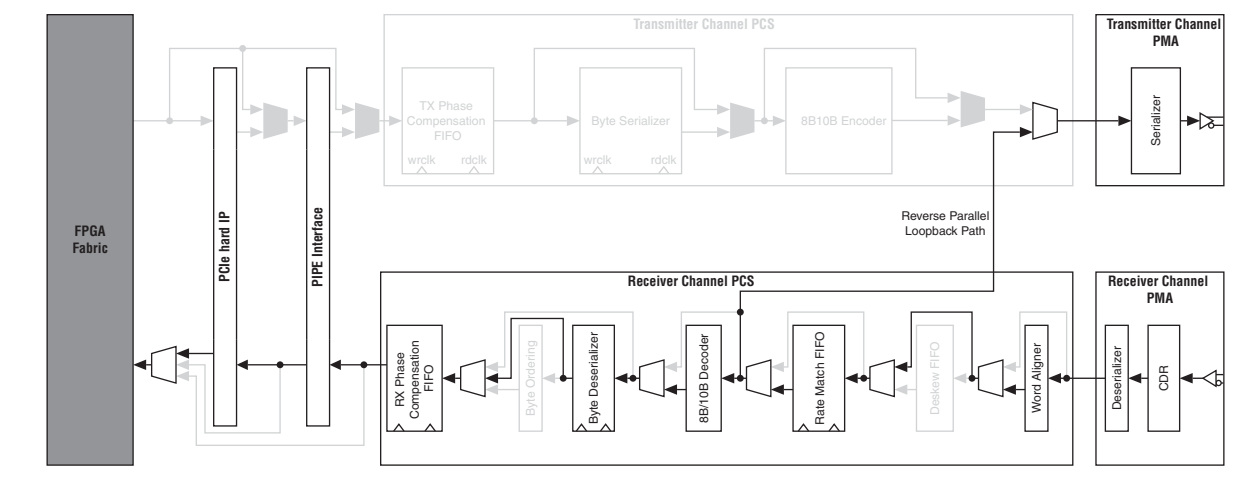


PCIe Reverse Parallel Loopback

PCIe reverse parallel loopback is only available in PCIe functional mode for Gen1 and Gen2 data rates. As shown in Figure 1-159, 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 PCIe specification 2.0. To enable this loopback mode, assert the tx_detectrxloopback port.

In Figure 1-159, the grayed areas show the inactive paths when the PCIe reverse parallel loopback mode is enabled.

Figure 1-159. PCIe 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 *DC and Switching Characteristics for Stratix IV Devices* chapter.

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 *DC and Switching Characteristics for Stratix IV Devices* chapter.

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 1-163 and Figure 1-164 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 1-160 shows the location of the ATX PLL blocks in two transceiver block device families.

Figure 1-160. Location of ATX PLL Blocks in a Four-Transceiver Block Stratix IV GX Device (Two on Each Side)

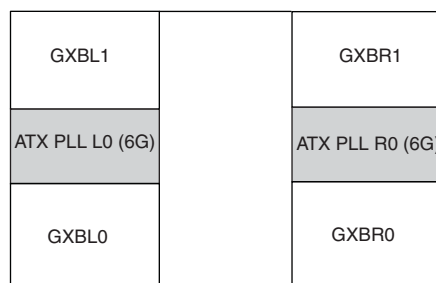


Figure 1-161 shows the location of the ATX PLL blocks in three transceiver block device families (for 230K and 530K devices and all other devices).

Figure 1-161. Location of ATX PLL Blocks with a Six Transceiver Block Stratix IV GX Device (Three on Each side)

In 230K and 530K Stratix IV GX Devices

GXBL2		GXBR2
GXBL1		GXBR1
ATX PLL L0 (6G)		ATX PLL R0 (6G)
GXBL0		GXBR0

In Stratix IV GX Devices Other than 230K and 530K

GXBL2		GXBR2
ATX PLL L1 (6G)		ATX PLL R1 (6G)
GXBL1		GXBR1
ATX PLL L0 (6G)		ATX PLL R0 (6G)
GXBL0		GXBR0

Figure 1-162 shows the location of the ATX PLL blocks in four transceiver block device families.

Figure 1-162. Location of ATX PLL Blocks in an Eight-Transceiver Block Stratix IV GX Device (Four on Each Side)

GXBL3		GXBR3
GXBL2		GXBR2
ATX PLL L1 (6G)		ATX PLL R1 (6G)
GXBL1		GXBR1
ATX PLL L0 (6G)		ATX PLL R0 (6G)
GXBL0		GXBR0

Figure 1-163 and Figure 1-164 show the locations of the 6G and 10G ATX PLLs in each Stratix IV GT device.

Figure 1-163. Location of Transceiver Channel and PLL in Stratix IV GT Devices (EP4S40G2F40, EP4S40G5H40, EP4S100G2F40 and EP4S100G5H40)

Transceiver Block GXBL2		Transceiver Block GXBR2
ATX PLL L1 (10G)		ATX PLL R1 (10G)
Transceiver Block GXBL1		Transceiver Block GXBR1
ATX PLL L0 (6G)		ATX PLL R0 (6G)
Transceiver Block GXBL0		Transceiver Block GXBR0

Figure 1-164. Location of Transceiver Channel and PLL in Stratix IV GT Devices (EP4S100G5F45)

Transceiver Block GXBL3		Transceiver Block GXBR3
ATX PLL L2 (10G)		ATX PLL R2 (10G)
Transceiver Block GXBL2		Transceiver Block GXBR2
ATX PLL L1 (6G)		ATX PLL R1 (6G)
Transceiver Block GXBL1		Transceiver Block GXBR1
ATX PLL L0 (6G)		ATX PLL R0 (6G)
Transceiver Block GXBL0		Transceiver Block GXBR0


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 using the REFCLK 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 `ref1clk` pins on the same side of the device.

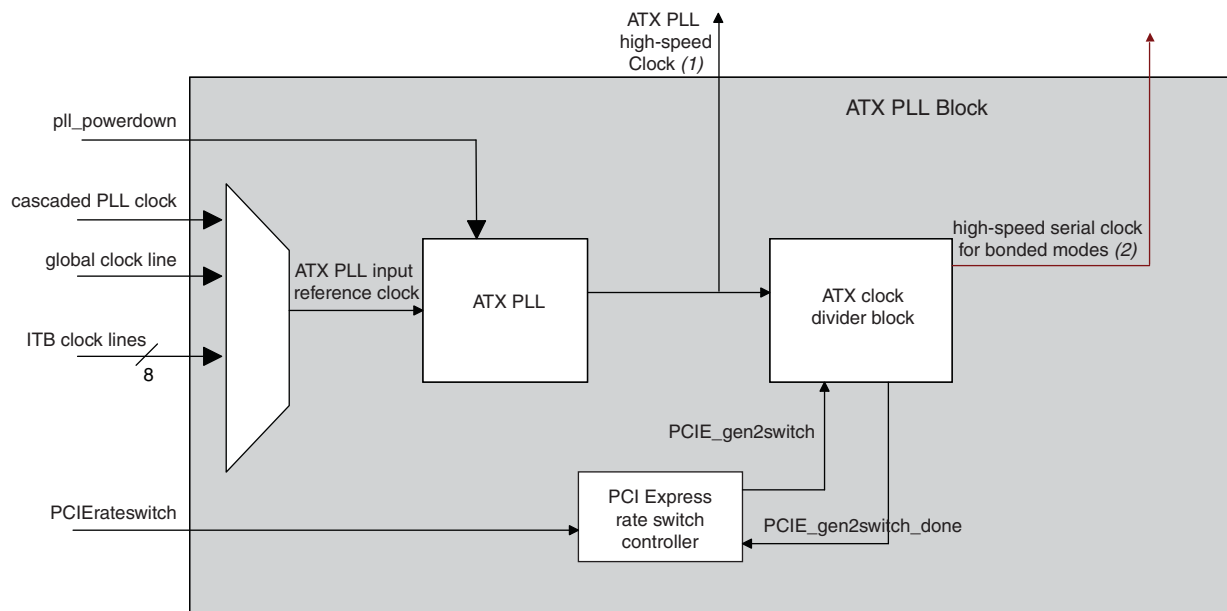
 For improved jitter performance, Altera strongly recommends using the REFCLK pins of the transceiver block located immediately below the 10G ATX PLL block to drive the input reference clock.

 For more information about the input reference clocks for ATX PLLs, refer to the *Transceiver Clocking for Stratix IV Devices* chapter.

Architecture of the ATX PLL Block

Figure 1-165 shows the ATX PLL block components (the ATX PLL, ATX clock divider, and a shared control signal generation block).

Figure 1-165. ATX PLL Block



Notes to Figure 1-165:

- (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 x4, x8, and PCIe x4 and x8 functional modes.

The functional blocks on the ATX PLL are similar to the blocks explained in “*CMU0 PLL*” on page 1-102. 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).

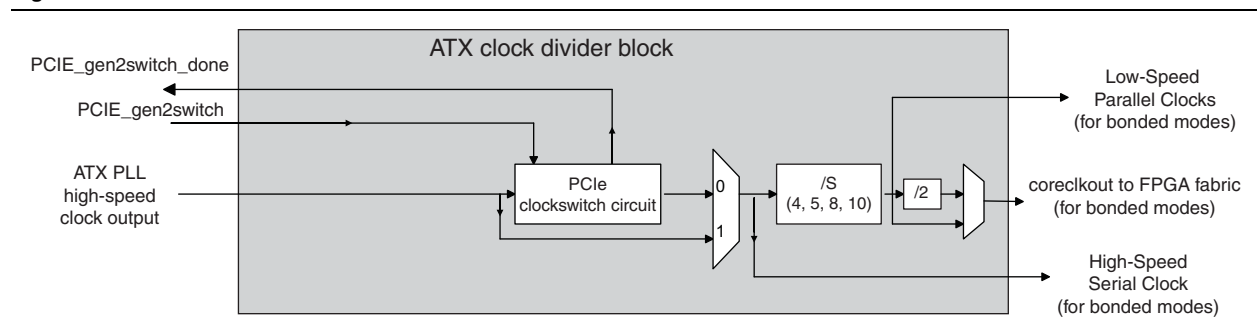
ATX Clock Divider

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 PCIe (×4 and ×8), Basic ×4 and ×8, and PMA-Direct mode with ×N configuration. For PCIe functional mode support, the ATX clock divider consists of the PCIe rateswitch circuit to enable dynamic rateswitch between PCIe Gen1 and Gen2 data rates. For more information on this circuit, refer to “CMU0 Channel” on page 1-101.

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 1-166.

 For more information, refer to the *Transceiver Clocking for Stratix IV Devices* chapter.

Figure 1-166. 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⁾	<ul style="list-style-type: none"> ■ Up to 8.5 for Stratix IV GX devices ■ Up to 11.3 for Stratix IV GT devices
Input reference clock options	Only dedicated <code>refclk</code> pins on the same side of the device ^{(2), (3)}	<ul style="list-style-type: none"> ■ 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⁾ 	<ul style="list-style-type: none"> ■ 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, <code>refclk0</code> and <code>refclk1</code> clock input, dedicated <code>refclks</code> 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 about 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 1-167 shows the location and number of calibration blocks available for different Stratix IV GX and GT devices. In Figure 1-167 through Figure 1-172, the calibration block R0 and L0 refer to the calibration blocks on the right and left side of the devices, respectively.

Figure 1-167. Calibration Block Locations in Stratix IV GX and GT Device with Two Transceiver Blocks (on Each Side)

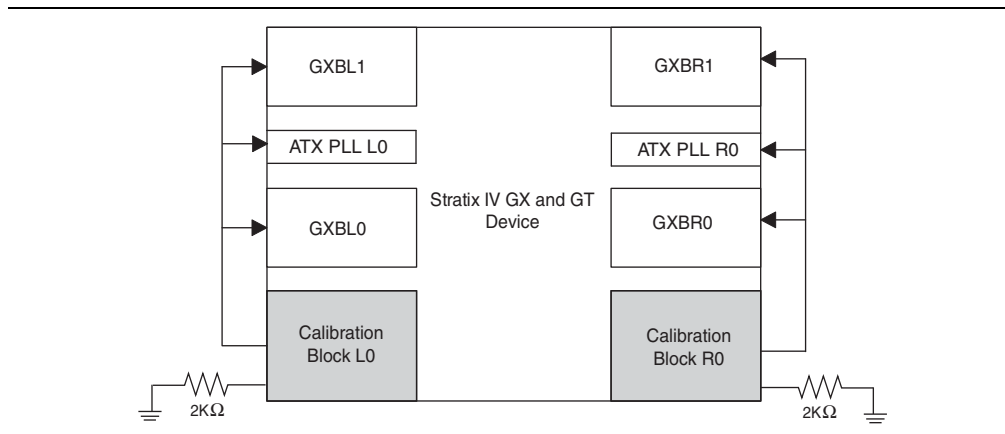


Figure 1-168 shows Stratix IV GX 230K and 530K devices that have three transceiver blocks each on the left and right side and one ATX PLL block on each side.

Figure 1-168. Calibration Block Locations in Stratix IV GX 230K and 530K Devices with Three Transceiver Blocks (on Each Side)

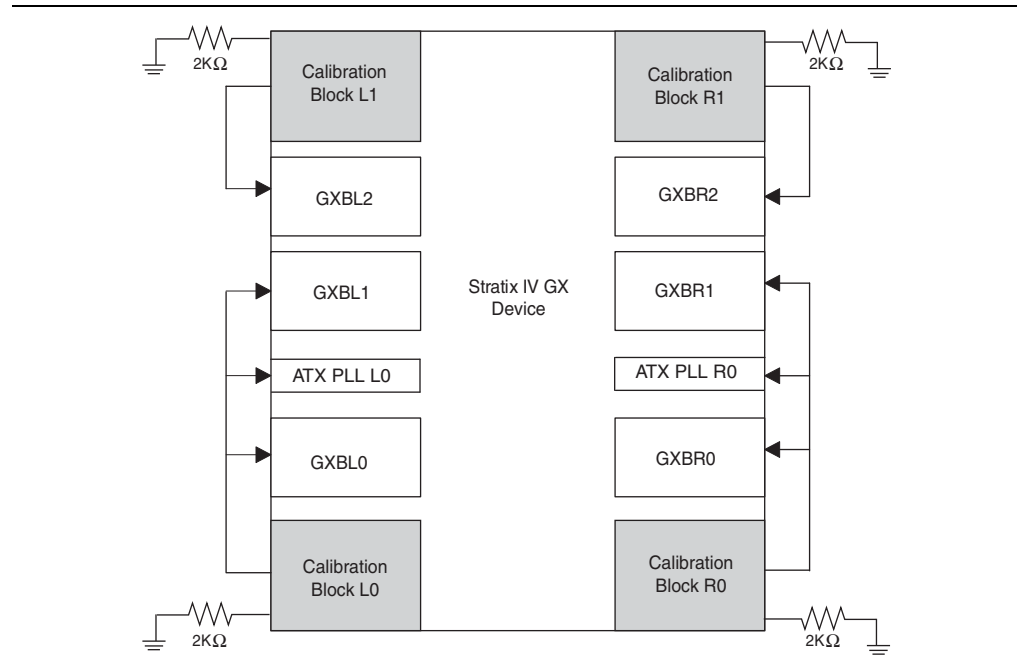


Figure 1-169 shows Stratix IV GX devices other than 230K and 530K that have three transceiver blocks each on the left and right side and two ATX PLL blocks on each side.

Figure 1-169. Calibration Block Locations in Stratix IV GX Devices Other than 230K and 530K with Three Transceiver Blocks (on Each Side)

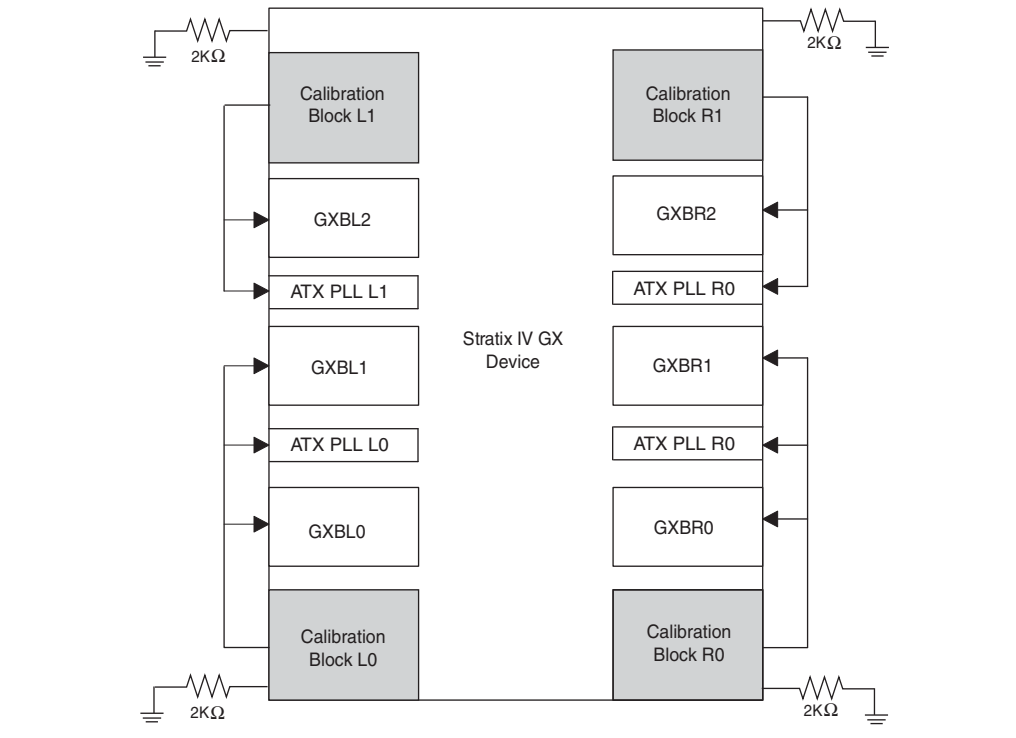


Figure 1-170 shows Stratix IV GX devices that have four transceiver blocks each on the left and right side and two ATX PLL blocks on each side.

Figure 1-170. Calibration Block Locations in Stratix IV GX Devices with Four Transceiver Blocks (on Each Side)

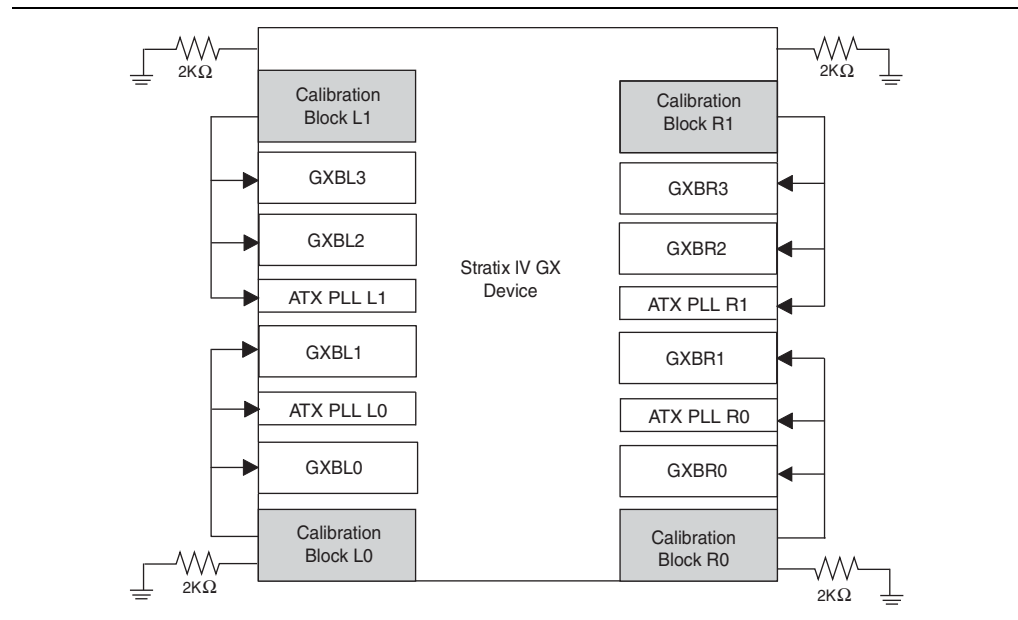
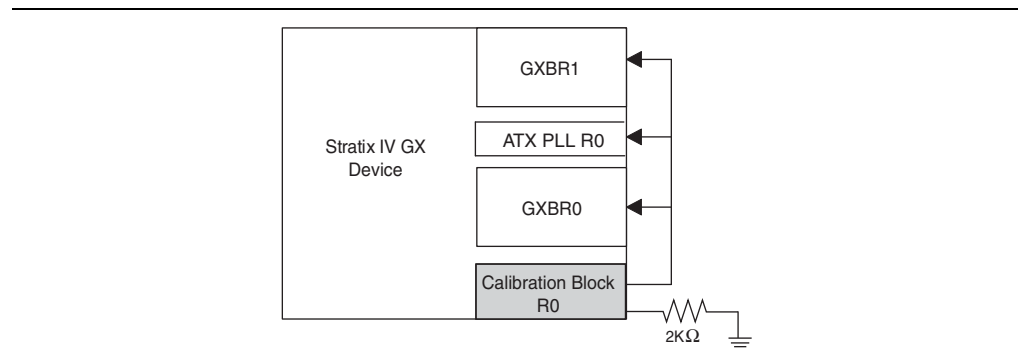


Figure 1-171 shows Stratix IV GX devices that have two transceiver blocks only on the right side of the device.

Figure 1-171. 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 k Ω (tolerance max \pm 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.

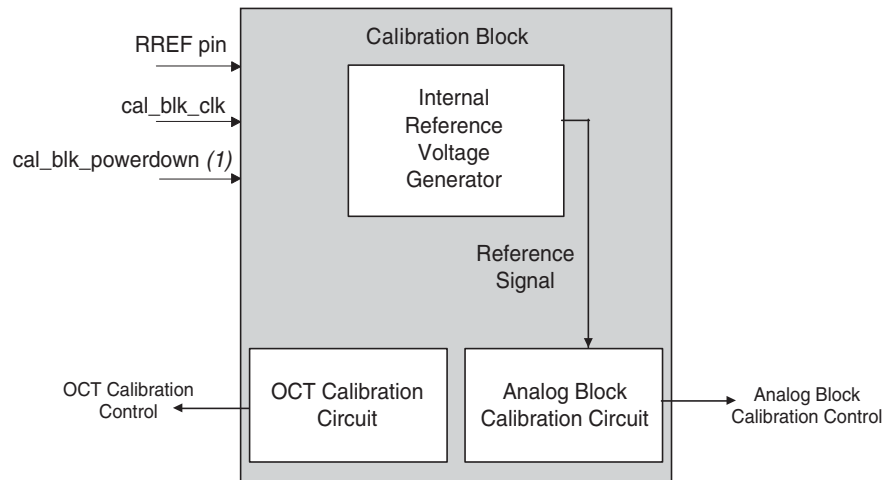
Input Signals to the Calibration Block

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 after 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. 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 1-172 shows the required inputs to the calibration block.

Figure 1-172. Input Signals to the Calibration Blocks



Note to Figure 1-172:

- (1) The transceiver on-chip termination calibration process takes approximately 33,000 cal_blk_clk cycles from the de-assertion of the cal_blk_powerdown signal.

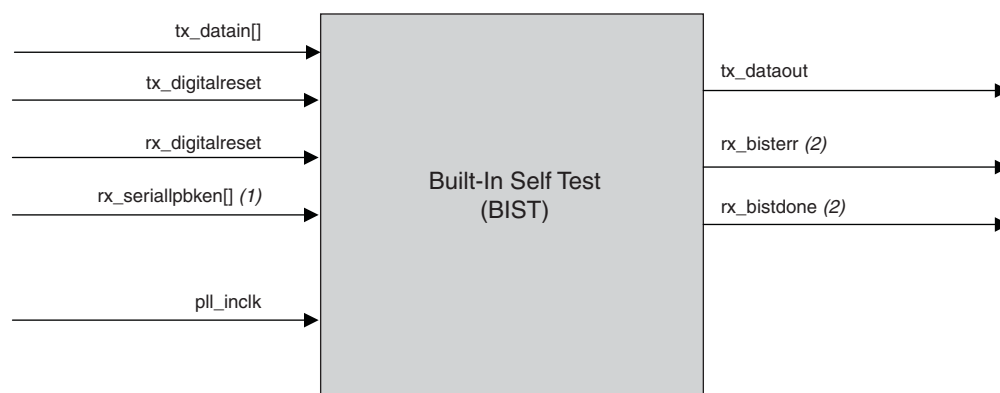
Built-In Self Test Modes

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 1-173 shows the enabled input and output ports when you select BIST mode (except incremental patterns).

Figure 1-173. Input and Output Ports for BIST Modes



Notes to Figure 1-173:

- (1) rx_serilalpbken 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:

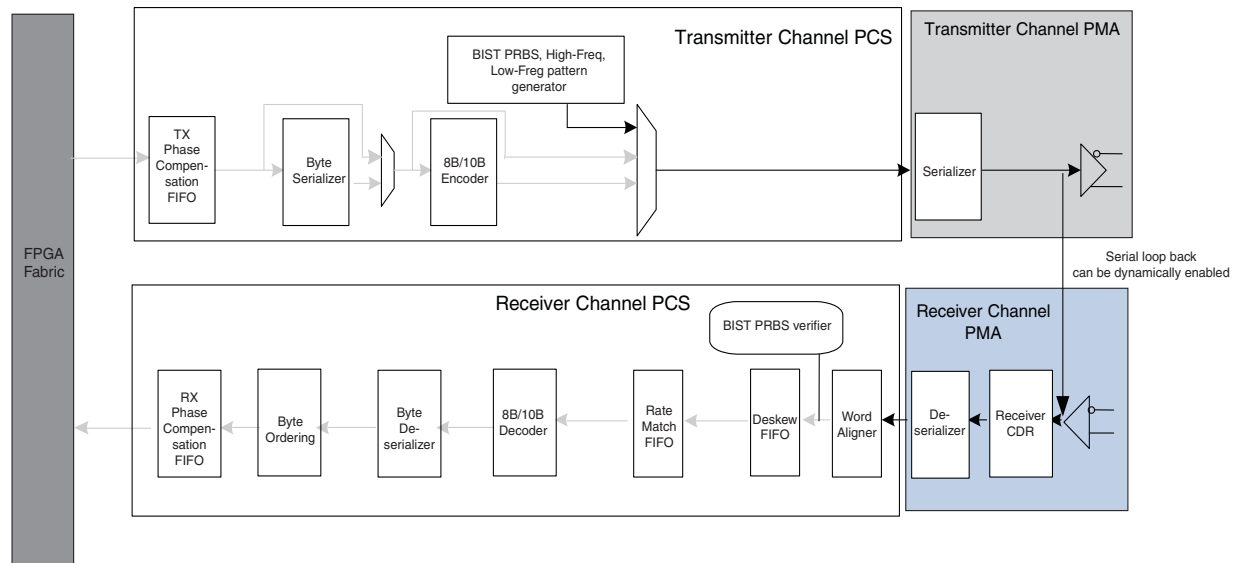
- BIST incremental data generator and verifier—This is only available in parallel loopback mode. For more information, refer to [“Serial Loopback” on page 1-190](#).
- High frequency and low frequency pattern generator—The high frequency patterns generate alternate ones and zeros and the low frequency 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_serialpbken 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_serialloopback` port that you can use to dynamically control the serial loopback. The 8B/10B encoder/decoder blocks are bypassed in Basic PRBS mode.

Figure 1-174 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 1-174. BIST PRBS, High Frequency, and Low Frequency Pattern Datapath



PRBS in Single-Width Mode

Table 1-71 lists the 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

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)
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 lists the 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

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'h007FFFF	5	Y	40'h00007FFF	6.375
High frequency ⁽²⁾	1010101010	Y	NA	5	Y	N/A	6.375
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 1-21](#).

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 list 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 3)

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. <ul style="list-style-type: none"> Bus width—depends on the channel width multiplied by the number of channels per instance. 	Channel
tx_clkout	Output	Clock signal	FPGA fabric-transceiver interface clock. <ul style="list-style-type: none"> 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. <ul style="list-style-type: none"> 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. <ul style="list-style-type: none"> A high level—the transmitter phase compensation FIFO is either full or empty. 	Channel

Table 1-73. Stratix IV GX and GT ALTX Megafunction Ports: Transmitter Ports (Part 2 of 3)

Port Name	Input/Output	Clock Domain	Description	Scope
8B/10B Encoder				
tx_ctrlenable	Input	Synchronous to tx_clkout/coreclkout clock signal.	<p>8B/10B encoder /Kx.y/ or /Dx.y/ control.</p> <ul style="list-style-type: none"> 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: <ul style="list-style-type: none"> 8—tx_ctrlenable = 1 16—tx_ctrlenable = 2 32—tx_ctrlenable = 4 	Channel
tx_forcedisp	Input	Asynchronous signal. Minimum pulse width is one parallel clock cycles.	<p>8B/10B encoder force disparity control.</p> <ul style="list-style-type: none"> 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: <ul style="list-style-type: none"> 8—tx_forcedisp = 1 16—tx_forcedisp = 2 32—tx_forcedisp = 4 	Channel
tx_dispval	Input	Asynchronous signal. Minimum pulse width is one parallel clock cycles.	<p>8B/10B encoder force disparity value.</p> <ul style="list-style-type: none"> 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: <ul style="list-style-type: none"> 8—tx_dispval = 1 16—tx_dispval = 2 32—tx_dispval = 4 	Channel

Table 1–73. Stratix IV GX and GT ALTGX Megafunction Ports: Transmitter Ports (Part 3 of 3)

Port Name	Input/Output	Clock Domain	Description	Scope
tx_invpolarity	Input	Asynchronous signal. Minimum pulse width is two parallel clock cycles.	<p>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.</p> <ul style="list-style-type: none"> ■ 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
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 PCIe 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 7)

Port Name	Input/Output	Clock Domain	Description	Scope
rx_syncstatus	Output	Synchronous to rx_clkout or coreclkout. rx_clkout for non-bonded modes. coreclkout for bonded modes	<p>Word alignment synchronization status indicator.</p> <ul style="list-style-type: none"> ■ 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. <p>For more information, refer to “Word Aligner in Single-Width Mode” on page 1-60 and “Word Aligner in Double-Width Mode” on page 1-66.</p> <ul style="list-style-type: none"> ■ 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.	<p>Bit-slip control for the word aligner configured in bit-slip mode.</p> <p>At every rising edge, word aligner slips one bit into the received data stream, effectively shifting the word boundary by one bit.</p>	Channel
rx_ala2size	Input	Asynchronous Signal. Minimum pulse width is two parallel clock cycles.	<p>Available only in SONET OC-12 and OC-48 modes. Select between these options:</p> <ul style="list-style-type: none"> ■ 0 = 16-bit A1A2 ■ 1 = 32-bit A1A1A2A2 	Channel
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.	<p>Run-length violation indicator.</p> <p>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.</p>	Channel

Table 1-74. Stratix IV GX and GT ALTGX Megafunction Ports: Receiver Ports (Part 2 of 7)

Port Name	Input/Output	Clock Domain	Description	Scope
rx_invpolarity	Input	Asynchronous Signal. Minimum pulse width is two parallel clock cycles.	<p>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.</p> <ul style="list-style-type: none"> 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.	<p>Receiver bit reversal control. This is a useful feature where the link transmission order is MSB to LSB.</p> <ul style="list-style-type: none"> 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
rx_revbyteorderwa	Input	Asynchronous Signal. Minimum pulse width is two parallel clock cycles.	<p>Receiver byte reversal control. This is a useful feature in situations where the MSByte and LSByte of the transmitted data are erroneously swapped.</p> <ul style="list-style-type: none"> 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

Table 1-74. Stratix IV GX and GT ALTGX Megafunction Ports: Receiver Ports (Part 3 of 7)

Port Name	Input/Output	Clock Domain	Description	Scope
Deskew FIFO				
rx_channelaligned	Output	Synchronous to coreclkout clock signal	<p>XAUI deskew FIFO channel aligned indicator.</p> <ul style="list-style-type: none"> 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_rmfifoinserted	Output	Synchronous to rx_clkout or coreclkout. rx_clkout for non-bonded modes. coreclkout for bonded modes.	<p>Rate match FIFO insertion status indicator.</p> <ul style="list-style-type: none"> 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_rmfiodeleted	Output	Synchronous to rx_clkout or coreclkout. rx_clkout for non-bonded modes. coreclkout for bonded modes	<p>Rate match FIFO deletion status indicator.</p> <ul style="list-style-type: none"> A high level—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
rx_rmfiopull	Output	Synchronous to rx_clkout or coreclkout. rx_clkout for non-bonded modes. coreclkout for bonded modes	<p>Rate match FIFO full status indicator.</p> <ul style="list-style-type: none"> 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_rmfiempty	Output	Synchronous to rx_clkout or coreclkout. rx_clkout for non-bonded modes. coreclkout for bonded modes	<p>Rate match FIFO empty status indicator.</p> <ul style="list-style-type: none"> 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

Table 1-74. Stratix IV GX and GT ALTGX Megafunction Ports: Receiver Ports (Part 4 of 7)

Port Name	Input/Output	Clock Domain	Description	Scope
8B/10B Decoder				
rx_ctrldetect	Output	Synchronous to rx_clkout, tx_clkout, and coreclkout clock signals	<p>Receiver control code indicator.</p> <ul style="list-style-type: none"> 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: <p>Channel Width: 8—rx_ctrldetect = 1 16—rx_ctrldetect = 2 32—rx_ctrldetect = 4</p>	Channel
rx_errdetect	Output	Synchronous to rx_clkout or coreclkout. rx_clkout for non-bonded modes. coreclkout for bonded modes	<p>8B/10B code group violation or disparity error indicator.</p> <ul style="list-style-type: none"> 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: <ul style="list-style-type: none"> [rx_errdetect: rx_disperr] 2'b00—no error 2'b10—code group violation 2'b11—disparity error or both <ul style="list-style-type: none"> Channel Width: 8—rx_errdetect = 1 16—rx_errdetect = 2 32—rx_errdetect = 4 	Channel
rx_disperr	Output	Synchronous to rx_clkout or coreclkout. rx_clkout for non-bonded modes. coreclkout for bonded modes	<p>8B/10B disparity error indicator port.</p> <ul style="list-style-type: none"> Available in configurations with 8B/10B decoder. A high level—a disparity error was detected on the associated received code group. <ul style="list-style-type: none"> Channel Width: 8—rx_disperr = 1 16—rx_disperr = 2 32—rx_disperr = 4 	Channel

Table 1-74. Stratix IV GX and GT ALTGX Megafunction Ports: Receiver Ports (Part 5 of 7)

Port Name	Input/Output	Clock Domain	Description	Scope
rx_runningdisp	Output	Synchronous to rx_clkout or coreclkout. rx_clkout for non-bonded modes. coreclkout for bonded modes	<p>8B/10B running disparity indicator.</p> <ul style="list-style-type: none"> 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: <ul style="list-style-type: none"> 8—rx_runningdisp = 1 16—rx_runningdisp = 2 32—rx_runningdisp = 4 	Channel
Byte Ordering Block				
rx_enabyteord	Input	Asynchronous signal	<p>Enable byte ordering control.</p> <ul style="list-style-type: none"> 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 rx_clkout or coreclkout. rx_clkout for non-bonded modes. coreclkout for bonded modes	<p>Byte ordering status indicator.</p> <ul style="list-style-type: none"> 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 rx_clkout or coreclkout. rx_clkout for non-bonded modes. coreclkout for bonded modes	<p>Parallel data output from the receiver to the FPGA fabric.</p> <ul style="list-style-type: none"> The bus width depends on the channel width multiplied by the number of channels per instance. 	Channel
rx_clkout	Output	Clock signal	<p>Recovered clock from the receiver channel.</p> <ul style="list-style-type: none"> Available only when the rate match FIFO is not used in the receiver datapath. 	Channel

Table 1-74. Stratix IV GX and GT ALTGX Megafunction Ports: Receiver Ports (Part 6 of 7)

Port Name	Input/Output	Clock Domain	Description	Scope
rx_coreclk	Input	Clock signal	Optional read clock port for the receiver phase compensation FIFO. <ul style="list-style-type: none"> ■ 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
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. <ul style="list-style-type: none"> ■ 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_crucclk	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. <ul style="list-style-type: none"> ■ 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. <ul style="list-style-type: none"> ■ 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. <ul style="list-style-type: none"> ■ 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

Table 1-74. Stratix IV GX and GT ALTGX Megafunction Ports: Receiver Ports (Part 7 of 7)

Port Name	Input/Output	Clock Domain	Description	Scope
rx_locktohrefclk	Input	Asynchronous signal	Receiver CDR lock-to-reference mode control signal. The rx_locktohrefclk 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
rx_signaldetect	Output	Asynchronous signal	Signal threshold detect indicator. <ul style="list-style-type: none"> ■ Available in Basic functional mode when the 8B/10B Encoder/Decoder is selected. ■ Available in PCIe mode. ■ A high level—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 PCIe mode, the rx_signaldetect signal is inverted and driven on the pipeelecidle port. 	Channel
rx_serialpbken	Input	Asynchronous signal	Serial loopback control port. <ul style="list-style-type: none"> ■ 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. <ul style="list-style-type: none"> ■ 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 <i>DC and Switching Characteristics</i> chapter.	<p>CMU PLL power down.</p> <ul style="list-style-type: none"> ■ Asserted high—the CMU PLL is powered down. ■ De-asserted low—the CMU PLL is active and locks to the input reference clock. <p>Note: Asserting the <code>pll_powerdown</code> signal does not power down the REFCLK buffers.</p>	Transceiver block
coreclkout	Output	Clock signal	<p>FPGA fabric-transceiver interface clock.</p> <ul style="list-style-type: none"> ■ Generated by the <code>CMU0</code> clock divider in the transceiver block in $\times 4$ bonded channel configurations. ■ Generated by the <code>CMU0</code> clock divider in the master transceiver block in $\times 8$ 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 <code>tx_datain</code> 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 <code>rx_dataout</code> 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	<p>Dynamic reconfiguration clock.</p> <ul style="list-style-type: none"> ■ Also used for offset cancellation in all modes except PCIe 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 PCIe interface ports.

Table 1-77. Stratix IV GX and GT ALTGX Megafunction Ports: PCIe Interface (Part 1 of 4)

Port Name	Input/Output	Clock Domain	Description	Scope
PCIe Interface (Available only in PCIe functional Mode)				
powerdn	Input	Asynchronous signal	<p>PCIe power state control.</p> <ul style="list-style-type: none"> ■ Functionally equivalent to the <code>powerdown[1:0]</code> signal defined in the PCIe specification revision 2.0. ■ The width of this signal is 2 bits and is encoded as follows: <ul style="list-style-type: none"> ■ 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	<p>Force 8B/10B encoder to encode with a negative running disparity.</p> <ul style="list-style-type: none"> ■ Functionally equivalent to the <code>txcompliance</code> signal defined in PCIe specification revision 2.0. ■ Must be asserted high only when transmitting the first byte of the PCIe compliance pattern to force the 8B/10B encode with a negative running disparity as required by the PCIe protocol. 	Channel
tx_forceelecidle	Input	Asynchronous signal	<p>Force transmitter buffer to PCIe electrical idle signal levels.</p> <ul style="list-style-type: none"> ■ Functionally equivalent to the <code>txelecidle</code> signal defined in the PCIe specification revision 2.0. ■ Available in the Basic mode. 	Channel
rateswitch	Input	Asynchronous signal	<p>PCIe rateswitch control.</p> <ul style="list-style-type: none"> ■ 1'b0—Gen1 (2.5 Gbps) ■ 1'b1—Gen2 (5 Gbps) 	

Table 1-77. Stratix IV GX and GT ALTGX Megafunction Ports: PCIe Interface (Part 2 of 4)

Port Name	Input/Output	Clock Domain	Description	Scope
tx_pipemargin	Input	Asynchronous signal	<p>Transmitter differential output voltage level control.</p> <ul style="list-style-type: none"> ■ Functionally equivalent to the txmargin signal defined in the PCIe specification revision 2.0. ■ Available only in PCIe Gen2 configuration. ■ The width of this signal is 3 bits and is decoded as follows: <ul style="list-style-type: none"> ■ 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	<p>Transmitter buffer de-emphasis level control.</p> <ul style="list-style-type: none"> ■ Functionally equivalent to the txdeemph signal defined in the PCIe specification revision 2.0. ■ Available only in PCIe Gen2 configuration. ■ 1'b0: -6 dB de-emphasis ■ 1'b1: -3.5 dB de-emphasis 	
pipe8b10binvpolarity	Input	Asynchronous signal	<p>PCIe polarity inversion control.</p> <ul style="list-style-type: none"> ■ Functionally equivalent to the rxpolarity signal defined in the PCIe specification revision 2.0. ■ Available only in PCIe 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: PCIe Interface (Part 3 of 4)

Port Name	Input/Output	Clock Domain	Description	Scope
tx_detectrxloopback	Input	Asynchronous signal	<p>Receiver detect or PCIe loopback control.</p> <ul style="list-style-type: none"> Functionally equivalent to the txdetectrx/loopback signal defined in the PCIe 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 “PCIe Reverse Parallel Loopback” on page 1-194. 	Channel
pipestatus	Output	N/A	<p>PCIe receiver status port.</p> <ul style="list-style-type: none"> Functionally equivalent to the rxstatus[2:0] signal defined in the PCIe specification revision 2.0. Synchronized with tx_clock. The width of this signal is 3 bits per channel. The encoding of receiver status on the pipestatus port is as follows: <ul style="list-style-type: none"> 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	<p>PHY function completion indicator.</p> <ul style="list-style-type: none"> Functionally equivalent to the phystatus signal defined in the PCIe 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). Synchronized with tx_clkout. 	Channel

Table 1–77. Stratix IV GX and GT ALTGX Megafunction Ports: PCIe 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. <ul style="list-style-type: none"> Functionally equivalent to the rxvalid signal defined in the PCIe specification revision 2.0. 	Channel
pipeelecidle	Output	Asynchronous signal	Electrical idle detected or inferred at the receiver indicator. <ul style="list-style-type: none"> Functionally equivalent to the rxelecidle signal defined in the PCIe 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 1–138. 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 <i>DC and Switching Characteristics</i> chapter.	Transceiver block power down. <ul style="list-style-type: none"> 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. <ul style="list-style-type: none"> 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. <ul style="list-style-type: none"> When asserted high—<i>analog circuitry within the receiver PMA gets reset. Refer to Reset Control and Power Down.</i> 	Channel
tx_digitalreset	Input	Asynchronous signal. Minimum pulse width is two parallel clock cycles.	Transmitter PCS reset. <ul style="list-style-type: none"> 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 1-181
8B/10B Decoder	page 1-89
8B/10B Encoder	page 1-23
AEQ	page 1-50
Auxiliary Transmit (ATX) PLL Block	page 1-195
Basic (PMA Direct) Functional Mode	page 1-187
Basic Functional Mode	page 1-111
Built-In Self Test Modes	page 1-207
Byte Ordering Block	page 1-95
Byte Serializer	page 1-94
Calibration Blocks	page 1-201
Clock and Data Recovery Unit (CDR)	page 1-53
CMU Channel Architecture	page 1-100
CMU0 PLL	page 1-102
CMU1 PLL	page 1-102

Table 1-80. Reference Information (Part 2 of 3)

Terms Used in this Chapter	Useful Reference Points
CPRI and OBSAI	page 1-124
Deserializer	page 1-58
Deskew FIFO	page 1-75
Deterministic Latency Mode	page 1-122
EyeQ	page 1-51
GIGE Mode	page 1-164
Lock-to-Data (LTD)	page 1-54
Lock-to-Reference (LTR)	page 1-53
Low Latency PCS Datapath	page 1-112
Offset Cancellation in the Receiver Buffer and Receiver CDR	page 1-56
Parallel loopback	page 1-191
PCIe Clock Switch Circuitry	page 1-55
PCIe Mode	page 1-127
PCIe Reverse Parallel Loopback	page 1-194
Programmable Common Mode Voltage	page 1-40
Programmable Equalization and DC Gain	page 1-49
Programmable Pre-Emphasis	page 1-36
Programmable Differential On-Chip Termination	page 1-42
Rate Match (Clock Rate Compensation) FIFO	page 1-77
Receiver Bit Reversal	page 1-73
Receiver Input Buffer	page 1-40
Receiver Phase Compensation FIFO	page 1-98
Receiver Polarity Inversion	page 1-71
Reverse Serial Loopback	page 1-193
Reverse serial Pre-CDR Loopback	page 1-194
SATA and SAS options	page 1-121
SDI Mode	page 1-178
Serial Loopback	page 1-190
Serial RapidIO Mode	page 1-182
Signal Threshold Detection Circuitry	page 1-49
SONET/SDH Mode	page 1-172
Transceiver Block Architecture	page 1-16
Transceiver Channel Locations	page 1-4
Transceiver Port Lists	page 1-210
Transmitter Bit Reversal	page 1-31
Transmitter Local Clock Divider Block	page 1-39
Transmitter Output Buffer	page 1-34
Transmitter Polarity Inversion	page 1-29
TX Phase Compensation FIFO	page 1-19

Table 1-80. Reference Information (Part 3 of 3)

Terms Used in this Chapter	Useful Reference Points
Word Aligner	page 1-59
XAUI Mode	page 1-153

Document Revision History

Table 1-81 lists the revision history for this chapter.

Table 1-81. Document Revision History (Part 1 of 2)

Date	Version	Changes
September 2015	4.7	<ul style="list-style-type: none"> Updated Figure 1-2.
January 2014	4.6	<ul style="list-style-type: none"> Updated Figure 1-22, Figure 1-36, and Figure 1-105. Updated introduction to Table 1-57. Updated the “GIGE Mode”, “Rate Match FIFO”, and “Rate Match FIFO in Basic Double-Width Mode” sections.
September 2012	4.5	<ul style="list-style-type: none"> Updated Figure 1-1, Figure 1-4, Figure 1-14, Figure 1-87, and Figure 1-89 to close FB #65098. Updated Figure 1-45 to match Stratix V CDR Unit (to close FB #65098). Updated Note 1 of Figure 1-101 to close FB #31792. Updated Table 1-73 to close FB #53976. Updated the “Reverse Serial Loopback” section to close FB #44323.
December 2011	4.4	<ul style="list-style-type: none"> Updated Figure 1-9, Figure 1-10, Figure 1-112, and Figure 1-172. Updated Table 1-5 and Table 1-17. Updated the “Compliance Pattern Transmission Support” and “PCIe Cold Reset Requirements” sections.
June 2011	4.3	<ul style="list-style-type: none"> Added military speed grade to Table 1-68.
February 2011	4.2	<ul style="list-style-type: none"> Applied new template. Updated the “Overview”, “Transceiver Block Architecture”, “DC-Coupled Links”, “Link Coupling for Stratix IV GX and GT Devices”, “Configuring CMU Channels for Clock Generation”, “Configuring CMU Channels as Transceiver Channels”, “Offset Cancellation in the Receiver Buffer and Receiver CDR”, “Modes of Operation of the AEQ”, “Word-Alignment-Based Byte Ordering”, “SATA and SAS Options”, “GIGE Mode”, “Loopback Modes”, “Reverse Serial Loopback”, “Input Signals to the Calibration Block”, and “PCI Express Electrical Gold Test with Compliance Base Board (CBB)” sections. Updated Figure 1-1, Figure 1-2, Figure 1-3, Figure 1-4, Figure 1-5, Figure 1-6, Figure 1-7, Figure 1-8, Figure 1-9, Figure 1-10, Figure 1-11, Figure 1-56, Figure 1-57, Figure 1-64, Figure 1-76, Figure 1-83, Figure 1-84, Figure 1-95, Figure 1-69, Figure 1-97, Figure 1-99, Figure 1-100, Figure 1-101, Figure 1-105, Figure 1-111, Figure 1-112, and Figure 1-157. Updated Table 1-2, Table 1-3, Table 1-4, Table 1-5, Table 1-13, Table 1-14, Table 1-16, Table 1-17, Table 1-18, Table 1-19, Table 1-21, Table 1-23, Table 1-24, Table 1-41, Table 1-43, Table 1-45, Table 1-68, Table 1-70, Table 1-74, and Table 1-77. Updated chapter title. Applied new template. Minor text edits.

Table 1–81. Document Revision History (Part 2 of 2)

Date	Version	Changes
March 2010	4.1	<ul style="list-style-type: none"> ■ Added two references to the beginning of the chapter. ■ Updated the “Configuring CMU Channels for Clock Generation” section. ■ Updated Figure 1–101. ■ Minor text edits.
November 2009	4.0	<ul style="list-style-type: none"> ■ 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	3.1	<ul style="list-style-type: none"> ■ Updated the “Introduction”, “Auxiliary Transmit (ATX) PLL Block”, “Rate Match (Clock Rate Compensation) FIFO”, “Transmitter Buffer Electrical Idle”, “PCIe 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	3.0	<ul style="list-style-type: none"> ■ Reorganized sections. ■ Added the section “Link Coupling”. ■ Updated the section “DC-Coupled Links”.
November 2008	2.0	Added Offset Cancellation in the Receiver Buffer and Receiver CDR to the Receiver Channel Datapath section
May 2008	1.0	Initial release.

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:

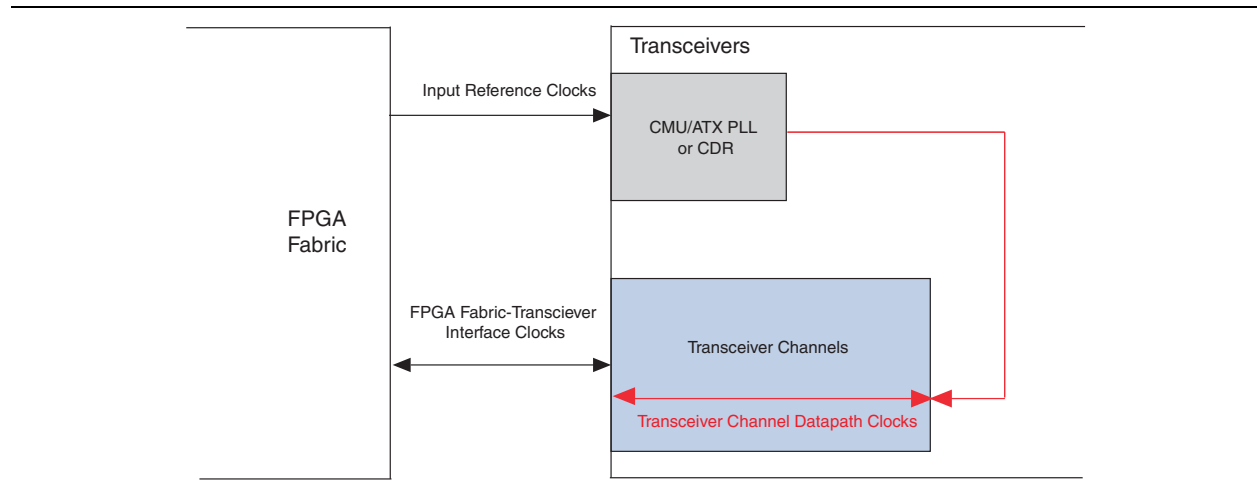
- “Input Reference Clocking” on page 2–2—describes how the reference clock is provided to the clock multiplier unit (CMU)/auxiliary transmit phase-locked loop (ATX PLL) to generate the clocks required for transceiver operation.
- “Transceiver Channel Datapath Clocking” on page 2–20—describes the clocking architecture internal to the transceiver block.
- “FPGA Fabric-Transceiver Interface Clocking” on page 2–51—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 2–71
- “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 <i>Transceiver Architecture in Stratix IV Devices</i> chapter.
CDR	Clock data recovery block. For more information, refer to the “Clock and Data Recovery Unit” section in the <i>Transceiver Architecture in Stratix IV Devices</i> chapter.
CMU	Clock multiplier unit. For more information, refer to “CMU Channel Architecture” section in the <i>Transceiver Architecture in Stratix IV Devices</i> chapter.
ITB lines	The Inter-Transceiver block (ITB) clock lines provide an input reference clock path from the <code>refclk</code> 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—the `CMU0_Channel1` and `CMU1_Channel1`


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 *Transceiver Architecture in Stratix IV Devices* 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 *Transceiver Architecture in Stratix IV Devices* 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 listed 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 ⁽⁴⁾
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 ITB clock lines ⁽²⁾	Yes	Yes	Yes	Yes	2 ⁽³⁾
3	Clock output from the left and right PLLs in the FPGA fabric with voltage controlled oscillator (VCO) bypass mode ^{(5), (6)}	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 better jitter performance, Altera strongly recommends using the refclk0 and refclk1 pins of the transceiver block located immediately below the ATX PLL.
- (4) Lowest number indicates best jitter performance.
- (5) For more information, refer to “Configuration Examples” on page 2-72.
- (6) When in VCO bypass mode, you can only divide the reference clock by the N integer. For more information, refer to “Left and Right, Left, or Right PLL in VCO Bypass Mode” on page 2-17.

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 (Gbps)	Allowed Divider Values	Reference Clock Frequency (MHz)
9.9 to 11.3	M = 16, N = 1	281.25 to 322
	M = 16, N = 2	562.5 to 706.25

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

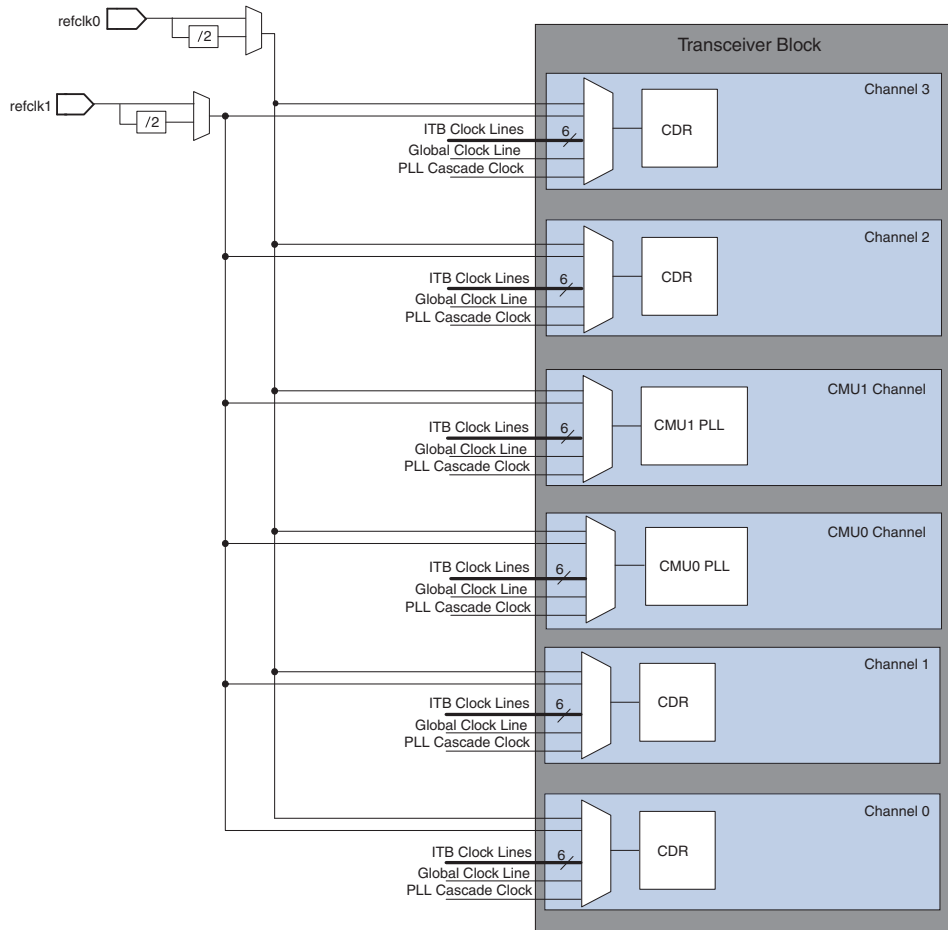


Figure 2-3 shows the input reference clock sources for CMU PLLs, ATX PLLs, and receiver CDRs in four transceiver blocks on the right side of the EP4SGX530F45 device. In this figure, the input reference clock sources for four transceiver blocks are located only on the right side of the device but the 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

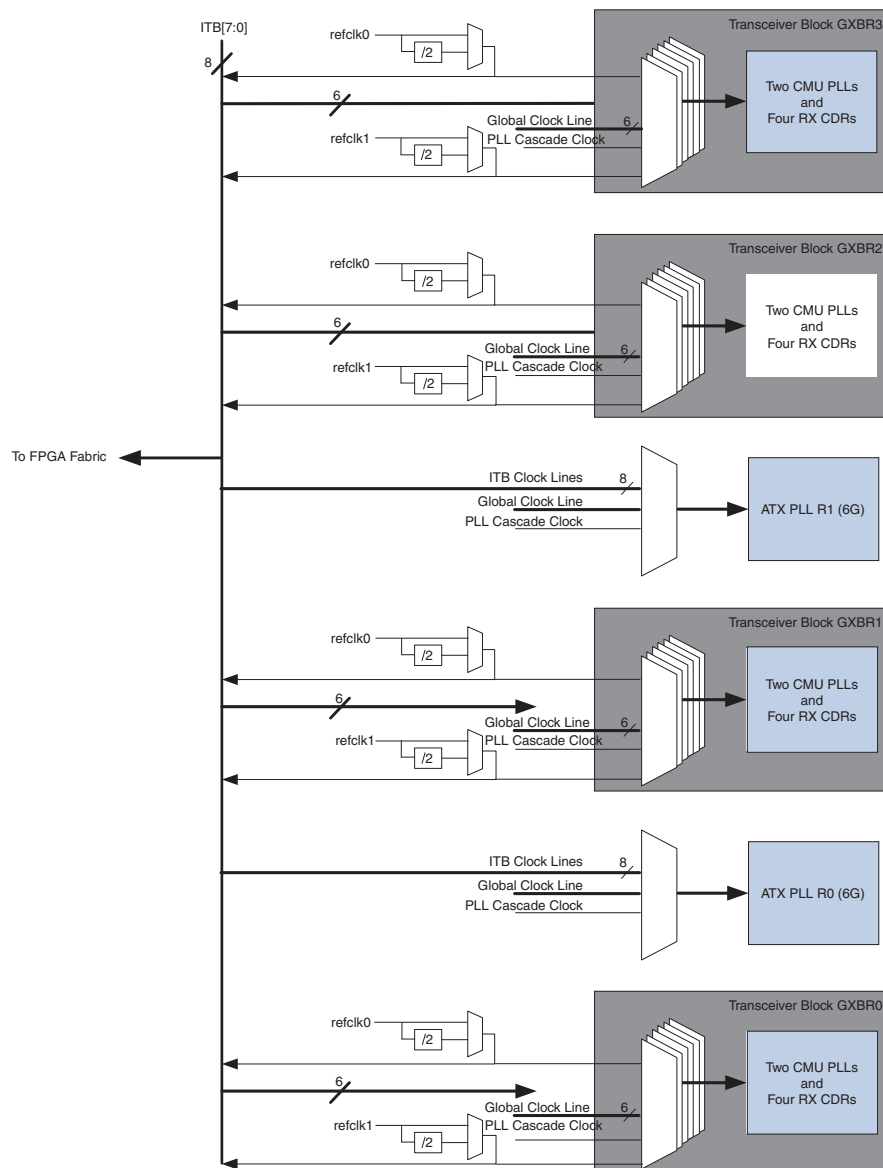
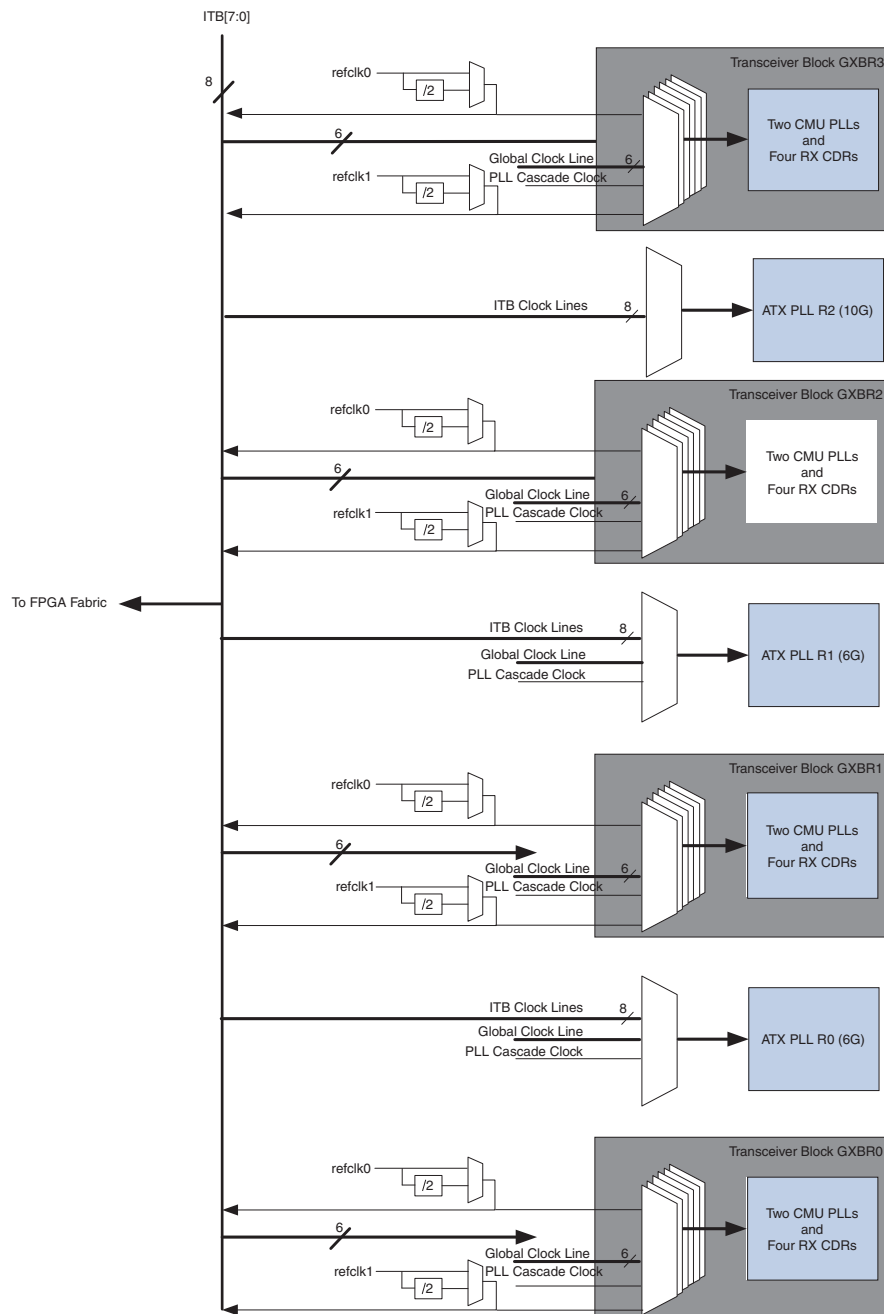


Figure 2-4 shows the input reference clock sources for CMU PLLs, ATX PLLs, and receiver CDRs in four transceiver blocks on the right side of the EP4S100G5F45 device. In this figure, the input reference clock sources for four transceiver blocks are located only on the right side of the EP4S100G5F45 device 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 for Stratix IV Devices* chapter.

