# 1. Transceiver Architecture in HardCopy IV Devices

HIV53001-1.1

This chapter provides an overview of the HardCopy® IV transceiver architecture, transceiver channels, available modes, and a description of transmitter and receiver channel datapaths.

HardCopy IV GX devices provide up to 24 full-duplex clock data recovery (CDR)-based transceivers with physical coding sublayer (PCS) and physical medium attachment (PMA), at serial data rates between 600 Mbps and 6.5 Gbps. Up to 12 additional full-duplex CDR-based transceivers with PMA, supporting serial data rates between 600 Mbps and 6.5 Gbps, are also provided.

The transceiver channels are designed to support the serial protocols listed in Table 1–1.

**Table 1–1. Serial Protocols Supported by the HardCopy 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 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 | 3.125 Gbps to 6.375 Gbps for Interlaken support |
| Serial Digital Interface (SDI) | HD-SDI at 1.485 Gbps and 1.4835 Gbps<br>3G-SDI at 2.97 Gbps and 2.967 Gbps |

The transceiver channels also support the following highly flexible functional modes to implement proprietary protocols:

- Basic
  - Basic single-width (600 Mbps to 3.75 Gbps)
  - Basic double-width (1 Gbps to 6.5 Gbps)

HardCopy IV GX devices have PCIe hard IP, PCS, and PMA blocks. For more information about the PCIe hard IP block, refer to the *PCI Express Compiler User Guide*.

This chapter contains the following sections:

## Transceiver Channel Locations

HardCopy IV GX 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.

Table 1–2 lists the total number of transceiver channels and transceiver block locations in each HardCopy IV GX device member.

**Table 1–2. Number of Transceiver Channels and Transceiver Block Locations in HardCopy IV GX Devices**

| Device Member | Total Number of Transceiver Channels | Transceiver Channel Location |
|---|---|---|
| HC4GX15LF780N<br>HC4GX15LAF780N | 8 | Eight transceiver channels located in two transceiver blocks, `GXBR0` and `GXBR1`, on the right side of the device. |
| HC4GX25LF1152N<br>HC4GX25LF780N | 16 | Eight transceiver channels located in two transceiver blocks, `GXBR0` and `GXBR1`, on the right side of the device.<br>Eight transceiver channels located in two transceiver blocks, `GXBL0` and `GXBL1`, on the left side of the device. |
| HC4GX25FF1152N<br>HC4GX35FF1152N | 24 | Eight regular transceiver channels and four CMU channels located in two transceiver blocks, `GXBR0` and `GXBR1`, on the right side of the device.<br>Eight regular transceiver channels and four CMU channels located in two transceiver blocks, `GXBL0` and `GXBL1`, on the left side of the device. |
| HC4GX35FF1517N | 36 | Twelve regular transceiver channels and six CMU channels located in three transceiver blocks, `GXBR0`, `GXBR1`, and `GXBR2` on the right side of the device.<br>Twelve regular transceiver channels and six CMU channels located in three transceiver blocks, `GXBL0`, `GXBL1`, and `GXBL2` on the left side of the device. |

Figure 1–1 through Figure 1–4 show the transceiver channel locations in each HardCopy IV GX device.

**Figure 1–1. Eight Transceiver Channels in HardCopy IV GX Devices**

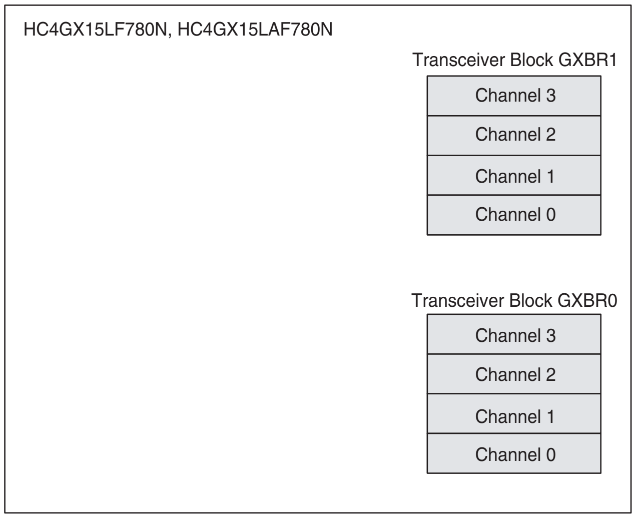


**Figure 1–2. Sixteen Transceiver Channels in HardCopy IV GX Devices**

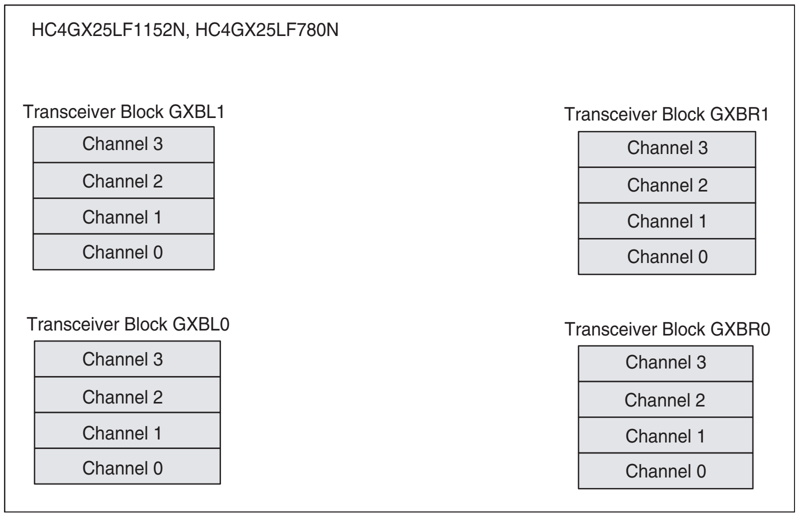


**Figure 1–3. Twenty-Four Transceiver Channels in HardCopy IV GX Devices**

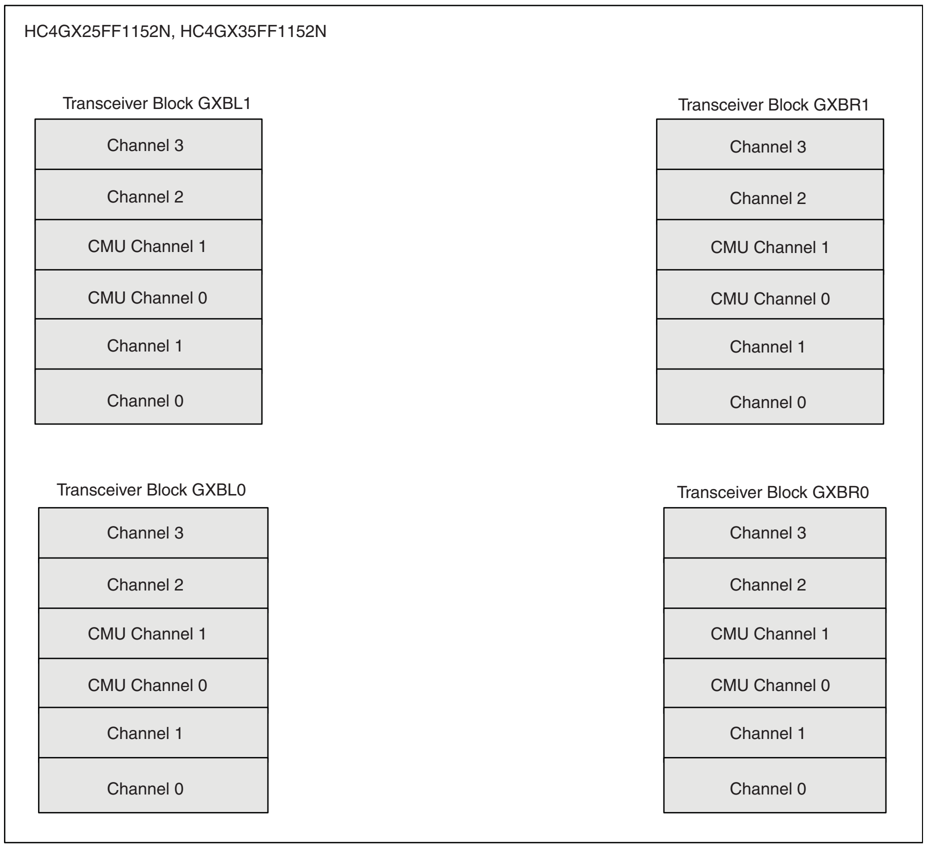
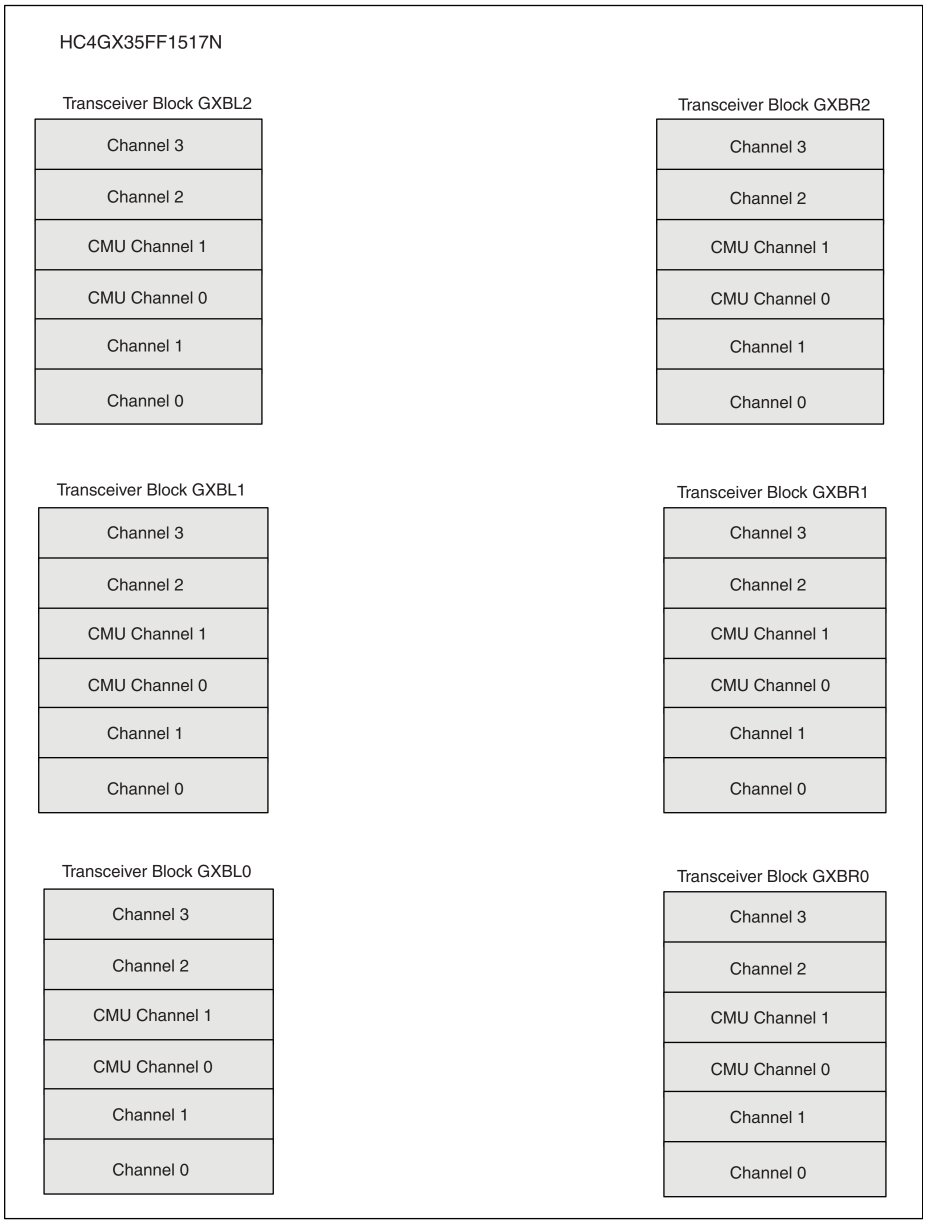


**Figure 1–4. Thirty-Six Transceiver Channels in HardCopy IV GX Devices**

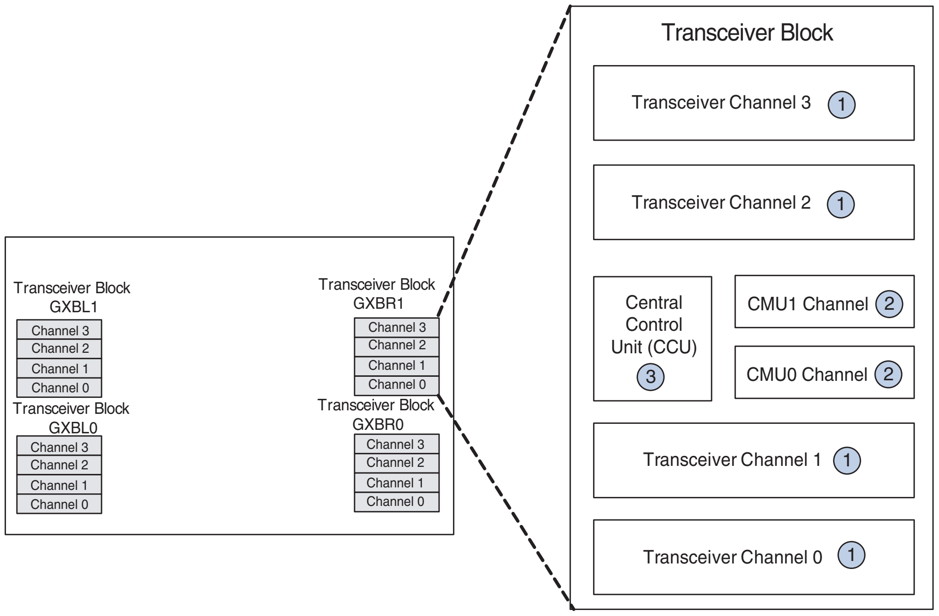


If you are migrating from an FPGA to a HardCopy IV ASIC, ensure that you select the HardCopy companion device. Likewise, select an FPGA companion device if you are planning to have an FPGA prototype.

## Transceiver Block Architecture

Figure 1–5 shows the transceiver block architecture of HardCopy IV GX devices.

**Figure 1–5. 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 6.5 Gbps in HardCopy IV GX devices. For more information, refer to "Transceiver Channel Architecture" on page 1–7.

2. Two clock multiplier unit (CMU) channels—`CMU0` and `CMU1`—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 1–45.

3. One 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.

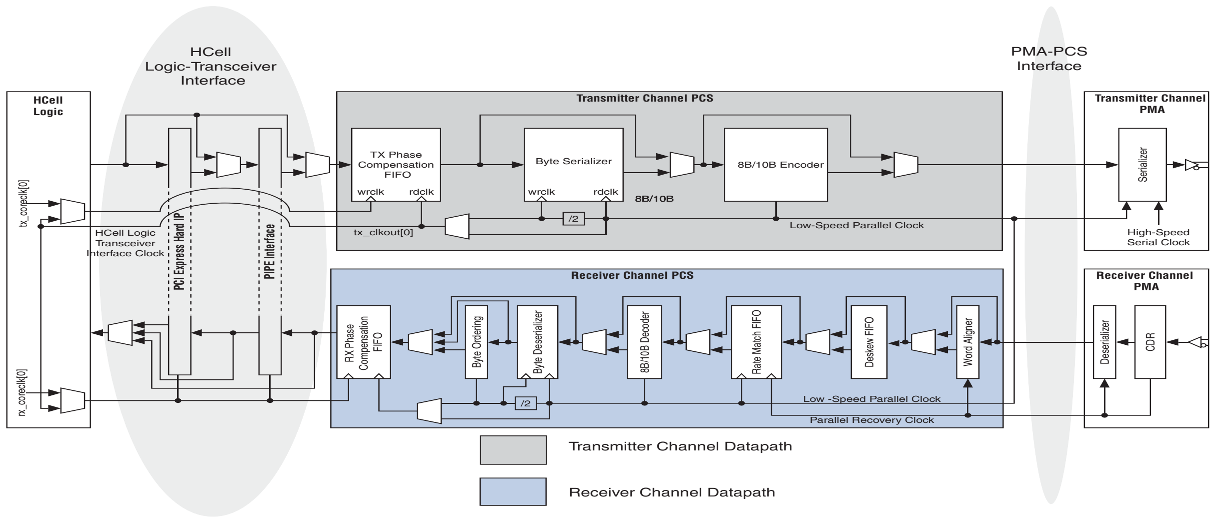
   - The shared control signal generation block provides control signals to the transceiver channels in bonded functional modes, such as XAUI, PCIe, and Basic ×4.

   - The PCIe rateswitch controller block controls the rateswitch circuit in the `CMU0` channel in ×4 configurations. In a PCIe ×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 1–76.

## Transceiver Channel Architecture

Figure 1–6 shows the HardCopy IV GX transceiver channel datapath.

**Figure 1–6. Transceiver Datapath for HardCopy IV GX Devices**



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 (the interface between the PCIe hard IP and the transceiver) or directly to the HCell Logic (HCell Logic-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 HCell Logic.

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

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

The transceiver channel datapath can be divided into the following two modes based on the HCell Logic-transceiver interface width (channel width) and the transceiver channel PMA-PCS width (serialization factor):

- Single-width mode
- Double-width mode

Table 1–3 lists the HCell Logic-transceiver interface widths (channel width) and transceiver PMA-PCS widths (serialization factor) allowed in single-width and double-width modes.

**Table 1–3. HCell Logic-Transceiver Interface Width and Transceiver PMA-PCS Widths**

| Name | Single-Width | Double-Width |
|---|---|---|
| PMA-PCS interface widths | 8/10 bit | 16/20 bit |
| HCell Logic-transceiver interface width | 8/10 bit<br>16/20 bit | 16/20 bit<br>32/40 bit |
| Supported functional modes | ■ PCIe Gen1 and Gen2<br>■ XAUI<br>■ GIGE<br>■ Serial RapidIO<br>■ SONET/SDH OC12 and OC48<br>■ SDI<br>■ Basic single-width | ■ (OIF) CEI PHY Interface<br>■ SONET/SDH OC96<br>■ Basic double-width |
| Data rate range in Basic functional mode | 0.6 Gbps to 3.75 Gbps | 1 Gbps to 6.5 Gbps |

## Transmitter Channel Datapath

The HardCopy IV transmitter channel datapath, shown in Figure 1–6 on page 1–7, consists of the following blocks:

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

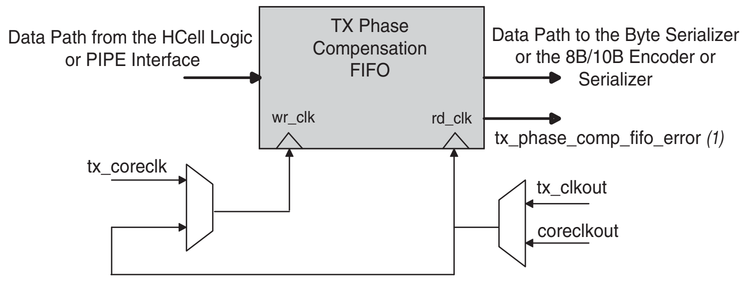
The HardCopy IV GX transceiver provides the **Enable low latency PCS mode** option in the ALTGX MegaWizard™ Plug-In Manager. If you select this option, the 8B/10B encoder in the datapath is disabled.

### TX Phase Compensation FIFO

The TX phase compensation FIFO interfaces the transmitter channel PCS and the HCell Logic PCIe interface. It compensates for the phase difference between the low-speed parallel clock and the HCell Logic interface clock. The TX phase compensation FIFO operates in low-latency and high-latency modes.

Figure 1–7 shows the datapath and clocking of the TX phase compensation FIFO.

**Figure 1–7. TX Phase Compensation FIFO**


Data Path from the HCell Logic or PIPE Interface
TX Phase Compensation FIFO
wr_clk
rd_clk
Data Path to the Byte Serializer or the 8B/10B Encoder or Serializer
tx_phase_comp_fifo_error (1)
tx_coreclk
tx_clkout
coreclkout

**Note to Figure 1–7:**

(1) The `tx_phase_comp_fifo_error` is optional. The signal is asserted high when an overflow or underflow occurs.

Table 1–4 lists the TX phase compensation FIFO modes.

**Table 1–4. TX Phase Compensation FIFO Modes**

| Modes | Description |
|---|---|
| Low-Latency | ■ FIFO is four words deep. ■ Latency through the FIFO is two to three HCell Logic parallel clock cycles (pending characterization). ■ Default setting for every mode. |
| High-Latency | ■ FIFO is eight words deep. ■ Latency through the FIFO is four to five HCell parallel cycles (pending characterization). |
| Non-Bonded Functional | The read port of the phase compensation FIFO is clocked by the low-speed parallel clock `tx_clkout`. The write clock is also fed by the `tx_clkout` port of the associated channel. |
| Bonded Functional | The write and read clocks of the FIFO are clocked by `coreclkout` provided by the `CMU0` clock divider block. You can also clock the write side using `tx_coreclk` provided from the HCell Logic by enabling the `tx_coreclk` port in the ALTGX MegaWizard Plug-In Manager. If you use this port, ensure that there is 0 parts-per-million (PPM) difference in frequency between the write and read side. The Quartus® II software requires that you provide a 0 PPM assignment in the Assignment Editor. |

### 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 HCell Logic.

### 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.

For more information about the TX Phase Compensation FIFO, refer to the *Stratix IV Transceiver Architecture* chapter in volume 2 of the *Stratix IV Device Handbook*.

### Byte Serializer

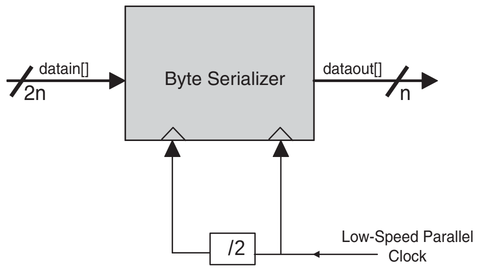
The byte serializer divides the input datapath by two. This allows you to run the transceiver channel at higher data rates while keeping the HCell Logic interface frequency within the maximum limit stated in the “Interface Frequency” section in the *DC and Switching Characteristics* chapter. In single-width mode, the byte serializer 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. The byte serializer is optional in configurations that do not exceed the HCell Logic-transceiver interface maximum frequency limit.

☞ The byte serializer is required in configurations that exceed the HCell Logic-transceiver interface maximum frequency limit.

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

Figure 1–8 shows the byte serializer datapath for both single- and double-width modes.

**Figure 1–8. Byte Serializer Datapath** ***(Note 1), (2)***



**Notes to Figure 1–8:**

(1) For the `datain[]` and `dataout[]` port widths, refer to Table 1–5.

(2) The `datain` signal is the input from the HCell Logic 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 HCell Logic, followed by `datain[19:10]`.

In double-width mode, assuming a channel width of 40, the byte serializer forwards `datain[19:0]` first, followed by `datain[39:20]`.

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

The output of the byte serializer is forwarded to the 8B/10B encoder if you select the **Encoder** option in the ALTGX MegaWizard Plug-In Manager. Otherwise, the byte serializer output is sent to the serializer in the PMA.

Table 1–5 lists the input and output data port widths.

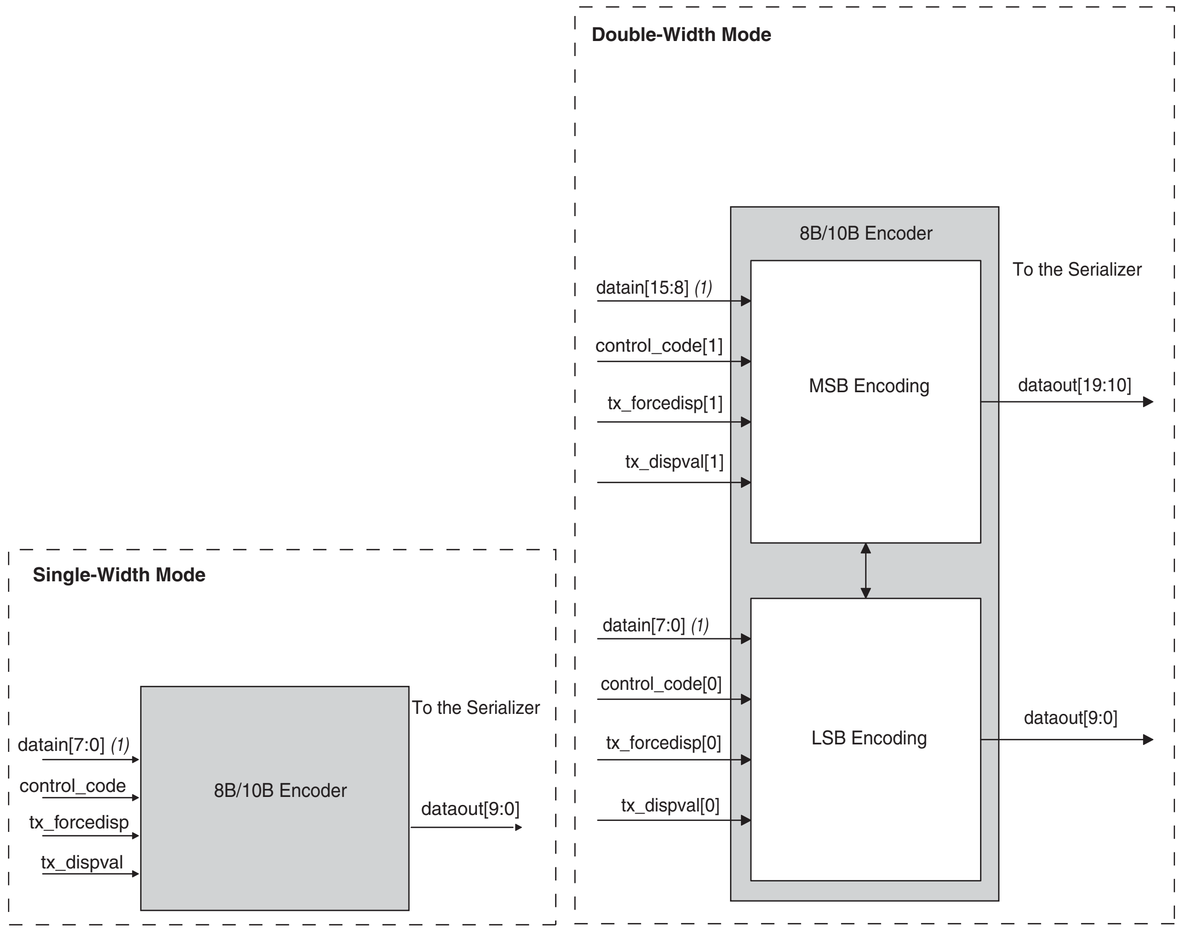
**Table 1–5. Input and Output Data Width of the Byte Serializer in Single and Double-Width Modes**

| 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 | 32 | 16 |
| | 40 | 20 |

### 8B/10B Encoder

The 8B/10B encoder generates a 10-bit code group (control word or data word) with proper disparity from an 8-bit data and a 1-bit control identifier. The 8B/10B encoder operates in two modes: single-width and double-width. Figure 1–9 shows the 8B/10B encoder in single- and double-width modes.

**Figure 1–9. 8B/10B Encoder in Single- and Double-Width Modes**



**Note to Figure 1–9:**

(1) Input data from the TX Phase-Compensation FIFO or Byte Serializer.

### Single-Width Mode

The left side of Figure 1–9 shows the 8B/10B encoder in single-width mode. 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, transmitting the LSB first.

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 1–14.

### Control Code Encoding

The ALTGX MegaWizard Plug-In Manager provides the `tx_ctrlenable` port to indicate whether the 8-bit data at the `tx_datain` port should be encoded as a control word (Kx.y). When `tx_ctrlenable` is low, the 8B/10B encoder block encodes the byte at the `tx_datain` port (the user-input port) as data (Dx.y). When `tx_ctrlenable` is high, the 8B/10B encoder encodes the input data as a Kx.y code group.

For more information about the Control Code Encoding scheme in single-width mode, refer to the *Control Code Encoding* section in the *Stratix IV Transceiver Architecture* chapter in volume 2 of the *Stratix IV Device Handbook*.

The IEEE 802.3 8B/10B encoder specification identifies only a set of 8-bit characters for which `tx_ctrlenable` must be asserted.

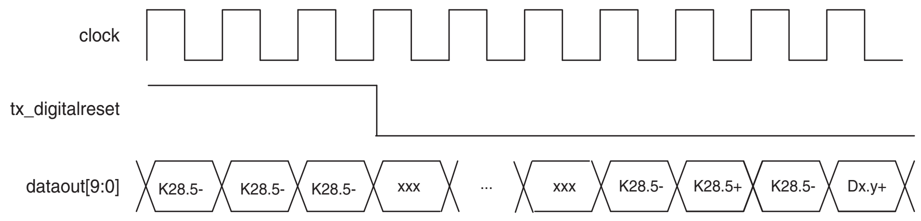
#### Reset Condition

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 HCell Logic 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 1–10 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 1–10. 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 1–9 on page 1–11. The LSByte of the input data is encoded and transmitted prior to the MSByte.

In double-width mode, the cascaded 8B/10B encoder generates two 10-bit code groups from two 8-bit data and their respective control code identifiers.

#### 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).

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.

Altera does not recommend sending invalid control words to the 8B/10B encoder.

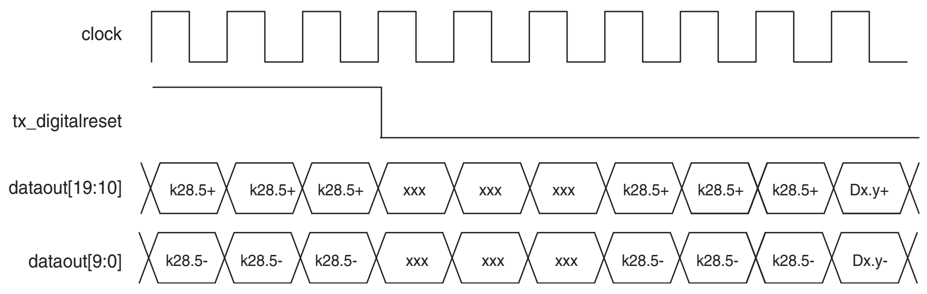
#### Reset Condition

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 1–11 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 1–11 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 1–11. 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 IEEE 802.3 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 1–6 lists the `tx_forcedisp` and `tx_dispval` port values.

**Table 1–6. tx_forcedisp and tx_dispval Port Values**

| tx_forcedisp | tx_dispval | Disparity Value |
|---|---|---|
| 0 | X | Current running disparity has no change |
| 1 | 0 | Encoded data has positive disparity |
| 1 | 1 | Encoded data has negative disparity |

Figure 1–12 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 1–12. 8B/10B Encoder Force Running Disparity Operation in Single-Width Mode**

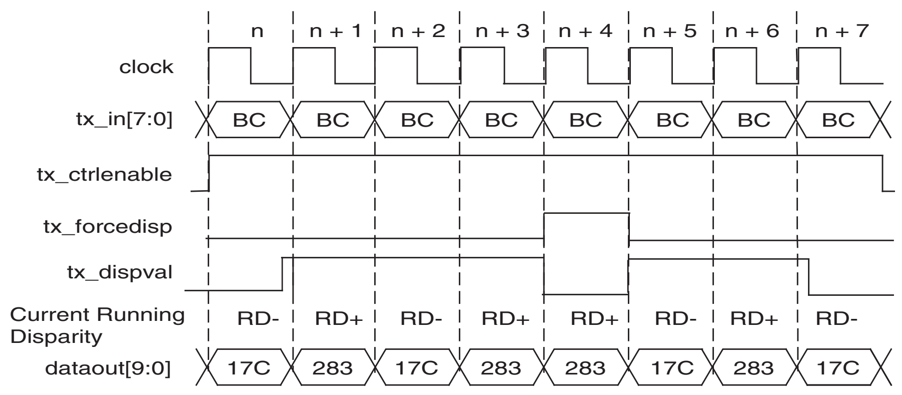
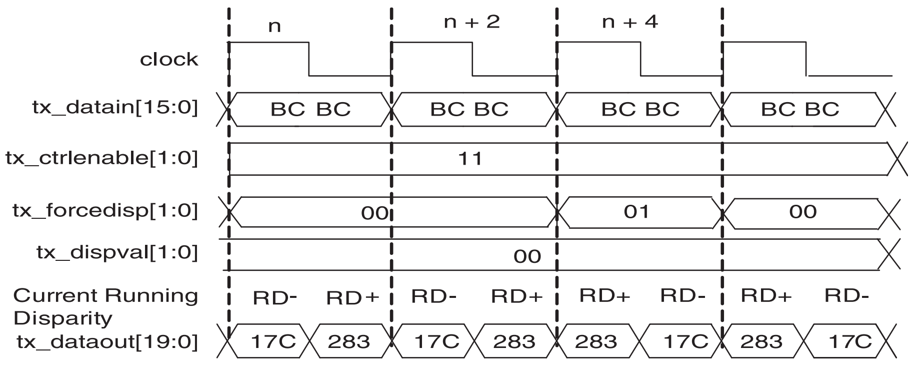


Figure 1–13 shows the current running disparity being altered in Basic double-width mode by forcing a positive disparity on a negative disparity K28.5. In this example, a series of K28.5 are continuously being sent. The stream alternates between a positive ending running disparity (RD+) K28.5 and a negative 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 1–13. 8B/10B Encoder Force Current Running Disparity in Double-Width Mode**

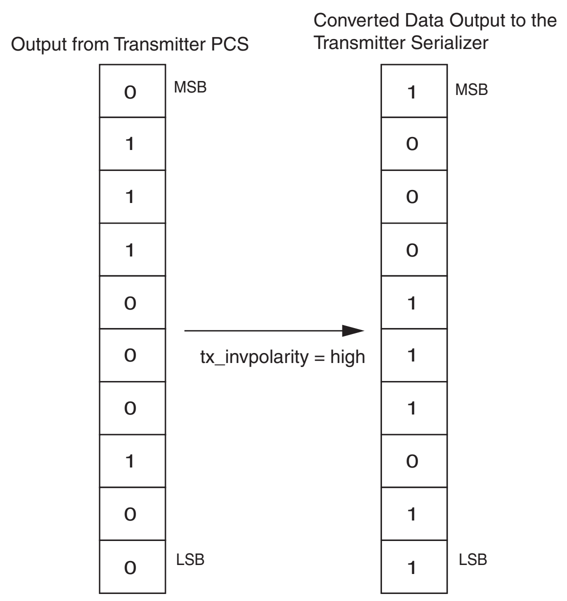


### Transmitter Polarity Inversion

The transmitter polarity inversion feature is provided to invert the polarity of each bit of the input data word to the serializer in the transmitter datapath for both single- and double-width modes. An optional `tx_invpolarity` port is available in all functional modes except (OIF) CEI PHY to dynamically enable the transmitter polarity inversion feature. A high value on the `tx_polarity` port inverts the polarity of every bit.

Figure 1–14 shows the transmitter polarity inversion feature in a 10-bit wide datapath configuration.

**Figure 1–14. Transmitter Polarity Inversion**


Output from Transmitter PCS
Converted Data Output to the Transmitter Serializer
0
1
1
1
0
0
0
1
0
0
MSB
LSB
tx_invpolarity = high
1
0
0
0
1
1
1
0
1
1
MSB
LSB

### Transmitter Bit Reversal

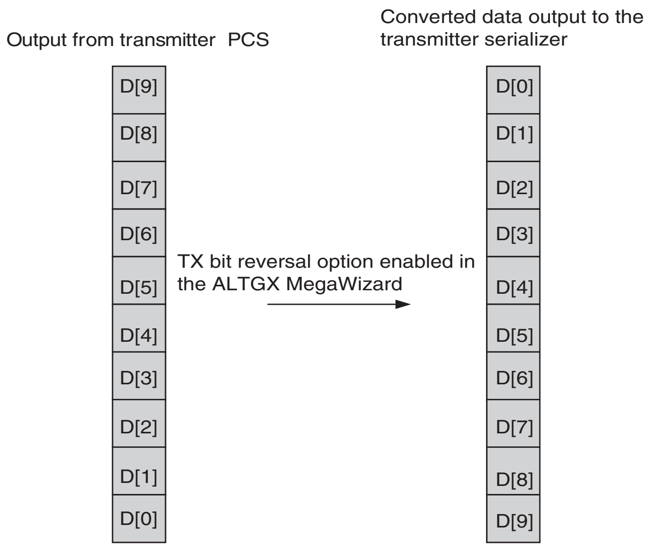
The LSB is transmitted first by default. The **Transmitter Bit Reversal** option in the ALTGX MegaWizard Plug-In Manager allows reversing of the transmit order of the bits (MSB first) before it is forwarded to the serializer. Table 1–7 lists the transmission bit order with and without the transmitter bit reversal enabled.

**Table 1–7. 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<br>For example:<br>■ 8-bit—`D[7:0]` rewired to `D[0:7]`<br>■ 10-bit— `D[9:0]` rewired to `D[0:9]` | MSB to LSB<br>For example:<br>■ 16-bit—`D[15:0]` rewired to `D[0:15]`<br>■ 20-bit—`D[19:0]` rewired to `D[0:19]` |

Figure 1–15 shows the transmitter bit reversal feature for a 10-bit wide datapath configuration.

