



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.



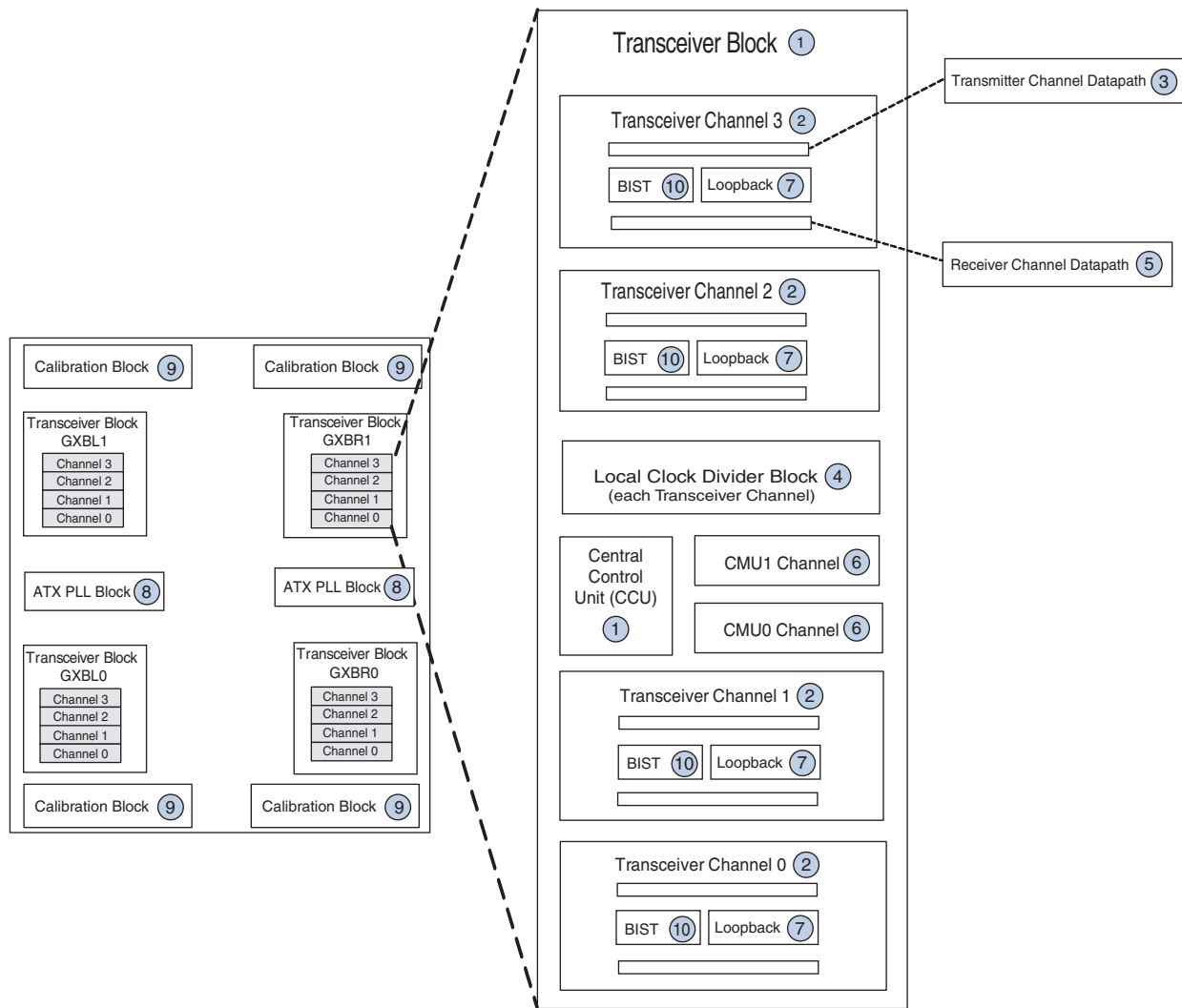
-  Stratix IV GX and GT devices have PCIe hard IP, PCS, and PMA blocks. For more information, refer to the [PCI Express Compiler User Guide](#).
-  For more information about Stratix IV GX and GT protocols, refer to the [Configuring Multiple Protocols and Data Rates in Stratix IV Devices](#) chapter.

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

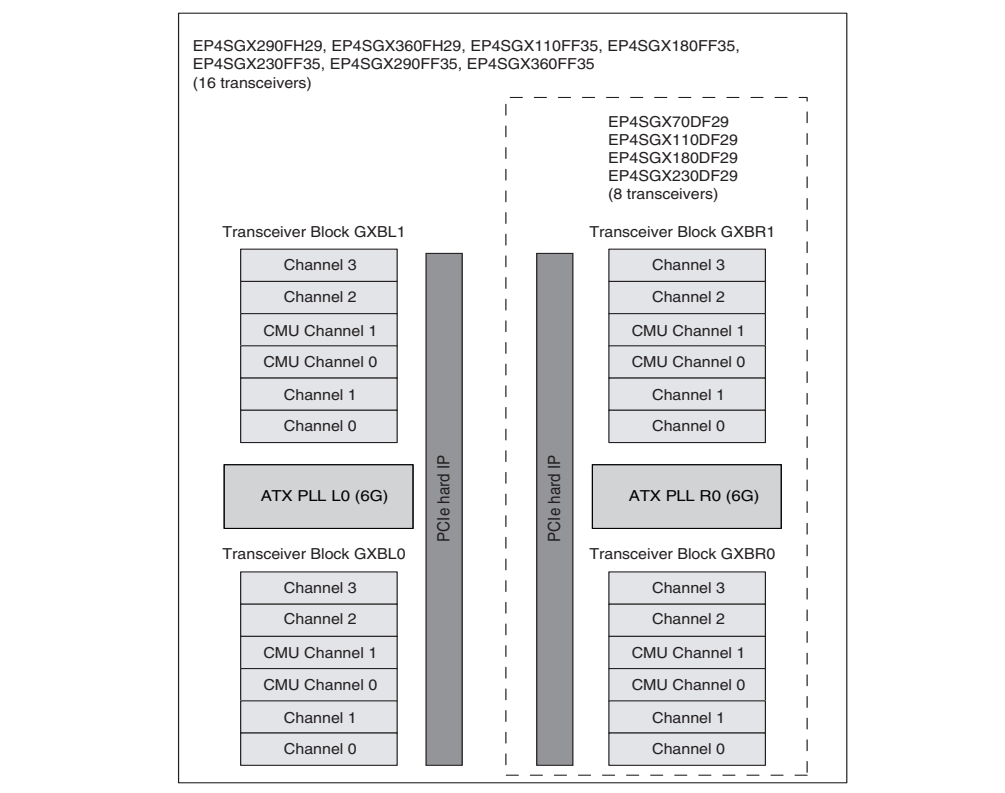
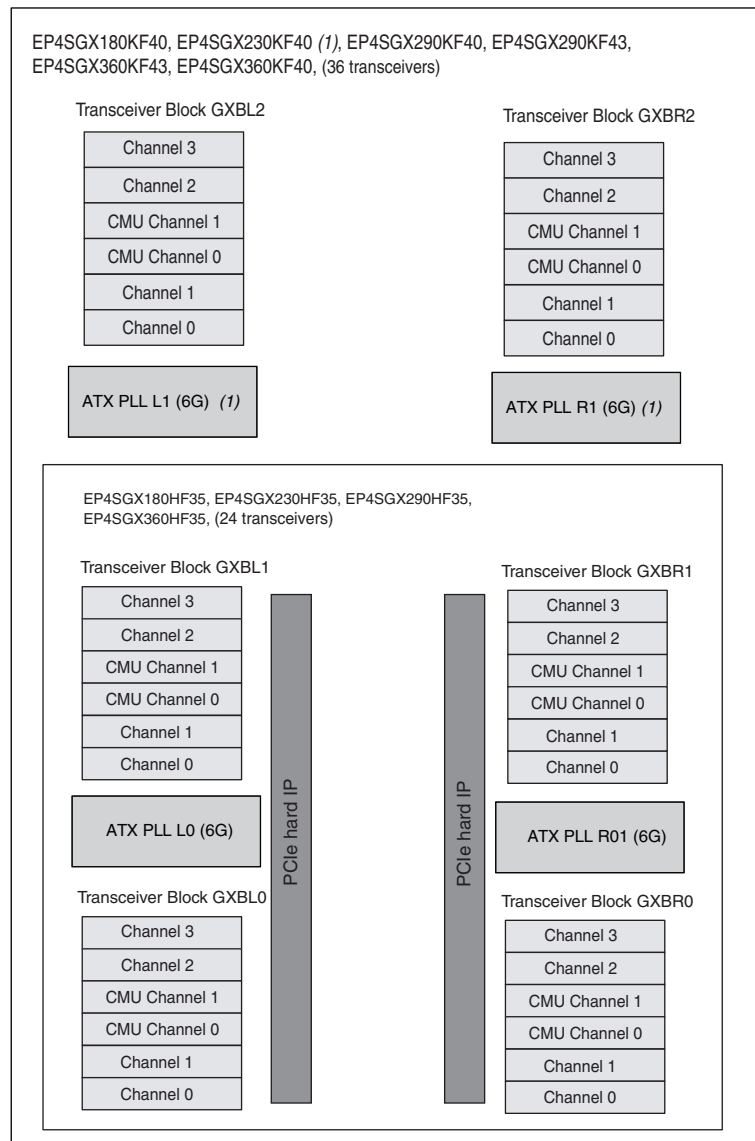


Figure 1-3 shows transceiver channel, PLL, and PCIe hard IP block locations in each Stratix IV GX device that has 24 or 36 transceiver channels (except for the EP4SGX530 device).

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)

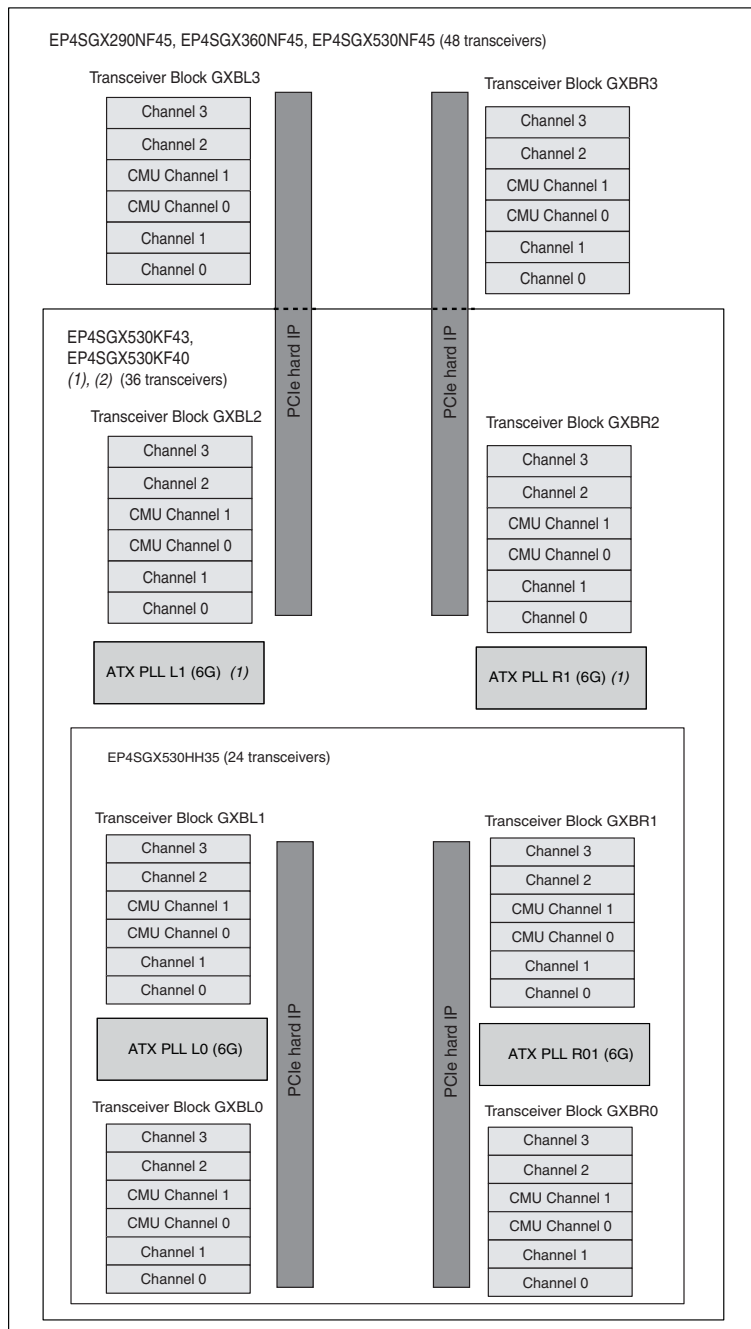


Note to Figure 1-3:

(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)

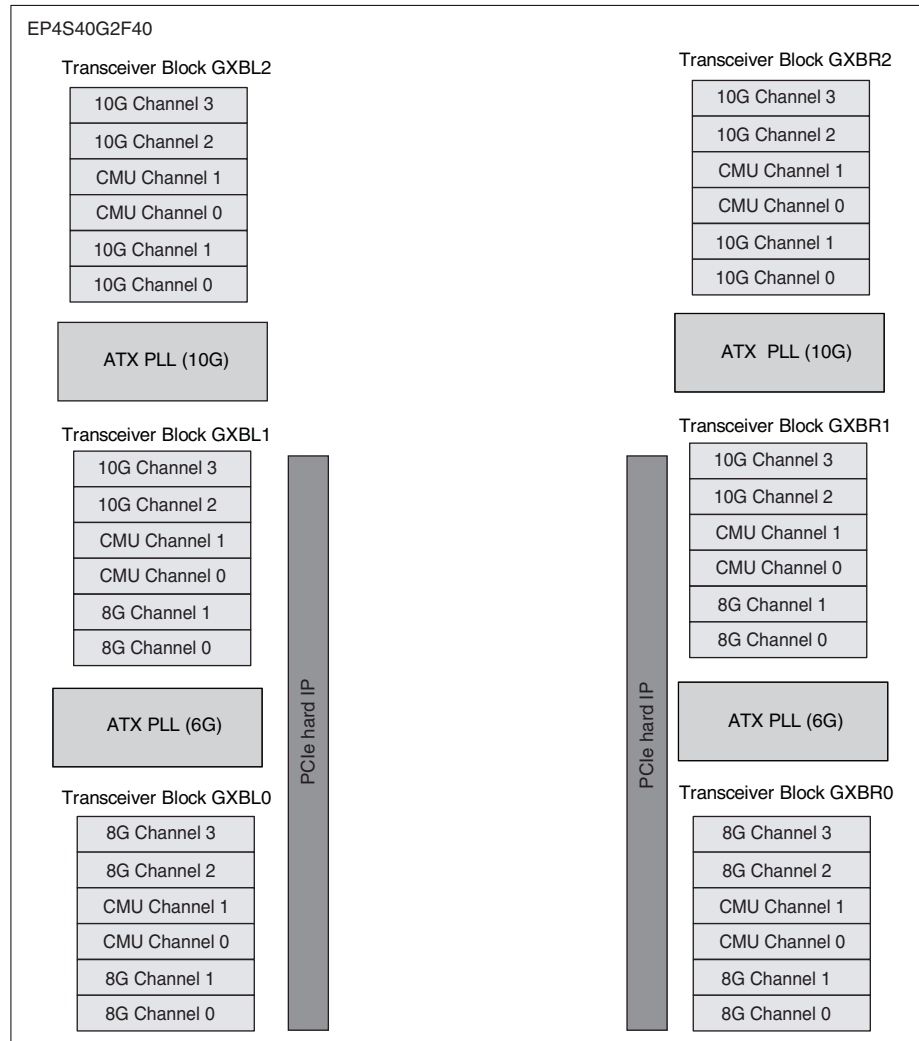
Device Member	Total number of Transceiver Channels	Transceiver Channel Location
EP4S40G2F40	36	12 regular transceiver channels (six 10G and six 8G), located in three transceiver blocks: Left Side—4 in GXBL2, 2 in GXBL1 Right Side—4 in GXBR2, 2 in GXBR1 Refer to Figure 1-5 on page 1-10.
EP4S40G5H40	36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

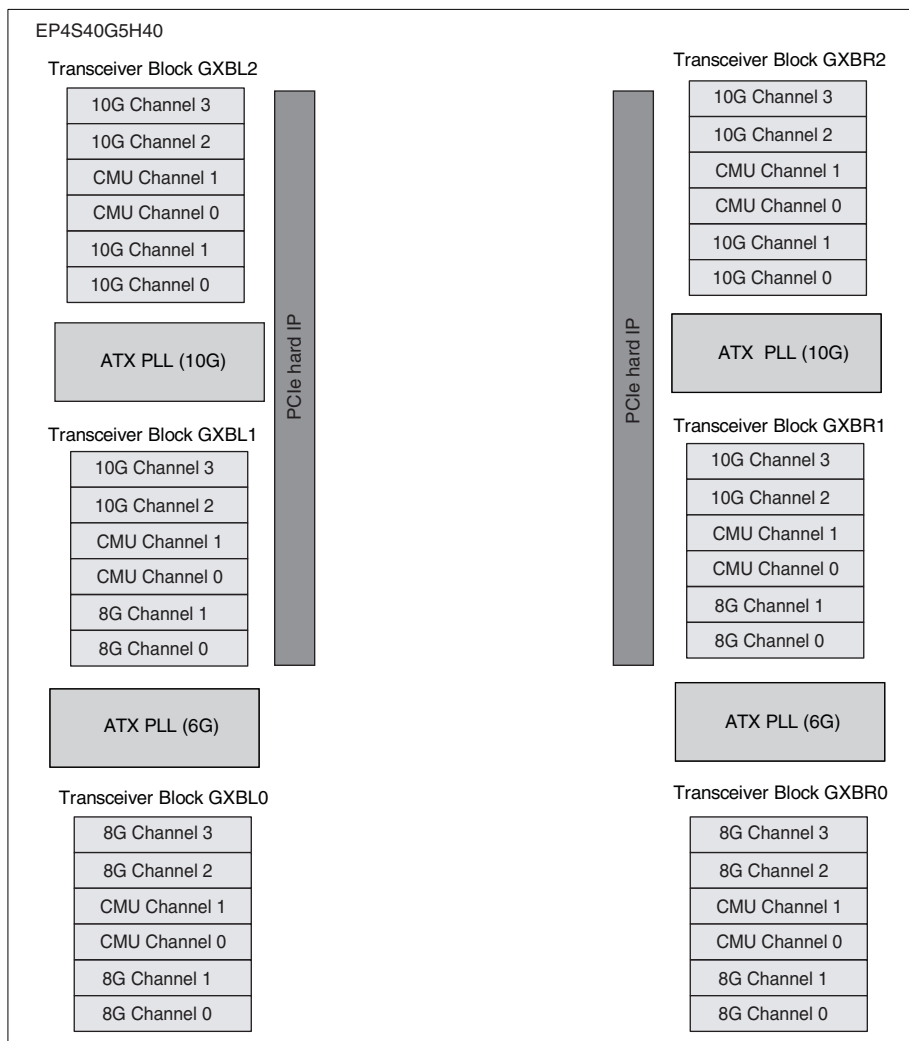


Notes to Figure 1-5:

- (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⁾

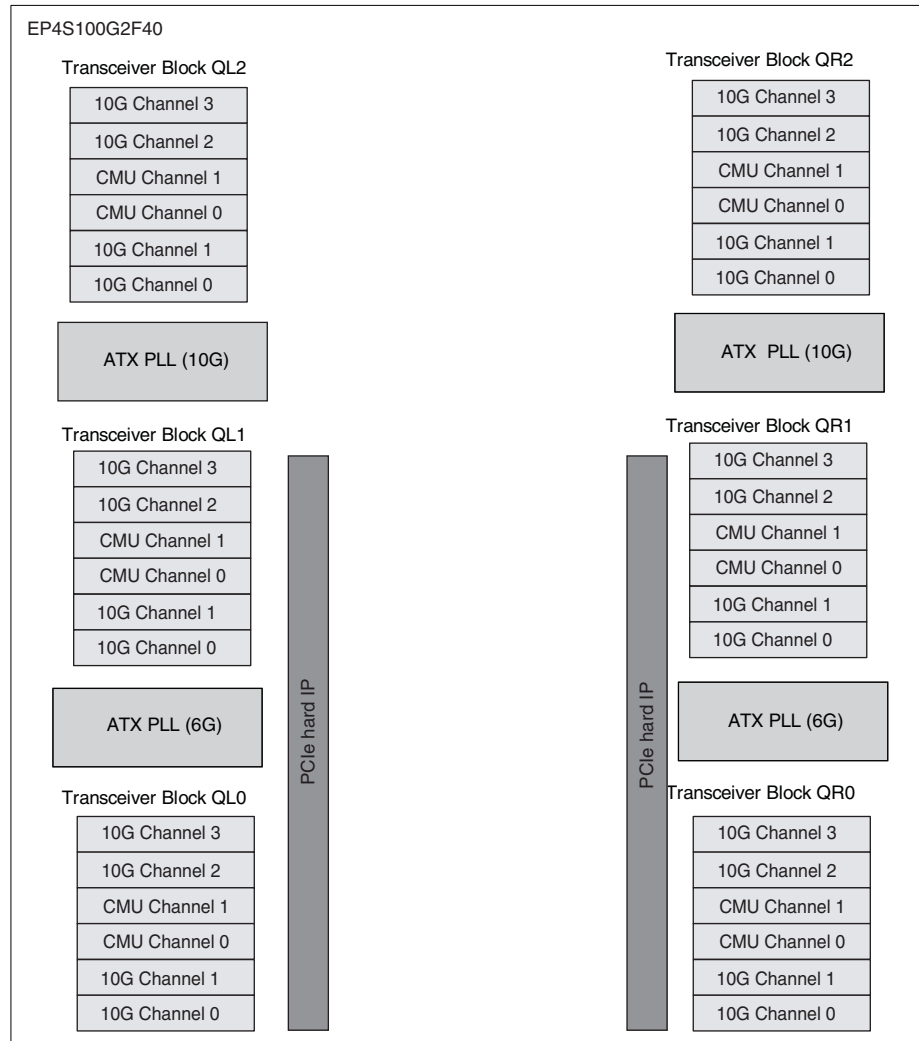


Note to Figure 1-6:

(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

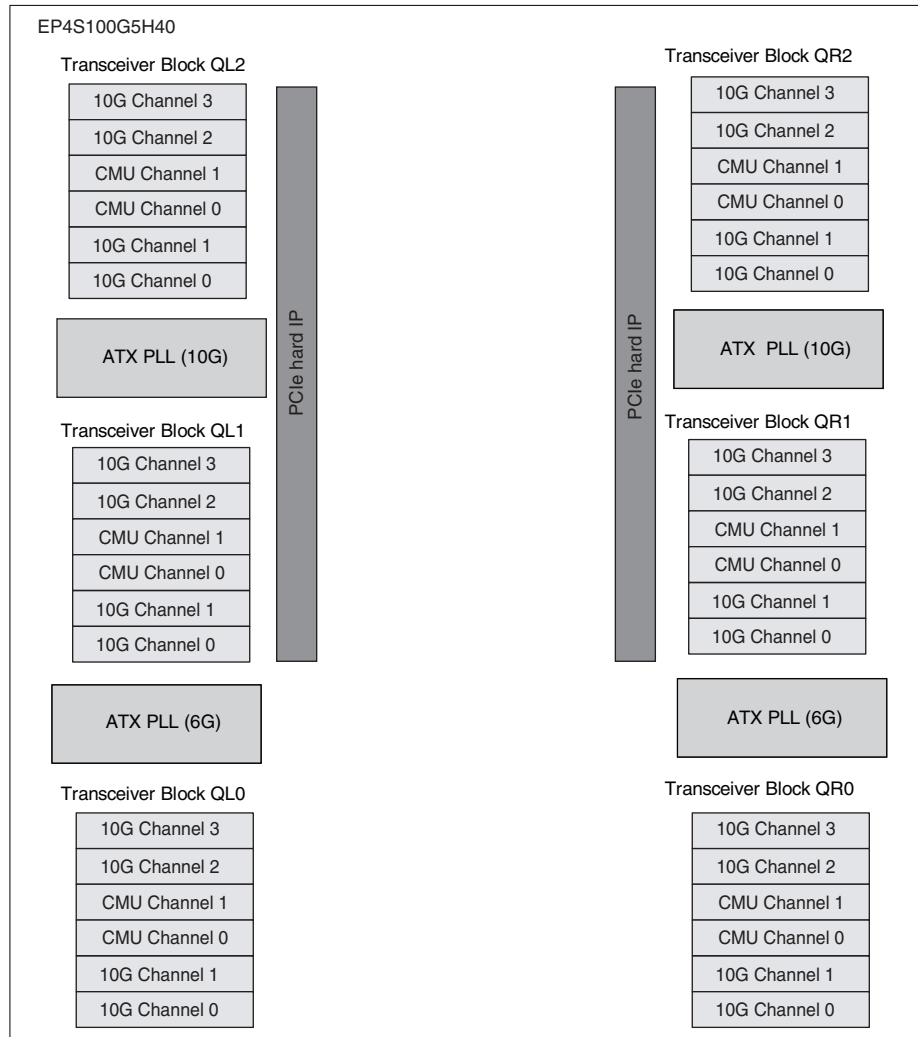


Notes to Figure 1-7:

- (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

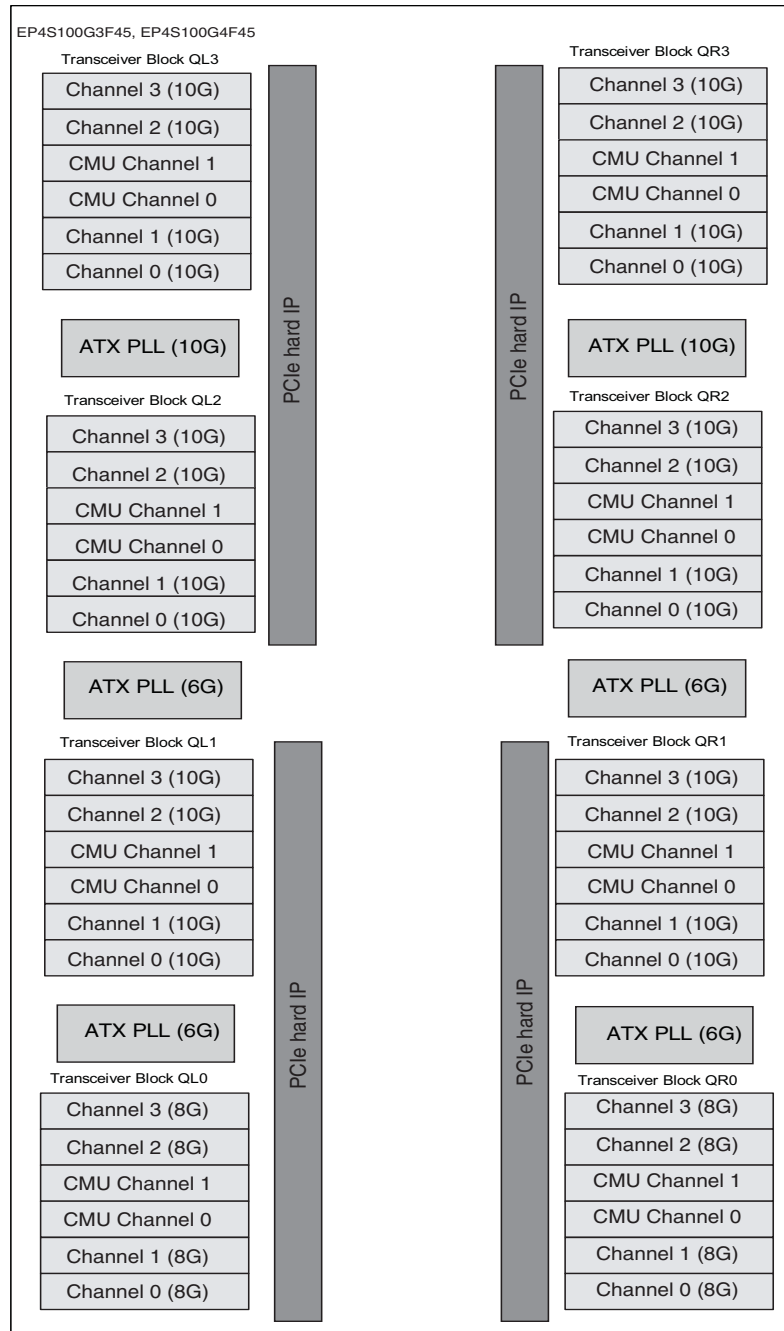


Note to Figure 1-8:

- (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⁾

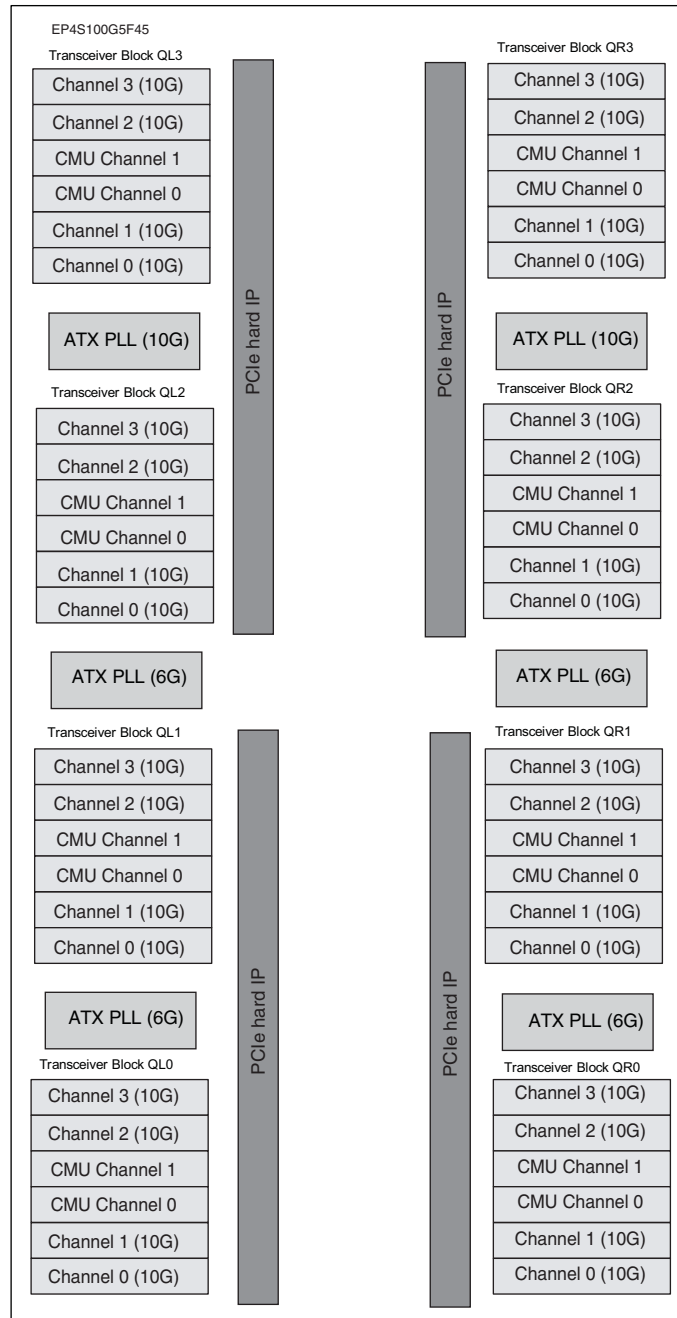


Note to Figure 1-9:

- (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⁾



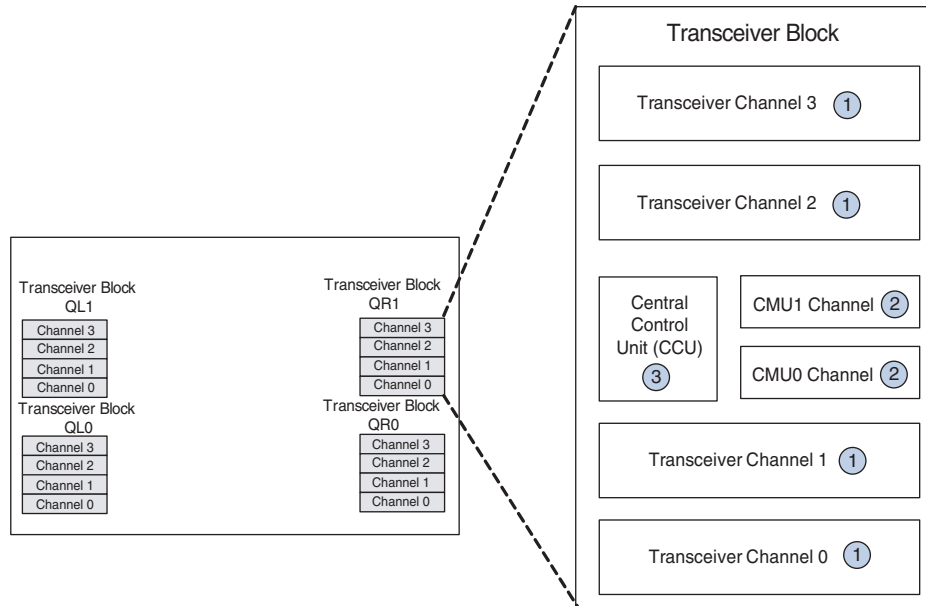
Note to Figure 1-10:

- (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

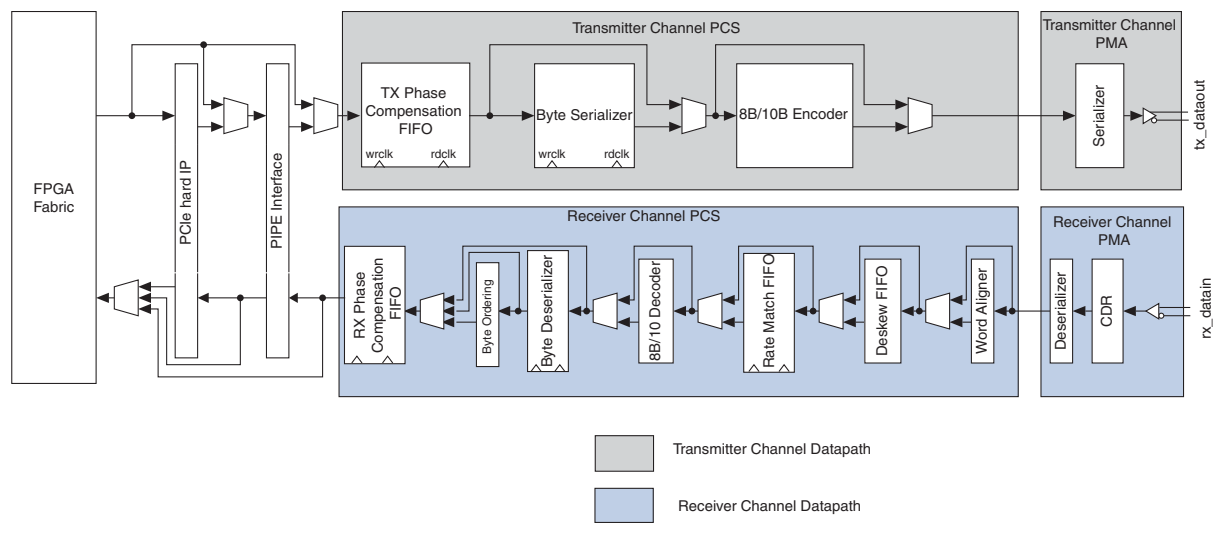
Four transceiver channels and two CMU channels are located in each transceiver block on the left and right sides of the device. Each Stratix IV GT device also has two 10G ATX PLLs that support data rates between 9.9 Gbps and 11.3 Gbps. Additionally, each Stratix IV GT device has two 6G ATX PLLs that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

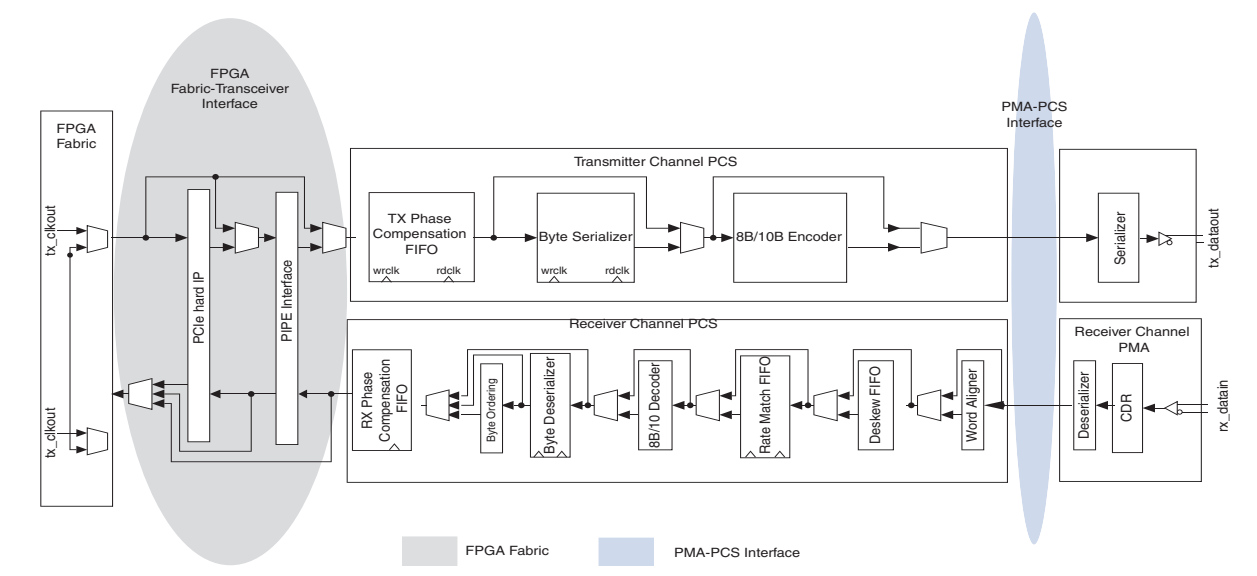
- 8G transceiver channel—supports data rates between 600 Mbps and 8.5 Gbps
- 10G Transceiver Channel—supports data rates between 600 Mbps and 11.3 Gbps

 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

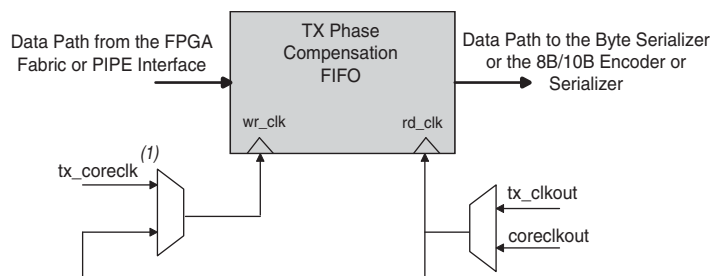
The Stratix IV GX and GT transceiver provides the **Enable low latency PCS mode** option in the ALTGX MegaWizard™ Plug-In Manager. If you select this option, the 8B/10B encoder in the datapath is disabled.

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.

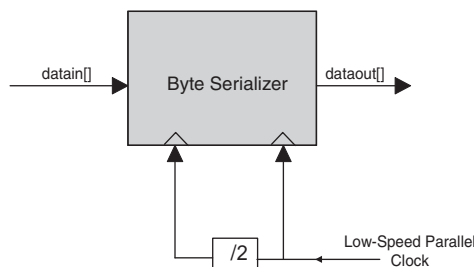


For more information about the maximum frequency limit for the transceiver interface, refer to the *DC and Switching Characteristics for Stratix IV Devices* chapter.

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.

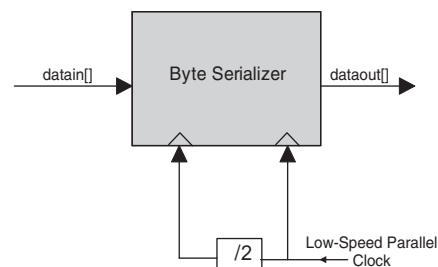
Table 1-9. Input and Output Data Width of the Byte Serializer in Single-Width Mode

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

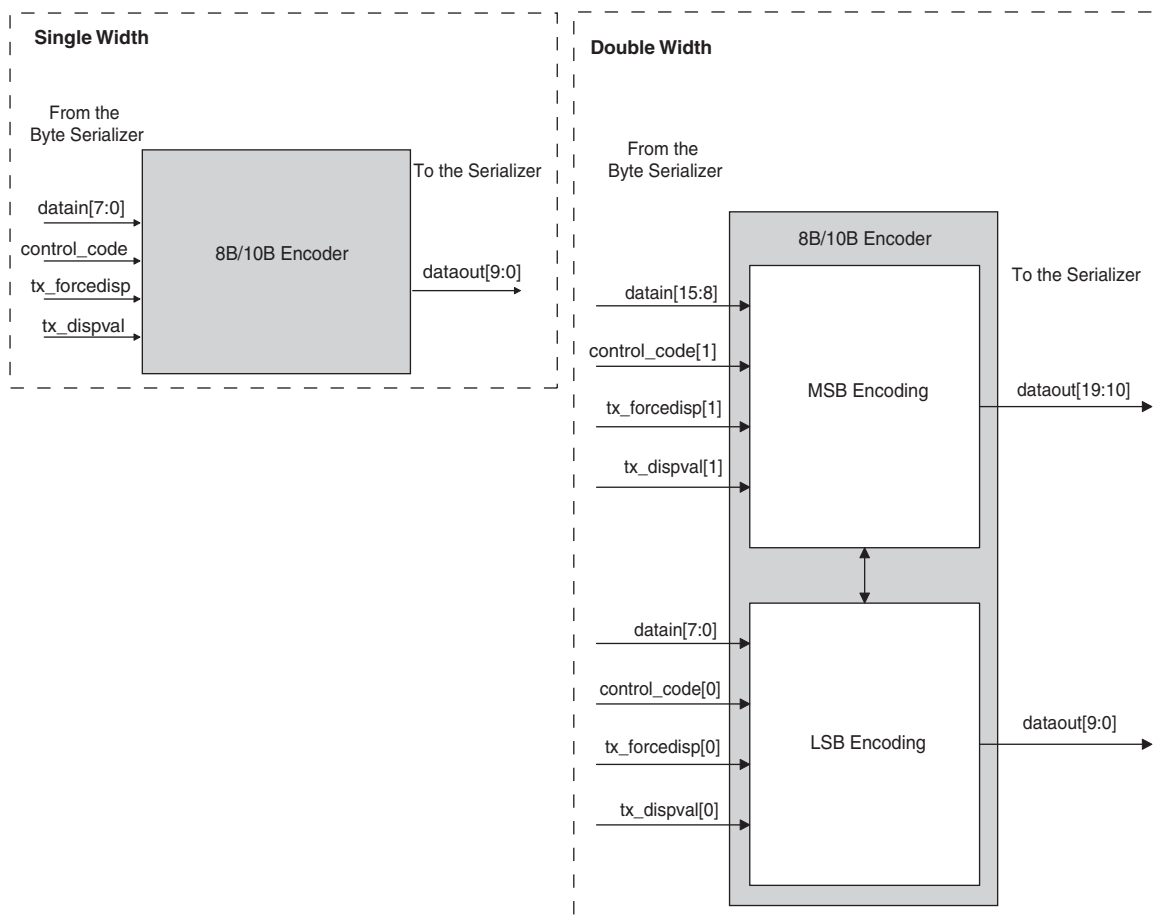
Asserting the `tx_digitalreset` signal resets the byte serializer block.