Table 2-4. Electrical Specifications for the Input Reference Clock

Protocol	I/O Standard	Coupling	Termination
<ul style="list-style-type: none"> ■ GIGE ■ XAUI ■ Serial RapidIO® ■ SONET/SDH ■ SDI ■ (OIF) CEI PHY Interface ■ Basic 	<ul style="list-style-type: none"> ■ 1.2-V PCML, 1.4 PCML ■ 1.4-V PCML ■ 1.5-V PCML ■ 2.5-V PCML ■ Differential LVPECL ■ LVDS 	AC	On-chip ⁽²⁾
PCI Express® (PCIe)	<ul style="list-style-type: none"> ■ 1.2-V PCML, 1.4 PCML ■ 1.4-V PCML ■ 1.5-V PCML ■ 2.5-V PCML ■ Differential LVPECL ■ LVDS 	AC	On-chip ⁽²⁾
	<ul style="list-style-type: none"> ■ HCSL ⁽¹⁾ 	DC	Off-chip ⁽³⁾

Notes to Table 2-4:

- (1) In PCIe mode, you have the option of selecting the HCSL standard for the reference clock if compliance to the PCIe protocol is required. You can select this I/O standard option only if you configured the transceiver in PCIe functional mode. For more information, refer to Figure 2-5 on page 2-8.
- (2) Termination values supported are the same as the Receiver pin differential on-chip termination resistors specified in the *DC and Switching Characteristics for Stratix IV Devices* chapter.
- (3) For an example termination scheme, refer to Figure 2-5 on page 2-8.

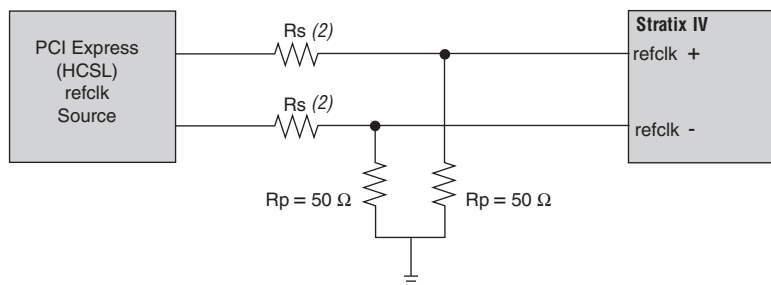


If you select the HCSL I/O standard for the PCIe reference clock, add the following assignment to your project quartus settings file (`.qsf`):

```
set_instance_assignment -name INPUT_TERMINATION OFF -to <refclk_pin_name>
```

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 PCIe specification.
- (2) Select resistor values as recommended by the PCIe 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 eight 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 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 global clock 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 a 4 Gbps Data Rate

Consider a channel configured for a 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.

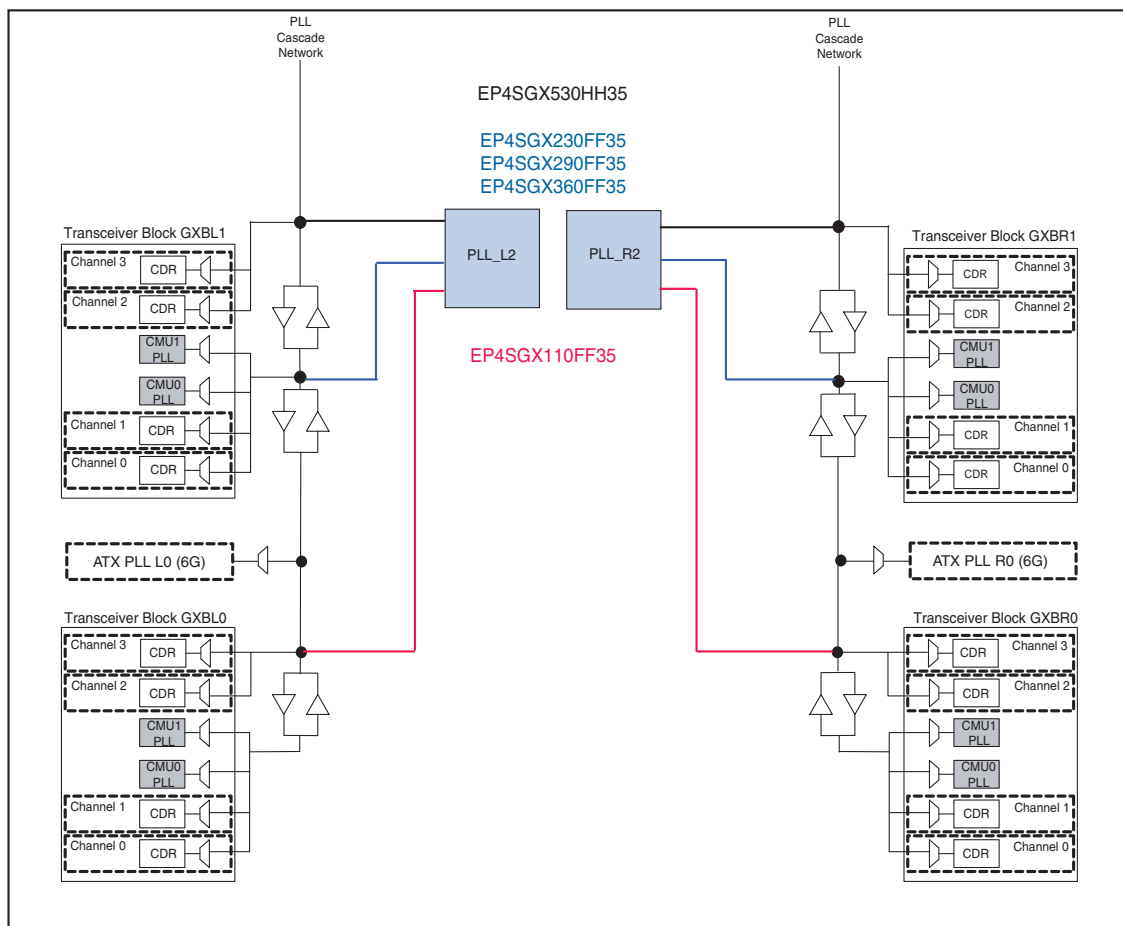
FPGA Fabric PLLs-Transceiver PLLs Cascading in the 780-Pin Package

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

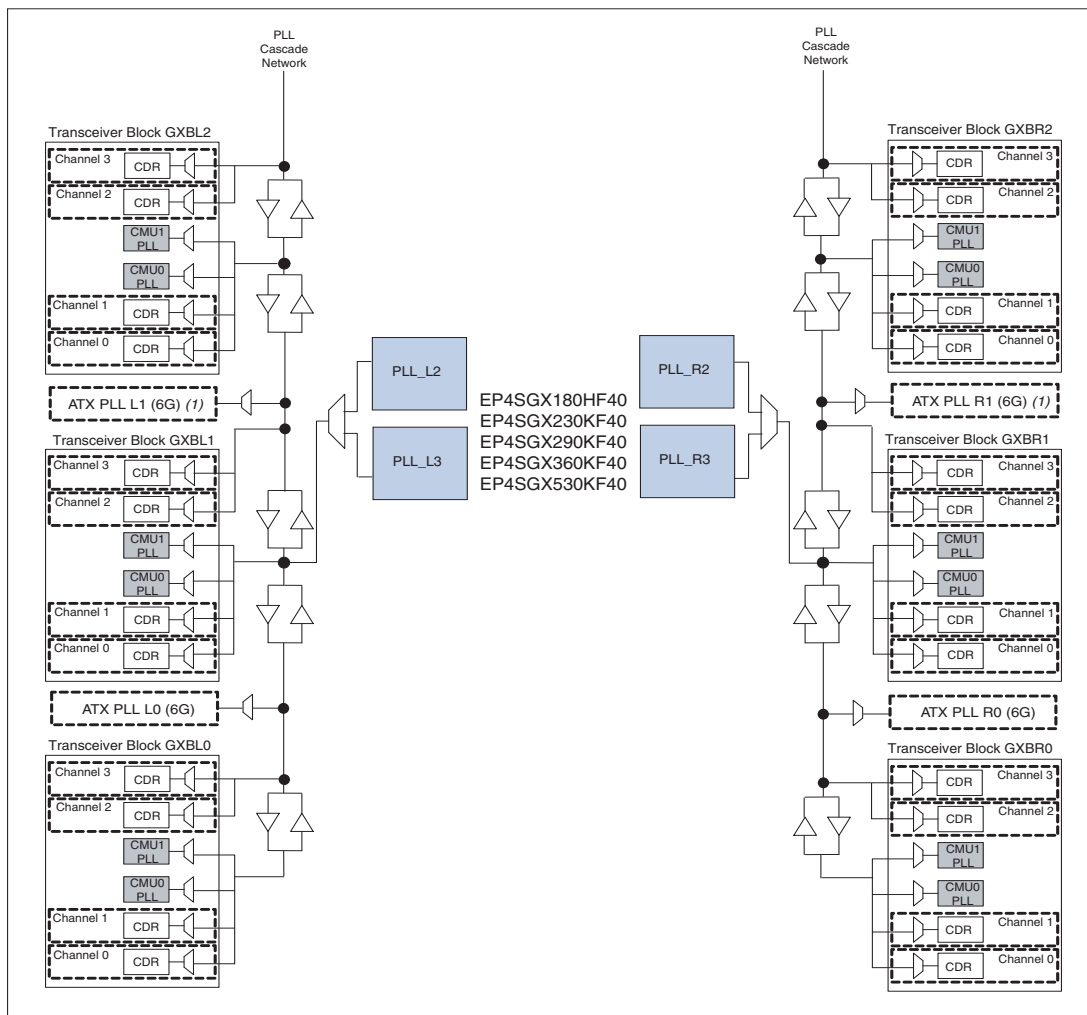


FPGA Fabric PLLs-Transceiver PLLs Cascading in the 1517-Pin Package

Figure 2-7 shows the FPGA fabric PLLs-Transceiver PLLs cascading options allowed in the EP4SGX180HF40, 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:

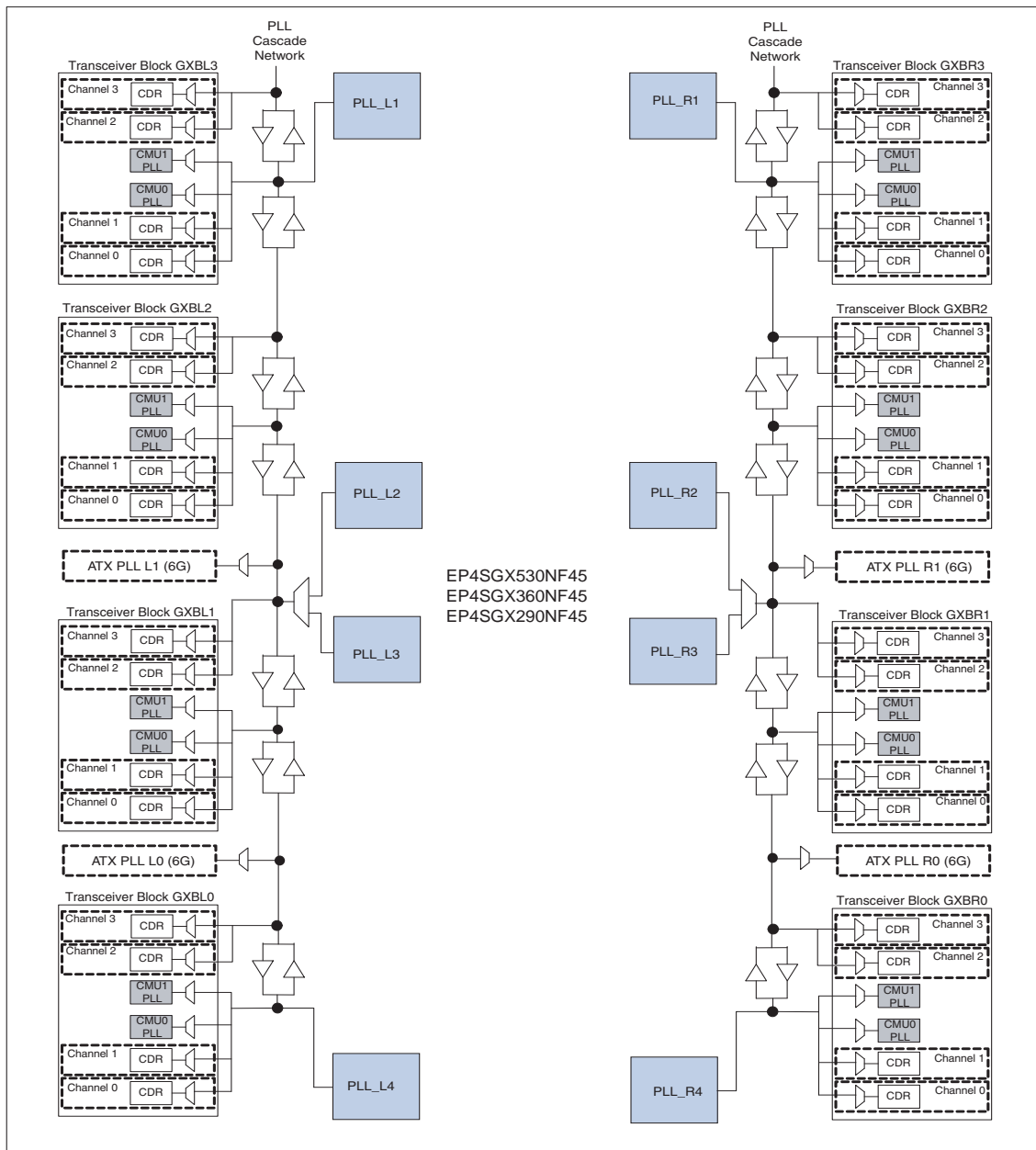
(1) ATX PLL L1 and ATX PLL R1 are not present in the EP4SGX230KF40 device.

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, EP4S100G4NF45, 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

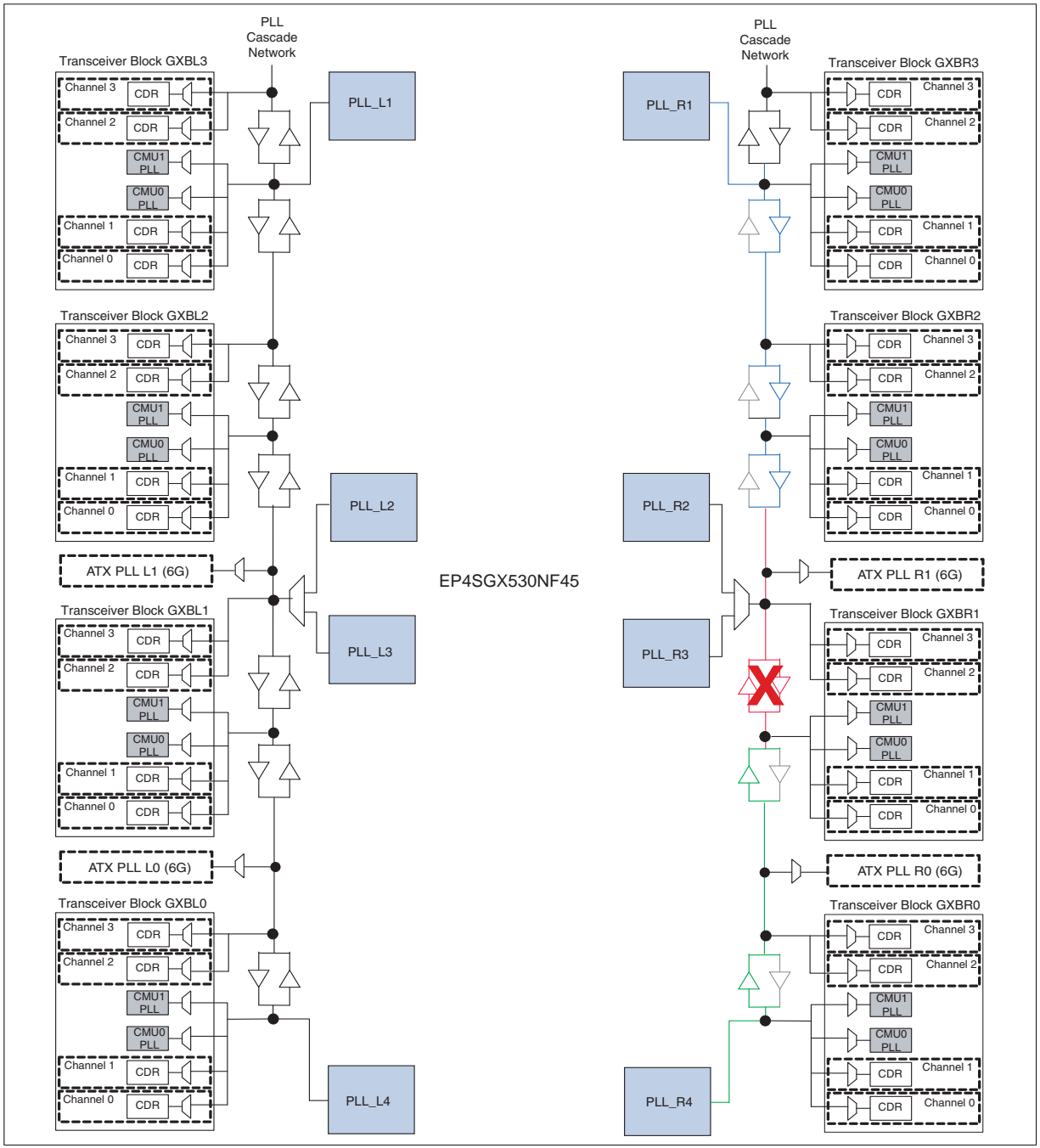
If your design is targeted for a EP4SGX530NF45 device, it requires providing input reference clocks to the following CMU PLLs and receiver CDRs from two right PLLs in the FPGA fabric:

- 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.

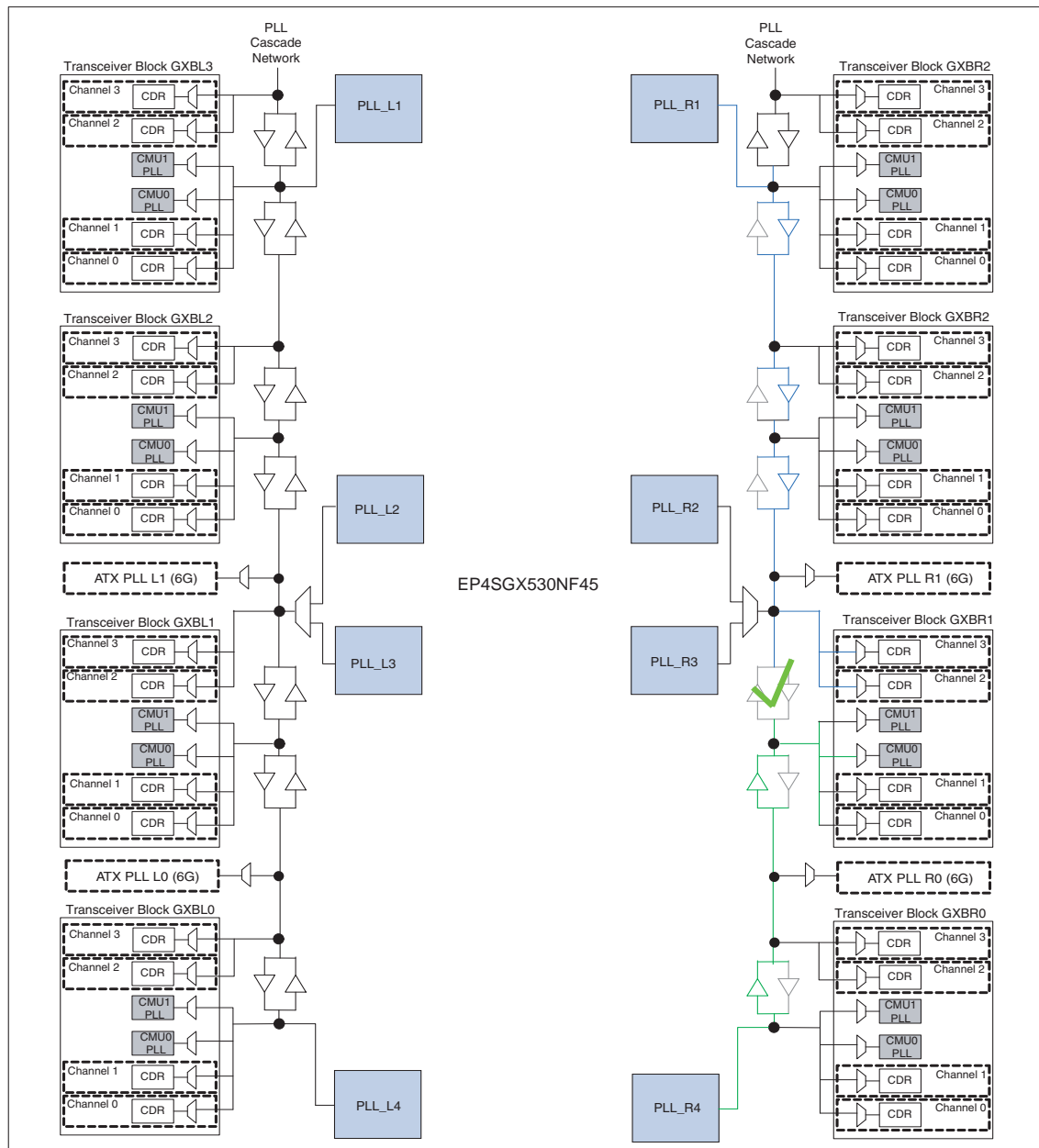
Figure 2-9. Illegal FPGA Fabric-Transceiver PLL Cascading Configuration



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. When in VCO bypass mode, you can only divide the reference clock by the N integer.

For more information about CMU PLLs, refer to “Configuring CMU Channels as Transceiver Channels” in the *Transceiver Architecture in Stratix IV Devices* 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

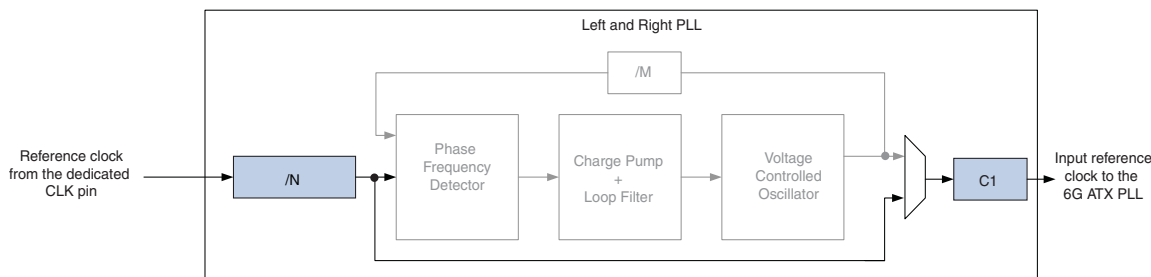
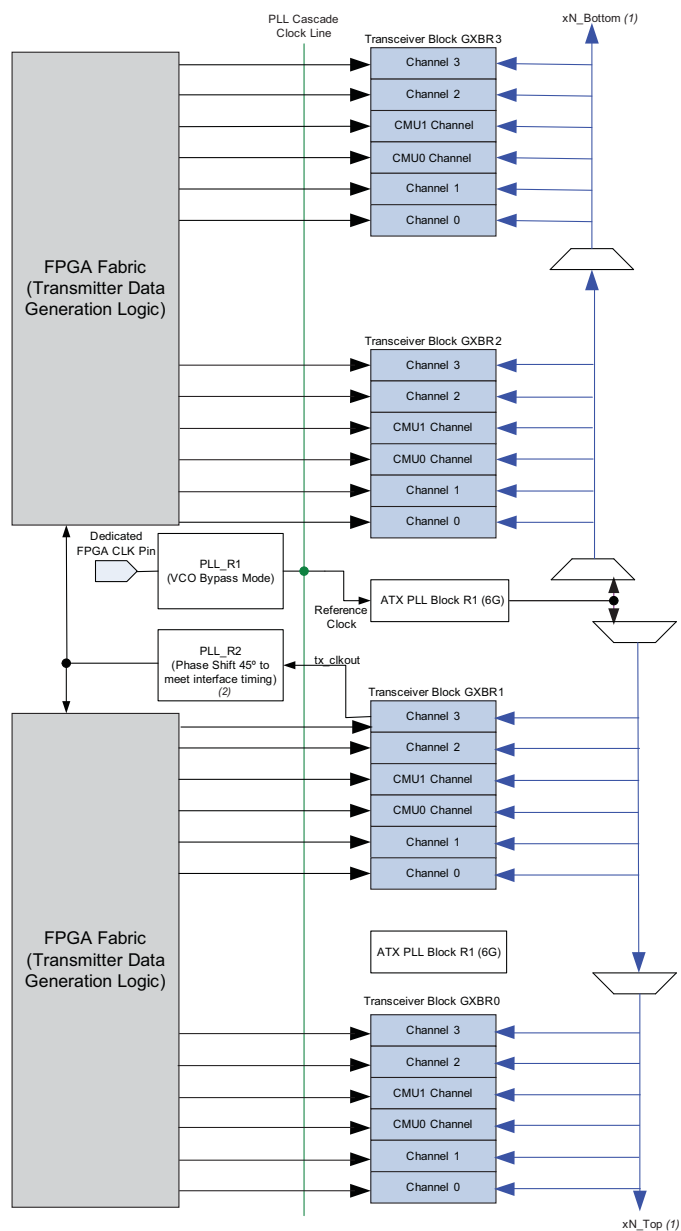


Figure 2-12 shows 24 channels on the right side of the EP4SGX530NF45 device configured in Basic (PMA Direct) $\times N$ mode running at 6.5 Gbps with a 20-bit FPGA fabric-PMA interface width. Because all 24 channels on the right side of the device are configured in Basic (PMA Direct) $\times N$ mode, 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 (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 2-78.

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 2-39



Clocking described in this section is internal to the transceiver and clock routing is primarily performed by the Quartus II software.



For more information about manually picking and placing CMU and ATX PLLs, refer to *AN 578: Manual Placement of CMU PLLs and ATX PLLs in Stratix IV GX and GT Devices*.

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 2-24
- “Bonded Channel Configurations” on page 2-27
- “Non-Bonded Basic (PMA Direct) Mode Channel Configurations” on page 2-34
- “Bonded Basic (PMA Direct) xN Mode Channel Configurations” on page 2-36

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] $\times N$ 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.

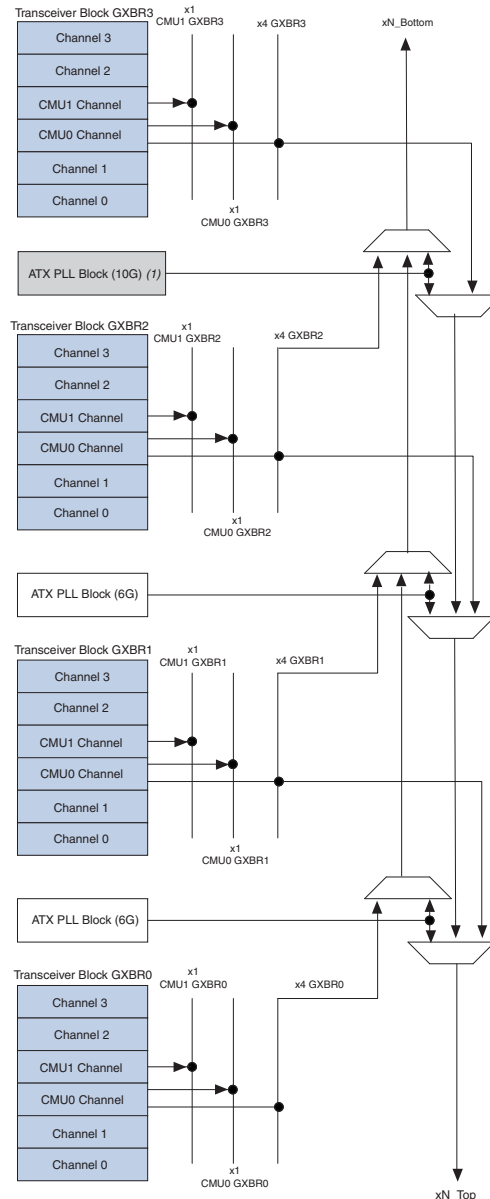
The transmitter phase compensation FIFO in each non-bonded channel (except in Basic [PMA Direct] mode) has its own pointers and control logic that can result in unequal latency in the transmitter phase compensation FIFO of each channel. The higher transceiver clock skew and unequal latency in the transmitter 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 $\times 1$, $\times 4$, and $\times N$ high-speed serial and low-speed parallel clock lines.

Figure 2-13 shows the transceiver clock distribution in $\times 1$, $\times 4$, $\times 8$, and $\times N$ 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:

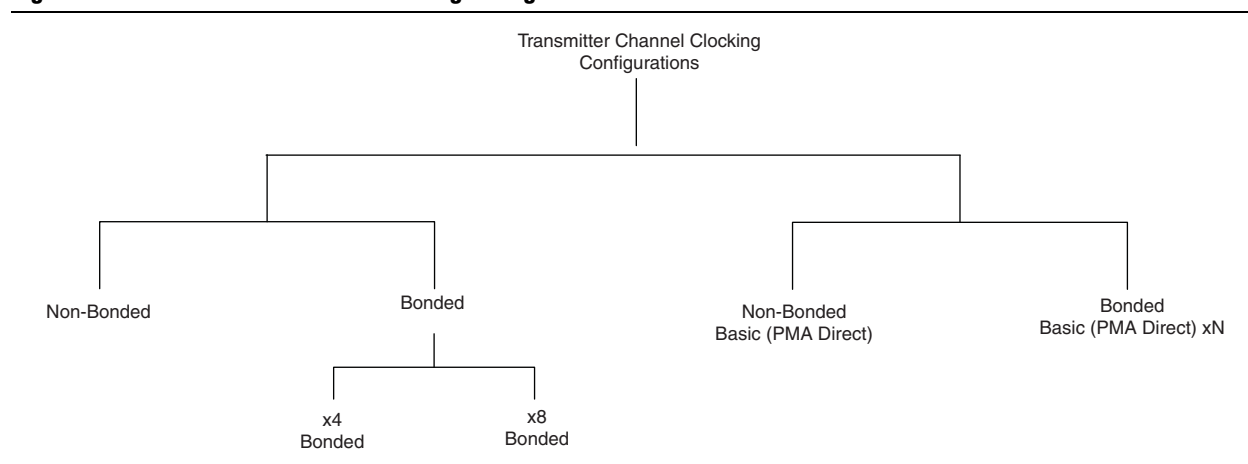
- $\times 1$ non-bonded configurations use the $\times 1$ clock lines to distribute only the high-speed serial transceiver clock synthesized by the CMU0 PLL or CMU1 PLL to the clock transmitter channels located in the same transceiver block. The low-speed parallel transceiver clock is generated in the transceiver channels using the local clock dividers.
- $\times 4$ bonded configurations use the $\times 4_GXB$ 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.
- $\times 8$ and $\times N$ bonded configurations use the $\times N_Top$ or $\times N_Bottom$ 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 $\times N$ lines to distribute the high-speed serial and low-speed parallel transceiver clocks. Use the $\times N_Top$ line if the CMU0 PLL or ATX PLL that generates the transceiver clocks is located at the top of the transmitter channel. Use the $\times N_Bottom$ line if the CMU0 PLL or ATX PLL is located at the bottom of the transmitter channel. Because there is only one $\times N_Top$ and $\times N_Bottom$ line on each side of the device, using an ATX PLL limits the use of the $\times N$ 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

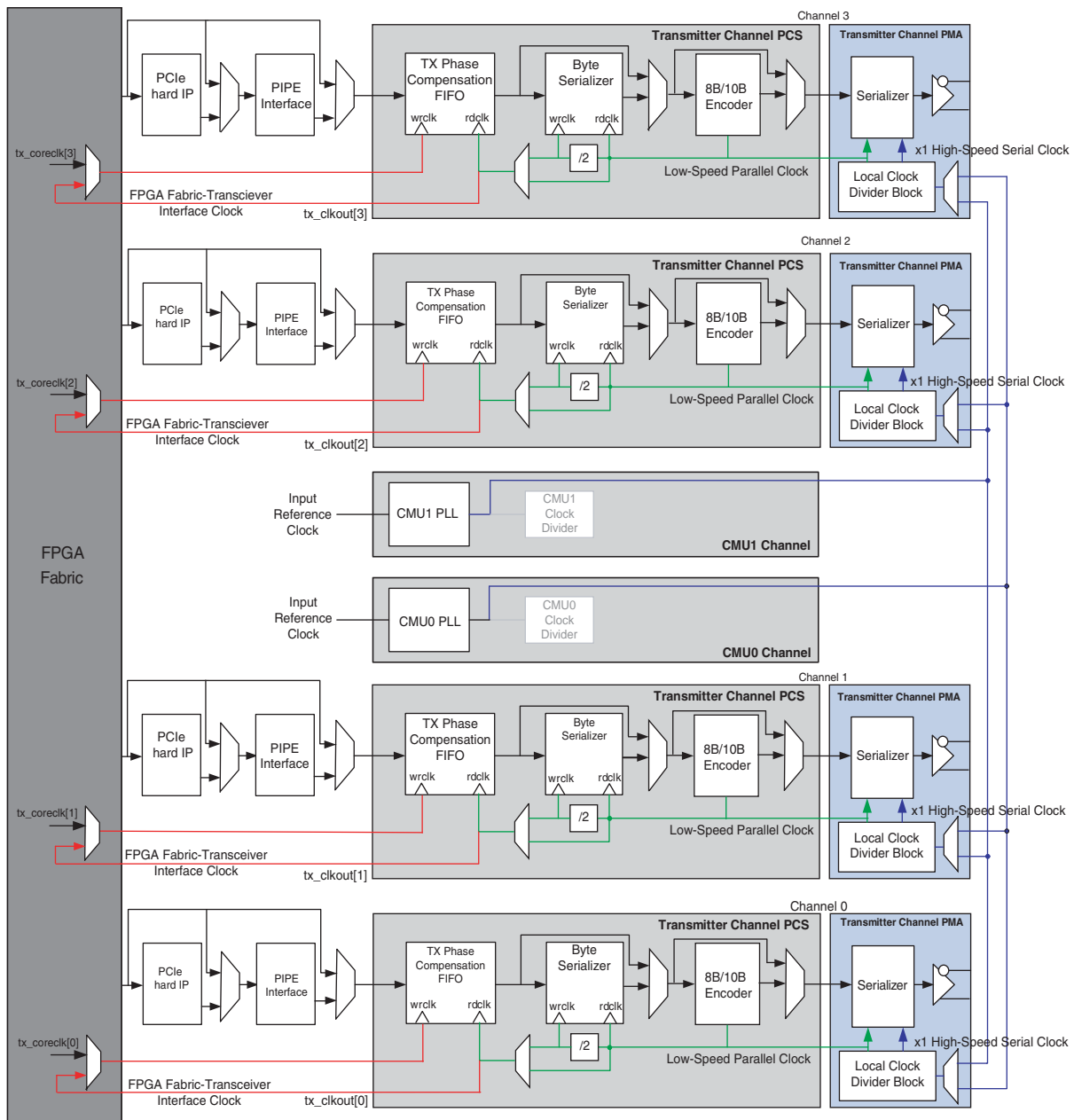
The following modes other than Basic (PMA Direct) functional mode have a non-bonded transmitter channel configuration:

- PCIe ×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, you may use the ATX PLLs if the configured data rate falls within the ATX PLL data rate range specified in the *DC and Switching Characteristics for Stratix IV Devices* 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 PLLs ⁽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 x1 high-speed serial clock.

In non-bonded channel configurations clocked by the 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 $\times 1$ 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 read 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 local 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 `xN_Top` or `xN_Bottom` lines.

 For more information, refer to the *Configuring Multiple Protocols and Data Rates in Stratix IV Devices* 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)
PCIe $\times 1$ (Gen 1)	2.5 Gbps	1.25 GHz	250	250	125
PCIe $\times 1$ (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:

- PCIe ×4—Gen1 and Gen2
- XAUI
- Basic ×4

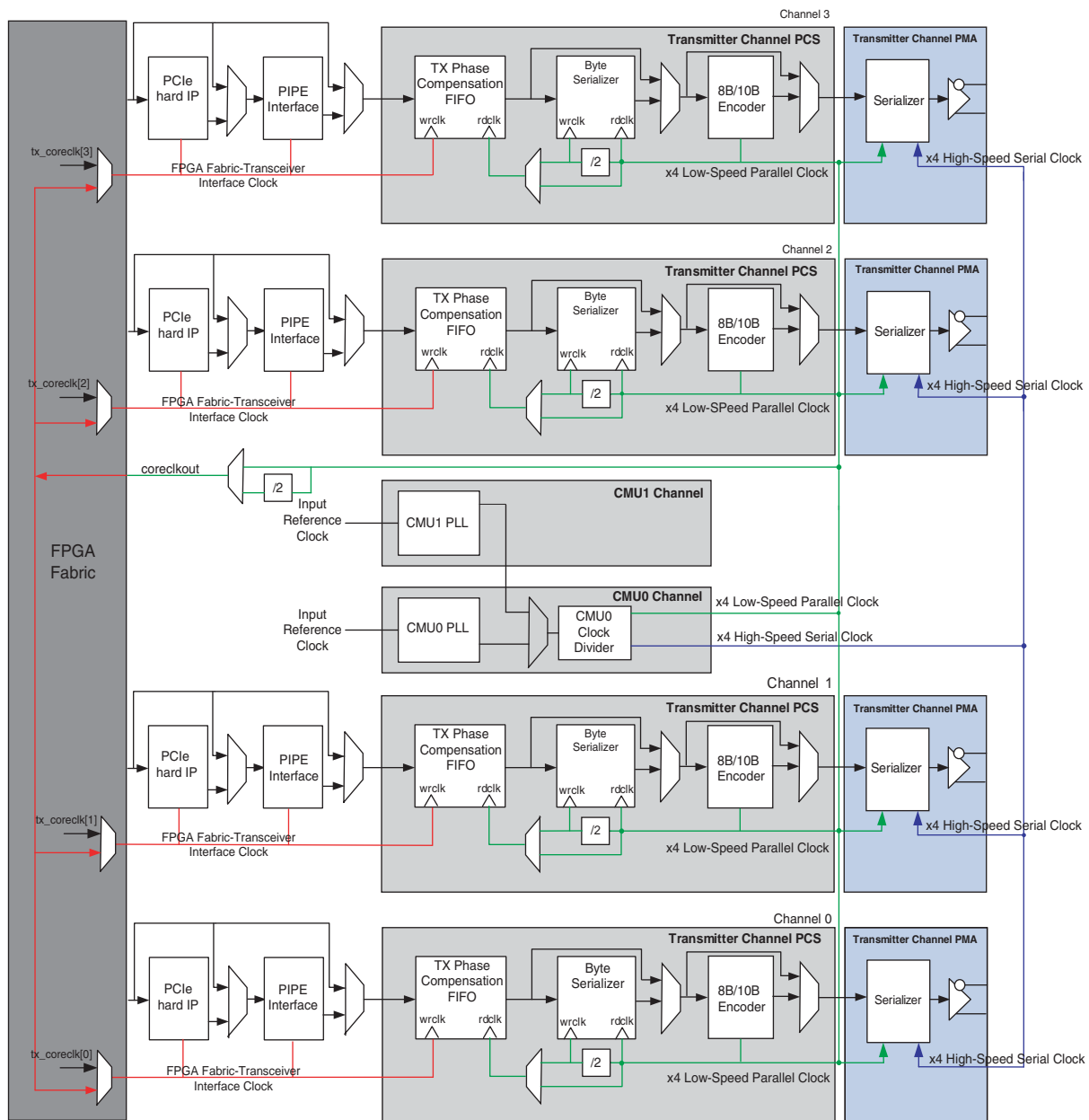
Use the CMU channels to generate the transceiver clocks for all ×4 bonded functional modes listed above. Additionally, you may use the ATX PLLs to generate the transceiver clocks for PCIe ×4 Gen 2 and Basic ×4 functional mode.



You must assign `tx_dataout [0]` of the ×4 bonded link (XAUI or PCIe ×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 x4 channel bonding configurations when clocked using the CMU0 channel.

Figure 2-16. Transmitter Datapath Clocking in x4 Bonded Configurations ⁽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 $\times 4$ high-speed serial and $\times 4$ 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 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 $\times N_{Top}$ or $\times N_{Bottom}$ lines.



For more information, refer to the *Configuring Multiple Protocols and Data Rates in Stratix IV Devices* chapter.

In $\times 4$ 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 $\times 4$ bonded functional modes that have a fixed data rate.

Table 2-7. Transmitter Datapath Clock Frequencies in $\times 4$ Bonded Functional Modes

Functional Mode	Data Rate (Gbps)	High-Speed Serial Clock Frequency (GHz)	Low-Speed Parallel Clock Frequency (MHz)	FPGA Fabric-Transceiver Interface Clock Frequency	
				Without Byte Serializer (MHz)	With Byte Serializer (MHz)
PCIe $\times 4$ (Gen 1)	2.5	1.25	250	250	125
PCIe $\times 4$ (Gen 2)	5	2.5	500	N/A	250
XAUI	3.125	1.5625	312.5	N/A	156.25

$\times 8$ PCS and PMA Bonded Channel Configuration


The following functional modes support $\times 8$ PCS and PMA bonded transmitter channel configuration:

- PCIe $\times 8$ —Gen1 and Gen2
- Basic $\times 8$

Use either the CMU PLL or the ATX PLL to generate the transceiver clocks in Basic $\times 8$ functional modes. Use the ATX PLL in PCIe $\times 8$ Gen2 mode in order to meet the transmitter jitter compliance.

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 $\times N_{\text{Top}}$ 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 $\times N_{\text{Top}}$ 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 $\times 8$ 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 $\times N_{\text{Bottom}}$ lines. It drives the high-speed serial clock and low-speed parallel clock to the slave transceiver block on the $\times N_{\text{Top}}$ lines.

 For more information, refer to the *Configuring Multiple Protocols and Data Rates in Stratix IV Devices* chapter.

In PCIe $\times 8$ and Basic $\times 8$ 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 CCU 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.


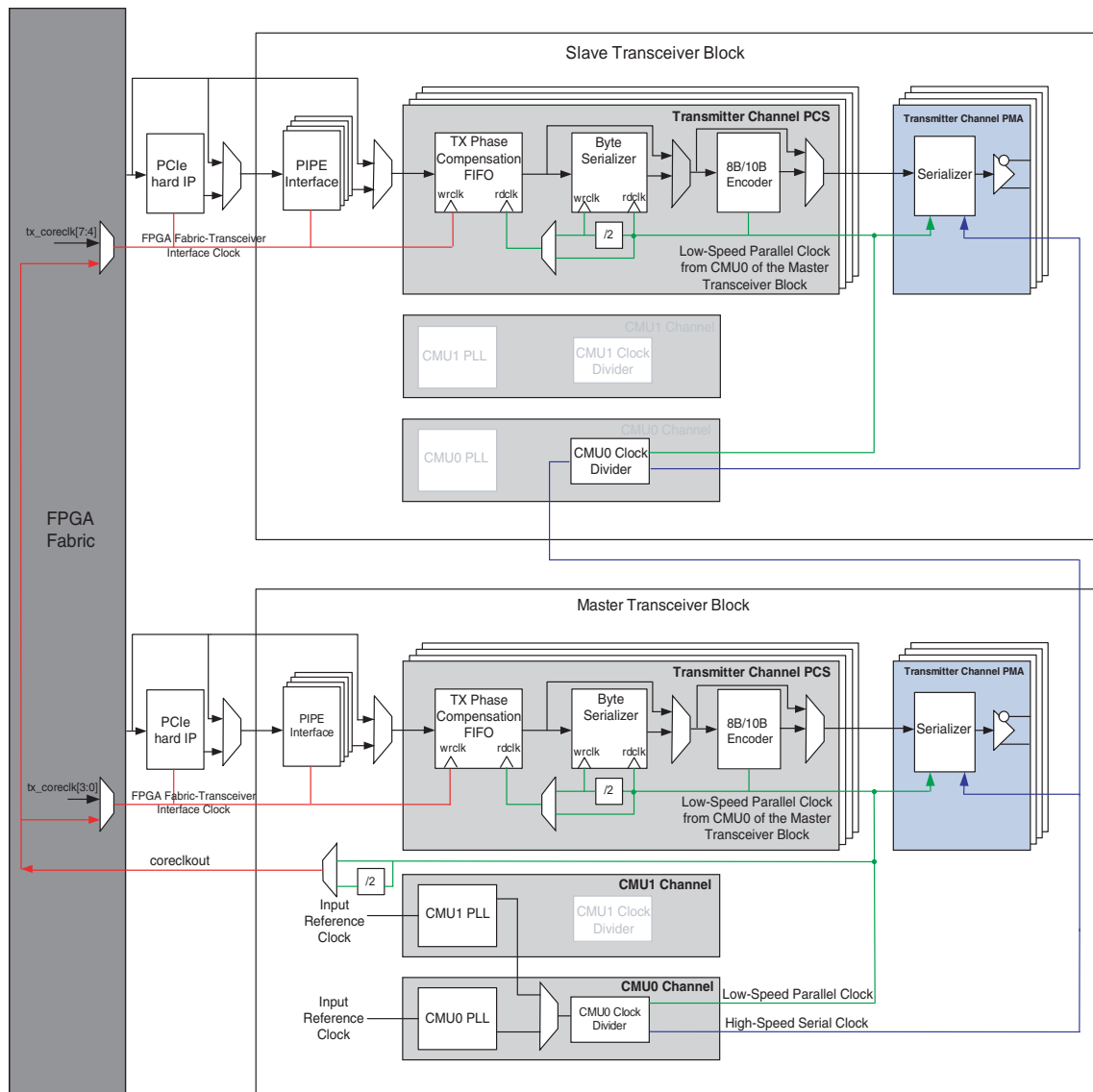
 The difference in clock routing delays between the $\times 4$ clock lines and the $\times N$ clock lines can result in higher transmitter channel-to-channel skew. To compensate for this difference in clock routing delays between the $\times 4$ and the $\times N$ clock lines, the Stratix IV transceivers introduce a fixed amount of delay in the $\times 4$ clock lines of the transceiver block whose CMU0 channel generates the transceiver clocks in Basic $\times 8$ bonded channel configuration.

Figure 2-17 shows the transmitter datapath clocking in PCIe x8 channel bonding configurations when clocked using the CMU channel in the master transceiver block.

Figure 2-17. Transmitter Datapath Clocking in x8 Bonded Configuration (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.

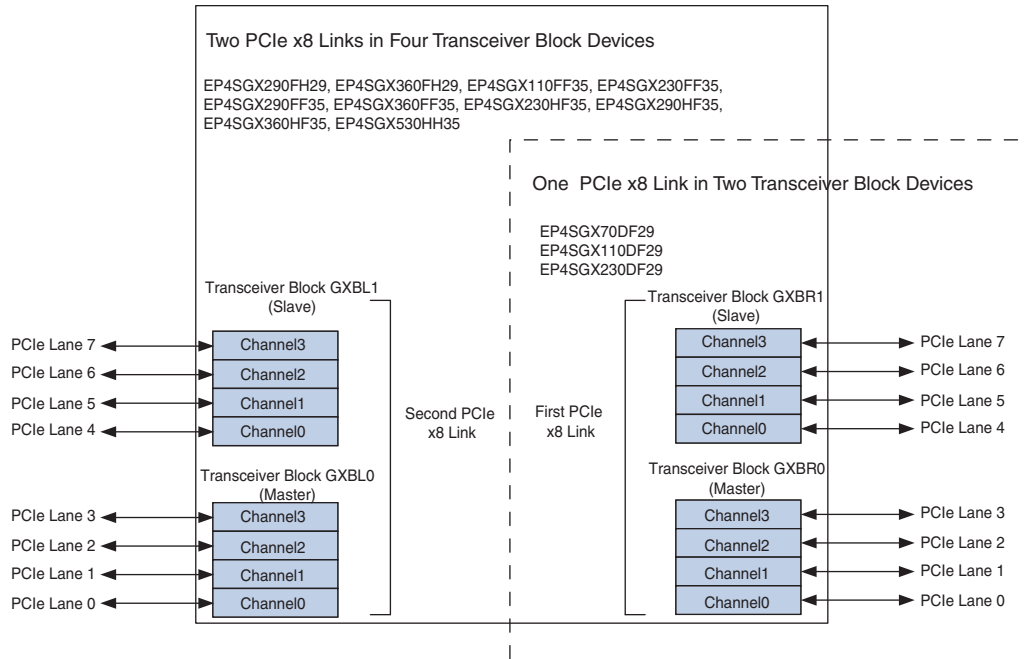
Figure 2-18 through Figure 2-20 show the allowed master and slave transceiver block locations and PCIe logical lane to physical transceiver channel mapping in all Stratix IV devices.



The Quartus II compilation errors out if you do not map the PCIe logical lanes to the physical transceiver channels, as shown in Figure 2-18 through Figure 2-20.

Figure 2-18 shows one PCIe x8 link in two transceiver block devices and two PCIe x8 links in four transceiver block devices.

Figure 2-18. One PCIe x8 Link in Two Transceiver Block Devices and Two PCIe x8 Links in Four Transceiver Block Devices (1)

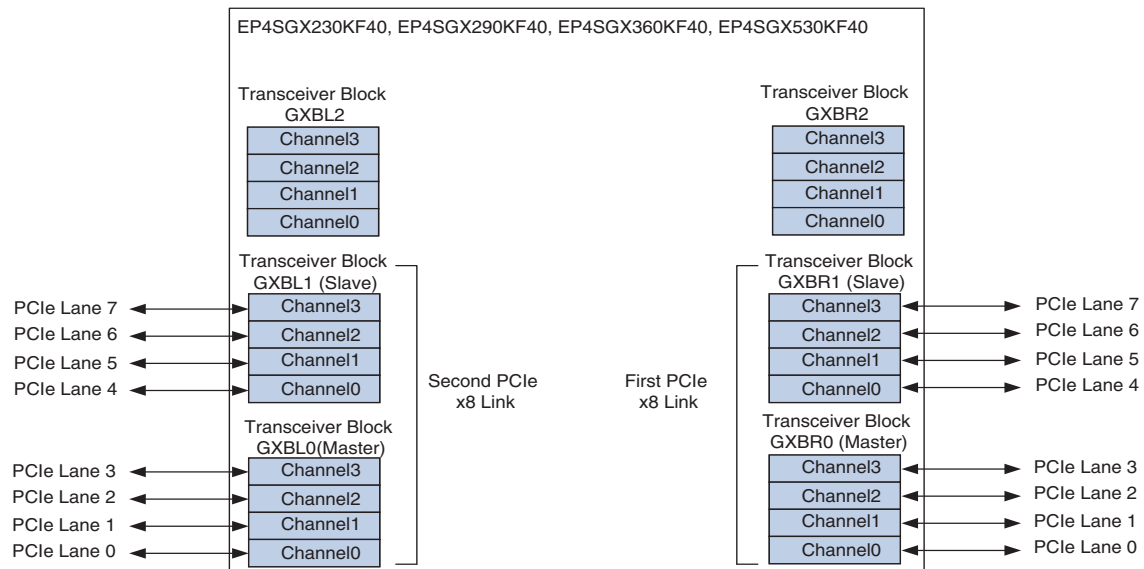


Note to Figure 2-18:

(1) You can use a x4 PCIe configuration in either a master or slave block.

Figure 2-19 shows two PCIe x8 links in six transceiver block devices.

Figure 2-19. Two PCIe x8 Links in Six Transceiver Block Devices (Note 1), (2)

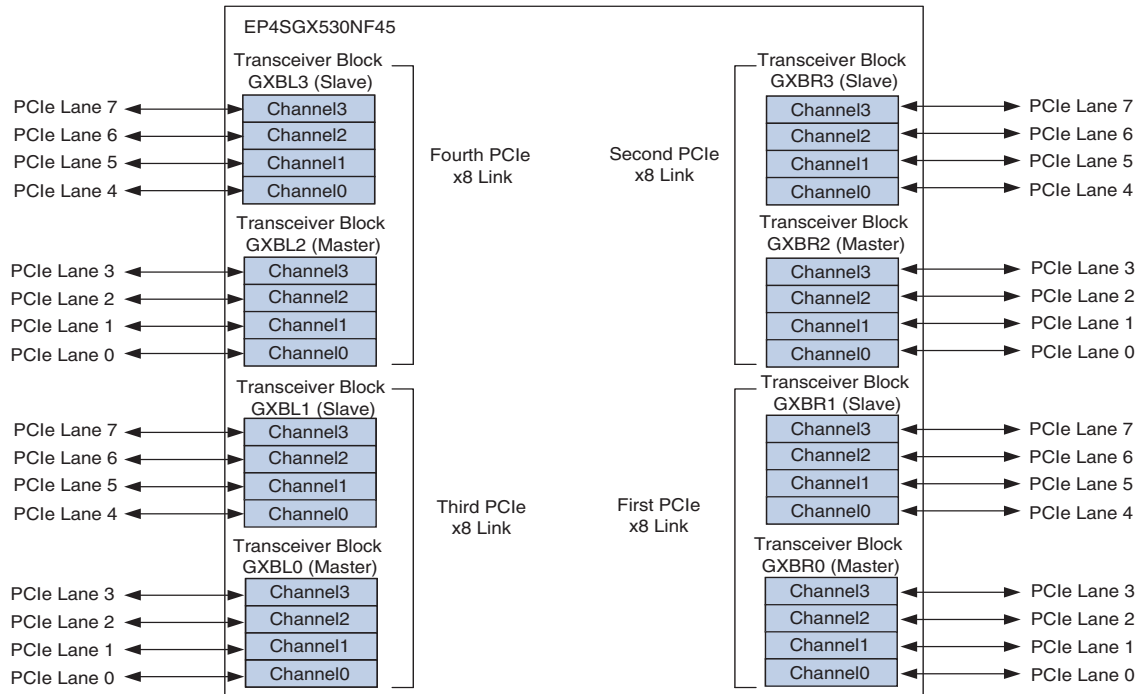


Notes to Figure 2-19:

- (1) Stratix IV devices with six transceiver blocks allow a maximum of two PCIe x8 links occupying four transceiver blocks. You can configure the other two transceiver blocks to implement other functional modes.
- (2) You can use a x4 PCIe configuration in either a master or slave block.

Figure 2–20 shows four PCIe x8 links in eight transceiver block devices.

Figure 2–20. Four PCIe x8 Links in Eight Transceiver Block Devices (1)



Note to Figure 2–20:

(1) You can use a x4 PCIe configuration in either a master or slave block.

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.


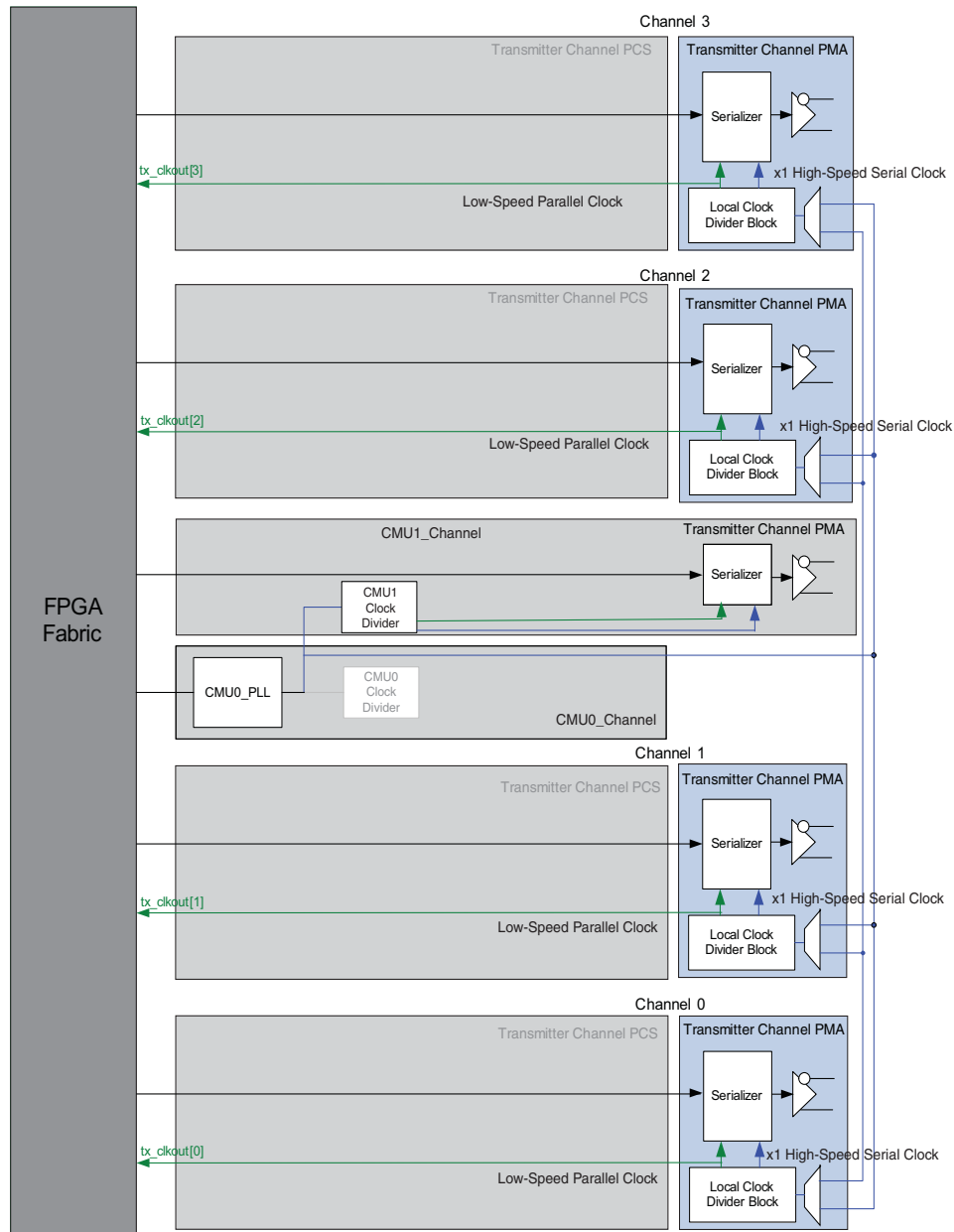

 For more information about Basic (PMA Direct) mode, refer to the *Transceiver Architecture in Stratix IV Devices* chapter.

Figure 2-21. Transmitter Channel PMA Directly Interfacing to the User Logic in the FPGA Fabric ⁽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 $\times 1$ 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) $\times N$ Mode Channel Configurations

Bonded Basic (PMA Direct) $\times N$ 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) $\times N$ 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) $\times N$ 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) $\times N$ configuration:

- If you use the ATX PLL to generate the transceiver datapath interface clocks, only the clock divider of the ATX PLL is used.
- If you use the CMU PLL to generate the transceiver datapath interface 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) $\times N$ mode.

Figure 2-22. Transmitter Channel Clocking for 17 Channels Configured in Basic (PMA Direct) $\times N$ Mode

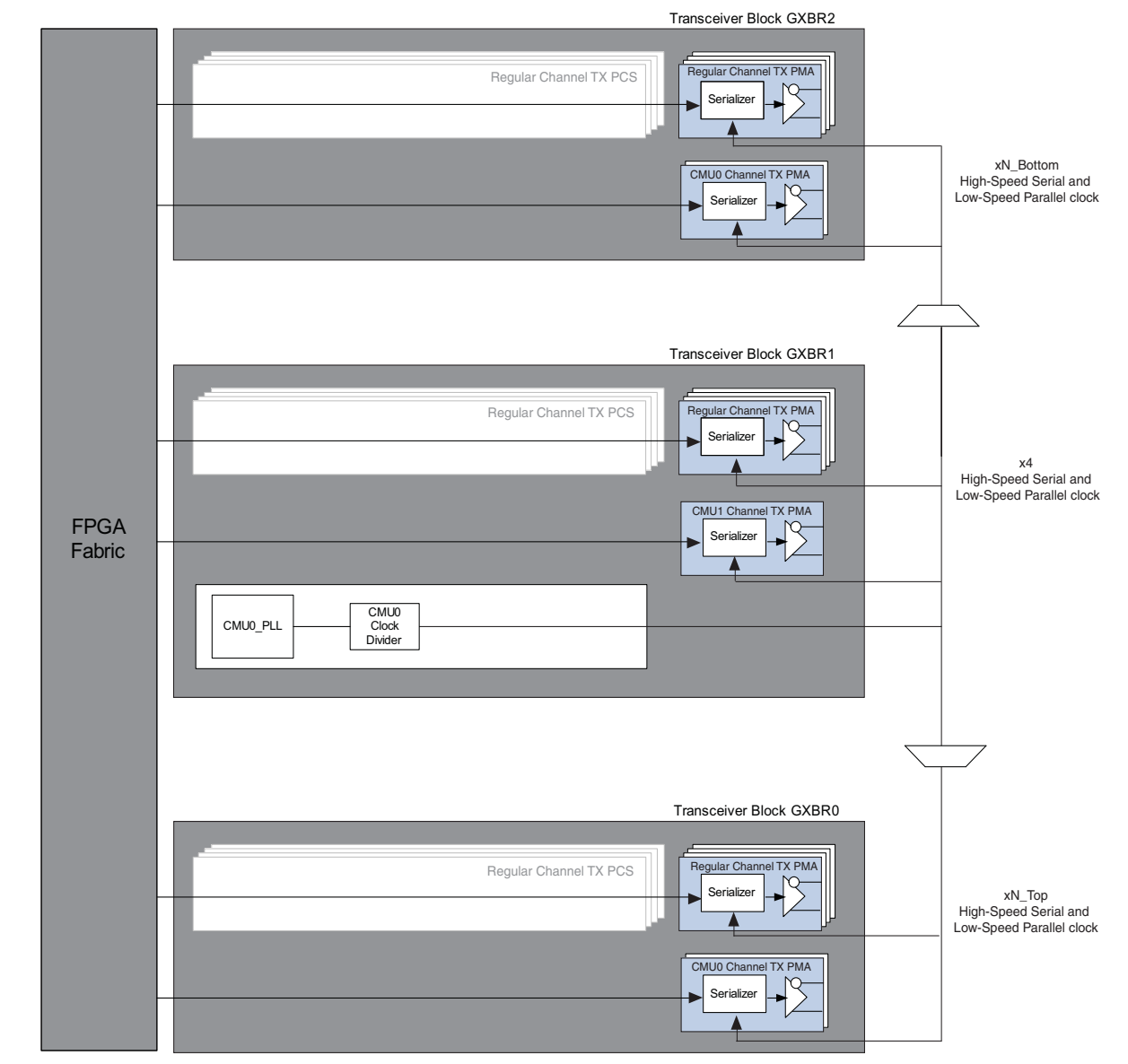




Figure 2-22 shows 17 channels configured in Basic (PMA Direct) $\times N$ mode and located across three transceiver blocks on the right side of the Stratix IV device. Each of the two transceiver blocks, GXBR0 and GXBR2, contain six of the 17 $\times N$ bonded channels located in four regular channels and two CMU channels. The remaining five of the 17 $\times N$ 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) $\times N$ mode.


 For more examples regarding this clocking scheme, refer to:

- “Example 1: Channel Configuration with a 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) $\times N$ Mode

In Basic (PMA-Direct) $\times N$ mode, the CMU0 channel distributes the transceiver clocks to the channels placed in the same transceiver block using the $\times 4$ clock lines. The $\times 4$ clock lines drive the $\times N_Top$ and $\times N_Bottom$ 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 $\times 4$ clock lines and the $\times N$ clock lines can result in higher transmitter channel-to-channel skew. To compensate for this difference in clock routing delays between the $\times 4$ and the $\times N$ clock lines, the Stratix IV transceivers introduce a fixed amount of delay in the $\times 4$ 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 $\times N$ 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 placing the channels in the transceiver blocks adjacent to the ATX PLL on both sides of the ATX PLL.

