Virtex-5 FPGA RocketIO GTX Transceiver

User Guide

UG198 (v2.1) November 17, 2008

XILINX®

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
03/31/08	1.0	Initial Xilinx release.
05/08/08	1.1	• Table 5-3: Corrected PCI Express Rev 2 parameter values. • Table 7-14: Corrected OOB Nominal Threshold Voltage sub-table for OOBDETECT_THRESHOLD_0/1 • Table 7-35 and Table 7-37: Corrected RXBUFSTATUS0/1 description. • Table E-1 and Table E-2: Replaced with new tables.

Date	Version	Revision
09/04/08	1.2	• Added 3G-SDI to Table 1-1 and Table 5-3. • Added (Pad) to all pins in Table 1-2. • Revised DIV = 5 and DIV = 4 bulleted conditions under Equation 5-1. • Revised descriptions of DFEEYEDACMONITOR0/1, DFETAP20/1, DFETAP2MONITOR0/1, DFETAP30/1, DFETAP3MONITOR0/1, DFETAP40/1, and DFETAP4MONITOR0/1 in Table 7-5. • Revised descriptions of DFE_CAL_TIME, DFE_CFG_0/1, and RX_EN_IDLE_HOLD_DFE_0/1 in Table 7-6. • Added "Channel BER Optimization Approach", "Use Mode – Fixed Tap Mode", "Example RX Linear Equalizer and DFE Settings for Chip-to-Chip Applications", and "Example RX Linear Equalizer and DFE Settings for Backplane Applications" to "Decision Feedback Equalization" in Chapter 7. • Corrected the `101` and `110` encodings for RXSTATUS0/1 in Table 7-13. • Revised the second sentence in the Overview section of "Configurable Channel Bonding (Lane Deskew)" in Chapter 7. • Added note about channel-bonded GTX transceivers being in the same column on page 222 and in Figure 7-34. • Added nominal rating to RREF in footnote 3 of Table 10-2 and in the note following the table. • Added Boundary-Scan footnote to page 267. • Revised "Special Conditions: Unused GTX_DUAL Column" in Chapter 10. • Added notes about BGA adjacency guidelines to "SelectIO to GTX Crosstalk Guidelines" in Chapter 10.
09/23/08	2.0	• Added TXT device throughout document: ♦ Inserted TXT device descriptions to paragraphs on page 23. ♦ Inserted Figure 1-2, page 25, showing the GTX_DUAL tiles in the XC5VTX150T. ♦ Added TXT analog pins to Table 1-2, page 27 and Table 10-1, page 249. ♦ Added TXT device to footnote 2 in Table 1-5, page 38. ♦ Described TXT columns in "Description" in Chapter 4 on page 60. ♦ Added TXT packages to "Package Placement Information" in Chapter 4 on page 62. ♦ Inserted Table 4-2, page 63, showing GTX_DUAL analog pin placement. ♦ Inserted TXT GTX placement diagrams (Figure 4-8, page 70 through Figure 4-19, page 81). ♦ Clarified use cases in "Channel Bonding Mode," page 220. ♦ Added TXT device to footnote 3 in Table 10-2, page 250. ♦ Inserted descriptions of external precision resistor requirements and resistor calibration circuits for TXT device in "Description" in Chapter 10 on page 251. ♦ Inserted TXT device to Table A-1, page 309, Table A-2, page 310, Table A-3, page 312, Table A-4, page 313, Table A-6, page 314, Table A-7, page 314, Table A-11, page 317, Table A-12, page 317, and Table A-13, page 318. • Moved note about BGA adjacency guidelines above Table 10-8, page 270.

Date	Version	Revision
11/17/08	2.1	• Added SIM_MODE attribute to Table 1-5, page 38 and Table 3-1, page 52. • Added "SIM_MODE," page 53. • Corrected MGTAVTTRXC_L/R pins for the TXT 1759 package in Table 4-2, page 63. • Revised "power control" to "power down" throughout "Generic GTX Power-Down Capabilities," page 109. • Removed Relative Power Savings and Recovery Time columns from Table 5-12, page 109. • In Table 6-4, page 126, corrected encoding of TXKERR[3:0] and TXRUNDISP[3:0] and defined them based on the interface width. • Added notes regarding GTX_DUAL tile connections for FXT and TXT devices above Figure 10-2, page 252. • Revised Figure 10-2, page 252. • Added Figure 10-3, page 253 and Figure 10-4, page 253. • Revised caption in Figure 10-7, page 256 to include reference to a column. • Added bulleted notes regarding calibration and powering for FXT and TXT devices on page 257. • Rewrote "Partially used GTX_DUAL column," page 268. • In Table 10-6, page 268, revised table note 1 and added table note 2. • Added bullet regarding BGA adjacency guideline for TXT devices on page 269.

Table of Contents

Chapter 4: Implementation

Chapter 5: Tile Features

Chapter 6: GTX Transmitter (TX)

Chapter 7: GTX Receiver (RX)

Chapter 8: Cyclic Redundancy Check

Chapter 9: Loopback

Chapter 10: GTX-to-Board Interface

Section 2: Board Level Design

Chapter 11: Design Constraints Overview

Chapter 12: PCB Materials and Traces

Chapter 13: Design of Transitions

Chapter 14: Guidelines and Examples

Section 3: Appendices

Appendix A: MGT to GTX Transceiver Design Migration

About This Guide

This document shows how to use the RocketIO™ GTX transceivers in Virtex®-5 FPGAs. Complete and up-to-date documentation of the Virtex-5 family of FPGAs is available on the Xilinx website at http://www.xilinx.com/virtex5.

Guide Contents

This manual contains the following chapters and appendices:

Additional Documentation

The following documents are also available for download at http://www.xilinx.com/virtex5.

- Virtex-5 Family Overview

  The features and product selection of the Virtex-5 family are outlined in this overview.