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



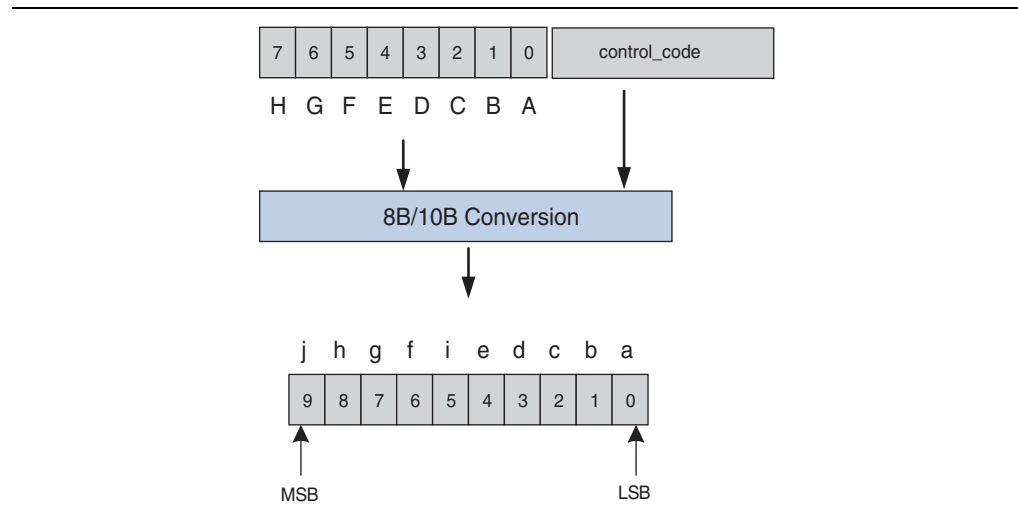
Single-Width Mode

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.

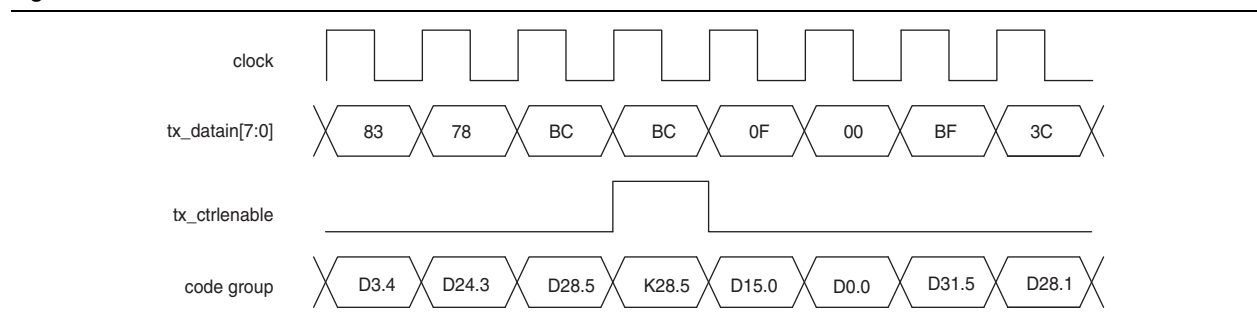
Figure 1-18. 8B/10B Conversion Format




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.


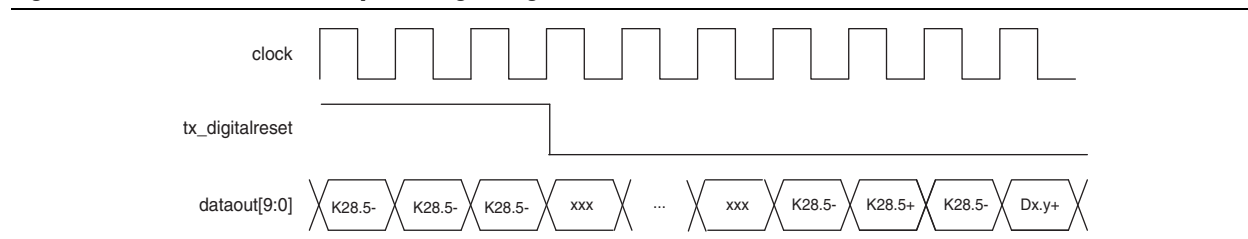
 While $tx_digitalreset$ is asserted, the downstream 8B/10B decoder that receives the data might observe synchronization or disparity errors.

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

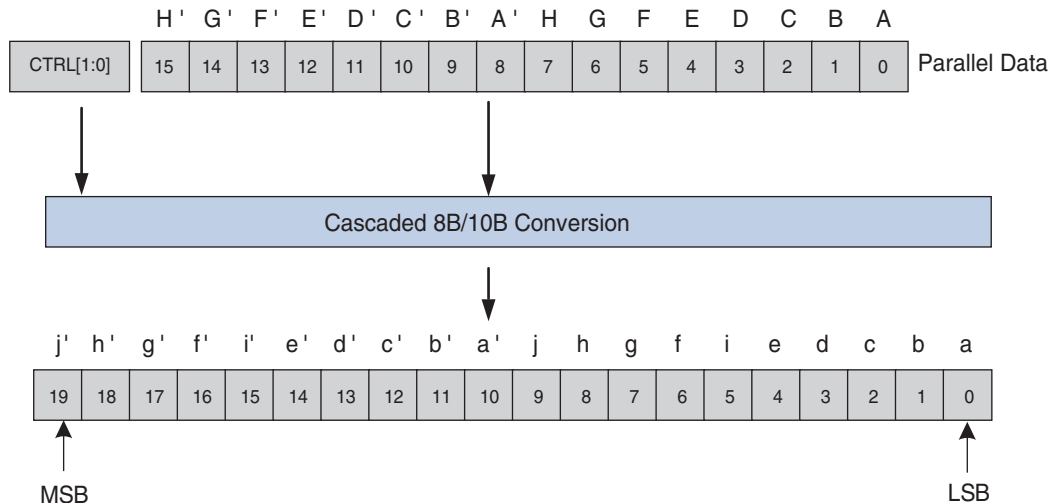


Double-Width Mode

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

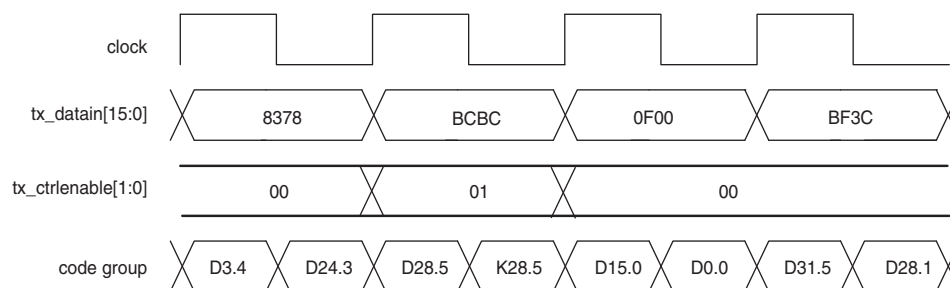


Control Code Encoding

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.


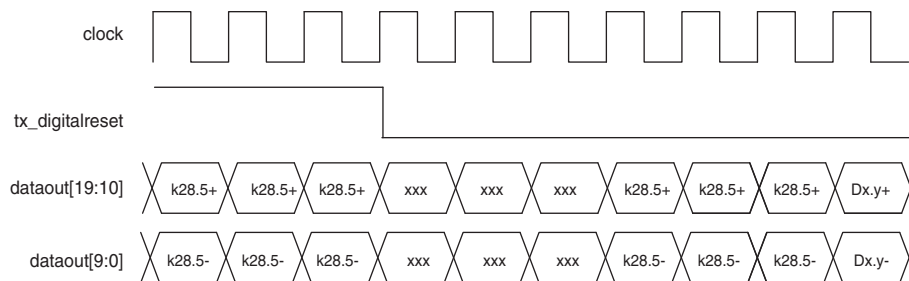
 If the `tx_digitalreset` signal is asserted, the downstream 8B/10B decoder receiving the data might get synchronization or disparity errors.

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

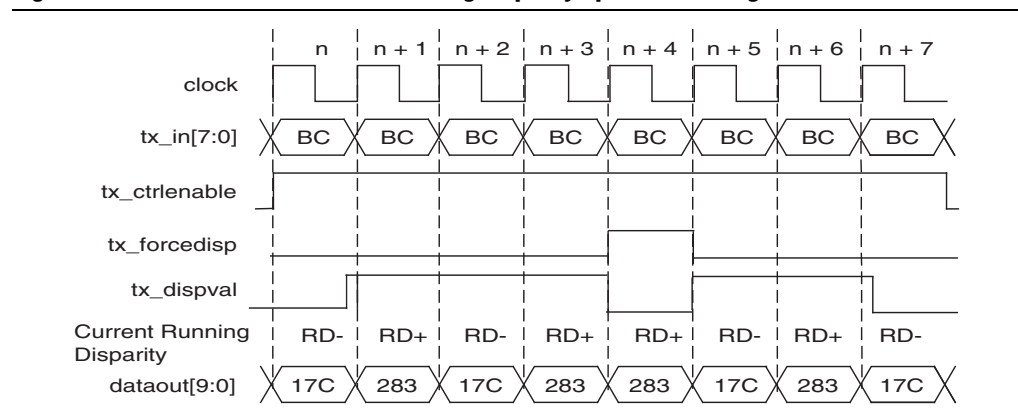
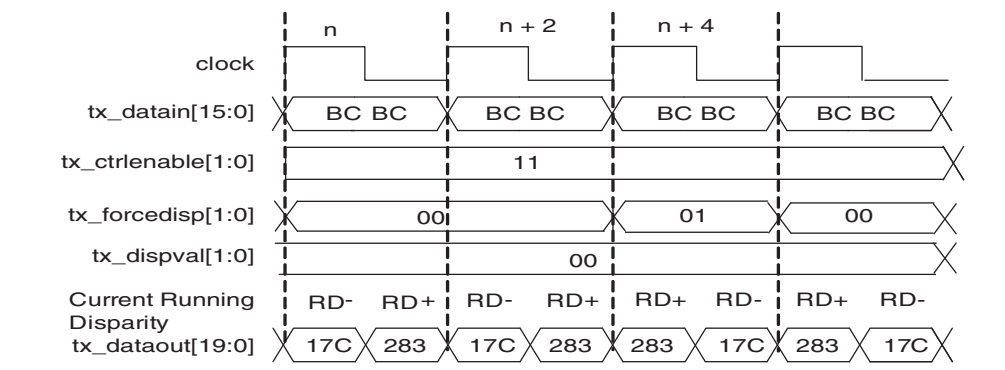


Figure 1-25 shows the current running disparity being altered in Basic double-width mode by forcing a positive disparity on a negative disparity K28.5. In this example, a series of K28.5 are continuously being sent. The stream alternates between a positive ending running disparity (RD+) K28.5 and a negative 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



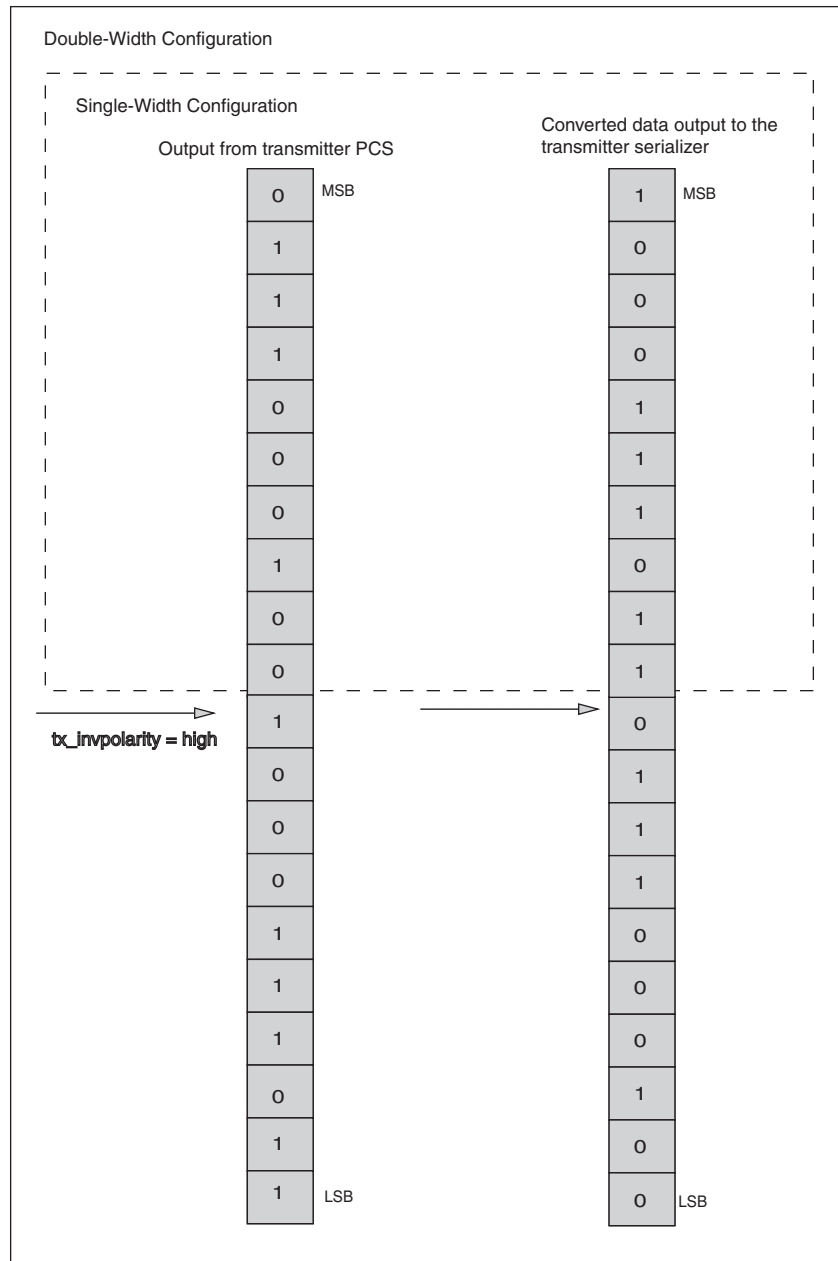
Transmitter Polarity Inversion

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

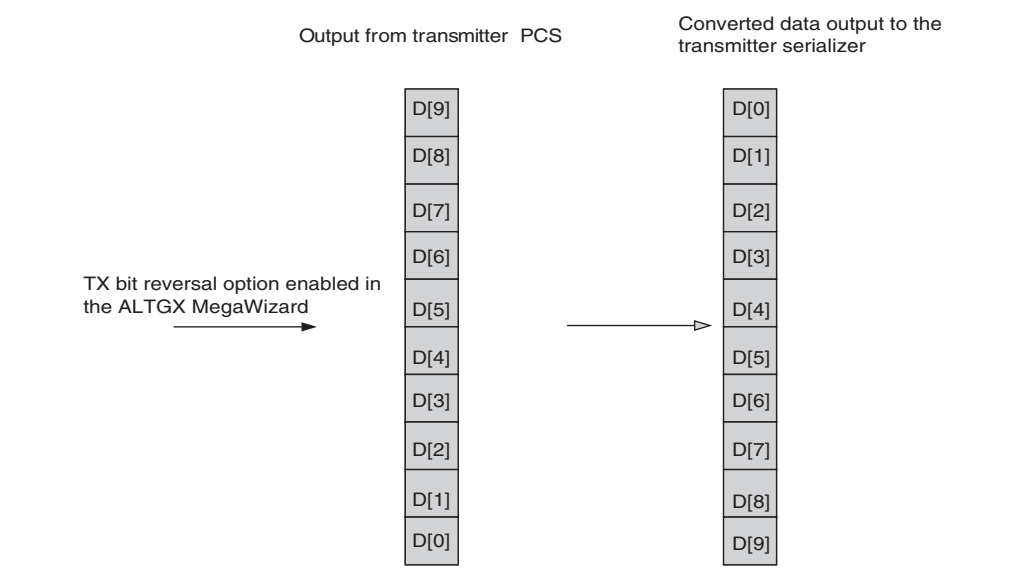
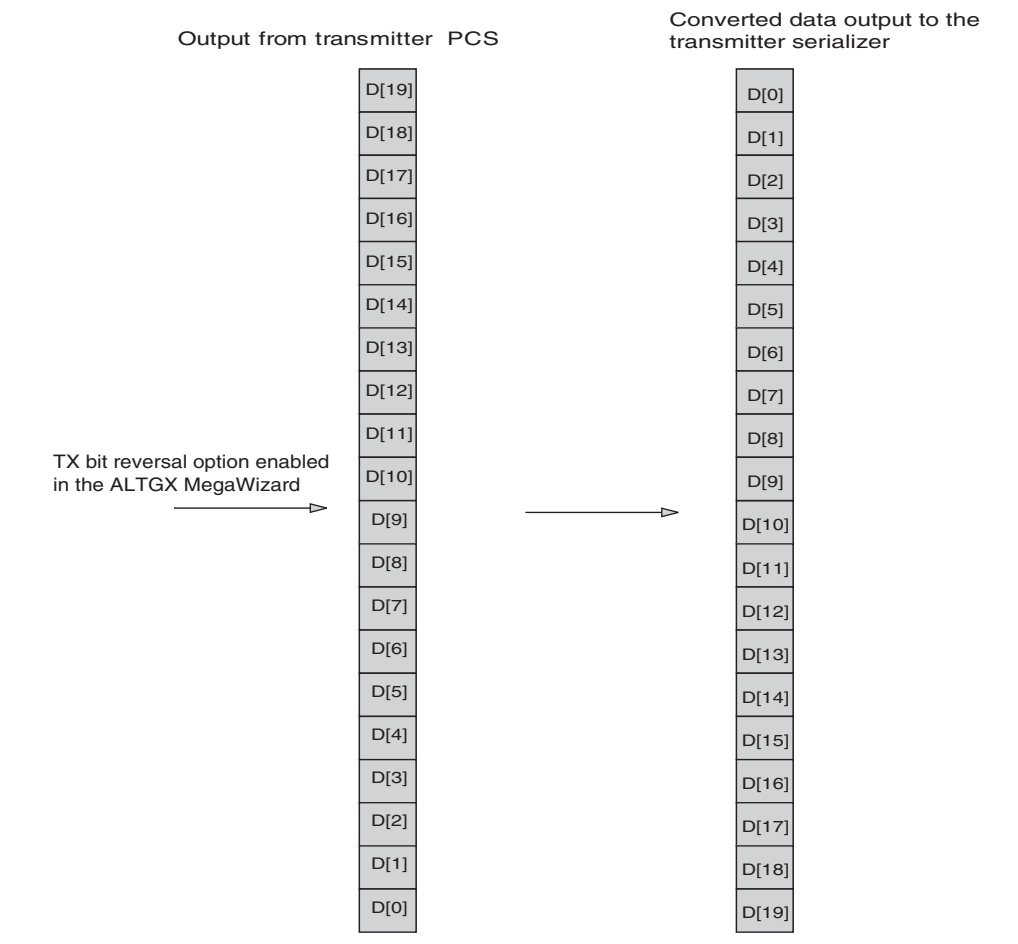


Figure 1-28 shows the transmitter bit reversal feature in Basic double-width mode for a 20-bit wide datapath configuration.

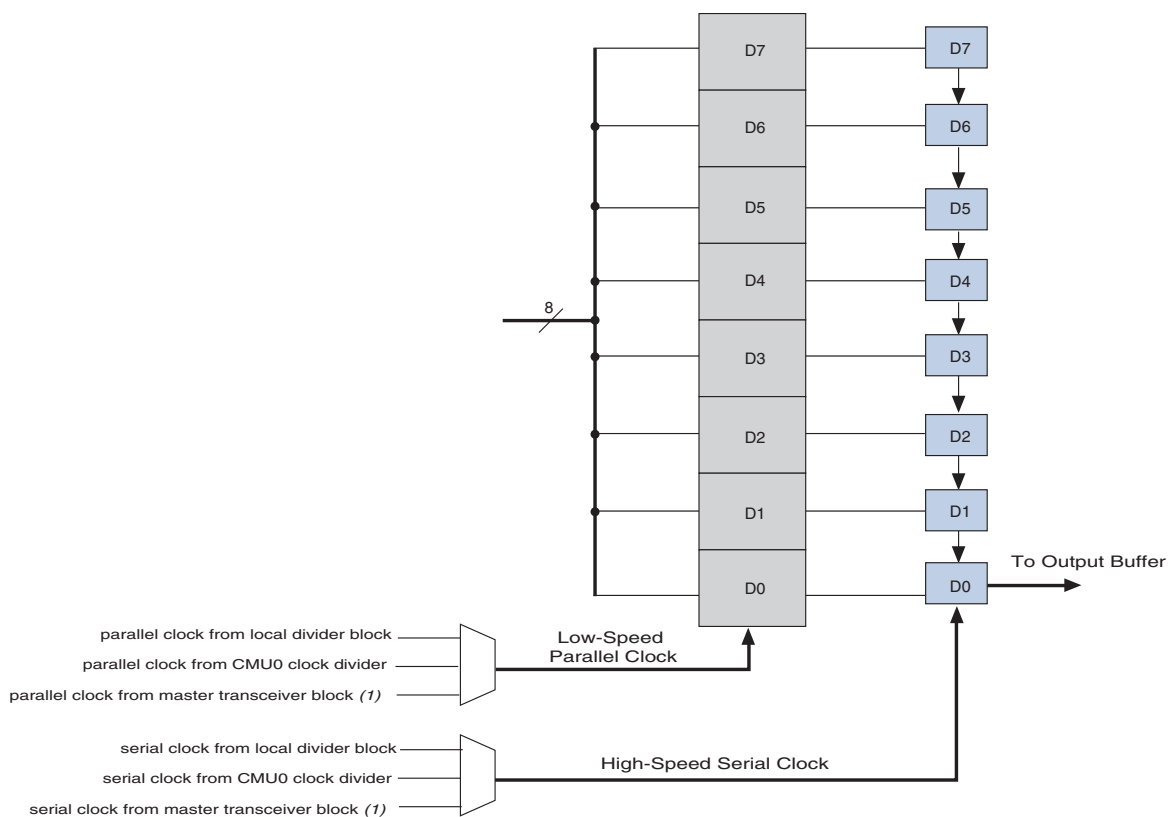
Figure 1-28. Transmitter Bit Reversal Operation in Basic Double-Width Mode



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.

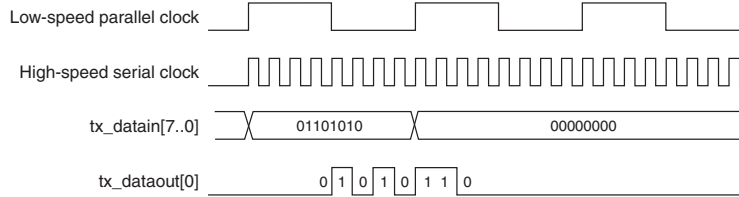
Figure 1-29. Serializer Block in 8-Bit PCS-PMA Interface



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.

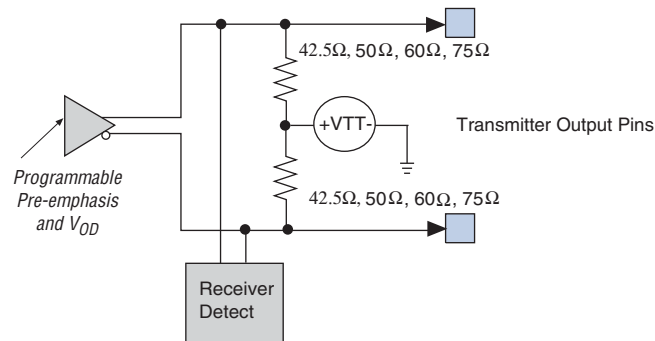
Figure 1-31. Transmitter Output Buffer

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.

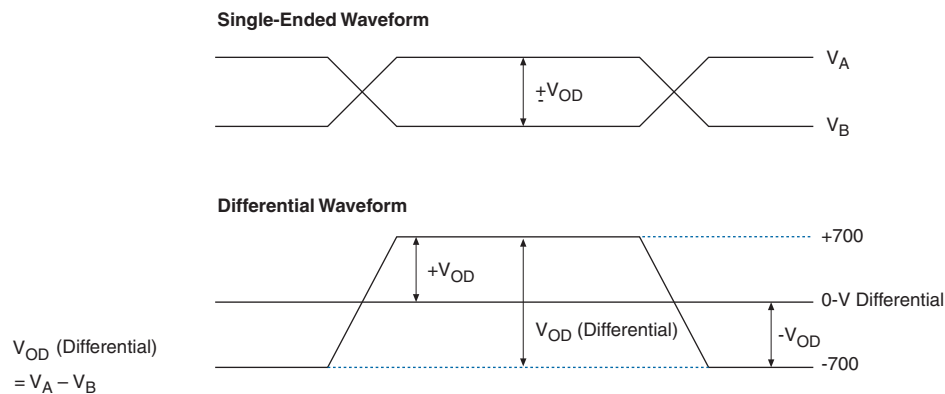
Table 1-15. Stratix IV GX and GT OCT Assignment Settings

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	—

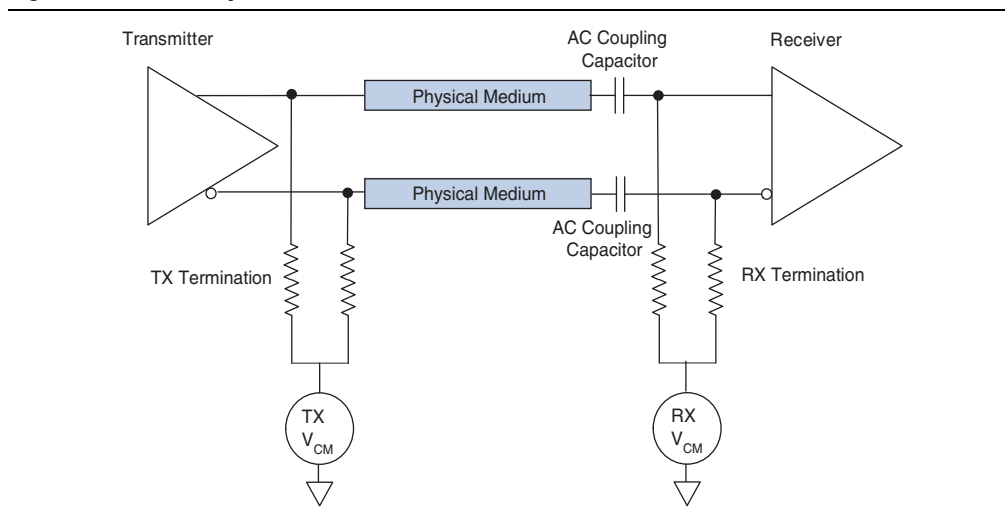
Link Coupling for Stratix IV GX and GT Devices

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:

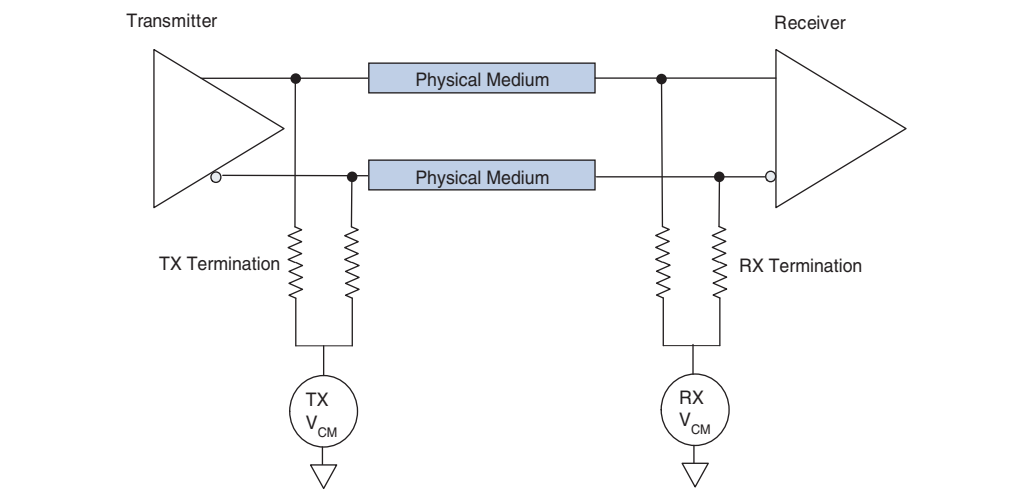
- PCIe
- Gigabit Ethernet
- Serial RapidIO
- XAUI
- SDI

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](#).

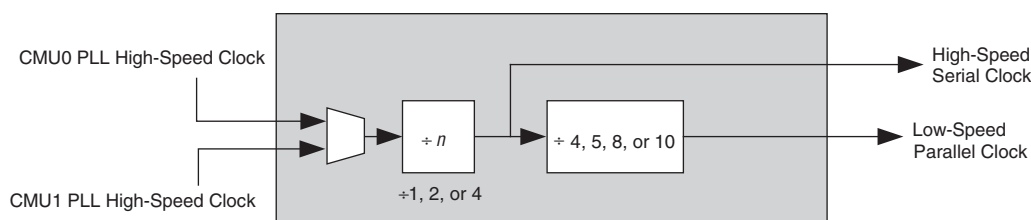
 For more information about using the `tx_forceelecidle` signal under different power states, refer to the PCIe specification 2.0.

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

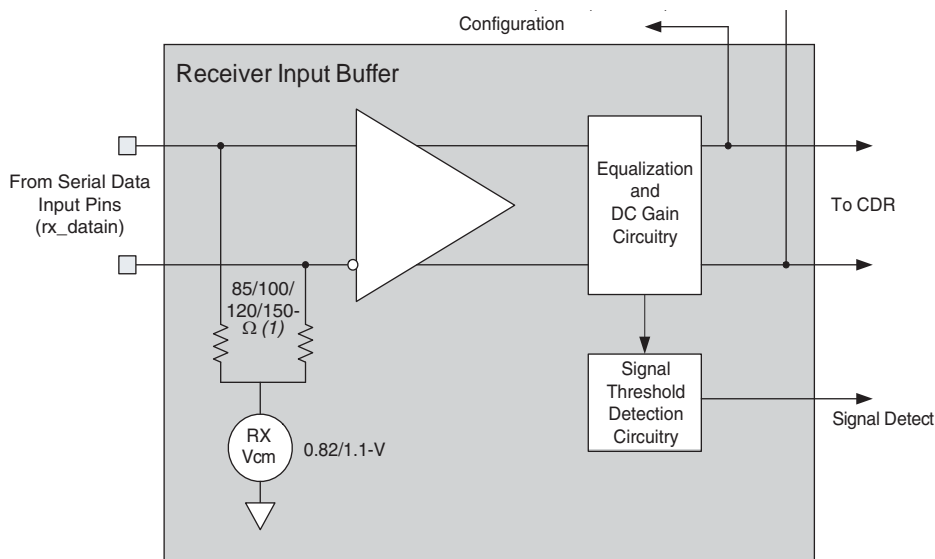
The receiver datapath is very flexible and allows multiple configurations, depending on the selected functional mode. You can configure the receiver datapath using the ALTGX MegaWizard Plug-In Manager.

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 ($R_x V_{CM}$), 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.

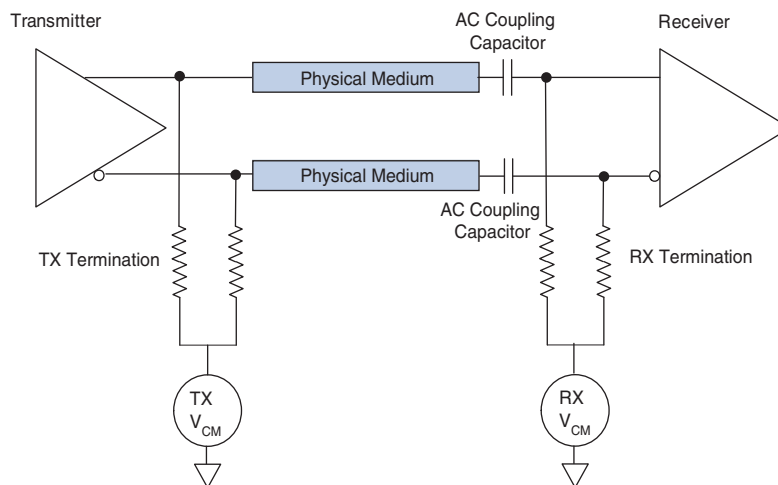
Link Coupling for Stratix IV GX Devices

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.

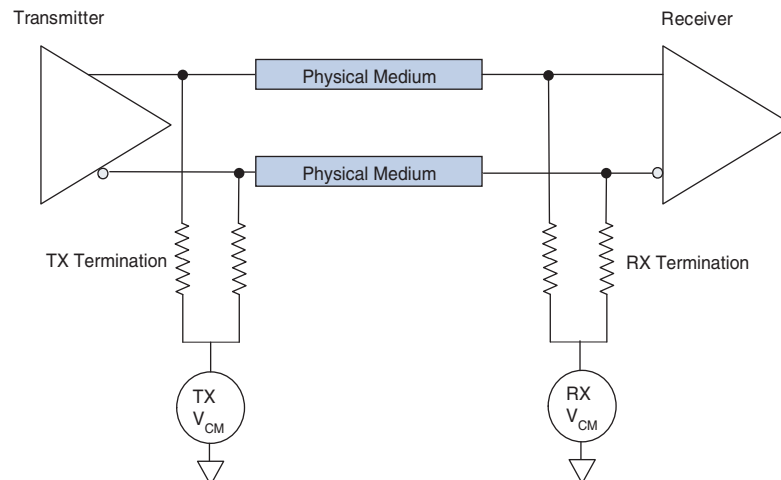
The following protocols supported by Stratix IV GX and GT devices mandate AC-coupled links:

- 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:

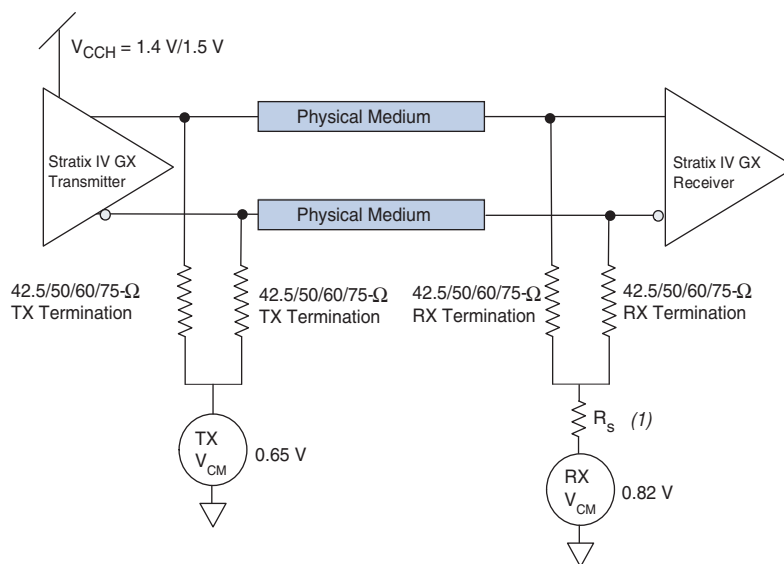
- Basic single- and double-width
- (OIF) CEI PHY interface

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

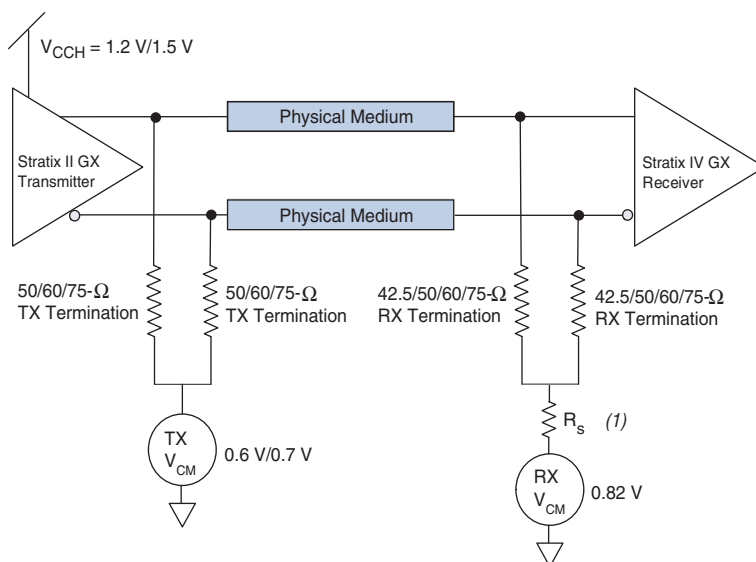
Transmitter (Stratix IV GX) Settings				Receiver (Stratix IV GX) Settings		
Data Rate	V_{CCH} (1)	TX V_{CM}	Differential Termination	Data Rate	RX V_{CM}	Differential Termination
600-8500 Mbps	1.4 V/1.5 V	0.65 V	85/100/120/150 Ω	600-8500 Mbps	0.82 V	85/100/120/150 Ω

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

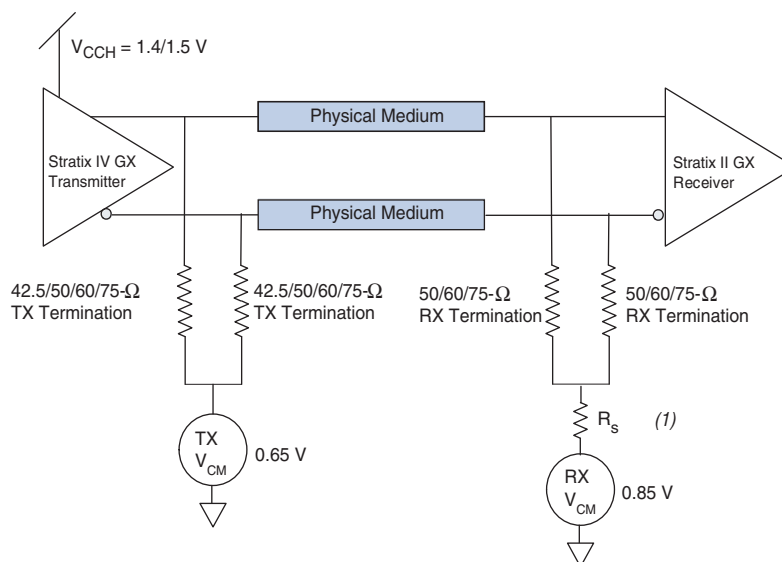
Transmitter (Stratix II GX) Settings				Receiver (Stratix IV GX) Settings		
Data Rate	V_{CCH} (1)	$\text{TX } V_{CM}$ (1)	Differential Termination	Data Rate	$\text{RX } V_{CM}$	Differential Termination
600-6375 Mbps	1.5 V (1.5 V PCML)	0.6 V/0.7 V	100/120/150 Ω	600-6375 Mbps	0.82 V	100/120/150 Ω

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

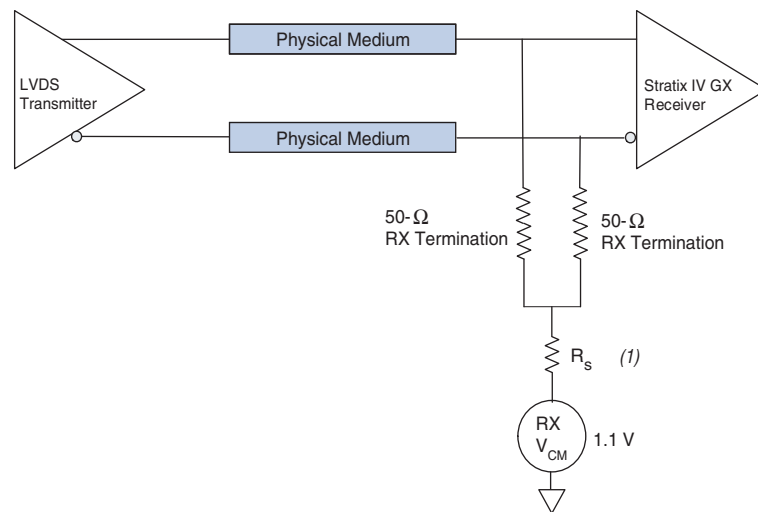
Transmitter (Stratix IV GX) Settings				Receiver (Stratix II GX) Settings			
Data Rate	V_{CCH} (1)	TX V_{CM}	Differential Termination	Data Rate	I/O Standard	RX V_{CM}	Differential Termination
600-6375 Mbps	1.4/1.5 V	0.65 V	100/120/150 Ω	600-6375 Mbps	1.4/1.5 V PCML	0.85 V	100/120/150 Ω

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.

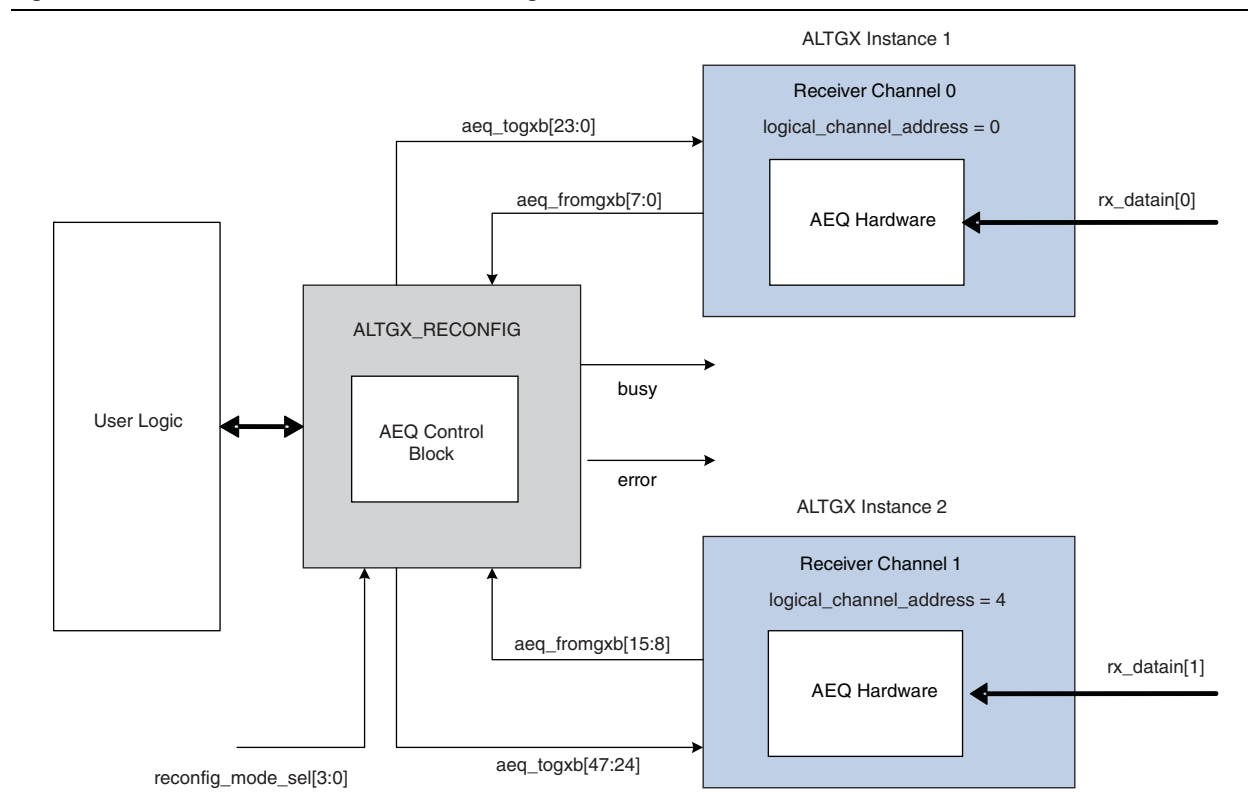
When you select AEQ, two ports, `aeq_fromgxb[]` and `aeq_togxb[]`, become available on the ALTGX and ALTGX_RECONFIG instances. These ports provide an interface between the PMA of the receiver channel and the AEQ control block in the ALTGX_RECONFIG MegaWizard Plug-In Manager.