 For manual placement of the CMU and ATX PLLs, 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*.

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. 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”
- “Bonded Channel Configurations” on page 2-43
- “Basic (PMA Direct) Mode Channel Configurations” on page 2-49

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 in Stratix IV Devices* chapter.

In non-bonded channel configurations, receiver channel datapath clocking has two scenarios:

- “Non-Bonded Receiver Clocking Without Rate Matcher”
- “Non-Bonded Receiver Clocking with Rate Matcher” on page 2-41

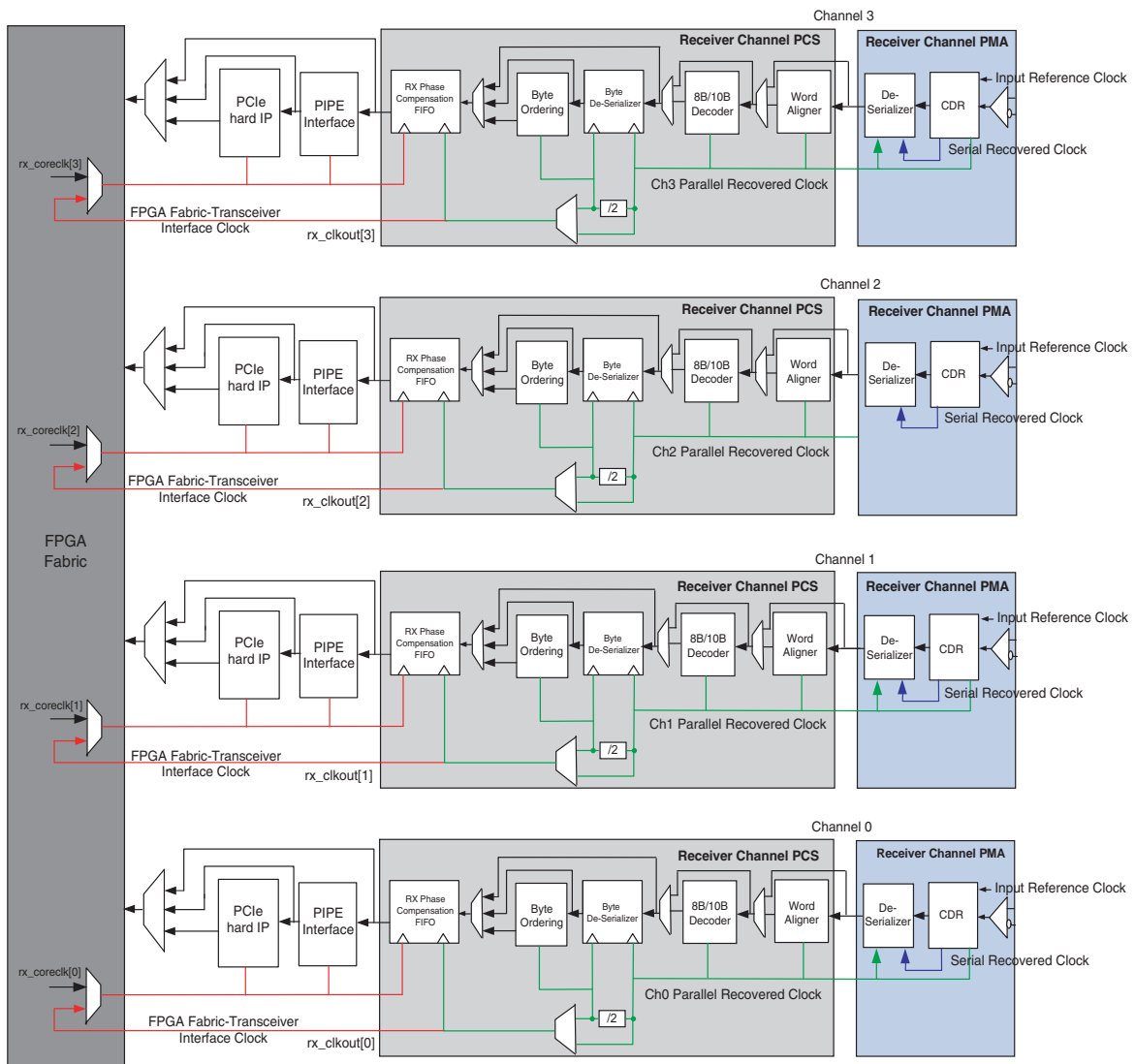
Non-Bonded Receiver Clocking Without Rate Matcher

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 ⁽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

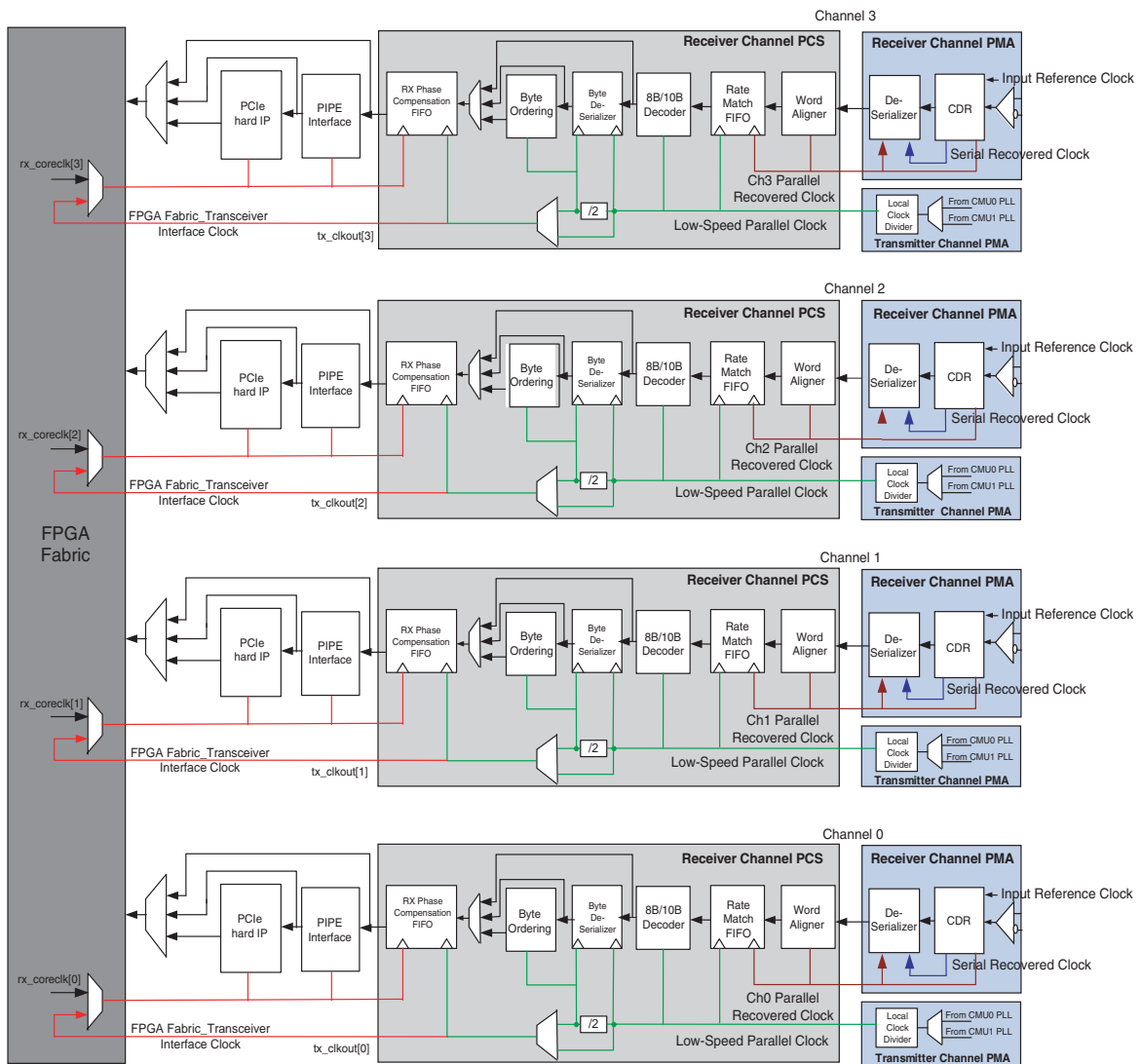
Non-Bonded Receiver Clocking with Rate Matcher

The following functional modes have non-bonded receiver channel configuration with rate-matcher:

- PCIe x1
- 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 (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 are 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 (Gbps)	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)
PCIe x1 (Gen 1)	2.5	1.25 GHz	250	250	125
PCIe x1 (Gen 2)	5	2.5 GHz	500	N/A	250
GIGE	1.25	625 MHz	125	125	N/A
Serial RapidIO	1.25	625 MHz	125	N/A	62.5
	2.5	1.25 GHz	250	N/A	125
	3.125	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 CCU, 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 PCIe ×4 mode, as shown in Figure 2-26 on page 2-46 and PCIe ×8 mode, as shown in Figure 2-27 on page 2-48.

×4 Bonded Channel Configuration

The following functional modes support ×4 receiver channel bonded configuration:

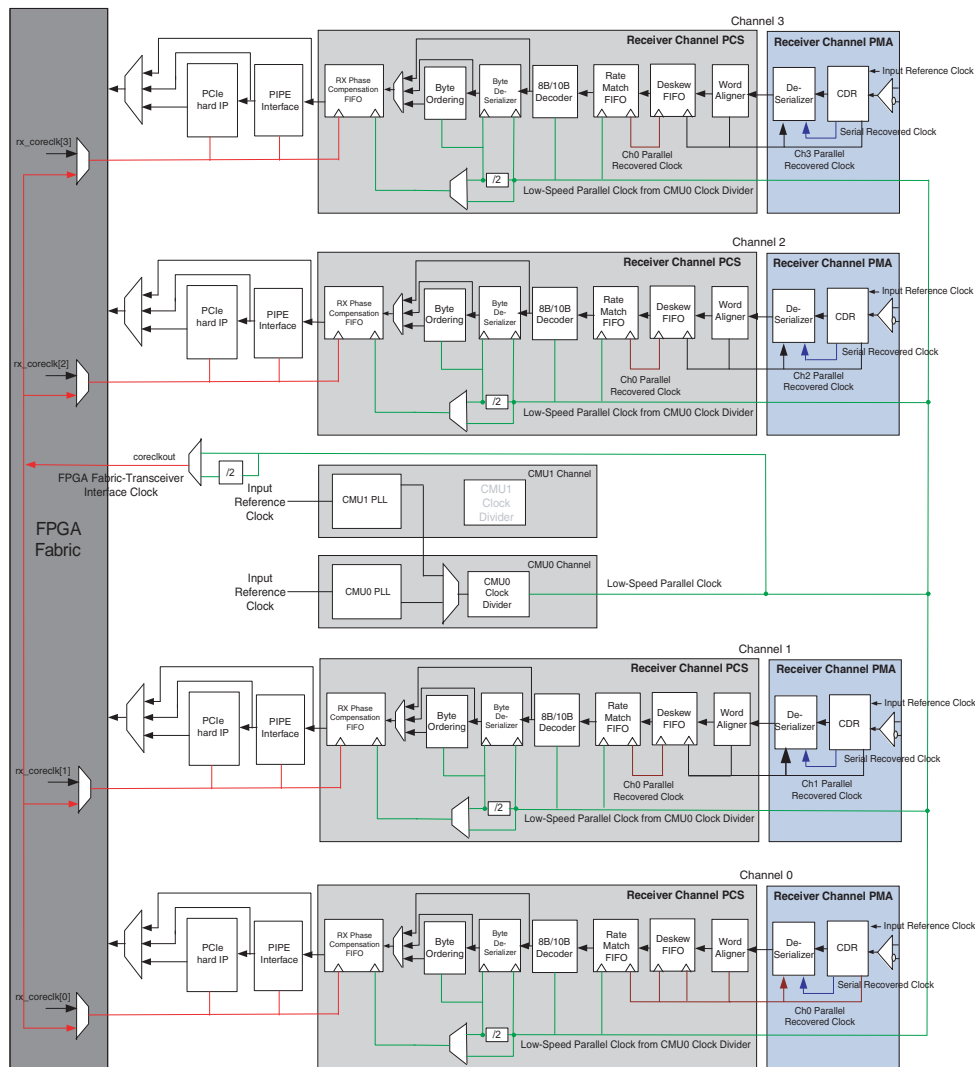
- XAUI (“×4 Bonded Channel Configuration with Deskew FIFO” on page 2-44)
- PCIe (“×4 Bonded Channel Configuration Without Deskew FIFO” on page 2-46)

x4 Bonded Channel Configuration with Deskew FIFO

XAUI functional mode has $\times 4$ bonded channel configuration with deskew FIFO.

Figure 2-25 shows the receiver datapath clocking in $\times 4$ channel bonding configurations with deskew FIFO.

Figure 2-25. Receiver Datapath Clocking in x4 Bonded Channel Configuration with Deskew FIFO ⁽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 $\times 4$ 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 ×4 Bonded Functional Modes with Deskew FIFO

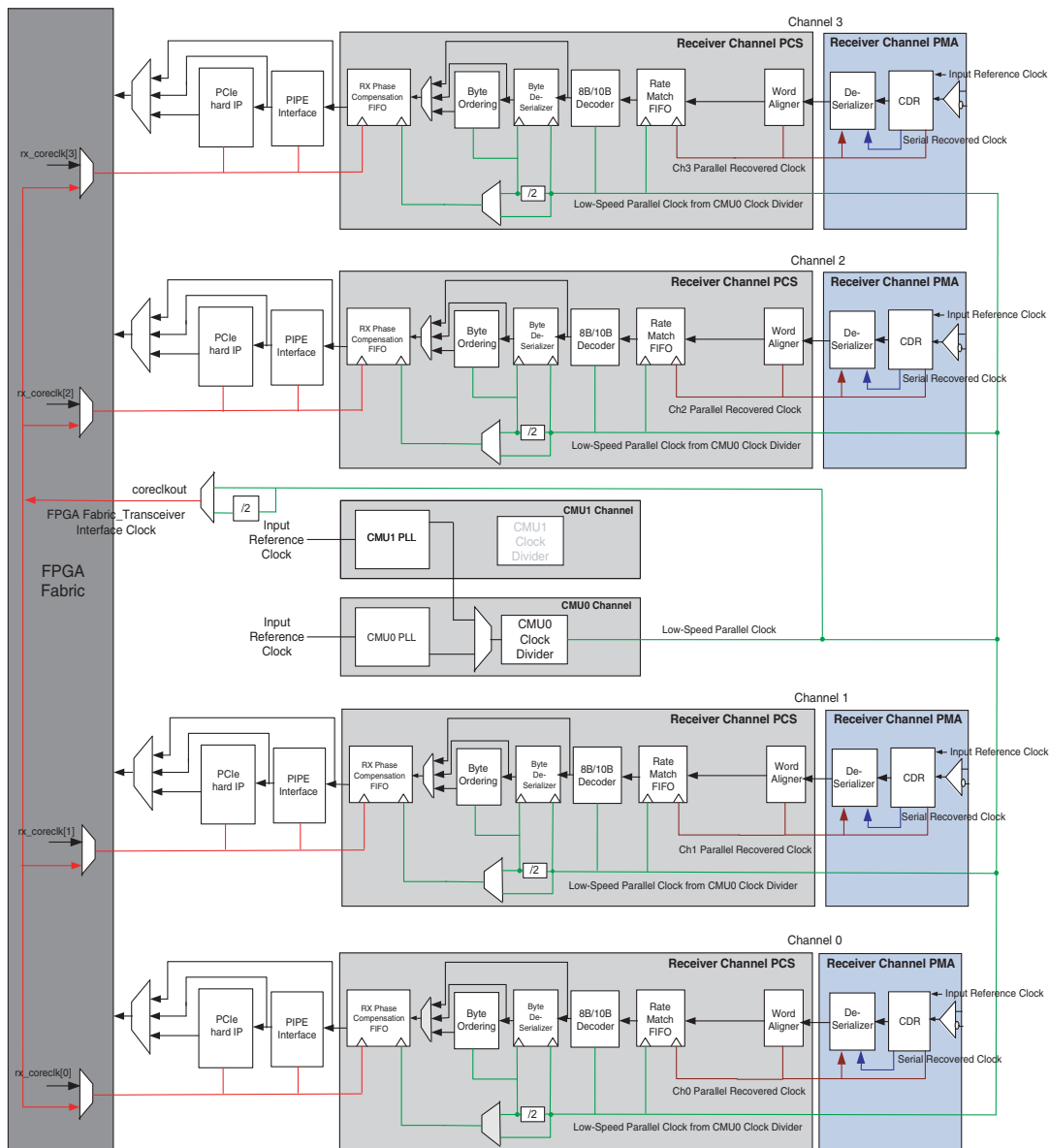
Functional Mode	Data Rate (Gbps)	Serial Recovered Clock Frequency	Parallel Recovered Clock and Parallel Transmitter PCS Clock Frequency (MHz)	FPGA-Fabric Transceiver Interface Clock Frequency	
				Without Byte Deserializer (MHz)	With Byte Deserializer (MHz)
PCIe ×4 (Gen 1)	2.5	1.25 GHz	250	250	125
PCIe ×4 (Gen 2)	5	2.5 GHz	500	N/A	250
XAUI	3.125	1.5625 MHz	312.5	N/A	156.25

x4 Bonded Channel Configuration Without Deskew FIFO

PCIe x4 functional modes supports the x4 bonded channel configuration without deskew FIFO.

Figure 2-26 shows the receiver datapath clocking in x4 channel bonding configurations without deskew FIFO.

Figure 2-26. Receiver Datapath Clocking in x4 Bonded Channel Configuration Without Deskew FIFO (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 $\times 4$ 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 $\times 4$ bonded functional modes without deskew FIFO.

Table 2-12. Receiver Datapath Clock Frequencies in $\times 4$ Bonded Functional Modes without Deskew FIFO

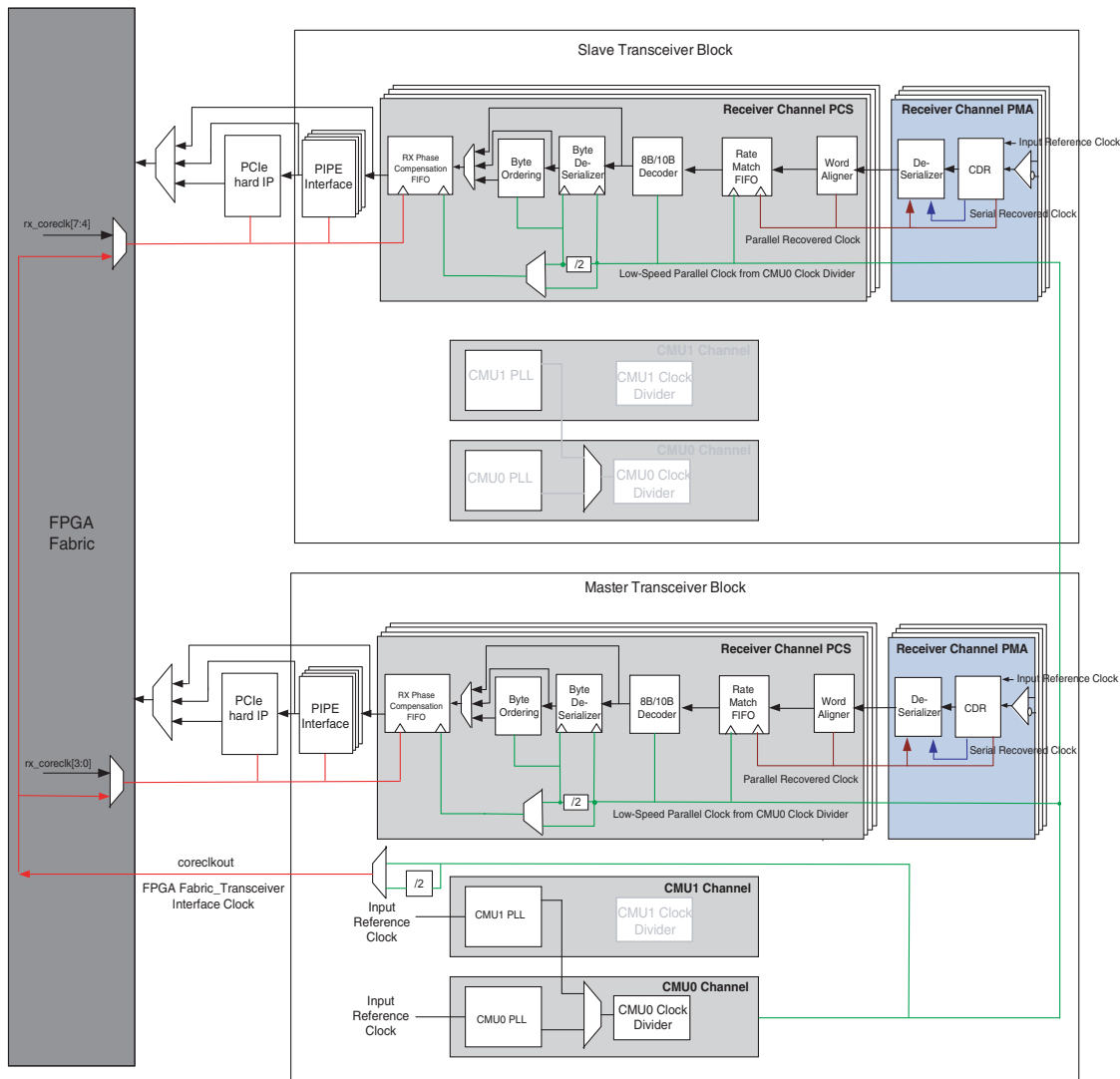
Functional Mode	Data Rate (Gbps)	Serial Recovered Clock Frequency (GHz)	Parallel Recovered Clock and Parallel Transmitter PCS Clock Frequency (MHz)	FPGA Fabric-Transceiver Interface Clock Frequency	
				Without Byte Deserializer (MHz)	With Byte Deserializer (MHz)
PCIe $\times 4$ (Gen 1)	2.5	1.25	250	250	125
PCIe $\times 4$ (Gen 2)	5	2.5	500	N/A	250

x8 Bonded Channel Configuration

PCIe x8 functional mode supports the x8 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 PCIe x8 bonded channel configuration.

Figure 2-27. Receiver Datapath Clocking in x8 Bonded Channel Configuration (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 PCIe x8 functional mode.

Table 2-13. Receiver Datapath Clock Frequencies PCIe x8 Functional Mode

Functional Mode	Data Rate (Gbps)	Serial Recovered Clock Frequency (GHz)	Parallel Recovered Clock and Parallel Transmitter PCS Clock Frequency (MHz)	FPGA Fabric-Transceiver Interface Clock Frequency	
				Without Byte Deserializer (MHz)	With Byte Deserializer (MHz)
PCIe x8 (Gen 1)	2.5	1.25	250	250	125
PCIe x8 (Gen 2)	5	2.5	500	N/A	250

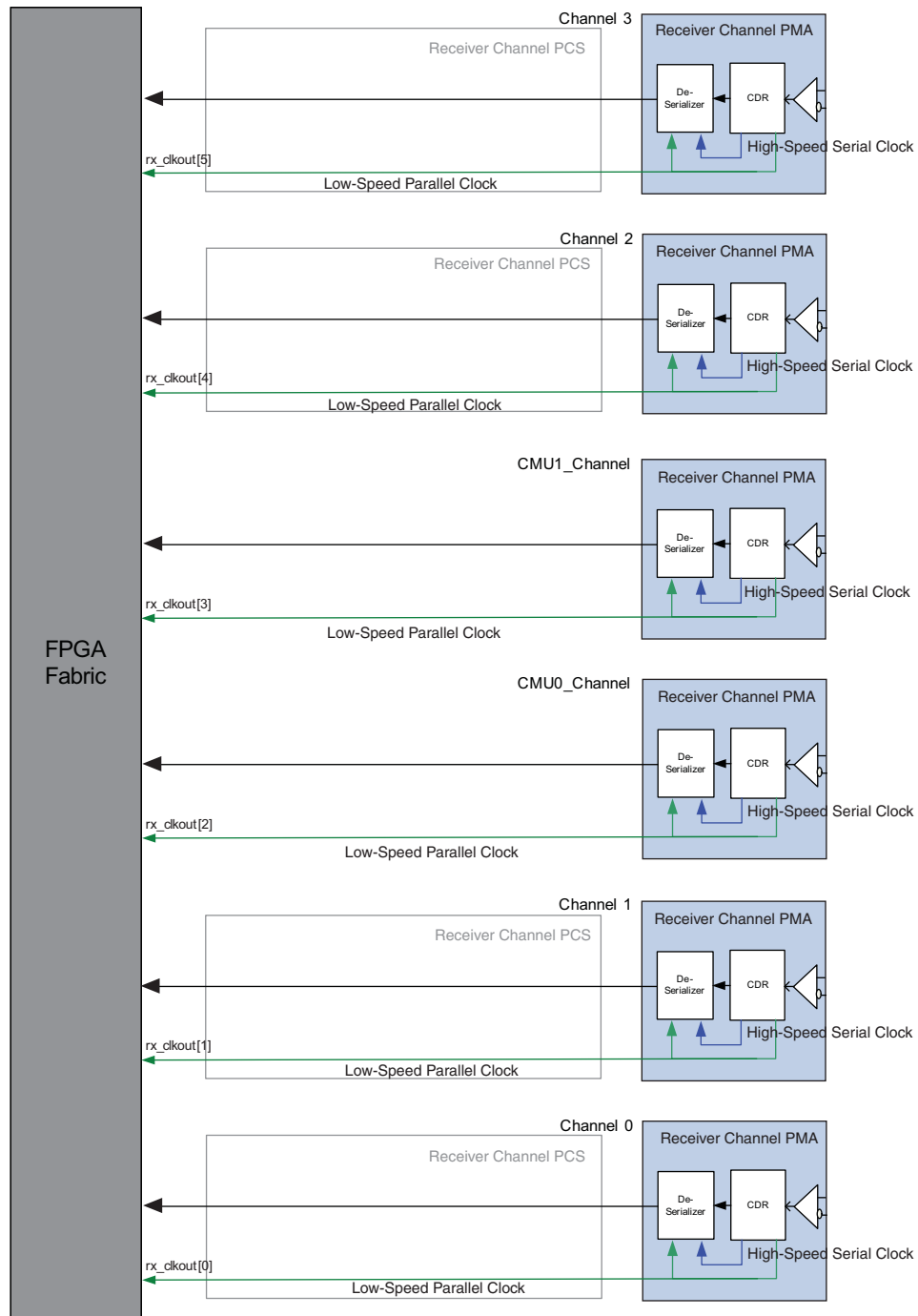
Basic (PMA Direct) Mode Channel Configurations

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 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 (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 Interface 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 PCIe 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	Global clock
rx_cruclk	Receiver CDR input reference clock when driven from an FPGA CLK input pin	FPGA fabric-to-transceiver	Global clock, Regional clock
tx_clkout	Phase compensation FIFO clock	Transceiver-to-FPGA fabric	Global clock, Regional clock, Periphery clock
coreclkout	Phase compensation FIFO clock	Transceiver-to-FPGA fabric	Global clock, Regional clock, Periphery clock
rx_clkout	Phase compensation FIFO clock	Transceiver-to-FPGA fabric	Global clock, Regional clock, Periphery clock
fixed_clk	PCIe receiver detect clock	FPGA fabric-to-transceiver	Global clock, Regional clock

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)
reconfig_clk (2)	Transceiver dynamic reconfiguration clock	FPGA fabric-to-transceiver	Global clock
cal_blk_clk	Transceiver calibration block clock	FPGA fabric-to-transceiver	Global clock, Regional clock

Notes to Table 2-11:

- (1) For more information about global, regional, and periphery clock 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 2-52 and “FPGA Fabric-Receiver Interface Clocking” on page 2-61 describe the criteria and methodology to share transmitter and receiver phase compensation FIFO clocks in order to reduce the global, regional, and periphery clock resource usage 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”
- “User-Selected Transmitter Phase Compensation FIFO Write Clock” on page 2-58



User-selection is provided to share transceiver datapath interface clocks in order to reduce the global, regional, and periphery clock resource usage 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

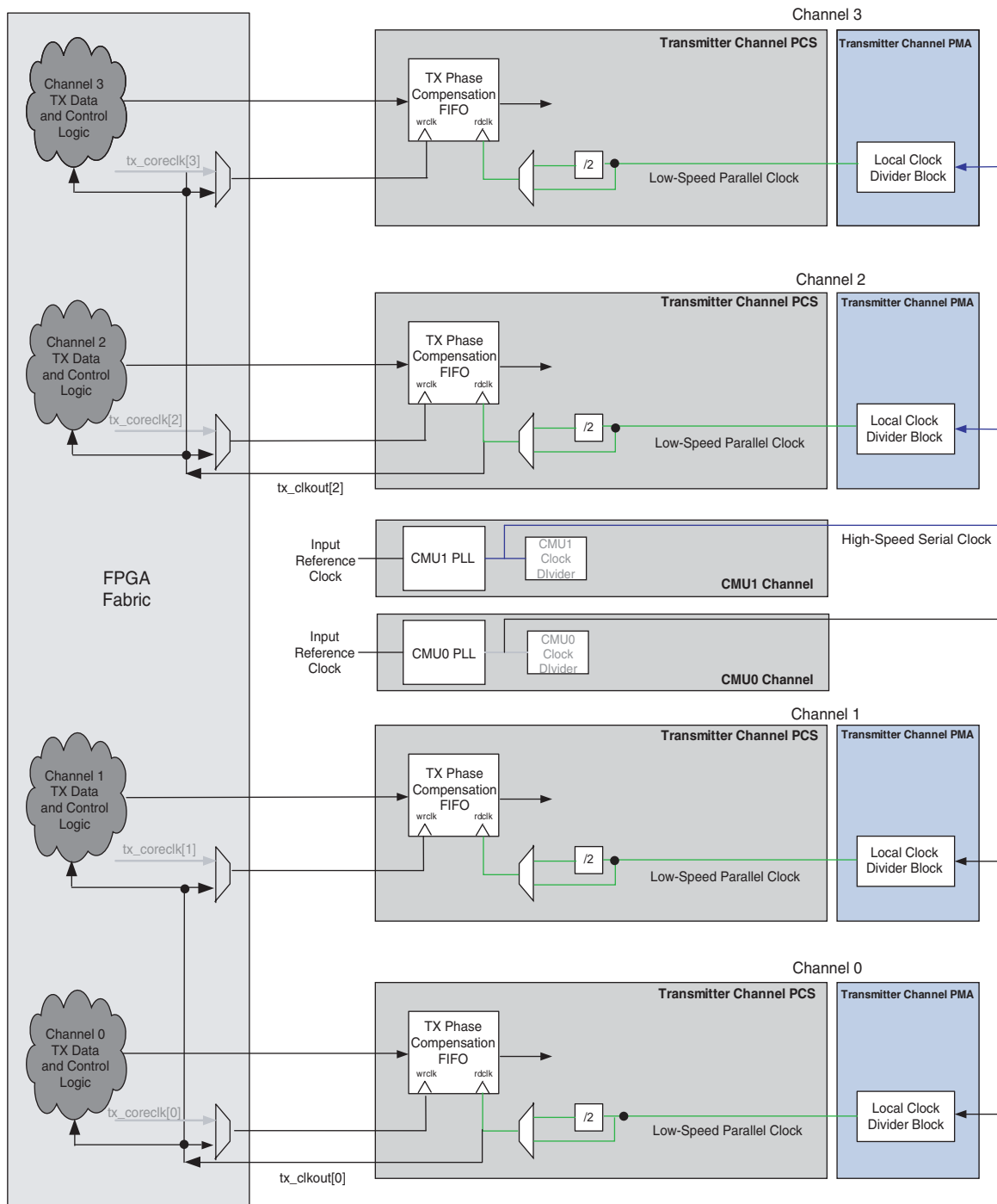
Example 3 assumes channels 0 and 1, driven by `CMU0_PLL` in a transceiver block, are identical. Also, channels 2 and 3, driven by `CMU1_PLL` in the same transceiver block, are identical. In this case, the Quartus II software automatically drives the write port of the transmitter phase compensation FIFO 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 ⁽¹⁾



Note to Figure 2-29:

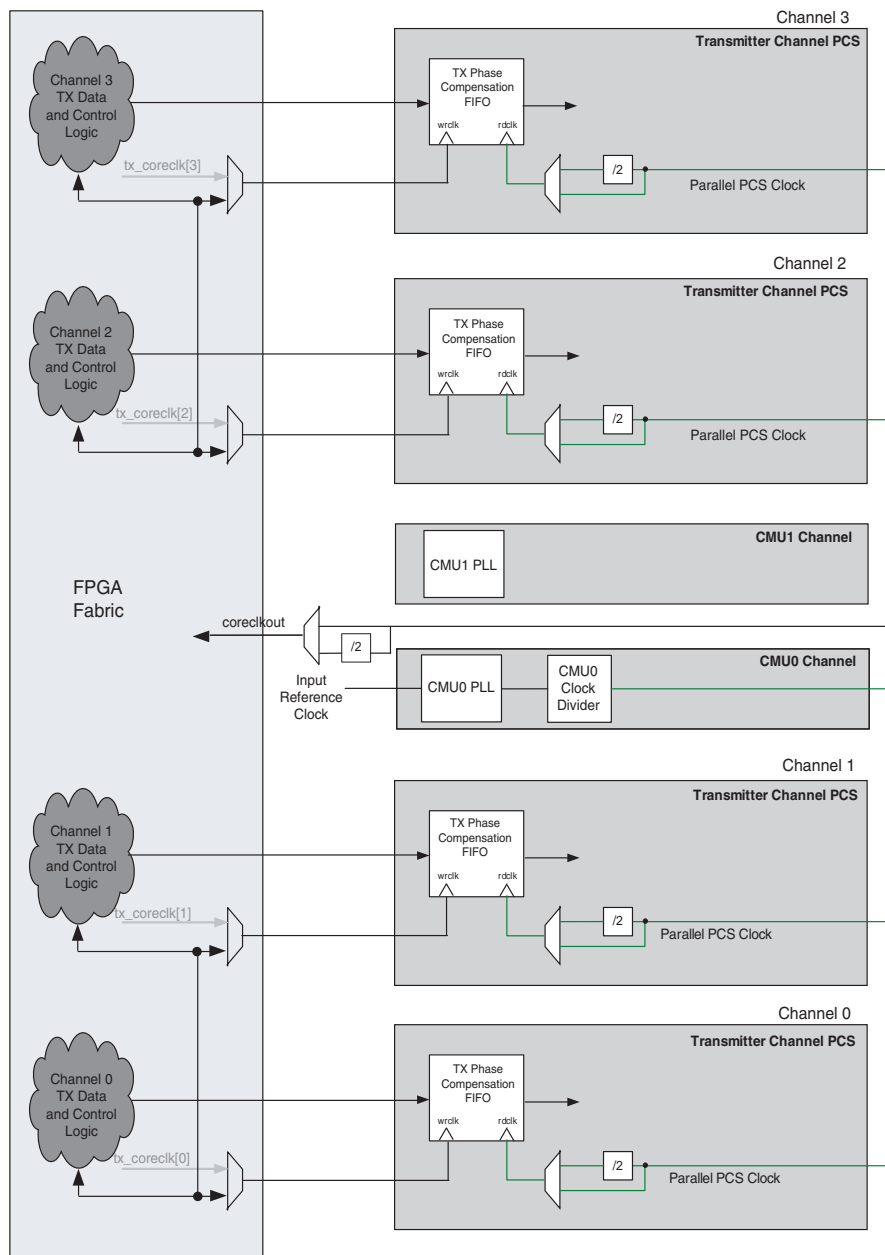
(1) The green lines represent the low-speed parallel clock and the blue lines represent the high-speed serial clock.

Bonded Channel Configuration

In $\times 4$ and $\times 8$ 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 $\times 4$ bonded channel configuration.

Figure 2-30. FPGA Fabric-Transmitter Interface Clocking in a $\times 4$ Bonded Channel Configuration (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

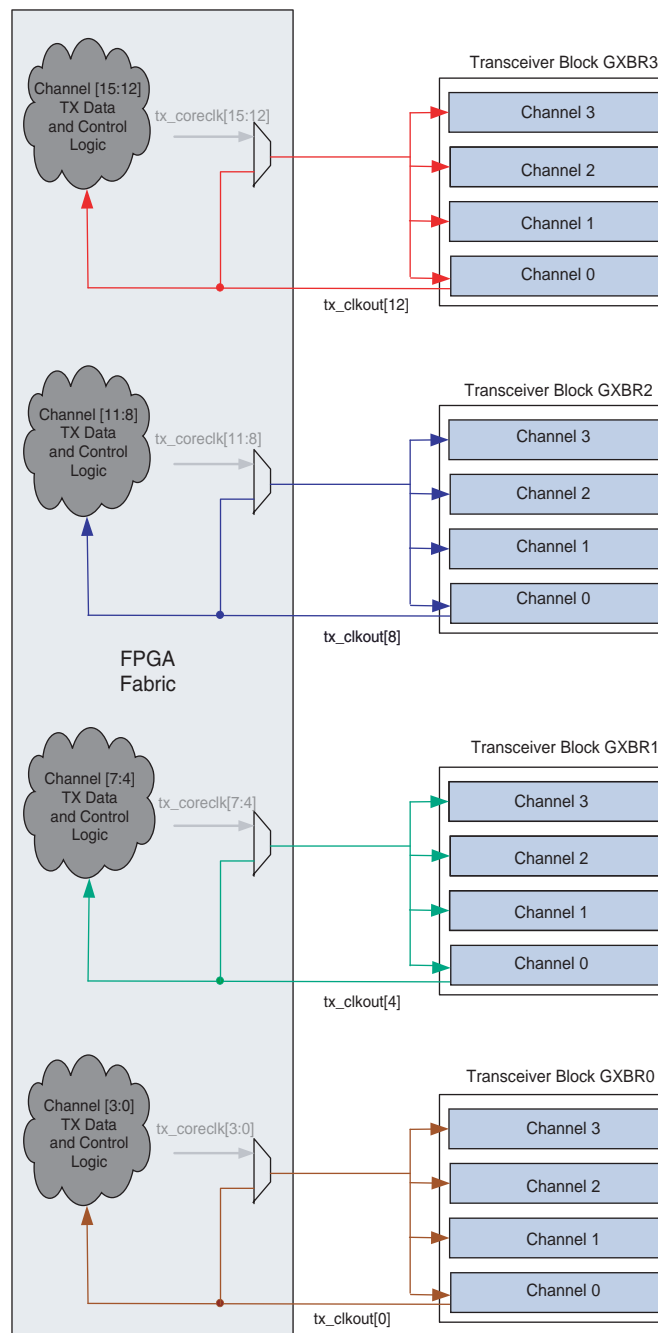
The Quartus II software uses a single `tx_clkout` signal to clock the transmitter phase compensation FIFO write port of all identical channels within a transceiver block. This results in one global and/or regional clock resource being used for each group of identical channels within a transceiver block.

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 usage.

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 (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 2-58.

User-Selected Transmitter Phase Compensation FIFO Write Clock

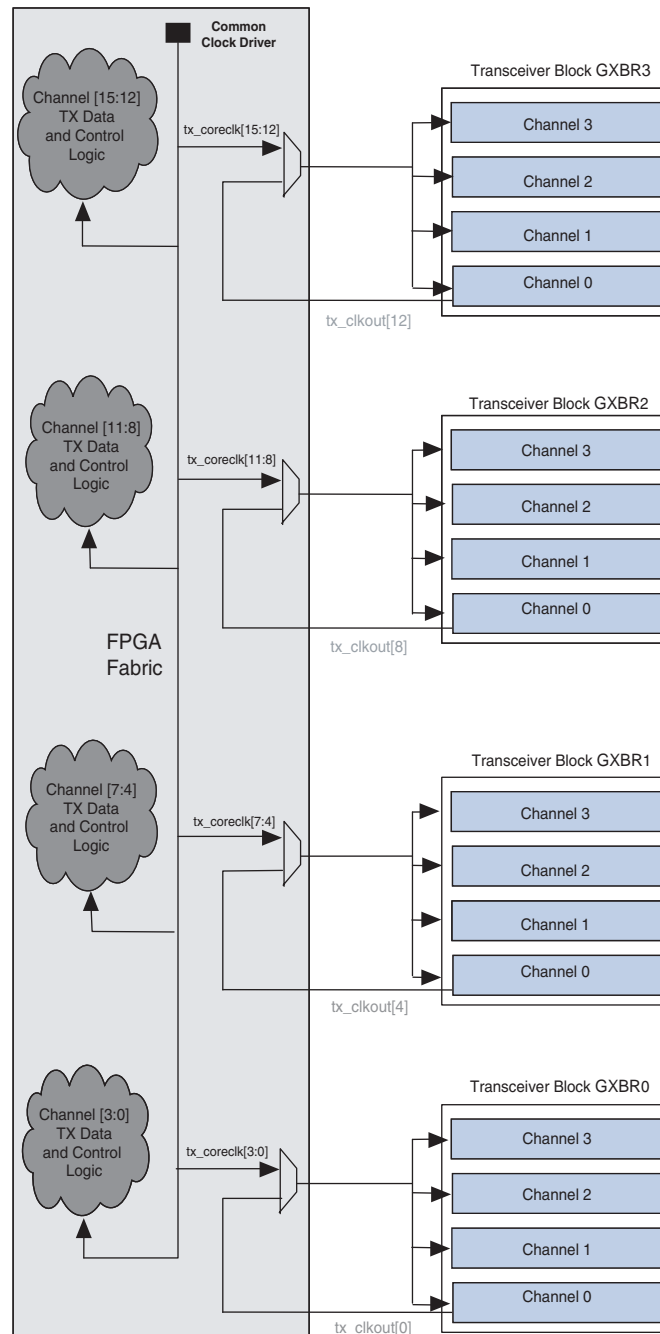
The ALTGX MegaWizard Plug-In Manager provides an optional port named `tx_coreclk` for each instantiated transmitter channel. If you enable this port, the Quartus II software does not automatically select the transmitter phase compensation FIFO write clock source. Instead, the signal that you drive on 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 usage. 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 (<code>tx_clkout</code>)	Divide-by-two version of the parallel transmitter PCS clock from the local clock divider in the associated channel (<code>tx_clkout</code>)
×4 Bonded Channel Configuration	Low-speed parallel clock from the <code>CMU0</code> clock divider of the associated transceiver block (<code>coreclkout</code>)	Divide-by-two version of the low-speed parallel clock from the <code>CMU0</code> clock divider of the associated transceiver block (<code>coreclkout</code>)
×8 Bonded Channel Configuration	Low-speed parallel clock from the <code>CMU0</code> clock divider of the master transceiver block (<code>coreclkout</code> from master transceiver block)	Divide-by-two version of the low-speed parallel clock from the <code>CMU0</code> clock divider of the master transceiver block (<code>coreclkout</code> 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	<p>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):</p> <ul style="list-style-type: none"> ■ 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 2-75.

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) xN 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) xN 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 2-62
- [“User-Selected Receiver Phase Compensation FIFO Read Clock”](#) on page 2-68



User-selection is provided to share transceiver datapath interface clocks in order to reduce the global, regional, and periphery clock resource usage 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 that do not use a rate matcher block (refer to “Non-Bonded Receiver Clocking Without Rate Matcher” on page 2-39)
- Receivers that use a rate matcher block (refer to “Non-Bonded Receiver Clocking with Rate Matcher” on page 2-41)

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

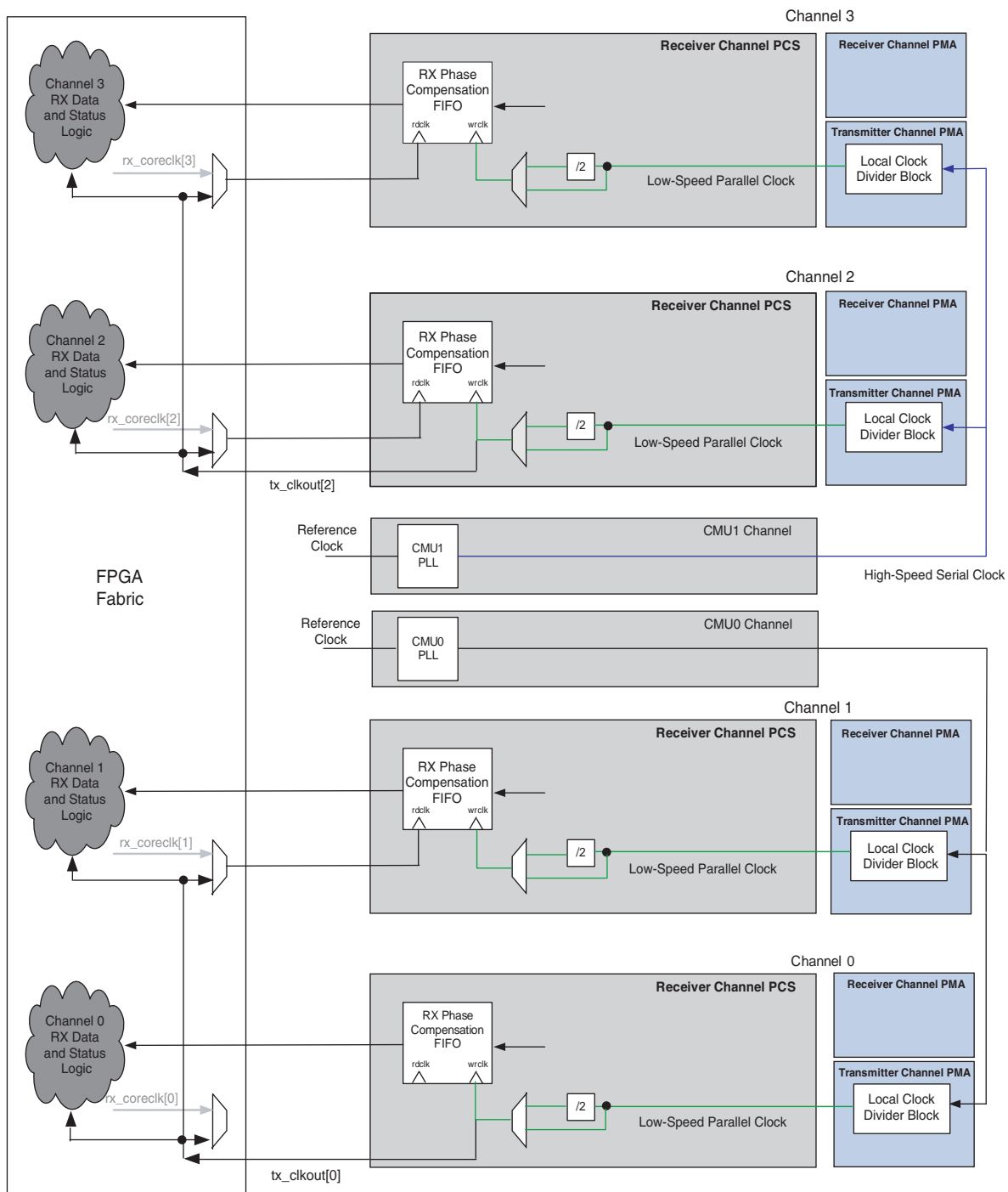
Example 6 assumes channels 0 and 1, driven by the `CMU0` PLL in a transceiver block, are identical. Also, channels 2 and 3, driven by the `CMU1` PLL in the same transceiver block, are identical. In this case, the Quartus II software automatically drives the read port of the receiver phase compensation 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 (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

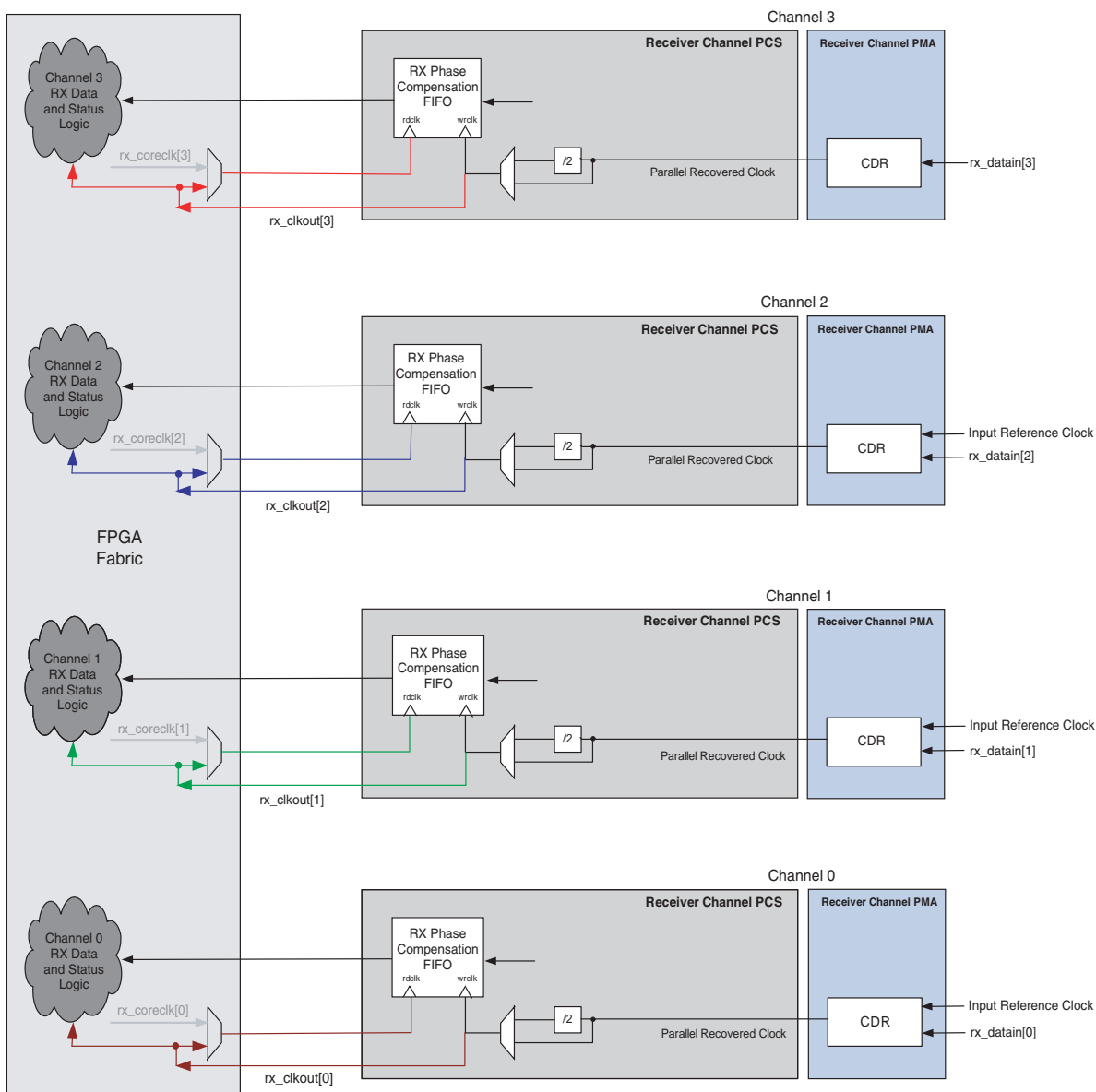
In non-bonded channel configuration without rate matcher, the Quartus II software cannot determine if the incoming serial data in all channels have a 0 PPM frequency difference. The Quartus II software automatically drives the read port of the receiver phase compensation FIFO in each channel with the 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 (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]`.

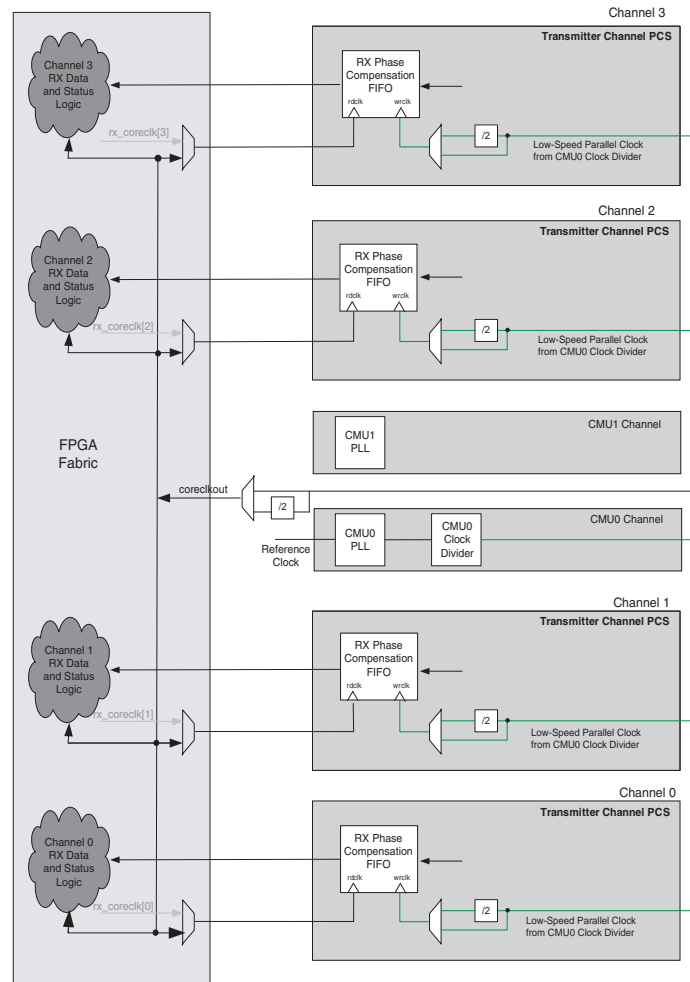
Bonded Channel Configuration

All bonded transceiver channel configurations have rate matcher in the receiver data path. In $\times 4$ and $\times 8$ 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 $\times 8$ 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 $\times 4$ bonded channel configuration.

Figure 2-35. FPGA Fabric-Receiver Interface Clocking in a $\times 4$ Bonded Channel Configuration ⁽¹⁾



Note to Figure 2-35:

(1) The green lines represent low-speed parallel clock from the `CMU0` clock divider.

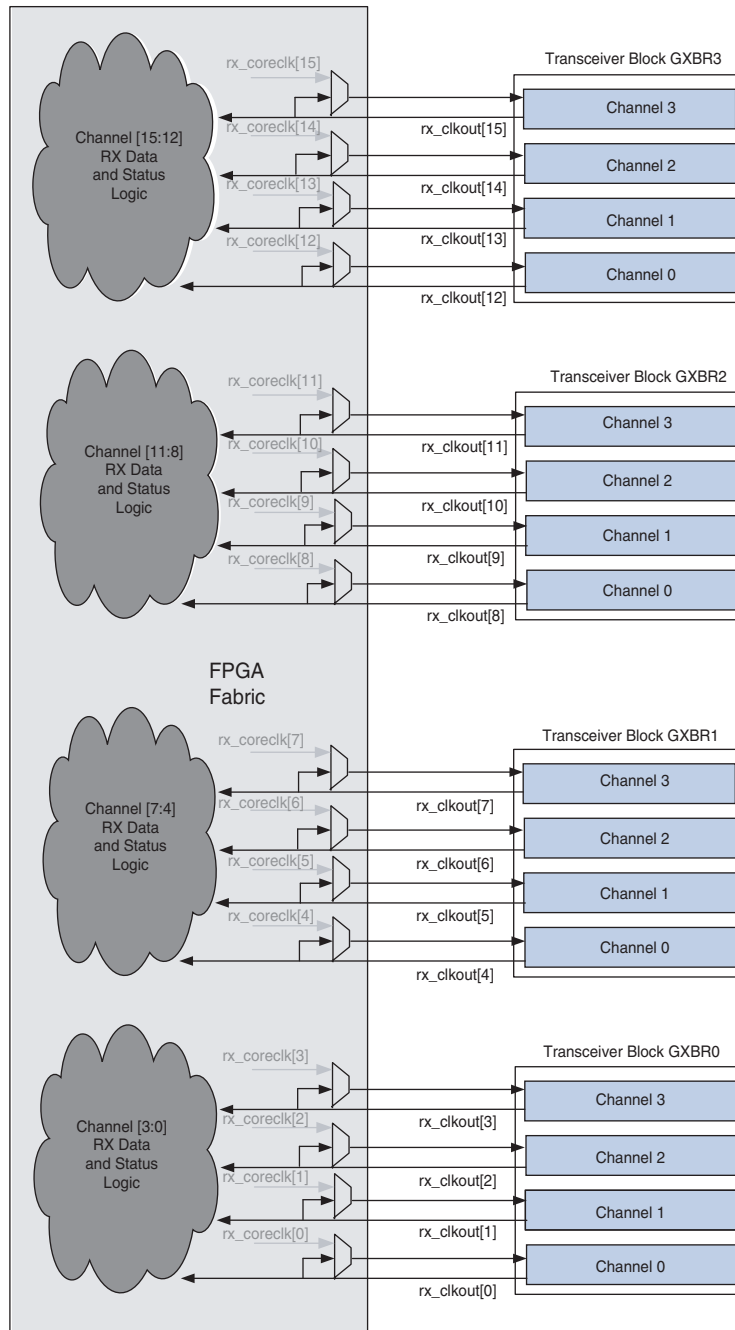
Limitations of the Quartus II Software-Selected Receiver Phase Compensation FIFO Read Clock

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 2-68.

User-Selected Receiver Phase Compensation FIFO Read Clock

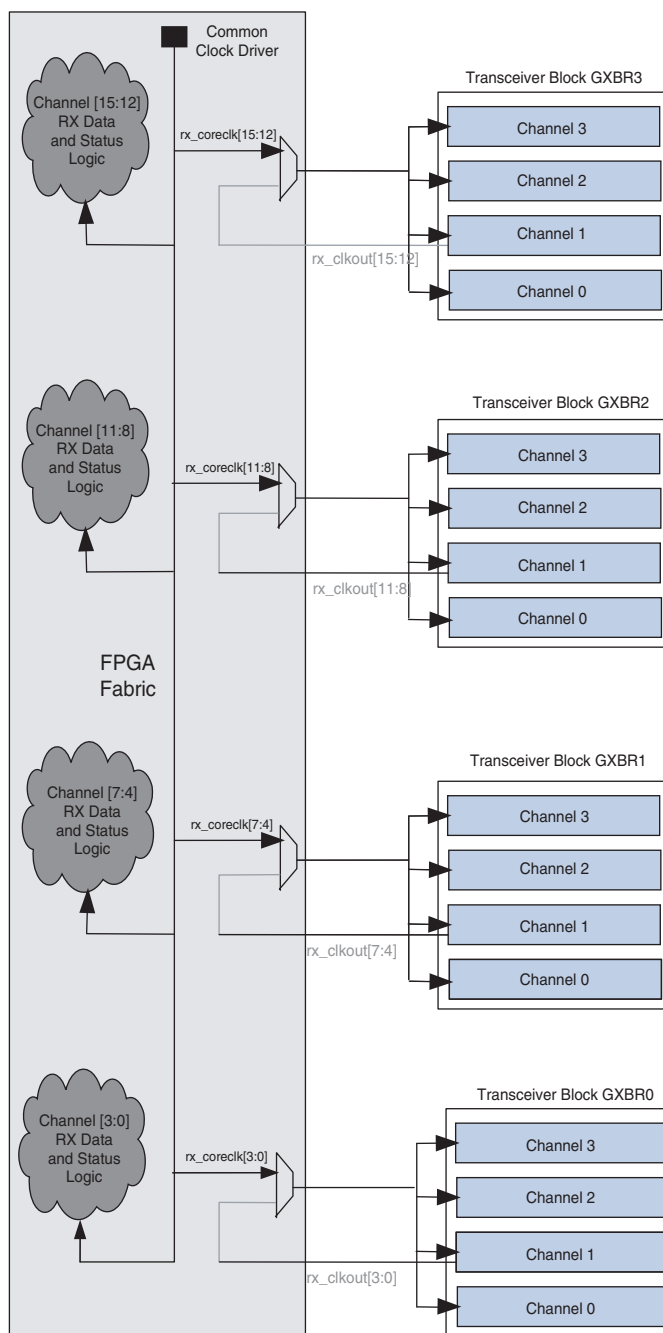
The ALTGX MegaWizard Plug-In Manager provides an optional port named `rx_coreclk` for each instantiated receiver channel. If you enable this port, the Quartus II software does not automatically select the receiver phase compensation FIFO read clock source. Instead, the signal that you drive on the `rx_coreclk` 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 usage. You can connect the `rx_coreclk` ports of all the 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 user assignment 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** user assignment 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 the 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	<p>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):</p> <ul style="list-style-type: none"> ■ 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 2–76.

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. To create the ALTGX instance, follow these steps:

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.

$$f_{\text{out}} = \frac{\text{data rate}}{\text{channel width}}$$

5. You can select the PLL bandwidth by choosing **Tx PLL bandwidth mode**.
6. You can instantiate the `p11_locked` port to indicate the PLL lock status.
7. You can instantiate `p11_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 2-73
- “Configuration Example 2: Configuring Sixteen Identical Channels Across Four Transceiver Blocks” on page 2-75
- “Configuration Example 3: Configuring Sixteen Channels Across Four Transceiver Blocks” on page 2-76
- “Configuration Example 4: Configuring Left and Right, Left, or Right PLL in VCO Bypass Mode” on page 2-78

Configuration Example 1: Configuring 24 Channels in Basic (PMA Direct) $\times N$ Mode in the EP4S100G5F45 Device

Each transceiver block has four regular channels and two CMU channels that you can configure in Basic (PMA Direct) $\times N$ 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) $\times N$ mode.

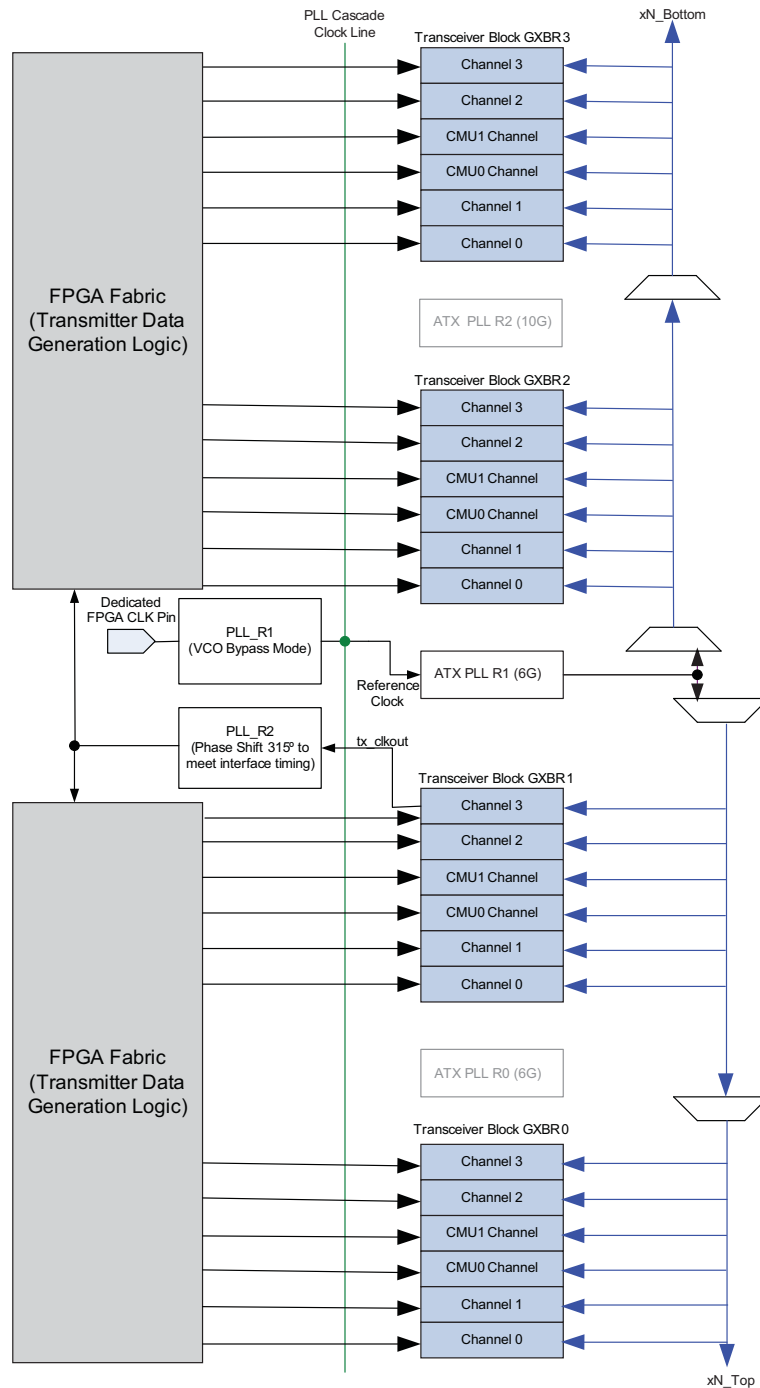
When all 24 channels on one side of the device are configured in Basic (PMA Direct) $\times N$ mode, all eight CMU channels (two in each transceiver block) are configured as PMA-Only channels.