- Virtex-5 FPGA Data Sheet: DC and Switching Characteristics

  This data sheet contains the DC and Switching Characteristic specifications for the Virtex-5 family.

- Virtex-5 FPGA User Guide

  This user guide includes chapters on:

  - Clocking Resources
  - Clock Management Technology (CMT)
  - Phase-Locked Loops (PLLs)
  - Block RAM
  - Configurable Logic Blocks (CLBs)
  - SelectIO™ Resources
  - SelectIO Logic Resources
  - Advanced SelectIO Logic Resources

- Virtex-5 FPGA RocketIO GTP Transceiver User Guide

  This guide describes the RocketIO™ GTP transceivers available in the Virtex-5 LXT and SXT platforms.

- Virtex-5 FPGA Embedded Processor Block for PowerPC® 440 Designs

  This reference guide is a description of the embedded processor block available in the Virtex-5 FXT platform.

- Virtex-5 FPGA Tri-Mode Ethernet Media Access Controller

  This guide describes the dedicated Tri-Mode Ethernet Media Access Controller available in the Virtex-5 LXT, SXT, FXT, and TXT platforms.

- Virtex-5 FPGA Integrated Endpoint Block User Guide for PCI Express Designs

  This guide describes the integrated Endpoint blocks in the Virtex-5 LXT, SXT, FXT, and TXT platforms used for PCI Express® designs.

- XtremeDSP Design Considerations

  This guide describes the XtremeDSP™ slice and includes reference designs for using the DSP48E slice.

- Virtex-5 FPGA Configuration Guide

  This all-encompassing configuration guide includes chapters on configuration interfaces (serial and SelectMAP), bitstream encryption, Boundary-Scan and JTAG configuration, reconfiguration techniques, and readback through the SelectMAP and JTAG interfaces.

- Virtex-5 FPGA System Monitor User Guide

  The System Monitor functionality available in all the Virtex-5 devices is outlined in this guide.

- Virtex-5 FPGA Packaging and Pinout Specifications

  This specification includes the tables for device/package combinations and maximum I/Os, pin definitions, pinout tables, pinout diagrams, mechanical drawings, and thermal specifications.

- Virtex-5 FPGA PCB Designer's Guide

  This guide provides information on PCB design for Virtex-5 devices, with a focus on strategies for making design decisions at the PCB and interface level.

The following documents provide supplemental material useful to this user guide:

1. Athavale, Abhijit and Carl Christensen. *High-Speed Serial I/O Made Simple*. http://www.xilinx.com/publications/books/serialio/serialio-book.pdf
2. UG200, *Embedded Processor Block in Virtex-5 FPGAs Reference Guide*.
3. *Synthesis and Simulation Design Guide* http://www.xilinx.com/support/software_manuals.htm
4. Granberg, Tom. *Handbook of Digital Techniques for High-Speed Design*. Prentice-Hall. ISBN-10: 0-13-142291-X. ISBN-13: 978-0131422919.
5. Grover, Frederick W., Ph.D. 1946. *Inductance Calculations: Working Formulas and Tables*. New York: D. Van Nostrand Company, Inc.
6. Johnson, Howard. *Signal Integrity Techniques and Loss Budgeting for RocketIO Transceivers*. http://www.xilinx.com/onlinestore/si_intro.htm
7. Johnson, Howard, Martin Graham. *High-Speed Signal Propagation: Advanced Black Magic*. Prentice-Hall. ISBN-10: 0-13-084408-X. ISBN-13: 978-0130844088.
8. Montrose, Mark I. 1999. *EMC and the Printed Circuit Board*. The Institute of Electrical and Electronics Engineers, Inc. ISBN 0-7803-4703-X.
9. Smith, Larry D. November 1984. *Decoupling Capacitor Calculations for CMOS Circuits*. Proceedings EPEP Conference.
10. Williams, Ross N. *The Painless Guide to CRC Error Detection Algorithms*. http://www.ross.net/crc/ (CRC pitstop).
11. Schlichthaerle, Dietrich. *Digital Filters: Basics and Design*. Springer. ISBN-10 3-540-66841-1.
12. DS083, *Virtex-II Pro and Virtex-II Pro X Platform FPGAs Complete Data Sheet*.
13. UG024, *RocketIO Transceiver User Guide*.
14. UG076, *Virtex-4 RocketIO Multi-Gigabit Transceiver User Guide*.
15. XAPP209, *IEEE 802.3 Cyclic Redundancy Check*.
16. XAPP562, *Configurable LocalLink CRC Reference Design*.
17. UG196, *Virtex-5 FPGA RocketIO GTP Transceiver User Guide*.
18. UG204, *Virtex-5 FPGA RocketIO GTX Transceiver Wizard Getting Started Guide*.

Additional Support Resources

To search the database of silicon and software questions and answers, or to create a technical support case in WebCase, see the Xilinx website at: http://www.xilinx.com/support.

Typographical Conventions

This document uses the following typographical conventions. An example illustrates each convention.

Convention	Meaning or Use	Example
Italic font	References to other documents	See the *Virtex-5 Configuration Guide* for more information.
	Emphasis in text	The address (F) is asserted *after* clock event 2.
Underlined Text	Indicates a link to a web page.	http://www.xilinx.com/virtex5

Online Document

The following conventions are used in this document:

Convention	Meaning or Use	Example
Blue text	Cross-reference link to a location in the current document	See the section "Additional Documentation" for details. Refer to "Clock Management Technology (CMT)" in Chapter 2 for details.
Red text	Cross-reference link to a location in another document	See Figure 6 in the *Virtex-5 FPGA Data Sheet*
Blue, underlined text	Hyperlink to a website (URL)	Go to http://www.xilinx.com for the latest documentation.