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.

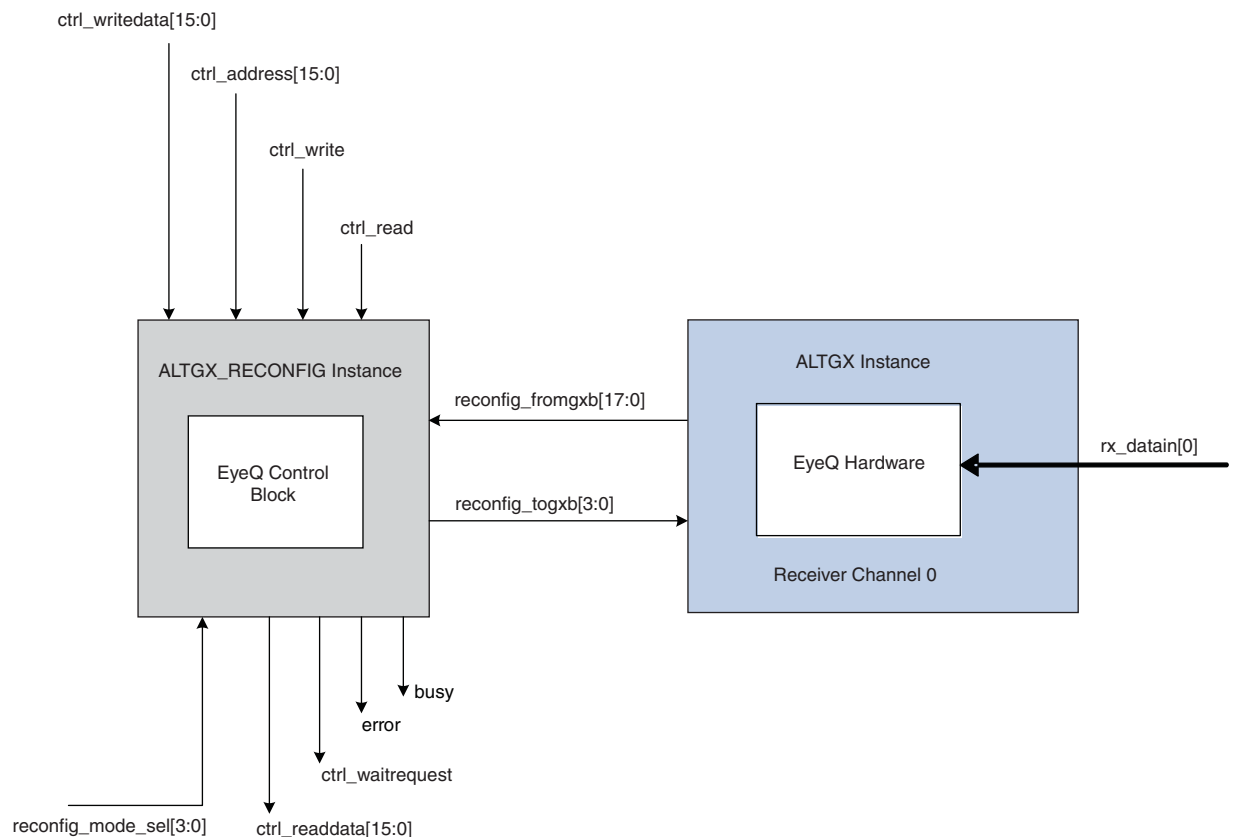
 The EyeQ hardware is available for both regular transceiver channels and CMU channels.

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

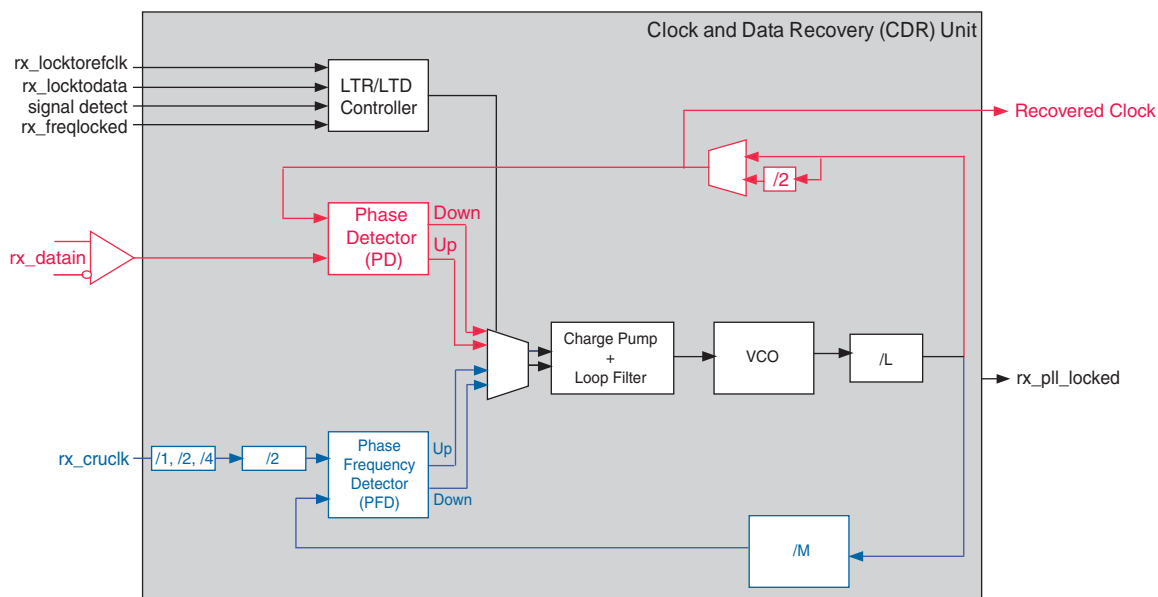


 For more information about using the EyeQ feature, refer to the *Dynamic Reconfiguration in Stratix IV Devices* chapter.

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

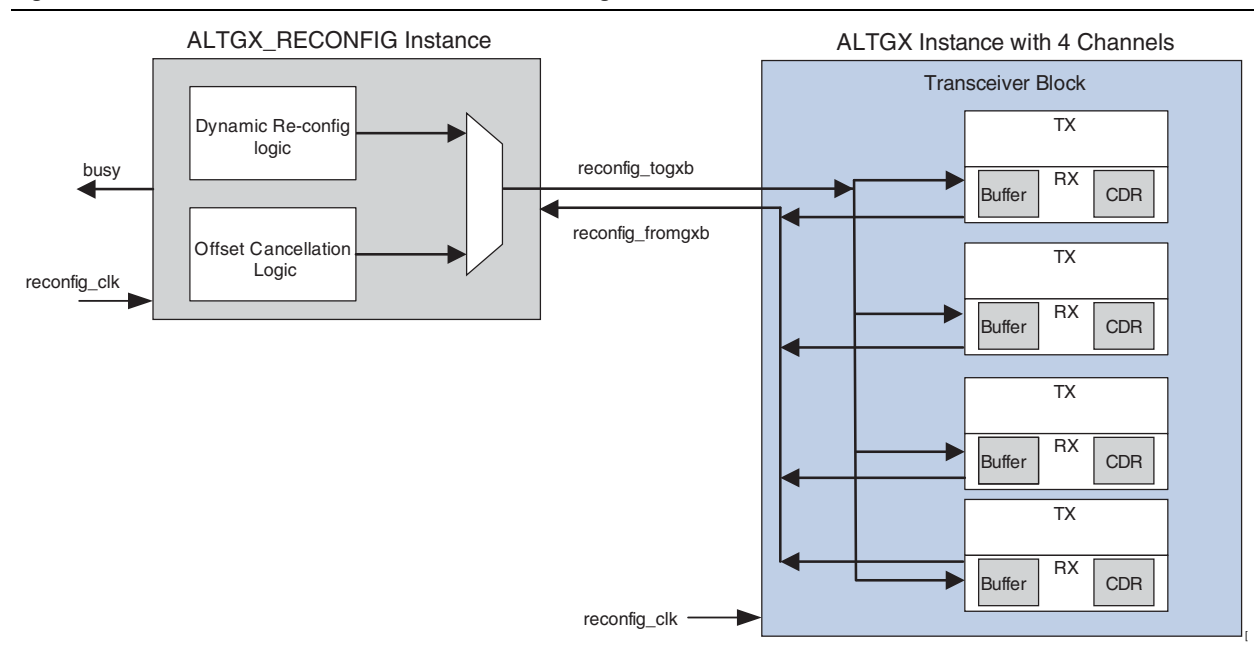
As silicon progresses towards smaller process nodes, the performance of circuits at these smaller nodes depends more on process variations. These process variations result in analog voltages that can be offset from the required ranges. Offset cancellation logic corrects these offsets. The receiver buffer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

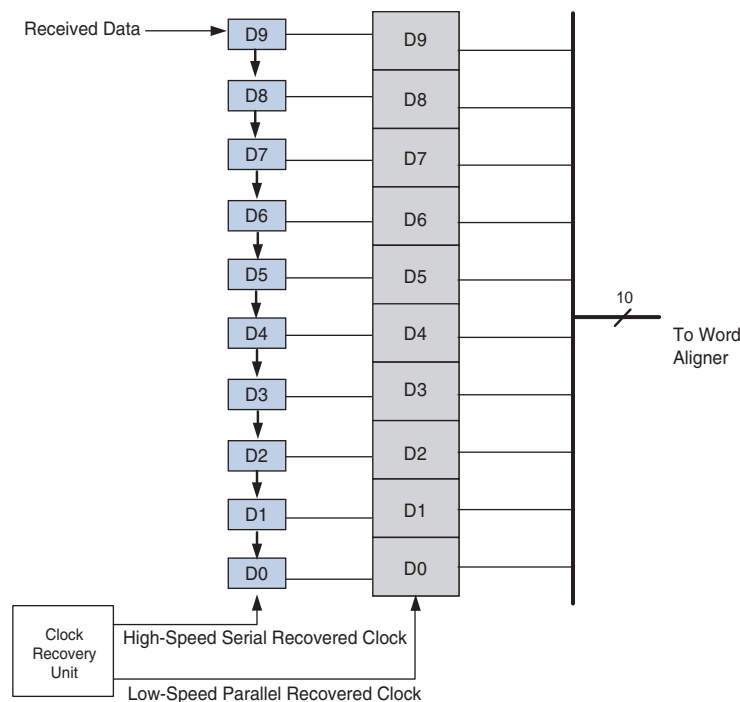
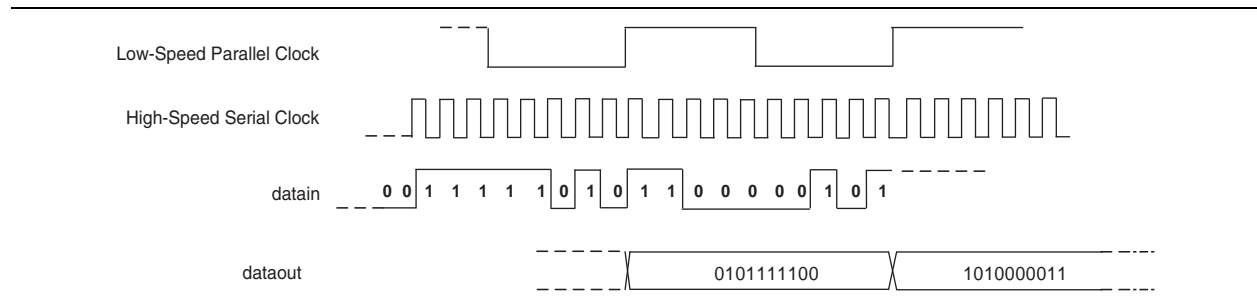


Figure 1-48 shows the serial bit order of the deserializer block input and the parallel data output of the deserializer block in single-width mode with a 10-bit deserialization factor. The serial stream (0101111100) is deserialized to a value 10'h17C. The serial data is assumed to be received LSB to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

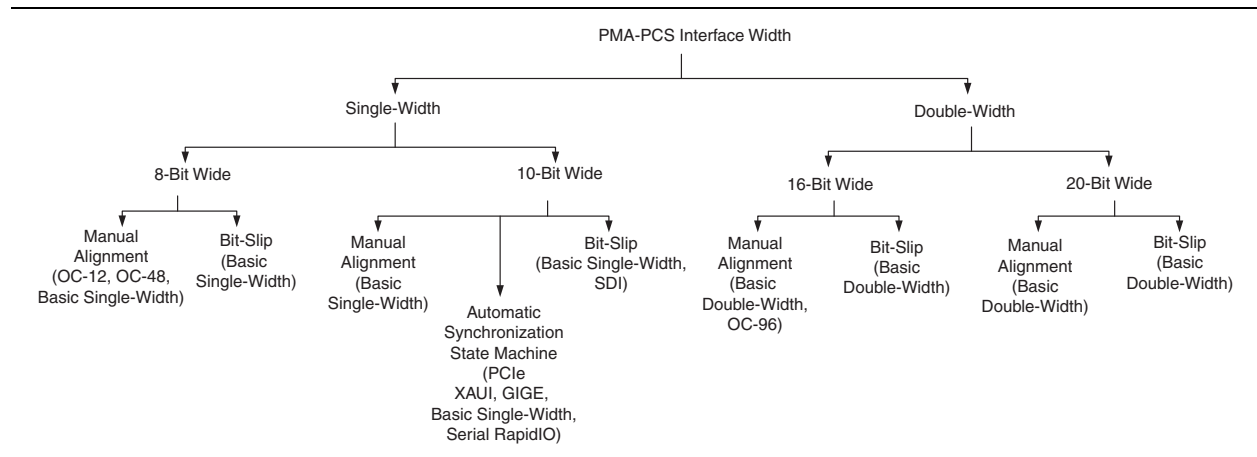
- Synchronization state machine in functional modes such as PCIe, XAUI, GIGE, Serial RapidIO, and Basic single-width
- Programmable run length violation detection in all functional modes
- Receiver polarity inversion in all functional modes except PCIe
- Receiver bit reversal in Basic single-width and 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.


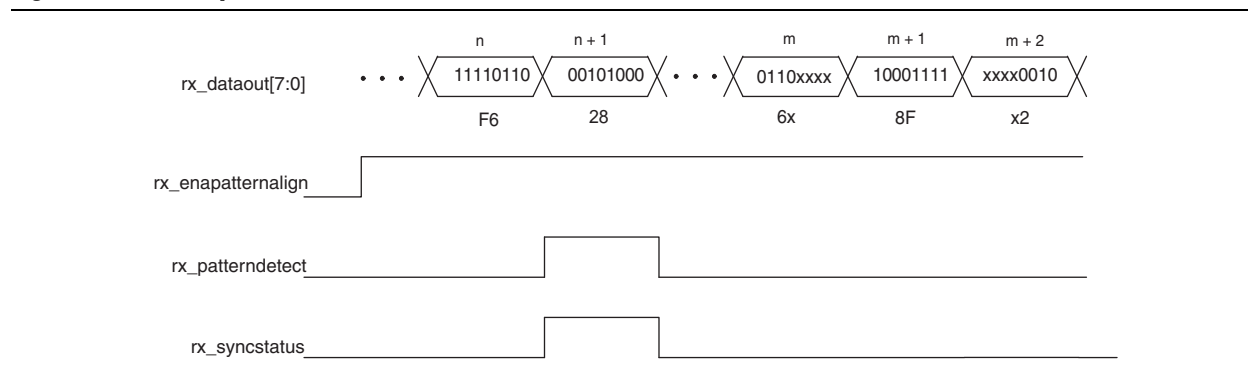
 For the word aligner to re-synchronize to a new word boundary, you must de-assert `rx_enaatternalign` and re-assert it again to create a rising edge. After a rising edge on the `rx_enaatternalign` signal, if the word alignment pattern is found in a different word boundary, the word aligner re-synchronizes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



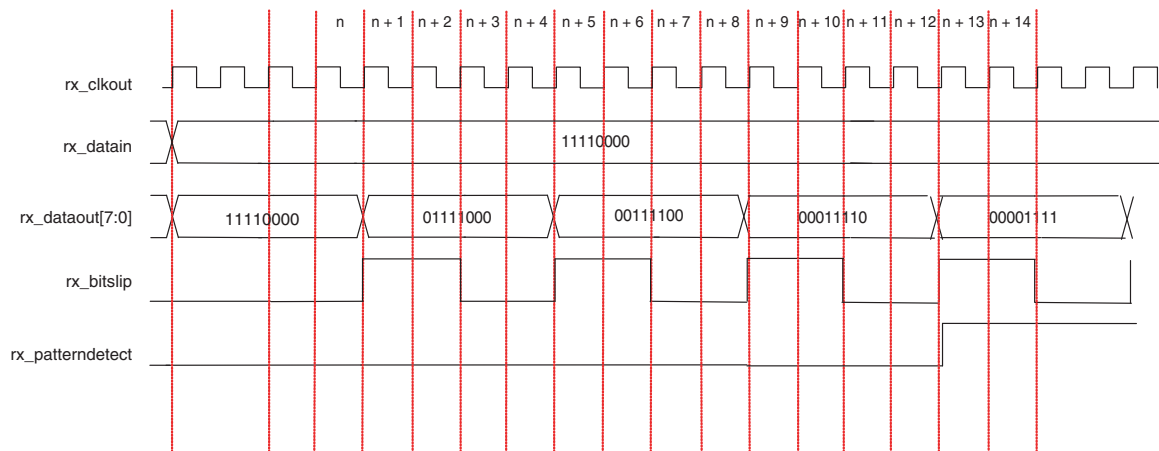
Bit-Slip Mode Word Aligner with 8-Bit PMA-PCS Interface Modes

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'b000011111`, 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

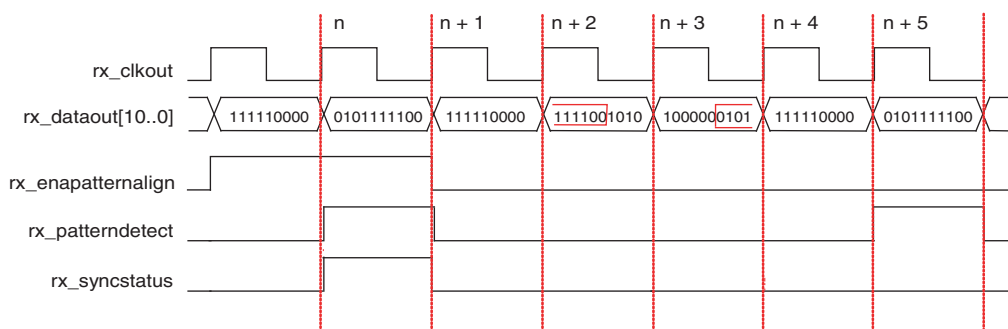
In Basic single-width mode with a 10-bit PMA-PCS interface, you can configure the word aligner in manual alignment mode by selecting the **Use manual word alignment mode** option in the ALTGX MegaWizard Plug-In Manager.


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.

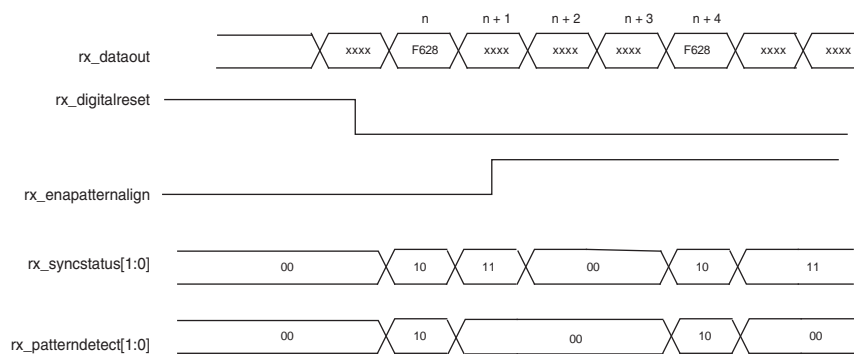
Two status signals, `rx_syncstatus` and `rx_patterndetect`, with the same latency as the datapath, are forwarded to the FPGA fabric to indicate word aligner status.

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

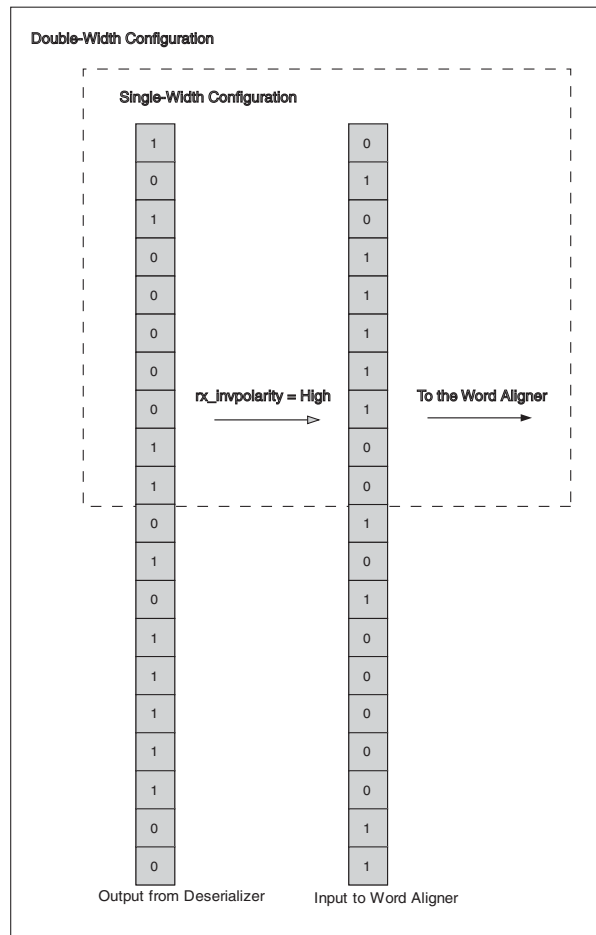
The positive and negative signals of a serial differential link are often erroneously swapped during board layout. Solutions like board re-spin or major updates to the PLD logic can be expensive. The receiver polarity inversion feature is provided to correct this situation.

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

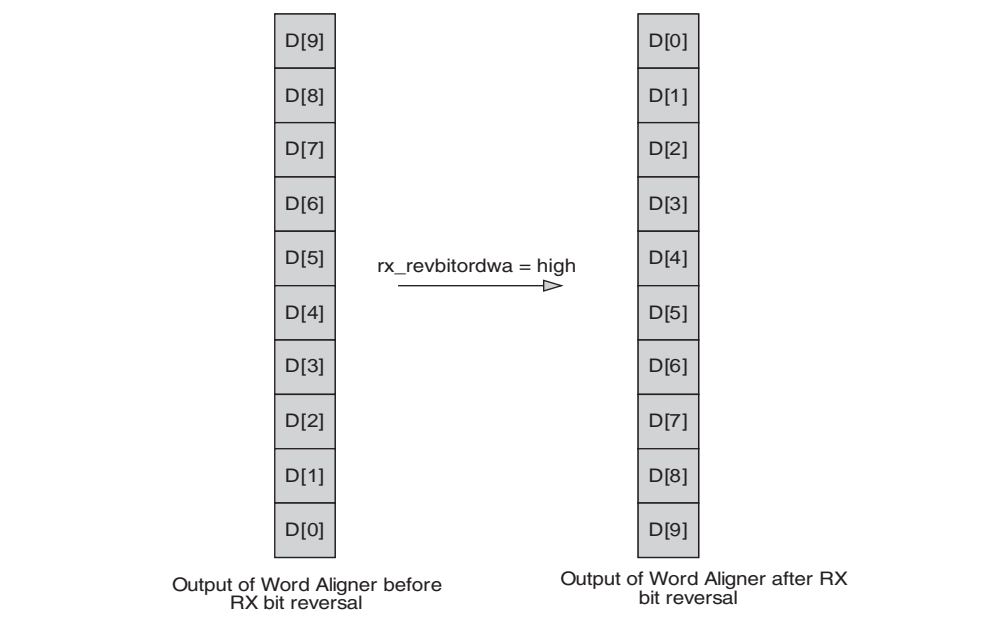
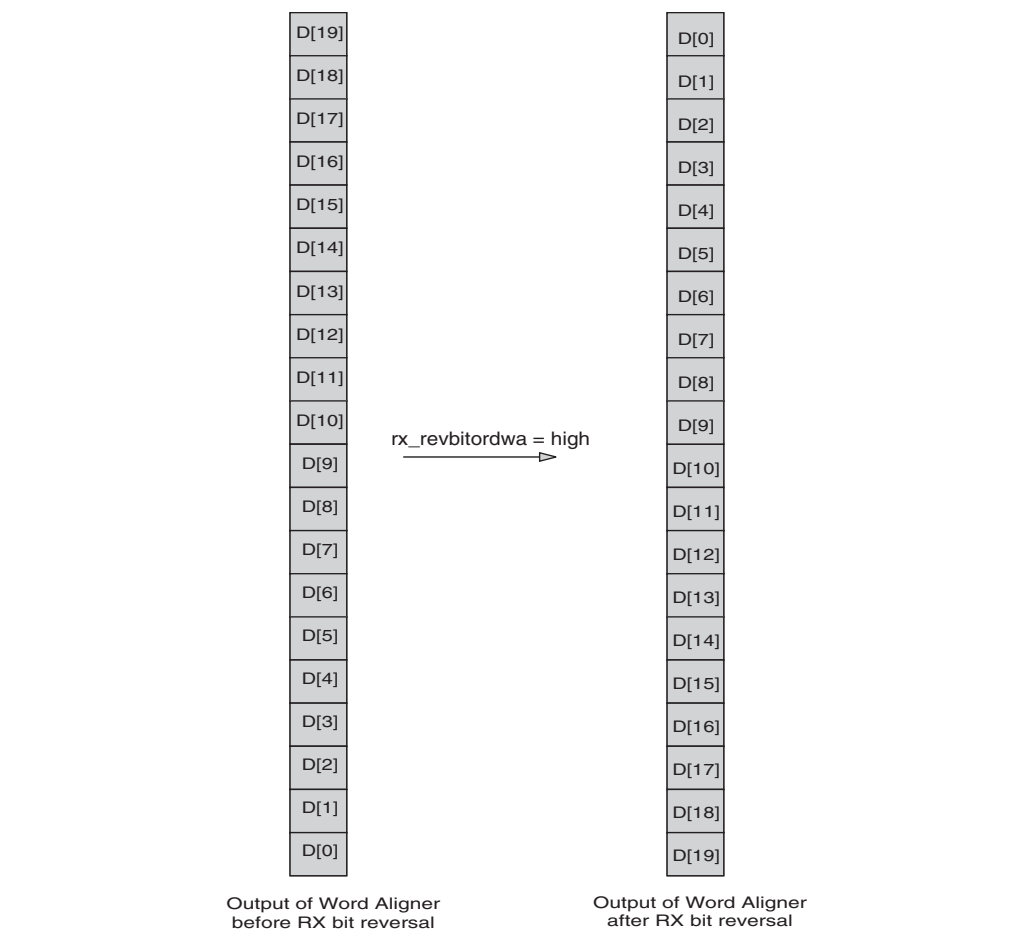


Figure 1–56 shows the receiver bit reversal feature in Basic double-width 20-bit wide datapath configurations.

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.

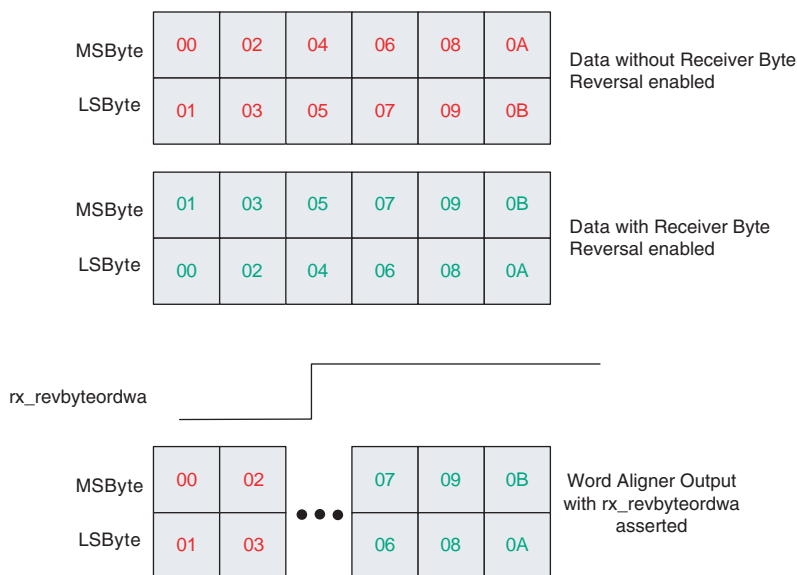
Receiver Byte Reversal in Basic Double-Width Modes

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

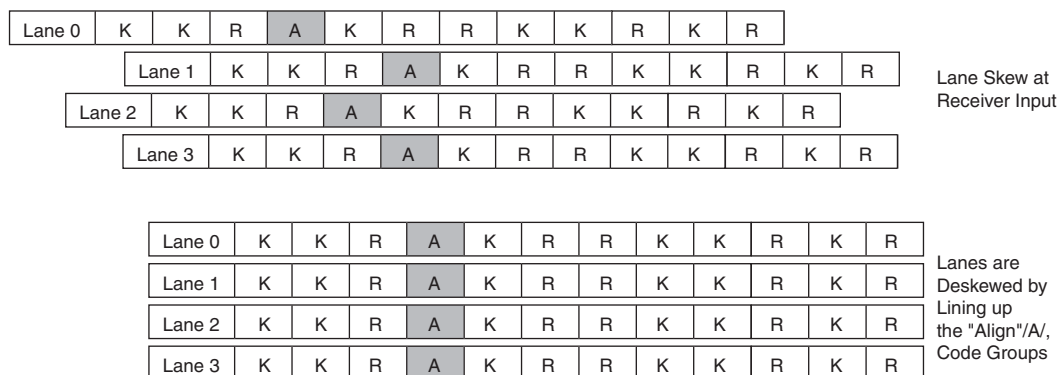


Deskew circuitry is only available in XAUI mode.

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.

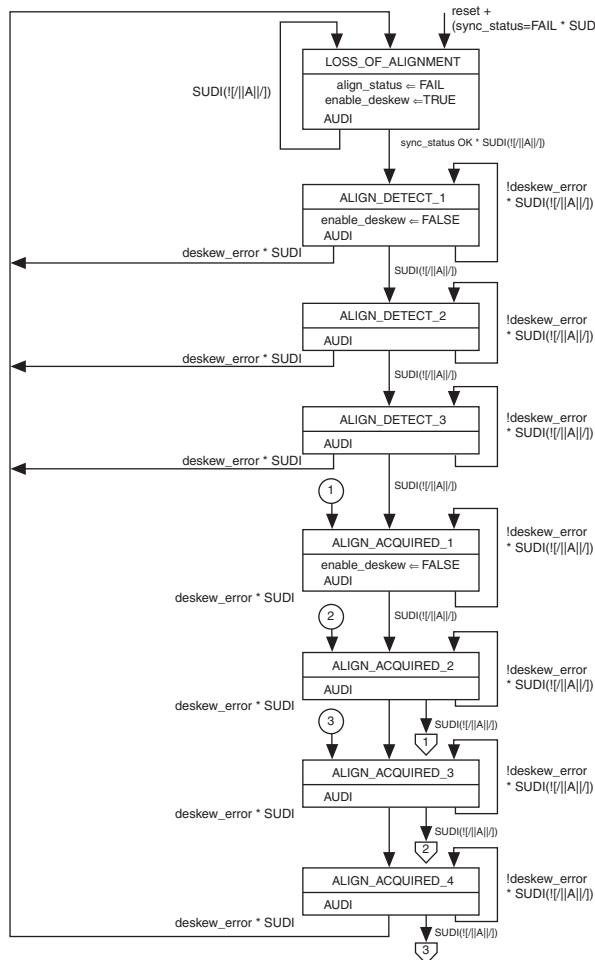
Figure 1-58. Deskew FIFO—Lane Skew at the Receiver Input



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

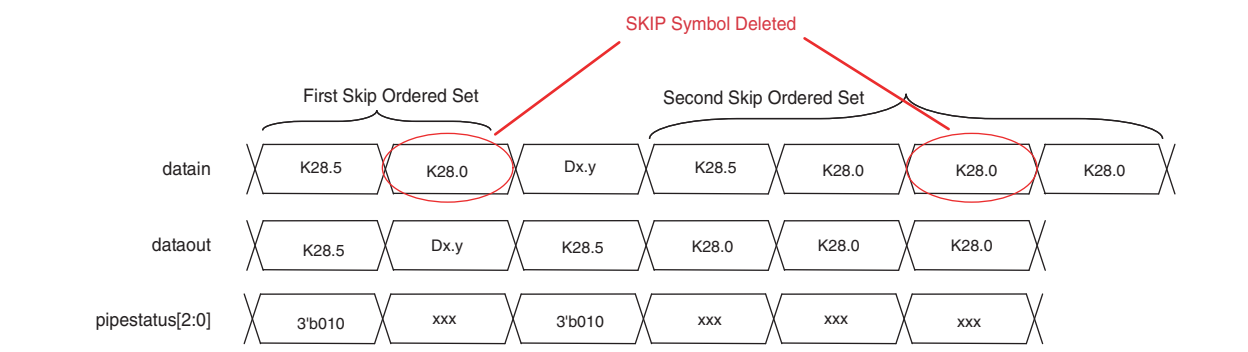
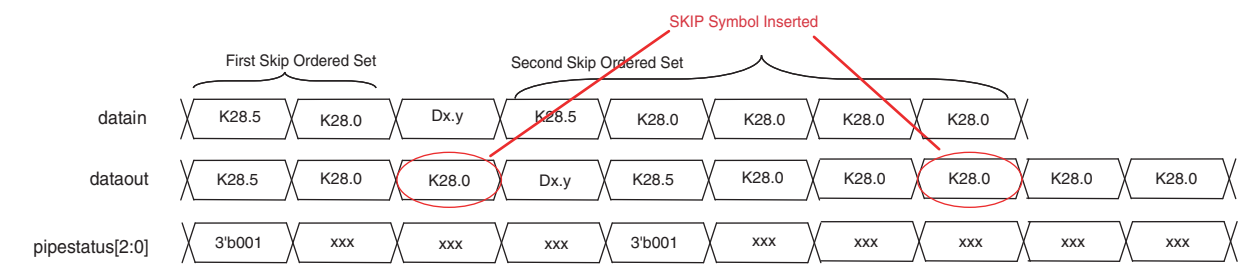


Figure 1-61 shows an example of rate match insertion in the case where two SKP symbols are required to be inserted. Only one `/K28.0/ SKP` symbol is inserted per SKP ordered set received.

Figure 1-61. Rate Match Insertion in PCIe Mode

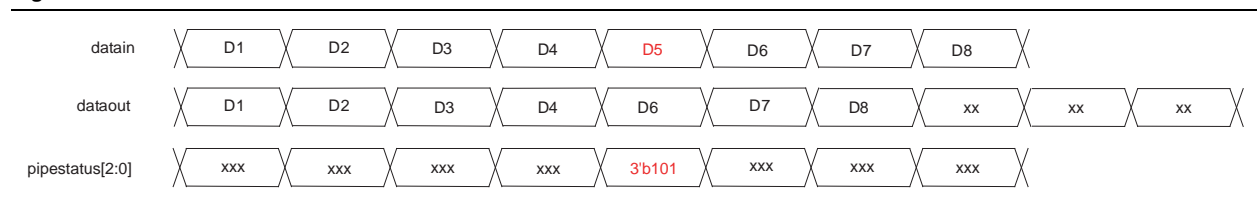


The rate match FIFO full and empty conditions are communicated to the FPGA fabric on the `pipestatus[2:0]` port from each channel.

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.

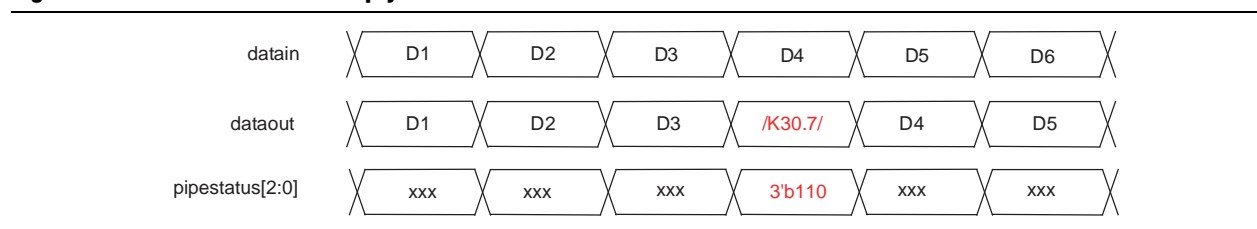
Figure 1-62. Rate Match FIFO Full Condition in PCIe Mode



The rate match FIFO automatically inserts `/K30.7/` (`9'h1FE`) after the data byte that causes the FIFO to go empty and drives `PCIstatus[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_rmfiointerinserted` 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

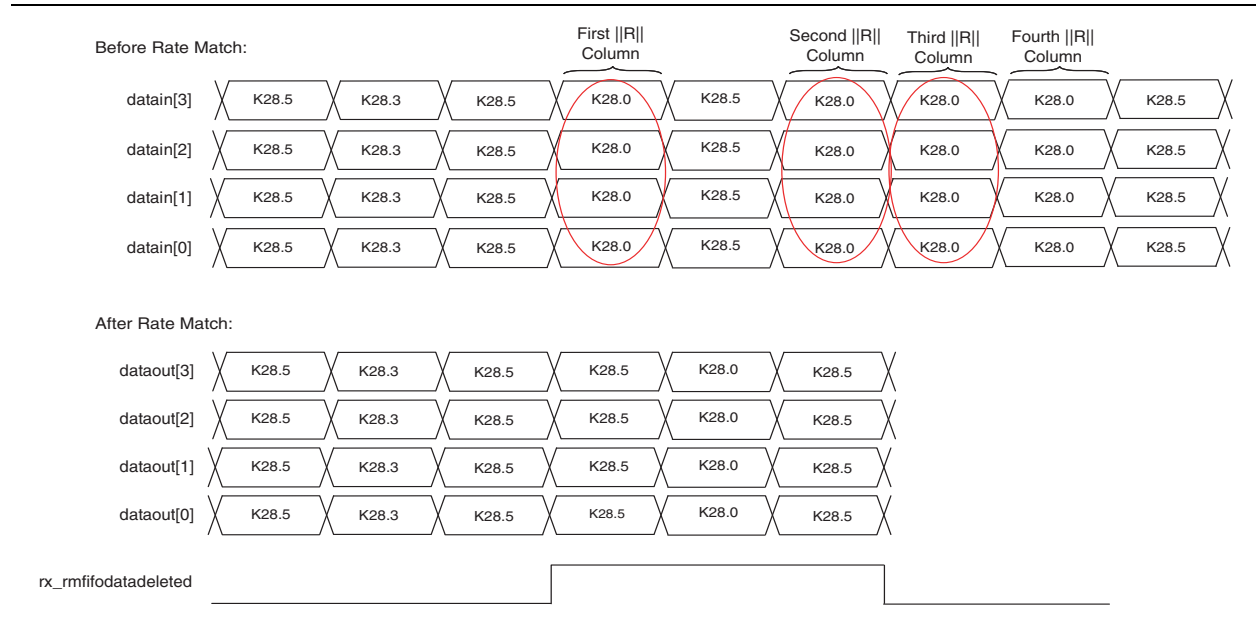
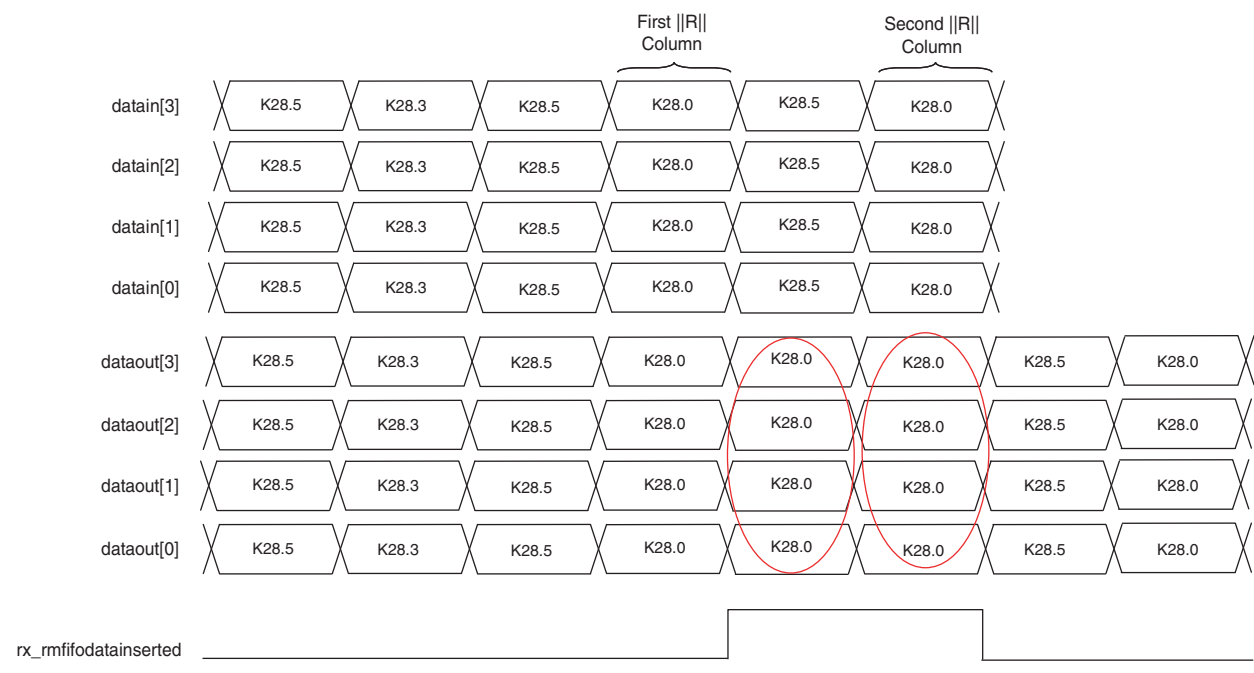


Figure 1-65 shows an example of rate match insertion in the case where two $||R||$ columns are required to be inserted.

