HardCopy IV Device Handbook, Volume 1

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

HC4_H5V1-2.2

Contents

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

Section III. Hot Socketing and Testing

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

Chapter 10. IEEE 1149.1 (JTAG) Boundary Scan Testing in HardCopy IV Devices

Section IV. Power and Thermal Management

Chapter 11. Power Supply and Temperature Sensing Diode in HardCopy IV Devices

Additional Information

Chapter Revision Dates

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

Chapter 1 HardCopy IV Device Family Overview
Revised: *January 2010*
Part Number: *HIV51001-2.2*

Chapter 2 Logic Array Block and Adaptive Logic Module Implementation in HardCopy IV Devices
Revised: *December 2008*
Part Number: *HIV51002-1.0*

Chapter 3 DSP Block Implementation in HardCopy IV Devices
Revised: *December 2008*
Part Number: *HIV51003-1.0*

Chapter 4 TriMatrix Embedded Memory Blocks in HardCopy IV Devices
Revised: *January 2010*
Part Number: *HIV51004-2.1*

Chapter 5 Clock Networks and PLLs in HardCopy IV Devices
Revised: *January 2010*
Part Number: *HIV51005-2.1*

Chapter 6 HardCopy IV Device I/O Features
Revised: *January 2010*
Part Number: *HIV51006-2.1*

Chapter 7 External Memory Interfaces in HardCopy IV Devices
Revised: *January 20010*
Part Number: *HIV51007-2.1*

Chapter 8 High-Speed Differential I/O Interfaces and DPA in HardCopy IV Devices
Revised: *January 2010*
Part Number: *HIV51008-2.1*

Chapter 9 Hot Socketing and Power-On Reset in HardCopy IV Devices
Revised: *December 2008*
Part Number: *HIV51009-1.0*

Chapter 10 IEEE 1149.1 (JTAG) Boundary Scan Testing in HardCopy IV Devices
Revised: *June 2009*
Part Number: *HIV51010-2.0*

Chapter 11 Power Supply and Temperature Sensing Diode in HardCopy IV Devices
Revised: *January 2010*
Part Number: *HIV51011-1.1*

This section provides a complete overview of all features relating to the HardCopy® IV device family. HardCopy IV devices are Altera's latest generation of low-cost, high-performance, low power ASICs with pin-outs, densities, and architecture that complement Stratix® IV devices. This section includes the following chapters:

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

Revision History

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

1. HardCopy IV Device Family Overview

HIV51001-2.2

This chapter provides an overview of features available in the HardCopy IV device family. More details about these features can be found in their respective chapters.

HardCopy® IV ASICs are the only 40-nm system-capable ASICs designed with an FPGA design flow. Altera's fifth generation of HardCopy IV ASICs deliver low-cost and high-performance at low-power. Based on a 0.9-V, 40-nm process, the HardCopy IV family is supported by Stratix IV FPGAs, which have complementary pin-outs, densities, and architectures that deliver in-system, at-speed prototyping—resulting in first-time-right ASICs. The Quartus® II software provides a complete set of tools for designing the Stratix IV FPGA prototypes and HardCopy IV ASICs. One design, one RTL, one set of intellectual property, and one tool deliver both ASIC and FPGA implementations. Other front-end design tools from Synopsys and Mentor Graphics® are also supported.

To reduce risk, HardCopy IV device features, such as phase-locked loops (PLLs), embedded memory, transceivers, and I/O elements (IOEs), are functionally and electrically equivalent to the Stratix IV FPGA features. To reduce cost, Altera® HardCopy IV devices are customized using only two metal and two via layers. The combination of the Quartus® II software for design, Stratix IV FPGAs for in-system prototype and design verification, and HardCopy IV devices for high-volume production provides the fastest time to market, lowest total cost, and lowest risk system design and production solution to meet your business needs.

The HardCopy IV device family contains two variants optimized to meet different application needs:

- HardCopy IV GX transceiver ASICs—up to 11.5 M usable ASIC equivalent gates, 20,736 Kbits dedicated RAM, 1,288 18 × 18-bit multipliers, and 36 full-duplex clock data recovery (CDR)-based transceivers at up to 6.5 Gbps
- HardCopy IV E ASICs—up to 14.6 M usable ASIC equivalent gates, 18,792 Kbits dedicated RAM, and 1,288 18 × 18 bit multipliers

This chapter contains the following sections:

Features

HardCopy IV devices offer the following features:

- General
  - Fine-grained HCell architecture resulting in a low-cost, high-performance, low-power ASIC
  - Fully tested production-quality samples typically available 14 weeks from the date of your design submission
  - Design functionality the same as the Stratix IV FPGA prototype
- System performance and power
  - Core logic performance up to double that of the Stratix IV FPGA prototype
  - Power consumption reduction of typically 50% from the Stratix IV FPGA prototype
  - Robust on-chip hot socketing and power sequencing support
  - Support for instant-on or instant-on-after-50 ms power-up modes
  - I/O:GND:PWR ratio of 8:1:1 along with on-die and on-package decoupling for robust signal integrity

  
  The actual performance and power consumption improvements described in this data sheet are design-dependent.

- Transceivers (HardCopy IV GX family)
  - Up to 36 full-duplex CDR-based transceivers in HardCopy IV GX devices supporting data rates up to 6.5 Gbps
  - Dedicated circuitry to support physical layer functionality for popular serial protocols, such as PCI Express (PIPE) Gen1 and Gen2, Gigabit Ethernet, Serial RapidIO, SONET/SDH, XAUI/HiGig, (OIF) CEI-6G, SD/HD/3G-SDI, Fibre Channel, SFI-5, and Interlaken
  - Complete PCI Express (PIPE) protocol solution with embedded PCI Express hard IP blocks that implement PHY-MAC layer, Data Link layer, and Transaction layer functionality
  - Programmable transmitter pre-emphasis and receiver equalization circuitry to compensate for frequency-dependent losses in the physical medium
  - Typical physical medium attachment (PMA) power consumption of 100 mW at 3.125 Gbps and 135 mW at 6.375 Gbps per channel
- Logic and Digital Signal Processing (DSP)
  - 3.8 to 15 million usable gates for both logic and DSP functions (as shown in Table 1–1)
  - High-speed DSP functions supporting 9 × 9, 12 × 12, 18 × 18, and 36 × 36 multipliers, multiple accumulate functions, and finite impulse response (FIR) filters

- Internal memory
  - TriMatrix memory, consisting of three RAM block sizes to implement true dual-port memory and first-in first-out (FIFO) buffers
  - Up to 20, 736 Kbits RAM in embedded RAM blocks (including parity bits)
  - Memory logic array blocks (MLAB) implemented in HCell logic fabric
- Clock resources PLLs
  - Up to 16 global clocks, 88 regional clocks, and 88 peripheral clocks per device
  - Clock control block supporting dynamic clock network enable/disable and dynamic global clock network source selection
  - Up to 12 PLLs per device supporting PLL reconfiguration, clock switchover, programmable bandwidth, clock synthesis, and dynamic phase shifting
- I/O standards, external memory interface, and intellectual property (IP)
  - Support for numerous single-ended and differential I/O standards, such as LVTTL, LVCMOS, PCI, PCI-X, SSTL, HSTL, and LVDS
  - High-speed differential I/O support with serializer/deserializer (SERDES) and dynamic phase alignment (DPA) circuitry for 1.25 Gbps performance
  - Support for high-speed networking and communications bus standards, including SPI-4.2, SFI-4, SGMII, Utopia IV, 10 Gigabit Ethernet XSLI, Rapid I/O, and NPSI
  - Memory interface support with dedicated DQS logic on all I/O banks
  - Dynamic On-Chip Termination (OCT) with auto-calibration support on all I/O banks
  - Support for high-speed external memory interfaces, including DDR, DDR2, DDR3 SDRAM, RLDRAM II, QDR II, and QDR II+ SRAM on up to 20 modular I/O banks
  - Support for multiple intellectual property megafunctions from Altera MegaCore® functions and Altera Megafunction Partners Program (AMPP℠)
  - Nios® II embedded processor support
- JTAG—IEEE 1149.1 boundary scan testing (BST) support
- Packaging
  - Pin-compatible with Stratix IV FPGA prototypes
  - Up to 880 user I/O pins available
  - Flip chip, space-saving FineLine BGA packages available (Table 1–5)

Table 1–1 and Table 1–2 list the HardCopy IV ASIC devices and available features.

Table 1–1. HardCopy IV GX ASIC Features

HardCopy IV GX ASIC	Stratix IV GX FPGA Prototype	ASIC Equivalent Gates *(1)*	Transceivers 6.5 Gbps *(2)*	M9K Blocks	M144K Blocks	Total Dedicated RAM Bits (not including MLABs) *(3)*	18 × 18-Bit Multipliers (FIR Mode)	PLLs
HC4GX15	EP4SGX70	2.8 M	8, 0	462	16	6,462 Kb	384	3
	EP4SGX110	3.8 M	8, 0	660	16	8,244 Kb	512	3
	EP4SGX180	6.7 M	8, 0	660	20	8,820 Kb	920	3
	EP4SGX230	9.2 M	8, 0	660	22	9,108 Kb	1,288	3
	EP4SGX290	7.7 M	8, 0	660	24	9,396 Kb	832	2
	EP4SGX360	9.4 M	8, 0	660	24	9,396 Kb	1,040	2
HC4GX25	EP4SGX110	3.8 M	16, 0	660	16	8,244 Kb	512	4
	EP4SGX180	6.7 M	16, 8 *(6)*	936	20	11,304 Kb	920	6
	EP4SGX230	9.2 M	16, 8 *(6)*	936	22	11,592 Kb	1,288	6
	EP4SGX290	7.7 M	16, 8 *(6)*	936	36	13,608 Kb	832	6 *(4)*
	EP4SGX360	9.4 M	16, 8 *(6)*	936	36	13,608 Kb	1,040	6 *(4)*
	EP4SGX530	11.5 M	16, 8 *(6)*	936	36	13,608 Kb	1,024	6
HC4GX35	EP4SGX180	6.7 M	24, 12 *(7)*	950	20	11,430 Kb	920	8
	EP4SGX230	9.2 M	24, 12 *(7)*	1,235	22	14,283 Kb	1,288	8 *(5)*
	EP4SGX290	7.7 M	24, 12 *(7)*	936	36	13,608 Kb	832	8
	EP4SGX360	9.4 M	24, 12 *(7)*	1,248	48	18,144 Kb	1,040	8 *(5)*
	EP4SGX530	11.5 M	24, 12 *(7)*	1,280	64	20,736 Kb	1,024	8 *(5)*

Notes to Table 1–1:

(1) This is the number of ASIC-equivalent gates available in the HardCopy IV base array, shared between both adaptive logic module (ALM) logic and DSP functions from a Stratix IV FPGA prototype. The number of usable ASIC-equivalent gates is bounded by the number of ALMs in the companion Stratix IV FPGA device.

(2) The first number indicates the number of transceivers with PMA and PCS; the second number indicates the number of CMU (PMA Only) transceivers.

(3) MLAB RAMs are implemented with HCells in the HardCopy IV ASICs.

(4) This device has six PLLs in the F1152 package and four PLLs in the F780 package.

(5) This device has eight PLLs in the F1517 package and six PLLs in the F1152 package.

(6) Devices in the cost-optimized LF780 and LF1152 packages have 16 transceivers and no CMU transceiver. Devices in the performance-optimized FF1152 package have 16 transceivers and eight CMU transceivers.

(7) Devices in the F1152 package have 16 transceivers and eight CMU transceivers. Devices in the performance-optimized FF1517 package have 24 transceivers and 12 CMU transceivers.

Table 1–2. HardCopy IV E ASIC Features

HardCopy IV E ASIC	Stratix IV E Prototype Device	ASIC Equivalent Gates *(1)*	M9K Blocks	M144K Blocks	Total Dedicated RAM Bits (not including MLABs) *(2)*	18 × 18-Bit Multipliers (FIR Mode)	PLLs
HC4E25	EP4SE230	9.2 M	864	22	10,944 Kb	1,288	4
	EP4SE360	9.4 M	864	32	12,384 Kb	1,040	4
HC4E35	EP4SE360	9.4 M	1,248	48	18,144 Kb	1,040	8
	EP4SE530	11.5 M	1,280	48	18,432 Kb	1,024	12 *(3)*
	EP4SE820	14.6 M	1,320	48	18,792 Kb	960	12 *(3)*

Notes to Table 1–2:

(1) This is the number of ASIC-equivalent gates available in the HardCopy IV E base array, shared between both adaptive logic module (ALM) logic and DSP functions from a Stratix IV E FPGA prototype. The number of usable ASIC-equivalent gates is bounded by the number of ALMs in the companion Stratix IV E FPGA device.

(2) MLAB RAMs are implemented with HCells in the HardCopy IV ASICs.

(3) This device has 12 PLLs in the F1517 package and eight PLLs in the F1152 package.

HardCopy IV ASIC and Stratix IV FPGA Mapping Paths

HardCopy IV devices offer pin-to-pin compatibility with the Stratix IV prototype, making them drop-in replacements for the FPGAs. Therefore, the same system board and software developed for prototyping and field trials can be retained, enabling the lowest risk and fastest time-to-market for high-volume production.

HardCopy IV devices also offer non-socket replacement mapping paths to allow for further cost reduction. For example, you can map the EP4SE230 device in the 780-pin FBGA package to the HC4E25 device in the 484-pin FBGA package. Because the pin-out for the two packages are not the same, you will need a separate board design for the Stratix IV device and the HardCopy IV device.

For the non-socket replacement path, select I/Os in the Stratix IV device that can be mapped to the HardCopy IV device. Not all I/Os in the Stratix IV device are available in the HardCopy IV non-socket replacement device. Check the pin-out information for both the Stratix IV device and HardCopy IV device to ensure that the I/Os can be mapped, and select the companion device in the Quartus II project setting during design development. By selecting the companion device, the Quartus II software ensures that common resources and compatible I/Os are used during the mapping from the Stratix FPGA to the HardCopy ASIC.

There are a number of FPGA prototype choices for each HardCopy IV device, as listed in Table 1–3 and Table 1–4. To obtain the best value and the lowest system cost, architect your system to maximize silicon resource utilization.

Table 1–3. Stratix IV GX FPGA Prototype-to-HardCopy IV GX ASIC Mapping Paths

HardCopy IV GX ASIC		Stratix IV GX FPGA Prototype and Package																
		EP4SGX70	EP4SGX110		EP4SGX180			EP4SGX230			EP4SGX290			EP4SGX360			EP4SGX530	
Device	Package	F780	F780	F1152	F780	F1152	F1517	F780	F1152	F1517	H780	F1152	F1517	H780	F1152	F1517	H1152	H1517
HC4GX15	780-pin FineLine BGA	✓	✓	—	✓	—	—	✓	—	—	✓ *(1)*	—	—	✓ *(1)*	—	—	—	—
HC4GX25	780-pin FineLine BGA	—	—	—	—	—	—	—	—	—	✓ *(1)*	—	—	✓ *(1)*	—	—	—	—
	1152-pin FineLine BGA	—	—	✓	—	✓	—	—	✓	—	—	✓	—	—	✓	—	✓ *(1)*	—
HC4GX35	1152-pin FineLine BGA	—	—	—	—	—	—	—	✓	—	—	—	—	—	✓	—	✓ *(1)*	—
	1517-pin FineLine BGA	—	—	—	—	—	✓	—	—	✓	—	—	✓	—	—	✓	—	✓

Note to Table 1–3:

(1) The Hybrid FBGA package for Stratix IV GX FPGAs requires additional unused board space along the edges beyond the footprint, but its footprint is compatible with the regular FBGA package. HardCopy IV GX ASICs are in the regular FBGA packages.

Table 1–4. Stratix IV E FPGA Prototype-to-HardCopy IV E ASIC Mapping Paths

HardCopy IV E ASIC		Stratix IV E FPGA Prototype and Package						
		EP4SE230	EP4SE360		EP4SE530		EP4SE820	
Device	Package	F780	H780	F1152	H1152	H1517	H1152	H1517
HC4E25	484-pin FineLine BGA	✓ (1)	—	—	—	—	—	—
	780-pin FineLine BGA	✓	✓ (2)	—	—	—	—	—
HC4E35	1152-pin FineLine BGA	—	—	✓	✓ (2)	—	✓ (2)	—
	1517-pin FineLine BGA	—	—	—	—	✓ (2)	—	✓

Notes to Table 1–4:

(1) This mapping is a non-socket replacement path that requires a different board design for the Stratix IV E device and the HardCopy IV E device.

(2) The Hybrid FBGA package for the Stratix IV E FPGAs requires additional unused board space along the edges beyond the footprint, but its footprint is compatible with the regular FBGA package. The HardCopy IV E ASICs are in the regular FBGA packages.

Three different FineLine BGA package substrate options are available for the HardCopy IV devices:

- Performance-optimized flip chip package (F)
- Cost-optimized flip chip package (L, LA)
- Low-cost wire bond package (W)—available for HardCopy IV E ASICs only

All three package types support direct replacement of the Stratix IV FPGA prototype. The performance-optimized flip chip package supports equivalent performance and the same number of I/Os as the corresponding FPGA prototype. The cost-optimized flip chip package uses a substrate with fewer layers and no on-package decoupling (OPD) capacitors to offer a low-cost package option. The performance is reduced from that of the FPGA prototype. However, the number of available I/Os remains the same. The wire bond package offers another low-cost package option, but with the trade-off of reduced performance and fewer available I/Os.

If you are going to use the low-cost wire bond package, make sure your design uses I/Os that are available in that package.

For HardCopy IV E non-socket replacement devices, only the performance-optimized flip chip package and the low-cost wire bond package are supported.

Table 1–5 and Table 1–6 show the available packages for HardCopy IV devices.

Table 1–5. HardCopy IV GX and Stratix IV GX Package, I/O Pin Count, LVDS Pair Count, and Transceiver Mapping *(Note 1)*, *(2)*

HardCopy IV GX ASIC	**LAF780** *(3)*	**LF780** *(4)*	**LF1152** *(5)*	**FF1152** *(5)*			**FF1517** *(6)*
HC4GX15	372, 28, 8+0	257, 0, 8+0	—	—	—	—	—
HC4GX25	—	289, 0, 16+0	564, 44, 16+0	564, 44, 16+8	—	564, 44, 16+8	—
HC4GX35	—	—	—	—	564, 44, 16+8	564, 44, 16+8	744, 88, 24+12
Companion Mapping	↕	↕	↕	↕	↕	↕	↕
Stratix IV GX FPGA Prototype	**F780**	**H780**	**F1152**	**F1152**	**F1152**	**H1152**	**F1517**
EP4SGX70	372, 28, 8+0	—	—	—	—	—	—
EP4SGX110	372, 28, 8+0	—	372, 28, 16+0	—	—	—	—
EP4SGX180	372, 28, 8+0	—	564, 44, 16+0	564, 44, 16+8	—	—	744, 88, 24+12
EP4SGX230	372, 28, 8+0	—	564, 44, 16+0	564, 44, 16+8	564, 44, 16+8	—	744, 88, 24+12
EP4SGX290	—	289, 0, 16+0	564, 44, 16+0	564, 44, 16+8	—	—	744, 88, 24+12
EP4SGX360	—	289, 0, 16+0	564, 44, 16+0	564, 44, 16+8	564, 44, 16+8	—	744, 88, 24+12
EP4SGX530	—	—	—	—	—	564, 44, 16+8	744, 88, 24+12

Notes to Table 1–5:

(1) The numbers in the table indicate I/O pin count, full duplex LVDS pairs, and transceivers (PMA and PCS) + CMU transceivers (PMA Only).

(2) The first letter (two letters in the LA package) in the HardCopy IV GX package name refers to the following: F–Performance-optimized flip chip package, L or LA –Cost-optimized flip chip package.

(3) The I/O pin count for the LAF780 package includes the four dedicated clock inputs (`CLK1n`, `CLK1p`, `CLK3n`, and `CLK3p`).

(4) The I/O pin count for the LF780 package includes one dedicated clock input (`CLK1p`).

(5) The I/O pin count for the F1152 package includes the four dedicated clock inputs (`CLK1n`, `CLK1p`, `CLK10n`, and `CLK10p`).

(6) The I/O pin count for the F1517 package includes the eight dedicated clock inputs (`CLK1n`, `CLK1p`, `CLK3n`, `CLK3p`, `CLK8n`, `CLK8p`, `CLK10n`, and `CLK10p`).

Table 1–6. HardCopy IV E and Stratix IV E Package, I/O Pin Count, and LVDS Pair Count Mapping *(Note 1), (2), (3), (4)*

HardCopy IV E ASIC	WF484 FF484	WF780		FF780		LF1152 FF1152		LF1517 FF1517	
HC4E25	296, 48	392, 48		488, 56		—		—	
HC4E35	—	—		—		744, 88		880, 88	
Companion Mapping	↕	↕		↕		↕		↕	
Stratix IV E FPGA Prototype	**F780**	**F780**	**H780**	**F780**	**H780**	**F1152**	**H1152**	**F1517**	**H1517**
EP4SE230	488, 56	488, 56	—	488, 56	—	—	—	—	—
EP4SE360	—	—	488, 56	—	488, 56	744, 88	—	—	—
EP4SE530	—	—	—	—	—	—	744, 88	—	976, 112
EP4SE820	—	—	—	—	—	—	744, 88	—	976, 112

Notes to Table 1–6:

(1) The numbers in the table indicate I/O pin count, full duplex LVDS pairs.

(2) The first letter in the HardCopy IV E package name refers to the following: F–Performance-optimized flip chip package, L–Cost optimized flip-chip package, W–Low-cost wire bond package.

(3) For the F484, F780, and F1152 packaged devices, the I/O pin counts include the eight dedicated clock inputs (`CLK1p`, `CLK1n`, `CLK3p`, `CLK3n`, `CLK8p`, `CLK8n`, `CLK10p`, and `CLK10n`) that you can use for inputs.

(4) For the F1517 packaged device, the I/O pin count includes the eight dedicated clock inputs (`CLK1p`, `CLK1n`, `CLK3p`, `CLK3n`, `CLK8p`, `CLK8n`, `CLK10p`, and `CLK10n`) and the eight dedicated corner PLL clock inputs (`PLL_L1_CLKp`, `PLL_L1_CLKn`, `PLL_L4_CLKp`, `PLL_L4_CLKn`, `PLL_R4_CLKp`, `PLL_R4_CLKn`, `PLL_R1_CLKp`, and `PLL_R1_CLKn`) that you can use for data inputs.

Differences Between HardCopy IV and Stratix IV Devices

HardCopy IV devices have several architectural differences from Stratix IV devices. When implementing your design and laying out your board, consider these differences. Use this information to ensure that your design maps from the Stratix IV FPGA to the HardCopy IV ASIC:

- Configuration is not required for HardCopy IV devices; therefore, the following Stratix IV features are not supported:
  - Programming modes and features such as remote update and Programmers Object File (**.pof**) encryption
  - Cyclical redundancy check (CRC) for configuration error detection
  - 256-bit (AES) volatile and non-volatile security keys to protect designs
  - JTAG instructions used for configuration.
- FPGA configuration emulation mode is not supported in HardCopy IV devices.
- Boundary scan (BSCAN) chain length is different and varies with device density.
- HardCopy IV devices contain up to a maximum of 20 I/O banks; Stratix IV devices contain up to a maximum of 24 I/O.

- Memory Initialization Files (**.mif**) for embedded memories used as RAM are not supported. The **.mifs** for memories used as ROM are supported, because the data are mask-programmed into the memory cells.
- Stratix IV LAB, MLAB, and DSP functions are implemented with HCells in HardCopy IV devices instead of dedicated blocks. However, they remain functionally and electrically equivalent between the FPGAs and the HardCopy ASICs.
- Stratix IV programmable power technology is not supported in HardCopy IV devices. However, the HardCopy IV architecture offers performance similar to Stratix IV devices with significantly lower power consumption.
- There are eight on-chip termination (OCT) calibration blocks in HardCopy IV devices instead of up to 10 OCT calibration blocks in Stratix IV devices.

Architectural Features

This section describes the architectural features of HardCopy IV ASICs.

Logic Array Block and Adaptive Logic Module Function Support

HardCopy IV devices fully support the Stratix IV LAB and ALM functions. The basic building blocks of Stratix IV LABs are composed of ALMs that you can configure to implement logic, arithmetic, and register functions. Each LAB consists of 10 ALMs, carry chains, shared arithmetic chains, LAB control signals, local interconnect, and register chain connection lines.

In HardCopy IV devices, the basic building blocks of the core array are HCells, which are a collection of logic transistors connected together to provide the same functionality as the Stratix IV LABs and ALMs. The Quartus II software maps these LAB and ALM functions to HCell macros, which define how the HCells are connected together in the HardCopy IV core array. Only HCells required to implement the customer design are used, and unused HCells are powered down. This allows efficient use of the core fabric and offers significant static power savings.

The Stratix IV LAB derivative, called MLAB, is also supported in HardCopy IV devices. MLAB adds static random access memory (SRAM) capability to the LAB and can provide a maximum of 640 bits of simple dual-port SRAM. Like the LAB functions, the Quartus II software maps MLAB functions to HCell macros in HardCopy IV devices to provide the same Stratix IV functionality.

For more information about LABs and ALMs, refer to the *Logic Array Block and Adaptive Logic Module Implementation in HardCopy IV Devices* chapter in volume 1 of the *HardCopy IV Device Handbook*.

For more information about MLAB modes, features, and design considerations, refer to the *TriMatrix Embedded Memory Blocks in HardCopy IV Devices* chapter in volume 1 of the *HardCopy IV Device Handbook*.

DSP Function Support

HardCopy IV devices fully support the DSP block functions of Stratix IV devices. Complex systems such as WiMAX, 3GPP WCDMA, CDMA2000, voice over Internet protocol (VoIP), H.264 video compression, and high-definition television (HDTV) require high-performance DSP circuits to handle large amounts of data with high throughput. These system designs typically use DSP to implement finite impulse response (FIR) filters, complex FIR filters, infinite impulse response (IIR) filters, fast Fourier transform (FFT) functions, and discrete cosine transform (DCT) functions.

In HardCopy IV devices, these DSP block functions are implemented with HCells. The Quartus II software maps the Stratix IV DSP functions to HCell macros in HardCopy IV devices, preserving the same functionality. Implementing DSP functions using HCells also allows efficient use of the HardCopy IV device core fabric and offers significant static power savings.

HardCopy IV devices support all Stratix IV DSP configurations (9 × 9, 12 × 12, 18 × 18, and 36 × 36 multipliers) and block features, such as dynamic sign controls, dynamic addition and subtraction, dynamic rounding and saturation, and dynamic input shift registers. All five operational modes of the Stratix IV DSP block are supported:

- Independent multiplier (9 × 9, 12 × 12, 18 × 18, and 36 × 36)
- Two-multiplier adder
- Four-multiplier adder
- Multiply accumulate
- Shift mode

For more information about DSP blocks, refer to the *DSP Block Implementation in HardCopy IV Devices* chapter in volume 1 of the *HardCopy IV Device Handbook*.

TriMatrix Embedded Memory Blocks

TriMatrix embedded memory blocks provide three different sizes of embedded SRAM to efficiently address the needs of HardCopy IV ASIC designs. TriMatrix memory includes the following types of blocks:

- 640-bit MLAB blocks optimized to implement filter delay lines, small FIFO buffers, and shift registers. MLAB blocks are implemented in HCell macros.
- 9-Kbit M9K blocks that can be used for general purpose memory applications.
- 144-Kbit M144K blocks that are ideal for processor code storage, packet, and video frame buffering.

You can configure each embedded memory block independently to be a single- or dual-port RAM, ROM, or shift register using the Quartus II MegaWizard™ Plug-In Manager. Multiple blocks of the same type can also be stitched together to produce larger memories with minimal timing penalty. TriMatrix memory provides up to an equivalent of 20.3 Mbits of dedicated, embedded SRAM.

For more information about TriMatrix memory blocks, modes, features, and design considerations, refer to the *TriMatrix Embedded Memory Blocks in HardCopy IV Devices* chapter in volume 1 of the *HardCopy IV Device Handbook*.

Clock Networks and PLLs

HardCopy IV devices provide dedicated global clock networks (GCLKs), regional clock networks (RCLKs), and periphery clock networks (PCLKs). These clocks are organized into a hierarchical clock structure that provides up to 192 unique clock domains (16 GCLK + 88 RCLK + 88 PCLK) within the HardCopy IV device and allows up to 60 unique GCLK/RCLK/PCLK clock sources (16 GCLK + 22 RCLK + 22 PCLK) per device quadrant.

HardCopy IV devices deliver abundant PLL resources, with up to 12 PLLs per device and up to 10 outputs per PLL. You can configure each output independently, creating a unique, customizable clock frequency with no fixed relation to any other input or output clock. Inherent jitter filtration and fine granularity control over multiply, divide ratios, and dynamic phase-shift reconfiguration provide the high-performance precision required in today's high-speed applications. HardCopy IV PLLs are feature-rich, supporting advanced capabilities such as clock switchover, reconfigurable phase shift, PLL reconfiguration, and reconfigurable bandwidth. You can use PLLs for general-purpose clock management, supporting multiplication, phase shifting, and programmable duty cycles. HardCopy IV PLLs also support external feedback mode, spread-spectrum input clock tracking, and post-scale counter cascading.

For more information about clock networks and PLLs, refer to the *Clock Networks and PLLs in HardCopy IV Devices* chapter in volume 1 of the *HardCopy IV Device Handbook*.

I/O Banks and I/O Structure

HardCopy IV devices contain up to 20 modular I/O banks, each containing 24, 32, 40, or 48 I/Os (not including dedicated clock inputs). The left- and right-side I/O banks contain circuitry to support external memory interfaces and high-speed differential I/O interfaces capable of performance at up to 1.25 Gbps. The top and bottom I/O banks also contain circuitry to support external memory interfaces.

HardCopy IV devices support a wide range of industry I/O standards, including single-ended, voltage referenced single-ended, and differential I/O standards. The HardCopy IV I/O supports bus hold, pull-up resistor, slew rate, output delay control, and open-drain output. HardCopy IV devices also support on-chip series (R_S) and on-chip parallel (R_T) termination with auto calibration for single-ended I/O standards. The left and right I/O banks support on-chip differential termination (R_D) to meet LVDS I/O standards. Bidirectional I/O pins on all I/O banks also support Dynamic OCT.

For more information about I/O features, refer to the *HardCopy IV Device I/O Features* chapter in volume 1 of the *HardCopy IV Device Handbook*.

External Memory Interfaces

The HardCopy IV I/O structure is equivalent to the Stratix IV I/O structure, providing high-performance support for existing and emerging external memory standards such as DDR, DDR2, DDR3, QDRII, QDRII+, and RLDRAM II.

Packed with features such as dynamic on-chip termination, trace mismatch compensation, read and write leveling, half-rate registers, and 4- to 36-bit DQ group widths, HardCopy IV I/Os supply the built-in functionality required for rapid and robust implementation of external memory interfaces. Double data-rate support is found on all sides of the HardCopy IV device. HardCopy IV devices provide an efficient architecture to quickly and easily fit wide external memory interfaces precisely.

A self-calibrating soft IP core (ALTMEMPHY) optimized to take advantage of HardCopy IV device I/Os along with the Quartus II timing analysis tool (the TimeQuest Timing Analyzer) provides the total solution for the highest reliable frequency of operation across process, voltage, and temperature (PVT).

For more information about external memory interfaces, refer to the *External Memory Interfaces in HardCopy IV Devices* chapter in volume 1 of the *HardCopy IV Device Handbook*.

High-Speed Differential I/O Interfaces with DPA

HardCopy IV devices contain dedicated circuitry for supporting differential standards at speeds up to 1.25 Gbps. High-speed differential I/O circuitry supports the following high-speed I/O interconnect standards and applications:

- Utopia IV
- SPI-4.2
- SFI-4
- 10 Gigabit Ethernet XSLI
- Rapid I/O
- NPSI

HardCopy IV devices support 2×, 4×, 6×, 7×, 8×, and 10× SERDES modes for high-speed differential I/O interfaces, and 4×, 6×, 7×, 8×, and 10× SERDES modes when using the dedicated DPA circuitry. DPA minimizes bit errors, simplifies PCB layout and timing management for high-speed data transfer, and eliminates channel-to-channel and channel-to-clock skews in high-speed data transmission systems. The Stratix IV soft CDR function can also be implemented using HCells in HardCopy IV devices, enabling low-cost 1.25-Gbps clock-embedded serial links.

HardCopy IV devices have the following dedicated circuitry for high-speed differential I/O support:

- Differential I/O buffer
- Transmitter serializer
- Receiver deserializer
- Data realignment
- Dynamic phase aligner (DPA)
- Soft CDR functionality

- Synchronizer (FIFO buffer)
- PLLs

For more information about dedicated circuitry for high-speed differential support, refer to the *High Speed Differential I/O Interfaces with DPA in HardCopy IV Devices* chapter in volume 1 of the *HardCopy IV Device Handbook*.

Hot Socketing and Power-On Reset

HardCopy IV devices offer hot socketing, which is also known as hot plug-in or hot swap, and power sequencing support without the use of any external devices. On-chip hot socketing and power-sequencing support ensures proper device operation independent of the power-up sequence. You can insert or remove a HardCopy IV board during system operation without causing undesirable effects to the running system bus or the board itself.

The hot socketing feature also makes it easier to use HardCopy IV devices on PCBs that contain a mixture of 3.0-V, 2.5-V, 1.8-V, 1.5-V, and 1.2-V devices. With the HardCopy IV hot socketing feature, you do not need to ensure a proper power-up sequence for each device on the board.

HardCopy IV devices have a maximum V_{CCIO} voltage of 3.0 V, but can tolerate a 3.3-V input level.

For more information about hot socketing, refer to the *Hot Socketing and Power-On Reset in HardCopy IV Devices* chapter in volume 1 of the *HardCopy IV Device Handbook*.

IEEE 1149.1 (JTAG) Boundary Scan Testing

HardCopy IV devices support the JTAG IEEE Std. 1149.1 specification. The Boundary-Scan Test (BST) architecture offers the capability to both test pin connections without using physical test probes and capture functional data while a device is operating normally. Boundary-scan cells in the HardCopy IV device can force signals onto pins or capture data from the pin or core signals. Forced test data is serially shifted into the boundary-scan cells. Captured data is serially shifted out and externally compared to expected results.

For more information about JTAG, refer to the *IEEE 1149.1 (JTAG) Boundary Scan Testing in HardCopy IV Devices* chapter in volume 1 of the *HardCopy IV Device Handbook*.

Signal Integrity

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

- 8:1:1 user I/O/GND/V_{CC} ratio to reduce loop inductance in the package
- Dedicated power supply for each I/O bank, with an I/O limit of 24 to 48 I/Os per bank to help limit simultaneous switching noise (SSN)

- Slew-rate support with up to four settings to match the desired I/O standard, control noise, and overshoot
- Output-current drive strength support with up to four settings to match desired I/O standard performance
- Output-delay support to control rise and fall times and adjust duty cycle, compensate for skew, and reduce simultaneous switching output (SSO) noise
- Dynamic OCT with auto-calibration support for series and parallel OCT and differential OCT support for LVDS I/O standard on the left and right banks

The supported settings for slew-rate control, output-current drive strength, and output-delay control are mask-programmed into the HardCopy IV devices and cannot be changed after the silicon is fabricated.

For more information about signal integrity support in the Quartus II software, refer to the *Quartus II Handbook*.

Software Support and Part Number Information

This section describes HardCopy IV device software support and part number information.

Software Support

HardCopy IV devices are supported by the Altera Quartus II design software, which provides a comprehensive environment for system-on-chip (SOC) design. The Quartus II software includes HDL and schematic design entry, compilation and logic synthesis, full simulation and advanced timing analysis, SignalTap® II logic analyzer, and device configuration.

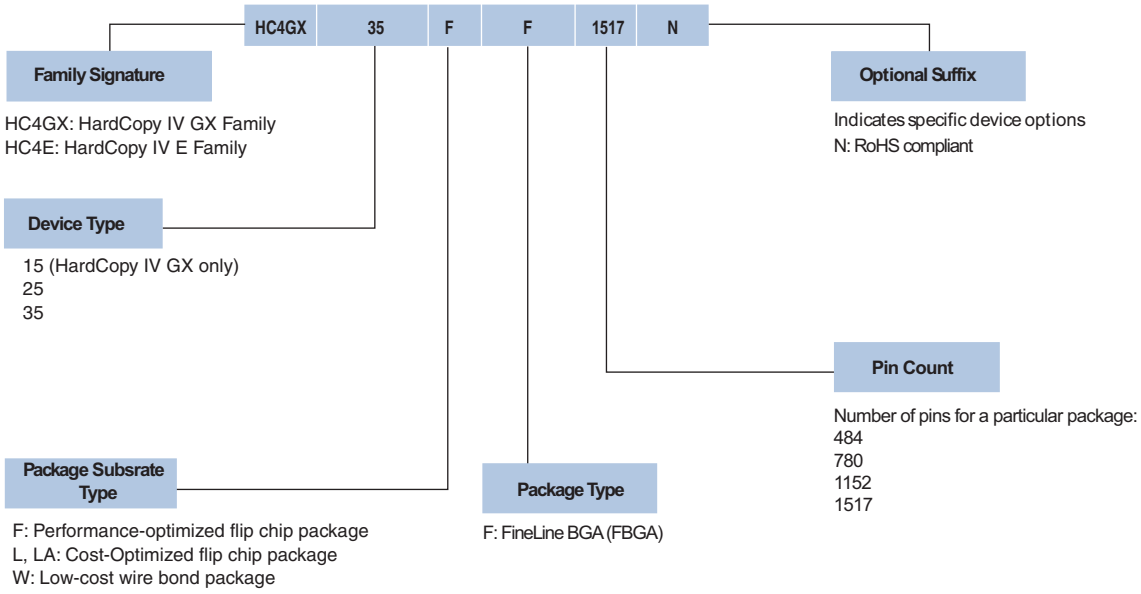
For more information about the Quartus II software features, refer to the *Quartus II Handbook*.

The Quartus II software supports the Windows and Linux Red Hat operating systems. You can obtains the specific operating system for the Quartus II software from the Quartus II Readme.txt file or http://www.altera.com/support/software/os_support/oss_index.html. It also supports seamless integration with industry-leading EDA tools through the NativeLink interface.

Part Number Information

Figure 1–1 shows the generic part number for HardCopy IV devices.

Figure 1–1. HardCopy IV Device Part Number Information



Document Revision History

Table 1–7 shows the revision history for this chapter.

Table 1–7. Document Revision History

Date	Version	Changes Made
January 2010	2.2	■ Updated Table 1–2. ■ Updated Table 1–4. ■ Updated Table 1–6. ■ Minor text edits.
July 2009	2.1	Updated “Features” on page 1–2
June 2009	2.0	■ Updated “Introduction” on page 1–1. ■ Updated “Features” on page 1–2. ■ Updated Table 1–1. ■ Added Table 1–2 ■ Updated Table 1–3. ■ Added Table 1–4 ■ Added Table 1–5. ■ Added Table 1–6 ■ Updated Figure 1–1.
December 2008	1.0	Initial release.

2. Logic Array Block and Adaptive Logic Module Implementation in HardCopy IV Devices

HIV51002-1.0

This chapter describes how the Stratix® IV's logic array blocks (LABs) and memory logic array blocks (MLABs) are implemented in a HardCopy® IV device. In Stratix IV devices, the core fabric consists of an array of LABs and MLABs. LABs and MLABs are composed of adaptive logic modules (ALMs) that are configurable and can implement various logic, arithmetic, and register functions of a customer design. In addition, MLABs can implement memory functions. By comparison, the core fabric in HardCopy IV devices are built using an array of flexible, fine-grain architecture blocks called HCells that can efficiently implement all the functionality of the ALMs, LABs, and MLABs. HardCopy IV devices offer improved performance and significant static power savings compared to Stratix IV FPGA prototype devices because only the HCells required to implement the customer design are used, while the unused HCells are powered down.

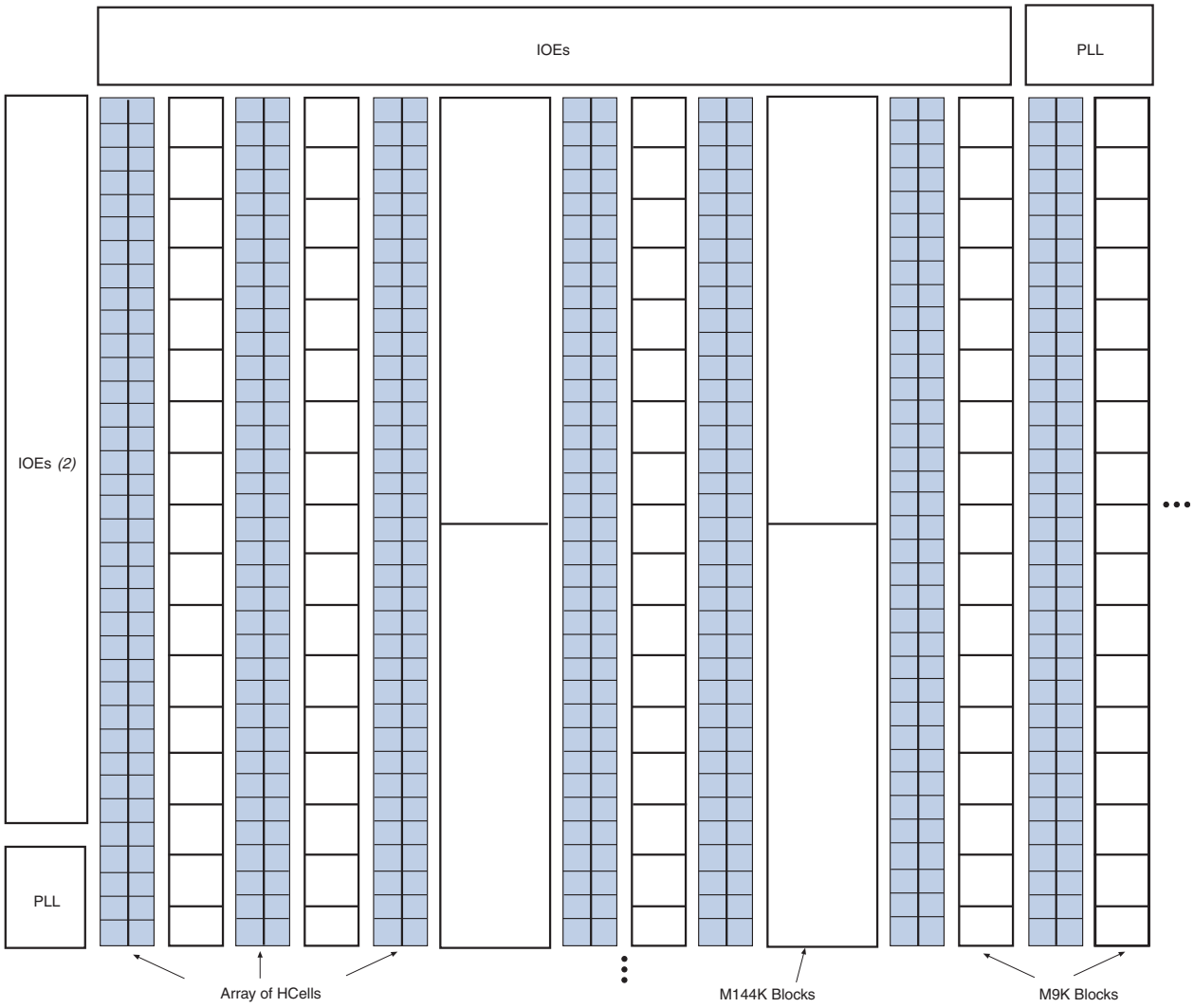
For more information about ALMs, LABs, and MLABs, refer to the *Logic Array Blocks and Adaptive Logic Modules in Stratix IV Devices* chapter in volume 1 of the *Stratix IV Device Handbook*.

This chapter contains the following sections:

HCells

HCells are a collection of logic transistors based on 0.9-V, 40-nm process technology. The construction of logic using HCells allows flexible functionality such that when HCells are combined, all viable logic combinations of Stratix IV functionality are replicated. These HCells constitute the array of the HCell area, as shown in Figure 2–1. Only the HCells needed to implement the design are assembled together, which optimizes HCell used. The unused area of the HCell logic fabric is powered down, resulting in significant static power savings compared with the Stratix IV FPGA prototype.

Figure 2–1. Example Block Diagram of HardCopy IV Device *(Note 1)*



Notes to Figure 2–1:

(1) Figure 2–1 shows a graphical representation of the device floorplan. A detailed floorplan is available in the Quartus® II software.

(2) IOEs represents I/O elements.

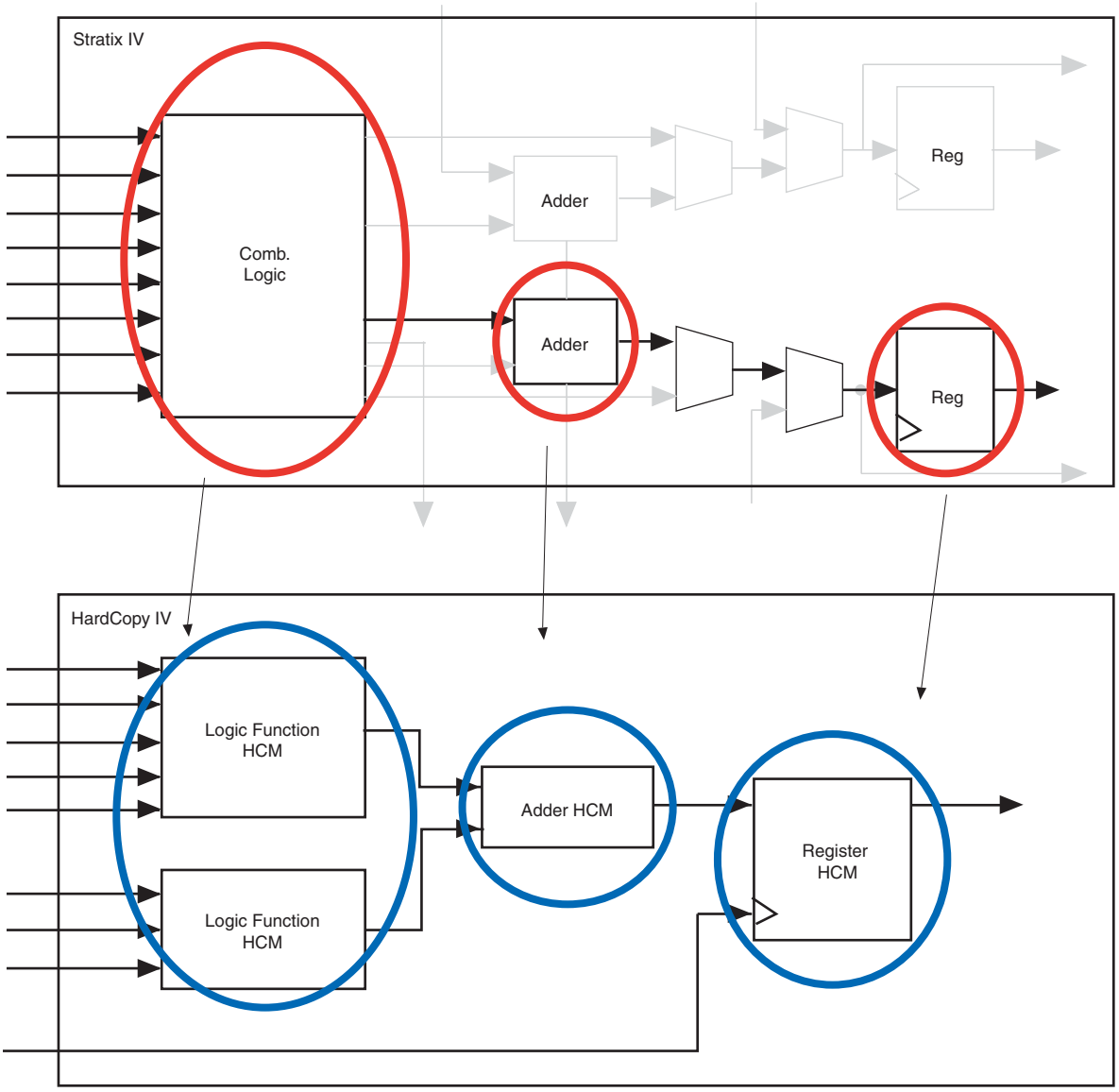
ALM and LAB Function Implementation

The Quartus II software uses a library of pre-characterized HCell macros (HCMs) to place Stratix IV ALM configurations into the HardCopy IV HCell-based logic fabric. An HCell macro defines how a group of HCells connect within the array. HCell macros can construct all combinations of combinational logic, adder, and register functions that can be implemented by a Stratix IV ALM. You can use HCells that are not used for ALM configurations to implement MLAB and DSP block functions.

For more details about implementing DSP block functions using HCells, refer to the *DSP Block Implementation in HardCopy IV Devices* chapter in volume 1 of the *HardCopy IV Device Handbook*.

Based on design requirements, the Quartus II software chooses the appropriate HCell macros to implement design functionality. For example, Stratix IV ALMs offer flexible look-up table (LUT) blocks, registers, arithmetic blocks, and LAB-wide control signals. In HardCopy IV devices, if your design requires these architectural elements, the Quartus II synthesis tool maps the design to the appropriate HCell macros, resulting in improved design performance compared to the Stratix IV FPGA prototype, as shown in Figure 2–2.

Figure 2–2. Example of ALM Functions Mapped to HCell Macros



MLAB Function Implementation

In Stratix IV devices, the MLAB is a LAB derivative that you can configure to support up to a maximum of 640 bits of simple dual-port static random access memory (SRAM). Similar to the LAB, each MLAB consists of ten ALMs and can implement all the functionality of the LAB in addition to the memory function. In HardCopy IV devices, the MLAB functions are mapped to HCell macros that provide the same memory functionality.

For more information about memory implementation using MLABs, refer to the *TriMatrix Embedded Memory Blocks in Stratix IV Devices* chapter in volume 1 of the *Stratix IV Device Handbook*.

For more information about HardCopy IV memory support, refer to the *TriMatrix Embedded Memory Blocks in HardCopy IV Devices* chapter in volume 1 of the *HardCopy IV Device Handbook*.

Conclusion

In HardCopy IV devices, the basic building block of the core array is the HCell. HCells are connected together to form HCell macros that can implement all the functionality of the ALMs, LABs, and MLABs in the Stratix IV devices. Only HCells required to implement the design are used, while unused HCells are powered down. This allows the core fabric to be efficiently used and offers significant static power savings compared to the Stratix IV FPGA prototype devices.

Referenced Documents

This chapter references the following documents:

- *DSP Block Implementation in HardCopy IV Devices* chapter in volume 1 of the *HardCopy IV Device Handbook*
- *Logic Array Blocks and Adaptive Logic Modules in Stratix IV Devices* chapter in volume 1 of the *Stratix IV Device Handbook*
- *TriMatrix Embedded Memory Blocks in HardCopy IV Devices* chapter in volume 1 of the *HardCopy IV Device Handbook*
- *TriMatrix Embedded Memory Blocks in Stratix IV Devices* chapter in volume 1 of the *Stratix IV Device Handbook*

Document Revision History

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

Table 2–1. Document Revision History

Date	Version	Changes Made
December 2008	1.0	Initial release.

3. DSP Block Implementation in HardCopy IV Devices

HIV51003-1.0

Stratix® IV devices have dedicated high-performance digital signal processing (DSP) blocks that are distributed throughout the core fabric. These hard-wired DSP blocks are ideal for applications such as high performance computing (HPC), video compression/decompression, and voice over internet protocol (VoIP). Such applications typically require a large number of mathematical computations. Stratix IV DSP blocks consist of a combination of dedicated elements that perform multiplication, addition, subtraction, accumulation, summation, and dynamic shift operations. In HardCopy® IV devices, these DSP functions are constructed using HCells instead of dedicated DSP blocks. HCells allow HardCopy IV devices to have the same functionality as Stratix IV DSP blocks. In addition, DSP blocks implemented with HCells provide significant static power savings because only the HCells needed to implement the functions are used.

This chapter contains the following sections:

DSP Function Implementation

A Stratix IV DSP block consists of an input register bank, multiplier adders, pipeline register bank, second-stage adders/accumulator, round and saturation units, and second adder register and output register bank. In the HardCopy IV devices, HCells make up the device logic fabric. HCells are a collection of logic transistors that are connected together to provide the same DSP functions as the Stratix IV DSP blocks. HCells are also used to implement the Stratix IV adaptive logic module (ALM) and logic array block (LAB) functions in the HardCopy IV devices.

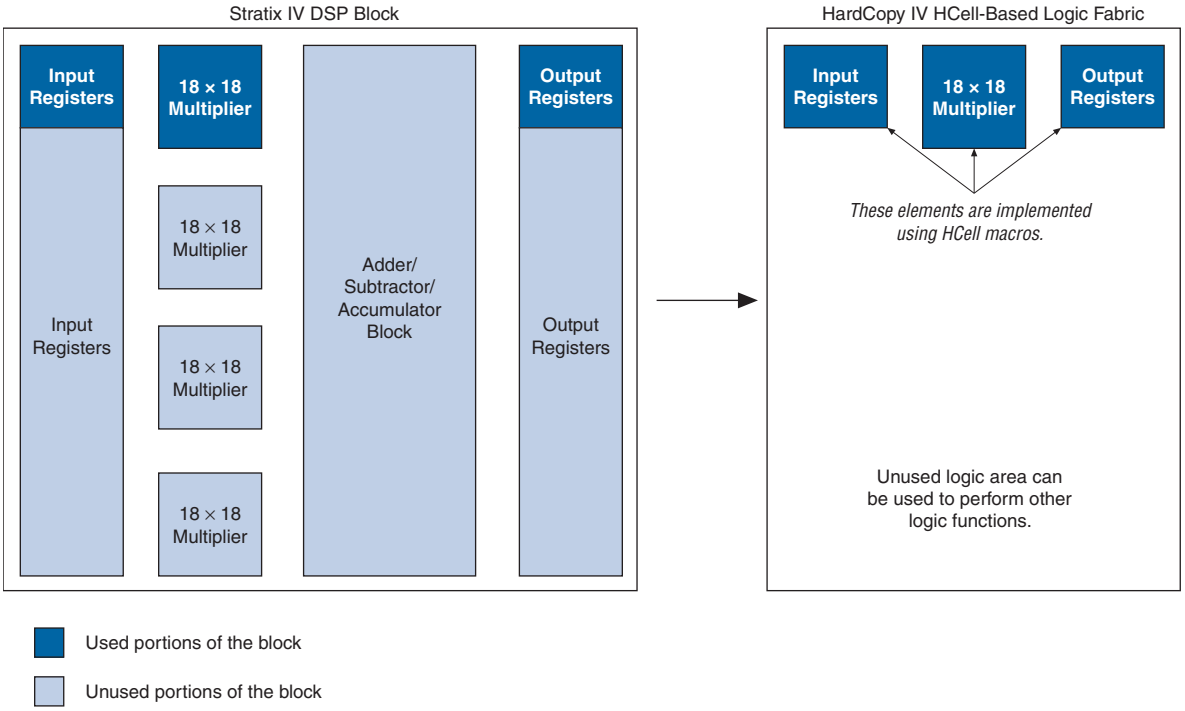
For more information about ALM, LAB, and memory logic array block (MLAB) implementation in HardCopy IV devices, refer to the *Logic Array Block and Adaptive Logic Module Implementation in HardCopy IV Devices* chapter.

The Quartus® II software uses a library of pre-characterized HCell macros to place Stratix IV DSP configurations into the HardCopy IV HCell-based logic fabric. An HCell macro (HCM) defines how a group of HCells are connected together. Based on design requirements, the Quartus II software chooses the appropriate DSP HCell macros to implement the DSP functionality. In HardCopy IV devices, HCell macros implement Stratix IV DSP block functionality with area efficiency and performance on par with the dedicated DSP blocks in Stratix IV devices.

Only HCells that are required to implement the design's DSP functions are enabled. HCells not needed for DSP functions can be used for ALM configurations, which results in efficient logic usage. In addition to area management, the placement of these HCell macros allows for optimized routing and performance.

An example of efficient logic area usage is evident when comparing the 18 × 18 independent multiplier implementation in Stratix IV devices using the dedicated DSP block versus the implementation in HardCopy IV devices using HCells. If the Stratix IV DSP function only calls for one 18 × 18 multiplier, the other three 18 × 18 multipliers and the DSP block's adder output block are not used, as shown in Figure 3–1. In HardCopy IV devices, the HCell-based logic fabric that is not used for DSP functions can be used to implement other combinational logic, adder, register, and MLAB functions.

Figure 3–1. Stratix IV DSP Block versus HardCopy IV HCell 18 × 18-Bit Independent Multiplier Implementation



DSP Operational Mode and Feature Support

HardCopy IV devices support all Stratix IV DSP configurations (9 × 9, 12 × 12, 18 × 18, and 36 × 36 multipliers) and all Stratix IV DSP block features, such as dynamic sign controls, dynamic addition/subtraction, dynamic rounding and saturation, and dynamic input shift registers.

HardCopy IV devices use DSP HCell macros to implement all five operational modes of the Stratix IV DSP block:

- Independent Multiplier (9 × 9, 12 × 12, 18 × 18, 36 × 36)
- Two-Multiplier Adder
- Four-Multiplier Adder
- Multiply Accumulate
- Shift

For more information about Stratix IV DSP blocks, refer to the *DSP Blocks in Stratix IV Devices* chapter in volume 1 of the *Stratix IV Device Handbook*.

Depending on the Stratix IV DSP configurations, the Quartus II software partitions the DSP function into a combination of DSP HCell macros for the HardCopy IV devices. This optimizes the DSP function and allows the core fabric to be used more efficiently.

Conclusion

HardCopy IV devices use HCells to implement the DSP block functions of Stratix IV devices. All the Stratix IV DSP operational modes are supported. Implementing DSP functions using HCells allows the HardCopy IV device core fabric to be used efficiently and offers significant static power savings compared with Stratix IV prototype devices.

Referenced Documents

This chapter references the following documents:

- *DSP Blocks in Stratix IV Devices* chapter in volume 1 of the *Stratix IV Device Handbook*
- *Logic Array Block and Adaptive Logic Module Implementation in HardCopy IV Devices* chapter in volume 1 of the *HardCopy IV Device Handbook*

Document Revision History

Table 3–1 shows the revision history for this document.

Table 3–1. Document Revision History

Date	Version	Changes Made
December 2008	1.0	Initial release.

4. TriMatrix Embedded Memory Blocks in HardCopy IV Devices

HIV51004-2.1

This chapter describes TriMatrix memory blocks, modes, features, and design considerations in HardCopy IV devices.

HardCopy® IV devices offer TriMatrix embedded memory blocks to efficiently address the needs of ASIC designs. TriMatrix memory comes in three different sizes and includes 640-bit memory logic array blocks (MLABs), 9-Kbit M9K blocks, and 144-Kbit M144K blocks. The MLABs have been optimized to implement filter delay lines, small first-in first-out (FIFO) buffers, and shift registers. You can use the M9K blocks for general purpose memory applications; you can use the M144K blocks for processor code storage, packet buffering, and video frame buffering.

TriMatrix memory in HardCopy IV devices support the same memory functions and features as Stratix® IV devices. You can independently configure each embedded memory block to be a single- or dual-port RAM, FIFO, ROM, or shift register using the MegaWizard™ Plug-in Manager in the Quartus® II software. You can stitch together multiple blocks of the same type to produce larger memories with minimal timing penalty. TriMatrix memory provides up to 20,736 Kbits of dedicated embedded static random access memory (SRAM). Memory Resources and Features

This chapter contains the following sections:

Memory Resources and Features

HardCopy IV embedded memory consists of MLAB, M9K, and M144K memory blocks and has a one-to-one mapping from Stratix IV memory. However, the number of available memory blocks differs based on density, package, and the Stratix IV device-to-HardCopy IV ASIC mapping paths, as shown in Table 4–1.

Table 4–1. HardCopy IV Embedded Memory Resources (Part 1 of 2) *(Note 1), (2)*

HardCopy IV ASIC	Stratix IV FPGA Prototype	M9K Blocks	M144K Blocks	Total Dedicated RAM Bits (not including MLABs)
HC4GX15	EP4SGX70	462	16	6,462 Kb
	EP4SGX110	660	16	8,244 Kb
	EP4SGX180	660	20	8,820 Kb
	EP4SGX230	660	22	9,108 Kb
	EP4SGX290	660	24	9,396 Kb
	EP4SGX360	660	24	9,396 Kb

Table 4–1. HardCopy IV Embedded Memory Resources (Part 2 of 2) *(Note 1)*, *(2)*

HardCopy IV ASIC	Stratix IV FPGA Prototype	M9K Blocks	M144K Blocks	Total Dedicated RAM Bits (not including MLABs)
HC4GX25	EP4SGX110	660	16	8,244 Kb
	EP4SGX180	936	20	11,304 Kb
	EP4SGX230	936	22	11,592 Kb
	EP4SGX290	936	36	13,608 Kb
	EP4SGX360	936	36	13,608 Kb
	EP4SGX530	936	36	13,608 Kb
HC4GX35	EP4SGX180	950	20	11,430 Kb
	EP4SGX230	1,235	22	14,283 Kb
	EP4SGX290	936	36	13,608 Kb
	EP4SGX360	1,248	48	18,144 Kb
	EP4SGX530	1,280	64	20,736 Kb
HC4E25	EP4SE230	864	22	10,944 Kb
	EP4SE360	864	32	12,384 Kb
HC4E35	EP4SE360	1,248	48	18,144 Kb
	EP4SE530	1,280	48	18,432 Kb
	EP4SE820	1,320	48	18,792 Kb

Notes to Table 4–1:

(1) In addition to device resource usage, Stratix IV device packages also determine the optimal HardCopy IV device mapping path. For example, the EP4SE360 device comes in H780 and F1152 packages. The mapping paths for the H780 and F1152 packages are the HC4E25 and HC4E35 devices, respectively.

(2) HardCopy IV devices do not have dedicated MLAB blocks but can support the same Stratix IV MLAB functionality. The number of MLABs that are supported in HardCopy IV devices varies depending on resource usage and the Stratix IV device-to-HardCopy IV device mapping path.

With regards to functionality, memory in HardCopy IV devices and Stratix IV devices is identical. The memory blocks can implement various types of memory with or without parity, including true dual-port, simple dual-port, and single-port RAM, ROM, and FIFO. Table 4–2 lists the size and features of the different memory blocks. In addition, unused memory blocks in HardCopy IV devices are powered down, allowing the HardCopy IV devices to have significant power savings.

Table 4–2. HardCopy IV Embedded Memory Features (Part 1 of 2)

Feature	MLABs	M9K Blocks	M144K Blocks
Maximum performance	TBD	TBD	TBD
Total RAM bits (including parity bits)	640	9,216	147,456

Table 4–2. HardCopy IV Embedded Memory Features (Part 2 of 2)

Feature	MLABs	M9K Blocks	M144K Blocks
Configurations (depth × width)	64 × 8 64 × 9 64 × 10 32 × 16 32 × 18 32 × 20	8K × 1 4K × 2 2K × 4 1K × 8 1K × 9 512 × 16 512 × 18 256 × 32 256 × 36	16K × 8 16K × 9 8K × 16 8K × 18 4K × 32 4K × 36 2K × 64 2K × 72
Parity bits	✓	✓	✓
Byte enable	✓	✓	✓
Packed mode	—	✓	✓
Address clock enable	✓	✓	✓
Single-port memory	✓	✓	✓
Simple dual-port memory	✓	✓	✓
True dual-port memory	—	✓	✓
Embedded shift register	✓	✓	✓
ROM	✓	✓	✓
FIFO buffer	✓	✓	✓
Simple dual-port mixed width support	—	✓	✓
True dual-port mixed width support	—	✓	✓
Memory initialization file (**.mif**)	Not supported, except in ROM mode	Not supported, except in ROM mode	Not supported, except in ROM mode
Mixed-clock mode	✓	✓	✓
Power-up condition	Outputs cleared if registered, otherwise reads memory contents *(1)*	Outputs cleared	Outputs cleared
Register clears	Outputs cleared	Outputs cleared	Outputs cleared
Write and Read operation triggering	Write: Falling clock edges Read: Rising clock edges	Write and Read: Rising clock edges	Write and Read: Rising clock edges
Same-port read-during-write	Outputs set to old data or don't care	Outputs set to old or new data	Outputs set to old or new data
Mixed-port read-during-write	Outputs set to old data or don't care	Outputs set to old data	Outputs set to old data
ECC Support	Soft IP support using the Quartus II software	Soft IP support using the Quartus II software	Built-in support in ×64-wide SDP mode or soft IP support using the Quartus II software

Note to Table 4–2:

(1) The memory contents for the MLAB in RAM mode are initialized to zero on power-up.

For more information about embedded memory support in Stratix IV devices, refer to the *TriMatrix Embedded Memory Blocks in Stratix IV Devices* chapter in volume 1 of the *Stratix IV Device Handbook*.

MLAB Implementation

While the M9K and M144K memory blocks are dedicated resources that function the same in Stratix IV and HardCopy IV devices, the MLABs are implemented differently in the two device families. In Stratix IV devices, the MLABs are dedicated blocks that you can configure for regular logic functions or memory functions. In HardCopy IV devices, the MLAB memory blocks are implemented using HCells. HCells are a collection of logic transistors connected together to form HCell macros (HCMs). The Quartus II software maps the Stratix IV MLAB function to the appropriate memory HCell macro that preserves the memory function. This allows the HardCopy IV core fabric to be used more efficiently, freeing up unused HCells for adaptive logic module (ALM) or digital signal processing (DSP) functions.

For more information about HCells in HardCopy IV devices, refer to the *Logic Array Block and Adaptive Logic Module Implementation in HardCopy IV Devices* chapter.

Design Considerations

Unlike Stratix IV devices, HardCopy IV devices do not have device configuration, so memories that are configured as RAM power up with random content. Therefore, the memory block contents cannot be pre-loaded or initialized with a memory initialization file (**.mif**) in HardCopy IV devices. You must ensure that your Stratix IV design does not require **.mifs** if you use the memory blocks as RAM. However, if you use the memory blocks as ROM, they are mask programmed to the design's ROM contents.

You can use the ALTMEM_INIT megafunction to initialize the RAM after power up for HardCopy IV devices. This megafunction reads from an internal ROM (inside the megafunction) or an external ROM (on chip or off chip) and writes to the RAM after power up.

When using non-registered output mode for the HardCopy IV MLAB memory blocks, the outputs power up with memory content. When using registered output mode for these memory blocks, the outputs are cleared on power up. You must take this into consideration when designing logic that might evaluate the initial power up values of the MLAB memory block.

Document Revision History

Table 4–3 lists the revision history for this chapter.

Table 4–3. Document Revision History

Date	Version	Changes Made
January 2010	2.1	■ Updated Table 4–1. ■ Minor text edits.
June 2009	2.0	■ Updated Table 4–1. ■ Minor text edits. ■ Removed the Conclusion and Referenced Documents sections.
December 2008	1.0	Initial release.

5. Clock Networks and PLLs in HardCopy IV Devices

HIV51005-2.1

This chapter provides a general description of clock networks and phase-locked loops (PLLs) in HardCopy® IV devices.

HardCopy IV devices support a hierarchical clock structure and multiple PLLs with advanced features equivalent to Stratix® IV devices. The large number of clocking resources in combination with clock synthesis precision offered by the PLLs, provides a complete clock management solution for your designs. HardCopy IV devices provide dedicated global clock networks (GCLKs), regional clock networks (RCLKs), and periphery clock networks (PCLKs). These clocks are organized into a hierarchical clock structure that provides up to 192 unique clock domains for the entire device and up to 60 unique clock sources per device quadrant. Altera's Quartus® II software compiler automatically turns off clock networks not used in the design, thereby reducing overall power consumption of the device.

HardCopy IV devices deliver abundant PLL resources with up to 12 PLLs per device and up to 10 outputs per PLL. These PLLs are feature rich, supporting advanced capabilities such as clock switchover, dynamic phase shifting, PLL reconfiguration, and reconfigurable bandwidth. HardCopy IV PLLs also support external feedback mode, spread-spectrum tracking, and post-scale counter cascading features. The Quartus II software enables the PLLs and their features without requiring any external devices.

All Stratix IV PLL features are supported by HardCopy IV PLLs.

For more information about clock networks and PLLs, refer to the *Clock Networks and PLLs in Stratix IV Devices* chapter in volume 1 of the *Stratix IV Device Handbook*.

This chapter contains the following sections:

Clock Networks in HardCopy IV Devices

HardCopy IV devices offer the same clock network resources and features as Stratix IV devices. Clock resources that are used in Stratix IV devices are mapped to equivalent clock resources in HardCopy IV devices, preserving the clocking functions. Unused clock resources are powered down to reduce power consumption.

Clock Network Resources

Similar to Stratix IV devices, HardCopy IV devices have up to 32 dedicated single-ended clock pins or 16 dedicated differential clock pins (`CLK[0:15]p` and `CLK[0:15]n`) that can drive either the GCLK or RCLK networks. These clock pins are arranged in the middle of the four sides of the HardCopy IV device.

You can drive the 16 GCLKs in HardCopy IV devices throughout the entire device, serving as low-skew clock sources for the core fabric and PLLs. You can also drive the GCLKs from the device I/O elements (IOEs) and internal logic to generate global clocks and other high fan-out control signals.

The RCLKs provide the lowest clock delay and skew for logic contained within a single device quadrant. You can drive RCLKs from IOEs and internal logic within a given quadrant.

The PCLKs are a collection of individual clock networks driven from the periphery of the HardCopy IV device. Clock outputs from the dynamic phase alignment (DPA) block, horizontal I/O pins, and internal logic can drive the PCLK networks. These PCLKs have higher skew when compared with the GCLK and RCLK networks and can be used instead of general purpose routing to drive signals into and out of the HardCopy IV device.

The GCLKs, RCLKs, and PCLKs available in HardCopy IV devices are organized into hierarchical clock structures that provide up to 192 unique clock domains (16 GCLK + 88 RCLK + 88 PCLK) across the entire device. HardCopy IV devices also allow up to 60 unique GCLK, RCLK, and PCLK clock sources (16 GCLK + 22 RCLK + 22 PCLK) per device quadrant.

Table 5–1 lists the clock resources available in HardCopy IV devices.

Table 5–1. Clock Resources in HardCopy IV Devices

Clock Resource	Number of Resources Available	Source of Clock Resource
Clock input pins	32 Single-ended (16 Differential)	`CLK[0..15]p` and `CLK[0..15]n` pins
Global clock networks	16	`CLK[0..15]p/n` pins, PLL clock outputs, and logic array
Regional clock networks	88	`CLK[0..15]p/n` pins, PLL clock outputs, and logic array
Peripheral clock networks	88 (22 per device quadrant) *(1)*	DPA clock outputs, horizontal I/O pins, and logic array
GCLKs/RCLKs per quadrant	38	16 GCLKs + 22 RCLKs
GCLKs/RCLKs per device	104	16 GCLKs + 88 RCLKs

Note to Table 5–1:

(1) There are 56 PCLKs in HC4E25 and HC4GX25 devices and 88 PCLKs in HC4E35 and HC4GX35 devices. The HC4GX15 devices have 28 PCLKs.

Clocking Regions

HardCopy IV devices can implement the four different types of Stratix IV clocking regions using GCLK and RCLK networks. These types of clocking regions include the following:

- Entire device clock region
- Regional clock region
- Dual-regional clock region
- Sub-regional clock region

Clock Control Block

HardCopy IV devices also support the same features as the Stratix IV clock control block, which is available for each GCKL and RCLK network. The clock control block provides the following features:

- Clock source selection (dynamic selection for GCLKs)

  You can statically or dynamically select the GCLK source. The RCLK source can only be statically selected. Static selection involves mask programming the clock multiplexer select inputs. The clock selection is fixed and cannot be changed when the HardCopy IV device is in user mode. Dynamic selection for the GCLK source uses internal logic to control the clock multiplexer select inputs when the device is in user mode. For dynamic clock source selection, you can either select two PLL outputs (such as `CLK0` or `CLK1`) or a combination of clock pins or PLL outputs.

- Clock power-down (static or dynamic clock enable or disable)

  You can statically or dynamically power-down the GCLK and RCLK networks, reducing overall power consumption of the device. Unused GCLK and RCLK networks are powered down through static settings that are automatically generated by the Quartus II software and that are mask programmed into the device. The dynamic clock enable or disable feature allows internal logic to synchronously control power-up or power-down on GCLK and RCLK networks, including dual-regional clock regions.

PLLs in HardCopy IV Devices

HardCopy IV devices offer up to 12 PLLs that support the same features as the Stratix IV PLLs. These PLLs provide robust clock management and synthesis for device clock management, external system clock management, and high-speed I/O interfaces. The nomenclature for the PLLs follows their geographical location in the device floorplan. The PLLs that reside on the top and bottom sides of the device are named PLL_T1, PLL_T2, PLL_B1, and PLL_B2

The PLLs that reside on the left and right sides of the device are named PLL_L1, PLL_L2, PLL_L3, PLL_L4, PLL_R1, PLL_R2, PLL_R3, and PLL_R4, respectively.

Table 5–2 and Table 5–3 list the number of PLLs available in the HardCopy IV device family.

Table 5–2. HardCopy IV E Device PLL Availability (Part 1 of 2)

HardCopy IV E Device	Stratix IV Prototype Device	L1	L2	L3	L4	T1	T2	B1	B2	R1	R2	R3	R4
HC4E25WF484N *(1)*	EP4SE230F29 (F780)	—	✓	—	—	✓	—	✓	—	—	✓	—	—
HC4E25FF484N *(1)*	EP4SE230F29 (F780)	—	✓	—	—	✓	—	✓	—	—	✓	—	—
HC4E25WF780N	EP4SE230F29 (F780)	—	✓	—	—	✓	—	✓	—	—	✓	—	—
	EP4SE360H29 (H780)	—	✓	—	—	✓	—	✓	—	—	✓	—	—
HC4E25FF780N	EP4SE230F29 (F780)	—	✓	—	—	✓	—	✓	—	—	✓	—	—
	EP4SE360H29 (H780)	—	✓	—	—	✓	—	✓	—	—	✓	—	—

Table 5–2. HardCopy IV E Device PLL Availability (Part 2 of 2)

HardCopy IV E Device	Stratix IV Prototype Device	L1	L2	L3	L4	T1	T2	B1	B2	R1	R2	R3	R4
HC4E35LF1152N	EP4SE360F35 (F1152)	—	✓	✓	—	✓	✓	✓	✓	—	✓	—	—
	EP4SE530H35 (H1152)	—	✓	✓	—	✓	✓	✓	✓	—	✓	—	—
	EP4SE820H35 (H1152)	—	✓	✓	—	✓	✓	✓	✓	—	✓	—	—
HC4E35FF1152N	EP4SE360F35 (F1152)	—	✓	✓	—	✓	✓	✓	✓	—	✓	✓	—
	EP4SE530H35 (H1152)	—	✓	✓	—	✓	✓	✓	✓	—	✓	✓	—
	EP4SE820H35 (H1152)	—	✓	✓	—	✓	✓	✓	✓	—	✓	✓	—
HC4E35LF1517N	EP4SE360F40 (F1517)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	EP4SE530H40 (H1517)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	EP4SE820H40 (H1517)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
HC4E35FF1517N	EP4SE360F40 (F1517)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	EP4SE530H40 (H1517)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	EP4SE820H40 (H1517)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Note to Table 5–2:

(1) You are migrating from 780 package in the FPGA to a 484 package in the HardCopy device. Board change is required for non-socket migration.

Table 5–3. HardCopy IV GX Device PLL Availability (Part 1 of 2)

HardCopy IV GX Device	Stratix IV Prototype Device	L1	L2	L3	L4	T1	T2	B1	B2	R1	R2	R3	R4
HC4GX15LF780N	EP4SGX70DF29 (F780)	—	✓	—	—	✓	—	✓	—	—	—	—	—
	EP4SGX110DF29 (F780)	—	✓	—	—	✓	—	✓	—	—	—	—	—
	EP4SGX180DF29 (F780)	—	✓	—	—	✓	—	✓	—	—	—	—	—
	EP4SGX230DF29 (F780)	—	✓	—	—	✓	—	✓	—	—	—	—	—
HC4GX15LA780N	EP4SGX290FH29 (H780)	—	—	—	—	✓	—	✓	—	—	—	—	—
	EP4SGX360FH29 (H780)	—	—	—	—	✓	—	✓	—	—	—	—	—
HC4GX25LF780N	EP4SGX290FH29 (H780)	—	—	—	—	✓	—	✓	—	—	—	—	—
	EP4SGX360FH29 (H780)	—	—	—	—	✓	—	✓	—	—	—	—	—
HC4GX25LF1152N	EP4SGX110FF35 (F1152)	—	✓	—	—	✓	—	✓	—	—	✓	—	—
	EP4SGX180FF35 (F1152)	—	✓	—	—	✓	✓	✓	✓	—	✓	—	—
	EP4SGX230FF35 (F1152)	—	✓	—	—	✓	✓	✓	✓	—	✓	—	—
	EP4SGX290FF35 (F1152)	—	✓	—	—	✓	✓	✓	✓	—	✓	—	—
	EP4SGX360FF35 (F1152)	—	✓	—	—	✓	✓	✓	✓	—	✓	—	—
HC4GX25FF1152N	EP4SGX180HF35 (F1152)	—	✓	—	—	✓	✓	✓	✓	—	✓	—	—
	EP4SGX230HF35 (F1152)	—	✓	—	—	✓	✓	✓	✓	—	✓	—	—
	EP4SGX290HF35 (F1152)	—	✓	—	—	✓	✓	✓	✓	—	✓	—	—
	EP4SGX360HF35 (F1152)	—	✓	—	—	✓	✓	✓	✓	—	✓	—	—
	EP4SGX530HH35 (H1152)	—	✓	—	—	✓	✓	✓	✓	—	✓	—	—
HC4GX35FF1152N	EP4SGX230HF35 (F1152)	—	✓	—	—	✓	✓	✓	✓	—	✓	—	—
	EP4SGX360HF35 (F1152)	—	✓	—	—	✓	✓	✓	✓	—	✓	—	—
	EP4SGX530HH35 (H1152)	—	✓	—	—	✓	✓	✓	✓	—	✓	—	—

Table 5–3. HardCopy IV GX Device PLL Availability (Part 2 of 2)

HardCopy IV GX Device	Stratix IV Prototype Device	L1	L2	L3	L4	T1	T2	B1	B2	R1	R2	R3	R4
HC4GX35LF1517N	EP4SGX230KF40 (F1517)	—	✓	—	—	✓	✓	✓	✓	—	✓	—	—
	EP4SGX360KF40 (F1517)	—	✓	—	—	✓	✓	✓	✓	—	✓	—	—
	EP4SGX530KH40 (H1517)	—	✓	—	—	✓	✓	✓	✓	—	✓	—	—
HC4GX35FF1517N	EP4SGX180KF40 (F1517)	—	✓	✓	—	✓	✓	✓	✓	—	✓	✓	—
	EP4SGX230KF40 (F1517)	—	✓	✓	—	✓	✓	✓	✓	—	✓	✓	—
	EP4SGX290KF40 (F1517)	—	✓	✓	—	✓	✓	✓	✓	—	✓	✓	—
	EP4SGX360KF40 (F1517)	—	✓	✓	—	✓	✓	✓	✓	—	✓	✓	—
	EP4SGX530KH40 (H1517)	—	✓	✓	—	✓	✓	✓	✓	—	✓	✓	—

The PLL functionality in HardCopy IV devices remains the same in Stratix IV PLLs. Therefore, HardCopy IV PLLs also support features such as PLL reconfiguration, where you can dynamically configure the PLL in user mode.

All HardCopy IV PLLs have the same core analog structure with only minor differences in features that are supported. Table 5–4 lists the features of the top/bottom and left/right PLLs in HardCopy IV devices.

For more information about Stratix IV PLL features, refer to the *Clock Networks and PLLs in Stratix IV Devices* chapter in volume 1 of the *Stratix IV Device Handbook*.

Table 5–4. HardCopy IV PLL Features (Part 1 of 2)

Feature	HardCopy IV Top/Bottom PLLs	HardCopy IV Left/Right PLLs
C (output) counters	10	7
M, N, C counter sizes	1 to 512	1 to 512
Dedicated clock outputs	6 single-ended or 4 single-ended and 1 differential pair	2 single-ended or 1 differential pair
Clock input pins	8 single-ended or 4 differential pin pairs	8 single-ended or 4 differential pin pairs
External feedback input pin	Single-ended or differential	Single-ended only
Spread-spectrum input clock tracking	Yes *(1)*	Yes *(1)*
PLL cascading	Through GCLK and RCLK and dedicated path between adjacent PLLs	Through GCLK and RCLK and dedicated path between adjacent PLLs *(2)*
Compensation modes	All except LVDS clock network compensation	All except external feedback mode when using differential I/Os
PLL drives LVDSCLK and LOADEN	No	Yes
VCO output drives DPA clock	No	Yes
Phase shift resolution	Down to 96.125ps *(3)*	Down to 96.125ps *(3)*
Programmable duty cycle	Yes	Yes
Output counter cascading	Yes	Yes

Table 5–4. HardCopy IV PLL Features (Part 2 of 2)

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

Notes to Table 5–4:

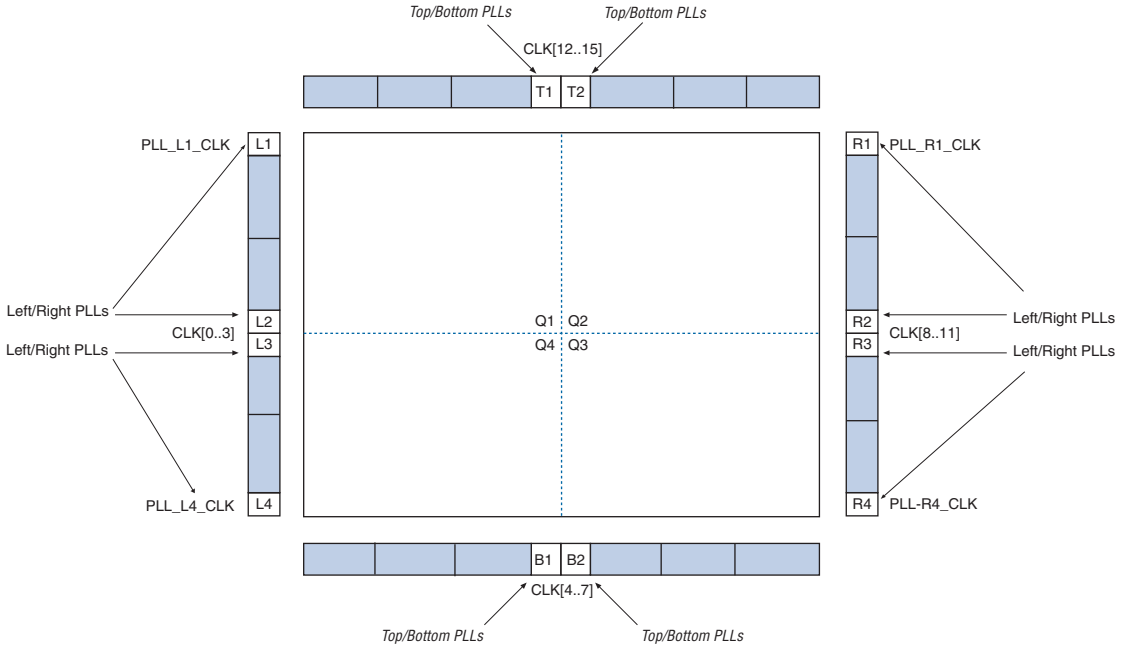
(1) Provided input clock jitter is within input jitter tolerance specifications.

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

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

Figure 5–1 shows the PLL locations in HardCopy IV devices. Some PLLs are not available depending on the density and package of the HardCopy IV device.

Figure 5–1. HardCopy IV PLL Locations



Design Considerations

To ensure that your Stratix IV design can be successfully mapped to the HardCopy IV design, follow these general guidelines when implementing your design. The following guidelines help make your design robust, ensuring it meets timing closure and achieves the performance you need:

- Match the PLL resources used in HardCopy IV devices and Stratix IV devices in order to successfully map your design from the FPGA design to the ASIC design, or vice-versa. This is necessary to ensure that all the resources used and the functions implemented in both designs match. Make sure to select the companion device during device selection in the Quartus II software. Doing this restricts the Quartus II software to resources that are common to both the FPGA and ASIC devices and ensures that the design can map successfully. Refer to Table 5–2 for the available PLLs in the HardCopy IV series devices for non-socket migration.
- Enable PLL reconfiguration for your design if it uses PLLs. The PLL settings in HardCopy IV devices may require different settings from the Stratix IV PLLs because of the different clock tree lengths and PLL compensations. By enabling PLL reconfiguration, you can adjust your PLL settings on the HardCopy IV device after the silicon has been fabricated. This allows you to fine tune and further optimize your system performance.
- Use dedicated clock input pins to drive the PLL reference clock inputs, particularly if your design is interfacing with an external memory. This minimizes reference clock input jitter to the PLLs, providing more margin for your design.

When you cascade PLLs for the ALTMEMPHY, ensure that:

- The input clock to the ALTMEMPHY PLL is fed by a dedicated input
- If the ALTMEMPHY PLL is fed by another PLL, the source PLL
  - input must be fed by a dedicated input pin
  - must be in no compensation mode
- If the input clock to the ALTMEMPHY is fed by another PLL, the ALTMEMPHY PLL's input clock must be from a dedicated clock output from the source PLL.

Document Revision History

Table 5–5 lists the revision history for this chapter.

Table 5–5. Document Revision History

Date	Version	Changes Made
January 2010	2.1	■ Updated Table 5–2. ■ Minor text edits.
June 2009	2.0	Added non-socket information and new part numbers.
December 2008	1.0	Initial release.

This section includes the following chapters:

Revision History

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

6. HardCopy IV Device I/O Features

HIV51006-2.1

This chapter documents I/O standards, features, termination schemes, and performance supported in HardCopy® IV devices. All HardCopy IV devices have configurable high-performance I/O drivers and receivers supporting a wide range of industry standard interfaces. Both the top/bottom (column) and left/right (row) I/O banks of HardCopy IV devices support the same I/O standards with different performance specifications.

This chapter includes the following sections:

Numerous I/O features assist in high-speed data transfer into and out of the HardCopy device.

HardCopy IV GX I/O Support:

- Up to 32 full-duplex clock data recovery (CDR)-based transceivers supporting data rates between 600 Mbps and 6.5 Gbps
- Dedicated circuitry to support physical layer functionality for popular serial protocols, such as PCI Express (PIPE) Gen1 and Gen2, Gigabit Ethernet, Serial RapidIO, SONET/SDH, XAUI/HiGig, (OIF) CEI-6G, SD/HD/3G-SDI, Fibre Channel, SFI-5, and Interlaken
- Complete PCI Express (PIPE) protocol solution with embedded PCI Express hard IP blocks that implement PHY-MAC layer, data-link layer, and transaction layer functionality

Supported I/O standards:

- Single-ended, non-voltage-referenced or voltage-referenced I/O standards; low-voltage differential signaling (LVDS); reduced swing differential signal (RSDS); mini-LVDS; high-speed transceiver logic (HSTL); and stub series terminated logic (SSTL)

- Single data rate (SDR) and half data rate (HDR, half frequency and twice the data width of SDR) input and output options
- Up to 88 full duplex 1.25 Gbps true LVDS channels (88Tx + 88Rx) on the row I/O banks

Features supported in a single-ended I/O interface:

- De-skew, read and write leveling, and clock-domain crossing functionality
- Multiple output current strength setting for different I/O standards
- Four slew rate settings
- Four output delay settings
- Six I/O delay settings
- Optional bus-hold
- Optional pull-up resistor
- Optional open-drain output
- Serial, parallel, and dynamic on-chip termination (OCT)

Features supported in a high-speed memory interface:

- Dedicated DQS logic in both column and row I/Os
- Each I/O bank is accessible by two delay-locked loops (DLLs) that have different frequencies and phase shift
- Low power option when the memory interface is not used

Features supported in a high-speed differential I/O interface:

- Four slew rate settings
- Differential OCT
- Hard dynamic phase alignment (DPA) block with serializer/deserializer (SERDES)
- Four pre-emphasis settings
- Four differential output voltage (V_{OD}) settings

Differences Between HardCopy IV ASICs and Stratix IV FPGAs

Both HardCopy IV and Stratix IV devices support the same speed, performance, I/O standards, and implementation guidelines. You must set the HardCopy IV companion device for your Stratix IV design project in the Quartus® II software. Otherwise, you may not be able to map to a HardCopy IV device, because of the varying amounts of resource availability. There are three major differences between HardCopy IV ASICs and Stratix IV FPGAs:

- There are eight calibration blocks in HardCopy IV devices instead of up to 10 calibration blocks in Stratix IV devices.
- Stratix IV devices support up to 24 I/O banks, while HardCopy IV devices support up to 20 I/O banks.

Stratix IV and HardCopy IV devices support different I/O counts per bank. Therefore, always set the HardCopy IV companion device for your Stratix IV design project in the Quartus II software. For more information, refer to Table 6–2.

Table 6–1 lists the differences between HardCopy IV GX and Stratix IV GX devices.

Table 6–1. Differences Between HardCopy IV GX and Stratix IV GX Devices

	Stratix IV GX	HardCopy IV GX
Max Data Rate	~8.5 Gbps	~6.5 Gbps
PCI Express (PIPE) Data Rate	Gen1 (2.5G) and Gen2 (5G)	Gen1 (2.5G) and Gen2 (5G) *(1)*
PMA ADCE	Yes	Yes
PMA Direct Mode	Yes	Yes
PMA 5th and 6th Channels	Yes	Yes
IOG Channel Support	Yes	No
6G LC Block	Yes	Yes
HIP Count	1 per 2 Quads (except orphan Quads)	1 per 2 Quads (except orphan Quads)
HSSI Location	Left and Right, outside regular I/O strip	Right Only—HA1GX, without regular I/O strip Left and Right—HA2GX/HA3GX, outside regular I/O strip
HIP Memory	3 MRAMs	MRAMs (with payload reduction)
HIP and PCS Powers	Dedicated	Dedicated
HIP and PCS Well-Biasing	Yes	No (shorted to VSS)
PCLK Multiplexer Location	In DPA (outside HSSI)	HA1GX—In the CORE fabric (outside HSSI) HA2GX/HA3GX—In DPA (outside HSSI)
Eye-Viewer Support	Yes	Yes
AC JTAG Support	No	No

Note to Table 6–1:

(1) Payload reduction for PCIe (PIPE) Gen2 x8 mode at 500 Mhz to 1 kB, error correction coding (ECC) not supported, restricted range on CSR and DPRIO settings, and MRAM size reduced to 8 kB for retry and receive buffers.

I/O Standards and Voltage Levels

HardCopy IV devices support a wide range of industry I/O standards, including single-ended, voltage-referenced single-ended, and differential I/O standards.

Table 6–2 lists the supported I/O standards and the typical values for input and output V_{CCIO}, V_{CCPD}, V_{REF}, and board V_{TT}.

Table 6–2. HardCopy IV I/O Standards and Voltage Levels (Part 1 of 2)

I/O Standard	Standard Support	V_{CCIO} (V) *(Note 1)*				V_{CCPD} (V) (Pre-Driver Voltage)	V_{REF} (V) (Input Ref Voltage)	V_{TT} (V) (Board Termination Voltage)
		Input Operation		Output Operation				
		Column I/O Banks	Row I/O Banks	Column I/O Banks	Row I/O Banks			
3.3-V LVTTL	JESD8-B	3.0/2.5	3.0/2.5	3.0	3.0	3.0	—	—
3.3-V LVCMOS	JESD8-B	3.0/2.5	3.0/2.5	3.0	3.0	3.0	—	—
2.5-V LVTTL/LVCMOS	JESD8-5	3.0/2.5	3.0/2.5	2.5	2.5	2.5	—	—
1.8-V LVTTL/LVCMOS	JESD8-7	1.8/1.5	1.8/1.5	1.8	1.8	2.5	—	—
1.5-V LVTTL/LVCMOS	JESD8-11	1.8/1.5	1.8/1.5	1.5	1.5	2.5	—	—
1.2-V LVTTL/LVCMOS	JESD8-12	1.2	1.2	1.2	1.2	2.5	—	—
3.0-V PCI	PCI Rev 2.1	3.0	3.0	3.0	3.0	3.0	—	—
3.0-V PCI-X	PCI-X Rev 1.0	3.0	3.0	3.0	3.0	3.0	—	—
SSTL-2 Class I	JESD8-9B	*(2)*	*(2)*	2.5	2.5	2.5	1.25	1.25
SSTL-2 Class II	JESD8-9B	*(2)*	*(2)*	2.5	2.5	2.5	1.25	1.25
SSTL-18 Class I	JESD8-15	*(2)*	*(2)*	1.8	1.8	2.5	0.90	0.90
SSTL-18 Class II	JESD8-15	*(2)*	*(2)*	1.8	1.8	2.5	0.90	0.90
SSTL-15 Class I	—	*(2)*	*(2)*	1.5	1.5	2.5	0.75	0.75
SSTL-15 Class II	—	*(2)*	*(2)*	1.5	—	2.5	0.75	0.75
HSTL-18 Class I	JESD8-6	*(2)*	*(2)*	1.8	1.8	2.5	0.90	0.90
HSTL-18 Class II	JESD8-6	*(2)*	*(2)*	1.8	1.8	2.5	0.90	0.90
HSTL-15 Class I	JESD8-6	*(2)*	*(2)*	1.5	1.5	2.5	0.75	0.75
HSTL-15 Class II	JESD8-6	*(2)*	*(2)*	1.5	—	2.5	0.75	0.75
HSTL-12 Class I	JESD8-16A	*(2)*	*(2)*	1.2	1.2	2.5	0.6	0.6
HSTL-12 Class II	JESD8-16A	*(2)*	*(2)*	1.2	—	2.5	0.6	0.6
Differential SSTL-2 Class I	JESD8-9B	*(2)*	*(2)*	2.5	2.5	2.5	—	1.25
Differential SSTL-2 Class II	JESD8-9B	*(2)*	*(2)*	2.5	2.5	2.5	—	1.25
Differential SSTL-18 Class I	JESD8-15	*(2)*	*(2)*	1.8	1.8	2.5	—	0.90
Differential SSTL-18 Class II	JESD8-15	*(2)*	*(2)*	1.8	1.8	2.5	—	0.90
Differential SSTL-15 Class I	—	*(2)*	*(2)*	1.5	1.5	2.5	—	0.75
Differential SSTL-15 Class II	—	*(2)*	*(2)*	1.5	—	2.5	—	0.75
Differential HSTL-18 Class I	JESD8-6	*(2)*	*(2)*	1.8	1.8	2.5	—	0.90
Differential HSTL-18 Class II	JESD8-6	*(2)*	*(2)*	1.8	1.8	2.5	—	0.90
Differential HSTL-15 Class I	JESD8-6	*(2)*	*(2)*	1.5	1.5	2.5	—	0.75

Table 6–2. HardCopy IV I/O Standards and Voltage Levels (Part 2 of 2)

I/O Standard	Standard Support	V_{CCIO} (V) *(Note 1)*				V_{CCPD} (V) (Pre-Driver Voltage)	V_{REF} (V) (Input Ref Voltage)	V_{TT} (V) (Board Termination Voltage)
		Input Operation		Output Operation				
		Column I/O Banks	Row I/O Banks	Column I/O Banks	Row I/O Banks			
Differential HSTL-15 Class II	JESD8-6	*(2)*	*(2)*	1.5	—	2.5	—	0.75
Differential HSTL-12 Class I	JESD8-16A	*(2)*	*(2)*	1.2	1.2	2.5	—	0.60
Differential HSTL-12 Class II	JESD8-16A	*(2)*	*(2)*	1.2	—	2.5	—	0.60
LVDS *(3)*, *(4)*	ANSI/TIA/EIA-644	*(2)*	*(2)*	2.5	2.5	2.5	—	—
RSDS *(5)*, *(6)*	—	*(2)*	*(2)*	2.5	2.5	2.5	—	—
mini-LVDS *(5)*, *(6)*	—	*(2)*	*(2)*	2.5	2.5	2.5	—	—
LVPECL	—	*(3)*	2.5	—	—	2.5	—	—

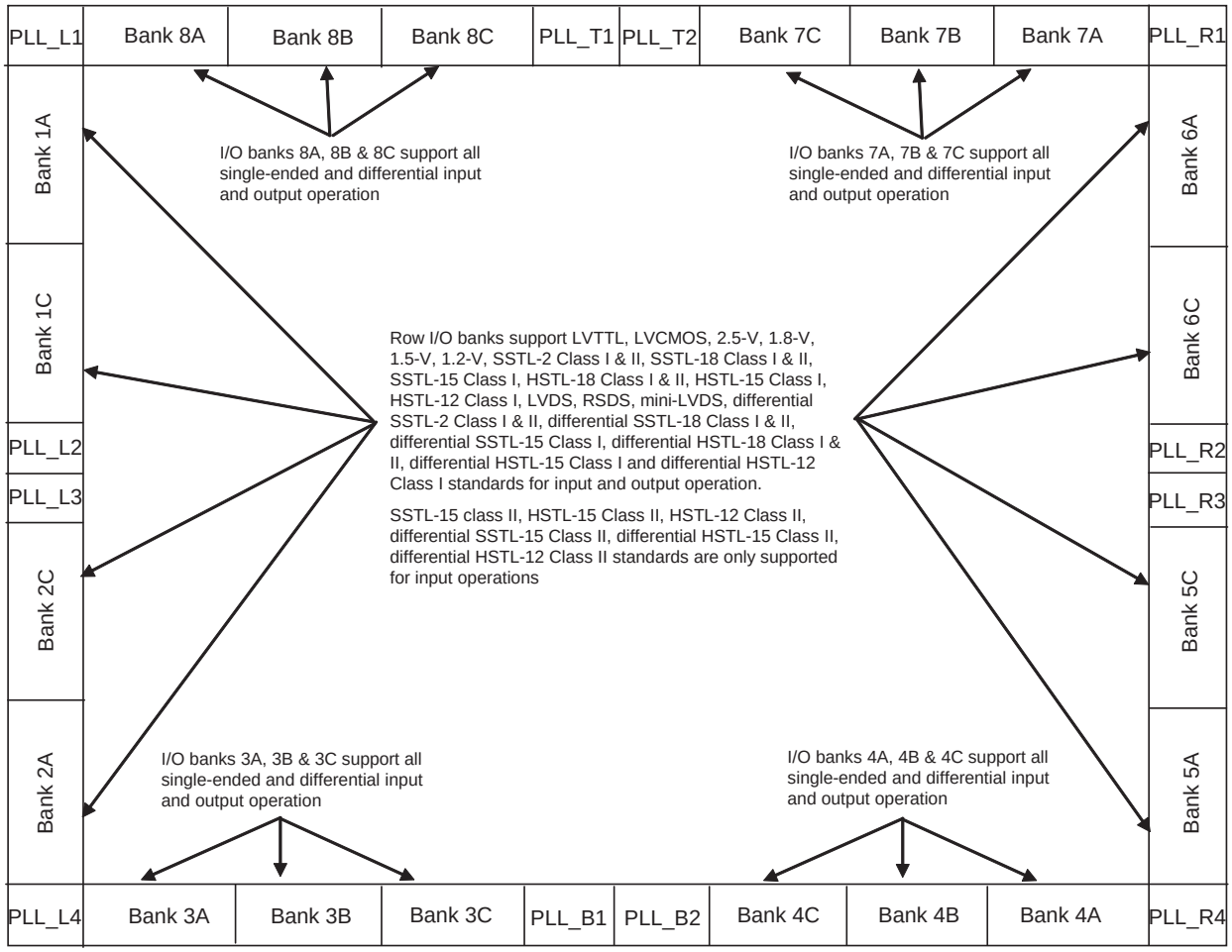
Notes to Table 6–2:

(1) V_{CCPD} is either 2.5 or 3.0 V. For V_{CCIO} = 3.0 V, V_{CCPD} = 3.0 V. For V_{CCIO} = 2.5 V or less, V_{CCPD} = 2.5 V.
(2) Single-ended HSTL/SSTL, differential SSTL/HSTL, and LVDS input buffers are powered by V_{CCPD}. Row I/O banks support both true differential input buffers and true differential output buffers. Column I/O banks support true differential input buffers, but not true differential output buffers. I/O pins are organized in pairs to support differential standards. Column I/O differential HSTL and SSTL inputs use LVDS differential input buffers without on-chip R_D support.
(3) Column I/O banks support LVPECL I/O standards for input clock operation. Clock inputs on column I/O are powered by $V_{CCCLKIN}$ when configured as differential clock input. They are powered by V_{CCIO} when configured as single-ended clock input. Differential clock inputs in row I/O are powered by V_{CCPD}.
(4) Column and row I/O banks support LVDS outputs using two single-ended output buffers, an external one-resistor (LVDS_E_1R), and a three-resistor (LVDS_E_3R) network.
(5) Row I/O banks support RSDS and mini-LVDS I/O standards using a dedicated LVDS output buffer without a resistor network.
(6) Column and row I/O banks support RSDS and mini-LVDS I/O standards using two single-ended output buffers with one-resistor (RSDS_E_1R and mini-LVDS_E_1R) and three-resistor (RSDS_E_3R and mini-LVDS_E_3R) networks.

HardCopy IV I/O

HardCopy IV devices contain up to 20 I/O banks, as shown in Figure 6–1. For the 1152- and 1517-pin packages there are 20 available I/O banks; for the 780-pin package there are 16 available I/O banks. Row I/O banks contain true differential input and output buffers and banks with dedicated circuitry to support differential standards at speeds up to 1.25 Gbps.

Figure 6–1. HardCopy IV I/O Banks *(Note 1), (2), (3), (4), (5), (6), (7), (8), (9), (10)*



Notes to Figure 6–1:

(1) There are 12 I/O banks for the 484-pin package, 16 I/O banks for the 780-pin package, and 20 I/O banks for 1152- and 1517-pin packages.

(2) Differential HSTL and SSTL outputs are not true differential outputs. They use two single-ended outputs with the second output programmed as inverted.

(3) Column I/O differential HSTL and SSTL inputs use LVDS differential input buffers without differential OCT support.

(4) Column I/O supports LVDS outputs using single-ended buffers and external resistor networks.

(5) Column I/O supports PCI/PCI-X with an on-chip clamping diode. Row I/O supports PCI/PCI-X with an external clamping diode.

(6) Differential clock inputs on column I/O use $V_{CCCLKIN}$. All outputs use the corresponding bank V_{CCIO}.

(7) Row I/O supports the dedicated LVDS output buffer.

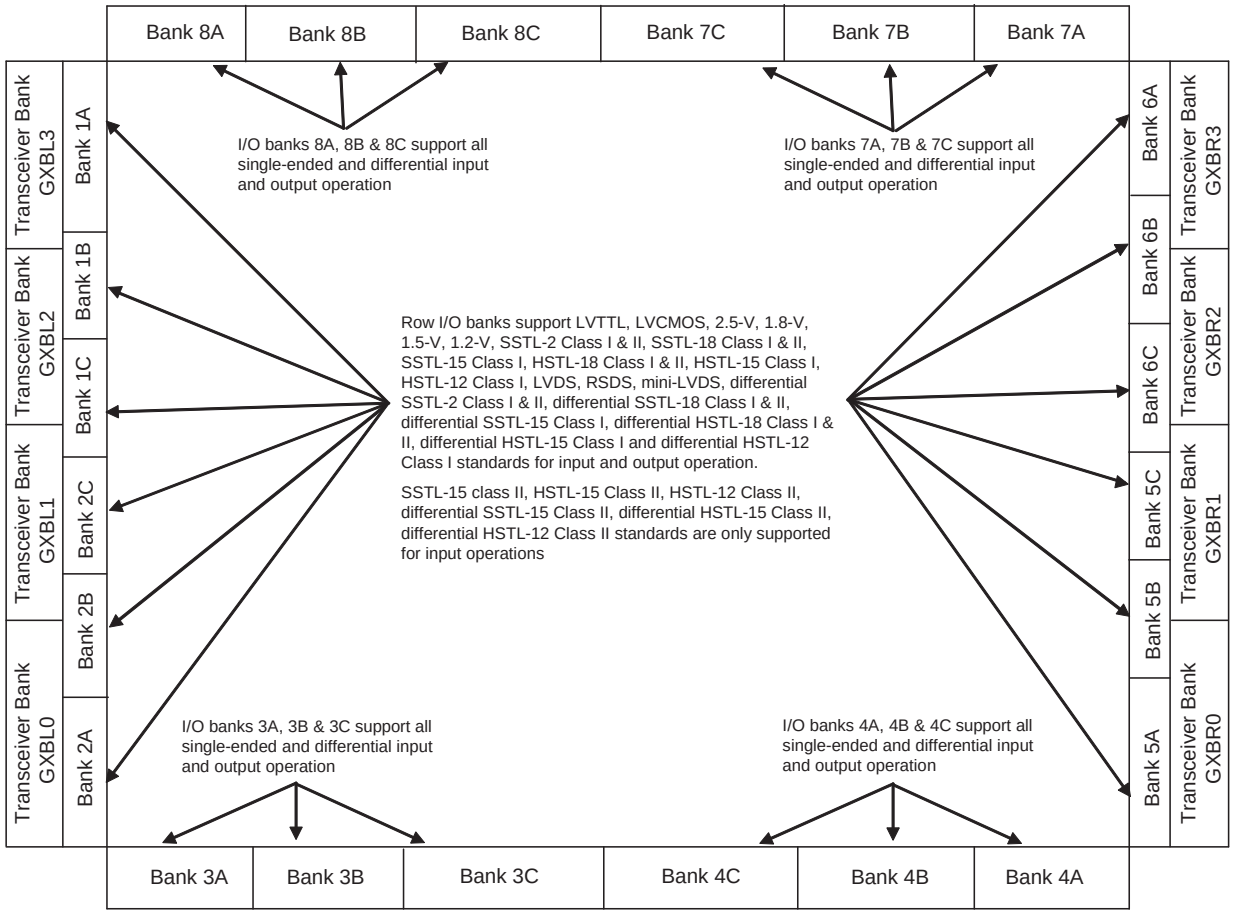
(8) Column I/O banks support LVPECL-only standards for input clock operation.

(9) Single-ended inputs and outputs are not allowed when true differential I/O (DPA and non-DPA) exist in an I/O bank.

(10) Figure 6–1 is a top view of the silicon die that corresponds to a reverse view for flip chip packages. It is a graphical representation only.

Figure 6–2 shows HardCopy IV GX devices I/O bank.

Figure 6–2. HardCopy IV GX Devices I/O Bank *(Note 1), (2), (3), (4), (5), (6), (7), (8), (9), (10), (11), (12)*



Notes to Figure 6–2:

(1) HC4GX15 devices do not have I/O Banks 5A, 5B, 5C, 6A, 6B, and 6C and have only two HSSI Quads on the right (GXBR1 and GXBR2).

(2) HC4GX25 devices have two HSSI Quads on the right and left (GXBL1, GXBL2, GXBR1, and GXBR2).

(3) HC4GX35 devices have three HSSI Quads on the right and left (GXBL0, GXBL1, GXBL2, GXBR0, GXBR1, and GXBR2).

(4) Differential HSTL and SSTL outputs are not true differential outputs. They use two single-ended outputs with the second output programmed as inverted.

(5) Column I/O differential HSTL and SSTL inputs use LVDS differential input buffers without differential OCT support.

(6) Column I/O supports LVDS outputs using single-ended buffers and external resistor networks.

(7) Column I/O supports PCI/PCI-X with on-chip clamp diode. Row I/O supports PCI/PCI-X with external clamp diode.

(8) Clock inputs on column I/O are powered by $V_{CCCLKIN}$ when configured as differential clock input. They are powered by V_{CCIO} when configured as single-ended clock inputs. All outputs use the corresponding bank V_{CCIO}.

(9) Row I/O banks support the dedicated LVDS output buffer.

(10) Column and row I/O banks support LVPECL standards for input clock operation.

(11) Single-ended inputs and outputs are not allowed when true differential I/O (DPA and non-DPA) exist in an I/O bank.

(12) Figure 6–2 is a top view of the silicon die that corresponds to a reverse view for flip chip packages. It is a graphical representation only.

Every I/O bank in HardCopy IV devices can support high-performance external memory interfaces with dedicated circuitry. The I/O pins are organized in pairs to support differential standards. Each I/O pin pair can support both differential input and output buffers. The only exceptions are the `clk1`, `clk3`, `clk8`, `clk10`, `PLL_L1_clk`, `PLL_L4_clk`, `PLL_R1_clk`, and `PLL_R4_clk` pins which support differential input operations only.

For more information about the number of channels available for the LVDS I/O standard, refer to the *High-Speed Differential I/O Interface with DPA in HardCopy IV Devices* chapter.

HardCopy IV I/O Banks

The I/O pins in HardCopy IV devices are arranged in groups called modular I/O banks. Depending on the device package, the number of I/O banks varies. The size of each bank can also vary. Table 6–3 lists the I/O count per bank for all available pin packages.

Table 6–3. HardCopy IV E I/O Count per Bank *(Note 1)* (Part 1 of 2)

Bank	Device Package Pin Count			
	484	780	1152	1517
1A	24	32	48	50
1C	24	26	42	42
2A	24	32	48	50
2C	24	26	42	42
3A	—	40	40	48
3B	—	—	24	48
3C	24	24	32	32
4A	—	40	40	48
4B	—	—	24	48
4C	24	24	32	32
5A	24	32	48	50
5C	24	26	42	42
6A	24	32	48	50
6C	24	26	42	42
7A	—	40	40	48
7B	—	—	24	48
7C	24	24	32	32
8A	—	40	40	48
8B	—	—	24	48
8C	24	24	32	32
Total Bank	**12**	**16**	**20**	**20**

Table 6–3. HardCopy IV E I/O Count per Bank *(Note 1)* (Part 2 of 2)

Bank	Device Package Pin Count			
	484	780	1152	1517
Total I/O	296	488	744	880 *(2)*

Notes to Table 6–3:

(1) These numbers include dedicated clock pins and regular I/Os.

(2) The HardCopy IV F1517-pin package supports less I/O count than the Stratix IV F1517-pin package. Therefore, always set the HardCopy companion devices in your Quartus II project to ensure proper mapping.

Table 6–4 lists the HardCopy IV GX I/O count per bank.

Table 6–4. HardCopy IV GX I/O Count per Bank *(Note 1)*

Bank	Device Package Count				
	LF780	LAF780	LF1152	FF1152	FF1517
1A	—	32	48	48	48
1C	—	24	40	40	40
2A	—	32	—	—	48
2C	—	24	—	—	40
3A	40	40	40	40	40
3B	—	—	24	24	24
3C	24	24	32	32	32
4A	40	40	40	40	40
4B	—	—	24	24	24
4C	24	24	32	32	32
5A	—	—	—	—	48
5C	—	—	—	—	40
6A	—	—	48	48	48
6C	—	—	40	40	40
7A	40	40	40	40	40
7B	—	—	24	24	24
7C	24	24	32	32	32
8A	40	40	40	40	40
8B	—	—	24	24	24
8C	24	24	32	32	32
Total Bank	**8**	**12**	**16**	**16**	**20**
Total I/O	**256**	**368**	**560**	**560**	**736**

Note to Table 6–4:

(1) These do not include the dedicated clock pins.

HardCopy IV I/O Structure

The I/O element (IOE) in HardCopy IV devices contains a bidirectional I/O buffer and I/O registers to support a complete embedded bidirectional single data rate or DDR transfer. Figure 6–1 shows that certain I/O banks support certain I/O standards. The IOEs are located in I/O blocks around the periphery of the HardCopy IV device.

The HardCopy IV bidirectional IOE also supports features such as:

- MultiVolt I/O interface
- Dedicated circuitry for external memory interface
- Input delay
- Four output-current strength settings for single-ended I/Os
- Four slew rate settings for both single-ended and differential I/Os
- Four output delay settings for single-ended I/Os
- Six I/O delay settings for single-ended I/Os
- Optional bus-hold
- Optional pull-up resistor
- Optional open-drain output
- Optional on-chip series termination with or without calibration
- Optional on-chip parallel termination with calibration
- Optional on-chip differential termination
- Optional PCI clamping diode

MultiVolt I/O Interface

The HardCopy IV architecture supports the MultiVolt I/O interface feature that allows HardCopy IV devices in all packages to interface with systems of different supply voltages.

The V_{CCIO} pins can be connected to a 1.2-, 1.5-, 1.8-, 2.5-, or 3.0-V power supply, depending on the output requirements. The output levels are compatible with systems of the same voltage as the power supply. (For example, when V_{CCIO} pins are connected to a 1.5-V power supply, the output levels are compatible with 1.5-V systems).

The HardCopy IV V_{CCPD} power pins must be connected to a 2.5-, or 3.0-V power supply. Using these power pins to supply the pre-driver power to the output buffers increases the performance of the output pins. Table 6–5 summarizes HardCopy IV MultiVolt I/O support.

Table 6–5. HardCopy IV MultiVolt I/O Support (Part 1 of 2) *(Note 1)*

V_{CCIO} *(V)*	Input Signal (V)						Output Signal (V)					
	1.2	1.5	1.8	2.5	3.0	3.3	1.2	1.5	1.8	2.5	3.0	3.3
1.2	✓	—	—	—	—	—	✓	—	—	—	—	—
1.5	—	✓	✓*(2)*	—	—	—	—	✓	—	—	—	—

Table 6–5. HardCopy IV MultiVolt I/O Support (Part 2 of 2) *(Note 1)*

V_{CCIO} *(V)*	Input Signal (V)						Output Signal (V)					
	1.2	1.5	1.8	2.5	3.0	3.3	1.2	1.5	1.8	2.5	3.0	3.3
1.8	—	✓*(2)*	✓	—	—	—	—	—	✓	—	—	—
2.5	—	—	—	✓	✓*(2)*	✓*(2)*	—	—	—	✓	—	—
3.0	—	—	—	✓	✓	✓*(2)*	—	—	—	—	✓	—
3.3 *(3)*	—	—	—	—	—	—	—	—	—	—	—	—

Notes to Table 6–5:

(1) The 3.3-V I/O standard is supported using V_{CCIO} at 3.0 V.

(2) The pin current may be slightly higher than the default value. You must verify that the driving device's V_{OL} maximum and V_{OH} minimum voltages do not violate the applicable HardCopy IV V_{IL} maximum and V_{IH} minimum voltage specifications.

(3) Use clamping diodes for all I/O pins when the input signal is 3.3 V. Altera recommends that you use an external clamping diode on the row I/O pins when the input signal is 3.3 V. Refer to "3.3- and 3.0-V I/O Interface" for more information.

3.3- and 3.0-V I/O Interface

Similar to Stratix IV buffers, HardCopy IV I/O buffers support 3.3-V I/O standards. You can use them as transmitters or receivers in your system. The output high voltage (V_{OH}), output low voltage (V_{OL}), input high voltage (V_{IH}), and input low voltage (V_{IL}) levels meet the 3.3-V I/O standards specifications defined by EIA/JEDEC Standard JESD8-B with margin when the HardCopy IV V_{CCIO} voltage is powered by 3.0 V.

To ensure device reliability and proper operation when interfacing with a 3.3-V I/O system using HardCopy IV devices, it is important to make sure that the absolute maximum ratings of HardCopy IV devices are not violated.

Altera recommends performing IBIS simulation to determine that the overshoot and undershoot voltages are within the guidelines.

When using the HardCopy IV device as a transmitter, some techniques can be used to limit the overshoot and undershoot at the I/O pins, such as using slow slew rate and series termination, but they are not required. Transmission line effects that cause large voltage deviation at the receiver are associated with impedance mismatch between the driver and transmission line. By matching the impedance of the driver to the characteristic impedance of the transmission line, you can significantly reduce overshoot voltage. You can use a series termination resistor placed physically close to the driver to match the total driver impedance to transmission line impedance. HardCopy IV devices support series on-chip termination (OCT) for all LVTTL/LVCMOS I/O standards in all I/O banks.

When using the HardCopy IV device as a receiver, a technique you can use to limit the overshoot, though not required, is using a clamping diode (on-chip or off-chip). HardCopy IV devices provide an optional on-chip PCI clamping diode for column I/O pins. You can use this diode to protect I/O pins against overshoot voltage.

The following features are identical to those in Stratix IV devices:

- External memory interface
- High-speed differential I/O with DPA support
- Four levels of pre-emphasis for LVDS transmitters
- Four levels of differential output voltage for LVDS transmitters

- Output current strength
- Slew rate control
- Output buffer delay
- Open-drain output
- Bus hold
- Pull-up resistor

For more information about particular features, refer to the *I/O Features in Stratix IV Devices* chapter in volume 1 of the *Stratix IV Device Handbook*.

External Memory Interfaces

In addition to the I/O registers in each IOE, HardCopy IV devices also have dedicated registers and phase-shift circuitry on all I/O banks for interfacing with external memory interfaces.

For more information about external memory interfaces, refer to the *External Memory Interfaces* chapter.

High-Speed Differential I/O with DPA Support

HardCopy IV devices have the following dedicated circuitry for high-speed differential I/O support:

- Differential I/O buffer
- Transmitter serializer
- Receiver deserializer
- Data realignment
- Dynamic phase aligner
- Synchronizer (FIFO buffer)
- Phase-locked loops (PLLs)

For more information about DPA support, refer to the *High-Speed Differential I/O Interfaces with DPA* chapter.

On-Chip Termination Support and I/O Termination Schemes

HardCopy IV devices support the same termination schemes and on-chip termination (OCT) architecture as Stratix IV devices. I/O termination provides impedance matching and helps maintain signal integrity while on-chip termination saves board space and reduces external component costs.

HardCopy IV devices support on-chip series termination (R_S) with or without calibration, parallel (R_T) with calibration, dynamic series and parallel termination for single-ended I/O standards, and on-chip differential termination (R_D) for differential LVDS I/O standards.

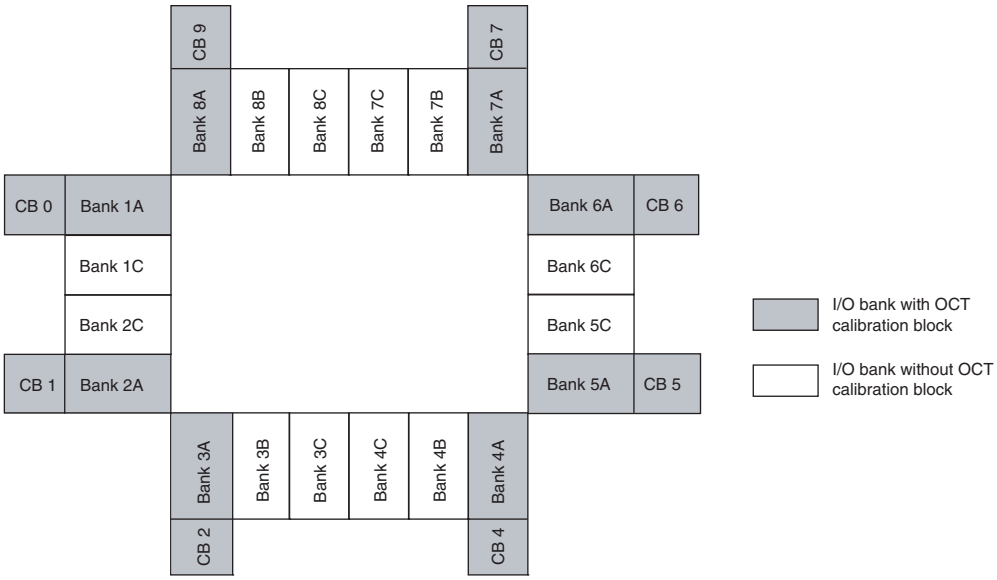
HardCopy IV devices support OCT in all I/O banks by selecting one of the OCT I/O standards. Unlike Stratix IV devices, which support up to 10 calibration blocks, HardCopy IV devices support up to eight OCT calibration blocks.

For more information on particular features, refer to the *I/O Features in Stratix IV Devices* chapter in volume 1 of the *Stratix IV Device Handbook*.

OCT Calibration Block Location

Figure 6–3 shows the location of OCT calibration blocks in HardCopy IV devices.

Figure 6–3. OCT Calibration Block (CB) Location in HardCopy IV Devices *(Note 1)*



Note to Figure 6–3:

(1) Figure 6–3 is a top view of the silicon die that corresponds to a reverse view for flipchip packages. It is a graphical representation only.

You can calibrate the I/O banks with any OCT calibration block with the same V_{CCIO}. Also, I/Os are allowed to transmit data during OCT calibration.

For more information about the OCT calibration modes of operation and their implementation, refer to the *I/O Features in Stratix IV Devices* chapter in volume 1 of the *Stratix IV Device Handbook*.

Design Considerations

While HardCopy IV devices feature various I/O capabilities for high-performance and high-speed system designs, there are several other considerations that require attention to ensure the success of those designs. These design practices are consistent with the design practices for Stratix IV devices.

I/O Banks Restrictions

Each I/O bank can simultaneously support multiple I/O standards. The following sections provide guidelines for mixing non-voltage-referenced and voltage-referenced I/O standards in HardCopy IV devices.

Non-Voltage-Referenced Standards

Each HardCopy IV I/O bank has its own VCCIO pins and can be powered by only one V_{CCIO} voltage supply level, either 1.2-, 1.5-, 1.8-, 2.5-, or 3.0-V. An I/O bank can simultaneously support any number of input signals with different I/O standard assignments, as shown in Table 6–5. For example, an I/O bank with a 2.5-V V_{CCIO} setting can support 2.5-V standard inputs and outputs and 3.0-V LVCMOS inputs (not output or bidirectional pins).

For output signals, a single I/O bank supports non-voltage-referenced output signals that are driving at the same voltage as V_{CCIO}. Because an I/O bank can only have one V_{CCIO} value, it can only drive out that one value for non-voltage-referenced signals.

Voltage-Referenced Standards

To accommodate voltage-referenced I/O standards, each HardCopy IV device's I/O bank, such as 1A and 1C, supports separate VREF pins feeding its individual VREF bus. You cannot use the VREF pins as generic I/O pins. Thus, if an I/O bank does not use any voltage-referenced I/O standards, the VREF pin for that I/O bank must be tied to V_{CCIO} or GND. Each bank can only have a single V_{CCIO} voltage level and a single V_{REF} voltage level at a given time.

An I/O bank featuring single-ended or differential standards can support voltage-referenced standards as long as all voltage-referenced standards use the same V_{REF} setting.

For performance reasons, voltage-referenced input standards use their own V_{CCPD} level as the power source. This feature allows you to place voltage-referenced input signals in an I/O bank with a V_{CCIO} of 2.5 or below. For example, you can place HSTL-15 input pins in an I/O bank with a 2.5-V V_{CCIO}.

Voltage-referenced bidirectional and output signals must be the same as the I/O bank's V_{CCIO} voltage. For example, you can only place SSTL-2 output pins in an I/O bank with a 2.5-V V_{CCIO}.

Mixing Voltage-Referenced and Non-Voltage-Referenced Standards

An I/O bank can support both non-voltage-referenced and voltage-referenced pins by applying each of the rule sets individually. For example, an I/O bank can support SSTL-18 inputs and 1.8-V inputs and outputs with a 1.8-V V_{CCIO} and a 0.9-V V_{REF}. Similarly, an I/O bank can support 1.5-V standards, 1.8-V inputs (but not outputs), and HSTL and HSTL-15 I/O standards with a 1.5-V V_{CCIO} and 0.75-V V_{REF}.

Non-Socket Replacement of the FPGA with HardCopy

HardCopy IV E devices offer non-socket replacement of the FPGA devices. Non-socket replacement of the FPGA with a HardCopy device requires a board redesign. Table 6–6 lists the non-socket replacement options.

To ensure I/O resource availability, refer to the *Mapping Stratix IV Device Resources to HardCopy IV Devices* chapter.

Table 6–6. HardCopy IV Non-Socket Replacement I/O Resource Availability

HardCopy IV E Device	Stratix IV Prototype Device	HardCopy IV E I/O Pins	Stratix IV I/O Pins	HardCopy IV E Full Duplex LVDS Pairs	Stratix IV Full Duplex LVDS Pairs
HC4E25WF484N	EP4SE230F29 (F780)	296	488	48	112
HC4E25FF484N	EP4SE230F29 (F780)	296	488	48	112

Document Revision History

Table 6–7 lists the revision history for this chapter.

Table 6–7. Document Revision History

Date	Version	Changes Made
January 2010	2.1	■ Updated Table 6–5. ■ Minor text edits.
June 2009	2.0	■ Added HardCopy IV GX support. ■ Added new Table 6–1. ■ Added new Figure 6–2. ■ Added new Table 6–6. ■ Added new sections “External Memory Interfaces”, “High-Speed Differential I/O with DPA Support”, and “Non-Socket Replacement of the FPGA with HardCopy.”
December 2008	1.0	Initial release.

7. External Memory Interfaces in HardCopy IV Devices

HIV51007-2.1

This chapter describes the hardware features that support high-speed memory interfacing for each double data rate (DDR) memory standard in HardCopy® IV devices. HardCopy IV devices feature delay-locked loops (DLLs), phase-locked loops (PLLs), dynamic on-chip termination (OCT), read and write leveling, and deskew circuitry.

This chapter contains the following sections:

Similar to the Stratix® IV I/O structure, the HardCopy IV I/O structure has been redesigned to provide flexible and high-performance support for existing and emerging external memory standards. These include high-performance DDR memory standards such as DDR3, DDR2, DDR SDRAM, QDRII+, QDRII SRAM, and RLDRAM II.

HardCopy IV devices offer the same external memory interface features found in Stratix IV devices. These features include DLLs, PLLs, dynamic OCT, trace mismatch compensation, read and write leveling, deskew circuitry, half data rate (HDR) blocks, 4- to 36-bit DQ group widths, and DDR external memory support on all sides of the HardCopy IV device. HardCopy IV devices provide an efficient architecture to quickly and easily fit wide external memory interfaces with the small modular I/O bank structure.

HardCopy IV devices are designed to support the same I/O standards and implementation guidelines for external memory interfaces as Stratix IV devices.

In addition, a self-calibrating megafunction (ALTMEMPHY), optimized to take advantage of the HardCopy IV I/O structure, and the Quartus® II timing analysis tool (TimeQuest Timing Analyzer) provide a complete solution for the highest reliable frequency of operation across process, voltage, and temperature (PVT) variations.

For more information about the ALTMEMPHY megafunction, refer to the *External DDR Memory PHY Interface Megafunction User Guide (ALTMEMPHY)*.

Altera recommends enabling the PLL reconfiguration feature and the DLL phase offset feature (DLL reconfiguration) for HardCopy IV devices. Because HardCopy IV devices are mask programmed, they cannot be changed after the silicon is fabricated. By implementing these two features, you can perform timing adjustments to improve or resolve timing issues after the silicon is fabricated.

Table 7–1 and Table 7–2 list the maximum clock rate HardCopy IV devices can support with external memory devices.

Table 7–1. HardCopy IV Maximum Clock Rate Support for External Memory Interfaces with Half-Rate Controller *(Note 1)*, *(2)*

Memory Standards	Performance-Optimized Flip Chip Package (F)		Cost-Optimized Flip Chip Package (L)		Low-Cost Wirebond Package (W)	
	Top and Bottom I/O Banks (MHz)	Left and Right I/O Banks (MHz)	Top and Bottom I/O Banks (MHz)	Left and Right I/O Banks (MHz)	Top and Bottom I/O Banks (MHz)	Left and Right I/O Banks (MHz)
DDR3 SDRAM	533 *(3)*	533 *(3)*	400	400	—	—
DDR2 SDRAM	333	333	333	333	200	200
DDR SDRAM	200	200	200	200	150	150
QDRII+ SRAM	350	350	350	350	150	150
QDRII SRAM	300	300	300	300	150	150
RLDRAM II	400 *(4)*	400 *(4)*	333	333	150	150

Notes to Table 7–1:

(1) Pending silicon characterization.

(2) The maximum clock rate support for external memory interfaces applies to commercial grade devices.

(3) Pending IP support. Actual achievable performance is based on design and system-specific factors. For 533MHz DDR3 support, contact Altera.

(4) Pending IP support. Actual achievable performance is based on design and system-specific factors. For 400MHz RLDRAM II support, contact Altera.

Table 7–2. HardCopy IV Maximum Clock Rate Support for External Memory Interfaces with Full-Rate Controller *(Note 1)*, *(2)*

Memory Standards	Performance-Optimized Flip Chip Package (F)		Cost-Optimized Flip Chip Package (L)		Low-Cost Wirebond Package (W)	
	Top and Bottom I/O Banks (MHz)	Left and Right I/O Banks (MHz)	Top and Bottom I/O Banks (MHz)	Left and Right I/O Banks (MHz)	Top and Bottom I/O Banks (MHz)	Left and Right I/O Banks (MHz)
DDR2 SDRAM *(3)*	267	267	233	233	200	200
DDR SDRAM	200	200	200	200	133	133

Notes to Table 7–2:

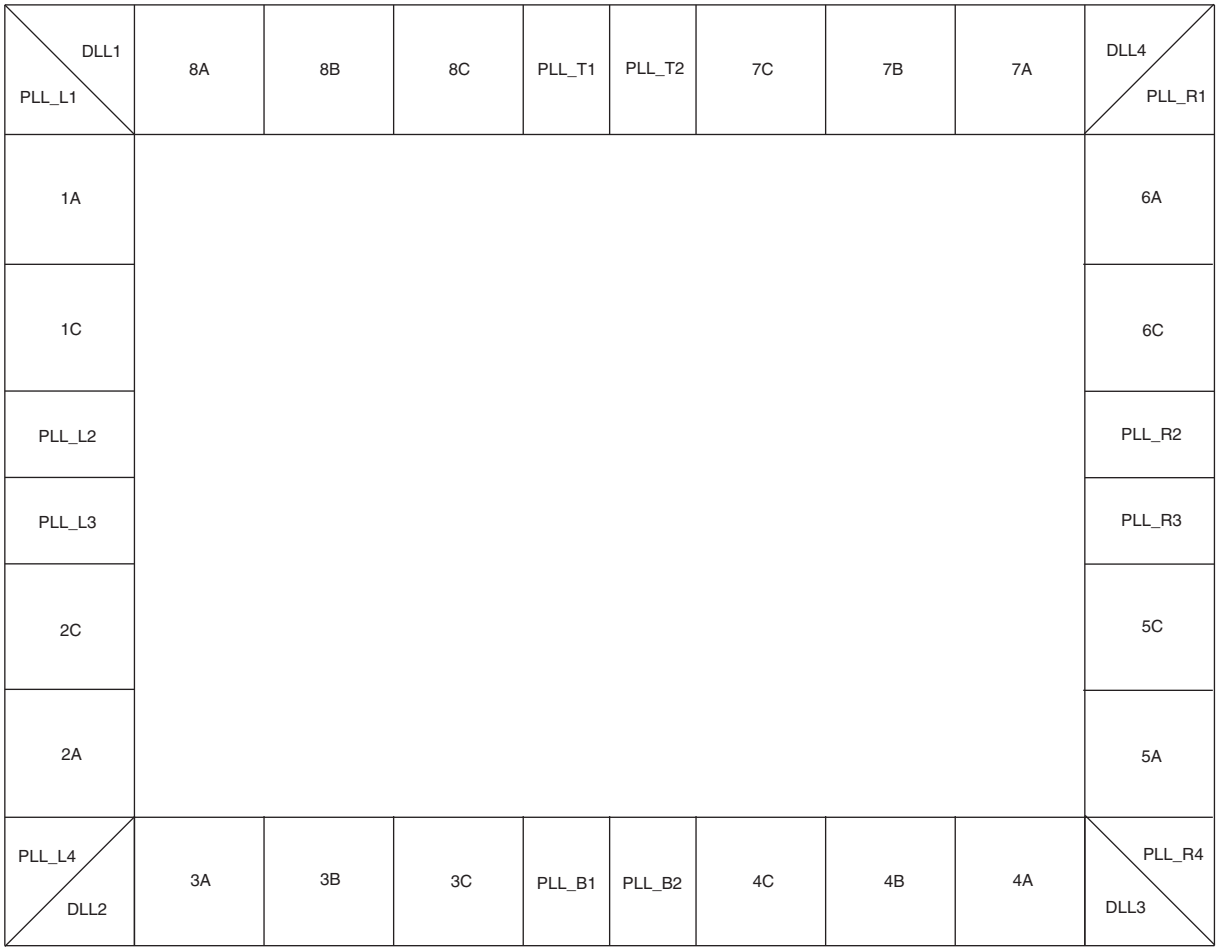
(1) Pending silicon characterization.

(2) The maximum clock rate support for external memory interfaces applies to commercial grade devices.

(3) Altera recommends using ALTMEMPHY AFI mode to achieve these quoted maximum clock rates due to the lower performance of non-AFI mode.

Figure 7–1 shows a package-bottom view for HardCopy IV E external memory support. Figure 7–2 shows a package-bottom view for HardCopy IV GX external memory support, PLLs, DLLs, and I/O banks. The number of available I/O banks and PLLs depends on the device density.

Figure 7–1. HardCopy IV E Package-Bottom View *(Note 1), (2), (3)*



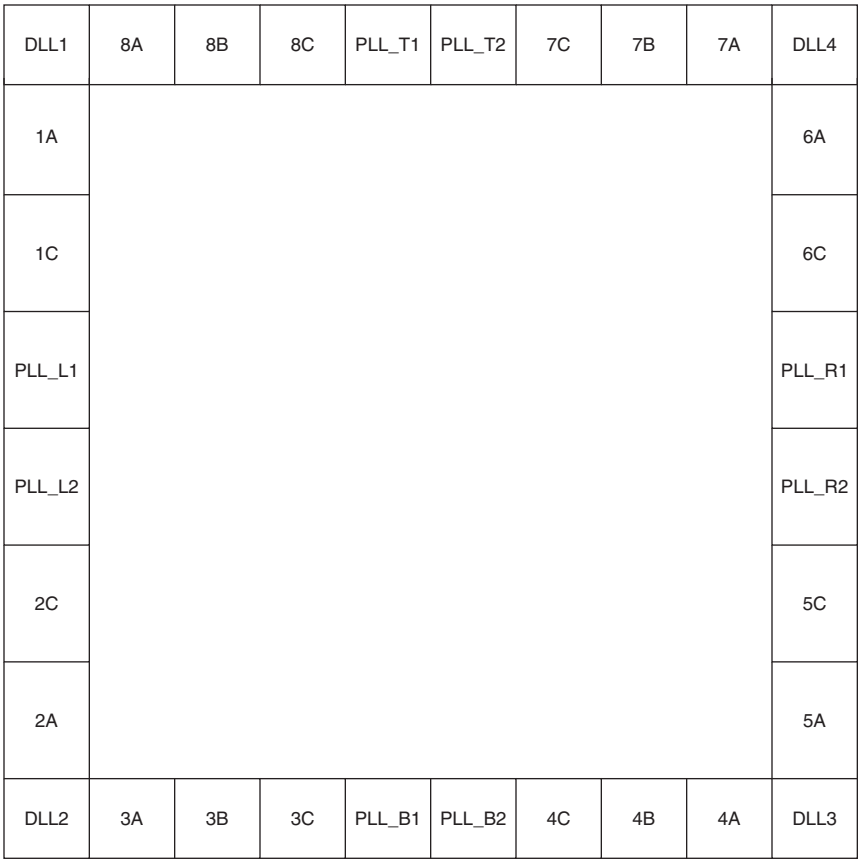
Notes to Figure 7–1:

(1) The number of I/O banks and PLLs available depend on the device density.

(2) Not all HardCopy IV E devices support I/O banks 1B, 2B, 5B, and 6B.

(3) There is only one PLL on each side of the HC4E21, HC4E31, and HC4E41 devices. These devices do not support I/O banks 3B, 4B, 7B, and 8B.

Figure 7–2. HardCopy IV GX Package-Bottom View *(Note 1)*, *(2)*, *(3)*



Notes to Figure 7–2:

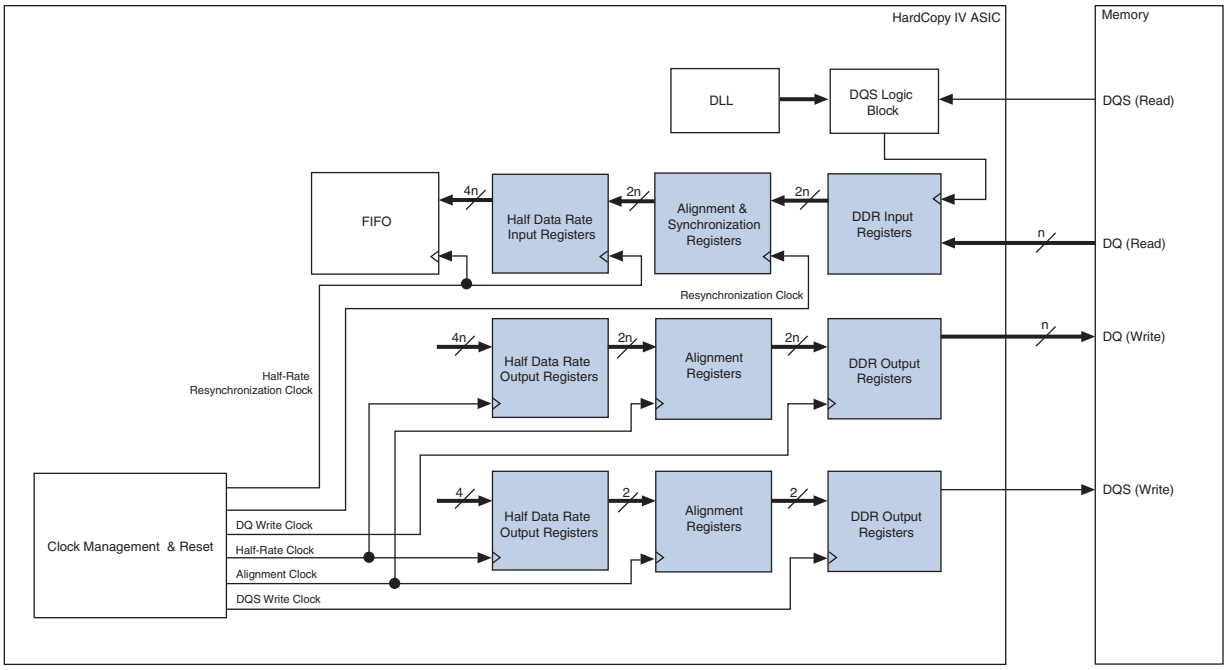
(1) The number of I/O banks and PLLs available depend on the device density.

(2) All HardCopy IV GX devices do not support 1B, 2B, 5B, and 6B I/O banks.

(3) There are no right side PLLs in HC4GX15L devices. These devices do not support I/O banks 3B, 4B, 7B, and 8B.

Figure 7–3 shows the memory interface data path that uses all the HardCopy IV I/O element (IOE) features.

Figure 7–3. External Memory Interface Data Path Overview *(Note 1)*, *(2)*, *(3)*



Notes to Figure 7–3:

(1) You can bypass each register block.

(2) The blocks for each memory interface may differ slightly.

(3) These signals may be bidirectional or unidirectional, depending on the memory standard. When bidirectional, the signal is active during both read and write operations.

Memory Interfaces Pin Support

A typical memory interface requires data (D, Q, or DQ), data strobe (DQS and DQSn/CQn), address, command, and clock pins. Some memory interfaces use data mask (DM) pins to enable write masking and QVLD pins to indicate that the read data is ready to be captured. This section describes how HardCopy IV devices support these pins.

Data and Data Clock/Strobe Pins

HardCopy IV DDR memory interface read data-strobes or clocks are called DQS pins. Depending on the memory specifications, the DQS pins can be bidirectional single-ended signals (in DDR2 and DDR SDRAM), bidirectional differential signals (DDR3 and DDR2 SDRAM), unidirectional differential signals (in RLDRAM II), or unidirectional complementary signals (QDRII+ and QDRII SRAM). Connect the unidirectional read and write data-strobes or clocks to HardCopy IV DQS pins.

HardCopy IV devices offer differential input buffers for differential read data-strobe/clock operations and provide an independent DQS logic block for each CQn pin for complementary read data-strobe/clock operations. In the HardCopy IV pin tables, the differential DQS pin-pairs are denoted as DQS and DQSn pins; the complementary DQS signals are denoted as DQS and CQn pins. DQSn and CQn pins are marked separately in the pin table. Each CQn pin connects to a DQS logic block and the shifted CQn signals go to the negative-edge input registers in the DQ IOE registers.

☞ Use differential DQS signaling for DDR2 SDRAM interfaces running higher than 333 MHz.

HardCopy IV DDR memory interface data pins are called DQ pins. The DQ pins can be bidirectional signals, as in DDR3, DDR2, and DDR SDRAM, and RLDRAM II common I/O (CIO) interfaces, or unidirectional signals, as in QDRII+, QDRII SRAM, and RLDRAM II separate I/O (SIO) devices. Connect the unidirectional read data signals to HardCopy IV DQ pins and the unidirectional write data signals to a DQS/DQ group other than the read DQS/DQ group. Furthermore, the write clocks must be assigned to the DQS/DQSn pins associated with this write DQS/DQ group. Do not use the DQS/CQn pin-pair for write clocks.

☞ Using a DQS/DQ group for the write data signals minimizes output skew and allows access to the write leveling circuitry (for DDR3 SDRAM interfaces). These pins also have access to deskewing circuitry that can compensate for delay mismatch between signals on the bus.

Table 7–3 lists the pin connections between a HardCopy IV device and an external memory device.

Table 7–3. HardCopy IV Memory Interfaces Pin Utilization

Pin Description	Memory Standard	HardCopy IV Pin Utilization
Read Data	All	DQ
Write Data	All	DQ *(1)*
Parity, DM, BWSn, NWSn, QVLD, ECC	All	DQ *(1)*, *(2)*
Read Strobes/Clocks	DDR3 SDRAM DDR2 SDRAM (with differential DQS signaling) *(3)* RLDRAM II	Differential DQS/DQSn
	DDR2 SDRAM (with single-ended DQS signaling) *(3)* DDR SDRAM	Single-ended DQS
	QDRII+ SRAM QDRII SRAM	Complementary DQS/CQn
Write Clocks	QDRII+ SRAM *(4)* QDRII SRAM *(4)* RLDRAM II SIO	Any unused DQS and DQSn pin pairs *(1)*
Memory Clocks	DDR3 SDRAM	Any unused DQ or DQS pins with `DIFFIO_RX` capability for the `mem_clk[0]` and `mem_clk_n[0]` signals.
		Any unused DQ or DQS pins with `DIFFOUT` capability for the `mem_clk[n:1]` *and* `mem_clk_n[n:1]` signals (where n is greater than or equal to 1).
	DDR2 SDRAM (with differential DQS signaling)	Any `DIFFIO_RX` pins for the `mem_clk[0]` and `mem_clk_n[0]` signals.
		Any unused `DIFFOUT` pins for the `mem_clk[n:1]` and `mem_clk_n[n:1]` signals (where n is greater than or equal to 1).
	DDR2 SDRAM (with single-ended DQS signaling) DDR SDRAM RLDRAM II	Any `DIFFOUT` pins
	QDRII+ SRAM *(4)* QDRII SRAM *(4)*	Any unused DQSn pin pairs *(1)*

Notes to Table 7–3:

(1) If the write data signals are unidirectional including the data mask pins, connect them to a separate DQS/DQ group other than the read DQS/DQ group. Connect the write clock to the DQS and DQSn pin-pair associated with that DQS/DQ group. Do not use the DQS and CQn pin-pair as write clocks.

(2) The BWSn, NWSn, and DM pins must be part of the write DQS/DQ group. Parity, QVLD, and ECC pins must be part of the read DQS/DQ group.

(3) DDR2 SDRAM supports either single-ended or differential DQS signaling.

(4) QDRII+/QDRII SRAM devices typically use the same clock signals for both write and memory clock pins (K/K# clocks) to latch data and address, and command signals. The clocks must be part of the DQS/DQ group in this case.

The DQS and DQ pin locations are fixed in the pin table. Memory interface circuitry is available in every HardCopy IV I/O bank. All memory interface pins support the I/O standards required to support DDR3, DDR2, DDR SDRAM, QDRII+, QDRII SRAM, and RLDRAM II devices.

HardCopy IV devices support DQS and DQ signals with DQ bus modes of ×4, ×8/×9, ×16/×18, or ×32/×36, although not all devices support DQS bus mode ×32/×36. When any of these pins are not used for memory interfacing, you can use them as user I/Os. In addition, you can use any DQSn or CQn pin not used for clocking as DQ (data) pins. Table 7–4 lists pin support per DQS/DQ bus mode, including the DQS and DQSn/CQn pin pair.

Table 7–4. HardCopy IV DQS/DQ Bus Mode Pins *(Note 1), (2), (3), (4), (5)*

Mode	DQSn Support	CQn Support	Parity or DM (Optional)	QVLD (Optional)	Typical Number of Data Pins per Group	Maximum Number of Data Pins per Group
×4	Yes	No	No	No	4	5
×8/×9	Yes	Yes	Yes	Yes	8 or 9	11
×16/×18	Yes	Yes	Yes	Yes	16 or 18	23
×32/×36	Yes	Yes	Yes	Yes	32 or 36	47

Notes to Table 7–4:

(1) The QVLD pin is not used in the ALTMEMPHY megafunction.

(2) This represents the maximum number of DQ pins (including parity, data mask, and QVLD pins) connected to the DQS bus network with single-ended DQS signaling. When you use differential or complementary DQS signaling, the maximum number of data per group decreases by one. This number may vary per DQS/DQ group in a particular device. Check the pin table for the accurate number per group.

(3) Two ×4 DQS/DQ groups are stitched to make a ×8/×9 group, so there are a total of 12 pins in this group.

(4) Four ×4 DQS/DQ groups are stitched to make a ×16/×18 group.

(5) Eight ×4 DQS/DQ groups are stitched to make a ×32/×36 group.

You can also use DQS/DQSn pins in some of the ×4 groups as R_{UP}/R_{DN} pins (listed in the pin table). You cannot use a ×4 DQS/DQ group for memory interfaces if any of its pin members are being used as R_{UP} and R_{DN} pins for OCT calibration. You may use the ×8/×9 group that includes this ×4 DQS/DQ group, if either of the following circumstances apply:

- You are not using DM pins with your differential DQS pins
- You are not using complementary or differential DQS pins

You can do this because a DQS/DQ ×8/×9 group is comprised of 12 pins, as the groups are formed by stitching two DQS/DQ groups in ×4 mode with six total pins each (Table 7–4). A typical ×8 memory interface contains 10 pins, consisting of one DQS, one DM, and eight DQ pins. If you choose your pin assignment carefully, you can use the two extra pins for R_{UP} and R_{DN}. In a DDR3 SDRAM interface, you must use differential DQS, which means that you only have one extra pin. In this case, pick different pin locations for the R_{UP} and R_{DN} pins (for example, in the bank that contains the address and command pins).

You cannot use the R_{UP} and R_{DN} pins shared with DQS/DQ group pins when using ×9 QDRII+/QDRII SRAM devices, as the R_{UP} and R_{DN} pins have a dual purpose with the CQn pins. In this case, pick different pin locations for the R_{UP} and R_{DN} pins to avoid conflict with memory interface pin placement. You have the choice of placing the R_{UP} and R_{DN} pins in the data-write group or in the same bank as the address and command pins. There is no restriction for using ×16/×18 or ×32/×36 DQS/DQ groups that include the ×4 groups in which the pin members are used as R_{UP} and R_{DN} pins. These groups contain enough extra pins that they can be used as DQS pins.

You must pick your DQS and DQ pins manually for the ×8/×9, ×16/×18, or ×32/×36 DQS/DQ group in which the members are used for R_{UP} and R_{DN}. Otherwise, the Quartus II software might not be able to place these pins correctly if there are no specific pin assignments and might give you a "no-fit" instead.

Table 7–5 lists the maximum number of DQS/DQ groups per side of the HardCopy IV E device.

Table 7–5. Number of DQS/DQ Groups in HardCopy IV E Devices per Side *(Note 1)*

Device	Package	Side	×4	×8/×9	×16/×18	×32/×36
HC4E25W	484-pin FineLine BGA	Left	12	4	0	0
		Bottom	5	2	0	0
		Right	12	4	0	0
		Top	5	2	0	0
	780-pin FineLine BGA	Left	12	4	0	0
		Bottom	13	6	0	0
		Right	12	4	0	0
		Top	13	6	0	0
HC4E25F	484-pin FineLine BGA	Left	12	4	0	0
		Bottom	5	2	0	0
		Right	12	4	0	0
		Top	5	2	0	0
	780-pin FineLine BGA	Left	14	6	2	0
		Bottom	17	8	2	0
		Right	14	6	2	0
		Top	17	8	2	0
HC4E35L HC4E35F	1152-pin FineLine BGA	Left	26	12	4	0
		Bottom	26	12	4	0
		Right	26	12	4	0
		Top	26	12	4	0
	1517-pin FineLine BGA	Left	26	12	4	0
		Bottom	38	18	8	4
		Right	26	12	4	0
		Top	38	18	8	4

Note to Table 7–5:

(1) In some ×4 groups, you can use the DQS/DQ pins as R_{UP}/R_{DN} pins. You cannot use these ×4 groups if the pins are used as R_{UP} and R_{DN} pins for OCT calibration. Ensure that the DQS/DQ groups you chose are not also used for OCT calibration.

Table 7–6 lists the maximum number of DQS/DQ groups per side of the HardCopy IV GX device.

Table 7–6. Number of DQS/DQ Groups in HardCopy IV GX Devices per Side

Device	Package	Side	×4 *(2)*	×8/×9	×16/×18	×32/×36
HC4GX15LA	780-pin FineLine BGA	Left *(1)*	14	6	2	0
		Bottom	17	8	2	0
		Right	0	0	0	0
		Top	17	8	2	0
HC4GX15L	780-pin FineLine BGA	Left	0	0	0	0
		Bottom	17	8	2	0
		Right	0	0	0	0
		Top	17	8	2	0
HC4GX25L	780-pin FineLine BGA	Left	0	0	0	0
		Bottom	18	8	2	0
		Right	0	0	0	0
		Top	18	8	2	0
HC4GX25L HC4GX25F	1152-pin FineLine BGA	Left	13	6	2	0
		Bottom	26	12	4	0
		Right	13	6	2	0
		Top	26	12	4	0
HC4GX35F	1152-pin FineLine BGA	Left	13	6	2	0
		Bottom	26	12	4	0
		Right	13	6	2	0
		Top	26	12	4	0
HC4GX35F	1517-pin FineLine BGA	Left	26	12	4	0
		Bottom	26	12	4	0
		Right	26	12	4	0
		Top	26	12	4	0

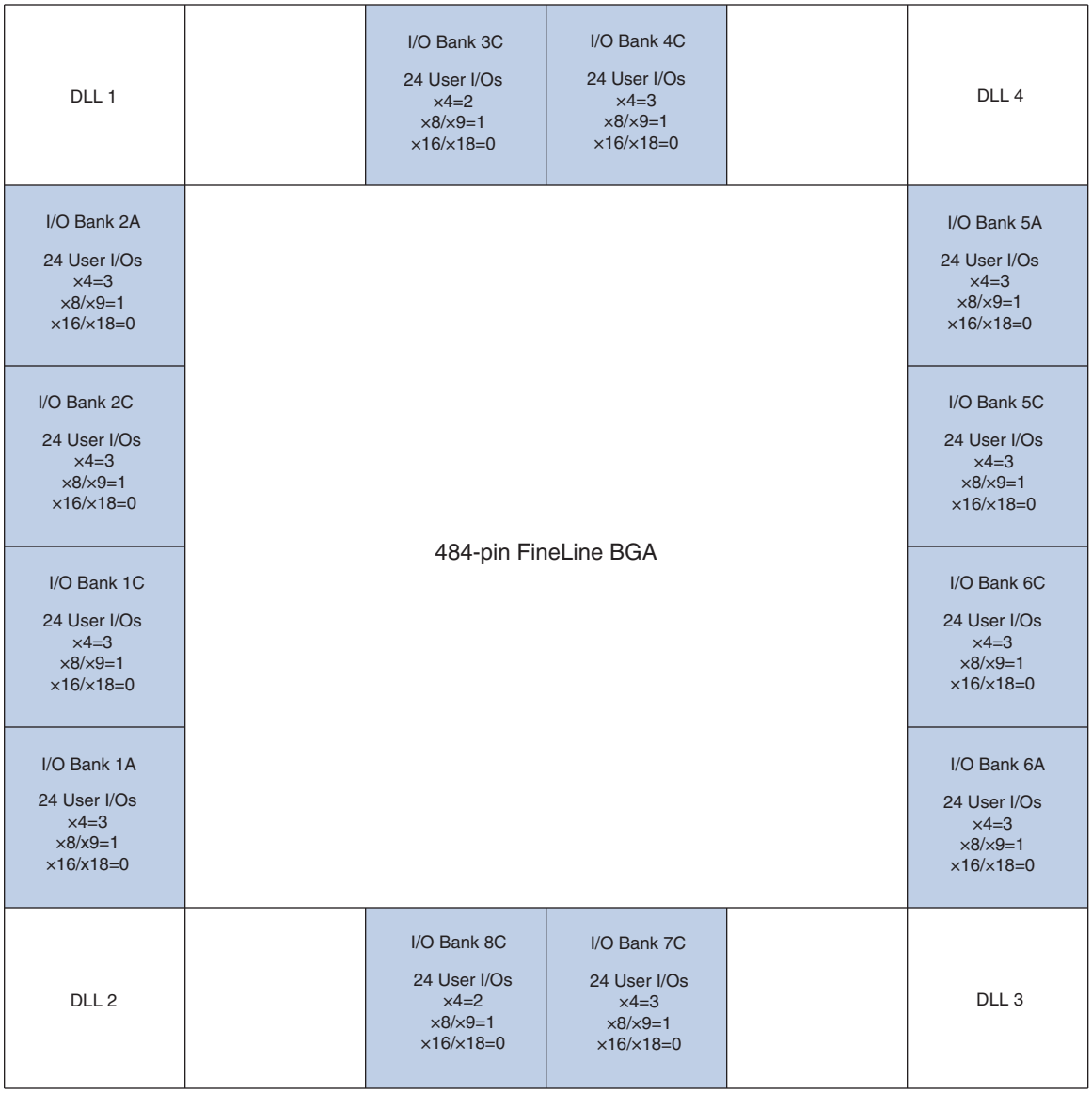
Notes to Table 7–6:

(1) There are no DQS/DQ groups in the left side of the HC4GX25LF780 devices.

(2) In some ×4 groups, you can use the DQS/DQ pins as R_{UP}/R_{DN} pins. You cannot use these ×4 groups if the pins are used as R_{UP} and R_{DN} pins for OCT calibration. Ensure that the DQS/DQ groups that you chose are not also used for OCT calibration.

Figure 7–4 through Figure 7–7 show the number of DQS/DQ groups available per bank in each HardCopy IV E device. These figures present the package-bottom view of the specified HardCopy IV E devices.

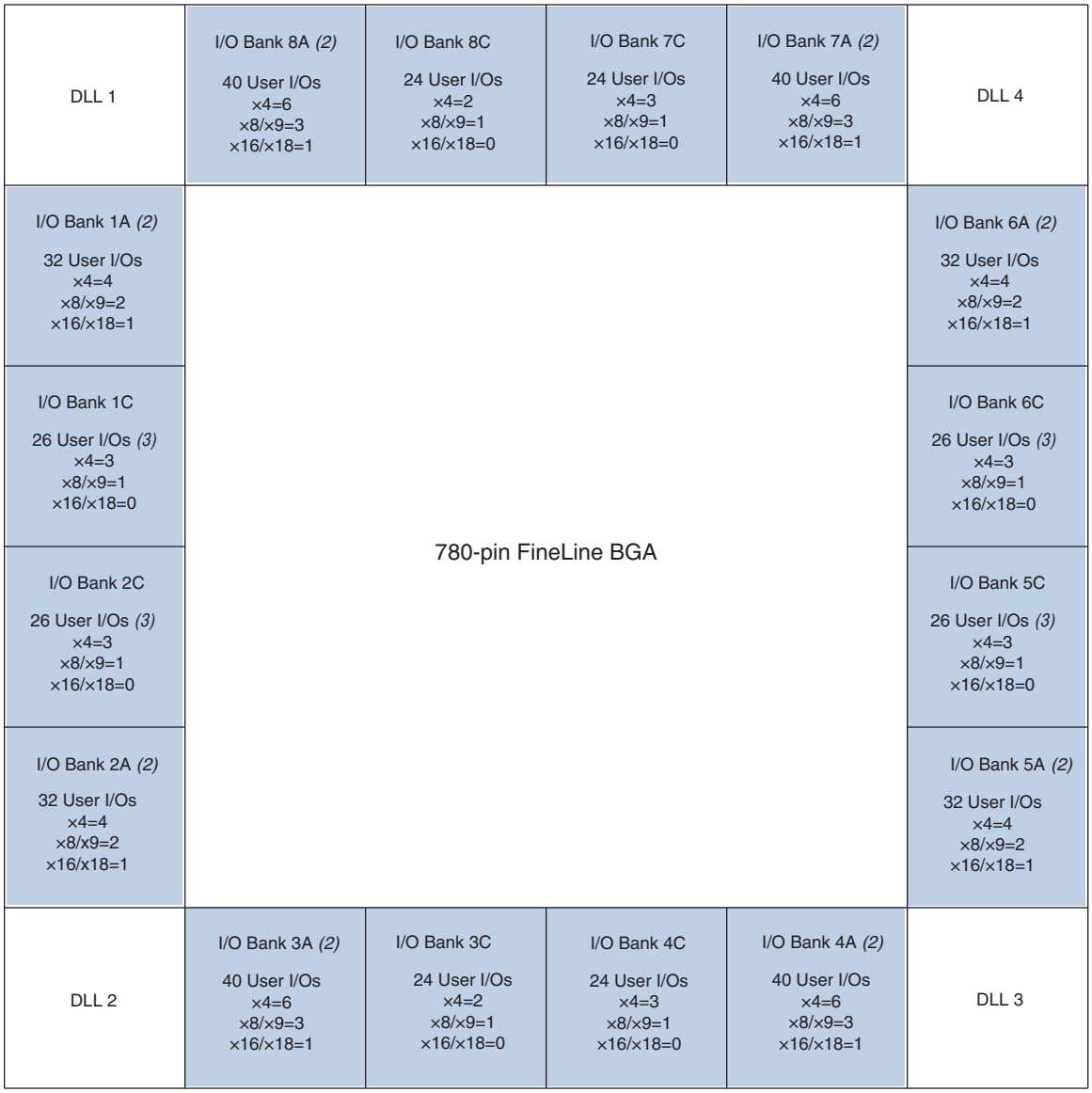
Figure 7–4. Number of DQS/DQ Groups per Bank in HC4E25W Devices in a 484-pin FineLine BGA Package *(Note 1)*



Note to Figure 7–4:

(1) These devices do not support ×32/×36 mode.

Figure 7–5. Number of DQS/DQ Groups per Bank in HC4E25F Devices in 780-pin FineLine BGA Package *(Note 1)*



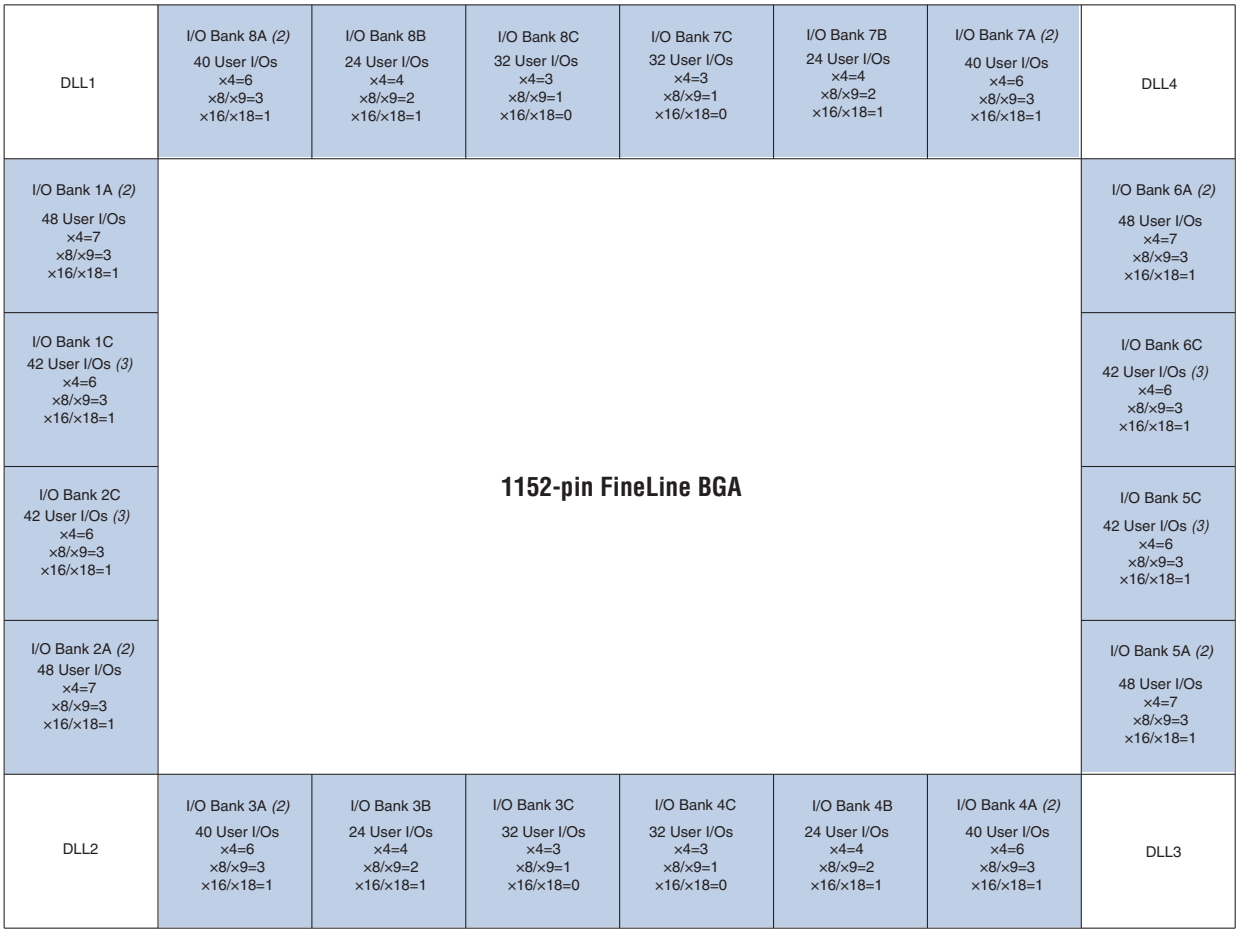
Notes to Figure 7–5:

(1) These devices do not support ×32/×36 mode.

(2) You can use the DQS/DQSn pins in some of the ×4 groups as R_{UP}/R_{DN} pins. You cannot use a ×4 group for memory interfaces if two pins in the group are used as R_{UP} and R_{DN} pins for OCT calibration. You can still use the ×16/×18 groups, including the ×4 groups. However, there are restrictions on using ×8/×9 groups that include these ×4 groups.

(3) All I/O pin counts include eight dedicated clock inputs (`CLK1p`, `CLK1n`, `CLK3p`, `CLK3n`, `CLK8p`, `CLK8n`, `CLK10p`, and `CLK10n`).

Figure 7–6. Number of DQS/DQ Groups in HC4E35L and HC4E35F Devices in 1152-pin FineLine BGA Package *(Note 1)*



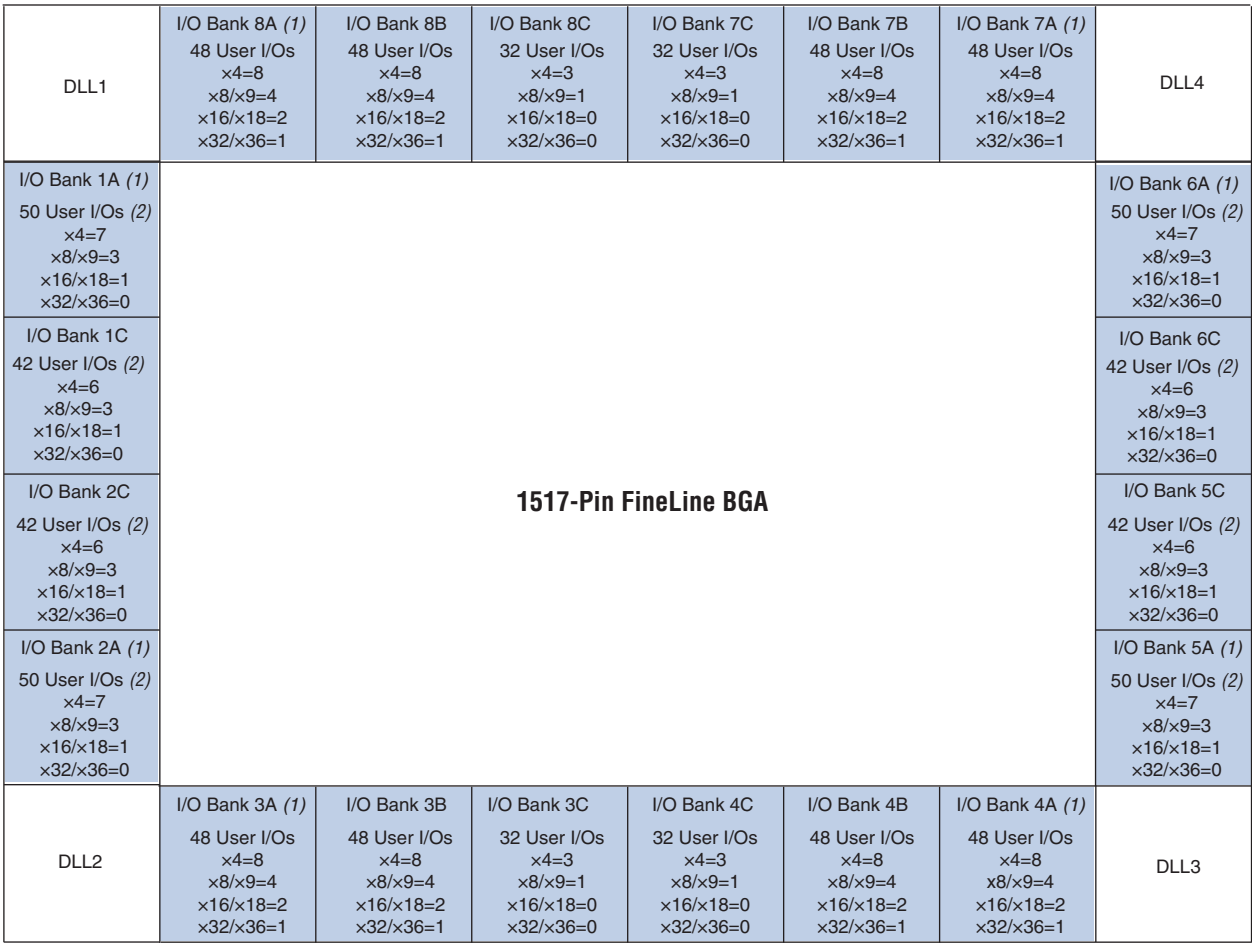
Notes to Figure 7–6:

(1) These devices do not support ×32/×36 mode.

(2) You can use the DQS/DQSn pins in some of the ×4 groups as R_{UP}/R_{DN} pins. You cannot use a ×4 group for memory interfaces if two pins in the group are used as R_{UP} and R_{DN} pins for OCT calibration. You can still use the ×16/×18 groups including the ×4 groups. However, there are restrictions on using ×8/×9 groups that include these ×4 groups.

(3) All I/O pin counts include eight dedicated clock inputs (`CLK1p`, `CLK1n`, `CLK3p`, `CLK3n`, `CLK8p`, `CLK8n`, `CLK10p`, and `CLK10n`).

Figure 7–7. Number of DQS/DQ Groups per Bank in HC4E35L and HC4E35F Devices in a 1517-pin FineLine BGA Package



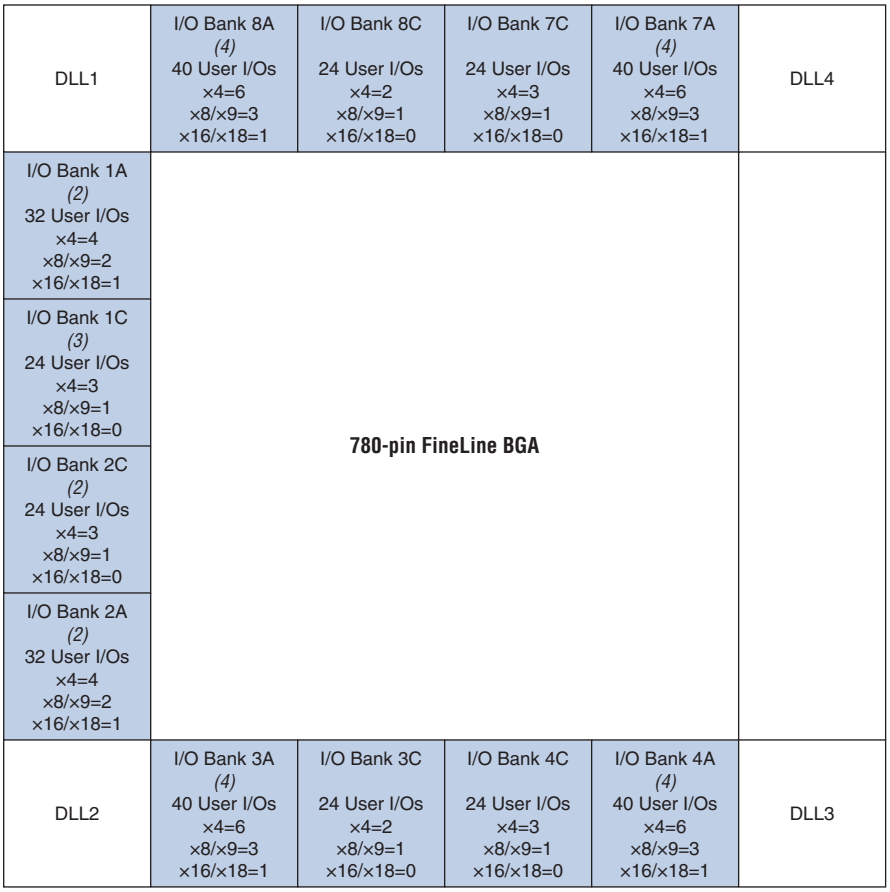
Notes to Figure 7–7:

(1) You can use the DQS/DQSn pins in some of the ×4 groups as R_{UP}/R_{DN} pins. You cannot use a ×4 group for memory interfaces if two pins in the group are used as R_{UP} and R_{DN} pins for OCT calibration. You can still use the ×16/×18 or ×32/×36 groups including the ×4 groups. However, there are restrictions on using ×8/×9 groups that include these ×4 groups.

(2) All I/O pin counts include eight dedicated clock inputs (`CLK1p`, `CLK1n`, `CLK3p`, `CLK3n`, `CLK8p`, `CLK8n`, `CLK10p`, and `CLK10n`) and eight dedicated corner PLL clock inputs (`PLL_L1_CLKp`, `PLL_L1_CLKn`, `PLL_L4_CLKp`, `PLL_L4_CLKn`, `PLL_R4_CLKp`, `PLL_R4_CLKn`, `PLL_R1_CLKp`, and `PLL_R1_CLKn`) that can be used for data inputs.

Figure 7–8 through Figure 7–11 show the number of DQS/DQ groups available per bank in HardCopy IV GX device. These figures present the package-bottom view of the specified HardCopy IV GX devices.

Figure 7–8. Number of DQS/DQ Groups per Bank in HC4GX15LA Devices in a 780-pin FineLine BGA Package *(Note 1)*



Notes to Figure 7–8:

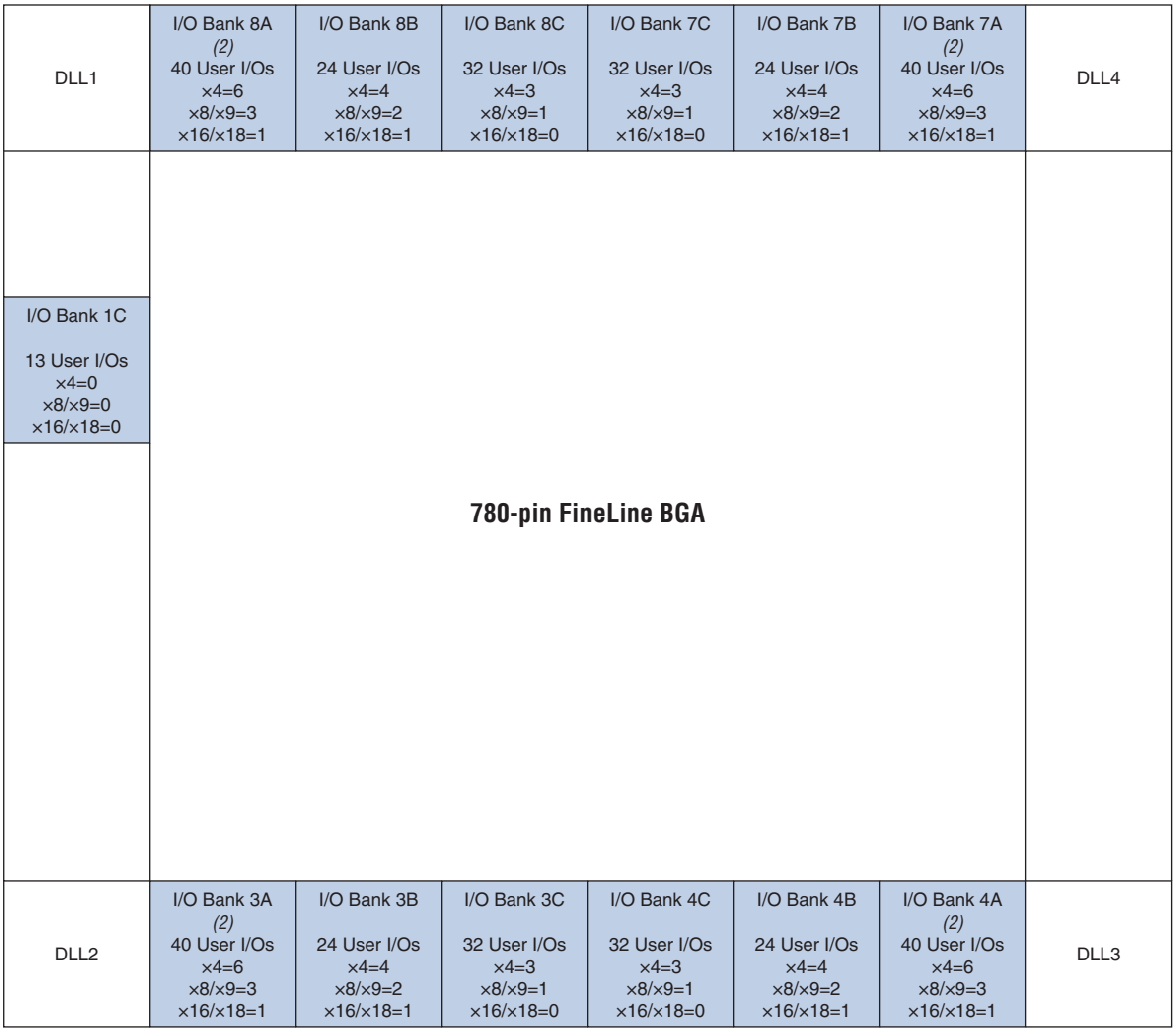
(1) These devices do not support ×32/×36 mode.

(2) There are no 1A, 2A, and 2C banks in HC4GX15L devices.

(3) There are no DQS/DQ groups, and 13 configuration pins in the bank 1C when migrating EP4SGX290 or EP4SGX360 prototype devices.

(4) You can use the DQS/DQSn pins in some of the ×4 groups as R_{UP}/R_{DN} pins. You cannot use a ×4 group for memory interfaces if two pins in the group are used as R_{UP} and R_{DN} pins for OCT calibration. You can still use the ×16/×18 groups, including the ×4 groups. However, there are restrictions on using ×8/×9 groups that include these ×4 groups.

Figure 7–9. Number of DQS/DQ Groups per Bank in HC4GX25L Devices in a 780-pin FineLine BGA Package *(Note 1)*, *(2)*

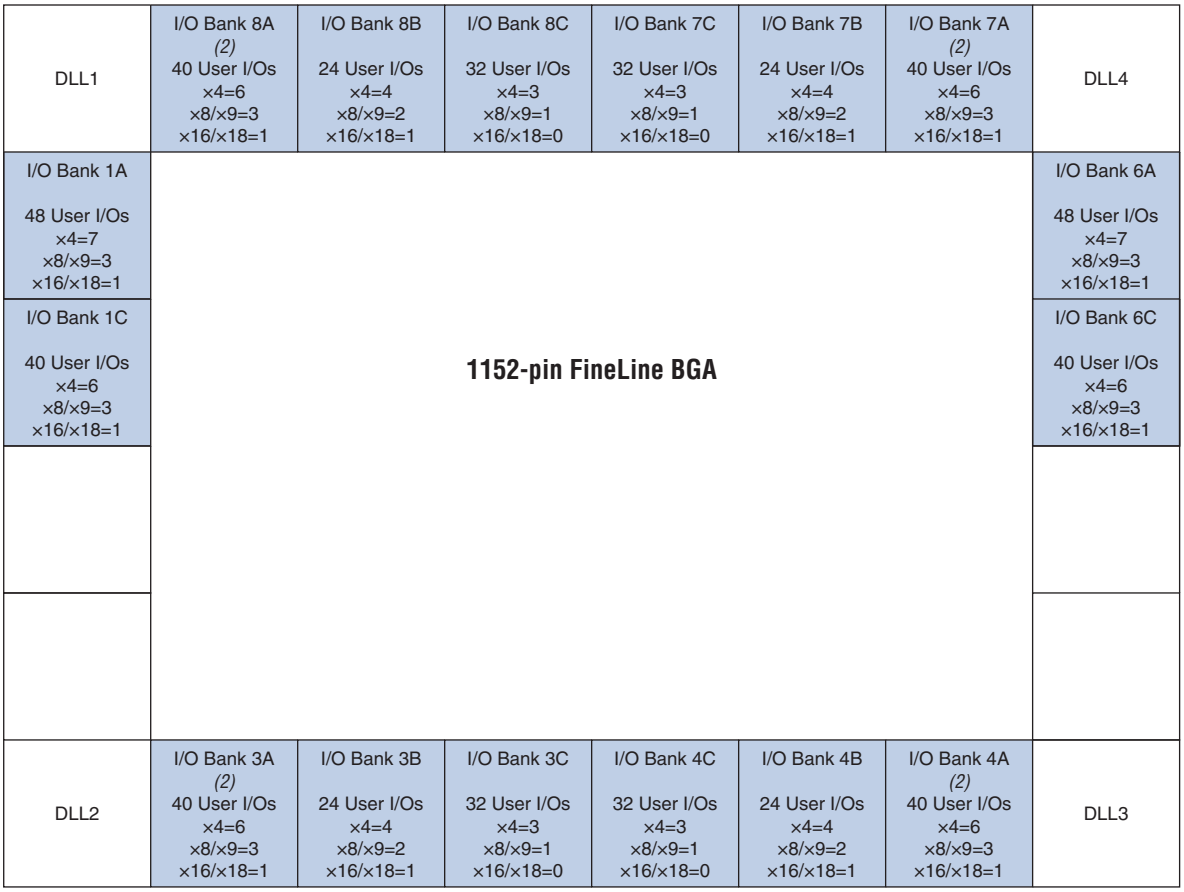


Notes to Figure 7–9:

(1) These devices do not support ×32/×36 mode.

(2) You can use the DQS/DQSn pins in some of the ×4 groups as R_{UP}/R_{DN} pins. You cannot use a ×4 group for memory interfaces if two pins in the group are used as R_{UP} and R_{DN} pins for OCT calibration. You can still use the ×16/×18 groups, including the ×4 groups. However, there are restrictions on using ×8/×9 groups that include these ×4 groups.

Figure 7–10. Number of DQS/DQ Groups per Bank in HC4GX25L, HC4GX25F and HC4GX35F Devices in a 1152-pin FineLine BGA Package *(Note 1)*, *(2)*

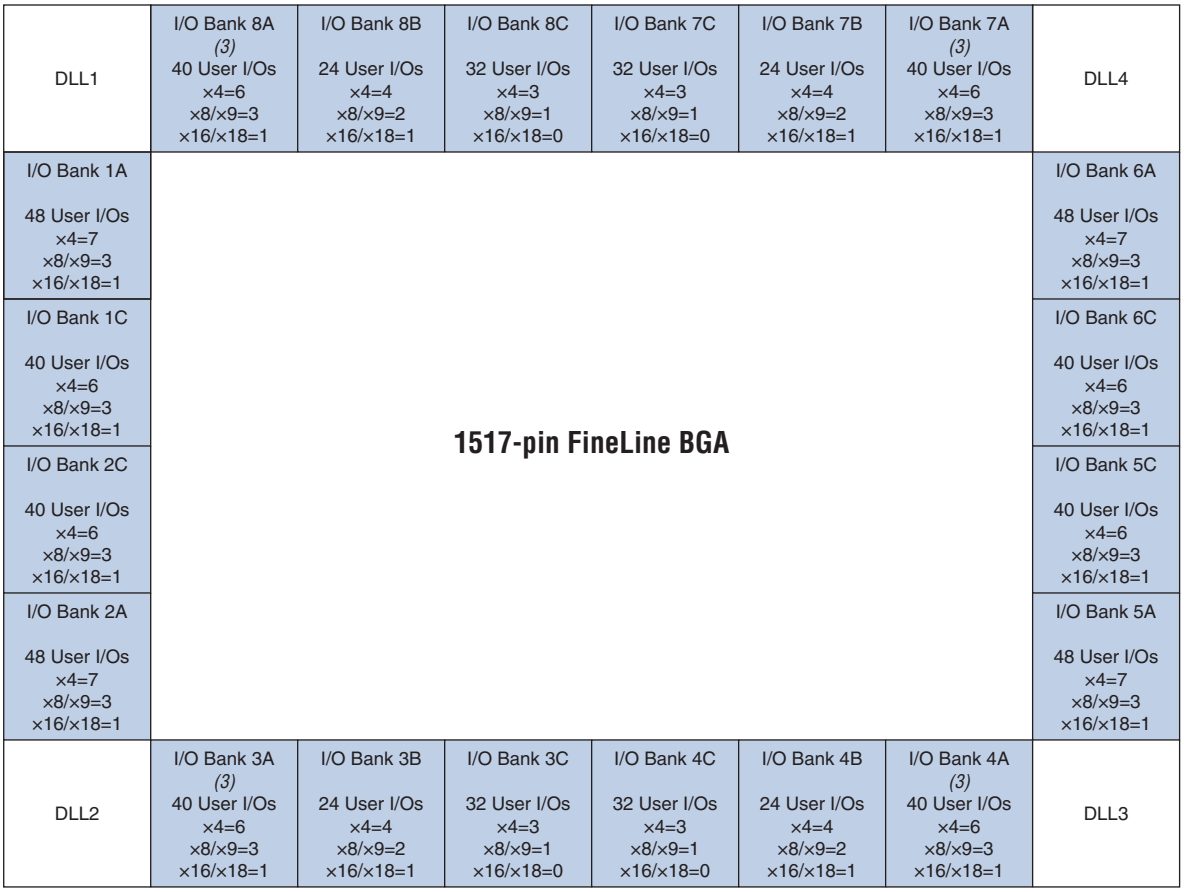


Notes to Figure 7–10:

(1) These devices do not support ×32/×36 mode.

(2) You can use the DQS/DQSn pins in some of the ×4 groups as R_{UP}/R_{DN} pins. You cannot use a ×4 group for memory interfaces if two pins in the group are used as R_{UP} and R_{DN} pins for OCT calibration. You can still use the ×16/×18 groups, including the ×4 groups. However, there are restrictions on using ×8/×9 groups that include these ×4 groups.

Figure 7–11. Number of DQS/DQ Groups per Bank in HC4GX35L and HC4GX35F Devices in a 1517-pin FineLine BGA Package *(Note 1)*, *(2)*



Notes to Figure 7–11:

(1) These devices do not support ×32/×36 mode.

(2) You can use the DQS/DQSn pins in some of the ×4 groups as R_{UP}/R_{DN} pins. You cannot use a ×4 group for memory interfaces if two pins in the group are used as R_{UP} and R_{DN} pins for OCT calibration. You can still use the ×16/×18 groups, including the ×4 groups. However, there are restrictions on using ×8/×9 groups that include these ×4 groups.

The DQS and DQSn pins are listed in the HardCopy IV pin tables as DQSXY and DQSnXY, respectively, where X denotes the DQS/DQ grouping number, and Y denotes whether the group is located on the top (T), bottom (B), left (L), or right (R) side of the device.

The corresponding DQ pins are marked as DQXY, where X indicates which DQS group the pins belong to and Y indicates whether the group is located on the top (T), bottom (B), left (L), or right (R) side of the device. For example, `DQS1L` indicates a DQS pin, located on the left side of the device, as shown in Figure 7–12. The DQ pins belonging to that group are shown as `DQ1L` in the pin table.

The numbering scheme starts from the top left side of the device going counter-clockwise. Figure 7–12 and Figure 7–13 show how the DQS/DQ groups are numbered in a package-bottom view of the device. The top and bottom sides of the HardCopy IV E device can contain up to 38 ×4 DQS/DQ groups; the left and right sides of the device can contain up to 26 ×4 DQS/DQ groups.The top and bottom sides of the HardCopy IV GX device can contain up to 26 ×4 DQS/DQ groups; the left and right sides of the device can contain up to 26 ×4 DQS/DQ groups.

The parity, DM, BWSn, ECC, and QVLD pins are shown as DQ pins in the pin table. When not used as memory interface pins, these pins are available as regular I/O pins.

Figure 7–12. DQS Pins in HardCopy IV E I/O Banks

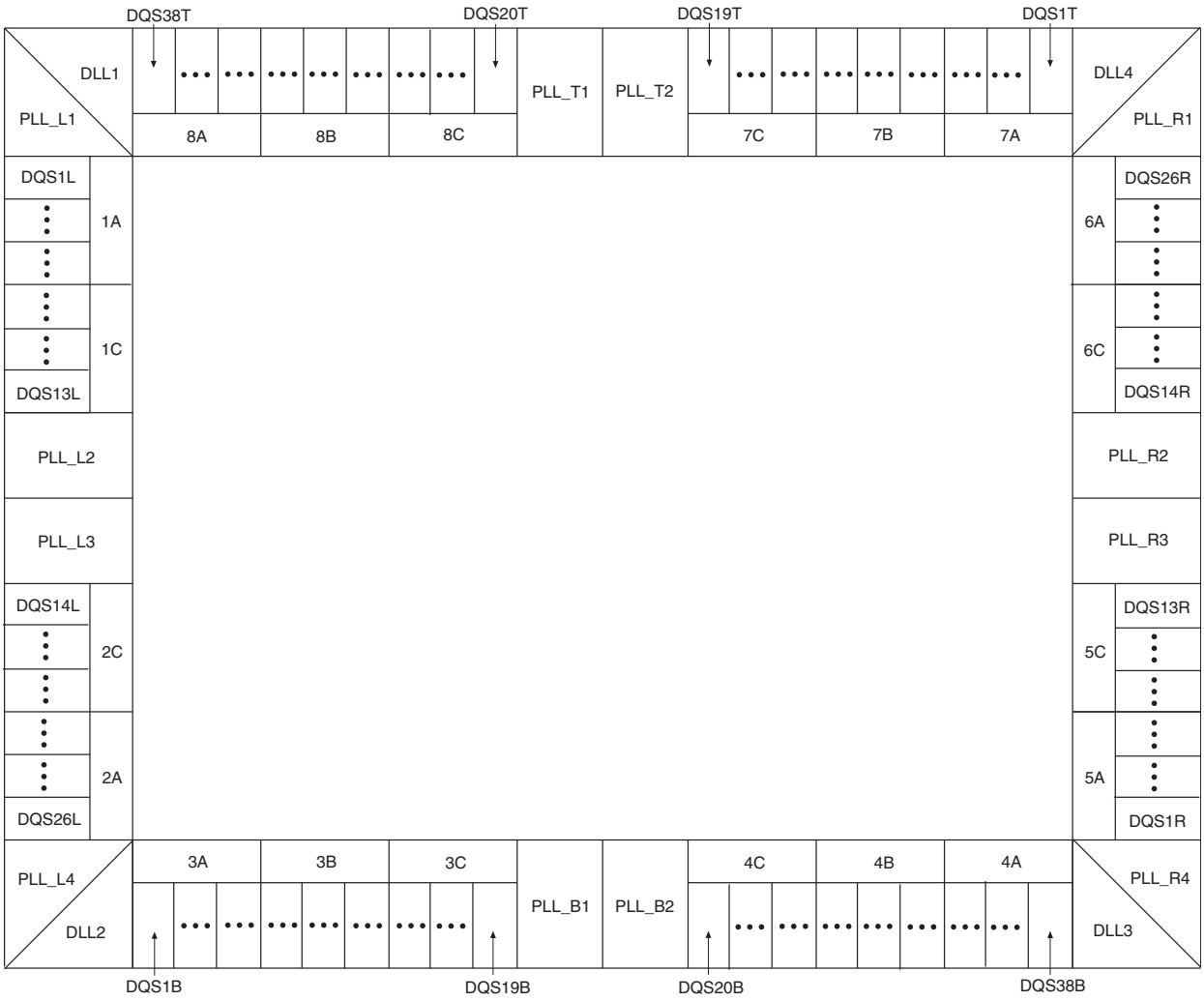
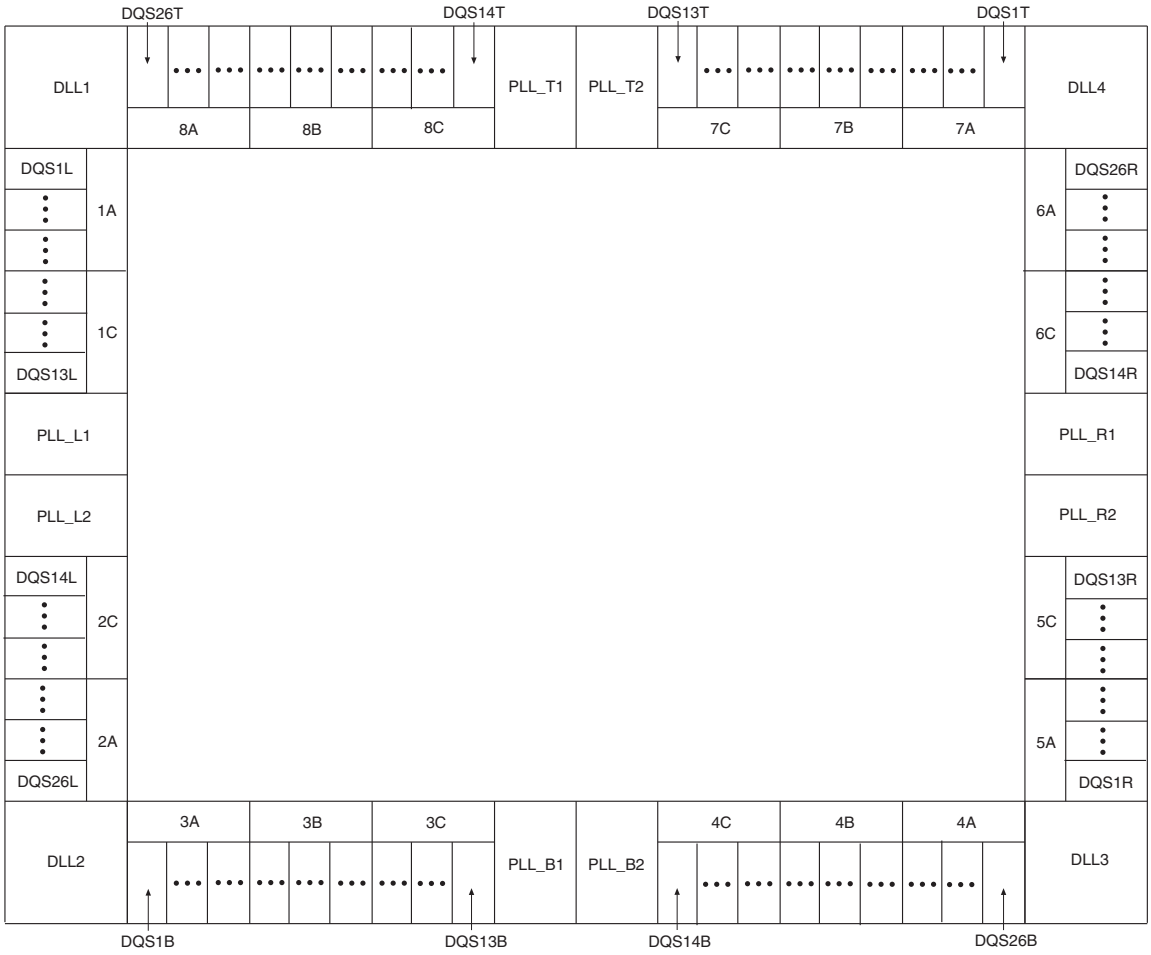


Figure 7–13. DQS Pins in HardCopy IV GX I/O Banks



The DQ pin numbering is based on ×4 mode. In ×4 mode, there are up to eight DQS/DQ groups per I/O bank. Each ×4 mode DQS/DQ group consists of a DQS pin, a DQSn pin, and four DQ pins. In ×8/×9 mode, the I/O bank combines two adjacent ×4 DQS/DQ groups; one pair of DQS and DQSn/CQn pins can drive all the DQ and parity pins in the new combined group that consists of up to 10 DQ pins (including parity or DM and QVLD pins) and a pair of DQS and DQSn/CQn pins. Similarly, in ×16/×18 mode, the I/O bank combines four adjacent ×4 DQS/DQ groups to create a group with a maximum of 19 DQ pins (including parity or DM and QVLD pins) and a pair of DQS and DQSn/CQn pins. In ×32/×36 mode, the I/O bank combines eight adjacent ×4 DQS/DQ groups together to create a group with a maximum of 37 DQ pins (including parity or DM and QVLD pins) and a pair of DQS and DQSn/CQn pins.

HardCopy IV modular I/O banks allow easy formation of the DQS/DQ groups. If all the pins in the I/O banks are user I/O pins and are not used for R_{UP}/R_{DN} OCT calibration or PLL clock output pins, you can divide the number of I/O pins in the bank by six to get the maximum possible number of ×4 groups. You can then divide that number by two, four, or eight to get the maximum possible number of ×8/×9, ×16/×18, or ×32/×36, respectively, as listed in Table 7–7. However, some of the pins in the I/O bank may be used for other functions.

Table 7–7. DQ/DQS Group in HardCopy IV Modular I/O Banks

Modular I/O Bank Size *(1)*	Maximum Possible Number of ×4 Groups	Maximum Possible Number of ×8/×9 Groups	Maximum Possible Number of ×16/×18 Groups	Maximum Possible Number of ×32/×36 Groups
24 pins	4 *(2)*	2	1	0
32 pins	5 *(3)*	2	1	0
40 pins	6	3	1	0
48 pins	8	4	2	1

Notes to Table 7–7:

(1) This I/O pin count does not include dedicated clock inputs or the dedicated corner PLL clock inputs.

(2) Some of the ×4 groups may use the R_{UP} and R_{DN} pins. You cannot use these groups if you use the HardCopy IV calibrated OCT feature.

(3) The actual maximum number of ×4 groups for an I/O bank with 32 pins is four in the HardCopy IV devices.

Optional Parity, DM, BWSn, ECC, and QVLD Pins

You can use any DQ pin from the same DQS/DQ group for data as parity pins in HardCopy IV devices. The HardCopy IV device family supports parity in the ×8/×9, ×16/×18, and ×32/×36 modes. There is one parity bit available per eight bits of data pins. Use any of the DQ (or D) pins in the same DQS/DQ group as data for parity because parity bits are treated, set, and generated similar to a DQ pin.

The DM pins are only required when writing to DDR3, DDR2, DDR SDRAM, and RLDRAM II devices. QDRII+ and QDRII SRAM devices use the BWSn signal to select which byte to write into the memory. A low on the DM or BWSn signals indicates the write is valid. If the DM or BWSn signal is high, the memory masks the DQ signals. If the system does not require write data masking, connect the memory DM pins low to indicate every write data is valid. You can use any of the DQ pins in the same DQS/DQ group as write data for the DM or BWSn signals.