Section 1: FPGA Level Design

This section provides the information needed to incorporate RocketIO™ GTX transceivers into an FPGA design, including:

- The features and characteristics of the GTX transceivers
- How to use the RocketIO GTX Wizard to configure the transceivers
- Mapping of transceiver instances to device resources
- Simulation of GTX transceiver designs
- Board-level clocking and power requirements

This section includes the following chapters:

XILINX®

Chapter 1

Introduction to the RocketIO GTX Transceiver

Overview

The RocketIO™ GTX transceiver is a power-efficient transceiver for Virtex®-5 FPGAs. The GTX transceiver is highly configurable and tightly integrated with the programmable logic resources of the FPGA. It provides the following features to support a wide variety of applications:

- Current Mode Logic (CML) serial drivers/buffers with configurable termination, voltage swing, and coupling.
- Programmable TX pre-emphasis, RX equalization, and linear and decision feedback equalization (DFE) for optimized signal integrity.
- Line rates from 750 Mb/s to 6.5 Gb/s, with optional 5x digital oversampling required for rates between 150 Mb/s and 750 Mb/s. The nominal operation range of the shared PMA PLL is from 1.5 GHz to 3.25 GHz. These are nominal values, see DS202: *Virtex-5 FPGA Data Sheet* for specifications.
- Optional built-in PCS features, such as 8B/10B encoding, comma alignment, channel bonding, and clock correction.
- Fixed latency modes for minimized, deterministic datapath latency.
- Beacon signaling for PCI Express® designs and Out-of-Band signaling including COM signal support for SATA designs.
- RX/TX Gearbox provides header insertion and extraction support for 64B/66B and 64B/67B (Interlaken) protocols.
- Receiver eye scan:
  - Vertical eye scan in the voltage domain for testing purposes
  - Horizontal eye scan in the time domain for testing purposes

The first-time user is recommended to read *High-Speed Serial I/O Made Simple* [Ref 1], which discusses high-speed serial transceiver technology and its applications.

Table 1-1 lists some of the standard protocols designers can implement using the GTX transceiver. The Xilinx CORE Generator™ tool includes a Wizard to automatically configure GTX transceivers to support one of these protocols or perform custom configuration (see Chapter 2, "RocketIO GTX Transceiver Wizard").

The GTX_DUAL tile offers a data rate range and features that allow physical layer support for various protocols as illustrated in Table 1-1.

Table 1-1: **List of Standards Supported by the GTX_DUAL Tile**

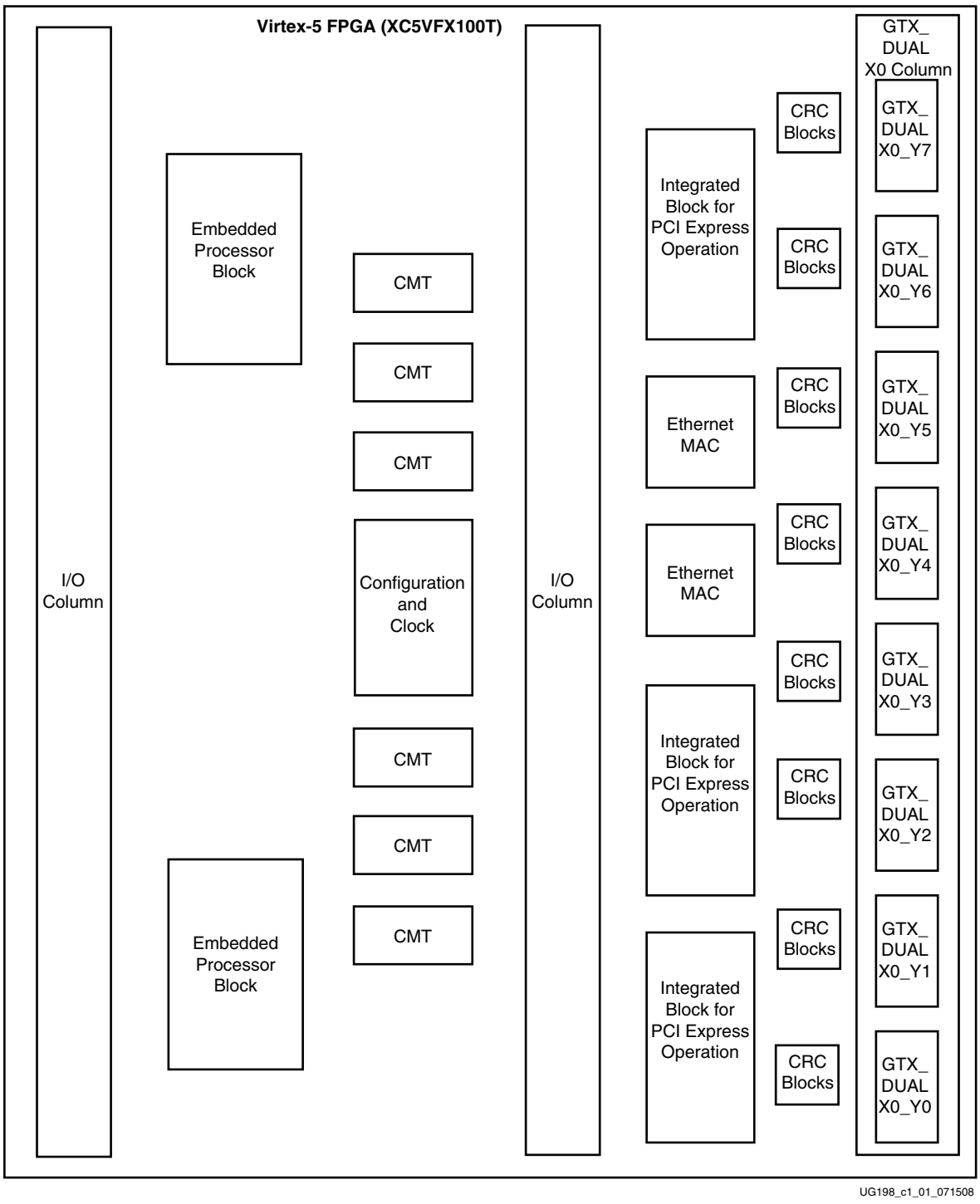
Compatible Protocols	Data Rates	Miscellaneous Features
PCI Express, Rev. 1.0a PCI Express, Rev. 1.1	2.5 Gb/s	• TX receive detect • Loss of Signal (LOS)/Idle state detect • Low power states • Beacon signaling • Ground referenced termination
PCI Express, Rev. 2.0	5.0 Gb/s	
Interlaken	3.125 Gb/s, 6.25 Gb/s	Header insertion/extraction for 64B/67B
XAUI 802.3ae D5p0	3.125 Gb/s	LOS
OIF-CEI6G	6.25 Gb/s	• No Equalizer (EQ), low power mode • 4-tap adaptive DFE
OC-12/48	622.08/2488.32 Mb/s	• Allows FIFO bypassing for synchronous operation • 5x digital oversampling
FC-1, Revision 4.0	1.0625 Gb/s	Rate negotiation (allows operating the TX and RX at different speeds)
FC-2, Revision 4.0	2.125 Gb/s	
FC-4, Revision 4.0	4.25 Gb/s	
10GFC	3.1875 Gb/s	
SDI HD-SDI DVB-ASI 3G-SDI	176/270/360 Mb/s 1.485/1.4835 Gb/s 270 Mb/s 2.970 Gb/s	5x digital oversampling
10G Base-CX4 802.3ak/D4.0	3.125 Gb/s	
Gigabit Ethernet (1000BASE-CX 802.3z/D5.0)	1.25 Gb/s	
SATA Generation 1/2, Rev. 1.0a SATA Generation 2, Rev. 1.0a SATA Generation 3, Rev. 1.0a	1.5 Gb/s 3.0 Gb/s 6.0 Gb/s	• Rate negotiation for Generation 2 (entire link operates at Generation 1/Generation 2 speeds) • LOS • OOB beacon
Serial RapidIO	1.25/2.5/3.125/6.25 Gb/s	
CPRI, Version 2.0	614.4/1228.8/2457.6 Mb/s	5x digital oversampling
Infiniband, Volume 2, Release 1.1	2.5 Gb/s	
SFI-5	2.488 – 3.125 Gb/s	Synchronous clocking (bypass FIFOs)
OBSAI RP3, Spec. Issue 1.0	768/1536/3072 Mb/s	
Aurora	150 Mb/s – 6.5 Gb/s	