Figure 1-65. Rate Match Insertion in XAUI Mode



Two flags, `rx_rmfifoempty` and `rx_rmfull`, 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_rmfull` and `rx_rmempty` 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/ (/K28.5/D2.2/Dx.y/Dx.y/)` ordered set 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

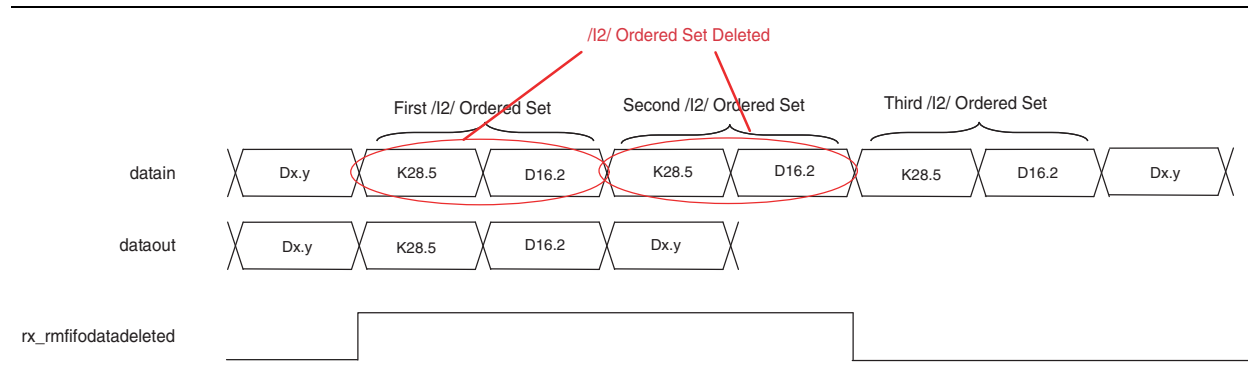
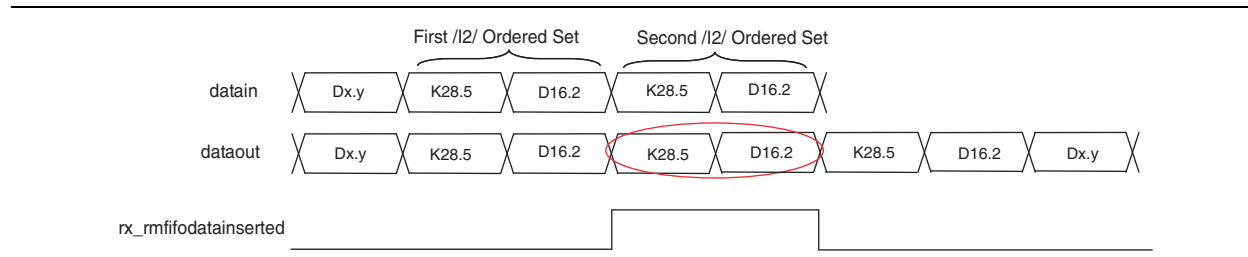



Figure 1-67 shows an example of rate match FIFO insertion in the case where one symbol is required to be inserted. Because the rate match FIFO can only insert a /I2/ ordered set, it inserts one /I2/ ordered set (two symbols inserted).

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

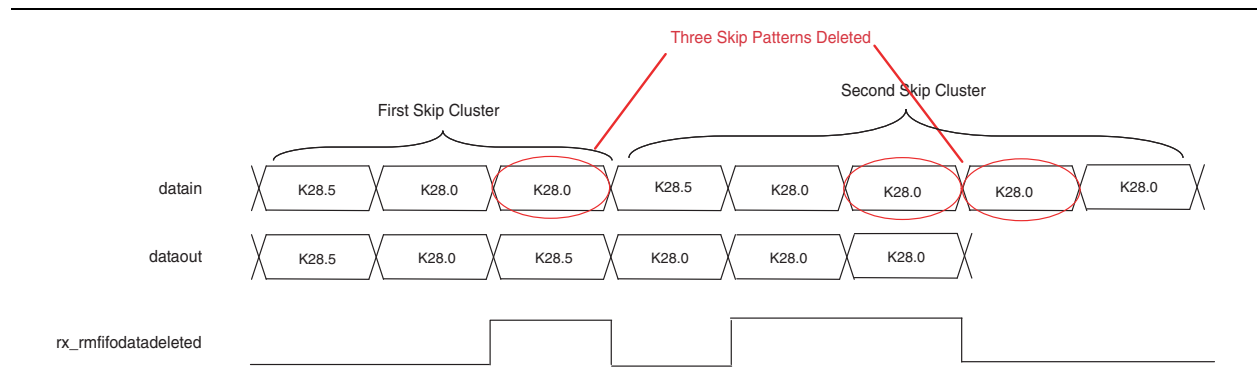
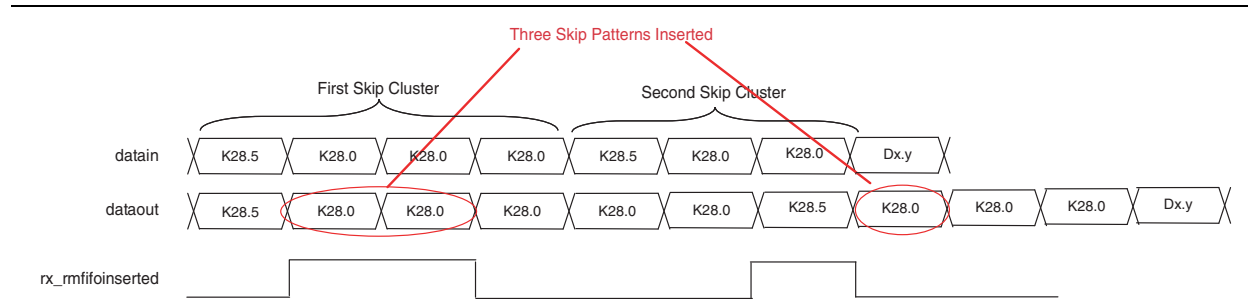


Figure 1-69 shows an example of rate match FIFO insertion in the case where three skip patterns are required to be inserted. In this example, /K28.5/ is the control pattern and neutral disparity /K28.0/ is the skip pattern. The first skip cluster has a /K28.5/ control pattern followed by three /K28.0/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

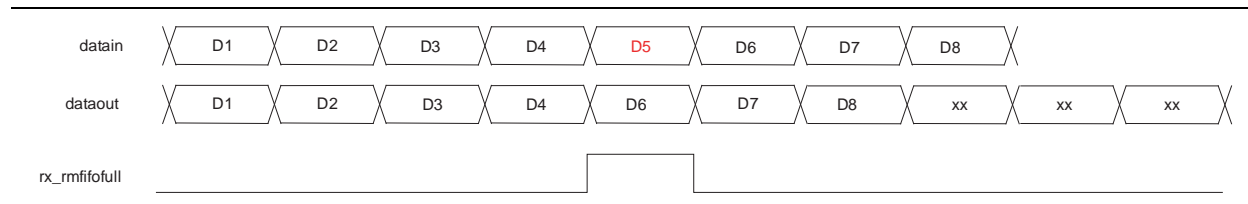


Two flags, `rx_rmffifo`full and `rx_rmffifo`empty, are forwarded to the FPGA fabric to indicate rate match FIFO full and empty conditions.

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.

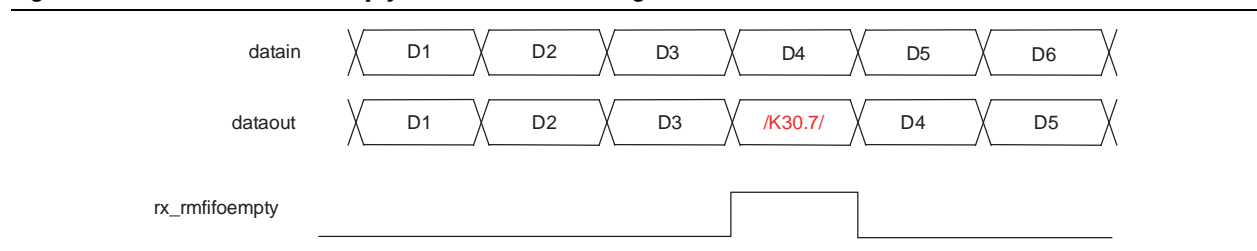
Figure 1-70. Rate Match FIFO Full Condition in Basic Single-Width Mode



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_rmfifoatadeleted` and `rx_rmfifoatinserted`, 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

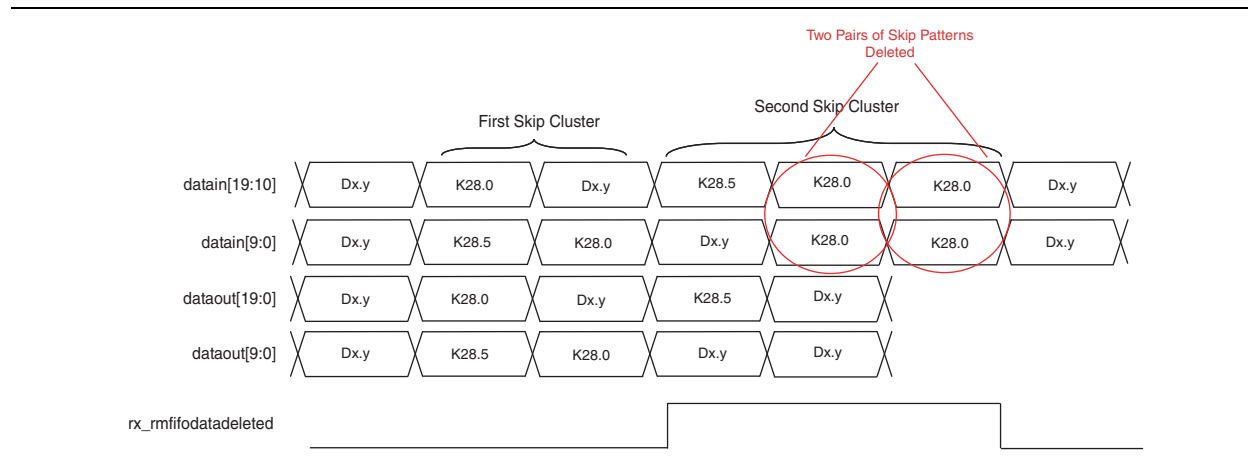
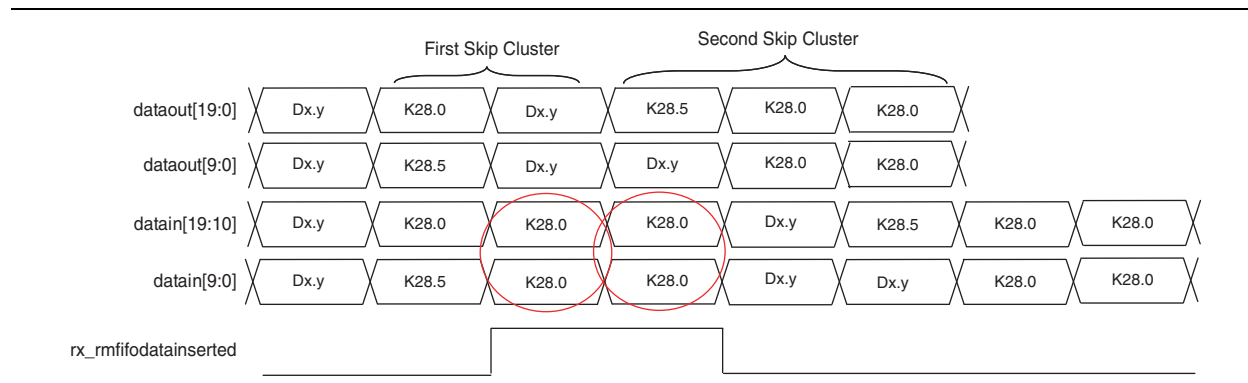


Figure 1-73 shows an example of rate match FIFO insertion in the case where three skip patterns are required to be inserted. In this example, /K28.5/ is the control pattern and neutral disparity /K28.0/ is the skip pattern. The first skip cluster has a /K28.5/ control pattern in the LSByte and /K28.0/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

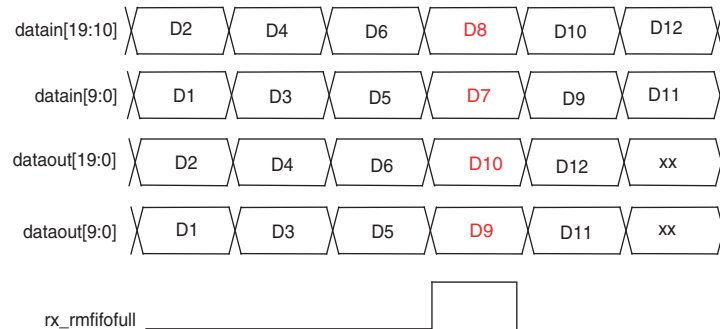


Two flags, `rx_rmfifoempty` and `rx_rmfull`, are forwarded to the FPGA fabric to indicate rate match FIFO full and empty conditions.

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_rmfull` 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.

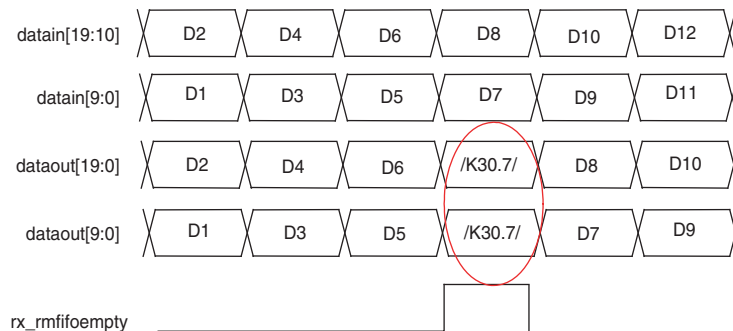
Figure 1-74. Rate Match FIFO Full Condition in Basic Double-Width Mode



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_fifoempty` 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

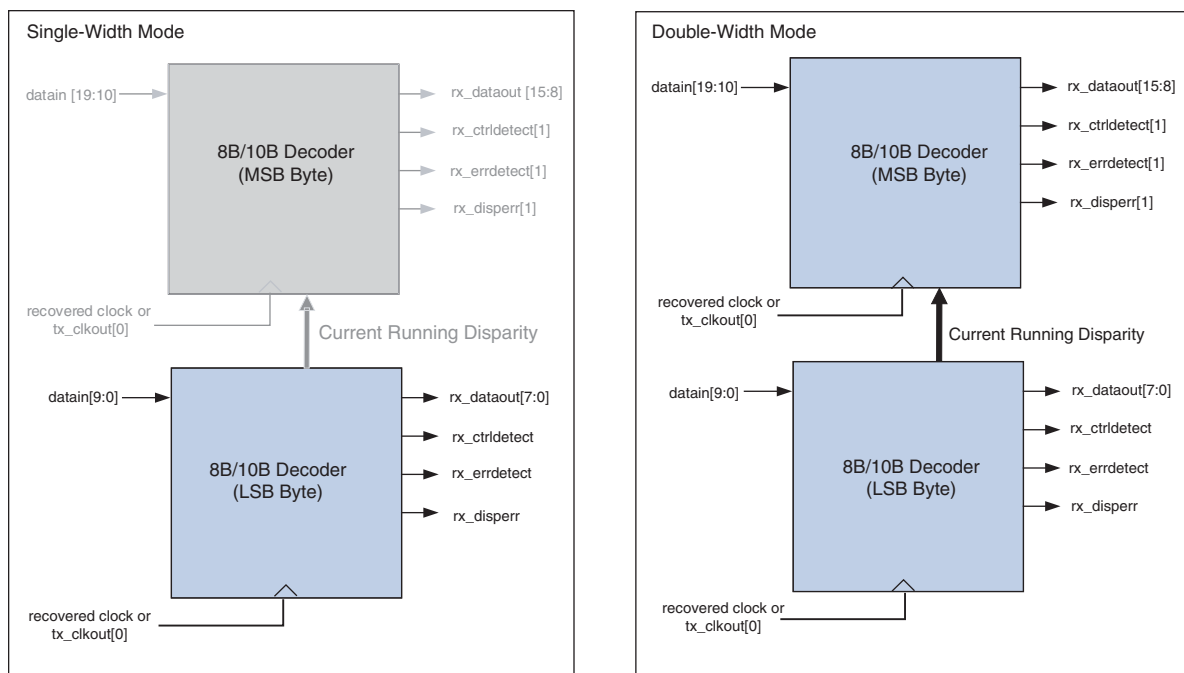
Protocols such as PCIe, XAUI, GIGE, and Serial RapidIO require the serial data sent over the link to be 8B/10B encoded to maintain the DC balance in the serial data transmitted. These protocols require the receiver PCS logic to implement an 8B/10B decoder to decode the data before forwarding it to the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

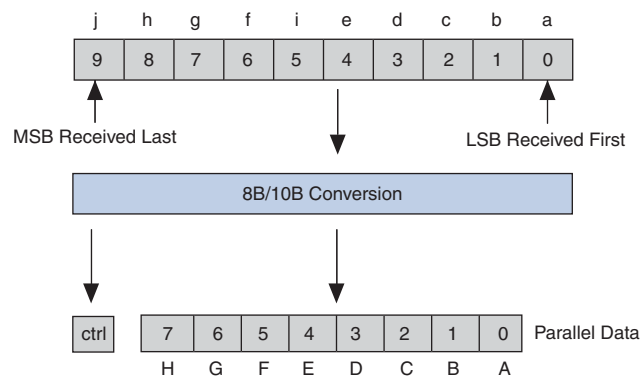
The 8B/10B decoder operates in single-width mode in the following functional modes:

- 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

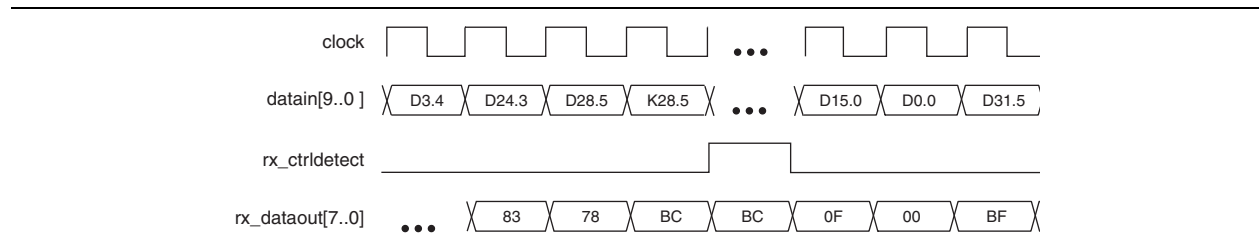


Control Code Group Detection

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

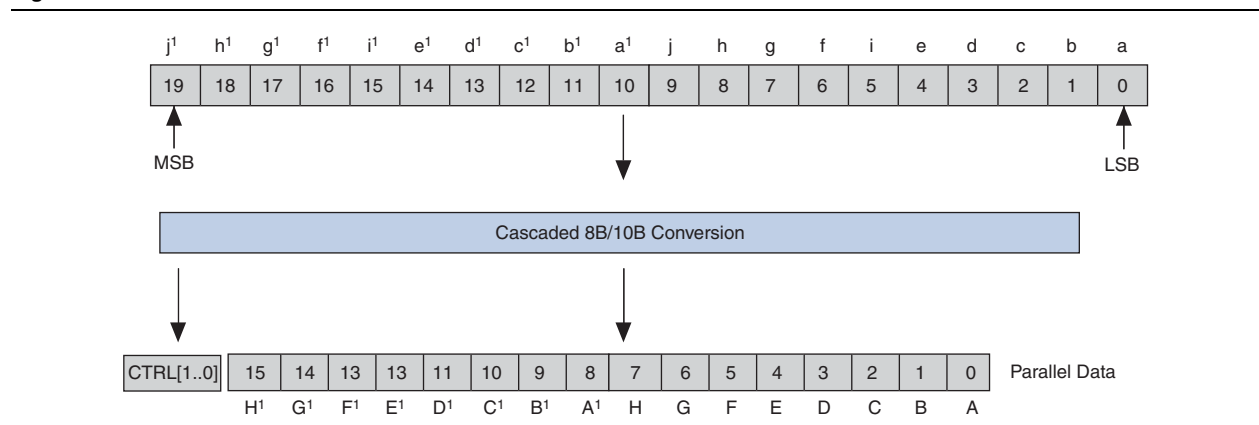
The left side of Figure 1-76 on page 1-89 shows the 8B/10B decoder in double-width mode. In this mode, two 8B/10B decoders are cascaded for decoding the 20-bit encoded data, as shown in Figure 1-79. The 10-bit LSB of the received 20-bit encoded data is decoded first and the ending running disparity is 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

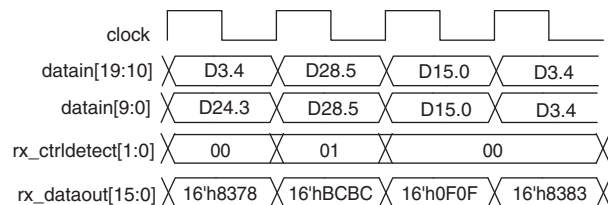


Control Code Group Detection

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:

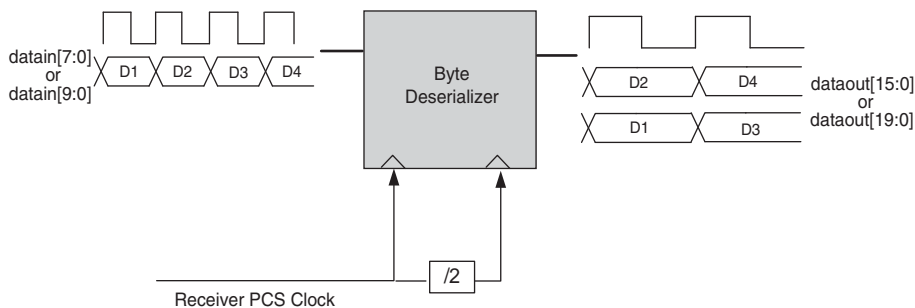
- Single-width mode
- Double-width mode

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

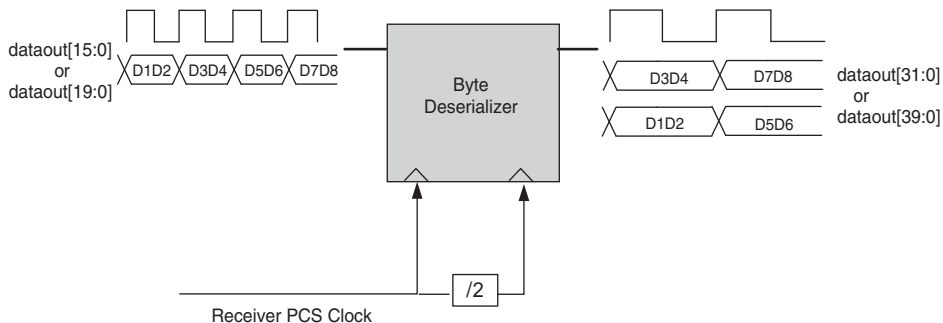


Byte Deserializer in Double-Width Mode

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.

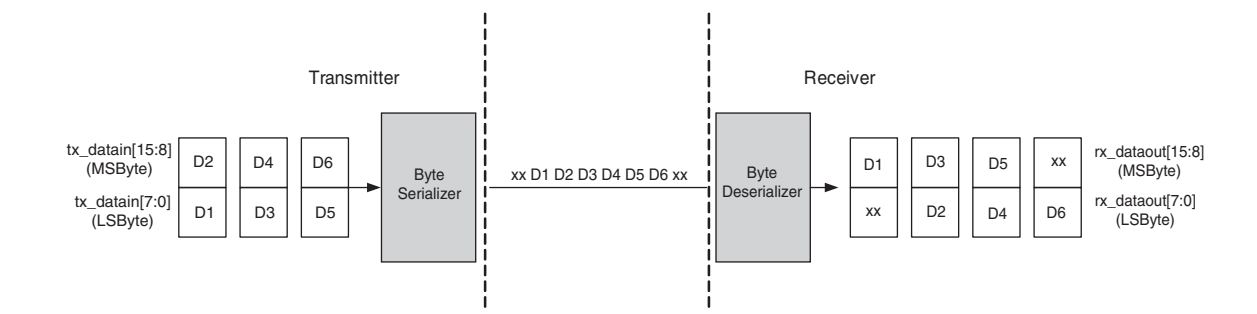
Figure 1-82. Byte Deserializer in Double-Width Mode



Byte Ordering Block

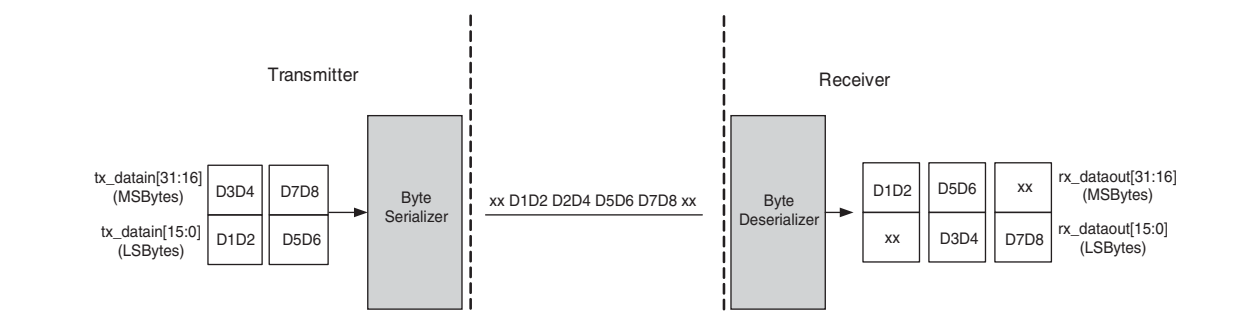
In single-width modes with the 16-bit or 20-bit FPGA fabric-transceiver interface, the byte deserializer receives one data byte (8 or 10 bits) and deserializes it into two data bytes (16 or 20 bits). Depending on when the receiver PCS logic comes out of reset, the byte ordering at the output of the byte 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.


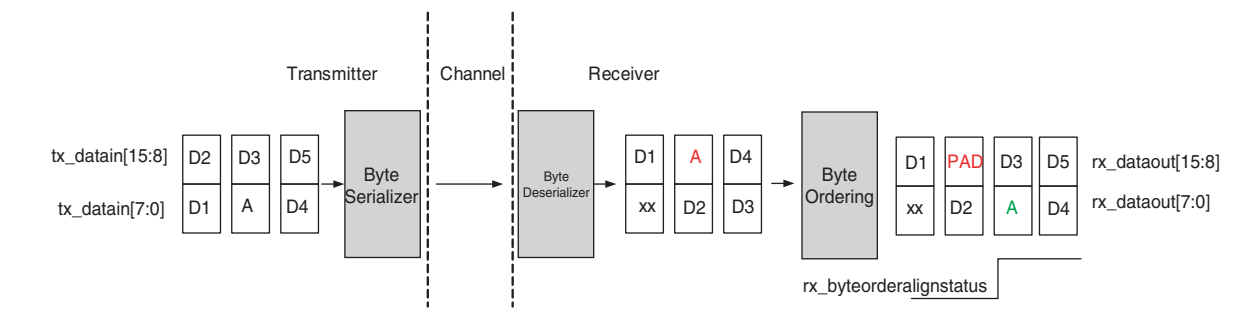
 You can choose word-alignment-based byte ordering in the **Rate match/Byte order** tab of the ALTGX MegaWizard Plug-In Manager. For the **What do you want the byte ordering to be based on?** question, select the **The sync status signal from the word aligner** option.

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

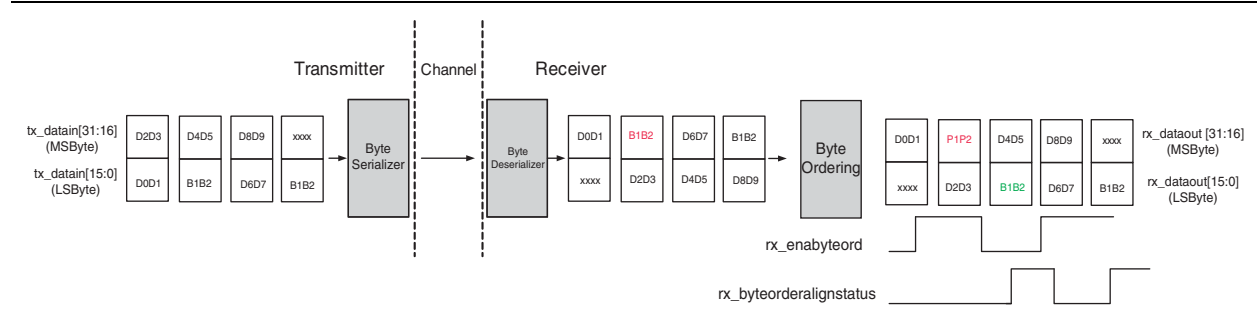


User-Controlled Byte Ordering

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

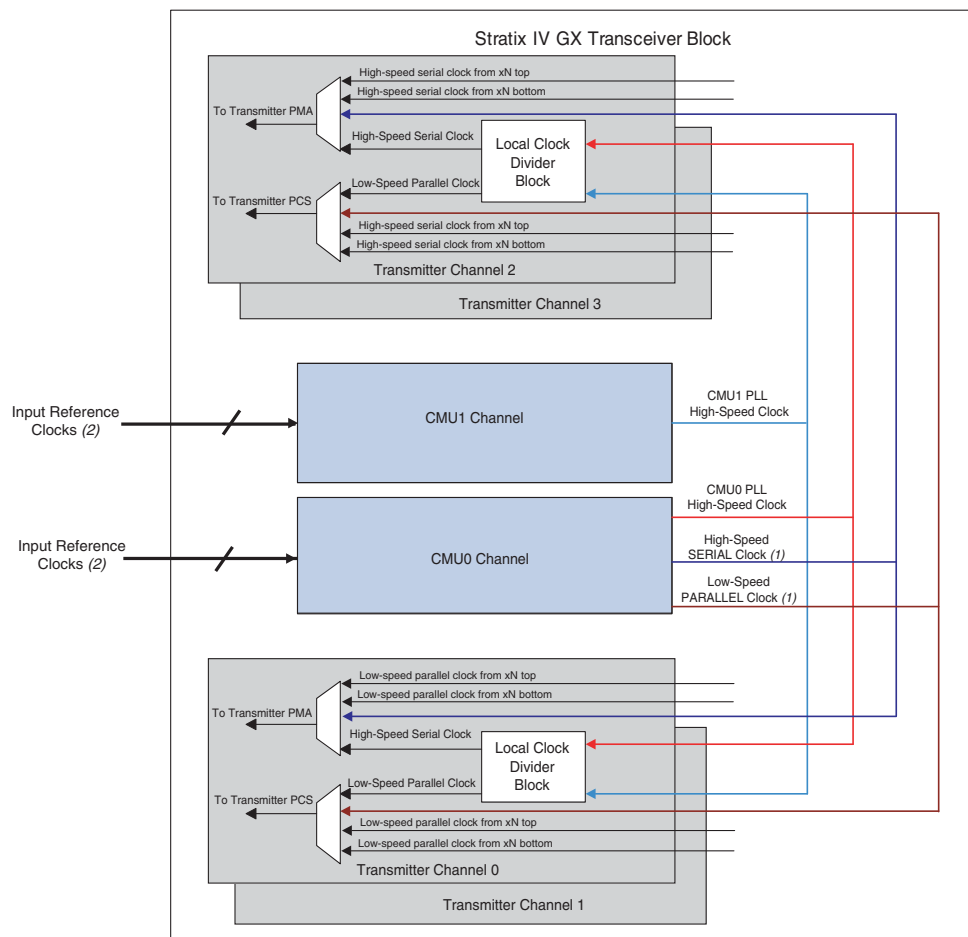
An optional `rx_phase_comp_fifo_error` port is available in all functional modes to indicate a receiver phase compensation FIFO under-run or overflow condition. The `rx_phase_comp_fifo_error` signal is asserted high when the phase compensation FIFO gets either full or empty. This feature is useful to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

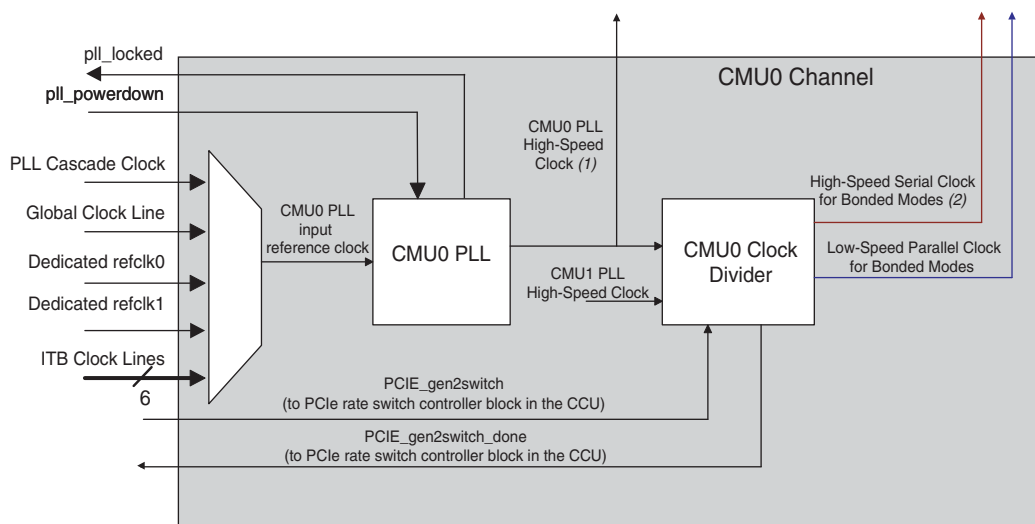
 For Stratix IV GT devices, you can use the CMU PLL to generate transceiver clocks at data rates between 600 Mbps and 11.3 Gbps.

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



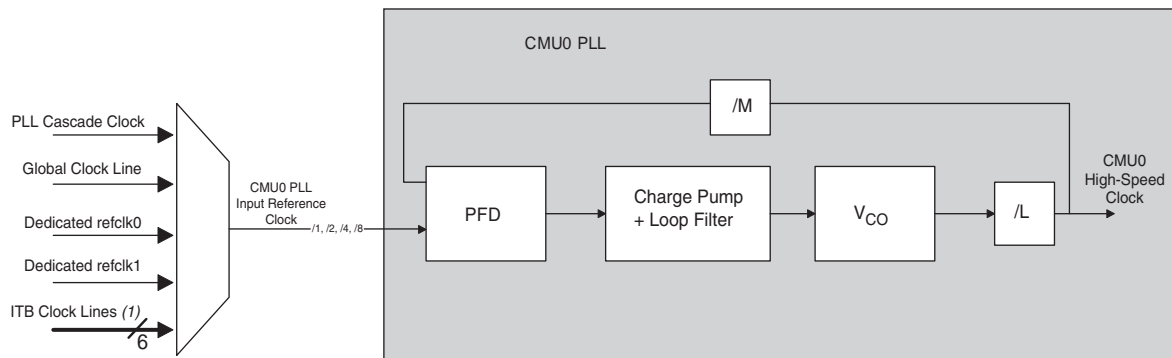
Notes to Figure 1-88:

- (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.

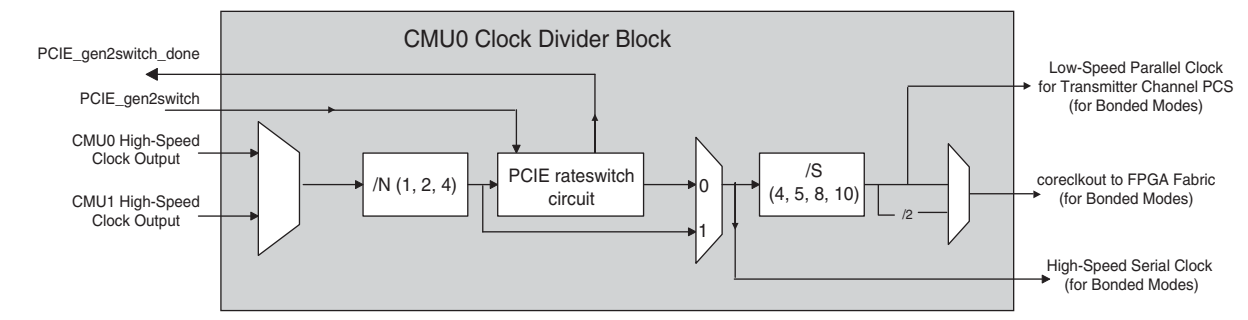
CMU0 Clock Divider

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.

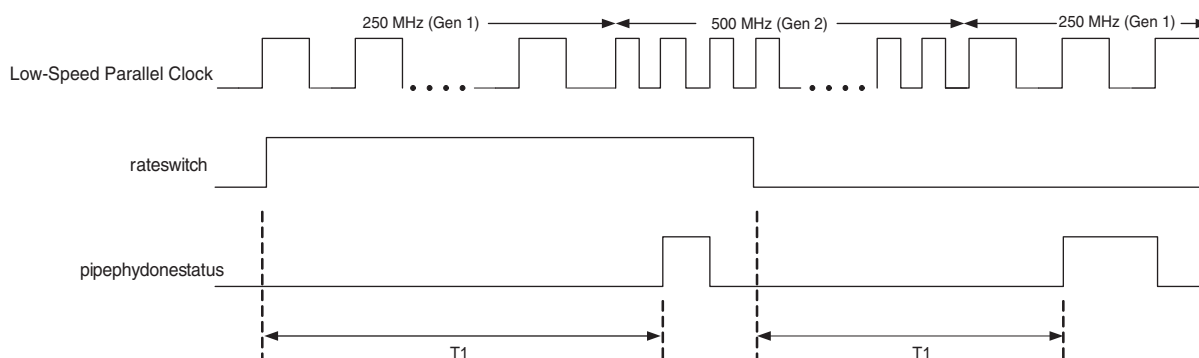
- When you set the `rateswitch` port to 0, the PCIe rateswitch controller (in the CCU) signals the PCIe rateswitch circuit to select the divide by /2 to provide a high-speed serial clock for the Gen1 (2.5 Gbps) data rate.
- When you set the `rateswitch` port to 1, the /N divider output is forwarded, 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

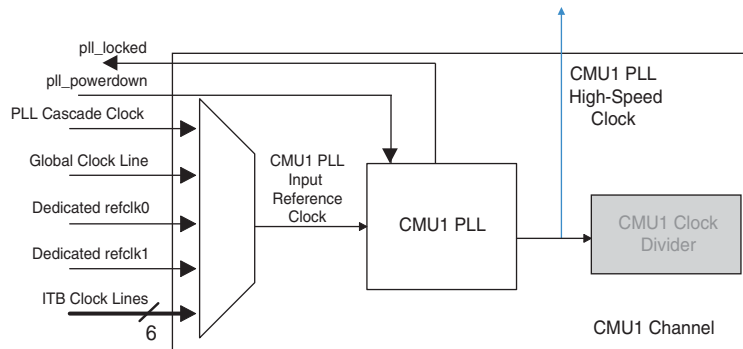
The /S divider receives the clock output from the /N divider or PCIE rateswitch circuit (only in PCIe mode) and generates the low-speed parallel clock for the PCS block of all transmitter channels and coreclkout for the FPGA fabric. If the byte serializer block is enabled in bonded channel modes, the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

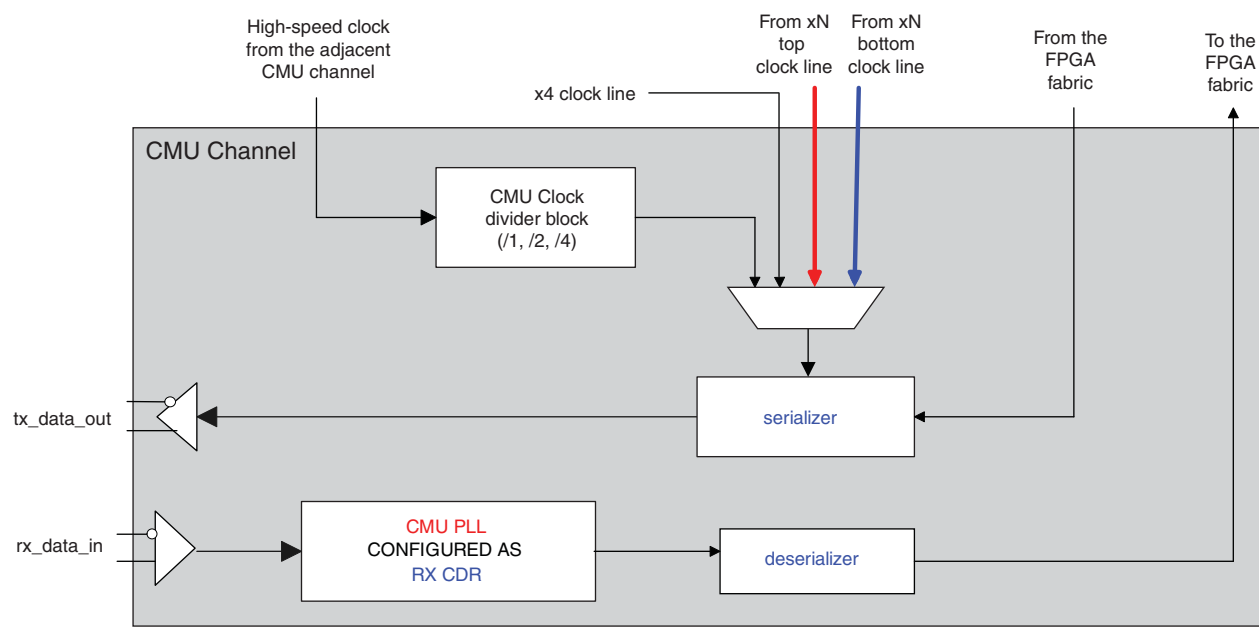
- For more information, refer to the [Reset Control and Power Down in Stratix IV Devices](#) chapter.


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.