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 [“Configuration Example 4: Configuring Left and Right, Left, or Right PLL in VCO Bypass Mode”](#) on page 2-78.

[Figure 2-38](#) shows 24 channels on the right side of the EP4S100G5F45 device configured in Basic (PMA Direct) $\times N$ 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) $\times N$ 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 through the `$\times N$ _Top` and `$\times N$ _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) xN Mode for Configuration Example 1 (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 2-68.

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

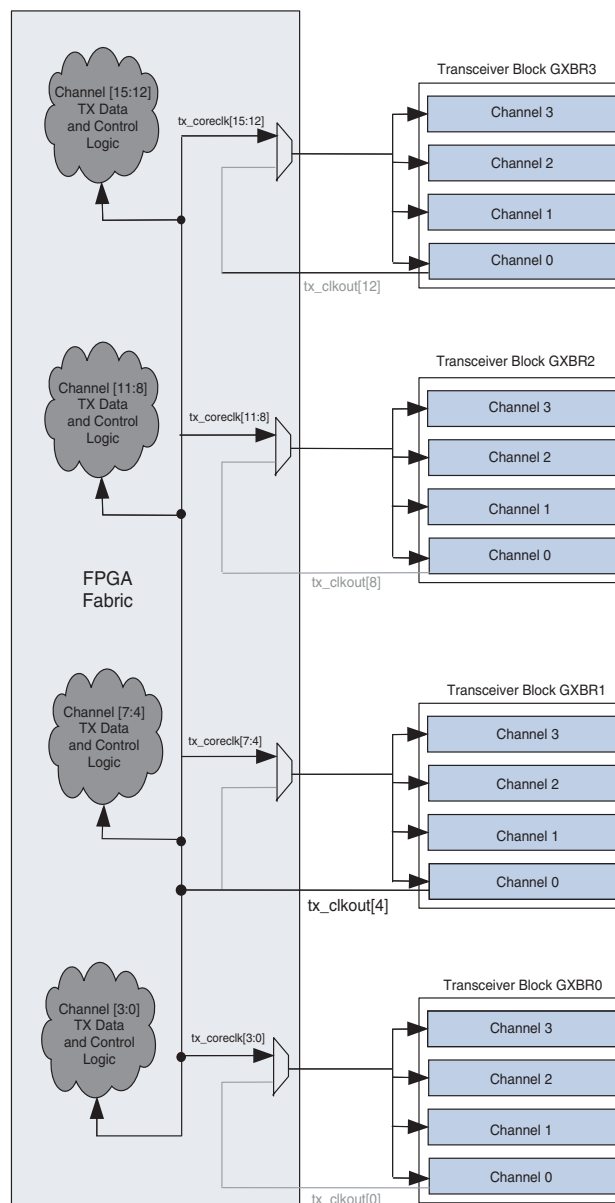


Table 2-19 lists the Quartus II assignments that you must make for the clocking scheme shown in Figure 2-38.

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 2-68.

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

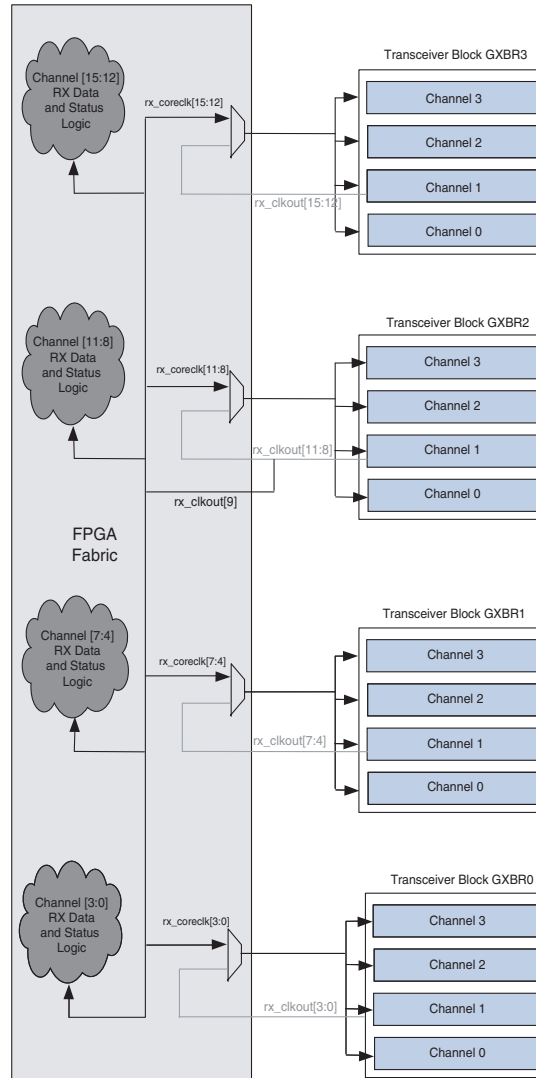


Table 2–20 lists the Quartus II assignments that you must make for the clocking scheme shown in Figure 2–40.


Table 2–20. Quartus II Assignments for Appendix Example 4

From	top_level/top_xcivr_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.

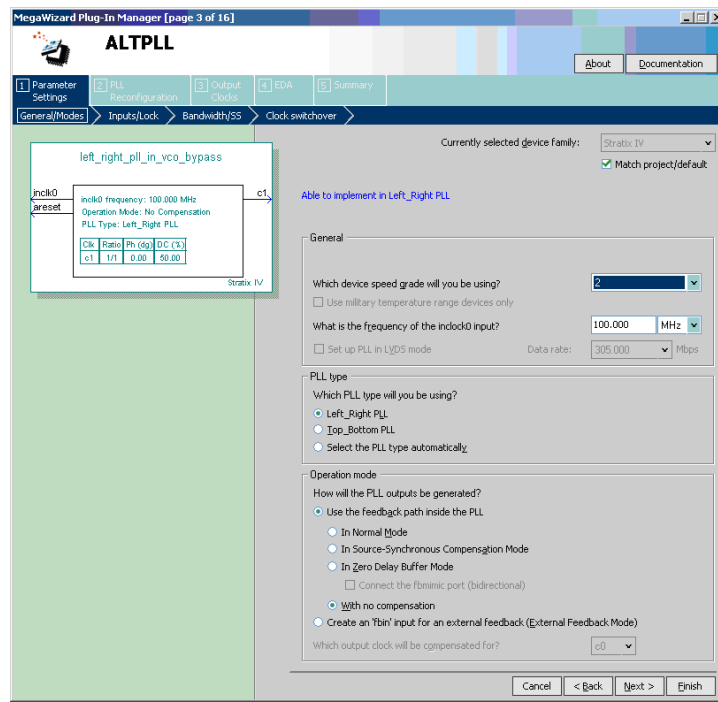
Configuration Example 4: Configuring Left and Right, Left, or Right PLL in VCO Bypass Mode

 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, follow these steps:

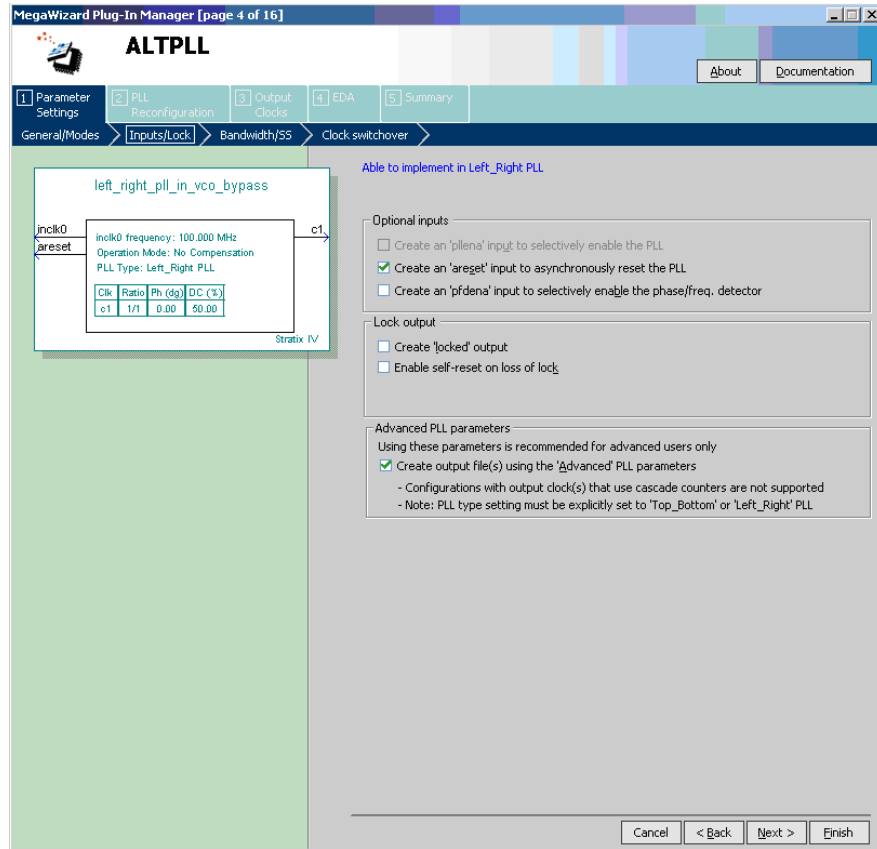
1. Under the **General/Modes** tab, enter the desired input reference clock frequency.
 - a. Under **PLL Type**, select **Left_Right_PLL**.
 - b. Under **Operation mode**, select the **With no compensation** option (Figure 2–41).

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



- Under the **Output Clocks** tab turn off **Use this clock** for clk c0.
- Turn on **Use this clock** for clk c1 (Figure 2-43).


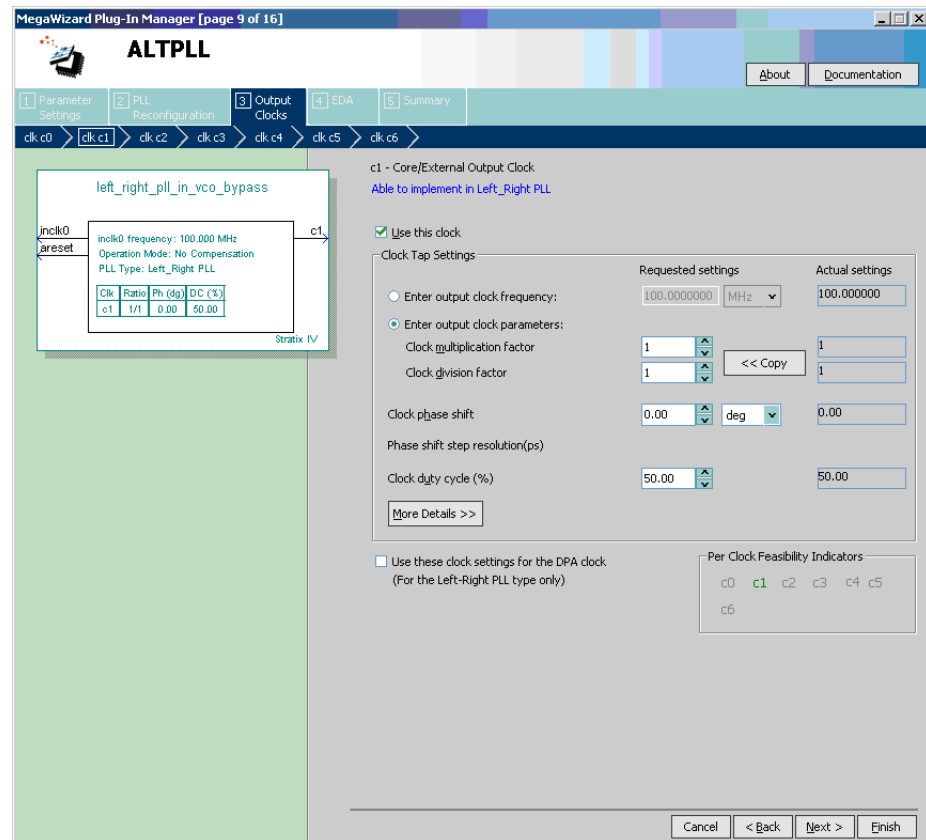
 The VCO bypass option is only enabled for clock output c1.


Figure 2-43. Use This Clock Option Used for Configuration Example 4



- 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 **.vhd1**) and type the following command:

```
qmegawiz -silent -wiz_override="c1_test_source=1,c1_mode=BYPASS,clk1_counter=C1" pll0.v
```

This command places your ALTPLL instance in VCO bypass mode. Revisit the **.v** or **.vhd1** file associated with the ALTPLL instance. Examine the file which is automatically updated to incorporate the PLL in a VCO bypass mode.

 VCO bypass mode is not supported in the **.mif** file. Therefore, you can not manually modify the **.mif** file to set the PLL in VCO bypass mode.

- Finally, connect **clk c1** output of the left and right, left, or right PLL to the input reference clock port of the ATX PLL used to generate the transceiver clocks.


Document Revision History

Table 2-21 lists the revision history for this chapter.

Table 2-21. Document Revision History

Date	Version	Changes
September 2012	3.4	<ul style="list-style-type: none"> Updated the “Non-Bonded Channel Configurations” section to close FB #65105.
December 2011	3.3	<ul style="list-style-type: none"> Updated Table 2-2. Updated the “Left and Right, Left, or Right PLL in VCO Bypass Mode” section.
February 2011	3.2	<ul style="list-style-type: none"> Updated Table 2-4. Updated Figure 2-7, Figure 2-18, Figure 2-19, Figure 2-20. Updated the “Configuration Example 4: Configuring Left and Right, Left, or Right PLL in VCO Bypass Mode” section. Applied new template. Updated chapter title. Applied new template.
March 2010	3.1	<ul style="list-style-type: none"> Updated Table 2-4. Updated Figure 2-7, Figure 2-8, Figure 2-16, and Figure 2-21. Updated the “Transceiver Channel Datapath Clocking” and “Configuration Example 3: Configuring Sixteen Channels Across Four Transceiver Blocks” sections. Added a note to the “refclk0 and refclk1 Pins” section. Changed “datapath clocks” to “datapath interface clocks”. Minor text edits.
November 2009	3.0	<ul style="list-style-type: none"> 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	2.2	<ul style="list-style-type: none"> 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	2.1	Minor updates.
November 2008	2.0	Update to chapter.
May 2008	1.0	Initial release.

This chapter describes the procedure for merging; for example, when combining multiple protocols and data rates within a transceiver block. 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 auxiliary transmit (ATX) PLL, refer to the “Transceiver Performance Specifications” section in the *DC and Switching Characteristics for Stratix IV Devices* chapter.

Overview

Each transceiver channel in a Stratix® IV GX and GT 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 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 3-10
- “Combining Receiver Only Channels” on page 3-10
- “Combining Transmitter Channel and Receiver Channel Instances” on page 3-11
- “Combining Transceiver Instances in Multiple Transceiver Blocks” on page 3-13
- “Combining Transceiver Instances Using PLL Cascade Clocks” on page 3-16
- “Combining Channels Configured in Protocol Functional Modes” on page 3-17
- “Combining Transceiver Channels in Basic (PMA Direct) Configurations” on page 3-25

- “Combination Requirements When You Enable Channel Reconfiguration” on page 3-42
- “Combining Transceiver Channels When You Enable the Adaptive Equalization (AEQ) Feature” on page 3-47
- “Combination Requirements for Stratix IV Devices” on page 3-49
- “Summary” on page 3-49

Each transmitter channel has a local divider (/1, /2, or /4) that divides the high-speed 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, $\times 1$ and $\times N$. The term “Basic (PMA Direct)” used in this chapter refers to both $\times 1$ and $\times N$ and to regular/CMU Channels. Any specific reference to $\times 1$ and $\times N$ 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® [PCIe] $\times 1$) or bonded channel configurations that have PCS enabled (for example, Basic $\times 4$ and $\times 8$, XAUI, PCIe $\times 4$ and $\times 8$). Also, any reference to a channel in non-Basic (PMA Direct) mode indicates that the channel is a regular transceiver channel.
Basic (PMA Direct) $\times 1$	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) $\times N$	A transceiver channel set up in this configuration uses the $\times N$ 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 $\times N$.

 For more information about transceiver channel set up using a Basic (PMA Direct) $\times N$ configuration, refer to the *Transceiver Clocking in Stratix IV Devices* 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 an example, refer to “[Combining Transceiver Instances in Multiple Transceiver Blocks](#)” on page 3–13.
- 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 that you 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. The Stratix IV GT device only allows 1.4 V for the V_{CCH} supply. To combine the channel instances within the same transceiver block, the Quartus II software requires that you 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 for Stratix IV Devices](#) chapter.

Transceiver Analog Power ($V_{CCA_L/R}$)

The Stratix IV GX and GT device contains two different power supply pins, V_{CCA_L} and V_{CCA_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 and GT device provides you the option to select 2.5 V or 3.0 V for the $V_{CCA_L/R}$ supply through the ALTGX MegaWizard Plug-In Manager. The Stratix IV GT device only allows 3.3 V for the $V_{CCA_L/R}$ supply. You must set the same $V_{CCA_L/R}$ 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 $V_{CCA_L/R}$ 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 for Stratix IV Devices* 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, and 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 *Dynamic Reconfiguration in Stratix IV Devices* chapter.

Calibration Clock and Power Down

Each calibration block in a Stratix IV GX and GT 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 *Transceiver Architecture in Stratix IV Devices* 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
- `GXB_TX_PLL_Reconfig_Group` assignment (refer to [Table 3-14 on page 3-42](#))
- If the selected functional mode in one instance is (OIF) CEI Phy Interface or PCIe, 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 a **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](#) lists 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 (Gbps)
inst0	1	Receiver and Transmitter	4.25
inst1	1	Receiver and Transmitter	2.125
inst2	1	Receiver and Transmitter	1.0625
inst3	1	Receiver and Transmitter	4.25

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 listed 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 (Gbps)
inst0	What is the effective data rate?	4.25
	Specify base data rate	4.25 ⁽¹⁾
inst1	What is the effective data rate?	2.125
	Specify base data rate	4.25 ⁽¹⁾
inst2	What is the effective data rate?	1.0625
	Specify base data rate	4.25 ⁽¹⁾

Table 3-3. ALTGX MegaWizard Plug-In Manager Settings for Example 1

Instance	General Screen Option	Setting (Gbps)
inst3	What is the effective data rate?	4.25
	Specify base data rate	4.25 ⁽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 the 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 `p11_powerdown` port. You must drive the `p11_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

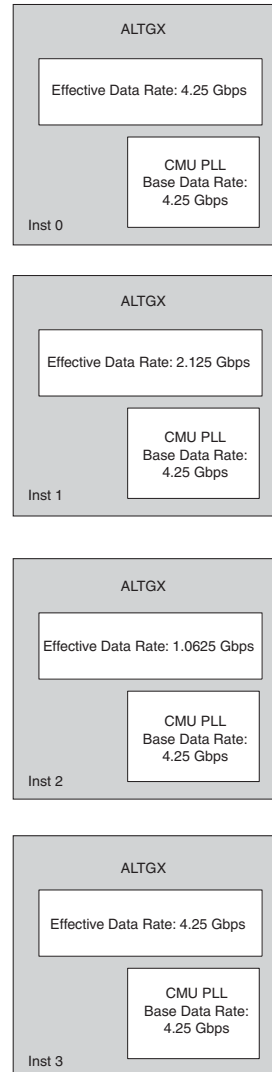
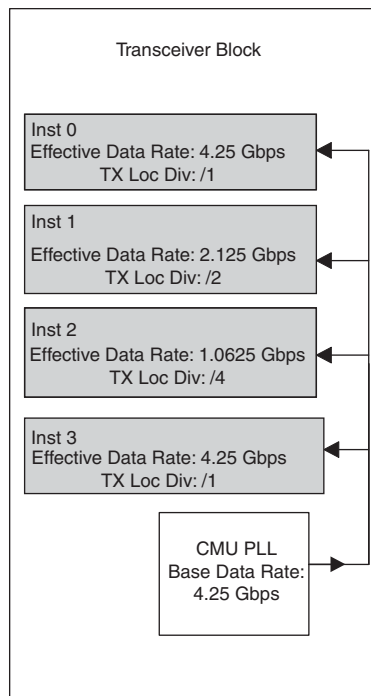


Figure 3-2 shows the scenario after the Quartus II software combines the transceiver channel instances.

Figure 3-2. Combined Instances after Compilation for Example 1




Example 2

Consider the example design listed 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 (Gbps)	Base Data Rate (Gbps)
inst0	1	Receiver and Transmitter	2.5	2.5
inst1	1	Receiver and Transmitter	2.5	5
inst2	1	Receiver and Transmitter	1	1

 Even though the effective data rate of inst1 is 2.5 Gbps ($5 \text{ Gbps} / 2 = 2.5 \text{ Gbps}$), 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 PCIe, 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 $\times N$ clock lines from the ATX PLL and the two instances.


 For more information about $\times N$ clocking, refer to the “Transmitter Channel Data Path Clocking” section in the *Transceiver Clocking in Stratix IV Devices* 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 3-10

Example 3

Consider the example design listed in Table 3-5 with four ALTGX instances.

Table 3-5. Four ALTGX Instances for Example 3

Instance Name	Configuration	Serial Data Rate (Gbps)	Input Reference Clock Frequency (MHz)
inst0	Transmitter only	3.125	156.25
inst1	Receiver only	2.5	156.25
inst2	Transmitter only	1.25	125
inst3	Receiver only	2	125

After you create the above instances, if you force the placement of the instances, as listed 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

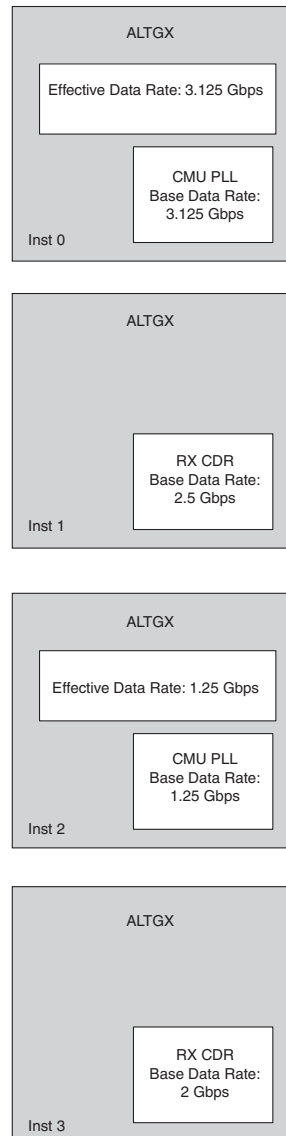
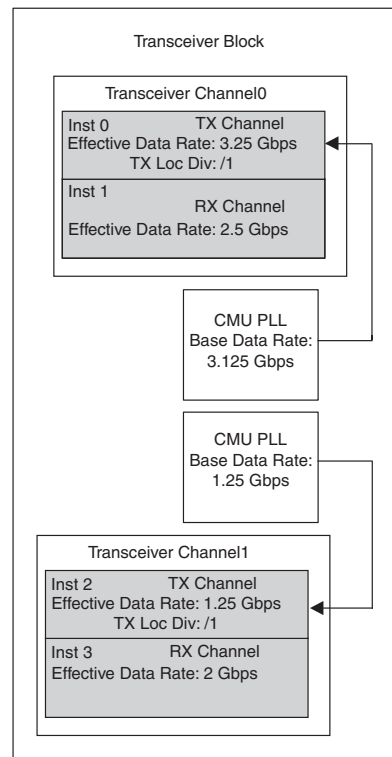


Figure 3-4 shows the transceiver channel instances after compilation.

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 is 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 listed 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 (Gbps)	Input Reference Clock (MHz)
inst0	7	Receiver and Transmitter	4.25	125 from refclk0
inst1	1	Receiver and Transmitter	4.25	125 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

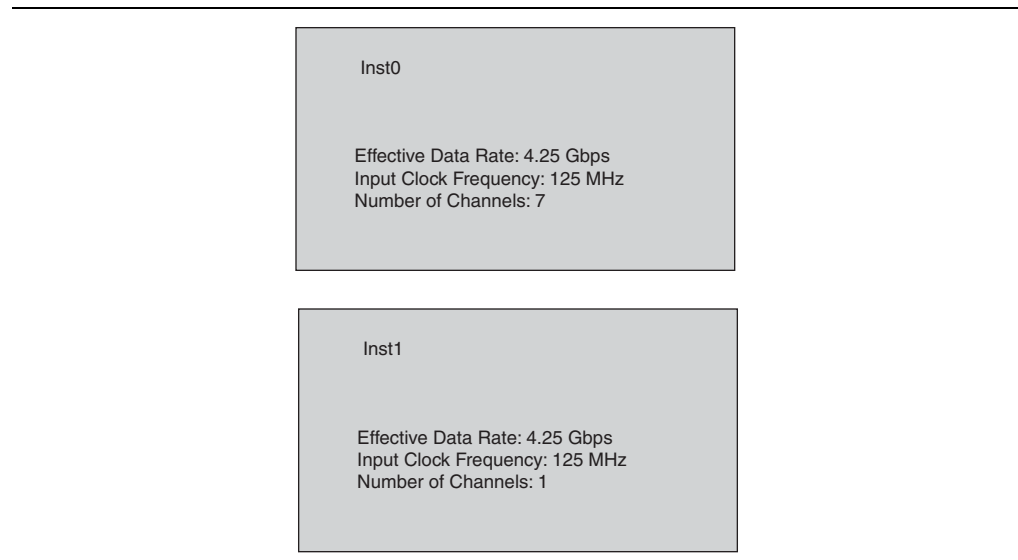
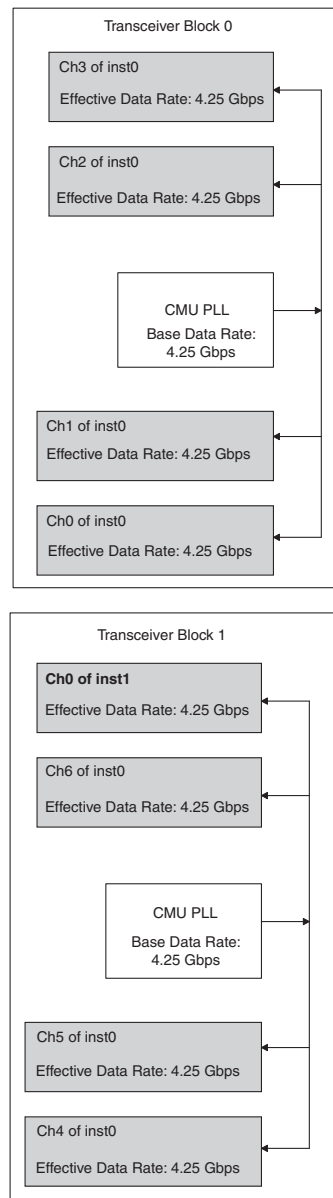



Figure 3-6 shows the transceiver instances after compilation.

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 `p11_inclk` port for `inst0`. In your design, provide only one clock input for the `p11_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 `p11_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_crucclk` 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 and GT 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 and GT 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 [Transceiver Clocking in Stratix IV Devices](#) 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”**—Examples of bonded ×4 mode:
 - Basic mode with the sub protocol set to ×4
 - XAUI
 - PCIe mode with the sub protocol set to Gen1 ×4 or Gen2 ×4.
- **“Bonded ×8 Functional Mode” on page 3-20**—Examples of bonded ×8 mode:
 - Basic mode with the sub protocol ×8
 - PCIe mode with the sub protocol ×8

Bonded ×4 Functional Mode

The combination requirements for Basic ×4, Deterministic Latency ×4, and PCIe ×4 functional modes (if you do not use the PCIe hard IP block) 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 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 *Transceiver Clocking in Stratix IV Devices* chapter.

Because you used the central clock divider, there are two restrictions on the channel combinations:

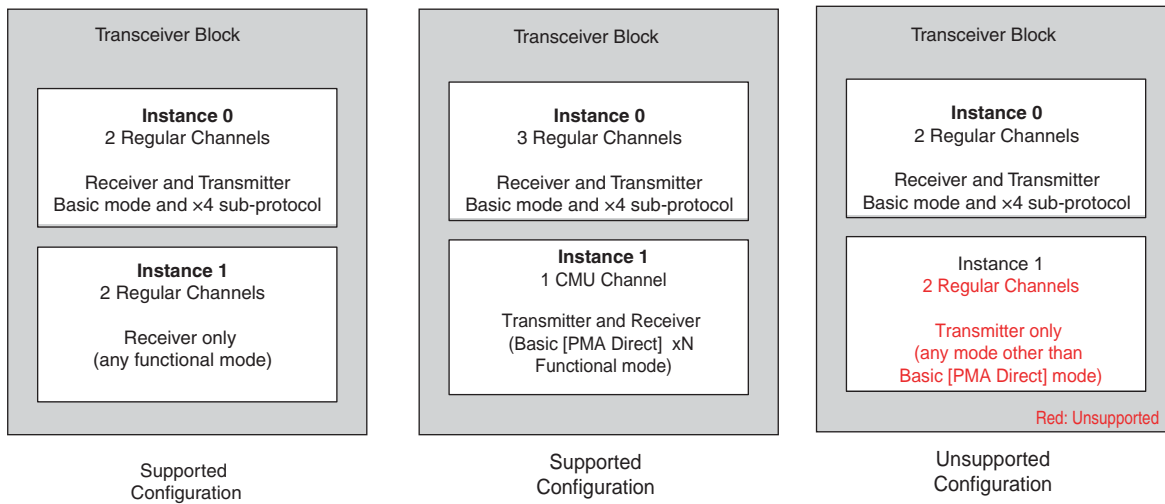
1. If you configure channels in bonded ×4 functional mode, the remaining transmitter channels (regular or CMU channels) within the transceiver block can be used only in Basic (PMA Direct) ×1 or ×N mode.

 If PCIe functional mode uses the PCIe hard IP block, the combination requirements are different. For more information, refer to **“Combining Channels Using the PCIe hard IP Block with Other Channels” on page 3-24**.

The receiver channels are clocked independently. Therefore, you can configure the unused receiver channels within a transceiver block in any allowed configuration.

Figure 3-7 shows examples of supported and unsupported combinations.

Figure 3-7. Examples of Supported and Unsupported Configurations to Combine Instances in Basic $\times 4$ 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 $\times 4$ 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 $\times 4$ functional mode.

Using the remaining channels in Basic (PMA Direct) $\times 1$ or $\times N$ 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) $\times 1$ configuration.
2. If you want to configure the remaining transmitter channels at the same data rate as the bonded $\times 4$ functional mode, you can configure the remaining transmitter channels in Basic (PMA Direct) $\times 1$ 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 $\times 4$ functional mode, you can configure the transmitter side of the CMU0 channel in Basic (PMA Direct) $\times N$ mode in single-width configuration only (double-width configuration is not supported). You can use the CMU1 channel in Basic (PMA Direct) $\times N$ 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 $\times 4$ 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 $\times 4$ 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.


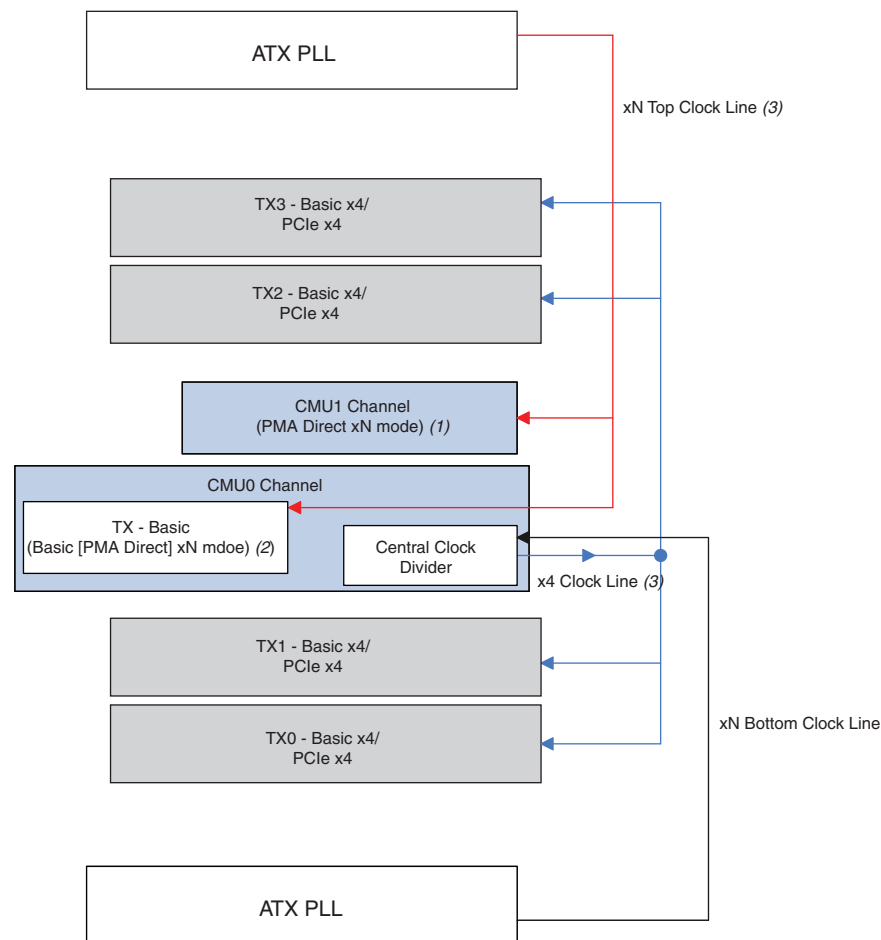
 For XAUI, the option to select ATX PLL is not available.

Figure 3-8 shows the combination of Basic/PCIe $\times 4$ functional mode with Basic (PMA Direct) $\times N$ mode within the same transceiver block.

Figure 3-8. Basic $\times 4$ Functional Mode Configuration when Combining Channels (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 xN top clock line, the blue lines represent the $4x$ clock line, and the black line represents the xN bottom clock line.
- (4) To simplify the illustration, only the transmitter side is shown. PCIe $\times 4$ refers to PCIe with the sub protocol set to **Gen1 $x4$** and **Gen2 $x4$** .


Bonded x8 Functional Mode

Bonded x8 functional mode is similar to bonded x4 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 x8 configuration. Therefore, when you 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 x8 configuration.
-  The master is the adjacent lower transceiver block. For more information about location requirements, refer to the “Bonded Channel Configurations” section of the *Transceiver Clocking in Stratix IV Devices* chapter.

In Basic x8 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) x1 or xN mode. In PCIe Gen1 x8 and Gen2 x8 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) x1 or xN mode depends on whether the x8 functional mode uses CMU PLL or ATX PLL, as described below.

-  If PCIe functional mode uses the PCIe hard IP block, the combination requirements are different. For more information, refer to “Combining Channels Using the PCIe hard IP Block with Other Channels” on page 3-24.


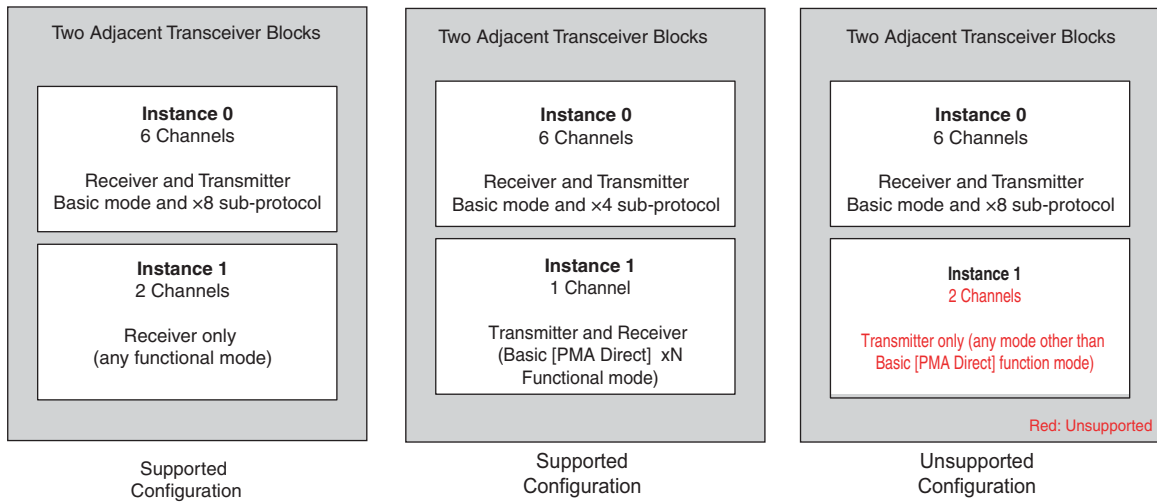
 Each receiver channel configured in Basic $\times 8$ functional mode is clocked independently by the recovered clock from its receiver CDR. You can use the available receiver channels in any configuration. Figure 3-9 shows examples of supported and unsupported configuration in Basic $\times 8$ mode.

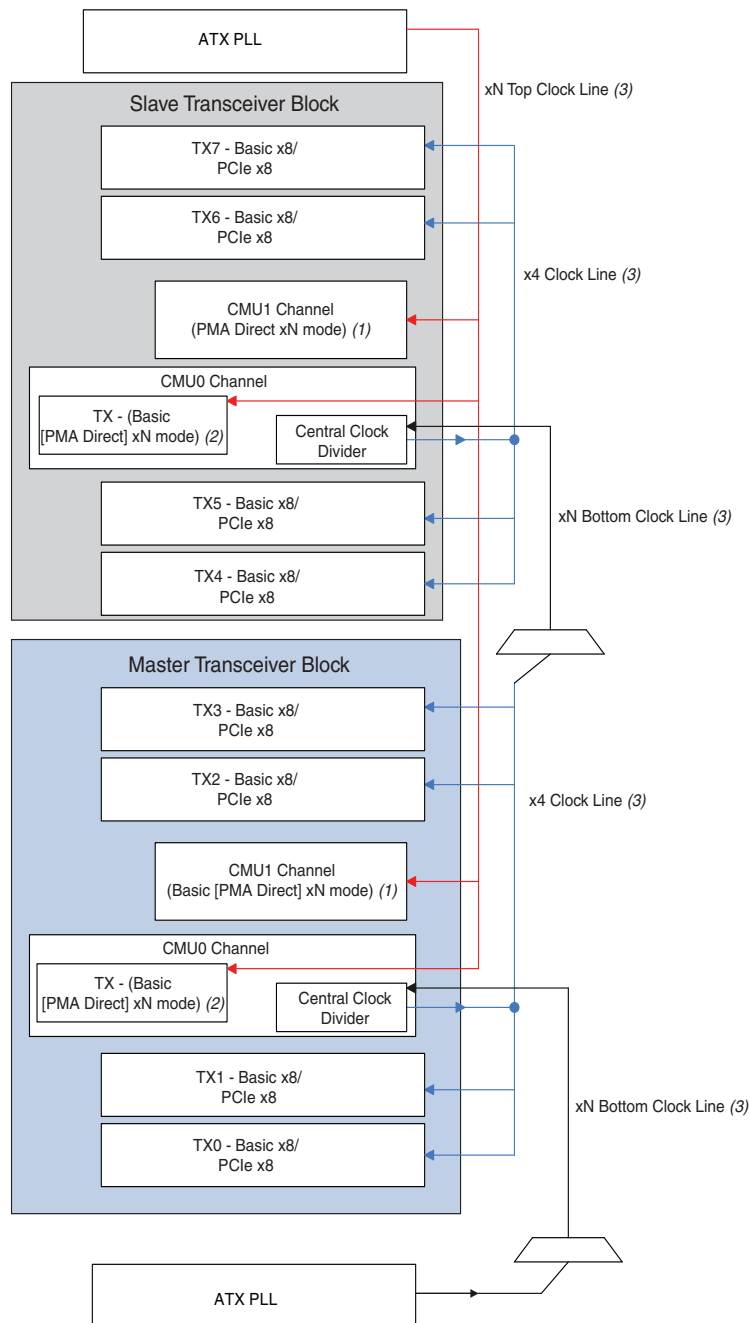
Figure 3-9. Examples of Supported and Unsupported Configurations to Combine Instances in Basic $\times 8$ Mode



When the eight regular channels are used up in bonded $\times 8$ functional mode:

- If the ATX PLL is used to generate clocks for the $\times 8$ 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) $\times N$ mode. Within Basic (PMA Direct) $\times N$ 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) $\times N$ mode in single-width or double-width configuration.

Figure 3-10. Basic x8/PCIe x8 Functional Mode Configuration when Combining Channels (ATX PLL) (4)

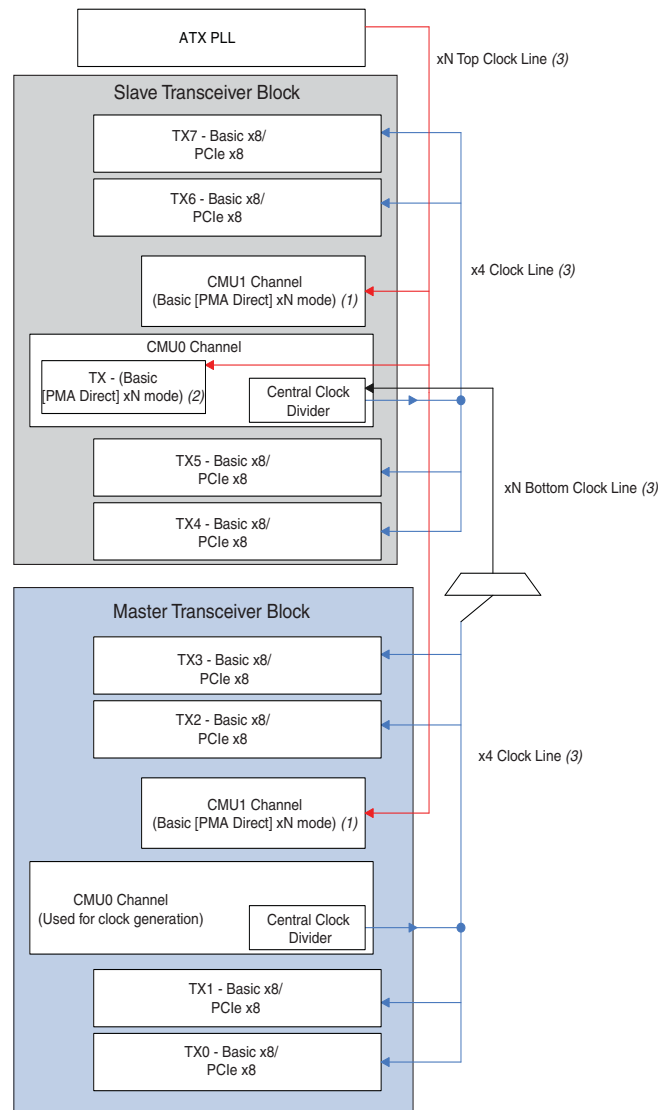
**Notes to Figure 3-10:**

- (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 xN top clock line, the blue lines represent the x4 clock line, and the black line represents the xN bottom clock line.
- (4) To simplify the illustration, only the transmitter side is shown. PCIe x8 refers to PCIe with the sub protocol set to **Gen1 x8** and **Gen2 x8**.

- If the CMU PLL is used to generate clocks for the $\times 8$ bonded functional mode, you can use the CMU0 channel in the slave transceiver block only in Basic (PMA Direct) $\times N$ mode in the single-width configuration. You can use the CMU1 channels in both the master and slave transceiver blocks in Basic (PMA Direct) $\times N$ mode in single-width or double-width configuration.

Figure 3–11 shows the Basic $\times 8$ functional mode configuration for these combination restrictions when using CMU PLL.

Figure 3–11. Basic $\times 8$ /PCIe $\times 8$ Functional Mode Configuration when Combining Channels (CMU PLL) (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 $\times N$ top clock line, the blue lines represent the $\times 4$ clock line, and the black line represents the $\times N$ bottom clock line.
- (4) To simplify the illustration, only the transmitter side is shown. PCIe $\times 8$ refers to PCIe with the sub protocol set to **Gen1 $\times 8$** and **Gen2 $\times 8$** .

Combining Channels Configured in Deterministic Latency Mode

The ALTGX MegaWizard Plug-In Manager provides Deterministic Latency mode with two variations ($\times 1$ and $\times 4$) 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 $\times 1$, the low-speed parallel clock from the transmitter serializer is fed back to the PFD input of the CMU PLL; for $\times 4$, the output of the low-speed parallel clock from the central clock divider is provided as feed back.

For the $\times 1$ variation, one CMU PLL is required for each transmitter channel in the instance. As a result, in $\times 1$ mode, you can configure only two channels within the transceiver block in this mode. The restrictions for Deterministic Latency mode in $\times 4$ mode are the same as that of the bonded $\times 4$ functional mode. For more information, refer to [“Bonded \$\times 4\$ Functional Mode”](#) on page 3-17.

Combining Channels Using the PCIe hard IP Block with Other Channels

The Stratix IV GX and GT device contains an embedded PCIe hard IP block that performs the phyMAC, datalink, and transaction layer functionality specified by PCIe base specification 2.0. Each PCIe hard IP block is shared by two transceiver blocks. The **PCI Express Compiler Wizard** provides you the options to configure the PCIe 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 PCIe hard IP block. The restrictions depend on the number of channels used ($\times 1$ or $\times 4$) and the number of virtual channels (VCs) selected in the PCI Express Compiler MegaWizard Plug-In Manager. [Table 3-8](#) lists the restrictions.



When you use the PCIe 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. PCIe Hard IP Block Restrictions When Combining Transceiver Channels with Different Functional and/or Protocol Modes (Part 1 of 2) ^{(1), (2), (7)}

PCIe Configuration (PCIe hard IP Options Enabled in the PCI Express Compiler MegaWizard Plug-In Manager) ⁽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
$\times 1$	64-bit	1	PCIe $\times 1$	Avail.	Avail.	Avail.	Avail.	Avail.	Avail.	Avail.
		2	PCIe $\times 1$	—	—	—	Avail.	Avail.	Avail.	Avail.
$\times 4$	64-bit	1	PCIe $\times 4$				Avail.	Avail.	Avail.	Avail.
		2	PCIe $\times 4$				Avail.	Avail.	Avail.	Avail.

Table 3–8. PCIe Hard IP Block Restrictions When Combining Transceiver Channels with Different Functional and/or Protocol Modes (Part 2 of 2) ^{(1), (2), (7)}

PCIe Configuration (PCIe hard IP Options Enabled in the PCI Express Compiler MegaWizard Plug-In Manager) ⁽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
x4	128-bit	1	PCIe x4				Avail.	Avail.	Avail.	Avail.
		2	PCIe x4				—	—	Avail.	Avail.
x8	—	—	PCIe x8							

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 PCIe x4 or x8 configuration.
- (5) Transceiver block 1—the adjacent transceiver block that shares the same PCIe 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 PCIe functional mode, refer to the “x8 Channel Configuration” section in the *Transceiver Clocking in Stratix IV Devices* chapter.
- (7) When you use the PCIe hard IP Block, you cannot configure the CMU channels within the transceiver block as transceiver channels.

 For more information about the *PCI Express Compiler MegaCore functions* and hard IP implementation, refer to the *PCI Express Compiler User Guide*.


If you configure a transceiver channel in PCIe 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) x1 or xN mode.

Combining Transceiver Channels in Basic (PMA Direct) Configurations

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 for Stratix IV Devices* 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 *Transceiver Clocking in Stratix IV Devices* 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 the 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) ×1 in single-width or double-width configuration, as shown in [Figure 3-17 on page 3-32](#).

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 3-26](#)
- [“One Instance in a Basic \(PMA Direct\) ×1 Configuration with Multiple Transceiver Channels” on page 3-26](#)
- [“Combining Multiple Instances of Transmitter Only and Receiver Only Configurations in Basic \(PMA Direct\) ×1 Mode” on page 3-29](#)
- [“Combining Channels Configured in Basic \(PMA Direct\) ×1 with Non-Basic \(PMA Direct\) Modes” on page 3-29](#)

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 a 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 provide the 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 `p11_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 `p11_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

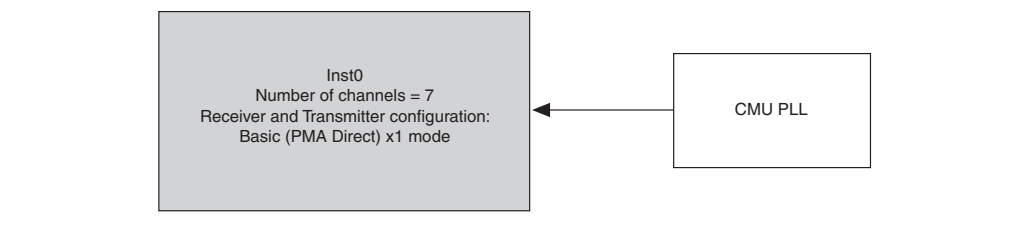
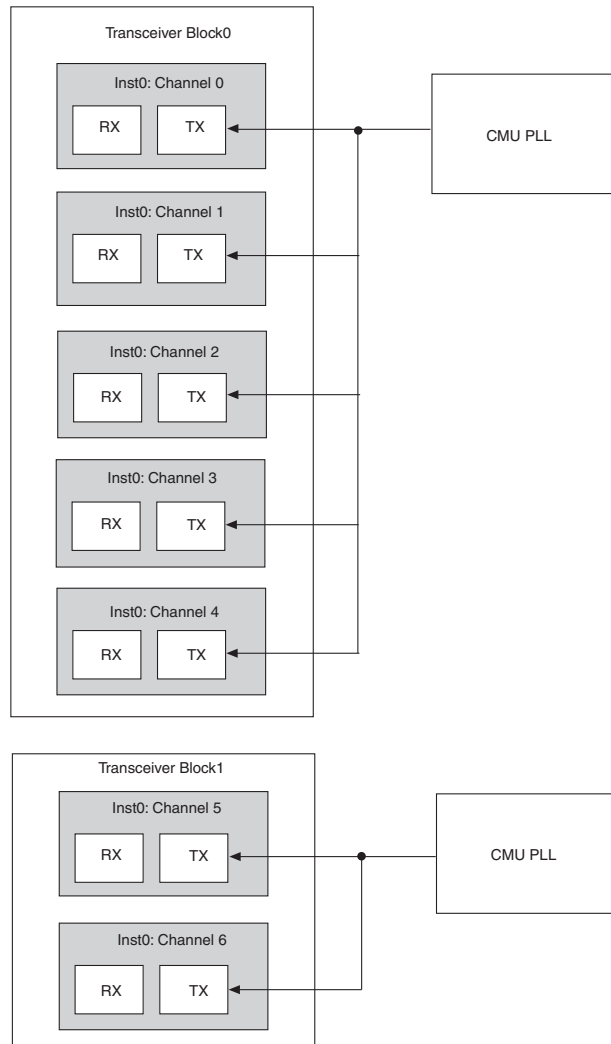


Figure 3-13 shows the conditions after compilation. In this example, the `gxb_powerdown` and `p11_powerdown` ports for channels 0 to 4 and channels 5 and 6 are driven from the same logic.

Figure 3-13. Combined Channels After Compilation for Example 5



If you connect each of the seven bits of the `gxb_powerdown` and `p11_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 Transmitter Only and Receiver Only Configurations in Basic (PMA Direct) ×1 Mode

The Quartus II software allows you to combine instances of **Transmitter Only** and **Receiver Only** configurations in Basic (PMA Direct) ×1 mode.

You can also combine **Transmitter Only** instances in non-Basic (PMA Direct) ×1 configuration (non-bonded only) and the **Receiver 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 3-11](#).

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, you cannot enable the **channel reconfiguration** option in the instance setup in non-Basic (PMA Direct) configuration.

Example 6

Consider the example design listed 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 (Gbps)	Transmitter Channel Effective Data Rate (Gbps)	Input Reference Clock Frequency (MHz)
inst0	GIGE	1.25	1.25	125 (assume refclk0)
inst1	Basic (PMA Direct) ×1 (four channels)	1.25	1.25	Same as inst0

Figure 3-14 shows Basic (PMA Direct) $\times 1$ and non-Basic (PMA Direct) configurations before compilation.

Figure 3-14. Logical View of the Instances in Basic (PMA Direct) $\times 1$ and Non-Basic (PMA Direct) Configurations Before Compilation for Example 6

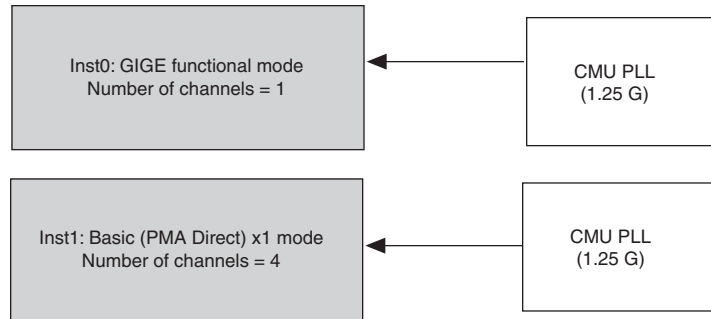
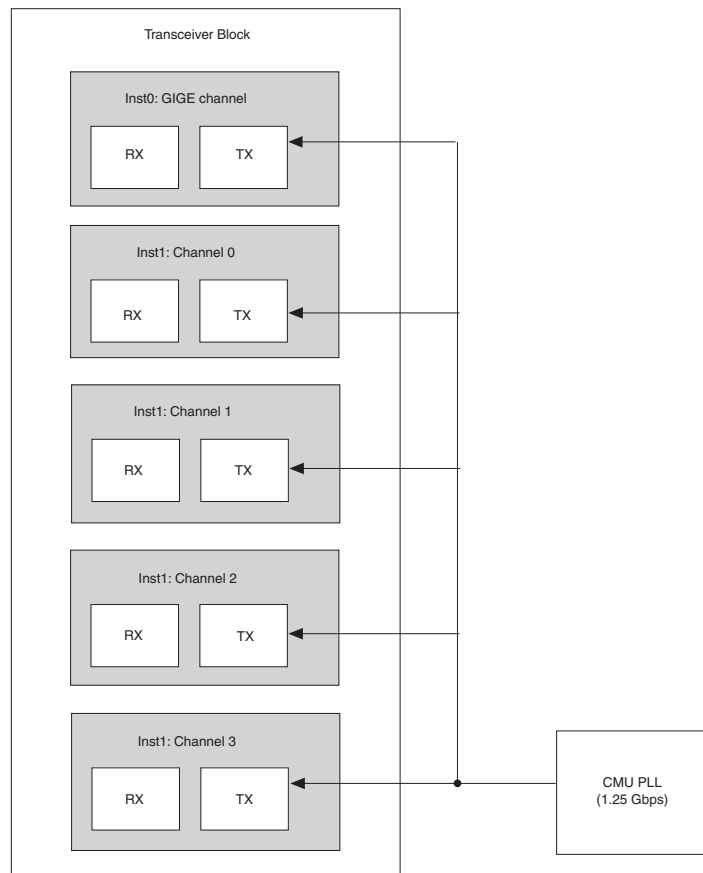


Figure 3-15 shows Basic (PMA Direct) $\times 1$ and non-Basic (PMA Direct) configurations after compilation.

Figure 3-15. Combining Basic (PMA-Direct) $\times 1$ and Non-Basic (PMA Direct) Configurations After Compilation for Example 6



Example 7

Consider the example design listed 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 (Gbps)	Transmitter Channel Effective Data Rate (Gbps)	Input Reference Clock Frequency (MHz)
inst0	GIGE	1.25	1.25	125 (assume refclk0)
inst1	Basic (PMA Direct) ×1 (three channels in Receiver and Transmitter configuration)	2	2	refclk1
inst2	Basic (PMA Direct) ×1 (one channel in Transmitter Only configuration)	2	2	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

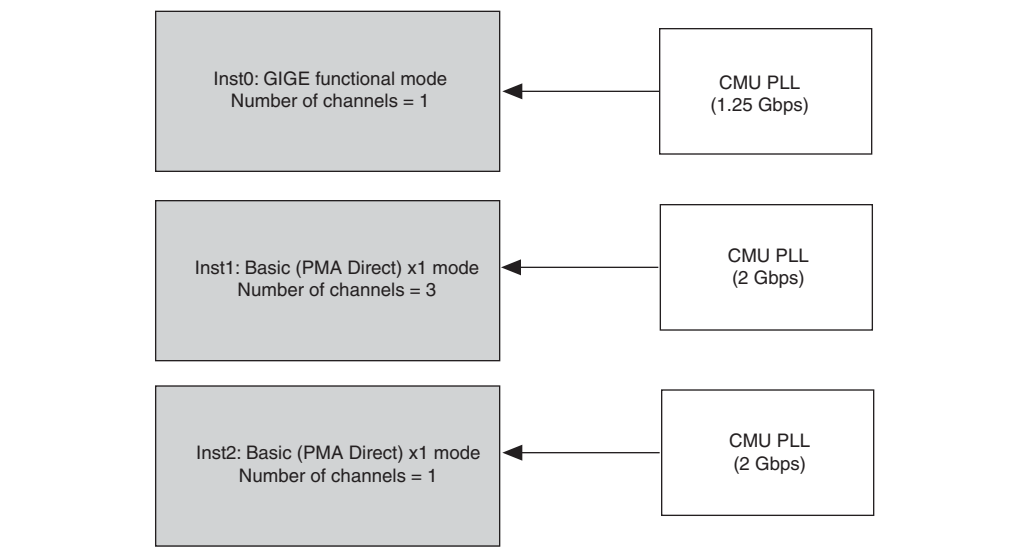
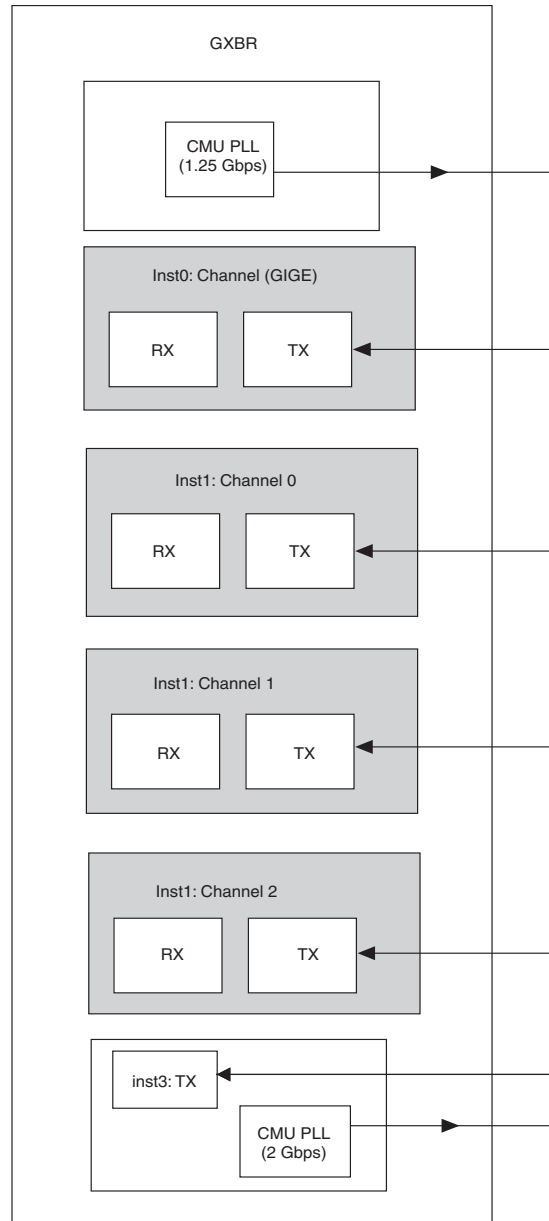


Figure 3-17 shows Basic (PMA Direct) $\times 1$ and non-Basic (PMA Direct) configurations after compilation.

Figure 3-17. Combining Basic (PMA Direct) $\times 1$ and Non-Basic (PMA Direct) Instances in a Transceiver Block After Compilation for Example 7



Key Observations



To combine 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) $\times N$ Configurations

When you configure a transceiver channel in Basic (PMA Direct) $\times N$ configuration, you can enable the Quartus II software to use the $\times N$ lines to provide clocks to the transmitter channels, as shown in [Figure 3-18](#).

The following are the possible sources driving the $\times N$ clock lines:

- The CMU0 central divider within the CMU0 channel. Only the CMU0 clock divider block can drive the $\times N$ 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 *Transceiver Architecture in Stratix IV Devices* chapter.
- The ATX PLL block.

Channel Placement in a Basic (PMA Direct) $\times N$ Mode Instance

If you compile a design with a transceiver instance configured in Basic (PMA Direct) $\times N$ mode, the Quartus II software, by default, places these channels contiguously.