GTX transceivers are placed as dual transceiver GTX_DUAL tiles in Virtex-5 FXT and TXT platform devices. This configuration allows two transceivers to share a single PLL with the TX and RX functions of both, reducing size and power consumption.

Figure 1-1 shows GTX_DUAL tile placement in an example Virtex-5 device (XC5VFX100T). All GTX_DUAL tiles form a single GTX_DUAL column on the right side of an FXT device as shown in Figure 1-1. TXT devices have one GTX_DUAL column on the left side of the device and one column on the right side of the device as shown in Figure 1-2. The embedded processor blocks in Virtex-5 FXT devices contain a PowerPC® 440 processor along with several additional modules.

Additional information on the functional blocks in Figure 1-1 is available in the following locations:

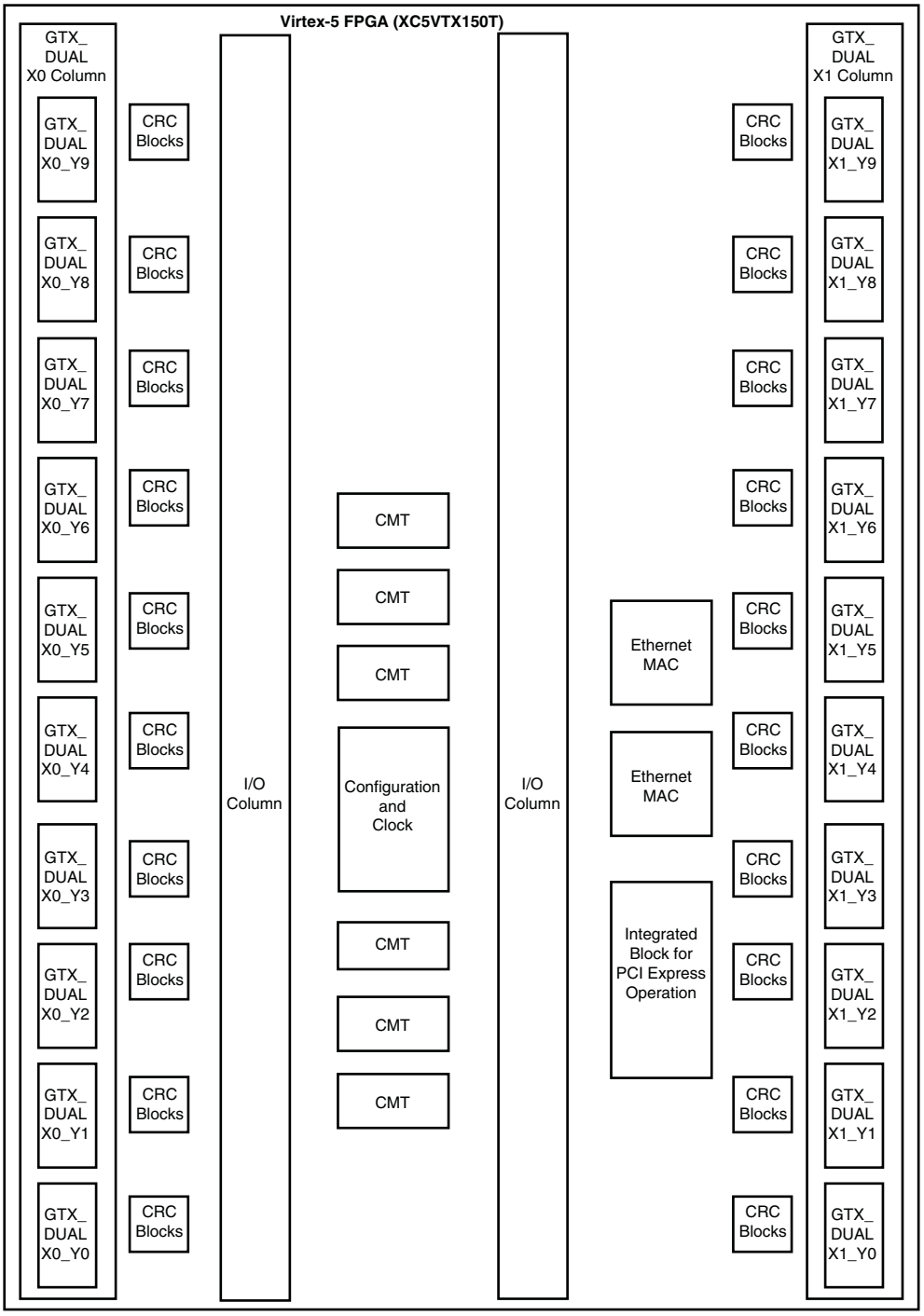
- The *Embedded Processor Block for Virtex-5 FPGAs Reference Guide* [Ref 2] provides more information on the embedded processor block for Virtex-5 FPGAs.
- Chapter 8, "Cyclic Redundancy Check," provides more details on the CRC blocks in Figure 1-1.
- The *Virtex-5 FPGA Configuration Guide* provides more information on the Configuration and Clock, CMT, and I/O blocks.
- The *Virtex-5 FPGA Embedded Tri-Mode Ethernet MAC User Guide* provides detailed information on the Ethernet MAC.
- The *Virtex-5 FPGA Integrated Endpoint Block User Guide for PCI Express Designs* provides detailed information on PCI Express compliance.