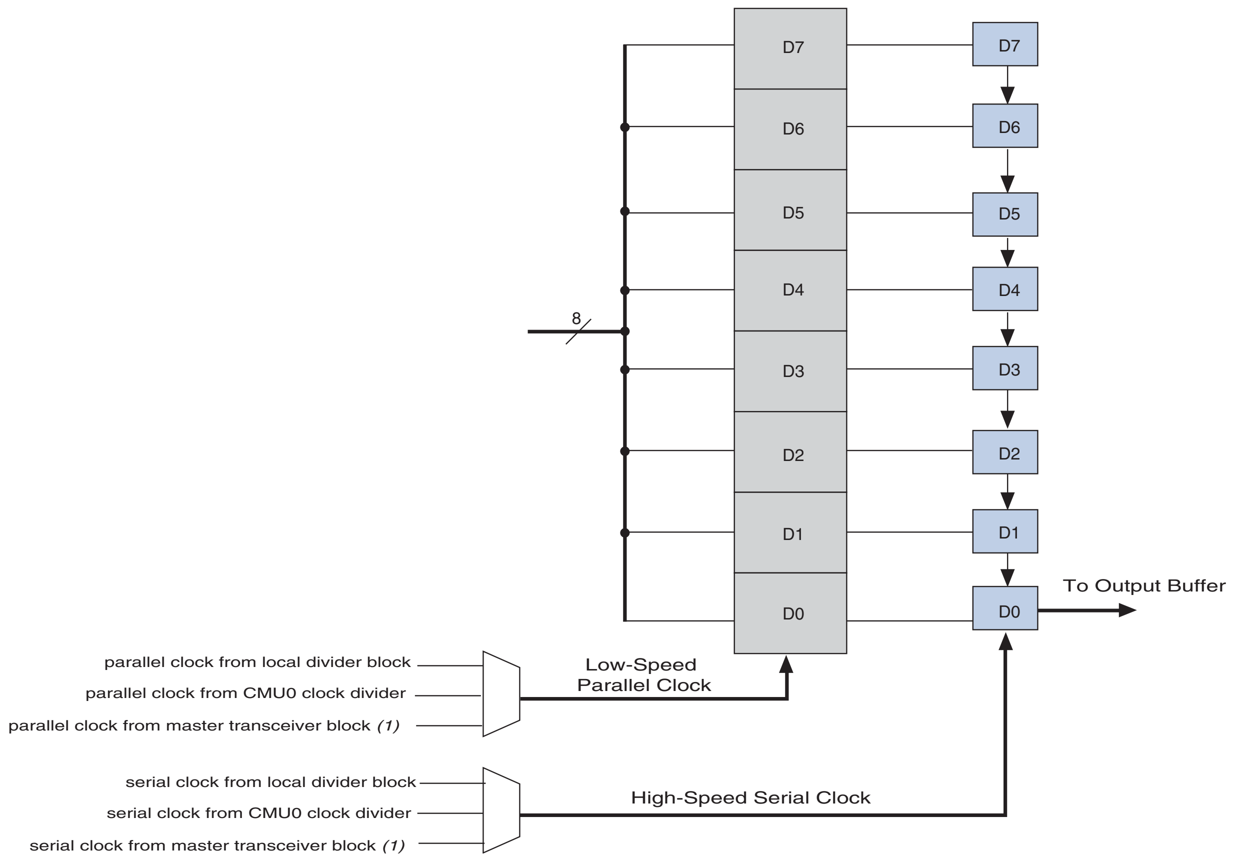
**Figure 1–15. Transmitter Bit Reversal Operation**


Output from transmitter PCS
Converted data output to the transmitter serializer
D[9]
D[8]
D[7]
D[6]
D[5]
D[4]
D[3]
D[2]
D[1]
D[0]
TX bit reversal option enabled in the ALTGX MegaWizard
D[0]
D[1]
D[2]
D[3]
D[4]
D[5]
D[6]
D[7]
D[8]
D[9]

## 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 1–16. The serializer block sends out the LSB of the input data.

**Figure 1–16. Serializer Block in 8-Bit PCS-PMA Interface**


D7
D6
D5
D4
D3
D2
D1
D0
8
To Output Buffer
parallel clock from local divider block
parallel clock from CMU0 clock divider
parallel clock from master transceiver block (1)
Low-Speed Parallel Clock
serial clock from local divider block
serial clock from CMU0 clock divider
serial clock from master transceiver block (1)
High-Speed Serial Clock

**Note to Figure 1–16:**

(1) The CMU0 clock divider of the master transceiver block provides the clocks. It is used only in bonded modes (for example, Basic ×8, PCIe ×8 mode).

### Transmitter Output Buffer

The HardCopy IV GX transmitter buffers support the following:

- On-chip termination (OCT) with selectable settings
- Reconfigurable output differential voltage (VOD) with selectable settings—using PMA Control Reconfiguration and Channel Reconfiguration modes
- Reconfigurable pre-emphasis with selectable settings—using PMA Control Reconfiguration and Channel Reconfiguration modes
- Receiver-detect capability to support PCIe functional mode
- Transmission of PCIe Electrical Idle (or individual transmitter tri-state)

The transmitter buffer power supply only provides voltage to the transmitter output buffers in the transceiver channels. The transmitter output buffer, shown in Figure 1–17, 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, which you can set in the ALTGX MegaWizard Plug-In Manager.

**Figure 1–17. Transmitter Output Buffer**

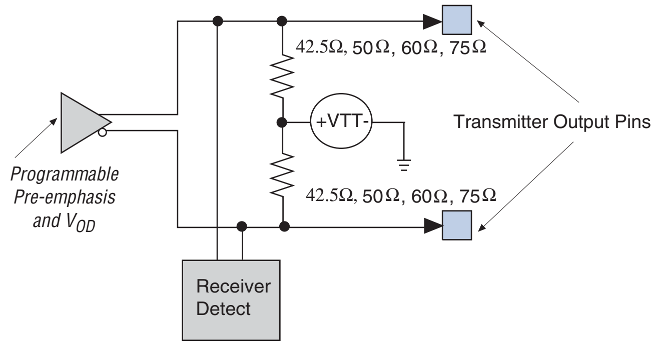


Table 1–8 lists the supported settings of the transmitter buffers in the HardCopy IV GX devices.

**Table 1–8. Supported Settings for the HardCopy IV GX Transmitter Buffer**

| Parameter | Setting |
|---|---|
| Data rate | 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 |

### Transmitter Termination

HardCopy IV GX 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 1–116), which compensates for temperature, voltage, and process changes. The HardCopy IV GX 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 "Output Differential Voltage" on page 1–20.

For a particular HardCopy IV design, run simulations with the transmitter buffer and transmission line to determine the optimal termination. You can also determine the optimal setting on a FPGA prototype device and then migrate it to the HardCopy IV device.

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 1–9 to the transmitter serial output pin.

**Table 1–9. HardCopy IV GX 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 |

#### Output Differential Voltage

HardCopy IV GX devices allow you to customize the differential output voltage to handle different trace lengths, various backplanes, and receiver requirements. 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**.

$V_{OD}$ settings are supported using both PMA Control Reconfiguration and Channel Reconfiguration modes.

For more information about HardCopy IV GX $V_{OD}$ values, refer to the *DC and Switching Characteristics of HardCopy IV Devices* chapter.

#### Pre-Emphasis

The 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.

HardCopy IV GX 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. You can change these settings using the dynamic reconfiguration controller.

Pre-emphasis settings are supported using both PMA Control Reconfiguration and Channel Reconfiguration modes.

For more information about pre-emphasis values, refer to the *DC and Switching Characteristics of HardCopy IV Devices* chapter.

#### Link Coupling for HardCopy IV GX Devices

A high-speed serial link can be AC-coupled or DC-coupled, depending on the serial protocol being implemented.

For more information about the AC-coupled and DC-coupled link setup, refer to the *Stratix IV Transceiver Architecture* chapter in *volume 2 of the Stratix IV Device Handbook*.

The following protocols supported by HardCopy IV GX devices mandate AC-coupled links:

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

The HardCopy IV GX transmitter can be DC-coupled to a HardCopy IV GX receiver for the entire operating data rate range of HardCopy IV GX, from 600 Mbps to 6.5 Gbps.

#### PCIe Receiver Detect

The HardCopy IV GX 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 and the *Stratix IV Transceiver Architecture* chapter in volume 2 of the *Stratix IV Device Handbook*.

#### PCIe Electrical Idle

The HardCopy IV GX 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.

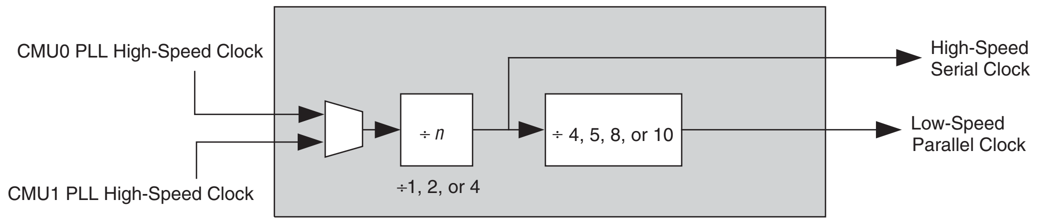
For more information about using the `tx_forceelecidle` signal under different power states, refer to the PCIe specification 2.0.

### Transmitter Local Clock Divider Block

Each transmitter channel contains a local clock divider block. It receives the high-speed clock from the `CMU0` phase-locked loop (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 HCell Logic (`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 1–18 shows the transmitter local clock divider block.

**Figure 1–18. Transmitter Local Clock Divider Block**



## Receiver Channel Datapath

This section describes the HardCopy IV GX 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 HCell Logic-transceiver interface. Figure 1–6 on page 1–7 shows the receiver channel datapath in HardCopy IV GX devices. The HardCopy IV GX receiver channel datapath consists of the same blocks as the Stratix IV GX receiver channel datapath.

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.

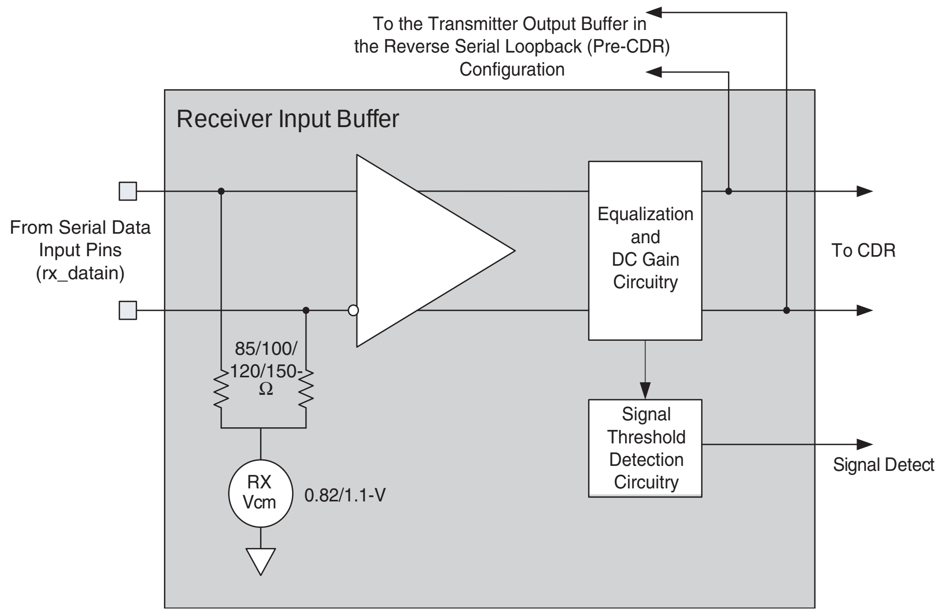
### Receiver Input Buffer

The receiver input buffer supports the following:

- Common mode voltage (Rx VCM) with selectable settings
- Equalization with reconfigurable settings (in PMA Reconfiguration mode only)
- DC gain with reconfigurable settings (using PMA Control Reconfiguration and Channel Reconfiguration modes)
- OCT with selectable settings

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 1–19 shows the receiver input buffer.

**Figure 1–19. Receiver Input Buffer**


To the Transmitter Output Buffer in the Reverse Serial Loopback (Pre-CDR) Configuration
Receiver Input Buffer
From Serial Data Input Pins (rx_datain)
Equalization and DC Gain Circuitry
To CDR
85/100/120/150-Ω
Signal Threshold Detection Circuitry
Signal Detect
RX Vcm
0.82/1.1-V

Equalization settings are only supported in PMA Reconfiguration mode. DC gain settings are supported using both PMA Control Reconfiguration and Channel Reconfiguration modes.

Table 1–10 lists the electrical features supported by the HardCopy IV GX receiver input buffer.

**Table 1–10. Electrical Features Supported by the Receiver Input Buffer for HardCopy IV GX Devices *(Note 1)***

| Data Rate (Gbps) | I/O Standard | Differential OCT with Calibration (Ω) | $V_{CM}$ (V) | Coupling | Programmable DC Gain (dB) |
|---|---|---|---|---|---|
| HardCopy IV GX 0.6 to 6.5 | 1.4 V PCML | 85, 100, 120, 150 | 0.82 | AC, DC | up to 16 |
| | 1.5 V PCML | 85, 100, 120, 150 | 0.82 | AC, DC | up to 16 |
| | 2.5 V PCML | 85, 100, 120, 150 | 0.82 | AC | up to 16 |
| | LVPECL | 85, 100, 120, 150 | 0.82 | AC | up to 16 |
| | LVDS | 85, 100, 120, 150 | 1.1 | AC, DC | up to 16 |

**Note to Table 1–10:**

(1) Programmable equalization settings are 0 to 16 dB for HardCopy IV GX devices.

The HardCopy IV GX receiver buffers support features similar to the Stratix IV GX buffers, such as the following:

- Programmable differential OCT
- Programmable $V_{CM}$
- AC and DC coupling
- Programmable equalization and DC gain
- Signal threshold detection circuitry

For more information about the receiver buffer features, refer to the *Stratix IV Transceiver Architecture* chapter in volume 2 of the *Stratix IV Device Handbook*.

### Adaptive Equalization (AEQ) Feature

HardCopy IV GX receivers offer an Adaptive Equalization feature that automatically compensates for losses on the receiver channels. The AEQ feature enables the HardCopy IV GX device to tune the receiver equalization settings based on the frequency content of the incoming signal and compare it with internally generated reference signals.

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.

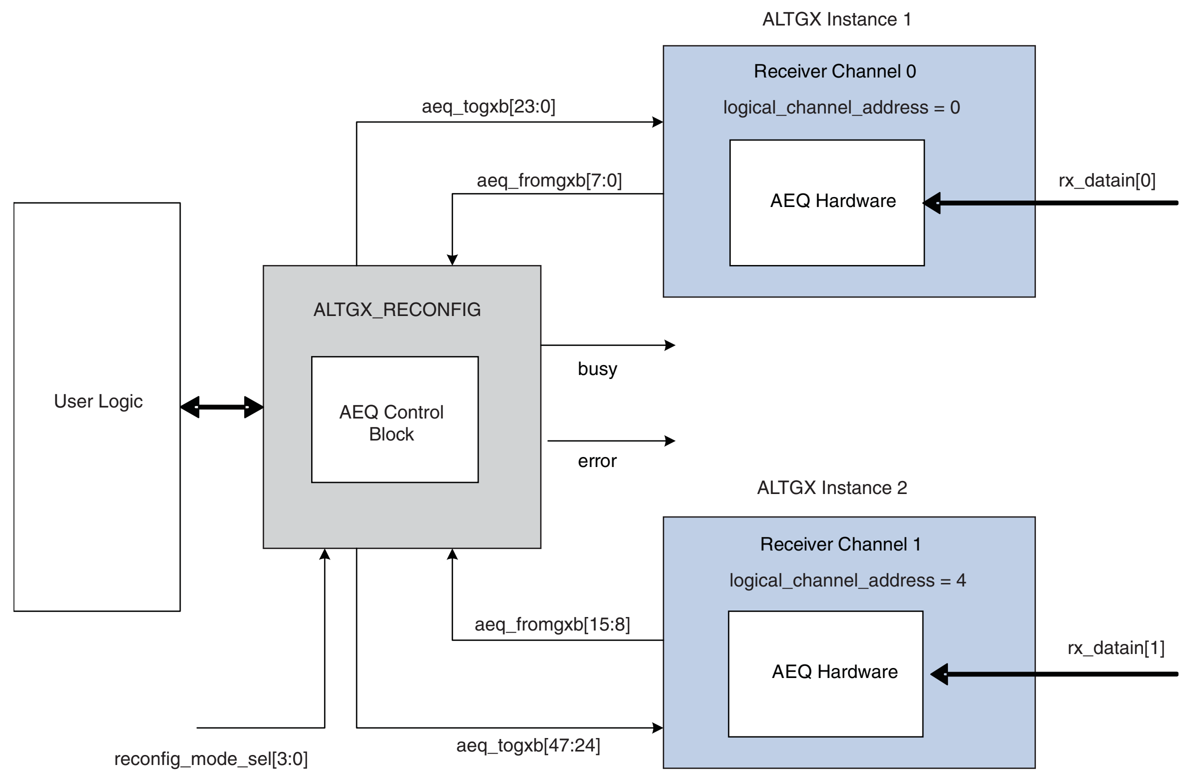
To enable the AEQ feature, in the ALTGX and ALTGX_RECONFIG MegaWizard Plug-In Managers, select the **Enable adaptive equalizer control** option.

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 1–20 shows the receiver channel data path with the AEQ feature.

**Figure 1–20. Receiver Channel Data Path Showing AEQ**



### Modes of Operation of the AEQ Feature

Depending on the value you set for `reconfig_mode_sel[3:0]`, the AEQ feature has two modes of operation:

- One-time mode—The AEQ finds a stable setting of the receiver equalizer and locks that value. Once locked, the equalizer values are no longer changed. This mode is available in one channel or all channels of the receiver. The `reconfig_mode_sel[3:0]` = 1001 in this mode. This mode is under device characterization.
- Powerdown mode—In this mode, the AEQ of the specific channel is placed in standby mode. This mode is available in one channel or all channels of the receiver. The `reconfig_mode_sel[3:0]` = 1010 in this mode.

The AEQ comes out of standby mode as soon as the value on `reconfig_mode_sel` is changed to another mode of the AEQ control modes. After the AEQ goes into powerdown mode and comes out, it does not remember the converged equalization value. By design, the AEQ starts at the maximum equalization value after powering up again.

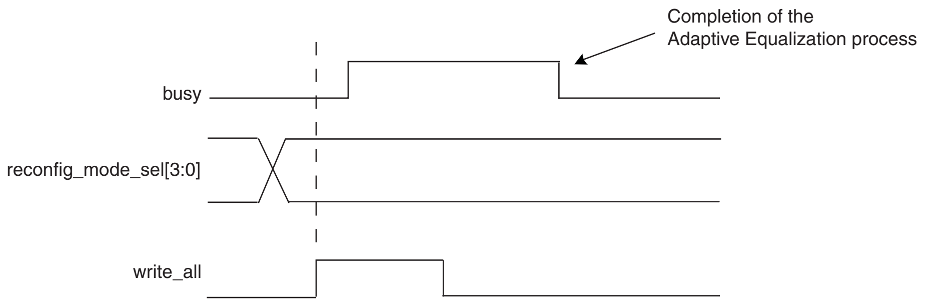
To control the AEQ hardware in any of the above modes, follow these steps:

1. Watch for `busy` to be low, then write the appropriate value on `reconfig_mode_sel[3:0]`.
2. Assert `write_all`, then watch for `busy` to be asserted.

3. De-assert `write_all` and watch for the de-assertion of `busy`. De-assertion of `busy` indicates that AEQ has ended.

Figure 1–21 shows a waveform of the AEQ process.

**Figure 1–21. AEQ Process Waveform**



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

## EyeQ

The EyeQ hardware is available in HardCopy IV GX 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.

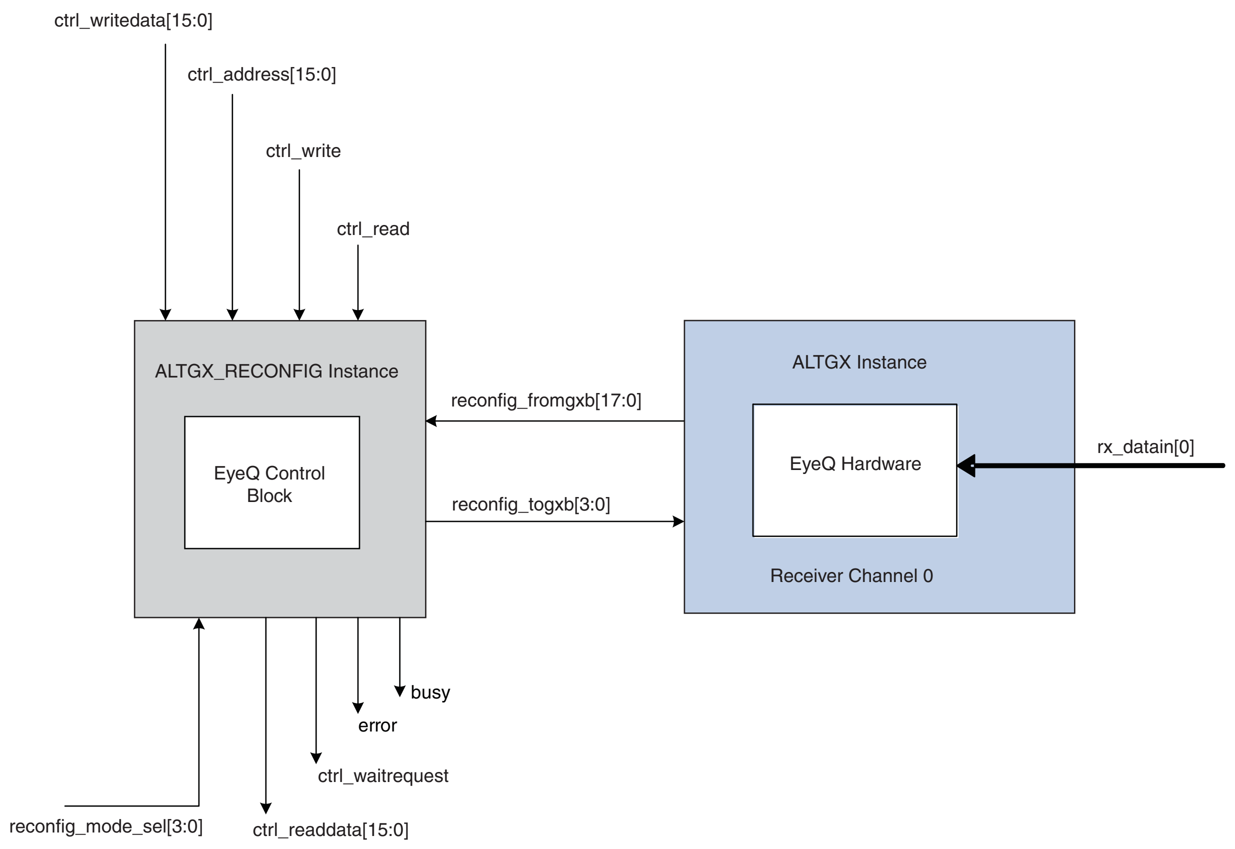
The EyeQ hardware is available for both regular transceiver channels and CMU channels.

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 1–22 shows the EyeQ feature within a receiver channel datapath.

You must implement logic to check the BER. This includes a pattern generator and checker.

Figure 1–22 shows the receiver channel data path using the EyeQ feature.

**Figure 1–22. Receiver Channel Data Path showing the EyeQ Feature**

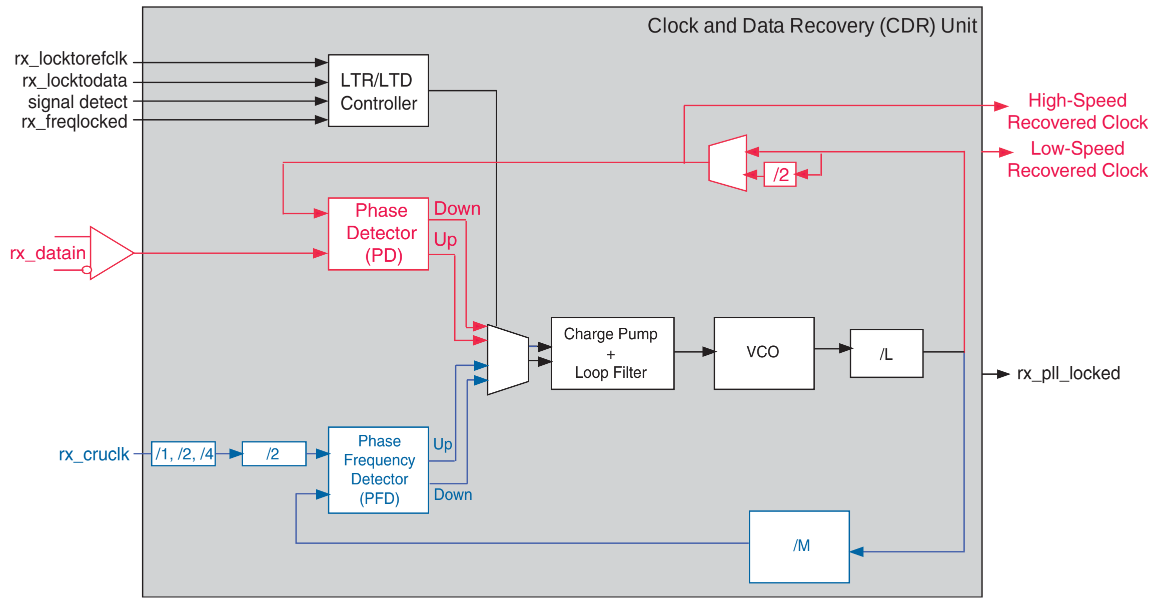


For more information about using the EyeQ feature, refer to the *Dynamic Reconfiguration in HardCopy IV GX Devices* chapter.

### Clock and Data Recovery Unit

Each HardCopy IV GX 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 1–23 shows the CDR block diagram.

**Figure 1–23. Clock and Data Recovery Unit** ***(Note 1)***



**Note to Figure 1–23:**

(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 lock-to-reference (LTR) mode or lock-to-data (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_cruclk`. The phase frequency detector (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 1–23 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

Table 1–11 lists the CDR specifications in LTR mode.

**Table 1–11. CDR Specifications in Lock-To-Reference Mode**

| Parameter | Value |
|---|---|
| Input Reference Clock Frequency | 50 MHz to 672 MHz |
| PFD Input Frequency | 50 MHz to 325 MHz |
| /M Divider | 4, 5, 8, 10, 16, 20, 25 |
| /L Divider | 1, 2, 4, 8 |

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.

For more information about the CDR specifications in LTR mode, refer to the *DC and Switching Characteristics of HardCopy IV Devices*.

#### 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 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 1–23 on page 1–28 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* chapter in volume 2 of the *Stratix IV Device Handbook*.

#### 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 1–76.

☞ 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 to either automatic lock mode or manual lock mode using the optional input ports `rx_locktorefclk` or `rx_locktodata`.

Table 1–12 lists the relationship between these optional input ports and the LTR/LTD controller lock mode.

**Table 1–12. Optional Input Ports and LTR/LTD Controller Lock Mode**

| rx_locktorefclk | rx_locktodata | LTR/LTD Controller Lock Mode |
|---|---|---|
| 1 | 0 | Manual – LTR Mode |
| x | 1 | Manual – LTD Mode |
| 0 | 0 | Automatic Lock Mode |

☞ If you do not instantiate the optional `rx_locktorefclk` and `rx_locktodata` signals, the Quartus II software automatically configures the LTR/LTD controller in automatic lock mode.

##### Automatic Lock Mode

In automatic lock mode, the LTR/LTD controller initially sets the CDR to lock to the input reference clock (LTR mode). After the CDR locks to the input reference clock, the LTR/LTD controller automatically sets it to lock to the incoming serial data (LTD mode) when the following three conditions are met:

- Signal threshold detection circuitry indicates the presence of valid signal levels at the receiver input buffer
  - Valid for PCIe mode only. The default condition is **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 frequency detector to set the CDR in LTR or LTD mode. The PPM detector and phase frequency 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.

The Altera-recommended transceiver reset sequence varies depending on the CDR lock mode.

For more information about reset sequence recommendations, refer to the *Reset Control and Power Down* chapter in volume 2 of the *Stratix IV Device Handbook*.

### Offset Cancellation in the Receiver Buffer and Receiver CDR

The offset cancellation logic corrects the analog voltage offsets from the required ranges due to process variations.The receiver buffer and receiver CDR require offset cancellation.

Offset cancellation is executed automatically once each time a HardCopy IV GX device is powered on. The control logic for offset cancellation is integrated into the ALTGX_RECONFIG megafunction. The `reconfig_fromgxb` and `reconfig_togxb` buses and the necessary clocks must be connected between the ALTGX instance and the ALTGX_RECONFIG instance.

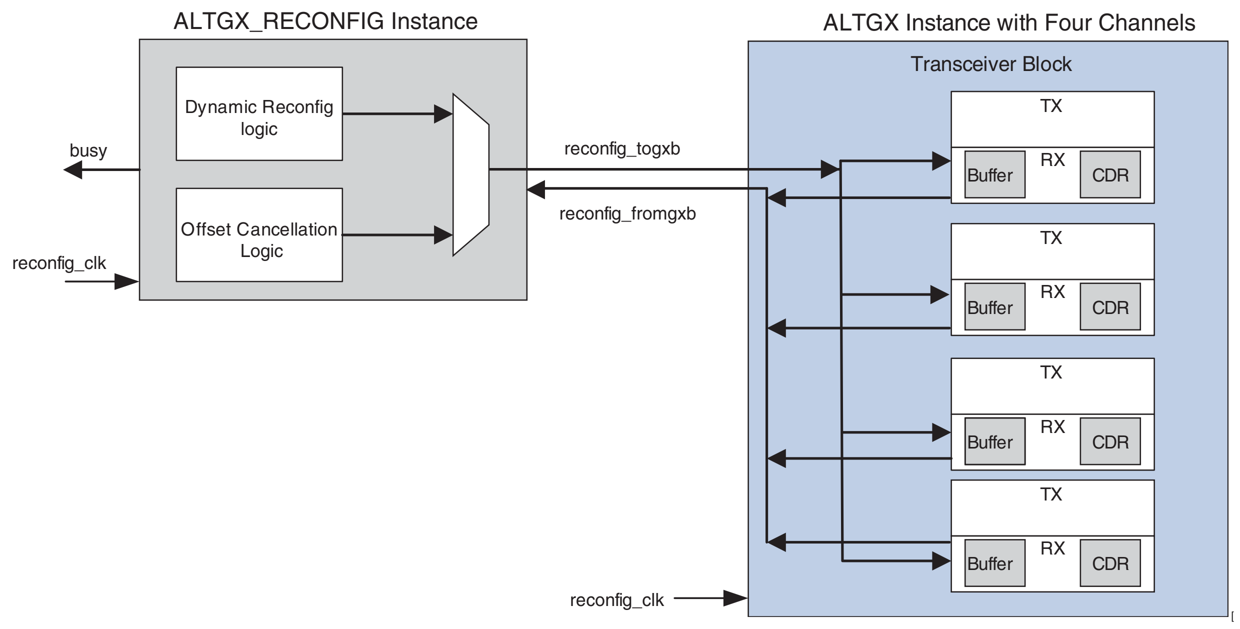
For more information about offset cancellation control logic connectivity, refer to the *Dynamic Reconfiguration in HardCopy IV GX Devices* chapter.

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 1–24 shows the interface of the offset cancellation control logic (ALTGX_RECONFIG instance) and the ALTGX instance.

**Figure 1–24. Interface of Offset Cancellation Control Logic to the ALTGX Instance**



For more information about the offset cancellation process and operation, refer to the *Offset Cancellation in the Receiver Buffer and Receiver CDR* section in the *Stratix IV Transceiver Architecture* chapter in volume 1 of the *Stratix IV Device Handbook*.

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* chapter in volume 2 of the *Stratix IV Device Handbook*.

## 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. The serial data is received from the LSB to MSB and the output of the deserializer goes to the word aligner block.

### Word Aligner

Because the data is serialized before transmission and then deserialized at the receiver, it loses the word boundary of the upstream transmitter after 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 HardCopy IV GX 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:

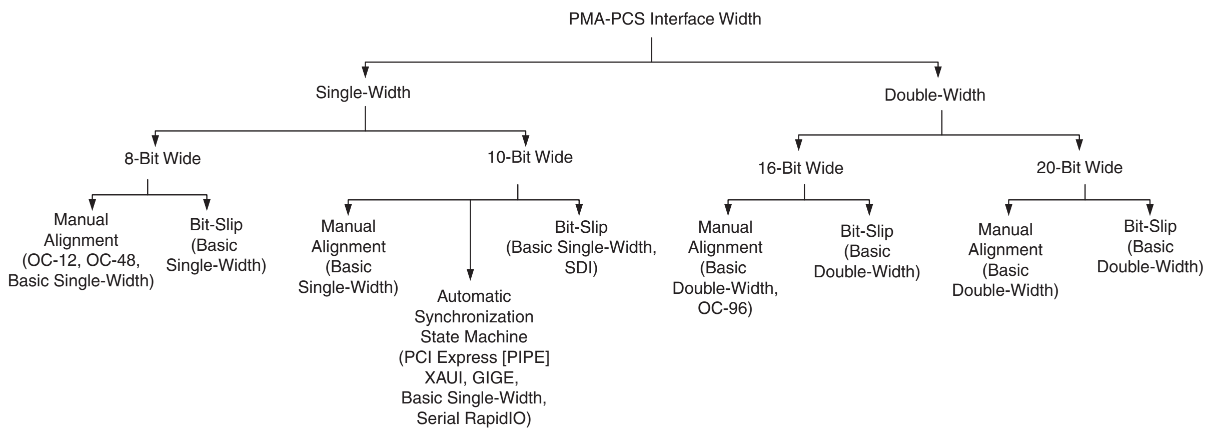
- Synchronization state machine in functional modes such as PCIe, XAUI, GIGE, Serial RapidIO, and Basic single-width
- Programmable run length violation detection in all functional modes
- Receiver polarity inversion in all functional modes except PCIe
- Receiver bit reversal in Basic single- and double-width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

Depending on the configured functional mode, the word aligner operates in one of the following three modes:

- Manual alignment
- Automatic synchronization state machine
- Bit-slip

Figure 1–25 shows the word aligner operation in all supported configurations.

**Figure 1–25. Word Aligner in All Supported Configurations**


PMA-PCS Interface Width
Single-Width
Double-Width
8-Bit Wide
10-Bit Wide
16-Bit Wide
20-Bit Wide
Manual Alignment (OC-12, OC-48, Basic Single-Width)
Bit-Slip (Basic Single-Width)
Manual Alignment (Basic Single-Width)
Automatic Synchronization State Machine (PCI Express [PIPE] XAUI, GIGE, Basic Single-Width, Serial RapidIO)
Bit-Slip (Basic Single-Width, SDI)
Manual Alignment (Basic Double-Width, OC-96)
Bit-Slip (Basic Double-Width)
Manual Alignment (Basic Double-Width)
Bit-Slip (Basic Double-Width)

The functionality of the word aligner for all the supported configurations is similar to that of the Stratix IV GX devices.

For more information about the word aligner, refer to the *Stratix IV Device Handbook*.

Table 1–13 lists the word aligner options available in Basic single-width and double-width modes.

**Table 1–13. Word Aligner Options Available in Basic Single- and Double-Width Modes** *(Note 1)* **(Part 1 of 2)**

| Functional Mode | PMA-PCS Interface Width | Word Alignment Mode | Word Alignment Pattern Length | rx_enapatternalign Sensitivity | rx_syncstatus Behavior | rx_patterndetect Behavior |
|---|---|---|---|---|---|---|
| Basic Single-Width | 8-bit | Manual Alignment | 16-bit | Rising Edge Sensitive | Asserted high for one parallel clock cycle when the word aligner aligns to a new word boundary. | Asserted high for one parallel clock cycle when the word alignment pattern appears in the current word boundary. |
| | | Bit-Slip | 16-bit | — | — | 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 | — | — | 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 | — | 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 1–13. Word Aligner Options Available in Basic Single- and Double-Width Modes** ***(Note 1)*** **(Part 2 of 2)**

| Functional Mode | PMA-PCS Interface Width | Word Alignment Mode | Word Alignment Pattern Length | rx_enapatternalign Sensitivity | rx_syncstatus Behavior | rx_patterndetect Behavior |
|---|---|---|---|---|---|---|
| Basic Double-Width | 16-bit | Manual Alignment | 8-, 16-, and 32-bit | Rising Edge Sensitive | Stays high after the word aligner aligns to the word alignment pattern. Goes low on receiving a rising edge on `rx_enapatternalign` until a new word alignment pattern is received. | Asserted high for one parallel clock cycle when the word alignment pattern appears in the current word boundary. |
| | | Bit-Slip | 8-, 16-, and 32-bit | — | — | 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 | — | — | Asserted high for one parallel clock cycle when the word alignment pattern appears in the current word boundary. |

**Note to Table 1–13:**

(1) For more information about the word aligner operation, refer to the *Stratix IV Transceiver Architecture* chapter in volume 2 of the *Stratix IV Device Handbook*.

#### 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 HCell Logic clock might have phase difference and/or PPM difference (in asynchronous systems) with respect to the recovered clock. To ensure that the HCell Logic 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 mode. You can assert the `rx_rlv` signal longer, depending on the run length of the received data.

Table 1–14 lists the detection capabilities of the run length violation circuit.

**Table 1–14. 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 receiver polarity inversion feature is provided to invert the polarity of the positive and negative signals of a serial differential link.

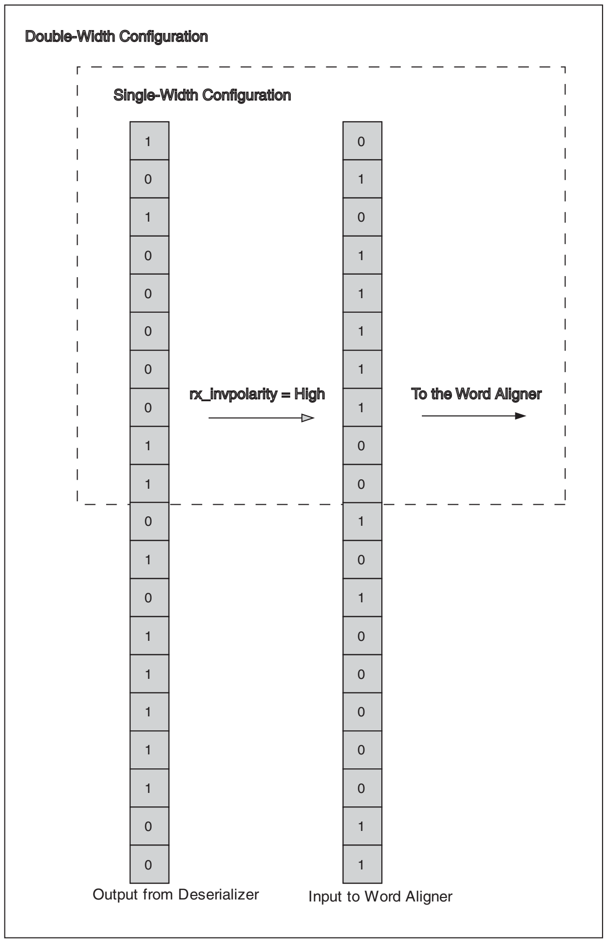
An optional `rx_invpolarity` port is available in all single- and double-width modes except (OIF) CEI PHY and PCIe modes to dynamically enable the receiver polarity inversion feature.