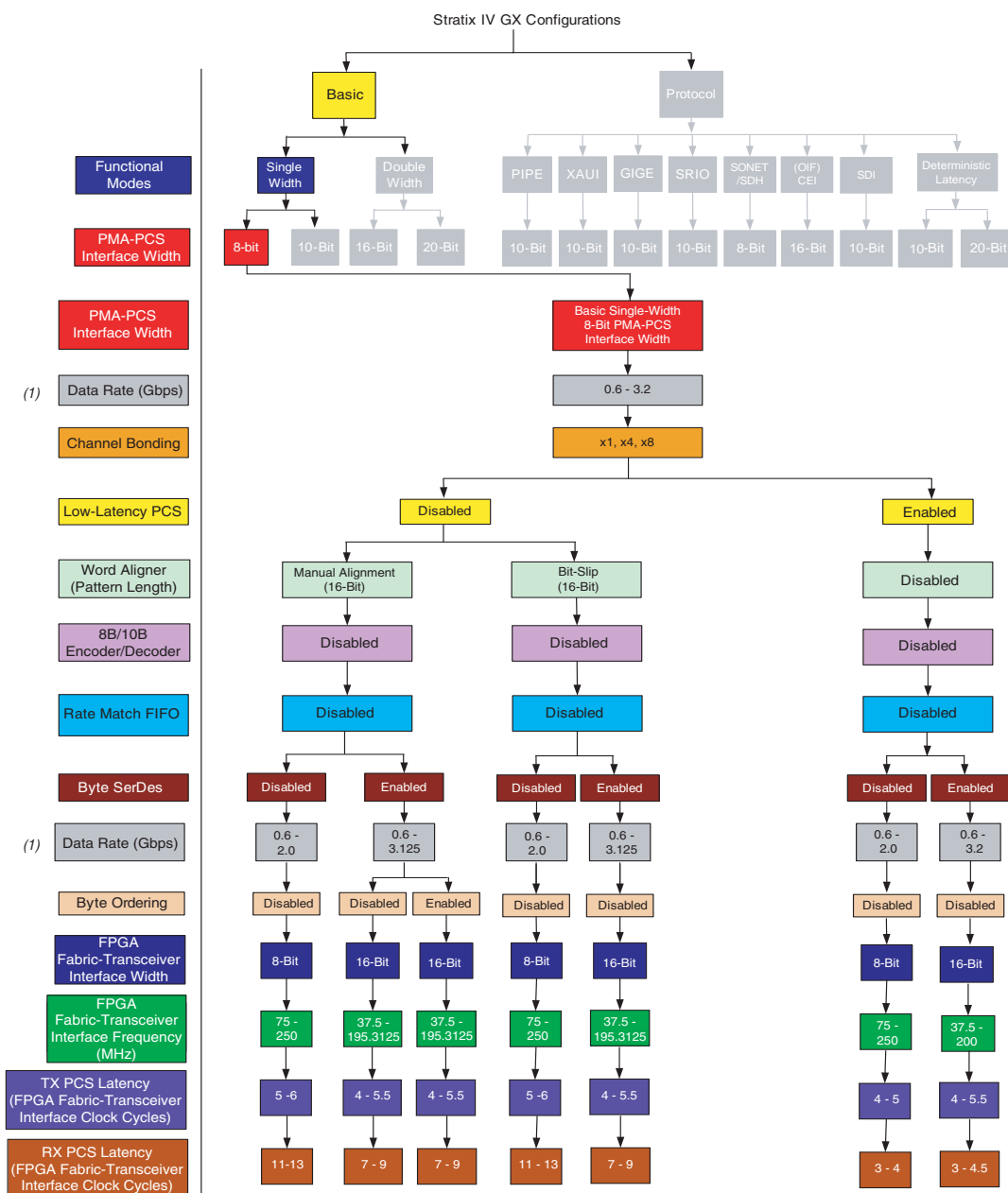
-  The PCS latency in Basic single-width and Basic double-width modes with and without the low latency PCS mode option is pending characterization.
-  Basic double-width mode configurations at data rates of > 6.5 Gbps are only allowed in low-latency PCS bypass mode.

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.

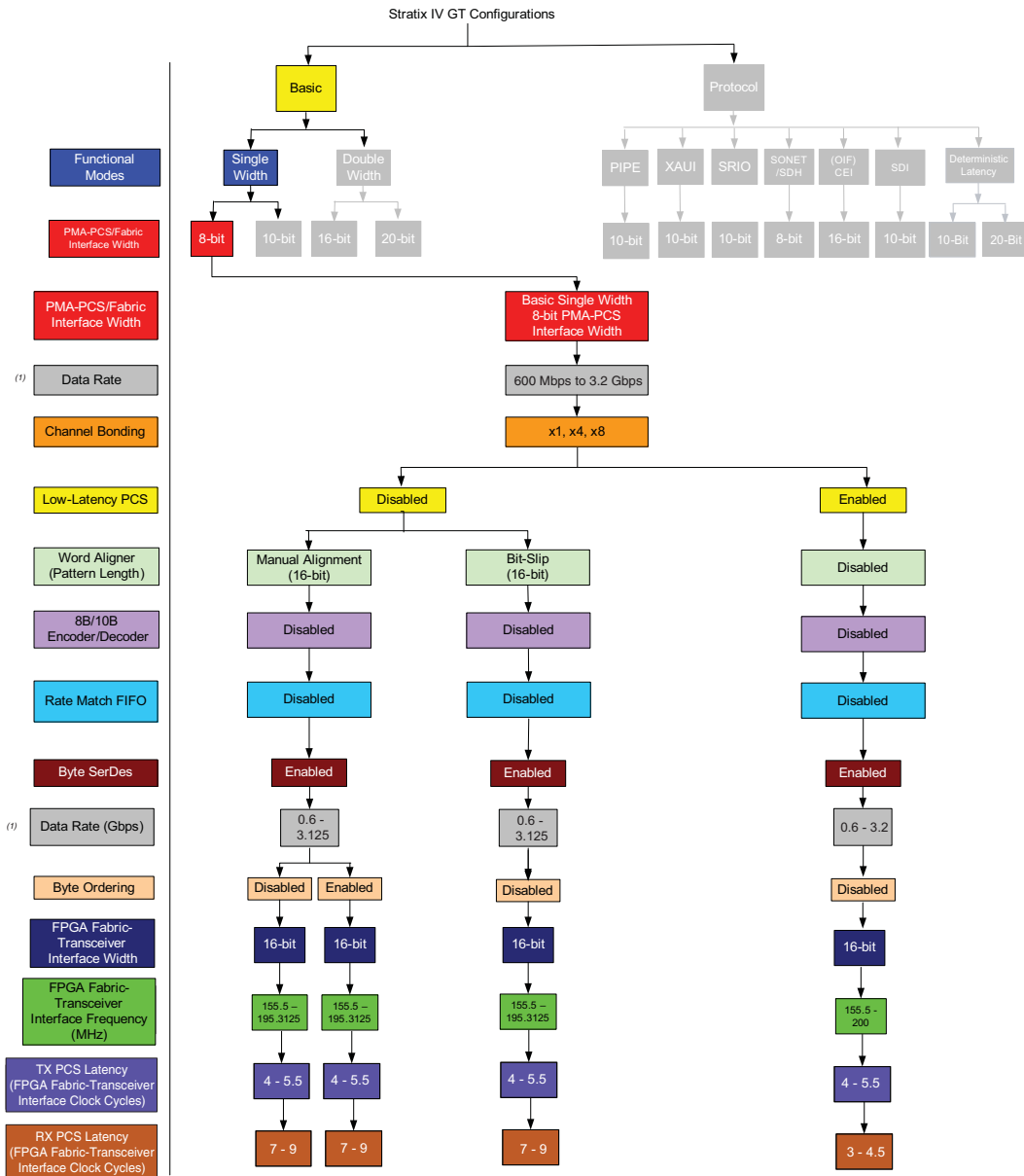
Figure 1-94. Transceiver Configurations in Basic Single-Width Mode with an 8-Bit PMA-PCS Interface for Stratix IV GX Devices



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



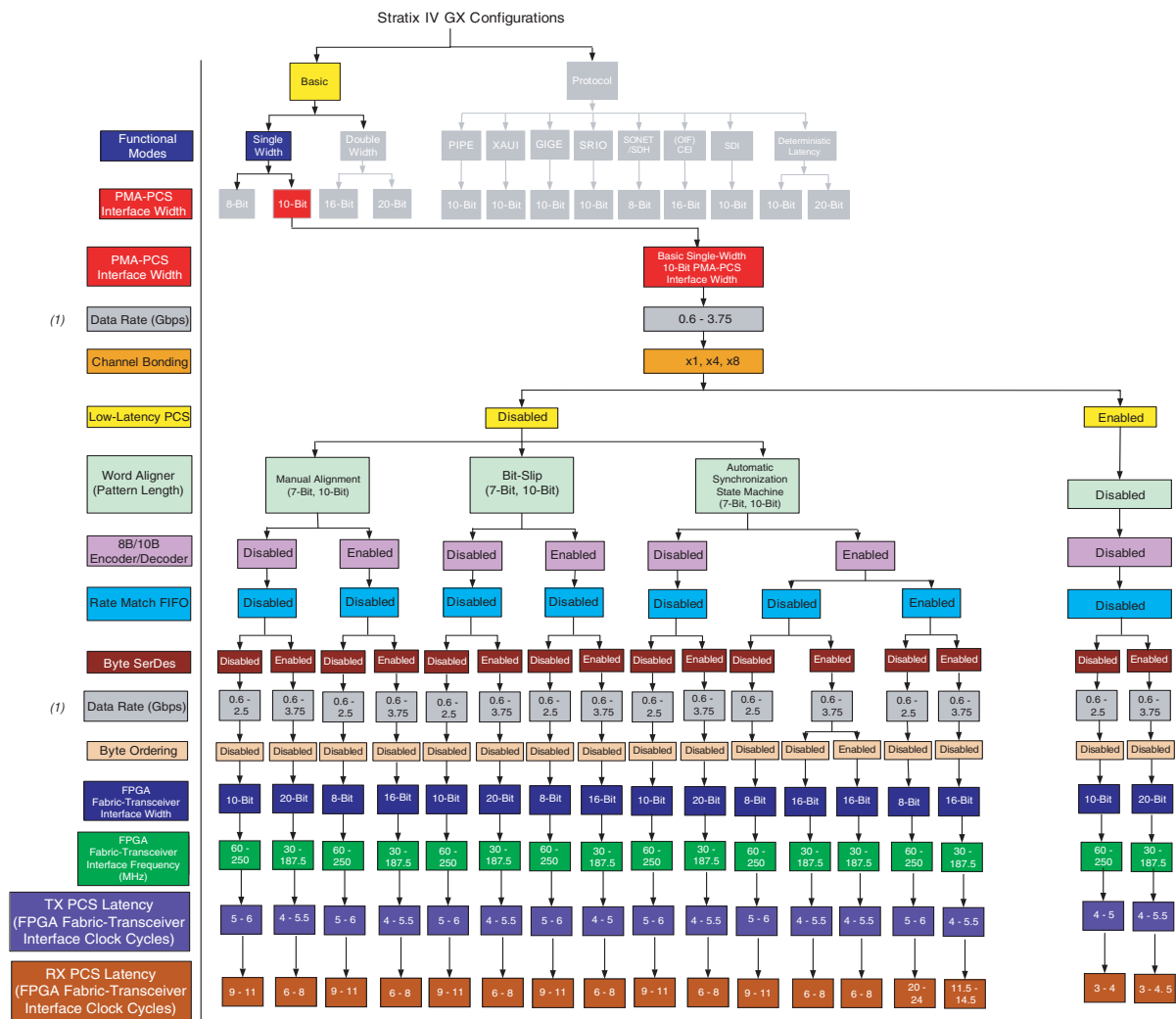
Note to Figure 1-95:

- (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.

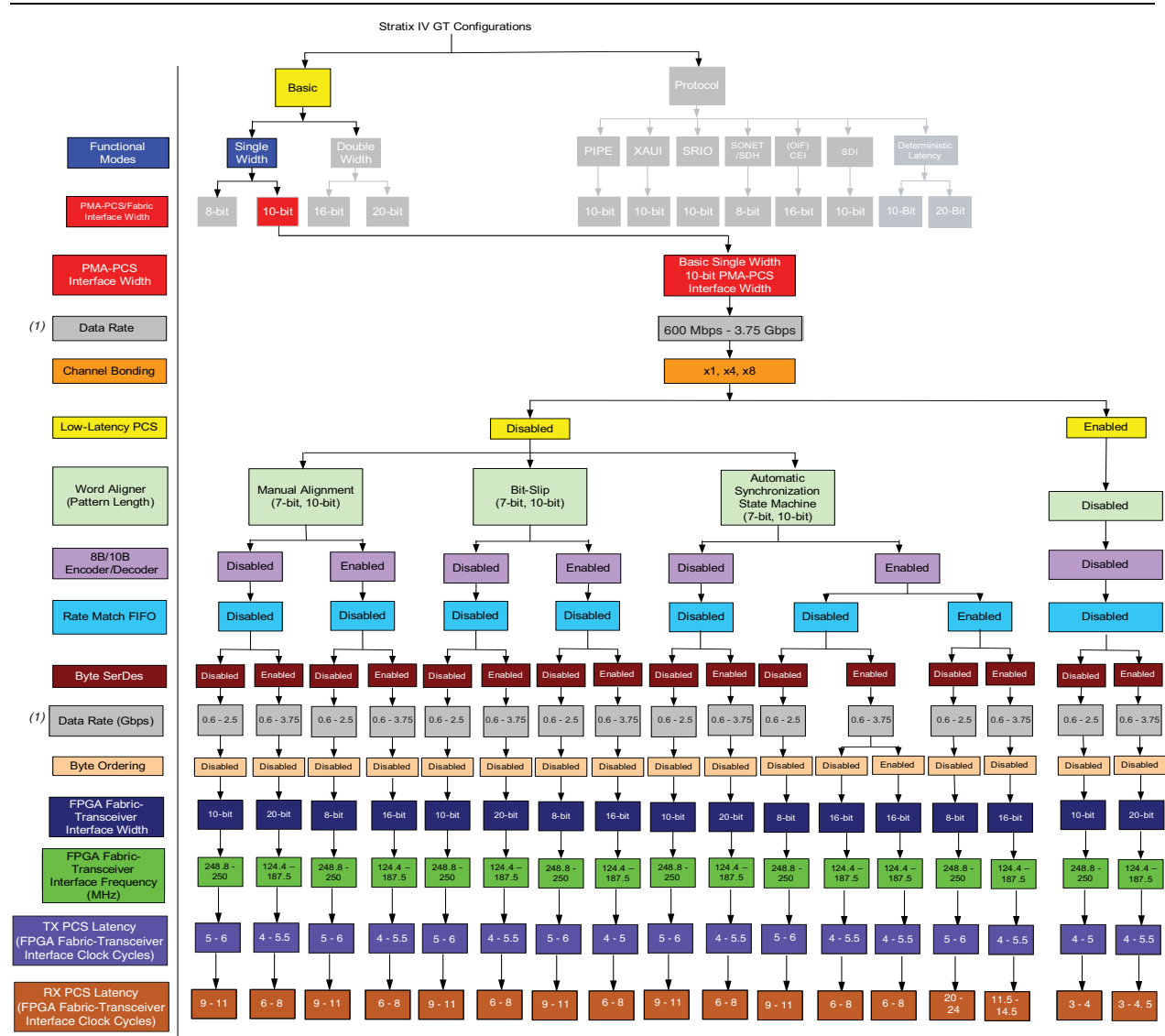
Figure 1-96. Transceiver Configurations in Basic Single-Width Mode with a 10-Bit PMA-PCS Interface for Stratix IV GX Devices



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:

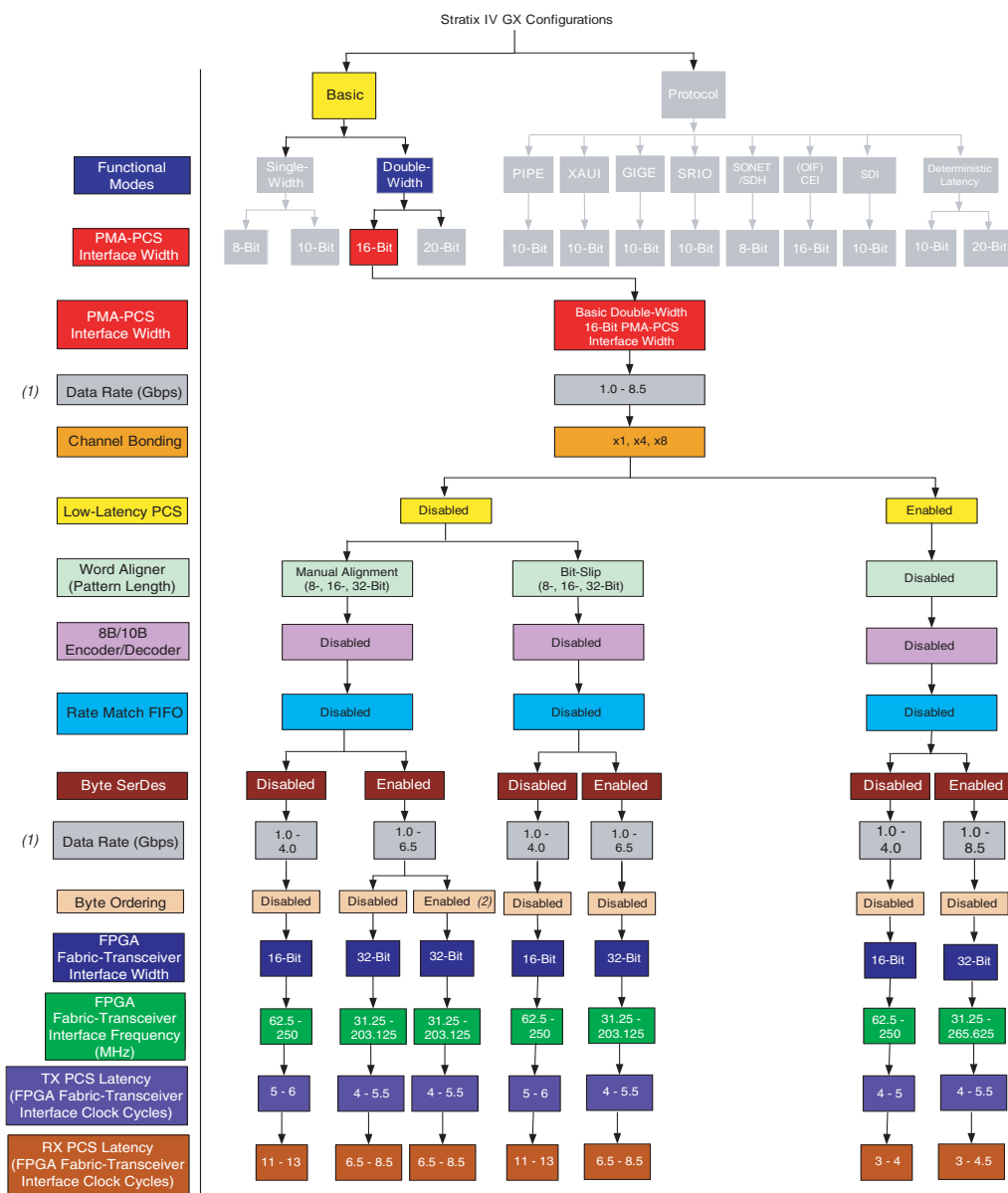
(1) The maximum data rate specification shown in Figure 1-97 is valid only for the -2 (fastest) speed grade devices. For data rate specifications for other speed grades offered, refer to the *DC and Switching Characteristics for Stratix IV Devices* chapter.

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.

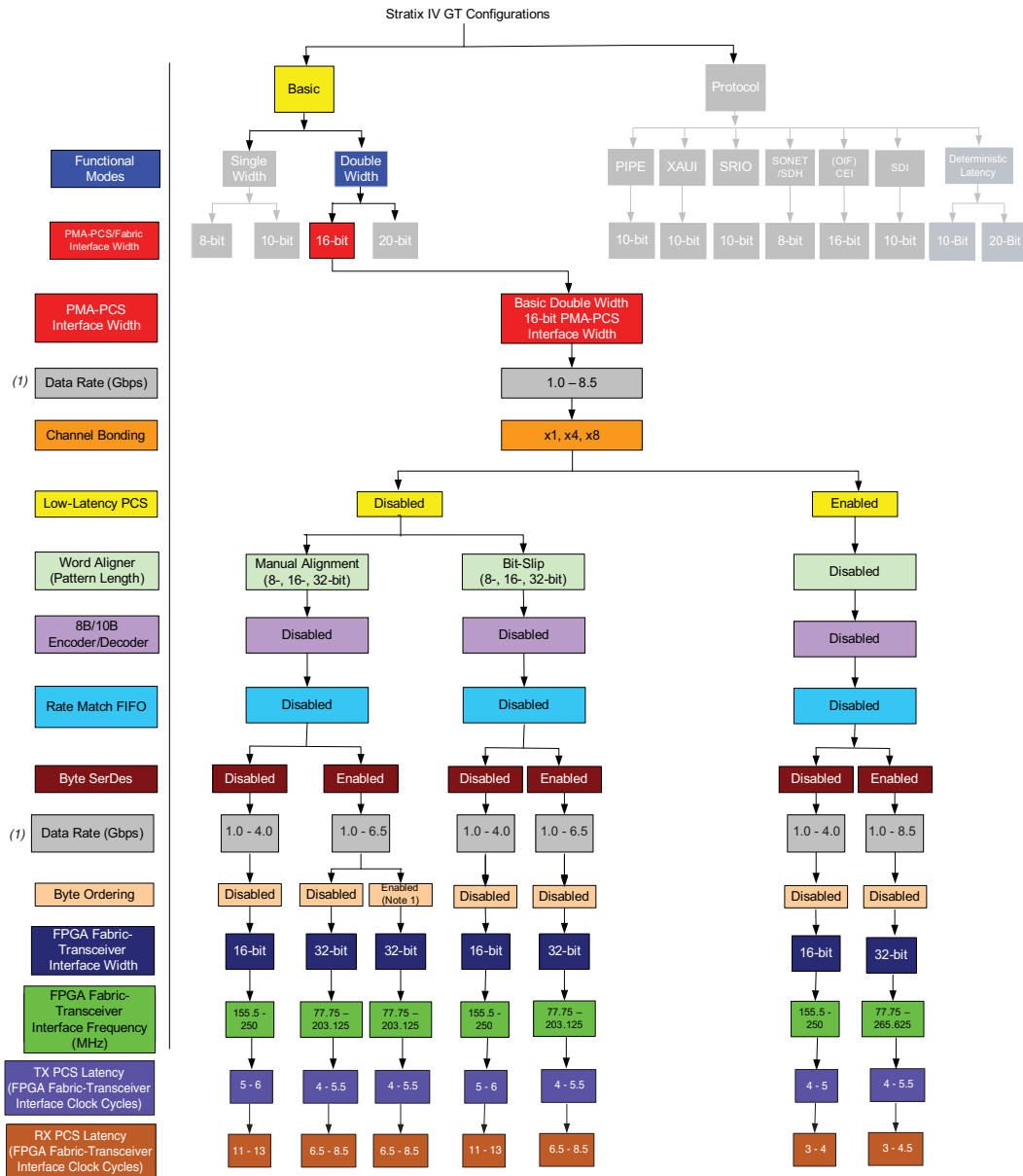
Figure 1-98. Transceiver Configurations in Basic Double-Width Mode with a 16-Bit PMA-PCS Interface for Stratix IV GX Devices



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



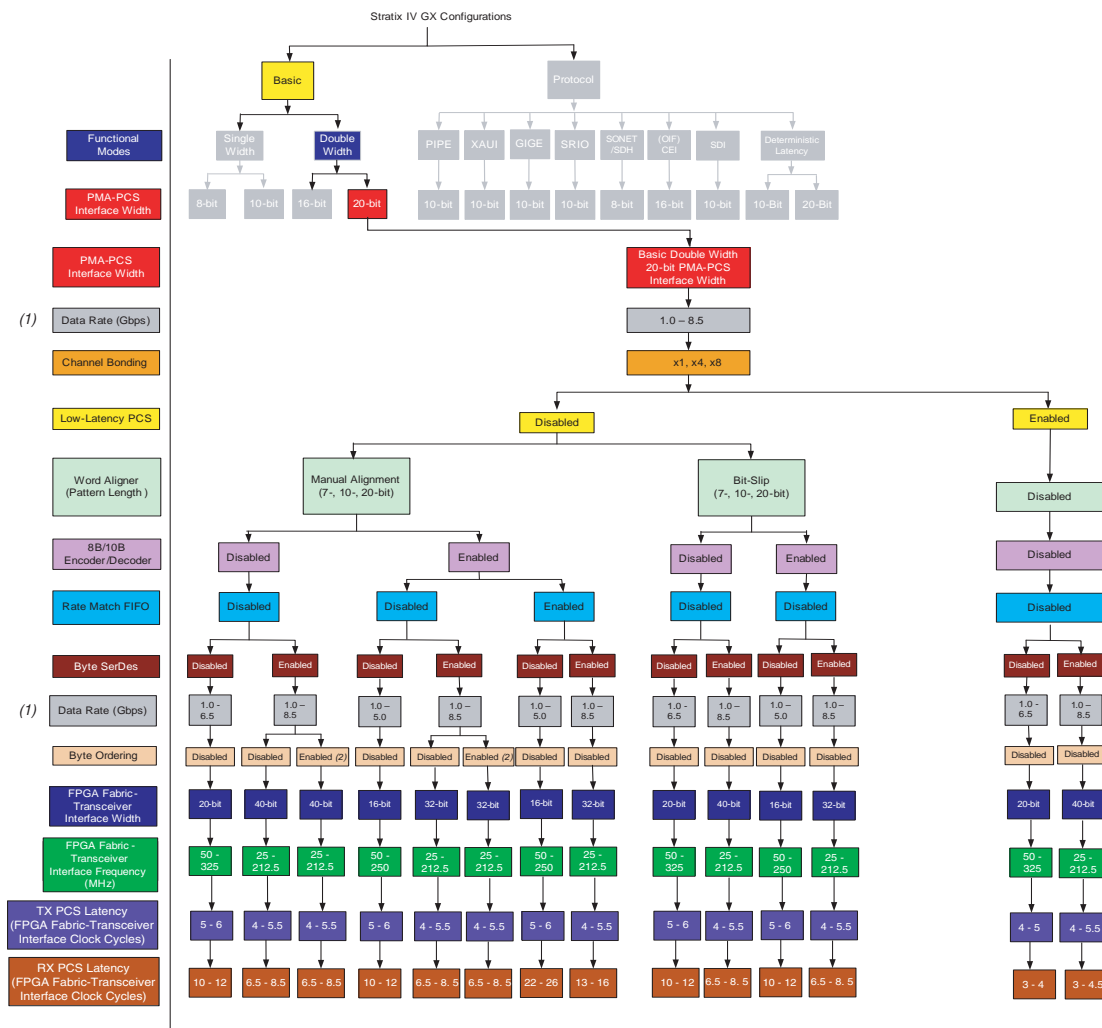
Note to Figure 1-99:

(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.

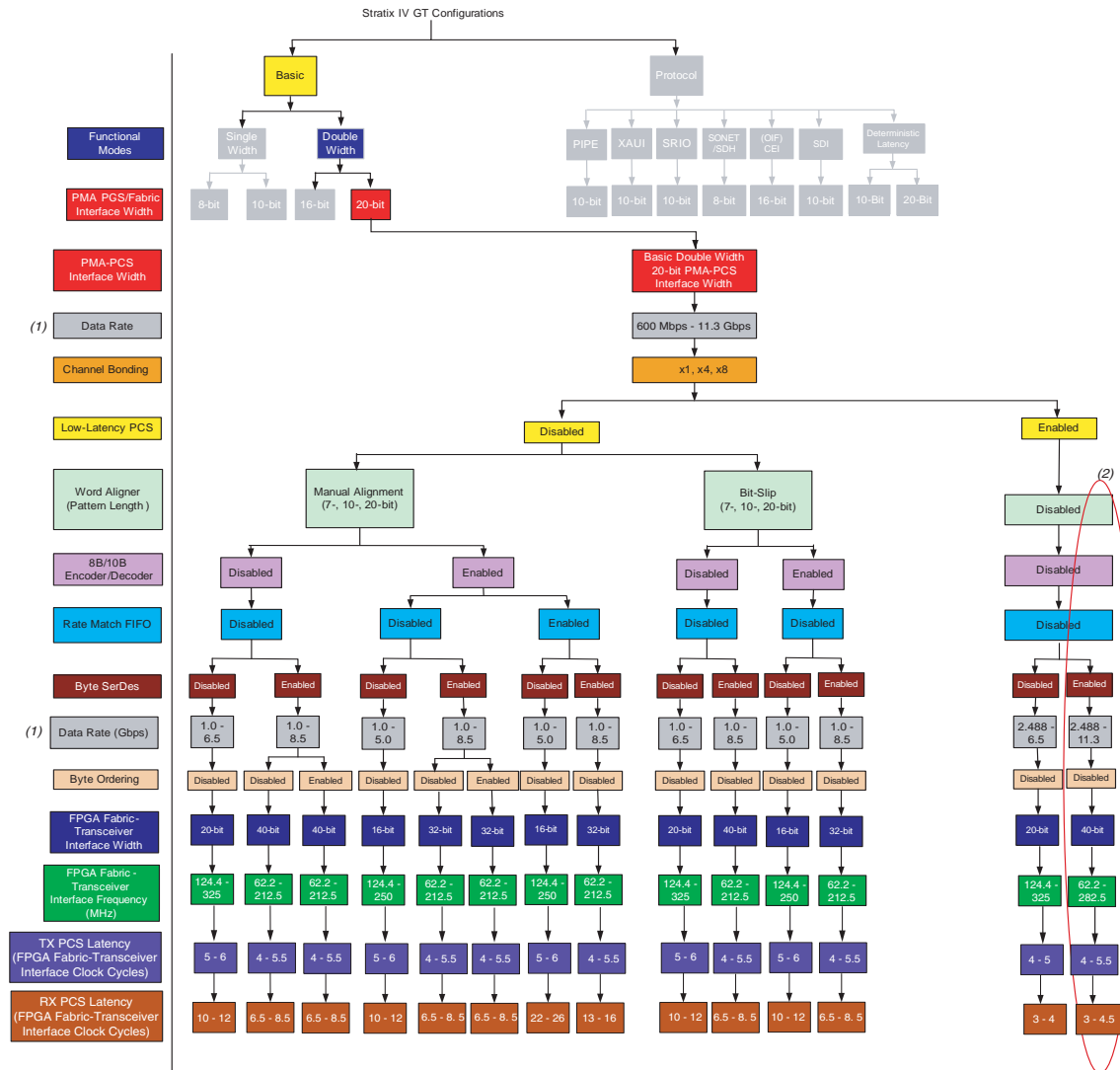
Figure 1-100. Transceiver Configurations in Basic Double-Width Mode with a 20-Bit PMA-PCS Interface for Stratix IV GX Devices



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.

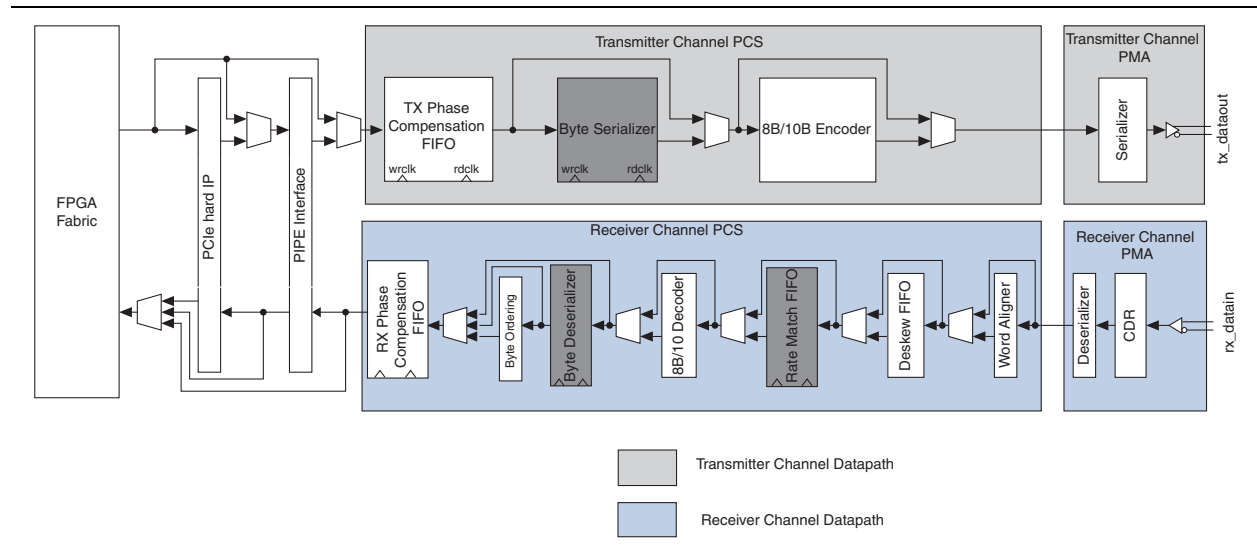
 For information about other protocols supported using Basic functional mode, refer to *AN 577: Recommended Protocol Configurations for Stratix IV FPGAs*.

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.

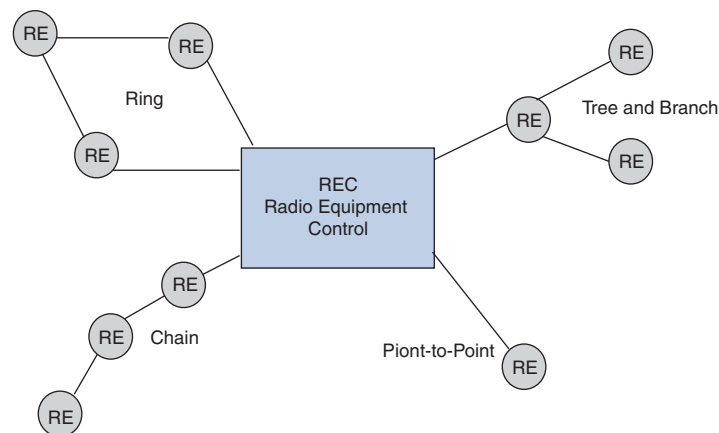
In the deterministic latency $\times 4$ option, up to four CPRI TX channels can be bundled in an $\times 4$ group so that they all have the same TX uncertainty and just require one TX PLL to compensate for it. This is allowed in cases where the data rates are multiples of a single PLL output frequency; 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

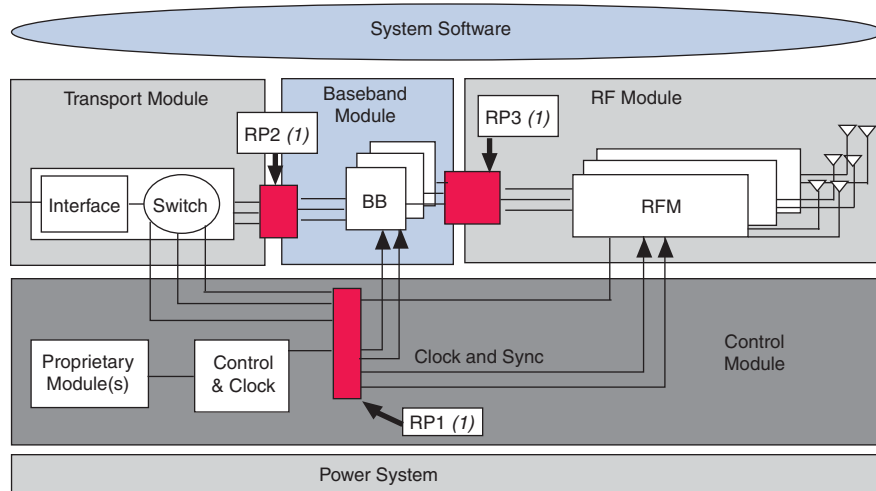


If the destination for high-speed serial data leaving the REC is the first RE, it is a single-hop connection. If serial data from the REC has to traverse through multiple REs before reaching the destination RE, it is a multi-hop connection. Remotely locating the RF transceiver from the main base station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

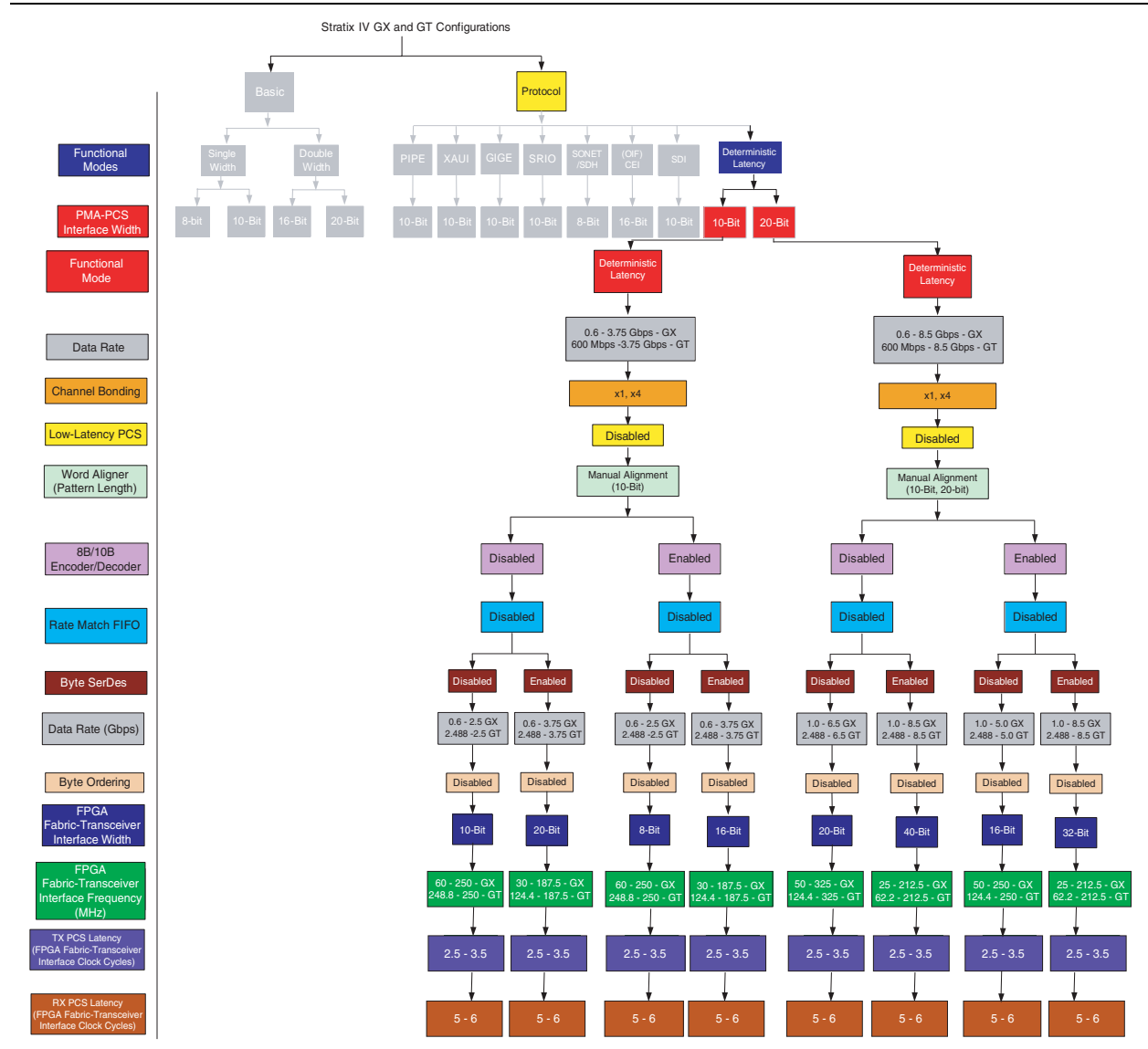


Note to Figure 1-104:

(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*.

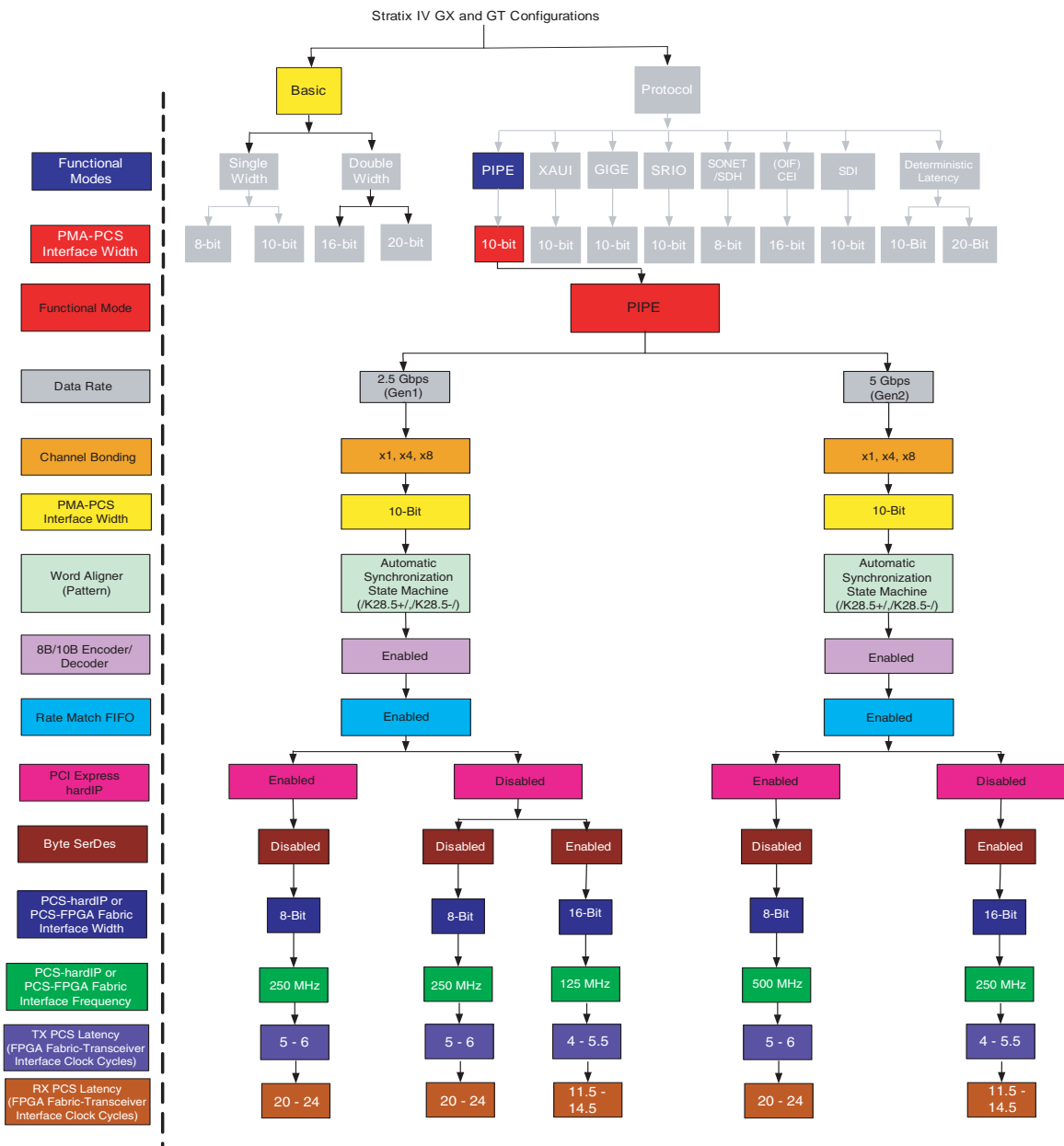
PCIe Mode Configurations

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.