Notes:

1. This figure does *not* illustrate exact size, location, or scale of the functional blocks to each other. It does show the correct number of available resources.
2. To improve clarity, this figure does *not* show the CLB, DSP, and block RAM columns.

Figure 1-1: **GTX_DUAL Tile Inside the Virtex-5 XC5VFX100T FPGA**

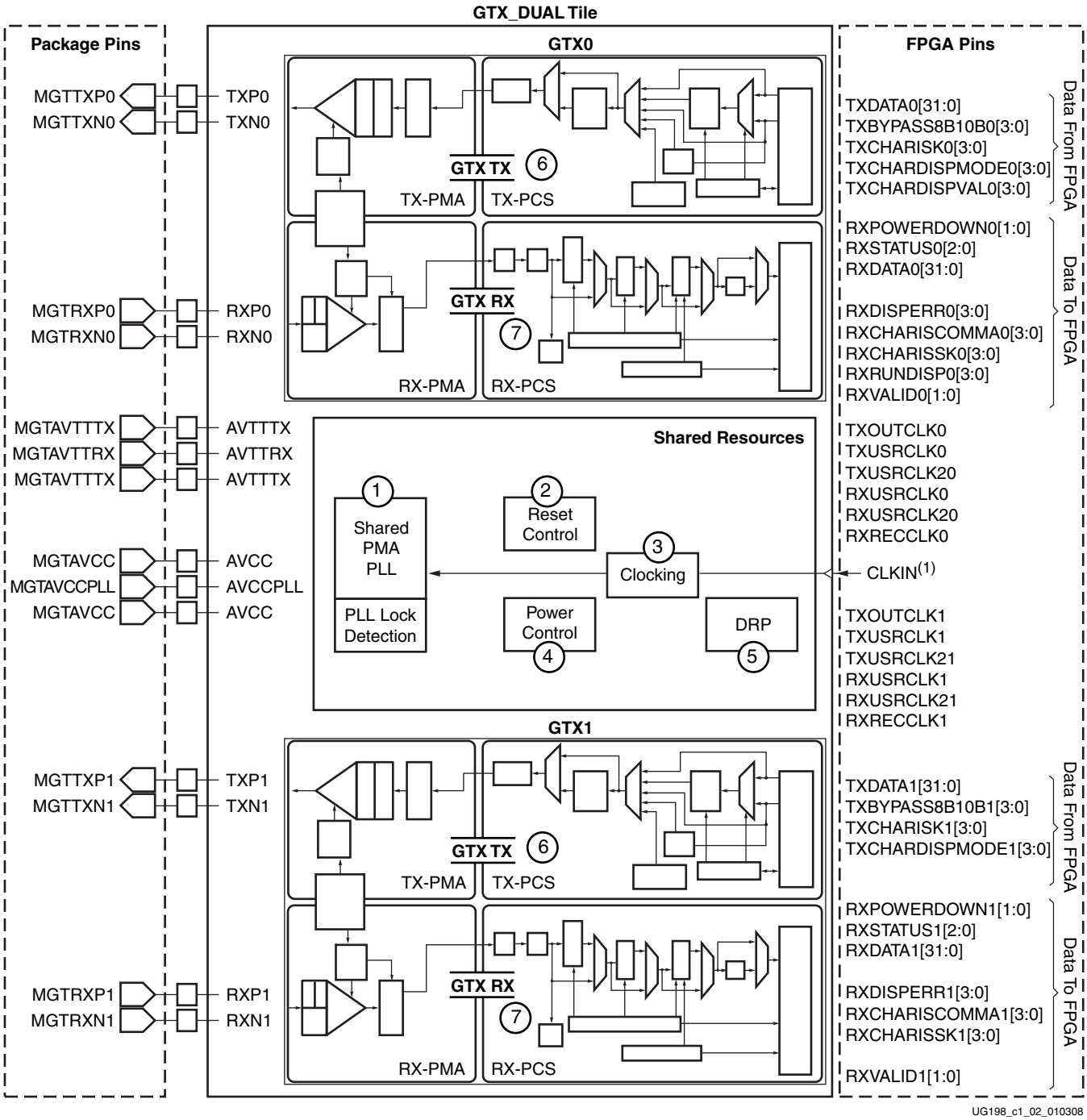


Notes:

1. This figure does *not* illustrate exact size, location, or scale of the functional blocks to each other. It does show the correct number of available resources.
2. To improve clarity, this figure does *not* show the CLB, DSP, and block RAM columns.
3. Channel bonding of multiple transceivers in different columns is an advanced feature. Contact your System I/O specialist for details.
4. GTX_DUAL tiles of column X1 must be used when connecting to the Ethernet MAC blocks or to the Integrated Block for PCI Express operation.
5. When migrating a design from an FXT device to a TXT device, the indices of the GTX_DUAL instantiations must be changed from X0 to X1.

Figure 1-2: **GTX_DUAL Tile Inside the Virtex-5 XC5VTX150T FPGA**

Figure 1-3 shows a diagram of a GTX_DUAL tile, containing two GTX transceivers and a shared resources block. The GTX_DUAL tile is the HDL primitive used to operate GTX transceivers in the FPGA.



Notes:

1. CLKIN is a simplification for a clock source. See Figure 5-3, page 94 for details on CLKIN.

Figure 1-3: **GTX_DUAL Tile Block Diagram**

The procedures for configuring and using each of the seven major blocks in the GTX_DUAL tile shown in Figure 1-3 are discussed in detail in the following sections:

1. "Shared PMA PLL," page 84 (Chapter 5)
2. "Reset," page 98 (Chapter 5)
3. "Clocking," page 93 (Chapter 5)
4. "Power Control," page 107 (Chapter 5)
5. "Dynamic Reconfiguration Port," page 113 (Chapter 5)
6. "GTX Transmitter (TX)," page 115 (Chapter 6)
7. "GTX Receiver (RX)," page 157 (Chapter 7)

Ports and Attributes

This section contains alphabetical tables of pins (Table 1-2), ports (Table 1-3 and Table 1-4), and attributes (Table 1-5 and Table 1-6). In all Port and Attribute tables in this guide, names that end with 0 are for the GTX0 transceiver on the tile, and names that end with 1 are for the GTX1 transceiver. Names that do not end with 0 or 1 are shared.

Table 1-2 lists alphabetically the signal names, directions, and descriptions of the GTX_DUAL analog pins, and provides links to their detailed descriptions.

Table 1-2: **GTX_DUAL Analog Pin Summary**

Pin	Dir	Description	Section (Page)
MGTAVCC	In (Pad)	Analog supply for the internal analog circuits of the GTX_DUAL tile.	Analog Design Guidelines (page 249)
MGTAVCCPLL	In (Pad)	Analog supply for the shared PMA PLL of the GTX_DUAL tile.	Analog Design Guidelines (page 249)
MGTAVTTRX	In (Pad)	Analog supply for the receiver circuits and the termination of the GTX_DUAL tile.	Analog Design Guidelines (page 249)
MGTAVTTRXC	In (Pad)	FXT only: Analog supply for the resistor calibration and the standby circuit of the entire device.	Analog Design Guidelines (page 249)
MGTAVTTRXC_R	In (Pad)	TXT only: Analog supply for the resistor calibration and standby circuits of the X1 column.	Analog Design Guidelines (page 250)
MGTAVTTRXC_L	In (Pad)	TXT only: Analog supply for the resistor calibration and standby circuits of the X0 column.	Analog Design Guidelines (page 250)
MGTAVTTTX	In (Pad)	Analog supply for the transmitter termination and driver circuits and the clock routing and muxing network of the GTX_DUAL tile.	Analog Design Guidelines (page 250)
MGTREFCLKP MGTREFCLKN	In (Pad)	Differential clock input pin pair for the reference clock of the GTX_DUAL tile.	Analog Design Guidelines (page 250)
MGTRREF	In (Pad)	FXT only: Reference resistor input for the entire device.	Analog Design Guidelines (page 250)

Table 1-2: **GTX_DUAL Analog Pin Summary *(Cont'd)***

Pin	Dir	Description	Section (Page)
MGTRREF_R	In (Pad)	TXT only: Reference resistor input for the X1 column.	Analog Design Guidelines (page 250)
MGTRREF_L	In (Pad)	TXT only: Reference resistor input for the X0 column.	Analog Design Guidelines (page 250)
MGTRXN0 MGTRXP0 MGTRXN1 MGTRXP1	In (Pad)	Differential complements forming a differential receiver input pair for each transceiver.	RX Termination and Equalization (page 158)
MGTTXN0 MGTTXP0 MGTTXN1 MGTTXP1	Out (Pad)	Differential complements forming a differential transmitter output pair for each transceiver.	RX Termination and Equalization (page 158)

Table 1-3 lists alphabetically the signal names, clock domains, directions, and descriptions for the GTX_DUAL ports, and provides links to their detailed descriptions.