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 1–26 shows the receiver polarity inversion feature in single- and double-width datapath configurations.

**Figure 1–26. Receiver Polarity Inversion in Single- and Double Width Mode**



### Receiver Bit Reversal

By default, the HardCopy IV GX 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 HCell Logic 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- and double-width modes with the word aligner configured in bit-slip mode.

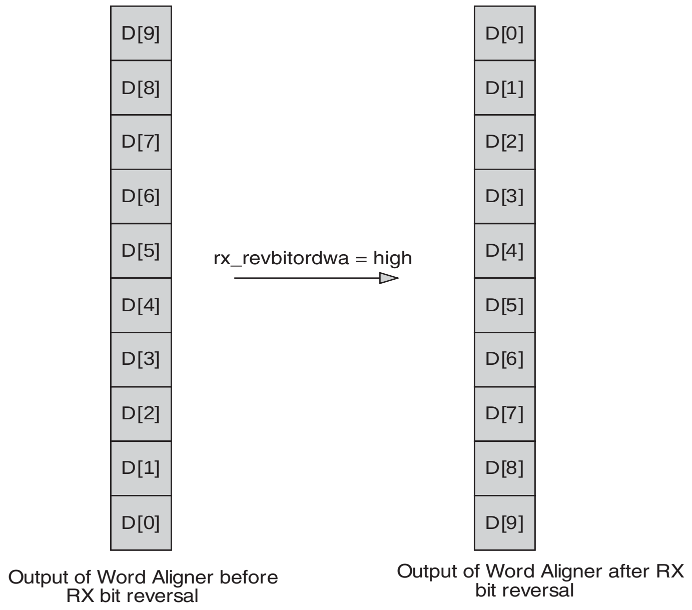
When the `rx_revbitordwa` signal is driven high in Basic single-width mode, the 8-bit or 10-bit data `D[7:0]` or `D[9:0]` at the output of the word aligner gets rewired to `D[0:7]` or `D[0:9]`, respectively.

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 HCell Logic on the `rx_dataout` port in the case of MSB-to-LSB transmission.

Figure 1–27 shows the receiver bit reversal feature in a 10-bit wide datapath configuration.

**Figure 1–27. Receiver Bit Reversal in a 10-Bit Wide Datapath**

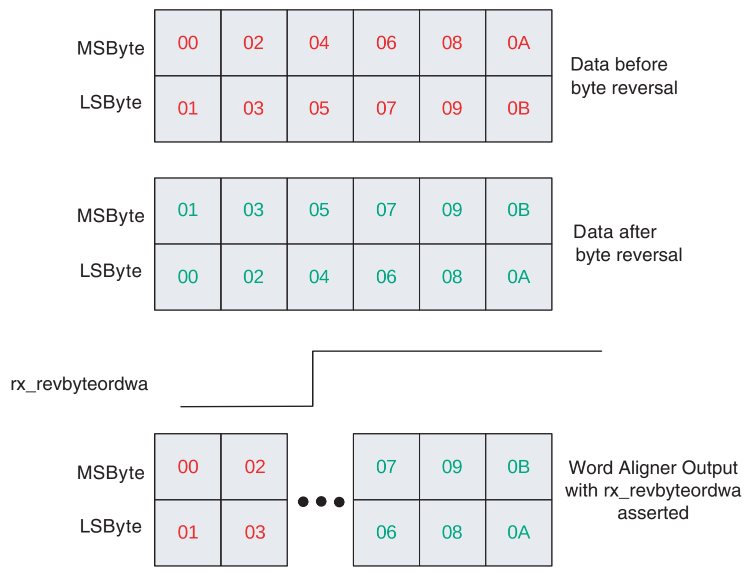


### Receiver Byte Reversal

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.

Figure 1–28 shows the receiver byte reversal feature.

**Figure 1–28. Receiver Byte Reversal Feature**



### Deskew FIFO

XAUI protocol requires the physical layer device to implement a deskew circuitry to align all four channels. The XAUI protocol allows a maximum skew of 40 UI (12.8 ns), as seen at the receiver of the four lanes.

The deskew circuitry consists of the following:

- 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 operation in XAUI functional mode is compliant to the PCS deskew state machine diagram specified in clause 48 of IEEE P802.3ae.

For more information about the deskew FIFO operations for channel deskew, refer to the *Deskew FIFO* section in the *Stratix IV Transceiver Architecture* chapter in volume 2 of the *Stratix IV Device Handbook.*

### Rate Match (Clock Rate Compensation) FIFO

The rate match FIFO compensates for small clock frequency differences between the upstream transmitter and the local receiver clocks by inserting or removing skip (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 symbol 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
- Serial RapidIO

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 HCell Logic:

- `rx_rmfifodatainserted`—indicates insertion of a skip character or ordered set
- `rx_rmfifodatadeleted`—indicates deletion of a skip character or ordered set
- `rx_rmfifofull`—indicates rate match FIFO full condition
- `rx_rmfifoempty`—indicates rate match FIFO empty condition

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

For more information about the rate match FIFO for the different functional modes, refer to the *Rate Match FIFO* section in the *Stratix IV Transceiver Architecture* chapter in volume 2 of the *Stratix IV Device Handbook*.

### 8B/10B Decoder

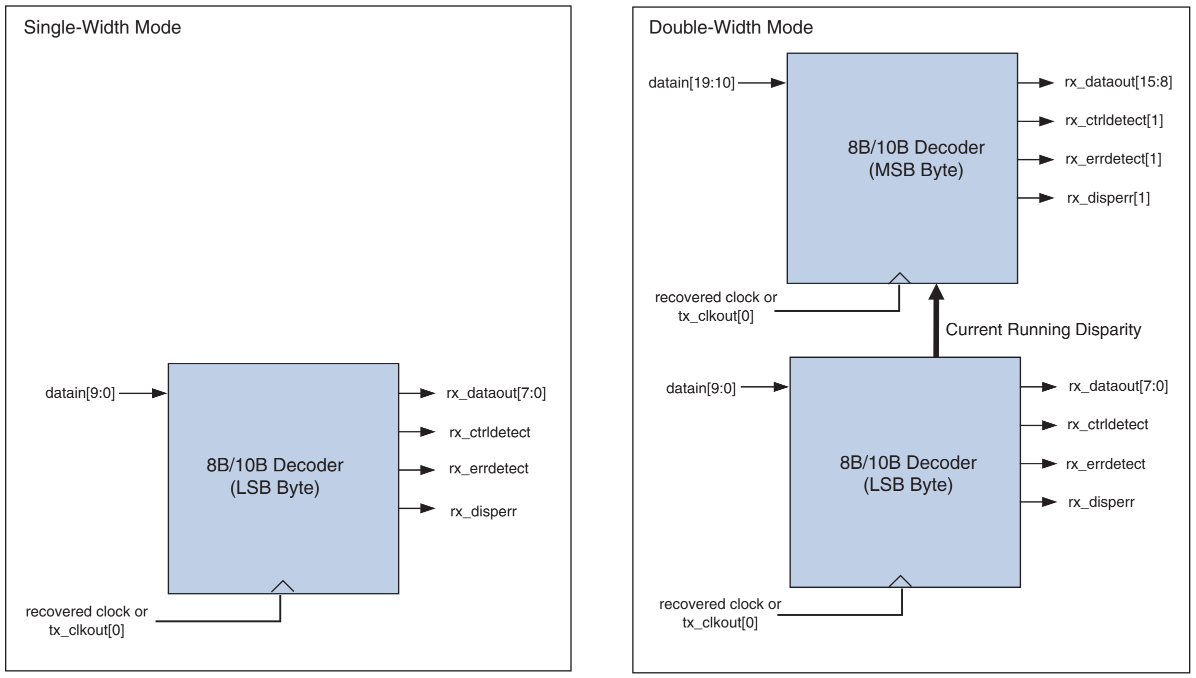
Protocols such as PCIe, XAUI, GIGE, and Serial RapidIO require the serial data sent over the link to be 8B/10B encoded to maintain the DC balance in the serial data transmitted. These protocols require the receiver PCS logic to implement an 8B/10B decoder to decode the data before forwarding it to the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

The HardCopy IV GX 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 1–29):

- Single-width mode
- Double-width mode

**Figure 1–29. 8B/10B Decoder in Single-Width and Double-Width Mode**



For more information about the 8B/10B decoder, refer to the *8B/10B Decoder* section in the *Stratix IV Transceiver Architecture* chapter in volume 2 of the *Stratix IV Device Handbook*.

### Byte Deserializer

The HCell Logic-transceiver interface frequency has an upper limit that is stated in the *Interface Frequency* section in the *DC and Switching Characteristics of HardCopy IV Devices* chapter. In functional modes that have a receiver PCS frequency greater than the upper limit stated in the *DC and Switching Characteristics of HardCopy IV Devices* chapter, the byte deserializer is required to reduce the HCell Logic-transceiver interface frequency to half while doubling the received parallel data width.

The byte deserializer is required in configurations that exceed the HCell Logic-transceiver interface clock upper frequency limit, but is optional in configurations that do not exceed this limit.

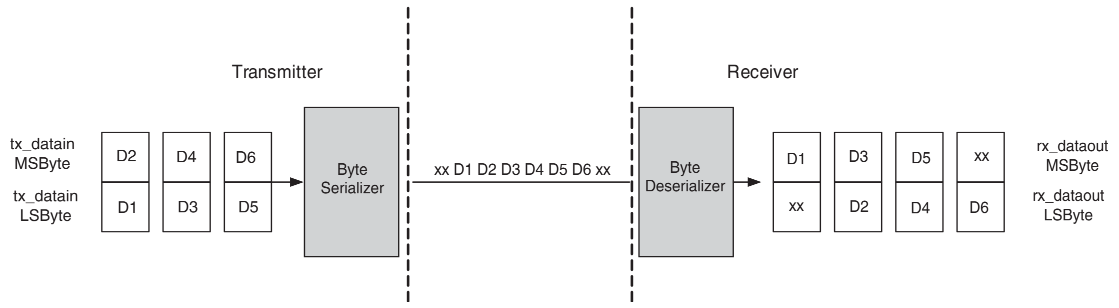
The byte deserializer operates in two modes:

- Single-width mode
- Double-width mode

### Byte Ordering Block

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 1–30 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 1–30. MSByte and LSByte of the Two-Byte Transmitter Data Straddled Across Two Word Boundaries**



HardCopy IV GX 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 HCell Logic. 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.

The byte ordering block modes of operation in both single- and double-width modes are:

- Word-alignment-based byte ordering
- User-controlled byte ordering

#### Word-Alignment-Based Byte Ordering

In word-alignment-based byte ordering, the byte ordering block starts looking for the byte ordering pattern in the byte-deserialized data every time it sees a rising edge on the `rx_syncstatus` signal. After a rising edge on the `rx_syncstatus` signal, if the byte ordering block finds the first data byte that matches the programmed byte ordering pattern in the MSByte position of the byte-deserialized data, it inserts one programmed PAD pattern to push the byte ordering pattern in the LSByte position. If the byte ordering block finds the first data byte that matches the programmed byte ordering pattern in the LSByte position of the byte-deserialized data, it considers the data to be byte ordered and does not insert any PAD pattern. In either case, the byte ordering block asserts the `rx_byteorderalignstatus` signal.

You can choose word-alignment-based byte ordering by selecting the `sync status` signal from the word aligner tab in the **What do you want the byte ordering to be based on?** field in the ALTGX MegaWizard Plug-In Manager.

#### 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.

For more information about byte ordering, refer to the *Stratix IV Transceiver Architecture* chapter in volume 2 of the *Stratix IV Handbook*.

### 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 HCell Logic. The receiver phase compensation FIFO compensates for the phase difference between the parallel receiver PCS clock (FIFO write clock) and the HCell Logic 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 1–15 lists the receiver phase compensation FIFO write clock source in different configurations.

**Table 1–15. Receiver Phase Compensation FIFO Write Clock Source (Part 1 of 2)**

| Configuration | Receiver Phase Compensation FIFO Write Clock | |
|---|---|---|
| | **Without Byte Serializer** | **With Byte Serializer** |
| Non-bonded channel configuration with rate matcher | Parallel transmitter PCS clock from the local clock divider in the associated channel (`tx_clkout`) | Divide-by-two version of the parallel transmitter PCS clock from the local clock divider in the associated channel (`tx_clkout`) |
| Non-bonded channel configuration without rate matcher | Parallel recovered clock from the receiver PMA in the associated channel (`rx_clkout`) | Divide-by-two version of the parallel recovered clock from the receiver PMA in the associated channel (`rx_clkout`) |

**Table 1–15. Receiver Phase Compensation FIFO Write Clock Source (Part 2 of 2)**

| Configuration | Receiver Phase Compensation FIFO Write Clock | |
|---|---|---|
| | **Without Byte Serializer** | **With Byte Serializer** |
| ×4 bonded channel configuration | Parallel transmitter PCS clock from the central clock divider in the `CMU0` of the associated transceiver block (`coreclkout`) | Divide-by-two version of the parallel transmitter PCS clock from the central clock divider in `CMU0` of the associated transceiver block (`coreclkout`) |
| ×8 bonded channel configuration | Parallel transmitter PCS clock from the central clock divider in `CMU0` of the master transceiver block (`coreclkout` from master transceiver block) | Divide-by-two version of the parallel transmitter PCS clock from the central clock divider in `CMU0` of the master transceiver block (`coreclkout` from master transceiver block) |

The receiver phase compensation FIFO read clock source varies depending on whether or not you instantiate the `rx_coreclk` port in the ALTGX MegaWizard Plug-In Manager. Table 1–16 lists the receiver phase compensation FIFO read clock source in different configurations.

**Table 1–16. Receiver Phase Compensation FIFO Read Clock Source**

| Configuration | Receiver Phase Compensation FIFO Read Clock | |
|---|---|---|
| | **rx_coreclk Port Not Instantiated** | **rx_coreclk Port Instantiated** *(1)* |
| Non-bonded channel configuration with rate matcher | HCell Logic clock driven by the clock signal on the `tx_clkout` port | HCell Logic clock driven by the clock signal on the `rx_coreclk` port |
| Non-bonded channel configuration without rate matcher | HCell Logic clock driven by the clock signal on the `rx_clkout` port | HCell Logic clock driven by the clock signal on the `rx_coreclk` port |
| ×4 bonded channel configuration | HCell Logic clock driven by the clock signal on the `coreclkout` port | HCell Logic clock driven by the clock signal on the `rx_coreclk` port |
| ×8 bonded channel configuration | HCell Logic clock driven by the clock signal on the `coreclkout` port | HCell Logic clock driven by the clock signal on the `rx_coreclk` port |

**Note to Table 1–16:**

(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

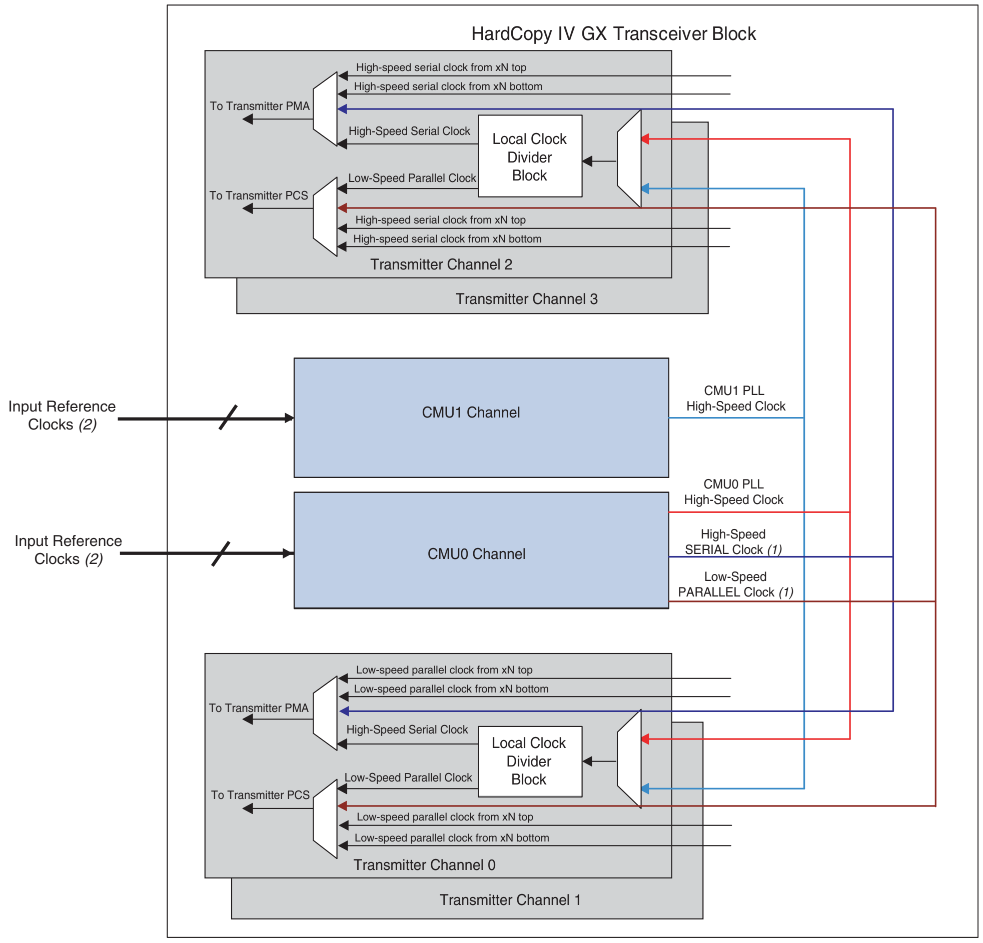
An optional `rx_phase_comp_fifo_error` port is available in all functional modes to indicate a receiver phase compensation FIFO under-run or overflow condition. The `rx_phase_comp_fifo_error` signal is asserted high when the phase compensation FIFO gets either full or empty. This feature is useful to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

## CMU Channel Architecture

HardCopy IV GX 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 1–31 shows the two CMU channels in a transceiver block.

**Figure 1–31. CMU Channels in a Transceiver Block**


HardCopy IV GX Transceiver Block
High-speed serial clock from xN top
High-speed serial clock from xN bottom
To Transmitter PMA
High-Speed Serial Clock
Local Clock Divider Block
Low-Speed Parallel Clock
To Transmitter PCS
High-speed serial clock from xN top
High-speed serial clock from xN bottom
Transmitter Channel 2
Transmitter Channel 3
Input Reference Clocks (2)
CMU1 Channel
CMU1 PLL High-Speed Clock
CMU0 PLL High-Speed Clock
Input Reference Clocks (2)
CMU0 Channel
High-Speed SERIAL Clock (1)
Low-Speed PARALLEL Clock (1)
Low-speed parallel clock from xN top
Low-speed parallel clock from xN bottom
To Transmitter PMA
High-Speed Serial Clock
Local Clock Divider Block
Low-Speed Parallel Clock
To Transmitter PCS
Low-speed parallel clock from xN top
Low-speed parallel clock from xN bottom
Transmitter Channel 0
Transmitter Channel 1

**Notes to Figure 1–31:**

(1) Clocks are provided to support bonded channel functional mode.

(2) For more information, refer to the *Clock Networks and PLLs in HardCopy 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 ×4, ×8, and ×N bonded mode configurations such as XAUI, Basic ×4, Basic ×8, and Basic (PMA-Direct) ×N.

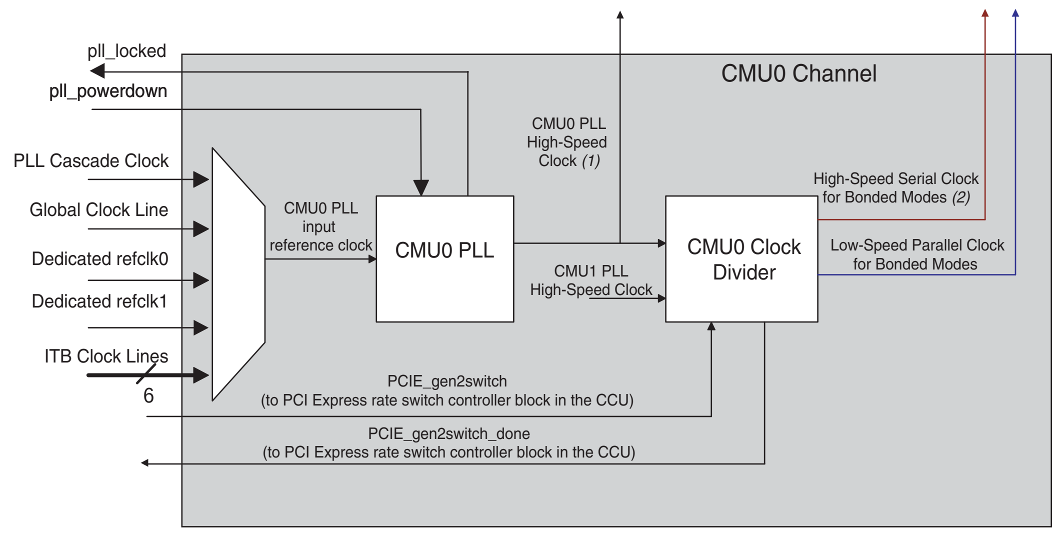
The `CMU0` channel has additional capabilities to support bonded protocol functional modes such as Basic ×4, XAUI, and PCIe. Use the ALTGX MegaWizard Plug-In Manager to select these functional modes (to enable Basic ×4 functional mode, select the **×4** option in Basic mode). For more information, refer to "Functional Modes" on page 1–55.

### CMU0 Channel

The `CMU0` channel, shown in Figure 1–32, contains the following blocks:

- `CMU0` PLL (refer to "CMU0 Clock Divider" on page 1–48)
- `CMU0` clock divider (refer to "CMU Clock Divider" on page 1–53)

**Figure 1–32. CMU0 Channel with the CMU0 PLL and CMU0 Clock Divider**


pll_locked
pll_powerdown
CMU0 Channel
CMU0 PLL High-Speed Clock (1)
PLL Cascade Clock
Global Clock Line
Dedicated refclk0
Dedicated refclk1
ITB Clock Lines
6
CMU0 PLL input reference clock
CMU0 PLL
CMU1 PLL High-Speed Clock
CMU0 Clock Divider
High-Speed Serial Clock for Bonded Modes (2)
Low-Speed Parallel Clock for Bonded Modes
PCIE_gen2switch (to PCI Express rate switch controller block in the CCU)
PCIE_gen2switch_done (to PCI Express rate switch controller block in the CCU)

**Notes to Figure 1–32:**

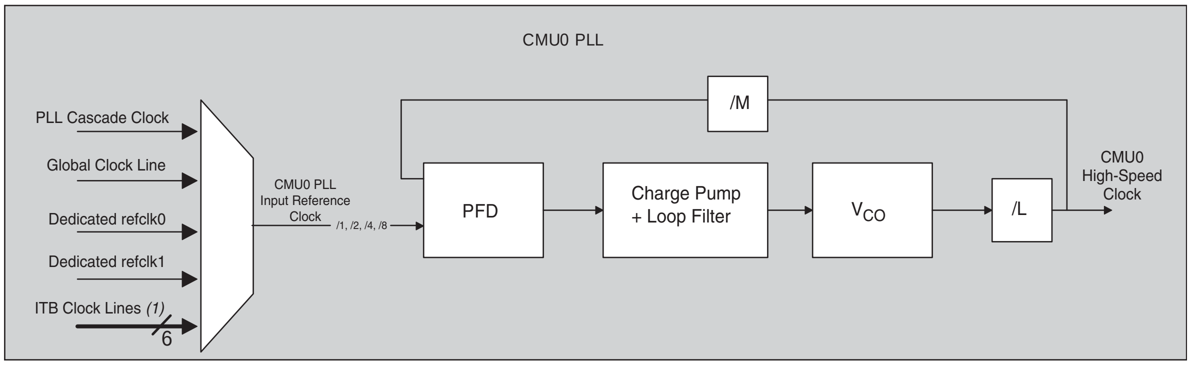
(1) To provide clocks for its PMA and PCS blocks in non-bonded functional modes (for example, GIGE functional mode), the transmitter channel uses the transmitter local clock divider to divide this high-speed clock output.

(2) Used in XAUI, Basic ×4, and PCIe ×4 functional modes. In PCIe ×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 1–33 shows the `CMU0` PLL.

**Figure 1–33. CMU0 PLL**



**Note to Figure 1–33:**

(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 HCell Logic
- 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 *Clock Networks and PLLs in HardCopy 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 6.5 Gbps, the `CMU0` PLL uses two multiplier blocks (/M and /L) in the feedback path (shown in Figure 1–33).

The ALTGX MegaWizard Plug-In Manager provides the list of input reference clock frequencies based on the data rate you select. The Quartus II software automatically selects the /M and /L settings based on the input reference clock frequency and serial data rate.

The `CMU0` and `CMU1` PLLs have a dedicated `pll_locked` signal that is asserted to indicate that the CMU PLL is locked to the input reference clock. You can use the `pll_locked` signal in your transceiver reset sequence, as described in the *Reset Control and Power Down* chapter in volume 2 of the *Stratix IV Device Handbook*.

#### PLL Bandwidth Setting

The bandwidth of a PLL is the measure of its ability to track input clock and jitter. It is determined by the –3 dB frequency of the closed-loop gain of the PLL. There are three bandwidth settings: high, medium, and low. You can program the PLL bandwidth setting using the ALTGX MegaWizard Plug-In Manager.

- The high bandwidth setting filters out internal noise from the VCO because it tracks the input clock above the frequency of the internal VCO noise.
- With the low bandwidth setting, if the noise on the input reference clock is greater than the internal noise of the VCO, the PLL filters out the noise above the –3 dB frequency of the closed-loop gain of the PLL.
- The medium bandwidth setting is a compromise between the high and low bandwidth settings.

The –3 dB frequencies for these settings can vary because of the non-linear nature and frequency dependencies of the circuit.

#### Power Down CMU0 PLL

You can power down the `CMU0` PLL by asserting the `pll_powerdown` signal.

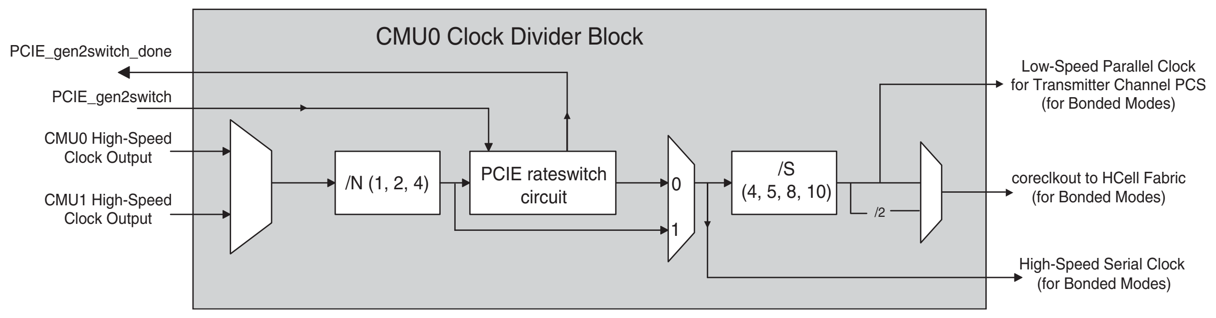
For more information about the `CMU0` PLL, refer to the *Reset Control and Power Down* chapter in volume 2 of the *Stratix IV Device Handbook*.

### 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.

You can configure the `CMU0` clock divider shown in Figure 1–34 to select the high-speed clock output from the `CMU0` or `CMU1` PLLs. The `CMU1` PLL is present in the `CMU1` channel.

**Figure 1–34. CMU0 Clock Divider**


CMU0 Clock Divider Block
PCIE_gen2switch_done
PCIE_gen2switch
CMU0 High-Speed Clock Output
CMU1 High-Speed Clock Output
/N (1, 2, 4)
PCIE rateswitch circuit
0
1
/S (4, 5, 8, 10)
/2
Low-Speed Parallel Clock for Transmitter Channel PCS (for Bonded Modes)
coreclkout to HCell Fabric (for Bonded Modes)
High-Speed Serial Clock (for Bonded Modes)

### High-Speed Serial Clock Generation

The /N divider receives the high-speed clock output from one of the CMU PLLs and produces a high-speed serial clock. Use this clock for bonded functional modes such as Basic ×4 and ×8, XAUI, and PCIe ×4 and ×8 configurations. In XAUI and Basic ×4/×8 modes, the Quartus II software chooses the path (shown by "1" in the multiplexer) and provides the high-speed serial clock to all the transmitter channels within the transceiver block.

- In PCIe ×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 ×8 mode and Basic ×8 mode, only the `CMU0` clock divider of the master transceiver block provides the high-speed serial clock to all eight channels.
- In PCIe ×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.

### PCIE Rateswitch Circuit

The PCIE rateswitch circuit is enabled only in PCIe ×4 mode. In PCIe ×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.

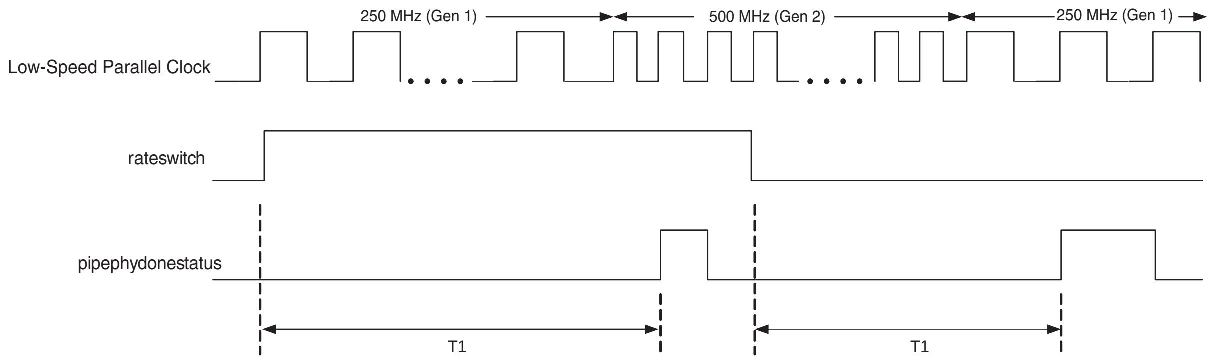
- When you set the `rateswitch` port to **0**, the PCIe rateswitch controller (in the CCU) signals the PCIE rateswitch circuit to select the divide by /2 to provide a high-speed serial clock for the Gen1 (2.5 Gbps) data rate.
- When you set the `rateswitch` port to **1**, the /N divider output is forwarded, 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 1–35 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 1–76.

**Figure 1–35. Rateswitch in PCIe Mode** ***(Note 1)***



**Note to Figure 1–35:**

(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

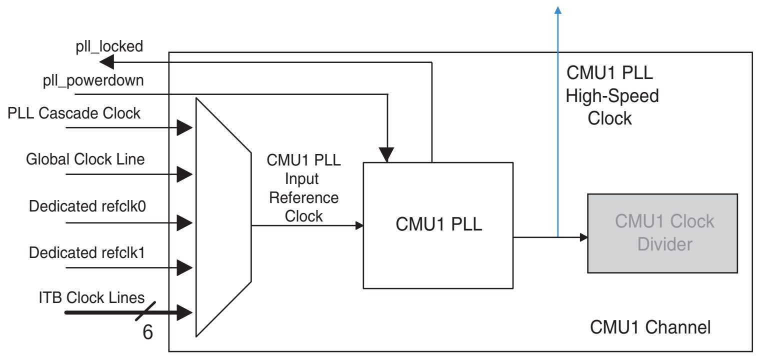
The /S divider receives the clock output from the /N divider or PCIe rateswitch circuit (only in PCIe mode) and generates the low-speed parallel clock for the PCS block of all transmitter channels and `coreclkout` for the HCell Logic. If the byte serializer block is enabled in bonded channel modes, the /S divider output is divided by the /2 divider and sent out as `coreclkout` to the HCell Logic. The Quartus II software automatically selects the /S values based on the deserialization width setting (single- or double-width mode) that you select in the ALTGX MegaWizard Plug-In Manager. For more information about single- or double-width mode, refer to "Transceiver Channel Architecture" on page 1–7.

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 1–36, 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 1–47).

**Figure 1–36. 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 a Transceiver Block* chapter in volume 2 of the *Stratix IV Device Handbook*.

#### Power Down CMU1 PLL

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

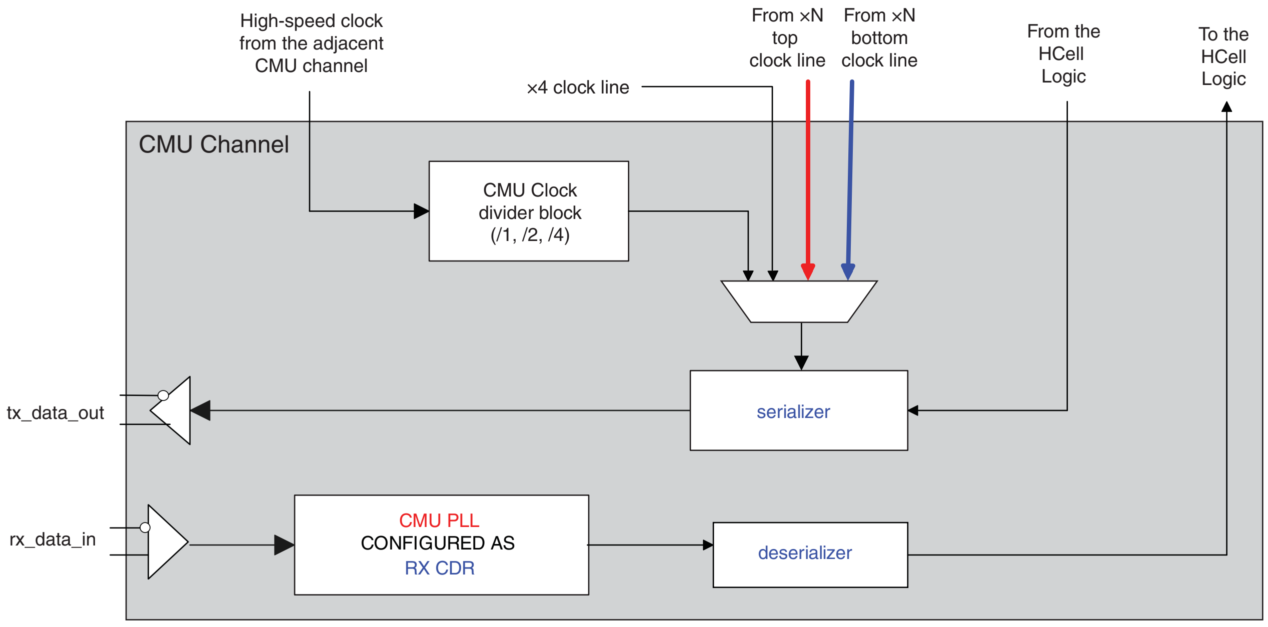
For more information about the `CMU1` PLL, refer to the *Reset Control and Power Down* chapter in volume 2 of the *Stratix IV Device Handbook*.

### Configuring CMU Channels as Transceiver Channels

You can configure the two CMU channels in the transceiver block of HardCopy IV GX devices as full-duplex PMA-only channels to run between 600 Mbps and 6.5 Gbps.

Figure 1–37 shows the functional blocks that are enabled to support the transceiver channel functionality.

**Figure 1–37. Functional Blocks Enabled to Support Transceiver Channel Functionality**


High-speed clock from the adjacent CMU channel
×4 clock line
From ×N top clock line
From ×N bottom clock line
From the HCell Logic
To the HCell Logic
CMU Channel
CMU Clock divider block (/1, /2, /4)
serializer
tx_data_out
rx_data_in
CMU PLL CONFIGURED AS RX CDR
deserializer

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 HCell Logic.

When configured as a full-duplex or transmitter-only channel, the serializer in the CMU channel serializes the parallel data from the HCell Logic and drives the serial data to the transmitter buffer.

Table 1–17 lists the pins that are used as transmit and receive serial pins.

**Table 1–17. 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 |

**Table 1–17. 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 |
|---|---|---|
| `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 |
| `GXB_TX_[L,R][1,3,5,7]P` *(3)* | Transmit serial output for CMU `Channel1` | Not available for use |

**Notes to Table 1–17:**

(1) These indexes are for the HardCopy IV GX 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 *HardCopy IV Device Family Overview* chapter.

(2) Pins 0,2,4, and 6 are hardwired to the CMU `channel0` in the corresponding transceiver blocks.

(3) Pins 1,3,5, and 7 are hardwired to the 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 1–17.

### Serializer and Deserializer

The serializer and deserializer convert the serial-to-parallel data on the transmitter and receiver side, respectively. The ALTGX MegaWizard Plug-In Manager provides the Basic (PMA Direct) functional mode (with a **none** and **×N** option) to configure a transceiver channel to enable the transmitter serializer and receiver deserializer. To configure a CMU channel as a transceiver channel, you must use this functional mode.

The input data width options to serializer/from deserializer for a channel configured in this mode are 8, 10, 16, and 20.

For information about the maximum HCell Logic-transceiver interface clock frequency limits, refer to the *Transmitter Channel Datapath Clocking* section in the *Stratix IV Transceiver Clocking* chapter in volume 2 of the *Stratix IV Device Handbook*.

### CMU Clock Divider

When you configure a CMU channel in Basic (PMA Direct) ×1 mode, the CMU clock divider divides the high-speed clock from the other CMU channel (used as a clock generation unit) within the same transceiver block and provides the high-speed serial clock and low-speed parallel clocks to the transmitter side of the CMU channel. The CMU clock divider can divide the high-speed clock by /1, /2, and /4.