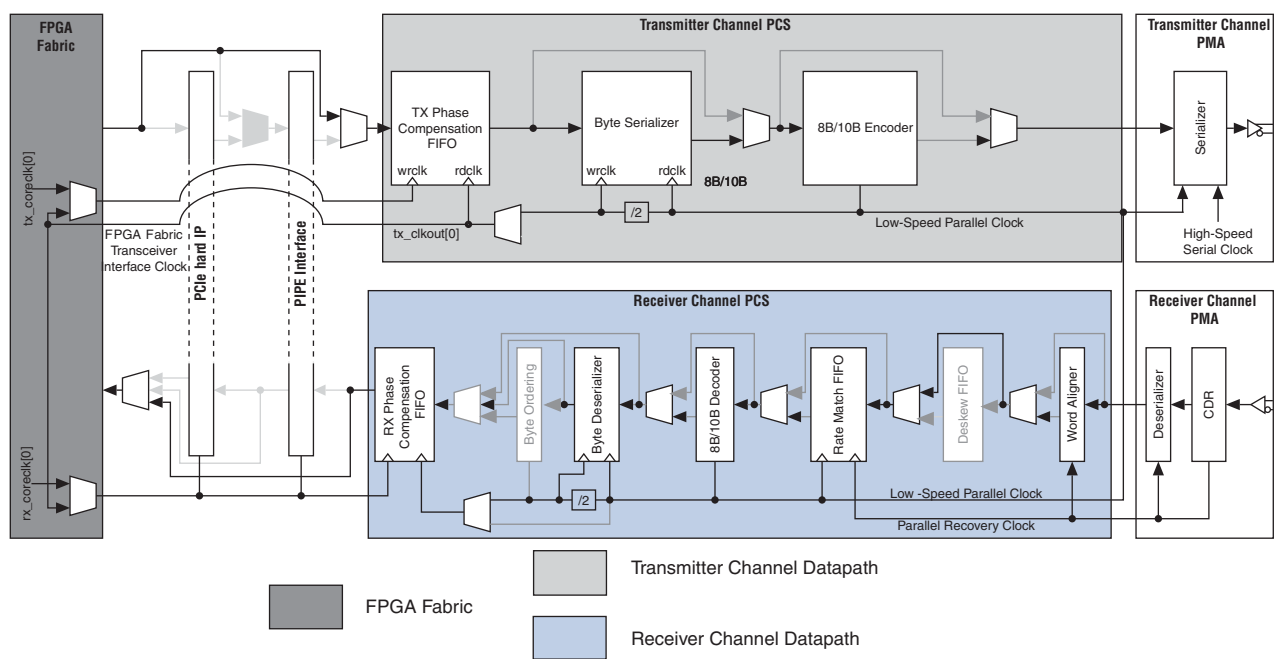
Figure 1-106. Stratix IV GX and GT Transceivers in PCIe Functional Mode



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.


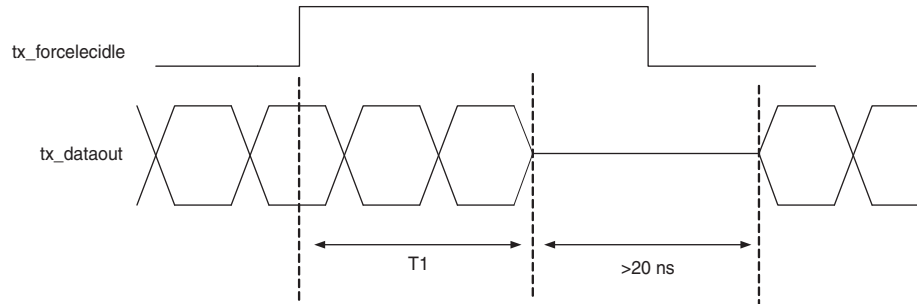
 The minimum period of time for which the `tx_forceidle` signal must be asserted high such that the transmitter buffer stays in electrical idle state for at least 20 ns is pending characterization.

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

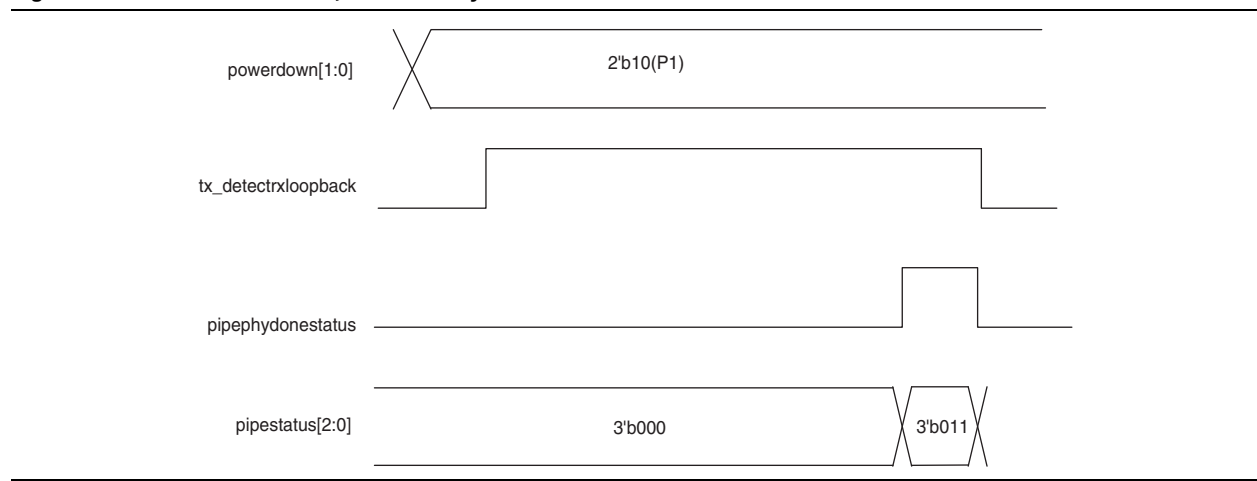
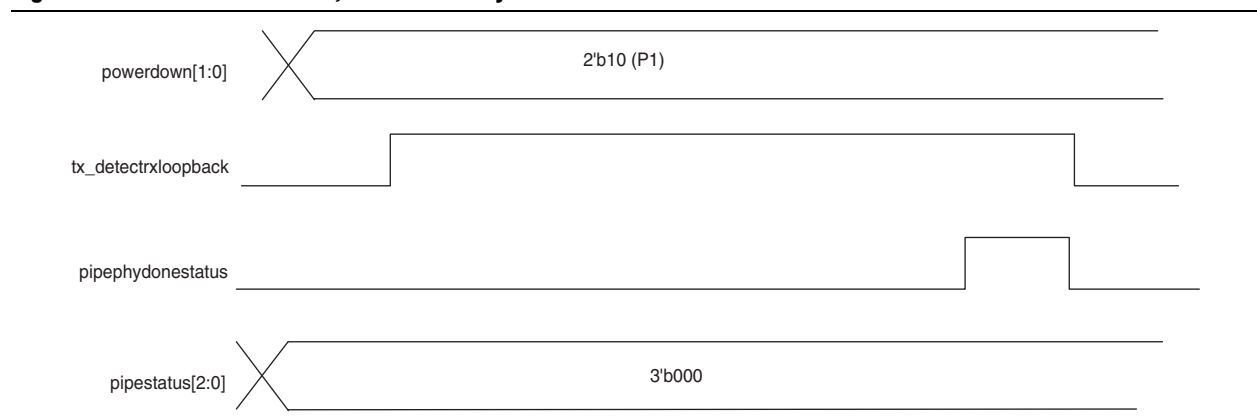


Figure 1–110. Receiver Detect, Unsuccessfully Detected



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

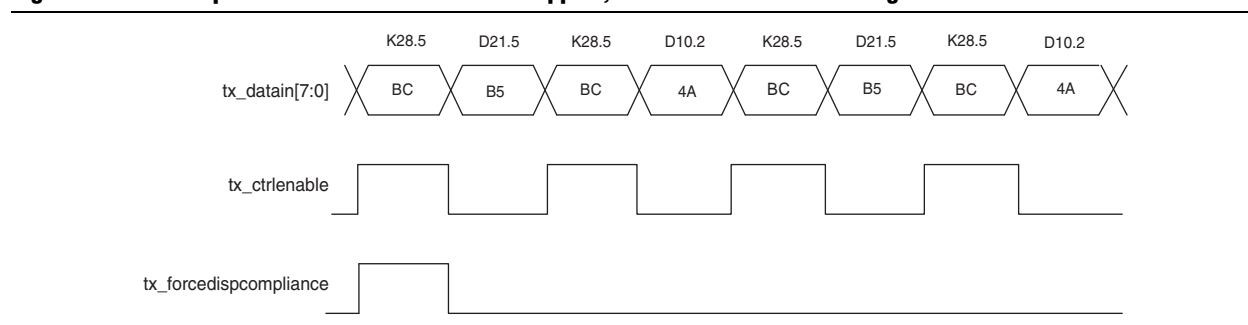
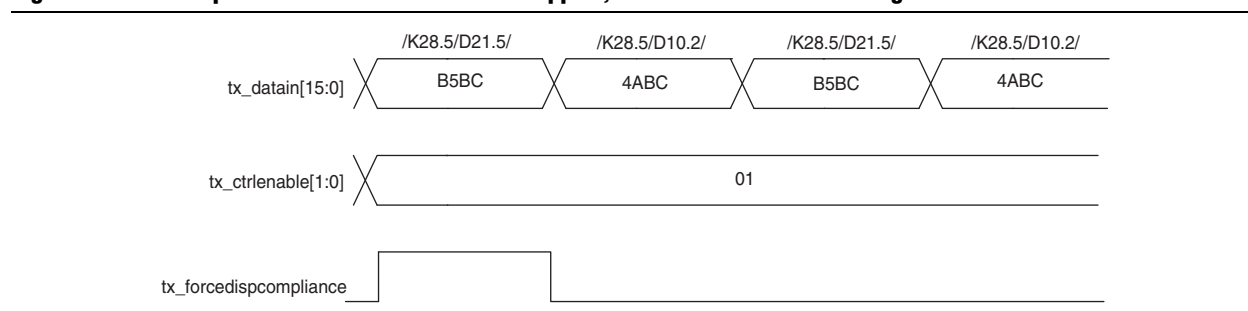


Figure 1-112. Compliance Pattern Transmission Support, 16-Bit Wide Channel Configurations



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

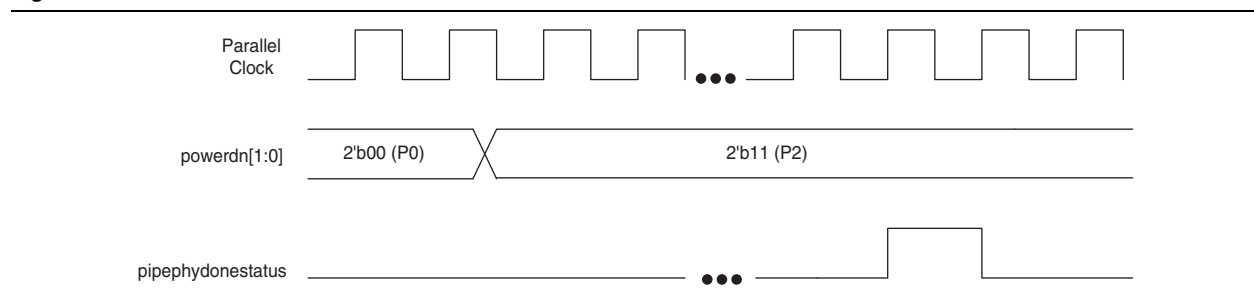
Power State	powerdn	Function	Description
P0	2'b00	Transmits normal data, transmits electrical idle, or enters into loopback mode	Normal operation mode
P0s	2'b01	Only transmits electrical idle	Low recovery time saving state
P1	2'b10	Transmitter buffer is powered down and can do a receiver detect while 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_elecidleinfersel[2:0]` input signal. The Electrical Idle Inference module in each receiver channel indicates whether the electrical idle condition is inferred or not on the `pipeelecidle` signal of that channel. The Electrical Idle Interface module drives the `pipeelecidle` signal high if it infers an electrical idle condition; otherwise, it drives it low.

Table 1-52 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.

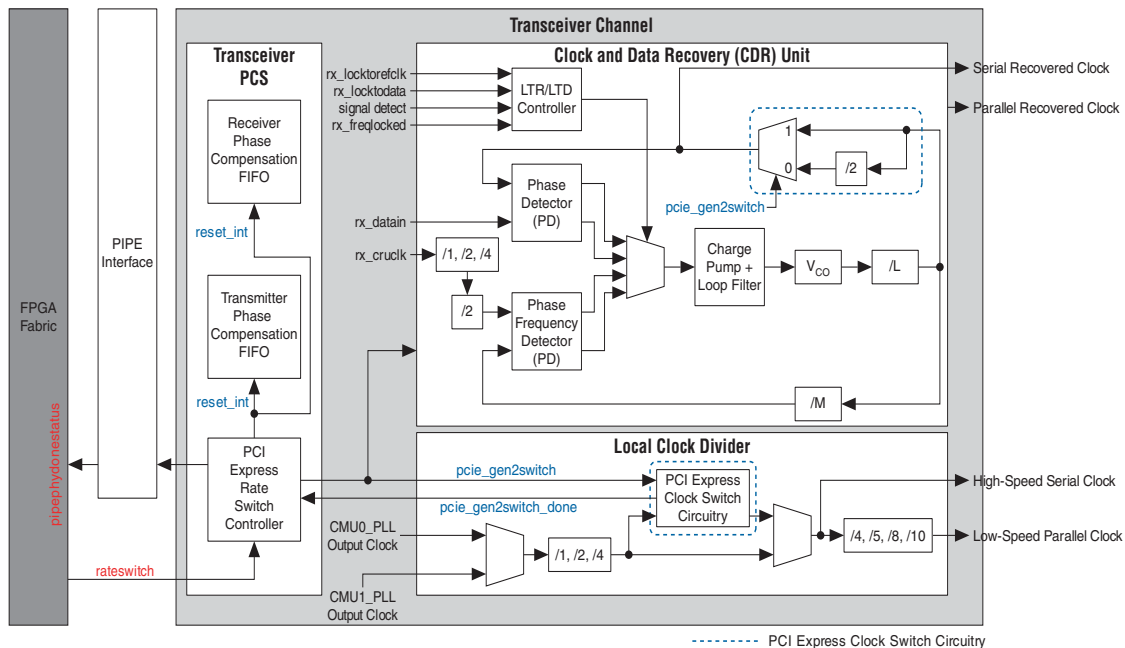
Table 1-53. PCIe Rateswitch Controller and Clock Switch Circuitry

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	CMU0_Channel	CMU0 clock divider in CMU0_Channel	CDR block in receiver PMA of each channel
×8	CMU0_Channel of the master transceiver block	CMU0 clock divider in CMU0_Channel of the master transceiver block	CDR block in receiver PMA of each channel

Dynamic Switch Between Gen1 (2.5 Gbps) and Gen2 (5 Gbps) Signaling Rates in 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.


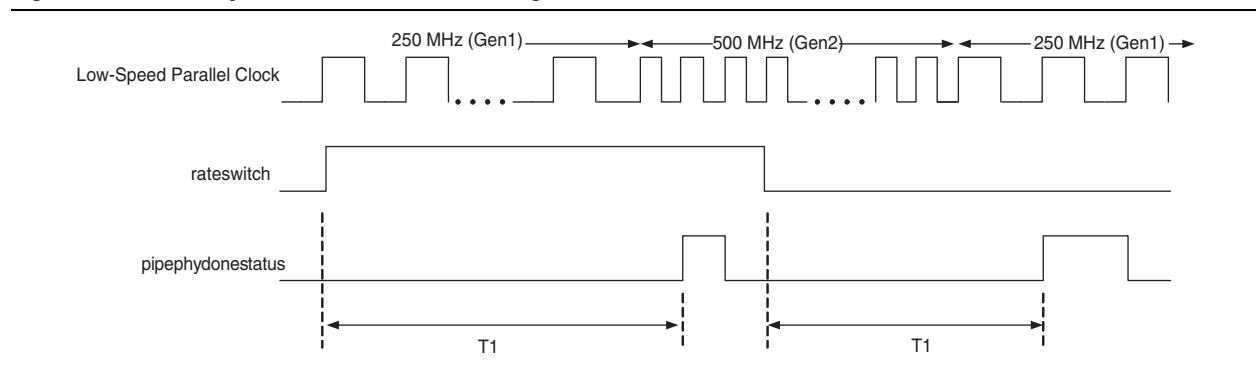
 Time T1 from a transition on the `rateswitch` signal to the assertion of `pipephydonestatus` is pending characterization.

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

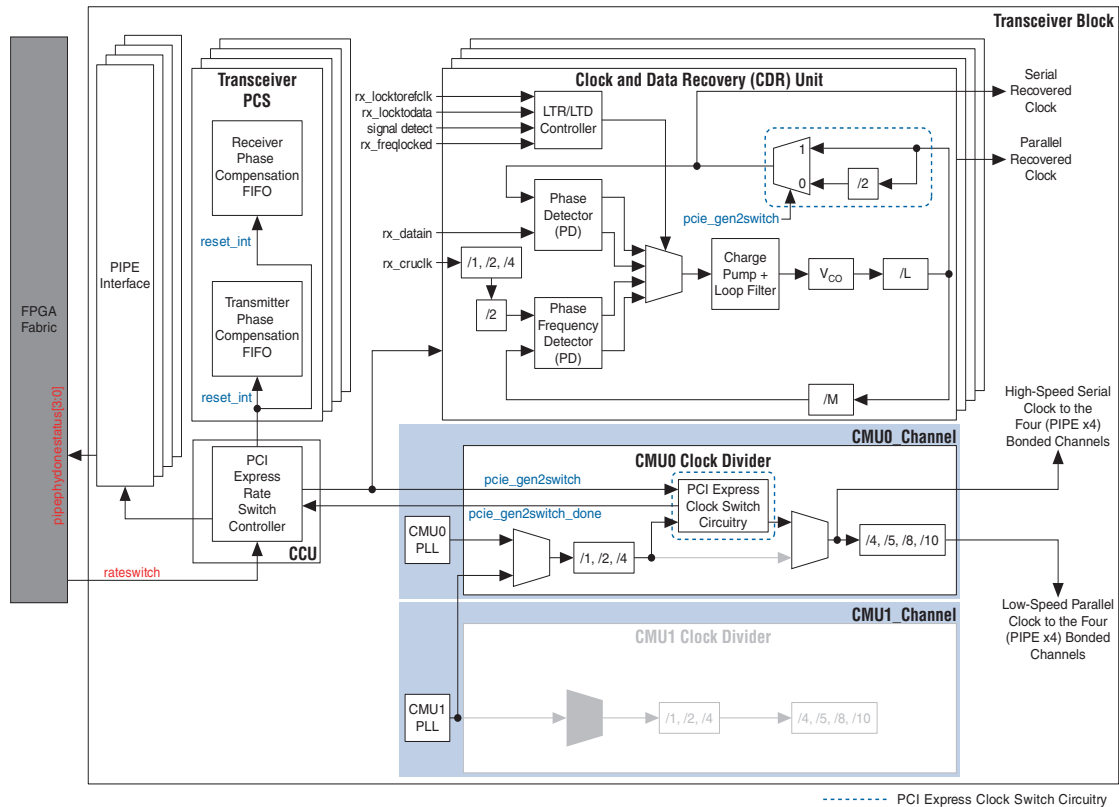


As a result of the signaling rateswitch between Gen1 (2.5 Gbps) and Gen2 (5 Gbps), the FPGA fabric-transceiver interface clock switches between 125 MHz and 250 MHz. The FPGA fabric-transceiver interface clock clocks the read side and write side of the transmitter phase compensation FIFO and the receiver 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.


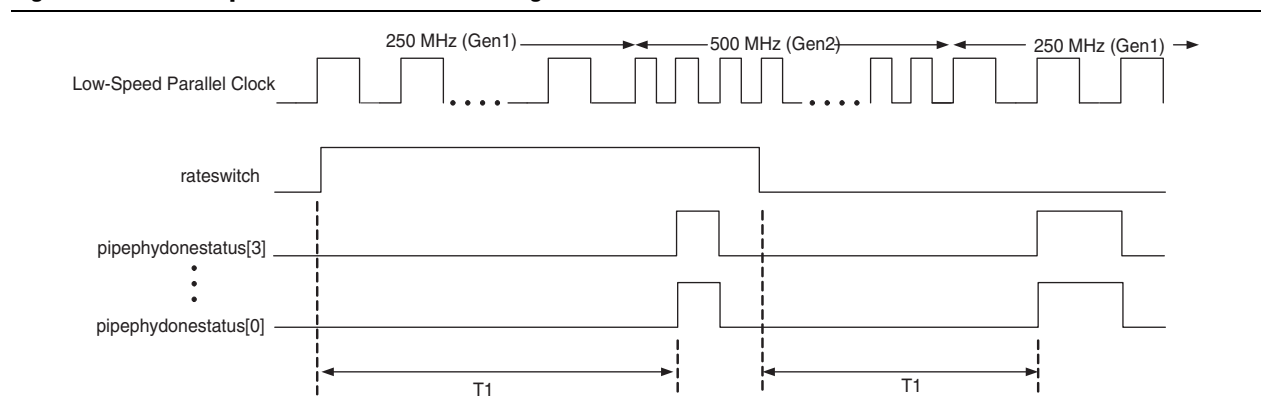
 Time T1 from a transition on the `rateswitch` signal to the assertion of `pipephydonestatus` is pending characterization.

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



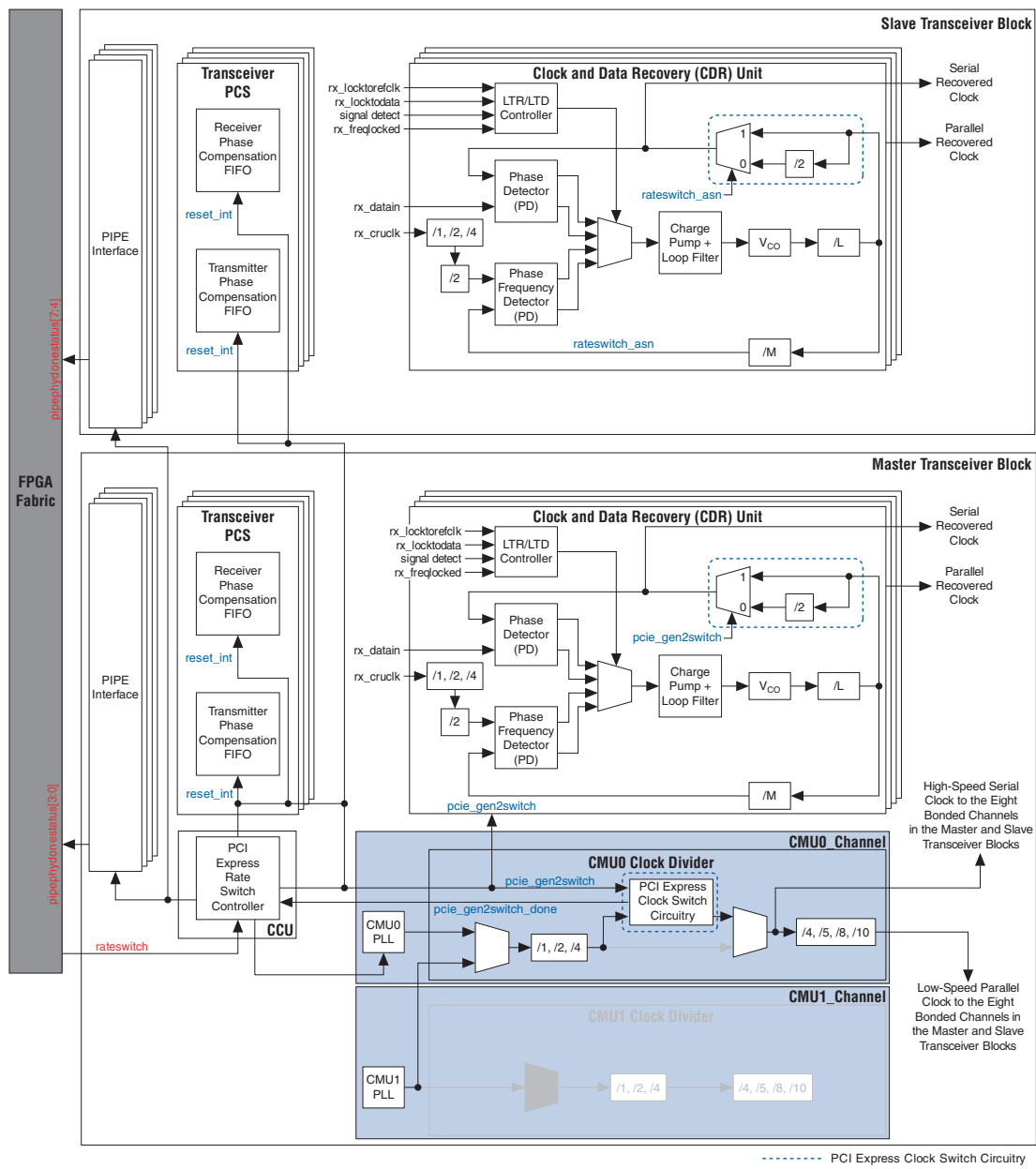
As a result of the signaling rateswitch between Gen1 (2.5 Gbps) and Gen2 (5 Gbps), the FPGA fabric-transceiver interface clock switches between 125 MHz and 250 MHz. The FPGA fabric-transceiver interface clock clocks the read side and write side of the transmitter phase compensation FIFO and the receiver 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

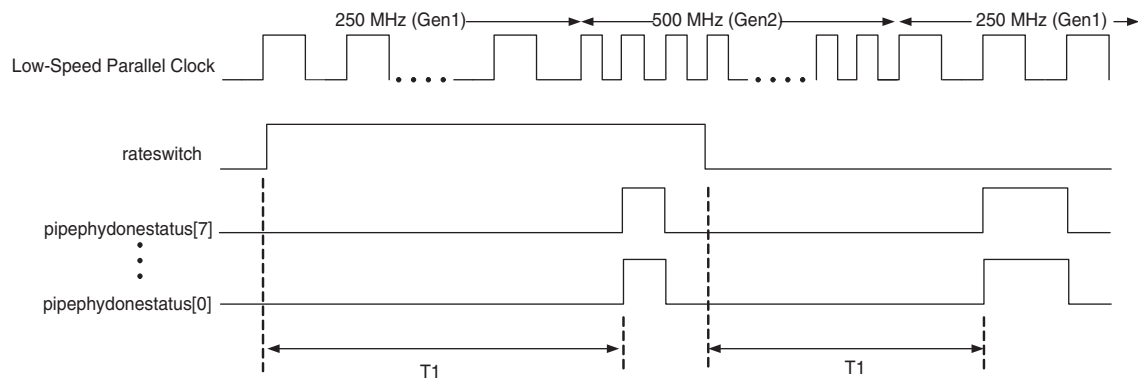
The PCIe clock switch circuitry in the `CMU0` clock divider of the master transceiver block performs the clock switch between 250 MHz and 500 MHz on the low-speed parallel clock when switching between Gen1 (2.5 Gbps) and Gen2 (5 Gbps) signaling rates. It indicates successful completion of clock switch 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.


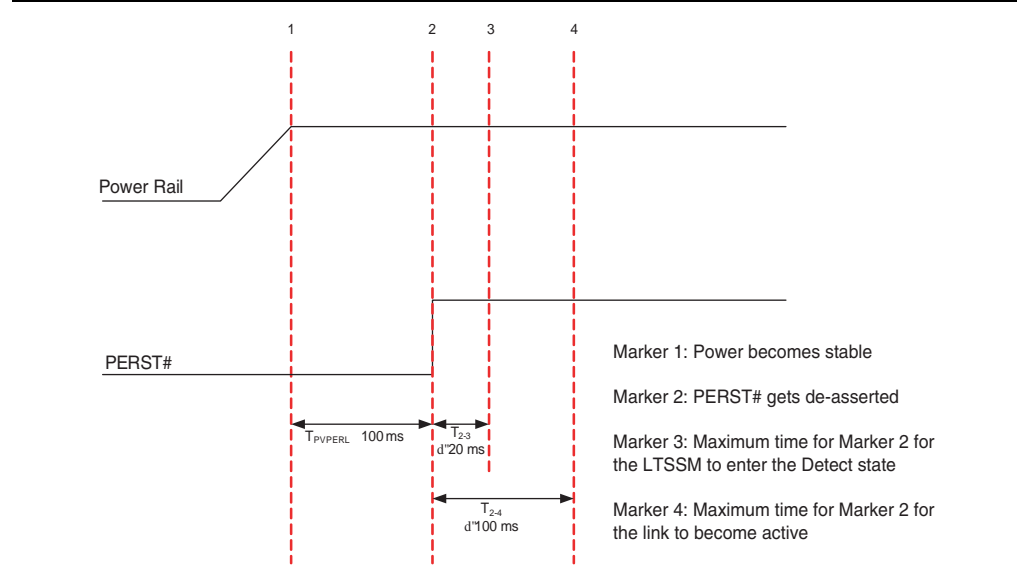
 The link being active is interpreted as the physical layer device coming out of electrical idle in the L0 state of the LTSSM state machine.

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*.

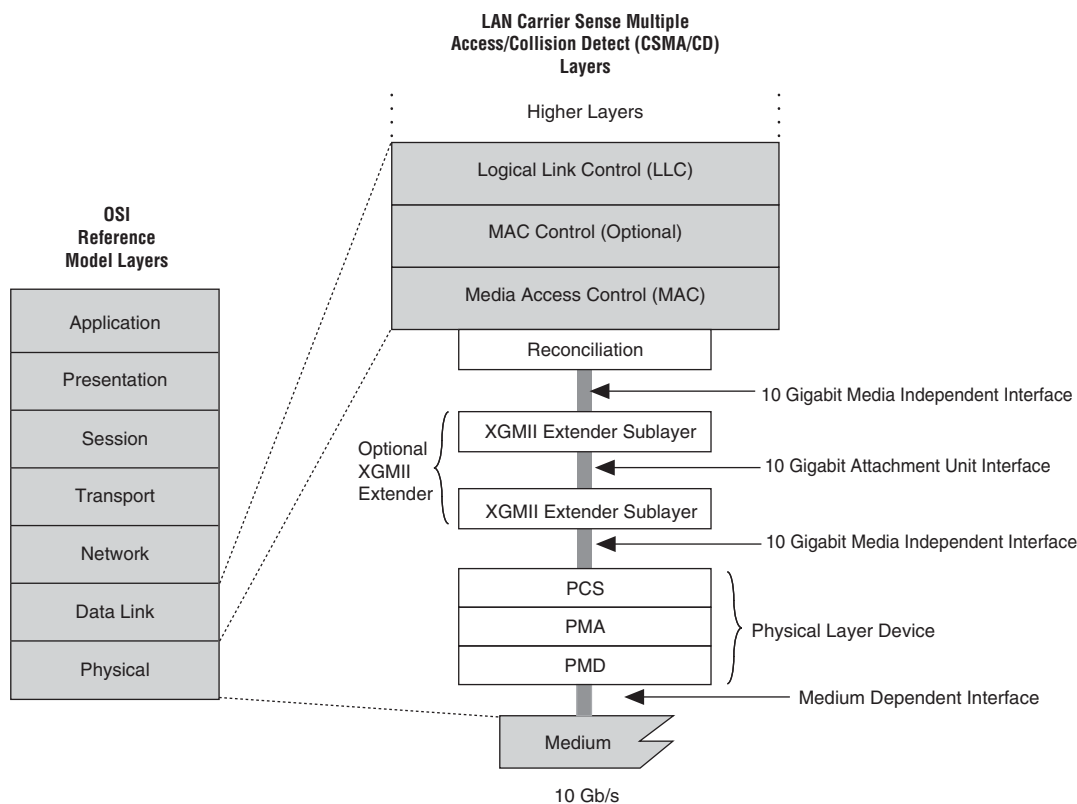
XAUI Mode

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

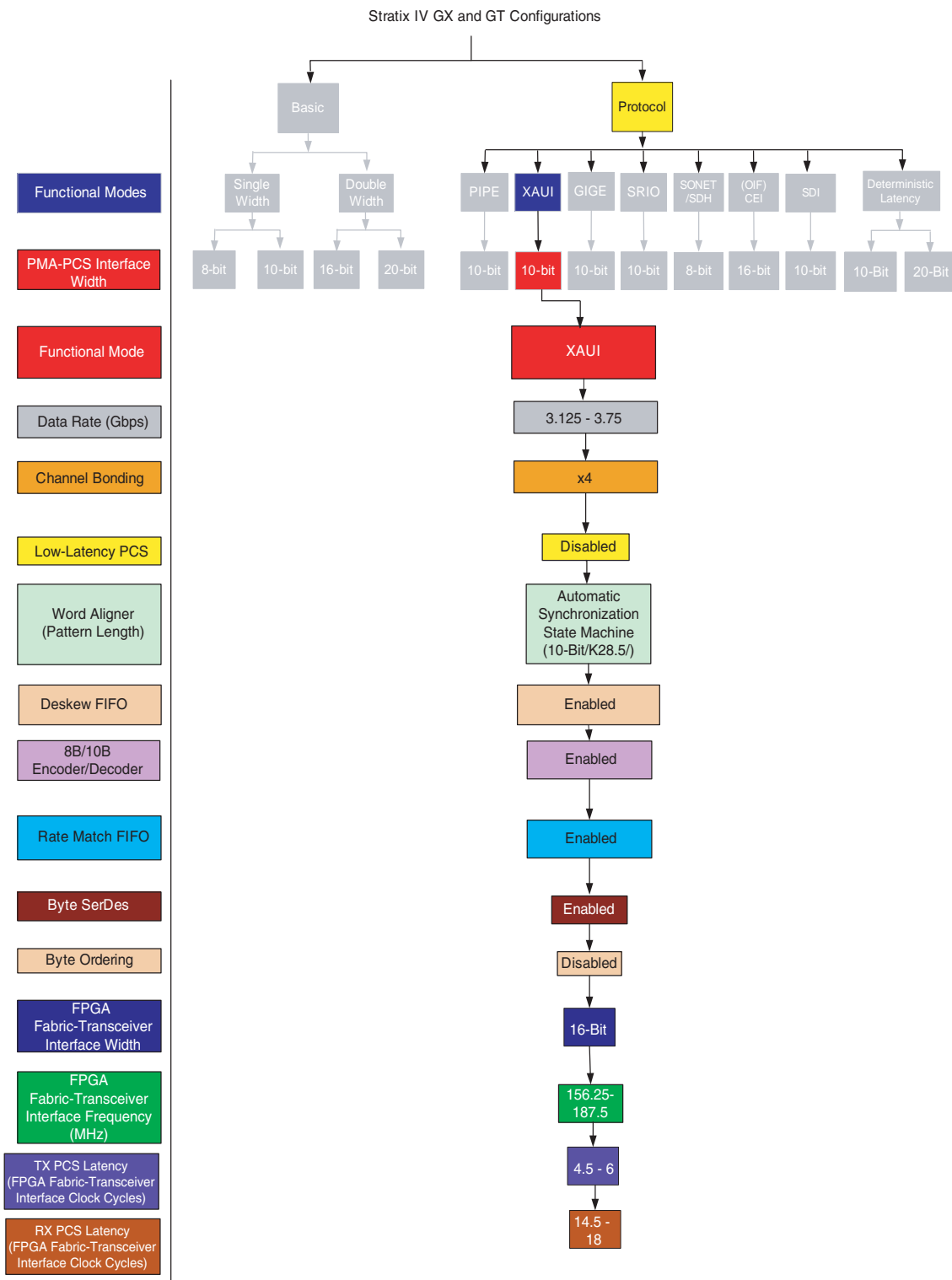
Code	Ordered Set	Number of Code Groups	Encoding
I	Idle		Substitute for XGMII Idle
K	Synchronization column	4	/K28.5/K28.5/K28.5/K28.5/
R	Skip column	4	/K28.0/K28.0/K28.0/K28.0/
A	Align column	4	/K28.3/K28.3/K28.3/K28.3/

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.

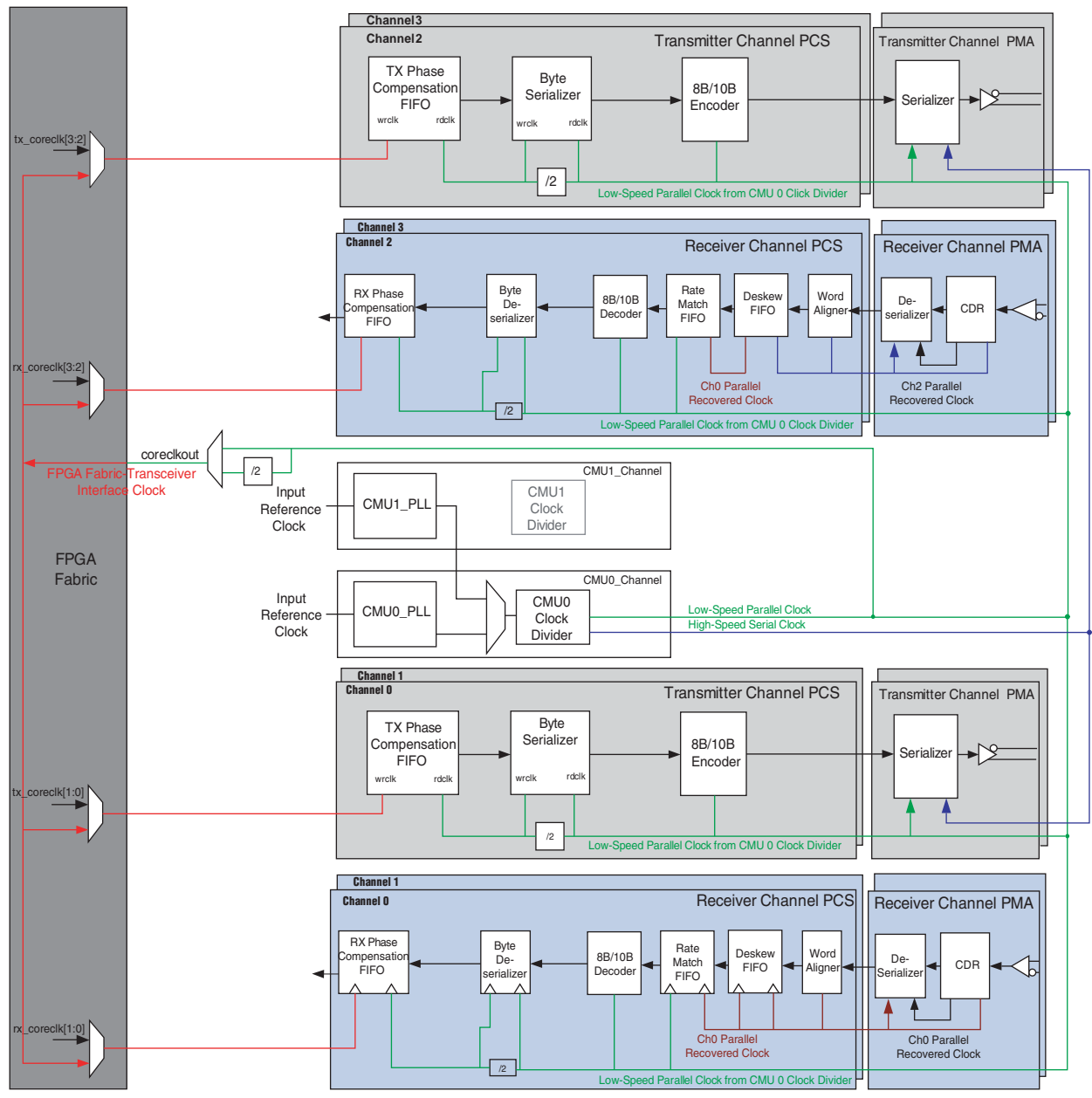
Figure 1-123. Stratix IV GX and GT XAUI Mode Configuration



XAUI Mode Datapath

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

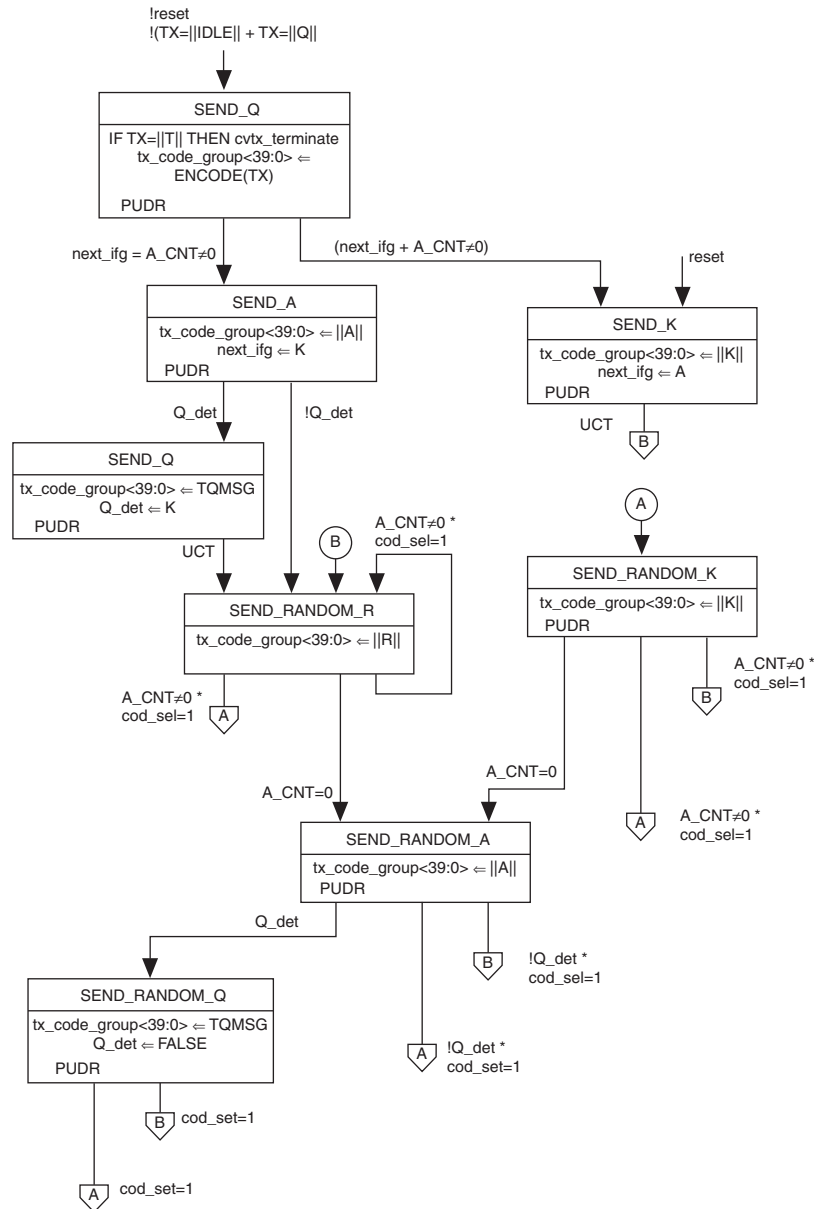
Figure 1-124. Transceiver Datapath in XAUI Mode



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

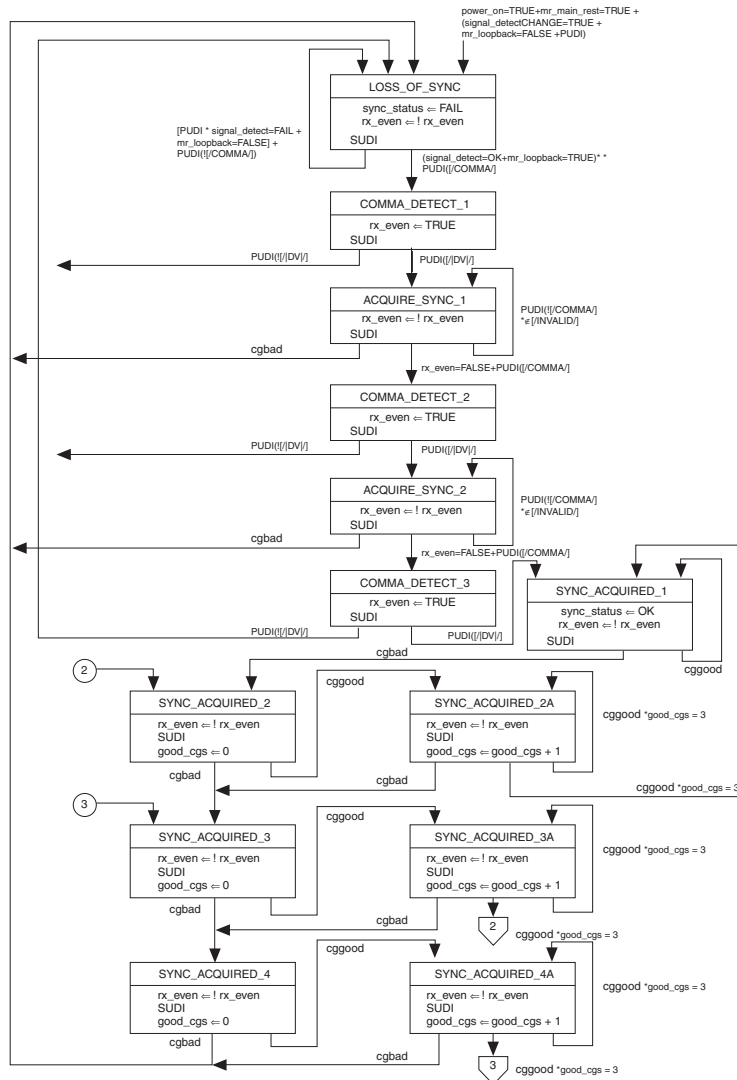
Note to Table 1-58:

(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.