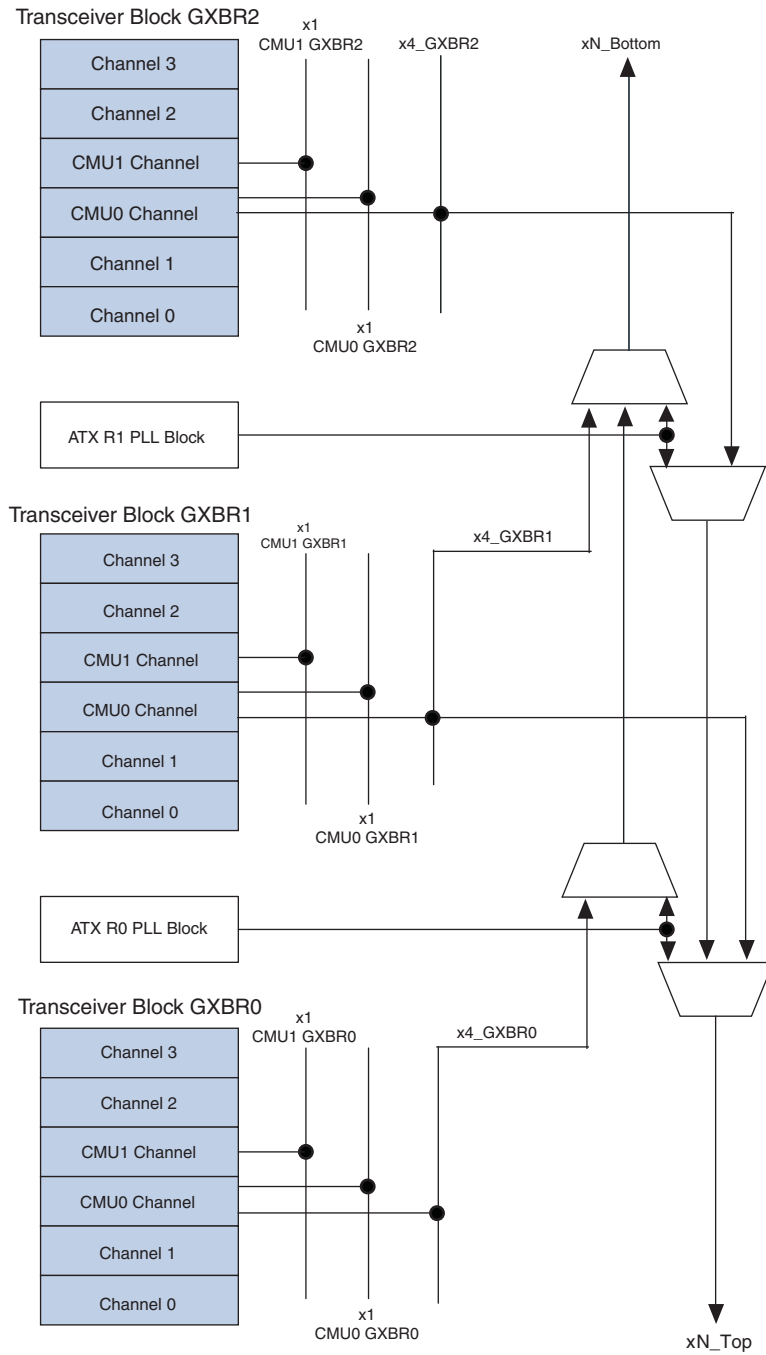
You can force the placement of the transceiver channels across multiple transceiver blocks on the same side of the device by assigning pins to the transmitter and receiver serial ports.

The logical channel 0 of the Basic (PMA Direct) $\times N$ 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 PCIe $\times 4$ 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 xN_Top and xN_Bottom clock lines.

Figure 3-18. The xN_Top and xN_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

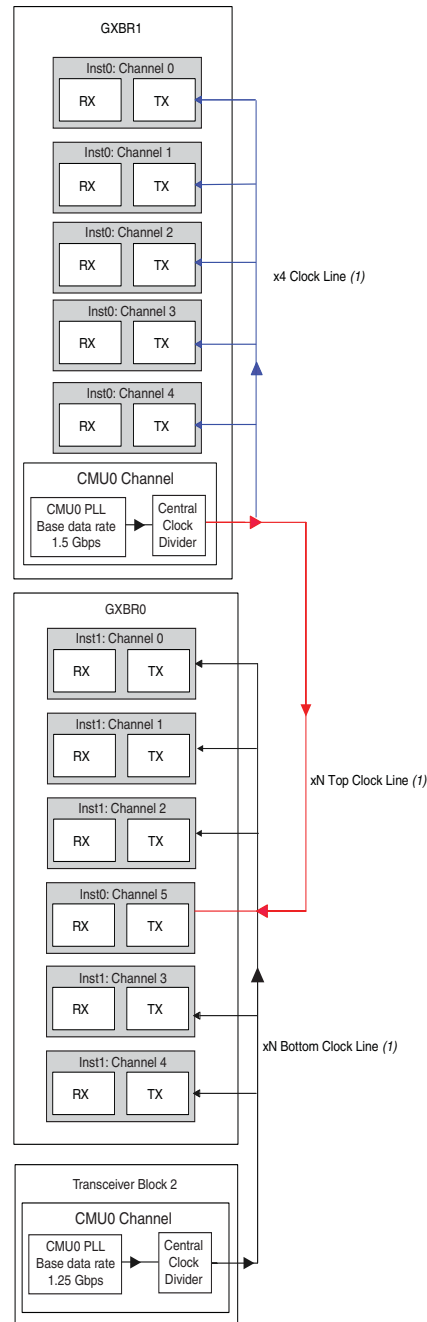
Consider the configuration for the two instances listed in [Table 3-11](#) when combining transceiver channels in Basic (PMA Direct) ×N mode with different data rates.

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 (Gbps)	Configuration
inst0	6	1.5	Receiver and transmitter
inst1	5	1.25	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 $\times N$ top clock line, the blue lines represent the $\times 4$ clock line, and the black lines represent the $\times N$ 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 `xN_Top` clock line.
- Some of the channels in transceiver block 1 receive their high-speed clock from the `xN_Bottom` clock line. Because the `xN_Top` and `xN_Bottom` lines are separate, this scenario is allowed. To understand the clock multiplexer on the `xN` clock lines, refer to [Figure 3-18 on page 3-34](#).

Combining Channels Configured in Basic (PMA Direct) xN 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) xN mode with instances in non-Basic (PMA Direct) configurations; for example, GIGE and SDI.

Example 9

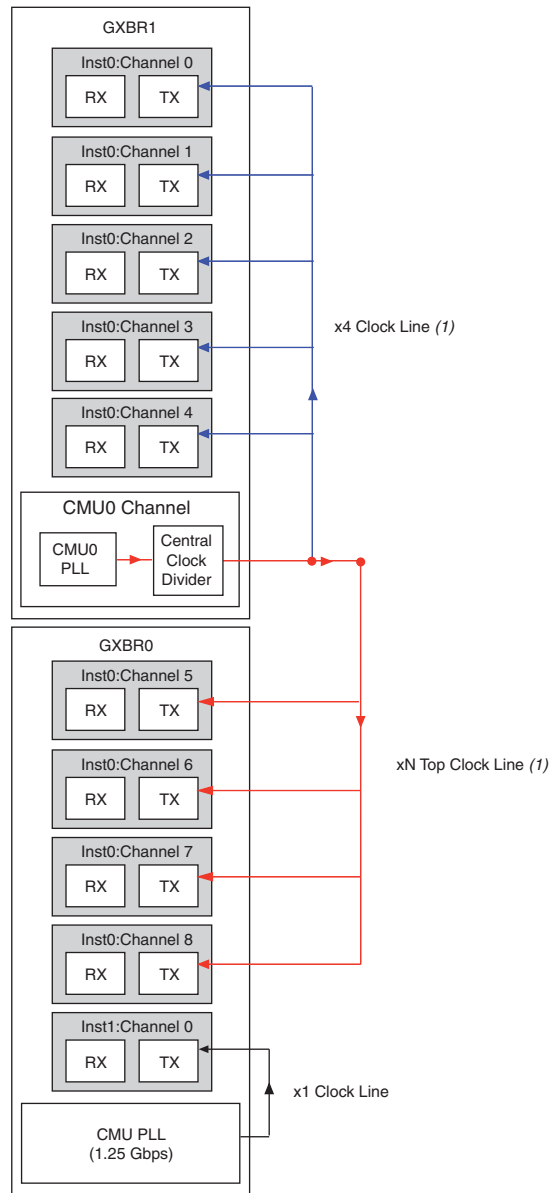
Consider the example design listed in [Table 3-12](#) for the two instances when combining a Basic (PMA Direct) xN configuration with a non-Basic (PMA Direct) configuration using a CMU PLL.

Table 3-12. Combining Basic (PMA Direct) xN Configuration with Non-Basic (PMA Direct) Configuration Using CMU PLL for Example 9

User Defined Instance Name	Number of Channels	Effective Data Rate (Gbps)	Configuration	Functional Mode
<code>inst0</code>	9	1.5	Receiver and Transmitter	Basic (PMA Direct) xN
<code>inst1</code>	1	1.25	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) $\times N$ 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 $\times N$ top clock line, the blue lines represent the $\times 4$ clock line, and the black line represents the $\times N$ bottom clock line.

Example 10

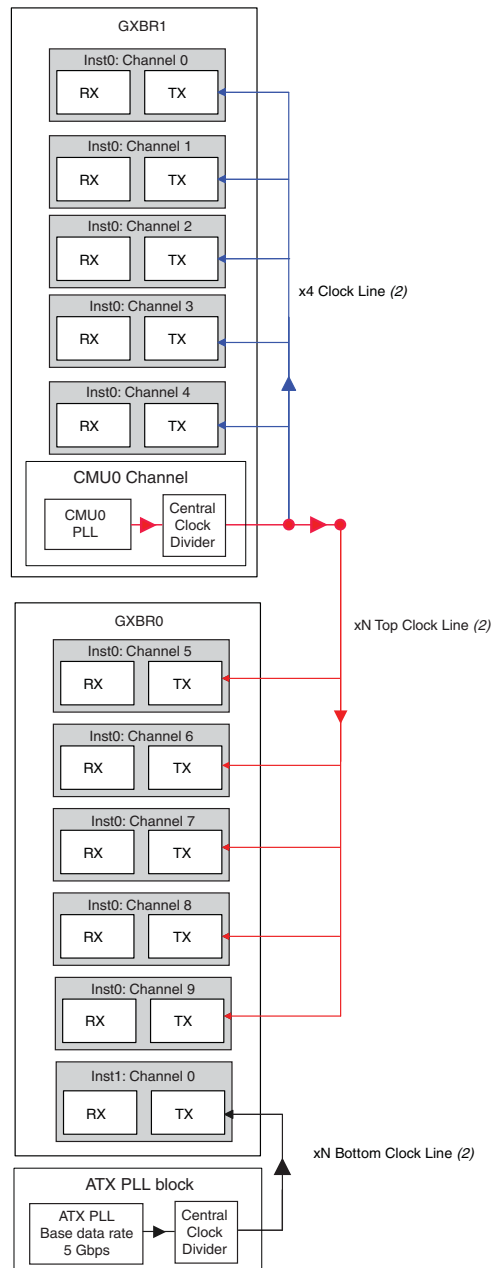
Consider the example design listed in Table 3-13 when combining a Basic (PMA Direct) $\times N$ configuration with a non-Basic (PMA Direct) configuration using an ATX PLL.

Table 3-13. Combining Basic (PMA Direct) $\times N$ Configuration with Non-Basic (PMA Direct) Configuration Using ATX PLL for Example 10

User Defined Instance Name	Number of Channels	Effective Data Rate (Gbps)	Configuration	Functional Mode
inst0	10	1.5	Receiver and Transmitter	Basic (PMA Direct) $\times N$ configuration
inst1	1	5	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 `inst1`. Therefore, you can combine 10 channels of `inst0` and one channel of `inst1` in two transceiver blocks, as shown in [Figure 3-21](#).

Figure 3-21. Combining Basic (PMA Direct) $\times N$ Configuration with Non-Basic (PMA Direct) Configuration Using an ATX PLL for Example 10 ⁽¹⁾



Notes to Figure 3-21:

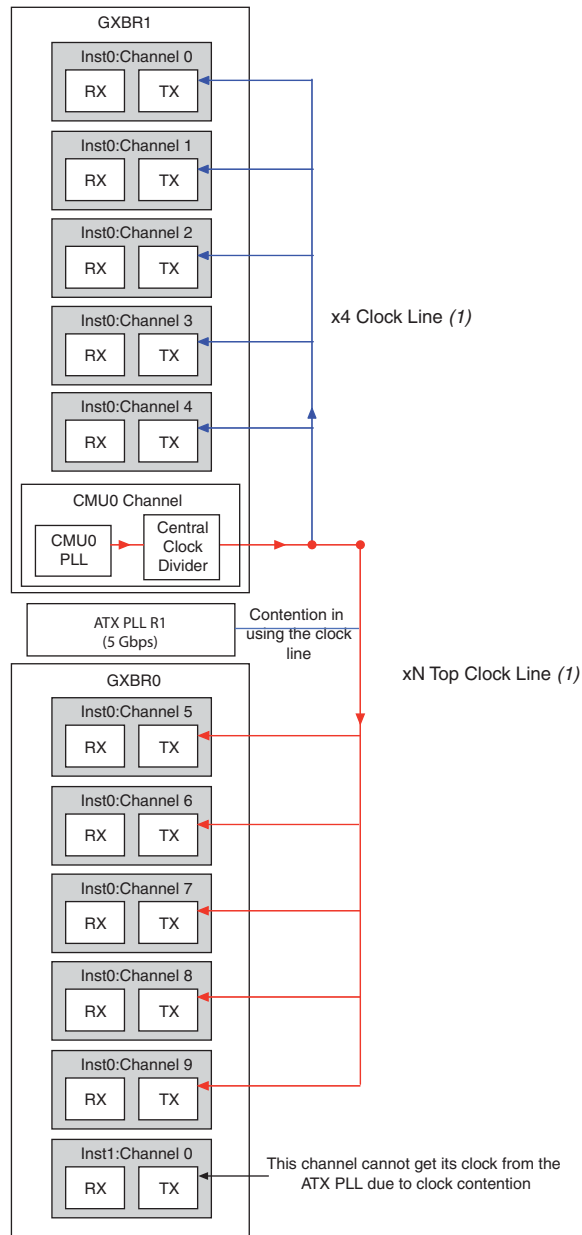
- (1) The ATX PLL provides the high-speed clock to channel 0 of `inst1`.
- (2) The red lines represent the $\times N$ top clock line, the blue lines represent the $\times 4$ clock line, and the black line represents the $\times N$ bottom clock line.

You can also combine channels configured in Basic (PMA Direct) $\times N$ mode with bonded $\times 4$ and $\times 8$ functional modes. For example scenarios, refer to [Figure 3-8 on page 3-19](#) and [Figure 3-10 on page 3-22](#).

Example 11

Consider the unsupported placement design example shown [Figure 3-22](#). The placement is unsupported because of the $\times N_{Top}$ clock line contention between the ATX PLL and the CMU0 PLL in transceiver block 0.

Figure 3-22. Unsupported Placement Due to $\times N$ Clock Line Contention for Example 11



Note to Figure 3-22:

(1) The red lines represent the $\times N$ top clock line and the blue lines represent the $\times 4$ clock line.

Combination Requirements When You Enable Channel Reconfiguration

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 *Dynamic Reconfiguration in Stratix IV Devices* 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 You Enable the Use Alternate CMU PLL Option

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. This setting is required for all instances and channels controlled by the shared `ALT_GX_Reconfig`.

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 second 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 listed 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 (Gbps)	Input Reference Clock (MHz)	PLL Logical Reference Index
Main PLL	1.25	125	0
Alternate PLL	2.488	155.5	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 (Gbps)	Input Reference Clock (MHz)	PLL Logical Reference Index
Main PLL	2.488	155.5	1
Alternate PLL	1.25	125	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 You Use Multiple TX PLLs

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 You Enable the Use Alternate CMU PLL Option](#)” on page 3-42.
- 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 3-10 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.

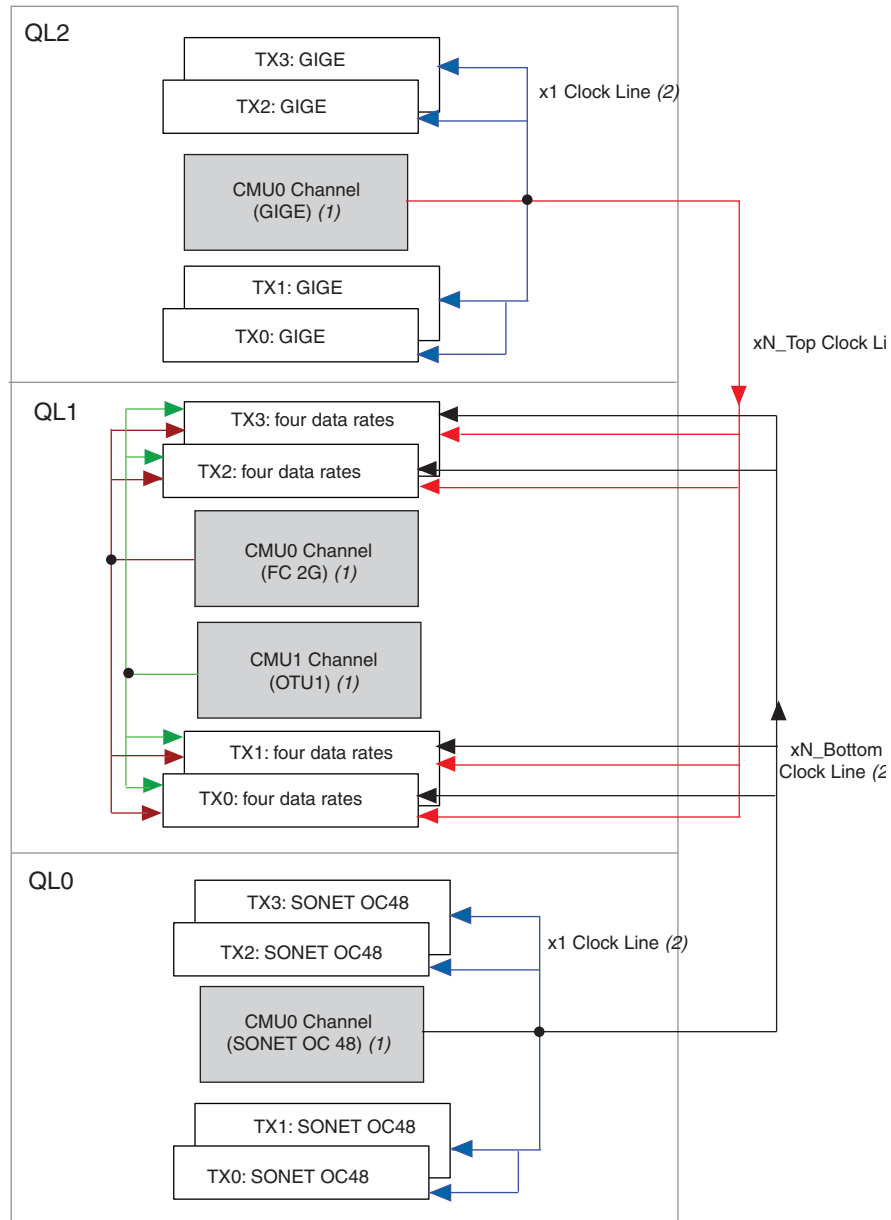
To implement step 1, you need four TX PLLs. Place the four channels in the middle transceiver block (QL1—the left side was chosen for illustration purposes), and provide one CMU PLL from QL0 for the SONET OC48 data rate and one CMU PLL from QL2 for the GIGE data rate. Use the two CMU PLLs from QL1 for 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 $\times N$ top clock line, the blue lines represent the $\times 1$ clock line, the black lines represent $\times N$ 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 You Enable the Use Alternate CMU PLL Option”](#) on page 3-42.

Combining Transceiver Channels When You Enable the Adaptive Equalization (AEQ) Feature

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 *Dynamic Reconfiguration in Stratix IV Devices* 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.

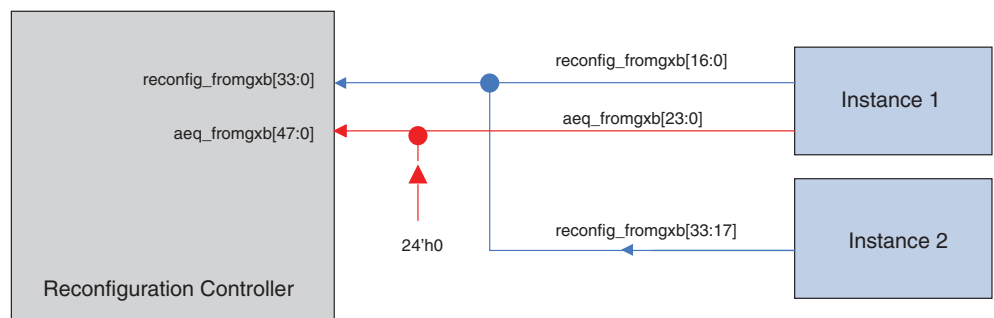
Because there are two instances, the starting channel numbers of Instance 1 and Instance 2 are spaced four apart (0 and 4, respectively).


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 *Dynamic Reconfiguration in Stratix IV Devices* 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 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 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 11.3 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 11.3 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 `p11_powerdown` signal. For each transceiver instance, the ALTGX MegaWizard Plug-In Manager provides an option to select the `p11_powerdown` port. If you want to share the same CMU PLL between multiple transceiver channels, connect the `p11_powerdown` ports of the instances and drive the signal from the same logic.

- If you enable the PCIe hard IP block using the PCI Express 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 PCIe hard IP Block with Other Channels”](#) on page 3-24.
- 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 lists the revision history for this chapter.

Table 3-23. Document Revision History (Part 1 of 2)

Date	Version	Changes
September 2012	4.2	Updated the “Overview” section to close FB #65273.
February 2011	4.1	<ul style="list-style-type: none"> ■ Updated Table 3-15. ■ Updated the “Multiple Channels Sharing a CMU PLL”, “Transmitter Buffer Voltage (VCCH)”, “Transceiver Analog Power (VCCA_L/R)”, “Calibration Clock and Power Down”, “Combining Transceiver Instances Using PLL Cascade Clocks”, “Combining Channels Using the PCIe hard IP Block with Other Channels”, “Combination Requirements When You Enable the Use Alternate CMU PLL Option”, and “Placement Rules for Transceiver Channels at 9.9 Gbps to 11.3 Gbps” sections. ■ Updated chapter title. ■ Applied new template. ■ Minor text edits.
November 2009	4.0	<ul style="list-style-type: none"> ■ Added “Sharing ATX PLLs” on page 3-9, “Combination Requirements When Channel Reconfiguration is Enabled” on page 3-42, “Combining Transceiver Channels When the Adaptive Equalization (AEQ) is Enabled” on page 3-47, and “Combination Requirements for Stratix IV GT Devices” on page 3-49. ■ 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.
June 2009	3.1	<ul style="list-style-type: none"> ■ Updated Table 3-7. ■ Minor text edits.
March 2009	3.0	<ul style="list-style-type: none"> ■ 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.

Table 3-23. Document Revision History (Part 2 of 2)

Date	Version	Changes
November 2008	2.0	<ul style="list-style-type: none">■ 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	1.0	Initial release.

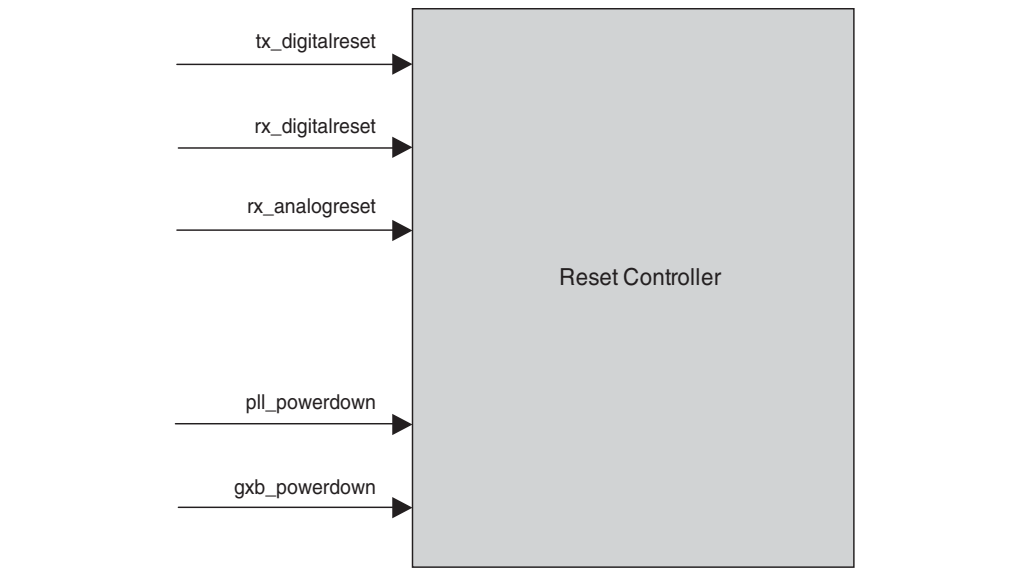
Stratix® IV devices offer multiple reset signals to control the transceiver channels and clock multiplier unit (CMU) phase-locked loops (PLLs) independently. The ALTGX Transceiver MegaWizard™ Plug-In Manager provides individual reset signals for each channel instantiated in your design. It also provides one power-down signal for each transceiver block.

This chapter includes the following sections:

- “User Reset and Power-Down Signals” on page 4–2
- “Transceiver Reset Sequences” on page 4–4
- “PMA Direct Drive Mode Reset Sequences” on page 4–24
- “Dynamic Reconfiguration Reset Sequences” on page 4–36
- “Power Down” on page 4–38
- “Simulation Requirements” on page 4–39
- “Reference Information” on page 4–39

Figure 4–1 shows the reset control and power-down block for a Stratix IV device.

Figure 4–1. Reset Control and Power-Down Block



User Reset and Power-Down Signals

Each transceiver channel in the Stratix IV 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⁾	<ul style="list-style-type: none"> ■ Transmitter Only ■ Receiver and Transmitter 	<p>Provides asynchronous reset to all digital logic in the transmitter PCS, including the XAUI transmit state machine.</p> <p>The minimum pulse width for this signal is two parallel clock cycles.</p>
rx_digitalreset ⁽¹⁾	<ul style="list-style-type: none"> ■ Receiver Only ■ Receiver and Transmitter 	<p>Resets all digital logic in the receiver PCS, including:</p> <ul style="list-style-type: none"> ■ XAUI receiver state machines ■ GIGE receiver state machines ■ XAUI channel alignment state machine ■ BIST-PRBS verifier ■ BIST-incremental verifier <p>The minimum pulse width for this signal is two parallel clock cycles.</p>
rx_analogreset	<ul style="list-style-type: none"> ■ Receiver Only ■ Receiver and Transmitter 	<p>Resets the receiver CDR present in the receiver channel.</p> <p>The minimum pulse width is two parallel clock cycles.</p>

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 the 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

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, it powers down: <ul style="list-style-type: none"> the PCS and PMA in all the transceiver channels the CMU PLLs This signal operates independently from the other reset signals and is common to the transceiver block.
pll_locked	A status signal. Indicates the status of the transmitter PLL. <ul style="list-style-type: none"> A high level—the transmitter PLL is locked to the incoming reference clock frequency. When <code>pll_locked</code> is low, <code>tx_digitalreset</code> must always be asserted. To de-assert <code>tx_digitalreset</code>, follow the initialization reset sequence for your specific mode.
rx_pll_locked	A status signal. <ul style="list-style-type: none"> A high level—the receiver CDR is locked to the incoming reference clock frequency.
rx_freqlocked	A status signal. Indicates the status of the receiver CDR lock mode. <ul style="list-style-type: none"> A high level—the receiver is in lock-to-data (LTD) mode. A low level—the receiver CDR is in lock-to-reference (LTR) mode. In automatic lock mode, when <code>rx_freqlocked</code> is low, <code>rx_digitalreset</code> must always be asserted. To de-assert <code>rx_digitalreset</code>, follow the initialization reset sequence for your specific 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 <code>reconfig_clk</code> clock cycle after power up. It then is asserted from the second <code>reconfig_clk</code> 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 *Dynamic Reconfiguration in Stratix IV Devices* 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	—	—	—	Y	Y
Transmitter Phase Compensation FIFO	—	—	Y	—	Y

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
Byte Serializer	—	—	Y	—	Y
8B/10B Encoder	—	—	Y	—	Y
Serializer	—	—	Y	—	Y
Transmitter Buffer	—	—	—	—	Y
Transmitter XAUI State Machine	—	—	Y	—	Y
Receiver Buffer	—	—	—	—	Y
Receiver CDR	—	Y	—	—	Y
Receiver Deserializer	—	—	—	—	Y
Receiver Word Aligner	Y	—	—	—	Y
Receiver Deskew FIFO	Y	—	—	—	Y
Receiver Clock Rate Compensation FIFO	Y	—	—	—	Y
Receiver 8B/10B Decoder	Y	—	—	—	Y
Receiver Byte Deserializer	Y	—	—	—	Y
Receiver Byte Ordering	Y	—	—	—	Y
Receiver Phase Compensation FIFO	Y	—	—	—	Y
Receiver XAUI State Machine	Y	—	—	—	Y
BIST Verifiers	Y	—	—	—	Y

Transceiver Reset Sequences

You can configure transceiver channels in Stratix IV 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[®] (PCIe) 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 devices described in this chapter are:

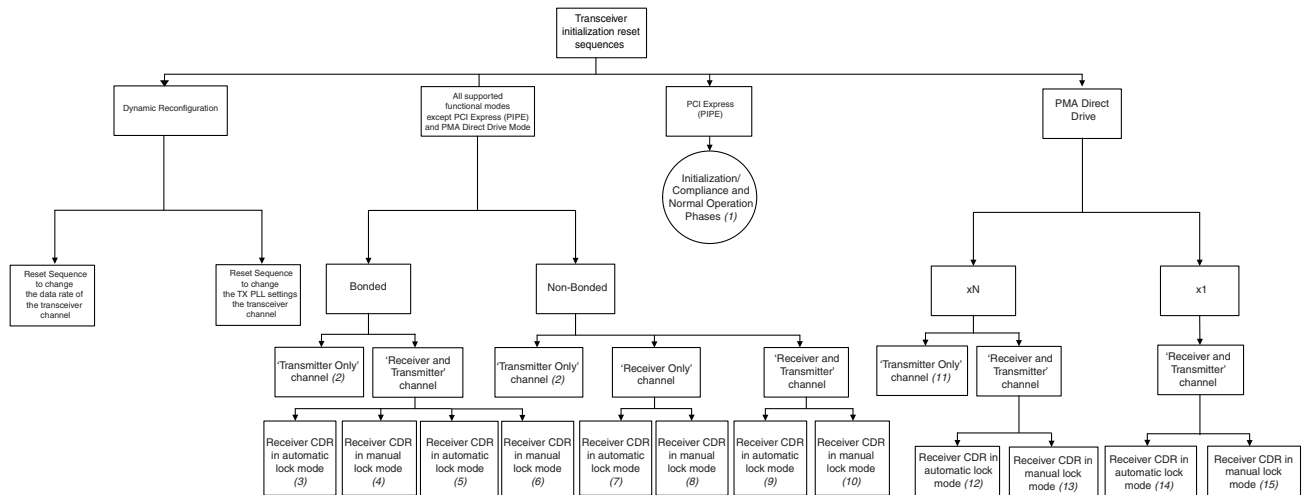
- [“All Supported Functional Modes Except PCIe Functional Mode” on page 4-6](#)—describes the reset sequences in bonded and non-bonded configurations.
- [“PCIe Functional Mode” on page 4-22](#)—describes the reset sequence for the initialization/compliance phase and the normal operation phase in PCIe functional modes.

The busy signal remains low for the first `reconfig_clk` clock cycle. It then is 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. For more information, refer to the reset sequences shown in Figure 4-2 and the associated references listed in the figure notes.

Altera strongly recommends adhering to these reset sequences for proper operation of the Stratix IV transceiver.

Figure 4-2 shows the transceiver reset sequences for Stratix IV 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 PCIe 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 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

<code>rx_locktorefclk</code>	<code>rx_locktodata</code>	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 XAUI, PCIe, and Basic $\times 4$ functional modes. In Basic $\times 4$ 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

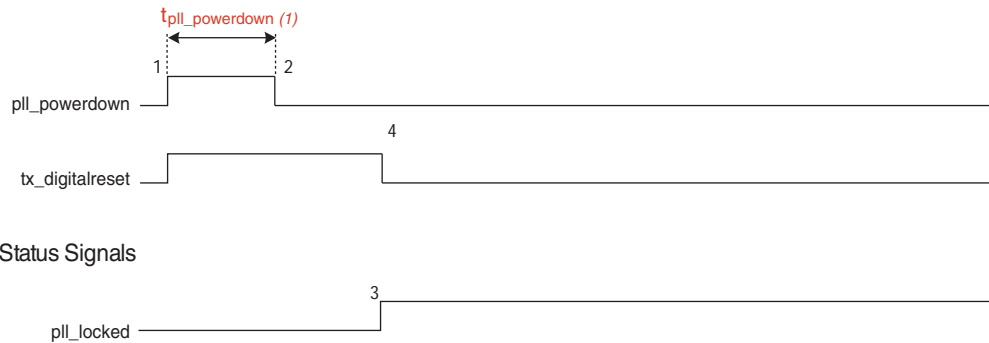
Channel Set Up	Receiver CDR Mode	Refer to
Transmitter Only	Basic $\times 4$	“Transmitter Only Channel” on page 4-7
Receiver and Transmitter	Automatic lock mode for XAUI functional mode	“Receiver and Transmitter Channel—Receiver CDR in Automatic Lock Mode” on page 4-8
Receiver and Transmitter	Manual lock 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 $\times 8$ functional mode	“Receiver and Transmitter Channel—Receiver CDR in Automatic Lock Mode” on page 4-12
Receiver and Transmitter	Manual lock mode for Basic $\times 8$ functional mode	“Receiver and Transmitter Channel—Receiver CDR in Manual Lock Mode” on page 4-14

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

Reset and Power-Down Signals



Output Status Signals

Note to Figure 4-3:

(1) For $t_{pll_powerdown}$ duration, refer to the *DC and Switching Characteristics for Stratix IV Devices* chapter.

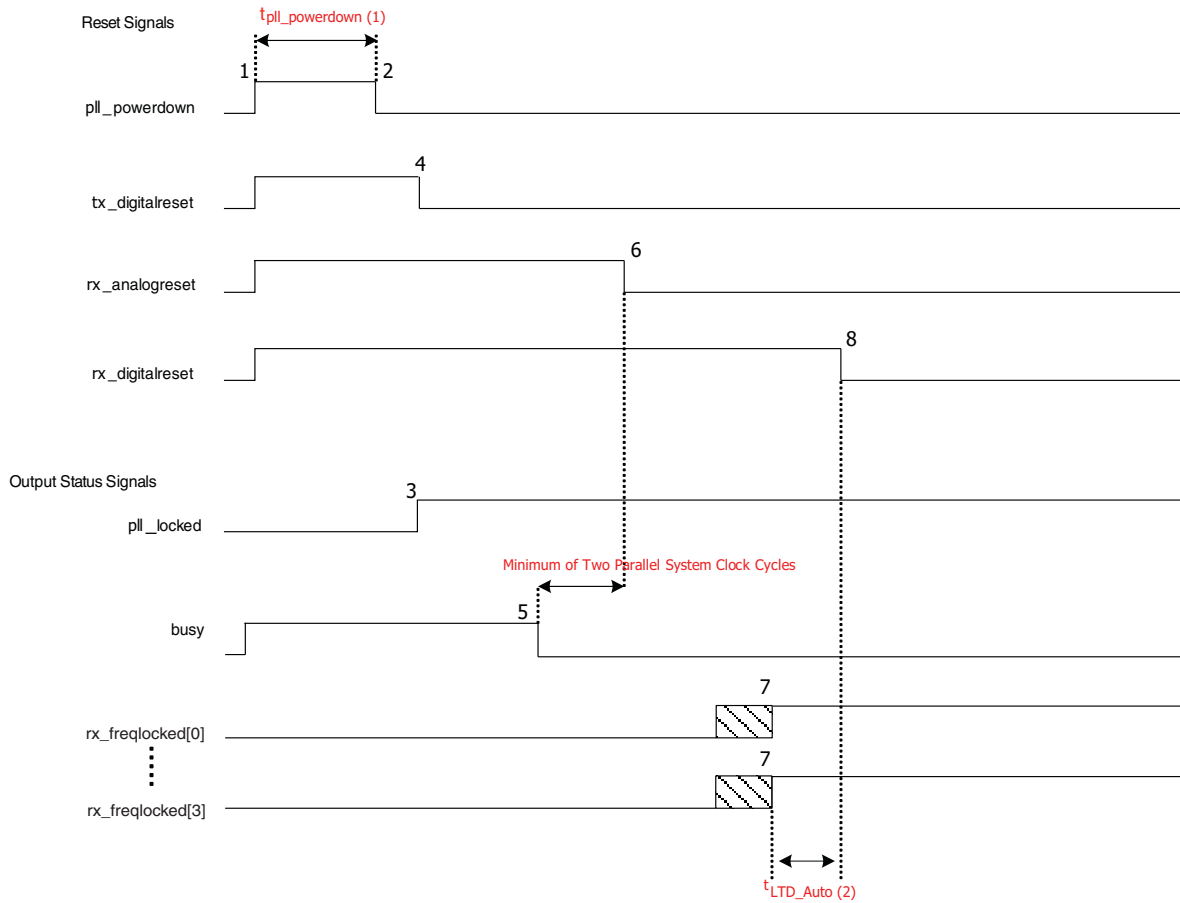
As shown in [Figure 4-3](#), for the **Transmitter Only** channel configuration, follow these reset steps:

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:**

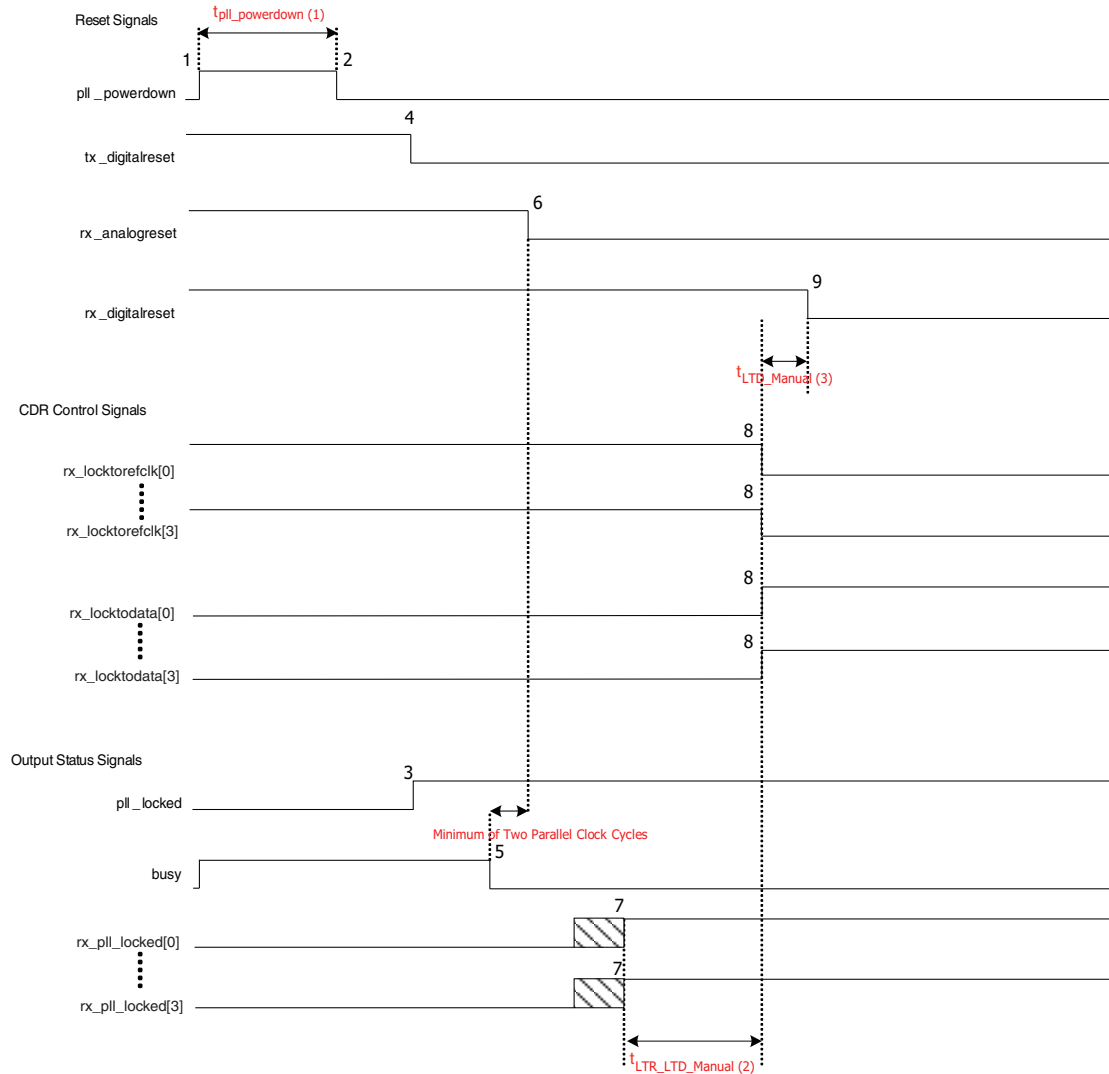
- (1) For $t_{pll_powerdown}$ duration, refer to the *DC and Switching Characteristics for Stratix IV Devices* chapter.
- (2) For t_{LTD_Auto} duration, refer to the *DC and Switching Characteristics for Stratix IV Devices* chapter.

As shown in [Figure 4-4](#), for the receiver CDR in automatic lock mode configuration, follow these reset steps:

1. After power up, assert `p11_powerdown` for a minimum period of $t_{p11_powerdown}$ (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 `p11_powerdown` signal, the transmitter PLL starts locking to the transmitter input reference clock.
3. After the transmitter PLL locks, as indicated by the `p11_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 a minimum of 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} 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. Note that `rx_digitalreset` must not be released if there is no data present at the receiver pins to avoid overflow/underflow of the phase compensation FIFOs.

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 for Stratix IV Devices* chapter.
- (2) For $t_{LTR_LTD_Manual}$ duration, refer to the *DC and Switching Characteristics for Stratix IV Devices* chapter.
- (3) For t_{LTD_Manual} duration, refer to the *DC and Switching Characteristics for Stratix IV Devices* chapter.

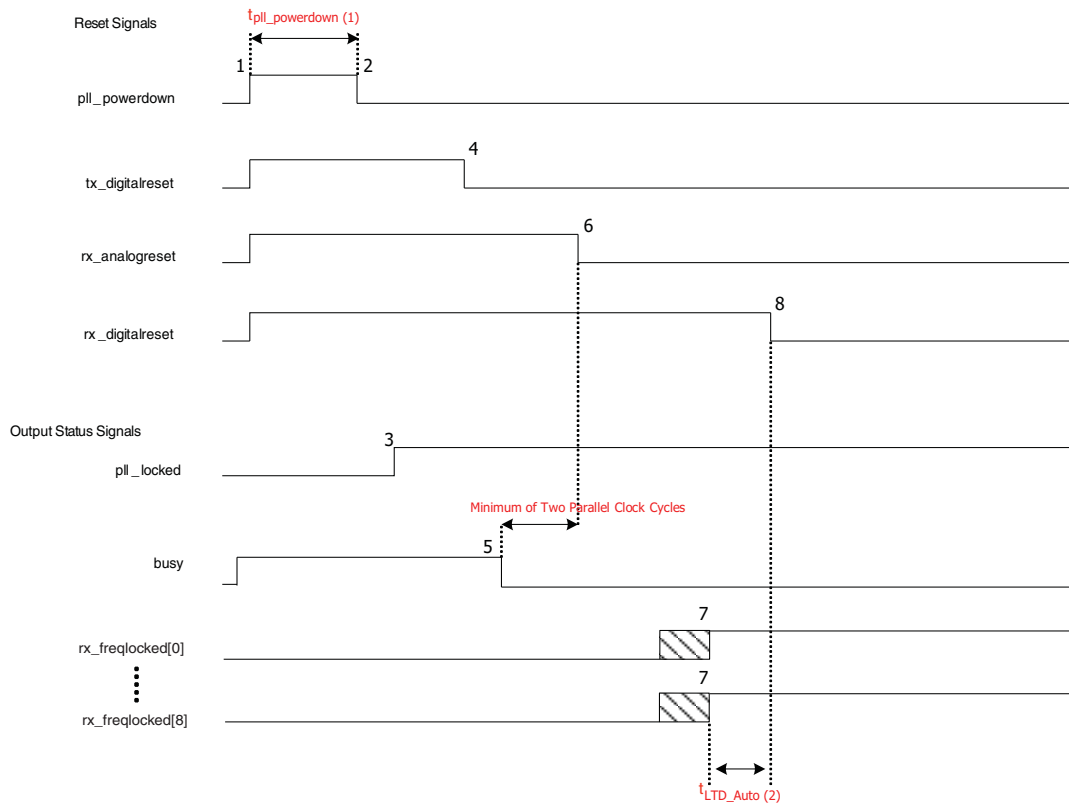
As shown in Figure 4-5, for the receiver CDR in manual lock mode configuration, follow these reset steps:

1. After power up, assert `p11_powerdown` for a minimum period of $t_{p11_powerdown}$ (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 `p11_powerdown` signal, the transmitter PLL starts locking to the transmitter input reference clock.
3. After the transmitter PLL locks, as indicated by the `p11_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 a minimum of **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_p11_locked` signal from each channel to go high. The `rx_p11_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_p11_locked` signal of all the channels have gone high, from that point onwards, wait for at least $t_{LTR_LTD_Manual}$, 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} 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 $\times 8$ 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:**

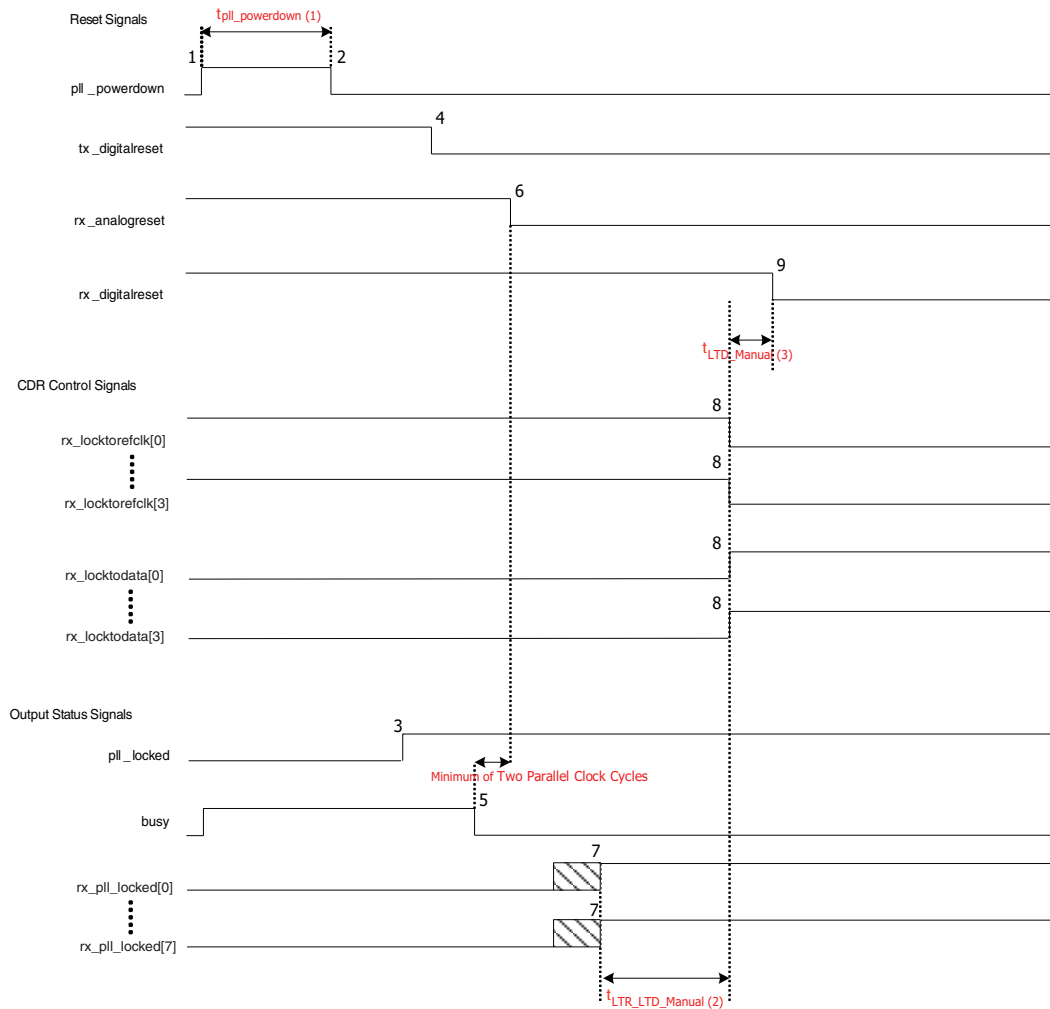
- (1) For $t_{pll_powerdown}$ duration, refer to the *DC and Switching Characteristics for Stratix IV Devices* chapter.
- (2) For t_{LTD_Auto} duration, refer to the *DC and Switching Characteristics for Stratix IV Devices* chapter.

As shown in Figure 4-6, for the receiver CDR in automatic lock mode, follow these reset steps:

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 a minimum of **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} 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. Note that `rx_digitalreset` must not be released if there is no data present at the receiver pins to avoid overflow/underflow of the phase compensation FIFOs.

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 for Stratix IV Devices* chapter.
- (2) For $t_{LTR_LTD_Manual}$ duration, refer to the *DC and Switching Characteristics for Stratix IV Devices* chapter.
- (3) For t_{LTD_Manual} duration, refer to the *DC and Switching Characteristics for Stratix IV Devices* chapter.

As shown in Figure 4-7, for the receiver CDR in manual lock mode, follow these reset steps:

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 a minimum of 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 $t_{LTR_LTD_Manual}$, 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} (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 x4	"Transmitter Only Channel" on page 4-16
Receiver Only	Automatic lock mode	"Receiver Only Channel—Receiver CDR in Automatic Lock Mode" on page 4-16
Receiver Only	Manual lock mode	"Receiver Only Channel—Receiver CDR in Manual Lock Mode" on page 4-17

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 4-18
Receiver and Transmitter	Manual lock mode	“Receiver and Transmitter Channel—Receiver CDR in Manual Lock Mode” on page 4-20



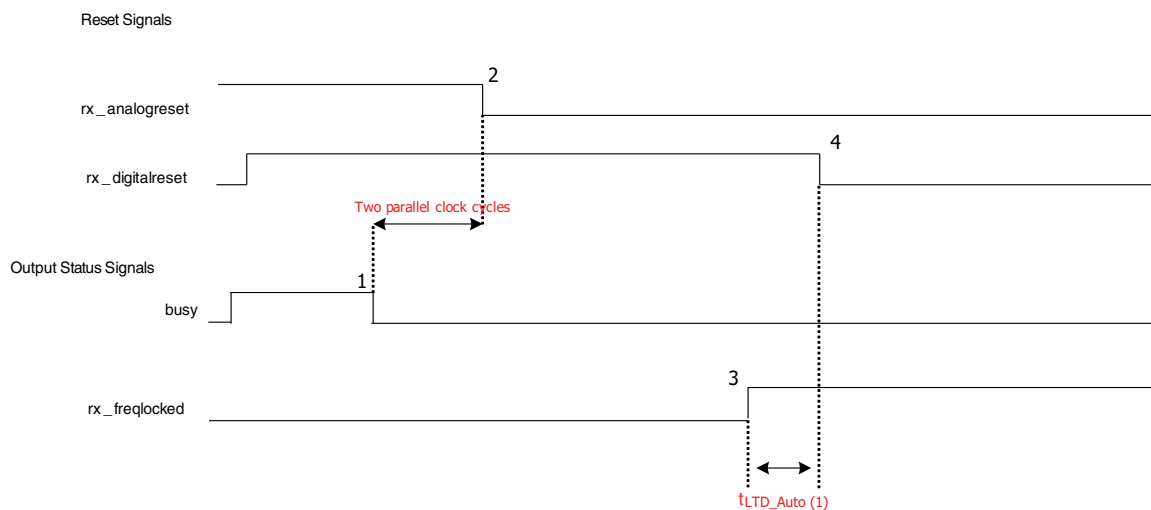
Follow the same reset sequence for all the other channels in the non-bonded configuration.

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 for Stratix IV Devices* chapter.

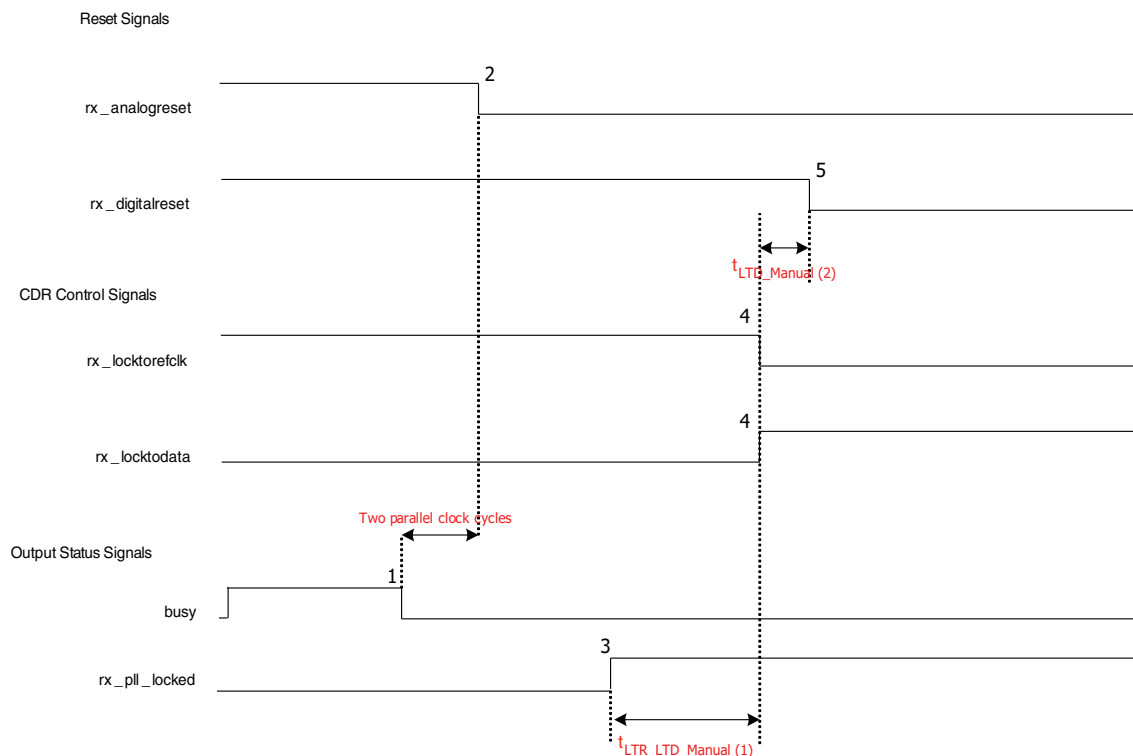
As shown in Figure 4-8, for the receiver in CDR automatic lock mode, follow these reset steps:

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 for Stratix IV Devices* chapter.
- (2) For t_{LTD_Manual} duration, refer to the *DC and Switching Characteristics for Stratix IV Devices* chapter.

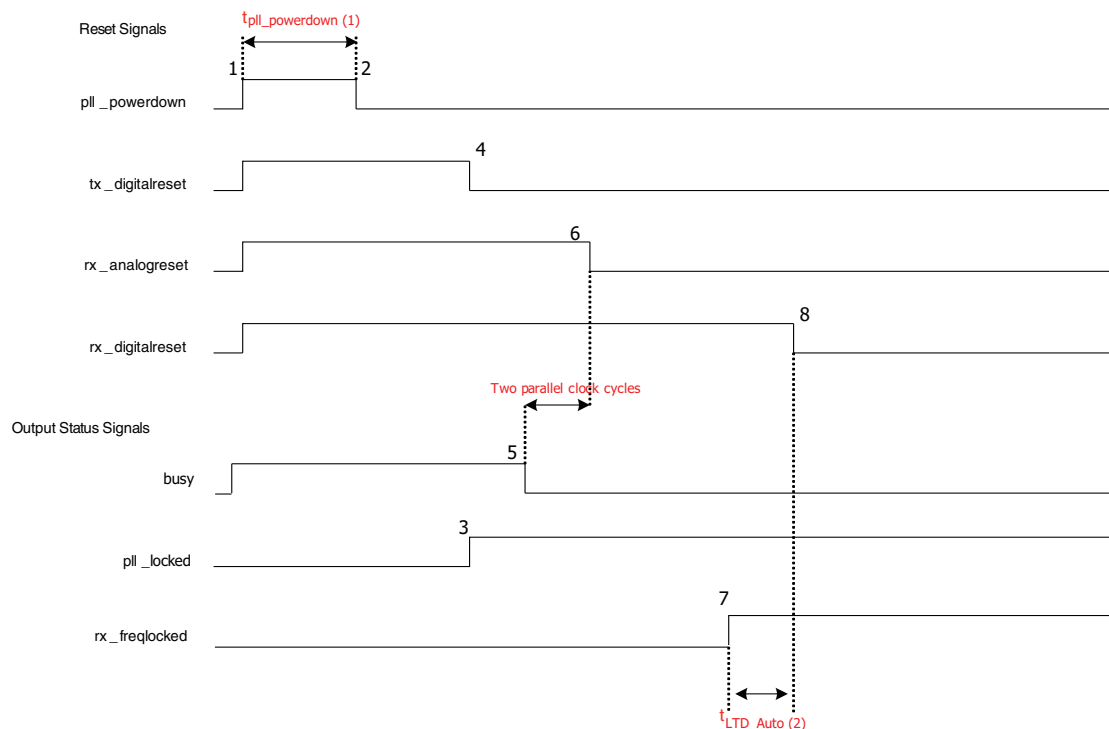
As shown in Figure 4-9, for the receiver CDR in manual lock mode, follow these reset steps:

1. After power up, wait for the busy signal to be asserted.
2. Keep the rx_digitalreset and rx_locktoefclk 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_locktoefclk 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_locktoefclk 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:

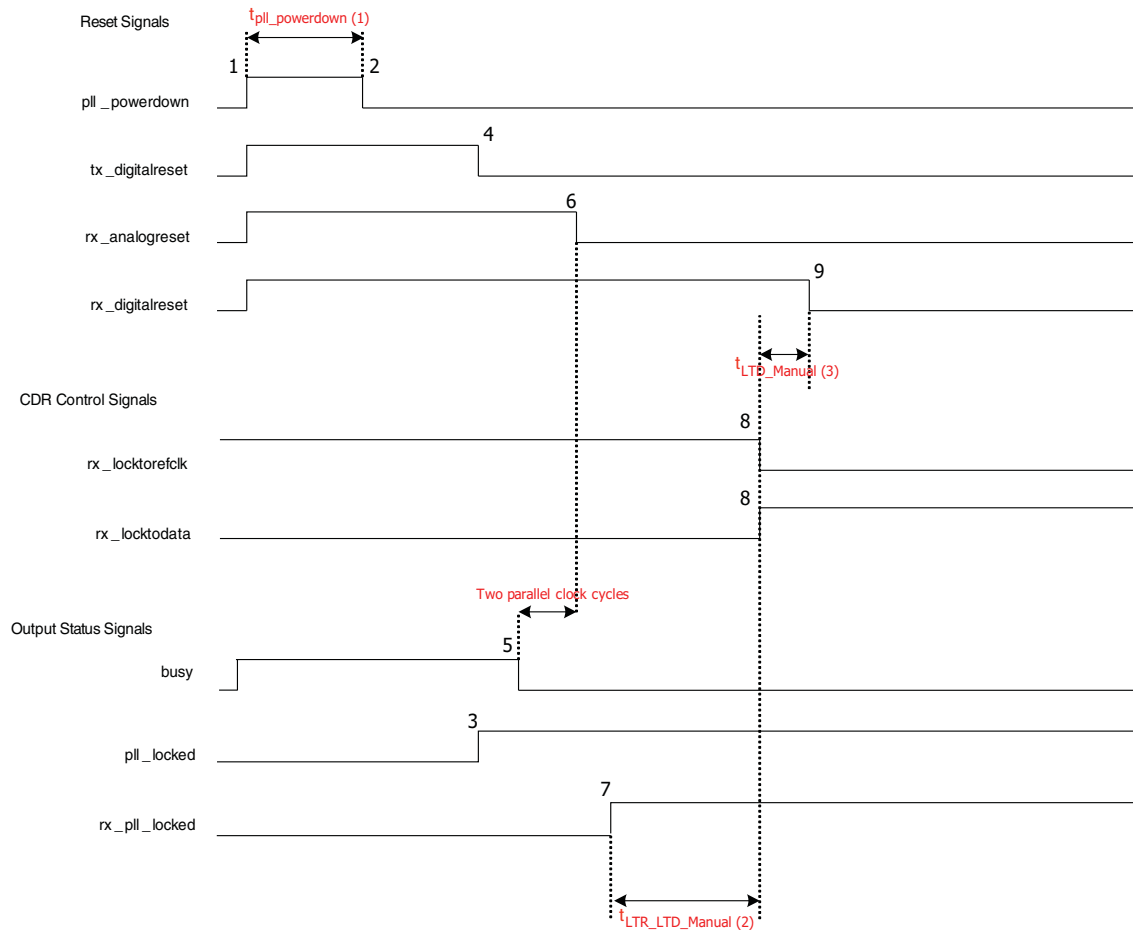
- (1) For $t_{pll_powerdown}$ duration, refer to the *DC and Switching Characteristics for Stratix IV Devices* chapter.
- (2) For t_{LTD_Auto} duration, refer to the *DC and Switching Characteristics for Stratix IV Devices* chapter.

As shown in [Figure 4-10](#), for the receiver in CDR automatic lock mode, follow these reset steps:

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). Note that `rx_digitalreset` must not be released if there is no data present at the receiver pins to avoid overflow/underflow of the phase compensation FIFOs. 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 for Stratix IV Devices* chapter.
- (2) For $t_{LTR_LTD_Manual}$ duration, refer to the *DC and Switching Characteristics for Stratix IV Devices* chapter.
- (3) For t_{LTD_Manual} duration, refer to the *DC and Switching Characteristics for Stratix IV Devices* chapter.

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.

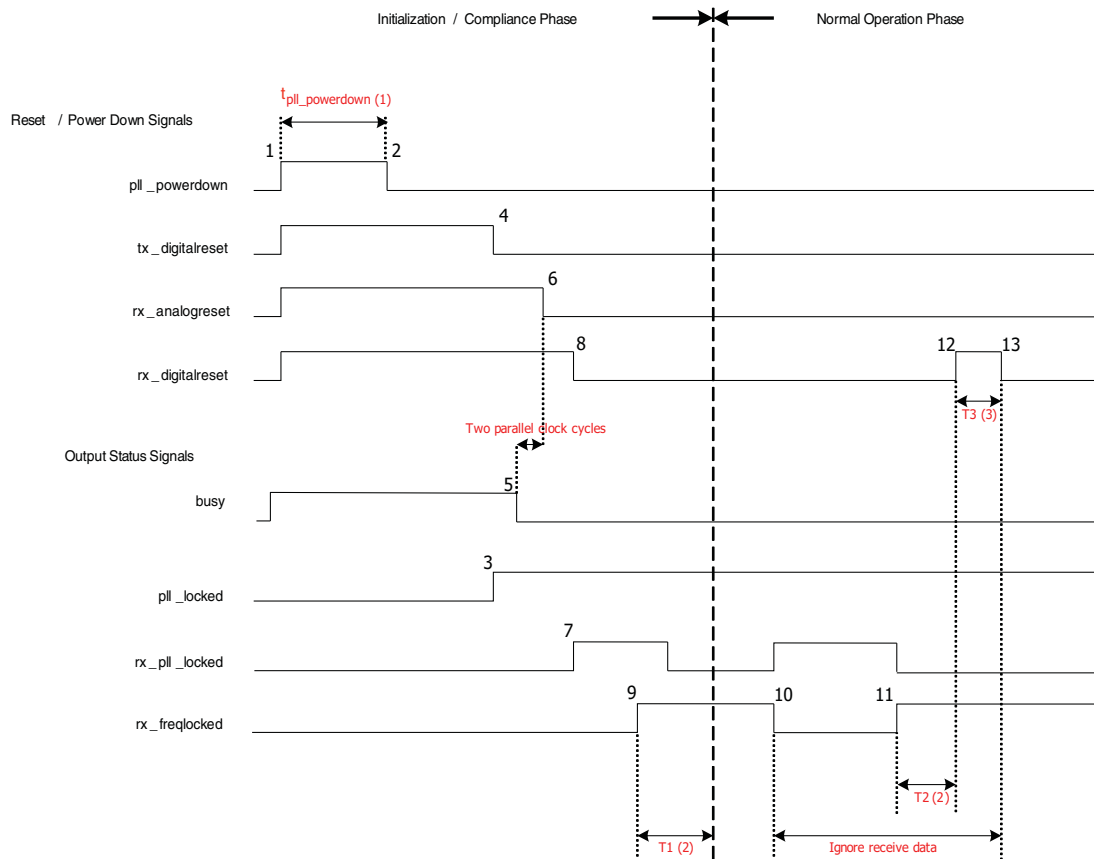
PCIe Functional Mode

You can configure PCIe functional mode with or without the receiver clock rate compensation FIFO in the Stratix IV device. The reset sequence remains the same whether or not you use the receiver clock rate compensation FIFO.