### Clocks for the Transmitter Serializer

When you configure the CMU channel as a transceiver channel, the clocks for the transmitter side is provided by one of the following sources:

- From the other CMU channel in the same transceiver block that is configured as a clock multiplication unit

- From CMU `channel0` on the other transceiver block on the same side of the device through the ×N clock line (the `×N_Top` or `×N_Bottom` clock line). You can use this clocking option if you configure a CMU channel in Basic (PMA Direct) ×N mode.
- From one of the ATX PLL blocks on the same side of the device through the ×N clock line (the `×N_Top` or `×N_Bottom` clock line)

#### Input Reference Clocks for the Receiver CDR

When you configure a CMU channel as a transceiver channel, there are multiple sources of input reference clocks for the receiver CDR:

- From adjacent REFCLKs within the same transceiver block, if the adjacent CMU channel is not used as a transceiver channel
- From the REFCLK of adjacent transceiver blocks on the same side of the device, if the corresponding CMU channels are not used as transceiver channels.

For more information about the input reference clocks, refer to the *Input Reference Clocking* section in the *Stratix IV Transceiver Clocking* chapter in volume 2 of the *Stratix IV Device Handbook*.

#### 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 1–18 and "Receiver Input Buffer" on page 1–22.

For information about loopback, refer to "Loopback Modes" on page 1–108.

### Dynamic Reconfiguration of the CMU Channel Analog Controls

For the dynamic reconfiguration capabilities of the CMU channels in Basic (PMA Direct) ×1 and ×N configurations, refer to the *Dynamic Reconfiguration in HardCopy IV GX Devices* chapter.

## Functional Modes

Table 1–18 lists the transceiver functional modes you can use to configure HardCopy IV GX devices using the ALTGX MegaWizard Plug-In Manager.

**Table 1–18. Functional Modes for the HardCopy IV GX Devices**

| Functional Mode | Data Rate | Refer To |
|---|---|---|
| Basic Single Width | 600 Mbps to 3.75 Gbps | "Basic Single-Width Mode Configurations" on page 1–57 |
| Basic Double Width | 1 Gbps to 6.5 Gbps | "Basic Double-Width Mode Configurations" on page 1–59 |
| PCIe | ■ Gen1 at 2.5 Gbps<br>■ Gen2 at 5 Gbps | "PCIe Mode" on page 1–66 |
| XAUI | 3.125 Gbps up to HiGig at 3.75 Gbps | "XAUI Mode Datapath" on page 1–90 |
| GIGE | 1.25 Gbps | "GIGE Mode Datapath" on page 1–94 |
| Serial RapidIO | ■ 1.25 Gbps<br>■ 2.5 Gbps<br>■ 3.125 Gbps | "Serial RapidIO Mode" on page 1–104 |
| SONET/SDH | ■ OC-12<br>■ OC-48 | "SONET/SDH Mode" on page 1–95 |
| SDI | ■ HD at 1.485/1.4835 Gbps<br>■ 3G at 2.97/2.967 Gbps | "SDI Mode Datapath" on page 1–102 |
| (OIF) CEI PHY Interface | > 4.976 Gbps to 6.375 Gbps | "(OIF) CEI PHY Interface Mode Datapath" on page 1–104 |

### Basic Functional Mode

The HardCopy IV GX 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 1–19 lists the HardCopy IV GX PCS-PMA interface widths and data rates supported in Basic single-width and double-width modes.

**Table 1–19. PCS-PMA Interface Widths and Data Rates in Basic Single-Width and Double-Width Modes for HardCopy IV GX 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 6.5 Gbps | 16 bit, 20 bit |

**Note to Table 1–19:**

(1) The data rate range supported in Basic single- 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 1–57 and "Basic Double-Width Mode Configurations" on page 1–59.

### Low Latency PCS Datapath

The ALTGX MegaWizard Plug-In Manager provides an **Enable low latency PCS mode** option when configured in Basic single- 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 1–57 and "Basic Double-Width Mode Configurations" on page 1–59.

☞ The PCS latency in Basic single- and Basic double-width modes with and without the low latency PCS mode option is pending characterization.

## Basic Single-Width Mode Configurations

Figure 1–38 shows HardCopy IV GX transceiver configurations allowed in Basic single-width functional mode with an 8-bit PMA-PCS interface.

**Figure 1–38. Transceiver Configurations in Basic Single-Width Mode with an 8-Bit PMA-PCS Interface for HardCopy IV GX Devices**

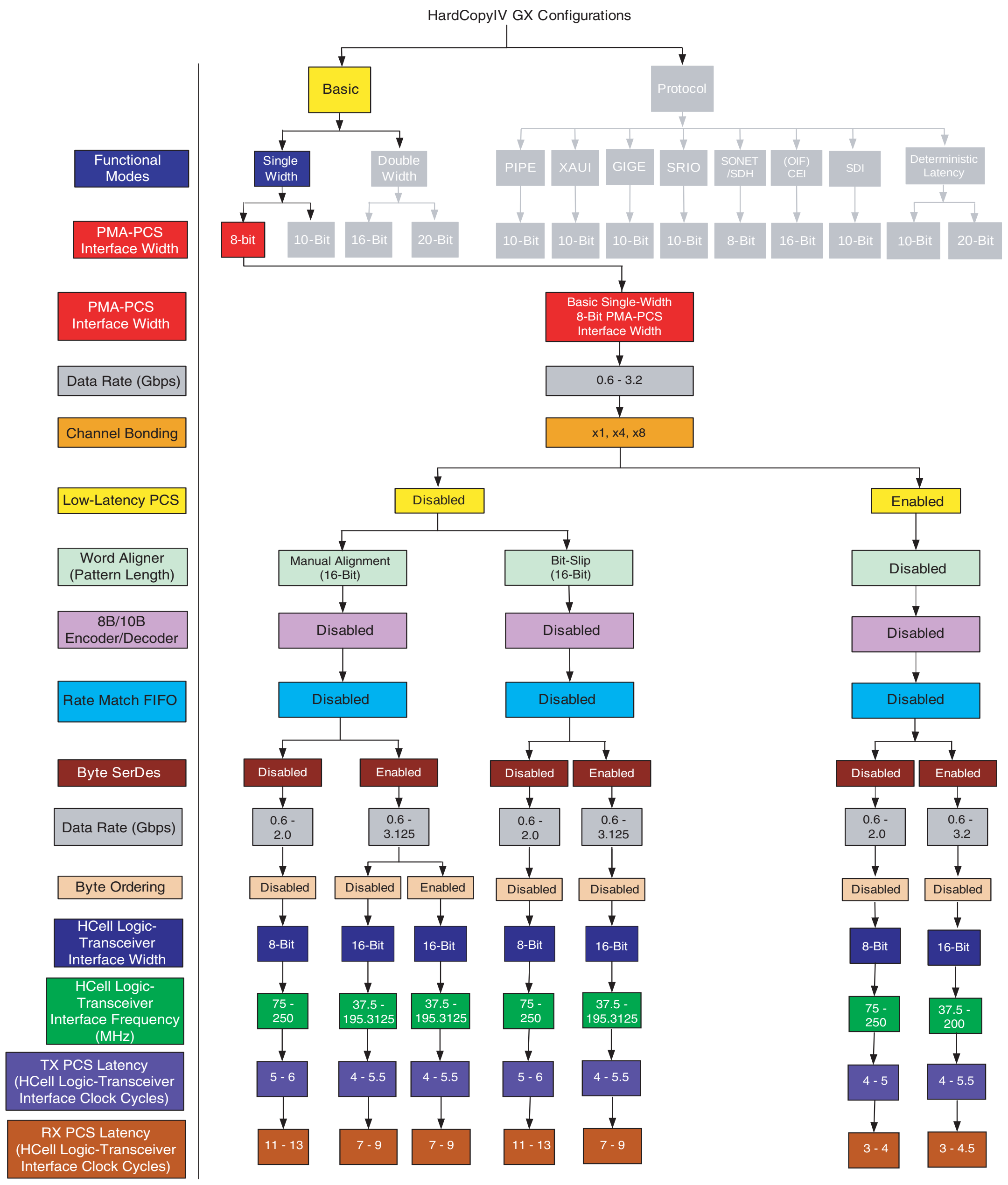
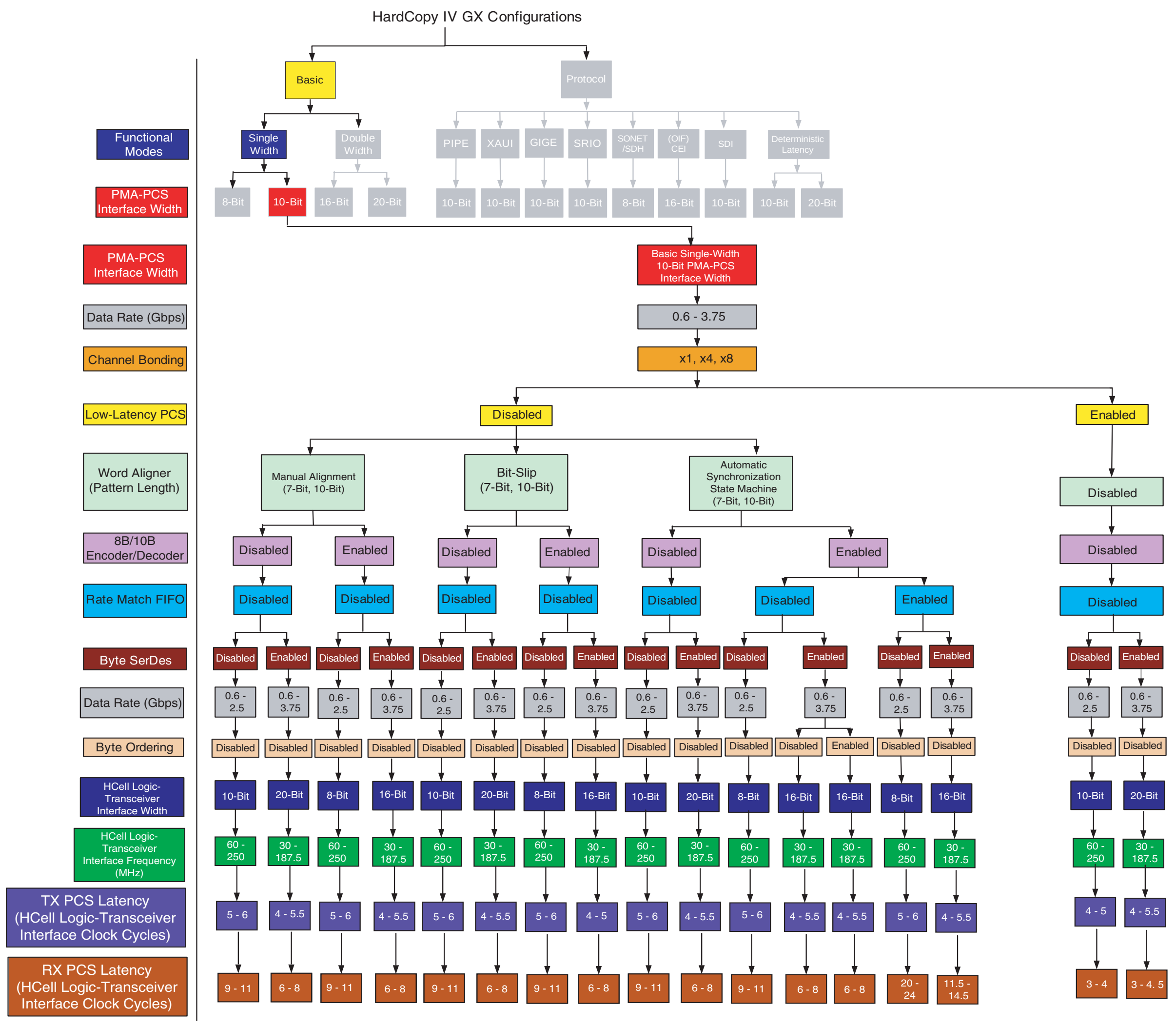


Figure 1–39 shows HardCopy IV GX transceiver configurations allowed in Basic single-width functional mode with a 10-bit PMA-PCS interface.

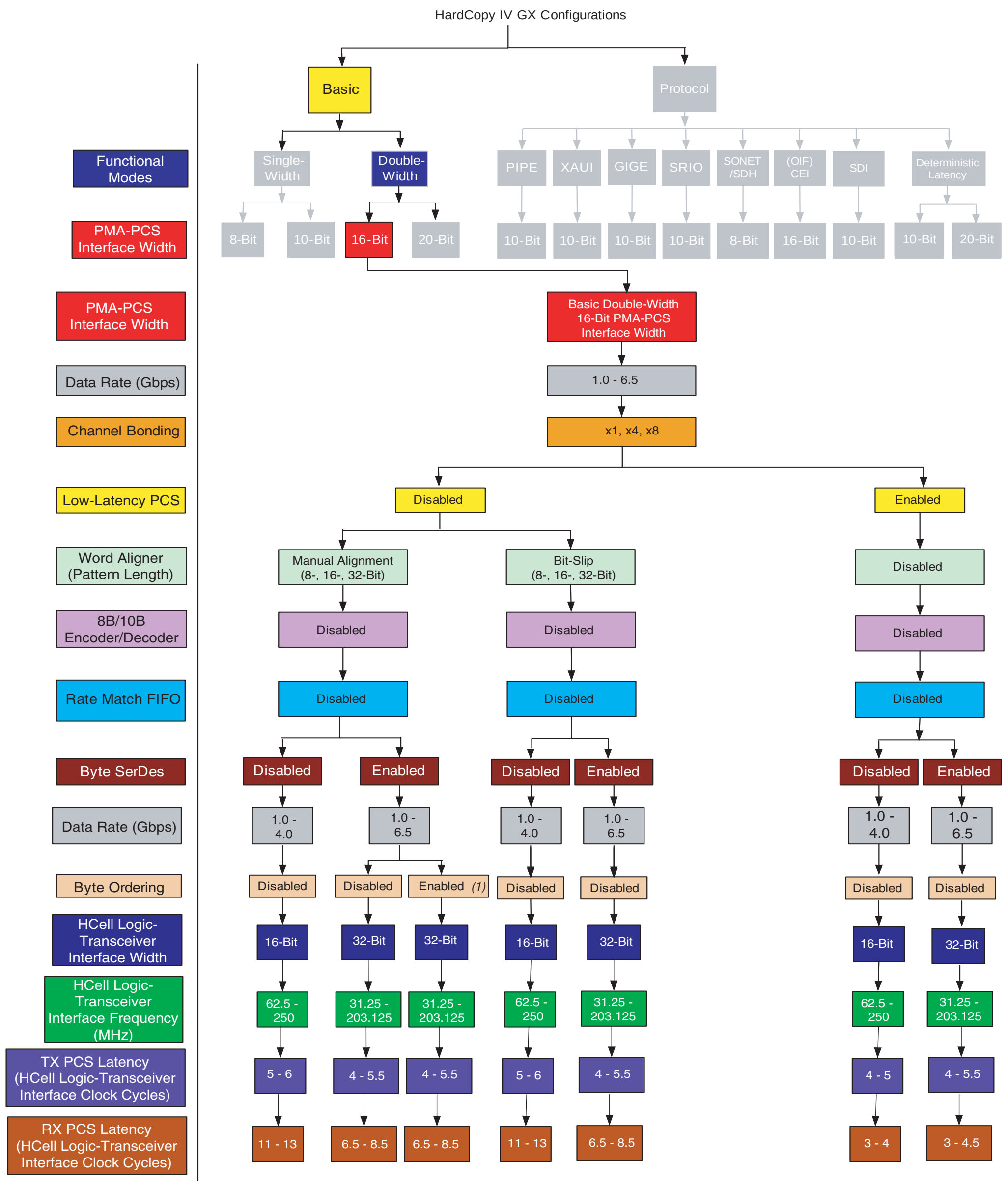
**Figure 1–39. Transceiver Configurations in Basic Single-Width Mode with a 10-Bit PMA-PCS Interface for HardCopy IV GX Devices**



## Basic Double-Width Mode Configurations

Figure 1–40 shows HardCopy IV GX transceiver configurations allowed in Basic double-width functional mode with a 16-bit PMA-PCS interface.

**Figure 1–40. Transceiver Configurations in Basic Double-Width Mode with a 16-Bit PMA-PCS Interface for HardCopy IV GX Devices**

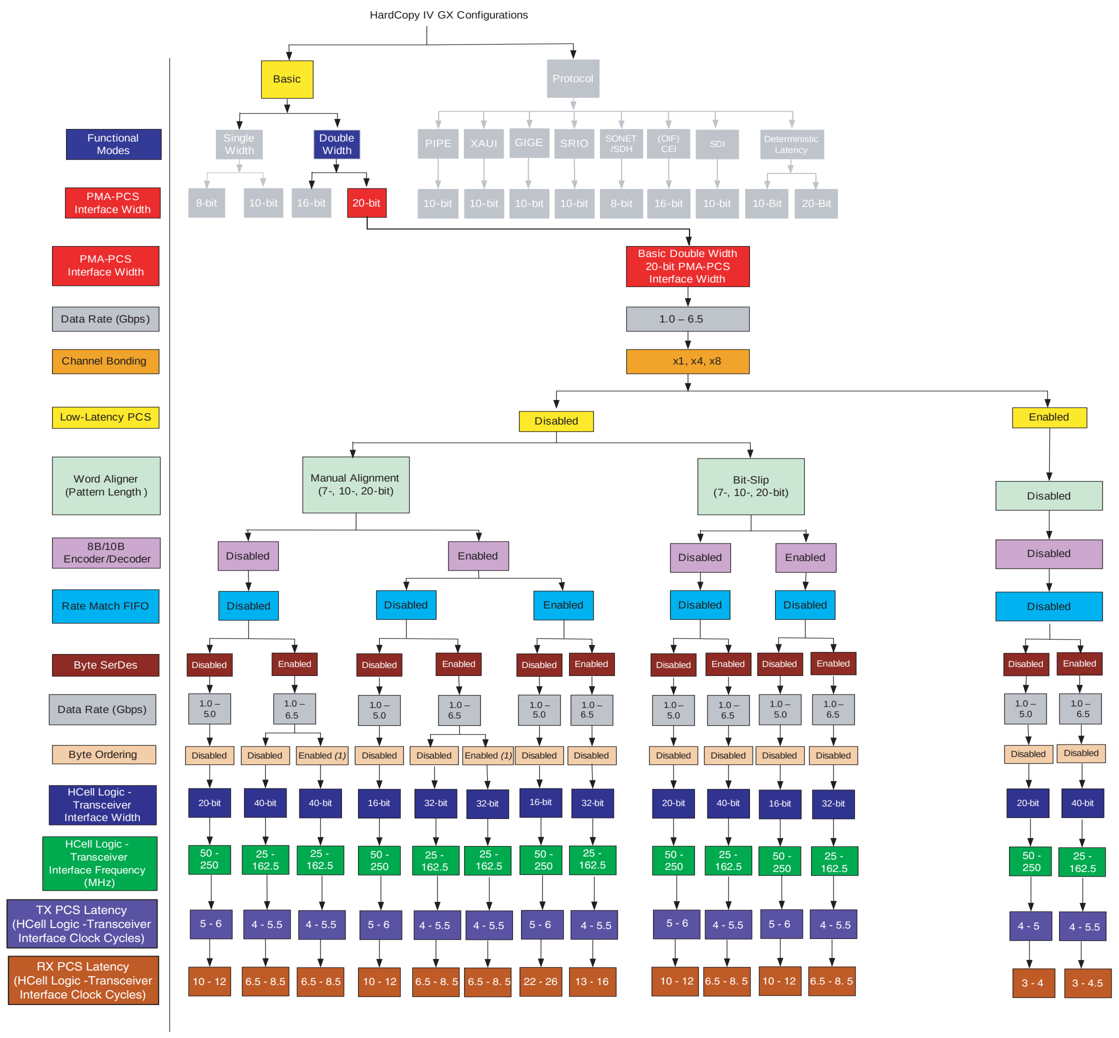


**Notes to Figure 1–40:**

(1) The byte ordering block is available only if you select the word alignment pattern length of 16 or 32 bits.

Figure 1–41 shows HardCopy IV GX transceiver configurations allowed in Basic double-width functional mode with a 20-bit PMA-PCS interface.

**Figure 1–41. Transceiver Configurations in Basic Double-Width Mode with a 20-Bit PMA-PCS Interface for HardCopy IV GX Devices**



**Notes to Figure 1–41:**

(1) The byte ordering block is available only if you select the word alignment pattern length of 20 bits.

## SATA and SAS Options

HardCopy IV GX devices offer options to implement a transceiver that satisfies SATA and SAS protocols. These options are:

- Transmitter in electrical idle mode
- Receiver signal detect functionality

These options and their selections are described in the following sections.

#### Transmitter Buffer Electrical Idle

In Basic functional mode, you can enable the optional input signal `tx_forceelecidle`. When this input signal of a channel is asserted high, the transmitter buffer in that channel is placed in the electrical idle state. During electrical idle, the output of the transmitter buffer is tri-stated.

This signal is used in applications like SATA and SAS for generating out of band (OOB) signals. An OOB signal is a pattern of idle times and burst times. Different OOB signals are distinguished by their different idle times.

☞ For more information about the transmitter buffer in the Electrical Idle state, refer to the *Transmitter Buffer Electrical Idle* section in "PCIe Mode" on page 1–66.

#### Receiver Input Signal Detect

In Basic functional mode, you can enable the optional `rx_signaldetect` signal (used for protocols such as SATA and SAS) only if you select the 8B/10B block. When you select the optional `rx_signaldetect` signal, an option is available to set the desired threshold level of the signal being received at the receiver's input buffer. If the signal threshold detection circuitry senses the signal level present at the receiver input buffer to be higher than the chosen signal detect threshold, it asserts the `rx_signaldetect` signal high. Otherwise, the signal threshold detection circuitry de-asserts the `rx_signaldetect` signal low. This signal is useful in applications such as SATA and SAS for detecting out of band (OOB) signals.

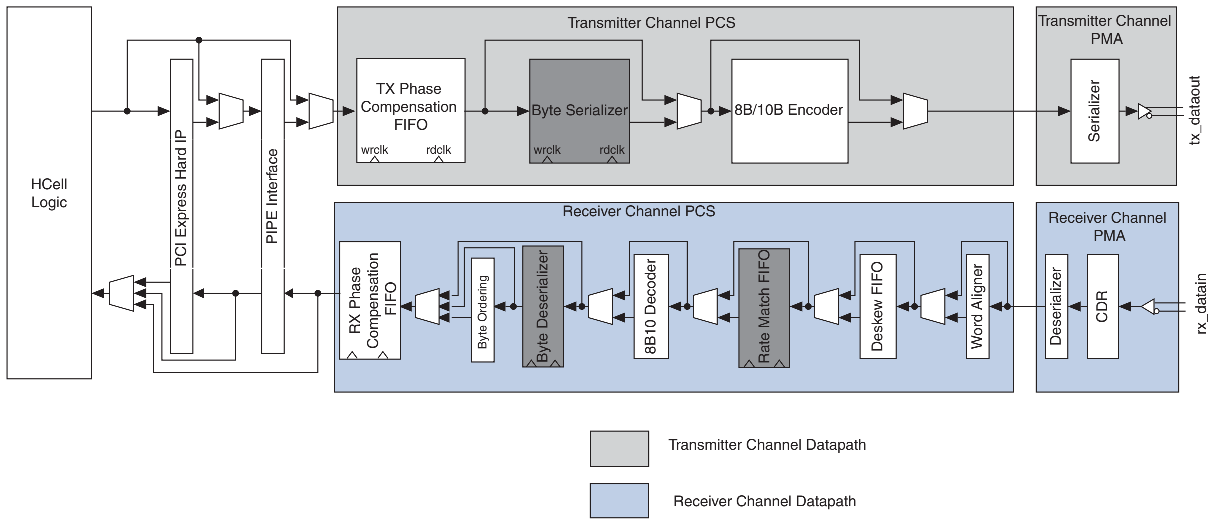
For information about other protocols supported using Basic functional mode, refer to *AN 577: Recommended Protocol Configurations for Stratix IV GX FPGAs*.

### Deterministic Latency Mode

HardCopy IV GX devices have a deterministic latency option available for use in high-speed serial interfaces such as Common Public Radio Interface (CPRI) 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 1–42 shows the transceiver datapath when using deterministic latency mode.

**Figure 1–42. 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 1–20 lists the number of bits that are allowed to be slipped under different channel widths.

**Table 1–20. Number of Transmitter Bits Allowed to be Slipped in Deterministic Latency Mode**

| Channel Width | Slip Zero |
|---|---|
| 8 bits, 10 bits | 9 bits |
| 16 bits, 20 bits | 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 the 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 the CMU PLL input reference clock.

In order to achieve deterministic latency through the transceiver, the reference clock to the CMU PLL must be the same as the low-speed parallel clock. For example, if you need a data rate of 1.2288 Gbps to be implemented for the CPRI protocol that places stringent requirements on the amount of latency variation, you must choose a reference clock of 122.88 MHz to allow for a feedback path from the CMU PLL to be used. This feedback path reduces the variations in latency.

When selecting this option, you must provide an input reference clock to the CMU PLL that is of the same frequency as the low-speed parallel clock.

In a CPRI implementation, the input reference clock to the CMU PLL must be the same as the low-speed parallel clock. Each CPRI channel uses one CMU PLL; therefore, each transceiver block can implement two CPRI ×1 channels only. ATX PLLs do not have the feedback path enabled; therefore, they cannot be used for implementing the CPRI configuration.

In the deterministic latency ×4 option, up to four CPRI TX channels can be bundled in an ×4 group so that they all have the same TX uncertainty and just require one TX PLL to compensate for it. This is allowed in cases where the data rates are multiples of a single PLL output frequency; for example, 0.6144 Gbps, 1.228 Gbps, 2.4576 Gbps, and 4.9152 Gbps. For ×4 bundled channels to maintain PLL lock during auto-negotiation, the IP must use over-sampling (sending the same bit multiple times) to output lower auto-negotiated line rates. Do not use the hard 8B/10B for oversampled channels.

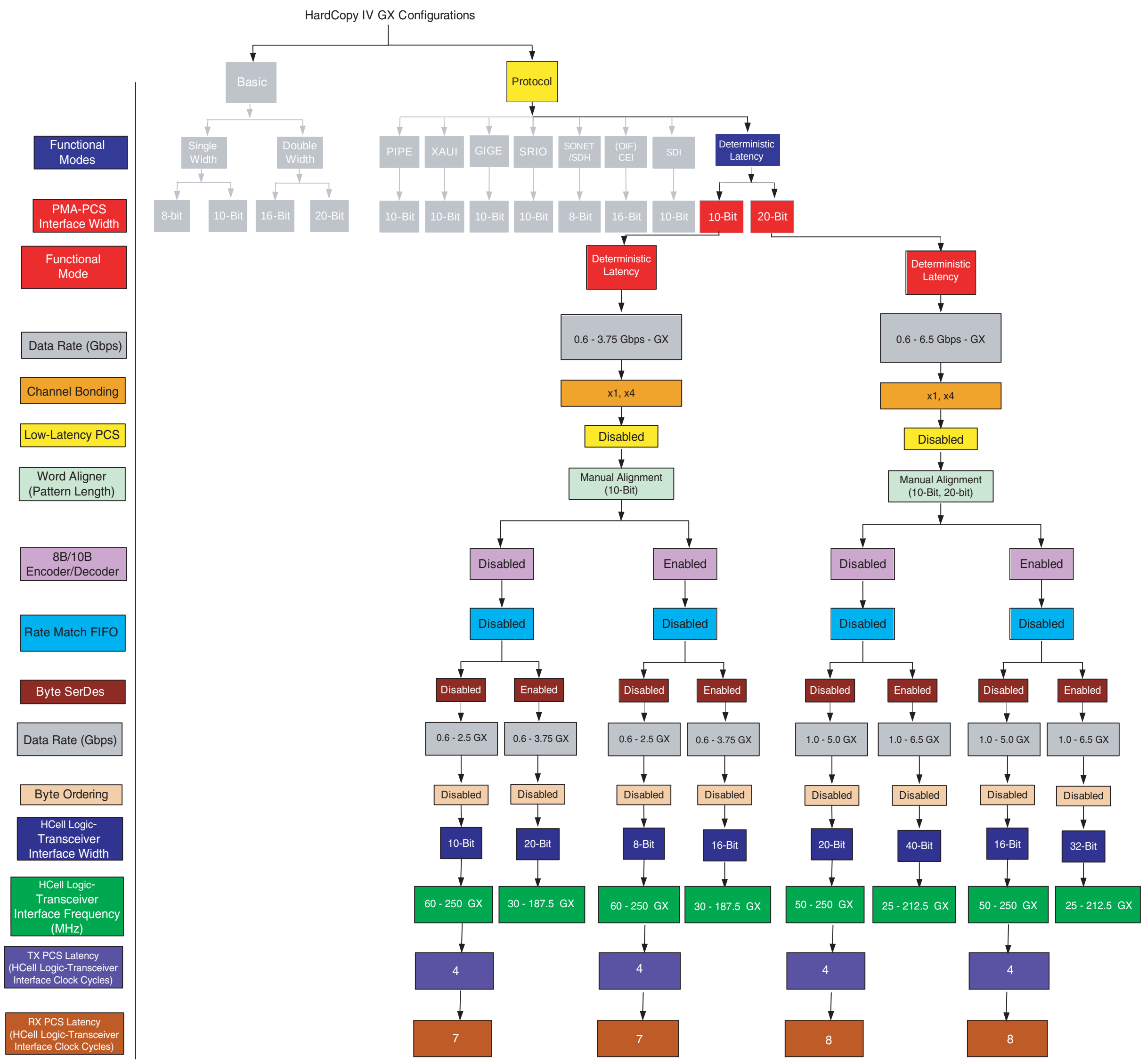
#### CPRI and OBSAI

You can use deterministic latency functional mode to implement protocols such as CPRI and OBSAI.

The CPRI interface defines a digital point-to-point interface between the Radio Equipment Control (REC) and the Radio Equipment (RE) allowing flexibility in either co-locating the REC and the RE or remote location of the RE.

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 1–43 shows the deterministic latency option.

**Figure 1–43. Deterministic Latency Option for HardCopy IV GX Devices**



To implement CPRI/OBSAI using deterministic latency mode, Altera recommends using configurations with the byte serializer/deserializer disabled.

Table 1–21 lists the PMA-PCS interface widths, CPRI, and OSBAI data rates in deterministic latency mode.

**Table 1–21. PMA-PCS Interface Widths, CPRI and OSBAI Data Rates in Deterministic latency Mode**

| Deterministic Latency Mode | Supported Data Rate Range | PMA-PCS Interface Width for CPRI & OBSAI | CPRI Data Rate (Gbps) | PCS Clock Frequency (MHz) | OBSAI Data Rate (Gbps) | PCS Clock Frequency (MHz) |
|---|---|---|---|---|---|---|
| Single-width mode | 600 Mbps to 3.75 Gbps | 8 bit, 10 bit | 0.6144 | 61.44 | 768 | 76.8 |
| | | | 1.2288 | 122.88 | 1.536 | 153.6 |
| | | | 2.4576 (1) | 245.76 | — | — |
| Double-width mode | > 1 Gbps | 16 bit, 20 bit | 3072 | 153.6 | 1.536 | 76.8 |
| | | 16 bit, 20 bit | 4915.2 (2), (3) | 245.76 | 3072 (3) | 153.6 |
| | | 32 bit, 40 bit | 6144 (2), (3), (4) | 307.2 | 6144 (3), (4) | 307.2 |

**Notes to Table 1–21:**

(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

HardCopy IV GX 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 HardCopy IV GX 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.

Version 2.0 of the PCIe specification provides implementation details for a PCIe-compliant physical layer device at both Gen1 (2.5 Gbps) and Gen2 (5 Gbps) signaling rates.

To implement a Version 2.0 PCIe-compliant PHY, you must configure the HardCopy IV GX transceivers in PCIe functional mode. HardCopy IV GX 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 HCell Logic using a soft IP. If you enable the PCIe hard IP blocks, the HardCopy IV GX transceivers interface with these hard IP blocks. Otherwise, the HardCopy IV GX transceivers interface with the HCell logic fabric.

You can configure the HardCopy IV GX transceivers in PCIe functional mode using one of the following two methods:

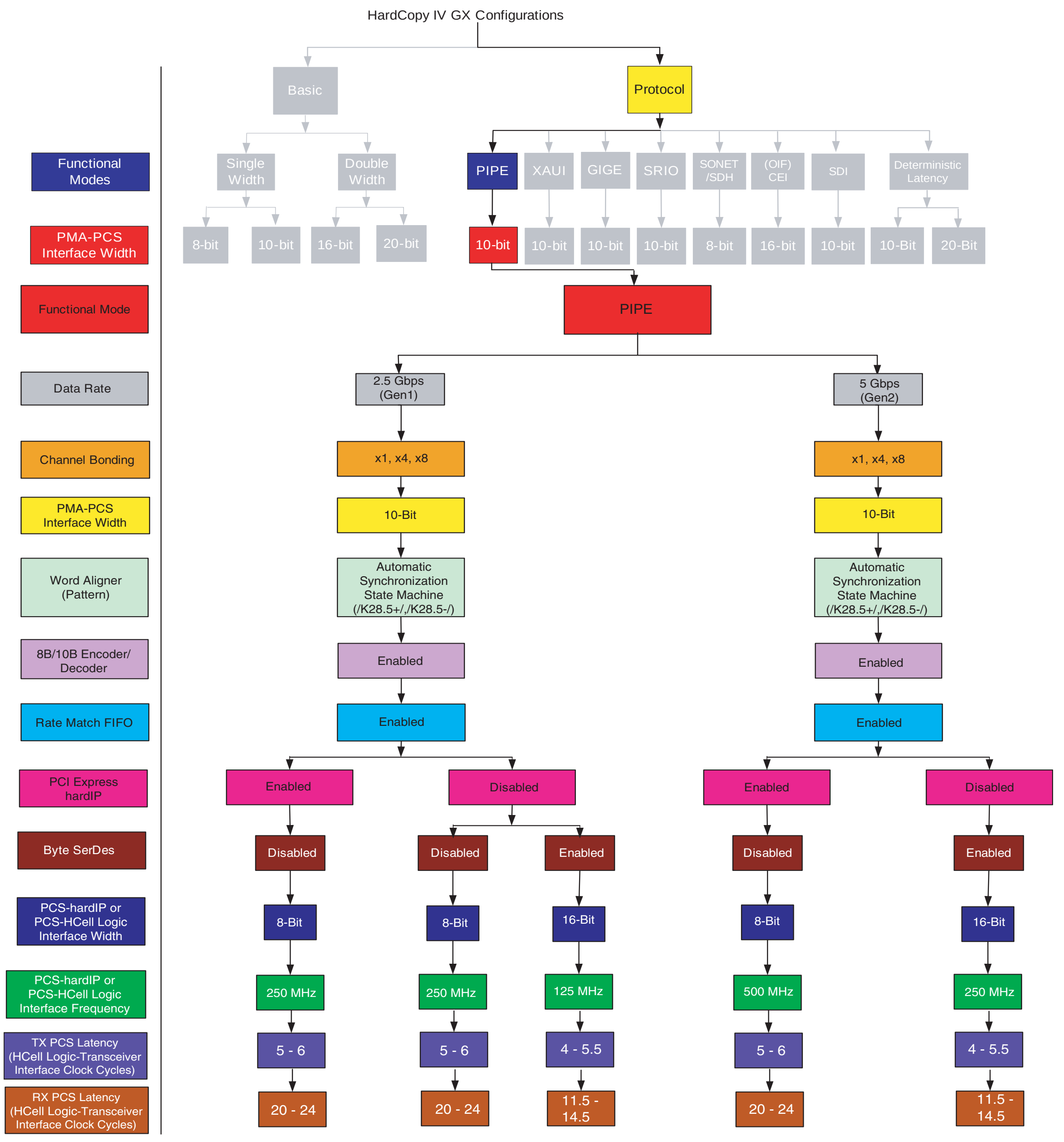
- ALTGX MegaWizard Plug-In Manager—if you do not use the PCIe hard IP block
- PCIe Compiler—if you use the PCIe hard IP block

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*.

HardCopy IV GX transceivers support ×1, ×4, and ×8 lane configurations in PCIe functional mode at both 2.5 Gbps and 5 Gbps data rates. In PCIe ×1 configuration, the PCS and PMA blocks of each channel are clocked and reset independently. PCIe ×4 and ×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 1–44 shows the HardCopy IV GX transceiver configurations allowed in PCIe functional mode.

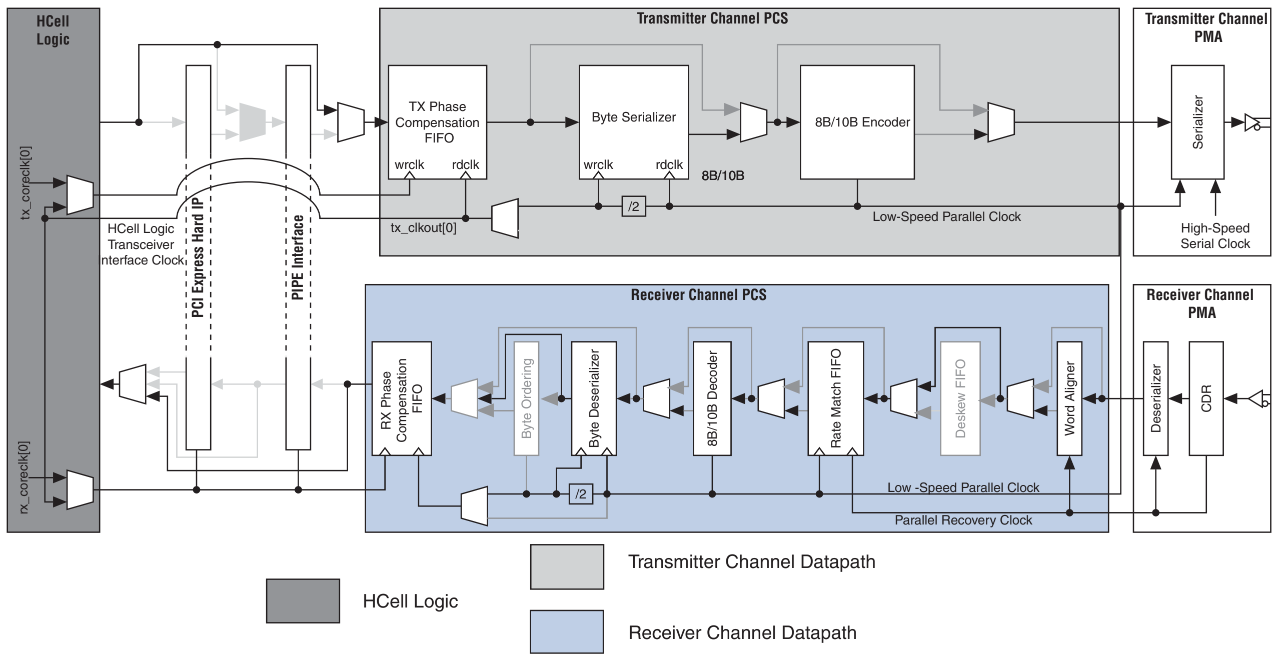
**Figure 1–44. Transceivers in PCIe Functional Mode for HardCopy IV GX Devices**



## PCIe Mode Datapath

Figure 1–45 shows the HardCopy IV GX transceiver datapath when configured in PCIe functional mode.