Table 1-3: **GTX_DUAL Port Summary**

Port	Dir	Domain	Description	Section (Page)
CLKIN	In	Async	Reference clock input to the shared PMA PLL.	Shared PMA PLL (page 85), Clocking (page 95), Power Control (page 107)
DADDR[6:0]	In	DCLK	DRP address bus.	Dynamic Reconfiguration Port (page 113)
DCLK	In	N/A	DRP interface clock.	Dynamic Reconfiguration Port (page 113)
DEN	In	DCLK	Enables DRP read or write operations.	Dynamic Reconfiguration Port (page 113)
DFECLKDLYADJ0[5:0] DFECLKDLYADJ1[5:0]	In	RXUSRCLK2	DFE clock delay adjust control for each transceiver.	Decision Feedback Equalization (page 163)
DFECLKDLYADJMONITOR0[5:0] DFECLKDLYADJMONITOR1[5:0]	Out	RXUSRCLK2	DFE clock delay adjust monitor for each transceiver.	Decision Feedback Equalization (page 163)
DFEEYEDACMONITOR0[4:0] DFEEYEDACMONITOR1[4:0]	Out	RXUSRCLK2	Vertical Eye Scan for each transceiver (voltage domain).	Decision Feedback Equalization (page 163)
DFESENSCAL0[2:0] DFESENSCAL1[2:0]	Out	RXUSRCLK2	DFE calibration status.	Decision Feedback Equalization (page 163)
DFETAP10[4:0] DFETAP11[4:0]	In	RXUSRCLK2	DFE tap 1 weight value control for each transceiver (5-bit resolution).	Decision Feedback Equalization (page 163)
DFETAP1MONITOR0[4:0] DFETAP1MONITOR1[4:0]	Out	RXUSRCLK2	DFE tap 1 weight value monitor for each transceiver (5-bit resolution).	Decision Feedback Equalization (page 163)

Table 1-3: **GTX_DUAL Port Summary *(Cont'd)***

Port	Dir	Domain	Description	Section (Page)
DFETAP20[4:0] DFETAP21[4:0]	In	RXUSRCLK2	DFE tap 2 weight value control for each transceiver (4-bit resolution plus 1-bit sign).	Decision Feedback Equalization (page 163)
DFETAP2MONITOR0[4:0] DFETAP2MONITOR1[4:0]	Out	RXUSRCLK2	DFE tap 2 weight value monitor for each transceiver (4-bit resolution plus 1-bit sign).	Decision Feedback Equalization (page 163)
DFETAP30[3:0] DFETAP31[3:0]	In	RXUSRCLK2	DFE tap 3 weight value control for each transceiver (3-bit resolution plus 1-bit sign).	Decision Feedback Equalization (page 163)
DFETAP3MONITOR0[3:0] DFETAP3MONITOR1[3:0]	Out	RXUSRCLK2	DFE tap 3 weight value monitor for each transceiver (3-bit resolution plus 1-bit sign).	Decision Feedback Equalization (page 164)
DFETAP40[3:0] DFETAP41[3:0]	In	RXUSRCLK2	DFE tap 4 weight value control for each transceiver (3-bit resolution plus 1-bit sign).	Decision Feedback Equalization (page 164)
DFETAP4MONITOR0[3:0] DFETAP4MONITOR1[3:0]	Out	RXUSRCLK2	DFE tap 4 weight value monitor for each transceiver (3-bit resolution plus 1-bit sign).	Decision Feedback Equalization (page 164)
DI[15:0]	In	DCLK	Data bus for writing configuration data from the FPGA logic to the GTX_DUAL tile.	Dynamic Reconfiguration Port (page 113)
DO[15:0]	Out	DCLK	Data bus for reading configuration data from the GTX_DUAL tile to the FPGA logic.	Dynamic Reconfiguration Port (page 113)
DRDY	Out	DCLK	Indicates the operation is complete for DRP write operations and data is valid for DRP read operations.	Dynamic Reconfiguration Port (page 113)
DWE	In	DCLK	Indicates whether the DRP operation is a read or a write.	Dynamic Reconfiguration Port (page 113)
GTXRESET	In	Async	Starts the full GTX_DUAL reset sequence.	Reset (page 99)
GTXTEST[13:0]	In	Async	Factory test pins. Do not change default value.	

Table 1-3: **GTX_DUAL Port Summary *(Cont'd)***

Port	Dir	Domain	Description	Section (Page)
INTDATAWIDTH	In	Async	Sets the internal datapath width for the GTX_DUAL tile. 0: 16-bit internal datapath width 1: 20-bit internal datapath width	Shared PMA PLL (page 85), FPGA TX Interface (page 116), Parallel In to Serial Out (page 146), Serial In to Parallel Out (page 181), PRBS Detection (page 188), Configurable RX Elastic Buffer and Phase Alignment (page 203), Configurable Clock Correction (page 211), Configurable Channel Bonding (Lane Deskew) (page 217), FPGA RX Interface (page 231)
LOOPBACK0[2:0] LOOPBACK1[2:0]	In	Async	Sets the loopback mode.	Loopback (page 244)
PHYSTATUS0 PHYSTATUS1	Out	Async	Indicates completion of several PHY functions, including power management state transitions and receiver detection.	Receive Detect Support for PCI Express Operation (page 151)
PLLLKDET	Out	Async	Indicates that the VCO rate is within acceptable tolerances of the desired rate.	Shared PMA PLL (page 85)
PLLLKDETEN	In	Async	Enables the PLL lock detector.	Shared PMA PLL (page 85)
PLLPOWERDOWN	In	Async	Powers down the shared PMA PLL.	Power Control (page 107)
PRBSCNTRESET0 PRBSCNTRESET1	In	RXUSRCLK2	Resets the PRBS error counter.	PRBS Detection (page 188)
REFCLKOUT	Out	N/A	Provides access to the reference clock provided to the shared PMA PLL (CLKIN).	Shared PMA PLL (page 85), Clocking (page 95), FPGA TX Interface (page 116), TX Buffering, Phase Alignment, and TX Skew Reduction (page 140), FPGA RX Interface (page 231)
REFCLKPWRDNB	In	Async	Powers down the GTX reference clock circuit (active Low).	Power Control (page 107)
RESETDONE0 RESETDONE1	Out	Async	Indicates when the GTX transceiver has finished reset and is ready for use.	Reset (page 99), RX Clock Data Recovery (page 177)