PCIe Reset Sequence

The PCIe 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 PCIe Functional Mode



Notes to Figure 4-12:

- (1) For $t_{pll_powerdown}$ duration, refer to the *DC and Switching Characteristics for Stratix IV Devices* chapter.
- (2) The minimum T1 and T2 period is 4 μ s.
- (3) The minimum T3 period is two parallel clock cycles.

PCIe Initialization/Compliance Phase

After the device is powered up, a PCIe-compliant device goes through the compliance phase during initialization. In this phase, the PCIe protocol requires the system to be operating at the Gen 1 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, follow these reset steps:


1. After power up, assert `p11_powerdown` for a minimum period of $t_{p11_powerdown}$ (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 `p11_powerdown` signal, the transmitter PLL starts locking to the transmitter input reference clock.
2. When the transmitter PLL locks, as indicated by the `p11_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_p11_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.

PCIe Normal Phase

For the normal PCIe phase, follow these steps:

1. After completion of the Initialization/Compliance phase, during the normal operation phase at the Gen 1 data rate, when the `rx_freqlocked` signal is de-asserted (marker 10 in [Figure 4-12](#)), wait for the `rx_p11_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 μ s 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 Gen 2 data rate, asserting the `rx_digitalreset` signal causes the PCIe rate switch circuitry to switch the transceiver to the Gen 1 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 device in $\times 1$, $\times 4$, and $\times 8$ PCIe configurations. The reset sequence described in “PCIe Reset Sequence” on page 4-22 applies to all these multi-lane configurations.

PMA Direct Drive Mode Reset Sequences

Stratix IV 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 $\times N$ functional mode.

Table 4-7. Reset and Power-Down Sequences for PMA Direct Drive $\times N$ Configurations

Channel Set Up	Functional Mode	Refer to
Transmitter Only with no PLL_L/R	Basic (PMA Direct) drive $\times 4$	“Transmitter Only Channel with No PLL_L/R” on page 4-25
Transmitter Only with a PLL_L/R	Manual lock mode	“Transmitter Only Channel with a PLL_L/R” on page 4-26
Receiver and Transmitter	Automatic lock mode for Basic (PMA Direct) drive $\times N$ mode	“Receiver and Transmitter Channel Set-up—Receiver CDR in Automatic Lock Mode” on page 4-28
Receiver and Transmitter	Manual lock mode for Basic (PMA Direct) drive $\times N$ mode	“Receiver and Transmitter Channel Set-up—Receiver CDR in Manual Lock Mode” on page 4-30

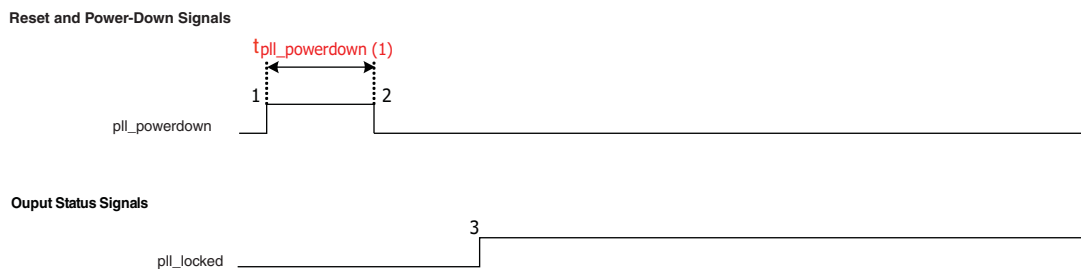
Basic (PMA Direct) Drive $\times N$ Mode

When bonding $\times N$ channels in a Basic (PMA Direct) drive mode configuration, you can reset all bonded channels simultaneously.

Transmitter Only Channel with No PLL_L/R

Figure 4-13 shows an example reset sequence timing diagram of four **Transmitter Only** channels in Basic (PMA Direct) drive $\times 4$ functional mode with no PLL_L/R.

Figure 4-13. Reset Sequence Timing in Basic (PMA Direct) Drive $\times 4$ Mode



Note to Figure 4-13:

(1) For $t_{p11_powerdown}$ duration, refer to the *DC and Switching Characteristics for Stratix IV Devices* chapter.

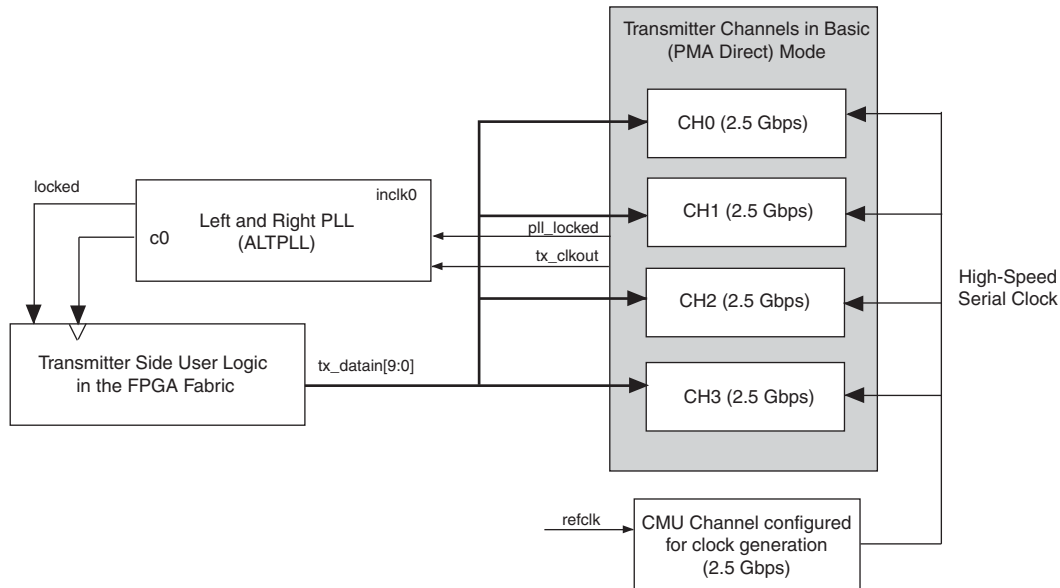
As shown in Figure 4-13, for the **Transmitter Only** channel in Basic (PMA Direct) drive functional $\times 4$ mode with no PLL_L/R, follow these reset steps:

1. After power up, assert `p11_powerdown` for a minimum of $t_{p11_powerdown}$ (the time between markers 1 and 2).
2. When the transmitter PLL locks, as indicated by the `p11_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
- 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, for the **Transmitter Only** channel in Basic (PMA Direct) Drive functional mode with a PLL_L/R configuration, follow these reset steps:

1. After power up, assert `p11_powerdown` for a minimum of $t_{p11_powerdown}$ (the time between markers 1 and 2).
2. After the transmitter PLL locks, as indicated by the `p11_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.

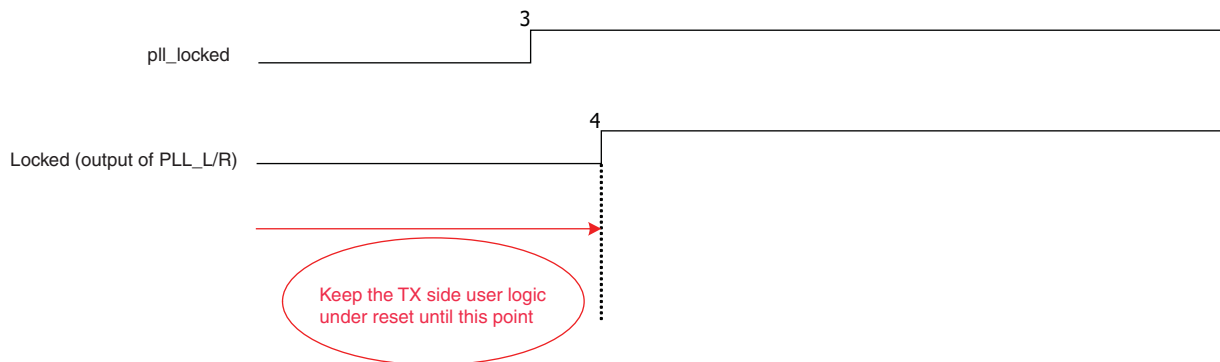
3. After the PLL_L/R locks, as indicated by the locked signal going high (marker 4), the transmitter is ready to accept parallel data from the FPGA fabric and subsequently transmitting serial data reliably.

Figure 4-15. Reset Sequence Timing Diagram of Four Transmitter-Only Channels in Basic (PMA Direct) Drive x4 Functional Mode

Reset and Power-Down Signals



Output Status Signals

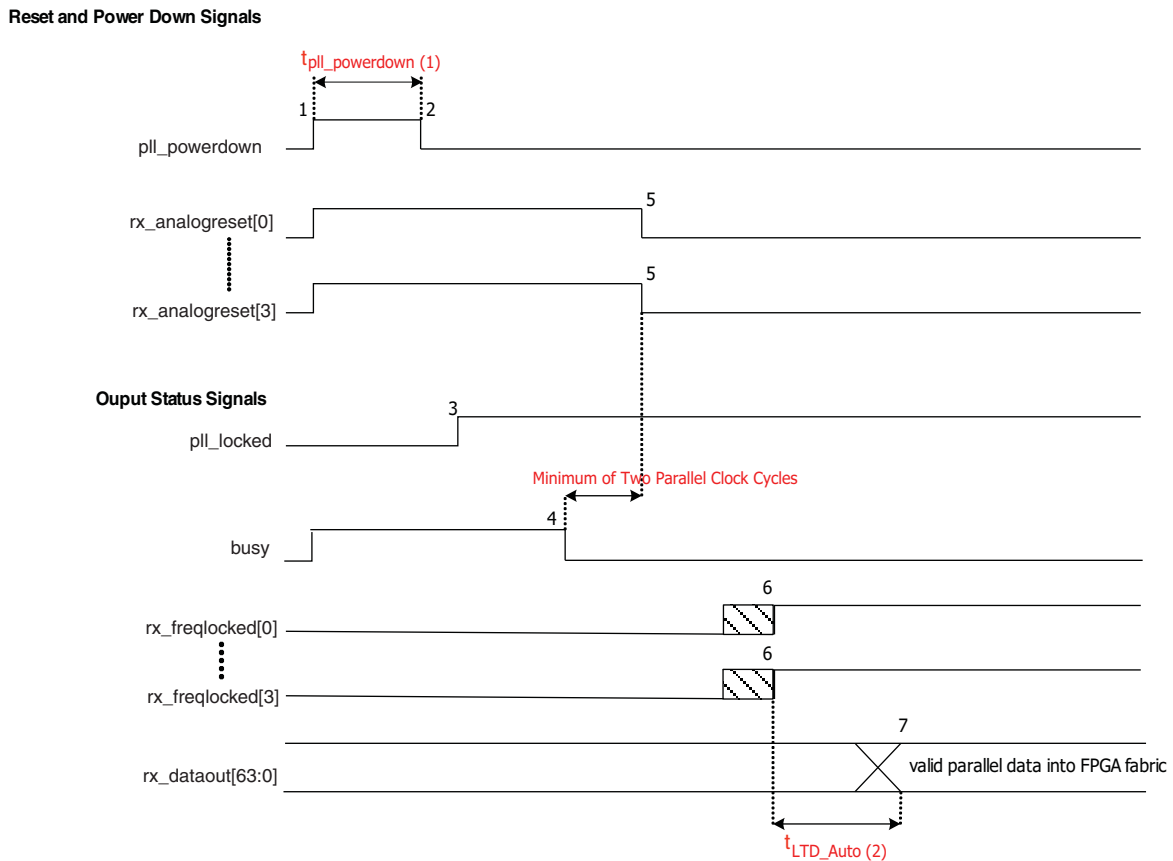


Note to Figure 4-15:

(1) For $t_{pll_powerdown}$ duration, refer to the *DC and Switching Characteristics for Stratix IV Devices* chapter.

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 $\times N$ 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:**

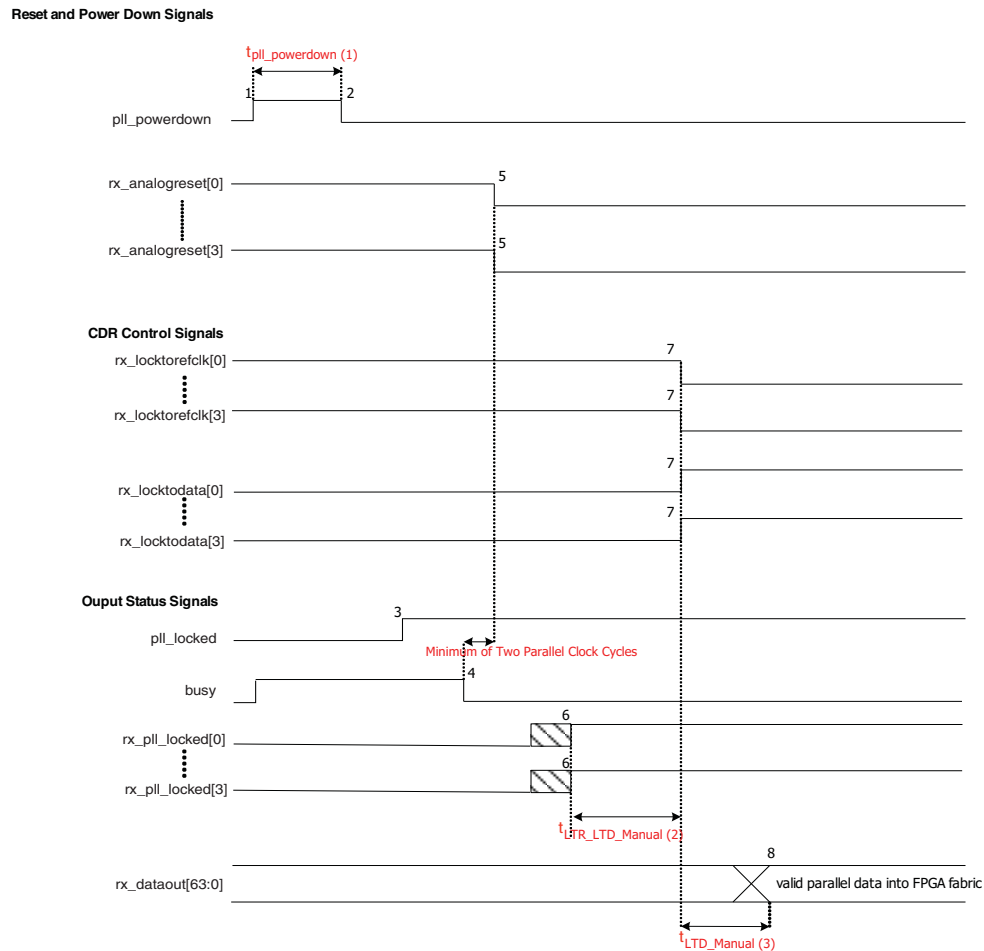
- (1) For $t_{pll_powerdown}$ duration, refer to the *DC and Switching Characteristics for Stratix IV Devices* chapter.
- (2) For t_{LTD_Auto} duration, refer to the *DC and Switching Characteristics for Stratix IV Devices* chapter.

As shown in Figure 4-16, for the receiver and transmitter channel in PMA Direct drive ×4 double-width configuration with CDR in automatic lock mode, follow these reset steps:

1. After power up, assert `p11_powerdown` for a minimum period of $t_{p11_powerdown}$ (the time between markers 1 and 2).
2. Keep the `rx_analogreset` signal asserted during this time period. After you de-assert the `p11_powerdown` signal, the transmitter PLL starts locking to the transmitter input reference clock.
3. When the transmitter PLL locks, as indicated by the `p11_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 a minimum of 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 $\times N$ 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 for Stratix IV Devices* chapter.
- (2) For t_{LTD_Manual} duration, refer to the *DC and Switching Characteristics for Stratix IV Devices* chapter.
- (3) For t_{LTD_Manual} duration, refer to the *DC and Switching Characteristics for Stratix IV Devices* chapter.

As shown in Figure 4-17, for the receiver and transmitter channel in PMA Direct drive ×4 double-width configuration with CDR in manual lock mode, follow these reset steps:

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 a minimum of 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_{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.

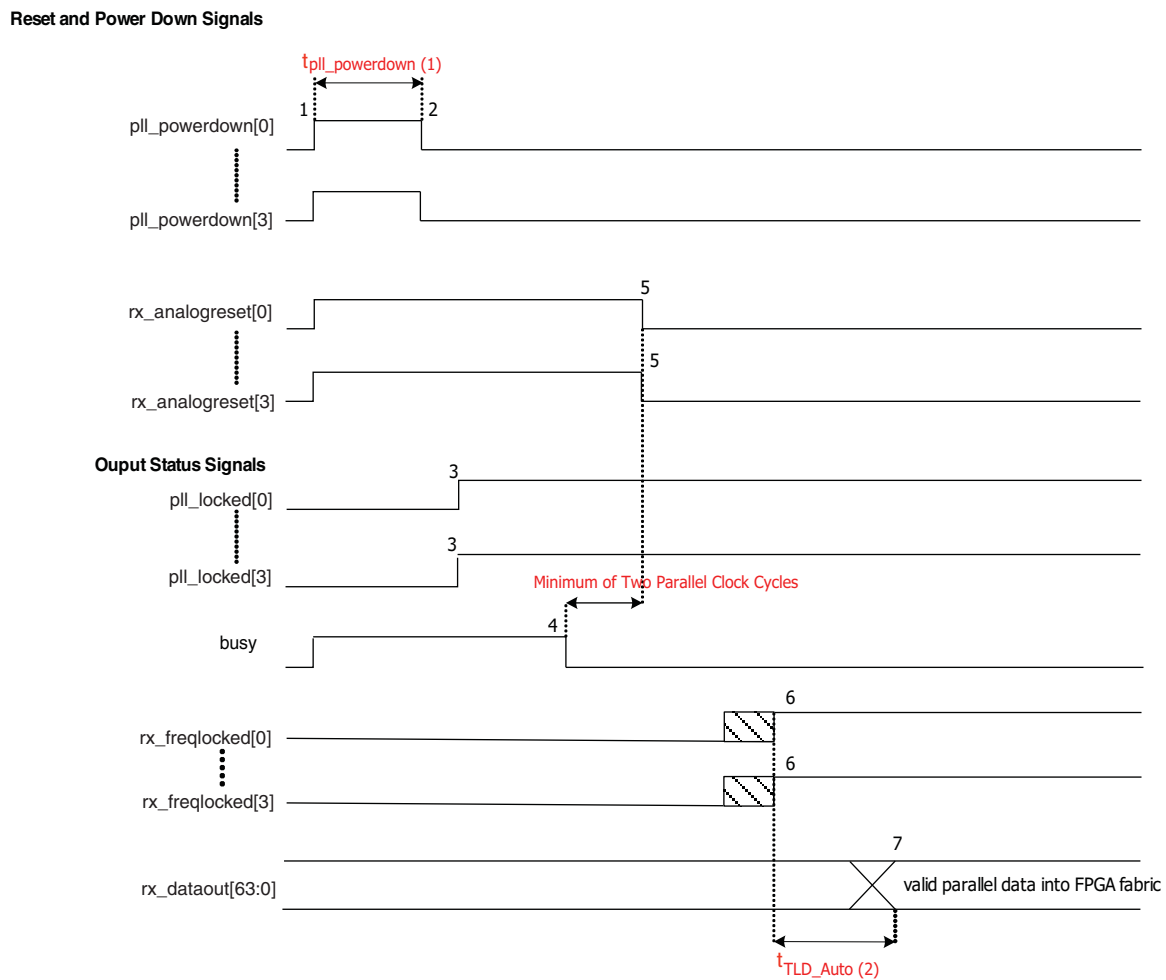
Table 4-8. Reset and Power-Down Sequences for Basic (PMA Direct) Drive ×1 Configurations

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 4-32
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 4-34

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 $\times 1$ 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 for Stratix IV Devices* chapter.
- (2) For t_{TLD_Auto} duration, refer to the *DC and Switching Characteristics for Stratix IV Devices* chapter.

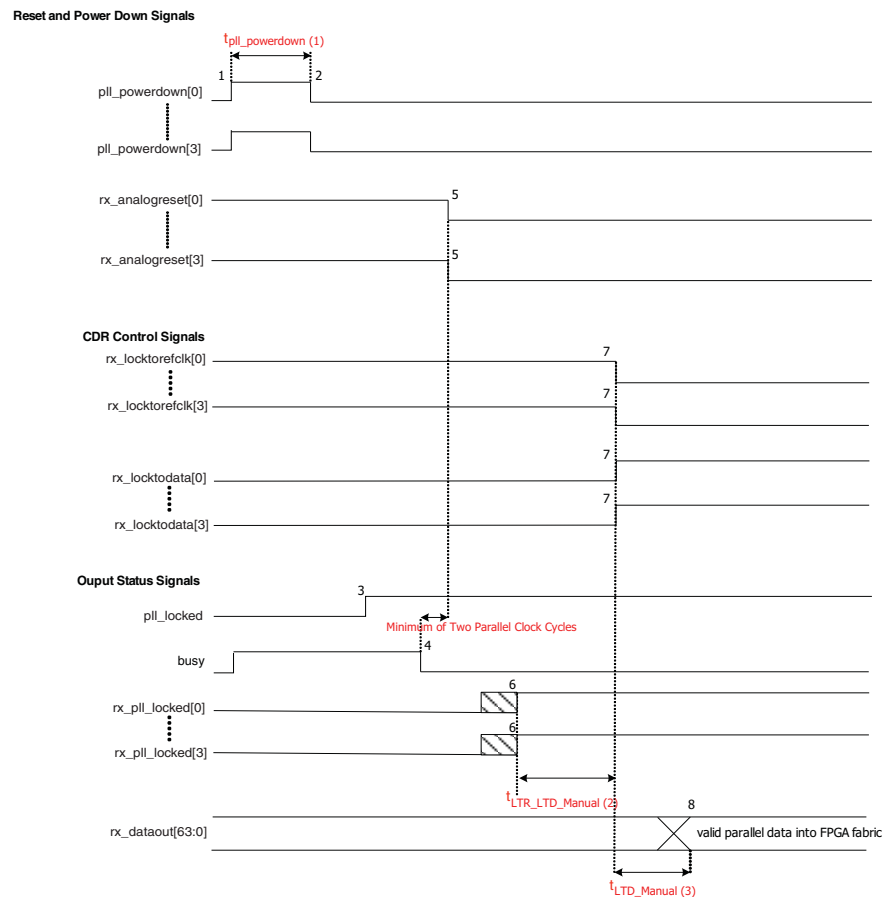
As shown in Figure 4-18, for the receiver and transmitter channel in Basic (PMA Direct) drive double-width configuration, non-bonded with CDR in automatic lock mode, follow these reset steps:

1. After power up, assert `p11_powerdown` of each channel for a minimum period of $t_{p11_powerdown}$ (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 `p11_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 `p11_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 a minimum of 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 $\times 1$ 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 $t_{\text{pll_powerdown}}$ duration, refer to the *DC and Switching Characteristics for Stratix IV Devices* chapter.
- (2) For $t_{\text{LTR_LTD_Manual}}$ duration, refer to the *DC and Switching Characteristics for Stratix IV Devices* chapter.
- (3) For $t_{\text{LTD_Manual}}$ duration, refer to the *DC and Switching Characteristics for Stratix IV Devices* chapter.

As shown in Figure 4-19, for the receiver and transmitter channel in Basic (PMA Direct) drive double-width configuration, non-bonded with CDR in manual lock mode, follow these reset steps:

1. After power up, assert `p11_powerdown` of each channel for a minimum period of $t_{p11_powerdown}$ (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 `p11_powerdown` signal, the transmitter PLL starts locking to the transmitter input reference clock.
3. When the transmitter PLL locks, as indicated by the `p11_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 a minimum of 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_p11_locked` signal from each channel to go high. The `rx_p11_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_p11_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).

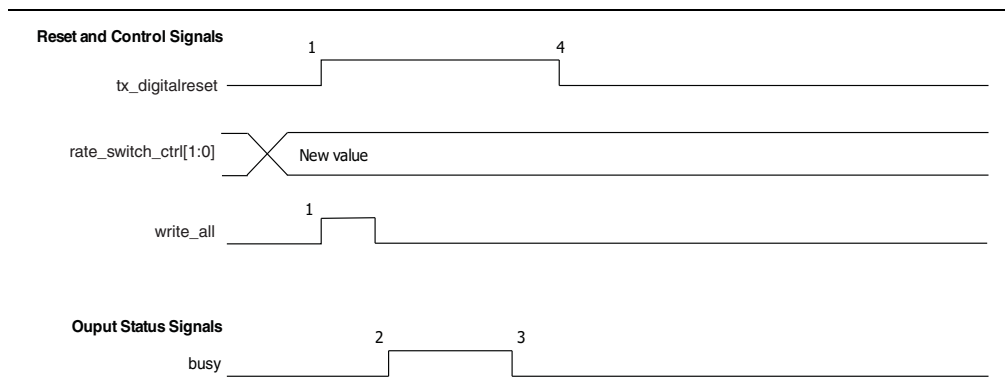
Dynamic Reconfiguration Reset Sequences

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 $\times 1$ 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](#), when using the dynamic reconfiguration controller to change the configuration of the transmitter channel, follow these reset steps:

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.

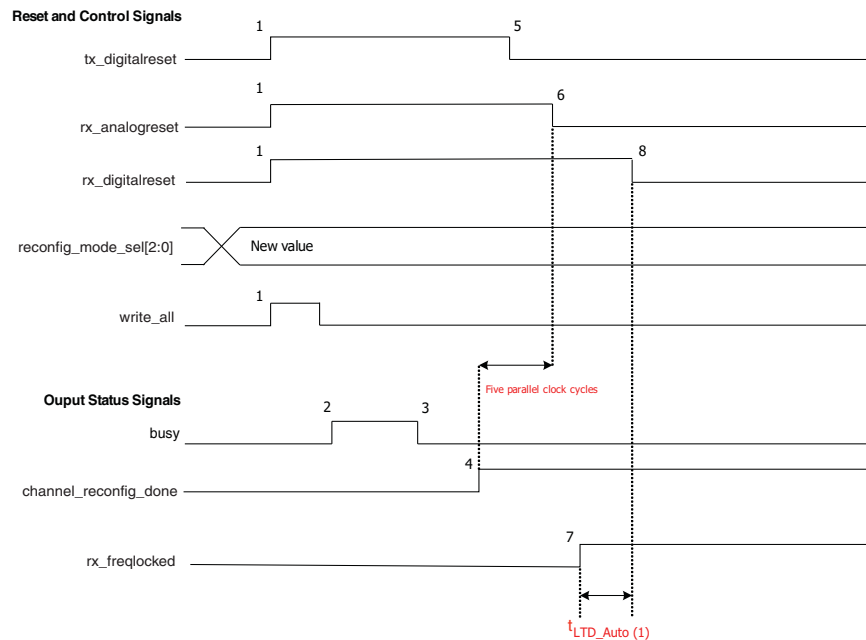
 For more information, refer to the *Dynamic Reconfiguration in Stratix IV Devices* chapter.

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 for Stratix IV Devices* chapter.

As shown in Figure 4–21, when using the dynamic reconfiguration controller to change the configuration of the transceiver channel, follow these reset steps:

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 *Dynamic Reconfiguration in Stratix IV Devices* 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.

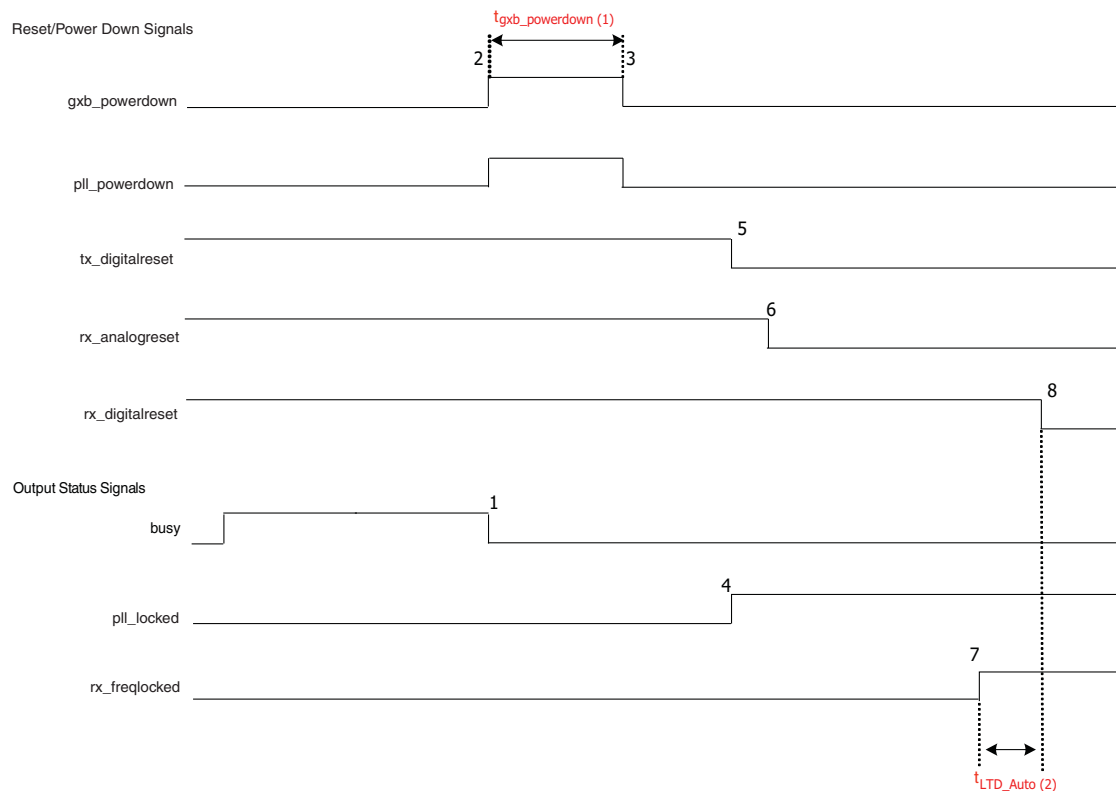
5. After assertion of the channel_reconfig_done signal, de-assert tx_digitalreset (marker 5) and wait for at least five parallel clock cycles to de-assert the rx_analogreset signal (marker 6).
6. Lastly, wait for the rx_freqlocked signal to go high. After rx_freqlocked goes high (marker 7), wait for 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 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 μ s. 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 μ s. 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 for Stratix IV Devices* chapter.
- (2) For t_{LTD_Auto} duration, refer to the *DC and Switching Characteristics for Stratix IV Devices* chapter.

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 `tLTD_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 PCIe mode simulation, you must assert the `tx_forceidle` 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 (Part 1 of 2)

Terms Used in this Chapter	Useful Reference Points
Automatic Lock Mode	page 4-8
Basic (PMA Direct) Drive x1 Mode	page 4-31
Basic (PMA Direct) Drive xN Mode	page 4-25
Bonded channel configuration	page 4-6
<code>busy</code>	page 4-3
Dynamic Reconfiguration Reset Sequences	page 4-36
<code>gxb_powerdown</code>	page 4-3
LTD	page 4-6
LTR	page 4-6
Manual Lock Mode	page 4-10
Non-Bonded channel configuration	page 4-15
PCIe	page 4-22
<code>pll_locked</code>	page 4-3
<code>pll_powerdown</code>	page 4-3
<code>rx_analogreset</code>	page 4-2
<code>rx_digitalreset</code>	page 4-2
<code>rx_freqlocked</code>	page 4-3

Table 4-9. Reference Information (Part 2 of 2)

Terms Used in this Chapter	Useful Reference Points
rx_pll_locked	page 4-3
tx_digitalreset	page 4-2

Document Revision History

Table 4-10 lists the revision history for this chapter.

Table 4-10. Document Revision History

Date	Version	Changes
January 2014	4.4	Updated Figure 4-4.
September 2012	4.3	Updated Table 4-2 to close FB #65274.
December 2011	4.2	Updated Table 4-2.
February 2010	4.1	<ul style="list-style-type: none"> ■ Updated the “Receiver and Transmitter Channel—Receiver CDR in Automatic Lock Mode”, “Receiver and Transmitter Channel—Receiver CDR in Manual Lock Mode”, “Receiver and Transmitter Channel—Receiver CDR in Automatic Lock Mode”, “Receiver and Transmitter Channel—Receiver CDR in Manual Lock Mode”, “Receiver and Transmitter Channel—Receiver CDR in Automatic Lock Mode”, “Receiver and Transmitter Channel—Receiver CDR in Manual Lock Mode”, “Receiver and Transmitter Channel Set-up—Receiver CDR in Automatic Lock Mode”, “Receiver and Transmitter Channel Set-up—Receiver CDR in Manual Lock Mode” ■ Updated Figure 4-4, Figure 4-5, Figure 4-6, Figure 4-7, Figure 4-16, Figure 4-17, Figure 4-18, and Figure 4-19. ■ Updated Table 4-2. ■ Updated chapter title. ■ Applied new template. ■ Minor text edits.
November 2009	4.0	<ul style="list-style-type: none"> ■ 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 <i>DC and Switching Characteristics</i> chapter in the <i>Stratix IV Device Datasheet</i> section. ■ 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.
June 2009	3.1	<ul style="list-style-type: none"> ■ 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	3.0	Added: <ul style="list-style-type: none"> ■ “PMA Direct Drive Mode Reset Sequences” ■ “Dynamic Reconfiguration Reset Sequences”
November 2008	2.0	Added chapter to the <i>Stratix IV Device Handbook</i> .

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 5–91
- “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	AEQ control logic is soft IP that you can enable in the dynamic reconfiguration controller.
AEQ Hardware	AEQ hardware is circuitry that you can enable in the receiver portion of the transceivers.

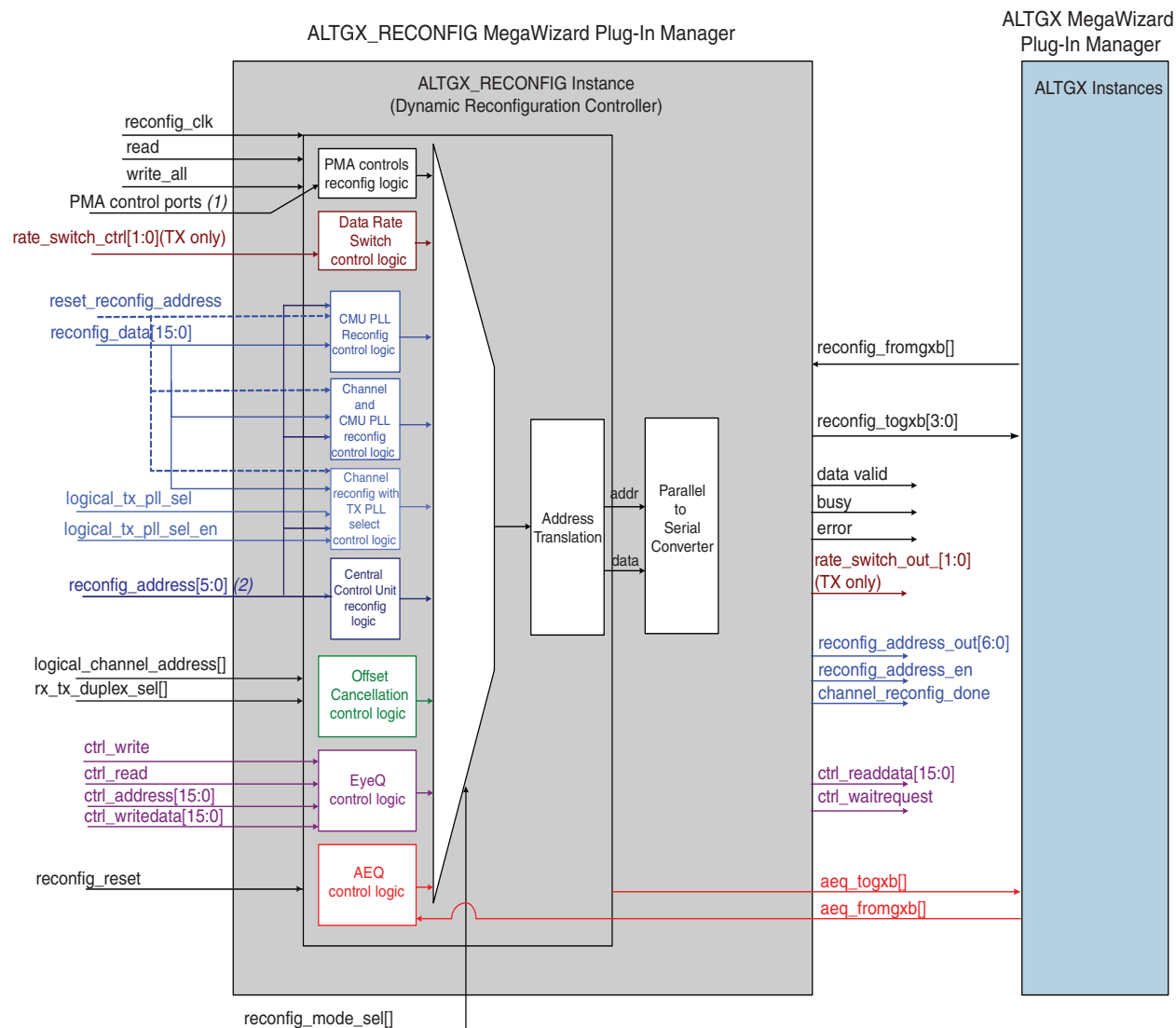
Table 5-1. Glossary of Terms Used in this Chapter (Part 2 of 2)

Term	Description
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.
Alternate CMU Transmitter PLL	Refers to one of the two CMU PLLs within a transceiver block.
Channel and Transmitter PLL Select/reconfig Mode	Refers to the following dynamic reconfiguration modes: <ul style="list-style-type: none"> ■ 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 <code>logical_channel_address</code> 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



Notes to Figure 5-1:

- (1) The PMA control ports consist of the V_{OD} , pre-emphasis, DC gain, and manual equalization controls.
- (2) For more information, refer to Table 5-16 on page 5-78.

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 the dynamic reconfiguration settings for the transceiver instances.



For more information, refer to the “Reconfiguration Settings” section of the *ALTGX Transceiver Setup Guide for Stratix IV Devices* 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
<code>reconfig_clk</code>	Receiver only and Transmitter only
<code>reconfig_clk</code>	Receiver and Transmitter
<code>fixedclk</code>	PCI Express® (PCIe)

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 driving the `reconfig_clk` signal on a global clock resource. 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 5-3. 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 PCIe	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 PCIe 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 *Stratix IV 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 the **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

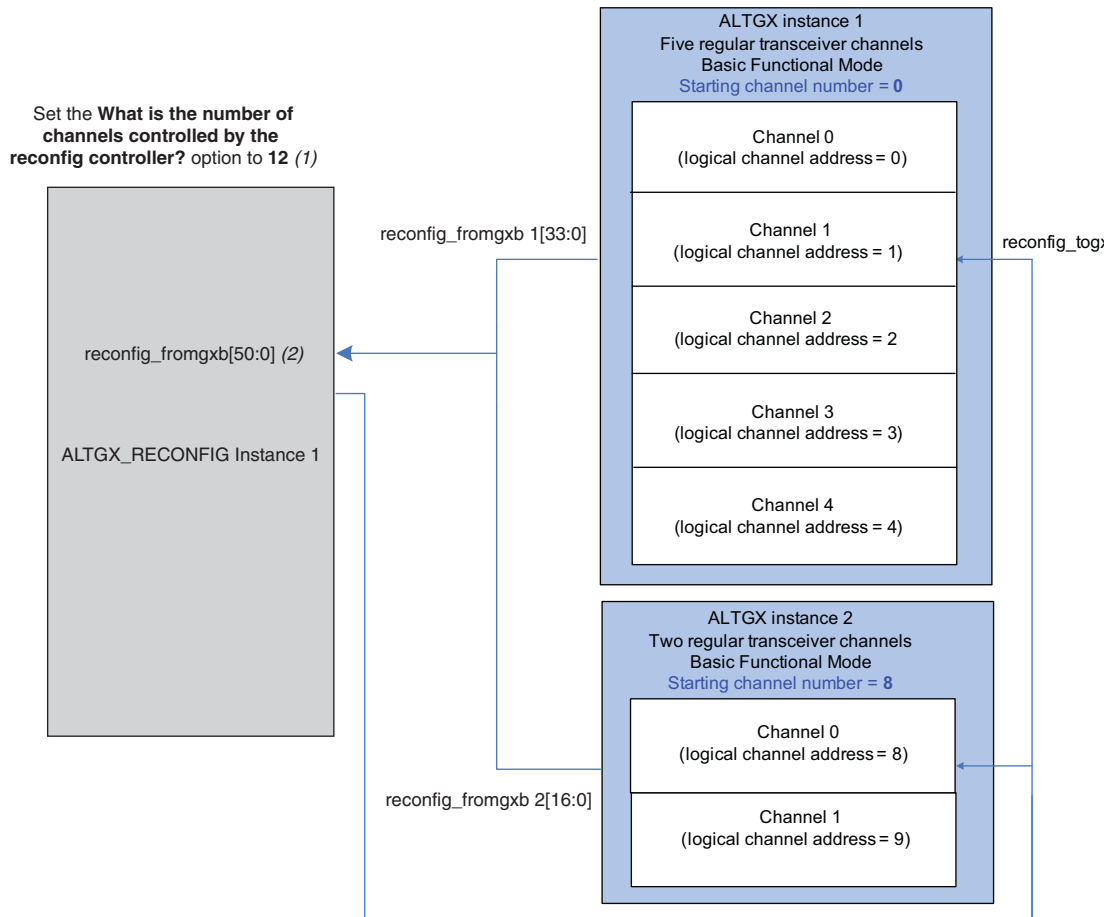
Logical Channel Addressing of Regular Transceiver Channels

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.


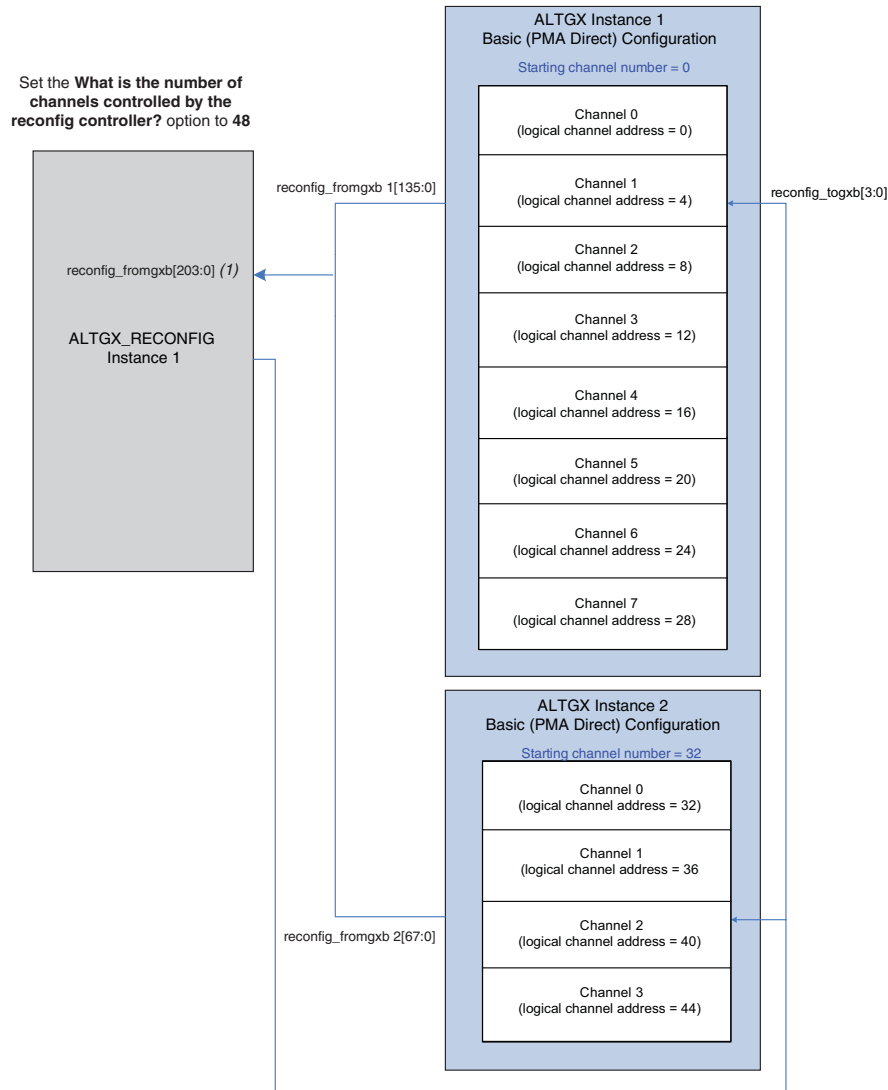
 When PMA-only channel reconfiguration involves a transmitter PLL, you also must account for the logical channel address of the PLL used. If there are four channels in Basic [PMA Direct] $\times N$ functional mode, each channel requires a logical channel address (**0, 4, 8, 12**), and the transmitter PLL used requires an address (**16**).

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]}`.

For example, if you a transceiver configuration with Instance 1 (Inst1) with two channels that use Basic (PMA Direct) mode. Instance 2 (Inst2) has two channels and uses Basic mode, which uses the PCS block in the transceiver. In order to leave the needed gap in the reconfiguration controller signals, follow these steps:

1. Set Inst1 to have a starting channel number **0**. The logical channel addresses 0 and 4 are then allocated to the two channels of Inst1.
2. Reserve logical channel address 8—do not use this address.
3. Set Inst2 to have a starting channel number **12**.

4. Set the **Number of channels** option in the ALTGX_RECONFIG controller megafunction to the nearest multiple of four from the highest logical channel address. In this example, set the **Number of channels** option to **16**.
5. When connecting the reconfig_fromgxb bus of the ALTGX_RECONFIG controller, connect the bits corresponding to the reserved address to 0. In this example, the ALTGX_RECONFIG controller provides 4*(16:0) bits.
6. Connect the reconfig_fromgxb bus of the ALTGX_RECONIF controller and the ALTGX transceiver instances as follows:
 - a. reconfig_fromgxb[33:0] of the ALTGX_RECONFIG instance - reconfig_fromgxb[33:0] of Inst1.
 - b. reconfig_fromgxb[50:34] of the ALTGX_RECONFIG instance - 17'h00000
 - c. reconfig_fromgxb[67:51] of the ALTGX_RECONFIG instance - reconfig_fromgxb[17:0] of Inst2

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 5-41 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 “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.

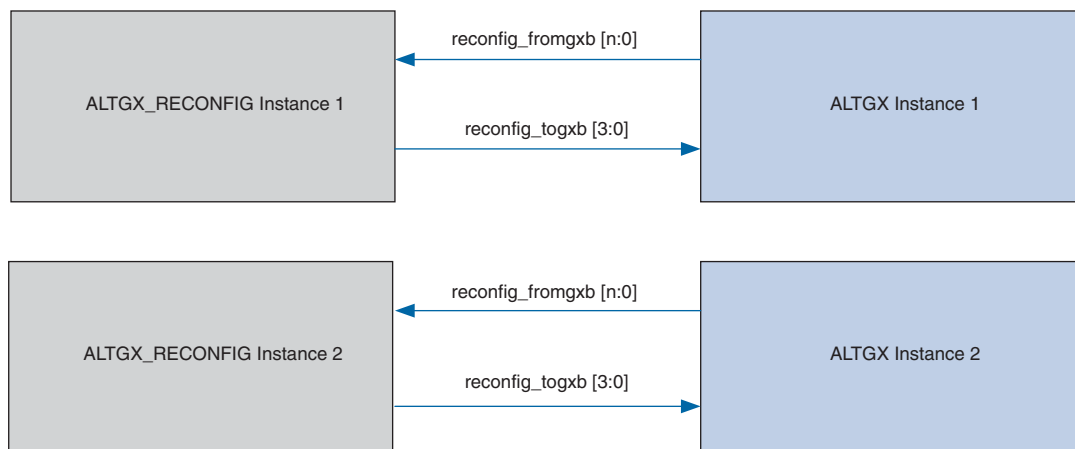
For more information, refer to [“Example 1”](#) on page 5-96.

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 4 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 5-67
- “EyeQ” on page 5-69
- “Adaptive Equalization (AEQ)” on page 5-75
- “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:

- Pre-emphasis settings
- DC gain settings
- Voltage output differential (V_{OD}) settings
- Equalization settings (channel reconfiguration mode does not support equalization settings)

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”](#).
- 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.

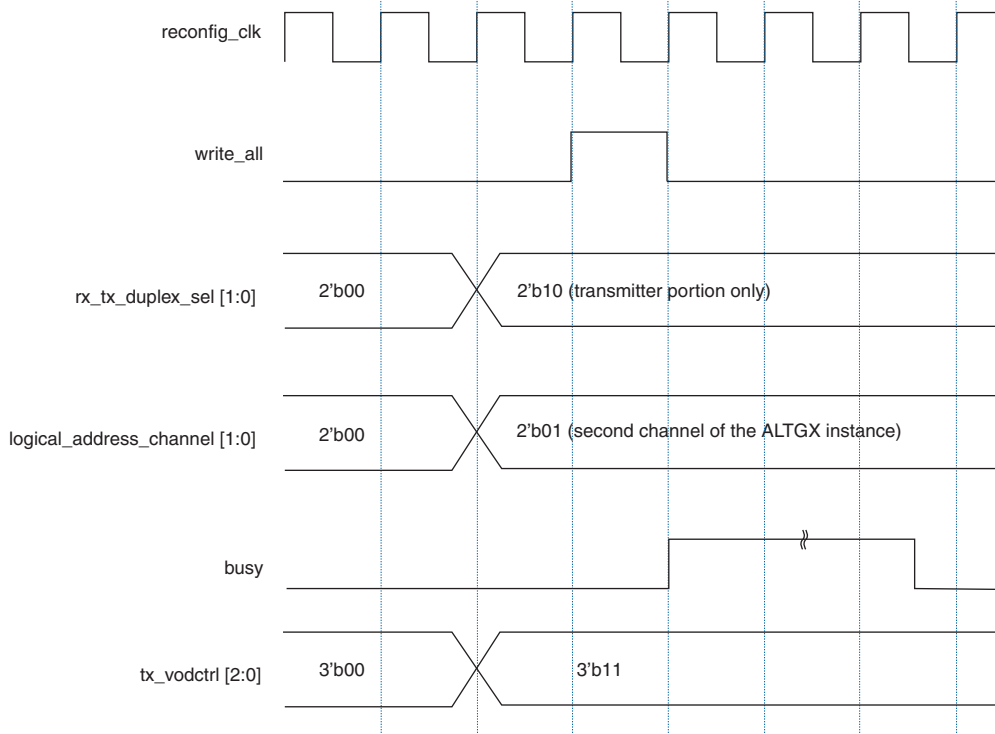
The value you set at the PMA control ports is only written into the specified transceiver channel.

 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.

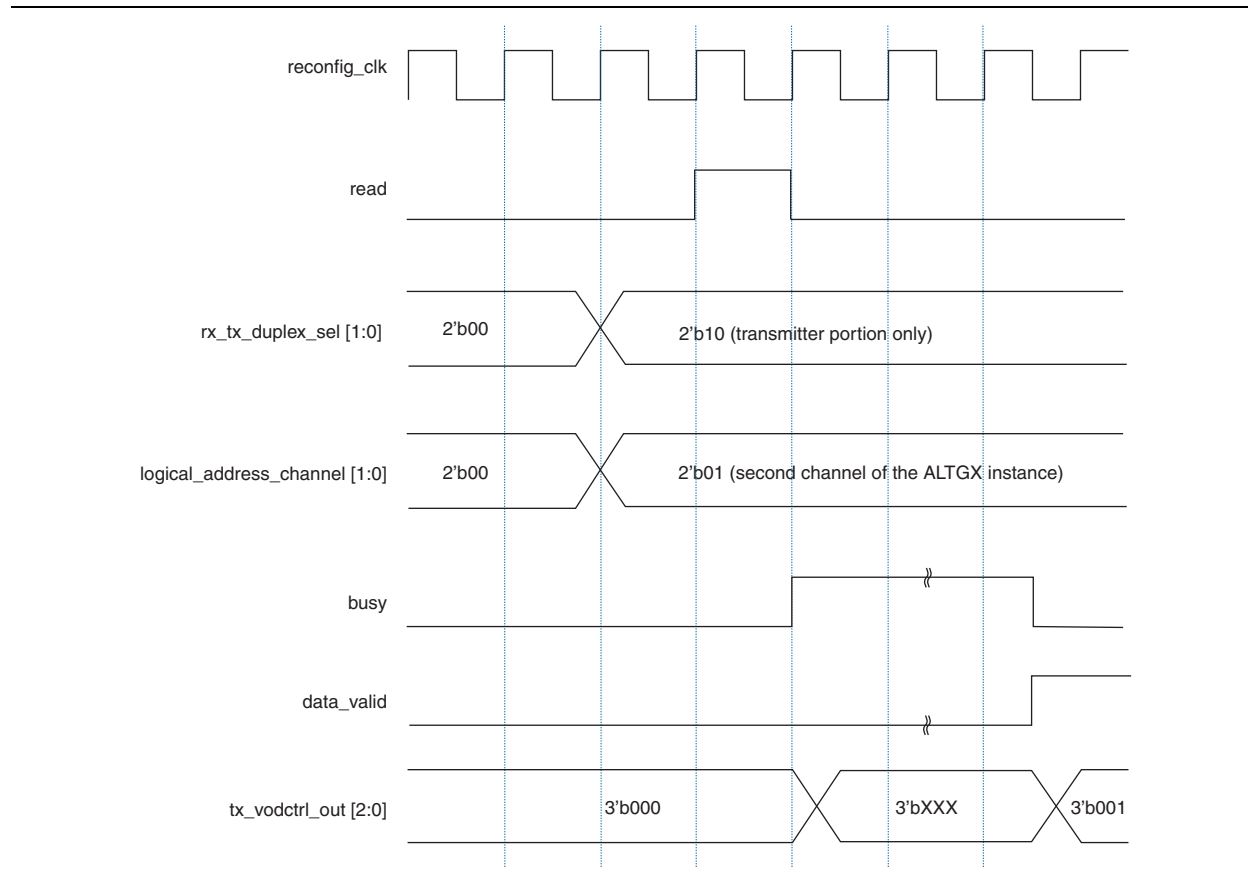
Figure 5-5. Method 1—Write Transaction Waveform




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.

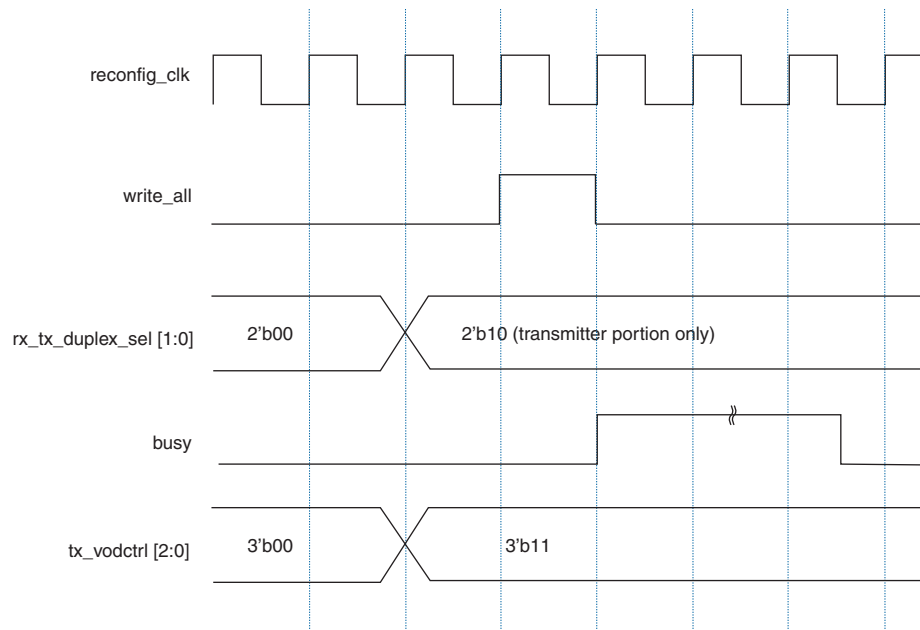
Using Method 2, you can write the same PMA control value into all the transceiver channels connected to the dynamic reconfiguration controller.

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



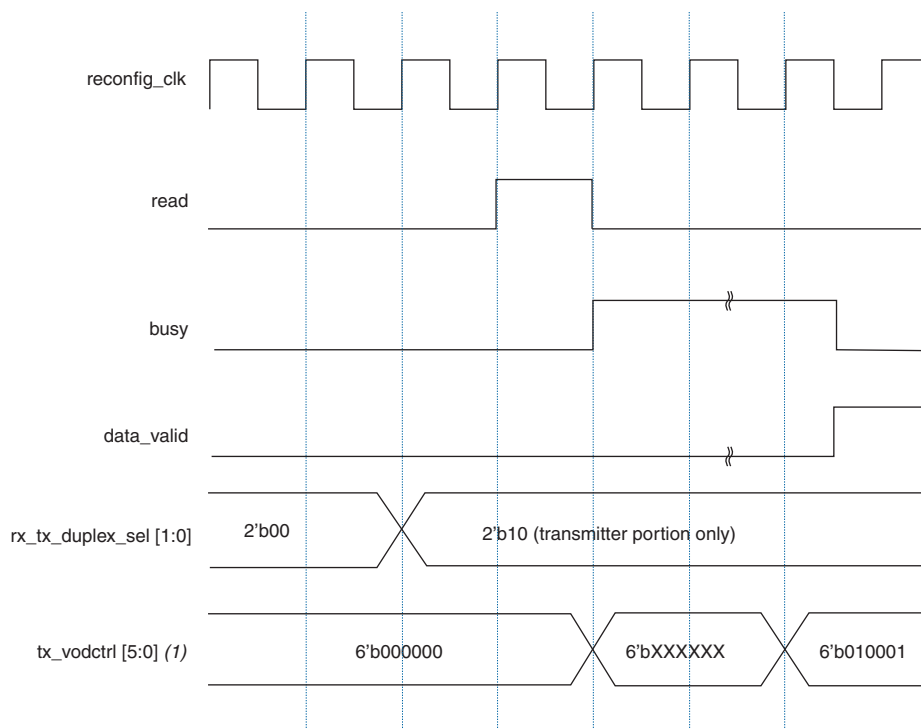
Read Transaction

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_vodctrl1_out [5:3]`.



Simultaneous write and read transactions are not allowed.

Method 3—Using Individual Control Signals for Each Channel

You can optionally use 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:

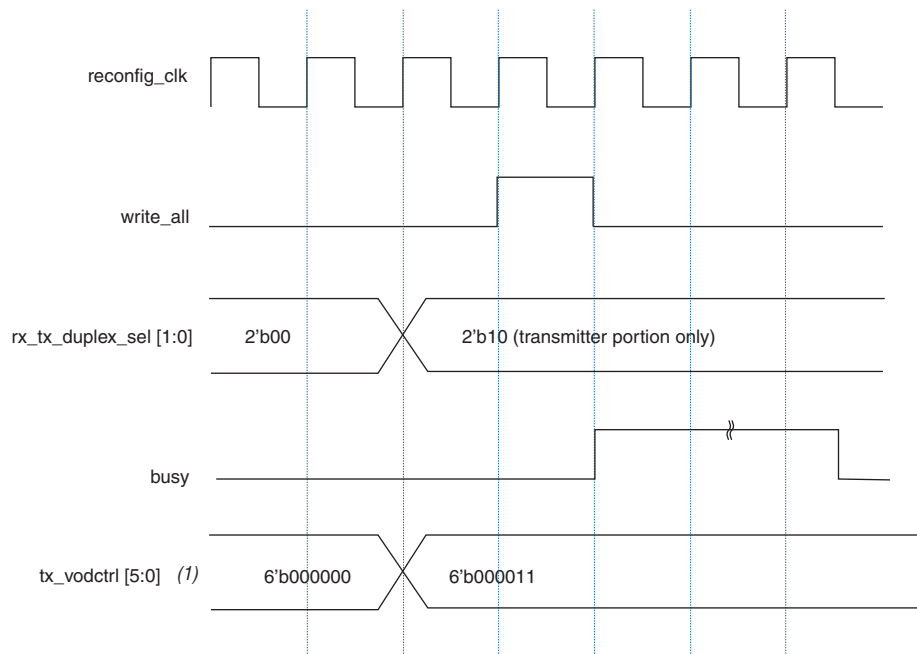
1. Retain the stored values of the other active channels using a read transaction.
2. Set the new value at the bits corresponding to the specific channel.
3. Perform a write transaction.

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 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	Y
	Basic (PMA Direct) ×1 configuration	Basic (PMA Direct) ×1 configuration	Y
	Basic (PMA Direct) ×N configuration	Basic (PMA Direct) ×N configuration	Y
Channel reconfiguration with transmitter PLL select	Non-bonded configurations of regular transceiver channels	Non-bonded configurations of regular transceiver channels	Y
	Basic (PMA Direct) ×1 configuration	Basic (PMA Direct) ×1 configuration	Y
	Basic (PMA Direct) ×N configuration	Basic (PMA Direct) ×N configuration	Y
Central control unit reconfiguration ⁽²⁾	×4 bonded mode	×4 bonded mode	Y
	×8 bonded mode	×8 bonded mode	Y
Data rate division in transmitter	All Transmitter only configurations of regular transceiver channels	All Transmitter only configurations of regular transceiver channels ⁽¹⁾	—

Note to Table 5-5:

- (1) Because the transmitter local divider is not available for bonded mode channels, data rate division is supported for non-bonded channels only.
- (2) Dynamic reconfiguration from a bonded mode with rate matcher to another bonded mode without rate matcher is not allowed.



You cannot dynamically reconfigure from Deterministic Latency mode to any other functional mode and vice-versa. Within Deterministic Latency mode, the following reconfigurations are not allowed:

- Phase Compensation FIFO register mode and a non-register mode
- PFD feedback mode and a non-PFD feedback mode

For instance, you can dynamically reconfigure the data rate for CPRI mode. However, you cannot dynamically reconfigure from CPRI mode to a non-CPRI mode.