**Figure 1–45. Transceiver Datapath in PCIe ×1 Mode for HardCopy IV GX Devices**



For more information, refer to "Rate Match (Clock Rate Compensation) FIFO" on page 1–39.

Table 1–22 lists the transceiver datapath clock frequencies in PCIe functional mode configured using the ALTGX MegaWizard Plug-In Manager.

**Table 1–22. Transceiver Datapath Clock Frequencies in PCIe Mode for HardCopy IV GX Devices**

| Functional Mode | Data Rate | High-Speed Serial Clock Frequency | Parallel Recovered Clock and Low-Speed Parallel Clock Frequency | HCell Logic-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 | — *(1)* | 250 MHz |

**Note to Table 1–22:**

(1) In PCIe functional mode at the Gen2 (5 Gbps) data rate, you cannot bypass the byte serializer/deserializer.

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 *Stratix IV Transceiver Clocking* chapter in volume 2 of the *Stratix IV Device Handbook*.

Table 1–23 lists the transmitter and receiver datapaths in PCIe mode.

**Table 1–23. Datapaths in PCIe Mode**

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

Table 1–24 lists the features supported in PCIe functional mode for 2.5 Gbps and 5 Gbps data rate configurations.

**Table 1–24. Supported Features in PCIe Mode (Part 1 of 2)**

| Feature | 2.5 Gbps (Gen1) | 5 Gbps (Gen2) |
|---|---|---|
| ×1, ×4, ×8 link configurations | ✓ | ✓ |
| PCIe-compliant synchronization state machine | ✓ | ✓ |
| ±300 PPM (total 600 PPM) clock rate compensation | ✓ | ✓ |
| 8-bit HCell Logic-transceiver interface | ✓ | — |
| 16-bit HCell Logic-transceiver interface | ✓ | ✓ |
| Transmitter buffer electrical idle | ✓ | ✓ |
| Receiver Detection | ✓ | ✓ |
| 8B/10B encoder disparity control when transmitting compliance pattern | ✓ | ✓ |
| Power state management | ✓ | ✓ |
| Receiver status encoding | ✓ | ✓ |
| Dynamic switch between 2.5 Gbps and 5 Gbps signaling rate | — | ✓ |

**Table 1–24. Supported Features in PCIe Mode (Part 2 of 2)**

| Feature | 2.5 Gbps (Gen1) | 5 Gbps (Gen2) |
|---|---|---|
| Dynamically selectable transmitter margining for differential output voltage control | — | ✓ |
| Dynamically selectable transmitter buffer de-emphasis of -3.5 db and -6 dB | — | ✓ |

### 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 HCell 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

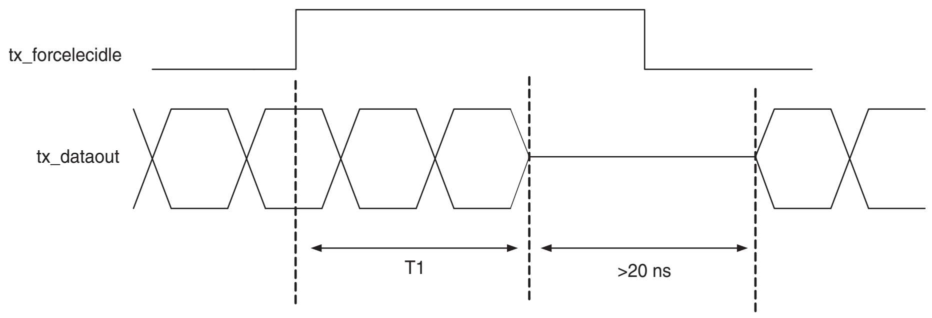
#### Transmitter Buffer Electrical Idle

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 1–46 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_forceelecidle` signal must be asserted high such that the transmitter buffer stays in electrical idle state for at least 20 ns is pending characterization.

**Figure 1–46. 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_forceelecidle` signal levels required in different PCIe power states, refer to Table 1–26 on page 1–73.

### 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 receiver detect circuitry in the transmitter PMA 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.

### 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 only the LSB of `tx_forcedispcompliance[1:0]` high in the same parallel clock cycle as /K28.5/D21.5/ of the compliance pattern on the `tx_datain` port.

Figure 1–47 and Figure 1–48 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 1–47. Compliance Pattern Transmission Support, 8-Bit Channel Width Configurations**

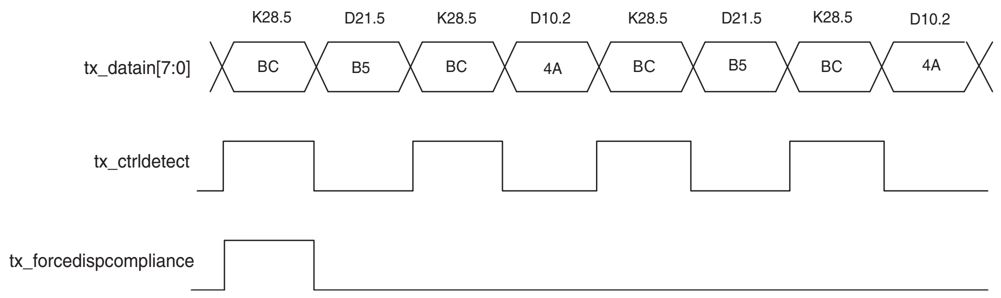
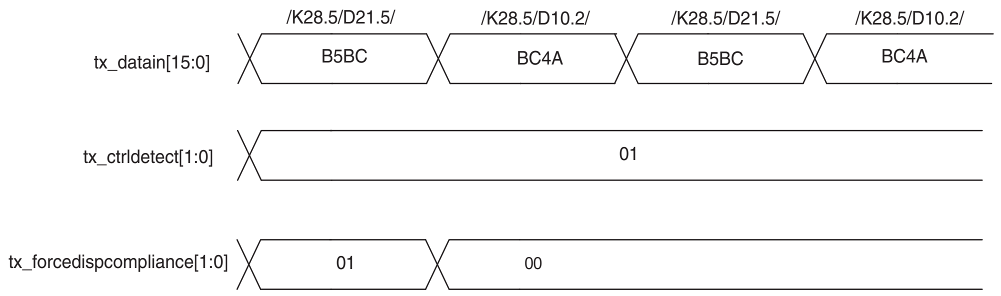


**Figure 1–48. Compliance Pattern Transmission Support, 16-Bit Wide Channel Configurations**



### 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 interface in HardCopy IV GX transceivers provides an input port, `powerdn[1:0]`, for each transceiver channel configured in PCIe mode. Table 1–25 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 1–25. 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. HardCopy IV GX transceivers do not implement these power saving measures except putting the transmitter buffer in electrical idle in the lower power states.

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_forceelecidle` signals appropriately in each power state to perform these functions. Table 1–26 lists the logic levels that the PHY-MAC layer must drive on the `tx_detectrxloopback` and `tx_forceelecidle` signals in each power state.

**Table 1–26. Logic Levels for tx_detectrxloopback and tx_forceelecidle in Different Power States**

| Power State | tx_detectrxloopback | tx_forceelecidle |
|---|---|---|
| P0 | 0: normal mode<br>1: datapath in loopback mode | 0: Must be de-asserted<br>1: Illegal mode |
| P0s | Don't care | 0: Illegal mode<br>1: Must be asserted in this state |
| P1 | 0: Electrical Idle<br>1: receiver detect | 0: Illegal mode<br>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 HCell logic fabric. The encoding of the status signals on `pipestatus[2:0]` is compliant with the PCIe specification and is listed in Table 1–27.

**Table 1–27. Encoding of the Status Signals on pipestatus[2:0]**

| pipestatus[2:0] | Description | Error Condition Priority |
|---|---|---|
| 3'b000 | Received data OK | — |
| 3'b001 | One SKP symbol added | 5 |
| 3'b010 | One SKP symbol deleted | 6 |
| 3'b011 | Receiver detected | — |
| 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 1–27 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 µs at Gen1 data rate and ~2 µs at Gen2 data rate).

If you have configured the HardCopy IV GX receiver CDR in Automatic Lock mode, the receiver cannot meet the PCIe specification of acquiring bit and byte synchronization within 4 µs (Gen1 data rate) or 2 µs (Gen2 data rate) due to the signal detect and PPM detector time. To meet this specification, each HardCopy IV GX 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 (×1, ×4, and ×8), each receiver channel PCS has an optional Electrical Idle Inference module designed to implement the electrical idle inference conditions specified in the 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 1–28 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_elecidleinfersel[2:0]` signal appropriately, as shown in Table 1–28.

**Table 1–28. Electrical Idle Inference Conditions**

| rx_elecidleinfersel[2:0] | LTSSM State | Gen1 (2.5 Gbps) | Gen2 (5 Gbps) |
|---|---|---|---|
| 3'b100 | L0 | Absence of skip ordered set in 128 µs window | Absence of skip ordered set in 128 µs window |
| 3'b101 | 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'b110 | 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'b111 | Loopback.Active (as slave) | Absence of an exit from Electrical Idle in 128 µs window | — |

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_elecidleinfersel[2:0] = 3'b0xx`, the Electrical Idle Inference block uses the EIOS detection from the Fast Recovery circuitry to drive the `pipeelecidle` signal.

If you do not select the **Enable electrical idle inference functionality** option in the ALTGX MegaWizard Plug-In Manager, the Electrical Idle Inference module is disabled. In this case, the `rx_signaldetect` signal from the signal detect circuitry in the receiver buffer is inverted and driven as the `pipeelecidle` signal.

### PCIe Gen2 (5 Gbps) Support

The PCIe functional mode supports the following additional features when configured for 5 Gbps data rate:

- Dynamic switching 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 HardCopy IV GX 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

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. HardCopy IV GX 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 1–29 lists the locations of the PCIe rateswitch controller and the PCIe clock switch circuitry in PCIe ×1, ×4, and ×8 modes.

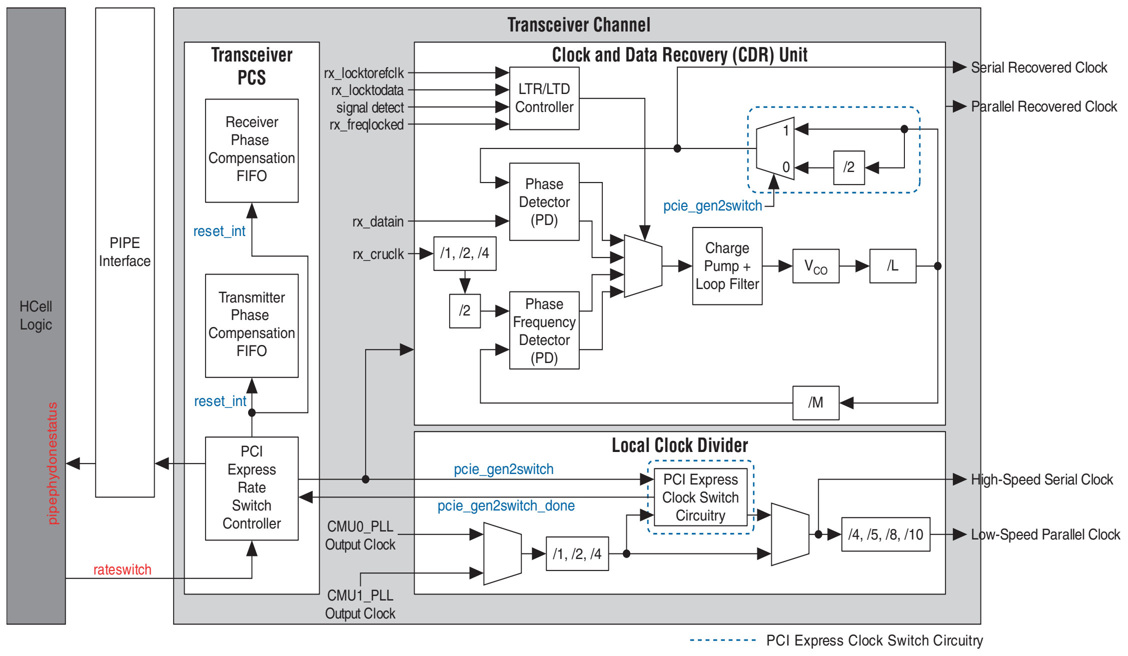
**Table 1–29. PCIe Rateswitch Controller and Clock Switch Circuitry**

| 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 1–49 shows the PCIe rateswitch circuitry in PCIe ×1 mode configured at Gen2 (5 Gbps) data rate.

**Figure 1–49. Dynamic Switch Signaling in PIPE ×1 Mode**



In PCIe ×1 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 1–30 lists the transceiver clock frequencies when switching between 2.5 Gbps and 5 Gbps signaling rates.

**Table 1–30. Transceiver Clock Frequencies Signaling Rates in PCIe ×1 Mode**

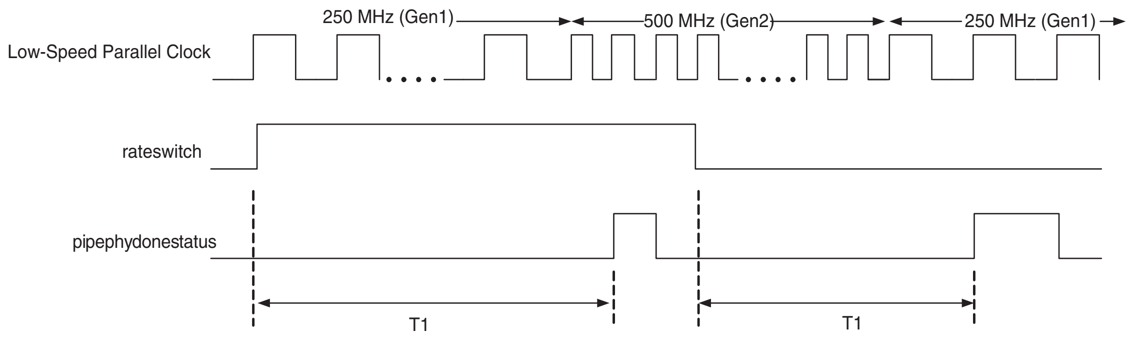
| Transceiver Clocks | Gen1 (2.5 Gbps) to Gen2 (5 Gbps) Switch (Low-to-High Transition on the rateswitch Signal) | Gen2 (5 Gbps) to Gen1 (2.5 Gbps) Switch (High-to-Low Transition on the rateswitch Signal) |
|---|---|---|
| High-Speed Serial Clock | 1.25 GHz to 2.5 GHz | 2.5 GHz to 1.25 GHz |
| Low-Speed Parallel Clock | 250 MHz to 500 MHz | 500 MHz to 250 MHz |
| Serial Recovered Clock | 1.25 GHz to 2.5 GHz | 2.5 GHz to 1.25 GHz |
| Parallel Recovered Clock | 250 MHz to 500 MHz | 500 MHz to 250 MHz |
| HCell 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 1–50 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 1–50. Low-Speed Parallel Clock Switching in PCIe ×1 Mode**

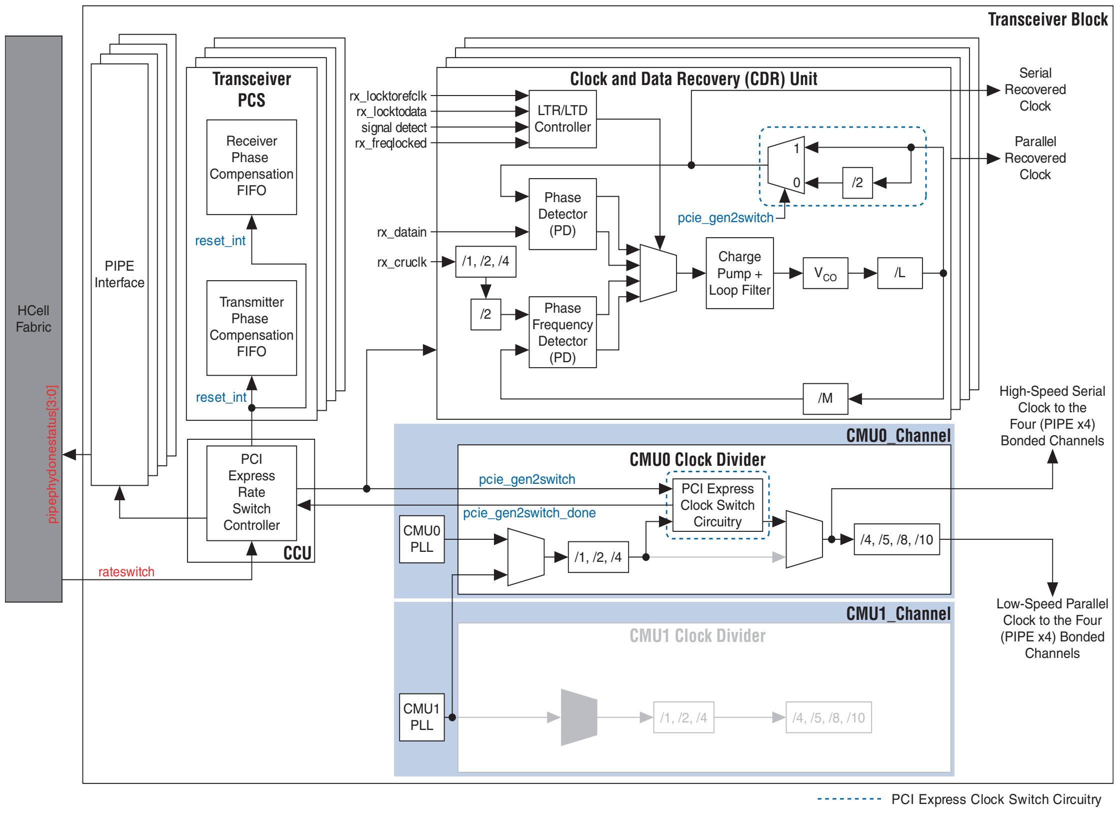


As a result of the signaling rateswitch between Gen1 (2.5 Gbps) and Gen2 (5 Gbps), the HCell fabric-transceiver interface clock switches between 125 MHz and 250 MHz. The HCell fabric-transceiver interface clock clocks the read side and write side of the transmitter phase compensation FIFO and the receiver phase compensation FIFO, respectively. It is also routed to the HCell 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 1–51 shows the PCIe rateswitch circuitry in PCIe ×4 mode configured at Gen2 (5 Gbps) data rate.

**Figure 1–51. 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 1–31 lists the transceiver clock frequencies when switching between the 2.5 Gbps and 5 Gbps signaling rates.

**Table 1–31. 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 1–31. Transceiver Clock Frequencies Signaling Rates in PCIe ×4 Mode (Part 2 of 2)**

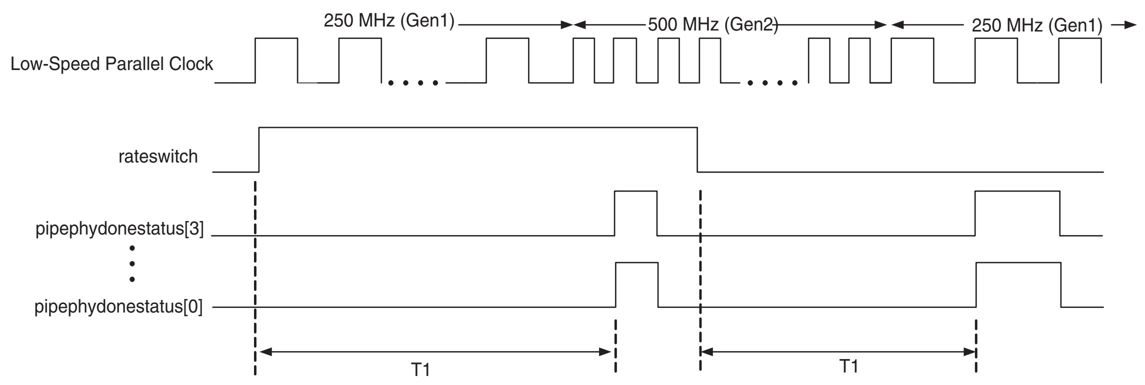
| Transceiver Clocks | Gen1 (2.5 Gbps) to Gen2 (5 Gbps) Switch (Low-to-High Transition on the rateswitch Signal) | Gen2 (5 Gbps) to Gen1 (2.5 Gbps) Switch (High-to-Low Transition on the rateswitch Signal) |
|---|---|---|
| Parallel Recovered Clock | 250 MHz to 500 MHz | 500 MHz to 250 MHz |
| HCell Fabric-Transceiver 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 1–52 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 1–52. Low-Speed Parallel Clock Switching in PCIe ×4 Mode**

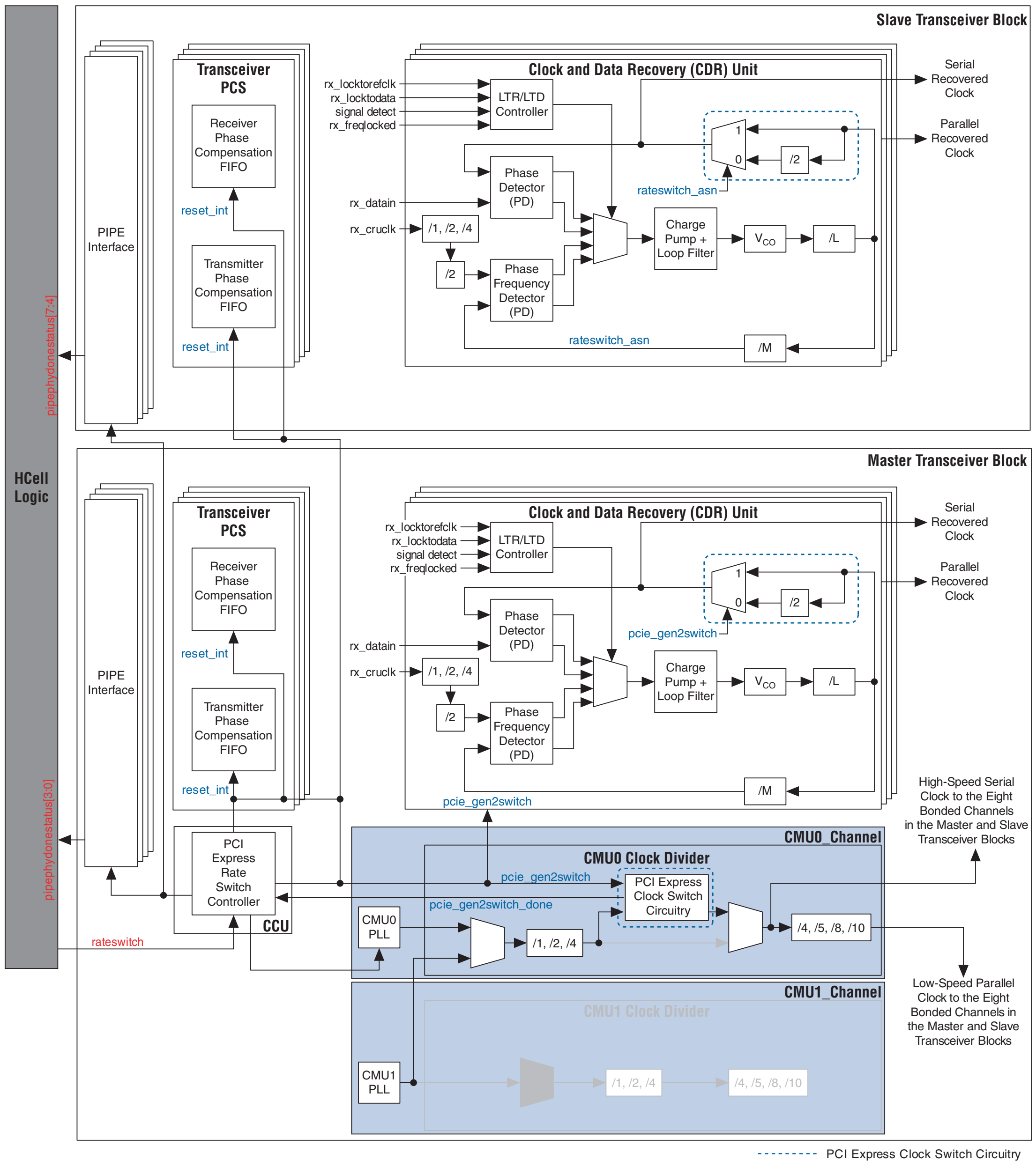


As a result of the signaling rateswitch between Gen1 (2.5 Gbps) and Gen2 (5 Gbps), the HCell fabric-transceiver interface clock switches between 125 MHz and 250 MHz. The HCell fabric-transceiver interface clock clocks the read side and write side of the transmitter phase compensation FIFO and the receiver phase compensation FIFO of all bonded channels, respectively. It is also routed to the HCell 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 1–53 shows the PCIe rateswitch circuitry in PCIe ×8 mode configured at Gen2 (5 Gbps) data rate.

**Figure 1–53. 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 1–32 lists the transceiver clock frequencies when switching between the 2.5 Gbps and 5 Gbps signaling rates.

**Table 1–32. Transceiver Clock Frequencies Signaling Rates in PCIe ×8 Mode**

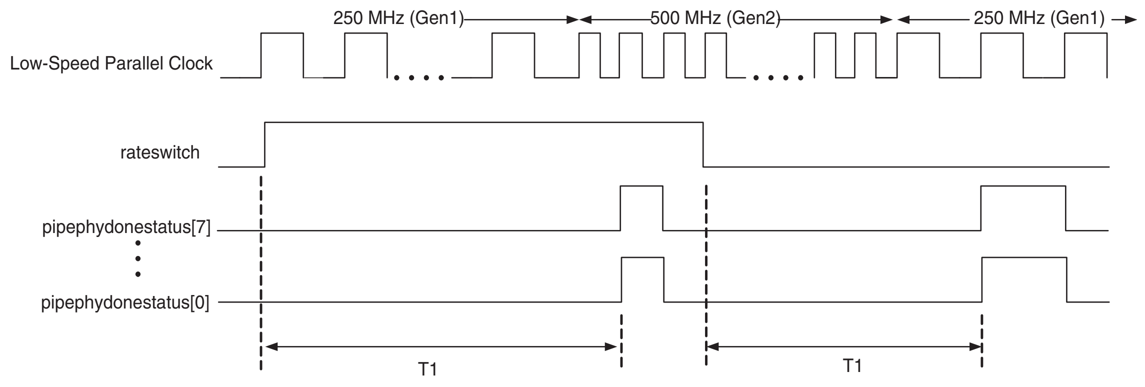
| Transceiver Clocks | Gen1 (2.5 Gbps) to Gen 2 (5 Gbps) Switch (Low-to-High Transition on the rateswitch Signal) | Gen2 (5 Gbps) to Gen1 (2.5 Gbps) Switch (High-to-Low Transition on the rateswitch Signal) |
|---|---|---|
| High-Speed Serial Clock | 1.25 GHz to 2.5 GHz | 2.5 GHz to 1.25 GHz |
| Low-Speed 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 |
| HCell 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 1–54 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 1–54. Low-Speed Parallel Clock Switching in PCIe ×8 Mode**



As a result of the signaling rateswitch between Gen1 (2.5 Gbps) and Gen2 (5 Gbps), the HCell Logic-transceiver interface clock switches between 125 MHz and 250 MHz. The HCell Logic-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 HCell Logic 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:

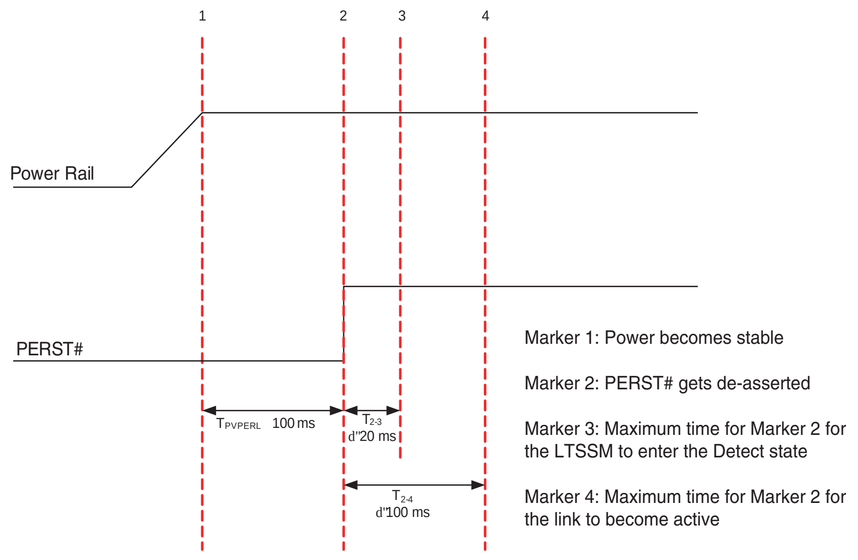
- Cold reset—fundamental reset after power up
- Warm reset—fundamental reset without removal and re-application of power
- Hot reset—In-band conventional reset initiated by the higher layer by setting the Hot Reset bit in the TS1 or TS2 training sequences

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 (`TPVPERL`) after the system power becomes stable in a cold reset situation. Additionally, all system components must enter the LTSSM Detect state within 20 ms and the link must become active within 100 ms after de-assertion of the `PERST#` signal. This implies that each PCIe system component must become active within 200 ms after the power becomes stable.

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

Figure 1–55 lists the PCIe cold reset timing requirements.

**Figure 1–55. PCIe Cold Reset Requirements**


1
2
3
4
Power Rail
PERST#
Marker 1: Power becomes stable
Marker 2: PERST# gets de-asserted
Marker 3: Maximum time for Marker 2 for the LTSSM to enter the Detect state
Marker 4: Maximum time for Marker 2 for the link to become active
TPVPERL 100 ms
T2-3 d"20 ms
T2-4 d"100 ms

The time taken by a PCIe port implemented using the HardCopy IV GX 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
- Delay phase with either no delay or 50 ms additional delay.
- 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 HardCopy IV GX device must complete the power-up sequence 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).

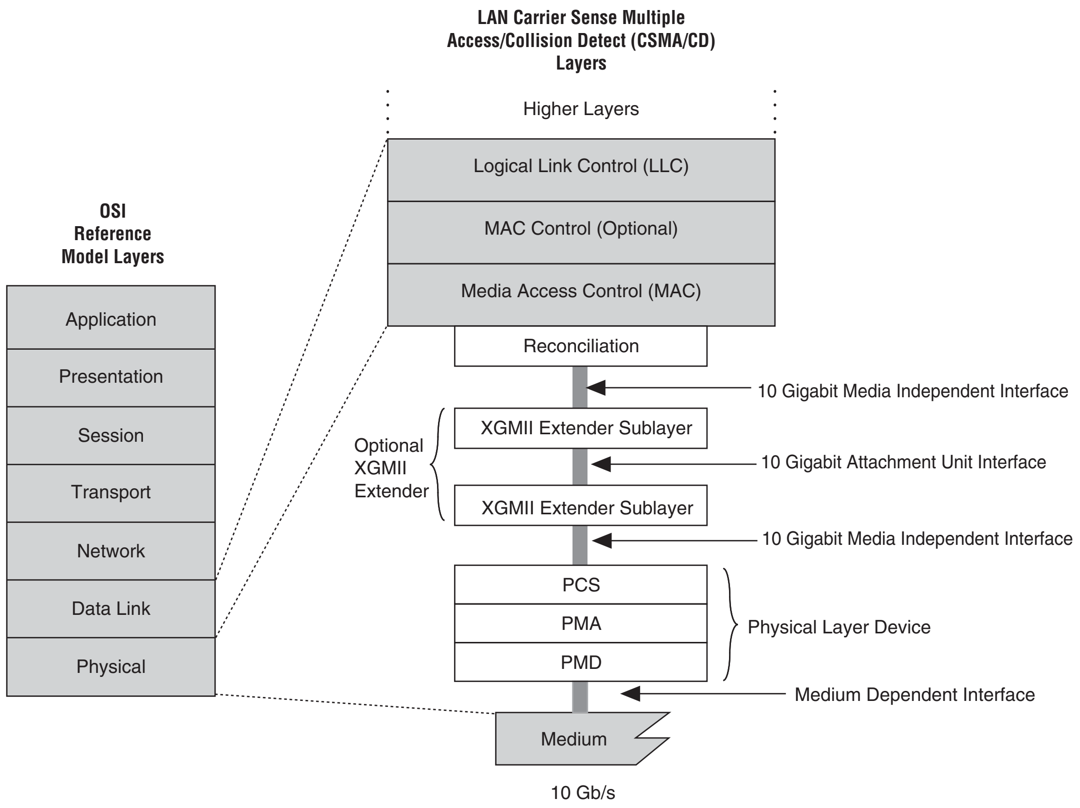
For more information about the PCIe hard IP block, refer to the *PCI Express Compiler User Guide*.

## 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.

Figure 1–56 shows the relationships between the XGMII and XAUI layers.

**Figure 1–56. XAUI and XGMII Layers**


LAN Carrier Sense Multiple Access/Collision Detect (CSMA/CD) Layers
Higher Layers
Logical Link Control (LLC)
MAC Control (Optional)
Media Access Control (MAC)
Reconciliation
10 Gigabit Media Independent Interface
Optional XGMII Extender
XGMII Extender Sublayer
10 Gigabit Attachment Unit Interface
XGMII Extender Sublayer
10 Gigabit Media Independent Interface
PCS
PMA
PMD
Physical Layer Device
Medium Dependent Interface
Medium
10 Gb/s
OSI Reference Model Layers
Application
Presentation
Session
Transport
Network
Data Link
Physical

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 HardCopy IV GX XAUI functional mode, the interface between the transceiver and HCell logic fabric is 64 bits wide (four channels of 16 bits each) at single data rate.

XAUI functions as a self-managed interface because code group synchronization, channel deskew, and clock domain decoupling is handled with no upper layer support requirements. This functionality is based on the PCS code groups that are used during the IPG time and idle periods. PCS code groups are mapped by the XGXS to XGMII characters, as specified in Table 1–33.

**Table 1–33. XGMII Character to PCS Code-Group Mapping**

| XGMII TXC | XGMII TXD *(1)* | PCD Code Group | Description |
|---|---|---|---|
| 0 | 00 through FF | Dxx,y | Normal data transmission |
| 1 | 07 | K28.0 or K28.3 or K28.5 | Idle in \|\|I\|\| |
| 1 | 07 | K28.5 | Idle in \|\|T\|\| |
| 1 | 9C | K28.4 | Sequence |
| 1 | FB | K27.7 | Start |
| 1 | FD | K29.7 | Terminate |
| 1 | FE | K30.7 | Error |
| 1 | Any other value | K30.7 | Invalid XGMII character |

**Note to Table 1–33:**

(1) The values in the XGMII TXD column are in hexadecimal.

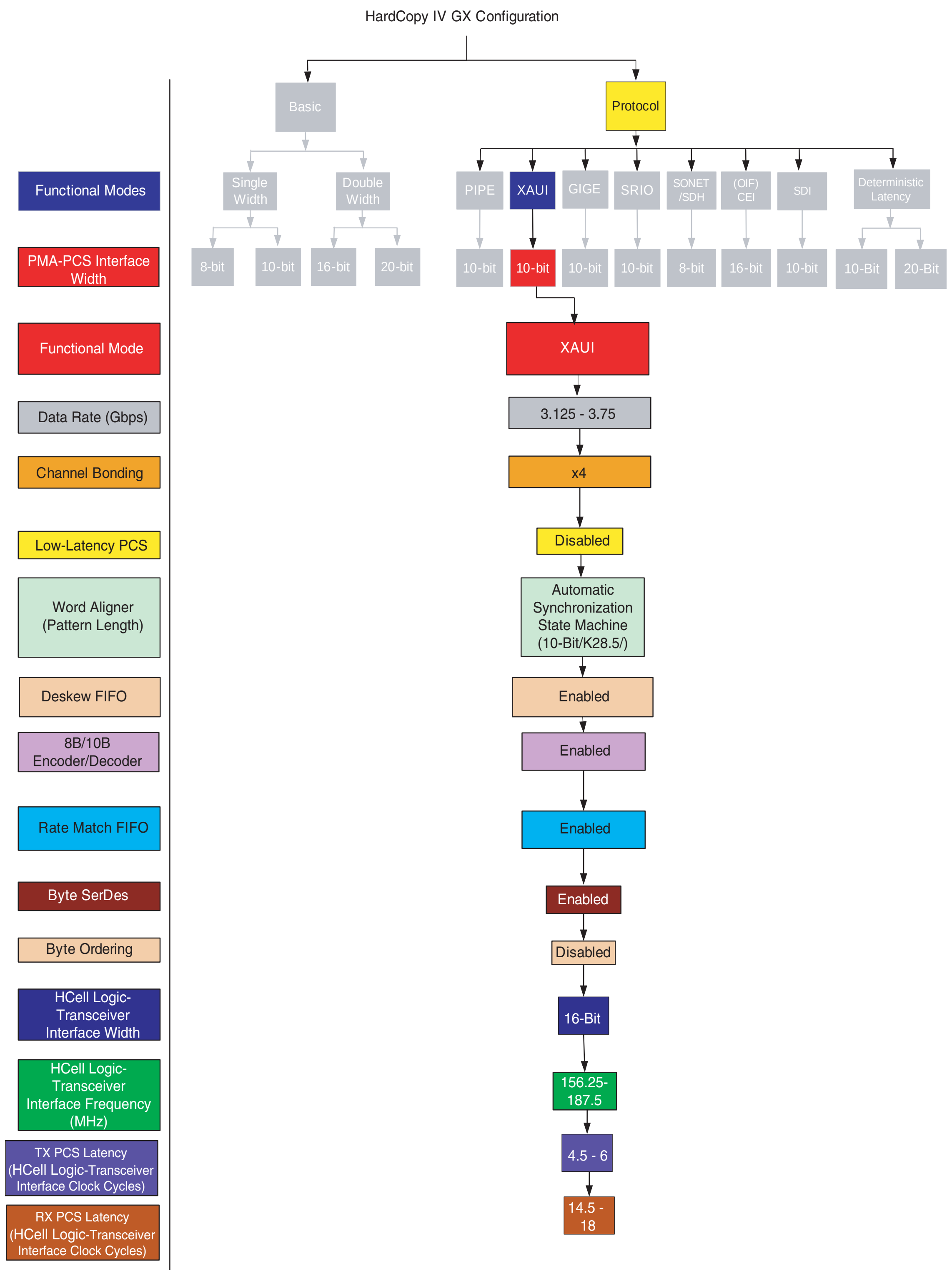
PCS code groups are sent using PCS ordered sets. PCS ordered sets consist of combinations of special and data code groups defined as a column of code groups. These ordered sets are composed of four code groups beginning in lane 0.