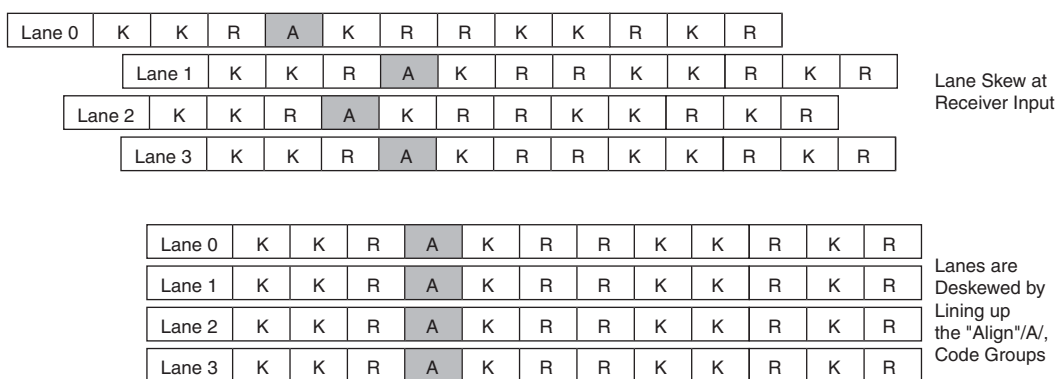
The XAUI protocol requires the physical layer device to implement a deskew circuitry to align all four channels. To enable the deskew circuitry at the receiver to align the four channels, the transmitter sends a /A/ (/K28.3/) code group simultaneously on all four channels during inter-packet gap. The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

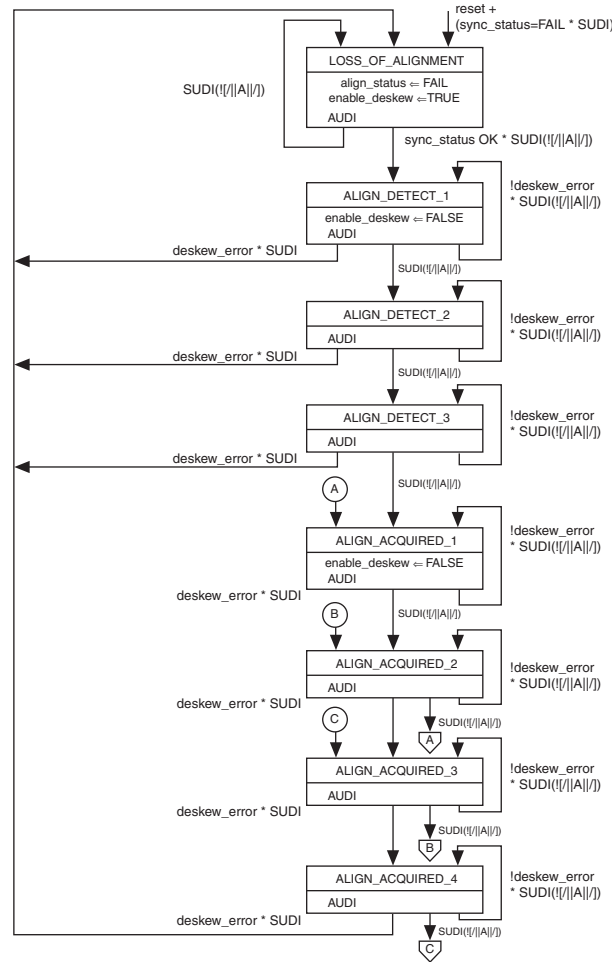
Figure 1-127. Receiver Input Lane Skew in XAUI Mode



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_rmfiodeinserted, 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_rmfiodeinserted 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

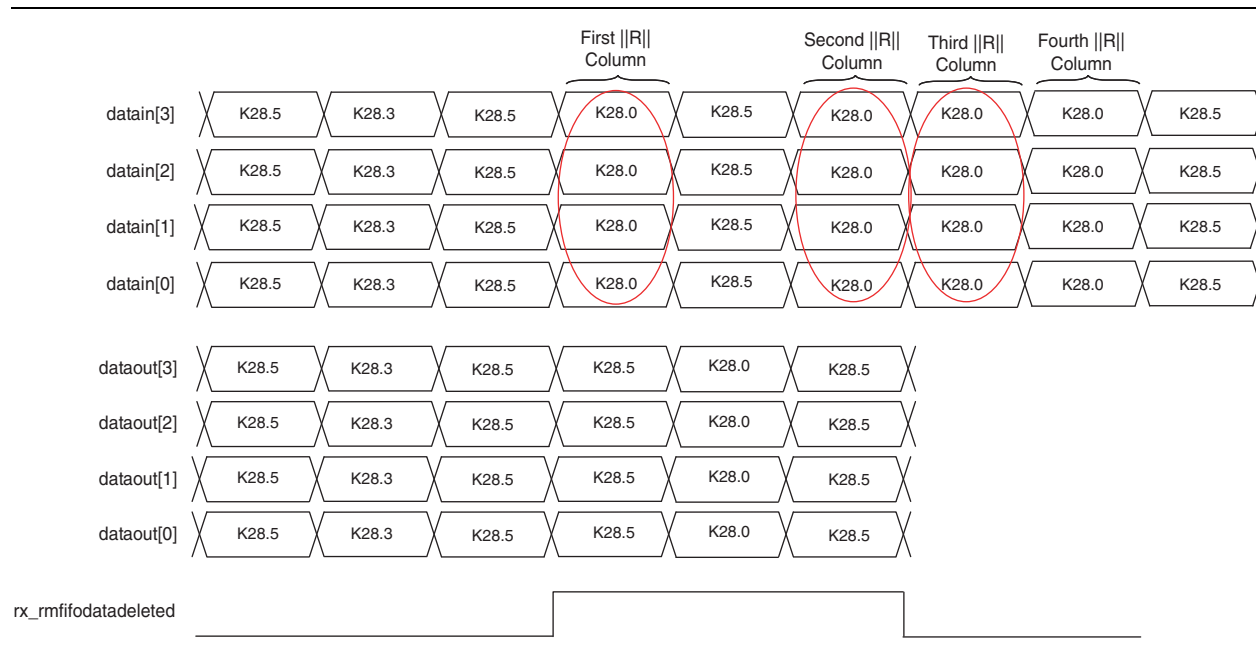
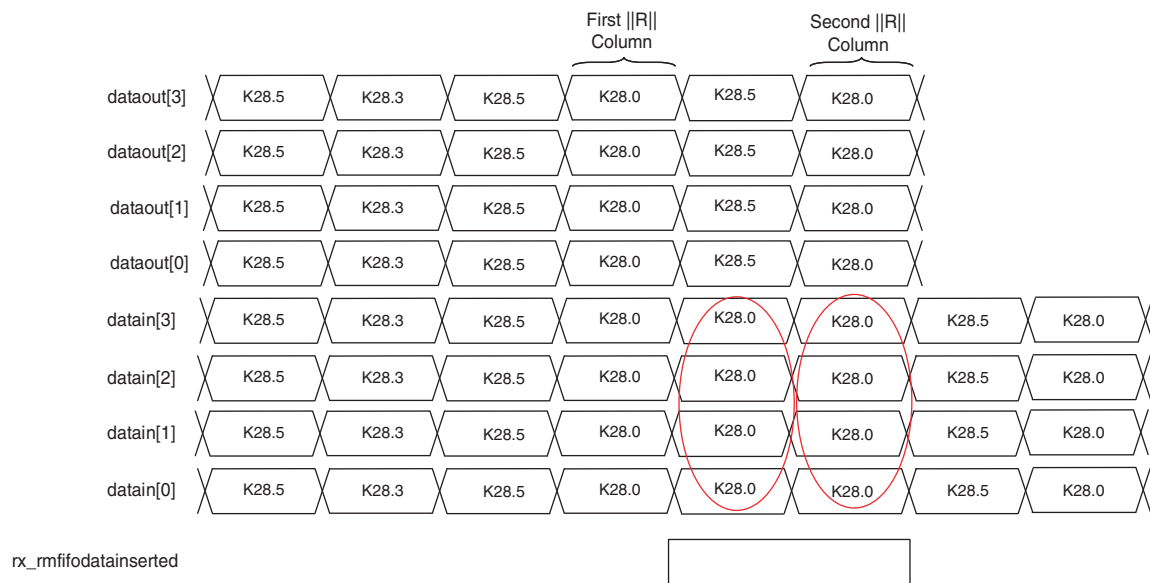


Figure 1-130 shows an example of rate match insertion in the case where two $||R||$ columns are required to be inserted.

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

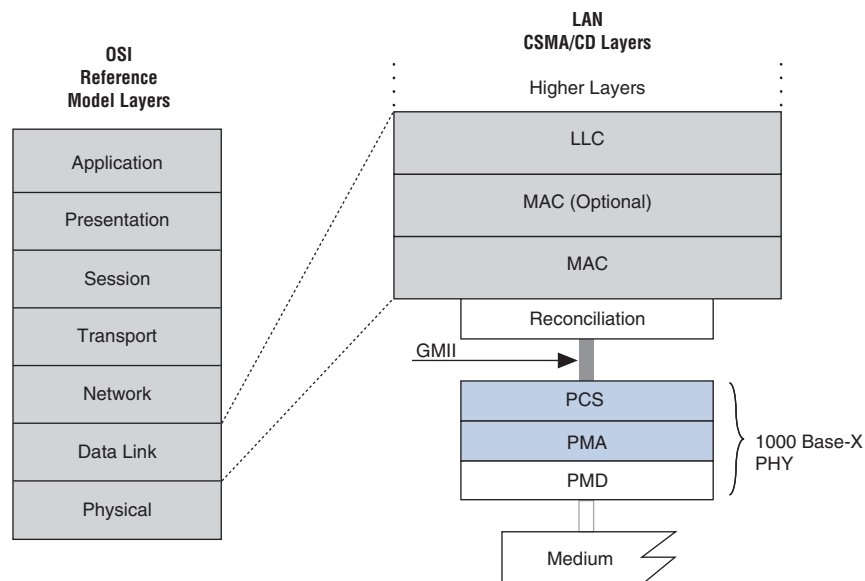
IEEE 802.3 defines the 1000 Base-X PHY as an intermediate, or transition, layer that interfaces various physical media with the media access control (MAC) in a gigabit ethernet system. It shields the MAC layer from the specific nature of the underlying medium. The 1000 Base-X PHY is divided into three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

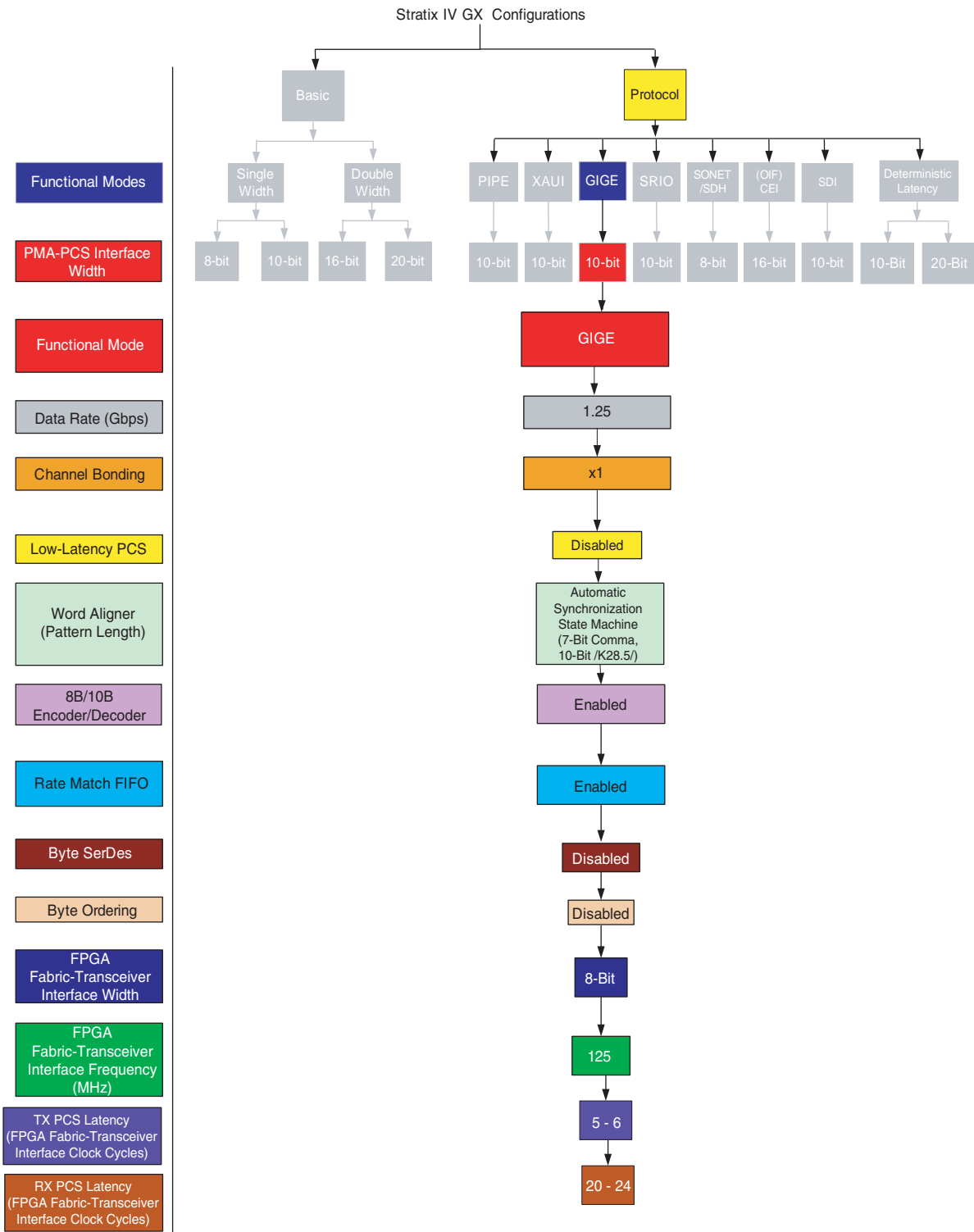
- 8B/10B encoding and decoding
- Synchronization
- Clock recovery from the encoded data forwarded by the receiver PMD
- Optional `rx_recovclkout` port enables recovered clock at the pin level (use with VCXO)
- Serialization and deserialization

 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



Functional Modes

PMA-PCS Interface Width

Functional Mode

Data Rate (Gbps)

Channel Bonding

Low-Latency PCS

Word Aligner (Pattern Length)

8B/10B Encoder/Decoder

Rate Match FIFO

Byte SerDes

Byte Ordering

FPGA Fabric-Transceiver Interface Width

FPGA Fabric-Transceiver Interface Frequency (MHz)

TX PCS Latency (FPGA Fabric-Transceiver Interface Clock Cycles)

RX PCS Latency (FPGA Fabric-Transceiver Interface Clock Cycles)

GIGE Mode Datapath

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

Figure 1-133. GIGE Mode Datapath

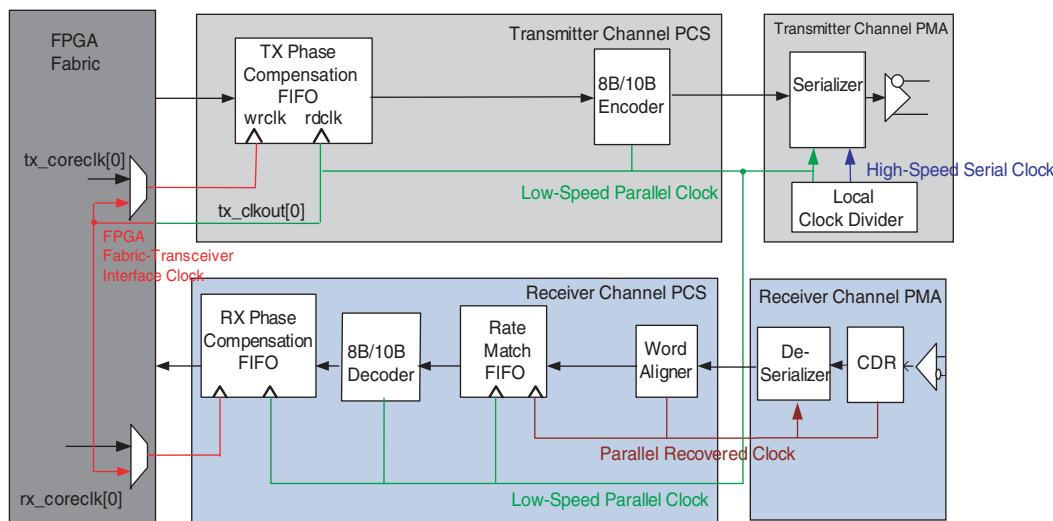


Table 1-62 lists the transceiver datapath clock frequencies in GIGE functional mode.

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.


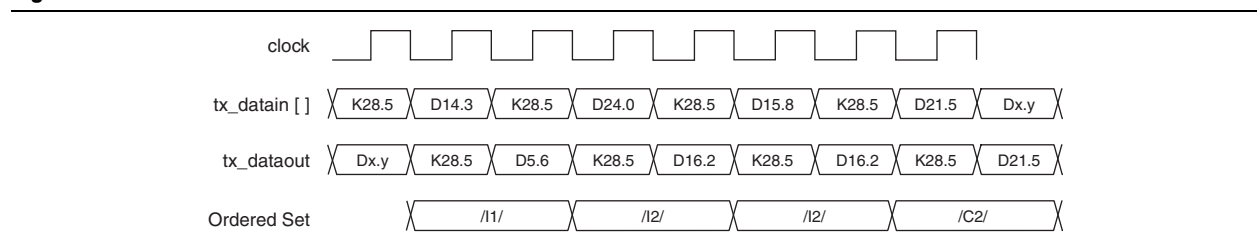
 Note that /D14.3/, /D24.0/, and /D15.8/ are replaced by /D5.6/ or /D16.2/ (for /I1/, /I2/ ordered sets). /D21.5/ (part of the /C1/ order set) is not replaced.

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

Figure 1-134. Automatic Ordered Set Generation

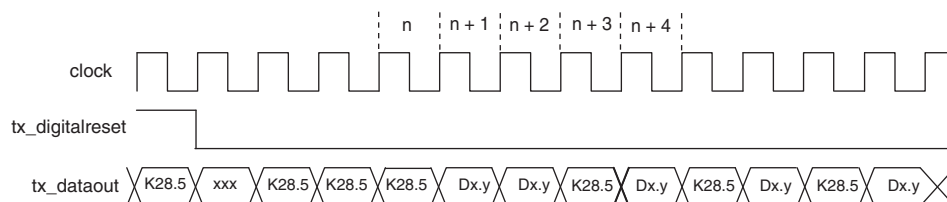
Reset Condition

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.

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

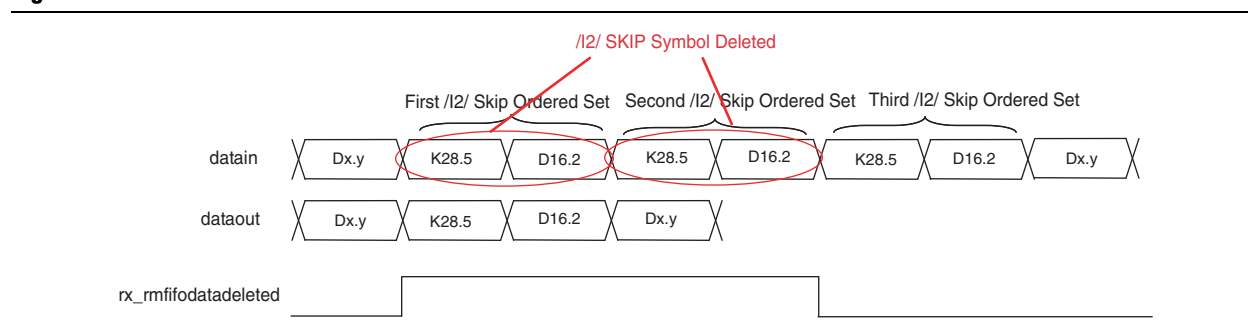
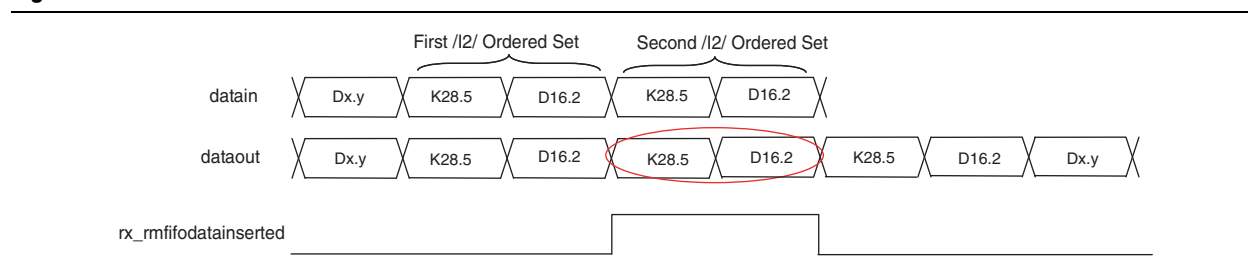



Figure 1-138 shows an example of rate match FIFO insertion in the case where one symbol is required to be inserted. Because the rate match FIFO can only delete /I2/ ordered set, it inserts one /I2/ ordered set (two symbols inserted).

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

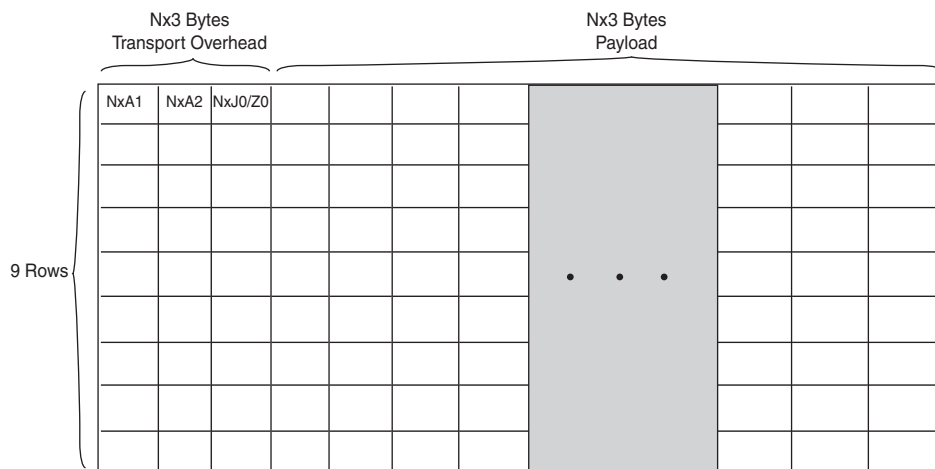
SONET/SDH is one of the most common serial-interconnect protocols used in backplanes deployed in communications and telecom applications. SONET/SDH defines various optical carrier (OC) sub-protocols for carrying signals of different capacities through a synchronous optical hierarchy.

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

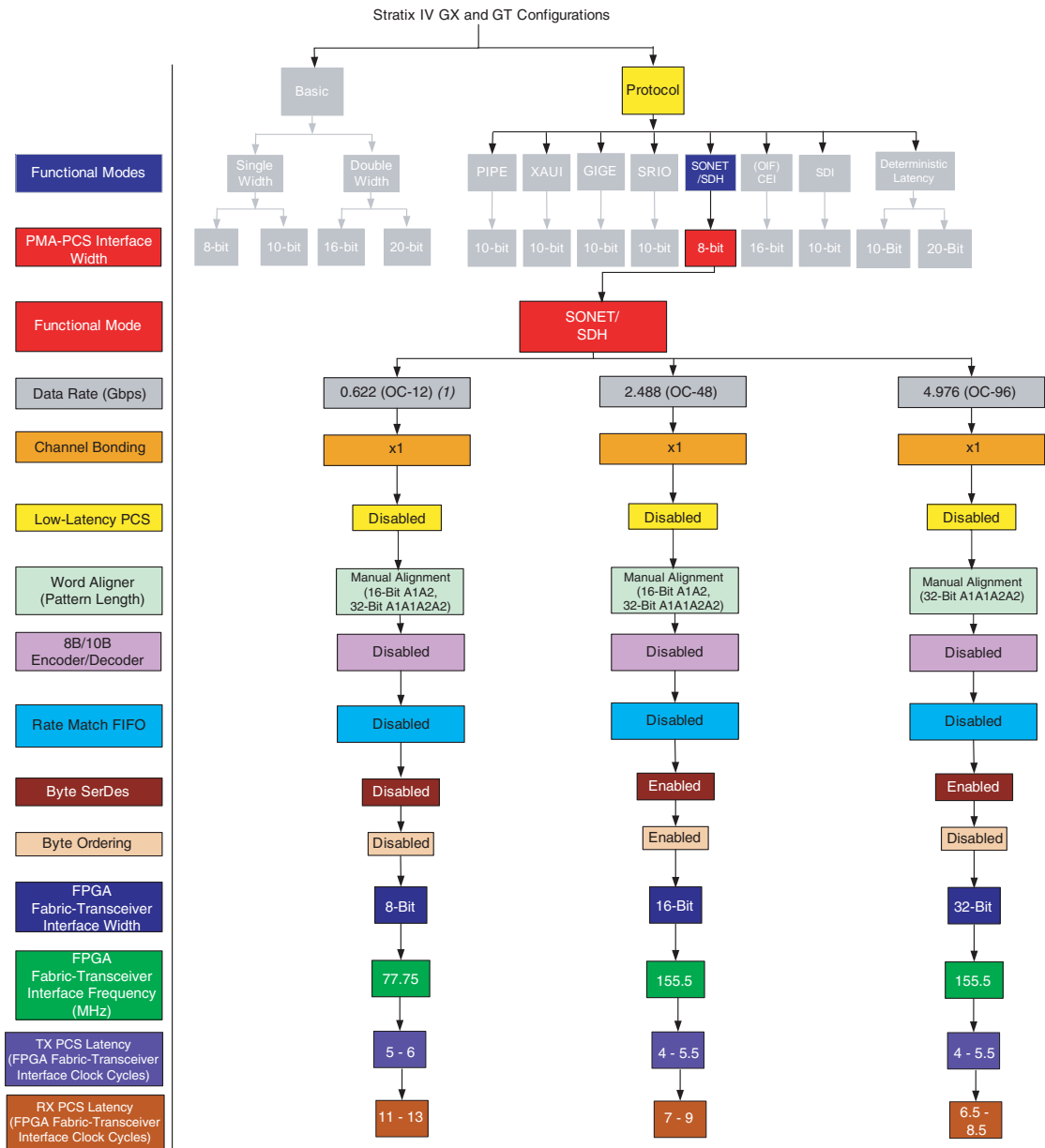
You can employ Stratix IV GX and GT transceivers as physical layer devices in a SONET/SDH system. These transceivers provide support for SONET/SDH protocol-specific functions and electrical features; for example, alignment to A1A2 or A1A1A2A2 pattern.

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



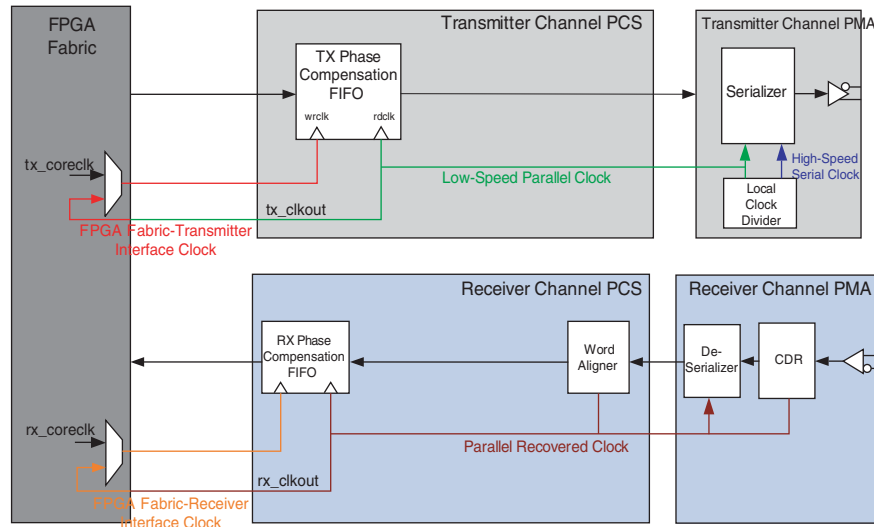
Note to Figure 1-41:

(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.

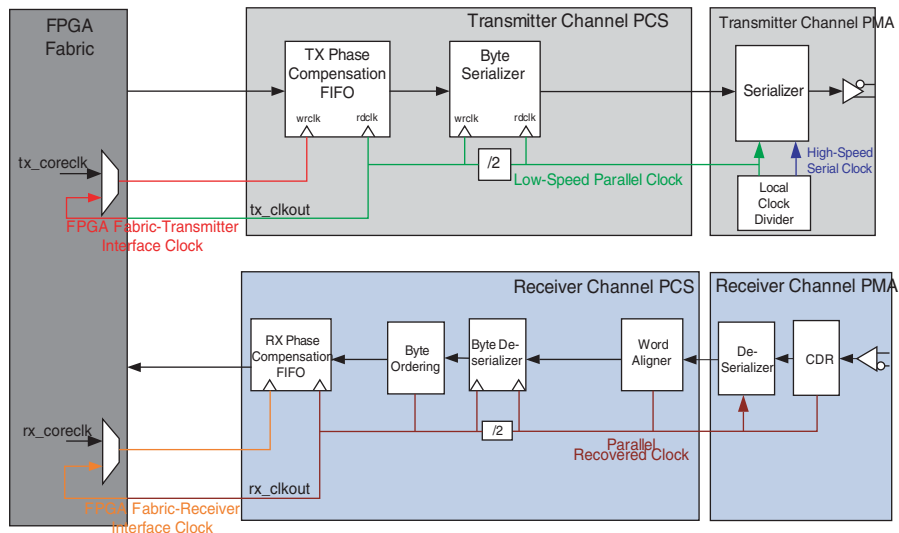
Figure 1-141. SONET/SDH OC-12 Datapath



SONET/SDH OC-48 Datapath

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

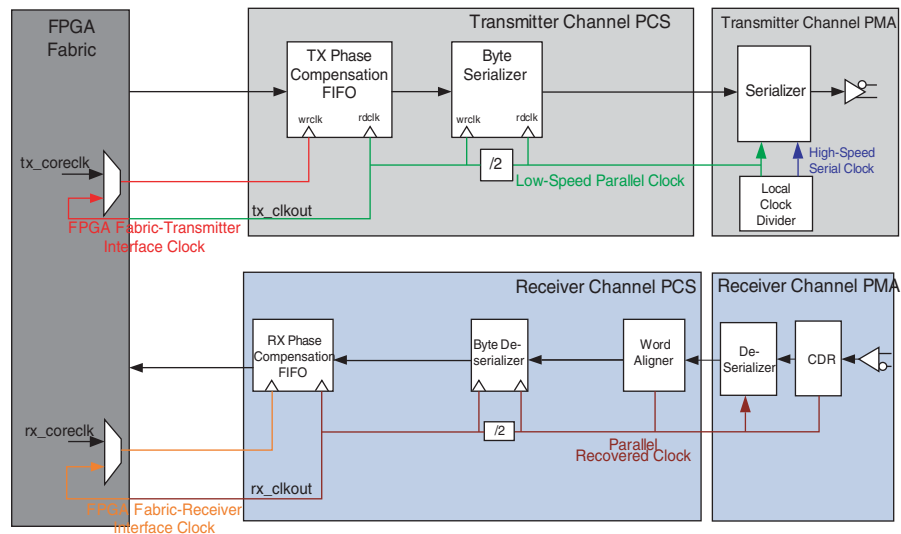
Figure 1-142. SONET/SDH OC-48 Datapath



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.


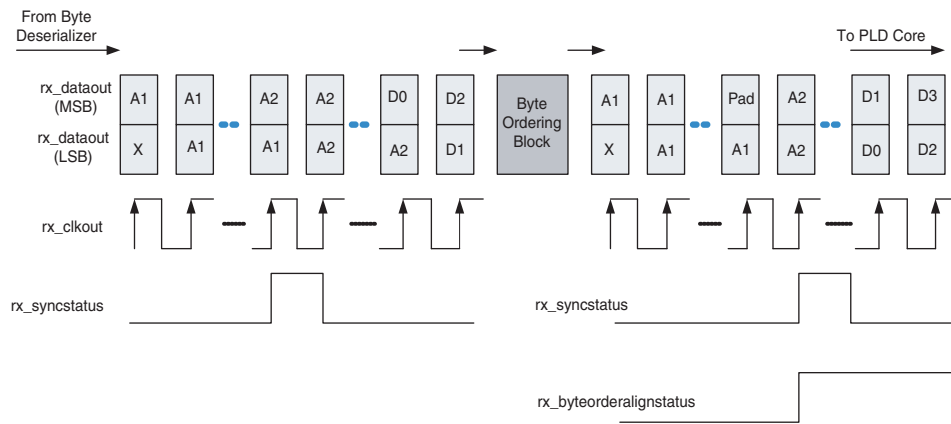
 The PAD character is defaulted to the A1 byte of the A1A2 alignment pattern.

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

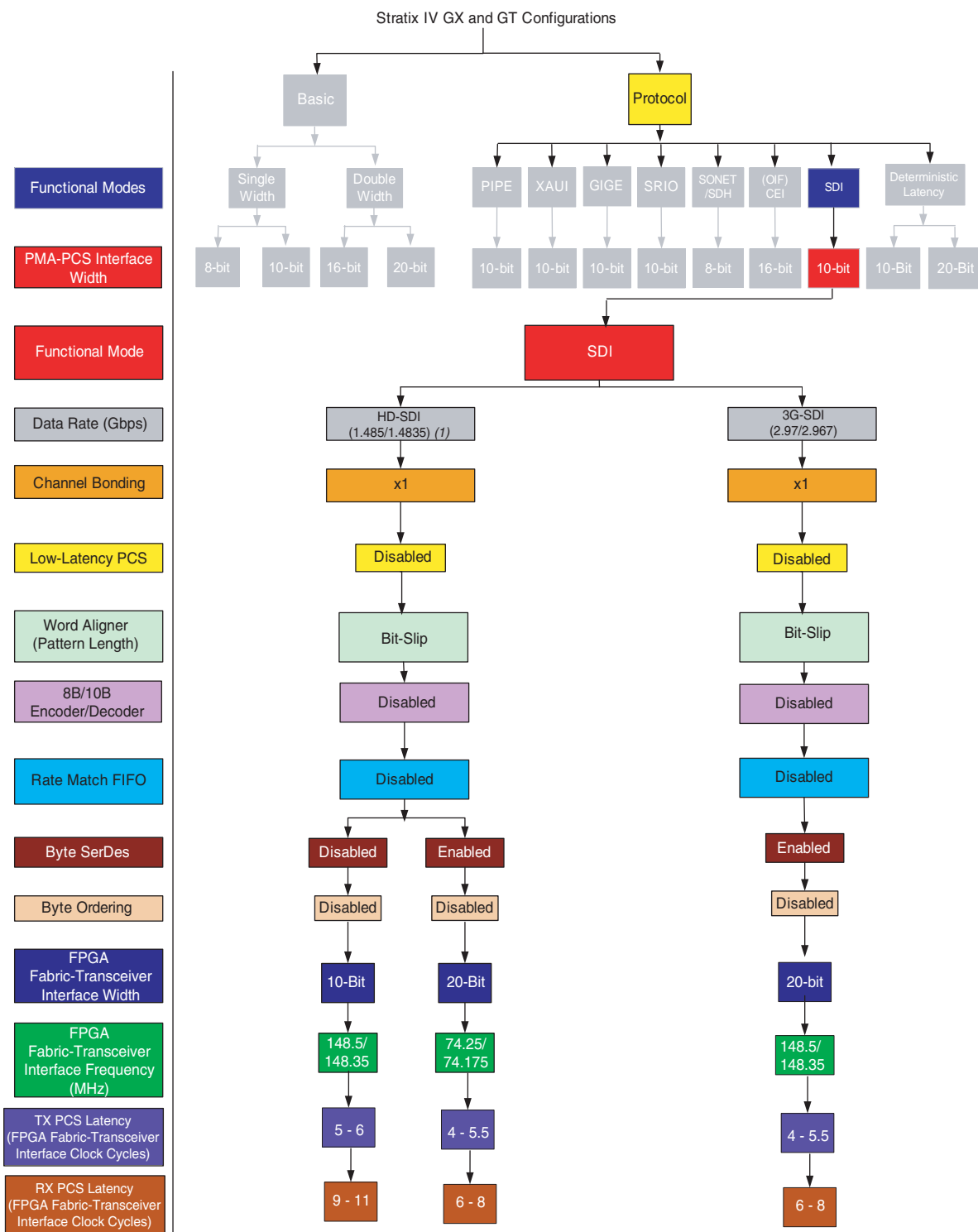
Configuration	Data Rate (Mbps)	REFCLK Frequencies (MHz)	FPGA Fabric-Transceiver Interface Width
HD ⁽¹⁾	1485	74.25, 148.5	10 bit and 20 bit
	1483.5	74.175, 148.35	10 bit and 20 bit
3G ⁽²⁾	2970	148.5, 297	Only 20-bit interface allowed in 3G
	2967	148.35, 296.7	Only 20-bit interface allowed in 3G

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



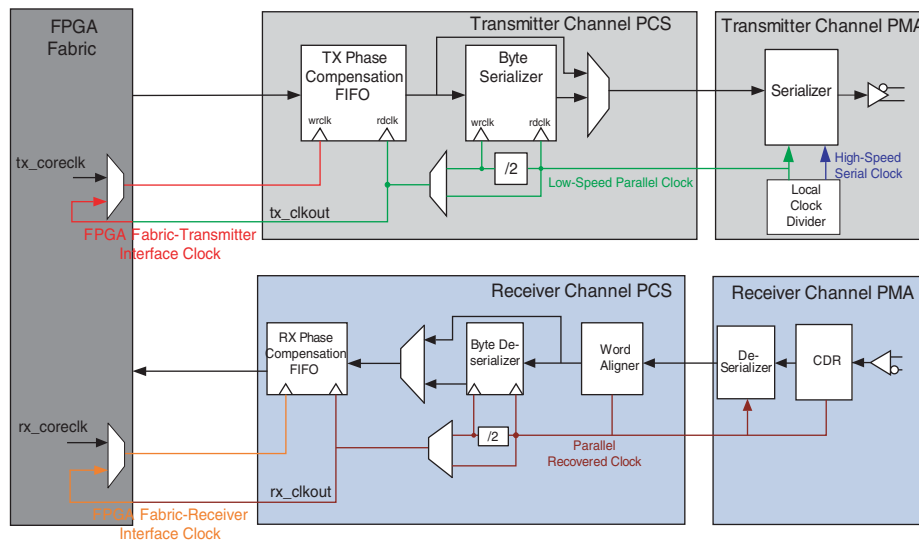
Note to Figure 1-45:

(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

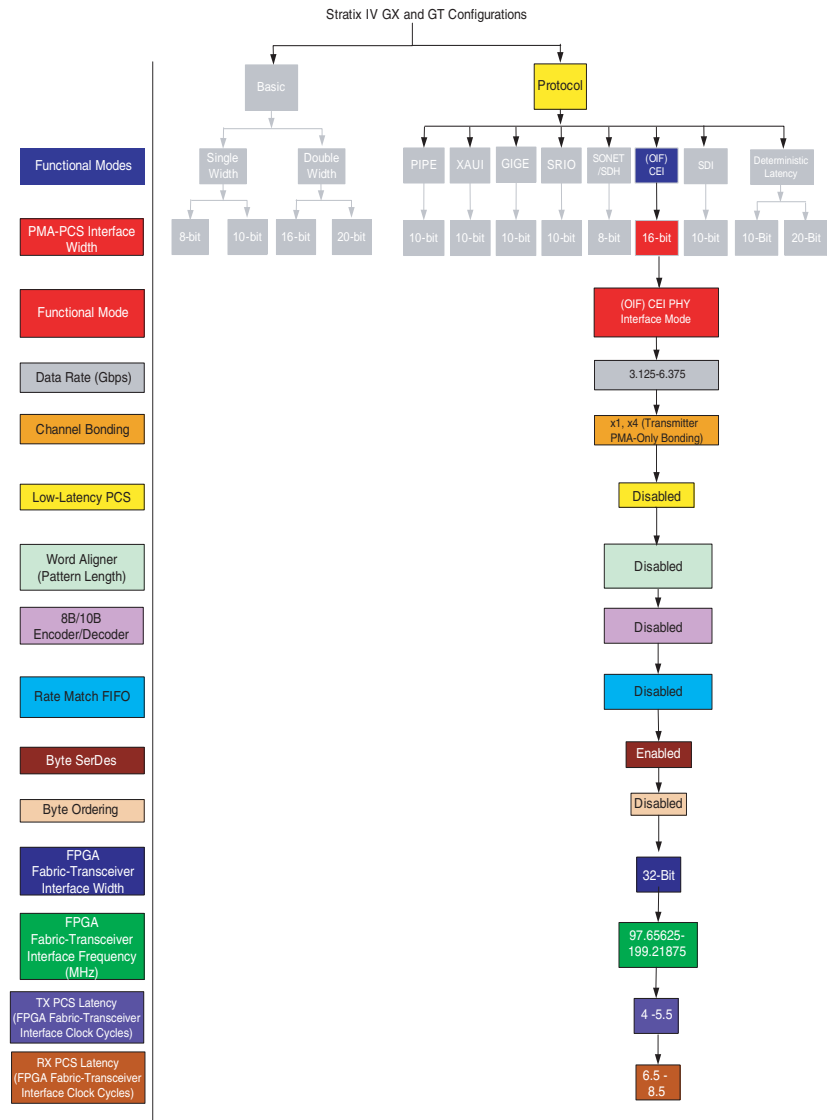
In SDI systems, the word aligner in the receiver datapath is not useful because word alignment and framing happens after de-scrambling. Altera recommends driving the ALTGX megafunction rx_bitslip signal low to avoid having the word aligner insert bits in the received data stream.