Each group of DQS and DQ signals in DDR3, DDR2, and DDR SDRAM devices requires a DM pin. There is one DM pin per RLDRAM II device and one BWSn pin per nine bits of data in ×9, ×18, and ×36 QDRII+/QDRII SRAM. The ×8 QDRII SRAM device has two BWSn pins per eight data bits, which are referred to as the NWSn pins. Generate the DM or BWSn signals using DQ pins and configure the signals similarly to the DQ (or D) output signals. HardCopy IV devices do not support the DM signal in ×4 DDR3 SDRAM or in ×4 DDR2 SDRAM interfaces with differential DQS signaling.

Some DDR3, DDR2, and DDR SDRAM devices or modules support error correction coding (ECC), which is a method of detecting and automatically correcting errors in data transmission. In 72-bit DDR3, DDR2, or DDR SDRAM interfaces, the typical eight ECC pins are used in addition to the 64 data pins. Connect the DDR3, DDR2, and DDR SDRAM ECC pins to a HardCopy IV device DQS/DQ group. These signals are also generated similar to DQ pins. The memory controller requires encoding and decoding logic for the ECC data. You can also use the extra byte of data for other error checking methods.

QVLD pins are used in RLDRAM II and QDRII+ SRAM interfaces to indicate the read data availability. There is one QVLD pin per memory device. A high on the QVLD pin indicates that the memory is outputting the data requested. Similar to DQ inputs, this signal is edge-aligned with the read clock signals (CQ/CQn in QDRII+/QDRII SRAM and QK/QK# in RLDRAM II) and is sent half a clock cycle before data starts from the memory. The QVLD pin is not used in the ALTMEMPHY solution for QDRII+ SRAM.

For more information about the parity, ECC, and QVLD pins, and when these pins are treated as DQ pins, refer to "Data and Data Clock/Strobe Pins" on page 7–6.

Address and Control/Command Pins

Address and control/command signals are typically sent at single data rate. The only exception is in QDRII SRAM burst-of-two devices, in which case the read address must be captured on the rising edge of the clock and the write address must be captured on the falling edge of the clock by the memory. There is no special circuitry required for the address and control/command pins. You can use any of the user I/O pins in the same I/O bank as the data pins.

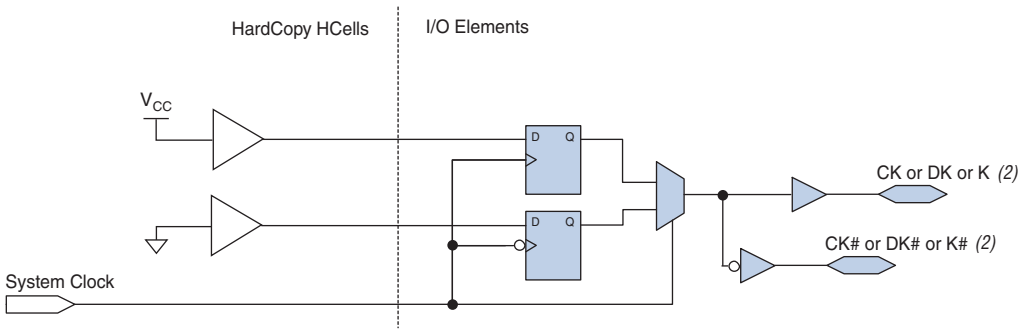
Memory Clock Pins

In addition to DQS (and CQn) signals to capture data, DDR3, DDR2, DDR SDRAM, and RLDRAM II use an extra pair of clocks, called CK and CK# signals, to capture the address and control/command signals. The CK and CK# signals must be generated to mimic the write data-strobe using HardCopy IV DDR I/O registers (DDIOs) to ensure that the timing relationships between the CK, CK#, and DQS signals (tDQSS in DDR3, DDR2, and DDR SDRAM or tCKDK in RLDRAM II) are met. QDRII+ and QDRII SRAM devices use the same clock (K/K#) to capture the data, address, and control/command signals.

Memory clock pins in HardCopy IV devices are generated using a DDIO register going to differential output pins, marked in the pin table with DIFFOUT, DIFFIO_TX, and DIFFIO_RX prefixes. For more information about which pins to use for memory clock pins, refer to Table 7–4 on page 7–8.

Figure 7–14 shows memory clock generation for HardCopy IV devices.

Figure 7–14. Memory Clock Generation Block Diagram *(Note 1)*



Notes to Figure 7–14:

(1) For the pin location requirements for these pins, refer to Table 7–3 on page 7–7.

(2) The mem_clk[0] and mem_clk_n[0] pins for DDR3, DDR2, and DDR SDRAM interfaces use the I/O input buffer for feedback. For memory interfaces using a differential DQS input, the input feedback buffer is configured as differential input; for memory interfaces using a single-ended DQS input, the input buffer is configured as a single-ended input. Using a single-ended input feedback buffer requires that V_{REF} is provided to that I/O bank's VREF pins.

HardCopy IV External Memory Interface Features

HardCopy IV devices are rich with features that allow robust high-performance external memory interfacing. The ALTMEMPHY megafunction allows you to set these external memory interface features and helps set up the physical interface (PHY) best suited for your system.

This section describes each HardCopy IV device feature that is used in external memory interfaces from the DQS phase-shift circuitry, DQS logic block, leveling multiplexers, dynamic OCT control block, IOE registers, IOE features, and the PLL.

When using the Altera® memory controller MegaCore® functions, the PHY is instantiated for you.

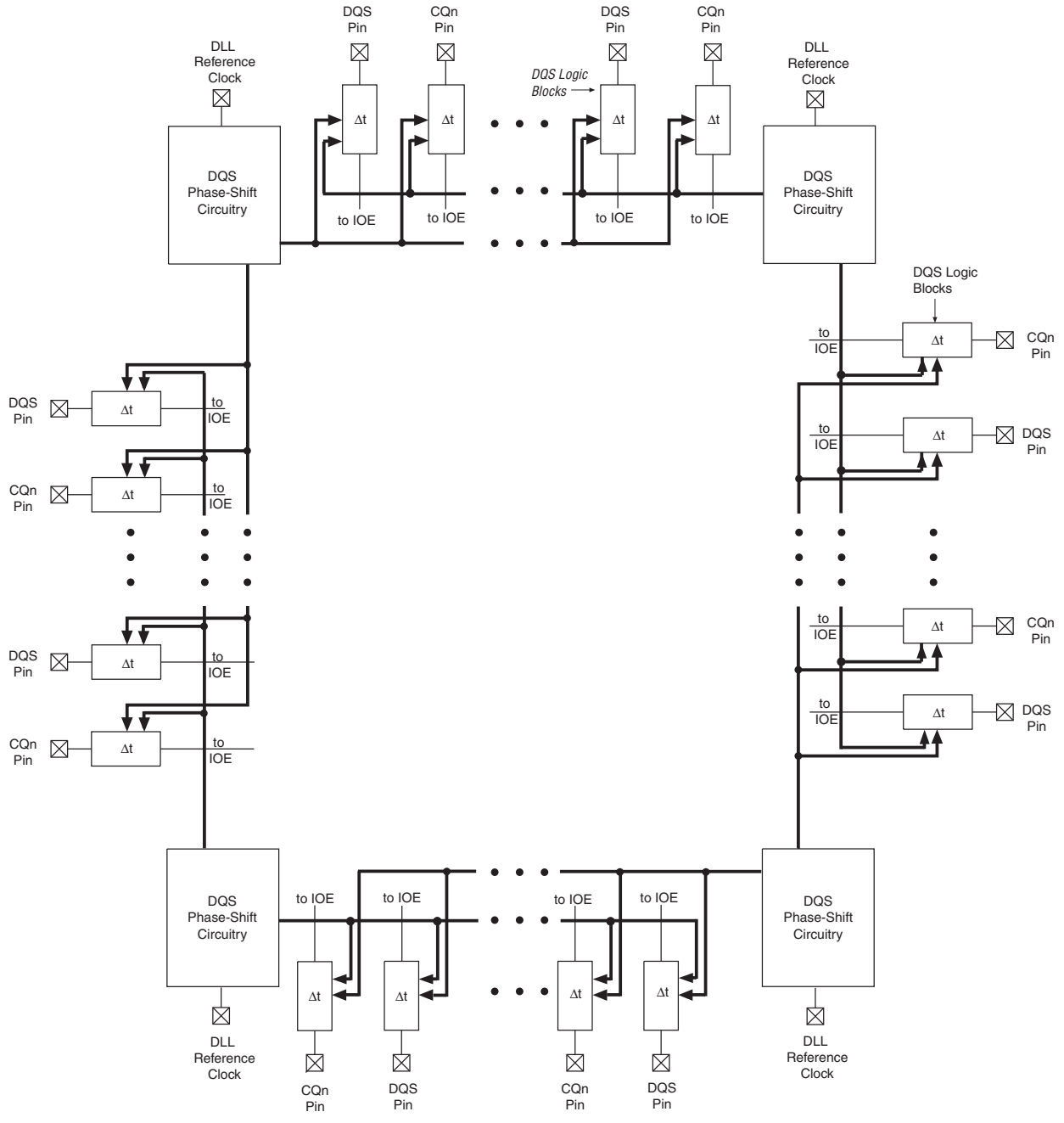
The ALTMEMPHY megafunction and the Altera memory controller MegaCore functions can run at half the frequency of the I/O interface of the memory devices to allow better timing management in high-speed memory interfaces. HardCopy IV devices have built-in registers to convert data from full-rate (I/O frequency) to half-rate (controller frequency) and vice versa. You can bypass these registers if your memory controller is not running at half the rate of the I/O frequency.

For more information about the ALTMEMPHY megafunction, refer to the *External DDR Memory PHY Interface Megafunction User Guide (ALTMEMPHY)*.

DQS Phase-Shift Circuitry

The HardCopy IV phase-shift circuitry provides phase shift to the DQS and CQn pins on read transactions when the DQS and CQn pins are acting as input clocks or strobes to the HardCopy IV device. The DQS phase-shift circuitry consists of DLLs that are shared between multiple DQS pins and the phase-offset module to further fine-tune the DQS phase shift for different sides of the device. Figure 7–15 shows how the DQS phase-shift circuitry is connected to the DQS and CQn pins in the device.

Figure 7–15. DQS and CQn Pins and DQS Phase-Shift Circuitry



The DQS phase-shift circuitry is connected to the DQS logic blocks that control each DQS or CQn pin. The DQS logic blocks allow the DQS delay settings to be updated concurrently at every DQS or CQn pin.

DLL

DQS phase-shift circuitry uses a DLL to dynamically measure the clock period required by the DQS/CQn pin. The DLL, in turn, uses a frequency reference to generate dynamically controlled signals for the delay chains in each of the DQS and CQn pins, allowing it to compensate for PVT variations. The DQS delay settings are Gray-coded to reduce jitter when the DLL updates the settings. The phase-shift circuitry requires a maximum of 1,280 clock cycles to calculate the correct input clock period. Data should not be sent during these clock cycles because there is no guarantee that it will be captured properly. Because the settings from the DLL may not be stable until this lock period has elapsed, anything using these settings (including the leveling delay system) may be unstable during this period.

You can still use the DQS phase-shift circuitry for any memory interfaces that are less than 100 MHz. The DQS signal is shifted by 2.5 ns. Even if the DQS signal is not shifted exactly to the middle of the DQ valid window, the IOE must be able to capture the data in low frequency applications where a large amount of timing margin is available.

There are four DLLs in a HardCopy IV device, located in each corner of the device. These four DLLs can support a maximum of four unique frequencies, with each DLL running at one frequency. Each DLL can have two outputs with different phase offsets, allowing one HardCopy IV device to have eight different DLL phase shift settings. Figure 7–16 shows the DLL and I/O bank locations in HardCopy IV E devices, from a package-bottom view. Figure 7–17 shows the DLL and I/O bank locations in HardCopy IV GX devices.

Altera recommends enabling the PLL reconfiguration feature and the DLL phase offset feature (DLL reconfiguration) for HardCopy IV devices. Because HardCopy IV devices are mask programmed, they cannot be changed after the silicon is fabricated. By implementing these two features, you can perform timing adjustments to improve or resolve timing issues after the silicon is fabricated.

Figure 7–16. HardCopy IV E DLL and I/O Bank Locations (Package-Bottom View)

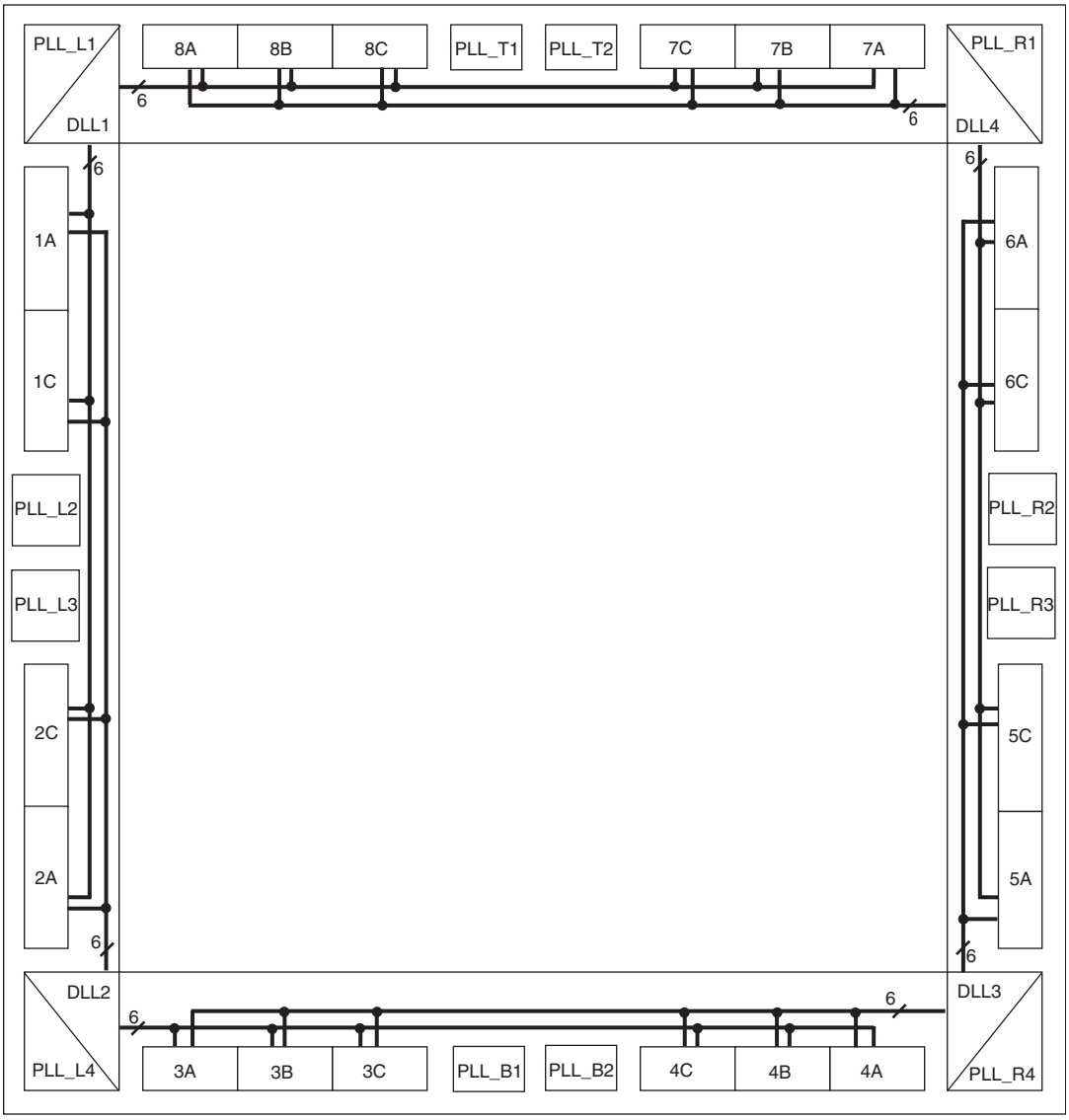
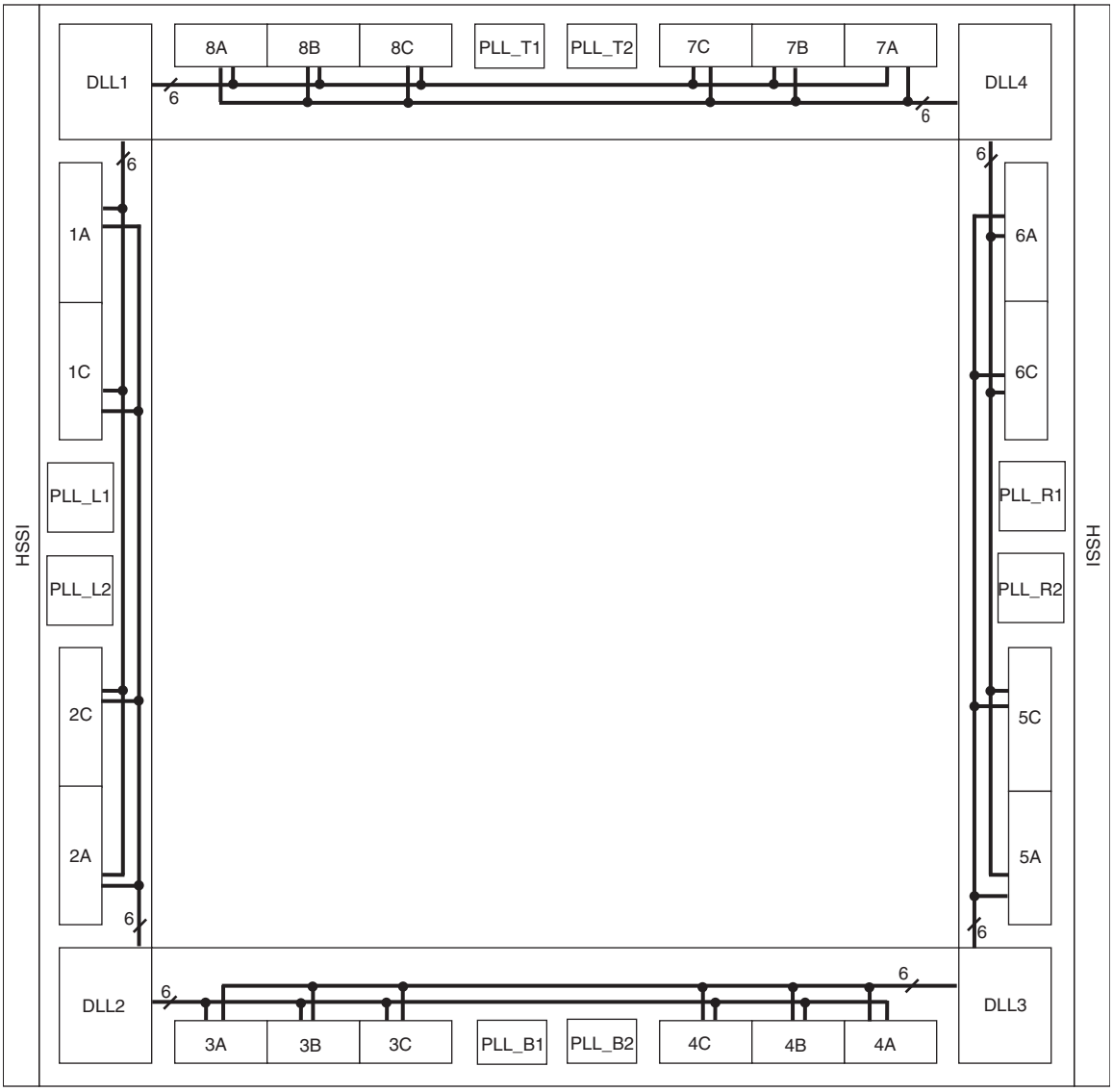


Figure 7–17. HardCopy IV GX DLL and I/O Bank Locations (Package-Bottom View)



The DLL can access the two adjacent sides from its location within the device. For example, DLL1 on the top left of the device can access the top side (I/O banks 7A, 7B, 7C, 8A, 8B, and 8C) and the left side of the device (I/O banks 1A, 1C, 2A, and 2C). This means that each I/O bank is accessible by two DLLs, giving more flexibility to create multiple frequencies and multiple-types interfaces. For example, you can design an interface spanning one side of the device or two sides adjacent to the DLL. The DLL outputs the same DQS delay settings for both sides of the device adjacent to the DLL.

Interfaces that span across two sides of the device are not recommended for high-performance memory interface applications.

Each bank can use settings from either or both DLLs of the adjacent bank. For example, DQS1L can use phase-shift settings from DLL1, and DQS2L can use phase-shift settings from DLL2. Table 7–8 lists the DLL location and supported I/O banks for HardCopy IV devices.

You can only have one memory interface in I/O banks with the same I/O bank number (such as I/O banks 1A and 1C) when the leveling delay chains are used, because there is only one leveling delay chain shared by these I/O banks.

Table 7–8. DLL Location and Supported I/O Banks

DLL	Location	Accessible I/O Banks
DLL1	Top left corner	1A, 1C, 2A, 2C, 7A, 7B, 7C, 8A, 8B, 8C
DLL2	Bottom left corner	1A, 1C, 2A, 2C, 3A, 3B, 3C, 4A, 4B, 4C
DLL3	Bottom right corner	3A, 3B, 3C, 4A, 4B, 4C, 5A, 5C, 6A, 6C
DLL4	Top right corner	5A, 5C, 6A, 6C, 7A, 7B, 7C, 8A, 8B, 8C

The reference clock for each DLL may come from the PLL output clocks or any of the two dedicated clock input pins located in either side of the DLL. Table 7–9 and Table 7–10 show the available DLL reference clock input resources for HardCopy IV E devices.

Table 7–9. DLL Reference Clock Input for HC4E25W and HC4E25F Devices

DLL	CLKIN (Top/Bottom)	CLKIN (Left/Right)	PLL (Top/Bottom)	PLL (Left/Right)
DLL1	CLK12P, CLK13P, CLK14P, CLK15P	CLK0P, CLK1P, CLK2P, CLK3P	PLL_T1	PLL_L2
DLL2	CLK4P, CLK5P, CLK6P, CLK7P	CLK0P, CLK1P, CLK2P, CLK3P	PLL_B1	PLL_L2
DLL3	CLK4P, CLK5P, CLK6P, CLK7P	CLK8P, CLK9P, CLK10P, CLK11P	PLL_B1	PLL_R2
DLL4	CLK12P, CLK13P, CLK14P, CLK15P	CLK8P, CLK9P, CLK10P, CLK11P	PLL_T1	PLL_R2

Table 7–10. DLL Reference Clock Input for HC4E35L and HC4E35F Devices

DLL	CLKIN (Top/Bottom)	CLKIN (Left/Right)	PLL (Top/Bottom)	PLL (Left/Right)
DLL1	CLK12P, CLK13P, CLK14P, CLK15P	CLK0P, CLK1P, CLK2P, CLK3P	PLL_T1	PLL_L1, PLL_L2
DLL2	CLK4P, CLK5P, CLK6P, CLK7P	CLK0P, CLK1P, CLK2P, CLK3P	PLL_B1	PLL_L3, PLL_L4
DLL3	CLK4P, CLK5P, CLK6P, CLK7P	CLK8P, CLK9P, CLK10P, CLK11P	PLL_B2	PLL_R3, PLL_R4
DLL4	CLK12P, CLK13P, CLK14P, CLK15P	CLK8P, CLK9P, CLK10P, CLK11P	PLL_T2	PLL_R1, PLL_R2

Table 7–11 through Table 7–13 list the available DLL reference clock input resources for HardCopy IV GX devices.

Table 7–11. DLL Reference Clock Input for HC4GX15LA and HC4GX25L Devices (Part 1 of 2)

DLL	CLKIN (Top/Bottom)	CLKIN (Left/Right)	PLL (Top/Bottom)	PLL (Left/Right)
DLL1	CLK12P, CLK13P, CLK14P, CLK15P	CLK0P, CLK1P, CLK2P, CLK3P *(1)*	PLL_T1	PLL_L1 *(2)*
DLL2	CLK4P, CLK5P, CLK6P, CLK7P	CLK0P, CLK1P, CLK2P, CLK3P *(1)*	PLL_B1	—
DLL3	CLK4P, CLK5P, CLK6P, CLK7P	—	PLL_B1	—

Table 7–11. DLL Reference Clock Input for HC4GX15LA and HC4GX25L Devices (Part 2 of 2)

DLL	CLKIN (Top/Bottom)	CLKIN (Left/Right)	PLL (Top/Bottom)	PLL (Left/Right)
DLL4	CLK12P, CLK13P, CLK14P, CLK15P	—	PLL_T1	—

Notes to Table 7–11:

(1) Dedicated clock inputs for DLL1 and DLL2 are not supported in the HC4GX15L devices.

(2) PLL_L1 is not supported in the HC4GX15L devices.

Table 7–12. DLL Reference Clock Input for HC4GX25L and HC4GX25F Devices

DLL	CLKIN (Top/Bottom)	CLKIN (Left/Right)	PLL (Top/Bottom)	PLL (Left/Right)
DLL1	CLK12P, CLK13P, CLK14P, CLK15P	CLK0P, CLK1P	PLL_T1	PLL_L1
DLL2	CLK4P, CLK5P, CLK6P, CLK7P	CLK0P, CLK1P	PLL_B1	—
DLL3	CLK4P, CLK5P, CLK6P, CLK7P	CLK11P, CLK10P	PLL_B2	—
DLL4	CLK12P, CLK13P, CLK14P, CLK15P	CLK11P, CLK10P	PLL_T2	PLL_R1

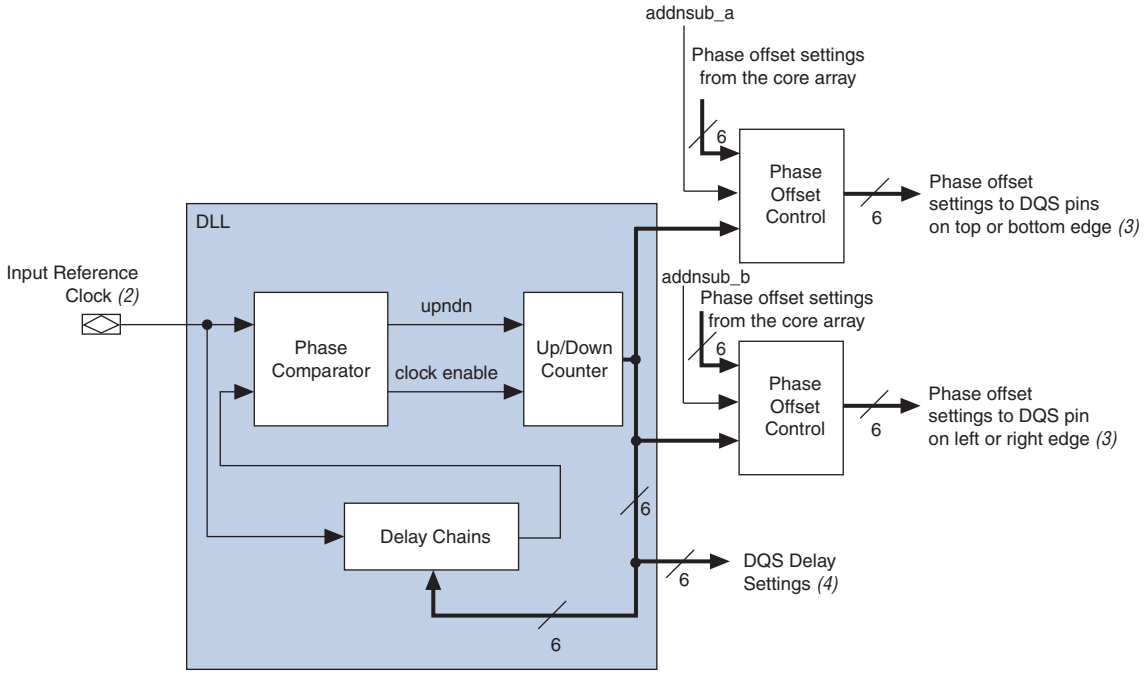
Table 7–13. DLL Reference Clock Input for HC4GX35L and HC4GX35F Devices

DLL	CLKIN (Top/Bottom)	CLKIN (Left/Right)	PLL (Top/Bottom)	PLL (Left/Right)
DLL1	CLK12P, CLK13P, CLK14P, CLK15P	CLK0P, CLK1P, CLK2P, CLK3P	PLL_T1	PLL_L1
DLL2	CLK4P, CLK5P, CLK6P, CLK7P	CLK0P, CLK1P, CLK2P, CLK3P	PLL_B1	PLL_L2
DLL3	CLK4P, CLK5P, CLK6P, CLK7P	CLK11P, CLK10P, CLK8P, CLK9P	PLL_B2	PLL_R2
DLL4	CLK12P, CLK13P, CLK14P, CLK15P	CLK11P, CLK10P, CLK8P, CLK9P	PLL_T2	PLL_R1

When you have a dedicated PLL that only generates the DLL input reference clock, set the PLL mode to **No Compensation**; otherwise, the Quartus II software changes it automatically. Because the PLL does not use any other outputs, it does not have to compensate for any clock paths.

Figure 7–18 shows a block diagram of the DLL. The input reference clock goes into the DLL to a chain of up to 16 delay elements. The phase comparator compares the signal coming out of the end of the delay chain block to the input reference clock. The phase comparator then issues the `upndn` signal to the Gray-code counter. This signal increments or decrements a six-bit delay setting (DQS delay settings) that increases or decreases the delay through the delay element chain to bring the input reference clock and the signals coming out of the delay element chain in phase.

Figure 7–18. Simplified Diagram of the DQS Phase Shift Circuitry *(Note 1)*



Note to Figure 7–18:

(1) All features of the DQS phase-shift circuitry are accessible from the ALTMEMPHY MegaWizard™ Plug-In Manager in the Quartus II software.

(2) For exact PLL and input clock pin information, the input reference clock for the DQS phase-shift circuitry can come from a PLL output clock or an input clock pin. Refer to Table 7–9 and Table 7–13.

(3) Phase offset settings can go only to the DQS logic blocks.

(4) DQS delay settings can go to the core array, the DQS logic block, and the leveling circuitry.

The DLL can be reset from either the core array or a user I/O pin. Each time the DLL is reset, you must wait for 1,280 clock cycles before you can capture the data properly.

Depending on the DLL frequency mode, the DLL can shift the incoming DQS signals by 0°, 22.5°, 30°, 36°, 45°, 60°, 67.5°, 72°, 90°, 108°, 120°, 135°, 144°, or 180°. The shifted DQS signal is then used as the clock for the DQ IOE input registers.

All DQS and CQn pins referenced to the same DLL can have their input signal phase shifted by a different degree amount, but all must be referenced at one particular frequency. For example, you can have a 90° phase shift on `DQS1T` and a 60° phase shift on `DQS2T` referenced from a 200-MHz clock. However, not all phase-shift combinations are supported. The phase shifts on the DQS pins referenced by the same DLL must all be a multiple of 22.5° (up to 90°), a multiple of 30° (up to 120°), a multiple of 36° (up to 144°), or a multiple of 45° (up to 180°).

There are seven different frequency modes for the HardCopy IV DLL, as shown in Table 7–14. Each frequency mode provides different phase shift selections. In frequency modes 0, 1, 2, and 3, the 6-bit DQS delay settings vary with PVT to implement the phase-shift delay. In frequency modes 4, 5, and 6, only 5 bits of the DQS delay settings vary to implement the phase-shift delay; the most significant bit of the DQS delay setting is set to 0.

Table 7–14. HardCopy IV DLL Frequency Modes

Frequency Mode	DQS Delay Setting Bus Width	Available Phase Shift	Number of Delay Chains
0	6 bits	22.5°, 45°, 67.5°, 90°	16
1	6 bits	30°, 60°, 90°, 120°	12
2	6 bits	36°, 72°, 108°, 144°	10
3	6 bits	45°, 90°, 135°, 180°	8
4	5 bits	30°, 60°, 90°, 120°	12
5	5 bits	36° 72°, 108°, 144°	10
6	5 bits	45°, 90°, 135°, 180°	8

For the 0° shift, the DQS signal bypasses both the DLL and the DQS logic blocks. The Quartus II software automatically sets DQ input delay chains so that the skew between the DQ and DQS pin at the DQ IOE registers is negligible when the 0° shift is implemented. You can feed the DQS delay settings to the DQS logic block and the core array.

The shifted DQS signal goes to the DQS bus to clock the IOE input registers of the DQ pins. The signal can also go into the core array for resynchronization if you are not using the IOE resynchronization registers. The shifted CQn signal can only go to the negative-edge input register in the DQ IOE and is only used for QDRII+ and QDRII SRAM interfaces.

Phase Offset Control

Each DLL has two phase-offset modules and can provide two separate DQS delay settings with independent offset, one for the top and bottom I/O banks and one for the left and right I/O banks, so you can fine-tune the DQS phase shift settings between two different sides of the device. Even though you have an independent phase offset control, the frequency of the interface using the same DLL must be the same. You should use the phase offset control module for making small shifts to the input signal and use the DQS phase-shift circuitry for larger signal shifts. For example, if the DLL only offers a multiple of a 30° phase shift, but your interface requires a 67.5° phase shift on the DQS signal, you can use two delay chains in the DQS logic blocks to give you a 60° phase shift and use the phase offset control feature to implement the extra 7.5° phase shift.

You can use either a static phase offset or a dynamic phase offset to implement the additional phase shift. The available additional phase shift is implemented in 2's-complement in Gray-code between settings –64 to +63 for frequency modes 0, 1, 2, and 3, and between settings –32 to +31 for frequency modes 4, 5, and 6. The DQS phase shift is the sum of the DLL delay settings and the user selected phase offset settings, which reaches a maximum at setting 64 for mode frequency modes 0, 1, 2 and 3, and a maximum at setting 32 for frequency modes 4, 5, and 6. The actual physical offset setting range is 64 or 32 subtracted by the DQS delay settings from the DLL.

You must monitor the DQS delay settings to determine how many offsets you can add and subtract in the system.

The DQS delay settings output by the DLL are also Gray-coded.

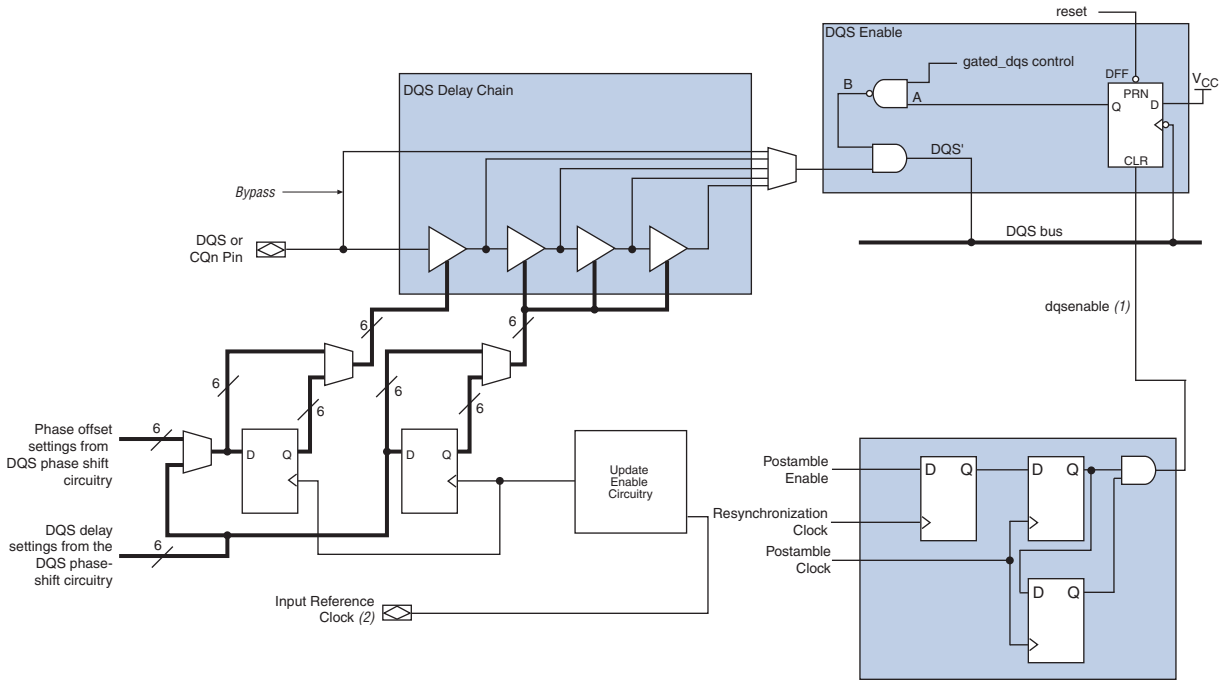
For example, if the DLL determines that a DQS delay setting of 28 is required to achieve a 30° phase shift in DLL frequency mode 1, you can subtract up to 28 phase offset settings and add up to 35 phase offset settings to achieve the optimal delay that you need. However, if the same DQS delay setting of 28 is required to achieve a 30° phase shift in DLL frequency mode 4, you can still subtract up to 28 phase offset settings, but you can only add up to 3 phase offset settings before the DQS delay settings reach their maximum settings. This is because DLL frequency mode 4 only uses 5-bit DLL delay settings.

If you use the static phase offset, specify the phase-offset amount in the ALTMEMPHY megafunction as a positive number for addition or a negative number for subtraction. You can also have a dynamic phase offset that is always added to, subtracted from, or both added to and subtracted from the DLL phase shift. When you always add or subtract, you can dynamically input the phase offset amount into the `dll_offset[5..0]` port. When you want to both add and subtract dynamically, you control the `addnsub` signal in addition to the `dll_offset[5..0]` signals.

DQS Logic Block

Each DQS and CQn pin is connected to a separate DQS logic block, which consists of the DQS delay chains, the update enable circuitry, and the DQS postamble circuitry as shown in Figure 7–19.

Figure 7–19. HardCopy IV DQS Logic Block



Notes to Figure 7–19:

(1) The `dqsenable` signal can also come from the HardCopy IV core fabric.

(2) The input reference clock for the DQS phase-shift circuitry can come from a PLL output clock or an input clock pin. Refer to Table 7–9 and Table 7–13 for the exact PLL and input clock pin.

DQS Delay Chain

The DQS delay chains consist of a set of variable delay elements to allow the input DQS and CQn signals to be shifted by the amount specified by the DQS phase-shift circuitry or the core array. There are four delay elements in the DQS delay chain; the first delay chain closest to the DQS pin can be shifted either by the DQS delay settings or by the sum of the DQS delay setting and the phase-offset setting. The number of delay chains required is transparent to you because the ALTMEMPHY megafunction automatically sets it when you choose the operating frequency. The DQS delay settings can come from the DQS phase-shift circuitry on either end of the I/O banks or from the core array.

The delay elements in the DQS logic block have the same characteristics as the delay elements in the DLL. When the DLL does not control the DQS delay chains, you can input your own Gray-coded 6-bit or 5-bit settings using the `dqs_delayctrlin[5..0]` signals available in the ALTMEMPHY megafunction. These settings control 1, 2, 3, or all 4 delay elements in the DQS delay chains. The ALTMEMPHY megafunction can also dynamically choose the number of DQS delay chains required for the system. The amount of delay is equal to the sum of the delay element's intrinsic delay and the product of the number of delay steps and the value of the delay steps.

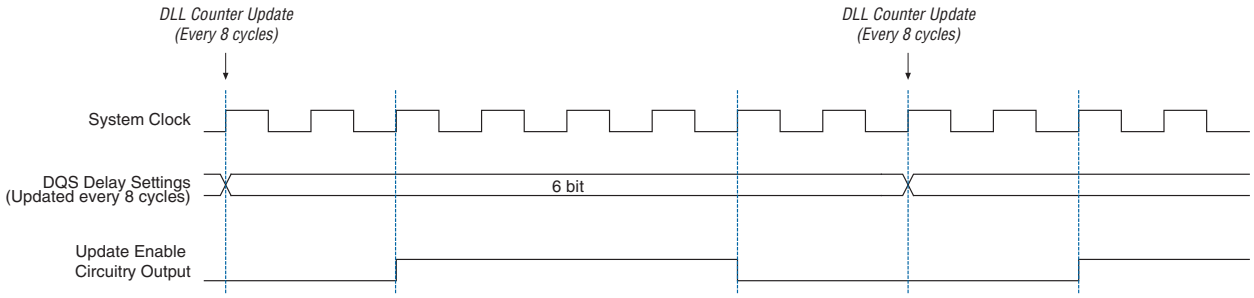
You can also bypass the DQS delay chain to achieve a 0° phase shift.

For more information about the ALTMEMPHY megafunction, refer to the *External DDR Memory PHY Interface Megafunction User Guide (ALTMEMPHY)*.

Update Enable Circuitry

Both the DQS delay settings and the phase-offset settings pass through a register before entering the DQS delay chains. The registers are controlled by the update enable circuitry to allow enough time for changes in the DQS delay setting bits to arrive at all the delay elements. This allows them to be adjusted at the same time. The update enable circuitry enables the registers to allow enough time for the DQS delay settings to travel from the DQS phase-shift circuitry or core logic to all the DQS logic blocks before the next change. It uses the input reference clock or a user clock from the core to generate the update enable output. The ALTMEMPHY megafunction uses this circuit by default. Figure 7–20 shows an example waveform of the update enable circuitry output.

Figure 7–20. DQS Update Enable Waveform

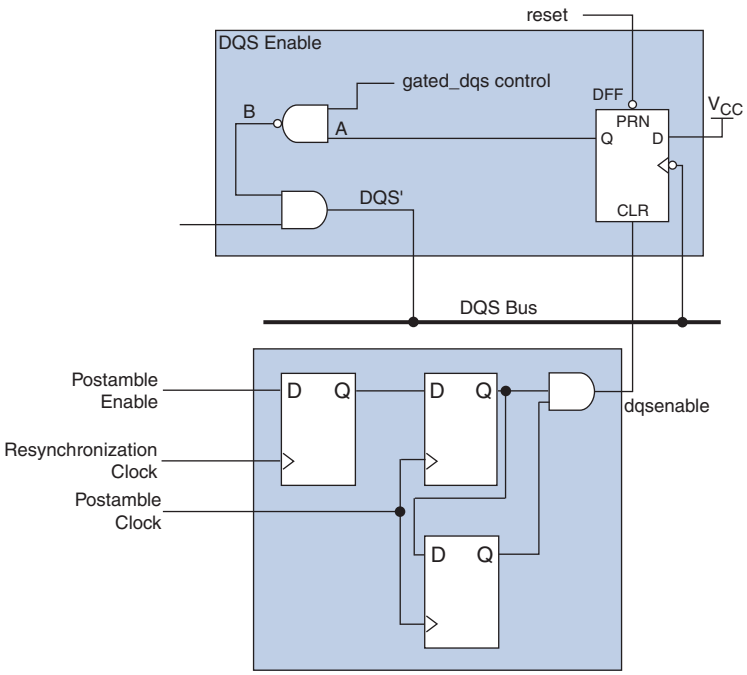


DQS Postamble Circuitry

For external memory interfaces that use a bidirectional read strobe such as DDR3, DDR2, and DDR SDRAM, the DQS signal is low before going to or coming from a high-impedance state. The state where DQS is low, just after a high-impedance state, is called the preamble state; the state where DQS is low, just before it returns to a high-impedance state, is called the postamble state. There are preamble and postamble specifications for both read and write operations in DDR3, DDR2, and DDR SDRAM.

The DQS postamble circuitry, shown in Figure 7–21, ensures that data is not lost when there is noise on the DQS line at the end of a read postamble time. HardCopy IV devices have a dedicated postamble register that can be controlled to ground the shifted DQS signal used to clock the DQ input registers at the end of a read operation. This ensures that any glitches on the DQS input signals at the end of the read postamble time do not affect the DQ IOE registers.

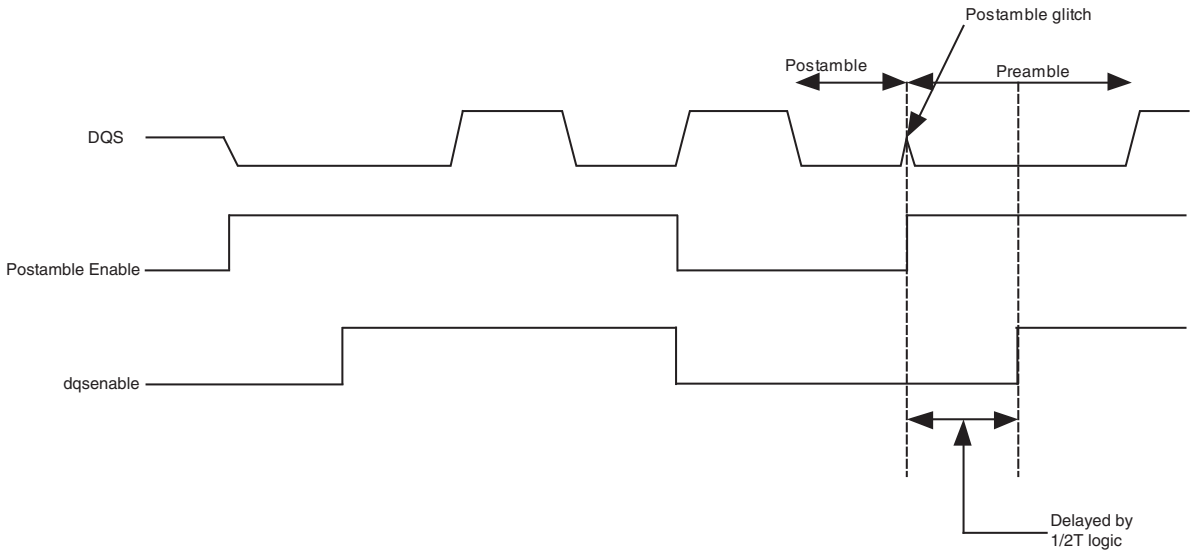
Figure 7–21. HardCopy IV DQS Postamble Circuitry



In addition to the dedicated postamble register, HardCopy IV devices also have an HDR block inside the postamble enable circuitry. These registers are used if the controller is running at half the frequency of the I/Os.

Using the HDR block as the first stage capture register in the postamble enable circuitry block in Figure 7–21 is optional. The HDR block is clocked by the half-rate resynchronization clock, which is the output of the I/O clock divider circuit (shown in Figure 7–27). The AND gate after the postamble register outputs is used to avoid postamble glitches from a previous read burst on a non-consecutive read burst. This scheme allows a half-a-clock cycle latency for `dqsenable` assertion and zero latency for `dqsenable` deassertion, as shown in Figure 7–22.

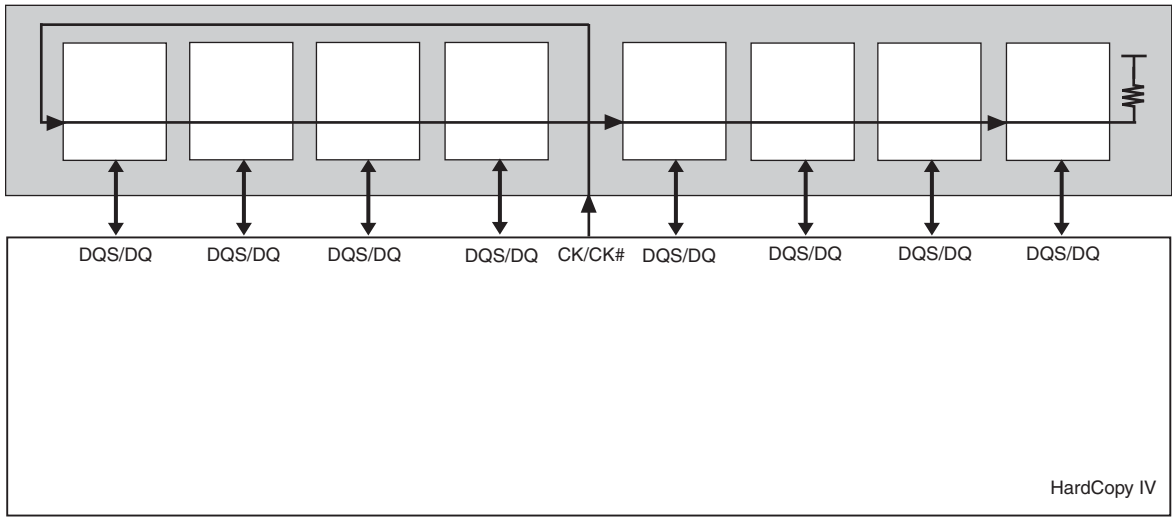
Figure 7–22. Avoiding Glitch on a Non-Consecutive Read Burst Waveform



Leveling Circuitry

DDR3 SDRAM unbuffered modules use a fly-by clock distribution topology for better signal integrity. This means that the CK/CK# signals arrive at each DDR3 SDRAM device in the module at different times. The difference in arrival time between the first DDR3 SDRAM device and the last device on the module can be as long as 1.6 ns. Figure 7–23 shows the clock topology in DDR3 SDRAM unbuffered modules.

Figure 7–23. DDR3 SDRAM Unbuffered Module Clock Topology

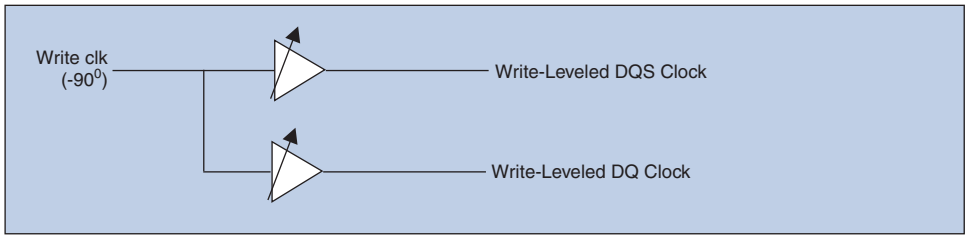


Because the data and read strobe signals are still point-to-point, special consideration must be taken to ensure that the timing relationship between the CK/CK# and DQS signals (tDQSS) during a write is met at every device on the modules. Furthermore, read data returning to the HardCopy IV ASIC from the memory is also staggered in a similar way. HardCopy IV ASICs have leveling circuitry to compensate for the different CK/CK# arrival time at each device in the memory module.

There is one group of leveling circuitry per I/O bank, with the same I/O number (for example, there is one leveling circuitry shared between I/O bank 1A and 1C) located in the middle of the I/O bank. These delay chains are PVT-compensated by the same DQS delay settings as the DLL and DQS delay chains. The generated clock phases are distributed to every DQS logic block that is available in the I/O bank. The delay chain taps, then feeds a multiplexer controlled by the ALTMEMPHY megafunction to select which clock phases are to be used for that ×4 or ×8 DQS group. Each group can use a different tap output from the read-leveling and write-leveling delay chains to compensate for the different CK/CK# delay going into each device on the module.

Figure 7–24 and Figure 7–25 show the HardCopy IV read-and-write leveling circuitry.

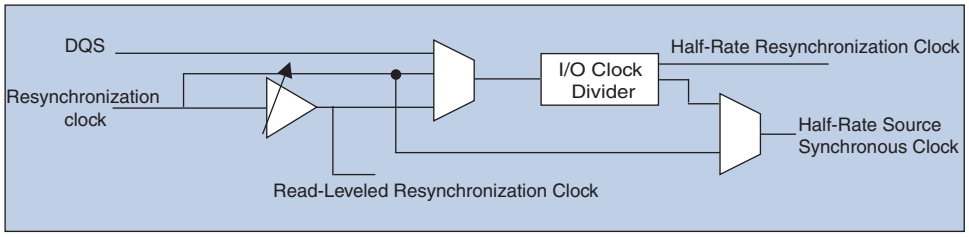
Figure 7–24. HardCopy IV Write-Leveling Delay Chains *(Note 1)*



Note to Figure 7–24:

(1) There is only one leveling delay chain per I/O bank with the same I/O number (for example, I/O banks 1A and 1C). You can only have one memory controller in these I/O banks when the leveling delay chains are used.

Figure 7–25. HardCopy IV Read-Leveling Delay Chains and Multiplexers *(Note 1)*



Note to Figure 7–25:

(1) There is only one leveling delay chain per I/O bank with the same I/O number (for example, I/O banks 1A and 1C). You can only have one memory controller in these I/O banks when the leveling delay chains are used.

The –90° write clock of the ALTMEMPHY megafunction feeds the write-leveling circuitry to produce the clock to generate the DQS and DQ signals. During initialization, the ALTMEMPHY megafunction picks the correct write-leveled clock for the DQS and DQ clocks for each DQS/DQ group after sweeping all the available clocks in the write calibration process. The DQ clock output is –90° phase-shifted compared to the DQS clock output.

Similarly, the resynchronization clock feeds the read-leveling circuitry to produce the optimal resynchronization and postamble clock for each DQS/DQ group in the calibration process. The resynchronization and postamble clocks can use different clock outputs from the leveling circuitry. The output from the read-leveling circuitry can also generate the half-rate resynchronization clock that goes to the core fabric.

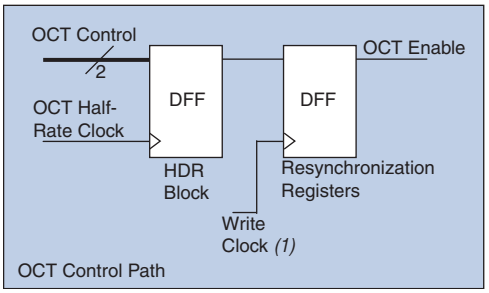
The ALTMEMPHY megafunction calibrates the alignment for read and write leveling dynamically during the initialization process. For more information about the ALTMEMPHY megafunction, refer to the *External DDR Memory PHY Interface Megafunction User Guide (ALTMEMPHY)*.

Dynamic On-Chip Termination Control

Figure 7–26 shows the dynamic OCT control block. The block includes all the registers required to dynamically turn OCT on during a read and turn OCT off during a write.

For more information about OCT, refer to "OCT" on page 7–41, or to the *HardCopy IV Device I/O Features* chapter.

Figure 7–26. HardCopy IV Dynamic OCT Control Block



Note to Figure 7–26:

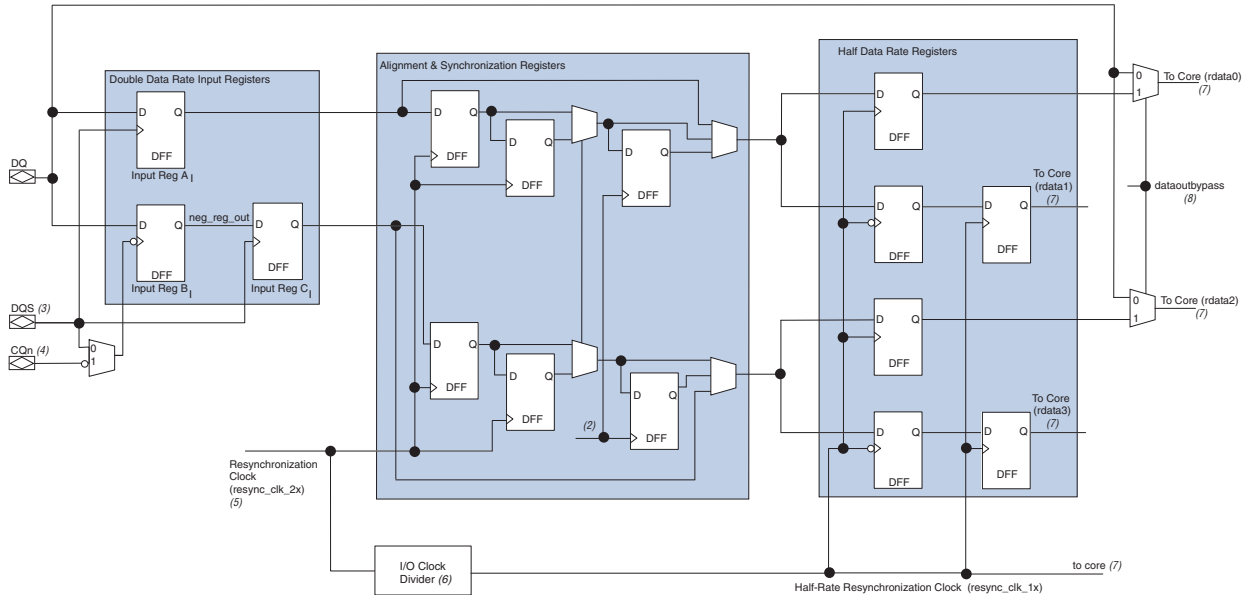
(1) The write clock comes from either the PLL or the write-leveling delay chain.

I/O Element Registers

The IOE registers have been expanded to allow source-synchronous systems to have faster register-to-register transfers and resynchronization. Both top, bottom, left, and right IOEs have the same capability, although left and right IOEs have extra features to support LVDS data transfer.

Figure 7–27 shows the registers available in the HardCopy IV input path. The input path consists of the DDR input registers, resynchronization registers, and HDR block. You can bypass each block of the input path.

Figure 7–27. HardCopy IV IOE Input Registers *(Note 1)*



Notes to Figure 7–27:

(1) You can bypass each register block in this path.
(2) This is the 0-phase resynchronization clock from the read-leveling delay chain.
(3) The input clock can be from the DQS logic block (whether the postamble circuitry is bypassed or not) or from a global clock line.
(4) This input clock comes from the CQn logic block.
(5) This resynchronization clock can come either from the PLL or from the read-leveling delay chain.
(6) The I/O clock divider resides adjacent to the DQS logic block. In addition to the PLL and read levelled resync clock, the I/O clock divider can also be fed by the DQS bus or CQn bus.
(7) The half-rate data and clock signals feed into a FIFO in the core.
(8) You can change the `dataoutbypass` signal dynamically after the device enters user mode.

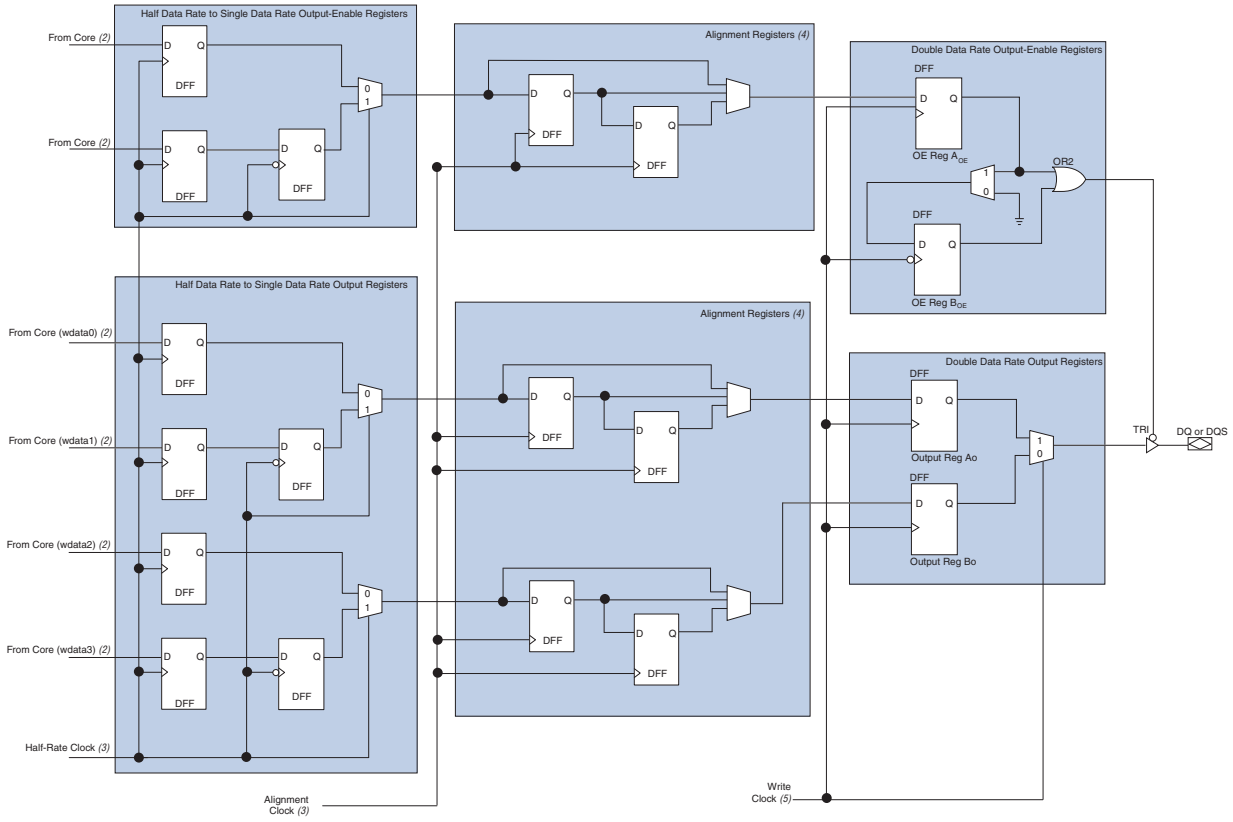
There are three registers in the DDR input registers block. Two registers capture data on the positive and negative edges of the clock, while the third register aligns the captured data. You can choose to have the same clock for the positive edge and negative edge registers, or two different clocks (DQS for positive-edge register, and CQn for negative-edge register). The third register that aligns the captured data uses the same clock as the positive-edge register.

The resynchronization registers consist of up to three levels of registers to resynchronize the data to the system clock domain. These registers are clocked by the resynchronization clock that is either generated by the PLL or the read-leveling delay chain. The outputs of the resynchronization registers can go straight to the core or to the HDR blocks, which are clocked by the divided-down resynchronization clock.

For more information about the read-leveling delay chain, refer to "Leveling Circuitry" on page 7–36.

Figure 7–28 shows the registers available in the HardCopy IV output and output-enable paths. The path is divided into the HDR block, resynchronization registers, and output/output-enable registers. The device can bypass each block of the output and output-enable path.

Figure 7–28. HardCopy IV IOE Output and Output-Enable Path Registers *(Note 1)*



Notes to Figure 7–28:

(1) You can bypass each register block of the output and output-enable paths.
(2) Data coming from the ASIC core are at half the frequency of the memory interface.
(3) Half-rate and alignment clocks come from the PLL.
(4) These registers are only used in DDR3 SDRAM interfaces.
(5) The write clock can come from either the PLL or from the write-leveling delay chain. The DQ write clock and DQS write clock have a 90° offset between them.

The output path is designed to route combinational or registered SDR outputs and full-rate or half-rate DDR outputs from the core. Half-rate data is converted to full-rate data using the HDR block, clocked by the half-rate clock from the PLL. The resynchronization registers are also clocked by the same 0° system clock, except in the DDR3 SDRAM interface. In DDR3 SDRAM interfaces, the leveling registers are clocked by the write-leveling clock.

For more information about the write leveling delay chain, refer to "Leveling Circuitry" on page 7–36.

The output-enable path has a structure similar to the output path. You can have a combinational or registered output in SDR applications and you can use half-rate or full-rate operation in DDR applications. You also have the resynchronization registers similar to the output path registers structure, ensuring that the output enable path goes through the same delay and latency as the output path.

IOE Features

This section describes how OCT, delay chains, output delay, slew rate control, and drive strength setting are useful in memory interfaces.

These IOE features are mask programmed and cannot be changed after the silicon is fabricated.

For more information about the features listed below, refer to the *HardCopy IV Device I/O Features* chapter.

OCT

HardCopy IV devices feature dynamic calibrated OCT, in which the series termination (OCT R_S) is turned on when driving signals and turned off when receiving signals, and the parallel termination (OCT R_T) is turned off when driving signals and turned on when receiving signals. This feature complements the DDR3/DDR2 SDRAM on-die termination (ODT), in which the memory termination is turned off when the memory is sending data and turned on when receiving data. You can use OCT for other memory interfaces to improve signal integrity.

You cannot use the drive strength and slew rate features when using OCT R_S.

To use the dynamic calibrated OCT feature, you must use the R_{UP} and R_{DN} pins to calibrate the OCT calibration block. You can use one OCT calibration block to calibrate one type of termination with the same V_{CCIO} on the entire device. There are up to eight OCT calibration blocks to allow for different types of terminations throughout the device. For more information, refer to "Dynamic On-Chip Termination Control" on page 7–38.

You have the option to use the OCT R_S feature with or without calibration. However, the OCT R_T feature is only available with calibration.

You can also use the R_{UP} and R_{DN} pins as DQ pins, so you cannot use the DQS/DQ groups where the R_{UP} and R_{DN} pins are located if you are planning to use dynamic calibrated OCT. The R_{UP} and R_{DN} pins are located in the first and last ×4 DQS/DQ group on each side of the device.

Use the OCT R_T or R_S setting for unidirectional read-and-write data and a dynamic OCT setting for bidirectional data signals.

IOE Delay Chains

You can use the delay chains in the HardCopy IV I/O registers as deskewing circuitry. Each pin can have a different input delay from the pin to the input register or a delay from the output register to the output pin to ensure that the bus has the same delay going into or out of the device. This feature helps read and write time margins as it minimizes the uncertainties between signals in the bus.

Output Buffer Delay

In addition to allowing for output buffer duty-cycle adjustment, the output buffer delay chain allows you to adjust the delays between the data bits in your output bus to introduce or compensate channel-to-channel skew. Incorporating skew to the output bus can help minimize simultaneous switching events by enabling smaller parts of the bus to switch simultaneously instead of the whole bus. This feature is useful in DDR3 SDRAM interfaces where the memory system clock delay can be much larger than the data and data clock/strobe delay. You can use this delay chain to add delay to the data and data clock/strobe to better match the memory system clock delay.

Slew Rate Control

HardCopy IV devices provide four levels of static output slew rate control: 0, 1, 2, and 3; Level 0 is the slowest slew rate setting and level 3 is the fastest slew rate setting. The default setting for the HSTL and SSTL I/O standards is **3**. A fast slew rate setting allows you to achieve higher I/O performance, and a slow slew-rate setting reduces system noise and signal overshoot. This feature is disabled if you use the OCT R_S features.

Drive Strength

You can choose the optimal drive strength required for your interface after performing board simulation. Higher drive strength helps provide a larger voltage swing, which in turn provides bigger eye diagrams with greater timing margin. However, higher drive strengths typically require more power, result in faster slew rates, and add to simultaneous switching noise. You can use the slew rate control with this feature to minimize simultaneous switching noise (SSN) with higher drive strengths. This feature is also disabled if you use the OCT R_S feature, which is the default drive strength in HardCopy IV devices. Use the OCT R_T/R_S setting for unidirectional read-and-write data and the dynamic OCT setting for bidirectional data signals. You must simulate the system to determine the drive strength required for command, address, and clock signals.

PLL

You can use PLLs to generate the memory interface controller clocks, such as the 0° system clock, the –90° or 270° phase-shifted write clock, the half-rate PHY clock, and the resynchronization clock. You can also use the PLL reconfiguration feature to calibrate the resynchronization phase shift to balance the setup and hold margin. The VCO and counter setting combinations may be limited for high-performance memory interfaces.

Altera recommends enabling the PLL reconfiguration feature and the DLL phase offset feature (DLL reconfiguration) for HardCopy IV devices. Because HardCopy IV devices are mask programmed, they cannot be changed after the silicon is fabricated. By implementing these two features, you can perform timing adjustments to improve or resolve timing issues after the silicon is fabricated.

For more information about HardCopy IV PLLs, refer to the *Clock Networks and PLLs in HardCopy IV Devices* chapter.

Document Revision History

Table 7–15 lists the revision history for this chapter.

Table 7–15. Document Revision History

Date	Version	Changes Made
January 2010	2.1	■ Updated Table 7–6. ■ Minor text edits.
June 2009	2.0	■ Removed "Conclusion" and "Referenced Documents" sections. ■ Updated Table 7–1. ■ Added HardCopy IV GX information. HardCopy IV is referred as HardCopy IV E. New device packages added.
December 2008	1.0	Initial release.

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

HIV51008-2.1

The HardCopy® IV device family offers up to 1.25-Gbps differential I/O capabilities to support source-synchronous communication protocols such as Utopia, RapidIO®, XSBI, SGMII, SFI, and SPI. HardCopy IV and Stratix® IV devices have identical circuitry for high-speed differential I/O interfaces and dynamic phase alignment (DPA). HardCopy IV high-speed I/Os support the same I/O standards and implementation guidelines as Stratix IV devices. You can prototype high-speed interfaces with Stratix IV devices and map the design to HardCopy IV devices.

Because of differences in resource availability, you must set the **HardCopy IV companion device** option in the Quartus® II software to map your Stratix IV project to a HardCopy IV device.

HardCopy IV devices have the same dedicated circuitry as Stratix IV devices for high-speed differential I/O support:

- Differential I/O buffer
- Transmitter serializer
- Receiver deserializer
- Data realignment
- DPA
- Synchronizer (FIFO buffer)
- Analog phase-locked looks (PLLs) located on the left and right sides of the device

For high-speed differential interfaces, HardCopy IV devices support the following differential I/O standards:

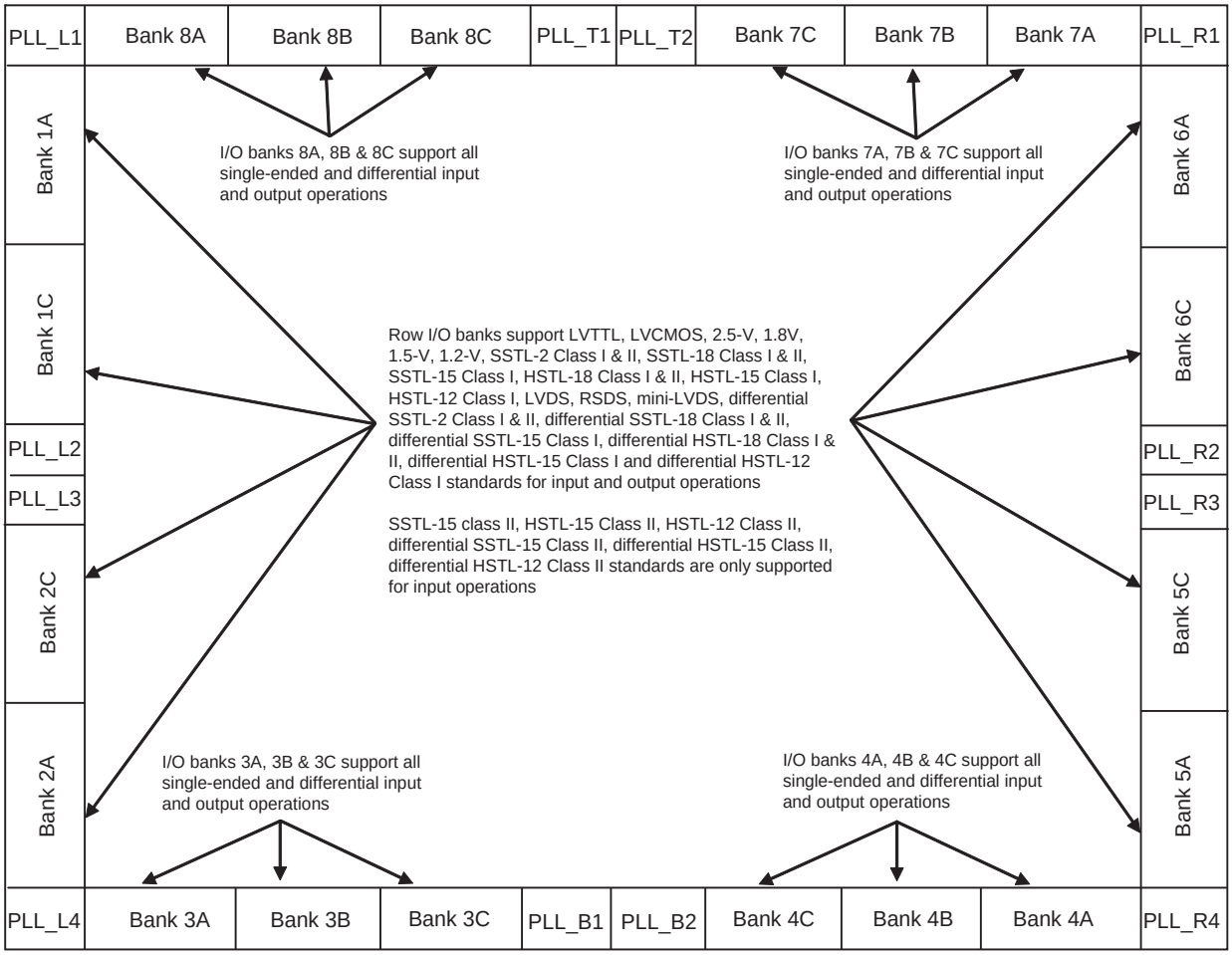
- Low voltage differential signaling (LVDS)
- Mini-LVDS
- Reduced swing differential signaling (RSDS)
- Differential HSTL
- Differential SSTL

You can use HSTL and SSTL I/O standards only for PLL clock inputs and outputs in differential mode.

I/O Banks

HardCopy IV E I/Os are divided into 16 to 20 I/O banks. The dedicated circuitry that supports high-speed differential I/Os is located in the left and right (row) I/O banks of the device. Figure 8–1 shows the different banks and the I/O standards supported by the banks.

Figure 8–1. I/O Banks in HardCopy IV E Devices *(Note 1), (2), (3), (4), (5), (6)*



Notes to Figure 8–1:

(1) The 1152- and 1517-pin packages have 20 I/O banks. The 780-pin package has 16 I/O banks.

(2) Figure 8–1 is a top view of the silicon die that corresponds to a reverse view for flip-chip packages. It is a graphical representation only. For exact locations, refer to the pin list and Quartus II software.

(3) Differential HSTL and SSTL I/Os use two single-ended outputs with the second output programmed as inverted for the transmitter and uses a true SSTL/HSTL differential input buffer for the receiver.

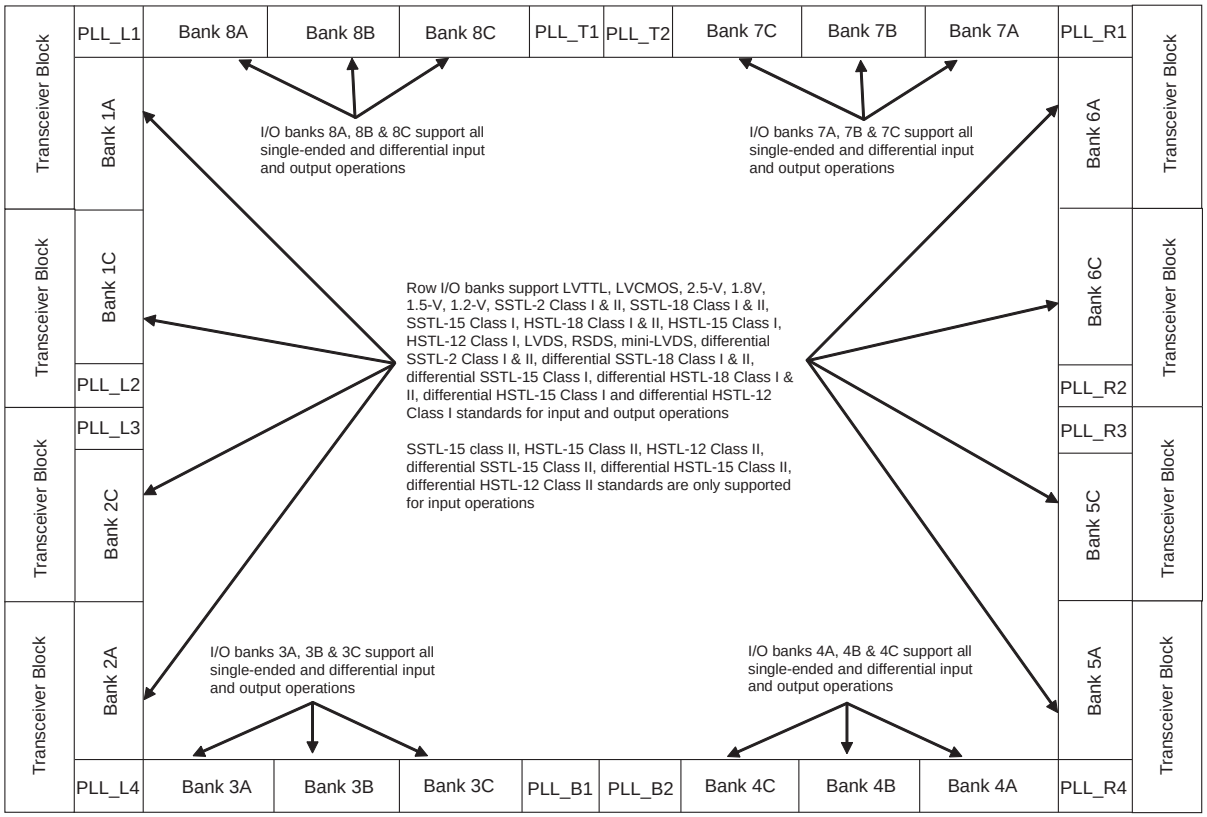
(4) Top and bottom I/O differential HSTL and SSTL inputs use LVDS differential input buffers without on-chip differential termination support.

(5) Top and bottom I/O supports LVDS outputs using single-ended buffers and external resistor networks.

(6) The PLL blocks are shown for location purposes only and are not considered additional banks. The PLL input and output uses the I/Os in adjacent banks.

Figure 8–2 shows a high-level chip overview of the HardCopy IV GX device.

Figure 8–2. High-Speed Differential I/Os with DPA Locations *(Note 1)*, *(2)*, *(3)*, *(4)*, *(5)*



Notes to Figure 8–2:

(1) Figure 8–2 is a top view of the silicon die that corresponds to a reverse view for flip-chip packages. It is a graphical representation only. for exact locations, refer to the pin list and Quartus® II software.

(2) Differential HSTL and SSTL I/Os use two single-ended outputs with the second output programmed as inverted for the transmitter, and uses a true SSTL/HSTL differential input buffer for the receiver.

(3) Top and bottom I/O differential HSTL and SSTL inputs use LVDS differential input buffers without on-chip differential termination support.

(4) Top and bottom I/O supports LVDS outputs using single-ended buffers and external resistor networks.

(5) The PLL blocks are shown for location purposes only and are not considered additional banks. The PLL input and output uses the I/Os in adjacent banks.

LVDS Channels

HardCopy IV devices support LVDS on both row I/O banks and column I/O banks. There are true LVDS input and output buffers on row I/O banks. On column I/O banks, there are true LVDS input buffers but neither true LVDS output buffers nor dedicated high-speed circuitry. However, you can configure all column user I/Os, including I/Os with true LVDS input buffers, as emulated LVDS output buffers. Table 8–1 shows the LVDS channels supported in HardCopy IV E device row I/O banks.

Table 8–1. LVDS Channels Supported in HardCopy IV E Device Left and Right (Row) I/O Banks *(Note 1)*

HardCopy IV E Device	484-Pin FineLine BGA	780-Pin FineLine BGA *(2)*	1152-Pin FineLine BGA	1517-Pin FineLine BGA *(3)*
HC4E25W	48Rx + 48Tx	48Rx + 48Tx *(2)*	—	—
HC4E25F	48Rx + 48Tx	56Rx + 56Tx	—	—
HC4E35L	—	—	88Rx + 88Tx	88Rx + 88Tx
HC4E35F	—	—	88Rx + 88Tx	88Rx + 88Tx

Notes to Table 8–1:

(1) The HardCopy IV E device family does not offer a 1760-pin package.

(2) The Stratix IV device EP4SE230F780 offers 56Rx + 56Tx or 112eTx channels and therefore has more transceiver channels than a HardCopy IV E device with the wire-bond package.

(3) Stratix IV devices EP4SE530F1517 and EP4SE820H1517 offer 112Rx + 112Tx or 224eTx channels and therefore have more transceiver channels than HardCopy IV E devices.

Table 8–2 shows the LVDS channels supported in HardCopy IV E device top and bottom (column) I/O banks.

Table 8–2. LVDS Channels Supported in HardCopy IV E Device Top and Bottom (Column) I/O Banks *(Note 1)*, *(2)*

HardCopy IV E Device	484-Pin FineLine BGA	780-Pin FineLine BGA *(3)*	1152-Pin FineLine BGA	1517-Pin FineLine BGA
HC4E25W	24Rx + 24eTx or 48eTx	24Rx + 24eTx or 48eTx	—	—
HC4E25F	24Rx + 24eTx or 48eTx	64Rx + 64eTx or 128eTx	—	—
HC4E35L	—	—	96Rx + 96eTx or 192eTx	128Rx + 128eTx or 256eTx
HC4E35F	—	—	96Rx + 96eTx or 192eTx	128Rx + 128eTx or 256eTx

Notes to Table 8–2:

(1) LVDS input buffers at top and bottom I/O banks are true LVDS input buffers. All user I/Os, including I/Os with true LVDS input buffers, can be configured as emulated LVDS output buffers.

(2) Rx = true LVDS input buffers with OCT RD, Tx = true LVDS output buffers, and eTx = emulated LVDS output buffers (either LVDS_E_1R or LVDS_E_3R).

(3) Stratix IV device EP4SE230F780 offers 64 Rx + 64 eTx or 128 eTx channels and therefore has more transceiver channels than a HardCopy IV E device with the wire-bond package.

Table 8–3 shows the LVDS channels supported in HardCopy IV GX device row I/O banks.

Table 8–3. LVDS Channels Supported in HardCopy IV GX Device Left and Right (Row) I/O Banks *(Note 1)*, *(2)*

HardCopy IV GX Device	780-Pin FineLine BGA	1152-Pin FineLine BGA	1152-Pin FineLine BGA *(3)*	1517-Pin FineLine BGA *(3)*
HC4GX15LA	28Rx + 28Tx	—	—	—
HC4GX15L	—	—	—	—
HC4GX25L	—	44Rx + 44Tx	—	—
HC4GX25F	—	—	44Rx + 44Tx	—
HC4GX35F	—	—	44Rx + 44Tx	88Rx + 88Tx

Notes to Table 8–3:

(1) The HardCopy IV GX device family does not offer a 1760-pin package.
(2) The LVDS channel count does not include dedicated clock input pins.
(3) This package supports PMA-only transceiver channels.

Table 8–4 shows the LVDS channels supported in HardCopy IV GX device top and bottom (column) I/O banks.

Table 8–4. LVDS Channels Supported in HardCopy IV GX Device Top and Bottom (Column) I/O Banks *(Note 1)*, *(2)*, *(3)*

HardCopy IV GX Device	780-Pin FineLine BGA	1152-Pin FineLine BGA	1152-Pin FineLine BGA *(4)*	1517-Pin FineLine BGA *(4)*
HC4GX15LA	64Rx + 64eTx or 128eTx	—	—	—
HC4GX15L	64Rx + 64eTx or 128eTx	—	—	—
HC4GX25L	72Rx + 72eTx or 144eTx	96Rx + 96eTx or 192eTx	—	—
HC4GX25F	—	—	96Rx + 96eTx or 192eTx	—
HC4GX35F	—	—	96Rx + 96eTx or 192eTx	96Rx + 96eTx or 192eTx

Note to Table 8–4:

(1) The HardCopy IV GX device family does not offer a 1760-pin package.
(2) The LVDS channel count does not include dedicated clock input pins.
(3) Rx = true LVDS input buffers with OCT RD, Tx = true LVDS output buffers, and eTx = emulated LVDS output buffers (either LVDS_E_1R or LVDS_E_3R).
(4) This package supports PMA-only transceiver channels.

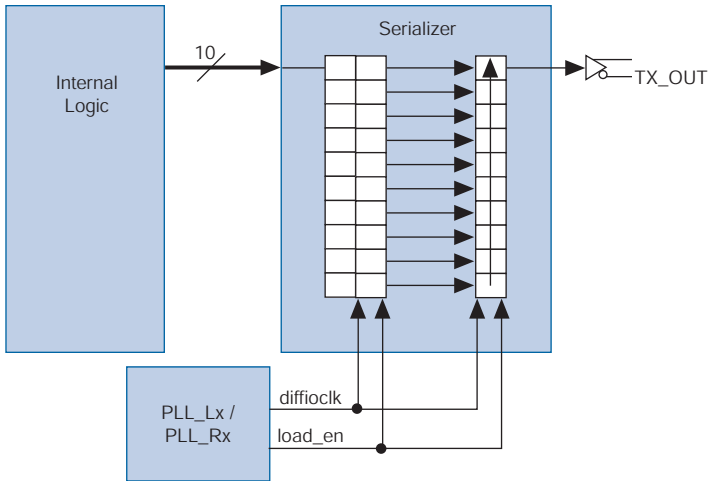
If you have read about high-speed differential I/O interfaces and DPA in the *Stratix IV Device Handbook*, refer to "Design Recommendations" on page 8–24 and "Differences Between Stratix IV and HardCopy IV Devices" on page 8–25.

Differential Transmitter

The HardCopy IV transmitter has dedicated circuitry to provide support for LVDS signaling. The dedicated circuitry consists of a differential buffer, a serializer, and a shared analog PLL (left or right PLL). The differential buffer can drive out LVDS, mini-LVDS, and RSDS signaling levels. The serializer takes up to 10 bits wide parallel data from the FPGA core, clocks it into the load registers, and serializes it using shift registers clocked by the left or right PLL before sending the data to the differential buffer. The most significant bit (MSB) of the parallel data is transmitted first.

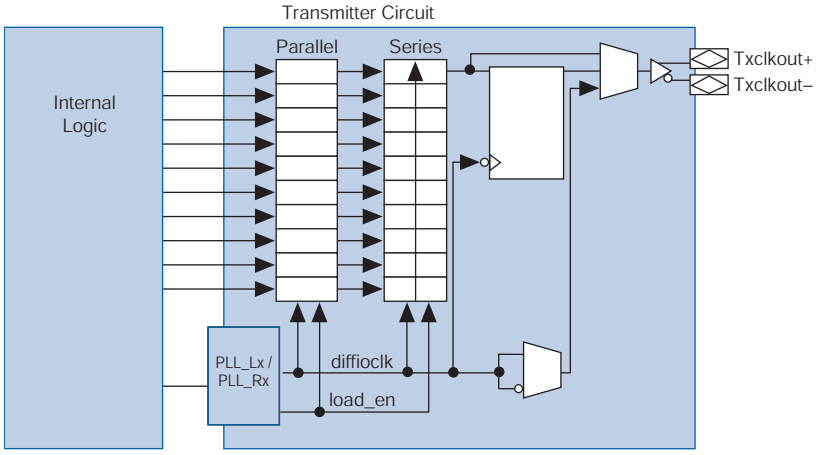
The load and shift registers are clocked by the load enable (`load_en`) signal and the `diffioclk` (clock running at serial data rate) signal generated from `PLL_Lx` (left PLL) or `PLL_Rx` (right PLL). The serialization factor can be statically set to ×4, ×6, ×7, ×8, or ×10 by using the Quartus II software. The load enable signal is derived from the serialization factor setting. Figure 8–3 is a block diagram of the HardCopy IV transmitter.

Figure 8–3. HardCopy IV Transmitter



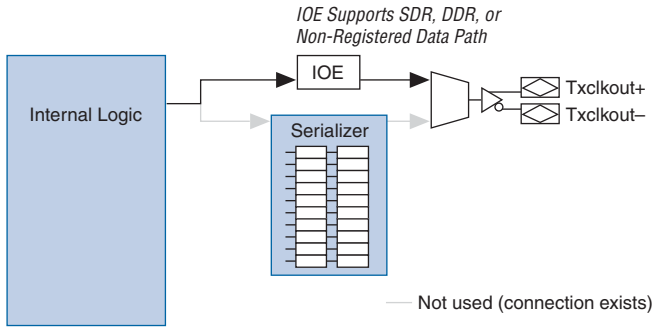
The HardCopy IV transmitter data channel can be configured to generate a source synchronous transmitter clock output, allowing you to place the output clock near the data outputs to simplify board layout and reduce clock-to-data skew. Different applications often require specific clock-to-data alignments or specific data rate to clock rate factors. The transmitter can output a clock signal at the same rate as the data. Depending on the serialization factor, the output clock can also be divided by a factor of 2, 4, 8, or 10. You can set the phase of the clock in relation to the data at 0° or 180° (edge or center aligned). The left and right PLLs (`PLL_Lx` and `PLL_Rx`) provide additional support for other phase shifts in 45° increments. These settings are made statically in the Quartus II MegaWizard™ Plug-In Manager. Figure 8–4 shows the HardCopy IV transmitter in clock output mode.

Figure 8–4. HardCopy IV Transmitter in Clock Output Mode



You can bypass the HardCopy IV serializer to support DDR (×2) and SDR (×1) operations to achieve a serialization factor of 2 and 1, respectively. The I/O element (IOE) contains two data output registers that can each operate in either DDR or SDR mode. The clock source for the registers in the IOE can come from any routing resource, from the left or right PLL (`PLL_Lx/PLL_Rx`), or from the top or bottom (`PLL_Tx/PLL_Bx`) PLL. Figure 8–5 shows the serializer bypass path.

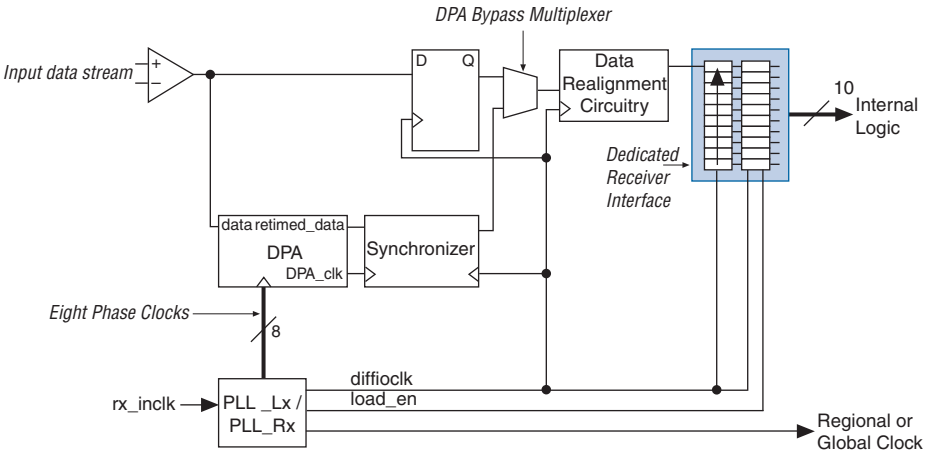
Figure 8–5. HardCopy IV Serializer Bypass



Differential Receiver

HardCopy IV devices have dedicated circuitry for receiving high-speed differential signals. Figure 8–6 shows a HardCopy IV receiver block diagram. The receiver has a differential buffer, a shared `PLL_Lx/PLL_Rx`, DPA, synchronization FIFO buffer, data realignment block, and a deserializer. The differential buffer can receive LVDS, mini-LVDS, and RSDS signal levels, which are statically set in the Quartus II software Assignment Editor. The PLL receives the external source clock input that is transmitted with the data and generates different phases of the same clock. The DPA block chooses one of the clocks from the left or right PLL and aligns the incoming data on each channel.

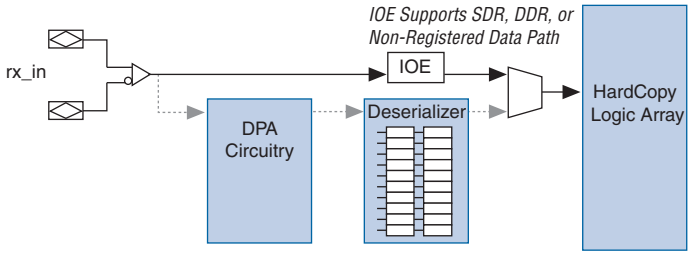
Figure 8–6. HardCopy IV Receiver Block Diagram



The synchronizer circuit is a 1-bit wide by 6-bit deep FIFO buffer that compensates for any phase difference between the DPA clock and the data realignment block. If necessary, the data realignment circuit inserts a single bit of latency in the serial bit stream to align to the word boundary. The deserializer includes shift registers and parallel load registers, and sends a maximum of 10 bits to the internal logic. The data path in the HardCopy IV receiver is clocked by either a `diffioclk` signal or the DPA recovered clock. The deserialization factor can be statically set to 4, 6, 7, 8, or 10 by using the Quartus II software. The left or right PLLs (`PLL_Lx/PLL_Rx`) generate the load enable signal, which is derived from the deserialization factor setting.

You can bypass the HardCopy IV deserializer in the Quartus II MegaWizard Plug-In Manager to support DDR (×2) or SDR (×1) operations. The DPA and the data realignment circuit cannot be used when the deserializer is bypassed. The IOE contains two data input registers that can operate in DDR or SDR mode. The clock source for the registers in the IOE can come from any routing resource, from the left or right PLLs, or from the top or bottom PLLs. Figure 8–7 shows the deserializer bypass data path.

Figure 8–7. Deserializer Bypass



Receiver Data Realignment Circuit (Bit Slip)

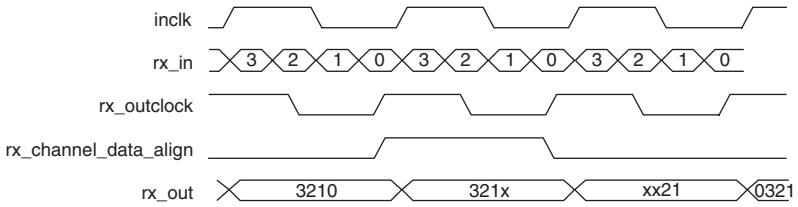
Skew in the transmitted data, along with skew added by the link, causes channel-to-channel skew on the received serial data streams. If the DPA is enabled, the received data is captured with different clock phases on each channel. This may cause the received data to be misaligned from channel to channel. To compensate for this channel-to-channel skew and establish the correct received word boundary at each channel, each receiver channel has a dedicated data realignment circuit that realigns the data by inserting bit latencies into the serial stream.

An optional RX_CHANNEL_DATA_ALIGN port controls the bit insertion of each receiver independently controlled from the internal logic. The data slips one bit for every pulse on RX_CHANNEL_DATA_ALIGN. The following conditions are required for the RX_CHANNEL_DATA_ALIGN signal:

- The minimum pulse width is one period of the parallel clock in the logic array
- The minimum low time between pulses is one period of the parallel clock
- There is no maximum high or low time
- Valid data is available two parallel clock cycles after the rising edge of RX_CHANNEL_DATA_ALIGN

Figure 8–8 shows the receiver output (rx_out) after one bit slip pulse with the serialization factor set to 4.

Figure 8–8. Data Realignment Timing

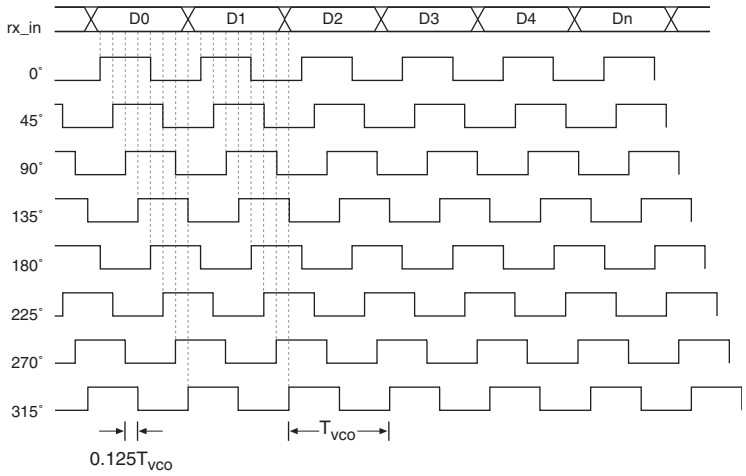


The data realignment circuit can have up to 11 bit-times of insertion before a rollover occurs. The bit rollover point can be from 1 to 11 bit-times, independent of the deserialization factor. An optional status port, rx_cda_max, is available to the FPGA from each channel to indicate when the preset rollover point is reached.

Dynamic Phase Aligner (DPA)

The DPA block takes in high-speed serial data from the differential input buffer and selects one of the eight phase clocks from the left or right PLL to sample the data. The DPA chooses a phase closest to the phase of the serial data. The maximum phase offset between the received data and the selected phase is 1/8 unit interval (UI), which is the maximum quantization error of the DPA. The eight phases of the clock are equally divided, giving a 45° resolution. Figure 8–9 shows the possible phase relationships between the DPA clocks and the incoming serial data.

Figure 8–9. DPA Clock Phase to Serial Data Timing Relationship



The DPA block continuously monitors the phase of the incoming serial data and selects a new clock phase if required. You can prevent the DPA from selecting a new clock phase by asserting the optional `rx_dpll_hold` port, which is available for each channel.

The DPA block requires a training pattern and a training sequence of at least 256 repetitions. The training pattern is not fixed, so you can use any training pattern with at least one transition on each channel. An optional output port (`rx_dpa_locked`) is available to the internal logic to indicate when the DPA block has settled on the closest phase to the incoming data phase. The DPA block de-asserts `rx_dpa_locked` depending on the option selected in the Quartus II MegaWizard Plug-In Manager, when either a new phase is selected, or when the DPA has moved two phases in the same direction. The `rx_dpa_locked` signal is synchronized to the DPA clock domain and should be considered as the initial indicator for the lock condition. Use data checkers to validate the data integrity.

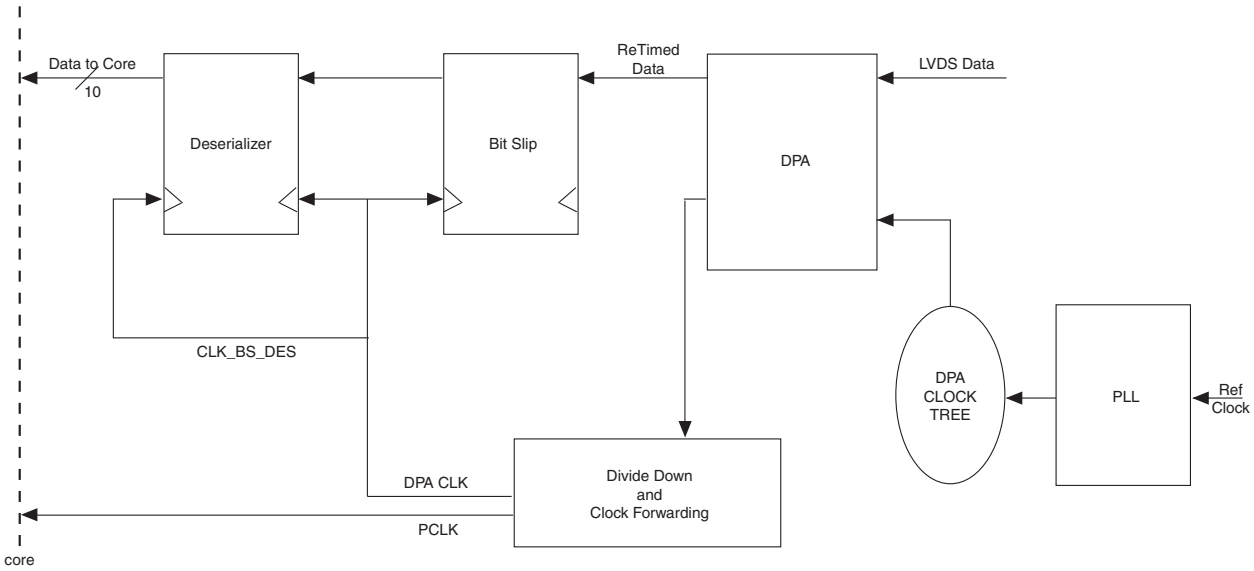
An independent reset port (`rx_reset`) is available to reset the DPA circuitry. The DPA circuitry must be retrained after reset.

Soft-CDR Mode

The HardCopy IV LVDS channel offers soft-CDR mode to support the Gigabit Ethernet/SGMII protocols. Clock-data recovery (CDR) is required to extract the clock out of the clock-embedded data to support SGMII. In HardCopy IV devices, the CDR circuit is implemented in HCells.

In soft-CDR mode, the DPA circuitry selects an optimal DPA clock phase to sample the data and carry on the bit-slip operation and deserialization. The selected DPA clock is also divided down by the deserialization factor and then forwarded to the PLD core along with the de-serialized data. The LVDS block has an output called `divclkout` for the forwarded clock signal. This signal is put on the newly introduced PCLK (periphery clock) network. In HardCopy IV devices, every LVDS channel can be used in soft-CDR mode and can drive the core via the PCLK network. Figure 8–10 shows the path enabled in soft-CDR mode.

Figure 8–10. Soft-CDR Mode Data and Clock Path *(Note 1)*



Note to Figure 8–10:

(1) The synchronizer FIFO is bypassed in soft-CDR mode. The reference clock frequency must be suitable for the PLL to generate a clock that matches the data rate of the interface. The DPA circuitry can track parts per million (PPM) differences between the reference clock and the data stream.

Synchronizer

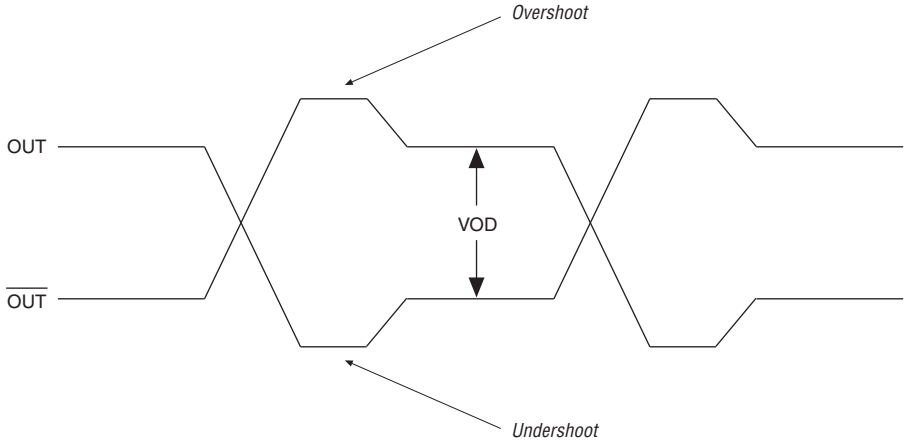
The synchronizer is a 1-bit × 6-bit deep FIFO buffer that compensates for the phase difference between the recovered clock from the DPA circuit and the `diffioclk` that clocks the rest of the logic in the receiver. The synchronizer can only compensate for phase differences, not frequency differences between the data and the receiver's `inclk`.

An optional port (`rx_fifo_reset`) is available to the internal logic to reset the synchronizer. The synchronizer is automatically reset when the DPA first locks to the incoming data. Altera recommends using `rx_fifo_reset` to reset the synchronizer when the DPA signals a loss-of-lock condition beyond the initial locking condition.

Pre-Emphasis and Output Differential Voltage (VOD)

HardCopy IV LVDS transmitters support four pre-emphasis and four VOD settings. Pre-emphasis increases the amplitude of the high frequency component of the output signal, and helps compensate for the frequency dependent attenuation along the transmission line. Figure 8–11 shows an LVDS output with pre-emphasis. The overshoot is produced by pre-emphasis. This overshoot must not be included in the VOD voltage. The definition of VOD is also shown in Figure 8–11.

Figure 8–11. Output Differential Voltage



Pre-emphasis is an important feature for high-speed transmission. Without pre-emphasis, the output current is limited by the VOD setting and the output impedance of the driver. At high frequency, the slew rate might not be fast enough to reach the full VOD before the next edge, producing a pattern dependent jitter.

With pre-emphasis, the output current is boosted momentarily during switching to increase the output slew rate. The overshoot introduced by the extra current happens only during switching and does not ring, unlike the overshoot caused by signal reflection. The amount of pre-emphasis required depends on the attenuation of the high-frequency component along the transmission line.

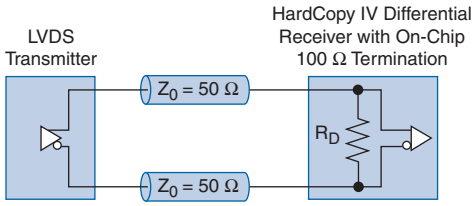
You can adjust pre-emphasis in HardCopy IV devices to create the right amount of overshoot at different transmission conditions. There are four settings for pre-emphasis: zero, low, medium, and high. The default setting is low. For a particular design, simulation with an LVDS buffer and transmission line can be used to determine the best pre-emphasis setting. The VOD can also be adjusted to any of the four settings: low, medium low, medium high, and high. The default setting is medium low.

Differential I/O Termination

HardCopy IV devices provide a 100-Ω, on-chip differential termination option on each differential receiver channel for LVDS standards. On-chip termination (OCT) saves board space by eliminating the need to add external resistors on the board. You can enable OCT in the Quartus II Assignment Editor.

On-chip differential termination is supported on all row I/O pins and serial/deserializer (SERDES) block clock pins: `clk[0,2,9,11]`. It is not supported for column I/O pins, high speed clock pins `clk[1,3,8,10]`, or the corner PLL clock inputs. Figure 8–12 illustrates device OCT.

Figure 8–12. On-Chip Differential Termination

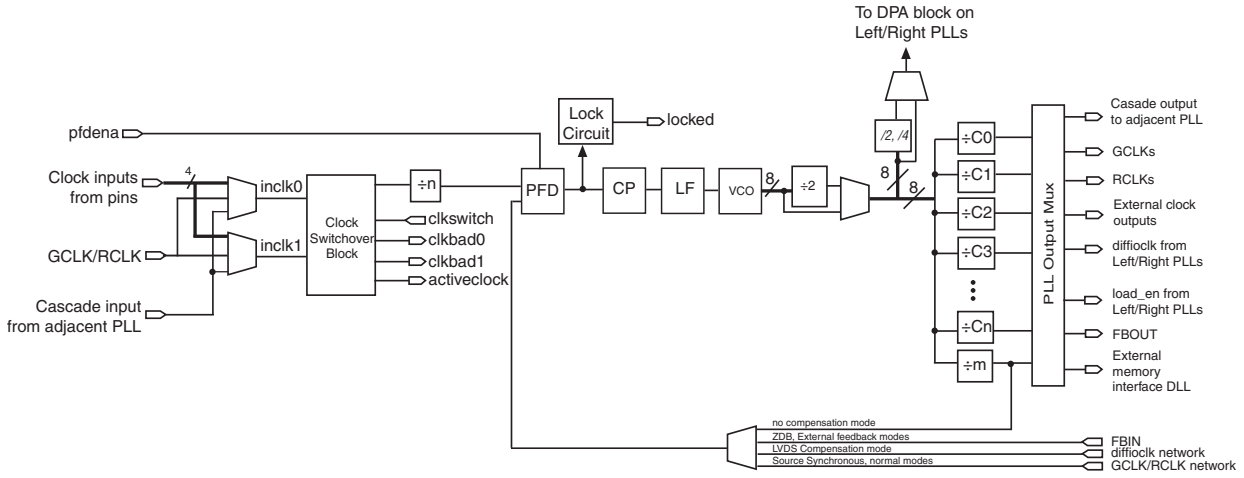


Left and Right PLLs (PLL_Lx and PLL_Rx)

HardCopy IV devices contain a maximum of eight left or right PLLs with up to four PLLs located on the left side (`PLL_L1`, `PLL_L2`, `PLL_L3`, and `PLL_L4`) and four on the right side (`PLL_R1`, `PLL_R2`, `PLL_R3`, and `PLL_R4`) of the device. The left PLLs can support high-speed differential I/O banks on the left side; the right PLLs can support banks only on the right side of the device. The high-speed differential I/O receiver and transmitter channels use these left and right PLLs to generate the parallel clocks (`rx_outclock` and `tx_outclock`) and high-speed clocks (`diffioclk`). Figure 8–1 on page 8–2 and Figure 8–2 on page 8–3 show the locations of the left/right PLLs for HardCopy IV devices E and HardCopy IV GX, respectively. The PLL VCO operates at the clock frequency of the data rate. Each left or right PLL offers a single serial data rate support, but up to two separate serialization or deserialization factors (from the `C0` and `C1` of left or right PLL clock outputs), or both. Clock switchover and dynamic left and right PLL reconfiguration are available in high-speed differential I/O support mode.

Figure 8–13 shows a simplified diagram of the major components of a HardCopy IV PLL.

Figure 8–13. HardCopy IV PLL



Clocking

The left and right PLLs feed into the differential transmitter and receiver channels through the LVDS and DPA clock networks. Figure 8–14 and Figure 8–15 show the corner and center PLL clock in HardCopy IV devices. Each left or right I/O bank consists of one LVDS clock network, for a total of four clock trees on the device. The center left and right PLLs can drive the LVDS clock network, therefore, clocking the transmitter and receiver channels above and below them. The corner left and right PLLs can drive the adjacent row-I/O banks only. For example, corner `PLL_L1` can drive the LVDS clock network only in I/O bank 1A and bank 1C. Therefore, with corner PLLs, each LVDS clock network can be driven by three PLLs: two center PLLs and one corner PLL. For HardCopy IV devices without a corner PLL, each clock tree can be driven by two center PLLs. Each clock network supports two full-duplex transceiver channels. However, Altera recommends you share the `diffioclk` and `load_en` signals between transmitting and receiving channels in the same I/O bank whenever possible. For more information about PLL clocking restrictions, refer to "Differential Pin Placement Guidelines" on page 8–16.

Figure 8–14. LVDS/DPA Clocks in HardCopy IV and Stratix IV Devices with Center PLLs

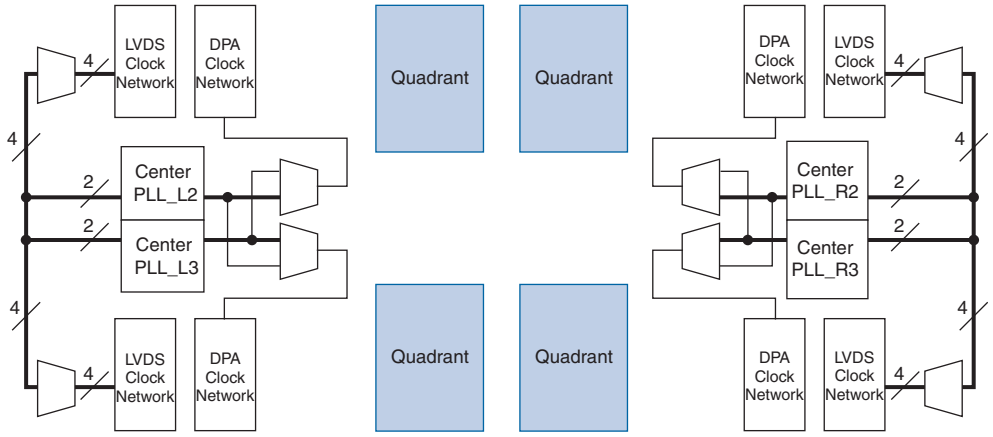
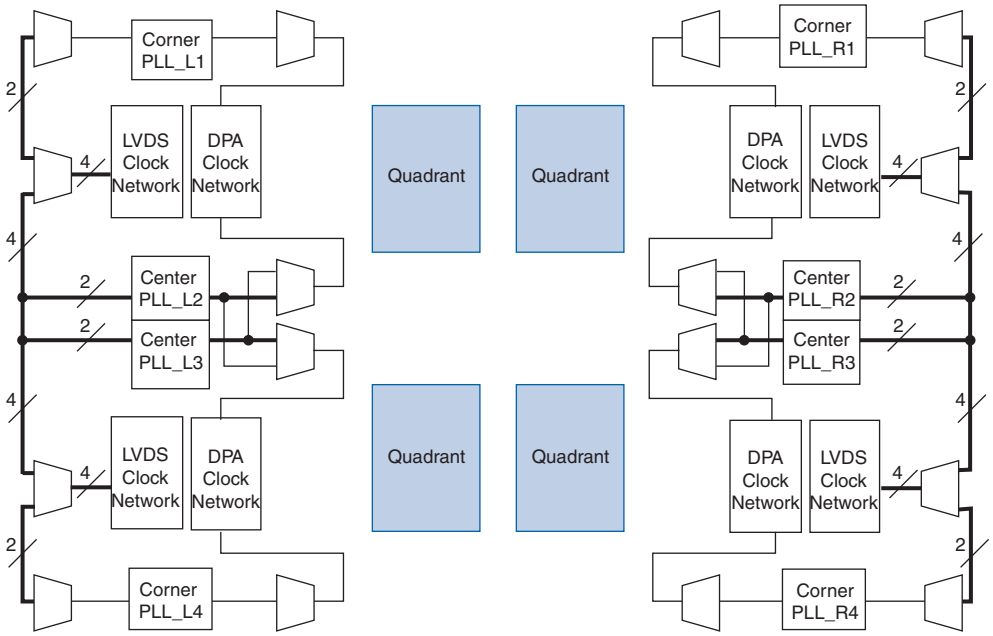


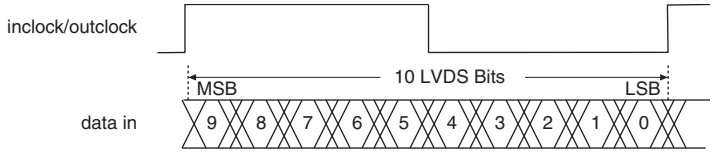
Figure 8–15. Clocks in HardCopy IV and Stratix IV Devices with Center and Corner PLLs



High-Speed Differential I/O Interfaces and DPA in HardCopy IV Devices Differential Data Orientation

There is a set relationship between an external clock and the incoming data. For operation at 1 Gbps with a SERDES factor of 10, the external clock is multiplied by 10, and phase-alignment is set in the PLL to coincide with the sampling window of each data bit. The data is sampled on the falling edge of the multiplied clock. Figure 8–16 shows the data bit orientation of the ×10 mode.

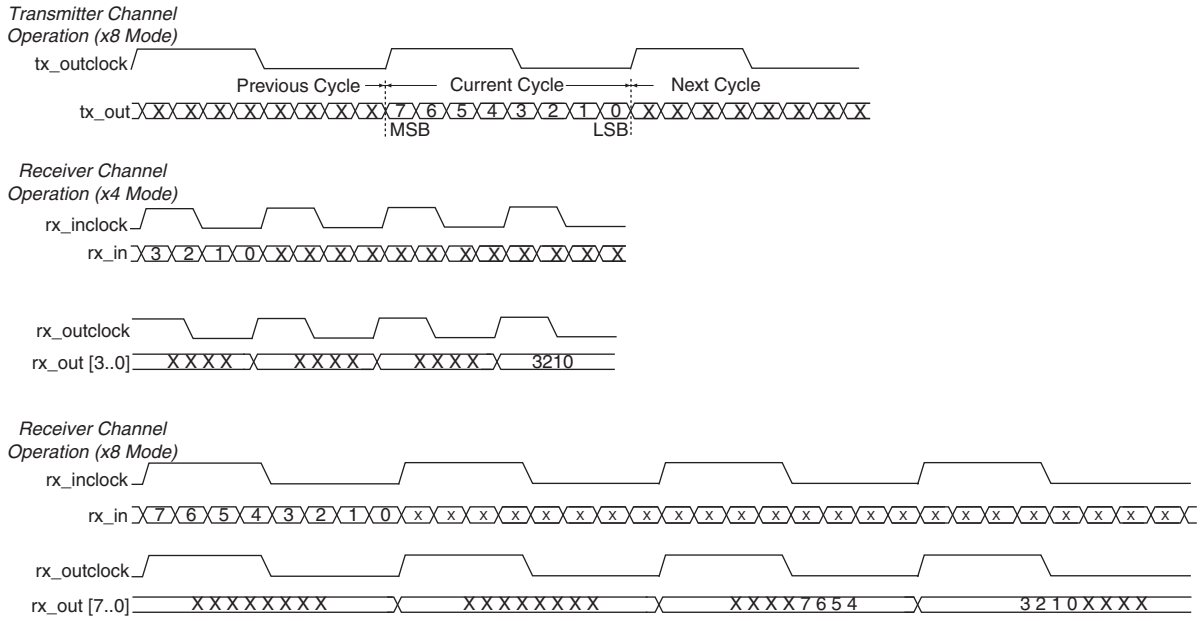
Figure 8–16. Bit Orientation in Quartus II Software Differential I/O Bit Position



Data synchronization is necessary for successful data transmission at high frequencies. Figure 8–17 shows the data bit orientation for a channel operation. This figure is based on the following:

- SERDES factor equals clock multiplication factor
- Edge alignment is selected for phase alignment
- Implemented in hard SERDES

Figure 8–17. Bit-Order and Word Boundary for One Differential Channel *(Note 1)*



Note to Figure 8–17:

(1) These are only functional waveforms and are not intended to convey timing information.

For other serialization factors, use the Quartus II software tools and find the bit position within the word. The bit positions after deserialization are listed in Table 8–5.

Table 8–5 shows the conventions for differential bit naming for eight differential channels. The MSB and LSB positions increase with the number of channels used in a system.

Table 8–5. LVDS Channels Supported in HardCopy IV Device Left and Right (Row) I/O Banks

Receiver Channel Number	Internal 8-Bit Parallel Data	
	MSB Position	LSB Position
1	7	0
2	15	8
3	23	16
4	31	24
5	39	32
6	47	40
7	55	48
8	63	56

Differential Pin Placement Guidelines

To ensure proper high-speed operation, differential pin placement guidelines have been established. Also, the Quartus II compiler automatically verifies these guidelines and issues an error message if they are not met.

Because DPA usage adds some constraints on the placement of high-speed differential channels, this section is divided into pin placement guidelines with and without DPA usage.

If you want to place both single-ended and differential I/Os in the same row or column I/O bank, refer to the *HardCopy IV Device I/O Features* chapter in volume 1 of the *HardCopy IV Device Handbook*.

Guidelines for DPA-Enabled Differential Channels

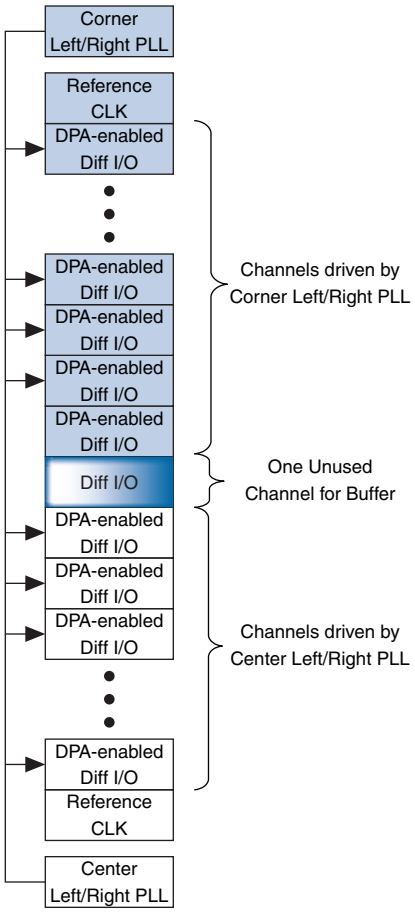
HardCopy IV GX devices have differential receivers and transmitters in I/O banks on the left and right sides of the device. Each receiver has a dedicated DPA circuit to align the phase of the clock to the data phase of its associated channel. When DPA-enabled channels are used in differential banks, you must adhere to the guidelines listed in the following sections.

Using Corner and Center Left/Right PLLs

If a differential bank is being driven by two left or right PLLs, and the corner left or right PLL is driving one group and the center left or right PLL is driving another group, there must be at least one row of separation between the two groups of DPA-enabled channels (refer to Figure 8–18). The two groups can operate at independent frequencies.

No separation is necessary if a single left or right PLL is driving DPA-enabled channels as well as DPA-disabled channels.

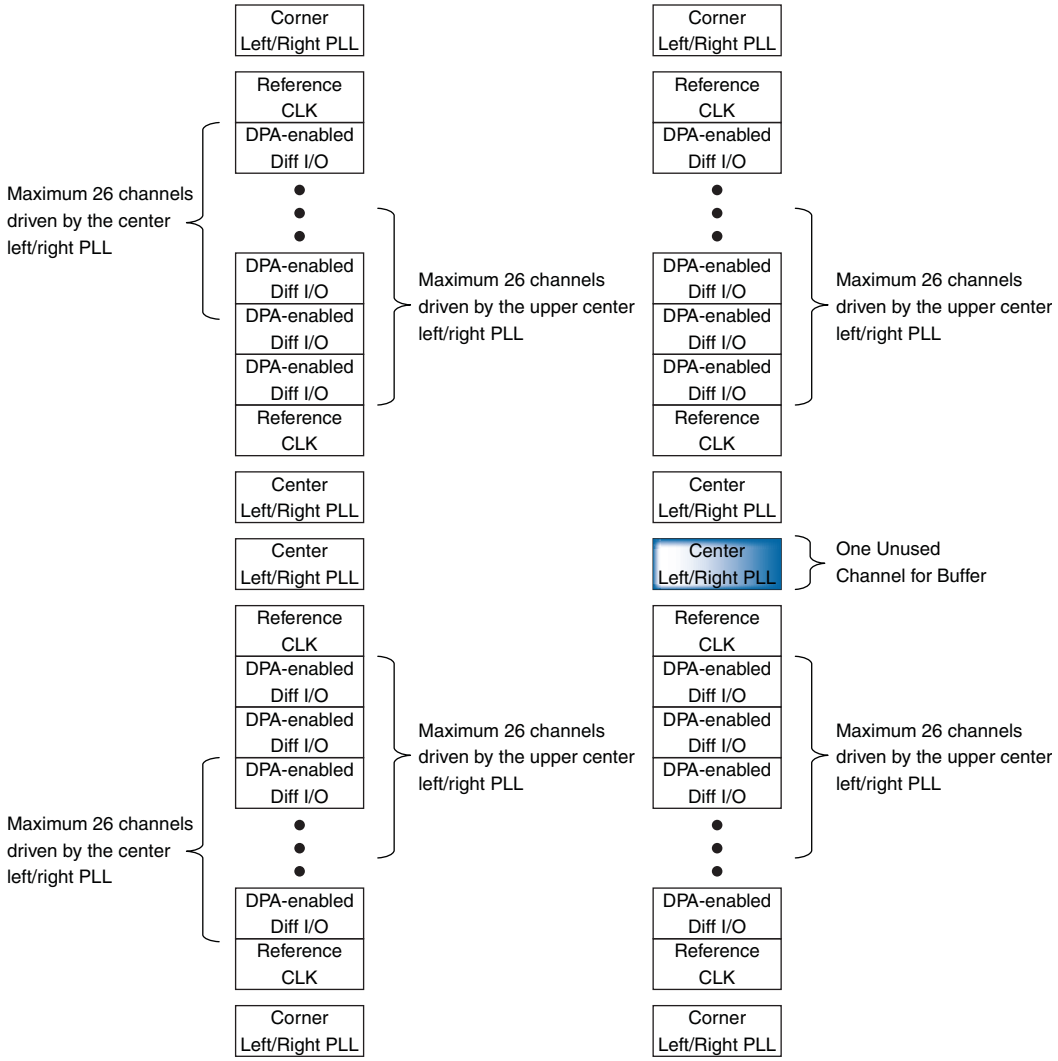
Figure 8–18. Corner and Center Left/Right PLLs Driving DPA-Enabled Differential I/Os in the Same Bank Using Both Center Left/Right PLLs



You can use center left or right PLLs to drive DPA-enabled channels simultaneously, as long as they drive these channels in their adjacent banks only, as shown in Figure 8–19.

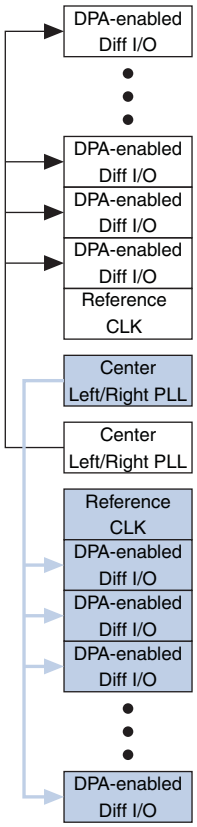
If one of the center left or right PLLs drives the top and bottom banks, the other center left or right PLL cannot be used to drive differential channels, as shown in Figure 8–19.

Figure 8–19. Center Left/Right PLLs Driving DPA-Enabled Differential I/Os



If the top PLL_L2/PLL_R2 drives DPA-enabled channels in the lower differential bank, the PLL_L3/PLL_R3 cannot drive DPA-enabled channels in the upper differential banks and vice versa. In other words, the center left or right PLLs cannot drive cross-banks simultaneously, as shown in Figure 8–20.

Figure 8–20. Invalid Placement of DPA-Enabled Differential I/Os Driven by Both Center Left/Right PLLs



Guidelines for DPA-Disabled Differential Channels

When DPA-disabled channels are used in the left and right banks of a HardCopy IV device, you must adhere to the guidelines in the following sections.

DPA-Disabled Channel Driving Distance

Each left or right PLL can drive all the DPA-disabled channels in the entire bank.

Using Corner and Center Left and Right PLLs

The following show how you can use corner and center left and right PLLs:

- You can use a corner left or right PLL (`PLL_L1`, `PLL_L4`, `PLL_R1`, and `PLL_R4`) to drive all transmitter channels and a center left or right PLL (`PLL_L2`, `PLL_L3`, `PLL_R2`, and `PLL_R3`) to drive all DPA-disabled receiver channels within the same differential bank. A transmitter channel and a receiver channel in the same LAB row can be driven by two different PLLs, as shown in Figure 8–21.
- A corner left or right PLL and a center left or right PLL can drive duplex channels in the same differential bank as long as the channels driven by each PLL are not interleaved. No separation is necessary between the group of channels driven by the corner and center left or right PLLs. Refer to Figure 8–21 and Figure 8–22.

Figure 8–21. Corner and Center Left and Right PLLs Driving DPA-Disabled Differential I/Os in the Same Bank

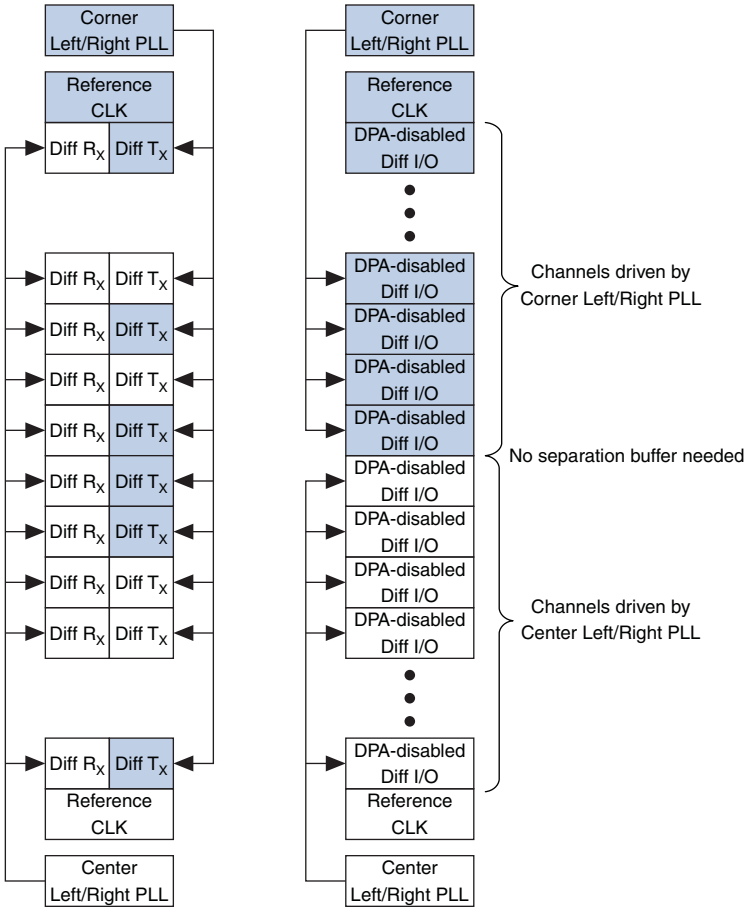
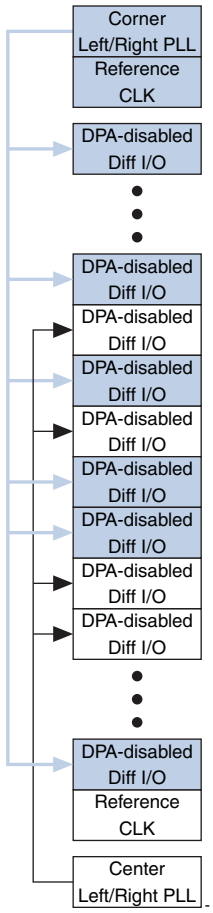


Figure 8–22. Invalid Placement of DPA-Disabled Differential I/Os Due to Interleaving of Channels Driven by the Corner and Center Left and Right PLLs



Using Both Center Left/Right PLLs

You can use both center left and right PLLs simultaneously to drive DPA-disabled channels on upper and lower differential banks, as shown in Figure 8–23. Unlike DPA-enabled channels, the center left and right PLLs can drive cross-banks. For example, the upper center left or right PLL can drive the lower differential bank while the lower center left or right PLL is driving the upper differential bank and vice versa, as shown in Figure 8–24.

Figure 8–23. Both Center Left and Right PLLs Simultaneously Driving DPA-Disabled Upper and Lower Bank Channels

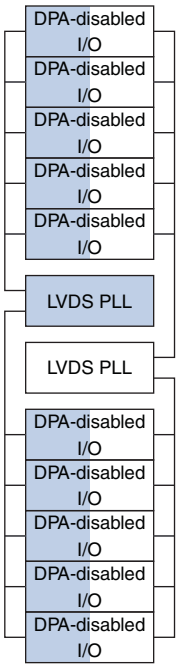
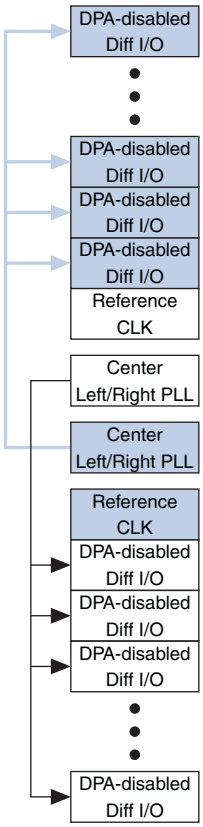


Figure 8–24. Both Center Left/Right PLLs Driving Cross-Bank DPA-Disabled Channels Simultaneously



Design Recommendations

To implement the high-speed differential interface successfully, Altera recommends that you follow these design guidelines:

1. Altera provides HardCopy IV IBIS models to verify I/O timing and characteristics. Altera strongly recommends you verify the I/O interfaces with simulation before you submit the design to the HardCopy Design Center.

For more information about signal integrity simulations with third-party tools, refer to the *Signal Integrity Analysis with Third-Party Tools* chapter in volume 3 of the *Quartus II Handbook*.

2. You can use center PLLs for both Tx and Rx, but corner PLLs are preferred for Tx applications over Rx applications.
3. Altera recommends you share the `lvdsclk` and `load_en` signals between transmitting and receiving channels in the same I/O bank whenever possible.

Differences Between Stratix IV and HardCopy IV Devices

- The HardCopy IV device family supports full high-speed differential I/O and DPA mapping from the Stratix IV family. Both families are designed with identical dedicated circuitry and thus support the same I/O standard, implementation guidelines, and performance.The HardCopy IV family does not offer a 1760-pin package.
- The Stratix IV E 780-pin package offers 56 Rx + 56 Tx or 112 eTX channels, while the HC4E25W 780-pin wire-bond packages offer 48Rx+48Tx channels on the row (left and right) I/O banks.
- The Stratix IV E 780-pin package offers 64 Rx + 64 eTx or 128 eTx channels, while the HC4E25W 780-pin wire-bond packages offer 24 Rx + 24 eTx or 48 eTx channels on the column (top and bottom) I/O banks.
- The HardCopy IV E 1517-pin package offers 88Rx+88Tx channels, while the EP4SE530 and EP4SE820 1517-pin packages offer 112 Rx + 112 Tx or 224 eTx channels on the row (left and right) I/O banks.

Document Revision History

Table 8–6 shows the revision history for this document.

Table 8–6. Document Revision History

Date	Version	Changes Made
January 2010	2.1	■ Updated Table 8–1, Table 8–2, Table 8–3, and Table 8–4. ■ Updated "Differences Between Stratix IV and HardCopy IV Devices" ■ Minor text edits.
June 2009	2.0	■ Added HardCopy IV GX information. ■ Updated tables for new device part numbers. ■ Removed "Referenced Documents" and "Conclusion."
December 2008	1.0	Initial release.

Section III. Hot Socketing and Testing

This section includes the following chapters:

- Chapter 9, Hot Socketing and Power-On Reset in HardCopy IV Devices
- Chapter 10, IEEE 1149.1 (JTAG) Boundary Scan Testing in HardCopy IV Devices

Revision History

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

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

HIV51009-1.0

This chapter contains information about hot-socketing specifications, power-on reset (POR) requirements, and their implementation in HardCopy® IV devices.

HardCopy IV devices offer hot socketing, which is also known as hot plug-in or hot swap, and power sequencing support without the use of any external devices. You can insert or remove a HardCopy IV device or a board in a system during system operation without causing undesirable effects to the running system bus or the board that was inserted into the system.

The hot-socketing feature also removes some of the difficulty when you use HardCopy IV devices or PCBs that contain a mixture of 3.0-, 2.5-, 1.8-, 1.5-, and 1.2-V devices. With the HardCopy IV hot-socketing feature, you no longer need to ensure a proper power-up sequence for each device on the board.

The HardCopy IV hot-socketing feature provides:

- Board or device insertion and removal without external components or board manipulation
- Support for any power-up sequence
- Non-intrusive I/O buffers to system buses during hot insertion

This chapter also discusses hot-socketing specification, its implementation, and the POR circuitry in HardCopy IV devices. The POR circuitry keeps the devices in the reset state until the power supplies are within operating range.

HardCopy IV Hot-Socketing Specifications

HardCopy IV devices are hot-socketing compliant without the need for any external components or special design requirements. Hot-socketing support in HardCopy IV devices has the following advantages:

- You can drive the device before power-up without damaging it.
- I/O pins remain tri-stated during power-up. The device does not drive out before or during power-up and does not affect other buses in operation.
- You can insert or remove a HardCopy IV device from a powered-up system board without damaging or interfering with normal system and board operation.

Devices Can Be Driven Before Power-Up

You can drive signals into I/O pins, dedicated input pins, and dedicated clock pins of HardCopy IV devices before or during power-up or power-down without damaging the device. HardCopy IV devices support power-up or power-down of all power supplies in any sequence to simplify system level design.

I/O Pins Remain Tri-Stated During Power-Up

A device that does not support hot socketing may interrupt system operation or cause contention by driving out before or during power-up. In a hot-socketing situation, the HardCopy IV device's output buffers are turned off during system power-up or power-down. Also, the HardCopy IV device does not drive out until the device is in user mode and working within recommended operating conditions.

Insertion or Removal of a HardCopy IV Device from a Powered-Up System

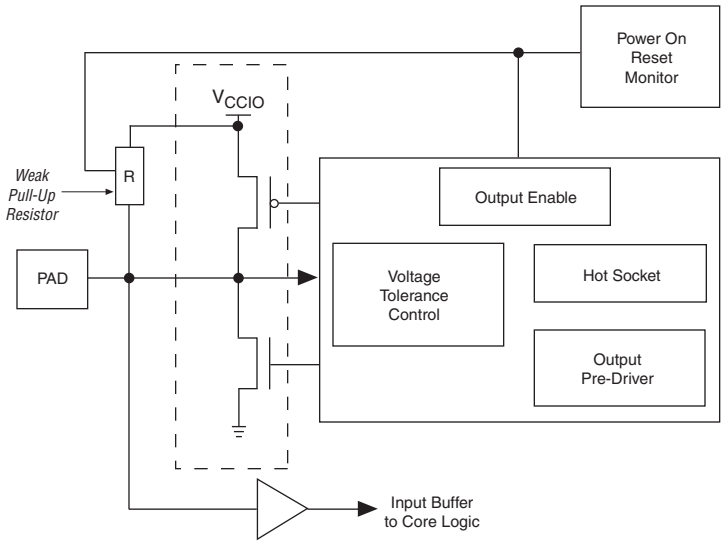
Devices that do not support hot socketing can short power supplies when powered up through the device signal pins. This irregular power-up can damage both the driving and driven devices and can disrupt card power-up.

A HardCopy IV device may be inserted into (or removed from) a powered-up system board without damaging or interfering with system board operation. You can power-up or power-down all power supplies in any sequence, as long as they are all ramped up to full rail before the HardCopy IV device starts to communicate with other devices on the board. This requirement is discussed in "Power-On Reset Circuitry" on page 9–3. HardCopy IV devices are immune to latch-up when performing hot socketing.

Hot-Socketing Feature Implementation in HardCopy IV Devices

The hot-socketing feature turns off the output buffer during power-up and power-down of the V_{CC}, V_{CCIO}, V_{CCPGM}, or V_{CCPD} power supplies. Each I/O pin has the circuitry shown in Figure 9–1.

Figure 9–1. Hot-Socketing Circuit Block Diagram for HardCopy IV Devices

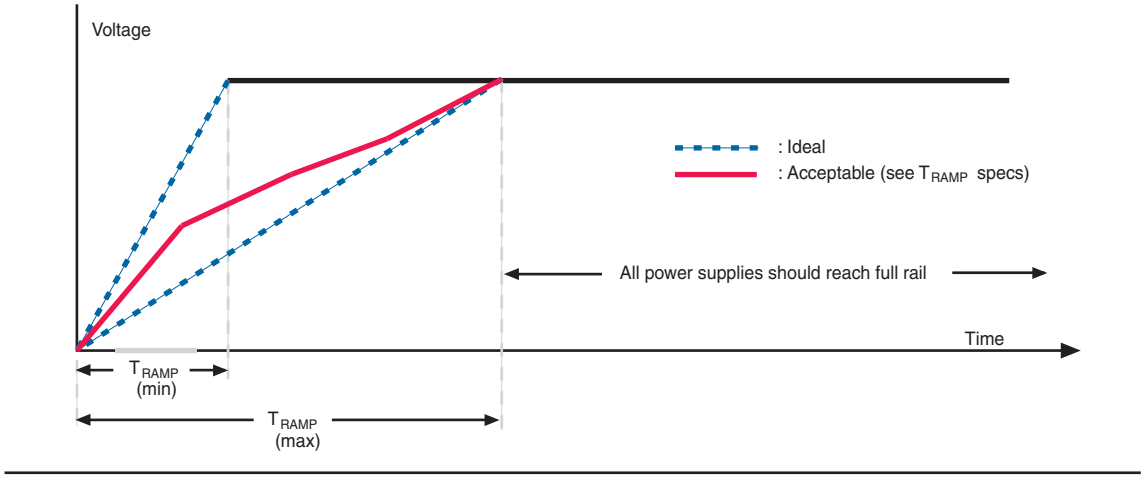


The POR circuit monitors the voltage level of power supplies (V_{CC}, V_{CCPD}, and V_{CCAUX}) and keeps the I/O pins tri-stated until the device is in user mode. The weak pull-up resistor (R) in the HardCopy IV input/output element (IOE) keeps the I/O pins from floating. The voltage tolerance control circuit permits the I/O pins to be driven by external voltages before V_{CC}, V_{CCIO}, V_{CCPGM}, and/or V_{CCPD} supplies are powered, and it prevents the I/O pins from driving out when the device is not in user mode.

Power-On Reset Circuitry

A power-on reset event occurs if all the POR-monitored power supplies, shown in Table 9–1 reach the recommended operating range within a certain period of time (specified as power supply ramp time, T_{RAMP}). Figure 9–2 shows the power supply specification. All power supplies' voltages have to rise monotonically within T_{RAMP}. This ensures the voltage levels do not remain indeterminate for a long time during power-up.

Figure 9–2. Power Supply Ramp Behavior

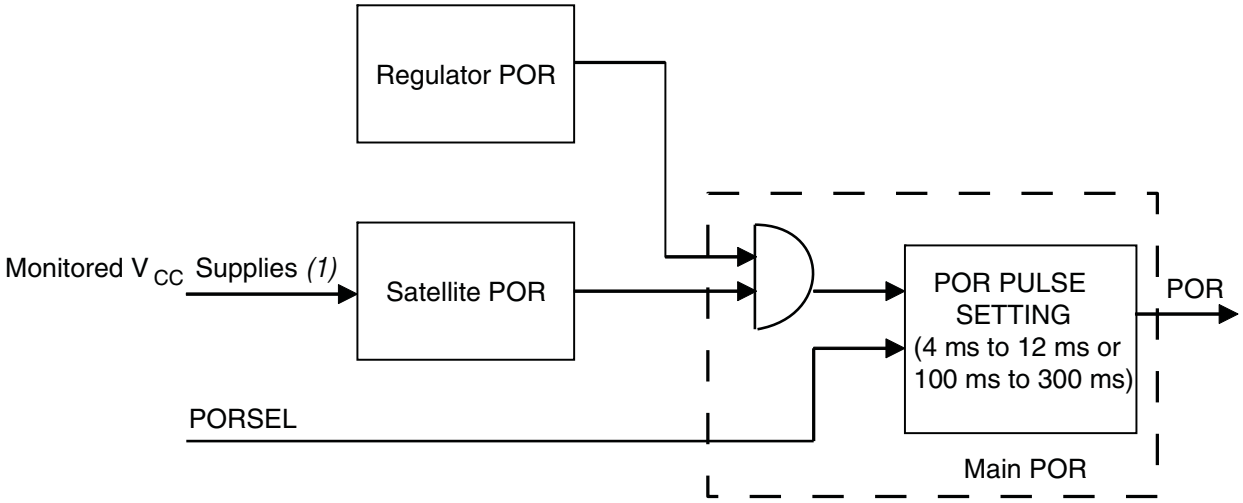


HardCopy IV devices provide a dedicated input pin (PORSEL) to select a T_{RAMP} range from 4 ms to 12 ms, or from 100 ms to 300 ms for all power supplies to ramp up. When the PORSEL pin is connected to ground, the T_{RAMP} can be from 100 ms to 300 ms. When the PORSEL pin is set to high, the T_{RAMP} can be from 4 ms to 12 ms.

The POR block consists of a regulator POR, satellite POR, and main POR to check the power supply levels for proper device operation. The regulator POR monitors the internal reference voltage for the temperature sensing diode and POR. The satellite POR monitors V_{CC}, V_{CCPD}, V_{CCPGM}, and V_{CCAUX} power supplies to ensure proper device operation. It also checks for functionality of I/O level shifters powered by V_{CCPD} and V_{CCPGM} during power-up mode. The main POR collects signals from both regulator and satellite PORs and generates POR pulse according to the PORSEL signal. A simplified block diagram of the POR block is shown in Figure 9–3.

All configuration-related dedicated and dual function I/O pins must be powered by V_{CCPGM}.

Figure 9–3. Simplified POR Block Diagram



Note to Figure 9–3:

(1) For more details about these supplies, refer to Table 9–1.

The POR circuit monitors the power supplies specified in Table 9–1.

Table 9–1. Power Supplies Monitored by the POR Circuitry

Power Supply	Description	Setting (V)
V_{CC}	Core voltage and periphery circuitry power supply	0.9
V_{CCAUX} *(1)*	Power supply for temperature sensing diode and POR circuitry	2.5
V_{CCPD}	I/O pre-driver power supply	2.5, 3.0
V_{CCPGM}	Configuration pins power supply	1.8, 2.5, 3.0

Note to Table 9–1:

(1) This power supply is for the auxiliary power supply in Stratix IV devices.

The POR circuit does not monitor the power supplies listed in Table 9–2.

Table 9–2. Power Supplies Not Monitored by the POR Circuitry

Voltage Supply	Description	Setting (V)
V_{CCIO}	I/O power supply	1.2, 1.5, 1.8, 2.5, 3.0
V_{CCA_PLL}	PLL analog global power supply	2.5
V_{CCD_PLL}	PLL digital power supply	0.9
V_{CC_CLKIN}	PLL differential clock input power supply (top and bottom I/O banks only)	2.5
V_{CCBAT}	Battery back-up power supply for design security volatile key storage	N/A

The POR specification is designed to ensure that all circuits in the HardCopy IV device are at certain known states during power up.

The POR signal pulse width is selectable using the PORSEL input pin. When PORSEL is set to low, the POR signal pulse width is set within the range of 100 ms to 300 ms. A POR pulse width of 100 ms to 300 ms allows serial flash devices with a 65 ms to 100 ms internal POR delay to be powered-up and ready to receive the nSTATUS signal from a HardCopy IV device. When the PORSEL is set to high, the POR signal pulse width is set within the range of 4 ms to 12 ms. A POR pulse width of 4 ms to 12 ms allows time for power supplies to ramp-up to full rail.

Because not all power supplies are monitored by POR, ensure that the power supplies are fully ramped up before the device starts to communicate with other devices on the system. Regardless of the voltage level of these power supplies, a HardCopy IV device continues to enter user-mode. One difference between Stratix IV and HardCopy IV devices is that Stratix IV devices allow more time for power supplies to ramp up during the configuration phase, before the device enters user mode. HardCopy IV devices, however, can enter user mode and release CONF_DONE within 12 ms or 100 ms. Therefore, you should always verify the voltage level of the power supply system before the HardCopy IV device starts to run.

Conclusion

HardCopy IV devices are hot-socketing compliant and allow successful device power-up without the need for any power sequencing. The POR circuitry keeps the devices in the reset state until the power supply voltage levels are within operating range.

Document Revision History

Table 9–3 shows the revision history for this chapter.

Table 9–3. Document Revision History

Date	Version	Changes Made
December 2008	1.0	Initial release.

10. IEEE 1149.1 (JTAG) Boundary Scan Testing in HardCopy IV Devices

HIV51010-2.0

All HardCopy® IV ASICs provide Joint Test Action Group (JTAG) boundary-scan test (BST) circuitry that complies with the IEEE Std. 1149.1 specification. The BST architecture offers the capability to efficiently test components on PCBs with tight lead spacing. Pin connections can be tested without using physical test probes, and functional data can be captured while a device is in normal operation. Boundary-scan cells in a device can force signals onto pins, or capture data from pin or core logic signals. Forced test data is serially shifted into the boundary-scan cells. Captured data is serially shifted out and externally compared to expected results.

A device using the JTAG interface uses four required pins: TDI, TDO, TMS, and TCK, and one optional pin, TRST. The TCK pin has an internal weak pull-down resistor, and the TDI, TMS, and TRST pins have internal weak pull-up resistors. The TDO output pin and all the JTAG input pins are powered by the 2.5-V/3.0-V V_{CCPD} supply of I/O bank 1A.

For more information about the JTAG pin description, refer to the *JTAG Boundary-Scan Testing in Stratix IV Devices* chapter in volume 1 of the *Stratix IV Device Handbook*.

JTAG Instructions

Table 10–1 shows the JTAG instructions supported in HardCopy IV devices for boundary-scan testing (BST). These 10-bit instructions are also supported in Stratix IV devices. However, HardCopy IV devices do not support the Stratix IV JTAG instructions used for in-circuit reconfiguration (ICR), because HardCopy IV devices do not require configuration.

For more information about the BST architecture and JTAG instructions supported in Stratix IV devices, refer to the *JTAG Boundary-Scan Testing in Stratix IV Devices* chapter in volume 1 of the *Stratix IV Device Handbook*.

Table 10–1. HardCopy IV JTAG Instructions (Part 1 of 2)

JTAG Instruction	Instruction Code	Description
SAMPLE/PRELOAD	00 0000 0101	Allows a snapshot of signals at the device pins to be captured and examined during normal device operation, and permits an initial data pattern to be output at the device pins.
EXTEST *(1)*	00 0000 1111	Allows the external circuitry and board-level interconnects to be tested by forcing a test pattern at the output pins and capturing test results at the input pins.
BYPASS	11 1111 1111	Places the 1-bit bypass register between the TDI and TDO pins, which allows the BST data to pass synchronously through selected devices to adjacent devices during normal device operation.
USERCODE	00 0000 0111	Loads the 32-bit user code into the device identification register and places the register between the TDI and TDO pins, allowing the user code to be serially shifted out of TDO.

Table 10–1. HardCopy IV JTAG Instructions (Part 2 of 2)

JTAG Instruction	Instruction Code	Description
IDCODE	00 0000 0110	Loads the 32-bit ID code into the device identification register and places the register between the TDI and TDO pins, allowing the ID code to be serially shifted out of TDO.
HIGHZ *(1)*	00 0000 1011	Places the 1-bit bypass register between the TDI and TDO pins, which allows the BST data to pass synchronously through selected devices to adjacent devices during normal device operation while tri-stating all of the I/O pins.
CLAMP *(1)*	00 0000 1010	Places the 1-bit bypass register between the TDI and TDO pins, which allows the BST data to pass synchronously through selected devices to adjacent devices during normal device operation while holding I/O pins to a state defined by the data in the boundary-scan register.

Note to Table 10–1:

(1) Bus hold and weak pull-up resistor features override the high-impedance state of HIGHZ, CLAMP, and EXTEST.

Similar to Stratix IV devices, HardCopy IV devices support the SignalTap® II Embedded Logic Analyzer, which monitors design operation over a period of time through the JTAG interface. The SignalTap II Embedded Logic Analyzer is a useful feature during the device prototyping phase, but should be removed if not required, after you map the design to a HardCopy IV device. HardCopy IV devices are mask programmed, and the SignalTap II logic cannot be removed after the HardCopy IV device is fabricated.

IDCODE and USERCODE

The IDCODE instruction gives you the ability to shift out a 32-bit identification (ID) code from HardCopy IV devices. ID codes are different in Stratix IV devices and unique for each HardCopy IV device. The ID code can be used to determine the correct device during BST. When the IDCODE instruction is issued, the ID code is loaded into a 32-bit device identification register for shifting out. Table 10–2 shows the ID codes for the HardCopy IV devices.

Table 10–2. 32-Bit HardCopy IV Device IDCODE *(Note 1)*, *(2)*

Device	IDCODE (32 Bits)			
	Version (4 Bits)	Part Number (16 Bits)	Manufacturer Identity (11 Bits)	LSB (1 Bit)
HC4GX15	0000	0010 0011 0001 0101	000 0110 1110	1
HC4GX25	0000	0010 0011 0010 0101	000 0110 1110	1
HC4GX35	0000	0010 0011 0011 0101	000 0110 1110	1
HC4E25	0000	0010 0110 0010 0101	000 0110 1110	1
HC4E35	0000	0010 0110 0011 0101	000 0110 1110	1

Notes to Table 10–2:

(1) The MSB is on the left.

(2) The LSB of IDCODE is always 1.

You can use the USERCODE instruction to shift out a 32-bit user code, which can also be used to uniquely identify the device. Unlike Stratix IV devices, the user code in HardCopy IV devices is mask programmed and cannot be changed after the silicon is fabricated. If the designer does not select a user code, the user code will be mask programmed to default values. When the USERCODE instruction is issued, the 32-bit user code is loaded into the same 32-bit device identification register used for the IDCODE instruction. The user code can then be serially shifted out.

Boundary-Scan Register

The boundary-scan register length for HardCopy IV devices differs from Stratix IV devices. The length also varies for each HardCopy IV device depending on device density and available I/O pin count. Table 10–3 lists the boundary-scan register length for HardCopy IV devices.

Table 10–3. HardCopy IV Boundary-Scan Register Length

Device	Boundary-Scan Register Length
HC4GX15	1146
HC4GX25	1722
HC4GX35	2262
HC4E25	1524
HC4E35	2670

Boundary-Scan Description Language (BSDL) Support

The boundary-scan description language (BSDL), a subset of VHDL, provides a syntax that allows you to describe the features of an IEEE Std. 1149.1 BST-capable device that can be tested.

There are two versions of the BSDL Customizer tool that you can use. The pre-configuration version generates a customized BSDL file for use before the device enters user mode, and the post-configuration version generates a customized BSDL file for use after the device enters user mode.

For more information about BSDL files for IEEE Std. 1149.1-compliant HardCopy IV devices, visit the Altera website at www.altera.com.

BSDL files for IEEE Std. 1149.1-compliant HardCopy IV devices can also be generated using the Quartus software version 8.1 or later. Visit the Altera website at www.altera.com for the procedure to generate the BSDL files using the Quartus II software.

For JTAG timing parameters and values, refer to the *DC and Switching Characteristics of HardCopy IV Devices* chapter in volume 4 of the *HardCopy IV Device Handbook*.

Document Revision History

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

Table 10–4. Document Revision History

Date	Version	Changes Made
June 2009	2.0	■ Updated "Boundary-Scan Description Language (BSDL) Support" on page 10–3. ■ Updated Table 10–2 and Table 10–3. ■ Made minor text edits.
December 2008	1.0	Initial release.

Section IV. Power and Thermal Management

This section includes the following chapter:

- Chapter 11, Power Supply and Temperature Sensing Diode in HardCopy IV Devices

Revision History

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

11. Power Supply and Temperature Sensing Diode in HardCopy IV Devices

HIV51011-1.1

Altera® HardCopy® IV devices and Stratix® IV devices are manufactured with the same process technology—they are based on a 0.9 V, 40 nm process. HardCopy IV devices do not use power for unused logic, memory blocks, or clock trees. Depending on resource utilization and frequency of operation, HardCopy IV power is typically reduced to 50% from the Stratix IV FPGA prototype.

Power consumption also affects thermal management. HardCopy IV devices offer a temperature sensing diode (TSD) that self-monitors the device junction temperature and that you can use with external circuitry for activities such as controlling air flow to the HardCopy IV device.

This chapter contains the following sections:

HardCopy IV Device External Power Supply Requirements

This section describes the different external power supplies needed to power HardCopy IV devices. Table 11–1 lists the external power supply pins for HardCopy IV devices. You can supply some of the power supply pins with the same external power supply, provided their supply voltage levels are the same.

Table 11–1. HardCopy IV E Power Supply Requirements (Part 1 of 2)

Power Supply Pin	Stratix IV Voltage Value (V)	HardCopy IV E Voltage Value (V)	Description
VCC	0.9	0.9	Core voltage and periphery circuitry power supply
VCCIO	1.2 / 1.5 / 1.8 / 2.5 / 3.0	1.2 / 1.5 / 1.8 / 2.5 / 3.0	I/O power supply
VCCPGM	1.8 / 2.5 / 3.0	1.8 / 2.5 / 3.0	Configuration pins power supply
VCCPD *(1)*	2.5 / 3.0	2.5 / 3.0	I/O pre-driver power supply
VCCA_PLL	2.5	2.5	PLL analog global power to the PLL regulator
VCCD_PLL	0.9	0.9	PLL digital global power supply
VCC_CLKIN	2.5	2.5	Differential clock input pins power supply (top and bottom I/O banks only)
VCCBAT	3.0	— *(4)*	Battery back-up power supply for design security volatile key register
VCCPT	1.5	— *(4)*	Power supply for programmable power technology *(2)*
VCCAUX	2.5	2.5	Power supply for the temperature sensing diode and POR
VREF	VREF *(3)*	VREF	Voltage-referenced I/O standards power supply

Table 11–1. HardCopy IV E Power Supply Requirements (Part 2 of 2)

Power Supply Pin	Stratix IV Voltage Value (V)	HardCopy IV E Voltage Value (V)	Description
GND	GND	GND	Ground

Notes to Table 11–1:

(1) VCCPD can be either 2.5 V or 3.0 V. For a 3.0-V I/O standard, VCCPD = 3.0 V. For a 2.5 V I/O standard and below, VCCPD = 2.5 V.
(2) HardCopy IV E devices do not require Programmable Power Technology.
(3) There is one V_{REF} pin per I/O bank. You can use an external power supply or a resistor divider network to supply this voltage.
(4) This power pin can be disconnected or remain connected on the board.

Table 11–2. HardCopy IV GX External Power Supply Requirements (Part 1 of 2)

Power Supply Pin	Stratix IV GX Voltage Value (V)	HardCopy IV GX Voltage Value (V)	Description
VCC	0.9	0.9	Core voltage and periphery circuitry power supply
VCCD_PLL	0.9	0.9	PLL digital power supply
VCCA_PLL	2.5	2.5	PLL analog power supply
VCCAUX	2.5	2.5	Auxiliary supply for programmable power technology
VCCPT	1.5	— *(4)*	Power supply for programmable power technology *(2)*
VCCPGM	1.8 / 2.5 / 3.0	1.8 / 2.5 /3.0	Configuration pins power supply
VCCPD	2.5 / 3.0	2.5 / 3.0	I/O pre-driver power supply
VCCIO	1.2 / 1.5 / 1.8 / 2.5 / 3.0	1.2 / 1.5 / 1.8 / 2.5 / 3.0	I/O power supply
VCC_CLKIN	2.5	2.5	Differential clock input pins power supply (top and bottom I/O banks only)
VCCBAT	1.2 – 3.0	— *(4)*	Battery back-up power supply for design security volatile key register
VREF	VCCIO / 2	VCCIO / 2	Voltage-referenced I/O standards power supply *(3)*
GND	GND	GND	Ground
VCCHIP_L	0.9	0.9	Transceiver HIP digital power (left side)
VCCHIP_R	0.9	0.9	Transceiver HIP digital power (right side)
VCCT_L	1.1	1.1	Transmitter power (left side)
VCCT_R	1.1	1.1	Transmitter power (right side)
VCCR_L	1.1	1.1	Receiver power (left side)
VCCR_R	1.1	1.1	Receiver power (right side)
VCCA_L	2.5 / 3.0	2.5 / 3.0	Transceiver high voltage power (left side)
VCCA_R	2.5 / 3.0	2.5 / 3.0	Transceiver high voltage power (right side)
VCCH_GXBL[#] *(1)*	1.4 / 1.5	1.4 / 1.5	Transceiver output buffer power for transceiver block # (left side)
VCCH_GXBR[#] *(1)*	1.4 / 1.5	1.4 /1.5	Transceiver output buffer power for transceiver block # (right side)
VCCL_GXBL[#] *(1)*	1.1	1.1	Transceiver clock power for transceiver block # (left side)

Table 11–2. HardCopy IV GX External Power Supply Requirements (Part 2 of 2)

Power Supply Pin	Stratix IV GX Voltage Value (V)	HardCopy IV GX Voltage Value (V)	Description
VCCH_GXBR[#] *(1)*	1.1	1.1	Transceiver clock power for transceiver block # (right side)

Notes to Table 11–2:

(1) The V_{CCH} and V_{CCL} powers are per transceiver block.
(2) HardCopy IV GX devices do not require Programmable Power Technology.
(3) If V_{REF} pins are not used, you must connect them to either V_{CCIO} in the same bank or GND.
(4) This power pin can be disconnected or remain connected on the board.

For possible values of each power supply, refer to the *DC and Switching Characteristics of HardCopy IV Devices* chapter in volume 4 of the *HardCopy IV Device Handbook*.

3.3-V I/O Standard Support

The maximum I/O power supply voltage for Stratix IV and HardCopy IV device families is the same because of the same process technology. Both Stratix IV and HardCopy IV devices support up to 3.3-V I/O voltage standard using a bank supply voltage (V_{CCIO}) of 3.0 V.

For more information about 3.3-V I/O standards refer to the *HardCopy IV Device I/O Features* chapter in volume 1 of the *HardCopy IV Device Handbook*.

Although Stratix IV and HardCopy IV devices support up to 3.0-V power supplies, HardCopy IV 3.0-V I/Os can properly interface with 3.3-V external ports with little loss in noise margin, given similar input and output voltage electrical characteristics.

Supporting HardCopy IV and Stratix IV Power Supplies

The core power rails in Stratix IV and HardCopy IV devices are V_{CC} and V_{CCD_PLL}. Both the Stratix IV and HardCopy IV core power rails are powered by a 0.9-V source. Table 11–3 shows the summary of core voltage requirements for these devices.

Table 11–3. Core Voltage Requirements for Stratix IV and HardCopy IV Devices

Symbol	Parameter	Stratix IV	HardCopy IV	Unit
V_{CC}	Core voltage and periphery circuitry power supply	0.9	0.9	V
V_{CCD_PLL}	PLL digital power supply	0.9	0.9	V

As Table 11–3 shows, Stratix IV-to-HardCopy IV device mapping requires all core voltages to be 0.9-V for both Stratix IV and HardCopy IV devices. Figure 11–1 shows an example of the power management of a HardCopy IV E device.

Figure 11–1. HardCopy IV E Power Management Example

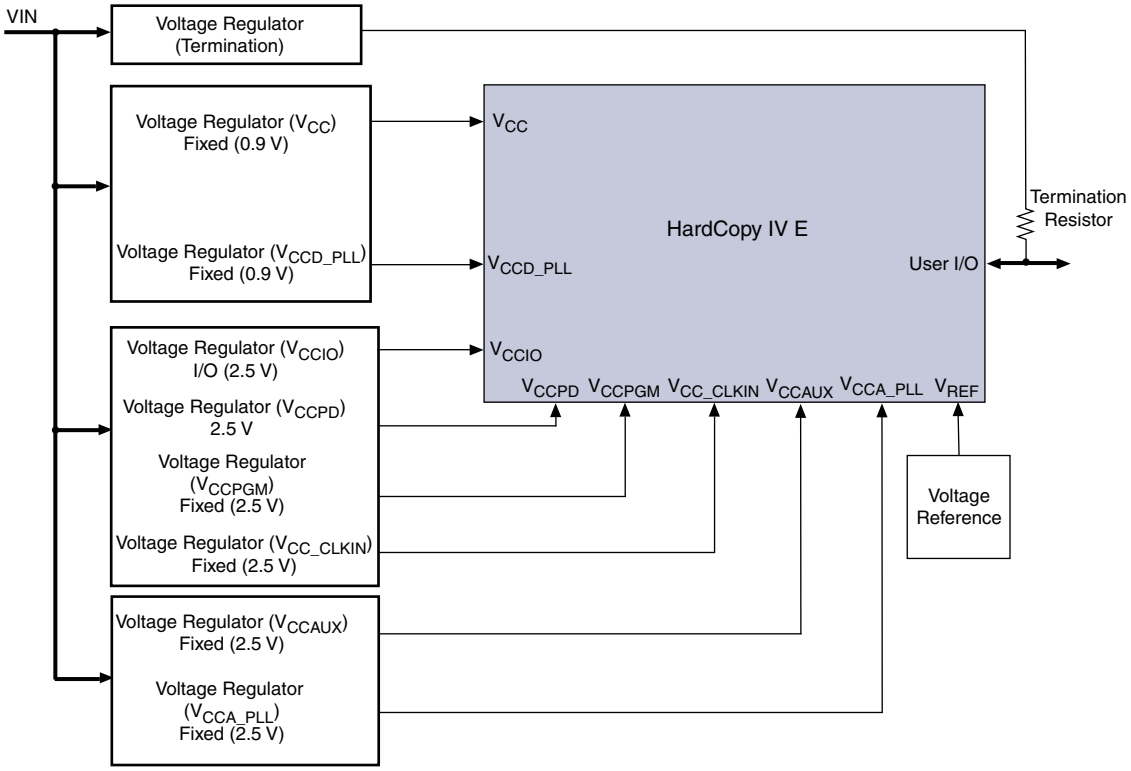
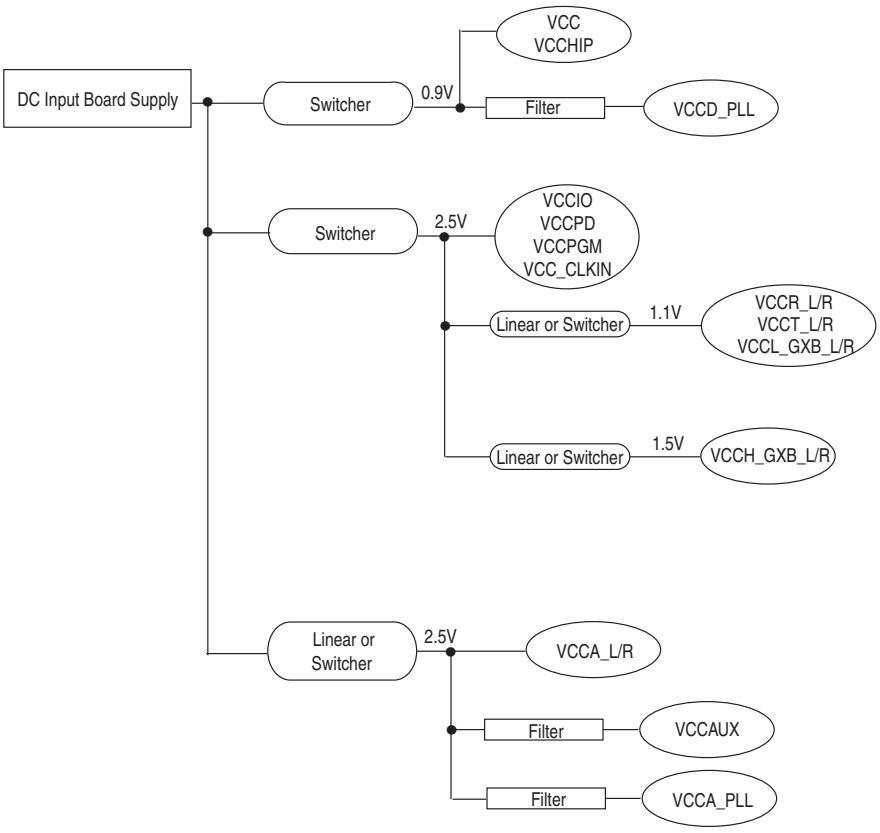


Figure 11–2 and Figure 11–3 show examples of power management in three different data rates of a HardCopy IV GX device.

Some power supplies can be combined and driven by the same linear regulator with isolation filters.

Figure 11–2. HardCopy IV GX Power Management Example (Data Rates <= 4.25 Gbps) *(Note 1)*, *(2)*

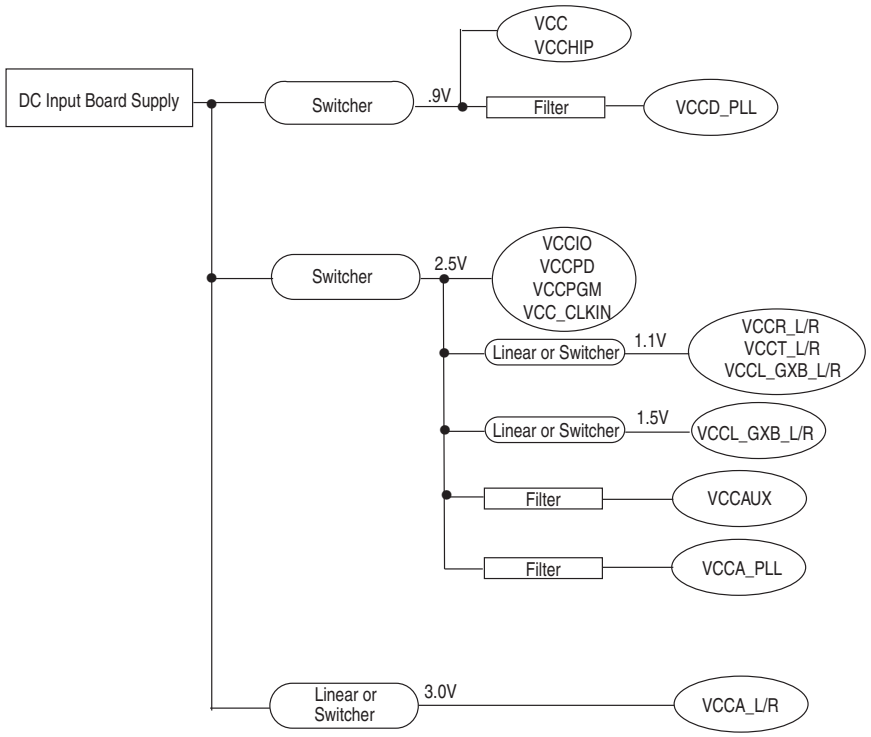


Notes to Figure 11–2:

(1) These guidelines are preliminary and are pending characterization.

(2) For best performance, Altera recommends keeping these rails isolated from each other.

Figure 11–3. HardCopy IV GX Power Management Example (Data Rates Between 4.25 Gbps and 6.5 Gbps) *(Note 1)*, *(2)*



Notes to Figure 11–3:

(1) These guidelines are preliminary and are pending characterization.

(2) For best performance, Altera recommends keeping these rails isolated from each other.

HardCopy IV Power Optimization

Because HardCopy IV devices have lower power than Stratix IV devices, HardCopy IV devices do not need programmable power technology. Therefore, this option is not needed in HardCopy IV devices. The Quartus® II software compiles your HardCopy IV design according to the timing requirements specified in the timing constraint file. Due to smaller device geometry and optimized device architecture, HardCopy IV devices generally achieve faster performance and consume less power than Stratix IV devices. Depending on resource utilization and frequency of operation, HardCopy IV core power is typically reduced 20% to 50%, when compared with Stratix IV FPGAs. Compilation reports show the power and performance of both the HardCopy IV ASIC and the Stratix IV FPGA.

Temperature Sensing Diode (TSD)

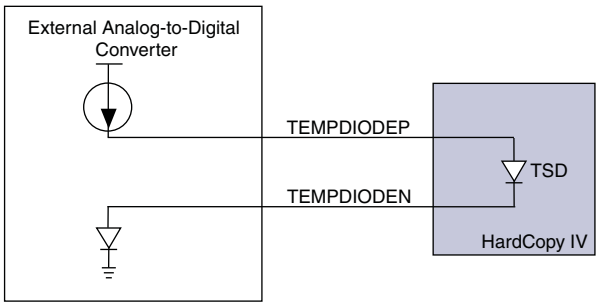
The HardCopy IV TSD uses the characteristics of a PN junction diode to determine die temperature. Knowing the junction temperature is crucial for thermal management. Junction temperature is calculated using ambient or case temperature, junction-to-ambient (θ_{JA}) or junction-to-case (θ_{JC}) thermal resistance, and the device power consumption.

A HardCopy IV device can monitor its die temperature with a TSD used with either external or embedded analog-to-digital converter (ADC) in the device. This enables you to control the air flow to the device. The ADC steers bias current through the HardCopy IV TSD, measuring forward voltage and converting this reading to temperature in the form of an 8-bit signed number (7 bits plus sign). The 8-bit output represents the junction temperature of the HardCopy IV device and can be used for intelligent power management.

External Pin Connections

The HardCopy IV TSD, located in the top-right corner of the die, requires two pins for voltage reference. You can connect the TSD with an external analog-to-digital converter (ADC) device as shown in Figure 11–4.

Figure 11–4. HardCopy IV TSD External Pin Connections



The TSD is a very sensitive circuit that can be influenced by noise coupled from other traces on the board and possibly within the device package itself, depending on device usage. The interfacing device registers temperature based on millivolts (mV) of difference, as seen at the TSD. Switching I/O near the TSD pins can affect the temperature reading. Altera recommends taking temperature readings during periods of no activity in the device.

Document Revision History

Table 11–4 shows the revision history for this chapter.

Table 11–4. Document Revision History

Date	Version	Changes Made
January 2010	1.1	■ Updated Table 11–2. ■ Updated Figure 11–2 and Figure 11–3. ■ Removed Figure 1–4 “HardCopy IV GX Power Management Example.” ■ Minor text edits.
June 2009	1.0	Initial release.

About this Handbook

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

How to Contact Altera

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

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

Note:

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

Typographic Conventions

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

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

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

HardCopy IV Device Handbook, Volume 2

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

HC4_H5V2-2.1

Contents

Chapter 2. HardCopy Design Center Implementation Process

Chapter 3. Mapping Stratix IV Device Resources to HardCopy IV Devices

Chapter 4. Matching Stratix IV Power and Configuration Requirements with HardCopy IV Devices

Additional Information

Chapter Revision Dates

The chapters in this book, *HardCopy IV Device Handbook, Volume 2*, were revised on the following dates. Where chapters or groups of chapters are available separately, part numbers are listed.

Chapter 1 HardCopy IV Design Flow Using the Quartus II Software
Revised: *January 2010*
Part Number: *HIV52001-2.1*

Chapter 2 HardCopy Design Center Implementation Process
Revised: *December 2008*
Part Number: *HIV52002-1.0*

Chapter 3 Mapping Stratix IV Device Resources to HardCopy IV Devices
Revised: *January 2010*
Part Number: *HIV52003-2.1*

Chapter 4 Matching Stratix IV Power and Configuration Requirements with HardCopy IV Devices
Revised: *June 2009*
Part Number: *HIV52004-2.0*

Section I. HardCopy IV Design Flow and Prototyping with Stratix IV Devices

This section provides a description of the design flow and the implementation process used by the HardCopy Design Center. It also provides information about mapping Stratix® IV devices to HardCopy® IV devices and associated power and configuration requirements. This section includes the following chapters:

- Chapter 1, HardCopy IV Design Flow Using the Quartus II Software
- Chapter 2, HardCopy Design Center Implementation Process
- Chapter 3, Mapping Stratix IV Device Resources to HardCopy IV Devices
- Chapter 4, Matching Stratix IV Power and Configuration Requirements with HardCopy IV Devices

Revision History

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

1. HardCopy IV Design Flow Using the Quartus II Software

HIV52001-2.1

This chapter provides recommendations for HardCopy® IV development, planning, and settings considerations in the Quartus® II software. Beginning with the Quartus II software version 9.1, both companion and compilation for HardCopy IV ASICs are supported. During HardCopy IV ASIC development, the Quartus II software ensures that both the Stratix IV FPGA and HardCopy IV ASIC have compatible pins, I/O standards, logic, and other resources.

HardCopy IV Development Flow

In the Quartus II software, two methods are available for the HardCopy IV development flow: Stratix IV device first flow and HardCopy IV device first flow.

- Stratix IV device first flow—Design the Stratix IV device first for system functional verification and then create the HardCopy IV companion device. Performing system verification early helps reduce overall total project development time.
- HardCopy IV device first flow—Design the HardCopy IV device first and then create the Stratix IV companion device for system functional verification. This method more accurately predicts the maximum performance of the HardCopy IV ASIC during development. If you optimize your design to maximize HardCopy IV ASIC performance, but are unable to meet your performance requirements with the Stratix IV FPGA, you can still map your design with decreased performance requirements for in-system verification.

Whichever design flow you choose for your HardCopy IV development, both the target design and the companion device design must be in one Quartus II project.

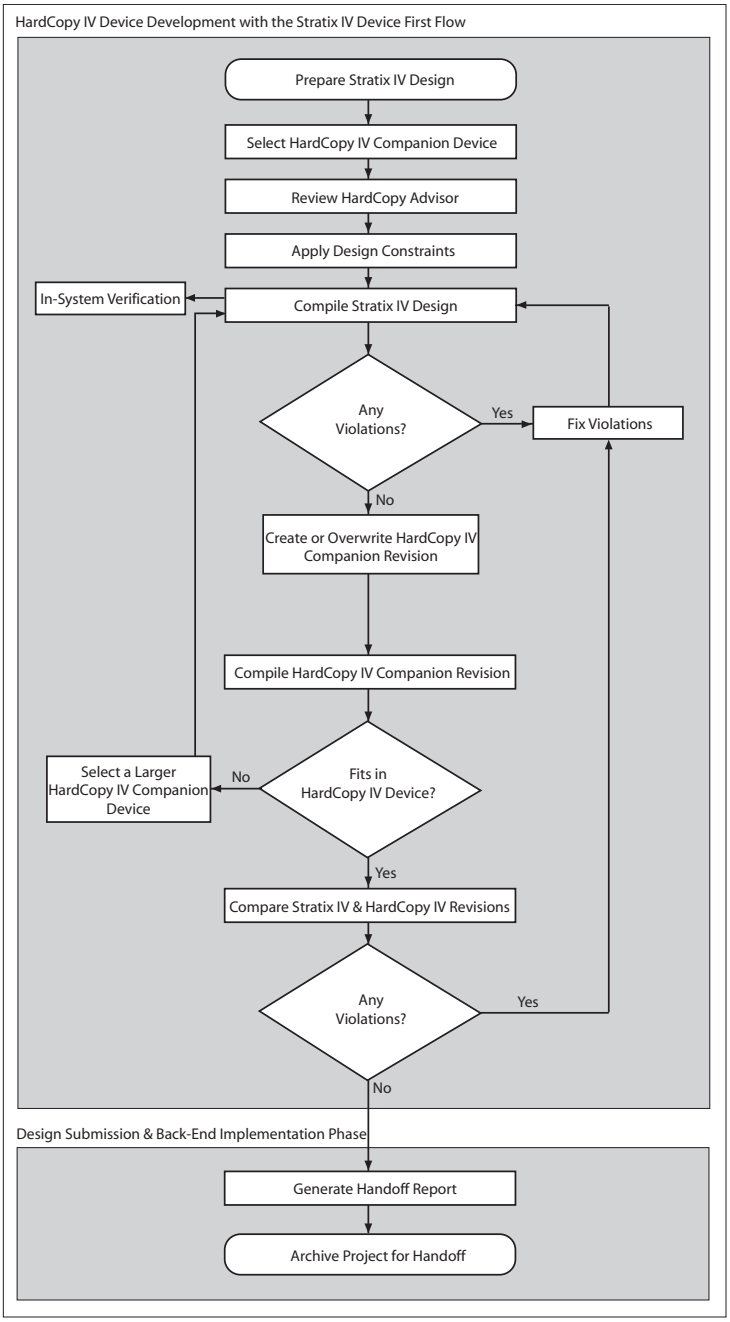
Designing with the Stratix IV Device First Flow

The HardCopy IV development flow beginning with the Stratix IV prototype is very similar to a traditional Stratix IV design flow, but requires that you perform a few additional tasks to map the design to the HardCopy IV companion device:

1. Choose a Stratix IV device for prototyping.
2. Specify a HardCopy IV device for conversion.
3. Compile the Stratix IV design.
4. Create and compile the HardCopy IV companion revision.
5. Compare the HardCopy IV companion revision compilation to the Stratix IV device compilation.
6. Generate the handoff files and reports.
7. Archive the design and send it to Altera to start the back-end design process.

Figure 1–1 shows the development process for designing with a Stratix IV device first and creating a HardCopy IV companion device second.

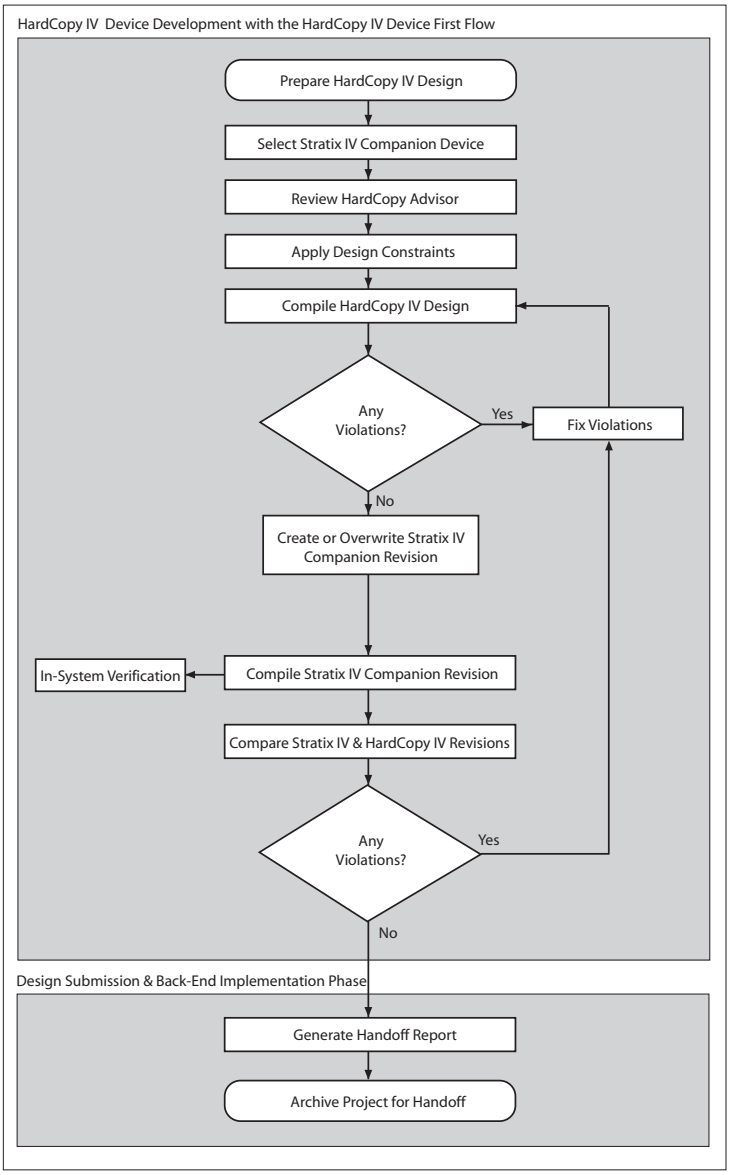
Figure 1–1. Designing with the Stratix IV Device First Flow



Designing with the HardCopy IV Device First Flow

Designing with the HardCopy IV device first flow in the Quartus II software allows you to maximize performance in the HardCopy IV device and map the design to the Stratix IV prototype for in-system verification. The performance of the Stratix IV prototype may be less than the HardCopy IV device. For this design flow, you must select **HardCopy IV** as the target device and **Stratix IV** as the companion device in the **Device Settings** dialog box. The remaining tasks required to complete your design are outlined in Figure 1–2. The HardCopy Advisor adjusts its list of tasks based on the device family you start with to guide you through the development process.

Figure 1–2. Designing with the HardCopy IV Device First Flow



HardCopy Advisor

The HardCopy Advisor in the Quartus II software plays an important role in HardCopy IV device development. The HardCopy Advisor guides you through a sequence of recommendations, descriptions, and actions. You can track your design progress, generate the design, and complete the comparison archiving and handoff file that you send to the Altera® HardCopy Design Center.

To develop the HardCopy IV design, run the HardCopy Advisor in the Quartus II software after you select the Stratix IV device and the HardCopy IV companion devices. To run the HardCopy Advisor, on the Project menu, point to **HardCopy Utilities** and click **HardCopy Advisor.**

FPGA and HardCopy Companion Device Planning

For both the HardCopy IV device and Stratix IV prototype planning, the first stage is to choose the device family, device density, speed grade, and package that best suits your design needs. Assuming a Stratix IV device first flow, select the Stratix IV device and HardCopy IV companion device based on the best resource balance for your design requirements. Perform this task before compiling your design in a third-party synthesis tool or the Quartus II software.

For information about the features available in each device density, including logic, memory blocks, multipliers, and phase-locked loops (PLLs), as well as the various package offerings and I/O pin counts, refer to volume 1 of the *HardCopy IV Device Handbook*.

Logic Resources

During HardCopy IV device planning, determine the required logic density of the Stratix IV prototype and HardCopy IV companion device. Devices with more logic resources can implement larger and potentially more complex designs. Smaller devices have less logic resources available and benefit from lower power consumption. Select a device that meets your design needs with some margin, in case you want to add more logic later in the design cycle.

For information about logic resources in HardCopy IV devices, refer to the *HardCopy IV Device Family Overview* chapter.

I/O Pin and Package Offering

HardCopy IV devices offer pin-to-pin compatibility with Stratix IV prototypes, making them drop-in replacements for FPGAs. Due to this compatibility, the same system board and software developed for the FPGA prototype can be retained, enabling faster time to market for high volume production.

To further reduce the cost for HardCopy devices, HardCopy E devices offer non-socket replacement mapping. For example, you can map the EP4SE230 device in the 780-pin FBGA package to the HC4E25 device in the 484-pin FBGA package. Because the pinout for the two packages are not the same, a separate board design is required for the Stratix IV device and the HardCopy IV E device.

For the non-socket replacement path, be sure to select I/Os in the Stratix IV E device that can be mapped to the HardCopy IV E device. Not all I/Os in the Stratix IV E device are available in the HardCopy IV E non-socket replacement device. Check the pinout information for both the Stratix IV E device and the HardCopy IV E device to ensure that you can map successfully and be sure to select the HardCopy IV E companion device when designing for the Stratix IV E device.

When mapping a specific Stratix IV device to a HardCopy IV companion device, there are a number of FPGA prototype choices.

Table 1–1 lists the Stratix IV GX FPGA-to-HardCopy IV GX ASIC mapping options; Table 1–2 lists the Stratix IV E FPGA-to-HardCopy IV E ASIC mapping options.

Table 1–1. Stratix IV GX FPGA Prototype to HardCopy IV GX ASIC Mapping Paths

HardCopy IV GX ASIC		Stratix IV GX FPGA Prototype and Package																
		EP4SGX70	EP4SGX110		EP4SGX180			EP4SGX230			EP4SGX290			EP4SGX360			EP4SGX530	
Device	Package	F780	F780	F1152	F780	F1152	F1517	F780	F1152	F1517	H780	F1152	F1517	H780	F1152	F1517	H1152	H1517
HC4GX15	780-pin FineLine BGA	✓	✓	—	✓	—	—	✓	—	—	✓ (1)	—	—	✓ (1)	—	—	—	—
HC4GX25	780-pin FineLine BGA	—	—	—	—	—	—	—	—	—	✓ (1)	—	—	✓ (1)	—	—	—	—
	1152-pin FineLine BGA	—	—	✓	—	✓	—	—	✓	—	—	✓	—	—	✓	—	✓ (1)	—
HC4GX35	1152-pin FineLine BGA	—	—	—	—	—	—	—	✓	—	—	—	—	—	✓	—	✓ (1)	—
	1517-pin FineLine BGA	—	—	—	—	—	✓	—	—	✓	—	—	✓	—	—	✓	—	✓

Note to Table 1–1:

(1) The Hybrid FBGA package requires additional unused board space along the edges beyond the footprint, but its footprint is compatible with the regular FBGA package.

Table 1–2. Stratix IV E FPGA Prototype-to-HardCopy IV E ASIC Mapping Paths

HardCopy IV E ASIC		Stratix IV E FPGA Prototype and Package						
		EP4SE230	EP4SE360		EP4SE530		EP4SE820	
Device	Package	F780	H780	F1152	H1152	H1517	H1152	H1517
HC4E25	484-pin FineLine BGA	✓ (1)	—	—	—	—	—	—
	780-pin FineLine BGA	✓	✓ (2)	—	—	—	—	—
HC4E35	1152-pin FineLine BGA	—	—	✓	✓ (2)	—	✓ (2)	—
	1517-pin FineLine BGA	—	—	—	—	✓ (2)	—	✓

Notes to Table 1–2:

(1) This mapping is a non-socket replacement path that requires a different board design for the Stratix IV E device and the HardCopy IV E device.

(2) The Hybrid FBGA package requires additional unused board space along the edges beyond the footprint, but its footprint is compatible with the regular FBGA package.

For more information about I/O features in HardCopy IV devices, refer to the *HardCopy IV I/O Features* chapter.

For more information about HardCopy IV device packages, refer to the *HardCopy IV Device Family Overview* chapter.

Memory Resources

The TriMatrix memory in HardCopy IV devices supports the same memory functions and features as Stratix IV devices. You can configure each embedded memory block to be a single- or dual-port RAM, FIFO, ROM, or shift register using the MegaWizard™ Plug-In Manager in the Quartus II software.

HardCopy IV embedded memory consists of memory logic array blocks (MLABs), M9K, and M144K memory blocks and has a one-to-one mapping from the Stratix IV memory. However, the number of available memory blocks differs based on density, package, and Stratix IV FPGA-to-HardCopy IV ASIC mapping paths.

For more information about HardCopy IV embedded memory resources, refer to the *Mapping Stratix IV Device Resources with HardCopy IV Devices* chapter.

While all three memory types are dedicated resources in Stratix IV devices, only the M9K and M144K memory blocks are dedicated resources in the HardCopy IV devices. However, the same functionality of the Stratix IV MLABs can be supported in HardCopy IV devices. The Quartus II software maps the Stratix IV MLAB function to the appropriate HCell macro that preserves the memory function. This allows the HardCopy IV core fabric to be used more efficiently, freeing up unused HCells for adaptive logic modules (ALMs) or digital signal processing (DSP) functions.

Although the memory in HardCopy IV devices supports the same memory functions and features as Stratix IV devices, you cannot pre-load or initialize HardCopy IV memory blocks with a Memory Initialization File (**.mif**) when they are used as RAM. Unlike Stratix IV devices, HardCopy IV devices do not have device configuration. The memory content of HardCopy IV devices are random after power-up. Therefore, you must ensure that your Stratix IV design does not require a **.mif** if the memory blocks are used as RAM. However, if the HardCopy IV memory block is designed as ROM, it powers up with the ROM contents.

Use the ALTMEM_INIT megafunction to initialize the RAM after power-up for HardCopy IV devices. This megafunction reads from an internal ROM (inside the megafunction) or an external ROM (on-chip or off-chip) and writes to the RAM after power-up.

When using non-registered output mode for the HardCopy IV MLABs, the output powers up with memory content. When using registered output mode for these memory blocks, the outputs are cleared on power-up. You must take this into consideration when designing logic that might evaluate the initial power-up values of the MLAB memory block.

For more information about memory blocks in HardCopy IV devices, refer to the *TriMatrix Embedded Memory Blocks in HardCopy IV Devices* chapter.

DSP Blocks Implementation

The Quartus II software uses a library of pre-characterized HCell macros to place Stratix IV DSP configurations into the HardCopy IV HCell-based logic fabric. Depending on the Stratix IV DSP configurations, the Quartus II software partitions the DSP function into a combination of DSP HCell macros in the HardCopy IV device. This optimizes the DSP function and allows the core fabric to be used more efficiently.

For more information about DSP blocks in HardCopy IV devices, refer to the *DSP Block Implementation in HardCopy IV Devices* chapter.

Clock and PLL Planning

To ensure that you map the Stratix IV design to a HardCopy IV design successfully, follow these guidelines when implementing your design. They can help make your design robust, ensuring it meets timing closure and achieves the performance you need.

For more information about the clock scheme and PLL features in HardCopy IV devices, refer to the *Clock Networks and PLLs in HardCopy IV Devices* chapter.

Clock Networks and PLL Resources

You must consider the system clocking scheme, timing requirements, and fan-out requirements during clock networks and PLL resources planning. Matching PLL resources between Stratix IV and HardCopy IV devices is determined by the design's clocking scheme and timing requirements. For high fan-out signals, use a dedicated clock resource.

In the Quartus II software, be sure the HardCopy IV companion device is selected in the device selection panel. This ensures that the PLL, other resources used, and the functions implemented in both the Stratix IV and HardCopy IV designs match. In addition, it ensures that the design converts successfully.

For more information about HardCopy IV PLL resources, refer to the *Mapping Stratix IV Device Resources with HardCopy IV Devices* chapter.

Add PLL Reconfiguration to Altera IP Blocks

Enable PLL reconfiguration for your design if it uses PLLs. The PLL settings in HardCopy IV companion devices may require different settings from the Stratix IV PLLs because of different clock tree lengths and PLL compensations. By enabling PLL reconfiguration, you can adjust your PLL settings on the HardCopy IV companion device after the silicon has been fabricated. This allows you to fine tune and further optimize your system performance.

Use Dedicated Clock Pins

During clock planning, use dedicated clock input pins for high fan-out control signals, such as asynchronous clears, presets, and clock enables for protocol signals, such as `TRDY` and `IRDY` for PCI Express (PIPE), in global or regional clock networks. These dedicated routing networks provide predictable delay and minimize skew for high fan-out signals.

Use dedicated clock pins to drive the PLL reference clock inputs, especially if the design interfaces with external memories. This minimizes the reference clock input jitter to the PLLs, providing more timing margin to make the timing closure successful. For external memory interfaces, Altera recommends using the double date rate (DDR) register in the I/O element to generate the external memory clocks.

For information about external memory interfaces, refer to the *External Memory Interfaces in HardCopy IV Devices* chapter.

Quartus II Settings for HardCopy IV Devices

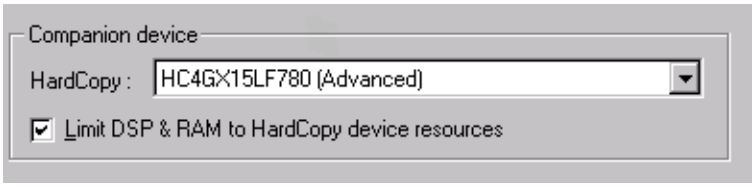
The HardCopy IV development flow requires additional Quartus II settings when compared with a typical FPGA-only design flow. This is because the HardCopy IV design is implemented in two devices: a Stratix IV prototype and a HardCopy IV companion device. You must take these settings into consideration when developing your design.

Limit DSP and RAM to HardCopy Device Resources

To maintain compatibility between the Stratix IV and HardCopy IV devices, your design must use resources that are common to both families. The Quartus II software turns on **Limit DSP & RAM to HardCopy device resources** by default when you select the Stratix IV device and HardCopy IV companion device in the Quartus II software. This prevents the Quartus II software from using resources in the Stratix IV device that are not available in the HardCopy IV device.

Figure 1–3 shows the appropriate setting to select in the **Companion device** section.

Figure 1–3. Limit DSP and RAM to HardCopy IV Device Resources Checkbox



The Altera HardCopy Design Center requires that your final Stratix IV and HardCopy IV designs be compiled with the **Limit DSP & RAM to HardCopy device resources** setting turned on before submission to the Altera HardCopy Design Center for back-end implementation.

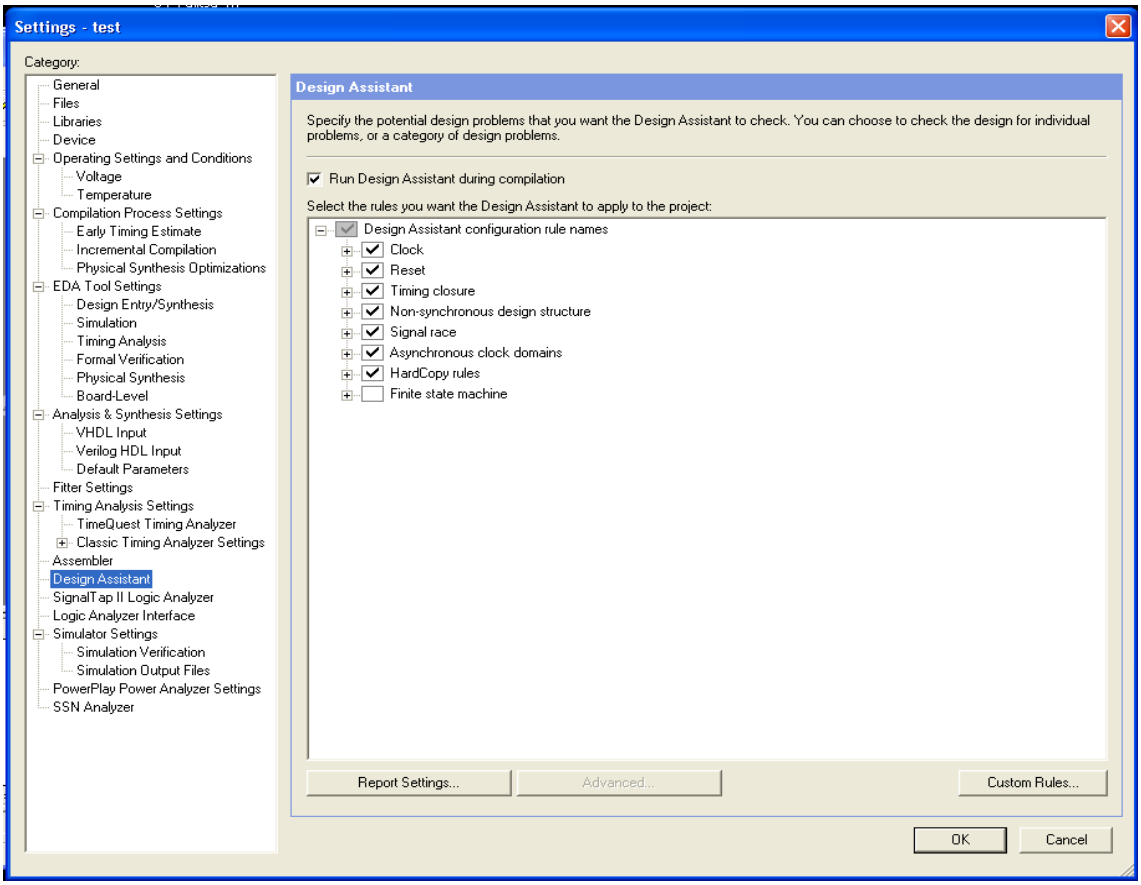
Enable Design Assistant to Run During Compile

You must use the Quartus II Design Assistant to check for design rule violations before submitting the designs to the Altera HardCopy Design Center. Additionally, you must fix all critical and high-level errors reported by the Quartus II Design Assistant. Altera recommends turning on the Design Assistant to run automatically during development.

To enable the Design Assistant to run during compilation, on the Assignments menu, click **Settings**. In the **Category** list, select **Design Assistant** and turn on **Run Design Assistant during compilation**.

Figure 1–4 shows the **Design Assistant**.

Figure 1–4. Enabling the Design Assistant



I/O Assignment Settings

Due to the complex rules governing the use of I/O cells and their availability for specific pins and packages, Altera recommends that I/O assignments be completed using the Pin Planner tool and the Assignment Editor in the Quartus II software. These tools ensure that all of the rules regarding each pin and I/O cell are applied correctly. The Quartus II software can export a **.Tcl** script containing all I/O assignments.

For more information about I/O location and type assignments using the Quartus II Assignment Editor and Pin Planner tools, refer to the *Assignment Editor* chapter in volume 2 of the *Quartus II Handbook*.

To ensure that the HardCopy IV mapping is successful, you must make accurate I/O assignments that include pin locations, I/O standards, drive strengths, and capacitance loading for the design. Ensure that the I/O assignments are compatible with all selected devices. Altera recommends not leaving any I/O with an unassigned I/O assignment.