HardCopy IV GX 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 1–57 shows the XAUI mode configuration supported in HardCopy IV GX devices.

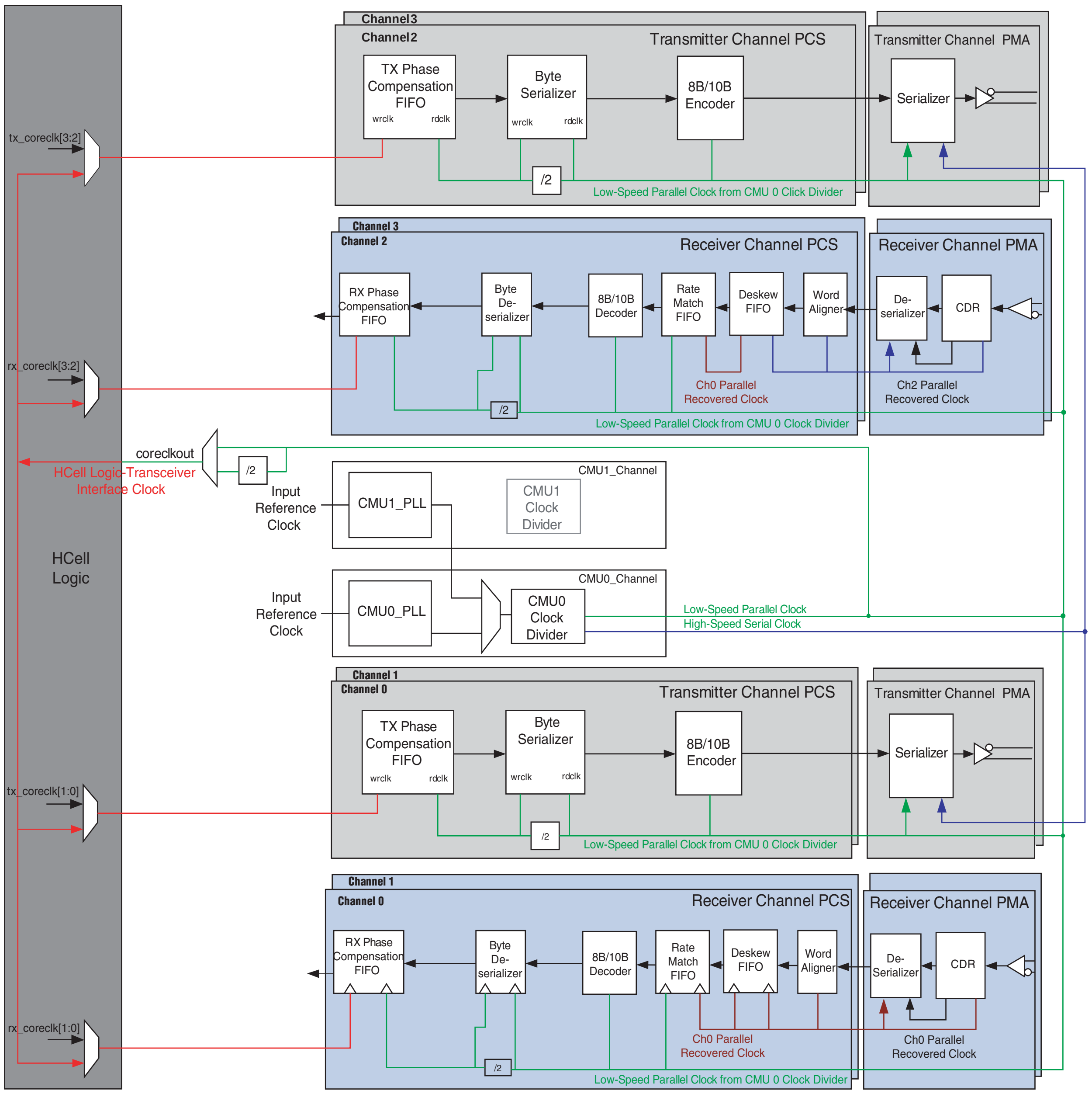
**Figure 1–57. XAUI Mode Configuration in HardCopy IV GX Devices**



### XAUI Mode Datapath

Figure 1–58 shows the ALTGX megafunction transceiver datapath when configured in XAUI mode.

**Figure 1–58. Transceiver Datapath in XAUI Mode**



### XGMII-To-PCS Code Conversion at the Transmitter

In XAUI mode, the 8B/10B encoder in the HardCopy IV GX 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.

For more information about the XGMII-to-PCS code conversion in XAUI functional mode, refer to the *XGMII-To-PCS Code Conversion at the Transmitter* section in the *Stratix IV Transceiver Architecture* chapter in volume 2 of the *Stratix IV GX Handbook*.

#### PCS-To-XGMII Code Conversion at the Receiver

In XAUI mode, the 8B/10B decoder in the HardCopy IV GX receiver datapath converts received PCS code groups into specific 8-bit XGMII codes.

For more information about the PCS-to-XGMII code conversion in XAUI functional mode, refer to the *PCS-To-XGMII Code Conversion at the Receiver* section in the *Stratix IV Transceiver Architecture* chapter in volume 2 of the *Stratix IV GX Handbook*.

#### 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.

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. 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.

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.

### 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.

Two flags, `rx_rmfifodatadeleted` and `rx_rmfifodatainserted`, indicating rate match FIFO deletion and insertion events, are forwarded to the HCell fabric.

For more information about the rate match FIFO in XAUI mode, refer to the *XAUI Mode* section of the *Stratix IV Device Handbook*.

## 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.

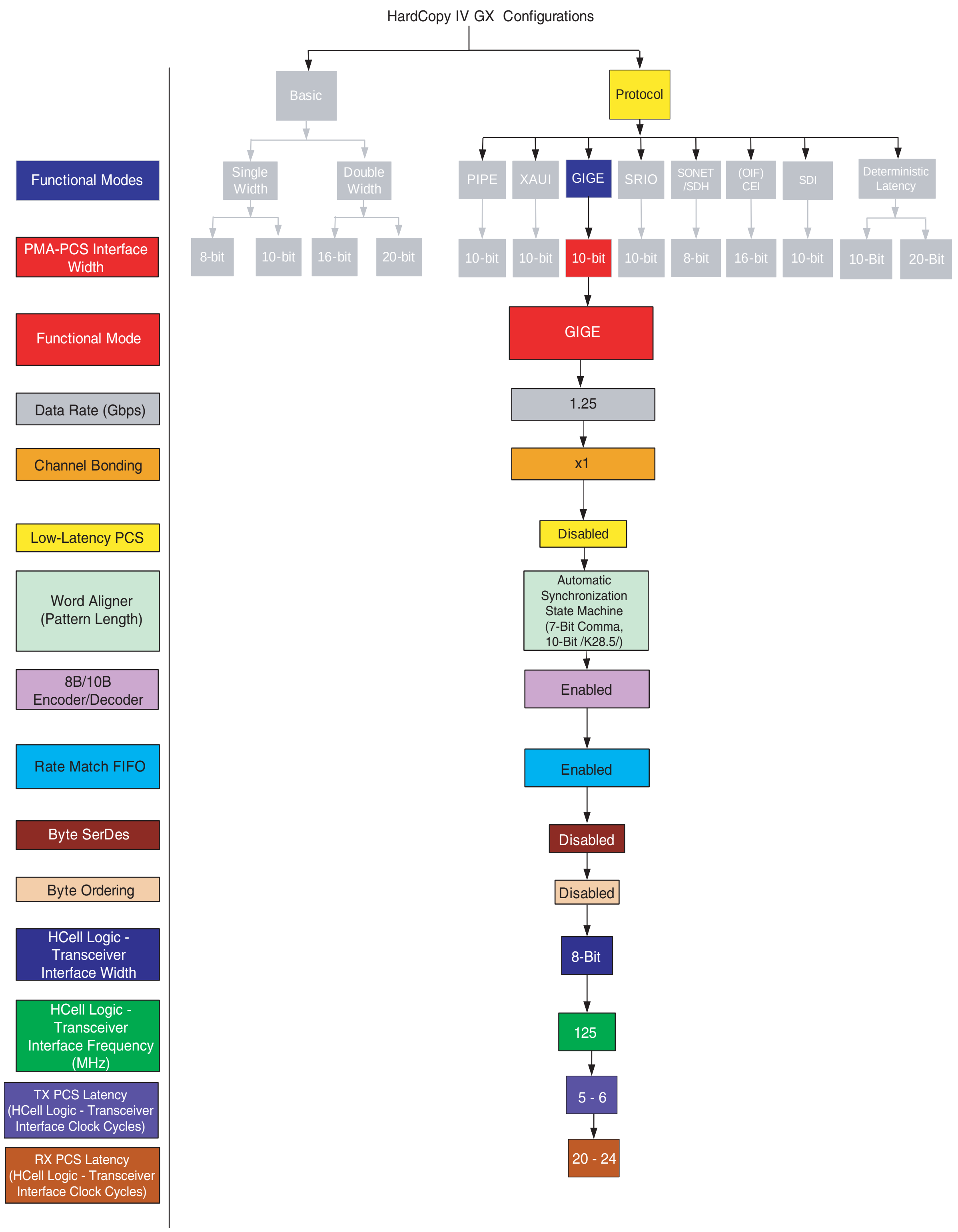
When configured in GIGE functional mode, HardCopy IV GX transceivers have built-in circuitry to support the following PCS and PMA functions, defined in the IEEE 802.3 specification:

- 8B/10B encoding and decoding
- Synchronization
- Upstream transmitter and local receiver clock frequency compensation (rate matching)
- Clock recovery from the encoded data forwarded by the receiver PMD
- Serialization and deserialization

HardCopy IV GX transceivers do not have built-in support for other PCS functions; for example, auto-negotiation state machine, collision-detect, and carrier-sense. If required, you must implement these functions in a PLD logic array or external circuits.

Figure 1–59 shows the GIGE mode configuration supported in HardCopy IV GX devices.

**Figure 1–59. GIGE Mode for HardCopy IV GX Devices**



### GIGE Mode Datapath

Figure 1–60 shows the transceiver datapath when configured in GIGE functional mode.

**Figure 1–60. GIGE Mode Datapath**

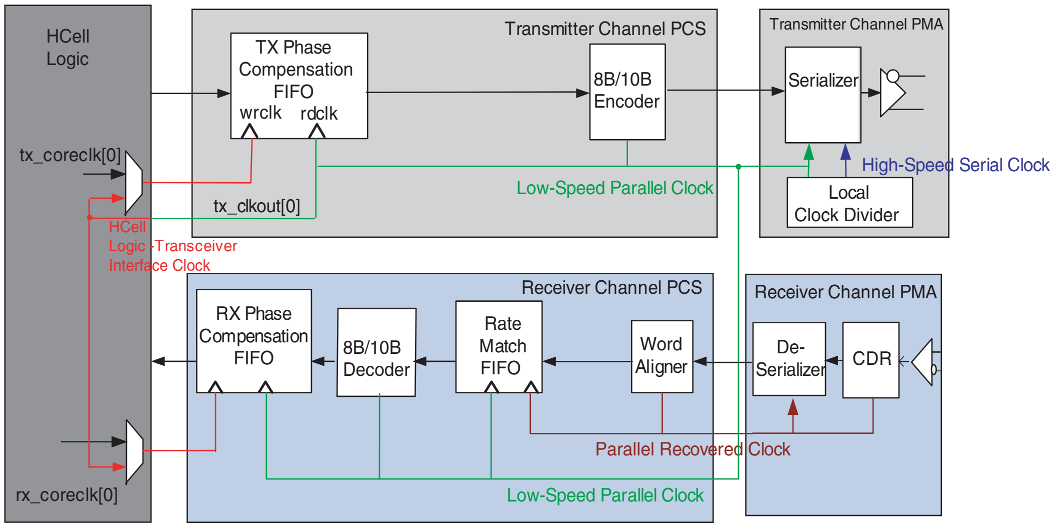


Table 1–34 lists the transceiver datapath clock frequencies in GIGE functional mode.

**Table 1–34. 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 | HCell 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 1–11.

#### 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.

For more information about the idle ordered-set generation in GIGE mode, refer to the *GIGE Mode* section in the *Stratix IV Transceiver Architecture* chapter in volume 2 of the *Stratix IV Device Handbook*.

#### 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 enters the loss of sync state.

#### 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.

For more information about the word aligner, refer to the *GIGE Mode* section in the *Stratix IV Transceiver Architecture* chapter in volume 2 of the *Stratix IV Device Handbook*.

#### 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.

Two flags, `rx_rmfifodatadeleted` and `rx_rmfifodatainserted`, indicating rate match FIFO deletion and insertion events, are forwarded to the HCell fabric.

For more information about the rate match FIFO, refer to the *GIGE Mode* section in the *Stratix IV Transceiver Architecture* chapter in volume 2 of the *Stratix IV Device Handbook*.

### SONET/SDH Mode

SONET/SDH defines various optical carrier (OC) sub-protocols for carrying signals of different capacities through a synchronous optical hierarchy.

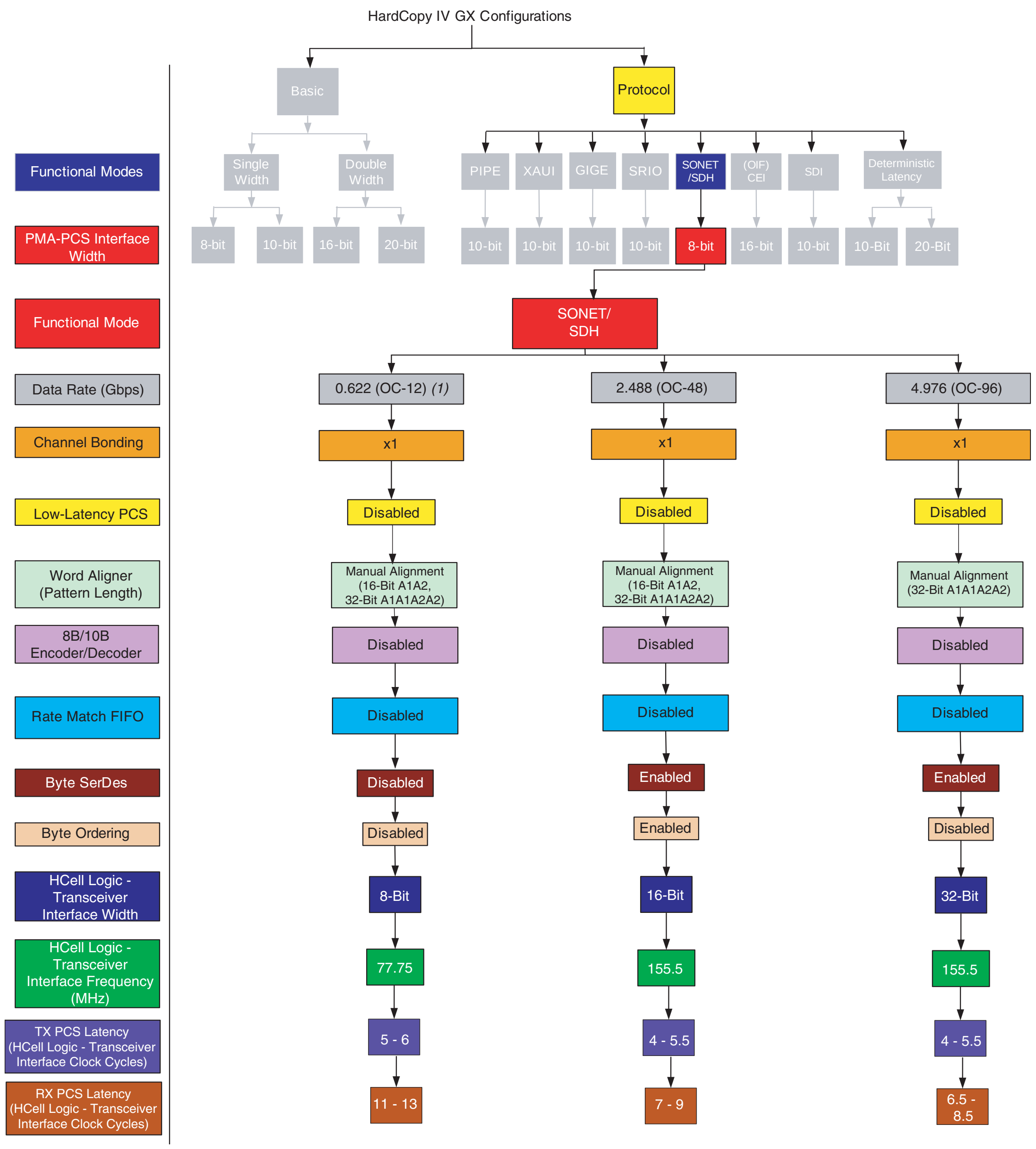
You can employ HardCopy IV GX transceivers as physical layer devices in a SONET/SDH system. These transceivers provide support for SONET/SDH protocol-specific functions and electrical features.

HardCopy IV GX transceivers are designed to support the following three SONET/SDH sub-protocols:

- OC-12 at 622 Mbps with 8-bit channel width
- OC-48 at 2488.32 Mbps with 16-bit channel width
- OC-96 at 4976 Mbps with 32-bit channel width

Figure 1–61 shows SONET/SDH mode configurations supported in HardCopy IV GX devices.

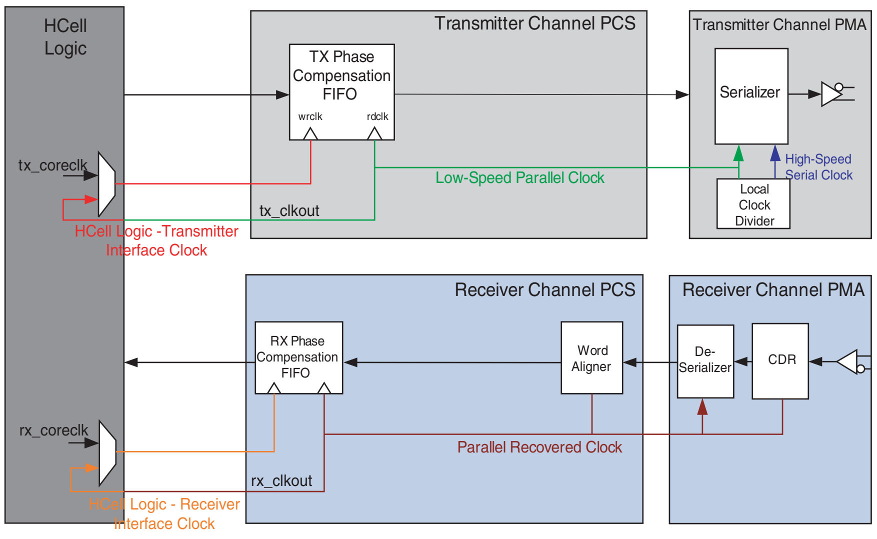
**Figure 1–61. SONET/SDH Mode Configurations for HardCopy IV GX Devices**



### SONET/SDH OC-12 Datapath

Figure 1–62 shows the transceiver datapath when configured in SONET/SDH OC-12 mode.

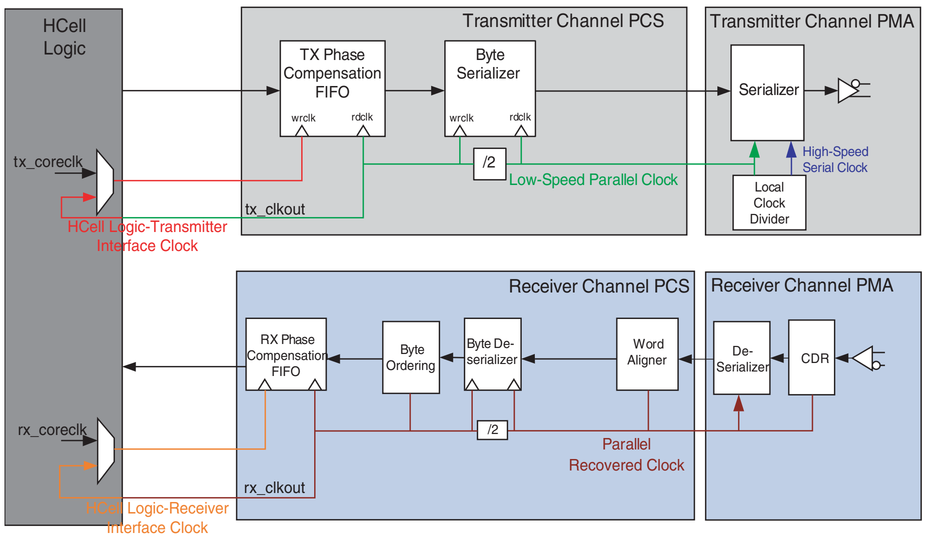
**Figure 1–62. SONET/SDH OC-12 Datapath**



### SONET/SDH OC-48 Datapath

Figure 1–63 shows the transceiver datapath when configured in SONET/SDH OC-48 mode.

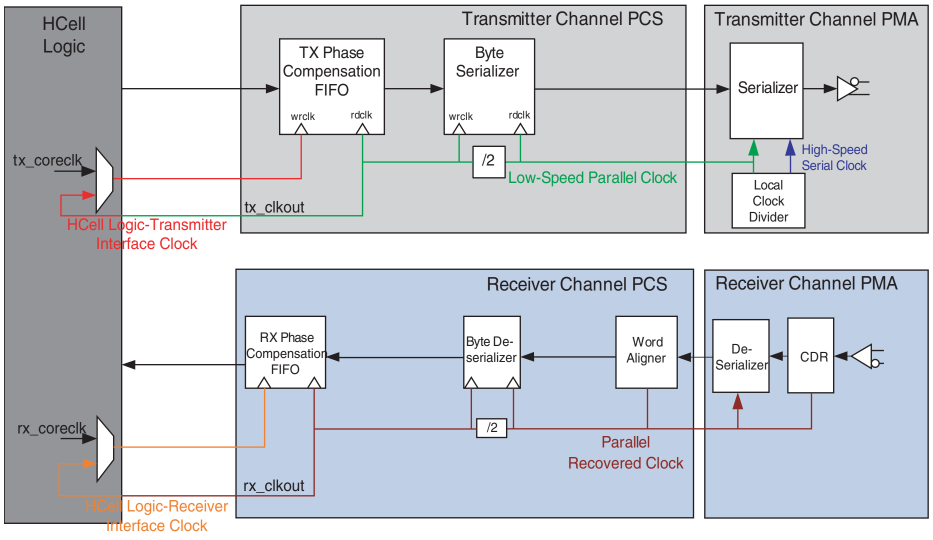
**Figure 1–63. SONET/SDH OC-48 Datapath**



### SONET/SDH OC-96 Datapath

Figure 1–64 shows the transceiver datapath when configured in SONET/SDH OC-96 mode.

**Figure 1–64. 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 1–35 on page 1–99 lists the correct word aligner settings for each bit transmission order.

### Word Alignment

In OC-12 and OC-48 configurations, you can configure the word aligner to either align to a 16-bit A1A2 pattern or a 32-bit A1A1A2A2 pattern. This is controlled by the `rx_a1a2size` input port to the transceiver.

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.

Table 1–35 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 1–35. 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) |

### 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 OC-48 byte serializer converts 16-bit data words from the HCell 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 HCell fabric at half the rate.

The OC-96 byte serializer converts 32-bit data words from the HCell 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 HCell fabric at half the rate.

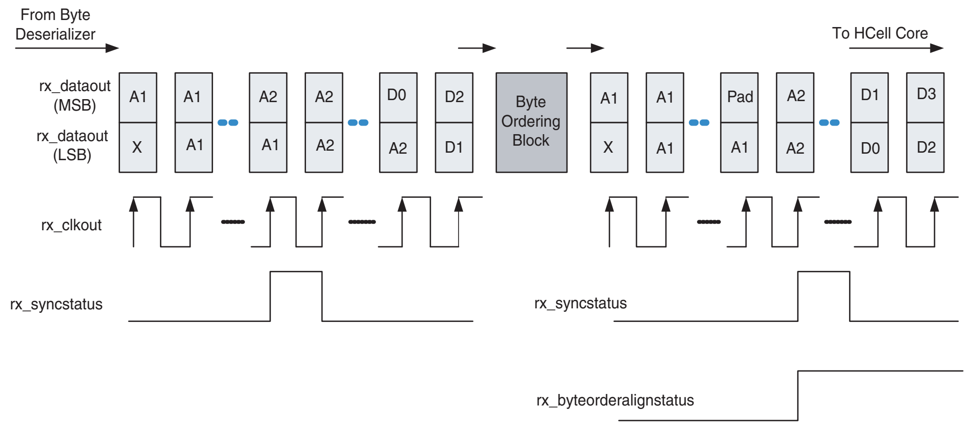
### OC-48 Byte Ordering

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.

In automatic mode, the byte ordering block is triggered by the rising edge of the `rx_syncstatus` signal. Figure 1–65 shows the byte ordering Automatic mode.

**Figure 1–65. OC-48 Byte Ordering in Automatic Mode**



## SDI Mode

You can configure HardCopy IV GX transceivers in HD-SDI or 3G-SDI configuration using the ALTGX MegaWizard Plug-In Manager.

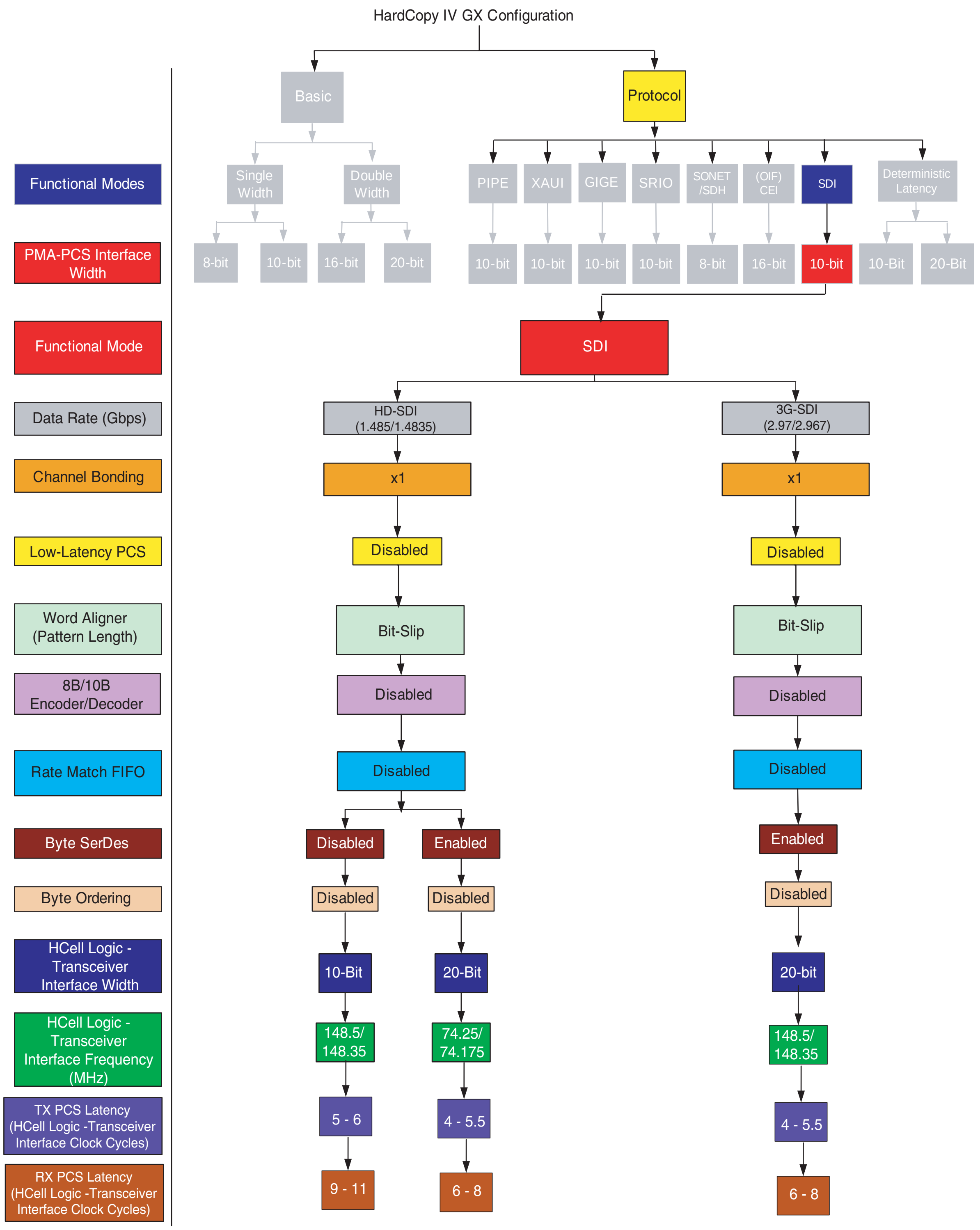
Table 1–36 lists the ALTGX configurations supported by HardCopy IV GX transceivers in SDI mode.

**Table 1–36. ALTGX Configurations in SDI Mode**

| SMPTE Standard | Configuration | Data Rate (Mbps) | REFCLK Frequencies (MHz) | HCell Logic-Transceiver Interface Width | Byte SERDES Usage |
|---|---|---|---|---|---|
| 292M | HD | 1483.5 | 74.175 | 20 bit | Used |
| | | | 148.35 | 10 bit | Not used |
| | | 1485 | 74.25 | 20 bit | Used |
| | | | 148.5 | 10 bit | Not used |
| 424M | 3G | 2967 | 148.35 | Only 20-bit interface allowed in 3G | Used |
| | | | 296.7 | | |
| | | 2970 | 148.5 | | |
| | | | 297 | | |

Figure 1–66 shows SDI mode configurations supported in HardCopy IV GX devices.

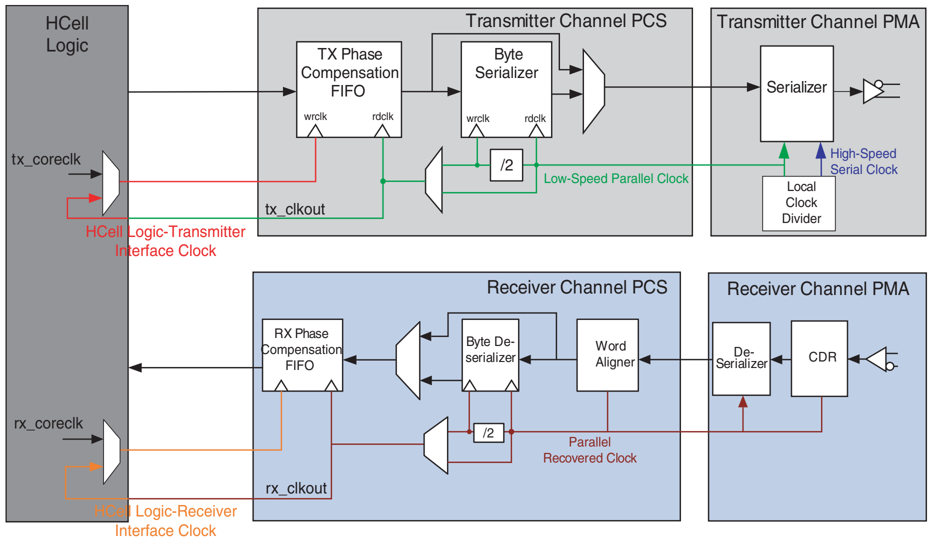
**Figure 1–66. SDI Mode for HardCopy IV GX Devices**



### SDI Mode Datapath

Figure 1–67 shows the transceiver datapath when configured in SDI mode.

**Figure 1–67. SDI Mode Datapath**



#### Transmitter Datapath

The transmitter datapath, in HD-SDI configuration with 10-bit wide HCell 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 HCell 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 HCell 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 HCell logic array.

#### Receiver Word Alignment and Framing

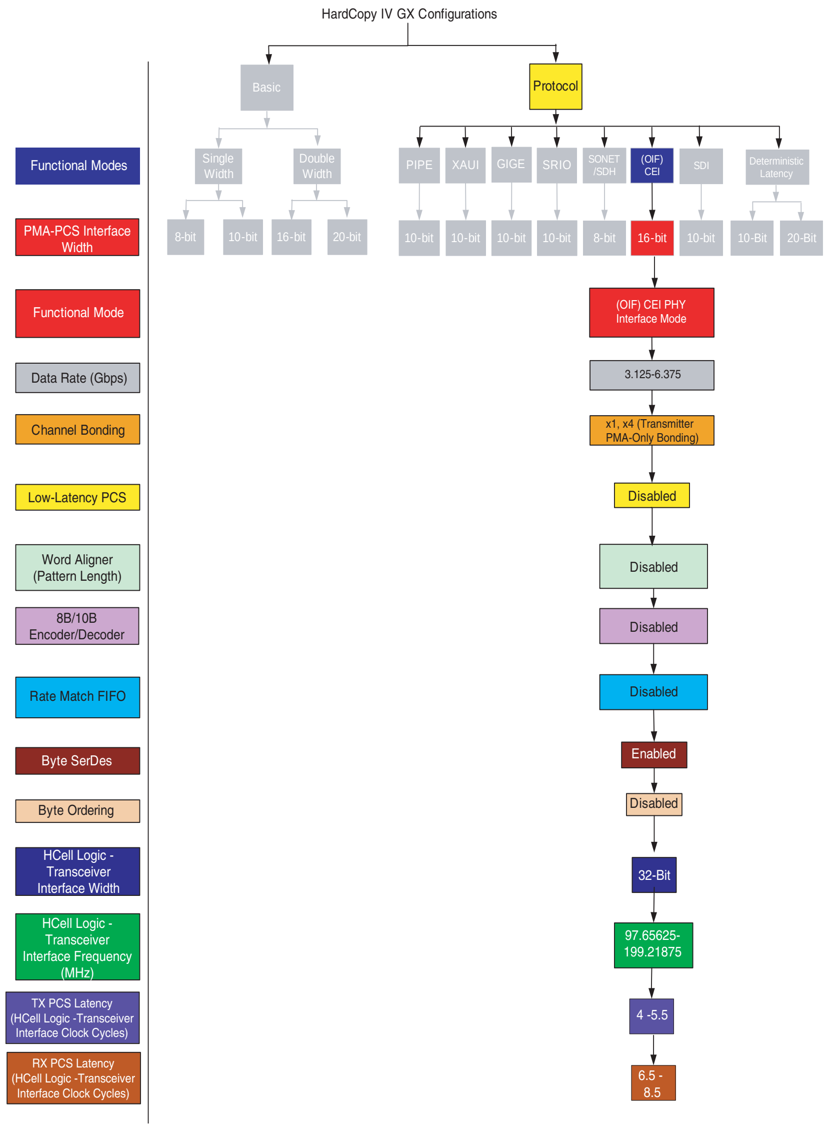
In SDI systems, the word aligner in the receiver datapath is not useful because word alignment and framing happens after de-scrambling. Altera recommends driving the ALTGX megafunction `rx_bitslip` signal low to avoid having the word aligner insert bits in the received data stream.

## (OIF) CEI PHY Interface Mode

HardCopy IV GX transceivers support a data rate between 4.976 Gbps and 6.375 Gbps in (OIF) CEI PHY interface mode.

Figure 1–68 shows (OIF) CEI PHY interface mode configurations supported in HardCopy IV GX devices.

**Figure 1–68. (OIF) CEI PHY Interface Mode for HardCopy IV GX Devices**



### (OIF) CEI PHY Interface Mode Datapath

Figure 1–69 shows the ALTGX megafunction transceiver datapath when configured in (OIF) CEI PHY interface mode.

**Figure 1–69. (OIF) CEI PHY Interface Mode Datapath**

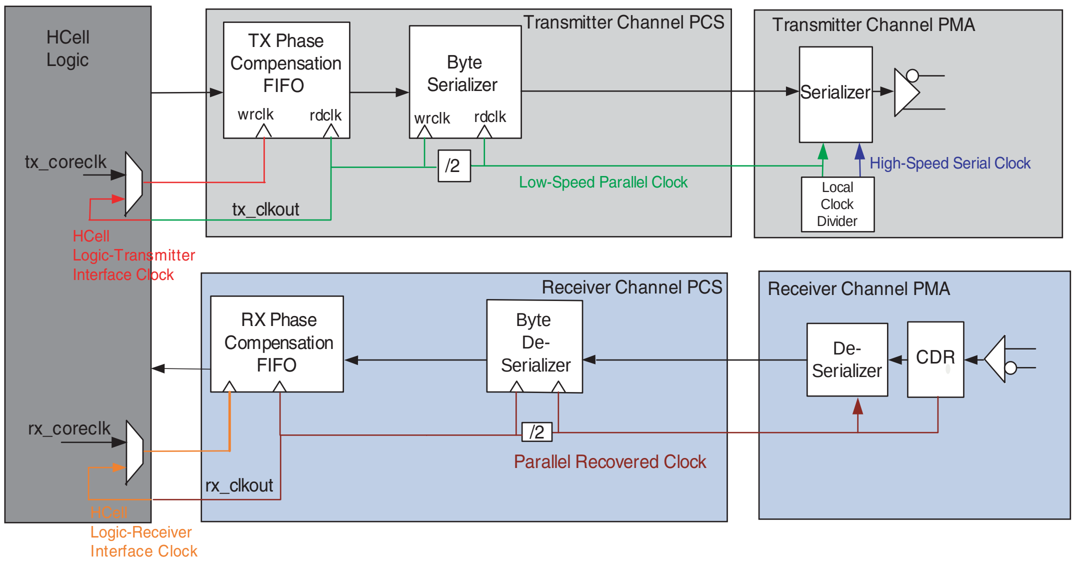
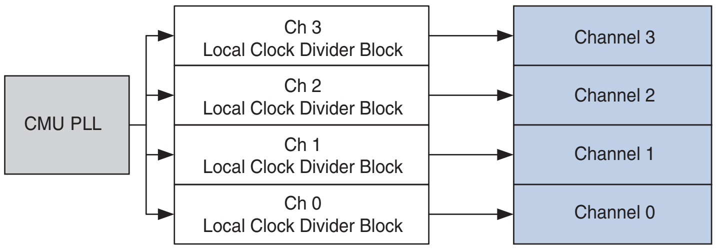


Figure 1–70 shows transceiver clocking in (OIF) CEI PHY interface mode.