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:

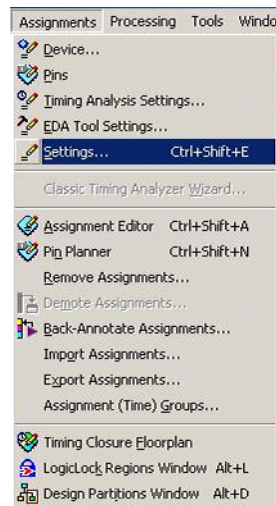
- How to generate a .mif?—The Quartus® II software generates .mifs when you provide the appropriate project settings and then compiles an ALTGX instance. For more information, refer to [“Quartus II Settings to Enable .mif Generation”](#) on [page 5-20](#).
- How is a .mif used between the ALTGX_RECONFIG 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:

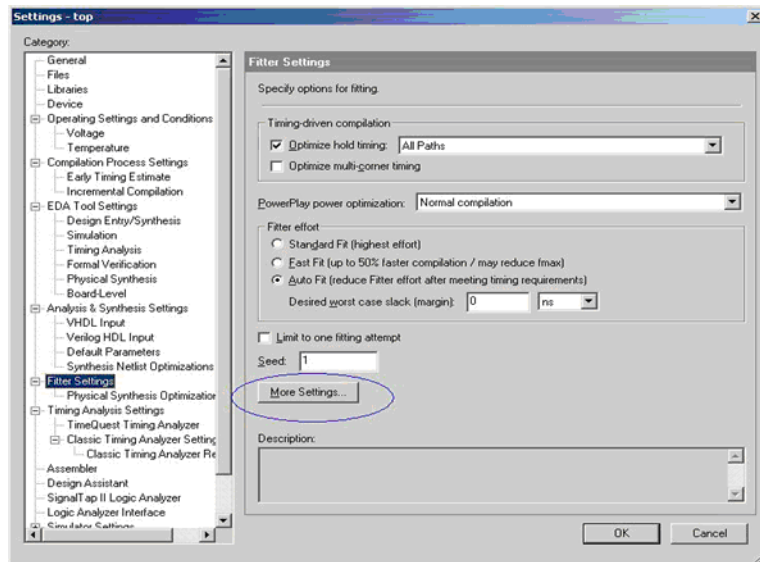
1. On the Assignments menu, select **Settings** ([Figure 5-10](#)).

Figure 5-10. Step 1 to Enable .mif Generation



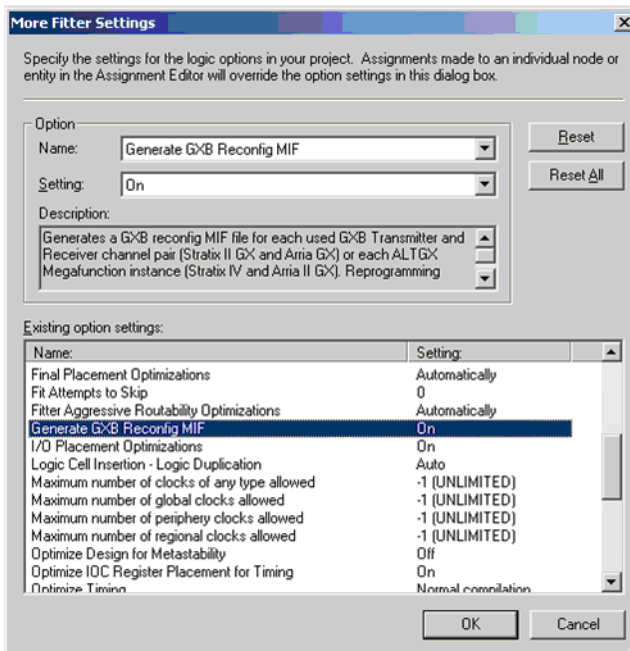
2. Select **Fitter settings**, then choose **More Settings** (Figure 5-11).

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)	PMA Direct Mode
Duplex (Receiver and Transmitter) + Central control unit	60	33
Duplex (Receiver and Transmitter)	55	28
Receiver only	38	14
Transmitter only	19	15

Note to Table 5-6:

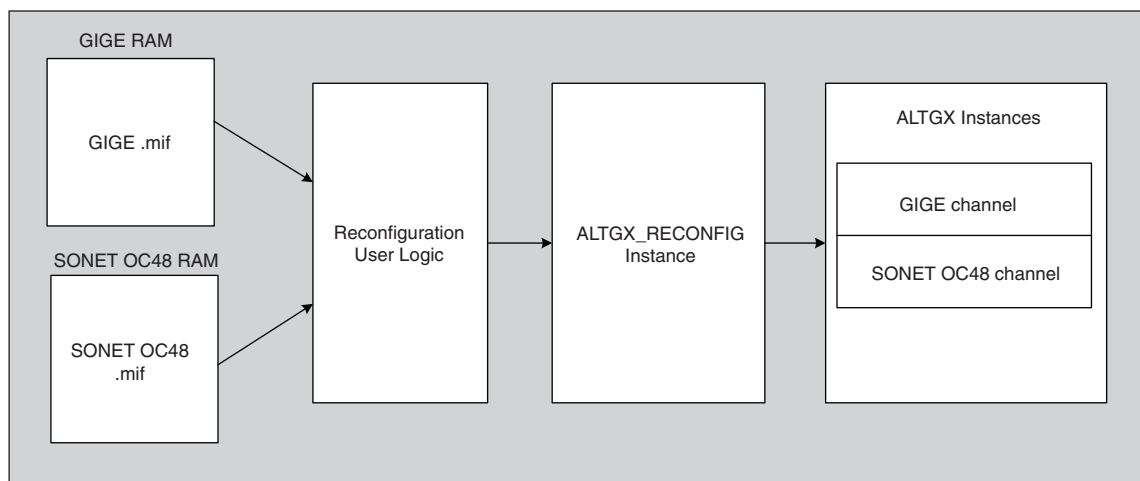
(1) Each word in the **.mif** is 16 bits wide.

You can store these **.mifs** in 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 are set to the default settings chosen in the ALTGX instance used for **.mif** generation.
-  The equalization settings of the receiver cannot be modified by a **.mif**.

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 auxiliary transmit (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
<ul style="list-style-type: none"> ■ 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. 	<ul style="list-style-type: none"> ■ 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.

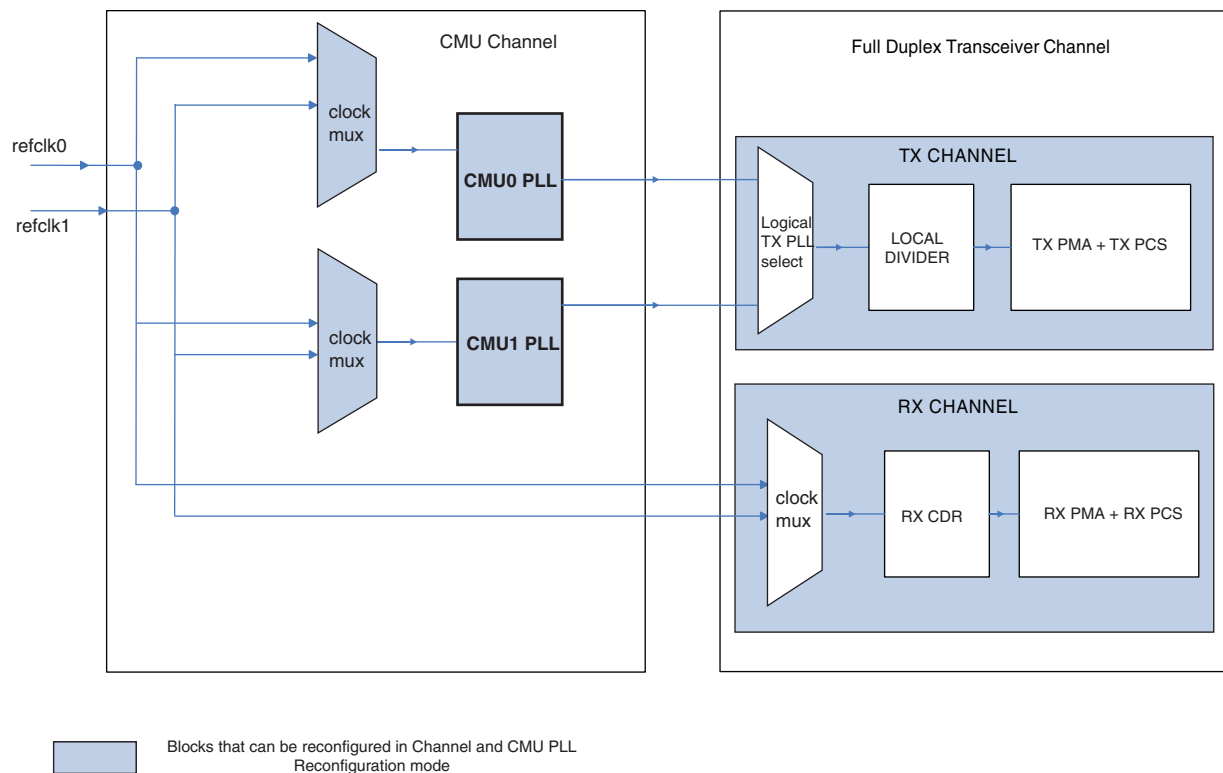
-  In addition to the categories mentioned, you can also choose to reconfigure both the data rate and functional mode of a transceiver channel.
-  For the following sections, assume that the transceiver channel has the **Receiver and Transmitter** configuration in the ALTGX MegaWizard Plug-In Manager, 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 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.

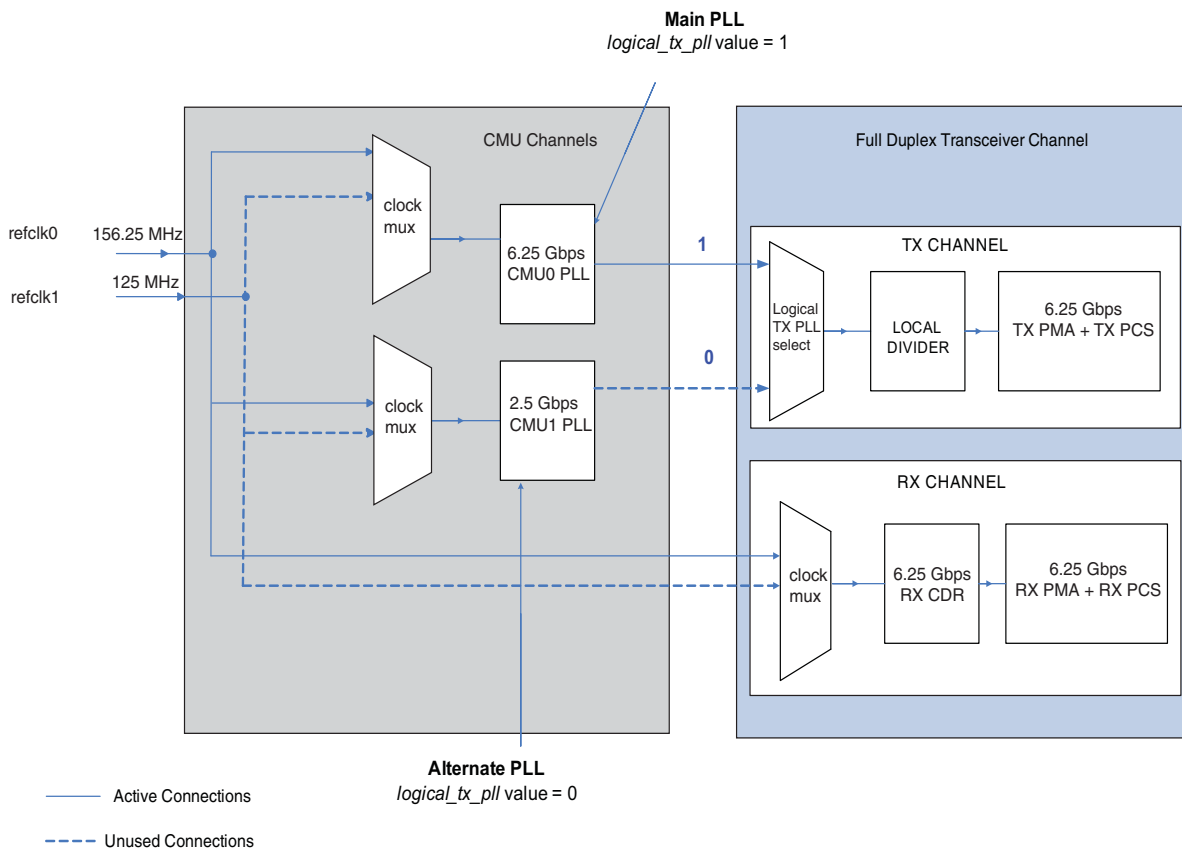
5. Provide the starting channel number in the **Modes** screen. For more information, refer to “[Logical Channel Addressing](#)” on page 5-5.
6. Provide the logical reference index of the CMU PLL in the **What is the PLL logical reference index?** option in the corresponding PLL screen. For more information, 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 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**.

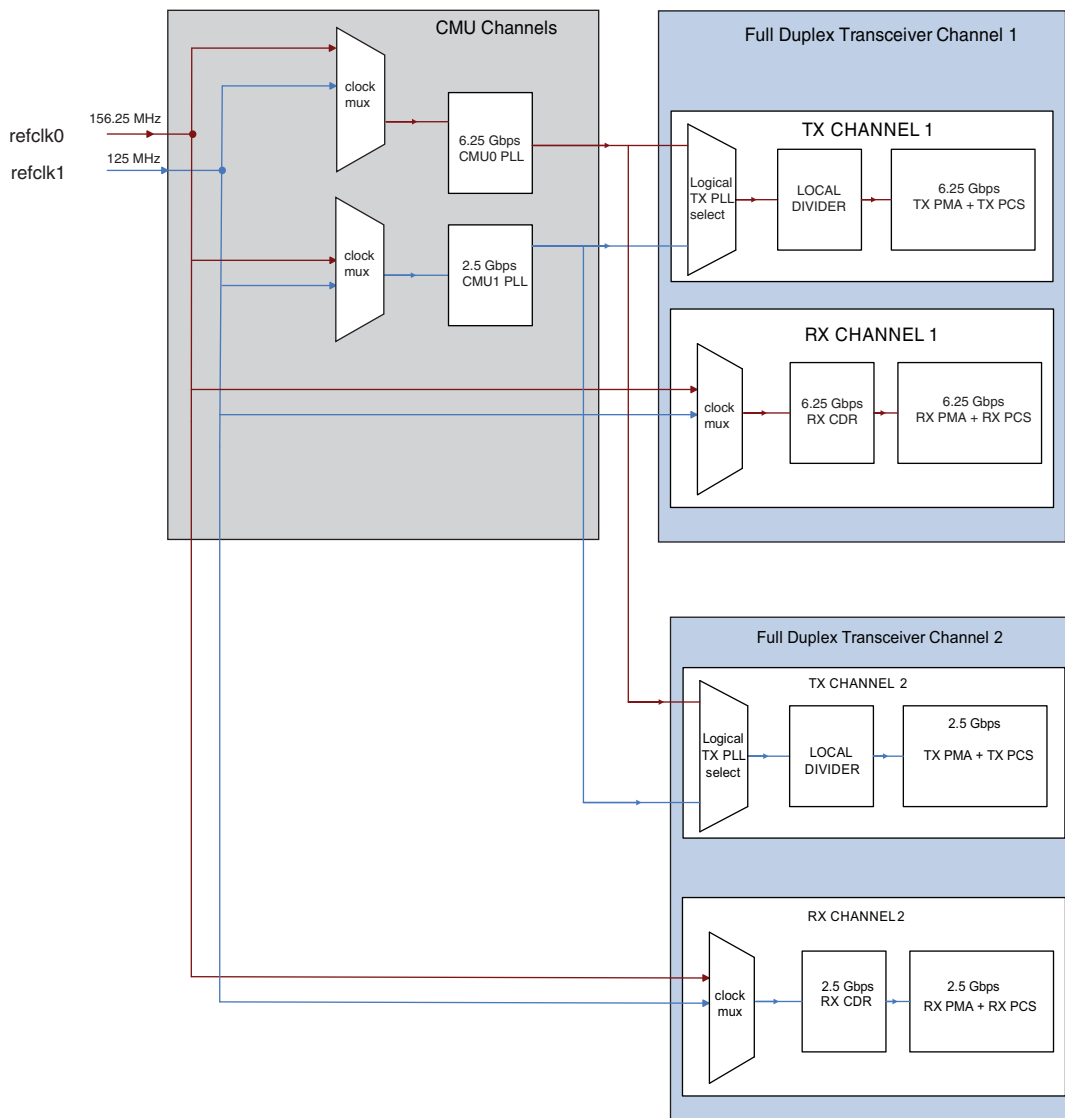


The main PLL corresponds to the CMU PLL configuration set in the **General** screen of the ALTGX MegaWizard Plug-In Manager. The alternate PLL corresponds to the CMU PLL configuration set in the **Alt PLL** screen.

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 (1)



Note to Figure 5-16:

(1) After the device powers up, the `busy` signal remains low for the first `reconfig_clk` cycle.

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 the CMU0 PLL within a transceiver block is always the complement of the logical reference index of the 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 (TX) phase compensation FIFO and the receive (RX) 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 TX 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 TX 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.

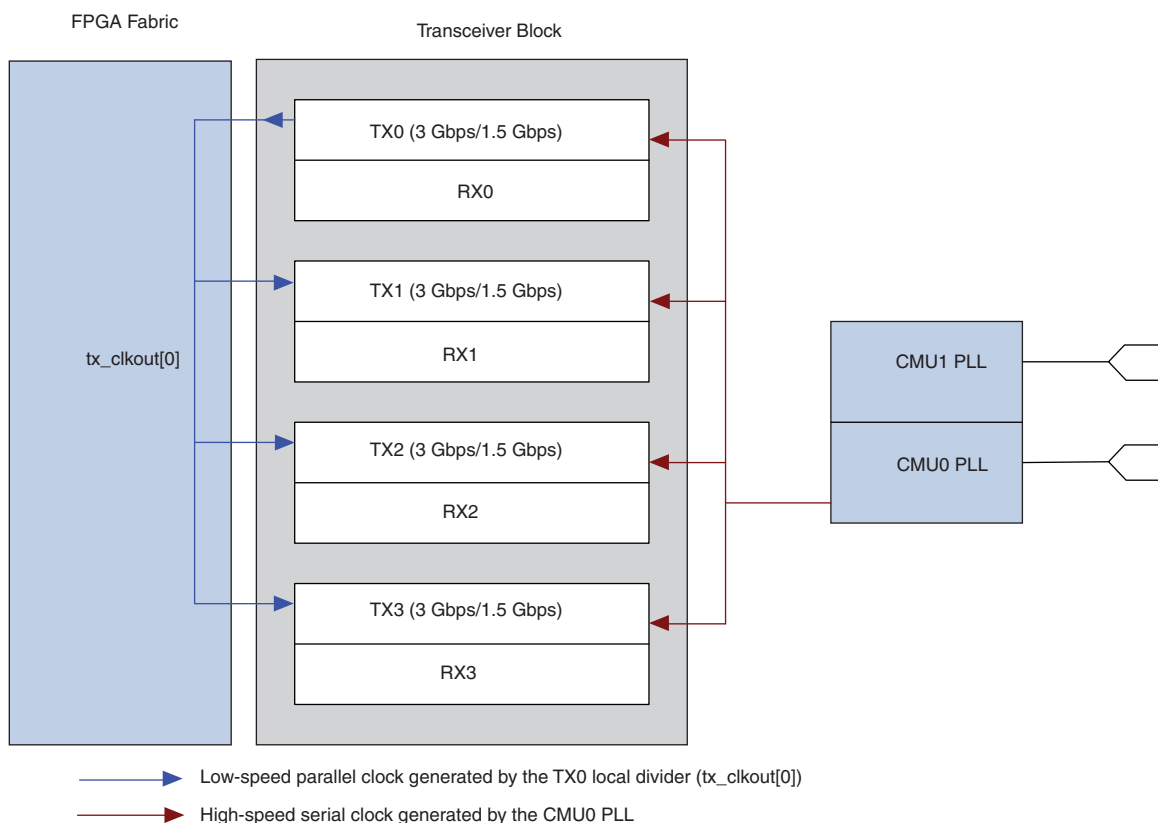
Consider the following scenario:

- 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.

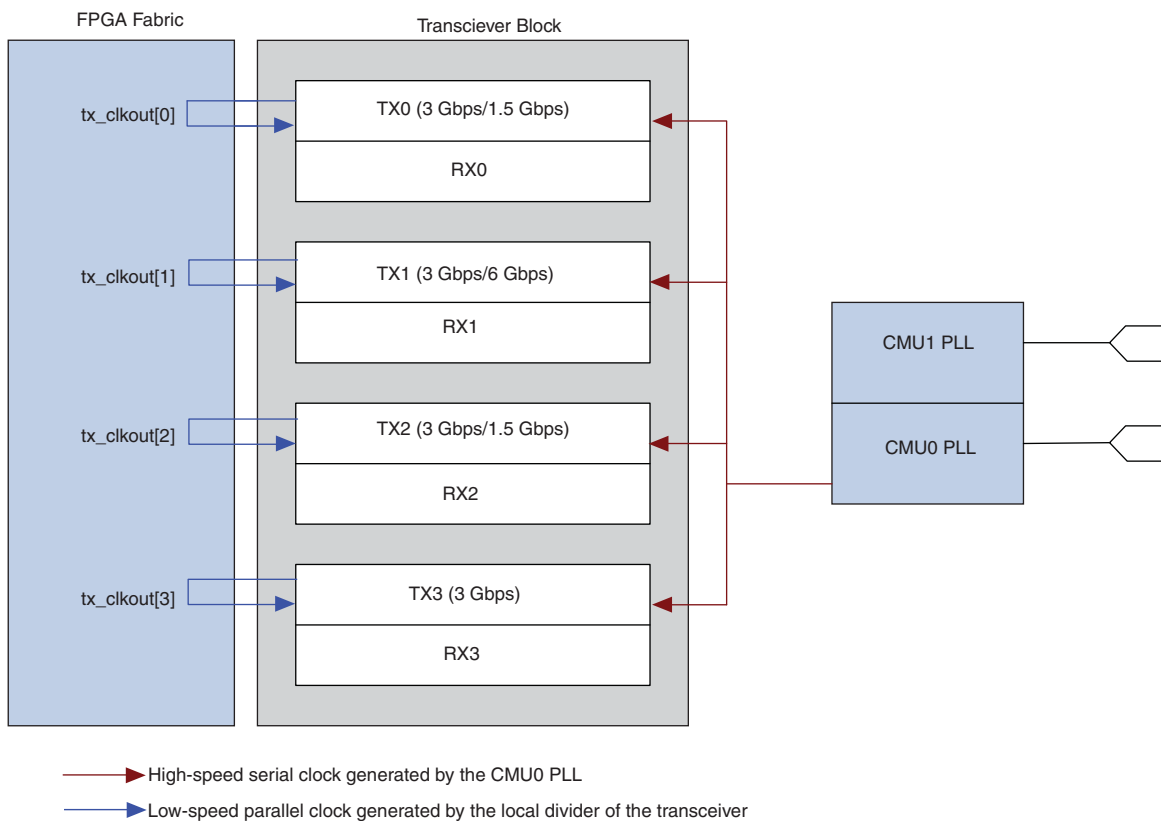
Consider the following scenario:

- 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.

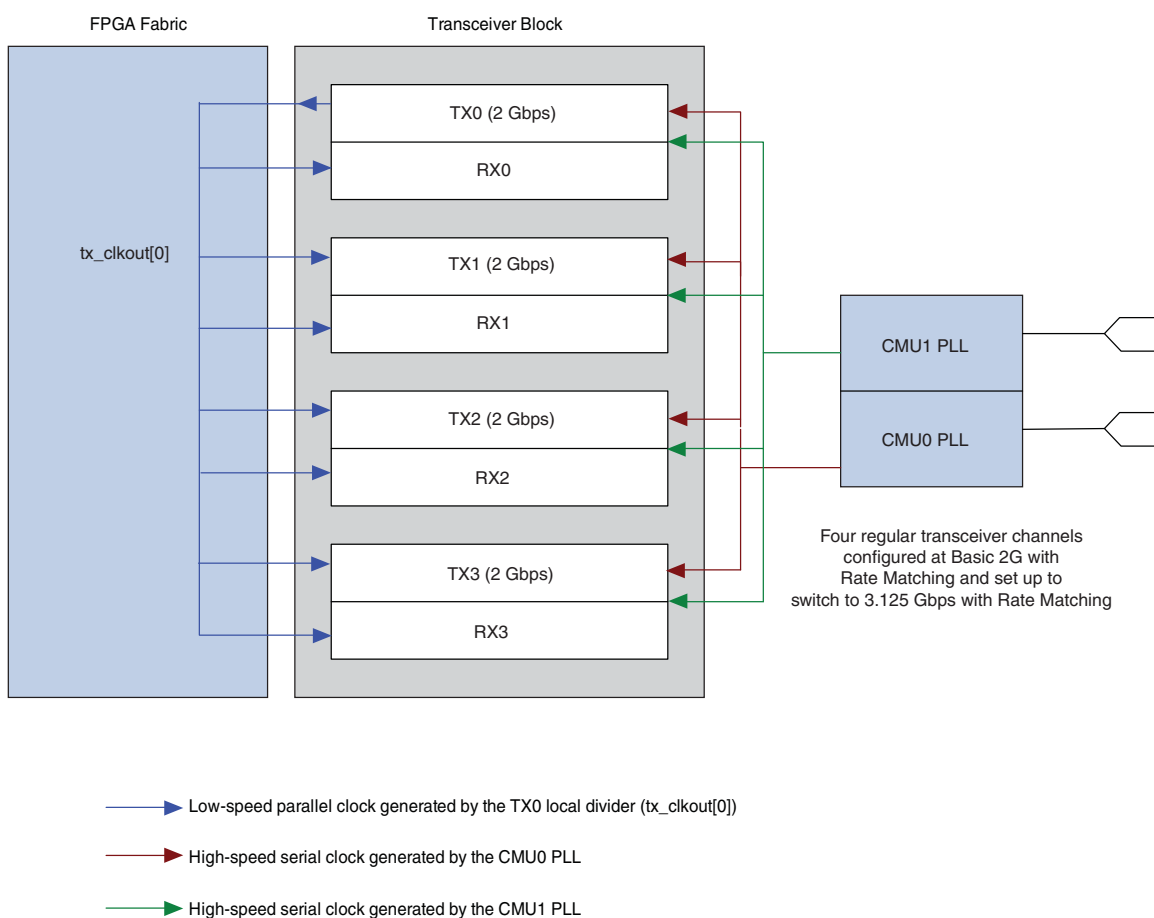
Consider the following scenario:

- Four regular transceiver channels configured to the 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.

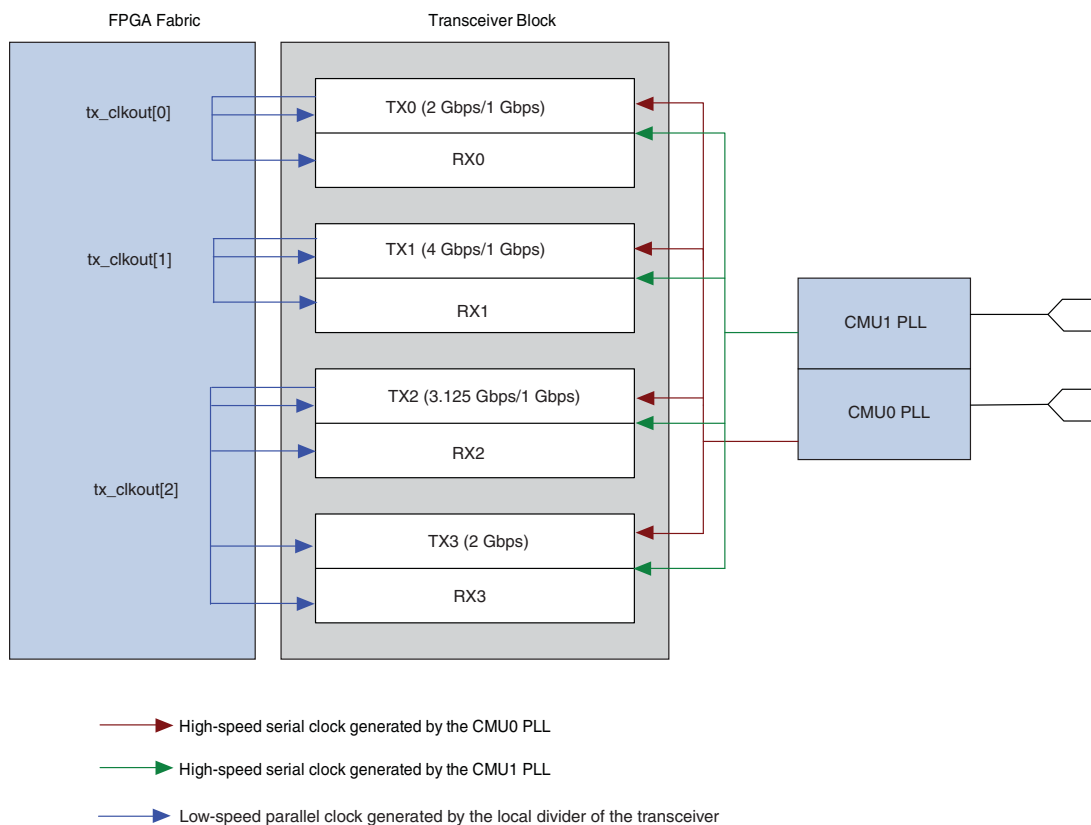
Consider the following scenario:

- TX0/RX0: You want to dynamically reconfigure the Basic 1 Gbps configuration with rate matching enabled to the Basic 2 Gbps configuration with rate matching enabled.
- TX1/RX1: You want to dynamically reconfigure the Basic 4 Gbps configuration with rate matching enabled to the Basic 1 Gbps configuration with rate matching enabled.
- TX2/RX2 and TX3/RX3: You want to dynamically reconfigure the Basic 3.125 Gbps configuration with rate matching enabled to the 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.

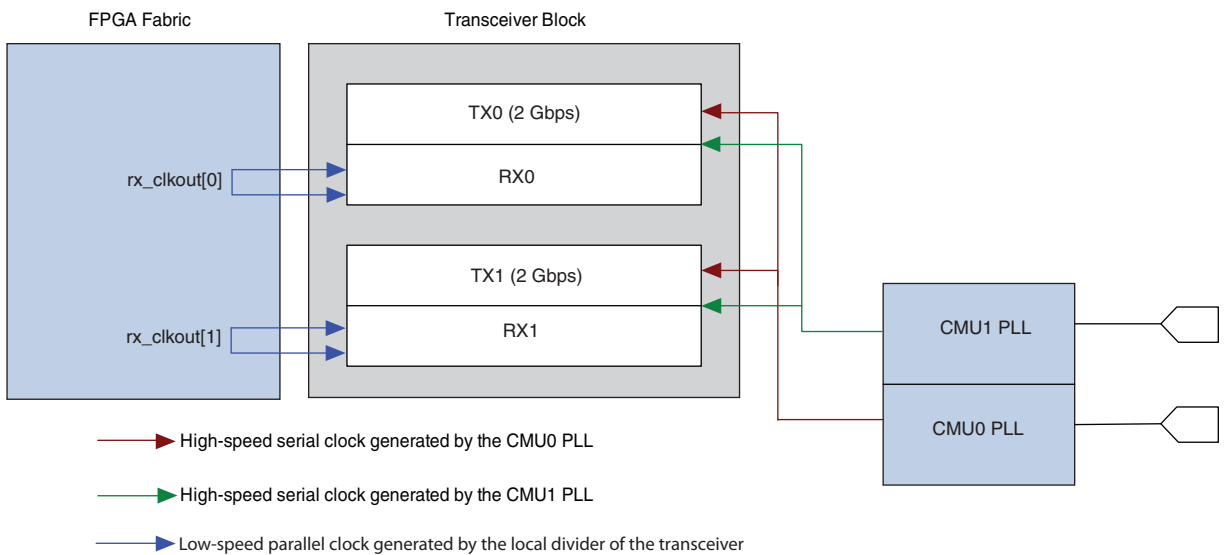
Consider the following scenario:

- 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 must 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

- enables and disables 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 channel has:


- changed the FPGA fabric-transceiver channel interface data width


OR

- changed the 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 *Transceiver Architecture in Stratix IV Devices* 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_ala2sizeout	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 PCIe functional mode. Table 5-9 lists 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 3) ⁽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 (PCIe) (tx_forcedisp) in all modes except PCIe. For PCIe mode, (tx_forcedispcompliance) is used.
	tx_datainfull [10]: Forced disparity value (tx_dispval)
10-bit FPGA fabric-transceiver Channel Interface	For Non-PIPE: tx_datainfull [10]: Forced disparity value (tx_dispval) For PCIe: tx_datainfull [10]: Forced electrical idle (tx_forceelecidle)
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)

Table 5-9. tx_datainfull[43:0] FPGA Fabric-Transceiver Channel Interface Signal Descriptions (Part 2 of 3) ⁽¹⁾

FPGA Fabric-Transceiver Channel Interface Description	Transmit 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 (tx_datain) tx_datainfull[7:0] - tx_datain (LSByte) and tx_datainfull[29:22] - tx_datain (MSByte)
	<p>The following signals are used only in 8B/10B modes:</p>
	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 PCIe: tx_datainfull[9] - tx_forcedispcompliance and tx_datainfull[31] - 0
	Force Disparity Value For non-PIPE: tx_datainfull[10]:tx_dispval (LSB) and tx_datainfull[32] -tx_dispval (MSB) For PCIe: tx_datainfull[10] - tx_forceelecidle and tx_datainfull[32] - tx_forceelecidle
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)

Table 5-9. tx_datainfull[43:0] FPGA Fabric-Transceiver Channel Interface Signal Descriptions (Part 3 of 3) ⁽¹⁾

FPGA Fabric-Transceiver Channel Interface Description	Transmit Signal Description (Based on Stratix IV GX Supported FPGA Fabric-Transceiver Channel Interface Widths)
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 *Transceiver Architecture for Stratix IV Devices* chapter.

Table 5-10 lists 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 6)

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_ctrlldetect)
	rx_dataoutfull[9]: Code violation status signal (rx_errrdetect)
	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-PCIe/PCIe modes.
	rx_dataoutfull[14]: Rate Match FIFO insertion status indicator (rx_rmfifodatainserted) in non-PCIe/PCIe modes.
	rx_dataoutfull[14:13]: non-PCIe/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_ala2sizeout
	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-PCIe/PCIe modes
	rx_dataoutfull[14]: Rate Match FIFO insertion status indicator (rx_rmfifodatainserted) in non-PCIe/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 6)

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-PCIe/PCIe modes
	rx_dataoutfull[14] and rx_dataoutfull[46]: Rate Match FIFO insertion status indicator (rx_rmfifodatainserted) in non-PCIe/PCIe modes
	Two 2-bit PCIe 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 6)

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_rmfiodeleted) in non-PCIe/PCIe modes
	rx_dataoutfull[14] and rx_dataoutfull[46]: Rate Match FIFO insertion status indicator (rx_rmfiodeinserted) in non-PCIe/PCIe modes
	Two 2-bit PCIe 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)
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)	

Table 5-10. rx_dataoutfull[63:0] FPGA Fabric-Transceiver Channel Interface Signal Descriptions (Part 4 of 6)

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 20 bits	Two 10-bit Data (rx_dataout) rx_dataoutfull[9:0] - rx_dataout (LSByte) and rx_dataoutfull[25:16] - rx_dataout (MSByte)
	Two 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-PCIe/PCIe modes
	rx_dataoutfull[14] and rx_dataoutfull[30]: Rate Match FIFO insertion status indicator (rx_rmfifodatainserted) in non-PCIe/PCIe modes
	rx_dataoutfull[15] and rx_dataoutfull[31]: 8B/10B running disparity indicator (rx_runningdisp)
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-PCIe/PCIe modes
	rx_dataoutfull[14] and rx_dataoutfull[46]: Rate Match FIFO insertion status indicator (rx_rmfifodatainserted) in non-PCIe/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 5 of 6)

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-PCIe/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-PCIe/PCIe modes

Table 5-10. rx_dataoutfull[63:0] FPGA Fabric-Transceiver Channel Interface Signal Descriptions (Part 6 of 6)

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, follow these 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]`

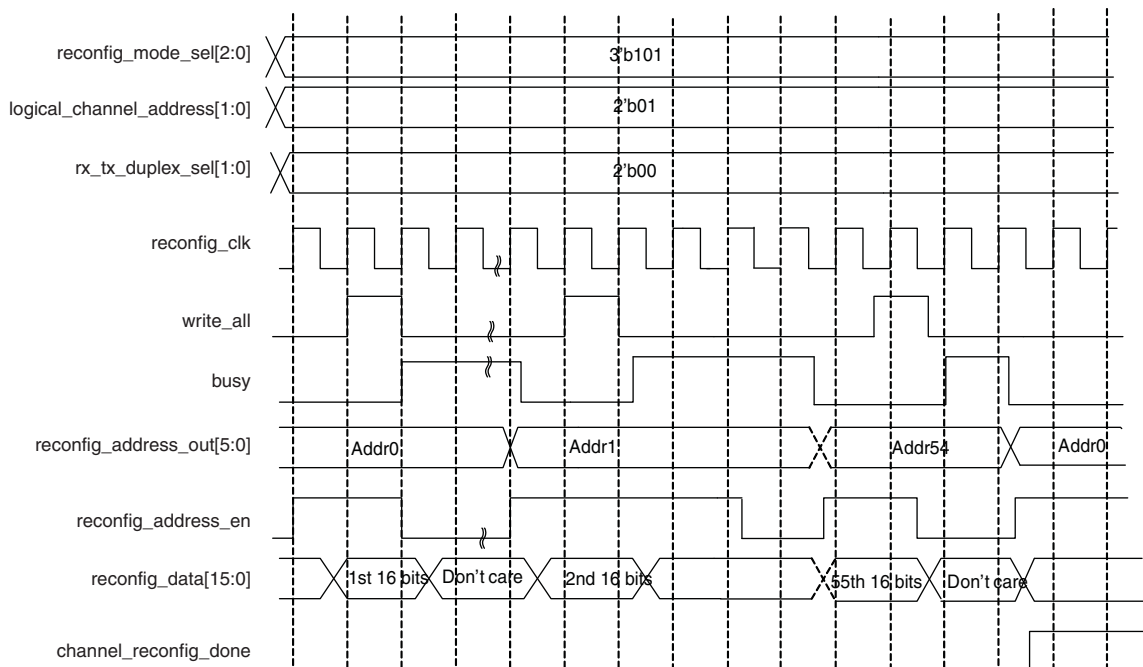
Channel and CMU PLL Reconfiguration Operation

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 in Stratix IV Devices* 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 5-20.



Channel reconfiguration with transmitter PLL select mode is not applicable to regular transceiver channels in $\times 4$ and $\times 8$ bonded mode configurations.

For guidelines regarding re-using `.mifs`, specifying input reference clocks, or using the `logical_tx_pll_sel` ports, refer to “[Special Guidelines](#)” on page 5-57.



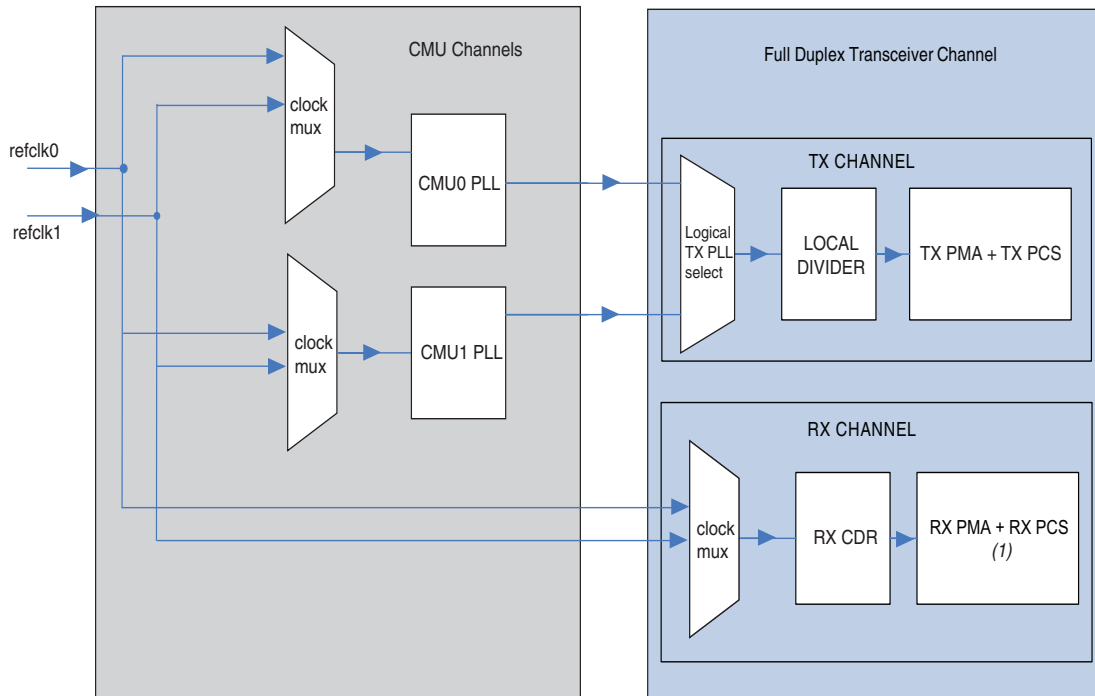
For more information about reset, refer to the “Reset Sequence when Using Dynamic Reconfiguration with the Channel and TX PLL select/reconfig Option” section in the *Reset Control and Power Down in Stratix IV Devices* chapter.

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

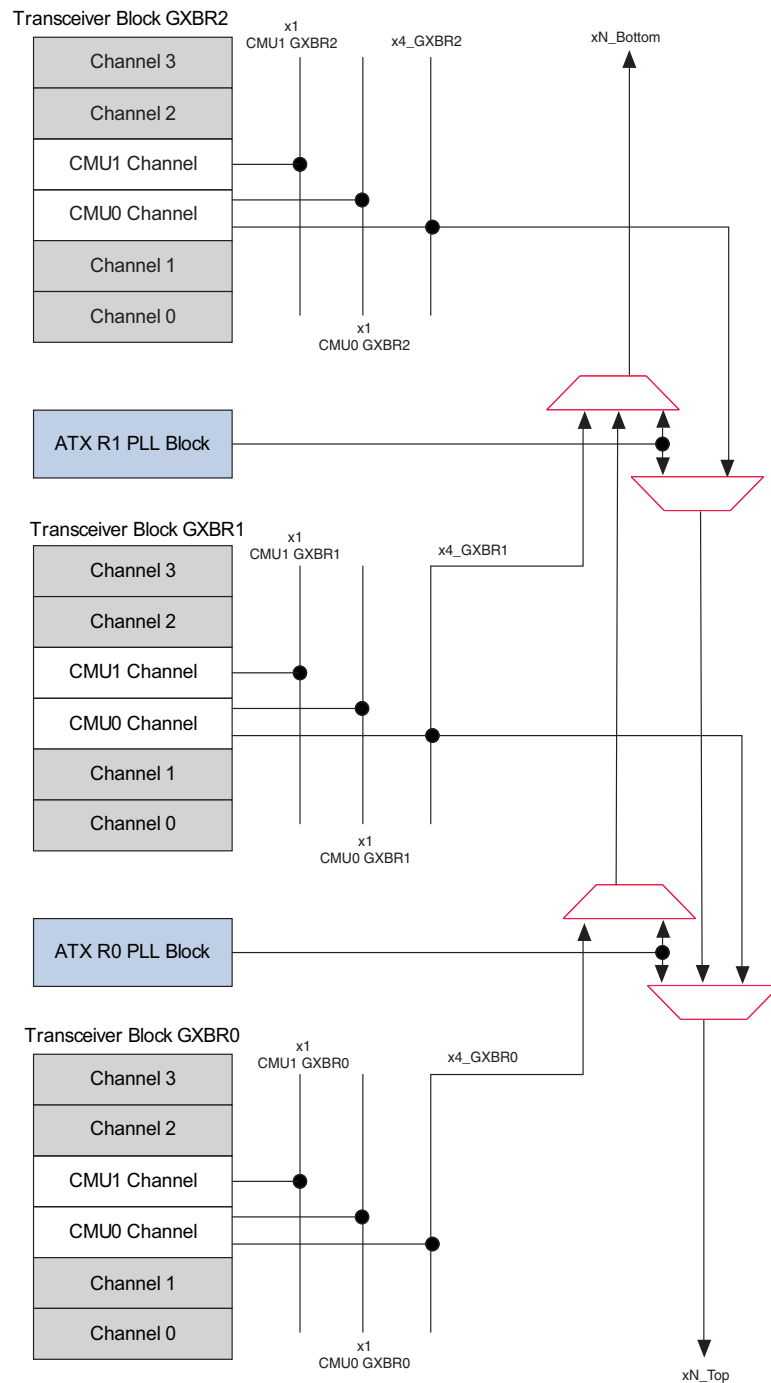


Note to Figure 5-23:

(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 $\times 4/\times N$ lines** option.

If you enable this option, the transmitter PLL is outside the transceiver block. Similarly, if you disable option, 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 $\times 4/\times N$ 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 $\times 4/\times N$ lines** option for an additional PLL, you can only select between 2 or 3 as the PLL logical reference index.
- When you disable the **Use central clock divider to drive the transmitter channels using $\times 4/\times N$ 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

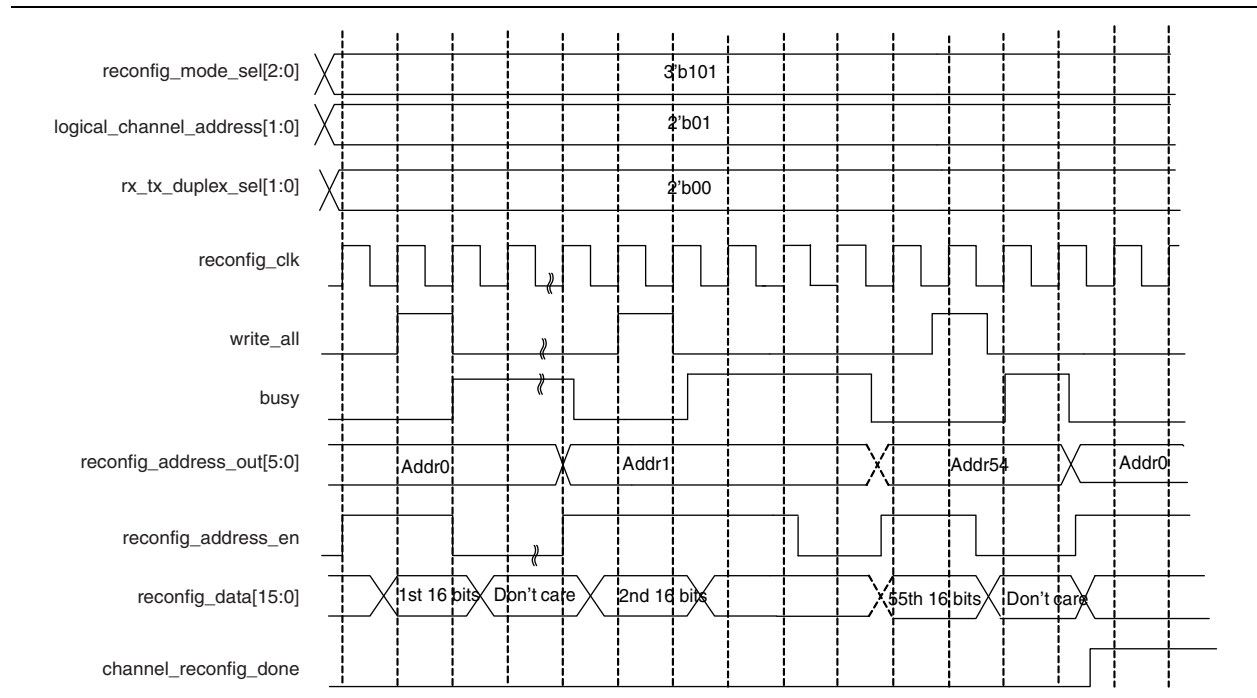
For more information, refer to the [“ALTGX_RECONFIG MegaWizard Plug-In Manager Setup for Channel and CMU PLL Reconfiguration Mode” on page 5-47](#).

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 for 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 in Stratix IV Devices](#) 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 must set location assignments using a central clock divider at location CMU0 PLL to place the CMU PLL that drives a transceiver channel if:

- The transceiver channel is in bonded mode configuration
- The **Use central clock divider to drive the transmitter channels using X4/XN lines** option on the Main PLL page of the **Reconfiguration Settings** tab is on



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

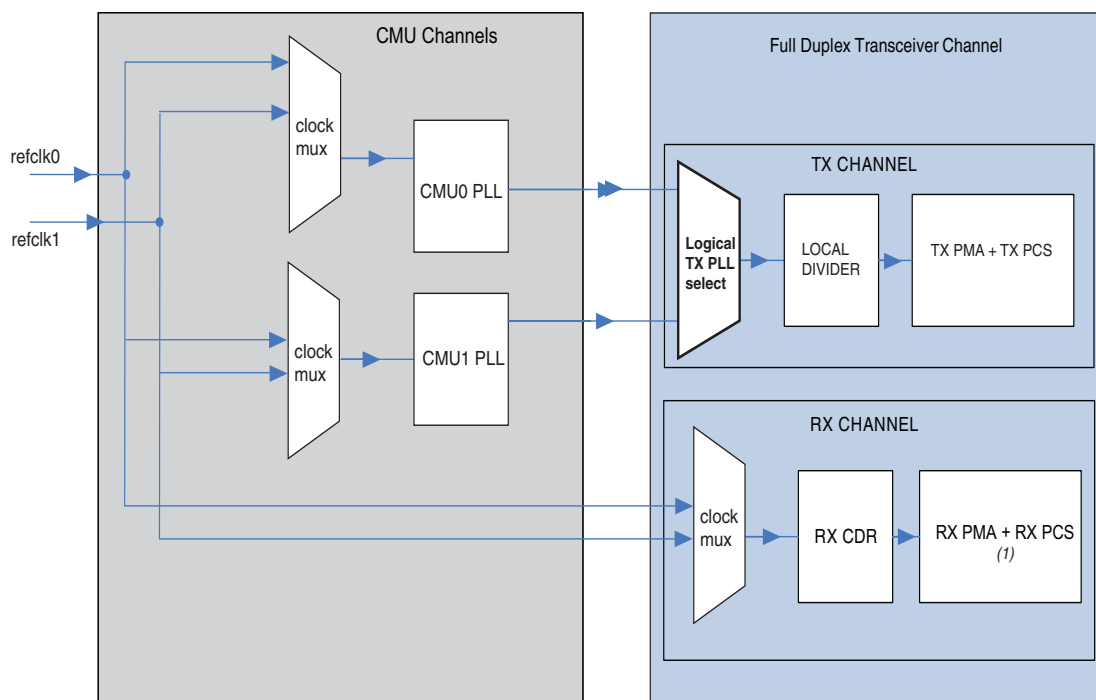
During CMU PLL reconfiguration mode, the dynamic reconfiguration controller automatically powers down the selected CMU PLL until it completes reconfiguration. The ALTGX_RECONFIG instance does not provide external ports to control the CMU PLL power down. When you reconfigure the CMU PLL, the `p11_locked` signal goes low. Therefore, after reconfiguring the transceiver, wait for the `p11_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

If 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.

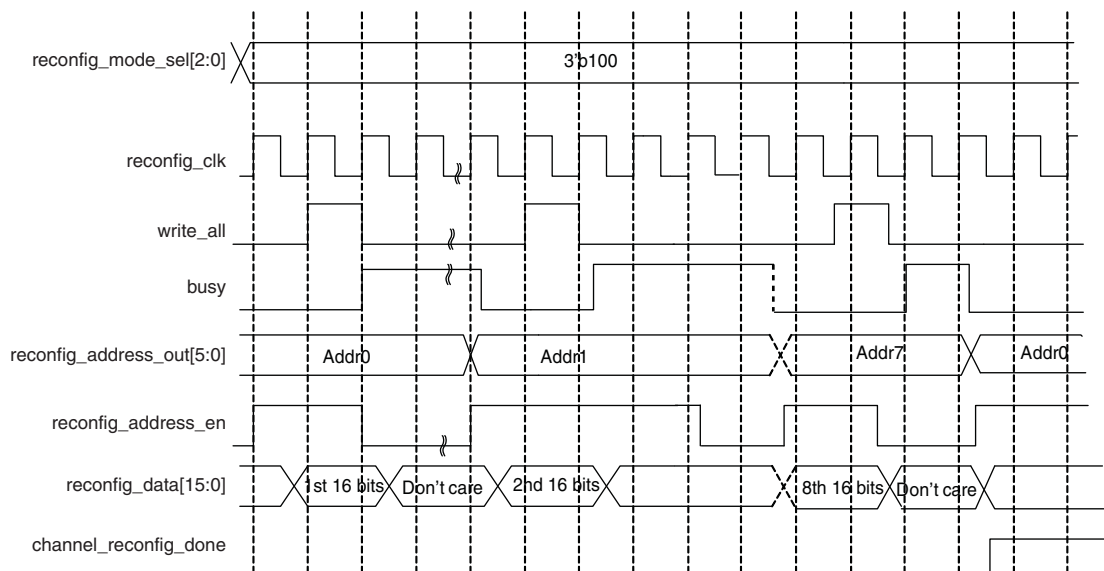
CMU PLL Reconfiguration Operation

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 in Stratix IV Devices* 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 and ×8, XAUI, and PCIe ×4 and ×8. For the allowed configurations, refer to [Table 5-5 on page 5-19](#).

For instance, to dynamically reconfigure an ALTGX instance in Basic ×4 configuration to a 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 PCIe mode.



To switch between one bonded PCS configuration and another, always use:

- 1) Channel and CMU PLL reconfiguration mode followed by
- 2) Central control unit reconfiguration mode

Use the same **.mif** for both of these steps. In step 1, a partial **.mif** is written and the remaining contents of the **.mif** is written in step 2. In step 1, reconfigure all the channels one-by-one. In step-2, reconfiguration of the central control unit is transceiver-block based. Reconfigure any one of the four channels in the transceiver block.

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.

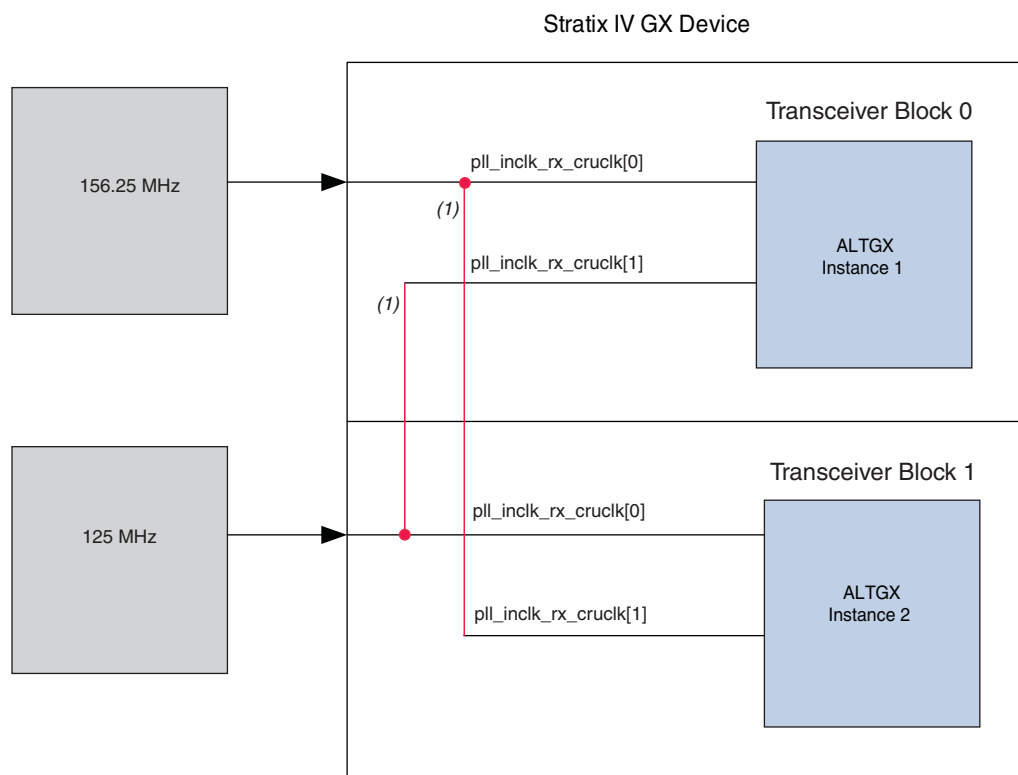
For more information about input reference clocking, refer to the “Input Reference Clocking” section of the *Transceiver Clocking in Stratix IV Devices* chapter.

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

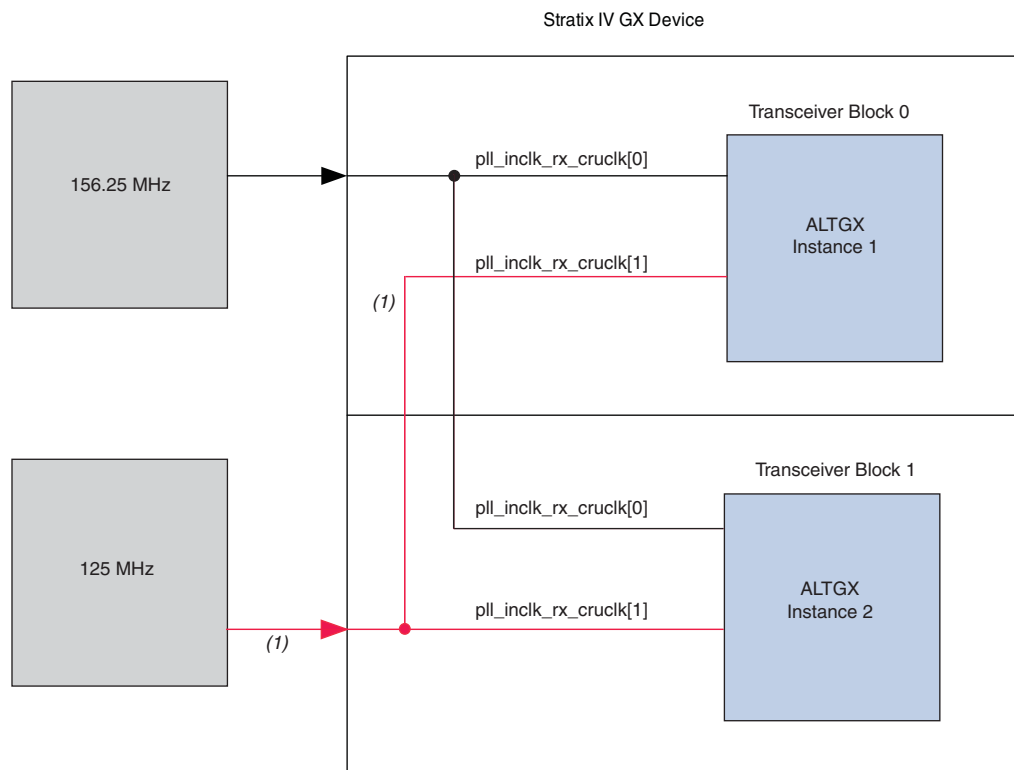


Note to Figure 5-28:

(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 the 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

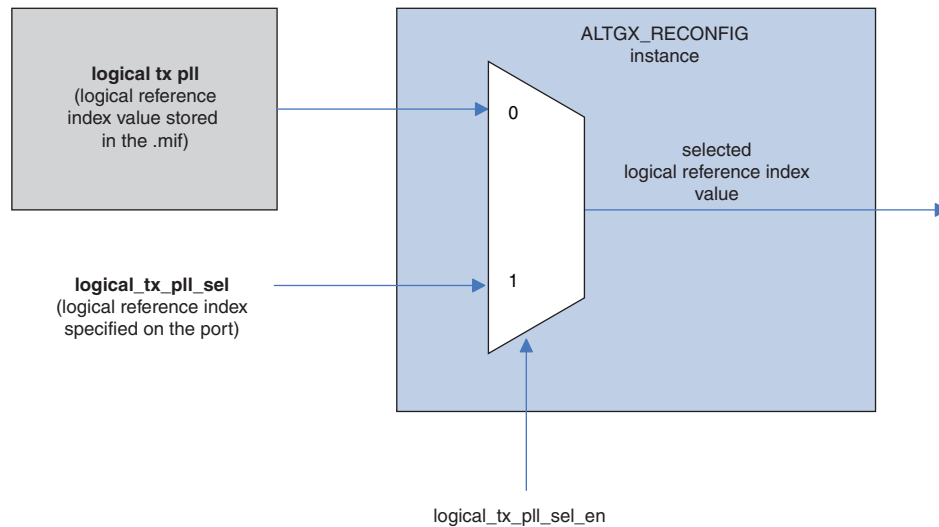


Table 5–11 lists how the dynamic reconfiguration controller selects between the logical reference index stored in the `.mif` (logical tx pll) and the logical reference index specified at the `logical_tx_pll_sel` port.

Table 5–11. Various Combinations of the `logical_tx_pll_sel` and `logical_tx_pll_sel_en` Ports

<code>logical_tx_pll_sel</code>	<code>logical_tx_pll_sel_en</code>	Logical Reference Index Value Selected by the <code>ALTGX_RECONFIG</code> Instance
enabled	enabled and value is 1	Value on the <code>logical_tx_pll_sel</code> port
enabled	enabled and value is 0	logical reference index value stored in the <code>.mif</code> (logical tx pll)
enabled	disabled	Value on the <code>logical_tx_pll_sel</code> port
disabled	disabled	logical reference index value stored in the <code>.mif</code> (logical tx pll)

Altera recommends keeping track of the transmitter PLL that drives the channel when you configure a transceiver channel in the ALTGX MegaWizard Plug-In Manager.



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
<p>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.</p> <ul style="list-style-type: none"> 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. 	<p>Assume that the transceiver channel listens to CMU1 PLL and the logical reference index assigned to it is 0.</p> <ul style="list-style-type: none"> 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. 	<p>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.</p> <ul style="list-style-type: none"> You can re-use this .mif created to reconfigure the transmitter PLL of another transceiver block under the following condition: <ul style="list-style-type: none"> 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. 	<p>Consider that you create a .mif containing the logical reference index of the transmitter PLL that the reconfigured transceiver channel needs to listen to.</p> <ul style="list-style-type: none"> 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.