The I/O pins of a Stratix IV device and a HardCopy IV device are arranged in groups called modular I/O banks. When mapping between a Stratix IV device and a HardCopy IV device, the I/O pin location must be assigned to the available common I/O banks for both devices. Because HardCopy IV devices have fewer I/O banks than Stratix IV devices, the Quartus II software limits the I/O banks to only those available in HardCopy IV devices.

For more information about I/O banks and pins in HardCopy IV devices, refer to the *HardCopy IV Device I/O Features* chapter.

HardCopy IV I/O buffers support 3.3-V I/O standards. You can use them as transmitters or receivers in your system. The 3.3-V I/O standard can be supported by using the bank supply voltage (V_{CCIO}) at 3.0 V. In this method, the clamp diode (on-chip or off-chip), when enabled, can sufficiently clamp overshoot voltage to within the DC and AC input voltage specification. The clamped voltage can be expressed as the sum of the supply voltage (V_{CCIO}) and the diode forward voltage.

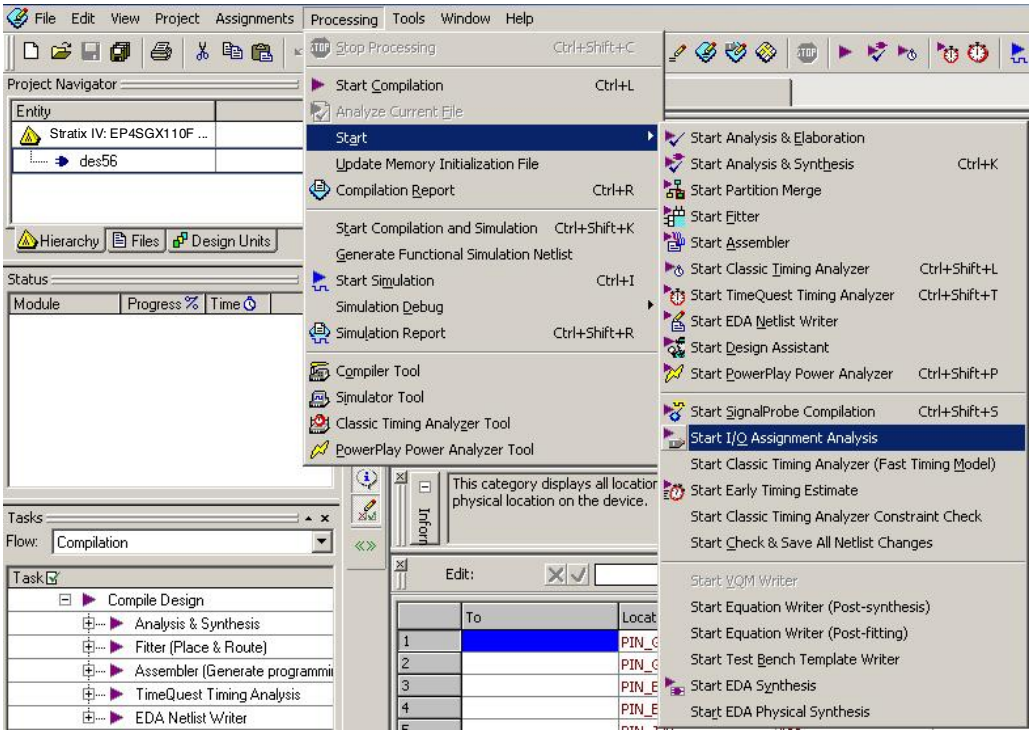
For more information about HardCopy IV I/O buffers and the standards they support, refer to the *DC and Switching Characteristics* chapter.

For more information about mapping Stratix IV to HardCopy IV I/Os, refer to the *Mapping Stratix IV Device Resources to HardCopy IV Devices* chapter.

It is essential to constrain the I/O standards for the design. If you leave the I/O with an unassigned I/O assignment, the Quartus II software assigns the I/O standard to **2.5 V** by default. This standard may not be compatible with your intended I/O standard. To check the supported I/O standards and identify incompatible I/O settings on the assigned I/Os, run the Quartus II I/O Assignment Analysis to verify the I/O settings and assignments. To run I/O Assignment Analysis, on the Processing menu, point to **Start**, then click **Start I/O Assignment Analysis**.

Figure 1–5 shows the I/O assignment analysis in the Quartus II software.

Figure 1–5. Start I/O Assignment Analysis



The default output drive strength in the Quartus II software might not be appropriate for your application. Altera recommends verifying the correct output drive strength for the design. Assigning the right output drive strength improves signal integrity while achieving timing requirements. In addition, the output capacitance loading for both the output and bidirectional pins must be set in the I/O assignment for a successful HardCopy compilation.

Physical Synthesis Optimization Settings

When you develop a HardCopy IV device with the Quartus II software, you can target physical synthesis optimizations to the FPGA architecture in Stratix IV-device first flow or the HardCopy architecture in the HardCopy IV-first flow. The optimizations in the base revision are mapped to the companion device architecture during the mapping process and the post-fitting netlists of both devices are generated and compared. Therefore, you must have the identical physical synthesis settings for both the HardCopy IV ASIC and Stratix IV FPGA revisions in order to avoid revision comparison failure.

To enable Physical Synthesis Optimizations for the Stratix IV FPGA revision of the design, on the **Assignments** menu, click **Settings**. In the **Settings** dialog box, in the **Category** list, expand **Fitter Settings**. These optimizations are passed into the HardCopy IV companion revision for placement and timing closure. When designing with a HardCopy IV device first, you can enable physical synthesis optimizations for the HardCopy IV device, and these post-fit optimizations are passed to the Stratix IV FPGA revision.

Beginning with the Quartus II v9.0 software, the **Physical Synthesis Optimizations** settings changed. If a HardCopy IV device is set as a companion device, the **Physical Synthesis Optimization** setting in the Stratix IV FPGA or HardCopy IV ASIC revision supports the **Perform physical synthesis for combination logic** and **Perform register retiming** options. In addition, the effort level of physical synthesis optimization is set to **Fast** by default.

Timing Settings

For HardCopy IV device development, you must use the TimeQuest Timing Analyzer. In the Quartus II software, TimeQuest Timing Analyzer is the default timing analyzer for Stratix IV and HardCopy IV designs. The TimeQuest Timing Analyzer guides the Quartus II Fitter and analyzes timing results during each Stratix IV and HardCopy IV design compilation.

For information about how to set the Quartus II Fitter to use timing-driven compilation, refer to "Quartus II Fitter Settings" on page 1–19.

Timing Constraints for the TimeQuest Timing Analyzer

The TimeQuest Timing Analyzer is a powerful ASIC-style timing analysis tool that validates timing in your design using industry-standard constraint, analysis, and reporting methodology. You can use the TimeQuest analyzer's GUI or command-line interface to constrain, analyze, and report results for all timing paths in your design.

Before running the TimeQuest analyzer, you must specify initial timing constraints that describe the clock characteristics, timing exceptions, signal transition arrival, and required times. You can specify timing constraints in the Synopsys Design Constraints File (**.sdc**) format using the TimeQuest analyzer GUI or the command-line interface. The Quartus II Fitter optimizes the placement of logic to meet your constraints.

The TimeQuest analyzer analyzes the timing paths in the design, calculates the propagation delay along each path, checks for timing constraint violations, and reports timing results as slack in the **Report** and **Console** panels. If the TimeQuest analyzer reports any timing violations, you can customize the reporting to view precise timing information about the specific paths, and then constrain those paths to correct the violations. When your design is free of timing violations, you can be confident that the logic will operate as intended in the target device.

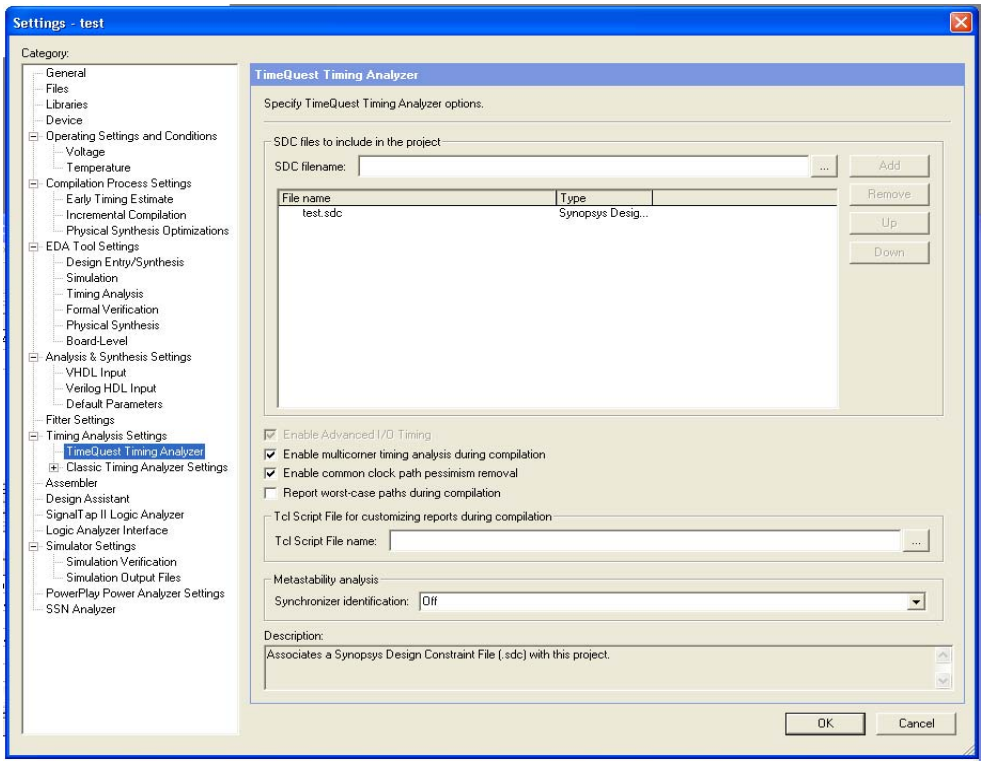
The TimeQuest analyzer is a complete static timing analysis tool that you can use as a sign-off tool for the Stratix IV design. For the HardCopy IV design, the Altera HardCopy Design Center uses the PrimeTime timing analyzer as the sign-off tool for back-end implementation.

For more information about how to create **.sdc** format timing constraints, refer to the *Quartus II TimeQuest Timing Analyzer* chapter in volume 3 of the *Quartus II Handbook*.

TimeQuest Multicorner Timing Analysis Setting

The Altera HardCopy Design Center requires that all HardCopy handoff files include a TimeQuest analyzer timing report for design review. In the TimeQuest analyzer timing report, you must include both fast- and slow-corner timing analysis for setup, hold, and I/O paths. To do this, enable the **Multicorner timing analysis during compilation** option on the **TimeQuest Timing Analyzer** page under **Timing Analysis Settings** in the Quartus II software. This option directs the TimeQuest analyzer to analyze the design and generate slack reports for the slow and fast corners. Figure 1–6 shows the settings you must enable so that the TimeQuest analyzer generates the appropriate reports.

Figure 1–6. TimeQuest Multicorner Timing Analysis Setting



Incremental Compilation

For the HardCopy development flow, the Quartus II design software offers incremental compilation to preserve the compilation results for unchanged logic in your design. This feature dramatically reduces your design iteration time by focusing new compilations only on changed design partitions. New compilation results are then merged with the previous compilation results from unchanged design partitions.

There are two approaches of incremental compilation in the Quartus II software:

- Top-down incremental compilation
- Bottom-up incremental compilation

Bottom-up incremental compilation flow is not supported for designs targeting HardCopy ASICs.

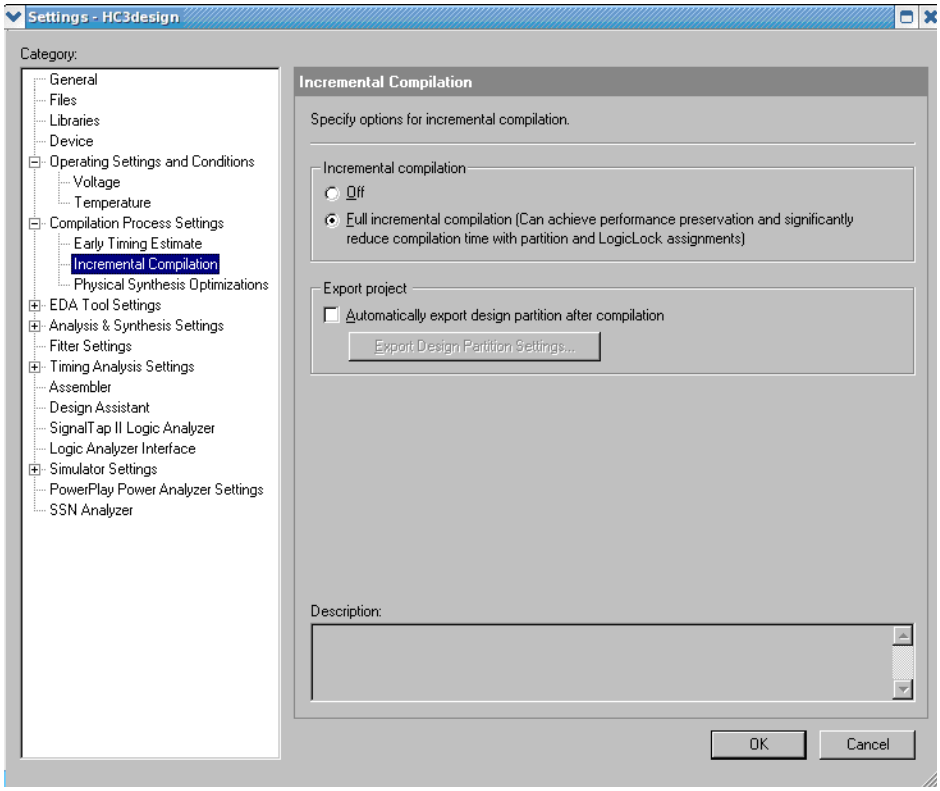
For large, high-density and high-performance designs in Stratix FPGAs and HardCopy ASICs, use top-down incremental compilation. Top-down incremental compilation facilitates team-based design environments, allowing designers to create and optimize design blocks independently. Begin planning for incremental compilation from the start of your design development. To take advantage of incremental compilation flow, split the design along any of its hierarchical boundaries into blocks called design partitions.

In the Quartus II software, the same procedures create design partitions in the HardCopy ASIC and Stratix FPGA revisions.

For more information about creating design partitions, refer to the *Quartus II Incremental Compilation for Hierarchical and Team-Based Design* chapter in volume 1 of the *Quartus II Handbook*.

In the Quartus II software, the full Incremental Compilation option is turned on by default, so the project is ready for you to create design partitions for incremental compilation. Figure 1–7 shows the full Incremental Compilation option in the Quartus II software.

Figure 1–7. Quartus II Incremental Compilation Option



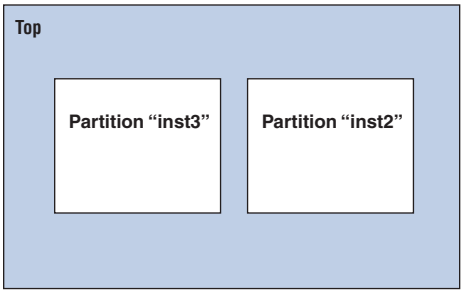
If you do not create design partitions in a design, the Quartus II software uses a flat compilation flow, and you cannot use Incremental Compilation.

Top-Down Incremental Compilation

Top-down incremental compilation is supported for the base revision for designs targeting HardCopy ASICs in both the FPGA first flow and HardCopy first flow. In the Quartus II design software, you must select the base and companion revisions before design partitions of the base revision are created.

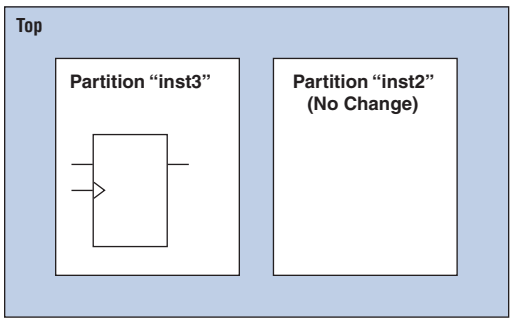
After the design partitions are created in the Quartus II design software, the base revision is compiled and the design partition assignments are mapped to the companion device. In the HardCopy development flow, you make changes only in the base revision's design and design partition assignments with the Quartus II design software. Therefore, you can perform top-down incremental compilation only in the base revision, but cannot perform incremental compilation to the companion revision. Figure 1–8 shows the design "Top" with two design partitions.

Figure 1–8. Quartus II Project with Design Partitions



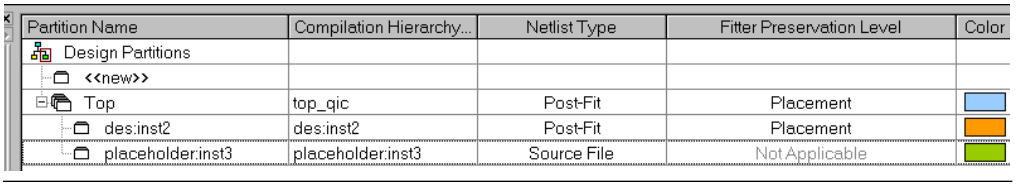
After the design partitions are created and compiled in the base family revision, you can modify the specific design partitions for additional area and performance improvement. Figure 1–9 shows that when the logic is modified in partition "inst3" the Quartus II software is ready to re-compile the individual hierarchical design partition separately, based on the preservation level in the Design Partition Window. Therefore, the optimization result of design partitions "top" and "inst2" are preserved while the partition "inst3" is re-compiled in the Quartus II design software.

Figure 1–9. Design Partition Modified



Before recompiling the design, you can set the Netlist Type in the Design Partition Window to **Source File**, **Post-Synthesis**, or **Post-Fitting** to preserve the Netlist Type of each design partition. Figure 1–10 shows the Design Partition Window with the Post-Fit preservation level for the design partitions "top" and "inst2." This allows the Quartus II design software to re-compile the design partition "inst2" from the source file, but preserves the post-fitting results of design partitions "top" and "inst3."

Figure 1–10. Design Partition Window



The Quartus II software merges the new compiled design partition "inst3" into a complete netlist for subsequent stages of the compilation flow. Design partitions "top" and "inst2" in the design do not perform incremental compilation because their logic must be preserved. Therefore, the compilation time for the overall design is reduced.

Top-Down Incremental Compilation with Empty Design Partitions

Bottom-up incremental compilation is not supported in the HardCopy development flow. During the initial stage of the design cycle, part of the design may be incomplete or developed by a different designer or IP provider. However, you can create an empty partition for this part of the design while compiling the completed partitions, and then save the results for the complete partitions while you optimize the imported part of the design.

You can often use a top-down flow with empty partitions to implement behavior similar to a bottom-up flow, as long as you do not change the global assignments between compilations. All global assignments must be the same for all compiled partitions, so the assignments can be reproduced in the companion device after mapping.

For the HardCopy development flow, create an empty partition in the base device because it cannot be created in the companion revision.

Creating an empty partition in a design is similar to creating a regular design partition in the Quartus II software. When the logic within a specific design partition is incomplete, use the following instructions to set the **Netlist Type** to **Empty**.

1. On the **Assignment** menu, click **Design Partitions Window**.
2. Double-click an entry under the **Netlist Type** column and select **Empty**.

This setting specifies that the Quartus II Compiler must use an empty placeholder netlist for the partition. Figure 1–11 shows the Empty partition setting in the Design Partition window.

Figure 1–11. Design Partition with Empty Netlist Type

Partition Name	Compilation Hierarchy...	Netlist Type	Fitter Preservation Level	Color
Design Partitions				
<<new>>				
Top	top_qic	Post-Fit	Placement	
des:inst2	des:inst2	Post-Fit	Placement	
placeholder:inst3	placeholder:inst3	Empty	Not Applicable	

When a partition Netlist Type is defined as **Empty**, virtual pins are automatically created at the boundary of the partition. This means that the software temporarily maps the I/O pins in the lower-level design entity to the internal cells instead of the pins during compilation. Any child partitions below an empty partition in the design hierarchy are also automatically treated as empty, regardless of their settings.

After the design partition with the empty Netlist Type is completed and you have defined "inst3" in the top module, the Quartus II software is ready to recompile the "inst3" design partition.

First, set the Netlist Type of the design partition "inst2" to the specific preservation target, such as **Post-Synthesis** or **Post-Fit** to preserve the performance results from the previous compilation. Before you recompile the design, ensure that you set the Netlist Type of the design partition "inst3" to **Source File** because this is new design source for the Quartus II software.

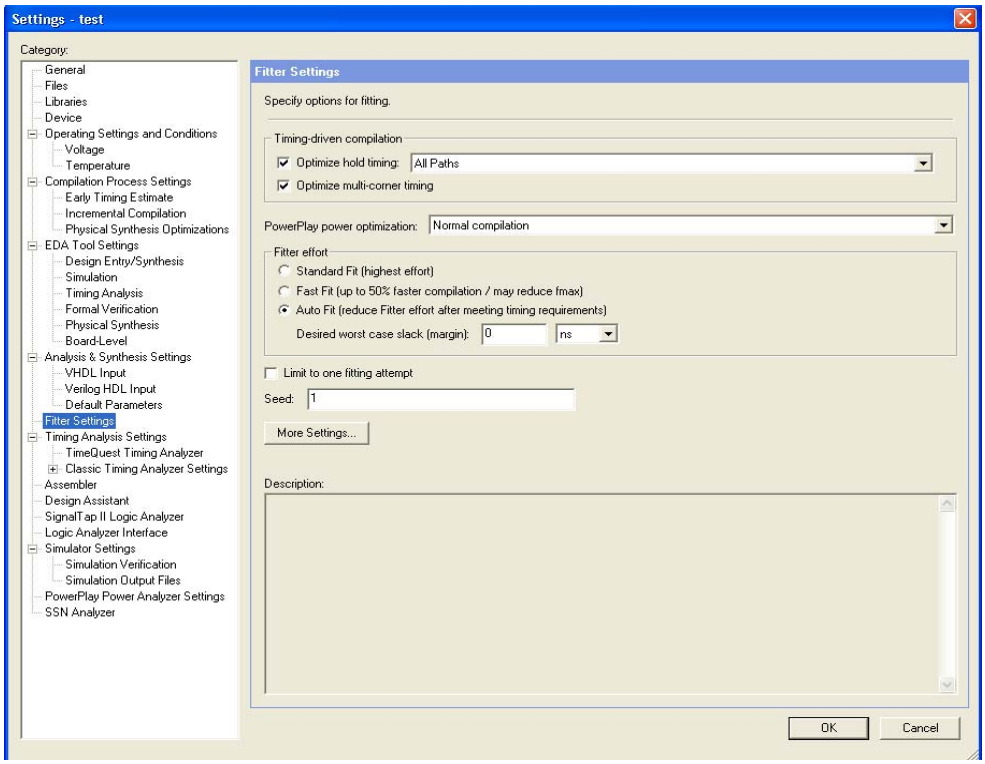
As in the traditional top-down design flow, the Quartus II software merges the new compiled design partition "inst3" into a complete netlist for the subsequent stages of the compilation flow. Therefore, the turnaround time for the design compilation is reduced.

Quartus II Fitter Settings

To make the HardCopy IV device implementation more robust across process, temperature, and voltage variations, the Altera HardCopy Design Center requires that you enable **Multicorner Optimization** for the Quartus II Fitter. This setting controls whether the Fitter optimizes a design to meet timing requirements at the fast-timing process corner and operating condition, as well as at the slow-timing process corner and operating condition. The Altera HardCopy Design Center also requires that you enable the **Optimize hold timing** setting for the Quartus II Fitter. This setting allows the Fitter to optimize hold time by adding delay to the appropriate paths.

Figure 1–12 shows the **Optimize multi-corner timing** and **Optimize hold timing** settings in the **Fitter Settings** panel.

Figure 1–12. Quartus II Fitter Settings for Optimization Multicorner and Hold Time Fix

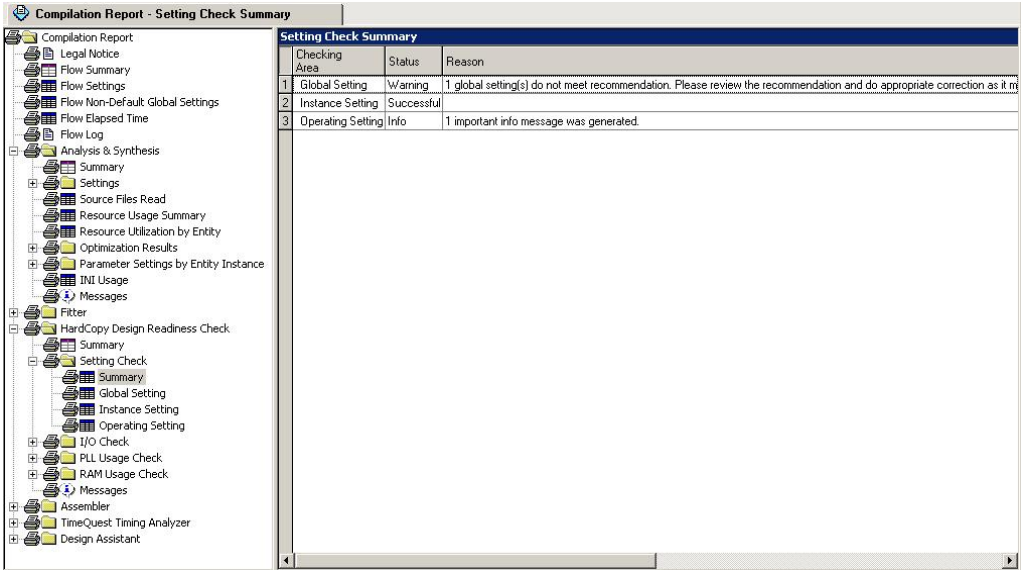


HardCopy Design Readiness Check

The HardCopy Design Readiness Check (HCDRC) feature checks issues that must be addressed before handing off the HardCopy IV design to the Altera HardCopy Design Center for the back-end implementation process. In the Quartus II software, the HCDRC includes logic checks such as PLL, RAM, and setting checks (Global Setting, Instance Setting, and Operating Setting) that were previously done in the HardCopy hand-off report. Beginning with the Quartus II software version 8.0, the default setting for running HCDRC is **On**. You can run HCDRC at post-Fitter either turned on through the Quartus Settings File (**.qsf**) or GUI.

Figure 1–13 shows the HardCopy Design Readiness Check in the Quartus II software.

Figure 1–13. HardCopy Design Readiness Check in the Quartus II Software



For more information about the HardCopy Design Readiness Check, refer to the *Quartus II Support for HardCopy Series Devices* chapter in volume 1 of the *Quartus II Handbook*.

Timing Closure and Verification

After compiling the project for the Stratix IV and HardCopy IV designs, check the device used and verify that the design meets your timing requirements. Analyze the messages generated by the Quartus II software during compilation to check for any potential problems. Also verify the design functionality between the Stratix IV and HardCopy IV devices with the **HardCopy Companion Revision Comparison** option in the Quartus II software.

Timing Closure with the TimeQuest Timing Analyzer

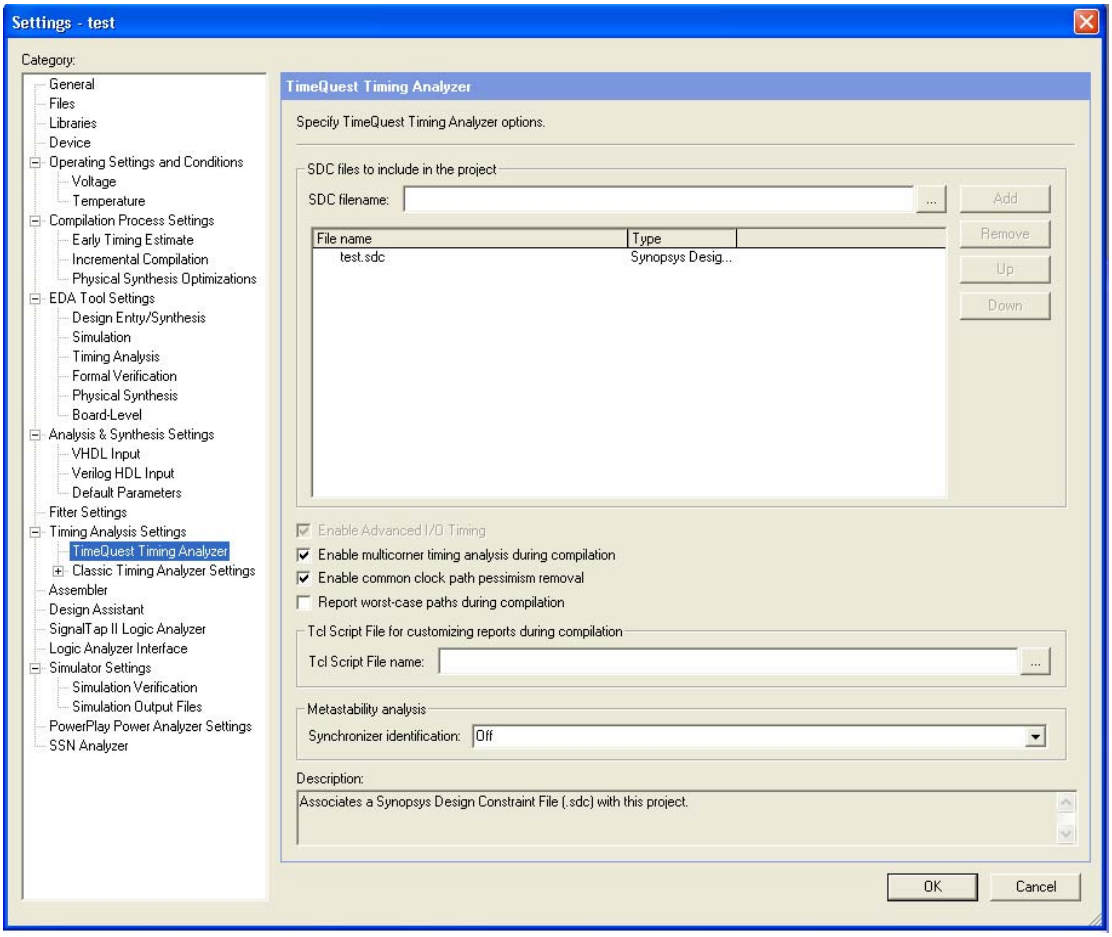
The TimeQuest Timing Analyzer is the timing analysis tool for all HardCopy IV devices during the front-end design process; it is the default timing analyzer for Stratix IV and HardCopy IV devices in the Quartus II software.

After you specify the initial timing constraints that describe the clock characteristics, timing exceptions, and signal transition arrival and required time in the **.sdc**, the TimeQuest analyzer analyzes the timing paths in the design, calculates the propagation delay along each path, checks for timing constraint violations, and reports timing results.

From the TimeQuest analyzer settings in the Quartus II software, ensure that the TimeQuest analyzer has the **Enable multicorner timing analysis during compilation** check box selected. This setting is necessary to achieve timing closure for the HardCopy IV ASIC design. By default, the TimeQuest analyzer has this setting enabled to analyze the design against best-case and worst-case operating conditions during compilation (Figure 1–14).

To direct the TimeQuest analyzer to remove the common clock path pessimism during slack computation in the Quartus II software, select the **Enable common clock path pessimism removal** option in the **TimeQuest Timing Analyzer** page (Figure 1–14).

Figure 1–14. TimeQuest Timing Analyzer Enable Multicorner Timing Analysis During Compilation and Enable Common Clock Path Pessimism Removal Options

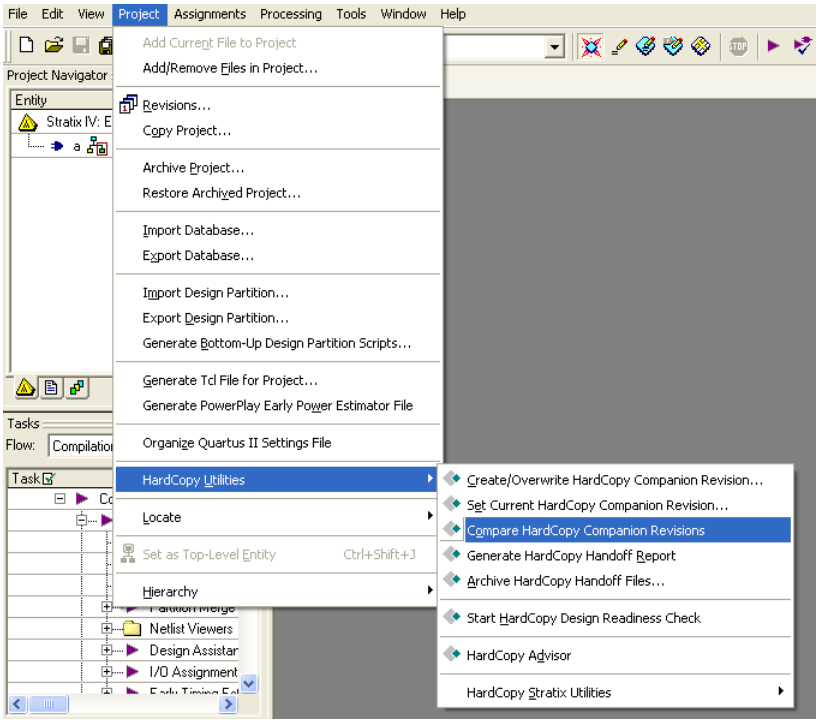


Verification

The Quartus II software uses companion revisions in a single project to promote conversion of your design from a Stratix IV FPGA to a HardCopy IV ASIC. This methodology allows you to design with one set of register transfer level (RTL) code to be used in both the Stratix IV and HardCopy IV designs, guaranteeing functional equivalency.

When making changes to your design in a companion revision, use the **Compare HardCopy Companion Revisions** feature in the Quartus II software to ensure that your Stratix IV and HardCopy IV designs match functionality and compilation settings. You must perform this comparison after both Stratix IV and HardCopy IV designs are compiled and before you hand off the design to the Altera HardCopy Design Center. Figure 1–15 shows how to navigate to the companion revisions comparison. On the Project menu, point to **HardCopy Utilities** and click **Compare HardCopy Companion Revisions**.

Figure 1–15. Compare HardCopy Companion Revisions



Engineering Change Order (ECO)

During the last stage of the design cycle, it is critical to implement a specific portion of the design, without affecting the rest of its logic. As described in the previous section, Incremental Compilation can implement and manage certain partitions of the design, and preserve the optimization results for the rest of the design. However, this becomes difficult to manage because Engineering Change Orders (ECOs) are often implemented as last-minute changes to your design.

The Quartus II software provides the Chip Planner tool and the Resource Property Editor for ECO operations to shorten the design cycle time significantly. For the HardCopy development flow, ECOs occur in the Stratix FPGA revision and you make the changes directly to the post place-and-route netlist. When you switch to the HardCopy ASIC revision, apply the same ECOs, run the timing analysis and assembler, perform a revision compare, and then run HardCopy Netlist Writer for design submission.

ECOs can be categorized in two ways for HardCopy development:

- Migrating one-to-one changes
- Migrating changes that must be implemented differently

Migrating One-to-One Changes

Some examples of migrating one-to-one changes are changes such as creating, deleting, or moving pins, changing pin or PLL properties, or changing pin connectivity (provided the source and destination of the connectivity changes are I/Os or PLLs).

To duplicate the same ECO in the Quartus II software, use the Change Manager and record all ECOs for the FPGA revision. Ensure that the same ECO operations occur on each revision for both the Stratix FPGA and HardCopy ASIC revisions to avoid a revision comparison failure.

To generate a **.tcl** script of the ECO operations in the Stratix FPGA revision and apply it to the HardCopy revision, follow these steps:

1. In the Stratix FPGA revision, open **Change Manager**.
2. On the **View** menu, click **Utility Windows** and select **Change Manager**.
3. Perform the ECO in the **Chip Planner** or **Resource Property Editor**. You will see the ECO operations in the **Change Manager**.

For the information about ECO operations in the Stratix FPGA revision, refer to the *Engineering Change Management with the Chip Planner* chapter in volume 2 of the *Quartus II HandBook*.

4. Export the ECO operations from the **Change Manager** to **Tcl Script**. On the **Change Manager**, right mouse click the entry. Click **Export**, and then click **Export All Changes AS…**
5. Save the **.tcl** script, to be used in the HardCopy revision.

In the HardCopy revision, apply the **.tcl** script to the companion revision using the following procedure.

1. Open the generated **.tcl** script from the Quartus II software or a text editor tool. Edit the line `project_open` <*project*> `- revision` <*revision*> to refer to the appropriate companion revision. Save the **.tcl** script.
2. Apply the **.tcl** script to the companion revision. On **Tools** menu, scroll the **Tcl Scripts** pull-down menu, and select **ECO Tcl** and click **Run**.

Migrating Changes that Must be Implemented Differently

Unlike migrating changes one-to-one, some changes must be implemented on the Stratix FPGA and HardCopy ASIC revisions differently. Changes affecting the logic of the design can fall into this category. Examples of these are LUTMASK changes, LC_COMB/HSADDER creation and deletion, and connectivity changes not covered in the previous section.

For a summary of suggested implementation changes, refer to the *Quartus II Support for HardCopy Series Devices* chapter in volume 1 of the *Quartus II Handbook*.

PLL settings are another example of implementing changes differently on the Stratix FPGA and the HardCopy ASIC revisions. Sometimes PLL-generated clocks must be modified to provide a higher-frequency clock in HardCopy devices to improve the performance of the HardCopy device without changing the performance of the Stratix FPGA. You must handle the modification correctly so that the HardCopy Companion Revision Comparison utility does not generate critical errors.

To set different PLL settings for the Stratix FPGA and HardCopy revisions, you must have different PLL source files. Each PLL must have the same module name, so that the same PLL in both revisions is not treated differently during the HardCopy Revision Comparison stage. You must have two PLL files with different names that reference the same module.

When starting HardCopy development with the FPGA first flow to override the file naming convention so that the same PLL module can be referenced by two different PLL files, complete the following steps:

1. Specify the PLL source file in the QSF file in the Stratix FPGA revision. For example, `set_global_assignment -name MIGRATION_DIFFERENT_SOURCE_FILE pll_fgpa.v`

   Note that when there are multiple PLL source files, you must use multiple assignments to specify the PLL source files.

2. Compile the Stratix FPGA revision in the Quartus II software.
3. After creating the HardCopy revision, modify the PLL source file manually or with the MegaWizard Plug-In Manager in order to improve the performance in the HardCopy revision.

   When the MegaWizard Plug-In Manager updates the new source file, it modifies the top-level name of the module or entity in the source file to match the name of the source file. Therefore, you must rename the module or entity after you have updated the file with the MegaWizard Plug-In Manager so that your top-level design instantiates the PLL with the newly modified PLL design file.

4. After updating the PLL source file in the HardCopy revision, verify that the QSF source file setting contains the newly modified PLL source file. For example, `set_global_assignment -name VERILOG_FILE pll_hc.v`
5. Compile the HardCopy revision in the Quartus II software.

After compilation is completed, run the **HardCopy companion Revision Comparison** utility to observe and track the changes made to the PLLs and design settings. These changes are captured as critical warnings in the revision comparison report and must be reviewed by the HardCopy Design Center before the design is accepted for mapping.

HardCopy IV Handoff Process

To submit a design to the Altera HardCopy Design Center for design review and back-end implementation, generate a HardCopy IV handoff report and archive the HardCopy IV project.

Before you generate the HardCopy IV handoff report, you must first successfully perform the following tasks:

- Compile both Stratix IV and HardCopy IV revisions of the design.
- Run the **Compare HardCopy Companion Revision** utility.

Archive the HardCopy IV project and submit it to the Altera HardCopy Design Center for back-end implementation. This is the last step in the HardCopy IV design flow. The HardCopy IV archive utility creates a different Quartus II Archive File (**.qar**) than the standard Quartus II project archive utility generates. This archive contains only the data from the Quartus II project needed to implement the design in the Altera HardCopy Design Center.

To use the **Archive HardCopy Handoff Files** utility, you must perform all tasks for generating a HardCopy IV handoff report.

After you generate a HardCopy IV handoff report, select the handoff option. On the Project menu, point to **HardCopy Utilities** and click **Archive HardCopy Handoff Files**.

The **Archive HardCopy Handoff Files** utility archives your design, settings, results, and database files for delivery to Altera. These files are generated at the same directory level as the targeted project created with an **_hc** extension.

Document Revision History

Table 1–3 lists the revision history for this chapter.

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

Date	Version	Changes Made
January 2010	2.1	■ Updated Table 1–2. ■ Added "Physical Synthesis Optimization Settings" on page 1–13. ■ Added "Incremental Compilation" on page 1–15. ■ Added "Engineering Change Order (ECO)" on page 1–23. ■ Minor text edits.
June 2009	2.0	■ Updated Table 1–1. ■ Removed Tables 1-2 and 1-3. ■ Added several references to other handbook chapters. ■ Minor text edits.

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

May 2009	1.1	■ Updated Table 1–3. ■ Updated Figure 1–1, Figure 1–2, Figure 1–4, Figure 1–6, Figure 1–7, and Figure 1–9. ■ Updated the "FPGA and HardCopy Companion Device Planning", "I/O Pin and Package Offering", "I/O Assignment Settings", "Timing Settings" and "Timing Closure with the TimeQuest Timing Analyzer" sections. ■ Removed the Setting Up the TimeQuest Timing Analyzer, Reference Documents, and Conclusion sections.
December 2008	1.0	Initial release.

2. HardCopy Design Center Implementation Process

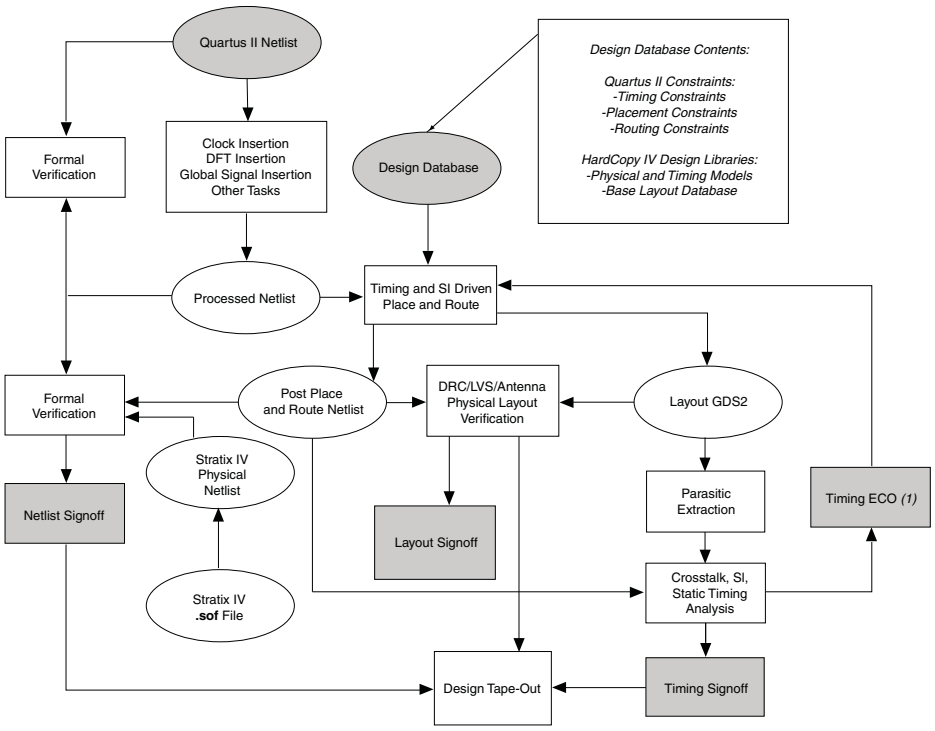
HIV52002-1.0

This chapter discusses the HardCopy® IV back-end design flow executed by the Altera® HardCopy Design Center when developing your HardCopy IV device.

HardCopy IV Back-End Design Flow

This section outlines the back-end design process for HardCopy IV devices. Figure 2–1 illustrates these steps. The design process uses both proprietary and third-party EDA tools.

Figure 2–1. HardCopy IV Back-End Design Flow



Note to Figure 2–1

(1) Refer to Figure 2–2 for more information about the timing ECO.

Design Netlist Generation

For HardCopy IV designs, the Quartus® II software generates a complete Verilog gate-level netlist of your design. The HardCopy Design Center uses the netlist to start the back-end process. In addition to the Verilog gate-level netlist, the Quartus II software generates information as part of the design database submitted by you to the Altera HardCopy Design Center. This information includes timing constraints, placement constraints, and global routing information. Generation of this database provides the HardCopy Design Center with the necessary information to complete the design of your HardCopy IV device.

Design for Testability

The HardCopy Design Center inserts the necessary test structures into the HardCopy IV Verilog netlist. These test structures include full-scan capable registers and scan chains, JTAG, and memory testing. After adding the test structures, the modified netlist is verified using third-party EDA formal verification software against the original Verilog netlist to ensure that the test structures have not broken your netlist functionality. "Formal Verification of the Processed Netlist" on page 2–2 explains the formal verification process.

Clock Tree and Global Signal Insertion

Along with test insertion, the HardCopy Design Center adds a local layer of clock tree buffering to connect the global clock resources to the locally placed registers in the design. Global signals with high fan-out can also use dedicated global clock resources built into the base layers of all HardCopy IV devices. The HardCopy Design Center does local buffering.

Tie-Off Connections for Unused Resources

If an unused resource in a customer design still exists in the HardCopy IV database, the HardCopy Design Center uses special handling on the tie-off connections for these resources. I/O ports of unused resources are connected to power or ground so that the resources are in a lower power state. This is achieved by using the same metal layers that are used to configure and connect all resources used in the design.

Formal Verification of the Processed Netlist

After all design-for-testability logic, clock tree buffering, global signal buffering, and tie-off connection are added to the processed netlist, the HardCopy Design Center uses third-party EDA formal verification software to compare the processed netlist with your submitted Verilog netlist generated by the Quartus II software. Added test structures are constrained to bypass mode during formal verification to verify that your design's intended functionality is unchanged.

Timing and Signal Integrity Driven Place and Route

Placement and global signal routing is principally done in the Quartus II software before submitting the HardCopy IV design to the HardCopy Design Center. With the Quartus II software, you control the placement and timing driven placement optimization of your design. The Quartus II software also does global routing of your signal nets, and passes this information in the design database to the HardCopy Design Center to do the final routing. After the design is submitted, Altera engineers use the placement and global routing information provided in the design database to do final routing and timing closure, and to perform signal integrity and crosstalk analysis. This may require buffer and delay cell insertion in the design through an engineering change order (ECO). The resulting post place and route netlist is verified again with the source netlist and the processed netlist to guarantee that functionality was not altered in the process. For more details about back-end timing closure and timing ECOs, refer to "Back-End Timing Closure" and "Timing ECOs".

Parasitic Extraction and Timing Analysis

After the HardCopy Design Center places and routes your design, a **.gds2** design file is generated. Parasitic extraction uses the physical layout of the design stored in the database to extract the resistance and capacitance values for all signal nets in the design. The HardCopy Design Center uses these parasitic values to calculate the path delays through the design for static timing analysis and crosstalk analysis.

Back-End Timing Closure

The Quartus II software provides a pre-layout estimation of your HardCopy IV design performance. The Altera HardCopy Design Center then uses industry leading EDA software to complete the back-end layout and extract the final timing results prior to tape-out. Altera performs rigorous timing analysis on the HardCopy IV design during its back-end implementation, ensuring that it meets the required timing constraints. After generating the customized metal interconnect for the HardCopy IV device, Altera checks the design timing with a static timing analysis tool. The static timing analysis tool may report timing violations, which are reviewed with the customer.

The critical timing paths of the HardCopy IV device may be different from the corresponding paths in the Stratix IV FPGA revision; these differences can exist for several reasons. While maintaining the same set of features as the corresponding Stratix IV FPGA, HardCopy IV devices have a highly optimized die size to make them as small as possible. Because of the customized interconnect structure that makes this optimization possible, the delay through each signal path is different from the original Stratix IV FPGA design. Therefore, it is important to constrain the Stratix IV FPGA and HardCopy IV devices to the exact, system-level timing requirements that need to be achieved. Timing violations seen in the Quartus II project or in the HardCopy Design Center back-end process must be fixed or waived prior to the design tape-out.

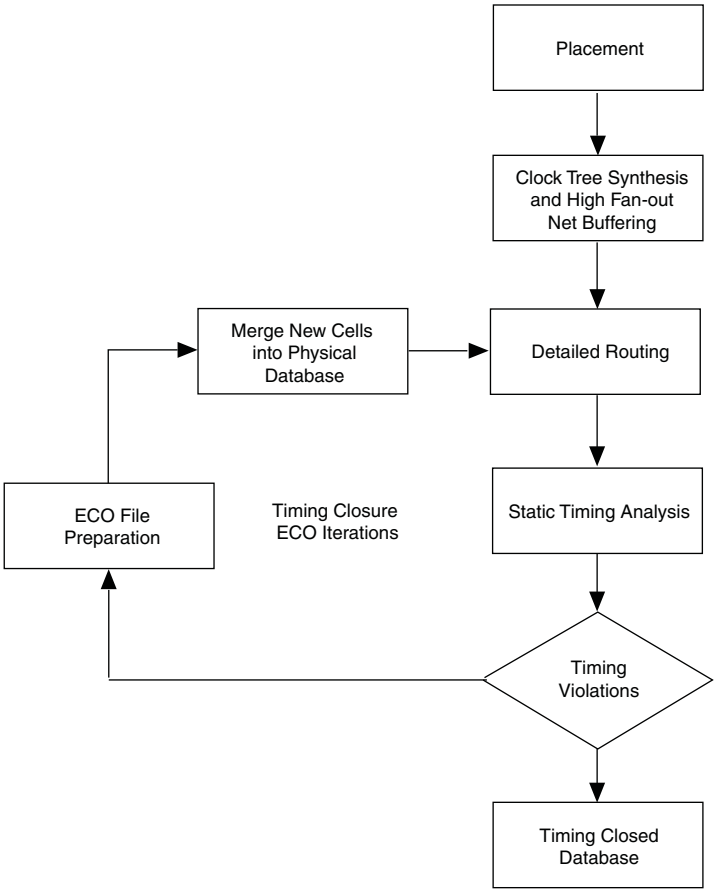
Timing ECOs

In an ASIC design, small incremental changes to a design database are termed ECOs. In the HardCopy IV design flow, timing closure ECOs are performed by Altera's HardCopy Design Center after the initial post-layout timing data is available.

The Altera HardCopy Design Center runs static timing analysis on the design. This analysis may show that the place and route tool was not able to close timing automatically on some paths. The HardCopy Design Center engineer will determine the best way to fix the timing on these paths (for example, by adding delay cells to fix a hold time violation). This list of changes is fed back into the place and route tool which subsequently implements the changes. The impact to the place and route database is minimized by maintaining all of the pre-existing placement and routing, and only changing the paths that need improvement.

The parasitic resistances and capacitances of the customized interconnect are extracted, and are used in conjunction with the static timing analysis tool to re-check the timing of the design. Detected crosstalk violations on signals are fixed by adding additional buffering to increase the setup or hold margin on victim signals. In-line buffering and small buffer tree insertion is done for signals with high fanout, high transition times, or high capacitive loading. Figure 2–2 shows this flow in more detail.

Figure 2–2. Timing Closure ECO Flow Diagram



The back-end flow produces the final signoff timing for your HardCopy IV device. The Quartus II software produces the timing report for HardCopy IV based on global routing and does not factor in the exact physical parasitic of the routed nets. The Quartus II software also does not factor in the crosstalk effect that neighboring nets can have on interconnect capacitance.

Formal Verification of the Post-Layout Netlist

Besides the **.gds2** file and parasitic files that are generated by the HardCopy Design Center, the post-layout netlist is also generated for formal verification with Stratix IV FPGAs. The HardCopy Design Center checks the functional equivalence between the Stratix IV FPGA prototype and HardCopy IV device according to the Stratix IV **.sof** file and HardCopy IV post-layout netlist.

Layout Verification

When the Timing Analysis reports that all timing requirements are met, the design layout goes into the final stage of verification for manufacturability. The HardCopy Design Center performs physical Design Rule Checking (DRC), antenna checking of long traces of signals in the layout, and a comparison of layout to the design netlist, commonly referred to as Layout Versus Schematic (LVS). These tasks guarantee that the layout contains the exact logic represented in the place-and-route netlist and the physical layout.

Design Signoff

The Altera HardCopy IV back-end design methodology has a thorough verification and signoff process, guaranteeing your design's functionality. Signoff occurs after completing the final place-and-route netlist functional verification, layout verification for manufacturability, and timing analysis. After achieving all three signoff points, Altera begins the manufacturing of the HardCopy IV devices.

Conclusion

Altera's back-end design methodology ensures that your design converts successfully from your Stratix IV FPGA prototype to the HardCopy IV ASIC. Altera's unique system development methodology offers an excellent way for you to benefit from using a Stratix IV FPGA for design prototyping and debugging, and using a HardCopy IV ASIC for volume production.

Document Revision History

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

Table 2–1. Document Revision History

December 2008	1.0	Initial release.

3. Mapping Stratix IV Device Resources to HardCopy IV Devices

HIV52003-2.1

This chapter discusses the available options for mapping from a Stratix® IV device to a HardCopy® IV device.

The Quartus II software limits resources to those available to both the Stratix IV FPGA and the HardCopy IV ASIC. It also ensures that the design revision targeting a HardCopy IV device retains the same functionality as the original Stratix IV design.

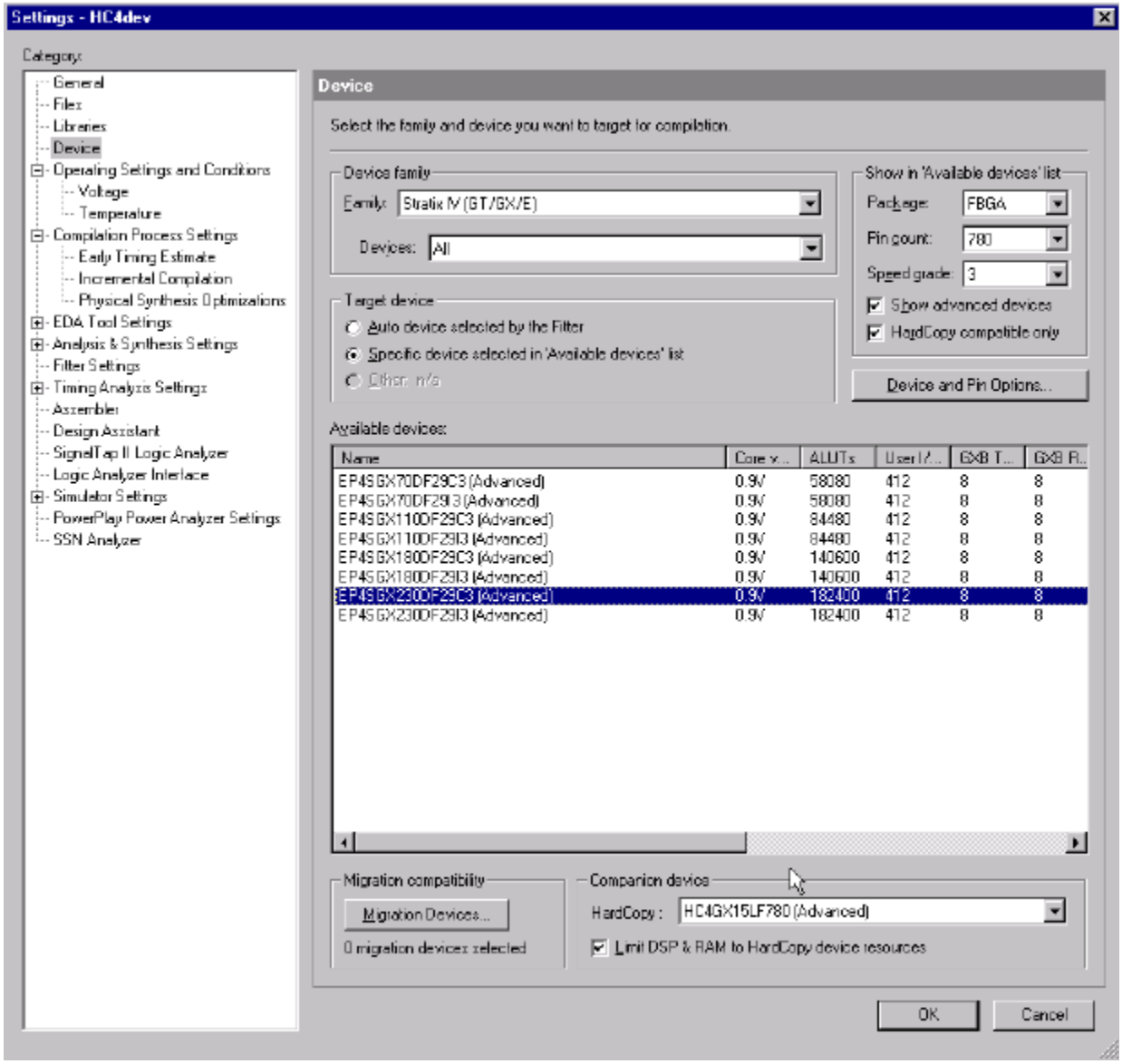
When compiling designs with the Quartus II software, you can specify one Stratix IV target device and one or more Stratix IV mapping devices. When you specify at least one mapping device, the Quartus II compiler constrains I/O pins and relevant hard IP blocks to the minimum resources available in any of the selected mapping devices. This feature allows vertical mapping between devices using the same package footprint.

Selecting a HardCopy IV device as a companion device is similar to adding another Stratix IV device to the mapping device chain. The Quartus II software compiles the design to use the common resources available in all of the selected Stratix IV and HardCopy IV devices.

The HardCopy IV companion device becomes the target device when you create the HardCopy Companion revision.

Figure 3–1 shows the **Device** page of the **Settings** dialog box, where you choose the companion device for the target device selected. The **Device** panel lists appropriate companion devices based on the target device you select.

Figure 3–1. Quartus II Device Settings Page with HardCopy IV Device Selected as Companion Device



When you select a HardCopy IV companion device, the Quartus II software fits your design to common resources in the I/Os, clock structures, PLLs, memory blocks, and core logic for digital signal processing (DSP).

For more information about compiling with Stratix IV and HardCopy IV companion revisions using the Quartus II software, refer to the *Quartus II Support for HardCopy Series Devices* chapter in volume 1 of the *Quartus II Handbook*.

HardCopy IV and Stratix IV Mapping Options

HardCopy IV ASICs offer a wide range of family options that can map with various Stratix IV FPGAs.

Table 3–1 and Table 3–2 lists the available HardCopy IV and Stratix IV companion pairs.

Table 3–1. HardCopy IV GX and Stratix IV GX Companion Devices

Companion Pair		HardCopy IV Package
HardCopy IV GX ASIC	**Stratix IV GX FPGA Prototype**	
HC4GX15LAF780N	EP4SGX70DF29 (F780)	780-pin FineLine BGA
	EP4SGX110DF29 (F780)	
	EP4SGX180DF29 (F780)	
	EP4SGX230DF29 (F780)	
HC4GX15LF780N	EP4SGX290FH29 (H780)	
	EP4SGX360FH29 (H780)	
HC4GX25LF780N	EP4SGX290FH29 (H780)	780-pin FineLine BGA
	EP4SGX360FH29 (H780)	
HC4GX25LF1152N	EP4SGX110FF35 (F1152)	1152-pin FineLine BGA
	EP4SGX180FF35 (F1152)	
	EP4SGX230FF35 (F1152)	
	EP4SGX290FF35 (F1152)	
	EP4SGX360FF35 (F1152)	
HC4GX25FF1152N	EP4SGX180HF35 (F1152)	1152-pin FineLine BGA
	EP4SGX230HF35 (F1152)	
	EP4SGX290HF35 (F1152)	
	EP4SGX360HF35 (F1152)	
	EP4SGX530HH35 (H1152)	
HC4GX35FF1152N	EP4SGX230HF35 (F1152)	1152-pin FineLine BGA
	EP4SGX360HF35 (F1152)	
	EP4SGX530HH35 (H1152)	
HC4GX35FF1517N	EP4SGX180KF40 (F1517)	1517-pin FineLine BGA
	EP4SGX230KF40 (F1517)	
	EP4SGX290KF40 (F1517)	
	EP4SGX360KF40 (F1517)	
	EP4SGX530KH40 (H1517)	

Table 3–2. HardCopy IV E and Stratix IV E Companion Devices

Companion Pair		HardCopy IV Package
HardCopy IV E ASIC	**Stratix IV E FPGA Prototype**	
HC4E25WF484N *(1)*	EP4SE230F29 (F780)	484-pin FineLine BGA Wire Bond
HC4E25FF484N *(1)*	EP4SE230F29 (F780)	484-pin FineLine BGA
HC4E25WF780N	EP4SE230F29 (F780)	780-pin FineLine BGA Wire Bond
	EP4SE360H29 (H780)	
HC4E25FF780N	EP4SE230F29 (F780)	780-pin FineLine BGA
	EP4SE360H29 (H780)	
HC4E35LF1152N	EP4SE360F35 (F1152)	1152-pin FineLine BGA
	EP4SE530H35 (H1152)	
	EP4SE820H35 (H1152)	
HC4E35FF1152N	EP4SE360F35 (F1152)	1152-pin FineLine BGA
	EP4SE530H35 (H1152)	
	EP4SE820H35 (H1152)	
HC4E35LF1517N	EP4SE530H40 (H1517)	1517-pin FineLine BGA
	EP4SE820H40 (H1517)	
HC4E35FF1517N	EP4SE530H40 (H1517)	1517-pin FineLine BGA
	EP4SE820H40 (H1517)	

Note to Table 3–2:

(1) This mapping is a non-socket replacement path that requires a different board design for the Stratix IV E device and the HardCopy IV E device. The Stratix IV E device is in a 780-pin FBGA package while the HardCopy IV E device is in a 484-pin FBGA package.

When the Quartus II software successfully compiles a design, the HardCopy Device Resource Guide in the Fitter Compilation Report contains information about mapping compatibility to a HardCopy IV device. Use this information to select the optimal HardCopy IV device for the prototype Stratix IV device based on resource and package requirements.

Table 3–3 and Table 3–4 show the available resources for prototyping on a Stratix IV device when choosing a HardCopy IV device.

Table 3–3. HardCopy IV GX ASIC Features

HardCopy IV GX ASIC	Stratix IV GX FPGA Prototype	ASIC Equivalent Gates *(1)*	Transceivers 6.5+Gbps *(2)*	M9K Blocks	M144K Blocks	Total Dedicated RAM Bits (not including MLABs) *(3)*	18 x 18-Bit Multipliers (FIR Mode)	PLLs
HC4GX15	EP4SGX70	2.8 M	8, 0	462	16	6,462 Kb	384	3
	EP4SGX110	3.8 M	8, 0	660	16	8,244 Kb	512	3
	EP4SGX180	6.7 M	8, 0	660	20	8,820 Kb	920	3
	EP4SGX230	9.2 M	8, 0	660	22	9,108 Kb	1288	3
	EP4SGX290	7.7 M	8, 0	660	24	9,396 Kb	832	2
	EP4SGX360	9.4 M	8, 0	660	24	9,396 Kb	1040	2
HC4GX25	EP4SGX110	3.8 M	16, 0	660	16	8,244 Kb	512	4
	EP4SGX180	6.7 M	16, 8 *(6)*	936	20	11,304 Kb	920	6
	EP4SGX230	9.2M	16, 8 *(6)*	936	22	11,592 Kb	1288	6
	EP4SGX290	7.7 M	16, 8 *(6)*	936	36	13,608 Kb	832	6 *(4)*
	EP4SGX360	9.4 M	16, 8 *(6)*	936	36	13,608 Kb	1040	6 *(4)*
	EP4SGX530	11.5 M	16, 8	936	36	13,608 Kb	1024	6
HC4GX35	EP4SGX180	6.7 M	24, 12 *(7)*	950	20	11,430 Kb	920	8
	EP4SGX230	9.2 M	24, 12 *(7)*	1235	22	14,283 Kb	1288	8 *(5)*
	EP4SGX290	7.7 M	24, 12 *(7)*	936	36	13,608 Kb	832	8
	EP4SGX360	9.4 M	24, 12 *(7)*	1248	48	18,144 Kb	1040	8 *(5)*
	EP4SGX530	11.5 M	24, 12 *(7)*	1280	64	20,736 Kb	1024	8 *(5)*

Notes to Table 3–3:

(1) This is the number of ASIC-equivalent gates available in the HardCopy IV GX base array, shared between both adaptive logic module (ALM) logic and DSP functions from a Stratix IV GX FPGA prototype. The number of usable ASIC equivalent gates is bounded by the number of ALMs in the companion Stratix IV GX FPGA device.

(2) The first number indicates the number of transceivers and the second number indicates the number of CMU (PMA only) transceivers.

(3) HardCopy IV GX devices do not have dedicated MLABs, but the Stratix IV GX MLAB features and functions are supported in HardCopy IV GX devices.

(4) This device has six PLLs in the F1152 package and four PLLs in the F780 package.

(5) This device has eight PLLs in the F1517 package and six PLLs in the F1152 package.

(6) Devices in the cost optimized LF780 and the LF1152 package have 16 transceivers and no CMU transceiver. Devices in the performance optimized FF1152 package have 16 transceivers and 8 CMU transceivers.

(7) Devices in the F1152 package have 16 transceivers and eight CMU transceivers. Devices in the performance optimized FF1517 package have 24 transceivers and 12 CMU transceivers.

Table 3–4. HardCopy IV E ASIC Features (Part 1 of 2)

HardCopy IV E ASIC	Stratix IV E FPGA Prototype	ASIC Equivalent Gates *(1)*	M9K Blocks	M144K Blocks	Total Dedicated RAM Bits (not including MLABs) *(2)*	18 x 18-Bit Multipliers (FIR Mode)	PLLs
HC4E25	EP4SE230	9.2 M	864	22	10,944 Kb	1288	4
	EP4SE360	9.4 M	864	32	12,384 Kb	1040	4

Table 3–4. HardCopy IV E ASIC Features (Part 2 of 2)

HardCopy IV E ASIC	Stratix IV E FPGA Prototype	ASIC Equivalent Gates *(1)*	M9K Blocks	M144K Blocks	Total Dedicated RAM Bits (not including MLABs) *(2)*	18 x 18-Bit Multipliers (FIR Mode)	PLLs
HC4E35	EP4SE360	9.4 M	1,248	48	18,144 Kb	1040	8
	EP4SE530	11.5 M	1,280	48	18,432 Kb	1024	12 *(3)*
	EP4SE820	14.6 M	1,320	48	18,792 Kb	960	12 *(3)*

Notes to Table 3–4:

(1) This is the number of ASIC-equivalent gates available in the HardCopy IV E base array, shared between both adaptive logic module (ALM) logic and DSP functions from a Stratix IV E FPGA prototype. The number of usable ASIC equivalent gates is bounded by the number of ALMs in the companion Stratix IV E FPGA device.

(2) HardCopy IV E devices do not have dedicated MLABs, but the Stratix IV E MLAB features and functions are supported in HardCopy IV devices.

(3) This device has 12 PLLs in the F1517 package and 8 PLLs in the F1152 package.

HardCopy IV ASICs offer pin-to-pin compatibility to the Stratix IV prototype, making them drop-in replacements for FPGAs. Due to this compatibility, the same system board and software developed for prototyping and field trials can be retained, enabling faster time-to-market for high-volume production.

HardCopy IV devices also offer non-socket replacement mapping for further cost reduction. For example, the EP4SE230 device in the 780-pin FBGA package can be mapped to the HC4E25 device in the 484-pin FBGA package. Because the pinout for the two packages are not the same, a separate board design is required for the Stratix IV device and the HardCopy IV device.

For the non-socket replacement path, select I/Os in the Stratix IV device that can be mapped to the HardCopy IV device. Not all I/Os in the Stratix IV device are available in the HardCopy IV non-socket replacement device. Check pinout information for both the Stratix IV device and the HardCopy IV device to ensure that you can map successfully, and select the HardCopy IV companion device when designing for the Stratix IV device.

Table 3–5 and Table 3–6 show available I/O pin counts by package for each Stratix IV and HardCopy IV companion pair.

Table 3–5. HardCopy IV GX and Stratix IV GX Package and I/O Pin Count Mapping (Part 1 of 2)

HardCopy IV GX ASIC *(1)*	Stratix IV GX FPGA Prototype	484-Pin FineLine BGA	780-Pin FineLine BGA *(2)*	1152-Pin FineLine BGA *(3)*	1517-Pin FineLine BGA *(4)*	1760-Pin FineLine BGA
HC4GX15LA	EP4SGX70	—	372	—	—	—
	EP4SGX110	—	372	—	—	—
	EP4SGX180	—	372	—	—	—
	EP4SGX230	—	372	—	—	—
HC4GX15L	EP4SGX290	—	257	—	—	—
	EP4SGX360	—	257	—	—	—

Table 3–5. HardCopy IV GX and Stratix IV GX Package and I/O Pin Count Mapping (Part 2 of 2)

HardCopy IV GX ASIC *(1)*	Stratix IV GX FPGA Prototype	484-Pin FineLine BGA	780-Pin FineLine BGA *(2)*	1152-Pin FineLine BGA *(3)*	1517-Pin FineLine BGA *(4)*	1760-Pin FineLine BGA
HC4GX25L	EP4SGX110	—	—	372	—	—
	EP4SGX180	—	—	564	—	—
	EP4SGX230	—	—	564	—	—
	EP4SGX290	—	289	564	—	—
	EP4SGX360	—	289	564	—	—
	EP4SGX530	—	—	—	—	—
HC4GX25F	EP4SGX110	—	—	—	—	—
	EP4SGX180	—	—	564	—	—
	EP4SGX230	—	—	564	—	—
	EP4SGX290	—	—	564	—	—
	EP4SGX360	—	—	564	—	—
	EP4SGX530	—	—	564	—	—
HC4GX35F	EP4SGX180	—	—	—	744	—
	EP4SGX230	—	—	564	744	—
	EP4SGX290	—	—	—	744	—
	EP4SGX360	—	—	564	744	—
	EP4SGX530	—	—	564	744	—

Notes to Table 3–5:

(1) The last letter (two letters in the LA package) in the HardCopy IV GX name refers to the following package types: F—Performance-optimized flip chip package, L or LA—Cost-optimized flip-chip package.

(2) The I/O pin count for the LAF780 package includes the four dedicated clock inputs (CLK1n, CLK1p, CLK3n, CLK3p). The I/O pin count for the LF780 package includes one dedicated clock input (CLK1p).

(3) All I/O pin counts include four dedicated clock inputs (CLK1p, CLK1n, CLK10p, and CLK10n) that can be used as data inputs.

(4) All I/O pin counts include eight dedicated clock inputs (CLK1p, CLK1n, CLK3p, CLK3n, CLK8p, CLK8n, CLK10p, and CLK10n) that can be used as data inputs.

Table 3–6. HardCopy IV E and Stratix IV E Package and I/O Pin Count Mapping (Part 1 of 2)

HardCopy IV E ASIC *(1)*	Stratix IV E FPGA Prototype	484-Pin FineLine BGA *(2)*	780-Pin FineLine BGA *(2)*	1152-Pin FineLine BGA *(2)*	1517-Pin FineLine BGA *(3)*	1760-Pin FineLine BGA
HC4E25W	EP4SE230	296 *(4)*	392	—	—	—
	EP4SE360	—	392	—	—	—
HC4E25F	EP4SE230	296 *(4)*	488	—	—	—
	EP4SE360	—	488	—	—	—
HC4E35L	EP4SE360	—	—	744	—	—
	EP4SE530	—	—	744	880	—
	EP4SE820	—	—	744	880	—

Table 3–6. HardCopy IV E and Stratix IV E Package and I/O Pin Count Mapping (Part 2 of 2)

HardCopy IV E ASIC *(1)*	Stratix IV E FPGA Prototype	484-Pin FineLine BGA *(2)*	780-Pin FineLine BGA *(2)*	1152-Pin FineLine BGA *(2)*	1517-Pin FineLine BGA *(3)*	1760-Pin FineLine BGA
HC4E35F	EP4SE360	—	—	744	—	—
	EP4SE530	—	—	744	880	—
	EP4SE820	—	—	744	880	—

Notes to Table 3–6:

(1) The last letter in the HardCopy IV E device name refers to the following package types: F–Performance-optimized flip-chip package, L-Cost-optimized flip-chip package, W–Low-cost wirebond package.

(2) All I/O pin counts include eight dedicated clock inputs (CLK1p, CLK1n, CLK3p, CLK3n, CLK8p, CLK8n, CLK10p, and CLK10n) that can be used for data inputs.

(3) All I/O pin counts include eight dedicated clock inputs (CLK1p, CLK1n, CLK3p, CLK3n, CLK8p, CLK8n, CLK10p, and CLK10n) and eight dedicated corner PLL clock inputs (PLL_L1_CLKp, PLL_L1_CLKn, PLL_L4_CLKp, PLL_L4_CLKn, PLL_R4_CLKp, PLL_R4_CLKn, PLL_R1_CLKp, and PLL_R1_CLKn) that can be used as data inputs.

(4) This mapping is a non-socket replacement path and requires a different board design for the Stratix IV E device and the HardCopy IV E device. The Stratix IV E device is in a 780-Pin FineLine BGA package while the HardCopy IV E device is in a 484-Pin FineLine BGA package.

Summary of Differences Between HardCopy IV and Stratix IV Devices

HardCopy IV ASICs are functionally equivalent to Stratix IV FPGAs, but they have architectural differences. When implementing your design and laying out your board, consider the differences to ensure successful design mapping from the Stratix IV FPGA to the HardCopy IV ASIC.

Architectural differences between the Stratix IV FPGA and the HardCopy IV ASIC include:

- HardCopy IV devices have up to 20 I/O banks and 880 I/O pins, while the largest Stratix IV companion devices have up to 24 I/O banks and 976 I/O pins in the 1517-pin FBGA package.
- The number of global and regional clocks is identical for Stratix IV and HardCopy IV devices, but Stratix IV devices have up to 116 peripheral clocks, while HardCopy IV devices have up to 88. The Quartus II software limits the clock availability on Stratix IV and HardCopy IV companion pairs to ensure device compatibility.
- Configuration is not required for HardCopy IV devices; therefore, these Stratix IV features are not supported:
  - Programming modes and features such as remote update and Programmer Object File (**.pof**) encryption.
  - Cyclical redundancy check (CRC) for configuration error detection.
  - 256-bit (AES) volatile and non-volatile security key to protect designs.
  - JTAG instructions used for configuration.
  - FPGA configuration emulation mode is not supported.
- Boundary scan (BSCAN) chain length is different and varies with device density.
- Memory Initialization Files (**.mif**) for embedded memories used as RAM are not supported.

- Stratix IV LAB/MLAB and DSP functions are implemented with HCells in HardCopy IV devices instead of dedicated blocks.
- Stratix IV Programmable Power Technology is not supported in HardCopy IV devices. However, the HardCopy IV ASIC architecture offers performance on par with the Stratix IV devices with significantly low power.

Designing with HardCopy IV I/Os

HardCopy IV ASICs support a wide range of industry standards that match Stratix IV supported standards. HardCopy IV devices support 3.3 V I/O standards. The 3.3 V LVTTL/LVCMOS I/O standard is supported using V_{CCIO} at 3.0 V.

HardCopy IV I/O standards support the same specifications as their Stratix IV companion equivalent. The I/O arrangement matches Stratix IV such that I/O pins located on the left and right side I/O banks contain circuits dedicated to high-speed differential I/O interfaces, but have the ability to support external memory devices if required. The top and bottom I/O banks contain dedicated circuitry to optimize external memory interfaces. They also have the ability to support high-speed differential inputs and outputs at lower speed than the left and right side banks.

For more information, refer to the *HardCopy IV Device I/O Features* chapter.

Mapping HardCopy IV and Stratix IV I/Os and Modular I/O Banks

I/O pins in Stratix IV and HardCopy IV devices are arranged in groups called modular I/O banks. On Stratix IV devices, the number of I/O banks can range from 16 to 24 banks. On HardCopy IV devices, the number of I/O banks can range from 12 to 20 banks.

In both Stratix IV and HardCopy IV devices, the maximum number of I/O banks per side is four or six, depending on the device density. When migrating between devices with a different number of I/O banks per side, the middle or "B" bank is removed or inserted. For example, when moving from a 24-bank Stratix IV device to a 16-bank Stratix IV or HardCopy IV device, the banks that are dropped are "B" banks, namely 1B, 2B, 3B, 4B, 5B, 6B, 7B, and 8B.

HardCopy IV devices do not have banks 1B, 2B, 5B, and 6B. When you design with a Stratix IV device that has 24 banks, the Quartus II software limits the available banks common to all devices selected if a HardCopy IV device is selected as a companion pair. If you try to assign I/O pins to a non-existent bank in a mapping or companion device, the Quartus II compilation halts with an error.

The following are examples of Quartus II compilation errors:

Error: I/O pins (xx) assigned in I/O bank 1B. The I/O bank does not exist in the selected device

Error: Device migration enabled -- compilation may have failed due to additional constraints when migrating

Error: Can't fit design in device

The sizes of each bank are 24, 26, 32, 40, 42, 48, or 50 I/O pins (including up to two dedicated input pins per bank). During mapping from a smaller device to a larger device, the bank size increases or remains the same but never decreases. For example, banks may increase from a size of 24 I/O to a bank of size 32, 40, 48, or 50 I/O, but will never decrease.

Table 3–7 summarizes the number of I/O pins available in each I/O bank for all companion pairs of Stratix IV GX and HardCopy IV GX devices.

Table 3–7. HardCopy IV GX ASIC and Stratix IV GX FPGA I/O Bank and Pin Count Mapping (Part 1 of 2) *(1)*

Bank	780-Pin FineLine BGA *(2)*		780-Pin FineLine BGA *(2)*		780-Pin FineLine BGA *(2)*		1152-Pin FineLine BGA *(3)*		1152-Pin FineLine BGA *(3)*		1517-Pin FineLine BGA *(4)*	
	HardCopy IV GX ASIC	Stratix IV GX FPGA Prototype	HardCopy IV GX ASIC	Stratix IV GX FPGA Prototype *(5)*	HardCopy IV GX ASIC	Stratix IV GX FPGA Prototype *(5)*	HardCopy IV GX ASIC	Stratix IV GX FPGA Prototype	HardCopy IV GX ASIC	Stratix IV GX FPGA Prototype	HardCopy IV GX ASIC	Stratix IV GX FPGA Prototype
	HC4GX15	EP4SGX70 EP4SGX110 EP4SGX180 EP4SGX230	HC4GX15	EP4SGX290 EP4SGX360	HC4GX25L	EP4SGX290 EP4SGX360	HC4GX25L *(8)*	EP4SGX110	HC4GX25L HC4GX25F HC4GX35F	EP4SGX180 EP4SGX230 EP4SGX290 EP4SGX360 EP4SGX530 *(6)*	HC4GX35F	EP4SGX180 EP4SGX230 EP4SGX290 EP4SGX360 EP4SGX530 *(7)*
1A	32	32	—	—	—	—	32	32	48	48	48	48
1B	—	—	—	—	—	—	—	—	—	—	—	—
1C	26	26	1	1	1	1	26	26	42	42	42	42
2A	32	32	—	—	—	—	—	—	—	—	48	48
2B	—	—	—	—	—	—	—	—	—	—	—	—
2C	26	26	—	—	—	—	—	—	—	—	42	42
3A	40	40	40	40	40	40	40	40	40	40	40	40
3B	—	—	—	—	—	—	—	—	24	24	24	24
3C	24	24	24	32	32	32	24	24	32	32	32	32
4A	40	40	40	40	40	40	40	40	40	40	40	40
4B	—	—	—	—	—	—	—	—	24	24	24	24
4C	24	24	24	32	32	32	24	24	32	32	32	32
5A	—	—	—	—	—	—	—	—	—	—	48	48
5B	—	—	—	—	—	—	—	—	—	—	—	—
5C	—	—	—	—	—	—	—	—	—	—	42	42
6A	—	—	—	—	—	—	32	32	48	48	48	48
6B	—	—	—	—	—	—	—	—	—	—	—	—
6C	—	—	—	—	—	—	26	26	42	42	42	42

Table 3–7. HardCopy IV GX ASIC and Stratix IV GX FPGA I/O Bank and Pin Count Mapping (Part 2 of 2) *(1)*

Bank	780-Pin FineLine BGA *(2)*		780-Pin FineLine BGA *(2)*		780-Pin FineLine BGA *(2)*		1152-Pin FineLine BGA *(3)*		1152-Pin FineLine BGA *(3)*		1517-Pin FineLine BGA *(4)*	
	HardCopy IV GX ASIC	Stratix IV GX FPGA Prototype	HardCopy IV GX ASIC	Stratix IV GX FPGA Prototype *(5)*	HardCopy IV GX ASIC	Stratix IV GX FPGA Prototype *(5)*	HardCopy IV GX ASIC	Stratix IV GX FPGA Prototype	HardCopy IV GX ASIC	Stratix IV GX FPGA Prototype	HardCopy IV GX ASIC	Stratix IV GX FPGA Prototype
	HC4GX15	EP4SGX70 EP4SGX110 EP4SGX180 EP4SGX230	HC4GX15	EP4SGX290 EP4SGX360	HC4GX25L	EP4SGX290 EP4SGX360	HC4GX25L *(8)*	EP4SGX110	HC4GX25L HC4GX25F HC4GX35F	EP4SGX180 EP4SGX230 EP4SGX290 EP4SGX360 EP4SGX530 *(6)*	HC4GX35F	EP4SGX180 EP4SGX230 EP4SGX290 EP4SGX360 EP4SGX530 *(7)*
7A	40	40	40	40	40	40	40	40	40	40	40	40
7B	—	—	—	—	—	—	—	—	24	24	24	24
7C	24	24	24	32	32	32	24	24	32	32	32	32
8A	40	40	40	40	40	40	40	40	40	40	40	40
8B	—	—	—	—	—	—	—	—	24	24	24	24
8C	24	24	24	32	32	32	24	24	32	32	32	32
Tota l I/O	372	372	257	289	289	289	372	372	564	564	744	744

Notes to Table 3–7:

(1) User I/O pin counts are preliminary.

(2) All I/O pin counts include four dedicated clock inputs (CLK1p, CLK1n, CLK3p, CLK3n) that can be used for data inputs. The EP4SGX290 and EP4SGX360 mappings include only one dedicated clock input (CLK1p) that can be used as data input.

(3) All I/O pin counts include four dedicated clock inputs (CLK1p, CLK1n, CLK10p, and CLK10n) that can be used as data inputs.

(4) All I/O pin counts include eight dedicated clock inputs (CLK1p, CLK1n, CLK3p, CLK3n, CLK8p, CLK8n, CLK10p, and CLK10n) that can be used as data inputs.

(5) The EP4SGX290 and EP4SGX360 FPGAs are offered in the H780 package.

(6) The EP4SGX530 FPGA is offered in the H1152 package.

(7) The EP4SGX530 FPGA is offered in the H1517 package.

(8) The HC4GX25L has 564 I/Os, but only 372 I/Os can be mapped if the FPGA EP4SGX110 is selected.

Table 3–8 and Table 3–9 summarize the number of I/O pins available in each I/O bank for all companion pairs of Stratix IV E and HardCopy IV E devices for socket replacement and non-socket replacement flows, respectively.

Table 3–8. HardCopy IV E ASIC and Stratix IV E FPGA Protyotype I/O Pin Bank and Pin Count Mapping with Socket Replacement Flow *(Note 1)*

	780-Pin FineLine BGA *(2)*		780-Pin FineLine BGA *(2)*		1152-Pin FineLine BGA		1517-Pin FineLine BGA	
	HardCopy IV E ASIC	Stratix IV E FPGA Prototype	HardCopy IV E ASIC	Stratix IV E FPGA Prototype	HardCopy IV E ASIC *(2)*	Stratix IV E FPGA Prototype	HardCopy IV E ASIC *(3)*	Stratix IV E FPGA Prototype
Bank	HC4E25W	EP4SE230 EP4SE360 *(4)*	HC4E25F	EP4SE230 EP4SE360 *(4)*	HC4E35F HC4E35L	EP4SE360 EP4SE530 *(5)* EP4SE820 *(5)*	HC4E35L HC4E35F	EP4SE530 *(6)* EP4SE820
1A	24	32	32	32	48	48	50	50
1B	—	—	—	—	—	—	—	24
1C	42	26	26	26	42	42	42	42
2A	24	32	32	32	48	48	50	50
2B	—	—	—	—	—	—	—	24
2C	42	26	26	26	42	42	42	42
3A	—	40	40	40	40	40	48	48
3B	—	—	—	—	24	24	48	48
3C	32	24	24	24	32	32	32	32
4A	—	40	40	40	40	40	48	48
4B	—	—	—	—	24	24	48	48
4C	32	24	24	24	32	32	32	32
5A	24	32	32	32	48	48	50	50
5B	—	—	—	—	—	—	—	24
5C	42	26	26	26	42	42	42	42
6A	24	32	32	32	48	48	50	50
6B	—	—	—	—	—	—	—	24
6C	42	26	26	26	42	42	42	42
7A	—	40	40	40	40	40	48	48
7B	—	—	—	—	24	24	48	48
7C	32	24	24	24	32	32	32	32
8A	—	40	40	40	40	40	48	48
8B	—	—	—	—	24	24	48	48
8C	32	24	24	24	32	32	32	32
Total I/O	392	488	488	488	744	744	880	976

Notes to Table 3–8:

(1) User I/O pin counts are preliminary.

(2) All I/O pin counts include eight dedicated clock inputs (CLK1p, CLK1n, CLK3p, CLK3n, CLK8p, CLK8n, CLK10p and CLK10n) that can be used for data inputs.

(3) All I/O pin counts include eight dedicated clock inputs (CLK1p, CLK1n, CLK3p, CLK3n, CLK8p, CLK8n, CLK10p and CLK10n) and eight dedicated corner PLL clock inputs (PLL_L1_CLKp, PLL_L1_CLKn, PLL_L4_CLKp, PLL_L4_CLKn, PLL_R4_CLKp, PLL_R4_CLKn, PLL_R1_CLKp, and PLL_R1_CLKn) that can be used for data inputs.

(4) The EP4SE360 FPGA is offered in the H780 package.

(5) The EP4SE530 and EP4SE820 FPGAs are offered in the H1152 package.

(6) The EP4SE530 FPGA is offered in the H1517 package.

HardCopy IV ASICs offer the non-socket replacement flow to reduce design board space and cost. Because HardCopy IV and Stratix IV device packages are different, some Stratix IV device I/Os are not available in the HardCopy IV device. Table 3–9 shows the number of I/Os in each bank on Stratix IV and HardCopy IV devices for the non-socket replacement flow.

Table 3–9. HardCopy IV E and Stratix IV E I/O Bank and Count Mapping with Non-Socket Replacement Flow *(Note 1)*

	484-pin FineLine BGA	780-Pin FineLine BGA
	HardCopy IV E ASIC	Stratix IV E FPGA Prototype
Bank	HC4E25W HC4E25F	EP4SE230
1A	24	32
1B	—	—
1C	26	26
2A	24	32
2B	—	—
2C	26	26
3A	—	40
3B	—	—
3C	24	24
4A	—	40
4B	—	—
4C	24	24
5A	24	32
5B	—	—
5C	26	26
6A	24	32
6B	—	—
6C	26	26
7A	—	40
7B	—	—
7C	24	24
8A	—	40
8B	—	—
8C	24	24
Total I/O	296	488

Note to Table 3–9:

(1) User I/O pin counts are preliminary.

HardCopy IV Supported I/O Standards

HardCopy IV ASICs support the same I/O standards as Stratix IV FPGAs.

Table 3–10 lists I/O standards that HardCopy IV ASIC support.

Table 3–10. I/O Standards and Voltage Levels for HardCopy IV Devices (Part 1 of 2) *(Note 1)*

I/O Standard	Standard Support	V_{CCIO} (V)				V_{CCPD} (V) (Pre-Driver Voltage)	V_{REF} (V) (Input Ref Voltage)	V_{TT} (V) (Board Termination Voltage)
		Input Operation		Output Operation				
		Column I/O Banks	Row I/O Banks	Column I/O Banks	Row I/O Banks			
3.3-V LVTTL *(2)*	JESD8-B	3.0/2.5	3.0/2.5	3.0	3.0	3.0	—	—
3.3-V LVCMOS *(2)*	JESD8-B	3.0/2.5	3.0/2.5	3.0	3.0	3.0	—	—
2.5-V LVTTL/LVCMOS	JESD8-5	3.0/2.5	3.0/2.5	2.5	2.5	2.5	—	—
1.8-V LVTTL/LVCMOS	JESD8-7	1.8/1.5	1.8/1.5	1.8	1.8	2.5	—	—
1.5-V LVTTL/LVCMOS	JESD8-11	1.8/1.5	1.8/1.5	1.5	1.5	2.5	—	—
1.2-V LVTTL/LVCMOS	JESD8-12	1.2	1.2	1.2	1.2	2.5	—	—
3.0-V PCI	PCI Rev 2.1	3.0	3.0	3.0	3.0	3.0	—	—
3.0-V PCI-X	PCI-X Rev 1.0	3.0	3.0	3.0	3.0	3.0	—	—
SSTL-2 Class I	JESD8-9B	*(3)*	*(3)*	2.5	2.5	2.5	1.25	1.25
SSTL-2 Class II	JESD8-9B	*(3)*	*(3)*	2.5	2.5	2.5	1.25	1.25
SSTL-18 Class I	JESD8-15	*(3)*	*(3)*	1.8	1.8	2.5	0.90	0.90
SSTL-18 Class II	JESD8-15	*(3)*	*(3)*	1.8	1.8	2.5	0.90	0.90
SSTL-15 Class I	—	*(3)*	*(3)*	1.5	1.5	2.5	0.75	0.75
SSTL-15 Class II	—	*(3)*	*(3)*	1.5	—	2.5	0.75	0.75
HSTL-18 Class I	JESD8-6	*(3)*	*(3)*	1.8	1.8	2.5	0.90	0.90
HSTL-18 Class II	JESD8-6	*(3)*	*(3)*	1.8	1.8	2.5	0.90	0.90
HSTL-15 Class I	JESD8-6	*(3)*	*(3)*	1.5	1.5	2.5	0.75	0.75
HSTL-15 Class II	JESD8-6	*(3)*	*(3)*	1.5	—	2.5	0.75	0.75
HSTL-12 Class I	JESD8-16A	*(3)*	*(3)*	1.2	1.2	2.5	0.6	0.6
HSTL-12 Class II	JESD8-16A	*(3)*	*(3)*	1.2	—	2.5	0.6	0.6
Differential SSTL-2 Class I	JESD8-9B	*(3)*	*(3)*	2.5	2.5	2.5	—	1.25
Differential SSTL-2 Class II	JESD8-9B	*(3)*	*(3)*	2.5	2.5	2.5	—	1.25
Differential SSTL-18 Class I	JESD8-15	*(3)*	*(3)*	1.8	1.8	2.5	—	0.90
Differential SSTL-18 Class II	JESD8-15	*(3)*	*(3)*	1.8	1.8	2.5	—	0.90
Differential SSTL-15 Class I	—	*(3)*	*(3)*	1.5	1.5	2.5	—	0.75
Differential SSTL-15 Class II	—	*(3)*	*(3)*	1.5	—	2.5	—	0.75

Table 3–10. I/O Standards and Voltage Levels for HardCopy IV Devices (Part 2 of 2) *(Note 1)*

I/O Standard	Standard Support	V_{CCIO} (V)				V_{CCPD} (V) (Pre-Driver Voltage)	V_{REF} (V) (Input Ref Voltage)	V_{TT} (V) (Board Termination Voltage)
		Input Operation		Output Operation				
		Column I/O Banks	Row I/O Banks	Column I/O Banks	Row I/O Banks			
Differential HSTL-18 Class I	JESD8-6	*(3)*	*(3)*	1.8	1.8	2.5	—	0.90
Differential HSTL-18 Class II	JESD8-6	*(3)*	*(3)*	1.8	1.8	2.5	—	0.90
Differential HSTL-15 Class I	JESD8-6	*(3)*	*(3)*	1.5	1.5	2.5	—	0.75
Differential HSTL-15 Class II	JESD8-6	*(3)*	*(3)*	1.5	—	2.5	—	0.75
Differential HSTL-12 Class I	JESD8-16A	*(3)*	*(3)*	1.2	1.2	2.5	—	0.60
Differential HSTL-12 Class II	JESD8-16A	*(3)*	*(3)*	1.2	—	2.5	—	0.60
LVDS *(4)*, *(5)*	ANSI/TIA/EIA-644	*(3)*	*(3)*	2.5	2.5	2.5	—	—
RSDS *(6)*, *(7)*	—	*(3)*	*(3)*	2.5	2.5	2.5	—	—
mini-LVDS *(6)*, *(7)*	—	*(3)*	*(3)*	2.5	2.5	2.5	—	—
LVPECL	—	*(4)*	2.5	—	—	2.5	—	—

Notes to Table 3–10:

(1) V_{CCPD} is either 2.5 or 3.0 V. For V_{CCIO} = 3.0 V, V_{CCPD} = 3.0 V. For V_{CCIO} = 2.5 V or less, V_{CCPD} = 2.5 V.

(2) The 3.3-V LVTTL/LVCMOS standard is supported using VCCIO at 3.0 V.

(3) Single-ended HSTL/SSTL, differential SSTL/HSTL, and LVDS input buffers are powered by V_{CCPD}. Row I/O banks support both true differential input buffers and true differential output buffers. Column I/O banks support true differential input buffers, but not true differential output buffers. I/O pins are organized in pairs to support differential standards. Column I/O differential HSTL and SSTL inputs use LVDS differential input buffers without on-chip R_D support.

(4) Column I/O banks support LVPECL I/O standards for input clock operation. Clock inputs on column I/O are powered by $V_{CCCLKIN}$ when configured as differential clock input. They are powered by V_{CCIO} when configured as single-ended clock input. Differential clock inputs in row I/O are powered by V_{CCPD}.

(5) Column and row I/O banks support LVDS outputs using two single-ended output buffers, an external one-resistor (LVDS_E_1R), and a three-resistor (LVDS_E_3R) network.

(6) Row I/O banks support RSDS and mini-LVDS I/O standards using a dedicated LVDS output buffer without a resistor network.

(7) Column and row I/O banks support RSDS and mini-LVDS I/O standards using two single-ended output buffers with one-resistor (RSDS_E_1R and mini-LVDS_E_1R) and three-resistor (RSDS_E_3R and mini-LVDS_E_3R) networks.

External Memory Interface I/Os in Stratix IV and HardCopy IV Devices

As with the Stratix IV I/O structure, the redesign of the HardCopy IV I/O structure provides flexible and high-performance support for existing and emerging external memory standards including DDR3, DDR2, DDR SDRAM, QDRII+, QDRII SRAM, and RLDRAM II.

HardCopy IV devices offer the same external memory interface features found in Stratix IV devices. These features include delay-locked loops (DLLs), phase-locked loops (PLLs), dynamic on-chip termination (OCT), trace mismatch compensation, read and write leveling, deskew circuitry, half data rate (HDR) blocks, 4- to 36-bit DQ group widths, and DDR external memory support on all sides of the HardCopy IV device.

As with Stratix IV devices, HardCopy IV devices allow a memory interface to be located on any side of the device. The only limitation is if the left and right sides have to be reserved for high-speed I/O applications, as described in the following section.

Table 3–11 and Table 3–12 show the number of DQ and DQS buses supported per companion device pair.

Table 3–11. Number of DQS/DQ Groups in HardCopy IV GX Devices per Side *(Note 1)*

HardCopy IV GX ASIC	Package	Side	x4 *(2)*	x8/x9	x16/x18	x32/x36
HC4GX15LA	780-pin FineLine BGA	Left	14	6	2	0
		Bottom	17	8	2	0
		Right	0	0	0	0
		Top	17	8	2	0
HC4GX15L	780-pin FineLine BGA	Left	0	0	0	0
		Bottom	17	8	2	0
		Right	0	0	0	0
		Top	17	8	2	0
HC4GX25	1152-pin FineLine BGA	Left	13	6	2	0
		Bottom	26	12	4	0
		Right	13	6	2	0
		Top	26	12	4	0
HC4GX35	1152-pin FineLine BGA	Left	13	6	2	0
		Bottom	26	12	4	0
		Right	13	6	2	0
		Top	26	12	4	0
HC4GX35	1517-pin FineLine BGA	Left	26	12	4	0
		Bottom	26	12	4	0
		Right	26	12	4	0
		Top	26	12	4	0

Notes to Table 3–11:

(1) These numbers are preliminary.

(2) Some of the DQS and DQ pins can also be used as R_{UP}/R_{DN} pins. You lose one DQS/DQ group if you use these pins as R_{UP}/R_{DN} pins for OCT calibration. Make sure that the DQS/DQ groups that you have chosen are not also used for OCT calibration.

Table 3–12. Number of DQS/DQ Groups in HardCopy IV E Devices per Side *(Note 1)* (Part 1 of 2)

HardCopy IV E ASIC	Package	Side	x4 *(2)*	x8/x9	x16/x18	x32/x36
HC4E25	484-pin FineLine BGA	Left	12	4	0	0
		Bottom	5	2	0	0
		Right	12	4	0	0
		Top	5	2	0	0

Table 3–12. Number of DQS/DQ Groups in HardCopy IV E Devices per Side *(Note 1)* (Part 2 of 2)

HardCopy IV E ASIC	Package	Side	x4 *(2)*	x8/x9	x16/x18	x32/x36
HC4E25	780-pin FineLine BGA	Left	14	6	2	0
		Bottom	17	8	2	0
		Right	14	6	2	0
		Top	17	8	2	0
HC4E35	1152-pin FineLine BGA	Left	26	12	4	0
		Bottom	26	12	4	0
		Right	26	12	4	0
		Top	26	12	4	0
HC4E35	1517-pin FineLine BGA	Left	26	12	4	0
		Bottom	38	18	8	4
		Right	26	12	4	0
		Top	38	18	8	4

Notes to Table 3–12:

(1) These numbers are preliminary.

(2) Some of the DQS and DQ pins can also be used as R_{UP}/R_{DN} pins. You lose one DQS/DQ group if you use these pins as R_{UP}/R_{DN} pins for OCT calibration. Make sure that the DQS/DQ groups that you have chosen are not also used for OCT calibration.

For more information about external memory interfaces, refer to the *External Memory Interfaces in HardCopy IV Devices* chapter.

Mapping Stratix IV High-Speed Differential I/O Interfaces with HardCopy IV

HardCopy IV ASICs have the same dedicated circuitry as Stratix IV devices for high-speed differential I/O support:

- Differential I/O buffer
- Transmitter serializer
- Receiver deserializer
- Data realignment
- Dynamic phase aligner (DPA)
- Synchronizer (FIFO buffer)
- Analog PLLs (located on left and right sides of the device)

For high-speed differential interfaces, HardCopy IV devices support the following differential I/O standards:

- Low-voltage differential signaling (LVDS)
- Mini-LVDS
- Reduced swing differential signaling (RSDS)
- Differential HSTL
- Differential SSTL

HardCopy IV ASICs support LVDS on all I/O banks. True LVDS makes use of dedicated LVDS I/O buffers that are optimized for performance. There are true LVDS input and output buffers at the left and right side I/O banks. There are true LVDS input buffers on the top and bottom I/O banks only.

You can configure all I/Os in all banks as emulated LVDS output buffers. Emulated output buffers make use of single-ended buffers and an external resistor network to mimic LVDS operation. Emulated LVDS is useful for low-speed, low-voltage differential applications.

For more information about high-speed I/O performance, refer to the *High Speed Differential I/O Interface with DPA in HardCopy IV Devices* chapter.

All dedicated circuitry for high-speed differential I/O applications are located in the left and right I/O banks of the Stratix IV and HardCopy IV devices. The top and bottom I/O banks also have support for high-speed receiver applications that do not require the use of the DPA, synchronizer, data realignment, and differential termination. Top and bottom differential I/O buffers have a slower data rate than the high-speed receivers on the left and right I/O banks.

Table 3–13 shows the LVDS channels supported in HardCopy IV GX and Stratix IV GX companion devices.

Table 3–13. LVDS Channels Supported In HardCopy IV GX and Stratix IV GX Companion Devices *(Note 1)*, *(2)* (Part 1 of 3)

	780-Pin FineLine BGA		1152-Pin FineLine BGA		1152-Pin FineLine BGA		1152-Pin FineLine BGA		1517-Pin FineLine BGA	
	HardCopy IV GX ASIC	Stratix IV GX FPGA Prototype	HardCopy IV GX ASIC	Stratix IV GX FPGA Prototype	HardCopy IV GX ASIC	Stratix IV GX FPGA Prototype	HardCopy IV GX ASIC	Stratix IV GX FPGA Prototype	HardCopy IV GX ASIC	Stratix IV GX FPGA Prototype
Bank	HC4GX15	EP4SGX70 EP4SGX110 EP4SGX180 EP4SGX230	HC4GX25	EP4SGX110	HC4GX25	EP4SGX180 EP4SGX230 EP4SGX290 EP4SGX360 EP4SGX530 *(3)*	HC4GX35	EP4SGX230 EP4SGX360 EP4SGX530 *(3)*	HC4GX35	EP4SGX180 EP4SGX230 EP4SGX290 EP4SGX360 EP4SGX530 *(4)*
1A	8Rx + 8Tx	8Rx + 8Tx	8Rx + 8Tx	8Rx + 8Tx	12Rx + 12Tx	12Rx + 12Tx	12Rx + 12Tx	12Rx + 12Tx	12Rx + 12Tx	12Rx + 12Tx
1B	—	—	—	—	—	—	—	—	—	—
1C	6Rx + 6Tx	6Rx + 6Tx	6Rx + 6Tx	6Rx + 6Tx	10Rx + 10Tx	10Rx + 10Tx	10Rx + 10Tx	10Rx + 10Tx	10Rx + 10Tx	10Rx + 10Tx
2A	8Rx + 8Tx	8Rx + 8Tx	8Rx + 8Tx	8Rx + 8Tx	-	12Rx + 12Tx	-	12Rx + 12Tx	12Rx + 12Tx	12Rx + 12Tx
2B	—	—	—	—	—	—	—	—	—	—
2C	6Rx + 6Tx	6Rx + 6Tx	6Rx + 6Tx	6Rx + 6Tx	-	10Rx + 10Tx	-	10Rx + 10Tx	10Rx + 10Tx	10Rx + 10Tx
3A *(5)*	10Rx + 10eTx or 20eTx	10Rx + 10eTx or 20eTx	10Rx + 10eTx or 20eTx	10Rx + 10eTx or 20eTx	10Rx + 10eTx or 20eTx	10Rx + 10eTx or 20eTx	10Rx + 10eTx or 20eTx	10Rx + 10eTx or 20eTx	10Rx + 10eTx or 20eTx	10Rx + 10eTx or 20eTx
3B *(5)*	—	—	—	—	6Rx + 6eTx or 12eTx	6Rx + 6eTx or 12eTx	6Rx + 6eTx or 12eTx	6Rx + 6eTx or 12eTx	6Rx + 6eTx or 12eTx	6Rx + 6eTx or 12eTx

Table 3–13. LVDS Channels Supported In HardCopy IV GX and Stratix IV GX Companion Devices *(Note 1)*, *(2)* (Part 2 of 3)

	780-Pin FineLine BGA		1152-Pin FineLine BGA		1152-Pin FineLine BGA		1152-Pin FineLine BGA		1517-Pin FineLine BGA	
	HardCopy IV GX ASIC	Stratix IV GX FPGA Prototype	HardCopy IV GX ASIC	Stratix IV GX FPGA Prototype	HardCopy IV GX ASIC	Stratix IV GX FPGA Prototype	HardCopy IV GX ASIC	Stratix IV GX FPGA Prototype	HardCopy IV GX ASIC	Stratix IV GX FPGA Prototype
Bank	HC4GX15	EP4SGX70 EP4SGX110 EP4SGX180 EP4SGX230	HC4GX25	EP4SGX110	HC4GX25	EP4SGX180 EP4SGX230 EP4SGX290 EP4SGX360 EP4SGX530 *(3)*	HC4GX35	EP4SGX230 EP4SGX360 EP4SGX530 *(3)*	HC4GX35	EP4SGX180 EP4SGX230 EP4SGX290 EP4SGX360 EP4SGX530 *(4)*
3C *(5)*	6Rx + 6eTx or 12eTx	6Rx + 6eTx or 12eTx	6Rx + 6eTx or 12eTx	6Rx + 6eTx or 12eTx	8Rx + 8eTx or 16eTx	8Rx + 8eTx or 16eTx	8Rx + 8eTx or 16eTx	8Rx + 8eTx or 16eTx	8Rx + 8eTx or 16eTx	8Rx + 8eTx or 16eTx
4A *(5)*	10Rx + 10eTx or 20eTx	10Rx + 10eTx or 20eTx	10Rx + 10eTx or 20eTx	10Rx + 10eTx or 20eTx	10Rx + 10eTx or 20eTx	10Rx + 10eTx or 20eTx	10Rx + 10eTx or 20eTx	10Rx + 10eTx or 20eTx	10Rx + 10eTx or 20eTx	10Rx + 10eTx or 20eTx
4B *(5)*	—	—	—	—	6Rx + 6eTx or 12eTx	6Rx + 6eTx or 12eTx	6Rx + 6eTx or 12eTx	6Rx + 6eTx or 12eTx	6Rx + 6eTx or 12eTx	6Rx + 6eTx or 12eTx
4C *(5)*	6Rx + 6eTx or 12eTx	6Rx + 6eTx or 12eTx	6Rx + 6eTx or 12eTx	6Rx + 6eTx or 12eTx	8Rx + 8eTx or 16eTx	8Rx + 8eTx or 16eTx	8Rx + 8eTx or 16eTx	8Rx + 8eTx or 16eTx	8Rx + 8eTx or 16eTx	8Rx + 8eTx or 16eTx
5A	—	—	—	—	—	12Rx + 12Tx	—	12Rx + 12Tx	12Rx + 12Tx	12Rx + 12Tx
5B	—	—	—	—	—	-	—	—	—	—
5C	—	—	—	—	—	10Rx + 10Tx	—	10Rx + 10Tx	10Rx + 10Tx	10Rx + 10Tx
6A	—	—	—	6Rx + 6Tx	12Rx + 12Tx	12Rx + 12Tx	12Rx + 12Tx	12Rx + 12Tx	12Rx + 12Tx	12Rx + 12Tx
6B	—	—	—	—	—	—	—	—	—	—
6C	—	—	—	6Rx + 6Tx	10Rx + 10Tx	10Rx + 10Tx	10Rx + 10Tx	10Rx + 10Tx	10Rx + 10Tx	10Rx + 10Tx
7A *(5)*	10Rx + 10eTx or 20eTx	10Rx + 10eTx or 20eTx	10Rx + 10eTx or 20eTx	10Rx + 10eTx or 20eTx	10Rx + 10eTx or 20eTx	10Rx + 10eTx or 20eTx	10Rx + 10eTx or 20eTx	10Rx + 10eTx or 20eTx	10Rx + 10eTx or 20eTx	10Rx + 10eTx or 20eTx
7B *(5)*	—	—	—	—	6Rx + 6eTx or 12eTx	6Rx + 6eTx or 12eTx	6Rx + 6eTx or 12eTx	6Rx + 6eTx or 12eTx	6Rx + 6eTx or 12eTx	6Rx + 6eTx or 12eTx
7C *(5)*	6Rx + 6eTx or 12eTx	6Rx + 6eTx or 12eTx	6Rx + 6eTx or 12eTx	6Rx + 6eTx or 12eTx	8Rx + 8eTx or 16eTx	8Rx + 8eTx or 16eTx	8Rx + 8eTx or 16eTx	8Rx + 8eTx or 16eTx	8Rx + 8eTx or 16eTx	8Rx + 8eTx or 16eTx
8A *(5)*	10Rx + 10eTx or 20eTx	10Rx + 10eTx or 20eTx	10Rx + 10eTx or 20eTx	10Rx + 10eTx or 20eTx	10Rx + 10eTx or 20eTx	10Rx + 10eTx or 20eTx	10Rx + 10eTx or 20eTx	10Rx + 10eTx or 20eTx	10Rx + 10eTx or 20eTx	10Rx + 10eTx or 20eTx
8B *(5)*	—	—	—	—	6Rx + 6eTx or 12eTx	6Rx + 6eTx or 12eTx	6Rx + 6eTx or 12eTx	6Rx + 6eTx or 12eTx	6Rx + 6eTx or 12eTx	6Rx + 6eTx or 12eTx

Table 3–13. LVDS Channels Supported In HardCopy IV GX and Stratix IV GX Companion Devices *(Note 1)*, *(2)* (Part 3 of 3)

	780-Pin FineLine BGA		1152-Pin FineLine BGA		1152-Pin FineLine BGA		1152-Pin FineLine BGA		1517-Pin FineLine BGA	
	HardCopy IV GX ASIC	Stratix IV GX FPGA Prototype	HardCopy IV GX ASIC	Stratix IV GX FPGA Prototype	HardCopy IV GX ASIC	Stratix IV GX FPGA Prototype	HardCopy IV GX ASIC	Stratix IV GX FPGA Prototype	HardCopy IV GX ASIC	Stratix IV GX FPGA Prototype
Bank	HC4GX15	EP4SGX70 EP4SGX110 EP4SGX180 EP4SGX230	HC4GX25	EP4SGX110	HC4GX25	EP4SGX180 EP4SGX230 EP4SGX290 EP4SGX360 EP4SGX530 *(3)*	HC4GX35	EP4SGX230 EP4SGX360 EP4SGX530 *(3)*	HC4GX35	EP4SGX180 EP4SGX230 EP4SGX290 EP4SGX360 EP4SGX530 *(4)*
8C *(5)*	6Rx + 6eTx or 12eTx	6Rx + 6eTx or 12eTx	6Rx + 6eTx or 12eTx	6Rx + 6eTx or 12eTx	8Rx + 8eTx or 16eTx	8Rx + 8eTx or 16eTx	8Rx + 8eTx or 16eTx	8Rx + 8eTx or 16eTx	8Rx + 8eTx or 16eTx	8Rx + 8eTx or 16eTx

Notes to Table 3–13:

(1) Channel counts are preliminary.

(2) Rx = true LVDS input buffers with OCT RD, Tx = true LVDS output buffers, and eTx = emulated LVDS output buffers (either LVDS_E_1R or LVDS_E_3R).

(3) The EP4SGX530 FPGA is offered only in the H1152 package.

(4) The EP4SGX530 FPGA is offered only in the H1517 package.

(5) Top and bottom I/O banks do not have DPA, synchronizer, data realignment, and differential termination support in Stratix IV GX and HardCopy IV GX devices. Use left and right I/O banks if these features and maximum performance is required.

Table 3–15 and Table 3–14 show the LVDS channels supported in HardCopy IV E and Stratix IV E companion devices for the socket replacement and non-socket replacement flows, respectively.

Table 3–14. LVDS Channels Supported In HardCopy IV E and Stratix IV E Companion Devices with Socket Replacement Flow *(Note 1)*, *(2)* (Part 1 of 3)

	780-Pin FineLine BGA		1152-Pin FineLine BGA		1517-Pin FineLine BGA	
	HardCopy IV E ASIC	Stratix IV E FPGA Prototype	HardCopy IV E ASIC	Stratix IV E FPGA Prototype	HardCopy IV E ASIC	Stratix IV E FPGA Prototype
Bank	HC4E25	EP4SE230 EP4SE360 *(3)*	HC4E35	EP4SE360 EP4SE530 *(4)* EP4SE820 *(4)*	HC4E35	EP4SE530 *(5)* EP4SE820
1A	8Rx + 8Tx *(7)*	8Rx + 8Tx	12Rx + 12Tx	12Rx + 12Tx	12Rx + 12Tx	12Rx + 12Tx
1B	—	—	—	—	—	6Rx + 6Tx
1C	6Rx + 6Tx	6Rx + 6Tx	10Rx + 10Tx	10Rx + 10Tx	10Rx + 10Tx	10Rx + 10Tx
2A	8Rx + 8Tx *(7)*	8Rx + 8Tx	12Rx + 12Tx	12Rx + 12Tx	12Rx + 12Tx	12Rx + 12Tx
2B	—	—	—	—	—	6Rx + 6Tx
2C	6Rx + 6Tx	6Rx + 6Tx	10Rx + 10Tx	10Rx + 10Tx	10Rx + 10Tx	10Rx + 10Tx
3A *(6)*	10Rx + 10eTx or 20eTx	10Rx + 10eTx or 20eTx	10Rx + 10eTx or 20eTx	10Rx + 10eTx or 20eTx	12Rx + 12eTx or 24eTx	12Rx + 12eTx or 24eTx
3B *(6)*	—	—	6Rx + 6eTx or 12eTx	6Rx + 6eTx or 12eTx	12Rx + 12eTx or 24eTx	12Rx + 12eTx or 24eTx

Table 3–14. LVDS Channels Supported In HardCopy IV E and Stratix IV E Companion Devices with Socket Replacement Flow *(Note 1)*, *(2)* (Part 2 of 3)

	780-Pin FineLine BGA		1152-Pin FineLine BGA		1517-Pin FineLine BGA	
	HardCopy IV E ASIC	Stratix IV E FPGA Prototype	HardCopy IV E ASIC	Stratix IV E FPGA Prototype	HardCopy IV E ASIC	Stratix IV E FPGA Prototype
Bank	HC4E25	EP4SE230 EP4SE360 *(3)*	HC4E35	EP4SE360 EP4SE530 *(4)* EP4SE820 *(4)*	HC4E35	EP4SE530 *(5)* EP4SE820
3C *(6)*	6Rx + 6eTx or 12eTx	6Rx + 6eTx or 12eTx	8Rx + 8eTx or 16eTx	8Rx + 8eTx or 16eTx	8Rx + 8eTx or 16eTx	8Rx + 8eTx or 16eTx
4A *(6)*	10Rx + 10eTx or 20eTx	10Rx + 10eTx or 20eTx	10Rx + 10eTx or 20eTx	10Rx + 10eTx or 20eTx	12Rx + 12eTx or 24eTx	12Rx + 12eTx or 24eTx
4B *(6)*	—	—	6Rx + 6eTx or 12eTx	6Rx + 6eTx or 12eTx	12Rx + 12eTx or 24eTx	12Rx + 12eTx or 24eTx
4C *(6)*	6Rx + 6eTx or 12eTx	6Rx + 6eTx or 12eTx	8Rx + 8eTx or 16eTx	8Rx + 8eTx or 16eTx	8Rx + 8eTx or 16eTx	8Rx + 8eTx or 16eTx
5A	8Rx + 8Tx *(7)*	8Rx + 8Tx	12Rx + 12Tx	12Rx + 12Tx	12Rx + 12Tx	12Rx + 12Tx
5B	—	—	—	—	—	6Rx + 6Tx
5C	6Rx + 6Tx	6Rx + 6Tx	10Rx + 10Tx	10Rx + 10Tx	10Rx + 10Tx	10Rx + 10Tx
6A	8Rx + 8Tx *(7)*	8Rx + 8Tx	12Rx + 12Tx	12Rx + 12Tx	12Rx + 12Tx	12Rx + 12Tx
6B	—	—	—	—	—	6Rx + 6Tx
6C	6Rx + 6Tx	6Rx + 6Tx	10Rx + 10Tx	10Rx + 10Tx	10Rx + 10Tx	10Rx + 10Tx
7A *(6)*	10Rx + 10eTx or 20eTx	10Rx + 10eTx or 20eTx	10Rx + 10eTx or 20eTx	10Rx + 10eTx or 20eTx	12Rx + 12eTx or 24eTx	12Rx + 12eTx or 24eTx
7B *(6)*	—	—	6Rx + 6eTx or 12eTx	6Rx + 6eTx or 12eTx	12Rx + 12eTx or 24eTx	12Rx + 12eTx or 24eTx
7C *(6)*	6Rx + 6eTx or 12eTx	6Rx + 6eTx or 12eTx	8Rx + 8eTx or 16eTx	8Rx + 8eTx or 16eTx	8Rx + 8eTx or 16eTx	8Rx + 8eTx or 16eTx
8A *(6)*	10Rx + 10eTx or 20eTx	10Rx + 10eTx or 20eTx	10Rx + 10eTx or 20eTx	10Rx + 10eTx or 20eTx	12Rx + 12eTx or 24eTx	12Rx + 12eTx or 24eTx
8B *(6)*	—	—	6Rx + 6eTx or 12eTx	6Rx + 6eTx or 12eTx	12Rx + 12eTx or 24eTx	12Rx + 12eTx or 24eTx

Table 3–14. LVDS Channels Supported In HardCopy IV E and Stratix IV E Companion Devices with Socket Replacement Flow *(Note 1)*, *(2)* (Part 3 of 3)

	780-Pin FineLine BGA		1152-Pin FineLine BGA		1517-Pin FineLine BGA	
	HardCopy IV E ASIC	Stratix IV E FPGA Prototype	HardCopy IV E ASIC	Stratix IV E FPGA Prototype	HardCopy IV E ASIC	Stratix IV E FPGA Prototype
Bank	HC4E25	EP4SE230 EP4SE360 *(3)*	HC4E35	EP4SE360 EP4SE530 *(4)* EP4SE820 *(4)*	HC4E35	EP4SE530 *(5)* EP4SE820
8C *(6)*	6Rx + 6eTx or 12eTx	6Rx + 6eTx or 12eTx	8Rx + 8eTx or 16eTx	8Rx + 8eTx or 16eTx	8Rx + 8eTx or 16eTx	8Rx + 8eTx or 16eTx

Notes to Table 3–14:

(1) Channel counts are preliminary.

(2) Rx = true LVDS input buffers with OCT RD, Tx = true LVDS output buffers, and eTx = emulated LVDS output buffers (either LVDS_E_1R or LVDS_E_3R).

(3) The EP4SE360 FPGA is offered only in the H780 package.

(4) The EP4SE530 and EP4SE820 FPGAs are offered only in the H1152 package.

(5) The EP4SE530 FPGA is offered only in H1517 package.

(6) Top and bottom I/O banks do not have DPA, synchronizer, data realignment, and differential termination support in Stratix IV E and HardCopy IV E devices. Use left and right I/O banks if these features and maximum performance is required.

(7) When the HardCopy IV E devices mapped to use 780-pin FineLine BGA Wire Bond package, I/O banks 1A, 2A, 5A, and 6A can support 6 pairs of LVDS channel (6RX + 6Tx) only.

Table 3–15. LVDS Channels Supported In HardCopy IV E and Stratix IV E Companion Devices with Non-Socket Replacement Flow *(Note 1)*, *(2)*, *(3)* (Part 1 of 2)

	484-Pin FineLine BGA	780-Pin FineLine BGA
	HardCopy IV E ASIC	Stratix IV E FPGA Prototype
Bank	HC4E25	EP4SE230
1A	6Rx + 6Tx	8Rx + 8Tx
1B	—	—
1C	6Rx + 6Tx	6Rx + 6Tx
2A	6Rx + 6Tx	8Rx + 8Tx
2B	—	—
2C	6Rx + 6Tx	6Rx + 6Tx
3A *(3)*	—	10Rx + 10eTx or 20eTx
3B *(3)*	—	—
3C *(3)*	6Rx + 6eTx or 12eTx	6Rx + 6eTx or 12eTx
4A *(3)*	—	10Rx + 10eTx or 20eTx

Table 3–15. LVDS Channels Supported In HardCopy IV E and Stratix IV E Companion Devices with Non-Socket Replacement Flow *(Note 1)*, *(2)*, *(3)* (Part 2 of 2)

	484-Pin FineLine BGA	780-Pin FineLine BGA
	HardCopy IV E ASIC	Stratix IV E FPGA Prototype
Bank	HC4E25	EP4SE230
4B *(3)*	—	—
4C *(3)*	6Rx + 6eTx or 12eTx	6Rx + 6eTx or 12eTx
5A	6Rx + 6Tx	8Rx + 8Tx
5B	—	—
5C	6Rx + 6Tx	6Rx + 6Tx
6A	6Rx + 6Tx	8Rx + 8Tx
6B	—	—
6C	6Rx + 6Tx	6Rx + 6Tx
7A *(3)*	—	10Rx + 10eTx or 20eTx
7B *(3)*	—	—
7C *(3)*	6Rx + 6eTx or 12eTx	6Rx + 6eTx or 12eTx
8A *(3)*	—	10Rx + 10eTx or 20eTx
8B *(3)*	—	—
8C *(3)*	6Rx + 6eTx or 12eTx	6Rx + 6eTx or 12eTx

Notes to Table 3–15:

(1) Channel counts are preliminary.

(2) Rx = true LVDS input buffers with OCT RD, Tx = true LVDS output buffers, and eTx = emulated LVDS output buffers (either LVDS_E_1R or LVDS_E_3R).

(3) Top and bottom I/O banks do not have DPA, synchronizer, data realignment, and differential termination support in Stratix IV E and HardCopy IV E devices. Use left and right I/O banks if these features and maximum performance is required.

HardCopy IV PLL Planning and Utilization

HardCopy IV devices offer up to 12 PLLs that support the same features as Stratix IV PLLs. The same nomenclature is used for HardCopy IV and Stratix IV PLLs that follow their geographical location in the device floorplan. The PLLs that reside on the top and bottom sides of the device are named `PLL_T1`, `PLL_T2`, `PLL_B1`, and `PLL_B2`; the PLLs that reside on the left and right sides of the device are named `PLL_L1`, `PLL_L2`, `PLL_L3`, `PLL_L4`, `PLL_R1`, `PLL_R2`, `PLL_R3`, and `PLL_R4`, respectively.

Table 3–16 and Table 3–17 show the number of PLLs available in HardCopy IV devices and their companion Stratix IV devices.

Table 3–16. HardCopy IV GX and Stratix IV GX PLL Mapping Options *(Note 1)*

	780-Pin FineLine BGA		780-Pin FineLine BGA		1152-Pin FineLine BGA		1152-Pin FineLine BGA		1517-Pin FineLine BGA	
	HardCopy IV GX ASIC	Stratix IV GX FPGA Prototype	HardCopy IV GX ASIC	Stratix IV GX FPGA Prototype	HardCopy IV GX ASIC	Stratix IV GX FPGA Prototype	HardCopy IV GX ASIC	Stratix IV GX FPGA Prototype	HardCopy IV GX ASIC	Stratix IV GX FPGA Prototype
PLL	HC4GX15	EP4SGX70 EP4SGX110 EP4SGX180 EP4SGX230 EP4SGX290 *(2)* EP4SGX360 *(2)*	HC4GX25	EP4SGX290 *(2)* EP4SGX360 *(2)*	HC4GX25	EP4SGX110 EP4SGX180 EP4SGX230 EP4SGX290 EP4SGX360 EP4SGX530 *(3)*	HC4GX35	EP4SGX230 EP4SGX360 EP4SGX530 *(3)*	HC4GX35	EP4SGX180 EP4SGX230 EP4SGX290 EP4SGX360 EP4SGX530 *(4)*
PLL_L1	—	—	—	—	—	—	—	—	—	—
PLL_L2	✓ *(5)*	✓	✓	✓	✓	✓	✓	✓	✓	✓
PLL_L3	—	—	—	—	—	—	—	—	✓	✓
PLL_L4	—	—	—	—	—	—	—	—	—	—
PLL_T1	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
PLL_T2	—	—	✓	✓	✓	✓	✓	✓	✓	✓
PLL_B1	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
PLL_B2	—	—	✓	✓	✓	✓	✓	✓	✓	✓
PLL_R1	—	—	—	—	—	—	—	—	—	—
PLL_R2	—	—	✓	✓	✓	✓	✓	✓	✓	✓
PLL_R3	—	—	—	—	—	—	—	—	✓	✓
PLL_R4	—	—	—	—	—	—	—	—	—	—

Notes to Table 3–16:

(1) The PLL availability table is preliminary. It is best to design with the Quartus II software to check if your design can use all available PLLs.
(2) The EP4SGX290 and EP4SGX360 FPGAs are offered only in the H780 package.
(3) The EP4SGX530 FPGA is offered only in the H1152 package.
(4) The EP4SGX530 FPGA is offered only in the H1517 package.
(5) The HC4GX15 does not have PLL_L2 if the FPGA EP4SGX290 or EP4SGX360 is selected.

Table 3–17. HardCopy IV E and Stratix IV E PLL Mapping Options *(Note 1)*

PLL	484-Pin FineLine BGA	780-Pin FineLine BGA	780-Pin FineLine BGA		1152-Pin FineLine BGA		1517-Pin FineLine BGA	
	HardCopy IV E ASIC	Stratix IV E FPGA Prototype	HardCopy IV E ASIC	Stratix IV E FPGA Prototype	HardCopy IV E ASIC	Stratix IV E FPGA Prototype	HardCopy IV E ASIC	Stratix IV E FPGA Prototype
	HC4E25	EP4SE230	HC4E25	EP4SE230 EP4SE360 *(2)*	HC4E35	EP4SE360 EP4SE530 *(3)* EP4SE820 *(3)*	HC4E35	EP4SE530 *(4)* EP4SE820
PLL_L1	—	—	—	—	—	—	✓	✓
PLL_L2	✓	✓	✓	✓	✓	✓	✓	✓
PLL_L3	—	—	—	—	✓	✓	✓	✓
PLL_L4	—	—	—	—	—	—	✓	✓
PLL_T1	✓	✓	✓	✓	✓	✓	✓	✓
PLL_T2	—	—	—	—	✓	✓	✓	✓
PLL_B1	✓	✓	✓	✓	✓	✓	✓	✓
PLL_B2	—	—	—	—	✓	✓	✓	✓
PLL_R1	—	—	—	—	—	—	✓	✓
PLL_R2	✓	✓	✓	✓	✓	✓	✓	✓
PLL_R3	—	—	—	—	✓	✓	✓	✓
PLL_R4	—	—	—	—	—	—	✓	✓

Notes to Table 3–17:

(1) The PLL availability table is preliminary. It is best to design with the Quartus II software to check if your design can use all available PLLs.
(2) The EP4SE360 FPGA is offered only in the H780 package.
(3) The EP4SE530 and EP4SE820 FPGAs are offered only in the H1152 package.
(4) The EP4SE530 FPGA is offered only in the H1517 package.

For more information about HardCopy IV PLLs, refer to the *Clock Networks and PLLs in HardCopy IV Devices* chapter.

HardCopy IV Memory Blocks

TriMatrix memory in HardCopy IV devices supports the same memory functions and features as Stratix IV devices. You can independently configure each embedded memory block to be a single- or dual-port RAM, FIFO, ROM, or shift register via the MegaWizard™ Plug-In Manager in the Quartus II software.

HardCopy IV embedded memory consists of MLAB, M9K, and M144K memory blocks, and has one-to-one mapping from Stratix IV memory. However, the number of available memory blocks differs based on physical density, package, and Stratix IV device to HardCopy IV ASIC mapping paths. The Quartus II software may not allow all available Stratix IV memory types to fit into a selected HardCopy IV device if your design has a very high resource utilization and performance target.

Altera recommends that you compile your design with the Quartus II software and verify the device resource guide to check for available resources in the HardCopy IV device.

For information about using the HardCopy Device Resource Guide, refer to the *Quartus II Support for HardCopy Series Devices* chapter in volume 1 of the *Quartus II Handbook*.

Functionally, memory in HardCopy IV and Stratix IV devices is identical. Memory blocks can implement various types of memory with or without parity, including true dual-port, simple dual-port, and single-port RAM, ROM, and FIFO.

MLAB Implementation

In Stratix IV devices, MLABs are dedicated blocks and can be configured for regular logic functions or memory functions. In HardCopy IV devices, MLAB memory blocks are implemented using HCells. The Quartus II software maps the Stratix IV MLAB function to the appropriate memory HCell macro that preserves memory function. This allows you to use the HardCopy IV core fabric more efficiently, freeing up unused HCells for ALM or DSP functions.

MLAB, M9K, and M144K Utilization

HardCopy IV MLAB, M9K, and M144K block functionality is similar to Stratix IV memory blocks; however, you cannot pre-load HardCopy IV MLAB, M9K, and M144K blocks with a **.mif** file when using them as RAM. Ensure that your Stratix IV design does not require **.mif** files if the memory blocks are used as RAM. However, if memory blocks are used as ROM, they are mask-programmed to the design's ROM contents.

You can use the ALTMEM_INIT megafunction to initialize a RAM block after power-up for Stratix IV and HardCopy IV devices. This megafunction reads from a ROM defined with the megafunction and writes to the RAM after power-up. This function allows you to have initialized contents on a RAM block. Refer to the Quartus II Help for implementation information about this function.

Unlike Stratix IV FPGAs, HardCopy IV MLAB, M9K, and M144K RAM contents are unknown after power-up. However, like Stratix IV devices, all HardCopy IV memory output registers power-up cleared, if used. When designing HardCopy IV memory blocks as RAM, Altera recommends a write-before-read of the memory block to avoid reading unknown initial power-up data conditions. If the HardCopy IV memory block is designated as ROM, it powers up with the ROM contents.

One advantage over Stratix IV RAM blocks is that unused M9K and M144K blocks are disconnected from the power rails and MLABs are only implemented as required by your design. These unused resources do not contribute to overall power consumption on HardCopy IV devices.

For a list of supported features in HardCopy IV memory blocks, refer to the *TriMatrix Embedded Memory Blocks in HardCopy IV Devices* chapter.

Using JTAG Features in HardCopy IV Devices

HardCopy IV ASICs support the same boundary-scan test (BST) functionality as Stratix IV FPGAs. However, no reconfiguration is possible because HardCopy IV devices are mask-programmed. Therefore, HardCopy IV devices do not support instructions to reconfigure the device through the JTAG pins. HardCopy IV boundary scan lengths also differ from Stratix IV devices.

For information about HardCopy IV JTAG functionality and support, refer to the *IEEE 1149.1 JTAG Boundary Scan Testing in HardCopy IV Devices* chapter.

Power-Up and Configuration Pin Compatibility with Stratix IV Devices

When designing a board for both HardCopy IV and Stratix IV devices, most configuration pins required by the Stratix IV device are not required by the HardCopy IV device. The functions of these Stratix IV configuration pins are not carried over to the HardCopy IV companion device because HardCopy IV devices are not programmable. To simplify the board connection for these configuration pins, Altera recommends minimizing the power-up and configuration pins that do not carry over from a Stratix IV device to a HardCopy IV device. You should ensure that the board can be used for both Stratix IV and HardCopy IV devices. Configuration pins for both devices must be properly connected. Otherwise, separate boards are required for the two devices.

Table 3–18 lists the main and optional functions on the configuration pins used by Stratix IV and HardCopy IV devices.

Table 3–18. Mapping Configuration Pins into HardCopy IV Devices (Part 1 of 2) *(Note 1), (2), (3), (4)*

Stratix IV FPGA Prototype		HardCopy IV ASIC		Board Connection
Main Function	**Optional Function**	**Main Function**	**Optional Function**	
MSEL[2..0]	—	—	—	Not required and no connection on board.
nCONFIG *(5)*	—	nCONFIG	—	Required connection.
I/O pin	DATA0	I/O pin	DATA0	DATA[0] retains both user I/O and optional EPCS access functions. DATA [7..1] retains user I/O functions only.
I/O pin	DATA [7..1]	I/O pin	—	
DCLK	—	DCLK	—	No Connection on Board, except when EPCS access is required in user mode.
I/O pin	INIT_DONE *(6)*	I/O pin	INIT_DONE	Retains the same I/O functions from Stratix IV.
I/O pin	CLKUSR	I/O pin	—	Retains the same I/O functions from Stratix IV except CLKUSR, because no device programming is required.
nSTATUS *(5)*	—	nSTATUS	—	Required connection.
CONF_DONE *(5)*	—	CONF_DONE	—	Required connection.
nCE	—	nCE	—	Required connection.
nCEO	—	nCEO	—	Not required and no connection on board.
PORSEL	—	PORSEL	—	Required connection.
I/O pin	ASDO	I/O pin	ASDO	No connection on board, except when EPCS access is required in user mode.

Table 3–18. Mapping Configuration Pins into HardCopy IV Devices (Part 2 of 2) *(Note 1), (2), (3), (4)*

Stratix IV FPGA Prototype		HardCopy IV ASIC		Board Connection
Main Function	**Optional Function**	**Main Function**	**Optional Function**	
I/O pin	nCSO	I/O pin	nCSO	No connection on board, except when EPCS access is required in user mode.
nIO_PULLUP	—	nIO_PULLUP	—	Required connection.
I/O pin	CRC_ERROR *(4)*	I/O pin	—	Retains the same I/O functions from Stratix IV, but not CRC_ERROR, because no device programming is required.
I/O pin	DEV_CLRn	I/O pin	DEV_CLRn	Retains the same I/O functions from Stratix IV.
I/O pin	DEV_OE	I/O pin	DEV_OE	Retains the same I/O functions from Stratix IV.

Notes to Table 3–18:

(1) For correct operation of a HardCopy IV device, pull the nSTATUS, nCONFIG, and CONF_DONE pins to V_{CCPGM}. In HardCopy IV devices, these pins are designed with weak internal resistors pulled up to V_{CCPGM}. Stratix IV configuration schemes require pull-up resistors on these I/O pins, so they may already be present on the board. You can remove these external pull-up resistors, if doing so does not affect other FPGAs on the board.

(2) HardCopy IV devices have a maximum V_{CCIO} voltage of 3.0 V, but the input I/O pin can tolerate a 3.3 V level. This applies to V_{CCPGM} voltage and all dedicated and dual-purpose pins.

(3) For HardCopy IV devices, there is weak pull-up on the nSTATUS, CONF_DONE, nCONFIG, and DCLK pins. Therefore, these pins can be left floating or remain connected to external pull-up resistors. If the EPCS is used in user mode as a boot-up RAM or data access for a Nios® II processor, DCLK, DATA[0], ASDO, and nCSO must be connected to the EPCS device.

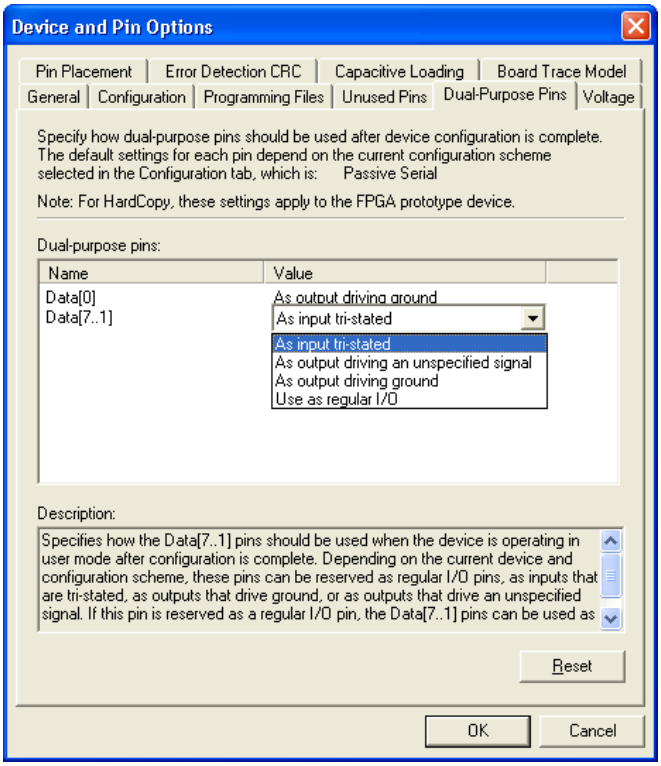
(4) In HardCopy IV devices, CRC_ERROR is hard-wired to logic 0 if the CRC feature is enabled in Stratix IV devices.

(5) The PORSEL pin setting delays the POR sequence for both HardCopy IV and Stratix IV devices.

(6) The INIT_DONE settings option is mask-programmed into the device. You must submit these settings to Altera with the final design prior to mapping to a HardCopy IV device. The use of the INIT_DONE option and other dual-purpose pins (for example, DEV_CLRn device-wide reset and DEV_OE device-wide output enable) are available in the **Fitter Device Options** section of the Quartus II report file.

For both the Stratix IV and HardCopy IV devices, the Quartus II software allows you to set the I/O pins listed in Table 3–18 as dual-purpose pins (as shown in Figure 3–2). As dual-purpose pins, they have I/O functionality when the device enters user mode (when INIT_DONE is asserted).

Figure 3–2. Device and Pin Options Dialog Box



If these dual-purpose pins are required to configure the Stratix IV device, but will be unused after configuration, these pins remain unused on the HardCopy IV device. It is important to consider the state of these pins after power-up and when the device is in user mode. For example, when replacing the Stratix IV device with a HardCopy IV device, these pins may be left floating when the configuration device is removed if you assign such pins as inputs. In this case, you will either require an external means to drive them to a stable level, or set the pins to output driving ground.

Revision History

Table 3–19 shows the revision history for this document.

Table 3–19. Document Revision History

Date	Version	Changes Made
January 2010	2.1	Updated Table 3–1, Table 3–2, Table 3–4, Table 3–5, Table 3–6, Table 3–8, Table 3–11, Table 3–13, Table 3–14, Table 3–15, Table 3–17, and Table 3–18.
June 2009	2.0	■ Added Table 3–13. ■ Updated the following tables: Table 3–1, Table 3–5, Table 3–8, Table 3–12, Table 3–17, Table 3–18, ■ Updated Figure 3–1. ■ Updated "HardCopy IV and Stratix IV Mapping Options" and "Mapping HardCopy IV and Stratix IV I/Os and Modular I/O Banks." ■ Removed "Referenced Documents."
December 2008	1.0	Initial release.

4. Matching Stratix IV Power and Configuration Requirements with HardCopy IV Devices

HIV52004-2.0

This chapter describes power-up options for both HardCopy® IV E and HardCopy IV GX devices and provides examples of how to replace FPGAs in the system with the HardCopy IV E and HardCopy IV GX devices. Both variants of HardCopy IV devices have the same power-up options.

Configuring an FPGA is the process of loading the design data into the device. The Altera® SRAM-based Stratix® IV FPGA requires configuration each time the device is powered up. After the device is powered down, the configuration data within the Stratix IV device is lost and must be loaded again on power up.

HardCopy IV devices are mask-programmed and do not require configuration. One of the advantages of HardCopy IV devices is their instant-on capability upon power up. In addition, there are options to increase delay to postpone HardCopy IV devices power up.

HardCopy IV Power-Up Options

HardCopy IV devices feature two power-up modes:

- Instant On (no added delay)
- Instant On After 50ms Delay

The intent of the power-up modes is to give customers the option of choosing between Instant On and Instant On After 50ms Delay.

Instant On mode is the fastest power-up option on a HardCopy IV device. This mode is used when the HardCopy IV device powers up independently while other components on the board require initialization and configuration. Therefore, you must verify all signals that propagate to and from the HardCopy IV device (for example, reference clocks and other input pins) are stable and do not interrupt HardCopy IV device operation.

Some customers use Instant On After 50ms Delay mode because the system might require the FPGA and HardCopy IV devices to wait until a neighboring processor initializes completely. This mode holds the design in reset for 50ms prior to startup. In addition to the considerations of the system, the software expects the delay from the FPGA device, which will now be replaced with the HardCopy device, and in these cases, customers choose this option.

You must choose the power-up option when submitting the design database to Altera for HardCopy IV devices. After the HardCopy IV devices are manufactured, the power-up option cannot be changed.

Instant On mode is the traditional power-up scheme of most ASIC and non-volatile devices. Similar to Stratix IV devices, HardCopy IV devices go through four phases before transitioning to user mode. However, because HardCopy IV devices do not require configuration, the configuration phase is replaced by a delay phase with either no added delay or a 50 ms delay.

The four phases are listed in order:

- Power-up phase
- Initialization phase
- Delay phase (replacing the configuration phase)
- Start-up phase

Instant On (No Added Delay)

In Instant On mode, after the power supplies ramp up above the HardCopy IV device's power-on reset (POR) trip point, the device initiates an internal POR sequence. After t_{POR} (as shown in Figure 4–1 and Figure 4–2), the power-up phase is complete and the HardCopy IV device transitions to an initialization phase, which releases the CONF_DONE signal to be pulled high. Pulling the CONF_DONE signal high indicates that the HardCopy IV device is nearly ready for normal operation. For more information, refer to Figure 4–1.

During the power-up sequence, weak internal pull-up resistors can pull the user I/O pins high. When the power-up and initialization phases are complete, the I/O pins are released. If the nIO_pullup pin transitions high, the weak pull-up resistors are disabled.

You can find the value of the internal weak pull-up resistors on the I/O pins in the Operating Conditions table of the specific FPGA's device handbook.

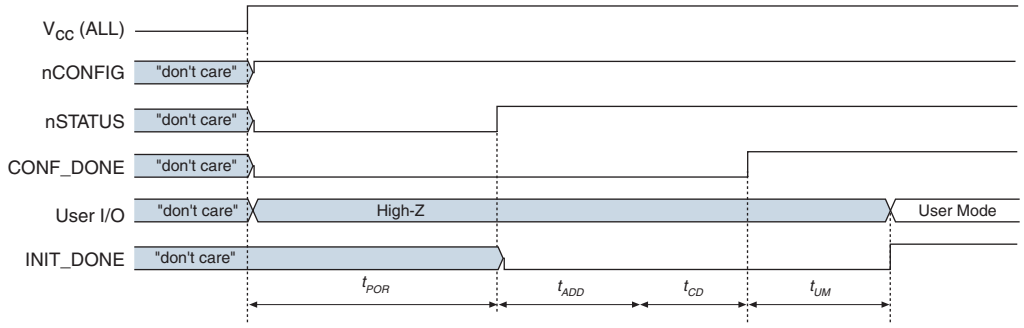
Instant On After 50 ms Delay

The Instant On After 50 ms delay mode is similar to Instant On mode. However, the device waits an additional 50 ms during delay phase before releasing the CONF_DONE pin. This delay is created by an on-chip oscillator. This option is beneficial if other devices on the board (such as a microprocessor) must be initialized prior to the normal operation of the HardCopy IV device.

A start-up phase occurs immediately after the internal registers are reset, all PLLs used are initialized, and any I/O pins used are enabled as the device transitions into user mode.

Figure 4–1 shows an Instant On power-up waveform in which the HardCopy IV device is powered up and the nCONFIG, nSTATUS, and CONF_DONE pins are driven high externally.

Figure 4–1. Timing Waveform for Instant On Option *(Note 1)*, *(2)*, *(3)*, *(4)*, *(5)*



Notes to Figure 4–1:

(1) V_{CC} (ALL) represents either all the power pins or the last power pin powered up to specified operating conditions.

(2) The nCONFIG, nSTATUS, and CONF_DONE pins are weakly pulled high by an external 10 K ohm resistor to V_{CCPGM}; they must be high for this waveform to apply.

(3) User I/O pins may be tri-stated or driven before and during power-up. The nIO_pullup pin can affect the state of the user I/O pins during the initialization phase.

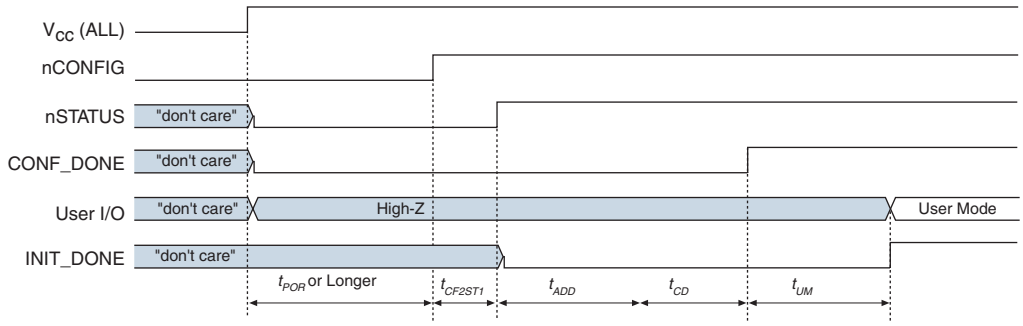
(4) INIT_DONE is an optional pin that can be enabled on the FPGA using the Quartus® II software. HardCopy IV devices carry over the INIT_DONE functionality from the prototyped FPGA design.

(5) The nCEO pin is asserted at approximately the same time as the CONF_DONE pin is released. However, the nCE pin must be driven low externally for this waveform to apply.

Figure 4–2 shows an alternative to the power-up waveform shown in Figure 4–1. The nCONFIG pin is externally held low longer than the PORSEL delay. This delays the initialization sequence by a small amount.

In addition, Figure 4–2 shows an Instant On power-up waveform where nCONFIG is momentarily held low and nSTATUS and CONF_DONE are driven high externally.

Figure 4–2. Timing Waveform for Instant On Option Where nCONFIG is Held Low After Power-Up



Notes to Figure 4–2:

(1) This waveform applies if nCONFIG is held low longer than t_{POR} delay.

(2) V_{CC} (ALL) represents either all the power pins or the last power pin powered up to specified operating conditions. All HardCopy IV power pins must be powered within specifications.

(3) The nCONFIG, nSTATUS, and CONF_DONE pins are weakly pulled high by an external 10 K ohm resistor to V_{CCPGM}; they must be high for this waveform to apply.

(4) User I/O pins may be tri-stated or driven before and during power-up.

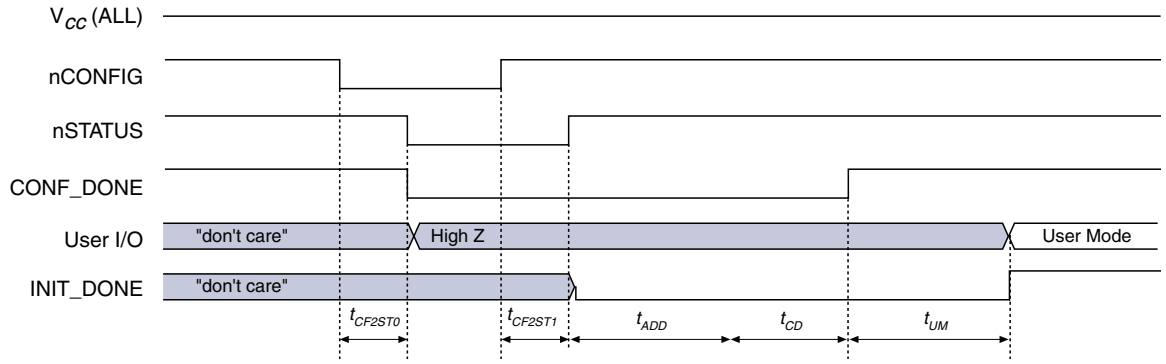
(5) INIT_DONE is an optional pin that can be enabled on the FPGA using the Quartus II software. HardCopy IV devices carry over the INIT_DONE functionality from the prototype FPGA design.

(6) The nCEO pin is asserted at approximately the same time as the CONF_DONE pin is released. However, the nCE pin must be driven low externally for this waveform to apply. Pulsing the nCONFIG signal on an FPGA re-initializes the configuration sequence. The nCONFIG signal on a HardCopy IV device also restarts the initialization sequence.

Pulsing the nCONFIG signal on an FPGA re-initializes the configuration sequence. This feature is the same for HardCopy ASIC devices; pulse the nCONFIG signal on a HardCopy ASIC device to restart the POR sequence.

Figure 4–3 shows the instant-on behavior of the configuration signals and user I/O pins if the nCONFIG pin is pulsed while the V_{CC} supplies are already powered up and stable.

Figure 4–3. Timing Waveform for the Instant-On Option When Pulsing the nCONFIG Signal *(Note 1)*, *(2)*, *(3)*, *(4)*, *(5)*



Notes to Figure 4–3:

(1) V_{CC} (ALL) represents either all the power pins or the last power pin powered up to specified operating conditions. All HardCopy IV power pins must be powered within specifications as described in *Hot Socketing* sections.

(2) The nSTATUS and CONF_DONE pins must not be driven low externally for this waveform to apply.

(3) The nIO_pullup pin can affect the state of the user I/O pins during the initialization phase.

(4) INIT_DONE is an optional pin that can be enabled on the FPGA using the Quartus II software. HardCopy IV devices carry over the INIT_DONE functionality from the prototyped FPGA design.

(5) The nCEO pin is asserted at approximately the same time as the CONF_DONE pin is released. However, the nCE pin must be driven low externally for this waveform to apply.

For more information about HardCopy IV configuration specifications, refer to the *Hot Socketing and Power-On Reset* chapter in volume 1 of the *HardCopy IV Device Handbook*.

Figure 4–4 shows the instant-on behavior of the configuration signals and user I/O pins if the nCONFIG pin is pulsed while the V_{CC} supplies are already powered up and stable.

Configuration Pin Compatibility

When designing a board for both a Stratix IV prototype device and its companion HardCopy IV device, most configuration pins required by the Stratix IV device are not required by the HardCopy IV device. The programmable capabilities of these configuration pins in Stratix IV devices cannot carry over to the HardCopy IV companion device because the HardCopy IV device is not programmable. To simplify the board connection for these configuration pins, Altera recommends minimizing the power-up and configuration pins that do not carry over from the Stratix IV device to the HardCopy IV device.

Table 4–1 lists the dedicated and optional configuration pins for Stratix IV and HardCopy IV devices. If the HardCopy IV device uses the pins' optional function found in Stratix IV devices, the Quartus II software allows you to set these pins as dual-purpose pins. As dual-purpose pins, they have I/O functionality after power up and initialization. These pins only switch to their I/O designation when the device enters user mode (when INIT_DONE is asserted). The design may require that some signals be present when the device transitions into user mode; therefore, it is important to consider the state of these pins after power up and after the device is in user mode when designing the board and selecting the state of dual-purpose pins.

Table 4–1. Configuration Pin Compatibility *(Note 1), (2), (3)*

Stratix IV		HardCopy IV
Pin Name	**Function**	**Board Connection**
MSEL [2..0]	Dedicated	No connect on board
nCONFIG *(5)*	Dedicated	Required connection
DATA [7..0]	Dual-Purpose	DATA[0] retains both user I/O and optional EPCS access functions. DATA[7..1] retains user I/O functions only
DCLK	Dedicated	No connect on board, except when EPCS access is required in user mode
INIT_DONE *(6)*	Dual-Purpose (Optional)	Retains the same I/O functions from the Stratix IV device
CLKUSR	Dual-Purpose (Optional)	Retains the same I/O functions from the Stratix IV device
nSTATUS *(5)*	Dedicated	Required connection
CONF_DONE *(5)*	Dedicated	Required connection
nCE	Dedicated	Required connection
nCEO	Dedicated	Required connection
PORSEL *(5)*	Dedicated	Required connection
ASDO	Dedicated	No connect on board, except when EPCS access is required in user mode
nCSO	Dedicated	No connect on board, except when EPCS access is required in user mode
nIO_PULLUP	Dedicated	Required connection
CRC_ERROR *(4)*	Dual-Purpose (Optional)	Retains the same I/O functions from the Stratix IV device, but not CRC_ERROR because no device programming is needed.
DEV_CLRn	Dual-Purpose (Optional)	Retains the same I/O functions from Stratix IV
DEV_OE	Dual-Purpose (Optional)	Retains the same I/O functions from Stratix IV

Notes to Table 4–1:

(1) For correct operation of the HardCopy IV device, pull the nSTATUS, nCONFIG, and CONF_DONE pins to V_{CCPGM}. In HardCopy IV devices, these pins are designed with weak internal resistors pulled up to V_{CCPGM}. Stratix IV configuration schemes require pull-up resistors on these I/O pins, so they may already be present on the board. You can remove these external pull-up resistors if doing so does not affect other FPGAs on the board.

(2) HardCopy IV devices have a maximum V_{CCIO} voltage of 3.0 V, but the input I/O pin can tolerate a 3.3-V level. This applies to V_{CCPGM} voltage and all dedicated and dual-purpose pins.

(3) For HardCopy IV devices, there is weak pull-up on the nSTATUS, CONF_DONE, nCONFIG, and DCLK pins. Therefore, these pins can be left floating or remain connected to external pull-up resistors. If you use Erasable Programmable Configurable Serial (EPCS) in user mode as a boot-up RAM or data access for a Nios® II processor, DCLK, DATA[0], ASDO, and nCSO need to be connected to the EPCS device.

(4) In HardCopy IV devices, CRC_ERROR is hard-wired to logic 0 if the CRC feature is enabled in Stratix IV devices.

(5) The PORSEL pin setting delays the POR sequence similar to the prototyping FPGA.

(6) The INIT_DONE settings option is mask-programmed into the device. You must submit these settings to Altera with the final design prior to mapping to a HardCopy IV device. Using the INIT_DONE option and other dual-purpose pins (for example, the DEV_CLRn device-wide reset and DEV_OE device-wide output enable) are available in the Fitter Device Options section of the Quartus II report file.

For more information about PORSEL settings for the FPGA, refer to the *Configuration Handbook*.

Most optional configuration pins listed in Table 4–1 on page 4–6 support the various configuration schemes available in Stratix IV FPGAs. Parallel programming and remote update configuration modes use most of the pins in Table 4–1 on page 4–6. HardCopy IV devices are not configurable and do not support configuration emulation mode. Therefore, Altera recommends that you minimize using the optional functionality of the configuration pin in the Stratix IV design by using another mode such as passive serial configuration mode.

If some of these dual-purpose pins are needed to configure the Stratix IV FPGA, but will be unused after configuration, these pins remain unused on the HardCopy IV device. Therefore, use caution when designing for these pins on the Stratix IV and HardCopy IV boards. The removal of the Stratix IV device and its corresponding configuration device may leave these pins floating on the HardCopy IV device if you assign them as inputs without any external means of driving them to a stable level. When selecting a Stratix IV device and its device options, consider the after-configuration requirements of these pins and set them appropriately in the Quartus II software.

Examples of Mapping a Stratix FPGA Configuration to a HardCopy ASIC

This section provides examples of how HardCopy IV devices replace Stratix IV FPGAs using different configuration schemes.

HardCopy IV Device Replacing a Stand-Alone Stratix IV Device

Figure 4–4 shows the Stratix IV device before it is replaced with the HardCopy IV device. The example in Figure 4–5 shows the single HardCopy IV device replacing a stand-alone Stratix IV device. The configuration device, now redundant, is removed, and no further board changes are necessary. You can remove he pull-up resistors on the nCONFIG, nSTATUS, and CONF_DONE pins.

Figure 4–4. Configuration of a Stand-Alone Stratix IV Device

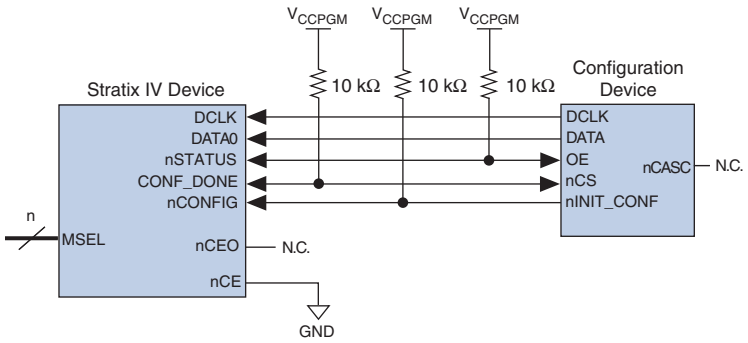
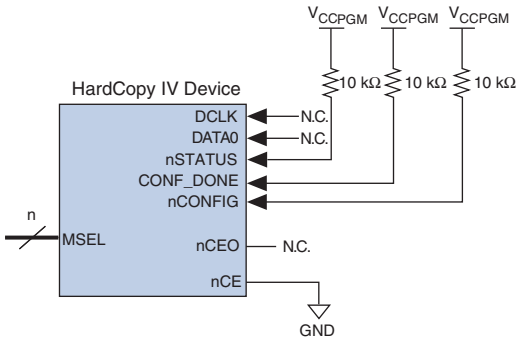


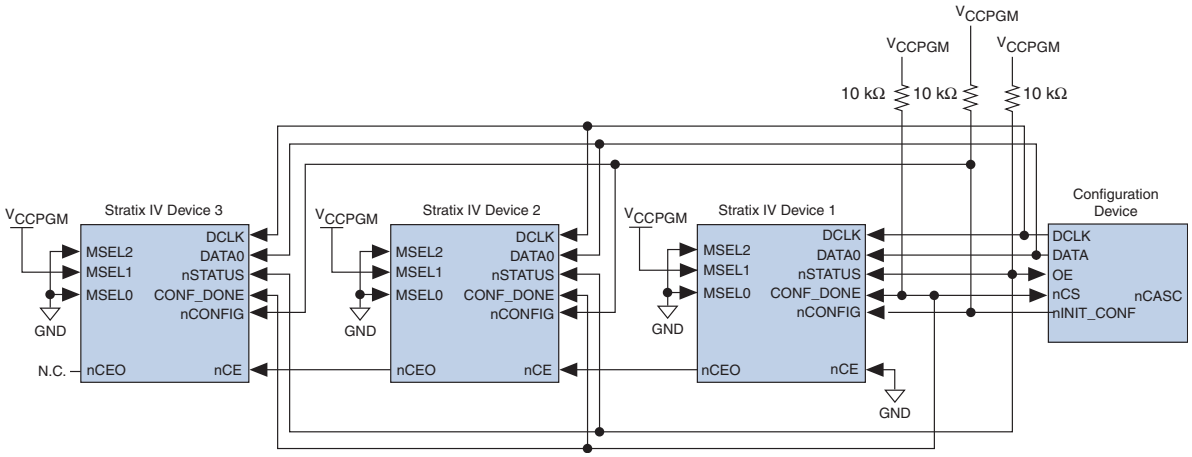
Figure 4–5. HardCopy IV Device Replacing a Stand-Alone Stratix IV Device



HardCopy IV Device Replacing a Stratix IV Device in a Cascaded Configuration Chain

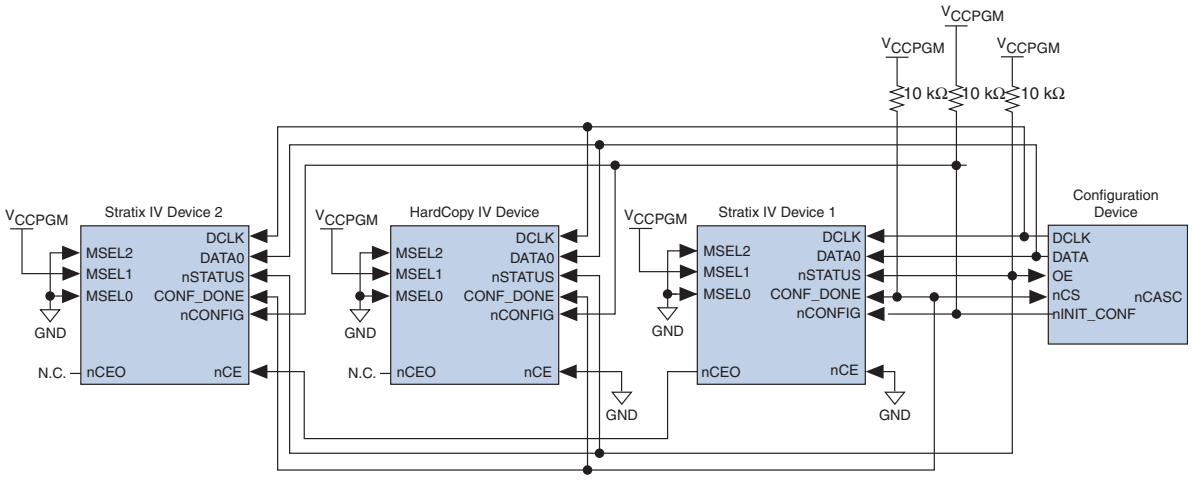
Figure 4–6 shows a design where the configuration data for the FPGAs is stored in a single configuration device and the Stratix IV devices are connected in a multiple-device configuration chain. The second device in the chain is replaced with a HardCopy IV device.

Figure 4–6. Configuration of Multiple FPGAs in a Cascade Chain



To configure FPGAs on a board with both HardCopy IV devices and FPGAs, you must remove the HardCopy IV device from the cascade chain. Figure 4–7 shows how the devices are connected with the HardCopy IV device removed from the chain. The data in the configuration device must be modified to exclude the HardCopy IV device configuration data.

Figure 4–7. Configuration with the HardCopy IV Device Removed from the Cascade Chain



Note to Figure 4–7:

(1) The `MSEL[2:0]` pins are not used on the HardCopy IV device but they preserve the pin assignment and direction from the Stratix IV device, allowing drop-in replacement.

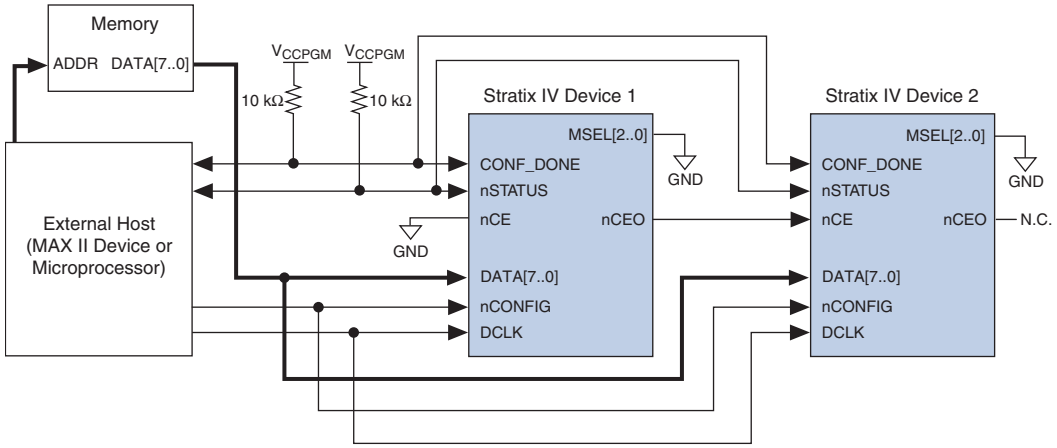
Eliminating the HardCopy IV device from the configuration chain requires the following changes:

- The configuration data stored in the configuration device must be updated to exclude the configuration data for the HardCopy IV device.
- The `nCE` pin of the HardCopy IV device must be tied to GND.
- The `nCE` pin of the FPGA that was driven by the HardCopy IV `nCEO` pin must now be driven by the `nCEO` pin of the FPGA that precedes the HardCopy IV device in the chain.
- The connections of the `MSEL[2:0] pins` are not required.

HardCopy IV Device Replacing a Stratix IV Device Configured with a Microprocessor

When you replace a Stratix IV FPGA with a HardCopy IV device, the microprocessor code must be modified to treat the HardCopy IV device as a non-configurable device. Figure 4–8 shows an example with two Stratix IV devices configured using a microprocessor or MAX® II device and the FPP configuration scheme. This example does not require changes to the board.

Figure 4–8. Multiple-Device FPP Configuration Using a Microprocessor or a MAX II Device

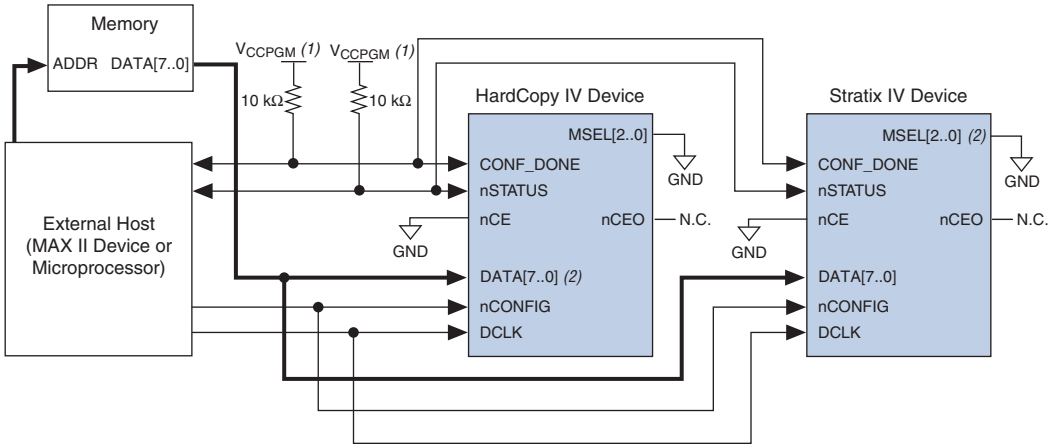


Note to Figure 4–8:

(1) Connect the pull-up resistor to a supply that provides an acceptable input signal for all devices in the chain. The V_{CCPGM} voltage meets the I/O standard's V_{IH} specification on the device and the external host.

Figure 4–9 shows how the first Stratix IV device is replaced by a HardCopy IV device. In this case, the microprocessor code must be modified to send configuration data only to the second device (the Stratix IV device) of the configuration chain. The microprocessor can only send this data after its nCE pin is asserted by the first device (the HardCopy IV device).

Figure 4–9. Replacement of the First FPGA in the FPP Configuration Chain with a HardCopy IV Device *(Note 1)*



Notes to Figure 4–9:

(1) Connect the pull-up resistor to a supply that provides an acceptable input signal for all devices in the chain. The V_{CCPGM} voltage meets the I/O standard's V_{IH} specification on the device and the external host.

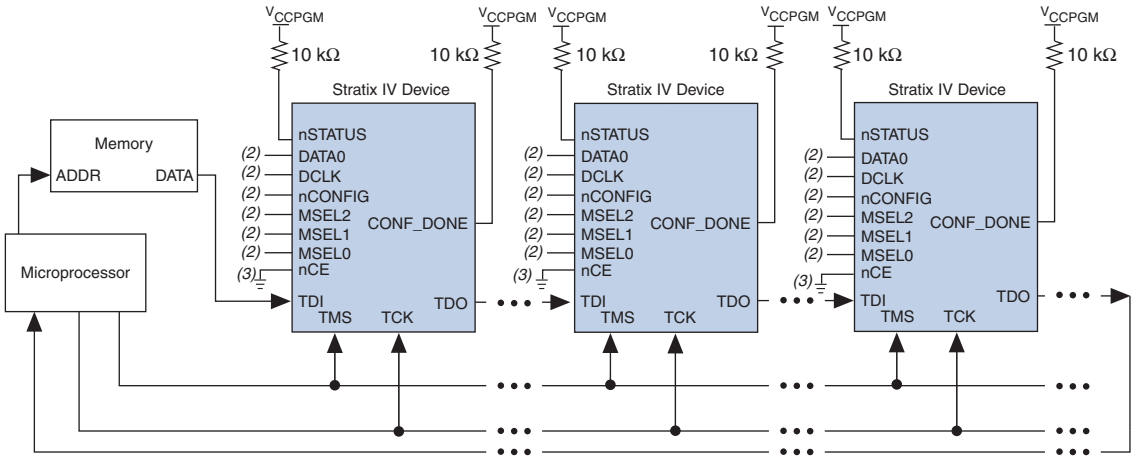
(2) The DATA[7..0] and MSEL[2:0] pins are not used on the HardCopy IV device but they preserve the pin assignment and direction from the Stratix IV device, allowing drop-in replacement.

HardCopy IV Device Replacing an FPGA Configured in a JTAG Chain

In this example, the circuit connectivity is maintained and there are no changes made to the board. You must modify the microprocessor code so that it treats the HardCopy IV device as a non-configurable device. The microprocessor can achieve this by issuing a BYPASS instruction to the HardCopy IV device. With the HardCopy IV device in BYPASS mode, the configuration data passes through it to the downstream FPGAs.

Figure 4–10 shows an example where there are multiple FPGAs. These devices are connected using the JTAG I/O pins for each device and programmed using the JTAG port. An on-board microprocessor generates the configuration data.

Figure 4–10. Configuring FPGAs in a JTAG Chain Using a Microprocessor



Notes to Figure 4–10:

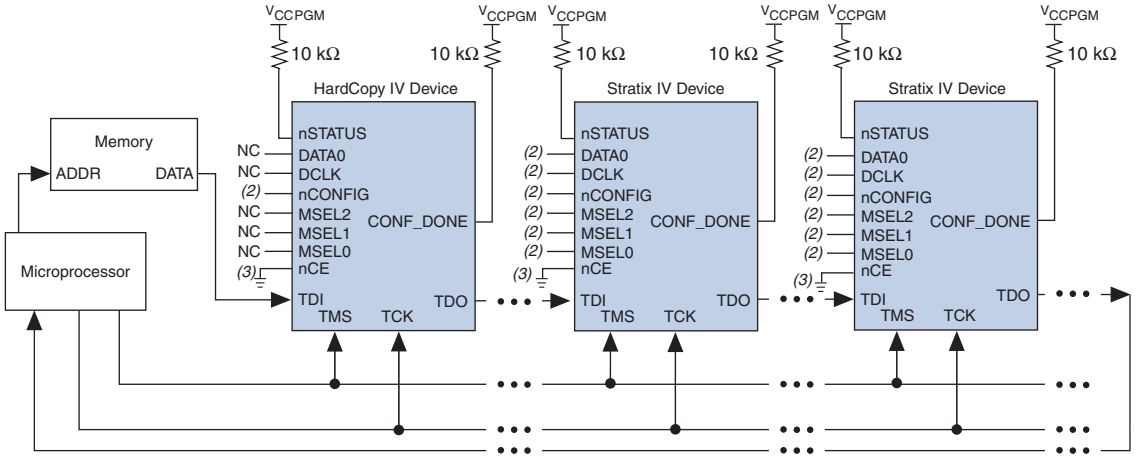
(1) You can place the Stratix IV, Stratix III, Stratix II, Stratix, Cyclone® III, Cyclone II, and Cyclone devices within the same JTAG chain for device programming and configuration.

(2) Connect the nCONFIG, MSEL0, MSEL1, and MSEL2 pins to support a non-JTAG configuration scheme. If you only use JTAG configuration, connect nCONFIG to V_{CCPGM}, and MSEL0, MSEL1, and MSEL2 to GND. Pull DATA0 and DCLK to either high or low.

(3) nCE must be connected to GND or driven low for successful JTAG configuration.

Figure 4–11 shows an example where the first Stratix IV device in the JTAG chain is replaced by a HardCopy IV device.

Figure 4–11. Replacement of the First FPGA in the JTAG Chain with a HardCopy IV Device *(Note 1)*



Notes to Figure 4–11:

(1) You can place the Stratix IV, Stratix III, Stratix II, Stratix, Cyclone III, Cyclone II, and Cyclone devices within the same JTAG chain for device programming configuration.

(2) Connect the nCONFIG, MSEL0, MSEL1, and MSEL2 pins to support a non-JTAG configuration scheme. If you use only JTAG configuration, connect nCONFIG to V_{CCPGM}, and MSEL0, MSEL1, and MSEL2 to GND. Pull DATA0 and DCLK to either high or low.

(3) nCE must be connected to GND or driven low for successful JTAG configuration.

Document Revision History

Table 4–2 shows the revision history for this chapter.

Table 4–2. Document Revision History

June 2009	2.0	Updated for HardCopy IV GX devices.
December 2008	1.0	Initial release.

About this Handbook

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

How to Contact Altera

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

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

Note:

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

Typographic Conventions

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

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

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

HardCopy IV Device Handbook, Volume 3

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

HC4_H5V3-1.0

Contents

Chapter 2. HardCopy IV GX Dynamic Reconfiguration

Chapter 3. HardCopy IV GX ALTGX_RECONFIG Megafunction User Guide

Additional Information

Chapter Revision Dates

The chapters in this book, *HardCopy IV Device Handbook, Volume 3*, were revised on the following dates. Where chapters or groups of chapters are available separately, part numbers are listed.

Chapter 1 HardCopy IV GX Transceiver Architecture
Revised: *June 2009*
Part Number: *HIV53001-1.0*

Chapter 2 HardCopy IV GX Dynamic Reconfiguration
Revised: *June 2009*
Part Number: *HIV53002-1.0*

Chapter 3 HardCopy IV GX ALTGX_RECONFIG Megafunction User Guide
Revised: *June 2009*
Part Number: *HIV53003-1.0*

Section I.
Transceiver Architecture

This section provides a description of transceiver architecture and dynamic reconfiguration for the HardCopy® IV device family. This section includes the following chapters:

- Chapter 1, HardCopy IV GX Transceiver Architecture
- Chapter 2, HardCopy IV GX Dynamic Reconfiguration
- Chapter 3, HardCopy IV GX ALTGX_RECONFIG Megafunction User Guide

Revision History

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

1. HardCopy IV GX Transceiver Architecture

HIV53001-1.0

Introduction

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

HardCopy IV GX devices deliver a very high level of system bandwidth and power efficiency for high-end applications, allowing you to innovate without compromise. HardCopy IV GX devices provide up to 24 full-duplex 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 following serial protocols:

- PCI Express (PIPE)
  - Gen1 at 2.5 Gbps
  - Gen2 at 5 Gbps
- XAUI (3.125 Gbps to 3.75 Gbps for HiGig support)
- GIGE (1.25 Gbps)
- Serial RapidIO (1.25 Gbps, 2.5 Gbps, and 3.125 Gbps)
- SONET/SDH
  - OC-12 at 622 Mbps
  - OC-48 at 2.488 Gbps
  - 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
  - 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 PCI Express hard IP, PCS, and PMA blocks. For more information about the PCI Express (PIPE) hard IP block, refer to the *PCI Express Compiler Guide*.

This chapter includes the following sections:

Transceiver Channel Locations

The 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 1–1 summarizes the total number of transceiver channels and transceiver block locations in each HardCopy IV GX device member.

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

Device Member	Total Number of Transceiver Channels	Transceiver Channel Location
HC4GX15LF780N	8	Eight transceiver channels located in two transceiver blocks, GXBR0 and GXBR1, on the right side of the device.
HC4GX25LF1152N	16	Eight transceiver channels located in two transceiver blocks, GXBR0 and GXBR1, on the right side of the device.
HC4GX25LF780N		Eight transceiver channels located in two transceiver blocks, GXBL0 and GXBL1, on the left side of the device.
HC4GX25FF1152N	24	Eight regular transceiver channels supporting data rates between 600 Mbps and 6.5 Gbps and four CMU channels supporting data rates between 600 Mbps and 6.5 Gbps located in two transceiver blocks, GXBR0 and GXBR1, on the right side of the device.
HC4GX35FF1152N		Eight regular transceiver channels supporting data rates between 600 Mbps and 6.5 Gbps and four CMU channels supporting data rates between 600 Mbps and 6.5 Gbps located in two transceiver blocks, GXBL0 and GXBL1, on the left side of the device.

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

Device Member	Total Number of Transceiver Channels	Transceiver Channel Location
HC4GX35LF1517N	24	Twelve regular transceiver channels located in three transceiver blocks, GXBR0, GXBR1, and GXBR2 on the right side of the device. Twelve regular transceiver channels located in three transceiver blocks, GXBL0, GXBL1, and GXBL2 on the left side of the device.
HC4GX35FF1517N	36	Twelve regular transceiver channels supporting data rates between 600 Mbps and 6.5 Gbps and six CMU channels supporting data rates between 600 Mbps and 6.5 Gbps located in three transceiver blocks, GXBR0, GXBR1, and GXBR2 on the right side of the device. Twelve regular transceiver channels supporting data rates between 600 Mbps and 6.5 Gbps and six CMU channels supporting data rates between 600 Mbps and 6.5 Gbps located in three transceiver blocks, GXBL0, GXBL1, and GXBL2 on the left side of the device.

Figure 1–1 through Figure 1–5 show transceiver channel locations in each HardCopy IV GX device member.

Figure 1–1. HardCopy IV GX Devices with Eight Transceiver Channels

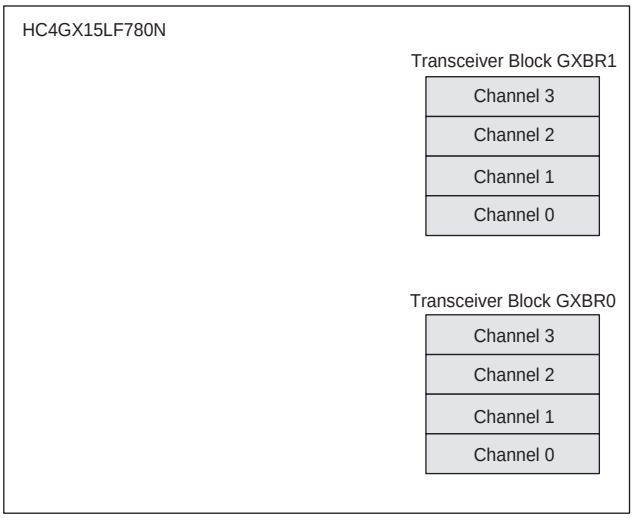


Figure 1–2. HardCopy IV GX Devices with Sixteen Transceiver Channels

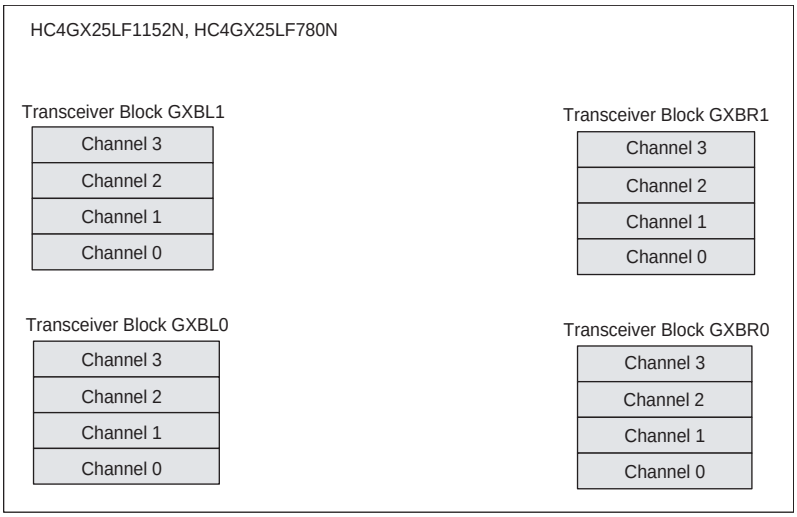


Figure 1–3. HardCopy IV GX Devices with Twenty-Four Transceiver Channels

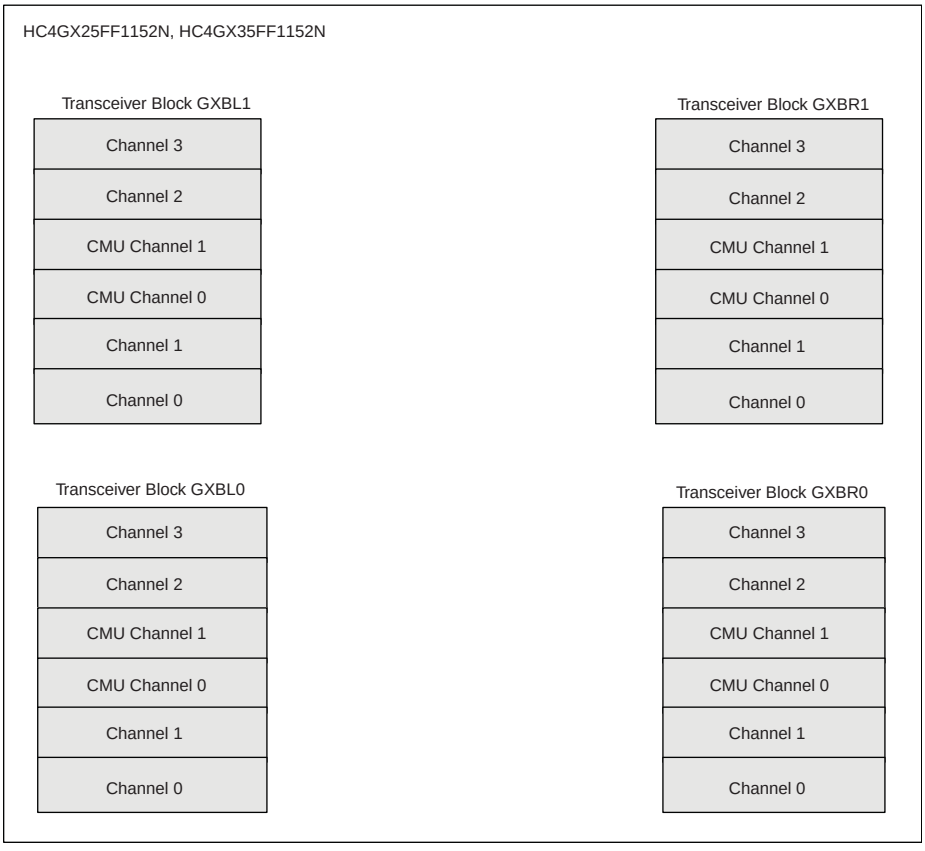


Figure 1–4. HardCopy IV GX Devices with Twenty-Four Transceiver Channels

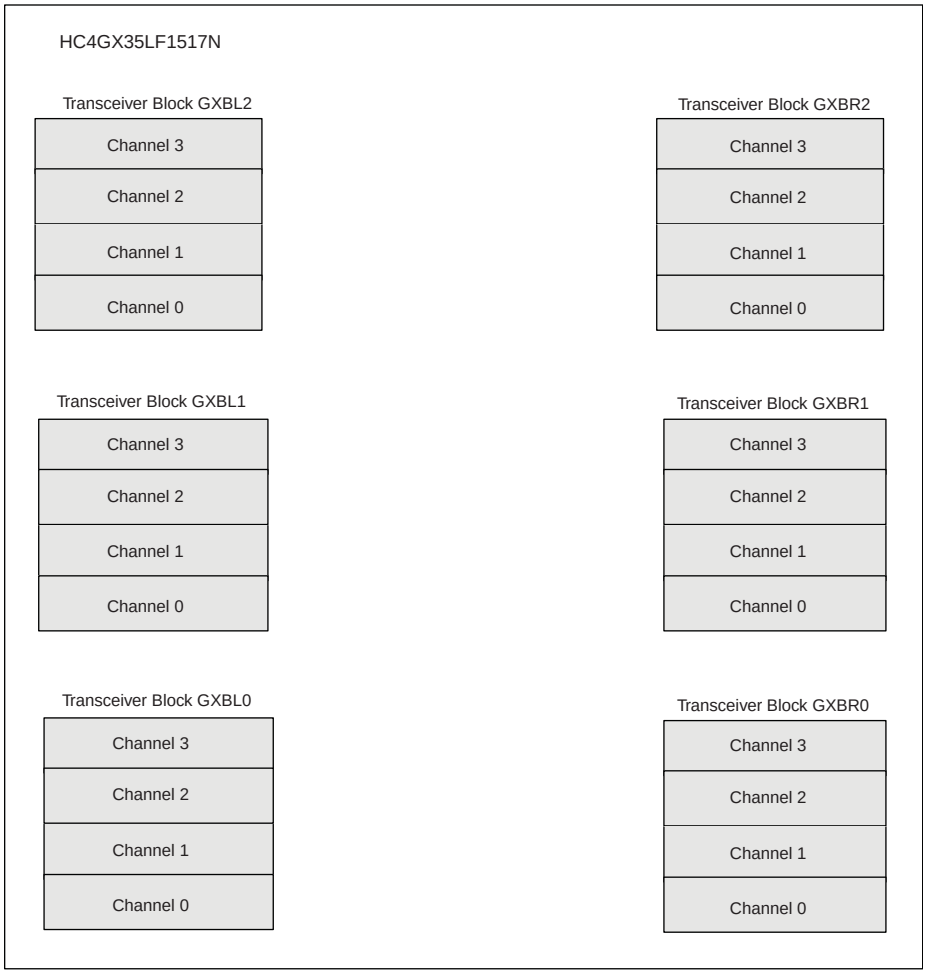
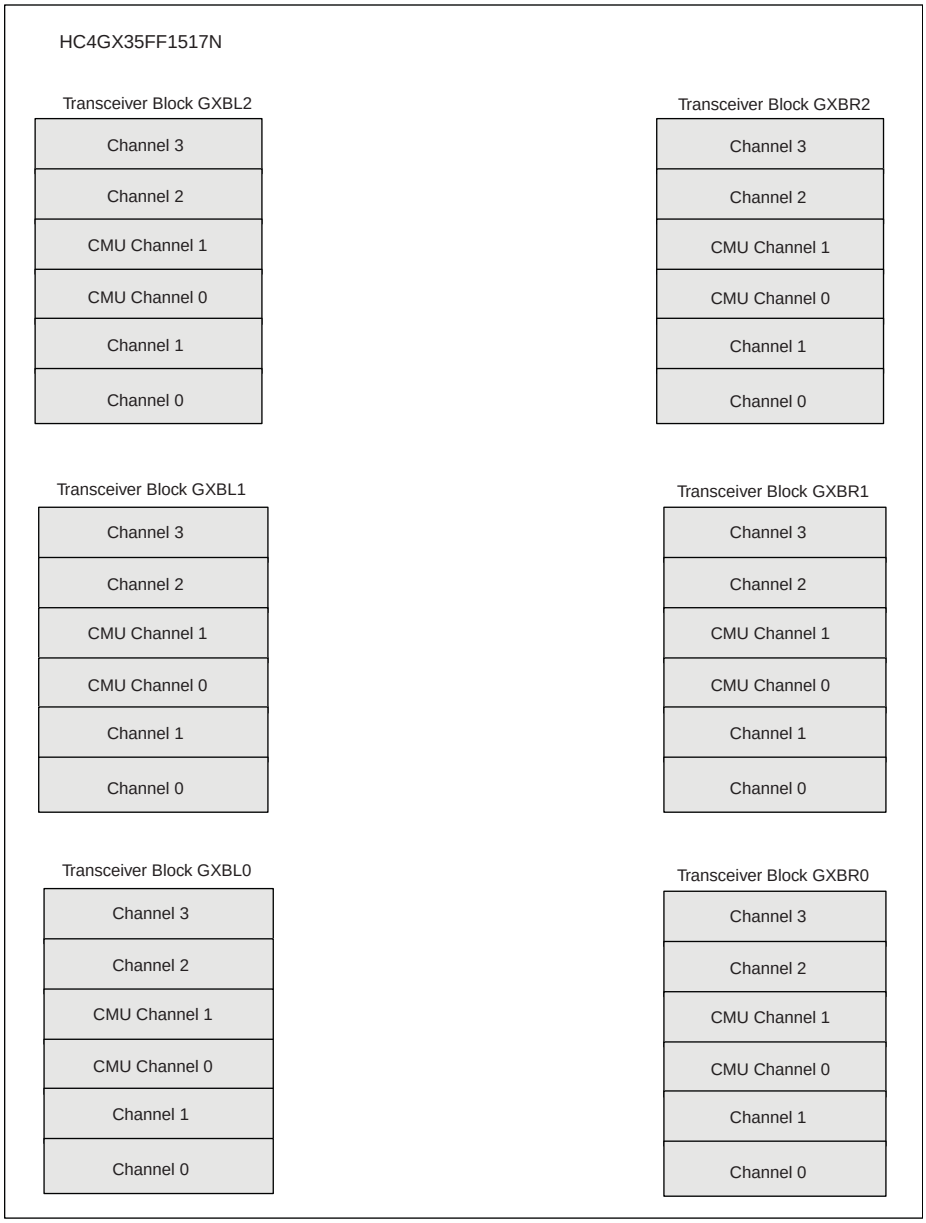


Figure 1–5. HardCopy IV GX Devices with Thirty-Six Transceiver Channels



Transceiver Block Architecture

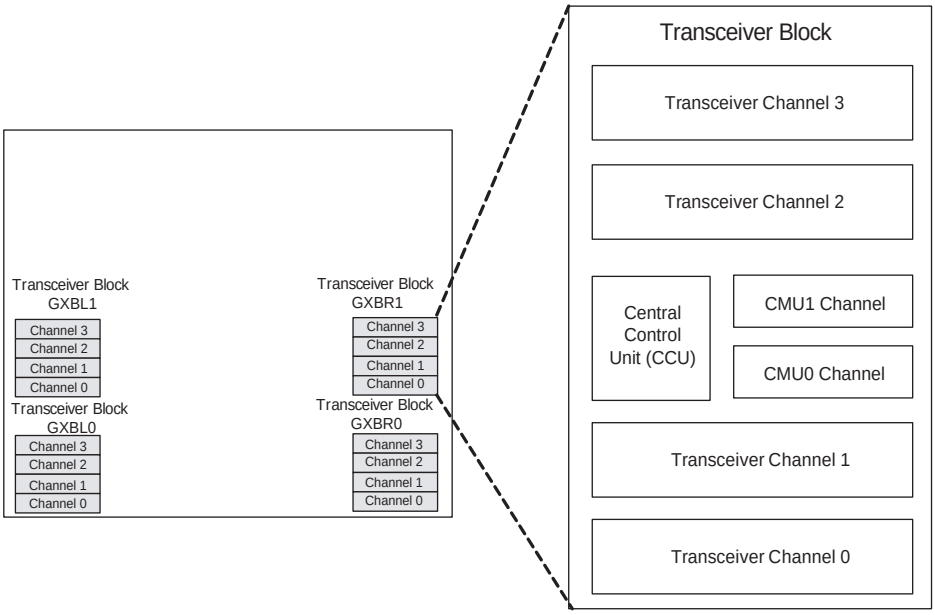
Each transceiver block has the following components:

- Two clock multiplier unit (CMU) channels—the `CMU0` and `CMU1` channels—that provide the high-speed serial and low-speed parallel clock to the transceiver channels
- Four full-duplex (transmitter and receiver) transceiver channels that support serial data rates from 600 Mbps to 6.5 Gbps

- Central control unit (CCU) that implements XAUI state machine for XGMII-to-PCS code group conversion, XAUI deskew state machine, shared control signal generation block, PCI Express (PIPE) rateswitch controller block, and reset control logic
  - The shared control signal generation block provides control signals to the transceiver channels in bonded functional modes such as XAUI, PIPE, and Basic ×4.
  - The PIPE rateswitch controller block controls the rateswitch circuit in the `CMU0` channel, in ×4 configurations. In PIPE ×8 configuration, the PIPE rateswitch controller block of the CCU in the master transceiver block is active. For more information about rateswitch in PIPE, refer to "PCI Express (PIPE) Gen2 (5 Gbps) Support" on page 1–137.

Figure 1–6 shows a block diagram of the transceiver block architecture.

Figure 1–6. Top-Level View of a Transceiver Block



For architecture information about CMU channels and transceiver channels, refer to "CMU Channels" on page 1–22 and "Transceiver Channel Architecture" on page 1–34.

Transceiver Port List

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 1–2 provides a brief description of the ALTGX megafunction ports.