**Figure 1–70. Transceiver Clocking in (OIF) CEI PHY Interface Mode *(Note 1)***


CMU PLL
Ch 3 Local Clock Divider Block
Ch 2 Local Clock Divider Block
Ch 1 Local Clock Divider Block
Ch 0 Local Clock Divider Block
Channel 3
Channel 2
Channel 1
Channel 0

**Note to Figure 1–70:**

(1) Transceiver Block Clocking with the **Use central clock divider to improve transmitter jitter** option disabled

## Serial RapidIO Mode

The Serial RapidIO physical layer specification defines three line rates:

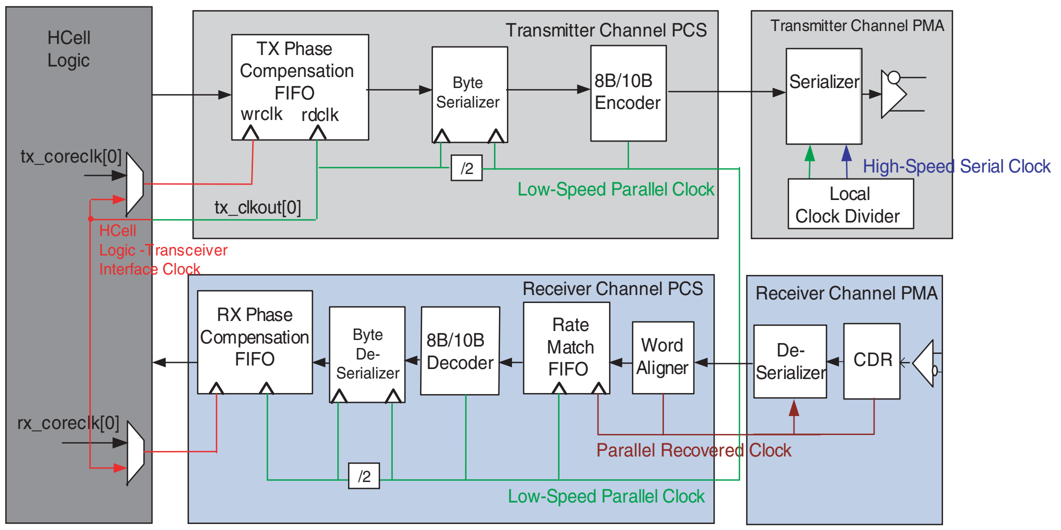
- 1.25 Gbps
- 2.5 Gbps
- 3.125 Gbps

It also defines two link widths—single-lane (1×) and bonded four-lane (4×) at each line rate.

HardCopy IV GX transceivers support only single-lane (1×) configuration at all three line rates. You can instantiate four 1× channels configured in Serial RapidIO mode 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 1–71 shows the ALTGX transceiver datapath when configured in Serial RapidIO mode.

**Figure 1–71. Serial RapidIO Mode Datapath**



HardCopy IV GX 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

HardCopy IV GX 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.

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 1–37 lists the ALTGX megafunction synchronization state machine parameters when configured in Serial RapidIO mode.

**Table 1–37. 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 |

### 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.

The 8B/10B encoder/decoder is always enabled in Serial RapidIO mode.

The rate matcher is an optional block available for selection in the Serial RapidIO functional mode. However, this block is not fully compliant to the Serial RapidIO 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 more information about the rate match FIFO, refer to the *Serial Rapid IO Mode* section in the of *Stratix IV Transceiver Architecture* chapter in volume 2 of the *Stratix IV Device Handbook*.

## Basic (PMA Direct) Functional Mode

In Basic (PMA Direct) functional mode, the HardCopy IV GX transceiver datapath contains only PMA blocks. Parallel data is transferred directly between the HCell 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 HCell 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.

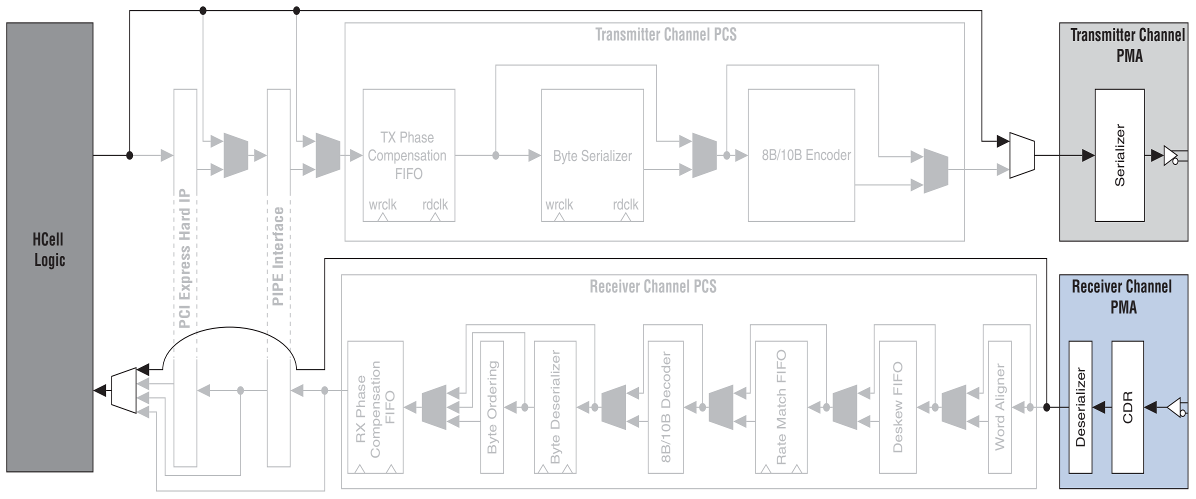
Use dynamic reconfiguration to dynamically reconfigure the various PMA controls to tailor the transceivers in PMA direct drive mode for a particular application.

For more information, refer to the *Dynamic Reconfiguration in HardCopy IV GX Devices* chapter. For more information about the reset sequence to follow in PMA-Direct mode, refer to the *Reset Control and Power Down* chapter in volume 2 of the *Stratix IV Device Handbook*.

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

Figure 1–72 shows the HardCopy IV GX transceiver configured in Basic (PMA Direct) functional mode. The grayed out blocks indicate areas that are not active in this mode.

**Figure 1–72. Transceiver Configured in Basic (PMA Direct) Mode in HardCopy IV GX *(Note 1)***



**Note to Figure 1–72:**

(1) The grayed out blocks shown in Figure 1–72 are not available in the CMU channels. Therefore, you can configure the CMU channels 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

Table 1–38 lists the HardCopy IV GX HCell-Core interface widths and data rates supported in Basic (PMA Direct) ×1 and ×N single-width and double-width modes.

**Table 1–38. HCell Logic-PMA Interface Widths and Data Rates Supported in Basic (PMA Direct) ×1 and ×N Single-Width and Double-Width Modes for HardCopy IV GX Devices**

| Basic (PMA Direct) Functional Mode | HCell Logic-PMA Interface Width | Supported Data Rate Range |
|---|---|---|
| | | C3/I3 Speed Grade |
| ×1/×N Single-width mode | 8 bit | 0.6Gbps to 2.6Gbps |
| | 10 bit | 0.6Gbps to 3.25Gbps |
| ×1/×N Double-width mode | 16 bit | 1.0Gbps to 5.2Gbps |
| | 20 bit | 1.0Gbps to 6.5Gbps |

For information about routing the clocks to transceiver channels in Basic (PMA Direct) ×1 mode, refer to the *Stratix IV Transceiver Clocking* chapter in volume 2 of the *Stratix IV Device Handbook*.

For more information about combining multiple transceiver channels, refer to the *Configuring Multiple Protocols and Data Rates in a Transceiver Block* chapter in volume 2 of the *Stratix IV Device Handbook*.

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

HardCopy IV GX 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 1–108—available in all functional modes except PCIe mode
- "Parallel Loopback" on page 1–109—available in either single- or double-width modes.
- "Reverse Serial Loopback" on page 1–111—available in Basic mode only
- "Reverse Serial Pre-CDR Loopback" on page 1–111—available in Basic mode only
- "PCIe Reverse Parallel Loopback" on page 1–112—supported in PCIe protocol only

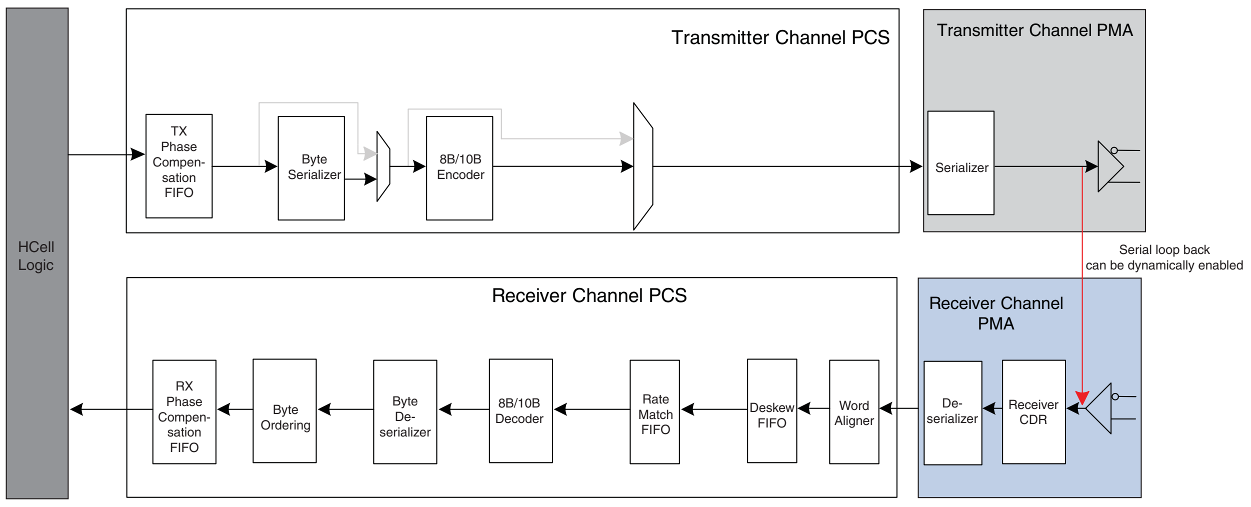
### Serial Loopback

The **serial loopback** option is available for all functional modes except PCIe mode. Figure 1–73 shows the datapath for serial loopback. The data from the HCell Logic passes through the transmitter channel and gets looped back to the receiver channel, bypassing the receiver buffer. The received data is available to the HCell logic for verification. Using this option, you can check the operation of all enabled PCS and PMA functional blocks in the transmitter and receiver channels.

When you enable the **serial loopback** option, the ALTGX MegaWizard Plug-In Manager provides the `rx_seriallpbken` port to dynamically enable serial loopback on a channel-by-channel basis. To use this option, you must provide an alignment pattern for the word aligner to enable the receiver channel to retrieve the byte boundary.

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

**Figure 1–73. 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- or double-width configuration.

The BIST pattern generator and pattern verifier are located near the HCell Logic 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.

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 1–74. The incremental data pattern is not available to the HCell Logic for verification.

**Figure 1–74. Enabled PCS Functional Blocks in Parallel Loopback**

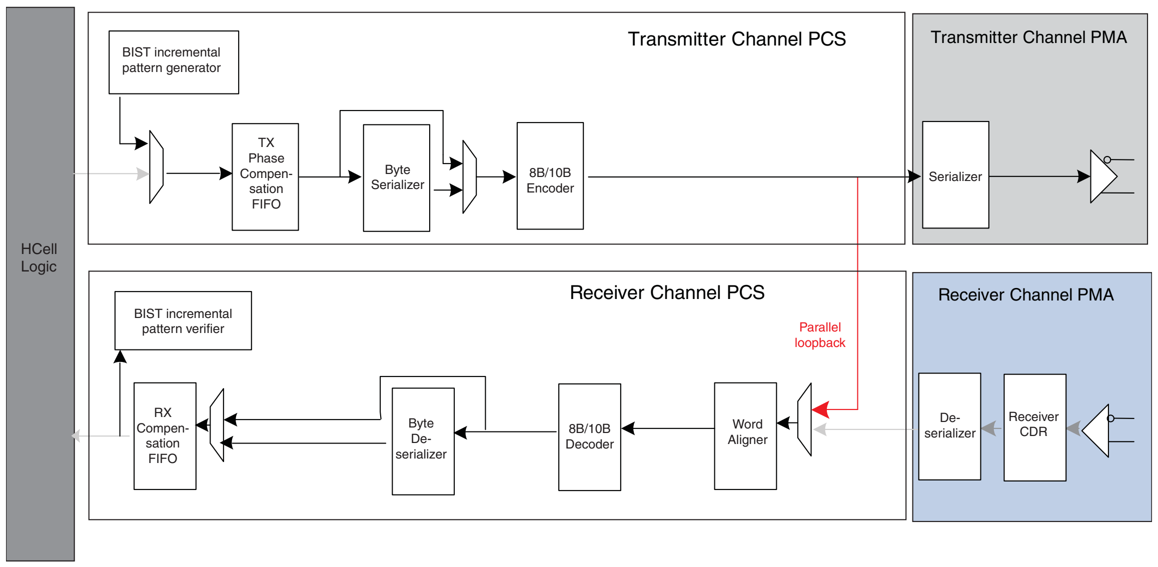


Table 1–39 lists the enabled PCS functional blocks for single-width and double-width mode. The last column in Table 1–39 lists the supported channel width setting for parallel loopback.

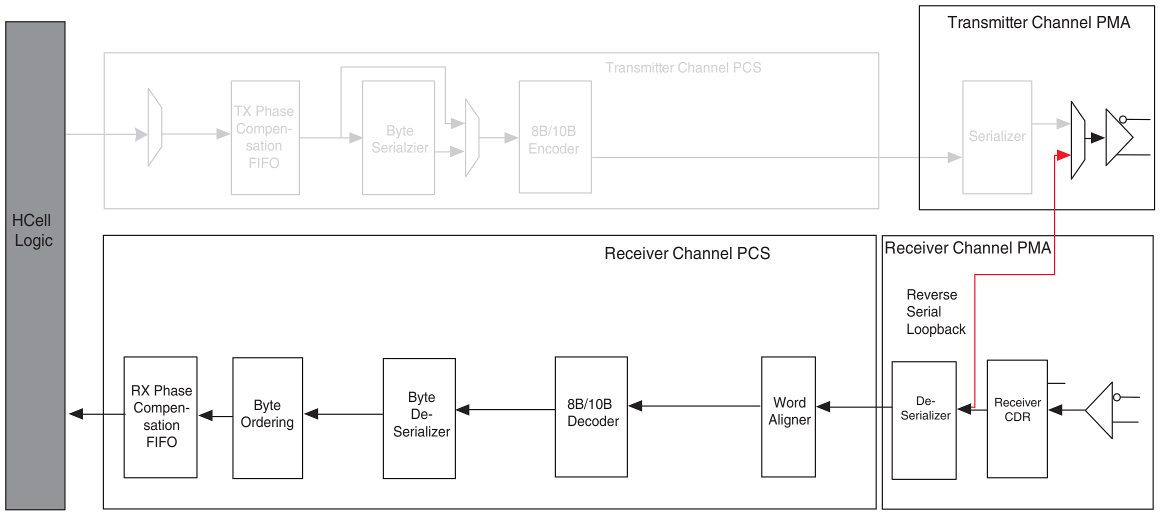
**Table 1–39. Enabled PCS Functional Blocks for Parallel Loopback**

| 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 |

### Reverse Serial Loopback

Reverse serial loopback is available as a subprotocol under Basic functional mode. In reverse serial loopback mode, the data is received through the `rx_datain` port, retimed through the receiver CDR and sent out to the `tx_dataout` port. The received data is also available to the HCell logic. Figure 1–75 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.

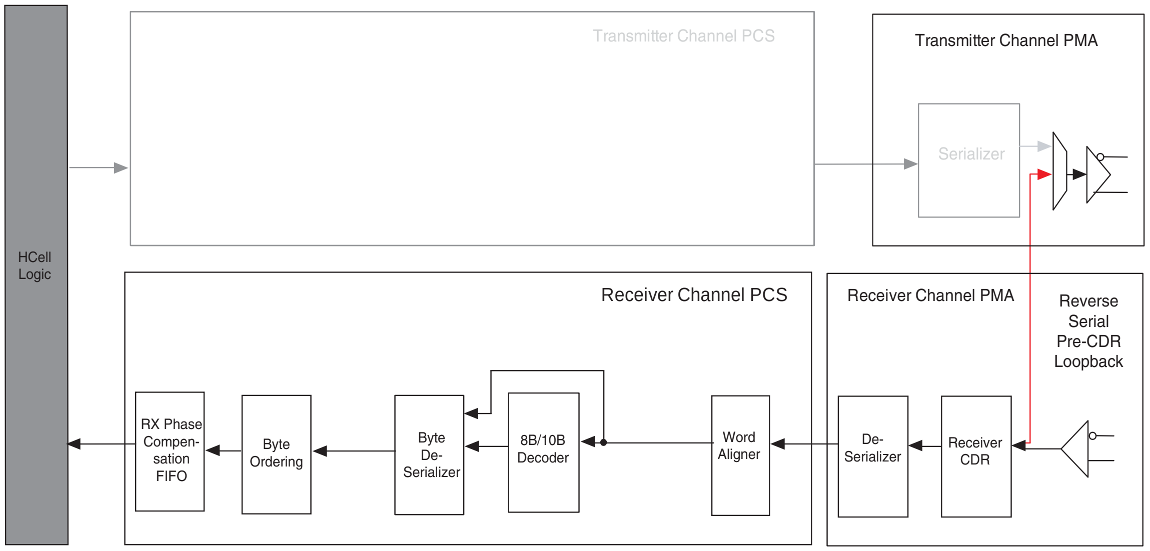
**Figure 1–75. Reverse Serial Loopback Datapath**



### 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 HCell Logic. Figure 1–76 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. You cannot change the pre-emphasis settings for the transmitter buffer in this configuration.

**Figure 1–76. Reverse Serial Pre-CDR Loopback Datapath**



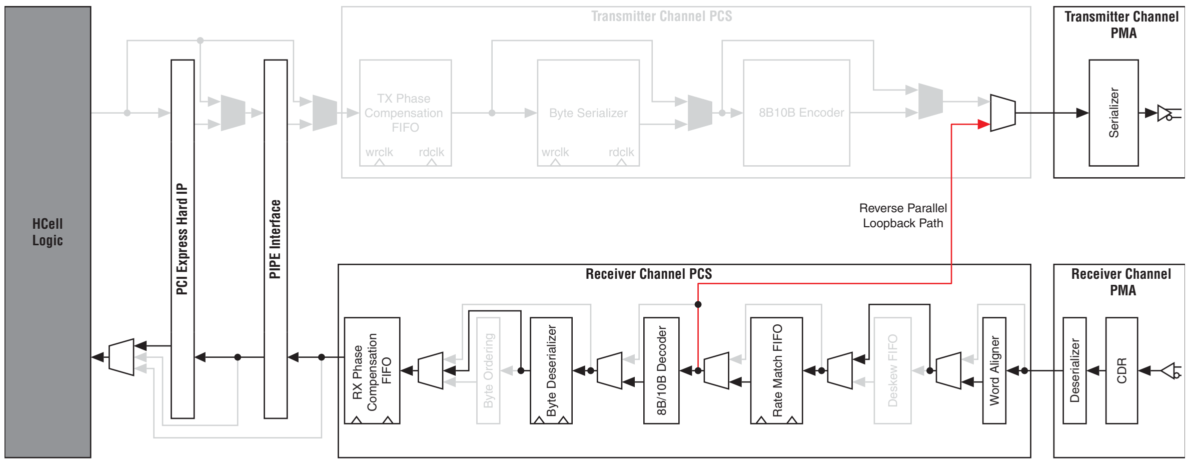
## PCIe Reverse Parallel Loopback

PCIe reverse parallel loopback is only available in PCIe functional mode for Gen1 and Gen2 data rates. As shown in Figure 1–77, 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 HCell Logic 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.

This is the only loopback option supported in PCIe functional mode.

In Figure 1–77, the grayed areas show the inactive paths when the PCIe reverse parallel loopback mode is enabled.

**Figure 1–77. PCIe Reverse Parallel Loopback Mode Datapath**



## Auxiliary Transmit (ATX) PLL Block

The HardCopy IV GX transceiver contains the ATX PLL block that you can use to generate high-speed clocks for the transmitter channels on the same side of the device. The following data rates are supported by the ATX PLLs:

- 4.8 Gbps to 5.4 Gbps and 6.0 Gbps to 6.5 Gbps

Using the dividers available in the ATX PLLs:

- 2.4 Gbps to 2.7 Gbps and 3.0 Gbps to 3.25 Gbps
- 1.2 Gbps to 1.35 Gbps and 1.5 Gbps to 1.625 Gbps

Figure 1–78, Figure 1–79, and Figure 1–80 show the location of the ATX PLL blocks in two, four, and six transceiver block device families.

**Figure 1–78. Location of ATX PLL Block in a Two-Transceiver Block Device**

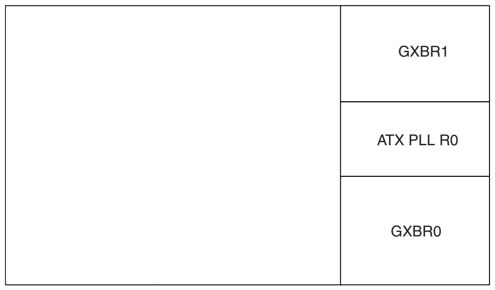


**Figure 1–79. Location of ATX PLL Blocks in a Four-Transceiver Block Device (Two on Each Side)**

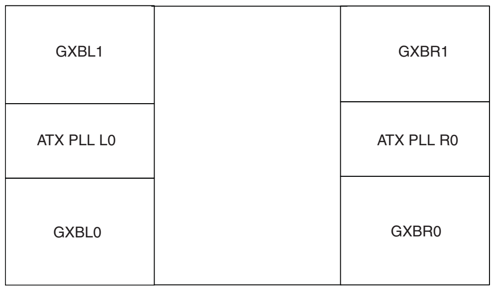
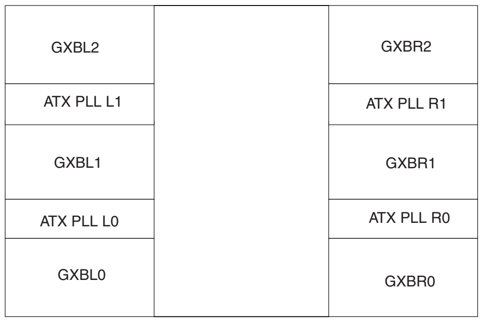


**Figure 1–80. Location of ATX PLL Blocks in a Six-Transceiver Block Device (Three on Each Side)**



## Input Reference Clocks for the ATX PLL Block

The ATX PLL block does not have a dedicated reference clock pin. The following are the possible input reference clock sources:

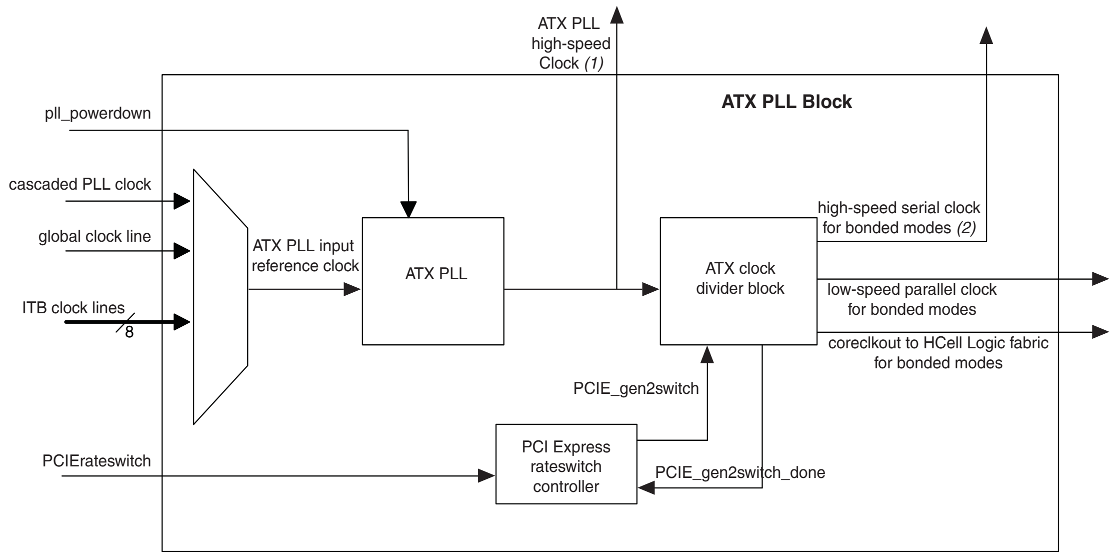
- Input reference clock provided through the PLL cascade clock network
- Clock inputs connected through the global clock lines
- REFCLKS from the transceiver blocks on the same side of the device, if the corresponding CMU channels are not used as transceiver channels

☞ Altera recommends using the REFCLK pins from the adjacent transceiver block below the ATX PLL block to improve performance.

## Architecture of the ATX PLL Block

The ATX PLL block contains the ATX PLL, ATX clock divider, and a shared control signal generation block, as shown in Figure 1–81.

**Figure 1–81. ATX PLL Block**


ATX PLL high-speed Clock (1)
ATX PLL Block
pll_powerdown
cascaded PLL clock
global clock line
ITB clock lines
8
ATX PLL input reference clock
ATX PLL
ATX clock divider block
high-speed serial clock for bonded modes (2)
low-speed parallel clock for bonded modes
coreclkout to HCell Logic fabric for bonded modes
PCIE_gen2switch
PCIErateswitch
PCI Express rateswitch controller
PCIE_gen2switch_done

**Notes to Figure 1–81:**

(1) In non-bonded functional modes (for example, CEI functional mode), the transmitter channel uses the transmitter local clock divider to divide this high-speed clock output to provide clocks for its PMA and PCS blocks.

(2) This is used in Basic ×4, ×8, and PCIe ×4 and ×8 functional modes.

The functional blocks on the ATX PLL are similar to the blocks explained in "CMU0 PLL" on page 1–47. 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.

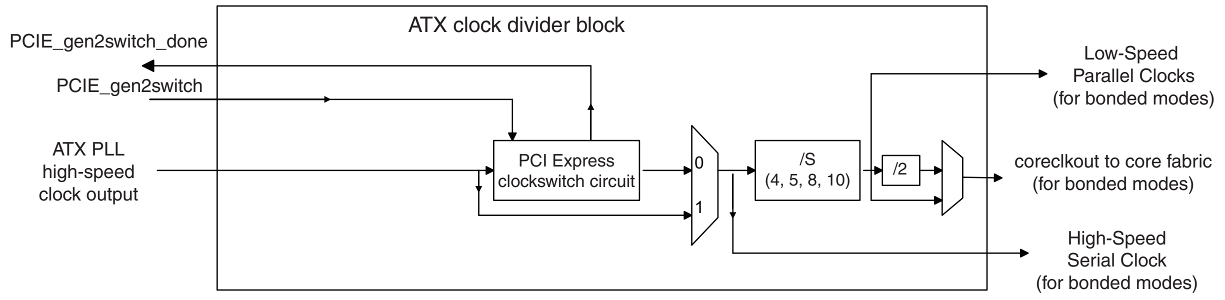
The ATX PLL high-speed clock output provides high-speed serial clocks for non-bonded functional modes such as CEI (with the subprotocol "none").

### 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, ×8), Basic ×4 and ×8, and PMA-Direct mode with ×N configuration. For PIPE functional mode support, the ATX clock divider consists of the PIPE rateswitch circuit to enable dynamic rateswitch between PIPE Gen1 and Gen2 data rates. For more information about this circuit, refer to "CMU0 Channel" on page 1–46.

Figure 1–82 shows the clock outputs from the ATX clock divider block.

**Figure 1–82. ATX Clock Divider**


ATX clock divider block
PCIE_gen2switch_done
PCIE_gen2switch
ATX PLL high-speed clock output
PCI Express clockswitch circuit
0
1
/S (4, 5, 8, 10)
/2
Low-Speed Parallel Clocks (for bonded modes)
coreclkout to core fabric (for bonded modes)
High-Speed Serial Clock (for bonded modes)

## Differences Between the ATX PLL and CMU PLL

Table 1–40 lists the differences between the ATX PLL and CMU PLL.

**Table 1–40. Differences Between the ATX PLL and CMU PLL (Part 1 of 2)**

| Difference Category/PLLs | 6G ATX PLL | CMU PLL |
|---|---|---|
| Data rates (Gbps) | 4.8 to 5.4 and 6.0 and 6.5<br>2.4 to 2.7 and 3.0 and 3.25 *(1)*<br>1.2 to 1.35 and 1.5 to 1.625 *(1)* | 0.6 to 6.5 |
| Input reference clock options | ■ Clock inputs connected through the ITB lines.<br>■ Clock inputs connected through the PLL cascade clock network.<br>■ Clock inputs connected through the global clock lines. *(2)* | ■ Clock inputs connected through the inter transceiver block (ITB) lines.<br>■ clock inputs connected through the PLL cascade clock network.<br>■ Clock inputs connected through the global clock lines, `refclk0` and `refclk1` clock input, dedicated `refclks` in the transceiver block. *(2)* |
| Power Supply—$V_{CCA\_L/R}$ (V) options for PLLs | 3.0 | 2.5 or 3.0 |

**Table 1–40. Differences Between the ATX PLL and CMU PLL (Part 2 of 2)**

| Difference Category/PLLs | 6G ATX PLL | CMU PLL |
|---|---|---|
| Phase noise | Lower when compared with the CMU PLL *(3)* | Higher when compared with the ATX PLLs *(3)* |

**Notes to Table 1–40:**

(1) Using the L dividers available in ATX PLLs.

(2) For more information, refer to the Input Reference Clock Source table in the *Stratix IV Transceiver Clocking* chapter in volume 2 of the *Stratix IV Device Handbook*.

(3) For more information about phase noise and PLL bandwidths of ATX and CMU PLLs, refer to the characterization reports.

## Calibration Blocks

HardCopy IV GX 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 (PVT) variations.

### Calibration Block Location

Figure 1–83 shows the location and number of calibration blocks available for different transceiver block device families. In Figure 1–83 through Figure 1–85, the calibration block R0 and L0 refer to the calibration blocks on the right and left side, respectively.

**Figure 1–83. Calibration Blocks in HardCopy IV Devices**

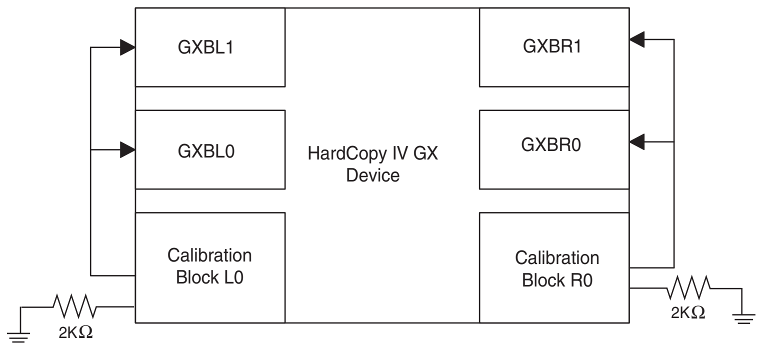


Figure 1–84 shows HardCopy IV GX device families that have three transceiver blocks each on the left and right side.

**Figure 1–84. Calibration Block Locations with Three-Transceiver Blocks (on Each Side) in HardCopy IV GX Device**

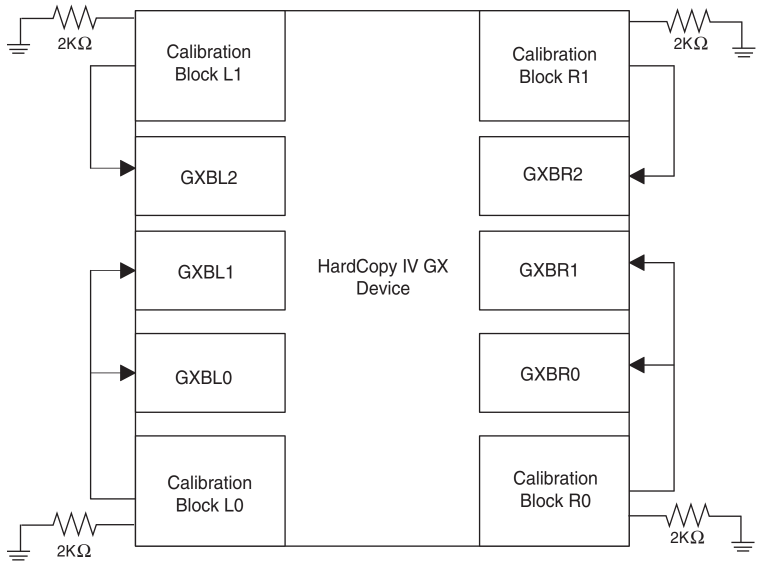
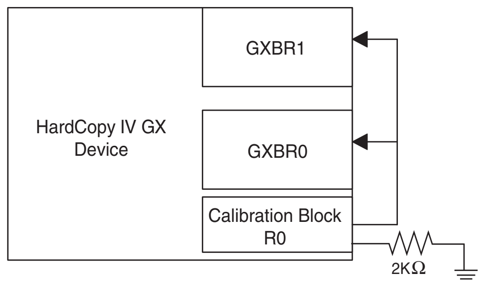


Figure 1–85 shows HardCopy IV GX device families that have two transceiver blocks only on the right side of the device.

**Figure 1–85. Calibration Two Transceiver Blocks, Right Side Only, in HardCopy IV GX Device Families**



The Quartus II software automatically selects the appropriate calibration block based on the assignment of the transceiver `tx_dataout` and `rx_datain` pins.

## Calibration

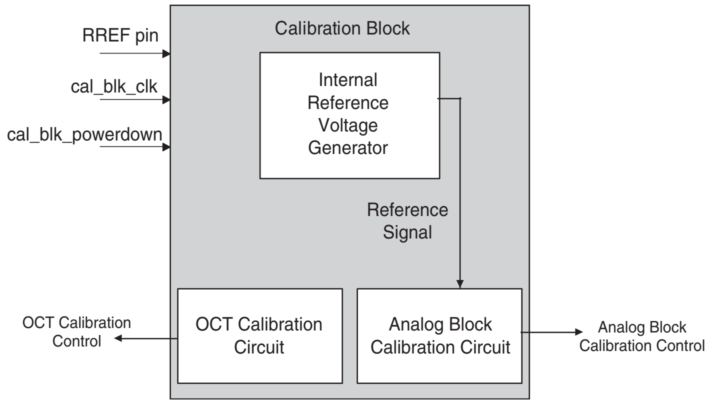
The calibration block internally generates a constant internal reference voltage, independent of process, voltage, or temperature variations. It uses the internal reference voltage and external reference resistor (you must connect the resistor to the `RREF` pin) to generate constant reference currents. These reference currents are used by the analog block calibration circuit to calibrate the transceiver blocks.

The OCT calibration circuit calibrates the OCT resistors present in the transceiver channels. You can enable the OCT resistors in the transceiver channels through the ALTGX MegaWizard Plug-In Manager.

You must connect a separate 2 kΩ (tolerance max ± 1%) external resistor on each `RREF` pin in the HardCopy IV GX device to ground. To ensure proper operation of the calibration block, the `RREF` resistor connection in the board must be free from external noise.

Figure 1–86 shows the required inputs to the calibration block.

**Figure 1–86. Input Signals to the Calibration Blocks**



For more information about the calibration blocks, refer to the *Calibration Blocks* section in the *Stratix IV Transceiver Architecture* chapter in volume 2 of the *Stratix IV Handbook*.

## Input Reference Clocking

Each transceiver block has:

- Two clock multiplier unit channels (`CMU0_Channel` and `CMU1_Channel`)

  You can configure each as either a CMU to generate transceiver clocks or as a PMA-Only channel.

  For more information about CMUs, refer to the "*CMU Channel Architecture*" section in the *Stratix IV Transceiver Architecture* chapter.

- 6G ATX PLLs

ATX PLLs generate the high-speed serial transceiver clock. The ATX PLLs also need an input reference clock for operation.

For more information about ATX PLLs, refer to "Auxiliary Transmit (ATX) PLL Block" on page 1–113 and "Transmitter Channel Datapath" on page 1–8.

## Input Reference Clock Source

Receiver CDRs, CMU PLLs (when the CMU channel is configured as a CMU), and ATX PLLs can derive the input reference clock from one of the sources listed in Table 1–41.