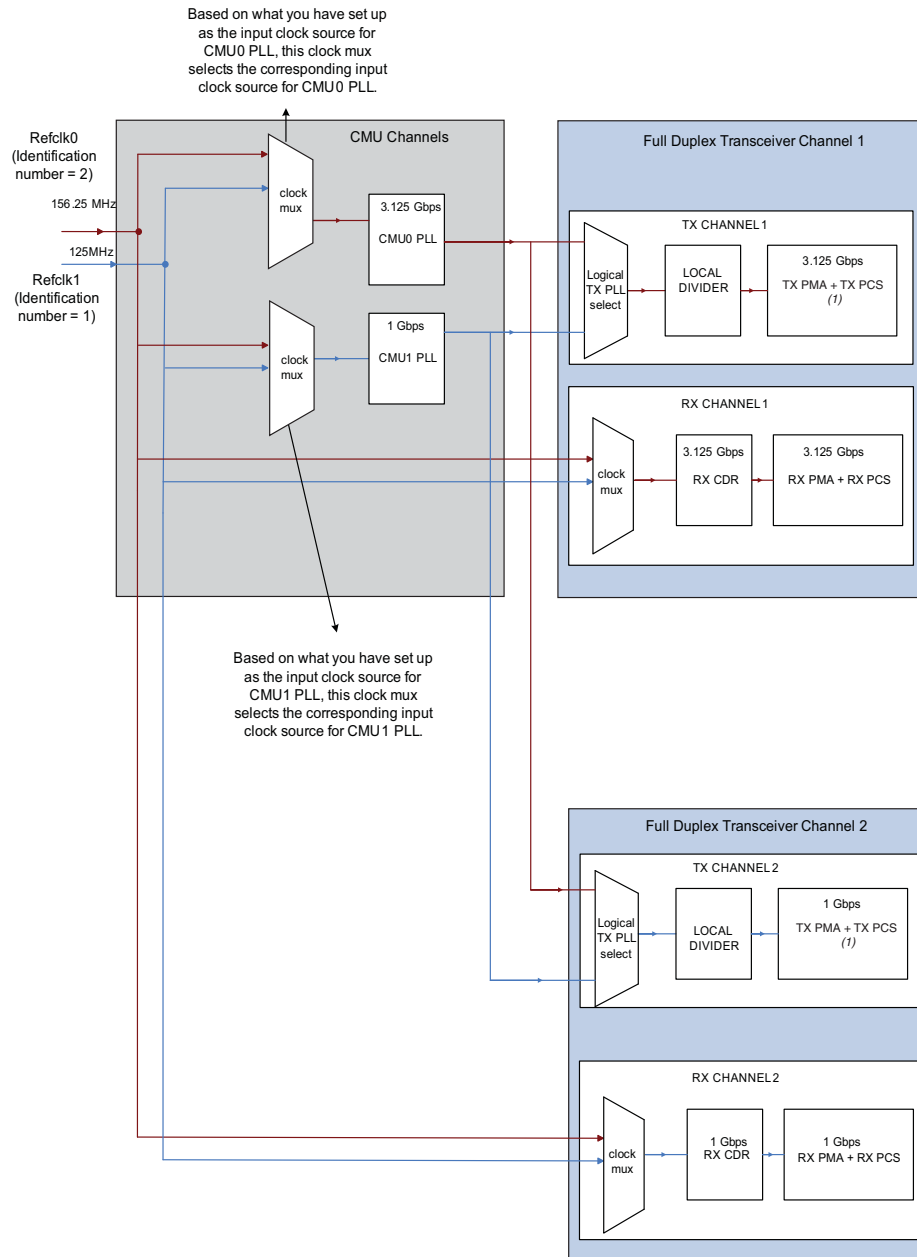
Guidelines for Specifying the Input Reference Clocks

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.
- 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, the PCS unit may or may not be present.

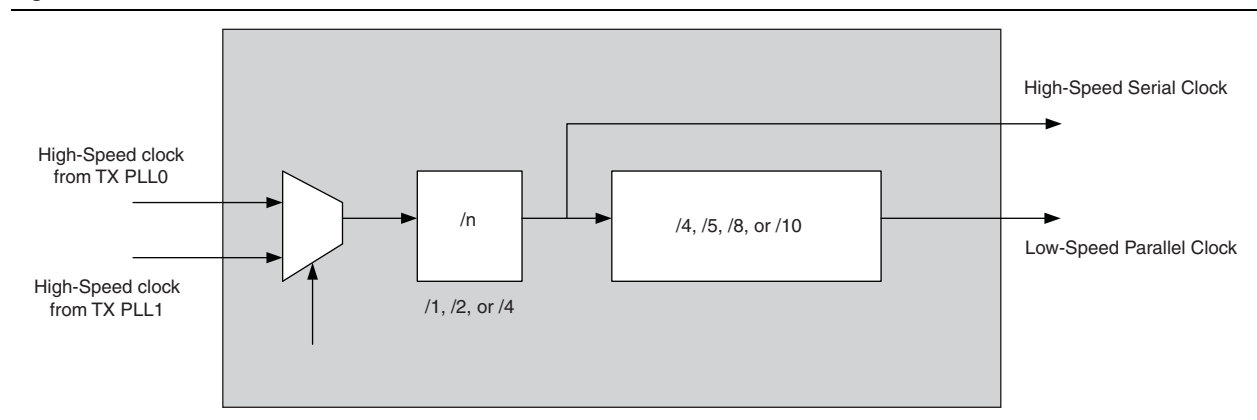
Data Rate Division in Transmitter Mode Details

You can use data 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 must ensure that a specific functional mode supports the data rate range before dividing the clock when using this rate switch option.



Data rate division in transmitter mode is applicable only to channels configured in non-bonded mode clocked by the CMU0/CMU1 located within the same transceiver block.

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.

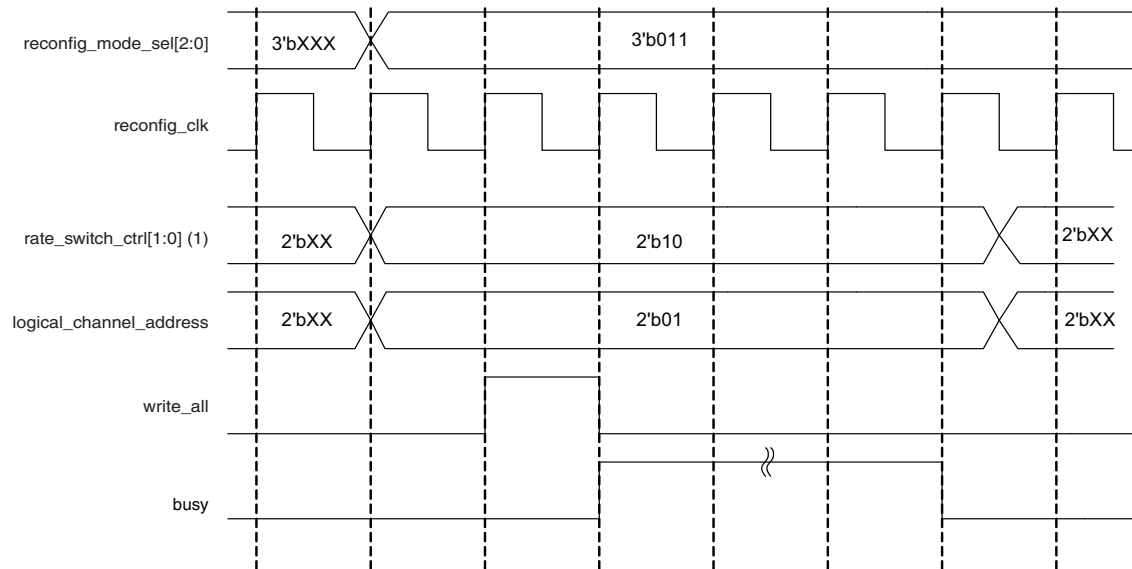
-  Dynamic rate switch has no effect on the dividers on the receive side of the transceiver channel. It can be used only for the transmitter.
-  Data rate division in transmitter mode does not require a `.mif`.

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

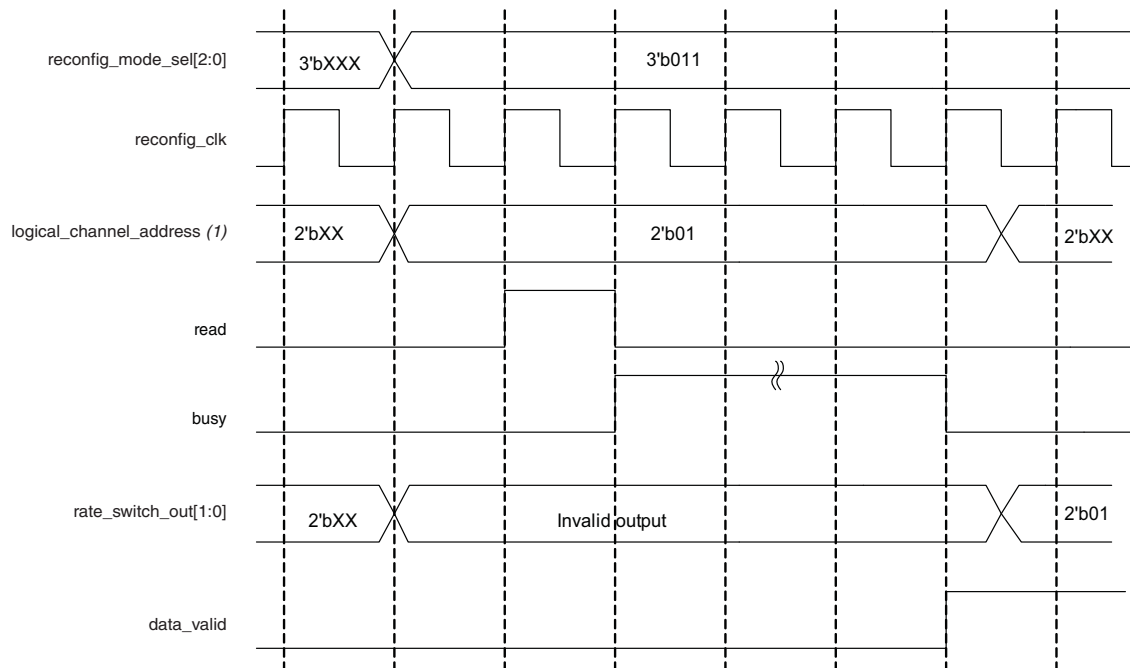


Note to Figure 5-33:

- (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 data 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 in Stratix IV Devices* 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. For the **Transmitter only** configuration, the ALTGX_RECONFIG must also be connected to the ALTGX by either selecting a dynamic reconfiguration mode or instantiating a dummy **Receiver only** ALTGX instance in each side of the device.

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. However, for **Transmitter only**, the ALTGX instance still needs to be connected to ALTGX_RECONFIG. For **Transmitter only**, the `reconfig_from_gxb`, `reconfig_to_gxb`, `busy`, and `reconfig_clk` signals can be enabled by selecting a dynamic reconfiguration mode. The other method to connect the ALTGX_RECONFIG for **Transmitter only** configuration is to instantiate a dummy **Receiver only** ALTGX instance in each side of the device.

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 logic element (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 5-91](#).

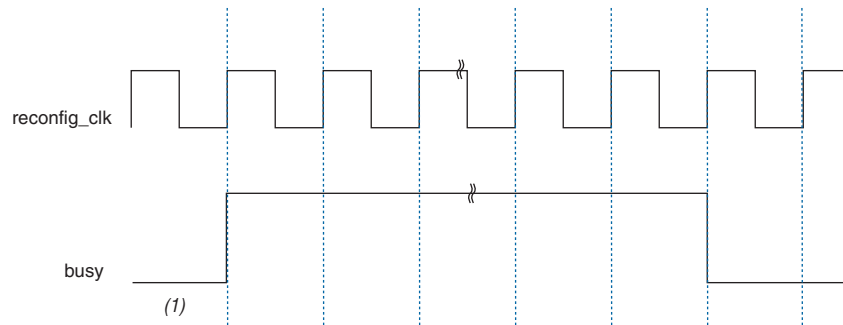
ALTGX_RECONFIG Instance Signals Transition during Offset Cancellation

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 the 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 in Stratix IV Devices* 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 32 different positions across one unit interval (UI) of the incoming data. You can manually control the sampling points and check the bit-error rate (BER) at each of these 32 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 for Stratix IV Devices* 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 Interface Specification](#).

Connections Between the ALTGX and ALTGX_RECONFIG Instances

To enable the EyeQ options, follow these 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 5-70.
2. Connect the `reconfig_{to/from}gxb` ports between the ALTGX and ALTGX_RECONFIG instances.

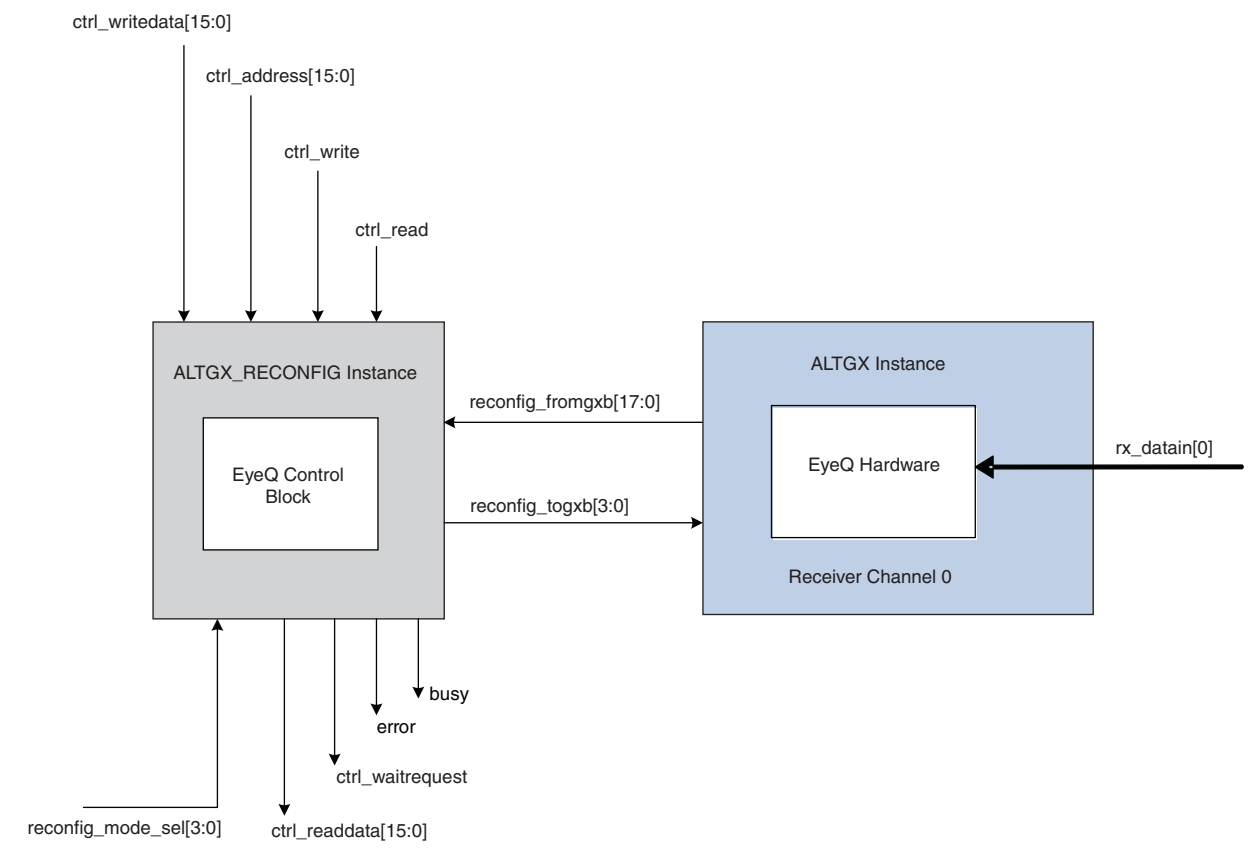
EyeQ control logic in the dynamic reconfiguration controller allows you to write to the registers in the EyeQ hardware. Therefore, you must have a state machine in the user design that communicates to the EyeQ control block of the ALTGX_RECONFIG instance. You can then access the internal registers of 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 lists the register memory of the 16-bit EyeQ registers.

Table 5-13. EyeQ Register Address Mapping

Address	Description
0x0	<ul style="list-style-type: none"> ■ Bit[0]—0/1: Disable/Enable EyeQ feature ■ Bit [15:1]—15'b0000000000000000
0x1	<ul style="list-style-type: none"> ■ Bits [5:0]—EyeQ phase step value. Refer to Table 5-14 for the EyeQ phase step encoding. ■ Bits [15:6]—10'b000000000000

Table 5-14 lists the EyeQ phase step encoding for the 32 phase steps spanning one unit interval (UI).

Table 5-14. EyeQ Phase Step Encoding

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
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 lists the register memory of the 16-bit EyeQ interface registers.

Table 5-15. EyeQ Interface Register Mapping

Address	Description
0x0	<p>Control/Status register (EyeQ CSR)</p> <ul style="list-style-type: none"> ■ 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 0 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 1 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.
0x1	<p>Channel address [15:0]—Specifies the transceiver channel for the desired EyeQ operation. This must match the <code>logical_channel_address</code> input port.</p>
0x2	<p>EyeQ register address [15:0]—Specifies the address EyeQ register to be read from or written to. The values supported are 0x0 or 0x1.</p>
0x3	<p>Data [15:0]—</p> <ul style="list-style-type: none"> ■ 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 0x0 (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 0x1 (the channel address register) to select the desired channel.
3. Issue a write to the EyeQ interface register 0x2 (the eye monitor register address) to select the desired EyeQ register.
4. Issue a write to the EyeQ interface register 0x3 (the data register) to provide the data to be written to the target EyeQ register.
5. Issue a write to EyeQ interface register 0x0 (the control and status register) to specify read/~write and to issue the start command.

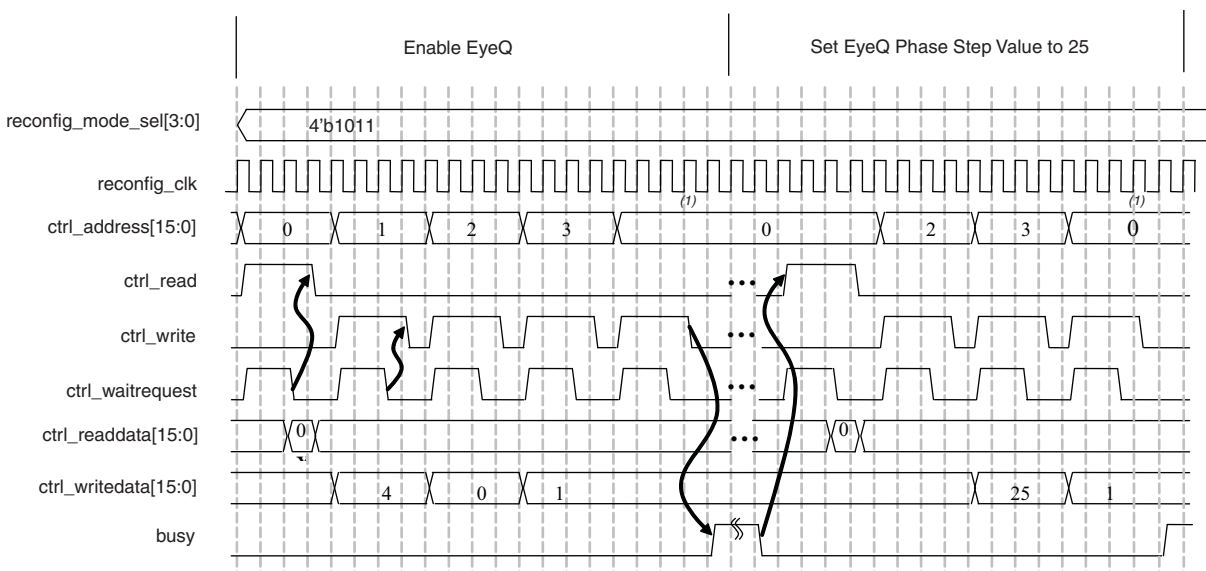
6. Poll the EyeQ interface register 0x0 (the control and status register) and wait for the busy status to be de-asserted. After the status is no longer busy, the data is considered successfully written for write transactions. For read transactions, this indicates that the contents of the data register 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:

- busy is 0 in the EyeQ CSR
- ctrl_waitrequest is low

Figure 5-37. Enabling EyeQ Mode



Note to Figure 5-37:

- (1) Writing a '1' here instructs the ALTGX_RECONFIG instance to program the EyeQ hardware.

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 PCIe 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
- The 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 for Stratix IV Devices* 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.

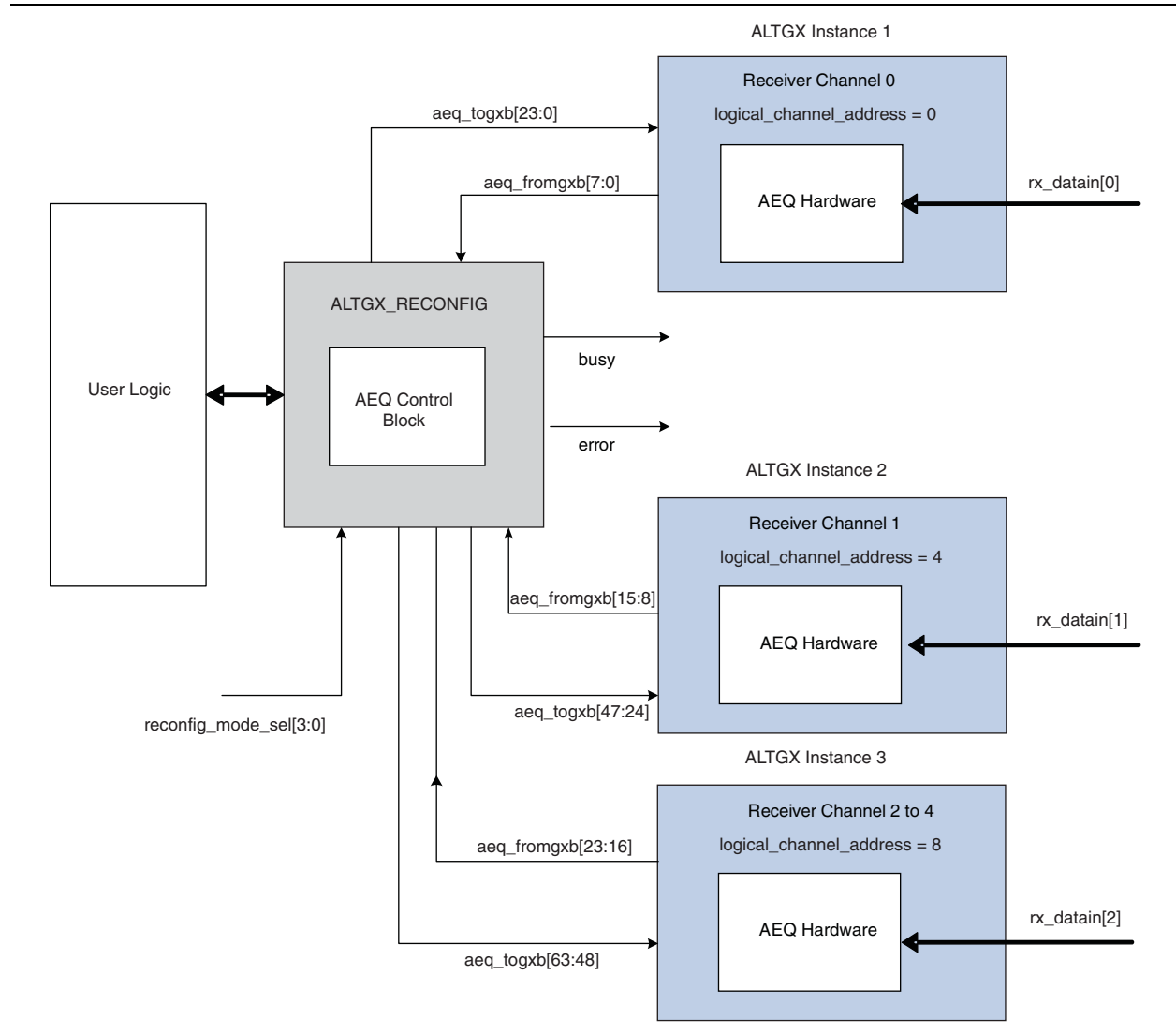
Connections Between the ALTGX and ALTGX_RECONFIG Instances

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.

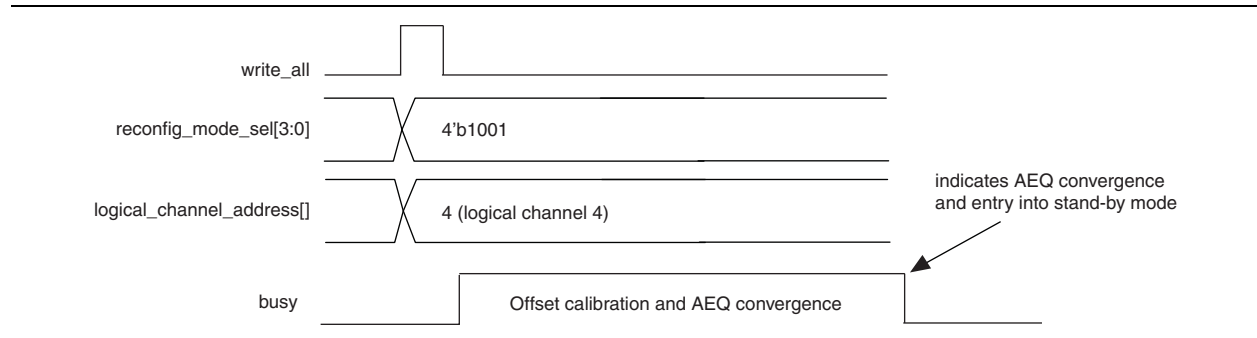
Figure 5-38. Connecting the ALTGX and ALTGX_RECONFIG Instances with AEQ Enabled



One Time Mode for a Single Channel

Stratix IV GX and GT devices only support one-time adaptation mode for the AEQ feature. Figure 5-39 shows the AEQ timing diagram in this mode.

Figure 5-39. AEQ Timing Diagram in One-Time Adaptation Mode



After assertion of the `write_all` signals, the dynamic reconfiguration controller performs the following steps sequentially:

1. Powers down the receiver buffer and performs offset calibration for the target channel.
2. Powers up the receiver buffer and runs the convergence algorithm to set the appropriate equalization settings.
3. Puts the AEQ circuitry in stand-by mode maintaining the converged equalization setting. In standby mode, no further adaptation occurs.

If you observe bit errors over time with the converged equalization settings, you can re-initiate one-time adaptation by following the timing diagram shown in Figure 5-39. Each time you re-initiate one-time adaptation, the receiver buffer is powered down for offset calibration, thereby interrupting the link during this time.

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) ⁽³⁾, ⁽⁴⁾

Port Name	Input/Output	Description
Clock Inputs to the ALTGX_RECONFIG Instance		
<code>reconfig_clk</code>	Input	<p>The frequency range of this clock depends on the following transceiver channel configuration modes:</p> <ul style="list-style-type: none"> ■ 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) <p>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 <code>REFCLK</code> pins or any clocks generated by transceivers.</p>

Table 5-16. Dynamic Reconfiguration Controller Port List (ALTGX_RECONFIG Instance) (Part 2 of 13) (3), (4)

Port Name	Input/ Output	Description
ALTGX and ALTGX_RECONFIG Interface Signals		
reconfig_fromgxb	Input	<p>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.</p> <p>In the ALTGX MegaWizard Plug-In Manager, the width of this signal depends on the following:</p> <ul style="list-style-type: none"> ■ 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. <p>For example, if the channels in the ALTGX instance are regular transceiver channels and if you select the number of channels as follows:</p> <p>$1 \leq \text{Channels} \leq 4$, then the output port <code>reconfig_fromgxb</code> = 17 bits</p> <p>$5 \leq \text{Channels} \leq 8$, then the output port <code>reconfig_fromgxb</code> = 34 bits</p> <p>$9 \leq \text{Channels} \leq 12$, then the output port <code>reconfig_fromgxb</code> = 51 bits</p> <p>However, if the channels in the ALTGX instance are PMA-only channels and if you select the number of channels as follows:</p> <p>Number of PMA-only channels = n, then the output port <code>reconfig_fromgxb</code> = $n * 17$ bits</p> <p>For example, <code>reconfig_fromgxb</code> = $6 * 17$ bits for 6 PMA-only channels.</p> <p>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.</p> <p>For example, if you select the total number of channels controlled by ALTGX_RECONFIG instance as follows:</p> <p>$1 \leq \text{Channels} \leq 4$, then the input port <code>reconfig_fromgxb</code> = 17 bits</p> <p>$5 \leq \text{Channels} \leq 8$, then the input port <code>reconfig_fromgxb</code> = 34 bits</p> <p>$9 \leq \text{Channels} \leq 12$, then the input port <code>reconfig_fromgxb</code> = 51 bits</p>

Table 5-16. Dynamic Reconfiguration Controller Port List (ALTGX_RECONFIG Instance) (Part 3 of 13) (3), (4)

Port Name	Input/ Output	Description
reconfig_fromgxb (continued)	Input	<p>To connect the <code>reconfig_fromgxb</code> port between the ALTGX_RECONFIG instance and multiple ALTGX instances, follow these rules:</p> <ul style="list-style-type: none"> ■ Connect the <code>reconfig_fromgxb[16:0]</code> of ALTGX Instance 1 to the <code>reconfig_fromgxb[16:0]</code> of the ALTGX_RECONFIG instance. Connect the <code>reconfig_fromgxb[]</code> port of the next ALTGX instance to the next available bits of the ALTGX_RECONFIG instance, and so on. ■ Connect the <code>reconfig_fromgxb</code> port of the ALTGX instance, which has the highest What is the starting channel number? option, to the MSB of the <code>reconfig_fromgxb</code> port of the ALTGX_RECONFIG instance. <p>The Quartus II Fitter produces an error if the dynamic reconfiguration option is enabled in the ALTGX instance but the <code>reconfig_fromgxb</code> and <code>reconfig_togxb</code> ports are not connected to the ALTGX_RECONFIG instance.</p> <p>For more information, refer to “Connecting the ALTGX and ALTGX_RECONFIG Instances” on page 5-11.</p>
reconfig_togxb[3:0]	Output	<p>An input port of the ALTGX instance and an output port of the ALTGX_RECONFIG instance. You must connect the <code>reconfig_togxb[3:0]</code> input port of every ALTGX instance controlled by the dynamic reconfiguration controller to the <code>reconfig_togxb[3:0]</code> output port of the ALTGX_RECONFIG instance.</p> <p>The width of this port is always fixed to 3 bits.</p> <p>For more information, refer to “Connecting the ALTGX and ALTGX_RECONFIG Instances” on page 5-11.</p>
FPGA Fabric and ALTGX_RECONFIG Interface Signals		
write_all	Input	<p>Assert this signal for one <code>reconfig_clk</code> clock cycle to initiate a write transaction from the ALTGX_RECONFIG instance to the ALTGX instance.</p> <p>You can use this signal in two ways for <code>.mif</code>-based modes:</p> <ul style="list-style-type: none"> ■ Continuous write operation—Select the Enable continuous write of all the words needed for reconfiguration option to pulse the <code>write_all</code> signal only once for writing a whole <code>.mif</code>. 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 <code>reconfig_clk</code> cycles. ■ Regular write operation—When the Enable continuous write of all the words needed for reconfiguration option is disabled, every word of the <code>.mif</code> requires its own write cycle.

Table 5-16. Dynamic Reconfiguration Controller Port List (ALTGX_RECONFIG Instance) (Part 4 of 13) (3), (4)

Port Name	Input/ Output	Description
busy	Output	<p>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 <code>reconfig_clk</code> 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.</p> <p>De-assertion of the <code>busy</code> signal indicates the successful completion of the offset cancellation process.</p> <p>For more information, refer to “Operation” on page 5-67.</p> <ul style="list-style-type: none"> ■ 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 <code>.mif</code> into the transceiver channel.
read	Input	<p>Assert this signal for one <code>reconfig_clk</code> clock cycle to initiate a read transaction. The <code>read</code> port is applicable only to the PMA controls reconfiguration mode and data rate division in transmitter mode. The <code>read</code> 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.</p> <p>For more information, refer to “Dynamically Reconfiguring PMA Controls” on page 5-13.</p>
data_valid	Output	<p>Applicable only to PMA controls reconfiguration mode. This port indicates the validity of the data read from the transceiver by the dynamic reconfiguration controller.</p> <p>The current data on the output read ports is the valid data ONLY if <code>data_valid</code> is high.</p> <p>This signal is enabled when you enable at least one PMA control port used in read transactions, for example <code>tx_vodctrl_out</code>.</p>
error	Output	<p>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 <code>busy</code> signal and asserts the <code>error</code> signal for two <code>reconfig_clk</code> cycles when you attempt an unsupported operation.</p> <p>For more information, refer to the “Error Indication During Dynamic Reconfiguration” on page 5-90.</p>

Table 5-16. Dynamic Reconfiguration Controller Port List (ALTGX_RECONFIG Instance) (Part 5 of 13) ⁽³⁾, ⁽⁴⁾

Port Name	Input/ Output	Description
logical_channel_address [8:0]	Input	<p>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.</p> <p>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.</p> <p>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.</p>
rx_tx_duplex_sel [1:0]	Input	<p>A 2 bit wide signal. You can select this in the Error checks/Data rate switch screen.</p> <p>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.</p> <p>For a setting of:</p> <ul style="list-style-type: none"> ■ 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) ⁽³⁾, ⁽⁴⁾

Port Name	Input/Output	Description																		
Analog Settings Control/Status Signals																				
tx_vodctrl[2:0] ⁽¹⁾	Input	<p>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.</p> <p>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.</p> <p>For more information, refer to “Dynamically Reconfiguring PMA Controls” on page 5-13.</p> <p>The following shows the V_{OD} values corresponding to the tx_vodctrl settings for 100-Ω termination.</p> <p>For more information, refer to the “Programmable Output Differential Voltage” section of the <i>Transceiver Architecture in Stratix IV Devices</i> chapter.</p> <table border="1"> <thead> <tr> <th>tx_vodctrl[2:0]</th> <th>V_{OD} (mV) for 1.4 V V_{CCH}</th> </tr> </thead> <tbody> <tr><td>3'b000</td><td>200</td></tr> <tr><td>3'b001</td><td>400</td></tr> <tr><td>3'b010</td><td>600</td></tr> <tr><td>3'b011</td><td>700</td></tr> <tr><td>3'b100</td><td>800</td></tr> <tr><td>3'b101</td><td>900</td></tr> <tr><td>3'b110</td><td>1000</td></tr> <tr><td>3'b111</td><td>1200</td></tr> </tbody> </table>	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
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																			
tx_vodctrla[2:0]	Input	<p>An optional transmit buffer V_{OD} control signal for Gen2. The signal is 3 bits per transmitter channel.</p> <p>The following shows the V_{OD} values corresponding to the tx_vodctrla port for 100 Ω termination:</p> <table border="1"> <thead> <tr> <th>tx_vodctrla</th> <th>V_{OD} (mV) Value</th> </tr> </thead> <tbody> <tr><td>0</td><td>200</td></tr> <tr><td>1</td><td>400</td></tr> <tr><td>2</td><td>600</td></tr> <tr><td>3</td><td>800</td></tr> <tr><td>4</td><td>1000</td></tr> <tr><td>5</td><td>1200</td></tr> <tr><td>6</td><td>700</td></tr> <tr><td>7</td><td>900</td></tr> </tbody> </table>	tx_vodctrla	V_{OD} (mV) Value	0	200	1	400	2	600	3	800	4	1000	5	1200	6	700	7	900
tx_vodctrla	V_{OD} (mV) Value																			
0	200																			
1	400																			
2	600																			
3	800																			
4	1000																			
5	1200																			
6	700																			
7	900																			
tx_vodctrla_out[2:0]	Output	<p>An optional transmit V_{OD} read control signal. The tx_vodctrla_out[2:0] signal reads out the Gen2 V_{OD} value written in the control register.</p>																		

Table 5-16. Dynamic Reconfiguration Controller Port List (ALTGX_RECONFIG Instance) (Part 7 of 13) (3), (4)

Port Name	Input/Output	Description
tx_preemp_0t[4:0] ⁽¹⁾	Input	<p>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.</p> <p>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.</p> <p>For more information, refer to “Dynamically Reconfiguring PMA Controls” on page 5-13.</p> <p>The following values are the legal settings allowed for this signal:</p> <ul style="list-style-type: none"> 0 represents 0 1-15 represents -15 to -1 16 represents 0 17 - 31 represents 1 to 15 <p>In the PCIe configuration, set tx_preemp_0t[4:0] to 5'b00000 when you do a rate switch from Gen 1 mode 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 PCIe Gen 2 mode.</p> <p>For more information, refer to the “Programmable Pre-Emphasis” section of the Transceiver Architecture in Stratix IV Devices chapter.</p>
tx_preemp_1t[4:0] ⁽¹⁾	Input	<p>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.</p> <p>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.</p> <p>For more information, refer to “Dynamically Reconfiguring PMA Controls” on page 5-13 and the “Programmable Pre-Emphasis” section of the Transceiver Architecture in Stratix IV Devices chapter.</p>
tx_preemp_1ta[4:0]	Input	<p>An optional pre-emphasis control for the first post-tap for the transmit buffer in Gen2 mode. 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.</p>
tx_preemp_1ta_out[4:0]	Output	<p>An optional first post-tap, pre-emphasis read control signal for Gen2. The tx_preemp_1ta_out[4:0] signal reads out the value written by its input control signal.</p>
tx_preemp_1tb[4:0]	Input	<p>An optional de-emphasis control for the first post-tap for the transmit buffer in Gen2 mode. Depending on what value you set at this input, the controller dynamically writes the value to the de-emphasis control register of the transmit buffer.</p>

Table 5-16. Dynamic Reconfiguration Controller Port List (ALTGX_RECONFIG Instance) (Part 8 of 13) (3), (4)

Port Name	Input/Output	Description
tx_preemp_1tb_out[4:0]	Output	An optional first post-tap, de-emphasis read control signal for Gen2. The tx_preemp_1tb_out[4:0] signal reads out the value written by its input control signal.
tx_preemp_2t[4:0] (1)	Input	<p>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.</p> <p>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.</p> <p>For more information, refer to “Dynamically Reconfiguring PMA Controls” on page 5-13.</p> <p>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</p> <p>In the PCIe configuration, set tx_preemp_2t[4:0] to 5'b00000 when you do a rate switch from Gen 1 mode 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_pipedeeemph take affect in PCIe Gen 2 mode.</p> <p>For more information, refer to the “Programmable Pre-Emphasis” section of the Transceiver Architecture in Stratix IV Devices chapter.</p>
rx_eqctrl1[3:0] (1)	Input	<p>An optional write control to write an equalization control value for the receive side of the PMA.</p> <p>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.</p> <p>For more information, refer to “Dynamically Reconfiguring PMA Controls” on page 5-13 and the “Programmable Equalization and DC Gain” section of the Transceiver Architecture in Stratix IV Devices chapter.</p>

Table 5-16. Dynamic Reconfiguration Controller Port List (ALTGX_RECONFIG Instance) (Part 9 of 13) (3), (4)

Port Name	Input/Output	Description
rx_eqdcgain[2:0] (1), (2)	Input	<p>An optional equalizer DC gain write control.</p> <p>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.</p> <p>For more information, refer to “Dynamically Reconfiguring PMA Controls” on page 5-13.</p> <p>The following values are the legal settings allowed for this signal:</p> <p>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</p> <p>For more information, refer to the “Programmable Equalization and DC Gain” section of the <i>Transceiver Architecture in Stratix IV Devices</i> chapter.</p>
tx_vodctrl_out[2:0]	Output	<p>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.</p>
tx_preemp_0t_out[4:0]	Output	<p>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.</p>
tx_preemp_1t_out[4:0]	Output	<p>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.</p>
tx_preemp_2t_out[4:0]	Output	<p>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.</p>
rx_eqctrl_out[3:0]	Output	<p>An optional read control signal to read the equalization setting of the ALTGX instance. The width of this output signal depends on the number of channels controlled by the dynamic reconfiguration controller.</p>
rx_eqdcgain_out[2:0]	Output	<p>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.</p>

Table 5-16. Dynamic Reconfiguration Controller Port List (ALTGX_RECONFIG Instance) (Part 10 of 13) ⁽³⁾, ⁽⁴⁾

Port Name	Input/ Output	Description
Transceiver Channel Reconfiguration Control/Status Signals		
reconfig_mode_sel[3:0]	Input	<p>Set the following values at this signal to activate the appropriate dynamic reconfiguration mode:</p> <p>3'b000 = PMA controls reconfiguration mode. This is the default value.</p> <p>3'b011 = data rate division in transmitter mode</p> <p>3'b100 = CMU PLL reconfiguration mode</p> <p>3'b101 = channel and CMU PLL reconfiguration mode</p> <p>3'b110 = channel reconfiguration with transmitter PLL select mode</p> <p>3'b111 = central control unit reconfiguration mode</p> <p>The reconfig_mode_sel signal is 4 bits wide when you enable Adaptive Equalization control or EyeQ control:</p> <p>4'b1000 = AEQ control (continuous mode for a single channel)</p> <p>4'b1001 = AEQ control (one time mode for a single channel)</p> <p>4'b1010 = AEQ control (power down for a single channel)</p> <p>4'b1011 = EyeQ control</p> <p>reconfig_mode_sel[] is available as an input only when you enable more than one dynamic reconfiguration mode.</p>
reconfig_address_out[5:0]	Output	<p>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 the Channel and TX PLL select/reconfig option.</p> <p>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).</p>
reconfig_address_en	Output	<p>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.</p> <p>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.</p>
reset_reconfig_address	Input	<p>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.</p> <p>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.</p>

Table 5-16. Dynamic Reconfiguration Controller Port List (ALTGX_RECONFIG Instance) (Part 11 of 13) (3), (4)

Port Name	Input/Output	Description
reconfig_data[15:0]	Input	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	Available for selection only in .mif -based transceiver channel reconfiguration modes. For more information, refer to “Reduced .mif Reconfiguration” on page 5-24.
rate_switch_ctrl[1:0]	Input	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	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	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	The channel_reconfig_done 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) ⁽³⁾, ⁽⁴⁾

Port Name	Input/Output	Description
reconfig_reset	Input	An optional signal that you can use to reset the ALTGX_RECONFIG instance. <code>reconfig_reset</code> must be held high for at least one clock cycle to take effect. When feeding into the <code>reconfig_reset</code> port, the reset signal must be synchronized to the <code>reconfig_clk</code> domain.
aeq_fromgxb[7:0]	Input	The width of the <code>aeq_fromgxb[7:0]</code> 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: $1 \leq \text{Channels} \leq 4$, then the input port <code>reconfig_fromgxb</code> = 8 bits $5 \leq \text{Channels} \leq 8$, then the input port <code>reconfig_fromgxb</code> = 16 bits $9 \leq \text{Channels} \leq 12$, then the input port <code>reconfig_fromgxb</code> = 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 the <code>aeq_togxb</code> 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: $1 \leq \text{Channels} \leq 4$, then the input port <code>reconfig_fromgxb</code> = 24 bits $5 \leq \text{Channels} \leq 8$, then the input port <code>reconfig_fromgxb</code> = 48 bits $9 \leq \text{Channels} \leq 12$, then the input port <code>reconfig_fromgxb</code> = 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 <code>ctrl_address</code> port.
ctrl_readdata[15:0]	Output	Used for EyeQ control. Contents of the EyeQ interface register selected using the <code>ctrl_address</code> 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 <code>ctrl_writedata</code> 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 <code>ctrl_readdata</code> port.

Table 5-16. Dynamic Reconfiguration Controller Port List (ALTGX_RECONFIG Instance) (Part 13 of 13) (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 PCIe mode, this input must be tied to 001 to be PCIe-compliant.
- (3) For the various dynamic reconfiguration controller input and output ports and the software settings, refer to the *ALTGX_RECONFIG Megafunction User Guide for Stratix IV Devices* chapter.
- (4) For the various transceiver input and output ports and the software settings, refer to the *ALTGX Transceiver Setup Guide for Stratix IV Devices* 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 signal 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` signal 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` signal 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
- **Adaptive Equalization option—read operation:**
 - The `reconfig_mode_sel` input port is set to 7, 8, 9, or 10
 - The `read` signal is asserted
- **EyeQ option—read operation:**
 - The `reconfig_mode_sel` input port is set to 11
 - 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.

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 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)

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 18500 `reconfig_clk` clock cycles per channel for **Receiver only** and **Receiver and Transmitter** channels. It takes 900 `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 additional 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:

- 900 cycles for the **Transmitter only** channel
- 18,500 cycles for the **Receiver only** channel
- 2 cycles each for non-existent channels with `logical_channel_addresses = 1, 2, and 3` and 130,000 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 = 149,406 cycles (900 + 18,500 + 6 + 130,000).

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 <code>reconfig_clk</code> Clock Cycles
Transmitter channel reconfiguration	1,518
Receiver channel reconfiguration	5,255
Transmitter and receiver channel reconfiguration	6,762
CMU PLL only reconfiguration	863
Transmitter channel and CMU PLL reconfiguration	2,370
Transceiver channel and CMU PLL reconfiguration	7,614
Central control unit reconfiguration	925

Dynamic Reconfiguration (ALTGX_RECONFIG Instance) Resource Utilization

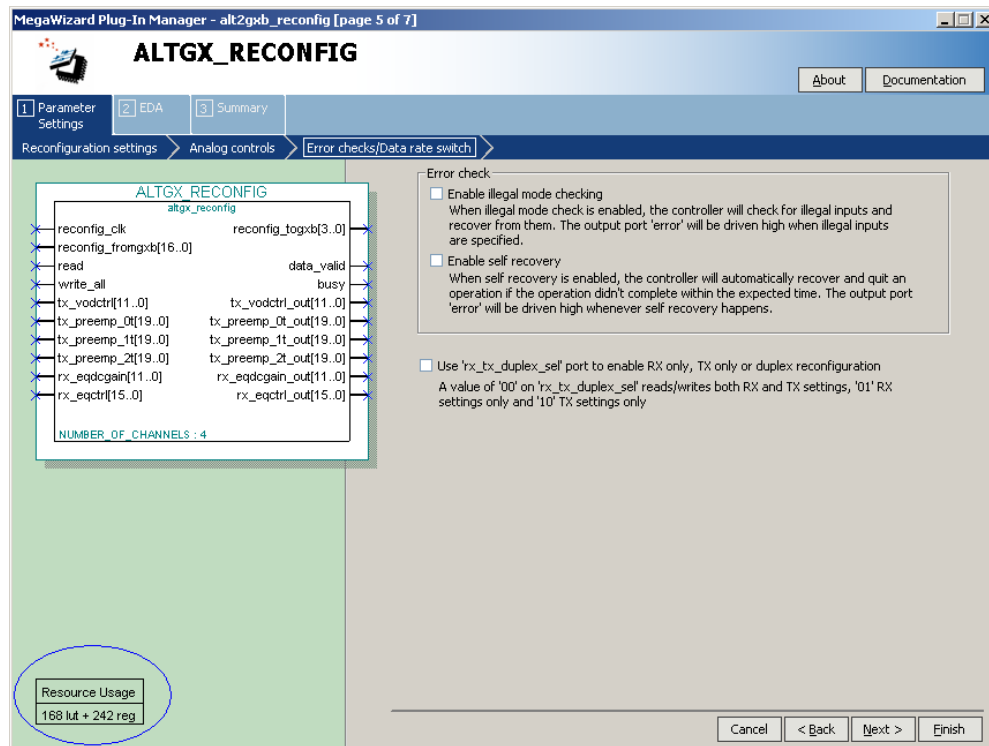
You can observe the resources used during dynamic reconfiguration in the ALTGX_RECONFIG MegaWizard Plug-In Manager. This section contains an estimate of the LE resources used during dynamic reconfiguration.

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_vodctr1` selected and the number of registers is 130.

Figure 5-40 shows resource utilization in the ALTGX_RECONFIG MegaWizard Plug-In Manager.

Figure 5-40. 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 for offset cancellation. Therefore, the duration of the offset cancellation 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.

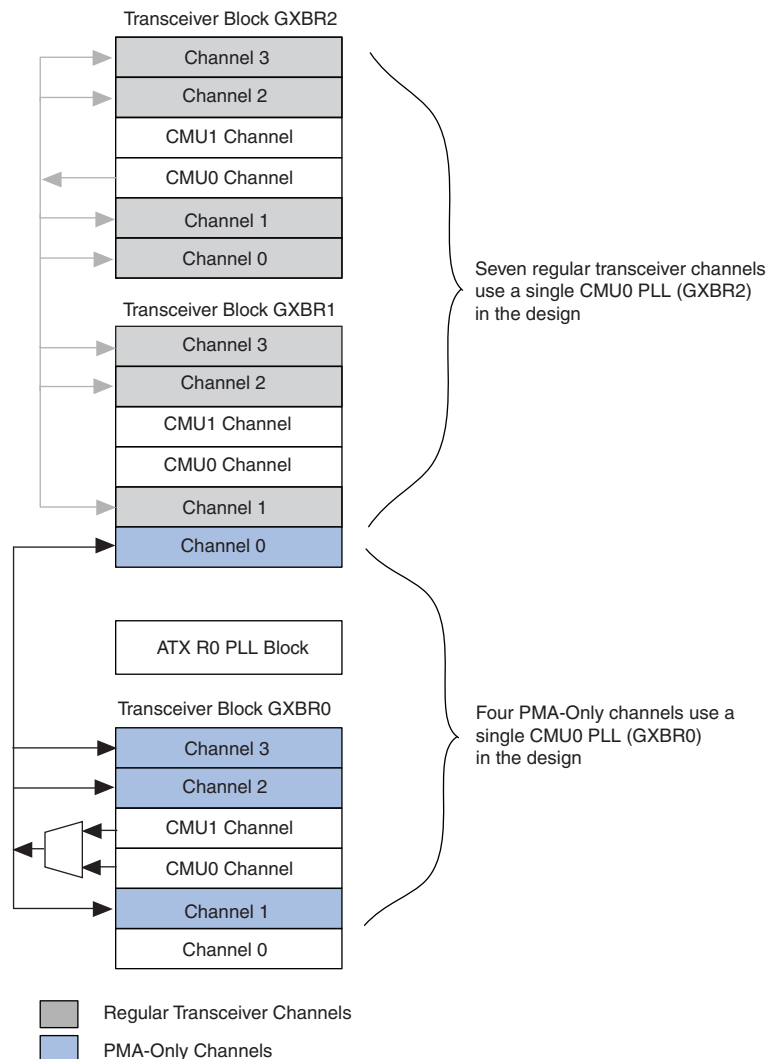
Example 1

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 5-41 shows the arrangement of these channels in the S4GX230 device.

Figure 5-41. Dynamic Reconfiguration Configuration for the S4GX230 Device (Example 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.

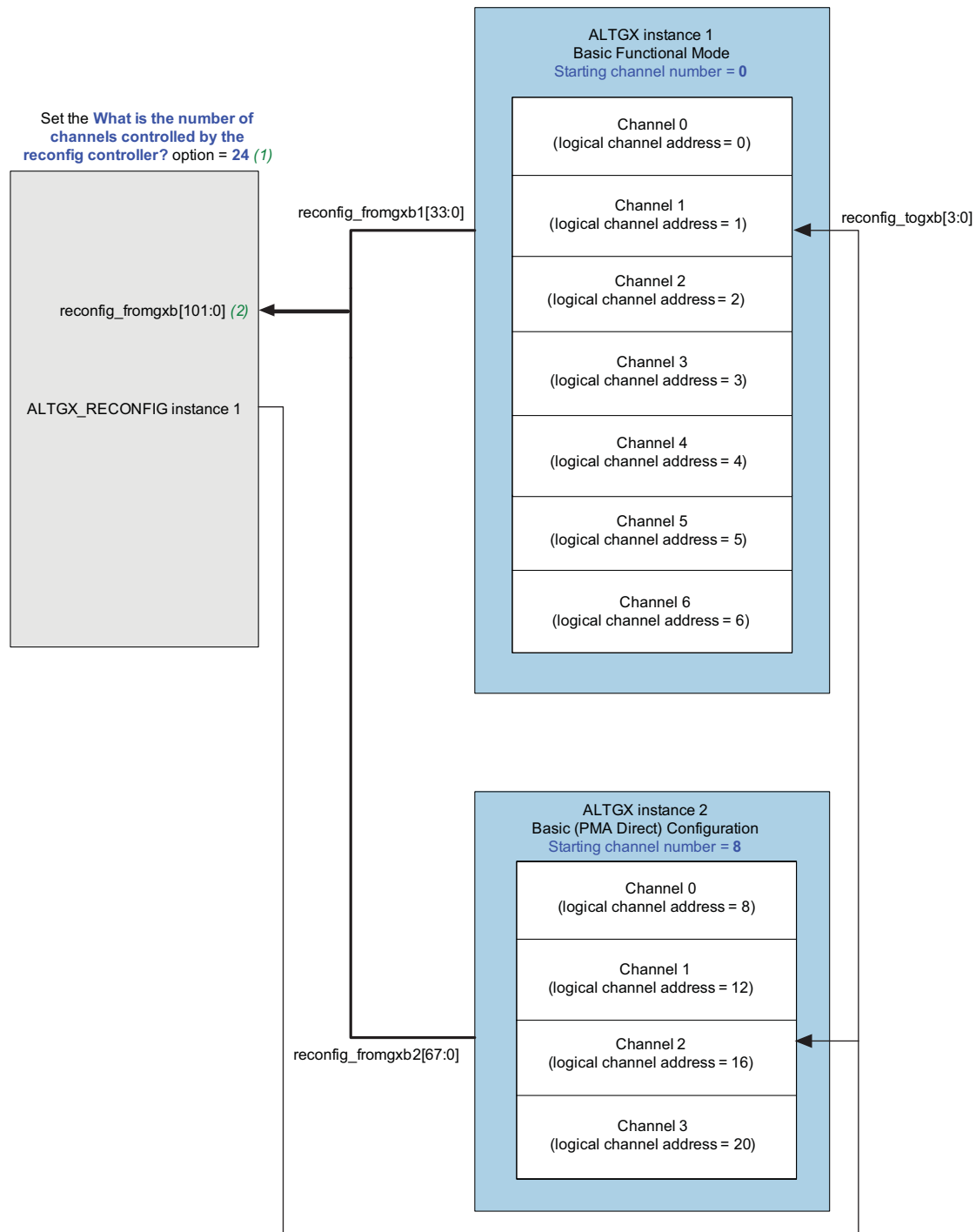
Table 5-18 lists how to set the starting channel number in the ALTGX MegaWizard Plug-In Manager, the total number of channels in the ALTGX_RECONFIG MegaWizard Plug-In Manager, and how to connect the ALTGX instances to the ALTGX_RECONFIG instance.

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.	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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. 	<ul style="list-style-type: none"> 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 5-42 shows how the logical channel addresses of all the channels are set based on what you set as the starting channel number.

Figure 5-42. Logical Channel Addresses for Example 1





Notes to Figure 5-42:

- (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] }`

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 the 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 must generate two **.mifs**. Use one to move from a data rate of 2.5 Gbps to 5 Gbps and the other to revert back to 2.5 Gbps.
- For the for PMA-only channels, you must generate two **.mifs**. Use one to move from a data rate of 3.125 Gbps to 5 Gbps and the other to revert back to 3.125 Gbps.

For more information, refer to [“Memory Initialization File \(.mif\)” on page 5-20](#).

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](#).

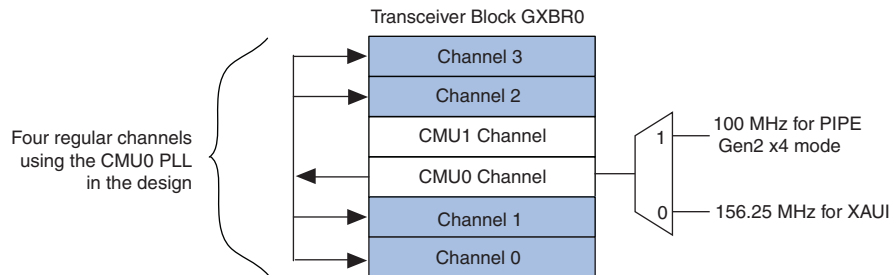
Example 2

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 PCIe Gen2 ×4 configuration (the secondary configuration) and vice versa.

Figure 5-43 shows the arrangement of these channels in the S4GX230 device.

Figure 5-43. Dynamic Reconfiguration Configuration for the S4GX230 Device (Example 2)



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)

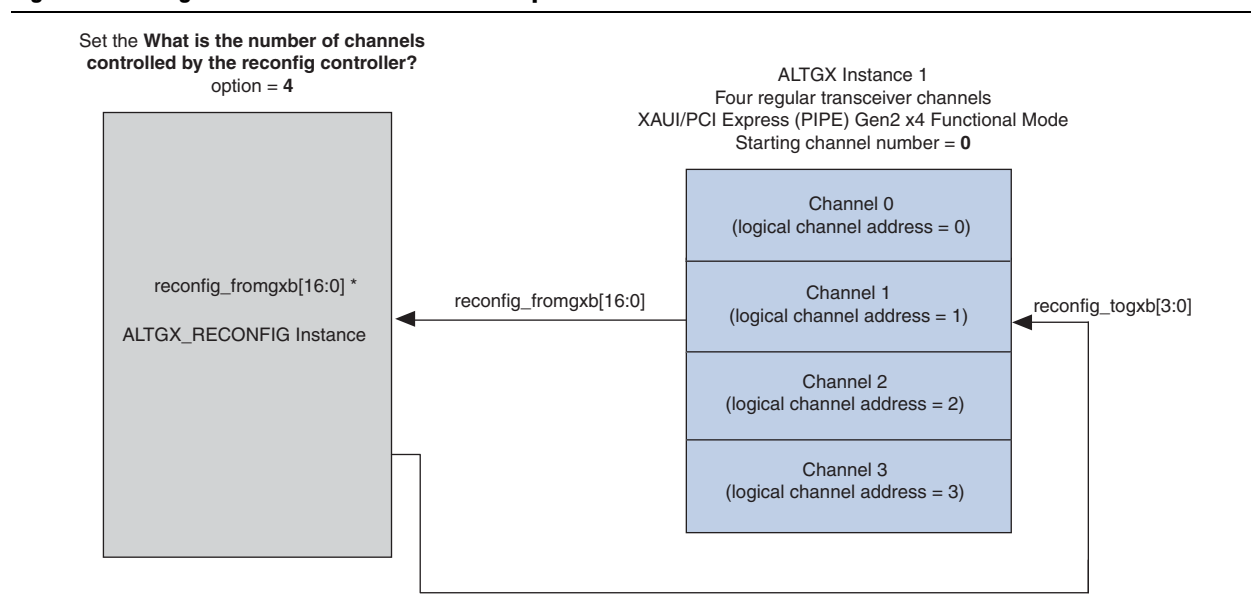
ALTGX Settings and Instances		ALTGX_RECONFIG Setting and Instance	
ALTGX Setting	ALTGX Instance 1 (XAUI Mode)	ALTGX_RECONFIG Setting	ALTGX_RECONFIG Instance 1
What is the number of channels? option in the General screen	4 (Regular transceiver channels)	What is the number of channels controlled by the reconfig controller? option in the Reconfiguration settings screen.	<ul style="list-style-type: none"> ■ 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	<ul style="list-style-type: none"> ■ 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	<ul style="list-style-type: none"> ■ 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. ■ Set 1 for the PCIe s×4 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 5-44 shows how the logical channel addresses of all the channels are set based on what you set as the starting channel number.

Figure 5-44. Logical Channel Addresses for Example 2



Different Dynamic Reconfiguration Modes Involved

1. Channel and CMU PLL reconfiguration mode—used for reconfiguring four regular transceiver channels and the CMU0 PLL (in GXBR0) from XAUI mode to PCIe ×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 the CMU1 PLL 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 PCIe ×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 required to move from XAUI mode to PCIe ×4 mode
- Another .mif is required to revert back to XAUI mode from PCIe ×4 mode

For more information, refer to “[Memory Initialization File \(.mif\)](#)” on page 5-20.

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 5-24.

Document Revision History

Table 5-20 lists the revision history for this chapter.

Table 5-20. Document Revision History (Part 1 of 2)

Date	Version	Changes
January 2014	3.6	<ul style="list-style-type: none"> ■ Updated the “Logical Channel Addressing of PMA-Only Channels” section.
November 2013	3.5	<ul style="list-style-type: none"> ■ Updated the “CMU PLL Reconfiguration Mode Details” section.
September 2012	3.4	<ul style="list-style-type: none"> ■ Updated Table 5-13 title. ■ Updated Figure 5-6. ■ Updated Figure 5-37.
December 2011	3.3	<ul style="list-style-type: none"> ■ Updated the “Logical Channel Addressing of PMA-Only Channels” and “Transceiver Channel Reconfiguration Mode Details” sections. ■ Updated Table 5-6, Table 5-9, and Table 5-16.

Table 5-20. Document Revision History (Part 2 of 2)

Date	Version	Changes
February 2011	3.2	<ul style="list-style-type: none"> ■ Updated Table 5-5, Table 5-6, and Table 5-16. ■ Updated Figure 5-1. ■ Updated the “Transceiver Channel Reconfiguration Mode Details”. “PMA Controls Reconfiguration Mode Details”, “Connecting the ALTGX and ALTGX_RECONFIG Instances”, “One Time Mode for a Single Channel”, “Applying a .mif in the User Design”, and “Functional Simulation of the Dynamic Reconfiguration Process” sections. ■ Removed the “Continuous Mode for a Single Channel” and “Powerdown for a Single Channel” sections. ■ Updated chapter title. ■ Applied new template. ■ Minor text edits.
March 2010	3.1	<ul style="list-style-type: none"> ■ Updated Table 5-5, Table 5-6, Table 5-15, Table 5-16, and Table 5-17. ■ Updated Figure 5-1, Figure 5-14, Figure 5-16, Figure 5-26, and Figure 5-37. ■ Updated the “Blocks Reconfigured in the Data Rate Division in Transmitter Mode”, “Logical Channel Addressing of PMA-Only Channels”, “Central Control Unit Reconfiguration Mode Details”, “EyeQ”, “Error Indication During Dynamic Reconfiguration”, and “Functional Simulation of the Dynamic Reconfiguration Process” sections. ■ Added a note to the “Central Control Unit Reconfiguration Mode Details” section. ■ Minor text edits.
November 2009	3.0	<ul style="list-style-type: none"> ■ Completely re-wrote and re-organized chapter. ■ Updated all graphics and tables.
June 2009	2.1	<ul style="list-style-type: none"> ■ 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	2.0	<p>Complete re-write and re-organization of the chapter.</p> <p>Added or revised:</p> <ul style="list-style-type: none"> ■ 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	1.0	Initial release.

This chapter provides additional information about the document and Altera.

How to Contact Altera

To locate the most up-to-date information about Altera products, refer to the following table.

Contact ⁽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
Nontechnical support (general) (software licensing)	Email	nacomp@altera.com
	Email	authorization@altera.com










Note to Table:

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

Typographic Conventions

The following table shows the typographic conventions this document uses.

Visual Cue	Meaning
Bold Type with Initial Capital Letters	Indicate 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, software utility names, and GUI labels. For example, \qdesigns directory, D: drive, and chiptrip.gdf file.
<i>Italic Type with Initial Capital Letters</i>	Indicate document titles. For example, <i>Stratix IV Design Guidelines</i> .
<i>italic type</i>	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	Indicate keyboard keys and menu names. For example, the Delete key and the Options menu.
“Subheading Title”	Quotation marks indicate references to sections in a document and titles of Quartus II Help topics. For example, “Typographic Conventions.”

Visual Cue	Meaning
Courier type	<p>Indicates signal, port, register, bit, block, and primitive names. For example, <code>data1</code>, <code>tdi</code>, and <code>input</code>. The suffix <code>n</code> denotes an active-low signal. For example, <code>resetn</code>.</p> <p>Indicates command line commands and anything that must be typed exactly as it appears. For example, <code>c:\qdesigns\tutorial\chiptrip.gdf</code>.</p> <p>Also indicates sections of an actual file, such as a Report File, references to parts of files (for example, the AHDL keyword <code>SUBDESIGN</code>), and logic function names (for example, <code>TRI</code>).</p>
	An angled arrow instructs you to press the Enter key.
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.
	The question mark directs you to a software help system with related information.
	The feet direct you to another document or website with related information.
	The multimedia icon directs you to a related multimedia presentation.
	A caution calls attention to a condition or possible situation that can damage or destroy the product or your work.
	A warning calls attention to a condition or possible situation that can cause you injury.
	The envelope links to the Email Subscription Management Center page of the Altera website, where you can sign up to receive update notifications for Altera documents.