Table 1–2. HardCopy IV GX ALTGX Megafunction Ports (Part 1 of 14)

Port Name	Input/Output	Description	Scope
Clock Multiplier Unit (CMU)			
pll_inclk	Input	Input reference clock for the CMU phase-locked loop (PLL).	Transceiver block
pll_locked	Output	CMU PLL lock indicator. A high level indicates that the CMU PLL is locked to the input reference clock; a low level indicates that the CMU PLL is not locked to the input reference clock. Asynchronous signal.	Transceiver block
pll_powerdown	Input	CMU PLL power down. When asserted high, the CMU PLL is powered down. When de-asserted low, the CMU PLL is active and locks to the input reference clock. Note: Asserting the pll_powerdown signal does not power down the REFCLK buffers. Asynchronous signal. The minimum pulse-width is 1 µs (pending characterization).	Transceiver block
coreclkout	Output	Core fabric-transceiver interface clock. Generated by the CMU0 clock divider in the transceiver block in ×4 bonded channel configurations. Generated by the CMU0 clock divider in the master transceiver block in ×8 bonded channel configurations. Not available in non-bonded channel configurations. This clock is used to clock the write port of the transmitter phase compensation FIFOs in all bonded channels. Use this clock signal to clock parallel data tx_datain from the core fabric into the transmitter phase compensation FIFO of all bonded channels. This clock is used to clock the read port of the receiver phase compensation FIFOs in all bonded channels with rate match FIFO enabled. Use this signal to clock parallel data rx_dataout from the receiver phase compensation FIFOs of all bonded channels (with rate match FIFO enabled) into the core fabric.	Transceiver block

Table 1–2. HardCopy IV GX ALTGX Megafunction Ports (Part 2 of 14)

Port Name	Input/Output	Description	Scope
Receiver Physical Coding Sublayer (PCS) Ports			
Word Aligner			
rx_enapatternalign	Input	Manual word alignment enable control. Enables the word aligner configured in manual alignment mode to align to the word alignment pattern. ■ In single-width modes with 10-bit PMA-PCS interface, this signal is level-sensitive. When high, the word aligner re-aligns if the word alignment pattern appears in a new word boundary. ■ In single-width modes with 8-bit PMA-PCS interface and all double-width modes, a low-to-high transition causes the word aligner to re-align once, if the word alignment pattern appears in a new word boundary. Asynchronous signal. The minimum pulse-width is two recovered clock cycles.	Channel
rx_patterndetect	Output	Word alignment pattern detect indicator. A high level indicates that the word alignment pattern is found on the current word boundary. The width of this signal depends on the following channel width: Channel Width rx_patterndetect 8/10 1 16/20 2 32/40 4	Channel

Table 1–2. HardCopy IV GX ALTGX Megafunction Ports (Part 3 of 14)

Port Name	Input/Output	Description	Scope
rx_syncstatus	Output	Word alignment synchronization status indicator. For the word aligner in automatic synchronization state machine mode, this signal is driven high if the conditions required to remain in synchronization are met. It is driven low if the conditions required to lose synchronization are met. For the word aligner in manual alignment mode, the behavior of this signal depends on whether the transceiver is configured in single-width or double-width mode. For more information, refer to "Word Aligner in Single-Width Mode" on page 1–74 and "Word Aligner in Double-Width Mode" on page 1–80. This signal is not available for the word aligner in bit-slip mode. The width of this signal depends on the following channel width: Channel Width rx_syncstatus 8/10 1 16/20 2 32/40 4	Channel
rx_bitslip	Input	Bit-slip control for the word aligner configured in bit-slip mode. At every rising edge of this signal, word aligner slips one bit into the received data stream, effectively shifting the word boundary by 1 bit. Asynchronous signal. The minimum pulse-width is two recovered clock cycles.	Channel
rx_ala2size	Input	Available only in SONET OC-12 and OC-48 modes to select between one of the following two word alignment options: ■ 0 -16-bit A1A2 ■ 1 - 32-bit A1A1A2A2	Channel
rx_rlv	Output	Run-length violation indicator. A high pulse is driven when the number of consecutive 1s or 0's in the received data stream exceeds the programmed run length violation threshold. 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.	Channel

Table 1–2. HardCopy IV GX ALTGX Megafunction Ports (Part 4 of 14)

Port Name	Input/Output	Description	Scope
rx_invpolarity	Input	Generic receiver polarity inversion control. Useful feature for correcting situations where the positive and negative signals of the differential serial link are accidentally swapped during board layout. When asserted high in single-width modes, the polarity of every bit of the 8-bit or 10-bit input data word to the word aligner gets inverted. When asserted high in double-width mode, the polarity of every bit of the 16-bit or 20-bit input data to the word aligner gets inverted. Asynchronous signal.	Channel
rx_revbitorderwa	Input	Receiver bit reversal control. This is available only in Basic single-width and double-width modes with the word aligner configured in bit-slip mode. This is a useful feature where the link transmission order is MSBit to LSBit. When asserted high in Basic single-width modes, the 8-bit or 10-bit data D[7:0] or D[9:0] at the output of the word aligner gets rewired to D[0:7] or D[0:9], respectively. When asserted high in Basic double-width modes, the 16-bit or 20-bit data D[15:0] or D[19:0] at the output of the word aligner gets rewired to D[0:15] or D[0:19], respectively. Asynchronous signal.	Channel
rx_revbyteorderwa	Input	Receiver byte reversal control. This is available only in Basic double-width mode. This is a useful feature in situations where the MSByte and LSByte of the transmitted data are erroneously swapped. When asserted high, the MSByte and LSByte of the 16 and 20 bit data at the output of the word aligner get swapped. Asynchronous signal.	Channel

Table 1–2. HardCopy IV GX ALTGX Megafunction Ports (Part 5 of 14)

Port Name	Input/Output	Description	Scope
Deskew FIFO			
rx_channelaligned	Output	Ten-gigabit attachment unit interface (XAUI) deskew FIFO channel aligned indicator. Available only in XAUI mode. A high level indicates that the XAUI deskew state machine is either in ALIGN_ACQUIRED_1, ALIGN_ACQUIRED_2, ALIGN_ACQUIRED_3, or ALIGN_ACQUIRED_4 state, as specified in the PCS deskew state diagram in the IEEE P802.3ae specification. A low level indicates that the XAUI deskew state machine is either in LOSS_OF_ALIGNMENT, ALIGN_DETECT_1, ALIGN_DETECT_2, or ALIGN_DETECT_3 state, as specified in the PCS deskew state diagram in the IEEE P802.3ae specification.	Transceiver block
Rate Match (Clock Rate Compensation) FIFO			
rx_rmfifodatainserted	Output	Rate match FIFO insertion status indicator. A high level indicates that the rate match pattern byte has inserted to compensate for the parts-per-million (PPM) difference in reference clock frequencies between the upstream transmitter and the local receiver.	Channel
rx_rmfifodatadeleted	Output	Rate match FIFO deletion status indicator. A high level indicates that the rate match pattern byte got deleted to compensate for the PPM difference in reference clock frequencies between the upstream transmitter and the local receiver.	Channel
rx_rmfifofull	Output	Rate match FIFO full status indicator. A high level indicates that the rate match FIFO is full. 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.	Channel
rx_rmfifoempty	Output	Rate match FIFO empty status indicator. A high level indicates that the rate match FIFO is empty. 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.	Channel

Table 1–2. HardCopy IV GX ALTGX Megafunction Ports (Part 6 of 14)

Port Name	Input/Output	Description	Scope
8B/10B Decoder			
rx_ctrldetect	Output	Receiver control code indicator. Available in configurations with 8B/10B decoder. A high level indicates that the associated received code group is a control (/Kx.y/) code group. A low level indicates that the associated received code group is a data (/Dx.y/) code group. The width of this signal depends on the following channel width: Channel Width rx_ctrldetect 8 1 16 2 32 4	Channel
rx_errdetect	Output	8B/10B code group violation or disparity error indicator. Available in configurations with 8B/10B decoder. A high level indicates that a code group violation or disparity error was detected on the associated received code group. Use with the rx_disperr signal to differentiate between a code group violation and/or a disparity error as follows: [rx_errdetect: rx_disperr] 2'b00—no error 2'b10—code group violation 2'b11—disparity error or both The width of this signal depends on the following channel width: Channel Width rx_errdetect 8 1 16 2 32 4	Channel
rx_disperr	Output	8B/10B disparity error indicator port. Available in configurations with 8B/10B decoder. A high level indicates that a disparity error was detected on the associated received code group. The width of this signal depends on the following channel width: Channel Width rx_disperr 8 1 16 2 32 4	Channel

Table 1–2. HardCopy IV GX ALTGX Megafunction Ports (Part 7 of 14)

Port Name	Input/Output	Description	Scope
rx_runningdisp	Output	8B/10B running disparity indicator. This feature is available in configurations with the 8B/10B decoder. A high level indicates that data on the rx_dataout port was received with a negative running disparity. A low level indicates that data on the rx_dataout port was received with a positive running disparity. The width of this signal depends on the following channel width: Channel Width — rx_runningdisp 8 — 1 16 — 2 32 — 4	Channel
Byte Ordering Block			
rx_enabyteord	Input	Enable byte ordering control. This feature is available in configurations with the byte ordering block enabled. The byte ordering block is rising-edge sensitive to this signal. A low-to-high transition triggers the byte ordering block to restart the byte ordering operation. Asynchronous signal.	Channel
rx_byteorderalignstatus	Output	Byte ordering status indicator. This feature is available in configurations with the byte ordering block enabled. A high level indicates that 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	Parallel data output from the receiver to the core fabric. The bus width depends on the channel width multiplied by the number of channels per instance.	Channel
rx_clkout	Output	Recovered clock from the receiver channel. This feature is available only when the rate match FIFO is not used in the receiver datapath.	Channel
rx_coreclk	Input	Optional read clock port for the receiver phase compensation FIFO. If not selected, the Quartus II software automatically selects rx_clkout/tx_clkout/coreclkout as the read clock for the receiver phase compensation FIFO. If selected, you must drive this port with a clock that has 0 PPM difference with respect to rx_clkout/tx_clkout/coreclkout.	Channel

Table 1–2. HardCopy IV GX ALTGX Megafunction Ports (Part 8 of 14)

Port Name	Input/Output	Description	Scope
rx_phase_comp_fifo_error	Output	Receiver phase compensation FIFO full or empty indicator. A high level indicates that 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	Input reference clock for the receiver clock and data recovery.	Channel
rx_pll_locked	Output	Receiver CDR lock-to-reference (LTR) indicator. A high level indicates that the receiver CDR is locked to the input reference clock. A low level indicates that the receiver CDR is not locked to the input reference clock. Asynchronous signal.	Channel
rx_freqlocked	Output	Receiver CDR lock mode indicator. A high level indicates that the receiver CDR is in lock-to-data (LTD) mode. A low level indicates that the receiver CDR is in lock-to-reference mode. Asynchronous signal.	Channel
rx_locktodata	Input	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	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 lock-to-reference or lock-to-data mode, as follows: rx_locktodata/ rx_locktorefclk 0/0–receiver CDR is in automatic mode 0/1–receiver CDR is in LTR mode 1/x–receiver CDR is in LTD mode Asynchronous signal.	Channel

Table 1–2. HardCopy IV GX ALTGX Megafunction Ports (Part 9 of 14)

Port Name	Input/Output	Description	Scope
rx_signaldetect	Output	Signal threshold detect indicator. This feature is available only in PCI Express (PIPE) mode. A high level indicates that the signal present at the receiver input buffer is above the programmed signal detection threshold value. If the electrical idle inference block is disabled in PIPE mode, the rx_signaldetect signal is inverted and driven on the pipeelecidle port. Asynchronous signal.	Channel
rx_seriallpbken	Input	Serial loopback control port. 0–normal datapath, no serial loopback 1–serial loopback	Channel
Transmitter Physical Coding Sublayer Ports			
Transmitter Phase Compensation FIFO			
tx_datain	Input	Parallel data input from the core fabric to the transmitter. The bus width depends on the channel width multiplied by the number of channels per instance.	Channel
tx_clkout	Output	Core fabric-transceiver interface clock. Each channel has a tx_clkout signal in non-bonded channel configurations. Use this clock signal to clock the parallel data tx_datain from the core fabric into the transmitter. This signal is not available in bonded channel configurations.	Channel
tx_coreclk	Input	Optional write clock port for the transmitter phase compensation FIFO. If not selected, the Quartus II software automatically selects tx_clkout/coreclkout as the write clock for transmitter phase compensation FIFO. If selected, you must drive this port with a clock that is frequency locked to tx_clkout/coreclkout.	Channel
tx_phase_comp_fifo_error	Output	Transmitter phase compensation FIFO full or empty indicator. A high level indicates that the transmitter phase compensation FIFO is either full or empty.	Channel

Table 1–2. HardCopy IV GX ALTGX Megafunction Ports (Part 10 of 14)

Port Name	Input/Output	Description	Scope
8B/10B Encoder			
tx_ctrlenable	Input	8B/10B encoder /Kx.y/ or /Dx.y/ control. When asserted high, the 8B/10B encoder encodes the data on the tx_datain port as a /Kx.y/ control code group. When de-asserted low, it encodes the data on the tx_datain port as a /Dx.y/ data code group. The width of this signal depends on the following channel width: Channel Width tx_ctrlenable 8 1 16 2 32 4	Channel
tx_forcedisp	Input	8B/10B encoder force disparity control. When asserted high, it forces the 8B/10B encoder to encode the data on the tx_datain port with a positive or negative disparity depending on the tx_dispval signal level. When de-asserted low, the 8B/10B encoder encodes the data on the tx_datain port according to the 8B/10B running disparity rules. The width of this signal depends on the following channel width: Channel Width tx_forcedisp 8 1 16 2 32 4	Channel
tx_dispval	Input	8B/10B encoder force disparity value. A high level on the tx_dispval signal when the tx_forcedisp signal is asserted high forces the 8B/10B encoder to encode the data on the tx_datain port with a negative starting running disparity. A low level on the tx_dispval signal when the tx_forcedisp signal is asserted high forces the 8B/10B encoder to encode the data on the tx_datain port with a positive starting running disparity. The width of this signal depends on the following channel width: Channel Width tx_dispval 8 1 16 2 32 4	Channel

Table 1–2. HardCopy IV GX ALTGX Megafunction Ports (Part 11 of 14)

Port Name	Input/Output	Description	Scope
tx_invpolarity	Input	Transmitter polarity inversion control. This feature is useful for correcting situations in which the positive and negative signals of the differential serial link are accidentally swapped during board layout. When asserted high in single-width modes, the polarity of every bit of the 8-bit or 10-bit input data to the serializer gets inverted. When asserted high in double-width modes, the polarity of every bit of the 16-bit or 20-bit input data to the serializer gets inverted. Asynchronous signal.	Channel
Transmitter Physical Media Attachment			
tx_dataout	Output	Transmitter serial data output port.	Channel
fixedclk	Input	125-MHz clock for receiver detect and offset cancellation in PCI Express (PIPE) mode.	Channel
Dynamic Reconfiguration			
reconfig_clk	Input	Dynamic reconfiguration clock. This clock is also used for offset cancellation in all modes except PIPE mode. The frequency range of this clock is 2.5 MHz to 50 MHz when the transceiver channel is configured in **Transmitter only** mode. The frequency range of this clock is 37.5 MHz to 50 MHz when the transceiver channel is configured in **Receiver only** or **Receiver and Transceiver** mode.	
reconfig_togxb	Input	From the dynamic reconfiguration controller.	
reconfig_fromgxb	Output	To the dynamic reconfiguration controller.	
PCI Express (PIPE) Interface (Available only in PIPE Functional Mode)			
powerdn	Input	PIPE power state control. Functionally equivalent to the powerdown[1:0] signal defined in the PIPE specification revision 2.0. The width of this signal is two bits and is encoded as follows: ■ 2'b00: P0–Normal Operation ■ 2'b01: P0s–Low Recovery Time Latency, Low Power State ■ 2'b10: P1–Longer Recovery Time Latency, Lower Power State ■ 2'b11: P2–Lowest Power State	Channel

Table 1–2. HardCopy IV GX ALTGX Megafunction Ports (Part 12 of 14)

Port Name	Input/Output	Description	Scope
tx_forcedispcompliance	Input	Force 8B/10B encoder to encode with a negative running disparity. Functionally equivalent to the txcompliance signal defined in PCI Express (PIPE) specification revision 2.0. Must be asserted high only when transmitting the first byte of the PIPE compliance pattern to force the 8B/10B encode with a negative running disparity as required by the PIPE protocol.	Channel
tx_forceelecidle	Input	Force transmitter buffer to PIPE electrical idle signal levels. Functionally equivalent to the txelecidle signal defined in the PCI Express (PIPE) specification revision 2.0.	Channel
rateswitch	Input	PIPE rateswitch control. ■ 1'b0—Gen1 (2.5 Gbps) ■ 1'b1—Gen2 (5 Gbps)	
tx_pipemargin	Input	Transmitter differential output voltage (V_{OD}) level control. This feature is functionally equivalent to the txmargin signal defined in the PIPE specification revision 2.0. Available only in PIPE Gen2 configuration. The width of this signal is 3 bits per channel and is decoded as follows: ■ 3'b000—Normal Operating Range ■ 3'b001—Full Swing = 800 – 1200 mV Low Swing = 400 – 700m V ■ 3'b010—TBD ■ 3'b011—TBD ■ 3'b100—If last value Full Swing = 200 – 400 mV ■ 3'b101—If last value Full Swing = 200 – 400 mV ■ 3'b110—If last value Full Swing = 200 – 400 mV ■ 3'b111—If last value Full Swing = 200 – 400 mV	
tx_pipedeemph	Input	Transmitter buffer de-emphasis level control. This feature is functionally equivalent to the txdeemph signal defined in the PIPE specification revision 2.0. Available only in PCI Express (PIPE) Gen2 configuration. ■ 1'b0: -6 dB de-emphasis ■ 1'b1:-3.5 dB de-emphasis	

Table 1–2. HardCopy IV GX ALTGX Megafunction Ports (Part 13 of 14)

Port Name	Input/Output	Description	Scope
pipe8b10binvpolarity	Input	PCI Express (PIPE) polarity inversion control. Functionally equivalent to the rxpolarity signal defined in the PIPE specification revision 2.0. This feature is available only in PIPE mode. When asserted high, the polarity of every bit of the 10-bit input data to the 8B/10B decoder gets inverted.	Channel
tx_detectrxloopback	Input	Receiver detect or PCI Express (PIPE) loopback control. This feature is functionally equivalent to the txdetectrx/loopback signal defined in the PIPE specification revision 2.0. When asserted high in the P1 power state with the tx_forceelecidle signal asserted, the transmitter buffer begins the receiver detection operation. After the receiver detect completion is indicated on the pipephydonestatus port, this signal must be de-asserted. When asserted high in the P0 power state with the tx_forceelecidle signal de-asserted, the transceiver datapath gets dynamically configured to support parallel loopback as described in "PCI Express (PIPE) Reverse Parallel Loopback" on page 1–191.	Channel
pipestatus	Output	PIPE receiver status port. This feature is functionally equivalent to the rxstatus[2:0] signal defined in the PIPE specification revision 2.0. The width of this signal is 3-bits per channel. The encoding of receiver status on the pipestatus port is as follows: ■ 000–Received data OK ■ 001–1 skip added ■ 010–1 skip removed ■ 011–Receiver detected ■ 100–8B/10B decoder error ■ 101–Elastic buffer overflow ■ 110–Elastic buffer underflow ■ 111–Received disparity error	Channel
pipephydonestatus	Output	PHY function completion indicator. This feature is functionally equivalent to the phystatus signal defined in the PIPE specification revision 2.0. Assert this signal high for one parallel clock cycle to communicate completion of several PHY functions, such as power state transition, receiver detection, and signaling rate change between Gen1 (2.5 Gbps) and Gen2 (5 Gbps).	Channel

Table 1–2. HardCopy IV GX ALTGX Megafunction Ports (Part 14 of 14)

Port Name	Input/Output	Description	Scope
rx_pipedatavalid	Output	Valid data and control on the rx_dataout and rx_ctrldetect ports indicator. Functionally equivalent to the rxvalid signal defined in the PIPE specification revision 2.0.	Channel
pipeelecidle	Output	Electrical idle detected or inferred at the receiver indicator. This feature is functionally equivalent to the rxelecidle signal defined in the PCI Express (PIPE) specification revision 2.0. If the electrical idle inference block is enabled, it drives this signal high when it infers an electrical idle condition, as described in "Electrical Idle Inference" on page 1–136. Otherwise, it drives this signal low. If the electrical idle inference block is disabled, the rx_signaldetect signal from the signal detect circuitry in the receiver buffer is inverted and driven on this port. Asynchronous signal.	Channel
Reset and Power Down			
gxb_powerdown	Input	Transceiver block power down. When asserted high, all digital and analog circuitry within the PCS, PMA, CMU channels, and the CCU of the transceiver block is powered down. Asserting the gxb_powerdown signal does not power down the REFCLK buffers. Asynchronous signal. The minimum pulse width is 1 µs (pending characterization).	Transceiver block
rx_digitalreset	Input	Receiver PCS reset. When asserted high, the receiver PCS blocks are reset. The minimum pulse width is two parallel clock cycles.	Channel
rx_analogreset	Input	Receiver PMA reset. When asserted high, analog circuitry within the receiver PMA gets reset. The minimum pulse width is two parallel clock cycles.	Channel
tx_digitalreset	Input	Transmitter PCS reset. When asserted high, the transmitter PCS blocks are reset. The minimum pulse width is two parallel clock cycles.	Channel
Calibration Block			
cal_blk_clk	Input	Clock for transceiver calibration blocks.	Device
cal_blk_powerdown	Input	Calibration block power down control.	Device

CMU Channels

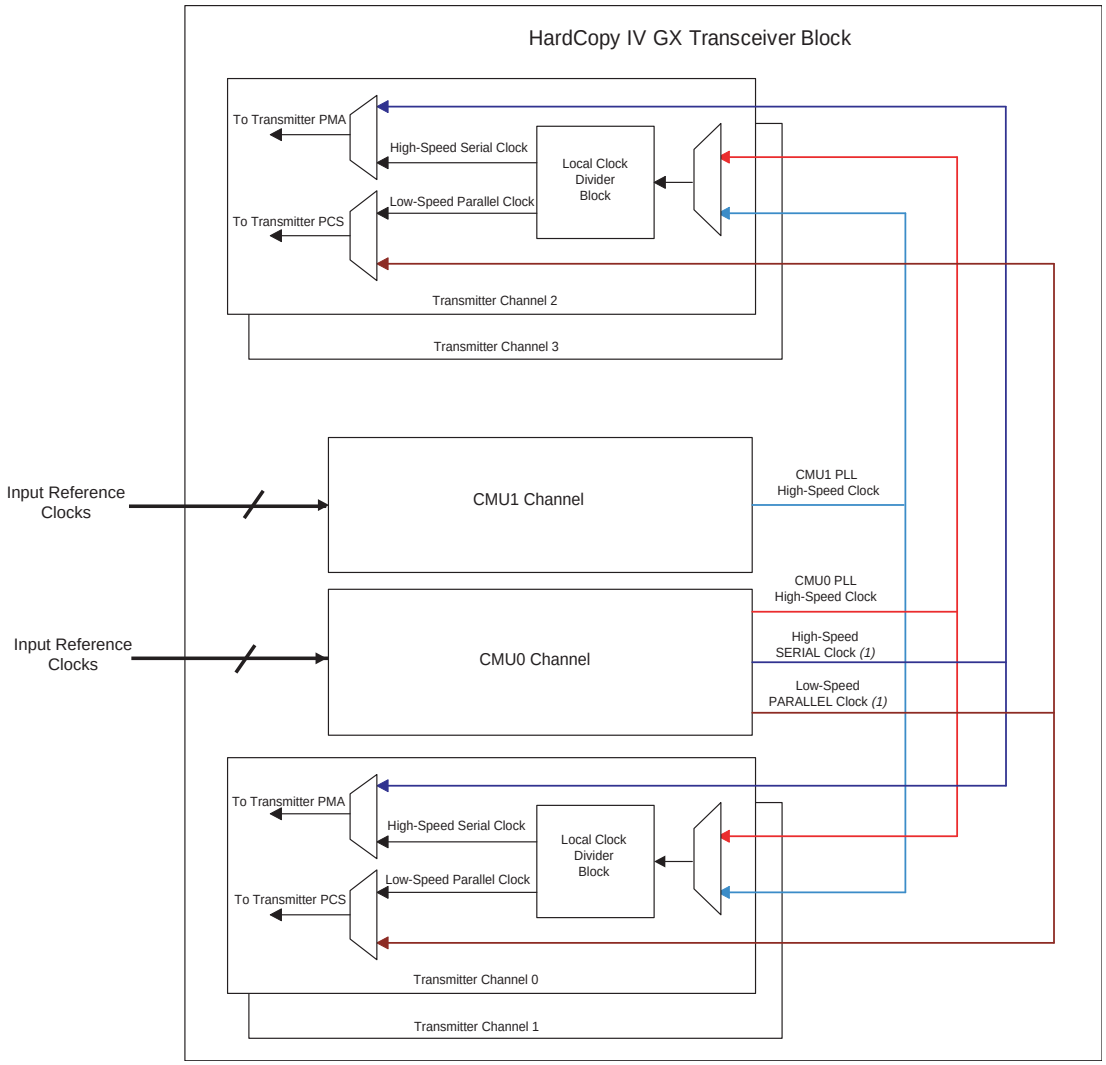
The HardCopy IV GX device contains two CMU channels—the CMU0 and CMU1 channels—within each transceiver block. The CMU channels can be configured as a transceiver channel or as a clock generation block. The building blocks used in the CMU channels to achieve this are described below. Each CMU channel contains a CMU PLL that provides clocks to the transmitter channels within the same transceiver block.

Configuring CMU Channels for Clock Generation

The CMU0 channel has additional capabilities to support bonded protocol functional modes such as Basic ×4, XAUI, and PCI Express (PIPE). You can select these functional modes from the ALTGX MegaWizard Plug-In Manager. You can enable Basic ×4 functional mode in the ALTGX MegaWizard Plug-In Manager by selecting the **×4** option in Basic mode.

Figure 1–7 shows a top-level block diagram of the CMU channels in a transceiver block.

Figure 1–7. Top-Level Diagram of CMU Channels in a Transceiver Block



Note to Figure 1–7:

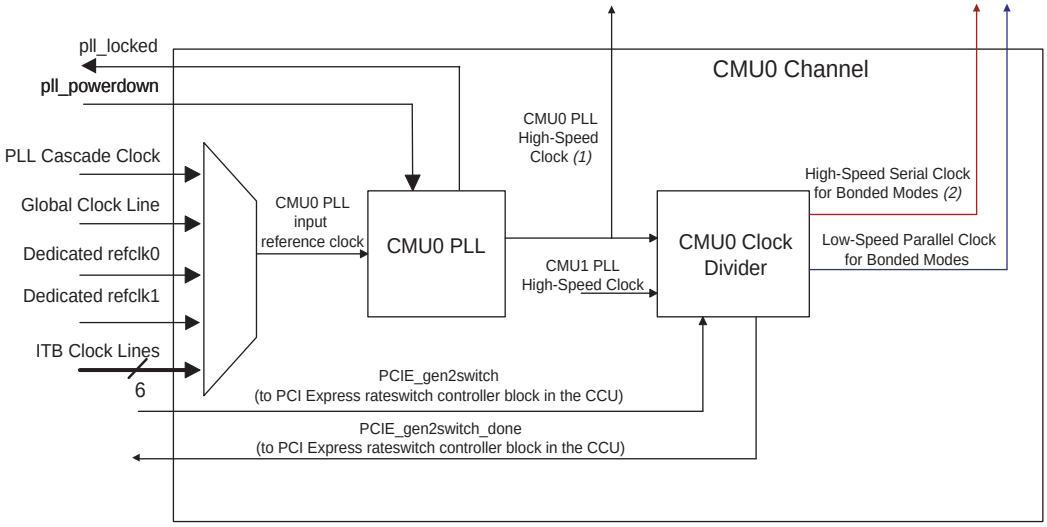
(1) Clocks are provided to support bonded channel functional mode.

CMU0 Channel

The CMU0 channel, shown in Figure 1–8, contains the following blocks:

- CMU0 PLL
- CMU0 clock divider

Figure 1–8. Block Diagram of the CMU0 Channel



Notes to Figure 1–8:

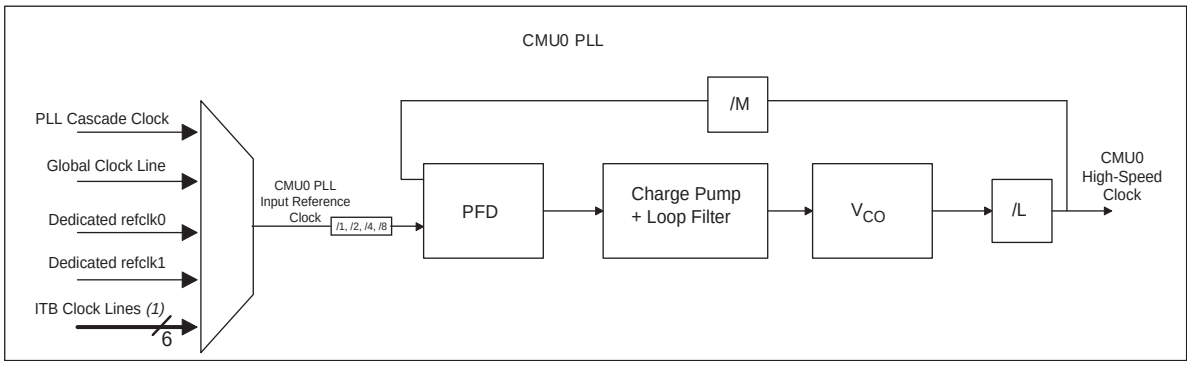
(1) In non-bonded functional modes (for example, GIGE functional mode), the transmitter channel uses the transmitter local clock divider to divide this high-speed clock output to provide clocks for its PMA and PCS blocks.

(2) Used in XAUI, Basic ×4, and PCI Express (PIPE) ×4 functional modes. In PIPE ×8 functional mode, only the CMU0 channel of the master transceiver block provides clock output to all eight transceiver channels configured in PIPE functional mode.

CMU0 PLL

Figure 1–9 shows a block diagram of the CMU0 PLL.

Figure 1–9. Block Diagram of the CMU0 PLL



Note to Figure 1–9:

(1) The inter transceiver block (ITB) clock lines shown are the maximum value. The actual number of ITB lines in your device depends on the number of transceiver blocks on one side of the device.

You can select the input reference clock to the CMU0 PLL from multiple clock sources. The various clock sources are:

- PLL cascade clock—the PLL cascade clock is the output from the general purpose PLLs in the core fabric
- Global clock line—the input reference clock from the dedicated CLK pins are connected to the global clock line
- refclk0—dedicated REFCLK in the transceiver block
- refclk1—dedicated REFCLK in the transceiver block
- Inter transceiver block (ITB) 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 phase frequency detector (PFD) tracks the voltage-controlled oscillator (VCO) output with the input reference clock.

The VCO in the CMU0 PLL is half rate and runs at half the serial data rate. The CMU0 PLL uses two multiplier blocks (/M and /L) in the feedback path (shown in Figure 1–9) to generate the high-speed clock needed to support a native data rate range of 600 Mbps to 6.5 Gbps.

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

Each CMU PLL (CMU0 PLL and CMU1 PLL) has a dedicated pll_locked signal that is asserted to indicate that the CMU PLL is locked to the input reference clock.

PLL Bandwidth Setting

You can program the PLL bandwidth setting using the ALTGX MegaWizard Plug-In Manager. 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.

- 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 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

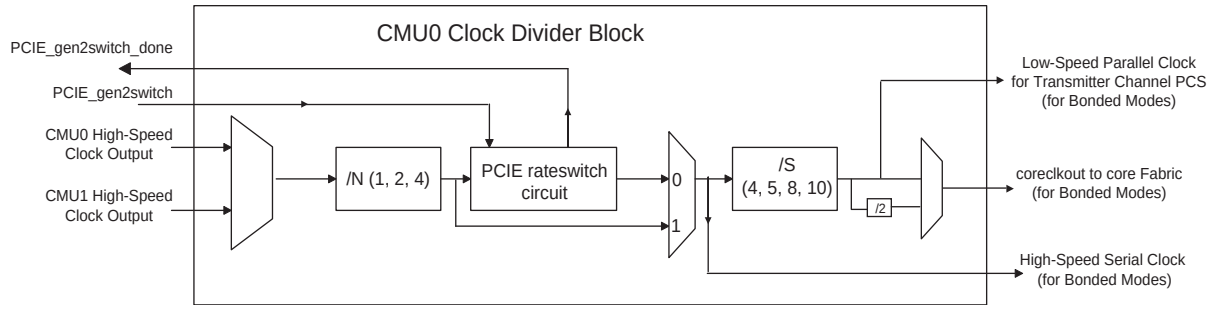
You can power down the CMU0 PLL by asserting the pll_powerdown signal.

CMU0 Clock Divider Block

The high-speed clock output from the CMU0 PLL is forwarded to two clock divider blocks: the CMU0 clock divider block and the transmitter channel local clock divider block. This clock divider block is used only in bonded channel functional modes. In all non-bonded functional modes (such as GIGE functional mode), the local clock divider block divides the high-speed clock to provide clocks for its PCS and PMA blocks. This section only describes the CMU0 clock divider block.

You can configure the CMU0 clock divider block, shown in Figure 1–10, to select the high-speed clock output from the CMU0 PLL or CMU1 PLL. The CMU1 PLL is present in the CMU1 channel.

Figure 1–10. CMU0 Clock Divider Block



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. This high-speed serial clock is used for bonded functional modes such as Basic ×4, XAUI, and PCI Express (PIPE) ×4 configurations. In XAUI and Basic ×4 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 PIPE ×4 mode, the clock path through the PIPE rateswitch circuit block is selected. This high-speed serial clock is provided to all the transmitter channels.

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

In PIPE ×1 mode, the CMU0 clock divider does not provide a high-speed serial clock. Instead, the local clock divider block in the transmitter channel receives the CMU0 PLL 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 PIPE ×4 mode. In PIPE ×8 mode, the PCIE rateswitch circuit of the CMU0 clock divider of the master transceiver block is active. There are two paths in the PCIE rateswitch circuit. One path divides the /N output by two. The other path forwards the /N divider output.

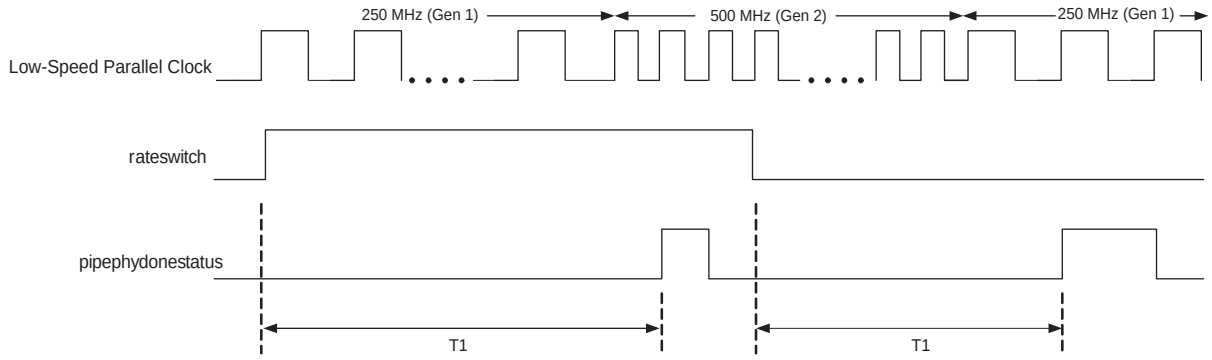
When you set the `rateswitch` port to **0**, the PCI Express (PIPE) rateswitch controller (in the CCU) signals the PCIE rateswitch circuit to select the divide by /2 to provide a high-speed serial clock for the Gen1 (2.5 Gbps) data rate. When the `rateswitch` port is set to **1**, the /N divider output 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 PIPE rateswitch controller in the CCU. The PIPE rateswitch controller asserts the `pipephydonestatus` signal for one clock cycle after the rateswitch operation is completed for both the transmit and receive channels. Figure 1–11 shows the timing diagram for the rateswitch operation.

For more information about PCI Express (PIPE) functional mode rateswitch, refer to "PCI Express (PIPE) Gen2 (5 Gbps) Support" on page 1–137.

Figure 1–11. Rateswitch in PCI Express (PIPE) Mode *(Note 1)*



Note to Figure 1–11:

(1) Time T1 is pending characterization.

When you create a PIPE Gen2 configuration, configure the CMU PLL to 5 Gbps. This helps to generate the 2.5 Gbps and 5 Gbps high-speed serial clock using the rateswitch circuit.

Low-Speed Parallel Clock Generation

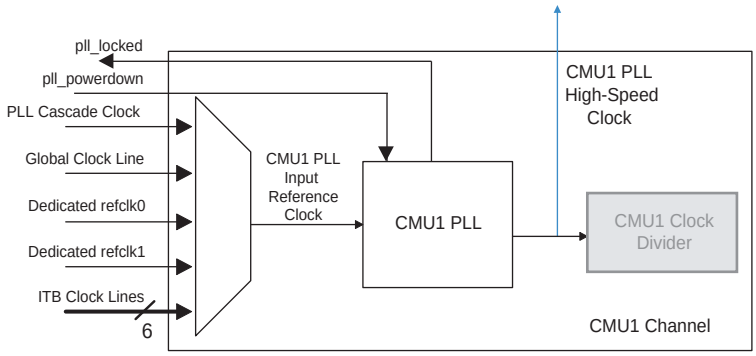
The /S divider receives the clock output from the /N divider or PCIE rateswitch circuit (only in PIPE mode) and generates the low-speed parallel clock for the PCS block of all transmitter channels and `coreclkout` for the core fabric. If the byte serializer block is enabled in bonded channel modes, the /S divider output is divided by the /2 divider and sent out as `coreclkout` to the core fabric. The Quartus II software automatically selects the /S values based on the deserialization width setting (single-width mode or double-width mode) that you select in the ALTGX MegaWizard Plug-In Manager. For more information about single-width or double-width mode, refer to "Transceiver Channel Architecture" on page 1–34.

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 1–12 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. The functionality of the CMU0 PLL is described in "CMU0 PLL" on page 1–24.

Figure 1–12. 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. 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.

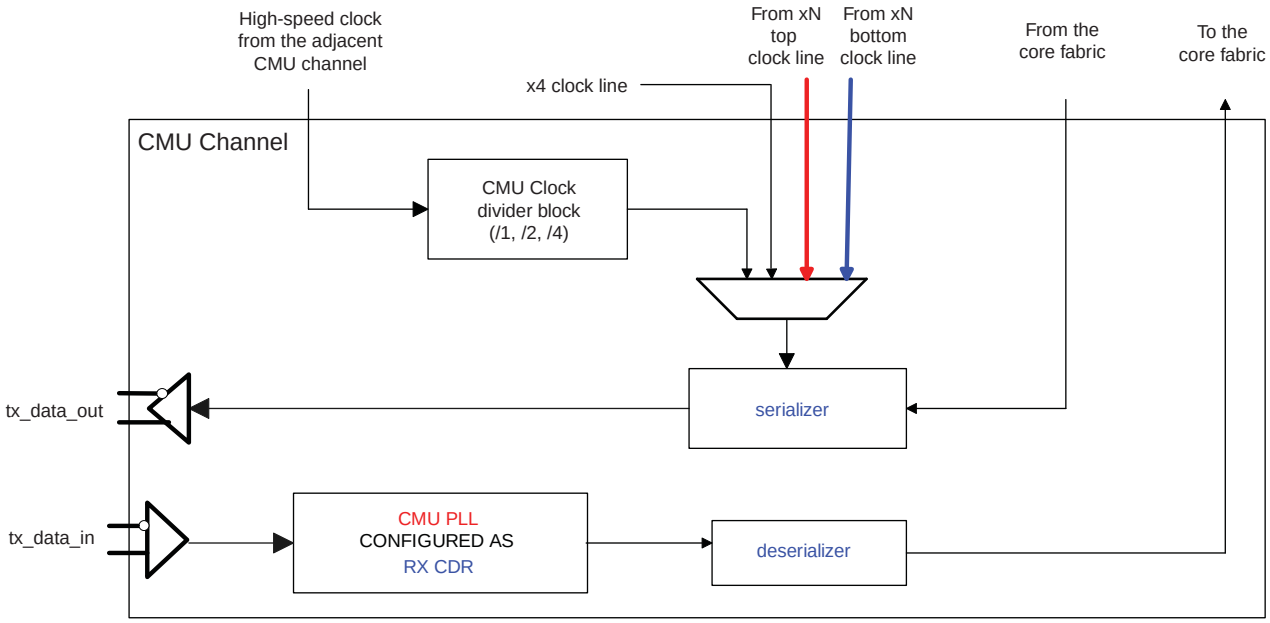
Power Down CMU1 PLL

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

Configuring CMU Channels as Transceiver Channels

The two CMU channels in the transceiver block can be configured as a transceiver channel to run between 600 Mbps and 6.5 Gbps. Figure 1–13 shows the functional blocks that are enabled to support the transceiver channel functionality.

Figure 1–13. Functional Blocks Enabled to Support Transceiver Channel Functionality



The CMU PLL is configured as a CDR to recover data. The dedicated input reference clock pin is configured to receive serial data.

Table 1–3 shows the pins that are used as transmit and receive serial pins.

Table 1–3. Pins Used as Transmit and Receive Serial Pins (Part 1 of 2)

Pins *(1)*	When a CMU Channel is Configured as a Transceiver Channel	When a CMU Channel is Configured for Clock Generation
`REFCLK_[L,R][0,2,4,6]P`, `GXB_CMURX_[L_R][0,2,4,6]P` *(2)*	Receive serial input for CMU `Channel0`	Input reference clocks
`GXB_TX_[L,R][0,2,4,6]` *(2)*	Transmit serial output for CMU `Channel0`	Not available for use
`REFCLK_[L,R][1,3,5,7]P`, `GXB_CMURX_[L_R][1,3,5,7]P` *(3)*	Receive serial input for CMU `Channel1`	Input reference clocks

Table 1–3. Pins Used as Transmit and Receive Serial Pins (Part 2 of 2)

Pins *(1)*	When a CMU Channel is Configured as a Transceiver Channel	When a CMU Channel is Configured for Clock Generation
GXB_TX_[L,R][1,3,5,7]P*(3)*	Transmit serial output for CMU Channel1	Not available for use

Notes to Table 1–3:

(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 in volume 1 of the *HardCopy IV Device Handbook*.

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

(3) Pins 1,3,5,7 are hardwired to CMU channel1 in the corresponding transceiver blocks.

Interpret the pin column as follows:

For pins REFCLK_[L,R][0,2,4,6]P, 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. Refer to the notes to Table 1–3.

Serializer and Deserializer

The serializer and deserializer convert the parallel-to-serial and serial-to-parallel data on the transmitter and receiver side, respectively. The ALTGX MegaWizard Plug-In Manager provides a new functional mode “Basic (PMA-Direct) 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.

CMU Clock Divider Block

When you configure a CMU channel in Basic (PMA-Direct)×1 mode, this block 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 block can divide the high-speed clock by /1, /2, and /4.

Clocks for the Transmitter Serializer

When the CMU channel is configured as a transceiver channel, the clocks for the transmitter side can be provided by one of these sources:

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

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

Input Reference Clocks for the Receiver CDR

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

1. From adjacent REFCLKs within the same transceiver block, if the adjacent CMU Channel is not used as a transceiver channel
2. From the REFCLK of adjacent transceiver blocks on the same side of the device, if the corresponding CMU channels are not used as transceiver channels. For REFCLK connections to the CMU channel from the global clock lines and PLL cascade network, refer to Table 1–4 on page 1–36.

Clocks for the Receiver Deserializer

The CDR provides the high-speed serial and low-speed parallel clock to the receiver deserializer high-speed clock. The receiver deserializer is provided by the high-speed serial and parallel clock of the CDR from the recovered data. There are multiple sources to provide input reference clocks to the CDR of the CMU channel.

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.
- On-chip termination (OCT)—CMU channels can have an on-chip termination feature. The allowed termination values are the same as regular channels (85, 100, 120, 150 Ω).
- Loopback—The available loop back options are serial, reverse serial (pre-CDR), and reverse serial (CDR) loopback.

For more information about analog controls and on-chip termination, refer to "Transmitter Output Buffer" on page 1–52, and "Receiver Input Buffer" on page 1–59.

For more information about loopback, refer to "Loopback Modes" on page 1–188.

Dynamic Reconfiguration

You can dynamically reconfigure the analog controls of the CMU channel using the dynamic reconfiguration controller.

For more information about dynamic reconfiguration options, refer to the *HardCopy IV GX Dynamic Reconfiguration* chapter in volume 3 of the *HardCopy IV Device Handbook*.

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 1–14, Figure 1–15, and Figure 1–16 show the location of the ATX PLL blocks in two, three, and four transceiver block device families.

Figure 1–14. Location of ATX PLL Block in a Two-Transceiver Block Device

Figure 1–15. Location of ATX PLL Blocks in a Four-Transceiver Block Device (Two on Each Side)

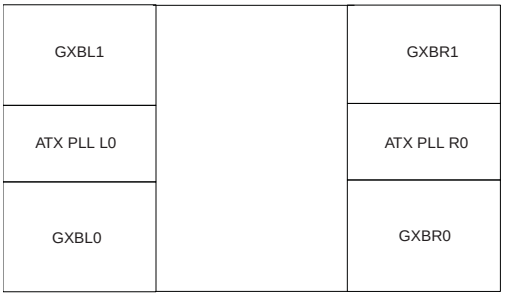
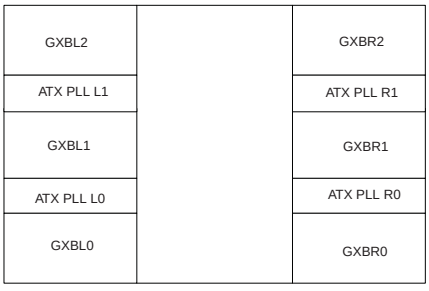


Figure 1–16. Location of ATX PLL Blocks in a Six-Transceiver Block Device



Input Reference Clocks for the ATX PLL

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

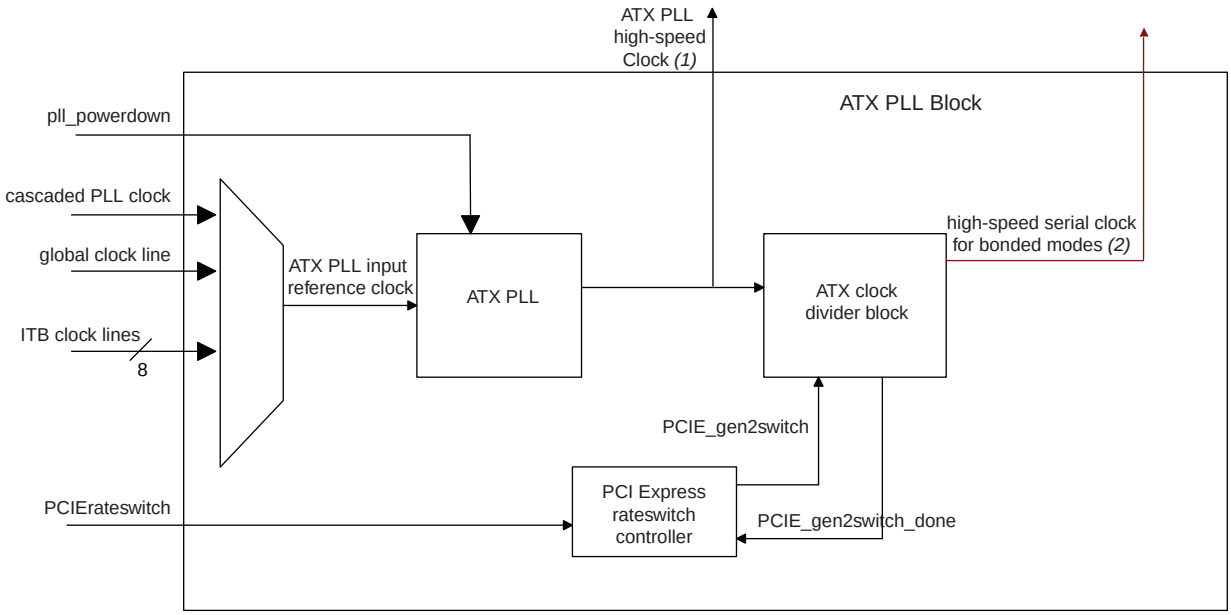
- REFCLKS from the transceiver blocks on the same side of the device, if the corresponding CMU channels are not used as transceiver channels
- Input reference clock provided through the PLL cascade clock network
- Clock inputs connected through the global clock lines

Altera recommends that you use the 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 1–17.

Figure 1–17. ATX PLL Block



Notes to Figure 1–17:

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

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

The functional blocks on the ATX PLL are similar to the blocks explained in "CMU0 PLL" on page 1–24. 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.

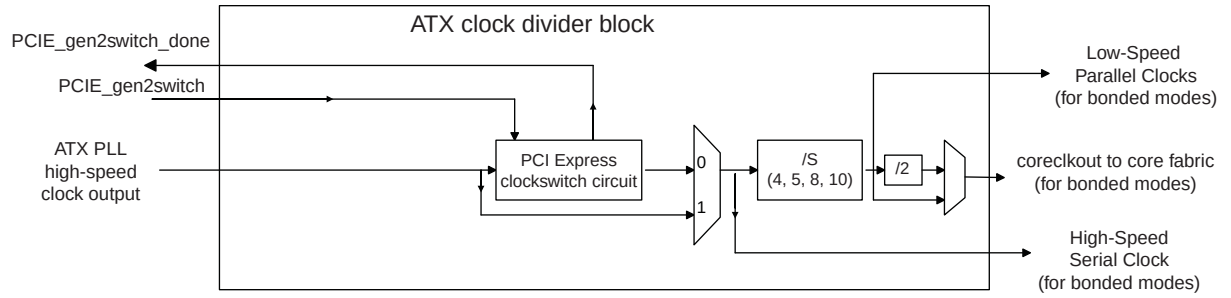
The ATX PLL high-speed clock output provides high-speed serial clocks for non-bonded functional modes such as CEI (with the subprotocol "none").

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 PCI Express (PIPE) (×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 1–24.

The clock outputs from the ATX PLL block are provided to the transmitter channels through the ×N_Top or ×N_bottom clock lines, as shown in Figure 1–18.

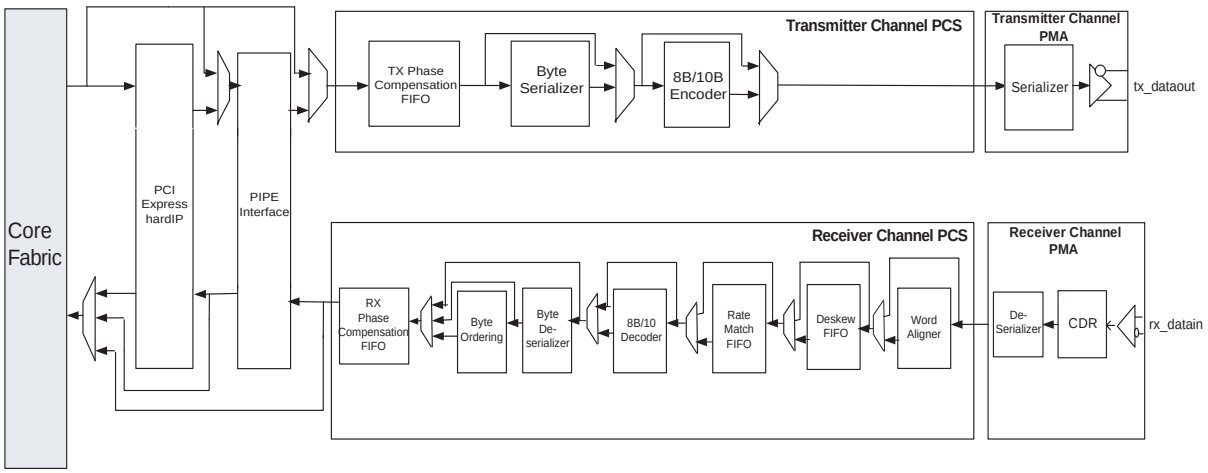
Figure 1–18. ATX Clock Divider



Transceiver Channel Architecture

Figure 1–19 shows the HardCopy IV GX transceiver channel datapath.

Figure 1–19. HardCopy IV Transceiver Datapath



Each transceiver channel consists of the following:

- Transmitter channel, further divided into
  - Transmitter channel PCS
  - Transmitter channel PMA

- Receiver channel, further divided into
  - Receiver channel PCS
  - Receiver channel PMA

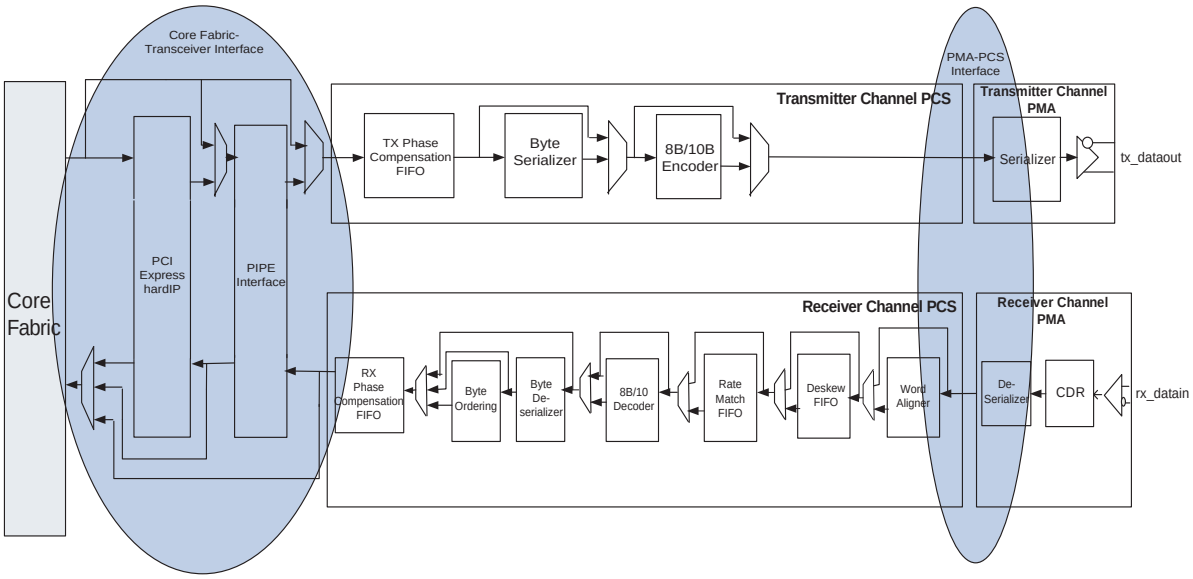
Each transceiver channel interfaces to either the PCI Express (PIPE) hard IP block (PIPE hard IP-transceiver interface) or directly to the core fabric (core fabric—transceiver interface). The transceiver channel interfaces to the PIPE hard IP block if the hard IP block is used to implement the PCI Express PHY MAC, data link layer, and transaction layer. Otherwise, the transceiver channel interfaces directly to the core fabric.

The PIPE hard IP—transceiver interface is out of the scope of this chapter. This chapter describes the core fabric-transceiver interface.

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

Figure 1–20 shows the core fabric-transceiver interface and transceiver PMA-PCS interface.

Figure 1–20. Core Fabric-Transceiver Interface and Transceiver PMA-PCS Interface



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

- Single-width mode
- Double-width mode

Table 1–4 shows the core fabric-transceiver interface widths (channel width) and transceiver PMA-PCS widths (serialization factor) allowed in single-width and double-width modes.

Table 1–4. Core Fabric-Transceiver Interface Width and Transceiver PMA-PCS Widths

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

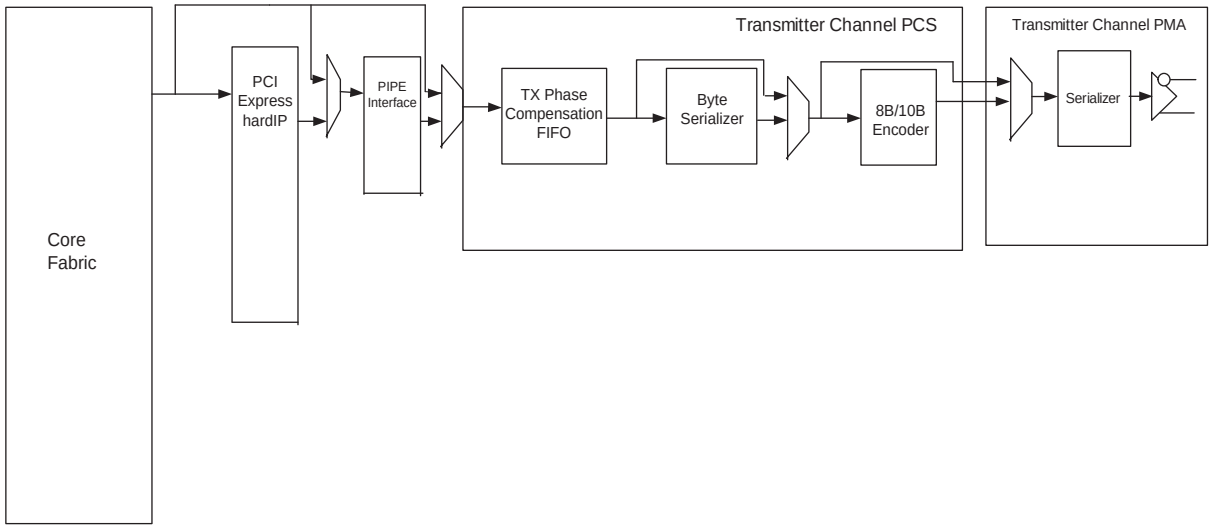
Transmitter Channel Datapath

The transmitter channel datapath, shown in Figure 1–21, 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.

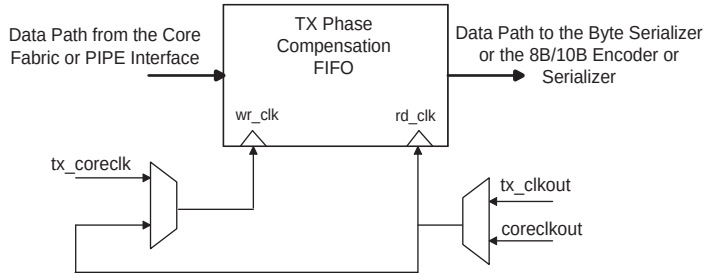
Figure 1–21. Transmitter Channel Datapath



TX Phase Compensation FIFO

The TX phase compensation FIFO interfaces the transmitter channel PCS and the core fabric PCI Express (PIPE) interface. It compensates for the phase difference between the low-speed parallel clock and the core fabric interface clock. The TX phase compensation FIFO operates in low-latency and high-latency mode. Figure 1–22 shows the datapath and clocking of the TX phase compensation FIFO.

Figure 1–22. TX Phase Compensation FIFO



TX phase compensation FIFO:

- In low-latency mode, the FIFO is four words deep. The latency through the FIFO is two to three core fabric parallel clock cycles (pending characterization). Low-latency mode is the default setting for every mode.
- In high-latency mode, the FIFO is eight words deep. The latency through the FIFO is approximately four to five parallel cycles (pending characterization).

In non-bonded functional modes such as GIGE, the read port of the phase compensation FIFO is clocked by the low-speed parallel clock. The write clock is fed by the `tx_clkout` port of the associated channel.

In bonded functional modes such as XAUI, the write clock of the FIFO is clocked by coreclkout provided by the CMU0 clock divider block. You can clock the write side using tx_coreclk provided from the core fabric by enabling the tx_coreclk port in the ALTGX MegaWizard Plug-In Manager. If you use this port, ensure that there is 0 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 PCI Express (PIPE) functional mode, the input data comes from the PIPE interface. In all other functional modes, the input data comes directly from the core fabric.

Output Data Destination Block

The output from the TX phase compensation FIFO is used by the byte serializer block, 8B/10B encoder, or serializer block. Table 1–5 lists the conditions under which the TX phase compensation FIFO outputs are provided to these blocks.

Table 1–5. Output Data Destination Block for TX Phase Compensation FIFO Output Data

Byte Serializer	8B/10B Encoder	Serializer
If you select: single-width mode and channel width = 16 or 20	If you select: single-width mode and channel width = 8 and 8B/10B encoder enabled	If you select: low-latency PCS bypass mode enabled or single-width mode and channel width = 8 or 10
If you select: double-width mode and channel width = 32 or 40	If you select: double-width mode and channel width = 16 and 8B/10B encoder enabled	If you select: low-latency PCS bypass mode enabled or double-width mode and channel width = 16 or 20

TX Phase Compensation FIFO Status Signal

An optional tx_phase_comp_fifo_error port is available in all functional modes to indicate a receiver phase compensation FIFO overflow or under-run condition. The tx_phase_comp_fifo_error signal is asserted high when the TX phase compensation FIFO either overflows or under-runs due to any frequency PPM difference between the FIFO read and write clocks. If the tx_phase_comp_fifo_error flag is asserted, verify the core fabric-transceiver interface clocking to ensure that there is 0 PPM difference between the TX phase compensation FIFO read and write clocks.

Byte Serializer

The byte serializer divides the input datapath by two. This allows the transceiver channel to run at higher data rates while keeping the core fabric interface frequency within the maximum limit of 250 MHz. In single-width mode, it converts the two byte wide datapath to a one byte wide datapath. In double-width mode, it converts the four-byte wide datapath to a two byte wide datapath.

For example, if you want to run the transceiver channel at 6.25 Gbps, without the byte serializer, in double-width mode, the core fabric interface clock frequency must be 312.5 MHz (6.25/20). This violates the core fabric interface frequency limit. When you use the byte serializer, the core fabric interface frequency is 156.25 MHz (6.25G/40). You can enable the byte serializer in single-width or double-width mode.

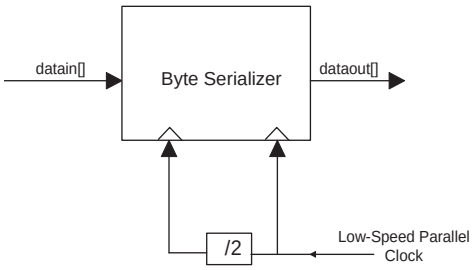
The byte deserializer is required in configurations that exceed the core fabric-transceiver interface maximum frequency limit. It is optional in configurations that do not exceed the core fabric-transceiver interface maximum frequency limit.

For more information about the maximum frequency limit for the transceiver interface, refer to the *HardCopy IV Device Datasheet* in volume 4 of the *HardCopy IV Device Handbook*.

Single-Width Mode

Figure 1–23 shows the byte serializer datapath in single-width mode.

Figure 1–23. Byte Serializer Datapath in Single-Width Mode *(Note 1)*, *(2)*



Notes to Figure 1–23:

(1) Refer to Table 1–6 on page 1–39 for the `datain[]` and `dataout[]` port widths.

(2) The `datain` signal is the input from the core fabric that has already passed through the TX phase compensation FIFO.

The byte serializer forwards the least significant byte first, followed by the most significant byte. The input data width to the byte serializer depends on the channel width option that you selected in the ALTGX MegaWizard Plug-In Manager. For example, in single-width mode, assuming a channel width of 20, the byte serializer sends out the least significant word `datain[9:0]` of the parallel data from the core fabric, followed by `datain[19:10]`. Table 1–6 shows the input and output data widths of the byte serializer in single-width mode.

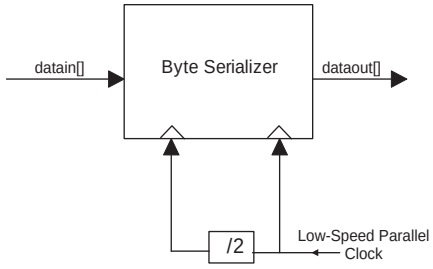
Table 1–6. Input and Output Data Width of the Byte Serializer in Single-Width Mode

Deserialization Width	Input Data Width to the Byte Serializer	Output Data Width from the Byte Serializer
Single-width mode	16	8
	20	10

Double-Width Mode

Figure 1–24 shows the byte serializer datapath in double-width mode.

Figure 1–24. Byte Serializer Datapath in Double-Width Mode *(Note 1)*, *(2)*



Notes to Figure 1–24:

(1) Refer to Table 1–7 for the `datain[]` and `dataout[]` port width.

(2) The `datain` signal is the input from the core fabric that has already passed through the TX phase compensation FIFO.

The operation in double-width mode is similar to that of single-width mode. For example, assuming a channel width of 40, the byte serializer forwards `datain[19:0]` first, followed by `datain[39:20]`. Table 1–7 shows the input and output data widths of the byte serializer in double-width mode.

Table 1–7. Input and Output Data Width of the Byte Serializer in Double-Width Mode

Deserialization Width	Input Data Width to the Byte Serializer	Output Data Width from the Byte Serializer
Double-width mode	32	16
	40	20

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

If you select the **8B/10B Encoder** option in the ALTGX MegaWizard Plug-In Manager, the 8B/10B encoder uses the output from the byte serializer. Otherwise, the byte serializer output is forwarded to the serializer.

8B/10B Encoder

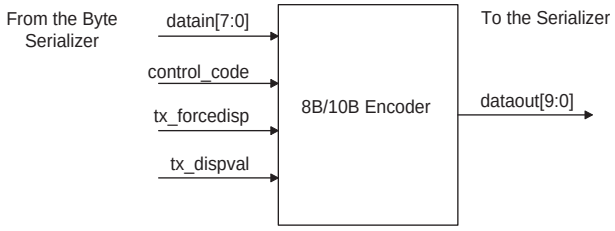
The 8B/10B encoder generates 10-bit code groups from the 8-bit data and 1-bit control identifier. The 8B/10B encoder operates in two modes: single-width and double-width.

- In single-width mode, the 8B/10B encoder generates a 10-bit code group from the 8-bit data and 1-bit control identifier.
- In double-width mode, there are two 8B/10B encoders that are cascaded together to generate two 10-bit code groups from two 8-bit data and their respective control identifiers.

Single-Width Mode

Figure 1–25 shows the 8B/10B encoder in single-width mode.

Figure 1–25. 8B/10B Encoder in Single-Width Mode

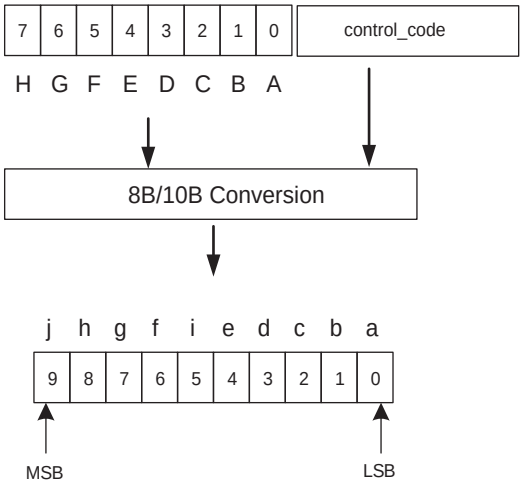


In single-width mode, the 8B/10B encoder translates the 8-bit data to a 10-bit code group (control word or data word) with proper disparity. If the `control_code` input is high, the 8B/10B encoder translates the input `data[7:0]` to a 10-bit control word. If the `control_code` input is low, the 8B/10B encoder translates the input `data[7:0]` to a 10-bit data word.

You can use the `tx_forcedisp` and `tx_dispval` ports to control the running disparity of the generated output data. For more information, refer to "Controlling Running Disparity" on page 1–45.

Figure 1–26 shows the conversion format. The LSB is transmitted first.

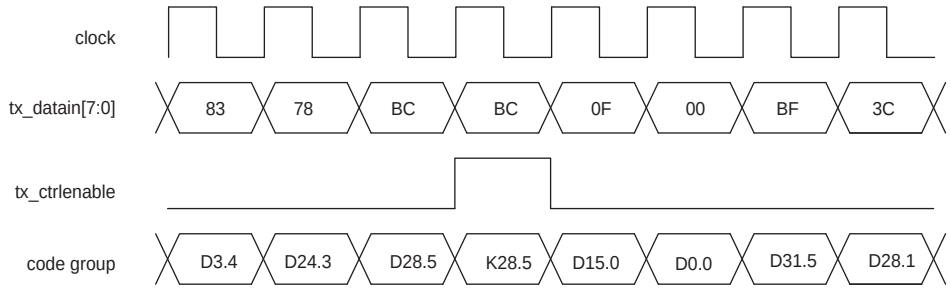
Figure 1–26. 8B/10B Conversion Format



Control Code Encoding

The ALTGX MegaWizard Plug-In Manager provides the `tx_ctrlenable` port to indicate whether the 8-bit data at the `tx_datain` port should be encoded as a control word (Kx.y). When `tx_ctrlenable` is low, the 8B/10B encoder block encodes the byte at the `tx_datain` port (the user-input port) as data (Dx.y). When `tx_ctrlenable` is high, the 8B/10B encoder encodes the input data as a Kx.y code group. The waveform in Figure 1–27 shows the second 0 × BC encoded as a control word (K28.5). The rest of the `tx_datain` bytes are encoded as a data word (Dx.y).

Figure 1–27. Control Word and Data Word Transmission



The IEEE 802.3 8B/10B encoder specification identifies only a set of 8-bit characters for which `tx_ctrlenable` should be asserted. If you assert `tx_ctrlenable` for any other set of bytes, the 8B/10B encoder might encode the output 10-bit code as an invalid code (it does not map to a valid Dx.y or Kx.y code), or unintended valid Dx.y code, depending on the value entered. It is possible for a downstream 8B/10B decoder to decode an invalid control word into a valid Dx.y code without asserting code error flags.

For example, depending on the current running disparity, the invalid code K24.1 (`tx_datain` = 8'h38 + `tx_ctrl` = 1'b1) can be encoded to 10'b0110001100 (0 × 18C), which is equivalent to a D24.6+ (8'hD8 from the RD+ column). Altera recommends that you do not assert `tx_ctrlenable` for unsupported 8-bit characters.

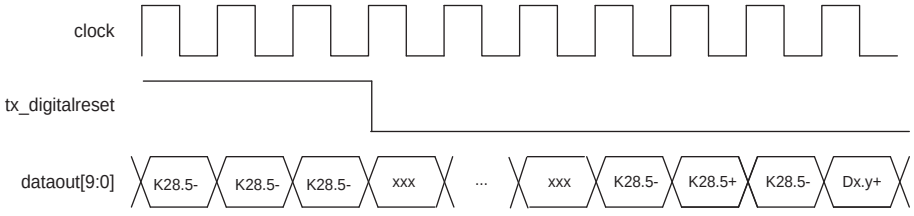
Reset Condition

The `tx_digitalreset` signal resets the 8B/10B encoder. During reset, running disparity and data registers are cleared. Also, the 8B/10B encoder outputs a K28.5 pattern from the RD- column continuously until `tx_digitalreset` is de-asserted. The input data and control code from the core fabric is ignored during the reset state. Once out of 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 1–28 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.

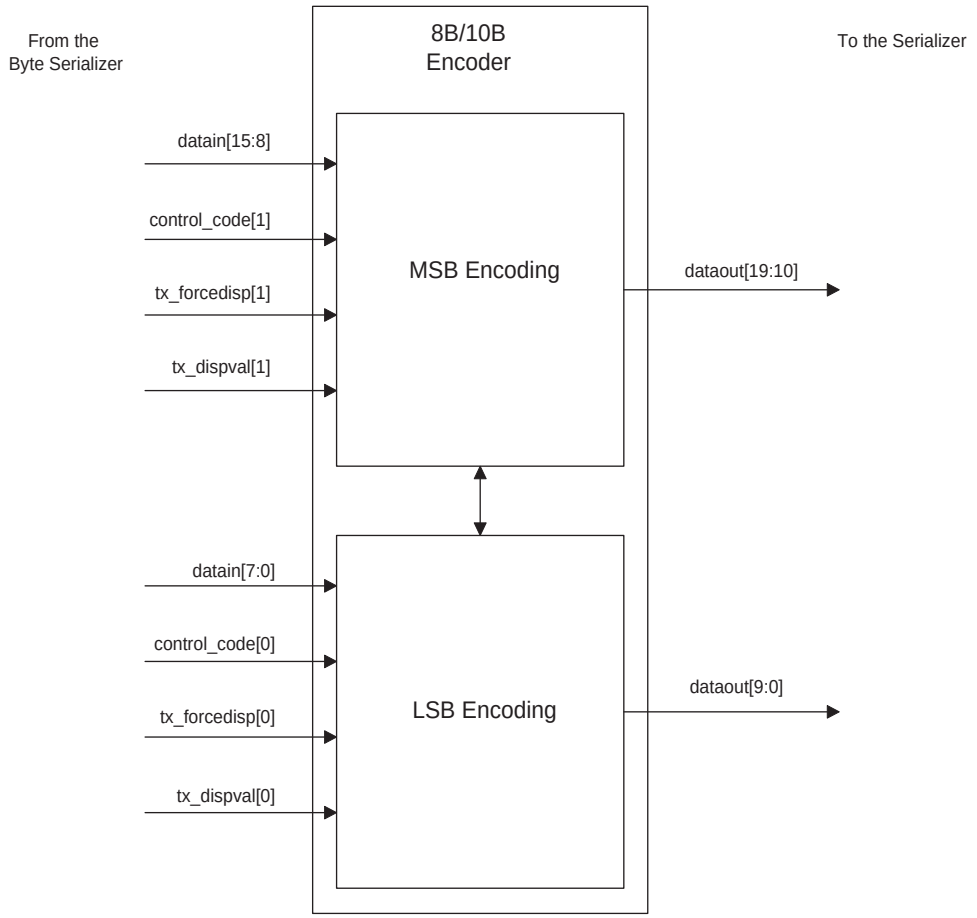
Figure 1–28. 8B/10B Encoder Output During tx_digitalreset Assertion



Double-Width Mode

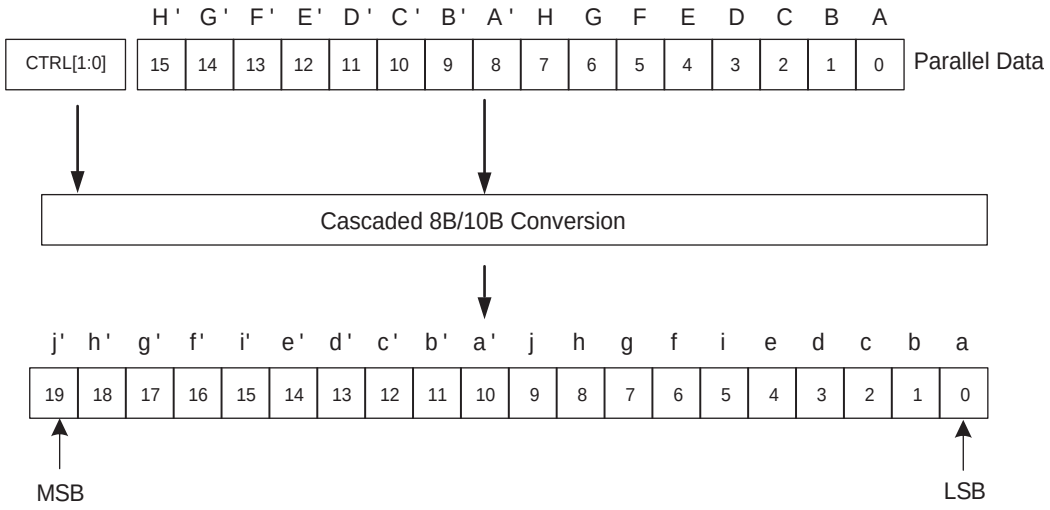
In double-width mode, the 8B/10B encoder operates in a cascaded mode, as shown in Figure 1–29. The LSByte of the input data is encoded and transmitted prior to the MSByte.

Figure 1–29. 8B/10B Encoder in Double-Width Mode



In double-width mode, the cascaded 8B/10B encoder generates two 10-bit code groups from two 8-bit data and their respective control code identifiers. Figure 1–30 shows the conversion format. The LSB shown in Figure 1–30 is transmitted first.

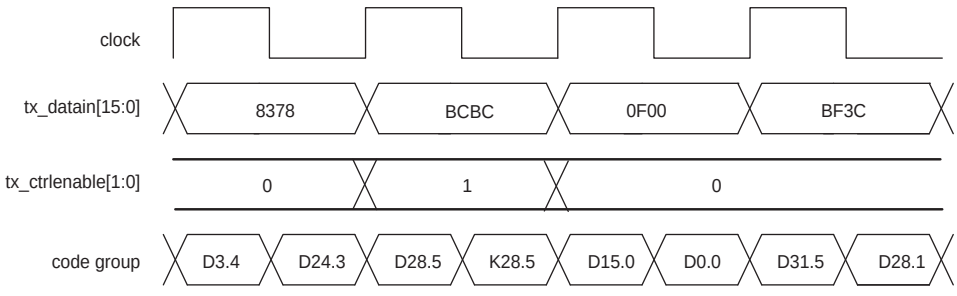
Figure 1–30. 8B/10B Conversion Format in Double-Width Mode



Control Code Encoding

In double-width mode, the `tx_ctrlenable[1:0]` port is used to identify which 8-bit data is to be encoded as a control word. The lower bit, `tx_ctrlenable[0]`, is associated with the LSByte; the upper bit, `tx_ctrlenable[1]`, is associated with the MSByte. When `tx_ctrlenable` is low, the byte at the `tx_datain` port of the transceiver is encoded as data (Dx.y); otherwise, it is encoded as a control code (Kx.y). Figure 1–31 shows that only the lower byte of the `tx_datain[15:0]` port is encoded as a control code because `tx_ctrlenable[0]` is high in the second clock cycle.

Figure 1–31. Encoded Control Word and Data Word Transmission



The 8B/10B encoder does not check to see if the code word entered is one of the 12 valid control code groups specified in the IEEE 802.3 8B/10B encoder specification. If an invalid control code is entered, the resulting 10-bit code may be encoded as an invalid code (it does not map to a valid Dx.y or Kx.y code), or unintended valid Dx.y code, depending on the value entered.

The following is an example of an invalid control word encoded into a valid Dx.y code. With an encoding of an invalid code K24.1 (`tx_datain` = 8'h38 + `tx_ctrl` = 1'b1), depending on the current running disparity, the K24.1 can be encoded to be 10'b0110001100 (0 × 18C), which is equivalent to a D24.6+ (8'hD8 from the RD+ column). An 8B/10B decoder can decode this and not assert a code error flag.

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

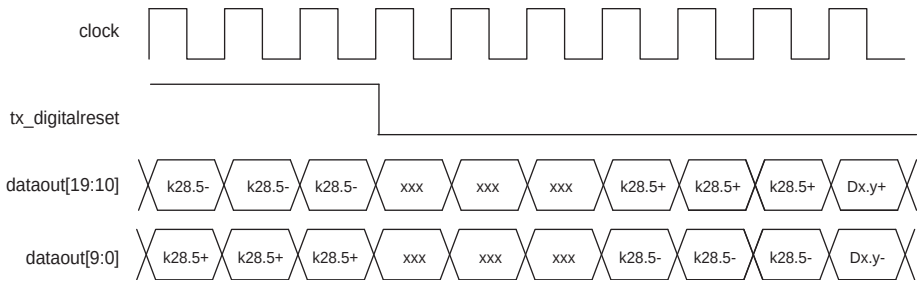
Reset Condition

The `tx_digitalreset` signal resets the 8B/10B encoder. During reset, the running disparity and data registers are cleared. Also, the 8B/10B encoder outputs a K28.5 pattern with proper disparity continuously until `tx_digitalreset` goes low. The inputs from the `tx_datain` and `tx_ctrlenable` ports are ignored during the reset state. After reset, the 8B/10B encoder starts the LSByte with a negative disparity (RD-) and the MSByte with a positive disparity (RD+) and transmits six K28.5 code groups (three on the LSByte and three on the MSByte encoder) for synchronizing before it starts encoding and transmitting data.

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

Figure 1–32 shows the reset behavior of the 8B/10B encoder. When in reset (`tx_digitalreset` is high), a K28.5- 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 1–32 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 1–32. Transmitted Output Data when tx_digitalreset is Asserted



Controlling Running Disparity

After power on or reset, the 8B/10B encoder has a negative disparity and chooses the 10-bit code from the RD- column (refer to the 8B/10B encoder specification for the RD+ and RD- column values). The ALTGX MegaWizard Plug-In Manager provides the `tx_forcedisp` and `tx_dispval` ports to control the running disparity of the output from the 8B/10B encoder. These ports are available only in Basic single-width and Basic double-width modes.

A high value on the tx_forcedisp port is the control signal to the disparity value of the output data. The disparity value (RD+ or RD–) is indicated by the value on the tx_dispval port. If the tx_forcedisp port is low, tx_dispval is ignored and the current running disparity is not altered. Forcing disparity can either maintain the current running disparity calculations if the forced disparity value (on the tx_dispval bit) happens to match 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 1–8 shows the tx_forcedisp and tx_dispval port values.

Table 1–8. 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 1–33 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 also high, the K28.5 at time n + 4 is encoded as a positive disparity code group.

Figure 1–33. 8B/10B Encoder Force Running Disparity Operation in Single-Width Mode

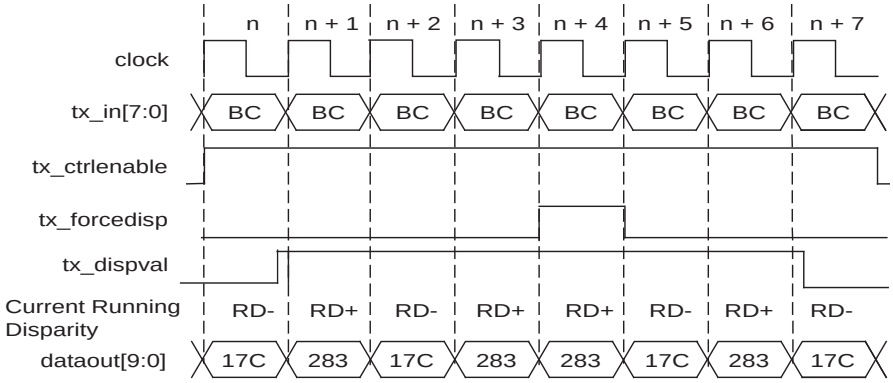
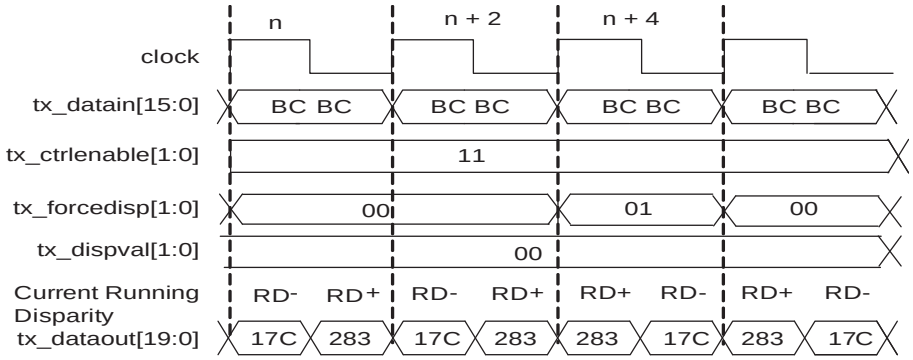


Figure 1–34 shows the current running disparity being altered in Basic double-width mode by forcing a positive disparity on a negative disparity K28.5. In this example, a series of K28.5 are continuously being sent. The stream alternates between a positive ending running disparity (RD+) K28.5 and a negative 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 1–34. 8B/10B Encoder Force Current Running Disparity in Double-Width Mode



Transmitter Polarity Inversion

The positive and negative signals of a serial differential link might accidentally be swapped during board layout. Solutions like a board re-spin or major updates to the logic in the core fabric can be expensive. The transmitter polarity inversion feature is provided to correct this situation. An optional `tx_invpolarity` port is available in all functional modes except (OIF) CEI PHY to dynamically enable the transmitter polarity inversion feature.

In single-width mode, a high value on the `tx_invpolarity` port inverts the polarity of every bit of the 8-bit or 10-bit input data word to the serializer in the transmitter datapath. In double-width mode, a high value on the `tx_invpolarity` port inverts the polarity of every bit of the 16-bit or 20-bit input data word to the serializer in the transmitter datapath. Because inverting the polarity of each bit has the same effect as swapping the positive and negative signals of the differential link, correct data is seen by the receiver. `tx_invpolarity` is a dynamic signal and might cause initial disparity errors at the receiver of an 8B/10B encoded link. The downstream system must be able to tolerate these disparity errors.

Figure 1–35 shows the transmitter polarity inversion feature in a single-width 10-bit wide datapath configuration.

Figure 1–35. Transmitter Polarity Inversion in Single-Width Mode

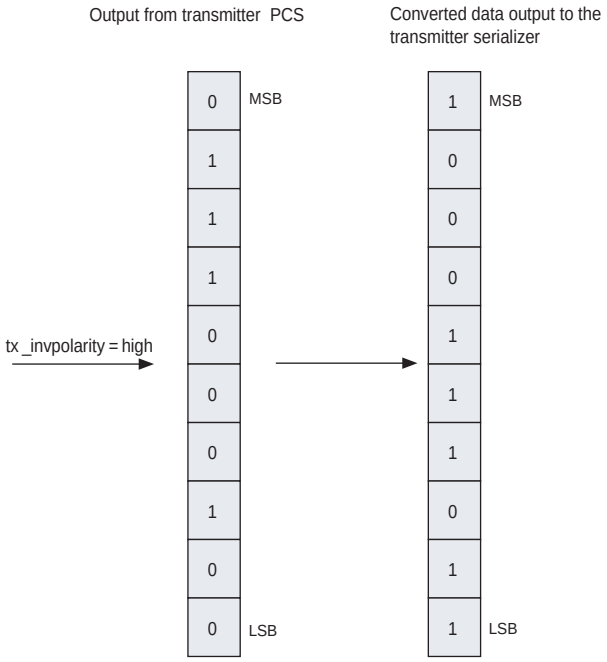
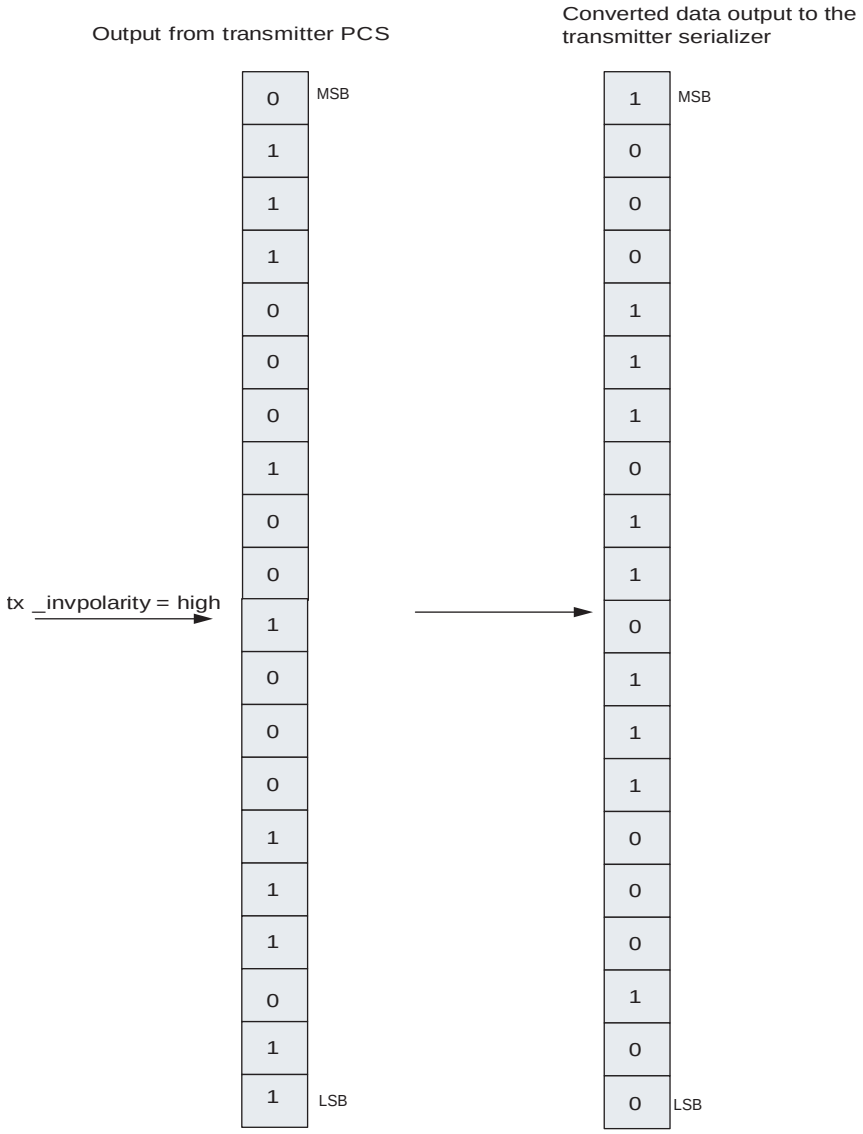


Figure 1–36 shows the transmitter polarity inversion in double-width mode.

Figure 1–36. Transmitter Polarity Inversion in Double-Width Mode



Transmitter Bit Reversal

By default, the HardCopy IV GX transmit bit order is LSBit to MSBit. In single-width mode, the least significant bit of the 8- or 10-bit data word is transmitted first, followed by the most significant bit. In double-width mode, the least significant bit of the 16- or 20-bit data word is transmitted first, followed by the most significant bit. The transmitter bit reversal feature allows reversing the transmit bit order as MSBit to LSBit before it is forwarded to the serializer.

If you enable the transmitter bit reversal feature in Basic single-width mode, the 8-bit `D[7:0]` or 10-bit `D[9:0]` data at the input of the serializer is rewired to `D[0:7]` or `D[0:9]`, respectively. If you enable the transmitter bit reversal feature in Basic double-width mode, the 16-bit `D[15:0]` or 20-bit `D[19:0]` data at the input of the serializer is rewired to `D[0:15]` or `D[0:19]`, respectively.

Figure 1–37 shows the transmitter bit reversal feature in Basic single-width for a 10-bit wide datapath configuration.

Figure 1–37. Transmitter Bit Reversal Operation in Basic Single-Width Mode

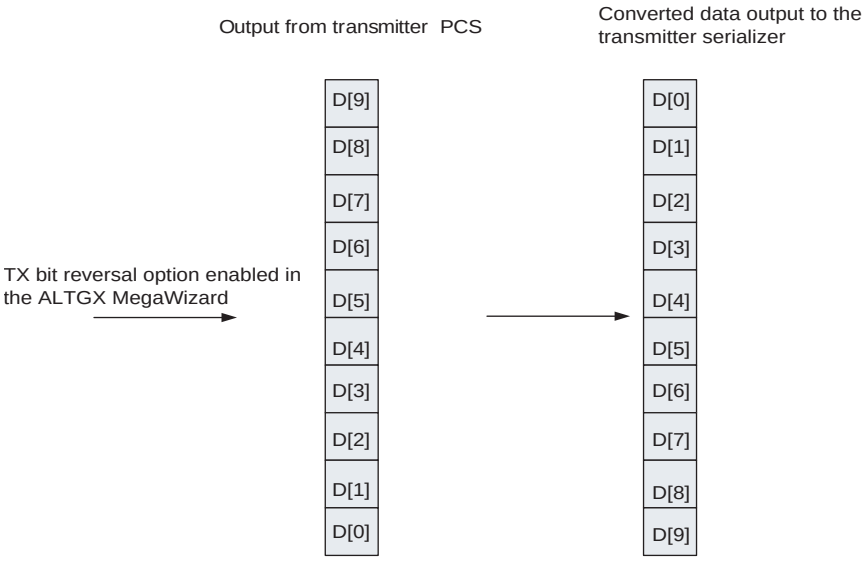
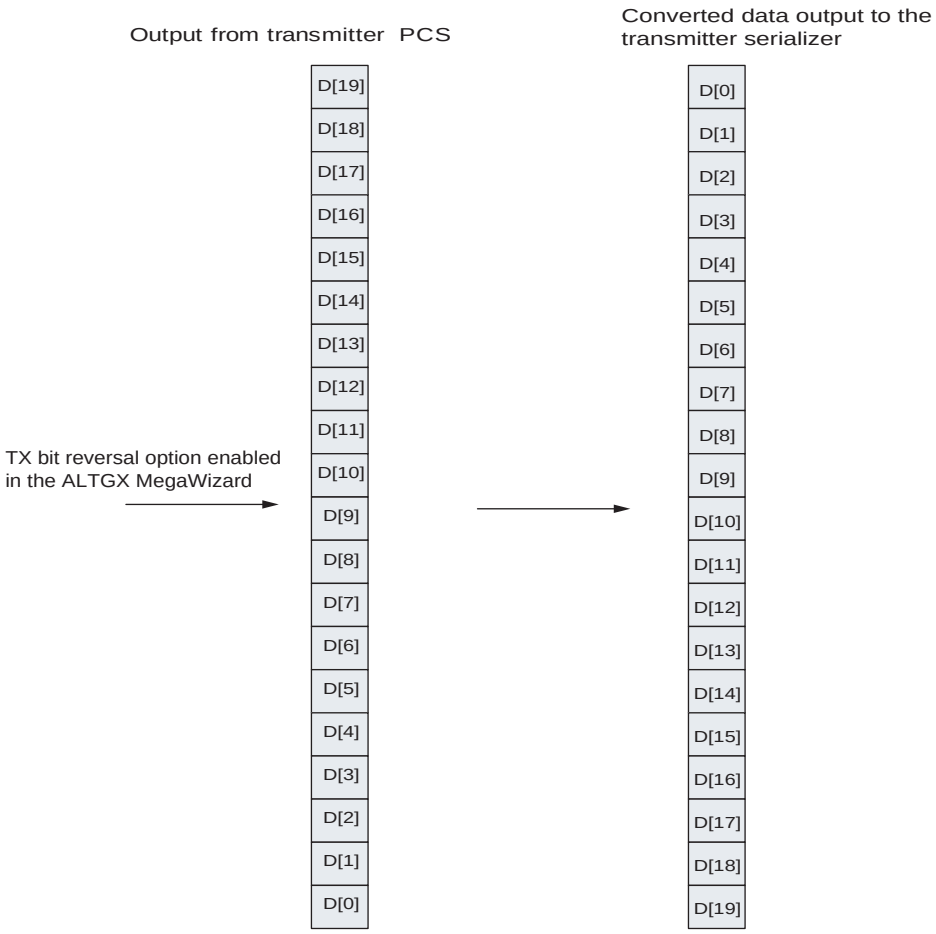


Figure 1–38 shows the transmitter bit reversal feature in Basic double-width mode for a 20-bit wide datapath configuration.

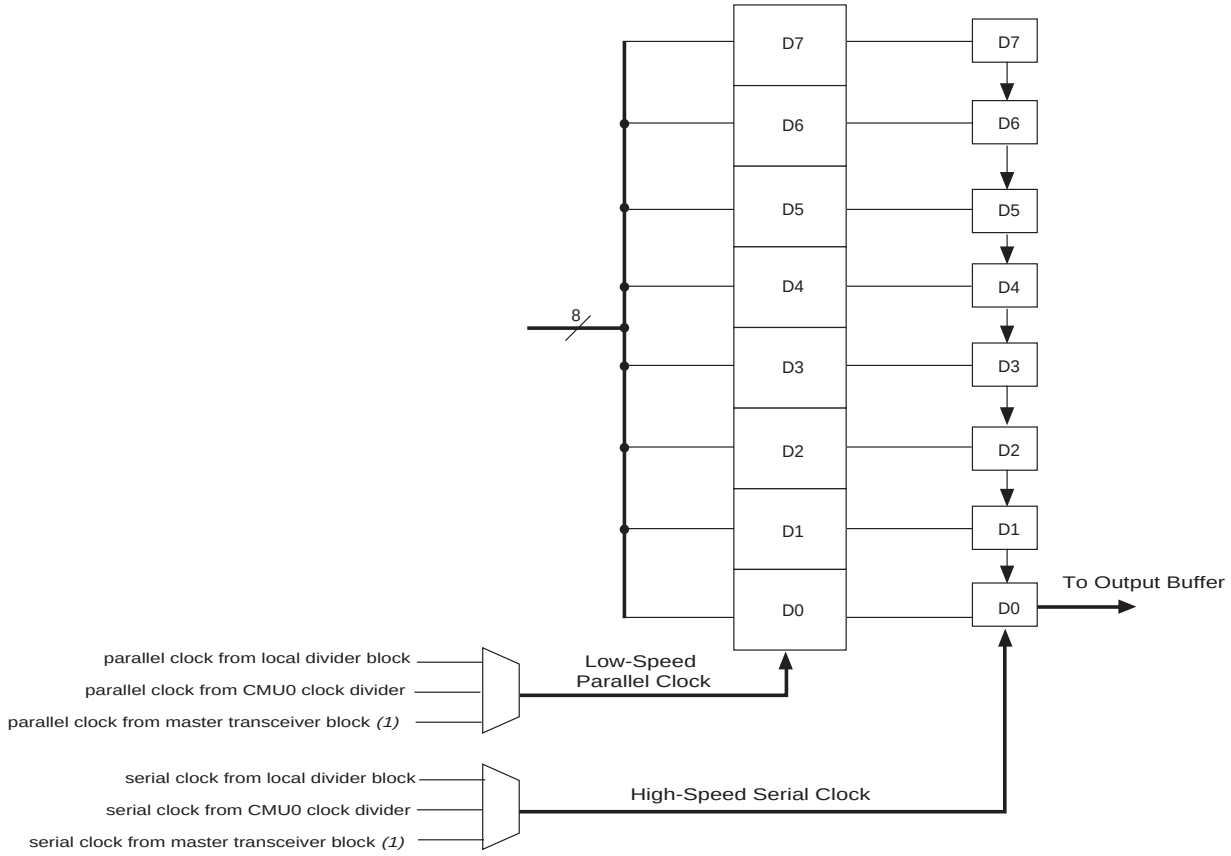
Figure 1–38. Transmitter Bit Reversal Operation in Basic Double-Width Mode



Serializer

The serializer converts the incoming low-speed parallel signal from the transceiver PCS to high-speed serial data and sends it to the transmitter buffer. The serializer supports an 8-bit or 10-bit serialization factor in single-width mode and a 16-bit or 20-bit serialization factor in double-width mode. The serializer block drives the serial data to the output buffer, as shown in Figure 1–39. The serializer block sends out the least significant bit of the input data. Figure 1–40 shows the serial bit order of the serializer block output. In this example, a constant 8'h6A (01101010) value is serialized and the serial data is transmitted from LSBit to MSBit.

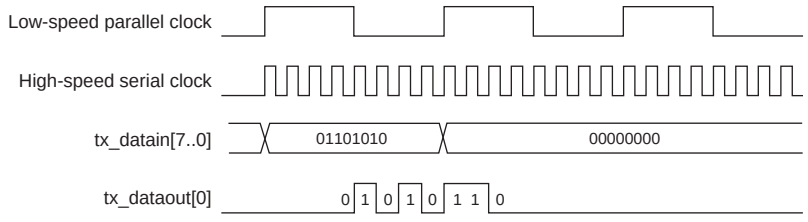
Figure 1–39. Serializer Block in 8-Bit PCS-PMA Interface



Note to Figure 1–39:

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

Figure 1–40. Serializer Bit Order *(Note 1)*



Note to Figure 1–40:

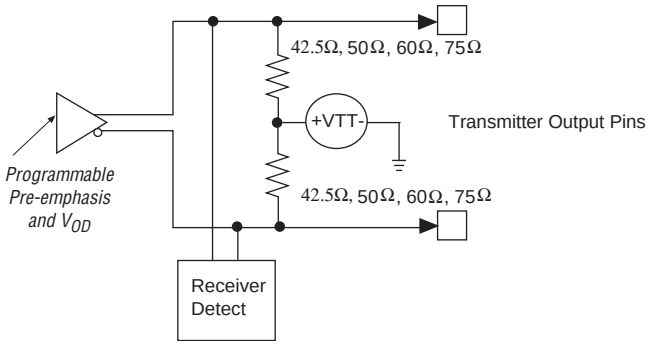
(1) It is assumed that the input data to the serializer is 8 bits (channel width = 8 bits or 16 bits with the 8B/10B encoder disabled).

Transmitter Output Buffer

The HardCopy IV GX transmitter buffers support 1.4-V and 1.5-V pseudo current mode logic (PCML) and can drive 40 inches of FR4 trace across two connectors. You can set the transmitter buffer voltage levels (V_{CCH}) through the ALTGX MegaWizard Plug-In Manager. With the 1.4 V and 1.5 V settings, you can run the transmitter channel from 600 Mbps to 6.5 Gbps and 600 Mbps to 4.25 Gbps, respectively. The

transmitter buffer power supply only provides voltage to the transmitter output buffers in the transceiver channels. The transmitter output buffer, shown in Figure 1–41, has additional circuitry to improve signal integrity, such as V_{OD}, programmable three-tap pre-emphasis circuit, internal termination circuitry, and receiver detect capability to support PCI Express (PIPE) functional mode.

Figure 1–41. Transmitter Output Buffer



Programmable Transmitter Termination

The HardCopy IV GX transmitter buffer 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 (refer to "Calibration Blocks" on page 1–192), 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 "Programmable Output Differential Voltage" on page 1–53.

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 OCT Assignment Settings

Assign To	Transmitter Serial Output Data Pin
Assignment Name:	output termination
Available Values:	OCT 85 Ω, OCT 100 Ω, OCT 120 Ω, OCT 150 Ω

Programmable Output Differential Voltage

The HardCopy IV GX device allows you to customize the differential output voltage to handle different trace lengths, various backplanes, and receiver requirements, as shown in Figure 1–42. You can change the V_{OD} values using the dynamic reconfiguration controller. Set the V_{OD} value through the `tx_vodctrl[2:0]` port of the dynamic reconfiguration controller. For example, to set V_{OD} to a value of 3, set the `tx_vodctrl[2:0]` to **011**.

Figure 1–42. V_{OD} (Differential) Signal Level

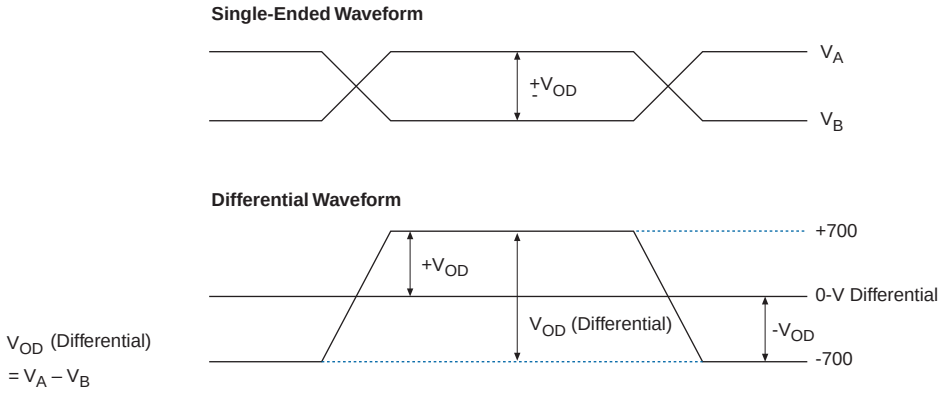


Table 1–10 shows the V_{OD} values for different termination resistor settings.

Table 1–10. Programmable V_{OD} Differential Peak-to-Peak (mV) *(Note 1)*

Values Shown in the ALTGX MegaWizard Plug-In Manager	85 Ω	100 Ω	120 Ω	150 Ω	Unit
0	170	200	240	300	mV
1	340	400	480	600	mV
2	510	600	720	900	mV
3	595	700	840	1050	mV
4	680	800	960	1200	mV
5	765	900	1080	1350	mV
6	850	1000	1200	—	mV
7	1020	1200	—	—	mV

Note to Table 1–10:

(1) These values are preliminary.

Programmable Pre-Emphasis

The programmable pre-emphasis module in each transmit buffer boosts high frequencies in the transmit data signal, which might be attenuated in the transmission media. Using pre-emphasis can maximize the data eye opening at the far-end receiver. The transmission line's transfer function can be represented in the frequency domain as a low-pass filter. Any frequency components below –3dB can pass through with minimal loss. Frequency components greater than –3dB are attenuated. This variation in frequency response yields data-dependent jitter and other inter-symbol interference (ISI) effects. By applying pre-emphasis, the high-frequency components are boosted; that is, pre-emphasized. Pre-emphasis equalizes the frequency response at the receiver so the difference between the low-frequency and high-frequency components is reduced, which minimizes the ISI effects from the transmission medium. Pre-emphasis requirements increase as data rates through legacy backplanes increase. You set the pre-emphasis settings in the ALTGX MegaWizard Plug-In Manager. The HardCopy IV GX transceiver provides three pre-emphasis taps: pre-tap, 1st post-tap, and 2nd 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 1st post-tap and 2nd post-tap sets the pre-emphasis on the transition bit and the successive bit, respectively. The pre-tap and 2nd post-tap also provide inversion control, shown by negative values on the corresponding tap settings in the ALTGX MegaWizard Plug-In Manager. The ALTGX MegaWizard Plug-In Manager only shows the valid pre-emphasis tap values for a selected V_{OD} and transmitter termination resistance setting.

Programmable Transmitter Output Buffer Power (VCCH)

The ALTGX MegaWizard Plug-In Manager provides an option to select V_{CCH}.

Two options are available for V_{CCH}: 1.4 V or 1.5 V. With 1.4 V, the data rate range is 600 Mbps to 6.5 Gbps; with 1.5 V, the data rate range is 600 Mbps to 4.25 Gbps.

Common Mode Voltage (VCM) Settings

HardCopy IV GX devices provide a VCM of 650 mV.

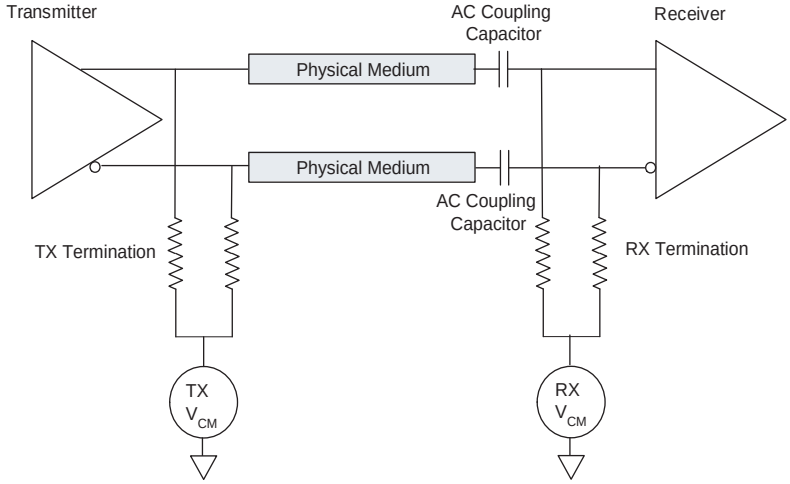
Link Coupling

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

AC-Coupled Links

In an AC-coupled link, the AC-coupling capacitor blocks the transmitter DC common mode voltage. The on-chip or off-chip receiver termination and biasing circuitry automatically restores the selected common mode voltage. Figure 1–43 shows an AC-coupled link.

Figure 1–43. AC-Coupled Link



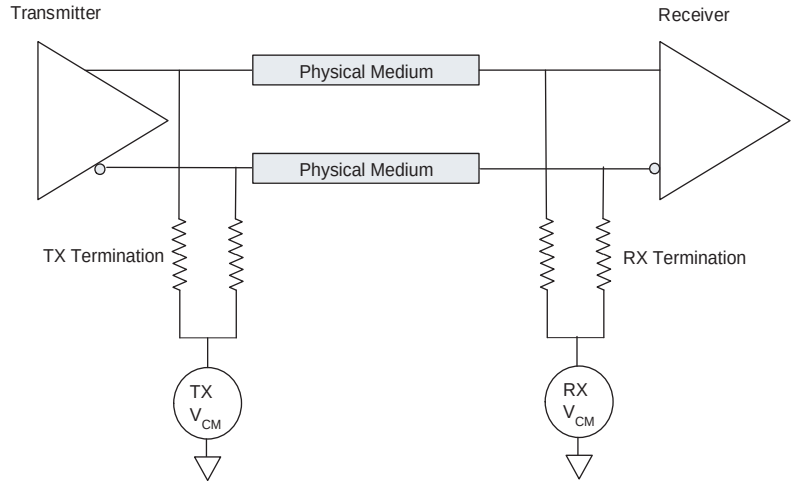
The following protocols supported by HardCopy IV GX devices mandate AC-coupled links:

- PCI Express (PIPE)
- Gigabit Ethernet
- Serial RapidIO
- XAUI
- SDI

DC-Coupled Links

In a DC-coupled link, the transmitter DC common mode voltage is seen unblocked at the receiver buffer. The link common mode voltage depends on the transmitter common mode voltage and the receiver common mode voltage. The on-chip or off-chip receiver termination and biasing circuitry must ensure compatibility between the transmitter and the receiver common mode voltage. Figure 1–44 shows a DC-coupled link.

Figure 1–44. DC-Coupled Link



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 devices, from 600 Mbps to 6.5 Gbps.

PCI Express (PIPE) Receiver Detect

The HardCopy IV GX transmitter buffer has a built-in receiver detection circuit for use in the PIPE mode for Gen1 and Gen2 data rates. This circuit detects if there is a receiver downstream by sending out a pulse on the common mode of the transmitter and monitoring the reflection. This mode requires the transmitter buffer to be tri-stated (in Electrical Idle mode), OCT utilization, and a 125 MHz `fixedclk` signal. You can enable this feature in PIPE mode by setting the `tx_forceelecidle` and `tx_detectrxloopback` ports to **1'b1**. Receiver detect circuitry is active only in the P1 power state.

For more information about power states, refer to the PCI Express (PIPE) 2.0 specification available from Intel.

In the P1 power state, the transmitter output buffer is tri-stated because the transmitter output buffer is in electrical idle. A high on the `tx_detectrxloopback` port triggers the receiver detect circuitry to alter the transmitter output buffer common mode voltage. The sudden change in common mode voltage effectively appears as a step voltage at the tri-stated transmitter buffer output. If a receiver (that complies with PIPE input impedance requirements) is present at the far end, the time constant of the step voltage is higher. If a receiver is not present or is powered down, the time constant of the step voltage is lower. The receiver detect circuitry snoops the transmitter buffer output for the time constant of the step voltage to detect the presence of the receiver at the far end. A high pulse is driven on the `pipephydonestatus` port and 3'b011 is driven on the `pipestatus` port to indicate that a receiver has been detected. There is some latency after asserting the `tx_detectrxloopback` signal, before the receiver detection is indicated on the `pipephydonestatus` port. For the signal timing to perform the receiver detect operation, refer to Figure 1–108 on page 1–132.

The `tx_forceelecidle` port must be asserted at least 10 parallel clock cycles prior to the `tx_detectrxloopback` port to ensure that the transmitter buffer is tri-stated.

PCI Express (PIPE) Electrical Idle

The HardCopy IV GX transmitter output buffer supports transmission of PIPE Electrical Idle (or individual transmitter tri-state). This feature is only active in PIPE mode. The `tx_forceelecidle` port puts the transmitter buffer in Electrical Idle mode. This port has a specific functionality in each power state. For the signal timing to perform the electrical idle transmission in PIPE mode, refer to Figure 1–107 on page 1–131.

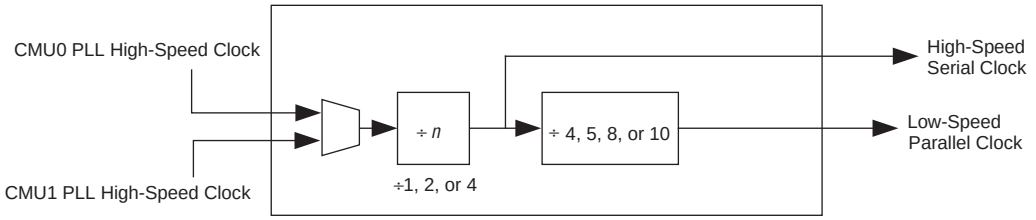
For use of the `tx_forceelecidle` signal under different power states, refer to the PIPE specification 2.0.

Transmitter Local Clock Divider Block

Each transmitter channel contains a local clock divider block. It receives the high-speed clock from the `CMU0` PLL or `CMU1` PLL and generates the high-speed serial clock for the serializer and the low-speed parallel clock for the transmitter PCS datapath. The low-speed parallel clock is also forwarded to the core fabric (`tx_clkout`). The local clock divider block allows each transmitter channel to run at /1, /2, or /4 of the CMU PLL data rate. The local clock divider block is used only in non-bonded functional modes (for example, GIGE, SONET/SDH, and SDI mode).

Figure 1–45 shows the transmitter local clock divider block.

Figure 1–45. Transmitter Local Clock Divider Block

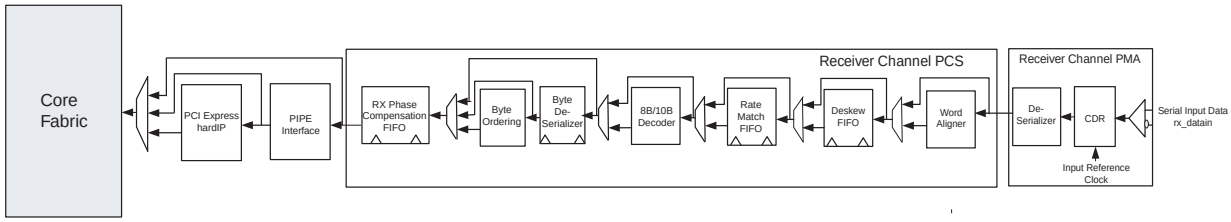


Receiver Channel Datapath

This section describes 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 core fabric-transceiver interface.

Figure 1–46 shows the receiver channel datapath in HardCopy IV GX devices.

Figure 1–46. 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
- PCI Express (PIPE) interface

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

Receiver Input Buffer

The 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 1–47 shows the receiver input buffer.

Figure 1–47. Receiver Input Buffer

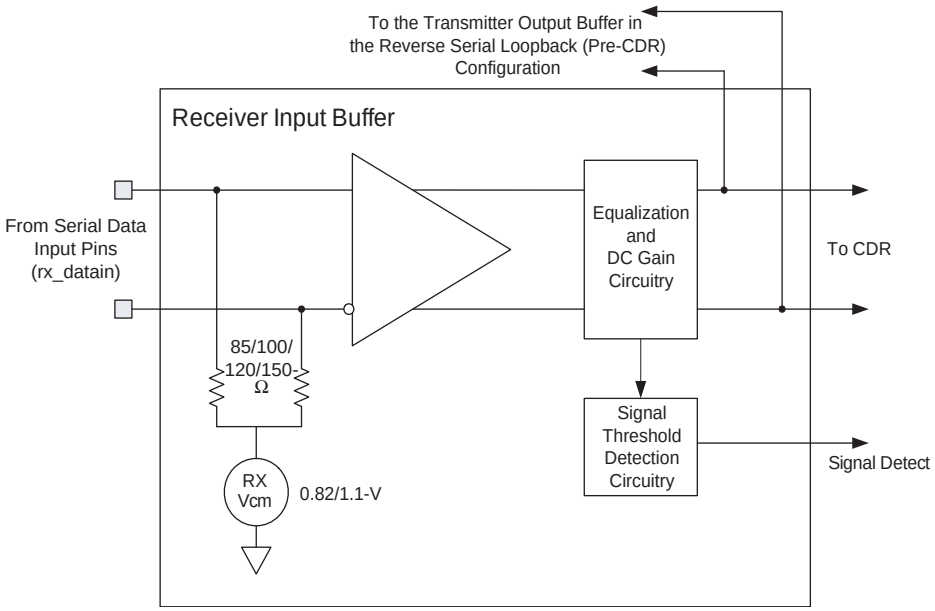


Table 1–11 shows the electrical features supported by the receiver input buffer.

Table 1–11. Electrical Features Supported by the Receiver Input Buffer

Data Rate (Gbps)	I/O Standard	Differential On-Chip Termination with Calibration (Ω)	Common Mode Voltage (V)	Coupling
0.6 to 6.5	1.4 V PCML	85, 100, 120, 150	0.82	AC, DC
	1.5 V PCML	85, 100, 120, 150	0.82	AC, DC
	2.5 V PCML	85, 100, 120, 150	0.82	AC
	LVPECL	85, 100, 120, 150	0.82	AC
	LVDS	85, 100, 120, 150	1.1	AC, DC

The HardCopy IV GX receiver buffer supports the following features:

- Programmable differential OCT
- Programmable common mode voltage
- AC and DC coupling
- Programmable equalization and DC gain

- Signal threshold detection circuitry

Programmable Differential On-Chip Termination

The HardCopy IV GX receiver buffers support optional differential on-chip termination resistors of 85, 100, 120, and 150 Ω. To select the desired receiver OCT resistor, make the assignments shown in Table 1–12 in the Quartus II software Assignment Editor.

Table 1–12. HardCopy IV GX Receiver On-Chip Termination Assignment Settings

Assign To	rx_datain (Receiver Input Data Pins)
Assignment Name:	Input Termination
Available Values:	OCT 85 Ω, OCT 100 Ω, OCT 120 Ω, OCT 150 Ω, Off

The HardCopy IV GX receiver OCT resistors have calibration support to compensate for process, voltage, and temperature variations. For more information about OCT calibration support, refer to "Calibration Blocks" on page 1–192.

Programmable Common Mode Voltage

The HardCopy IV GX receiver buffers have on-chip biasing circuitry to establish the required common mode voltage at the receiver input. It supports common mode voltage settings of 0.82 V and 1.1 V that you can select in the ALTGX MegaWizard Plug-In Manager.

You must select **0.82 V** as the receiver buffer common mode voltage for the following receiver input buffer I/O standards:

- 1.4-V PCML
- 1.5-V PCML
- 2.5-V PCML
- LVPECL

You must select **1.1 V** as the receiver buffer common mode voltage for the LVDS receiver input buffer I/O standard.

On-chip biasing circuitry is effective only if you select **on-chip receiver termination**. If you select **external termination**, you must implement off-chip biasing circuitry to establish the common mode voltage at the receiver input buffer.

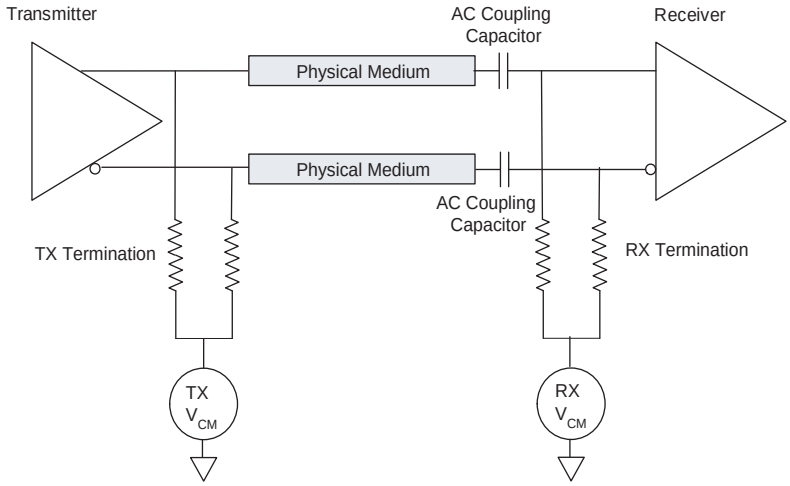
Link Coupling

A high-speed serial link can either be AC-coupled or DC-coupled, depending on the serial protocol being implemented. Most of the serial protocols require links to be AC-coupled, but protocols such as Common Electrical I/O (CEI) optionally allow DC coupling.

AC-Coupled Links

In an AC-coupled link, the AC coupling capacitor blocks the transmitter DC common mode voltage. The on-chip or off-chip receiver termination and biasing circuitry automatically restores the selected common mode voltage. Figure 1–48 shows an AC-coupled link.

Figure 1–48. AC-Coupled Link



Note to Figure 1–48:

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

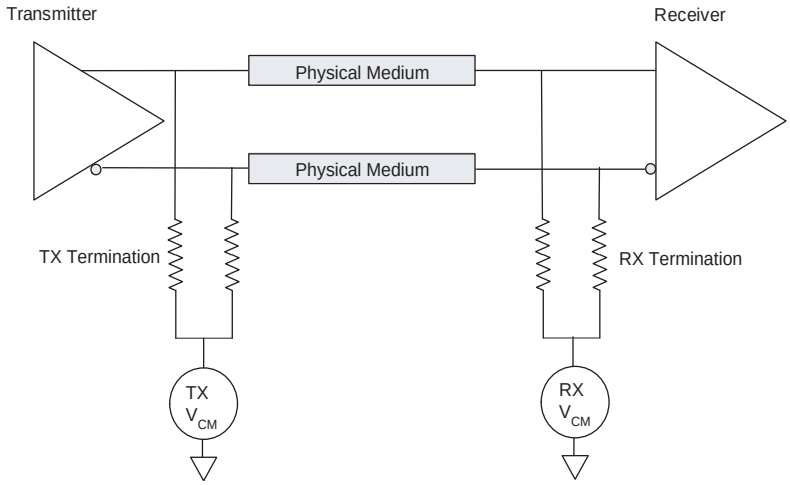
The following protocols supported by HardCopy IV GX devices mandate AC-coupled links:

- PCI Express (PIPE)
- Gigabit Ethernet
- Serial RapidIO
- XAUI
- SDI

DC-Coupled Links

In a DC-coupled link, the transmitter DC common mode voltage is seen unblocked at the receiver buffer. Link common mode voltage depends on the transmitter common mode voltage and the receiver common mode voltage. The on-chip or off-chip receiver termination and biasing circuitry must ensure compatibility between the transmitter and the receiver common mode voltage. Figure 1–49 shows a DC-coupled link.

Figure 1–49. DC-Coupled Link



Note to Figure 1–49:

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

You might choose to use the DC-coupled high-speed link for these functional modes only:

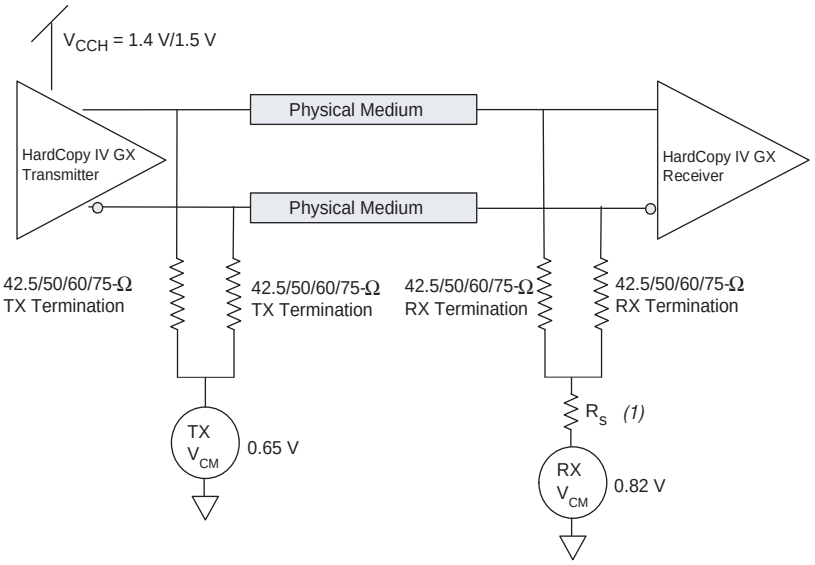
- Basic single- and double-width
- (OIF) CEI PHY interface

The following sections describe DC-coupling requirements for a high-speed link with a HardCopy IV GX device used as the transmitter, receiver, or both. Specifically, the following link configurations are described:

- HardCopy IV GX Transmitter (PCML) to HardCopy IV GX Receiver (PCML)
- Stratix II GX Transmitter (PCML) to HardCopy IV GX Receiver (PCML)
- HardCopy IV GX Transmitter (PCML) to Stratix II GX Receiver (PCML)
- LVDS Transmitter to HardCopy IV GX Receiver (PCML)

Figure 1–50 shows a typical HardCopy IV GX transmitter (PCML) to HardCopy IV GX Receiver (PCML) DC-coupled link.

Figure 1–50. HardCopy IV GX Transmitter (PCML) to HardCopy IV GX Receiver (PCML)



Note to Figure 1–50:

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

Table 1–13 shows the allowed transmitter and receiver settings in a HardCopy IV GX transmitter (PCML) to HardCopy IV GX receiver (PCML) DC-coupled link.

Table 1–13. Settings for a HardCopy IV GX Transmitter (PCML) to HardCopy IV GX Receiver (PCML) DC-Coupled Link

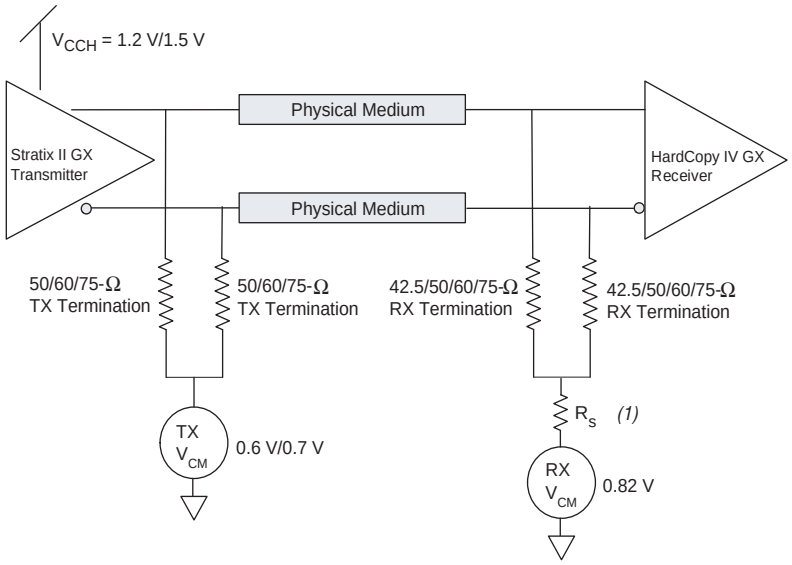
Transmitter (HardCopy IV GX) Settings				Receiver (HardCopy IV GX) Settings		
Data Rate	**VCCH *(1)***	**TX VCM**	**Differential Termination**	**Data Rate**	**RX VCM**	**Differential Termination**
600 - 6500 Mbps	1.4 V/1.5 V	0.65 V	85/100/120/150-Ω	600 - 6500 Mbps	0.82 V	85/100/120/150-Ω

Note to Table 1–13:

(1) V_{CCH} = 1.5 V can support data rates from 600 Mbps to 4250 Mbps. V_{CCH} = 1.4 V can support data rates from 600 Mbps to 6500 Mbps.

Figure 1–51 shows the Stratix II GX transmitter (PCML) to HardCopy IV GX receiver (PCML) coupled link.

Figure 1–51. Stratix II GX Transmitter (PCML) to HardCopy IV GX Receiver (PCML)



Note to Figure 1–51:

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

Table 1–14 shows the allowed transmitter and receiver settings in a Stratix II GX to HardCopy IV GX DC-coupled link.

Table 1–14. Settings for a Stratix II GX to HardCopy IV GX DC-Coupled Link

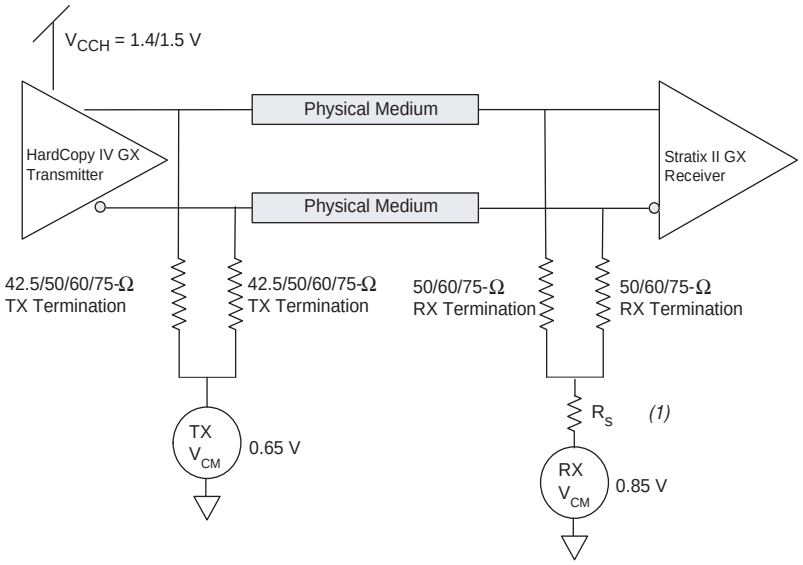
Transmitter (Stratix II GX) Settings				Receiver (HardCopy IV GX) Settings		
Data Rate	**VCCH *(1)***	**TX VCM *(1)***	**Differential Termination**	**Data Rate**	**RX VCM**	**Differential Termination**
600 - 6375 Mbps	1.5 V (1.5 V PCML)	0.6 V/0.7 V	100/120/150 Ω	600 - 6375 Mbps	0.82 V	100/120/150 Ω

Note to Table 1–14:

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

Figure 1–52 shows the HardCopy IV GX transmitter (PCML) to Stratix II GX receiver (PCML) DC-coupled link.

Figure 1–52. HardCopy IV GX Transmitter (PCML) to Stratix II GX Receiver (PCML)



Note to Figure 1–52:

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

Table 1–15 shows the allowed transmitter and receiver settings in a HardCopy IV GX transmitter (PCML) to Stratix II GX receiver (PCML) DC-coupled link.

Table 1–15. Settings for a HardCopy IV GX to Stratix II GX DC-Coupled Link

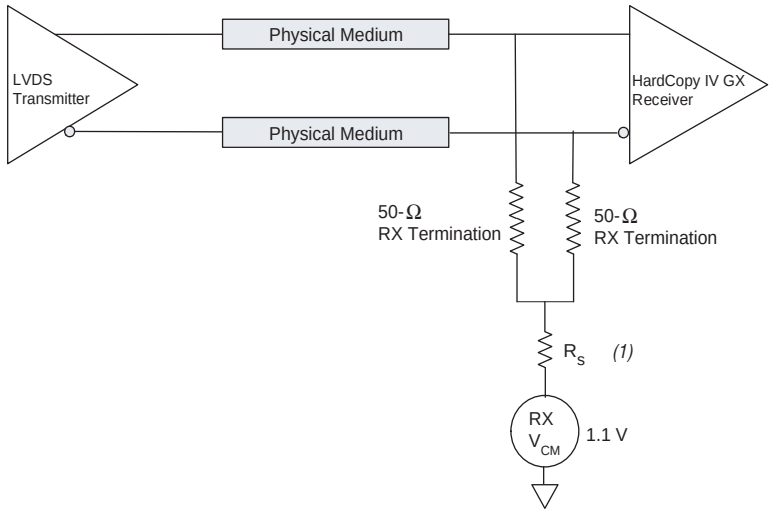
Transmitter (HardCopy IV GX) Settings				Receiver (Stratix II GX) Settings			
Data Rate	**VCCH *(1)***	**TX VCM**	**Differential Termination**	**Data Rate**	**I/O Standard**	**RX VCM**	**Differential Termination**
600 - 6375 Mbps	1.4/1.5 V	0.65 V	100/120/150 Ω	600 - 6375 Mbps	1.4/1.5 V PCML	0.85 V	100/120/150 Ω

Note to Table 1–15:

(1) V_{CCH} = 1.5 V can support data rates from 600 Mbps to 4250 Mbps. V_{CCH} = 1.4 V can support data rates from 600 Mbps to 6375 Mbps.

Figure 1–53 shows the LVDS transmitter to HardCopy IV GX receiver (PCML) coupled link.

Figure 1–53. LVDS Transmitter to HardCopy IV GX Receiver (PCML)



Note to Figure 1–53:

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

Table 1–16 shows the allowed transmitter and receiver settings in a LVDS transmitter to HardCopy IV GX receiver DC-coupled link.

Table 1–16. Settings for a LVDS transmitter to HardCopy IV GX Receiver DC-Coupled Link *(Note 1)*

Receiver (HardCopy IV GX) Settings		
RX VCM	**Differential Termination**	**R_S**
1.1 V	100-Ω	*(2)*

Notes to Table 1–16:

(1) When DC-coupling an LVDS transmitter to the HardCopy IV GX receiver, use RX Vcm = 1.1 V and series resistance value RS to verify compliance with the LVDS specification.

(2) Pending characterization.

Programmable Equalization and DC Gain

The transfer function of the physical medium can be represented as a low-pass filter in the frequency domain. Frequency components below –3 dB frequency pass through with minimal loss. Frequency components greater than –3 dB frequency are attenuated as a function of frequency due to skin-effect and dielectric losses. This variation in frequency response yields data-dependent jitter and other ISI effects, which can cause incorrect sampling of the input data.

Each HardCopy IV GX receiver buffer has independently programmable equalization circuitry that boosts the high-frequency gain of the incoming signal, thereby compensating for the low-pass filter effects of the physical medium. The amount of high-frequency gain required depends on the loss characteristics of the physical medium. The HardCopy IV GX equalization circuitry supports 16 equalization settings that provide up to 16 dB of high-frequency boost. You can select the appropriate equalization setting in the ALTGX MegaWizard Plug-In Manager.

The HardCopy IV GX receiver buffer also supports programmable DC gain circuitry. Unlike equalization circuitry, DC gain circuitry provides equal boost to the incoming signal across the frequency spectrum. The receiver buffer supports DC gain settings of 0, 3, 6, 9, and 12 dB. You can select the appropriate DC gain setting in the ALTGX MegaWizard Plug-In Manager.

Signal Threshold Detection Circuitry

In PCI Express (PIPE) mode, you can enable the optional signal threshold detection circuitry by not selecting the **Force signal detection** option in the ALTGX MegaWizard Plug-In Manager. If enabled, this option senses whether the signal level present at the receiver input buffer is above the signal detect threshold voltage that you specified in the **What is the signal detect and signal loss threshold?** option in the ALTGX MegaWizard Plug-In Manager.

The appropriate signal detect threshold level that complies with the PIPE compliance parameter VRX-IDLE-DETDIFFp-p is pending characterization.

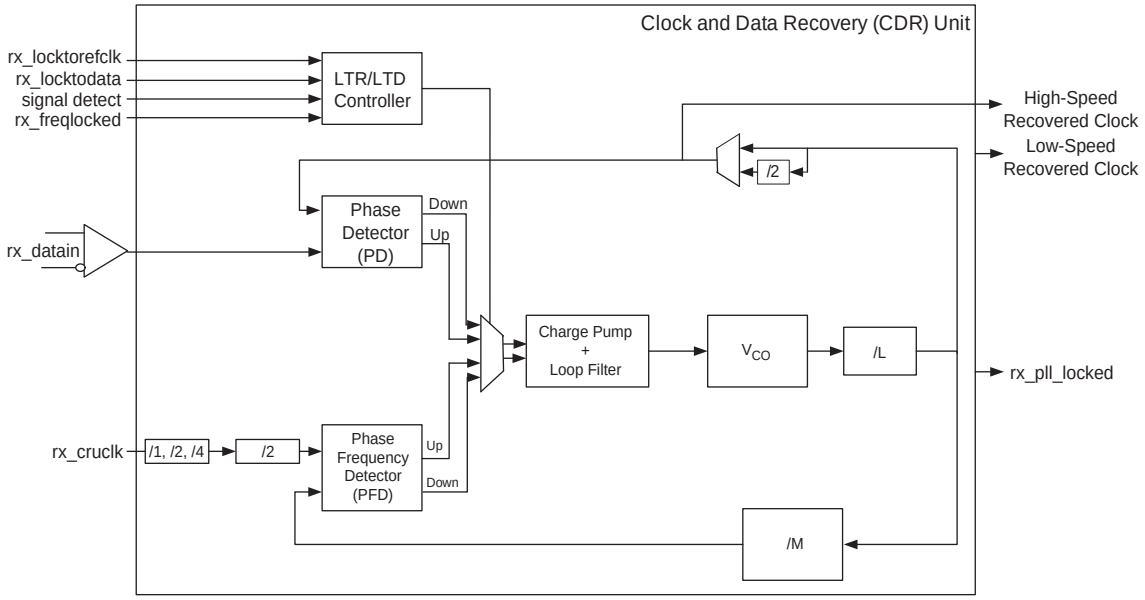
Signal threshold detection circuitry has a hysteresis response that filters out any high-frequency ringing caused by inter-symbol interference or high-frequency losses in the transmission medium. If the signal threshold detection circuitry senses the signal level present at the receiver input buffer to be higher than the signal detect threshold, it asserts the `rx_signaldetect` signal high. Otherwise, the signal threshold detection circuitry de-asserts the `rx_signaldetect` signal low. If you select the **Force signal detection** option in the ALTGX MegaWizard Plug-In Manager, `rx_signaldetect` is always asserted high, irrespective of the signal level on the receiver input buffer.

The `rx_signaldetect` signal is also used by the lock-to-reference/lock-to-data (LTR/LTD) controller in the receiver CDR to switch between the LTR and LTD lock modes. When the signal threshold detection circuitry de-asserts the `rx_signaldetect` signal, the LTR/LTD controller switches the receiver CDR from LTD to LTR lock mode. For more information, refer to "LTR/LTD Controller" on page 1–71.

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 1–54 shows the CDR block diagram.

Figure 1–54. Clock and Data Recovery Unit



The CDR operates either in LTR mode or LTD mode. In LTR mode, the CDR tracks the input reference clock. In LTD mode, the CDR tracks the incoming serial data.

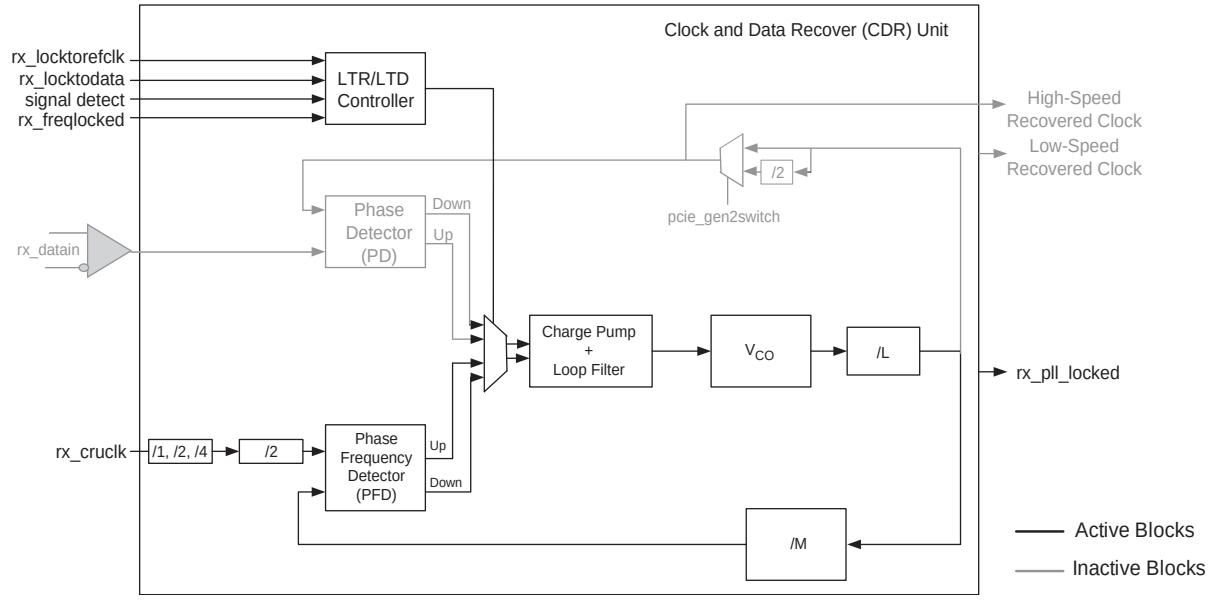
After the receiver power up and reset cycle, the CDR must be kept in LTR mode until it locks to the input reference clock. Once locked to the input reference clock, the CDR output clock is trained to the configured data rate. The CDR can now switch to LTD mode to recover the clock from incoming data. The LTR/LTD controller controls the switch between LTR and LTD modes.

Lock-to-Reference Mode

In LTR mode, the phase frequency detector in the CDR tracks the receiver input reference clock, `rx_cruclk`. The PFD controls the charge pump that tunes the VCO in the CDR. Depending on the data rate and the selected input reference clock frequency, the Quartus II software automatically selects the appropriate /M and /L divider values such that the CDR output clock frequency is half the data rate. An active high, the `rx_pll_locked` status signal is asserted to indicate that the CDR has locked to phase and frequency of the receiver input reference clock. Figure 1–55 shows active blocks when CDR is in LTR mode.

The phase detector (PD) is inactive in LTR mode.

Figure 1–55. CDR in Lock-To-Reference Mode



You can drive the receiver input reference clock with the following clock sources:

- Dedicated REFCLK pins (`refclk0` and `refclk1`) of the associated transceiver block
- Inter-transceiver block (ITB) clock lines from other transceiver blocks on the same side of the device (up to six ITB clock lines, two from each transceiver block)
- Global PLD clock driven by a dedicated clock input pin
- Clock output from the left and right PLLs in the core fabric

Table 1–17 shows CDR specifications in LTR mode.

Table 1–17. CDR Specifications in Lock-To-Reference Mode

Parameter	Value
Input Reference Clock Frequency	50 MHz to 637.5 MHz
PFD Input Frequency	50 MHz to 325 MHz
/M Divider	4, 5, 8, 10, 16, 20, 25
/L Divider	1, 2, 4, 8

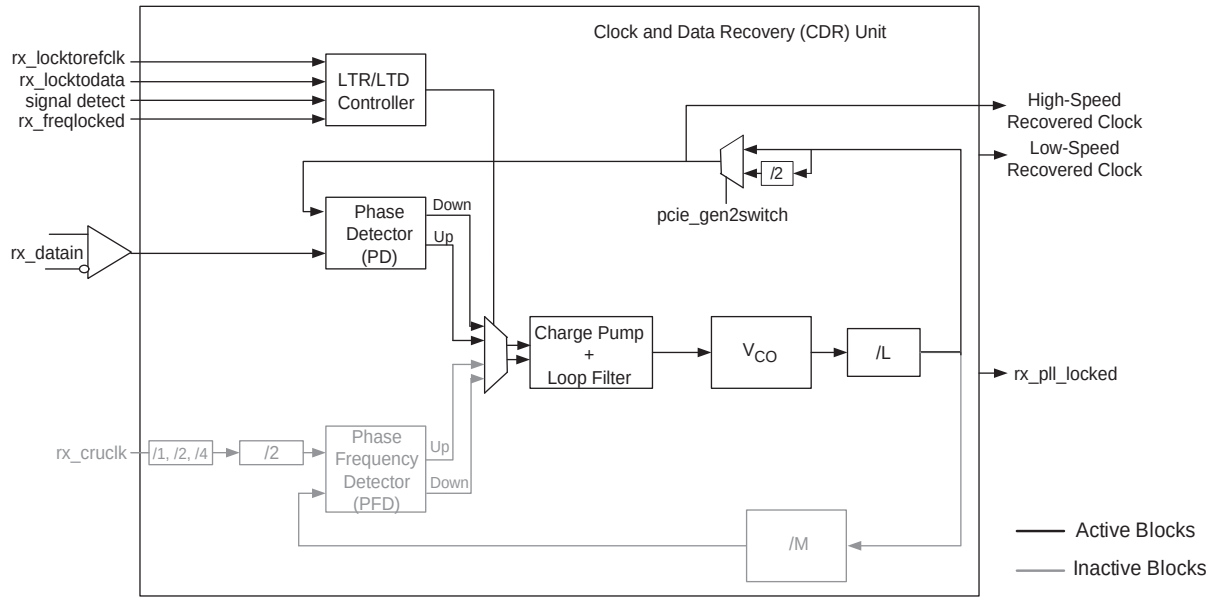
For input reference clock frequencies greater than 325 MHz, the Quartus II software automatically selects the appropriate /1, /2, /4 pre-divider to meet the PFD input frequency limitation of 325 MHz.

Lock-to-Data Mode

The CDR must be in LTD mode to recover the clock from the incoming serial data during normal operation. In LTD mode, the phase detector (PD) in the CDR tracks the incoming serial data at the receiver buffer. Depending on the phase difference between the incoming data and the CDR output clock, the PD controls the CDR charge pump that tunes the VCO. Figure 1–56 shows active blocks 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.

Figure 1–56. CDR in Lock-To-Data 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.

PCI Express (PIPE) Clock Switch Circuitry

The feedback path from the CDR VCO to the PD has a /2 divider that is used in PIPE mode configured at Gen2 (5 Gbps) data rate for the dynamic switch between Gen1 (2.5 Gbps) and Gen2 (5 Gbps) signaling rates. When the PHY-MAC layer instructs a Gen2-to-Gen1 signaling rateswitch, the /2 divider is enabled. When the PHY-MAC layer instructs a Gen1-to-Gen2 signaling rateswitch, the /2 divider is disabled. For more information about the PIPE signaling rateswitch, refer to "Dynamic Switch Between Gen1 (2.5 Gbps) and Gen2 (5 Gbps) Signaling Rate" on page 1–138.

The /2 divider in the receiver CDR between the VCO and the PD is disabled in all other functional modes.

LTR/LTD Controller

The LTR/LTD controller controls whether the CDR is in LTR or LTD mode. You can configure the LTR/LTD controller either in automatic lock mode or manual lock mode.

Two optional input ports (`rx_locktorefclk` and `rx_locktodata`) allow you to configure the LTR/LTD controller in either automatic lock mode or manual lock mode. Table 1–18 shows the relationship between these optional input ports and the LTR/LTD controller lock mode.

Table 1–18. 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
- 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
- The CDR output clock is not within the configured PPM frequency threshold setting with respect to the input reference clock

The switch from LTD to LTR mode is indicated by the de-assertion of the `rx_freqlocked` signal.

Manual Lock Mode

In automatic lock mode, the LTR/LTD controller relies on the PPM detector and the phase relationship detector to set the CDR in LTR or LTD mode. The PPM detector and phase relationship detector reaction times can be too long for some applications that require faster CDR lock time. You can manually control the CDR to reduce its lock time using the `rx_locktorefclk` and `rx_locktodata` ports. In manual lock mode, the LTR/LTD controller sets the CDR in LTR or LTD mode depending on the logic level on the `rx_locktorefclk` and `rx_locktodata` signals.

When the `rx_locktorefclk` signal is asserted high, the LTR/LTD controller forces the CDR to lock to the reference clock. When the `rx_locktodata` signal is asserted high, it forces the CDR to lock to data. When both signals are asserted, the `rx_locktodata` signal takes precedence over the `rx_locktorefclk` signal, forcing the CDR to lock to data.

When the `rx_locktorefclk` signal is asserted high, the `rx_freqlocked` signal does not have any significance and is always driven low, indicating that the CDR is in LTR mode. When the `rx_locktodata` signal is asserted high, the `rx_freqlocked` signal is always driven high, indicating that the CDR is in LTD mode. If both signals are de-asserted, the CDR is in automatic lock mode.

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

Deserializer

The deserializer block clocks in serial input data from the receiver buffer using the high-speed serial recovered clock and deserializes it using the low-speed parallel recovered clock. It forwards the deserialized data to the receiver PCS channel.

In single-width mode, the deserializer supports 8-bit and 10-bit deserialization factors. In double-width mode, the deserializer supports 16-bit and 20-bit deserialization factors.

Figure 1–57 shows the deserializer operation in single-width mode with a 10-bit deserialization factor.

Figure 1–57. Deserializer Operation in Single-Width Mode

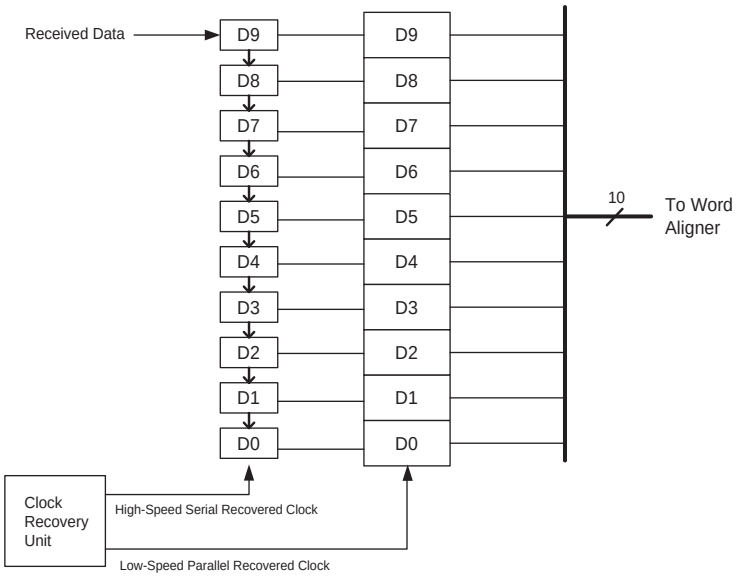
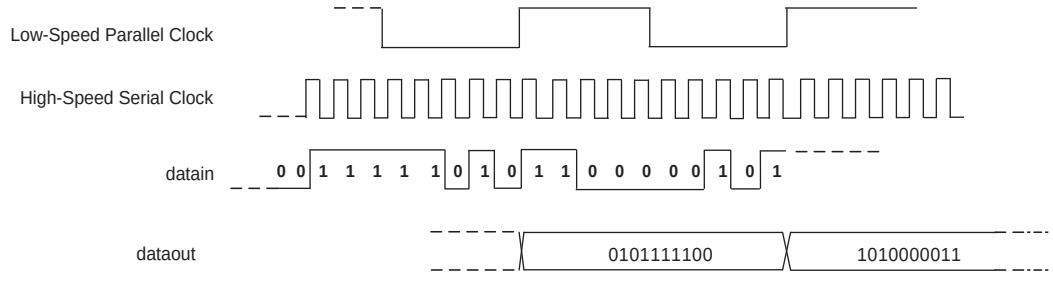


Figure 1–58 shows the serial bit order of the deserializer block input and the parallel data output of the deserializer block in single-width mode with a 10-bit deserialization factor. The serial stream (0101111100) is deserialized to a value 10'h17C. The serial data is assumed to be received LSB to MSB.

Figure 1–58. Deserializer Bit Order in Single-Width Mode



Word Aligner

Because the data is serialized before transmission and then deserialized at the receiver, it loses the word boundary of the upstream transmitter upon deserialization. The word aligner receives parallel data from the deserializer and restores the word boundary based on a pre-defined alignment pattern that must be received during link synchronization.

Serial protocols such as PCI Express (PIPE), XAUI, Gigabit Ethernet, Serial RapidIO, and SONET/SDH, specify a standard word alignment pattern. For proprietary protocols, the 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:

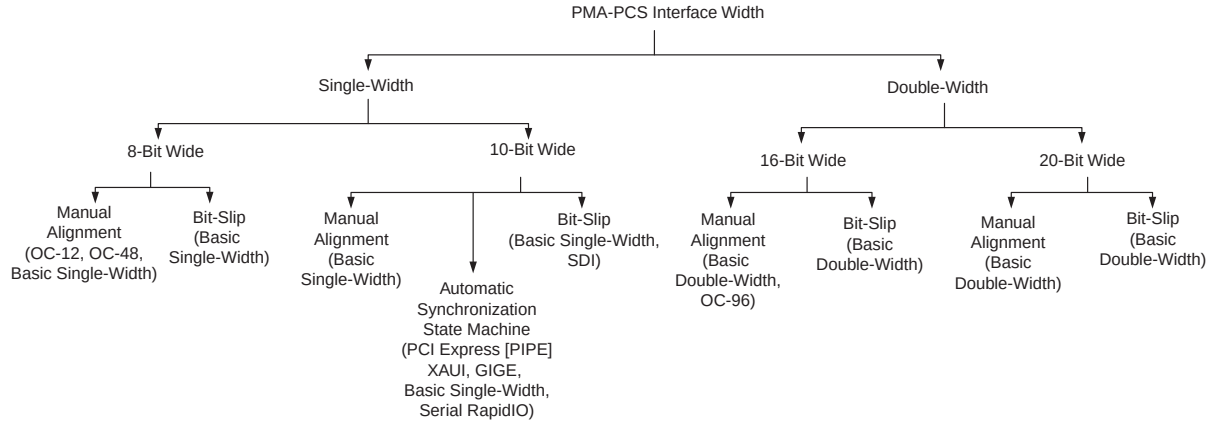
- Synchronization state machine in functional modes such as PCI Express (PIPE), XAUI, GIGE, Serial RapidIO, and Basic single-width
- Programmable run length violation detection in all functional modes
- Receiver polarity inversion in all functional modes except PIPE
- Receiver bit reversal in Basic single-width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

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

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

Figure 1–59 shows the word aligner operation in all supported configurations.

Figure 1–59. Word Aligner in All Supported Configurations



Word Aligner in Single-Width Mode

In single-width mode, the PMA-PCS interface is either 8 bit or 10 bit wide. In 8 bit wide PMA-PCS interface modes, the word aligner receives 8 bit wide data from the deserializer. In 10 bit wide PMA-PCS interface modes, the word aligner receives 10 bit wide data from the deserializer. Depending on the configured functional mode, you can configure the word aligner in manual alignment mode, automatic synchronization state machine mode, or bit-slip mode.

Word Aligner in Single-Width Mode with 8-Bit PMA-PCS Interface Modes

The following functional modes support the 8-bit PMA-PCS interface:

- SONET/SDH OC-12
- SONET/SDH OC-48
- Basic single-width

Table 1–19 shows the word aligner configurations allowed in functional modes with an 8-bit PMA-PCS interface.

Table 1–19. Word Aligner Configurations with an 8-Bit PMA-PCS Interface

Functional Mode	Allowed Word Configurations	Allowed Word Alignment Pattern Length
SONET/SDH OC-12	Manual Alignment	16 bits
SONET/SDH OC-48	Manual Alignment	16 bits
Basic single-width	Manual Alignment, Bit-Slip	16 bits

Manual Alignment Mode Word Aligner with 8-Bit PMA-PCS Interface Modes

In manual alignment mode, the word aligner operation is controlled by the input signal `rx_enapatternalign`. The word aligner operation is edge-sensitive to the `rx_enapatternalign` signal. After de-assertion of `rx_digitalreset`, a rising edge on the `rx_enapatternalign` signal triggers the word aligner to look for the word alignment pattern in the received data stream. In SONET/SDH OC-12 and OC-48 modes, the word aligner looks for 16'hF628 (A1A2) or 32'hF6F62828 (A1A1A2A2), depending on whether the input signal `rx_a1a2size` is driven low or high, respectively. In Basic single-width mode, the word aligner looks for the 16-bit word alignment pattern programmed in the ALTGX MegaWizard Plug-In Manager. The word aligner aligns the 8-bit word boundary to the first word alignment pattern received after the rising edge on the `rx_enapatternalign` signal.

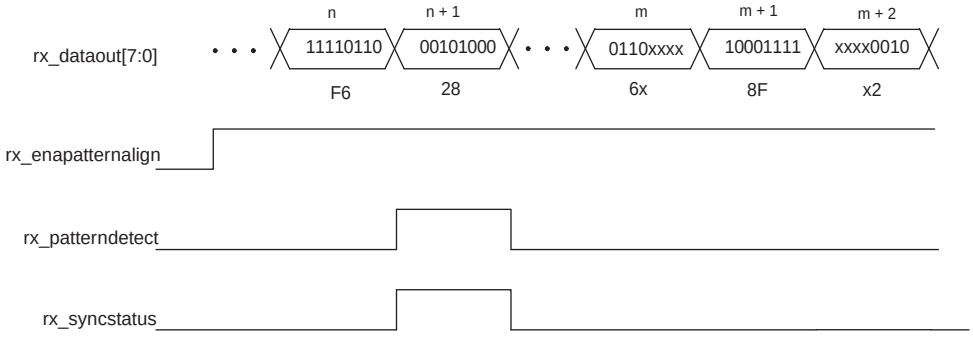
Two status signals, `rx_syncstatus` and `rx_patterndetect`, with the same latency as the datapath, are forwarded to the core fabric to indicate word aligner status. On receiving the first word alignment pattern after the rising edge on the `rx_enapatternalign` signal, both the `rx_syncstatus` and `rx_patterndetect` signals are driven high for one parallel clock cycle synchronous to the MSByte of the word alignment pattern. Any word alignment pattern received thereafter in the same word boundary causes only the `rx_patterndetect` signal to go high for one clock cycle.

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

Figure 1–60 shows word aligner behavior in SONET/SDH OC-12 functional mode. The LSByte (8'hF6) and the MSByte (8'h28) of the 16-bit word alignment pattern are received in parallel clock cycles n and n + 1, respectively. The `rx_syncstatus` and `rx_patterndetect` signals are both driven high for one parallel clock cycle synchronous to the MSByte (8'h28) of the word alignment pattern. After initial word alignment, the 16-bit word alignment pattern is again received across the word boundary in clock cycles m, m + 1, and m + 2. The word aligner does not re-align to the new word boundary because of the lack of a preceding rising edge on the

rx_enapatternalign signal. If you create a rising edge on the rx_enapatternalign signal before the word alignment pattern is received across clock cycles m, m + 1, and m + 2, the word aligner re-aligns to the new word boundary, causing both the rx_syncstatus and rx_patterndetect signals to go high for one parallel clock cycle.

Figure 1–60. Bit-Slip Mode in 8-Bit PMA-PCS Interface Mode



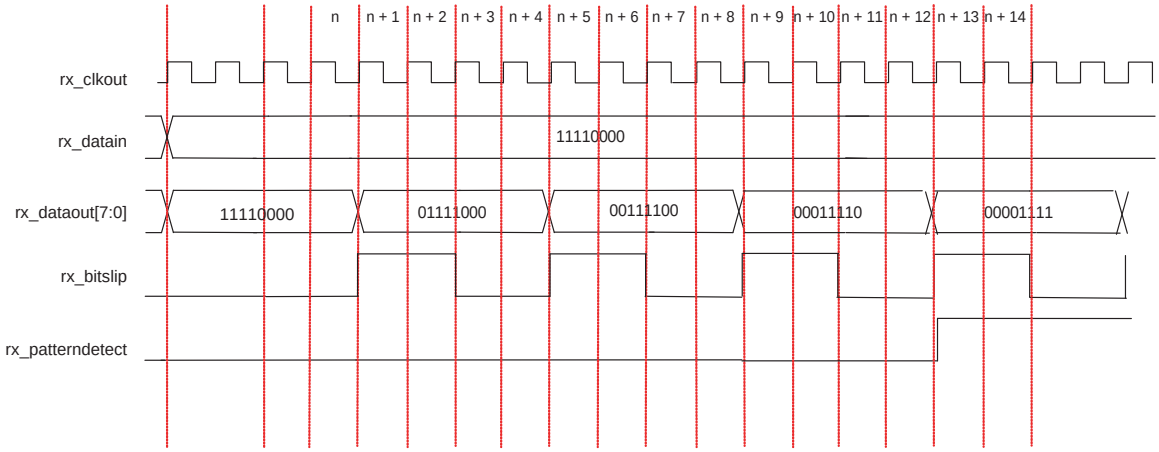
Bit-Slip Mode Word Aligner with 8-Bit PMA-PCS Interface Modes

Basic single-width mode with 8-bit PMA-PCS interface width allows the word aligner to be configured in bit-slip mode. The word aligner operation is controlled by the input signal rx_bitslip in bit-slip mode. At every rising edge of the rx_bitslip signal, the bit-slip circuitry slips one bit into the received data stream, effectively shifting the word boundary by one bit. In bit-slip mode, the word aligner status signal rx_patterndetect is driven high for one parallel clock cycle when the received data after bit-slipping matches the 16-bit word alignment pattern programmed in the ALTGX MegaWizard Plug-In Manager.

You can implement a bit-slip controller in the core fabric that monitors either the rx_dataout signal and/or the rx_patterndetect signal and controls the rx_bitslip signal to achieve word alignment.

Figure 1–61 shows an example of the word aligner configured in bit-slip mode. For this example, consider that 8'b11110000 is received back-to-back and 16'b0000111100011110 is specified as the word alignment pattern. A rising edge on the rx_bitslip signal at time n + 1 slips a single bit 0 at the MSB position, forcing the rx_dataout to 8'b01111000. Another rising edge on the rx_bitslip signal at time n + 5 forces rx_dataout to 8'b00111100. Another rising edge on the rx_bitslip signal at time n + 9 forces rx_dataout to 8'b00011110. Another rising edge on the rx_bitslip signal at time n + 13 forces the rx_dataout to 8'b00001111. At this instance, rx_dataout in cycles n + 12 and n + 13 is 8'b00011110 and 8'b00001111, respectively, which matches the specified 16-bit alignment pattern 16'b0000111100011110. This results in the assertion of the rx_patterndetect signal.

Figure 1–61. Word Aligner Configured in Bit-Slip Mode



Word Aligner in Single-Width Mode with 10-Bit PMA-PCS Interface Modes

The following functional modes support the 10-bit PMA-PCS interface:

- PCI Express (PIPE) Gen1 and Gen2
- Serial RapidIO
- XAUI
- GIGE
- SDI
- Basic single-width mode

This section describes the following word aligner 10-bit PMA-PCS interface modes:

- Automatic synchronization state machine mode with 10-bit PMA-PCS interface mode
- Manual alignment mode with 10-bit PMA-PCS interface mode
- Bit-slip mode in 10-bit PMA-PCS interface mode

Table 1–20 shows the word aligner configurations allowed in functional modes with a 10-bit PMA-PCS interface.

Table 1–20. Word Aligner Configurations with a 10-Bit PMA-PCS Interface

Functional Mode	Allowed Word Aligner Configurations	Allowed Word Alignment Pattern Length
PCI Express (PIPE)	Automatic synchronization state machine	10 bits
Serial RapidIO	Automatic synchronization state machine	10 bits
XAUI	Automatic synchronization state machine	7 bits, 10 bits
GIGE	Automatic synchronization state machine	7 bits, 10 bits
SDI	Bit-slip	N/A
Basic single-width mode	Manual Alignment, Automatic synchronization state machine, Bit-slip	7 bits, 10 bits

Automatic Synchronization State Machine Mode Word Aligner with 10-Bit PMA-PCS Interface Mode

Protocols such as PCI Express (PIPE), XAUI, Gigabit Ethernet, and Serial RapidIO require the receiver PCS logic to implement a synchronization state machine to provide hysteresis during link synchronization. Each of these protocols defines a specific number of synchronization code groups that the link must receive to acquire synchronization and a specific number of erroneous code groups that it must receive to fall out of synchronization.

In PIPE, XAUI, Gigabit Ethernet, and Serial RapidIO functional modes, the Quartus II software configures the word aligner in automatic synchronization state machine mode. It automatically selects the word alignment pattern length and pattern as specified by each protocol. In each of these functional modes, the protocol-compliant synchronization state machine is implemented in the word aligner.

In Basic single-width functional mode with 10-bit PMA-PCS interface, you can configure the word aligner in automatic synchronization state machine mode by selecting the **Use the built-in synchronization state machine** option in the ALTGX MegaWizard Plug-In Manager. It also allows you to program a custom 7-bit or 10-bit word alignment pattern that the word aligner uses for synchronization.

The 10-bit input data to the word aligner configured in automatic synchronization state machine mode must be 8B/10B encoded.

Table 1–21 shows the synchronization state machine parameters that the Quartus II software allows in supported functional modes. The synchronization state machine parameters are fixed for PIPE, XAUI, GIGE, and Serial RapidIO modes as specified by the respective protocol. For Basic single-width mode, you can program these parameters as suited to your proprietary protocol implementation.

Table 1–21. Synchronization State Machine Functional Modes

Functional Mode	PCI Express (PIPE)	XAUI	GIGE	Serial RapidIO	Basic Single-Width Mode
Number of valid synchronization code groups or ordered sets received to achieve synchronization	4	4	3	127	1 to 256
Number of erroneous code groups received to lose synchronization	17	4	4	3	1 to 64
Number of continuous good code groups received to reduce the error count by one	16	4	4	255	1 to 256

After de-assertion of the `rx_digitalreset` signal in automatic synchronization state machine mode, the word aligner starts looking for the word alignment pattern or synchronization code groups in the received data stream. When the programmed number of valid synchronization code groups or ordered sets is received, the `rx_syncstatus` signal is driven high to indicate that synchronization is acquired. The `rx_syncstatus` signal is constantly driven high until the programmed number of erroneous code groups is received without receiving intermediate good groups; after which the `rx_syncstatus` is driven low. The word aligner indicates loss of synchronization (`rx_syncstatus` remains low) until the programmed number of valid synchronization code groups are received again.

Manual Alignment Mode Word Aligner with 10-Bit PMA-PCS Interface Mode

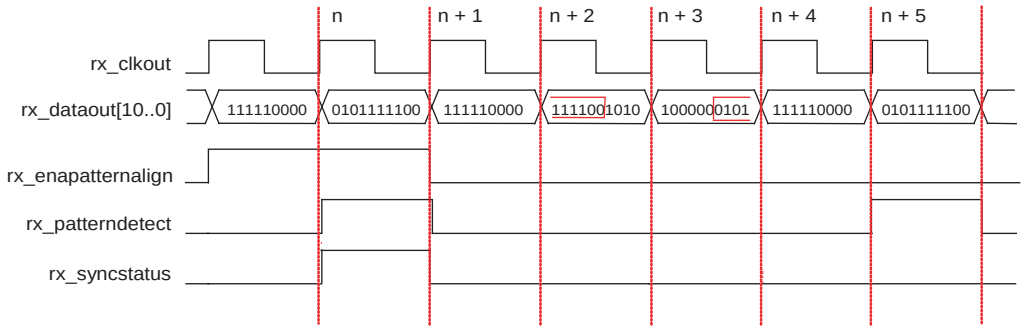
In Basic single-width mode with a 10-bit PMA-PCS interface, you can configure the word aligner in manual alignment mode by selecting the **Use manual word alignment mode** option in the ALTGX MegaWizard Plug-In Manager.

In manual alignment mode, the word aligner operation is controlled by the input signal `rx_enapatternalign`. The word aligner operation is level-sensitive to the `rx_enapatternalign` signal. If the `rx_enapatternalign` signal is held high, the word aligner looks for the programmed 7-bit or 10-bit word alignment pattern in the received data stream. It updates the word boundary if it finds the word alignment pattern in a new word boundary. If the `rx_enapatternalign` signal is de-asserted low, the word aligner maintains the current word boundary even when it sees the word alignment pattern in a new word boundary.

Two status signals, `rx_syncstatus` and `rx_patterndetect`, with the same latency as the datapath, are forwarded to the core fabric to indicate the word aligner status. After receiving the first word alignment pattern after the `rx_enapatternalign` signal is asserted high, both the `rx_syncstatus` and `rx_patterndetect` signals are driven high for one parallel clock cycle. Any word alignment pattern received thereafter in the same word boundary causes only the `rx_patterndetect` signal to go high for one clock cycle. Any word alignment pattern received thereafter in a different word boundary causes the word aligner to re-align to the new word boundary only if the `rx_enapatternalign` signal is held high. The word aligner asserts the `rx_syncstatus` signal for one parallel clock cycle whenever it re-aligns to the new word boundary.

Figure 1–62 shows the manual alignment mode word aligner operation with 10-bit PMA-PCS interface mode. In this example, a /K28.5/ (10'b0101111100) is specified as the word alignment pattern. The word aligner aligns to the /K28.5/ alignment pattern in cycle n because the `rx_enapatternalign` signal is asserted high. The `rx_syncstatus` signal goes high for one clock cycle, indicating alignment to a new word boundary. The `rx_patterndetect` signal also goes high for one clock cycle to indicate initial word alignment. At time n + 1, the `rx_enapatternalign` signal is de-asserted to instruct the word aligner to lock the current word boundary. The alignment pattern is detected again in a new word boundary across cycles n + 2 and n + 3. The word aligner does not align to this new word boundary because the `rx_enapatternalign` signal is held low. The /K28.5/ word alignment pattern is detected again in the current word boundary during cycle n + 5, causing the `rx_patterndetect` signal to go high for one parallel clock cycle.

Figure 1–62. Word Aligner with 10-Bit PMA-PCS Manual Alignment Mode



If the word alignment pattern is known to be unique and does not appear between word boundaries, you can constantly hold the rx_enapatternalign signal high because there is no possibility of false word alignment. If there is a possibility of the word alignment pattern occurring across word boundaries, you must control the rx_enapatternalign signal to lock the word boundary after the desired word alignment is achieved to avoid re-alignment to an incorrect word boundary.

Bit-Slip Mode Word Aligner with 10-Bit PMA-PCS Interface Mode

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

The word aligner operation for Basic single-width with a 10-bit PMA-PCS interface is similar to the word aligner operation in Basic single-width mode with 8-bit PMA-PCS interface. For word aligner operation in bit-slip mode, refer to "Manual Alignment Mode Word Aligner with 8-Bit PMA-PCS Interface Modes" on page 1–75. The only difference is that the bit-slip word aligner with 10-bit PMA-PCS interface modes allow 7-bit and 10-bit word alignment patterns, whereas 8-bit PMA-PCS interface modes allow only 16-bit word alignment patterns.

Word Aligner in Double-Width Mode

In double-width mode, the PMA-PCS interface is either 16 bit or 20 bit wide. In 16-bit PMA-PCS interface modes, the word aligner receives 16-bit wide data from the deserializer. In 20-bit PMA-PCS interface modes, the word aligner receives 10 bit wide data from the deserializer. Depending on the configured functional mode, you can configure the word aligner in manual alignment mode or bit-slip mode. The automatic synchronization state machine mode is not supported for word aligner in double-width mode.

Word Aligner in Double-Width Mode with 16-Bit PMA-PCS Interface Modes

The following functional modes support the 16-bit PMA-PCS interface:

- SONET/SDH OC-96
- (OIF) CEI PHY interface
- Basic double-width

Table 1–22 shows the word aligner configurations allowed in functional modes with 16-bit PMA-PCS interface.

Table 1–22. Word Aligner Configurations with a 16-Bit Wide PMA-PCS Interface *(Note 1)*

Functional Mode	Allowed Word Aligner Configurations	Allowed Word Alignment Pattern Length
SONET/SDH OC-96	Manual Alignment	16 bits, 32 bits
Basic double-width	Manual Alignment, Bit-Slip	8 bits, 16 bits, 32 bits

Note to Table 1–22:

(1) The word aligner is bypassed in (OIF) CEI PHY interface mode.

Manual Alignment Mode Word Aligner with 16-Bit PMA-PCS Interface Modes

In manual alignment mode, the word aligner starts looking for the programmed 8-bit, 16-bit, or 32-bit word alignment pattern in the received data stream as soon as `rx_digitalreset` is de-asserted low. It aligns to the first word alignment pattern received regardless of the logic level driven on the `rx_enapatternalign` signal. Any word alignment pattern received thereafter in a different word boundary does not cause the word aligner to re-align to this new word boundary. After the initial word alignment following de-assertion of the `rx_digitalreset` signal, if a word re-alignment is required, you must use the `rx_enapatternalign` signal.

Word aligner operation is controlled by the input signal `rx_enapatternalign` and is edge-sensitive to the `rx_enapatternalign` signal. A rising edge on the `rx_enapatternalign` signal triggers the word aligner to look for the word alignment pattern in the received data stream. The word aligner aligns the 16-bit word boundary to the first word alignment pattern received after the rising edge on the `rx_enapatternalign` signal. Any word alignment pattern received thereafter in a different word boundary does not cause the word aligner to re-align to this new word boundary. If another word re-alignment is required, you must de-assert and re-assert the `rx_enapatternalign` signal to create a rising edge on this signal.

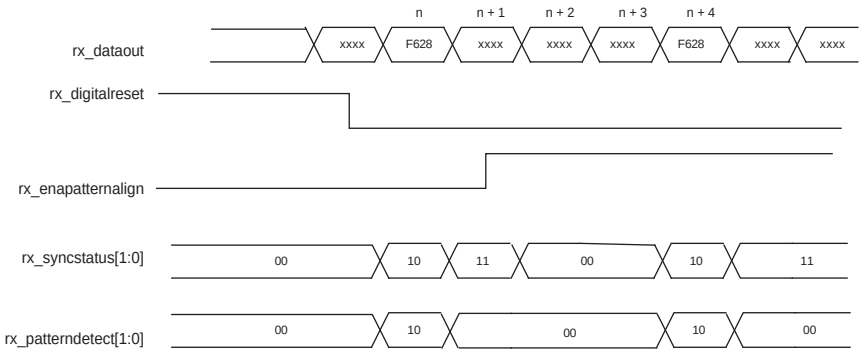
Two status signals, `rx_syncstatus` and `rx_patterndetect`, with the same latency as the datapath, are forwarded to the core fabric to indicate word aligner status.

After receiving the first word alignment pattern, the `rx_patterndetect` signal is driven high for one parallel clock cycle synchronous to the data that matches the MSByte of the word alignment pattern. Any word alignment pattern received thereafter in the same word boundary causes `rx_patterndetect` to go high for one parallel clock cycle.

After receiving the first word alignment pattern, the `rx_syncstatus` signal is constantly driven high until the word aligner sees another rising edge on the `rx_enapatternalign` signal. The rising edge on the `rx_enapatternalign` signal re-triggers the word alignment operation.

Figure 1–63 shows the manual alignment mode word aligner operation in 16-bit PMA-PCS interface mode. In this example, a 16'hF628 is specified as the word alignment pattern. The word aligner aligns to the 16'hF628 pattern received in cycle n after de-assertion of `rx_digitalreset`. The `rx_patterndetect[1]` signal is driven high for one parallel clock cycle. The `rx_syncstatus[1]` signal is driven high constantly until cycle n + 2, after which it is driven low because of the rising edge on the `rx_enapatternalign` signal that re-triggers the word aligner operation. The word aligner receives the word alignment pattern again in cycle n + 4, causing the `rx_patterndetect[1]` signal to be driven high for one parallel clock cycle and the `rx_syncstatus[1]` signal to be driven high constantly.

Figure 1–63. Manual Alignment Mode Word Aligner with 16-Bit PMA-PCS Interface Modes



Bit-Slip Mode Word Aligner with 16-Bit PMA-PCS Interface Modes

In some Basic double-width configurations with a 16-bit PMA-PCS interface, you can configure the word aligner in bit-slip mode by selecting the **Use manual bit slipping mode** option in the ALTGX MegaWizard Plug-In Manager.

The word aligner operation for Basic double-width with a 16-bit PMA-PCS interface is similar to the word aligner operation in Basic single-width mode with an 8-bit PMA-PCS interface. For word aligner operation in bit-slip mode, refer to "Word Aligner in Single-Width Mode with 8-Bit PMA-PCS Interface Modes" on page 1–74. The only difference is that the bit-slip word aligner in 16-bit PMA-PCS interface modes allows 8-bit and 16-bit word alignment patterns, whereas the bit-slip word aligner in 8-bit PMA-PCS interface modes allows only a 16-bit word alignment pattern.

Word Aligner in Double-Width Mode with 20-Bit PMA-PCS Interface Modes

A 20-bit PMA-PCS interface is supported only in Basic double-width mode.

Table 1–23 shows the word aligner configurations allowed in functional modes with a 20-bit PMA-PCS interface.

Table 1–23. Word Aligner in 20-Bit PMA-PCS Interface Modes

Functional Mode	Allowed Word Aligner Configurations	Allowed Word Alignment Pattern Length
Basic double-width	Manual Alignment, Bit-Slip	7 bits, 10 bits, 20 bits

Manual Alignment Mode Word Aligner with 20-Bit PMA-PCS Interface Modes

The word aligner operation in Basic double-width mode with a 20-bit PMA-PCS interface is similar to the word aligner operation in Basic double-width mode with a 16-bit PMA-PCS interface. For word aligner operation in manual alignment mode, refer to "Word Aligner in Double-Width Mode with 16-Bit PMA-PCS Interface Modes" on page 1–80. The only difference is that the manual alignment mode word aligner in 20-bit PMA-PCS interface modes allows 7-, 10-, and 20-bit word alignment patterns, whereas the manual alignment mode word aligner in 16-bit PMA-PCS interface modes allows only 8-, 16-, and 32-bit word alignment patterns.

Bit-Slip Mode Word Aligner with 20-Bit PMA-PCS Interface Modes

In some Basic single-width configurations with 20-bit PMA-PCS interfaces, you can configure the word aligner in bit-slip mode by selecting the **Use manual bit slipping mode** option in the ALTGX MegaWizard Plug-In Manager.

The word aligner operation for Basic double-width with a 20-bit PMA-PCS interface is similar to the word aligner operation in Basic single-width mode with an 8-bit PMA-PCS interface. For word aligner operation in bit-slip mode, refer to "Word Aligner in Single-Width Mode with 8-Bit PMA-PCS Interface Modes" on page 1–74. The only difference is that the bit-slip word aligner in 20-bit PMA-PCS interface modes allows only 7-, 10-, and 20-bit word alignment patterns, whereas the bit-slip word aligner in 8-bit PMA-PCS interface modes allows only a 16-bit word alignment pattern.

Table 1–24 summarizes the word aligner options available in Basic single-width and double-width modes.

Table 1–24. Word Aligner Options Available in Basic Single-Width and Double-Width Modes *(Note 1)* (Part 1 of 2)

Functional Mode	PMA-PCS Interface Width	Word Alignment Mode	Word Alignment Pattern Length	rx_enapatternalign Sensitivity	rx_syncstatus Behavior	rx_patterndetect Behavior
Basic Single-Width	8 bit	Manual Alignment	16 bit	Rising Edge Sensitive	Asserted high for one parallel clock cycle when the word aligner aligns to a new word boundary.	Asserted high for one parallel clock cycle when the word alignment pattern appears in the current word boundary.
		Bit-Slip	16 bit	—	—	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 1–24. Word Aligner Options Available in Basic Single-Width and Double-Width Modes *(Note 1)* (Part 2 of 2)

Functional Mode	PMA-PCS Interface Width	Word Alignment Mode	Word Alignment Pattern Length	rx_enapatternalign Sensitivity	rx_syncstatus Behavior	rx_patterndetect Behavior
Basic Double-width	16 bit	Manual Alignment	8, 16, and 32-bit	Rising Edge Sensitive	Stays high after the word aligner aligns to the word alignment pattern. Goes low on receiving a rising edge on `rx_enapatternalign` until a new word alignment pattern is received.	Asserted high for one parallel clock cycle when the word alignment pattern appears in the current word boundary.
		Bit Slip	8, 16, and 32-bit	—	—	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 1–24:

(1) For more information about word aligner operation, refer to "Word Aligner in Single-Width Mode" on page 1–74 and "Word Aligner in Double-Width Mode" on page 1–80.

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 core fabric clock might have a phase difference and/or PPM difference (in asynchronous systems) with respect to the recovered clock. To ensure that the core fabric clock can latch the rx_rlv signal reliably, the run length violation circuitry asserts the rx_rlv signal for a minimum of two recovered clock cycles in single-width modes and a minimum of three recovered clock cycles in double-width modes. The rx_rlv signal can be asserted longer, depending on the run length of the received data.

In single-width mode, the run length violation circuit detects up to a run length of 128 (for an 8-bit deserialization factor) or 160 (for a 10-bit deserialization factor). The settings are in increments of four or five for the 8-bit or 10-bit deserialization factors, respectively.

In double-width mode, the run length violation circuit maximum run length detection is 512 (with a run length increment of eight) and 640 (with a run length increment of 10) for the 16-bit and 20-bit deserialization factors, respectively.

Table 1–25 summarizes the detection capabilities of the run length violation circuit.

Table 1–25. Detection Capabilities of the Run Length Violation Circuit

Mode	PMA-PCS Interface Width	Run Length Violation Detector Range	
		Minimum	Maximum
Single-width mode	8-bit	4	128
	10-bit	5	160
Double-width mode	16-bit	8	512
	20-bit	10	640

Receiver Polarity Inversion

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

An optional rx_invpolarity port is available in all single-width and double-width modes except (OIF) CEI PHY and PCI Express (PIPE modes) to dynamically enable the receiver polarity inversion feature. In single-width modes, a high value on the rx_invpolarity port inverts the polarity of every bit of the 8-bit or 10-bit input data word to the word aligner in the receiver datapath. In double-width modes, a high value on the rx_invpolarity port inverts the polarity of every bit of the 16-bit or 20-bit input data word to the word aligner in the receiver datapath. Because inverting the polarity of each bit has the same effect as swapping the positive and negative signals of the differential link, correct data is seen by the receiver. rx_invpolarity is a dynamic signal and can cause initial disparity errors in an 8B/10B encoded link. The downstream system must be able to tolerate these disparity errors.

The generic receiver polarity inversion feature is different from the PCI Express (PIPE) 8B/10B polarity inversion feature. The generic receiver polarity inversion feature inverts the polarity of the data bits at the input of the word aligner and is not available in PIPE mode. The PIPE 8B/10B polarity inversion feature inverts the polarity of the data bits at the input of the 8B/10B decoder and is available only in PIPE mode.

Figure 1–64 shows the receiver polarity inversion feature in single-width 10 bit wide datapath configurations.

Figure 1–64. Receiver Polarity Inversion in Single-Width Mode

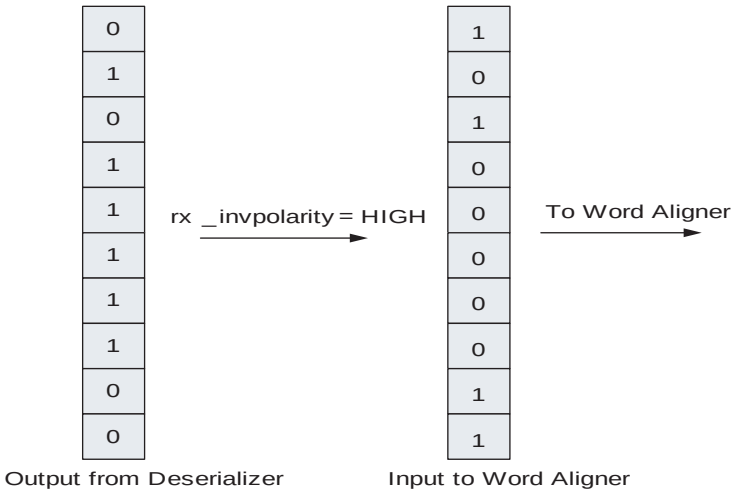
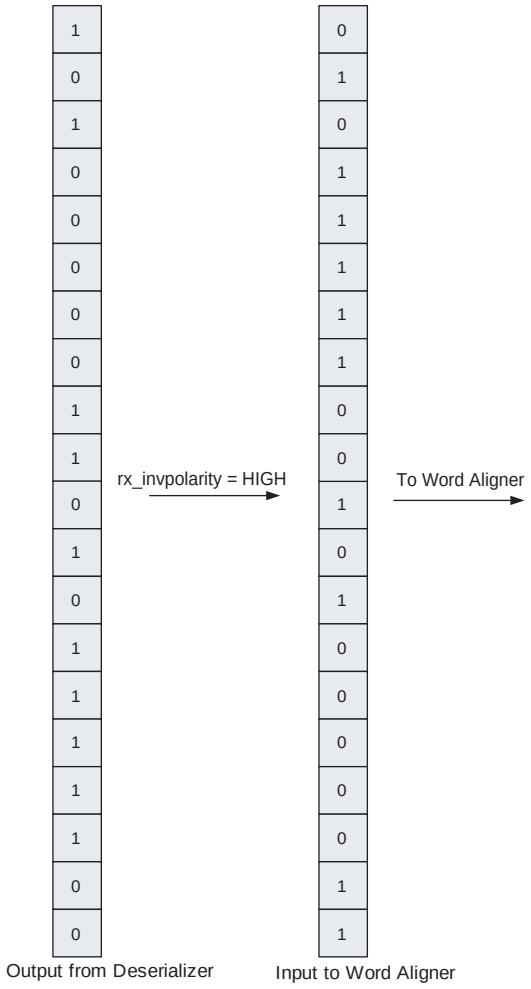


Figure 1–65 shows the receiver polarity inversion feature in double-width 20 bit wide datapath configurations.

Figure 1–65. Receiver Polarity Inversion in Double-Width Mode



Receiver Bit Reversal

By default, the HardCopy IV GX receiver assumes a LSBit-to-MSBit transmission. If the transmission order is MSBit-to-LSBit, the receiver forwards the bit-flipped version of the parallel data to the core fabric on the `rx_dataout` port. The receiver bit reversal feature is available to correct this situation.

The receiver bit reversal feature is available through the `rx_revbitordwa` port only in Basic single-width and double-width modes with the word aligner configured in bit-slip mode.

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

When the `rx_revbitordwa` signal is driven high in Basic double-width mode, the 16-bit or 20-bit data `D[15:0]` or `D[19:0]` at the output of the word aligner gets rewired to `D[0:15]` or `D[0:19]`, respectively.

Flipping the parallel data using this feature allows the receiver to forward the correct bit-ordered data to the core fabric on the `rx_dataout` port in the case of MSBit-to-LSBit transmission.

Figure 1–66 shows the receiver bit reversal feature in Basic single-width 10 bit wide datapath configurations.

Figure 1–66. Receiver Bit Reversal in Single-Width Mode

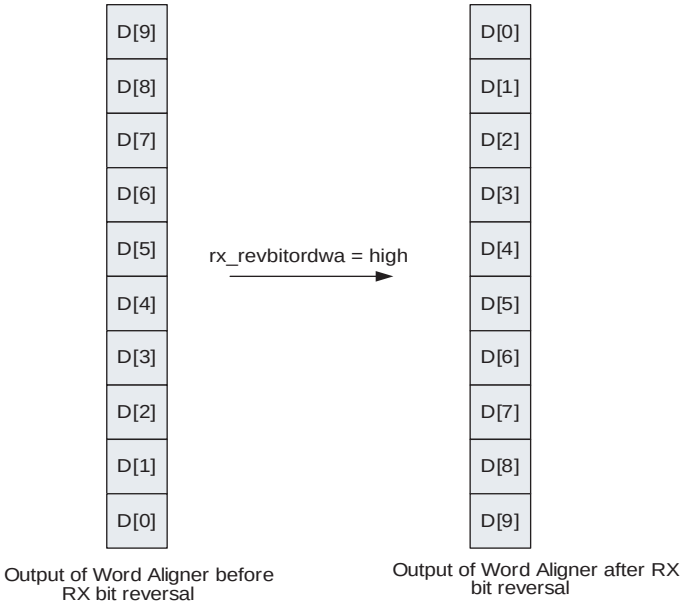
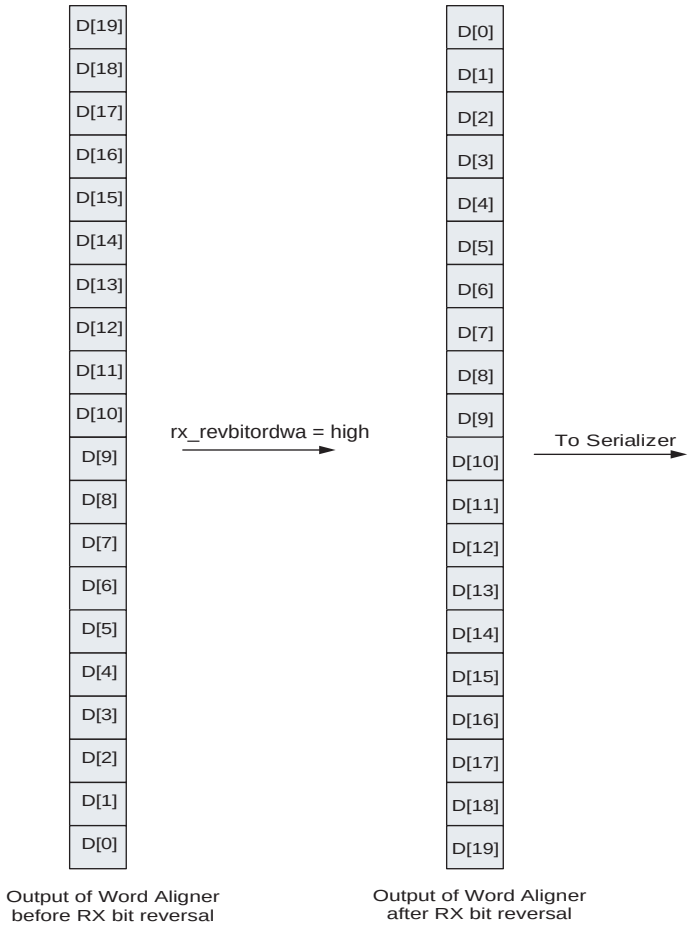


Figure 1–67 shows the receiver bit reversal feature in Basic double-width 20-bit wide datapath configurations.

Figure 1–67. Receiver Bit Reversal in Double-Width Mode



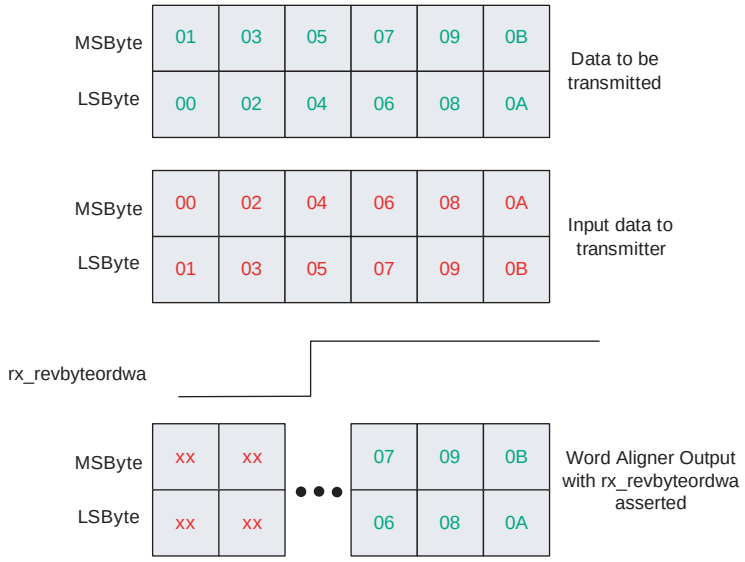
Receiver Byte Reversal in Basic Double-Width Modes

The MSByte and LSByte of the input data to the transmitter may be erroneously swapped. The receiver byte reversal feature is available to correct this situation.

An optional port, `rx_revbyteordwa`, is available only in Basic double-width mode to enable receiver byte reversal. In 8B/10B enabled mode, a high value on `rx_revbyteordwa` exchanges the 10-bit MSByte for the LSByte of the 20-bit word at the output of the word aligner in the receiver datapath. In non-8B/10B enabled mode, a high value on `rx_revbyteordwa` exchanges the 8-bit MSByte for the LSByte of the 16-bit word at the output of the word aligner in the receiver datapath. This compensates for the erroneous exchanging at the transmitter and corrects the data received by the downstream systems. `rx_revbyteorderwa` is a dynamic signal and can cause an initial disparity error at the receiver of an 8B/10B encoded link. The downstream system must be able to tolerate this disparity error.

Figure 1–68 shows the receiver byte reversal feature.

Figure 1–68. Receiver Byte Reversal Feature



Deskew FIFO

Code groups received across four lanes in a XAUI link can be misaligned with respect to one another because of skew in the physical medium or differences between the independent clock recoveries per lane. The XAUI protocol allows a maximum skew of 40 UI (12.8 ns) as seen at the receiver of the four lanes.

XAUI protocol requires the physical layer device to implement a deskew circuitry to align all four channels. To enable the deskew circuitry at the receiver to align the four channels, the transmitter sends a /A/ (/K28.3/) code group simultaneously on all four channels during inter-packet gap (IPG). The skew introduced in the physical medium and the receiver channels can cause the /A/ code groups to be received misaligned.

Deskew circuitry performs the deskew operation in XAUI functional mode. Deskew circuitry consists of:

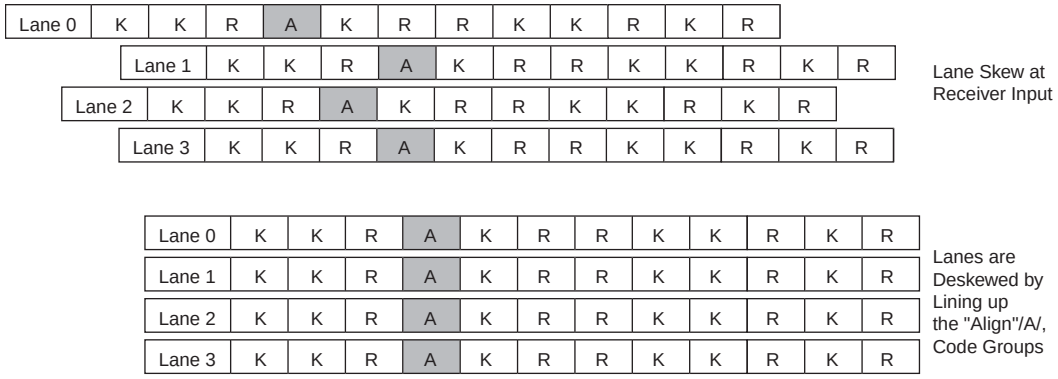
- A 16-word deep deskew FIFO in each of the four channels
- Control logic in the `CMU0` channel of the transceiver block that controls the deskew FIFO write and read operations in each channel

Deskew circuitry is only available in XAUI mode.

The deskew FIFO in each channel receives data from its word aligner. The deskew operation begins only after link synchronization is achieved on all four channels as indicated by a high level on the rx_syncstatus signal from the word aligner in each channel. Until the first /A/ code group is received, the deskew FIFO read and write pointers in each channel are not incremented. After the first /A/ code group is received, the write pointer starts incrementing for each word received but the read pointer is frozen. If the /A/ code group is received on each of the four channels within 10 recovered clock cycles of each other, the read pointer for all four deskew FIFOs is released simultaneously, aligning all four channels.

Figure 1–69 shows lane skew at the receiver input and how the deskew FIFO uses the /A/ code group to align the channels.

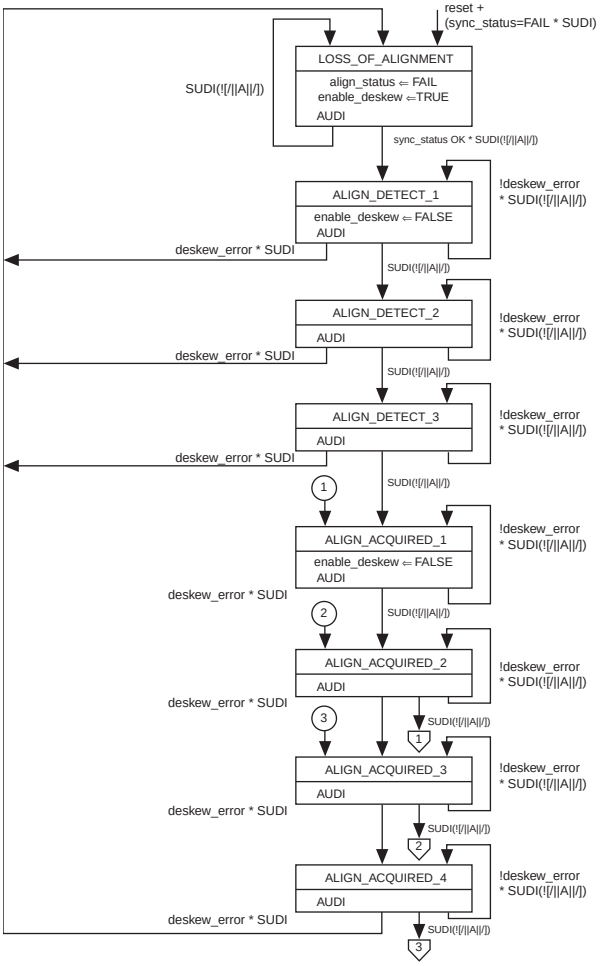
Figure 1–69. Deskew FIFO—Lane Skew at the Receiver Input



After alignment of the first ||A|| column, if three additional aligned ||A|| columns are observed at the output of the deskew FIFOs of the four channels, the rx_channelaligned signal is asserted high, indicating channel alignment is acquired. After acquiring channel alignment, if four misaligned ||A|| columns are seen at the output of the deskew FIFOs in all four channels with no aligned ||A|| columns in between, the rx_channelaligned signal is de-asserted low, indicating loss of channel alignment.

The deskew operation in XAUI functional mode is compliant to the PCS deskew state machine diagram specified in clause 48 of IEEE P802.3ae, as shown in Figure 1–70.

Figure 1–70. Deskew FIFO Operation in XAUI Functional Mode *(Note 1)*



Note to Figure 1–70:

(1) This figure is from IEEE P802.3ae.

Rate Match (Clock Rate Compensation) FIFO

In asynchronous systems, the upstream transmitter and local receiver can be clocked with independent reference clocks. Frequency differences in the order of a few hundred PPM can corrupt the data when latching from the recovered clock domain (the same clock domain as the upstream transmitter reference clock) to the local receiver reference clock domain.

The rate match FIFO compensates for small clock frequency differences between the upstream transmitter and the local receiver clocks by inserting or removing SKP symbols or ordered sets from the IPG or idle streams. It deletes SKP symbols or ordered sets when the upstream transmitter reference clock frequency is higher than the local receiver reference clock frequency. It inserts SKP symbols or ordered-sets when the local receiver reference clock frequency is higher than the upstream transmitter reference clock frequency.

The rate match FIFO consists of a 20-word deep FIFO and necessary logic that controls insertion and deletion of a skip character or ordered set, depending on the PPM difference.

The rate match FIFO is mandatory and cannot be bypassed in the following functional modes:

- PCI Express (PIPE)
- XAUI
- GIGE

The rate match FIFO is optional in the following functional modes:

- Basic single-width
- Basic double-width

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 core fabric:

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

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

Rate Match FIFO in PCI Express (PIPE) Mode

In PIPE mode, the rate match FIFO is capable of compensating up to ± 300 PPM (total 600 PPM) difference between the upstream transmitter and the local receiver. The PIPE protocol requires the transmitter to send SKP ordered sets during IPGs, adhering to rules listed in the base specification. The SKP ordered set is defined as a /K28.5/ COM symbol followed by three consecutive /K28.0/ SKP symbols groups. The PIPE protocol requires the receiver to recognize a SKP ordered set as a /K28.5/ COM symbol followed by one to five consecutive /K28.0/ SKP symbols.

The rate match FIFO operation is compliant to PIPE Base Specification 2.0. The rate match operation begins after the synchronization state machine in the word aligner indicates synchronization is acquired by driving the `rx_syncstatus` signal high. The rate match FIFO looks for the SKP ordered set and deletes or inserts SKP symbols as necessary to prevent the rate match FIFO from overflowing or under-running.

The rate match FIFO inserts or deletes only one SKP symbol per SKP ordered set received. The rate match FIFO insertion and deletion events are communicated to the core fabric on the `pipestatus[2:0]` port from each channel. The `pipestatus[2:0]` signal is driven to 3'b001 for one clock cycle synchronous to the /K28.5/ COM symbol of the SKP ordered set in which the /K28.0/ SKP symbol is inserted. The `pipestatus[2:0]` signal is driven to 3'b010 for one clock cycle synchronous to the /K28.5/ COM symbol of the SKP ordered set from which the /K28.0/ SKP symbol is deleted.

Figure 1–71 shows an example of rate match deletion in the case where two /K28.0/ SKP symbols are required to be deleted. Only one /K28.0/ SKP symbol is deleted per SKP ordered set received.

Figure 1–71. Rate Match Deletion in PCI Express (PIPE) Mode

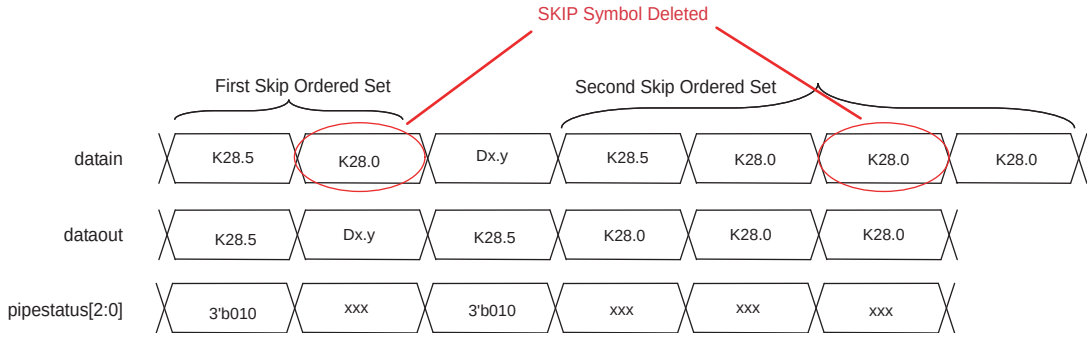
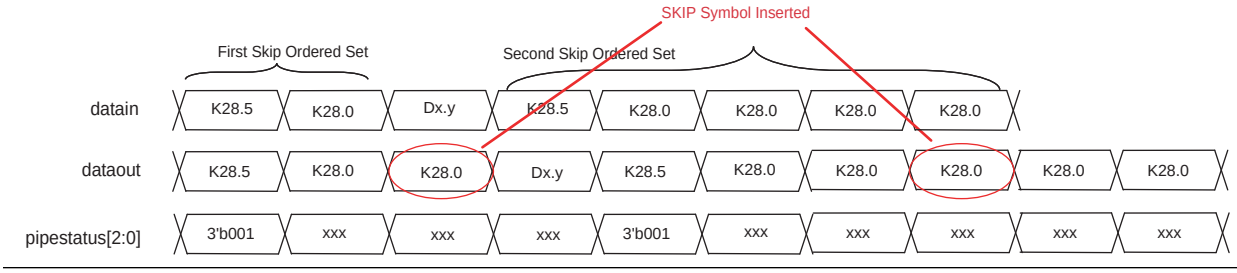


Figure 1–72 shows an example of rate match insertion in the case where two SKP symbols are required to be inserted. Only one /K28.0/ SKP symbol is inserted per SKP ordered set received.

Figure 1–72. Rate Match Insertion in PCI Express (PIPE) Mode

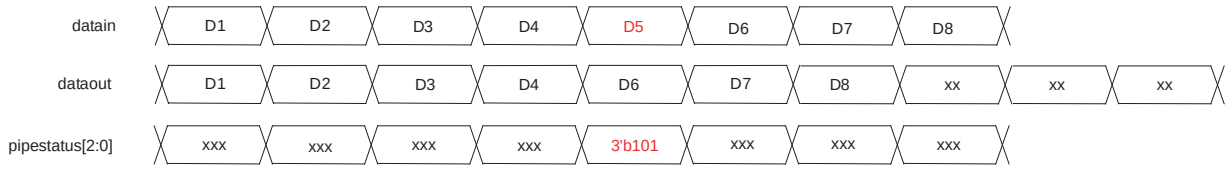


The rate match FIFO full and empty conditions are communicated to the core fabric on the `pipestatus[2:0]` port from each channel.

The rate match FIFO in PIPE mode automatically deletes the data byte that causes the FIFO to go full and drives `pipestatus[2:0] = 3'b101` synchronous to the subsequent data byte.

Figure 1–73 shows the rate match FIFO full condition in PIPE mode. The rate match FIFO becomes full after receiving data byte D4.