**Table 1–41. Input Reference Clock Source**

| Index | Clock Source | CMU PLL | 6G ATX PLL | Jitter Performance *(1)* |
|---|---|---|---|---|
| 1 | `refclk0` and `refclk1` pins of the same transceiver block | Yes | No *(2)* | 1 |
| 2 | `refclk0` and `refclk1` pins of other transceiver blocks on the same side of the device using the inter-transceiver block (ITB) clock lines | Yes | Yes | 2 *(3)* |
| 3 | Clock output from the left and right PLLs in the HCell Logic with voltage controlled oscillator (VCO) bypass mode | Yes | Yes | 3 |
| 4 | Clock output from the left and right PLLs in the HCell Logic | Yes | Yes | 4 |
| 5 | Dedicated CLK input pins on the HCell Logic global clock network | Yes | Yes | 4 |

**Notes to Table 1–41:**

(1) Lowest number indicates best jitter performance.

(2) ATX PLLs do not have dedicated `refclk` pins.

(3) For better jitter performance, Altera strongly recommends using the `refclk0` and `refclk1` pins of the transceiver block located immediately below the ATX PLL.

## HCell Logic PLLs-Transceiver PLLs Cascading

You can fulfill multiple reference clock frequency needs by taking advantage of the PLL cascading option. For the best results, ensure that your design meets the following rules:

- You can only cascade the PLLs to the transceiver blocks located on the same side of the device. The PLL cascade networks are single clock lines segmented by bidirectional tri-state buffers located along the clock line.
- Segmentation of the PLL cascade network allows two left and right PLLs to drive the cascade clock line simultaneously and provides the input reference clock to the CMU PLLs and receiver CDRs in different transceiver blocks.
- When cascading two or more HCell Logic PLLs to the CMU PLLs and receiver CDRs, there must be no crossover in the cascaded clock paths on the PLL cascade network.

For better noise rejection, ensure that the bandwidth setting of the HCell Logic PLL (the upstream PLL) is lower than the transceiver PLL (the downstream PLL).

### HCell Logic-Transceiver Interface Clocking

The HCell Logic-transceiver interface clocks consists of clock signals from the HCell Logic to the transceiver blocks, and clock signals from the transceiver blocks to the HCell Logic. These clock resources use the clock networks in the HCell Logic core that include the global, regional, and periphery clock networks. Transceiver datapath interface clocks are used to transfer data, control, and status signals between the HCell Logic and the transceiver channels. The transceiver channel forwards the `tx_clkout` signal (in non-bonded modes) or the `coreclkout` signal (in bonded channel modes) to the HCell Logic to clock the data and control signals into the transmitter. The transceiver channel also forwards the recovered `rx_clkout` clock (in configurations without the rate matcher) or `tx_clkout/coreclkout` (in configurations with the rate matcher) to the HCell Logic to clock the data and status signals from the receiver into the HCell Logic.

In Basic (PMA Direct) functional mode, only `tx_clkout` and `rx_clkout` are available to clock the logic in the core. In bonded mode, you can use the `tx_clkout` from one of the channels to clock all of the channels. For receivers in bonded mode, you must use a separate `rx_clkout` for each channel.

## User Reset and Power-Down Signals

HardCopy IV GX devices offer multiple reset signals to control transceiver channels and CMU PLLs independently. The ALTGX Transceiver MegaWizard™ Plug-In Manager provides individual reset signals for each channel instantiated in your design. It also provides one power down signal for each transceiver block.

HardCopy IV GX devices share the same transceiver reset and power-down sequence as Stratix IV GX devices. Each transceiver channel in the HardCopy IV GX device has individual reset signals to reset its physical coding sublayer (PCS) and physical medium attachment (PMA) blocks. Each CMU PLL in the transceiver block has a dedicated reset signal. The transceiver block also has a power-down signal that affects all the channels and CMU PLLs in the transceiver block.

If none of the channels are instantiated in a transceiver block, the Quartus II software automatically powers down the entire transceiver block.

### Blocks Affected by the Reset and Power-Down Signals

Table 1–42 lists the blocks that are affected by specific reset and power-down signals.

**Table 1–42. Blocks Affected by Reset and Power-Down Signals (Part 1 of 2)**

| Transceiver Block | rx_digitalreset | rx_analogreset | tx_digitalreset | pll_powerdown | gxb_powerdown |
|---|---|---|---|---|---|
| CMU PLLs | — | — | — | ✓ | ✓ |
| Transmitter Phase Compensation FIFO | — | — | ✓ | — | ✓ |
| Byte Serializer | — | — | ✓ | — | ✓ |
| 8B/10B Encoder | — | — | ✓ | — | ✓ |
| Serializer | — | — | ✓ | — | ✓ |
| Transmitter Buffer | — | — | — | — | ✓ |

**Table 1–42. Blocks Affected by Reset and Power-Down Signals (Part 2 of 2)**

| Transceiver Block | rx_digitalreset | rx_analogreset | tx_digitalreset | pll_powerdown | gxb_powerdown |
|---|---|---|---|---|---|
| Transmitter XAUI State Machine | — | — | ✓ | — | ✓ |
| Receiver Buffer | — | — | — | — | ✓ |
| Receiver CDR | — | ✓ | — | — | ✓ |
| Receiver Deserializer | — | — | — | — | ✓ |
| Receiver Word Aligner | ✓ | — | — | — | ✓ |
| Receiver Deskew FIFO | ✓ | — | — | — | ✓ |
| Receiver Clock Rate Compensation FIFO | ✓ | — | — | — | ✓ |
| Receiver 8B/10B Decoder | ✓ | — | — | — | ✓ |
| Receiver Byte Deserializer | ✓ | — | — | — | ✓ |
| Receiver Byte Ordering | ✓ | — | — | — | ✓ |
| Receiver Phase Compensation FIFO | ✓ | — | — | — | ✓ |
| Receiver XAUI State Machine | ✓ | — | — | — | ✓ |
| BIST Verifiers | ✓ | — | — | — | ✓ |

### Transceiver Reset Sequences

You can configure transceiver channels in HardCopy IV GX devices in various configurations. In all functional modes except XAUI functional mode, transceiver channels can be either bonded or non-bonded. In PCIe functional mode, transceiver channels can be either bonded or non-bonded, but must follow a specific reset sequence.

For more information about transceiver reset sequences, refer to the *Reset Control and Power Down* chapter in the *Stratix IV GX Handbook*.

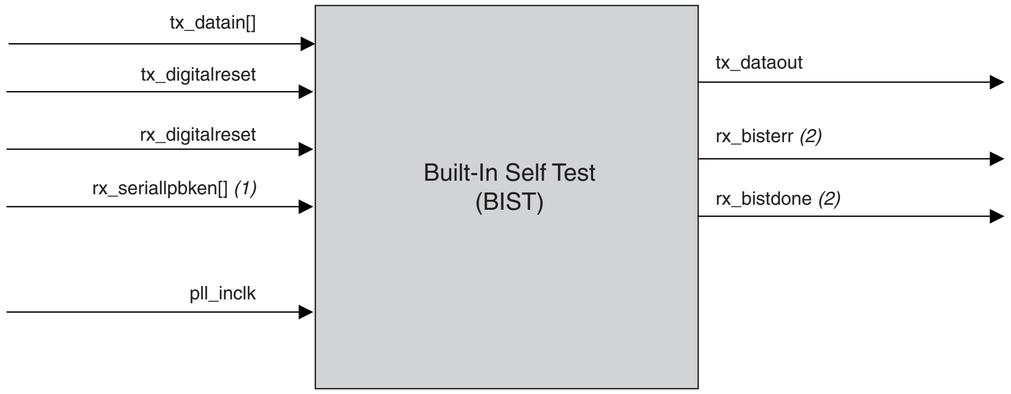
## Built-In Self Test Modes

This section describes Built-In Self Test (BIST) modes.

### BIST Mode Pattern Generators and Verifiers

Each transceiver channel in HardCopy IV GX 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 1–87 shows the enabled input and output ports when you select BIST mode (except incremental patterns).

**Figure 1–87. Input and Output Ports for BIST Modes**



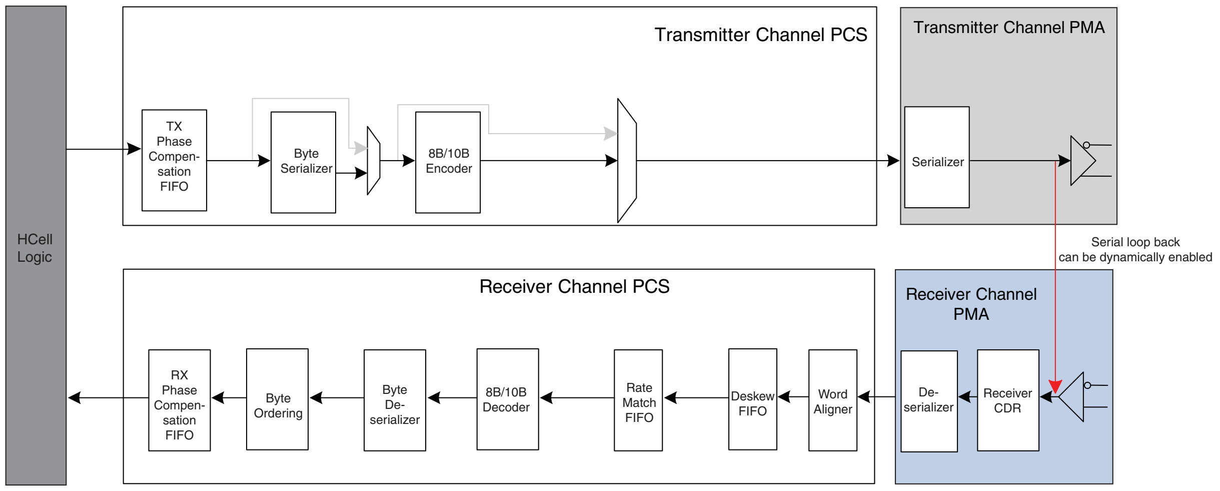
**Notes to Figure 1–87:**

(1) `rx_serilalpbken` is required in PRBS.

(2) `rx_bisterr` and `rx_bistdone` are only available in PRBS and BIST modes.

Figure 1–88 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 1–88. BIST PRBS, High Frequency, and Low Frequency Pattern Datapath**



For more information about BIST modes, refer to the *Built-In Self Test Modes* section in the *Stratix IV Device Handbook*.

## Transceiver Port Lists

Instantiate the HardCopy IV GX 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 1–43 through Table 1–49 provide a brief description of the ALTGX megafunction ports.

Table 1–43 lists the ALTGX megafunction transmitter ports.

**Table 1–43. ALTGX Megafunction Ports: Transmitter Ports in HardCopy IV GX Devices (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 HCell Logic to the transmitter.<br>■ Bus width—depends on the channel width multiplied by the number of channels per instance. | Channel |
| `tx_clkout` | Output | Clock signal | HCell fabric-transceiver interface clock.<br>■ Bonded channel configurations—not available.<br>■ Non-bonded channel configurations—each channel has a `tx_clkout` signal.<br>■ Use this clock signal to clock the parallel data `tx_datain` from the HCell fabric into the transmitter. | Channel |
| `tx_coreclk` | Input | Clock signal | Optional write clock port for the transmitter phase compensation FIFO.<br>■ If not selected—the Quartus II software automatically selects `tx_clkout/coreclkout` as the write clock for transmitter phase compensation FIFO.<br>■ 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.<br>■ A high level—the transmitter phase compensation FIFO is either full or empty. | Channel |

**Table 1–43. ALTGX Megafunction Ports: Transmitter Ports in HardCopy IV GX Devices (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. | 8B/10B encoder /Kx.y/ or /Dx.y/ control.<br>■ When asserted high—the 8B/10B encoder encodes the data on the `tx_datain` port as a /Kx.y/ control code group.<br>■ When de-asserted low—it encodes the data on the `tx_datain` port as a /Dx.y/ data code group.<br>■ Channel Width:<br>8—`tx_ctrlenable` = 1<br>16—`tx_ctrlenable` = 2<br>32—`tx_ctrlenable` = 4 | Channel |
| `tx_forcedisp` | Input | Asynchronous signal. Minimum pulse width is two parallel clock cycles. | 8B/10B encoder force disparity control.<br>■ 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.<br>■ When de-asserted low—the 8B/10B encoder encodes the data on the `tx_datain` port according to the 8B/10B running disparity rules.<br>■ Channel Width:<br>8—`tx_forcedisp` = 1<br>16—`tx_forcedisp` = 2<br>32—`tx_forcedisp` = 4 | Channel |
| `tx_dispval` | Input | Asynchronous signal. Minimum pulse width is two parallel clock cycles. | 8B/10B encoder force disparity value.<br>■ 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.<br>■ 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.<br>■ Channel Width:<br>8—`tx_dispval` = 1<br>16—`tx_dispval` = 2<br>32—`tx_dispval` = 4 | Channel |

**Table 1–43. ALTGX Megafunction Ports: Transmitter Ports in HardCopy IV GX Devices (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. | 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.<br>■ 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.<br>■ When asserted high in double-width mode—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 | — | Transmitter serial data output port. | Channel |
| `fixedclk` | Input | Clock signal | 125-MHz clock for receiver detect and offset cancellation in PCIe mode. | Channel |

Table 1–44 lists the ALTGX megafunction receiver ports.

**Table 1–44. ALTGX Megafunction Ports: Receiver Ports HardCopy IV GX Devices (Part 1 of 8)**

| Port Name | Input/ Output | Clock Domain | Description | Scope |
|---|---|---|---|---|
| `rx_syncstatus` | Output | Synchronous to `coreclkout` clock signal | Word alignment synchronization status indicator.<br>■ 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.<br>■ Manual alignment mode—the behavior of this signal depends on whether the transceiver is configured in single-width or double-width mode.<br>■ Bit-Slip mode—not available.<br>■ Channel width:<br>8/10—`rx_syncstatus` = 1<br>16/20—`rx_syncstatus` = 2 32/40—<br>`rx_syncstatus` = 4 | Channel |
| `rx_bitslip` | Input | Asynchronous signal. Minimum pulse width is two parallel clock cycles. | Bit-slip control for the word aligner configured in bit-slip mode.<br>At every rising edge, word aligner slips one bit into the received data stream, effectively shifting the word boundary by one bit. | Channel |

**Table 1–44. ALTGX Megafunction Ports: Receiver Ports HardCopy IV GX Devices (Part 2 of 8)**

| Port Name | Input/ Output | Clock Domain | Description | Scope |
|---|---|---|---|---|
| `rx_a1a2size` | Input | Asynchronous Signal. Minimum pulse width is two parallel clock cycles. | Available only in SONET OC-12 and OC-48 modes. Select between these options:<br>■ 0 = 16-bit A1A2<br>■ 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. | Run-length violation indicator.<br>A high pulse is driven when the number of consecutive 1s or 0s in the received data stream exceeds the programmed run length violation threshold. | Channel |
| `rx_invpolarity` | Input | Asynchronous Signal. Minimum pulse width is two parallel clock cycles. | Generic receiver polarity inversion control. Useful feature for correcting situations where the positive and negative signals of the differential serial link are accidentally swapped during board layout.<br>■ 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.<br>■ When asserted high in double-width modes—the polarity of every bit of the 16-bit or 20-bit input data to the word aligner gets inverted. | Channel |
| `rx_revbitorderwa` | Input | Asynchronous Signal. Minimum pulse width is two parallel clock cycles. | Receiver bit reversal control. This is a useful feature where the link transmission order is MSB to LSB.<br>■ Available only in Basic single-width and double-width modes with the word aligner configured in bit-slip mode.<br>■ 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.<br>■ When asserted high in Basic double-width modes—the 16-bit or 20-bit data `D[15:0]` or `D[19:0]` at the output of the word aligner gets rewired to `D[0:15]` or `D[0:19]`, respectively. | Channel |

**Table 1–44. ALTGX Megafunction Ports: Receiver Ports HardCopy IV GX Devices (Part 3 of 8)**

| Port Name | Input/ Output | Clock Domain | Description | Scope |
|---|---|---|---|---|
| `rx_revbyteorderwa` | Input | Asynchronous Signal. Minimum pulse width is two parallel clock cycles. | Receiver byte reversal control. This is a useful feature in situations where the MSByte and LSByte of the transmitted data are erroneously swapped.<br>■ Available only in Basic double-width mode.<br>■ When asserted high, the MSByte and LSByte of the 16- and 20-bit data at the output of the word aligner get swapped. | Channel |
| **Deskew FIFO** | | | | |
| `rx_channelaligned` | Output | Synchronous to `coreclkout` clock signal | XAUI deskew FIFO channel aligned indicator.<br>■ Available only in XAUI mode.<br>■ 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.<br>■ 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_`<br>`rmfifodatainserted` | Output | Synchronous to `tx_clkout` or `coreclkout`.<br>`tx_clkout` for non-bonded modes.<br>`coreclkout` for bonded modes. | Rate match FIFO insertion status indicator.<br>■ A high level—the rate match pattern byte has inserted to compensate for the PPM difference in reference clock frequencies between the upstream transmitter and the local receiver. | Channel |
| `rx_rmfifodatadeleted` | Output | Synchronous to `tx_clkout` or `coreclkout`.<br>`tx_clkout` for non-bonded modes.<br>`coreclkout` for bonded modes. | Rate match FIFO deletion status indicator.<br>■ 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 |

**Table 1–44. ALTGX Megafunction Ports: Receiver Ports HardCopy IV GX Devices (Part 4 of 8)**

| Port Name | Input/ Output | Clock Domain | Description | Scope |
|---|---|---|---|---|
| `rx_rmfifofull` | Output | Synchronous to `tx_clkout` or `coreclkout`.<br>`tx_clkout` for non-bonded modes.<br>`coreclkout` for bonded modes. | Rate match FIFO full status indicator.<br>■ A high level indicates that the rate match FIFO is full.<br>■ Without byte serializer —driven a minimum of two recovered clock cycles.<br>■ With byte serializer—driven a minimum of three recovered clock cycles. | Channel |
| `rx_rmfifoempty` | Output | Synchronous to `tx_clkout` or `coreclkout`.<br>`tx_clkout` for non-bonded modes.<br>`coreclkout` for bonded modes. | Rate match FIFO empty status indicator.<br>■ A high level—the rate match FIFO is empty.<br>■ Without byte serializer—driven a minimum of two recovered clock cycles.<br>■ With byte serializer—driven a minimum of three recovered clock cycles. | Channel |
| **8B/10B Decoder** | | | | |
| `rx_ctrldetect` | Output | Synchronous to `coreclkout` clock signal | Receiver control code indicator.<br>■ Available in configurations with 8B/10B decoder.<br>■ A high level—the associated received code group is a control (/Kx.y/) code group.<br>■ A low level—the associated received code group is a data (/Dx.y/) code group.<br>■ The width of this signal depends on the following channel width:<br>Channel Width:<br>8—`rx_ctrldetect` = 1<br>16—`rx_ctrldetect` = 2<br>32—`rx_ctrldetect` = 4 | Channel |

**Table 1–44. ALTGX Megafunction Ports: Receiver Ports HardCopy IV GX Devices (Part 5 of 8)**

| Port Name | Input/ Output | Clock Domain | Description | Scope |
|---|---|---|---|---|
| `rx_errdetect` | Output | Synchronous to `tx_clkout` or `coreclkout`.<br>`tx_clkout` for non-bonded modes.<br>`coreclkout` for bonded modes. | 8B/10B code group violation or disparity error indicator.<br>■ Available in configurations with 8B/10B decoder.<br>■ 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:<br>■ `[rx_errdetect: rx_disperr]`<br>■ 2'b00—no error<br>■ 2'b10—code group violation<br>■ 2'b11—disparity error or both<br>■ Channel Width:<br>8—`rx_errdetect` = 1<br>16—`rx_errdetect` = 2<br>32—`rx_errdetect` = 4 | Channel |
| `rx_disperr` | Output | Synchronous to `tx_clkout` or `coreclkout`.<br>`tx_clkout` for non-bonded modes.<br>`coreclkout` for bonded modes. | 8B/10B disparity error indicator port.<br>■ Available in configurations with 8B/10B decoder.<br>■ A high level—a disparity error was detected on the associated received code group.<br>■ Channel Width:<br>8—`rx_disperr` = 1<br>16—`rx_disperr` = 2<br>32—`rx_disperr` = 4 | Channel |
| `rx_runningdisp` | Output | Synchronous to `tx_clkout` or `coreclkout`.<br>`tx_clkout` for non-bonded modes.<br>`coreclkout` for bonded modes. | 8B/10B running disparity indicator.<br>■ Available in configurations with the 8B/10B decoder.<br>■ A high level—the data on the `rx_dataout` port was received with a negative running disparity.<br>■ A low level—the data on the `rx_dataout` port was received with a positive running disparity.<br>■ Channel Width:<br>8—`rx_runningdisp` = 1<br>16—`rx_runningdisp` = 2<br>32—`rx_runningdisp` = 4 | Channel |

**Table 1–44. ALTGX Megafunction Ports: Receiver Ports HardCopy IV GX Devices (Part 6 of 8)**

| Port Name | Input/ Output | Clock Domain | Description | Scope |
|---|---|---|---|---|
| **Byte Ordering Block** | | | | |
| `rx_enabyteord` | Input | Asynchronous signal | Enable byte ordering control.<br>■ Available in configurations with the byte ordering block enabled. The byte ordering block is rising-edge sensitive to this signal.<br>■ A low-to-high transition triggers the byte ordering block to restart the byte ordering operation. | Channel |
| `rx_ byteorderalignstatus` | Output | Synchronous to `tx_clkout` or `coreclkout`.<br>`tx_clkout` for non-bonded modes.<br>`coreclkout` for bonded modes. | Byte ordering status indicator.<br>■ Available in configurations with the byte ordering block enabled.<br>■ A high level—the byte ordering block has detected the programmed byte ordering pattern in the LSByte of the received data from the byte deserializer. | Channel |
| **Receiver Phase Compensation FIFO** | | | | |
| `rx_dataout` | Output | Synchronous to `tx_clkout` or `coreclkout`.<br>`tx_clkout` for non-bonded modes.<br>`coreclkout` for bonded modes. | Parallel data output from the receiver to the HCell fabric.<br>■ The bus width depends on the channel width multiplied by the number of channels per instance. | Channel |
| `rx_clkout` | Output | Clock signal | Recovered clock from the receiver channel.<br>■ Available only when the rate match FIFO is not used in the receiver datapath. | Channel |
| `rx_coreclk` | Input | Clock signal | Optional read clock port for the receiver phase compensation FIFO.<br>■ If not selected—the Quartus II software automatically selects `rx_clkout/tx_clkout/ coreclkout` as the read clock for the receiver phase compensation FIFO.<br>■ If selected—drive this port with a clock that has 0 PPM difference with respect to `rx_clkout/tx_clkout/ coreclkout`. | Channel |

**Table 1–44. ALTGX Megafunction Ports: Receiver Ports HardCopy IV GX Devices (Part 7 of 8)**

| Port Name | Input/ Output | Clock Domain | Description | Scope |
|---|---|---|---|---|
| `rx_phase_comp_fifo_error` | Output | Synchronous to `tx_clkout` or `coreclkout`. `tx_clkout` for non-bonded modes. `coreclkout` for bonded modes. | Receiver phase compensation FIFO full or empty indicator. ■ A high level—the receiver phase compensation FIFO is either full or empty. | Channel |
| **Receiver Physical Media Attachment (PMA)** | | | | |
| `rx_datain` | Input | — | Receiver serial data input port. | Channel |
| `rx_cruclk` | Input | Clock signal | Input reference clock for the receiver clock and data recovery. | Channel |
| `rx_pll_locked` | Output | Asynchronous signal | Receiver CDR lock-to-reference indicator. ■ A high level—the receiver CDR is locked to the input reference clock. ■ A low level—the receiver CDR is not locked to the input reference clock. | Channel |
| `rx_freqlocked` | Output | Asynchronous signal | Receiver CDR lock mode indicator. ■ A high level—the receiver CDR is in lock-to-data mode. ■ A low level—the receiver CDR is in lock-to-reference mode. | Channel |
| `rx_locktodata` | Input | Asynchronous signal | Receiver CDR lock-to-data mode control signal. ■ When asserted high—the receiver CDR is forced to lock-to-data mode. ■ When de-asserted low—the receiver CDR lock mode depends on the `rx_locktorefclk` signal level. | Channel |
| `rx_locktorefclk` | Input | Asynchronous signal | Receiver CDR lock-to-reference mode control signal. The `rx_locktorefclk` signal, along with the `rx_locktodata` signal, controls whether the receiver CDR is in automatic (0/0), lock-to-reference (0/1), or lock-to-data (1/x) mode. | Channel |

**Table 1–44. ALTGX Megafunction Ports: Receiver Ports HardCopy IV GX Devices (Part 8 of 8)**

| Port Name | Input/ Output | Clock Domain | Description | Scope |
|---|---|---|---|---|
| `rx_signaldetect` | Output | Asynchronous signal | Signal threshold detect indicator.<br>■ Available in Basic functional mode when the 8B/10B Encoder/Decoder is selected.<br>■ Available in PCIe mode.<br>■ A high level—the signal present at the receiver input buffer is above the programmed signal detection threshold value.<br>■ 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_seriallpbken` | Input | Asynchronous signal | Serial loopback control port.<br>■ 0–normal datapath, no serial loopback<br>■ 1–serial loopback | Channel |

Table 1–45 lists the ALTGX megafunction CMU ports.

**Table 1–45. ALTGX Megafunction Ports: CMU in HardCopy IV GX Devices (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.<br>■ A high level—the CMU PLL is locked to the input reference clock.<br>■ A low level—the CMU PLL is not locked to the input reference clock. | Transceiver block |

**Table 1–45. ALTGX Megafunction Ports: CMU in HardCopy IV GX Devices (Part 2 of 2)**

| Port Name | Input/ Output | Clock Domain | Description | Scope |
|---|---|---|---|---|
| `pll_powerdown` | Input | Asynchronous signal.<br>For minimum pulse width requirements, refer to the *DC and Switching Characteristics of HardCopy IV Devices* chapter. | CMU PLL power down.<br>■ Asserted high—the CMU PLL is powered down.<br>■ De-asserted low—the CMU PLL is active and locks to the input reference clock.<br>Note: Asserting the `pll_powerdown` signal does not power down the REFCLK buffers. | Transceiver block |
| `coreclkout` | Output | Clock signal | HCell fabric-transceiver interface clock.<br>■ Generated by the `CMU0` clock divider in the transceiver block in ×4 bonded channel configurations.<br>■ Generated by the `CMU0` clock divider in the master transceiver block in ×8 bonded channel configurations.<br>■ Not available in non-bonded channel configurations.<br>■ Use to clock the write port of the transmitter phase compensation FIFOs in all bonded channels and to clock parallel data `tx_datain` from the HCell fabric into the transmitter phase compensation FIFO of all bonded channels.<br>■ Use to clock the read port of the receiver phase compensation FIFOs in all bonded channels with rate match FIFO enabled and to clock parallel data `rx_dataout` from the receiver phase compensation FIFOs of all bonded channels (with rate match FIFO enabled) into the HCell fabric. | Transceiver block |

Table 1–46 lists the ALTGX megafunction dynamic reconfiguration ports.

**Table 1–46. ALTGX Megafunction Ports: Dynamic Reconfiguration in HardCopy IV GX Devices**

| Port Name | Input/ Output | Clock Domain | Description | Scope |
|---|---|---|---|---|
| reconfig_clk | Input | Clock signal | Dynamic reconfiguration clock.<br>■ Also used for offset cancellation in all modes except PCIe mode.<br>■ If configured in **Transmitter only** mode—the frequency range is 2.5 MHz to 50 MHz.<br>■ If configured in **Receiver only** or **Receiver and Transceiver** mode—the frequency range of this clock is 37.5 MHz to 50 MHz. | |
| reconfig_togxb | Input | Asynchronous signal | From the dynamic reconfiguration controller. | |
| reconfig_fromgxb | Output | Asynchronous signal | To the dynamic reconfiguration controller. | |

Table 1–47 lists the ALTGX megafunction PCIe interface ports.

**Table 1–47. ALTGX Megafunction Ports: PCIe Interface in HardCopy IV GX Devices (Part 1 of 4)**

| Port Name | Input/ Output | Clock Domain | Description | Scope |
|---|---|---|---|---|
| **PCIe Interface (Available only in PCIe functional Mode)** | | | | |
| `powerdn` | Input | Asynchronous signal | PCIe power state control.<br>■ Functionally equivalent to the `powerdown[1:0]` signal defined in the PCIe specification revision 2.0.<br>■ The width of this signal is 2 bits and is encoded as follows:<br>  ■ 2'b00: P0—Normal Operation<br>  ■ 2'b01: P0s—Low Recovery Time Latency, Low Power State<br>  ■ 2'b10: P1—Longer Recovery Time Latency, Lower Power State<br>  ■ 2'b11: P2—Lowest Power State | Channel |
| `tx_ forcedispcompliance` | Input | Asynchronous signal | Force 8B/10B encoder to encode with a negative running disparity.<br>■ Functionally equivalent to the `txcompliance` signal defined in PCIe specification revision 2.0.<br>■ 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 |

**Table 1–47. ALTGX Megafunction Ports: PCIe Interface in HardCopy IV GX Devices (Part 2 of 4)**

| Port Name | Input/ Output | Clock Domain | Description | Scope |
|---|---|---|---|---|
| `tx_forceelecidle` | Input | Asynchronous signal | Force transmitter buffer to PCIe electrical idle signal levels.<br>■ Functionally equivalent to the `txelecidle` signal defined in the PCIe specification revision 2.0.<br>■ Available in the Basic mode. | Channel |
| `rateswitch` | Input | Asynchronous signal | PCIe rateswitch control.<br>■ 1'b0—Gen1 (2.5 Gbps)<br>■ 1'b1—Gen2 (5 Gbps) | |
| `tx_pipemargin` | Input | Asynchronous signal | Transmitter differential output voltage level control.<br>■ Functionally equivalent to the `txmargin` signal defined in the PCIe specification revision 2.0.<br>■ Available only in PCIe Gen2 configuration.<br>■ The width of this signal is 3 bits and is decoded as follows:<br>  ■ 3'b000—Normal Operating Range<br>  ■ 3'b001—Full Swing = 800 - 1200 mV<br>  ■ 3'b010—TBD<br>  ■ 3'b011—TBD<br>  ■ 3'b100—If last value, full Swing = 200 to 400 mV<br>  ■ 3'b101—If last value, full Swing = 200 to 400 mV<br>  ■ 3'b110—If last value, full Swing = 200 to 400 mV<br>  ■ 3'b111—If last value, full Swing = 200 to 400 mV | |
| `tx_pipedeemph` | Input | Asynchronous signal | Transmitter buffer de-emphasis level control.<br>■ Functionally equivalent to the `txdeemph` signal defined in the PCIe specification revision 2.0.<br>■ Available only in PCIe Gen2 configuration.<br>■ 1'b0: -6 dB de-emphasis<br>■ 1'b1:-3.5 dB de-emphasis | |
| `pipe8b10binvpolarity` | Input | Asynchronous signal | PCIe polarity inversion control.<br>■ Functionally equivalent to the `rxpolarity` signal defined in the PCIe specification revision 2.0.<br>■ Available only in PCIe mode.<br>■ When asserted high—the polarity of every bit of the 10-bit input data to the 8B/10B decoder gets inverted. | Channel |

**Table 1–47. ALTGX Megafunction Ports: PCIe Interface in HardCopy IV GX Devices (Part 3 of 4)**

| Port Name | Input/ Output | Clock Domain | Description | Scope |
|---|---|---|---|---|
| `tx_detectrxloopback` | Input | Asynchronous signal | Receiver detect or PCIe loopback control.<br>■ Functionally equivalent to the `txdetectrx/loopback` signal defined in the PCIe specification revision 2.0.<br>■ 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.<br>■ 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 1–112. | Channel |
| `pipestatus` | Output | — | PCIe receiver status port.<br>■ Functionally equivalent to the `rxstatus[2:0]` signal defined in the PCIe specification revision 2.0.<br>■ The width of this signal is 3 bits per channel. The encoding of receiver status on the `pipestatus` port is as follows:<br>■ 000—Received data OK<br>■ 001—1 skip added<br>■ 010—1 skip removed<br>■ 011—Receiver detected<br>■ 100—8B/10B decoder error<br>■ 101—Elastic buffer overflow<br>■ 110—Elastic buffer underflow<br>■ 111—Received disparity error | Channel |
| `pipephydonestatus` | Output | — | PHY function completion indicator.<br>■ Functionally equivalent to the `phystatus` signal defined in the PCIe specification revision 2.0.<br>■ Assert this signal high for one parallel clock cycle to communicate completion of several PHY functions, such as power state transition, receiver detection, and signaling rate change between Gen1 (2.5 Gbps) and Gen2 (5 Gbps). | Channel |

**Table 1–47. ALTGX Megafunction Ports: PCIe Interface in HardCopy IV GX Devices (Part 4 of 4)**

| Port Name | Input/ Output | Clock Domain | Description | Scope |
|---|---|---|---|---|
| `rx_pipedatavalid` | Output | — | Valid data and control on the `rx_dataout` and `rx_ctrldetect` ports indicator.<br>■ 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.<br>■ Functionally equivalent to the `rxelecidle` signal defined in the PCIe specification revision 2.0.<br>■ 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 1–75. Otherwise, it drives this signal low.<br>■ 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 1–48 lists the ALTGX megafunction reset and power down ports.

**Table 1–48. ALTGX Megafunction Ports: Reset and Power Down HardCopy IV GX Devices (Part 1 of 2)**

| Port Name | Input/ Output | Clock Domain | Description | Scope |
|---|---|---|---|---|
| `gxb_powerdown` | Input | Asynchronous signal.<br>For minimum pulse width requirements, refer to the *DC and Switching Characteristics of HardCopy IV Devices* chapter. | Transceiver block power down.<br>■ When asserted high—all digital and analog circuitry within the PCS, PMA, CMU channels, and the CCU of the transceiver block, is powered down.<br>■ 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.<br>■ When asserted high—the receiver PCS blocks are reset. For more information, refer to the *Reset Control and Power Down* chapter in the *Stratix IV Device Handbook*. | Channel |

**Table 1–48. ALTGX Megafunction Ports: Reset and Power Down HardCopy IV GX Devices (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. <br> ■ When asserted high—analog circuitry within the receiver PMA gets reset. For more information, refer to the *Reset Control and Power Down* chapter in the *Stratix IV Device Handbook*. | Channel |
| `tx_digitalreset` | Input | Asynchronous signal. Minimum pulse width is two parallel clock cycles. | Transmitter PCS reset. <br> ■ When asserted high, the transmitter PCS blocks are reset. For more information, refer to the *Reset Control and Power Down* chapter in the *Stratix IV Device Handbook*. | Channel |

Table 1–49 lists the ALTGX megafunction calibration block ports.

**Table 1–49. ALTGX Megafunction Ports: Calibration Block for HardCopy IV GX Devices**

| 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 1–50 for more information about some useful reference terms used in this chapter.

**Table 1–50. Reference Information (Part 1 of 2)**

| Terms Used in this Chapter | Useful Reference Points |
|---|---|
| (OIF) CEI PHY Interface Mode | page 1–103 |
| 8B/10B Decoder | page 1–40 |
| 8B/10B Encoder | page 1–11 |
| AEQ | page 1–24 |
| Auxiliary Transmit (ATX) PLL Block | page 1–113 |
| Basic (PMA Direct) Functional Mode | page 1–106 |
| Basic Functional Mode | page 1–55 |
| Built-In Self Test Modes | page 1–121 |
| Byte Ordering Block | page 1–42 |
| Byte Serializer | page 1–42 |
| Calibration Blocks | page 1–116 |
| Clock and Data Recovery Unit (CDR) | page 1–28 |
| CMU Channel Architecture | page 1–45 |
| CMU0 PLL | page 1–47 |
| CMU1 PLL | page 1–47 |

**Table 1–50. Reference Information (Part 2 of 2)**

| Terms Used in this Chapter | Useful Reference Points |
|---|---|
| CPRI and OBSAI | page 1–64 |
| Deserializer | page 1–32 |
| Deskew FIFO | page 1–39 |
| Deterministic Latency Mode | page 1–61 |
| EyeQ | page 1–26 |
| GIGE Mode | page 1–92 |
| Lock-to-Data (LTD) | page 1–29 |
| Lock-to-Reference (LTR) | page 1–28 |
| Low Latency PCS Datapath | page 1–56 |
| Offset Cancellation in the Receiver Buffer and Receiver CDR | page 1–31 |
| Parallel loopback | page 1–109 |
| PCIe Clock Switch Circuitry | page 1–30 |
| PCIe Mode | page 1–66 |
| PCIe Reverse Parallel Loopback | page 1–112 |
| Programmable Common Mode Voltage | page 1–22 |
| Programmable Pre-Emphasis | page 1–20 |
| Rate Match (Clock Rate Compensation) FIFO | page 1–39 |
| Receiver Bit Reversal | page 1–37 |
| Receiver Input Buffer | page 1–22 |
| Receiver Phase Compensation FIFO | page 1–43 |
| Receiver Polarity Inversion | page 1–36 |
| Reverse Serial Loopback | page 1–111 |
| Reverse serial Pre-CDR Loopback | page 1–111 |
| SATA and SAS options | page 1–60 |
| SDI Mode | page 1–100 |
| Serial Loopback | page 1–108 |
| Serial RapidIO Mode | page 1–104 |
| SONET/SDH Mode | page 1–95 |
| Transceiver Block Architecture | page 1–6 |
| Transceiver Channel Locations | page 1–2 |
| Transceiver Port Lists | page 1–123 |
| Transmitter Bit Reversal | page 1–16 |
| Transmitter Local Clock Divider Block | page 1–21 |
| Transmitter Output Buffer | page 1–18 |
| Transmitter Polarity Inversion | page 1–15 |
| TX Phase Compensation FIFO | page 1–8 |
| Word Aligner | page 1–33 |
| XAUI Mode | page 1–87 |

## Document Revision History

Table 1–51 lists the revision history for this chapter.

**Table 1–51. Document Revision History**

| Date | Version | Changes |
|---|---|---|
| January 2011 | 1.1 | ■ Updated and edited "Transceiver Channel Locations" on page 1–2.<br>■ Updated and edited "Transceiver Block Architecture" on page 1–6<br>■ Updated and edited "Auxiliary Transmit (ATX) PLL Block" on page 1–113.<br>■ Updated and edited "Calibration Blocks" on page 1–116.<br>■ Updated and edited "Built-In Self Test Modes" on page 1–121.<br>■ Minor text edits. |
| June 2009 | 1.0 | Initial release. |