(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.

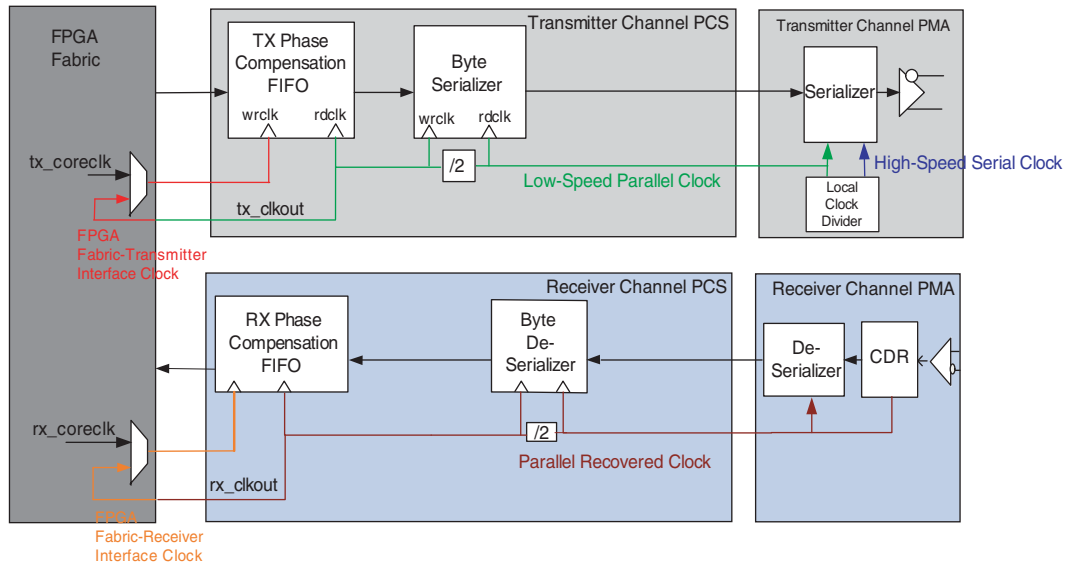
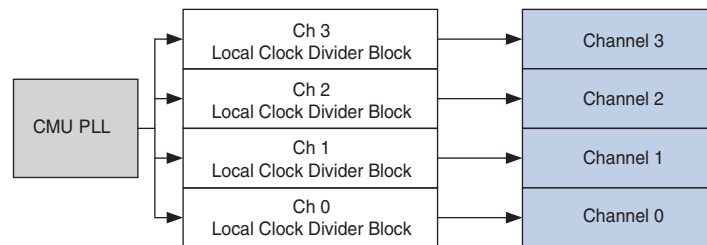
Figure 1-148. (OIF) CEI PHY Interface Mode Datapath

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:

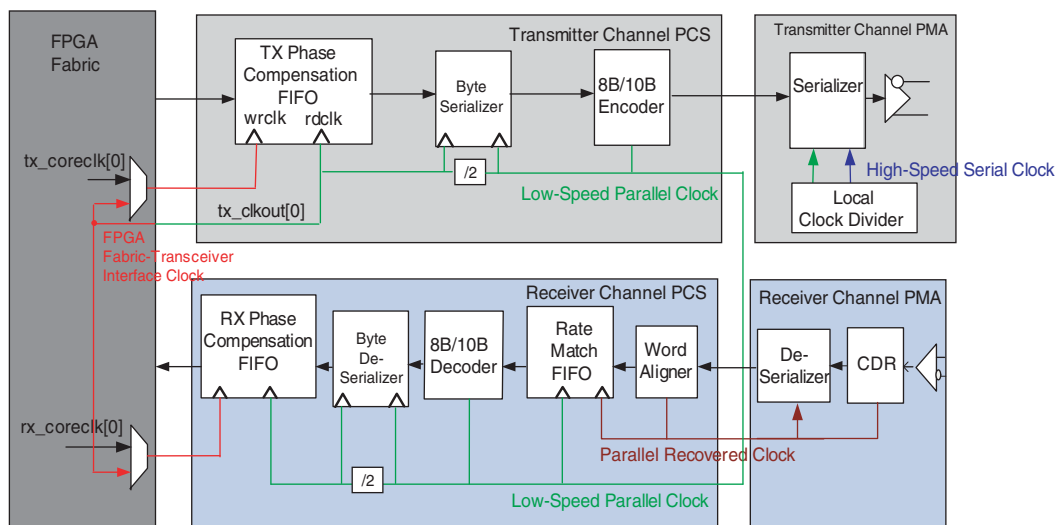
- 1.25 Gbps
- 2.5 Gbps
- 3.125 Gbps

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.

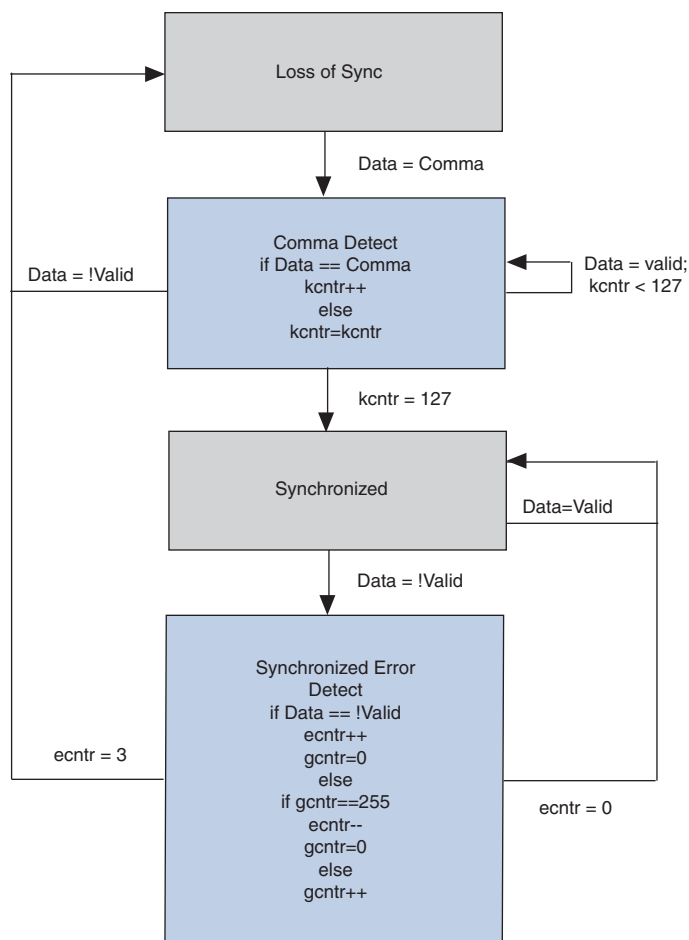
Table 1-67 lists the ALTGX megafunction synchronization state machine parameters when configured in Serial RapidIO mode.

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

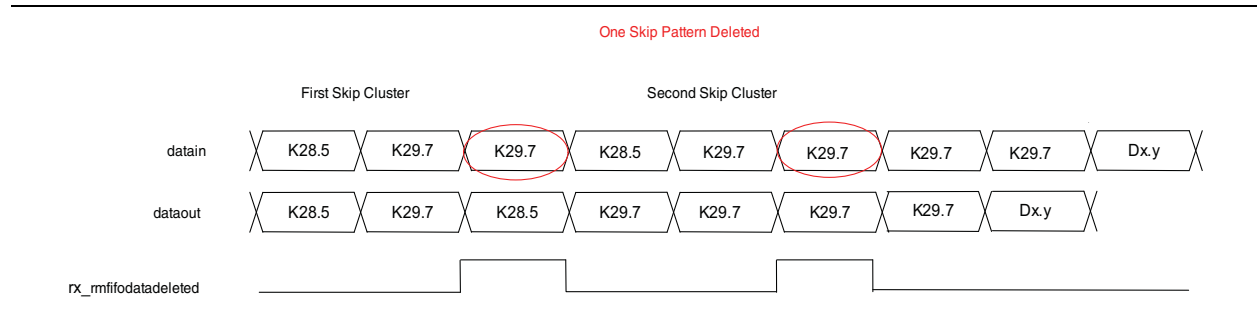
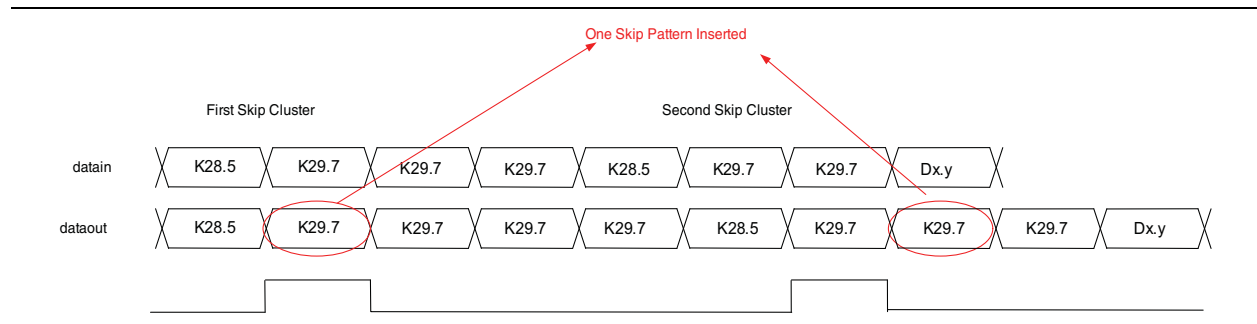


Figure 1-153 shows an example of rate match FIFO insertion in the case where one skip pattern is required to be inserted. In this example, the first skip cluster has a /K28.5/ control pattern followed by three /K29.7/ skip patterns. The second skip cluster has a /K28.5/ control pattern followed by two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


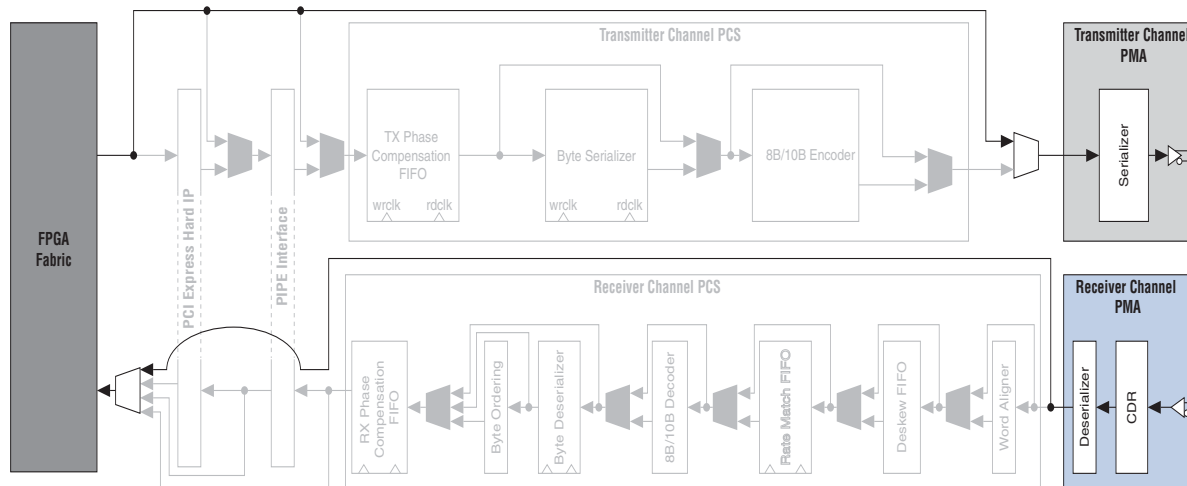
 In Basic (PMA Direct) mode, all the PCS blocks are bypassed; therefore, any PCS-type features (for example, phase compensation FIFOs, byte serializer, 8B/10B encoder/decoder, word aligner, deskew FIFO, rate match FIFO, byte deserializer, and byte ordering), must be implemented in the FPGA fabric. In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


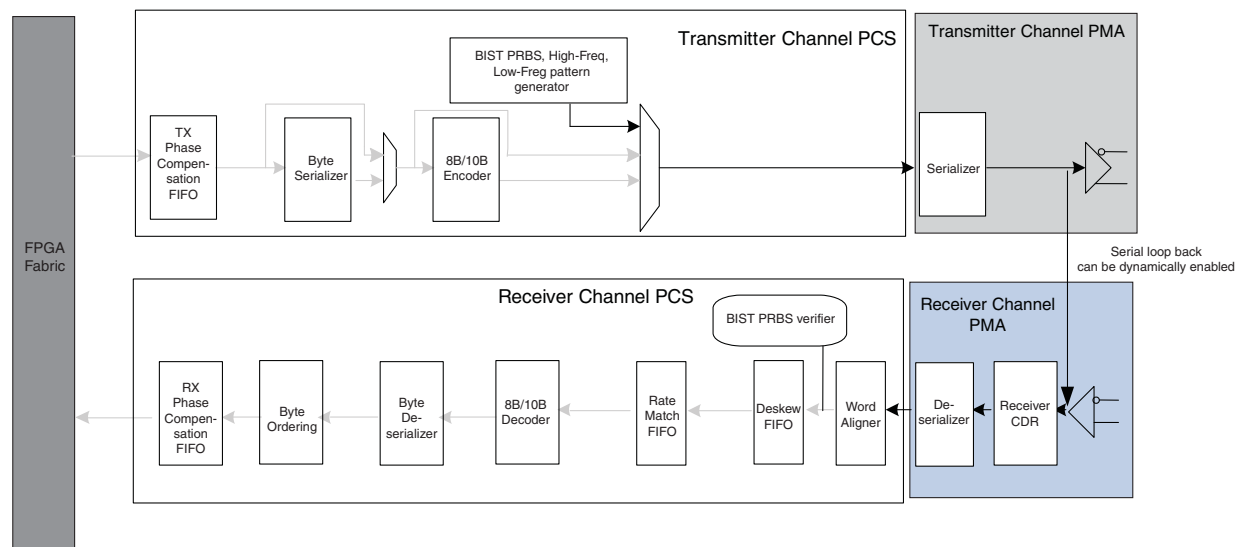
 When moving into or out of serial loopback, you must assert `rx_digitalreset` for a minimum of two parallel clock cycles.

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

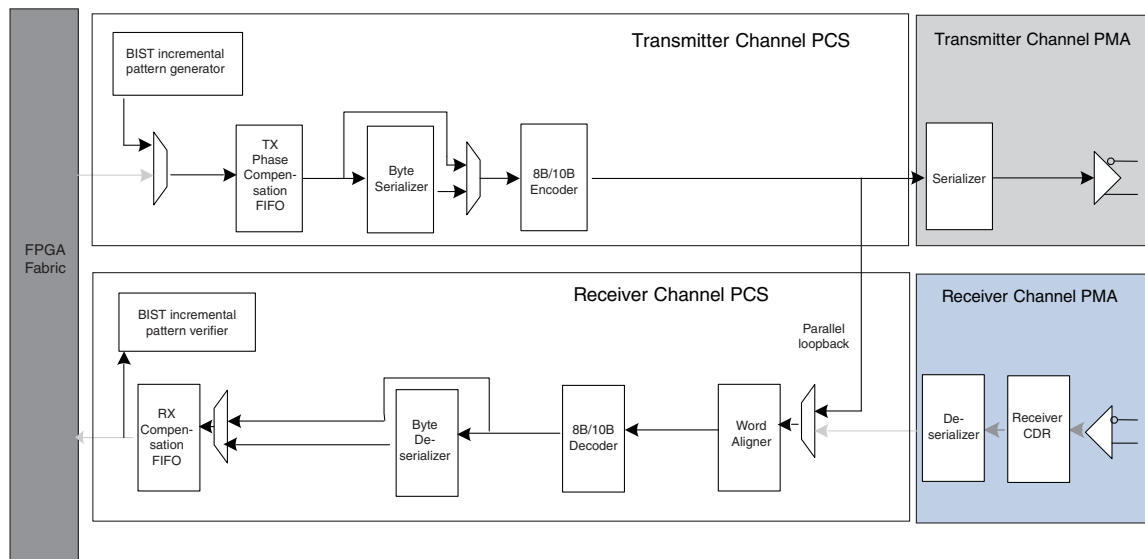


Table 1-69 lists the enabled PCS functional blocks for single-width and double-width mode. The last column in Table 1-69 lists the supported channel width setting for parallel loopback.

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.

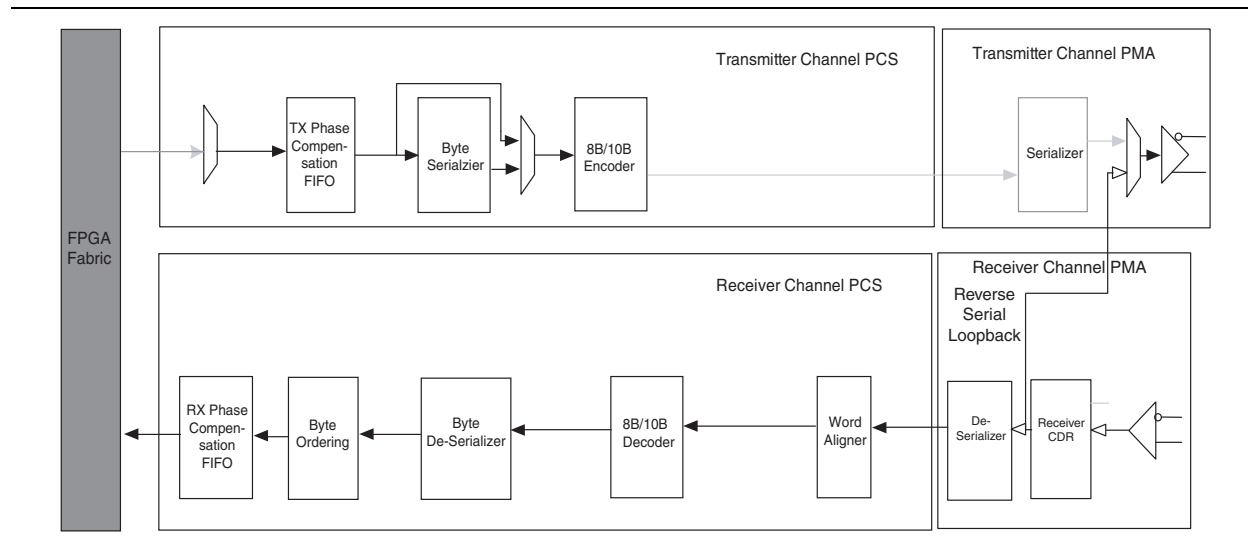
Reverse Serial Loopback

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.

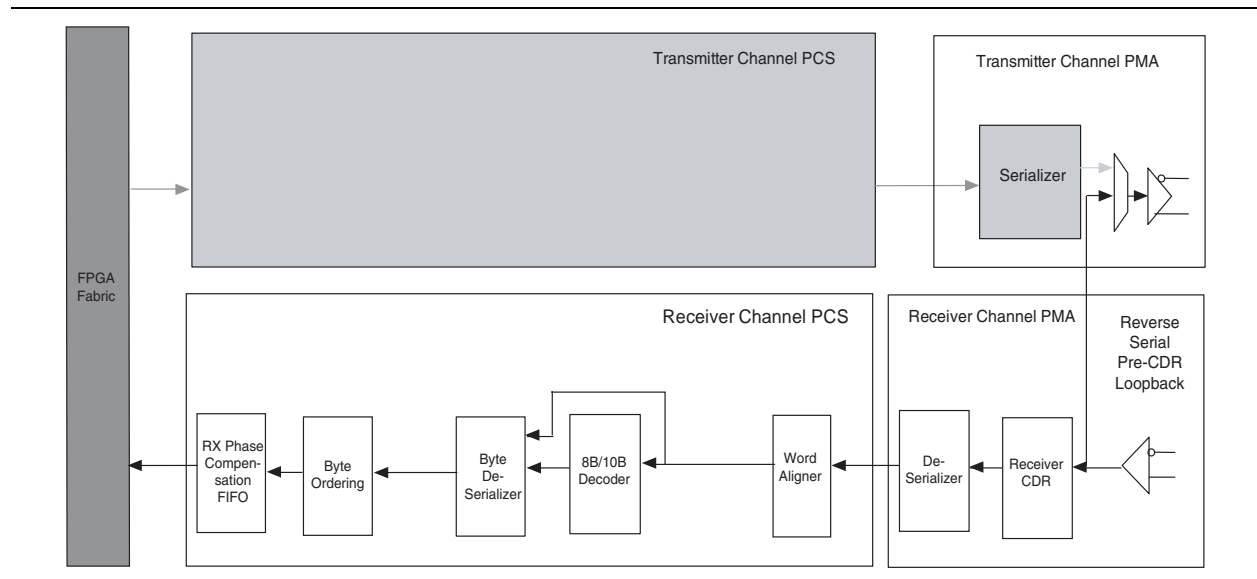
Figure 1-157. Reverse Serial Loopback Datapath (Grayed-Out Blocks are Not Active in this Mode)



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

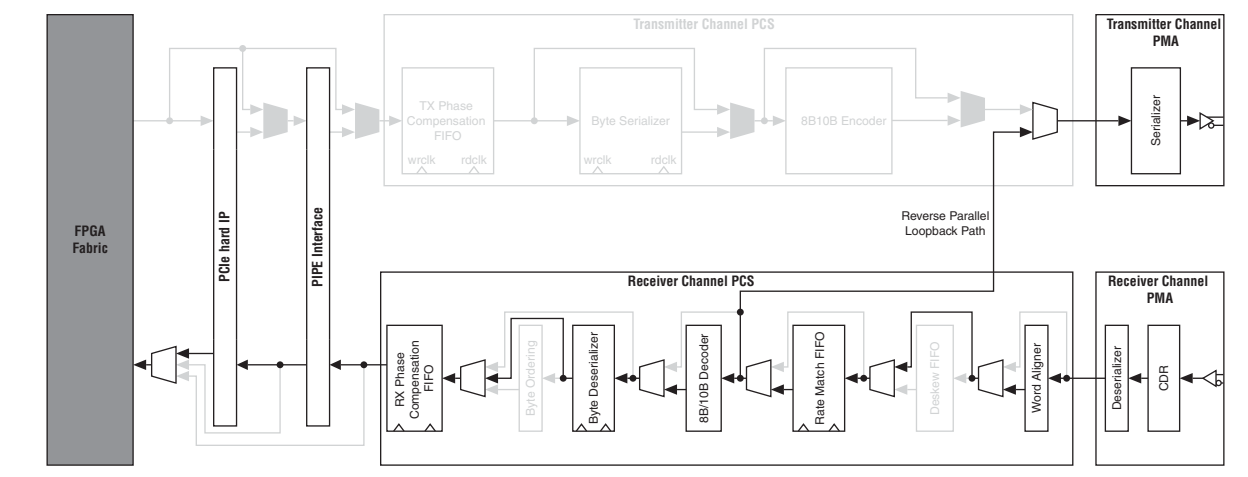


PCIe Reverse Parallel Loopback

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)

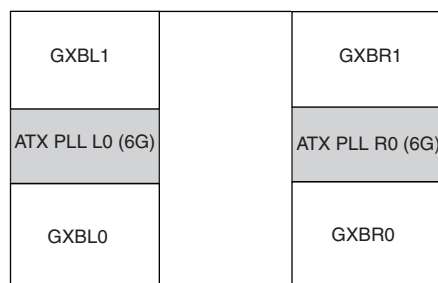


Figure 1-161 shows the location of the ATX PLL blocks in three transceiver block device families (for 230K and 530K devices and all other devices).

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.

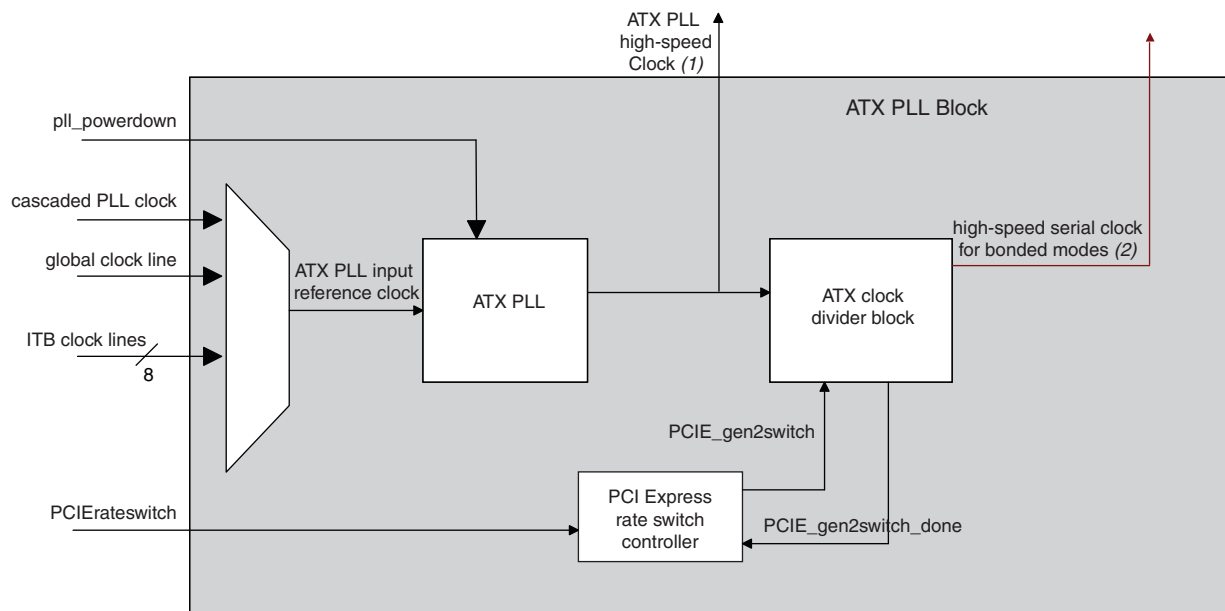
 For improved jitter performance, Altera strongly recommends using the REFCLK pins of the transceiver block located immediately below the 10G ATX PLL block to drive the input reference clock.

 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).

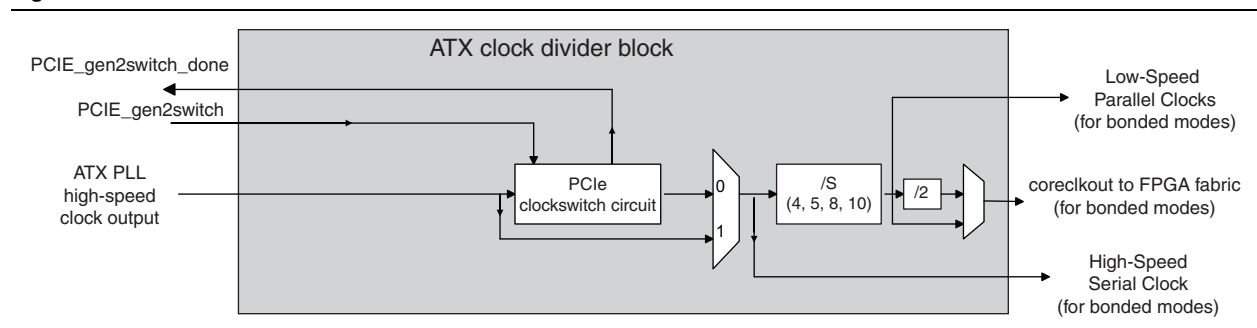
ATX Clock Divider

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)

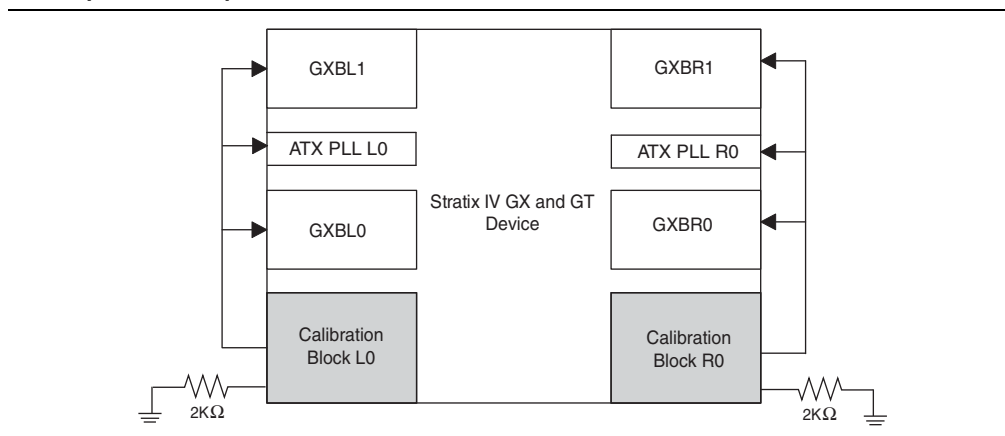


Figure 1-168 shows Stratix IV GX 230K and 530K devices that have three transceiver blocks each on the left and right side and one ATX PLL block on each side.

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

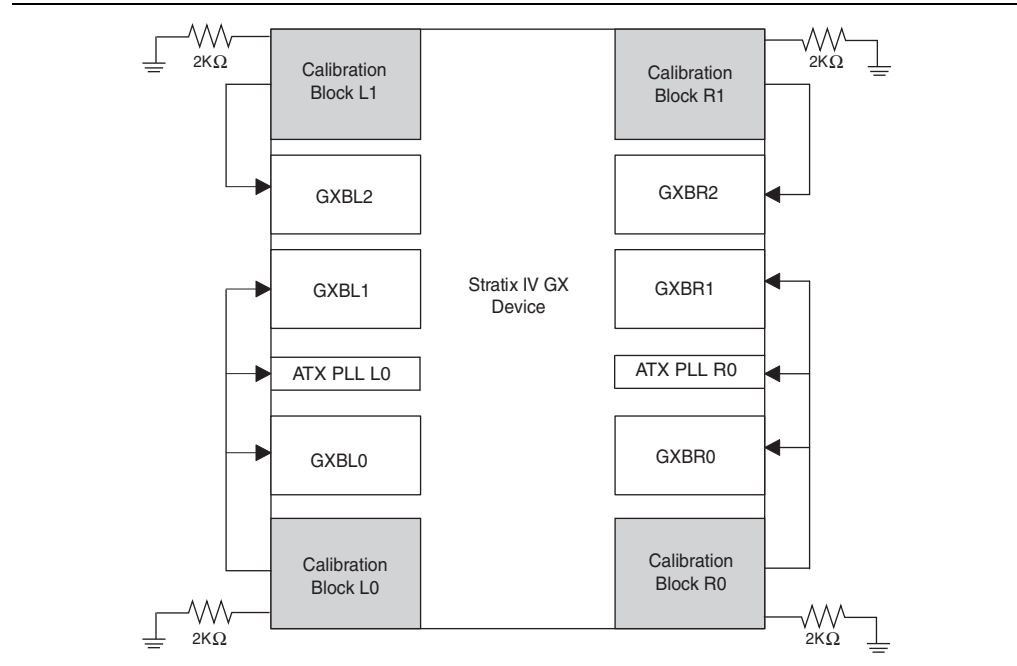


Figure 1-169 shows Stratix IV GX devices other than 230K and 530K that have three transceiver blocks each on the left and right side and two ATX PLL blocks on each side.

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

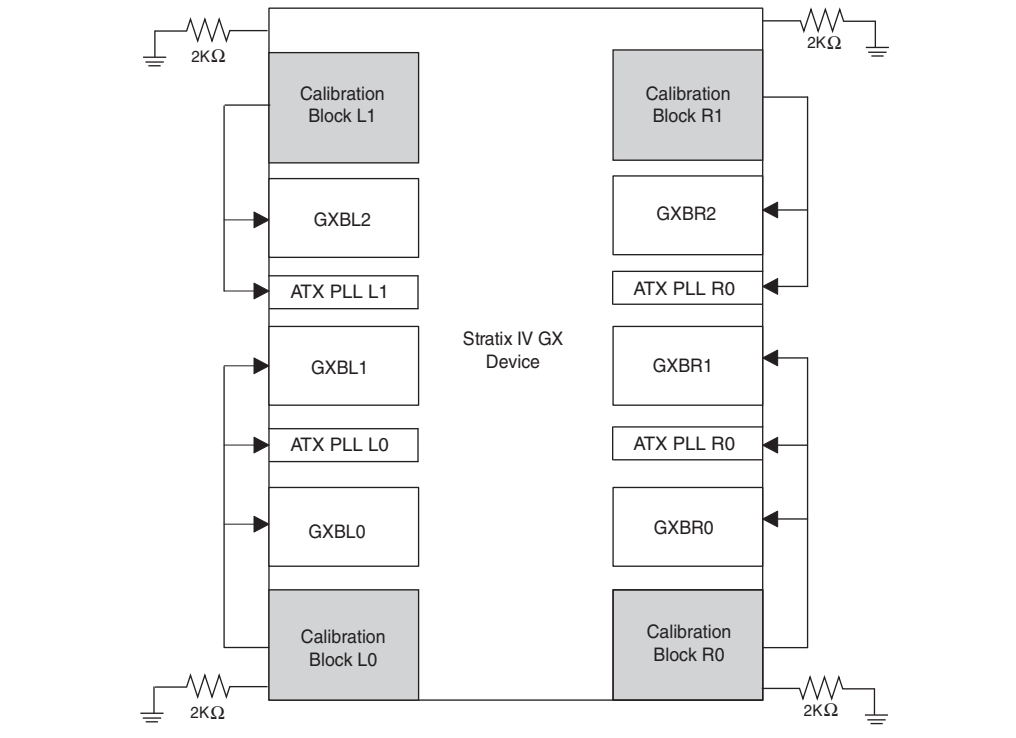


Figure 1-170 shows Stratix IV GX devices that have four transceiver blocks each on the left and right side and two ATX PLL blocks on each side.

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

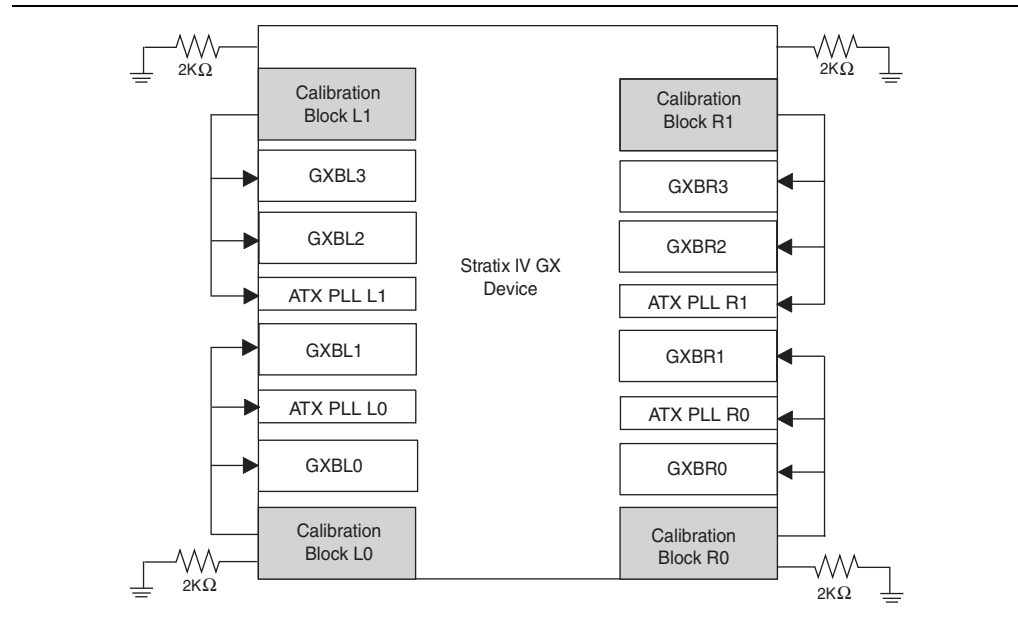
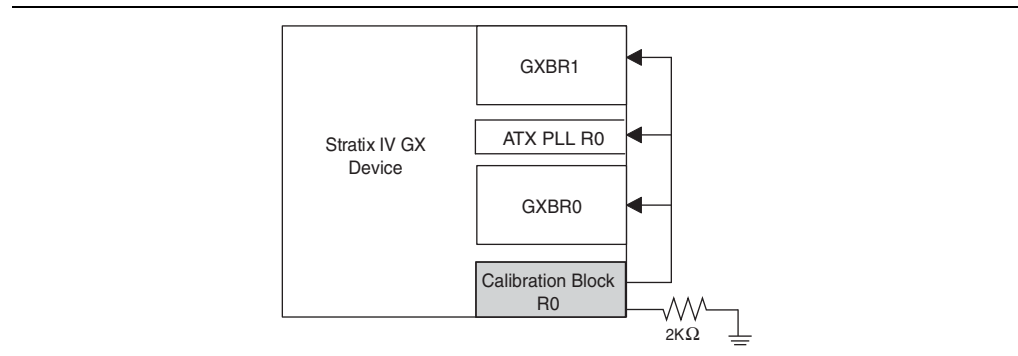


Figure 1-171 shows Stratix IV GX devices that have two transceiver blocks only on the right side of the device.

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.

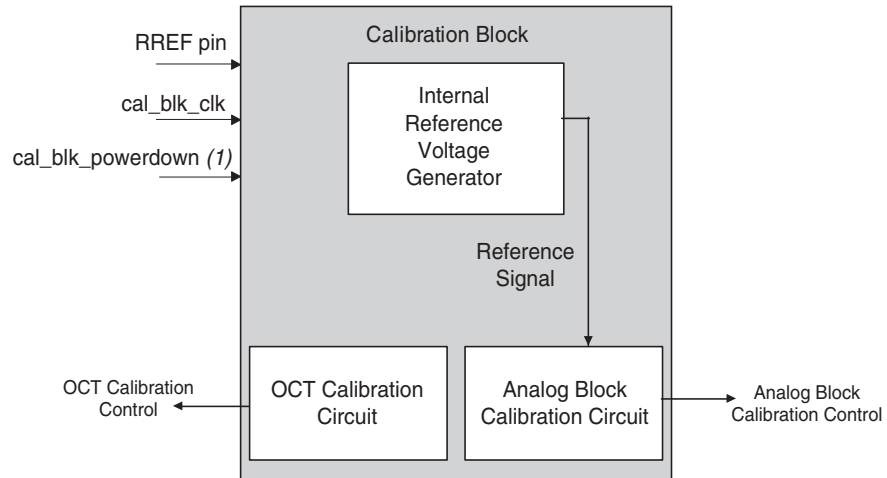
Input Signals to the Calibration Block

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.

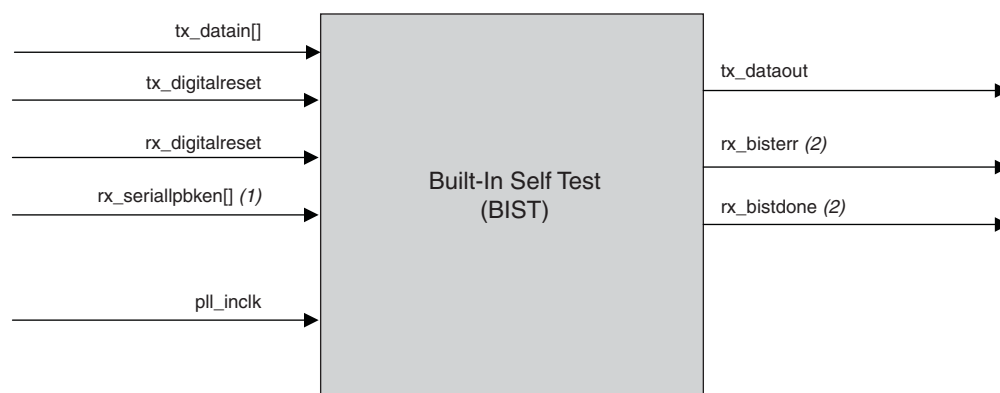
Built-In Self Test Modes

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:

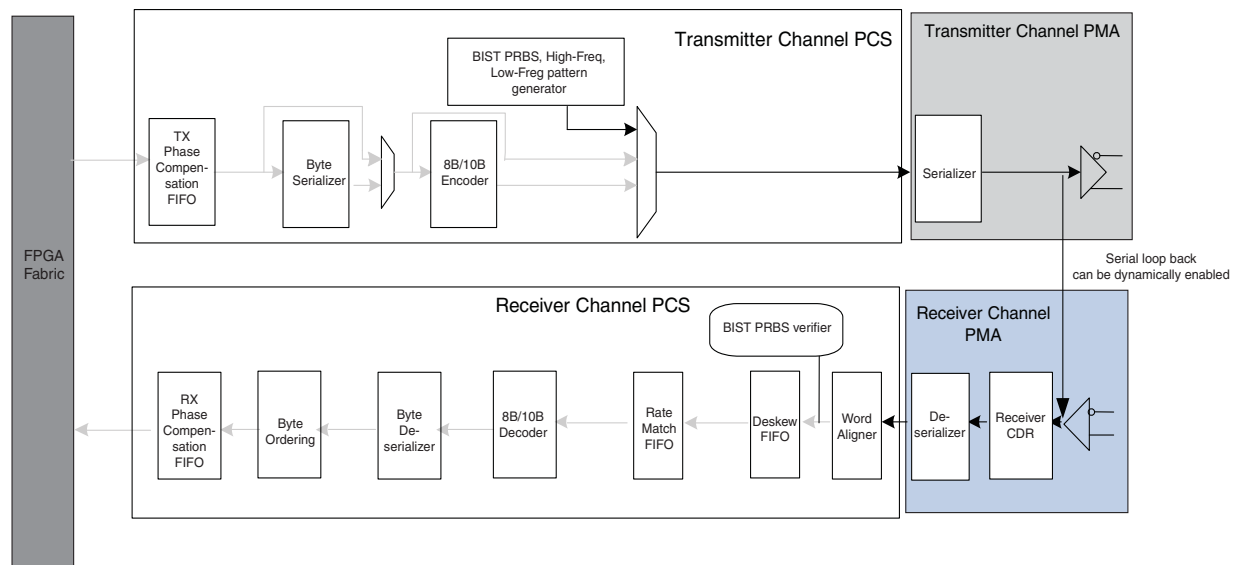
- BIST incremental data generator and verifier—This is only available in parallel loopback mode. For more information, refer to [“Serial Loopback” on page 1-190](#).
- High frequency and low frequency pattern generator—The high frequency patterns generate alternate ones and zeros and the low frequency 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 <p>Channel Width: 8—rx_errdetect = 1 16—rx_errdetect = 2 32—rx_errdetect = 4</p>	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. <p>Channel Width: 8—rx_disperr = 1 16—rx_disperr = 2 32—rx_disperr = 4</p>	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.	CMU PLL power down. <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. Note: Asserting the <code>pll_powerdown</code> signal does not power down the REFCLK buffers.	Transceiver block
coreclkout	Output	Clock signal	FPGA fabric-transceiver interface clock. <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	Dynamic reconfiguration clock. <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_forcedispliance	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— analog circuitry within the receiver PMA gets reset. Refer to <i>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 <i>Reset Control and Power Down</i>. 	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.