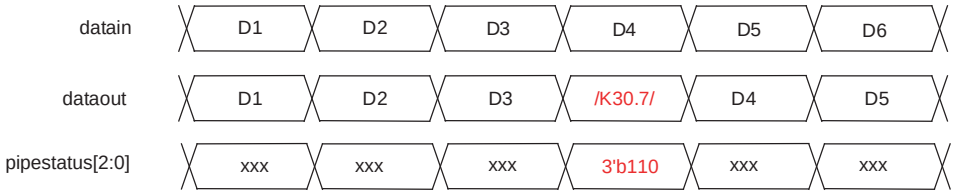
Figure 1–73. Rate Match FIFO Full Condition in PCI Express (PIPE) Mode



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

Figure 1–74 shows rate match FIFO empty condition in PCI Express (PIPE) mode. The rate match FIFO becomes empty after reading out data byte D3.

Figure 1–74. Rate Match FIFO Empty Condition in PCI Express (PIPE) Mode



You can configure the rate match FIFO in low latency mode by turning off the **Enable Rate Match FIFO** option in the ALTGX MegaWizard Plug-In Manager.

Rate Match FIFO in XAUI Mode

In XAUI mode, the rate match FIFO is capable of compensating for up to ±100 PPM (200 PPM total) difference between the upstream transmitter and the local receiver reference clock. The XAUI protocol requires the transmitter to send /R/ (/K28.0/) code groups simultaneously on all four lanes (denoted as ||R|| column) during inter-packet gaps, adhering to rules listed in the IEEE P802.3ae specification. The rate match FIFO operation in XAUI mode is compliant to the IEEE P802.3ae specification.

The rate match operation begins after:

- The synchronization state machine in the word aligner of all four channels indicates synchronization was acquired by driving its `rx_syncstatus` signal high
- The deskew FIFO block indicates alignment was acquired by driving the `rx_channelaligned` signal high

The rate match FIFO looks for the ||R|| column (simultaneous /R/ code group on all four channels) and deletes or inserts the ||R|| column to prevent the rate match FIFO from overflowing or under-running. It can insert or delete as many ||R|| columns as necessary to perform the rate match operation.

Two flags, `rx_rmfifodatadeleted` and `rx_rmfifodatainserted`, that indicate rate match FIFO deletion and insertion events, respectively, are forwarded to the core fabric. If an ||R|| column is deleted, the `rx_rmfifodeleted` flag from each of the four channels goes high for one clock cycle per deleted ||R|| column. If an ||R|| column is inserted, the `rx_rmfifoinserted` flag from each of the four channels goes high for one clock cycle per inserted ||R|| column.

Figure 1–75 shows an example of rate match deletion in the case where three ||R|| columns must be deleted.

Figure 1–75. Rate Match Deletion in XAUI Mode

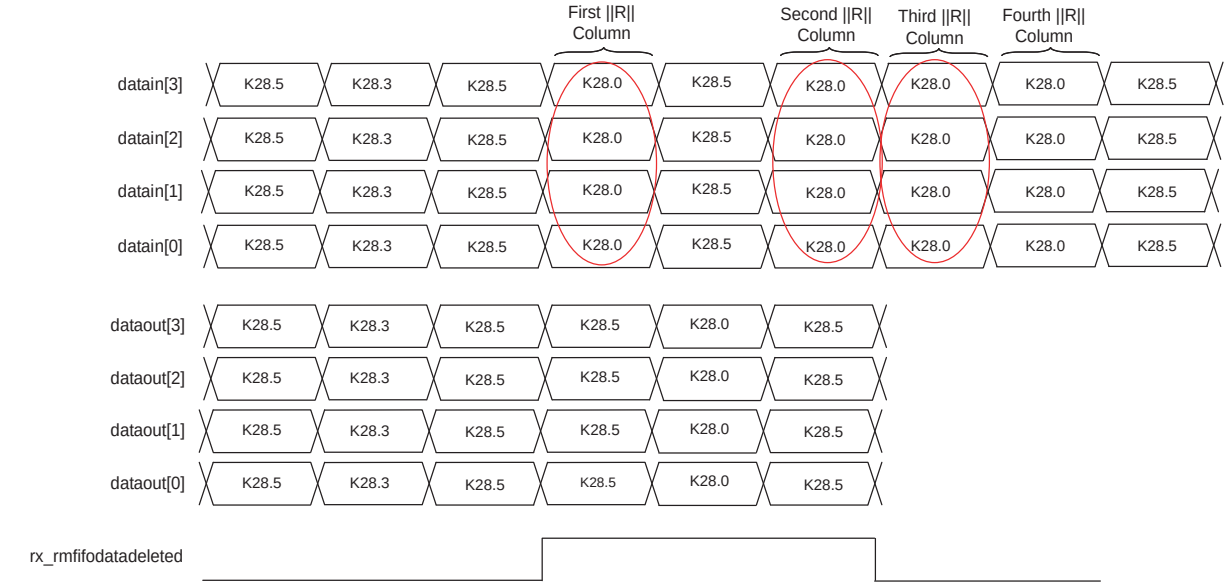
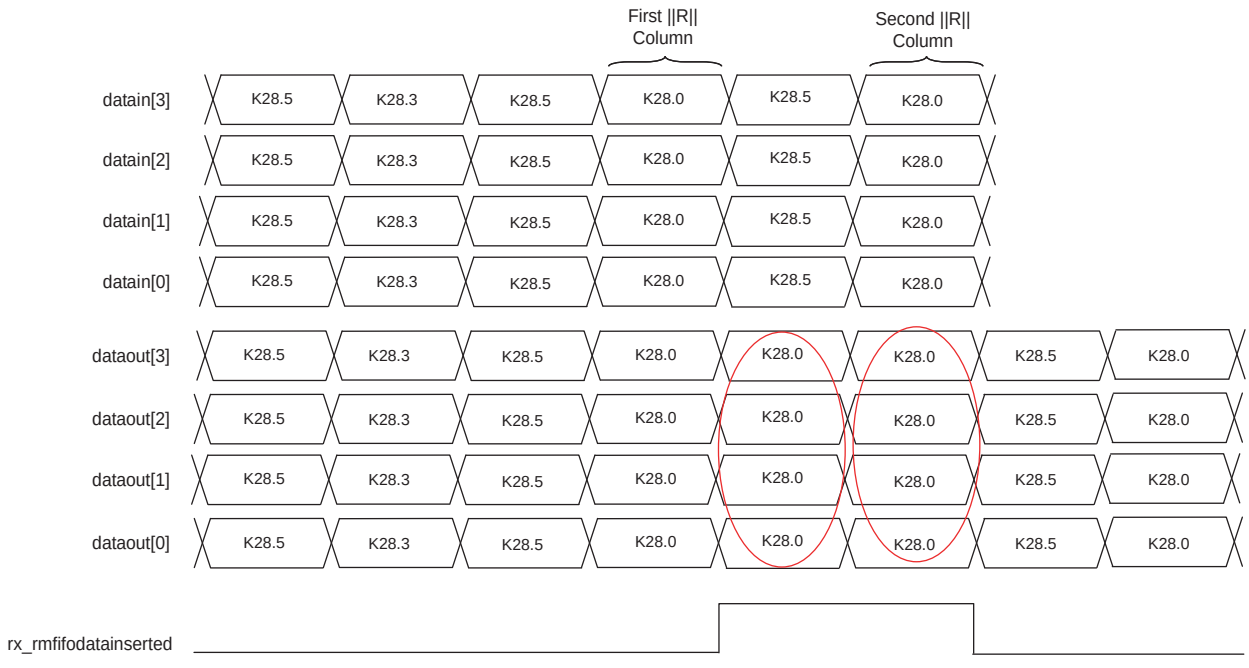


Figure 1–76 shows an example of rate match insertion in the case where two ||R|| columns are required to be inserted.

Figure 1–76. Rate Match Insertion in XAUI Mode



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

In XAUI mode, the rate match FIFO does not automatically insert or delete code groups to overcome FIFO empty and full conditions, respectively. It asserts the `rx_rmfifofull` and `rx_rmfifoempty` flags for at least three recovered clock cycles to indicate rate match FIFO full and empty conditions, respectively.

In the case of rate match FIFO full and empty conditions, you must assert the `rx_digitalreset` signal to reset the receiver PCS blocks.

Rate Match FIFO in GIGE Mode

In GIGE mode, the rate match FIFO is capable of compensating for up to ±100 PPM (200 PPM total) difference between the upstream transmitter and the local receiver reference clock. The GIGE protocol requires the transmitter to send idle ordered sets /I1/ (/K28.5/D5.6/) and /I2/ (/K28.5/D16.2/) during inter-packet gaps, adhering to rules listed in the IEEE 802.3 specification.

The rate match operation begins after the synchronization state machine in the word aligner indicates synchronization is acquired by driving the `rx_syncstatus` signal high. The rate match FIFO is capable of deleting or inserting the /I2/ (/K28.5/D16.2/) ordered set to prevent the rate match FIFO from overflowing or under-running during normal packet transmission. The rate match FIFO is also capable of deleting or inserting the first two bytes of the /C2/ ordered set (/K28.5/D2.2/Dx.y/Dx.y/) to prevent the rate match FIFO from overflowing or under-running during the auto negotiation phase.

The rate match FIFO can insert or delete as many /I2/ or /C2/ (first two bytes) as necessary to perform the rate match operation.

Two flags, `rx_rmfifodatadeleted` and `rx_rmfifodatainserted`, that indicate rate match FIFO deletion and insertion events, respectively, are forwarded to the core fabric. Both the `rx_rmfifodatadeleted` and `rx_rmfifodatainserted` flags are asserted for two clock cycles for each deleted and inserted /I2/ ordered set, respectively.

Figure 1–77 shows an example of rate match FIFO deletion in the case where three symbols are required to be deleted. Because the rate match FIFO can only delete /I2/ ordered set, it deletes two /I2/ ordered sets (four symbols deleted).

Figure 1–77. Rate Match Deletion in GIGE Mode

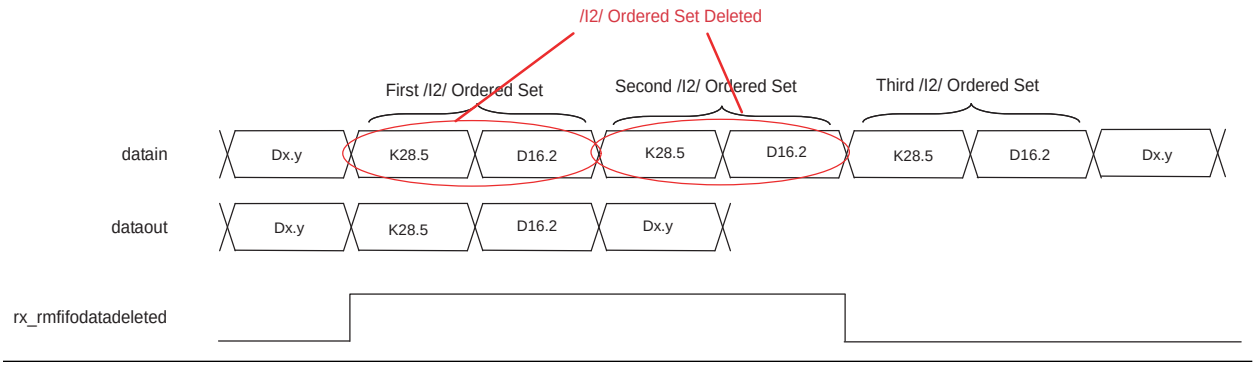
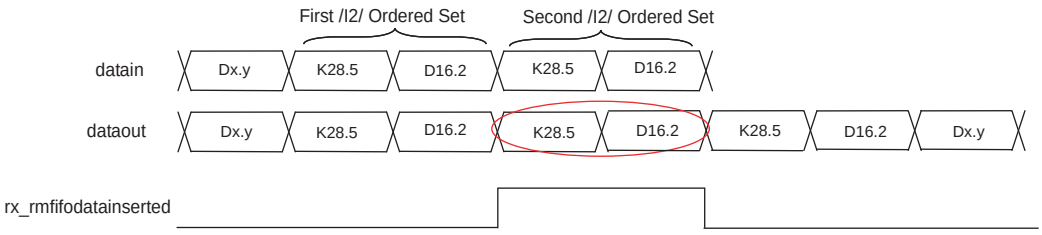


Figure 1–78 shows an example of rate match FIFO insertion in the case where one symbol is required to be inserted. Because the rate match FIFO can only delete /I2/ ordered set, it inserts one /I2/ ordered sets (two symbols inserted).

Figure 1–78. Rate Match Insertion in GIGE Mode



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

In GIGE mode, the rate match FIFO does not insert or delete code groups automatically to overcome FIFO empty and full conditions, respectively. It asserts the `rx_rmfifofull` and `rx_rmfifoempty` flags for at least two recovered clock cycles to indicate rate match FIFO full and empty conditions, respectively.

In the case of rate match FIFO full and empty conditions, you must assert the `rx_digitalreset` signal to reset the receiver PCS blocks.

Rate Match FIFO in Basic Single-Width Mode

In Basic single-width mode, the rate match FIFO is capable of compensating for up to ±300 PPM (600 PPM total) difference between the upstream transmitter and the local receiver reference clock.

To enable the rate match FIFO in Basic single-width mode, the transceiver channel must have both the transmitter and the receiver channel instantiated. You must select the **Receiver and Transmitter** option in the **What is the operation mode?** field in the ALTGX MegaWizard Plug-In Manager. You must also enable the 8B/10B encoder/decoder in Basic single-width mode with rate match FIFO enabled.

Depending on your proprietary protocol implementation, you can select two 20-bit rate match patterns in the ALTGX MegaWizard Plug-In Manager under the **What is the rate match pattern1** and **What is the rate match pattern2** fields. Each of the two programmed 20-bit rate match patterns consists of a 10-bit skip pattern and a 10-bit control pattern. You must choose 10-bit code groups that have a neutral disparity as the skip patterns. The rate match FIFO operation begins after the word aligner synchronization status `rx_syncstatus` goes high. When the rate matcher receives either of the two 10-bit control patterns followed by the respective 10-bit skip pattern, it inserts or deletes the 10-bit skip pattern as necessary to avoid the rate match FIFO from overflowing or under-running.

The rate match FIFO can delete a maximum of four skip patterns from a cluster, if there is one skip pattern left in the cluster after deletion. The rate match FIFO can insert a maximum of four skip patterns in a cluster, if there are no more than five skip patterns in the cluster after insertion. Two flags, `rx_rmfifodatadeleted` and `rx_rmfifodatainserted`, indicating rate match FIFO deletion and insertion events, respectively, are forwarded to the core fabric.

Figure 1–79 shows an example of rate match FIFO deletion in the case where three skip patterns are required to be deleted. In this example, /K28.5/ is the control pattern and neutral disparity /K28.0/ is the skip pattern. The first skip cluster has a /K28.5/ control pattern followed by two /K28.0/ skip patterns. The second skip cluster has a /K28.5/ control pattern followed by four /K28.0/ skip patterns. The rate match FIFO deletes only one /K28.0/ skip pattern from the first skip cluster to maintain at least one skip pattern in the cluster after deletion. Two /K28.0/ skip patterns are deleted from the second cluster for a total of three skip patterns deletion requirement.

Figure 1–79. Rate Match Deletion in Basic Single-Width Mode

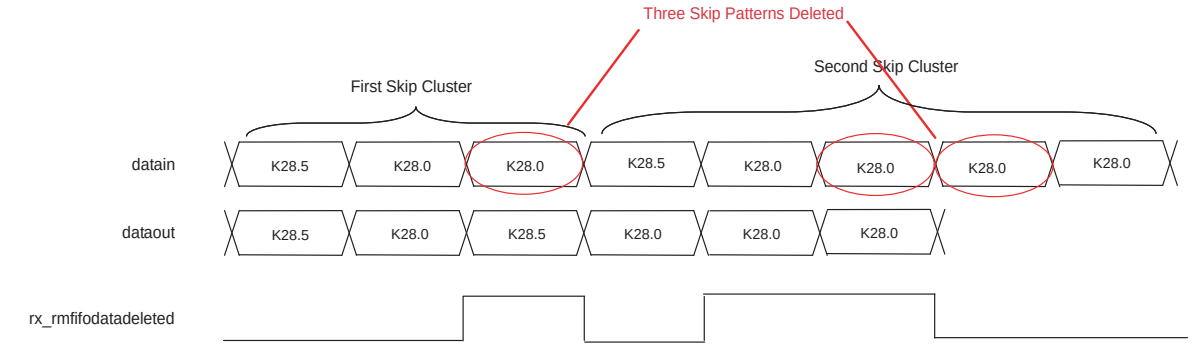
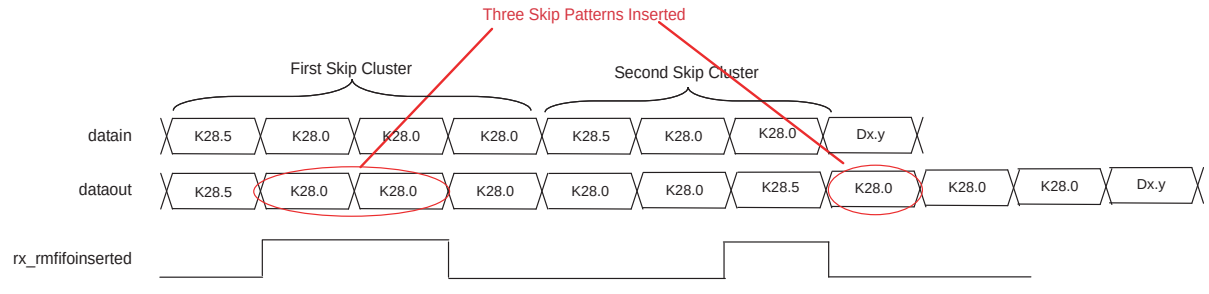


Figure 1–80 shows an example of rate match FIFO insertion in the case where three skip patterns are required to be inserted. In this example, /K28.5/ is the control pattern and neutral disparity /K28.0/ is the skip pattern. The first skip cluster has a /K28.5/ control pattern followed by three /K28.0/ skip patterns. The second skip cluster has a /K28.5/ control pattern followed by one /K28.0/ skip pattern. The rate match FIFO inserts only two /K28.0/ skip patterns into the first skip cluster to maintain a maximum of five skip patterns in the cluster after insertion. One /K28.0/ skip pattern is inserted into the second cluster for a total of three skip patterns to meet the insertion requirement.

Figure 1–80. Rate Match Insertion in Basic Single-Width Mode

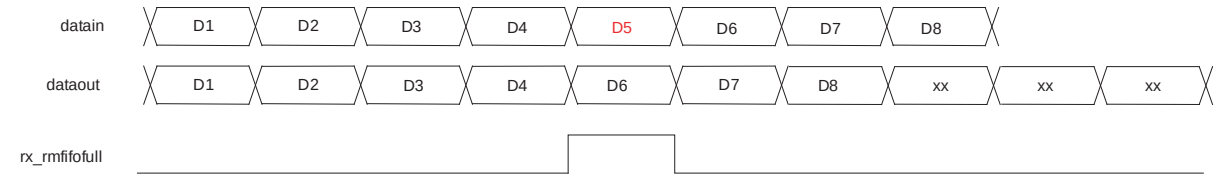


Two flags, `rx_rmfifofull` and `rx_rmfifoempty`, are forwarded to the core fabric to indicate rate match FIFO full and empty conditions.

The rate match FIFO in Basic single-width mode automatically deletes the data byte that causes the FIFO to go full and asserts the `rx_rmfifofull` flag synchronous to the subsequent data byte.

Figure 1–81 shows the rate match FIFO full condition in Basic single-width mode. The rate match FIFO becomes full after receiving data byte D4.

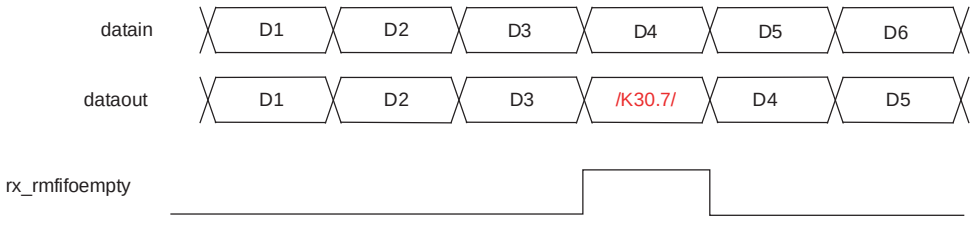
Figure 1–81. Rate Match FIFO Full Condition in Basic Single-Width Mode



The rate match FIFO automatically inserts /K30.7/ (9'h1FE) after the data byte that causes the FIFO to go empty and asserts the `rx_fifoempty` flag synchronous to the inserted /K30.7/ (9'h1FE).

Figure 1–82 shows the rate match FIFO empty condition in Basic single-width mode. The rate match FIFO becomes empty after reading out data byte D3.

Figure 1–82. Rate Match FIFO Empty Condition in Basic Single-Width Mode



Rate Match FIFO in Basic Double-Width Mode

In Basic double-width mode, the rate match FIFO is capable of compensating up to ±300 PPM (total 600 PPM total) difference between the upstream transmitter and the local receiver reference clock.

To enable the rate match FIFO in Basic double-width mode, the transceiver channel must have both the transmitter and the receiver channel instantiated. You must select the **Receiver and Transmitter** option in the **What is the operation mode?** field in the ALTGX MegaWizard Plug-In Manager. You must also enable the 8B/10B encoder/decoder in Basic double-width mode with rate match FIFO enabled.

Depending on your proprietary protocol implementation, you can select two 20-bit rate match patterns in the ALTGX MegaWizard Plug-In Manager under the **What is the rate match pattern1** and **What is the rate match pattern2** fields. Each of the two programmed 20-bit rate match patterns consists of a 10-bit skip pattern and a 10-bit control pattern. You must choose 10-bit code groups that have a neutral disparity as the skip patterns. The rate match FIFO operation begins after the word aligner synchronization status `rx_syncstatus` goes high. When the rate matcher receives either of the two 10-bit control patterns followed by the respective 10-bit skip pattern, it inserts or deletes a pair of 10-bit skip patterns as necessary to avoid the rate match FIFO from overflowing or under-running.

The rate match FIFO can delete as many pairs of skip patterns from a cluster necessary to avoid the rate match FIFO from overflowing. The rate match FIFO can delete a pair of skip patterns only if the two 10-bit skip patterns appear in the same clock cycle on the LSByte and MSByte of the 20-bit word. If the two skip patterns appear straddled on the MSByte of a clock cycle and the LSByte of the next clock cycle, the rate match FIFO cannot delete the pair of skip patterns. The rate match FIFO can insert as many pairs of skip patterns into a cluster necessary to avoid the rate match FIFO from under-running. The 10-bit skip pattern can appear on MSByte or LSByte, or both, of the 20-bit word.

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

Figure 1–83 shows an example of rate match FIFO deletion in the case where three skip patterns are required to be deleted. In this example, /K28.5/ is the control pattern and neutral disparity /K28.0/ is the skip pattern. The first skip cluster has a /K28.5/ control pattern in the LSByte and /K28.0/ skip pattern in the MSByte of a clock cycle followed by one /K28.0/ skip pattern in the LSByte of the next clock cycle. The rate match FIFO cannot delete the two skip patterns in this skip cluster because they do not appear in the same clock cycle. The second skip cluster has a /K28.5/ control pattern in the MSByte of a clock cycle followed by two pairs of /K28.0/ skip patterns in the next two cycles. The rate match FIFO deletes both pairs of /K28.0/ skip patterns (for a total of four skip patterns deleted) from the second skip cluster to meet the three skip pattern deletion requirement.

Figure 1–83. Rate Match Deletion in Basic Double-Width Mode

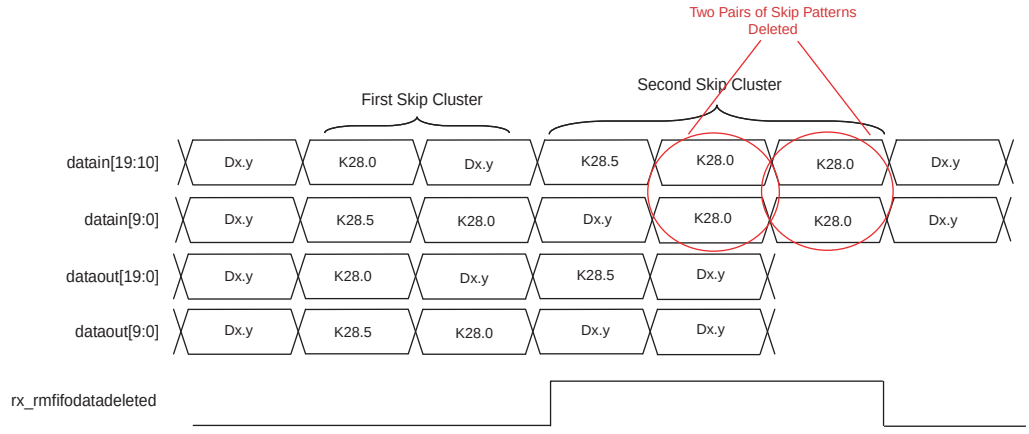
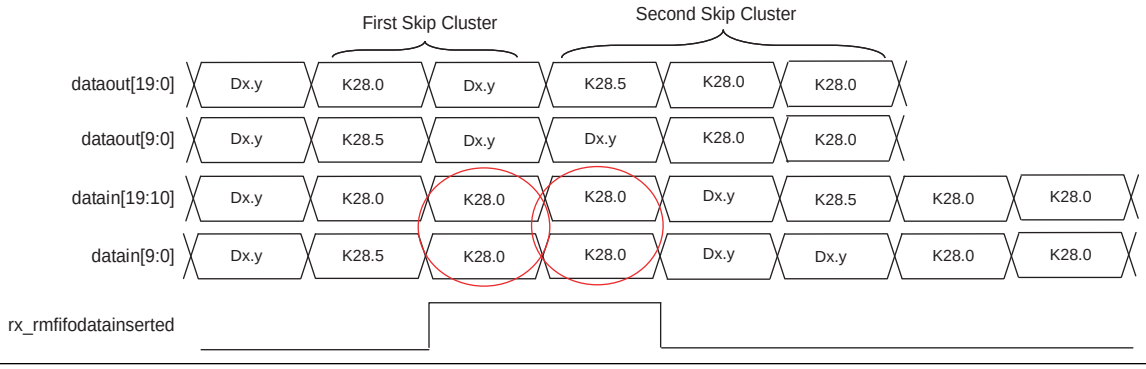


Figure 1–84 shows an example of rate match FIFO insertion in the case where three skip patterns are required to be inserted. In this example, /K28.5/ is the control pattern and neutral disparity /K28.0/ is the skip pattern. The first skip cluster has a /K28.5/ control pattern in the LSByte and /K28.0/ skip pattern in the MSByte of a clock cycle followed by one /K28.0/ skip pattern in the LSByte of the next clock cycle. The rate match FIFO inserts pairs of skip patterns in this skip cluster to meet the three skip pattern insertion requirement.

Figure 1–84. Rate Match Insertion in Basic Double-Width Mode

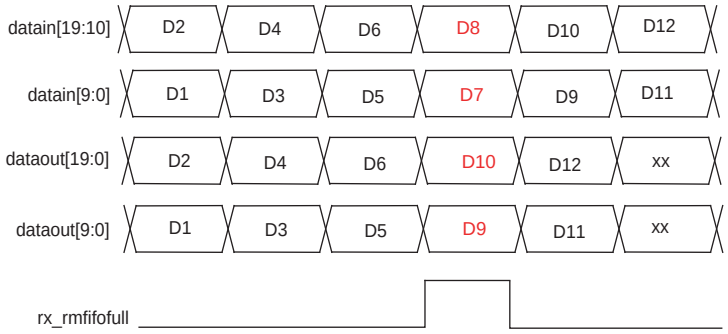


Two flags, `rx_rmfifofull` and `rx_rmfifoempty`, are forwarded to the core fabric to indicate rate match FIFO full and empty conditions.

The rate match FIFO in Basic double-width mode automatically deletes the pair of data byte that causes the FIFO to go full and asserts the `rx_rmfifofull` flag synchronous to the subsequent pair of data bytes.

Figure 1–85 shows the rate match FIFO full condition in Basic double-width mode. The rate match FIFO becomes full after receiving the 20-bit word D5D6.

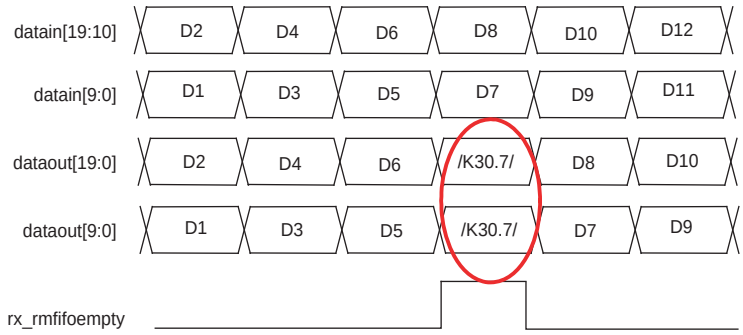
Figure 1–85. Rate Match FIFO Full Condition in Basic Double-Width Mode



The rate match FIFO automatically inserts a pair of /K30.7/ ({9'h1FE,9'h1FE}) after the data byte that causes the FIFO to go empty and asserts the `rx_fifoempty` flag synchronous to the inserted pair of /K30.7/ ({9'h1FE,9'h1FE}).

Figure 1–86 shows the rate match FIFO empty condition in Basic double-width mode. The rate match FIFO becomes empty after reading out the 20-bit word D5D6.

Figure 1–86. Rate Match FIFO Empty Condition in Basic Double-Width Mode



8B/10B Decoder

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

The HardCopy IV GX receiver channel PCS datapath implements 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.

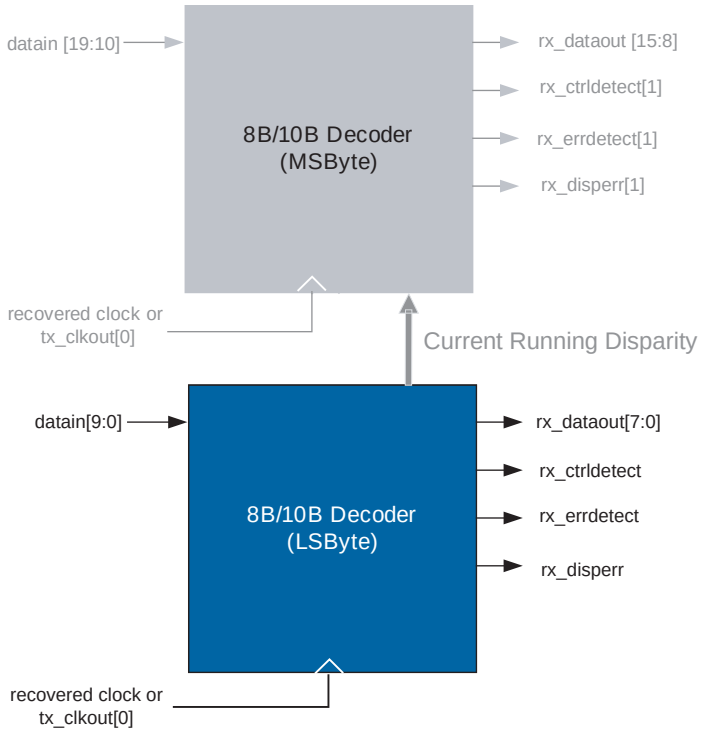
The 8B/10B decoder operates in two modes:

- Single-width mode
- Double-width mode

8B/10B Decoder in Single-Width Mode

Figure 1–87 shows the 8B/10B decoder in single-width mode.

Figure 1–87. 8B/10B Decoder in Single-Width Mode



In single-width mode, the 8B/10B decoder receives 10-bit data from the rate matcher or word aligner (when rate matcher is disabled) and decodes it into an 8-bit data + 1-bit control identifier. The decoded data is fed to the byte deserializer or the receiver phase compensation FIFO (if byte deserializer is disabled).

The 8B/10B decoder is compliant to Clause 36 in the IEEE802.3 specification.

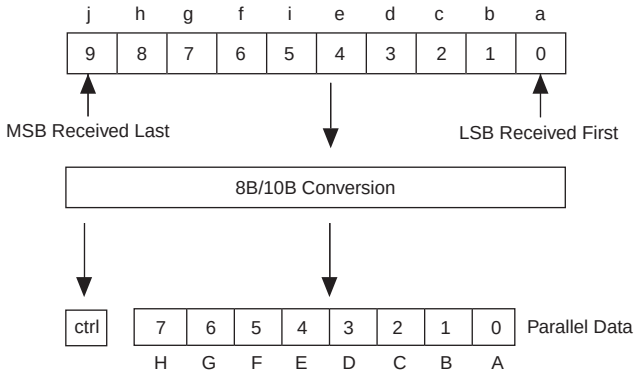
The 8B/10B decoder operates in single-width mode in the following functional modes:

- PCI Express (PIPE)
- XAUI
- GIGE
- Serial RapidIO
- Basic single-width

For PIPE, XAUI, GIGE, and Serial RapidIO functional modes, the ALTGX MegaWizard Plug-In Manager forces selection of the 8B/10B decoder in the receiver datapath. In Basic single-width mode, it allows you to enable or disable the 8B/10B decoder depending on your proprietary protocol implementation.

Figure 1–88 shows a 10-bit code group decoded into an 8-bit data and a 1-bit control identifier by the 8B/10B decoder in single-width mode.

Figure 1–88. 8B/10B Decoder in Single-Width Mode

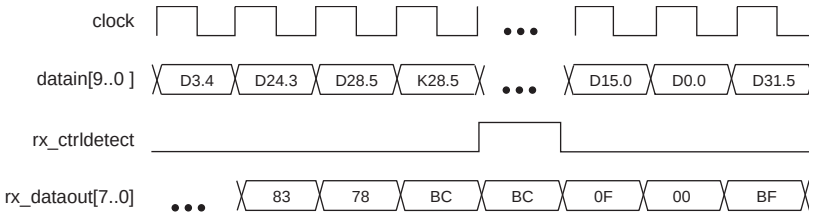


Control Code Group Detection

The 8B/10B decoder indicates whether the decoded 8-bit code group is a data or control code group on the `rx_ctrldetect` port. If the received 10-bit code group is one of the 12 control code groups (/Kx.y/) specified in the IEEE802.3 specification, the `rx_ctrldetect` signal is driven high. If the received 10-bit code group is a data code group (/Dx.y/), the `rx_ctrldetect` signal is driven low.

Figure 1–89 shows the 8B/10B decoder decoding the received 10-bit /K28.5/ control code group into an 8-bit data code group (8'hBC) driven on the `rx_dataout` port. The `rx_ctrldetect` signal is asserted high synchronous with 8'hBC on the `rx_dataout` port, indicating that it is a control code group. The rest of the codes received are data code groups /Dx.y/.

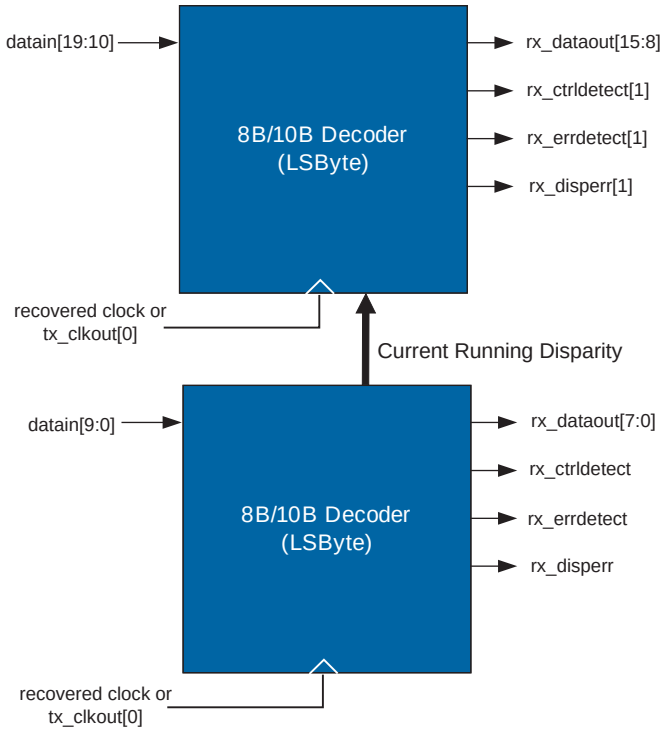
Figure 1–89. 8B/10B Decoder in Control Code Group Detection



8B/10B Decoder in Double-Width Mode

Figure 1–90 shows the 8B/10B decoder in double-width mode.

Figure 1–90. 8B/10B Decoder in Double-Width Mode



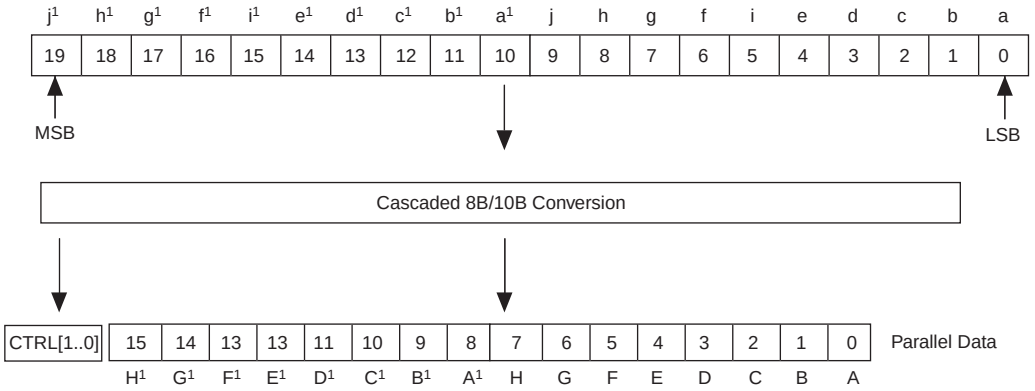
In double-width mode, two 8B/10B decoders are cascaded for decoding the 20-bit encoded data, as shown in Figure 1–91. The 10-bit LSByte of the received 20-bit encoded data is decoded first and the ending running disparity is forwarded to the 8B/10B decoder responsible for decoding the 10-bit MSByte. The cascaded 8B/10B decoder decodes the 20-bit encoded data into 16-bit data + 2-bit control identifier. The MSB and LSB of the 2-bit control identifier corresponds to the MSByte and LSByte of the 16-bit decoded data code group. The decoded data is fed to the byte deserializer or the receiver phase compensation FIFO (if byte deserializer is disabled).

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

The 8B/10B decoder operates in double-width mode only in Basic double-width functional mode. You can enable or disable the 8B/10B decoder depending on your proprietary protocol implementation.

Figure 1–91 shows a 20-bit code group decoded into 16-bit data and 2-bit control identifier by the 8B/10B decoder in double-width mode.

Figure 1–91. 8B/10 Decoder in 20-Bit Double-Width Mode

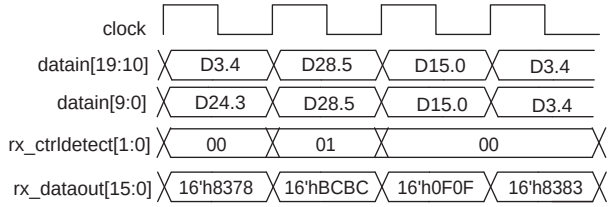


Control Code Group Detection

The cascaded 8B/10B decoder indicates whether the decoded 16-bit code group is a data or control code group on the 2-bit `rx_ctrldetect[1:0]` port. The `rx_ctrldetect[0]` signal is driven high or low depending on whether decoded data on the `rx_dataout[7:0]` port (LSByte) is a control or data code group, respectively. The `rx_ctrldetect[1]` signals are driven high or low depending on whether decoded data on the `rx_dataout[15:8]` port (MSByte) is a control or data code group, respectively.

Figure 1–92 shows the 8B/10B decoding of the received 10-bit /K28.5/ control code group into 8-bit data code group (8'hBC) driven on the `rx_dataout` port. The `rx_ctrldetect` signal is asserted high synchronous with 8'hBC on the `rx_dataout` port, indicating that it is a control code group. The rest of the codes received are data code groups /Dx.y/.

Figure 1–92. 8B/10B Decoder 10-Bit Control Code Group



Byte Deserializer

The core fabric-transceiver interface frequency has an upper limit of 250 MHz. In functional modes that have a receiver PCS frequency greater than 250 MHz, the parallel received data and status signals cannot be forwarded directly to the core fabric because it violates the upper limit of the 250 MHz core fabric-transceiver interface frequency. In such configurations, the byte deserializer is required to reduce the core fabric-transceiver interface frequency to half while doubling the parallel data

width. For example, at 3.2 Gbps data rate with a deserialization factor of 10, the receiver PCS datapath runs at 320 MHz. The 10-bit parallel received data and status signals at 320 MHz cannot be forwarded to the core fabric because it violates the upper limit of 250 MHz. The byte serializer converts the 10-bit parallel received data at 320 MHz into 20-bit parallel data at 160 MHz before forwarding to the core fabric.

The byte deserializer is required in configurations that exceed the core fabric-transceiver interface clock upper frequency limit. It is optional in configurations that do not exceed the core fabric-transceiver interface clock upper frequency limit.

The byte deserializer operates in two modes:

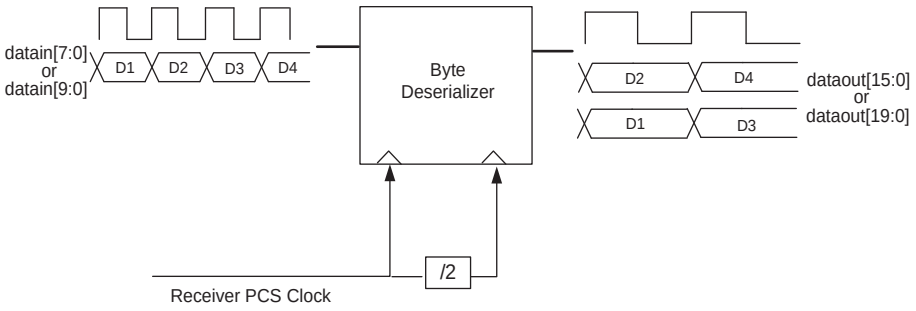
- Single-width mode
- Double-width mode

Byte Deserializer in Single-Width Mode

In single-width mode, the byte deserializer receives 8 bit wide data from the 8B/10B decoder or 10 bit wide data from the word aligner (if the 8B/10B decoder is disabled) and deserializes it into 16 bit or 20 bit wide data at half the speed.

Figure 1–93 shows the byte deserializer in single-width mode.

Figure 1–93. Byte Deserializer in Single-Width Mode

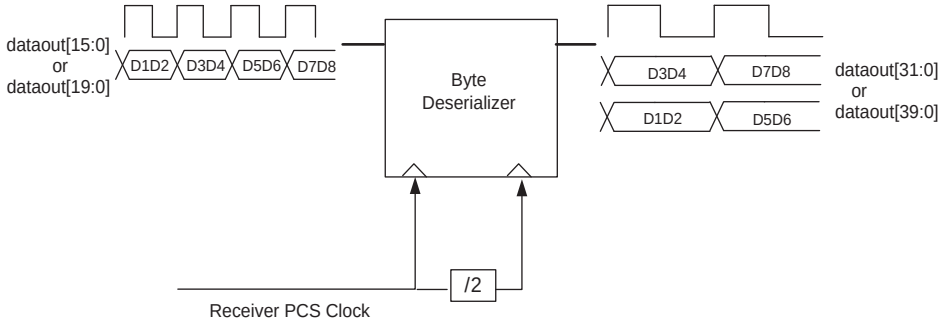


Byte Deserializer in Double-Width Mode

In double-width mode, the byte deserializer receives 16 bit wide data from the 8B/10B decoder or 20 bit wide data from the word aligner (if the 8B/10B decoder is disabled) and deserializes it into 32 bit or 40 bit wide data at half the speed.

Figure 1–94 shows the byte deserializer in double-width mode.

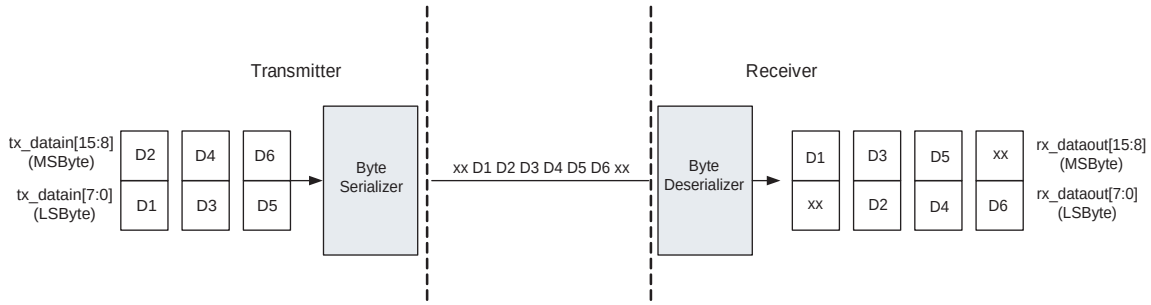
Figure 1–94. Byte Deserializer in Double-Width Mode



Byte Ordering Block

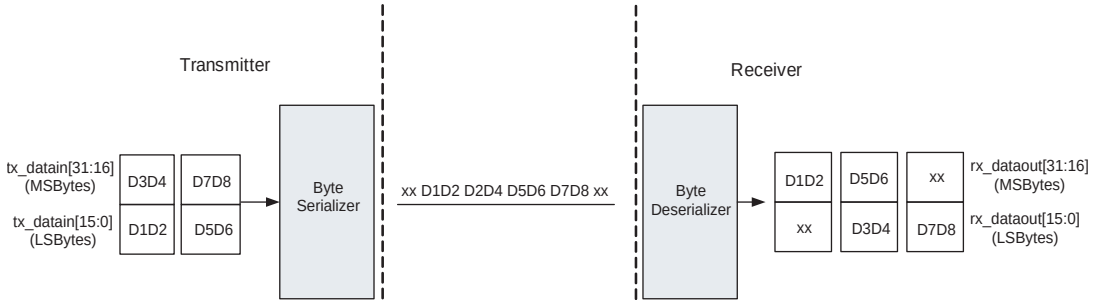
In single-width modes with the 16 bit or 20 bit core fabric-transceiver interface, the byte deserializer receives one data byte (8 or 10 bit) and deserializes it into two data bytes (16 or 20 bit). Depending on when the receiver PCS logic comes out of reset, the byte ordering at the output of the byte 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 1–95 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 1–95. MSByte and LSByte of the Two-Bit Transmitter Data Straddled Across Two Word Boundaries



In double-width modes with the 32 bit or 40 bit core fabric-transceiver interface, the byte deserializer receives two data bytes (16 or 20 bit) and deserializes it into four data bytes (32 or 40 bit). Figure 1–96 shows a scenario in which the two MSBytes and LSBytes of the four-byte transmitter data appears straddled across two word boundaries after getting byte deserialized at the receiver.

Figure 1–96. MSByte and LSByte of the Four-Bit 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 core fabric. The byte ordering block looks for the user-programmed byte ordering pattern in the byte-deserialized data. You must select a byte ordering pattern that you know appears at the LSByte(s) position of the parallel transmitter data. If the byte ordering block finds the programmed byte ordering pattern in the MSByte(s) position of the byte-deserialized data, it inserts the appropriate number of user-programmed PAD bytes to push the byte ordering pattern to the LSByte(s) position, thereby restoring proper byte ordering.

Byte Ordering Block in Single-Width Modes

The byte ordering block is available in the following single-width functional modes:

- SONET/SDH OC-48
- Basic single-width mode with:
  - 16-bit core fabric-transceiver interface
  - No 8B/10B decoder (8-bit PMA-PCS interface)
  - Word aligner in manual alignment mode
- Basic single-width mode with:
  - 16-bit core fabric-transceiver interface
  - 8B/10B decoder
  - Word aligner in automatic synchronization state machine mode

For more information about configurations that allow the byte ordering block in the receiver datapath, refer to "Basic Single-Width Mode Configurations" on page 1–120.

The Quartus II software automatically configures the byte ordering pattern and byte ordering PAD pattern for SONET/SDH OC-48 functional mode. For more information, refer to "OC-48 Byte Ordering" on page 1–173.

In Basic single-width mode, you can program a custom byte ordering pattern and byte ordering PAD pattern in the ALTGX MegaWizard Plug-In Manager. Table 1–26 shows the byte ordering pattern length allowed in Basic single-width mode.

Table 1–26. Byte Ordering Pattern Length in Basic Single-Width Mode

Functional Mode	Byte Ordering Pattern Length	Byte Ordering PAD Pattern Length
Basic single-width mode with: ■ 16-bit core fabric-transceiver interface ■ No 8B/10B decoder ■ Word aligner in manual alignment mode	8 bit	8 bit
Basic single-width mode with: ■ 16-bit core fabric-transceiver interface ■ 8B/10B decoder ■ Word aligner in automatic synchronization state machine mode	9 bit *(1)*	9 bit

Note to Table 1–26:

(1) If a /Kx.y/ control code group is selected as the byte ordering pattern, the MSB of the 9-bit byte ordering pattern must be 1'b1. If a /Dx.y/ data code group is selected as the byte ordering pattern, the MSB of the 9-bit byte ordering pattern must be 1'b0. The least significant 8 bits must be the 8B/10B decoded version of the code group used for byte ordering.

Byte Ordering Block in Double-Width Modes

The byte ordering block is available in the following double-width functional modes:

- Basic double-width mode with:
  - 32-bit core fabric-transceiver interface
  - No 8B/10B decoder (16-bit PMA-PCS interface)
  - Word aligner in manual alignment mode
- Basic double-width mode with:
  - 32-bit core fabric-transceiver interface
  - 8B/10B decoder (20-bit PMA-PCS interface)
  - Word aligner in manual alignment mode
- Basic double-width mode with:
  - 40-bit core fabric-transceiver interface
  - No 8B/10B decoder (20-bit PMA-PCS interface)
  - Word aligner in manual alignment mode

For more information about configurations that allow the byte ordering block in the receiver datapath, refer to "Basic Double-Width Mode Configurations" on page 1–122.

In Basic double-width modes, you can program a custom byte ordering pattern and byte ordering PAD pattern in the ALTGX MegaWizard Plug-In Manager. Table 1–27 shows the byte ordering pattern length allowed in Basic double-width mode.

Table 1–27. Byte Ordering Pattern Length in Basic Double-Width Mode

Functional Mode	Byte Ordering Pattern Length	Byte Ordering PAD Pattern Length
Basic double-width mode with: ■ 32-bit core fabric-transceiver interface ■ No 8B/10B decoder (16-bit PMA-PCS interface) ■ Word aligner in manual alignment mode	16 bit, 8 bit	8 bit
Basic double-width mode with: ■ 32-bit core fabric-transceiver interface ■ 8B/10B decoder (20-bit PMA-PCS interface) ■ Word aligner in manual alignment mode	18 bit, 9 bit *(1)*	9 bit
Basic double-width mode with: ■ 40-bit core fabric-transceiver interface ■ No 8B/10B decoder (20-bit PMA-PCS interface) ■ Word aligner in manual alignment mode	20 bit, 10 bit	10 bit

Note to Table 1–27:

(1) The 18-bit byte ordering pattern `D[17:0]` consists of MSByte `D[17:9]` and LSByte `D[8:0]`, `D[17]` corresponds to `rx_ctrldetect[1]` and `D[16:9]` corresponds to `rx_dataout[15:8]`. Similarly, `D[9]` corresponds to `rx_ctrldetect[0]` and `D[7:0]` corresponds to `rx_dataout[7:0]`.

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

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

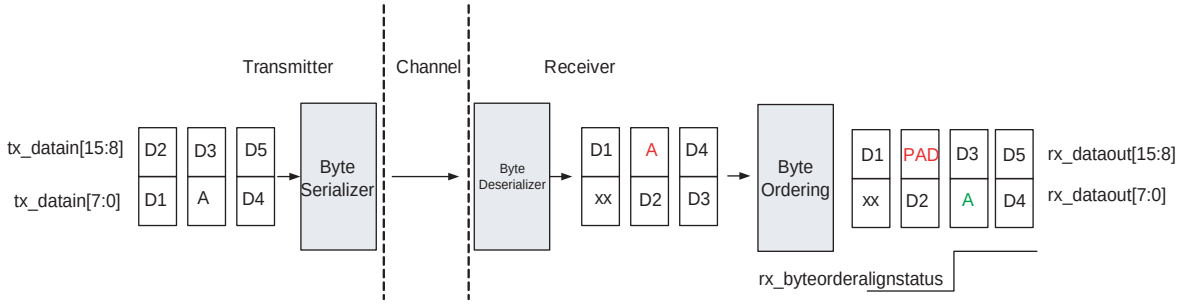
Word-Alignment-Based Byte Ordering

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

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

Figure 1–97 shows an example of the byte ordering operation in single-width modes. In this example, A is the programmed byte ordering pattern and PAD is the programmed PAD pattern. The byte deserialized data places the byte ordering pattern A in the MSByte position, resulting in incorrect byte ordering. Assuming that a rising edge on the `rx_syncstatus` signal had occurred before the byte ordering block sees the byte ordering pattern A in the MSByte position, the byte ordering block inserts a PAD byte and pushes the byte ordering pattern A in the LSByte position. The data at the output of the byte ordering block has correct byte ordering as reflected on the `rx_byteorderalignstatus` signal.

Figure 1–97. Byte Ordering in Single-Width Modes



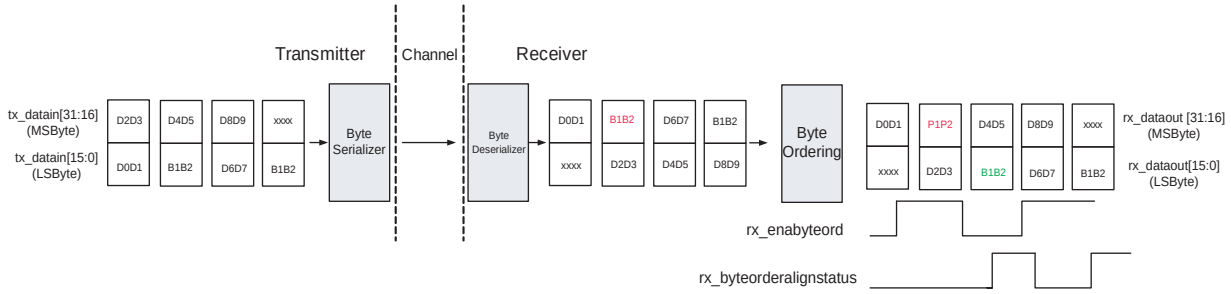
If the byte ordering block sees another rising edge on the `rx_syncstatus` signal from the word aligner, it de-asserts the `rx_byteorderalignstatus` signal and repeats the byte ordering operation as previously described.

User-Controlled Byte Ordering

Unlike word-alignment-based byte ordering, user-controlled byte ordering provides control to the user logic to restore correct byte ordering at the receiver. When enabled, an `rx_enabyteord` port is available that you can use to trigger the byte ordering operation. A rising edge on the `rx_enabyteord` port triggers the byte ordering block. After a rising edge on the `rx_enabyteord` signal, if the byte ordering block finds the first data byte that matches the programmed byte ordering pattern in the MSByte position of the byte-deserialized data, it inserts one programmed PAD pattern to push the byte ordering pattern in the LSByte position. If the byte ordering blocks finds the first data byte that matches the programmed byte ordering pattern in the LSByte position of the byte-deserialized data, it considers the data to be byte ordered and does not insert any PAD byte. In either case, the byte ordering block asserts the `rx_byteorderalignstatus` signal.

Figure 1–98 shows user-controlled byte ordering in Basic double-width Mode.

Figure 1–98. User-Controlled Byte Ordering in Basic Double-Width Mode



Receiver Phase Compensation FIFO

The receiver phase compensation FIFO in each channel ensures reliable transfer of data and status signals between the receiver channel and the core fabric. The receiver phase compensation FIFO compensates for the phase difference between the parallel receiver PCS clock (FIFO write clock) and the core fabric clock (FIFO read clock).

The receiver phase compensation FIFO operates in one of the following two modes:

- Low latency mode—The Quartus II software automatically configures the receiver phase compensation FIFO in low latency mode in all functional modes. In this mode, the FIFO is four words deep and the latency through the FIFO is two to three parallel clock cycles (pending characterization).
- High latency mode—In this mode, the FIFO is eight words deep and the latency through the FIFO is four to five parallel clock cycles (pending characterization).

The receiver phase compensation FIFO write clock source varies with the receiver channel configuration. Table 1–28 shows the receiver phase compensation FIFO write clock source in different configurations.

Table 1–28. 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 1–28. 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 1–29 shows the receiver phase compensation FIFO read clock source in different configurations.

Table 1–29. 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	Core fabric clock driven by the clock signal on the tx_clkout port	Core fabric clock driven by the clock signal on the rx_coreclk port
Non-bonded channel configuration without rate matcher	Core fabric clock driven by the clock signal on the rx_clkout port	Core fabric clock driven by the clock signal on the rx_coreclk port
×4 bonded channel configuration	Core fabric clock driven by the clock signal on the coreclkout port	Core fabric clock driven by the clock signal on the rx_coreclk port
×8 bonded channel configuration	Core fabric clock driven by the clock signal on the coreclkout port	Core fabric clock driven by the clock signal on the rx_coreclk port

Note to Table 1–29:

(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 underrun 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 underrun or overflow condition as a probable cause of link errors.

Offset Cancellation in the Receiver Buffer and Receiver CDR

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

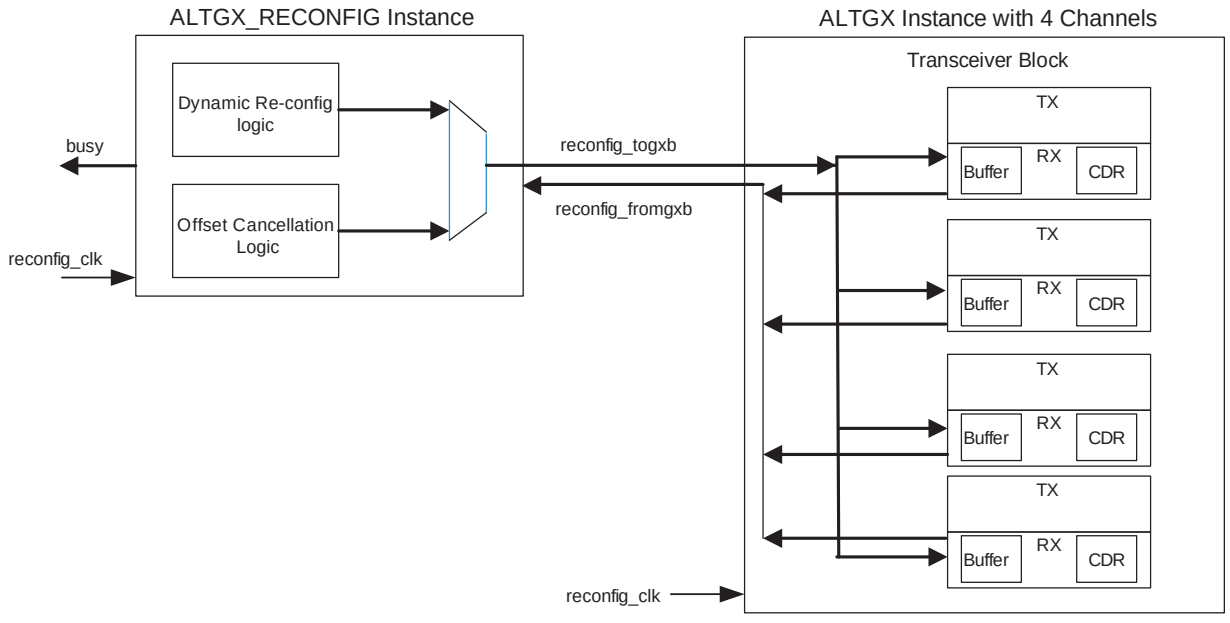
Offset cancellation is executed automatically once each time a HardCopy IV GX device is powered on. The control logic for offset cancellation is integrated into the ALTGX_RECONFIG megafunction. To use this logic, you need to enable the offset cancellation option in the ALTGX_RECONFIG MegaWizard Plug-In Manager. Additionally, the `reconfig_fromgxb` and `reconfig_togxb` buses and the necessary clocks need to be connected between the ALTGX instance and the ALTGX_RECONFIG instance.

For more information about offset cancellation control logic connectivity, refer to the *HardCopy IV GX Dynamic Reconfiguration* chapter in volume 3 of the *HardCopy IV Device Handbook*.

During offset cancellation, signified by high on the `busy` signal, the `rx_analogreset` is not relevant until the `busy` signal goes low.

Offset cancellation logic requires a separate clock. In PIPE mode, you must connect the clock input to the `fixedclk` port provided by the ALTGX MegaWizard Plug-In Manager. The frequency of this clock input must be 125 MHz. For all other functional modes, connect the clock input to the `reconfig_clk` port provided by the ALTGX MegaWizard Plug-In Manager. The frequency of the clock connected to the `reconfig_clk` port must be within the range of 37.5 to 50 MHz. Figure 1–99 shows the interface of the offset cancellation control logic (ALTGX_RECONFIG instance) and the ALTGX instance.

Figure 1–99. Interface of Offset Cancellation Control Logic to the ALTGX Instance



The offset cancellation process begins by disconnecting the path from the receiver input buffer to the receiver CDR. It then sets the receiver CDR into a fixed set of dividers to guarantee a V_{CO} clock rate that is within the range necessary to provide proper offset cancellation. Subsequently, the offset cancellation process goes through various states and culminates in the offset cancellation of the receiver buffer and the receiver CDR.

After offset cancellation is complete, your divider settings are restored. Then the reconfiguration block sends and receives data to the ALTGX using the `reconfig_togxb` and `reconfig_fromgxb` buses. Connect the buses between the ALTGX_RECONFIG and ALTGX instances. The de-assertion of the `busy` signal from the offset cancellation control logic indicates the offset cancellation process is complete.

Functional Modes

You can configure HardCopy IV GX transceivers in one of the following functional modes using the ALTGX MegaWizard Plug-In Manager:

- Basic
  - Basic single-width at 600 Mbps to 3.75 Gbps
  - Basic double-width at 1 Gbps to 6.5 Gbps
- PCI Express (PIPE) (Gen1 at 2.5 Gbps and Gen2 at 5 Gbps)
- XAUI (3.125 Gbps up to HiGig at 3.75 Gbps)
- GIGE (1.25 Gbps)
- Serial RapidIO (1.25 Gbps, 2.5 Gbps, and 3.125 Gbps)
- SONET/SDH (OC-12 and OC-48)
- (OIF) CEI PHY Interface (>3.135 Gbps to 6.375 Gbps)
- SDI (HD at 1.485/1.4835 Gbps and 3G at 2.97/2.967 Gbps)

Basic Functional Mode

The HardCopy IV GX transceiver datapath is extremely flexible in Basic functional mode. To configure the transceiver 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 1–30 shows the PCS-PMA interface widths and data rates supported in Basic single-width and double-width modes.

Table 1–30. PCS-PMA Interface Widths and Data Rates in Basic Single-Width and Double-Width Modes

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 1–30:

(1) The data rate range supported in Basic single-width and double-width modes varies depending on whether or not you use the byte serializer/deserializer. For more information, refer to "Basic Single-Width Mode Configurations" on page 1–120 and "Basic Double-Width Mode Configurations" on page 1–122.

Low Latency PCS Datapath

The ALTGX MegaWizard Plug-In Manager provides an **Enable low latency PCS mode** option when configured in Basic single-width or Basic double-width mode. If you select this option, the following transmitter and receiver channel PCS blocks are bypassed to yield a low latency PCS datapath:

- 8B/10B encoder and decoder
- Word aligner
- Deskew FIFO
- Rate match (clock rate compensation) FIFO
- Byte ordering

In low latency PCS modes, the transmitter and receiver phase compensation FIFOs are always enabled. Depending on the targeted data rate, you can optionally bypass the byte serializer and deserializer blocks. Refer to "Basic Single-Width Mode Configurations" on page 1–120 and "Basic Double-Width Mode Configurations" on page 1–122 for more information.

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

Basic Single-Width Mode Configurations

Figure 1–100 shows HardCopy IV GX transceiver configurations allowed in Basic single-width functional mode with an 8-bit PMA-PCS interface.

Figure 1–100. Transceiver Configurations in Basic Single-Width Mode with an 8-Bit PMA-PCS Interface

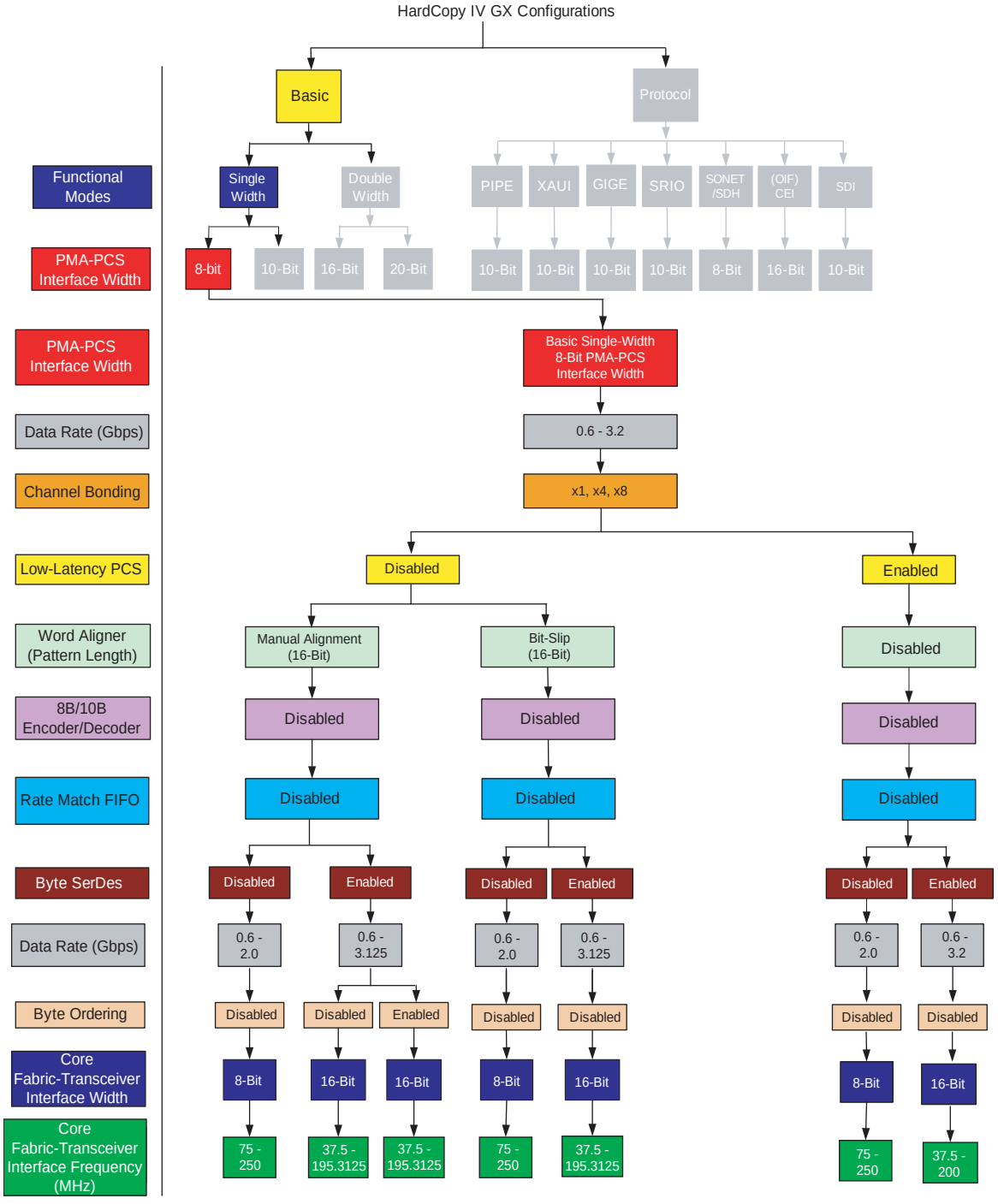
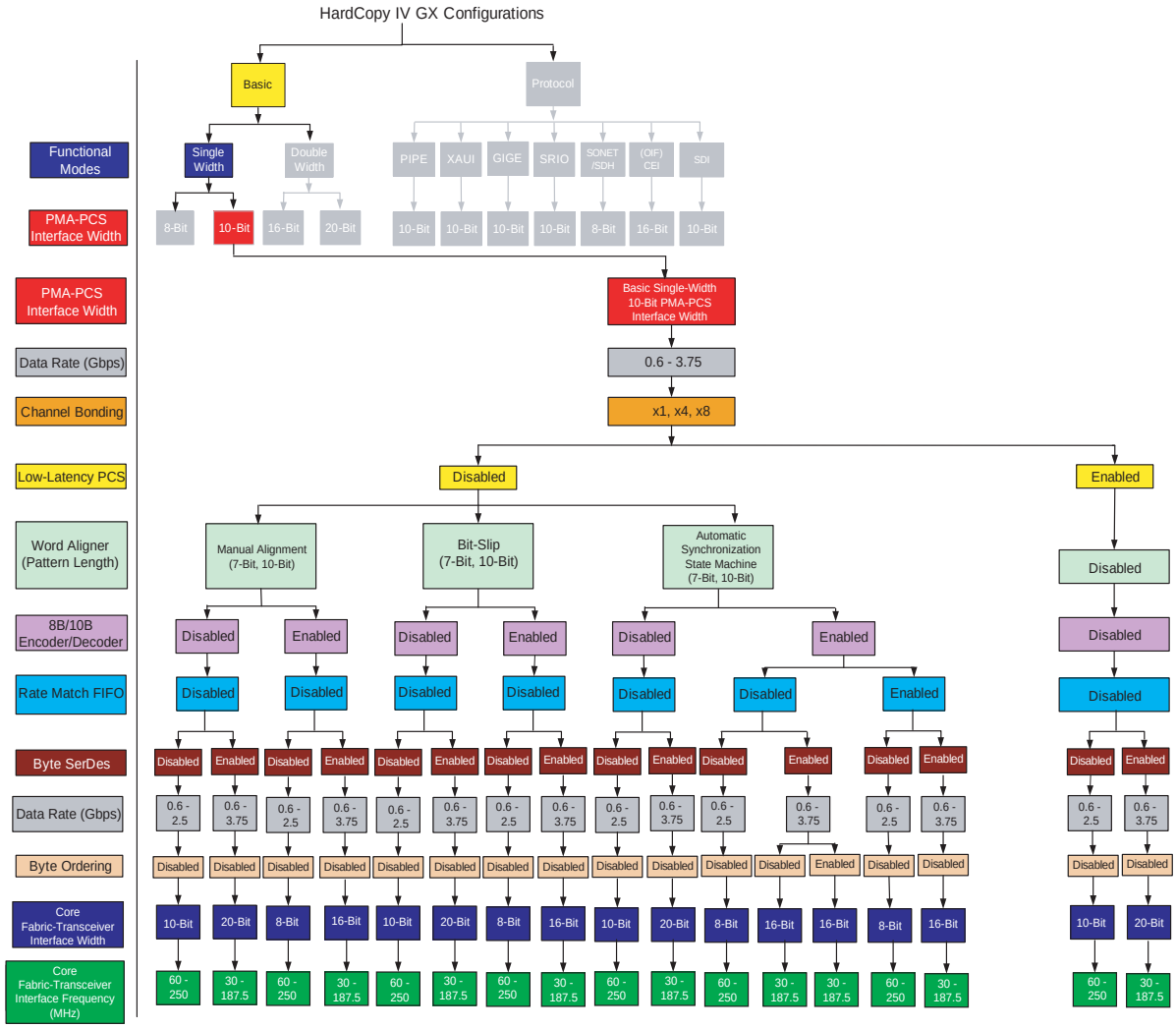


Figure 1–101 shows HardCopy IV GX transceiver configurations allowed in Basic single-width functional mode with a 10-bit PMA-PCS interface.

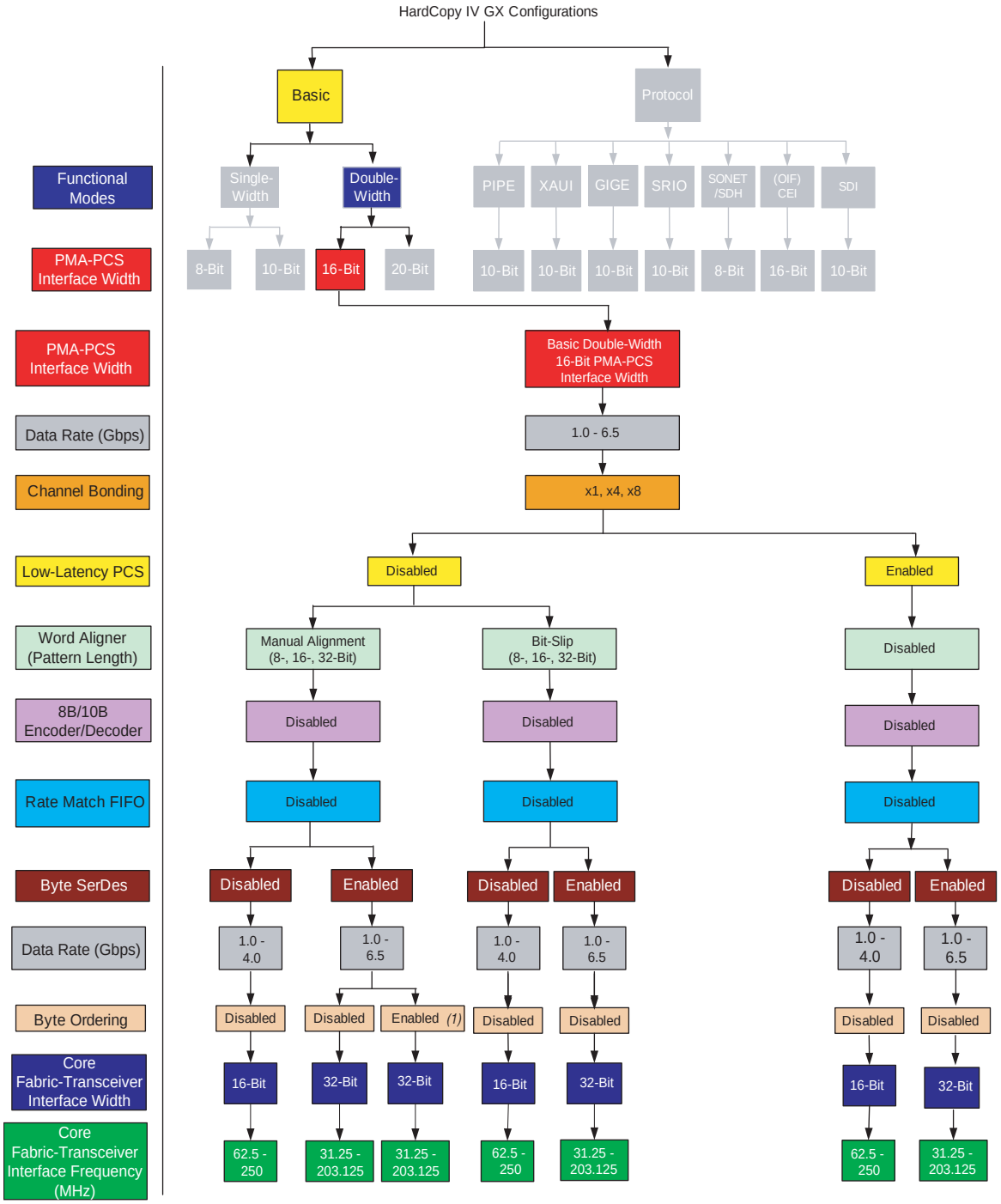
Figure 1–101. Transceiver Configurations in Basic Single-Width Mode with a 10-Bit PMA-PCS Interface



Basic Double-Width Mode Configurations

Figure 1–102 shows HardCopy IV GX transceiver configurations allowed in Basic double-width functional mode with a 16-bit PMA-PCS interface.

Figure 1–102. Transceiver Configurations in Basic Double-Width Mode with a 16-Bit PMA-PCS Interface

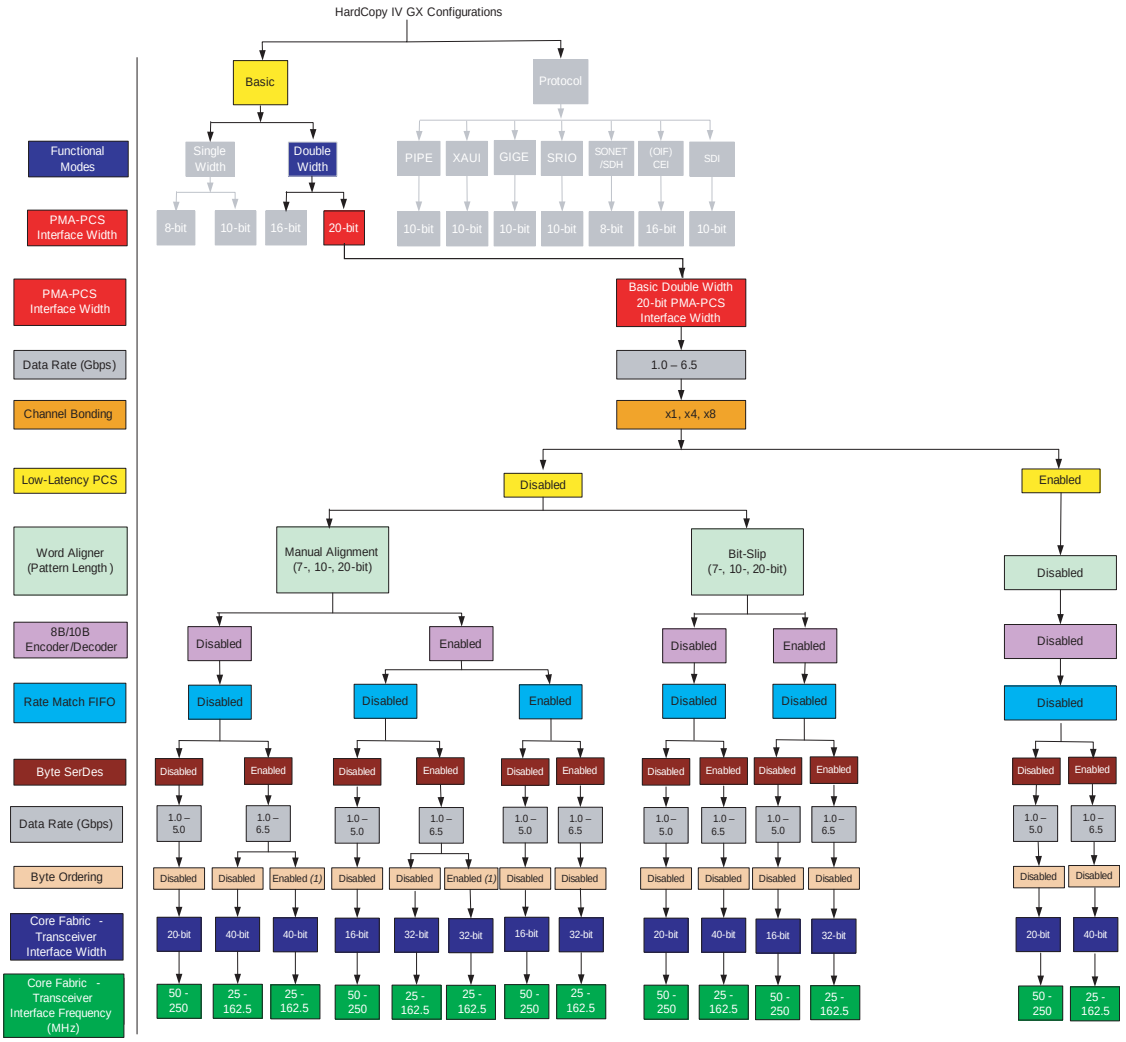


Note to Figure 1–102:

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

Figure 1–103 shows HardCopy IV GX transceiver configurations allowed in Basic double-width functional mode with a 20-bit PMA-PCS interface.

Figure 1–103. Transceiver Configurations in Basic Double-Width Mode with a 20-Bit PMA-PCS Interface



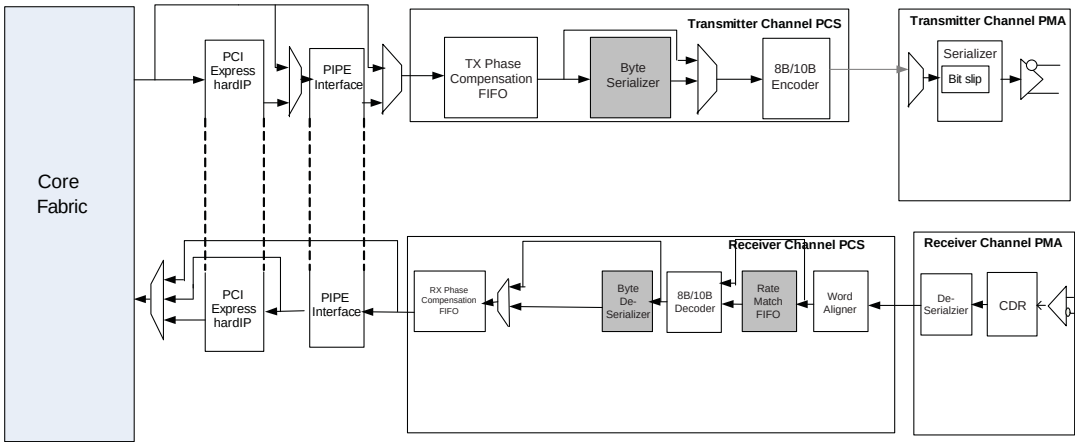
Note to Figure 1–103:

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

Deterministic Latency Options

The HardCopy IV GX device has a deterministic latency option available for use in high-speed serial interfaces such as CPRI (Common Public Radio Interface). This option is available in single-width mode with 8/10-bit channel width and double-width mode with 16/20-bit channel width options only.

Figure 1–104. Transceiver Datapath When in Deterministic Latency Mode



To implement deterministic latency mode under the Basic functional mode, the transmitter must be placed in bit-slip mode, the receiver's phase compensation FIFO must be placed in register mode and a port on the receiver, `rx_bitslipboundaryselectout[4:0]` should be used. For example, for a full duplex (with both receiver and transmitter channels) link, you must select **Enable the RX phase comp FIFO in register mode** in the ALTGX MegaWizard Plug-In Manager. When this option is selected, the transmitter is placed in bit-slipping mode and the `tx_bitslipboundaryselect[4:0]` port is automatically available. Similarly, the `rx_bitslipboundaryselectout[4:0]` output port is automatically available.

Rx 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 0 bits are slipped, the output on `rx_bitslipboundaryselectout[4:0]` shows a value of 19 (10011); if two bits are slipped, the output on `rx_bitslipboundaryselectout[4:0]` shows a value of 17 (10001).

The information on 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 Bit Slipping

The transmitter is bit slipped to achieve deterministic latency. To use this feature, select the **Create the 'tx_bitslipboundaryselect[4:0] port to control the number of bits slipped in the TX bitslipper** option in the ALTGX MegaWizard Plug-In Manager. The `tx_bitslipboundaryselect[4:0]` input port is used to set the number of bits that the transmitter block needs to slip.

- In the Basic single—width mode with 8/10-bit channel width you can slip zero to 9 bits.
- In the Basic double—width mode with 16/20-bit channel width you can slip zero to 19 bits.

PCI Express (PIPE) Mode

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

To implement a Version 2.0 PIPE-compliant PHY, you must configure the HardCopy IV GX transceivers in PIPE functional mode. HardCopy IV GX devices have built-in PIPE hard IP blocks that you can use to implement the PHY-MAC layer, data link layer, and transaction layer of the PIPE protocol stack. You can also bypass the PIPE hard IP blocks and implement the PHY-MAC layer, data link layer, and transaction layer in the FGPA fabric using a soft IP. If you enable the PIPE hard IP blocks, the HardCopy IV GX transceivers interface with these hard IP blocks. Otherwise, the HardCopy IV GX transceivers interface with the core fabric.

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

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

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

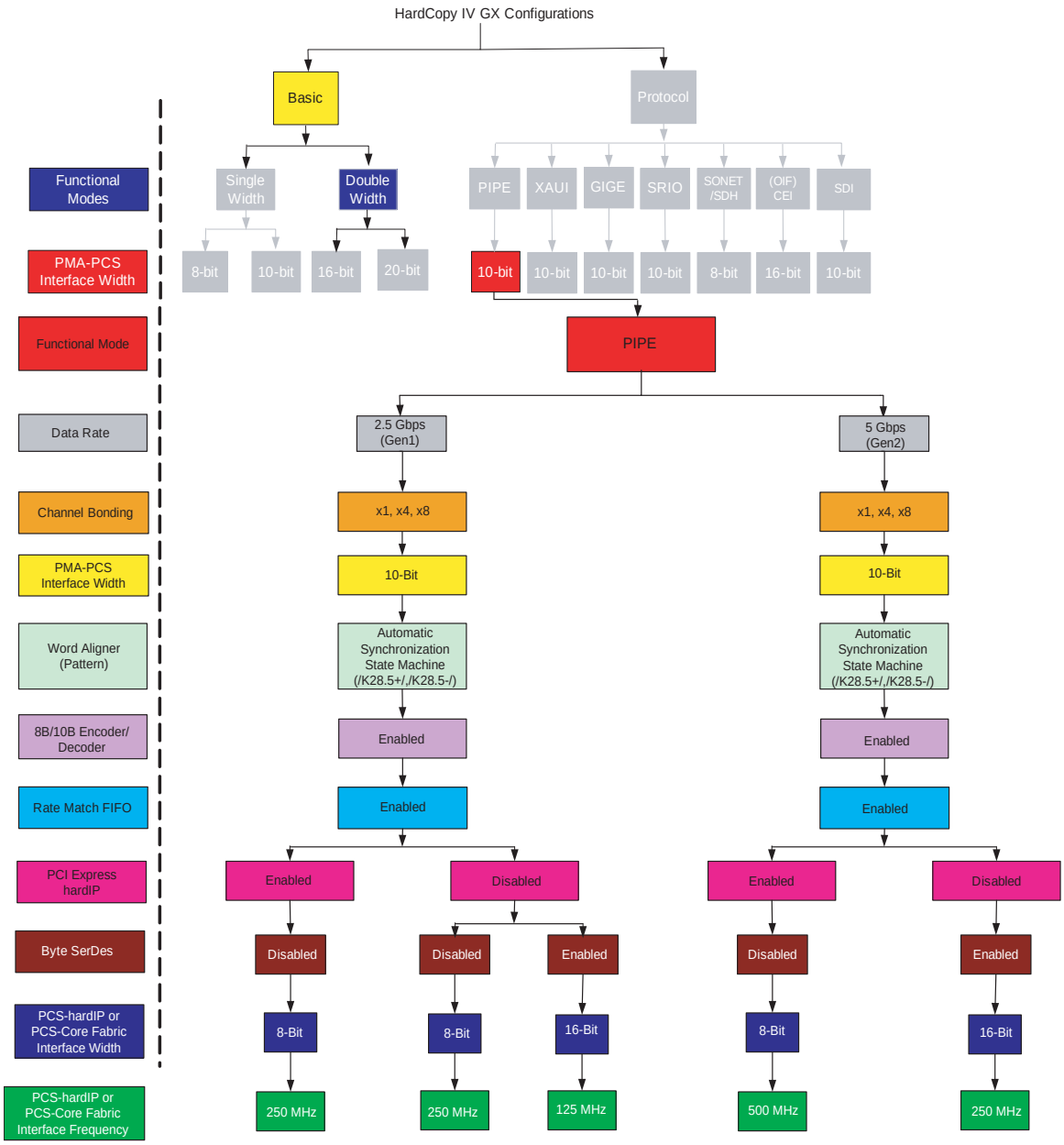
PCI Express (PIPE) Mode Configurations

HardCopy IV GX transceivers support both Gen1 (2.5 Gbps) and Gen2 (5 Gbps) data rates in PIPE functional mode. When configured for a 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 PIPE signaling rates is critical for speed negotiation during link training.

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

Figure 1–105 shows the HardCopy IV GX transceiver configurations allowed in PCI Express (PIPE) functional mode.

Figure 1–105. HardCopy IV GX Transceivers in PCI Express (PIPE) Functional Mode



PCI Express (PIPE) Mode Datapath

Figure 1–106 shows the HardCopy IV GX transceiver datapath when configured in PCI Express (PIPE) functional mode.

Figure 1–106. HardCopy IV GX Transceiver Datapath in PCI Express (PIPE) ×1 Mode

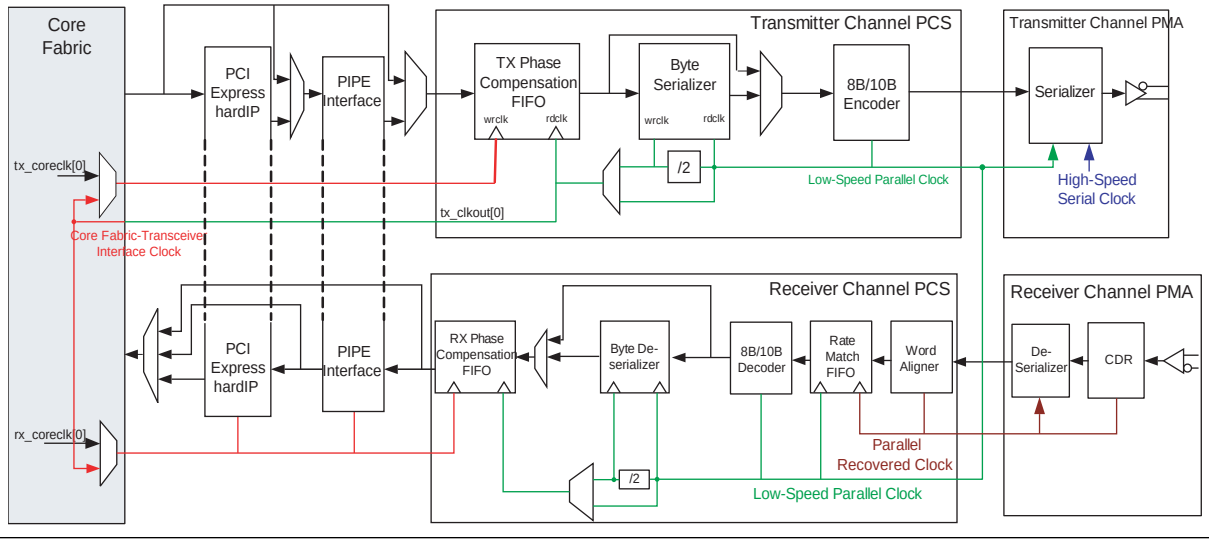


Table 1–31 shows the transceiver datapath clock frequencies in PIPE functional mode configured using the ALTGX MegaWizard Plug-In Manager.

Table 1–31. HardCopy IV GX Transceiver Datapath Clock Frequencies in PCI Express (PIPE) Mode

Functional Mode	Data Rate	High-Speed Serial Clock Frequency	Parallel Recovered Clock and Low-Speed Parallel Clock Frequency	Core Fabric-Transceiver Interface Clock Frequency	
				Without Byte Serializer/ Deserializer (8-Bit Wide)	With Byte Serializer/ Deserializer (16-Bit Wide)
PCI Express (PIPE) ×1, ×4, ×8 (Gen1)	2.5 Gbps	1.25 GHz	250 MHz	250 MHz	125 MHz
PCI Express (PIPE) ×1, ×4, ×8 (Gen2)	5 Gbps	2.5 GHz	500 MHz	N/A *(1)*	250 MHz

Note to Table 1–31:

(1) In PCI Express (PIPE) functional mode at Gen2 (5 Gbps) data rate, the byte serializer/deserializer cannot be bypassed.

Transceiver datapath clocking varies between non-bonded (×1) and bonded (×4 and ×8) configurations in PIPE mode.

The transmitter datapath in PCI Express (PIPE) mode consists of:

- PCI Express (PIPE) interface
- Transmitter phase compensation FIFO

- Optional byte serializer (enabled for 16-bit and disabled for 8-bit core fabric-transceiver interface)
- 8B/10B encoder
- 10:1 serializer
- Transmitter buffer with receiver detect circuitry

The receiver datapath in PCI Express (PIPE) mode consists of:

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

Table 1–32 shows features supported in PIPE functional mode for 2.5 Gbps and 5 Gbps data rate configurations.

Table 1–32. Supported Features in PCI Express (PIPE) Mode

Feature	2.5 Gbps (Gen1)	5 Gbps (Gen2)
×1, ×4, ×8 link configurations	✓	✓
PIPE-compliant synchronization state machine	✓	✓
±300 PPM (total 600 PPM) clock rate compensation	✓	✓
8-bit core fabric-transceiver interface	✓	—
16-bit core fabric-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	—	✓
Dynamically selectable transmitter margining for differential output voltage control	—	✓
Dynamically selectable transmitter buffer de-emphasis of -3.5 db and -6 dB	—	✓
Dynamically selectable full-swing and half-swing transmitter output voltage levels	—	✓

PCI Express (PIPE) Interface

In PIPE mode, each channel has a PIPE interface block that transfers data, control, and status signals between the PHY-MAC layer and the transceiver channel PCS and PMA blocks. The PIPE interface block is compliant to version 2.0 of the PIPE specification. If you use the PIPE hard IP block, the PHY-MAC layer is implemented in the hard IP block. Otherwise, the PHY-MAC layer can be implemented using soft IP in the core fabric.

The PIPE interface block is only used in PIPE mode and cannot be bypassed.

Besides transferring data, control, and status signals between the PHY-MAC layer and the transceiver, the PIPE interface block implements the following functions required in a PIPE-compliant physical layer device:

- Forces the transmitter buffer in electrical idle state
- Initiates the receiver detect sequence
- 8B/10B encoder disparity control when transmitting compliance pattern
- Manages the PIPE power states
- Indicates the completion of various PHY functions; for example, receiver detection and power state transitions on the `pipephydonestatus` signal
- Encodes the receiver status and error conditions on the `pipestatus[2:0]` signal as specified in the PIPE specification

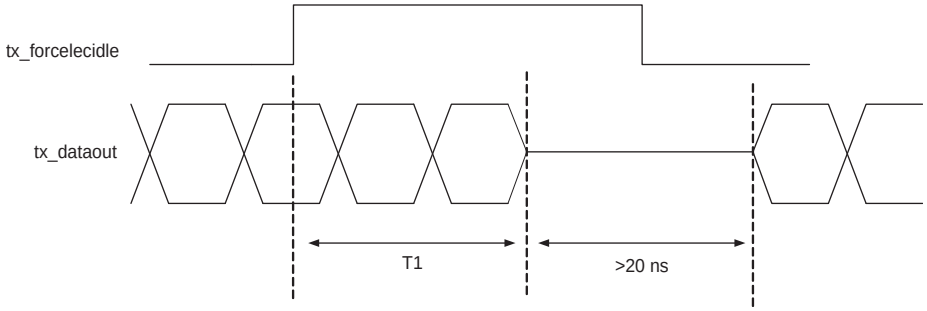
Transmitter Buffer Electrical Idle

When the input signal `tx_forceelecidle` is asserted high, the PIPE interface block puts the transmitter buffer in that channel in the electrical idle state. During electrical idle, the transmitter buffer differential and common mode output voltage levels are compliant to the PIPE Base Specification 2.0 for both PIPE Gen1 and Gen2 data rates.

Figure 1–107 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 PIPE protocol requires the transmitter buffer to stay in electrical idle for a minimum of 20 ns for both Gen1 and Gen2 data rates.

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

Figure 1–107. Transmitter Buffer Electrical Idle State



The PCI Express (PIPE) specification requires the transmitter buffer to be in electrical idle in certain power states. For more information about the `tx_forceelecidle` signal levels required in different PIPE power states, refer to Table 1–31 on page 1–128.

Receiver Detection

During the detect substate of the link training and status state machine (LTSSM), the PIPE protocol requires the transmitter channel to perform a receiver detect sequence to detect if a receiver is present at the far end of each lane. The PIPE specification requires the receiver detect operation to be performed during the P1 power state.

The PIPE interface block in HardCopy IV GX transceivers provides an input signal `tx_detectrxloopback` for the receiver detect operation. When the input signal `tx_detectrxloopback` is asserted high in the P1 power state, the PIPE interface block sends a command signal to the transmitter buffer in that channel to initiate a receiver detect sequence. In the P1 power state, the transmitter buffer must always be in the electrical idle state. After receiving this command signal, the receiver detect circuitry creates a step voltage at the output of the transmitter buffer. If an active receiver (that complies to the PIPE input impedance requirements) is present at the far end, the time constant of the step voltage on the trace is higher than when the receiver is not present. Receiver detect circuitry monitors the time constant of the step signal seen on the trace to determine if a receiver was detected. The receiver detect circuitry monitor needs a 125-MHz clock for operation that you must drive on the `fixedclk` port.

For the receiver detect circuitry to function reliably, the AC-coupling capacitor on the serial link and the receiver termination values used in your system must be compliant to the PCI Express (PIPE) Base Specification 2.0.

Receiver detect circuitry communicates the status of the receiver detect operation to the PIPE interface block. If a far-end receiver is successfully detected, the PIPE interface block asserts `pipephydonestatus` for one clock cycle and synchronously drives the `pipestatus[2:0]` signal to 3'b011. If a far-end receiver is not detected, the PIPE interface block asserts `pipephydonestatus` for one clock cycle and synchronously drives the `pipestatus[2:0]` signal to 3'b000.

Figure 1–108 and Figure 1–109 show the receiver detect operation where a receiver was successfully detected and where a receiver was not detected, respectively.

Figure 1–108. Receiver Detect, Successfully Detected

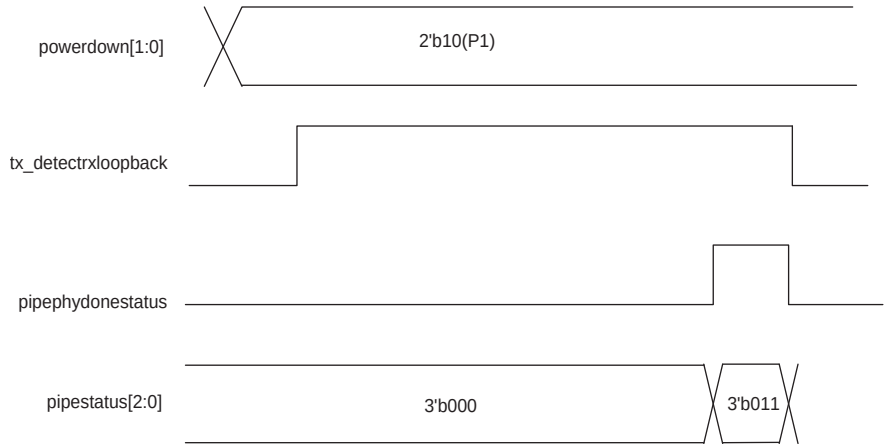
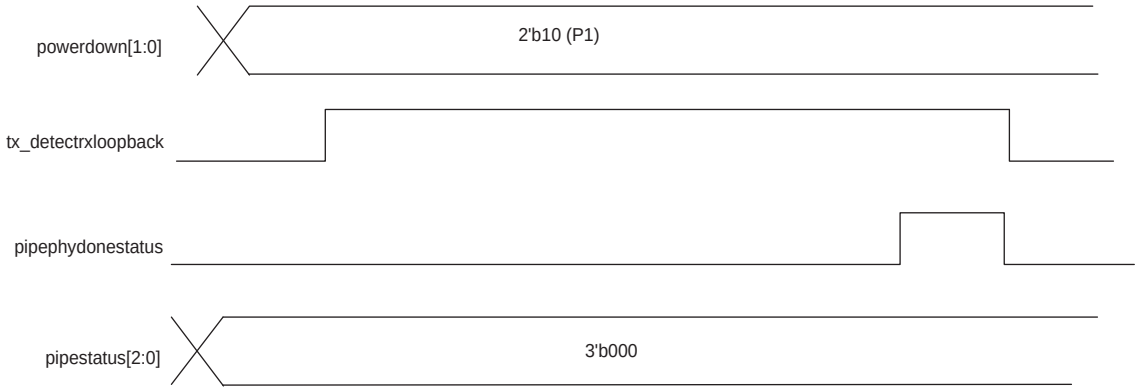


Figure 1–109. Receiver Detect, Unsuccessfully Detected



Compliance Pattern Transmission Support

The LTSSM state machine can enter the polling.compliance substate where the transmitter is required to transmit a compliance pattern as specified in the PCI Express (PIPE) Base Specification 2.0. The polling.compliance substate is intended to assess if the transmitter is electrically compliant with the PIPE voltage and timing specifications.

The compliance pattern is a repeating sequence of the following four code groups:

- /K28.5/
- /D21.5/
- /K28.5/
- /D10.2/

The PCI Express (PIPE) protocol requires the first /K28.5/ code group of the compliance pattern to be encoded with negative current disparity. To satisfy this requirement, the PIPE interface block provides the input signal `tx_forcedispcompliance`. A high level on `tx_forcedispcompliance` forces the associated parallel transmitter data on the `tx_datain` port to transmit with negative current running disparity.

- For 8-bit transceiver channel width configurations, you must drive `tx_forcedispcompliance` high in the same parallel clock cycle as the first /K28.5/ of the compliance pattern on the `tx_datain` port.
- For 16-bit transceiver channel width configurations, you must drive only the LSB of `tx_forcedispcompliance[1:0]` high in the same parallel clock cycle as /K28.5/D21.5/ of the compliance pattern on the `tx_datain` port.

Figure 1–110 and Figure 1–111 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 1–110. Compliance Pattern Transmission Support, 8-Bit Channel Width Configurations

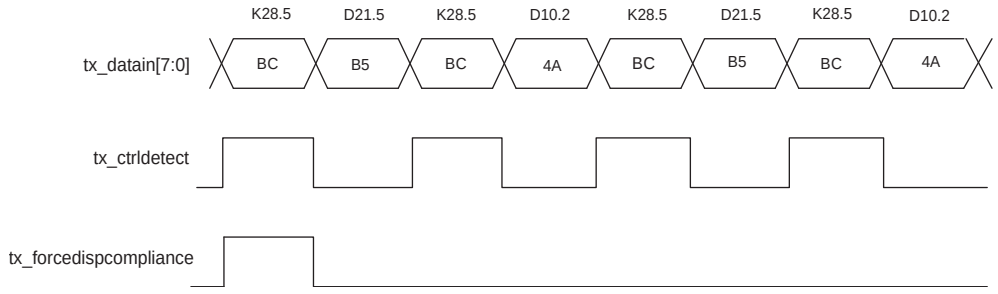
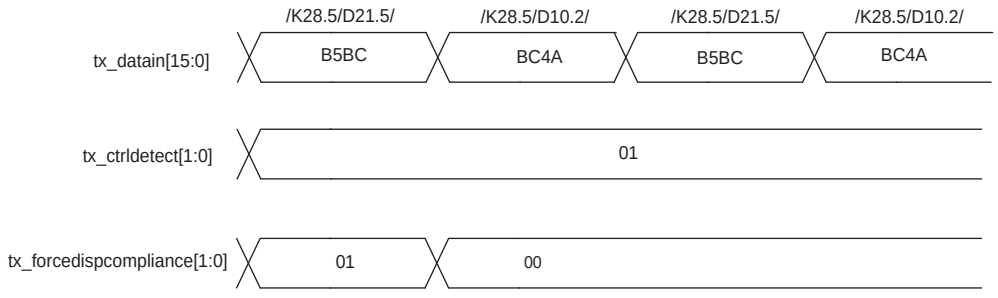


Figure 1–111. Compliance Pattern Transmission Support, 16-Bit Channel Width Configurations



Power State Management

The PCI Express (PIPE) specification defines four power states—P0, P0s, P1, and P2— that the physical layer device must support to minimize power consumption.

- P0 is the normal operating state during which packet data is transferred on the PIPE link.
- P0s, P1, and P2 are low-power states into which the physical layer must transition as directed by the PHY-MAC layer to minimize power consumption.

The PIPE specification provides the mapping of these power states to the LTSSM states specified in the PIPE Base Specification 2.0. The PHY-MAC layer is responsible for implementing the mapping logic between the LTSSM states and the four power states in the PIPE-compliant PHY.

The PCI Express (PIPE) interface in HardCopy IV GX transceivers provides an input port, `powerdn[1:0]`, for each transceiver channel configured in PIPE mode. Table 1–33 shows mapping between the logic levels driven on the `powerdn[1:0]` port and the resulting power state that the PIPE interface block puts the transceiver channel into.

Table 1–33. Power State Functions and Descriptions

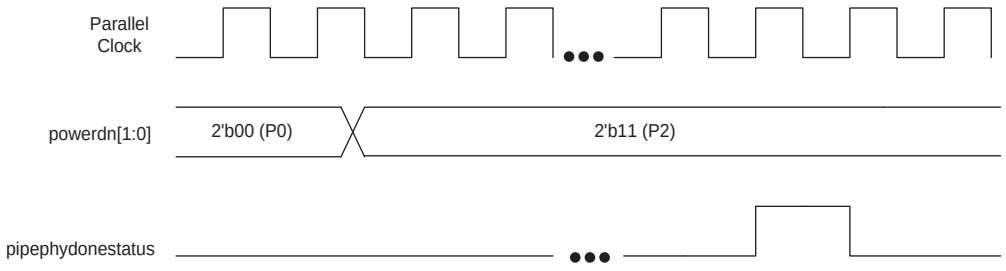
Power State	powerdn	Function	Description
P0	2'b00	Transmits normal data, transmits electrical idle, or enters into loopback mode	Normal operation mode
P0s	2'b01	Only transmits electrical idle	Low recovery time saving state
P1	2'b10	Transmitter buffer is powered down and can do a receiver detect while 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 PIPE 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 PIPE interface block indicates successful power state transition by asserting the `pipephydonestatus` signal for one parallel clock cycle as specified in the PIPE specification. The PHY-MAC layer must not request any further power state transition until the `pipephydonestatus` signal has indicated the completion of the current power state transition request.

Figure 1–112 shows an example waveform for a transition from the P0 to P2 power state.

Figure 1–112. Power State Transition from P0 to P2



The PIPE specification allows the PIPE interface to perform protocol functions; for example, receiver detect, loopback, and beacon transmission, in specified power states only. This requires the PHY-MAC layer to drive the `tx_detectrxloopback` and `tx_forceelecidle` signals appropriately in each power state to perform these functions. Table 1–34 summarizes the logic levels that the PHY-MAC layer must drive on the `tx_detectrxloopback` and `tx_forceelecidle` signals in each power state.

Table 1–34. Logic Levels for the PHY-MAC Layer

Power State	tx_detectrxloopback	tx_forceelecidle
P0	0: normal mode 1: datapath in loopback mode	0: Must be de-asserted 1: Illegal mode
P0s	Don't care	0: Illegal mode 1: Must be asserted in this state
P1	0: Electrical Idle 1: receiver detect	0: Illegal mode 1: Must be asserted in this state
P2	Don't care	De-asserted in this state for sending beacon. Otherwise asserted.

Receiver Status

The PCI Express (PIPE) specification requires the PHY to encode the receiver status on a 3-bit `RxStatus[2:0]` signal. This status signal is used by the PHY-MAC layer for its operation.

The PIPE interface block receives status signals from the transceiver channel PCS and PMA blocks and encodes the status on the 3-bit output signal `pipestatus[2:0]` to the core fabric. The encoding of the status signals on `pipestatus[2:0]` is compliant with the PIPE specification and is listed in Table 1–35.

Table 1–35. 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 PIPE interface follows the priority listed in Table 1–35 while encoding the receiver status on the `pipestatus[2:0]` port. For example, if the PIPE interface receives an 8B/10B decode error and disparity error for the same symbol, it drives 3'b100 on the `pipestatus[2:0]` signal.

Fast Recovery Mode

The PIPE Base specification fast training sequences (FTS) are used for bit and byte synchronization to transition from L0s to L0 (PIPE P0s to P0) power states. When transitioning from L0s to L0 power state, the PIPE Base Specification requires the physical layer device to acquire bit and byte synchronization after receiving a maximum of 255 FTS (~4 us at Gen1 data rate and ~2 us at Gen2 data rate).

If the HardCopy IV GX receiver CDR is configured in Automatic Lock mode, the receiver cannot meet the PCI Express (PIPE) specification of acquiring bit and byte synchronization within 4 µs (Gen1 data rate) or 2 µs (Gen2 data rate) due to the signal detect and PPM detector time. To meet this specification, each 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 PIPE protocol allows inferring the electrical idle condition at the receiver instead of detecting the electrical idle condition using analog circuitry. Clause 4.2.4.3 in the PIPE Base Specification 2.0 specifies conditions to infer electrical idle at the receiver in various substates of the LTSSM state machine.

In all PIPE modes (×1, ×4, and ×8), each receiver channel PCS has an optional Electrical Idle Inference module designed to implement the electrical idle inference conditions specified in the PIPE Base Specification 2.0. You can enable the Electrical Idle Inference module by selecting the **Enable electrical idle inference functionality** option in the ALTGX MegaWizard Plug-In manager.

If enabled, this module infers electrical idle depending on the logic level driven on the `rx_elecidleinfersel[2:0]` input signal. The Electrical Idle Inference module in each receiver channel indicates whether the electrical idle condition is inferred or not on the `pipeelecidle` signal of that channel. The Electrical Idle Interface module drives the `pipeelecidle` signal high if it infers an electrical idle condition; otherwise, it drives it low.

Table 1–36 shows electrical idle inference conditions specified in the PIPE Base Specification 2.0 and implemented in the Electrical Idle Inference module to infer electrical idle in various substates of the LTSSM state machine. For the Electrical Idle Inference Module to correctly infer an electrical idle condition in each LTSSM substate, you must drive the `rx_elecidleinfersel[2:0]` signal appropriately, as shown in Table 1–36.

Table 1–36. Electrical Idle Inference Conditions (Part 1 of 2)

LTSSM State	Gen1 (2.5 Gbps)	Gen2 (5 Gbps)	rx_elecidleinfersel[2:0]
L0	Absence of skip ordered set in 128 µs window	Absence of skip ordered set in 128 µs window	3'b100
Recovery.RcvrCfg	Absence of TS1 or TS2 ordered set in 1280 UI interval	Absence of TS1 or TS2 ordered set in 1280 UI interval	3'b101

Table 1–36. Electrical Idle Inference Conditions (Part 2 of 2)

LTSSM State	Gen1 (2.5 Gbps)	Gen2 (5 Gbps)	rx_elecidleinfersel[2:0]
Recovery.Speed when successful speed negotiation = 1'b1	Absence of TS1 or TS2 ordered set in 1280 UI interval	Absence of TS1 or TS2 ordered set in 1280 UI window	3'b101
Recovery.Speed when successful speed negotiation = 1'b0	Absence of an exit from Electrical Idle in 2000 UI interval	Absence of an exit from Electrical Idle in 16000 UI interval	3'b110
Loopback.Active (as slave)	Absence of an exit from Electrical Idle in 128 µs window	—	3'b111

In the Recovery.Speed substate of the LTSSM state machine with unsuccessful speed negotiation (`rx_elecidleinfersel[2:0]` = `3'b110`), the PCI Express (PIPE) Base Specification requires the receiver to infer an electrical idle condition (`pipeelecidle` = high) if absence of an exit from Electrical Idle is detected in a 2000 UI interval for Gen1 data rate and 16000 UI interval for Gen2 data rate. The electrical idle inference module detects an absence of exit from Electrical Idle if four /K28.5/ COM code groups are not received in the specified interval.

In other words, when configured for Gen1 data rate and `rx_elecidleinfersel[2:0]` = 3'b110, the Electrical Idle Inference module asserts `pipeelecidle` high if it does not receive four /K28.5/ COM code groups in a 2000 UI interval. When configured for Gen1 data rate and `rx_elecidleinfersel[2:0]` = 3'b111 in the Loopback Active substate of the LTSSM state machine, the Electrical Idle Inference module asserts `pipeelecidle` high if it does not receive four /K28.5/ COM code groups in a 128 µs interval.

When configured for Gen2 data rate and `rx_elecidleinfersel[2:0]` = 3'b110, the Electrical Idle Inference module asserts `pipeelecidle` high if it does not receive four /K28.5/ COM code groups in a 16000 UI interval.

The Electrical Idle Inference module does not have the capability to detect the electrical idle exit condition based on reception of the electrical idle exit ordered set (EIEOS), as specified in the PCI Express (PIPE) Base Specification.

If you select the **Enable Electrical Idle Inference Functionality** option in the ALTGX MegaWizard Plug-In Manager and drive `rx_elecidleinfersel[2:0] = 3'b0xx`, the Electrical Idle Inference block uses the EIOS detection from the Fast Recovery circuitry to drive the `pipeelecidle` signal.

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

PCI Express (PIPE) Gen2 (5 Gbps) Support

The PCI Express (PIPE) functional mode supports the following additional features when configured for 5 Gbps data rate:

- Dynamic switch between 2.5 Gbps and 5 Gbps signaling rate

- Dynamically selectable transmitter margining for differential output voltage control
- Dynamically selectable transmitter buffer de-emphasis of -3.5 db and -6 dB
- Dynamically selectable full-swing and half-swing transmitter output voltage levels

Dynamic Switch Between Gen1 (2.5 Gbps) and Gen2 (5 Gbps) Signaling Rate

During link training, the upstream and downstream PCI Express (PIPE) ports negotiate the speed (2.5 Gbps or 5 Gbps) at which the link operates. Because the upstream and downstream PIPE ports do not know the speed capabilities of their link partner, the PIPE protocol requires each port to start with a Gen1 (2.5 Gbps) signaling rate. One of the ports capable of supporting the Gen2 (5 Gbps) signaling rate might initiate a speed change request by entering the Recovery state of the LTSSM. In the Recovery state, each port advertises its speed capabilities by transmitting training sequences as specified in the PIPE Base Specification 2.0. If both ports are capable of operating at the Gen2 (5 Gbps) signaling rate, the PHY-MAC layer instructs the physical layer device to operate at the Gen2 (5 Gbps) signaling rate.

To support speed negotiation during link training, the PIPE specification requires a PIPE-compliant physical layer device to provide an input signal (`Rate`) to the PHY-MAC layer. When this input signal is driven low, the physical layer device must operate at the Gen1 (2.5 Gbps) signaling rate; when driven high, this input signal must operate at the Gen2 (5 Gbps) signaling rate. The PIPE specification allows the PHY-MAC layer to initiate a signaling rateswitch only in power states P0 and P1 with the transmitter buffer in Electrical Idle state. The PIPE specification allows the physical layer device to implement the signaling rateswitch using either of the following approaches:

- Change the transceiver datapath clock frequency, keeping the transceiver interface width constant
- Change the transceiver interface width between 8 bit and 16 bit, keeping the transceiver clock frequency constant

When configured in PIPE functional mode at Gen2 (5 Gbps) data rate, the ALTGX MegaWizard Plug-In Manager provides the input signal `rateswitch`. The `rateswitch` signal is functionally equivalent to the `Rate` signal specified in the PIPE 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 PIPE rateswitch circuitry performs the dynamic switch between the Gen1 (2.5 Gbps) and Gen2 (5 Gbps) signaling rate. The PIPE rateswitch circuitry consists of:

- PCI Express (PIPE) rateswitch controller
- PCI Express (PIPE) clock switch circuitry

PCI Express (PIPE) Rateswitch Controller

The `rateswitch` signal serves as the input signal to the PCI Express (PIPE) rateswitch controller. After seeing a transition on the `rateswitch` signal from the PHY-MAC layer, the PCI Express (PIPE) rateswitch controller performs the following operations:

- Controls the PIPE clock switch circuitry to switch between Gen1 (2.5 Gbps) and Gen2 (5 Gbps) signaling rate, depending on the `rateswitch` signal level
- Disables and resets the transmitter and receiver phase compensation FIFO pointers until the PIPE clock switchover circuitry indicates successful rateswitch completion
- Communicates completion of rateswitch to the PIPE interface module, which in turn communicates completion of the rateswitch to the PHY-MAC layer on the `pipephydonestatus` signal

PCI Express (PIPE) rateswitch controller location:

- In PIPE ×1 mode, the PIPE rateswitch controller is located in the transceiver PCS of each channel.
- In PIPE ×4 mode, the PIPE rateswitch controller is located in `CMU0 Channel` within the transceiver block.
- In PIPE ×8 mode, the PIPE rateswitch controller is located in `CMU0_Channel` within the master transceiver block.

When operating at the Gen 2 data rate, asserting the `rx_digitalreset` signal causes the PIPE rateswitch circuitry to switch the transceiver to Gen 1 data rate.

When switching from Gen1 to Gen2 using the dynamic reconfiguration controller, you must set the two ports of the dynamic reconfiguration controller, `tx_preemp_0t` and `tx_preemp_2t`, to zero to meet the Gen2 de-emphasis specifications. When switching from Gen2 to Gen1, if your system requires specific settings on `tx_preemp_01` and `tx_preemp_2t`, those values must to be set at the respective two ports of the dynamic reconfiguration controller to meet your system requirements.

PCI Express (PIPE) Clock Switch Circuitry

When the PHY-MAC layer instructs a rateswitch between the Gen1 (2.5 Gbps) and Gen2 (5 Gbps) signaling rates, both the transmitter high-speed serial and low-speed parallel clock and the CDR recovered clock must switch to support the instructed data rate. HardCopy IV GX transceivers have dedicated PIPE clock switch circuitry located in the following blocks:

- Local clock divider in transmitter PMA of each transceiver channel
- `CMU0` clock divider in `CMU0_Channel` of each transceiver block
- Receiver CDR in receiver PMA of each transceiver channel

PCI Express (PIPE) transmitter high-speed serial and low-speed parallel clock switch occurs:

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

In PIPE ×1, ×4, and ×8 modes, the recovered clock switch happens in the receiver CDR of each transceiver channel.

Table 1–37 lists the locations of the PIPE rateswitch controller and the PIPE clock switch circuitry in PIPE ×1, ×2, ×4, and ×8 modes.

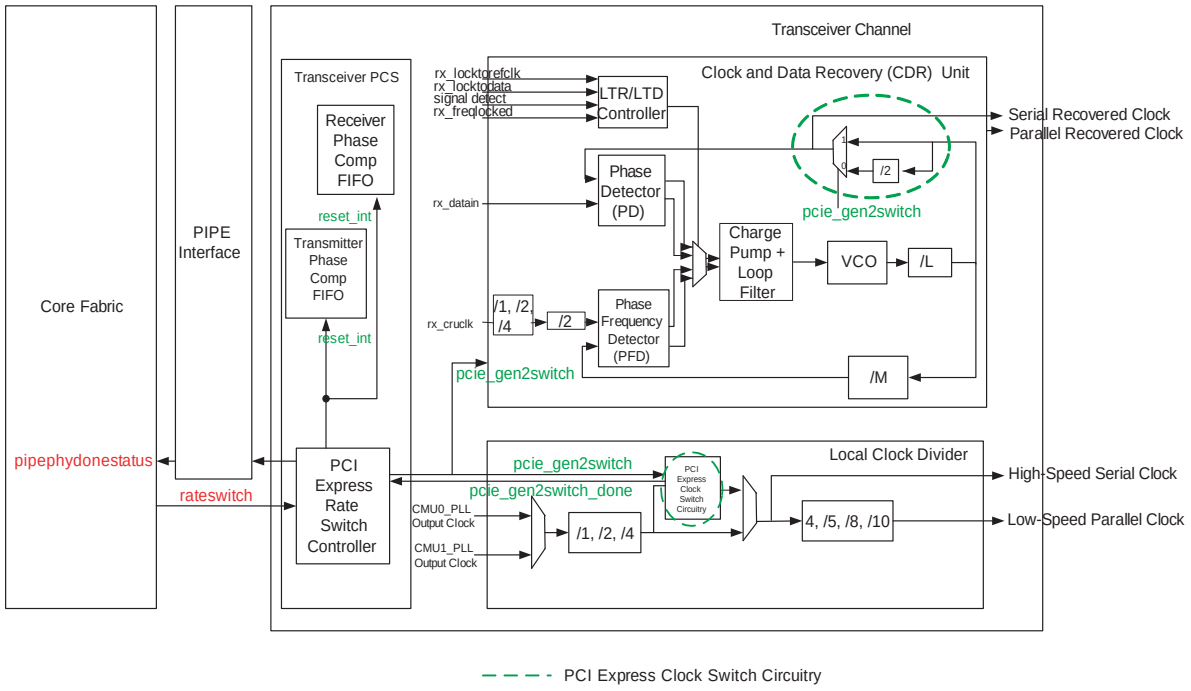
Table 1–37. PCI Express (PIPE) Rateswitch Controller and Clock Switch Circuitry

Channel Bonding Option	Location of PCI Express (PIPE) Rateswitch Controller Module	Location of PCI Express (PIPE) Clock Switch Circuitry	
		Transmitter High-Speed Serial and Low-Speed Parallel Clock Switch Circuitry	Recovered Clock Switch Circuitry
×1	Individual channel PCS block	Local clock divider in transmitter PMA of each channel	CDR block in receiver PMA of each channel
×4	CMU0 Channel	CMU0 clock divider in CMU0_Channel	CDR block in receiver PMA of each channel
×8	CMU0 Channel of the master transceiver block	CMU0 clock divider in CMU0_Channel of the master transceiver block	CDR block in receiver PMA of each channel

Dynamic Switch Between Gen1 (2.5 Gbps) and Gen2 (5 Gbps) Signaling Rates in PCI Express (PIPE) x1 Mode

Figure 1–113 shows the PCI Express (PIPE) rateswitch circuitry in PCI Express (PIPE) ×1 mode configured at Gen2 (5 Gbps) data rate.

Figure 1–113. Dynamic Switch Signaling in PIPE ×1 Mode



In PIPE ×1 mode configured at Gen2 (5 Gbps) data rate, when the PIPE rateswitch controller sees a transition on the `rateswitch` signal, it sends control signal `pcie_gen2switch` to the PIPE clock switch circuitry in the local clock divider block and the receiver CDR to switch to the instructed signaling rate. A low-to-high transition on the `rateswitch` signal initiates a Gen1 (2.5 Gbps) to Gen2 (5 Gbps) signaling rateswitch. A high-to-low transition on the `rateswitch` signal initiates a Gen2 (5 Gbps) to Gen1 (2.5 Gbps) signaling rateswitch.

Table 1–38 shows transceiver clock frequencies when switching between 2.5 Gbps and 5 Gbps signaling rates.

Table 1–38. Transceiver Clock Frequencies Signaling Rates in PCI Express (PIPE) ×1 Mode

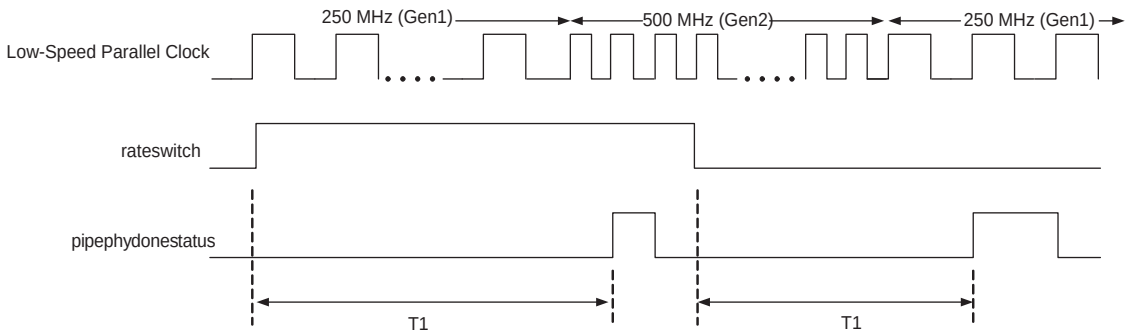
Transceiver Clocks	Gen1 (2.5 Gbps) to Gen2 (5 Gbps) Switch (Low-to-High Transition on the rateswitch Signal)	Gen2 (5 Gbps) to Gen1 (2.5 Gbps) Switch (High-to-Low Transition on the rateswitch Signal)
High-Speed Serial Clock	1.25 GHz to 2.5 GHz	2.5 GHz to 1.25 GHz
Low-Speed Parallel Clock	250 MHz to 500 MHz	500 MHz to 250 MHz
Serial Recovered Clock	1.25 GHz to 2.5 GHz	2.5 GHz to 1.25 GHz
Parallel Recovered Clock	250 MHz to 500 MHz	500 MHz to 250 MHz
Core Fabric-Transceiver Interface Clock	125 MHz to 250 MHz	250 MHz to 125 MHz

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

Figure 1–114 shows low-speed parallel clock switch between Gen1 (250 MHz) and Gen2 (500 MHz) in response to the change in the logic level on the `rateswitch` signal. The rateswitch completion is shown marked with a one clock cycle assertion of the `pipephydonestatus` signal.

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

Figure 1–114. Low-Speed Parallel Clock Switching in PCI Express (PIPE) ×1 Mode

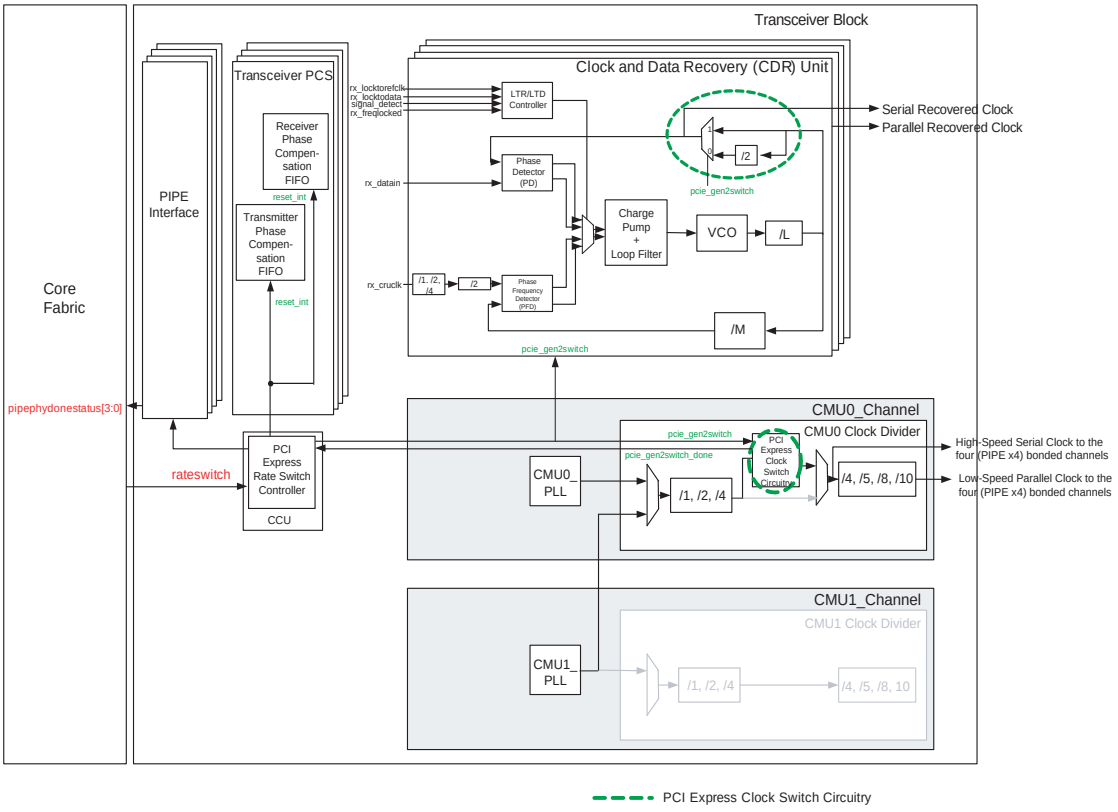


As a result of the signaling rateswitch between Gen1 (2.5 Gbps) and Gen2 (5 Gbps), the core fabric-transceiver interface clock switches between 125 MHz and 250 MHz. The core fabric-transceiver interface clock clocks the read side and write side of the transmitter phase compensation FIFO and the receiver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

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

Figure 1–115 shows the PIPE rateswitch circuitry in PIPE ×4 mode configured at Gen2 (5 Gbps) data rate.

Figure 1–115. Dynamic Switch Signaling in PCI Express (PIPE) ×4 Mode



In PIPE ×4 mode configured at Gen2 (5 Gbps) data rate, when the PIPE rateswitch controller sees a transition on the `rateswitch` signal, it sends the `pcie_gen2switch` control signal to the PIPE clock switch circuitry in the `CMU0` clock divider block and the receiver CDR to switch to the instructed signaling rate. A low-to-high transition on the `rateswitch` signal initiates a Gen1 (2.5 Gbps) to Gen2 (5 Gbps) signaling rateswitch. A high-to-low transition on the `rateswitch` signal initiates a Gen2 (5 Gbps) to Gen1 (2.5 Gbps) signaling rateswitch.

Table 1–39 shows the transceiver clock frequencies when switching between 2.5 Gbps and 5 Gbps signaling rates.

Table 1–39. Transceiver Clock Frequencies Signaling Rates in PCI Express (PIPE) ×4 Mode (Part 1 of 2)

Transceiver Clocks	Gen1 (2.5 Gbps) to Gen2 (5 Gbps) Switch (Low-to-High Transition on the rateswitch Signal)	Gen2 (5 Gbps) to Gen1 (2.5 Gbps) Switch (High-to-Low Transition on the rateswitch Signal)
High-Speed Serial Clock	1.25 GHz to 2.5 GHz	2.5 GHz to 1.25 GHz
Low-Speed Parallel Clock	250 MHz to 500 MHz	500 MHz to 250 MHz
Serial Recovered Clock	1.25 GHz to 2.5 GHz	2.5 GHz to 1.25 GHz

Table 1–39. Transceiver Clock Frequencies Signaling Rates in PCI Express (PIPE) ×4 Mode (Part 2 of 2)

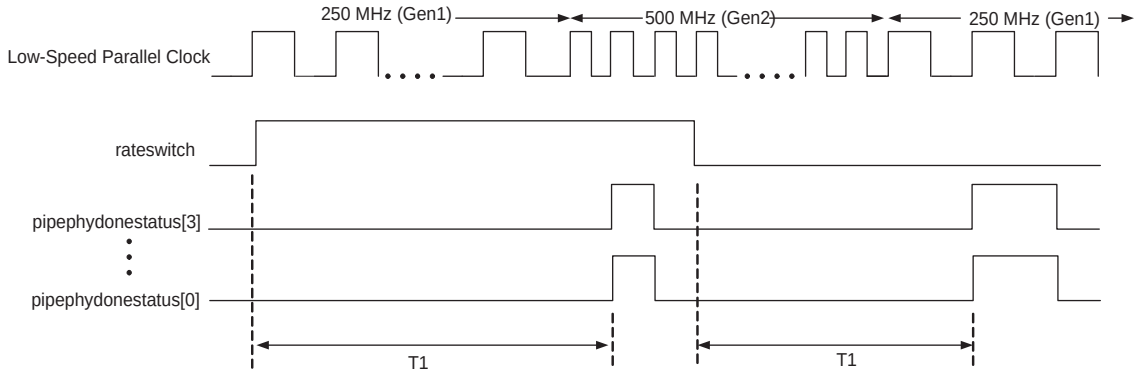
Transceiver Clocks	Gen1 (2.5 Gbps) to Gen2 (5 Gbps) Switch (Low-to-High Transition on the rateswitch Signal)	Gen2 (5 Gbps) to Gen1 (2.5 Gbps) Switch (High-to-Low Transition on the rateswitch Signal)
Parallel Recovered Clock	250 MHz to 500 MHz	500 MHz to 250 MHz
Core Fabric-Transceiver Interface Clock	125 MHz to 250 MHz	250 MHz to 125 MHz

The PCI Express (PIPE) clock switch circuitry in the `CMU0` clock divider block performs the clock switch between 250 MHz and 500 MHz on the low-speed parallel clock when switching between Gen1 (2.5 Gbps) and Gen2 (5 Gbps) signaling rates. It indicates successful completion of clock switch on the `pcie_gen2switchdone` signal to the PIPE rateswitch controller. The PIPE rateswitch controller forwards the clock switch completion status to the PIPE interface block. The PIPE interface block communicates the clock switch completion status to the PHY-MAC layer by asserting the `pipephydonestatus` signal of all bonded channels for one parallel clock cycle.

Figure 1–116 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 1–116. Low-Speed Parallel Clock Switching in PCI Express (PIPE) ×4 Mode

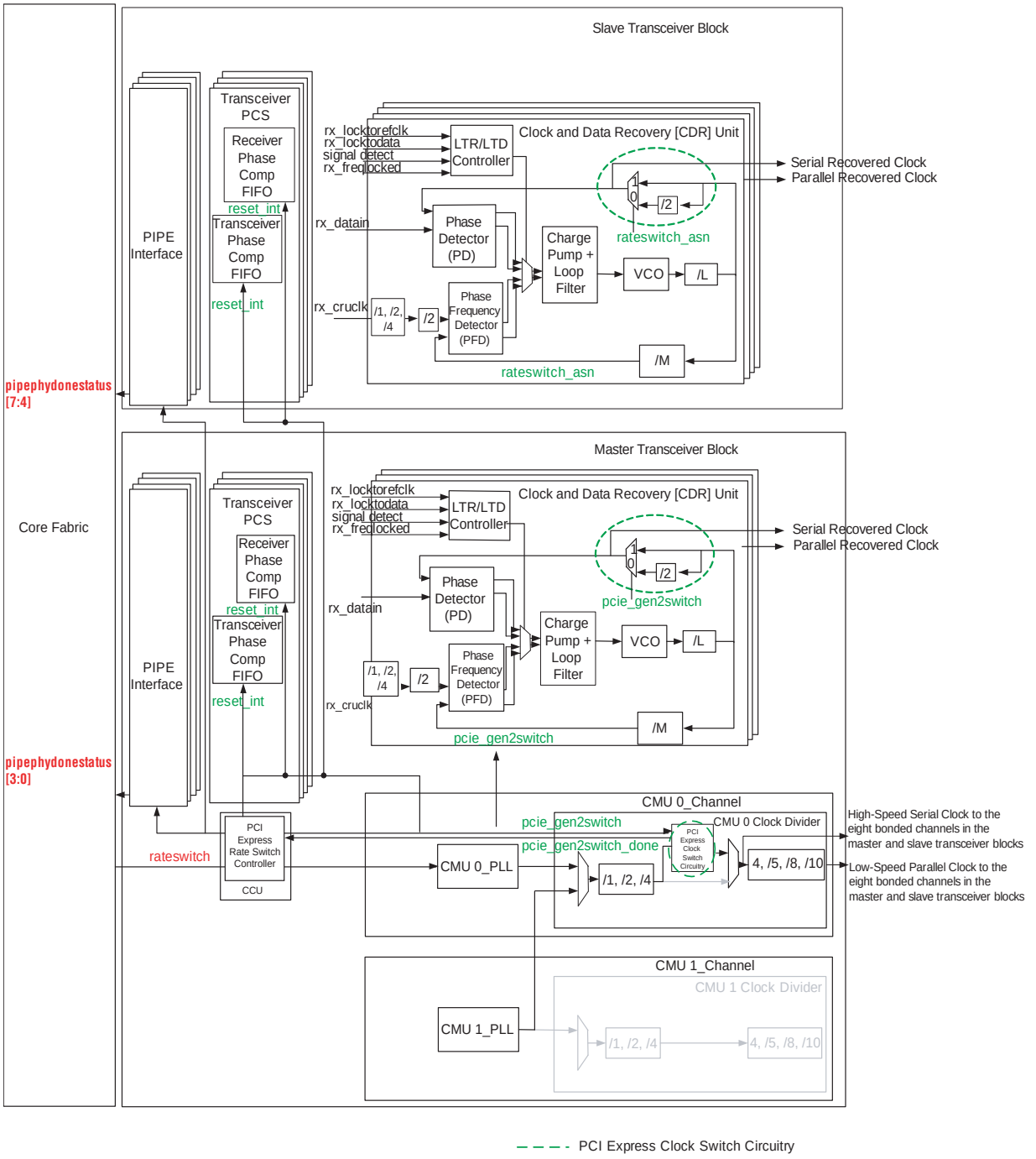


As a result of the signaling rateswitch between Gen1 (2.5 Gbps) and Gen2 (5 Gbps), the core fabric-transceiver interface clock switches between 125 MHz and 250 MHz. The core fabric-transceiver interface clock clocks the read side and write side of the transmitter phase compensation FIFO and the receiver phase compensation FIFO of all bonded channels, respectively. It is also routed to the core fabric on a global or regional clock resource and looped back to clock the write port and read port of the transmitter phase compensation FIFO and the receiver phase compensation FIFO, respectively. Due to the routing delay between the write and read clock of the transmitter and receiver phase compensation FIFOs, the write pointers and read pointers might collide during a rateswitch between 125 MHz and 250 MHz. To avoid collision of the phase compensation FIFO pointers, the PCI Express (PIPE) rateswitch controller automatically disables and resets the phase compensation FIFO pointers of all bonded channels during clock switch. When the PIPE clock switch circuitry in the local clock divider indicates successful clock switch completion, the PIPE rateswitch controller releases the phase compensation FIFO pointer resets.

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

Figure 1–117 shows the PCI Express (PIPE) rateswitch circuitry in PIPE ×8 mode configured at Gen2 (5 Gbps) data rate.

Figure 1–117. Dynamic Switch Signaling in PCI Express (PIPE) ×8 Mode



In PCI Express (PIPE) ×8 mode configured at 5 Gbps data rate, when the PIPE rateswitch controller sees a transition on the `rateswitch` signal, it sends the `pcie_gen2switch` control signal to the PIPE clock switch circuitry in the `CMU0` clock divider of the master transceiver block and the receiver CDR in all eight bonded channels to switch to the instructed signaling rate. A low-to-high transition on the `rateswitch` signal initiates a Gen1 (2.5 Gbps) to Gen2 (5 Gbps) signaling rateswitch. A high-to-low transition on the `rateswitch` signal initiates a Gen2 (5 Gbps) to Gen1 (2.5 Gbps) signaling rateswitch.

Table 1–40 shows the transceiver clock frequencies when switching between 2.5 Gbps and 5 Gbps signaling rates.

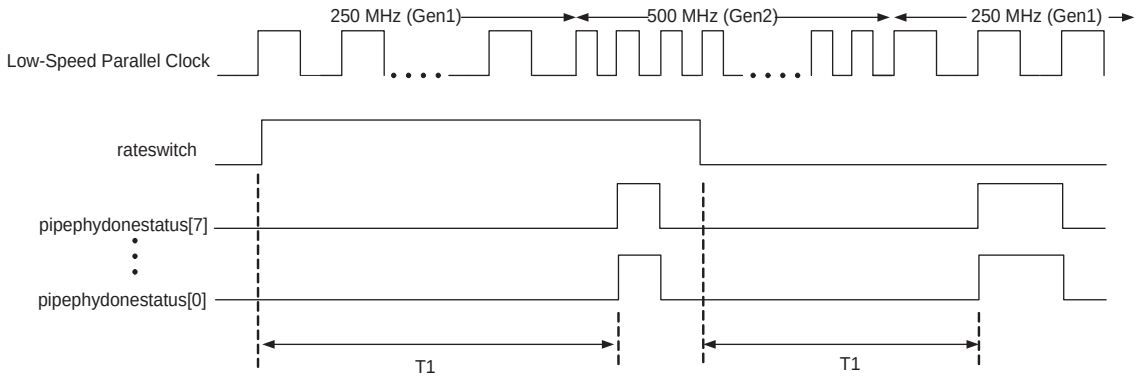
Table 1–40. Transceiver Clock Frequencies Signaling Rates in PCI Express (PIPE) ×8 Mode

Transceiver Clocks	Gen1 (2.5 Gbps) to Gen 2 (5 Gbps) Switch (Low-to-High Transition on the rateswitch Signal)	Gen2 (5 Gbps) to Gen1 (2.5 Gbps) Switch (High-to-Low Transition on the rateswitch Signal)
High-Speed Serial Clock	1.25 GHz to 2.5 GHz	2.5 GHz to 1.25 GHz
Low-Speed Parallel Clock	250 MHz to 500 MHz	500 MHz to 250 MHz
Serial Recovered Clock	1.25 GHz to 2.5 GHz	2.5 GHz to 1.25 GHz
Parallel Recovered Clock	250 MHz to 500 MHz	500 MHz to 250 MHz
Core Fabric-Transceiver Interface Clock	125 MHz to 250 MHz	250 MHz to 125 MHz

The PIPE clock switch circuitry in the `CMU0` clock divider of the master transceiver block performs the clock switch between 250 MHz and 500 MHz on the low-speed parallel clock when switching between Gen1 (2.5 Gbps) and Gen2 (5 Gbps) signaling rates. It indicates successful completion of clock switch on the `pcie_gen2switchdone` signal to the PIPE rateswitch controller. The PIPE rateswitch controller forwards the clock switch completion status to the PIPE interface block. The PIPE interface block communicates the clock switch completion status to the PHY-MAC layer by asserting the `pipephydonestatus` signal of all eight bonded channels for one parallel clock cycle.

Figure 1–118 shows the low-speed parallel clock switch between Gen1 (250 MHz) and Gen2 (500 MHz) in response to the change in the logic level on the `rateswitch` signal. The rateswitch completion is shown marked with a one clock cycle assertion of the `pipephydonestatus` signal of all eight bonded channels.

Figure 1–118. Low-Speed Parallel Clock Switching in PCI Express (PIPE) ×8 Mode *(Note 1)*



Note to Figure 1–118:

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

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

PCI Express (PIPE) Cold Reset Requirements

The PIPE Base Specification 2.0 defines the following three types of conventional resets to the PIPE system components:

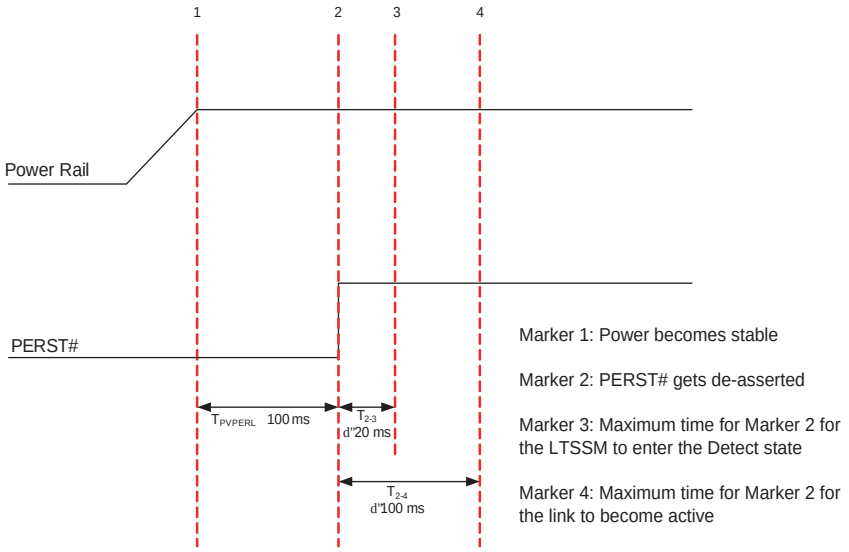
- Cold reset—fundamental reset after power up
- Warm reset—fundamental reset without removal and re-application of power
- Hot reset—In-band conventional reset initiated by the higher layer by setting the Hot Reset bit in the TS1 or TS2 training sequences

Fundamental reset is provided by the system to the component or adapter card using the auxiliary signal PERST#. The PIPE Base Specification 2.0 specifies that PERST# must be kept asserted for a minimum of 100 ms (TPVPERL) after the system power becomes stable in a cold reset situation. Additionally, all system components must enter the LTSSM Detect state within 20 ms and the link must become active within 100 ms after de-assertion of the PERST# signal. This implies that each PIPE system component must become active within 200 ms after the power becomes stable.

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

Figure 1–119 shows the PCI Express (PIPE) cold reset timing requirements.

Figure 1–119. PCI Express (PIPE) Cold Reset Requirements



The time taken by a PCI Express (PIPE) port implemented using the HardCopy IV GX device to go from power up to link active state is described below:

- Power-on reset—begins after power rails become stable. Typically takes 12 ms
- Time taken from de-assertion of PERST# to link active—typically takes 40 ms (pending characterization and verification of PIPE soft IP and hard IP)

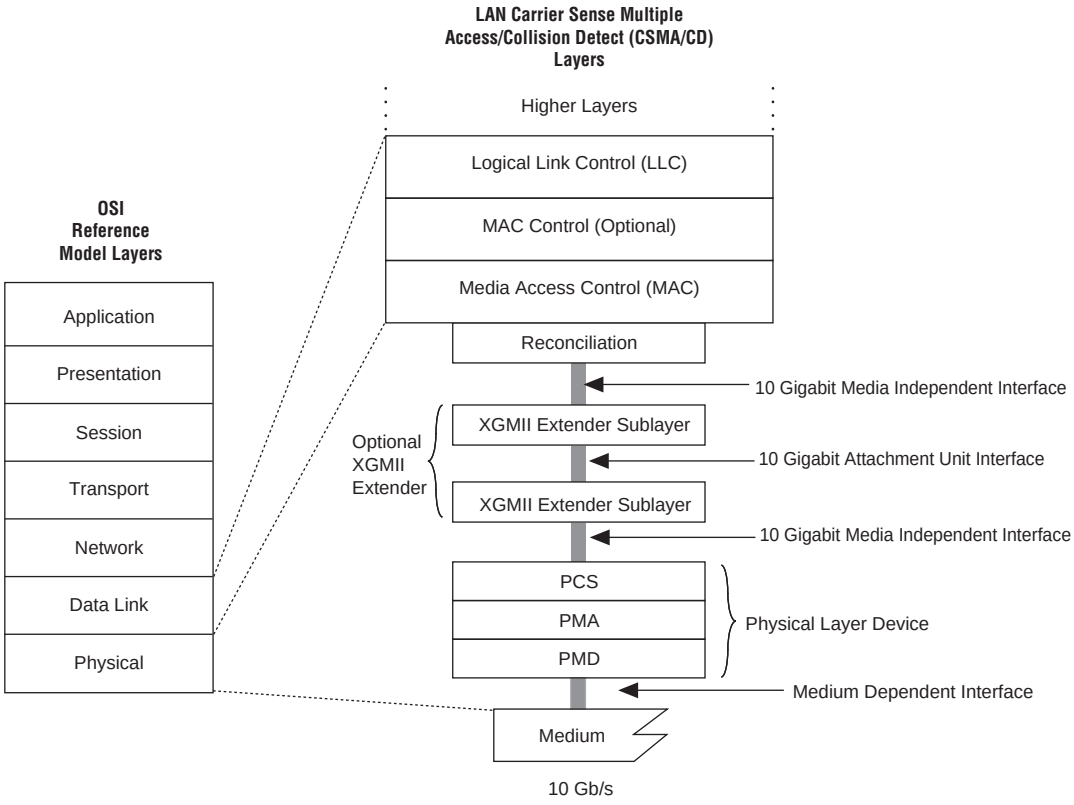
XAUI Mode

XAUI is an optional, self-managed interface that you can insert between the reconciliation sublayer and the PHY layer to transparently extend the physical reach of the XGMII.

XAUI addresses several physical limitations of the XGMII. XGMII signaling is based on the HSTL Class 1 single-ended I/O standard, which has an electrical distance limitation of approximately 7 cm. Because XAUI uses a low-voltage differential signaling method, the electrical limitation is increased to approximately 50 cm. Another advantage of XAUI is simplification of backplane and board trace routing. XGMII is composed of 32 transmit channels, 32 receive channels, 1 transmit clock, 1 receive clock, 4 transmitter control characters, and 4 receive control characters for a 74-pin wide interface. XAUI, on the other hand, only consists of 4 differential transmitter channels and 4 differential receiver channels for a 16-pin wide interface. This reduction in pin count significantly simplifies the routing process in the layout design.

Figure 1–120 shows the relationships between the XGMII and XAUI layers.

Figure 1–120. XAUI and XGMII Layers



The XGMII interface consists of four lanes of 8 bits. At the transmit side of the XAUI interface, the data and control characters are converted within the XGXS into an 8B/10B encoded data stream. Each data stream is then transmitted across a single differential pair running at 3.125 Gbps (3.75 Gbps for HiGig). At the XAUI receiver, the incoming data is decoded and mapped back to the 32-bit XGMII format. This provides a transparent extension of the physical reach of the XGMII and also reduces the interface pin count.

In HardCopy IV GX XAUI functional mode, the interface between the transceiver and core 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 1–41.

Table 1–41. XGMII Character to PCS Code-Group Mapping (Part 1 of 2)

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||

Table 1–41. XGMII Character to PCS Code-Group Mapping (Part 2 of 2)

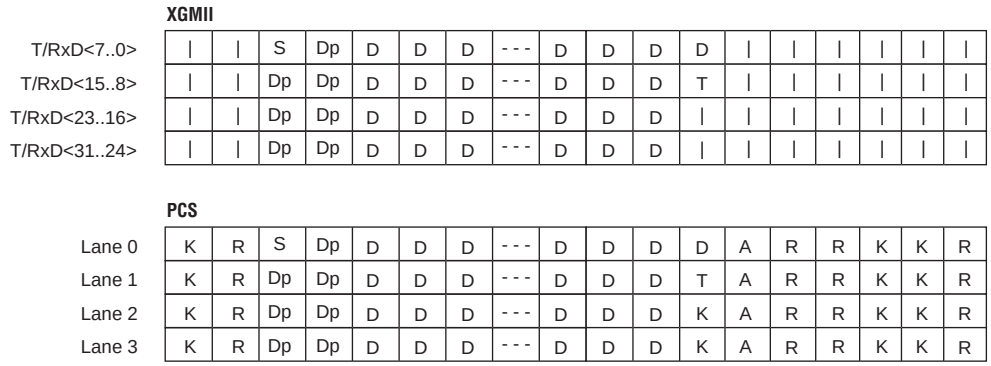
XGMII TXC	XGMII TXD *(1)*	PCD Code Group	Description
1	07	K28.5	Idle in ||T||
1	9C	K28.4	Sequence
1	FB	K27.7	Start
1	FD	K29.7	Terminate
1	FE	K30.7	Error
1	Any other value	K30.7	Invalid XGMII character

Note to Table 1–41:

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

Figure 1–121 shows an example of mapping between XGMII characters and the PCS code groups that are used in XAUI. The idle characters are mapped to a pseudo-random sequence of /A/, /R/, and /K/ code groups.

Figure 1–121. Example of Mapping XGMII Characters to PCS Code Groups



PCS code groups are sent via PCS ordered sets. PCS ordered sets consist of combinations of special and data code groups defined as a column of code groups. These ordered sets are composed of four code groups beginning in lane 0. Table 1–42 lists the defined idle ordered sets (||I||) that are used for the self-managed properties of XAUI.

Table 1–42. Defined Idle Ordered Set

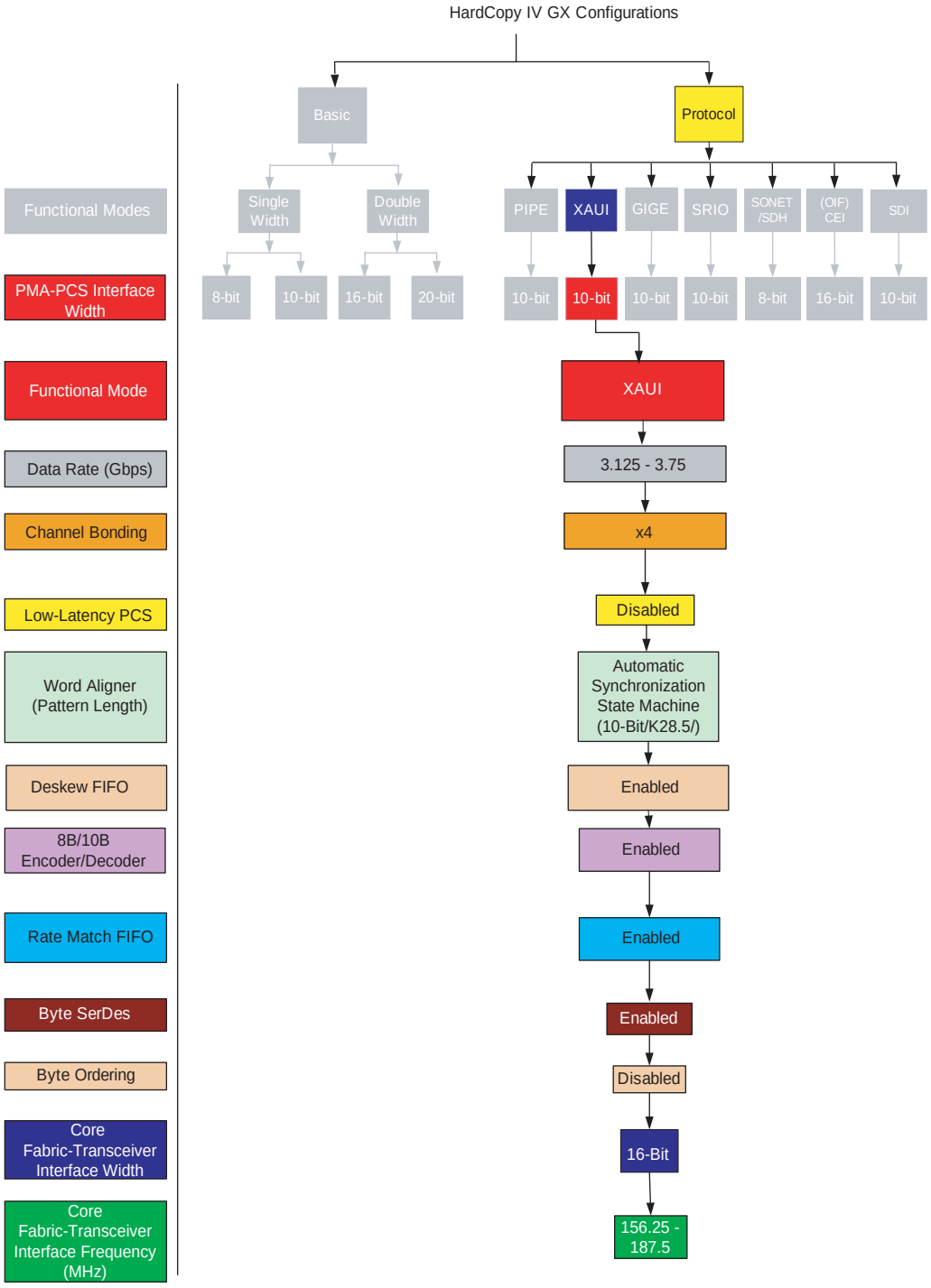
Code	Ordered Set	Number of Code Groups	Encoding
||I||	**Idle**		**Substitute for XGMII Idle**
||K||	Synchronization column	4	/K28.5/K28.5/K28.5/K28.5/
||R||	Skip column	4	/K28.0/K28.0/K28.0/K28.0/
||A||	Align column	4	/K28.3/K28.3/K28.3/K28.3/

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 1–122 shows the XAUI mode configuration supported in HardCopy IV GX devices.

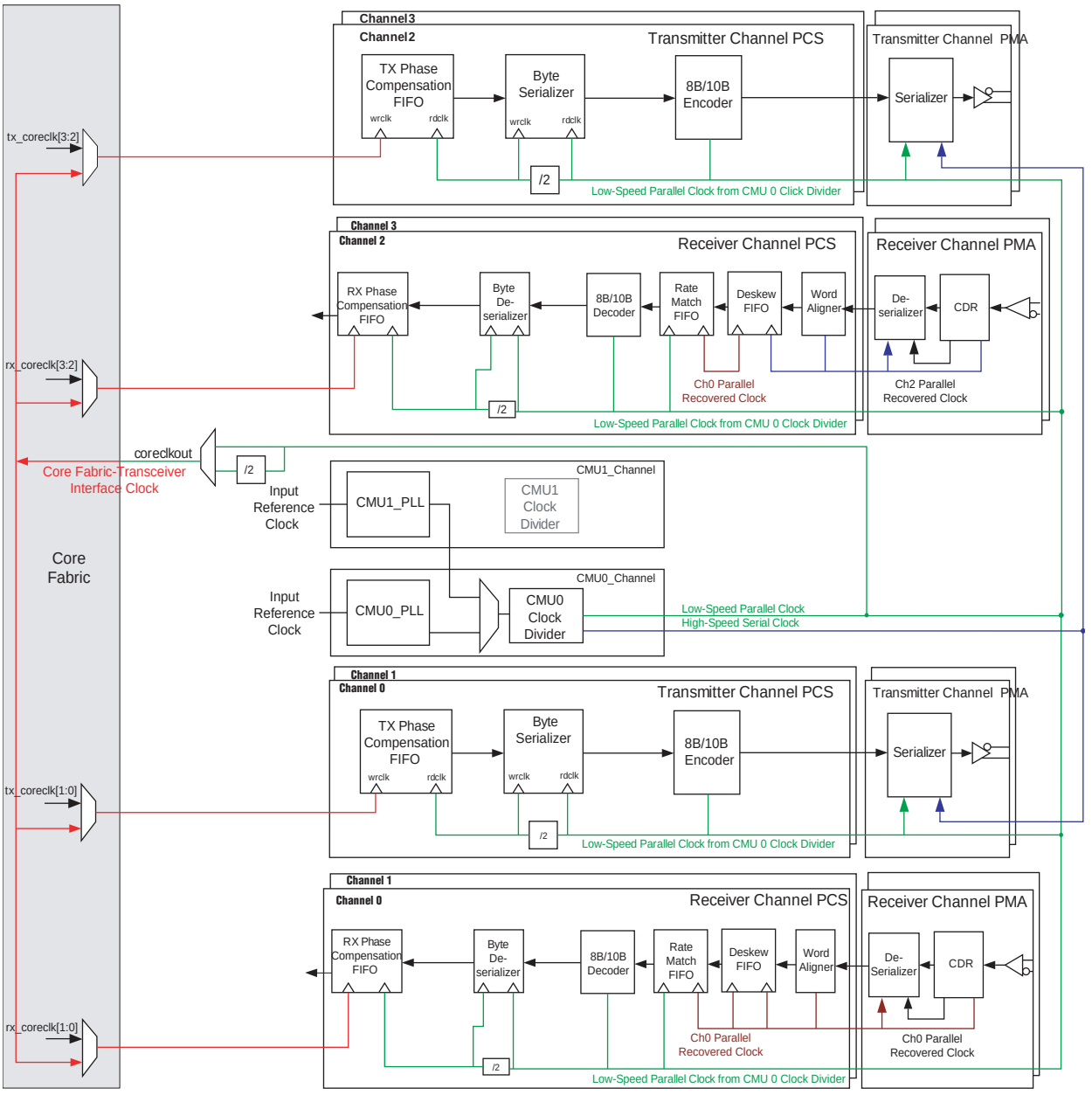
Figure 1–122. HardCopy IV GX XAUI Mode Configuration



XAUI Mode Datapath

Figure 1–123 shows the ALTGX megafunction transceiver datapath when configured in XAUI mode.

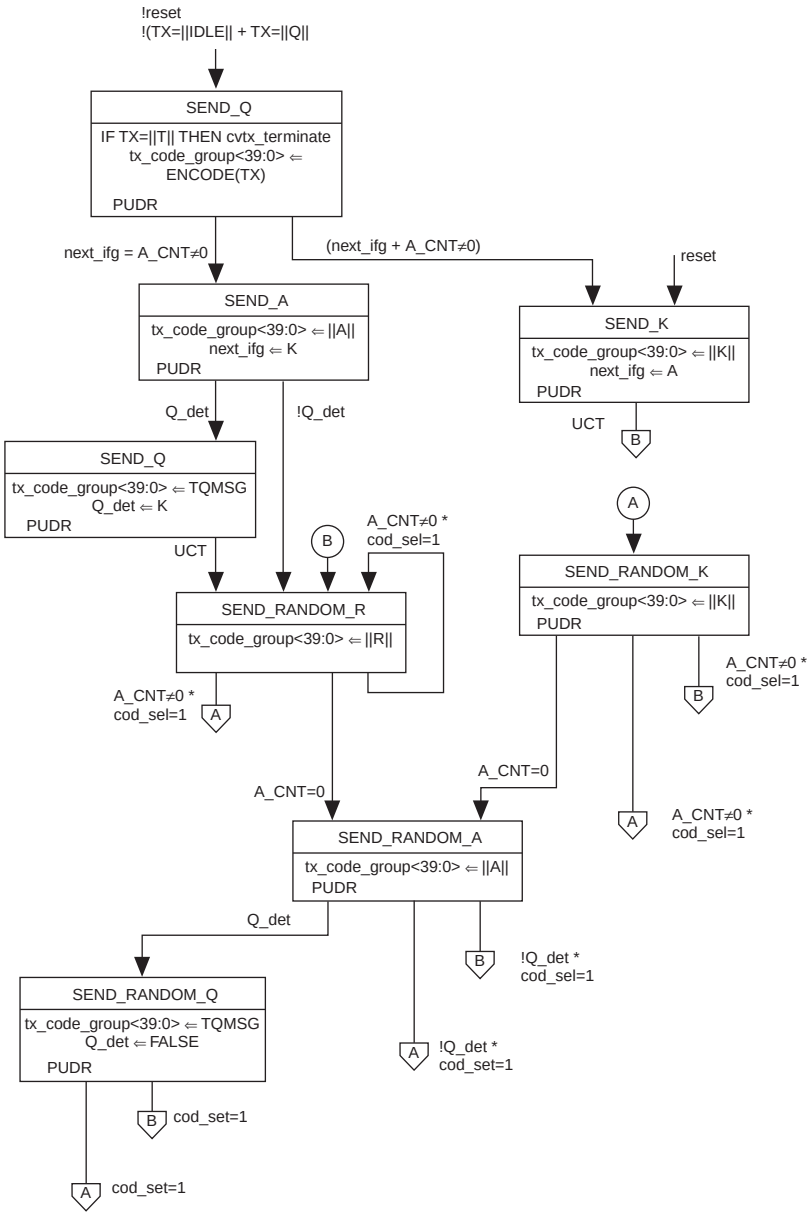
Figure 1–123. Transceiver Datapath in XAUI Mode



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 shown in Figure 1–124.

Figure 1–124. XGMII-To-PCS Code Conversion in XAUI Mode *(Note 1)*



Note to Figure 1–124:

(1) This figure is from IEEE P802.3ae.

Table 1–43 lists the XGMII-to-PCS code group conversion in XAUI functional mode. The XGMII TXC control signal is equivalent to the `tx_ctrlenable` signal; the XGMII TXD control signal is equivalent to the `tx_datain[7:0]` signal.

Table 1–43. 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 1–41:

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

PCS-To-XGMII Code Conversion at the Receiver

In XAUI mode, the 8B/10B decoder in the HardCopy IV GX receiver datapath is controlled by a XAUI receiver state machine that converts received PCS code groups into specific 8-bit XGMII codes. This state machine complies with the IEEE P802.3ae specifications.

Table 1–44 lists the PCS-to-XGMII code group conversion in XAUI functional mode. The XGMII RXC control signal is equivalent to the `rx_ctrldetect` signal; the XGMII RXD control signal is equivalent to the `rx_dataout[7:0]` signal.

Table 1–44. PCS Code Group to XGMII Character Mapping

XGMII RXC	XGMII RXD *(1)*	PCD Code Group	Description
0	00 through FF	Dxx,y	Normal data transmission
1	07	K28.0 or K28.3 or K28.5	Idle in ||I||
1	07	K28.5	Idle in ||T||
1	9C	K28.4	Sequence
1	FB	K27.7	Start
1	FD	K29.7	Terminate
1	FE	K30.7	Error
1	FE	Invalid code group	Received code group

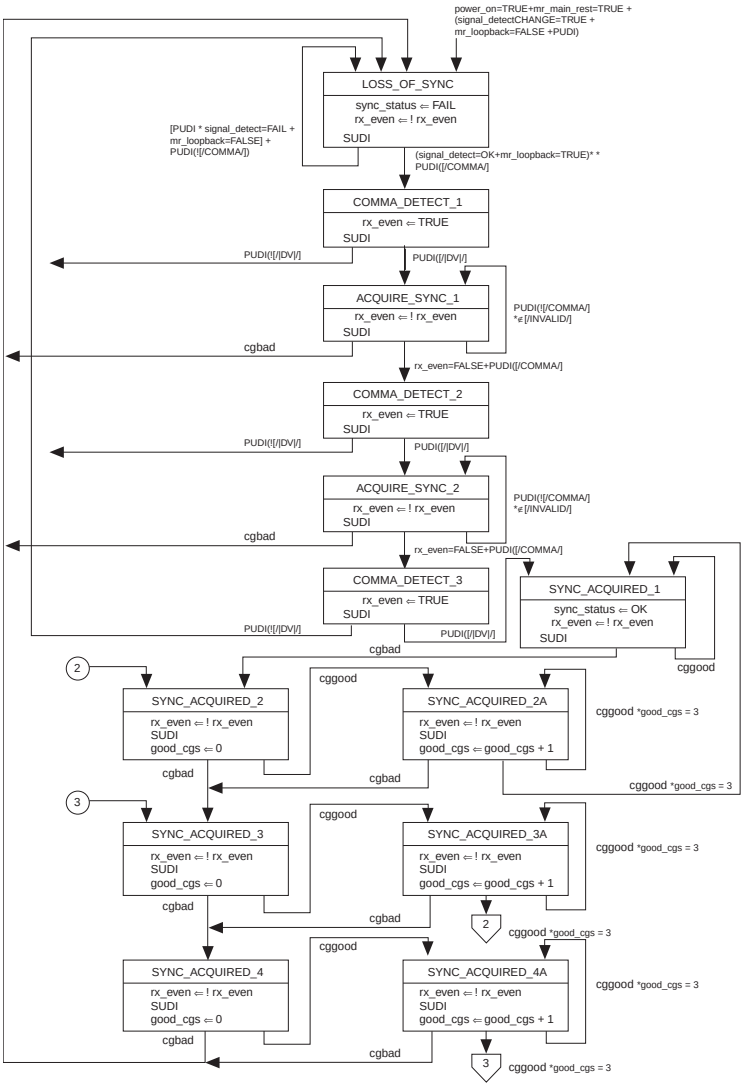
Note to Table 1–41:

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

Word Aligner

The word aligner in XAUI functional mode is configured in automatic synchronization state machine mode. The Quartus II software automatically configures the synchronization state machine to indicate synchronization when the receiver receives four /K28.5/ comma code groups without intermediate invalid code groups. The synchronization state machine implemented in XAUI mode is compliant to the PCS synchronization state diagram specified in Clause 48 of the IEEE P802.3ae specification and is shown in Figure 1–125.

Figure 1–125. IEEE 802.3ae PCS Synchronization State Diagram *(Note 1)*



Note to Figure 1–125:

(1) This figure is from IEEE P802.3ae.

Receiver synchronization is indicated on the rx_syncstatus port of each channel. A high on the rx_syncstatus port indicates that the lane is synchronized; a low on the rx_syncstatus port indicates that it has fallen out of synchronization. The receiver loses synchronization when it detects four invalid code groups separated by less than four valid code groups or when it is reset.

Deskew FIFO

Code groups received across four lanes in a XAUI link can be misaligned with respect to one another because of skew in the physical medium or differences between the independent clock recoveries per lane. The XAUI protocol allows a maximum skew of 40 UI (12.8 ns) as seen at the receiver of the four lanes.

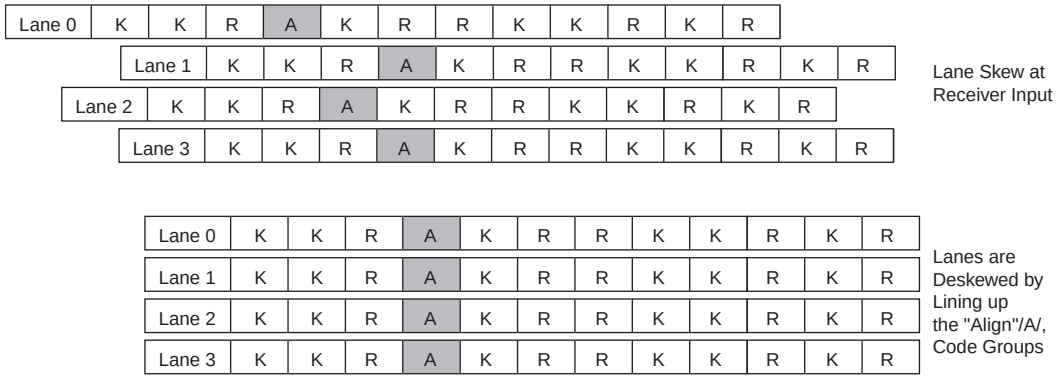
The XAUI protocol requires the physical layer device to implement a deskew circuitry to align all four channels. To enable the deskew circuitry at the receiver to align the four channels, the transmitter sends a /A/ (/K28.3/) code group simultaneously on all four channels during inter-packet gap. The skew introduced in the physical medium and the receiver channels can be /A/ code groups to be received misaligned with respect to each other.

The deskew operation is performed by the deskew FIFO in XAUI functional mode.

The deskew FIFO in each channel receives data from its word aligner. The deskew operation begins only after link synchronization is achieved on all four channels as indicated by a high on the rx_syncstatus signal from the word aligner in each channel. Until the first /A/ code group is received, the deskew FIFO read and write pointers in each channel are not incremented. After the first /A/ code group is received, the write pointer starts incrementing for each word received but the read pointer is frozen. If the /A/ code group is received on each of the four channels within 10 recovered clock cycles of each other, the read pointer of all four deskew FIFOs is released simultaneously, aligning all four channels.

Figure 1–126 shows lane skew at the receiver input and how the deskew FIFO uses the /A/ code group to align the channels.

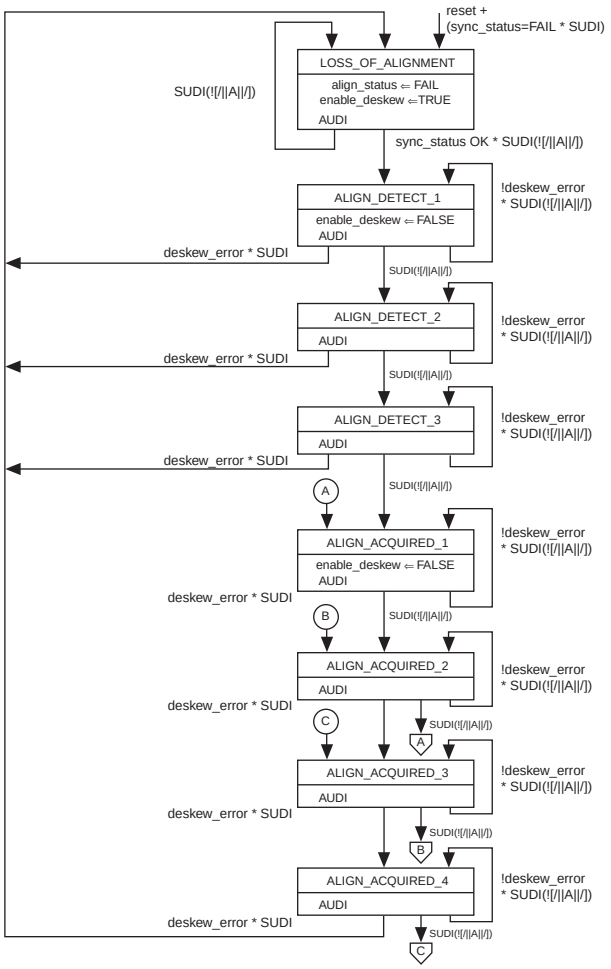
Figure 1–126. Receiver Input Lane Skew in XAUI Mode



After alignment of the first ||A|| column, if three additional aligned ||A|| columns are observed at the output of the deskew FIFOs of the four channels, the `rx_channelaligned` signal is asserted high, indicating channel alignment is acquired. After acquiring channel alignment, if four misaligned ||A|| columns are seen at the output of the deskew FIFOs in all four channels with no aligned ||A|| columns in between, the `rx_channelaligned` signal is de-asserted low, indicating loss-of-channel alignment.

The deskew FIFO operation in XAUI functional mode is compliant with the PCS deskew state machine diagram specified in clause 48 of the IEEE P802.3ae, as shown in Figure 1–127.

Figure 1–127. Deskew FIFO in XAUI Mode *(Note 1)*



Note to Figure 1–127:

(1) This figure is from IEEE P802.3ae.

Rate Match FIFO

In XAUI mode, the rate match FIFO is capable of compensating for up to ±100 PPM (200 PPM total) difference between the upstream transmitter and the local receiver reference clock. The XAUI protocol requires the transmitter to send /R/ (/K28.0/) code groups simultaneously on all four lanes (denoted as ||R|| column) during inter-packet gaps, adhering to rules listed in the IEEE P802.3ae specification. The rate match FIFO operation in XAUI mode is compliant to the IEEE P802.3ae specification.

The rate match operation begins after:

- The synchronization state machine in the word aligner of all four channels indicates synchronization has been acquired by driving the `rx_syncstatus` signal high
- The deskew FIFO indicates alignment has been acquired by driving the `rx_channelaligned` signal high

The rate match FIFO looks for the ||R|| column (simultaneous /R/ code group on all four channels) and deletes or inserts ||R|| column to prevent the rate match FIFO from overflowing or under-running. The rate match FIFO can insert or delete as many ||R|| columns as necessary to perform the rate match operation.

Two flags, `rx_rmfifodatadeleted` and `rx_rmfifodatainserted`, indicating rate match FIFO deletion and insertion events, respectively, are forwarded to the core fabric. If an ||R|| column is deleted, the `rx_rmfifodeleted` flag from each of the four channels goes high for one clock cycle per deleted ||R|| column. If an ||R|| column is inserted, the `rx_rmfifoinserted` flag from each of the four channels goes high for one clock cycle per inserted ||R|| column.

Figure 1–128 shows an example of rate match deletion in the case where three ||R|| columns are required to be deleted.

Figure 1–128. Rate Match Deletion in XAUI Mode

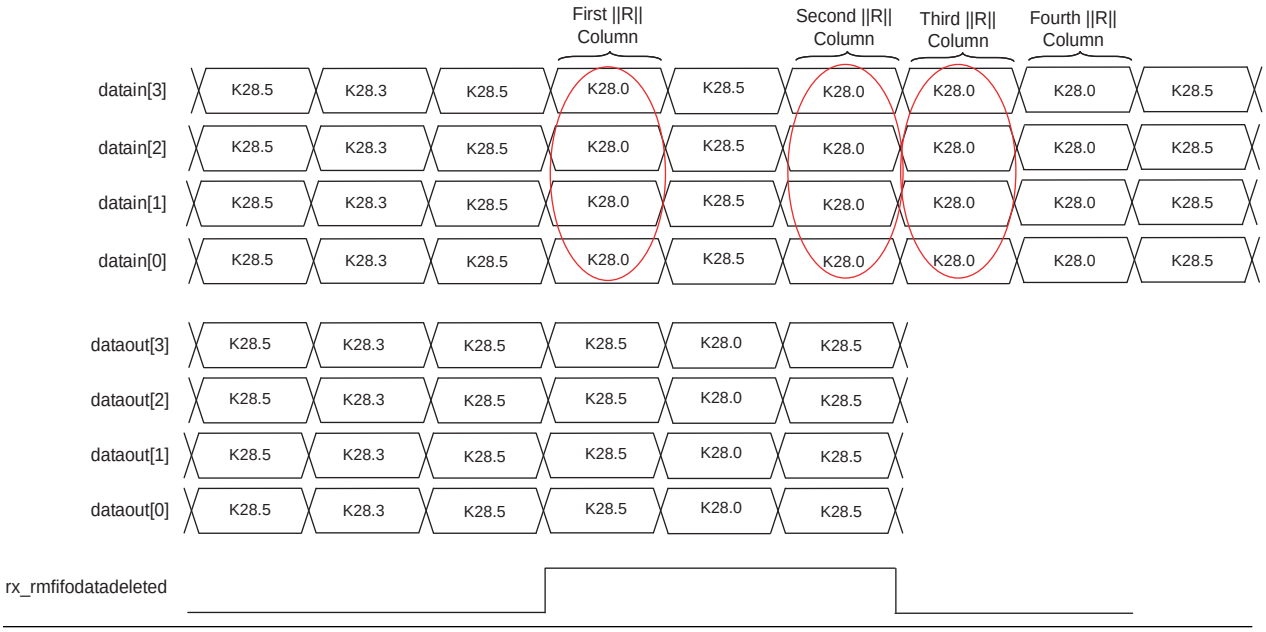


Figure 1–129 shows an example of rate match insertion in the case where two ||R|| columns are required to be inserted.

Figure 1–129. Rate Match Insertion in XAUI Mode

GIGE Mode

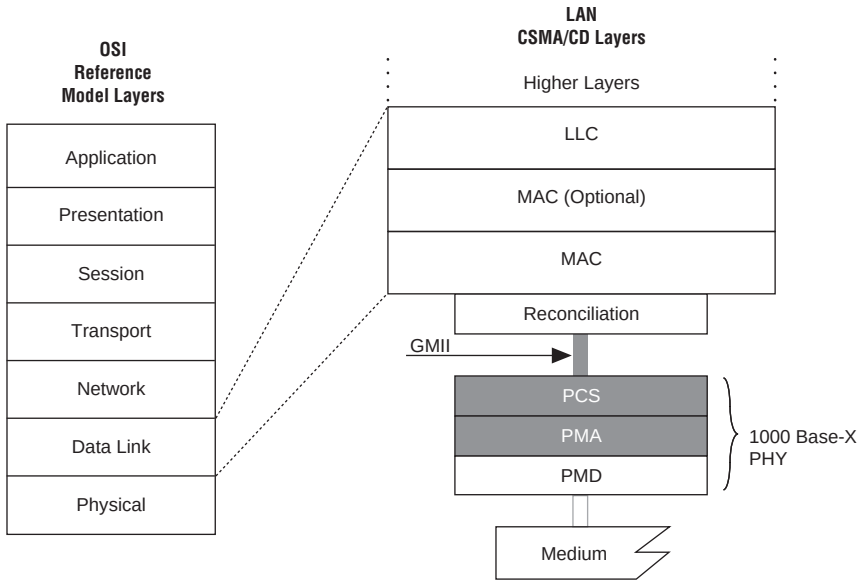
IEEE 802.3 defines the 1000 Base-X PHY as an intermediate, or transition, layer that interfaces various physical media with the media access control (MAC) in a gigabit ethernet system. It shields the MAC layer from the specific nature of the underlying medium. The 1000 Base-X PHY is divided into three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

- Physical coding sublayer
- Physical media attachment
- Physical medium dependent (PMD)

The PCS sublayer interfaces with the MAC through the gigabit medium independent interface (GMII). The 1000 Base-X PHY defines a physical interface data rate of 1 Gbps.

Figure 1–130 shows the 1000 Base-X PHY position in a Gigabit Ethernet OSI reference model.

Figure 1–130. 1000 Base-X PHY in a Gigabit Ethernet OSI Reference Model



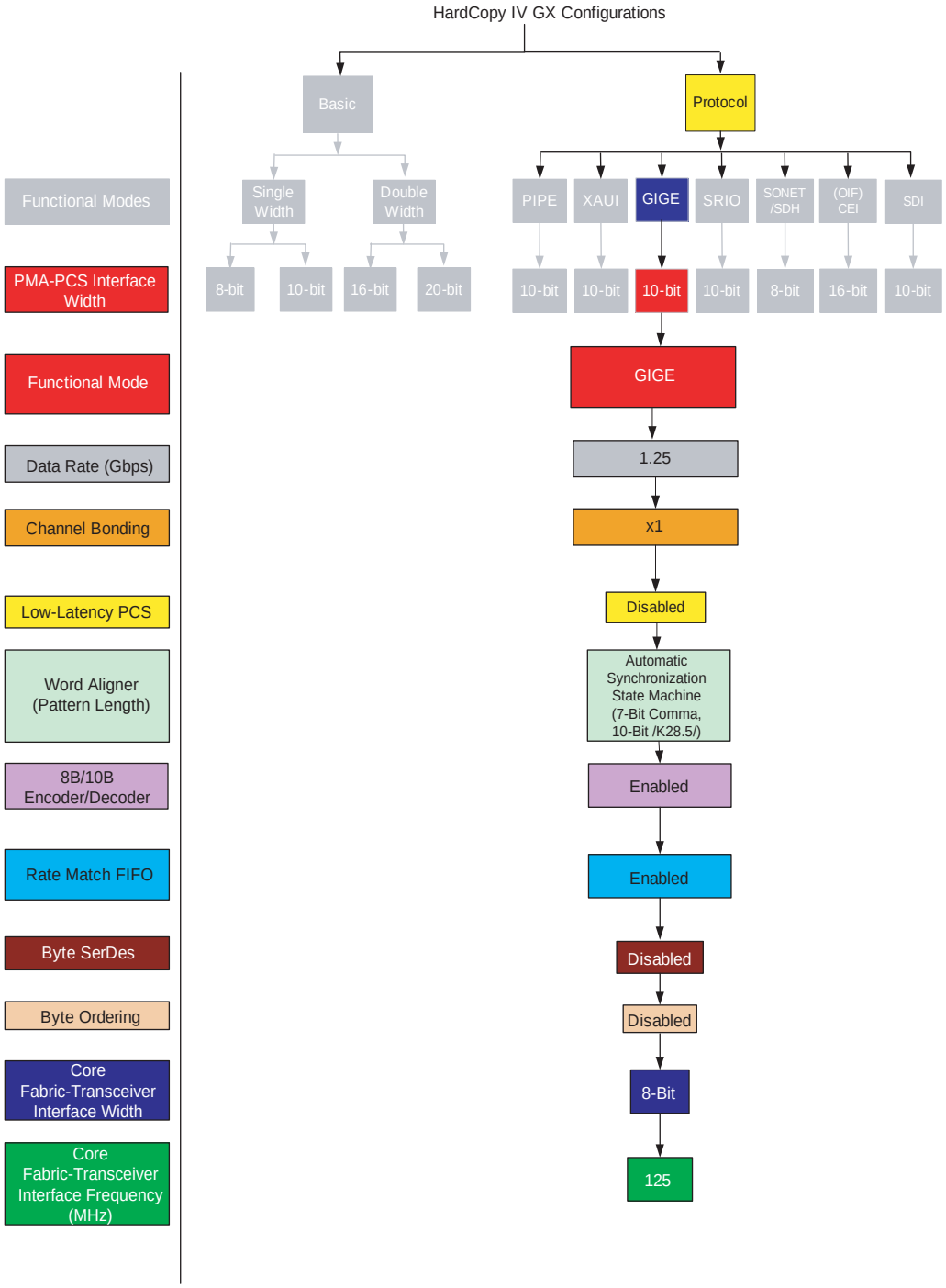
HardCopy IV GX transceivers, when configured in GIGE functional mode, have built-in circuitry to support the following PCS and PMA functions defined in the IEEE 802.3 specification:

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

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 1–131 shows the GIGE mode configuration supported in HardCopy IV GX devices.

Figure 1–131. GIGE Mode



GIGE Mode Datapath

Figure 1–132 shows the transceiver datapath when configured in GIGE functional mode.

Figure 1–132. GIGE Mode Datapath

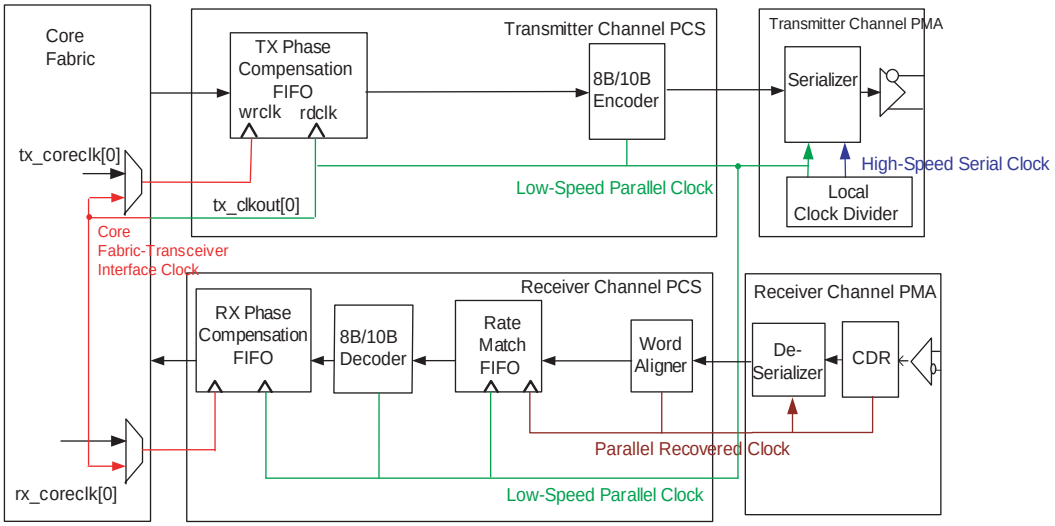


Table 1–45 shows the transceiver datapath clock frequencies in GIGE functional mode.

Table 1–45. 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	Core 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 1–40.

GIGE Protocol—Ordered Sets and Special Code Groups

Table 1–46 lists ordered sets and special code groups specified in the IEEE 802.3 specification.

Table 1–46. GIGE Ordered Sets (Part 1 of 2)

Code	Ordered Set	Number of Code Groups	Encoding
/C/	Configuration	—	Alternating /C1/ and /C2/
/C1/	Configuration 1	4	/K28.5/D21.5/Config_Reg *(1)*
/C2/	Configuration 2	4	/K28.5/D2.2/Config_Reg *(1)*
/I/	IDLE	—	Correcting /I1/, Preserving /I2/

Table 1–46. GIGE Ordered Sets (Part 2 of 2)

Code	Ordered Set	Number of Code Groups	Encoding
/I1/	IDLE 1	2	/K28.5/D5.6
/I2/	IDLE 2	2	/K28.5/D16.2
	Encapsulation	—	—
/R/	`Carrier_Extend`	1	/K23.7/
/S/	`Start_of_Packet`	1	/K27.7/
/T/	`End_of_Packet`	1	/K29.7/
/V/	`Error_Propagation`	1	/K30.7/

Note to Table 1–46:

(1) Two data code groups representing the `Config_Reg` value.

Idle Ordered-Set Generation

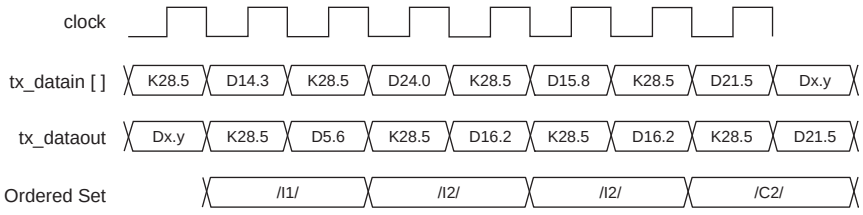
The IEEE 802.3 specification requires the GIGE PHY to transmit idle ordered sets (/I/) continuously and repetitively whenever the GMII is idle. This ensures that the receiver maintains bit and word synchronization whenever there is no active data to be transmitted.

In GIGE functional mode, any /Dx.y/ following a /K28.5/ comma is replaced by the transmitter with either a /D5.6/ (/I1/ ordered set) or a /D16.2/ (/I2/ ordered set), depending on the current running disparity. The exception is when the data following the /K28.5/ is /D21.5/ (/C1/ ordered set) or /D2.2/ (/C2/) ordered set. If the running disparity before the /K28.5/ is positive, an /I1/ ordered set is generated. If the running disparity is negative, a /I2/ ordered set is generated. The disparity at the end of a /I1/ is the opposite of that at the beginning of the /I1/. The disparity at the end of a /I2/ is the same as the beginning running disparity (right before the idle code). This ensures a negative running disparity at the end of an idle ordered set. A /Kx.y/ following a /K28.5/ is not replaced.

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

Figure 1–133 shows the automatic idle ordered set generation.

Figure 1–133. Automatic Ordered Set Generation



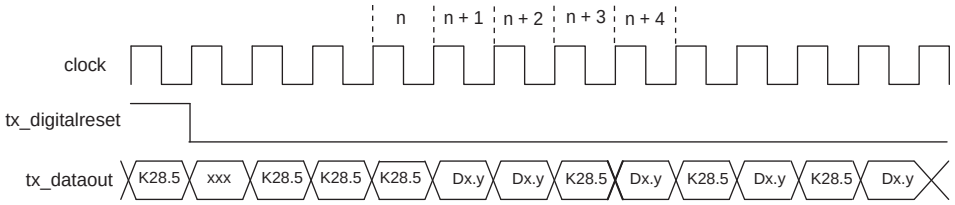
Reset Condition

After de-assertion of `tx_digitalreset`, the GIGE transmitter automatically transmits three /K28.5/ comma code groups before transmitting user data on the `tx_datain` port. This could affect the synchronization state machine behavior at the receiver.

Depending on when you start transmitting the synchronization sequence, there could be an even or odd number of /Dx.y/ code groups transmitted between the last of the three automatically sent /K28.5/ code groups and the first /K28.5/ code group of the synchronization sequence. If there is an even number of /Dx.y/ code groups received between these two /K28.5/ code groups, the first /K28.5/ code group of the synchronization sequence begins at an odd code group boundary (`rx_even` = FALSE). An IEEE802.3-compliant GIGE synchronization state machine treats this as an error condition and goes into the Loss of Sync state.

Figure 1–134 shows an example of even numbers of /Dx.y/ between the last automatically sent /K28.5/ and the first user-sent /K28.5/. The first user-sent /K28.5/ code group received at an odd code group boundary in cycle n + 3 takes the receiver synchronization state machine in the Loss of Sync state. The first synchronization ordered set /K28.5/Dx.y/ in cycles n + 3 and n + 4 is discounted and three additional ordered sets are required for successful synchronization.

Figure 1–134. Reset Condition in GIGE Mode



Word Aligner

The word aligner in GIGE functional mode is configured in automatic synchronization state machine mode. The Quartus II software automatically configures the synchronization state machine to indicate synchronization when the receiver receives three consecutive synchronization ordered sets. A synchronization ordered set is a /K28.5/ code group followed by an odd number of valid /Dx.y/ code groups. The fastest way for the receiver to achieve synchronization is to receive three continuous {/K28.5/, /Dx.y/} ordered sets.

Receiver synchronization is indicated on the `rx_syncstatus` port of each channel. A high on the `rx_syncstatus` port indicates that the lane is synchronized; a low on the `rx_syncstatus` port indicates that the lane has fallen out of synchronization. The receiver loses synchronization when it detects four invalid code groups separated by less than three valid code groups or when it is reset.

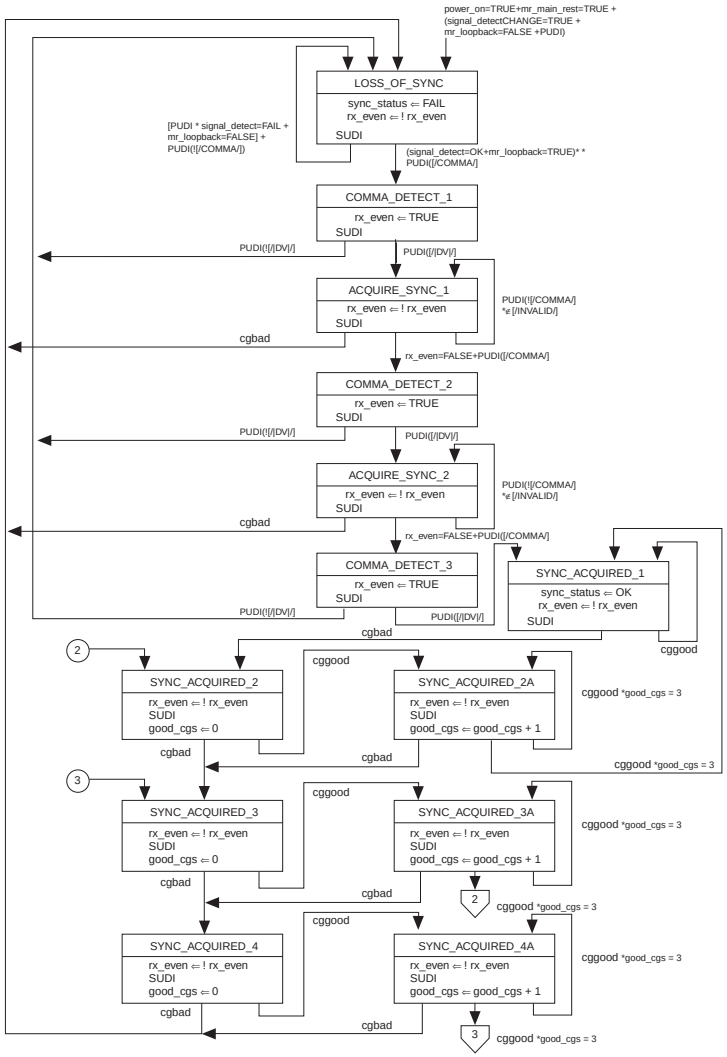
Table 1–47 lists the synchronization state machine parameters when configured in GIGE mode.

Table 1–47. Synchronization State Machine Parameters in GIGE Functional Mode

Synchronization State Machine Parameters	Setting
Number of valid {/K28.5/, /Dx,y/} ordered sets received to achieve synchronization	3
Number of errors received to lose synchronization	4
Number of continuous good code groups received to reduce the error count by 1	4

Figure 1–135 shows the synchronization state machine implemented in GIGE mode.

Figure 1–135. Synchronization State Machine in GIGE Mode *(Note 1)*



Note to Figure 1–135:

(1) This figure is from IEEE P802.3ae.

Rate Match FIFO

In GIGE mode, the rate match FIFO is capable of compensating for up to ±100 PPM (200 PPM total) difference between the upstream transmitter and the local receiver reference clock. The GIGE protocol requires the transmitter to send idle ordered sets /I1/ (/K28.5/D5.6/) and /I2/ (/K28.5/D16.2/) during inter-packet gaps adhering to the rules listed in the IEEE 802.3 specification.

The rate match operation begins after the synchronization state machine in the word aligner indicates synchronization is acquired by driving the `rx_syncstatus` signal high. The rate matcher deletes or inserts both symbols (/K28.5/ and /D16.2/) of the /I2/ ordered sets, even if it requires deleting only one symbol to prevent the rate match FIFO from overflowing or under-running. It can insert or delete as many /I2/ ordered sets as necessary to perform the rate match operation.

Two flags, rx_rmfifodatadeleted and rx_rmfifodatainserted, indicating rate match FIFO deletion and insertion events, respectively, are forwarded to the core fabric. Both the rx_rmfifodatadeleted and rx_rmfifodatainserted flags are asserted for two clock cycles for each deleted and inserted /I2/ ordered set, respectively.

Figure 1–136 shows an example of rate match FIFO deletion in the case where three symbols are required to be deleted. Because the rate match FIFO can only delete /I2/ ordered set, it deletes two /I2/ ordered sets (four symbols deleted).

Figure 1–136. Rate Match Deletion in GIGE Mode

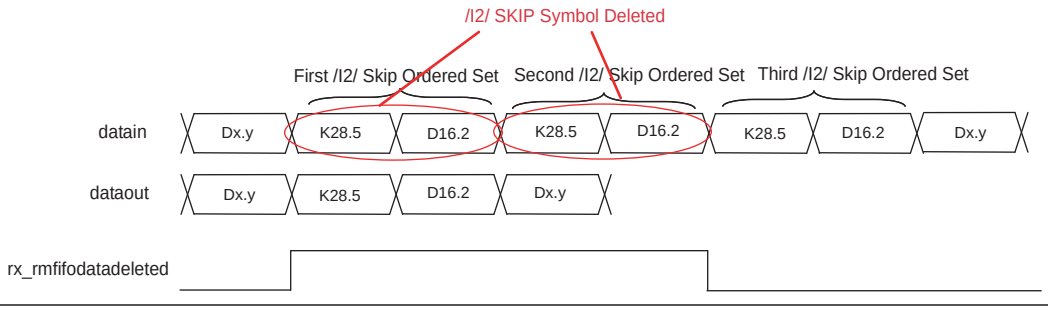
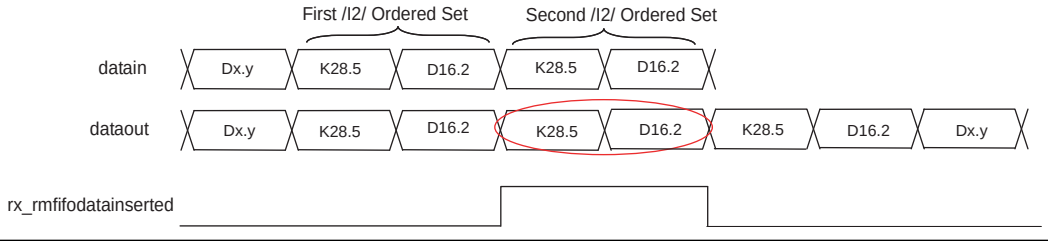


Figure 1–137 shows an example of rate match FIFO insertion in the case where one symbol is required to be inserted. Because the rate match FIFO can only delete /I2/ ordered set, it inserts one /I2/ ordered set (two symbols inserted).

Figure 1–137. Rate Match Insertion in GIGE Mode



SONET/SDH Mode

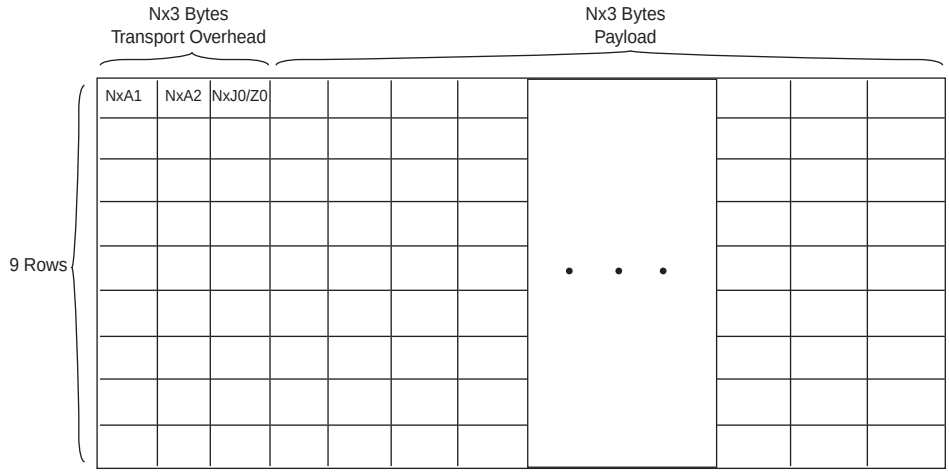
SONET/SDH is one of the most common serial-interconnect protocols used in backplanes deployed in communications and telecom applications. SONET/SDH defines various optical carrier (OC) sub-protocols for carrying signals of different capacities through a synchronous optical hierarchy.

SONET/SDH Frame Structure

Base OC-1 frames are byte-interleaved to form SONET/SDH frames. For example, 12 OC-1 frames are byte-interleaved to form one OC-12 frame; 48 OC-1 frames are byte-interleaved to form one OC-48 frame, and so on. SONET/SDH frame sizes are constant, with a frame transfer rate of 125 µs.

Figure 1–138 shows the SONET/SDH frame structure.

Figure 1–138. SONET/SDH Mode



Transport overhead bytes A1 and A2 are used for restoring frame boundary from the serial data stream. Frame sizes are fixed, so the A1 and A2 bytes appear within the serial data stream every 125 µs. In an OC-12 system, 12 A1 bytes are followed by 12 A2 bytes. Similarly, in an OC-48 system, 48 A1 bytes are followed by 48 A2 bytes.

In SONET/SDH systems, byte values of A1 and A2 are fixed as follows:

- A1 = 11110110 or 8'hF6
- A2 = 00101000 or 8'h28

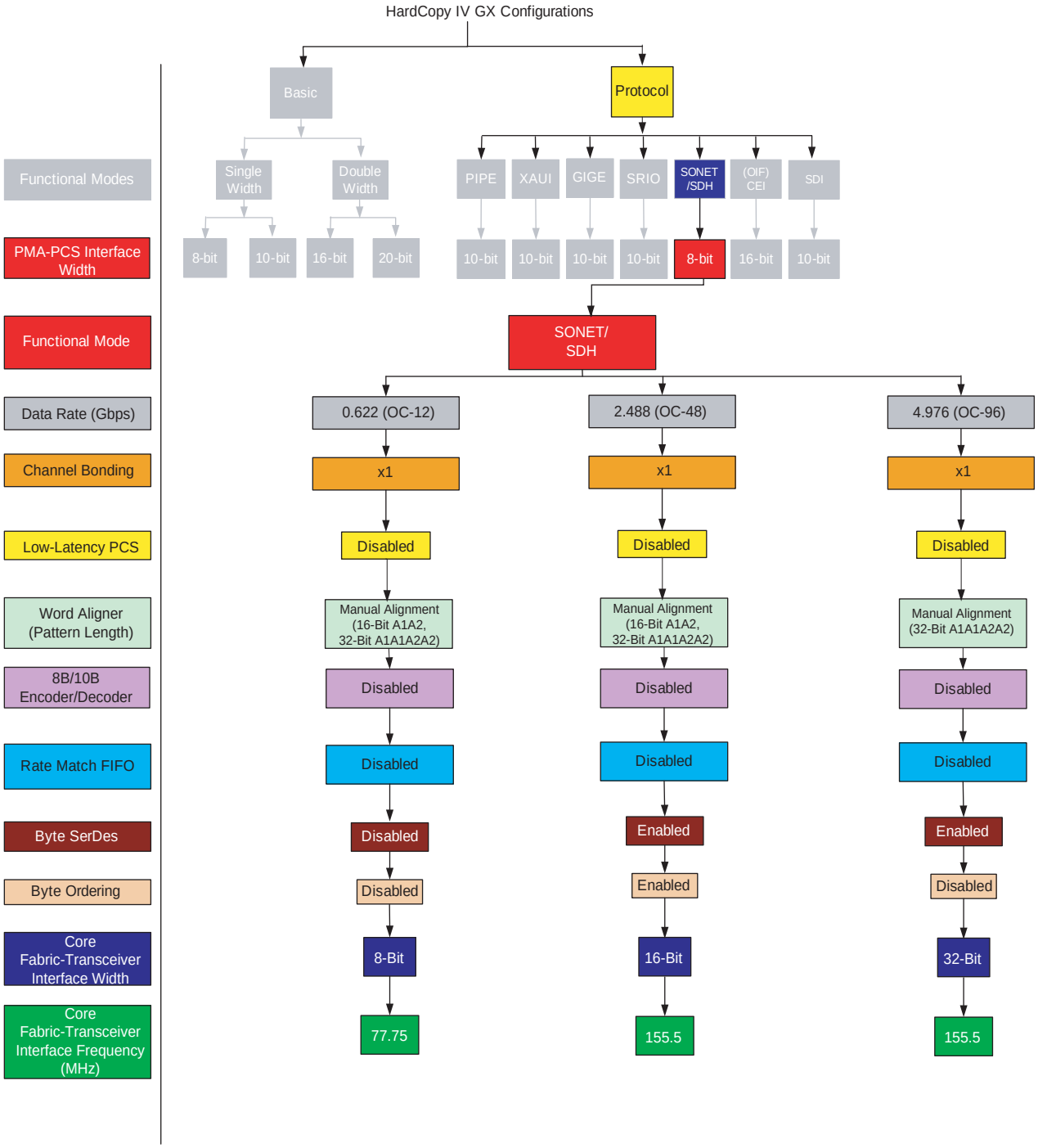
You can employ HardCopy IV GX transceivers as physical layer devices in a SONET/SDH system. These transceivers provide support for SONET/SDH protocol-specific functions and electrical features; for example, alignment to A1A2 or A1A1A2A2 pattern.

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 1–139 shows SONET/SDH mode configurations supported in HardCopy IV GX devices.

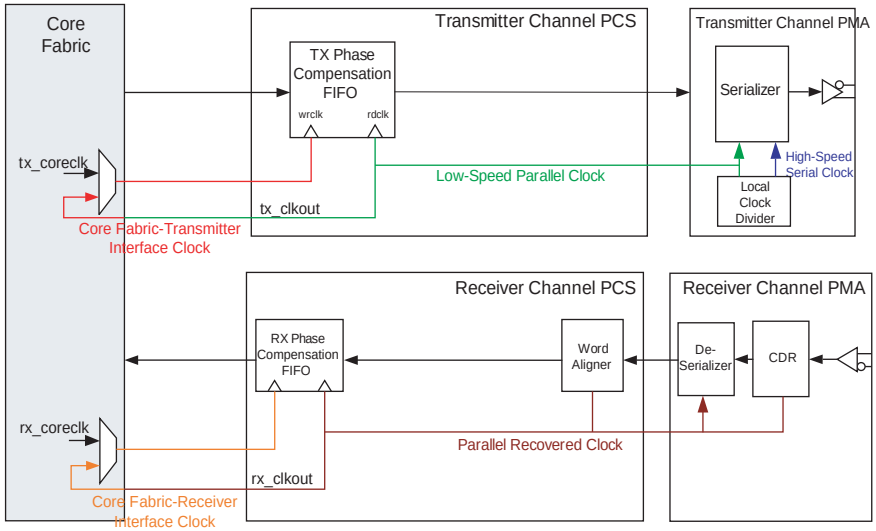
Figure 1–139. SONET/SDH Mode Configurations in HardCopy IV Devices



SONET/SDH OC-12 Datapath

Figure 1–140 shows the transceiver datapath when configured in SONET/SDH OC-12 mode.

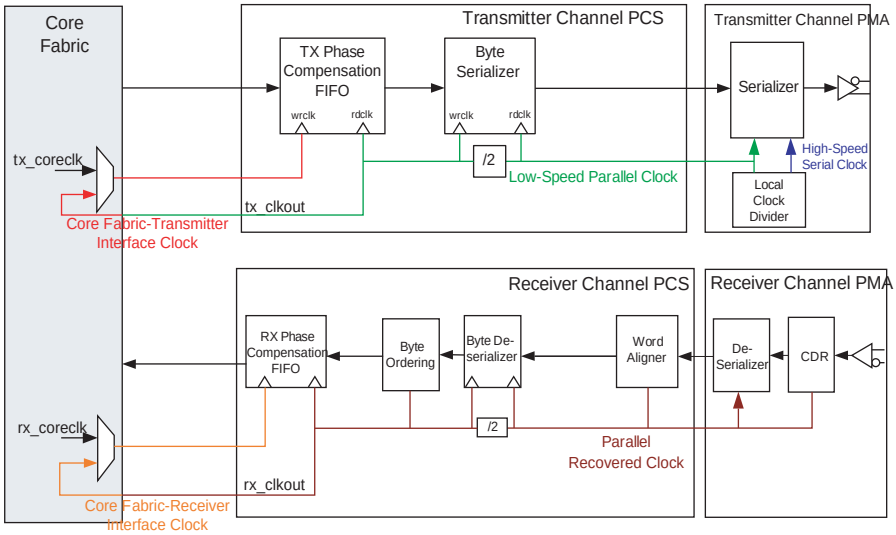
Figure 1–140. SONET/SDH OC-12 Datapath



SONET/SDH OC-48 Datapath

Figure 1–141 shows the transceiver datapath when configured in SONET/SDH OC-48 mode.

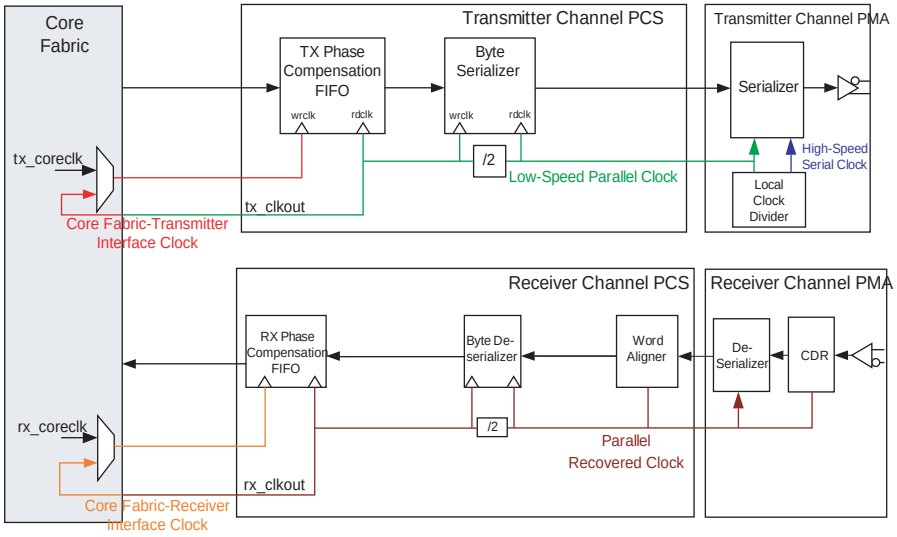
Figure 1–141. SONET/SDH OC-48 Datapath



SONET/SDH OC-96 Datapath

Figure 1–142 shows the transceiver datapath when configured in SONET/SDH OC-96 mode.

Figure 1–142. 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 1–48 on page 1–173 lists the correct word aligner settings for each bit transmission order.

Word Alignment

The word aligner in SONET/SDH OC-12, OC-48, and OC-96 modes is configured in manual alignment mode, as described in "Word Aligner in Single-Width Mode with 8-Bit PMA-PCS Interface Modes" on page 1–74.

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

In OC-96 configuration, the word aligner is only allowed to align to a A1A1A2A2 pattern, so input port `rx_ala2size` is unavailable. Barring this difference, the OC-96 word alignment operation is similar to that of the OC-12 and OC-48 configurations.

You can configure the word aligner to flip the alignment pattern bits programmed in the MegaWizard and compare them with the incoming data for alignment. This feature offers flexibility to the SONET backplane system for either a MSBit-to-LSBit or LSBit-to-MSBit data transfer. Table 1–48 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 1–48. Word Aligner Settings

Serial Bit Transmission Order	Word Alignment Bit Flip	Word Alignment Pattern
MSBit-to-LSBit	On	1111011000101000 (16'hF628)
MSBit-to-LSBit	Off	0001010001101111 (16'h146F)
LSBit-to-MSBit	Off	0010100011110110 (16'h28F6)

The behavior of the SONET/SDH word aligner control and status signals, along with an operational timing diagram, are explained in "Word Aligner in Single-Width Mode with 8-Bit PMA-PCS Interface Modes" on page 1–74.

OC-48 and OC-96 Byte Serializer and Deserializer

The OC-48 and OC-96 transceiver datapath includes the byte serializer and deserializer to allow the PLD interface to run at a lower speed. The OC-12 configuration does not use the byte serializer and deserializer blocks.

The byte serializer and deserializer blocks are explained in "Byte Serializer" on page 1–38 and "Byte Deserializer" on page 1–108, respectively.

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

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

OC-48 Byte Ordering

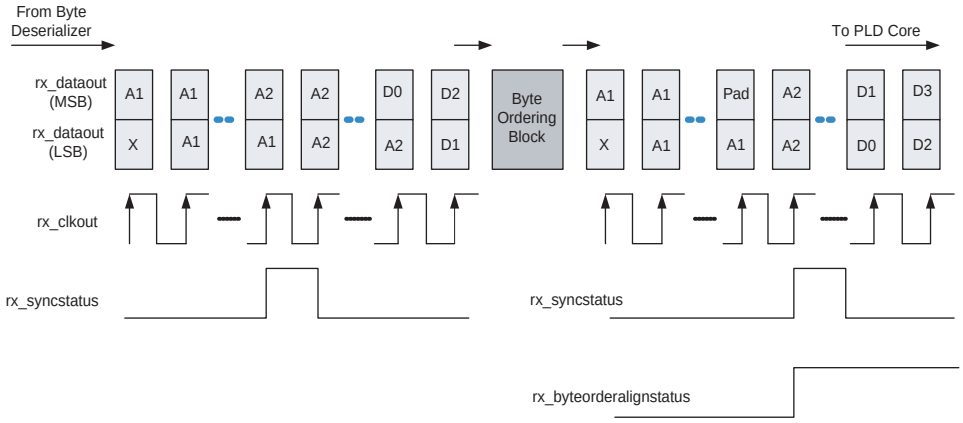
Because of byte deserialization, the MSByte of a word might appear at the `rx_dataout` port along with the LSByte of the next word.

In an OC-48 configuration, the byte ordering block is built into the datapath and can be leveraged to perform byte ordering. Byte ordering in an OC-48 configuration is automatic, as explained in "Word-Alignment-Based Byte Ordering" on page 1–113.

In automatic mode, the byte ordering block is triggered by the rising edge of the `rx_syncstatus` signal. As soon as the byte ordering block sees the rising edge of the `rx_syncstatus` signal, it compares the LSByte coming out of the byte deserializer with the A2 byte of the A1A2 alignment pattern. If the LSByte coming out of the byte deserializer does not match the A2 byte set in the ALTGX MegaWizard Plug-In Manager, the byte ordering block inserts a PAD character, as seen in Figure 1–143. Insertion of this PAD character enables the byte ordering block to restore the correct byte order.

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

Figure 1–143. OC-48 Byte Ordering in Automatic Mode



SDI Mode

The Society of Motion Picture and Television Engineers (SMPTE) defines various SDI standards for transmission of uncompressed video.

The following three SMPTE standards are popular in video broadcasting applications:

- SMPTE 259M standard—more popularly known as the standard-definition (SD) SDI, is defined to carry video data at 270 Mbps
- SMPTE 292M standard—more popularly known as the high-definition (HD) SDI, is defined to carry video data at either 1485 Mbps or 1483.5 Mbps
- SMPTE 424M standard—more popularly known as the third-generation (3G) SDI, is defined to carry video data at either 2970 Mbps or 2967 Mbps

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

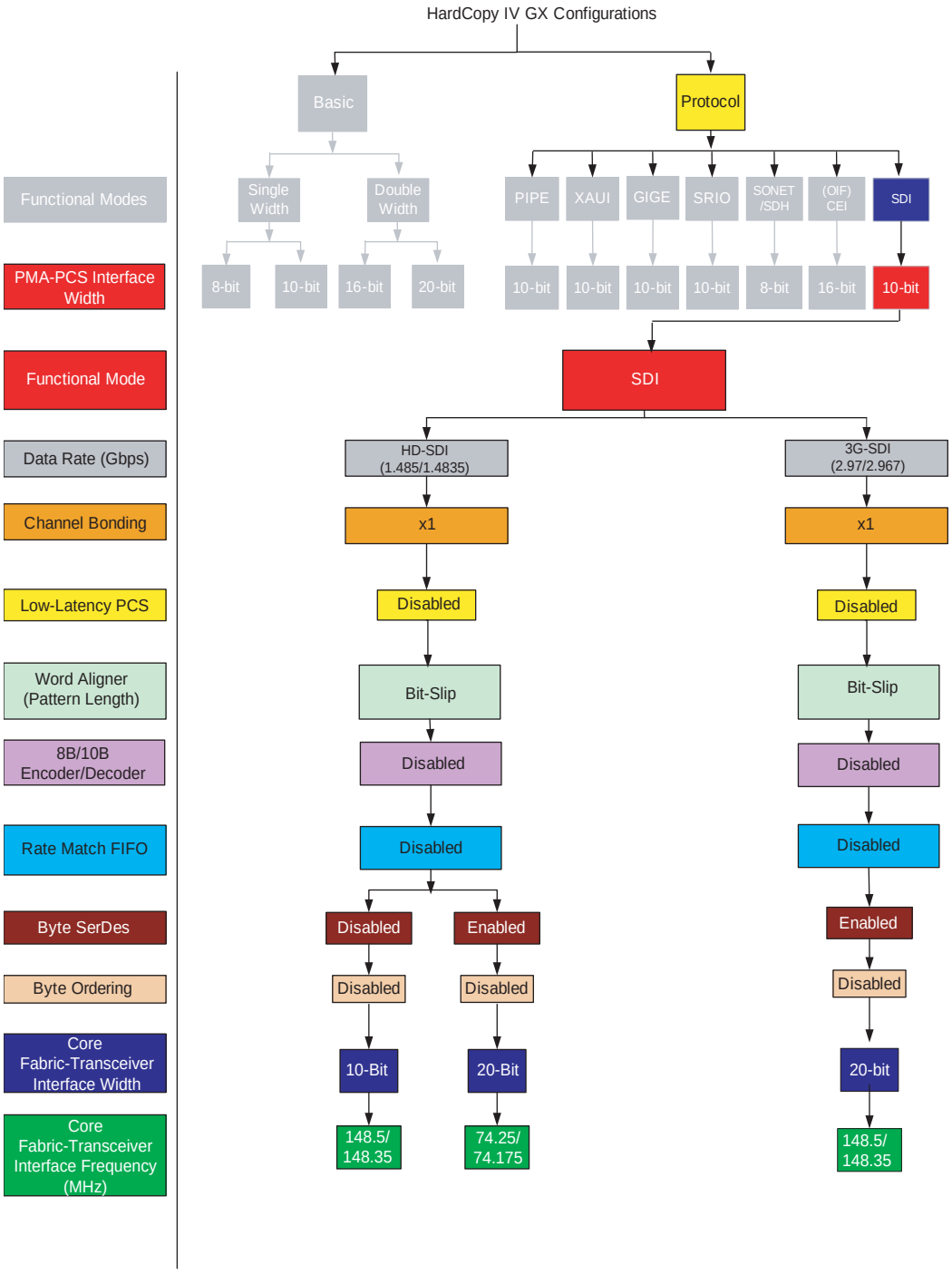
Table 1–49 shows ALTGX configurations supported by HardCopy IV GX transceivers in SDI mode.

Table 1–49. ALTGX Configurations in SDI Mode

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

Figure 1–144 shows SDI mode configurations supported in HardCopy IV GX devices.

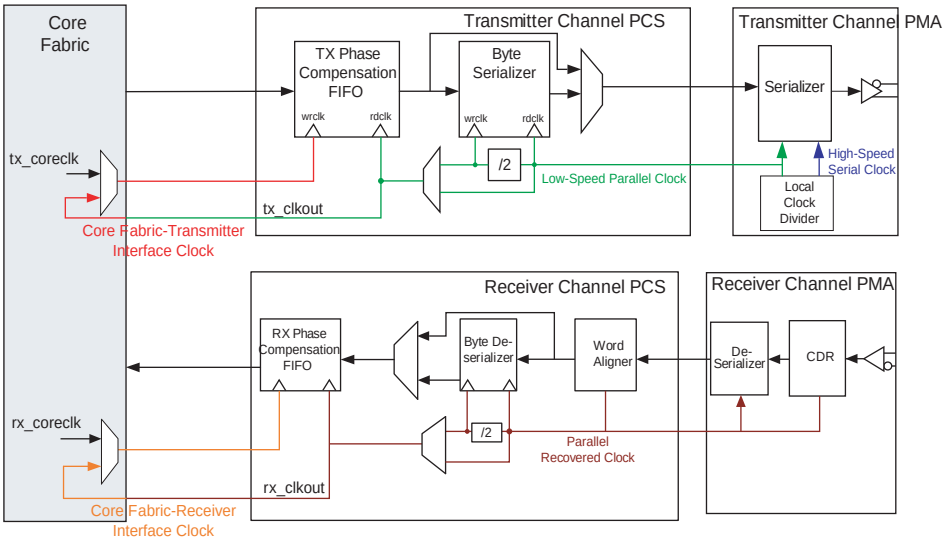
Figure 1–144. SDI Mode



SDI Mode Datapath

Figure 1–145 shows the transceiver datapath when configured in SDI mode.

Figure 1–145. SDI Mode Datapath



Transmitter Datapath

The transmitter datapath, in HD-SDI configuration with 10 bit wide core 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 core 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 core 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 core 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

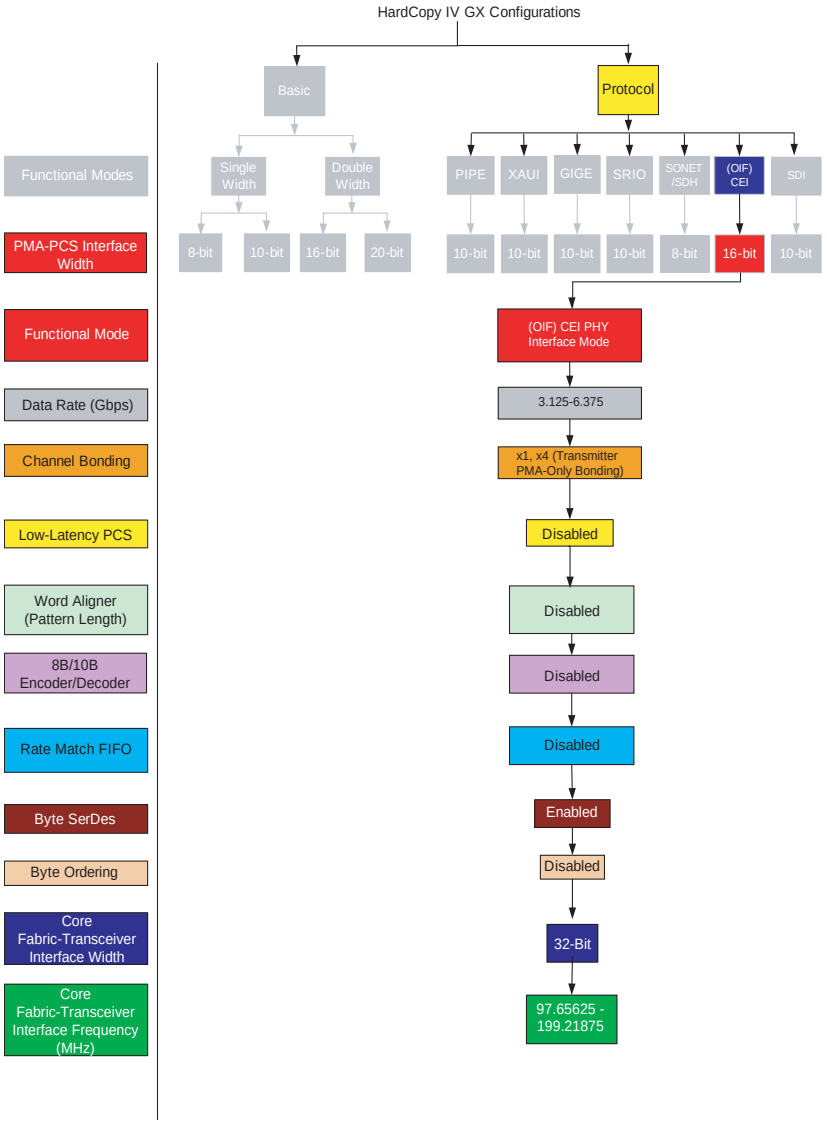
Optical Internetworking Forum (OIF) CEI PHY interface mode is intended to support two main protocols:

- Common Electrical I/O (CEI-6G) protocol defined by the Optical Internetworking Forum at data rates between 4.976 Gbps and 6.375 Gbps
- Interlaken protocol at data rates between 3.125 Gbps and 6.375 Gbps

HardCopy IV GX transceivers support a data rate between 3.125 Gbps and 6.375 Gbps in (OIF) CEI PHY interface mode.

Figure 1–146 shows (OIF) CEI PHY interface mode configurations supported in HardCopy IV GX devices.

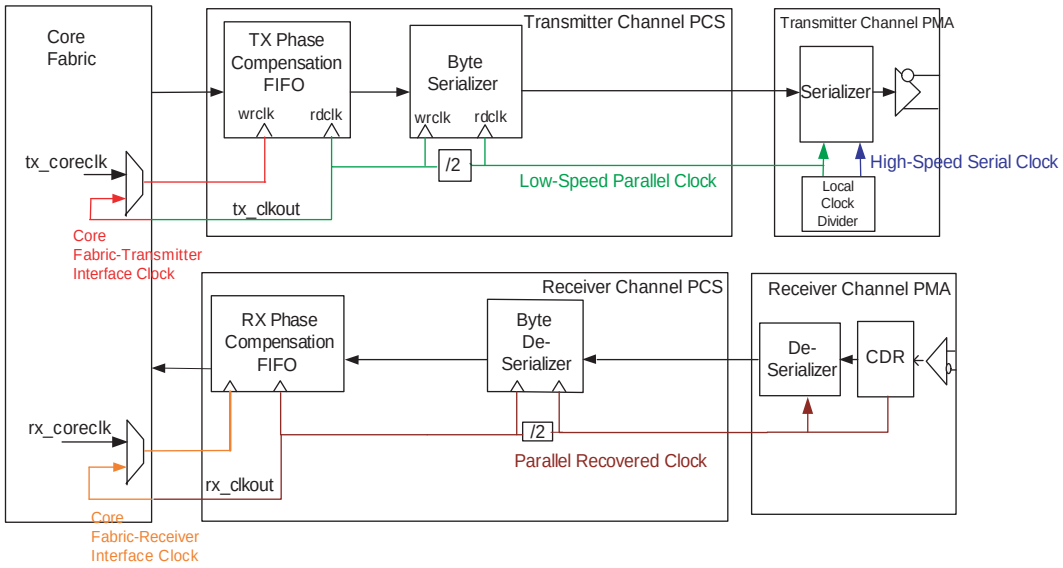
Figure 1–146. (OIF) CEI PHY Interface Mode



(OIF) CEI PHY Interface Mode Datapath

Figure 1–147 shows the ALTGX megafunction transceiver datapath when configured in (OIF) CEI PHY interface mode.

Figure 1–147. (OIF) CEI PHY Interface Mode Datapath



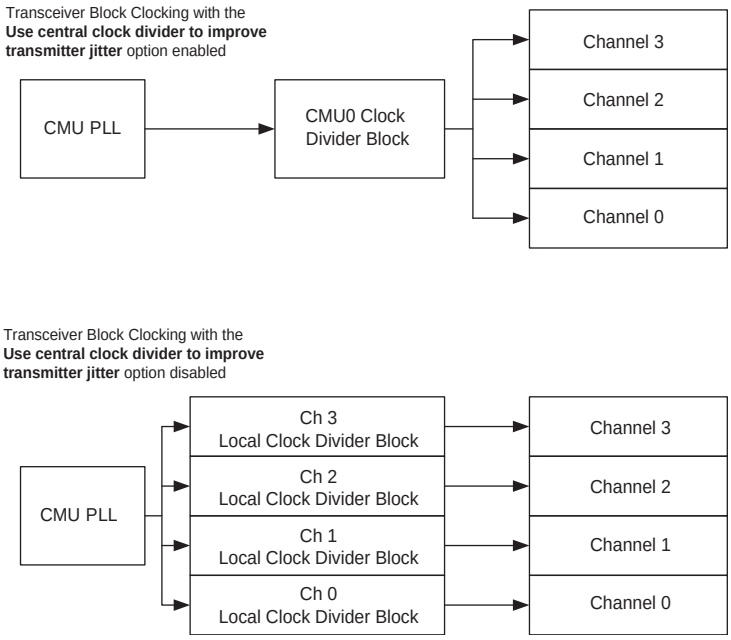
(OIF) CEI PHY Interface Mode Clocking

For improved transmitter jitter performance, the ALTGX MegaWizard Plug-In Manager provides the **Use central clock divider to improve transmitter jitter** option. If you select this option, the high-speed serial clock generated by the `CMU0` clock divider block clocks all four transceiver channels within the same transceiver block. Otherwise, the high-speed serial clock generated by the local clock divider in each channel clocks the respective channel.

Unlike PCI Express (PIPE) 4, XAUI or Basic ×4 mode, the transmitter PCS is not bonded in the (OIF) CEI PHY interface mode with the low-jitter option selected.

Figure 1–148 shows transceiver clocking in (OIF) CEI PHY interface mode with and without the improved transmitter jitter option enabled.

Figure 1–148. Transceiver Clocking in (OIF) CEI PHY Interface Mode



Transceiver Placement Limitations with the Use Central Clock Divider to Improve Transmitter Jitter Option Enabled

If one or more channels in a transceiver block are configured to (OIF) CEI PHY interface mode with the improved jitter clocking option enabled, the remaining channels in that transceiver block must either be configured in (OIF) CEI PHY interface mode with this option enabled or must be unused. All used channels within a transceiver block configured in (OIF) CEI PHY interface mode with the improved jitter clocking option enabled must also run at the same data rate.

Figure 1–149 and Figure 1–150 show two examples each of legal and illegal transceiver placements with respect to the improved jitter clocking option in (OIF) CEI PHY interface mode.

Figure 1–149. Examples of Legal Transceiver Placement in (OIF) CEI PHY Interface Mode

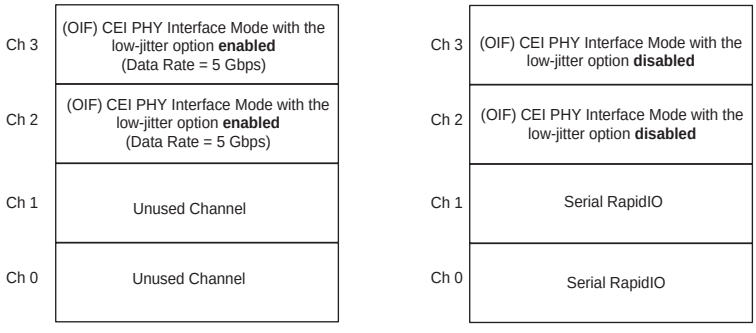
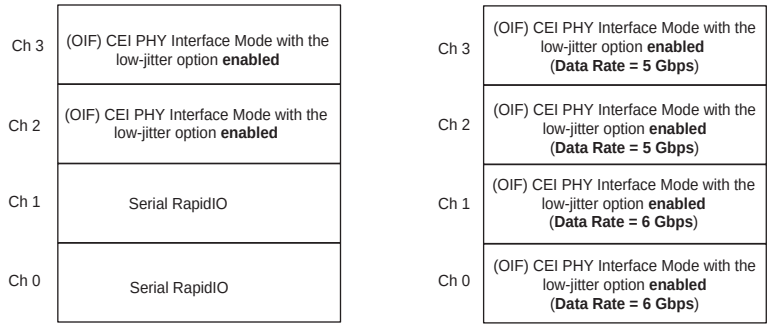


Figure 1–150. Examples of Illegal Transceiver Placement in (OIF) CEI PHY Interface Mode



Serial RapidIO Mode

The RapidIO Trade Association defines a high-performance, packet-switched interconnect standard to pass data and control information between microprocessors, digital signal, communications, and network processors, system memories, and peripheral devices.

Serial RapidIO physical layer specification defines three line rates:

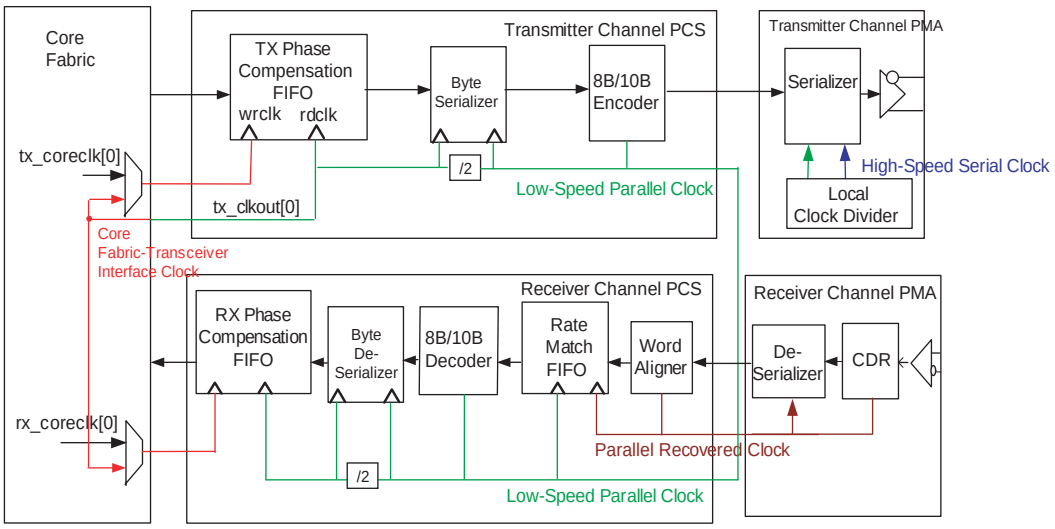
- 1.25 Gbps
- 2.5 Gbps
- 3.125 Gbps

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

HardCopy IV GX transceivers support only single-lane (1×) configuration at all three line rates. Four 1× channels configured in Serial RapidIO mode can be instantiated to achieve a 4× Serial RapidIO link. The four transmitter channels in this 4× Serial RapidIO link are not bonded. The four receiver channels in this 4× Serial RapidIO link do not have lane alignment or deskew capability.

Figure 1–151 shows the ALTGX transceiver datapath when configured in Serial RapidIO mode.

Figure 1–151. 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.

The ALTGX MegaWizard Plug-In Manager automatically defaults the synchronization state machine to indicate synchronization when the receiver receives 127 K28.5 (10'b0101111100 or 10'b1010000011) synchronization code groups without receiving an intermediate invalid code group. Once synchronized, the state machine indicates loss of synchronization when it detects three invalid code groups separated by less than 255 valid code groups or when it is reset.

Receiver synchronization is indicated on the `rx_syncstatus` port of each channel. A high on the `rx_syncstatus` port indicates that the lane is synchronized and a low indicates that it has fallen out of synchronization.

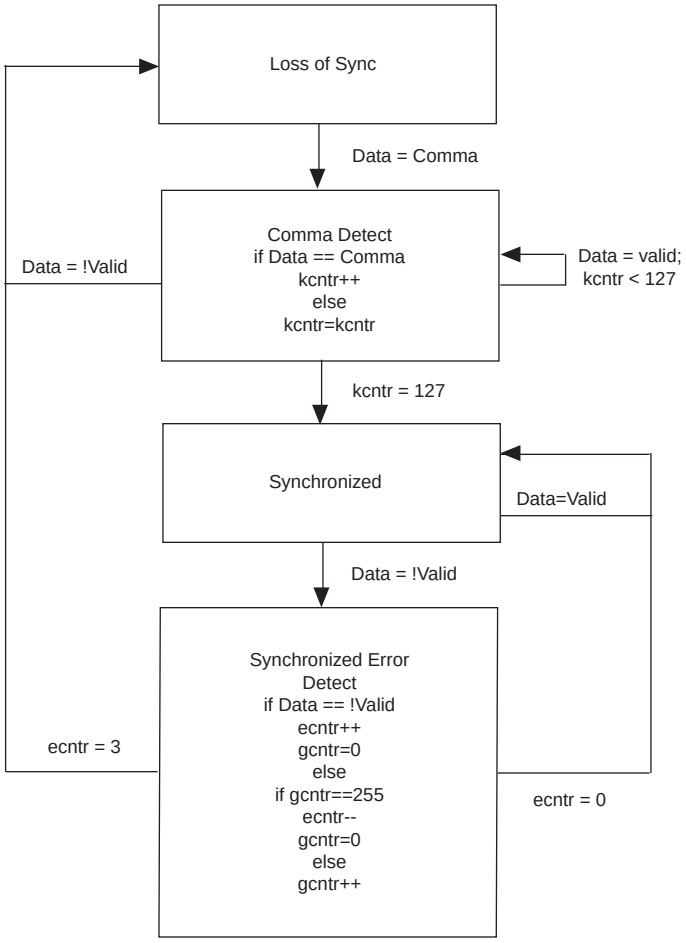
Table 1–50 lists the ALTGX megafunction synchronization state machine parameters when configured in Serial RapidIO mode.

Table 1–50. Synchronization State Machine Parameters in Serial RapidIO Mode

Parameters	Number
Number of valid K28.5 code groups received to achieve synchronization.	127
Number of errors received to lose synchronization.	3
Number of continuous good code groups received to reduce the error count by one.	255

Figure 1–152 shows a conceptual view of the synchronization state machine implemented in Serial RapidIO functional mode.

Figure 1–152. Synchronization State Machine in Serial RapidIO Mode



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 core 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 core fabric.

You can configure four regular transceiver channels inside each transceiver block in Basic (PMA-Direct) functional mode. Two CMU channels inside each transceiver block can be configured 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.

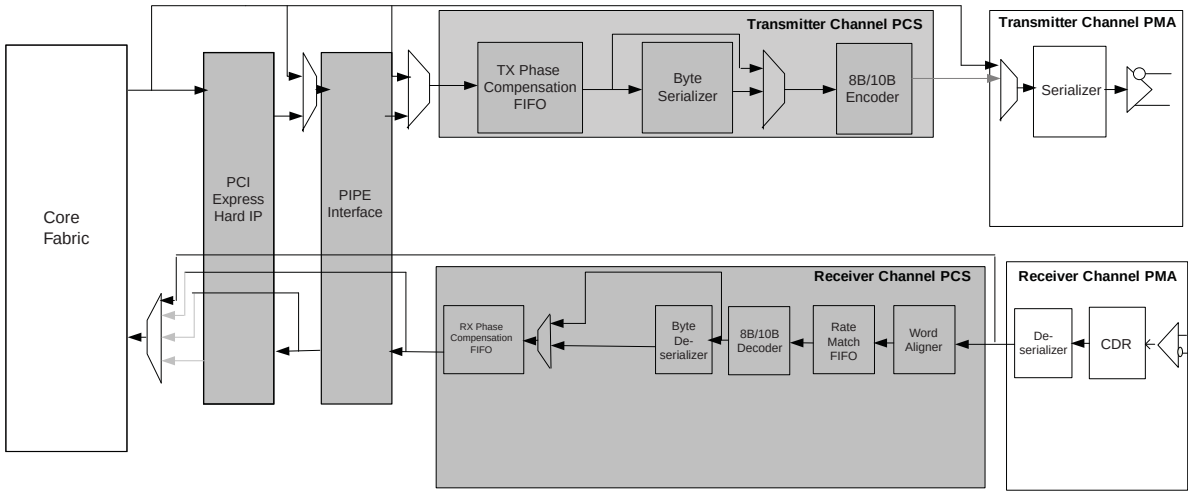
For more information, refer to the *HardCopy IV GX Dynamic Reconfiguration* chapter in volume 3 of the *HardCopy IV Device Handbook*.

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

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

Figure 1–153 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 1–153. HardCopy IV GX Transceiver Configured in Basic (PMA-Direct) Mode



Note to Figure 1–153:

(1) The grayed out blocks shown in Figure 1–153 are not available in the CMU channels. Therefore, the CMU channels can be configured to operate as transceiver channels in PMA direct mode only.

The grayed out blocks shown in Figure 1–153 are not available in the CMU channels. Therefore, the CMU channels can be configured to operate as transceiver channels in PMA-Direct mode only.

In Basic (PMA-Direct) Mode, you can configure the transceiver channel in two main configurations:

- Basic PMA-Direct ×1 configuration and
- Basic PMA-Direct ×N configuration.

You can configure the transceiver in Basic PMA-Direct ×1/ ×N mode by setting the appropriate sub-protocol in the **Which sub protocol will you be using**? field. You can select single-width or double-width by selecting **Single/Double** in the **What is the deserializer block width?** field in the ALTGX MegaWizard Plug-In Manager.

In single-width mode, the PMA-PLD interface is 8 bit/10 bit wide; whereas in double-width mode, the PMA-PLD interface is 16 bit/20 bit wide.

Table 1–51 shows the PLD-PMA interface widths and data rates supported in Basic PMA-Direct ×1/×N single-width and double-width modes.

Table 1–51. Basic Deterministic Latency Mode

Basic PMA Direct ×1/×N Functional Mode	Supported Data Rate Range at Speed Grade -3	Core Fabric frequency (Max)	Core Fabric-PMA Interface Width
Basic PMA Direct ×1/ ×N Single-Width Mode	600 Mbps to 1.28 Gbps	318.75 MHz	8 bit
	600 Mbps to 1.6 Gbps	318.75 MHz	10 bit
Basic PMA Direct ×1/ ×N Double-Width Mode	1 Gbps to 2.56 Gbps	318.75 MHz	16 bit
	1 Gbps to 3.2 Gbps	318.75 MHz	20 bit

Basic PMA-Direct x1 Configuration

You can configure a transceiver channel in this mode by setting the **which protocol will you be using?** field to **Basic (PMA-Direct)** and the **which sub protocol will you be using?** field to **none**. In this configuration, the Quartus II software requires one of the two CMU PLLs within the same transceiver block to provide high-speed clocks to the transmitter side of the channel.

Basic PMA-Direct xN Configuration

You can configure a transceiver channel in this mode by setting the **which protocol will you be using** field to **Basic (PMA-Direct)** and the **which sub protocol will you be using** field to **×N**. In this mode, all the transmitter channels can receive their high-speed clock from the `CMU PLL0` from the transceiver blocks or the ATX PLL present on the same side of the device. These clocks are provided through the `×N_Top` or `×N_Bottom` clock line.

Each receiver in a receiver channel has a dedicated CDR that provides a high-speed clock.

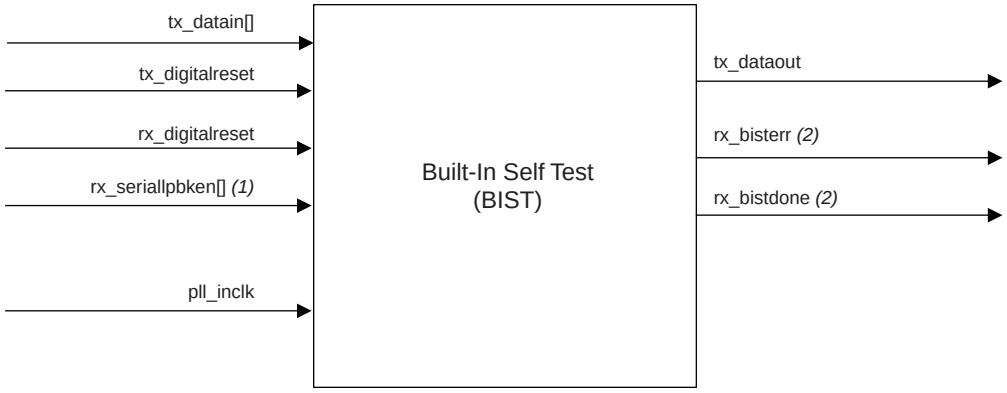
Built-In Self Test MODES

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

BIST Mode Pattern Generators and Verifiers

Each transceiver channel in the HardCopy IV GX device contains 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 1–154 shows the enabled input and output ports when you select BIST mode (except incremental patterns).

Figure 1–154. Input and Output Ports for BIST Modes



Notes to Figure 1–154:

(1) `rx_serialpbken` is required in PRBS.

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

Three types of pattern generators and verifiers are available:

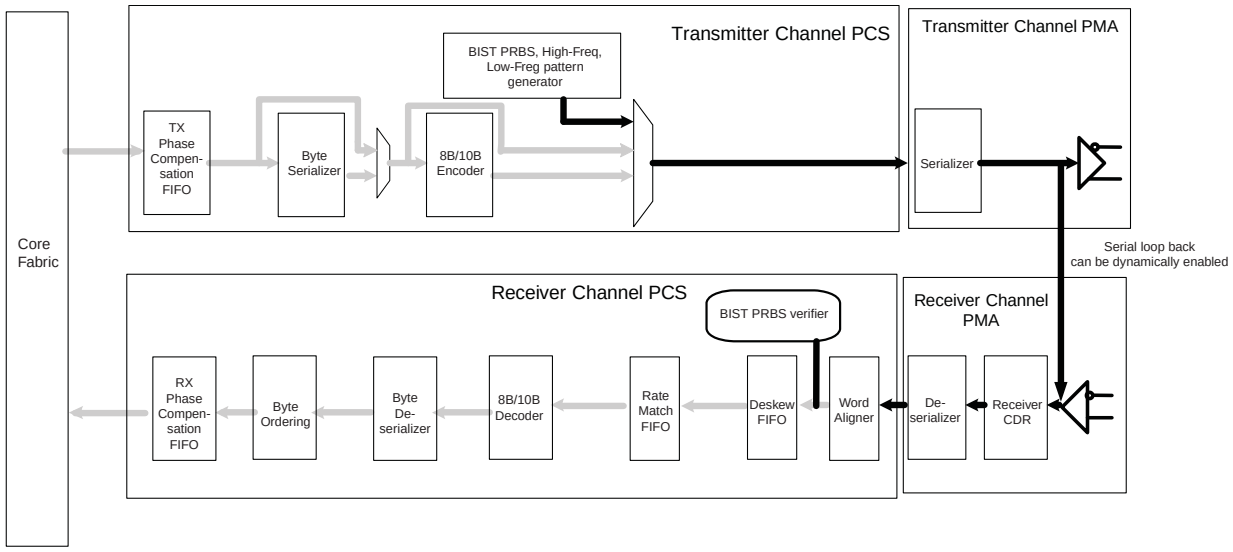
- BIST incremental data generator and verifier—This is only available in parallel loop back mode. For more information, refer to "Serial Loopback" on page 1–188.
- High frequency and low frequency pattern generator—The high frequency patterns generate alternate ones and zeros and the low frequency patterns generate five ones and five zeroes in single-width mode and ten ones and ten zeroes in double-width mode. These patterns do not have a corresponding verifier. You can enable the serial loopback option to dynamically loop the generated pattern to the receiver channel using the `rx_seriallpbken` port. Therefore, the 8B/10B encoder/decoder blocks are bypassed in the Basic PRBS mode.
- Pseudo Random Binary Sequence (PRBS) generator and verifier—The PRBS generator and verifier interface with the serializer and deserializer in the PMA blocks. The advantage of using a PRBS data stream is that the randomness yields an environment that stresses the transmission medium. In the data stream, you can observe both random jitter and deterministic jitter using a time interval analyzer, bit error rate tester, or oscilloscope. The PRBS repeats after completing an iteration. The number of bits the PRBSx pattern sends before repeating the pattern is $(2^{x} -1)$ bits.

Different PRBS patterns are available as a subprotocol under Basic functional mode for single-width and double-width mode, as shown in the following sections.

You can enable the **serial loopback** option in Basic PRBS mode to loop the generated pattern to the receiver channel. This creates a `rx_seriallpbken` port that you can use to dynamically control the serial loopback. The 8B/10B encoder/decoder blocks are bypassed in Basic PRBS mode.

Figure 1–155 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 1–155. BIST PRBS, High Frequency, and Low Frequency Pattern Datapath



PRBS in Single-Width Mode

Table 1–52 shows various PRBS patterns and corresponding word alignment patterns for PRBS in single-width mode configuration.

Table 1–52. Available PRBS, High Frequency, and Low Frequency Patterns in Single-Width Mode (Part 1 of 2)

Patterns	Polynomial	Channel Width 8 Bit *(1)*	Word Alignment Pattern with Channel Width 8 Bit	Maximum Data Rate With Channel Width 8 Bit (Gbps)	Channel Width 10 Bit *(1)*	Word Alignment Pattern	Maximum Data Rate with Channel Width 10 Bit (Gbps)
PRBS 7	$X^7 + X^6 + 1$	Y	16'h3040	2.5	N	—	—
PRBS 8	$X^8 + X^7 + 1$	Y	16'hFF5A	2.5	N	—	—
PRBS 10	$X^{10} + X^7 + 1$	N	—	N/A	Y	10'h3FF	3.125
PRBS 23	$X^{23} + X^{18} + 1$	Y	16'hFFFF	2.5	N	—	—
High frequency *(2)*	1010101010	Y	—	2.5	Y	—	3.125

Table 1–52. Available PRBS, High Frequency, and Low Frequency Patterns in Single-Width Mode (Part 2 of 2)

Patterns	Polynomial	Channel Width 8 Bit *(1)*	Word Alignment Pattern with Channel Width 8 Bit	Maximum Data Rate With Channel Width 8 Bit (Gbps)	Channel Width 10 Bit *(1)*	Word Alignment Pattern	Maximum Data Rate with Channel Width 10 Bit (Gbps)
Low Frequency *(2)*	0000011111	N	—	—	Y	—	3.125

Notes to Table 1–52:

(1) Channel width refers to the **What is the channel width?** option in the **General** screen of the ALTGX MegaWizard Plug-In Manager. Based on the selection, an 8 or 10 bits wide pattern is generated as indicated by a **Yes (Y)** or **No (N)**.

(2) A verifier is not available for the specified patterns.

The status signals `rx_bistdone` and `rx_bisterr` indicate the status of the verifier. The `rx_bistdone` port gets asserted and stays high when the verifier either receives one full cycle of incremental pattern or it detects an error in the receiver data. The `rx_bisterr` signal gets asserted and stays high when the verifier detects an error. You can reset the PRBS pattern generator and verifier by asserting the `tx_digitalreset` and `rx_digitalreset` signals, respectively.

PRBS in Double-Width Mode

Table 1–53 shows various PRBS patterns and corresponding word alignment patterns for PRBS in double-width mode configuration.

Table 1–53. Available PRBS, High Frequency, and Low Frequency Patterns in Double-Width Mode

Patterns	Polynomial	Channel Width 16 Bit *(1)*	Word Alignment Pattern with Channel Width 16 Bit	Maximum Data Rate with Channel Width 16 Bit (Gbps)	Channel Width 20 Bit *(1)*	Word Alignment Pattern	Maximum Data Rate with Channel Width 20 Bit (Gbps)
PRBS 7	$X^7 + X^6 + 1$	Y	16'h3040	5	Y	20'h43040	6.375
PRBS 23	$X^{23} + X^{18} + 1$	Y	32'h007FFFF F	5	Y	40'h00007FF FFF	6.375
High frequency *(2)*	1010101010	Y	—	5	Y	—	6.375
Low Frequency *(2)*	0000011111	N	—	—	Y	—	6.375

Notes to Table 1–53:

(1) Channel width refers to the **what is the channel width?** option in the **General** screen of the ALTGX MegaWizard Plug-In Manager. Based on the selection, a 16 or 20 bits wide pattern is generated as indicated by a **Yes** (**Y**) or **No** (**N**).

(2) Verifier is not available for the specified patterns.

The status signals `rx_bisterr` and `rx_bistdone` are available to indicate the status of the verifier. For more information about the behavior of these status signals, refer to "Single-Width Mode" on page 1–39.

Loopback Modes

HardCopy IV GX devices provide various loopback options that allow you to verify the working of different functional blocks in the transceiver channel. The available loopback options are:

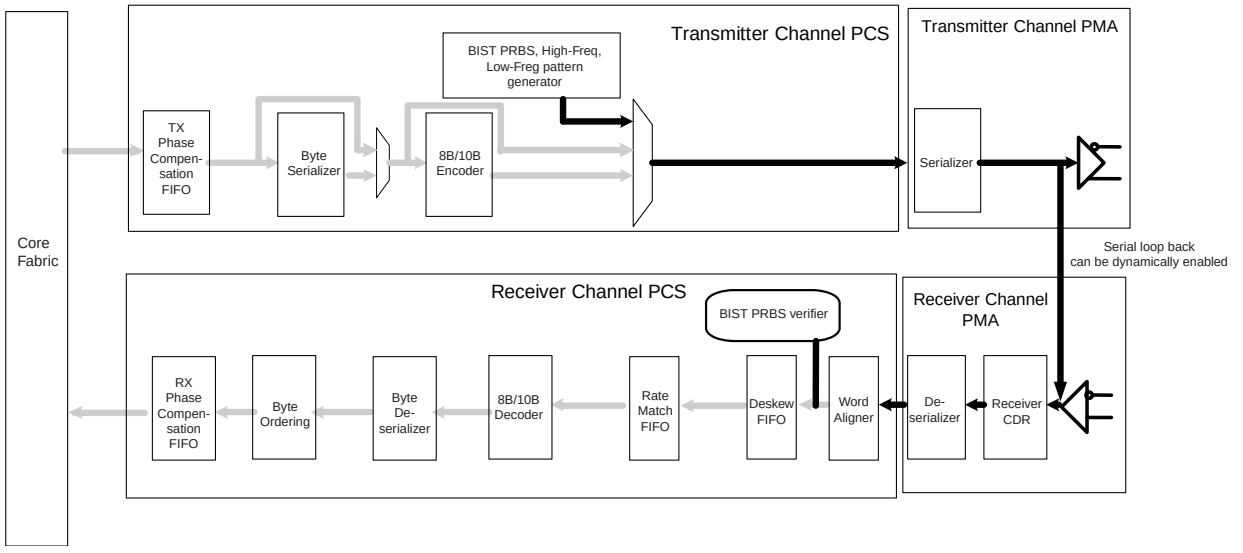
- Serial loopback—available in all functional modes except PCI Express (PIPE) mode
- Reverse serial loopback—available in Basic mode only
- Reverse serial pre-CDR loopback—available in Basic mode only
- PIPE reverse parallel loopback—supported in PIPE protocol only

Serial Loopback

The **serial loopback** option is available for all functional modes except PCI Express (PIPE) mode. Figure 1–156 shows the datapath for serial loopback. The data from the core fabric passes through the transmitter channel and gets looped back to the receiver channel, bypassing the receiver buffer. The received data is available to the core logic for verification. Using this option, you can check the working for all enabled PCS and PMA functional blocks in the transmitter and receiver channel. When you enable the **serial loopback** option, the ALTGX MegaWizard Plug-In Manager provides the `rx_seriallpbken` port to dynamically enable serial loopback on a channel-by-channel basis. Set the `rx_seriallpbken` signal to logic high to enable serial loopback.

When serial loopback is enabled, the transmitter channel sends the data to both the `tx_dataout` output port and to the receiver channel. The differential output voltage on the `tx_dataout` ports is based on the selected V_{OD} settings. The looped back data is received by the receiver CDR and is retimed through different clock domains. You must provide an alignment pattern for the word aligner to enable the receiver channel to retrieve the byte boundary.

Figure 1–156. Serial Loopback Datapath



Parallel Loopback Mode

You can configure a transceiver channel in this mode by setting the **which protocol will you be using**? field to **Basic** and the **which sub protocol will you be using?** field to **BIST**. You can only configure a **Receiver and Transmitter** transceiver channel in this functional mode. You can configure a transceiver channel in this mode in either a single-width or double-width configuration.

The BIST pattern generator and pattern verifier are located near the core fabric in the PCS block of the transceiver channel. This placement allows for testing the complete transmitter PCS and receiver PCS datapaths for bit errors. This mode is primarily used for transceiver channel debugging, if needed.

The parallel loopback mode is available only with a built-in 16 bit incremental pattern generator and verifier. The channel width is fixed to 16 bits in this mode. Also in this mode, the incremental pattern 00-FF is looped back to the receiver channel at the PCS functional block boundary before the PMA and is sent out 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 1–157. The incremental data pattern is not available to the core logic for verification.

Figure 1–157. Enabled PCS Functional Blocks in Parallel Loopback

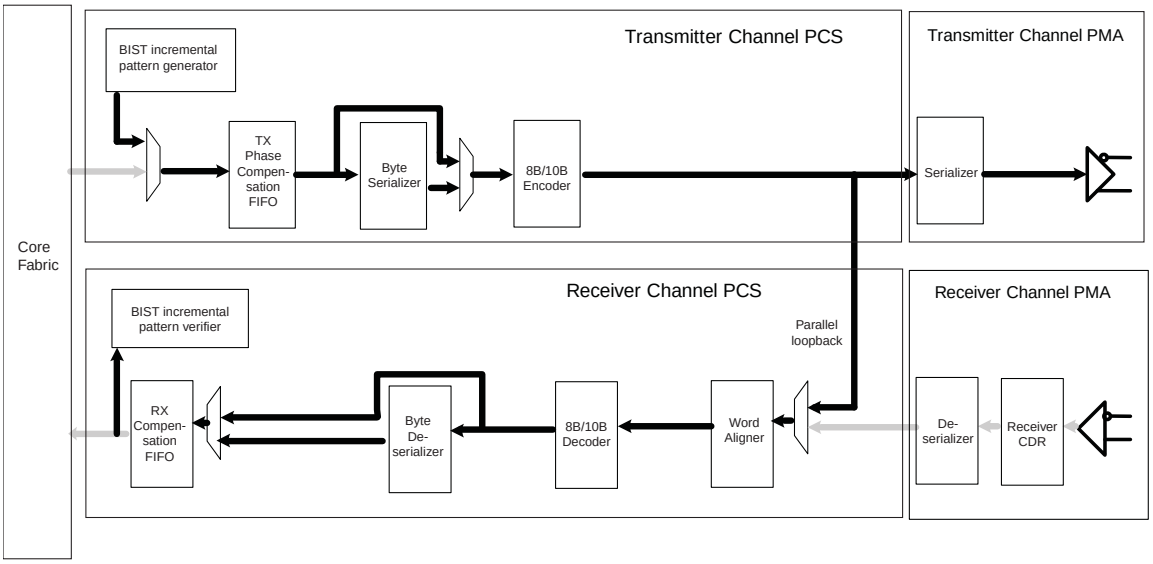


Table 1–54 shows the enabled PCS functional blocks for single-width and double-width mode. The last column in Table 1–54 shows the supported channel width setting for parallel loopback.

Table 1–54. Enabled PCS Functional Blocks for Parallel Loopback

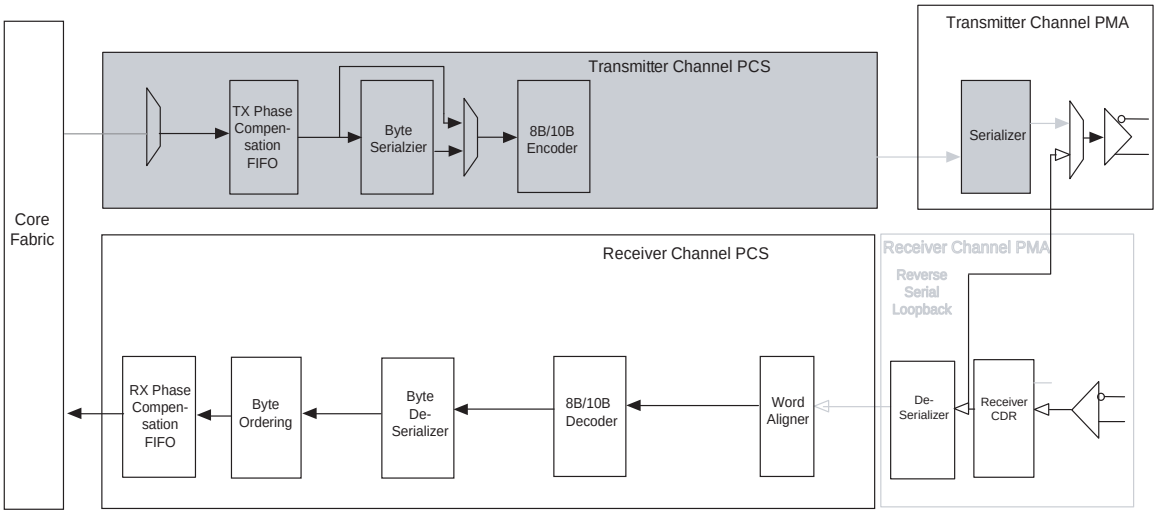
Configuration	8B/10B Encoder	Byte Serializer	Data Rate Range	Supported Channel Width Setting in the ALTGX MegaWizard Plug-In Manager for Parallel Loopback
Single-width mode	Enabled	Enabled	600 Mbps to 3.125Gbps	16
Double-width mode	Enabled	Disabled	1Gbps to 5Gbps	16

The status signals rx_bistdone and rx_bisterr indicate the status of the verifier. The rx_bistdone port is asserted and stays high when the verifier either receives one full cycle of incremental pattern or it detects an error in the receiver data. The rx_bisterr signal is asserted and stays high when the verifier detects an error. You can reset the incremental pattern generator and verifier by asserting the tx_digitalreset and rx_digitalreset signals, respectively.

Reverse Serial Loopback

Reverse serial loopback is available as a subprotocol under Basic functional mode. In reverse serial loopback mode, the data is received through the rx_datain port, retimed through the receiver CDR and sent out to the tx_dataout port. The received data is also available to the core logic. Figure 1–158 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 on the transmitter buffer through the ALTGX MegaWizard Plug-In Manager. The pre-emphasis settings for the transmitter buffer cannot be changed in this configuration. Reverse serial loopback is often implemented when using a bit error rate tester (BERT) on the upstream transmitter.

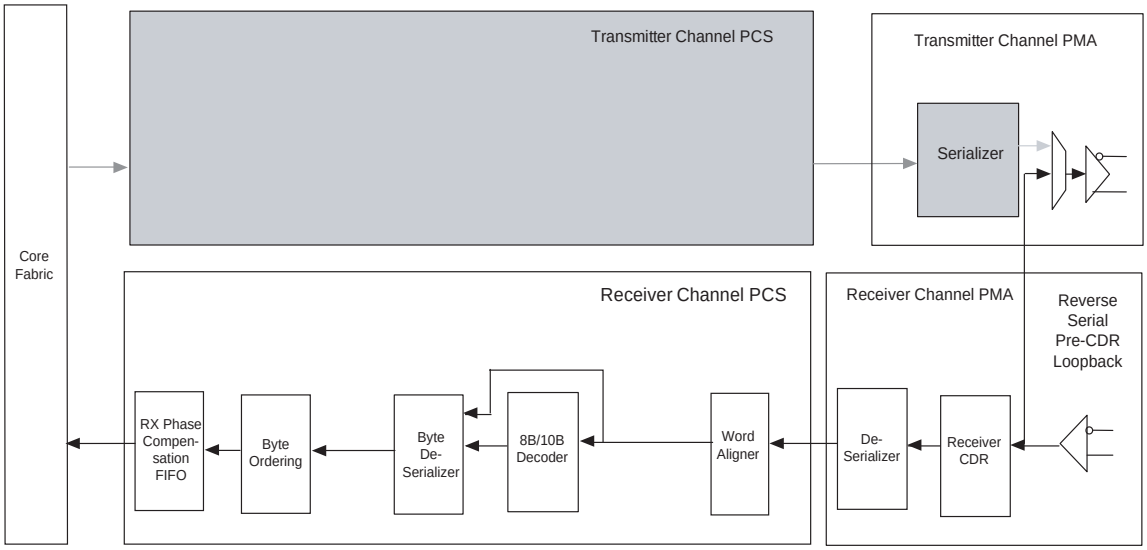
Figure 1–158. Reverse Serial Loopback Datapath (Grayed-Out Blocks are Not Active in this Mode)



Reverse Serial Pre-CDR Loopback

The reverse serial pre-CDR loopback is available as a subprotocol under Basic functional mode. In reverse serial pre-CDR loopback, the data received through the rx_datain port is looped back to the tx_dataout port *before* the receiver CDR. The received data is also available to the core logic. Figure 1–159 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 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. The pre-tap and second post-tap values cannot be changed in this loop back configuration.

Figure 1–159. Reverse Serial Pre-CDR Loopback Datapath



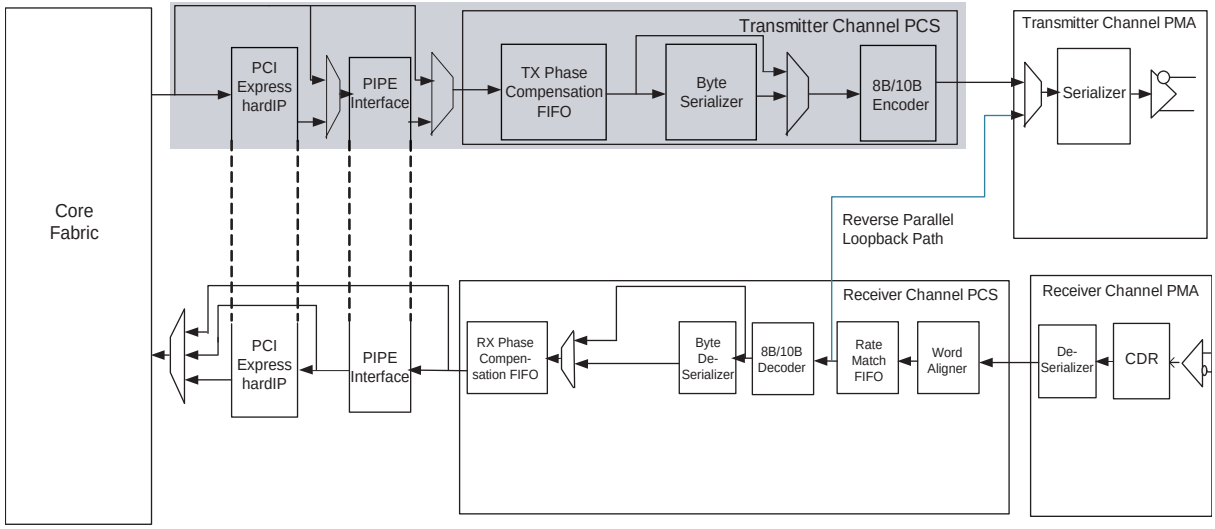
PCI Express (PIPE) Reverse Parallel Loopback

PCI Express (PIPE) reverse parallel loopback is only available in PIPE functional mode for Gen1 and Gen2 data rates. As shown in Figure 1–160, 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 core fabric through the `rx_dataout` port. This loopback mode is compliant with the PCI Express (PIPE) specification 2.0. To enable this loopback mode, assert the `tx_detectrxloopback` port.

This is the only loopback option supported in PCI Express (PIPE) functional mode.

In Figure 1–160, the grayed areas show the inactive paths when the PCI Express (PIPE) reverse parallel loopback mode is enabled.

Figure 1–160. PCI Express (PIPE) Reverse Parallel Loopback Mode Datapath (Grayed-Out Blocks are Not Active in this Mode)



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 variations.

Calibration Block Location

Figure 1–161 shows the location and number of calibration blocks available for different transceiver block device families. In Figure 1–161 through Figure 1–163, the calibration block R0 and L0 refer to the calibration blocks on the right and left side, respectively.

Figure 1–161. Calibration Blocks

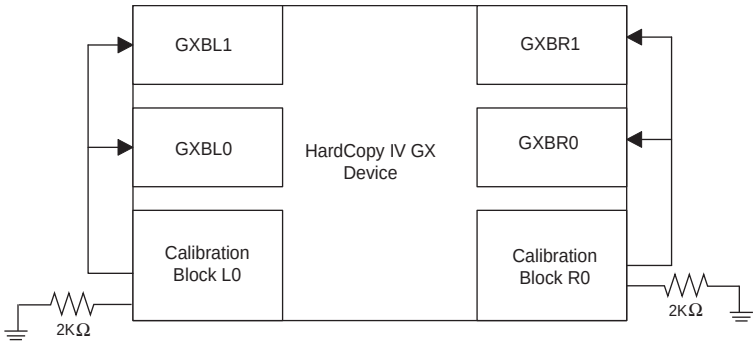


Figure 1–162 shows HardCopy IV GX device families that have three transceiver blocks each on the left and right side.

Figure 1–162. Calibration Block Locations in HardCopy IV GX Device Families with Three-Transceiver Blocks (on Each Side)

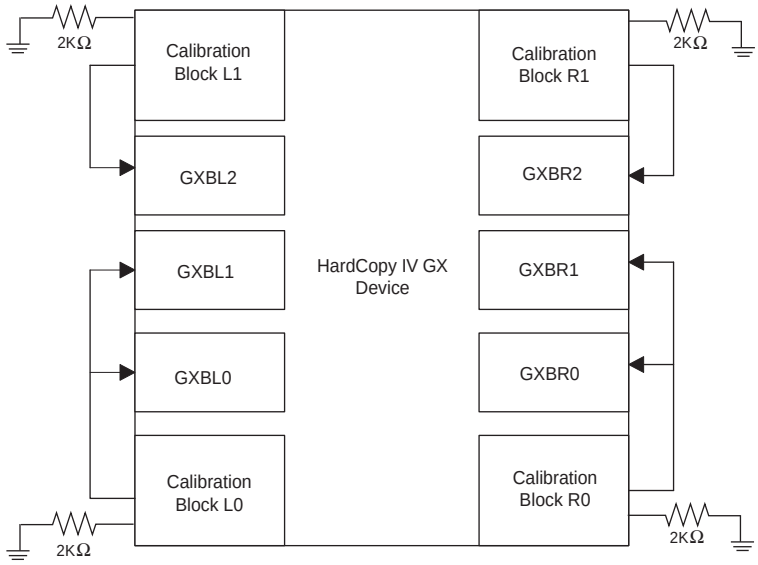
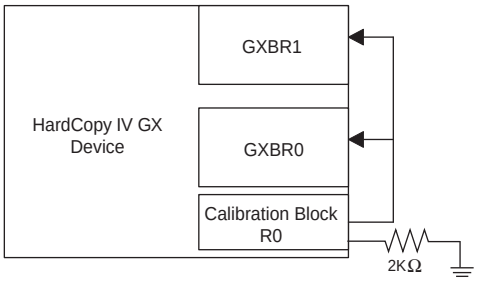


Figure 1–163 shows HardCopy IV GX device families that have two transceiver blocks only on the right side of the device.

Figure 1–163. Calibration Two Transceiver Blocks, Right Side Only



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

Calibration

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

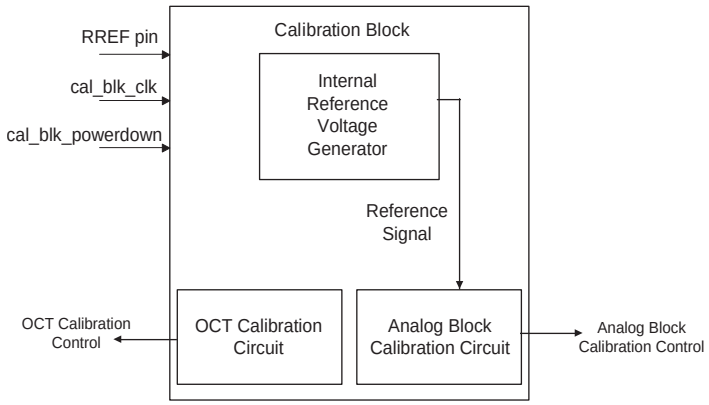
The OCT calibration circuit calibrates the OCT resistors present in the transceiver channels. You can enable the OCT resistors in the transceiver channels through the ALTGX MegaWizard Plug-In Manager.

You must connect a separate 2 kΩ (tolerance max ± 1%) external resistor on each RREF pin in the 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.

Input Signals to the Calibration Block

Figure 1–164 shows the required inputs to the calibration block. The ALTGX MegaWizard Plug-In Manager provides the `cal_blk_clk` and `cal_blk_powerdown` ports to control the calibration block.

Figure 1–164. Input Signals to the Calibration Blocks



- `cal_blk_clk`—you must use the `cal_blk_clk` port to provide input clock to the calibration clock. The frequency of `cal_blk_clk` must be within 10 MHz to 125 MHz (this range is preliminary. Final values will be available upon characterization). You can use dedicated clock routes such as the global or regional clock. If you do not have suitable input reference clock or dedicated clock routing resources available, use divide-down logic from the core fabric to generate a slow clock and use local clocking routing. Drive the `cal_blk_clk` port of all ALTGX instances that are associated with the same calibration block from the same input pin or logic.
- `cal_blk_powerdown`—you can perform calibration multiple times by using the `cal_blk_powerdown` port available through the ALTGX MegaWizard Plug-In Manager. Assert this signal for approximately 500 ns (this is preliminary. Final values will be available upon characterization). Following de-assertion of `cal_blk_powerdown`, the calibration block restarts the calibration process. Drive the `cal_blk_powerdown` port of all ALTGX instances that are associated with the same calibration block from the same input pin or logic.

Document Revision History

Table 1–55 shows the revision history for this chapter.

Table 1–55. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
June 2009, v1.0	Initial release.	—

2. HardCopy IV GX Dynamic Reconfiguration

HIV53002-1.0

Introduction

Dynamic reconfiguration is a feature available for HardCopy® IV GX transceivers. Each transceiver channel has multiple physical medium attachment (PMA) controls that you can program to achieve the desired bit error ratio (BER) for your system. When you enable the dynamic reconfiguration feature, you can reconfigure the PMA controls, functional blocks, CMU phased-locked loops (PLLs), receiver clock data recovery (CDR), and input reference clocks of a transceiver channel without powering down other transceiver channels or the core fabric logic of the device.

This chapter contains the following sections:

Conventions Used in this Chapter

The following conventions are used in this chapter:

- ALTGX_RECONFIG Instance—This term represents the dynamic reconfiguration controller instance generated by the ALTGX_RECONFIG MegaWizard™ Plug-In Manager. This term is used when the various inputs, outputs, and connections to the controller are explained.
- ALTGX Instance—This term represents the transceiver instance generated by the ALTGX MegaWizard Plug-In Manager. This term is used when the various inputs, outputs, and connections to the transceiver channels are explained.

- Alternate transmitter PLL—This term refers to one of the two CMU PLLs of a transceiver block. It refers to the CMU PLL configured in the **Reconfig Alt PLL** screen of the ALTGX MegaWizard Plug-In Manager.
- Channel and TX PLL select/reconfig—This term refers to the three dynamic reconfiguration modes: CMU PLL reconfiguration, Channel and CMU PLL reconfiguration, and Channel Reconfiguration with TX PLL select.
- CMU channel—This term refers to the CMU PLLs of a transceiver block configured as PMA-only channels.
- Dynamic Reconfiguration Controller—This term represents the dynamic reconfiguration controller. This term is used when a concept related to the controller is explained.
- Logical Channel Addressing—This term is used whenever the concept of logical channel addressing is explained. This term does not refer to the `logical_channel_address` port or the **Use 'logical_channel_address' port for Analog controls reconfiguration** option available in the ALTGX_RECONFIG MegaWizard Plug-In Manager.
- Logical reference index—This term refers to the logical identification value of 0 or 1 assigned to the main transmitter PLL and the alternate transmitter PLL. You set this value in the **Reconfig Clks 1** and **Reconfig Alt PLL** screens of the ALTGX MegaWizard Plug-In Manager.
- logical tx pll—This term refers to the logical reference index value of the transmitter PLLs stored in the **.mif**.
- Main transmitter PLL—This term refers to one of the two CMU PLLs of a transceiver block. It refers to the CMU PLL configured in the **General** screen of the ALTGX MegaWizard Plug-In Manager.
- Memory Initialization File, also known as **.mif**—When you enable **.mif** generation in your design, a file with the extension **.mif** gets generated. This file contains information about the various ALTGX MegaWizard Plug-In Manager options you can set. Each word in the **.mif** is 16 bits wide. The dynamic reconfiguration controller writes information from the **.mif** into the transceiver channel, but only when you use the 'Channel and TX PLL select/reconfig' dynamic reconfiguration mode. For more information about implementing a **.mif** in a HardCopy ASIC, refer to the *HardCopy IV Device Family Overview* chapter in volume 1 of the *HardCopy IV Device Handbook*.
- PMA controls—This term represents **Analog controls (VOD, Pre-emphasis, Manual Equalization)** as displayed in both the ALTGX and ALTGX_RECONFIG MegaWizard Plug-In Managers.
- PMA-only channels—This term refers to both the CMU channels as well as the regular transceiver channels with only the PMA blocks enabled. When you configure the ALTGX MegaWizard Plug-In Manager in Basic (PMA Direct) protocol in the **General** screen, all the channels get configured as PMA-only channels.
- Regular transceiver channel—This term refers to a transmitter channel or a receiver channel or a duplex channel that has both PMA and physical coding sublayer (PCS) blocks.

Dynamic Reconfiguration Modes

The different modes of dynamic reconfiguration are as follows:

- Offset cancellation for receiver channels
- PMA controls reconfiguration
- Transceiver channel reconfiguration

Offset cancellation for receiver channels mode and PMA controls reconfiguration mode are the same for both the regular transceiver channels and PMA-only channels.

These modes are described briefly in the following sections.

Offset Cancellation

The HardCopy IV GX device provides an offset cancellation circuit per receiver channel to counter the offset variations due to process, voltage, and temperature (PVT). These variations create an offset in the analog circuit voltages, pushing them out of the expected range. In addition to reconfiguring the transceiver channel, the dynamic reconfiguration controller performs offset cancellation on all receiver channels connected to it on power up.

The **Offset cancellation for Receiver channels** option is automatically enabled in both the ALTGX and ALTGX_RECONFIG MegaWizard Plug-In Managers for **Receiver and Transmitter** and **Receiver only** configurations. It is not available for **Transmitter only** configurations. For **Receiver and Transmitter** and **Receiver only** configurations, you must connect the necessary interface signals between the ALTGX_RECONFIG and ALTGX (with receiver channels) instances. For more information, refer to "Offset Cancellation Control for Receiver Channels" on page 2–43.

For proper device operation, you must always connect the ALTGX_RECONFIG and ALTGX (with receiver channels) instances.

PMA Controls Reconfiguration

You can dynamically reconfigure the following PMA controls:

- Pre-emphasis settings
- Equalization settings
- DC gain settings
- Voltage output differential (V_{OD}) settings

For more information, refer to "PMA Controls Reconfiguration" on page 2–52.

Transceiver Channel Reconfiguration Modes

Each transceiver block has four regular transceiver channels and two CMU channels. The regular transceiver channels have both PMA and PCS blocks. The CMU channels are CMU PLLs configured in Basic (PMA Direct) functional mode as PMA-only channels. Therefore, the CMU channels have only PMA blocks.

Regular Transceiver Channels

For the regular transceiver channels, dynamic reconfiguration involves the reconfiguration of the following:

- Transceiver channel functional mode
- Transceiver channel data rate switch
- Transceiver channel functional mode and data rate switch

However, the following dynamic reconfigurations cannot be achieved for the regular transceiver channels in the HardCopy IV GX device:

- Mode switch to and from any ×4 and ×8 configurations
- Prototype design with Stratix IV GX device is not backward compatible with Stratix GX or Stratix II GX devices
- Testability features (Pseudo-Random Binary Sequence [PRBS] and Built-In Self Test [BIST])
- Mode switch to and from any ×N Basic (PMA Direct) configuration
- Mode switch to and from any ×1 Basic (PMA Direct) configuration

For more information about dynamic reconfiguration of the CMU channels, refer to "PMA Controls Reconfiguration" on page 2–52.

Depending on how you want to reconfigure a transceiver channel, the transceiver channel reconfiguration is further classified into the following dynamic reconfiguration modes:

- Data rate division in TX

  Data rate division in TX mode is not available for non-Basic (PMA Direct) bonded configurations, Basic (PMA Direct) ×1, and Basic (PMA Direct) ×N configurations. For more information, refer to "Data Rate Division in TX Mode" on page 2–63.

- Channel and TX PLL select/reconfig
  - Channel and CMU PLL reconfiguration
  - Channel reconfiguration with TX PLL select
  - CMU PLL reconfiguration

  The modes grouped under Channel and TX PLL select/reconfig mode are not available for bonded non-Basic (PMA Direct) configurations, Basic (PMA Direct) ×1, and Basic (PMA Direct) ×N configurations. For more information, refer to "Channel and TX PLL Select/Reconfig Modes" on page 2–68.

Offset cancellation is enabled by default. All other dynamic reconfiguration modes are available for selection through the `reconfig_mode_sel[2:0]` signal. Based on which part of the transceiver channel you want to reconfigure, you can select one or more of these dynamic reconfiguration modes.

The `reconfig_mode_sel[2:0]` signal is available as an input to the dynamic reconfiguration controller only when you select multiple dynamic reconfiguration modes.

Based on the value you set at the `reconfig_mode_sel[2:0]` signal, the respective dynamic reconfiguration mode is enabled. The `reconfig_mode_sel[2:0]` signal is 3 bits wide.

Table 2–1 shows the various dynamic reconfiguration modes supported by the regular transceiver channels and CMU channels, and the value that you need to set at the `reconfig_mode_sel[2:0]` input port to activate one of the selected dynamic reconfiguration modes.

Table 2–1. Dynamic Reconfiguration Modes Supported by Regular Transceiver Channels and CMU Channels

Dynamic Reconfiguration Mode	Regular Transceiver Channels	CMU Channels	reconfig_mode_sel[2:0] *(1)*
PMA Controls Reconfiguration	Yes	Not Supported	3'b000
Data Rate Division in TX	Yes	Not Supported	3'b011
Channel and TX PLL select/reconfig			
CMU PLL Reconfiguration	Yes	Not Supported	3'b100
Channel and CMU PLL Reconfiguration	Yes	Not Supported	3'b101
Channel Reconfiguration with TX PLL Select	Yes	Not Supported	3'b110
Not Supported	Not Supported	Not Supported	3'b111

Note to Table 2–1:

(1) `reconfig_mode_sel [2:0]` = 3'b001 and 3'b010 are not supported.

The `reconfig_mode_sel[2:0]` input is available when you select more than one dynamic reconfiguration mode in the ALTGX_RECONFIG MegaWizard Plug-In Manager. The default value for this `reconfig_mode_sel[2:0]` input is **000**.

Quartus II MegaWizard Plug-In Manager Interfaces to Support Dynamic Reconfiguration

The HardCopy IV GX device provides two MegaWizard Plug-In Manager interfaces to support dynamic reconfiguration: ALTGX and ALTGX_RECONFIG.

This section provides information about the dynamic reconfiguration options available in the ALTGX and ALTGX_RECONFIG MegaWizard Plug-In Managers.

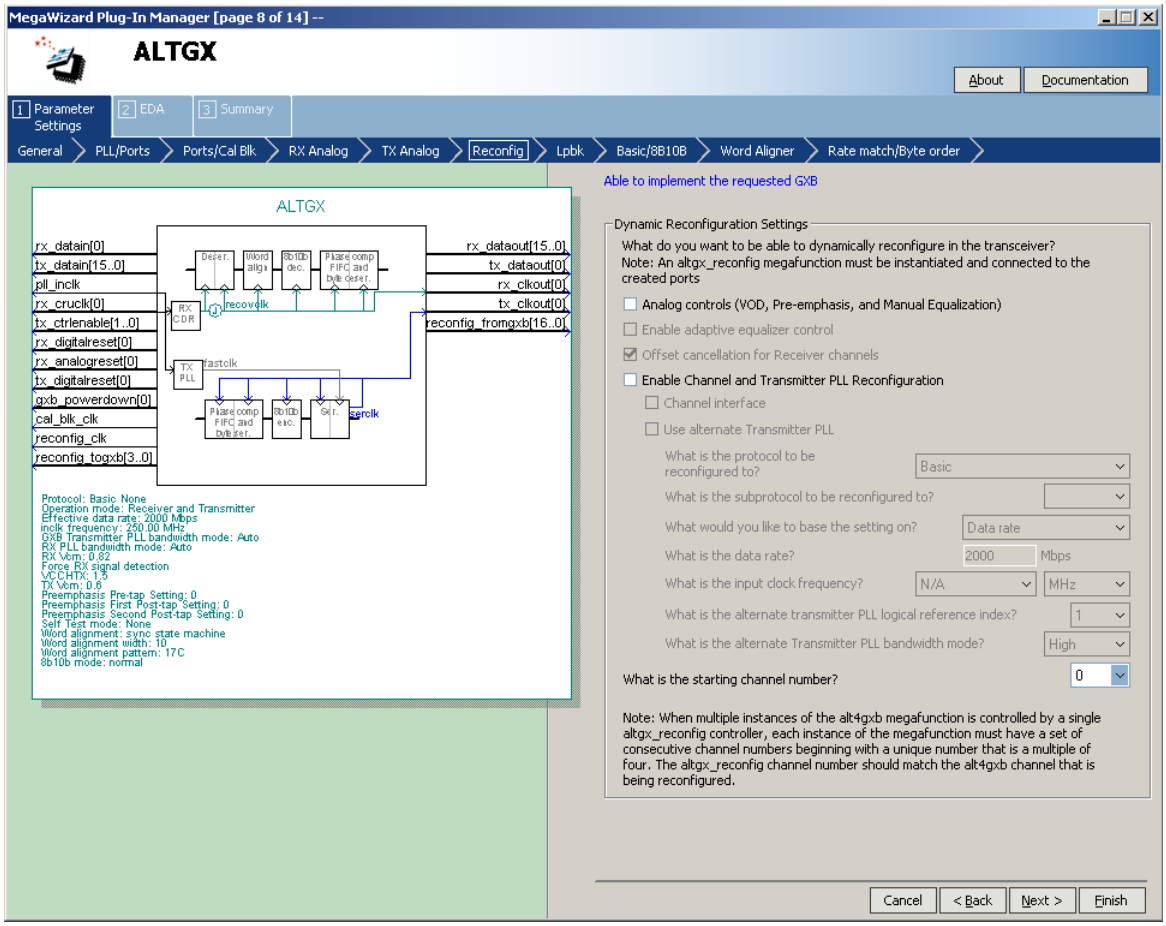
ALTGX MegaWizard Plug-In Manager

The ALTGX MegaWizard Plug-In Manager provides the following options in the **Reconfig** screen to enable the various dynamic reconfiguration modes (shown in Figure 2–1):

- The **Analog controls (VOD, Pre-emphasis, and Manual Equalization)** option for the PMA controls reconfiguration mode.
- The **Offset cancellation for Receiver channels** option is enabled by default and grayed out for the offset cancellation mode.
- The **Enable Channel and Transmitter PLL Reconfiguration** option for the Data Rate Division in TX, CMU PLL reconfiguration, Channel and CMU PLL reconfiguration, and Channel reconfiguration with TX PLL select modes.

Select the options in the preceding list to enable the corresponding dynamic reconfiguration modes.

Figure 2–1. Dynamic Reconfiguration Settings in the ALTGX MegaWizard Plug-In Manager



ALTGX_RECONFIG MegaWizard Plug-In Manager

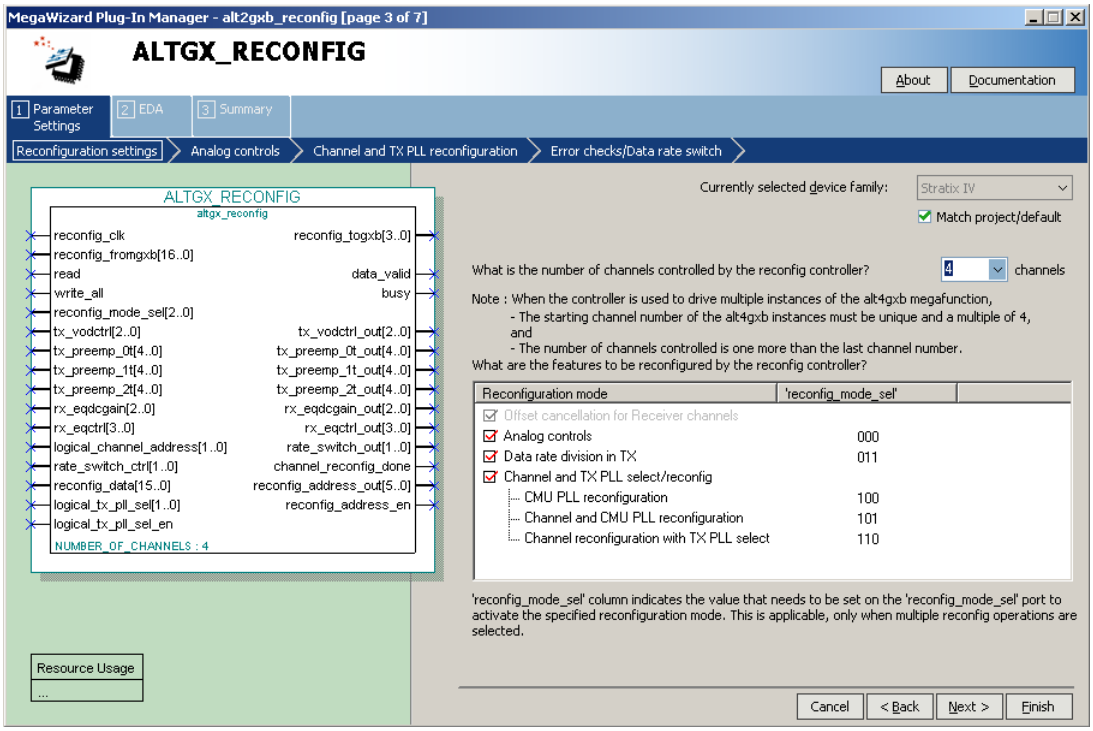
The Quartus® II software provides the ALTGX_RECONFIG MegaWizard Plug-In Manager to instantiate the dynamic reconfiguration controller.

The following options are available in the **Reconfiguration Settings** screen to enable the various dynamic reconfiguration modes (shown in Figure 2–2):

- The **Offset cancellation for Receiver channels** option is enabled by default and grayed out for the offset cancellation mode.
- The **Analog controls** option for the PMA controls reconfiguration mode. To dynamically reconfigure the PMA controls, enable at least one of the PMA control ports in the **Analog controls** screen.
- The **Data rate division in TX** option for the Data Rate Division in TX mode.
- The **Channel and TX PLL select/reconfig** option for the CMU PLL reconfiguration, Channel and CMU PLL reconfiguration, and Channel reconfiguration with TX PLL select modes.

Select the options listed above to enable the corresponding dynamic reconfiguration modes.

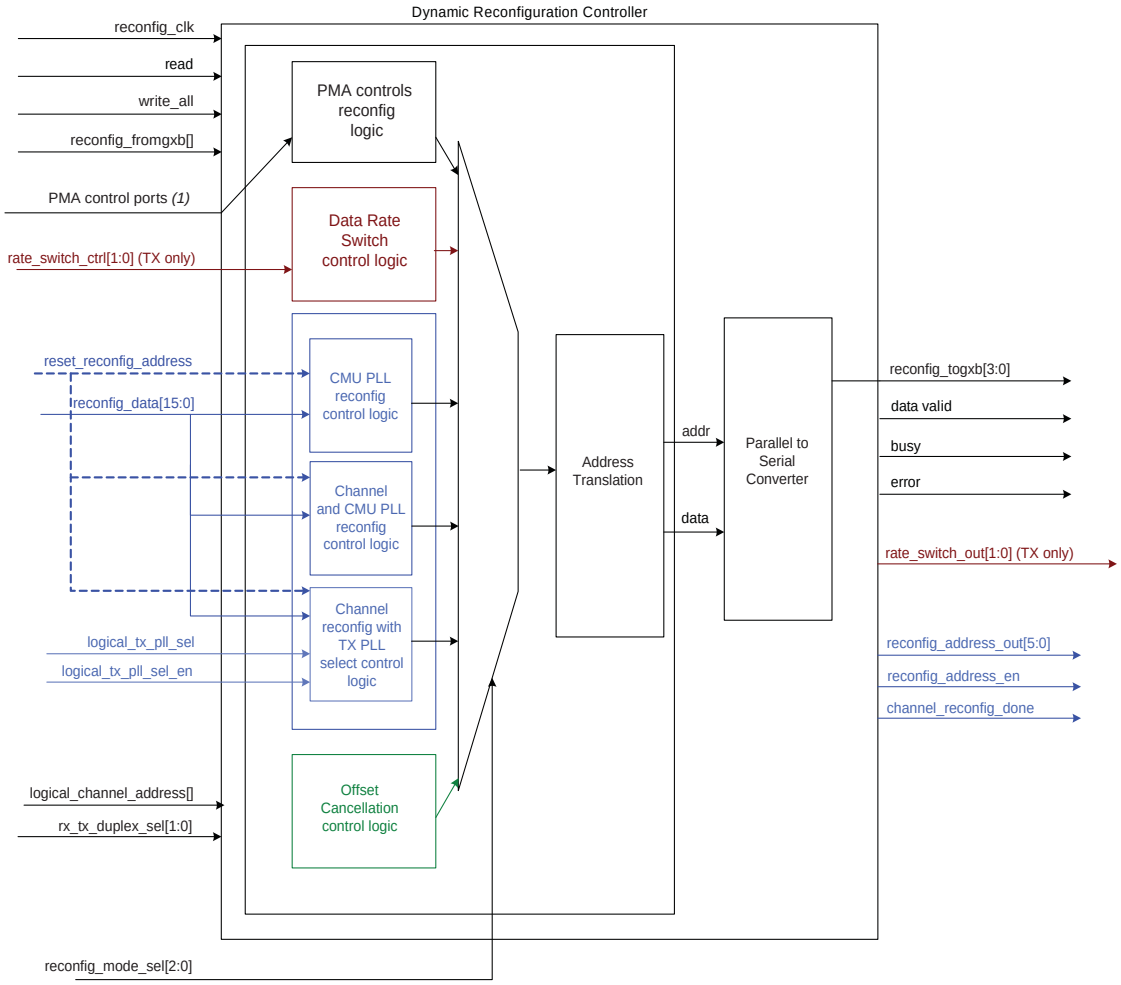
Figure 2–2. Dynamic Reconfiguration Settings in the ALTGX_RECONFIG MegaWizard Plug-In Manager



Dynamic Reconfiguration Controller Architecture

The dynamic reconfiguration controller is a soft IP that utilizes the core fabric resources. You can use only one controller per transceiver block. You cannot use the dynamic reconfiguration controller to control multiple HardCopy IV GX devices or any off-chip interfaces. Figure 2–3 shows the conceptual view of the dynamic reconfiguration controller architecture.

Figure 2–3. Block Diagram of the Dynamic Reconfiguration Controller



Note to Figure 2–3:

(1) The PMA control ports consist of the V_{OD} controls, pre-emphasis controls, DC gain controls, and manual equalization controls. For a detailed description of all the inputs and outputs of the ALTGX_RECONFIG instance, refer to "Dynamic Reconfiguration Controller Port List" on page 2–11.

The dynamic reconfiguration controller has the following control logic modules:

- PMA controls reconfiguration control logic
- Offset cancellation control logic for receiver channels
- Data rate division control logic to the TX local divider
- Channel reconfiguration with TX PLL select/reconfig control logic
- CMU PLL reconfiguration control logic
- Channel and CMU PLL reconfiguration control logic
- Channel reconfiguration with TX PLL select control logic

For PMA controls reconfiguration, the dynamic reconfiguration control inputs to the controller are translated into address and data bus. The address and data bus are then converted into serial data and forwarded to the transceiver channel selected.

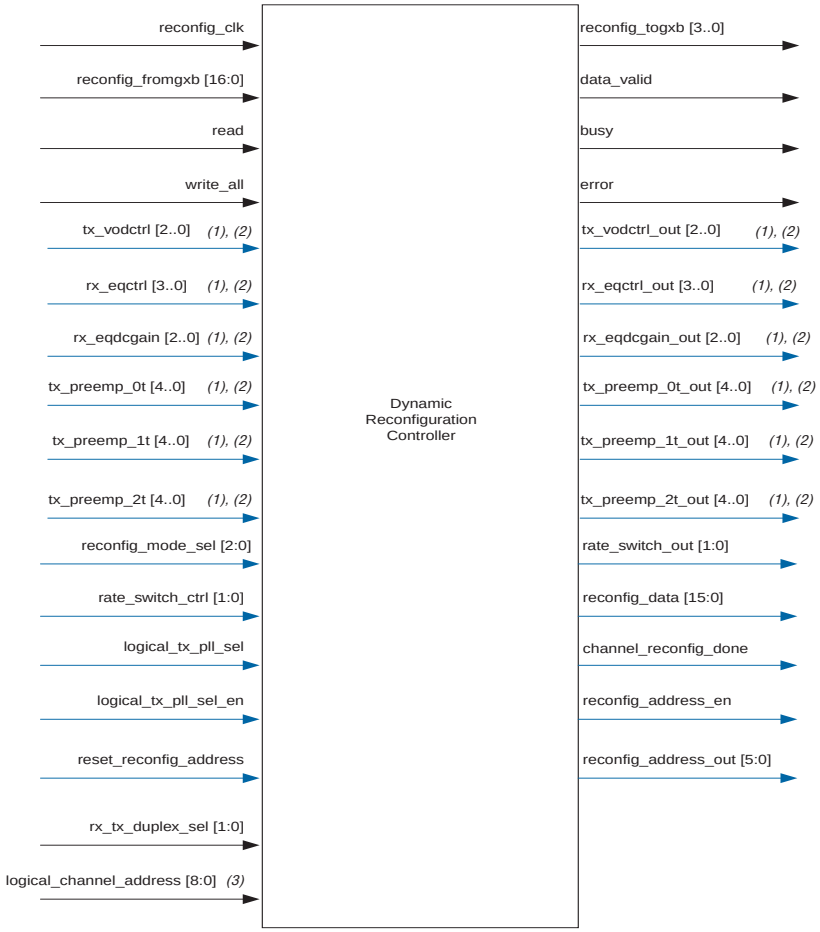
For the data rate division control logic to the TX local divider, the `rate_switch_ctrl[1:0]` input to the controller is translated into address and data bus. The address and data bus are then converted into serial data and forwarded to the local divider in the transmitter channel.

For the CMU PLL reconfiguration, Channel and CMU PLL reconfiguration, and Channel reconfiguration with TX PLL select modes, the dynamic reconfiguration controller receives 16-bit words from a **.mif** that you generate and sends this information to the transceiver channel selected. For more information regarding **.mif** generation, refer to ".mif Generation" on page 2–69.

Dynamic Reconfiguration Controller Interface

The dynamic reconfiguration controller interface consists of certain control input and output status signals. Figure 2–4 shows the dynamic reconfiguration interface list, which contains all the inputs and outputs to the dynamic reconfiguration controller.

Figure 2–4. Dynamic Reconfiguration Controller Interface



Notes to Figure 2–4:

(1) These ports assume that the dynamic reconfiguration controller is connected to a single channel in the design.

(2) These are the optional PMA control input signals and the optional PMA control output status signals. You must select at least one of these PMA control ports if you want to dynamically configure the PMA controls of a transceiver channel. For a detailed description of all the inputs and outputs of the ALTGX_RECONFIG instance, refer to "Dynamic Reconfiguration Controller Port List" on page 2–11.

(3) The `logical_channel_address` port is available for selection only when the number of channels controlled by the dynamic reconfiguration controller is more than one. The port is shown here to represent the complete port list.

Dynamic Reconfiguration Controller Port List

Table 2–2 describes the input control ports and output status ports of the dynamic reconfiguration controller.

Table 2–2. Dynamic Reconfiguration Controller Port List (ALTGX_RECONFIG Instance) (Part 1 of 9)

Port Name	Input/ Output	Description
Clock Inputs to ALTGX_RECONFIG Instance		
reconfig_clk	Input	The frequency range of this clock depends on the following transceiver channel configuration modes: ■ **Receiver only** (37.5 MHz to 50 MHz) ■ **Receiver and Transmitter** (37.5 MHz to 50 MHz) ■ **Transmitter only** (2.5 MHz to 50 MHz) For more information, refer to Table 2–3 on page 2–20. By default, the Quartus II software assigns a global clock resource to this port.
ALTGX and ALTGX_RECONFIG Interface Signals		
reconfig_fromgxb	Input	The width of this signal is determined by the value you set in the **What is the number of channels controlled by the reconfig controller?** option in the **Reconfiguration settings** screen. For more information, refer to "Connecting the reconfig_fromgxb and reconfig_togxb Ports" on page 2–40.
reconfig_togxb[3..0]	Output	The width of this signal is fixed to 4 bits. It is independent of the value you set in the **What is the number of channels controlled by the reconfig controller?** option in the **Reconfiguration settings** screen. For more information, refer to "Connecting the reconfig_fromgxb and reconfig_togxb Ports" on page 2–40.
Core Fabric and ALTGX_RECONFIG Interface Signals		
write_all	Input	Assert this signal for one reconfig_clk clock cycle to initiate a write transaction from the ALTGX_RECONFIG instance to the ALTGX instance. For more information, refer to "Dynamically Reconfiguring PMA Controls" on page 2–53.

Table 2–2. Dynamic Reconfiguration Controller Port List (ALTGX_RECONFIG Instance) (Part 2 of 9)

Port Name	Input/ Output	Description
busy	Output	This signal is used to indicate the busy status of the dynamic reconfiguration controller during: ■ Offset Cancellation—After the device powers up, this signal remains low for the first reconfig_clk clock cycle. It then is asserted and remains high when the dynamic reconfiguration controller performs offset cancellation on all the receiver channels connected to the ALTGX_RECONFIG instance. De-assertion of the busy signal indicates the successful completion of the offset cancellation process. For more information, refer to "Operation" on page 2–44. ■ PMA controls reconfiguration mode—This signal is high when the dynamic reconfiguration controller performs a read or write transaction. ■ All other dynamic reconfiguration modes—This signal is high when the dynamic reconfiguration controller writes the **.mif** into the transceiver channel.
read	Input	Assert this signal for one reconfig_clk clock cycle to initiate a read transaction. The read port is applicable only to the PMA controls Reconfiguration mode. The read port is available when you select **Analog controls** in the **Reconfiguration settings** screen and select at least one of the PMA control ports in the **Analog controls** screen. For more information, refer to "Dynamically Reconfiguring PMA Controls" on page 2–53.
data_valid	Output	The data_valid port is applicable only to PMA controls reconfiguration mode. This port indicates the validity of the data read from the transceiver by the dynamic reconfiguration controller. The current data on the output read ports is the valid data ONLY if data_valid is high. This signal is enabled when you enable at least one PMA control port used in read transactions, for example tx_vodctrl_out.
error	Output	This indicates that an unsupported operation is attempted. You can select this in the **Error checks/Data rate switch** screen. The dynamic reconfiguration controller de-asserts the busy signal and asserts the error signal for two reconfig_clk cycles when you attempt an unsupported operation. For more information, refer to the "Error Indication in the ALTGX_RECONFIG MegaWizard Plug-In Manager" on page 2–139.

Table 2–2. Dynamic Reconfiguration Controller Port List (ALTGX_RECONFIG Instance) (Part 3 of 9)

Port Name	Input/ Output	Description
`logical_channel_address [8:0]`	Input	The `logical_channel_address` port is enabled by the ALTGX_RECONFIG MegaWizard Plug-In Manager when you enable the **Use 'logical_channel_address' port for Analog controls reconfiguration** option in the **Analog controls** screen. The width of the `logical_channel_address` port depends on the value you set in the **What is the number of channels controlled by the reconfig controller?** option in the **Reconfiguration settings** screen. The `logical_channel_address` port can be enabled only when the number of channels controlled by the dynamic reconfiguration controller is more than one. For more information, refer to "Logical Channel Addressing of Regular Transceiver Channels" on page 2–23 and "Logical Channel Addressing of PMA-Only Channels" on page 2–32.
`rx_tx_duplex_sel[1:0]`	Input	This is a 2-bit wide signal. You can select this in the **Error checks/Data rate switch** screen. The advantage of using this optional port is that it allows you to reconfigure only the transmitter portion of a channel, even if the channel configuration is duplex. For a setting of: ■ `rx_tx_duplex_sel[1:0]` = 2'b00 => the transmitter and receiver portion of the channel is reconfigured. ■ `rx_tx_duplex_sel[1:0]` = 2'b01 => the receiver portion of the channel is reconfigured. ■ `rx_tx_duplex_sel[1:0]` = 2'b10 => the transmitter portion of the channel is reconfigured.

Table 2–2. Dynamic Reconfiguration Controller Port List (ALTGX_RECONFIG Instance) (Part 4 of 9)

Port Name	Input/ Output	Description
Analog Settings Control/Status Signals		
`tx_vodctrl[2..0]` *(1)*	Input	This is an optional transmit buffer V_{OD} control signal. It is 3 bits per transmitter channel. The number of settings varies based on the transmit buffer supply setting and the termination resistor setting on the **TX Analog** screen of the ALTGX MegaWizard Plug-In Manager. The width of this signal is fixed to 3 bits if you enable either the **Use 'logical_channel_address' port for Analog controls reconfiguration** option or the **Use same control signal for all the channels** option in the **Analog controls** screen. Otherwise, the width of this signal is 3 bits per channel. For more information, refer to "Dynamically Reconfiguring PMA Controls" on page 2–53. The following shows the V_{OD} values corresponding to the `tx_vodctrl` settings for 100-Ω termination. For more information, refer to the *Programmable Output Differential Voltage* section of the *HardCopy IV GX Transceiver Architecture* chapter in volume 3 of the *HardCopy IV Device Handbook*. `tx_vodctrl[2:0]` — V_{OD} (mV) for 1.4 V V_{CCH} 3'b000 — 200 3'b001 — 400 3'b010 — 600 3'b011 — 700 3'b100 — 800 3'b101 — 900 3'b110 — 1000 3'b111 — 1200

Table 2–2. Dynamic Reconfiguration Controller Port List (ALTGX_RECONFIG Instance) (Part 5 of 9)

Port Name	Input/ Output	Description
`tx_preemp_0t[4..0]` *(1)*	Input	This is an optional pre-emphasis control for pre-tap for the transmit buffer. Depending on what value you set at this input, the controller dynamically writes the value to the pre-emphasis control register of the transmit buffer. This signal controls both pre-emphasis positive and its inversion. The width of this signal is fixed to 5 bits if you enable either the **Use 'logical_channel_address' port for Analog controls reconfiguration** option or the **Use same control signal for all the channels** option in the **Analog controls** screen. Otherwise, the width of this signal is 5 bits per channel. For more information, refer to "Dynamically Reconfiguring PMA Controls" on page 2–53. The following values are the legal settings allowed for this signal: 0 represents 0 1-15 represents -15 to -1 16 represents 0 17 - 31 represents 1 to 15 In PCI Express (PIPE) configuration, set `tx_preemp_0t[4:0]` to **5'b00000** when you do a rate switch from Gen 1 to Gen 2 mode. This is to ensure that `tx_preemp_0t[4:0]` does not add to the signal boost, when `tx_pipemargin`, `tx_pipeswing`, and `tx_pipedeemph` take affect in PCI Express (Gen 2) mode. For more information, refer to the *Programmable Pre-Emphasis* section of the *HardCopy IV GX Transceiver Architecture* chapter in volume 3 of the *HardCopy IV Device Handbook*.
`tx_preemp_1t[4..0]` *(1)*	Input	This is an optional pre-emphasis write control for the first post-tap for the transmit buffer. Depending on what value you set at this input, the controller dynamically writes the value to the first post-tap control register of the transmit buffer. The width of this signal is fixed to 5 bits if you enable either the **Use 'logical_channel_address' port for Analog controls reconfiguration** option or the **Use same control signal for all the channels** option in the **Analog controls** screen. Otherwise, the width of this signal is 5 bits per channel. For more information, refer to "Dynamically Reconfiguring PMA Controls" on page 2–53 for additional details and the *Programmable Pre-Emphasis* section of the *HardCopy IV GX Transceiver Architecture* chapter in volume 3 of the *HardCopy IV Device Handbook*.

Table 2–2. Dynamic Reconfiguration Controller Port List (ALTGX_RECONFIG Instance) (Part 6 of 9)

Port Name	Input/ Output	Description
`tx_preemp_2t[4..0]` *(1)*	Input	This is an optional pre-emphasis write control for the second post-tap for the transmit buffer. This signal controls both pre-emphasis positive and its inversion. Depending on what value you set at this input, the controller dynamically writes the value to the pre-emphasis control register of the transmit buffer. The width of this signal is fixed to 5 bits if you enable either the **Use 'logical_channel_address' port for Analog controls reconfiguration** option or the **Use same control signal for all the channels** option in the **Analog controls** screen. Otherwise, the width of this signal is 5 bits per channel. For more information, refer to "Dynamically Reconfiguring PMA Controls" on page 2–53. The following values are the legal settings allowed for this signal: 0 represents 0 1-15 represents -15 to -1 16 represents 0 17-31 represents 1 to 15 In PCI Express (PIPE) configuration, set `tx_preemp_2t[4:0]` to **5'b00000** when you do a rate switch from Gen 1 to Gen 2 mode. This is to ensure that `tx_preemp_2t[4:0]` does not add to the signal boost when `tx_pipemargin`, `tx_pipeswing`, and `tx_pipedeemph` take affect in PCI Express (Gen 2) mode. For more information, refer to the *Programmable Pre-Emphasis* section of the *HardCopy IV GX Transceiver Architecture* chapter in volume 3 of the *HardCopy IV Device Handbook*.
`rx_eqctrl[3..0]` *(1)*	Input	This is an optional write control to write an equalization control value for the receive side of the PMA. The width of this signal is fixed to 4 bits if you enable either the **Use 'logical_channel_address' port for Analog controls reconfiguration** option or the **Use same control signal for all the channels** option in the **Analog controls** screen. Otherwise, the width of this signal is 4 bits per channel. For more information, refer to "Dynamically Reconfiguring PMA Controls" on page 2–53 and the *Programmable Equalization and DC Gain* section of the *HardCopy IV GX Transceiver Architecture* chapter in volume 3 of the *HardCopy IV Device Handbook*.

Table 2–2. Dynamic Reconfiguration Controller Port List (ALTGX_RECONFIG Instance) (Part 7 of 9)

Port Name	Input/ Output	Description
rx_eqdcgain[2..0] *(1), (2)*	Input	This is an optional equalizer DC gain write control. The width of this signal is fixed to 3 bits if you enable either the **Use 'logical_channel_address' port for Analog controls reconfiguration** option or the **Use same control signal for all the channels** option in the **Analog controls** screen. Otherwise, the width of this signal is 3 bits per channel. For more information, refer to "Dynamically Reconfiguring PMA Controls" on page 2–53. The following values are the legal settings allowed for this signal: 3'b000 => 0 dB 3'b001 => 3 dB 3'b010 => 6 dB 3'b011 => 9 dB 3'b100 => 12 dB All other values => N/A For more information, refer to the *Programmable Equalization and DC Gain* section of the *HardCopy IV GX Transceiver Architecture* chapter in volume 3 of the *HardCopy IV Device Handbook*.
tx_vodctrl_out[2..0]	Output	This is an optional transmit V_{OD} read control signal. This signal reads out the value written into the V_{OD} control register. The width of this output signal depends on the number of channels controlled by the dynamic reconfiguration controller.
tx_preemp_0t_out[4..0]	Output	This is an optional pre-tap, pre-emphasis read control signal. This signal reads out the value written by its input control signal. The width of this output signal depends on the number of channels controlled by the dynamic reconfiguration controller.
tx_preemp_1t_out[4..0]	Output	This is an optional first post-tap, pre-emphasis read control signal. This signal reads out the value written by its input control signal. The width of this output signal depends on the number of channels controlled by the dynamic reconfiguration controller.
tx_preemp_2t_out[4..0]	Output	This is an optional second post-tap pre-emphasis read control signal. This signal reads out the value written by its input control signal. The width of this output signal depends on the number of channels controlled by the dynamic reconfiguration controller.
rx_eqctrl_out[3..0]	Output	This is an optional read control signal to read the setting of equalization setting of the ALTGX instance. The width of this output signal depends on the number of channels controlled by the dynamic reconfiguration controller.
rx_eqdcgain_out[2..0]	Output	This is an optional equalizer DC gain read control signal. This signal reads out the settings of the ALTGX instance DC gain. The width of this output signal depends on the number of channels controlled by the dynamic reconfiguration controller.

Table 2–2. Dynamic Reconfiguration Controller Port List (ALTGX_RECONFIG Instance) (Part 8 of 9)

Port Name	Input/ Output	Description
Transceiver Channel Reconfiguration Control/Status Signals		
reconfig_mode_sel[2:0]	Input	Set the following values at this signal to activate the appropriate dynamic reconfiguration mode: 3'b000 – PMA controls reconfiguration mode. This is the default value. 3'b011 – Data Rate Division in TX mode 3'b100 – CMU PLL reconfiguration mode 3'b101 – Channel and CMU PLL reconfiguration mode 3'b110 – Channel reconfiguration with TX PLL select mode 3'b111 – Not supported
reconfig_address_out[5:0]	Output	This signal is always available for you to select in the **Channel and TX PLL reconfiguration** screen. This signal is applicable only in the dynamic reconfiguration modes grouped under **Channel and TX PLL select/reconfig** option. This signal represents the current address used by the ALTGX_RECONFIG instance when writing the **.mif** into the transceiver channel. This signal increments by 1, from 0 to last address, then starts at 0 again. You can use this signal to indicate the end of all the **.mif** write transactions (reconfig_address_out[5:0] changes from the last address to 0 at the end of all the **.mif** write transactions).
reconfig_address_en	Output	This is an optional signal you can select in the **Channel and TX PLL reconfiguration** screen. This signal is applicable only in dynamic reconfiguration modes grouped under the **Channel and TX PLL select/reconfig** option. The dynamic reconfiguration controller asserts reconfig_address_en to indicate that reconfig_address_out[5:0] has changed. This signal gets asserted only after the dynamic reconfiguration controller completes writing one 16-bit word of the **.mif**.
reset_reconfig_address	Input	This is an optional signal you can select in the **Channel and TX PLL reconfiguration** screen. This signal is applicable only in dynamic reconfiguration modes grouped under the **Channel and TX PLL select/reconfig** option. Enable this signal and assert it for one reconfig_clk clock cycle if you want to reset the reconfiguration address used by the ALTGX_RECONFIG instance during reconfiguration.
reconfig_data[15:0]	Input	This signal is applicable only in the dynamic reconfiguration modes grouped under the **Channel and TX PLL select/reconfig** option. This is a 16-bit word carrying the reconfiguration information. It is stored in a **.mif** file that you need to generate. The ALTGX_RECONFIG instance requires that you provide reconfig_data[15:0]on every **.mif** write transaction using the write_all signal.

Table 2–2. Dynamic Reconfiguration Controller Port List (ALTGX_RECONFIG Instance) (Part 9 of 9)

Port Name	Input/ Output	Description
rate_switch_ctrl[1:0]	Input	This signal is available when you select Data Rate Division in TX mode. Based on the value you set here, the divide-by setting of the local divider in the transmitter channel gets modified. The legal values for this port are: 2'b00 – Divide by 1 2'b01 – Divide by 2 2'b10 – Divide by 4 2'b11 – Not supported For more information, refer to "Data Rate Division in TX Mode" on page 2–63.
rate_switch_out[1:0]	Input	This signal is available when you select Data Rate Division in TX mode. You can read the existing local divider settings of a transmitter channel at this port. The decoding for this signal is listed below: 2'b00 – Division of 1 2'b01 – Division of 2 2'b10 – Division of 4 2'b11– Not supported For more information, refer to "Data Rate Division in TX: Operation" on page 2–66.
logical_tx_pll_sel	Input	At this port you specify the identity of the TX PLL you want to reconfigure. You can also specify the identity of the TX PLL that you want the transceiver channel to listen to. When you enable this signal, the value set at this signal overwrites the logical_tx_pll value contained in the **.mif**. The value at this port needs to be held at a constant logic level until reconfiguration is done.
logical_tx_pll_sel_en	Input	If you want to use the logical_tx_pll_sel port only under some conditions and use the logical_tx_pll value contained in the **.mif** otherwise, enable this optional logical_tx_pll_sel_en port. Only when logical_tx_pll_sel_en is enabled and set to **1** does the dynamic reconfiguration controller use logical_tx_pll_sel to identify the TX PLL. The value at this port needs to be held at a constant logic level until reconfiguration is done.
channel_reconfig_done	Output	This signal goes high for one reconfig_clk clock cycle to indicate that the dynamic reconfiguration controller has finished writing all the words of the **.mif**. This signal is applicable only in Channel and CMU PLL reconfiguration and Channel reconfiguration with TX PLL select modes.

Notes to Table 2–2:

(1) Not all combinations of the input bits are legal values.

(2) In PCI Express (PIPE) mode, this input needs to be tied to 001 to be PCI E-compliant.

Clock Requirements for the ALTGX Instance and ALTGX_RECONFIG Instance

This section describes the dynamic reconfiguration clock requirements for both the ALTGX instance (transceiver instance) and the ALTGX_RECONFIG instance (dynamic reconfiguration controller instance).

Clock Requirements for the ALTGX Instance

For all functional mode configurations except PCI Express (PIPE) configurations, you must connect the `reconfig_clk` input port of the ALTGX instance to the same clock that is connected to the `reconfig_clk` input port of the ALTGX_RECONFIG instance.

The offset cancellation circuit requires that the minimum frequency of its input clock is ≥ 37.5 MHz. Therefore, the ALTGX configurations, including a receiver, require that the minimum frequency of `reconfig_clk` (the clock used by the offset cancellation circuit) is 37.5 MHz.

For PCI Express (PIPE) configurations only, the `fixedclk` input port is used to clock the offset cancellation circuit instead of the `reconfig_clk` input port. Therefore, the `reconfig_clk` input port frequency range can vary from 2.5 MHz to 50 MHz in PCI Express (PIPE) configurations.

Table 2–3 shows the range of frequency values for the `reconfig_clk` port.

Clock Requirements for the ALTGX_RECONFIG Instance

You must connect the `reconfig_clk` input port of the ALTGX_RECONFIG instance to the same clock that is connected to the `reconfig_clk` input port of the ALTGX instance.

Table 2–3 shows the range of frequency values of the `reconfig_clk` input port for **Receiver only**, **Receiver and Transmitter**, and **Transmitter only** configuration modes of the ALTGX instance. Table 2–3 also shows the clock requirements for the `reconfig_clk` input port to the ALTGX_RECONFIG instance based on ALTGX configurations.

Based on ALTGX configurations (**Receiver only**, **Transmitter only**, **Receiver and Transmitter** configurations) controlled by the ALTGX_RECONFIG instance, select the fastest `reconfig_clk` frequency value. This satisfies both the offset cancellation control for receiver channels and the dynamic reconfiguration of the transmitter and receiver channels.

Table 2–3. reconfig_clk Settings for ALTGX Instance *(Note 1)*

ALTGX Instance Configuration	reconfig_clk Frequency Range
Receiver and Transmitter reconfiguration configuration	37.5 MHz to 50 MHz
Receiver only reconfiguration configuration	37.5 MHz to 50 MHz
Transmitter only reconfiguration configuration	2.5 MHz to 50 MHz

Note to Table 2–3:

(1) Altera recommends the `reconfig_clk` signal be driven on a global clock resource.

Interfacing ALTGX_RECONFIG and ALTGX Instances

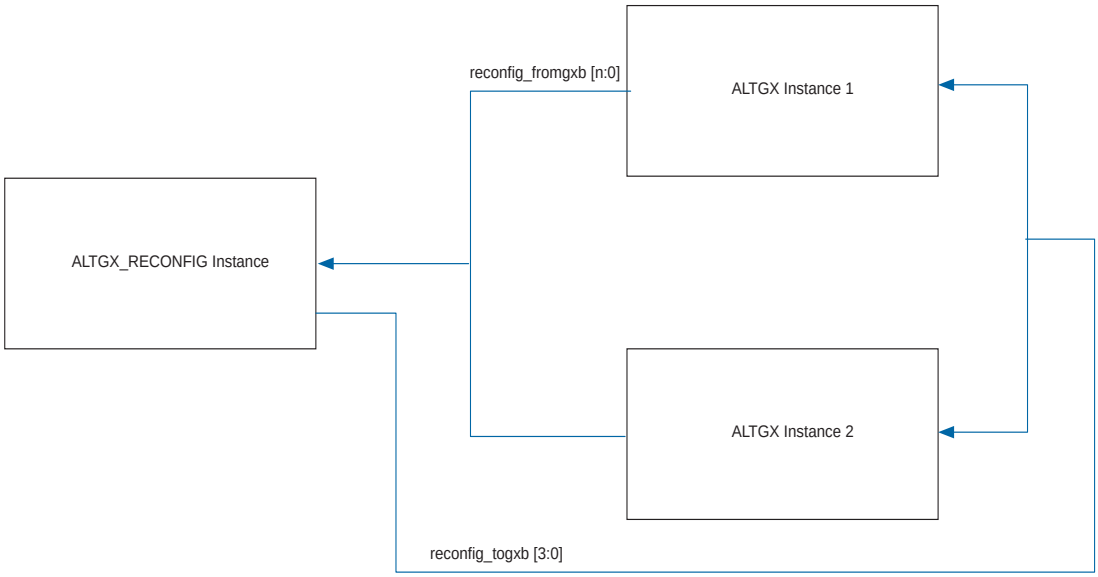
This section describes the various dynamic reconfiguration settings available in the ALTGX_RECONFIG and ALTGX MegaWizard Plug-In Managers and how to set them. It also provides information about the interface signals and connections between the ALTGX_RECONFIG and ALTGX instances.

There are two ways to connect the ALTGX_RECONFIG instance to the ALTGX instance in your design:

- Single dynamic reconfiguration controller—You can use a single ALTGX_RECONFIG instance to control all the ALTGX instances in your design.

Figure 2–5 shows a block diagram of a single ALTGX_RECONFIG instance controlling multiple ALTGX instances.

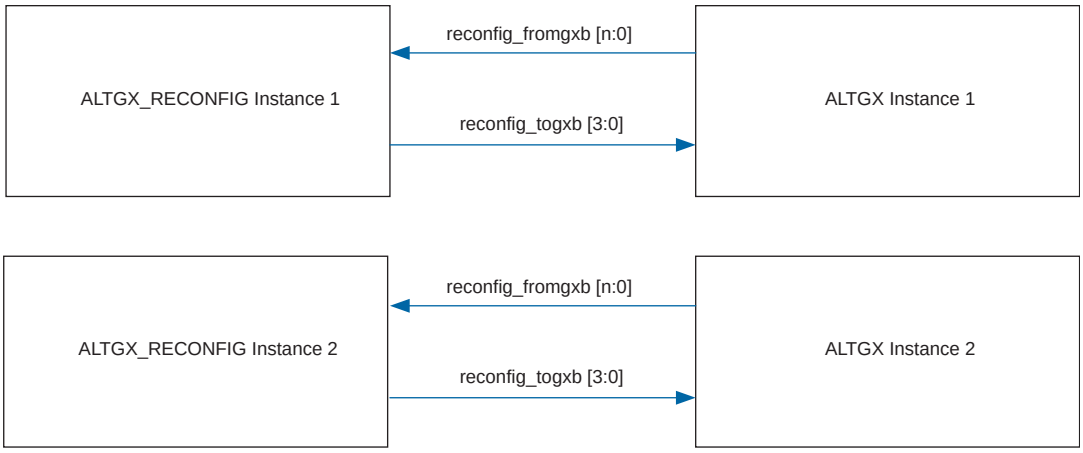
Figure 2–5. Block Diagram of a Single Dynamic Reconfiguration Controller in a Design



- Multiple dynamic reconfiguration controllers—Your design can have multiple ALTGX_RECONFIG instances, where every ALTGX instance is controlled by its own ALTGX_RECONFIG instance.

Figure 2–6 shows a block diagram of multiple ALTGX_RECONFIG instances, each controlling a single ALTGX instance.

Figure 2–6. Block Diagram of Multiple Dynamic Reconfiguration Controllers in a Design



To enable dynamic reconfiguration of a transceiver channel, you must understand the following:

- Logical Channel Addressing—The dynamic reconfiguration controller identifies a transceiver channel by using the logical channel address. The **What is the starting channel number?** option in the **Reconfig screen** of the ALTGX MegaWizard Plug-In Manager allows you to set the logical channel address of all the channels within the ALTGX instance. This concept is explained in detail in "Logical Channel Addressing" on page 2–23.
- Total number of channels controlled by the ALTGX_RECONFIG instance—Every dynamic reconfiguration controller in a design might be connected to either a single ALTGX instance or multiple ALTGX instances. Depending on the number of channels within each of these ALTGX instances, you must set the total number of channels controlled by the dynamic reconfiguration controller in the ALTGX_RECONFIG MegaWizard Plug-In Manager. This concept is explained in "Total Number of Channels Controlled by the ALTGX_RECONFIG Instance" on page 2–35.
- Connecting the `reconfig_fromgxb` and `reconfig_togxb` ports between the ALTGX and ALTGX_RECONFIG instances.

Logical Channel Addressing

When you configure an ALTGX instance in Basic (PMA Direct) functional mode, all the channels within the ALTGX instance are PMA-only channels. Depending on the pins you assign to these PMA-only channels, they can either be:

- Regular transceiver channels with PMA-only

  or

- CMU channels

However, when you configure an ALTGX instance with both the PCS and PMA blocks, all the channels within the ALTGX instance can only be regular transceiver channels.

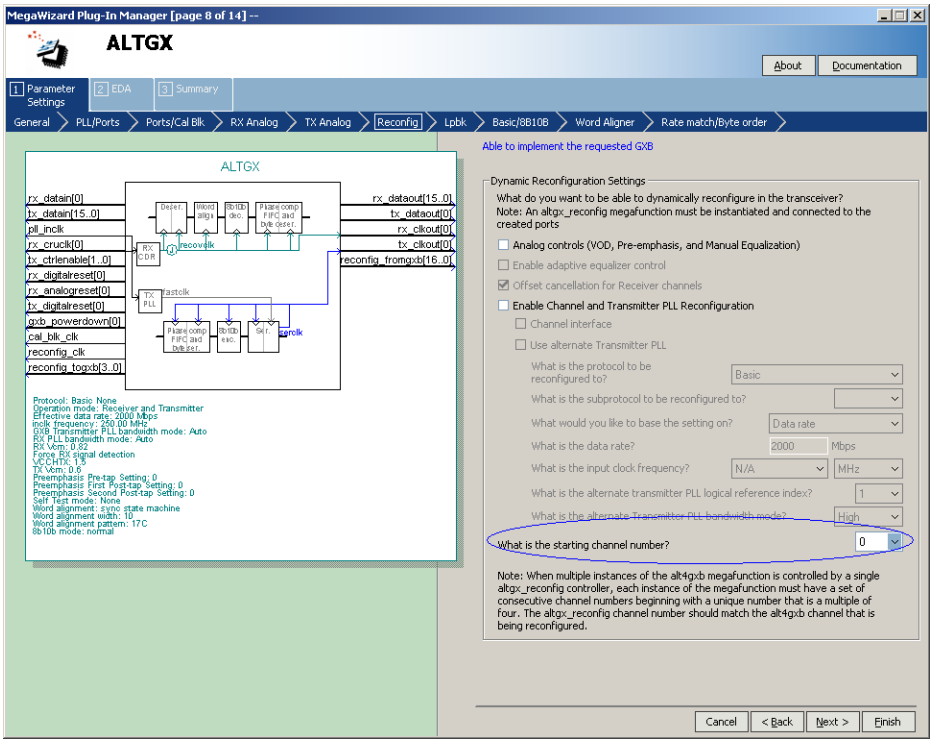
The following sections describe the concept of logical channel addressing for regular transceiver channels, PMA-only channels, and a combination of PMA-only channels and regular transceiver channels that are not in Basic (PMA direct) functional mode.

Logical Channel Addressing of Regular Transceiver Channels

This section describes how to set the **What is the starting channel number?** option using five different case scenarios for the regular transceiver channels.

Figure 2–7 shows the **What is the starting channel number?** option in the **Reconfig** screen of the ALTGX MegaWizard Plug-In Manager.

Figure 2–7. What is the Starting Channel Number? Option in the ALTGX MegaWizard Plug-In Manager



This value determines the logical channel address of all the transceiver channels in the ALTGX instance. You must always set the starting channel number in an ALTGX instance as a multiple of 4.

☞ You must set the next multiple of 4 for the **What is the starting channel number?** option of the following ALTGX instances, if any.

☞ Based on the value you entered for the **What is the starting channel number?** option, the ATLGX MegaWizard Plug-In Manager assigns consecutive numbers as the logical channel address for all the channels within an ALTGX instance.

The following sections describe the logical channel addressing of regular transceiver channels for five example scenarios in detail.

☞ The first three example scenarios: Case 1, 2, and 3 have a single ALTGX_RECONFIG instance connected to two ALTGX instances. The difference between these three cases is the number of channels configured in the ALTGX instances.

Regular Transceiver Channels: Case 1

Table 2–4 shows an example scenario for Case 1.

Table 2–4. Example Scenarios for Logical Channel Addressing in ALTGX Instances for Case 1

Example Scenario	Number of ALTGX Instances	Number of ALTGX_RECONFIG Instances
Case 1	Two ALTGX instances: ■ ALTGX instance 1 (1 channel) ■ ALTGX instance 2 (3 channels)	One ALTGX_RECONFIG instance controlling both the ALTGX instances

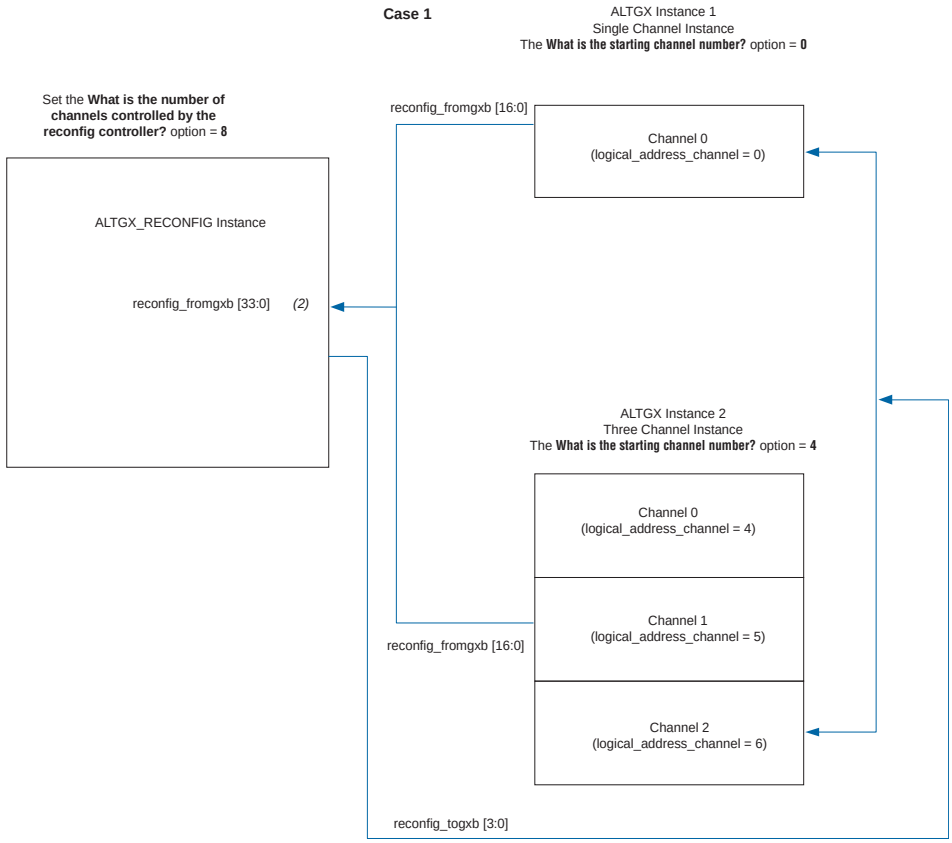
Table 2–5 shows the logical channel addressing of regular transceiver channels in Case 1.

Table 2–5. Logical Channel Addressing of Regular Transceiver Channels for Case 1

ALTGX Instances			ALTGX_RECONFIG Instance
ALTGX MegaWizard Plug-In Manager Setting	ALTGX instance 1	ALTGX instance 2	ALTGX_RECONFIG instance 1:
What is the number of channels? in the **General** screen	**1**	**3**	Controls both ALTGX instance 1 and ALTGX instance 2.
What is the starting channel number? in the **Reconfig** screen	Set this option to **0**	Because the starting channel number increments in steps of 4, you must set the starting channel number of the next ALTGX instance as a multiple of 4. Therefore, set this option to **4**.	Refer to Table 2–16 on page 2–37 for the ALTGX_RECONFIG instance 1 settings.
	The logical channel addresses of the first channel is **0**	The logical channel addresses of the first to third channels are **4**, **5**, and **6**, respectively.	

Figure 2–8 shows the logical channel addresses of all the channels within ALTGX instance 1 and ALTGX instance 2.

Figure 2–8. Case 1—Block Diagram of the ALTGX Instances and ALTGX_RECONFIG Instance *(Note 1)*



Notes to Figure 2–8:

(1) For more information, refer to "Regular Transceiver Channels: Case 1" on page 2–24.

(2) `reconfig_fromgxb[33:0]` = {`reconfig_fromgxb[16:0]`, `reconfig_fromgxb[16:0]`}.

Regular Transceiver Channels: Case 2

Table 2–6 shows an example scenario for Case 2.

Table 2–6. Example Scenarios for Logical Channel Addressing in ALTGX Instances for Case 2

Example Scenario	Number of ALTGX Instances	Number of ALTGX_RECONFIG Instances
Case 2	Two ALTGX instances: ■ ALTGX instance 1 (6 channels) ■ ALTGX instance 2 (3 channels)	One ALTGX_RECONFIG instance controlling both the ALTGX instances.

Table 2–7 shows the logical channel addressing of regular transceiver channels in Case 2.

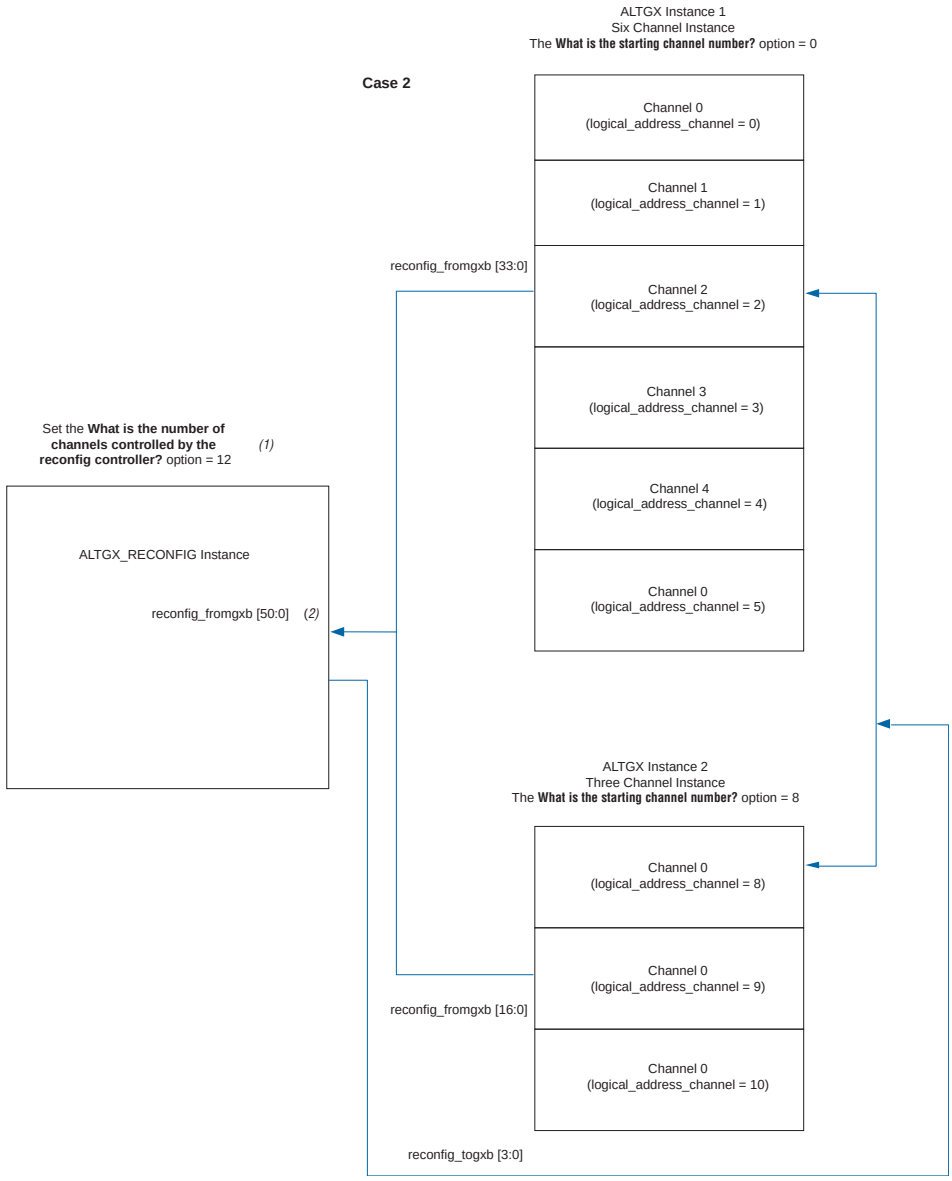
Table 2–7. Logical Channel Addressing of Regular Transceiver Channels for Case 2

ALTGX Instances			**ALTGX_RECONFIG Instance**
ALTGX MegaWizard Plug-In Manager Setting	ALTGX instance 1	ALTGX instance 2	ALTGX_RECONFIG instance 1:
What is the number of channels? in the **General** screen	**6**	**3**	Controls both ALTGX instance 1 and ALTGX instance 2.
What is the starting channel number? in the **Reconfig** screen	Set this option to **0**	Because the starting channel number increments in steps of 4, you must set the starting channel number of the next ALTGX instance as a multiple of 4. Therefore, set this option to **8**.	For more information, refer to Table 2–16 on page 2–37 for the ALTGX_RECONFIG instance 1 settings.
	The logical channel addresses of the first to sixth channels are **0**, **1**, **2**, **3**, **4**, and **5**, respectively.	The logical channel addresses of the first to third channels are **8**, **9**, and **10**, respectively.	

For Case 2, do not set the **What is the starting channel number?** option to **4** for ALTGX instance 2 (as in Case 1) because the logical channel addresses **4** and **5** are used already for the fifth and sixth channels of ALTGX instance 1. Instead, use the next multiple of 4 to assign it as the starting channel number of ALTGX instance 2.

Figure 2–9 shows the logical channel addresses of all the channels within ALTGX instance 1 and ALTGX instance 2.

Figure 2–9. Case 2—Block Diagram of the ALTGX Instances and ALTGX_RECONFIG Instance *(Note 1)*



Notes to Figure 2–9:

(1) For more information, refer to "Regular Transceiver Channels: Case 2" on page 2–25.

(2) `reconfig_fromgxb[50:0]` = `{reconfig_fromgxb[16:0], reconfig_fromgxb[33:0]}`.

Regular Transceiver Channels: Case 3

Table 2–8 shows an example scenario for Case 3.

Table 2–8. Example Scenarios for Logical Channel Addressing in ALTGX Instances for Case 3

Example Scenario	Number of ALTGX Instances	Number of ALTGX_RECONFIG Instances
Case 3	Two ALTGX instances: ■ ALTGX instance 1 (6 channels) ■ ALTGX instance 2 (6 channels)	One ALTGX_RECONFIG instance controlling both the ALTGX instances.

Table 2–9 shows the logical channel addressing of regular transceiver channels in Case 3.

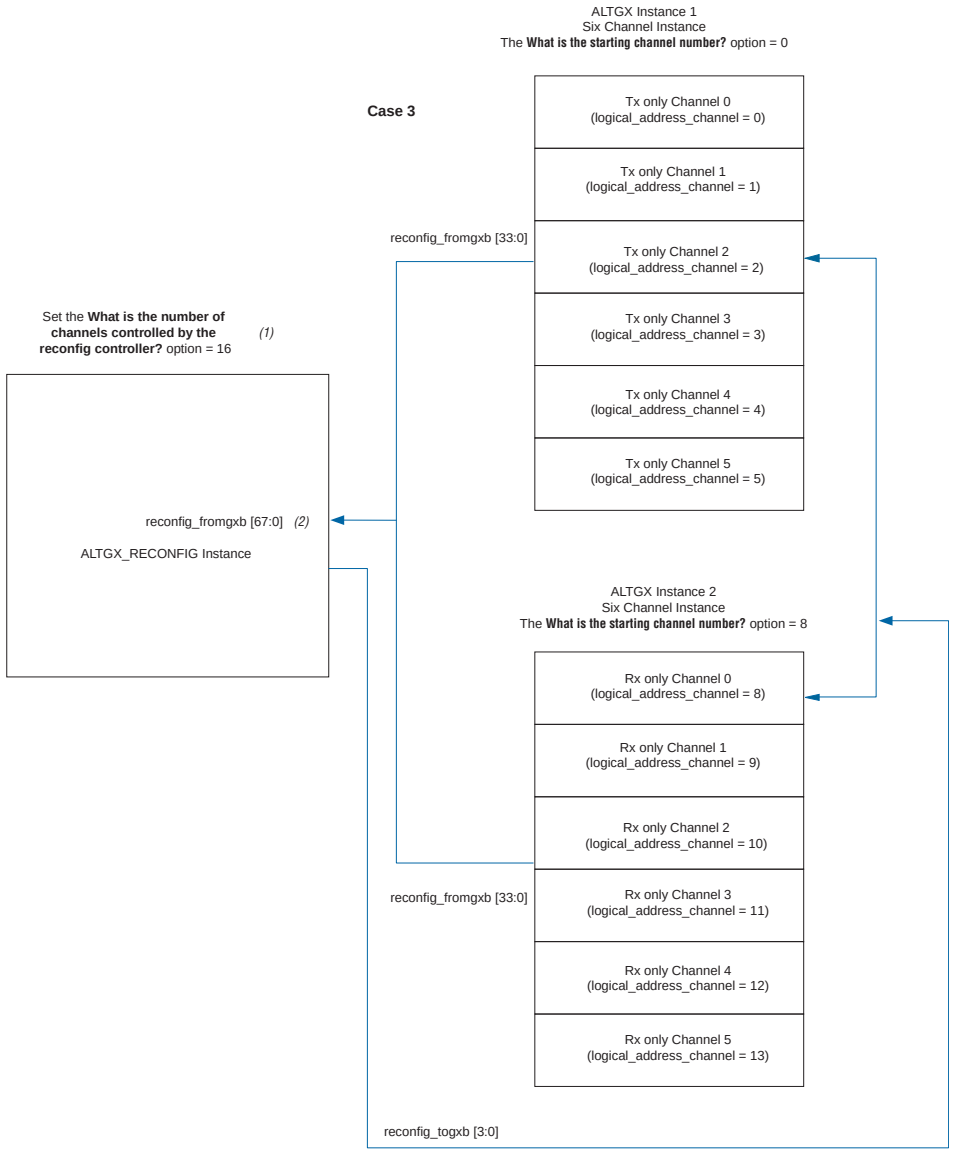
Table 2–9. Logical Channel Addressing of Regular Transceiver Channels for Case 3

ALTGX Instances			ALTGX_RECONFIG Instance
ALTGX MegaWizard Plug-In Manager Setting	ALTGX instance 1	ALTGX instance 2	ALTGX_RECONFIG instance 1:
What is the number of channels? in the **General** screen	**6**	**6**	Controls both ALTGX instance 1 and ALTGX instance 2.
What is the starting channel number? in the **Reconfig** screen	Set this option to **0**	Because the starting channel number increments in steps of 4, you must set the starting channel number of the next ALTGX instance as a multiple of 4. Therefore, set this option to **8**.	For more information, refer to Table 2–16 on page 2–37 for the ALTGX_RECONFIG instance 1 settings.
	The logical channel addresses of the first to sixth channels are **0**, **1**, **2**, **3**, **4**, and **5**, respectively.	The logical channel addresses of the first to third channels are **8**, **9**, **10**, **11**, **12**, and **13**, respectively.	

For Case 3, do not set the **What is the starting channel number?** option to **4** for ALTGX instance 2 (as in Case 1) because the logical channel addresses **4** and **5** are used already for the fifth and sixth channels of ALTGX instance 1. Instead, use the next multiple of 4 to assign it as the starting channel number of ALTGX instance 2.

Figure 2–10 shows the logical channel addresses of all the channels within ALTGX instance 1 and ALTGX instance 2.

Figure 2–10. Block Diagram of the ALTGX Instances and ALTGX_RECONFIG Instance for Case 3 *(Note 1)*



Notes to Figure 2–10:

(1) For more information, refer to "Regular Transceiver Channels: Case 3" on page 2–28.

(2) `reconfig_fromgxb[67:0]` = {`reconfig_fromgxb[33:0]`, `reconfig_fromgxb[33:0]`}.

Regular Transceiver Channels: Case 4

Similarly, in this example there are multiple ALTGX_RECONFIG instances, each controlling a single ALTGX instance in your design.

Case 4 explains the scenario under which you must set the **What is the starting channel number?** option differently. This scenario assumes that each ALTGX instance has its own ALTGX_RECONFIG instance.

Table 2–10 shows an example scenario for Case 4.

Table 2–10. Example Scenarios for Logical Channel Addressing in ALTGX Instances for Case 4

Example Scenario	Number of ALTGX Instances	Number of ALTGX_RECONFIG Instances
Case 4	Two ALTGX instances: ■ ALTGX instance 1 ■ ALTGX instance 2	Two ALTGX_RECONFIG instances: ALTGX_RECONFIG instance 1 and ALTGX_RECONFIG instance 2 ALTGX_RECONFIG instance 1 controls ALTGX instance 1 ALTGX_RECONFIG instance 2 controls ALTGX instance 2

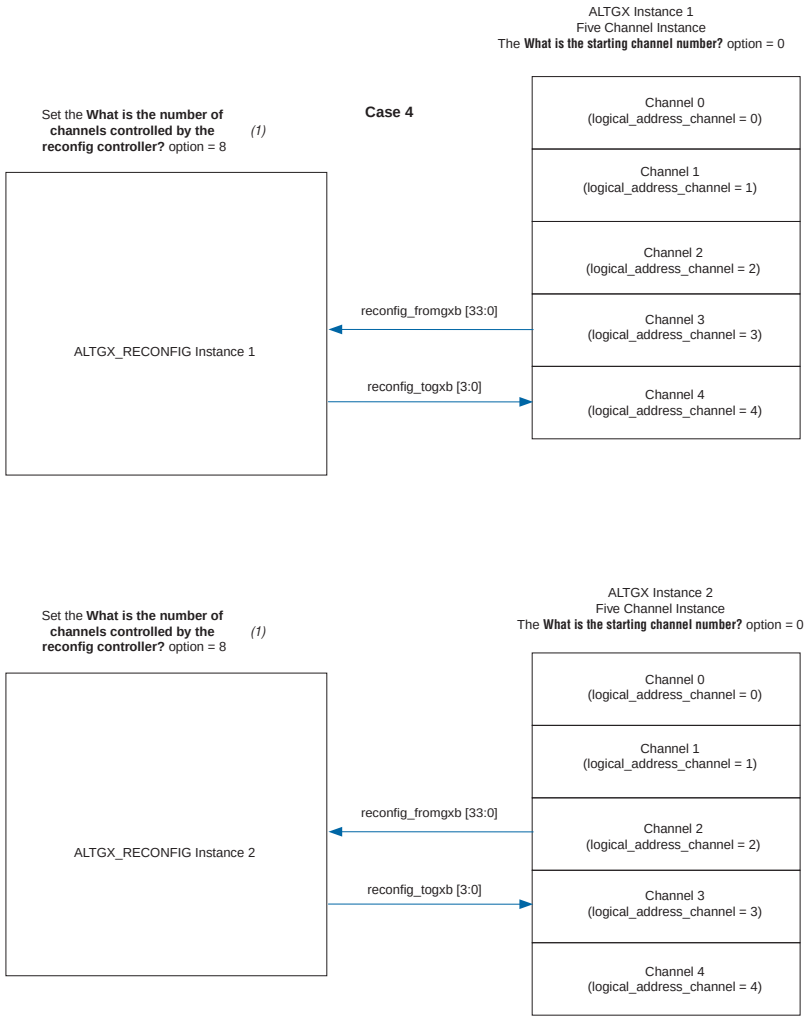
Table 2–11 shows the logical channel addressing of regular transceiver channels in Case 4.

Table 2–11. Logical Channel Addressing of Regular Transceiver Channels for Case 4

ALTGX Instances			ALTGX_RECONFIG Instance	ALTGX_RECONFIG Instance
ALTGX MegaWizard Plug-In Manager Setting	ALTGX instance 1	ALTGX instance 2	ALTGX_RECONFIG instance 1: Controls ALTGX instance 1. For more information, refer to Table 2–16 on page 2–37 for the ALTGX_RECONFIG instance 1 settings.	ALTGX_RECONFIG instance 2: Controls ALTGX instance 2. For more information, refer to Table 2–16 on page 2–37 for the ALTGX_RECONFIG instance 2 settings.
What is the number of channels? in the **General** screen	**5**	**5**		
What is the starting channel number? in the **Reconfig** screen	Set this option to **0**	Set this option to **0** (Because ALTGX instance 2 is controlled by a separate ALTGX_RECONFIG instance 2)		
	The logical channel addresses of the first to fifth channels are **0**, **1**, **2**, **3**, and **4**, respectively.	The logical channel addresses of the first to third channels are **0**, **1**, **2**, **3**, and **4**, respectively.		

Figure 2–11 shows the logical channel addresses of all the channels within ALTGX instance 1 and ALTGX instance 2.

Figure 2–11. Block Diagram of the ALTGX Instances and ALTGX_RECONFIG Instances for Case 4 *(Note 1)*



Note to Figure 2–11:

(1) For more information, refer to "Regular Transceiver Channels: Case 4" on page 2–30.

Regular Transceiver Channels: Case 5

In this example, you have only one ALTGX instance (ALTGX instance 1) and one ALTGX_RECONFIG instance in the design:

- ALTGX instance 1—The number of channels in this instance is 1
- **ALTGX .v** or **ALTGX.vhd** is stamped five times in the design

Table 2–12 shows an example scenario for Case 5.

Table 2–12. Example Scenarios for Logical Channel Addressing in ALTGX Instances for Case 5

Example Scenario	Number of ALTGX Instances	Number of ALTGX_RECONFIG Instances
Case 5	One ALTGX instance (ALTGX instance 1) stamped five times	One ALTGX_RECONFIG instance controlling all five stamped **ALTGX.v** or **ALTGX.vhd** instances

ALTGX Instance 1 Parameter Settings

Set the **What is the starting channel number?** option to **0**. This implies that the ATLGX MegaWizard Plug-In Manager sets the logical channel address of the single channel of ALTGX instance 1 = **0**.

When you stamp the above configured transceiver instance five times, the starting channel numbers of the other four instances (assume instance 2, instance 3, instance 4, instance 5) are **4**, **8**, **12**, and **16**, respectively.

Specify the starting channel number of the other stamped instances using the `defparam` parameter (for Verilog) as shown:

```
defparam instance2. starting_channel_number = 4;
```

`defparam instance3. starting_channel_number = 8;` and so on for the remaining stamped instances.

Logical Channel Addressing of PMA-Only Channels

This section describes how to set the **What is the starting channel number?** option for ALTGX instances with PMA-only channels. If your design requires only the PMA portion of the transceiver channels, configure the ALTGX MegaWizard Plug-In Manager in Basic (PMA direct) functional mode in the **General** screen. Therefore, all the channels in the ALTGX instance are configured as PMA-only channels.

The PMA-only channels are comprised of CMU channels, regular transceiver channels configured in Basic (PMA direct) functional mode, or both CMU channels and regular transceiver channels configured in Basic (PMA direct) functional mode.

CMU channels are always PMA-only channels. The regular transceiver channels can be optionally configured as PMA-only channels.

Set the starting channel number for the PMA-only channels in the same **What is the starting channel number?** option shown in Figure 2–7 on page 2–23.

The value you set in the **What is the starting channel number?** option determines the logical channel addresses of all the PMA-only channels within the ALTGX instance. You must always set the **What is the starting channel number?** option as a multiple of **4**.

The logical channel addresses of the PMA-only channels within the same ALTGX instance also increment in multiples of 4 (unlike the logical channel addressing of regular transceiver channels that are not configured in Basic (PMA direct) functional mode, where the logical channel address increments in steps of 1 within the same ALTGX instance).

Table 2–13 shows an example scenario in which the logical channel address of the PMA-only channels is set based on the value you entered in the **What is the starting channel number?** option.

Table 2–13 shows the logical channel addressing of regular transceiver channels in Case 6.

Table 2–13. Logical Channel Addressing of Regular Transceiver Channels for Case 6

ALTGX Instances		ALTGX_RECONFIG Instance
ALTGX MegaWizard Plug-In Manager Setting	ALTGX instance 1 configured in **Basic (PMA Direct)** functional mode	ALTGX_RECONFIG instance 1: Controls ALTGX instance 1. For more information, refer to Table 2–17 on page 2–39 for the ALTGX_RECONFIG instance 1 settings.
What is the number of channels? in the **General** screen	**8 (PMA-only channels)**	
What is the starting channel number? in the **Reconfig** screen	Set this option to **0**	
	The logical channel addresses of the first to sixth channels are **0**, **4**, **8**, **12**, **16**, **20**, **24**, and **28**, respectively.	

You must set the next multiple of 4 as the starting channel number for the following ALTGX instances, if any.

Logical Channel Addressing: Combination of PMA-Only Channels and Regular Transceiver Channels that are Not in Basic (PMA Direct) Functional Mode

This section describes how to set the **What is the starting channel number?** option for multiple ALTGX instances controlled by the same ALTGX_RECONFIG instance, where one ALTGX instance can have PMA-only channels and the other ALTGX instance can have regular transceiver channels that are not in Basic (PMA direct) functional mode.

Table 2–14 shows an example scenario in which the logical channel address of both the PMA-only channels and regular transceiver channels is set based on the value you entered in the **What is the starting channel number?** option.

Table 2–14. Logical Channel Addressing of a Combination of PMA-Only Channels and Regular Transceiver Channels that are Not in Basic (PMA Direct) Functional Mode for Case 7

ALTGX Instances			ALTGX_RECONFIG Instance
ALTGX MegaWizard Plug-In Manager Setting	ALTGX instance 1 configured in Basic functional mode	ALTGX instance 2	ALTGX_RECONFIG instance 1:
What is the number of channels? in the **General** screen	**6**	**6**	Controls both ALTGX instance 1 and ALTGX instance 2.
What is the starting channel number? in the **Reconfig** screen	Set this option to **0**	Because the starting channel number increments in steps of 4, you must set the starting channel number of the next ALTGX instance as a multiple of 4. Therefore, set this option as **8** for ALTGX instance 2. This is because the starting channel numbers 0 and 4 have already been used in ALTGX instance 1.	For more information, refer to Table 2–18 on page 2–39 for the ALTGX_RECONFIG instance 1 settings.
	The logical channel addresses of the first to fifth channels are **0**, **1**, **2**, **3**, and **4**, respectively.	The logical channel addresses of the first to fourth channels are **8**, **12**, **16**, and **20**, respectively.	

Highest Possible Logical Channel Address

Table 2–15 shows the highest possible logical channel address assigned to a transceiver channel in a HardCopy IV GX device.

The maximum number of transceiver channels in the largest HardCopy IV GX device is 24 (12 transceiver channels located in two transceiver blocks on the right side of the device and 12 transceiver channels located in two transceiver blocks on the left side of the device).

These 24 transceiver channels can be individually configured as 24 **Transmitter only** and 24 **Receiver only** channels. You can achieve this by using 24 **Transmitter only** ALTGX instances and 24 **Receiver only** ALTGX instances in your design.

Table 2–15. Highest Possible Logical Channel Address

48 ALTGX Instances			ALTGX_RECONFIG Instance
ALTGX MegaWizard Plug-In Manager Setting	ALTGX instance 1	ALTGX instance 2	ALTGX_RECONFIG instance 1:
What is the number of channels? in the **General** screen	**24**	**24**	Controls all 48 ALTGX instances.
What is the starting channel number? in the **Reconfig** screen	TX instance 1: **0** TX instance 2: **4** . . . . . . TX instance 24: **96**	RX instance 1: **100** RX instance 2: **104** . . . . . . RX instance 24: **184**	

The highest logical channel address is assigned to the **Receiver only** channel in the 48th ALTGX instance; therefore, the setting is **184**.

The highest possible logical channel address assigned to a transceiver channel in a HardCopy IV GX device is the same whether the channel is a regular transceiver channel or a PMA-only channel.

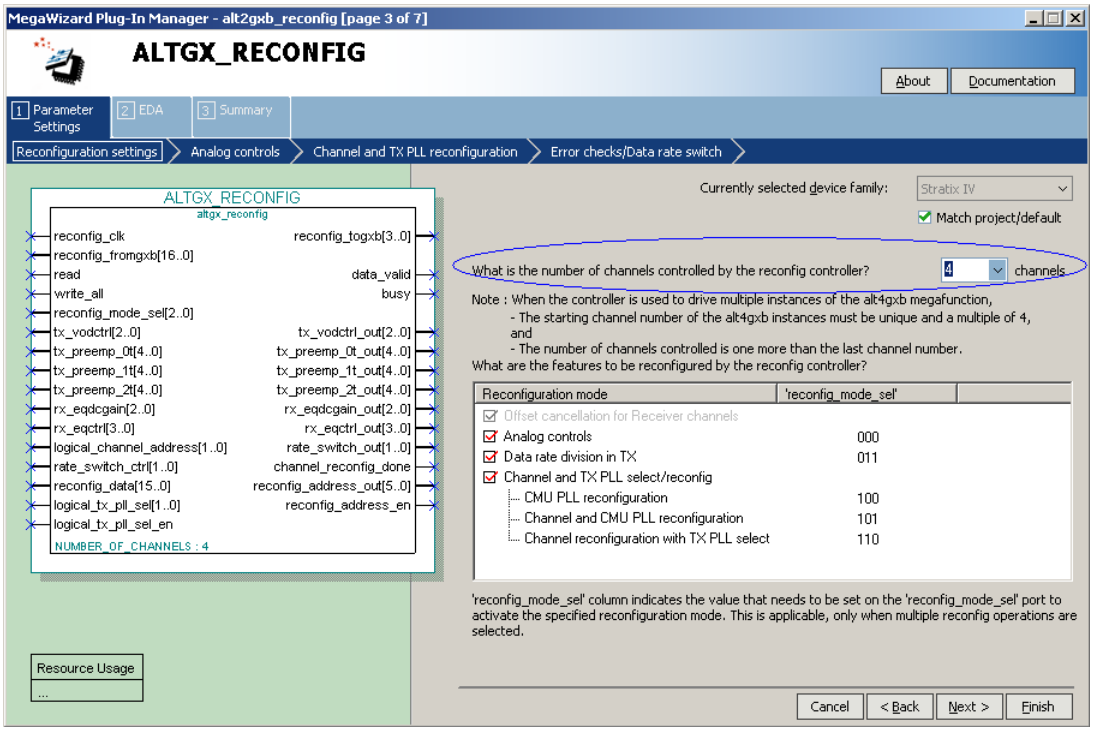
The Quartus II software automatically packs the logical channels into the physical placements. The physical placement includes combining channels into the same transceiver block. For more information, refer to "Combining Transceiver Channels with Dynamic Reconfiguration Enabled" on page 2–140.

Total Number of Channels Controlled by the ALTGX_RECONFIG Instance

The dynamic reconfiguration controller requires information about the total number of channels connected to it. Based on this information, the `reconfig_fromgxb` and `logical_channel_address` input ports vary in width. Therefore, provide this information in the **What is the number of channels controlled by the reconfig controller?** option in the **Reconfiguration settings** screen of the ALTGX_RECONFIG MegaWizard Plug-In Manager, as shown in Figure 2–12.

This section describes how to set the total number of channels controlled by the dynamic reconfiguration controller (ALTGX_RECONFIG instance). The maximum number of channels that you can set in this option is 256.

Figure 2–12. What is the Number of Channels Controlled by the Reconfig Controller? Option in the ALTGX_RECONFIG MegaWizard Plug-In Manager



In this example, one ALTGX_RECONFIG instance is controlling all the ALTGX instances in a design. Use the following rules for setting the **What is the number of channels controlled by the controller?** option:

- Determine the highest logical channel address among all the transceiver instances connected to the same dynamic reconfiguration controller. For information about determining the logical channel address using the starting channel number, refer to "Logical Channel Addressing" on page 2–23.
- Round the logical channel address value to the next higher multiple of 4.
- Use this value to set the **What is the number of channels controlled by the reconfig controller**? option.

Total Number of Channels Controlled by the ALTGX_RECONFIG Instance—Regular Transceiver Channels

Consider the example scenarios described in:

- Table 2–3 on page 2–20
- Table 2–5 on page 2–24
- Table 2–7 on page 2–26
- Table 2–9 on page 2–28
- Table 2–11 on page 2–30

Using the information from the tables listed, set the **What is the number of channels controlled by the reconfig controller** option for the same example scenarios.

Table 2–16 shows the ALTGX and ALTGX_RECONFIG settings for Instances 1 and 2 in Cases 1 through 5.

Table 2–16. ALTGX and ALTGX_RECONFIG Settings for Instances 1 and 2 in Cases 1 through 5 (Part 1 of 2)

Example Scenario	ALTGX Instance 1 Settings	ALTGX Instance 2 Settings	ALTGX_RECONFIG Instance 1 Settings	ALTGX_RECONFIG Instance 2 Settings
Case 1 For a block diagram of the ALTGX instances and the ALTGX_RECONFIG instance in this design, refer to Figure 2–8 on page 2–25.	■ One regular transceiver channel. ■ Set the **What is the starting channel number?** option in the **Reconfig** screen to **0**. ■ The logical channel address of the first channel is **0**.	■ Three regular transceiver channels. ■ Set the **What is the starting channel number?** option in the **Reconfig** screen to **4**. ■ The logical channel address of the first to third channels are **4**, **5**, and **6**, respectively.	■ Determine the highest logical channel address (6) for ALTGX instance 1. ■ Round it up to the next multiple of 4. ■ Set the **What is the number of channels controlled by the controller?** option in the **Reconfiguration settings** screen to **8**.	—
Case 2 For a block diagram of the ALTGX instances and the ALTGX_RECONFIG instance in this design, refer to Figure 2–9 on page 2–27.	■ Six regular transceiver channels. ■ Set the **What is the starting channel number?** option in the **Reconfig** screen to **0**. ■ The logical channel address of the first to sixth channels are **0**, **1**, **2**, **3**, **4**, and **5**, respectively.	■ Three regular transceiver channels. ■ Set the **What is the starting channel number?** option in the **Reconfig** screen to **8**. ■ The logical channel address of the first to third channels are **8**, **9**, and **10**, respectively.	■ Determine the highest logical channel address (10). ■ Round it up to the next multiple of 4. ■ Set the **What is the number of channels controlled by the controller?** option in the **Reconfiguration settings** screen to **12**.	—
Case 3 For a block diagram of the ALTGX instances and the ALTGX_RECONFIG instance in this design, refer to Figure 2–10 on page 2–29.	■ Six regular transceiver channels. ■ Set the **What is the starting channel number?** option in the **Reconfig** screen to **0**. ■ The logical channel address of the first to sixth channels are **0**, **1**, **2**, **3**, **4**, and **5**, respectively.	■ Six regular transceiver channels. ■ Set the **What is the starting channel number?** option in the **Reconfig** screen to **8**. ■ The logical channel address of the first to sixth channels are **8**, **9**, **10**, **11**, **12**, and **13**, respectively.	■ Determine the highest logical channel address (13). ■ Round it up to the next multiple of 4. ■ Set the **What is the number of channels controlled by the controller?** option in the **Reconfiguration settings** screen to **16**.	—

Table 2–16. ALTGX and ALTGX_RECONFIG Settings for Instances 1 and 2 in Cases 1 through 5 (Part 2 of 2)

Example Scenario	ALTGX Instance 1 Settings	ALTGX Instance 2 Settings	ALTGX_RECONFIG Instance 1 Settings	ALTGX_RECONFIG Instance 2 Settings
Case 4 For a block diagram of the ALTGX instances and the ALTGX_RECONFIG instance in this design, refer to Figure 2–11 on page 2–31.	■ Five regular transceiver channels controlled by ALTGX_RECONFIG instance 1. ■ Set the **What is the starting channel number?** option in the **Reconfig** screen to **0**. ■ The logical channel address of the first to fifth channels are **0**, **1**, **2**, **3**, and **4**, respectively.	■ Five regular transceiver channels controlled by ALTGX_RECONFIG instance 2. ■ Set the **What is the starting channel number?** option in the **Reconfig** screen to **0**. ■ The logical channel address of the first to fifth channels are **0, 1**, **2**, **3**, and **4**, respectively.	■ Determine the highest logical channel address (4) for ALTGX instance 1. ■ Round it up to the next multiple of 4. ■ Set the **What is the number of channels controlled by the controller?** option in the **Reconfiguration settings** screen to **8**.	■ Determine the highest logical channel address (4) for ALTGX instance 2. ■ Round it up to the next multiple of 4. ■ Set the **What is the number of channels controlled by the controller?** option in the **Reconfiguration settings** screen to **8**.
Case 5	■ ALTGX instance 1 stamped five times. ■ Specify the starting channel number of the other stamped instances using the `defparam` parameter (for Verilog) as shown: `defparam instance2.starting_channel_number = 4; defparam instance3.starting_channel_number = 8;` and so on for the remaining stamped instances. ■ The logical channel addresses of the channels in the stamped instances are **0**, **4**, **8**, **12**, and **16**, respectively.	—	■ Determine the highest logical channel address (16). ■ Round it up to the next multiple of 4. ■ Set the **What is the number of channels controlled by the controller?** option in the **Reconfiguration settings** screen to **20**.	—

Total Number of Channels Controlled by the ALTGX_RECONFIG Instance—PMA-Only Channels

Consider the example scenario shown in Table 2–13 on page 2–33 to see how to set the **What is the number of channels controlled by the reconfig controller?** option in the ALTGX_RECONFIG MegaWizard Plug-In Manager for the same example scenario.

Table 2–17 shows the ALTGX and ALTGX_RECONFIG settings in Case 6.

Table 2–17. ALTGX and ALTGX_RECONFIG Settings for Instances 1 and 2 in Case 6

Example Scenario	ALTGX Instance 1 Settings	ALTGX Instance 2 Settings	ALTGX_RECONFIG Instance 1 Settings
Case 6	■ Eight PMA-only channels. ■ Set the **What is the starting channel number?** option in the **Reconfig** screen to **0**. ■ The logical channel address of the first to eighth channels are **0**, **4**, **8**, **12**, **16**, **20**, **24**, and **28**, respectively.	—	■ Determine the highest logical channel address (28). ■ Round it up to the next multiple of 4. ■ Set the **What is the number of channels controlled by the controller?** option in the **Reconfiguration settings** screen to **32**.

Total Number of Channels Controlled by the ALTGX_RECONFIG Instance—PMA-Only Channels and Regular Transceiver Channels that are Not in Basic (PMA Direct) Functional Mode

Consider the example scenario shown in Table 2–14 on page 2–34 to set the **What is the number of channels controlled by the reconfig controller?** option in the ALTGX_RECONFIG MegaWizard Plug-In Manager.

Table 2–18 shows the ALTGX and ALTGX_RECONFIG settings for Instances 1 and 2 for Case 7.

Table 2–18. ALTGX and ALTGX_RECONFIG Settings for Instances 1 and 2 in Case 7

Example Scenario	ALTGX Instance 1 Settings	ALTGX Instance 2 Settings	ALTGX_RECONFIG Instance 1 Settings
Case 7	■ Five regular transceiver channels that are not in **Basic (PMA direct)** functional mode. ■ Set the **What is the starting channel number?** option in the **Reconfig** screen to **0**. ■ The logical channel address of the first to fifth channels are **0**, **1**, **2**, **3**, and **4**, respectively.	■ Four PMA-only channels. ■ Set the **What is the starting channel number?** option in the **Reconfig** screen to **8**. ■ The logical channel address of the first to fourth channels are **8**, **12**, **16**, and **20**, respectively.	■ Determine the highest logical channel address (20). ■ Round it up to the next multiple of 4. ■ Set the **What is the number of channels controlled by the controller?** option in the **Reconfiguration settings** screen to **24**.

Connecting the reconfig_fromgxb and reconfig_togxb Ports

The `reconfig_fromgxb` and `reconfig_togxb` signals in the dynamic reconfiguration interface must be connected between the ALTGX_RECONFIG instance and the ALTGX instance to successfully complete the dynamic reconfiguration process:

- `reconfig_togxb[3:0]`—This is an input port of the ALTGX instance and an output port of the ALTGX_RECONFIG instance. You must connect the `reconfig_togxb[3:0]` input port of every ALTGX instance controlled by the dynamic reconfiguration controller to the `reconfig_togxb[3:0]` output port of the ALTGX_RECONFIG instance. For more information, refer to Figure 2–13 on page 2–41.
- `reconfig_fromgxb`—This is an output port in the ALTGX instance and an input port in the ALTGX_RECONFIG instance. This signal is transceiver-block based. Therefore, the width of this signal increases in steps of 17 bits per transceiver block.

In the ALTGX MegaWizard Plug-In Manager, the width of this signal depends on the following:

- Whether the channels configured in the ALTGX instance are regular transceiver channels or PMA-only channels.
- The number of channels you select in the **What is the number of channels?** option in the **General** screen.

For example, if the channels in the ALTGX instance are regular transceiver channels and if you select the number of channels as follows:

- 1 ≤ Channels ≤ 4, then the output port `reconfig_fromgxb` = 17 bits
- 5 ≤ Channels ≤ 8, then the output port `reconfig_fromgxb` = 34 bits
- 9 ≤ Channels ≤ 12, then the output port `reconfig_fromgxb` = 51 bits

However, if the channels in the ALTGX instance are PMA-only channels and if you select the number of channels as follows:

- Number of PMA-only channels = n, then the output port `reconfig_fromgxb` = n*17 bits (for example, `reconfig_fromgxb` = 6 * 17 bits for 6 PMA-only channels)

In the ALTGX_RECONFIG MegaWizard Plug-In Manager, the width of this signal depends on the value you select in the **What is the number of channels controlled by the reconfig controller?** option in the **Reconfiguration settings** screen.

For example, if you select the total number of channels controlled by ALTGX_RECONFIG instance as follows:

- 1 ≤ Channels ≤ 4, then the input port `reconfig_fromgxb` = 17 bits
- 5 ≤ Channels ≤ 8, then the input port `reconfig_fromgxb` = 34 bits
- 9 ≤ Channels ≤ 12, then the input port `reconfig_fromgxb` = 51 bits

To connect the `reconfig_fromgxb` port between the ALTGX_RECONFIG instance and multiple ALTGX instances, follow these rules:

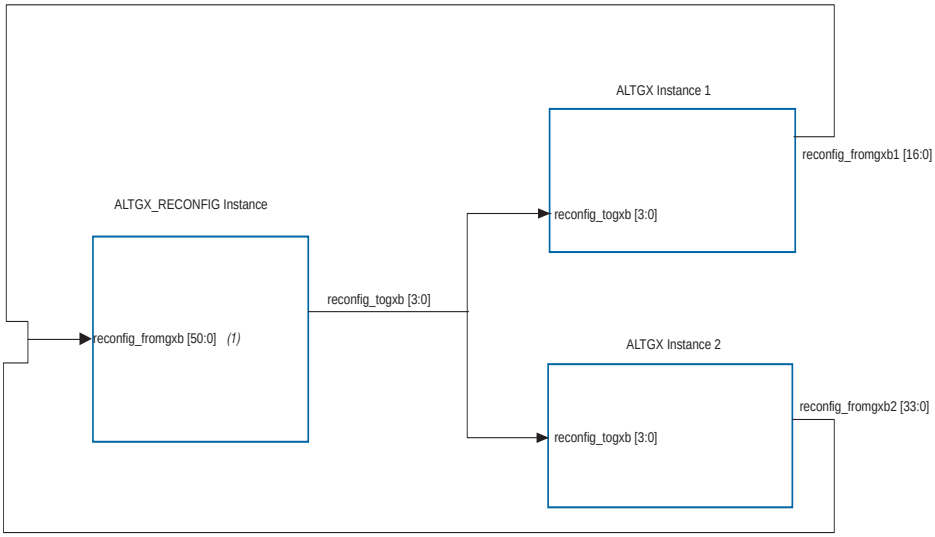
- Take the `reconfig_fromgxb[16:0]` of ALTGX instance 1 and connect it to the `reconfig_fromgxb[16:0]` of the ALTGX_RECONFIG instance. Connect the `reconfig_fromgxb[]` port of the next ALTGX instance to the next available bits of the ALTGX_RECONFIG instance, and so on.
- Similarly, connect the `reconfig_fromgxb` port of the ALTGX instance, which has the highest **What is the starting channel number?** option, to the MSB of the `reconfig_fromgxb` port of the ALTGX_RECONFIG instance.

The Quartus II Fitter produces an error if the dynamic reconfiguration option is enabled in the ALTGX instance but the `reconfig_fromgxb` and `reconfig_togxb` ports are not connected to the ALTGX_RECONFIG instance.

Connecting reconfig_fromgxb for Regular Transceiver Channels

Figure 2–13 shows how to connect the `reconfig_fromgxb` output port of the ALTGX instance to the `reconfig_fromgxb` input port of the ALTGX_RECONFIG instance for regular transceiver channels.

Figure 2–13. reconfig_fromgxb and reconfig_togxb Connections Between the ALTGX_RECONFIG Instance and the ALTGX Instances *(Note 1)*, *(2)*



Notes to Figure 2–13:

(1) `reconfig_fromgxb[50:0]` = {`reconfig_fromgxb[16:0]`, `reconfig_fromgxb[33:0]`}.

(2) Figure 2–13 assumes that all the channels depicted are regular transceiver channels.

Table 2–19 shows the connecting reconfig_fromgxb port for regular transceiver channels.

Table 2–19. Connecting the reconfig_fromgxb Port for Regular Transceiver Channels

ALTGX Settings and Instances			ALTGX_RECONFIG Setting and Instance	
ALTGX Setting	**ALTGX instance 1 (Basic Functional Mode)**	**ALTGX instance 2 (Basic Functional Mode)**	**ALTGX_RECONFIG Setting**	**ALTGX_RECONFIG Instance 1**
What is the number of channels? option in the **General** screen	**3** (Regular Transceiver Channels)	**5** (Regular Transceiver Channels)	**What is the number of channels controlled by the reconfig controller?** option in the **Reconfiguration settings** screen.	■ Determine the highest logical channel address (8). ■ Round it up to the next multiple of 4. ■ Set this option to **12**.
What is the starting channel number? option in the **Reconfig** screen	■ Set this option to **0**. ■ The logical channel addresses of the 1st to 3rd channels are 0, 1, and 2, respectively.	■ Set this option to 4. ■ The logical channel addresses of the 1st to 5th channels are 4, 5, 6, 7, and 8, respectively.		
reconfig_fromgxb1 and reconfig_fromgxb2 outputs	reconfig_fromgxb1 is 17 bits wide (**1** * 17)	reconfig_fromgxb2 is 34 bits wide (**2** * 17)	reconfig_fromgxb input	reconfig_fromgxb is 51 bits wide (**12** regular transceiver channels can logically fit into 3 transceiver blocks)

You must connect the reconfig_fromgxb input port of the ALTGX_RECONFIG instance to both the reconfig_fromgxb1 output port of ALTGX instance 1 and the reconfig_fromgxb2 output port of ALTGX instance 2, as shown in Figure 2–13 on page 2–41. The lowest **What is the starting channel number?** option transceiver block is connected to the lowest significant bit and so on. The reconfig_fromgxb1 of ALTGX instance 1 must be connected to the reconfig_fromgxb[16:0] of the ALTGX_RECONFIG instance. Similarly, the reconfig_fromgxb2 of ALTGX instance 2 must be connected to the reconfig_fromgxb[50:17] of the ALTGX_RECONFIG instance.

Connecting reconfig_fromgxb for the PMA-Only Channels

Consider the scenario for PMA-only channels.

Table 2–20. Connecting reconfig_fromgxb and reconfig_togxb for PMA-Only Channels (Part 1 of 2)

ALTGX Setting and Instances			ALTGX_RECONFIG Settings and Instance	
ALTGX MegaWizard Plug-In Manager Settings	ALTGX instance 1 configured in **Basic (PMA Direct)** functional mode	ALTGX instance 2 configured in **Basic (PMA Direct)** functional mode	ALTGX_RECONFIG MegaWizard Plug-In Manager Settings	ALTGX_RECONFIG instance 1

Table 2–20. Connecting reconfig_fromgxb and reconfig_togxb for PMA-Only Channels (Part 2 of 2)

ALTGX Setting and Instances			ALTGX_RECONFIG Settings and Instance	
What is the number of channels? option in the **General** screen	**1** (PMA-only channels)	**5** (PMA-only channels)	**What is the number of channels controlled by the reconfig controller?** option in the **Reconfiguration settings** screen	■ Determine the highest logical channel address (20). ■ Round it up to the next multiple of 4. ■ Set this option to **24**
What is the starting channel number? option in the **Reconfig** screen	Set this option to **0**. The logical channel address of the first channel is **0**.	Set this option to **4**. The logical channel addresses of the first to fifth channels are **4**, **8**, **12**, **16**, and **20**, respectively.		
`reconfig_fromgxb1` and `reconfig_fromgxb2` outputs	`reconfig_fromgxb1` is 17 bits wide (**1** * 17)	`reconfig_fromgxb2` is 85 bits wide (**5** * 17)	`reconfig_fromgxb` input	`reconfig_fromgxb` is 102 bits wide (**6** * 17, **24** channels can logically fit into 6 transceiver blocks)

You must connect the `reconfig_fromgxb` input port of the ALTGX_RECONFIG instance to both the `reconfig_fromgxb1` output port of ALTGX instance 1 and the `reconfig_fromgxb2` output port of ALTGX instance 2, as follows:

```
reconfig_fromgxb[101:0] = {reconfig_fromgxb2[84:0],
reconfig_fromgxb1[16:0]}
```

The lowest **What is the starting channel number?** option transceiver block is connected to the lowest significant bit, and so on.

The `reconfig_fromgxb1` of ALTGX instance 1 must be connected to `reconfig_fromgxb[16:0]` of the ALTGX_RECONFIG instance.

Similarly, `reconfig_fromgxb2` of ALTGX instance 2 must be connected to `reconfig_fromgxb[101:17]` of the ALTGX_RECONFIG instance.

The `reconfig_togxb` output of ALTGX_RECONFIG instance 1 is fixed to 3 bits and must be connected to the same `reconfig_togxb` ports of both the ALTGX instances.

Offset Cancellation Control for Receiver Channels

As the silicon progresses toward smaller process nodes, the performance of circuits at these smaller nodes depends more on process variations. These process variations result in analog voltages being offset from required ranges. The HardCopy IV GX device provides an offset cancellation circuit per receiver channel to counter offset variations due to process, voltage, and temperature. The offset cancellation logic corrects these offsets. The receiver buffer and receiver clock data recovery require offset cancellation.

Offset cancellation is automatically executed once every time the device is powered on. The control logic for offset cancellation is integrated into the dynamic reconfiguration controller. You must connect the ALTGX_RECONFIG instance to the ALTGX instances with receiver channels in your design. You must connect the `reconfig_fromgxb`, `reconfig_togxb`, and necessary clock signals to both the ALTGX_RECONFIG and ALTGX (with receiver channels) instances.

☞ For proper device operation, you must always connect the ALTGX_RECONFIG and ALTGX (with receiver channels) instances.

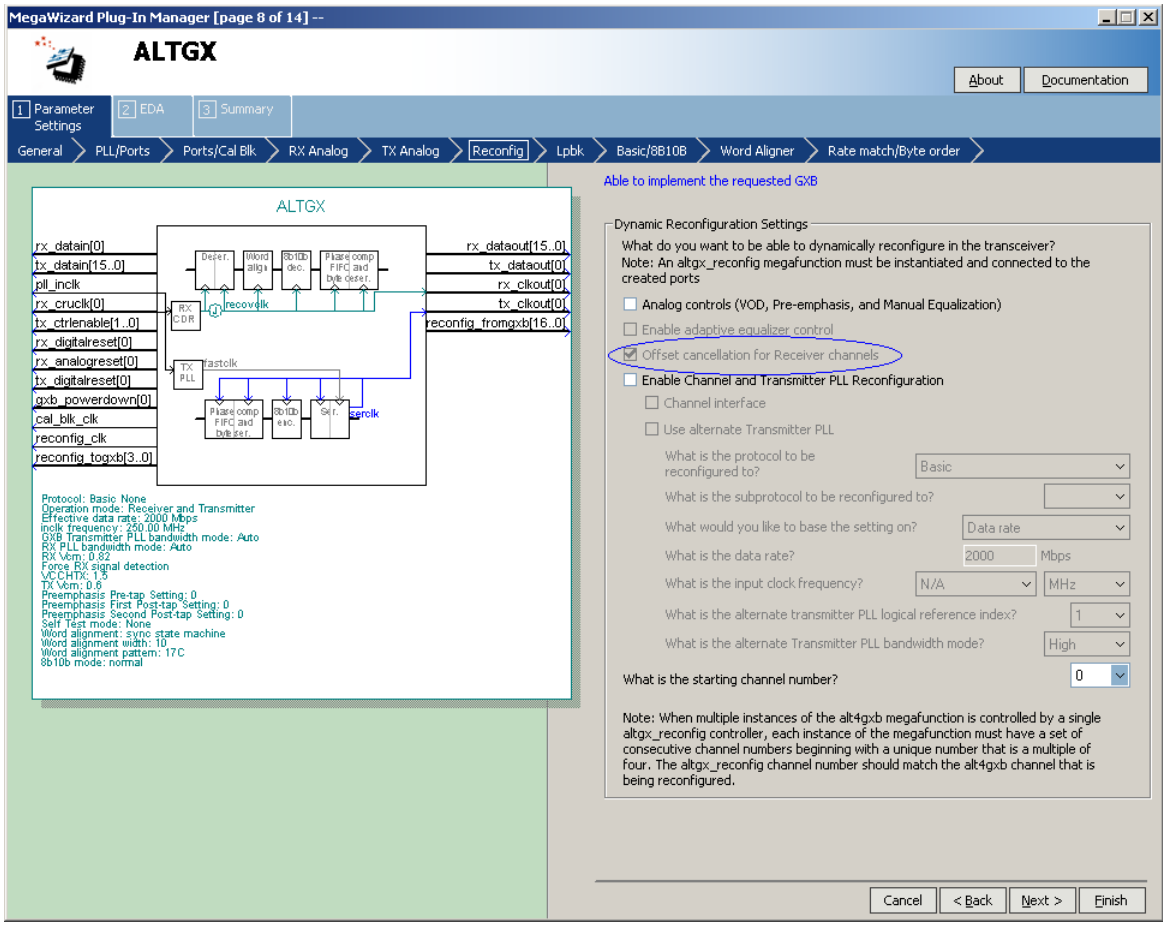
☞ The offset cancellation control functionality remains the same for both regular transceiver channels and PMA-only channels.

Operation

Every ALTGX instance for **Receiver and Transmitter** or **Receiver only** configurations require that the **Offset cancellation for Receiver channels** option is enabled in the **Reconfig** screen of the ALTGX MegaWizard Plug-In Manager. This option is enabled by default for the above two configurations, as shown in Figure 2–14. It is disabled for the **Transmitter only** configuration.

Figure 2–14 shows the **Offset cancellation for Receiver channels** option enabled by default in the ALTGX instance.

Figure 2–14. Offset Cancellation for Receiver Channels Option in the ALTGX MegaWizard Plug-In Manager

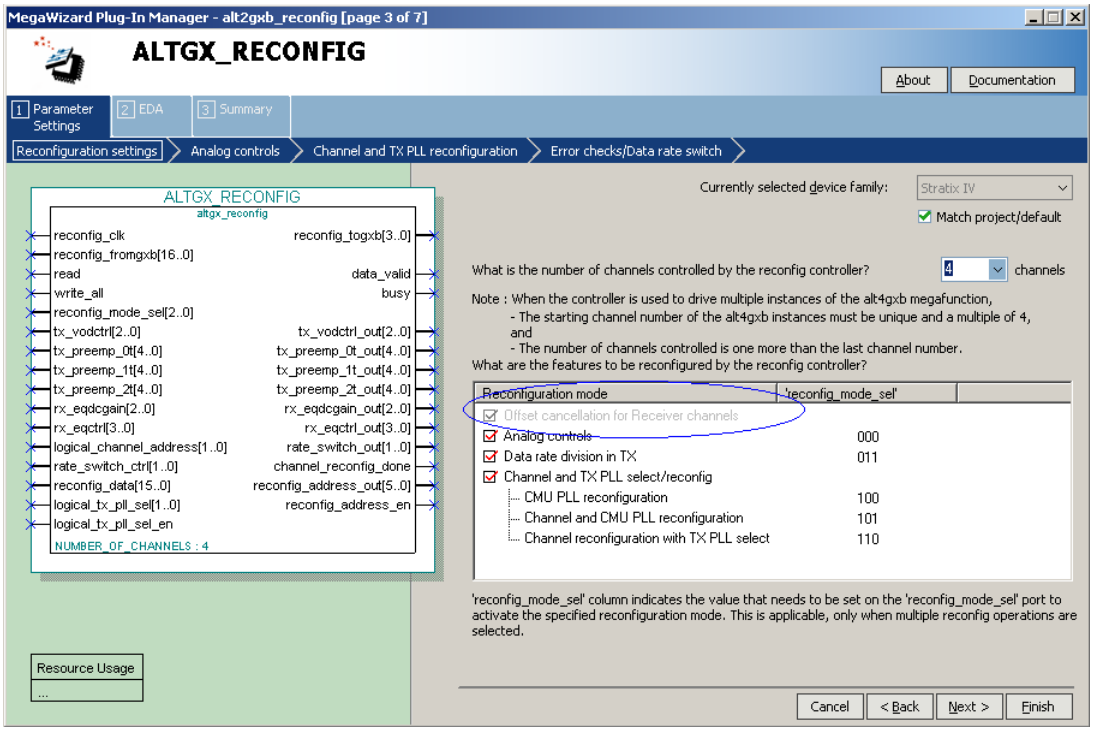


Because this option is enabled by default, the ALTGX instance must be connected to an ALTGX_RECONFIG instance (dynamic reconfiguration controller). The offset cancellation controls are also enabled by default in the **Reconfiguration settings** screen of the ALTGX_RECONFIG instance.

You must also set the starting channel number in the **What is the starting channel number?** option for every ALTGX instance connected to the ALTGX_RECONFIG instance. For more information, refer to "Logical Channel Addressing" on page 2–23.

Figure 2–15 shows the **Offset cancellation for Receiver channels** option enabled by default in the ALTGX_RECONFIG MegaWizard Plug-In Manager.

Figure 2–15. Offset Cancellation for Receiver Channels Option in the ALTGX_RECONFIG MegaWizard Plug-In Manager



When the device powers up, the dynamic reconfiguration controller initiates offset cancellation on the receiver channel by disconnecting the receiver input pins from the receiver data path. It also sets the receiver CDR into a fixed set of dividers to guarantee a voltage controlled oscillator (VCO) clock rate within the range necessary to provide proper offset cancellation. Subsequently, the offset cancellation process goes through different states and culminates in the offset cancellation of the receiver buffer and receiver CDR. After offset cancellation is complete, the user divider settings are restored.

The dynamic reconfiguration controller sends and receives data to the transceiver channel through the `reconfig_togxb` and `reconfig_fromgxb` signals. You must connect these signals between the ALTGX_RECONFIG instance and the ALTGX instance. You must also set the **What is the number of channels controlled by the reconfig controller?** option in the **Reconfiguration settings** screen of the ALTGX_RECONFIG MegaWizard Plug-In Manager. For more information, refer to "Total Number of Channels Controlled by the ALTGX_RECONFIG Instance" on page 2–35.

The **Use 'logical_channel_address' port for Analog controls reconfiguration** option in the **Analog controls** screen of the ALTGX_RECONFIG MegaWizard Plug-In Manager is not applicable for the receiver offset cancellation process.

If the design does not require PMA controls reconfiguration and uses optimum LE resources, you can connect all the ALTGX instances in the design to a single dynamic reconfiguration controller (ALTGX_RECONFIG instance).

The `gxb_powerdown` signal must not be asserted during the offset cancellation sequence.

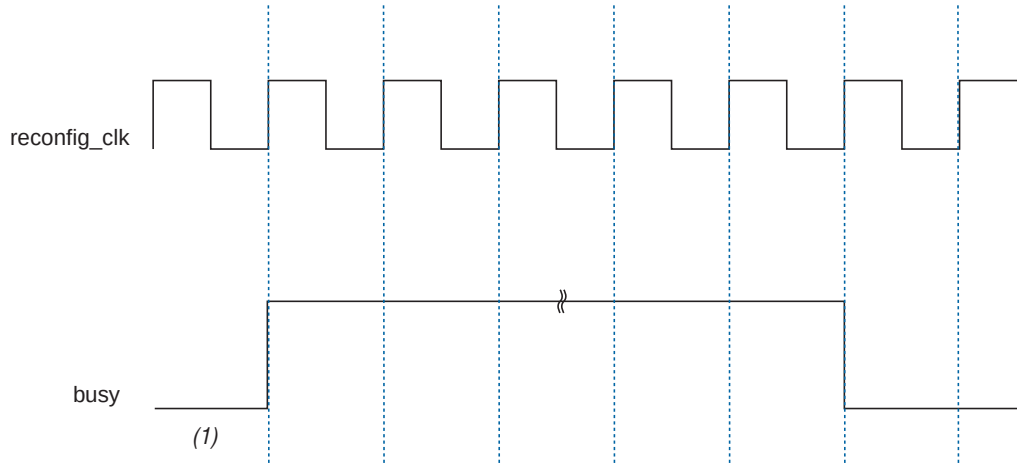
To understand the impact on system start-up when you control all the transceiver channels using a single dynamic reconfiguration controller, refer to "PMA Controls Reconfiguration Duration" on page 2–143.

In this example, the design has ALTGX instances with channels of both **Transmitter only** and **Receiver only** configurations. You must include the **Transmitter only** channels while setting the **What is the starting channel number?** option in the ALTGX instance and setting the **What is the number of channels controlled by the reconfig controller?** option in the ALTGX_RECONFIG instance for receiver offset cancellation.

- After the device powers up, the `busy` signal remains low for the first `reconfig_clk` clock cycle.
- The `busy` signal then gets asserted for the second `reconfig_clk` clock cycle, when the dynamic reconfiguration controller initiates the offset cancellation process.
- The de-assertion of the `busy` signal indicates the successful completion of the offset cancellation process.

Figure 2–16 shows the dynamic reconfiguration signals transition during offset cancellation on receiver channels.

Figure 2–16. Dynamic Reconfiguration Signals Transition during Offset Cancellation on Receiver Channels



Note to Figure 2–16:

(1) After device power up, the `busy` signal remains low for the first `reconfig_clk` cycle.

Due to the offset cancellation process, the transceiver reset sequence has changed. For more information, refer to the *Reset Control and Power Down* chapter in volume 2 of the *Stratix IV Device Handbook*.

Example for the Offset Cancellation Process of a Receiver Channel

You must always connect the ALTGX_RECONFIG instance to the ALTGX instances. Even if you do not require dynamic reconfiguration, you must set the options described in Table 2–21.

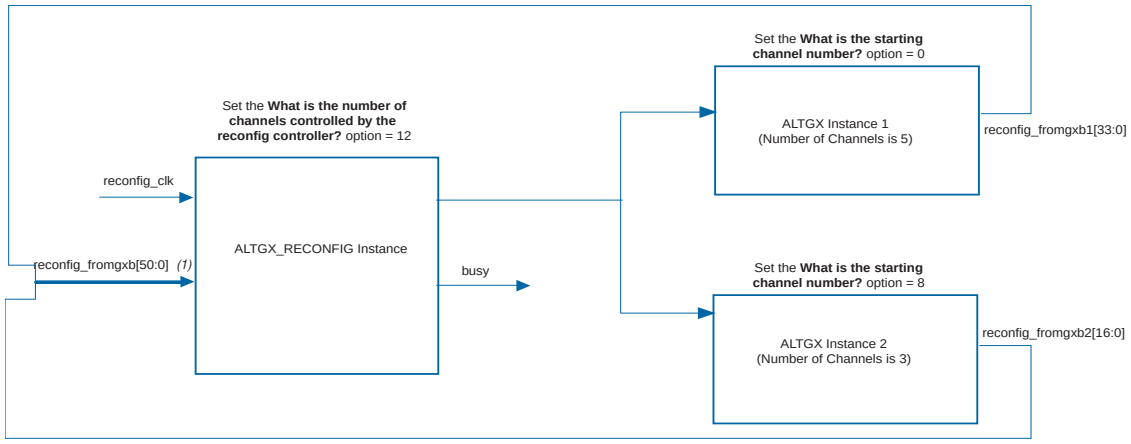
Table 2–21 describes the various ALTGX_RECONFIG and ALTGX MegaWizard Plug-In Manager settings that you must set to perform the offset cancellation process of a receiver channel.

Table 2–21. Example for the Offset Cancellation Process of a Receiver Channel

ALTGX Setting and Instances			ALTGX_RECONFIG Settings and Instance	
ALTGX Setting	**ALTGX Instance 1**	**ALTGX Instance 2**	**ALTGX_RECONFIG Setting**	**ALTGX_RECONFIG Instance 1**
What is the number of channels? option in the **General** screen	**5** (Regular transceiver channels)	**3** (Regular transceiver channels)	**What is the number of channels controlled by the reconfig controller?** option in the **Reconfiguration settings** screen. For more information about this setting, refer to "Total Number of Channels Controlled by the ALTGX_RECONFIG Instance" on page 2–35.	■ Determine the highest logical channel address (10). ■ Round it up to the next multiple of 4. ■ Set this option to **12**.
What is the starting channel number? option in the **Reconfig** screen. For more information, refer to "Logical Channel Addressing" on page 2–23.	■ Set this option to **0**. ■ The logical channel addresses of the 1st to 5th channels are **0**, **1**, **2**, **3**, and **4**, respectively.	■ Set this option to **8**. ■ The logical channel addresses of the 1st to 3rd channels are **8**, **9**, and **10**, respectively.		
`reconfig_fromgxb1` and `reconfig_fromgxb2` outputs.	`reconfig_fromgxb1` is 34 bits wide (2 * 17)	`reconfig_fromgxb2` is 17 bits wide (1 * 17)	`reconfig_fromgxb` input	`reconfig_fromgxb` is 51 bits wide (3 * 17, 12 channels can logically fit into 3 transceiver blocks)

Figure 2–17 shows the ALTGX_RECONFIG instance connected to both the ALTGX instance 1 and ALTGX instance 2.

Figure 2–17. Example of the ALTGX_RECONFIG and ALTGX Instances Set Up for the Offset Cancellation Process of a Receiver Channel



Notes to Figure 2–17:

(1) `reconfig_fromgxb[50:0]` = {`reconfig_fromgxb2[16:0]`, `reconfig_fromgxb1[33:0]`}.

(2) All the channels are regular transceiver channels.

ALTGX Instances and ALTGX_RECONFIG Instance Connections

1. Connect the `reconfig_fromgxb1[33:0]` output port from ALTGX instance 1 to the `reconfig_fromgxb[33:0]` input port of the ALTGX_RECONFIG instance.
2. Similarly, connect the `reconfig_fromgxb2[16:0]` output port from ALTGX instance 2 to the `reconfig_fromgxb[50:17]` input port of the ALTGX_RECONFIG instance.
3. Connect the `reconfig_togxb[3:0]` output port of the ALTGX_RECONFIG instance to the `reconfig_togxb[3:0]` input ports of both ALTGX instance 1 and ALTGX instance 2. For more information, refer to "Connecting the reconfig_fromgxb and reconfig_togxb Ports" on page 2–40.

Dynamic Reconfiguration Controller-Offset Cancellation Control Sequence

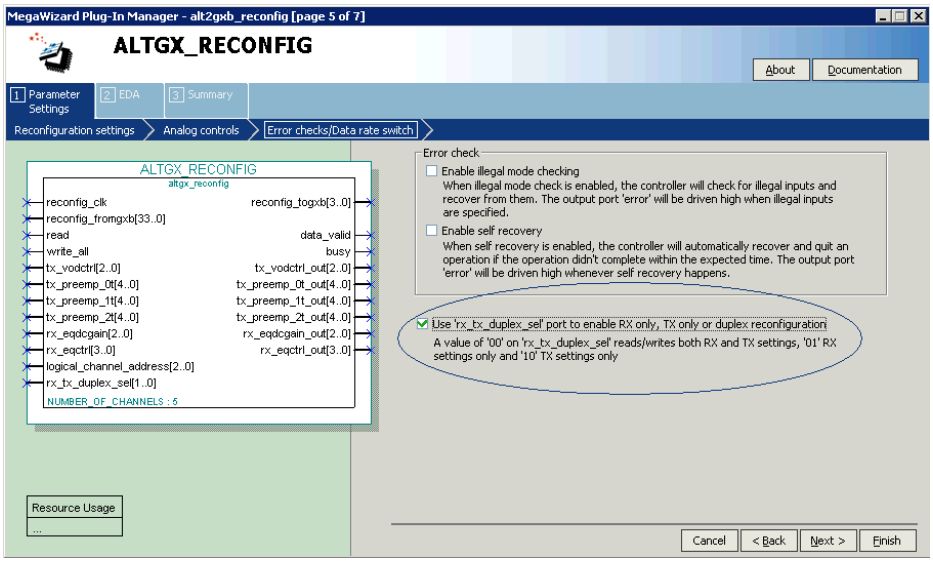
1. The ALTGX_RECONFIG instance automatically performs offset cancellation on all receiver channels of the ALTGX instances connected to it on power up.
2. The `busy` signal is low for the first `reconfig_clk` clock cycle after power up.
3. The `busy` signal gets asserted for the second `reconfig_clk` clock cycle after power up.
4. The de-assertion of the `busy` signal indicates the successful completion of the offset cancellation process.

The rx_tx_duplex_sel[1:0] Port

The width of the `rx_tx_duplex_sel` port is fixed to 2 bits. The `rx_tx_duplex_sel[1:0]` port is available for selection in all dynamic reconfiguration modes.

You can enable this port by selecting the **Use 'rx_tx_duplex_sel' port to enable RX only** and **TX only or duplex reconfiguration** options in the **Error checks/Data rate switch** screen of the ALTGX_RECONFIG MegaWizard Plug-In Manager, as shown in Figure 2–18.

Figure 2–18. Use 'rx_tx_duplex_sel' Port to Enable RX Only Option and the TX only or Duplex Reconfiguration Options in the ALTGX_RECONFIG MegaWizard Plug-In Manager



This option is available only when you select one of the dynamic reconfiguration modes (in addition to the default **Offset cancellation for Receiver channels** option) in the **Reconfiguration settings** screen of the ALTGX_RECONFIG MegaWizard Plug-In Manager. Table 2–22 shows the settings for the `rx_tx_duplex_sel[1:0]` input port of the ALTGX_RECONFIG instance for this port.

Table 2–22. Setting the rx_tx_duplex_sel[1:0] Input Port of the ALTGX_RECONFIG Instance

rx_tx_duplex_sel[1:0]	Reconfiguration Mode
2'b00	Receiver and Transmitter
2'b01	Receiver only
2'b10	Transmitter only
2'b11	Unsupported value

Note to Table 2–22:

(1) For more information, refer to "Dynamic Reconfiguration Controller Port List" on page 2–11.

For **.mif**-based dynamic reconfiguration modes, the advantage of using the `rx_tx_duplex_sel[1:0]` port is that you can inform the dynamic reconfiguration controller to load only the receiver settings, transmitter settings, or both receiver and transmitter settings, through this port.

Consider a scenario where a **Receiver only** ALTGX instance is controlled by the dynamic reconfiguration controller and the **.mif** contains all the transmitter settings. Enabling the `rx_tx_duplex_sel[1:0]` port in this scenario and setting it to **2'b01** ensures that the dynamic reconfiguration controller writes nothing into the physical transmitter portion of the **Receiver only** ALTGX instance (because the **.mif** contains transmitter settings that are not applicable to the **Receiver only** configuration).

The logical_channel_address Port

The `logical_channel_address` port is used to select a specific channel that is controlled by the ALTGX_RECONFIG instance.

The `logical_channel_address` port is optional for the following mode:

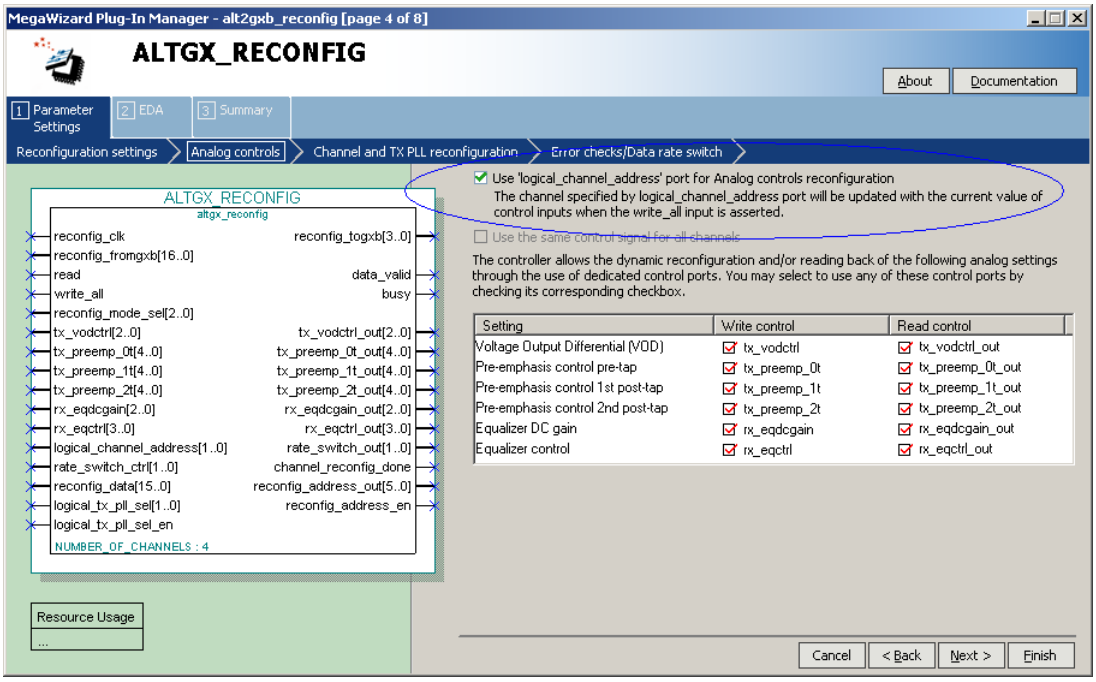
- PMA controls reconfiguration mode

However, it is always available for the following dynamic reconfiguration modes:

- Data Rate Division in TX mode
- Channel and TX PLL select/reconfig modes

Figure 2–19 shows the **Use 'logical_channel_address' port for Analog controls reconfiguration** option in the **Analog controls** screen of the ALTGX_RECONFIG MegaWizard Plug-In Manager.

Figure 2–19. The logical_channel_address Port in the ALTGX_RECONFIG MegaWizard Plug-In Manager



The width of the `logical_channel_address` port depends on the following two conditions:

- What number you set in the **What is the number of channels controlled by the reconfig controller?** option in the **Reconfiguration Settings** screen. The maximum width of this port is 9 bits (because 9 bits are required to represent a maximum number of 384 channels).
- Whether the channel controlled by the ALTGX_RECONFIG instance is a regular transceiver channel or PMA-only channel.

Table 2–23 shows the width of the `logical_channel_address` port set differently for regular transceiver channels and PMA-only channels.

Table 2–23. Width of the logical_channel_address Port for Regular Transceiver Channels versus PMA-Only Channels

Eight Regular Transceiver Channels	Eight PMA-Only Channels
`logical_channel_address` port width = 3 bits	`logical_channel_address` port width = 5 bits

PMA Controls Reconfiguration

You can reconfigure the following PMA controls:

- Pre-emphasis settings
- Equalization settings
- DC gain settings
- Voltage output differential (V_{OD}) settings

CMU Channels

These are CMU PLLs configured in Basic (PMA Direct) functional mode (PMA-only channels).

For the CMU channels, dynamic reconfiguration involves reconfiguring the following:

- PMA controls reconfiguration only

For Basic (PMA Direct) ×1 and ×N configurations, only PMA controls reconfiguration is available.

The following dynamic reconfigurations CANNOT be achieved for the CMU channels in the HardCopy IV GX device:

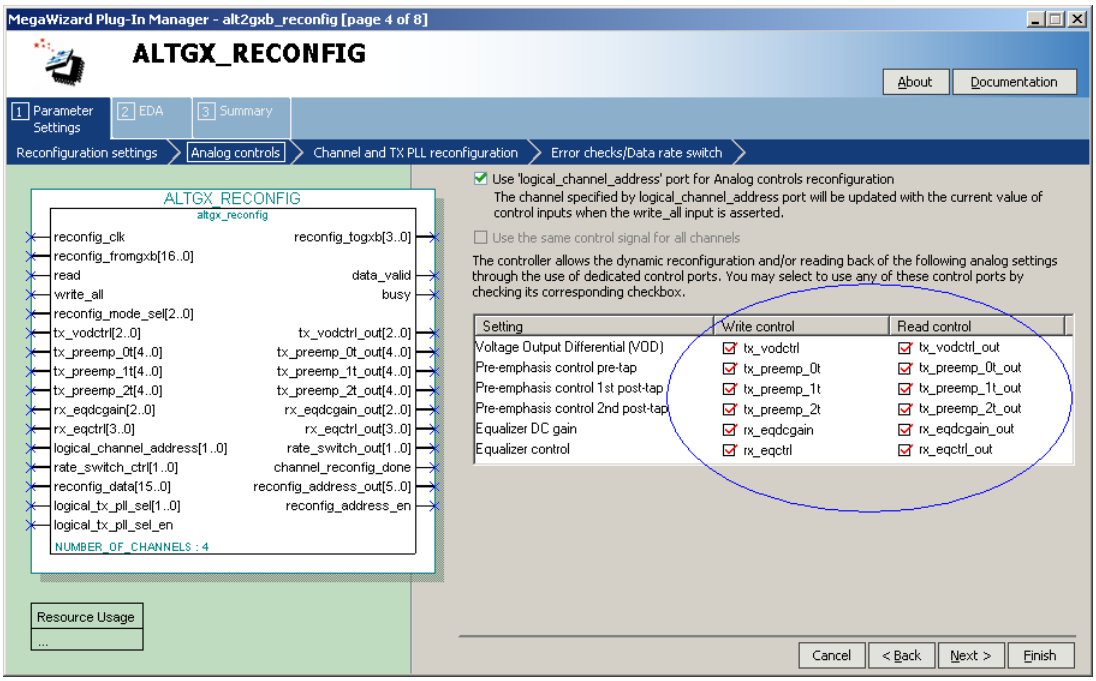
- ×1 to ×N, or vice versa
- ×N mode to another ×N mode

PMA controls reconfiguration is available for both regular transceiver channels and CMU channels.

Dynamic Reconfiguration Controller Ports for PMA Controls

The PMA control ports for the ALTGX_RECONFIG MegaWizard Plug-In Manager are available in the **Analog controls** screen, as shown in Figure 2–20. Here you can select the PMA control ports you want to reconfigure (for example, use `tx_vodctrl` to write new V_{OD} settings or `tx_vodctrl_out` to read the existing V_{OD} settings).

Figure 2–20. Dynamic Reconfiguration Controller Ports for PMA Controls in the ALTGX_RECONFIG MegaWizard Plug-In Manager



Dynamically Reconfiguring PMA Controls

You can dynamically reconfigure the PMA controls of a transceiver channel using two methods:

- Method 1—You can reconfigure the PMA controls of a specific transceiver channel. For more information, refer to "Method 1" on page 2–54.
- Method 2—You can dynamically reconfigure the PMA controls of the transceiver channels without using the `logical_channel_address` port. If you use this method, the PMA controls of all the transceiver channels connected to the dynamic reconfiguration controller are reconfigured. For more information, refer to "Method 2" on page 2–56.

For both methods, you can additionally use the `rx_tx_duplex_sel[1:0]` port. The width of this port is fixed to 2 bits.

The two methods are described in the following sections.

Method 1

Using Method 1, you can dynamically reconfigure the PMA controls of a transceiver channel by using the `logical_channel_address` port without affecting the remaining active channels. You can enable the `logical_channel_address_port` port by selecting the **Use 'logical_channel_address' port for Analog controls reconfiguration** option in the **Analog controls** screen, as shown in Figure 2–19 on page 2–51. This method is applicable only for a design where the dynamic reconfiguration controller controls more than one channel.

You can additionally reconfigure either the receiver portion, transmitter portion, or both the receiver and transmitter portions of the transceiver channel by setting the corresponding value on the `rx_tx_duplex_sel[1:0]` input port.

Connecting the PMA Control Ports

When using Method 1, the selected PMA control ports remain fixed in width, regardless of the number of channels controlled by the ALTGX_RECONFIG instance:

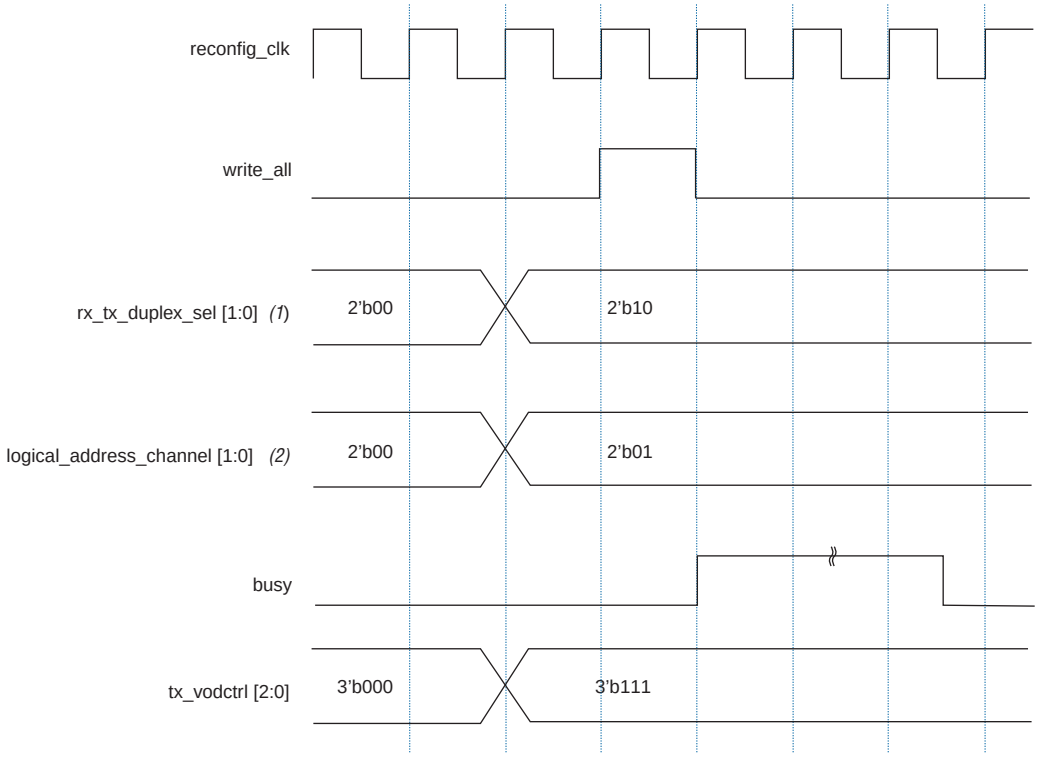
- `tx_vodctrl` and `tx_vodctrl_out` are fixed to 3 bits
- `tx_preemp_0t`, `tx_preemp_1t`, `tx_preemp_2t`, `tx_preemp_0t_out`, `tx_preemp_1t_out` and `tx_preemp_2t_out` are fixed to 5 bits
- `rx_eqdcgain` and `rx_eqdcgain_out` are fixed to 3 bits
- `rx_eqctrl` and `rx_eqctrl_out` are fixed to 4 bits

Write Transaction

Set the selected PMA control ports to the desired settings (for example, `tx_vodctrl` = 3'b000). Set the input port `logical_channel_address` to the logical channel address of the transceiver channel whose PMA controls you want to reconfigure. Set the `rx_tx_duplex_sel[1:0]` port to **2'b10** so that only the transmit PMA controls are written to the transceiver channel. Ensure that the `busy` signal is low before you start a write transaction. Assert the `write_all` signal for one `reconfig_clk` clock cycle. This initiates the write transaction.

The `busy` output status signal is asserted high to indicate that the dynamic reconfiguration controller is busy writing the PMA control values. When the write transaction has completed, the `busy` signal goes low. Figure 2–21 shows the write transaction waveform.

Figure 2–21. Method 1—Write Transaction Waveform



Notes to Figure 2–21:

(1) Consider that you want to write to only the transmitter portion of the channel.

(2) This waveform assumes that the number of channels connected to the dynamic reconfiguration controller is four. Therefore, the `logical_channel_address` port is 2 bits wide.

Read Transaction

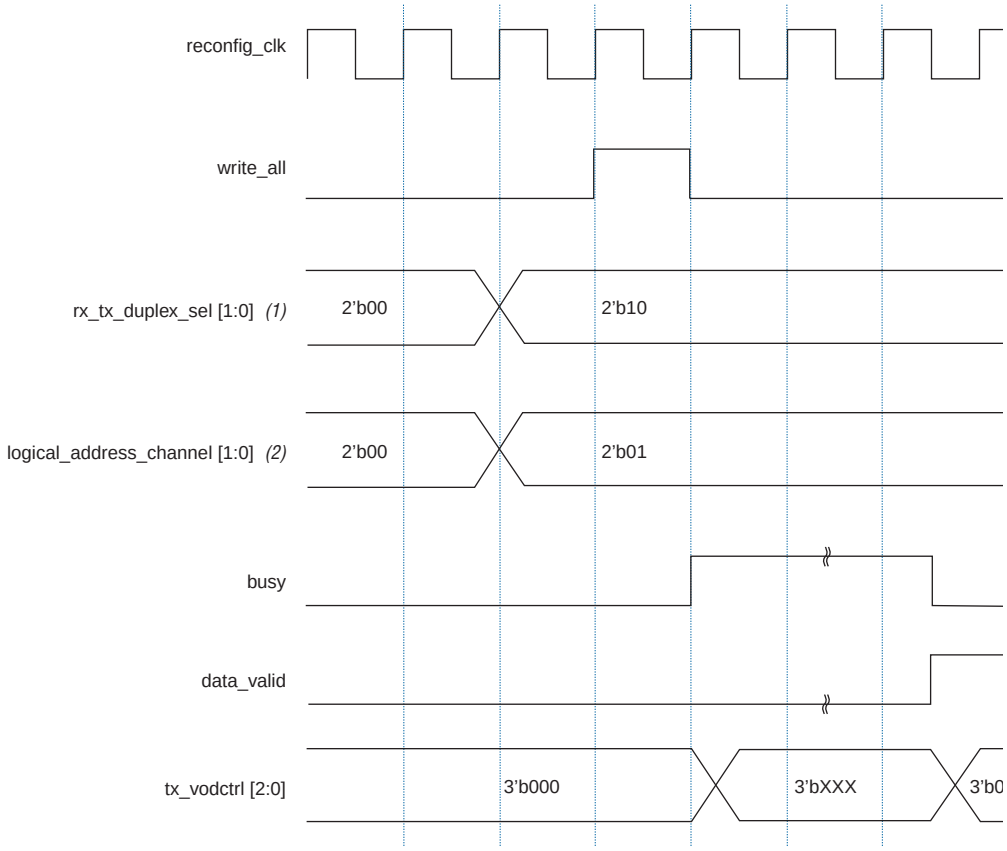
In this example, you want to read the existing V_{OD} values from the transmit V_{OD} control registers of the transmitter portion of a specific channel controlled by the ALTGX_RECONFIG instance. The read transaction in this scenario is explained in the following steps:

1. Set the input `logical_channel_address` port to the logical channel address of the transceiver channel whose PMA controls you want to read (for example, `tx_vodctrl_out`).
2. Set the `rx_tx_duplex_sel[1:0]` port to **2'b10** so that only the transmit PMA controls are read from the transceiver channel.
3. Ensure that the `busy` signal is low before you start a read transaction.
4. Assert the `read` signal for one `reconfig_clk` clock cycle. This initiates the read transaction.

The `busy` output status signal is asserted high to indicate that the dynamic reconfiguration controller is busy reading the PMA control values. Once the read transaction has completed, the `busy` signal goes low. The `data_valid` signal gets asserted indicating that the data available at the read control signal is valid.

Figure 2–22 shows the read transaction waveform.

Figure 2–22. Method 1—Read Transaction Waveform



Notes to Figure 2–22:

(1) Consider that you want to read from only the transmitter portion of the channel.

(2) This waveform assumes that the number of channels connected to the dynamic reconfiguration controller is four. Therefore, the `logical_channel_address` port is 2 bits wide.

Simultaneous write and read transactions are not allowed.

Method 2

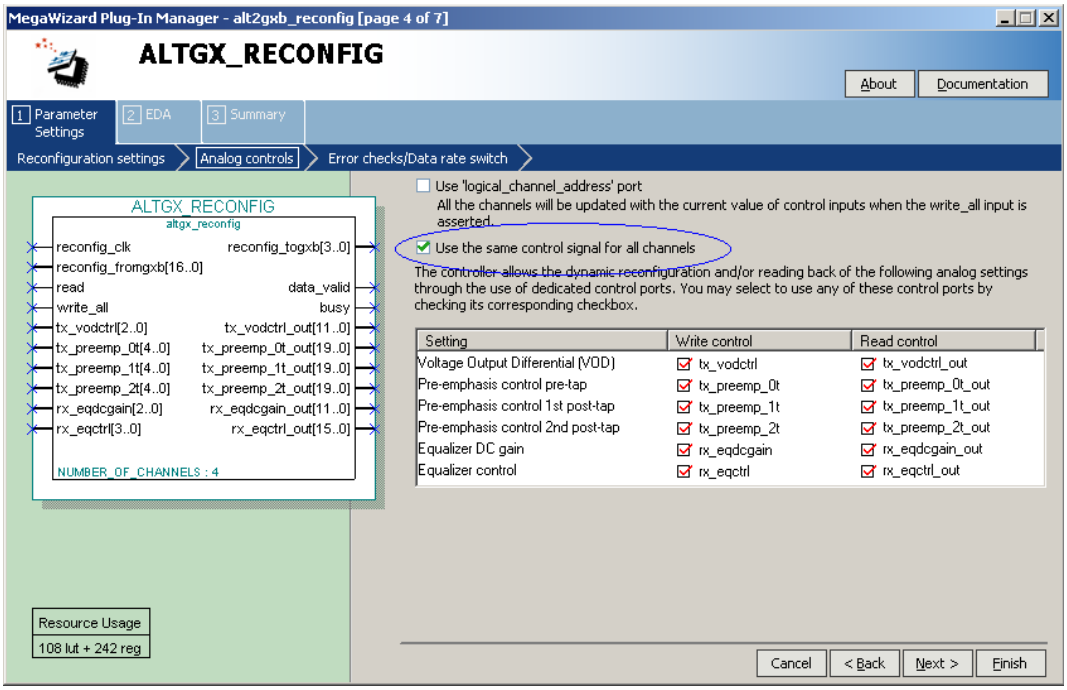
Method 2 does not require the `logical_channel_address` port to dynamically reconfigure the PMA controls of the transceiver channels. Using Method 2, the PMA controls of all the transceiver channels connected to the ALTGX_RECONFIG instance are reconfigured.

Method 2 is further classified into:

- The **Use the same control signal for all the channels** option enabled
- The **Use the same control signal for all the channels** option disabled

Figure 2–23 shows the **Use the same control signal for all channels** option in the ALTGX_RECONFIG MegaWizard Plug-In Manager.

Figure 2–23. Use the Same Control Signals for all Channels Option in the ALTGX_RECONFIG MegaWizard Plug-In Manager



Use the Same Control Signal for All Channels Option Enabled

The **Use the same control signal for all channels** option is available in the **Analog controls** screen of the ALTGX_RECONFIG MegaWizard Plug-In Manager. If enabled, the width of the PMA control ports are fixed as shown below.

PMA Control Ports Used in a Write Transaction:

- `tx_vodctrl` is fixed to 3 bits
- `tx_preemp_0t`, `tx_preemp_1t`, and `tx_preemp_2t` are fixed to 5 bits
- `rx_eqdcgain` is fixed to 3 bits
- `rx_eqctrl` is fixed to 4 bits

PMA Control Ports Used in a Read Transaction:

- `tx_vodctrl_out` is 3 bits per channel
- `tx_preemp_0t_out`, `tx_preemp_1t_out`, and `tx_preemp_2t_out` are 5 bits per channel
- `rx_eqdcgain_out` is 3 bits per channel
- `rx_eqctrl_out` is to 4 bits per channel

For example, if the number of channels controlled by the dynamic reconfiguration controller is 2, `tx_vodctrl_out` is 6 bits wide.

Write Transaction

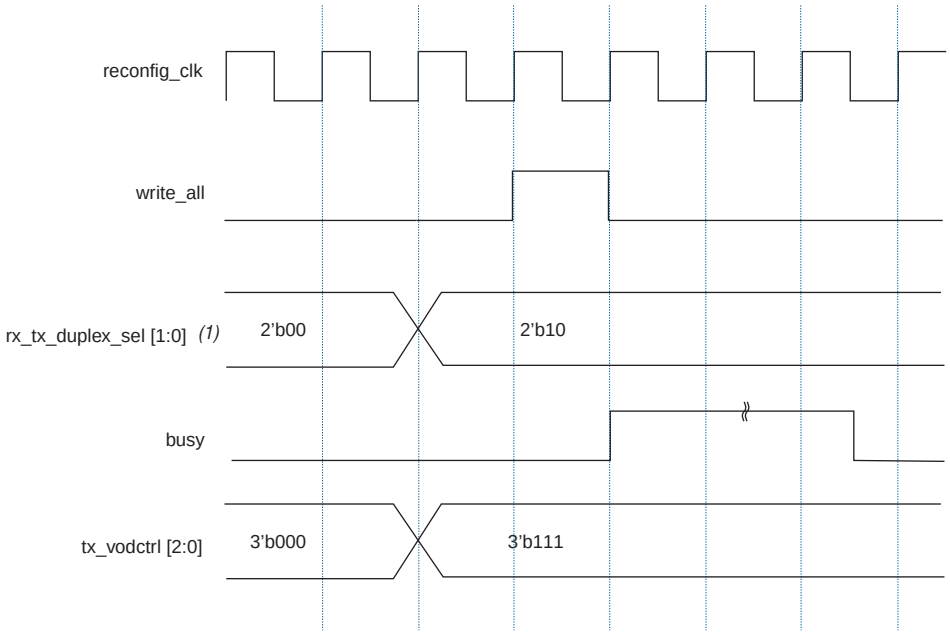
The value you set at the selected PMA control ports gets written to all the transceiver channels connected to the ALTGX_RECONFIG instance.

Consider that you have enabled `tx_vodctrl` in the ALTGX_RECONFIG MegaWizard Plug-In Manager to reconfigure the V_{OD} of the transceiver channels.

The following are the steps involved in the write transaction to reconfigure the V_{OD}, as shown in Figure 2–24:

1. Before you initiate a write transaction, set the selected PMA control ports to the desired settings (for example, `tx_vodctrl` = 3'b000).
2. Set the `rx_tx_duplex_sel[1:0]` port to **2'b10** so that only the transmit PMA controls are written to the transceiver channel.
3. Ensure that the `busy` signal is low before you start a write transaction.
4. Assert the `write_all` signal for one `reconfig_clk` clock cycle. This initiates the write transaction.
5. The `busy` output status signal is asserted high to indicate that the dynamic reconfiguration controller is busy writing the PMA control values. Once the write transaction has completed, the `busy` signal goes low.

Figure 2–24. Method 2: Write Transaction Waveform—Use the Same Control Signal for All Channels Option Enabled



Note to Figure 2–24:

(1) Consider that you want to write to only the transmitter portion of the channel.

Read Transaction

If you want to read the existing values from a specific channel connected to the ALTGX_RECONFIG instance, observe the corresponding byte positions of the PMA control output port after the read transaction is complete.

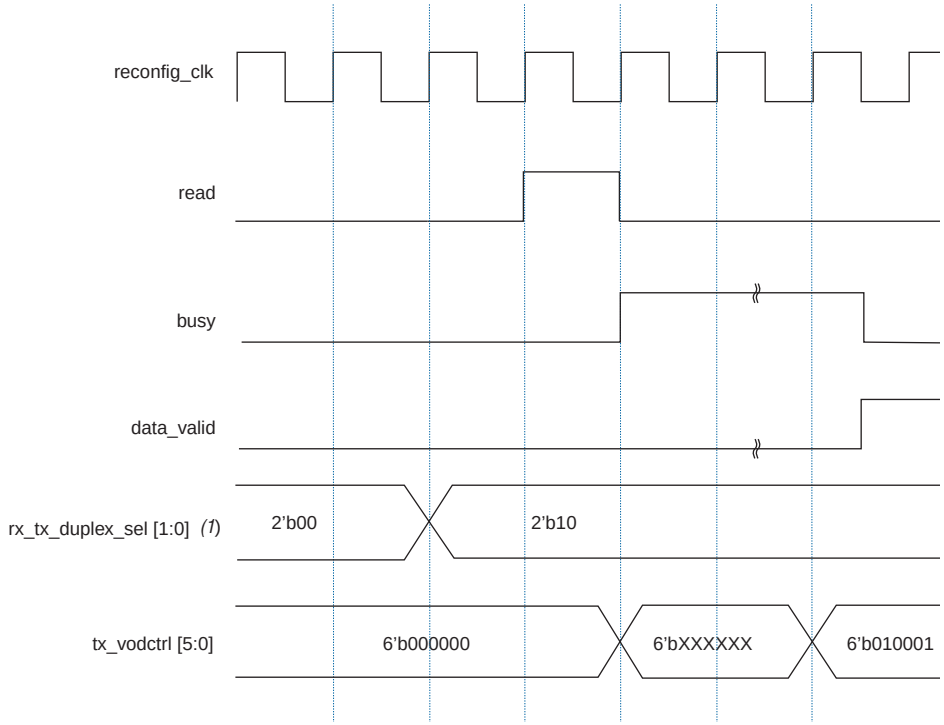
For example, if the number of channels controlled by the ALTGX_RECONFIG instance is 2, the `tx_vodctrl_out` is 6 bits wide. The `tx_vodctrl_out[2:0]` corresponds to channel 1 and similarly, `tx_vodctrl_out[5:3]` corresponds to channel 2.

The following list shows the steps to read the V_{OD} values of the second channel:

1. Before you initiate a read transaction, set the `rx_tx_duplex_sel[1:0]` port to **2'b10** so that only the transmit PMA controls are read from the transceiver channel.
2. Ensure that the `busy` signal is low before you start a read transaction.
3. Assert the `read` signal for one `reconfig_clk` clock cycle. This initiates the read transaction.
4. The `busy` output status signal is asserted high to indicate that the dynamic reconfiguration controller is busy reading the PMA control settings.
5. Once the read transaction has completed, the `busy` signal goes low. The `data_valid` signal is asserted, indicating that the data available at the read control signal is valid. To read the current V_{OD} values in channel 2, observe the values in `tx_vodctrl_out[5:3]`.

Figure 2–25 shows the read transaction waveform. It assumes that the transmit V_{OD} settings written in channels 1 and 2 prior to the read transaction are 3'b001 and 3'b010, respectively.

Figure 2–25. Method 2—Read Transaction Waveform



Note to Figure 2–25:

(1) Consider that you want to read from only the transmitter portion of all the channels.

Simultaneous write and read transactions are not allowed.

Use the Same Control Signal for All Channels Option Disabled

When the **Use the same control signal for all channels** option is disabled, the PMA control ports for write transaction are separate for each channel.

PMA Control Ports Used in a Write Transaction:

- `tx_vodctrl` is 3 bits per channel
- `tx_preemp_0t`, `tx_preemp_1t`, and `tx_preemp_2t` are 5 bits per channel
- `rx_eqdcgain` is 3 bits per channel
- `rx_eqctrl` is 4 bits per channel

For example, if you have two channels, `tx_vodctrl` is 6 bits wide (`tx_vodctrl[2:0]` corresponds to channel 1 and `tx_vodctrl[5:3]` corresponds to channel 2).

PMA Control Ports Used in a Read Transaction

The width of the PMA control ports for a read transaction are always separate for each channel (the same as the PMA control ports, as explained in "Use the Same Control Signal for All Channels Option Enabled" on page 2–57).

Write Transaction

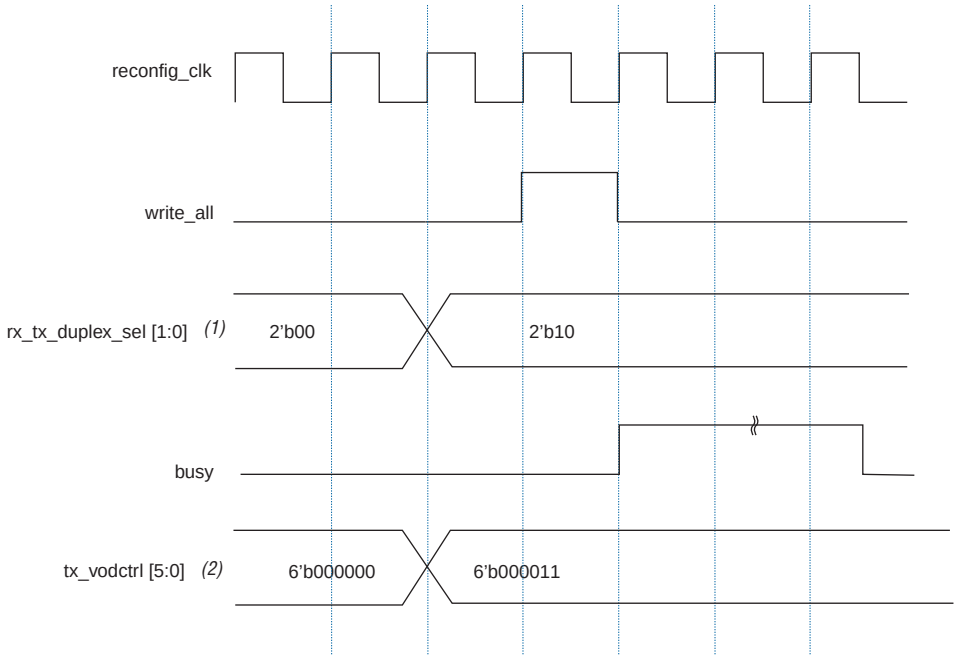
Because the PMA controls of all channels are written, if you want to reconfigure a specific channel connected to the ALTGX_RECONFIG instance, set the new value to the corresponding PMA control port of the channel under consideration and retain the previously stored values in the other active channels using a read transaction prior to this write transaction.

For example, assume that the number of channels controlled by the ALTGX_RECONFIG is 2, `tx_vodctrl` in this case is 6 bits wide. The `tx_vodctrl[2:0]` corresponds to channel 1, and similarly, `tx_vodctrl[5:3]` corresponds to channel 2. Follow these steps:

1. If you want to dynamically reconfigure the PMA controls of only channel 2 with a new value, first perform a read transaction to retrieve the existing PMA control values from `tx_vodctrl_out[5:0]`. Take `tx_vodctrl_out[2:0]` and provide this value in `tx_vodctrl[2:0]` to the write in channel 1. By doing so, channel 1 gets overwritten with the same value.
2. Perform a write transaction. This ensures that the new values are written only to channel 2, while channel 1 remains unchanged.

Figure 2–26 shows a write transaction waveform with the **Use the same control signal for all channels** option disabled.

Figure 2–26. Method 2: Write Transaction Waveform with the Use the Same Control Signal for All Channels Option Disabled



Notes to Figure 2–26:

(1) Consider that you want to write to only the transmitter portion of the channel.

(2) The waveform assumes that the number of channels controlled by the dynamic reconfiguration controller (ALTGX_RECONFIG instance) is 2 and that the `tx_vodctrl` control port is enabled.

Simultaneous write and read transactions are not allowed.

Read Transaction

The read transaction is explained in "Read Transaction" on page 2–59.

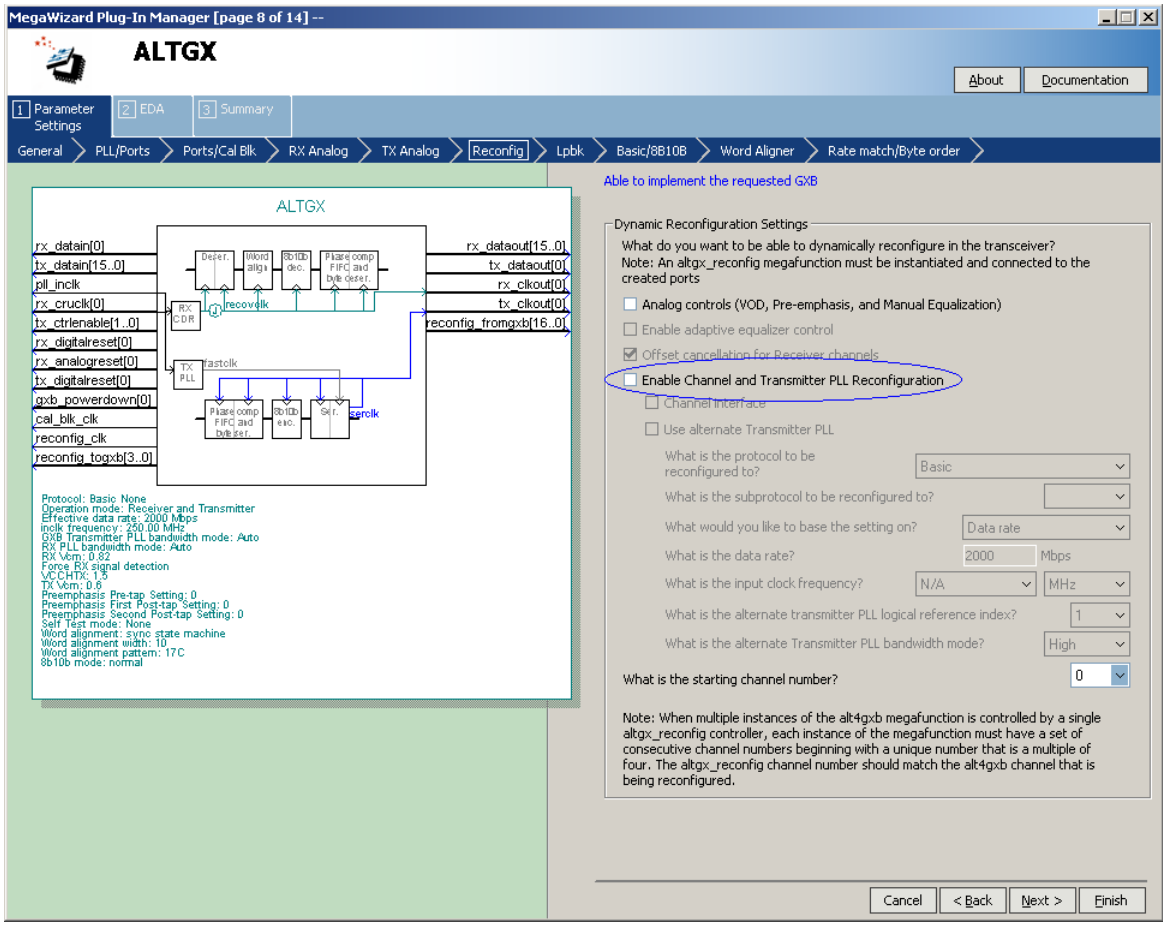
Description of Transceiver Channel Reconfiguration Modes

This section describes the following dynamic reconfiguration modes:

- Data Rate Division in TX mode
- Channel and TX PLL select/reconfig modes
  - Channel and CMU PLL reconfiguration
  - Channel reconfiguration with TX PLL select
  - CMU PLL reconfiguration

You can enable all the above dynamic reconfiguration modes together by selecting the **Enable Channel and Transmitter PLL Reconfiguration** option in the **Reconfig** screen of the ALTGX MegaWizard Plug-In Manager (shown in Figure 2–27). (You do not have the option to individually enable any of the above dynamic reconfiguration modes in the ALTGX MegaWizard Plug-In Manager.)

Figure 2–27. The Enable Channel and Transmitter PLL Reconfiguration Option in the ALTGX MegaWizard Plug-In Manager



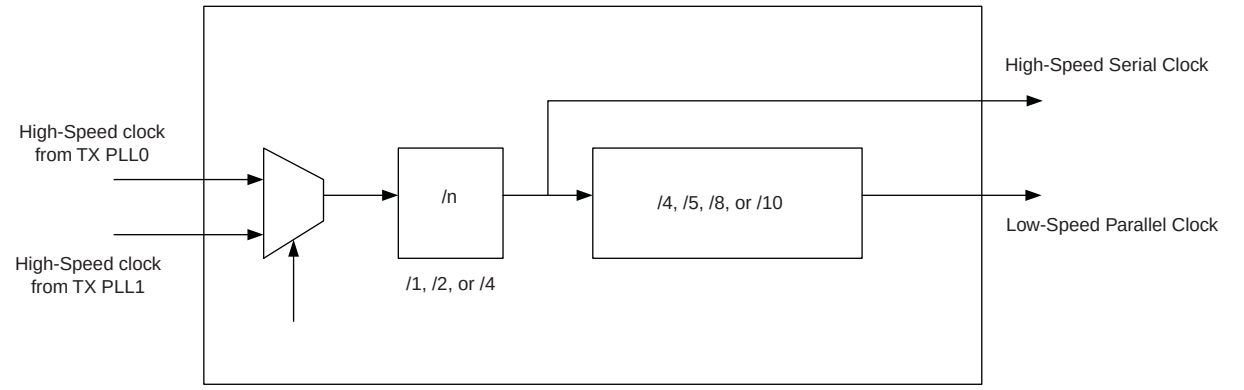
Data Rate Division in TX Mode

You can use the Date Rate Division in TX mode to modify the data rate of the transmitter channel in multiples of 1, 2, and 4. This dynamic reconfiguration mode is available only for the transmit side and not for the receive side.

Blocks Reconfigured in the Data Rate Division in TX Mode

The only block that is reconfigured by this mode is the TX local divider block of a transmitter channel. The TX local divider can be set to a divide by value of /1, /2, or /4, as shown in Figure 2–28.

Figure 2–28. Local Divider of a Transmitter Channel



You must be aware of the device operating range before you enable and use this feature. There are no legal checks that are imposed by the Quartus II software because it is an on-the-fly control feature. You also need to ensure that a specific functional mode supports the data rate range before dividing the clock when using this rate switch option.

The Date Rate Division in TX mode is applicable only to regular transceiver channels and not the PMA-only channels.

ALTGX MegaWizard Plug-In Manager Setup

Enable the following settings in the ALTGX MegaWizard Plug-In Manager:

1. Select the **Channel and Transmitter PLL Reconfiguration** option in the **Reconfig** screen to enable the ALTGX_RECONFIG instance to modify the TX channel local divider values dynamically.
2. Set the **What is the starting channel number?** option in the **Reconfig** screen. For more information, refer to "Logical Channel Addressing" on page 2–23.

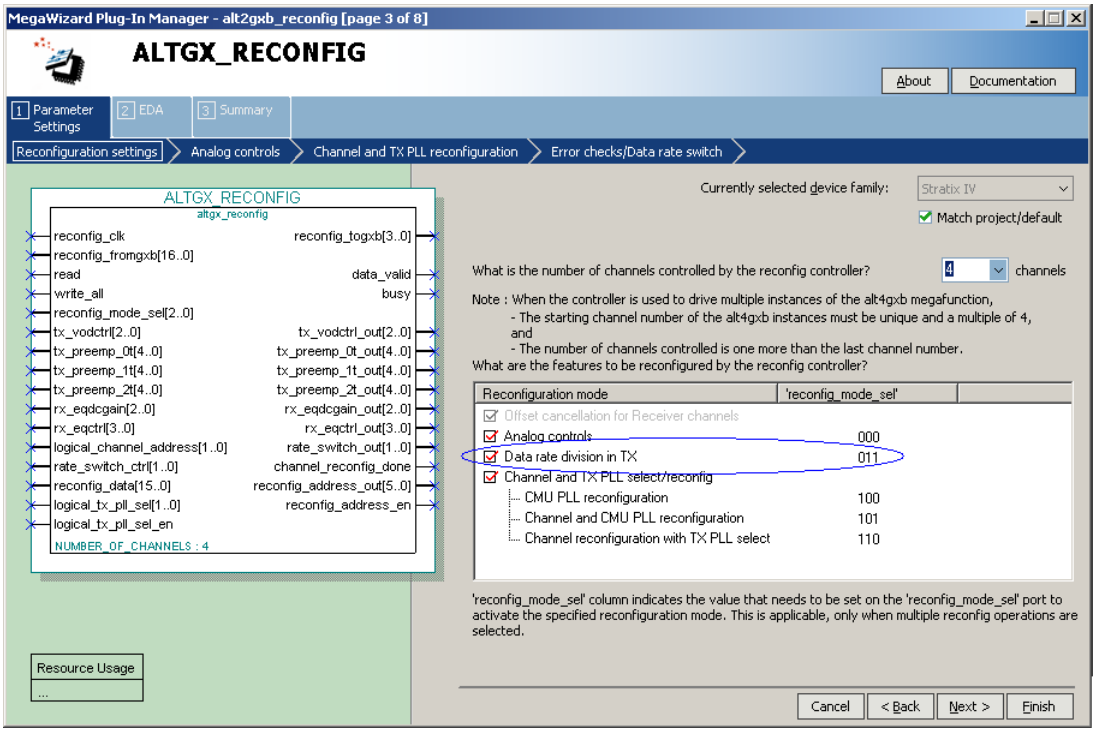
The alternate reference clock is not required because a single clock source is used. The /1, /2, or /4 data rates can be derived from the single input reference clock.

ALTGX_RECONFIG MegaWizard Plug-In Manager Setup

Enable the following settings in the ALTGX_RECONFIG MegaWizard Plug-In Manager for Data Rate Division in TX mode:

1. Set the **What is the number of channels controlled by the reconfig controller?** option in the **Reconfiguration settings** screen. For more information, refer to "Total Number of Channels Controlled by the ALTGX_RECONFIG Instance" on page 2–35.
2. Specify the logical channel address of the transmitter channel at the `logical_channel_address` input port.
3. Select the **Data rate division in TX** option in the **Reconfiguration settings** screen, as shown in Figure 2–29.

Figure 2–29. The Data Rate Division in TX Option in the ALTGX_RECONFIG MegaWizard Plug-In Manager



4. The `rate_switch_ctrl[1:0]` input port is available when you enable the **Data rate division in TX** option. The value you set at the `rate_switch_ctrl[1:0]` signal determines the TX local divider settings, as shown in Table 2–24.

Table 2–24. TX Local Divider Settings Based on the rate_switch_ctrl[1:0] Port

rate_switch_ctrl[1:0]	Local Divider Settings
2'b00	Divide by 1
2'b01	Divide by 2
2'b10	Divide by 4
2'b11	Not supported

If you want to read the existing local divider settings of the transmitter channel, select the **Use 'rate_switch_out' port to read out the current data rate division** option in the **Error checks/Data rate switch** screen.

Decoding for the `rate_switch_out[1:0]` output signal is the same as the `rate_switch_ctrl[1:0]` input signal.

Dynamic rate switch has no effect on the dividers on the receive side of the transceiver channel. It can be used only for the transmitter.

The Data Rate Division in TX mode does not require a **.mif**.

For more information about read and write transactions, refer to "Data Rate Division in TX: Operation" in the following section.

Data Rate Division in TX: Operation

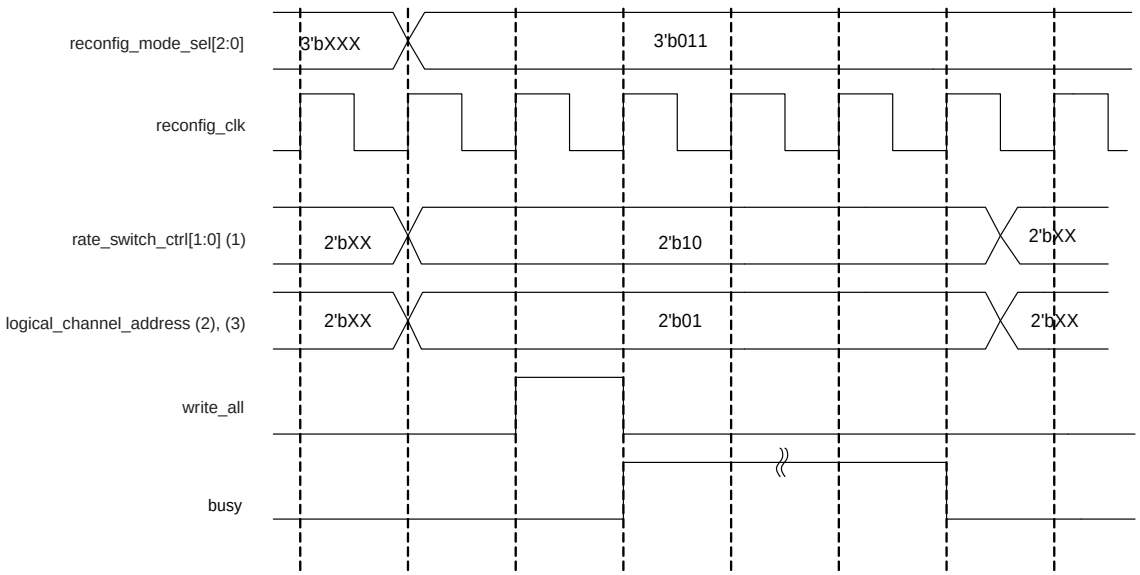
The following sections describe the steps involved in write and read transactions for the Data Rate Division in TX mode.

Data Rate Division in TX: Write Transaction

1. Set the `reconfig_mode_sel[2:0]` signal to **3'b011** to activate this mode.
2. Set the `rate_switch_ctrl[1:0]` signal to the corresponding TX local divider setting.
3. Set the `logical_channel_address` port to the logical channel address of the transmitter channel whose local divider settings you want to reconfigure.
4. Ensure that the `busy` signal is low.
5. Initiate a write transaction by asserting the `write_all` signal for one `reconfig_clk` cycle.

Figure 2–30 shows a write transaction in Data Rate Division in TX mode.

Figure 2–30. Write Transaction in Data Rate Division in TX Mode



Notes to Figure 2–30:

(1) The waveform assumes that you want to reconfigure the local divider settings of the transmitter channel to "Divide by 4". Therefore, the value set at `rate_switch_ctrl[1:0]` is **2'b10**.

(2) The waveform assumes that the value set in the **What is the number of channels controlled by the reconfig controller?** option of the ALTGX_RECONFIG MegaWizard Plug-In Manager is **4**. Therefore, the `logical_channel_address` input is 2 bits wide.

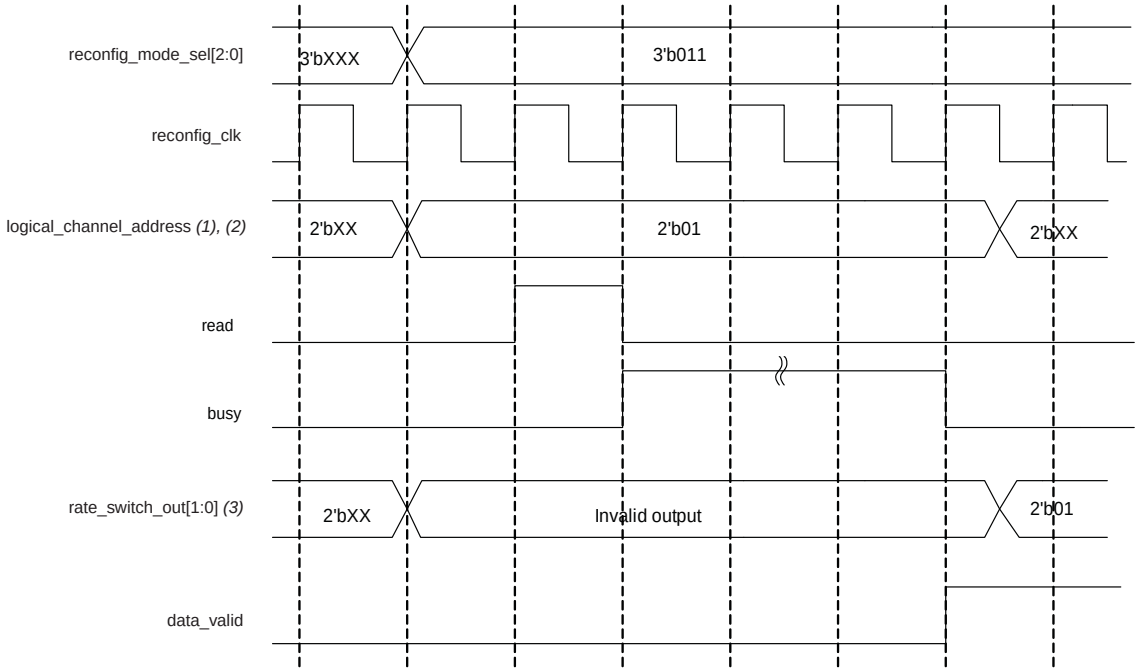
(3) The waveform also assumes that you want to reconfigure the local divider settings of the transmitter channel whose logical channel address is 2'b01.

Data Rate Division in TX: Read Transaction

1. Set the `reconfig_mode_sel[2:0]` signal to **3'b011** to activate this mode.
2. Select the `rate_switch_out[1:0]` signal to read out the existing TX local divider settings.
3. Set the `logical_channel_address` port to the logical channel address of the transmitter channel whose local divider settings you want to read.
4. Ensure that the `busy` signal is low.
5. Assert the read signal for one `reconfig_clk` cycle.

Figure 2–31 shows a read transaction waveform in Data Rate Division in TX mode.

Figure 2–31. Read Transaction in Data Rate Division in TX Mode



Notes to Figure 2–31:

(1) The waveform assumes that the existing local divider settings of the transmitter channel are "Divide by 2". Therefore, the value read out at `rate_switch_out[1:0]` is **2'b01**.

(2) The waveform assumes that the value set in the **What is the number of channels controlled by the reconfig controller?** option of the ALTGX_RECONFIG MegaWizard Plug-In Manager is **4**. Therefore, the `logical_channel_address` input is 2 bits wide.

(3) The waveform assumes that you want to read the existing local divider settings of the transmitter channel whose logical channel address is 2'b01.

Do not perform a read transaction in Date Rate Division in TX mode if `rate_switch_out[1:0]` is not selected in the ALTGX_RECONFIG MegaWizard Plug-In Manager.

Channel and TX PLL Select/Reconfig Modes

A **.mif** is required when you want to dynamically:

- Enable or disable functional blocks of the transceiver channel (use Channel and CMU PLL reconfiguration mode)
- Reconfigure the functional mode of the transceiver channel from one to another (use Channel and CMU PLL reconfiguration mode)
- Reconfigure the core fabric Transceiver channel interface from one width to another (use Channel and CMU PLL reconfiguration mode)
- Reconfigure the data rate of the transceiver channel by selecting another transmitter PLL (use Channel reconfiguration with TX PLL select mode)
- Reconfigure the data rate of the transceiver channel by reconfiguring the CMU PLL (use CMU PLL Reconfiguration mode)

.mif Generation

To understand using **.mifs**, it is helpful to understand these two concepts:

- How to generate a **.mif**?—The Quartus II software generates **.mifs** when you provide the appropriate project settings (for more information, refer to "Quartus II Settings for .mif Generation" on page 2–70) and then compile an ALTGX instance.
- How is a **.mif** used between the ALTGX_RECONFIG instance and the ALTGX instance?—The Quartus II software provides a design flow called the user memory initialization file flow, as explained below.

.mif-Based Design Flow

The **.mif**-based design flow involves writing the contents of the **.mif** to the transceiver channel or CMU PLL.

When you want to reconfigure the transceiver channel or CMU PLL, you must configure the required settings for the transceiver channel or CMU PLL in the ALTGX MegaWizard Plug-In Manager and compile the ALTGX instance. The dynamic reconfiguration controller requires that you write these configured settings through the **.mif** into the transceiver channel or CMU PLL (using the `write_all` and `reconfig_data[15:0]` signals). The maximum possible size of the **.mif** is 55 words. Each word contains legal register settings of the transceiver channel stored in 16 bits. `reconfig_address_out[5:0]` provides the address (location) of the 16-bit word in the **.mif**.

The **.mif** size depends on the ALTGX configuration shown in Table 2–25.

Table 2–25. .mif Size for the ALTGX Configuration

ALTGX Configuration	.mif Size in Words *(1)*
Duplex (Receiver and Transmitter)	55
Receiver only	37
Transmitter only	19

Note to Table 2–25:

(1) Each word in the **.mif** is 16 bits wide.

The dynamic reconfiguration controller ignores a new 16-bit word if the previously initiated write transaction is not complete. As explained previously, an ongoing or active write transaction is signified by the `busy` signal. You can only input a new word of 16 bits when the `busy` signal is de-asserted.

The Quartus II software creates the **.mif** under the <*Project_DIR*>*/reconfig_mif* folder. The file name is based on the ALTGX instance name (<*instance name*>**.mif**), for example, *basic_gxb***.mif**.

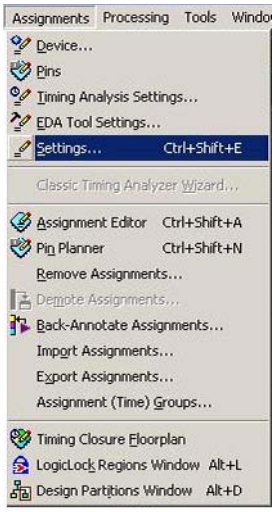
One design can have multiple **.mifs** (there is no limit) and you can use one **.mif** to reconfigure multiple channels. You can store these **.mifs** in on-chip or off-chip memory.

Quartus II Settings for .mif Generation

The **.mif** is not generated by default in a Quartus II compilation. This following are the Quartus II software settings you must enable to generate a **.mif** file:

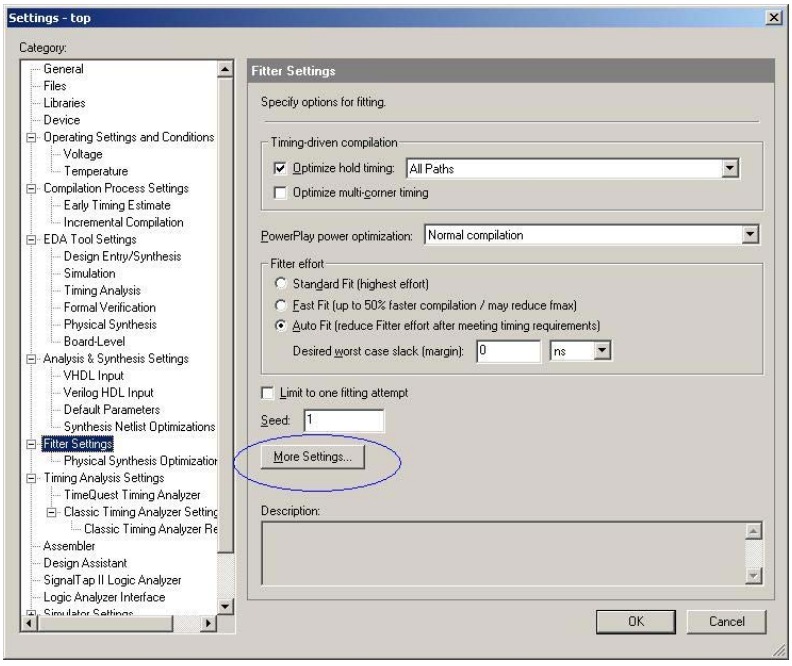
1. On the Assignments menu, select **Settings** (Figure 2–32).

Figure 2–32. Step 1 to Enable .mif Generation



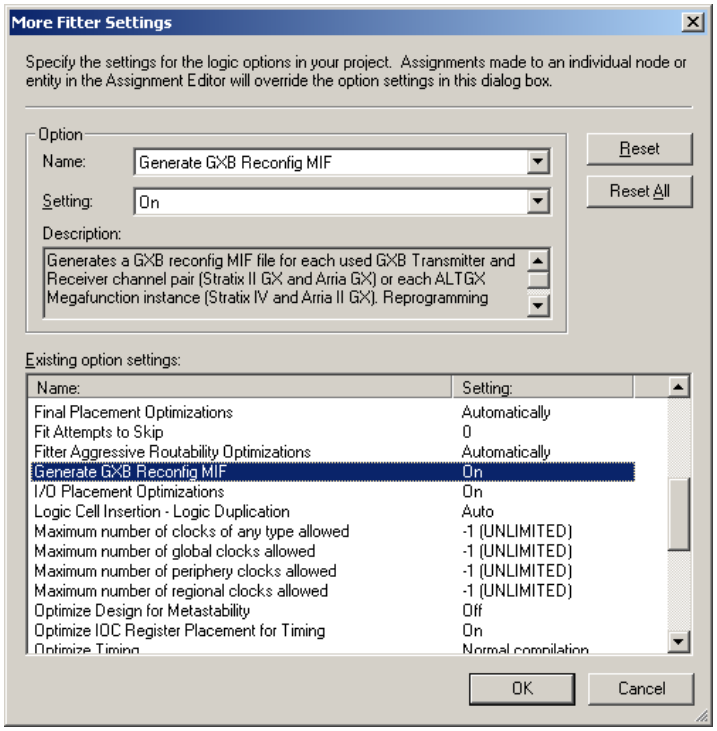
2. Select **Fitter** settings, then choose **More Settings** (Figure 2–33).

Figure 2–33. Step 2 to Enable .mif Generation



3. In the **Option** box of the **More Fitter Settings** page, set the **Generate GXB Reconfig MIF** option to **On** based on the dynamic reconfiguration mode enabled (Figure 2–34).

Figure 2–34. Step 3 to Enable .mif Generation



The **.mif** is generated in the Assembler stage of the compilation process. However, for any change in the design or the above settings, the Quartus II software runs through the fitter stage before starting the assembler stage.

Reusing .MIFs

To configure the transceiver PLLs and receiver PLLs for multiple data rates, it is important to understand the input reference clock requirements. This helps you to efficiently create the clocking scheme for reconfiguration and to reuse the **.mifs** across all channels in the device. This section reviews the new clocking enhancements and the implications of using input clocks from various clock sources.

The available clock inputs appear as a `pll_inclk_rx_cruclk[]` port and can be provided from the inter-transceiver block lines (also known as Inter Quad [IQ] lines), from the global clock networks that are driven by an input pin, or by a PLL cascade clock.

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

The following section describes the clocking requirements to reuse **.mifs**.

Input Reference Clock Requirements for Reusing .mifs

The **.mif** contains information about the input clock multiplexer settings and the functional blocks that you selected during the ALTGX MegaWizard Plug-In Manager instantiation. When you enable the Quartus II settings described in the section, the Quartus II software generates a **.mif** for each channel. You can use this **.mif** to dynamically reconfigure any of the other transceiver channels in the device if you satisfy the following two requirements for the input reference clocks:

- The order of the clock inputs must be consistent. For example, assume that a **.mif** is generated for a transceiver channel in transceiver block 0 and the input clock source is connected to the `pll_inclk_rx_cruclk [0]` port. When the generated **.mif** is used for a channel in other transceiver blocks (for example, transceiver block 1), the same clock source needs to be connected to the `pll_inclk_rx_cruclk [0]` port. Figure 2–35 and Figure 2–36 show the incorrect and correct order of input reference clocks, respectively.
- In Figure 2–35, the clocking is incorrect when reusing the **.mif** because the input reference clock is not connected to the corresponding `pll_inclk_rx_cruclk []` ports in the two instances.

Figure 2–35 shows the incorrect input reference clock connections when reusing a **.mif**.

Figure 2–35. Incorrect Input Reference Clock Connections When Reusing a .mif

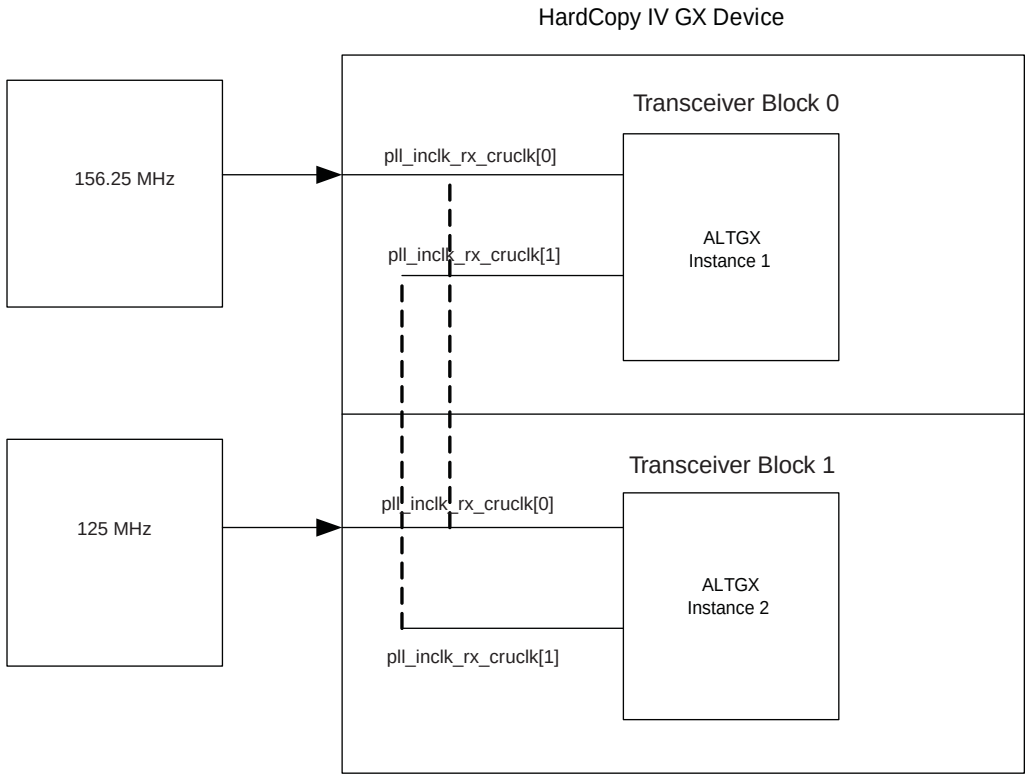
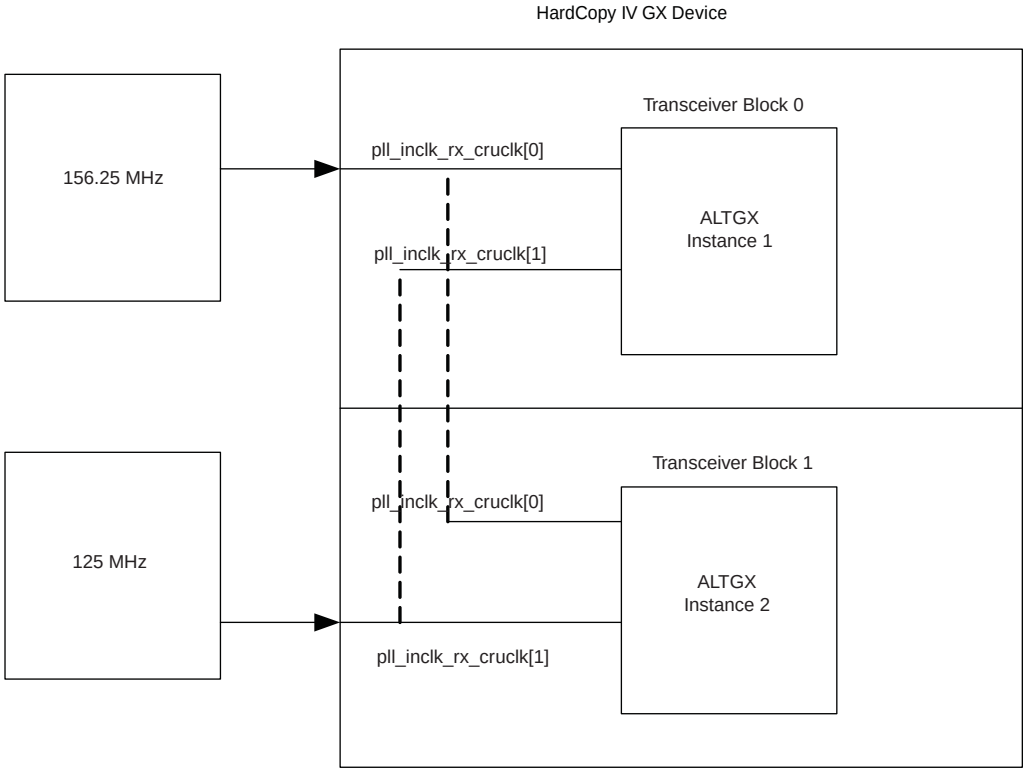


Figure 2–36 shows the correct input reference clock connections when reusing a **.mif**.

Figure 2–36. Correct Input Reference Clock Connections When Reusing a .mif



You can reuse the **.mif** generated for a transceiver channel on one side of the device, for a transceiver channel on the other side of device, only if the input reference clock frequencies and order of the `pll_inclk_rx_cruclk []` ports in the ALTGX instances on both sides are identical.

In addition to the input reference clock requirements when reusing a **.mif**, refer to "The logical_tx_pll_sel and logical_tx_pll_sel_en Ports" on page 2–109 for additional ways to reuse a .**mif**.

The logical_channel_address Port in .mif-Based Dynamic Reconfiguration Modes

Based on the value you set at the `logical_channel_address [8:0]` port, the dynamic reconfiguration controller writes the **.mif** contents into the transceiver channel you specify. This signal is enabled only when the number of channels controlled by the dynamic reconfiguration controller is more than one. Because transceiver channel reconfiguration is done on a per-channel basis, you must use this signal and provide the necessary logical channel address to write the **.mif** words successfully into the channel. For more information about the `logical_channel_address` port, refer to "Dynamic Reconfiguration Controller Port List" on page 2–11.

In CMU PLL reconfiguration mode, use `logical_channel_address` to specify the transceiver channel that the CMU PLL (under reconfiguration) is connected to.

Channel and CMU PLL Reconfiguration Mode

You can reconfigure a transceiver channel to a different functional mode and data rate by using this dynamic reconfiguration mode. To reconfigure a channel successfully, select the appropriate options in the ALTGX MegaWizard Plug-In Manager (described in the following sections), and generate a **.mif**. The ALTGX_RECONFIG instance reconfigures the transceiver channel by writing the **.mif** contents into the channel.

☞ Channel and CMU PLL Reconfiguration mode only affects the channel involved in the reconfiguration (the transceiver channel specified by the `logical_channel_address` port), without affecting the remaining transceiver channels controlled by the dynamic reconfiguration controller.

☞ Channel and CMU PLL Reconfiguration mode is applicable to the regular transceiver channels configured in non Basic (PMA Direct) mode only.

☞ Channel and CMU PLL Reconfiguration mode is not applicable to the following:

- Regular transceiver channels configured in Basic (PMA Direct) ×1 and ×N configurations
- CMU channels configured in Basic (PMA Direct) ×1 and ×N configurations
- Regular transceiver channels configured in bonded configurations
- Regular transceiver channels configured in CEI configuration

Channel and CMU PLL Reconfiguration mode can be classified as follows:

- Data rate reconfiguration
- Functional mode reconfiguration
- Data rate and functional mode reconfiguration

Data Rate Reconfiguration

- You can reconfigure the data rate of the transceiver channel by reconfiguring the CMU PLL connected to the transceiver channel.
- You can reconfigure the data rate of the transceiver channel by selecting another transmitter PLL to supply clocks to the transceiver channel.
- Every transmitter channel has one local clock divider. Similarly, every receiver channel has one local clock divider. You can reconfigure the data rate of a transceiver channel by reconfiguring these local clock dividers to 1, 2, or 4. When you reconfigure these local clock dividers, ensure that the functional mode of the transceiver channel supports the reconfigured data rate.

Functional Mode Reconfiguration

You can reconfigure the existing functional mode of the transceiver channel to a totally different functional mode using this feature. The following are the various ways you can reconfigure the existing functional mode:

- You can switch only between one non-Basic (PMA Direct) configuration to another non-Basic (PMA Direct) configuration
- You can switch between one protocol configuration (for example, XAUI configuration) to another protocol configuration (for example, GIGE configuration)
- You can switch between one protocol configuration (for example, Serial RapidIO configuration) to a Basic configuration
- You can switch between Basic configuration to another Basic functional mode

You cannot dynamically reconfigure the following:

- You cannot switch from a CEI configuration to any other functional mode
- You cannot switch between Basic (PMA Direct) ×1 configuration to Basic (PMA Direct) ×1 configuration
- You cannot switch between Basic (PMA Direct) ×N configuration to Basic (PMA Direct) ×N configuration
- You cannot switch from a bonded mode configuration (for example: PCI Express [PIPE] ×4 configuration) to any other functional mode

There is no limit to the number of functional modes you can reconfigure the transceiver channel to, if the various clocks involved support the transition. For more information about core clocks, refer to "Core Clocking Setup" on page 2–79.

In addition to the categories mentioned, you can also choose to reconfigure both the data rate and functional mode of a transceiver channel.

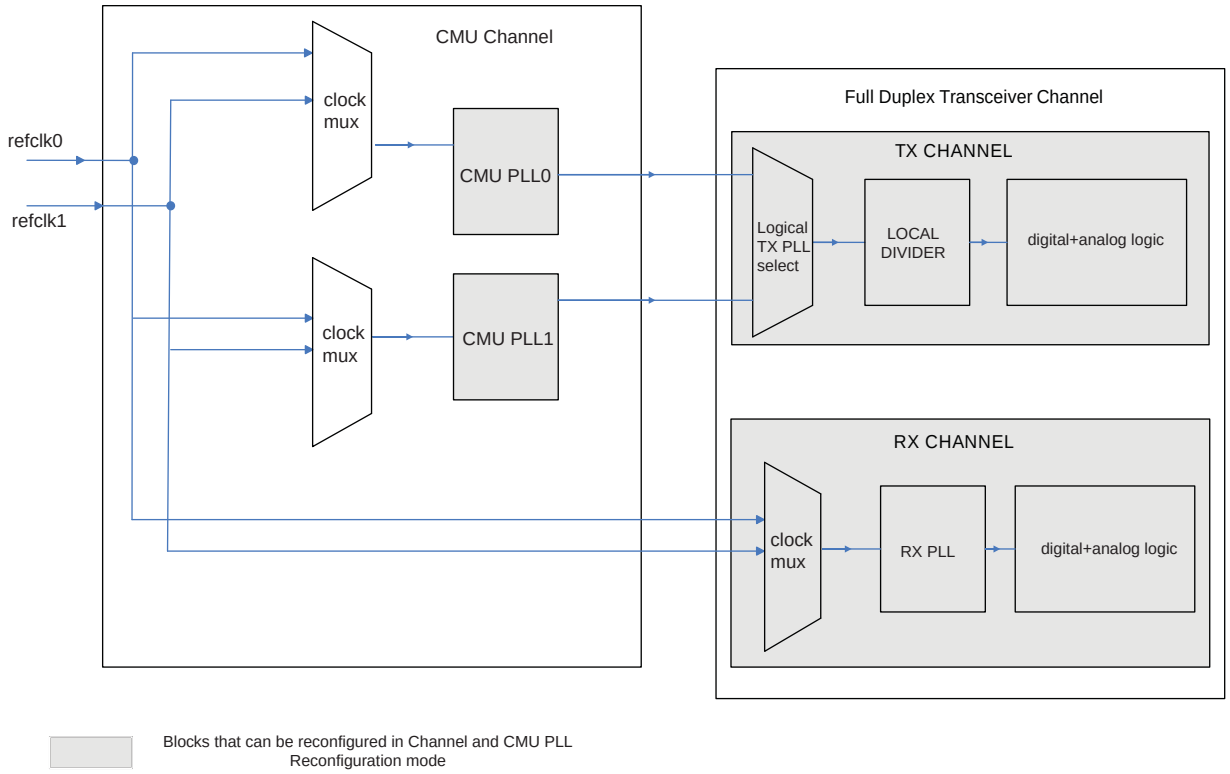
For the following sections, assume that the transceiver channel has the **Receiver and Transmitter** configuration in the ALTGX MegaWizard Plug-In Manager, unless specified as **Transmitter only** or **Receiver only**.

Blocks Reconfigured in Channel and CMU PLL Reconfiguration Mode

The blocks that get reconfigured by this dynamic reconfiguration mode are the PCS and PMA blocks of a transceiver channel, local divider settings of the transmitter and receiver channel, and the CMU PLL.

Figure 2–37 shows the functional blocks that you can dynamically reconfigure using this feature.

Figure 2–37. Channel and CMU PLL Reconfiguration in a Transceiver Block



Channel Reconfiguration Supported Configurations

Channel reconfiguration is applicable only to the following non-bonded configurations of a physical transceiver channel:

- **Receiver and Transmitter** configuration
- **Transmitter only** configuration
- **Receiver only** configuration
- Independent Transmitter and Independent Receiver combined into the same physical channel

In **Transmitter only** configuration, the physical transceiver channel has only one transmitter. The **.mif** for the **Transmitter only** file has the bits for the transmitter portion only. The **Receiver only** configuration is the same as the **Transmitter only** configuration except it pertains to the receiver.

Channel reconfiguration from a **Transmitter only** configuration to a **Receiver only** configuration and vice versa is not allowed.

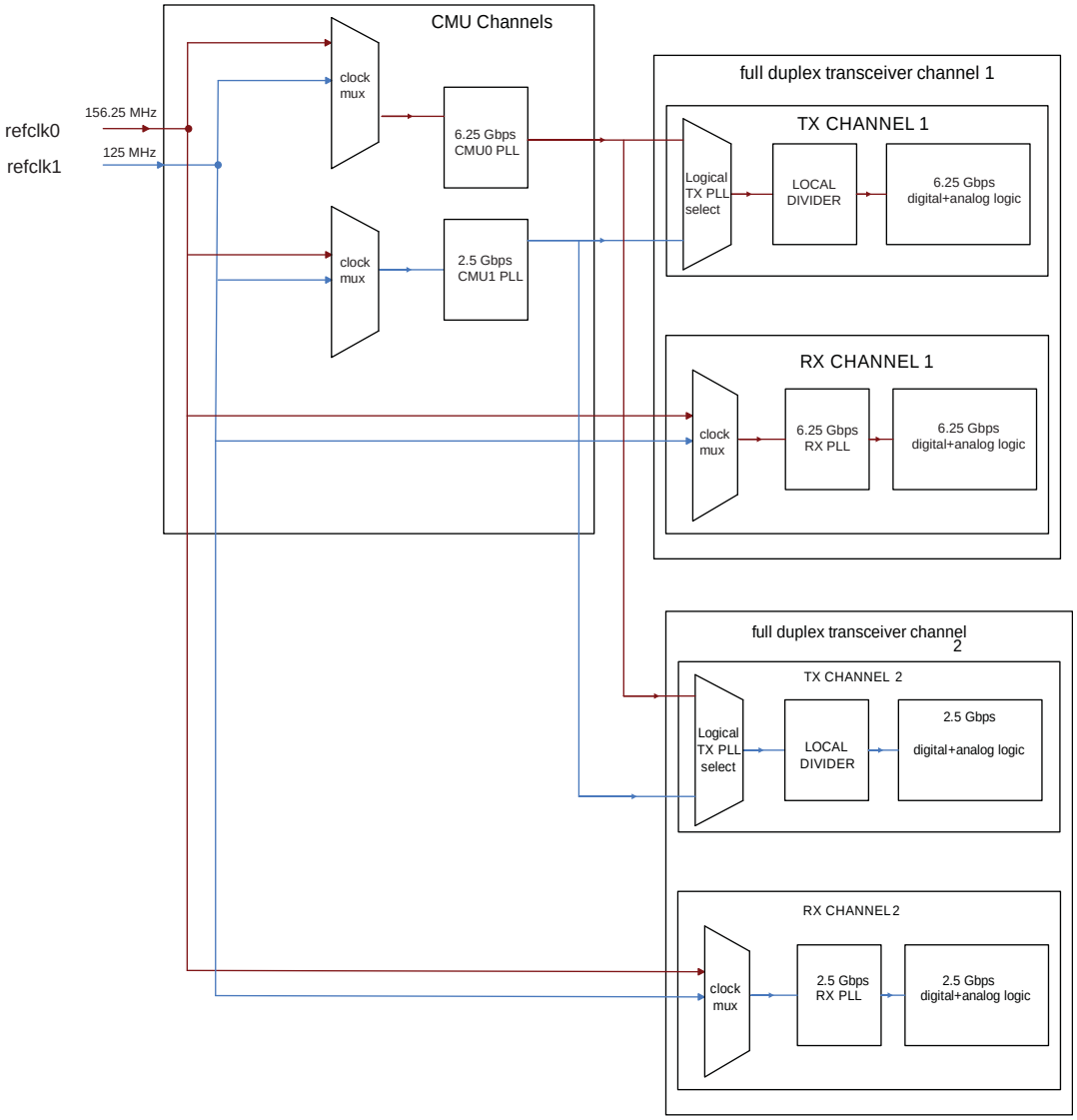
ALTGX MegaWizard Plug-In Manager Setup for Channel and CMU PLL Reconfiguration Mode

To reconfigure the transceiver channel and CMU PLL, set up the ALTGX MegaWizard Plug-In Manager as shown in the following steps. The dynamic reconfiguration controller reconfigures the transceiver channel and the CMU PLL with the new information stored in the **.mif**.

1. Select the **Channel and Transmitter PLL reconfiguration** option in the **Reconfig** screen.
2. If you want to reconfigure the data rate of the transceiver channel by reconfiguring the CMU PLL, provide the new data rate you want the CMU PLL to run at in the **General** screen.
3. Provide the logical reference index value in the **What is the main PLL logical reference index?** option in the **Reconfig Clks** screen.
   - Selecting the Logical Reference Index of the CMU PLL

   In Figure 2–38, transceiver channel 1 listens to CMU0 PLL of the transceiver block. Similarly, transceiver channel 2 listens to CMU1 PLL of the transceiver block.

Figure 2–38. Logical Reference Index of CMU PLLs in a Transceiver Block



You can direct the ALTGX_RECONFIG instance to dynamically reconfigure CMU0 PLL by specifying its logical reference index (the identity of a CMU PLL). Similarly, you can direct the ALTGX_RECONFIG instance to dynamically reconfigure CMU1 PLL instead by providing the logical reference index of CMU1 PLL. The allowed values for the logical reference index are 0 or 1. Similarly, the CMU PLLs in all the transceiver blocks can be assigned a logical reference index value of 0 or 1.

The logical reference index of CMU0 PLL within a transceiver block is always the complement of the logical reference index of CMU1 PLL within the same transceiver block.

This logical reference index value is stored as logical tx pll, along with the other transceiver channel settings in the **.mif**.

4. Provide the number of input reference clocks available for the CMU PLL in the **How many input clocks?** option in the **Reconfig Clks** screen. The maximum number of input reference clocks allowed is 10. For more information about this setting, refer to "General Guidelines for Specifying the Input Reference Clocks" on page 2–115.
5. Provide the identification of the input reference clock used by the CMU PLL in the **What is the selected input clock source for the Transmitter PLL and Receiver PLL?** option in the **Reconfig Clks** screen.
6. Set up transceiver and core clocking, which is explained in the following section.
   - Transceiver Clocking Setup

     You must set up the transceiver clocking options as part of channel reconfiguration for functional mode switch over or data rate transition. Transceiver clocking covers all the clock options you need to set up.

     Two CMU PLLs for data rates and functional modes

     Input reference clocks for transmit and receive
   - Core Clocking Setup

     The transceiver core clocks are the write and read clocks of the Transmit Phase Compensation FIFO and the Receive Phase Compensation FIFO, respectively. Core clocking is classified as transmitter core clocking and receiver core clocking. Table 2–26 explains transmitter core clocking. Similarly, Table 2–27 explains receiver core clocking.

Table 2–26. Transmitter Core Clocking (Part 1 of 2)

Transmitter core clocking refers to the clock that is used to write the parallel data from the core fabric into the Transmit Phase Compensation FIFO. You can use one of the following clocks to write into the Transmit Phase Compensation FIFO: ■ `tx_coreclk`—You can use a clock of the same frequency as `tx_clkout` from the core fabric to provide the write clock to the Transmit Phase Compensation FIFO. If you use `tx_coreclk`, it overrides the `tx_clkout` options in the ALTGX MegaWizard Plug-In Manager. ■ `tx_clkout`—The Quartus II software automatically routes `tx_clkout` to the core fabric and back into the Transmit Phase Compensation FIFO. There are two options available within the `tx_clkout` option in the **Reconfig 2** screen, as shown in Figure 2–39.	
The following are the two `tx_clkout` options in the **Reconfig 2** screen of the ALTGX MegaWizard Plug-In Manager *(1)*:	
Option 1: Share a Single Transmitter Clock between Transmitters	**Option 2: Use the Respective Channel Transmitter Core Clocks**

Table 2–26. Transmitter Core Clocking (Part 2 of 2)

■ Enable this option if you want `tx_clkout` of the first channel (channel 0) of the transceiver block to provide the write clock to the Transmitter Phase Compensation FIFOs of the remaining channels in the transceiver block. ■ This option is typically enabled when all the channels of a transceiver block are of the same functional mode and data rate, and are reconfigured to the identical functional mode and data rate. ■ Consider the following scenario: → Four regular transceiver channels configured at 3 Gbps and in the same functional mode. → Channel and CMU PLL Reconfiguration mode is enabled in the ALTGX_RECONFIG MegaWizard Plug-In Manager. → You want to reconfigure all four regular transceiver channels to 1.5 Gbps and vice versa. ■ Option 1 is applicable in this scenario because it saves clock resources. ■ Figure 2–40 shows the sharing of channel 0's `tx_clkout` between all four regular channels of a transceiver block.	■ Enable this option if you want the individual transmitter channel `tx_clkout` signals to provide the write clock to their respective Transmit Phase Compensation FIFOs. ■ This option is typically enabled when each transceiver channel is reconfigured to a different functional mode using channel reconfiguration. ■ Consider the following scenario: → Four regular transceiver channels configured at 3 Gbps and different functional modes. → Channel and CMU PLL Reconfiguration mode is enabled in the ALTGX_RECONFIG MegaWizard Plug-In Manager. → You want to reconfigure each of the four regular transceiver channels to different data rates and different functional modes. ■ Option 2 is applicable in this scenario because the design requires all four regular transceiver channels to be reconfigured to different data rates and functional modes. ■ Figure 2–41 shows how each transmitter channel's `tx_clkout` signal provides a clock to the Transmit Phase Compensation FIFOs of the respective transceiver channels. ■ Therefore, each channel can be reconfigured to a different functional mode using the Channel and CMU PLL Reconfiguration mode.

Note to Table 2–26:

(1) The **Reconfig 2** screen is not available for PMA-only channels (channels configured in Basic [PMA Direct] ×1 and ×N modes).

Figure 2–39. Two Options within the tx_clkout Option in the Reconfig 2 Screen of the ALTGX MegaWizard Plug-In Manager

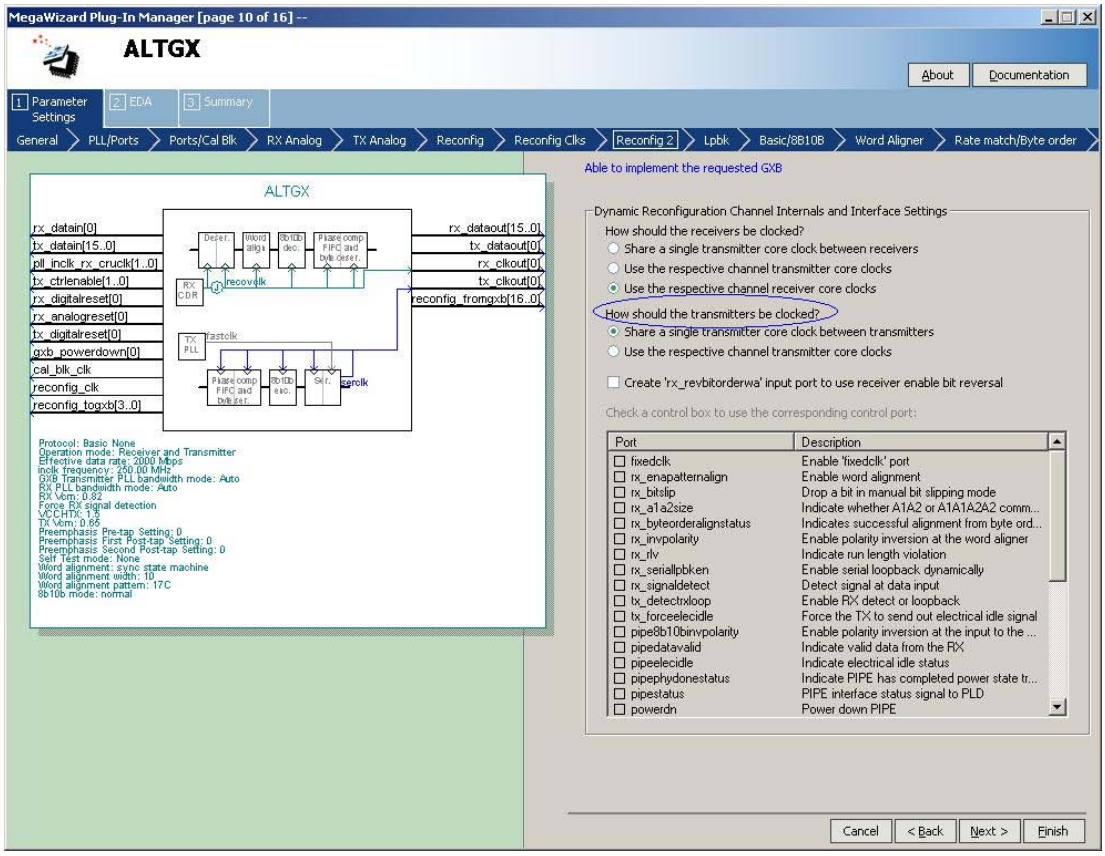


Figure 2–40. Option 1 for Transmitter Core Clocking (Channel and CMU PLL Reconfiguration Mode)

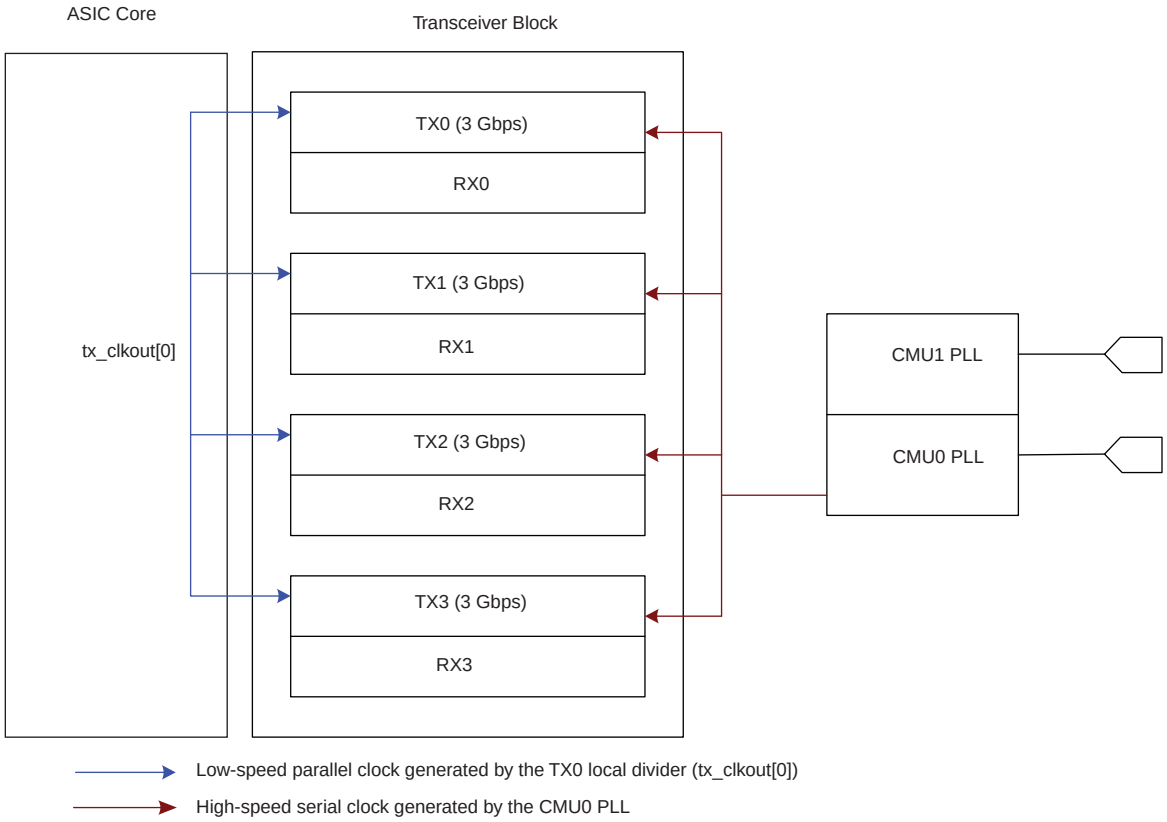


Figure 2–41. Option 2 for Transmitter Core Clocking (Channel and CMU PLL Reconfiguration Mode)

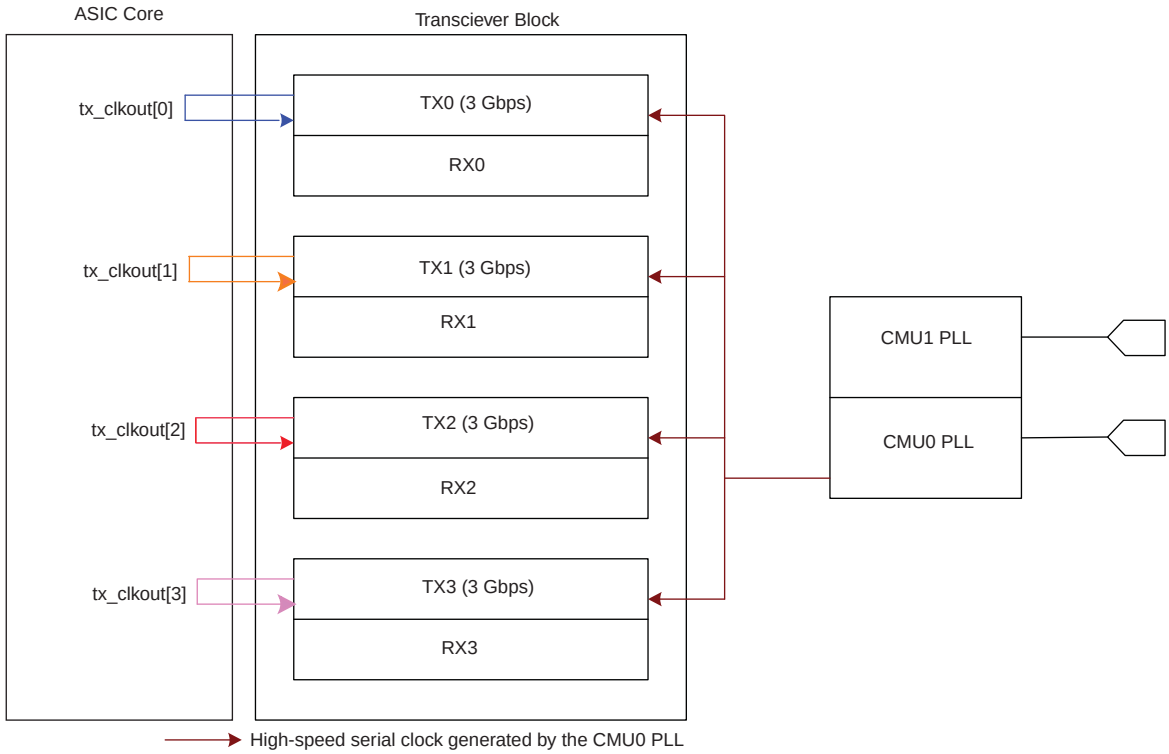


Table 2–27. Receiver Core Clocking (Part 1 of 2)

Receiver core clocking refers to the clock that is used to read the parallel data from the Receiver Phase Compensation FIFO into the core fabric. You can use one of the following clocks to read from the Receive Phase Compensation FIFO: ■ `rx_coreclk`—You can use a clock of the same frequency as `rx_clkout` from the core fabric to provide the read clock to the Receive Phase Compensation FIFO. If you use `rx_coreclk`, it overrides the `rx_clkout` options in the ALTGX MegaWizard Plug-In Manager. ■ `rx_clkout`—The Quartus II software automatically routes `rx_clkout` to the core fabric and back into the Receive Phase Compensation FIFO. There are three options available within the `rx_clkout` option in the **Reconfig 2** screen.		
The following are the three `rx_clkout` options in the **Reconfig 2** screen of the ALTGX MegaWizard Plug-In Manager *(1)*:		
Option 1: Share a Single Transmitter Core Clock between Receivers	**Option 2: Use Respective Channel Transmitter Core Clocks**	**Option 3: Use Respective Channel Receiver Core Clocks**

Table 2–27. Receiver Core Clocking (Part 2 of 2)

■ Enable this option if you want `tx_clkout` of the first channel (channel 0) of the transceiver block to provide the read clock to the Receive Phase Compensation FIFOs of the remaining receiver channels in the transceiver block. ■ This option is typically enabled when all the channels of a transceiver block are in a Basic or Protocol configuration with Rate Matching enabled and are reconfigured to another Basic or Protocol configuration with Rate Matching enabled. ■ Consider the following scenario: → Four regular transceiver channels configured to Basic 2 Gbps functional mode with Rate Matching enabled. → Channel and CMU PLL Reconfiguration mode is enabled in the ALTGX_RECONFIG MegaWizard Plug-In Manager. → You want to reconfigure all four regular transceiver channels to 3.125 Gbps configuration with Rate Matching enabled. ■ Option 1 is applicable in this scenario. ■ Figure 2–42 shows the sharing of channel 0's `tx_clkout` between all four channels of a transceiver block.	■ Enable this option if you want the individual transmitter channel's `tx_clkout` signal to provide the read clock to its respective Receive Phase Compensation FIFO. ■ This option is typically enabled when all the transceiver channels have Rate Matching enabled with different data rates and are reconfigured to another Basic or Protocol functional mode with Rate Matching enabled. ■ Consider the following scenario: → TX1/RX1: You want to dynamically reconfigure Basic 1 Gbps configuration with Rate Matching enabled to Basic 2 Gbps configuration with Rate Matching enabled. → TX3/RX3: You want to dynamically reconfigure Basic 4 Gbps configuration with Rate Matching enabled to Basic 1 Gbps configuration with Rate Matching enabled. → TX0/RX0: You want to dynamically reconfigure Basic 3.125 Gbps configuration with Rate Matching enabled to 1 Gbps configuration with Rate Matching and vice versa. → Channel and CMU PLL Reconfiguration mode is enabled in the ALTGX_RECONFIG MegaWizard Plug-In Manager. ■ Option 2 is applicable in this scenario because the design requires the individual transceiver channels to be reconfigured with different data rates to another Basic or Protocol functional mode with Rate Matching. Therefore, each channel can be reconfigured to another Basic or Protocol functional mode with Rate Matching enabled and a different data rate. ■ Figure 2–43 shows the respective `tx_clkout` of each channel clocking the respective channels of a transceiver block.	■ Enable this option if you want the individual channel's `rx_clkout` signal to provide the read clock to its respective Receive Phase Compensation FIFO. ■ This option is typically enabled when the channel is reconfigured from a Basic or Protocol configuration with or without Rate Matching to another Basic or Protocol configuration with or without Rate Matching. ■ Consider the following scenario: → TX1/RX1: GIGE configuration to SONET/SDH OC48 configuration. → TX2/RX2: Basic 2.5 Gbps configuration with Rate Matching disabled to Basic 1.244 Gbps configuration with Rate Matching disabled. → Channel and CMU PLL Reconfiguration mode is enabled in the ALTGX_RECONFIG MegaWizard Plug-In Manager ■ Option 3 is applicable in this scenario. ■ Figure 2–44 shows the respective `rx_clkout` of each channel, clocking the respective receiver channels of a transceiver block.

Note to Table 2–27:

(1) The **Reconfig 2** screen is not available for PMA-only channels (channels configured in Basic [PMA Direct] ×1 and ×N modes).

Figure 2–42. Receiver Core Clocking for Option 1

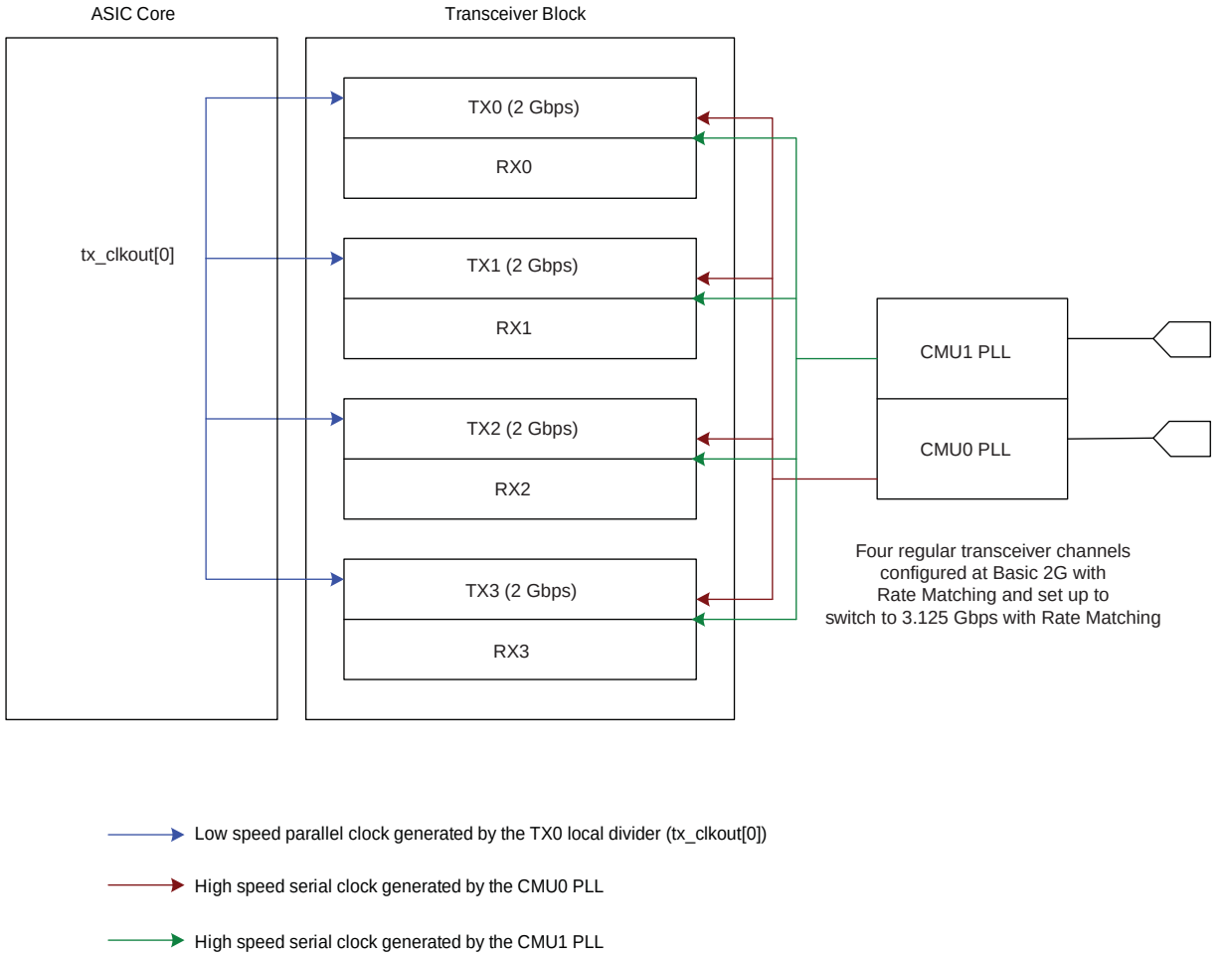


Figure 2–43. Receiver Core Clocking for Option 2

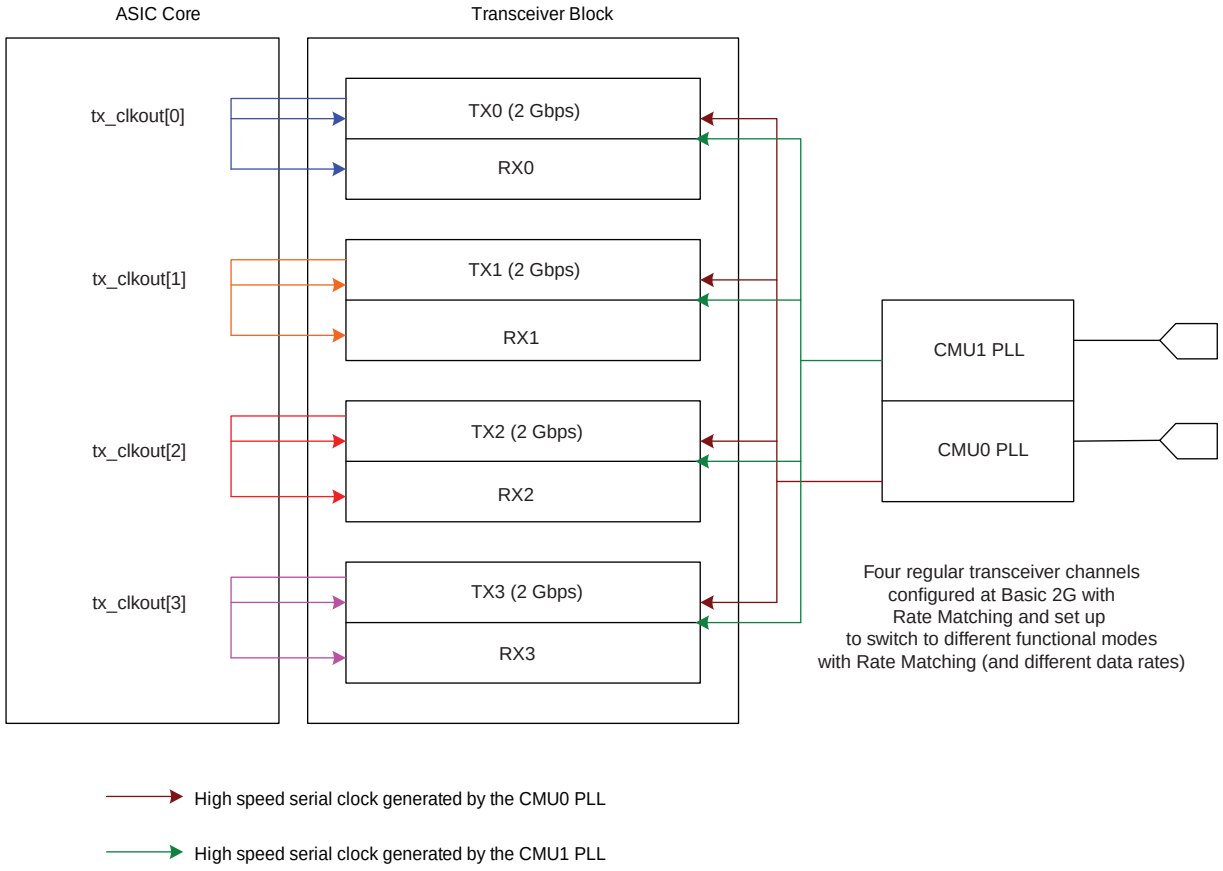


Figure 2–44. Receiver Core Clocking for Option 3

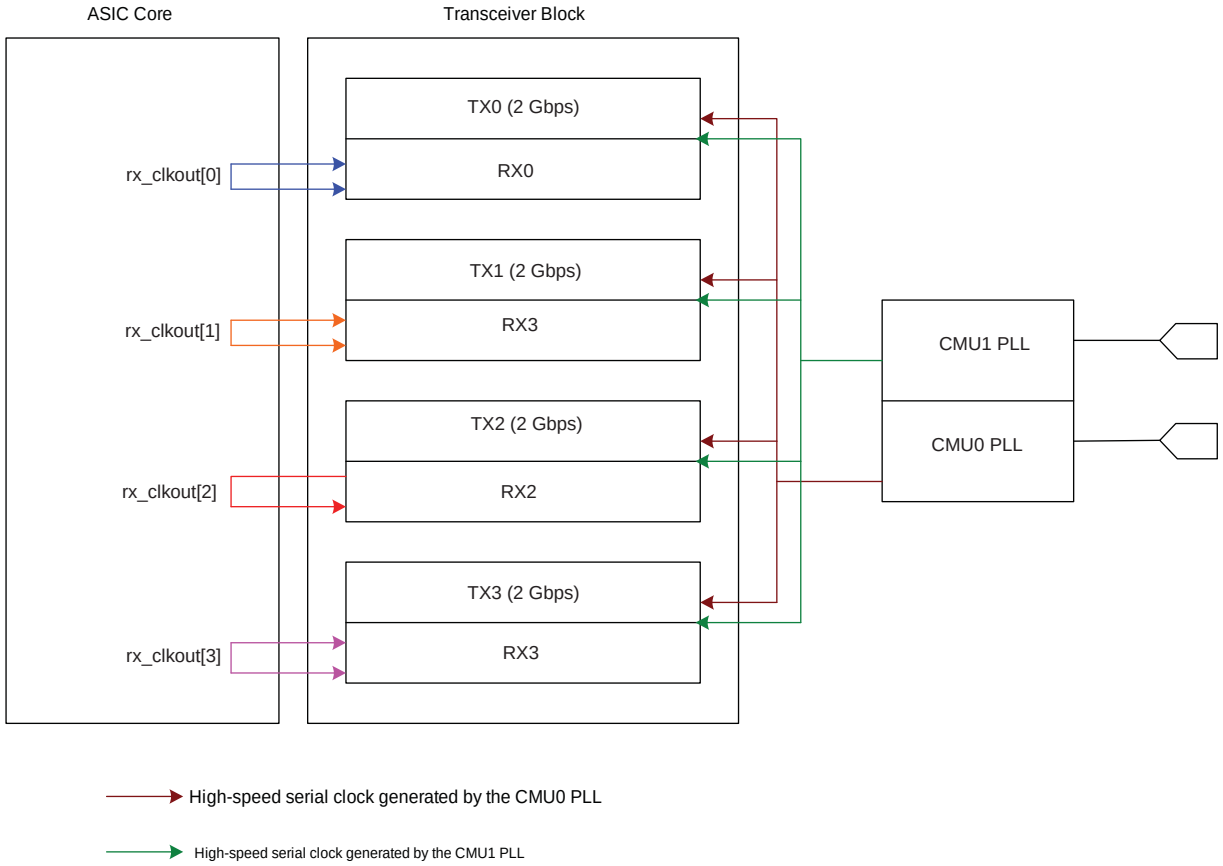
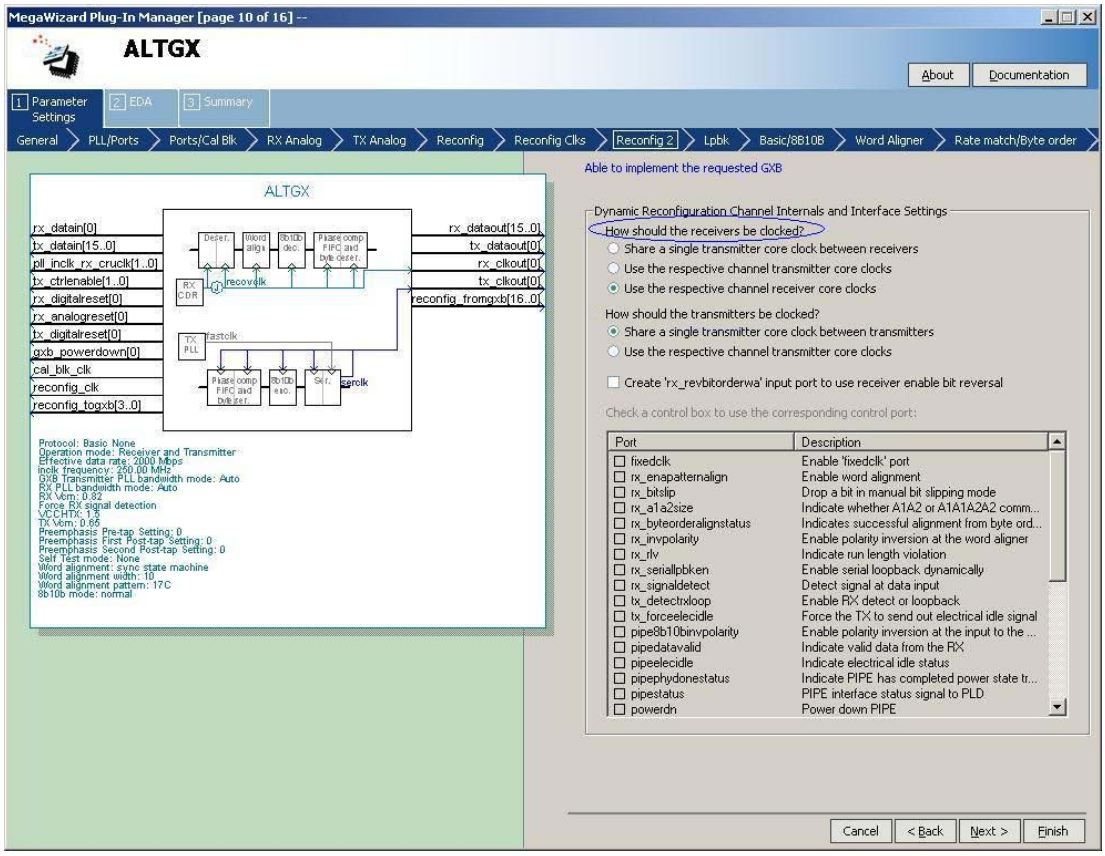


Figure 2–45. rx_clkout Options in the ALTGX MegaWizard Plug-In Manager



7. Core Fabric-Transceiver Channel Interface Selection

This section describes the ALTGX MegaWizard Plug-In Manager settings related to the core fabric-Transceiver channel interface data width when you select and activate Channel and CMU PLL Reconfiguration mode. You need to set up the core fabric-Transceiver channel interface data width when channel reconfiguration involves the following:

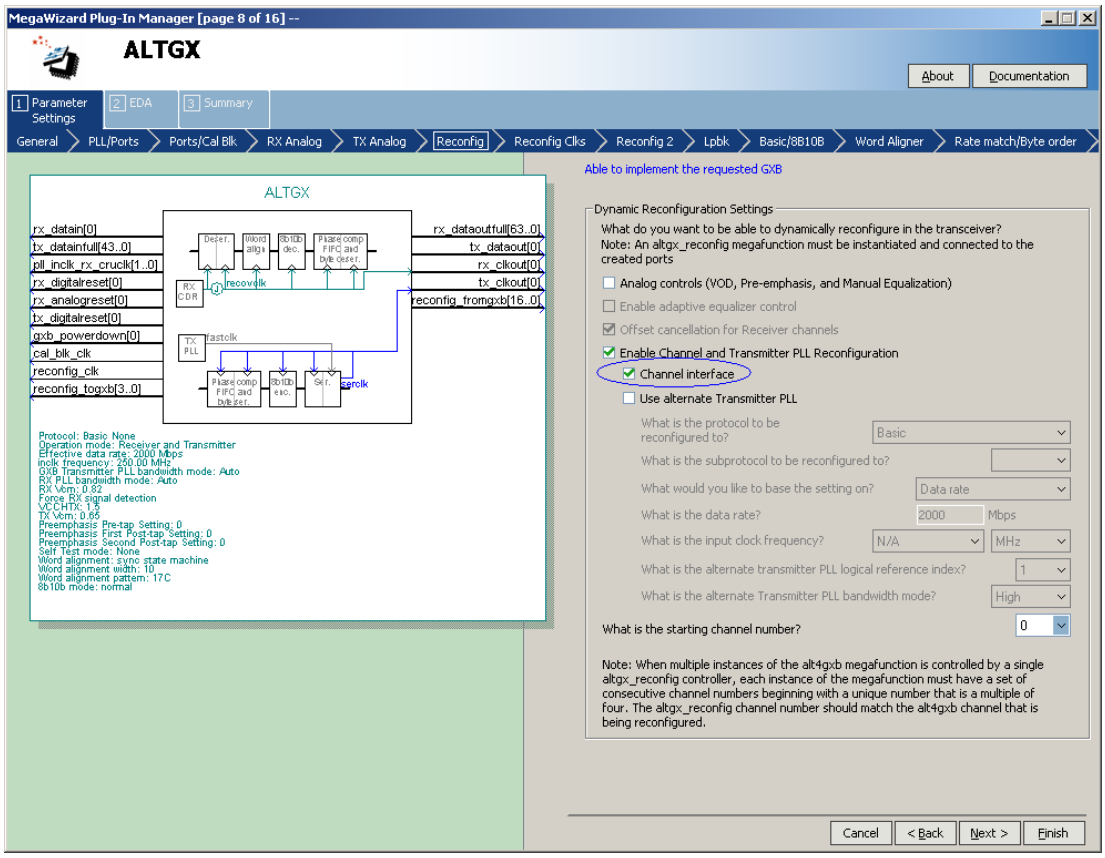
- Functional mode reconfiguration involving changes in the core fabric-Transceiver channel data width
- Functional mode reconfiguration involving enabling and disabling the static PCS blocks of the transceiver channel

You can set up the core fabric-Transceiver channel interface data width by enabling the **Channel Interface** option in the **Reconfig** screen, as shown in Figure 2–46.

Enable the **Channel Interface** option if the reconfiguration of the transceiver channel involves the following changes:

- The reconfigured channel has a changed core fabric-Transceiver channel interface data width
- The reconfigured channel has changed input control signals and output status signals

Figure 2–46. Channel Interface Option in the ALTGX MegaWizard Plug-In Manager



There are two new signals available when you enable this option:

- `tx_datainfull`—The width of this input signal depends on the number of channels you set up in the **General** screen. It is 44 bits wide per channel. This signal is available only for **Transmitter only** and **Receiver and Transmitter** configurations. This port replaces the existing `tx_datain` port.
- `rx_dataoutfull`—The width of this output signal depends on the number of channels you set up in the **General** screen. It is 64 bits wide per channel. This signal is available only for **Receiver only** and **Receiver and Transmitter** configurations. This port replaces the existing `rx_dataout` port.

In addition to these two new ports, you can select the necessary control and status signals for the reconfigured channel in the **Reconfig 2** screen.

For more information about control and status signals, refer to the Transceiver Port List in the *HardCopy IV GX Transceiver Architecture* chapter in volume 3 of the *HardCopy IV Device Handbook*.

These control and status signals are not applicable in Basic (PMA Direct) functional mode. The following signals are not available when you enable this option.

Core fabric-Receiver interface:

- `rx_dataout`
- `rx_syncstatus`
- `rx_patterndetect`
- `rx_a1a2sizeout`
- `rx_ctrldetect`
- `rx_errdetect`
- `rx_disperr`

Core fabric-Transmitter interface:

- `tx_datain`
- `tx_ctrlenable`
- `tx_forcedisp`
- `tx_dispval`

The Quartus II software has legal checks for the connectivity of `tx_datainfull` and `rx_dataoutfull` and the various control and status signals you enable in the **Reconfig 2** screen.

For example, the Quartus II software allows you to select and connect the `pipestatus` and `powerdn` signals. It assumes that you are planning to switch to and from PCI Express (PIPE) functional mode. Table 2–28 describes the `tx_datainfull[43:0]` core fabric-Transceiver channel interface signals.

Table 2–28. tx_datainfull[43:0] Core fabric-Transceiver Channel Interface Signal Descriptions (Part 1 of 3)

Core fabric-Transceiver Channel Interface Description	Transmit Signal Description (Based on HardCopy IV GX Supported Core fabric-Transceiver Channel Interface Widths)
8-bit Core fabric-Transceiver Channel Interface	`tx_datainfull[7:0]`: 8-bit data (`tx_datain`)
	The following signals are used only in 8B/10B modes:
	`tx_datainfull[8]`: Control bit (`tx_ctrlenable`)
	`tx_datainfull[9]`: Force disparity enable for `tx_datainfull[7:0]` (non PIPE mode). Transmitter force disparity Compliance (PIPE) (`tx_forcedisp`) in all modes except PIPE. For PIPE mode, (`tx_forcedispcompliance`) is used.
	`tx_datainfull[10]`: Forced disparity value for `tx_datainfull[7:0]` (`tx_dispval`)
10-bit Core fabric-Transceiver Channel Interface	`tx_datainfull[9:0]`: 10-bit data (`tx_datain`)
16-bit Core fabric-Transceiver Channel Interface with PCS-PMA set to 16/20 bits	Two 8-bit Data (`tx_datain`) `tx_datainfull[7:0]` - `tx_datain` (LSByte) and `tx_datainfull[18:11]` - `tx_datain` (MSByte)
	The following signals are used only in 8B/10B modes:
	`tx_datainfull[8]` - `tx_ctrlenable` (LSB) and `tx_datainfull[19]` - `tx_ctrlenable` (MSB)
	Force Disparity Enable `tx_datainfull[9]` - `tx_forcedisp` (LSB) and `tx_datainfull[20]` - `tx_forcedisp` (MSB)
	Force Disparity Value `tx_datainfull[10]` - `tx_dispval` (LSB) and `tx_datainfull[21]` - `tx_dispval` (MSB)

Table 2–28. tx_datainfull[43:0] Core fabric-Transceiver Channel Interface Signal Descriptions (Part 2 of 3)

Core fabric-Transceiver Channel Interface Description	Transmit Signal Description (Based on HardCopy IV GX Supported Core fabric-Transceiver Channel Interface Widths)
16-bit Core fabric-Transceiver Channel Interface with PCS-PMA set to 8/10 bits	Two 8-bit Data (`tx_datain`) `tx_datainfull[7:0]` - `tx_datain` (LSByte) and `tx_datainfull[29:22]` - `tx_datain` (MSByte)
	The following signals are used only in 8B/10B modes:
	Two Control Bits (`tx_ctrlenable`) `tx_datainfull`[8] - `tx_ctrlenable` (LSB) and `tx_datainfull`[30] - `tx_ctrlenable` (MSB)
	Force Disparity Enable For non-PIPE: `tx_datainfull[9]` - `tx_forcedisp` (LSB) and `tx_datainfull[31]` - `tx_forcedisp` (MSB) For PIPE: `tx_datainfull[9]` - `tx_forcedispcompliance` (LSB) and `tx_datainfull[31]` - `tx_forcedispcompliance` (MSB)
	Force Disparity Value `tx_datainfull[10]` - `tx_dispval` (LSB) and `tx_datainfull[32]` - `tx_dispval` (MSB)
20-bit Core fabric-Transceiver Channel Interface with PCS-PMA set to 20 bits	Two 10-bit Data (`tx_datain`) `tx_datainfull[9:0]` - `tx_datain` (LSByte) and `tx_datainfull[20:11]` - `tx_datain` (MSByte)
20-bit Core fabric-Transceiver Channel Interface with PCS-PMA set to 10 bits	Two 10-bit Data (`tx_datain`) `tx_datainfull[9:0]` - `tx_datain` (LSByte) and `tx_datainfull[31:22]` - `tx_datain` (MSByte)

Table 2–28. tx_datainfull[43:0] Core fabric-Transceiver Channel Interface Signal Descriptions (Part 3 of 3)

Core fabric-Transceiver Channel Interface Description	Transmit Signal Description (Based on HardCopy IV GX Supported Core fabric-Transceiver Channel Interface Widths)
32-bit Core fabric-Transceiver Channel Interface with PCS-PMA set to 16/20 bits	Four 8-bit Data (tx_datain) tx_datainfull[7:0]- tx_datain (LSByte) and tx_datainfull[18:11] tx_datainfull[29:22] tx_datainfull[40:33] - tx_datain (MSByte)
	The following signals are used only in 8B/10B modes:
	Four Control Bits (tx_ctrlenable) tx_datainfull[8] - tx_ctrlenable (LSB) and tx_datainfull[19] tx_datainfull[30] tx_datainfull[41]- tx_ctrlenable (MSB)
	Force Disparity Enable (tx_forcedisp) tx_datainfull[9]- tx_forcedisp (LSB) and tx_datainfull[20] tx_datainfull[31] tx_datainfull[42]- tx_forcedisp (MSB)
	Force Disparity Value (tx_dispval) tx_datainfull[10]- tx_dispval (LSB) and tx_datainfull[21] tx_datainfull[32] tx_datainfull[43]- tx_dispval (MSB)
40-bit Core fabric-Transceiver Channel Interface with PCS-PMA set to 20 bits	Four 10-bit Data (tx_datain) tx_datainfull[9:0] - tx_datain (LSByte) and tx_datainfull[20:11] tx_datainfull[31:22] tx_datainfull[42:33]- tx_datain (MSByte)

Table 2–29. rx_dataoutfull[63:0] Core fabric-Transceiver Channel Interface Signal Descriptions (Part 1 of 7)

Core fabric-Transceiver Channel Interface Description	Receive Signal Description (Based on HardCopy IV GX Supported Core fabric-Transceiver Channel Interface Widths)
8-bit Core fabric-Transceiver Channel Interface	**The following signals are used in 8-bit 8B/10B modes:**
	rx_dataoutfull[7:0]: 8-bit decoded data (rx_dataout)
	rx_dataoutfull[8]: Control bit (rx_ctrldetect)
	rx_dataoutfull[9]: Code violation status signal. It indicates error detected in rx_dataoutfull[7:0], which is replaced by invalid code-group (invalid or running disp.error) in GIGE mode. In PCI Express, when code violation occurs, the EDB character is placed on the erroneous data byte (= K30.7) (rx_errdetect)
	rx_dataoutfull[10]: rx_syncstatus
	rx_dataoutfull[11]: Disparity error status signal. It indicates disparity error detected in rx_dataoutfull[7:0] (rx_disperr)
	rx_dataoutfull[12]: Pattern detect status signal (rx_patterndetect)
	rx_dataoutfull[13]: Rate Match FIFO deletion status indicator (rx_rmfifodatadeleted) in non-PIPE/PCIe modes.
	rx_dataoutfull[14]: Rate Match FIFO insertion status indicator (rx_rmfifodatainserted) in non-PIPE/PCIe modes.
	rx_dataoutfull[14:13]: PIPE/PCI-E mode: 2'b00: data OK; 2'b01: 1 SKP deletion; 2'b10: elastic buffer underflow if data is 0xFE, else 1 SKP insertion; 2b11: elastic buffer overflow (rx_pipestatus)
	rx_dataoutfull[15]: 8B/10B running disparity indicator (rx_runningdisp)
	The following signals are used in 8-bit SONET/SDH mode:
	rx_dataoutfull[7:0]: 8-bit un-encoded data (rx_dataout)
	rx_dataoutfull[8]: rx_a1a2sizeout
	rx_dataoutfull[10]: rx_syncstatus
	rx_dataoutfull[11]: Reserved
	rx_dataoutfull[12]: rx_patterndetect
10-bit Core fabric-Transceiver Channel Interface	rx_dataoutfull[9:0]: 10-bit un-encoded data (rx_dataout)
	rx_dataoutfull[10]: rx_syncstatus
	rx_dataoutfull[11]: 8B/10B disparity error indicator (rx_disperr)
	rx_dataoutfull[12]: rx_patterndetect
	rx_dataoutfull[13]: Rate Match FIFO deletion status indicator (rx_rmfifodatadeleted) in non PIPE/PCIe modes
	rx_dataoutfull[14]: Rate Match FIFO insertion status indicator (rx_rmfifodatainserted) in non PIPE/PCIe modes
	rx_dataoutfull[15]: 8B/10B running disparity indicator (rx_runningdisp)

Table 2–29. rx_dataoutfull[63:0] Core fabric-Transceiver Channel Interface Signal Descriptions (Part 2 of 7)

Core fabric-Transceiver Channel Interface Description	Receive Signal Description (Based on HardCopy IV GX Supported Core fabric-Transceiver Channel Interface Widths)
16-bit Core fabric-Transceiver Channel Interface with PCS-PMA set to 16/20 bits	Two 8-bit unencoded Data (rx_dataout) rx_dataoutfull[7:0] - rx_dataout (LSByte) and rx_dataoutfull[23:16]- rx_dataout (MSByte)
	The following signals are used in 16-bit 8B/10B modes:
	Two Control Bits rx_dataoutfull[8] - rx_ctrldetect (LSB) and rx_dataoutfull[24]- rx_ctrldetect (MSB)
	Two Receiver Error Detect Bits rx_dataoutfull[9] - rx_errdetect (LSB) and rx_dataoutfull[25]- rx_errdetect (MSB)
	Two Receiver Sync Status Bits rx_dataoutfull [10] - rx_syncstatus (LSB) and rx_dataoutfull[26] - rx_syncstatus (MSB)
	Two Receiver Disparity Error Bits rx_dataoutfull [11] - rx_disperr (LSB) and rx_dataoutfull[27] - rx_disperr (MSB)
	Two Receiver Pattern Detect Bits rx_dataoutfull[12] - rx_patterndetect (LSB) and rx_dataoutfull[28]- rx_patterndetect (MSB)
	rx_dataoutfull[13] and rx_dataoutfull[45]: Rate Match FIFO deletion status indicator (rx_rmfifodatadeleted) in non-PIPE/PCIe modes
	rx_dataoutfull[14] and rx_dataoutfull[46]: Rate Match FIFO insertion status indicator (rx_rmfifodatainserted) in non-PIPE/PCIe modes
	Two 2-bit PIPE Status Bits rx_dataoutfull[14:13] - rx_pipestatus (LSB) and rx_dataoutfull[30:29] - rx_pipestatus (MSB)
	PIPE/PCI-E mode: 2'b00: data OK 2'b01: 1 SKP deletion 2'b10: elastic buffer underflow if data is 8'hFE, else 1 SKP insertion 2'b11: elastic buffer overflow
	rx_dataoutfull[15] and rx_dataoutfull[47]: 8B/10B running disparity indicator (rx_runningdisp)
16-bit Core fabric-Transceiver Channel Interface with PCS-PMA set to 8/10 bits	Two 8-bit Data rx_dataoutfull[7:0] - rx_dataout (LSByte) and rx_dataoutfull[39:32] - rx_dataout (MSByte)

Table 2–29. rx_dataoutfull[63:0] Core fabric-Transceiver Channel Interface Signal Descriptions (Part 3 of 7)

Core fabric-Transceiver Channel Interface Description	Receive Signal Description (Based on HardCopy IV GX Supported Core fabric-Transceiver Channel Interface Widths)
16-bit Core fabric-Transceiver Channel Interface with PCS-PMA set to 8/10 bits (continued)	**The following signals are used in 16-bit 8B/10B mode:**
	Two Control Bits `rx_dataoutfull[8]` - `rx_ctrldetect` (LSB) and `rx_dataoutfull[40]` - `rx_ctrldetect` (MSB)
	Two Receiver Error Detect Bits `rx_dataoutfull[9]` - `rx_errdetect` (LSB) and `rx_dataoutfull[41]`- `rx_errdetect` (MSB)
	Two Receiver Sync Status Bits `rx_dataoutfull[10]` - `rx_syncstatus` (LSB) and `rx_dataoutfull[42]`- `rx_syncstatus` (MSB)
	Two Receiver Disparity Error Bits `rx_dataoutfull[11]` - `rx_disperr` (LSB) and `rx_dataoutfull[43]` - `rx_disperr` (MSB)
	Two Receiver Pattern Detect Bits `rx_dataoutfull[12]` - `rx_patterndetect` (LSB) and `rx_dataoutfull[44]` - `rx_patterndetect` (MSB)
	`rx_dataoutfull[13]` and `rx_dataoutfull[45]`: Rate Match FIFO deletion status indicator (`rx_rmfifodatadeleted`) in non-PIPE/PCIe modes
	`rx_dataoutfull[14]` and `rx_dataoutfull[46]`: Rate Match FIFO insertion status indicator (`rx_rmfifodatainserted`) in non-PIPE/PCIe modes
	Two 2-bit PIPE Status Bits `rx_dataoutfull[14:13]` - `rx_pipestatus` (LSB) and `rx_dataoutfull[46:45]`- `rx_pipestatus` (MSB)
	PIPE/PCI-E mode: 2'b00: data OK 2'b01: 1 SKP deletion 2'b10: elastic buffer underflow if data is 8'hFE, else 1 SKP insertion 2'b11: elastic buffer overflow (`rx_pipestatus`)
	`rx_dataoutfull`[15] and `rx_dataoutfull[47]`: 8B/10B running disparity indicator (`rx_runningdisp`)
	The following signals are used in 16-bit SONET/SDH mode:
	Two 8-bit Data `rx_dataoutfull[7:0]` - `rx_dataout` (LSByte) and `rx_dataoutfull[39:32]` - `rx_dataout` (MSByte)
	Two Receiver Alignment Pattern Length Bits `rx_dataoutfull[8]` - `rx_a1a2sizeout` (LSB) and `rx_dataoutfull[40]`- `rx_a1a2sizeout` (MSB)

Table 2–29. rx_dataoutfull[63:0] Core fabric-Transceiver Channel Interface Signal Descriptions (Part 4 of 7)

Core fabric-Transceiver Channel Interface Description	Receive Signal Description (Based on HardCopy IV GX Supported Core fabric-Transceiver Channel Interface Widths)
16-bit Core fabric-Transceiver Channel Interface with PCS-PMA set to 8/10 bits (continued)	Two Receiver Sync Status Bits `rx_dataoutfull[10]` - `rx_syncstatus` (LSB) and `rx_dataoutfull[42]` - `rx_syncstatus` (MSB)
	Two Receiver Pattern Detect Bits `rx_dataoutfull[12]` - `rx_patterndetect` (LSB) and `rx_dataoutfull[44]` - `rx_patterndetect` (MSB)
20-bit Core fabric-Transceiver Channel Interface with PCS-PMA set to 20 bits	Two 10-bit Data (`rx_dataout`) `rx_dataoutfull[9:0]` - `rx_dataout` (LSByte) and `rx_dataoutfull[25:16]` - `rx_dataout` (MSByte)
	wo Receiver Sync Status Bits `rx_dataoutfull[10]` - `rx_syncstatus` (LSB) and `rx_dataoutfull[26]` - `rx_syncstatus` (MSB)
	`rx_dataoutfull[11]` and `rx_dataoutfull`[27]: 8B/10B disparity error indicator (`rx_disperr`)
	Two Receiver Pattern Detect Bits `rx_dataoutfull[12]` - `rx_patterndetect` (LSB) and `rx_dataoutfull[28]` - `rx_patterndetect` (MSB)
	`rx_dataoutfull[13]` and `rx_dataoutfull[29]`: Rate Match FIFO deletion status indicator (`rx_rmfifodatadeleted`) in non-PIPE/PCIe modes
	`rx_dataoutfull[14]` and `rx_dataoutfull[30]`: Rate Match FIFO insertion status indicator (`rx_rmfifodatainserted`) in non-PIPE/PCIe modes
	`rx_dataoutfull[15]` and `rx_dataoutfull[31]`: 8B/10B running disparity indicator (`rx_runningdisp`)

Table 2–29. rx_dataoutfull[63:0] Core fabric-Transceiver Channel Interface Signal Descriptions (Part 5 of 7)

Core fabric-Transceiver Channel Interface Description	Receive Signal Description (Based on HardCopy IV GX Supported Core fabric-Transceiver Channel Interface Widths)
20-bit Core fabric-Transceiver Channel Interface with PCS-PMA set to 10 bits	Two 10-bit Data `rx_dataoutfull[9:0]` - `rx_dataout` (LSByte) and `rx_dataoutfull[41:32]` - `rx_dataout` (MSByte)
	Two Receiver Sync Status Bits `rx_dataoutfull[10]` - `rx_syncstatus` (LSB) and `rx_dataoutfull[42]` - `rx_syncstatus` (MSB)
	`rx_dataoutfull[11]` and `rx_dataoutfull[43]`: 8B/10B disparity error indicator (`rx_disperr`)
	Two Receiver Pattern Detect Bits `rx_dataoutfull[12]` - `rx_patterndetect` (LSB) and `rx_dataoutfull[44]` - `rx_patterndetect` (MSB)
	`rx_dataoutfull[13]` and `rx_dataoutfull[45]`: Rate Match FIFO deletion status indicator (`rx_rmfifodatadeleted`) in non-PIPE/PCIe modes
	`rx_dataoutfull[14]` and `rx_dataoutfull[46]`: Rate Match FIFO insertion status indicator (`rx_rmfifodatainserted`) in non-PIPE/PCIe modes
	`rx_dataoutfull[15]` and `rx_dataoutfull[47]`: 8B/10B running disparity indicator (`rx_runningdisp`)

Table 2–29. rx_dataoutfull[63:0] Core fabric-Transceiver Channel Interface Signal Descriptions (Part 6 of 7)

Core fabric-Transceiver Channel Interface Description	Receive Signal Description (Based on HardCopy IV GX Supported Core fabric-Transceiver Channel Interface Widths)
32-bit mode	Four 8-bit un-encoded Data (rx_dataout) rx_dataoutfull[7:0]- rx_dataout (LSByte) rx_dataoutfull[23:16] rx_dataoutfull[39:32] rx_dataoutfull[55:48] - rx_dataout (MSByte)
	The following signals are used in 32-bit 8B/10B mode:
	Four Control Data Bits (rx_dataout) rx_dataoutfull[8] - rx_ctrldetect (LSB) rx_dataoutfull[24] rx_dataoutfull[40] rx_dataoutfull[56] - rx_ctrldetect (MSB)
	Four Receiver Error Detect Bits rx_dataoutfull[9]- rx_errdetect (LSB) rx_dataoutfull[25] rx_dataoutfull[41] rx_dataoutfull[57] - rx_errdetect (MSB)
	Four Receiver Pattern Detect Bits rx_dataoutfull[10]- rx_syncstatus (LSB) and rx_dataoutfull[26] rx_dataoutfull[42] rx_dataoutfull[58] - rx_syncstatus (MSB)
	Four Receiver Disparity Error Bits rx_dataoutfull[11]- rx_disperr (LSB) rx_dataoutfull[27] rx_dataoutfull[43] rx_dataoutfull[59] - rx_disperr (MSB)
	Four Receiver Pattern Detect Bits rx_dataoutfull[12]- rx_patterndetect (LSB) rx_dataoutfull[28] rx_dataoutfull[44] rx_dataoutfull[60] - rx_patterndetect (MSB)
	rx_dataoutfull[13], rx_dataoutfull[29], rx_dataoutfull[45] and rx_dataoutfull[61]: Rate Match FIFO deletion status indicator (rx_rmfifodatadeleted) in non-PIPE/PCIe modes
	rx_dataoutfull[14], rx_dataoutfull[30], rx_dataoutfull[46], and rx_dataoutfull[62]: Rate Match FIFO insertion status indicator (rx_rmfifodatainserted) in non-PIPE/PCIe modes

Table 2–29. rx_dataoutfull[63:0] Core fabric-Transceiver Channel Interface Signal Descriptions (Part 7 of 7)

Core fabric-Transceiver Channel Interface Description	Receive Signal Description (Based on HardCopy IV GX Supported Core fabric-Transceiver Channel Interface Widths)
32-bit mode (continued)	rx_dataoutfull[15], rx_dataoutfull[31], rx_dataoutfull[47], and rx_dataoutfull[63]: 8B/10B running disparity indicator (rx_runningdisp)
	The following signals are used in 32-bit SONET/SDH scrambled backplane mode:
	Four Control Data Bits (rx_dataout) rx_dataoutfull[7:0]- rx_dataout (LSByte) rx_dataoutfull[23:16] rx_dataoutfull[39:32] rx_dataoutfull[55:48] - rx_dataout (MSByte)
	rx_dataoutfull[8], rx_dataoutfull[24], rx_dataoutfull[40], and rx_dataoutfull[56]: four rx_a1a2sizeout
	Four Receiver Sync Status Bits rx_dataoutfull[10]- rx_syncstatus (LSB) rx_dataoutfull[26] rx_dataoutfull[42] rx_dataoutfull[58] - rx_syncstatus (MSB)
	Four Receiver Pattern Detect Bits rx_dataoutfull[12]- rx_patterndetect (LSB) rx_dataoutfull[28] rx_dataoutfull[44] rx_dataoutfull[60] - rx_patterndetect (MSB)
40-bit mode	Four 10-bit Control Data Bits (rx_dataout) rx_dataoutfull[9:0]- rx_dataout (LSByte) rx_dataoutfull[25:16] rx_dataoutfull[41:32] rx_dataoutfull[57:48] - rx_dataout (MSByte)
	Four Receiver Sync Status Bits rx_dataoutfull[10]- rx_syncstatus (LSB) rx_dataoutfull[26] rx_dataoutfull[42] rx_dataoutfull[58] - rx_syncstatus (MSB)
	Four Receiver Pattern Detect Bits rx_dataoutfull[12]- rx_patterndetect (LSB) rx_dataoutfull[28] rx_dataoutfull[44] rx_dataoutfull[60] - rx_patterndetect (MSB)

ALTGX_RECONFIG MegaWizard Plug-In Manager Setup for Channel and CMU PLL Reconfiguration Mode

Use the following steps to setup channel and CMU PLL reconfiguration mode in the ALTGX_RECONFIG MegaWizard Plug-In Manager:

1. Set the **What is the number of channels controlled by the reconfig controller?** option in the **Reconfiguration settings** screen. For more information, refer to "Total Number of Channels Controlled by the ALTGX_RECONFIG Instance" on page 2–35.
2. Select the C**hannel and TX PLL select/reconfig** option in the **Reconfiguration settings** screen, as shown in Figure 2–47.

   The following control signals are always available when you enable the **Channel and TX PLL select/reconfig** option:

   - `channel_reconfig_done`—The dynamic reconfiguration controller asserts the `channel_reconfig_done` signal to indicate that all the words of the **.mif** have been written into the transceiver.
   - `reconfig_address_out[5:0]`—This output signal provides the address value that you can use to read the appropriate word from the **.mif**. Use the value at this port in combination with the `reconfig_address_en` signal to decide when to write the next word.

   The following options are optional and available for selection in the **Channel and TX PLL Reconfiguration** screen:

   - `reset_reconfig_address`—Use this signal to reset the `reconfig_address_out[5:0]` value to **0**.
   - `reconfig_address_en`—The ALTGX_RECONFIG instance asserts this output signal to indicate the change in value on the `reconfig_address_out[5:0]` port. This signal only gets asserted after the dynamic reconfiguration controller completes writing a 16-bit word of the **.mif**.
   - `logical_tx_pll_sel` and `logical_tx_pll_sel_en`—For more information about these two ports, refer to "The logical_tx_pll_sel and logical_tx_pll_sel_en Ports" on page 2–109.
   - Through the `rx_tx_duplex_sel[1:0]` signal, the dynamic reconfiguration controller has information about whether the targeted transceiver channel is an **Receiver only**, **Transceiver only**, or **Receiver and Transmitter** configuration. For more information about this signal, refer to "Dynamic Reconfiguration Controller Port List" on page 2–11. The **.mif** also contains information about the transceiver channel configuration. The ALTGX_RECONFIG MegaWizard Plug-In Manager asserts the error signal if there is a mismatch between the `rx_tx_duplex_sel[1:0]` signal and the **.mif** contents.

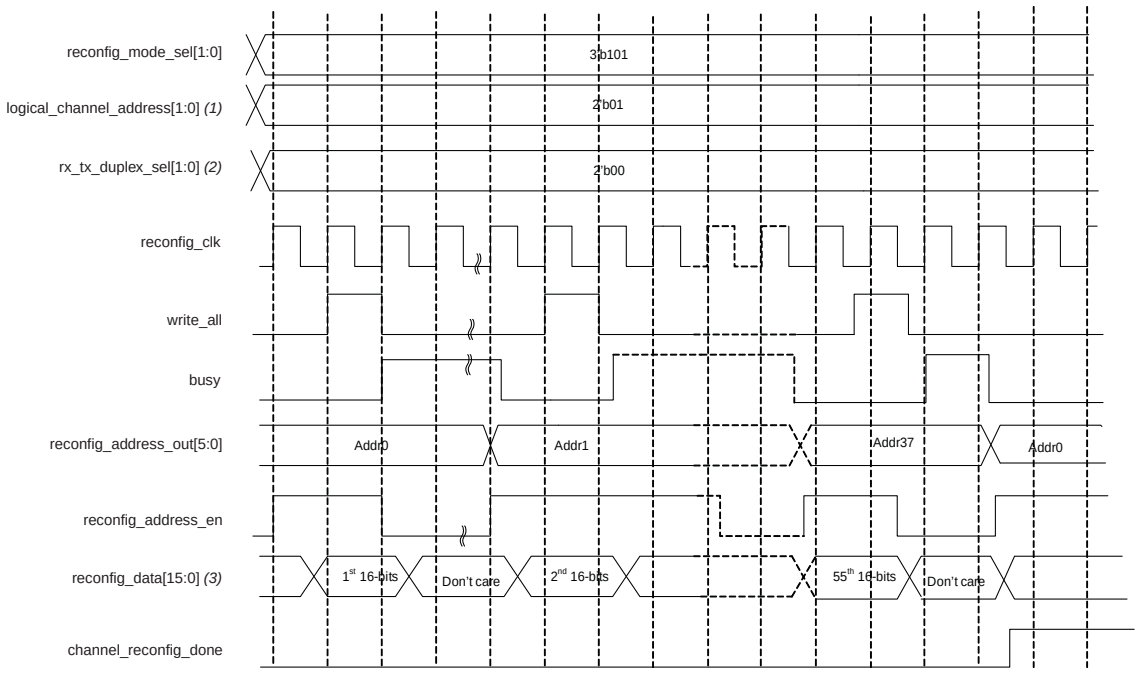
Channel and CMU PLL Reconfiguration Operation

In channel reconfiguration, only a write transaction can occur; no read transactions are allowed.

1. Set the `reconfig_mode_sel[2:0]` control signal to **3'b101**. When you use this feature, the dynamic reconfiguration controller requires that you provide the 16-bit words through the **.mif** on every write transaction.

2. Set the `rx_tx_duplex_sel[1:0]` port to enable the transmitter portion, receiver portion, or receiver and transmitter portion for reconfiguration.
3. Set the `logical_channel_address` port to specify the logical channel address of the transceiver channel.
4. Ensure the `busy` signal is low and assert the `write_all` signal for one `reconfig_clk` clock cycle.
5. Figure 2–47 shows a **.mif** write transaction when using Channel and CMU PLL Reconfiguration mode.

Figure 2–47. .mif write transaction in Channel and CMU PLL Reconfiguration Mode



Notes to Figure 2–47:

(1) The waveform assumes that the ALTGX_RECONFIG controls two channels. Therefore, the `logical_channel_address` signal is 2 bits wide. The `logical_channel_address` port is set to **2'b01** to reconfigure the second transceiver channel.

(2) The `rx_tx_duplex_sel[1:0]` port is set to **2'b00** to match the Receiver and Transmitter configuration of the specified transceiver channel.

(3) This waveform assumes that the transceiver channel is configured in Basic mode with the Receiver and Transmitter configuration. Therefore, the **.mif** size is 54.

Channel Reconfiguration with TX PLL Select Mode

This section describes the Channel Reconfiguration with TX PLL select mode.

You can reconfigure the data rate of a transceiver channel by switching between the two CMU PLLs present in a transceiver block and reconfiguring the receiver PLLs. The two CMU PLLs can be set to different base rates. You can choose one of the CMU PLLs based on the data rate you want the channel to run at.

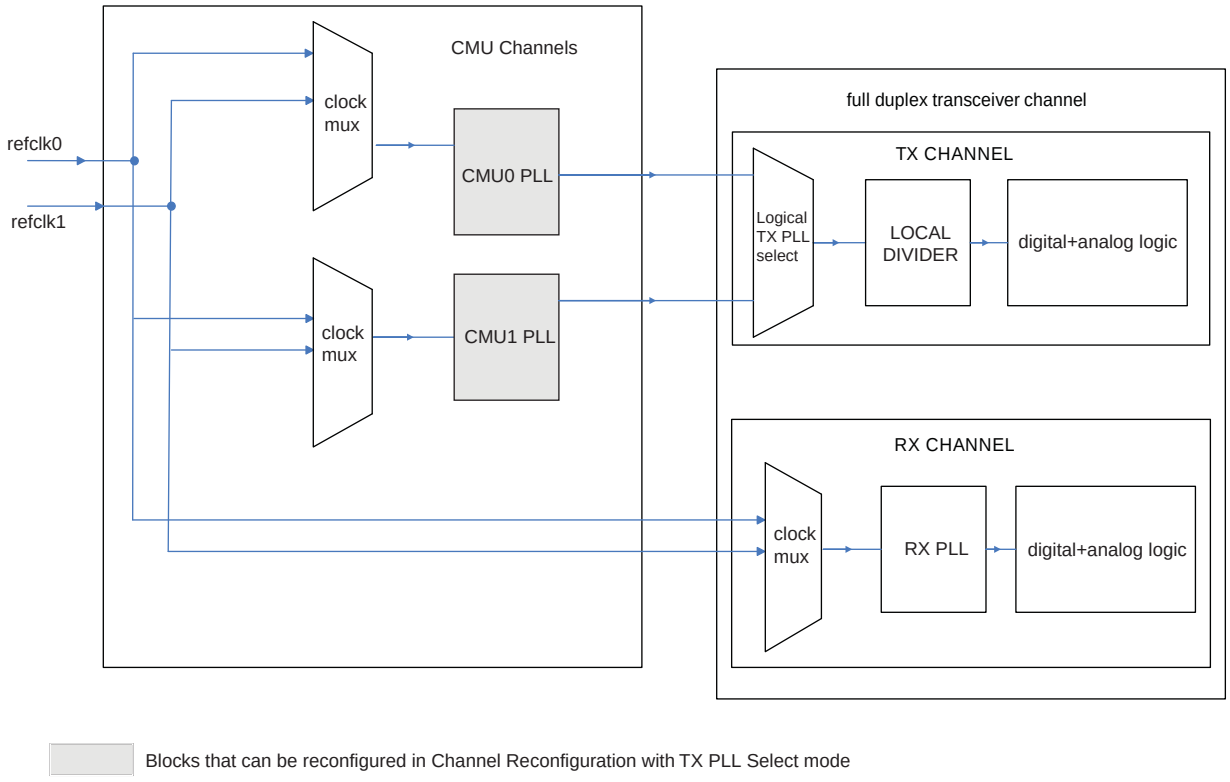
You can use the Channel Reconfiguration with TX PLL select mode along with the CMU PLL Reconfiguration mode. You can first reconfigure the second PLL to the desired data rate using CMU PLL Reconfiguration mode. Then use Channel Reconfiguration with TX PLL Select mode to reconfigure the transceiver channel to listen to the second PLL.

Blocks Reconfigured in the Channel Reconfiguration with TX PLL Select Mode

The block that gets reconfigured in this mode is the multiplexer, which selects the output of one of the CMU PLLs and forwards it to the transceiver channel.

Figure 2–48 shows the multiplexer that you can dynamically reconfigure using the Channel Reconfiguration with TX PLL select feature.

Figure 2–48. Channel Reconfiguration with TX PLL Select in a Transceiver Block



ALTGX MegaWizard Plug-In Manager Setup for Channel Reconfiguration with TX PLL Select Mode

To reconfigure the transceiver channel by selecting an alternate transmitter PLL instead of the existing CMU PLL, you must set up the ALTGX MegaWizard Plug-In Manager as shown in the following sections. The dynamic reconfiguration controller reconfigures the multiplexer, selecting between the outputs of the two CMU PLLs with the new information stored in the **.mif**.

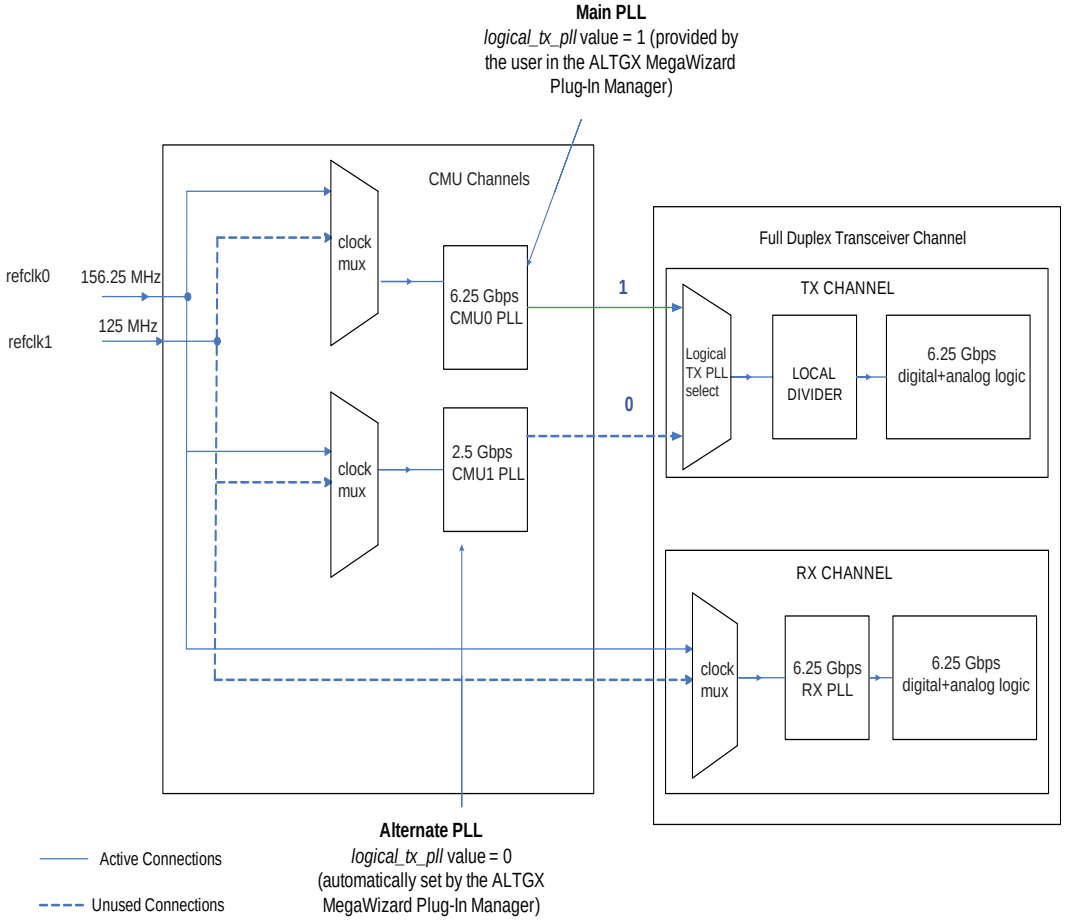
In addition to the seven settings explained in "ALTGX MegaWizard Plug-In Manager Setup for Channel and CMU PLL Reconfiguration Mode" on page 2–77, you must also set up the following design aspect:

Main PLL and Alternate PLL

To reconfigure the CMU PLL during run time, you need the flexibility to select one of the two CMU PLLs.

Consider that the transceiver channel is listening to CMU0 PLL and that you want to reconfigure CMU0 PLL, as shown in Figure 2–49.

Figure 2–49. Reconfiguring the CMU0 PLL



You can select CMU0 PLL by specifying its identity in the ALTGX MegaWizard Plug-In Manager. This identification is referred to as the logical tx pll value.

This value provides a logical identification to CMU0 PLL and associates it with a transceiver channel without requiring the knowledge of its physical location.

In the ALTGX MegaWizard Plug-In Manager, the transmitter PLL configuration set in the **General** screen is called the main PLL. When you provide the main PLL with a logical tx pll value, for example 1, the alternate PLL automatically takes the complement value 0. The logical tx pll value for the main PLL is stored along with the other transceiver channel information in the generated **.mif**.

You can reuse the **.mif** generated for one CMU PLL to reconfigure the other CMU PLL in the same or other transceiver blocks.

Provide the logical tx pll value for the main PLL in the **What is the main PLL logical reference index?** option in the **Reconfig Clks** screen.

CMU PLL Reconfiguration mode is also useful when used in combination with the dynamic reconfiguration mode: Channel Reconfiguration with CMU PLL select.

Consider that you have one transceiver channel listening to one CMU PLL of the transceiver block. You want to reconfigure the transceiver channel to a different data rate.

- You can first use CMU PLL Reconfiguration mode (set `reconfig_mode_sel[2:0]` to **3'b100**) and reconfigure the second unused CMU PLL of the transceiver block to the desired data rate. For more information, refer to "CMU PLL Reconfiguration Operation" on page 2–108.
- You can then use Channel Reconfiguration with CMU PLL select mode (set `reconfig_mode_sel[2:0]` to **3'b110**) and reconfigure the transceiver channel to listen to the second reconfigured CMU PLL. For more information, refer to "Channel Reconfiguration with TX PLL Select: Operation" on page 2–106.

The main PLL corresponds to the CMU PLL configuration set in the **General** screen of the ALTGX MegaWizard Plug-In Manager. The alternate PLL corresponds to the CMU PLL configuration set in the **Reconfig Alt PLL** screen.

ALTGX_RECONFIG MegaWizard Plug-In Manager Setup for Channel Reconfiguration with TX PLL Select Mode

The following sections describe the TX PLL selection operation in the ALTGX MegaWizard Plug-In Manager.

In addition to the first six settings listed in "ALTGX_RECONFIG MegaWizard Plug-In Manager Setup for Channel and CMU PLL Reconfiguration Mode" on page 2–101, you must set up the following input ports that are available for selection in the **Channel and TX PLL Reconfiguration** screen:

1. `logical_tx_pll_sel` and `logical_tx_pll_sel_en`—For more information about these two ports, refer to "The logical_tx_pll_sel and logical_tx_pll_sel_en Ports" on page 2–109.

   The following input port is available for selection in the **Error checks/Data rate switch** screen:

2. `rx_tx_duplex_sel[1:0]`—This signal informs the dynamic reconfiguration controller if the targeted transceiver channel configuration is **Receiver only**, **Transmitter only**, or **Receiver and Transmitter**. The **.mif** also contains information about the transceiver channel configuration. The ALTGX_RECONFIG MegaWizard Plug-In Manager asserts the error signal if there is a mismatch between the `rx_tx_duplex_sel [1:0]` signal and the **.mif** contents. For more information, refer to "Dynamic Reconfiguration Controller Port List" on page 2–11.

Channel Reconfiguration with TX PLL Select: Operation

In channel reconfiguration, only a write transaction can occur; no read transactions are allowed.

- Set the `reconfig_mode_sel[2:0]` control signal to **3'b001** to use the channel reconfiguration feature. When you use this feature, the dynamic reconfiguration controller requires that you provide a 16-bit word (`reconfig_data[15:0]`) on every write transaction using the `write_all` signal. This 16-bit word is part of a **.mif** that is generated by the Quartus II software when an ALTGX instance is compiled.
- Set the `rx_tx_duplex_sel[1:0]` port to enable the transmitter, receiver, or receiver and transmitter portion for reconfiguration.
- Set the `logical_channel_address` port to specify the logical channel address of the transceiver channel.
- Ensure the `busy` signal is low and assert the `write_all` signal for one `reconfig_clk` clock cycle.
- Figure 2–47 depicts a **.mif** write transaction when dynamically reconfiguring a transceiver channel. The **.mif** write transaction in Channel Reconfiguration with TX PLL Select mode remains the same, except for the value you must set at `reconfig_mode_sel[2:0]` and the differences in the **.mif** words utilized.
- Set `reconfig_mode_sel[2:0]` to **3'b001** for Channel Reconfiguration with TX PLL Select mode.

CMU PLL Reconfiguration Mode

You can use this mode to reconfigure only the CMU PLL without affecting the remaining blocks of the transceiver channel. When you reconfigure the CMU PLL of a transceiver block to run at a different data rate, all the transceiver channels listening to this CMU PLL also get reconfigured to the new data rate.

This reconfiguration mode is a **.mif**-based approach.

☞ CMU PLL Reconfiguration mode is applicable only to regular transceiver channels configured in non-Basic (PMA Direct) modes.

☞ CMU PLL Reconfiguration mode is not applicable to bonded, Basic (PMA Direct) ×1, Basic (PMA Direct) ×N, and CEI configurations.

☞ The `logical_channel_address` port is not applicable in CMU PLL Reconfiguration mode, even though it is available as an input.

TX PLL Powerdown

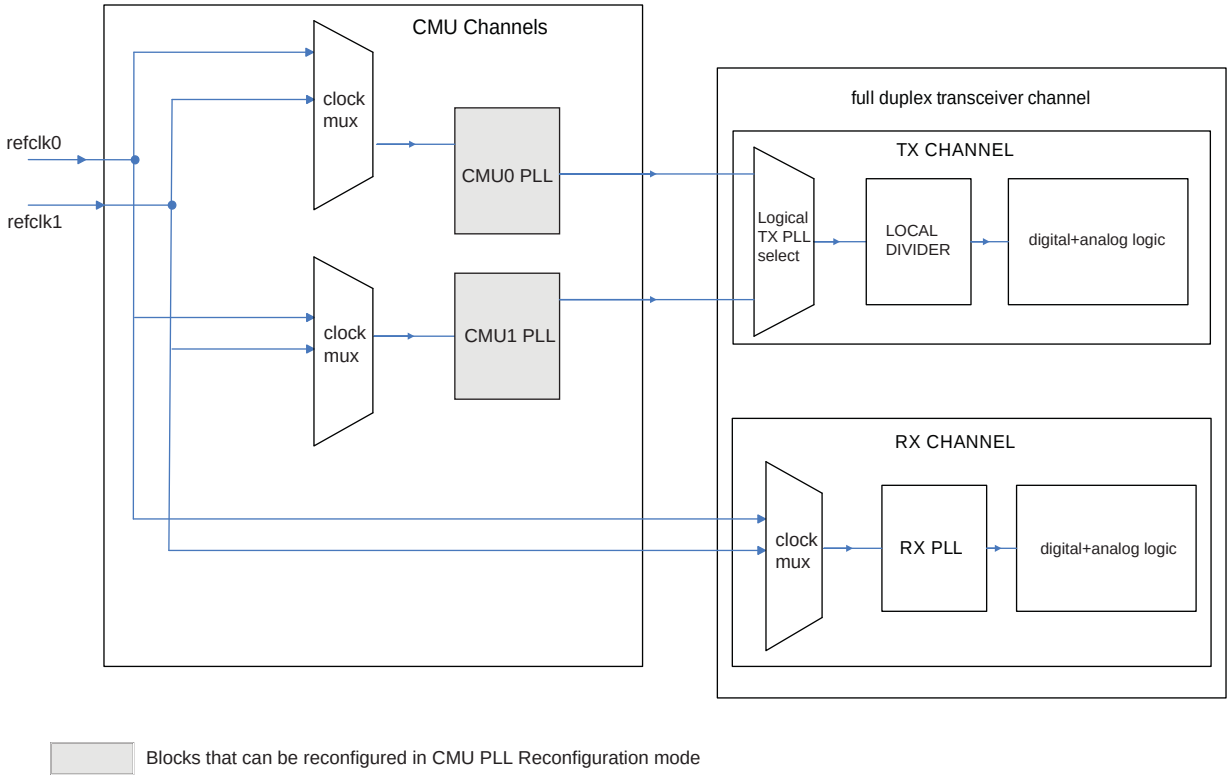
During CMU PLL Reconfiguration mode, the dynamic reconfiguration controller automatically powers down the selected CMU PLL until it completes reconfiguration. The ALTGX_RECONFIG instance does not provide any external ports to control the CMU PLL power down. When you reconfigure the CMU PLL, the `pll_locked` signal goes low. Therefore, after reconfiguring the transceiver, wait for the `pll_locked` signal from the ALTGX instance before continuing normal operation.

The dynamic reconfiguration controller powers down only the selected CMU PLL. The other CMU PLL is not affected.

Blocks Reconfigured in CMU PLL Reconfiguration Mode

Each transceiver block has two CMU PLLs—CMU0 PLL and CMU1 PLL.You can reconfigure each of these CMU PLLs to a different data rate in this dynamic reconfiguration mode. Figure 2–50 shows a conceptual view of both CMU PLLs in a transceiver block.

Figure 2–50. CMU PLLs in a Transceiver Block



ALTGX MegaWizard Plug-In Manager Setup for CMU PLL Reconfiguration Mode

When you want to reconfigure the CMU PLL to another data rate, enable **.mif** generation and set up the ALTGX MegaWizard Plug-In Manager as described in the following steps. The dynamic reconfiguration controller reconfigures the CMU PLL with the new information stored in the **.mif**.

1. Select the **Channel and Transmitter PLL reconfiguration** option in the **Reconfig** screen.
2. Provide the new data rate you want the CMU PLL to run at in the **General** screen.
3. Provide the logical reference index value in the **What is the main PLL logical reference index?** option in the **Reconfig Clks** screen.

   For more information, refer to "Selecting the Logical Reference Index of the CMU PLL" on page 2–77.

The logical reference index of CMU0 PLL within a transceiver block is always the complement of the logical reference index of CMU1 PLL.

This logical reference index value is stored as logical tx pll, along with the other transceiver channel settings in the **.mif**.

You can reuse the **.mif** generated for one CMU PLL to reconfigure the other CMU PLL in the same or in other transceiver blocks.

ALTGX_RECONFIG MegaWizard Plug-In Manager Setup for CMU PLL Reconfiguration Mode

The settings available for CMU PLL Reconfiguration mode are listed below.

In addition to the first six settings listed in "ALTGX_RECONFIG MegaWizard Plug-In Manager Setup for Channel and CMU PLL Reconfiguration Mode" on page 2–101, you must set up the following input ports that are available for selection in the **Channel and TX PLL Reconfiguration** screen:

The following input port is available for selection in the **Error checks/Data rate switch** screen:

1. `logical_tx_pll_sel` and `logical_tx_pll_sel_en`—For more information about these two ports, refer to "The logical_tx_pll_sel and logical_tx_pll_sel_en Ports" on page 2–109.
2. `rx_tx_duplex_sel[1:0]`—This signal informs the dynamic reconfiguration controller if the targeted transceiver channel configuration is **Receiver only**, **Transmitter only**, or **Receiver and Transmitter**. The **.mif** also contains information about the transceiver channel configuration. The ALTGX_RECONFIG MegaWizard Plug-In Manager asserts the error signal if there is a mismatch between the `rx_tx_duplex_sel[1:0]` signal and the **.mif** contents. For more information, refer to "Dynamic Reconfiguration Controller Port List" on page 2–11.

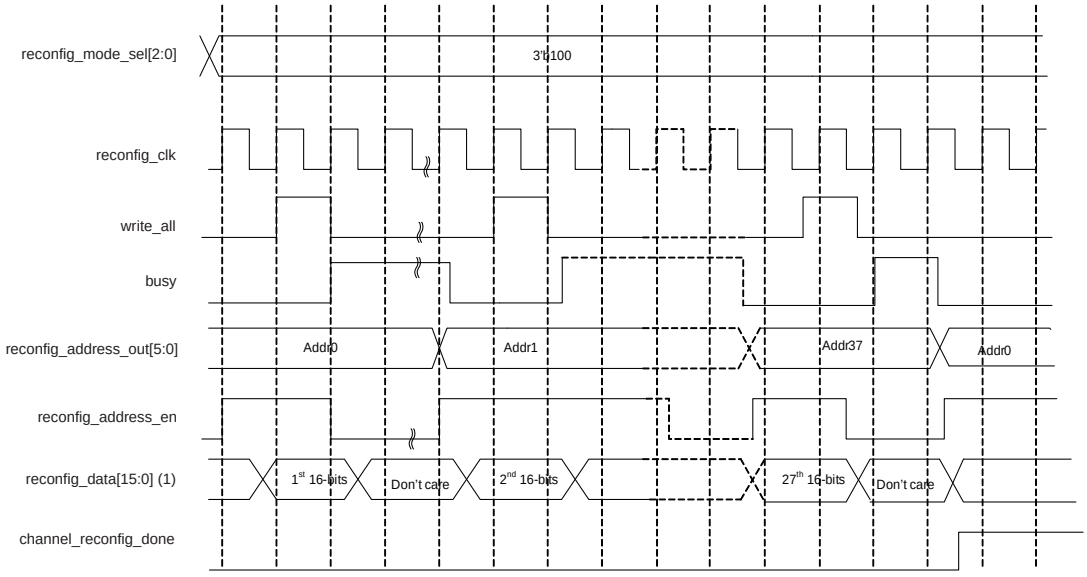
CMU PLL Reconfiguration Operation

1. Set the `reconfig_mode_sel[2:0]` signal to **3'b100** to activate this mode.
2. Set the `logical_channel_address` port to the logical channel address of the transceiver channel connected to the CMU PLL.
3. Ensure that the `busy` signal is low.
4. Initiate a write transaction by asserting the `write_all` signal for one `reconfig_clk` cycle to write the first 16-bit word of the **.mif**. Similarly, initiate a write transaction to write all the words of the **.mif**. You can use the `reconfig_address_out_en` port to determine when to initiate the next write transaction.

The dynamic reconfiguration controller asserts the `busy` signal for every write transaction initiated by you. The `busy` signal remains asserted until the complete 16-bit word has been written. The dynamic reconfiguration controller automatically increments the values on the `reconfig_address_out[5:0]` port. During reconfiguration, the dynamic reconfiguration controller powers down the CMU PLL until new values are written. The dynamic reconfiguration controller asserts the `channel_reconfig_done` signal to indicate that the CMU PLL reconfiguration is complete.

Figure 2–51 depicts a **.mif** write transaction in CMU PLL Reconfiguration mode.

Figure 2–51. CMU PLL Reconfiguration—.mif Write Transaction



Note to Figure 2–51:

(1) This waveform assumes that the transceiver channel is configured in **Receiver and Transmitter** configuration. Therefore, the **.mif** size is 8.

The logical_tx_pll_sel and logical_tx_pll_sel_en Ports

This section describes when to enable the `logical_tx_pll_sel` and `logical_tx_pll_sel_en` ports and how to use them in the following dynamic reconfiguration modes:

- Channel and CMU PLL Reconfiguration mode
- Channel Reconfiguration with TX PLL select mode
- CMU PLL Reconfiguration mode

These are optional input ports to the ALTGX_RECONFIG instance. The `logical_tx_pll_sel` and `logical_tx_pll_sel_en` ports are enabled by default when you enable the **Channel and TX PLL select/reconfig** option.

When you disable the `logical_tx_pll_sel` and `logical_tx_pll_sel_en` ports, the dynamic reconfiguration controller uses the logical reference index of the CMU PLL stored in the **.mif** generated (logical tx pll).

When you enable the `logical_tx_pll_sel` and `logical_tx_pll_sel_en` ports, the dynamic reconfiguration controller uses the value at `logical_tx_pll_sel` to identify the CMU PLL, only when you set `logical_tx_pll_sel_en` to **1'b1**.

How to Use the logical_tx_pll_sel Port?

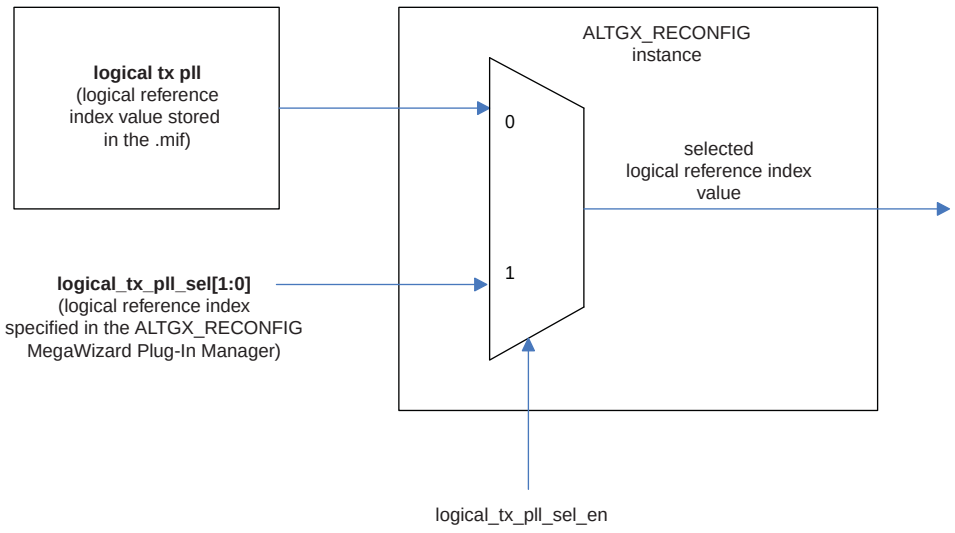
- Enable both the `logical_tx_pll_sel` and `logical_tx_pll_sel_en` ports.
- Specify the logical reference index of the CMU PLL used in the reconfiguration at `logical_tx_pll_sel`.
- Set a value of **0** or **1** at this port. These 0 and 1 values correspond to the logical reference index that you have already set in the ALTGX MegaWizard Plug-In Manager in the following options:
  - **What is the main transmitter PLL logical reference index?**
  - **What is the alternate transmitter PLL logical reference index?**
- Set `logical_tx_pll_sel_en` to **1'b1** (the dynamic reconfiguration controller uses the logical reference index value set at `logical_tx_pll_sel` instead of logical tx pll stored in the **.mif**).

When Can the logical_tx_pll_sel and logical_tx_pll_sel_en Ports be Used?

By default, the ALTGX_RECONFIG MegaWizard Plug-In Manager enables the `logical_tx_pll_sel` and `logical_tx_pll_sel_en` ports when you select the **Channel and TX PLL select/reconfig** option as the reconfiguration mode.

- Condition 1: If you want to use the `logical_tx_pll_sel` only under some conditions and use the logical tx pll (logical reference index value stored in the **.mif**) otherwise:
  - Both the `logical_tx_pll_sel` and the `logical_tx_pll_sel_en` ports are enabled by default.
  - Therefore, if you want to use the logical tx pll value stored in the **.mif**, set **logical_tx_pll_sel_en** to **1'b0**.
  - Instead, if you want to use the value at `logical_tx_pll_sel`, set `logical_tx_pll_sel_en` to **1'b1**. (The dynamic reconfiguration controller uses the value set at `logical_tx_pll_sel` only when you set `logical_tx_pll_sel_en` to **1'b1**, as shown in Figure 2–52).

Figure 2–52. Using logical_tx_pll_sel and logical_tx_pll_sell_en Ports



The values at `logical_tx_pll_sel` and `logical_tx_pll_sel_en` need to be held at a constant logic level until the `channel_reconfig_done` signal is asserted.

Table 2–30 shows how the dynamic reconfiguration controller selects between the logical reference index stored in the **.mif** (logical tx pll) and the logical reference index specified at the `logical_tx_pll_sel` port.

Table 2–30. Various Combinations of the logical_tx_pll_sel and logical_tx_pll_sel_en Ports

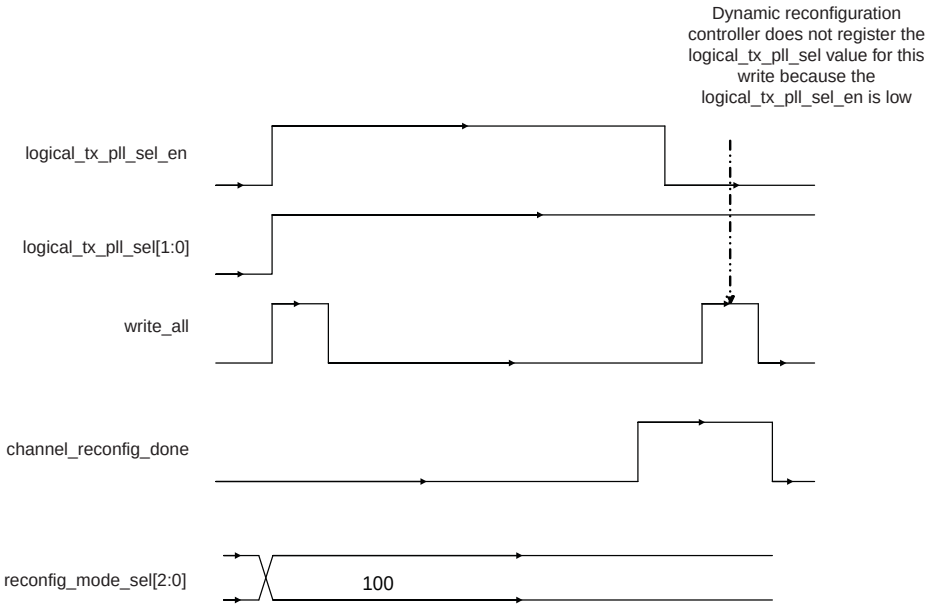
logical_tx_pll_sel	logical_tx_pll_sel_en	Logical Reference Index Value Selected by the ALTGX_RECONFIG Instance
enabled	enabled and value is 1	Value on the `logical_tx_pll_sel` port
enabled	enabled and value is 0	logical reference index value stored in the **.mif** (logical tx pll)
enabled	disabled	Value on the `logical_tx_pll_sel` port
enabled	disabled	logical reference index value stored in the **.mif** (logical tx pll)

When you configure a transceiver channel in the ALTGX MegaWizard Plug-In Manager, Altera recommends that you keep track of the transmitter PLL that drives the channel.

The `logical_tx_pll_sel` port does not modify transceiver settings on the RX side.

Figure 2–53 shows the required signal transitions to reconfigure the CMU PLL with a logical tx pll value of 1. Keep the `logical_tx_pll_sel` and `logical_tx_pll_sel_en` signals at a constant logic level until the dynamic reconfiguration controller asserts the `channel_reconfig_done` signal.

Figure 2–53. Signal Transition to Reconfigure a Transmitter PLL with a Reference Index Value of 1



- Condition 2: You need to reuse the **.mif** created for one CMU PLL on the other CMU PLL of the same transceiver block. Consider the following ALTGX settings:
  - The main PLL in the **General** screen is configured to switch from 2 Gbps to 3.125 Gbps. You have assigned a logical reference index of 1 for this main PLL (CMU0 PLL).
  - You generate a **.mif** for these settings.
  - The logical tx pll value stored in the **.mif** is **1**.
  - You want to reuse the same **.mif** to reconfigure `CMU1` PLL instead (your intention is to switch both CMU PLLs in the transceiver block to 3.125 Gbps). However, the logical reference index stored in the **.mif** is 1 (applicable only to `CMU0` PLL). You need to overwrite this logical tx pll value of 1 stored in the **.mif** with the logical reference index of `CMU1` PLL.
  - You can achieve this by using `logical_tx_pll_sel` and **logical_tx_pll_sel_en** and by setting `logical_tx_pll_sel` = **1'b0** (logical reference index of `CMU1` PLL) and `logical_tx_pll_sel_en` = **1'b1**.
  - By doing so, the dynamic reconfiguration controller writes the **.mif** contents to `CMU1` PLL (the logical reference index of this PLL is automatically **0** when you set the logical reference index of `CMU0` PLL as **1**).

The following section describes when to re-use the **.mif** generated for one CMU PLL on another CMU PLL, during the various dynamic reconfiguration modes.

For Channel and CMU PLL Reconfiguration and CMU PLL Reconfiguration Modes

- Consider that you create a **.mif** containing the desired ALTGX settings to reconfigure the `CMU0` PLL. Assume that the logical reference index you assigned to `CMU0` PLL is **0**.
- You can reuse this **.mif** created for `CMU0` PLL on `CMU1` PLL of the same transceiver block if you want to reconfigure `CMU1` PLL to the new data rate information stored in the **.mif**.
- You must set `logical_tx_pll_sel` to **1'b1** and `logical_tx_pll_sel_en` to **1'b1** and then write this **.mif** into the transceiver channel. By doing so, the dynamic reconfiguration controller overwrites the logical tx pll value stored in the **.mif** with the logical reference index of `CMU1` PLL.

For Channel Reconfiguration with TX PLL Select Mode

- Consider that you create a **.mif** containing the logical reference index of the TX PLL that the reconfigured transceiver channel needs to listen to.
  - `CMU1` PLL is reconfigured with the new data rate information stored in the **.mif**.
  - Assume that the logical reference index you assigned to `CMU1` PLL is **0**. The **.mif** then contains the logical reference index of `CMU1` PLL as **0** (logic tx pll = 0).
  - When you use Channel Reconfiguration with TX PLL Select mode, and reconfigure the transceiver channel with this **.mif**, the transceiver channel is reconfigured to listen to `CMU1` PLL.
  - If you want to reconfigure the transceiver channel to listen to `CMU0` PLL instead, you can reuse this **.mif**.
  - You must set `logical_tx_pll_sel`to **1'b1** and `logical_tx_pll_sel_en` to **1'b1** and then write this **.mif** into the transceiver channel. By doing so, the dynamic reconfiguration controller overwrites the logical tx pll value stored in the **.mif** with the logical reference index of `CMU0` PLL.
  - The transceiver channel is reconfigured to listen to `CMU0` PLL.
- Condition 3: Reuse the **.mif** created for one CMU PLL on a CMU PLL of another transceiver block.

  The following section describes when to re-use the **.mif** generated for one CMU PLL on a CMU PLL of another transceiver block during the various dynamic reconfiguration modes.

For Channel and CMU PLL Reconfiguration and CMU PLL Reconfiguration Modes:

- Consider that you create a **.mif** containing the desired ALTGX settings to reconfigure the `CMU0` PLL. Assume that the logical reference index you assigned to `CMU0` PLL is **1**.
- You can reuse this **.mif** created for `CMU0` PLL on `CMU0` PLL of another transceiver block if you want to reconfigure the `CMU0` PLL of the other transceiver block to exactly the same data rate information stored in the **.mif**.
- If the logical reference index of the other `CMU0` PLL is also the same as the logical tx pll value stored in the **.mif**, you can write the same **.mif** without using the `logical_tx_pll_sel` (set `logical_tx_pll_sel_en` to **1'b0**). You must set `logical_channel_address` to the logical channel address of the transceiver channel associated to the other `CMU0` PLL that you want to reconfigure.
- If the logical reference index of the other `CMU0` PLL is different from the logical tx pll value stored in the **.mif**, you must use the `logical_tx_pll_sel` port. You must also set `logical_channel_address` to the logical channel address of the transceiver channel associated to the other `CMU0` PLL you want to reconfigure.
- By doing so, the dynamic reconfiguration controller overwrites the logical tx pll value stored in the **.mif** with the logical reference index of the other `CMU0` PLL.
- The `CMU0` PLL of the other transceiver block is reconfigured with the new data rate information stored in the **.mif**.

For Channel Reconfiguration with TX PLL Select Mode

- Consider that you create a **.mif** containing the logical reference index of the TX PLL that the reconfigured transceiver channel needs to listen to.
- Assume that the logical reference index you assigned to `CMU0` PLL is **0**. The **.mif** then contains the logical reference index of `CMU0` PLL as **0** (logic tx pll = 0).
- When you use Channel Reconfiguration with TX PLL Select mode, and reconfigure the transceiver channel with this **.mif**, the transceiver channel is reconfigured to listen to `CMU0` PLL.
- If you want to reconfigure another transceiver channel of another transceiver block to listen to its own `CMU0` PLL, you can reuse this **.mif**.
- If the logical reference index of the other `CMU0` PLL is also the same as the logical tx pll value stored in the **.mif**, you can write the same **.mif** without using the `logical_tx_pll_sel` (set `logical_tx_pll_sel_en` to **1'b0**). You must set `logical_channel_address` to the logical channel address of the transceiver channel associated to the other `CMU0` PLL that you intend to reconfigure.
- If the logical reference index of the other `CMU0` PLL is different from the logical tx pll value stored in the **.mif**, you must use the `logical_tx_pll_sel` port. Set `logical_channel_address` to the logical channel address of the transceiver channel associated to the other `CMU0` PLL you intend to reconfigure.

- By doing so, the dynamic reconfiguration controller overwrites the logical tx pll value stored in the **.mif** with the logical reference index of the other `CMU0` PLL.
- The transceiver channel of another transceiver block is then reconfigured to listen to the `CMU0` PLL of its other transceiver block.

General Guidelines for Specifying the Input Reference Clocks

The following are general guidelines for setting up the input reference clocks in the **Reconfig Clks** screen of the ALTGX MegaWizard Plug-In Manager.

- Assign the identification numbers to all input reference clocks that are used in the design in the **Reconfig Clks** screen. You can set up a maximum of 10 input reference clocks and assign identification numbers from 1 to 10 (1, 2, 3, 4, 5, 6, 7, 8, 9, and 10).
- Keep the identification numbers consistent for all the **.mifs** generated in the design.
- Maintain the input reference clock frequencies settings for all the **.mifs**.

Consider you have two ALTGX instances in your design. Table 2–31 describes how to set up the identification numbers for the input reference clocks for all the **.mifs** involved in the design.

Table 2–31. Example for the Specifying the Input Reference Clocks (Part 1 of 4)

ALTGX Instances and Settings		
ALTGX Setting	**ALTGX Instance 1**	**ALTGX Instance 2**
What is the number of channels? option in the **General** screen	1 (Regular transceiver channel)	1 (Regular transceiver channel)

Table 2–31. Example for the Specifying the Input Reference Clocks (Part 2 of 4)

ALTGX Instances and Settings		
ALTGX Setting	**ALTGX Instance 1**	**ALTGX Instance 2**
Enable Channel and Transmitter PLL Reconfiguration option in the **Reconfig** screen	Enabled Assume the following settings for ALTGX Instance 1: ■ Consider that you have set up the transmitter PLL in the **General** screen to run at 3.125 Gbps (main PLL) ■ Consider that you have set up the input reference clock frequency of the main PLL as 156.25 MHz ■ The regular transceiver channel configured in ALTGX Instance 1 is therefore listening to this main PLL ■ Consider that your intention is to reconfigure the data rate of the regular transceiver channel to 2 Gbps ■ Select the **Use alternate Transmitter PLL** option to dynamically reconfigure the transceiver channel to listen to an alternate PLL (Channel Reconfiguration with TX PLL Select mode)	Enabled Assume the following settings for ALTGX Instance 2: ■ Consider that you have set up the transmitter PLL in the **General** screen to run at 1 Gbps (main PLL) ■ Consider that you have set up the input reference clock frequency of the main PLL as 125 MHz ■ The regular transceiver channel configured in ALTGX Instance 2 is therefore listening to this main PLL ■ Consider that your intention is to reconfigure the data rate of the regular transceiver channel to 1.250 Gbps ■ Select the **Use alternate Transmitter PLL** option to dynamically reconfigure the transceiver channel to listen to an alternate PLL (Channel Reconfiguration with TX PLL Select mode)
Use alternate Transmitter PLL option in the **Reconfig** screen	Enabled ■ Set up the alternate transmitter PLL to run at 2 Gbps ■ Set up the logical reference index of the alternate transmitter PLL (for example, **0**) ■ Set up the protocol and starting channel number settings ■ Consider that you have set up the input reference clock of the alternate transmitter PLL as 125 MHz.	Enabled ■ Set up the alternate transmitter PLL to run at 1.250 Gbps ■ Set up the logical reference index of the alternate transmitter PLL (for example: **1**) ■ Set up the protocol and starting channel number settings ■ Consider that you have set up the input reference clock of the alternate transmitter PLL as 156.25 MHz

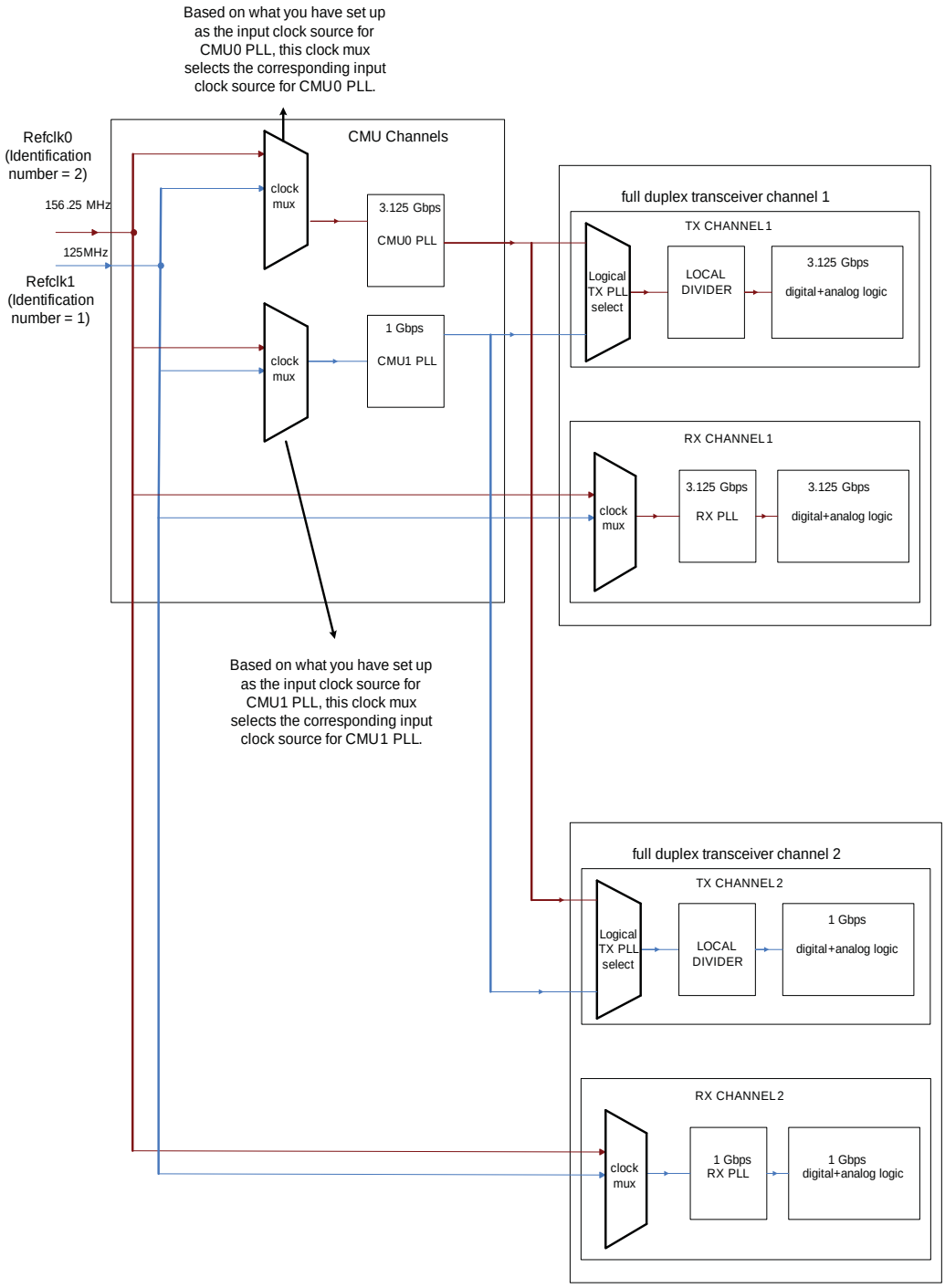
Table 2–31. Example for the Specifying the Input Reference Clocks (Part 3 of 4)

ALTGX Instances and Settings		
ALTGX Setting	**ALTGX Instance 1**	**ALTGX Instance 2**
What is the main transmitter PLL logical reference index? option in the **Reconfig Clks** screen	■ The ALTGX MegaWizard Plug-In Manager automatically sets up the logical reference index of the main PLL as the inverse of the logical reference index of the alternate transmitter PLL ■ In this case, the logical reference index of the main transmitter PLL is set up as **1**.	■ The ALTGX MegaWizard Plug-In Manager automatically sets up the logical reference index of the main PLL as the inverse of the logical reference index of the alternate transmitter PLL ■ In this case, the logical reference index of the main transmitter PLL is set up as **0**.
What is the selected input clock source for the Transmitter PLL and Receiver PLL? option in the **Reconfig Clks** screen	■ In ALTGX instance 1, you have set up the input reference clock as 156.25 MHz for main transmitter PLL ■ Assign an identification number to 156.25 MHz. Because there are a total of two input reference clock frequencies in this design (156.25 MHz and 125 MHz), you can assign a value of 1 or 2 (for example, **2**)	■ In ALTGX instance 2, you have set up the input reference clock as 125 MHz for main transmitter PLL ■ Assign an identification number to 125 MHz. Because there are a total of two input reference clock frequencies in this design (156.25 MHz and 125 MHz), you can assign a value of 1 or 2 (for example, **1**)
What is the selected input clock source for the alternate transmitter PLL? option in the **Reconfig Clks** screen	When you set up the identification number of the input clock source for the main transmitter PLL (156.25 MHz) as 2, you must set up the identification of the input clock source for the alternate transmitter PLL (125 MHz) as **1**.	When you set up the identification number of the input clock source for the main transmitter PLL (125 MHz) as 2, you must set up the identification of the input clock source for the alternate transmitter PLL (156.25 MHz) as **1**.

Table 2–31. Example for the Specifying the Input Reference Clocks (Part 4 of 4)

ALTGX Instances and Settings		
ALTGX Setting	**ALTGX Instance 1**	**ALTGX Instance 2**
.mif Generation	■ Enable the Quartus II settings for generating a **.mif** for ALTGX instance 1 ■ For more information about these settings, refer to "Quartus II Settings for .mif Generation" on page 2–70 ■ Consider that the name of the **.mif** generated is *ALTGX_instance1*.**mif**	■ Enable the Quartus II settings for generating a **.mif** for ALTGX instance 2 ■ For more information about these settings, refer to "Quartus II Settings for .mif Generation" on page 2–70 ■ Consider that the name of the **.mif** generated is *ALTGX_instance2*.**mif**
	Because there are two **.mifs** generated for this design, you must maintain the identification numbers of the input reference clocks in this design for both **.mifs**. Figure 2–54 shows the input reference clock connections to the transceiver channels based on what you set as the input clock source for each of the two transmitter PLLs in the design.	

Figure 2–54. Input Reference Clocks Connections to the Transceiver Channels



Design Examples: Dynamic Reconfiguration Controller (ALTGX_RECONFIG)

The following design examples describe the possible topologies of the dynamic reconfiguration controller with ALTGX instances. Table 2–32 lists the various dynamic reconfiguration examples described in this section.

Table 2–32. Design Examples Using the Dynamic Reconfiguration Controller

Example	Overview
Example 1	This example describes a single controller controlling multiple instances of an ALTGX megafunction. This example illustrates "Method 1" on page 2–54 of the PMA controls reconfiguration mode.
Example 2	This example describes a single controller controlling a single ALTGX instance. This example illustrates "Method 1" on page 2–54 of the PMA controls reconfiguration mode.
Example 3	This example describes the HDL construct requirements if you are stamping the ALTGX instances several times. Each ALTGX instance in turn can have more than one transceiver channel. This example illustrates "Method 2" on page 2–56 of the PMA controls reconfiguration mode.
Example 4	This design example explains the steps to dynamically divide the transmit data rate of a transceiver channel by 4, 2, or 1 without requiring **.mif** generation.
Example 5	This design example explains the steps to dynamically reconfigure a CMU PLL using CMU PLL Reconfiguration mode. This example illustrates the generation and usage of a **.mif** file.
Example 6	This design example explains the steps to dynamically reconfigure a transceiver channel between a GIGE configuration and a SONET/SDH OC48 configuration.

Example 1: One Reconfiguration Controller Connected to Multiple ALTGX Instances

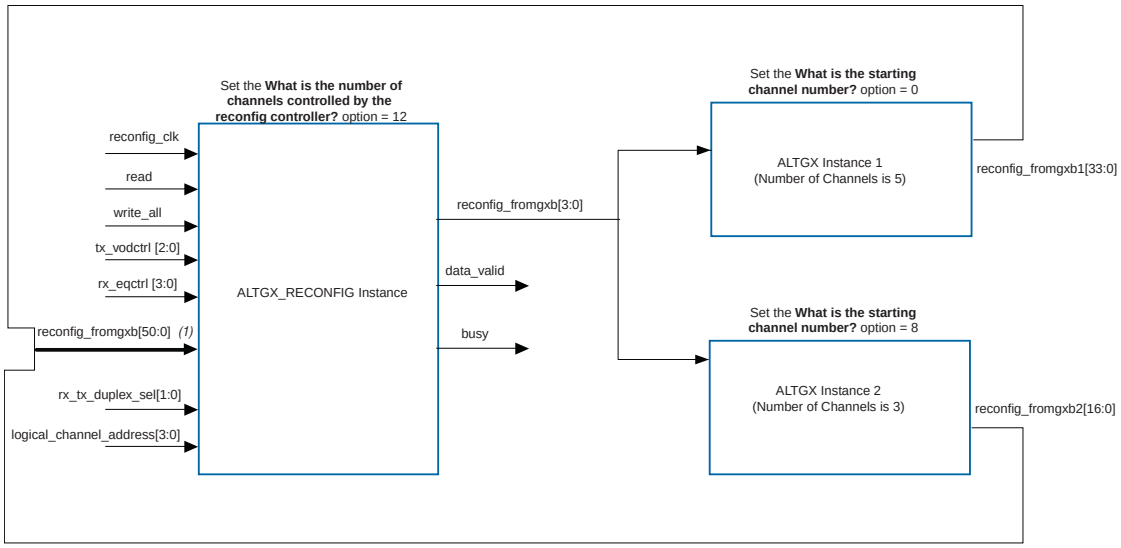
Consider a design with two ALTGX instances: ALTGX instance 1 with five transceiver channels and ALTGX instance 2 with three transceiver channels.

Assume the following for Example 1:

- ALTGX instance 1 and ALTGX instance 2 cannot be physically packed into the same transceiver block.
- One dynamic reconfiguration controller controls both ALTGX instances.
- You want to dynamically reconfigure the transmit V_{OD} PMA control (`tx_vodctrl`) of the first channel of ALTGX instance 1 and receiver equalization PMA control (`rx_eqctrl`) of the second channel of ALTGX instance 2.

Figure 2–55 shows the ALTGX_RECONFIG instance connected to both ALTGX instance 1 and ALTGX instance 2.

Figure 2–55. PMA Controls Reconfiguration for Example 1



Note to Figure 2–55:

(1) `reconfig_fromgxb[50:0]` = {`reconfig_fromgxb2[16:0]`, `reconfig_fromgxb1[33:0]`}.

Table 2–33 lists the ALTGX and ALTGX_RECONFIG Settings and Instances for Example 1.

Table 2–33. ALTGX and ALTGX_RECONFIG Settings and Instances for Example 1 (Part 1 of 2)

ALTGX Settings and Instances			ALTGX_RECONFIG Settings and Instance	
ALTGX Setting	**ALTGX Instance 1**	**ALTGX Instance 2**	**ALTGX_RECONFIG Setting**	**ALTGX_RECONFIG Instance 1**
What is the number of channels? option in the **General** screen	**5** (Regular transceiver channels)	**3** (Regular transceiver channels)	**What is the number of channels controlled by the reconfig controller?** option in the **Reconfiguration settings** screen. *For more information, refer to* "Total Number of Channels Controlled by the ALTGX_RECONFIG Instance" on page 2–35.	■ Determine the highest logical channel address (10). ■ Round it up to the next multiple of 4. ■ Set this option to **12**.

Table 2–33. ALTGX and ALTGX_RECONFIG Settings and Instances for Example 1 (Part 2 of 2)

ALTGX Settings and Instances			ALTGX_RECONFIG Settings and Instance	
ALTGX Setting	**ALTGX Instance 1**	**ALTGX Instance 2**	**ALTGX_RECONFIG Setting**	**ALTGX_RECONFIG Instance 1**
What is the starting channel number? option in the **Reconfig** screen. For more information, refer to "Logical Channel Addressing" on page 2–23.	■ Set this option to **0**. ■ The logical channel addresses of the 1st to 5th channels are **0**, **1**, **2**, **3**, and **4**, respectively.	■ Set this option to **8**. ■ The logical channel addresses of the 1st to 3rd channels are **8**, **9**, and **10**, respectively.	**Use the 'logical_channel_address' port for Analog controls reconfiguration** option in the **Analog controls** screen.	■ Select this option. ■ The ALTGX_RECONFIG MegaWizard Plug-In Manager enables the `logical_channel_address[3:0]` input port
Analog controls (VOD, Pre-emphasis, and Manual Equalization) option in the **Reconfig** screen	Enable this option.	Enable this option.	Use the `rx_tx_duplex_sel` port to enable the RX only, TX only, or duplex reconfiguration option in the **Error checks/Data rate switch** screen.	■ Select this option. ■ The ALTGX_RECONFIG MegaWizard Plug-In Manager enables the `rx_tx_duplex_sel[1:0]` input port.
			The various analog controls in the **Analog controls** screen.	■ Select the `tx_vodctrl` and `rx_eqctrl` controls. ■ `tx_vodctrl` is 3 bits wide ■ `rx_eqctrl` is 4 bits wide.
`reconfig_fromgxb1` and `reconfig_fromgxb2` outputs.	`reconfig_fromgxb1` is 34 bits wide (2 * 17)	`reconfig_fromgxb2` is 17 bits wide (1 * 17)	`reconfig_fromgxb` input	`reconfig_fromgxb` is 51 bits wide (3 * 17, 12 channels can logically fit into three transceiver blocks)

ALTGX Instances and ALTGX_RECONFIG Instances Connections

Use the following steps to connect the ALTGX and ALTGX_RECONFIG instances:

1. Connect the `reconfig_fromgxb1[33:0]` output port from ALTGX instance 1 to the `reconfig_fromgxb [33:0]` input port of the ALTGX_RECONFIG instance.
2. Similarly, connect the `reconfig_fromgxb2[16:0]` output port from ALTGX instance 2 to the `reconfig_fromgxb[50:34]` input port of the ALTGX_RECONFIG instance.
3. Connect the `reconfig_togxb[3:0]` output port of the ALTGX_RECONFIG instance to the `reconfig_togxb[3:0]` input ports of both ALTGX instance 1 and ALTGX instance 2.

For more information, refer to "Connecting the reconfig_fromgxb and reconfig_togxb Ports" on page 2–40.

Dynamically Reconfiguring the tx_vodctrl and rx_eqctrl PMA Controls Using Method 1

For more information about dynamically reconfiguring the PMA controls using Method 1, refer to "Write Transaction" on page 2–58 in "Method 1" on page 2–54.

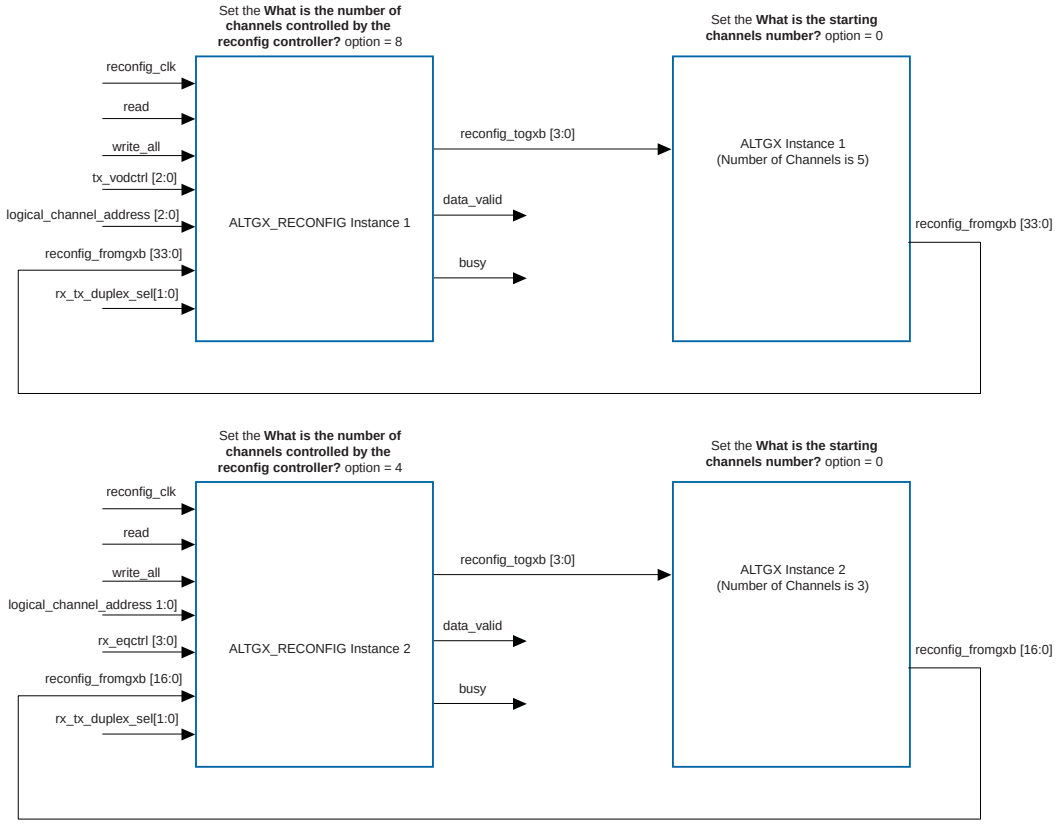
Example 2: Two ALTGX_RECONFIG Instances Connected to Two ALTGX Instances

This design example has two instances of distinct configurations: ALTGX instance 1 with five transceiver channels and ALTGX instance 2 with three transceiver channels.

This configuration requires separate dynamic reconfiguration controllers for the two instances. This scenario covers the case of multiple dynamic reconfiguration controllers controlling multiple instances of the ALTGX.

Figure 2–56 shows ALTGX_RECONFIG instance 1 connected to ALTGX instance 1 and ALTGX_RECONFIG instance 2 connected to ALTGX instance 2.

Figure 2–56. PMA Controls Reconfiguration for Example 2



Assume that you want to reconfigure the transmit V_{OD} PMA control of the second channel of the ALTGX instance 1 and the receive equalization PMA control of the third channel of ALTGX instance 2. The following are the steps to set up the configuration.

Table 2–34 lists the ALTGX and ALTGX_RECONFIG Settings and Instances for Example 2.

Table 2–34. ALTGX and ALTGX_RECONFIG Settings and Instances for Example 2

ALTGX Settings and Instances			ALTGX_RECONFIG Settings and Instance	
ALTGX Setting	**ALTGX Instance 1**	**ALTGX Instance 2**	**ALTGX_RECONFIG Setting**	**ALTGX_RECONFIG Instance 1**
What is the number of channels? option in the **General** screen	**5** (Regular transceiver channels)	**3** (Regular transceiver channels)	**What is the number of channels controlled by the reconfig controller?** option in the **Reconfiguration settings** screen *For more information, refer to* "Total Number of Channels Controlled by the ALTGX_RECONFIG Instance" on page 2–35.	■ Determine the highest logical channel address (4). ■ Round it up to the next multiple of 4. ■ Set this option to **8**.
What is the starting channel number? option in the **Reconfig** screen. For more information, refer to "Logical Channel Addressing" on page 2–23.	■ Set this option to **0**. ■ The logical channel addresses of the 1st to 5th channels are **0**, **1**, **2**, **3**, and **4**, respectively.	■ Set this option to **0**. ■ The logical channel addresses of the 1st to 3rd channels are **0**, **1**, and **2**, respectively.	**Use 'logical_channel_address' port for Analog controls reconfiguration** option in the **Analog controls** screen	■ Select this option. ■ The ALTGX_RECONFIG MegaWizard Plug-In Manager enables the `logical_channel_address[3:0]` input port.
Analog controls (VOD, Pre-emphasis, and Manual Equalization) option in the **Reconfig** screen	Enable this option.	Enable this option.	Use the `rx_tx_duplex_sel` port to enable the RX only, TX only, or duplex reconfiguration option in the **Error checks/Data rate switch** screen.	■ Select this option. ■ The ALTGX_RECONFIG MegaWizard Plug-In Manager enables the `rx_tx_duplex_sel[1:0]` input port.
			The various analog controls in the **Analog controls** screen.	■ Select the `tx_vodctrl` control. ■ `tx_vodctrl` is 3 bits wide.
`reconfig_fromgxb` output	`reconfig_fromgxb` is 34 bits wide (2 * 17)	`reconfig_fromgxb` is 17 bits wide (1 * 17)	`reconfig_fromgxb` input	`reconfig_fromgxb` is 34 bits wide (2 * 17, 8 channels can logically fit into three transceiver blocks).

ALTGX Instances and ALTGX_RECONFIG Instances Connections

Use the following steps to connect the ALTGX and ALTGX_RECONFIG instances:

1. Connect the `reconfig_fromgxb` signal from each ALTGX instance to the same signal of the corresponding ALTGX_RECONFIG instance. For more information, refer to Figure 2–56.
2. Connect the `reconfig_togxb` signal from each ALTGX_RECONFIG instance to the same signal of the corresponding ALTGX instance.

Dynamically Reconfiguring the tx_vodctrl PMA Control of ALTGX Instance 1 from ALTGX_RECONFIG Instance 1 Using Method 1

For more information about dynamically reconfiguring the PMA controls using Method 1, refer to "Write Transaction" in "Method 1" on page 2–54.

Dynamically Reconfiguring the rx_eqctrl PMA Control of ALTGX Instance 2 from ALTGX_RECONFIG Instance 2 Using Method 1:

For more information about dynamically reconfiguring the PMA controls using Method 1, refer to "Write Transaction" in "Method 1" on page 2–54.

Example 3: One ALTGX_RECONFIG Instance Connected to an ALTGX Instance Stamped Five Times

This design example consists of five channels of transceivers. This configuration has one dynamic reconfiguration controller to control five channels and includes stamping five instantiations of one channel ALTGX instance configuration. This example assumes the instantiation name is "instance1".

ALTGX Instance with One Transceiver Channel

Use the following steps for ALTGX instances with one transceiver channel:

1. Set the **What is the number of channels?** option in the **General** screen of the ALTGX MegaWizard Plug-In Manager to **1**.
2. Enable the **Analog controls (VOD, Pre-emphasis, and Manual Equalization)** option in the **Reconfig** screen of the ALTGX MegaWizard Plug-In Manager.
3. The `reconfig_fromgxb` output signal is transceiver block based; the number of bits for this instance is 17. This is because the number of channels is one and it can logically fit into a single transceiver block. The `reconfig_togxb` input signal is a fixed bus (4 bits wide).
4. Set the **What is the starting channel number**? option in the **Reconfig** screen of the ALTGX MegaWizard Plug-In Manager to **0**. For more information, refer to "Logical Channel Addressing" on page 2–23.
5. Click **Finish**.

Instantiating Five Transceiver Channels Using the Same ALTGX Instance

When you stamp instance 1 five times, the **What is the starting channel number?** options of the other five stamped instances (assume instance2, instance3, instance4, instance5, and instance6) are **4**, **8**, **12**, and **16**, respectively. For more information, refer to "Logical Channel Addressing" on page 2–23.

Dynamic Reconfiguration Controller Instance (ALTGX_RECONFIG Instance)

Use the following steps for the dynamic reconfiguration controller instance (ALTGX_RECONFIG instance):

1. Launch the ALTGX_RECONFIG MegaWizard Plug-In Manager.
2. Set the **What is the number of channels controlled by the reconfig controller?** option in the **Reconfiguration settings** screen of the ALTGX_RECONFIG MegaWizard Plug-In Manager to **20**. This enables five sets of the MegaWizard Plug-In Manager signals (`reconfig_fromgxb[84:0]`).
3. Connect each of the stamped ALTGX instances to one set of the MegaWizard Plug-In Manager signals.
4. Select the necessary write and read controls to write in and read out from the V_{OD}, pre-emphasis, equalization, and DC gain options. For example, if you select the V_{OD} setting, the `tx_vodctrl` signal is 60 bits wide (3 bits per channel). The `tx_vodctrl [2:0]` corresponds to the single channel of the first stamped instance. The bits `tx_vodctrl [11:3]` are not used because they correspond to the unused channels in the first stamped instance with logical channel addresses 1 to 3. Similarly, `tx_vodctrl [14:12]` corresponds to the single channel of the second stamped instance, and so on.

ALTGX Instances and ALTGX_RECONFIG Instance Connections

Use the following steps to connect the ALTGX and ALTGX_RECONFIG instances:

1. Connect the `reconfig_fromgxb` signal from each ALTGX instance to the same signal in the ALTGX_RECONFIG instance. You must connect it in such a way that the `reconfig_fromgxb` output port of the first ALTGX instance (ALTGX instance with the **What is the starting channel number?** option of 0) is connected to the LSB of the `reconfig_fromgxb` input port of the ALTGX_RECONFIG instance, and so on.
2. Connect the `reconfig_togxb` signal from the ALTGX_RECONFIG instance to the same signal in each of the ALTGX instances.

Dynamically Reconfiguring the tx_vodctrl of Instance 1 Using Method 2

Use the following steps to dynamically reconfigure `tx_vodctrl` on instance 1 using Method 2:

1. Set the `tx_vodctrl` port to the desired setting. For example, if you want to write a V_{OD} value of 2, set the `tx_vodctrl [2:0]` port to **3'b010**.
2. For more information, refer to "Method 2" on page 2–56.

When you perform a write transaction using Method 2, the values on the PMA control ports are written on all transceiver channels connected to the dynamic reconfiguration controller. Therefore, ensure that you also have the desired values on `tx_vodctrl [59:3]`. If you want to ensure that the V_{OD} settings of the remaining channels are not affected, you can optionally perform a read transaction and obtain the existing values and write back the same values.

Example 4: Data Rate Division in TX Mode

This design example explains the steps to dynamically divide the transmit data rate of a transceiver channel by 4, 2, or 1 without requiring **.mif** generation.

The design contains the following two instances:

- ALTGX instance 1—Two regular transceiver channels configured in Basic functional mode with 8B/10B enabled and running at 4.25 Gbps data rate. You can reconfigure the mode dynamically between 4.25 Gbps, 2.125 Gbps, and 1062.5 Mbps.
- ALTGX_RECONFIG instance 1—A single dynamic reconfiguration controller connected to ALTGX instance 1.

Use the following steps to dynamically reconfigure the transmit data rate of the transceiver channel:

1. Create a Basic functional mode by setting the operation mode to **Receiver and Transmitter** configuration and the **What is the number of channels?** option to **2**.
2. Set up the options shown in Table 2–35 for both the ALTGX and ALTGX_RECONFIG MegaWizard Plug-In Managers.

Table 2–35. Data Rate Division in TX Dynamic Reconfiguration Mode for Example 4 (Part 1 of 3)

ALTGX Settings and Instances		ALTGX_RECONFIG Settings and Instance	
ALTGX Setting	**ALTGX Instance 1 (Basic Functional Mode, Receiver and Transmitter Operation Mode)**	**ALTGX_RECONFIG Setting**	**ALTGX_RECONFIG Instance 1**
What is the deserializer block width? option	Select double-width mode. This is required because the highest data rate in this example is 4.25 Gbps (single-width mode can be selected only up to 3.750 Gbps).	**What is the number of channels controlled by the reconfig controller?** option in the **Reconfiguration settings** screen For more information about this setting, refer to "Total Number of Channels Controlled by the ALTGX_RECONFIG Instance" on page 2–35.	■ Determine the highest logical channel address (1). ■ Round it up to the next multiple of 4. ■ Set this option to **4**.
What is the channel width? option	■ You can set the channel width to **16** or **32**. ■ The lowest core fabric frequency allowed in the Quartus II software is 25 MHz. Therefore, the transceiver runs at 1062.5 Mbps with a 32-bit core fabric-Transceiver interface. The core fabric clock frequency in this case is 26.5 MHz (1062.5/40 = 26.5625).	**Data Rate Division in TX** option in the **Reconfiguration Settings** screen	■ Enable this option. ■ This creates the `rate_switch_ctrl [1:0]` input signal. ■ Refer to Table 2–24 to set a value at the `rate_switch_ctrl [1:0]` port.

Table 2–35. Data Rate Division in TX Dynamic Reconfiguration Mode for Example 4 (Part 2 of 3)

ALTGX Settings and Instances		ALTGX_RECONFIG Settings and Instance	
ALTGX Setting	**ALTGX Instance 1 (Basic Functional Mode, Receiver and Transmitter Operation Mode)**	**ALTGX_RECONFIG Setting**	**ALTGX_RECONFIG Instance 1**
What is the input clock frequency? option	Set the input frequency to **106.25 MHz**.	Use the `rate_switch_out` port to read out the current data rate division option in the **Error checks/Data rate switch** screen.	■ You can optionally enable the `rate_switch_out [1:0]` output signal by selecting this option. ■ For more information, refer to "Total Number of Channels Controlled by the ALTGX_RECONFIG Instance" on page 2–35, which shows the values for each of the `rate_switch_ctrl [1:0]` settings. ■ Use the `rate_switch_ctrl [1:0]` signal only for dividing the data rate of the transmit side. To divide the data rate for both transmit and receive sides, a **.mif**-based approach is required.
What is the starting channel number? option in the **Reconfig** screen. (For more information, refer to "Total Number of Channels Controlled by the ALTGX_RECONFIG Instance" on page 2–35).	■ Set this option to **0**. ■ The logical channel addresses of the 1st and 2nd channels are **0** and **1**, respectively.	The `logical_channel_address[1:0]` port	■ This port is enabled automatically because the number of channels controlled is more than 1. ■ Specify the channel to be reconfigured at this `logical_channel_address [1:0]` port.
`reconfig_fromgxb1` output	■ `reconfig_fromgxb1` is 17 bits wide (1 * 17). ■ Connect `reconfig_fromgxb1 [16:0]` to the `reconfig_fromgxb [16:0]` input of the ALTGX_RECONFIG instance.	`reconfig_fromgxb` input	■ `reconfig_fromgxb` is 17 bits wide (1 * 17, 4 regular transceiver channels can logically fit into one transceiver block). ■ Connect `reconfig_fromgxb [16:0]` to the `reconfig_fromgxb1 [16:0]` output of the ALTGX instance.

Table 2–35. Data Rate Division in TX Dynamic Reconfiguration Mode for Example 4 (Part 3 of 3)

ALTGX Settings and Instances		ALTGX_RECONFIG Settings and Instance	
ALTGX Setting	**ALTGX Instance 1 (Basic Functional Mode, Receiver and Transmitter Operation Mode)**	**ALTGX_RECONFIG Setting**	**ALTGX_RECONFIG Instance 1**
`reconfig_togxb [3:0]` input	Connect the `reconfig_togxb[3:0]` signals between the ALTGX instance 1 and ALTGX_RECONFIG instance 1.	`reconfig_togxb[3:0]` output	Connect the `reconfig_togxb[3:0]` signals between the ALTGX instance 1 and ALTGX_RECONFIG instance 1.
Channel and Transmitter PLL Reconfiguration option in the **Reconfig** screen	■ Enable this option. ■ This is required to allow the ALTGX_RECONFIG instance to modify the transmitter channel local divider values dynamically.	**Use the 'rx_tx_duplex_sel' port to enable RX only, TX only, or duplex configuration** option in the **Error checks/Data rate switch** screen	This setting is optional and not required in the Data Rate Division in TX mode.

The alternate reference clock is not required because one clock source is used. Also, all data rates can be derived from the 106.25 MHz clock.

Create the ALTGX_RECONFIG instance control logic, reset control logic, and the core fabric logic to handle the data path. For more information about transceiver resets, refer to "Error Indication in the ALTGX_RECONFIG MegaWizard Plug-In Manager" on page 2–139.

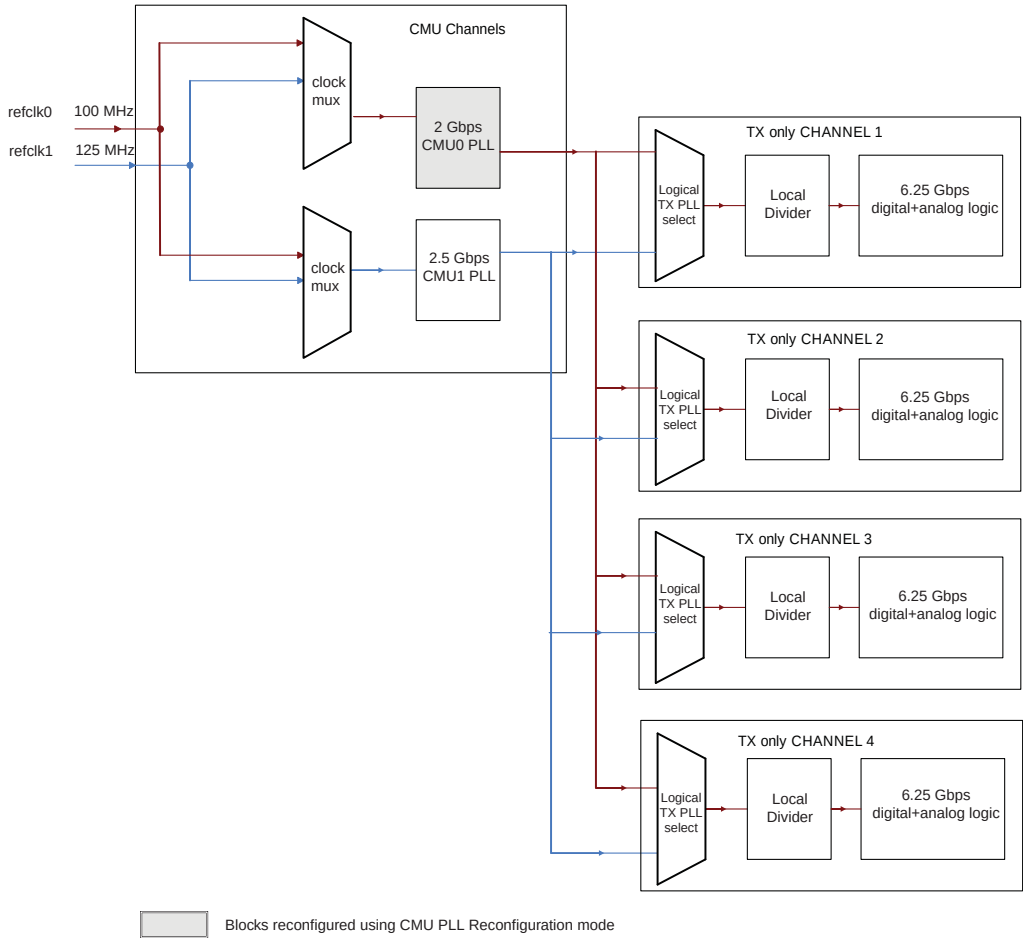
Example 5: CMU PLL Reconfiguration Mode with ALTGX Instances in Transmitter Only Configuration

Consider the following scenario:

- The design has four **Transmitter only** ALTGX instances in the same transceiver block.
- All four instances are configured in Basic functional mode and 2.5 Gbps data rate.
- All four channels can listen to CMU0 PLL.
- The input reference clock used by CMU0 PLL is 100 MHz.
- You want to reconfigure all four channels identically to 2 Gbps together.

Figure 2–57 shows this scenario before and after dynamic reconfiguration.

Figure 2–57. Before and After Example of CMU PLL Reconfiguration



You can achieve this by reconfiguring the CMU0 PLL once to run for 2 Gbps. This, in turn, changes the transmit data rate of all four channels listening to this CMU0 PLL.

You must enable **.mif** generation. Change the settings shown in Table 2–36 and save them in the **.mif**.

Table 2–36. CMU PLL Reconfiguration Scenario (Part 1 of 2)

ALTGX Instances		**ALTGX_RECONFIG Instance**	
ALTGX Setting	**Four TX Only Instances**	**ALTGX_RECONFIG Setting**	**ALTGX_RECONFIG Instance**
What is the effective data rate? option	2 Gbps	**Channel and TX PLL select/reconfig** option	Enabled

Table 2–36. CMU PLL Reconfiguration Scenario (Part 2 of 2)

ALTGX Instances		**ALTGX_RECONFIG Instance**	
ALTGX Setting	**Four TX Only Instances**	**ALTGX_RECONFIG Setting**	**ALTGX_RECONFIG Instance**
Enable Channel and Transmitter PLL Reconfiguration option	Enabled	**Use 'reconfig_address_en'** option	Enabled
Enable Channel and Transmitter PLL Reconfiguration option	Enabled		
What is the main transmitter PLL reference index? option	1		
How many input clocks? option	2		
What is the selected input clock source for the Transmitter PLL and Receiver PLL? option	1		
What is clock 0 input frequency? option	125 MHz		

After generating the **.mif**, follow the steps listed in "CMU PLL Reconfiguration Operation" on page 2–108 to write all the words.

Example 6: Dynamically Reconfiguring a Transceiver Channel between a GIGE Configuration and a SONET/SDH OC48 Configuration

The ALTGX MegaWizard Plug-In Manager settings—for example, data path, clocking, and core fabric-Transceiver interface width—are different for the GIGE configuration versus the SONET/SDH OC48 configuration. The differences between the two configurations are listed in Table 2–36.

Table 2–37. Differences between GIGE and SONET/SDH OC48 (Part 1 of 2)

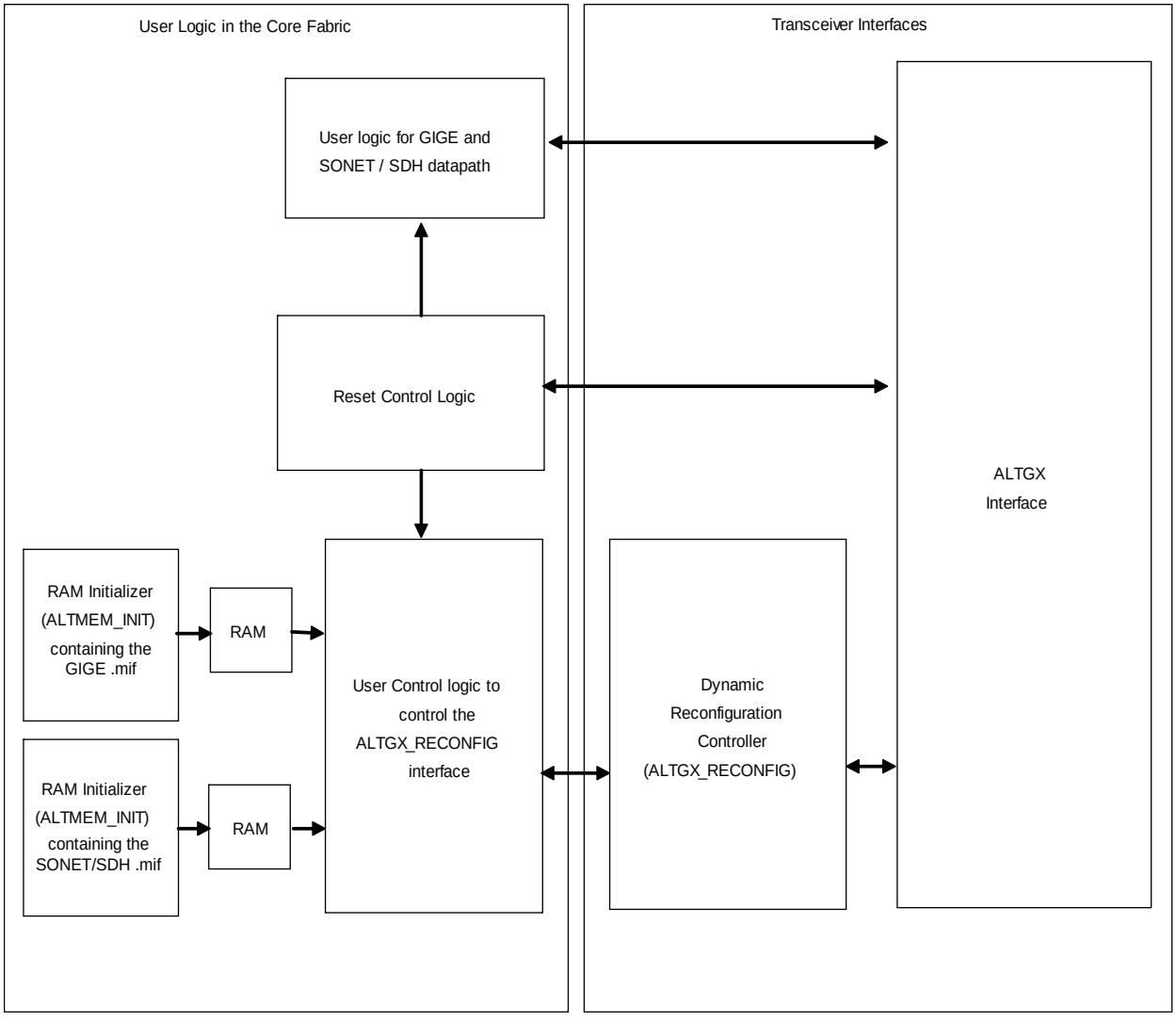
Number	**Functional Block**	**GIGE**	**SONET/SDH OC48**
1	Core fabric-Transceiver interface width	8	16
2	8B/10B enabled	Yes	No
3	Rate matcher enabled	Yes	No
4	Byte ordering block enabled	No	Yes
5	Clock used for synchronizing the receive output data (`tx_dataout`)	`tx_clkout` (because rate matcher is used)	`rx_clkout`
6	Data rate	1.25 Gbps	2.488 Gbps

Table 2–37. Differences between GIGE and SONET/SDH OC48 (Part 2 of 2)

Number	Functional Block	GIGE	SONET/SDH OC48
7	Allowed input reference clock frequencies	62.5 MHz 125 MHz	77.76 MHz 155.52 MHz 311.04 MHz 622.08 MHz
8	PCS-PMA interface width	10 (because data is 8B/10B encoded)	8

These differences determine the selection of parameters in the ALTGX MegaWizard Plug-In Manager and the required core array to dynamically reconfigure a transceiver channel between these two configurations. Figure 2–58 shows the required functional blocks to perform channel reconfiguration.

Figure 2–58. Dynamically Reconfiguring between GIGE and SONET/SDH OC48 Configurations



The description of the functional blocks is divided into four sections. The topics described in each section are as follows:

- Section I—Lists the steps to configure the ALTGX instance to generate the **.mif** for GIGE and SONET/SDH OC48 configurations. It also lists the steps to create the ALTGX_RECONFIG instance.
- Section II—Sets up the user control logic for the ALTGX_RECONFIG controller.
- Section III—Explains the logic to process the GIGE and SONET/SDH data. This logic is required due to the differences in the data interface widths and the clocking between the two configurations (shown in Table 2–36).
- Section IV—Explains how to reset the user control logic to control the transceiver channel and system resets.

Section I—Configure the ALTGX Instance to Generate the .mif

Use the following steps to generate a **.mif** for GIGE and SONET/SDH OC48 configurations:

1. Set up the ALTGX MegaWizard Plug-In Manager with the desired settings for the GIGE configuration. Generate the ALTGX instance for the GIGE configuration. Name this instance "GIGE_GXB". Table 2–38 explains the various options to set in the GIGE_GXB instance.
2. Generate the ALTGX_RECONFIG instance to control the GIGE_GXB instance.
3. Create a top-level design only with the GIGE_GXB instance. Enable the Quartus II settings to generate a **.mif**. Generate the **.mif** for the GIGE configuration. Compile the top-level design. The Quartus II software generates a **.mif** for the GIGE configuration. The *GIGE_GXB*.**mif** contains the desired GIGE configuration settings.
4. Modify the ALTGX instance with the desired settings for the SONET/SDH OC48 configuration. Name this instance "SONET_GXB".
5. Create a top-level design only with the SONET_GXB instance. Enable the Quartus II settings to generate a **.mif**. Compile the top-level design. The Quartus II software generates a **.mif** for the SONET/SDH OC4 configuration. The *SONET_GXB*.**mif** contains the desired SONET/SDH configuration settings.
6. Initialize two memory elements with the **.mif** contents and write user logic to select the appropriate **.mif** to use with the ALTGX_RECONFIG instance.

Table 2–38 shows the ALTGX instance settings for a single regular transceiver channel in GIGE configuration.

Table 2–38. Step 1: Generate the ALTGX Instance for the GIGE Configuration (Part 1 of 3)

ALTGX MegaWizard Plug-In Manager Option	Setting
General Screen	
Which protocol will you be using? option	Set the protocol to **GIGE** configuration.

Table 2–38. Step 1: Generate the ALTGX Instance for the GIGE Configuration (Part 2 of 3)

ALTGX MegaWizard Plug-In Manager Option	Setting
PLL/Ports screen	
In the **Optional Ports** section:	Select the following control and status signals: ■ `rx_digitalreset` ■ `tx_digitalreset` ■ `rx_analogreset` ■ `rx_pll_locked` ■ `rx_freqlocked` Add the other required status signals (For a list of ALTGX signals and their functionality, refer to the Transceiver Port List in the *HardCopy IV GX Transceiver Architecture* chapter in volume 3 of the *HardCopy IV Device Handbook*.)
Reconfig Screen	
Analog Controls (VOD, Pre-emphasis, and Manual Equalization) option	If you need control of the transceiver PMA controls, select **Analog PMA controls**. For more information about PMA controls, refer to "PMA Controls Reconfiguration" on page 2–52.
Enable Channel and Transmitter PLL Reconfiguration option	Because you need to reconfigure the transceiver channel from a GIGE configuration to a SONET/SDH OC48 configuration, select this option.
Channel Interface option	Selecting this option creates the data interface signals `tx_datainfull[43:0]` and `rx_dataoutfull[63:0]` that are comprised of control and data signals. This selection is required because the core fabric-Transceiver Interface is different for a GIGE configuration versus a SONET/SDH configuration (Refer to row 8 in Table 2–37 on page 2–131). The description of the individual bits of `tx_datainfull[43:0]` and `rx_datainfull[63:0]` are provided in Table 2–28 on page 2–91 and Table 2–29 on page 2–94.
Use alternate Transmitter PLL option	Selecting this option enables the second PLL for the SONET/SDH OC48 configuration. A second PLL is needed because of the difference in the required input clock frequency and data rate between the GIGE and SONET/SDH OC48 configurations (Refer to rows 6 and 7 in Table 2–37 on page 2–131).
What is the protocol to be reconfigured to? option	Set this option to **SONET/SDH**.
What is the subprotocol to be reconfigured to? option	Set **sub protocol** to **OC48**.
What is the input clock frequency? and **What is the alternate Transmitter PLL bandwidth mode?** options	Select the **input clock frequency** and **alternate transmitter PLL bandwidth mode** options based on the requirements. The allowed reference clock input frequencies for SONET/SDH OC48 are specified in row 7 of Table 2–37 on page 2–131.

Table 2–38. Step 1: Generate the ALTGX Instance for the GIGE Configuration (Part 3 of 3)

ALTGX MegaWizard Plug-In Manager Option	Setting
What is the alternate transmitter PLL logical reference index? option	For the logical reference index option, select **1** or **0**. The Quartus II software uses the logical reference index to select the PLL clock outputs for transmit and receive channels when configured to SONET/SDH OC48 protocol. The MUX values selected for the GIGE and SONET/SDH OC48 modes must be different. For example, if you select a logical reference index of **1** for the SONET/SDH OC48 configuration, you need to select **0** for the GIGE configuration. If you select the same values for the two configurations, the transceiver behavior after reconfiguration becomes unpredictable.
Reconfig 2 Screen	
How should the receivers be clocked? option	Select the **Use the respective channel core clocks** option. Selecting this option creates the `rx_clkout` port. Select this option because of the clocking differences between the two modes (row 5 of Table 2–37 on page 2–131). The core fabric logic can clock the receiver output of the ALTGX instance with `rx_clkout` for SONET/SDH mode and `tx_clkout` for GIGE mode.
How should the transmitters be clocked? option	Select any option. Because this example assumes a one channel reconfiguration in the transceiver block, the preceding options will not make a difference. However, if the number of channels used in channel reconfiguration is more than one, Altera recommends you select the **Share single transmitter core clock between transmitters** option to conserve clock routing resources.
Check a control box to use the corresponding control port: option	Select signals in this option based on the requirements. The signals in this tab can be selected only if the **Channel interface** option is enabled in the **Reconfig** screen. For this example, select the `rx_byteorderalignstatus` and `rx_a1a2sizeout` signals because these signals are required for SONET/SDH OC48 configuration. Some of the signals are meaningful only for the modes for which they are intended. For example, the `rx_byteorderalignstatus` signal is only meaningful in the SONET/SDH OC48 configuration. The core fabric logic does not use these signals for GIGE configuration. For more information about the protocol specific ALTGX interface signals, refer to the Transceiver Port List in the *HardCopy IV GX Transceiver Architecture* chapter in volume 3 of the *HardCopy IV Device Handbook*.
In the subsequent screens, select the required signals and complete the MegaWizard Plug-In Manager instantiation.	

Step 2—Create the ALTGX_RECONFIG Instance for the GIGE Configuration

1. Set the **What is number of channels controlled by the reconfig controller?** option to **1**.
2. Select **Analog controls** to modify the PMA values, if desired.
3. Select the **Channel reconfiguration with TX PLL select/reconfig** option. This selection is required to perform a channel reconfiguration.
4. Select the required signals under the write control and read control options, if the **Analog controls** option in **screen 1** is selected.

   ☞ Refer to the *ALTGX_RECONFIG Megafunction User Guide* chapter in volume 3 of the *HardCopy IV Device Handbook* for information about write control and read control signals.

5. Complete the ALTGX_RECONFIG MegaWizard Plug-In Manager instantiation.

Step 3—Create a Top-Level Design and Generate the .mif for the GIGE Configuration

Clock input connections for the ALTGX instance are listed below. The clock source needs to be feed the following clock inputs:

- GIGE configuration—`pll_inclk` and `rx_cruclk` inputs.
- SONET/SDH OC48 configuration—`pll_inclk_alt` and `rx_cruclk_alt` inputs.

1. Because GIGE is the protocol mode you selected in the first page of the ALTGX MegaWizard Plug-In Manager, the Quartus II software requires the GIGE clock source to be connected to the `pll_inclk` and `rx_cruclk` inputs.
2. Connect the `cal_blk_clk` input of the ALTGX instance to a clock source.

   ☞ For the `cal_blk_clk` signal requirements, refer to the Transceiver Port List in the *HardCopy IV GX Transceiver* Architecture chapter in volume 3 of the *HardCopy IV Device Handbook*.

3. Connect the `tx_dataout` and `rx_datain` ports to the top-level module. This is required for the Quartus II software to compile successfully. To generate the **.mif**, connecting the other input and output ports of the ALTGX instance is not mandatory.
4. Assign pins for the clock ports (`pll_inclk, rx_cruclk, pll_inclk_alt,` and `rx_cruclk_alt`). If pin assignments are not made for the `tx_dataout` and `rx_datain` ports of the ALTGX instance, the Quartus II software automatically selects pins for these ports and names the **.mif** with the instance name. The **.mif** can still be used by any physical transceiver channel to perform reconfiguration.

After compilation of the design, the Quartus II software creates the **.mif** in the *reconfig_mif* folder under the project directory. Copy the **.mif** and save it in a separate folder. Otherwise, the new **.mif** that is generated for the SONET/SDH configuration will overwrite the current **.mif**.

Step 4—Modify the ALTGX Instance for a SONET/SDH OC48 Configuration

1. To create a **.mif** for the SONET/SDH OC48 configuration, either modify the existing ALTGX instance created for the GIGE configuration or create a new instance for the SONET configuration. However, the first method is easier because it does not require major RTL or schematic changes.
2. Open the existing ALTGX instance. Select the **Which protocol you will be using?** option and set it to **SONET/SDH**. Set the **sub protocol** option to **OC48**. All the other signals selected for GIGE mode do not need to be changed.
3. In the **Reconfig** screen, select the **Channel Interface** and **alternate reference clock** options. In the **protocol** section, select **GIGE**. Select the same input clock frequency selected for the GIGE instance in the **General** screen.
4. For the **logical reference clock index** option, choose the complement of what you selected for the GIGE instance.
5. Complete the instantiation.

Step 5—Generate the .mif for the SONET/SDH OC48 Configuration

1. Before compiling the design, in the RTL or schematic, connect `pll_inclk` and `rx_cruclk` to the clock source that provides the SONET/SDH OC48 clock. Similarly, connect `pll_inclk_alt` and `rx_cruclk_alt` to the clock source that provides the GIGE clock.

The Quartus II software generates the new **.mif** in the **/reconfig_mif** directory.

Step 6—Initialize Two Memory Elements with the .mif Contents and Write Logic to Select the .mif and to Control the ALTGX_RECONFIG Instance

1. Assign the two **.mifs** to the ALTMEM_INIT megafunction in the MegaWizard Plug-In Manager to initialize each of the RAM. This megafunction reads from an internal ROM (inside the megafunction) or an external ROM (on-chip or off-chip), and writes to the RAM after power up.

HardCopy IV GX ASIC does not support pre-loading or initializing internal memory blocks with **.mif** when used as RAM. For more information, refer to *RAM Initializer (ALTMEM_INIT) Megafunction User Guide*.

Section II—Control the Logic for the Dynamic Reconfiguration Controller

The control logic block is required to perform the following functions:

- Select the memory to configure a channel to the GIGE or SONET/SDH configuration.
- Control the reconfiguration mode (namely the PMA controls reconfiguration mode or the Channel and CMU PLL reconfiguration mode).
- Control the read and write signals to the ALTGX_RECONFIG instance based on the busy and data valid signals.

Section III—Logic and Clocking for the GIGE and SONET/SDH OC48 Datapath

In the ALTGX MegaWizard Plug-In Manager, the channel interface that created tx_datainfull[43:0] and rx_dataoutfull[63:0] was selected. In addition, the rx_byteorderalignstatus and rx_a1a2size signals were selected. The core fabric selectively uses some of these signals based on whether the transceiver channel is configured in a GIGE configuration or a SONET/SDH OC48 configuration.

Table 2–39 provides descriptions for the tx_datainfull[43:0] and rx_dataoutfull[63:0] signals for GIGE and SONET/SDH OC48 configurations.

Table 2–39. Core Fabric-Transceiver Interface Signals—GIGE and SONET/SDH OC48 Configurations

SIgnal Name	Description
GIGE Configuration	
tx_datainfull[7:0]	8-bit unencoded data input to the transceiver channel
tx_datainfull[8]	tx_ctrlenable (control signal K/D)
rx_dataoutfull[7:0]	8-bit unencoded data output from the transceiver channel
rx_dataoutfull[8]	rx_ctrldetect (control signal K/D)
rx_dataoutfull[9]	rx_errdetect
rx_dataoutfull[10]	rx_syncstatus
rx_dataoutfull[11]	rx_disperr
rx_dataoutfull[12]	rx_patterndetect
SONET/SDH OC48 Configuration	
tx_datainfull[7:0]	LSB data input to the transceiver channel
tx_datainfull[29:22]	MSB data input to the transceiver channel
rx_dataoutfull[7:0]	LSB data output from the transceiver channel
rx_dataoutfull[29:22	MSB data output from the transceiver channel
rx_dataoutfull[10], rx_dataoutfull[42]	rx_syncstatus[1:0]
rx_dataoutfull[12], rx_dataoutfull[44]	rx_patterndetect[1:0]

Clocking

For the transmit side, the core fabric user logic for the SONET/SDH OC48 and GIGE configurations sends the data synchronized to the tx_clkout signal. Therefore, the clocking for the transmit side remains the same for the two modes.

For the receive side, the data and status signals from the ALTGX instance for the GIGE configuration is synchronized to tx_clkout because rate matching is used. For the SONET/SDH OC48 configuration, the signals are synchronized to rx_clkout. Therefore, the core fabric user logic has two functional protocol specific logic blocks to handle data for the GIGE and SONET/SDH OC48 configurations. Based on the configured protocol mode, the receive side logic selects the appropriate data path.

Section IV—Reset Control Logic

The reset control sequence for channel reconfiguration must be followed during and after the channel configuration process. In addition, when resetting the transceiver channel, the reset control logic needs to reset the data path in the core fabric to clear the error data received during the reconfiguration process.

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

For a PMA controls only configuration (for example, changing the VOD, equalization, DC gain, or pre-emphasis), the transceiver channel or the datapath in the core fabric does not require a reset after reconfiguration. Reset is required only for channel reconfiguration or rate switch.

Simulation

To simulate channel reconfiguration, some simulation tools only allow **.ram** or Hexadecimal (Intel-Format) file (**.hex**) files to initialize the memory. To convert the generated **.mif** to a **.hex** file, open the **.mif** in the Quartus II software and save it as a **.hex** file. Initialize the memory elements with the **.hex** file to simulate the design.

Error Indication in the ALTGX_RECONFIG MegaWizard Plug-In Manager

The ALTGX_RECONFIG MegaWizard Plug-In Manager provides an `error` status signal when you select the **Enable illegal mode checking** option or the **Enable self recovery** option in the **Error checks/data rate switch** screen. The conditions under which the `error` signal is asserted are:

- **Enable illegal mode checking** option—When you select this option, the dynamic reconfiguration controller checks whether an attempted operation falls under one of the conditions listed below. The dynamic reconfiguration controller detects these conditions within two `reconfig_clk` cycles, de-asserts the `busy` signal, and asserts the `error` signal for two `reconfig_clk` cycles.
  - PMA controls, read operation—None of the output ports (`rx_eqctrl_out`, `rx_eqdcgain_out`, `tx_vodctrl_out`, `tx_preemp_0t_out`, `tx_preemp_1t_out`, and `tx_preemp_2t_out`) are selected in the ALTGX_RECONFIG instance. The `read` signal is asserted.
  - PMA controls, write operation—None of the input ports (`rx_eqctrl`, `rx_eqdcgain`, `tx_vodctrl`, `tx_preemp_0t`, `tx_preemp_1t`, and `tx_preemp_2t`) are selected in the ALTGX_RECONFIG instance. The `write_all` signal is asserted.
- **Enable self recovery** option—When you select this option, the controller automatically recovers if the operation did not complete within the expected time. The `error` signal is driven high whenever the controller performs a self recovery.
- **TX Data Rate Switch using Local Divider-read operation** option—The read transaction is valid only for the following two dynamic reconfiguration modes:
  - PMA controls reconfiguration mode
  - TX Data Rate switch using Local Divider mode

- **TX Data Rate Switch using Local Divider-write operation with unsupported value** option:
  - The `rate_switch_ctrl[1:0]` input port is set to **11**
  - The `reconfig_mode_sel[2:0]` input port is set to **4** (if other reconfiguration mode options are selected in the **Reconfiguration settings** screen)
  - The `write_all` is asserted
- **TX Data Rate Switch using Local Divider-write operation without input port** option:
  - The `rate_switch_ctrl[1:0]` input port is not used
  - The `reconfig_mode_sel[2:0]` port is set to **4** (if other reconfiguration mode options are selected in the **Reconfiguration settings** screen)
  - The `write_all` is asserted
- **TX Data Rate Switch using Local Divider- read operation without output port** option:
  - The `rate_switch_out[1:0]` output port is not used
  - The `reconfig_mode_sel[2:0]` port is set to **4** (if other reconfiguration mode options are selected in the **Reconfiguration settings** screen)
  - The `read` is asserted
- **Channel and/or TX PLL reconfiguration-read operation** option:
  - The `reconfig_mode_sel[2:0]` input port is set to **1**, **4**, **5**, or **6**
  - The `read` signal is asserted

Combining Transceiver Channels with Dynamic Reconfiguration Enabled

Packing the transceiver channels into the same physical transceiver block is called "combining". You can combine the transceiver channels in a design into the same physical transceiver block by assigning the `tx_dataout` and `rx_datain` pins of the channels to the same transceiver block.

The Quartus II software also allows you to combine multiple channels into the same physical transceiver block based on the same requirements described in the following sections.

Requirements

When you enable dynamic reconfiguration, the Quartus II software has certain requirements for combining multiple transceiver channels in the same physical transceiver block:

- All the channels that you want to combine into the same transceiver block must have the same options enabled in the **Reconfig** screen of the ALTGX MegaWizard Plug-In Manager. For example, when you enable the **Analog controls (VOD, Pre-emphasis, and Manual Equalization)** option in the **Reconfig** screen of the ALTGX MegaWizard Plug-In Manager for a channel, you need to enable the same option for all the other channels to be combined.
- All the channels must be controlled by the same ALTGX_RECONFIG (dynamic reconfiguration controller) instance. The transceiver channels connected to multiple ALTGX_RECONFIG instances cannot be combined into the same physical transceiver block, even if they are configured to the same functional mode and data rate.

The preceding two constraints are mandatory to combine transceiver channels into the same transceiver block.

For additional requirements to combine the ALTGX instances within the same transceiver block or in transceiver blocks on the same side of the device, refer to the *Configuring Multiple Protocols and Data Rates* chapter in volume 2 of the *Stratix IV Device Handbook*.

Combining a Transmitter Only Instance and Receiver Only Instance

Consider that you want to combine one **Receiver only** instance and another **Transmitter only** instance in the same transceiver block:

- The **Receiver only** instance must be controlled by an ALTGX_RECONFIG instance for offset cancellation control.
- Because you want to combine the **Receiver only** instance with another **Transmitter only** instance into the same transceiver block, you must control the **Transmitter only** instance using the same ALTGX_RECONFIG instance.
- You must enable the same options in the **Reconfig** screen of the ALTGX MegaWizard Plug-In Manager for both the **Transmitter only** and **Receiver only** instances.

There are constraints with the independent **Transmitter only** and independent **Receiver only** configurations. Both transmitter and receiver have to go through a reset sequence, even if the transmitter or receiver is reconfigured.

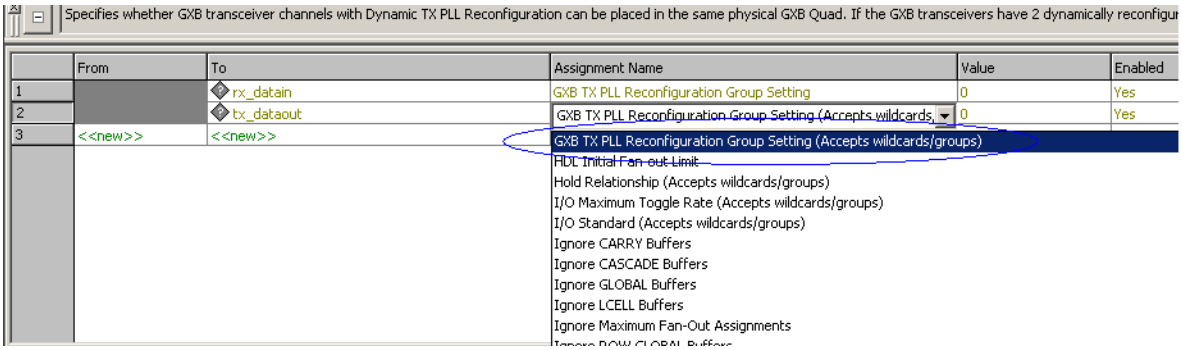
Merging Transceiver Channels with Channel and CMU PLL Reconfiguration Mode Enabled into the Same Transceiver Block

The Quartus II software requires the following Assignment Editor setting for all channels assigned to the same transceiver bank when you enable the **Channel and CMU PLL reconfiguration** option.

- Assignment setting: Assignment Name - GXB TX PLL Reconfiguration group setting (as shown in Figure 2–59).

If you have more than one channel with the **Channel and CMU PLL Reconfiguration** option enabled, and if you assign them to different reconfiguration groups without pin assignments for the `tx_dataout` pins, the Quartus II software automatically assigns these channels to different transceiver blocks. If you use a HardCopy IV GX device with one transceiver block, you cannot compile the design if you assign different TX PLL reconfig group values for the channels in your design.

Figure 2–59. Reconfiguration Group Setting Required for Channel and CMU PLL Reconfiguration



To understand this assignment setting, assume that you have two transmit channels in the same transceiver bank with the **Channel and CMU PLL Reconfiguration** option enabled. If the transmit output pins are `tx_dataout_ch0` and `tx_dataout_ch1`, set the assignments shown in Table 2–40 and Table 2–41 to compile the design.

Table 2–40. tx_dataout_ch0 Reconfiguration Assignment Settings

Assignment	Setting
To:	`tx_dataout_ch0`
Assignment Name:	GXB TX PLL Reconfiguration group setting
Value:	0

Table 2–41. tx_dataout_ch1 Reconfiguration Assignment Settings

Assignment	Setting
To:	`tx_dataout_ch1`
Assignment Name:	GXB TX PLL Reconfiguration group setting
Value:	0

Merging Transceiver Channels Listening to Two Transmitter PLLs

Consider that you create an ALTGX instance for a **Receiver and Transmitter** or **Transmitter Only** configuration that has a main and alternate transmitter PLL. If you want to place other channels in the same transceiver bank, the other channels need to also have a main and alternate transmitter PLL option to merge successfully.

For example, consider that you create the following instances:

- ALTGX Instance 1: one channel with a **Receiver and Transmitter** configuration, with the main transmitter PLL (assume a logical tx pll value of **0**), configured to 6.25 Gbps data rate and the alternate transmitter PLL configured to 2.500 Gbps.

Assume that you create another instance with the following configuration:

- ALTGX Instance 2: one channel with a **Receiver and Transmitter** configuration, with only one transmitter PLL (assume a logical tx pll value of **0**), configured to 6.25 Gbps.

In this case, you cannot merge ALTGX instance 1 and ALTGX instance 2 in the same transceiver bank because ALTGX instance 2 listens to only one transmitter PLL. To successfully merge the two instances, create ALTGX instance 2 with an alternate transmitter PLL configured to 2.500 Gbps.

Merging Transceiver Channels Listening to One Transmitter PLL

Consider that you create an ALTGX instance (**Receiver and Transmitter** or **Transmitter Only** configuration) that has only one transmitter PLL. If you want to create another ALTGX instance configured at a different data rate in the same transceiver bank, provide different logical tx pll values for the two instantiations.

For example, to merge the following instantiations in the same transceiver bank:

- ALTGX Instance 1: one channel with a **Receiver and Transmitter** configuration, configured at 3.125 Gbps data rate.
- ALTGX Instance 2: one channel with a **Receiver and Transmitter** configuration, configured at 2.500 Gbps. If you set the **What is the main transmitter PLL logical reference index?** option (in the **Reconfig Clks** screen) for ALTGX instance 1 to 0, set this option to **1** for ALTGX instance 2. The two ALTGX instances need to have different logical tx pll values because the Quartus II software requires separate transmitter PLLs for these two channels.

Dynamic Reconfiguration Duration and Core Fabric Resource Utilization

This section describes the time taken for dynamic reconfiguration transactions and core fabric resources used by the dynamic reconfiguration controller when used in different modes of reconfiguration.

Dynamic Reconfiguration Duration

Dynamic reconfiguration duration is the number of cycles for which the `busy` signal is asserted when the dynamic reconfiguration controller performs write transactions, read transactions, or offset cancellation of the receiver channels.

PMA Controls Reconfiguration Duration

The following section contains an estimate of the number of `reconfig_clk` clock cycles for which the `busy` signal is asserted during PMA controls reconfiguration using Method 1 and Method 2. For more information, refer to "Dynamically Reconfiguring PMA Controls" on page 2–53.

PMA Controls Reconfiguration Duration When Using Method 1

Use the `logical_channel_address` port in Method 1. The write transaction and read transaction duration is as follows:

Write Transaction Duration

For writing values to the following PMA controls, the `busy` signal is asserted for 260 `reconfig_clk` clock cycles for each of these controls:

- `tx_preemp_1t` (pre-emphasis control first post-tap)
- `tx_vodctrl` (voltage output differential)
- `rx_eqctrl` (equalizer control)
- `rx_eqdcgain` (equalizer DC gain)

For writing values to the following PMA controls, the `busy` signal is asserted for 520 `reconfig_clk` clock cycles for each of these controls:

- `tx_preemp_0t` (pre-emphasis control pre-tap)
- `tx_preemp_2t` (pre-emphasis control second post-tap)

Read Transaction Duration

For reading the existing values of the following PMA controls, the `busy` signal is asserted for 130 `reconfig_clk` clock cycles for each of these controls. The `data_valid` signal is then asserted once the `busy` signal goes low.

- `tx_preemp_1t_out` (pre-emphasis control first post-tap)
- `tx_vodctrl_out` (voltage output differential)
- `rx_eqctrl_out` (equalizer control)
- `rx_eqdcgain_out` (equalizer DC gain)

For reading the existing values of the following PMA controls, the `busy` signal is asserted for 260 `reconfig_clk` clock cycles for each of these controls. The `data_valid` signal is then asserted once the `busy` signal goes low.

- `tx_preemp_0t_out` (pre-emphasis control pre-tap)
- `tx_preemp_2t_out` (pre-emphasis control second post-tap)

PMA Controls Reconfiguration Duration When Using Method 2

The `logical_channel_address` port is not used in Method 2. The write transaction duration and read transaction duration are as follows:

Write Transaction Duration

For writing values to the following PMA controls, the `busy` signal is asserted for 260 `reconfig_clk` clock cycles per channel for each of these controls:

- `tx_preemp_1t` (pre-emphasis control first post-tap)
- `tx_vodctrl` (voltage output differential)
- `rx_eqctrl` (equalizer control)
- `rx_eqdcgain` (equalizer DC gain)

For writing values to the following PMA controls, the busy signal is asserted for 520 reconfig_clk clock cycles per channel for each of these controls:

- tx_preemp_0t (pre-emphasis control pre-tap)
- tx_preemp_2t (pre-emphasis control second post-tap)

Read Transaction Duration

For reading the existing values of the following PMA controls, the busy signal is asserted for 130 reconfig_clk clock cycles per channel for each of these controls. The data_valid signal is then asserted once the busy signal goes low.

- tx_preemp_1t_out (pre-emphasis control first post-tap)
- tx_vodctrl_out (voltage output differential)
- rx_eqctrl_out (equalizer control)
- rx_eqdcgain_out (equalizer DC gain)

For reading the existing values of the following PMA controls, the busy signal is asserted for 260 reconfig_clk clock cycles per channel for each of these controls. The data_valid signal is then asserted once the busy signal goes low.

- tx_preemp_0t_out (pre-emphasis control pre-tap)
- tx_preemp_2t_out (pre-emphasis control second post-tap)

Offset Cancellation Duration

When the device powers up, the busy signal remains low for the first reconfig_clk clock cycle. After the device powers up, it takes 70 reconfig_clk clock cycles for the dynamic reconfiguration controller to identify the receiver channels.

When the dynamic reconfiguration controller identifies the receiver channels and verifies the logical channel address to physical channel mapping, it takes another 7872 reconfig_clk clock cycles per receiver channel to perform the offset cancellation process. In other words, the busy signal goes low after 7924 reconfig_clk clock cycles per receiver channel (50 + 2 + 7872).

If the design does not require PMA controls reconfiguration, each ALTGX instance in the design can have its own dynamic reconfiguration controller (ALTGX_RECONFIG instance). This minimizes offset cancellation duration.

Dynamic Reconfiguration Duration for Channel and TX PLL Select/Reconfig Modes

Table 2–42 shows the number of reconfig_clk clock cycles it takes for the dynamic reconfiguration controller to reconfigure various parts of the transceiver channel and CMU PLL.

Table 2–42. Dynamic Reconfiguration Duration for Transceiver Channel and CMU PLL Reconfiguration (Part 1 of 2)

Transceiver Portion Under Reconfiguration	Number of reconfig_clk Clock Cycles
Transmitter channel reconfiguration	1690 clock cycles
Receiver channel reconfiguration	5181 clock cycles

Table 2–42. Dynamic Reconfiguration Duration for Transceiver Channel and CMU PLL Reconfiguration (Part 2 of 2)

Transceiver Portion Under Reconfiguration	Number of reconfig_clk Clock Cycles
Transmitter and receiver channel reconfiguration	6861 clock cycles
CMU PLL only reconfiguration	970 clock cycles
Transmitter channel and CMU PLL reconfiguration	2650 clock cycles
Transceiver channel and CMU PLL reconfiguration	7850 clock cycles

Dynamic Reconfiguration (ALTGX_RECONFIG Instance) Resource Utilization

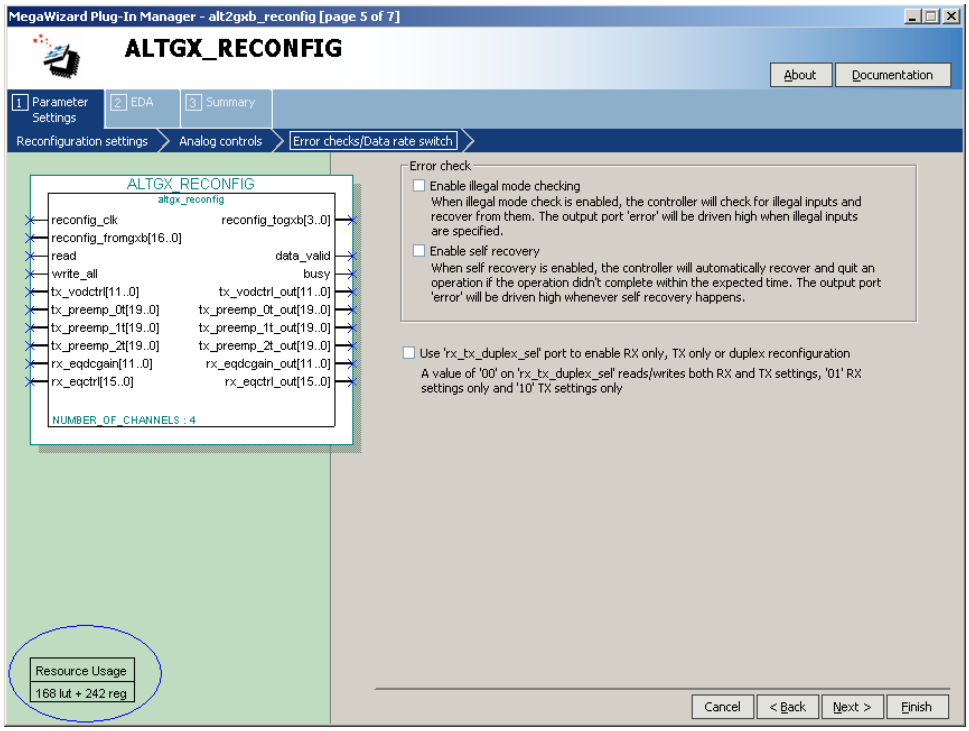
You can observe the resources utilized during dynamic reconfiguration in the ALTGX_RECONFIG MegaWizard Plug-In Manager. This section contains an estimate of the logic elements (LE) resources utilized during dynamic reconfiguration.

You can obtain resource utilization for all other PMA controls from the ALTGX_RECONFIG MegaWizard Plug-In Manager.

For example, the number of LEs used by one dynamic reconfiguration controller is 43 with only `tx_vodctrl` selected. Similarly, the number of registers is 130.

Figure 2–60 shows the resource utilization in the ALTGX_RECONFIG MegaWizard Plug-In Manager.

Figure 2–60. Resource Utilization in the ALTGX_RECONFIG MegaWizard Plug-In Manager



Functional Simulation of the Offset Cancellation Process

This section contains the points to be considered during the functional simulation of the dynamic reconfiguration process.

You must connect the ALTGX_RECONFIG instance to the ALTGX_instance/ALTGX instances in your design for functional simulation. The functional simulation uses a reduced timing model of the dynamic reconfiguration controller. Therefore, the duration of the dynamic reconfiguration process is 16 `reconfig_clk` clock cycles for functional simulation only. The `gxb_powerdown` signal must not be asserted during the offset cancellation sequence (for functional simulation and silicon).

Document Revision History

Table 2–43 shows the revision history for this chapter.

Table 2–43. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
June 2009, v1.0	Initial release.	—

3. HardCopy IV GX ALTGX_RECONFIG Megafunction User Guide

HIV53003-1.0

Introduction

The MegaWizard™ Plug-In Manager in the Quartus® II software creates or modifies design files that contain custom megafunction variations. These auto-generated MegaWizard Plug-In Manager files can then be instantiated in a design file. The MegaWizard Plug-In Manager allows you to specify options for the ALTGX_RECONFIG megafunction.

Start the MegaWizard Plug-In Manager using one of the following methods:

- On the Tools menu, choose the **MegaWizard Plug-In Manager** command.
- When working in the **Block Editor** (schematic symbol), open the Edit menu and choose **Insert Symbol**. The **Symbol** dialog box appears. In the **Symbol** dialog box, click **MegaWizard Plug-In Manager**.
- Start the stand-alone version of the MegaWizard Plug-In Manager by typing the following command at the command prompt: `qmegawiz`.

Dynamic Reconfiguration

This section describes the options available on the individual pages of the ALTGX_RECONFIG MegaWizard Plug-In Manager.

The MegaWizard Plug-In Manager provides a warning if any of the settings you choose are illegal.

Figure 3–1 shows the first page of the MegaWizard Plug-In Manager. To generate an ALTGX_RECONFIG custom megafunction variation, select **Create a new custom megafunction variation**. Click **Next**.

Figure 3–1. MegaWizard Plug-In Manager (Page 1)

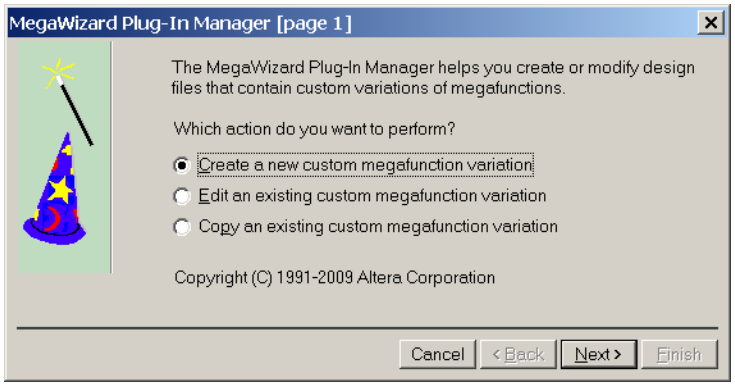


Figure 3–2 shows the second page of the MegaWizard Plug-In Manager. Select the following options (click **Next** when you are done):

1. In the list of megafunctions on the left, click the "+" icon beside the I/O item. From the options presented, choose **ALTGX_RECONFIG megafunction**.
2. From the drop-down menu beside **Which device family will you be using?**, select **Stratix IV**. Note that the project target device is Stratix IV GX and a HardCopy IV GX companion device must be selected.
3. From the radio buttons under **Which type of output file do you want to create?**, choose your output file format (**AHDL**, **VHDL**, or **Verilog HDL**).
4. In the box beneath **What name do you want for the output file?**, enter the file name or click the **Browse** button to search for it.

☞ For the design to compile successfully, always enable the dynamic reconfiguration controller for all the ALTGX instances in the design.

Figure 3–2. MegaWizard Plug-In Manager—ALTGX_RECONFIG (Page 2)

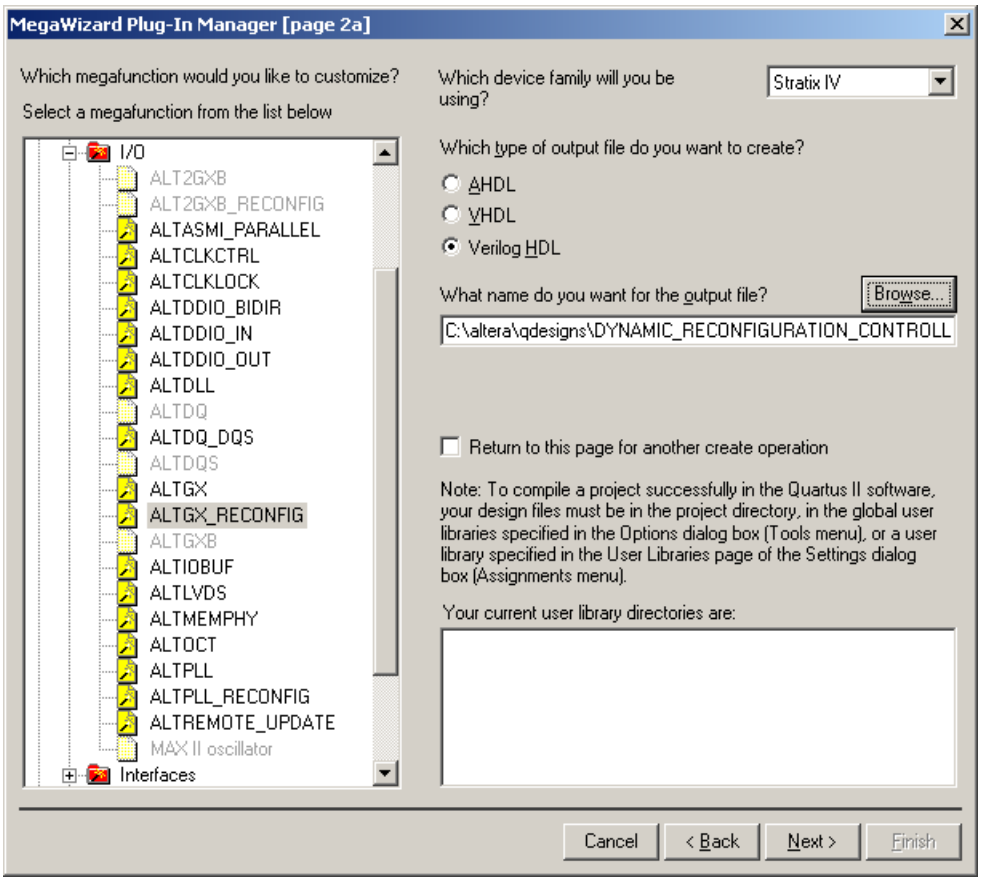


Figure 3–3 shows page 3 of the ALTGX_RECONFIG MegaWizard Plug-In Manager. From the drop-down menu, select the number of channels controlled by the dynamic reconfiguration controller.

Figure 3–3. MegaWizard Plug-In Manager—ALTGX_RECONFIG (Reconfiguration Settings) (Page 3)

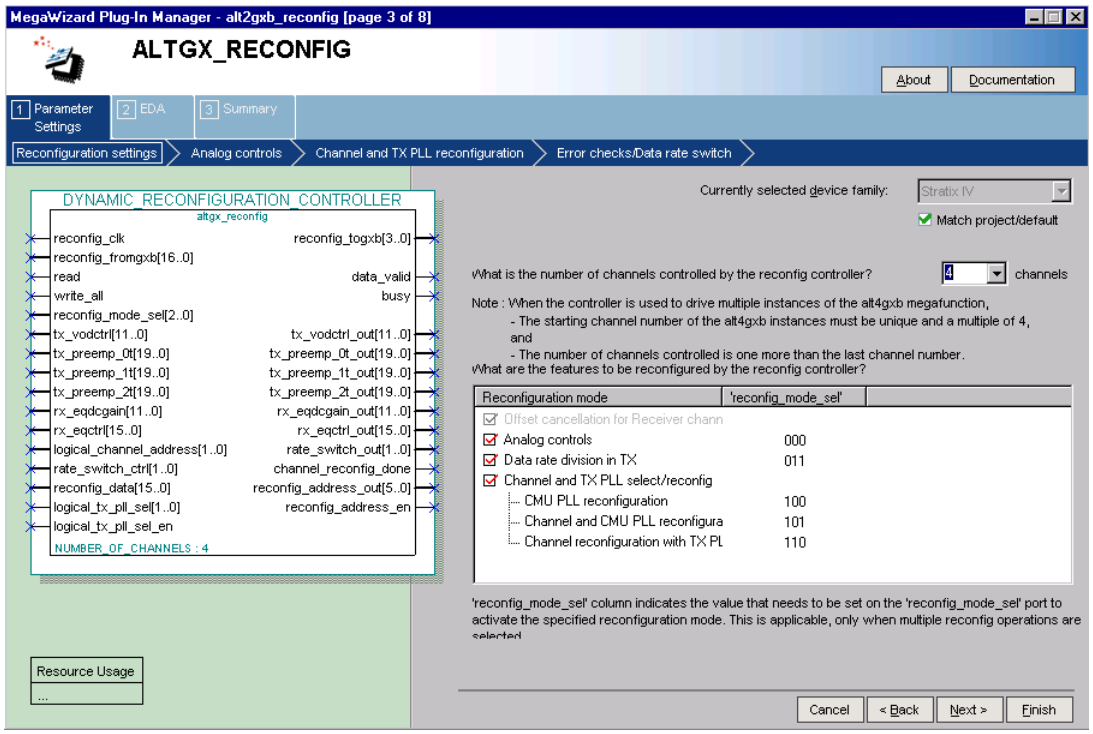


Table 3–1 describes the available options on page 3 of the MegaWizard Plug-In Manager for your ALTGX_RECONFIG custom megafunction variation. Select the **Match project/default** option if you want to change the device **Currently selected device family** options.

Make your selections on page 3, then click **Next**.

Table 3–1. MegaWizard Plug-In Manager Options (Page 3) (Part 1 of 2)

ALTGX_RECONFIG Setting	Description	Reference
What is the number of channels controlled by the reconfig controller?	Determine the highest logical channel address amongst all the ALTGX instances connected to the ALTGX_RECONFIG instance. Round it up to the next multiple of four and set that number in this option. Depending on this setting, the ALTGX_RECONFIG MegaWizard Plug-in Manager generates the appropriate signal width for the interface signal (`reconfig_fromgxb`) between the ALTGX_RECONFIG and ALTGX instances. It also gives the necessary bus width for all the selected physical media attachment (PMA) signals. Depending on the number of channels set, the resource estimate changes because this is a soft implementation that uses fabric logic resources. The resource estimate is shown in the bottom left of Page 3 of the MegaWizard Plug-in Manager.	"Total Number of Channels Controlled by the ALTGX_RECONFIG Instance" section of the *HardCopy IV GX Dynamic Reconfiguration* chapter in volume 3 of the *HardCopy IV Device Handbook*

Table 3–1. MegaWizard Plug-In Manager Options (Page 3) (Part 2 of 2)

ALTGX_RECONFIG Setting	Description	Reference
What are the features to be reconfigured by the reconfig controller?	This feature is always enabled by default: ■ **Offset Cancellation for Receiver Channels**—After the device powers up, the dynamic reconfiguration controller performs offset cancellation on the receiver portion of all the transceiver channels controlled by it.	"Offset Cancellation" section of the *HardCopy IV GX Dynamic Reconfiguration* chapter in volume 3 of the *HardCopy IV Device Handbook*
	This feature are available for selection: ■ **Analog Controls**—Allows dynamic reconfiguration of PMA controls such as Equalization, Pre-emphasis, DC Gain, and VOD. ■ **Data rate division in TX**—Allows dynamic reconfiguration of the transmitter local divider settings to 1,2, or 4. The transmitter channel data rate is reconfigured based on the local divider settings. ■ **Channel and TX PLL select/reconfig**—The following features are available under this option: → **CMU PLL Reconfiguration**—Allows the dynamic reconfiguration of the CMU PLL to a different data rate. → **Channel and CMU PLL reconfiguration**—Allows the dynamic reconfiguration of the transceiver channel from one functional mode to another and also the reconfiguration of the CMU PLL. → **Channel reconfiguration with TX PLL select**—Allows you to select another transmitter PLL for the transceiver channel and reconfigure the channel to another data rate.	"PMA Controls Reconfiguration" section of the *HardCopy IV GX Dynamic Reconfiguration* chapter in volume 3 of the *HardCopy IV Device Handbook* "CMU PLL Reconfiguration Mode" section of the *HardCopy IV GX Dynamic Reconfiguration* chapter in volume 3 of the *HardCopy IV Device Handbook* "Channel and CMU PLL Reconfiguration Mode" section of the *HardCopy IV GX Dynamic Reconfiguration* chapter in volume 3 of the *HardCopy IV Device Handbook* "CMU PLL Reconfiguration Mode" section of the *HardCopy IV GX Dynamic Reconfiguration* chapter in volume 3 of the *HardCopy IV Device Handbook*

Figure 3–4 shows page 4 of the ALTGX_RECONFIG MegaWizard Plug-In Manager.

Figure 3–4. MegaWizard Plug-In Manager—ALTGX_RECONFIG (Analog Controls) (Page 4)

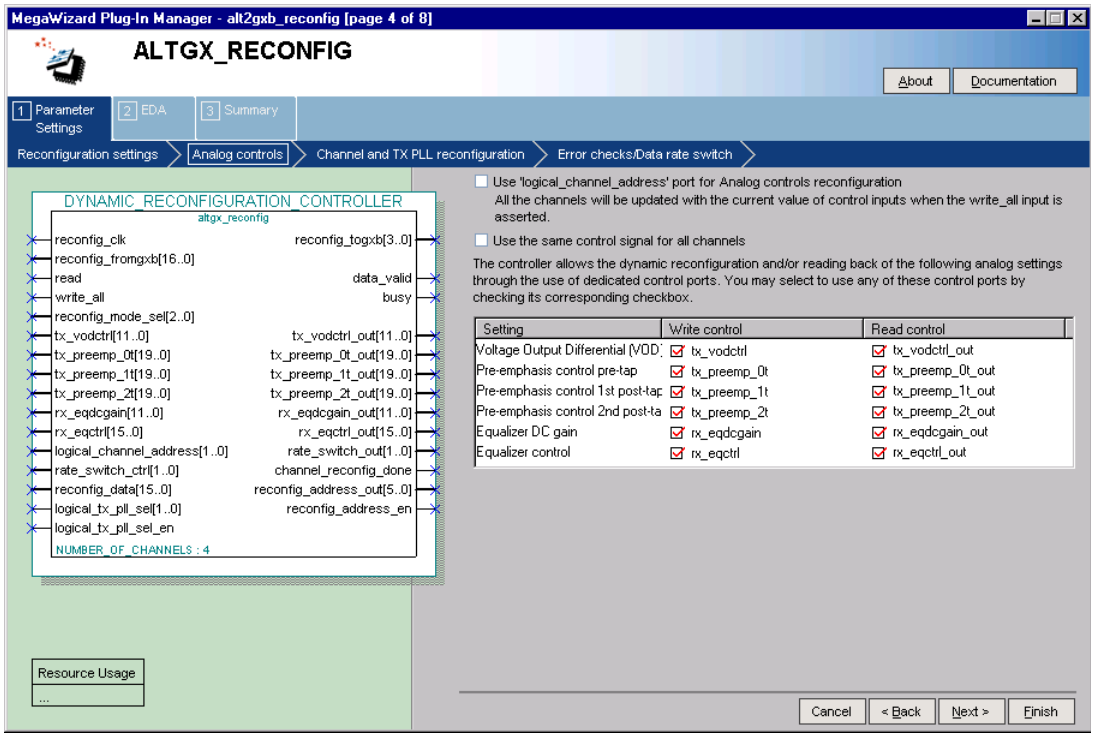


Table 3–2 describes the available options on page 4 of the MegaWizard Plug-In Manager for your ALTGX_RECONFIG custom megafunction variation.

Make your selections on page 4, then click **Next**.

Table 3–2. MegaWizard Plug-In Manager Options (Page 4) (Part 1 of 2)

ALTGX_RECONFIG Setting	Description	Reference
Use '`logical_channel_address`' port for Analog controls reconfiguration	This option is applicable only for Analog controls reconfiguration and is available for selection when the number of channels controlled by the ALTGX_RECONFIG instance is more than one. The dynamic reconfiguration controller reconfigures only the channel whose logical channel address is specified at the `logical_channel_address` port. The width of this port is selected by the ALTGX_RECONFIG MegaWizard Plug-In Manager depending on the number of channels controlled by the dynamic reconfiguration controller. The maximum width of the `logical_channel_address` port is 9 bits.	"Dynamically Reconfiguring PMA Controls" section of the *HardCopy IV GX Dynamic Reconfiguration* chapter in volume 3 of the *HardCopy IV Device Handbook*
Use the same control signal for all channels	This option is available for selection when the number of channels controlled by the ALTGX_RECONFIG instance is more than one. The dynamic reconfiguration controller writes the same control signals to all the channels connected to it when you enable this option. This option is not available for selection when you enable the **Use 'logical_channel_address' port for Analog controls reconfiguration** option.	

Table 3–2. MegaWizard Plug-In Manager Options (Page 4) (Part 2 of 2)

ALTGX_RECONFIG Setting	Description	Reference
Write Control	The PMA control ports available to write various analog settings to the transceiver channels controlled by the dynamic reconfiguration controller are as follows: ■ `tx_vodctrl`—voltage Output Differential (VOD) — 3 bits per channel ■ `tx_preemp_0t`—pre-emphasis control pre-tap — 5 bits per channel ■ `tx_preemp_1t`—pre-emphasis control 1st post-tap—5 bits per channel ■ `tx_preemp_2t`—pre-emphasis control 2nd post-tap—5 bits per channel ■ `rx_eqdcgain`—equalizer DC gain—3 bits per channel ■ `rx_eqctrl`—equalizer control—4 bits per channel These are optional signals. The signal widths are based on the setting you entered for the **What is the number of channels controlled by the reconfig controller?** option and whether you enable the **Use 'logical_channel_address' port for Analog controls reconfiguration** option. At least one of these PMA control ports must be enabled to configure and use the dynamic reconfiguration controller.	"Dynamically Reconfiguring PMA Controls" section of the *HardCopy IV GX Dynamic Reconfiguration* chapter in volume 3 of the *HardCopy IV Device Handbook*
Read Control	The PMA control ports available to read the existing values from the transceiver channels controlled by the dynamic reconfiguration controller are as follows: ■ `tx_vodctrl_out`—VOD)—3 bits per channel ■ `tx_preemp_0t_out`—pre-emphasis control pre-tap—5 bits per channel ■ `tx_preemp1t_out`—pre-emphasis control 1st post-tap—5 bits per channel ■ `tx_preemp_2t_out`—pre-emphasis control 2nd post-tap—5 bits per channel ■ `rx_eqdcgain_out`—Equalizer DC gain—3 bits per channel ■ `rx_eqctrl_out`—Equalizer control—4 bits per channel These are optional signals. The signal widths are based on the setting you entered for the **What is the number of channels controlled by the reconfig controller?** option and whether you enable the **Use 'logical_channel_address' port for Analog controls reconfiguration** option. The PMA controls are available for selection only if the corresponding write control is selected. Read and write transactions cannot be performed simultaneously.	

Figure 3–5 shows page 5 of the ALTGX_RECONFIG MegaWizard Plug-In Manager.

Figure 3–5. MegaWizard Plug-In Manager—ALTGX_RECONFIG (Analog Controls)

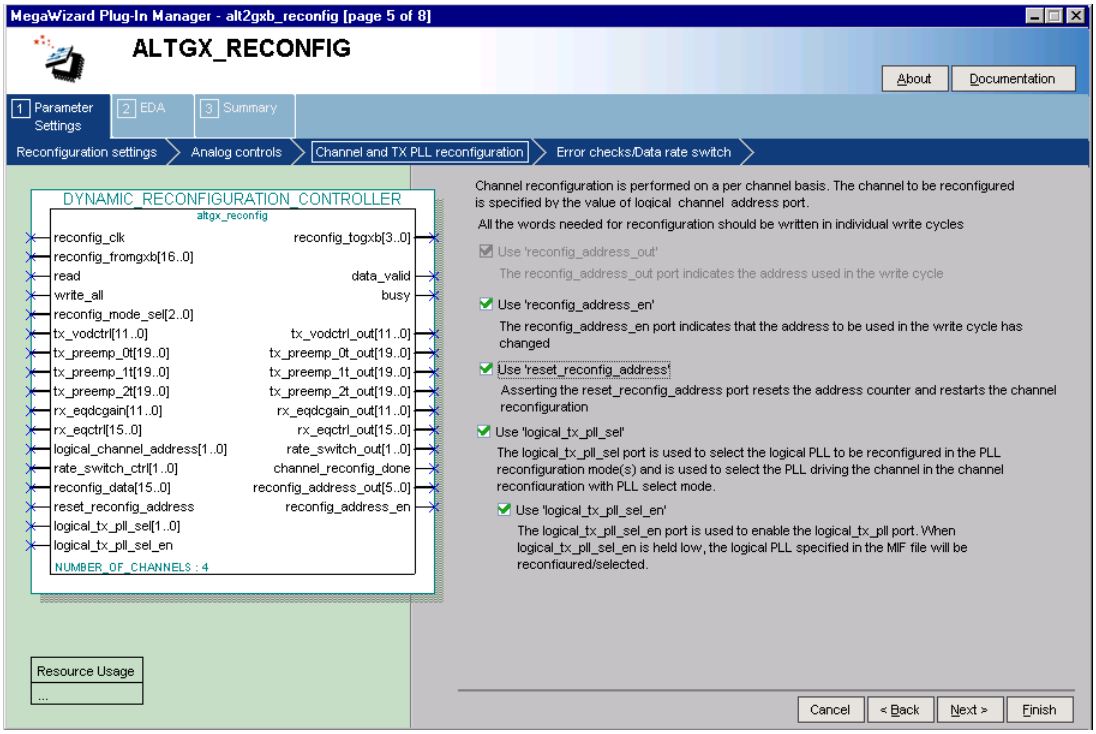


Table 3–3 describes the available options on page 5 of the MegaWizard Plug-In Manager for your ALTGX_RECONFIG custom megafunction variation.

Table 3–3. MegaWizard Plug-In Manager Options (Page 5) (Part 1 of 2)

ALTGX_RECONFIG Setting	Description	Reference
Use 'reconfig_address_out'	This option is enabled by default when you select the **Channel and TX PLL select/reconfig** option. The value on `reconfig_address_out[5:0]` indicates the address associated with the words in the **.mif**, which contains the dynamic reconfiguration instructions. The dynamic reconfiguration controller automatically increments the address at the end of each **.mif** write transaction.	"Dynamic Reconfiguration Controller Port List" section in the *HardCopy IV GX Dynamic Reconfiguration* chapter in volume 3 of the *HardCopy IV Device Handbook*
Use 'reconfig_address_en'	When high, this optional output status signal indicates that the address to be used in the **.mif** write transaction cycle has changed. This signal gets asserted when the **.mif** write transaction is completed (the `busy` signal de-asserted).	"Dynamic Reconfiguration Controller Port List" section in the *HardCopy IV GX Dynamic Reconfiguration* chapter in volume 3 of the *HardCopy IV Device Handbook*
Use 'reset_reconfig_address'	When asserted, this optional control signal resets the `reconfig_address_out` (current reconfiguration address) to 0.	"Dynamic Reconfiguration Controller Port List" section in the *HardCopy IV GX Dynamic Reconfiguration* chapter in volume 3 of the *HardCopy IV Device Handbook*

Table 3–3. MegaWizard Plug-In Manager Options (Page 5) (Part 2 of 2)

ALTGX_RECONFIG Setting	Description	Reference
Use 'logical_tx_pll_sel'	This is an optional control signal. The `logical_tx_pll_sel[1:0]` signal refers to the logical reference index of the CMU PLL. The functionality of the signal depends on the feature activated, as shown below: ■ **CMU PLL reconfiguration**—The corresponding CMU PLL is reconfigured based on the value at `logical_tx_pll_sel[1:0]`. ■ **Channel and CMU PLL reconfiguration**—The corresponding CMU PLL is reconfigured based on the value at this signal. The transceiver channel listens to the CMU PLL selected by `logical_tx_pll_sel[1:0]`. ■ **Channel reconfiguration with TX PLL select**—The transceiver channel listens to the TX PLL selected by `logical_tx_pll_sel[1:0]`.	"The logical_tx_pll_sel and logical_tx_pll_sel_en Ports" section in the *HardCopy IV GX Dynamic Reconfiguration* chapter in volume 3 of the *HardCopy IV Device Handbook*
Use 'logical_tx_pll_sel_en'	This is an optional control signal. When you enable this signal, the value set on the `logical_tx_pll_sel[1:0]` signal is valid only if `logical_tx_pll_sel_en` is set to **1**.	"The logical_tx_pll_sel and logical_tx_pll_sel_en Ports" section in the *HardCopy IV GX Dynamic Reconfiguration* chapter in volume 3 of the *HardCopy IV Device Handbook*

Figure 3–6 shows page 6 of the ALTGX_RECONFIG MegaWizard Plug-In Manager.

Figure 3–6. MegaWizard Plug-In Manager—ALTGX_RECONFIG (Error Checks/Data Rate Switch)

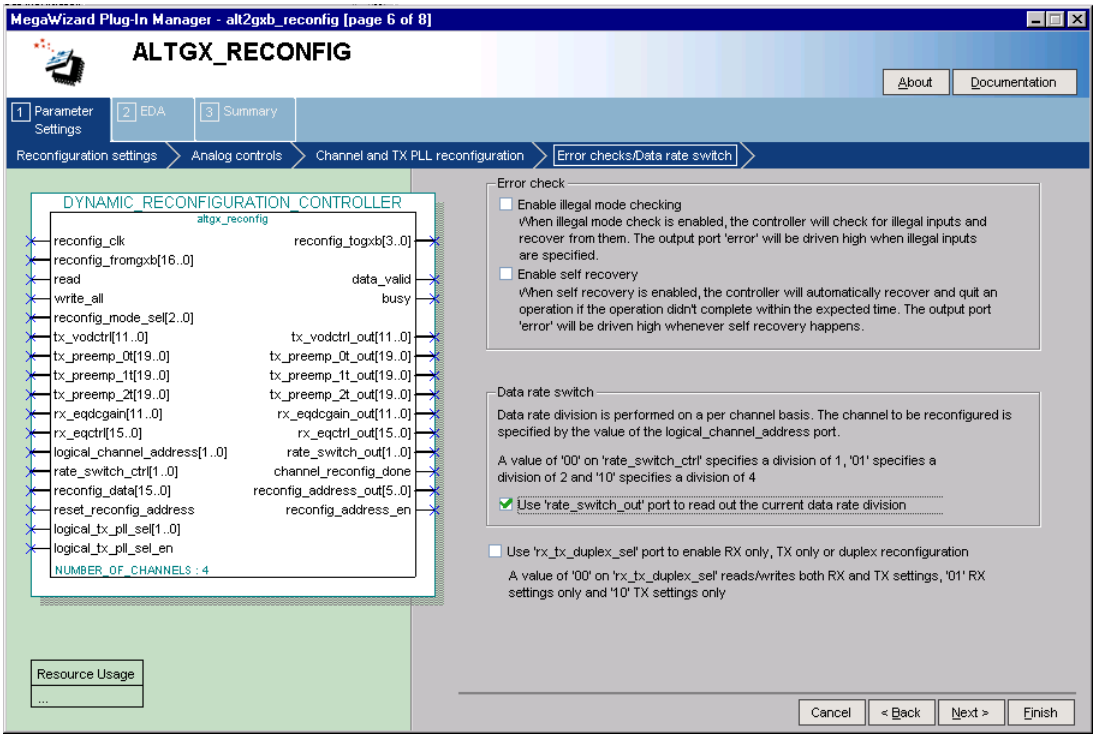


Table 3–4 describes the available options on page 6 of the MegaWizard Plug-In Manager for your ALTGX_RECONFIG custom megafunction variation.

Make your selections on page 6, then click **Next**.

Table 3–4. MegaWizard Plug-In Manager Options (Page 6)

ALTGX_RECONFIG Setting	Description	Reference
Enable illegal mode checking	When you select this option, the ALTGX_RECONFIG MegaWizard Plug-In Manager provides the `error` output port. The dynamic reconfiguration controller checks for specific unsupported options within two `reconfig_clk` cycles, de-asserts the `busy` signal and asserts the `error` output port for two `reconfig_clk` cycles. The dynamic reconfiguration controller does not execute the unsupported operation.	"Error Indication in the ALTGX_RECONFIG MegaWizard Plug-In Manager" section of the *HardCopy IV GX Dynamic Reconfiguration* chapter in volume 3 of the *HardCopy IV Device Handbook*
Enable self recovery	When you select this option, the ALTGX_RECONFIG MegaWizard Plug-In Manager provides the `error` output port. The dynamic reconfiguration controller quits an operation if it did not complete within the expected number of clock cycles. After recovering from the illegal operation, the dynamic reconfiguration controller de-asserts the `busy` signal and asserts the `error` output port for two `reconfig_clk` cycles.	"Error Indication in the ALTGX_RECONFIG MegaWizard Plug-In Manager" section of the *HardCopy IV GX Dynamic Reconfiguration* chapter in volume 3 of the *HardCopy IV Device Handbook*
Use `rate_switch_out` port to read out the current data rate division	The `rate_switch_out[1:0]` signal is available when you select Data Rate Division in TX mode. You can read the existing local divider settings of a transmitter channel at this port. The decoding for this signal is listed below: 2'b00—Division of 1 2'b01—Division of 2 2'b10—Division of 4 2'b11—Not supported	"Data Rate Division in TX mode" section in the *HardCopy IV GX Dynamic Reconfiguration* chapter in volume 3 of the *HardCopy IV Device Handbook*
Use the `rx_tx_duplex_sel` port to enable RX only, TX only or duplex configuration.	You can read or write the receiver and transmitter settings, or only the receiver settings, or only the transmitter settings, based on the value you set at the `rx_tx_duplex_sel[1:0]` port. ■ 2'b00—Duplex mode ■ 2'b01—RX only mode ■ 2'b10—TX only mode ■ 2'b11—Unsupported value (do not use this value) If you disable the `rx_tx_duplex_sel[1:0]` port, the dynamic reconfiguration controller will read or write both the receiver and transmitter settings.	"Dynamically Reconfiguring PMA Controls" section of the *HardCopy IV GX Dynamic Reconfiguration* chapter in volume 3 of the *HardCopy IV Device Handbook*

Figure 3–7 shows page 7 (the Simulation Libraries page) of the MegaWizard Plug-In Manager, which is used for dynamic reconfiguration selection.

Make your selections, then click **Next**.

Figure 3–7. MegaWizard Plug-In Manager—ALTGX_RECONFIG (Simulation Libraries)

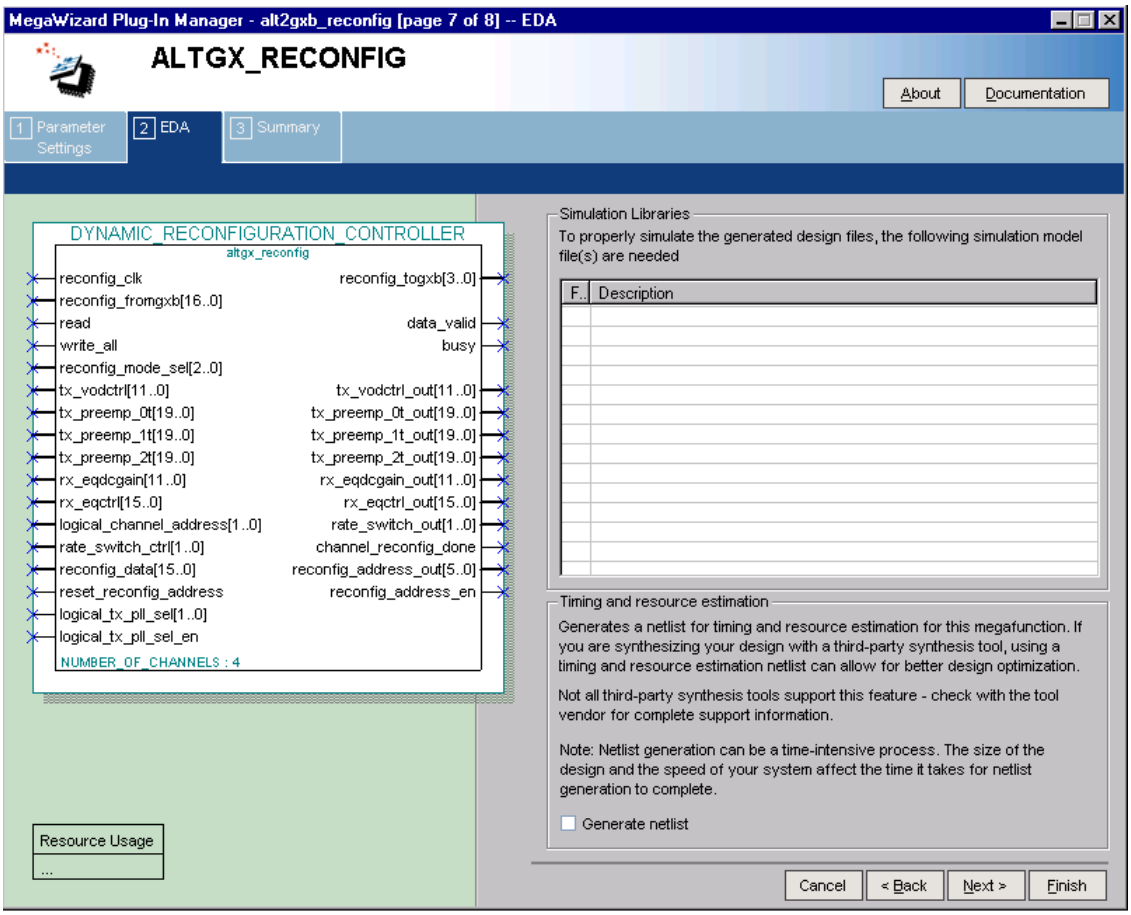


Table 3–5 describes the available option on page 7 of the MegaWizard Plug-In Manager for your ALTGX_RECONFIG custom Megafunction variation.

Make your selections on page 6, then click **Next**.

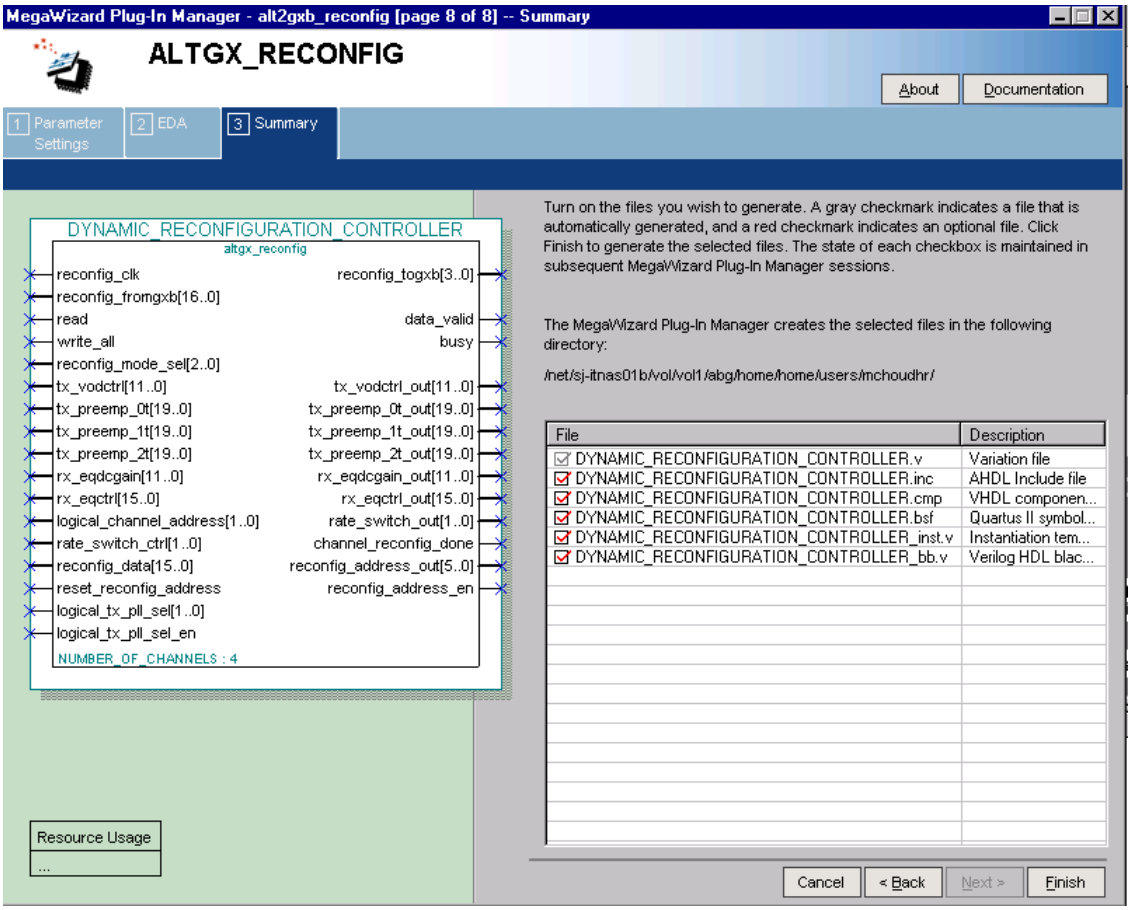
Table 3–5. MegaWizard Plug-In Manager Options (Page 7)

ALTGX_RECONFIG Setting	Description
Generate a netlist for synthesis area and timing estimation	Selecting this option generates a netlist file that third-party synthesis tools use to estimate timing and resource usage

Figure 3–8 shows page 7 (the last page) of the MegaWizard Plug-In Manager for the dynamic reconfiguration protocol set up. You can select optional files on this page.

After you make your selections, click **Finish** to generate the files.

Figure 3–8. MegaWizard Plug-In Manager—ALTGX_RECONFIG (Summary)



Document Revision History

Table 3–6 shows the revision history for this chapter.

Table 3–6. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
June 2009, v1.0	Initial release.	—

About this Handbook

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

How to Contact Altera

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

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

Note:

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

Typographic Conventions

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

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

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

HardCopy IV Device Handbook, Volume 4

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

HC4_H5V4-1.0

Contents

Chapter Revision Dates

The chapter in this book, *HardCopy IV Device Handbook, Volume 4*, was revised on the following date. Where chapters or groups of chapters are available separately, part numbers are listed.

Chapter 1 DC and Switching Characteristics of HardCopy IV Devices
Revised: *June 2009*
Part Number: *HIV54001-1.0*

Section I. HardCopy IV Device Datasheet

This section provides the datasheet for the HardCopy® IV device family. This section includes the following chapter:

- Chapter 1, DC and Switching Characteristics of HardCopy IV Devices

Revision History

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

1. DC and Switching Characteristics of HardCopy IV Devices

HIV54001-1.0

Electrical Characteristics

This chapter covers the electrical characteristics for HardCopy® IV devices.

Operating Conditions

When HardCopy IV devices are implemented in a system, they are rated according to a set of defined parameters. To maintain the highest possible performance and reliability, consider the operating requirements described in this chapter. HardCopy IV devices are not speed binned like Stratix® IV devices because HardCopy IV devices are designed and built to function at a target frequency based on timing constraints. HardCopy IV devices are offered in both commercial and industrial grades.

Absolute Maximum Ratings

Absolute maximum ratings define the maximum operating conditions for HardCopy IV devices. The values are based on experiments conducted with the devices and theoretical modeling of breakdown and damage mechanisms. The functional operation of the device is not implied for these conditions.

Conditions other than those listed in Table 1–1 and Table 1–3 may cause permanent damage to the device. Additionally, device operation at the absolute maximum ratings for extended periods of time may have adverse effects on the device.

Table 1–1. HardCopy IV Device Absolute Maximum Ratings—Preliminary (Part 1 of 2) *(Note 1)*

Symbol	Description	Minimum	Maximum	Unit
V_{CC}	Core voltage and periphery circuitry power supply	-0.5	1.35	V
V_{CCPT} *(2)*	Power supply for programmable power technology	—	—	V
V_{CCPGM}	Configuration pins power supply	-0.5	3.75	V
V_{CCAUX}	Power supply for temperature sensing diode and POR	-0.5	3.75	V
V_{CCBAT} *(3)*	Battery back-up power supply for design security volatile key register	—	—	V
V_{CCPD}	I/O pre-driver power supply	-0.5	3.75	V
V_{CCIO}	I/O power supply	-0.5	3.9	V
V_{CC_CLKIN}	Differential clock input power supply	-0.5	3.75	V
V_{CCD_PLL}	PLL digital power supply	-0.5	1.35	V
V_{CCA_PLL}	PLL analog power supply	-0.5	3.75	V
V_I	DC input voltage	-0.5	4.0	V
I_{OUT}	DC output current per pin	-25	40	mA
T_J	Operating junction temperature	-55	125	C

Table 1–1. HardCopy IV Device Absolute Maximum Ratings—Preliminary (Part 2 of 2) *(Note 1)*

Symbol	Description	Minimum	Maximum	Unit
T_{STG}	Storage temperature (No bias)	-65	150	C

Notes for Table 1–1:

(1) Supply voltage specifications apply to voltage readings taken at the device pins and not the power supply.
(2) HardCopy IV devices do not require Programmable Power Technology.
(3) This power supply is not used in HardCopy IV devices.

Table 1–2 HardCopy IV GX Transceiver Power Supply Absolute Maximum Ratings.

Table 1–2. HardCopy IV GX Transceiver Power Supply Absolute Maximum Ratings

Symbol	Description	Minimum	Maximum	Unit
V_{CCA_L}	Transceiver high voltage power (left side)	-0.5	3.75	V
V_{CCA_R}	Transceiver high voltage power (right side)	-0.5	3.75	V
V_{CCHIP_L}	Transceiver hard IP digital power (right side)	-0.5	1.35	V
V_{CCHIP_R}	Transceiver hard IP digital power (left side)	-0.5	1.35	V
V_{CCR_L}	Receiver power (left side)	-0.5	1.35	V
V_{CCR_R}	Receiver power (right side)	-0.5	1.35	V
V_{CCT_L}	Transmitter power (left side)	-0.5	1.35	V
V_{CCT_R}	Transmitter power (right side)	-0.5	1.35	V
V_{CCL_GXBLn} *(1)*	Transceiver clock power (left side)	-0.5	1.35	V
V_{CCL_GXBRn} *(1)*	Transceiver clock power (right side)	-0.5	1.35	V
V_{CCH_GXBLn} *(1)*	Transmitter output buffer power (left side)	-0.5	1.65	V
V_{CCH_GXBRn} *(1)*	Transmitter output buffer power (right side)	-0.5	1.65	V

Notes to Table 1–2:

(1) The V_{CCH} and V_{CCL} powers are per transceiver block.

Maximum Allowed Overshoot/Undershoot Voltage

During transitions, input signals may overshoot to the voltage shown in Table 1–3 and undershoot to -2.0 V for input currents less than 100 mA and periods shorter than 20 ns.

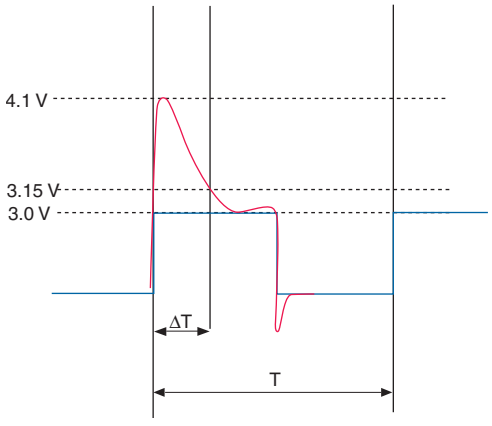
Table 1–3 lists the maximum allowed input overshoot voltage and the duration of the overshoot voltage as a percentage of device lifetime. The maximum allowed overshoot duration is specified as a percentage of high-time over the lifetime of the device. A DC signal is equivalent to 100% duty cycle.

Table 1–3. Maximum Allowed Overshoot During Transitions—Preliminary

Symbol	Description	Condition	Overshoot Duration as Percentage of High Time	Unit
Vi (AC)	AC input voltage	4.0 V	100.000	%
		4.05 V	79.330	%
		4.1 V	46.270	%
		4.15 V	27.030	%
		4.2 V	15.800	%
		4.25 V	9.240	%
		4.3 V	5.410	%
		4.35 V	3.160	%
		4.4 V	1.850	%
		4.45 V	1.080	%
		4.5 V	0.630	%
		4.55 V	0.370	%
		4.6 V	0.220	%
		4.65 V	0.130	%
		4.7 V	0.074	%
		4.75 V	0.043	%
		4.8 V	0.025	%
		4.85 V	0.015	%

Figure 1–1 shows the methodology to determine overshoot duration. The overshoot voltage is displayed in red and is present at the HardCopy IV pin, up to 4.1 V. In Table 1–3, for an overshoot of up to 4.1 V, the percentage of high time for overshoot is greater than 3.15 V can be as high as 46% over an 11.4 year period. The percentage of high time is calculated as (Δ T/T) × 100. This 11.4 year period assumes the device is always turned on with 100% I/O toggle rate and 50% duty cycle signal. For lower I/O toggle rates and situations where the device is in an idle state, lifetimes are increased.

Figure 1–1. Overshoot Duration



Recommended Operating Conditions

This section lists the functional operation limits for AC and DC parameters for HardCopy IV devices. Table 1–4 shows the steady-state voltage and current values expected from HardCopy IV devices. All supplies are required to monotonically reach their full-rail values within t_{RAMP} maximum. Allowed ripple on power supplies is bounded by the minimum and maximum specifications listed in Table 1–4.

Table 1–4. HardCopy IV Device Recommended Operating Conditions—Preliminary (Part 1 of 2)

Symbol	Description	Condition	Minimum	Typical	Maximum	Unit
V_{CC}	Core voltage and periphery circuitry power supply	—	0.87	0.90	0.93	V
V_{CCPT} *(1)*	Power supply for programmable power technology	—	—	—	—	V
V_{CCAUX}	Power supply for the temperature sensing diode and POR	—	2.375	2.5	2.625	V
V_{CCPD}	I/O pre-driver (3.0 V) power supply	—	2.85	3	3.15	V
	I/O pre-driver (2.5 V) power supply	—	2.375	2.5	2.625	V
V_{CCIO}	I/O buffers (3.0-V) power supply	—	2.85	3	3.15	V
	I/O buffers (2.5-V) power supply	—	2.375	2.5	2.625	V
	I/O buffers (1.8-V) power supply	—	1.71	1.8	1.89	V
	I/O buffers (1.5-V) power supply	—	1.425	1.5	1.575	V
	I/O buffers (1.2-V) power supply	—	1.14	1.2	1.26	V
V_{CCPGM}	Configuration pins (3.0-V) power supply	—	2.85	3	3.15	V
	Configuration pins (2.5-V) power supply	—	2.375	2.5	2.625	V
	Configuration pins (1.8-V) power supply	—	1.71	1.8	1.89	V
V_{CCA_PLL}	PLL analog voltage regulator power supply	—	2.375	2.5	2.625	V
V_{CCD_PLL}	PLL digital voltage regulator power supply	—	0.87	0.90	0.93	V
V_{CC_CLKIN}	Differential clock input power supply	—	1.075	1.2	1.325	V
	Differential clock input power supply	—	1.375	1.5	1.625	V
	Differential clock input power supply	—	1.675	1.8	1.925	V
	Differential clock input power supply	—	2.375	2.5	2.625	V
	Differential clock input power supply	—	2.875	3.0	3.125	V
V_{CCBAT} *(2)*	Battery back-up power supply (for design security volatile key register)	—	—	—	—	V
V_I	DC input voltage	—	–0.5	—	3.6	V
V_O	Output voltage	—	0	—	V_{CCIO}	V
T_J	Operating junction temperature	Commercial	0	—	85	°C
		Industrial	–40	—	100	°C

Table 1–4. HardCopy IV Device Recommended Operating Conditions—Preliminary (Part 2 of 2)

Symbol	Description	Condition	Minimum	Typical	Maximum	Unit
t_{RAMP}	Power supply ramp time	Normal POR (PORSEL = 1)	0.05	—	100	ms
		Fast POR (PORSEL = 1) *(3)*	0.05	—	4	ms

Notes to Table 1–4:

(1) HardCopy IV devices do not require Programmable Power Technology.

(2) In HardCopy IV devices, this power supply is not used.

(3) If the PORSEL pin is connected to V_{CC}, all supplies must ramp up within 4 ms.

Table 1–5 shows the transceiver power supply recommended operating conditions.

Table 1–5. HardCopy IV GX Transceiver Power Supply Recommended Operating Conditions

Symbol	Description	Minimum	Typical	Maximum	Unit
V_{CCA_L}	Transceiver high voltage power (left side)	2.85/2.375	3.0/2.5 *(3)*	3.15/2.625	V
V_{CCA_R}	Transceiver high voltage power (right side)				V
V_{CCHIP_L} *(1)*	Transceiver hard IP digital power (right side)	0.87	0.90	0.93	V
V_{CCHIP_R} *(1)*	Transceiver hard IP digital power (left side)				V
V_{CCR_L}	Receiver power (left side)	1.05	1.1	1.15	V
V_{CCR_R}	Receiver power (right side)				V
V_{CCT_L}	Transmitter power (left side)	1.05	1.1	1.15	V
V_{CCT_R}	Transmitter power (right side)				V
V_{CCL_GXBLn} *(2)*	Transceiver clock power (left side)	1.05	1.1	2.35	V
V_{CCL_GXBRn} *(2)*	Transceiver clock power (right side)				V
V_{CCH_GXBLn} *(2)*	Transmitter output buffer power (left side)	1.33/1.425	1.4/1.5 *(4)*	1.47/1.575	V
V_{CCH_GXBRn} *(2)*	Transmitter output buffer power (right side)				V

Notes to Table 1–5:

(1) If $V_{CCHIP\_L/R}$ is connected to the same power supply source as Vcc, the recommended minimum and maximum operating supply levels are 0.87 V and 0.93 V respectively.

(2) The V_{CCH} and V_{CCL} powers are per transceiver block

(3) $V_{CCA\_L/R}$ must be connected to a 3.0 V supply if the CMU PLL, receiver CDR, or both are configured at a base data rate > 4.25Gbps. For data rates up to 4.25 Gbps, you can connect $V_{CCA\_L/R}$ to either 3.0 V or 2.5 V.

(4) For data rates up to 6.5 Gbps, you can connect $V_{CCH\_GXBL/R}$ to either 1.4 V or 1.5 V.

DC Characteristics

This section lists the supply current, I/O pin leakage current, input pin capacitance, on-chip termination (OCT) tolerance, and hot socketing specifications.

Supply Current

Standby current is the current the device draws after the device is configured, with no inputs or outputs toggling and no activity in the device. Because these currents vary largely with the resources you use, use the Excel-based PowerPlay Early Power Estimator (EPE) to get supply current estimates for your design.

Table 1–6 lists supply current specifications for V_{CC_CLKIN} and V_{CCPGM}. Use the PowerPlay EPE to get supply current estimates for the remaining power supplies.

Table 1–6. Supply Current Specifications for V_{CC_CLKIN} and V_{CCPGM}—Preliminary *(Note 1)*

Symbol	Parameter	Min	Max	Unit
I_{CLKIN}	V_{CC_CLKIN} current specifications	0	TBD	mA
I_{PGM}	V_{CCPGM} current specifications	0	TBD	mA

Note to Table 1–6:

(1) Pending silicon characterization.

I/O Pin Leakage Current

Table 1–7 defines the HardCopy IV I/O pin leakage current specifications.

Table 1–7. HardCopy IV I/O Pin Leakage Current—Preliminary *(Note 1)*, *(2)*

Symbol	Description	Conditions	Min	Typ	Max	Unit
I_I	Input pin	V_I = 0V to $V_{CCIOMAX}$	-10	—	10	µA
I_{OZ}	Tri-stated I/O pin	V_O = 0V to $V_{CCIOMAX}$	-10	—	10	µA

Notes to Table 1–7:

(1) This value is specified for normal device operation. The value may vary during power up. This applies for all V_{CCIO} settings (3.0, 2.5, 1.8, 1.5, and 1.2 V).

(2) The 10 mA I/O leakage current limit is applicable when the internal clamping diode is off. A higher current is observed when the diode is on.

OCT Specifications

If OCT calibration is enabled, calibration is automatically performed at power up for I/Os connected to the calibration block. Table 1–8 lists the HardCopy IV OCT calibration block accuracy specifications.

Table 1–8. HardCopy IV OCT With Calibration Specification for I/Os—Preliminary *(Note 1)*

Symbol	Description	Conditions	Calibration Accuracy		Unit
			Commercial *(2)*	Industry	
25-Ω R_S 3.0/2.5	Internal series termination with calibration (25-Ω setting)	V_{CCIO} = 3.0/2.5 V	TBD	—	%
50-Ω R_S 3.0/2.5	Internal series termination with calibration (50-Ω setting)	V_{CCIO} = 3.0/2.5 V	TBD	—	%
50-Ω R_T 2.5	Internal series termination with calibration (50-Ω setting)	V_{CCIO} = 2.5 V	TBD	—	%
25-Ω R_S 1.8	Internal series termination with calibration (25-Ω setting)	V_{CCIO} = 1.8 V	TBD	—	%
50-Ω R_S 1.8	Internal series termination with calibration (50-Ω setting)	V_{CCIO} = 1.8 V	TBD	—	%
50-Ω R_T 1.8	Internal series termination with calibration (50-Ω setting)	V_{CCIO} = 1.8 V	TBD	—	%
50-Ω R_S 1.5	Internal series termination with calibration (50-Ω setting)	V_{CCIO} = 1.5 V	TBD	—	%
50-Ω R_T 1.5	Internal series termination with calibration (50-Ω setting)	V_{CCIO} = 1.5 V	TBD	—	%
50-Ω R_S 1.2	Internal series termination with calibration (50-Ω setting)	V_{CCIO} = 1.2 V	TBD	—	%
50-Ω R_T 1.2	Internal series termination with calibration (50-Ω setting)	V_{CCIO} = 1.2 V	TBD	—	%

Notes to Table 1–8:

(1) OCT calibration accuracy is valid at the time of calibration only.

(2) Pending silicon characterization.

The calibration accuracy for calibrated series and parallel OCTs are applicable at the moment of calibration. If the voltage or temperature changes, the termination resistance value varies. Table 1–9 lists the resistance tolerance for HardCopy IV OCT.

Table 1–9. I/O OCT Resistance Tolerance—Preliminary *(Note 1)*

Symbol	Description	Resistance Tolerance		
		Commercial Max	Industrial Max	Unit
R_{OCT_CAL}	Internal series/parallel OCT with calibration	*(2)*	—	%
R_{OCT_UNCAL}	Internal series/parallel OCT without calibration	TBD	—	%

Notes to Table 1–9:

(1) Pending silicon characterization.

(2) For resistance tolerance after power-up calibration, refer to Table 1–10.

Table 1–10 lists OCT variation with temperature and voltage after power-up calibration. Use Table 1–10 and Equation 1–1 to determine the OCT variation when voltage and temperature vary after power-up calibration.

Equation 1–1. OCT Variation Without Re-Calibration *(Note 1)*

$$R_{OCT} = R_{CAL}\left(1 + \frac{dR}{dT} \times \Delta T + \frac{dR}{dV} \times \Delta V\right)$$

Note to Equation 1–1:

(1) R_{CAL} is calibrated OCT at power up. ΔT and ΔV are variations in temperature and voltage with respect to temperature and V_{CCIO} values, respectively, at power up.

Table 1–10. OCT Variation after Power-Up Calibration—Preliminary *(Note 1)*, *(2)*

Symbol	Description	V_{CCIO} (V)	Commercial Typical	Unit
dR/dV	OCT variation with voltage without re-calibration	3.0	TBD	%/mV
		2.5	TBD	
		1.8	TBD	
		1.5	TBD	
		1.2	TBD	
dR/dT	OCT variation with temperature without re-calibration	3.0	TBD	%/°C
		2.5	TBD	
		1.8	TBD	
		1.5	TBD	
		1.2	TBD	

Notes to Table 1–10:

(1) Valid for V_{CCIO} range of ± 5% and a temperature range of 0° to 85°C.

(2) Pending silicon characterization.

Pin Capacitance

Table 1–11 shows the HardCopy IV device family pin capacitance.

Table 1–11. HardCopy IV Device Capacitance—Preliminary *(Note 1)*

Symbol	Description	Typical	Unit
C_{IOTB}	Input capacitance on top/bottom I/O pins	TBD	pF
C_{IOLR}	Input capacitance on left/right I/O pins	TBD	pF
C_{CLKTB}	Input capacitance on top/bottom dedicated clock input pins	TBD	pF
C_{CLKLR}	Input capacitance on left/right dedicated clock input pins	TBD	pF
C_{OUTFB}	Input capacitance on dual-purpose clock output/feedback pins	TBD	pF
C_{CLK1} C_{CLK3} C_{CLK8} C_{CLK10}	Input capacitance for dedicated clock input pins	TBD	pF

Note to Table 1–11:

(1) Pending silicon characterization.

Hot Socketing

Table 1–12 defines the hot socketing specification for HardCopy IV devices.

Table 1–12. HardCopy IV Hot Socketing Specifications—Preliminary *(Note 1)*

Symbol	Description	Maximum
$I_{IOPIN(DC)}$	DC current per I/O pin	300 µA
$I_{IOPIN(AC)}$	AC current per I/O pin	8 mA for t_{RISE} > 10 ns

Note to Table 1–12:

(1) Pending silicon characterization.

Internal Weak Pull-Up Resistor

Table 1–13 lists the weak pull-up resistor values for HardCopy IV devices.

Table 1–13. HardCopy IV Internal Weak Pull-Up Resistor—Preliminary *(Note 1)*, *(2)*

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
R_{PU}	The value of I/O pin pull-up resistor before and during user mode, if the pull-up resistor option is enabled.	V_{CCIO} = 3.0 V ± 5% *(3)*	—	25	—	kΩ
		V_{CCIO} = 2.5 V ± 5% *(3)*	—	25	—	kΩ
		V_{CCIO} = 1.8 V ± 5% *(3)*	—	25	—	kΩ
		V_{CCIO} = 1.5 V ± 5% *(3)*	—	25	—	kΩ
		V_{CCIO} = 1.2 V ± 5% *(3)*	—	25	—	kΩ

Notes to Table 1–13:

(1) Pending silicon characterization.

(2) All I/O pins have an option to enable weak pull-up except test and JTAG pins.

(3) Pin pull-up resistance values may be lower if an external source drives the pin higher than V_{CCIO}.

I/O Standard Specifications

Table 1–14 through Table 1–19 list input voltage (V_{IH} and V_{IL}), output voltage (V_{OH} and V_{OL}), and current drive characteristics (I_{OH} and I_{OL}) for various I/O standards supported by HardCopy IV devices. These tables also show the HardCopy IV device family I/O standard specifications. Refer to the "Glossary" on page 1–32 for an explanation of terms used in Table 1–14 through Table 1–19. V_{OL} and V_{OH} values are valid at the corresponding I_{OH} and I_{OL}, respectively.

Table 1–14. Single-Ended I/O Standards—Preliminary

I/O Standard	V_{CCIO} (V)			V_{IL} (V)		V_{IH} (V)		V_{OL} (V)	V_{OH} (V)	I_{OL} (mA)	I_{OH} (mA)
	Min	Typ	Max	Min	Max	Min	Max	Max	Min		
3.0-V LVTTL	2.85	3	3.15	-0.3	0.8	1.7	3.6	0.4	2.4	2	-2
3.0-V LVCMOS	2.85	3	3.15	-0.3	0.8	1.7	3.6	0.2	V_{CCIO} - 0.2	0.1	-0.1
2.5-V LVTTL/LVCMOS	2.375	2.5	2.625	-0.3	0.7	1.7	3.6	0.2	2.1	0.1	-0.1
								0.4	2	1	-1
								0.7	1.7	2	-2
1.8-V LVTTL/LVCMOS	1.71	1.8	1.89	-0.3	0.35 * V_{CCIO}	0.65 * V_{CCIO}	V_{CCIO} + 0.3	0.45	V_{CCIO} -0.45	2	-2
1.5-V LVTTL/LVCMOS	1.425	1.5	1.575	-0.3	0.35 * V_{CCIO}	0.65 * V_{CCIO}	V_{CCIO} + 0.3	0.25 * V_{CCIO}	0.75 * V_{CCIO}	2	-2
1.2-V LVTTL/LVCMOS	1.14	1.2	1.26	-0.3	0.35 * V_{CCIO}	0.65 * V_{CCIO}	V_{CCIO} + 0.3	0.25 * V_{CCIO}	0.75 * V_{CCIO}	2	-2
3.0-V PCI	2.85	3	3.15	—	0.3 * V_{CCIO}	0.5 * V_{CCIO}	3.6	0.1 * V_{CCIO}	0.9 * V_{CCIO}	1.5	-0.5
3.0-V PCI-X	2.85	3	3.15	—	0.35 * V_{CCIO}	0.5 * V_{CCIO}	—	0.1 * V_{CCIO}	0.9 * V_{CCIO}	1.5	-0.5

Refer to "Single-Ended Voltage Referenced I/O Standard" in Table 1–39 on page 1–32 for an example of a voltage referenced receiver input waveform and explanation of terms used in Table 1–15.

Table 1–15. Single-Ended SSTL and HSTL I/O Reference Voltage Specifications—Preliminary

I/O Standard	V_{CCIO} (V)			V_{REF} (V)			V_{TT} (V)		
	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max
SSTL-2 Class I, II	2.375	2.5	2.625	0.49 * V_{CCIO}	0.5 * V_{CCIO}	0.51 * V_{CCIO}	V_{REF} - 0.04	V_{REF}	V_{REF} + 0.04
SSTL-18 Class I, II	1.71	1.8	1.89	0.49 * V_{CCIO}	0.5 * V_{CCIO}	0.51 * V_{CCIO}	V_{REF} - 0.04	V_{REF}	V_{REF} + 0.04
SSTL-15 Class I, II	1.425	1.5	1.575	0.49 * V_{CCIO}	0.5 * V_{CCIO}	0.51 * V_{CCIO}	V_{REF} - 0.04	V_{REF}	V_{REF} + 0.04
HSTL-18 Class I, II	1.71	1.8	1.89	0.85	0.9	0.95	—	V_{CCIO}/2	—
HSTL-15 Class I, II	1.425	1.5	1.575	0.68	0.75	0.9	—	V_{CCIO}/2	—
HSTL-12 Class I, II	1.14	1.2	1.26	0.48 * V_{CCIO}	0.5 * V_{CCIO}	0.52 * V_{CCIO}	—	V_{CCIO}/2	—

Table 1–16. Single-Ended SSTL and HSTL I/O Standards Signal Specifications—Preliminary (Part 1 of 2)

I/O Standard	$V_{IL(DC)}$ (V)		$V_{IH(DC)}$ (V)		$V_{IL(AC)}$ (V)	$V_{IH(AC)}$ (V)	V_{OL} (V)	V_{OH} (V)	I_{ol} (mA)	I_{oh} (mA)
	Min	Max	Min	Max	Max	Min	Max	Min		
SSTL-2 Class I	-0.3	V_{REF} - 0.15	V_{REF} + 0.15	V_{CCIO} + 0.3	V_{REF} - 0.31	V_{REF} + 0.31	V_{TT} - 0.57	V_{TT} + 0.57	8.1	-8.1
SSTL-2 Class II	-0.3	V_{REF} - 0.15	V_{REF} + 0.15	V_{CCIO} + 0.3	V_{REF} - 0.31	V_{REF} + 0.31	V_{TT} - 0.76	V_{TT} + 0.76	16.2	-16.2
SSTL-18 Class I	-0.3	V_{REF} - 0.125	V_{REF} + 0.125	V_{CCIO} + 0.3	V_{REF} - 0.25	V_{REF} + 0.25	V_{TT} - 0.475	V_{TT} + 0.475	6.7	-6.7
SSTL-18 Class II	-0.3	V_{REF} - 0.125	V_{REF} + 0.125	V_{CCIO} + 0.3	V_{REF} - 0.25	V_{REF} + 0.25	0.28	V_{CCIO} - 0.28	13.4	-13.4

Table 1–16. Single-Ended SSTL and HSTL I/O Standards Signal Specifications—Preliminary (Part 2 of 2)

I/O Standard	$V_{IL(DC)}$ (V)		$V_{IH(DC)}$ (V)		$V_{IL(AC)}$ (V)	$V_{IH(AC)}$ (V)	V_{OL} (V)	V_{OH} (V)	I_{ol} (mA)	I_{oh} (mA)
	Min	Max	Min	Max	Max	Min	Max	Min		
SSTL-15 Class I	-0.3	V_{REF} - 0.1	V_{REF} + 0.1	V_{CCIO} + 0.3	V_{REF} - 0.175	V_{REF} + 0.175	0.2 * V_{CCIO}	0.8 * V_{CCIO}	8	-8
SSTL-15 Class II	-0.3	V_{REF} - 0.1	V_{REF} + 0.1	V_{CCIO} + 0.3	V_{REF} - 0.175	V_{REF} + 0.175	0.2 * V_{CCIO}	0.8 * V_{CCIO}	16	-16
HSTL-18 Class I	-0.3	V_{REF} - 0.1	V_{REF} + 0.1	V_{CCIO} + 0.3	V_{REF} - 0.2	V_{REF} + 0.2	0.4	V_{CCIO} - 0.4	8	-8
HSTL-18 Class II	-0.3	V_{REF} - 0.1	V_{REF} + 0.1	V_{CCIO} + 0.3	V_{REF} - 0.2	V_{REF} + 0.2	0.4	V_{CCIO} - 0.4	16	-16
HSTL-15 Class I	-0.3	V_{REF} - 0.1	V_{REF} + 0.1	V_{CCIO} + 0.3	V_{REF} - 0.2	V_{REF} + 0.2	0.4	V_{CCIO} - 0.4	8	-8
HSTL-15 Class II	-0.3	V_{REF} - 0.1	V_{REF} + 0.1	V_{CCIO} + 0.3	V_{REF} - 0.2	V_{REF} + 0.2	0.4	V_{CCIO} - 0.4	16	-16
HSTL-12 Class I	-0.15	V_{REF} - 0.08	V_{REF} + 0.08	V_{CCIO} + 0.15	V_{REF} - 0.15	V_{REF} + 0.15	0.25* V_{CCIO}	0.75* V_{CCIO}	8	-8
HSTL-12 Class II	-0.15	V_{REF} - 0.08	V_{REF} + 0.08	V_{CCIO} + 0.15	V_{REF} - 0.15	V_{REF} + 0.15	0.25* V_{CCIO}	0.75* V_{CCIO}	16	-16

Refer to "Differential I/O Standards" in Table 1–39 on page 1–32 for receiver input and transmitter output waveforms and for all differential I/O standards (LVDS, mini-LVDS, and RSDS). V_{CC_CLKIN} is the power supply for differential column clock input pins. V_{CCPD} is the power supply for row I/Os and all other column I/Os.

Table 1–17. Differential SSTL I/O Standard Specifications—Preliminary

I/O Standard	V_{CCIO} (V)			$V_{SWING(DC)}$ (V)		$V_{X(AC)}$ (V)			$V_{SWING(AC)}$ (V)		$V_{OX(AC)}$ (V)		
	Min	Typ	Max	Min	Max	Min	Typ	Max	Min	Max	Min	Typ	Max
SSTL-2 Class I, II	2.375	2.5	2.625	0.3	V_{CCIO} + 0.6	$V_{CCIO}/2$ - 0.2	—	$V_{CCIO}/2$ + 0.2	0.6	V_{CCIO} + 0.6	$V_{CCIO}/2$ - 0.15	—	$V_{CCIO}/2$ + 0.15
SSTL-18 Class I, II	1.71	1.8	1.89	0.3	V_{CCIO} + 0.6	$V_{CCIO}/2$ - 0.175	—	$V_{CCIO}/2$ + 0.175	0.5	V_{CCIO} + 0.6	$V_{CCIO}/2$ - 0.125	—	$V_{CCIO}/2$ + 0.125
SSTL-15 Class I, II	1.425	1.5	1.575	0.2	—	—	$V_{CCIO}/2$	—	0.4	—	—	$V_{CCIO}/2$	—

Table 1–18. Differential HSTL I/O Standard Specifications—Preliminary

I/O Standard	V_{CCIO} (V)			$V_{DIF(DC)}$ (V)		$V_{X(AC)}$ (V)			$V_{CM(DC)}$ (V)			$V_{DIF(AC)}$ (V)	
	Min	Typ	Max	Min	Max	Min	Typ	Max	Min	Typ	Max	Min	Max
HSTL-18 Class I, II	1.71	1.8	1.89	0.2	—	0.78	—	1.12	0.8	—	1.12	0.4	—
HSTL-15 Class I, II	1.425	1.5	1.575	0.2	—	0.68	—	0.9	0.7	—	0.9	0.4	—
HSTL-12 Class I, II	1.14	1.2	1.26	0.2	—	—	0.5* V_{CCIO}	—	0.4* V_{CCIO}	0.5* V_{CCIO}	0.6* V_{CCIO}	0.3	—

Table 1–19. Differential I/O Standard Specifications—Preliminary *(Note 1)* (Part 1 of 2)

I/O Standard	V_{CCIO} (V)			V_{ID} (mV)			$V_{ICM(DC)}$ (V)			V_{OD} (V) *(2)*			V_{OCM} (V) *(2)*		
	Min	Typ	Max	Min	Condition	Max	Min	Condition	Max	Min	Typ	Max	Min	Typ	Max
2.5V LVDS (HIO)	2.375	2.5	2.625	100	V_{CM} = 1.25 V	—	0.05 *(4)*	$D_{MAX} \leq 700$ Mbps	1.8 *(4)*	0.247	—	0.6	1.125	1.25	1.375
						—	1.05 *(4)*	$D_{MAX} > 700$ Mbps	1.55 *(4)*	—	—	—	—	—	—

Table 1–19. Differential I/O Standard Specifications—Preliminary *(Note 1)* (Part 2 of 2)

I/O Standard	V_{CCIO} (V)			V_{ID} (mV)			$V_{ICM(DC)}$ (V)			V_{OD} (V) *(2)*			V_{OCM} (V) *(2)*		
	Min	Typ	Max	Min	Condition	Max	Min	Condition	Max	Min	Typ	Max	Min	Typ	Max
2.5V LVDS (VIO)	2.375	2.5	2.625	100	V_{CM} = 1.25 V	—	0.05 *(4)*	$D_{MAX} \leq 700$ Mbps	1.8 *(4)*	0.247	—	0.6	1	1.25	1.5
						—	1.05 *(4)*	$D_{MAX} > 700$ Mbps	1.55 *(4)*	—	—	—	—	—	—
RSDS (HIO)	2.375	2.5	2.625	100	V_{CM} = 1.25 V	—	0.3	—	1.4	0.1	0.2	0.6	0.5	1.2	1.4
RSDS (VIO)	2.375	2.5	2.625	100	V_{CM} = 1.25 V	—	0.3	—	1.4	0.1	0.2	0.6	0.5	1.2	1.5
Mini-LVDS (HIO)	2.375	2.5	2.625	200	—	600	0.4	—	1.325	0.25	—	0.6	0.5	1.2	1.4
Mini-LVDS (VIO)	2.375	2.5	2.625	200	—	600	0.4	—	1.325	0.25	—	0.6	0.5	1.2	1.5
LVPECL (VIO) *(5)*	2.375 *(6)*	2.5 *(6)*	2.625 *(6)*	300	—	—	0.6	$D_{MAX} \leq 700$ Mbps	1.8 *(3)*	—	—	—	—	—	—
	—	—	—	—	—	—	0.6	$D_{MAX} > 700$ Mbps	1.6 *(3)*	—	—	—	—	—	—

Notes to Table 1–19:

(1) VIO (vertical I/O) is top and bottom I/Os; HIO (horizontal I/O) is left and right I/Os.

(2) RL range: $90 \leq RL \leq 110\ \Omega$.

(3) For $D_{MAX} > 700$ Mbps, the minimum input voltage is 0.85 V; the maximum input voltage is 1.75 V. For $F_{MAX} \leq 700$ Mbps, the minimum input voltage is 0.45 V; the maximum input voltage is 1.95 V.

(4) For data rate: $D_{MAX} > 700$ Mbps, the minimum input voltage is 1.0 V, the maximum input voltage is 1.6 V. For $D_{MAX} \leq 700$ Mbps, the minimum input voltage is 0 V, the maximum input voltage is 1.85 V.

(5) Column and row I/O banks support LVPECL I/O standards for input operation only on dedicated clock input pins. Differential clock inputs in column I/O use V_{CC_CLKIN}, which must be powered by 2.5 V. Differential clock inputs in row I/Os are powered by V_{CCPD}.

(6) The power supply for column I/O LVPECL differential clock input buffer is V_{CC_CLKIN}.

Power Consumption

Altera offers two ways to estimate power consumption for a design: the Excel-based PowerPlay EPE and the Quartus® II PowerPlay Power Analyzer feature.

Use the interactive Excel-based PowerPlay EPE prior to designing to get a magnitude estimate of the device power. The Quartus II PowerPlay Power Analyzer provides better quality estimates based on the specifics of the design after place-and-route is complete. The PowerPlay Power Analyzer can apply a combination of user-entered, simulation-derived, and estimated signal activities that, combined with detailed circuit models, can yield very accurate power estimates.

Refer to Table 1–6 on page 1–6 for supply current estimates for V_{CCPGM} and V_{CC_CLKIN}. Use the PowerPlay EPE and Power Analyzer for current estimates of the remaining power supplies.

For more information about power estimation tools, refer to the PowerPlay Early Power Estimator page on the Altera website and the *PowerPlay Power Analysis* chapter in volume 3 of the *Quartus II Handbook*.

Switching Characteristics

This section provides performance characteristics of HardCopy IV core and periphery blocks for commercial grade devices.

HardCopy IV devices are designed to meet, at minimum, the –3 speed grade of the HardCopy IV devices. Silicon characterization determines the actual performance of the HardCopy IV devices. These characteristics are designated as **Preliminary** or **Final**, as defined in the following:

- **Preliminary**—Preliminary characteristics are created using simulation results, process data, and other known parameters.
- **Final**—Final numbers are based on actual silicon characterization and testing.

These numbers reflect the actual performance of the device under worst-case silicon process, voltage, and junction temperature conditions.

Transceiver Performance Specifications

This section describes transceiver performance specifications. Table 1–20 lists HardCopy IV GX transceiver specifications.

Table 1–20. HardCopy IV GX Transceiver Specifications (Part 1 of 5)

Symbol	Parameter	Minimum	Typical	Maximum	Unit
Reference Clock					
Input frequency from REFCLK input pins	—	50	—	672	MHz
Phase frequency detector (CMU PLL and receiver CDR)	—	50	—	325	MHz
Absolute Vmax for a REFCLK pin	—	—	—	1.6	V
Operational Vmax for a REFCLK pin	—	—	—	1.5	V
Absolute Vmin for a REFCLK pin	—	-0.4	—	—	V
Rise/fall time	—	—	—	0.2	UI
Duty cycle	—	45	—	55	%
Peak-to-peak differential input voltage	—	200	—	1600	mV
Spread-spectrum modulating clock frequency	PCI Express	30	—	33	kHz
Spread-spectrum downspread	PCI Express	—	0 to -0.5%	—	—
On-chip termination resistors	—	—	100	—	Ohm
VICM (AC coupled)	—	—	1100	—	mV
VICM (DC coupled)	HCSL I/O standard for Pci Express reference clock	250	—	550	mV
Rref	—	—	2000 +/-1%	—	Ohm
Transceiver Clocks					
Calibration block clock frequency	—	10	—	125	MHz

Table 1–20. HardCopy IV GX Transceiver Specifications (Part 2 of 5)

Symbol	Parameter	Minimum	Typical	Maximum	Unit
fixedclk clock frequency	PCI Express Receiver Detect	—	125	—	MHz
reconfig_clk clock frequency	Dynamic reconfiguration clock frequency	2.5/37.5 *(2)*	—	50	—
Transceiver block minimum power-down pulse width	—	—	1	—	µs
Receiver					
Data rate (Single width, non-PMA Direct)	—	600	—	3750	Mbps
Data rate (Double width, non-PMA Direct)	—	1000	—	6500	Mbps
Data rate (Single width, PMA Direct)	—	600	—	3250	Mbps
Data rate (Double width, PMA Direct)	—	1000	—	6500	Mbps
Absolute Vmax for a receiver pin *(3)*	—	—	—	1.6	V
Operational Vmax for a receiver pin	—	—	—	1.5	V
Absolute Vmin for receiver pin	—	-0.4	—	-	V
Maximum peak-to-peak differential input voltage VID (diff p-p)	VICM = 0.82V setting	—	—	2.7	V
	VICM = 1.1V setting *(4)*	—	—	1.6	V
Minimum peak-to-peak differential input voltage VID (diff p-p)	Data Rate = 600 Mbps to 5 Gbps	100	—	—	mV
	Data Rate > 5 Gbps	165	—	—	mV
VICM	VICM = 0.82V setting	—	820	—	mV
	VICM = 1.1V setting *(4)*	—	1100	—	mV
Differential on-chip termination resistors	85-ohm setting	—	85	—	Ohm
	100-ohm setting	—	100	—	Ohm
	120-ohm setting	—	120	—	Ohm
	150-ohm setting	—	150	—	Ohm
Return loss differential mode	PCI Express	50 MHz to 1.25GHx: -10dB	—	—	—
	XAUI	100 MHz to 2.5 GHz: -10dB	—	—	—
	(OIF) CEI	100 MHz to 4.875 GHz: -8dB 4.875GHz to 10GHz: 16.6 dB/decade slope	—	—	—

Table 1–20. HardCopy IV GX Transceiver Specifications (Part 3 of 5)

Symbol	Parameter	Minimum	Typical	Maximum	Unit
Return loss common mode	PCI Express	50 MHz to 1.25GHx: -6dB	—	—	—
	XAUI	100 MHz to 2.5 GHz: -6dB	—	—	—
	(OIF) CEI	100 MHz to 4.875 GHz: -6dB 4.875GHz to 10GHz: 16.6 dB/decade slope	—	—	—
Programmable PPM detector *(5)*	—	+/- 62.5, 100, 125, 200, 250, 300, 500, 1000	—	—	ppm
Run length	—	—	80	—	Ui
Programmable equalization	—	—	—	16	dB
Signal detect/loss threshold	PCI Express (PIPE) Mode	65	—	175	mV
CDR LTR time *(6)*	—	—	—	75	µs
CDR minimum T1b *(7)*	—	15	—	—	µs
LTD lock time *(8)*	—	0	100	4000	ns
Data lock time from rx_freqlocked *(9)*	—	—	—	4000	ns
Receiver buffer and CDR offset cancellation time (per channel)	—	—	—	7872	reconfig_clk cycles
Programmable DC gain	DC Gain Setting = 0	—	0	—	dB
	DC Gain Setting = 1	—	3	—	dB
	DC Gain Setting = 2	—	6	—	dB
	DC Gain Setting = 3	—	9	—	dB
	DC Gain Setting = 4	—	12	—	dB
Transmitter					
Data rate (Single width, non-PMA Direct)	—	600	—	3750	Mbps
Data rate (Double width, non-PMA Direct)	—	1000	—	6500	Mbps
Data rate (Single width, PMA Direct) *(10)*	—	600	—	3250	Mbps
Data rate (Double width, PMA Direct) *(10)*	—	1000	—	6500	Mbps
VOCM	0.65V setting	—	650	—	mV
Differential on-chip termination resistors	85-ohm setting	—	85	—	Ohm
	100-ohm setting	—	100	—	Ohm
	120-ohm setting	—	120	—	Ohm
	150-ohm setting	—	150	—	Ohm

Table 1–20. HardCopy IV GX Transceiver Specifications (Part 4 of 5)

Symbol	Parameter	Minimum	Typical	Maximum	Unit
Return loss differential mode	PCI Express	50 MHz to 1.25GHx: -10dB	—	—	—
	XAUI	312 MHz to 625 MHz: -10dB 625MHz to 3.125GHz: -10dB/decade slope	—	—	—
	(OIF) CEI	100 MHz to 4.875 GHz: -8dB 4.875GHz to 10GHz: 16.6 dB/decade slope	—	—	—
Return loss common mode	PCI Express	50 MHz to 1.25GHx: -6dB	—	—	—
	(OIF) CEI	100 MHz to 4.875 GHz: -6dB 4.875GHz to 10GHz: 16.6 dB/decade slope	—	—	—
Rise time	—	50	—	200	ps
Fall time *(11)*	—	50	—	200	ps
Intra differential pair skew	—	—	—	15	ps
Intra-transceiver block skew x4 PMA and PCS bonded	XAUI, PCI Express (PIPE) x4, Basic x4	—	—	120	ps
Inter-transceiver block skew x8 PMA and PCS bonded	PCI Express (PIPE) x8, Basic x8	—	—	500	ps
Inter-transceiver block skew xN PMA-Only bonded *(12)*	Basic (PMA Direct) xN (For N < 18 channels located across 3 transceiver blocks)	—	*(13)*	—	ps
Inter-transceiver block skew xN PMA-Only bonded *(12)*	Basic (PMA Direct) xN (For N > or = 18 channels located across 4 transceiver blocks)	—	*(13)*	—	ps
CMU PLL0 and CMU PLL1					
Supported Data Range	—	600	—	6500	Mbps
CMU PLL lock time from pll_powerdown deassertion	—	—	—	100	µs
ATX PLL					
Supported Data Range *(14)*	/L = 1	4800-5400 and 6000-6375	—	—	Mbps
	/L = 2	2400-2700 and 3000-3187.5	—	—	Mbps
	/L = 4	1200-1350 and 1500-1593.75	—	—	Mbps
Transceiver-FPGA Fabric Interface	—	—	—	—	—
Interface speed (non-PMA Direct)	—	25	—	250	MHz
Interface speed (PMA Direct)	—	50	—	325	MHz

Table 1–20. HardCopy IV GX Transceiver Specifications (Part 5 of 5)

Symbol	Parameter	Minimum	Typical	Maximum	Unit
Digital reset pulse width	—	Minimum is 2 parallel clock cycles	—	—	—

Notes to Table 1–20:

(1) The -2X speed grade is the fastest speed grade offered in the following HardCopy IV GX devices: HC4GX15LF780N, HC4GX25LF780N, HC4GX25LF1152N, HC4GX25FF1152N, HC4GX35FF1152N, HC4GX35LF1517N and HC4GX35FF1517N.

(2) The minimum reconfig_clk frequency is 2.5MHz if the transceiver channel is configured in transmitter only mode. The minimum reconfig_clk frequency is 37.5MHz if the transceiver channel is configured in receiver only or receiver and transmitter mode. For more details, refer to HardCopy IV Dynamic Reconfiguration chapter in volume 3 of the HardCopy IV Device Handbook.

(3) The device cannot tolerate prolonged operation at this absolute maximum.

(4) The 1.1-V RX VICM setting must be used if the input serial data standard is LVDS and the link is DC coupled.

(5) The rate matcher supports only up to +/- 300ppm.

(6) Time taken to rx_pll_locked goes high from rx_analogreset deassertion. Refer to Figure 1–2.

(7) Time for which the CDR must be kept in lock-to-reference mode after rx_pll_locked goes high and before rx_locktodata is asserted in manual mode. Refer to Figure 1–2.

(8) Time taken to recover valid data after the rx_locktodata signal is asserted in manual mode. Refer to Figure 1–2.

(9) Time taken to recover valid data after the rx_freqlocked signal goes high in automatic mode. Refer to Figure 1–3.

(10) A GPLL may be required to meet PMA-HardCopy fabric interface timing above certain data rates and this requirement is same as PMA-FPGA fabric interface. Refer to section "Left/Right PLL Requirements in Basic (PMA Direct) Mode" in the *Stratix IV Transceiver Clocking* chapter in volume 2 of the *Stratix IV Device Handbook*.

(11) The Quartus II software automatically selects the appropriate slew rate depending on the configured data rate or functional mode.

(12) For applications that require low transmit lane-to-lane skew, use Basic (PMA Direct) xN to achieve PMA-Only bonding across all channels in the link. You can bond all channels on one side of the device by configuring them in Basic (PMA-Direct) xN mode. Refer to the Basic (PMA Direct) mode clocking section in the *Stratix IV Transceiver Clocking* chapter for details on clocking requirements in this mode.

(13) Pending characterization.

(14) The Quartus II software automatically selects the appropriate /L divider depending upon the configured data rate.

Figure 1–2 shows the lock time parameters in manual mode. Figure 1–3 shows the lock time parameters in automatic mode.

LTD = Lock-To-Data LTR = Lock-To-Reference

Figure 1–2. Lock Time Parameters for Manual Mode

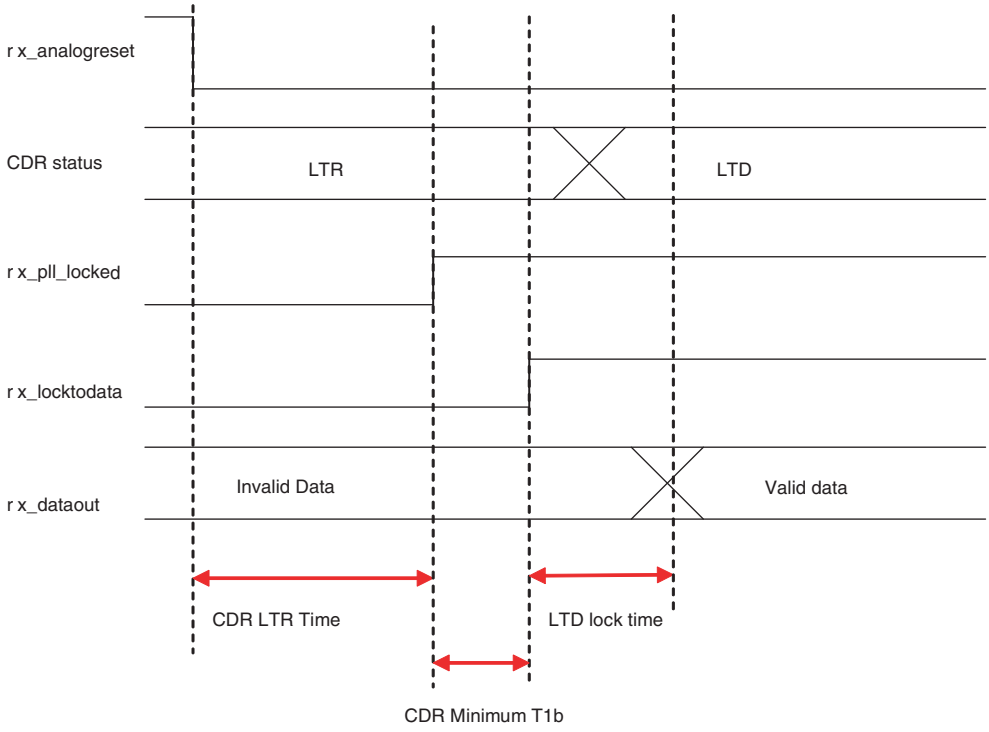


Figure 1–3. Lock Time Parameters for Automatic Mode

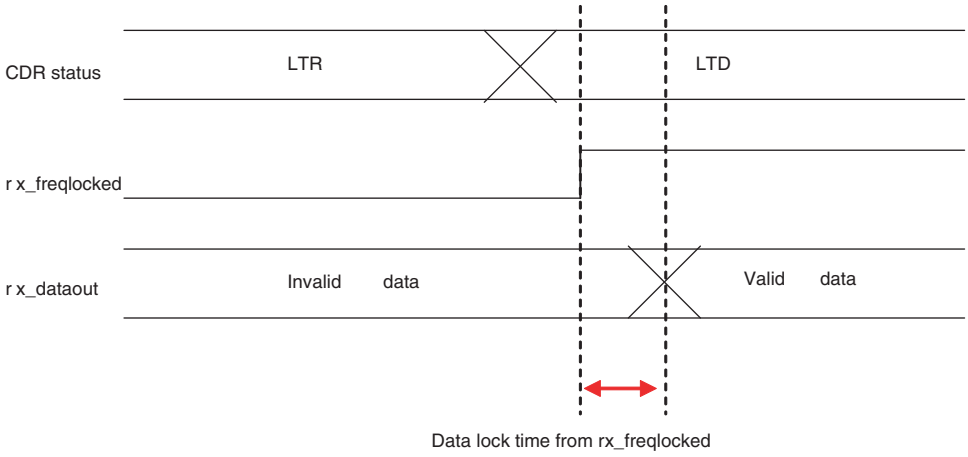


Table 1–21 through Table 1–23 show the typical VOD for various differential termination settings for HardCopy IV GX devices.

Table 1–21. Typical VOD Setting, TX Term = 85 Ohm

Symbol	VOD Setting (mV)							
	0	1	2	3	4	5	6	7
VOD differential peak-peak Typical	170	340	510	595	680	765	850	1020

Table 1–22. Typical VOD Setting, TX Term = 100 Ohm

Symbol	VOD Setting (mV)							
	0	1	2	3	4	5	6	7
VOD differential peak-peak Typical	200	400	600	700	800	900	1000	1200

Table 1–23. Typical VOD Setting, TX Term = 120 Ohm

Symbol	VOD Setting (mV)					
	0	1	2	3	4	5
VOD differential peak-peak Typical	300	600	900	1050	1200	1350

Table 1–24 shows HardCopy IV GX transceiver jitter specifications for all supported protocols.

Table 1–24. HardCopy IV GX Transceiver Block Jitter Specification *(Note 1)* and *(2)* (Part 1 of 6)

Symbol/Description	Conditions	Min	Typ	Max	Unit
SONET/SDH Transmit Jitter Generation *(3)*					
Peak-to-peak jitter at 622.08 Mbps	Pattern = PRBS23	—	—	0.1	UI
RMS jitter at 622.08 Mbps	Pattern = PRBS23	—	—	0.01	UI
Peak-to-peak jitter at 2488.32 Mbps	Pattern = PRBS23	—	—	0.1	UI
RMS jitter at 2488.32 Mbps	Pattern = PRBS23	—	—	0.01	UI
SONET/SDH Receiver Jitter Tolerance *(3)*					
Jitter tolerance at 622.08Mbps	Jitter frequency = 0.03KHz Pattern = PRBS23	> 15	> 15	> 15	UI
	Jitter frequency = 25KHz Pattern = PRBS24	> 1.5	> 1.5	> 1.5	UI
	Jitter frequency = 250KHz Pattern = PRBS25	> 0.15	> 0.15	> 0.15	UI

Table 1–24. HardCopy IV GX Transceiver Block Jitter Specification *(Note 1)* and *(2)* (Part 2 of 6)

Symbol/Description	Conditions	Min	Typ	Max	Unit
Jitter tolerance at 2488.32Mbps	Jitter frequency = 0.06KHz Pattern = PRBS23	> 15	> 15	> 15	UI
	Jitter frequency = 100KHz Pattern = PRBS24	> 1.5	> 1.5	> 1.5	UI
	Jitter frequency = 1MHz Pattern = PRBS25	> 0.15	> 0.15	> 0.15	UI
	Jitter frequency = 10MHz Pattern = PRBS26	> 0.15	> 0.15	> 0.15	UI
Fibre Channel Transmit Jitter Generation *(4)*, *(12)*					
Total jitter FC-1	Pattern = CRPAT	—	—	0.23	UI
Deterministic jitter FC-1	Pattern = CRPAT	—	—	0.11	UI
Total jitter FC-2	Pattern = CRPAT	—	—	0.33	UI
Deterministic jitter FC-2	Pattern = CRPAT	—	—	0.2	UI
Total jitter FC-4	Pattern = CRPAT	—	—	0.52	UI
Deterministic jitter FC-4	Pattern = CRPAT	—	—	0.33	UI
Fibre Channel Receiver Jitter Tolerance *(4)*, *(13)*					
Deterministic jitter FC-1	Pattern = CJTPAT	> 0.37			UI
Random jitter FC-1	Pattern = CJTPAT	> 0.31			UI
Sinusoidal jitter FC-1	Fc/25000	> 1.5			UI
	Fc/1667	> 0.1			UI
Deterministic jitter FC-2	Pattern = CJTPAT	> 0.33			UI
Random jitter FC-2	Pattern = CJTPAT	> 0.29			UI
Sinusoidal jitter FC-2	Fc/25000	> 1.5			UI
	Fc/1667	> 0.1			UI
Deterministic jitter FC-4	Pattern = CJTPAT	> 0.33			UI
Random jitter FC-4	Pattern = CJTPAT	> 0.29			UI
Sinusoidal jitter FC-4	Fc/25000	> 1.5			UI
	Fc/1667	> 0.1			UI
XAUI Transmit Jitter Generation *(5)*					
Total jitter at 3.125Gbps	Pattern = CJPAT	—	—	0.3	UI
Deterministic jitter at 3.125Gbps	Pattern = CJPAT	—	—	0.17	UI
XAUI Receiver Jitter Tolerance *(5)*					
Total jitter	—	> 0.65			UI
Deterministic jitter	—	> 0.37			UI
Peak-to-peak jitter	Jitter frequency = 22.1KHz	> 8.5			UI

Table 1–24. HardCopy IV GX Transceiver Block Jitter Specification *(Note 1)* and *(2)* (Part 3 of 6)

Symbol/Description	Conditions	Min	Typ	Max	Unit
Peak-to-peak jitter	Jitter frequency = 1.875MHz	> 0.1			UI
Peak-to-peak jitter	Jitter frequency = 20MHz	> 0.1			UI
PCI Express Transmit Jitter Generation *(6)*					
Total jitter at 2.5Gbps (Gen1)	Compliance pattern	—	—	0.25	UI
Total jitter at 5Gbps (Gen2)	Compliance pattern	—	—	—	UI
PCI Express Receiver Jitter Tolerance *(6)*					
Total jitter at 2.5Gbps (Gen1)	Compliance pattern	> 0.6			UI
Total jitter at 2.5Gbps (Gen2)	Compliance pattern	—	—	—	UI
Serial RapidIO Transmit Jitter Generation *(7)*					
Deterministic jitter (peak-to-peak)	Data Rate = 1.25, 2.5, 3.125 Gbps Pattern = CJPAT	—	—	0.17	UI
Total jitter (peak-to-peak)	Data Rate = 1.25, 2.5, 3.125 Gbps Pattern = CJPAT	—	—	0.35	UI
Serial RapidIO Receiver Jitter Tolerance *(7)*					
Deterministic jitter (peak-to-peak)	Data Rate = 1.25, 2.5, 3.125 Gbps Pattern = CJPAT	> 0.37			UI
Combined deterministic and random jitter tolerance (peak-to-peak)	Data Rate = 1.25, 2.5, 3.125 Gbps Pattern = CJPAT	> 0.55			UI
Sinusoidal jitter tolerance (peak-to-peak)	Jitter Frequency = 22.1KHz Data Rate = 1.25, 2.5, 3.125 Gbps Pattern = CJPAT	> 8.5			UI
	Jitter Frequency = 1.875MHz Data Rate = 1.25, 2.5, 3.125 Gbps Pattern = CJPAT	> 0.1			UI
	Jitter Frequency = 20MHz Data Rate = 1.25, 2.5, 3.125 Gbps Pattern = CJPAT	> 0.1			UI
GIGE Transmit Jitter Generation *(8)*					
Deterministic jitter (peak-to-peak)	Pattern = CRPAT	—	—	0.14	UI
Total jitter (peak-to-peak)	Pattern = CRPAT	—	—	0.279	UI
GIGE Receiver Jitter Tolerance *(8)*					
Deterministic jitter (peak-to-peak)	Pattern = CJPAT	> 0.4			UI
Combined deterministic and random jitter tolerance (peak-to-peak)	Pattern = CJPAT	> 0.66			UI

Table 1–24. HardCopy IV GX Transceiver Block Jitter Specification *(Note 1)* and *(2)* (Part 4 of 6)

Symbol/Description	Conditions	Min	Typ	Max	Unit
HiGig Transmit Jitter Generation *(9)*					
Deterministic jitter (peak-to-peak)	Data Rate = 3.75Gbps Pattern = CJPAT	—	—	—	UI
Total jitter (peak-to-peak)	Data Rate = 3.75Gbps Pattern = CJPAT	—	—	—	UI
HiGig Receiver Jitter Tolerance *(9)*					
Deterministic jitter tolerance (peak-to-peak)	Data Rate = 3.75Gbps Pattern = CJPAT	—	—	—	UI
Combined deterministic and random jitter tolerance (peak-to-peak)	Data Rate = 3.75Gbps Pattern = CJPAT	—	—	—	UI
Sinusoidal jitter tolerance (peak-to-peak)	Jitter Frequency = 22.1KHz Data Rate = 3.75 Gbps Pattern = CJPAT	—	—	—	UI
	Jitter Frequency = 1.875MHz Data Rate = 3.75 Gbps Pattern = CJPAT	—	—	—	UI
	Jitter Frequency = 20MHz Data Rate = 3.75 Gbps Pattern = CJPAT	—	—	—	UI
(OIF) CEI Transmitter Jitter Generation *(10)*					
Total jitter (peak-to-peak)	Data Rate = 6.375 Gbps Pattern = PRBS15 BER = 10exp-12	—	—	—	UI
(OIF) CEI Receiver Jitter Tolerance *(10)*					
Deterministic jitter tolerance (peak-to-peak)	Data Rate = 6.375 Gbps Pattern = PRBS31 BER = 10exp-12	—	—	—	UI
Combined deterministic and random jitter tolerance (peak-to-peak)	Data Rate = 6.375 Gbps Pattern = PRBS31 BER = 10exp-13	—	—	—	UI

Table 1–24. HardCopy IV GX Transceiver Block Jitter Specification *(Note 1)* and *(2)* (Part 5 of 6)

Symbol/Description	Conditions	Min	Typ	Max	Unit
Sinusoidal jitter tolerance (peak-to-peak)	Jitter Frequency = 38.2KHz Data Rate = 6.375 Gbps Pattern = PRBS31 BER = 10exp-12	—	—	—	UI
	Jitter Frequency = 3.82MHz Data Rate = 6.375 Gbps Pattern = PRBS31 BER = 10exp-12	—	—	—	UI
	Jitter Frequency = 20MHz Data Rate = 6.375 Gbps Pattern = PRBS31 BER = 10exp-12	—	—	—	UI
SDI Transmitter Jitter Generation *(11)*					
Alignment jitter (peak-to-peak)	Data Rate = 1.485Gbps (HD) Pattern = Color Bar Low-Frequency Roll-Off = 100KHz	0.2	—	—	UI
	Data Rate = 2.97Gbps (3G) Pattern = Color Bar Low-Frequency Roll-Off = 100KHz	0.3	—	—	UI
SDI Receiver Jitter Tolerance *(11)*					
Sinusoidal jitter tolerance (peak-to-peak)	Jitter Frequency = 15KHz Data Rate = 2.97Gbps (3G) Pattern = Single Line Scramble Color Bar	> 2			UI
	Jitter Frequency = 100KHz Data Rate = 2.97Gbps (3G) Pattern = Single Line Scramble Color Bar	> 0.3			UI
	Jitter Frequency = 148.5MHz Data Rate = 2.97Gbps (3G) Pattern = Single Line Scramble Color Bar	> 0.3			UI

Table 1–24. HardCopy IV GX Transceiver Block Jitter Specification *(Note 1)* and *(2)* (Part 6 of 6)

Symbol/Description	Conditions	Min	Typ	Max	Unit
Sinusoidal jitter tolerance (peak-to-peak)	Jitter Frequency = 20KHz Data Rate = 1.485Gbps (HD) Pattern = 75% Color Bar	> 1			UI
	Jitter Frequency = 100KHz Data Rate = 1.485Gbps (HD) Pattern = 75% Color Bar	> 0.2			UI
	Jitter Frequency = 148.5MHz Data Rate = 1.485Gbps (HD) Pattern = 75% Color Bar	> 0.2			UI

Notes to Table 1–24:

(1) Dedicated refclk pins were used to drive the input reference clocks.
(2) Jitter numbers specified are valid for the stated conditions only.
(3) The jitter numbers for SONET/SDH are compliant to the GR-253-CORE Issue 3 Specification.
(4) The jitter numbers for Fibre Channel are compliant to the FC-PI-4 Specification revision 6.1.0.
(5) The jitter number for XAUI are compliant to the IEEE802.3ae-2002 Specification.
(6) The jitter numbers for PCI Express are compliant to the PCIe Base Specification 2.0.
(7) The jitter numbers for Serial RapidIO are compliant to the RapidIO Specification 1.3.
(8) The jitter numbers for GIGE are compliant to the IEEE802.3-2002 Specification.
(9) The jitter numbers for HiGig are compliant to the IEEE802.3ae-2002 Specification.
(10) The jitter numbers for (OIF) CEI are compliant to the OIF-CEI-02.0 Specification.
(11) The HD-SDI and 3G-SDI jitter numbers are compliant to the SMPTE292M and SMPTE424M Specifications.
(12) The fibre channel transmitter jitter generation numbers are compliant to the specification at δ_T interoperability point.
(13) The fibre channel receiver jitter tolerance numbers are compliant to the specification at δ_R interoperability point.

Core Performance Specifications

This section describes the clock tree, PLL, DSP, TriMatrix, and configuration and JTAG specifications.

Clock Tree Specifications

Table 1–25 lists clock tree performance specifications for the logic array, DSP blocks, and TriMatrix memory blocks for HardCopy IV devices.

Table 1–25. HardCopy IV Clock Tree Performance—Preliminary *(Note 1)*

Device	Commercial Grade (MHz)	Unit
HC4E25	600	MHz
HC4E35	600	MHz

Note to Table 1–25:

(1) Pending silicon characterization.

PLL Specifications

Table 1–26 describes the HardCopy IV PLL specifications when operating in both the commercial junction temperature range (0° to 85°C) and the industrial junction temperature range (–40° to 100°C). For a PLL block diagram, refer to the "PLL Specifications" row in Table 1–39 on page 1–32.

Table 1–26. HardCopy IV PLL Specifications—Preliminary (Part 1 of 2) *(Note 1)*

Symbol	Parameter	Min	Typ	Max	Unit
f_{IN}	Input clock frequency	5	—	717 *(2)*	MHz
f_{INPFD}	Input frequency to the PFD	5	—	325	MHz
f_{VCO}	PLL VCO operating range	600	—	1300	MHz
$t_{EINDUTY}$	Input clock or external feedback clock input duty cycle	40	—	60	%
f_{OUT}	Output frequency for internal global or regional clock	—	—	600 *(3)*	MHz
f_{OUT_EXT}	Output frequency for external clock input (–3 speed grade)	—	—	717 *(3)*	MHz
$t_{OUTDUTY}$	Duty cycle for external clock output (when set to 50%)	45	50	55	%
t_{FCOMP}	External feedback clock compensation time	—	—	10	ns
$t_{CONFIGPLL}$	Time required to reconfigure PLL scan chain	—	—	—	`scanclk` cycles
$t_{CONFIGPHASE}$	Time required or reconfigure phase shift	—	—	—	`scanclk` cycles
$f_{SCANCLK}$	`scanclk` frequency	—	—	100	MHz
t_{LOCK}	Time required to lock from end of device power	—	—	—	ms
t_{DLOCK}	Time required to lock dynamically (after switchover or reconfiguring any non-post-scale counters/delays)	—	—	—	ms
f_{CLBW}	PLL closed-loop low bandwidth	—	—	—	MHz
	PLL closed-loop medium bandwidth	—	—	—	MHz
	PLL closed-loop high bandwidth *(6)*	—	—	—	MHz
t_{PLL_PSERR}	Accuracy of PLL phase shift	—	—	—	ps
t_{ARESET}	Minimum pulse width on a reset signal	10	—	—	ns
t_{INCCJ} *(4)*	Input clock cycle to cycle jitter ($F_{REF} \geq 100$ MHz)	—	—	—	UI (p-p)
	Input clock cycle to cycle jitter ($F_{REF} < 100$ MHz)	—	—	—	ps (p-p)
t_{OUTPJ_DC} *(5)*	Period jitter for dedicated clock output ($F_{OUT} \geq 100$ MHz)	—	—	—	ps (p-p)
	Period jitter for dedicated clock output ($F_{OUT} < 100$ MHz)	—	—	—	mUI (p-p)
t_{OUTCCJ_DC} *(5)*	Cycle-to-cycle jitter for dedicated clock output ($F_{OUT} \geq 100$ MHz)	—	—	—	ps (p-p)
	Cycle-to-cycle jitter for dedicated clock output ($F_{OUT} < 100$ MHz)	—	—	—	mUI (p-p)
t_{OUTPJ_IO} *(5)*	Period Jitter for clock output on regular IO ($F_{OUT} \geq 100$ MHz)	—	—	—	ps (p-p)
	Period Jitter for clock output on regular IO ($F_{OUT} < 100$ MHz)	—	—	—	mUI (p-p)
t_{OUTCCJ_IO} *(5)*	Cycle-to-cycle jitter for clock output on regular IO ($F_{OUT} \geq 100$ MHz)	—	—	—	ps (p-p)
	Cycle-to-cycle jitter for clock output on regular IO ($F_{OUT} < 100$ MHz)	—	—	—	mUI (p-p)

Table 1–26. HardCopy IV PLL Specifications—Preliminary (Part 2 of 2) *(Note 1)*

Symbol	Parameter	Min	Typ	Max	Unit
f_{DRIFT}	Frequency drift after PFDENA is disabled for duration of 100 us	—	—	±10	%

Notes to Table 1–26:

(1) Pending silicon characterization.

(2) This specification is limited in Quartus II software by the I/O maximum frequency. The maximum I/O frequency is different for each I/O standard.

(3) This specification is limited by the lower of the two: I/O F_{MAX} or F_{OUT} of the PLL.

(4) A high input jitter directly affects the PLL output jitter. To have low PLL output clock jitter, you must provide a clean clock source, which is less than 200 ps.

(5) Peak-to-peak jitter with a probability level of 10^{-12} (14 sigma, 99.99999999974404% confidence level). The output jitter specification applies to the intrinsic jitter of the PLL, when an input jitter of 30 ps is applied. The external memory interface clock output jitter specifications use a different measurement method and are available in Table 1–38.

(6) High bandwidth PLL settings are not supported in external feedback mode.

DSP Block Specifications

Table 1–27 describes the HardCopy IV DSP block performance specifications.

Table 1–27. HardCopy IV DSP Block Performance Specifications—Preliminary *(Note 1)*, *(2)*

Mode	Number of Multipliers	Max	Unit
9×9-bit multiplier	1	410	MHz
12×12-bit multiplier	1	410	MHz
18×18-bit multiplier	1	495	MHz
36×36-bit multiplier	1	365	MHz
18×18-bit multiply accumulator	4	390	MHz
18×18-bit multiply adder	4	405	MHz
18×18-bit multiply adder-signed full precision	2	405	MHz
18×18-bit multiply adder with loopback *(3)*	2	405	MHz
36-bit shift (32 bit data)	1	390	MHz
Double mode	1	365	MHz

Notes to Table 1–27:

(1) Maximum is for fully pipelined block with **Round** and **Saturation** disabled.

(2) Pending silicon characterization.

(3) Maximum is for non-pipelined block with loopback input registers disabled and **Round** and **Saturation** disabled.

TriMatrix Memory Block Specifications

Table 1–28 describes the HardCopy IV TriMatrix memory block specifications.

Table 1–28. HardCopy IV TriMatrix Memory Block Performance Specifications—Preliminary (Part 1 of 2) *(Note 1)*

Memory	Mode	TriMatrix Memory	Max	Unit
MLAB	Single port 64×10	1	500	MHz
	Simple dual-port 32×20 single clock	1	500	MHz
	Simple dual-port 64×10 single clock	1	500	MHz

Table 1–28. HardCopy IV TriMatrix Memory Block Performance Specifications—Preliminary (Part 2 of 2) *(Note 1)*

Memory	Mode	TriMatrix Memory	Max	Unit
M9K Block	Single-port 256×36	1	575	MHz
	Simple dual-port 256×36 single CLK	1	575	MHz
	True dual port 512×18 single CLK	1	575	MHz
M144K	Single-port 2K×72	1	460	MHz
	Simple dual-port 2K×72 dual CLK	1	460	MHz
	Simple dual-port 2K×64 dual CLK (with ECC)	1	250	MHz
	True dual-port 4K×36 dual CLK	1	460	MHz

Note to Table 1–28:

(1) Pending silicon characterization.

JTAG Specification

Table 1–29 shows the JTAG timing parameters and values for HardCopy IV devices. Refer to the "HIGH-SPEED I/O Block" row in Table 1–39 on page 1–32 for JTAG timing requirements.

Table 1–29. HardCopy IV JTAG Timing Parameters and Values—Preliminary

Symbol	Description	Min	Max	Unit
t_{JCP}	TCK clock period	30	—	ns
t_{JCH}	TCK clock high time	14	—	ns
t_{JCL}	TCK clock low time	14	—	ns
$t_{JPSU\ (TDI)}$	TDI JTAG port setup time	1	—	ns
$t_{JPSU\ (TMS)}$	TMS JTAG port setup time	3	—	ns
t_{JPH}	JTAG port hold time	5	—	ns
t_{JPCO}	JTAG port clock to output	—	11 *(1)*	ns
t_{JPZX}	JTAG port high impedance to valid output	—	14 *(1)*	ns
t_{JPXZ}	JTAG port valid output to high impedance	—	14 *(1)*	ns

Note to Table 1–29:

(1) A 1 ns adder is required for each V_{CCIO} voltage step down from 3.0 V. For example, t_{JPCO} = 12 ns if V_{CCIO} of the TDO I/O bank = 2.5 V, or 13 ns if it equals 1.8 V.

Periphery Performance

This section describes periphery performance, including high-speed I/O and external memory interface, and OCT calibration block specifications.

High-Speed I/O Specification

Table 1–30 shows the high-speed I/O timing for HardCopy IV devices.

Table 1–30. High-Speed I/O Specifications—Preliminary (Part 1 of 2) *(Note 1), (2), (3)*

Symbol	Conditions	Min	Typ	Max	Unit
f_{HSCLK} (input clock frequency)	Clock boost factor W = 1 to 40 *(4)*	5	—	717	MHz
f_{HSCLK_OUT} (output clock frequency)		5	—	717	MHz

Table 1–30. High-Speed I/O Specifications—Preliminary (Part 2 of 2) *(Note 1), (2), (3)*

Symbol	Conditions	Min	Typ	Max	Unit
Transmitter					
Dedicated LVDS—f_{HSDR} (data rate)	SERDES factor J = 3 to 10	150	—	1250	Mbps
	SERDES factor J = 2, Uses DDR Registers	*(5)*	—	1250	Mbps
	SERDES factor J = 1, Uses SDR Register	*(5)*	—	717	Mbps
LVDS_E_3R—$f_{HSDRDPA}$ (data rate)	SERDES factor J =4 to 10	*(5)*	—	1000	Mbps
LVDS_E_1R—$f_{HSDRDPA}$ (data rate)		*(5)*	—	200	Mbps
$t_{x\ Jitter}$	Total Jitter for Data Rate, 600Mbps - 1.6Gbps	—	—	160	ps
	Total Jitter for Data Rate, < 600Mbps	—	—	0.1	UI
t_{DUTY}	Tx output clock duty cycle	45	50	55	%
t_{RISE} & t_{FALL}	Dedicated LVDS	—	—	200	ps
	LVDS_E_3R	—	—	350	ps
	LVDS_E_1R	—	—	500	ps
TCCS	Dedicated LVDS	—	—	100	ps
	LVDS_E_3R/ LVDS_E_1R	—	—	250	ps
Receiver					
$f_{HSDRDPA}$ (data rate)	SERDES factor J = 3 to 10	150	—	1250	Mbps
DPA Mode					
DPA run length	—	—	—	*(6)*	UI
Soft CDR mode					
Soft-CDR PPM tolerance	—	—	—	*(6)*	PPM
Non DPA Mode					
Sampling Window	All differential I/O standards	—	—	*(6)*	ps

Notes to Table 1–30:

(1) Numbers are preliminary pending characterization.
(2) When J = 3 to 10, the SERDES block is used.
(3) When J = 1 or 2, the SERDES block is bypassed.
(4) Clock Boost Factor (Ω) is the ratio between the input data rate to the input clock rate.
(5) The minimum specification is dependent on the clock source (for example, PLL and clock pin) and the clock routing resource (global, regional, or local) is used.
(6) Pending silicon characterization.

Table 1–31 shows the DPA lock time specifications for HardCopy IV devices.

Table 1–31. DPA Lock Time Specifications—Preliminary *(Note 1)*, *(2)*, *(3)* (Part 1 of 2)

Standard	Training Pattern	Number of Data Transitions in one repetition of training pattern	Number of repetition per 256 data transition *(4)*	Condition	Min	Typ	Max
SPI-4	00000000001111111111	2	128	without DPA PLL calibration	256 data transitions	—	—
				with DPA PLL calibration	3×256 data transitions + 2×96 slow clock cycles *(5)*, *(6)*	—	—

Table 1–31. DPA Lock Time Specifications—Preliminary *(Note 1)*, *(2)*, *(3)* (Part 2 of 2)

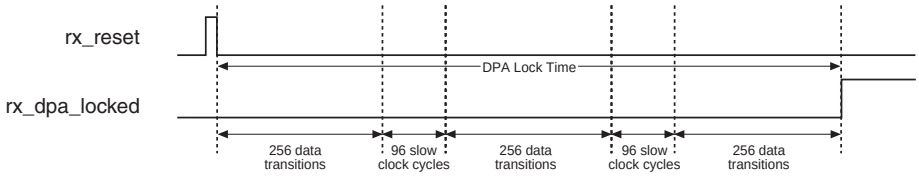
Standard	Training Pattern	Number of Data Transitions in one repetition of training pattern	Number of repetition per 256 data transition *(4)*	Condition	Min	Typ	Max
Parallel Rapid I/O	00001111	2	128	without DPA PLL calibration	256 data transitions	—	—
				with DPA PLL calibration	3×256 data transitions + 2×96 slow clock cycles *(5)*, *(6)*	—	—
	10010000	4	64	without DPA PLL calibration	256 data transitions	—	—
				with DPA PLL calibration	3×256 data transitions + 2×96 slow clock cycles *(5)*, *(6)*	—	—
Misc.	10101010	8	32	without DPA PLL calibration	256 data transitions	—	—
				with DPA PLL calibration	3×256 data transitions + 2×96 slow clock cycles *(5)*, *(6)*	—	—
	01010101	8	32	without DPA PLL calibration	256 data transitions	—	—
				with DPA PLL calibration	3×256 data transitions + 2×96 slow clock cycles *(5)*, *(6)*	—	—

Notes to Table 1–31:

(1) The DPA lock time is for one channel.
(2) One data transition is defined as a 0-to-1 or 1-to-0 transition.
(3) The DPA lock time stated in the table applies to both commercial and industrial grade.
(4) This is the number of repetition for the stated training pattern to achieve 256 data transitions.
(5) Slow clock = data rate (MHz)/deserialization factor.
(6) The DPA lock time with **DPA PLL Calibration** enabled is preliminary.

Figure 1–4 shows the DPA lock time specifications with DPA PLL calibration enabled.

Figure 1–4. DPA Lock Time Specification with DPA PLL Calibration Enabled



External Memory Interface Specifications

Table 1–32 and Table 1–33 list the external memory interface specifications for the HardCopy IV device family. Use these tables for memory interface timing analysis.

Table 1–32. HardCopy IV Maximum Clock Rate Support for External Memory Interfaces with Half-Rate Controller—Preliminary *(Note 1)* (Part 1 of 2)

Memory Standards	MHz	
	Top and Bottom I/O Banks	Left and Right I/O Banks
DDR3 SDRAM	533 *(2)*	533 *(2)*
DDR2 SDRAM	333	333
DDR SDRAM	200	200
QDRII+ SRAM	350	350

Table 1–32. HardCopy IV Maximum Clock Rate Support for External Memory Interfaces with Half-Rate Controller—Preliminary *(Note 1)* (Part 2 of 2)

Memory Standards	MHz	
	Top and Bottom I/O Banks	Left and Right I/O Banks
QDRII SRAM	300	300
RLDRAM II	400 *(3)*	400 *(3)*

Notes to Table 1–32:

(1) Pending silicon characterization.

(2) Pending IP support. Actual achievable performance is based on design and system-specific factors. For 533MHz DDR3 support, please contact Altera.

(3) Pending IP support. Actual achievable performance is based on design and system-specific factors. For 400MHz RLDRAM II support, please contact Altera.

Table 1–33. HardCopy IV Maximum Clock Rate Support for External Memory Interfaces with Full-Rate Controller—Preliminary *(Note 1)*

Memory Standards	MHz	
	Top and Bottom I/O Banks	Left and Right I/O Banks
DDR2 SDRAM	267	267
DDR SDRAM	200	200

Notes to Table 1–33:

(1) Pending silicon characterization.

External Memory I/O Timing Specifications

Table 1–34 and Table 1–35 list HardCopy IV device timing uncertainties on the read and write data paths. Use these specifications to determine timing margins for source synchronous paths between a HardCopy IV device and an external memory device.

Refer to "SW (sampling window)" in Table 1–39 on page 1–32.

Table 1–34. Sampling Window (SW) - Read Side—Preliminary *(Note 1)* (Part 1 of 2)

Location *(2)*	Memory Type	Sampling Window (ps)	
		Setup	Hold
VIO	DDR3 (>400MHz) *(3)*	282	56
	DDR3	344	85
	DDR2	213	162
	DDR1	236	178
	QDRII/II+	218	203
	RLDRAM II (>333MHz) *(4)*	198	183

Table 1–34. Sampling Window (SW) - Read Side—Preliminary *(Note 1)* (Part 2 of 2)

Location *(2)*	Memory Type	Sampling Window (ps)	
		Setup	Hold
HIO	DDR3 (>400MHz) *(3)*	282	56
	DDR3	344	85
	DDR2	213	162
	DDR1	236	178
	QDRII/II+	218	203
	RLDRAM II (>333MHz) *(4)*	198	183

Note to Table 1–34:

(1) Pending silicon Characterization.

(2) VIO (vertical I/O) refers to I/Os in the top and bottom banks; HIO (horizontal I/O) refers to I/Os in the left and right banks.

(3) Pending IP support. Actual achievable performance is based on design and system-specific factors. For DDR3 > 400MHz support, contact Altera.

(4) Pending IP support. Actual achievable performance is based on design and system-specific factors. For RLDRAM II > 333MHz support, contact Altera.

Table 1–35. Transmitter Channel-to-Channel Skew (TCCS)—Write Side *(Note 1)*

Location *(2)*	Memory Type	TCCS (ps)	
		Lead	Lag
VIO	DDR3 (>400MHz) *(3)*	234	286
	DDR3	344	347
	DDR2	270	380
	DDR1	275	396
	QDRII/II+	294	408
	RLDRAM II (>333MHz) *(4)*	346	356
HIO	DDR3 (>400MHz) *(3)*	282	56
	DDR3	344	347
	DDR2	270	380
	DDR1	275	396
	QDRII/II+	294	408
	RLDRAM II (>333MHz) *(4)*	346	356

Notes to Table 1–35:

(1) Pending silicon characterization.

(2) VIO (vertical I/O) refers to I/Os in the top and bottom banks; HIO (horizontal I/O) refers to I/Os in the left and right banks.

(3) Pending IP support. Actual achievable performance is based on design and system-specific factors. For DDR3 > 400 MHz support, contact Altera.

(4) Pending IP support. Actual achievable performance is based on design and system-specific factors. For RLDRAM II > 333 MHz support, contact Altera.

DLL and DQS Logic Block Specifications

Table 1–36 describes the delay-locked loop (DLL) frequency range specifications for HardCopy IV devices.

Table 1–36. HardCopy IV DLL Frequency Range Specifications—Preliminary *(Note 1)*

Frequency Mode	Frequency Range (MHz)	Available Phase Shift	DQS Delay Buffer Mode *(2)*	Number of Delay Chains
0	90-140	45°, 90°, 135°, 180°	Low	16
1	120-190	30°, 60, 90°, 120°	Low	12
2	150-230	36°, 72°, 108°, 144°	Low	10
3	180-290	45°, 90°, 135°, 180°	Low	8
4	240-350	30°, 60, 90°, 120°	High	12
5	290-420	36°, 72°, 108°, 144°	High	10
6	360-540 *(3)*	45°, 90°, 135°, 180°	High	8

Note to Table 1–36:

(1) Pending silicon characterization

(2) Low indicates 6-bit DQS delay setting, high indicates 5-bit DQS delay setting.

(3) Frequency > 530 MHz on Mode 6 is to for 533 MHz DDR3 support. Pending IP support. Actual achievable performance is based on design and system-specific factors. For DDR3 > 400 MHz, please contact Altera.

OCT Calibration Block Specifications

Table 1–37 describes the OCT calibration block specifications for HardCopy IV devices.

Table 1–37. OCT Calibration Block Specifications—Preliminary

Symbol	Description	Min	Typ	Max	Unit
OCTUSRCLK	Clock required by OCT calibration blocks	—	—	20	MHz
T_{OCTCAL}	Number of OCTUSRCLK clock cycles required for OCT R_S/R_T calibration	—	1000	—	Cycles
$T_{OCTSHIFT}$	Number of OCTUSRCLK clock cycles required for OCT code to shift out	—	28	—	Cycles
T_{RS_RT}	Time required to dynamically switch from R_S to R_T	—	2.5	—	ns

Duty Cycle Distortion (DCD) Specifications

Table 1–38 lists the worst case DCD for HardCopy IV devices.

Table 1–38. DCD on HardCopy IV I/O Pins—Preliminary

Symbol	Min	Max	Unit
Output Duty Cycle	45	55	%

I/O Timing Model

The I/O timing specifications for HardCopy IV devices will be available in a future revision of this chapter.

Glossary

Table 1–39 shows the glossary for this chapter.

Table 1–39. Glossary Table (Part 1 of 4)

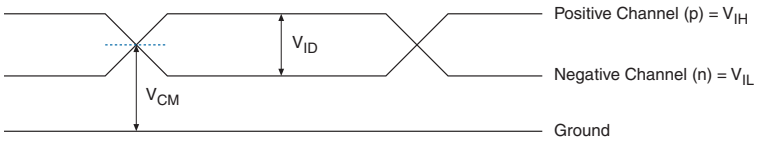
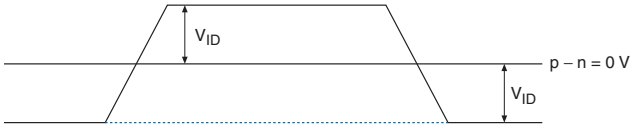
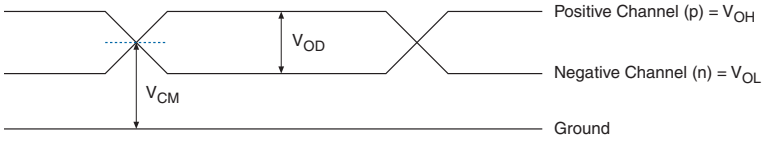
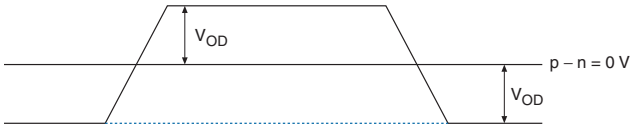
Letter	Subject	Definitions
A	—	—
B	—	—
C	—	—
D	Differential I/O Standards	*Receiver Input Waveforms* **Single-Ended Waveform**  **Differential Waveform**  *Transmitter Output Waveforms* **Single-Ended Waveform**  **Differential Waveform** 
E	—	—
F	f_{HSCLK}	Left/Right PLL input clock frequency.
	f_{HSDR}	HIGH-SPEED I/O Block: Maximum/minimum LVDS data transfer rate (f_{HSDR} = 1/TUI), non-DPA.
	$f_{HSDRDPA}$	HIGH-SPEED I/O Block: Maximum/minimum LVDS data transfer rate ($f_{HSDRDPA}$ = 1/TUI), DPA.
G	—	—
H	—	—
I	—	—

Table 1–39. Glossary Table (Part 2 of 4)

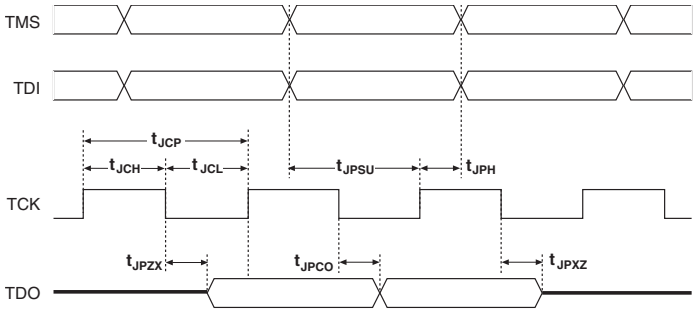
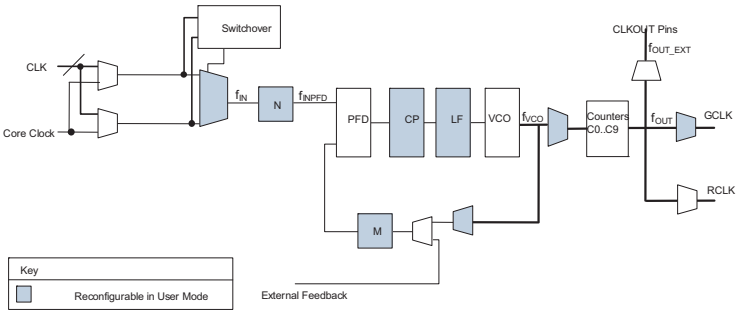
Letter	Subject	Definitions
	J	HIGH-SPEED I/O Block: Deserialization factor (width of parallel data bus).
J	JTAG Timing Specifications	JTAG Timing Specifications are in the following figure: 
K	—	—
L	—	—
M	—	—
N	—	—
O	—	—
P	**PLL Specifications**	The following block diagram highlights the PLL specification parameters: **Diagram of PLL Specifications** *(1)*  **Note:** (1) `Core Clock` can only be fed by dedicated clock input pins or PLL outputs.
Q	—	—
R	R_L	Receiver differential input discrete resistor (external to HardCopy IV device).

Table 1–39. Glossary Table (Part 3 of 4)

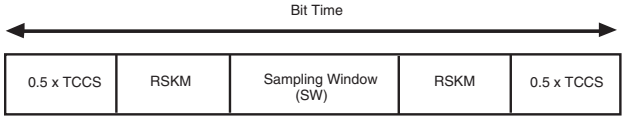
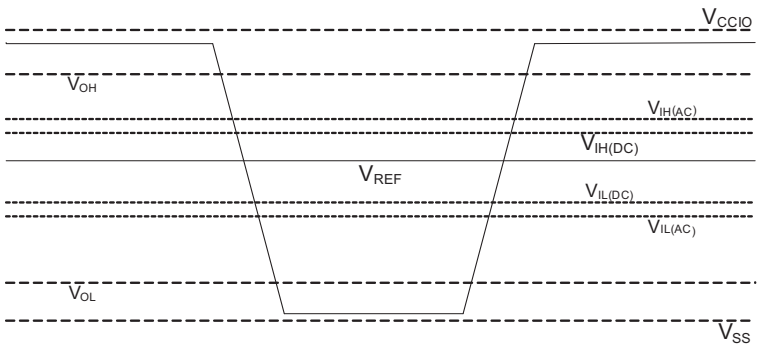
Letter	Subject	Definitions
S	**SW (sampling window)**	The period of time during which the data must be valid in order to capture it correctly. The setup and hold times determine the ideal strobe position within the sampling window, as shown in the following figure:  *Timing Diagram*
	Single-ended voltage referenced I/O standard	The JEDEC standard for SSTI and HSTL I/O defines both the AC and DC input signal values. The AC values indicate the voltage levels at which the receiver must meet its timing specifications. The DC values indicate the voltage levels at which the final logic state of the receiver is unambiguously defined. Once the receiver input has crossed the AC value, the receiver changes to the new logic state. The new logic state is then maintained as long as the input stays beyond the AC threshold. This approach is intended to provide predictable receiver timing in the presence of input waveform ringing as, shown in the following figure: *Single-Ended Voltage Referenced I/O Standard* 
T	t_C	High-speed receiver/transmitter input and output clock period.
	TCCS (channel-to-channel-skew)	The timing difference between the fastest and slowest output edges, including t_{co} variation and clock skew, across channels driven by the same PLL. The clock is included in the TCCS measurement (refer to the *Timing Diagram* figure under **S** in this table).
	t_{DUTY}	HIGH-SPEED I/O Block: Duty cycle on high-speed transmitter output clock.
		Timing Unit Interval (TUI) The timing budget allowed for skew, propagation delays, and data sampling window. (TUI = 1/(Receiver Input Clock Frequency Multiplication Factor) = t_c/w)
	t_{FALL}	Signal high-to-low transition time (80-20%)
	t_{INCCJ}	Cycle-to-cycle jitter tolerance on PLL clock input
	t_{OUTPJ_IO}	Period jitter on general purpose I/O driven by a PLL
	t_{OUTPJ_DC}	Period jitter on dedicated clock output driven by a PLL
	t_{RISE}	Signal low-to-high transition time (20-80%)
U	—	—

Table 1–39. Glossary Table (Part 4 of 4)

Letter	Subject	Definitions
V	$V_{CM(DC)}$	DC common mode input voltage.
	V_{ICM}	Input common mode voltage: The common mode of the differential signal at the receiver.
	V_{ID}	Input differential voltage swing: The difference in voltage between the positive and complementary conductors of a differential transmission at the receiver.
	$V_{DIF(AC)}$	AC differential input voltage: Minimum AC input differential voltage required for switching.
	$V_{DIF(DC)}$	DC differential input voltage: Minimum DC input differential voltage required for switching.
	V_{IH}	Voltage input high: The minimum positive voltage applied to the input that is accepted by the device as a logic high.
	$V_{IH(AC)}$	High-level AC input voltage
	$V_{IH(DC)}$	High-level DC input voltage
	V_{IL}	Voltage input low: The maximum positive voltage applied to the input that is accepted by the device as a logic low.
	$V_{IL(AC)}$	Low-level AC input voltage
	$V_{IL(DC)}$	Low-level DC input voltage
	V_{OCM}	Output common mode voltage: The common mode of the differential signal at the transmitter.
	V_{OD}	Output differential voltage swing: The difference in voltage between the positive and complementary conductors of a differential transmission at the transmitter.
	V_{SWING}	Differential input voltage
	V_X	Input differential cross point voltage
	V_{OX}	Output differential cross point voltage
W	W	HIGH-SPEED I/O BLOCK: Clock boost factor
X	—	—
Y	—	—
Z	—	—

Document Revision History

Table 1–40 shows the revision history for this chapter.

Table 1–40. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
June 2009, v1.0	Initial release.	—

About this Handbook

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

How to Contact Altera

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

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

Note:

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

Typographic Conventions

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

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

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