Table 1-3: **GTX_DUAL Port Summary *(Cont'd)***

Port	Dir	Domain	Description	Section (Page)
RXBUFRESET0 RXBUFRESET1	In	Async	Resets the RX elastic buffer logic.	Reset (page 99), Configurable RX Elastic Buffer and Phase Alignment (page 203), Configurable Clock Correction (page 211)
RXBUFSTATUS0[2:0] RXBUFSTATUS1[2:0]	Out	RXUSRCLK2	Indicates the overflow/underflow status of the RX elastic buffer.	Configurable RX Elastic Buffer and Phase Alignment (page 203), Configurable Clock Correction (page 211)
RXBYTEISALIGNED0 RXBYTEISALIGNED1	Out	RXUSRCLK2	Indicates if the parallel data stream is properly aligned on byte boundaries according to comma detection. When PCOMMA_ALIGN = TRUE, asserted for alignment to PCOMMA value. When MCOMMA_ALIGN = TRUE, asserted for alignment to MCOMMA value.	Configurable Comma Alignment and Detection (page 190)
RXBYTEREALIGN0 RXBYTEREALIGN1	Out	RXUSRCLK2	Indicates if the byte alignment within the serial data stream has changed due to a comma detection.	Configurable Comma Alignment and Detection (page 190)
RXCDRRESET0 RXCDRRESET1	In	RXUSRCLK2	Reset for the RX CDR. Also resets the rest of the RX PCS.	Reset (page 99), RX Clock Data Recovery (page 177)
RXCHANBONDSEQ0 RXCHANBONDSEQ1	Out	RXUSRCLK2	Indicates when RXDATA contains the start of a channel bonding sequence.	Configurable Channel Bonding (Lane Deskew) (page 217)
RXCHANISALIGNED0 RXCHANISALIGNED1	Out	RXUSRCLK2	Indicates if the channel is properly aligned with the master transceiver according to observed channel bonding sequences in the data stream.	Configurable Channel Bonding (Lane Deskew) (page 217)
RXCHANREALIGN0 RXCHANREALIGN1	Out	RXUSRCLK2	Held High for at least one cycle when the receiver has changed.	Configurable Channel Bonding (Lane Deskew) (page 217)

Table 1-3: **GTX_DUAL Port Summary *(Cont'd)***

Port	Dir	Domain	Description	Section (Page)
RXCHARISCOMMA0[3:0] RXCHARISCOMMA1[3:0]	Out	RXUSRCLK2	Asserted when RXDATA is an 8B/10B comma. RXCHARISCOMMA is influenced by the setting of these attributes: DEC_MCOMMA_DETECT_0 DEC_MCOMMA_DETECT_1 DEC_PCOMMA_DETECT_0 DEC_PCOMMA_DETECT_1	Configurable 8B/10B Decoder (page 198)
RXCHARISK0[3:0] RXCHARISK1[3:0]	Out	RXUSRCLK2	Asserted when RXDATA is an 8B/10B K character.	Configurable 8B/10B Decoder (page 198)
RXCHBONDI0[3:0] RXCHBONDI1[3:0]	In	RXUSRCLK	FPGA channel bonding control. Used only by slaves.	Configurable Channel Bonding (Lane Deskew) (page 217)
RXCHBONDO0[3:0] RXCHBONDO1[3:0]	Out	RXUSRCLK	FPGA channel bonding control.	Configurable Channel Bonding (Lane Deskew) (page 217)
RXCLKCORCNT0[2:0] RXCLKCORCNT1[2:0]	Out	RXUSRCLK2	Reports the status of the elastic buffer clock correction.	Configurable Clock Correction (page 211)
RXCOMMADET0 RXCOMMADET1	Out	RXUSRCLK2	Asserted when the comma alignment block detects a comma.	Configurable Comma Alignment and Detection (page 190)
RXCOMMADETUSE0 RXCOMMADETUSE1	In	RXUSRCLK2	Activates the comma detection and alignment circuit.	Configurable Comma Alignment and Detection (page 190)
RXDATA0[31:0] RXDATA1[31:0]	Out	RXUSRCLK2	Receive data bus of the receive interface to the FPGA.	FPGA RX Interface (page 231)
RXDATAVALID0 RXDATAVALID1	Out	RXUSRCLK2	Data valid for RX Gearbox.	RX Gearbox (page 226)
RXDATAWIDTH0[1:0] RXDATAWIDTH1[1:0]	In	RXUSRCLK2	Selects the width of the RXDATA receive data connection to the FPGA.	Configurable 8B/10B Decoder (page 198), FPGA RX Interface (page 231)
RXDEC8B10BUSE0 RXDEC8B10BUSE1	In	RXUSRCLK2	Enables the 8B/10B decoder.	Configurable 8B/10B Decoder (page 198)
RXDISPERR0[3:0] RXDISPERR1[3:0]	Out	RXUSRCLK2	Indicates if RXDATA was received with a disparity error.	Configurable 8B/10B Decoder (page 198)
RXELECIDLE0 RXELECIDLE1	Out	Async	Indicates the differential voltage between RXN and RXP dropped below the minimum threshold.	RX OOB/Beacon Signaling (page 171)
RXENCHANSYNC0 RXENCHANSYNC1	In	RXUSRCLK2	Enables channel bonding.	Configurable Channel Bonding (Lane Deskew) (page 217)

Table 1-3: **GTX_DUAL Port Summary *(Cont'd)***

Port	Dir	Domain	Description	Section (Page)
RXENEQB0 RXENEQB1	In	Async	This port has no effect on transceiver performance.	–
RXENMCOMMAALIGN0 RXENMCOMMAALIGN1	In	RXUSRCLK2	Aligns the byte boundary when comma minus is detected.	Configurable Comma Alignment and Detection (page 190)
RXENPCOMMAALIGN0 RXENPCOMMAALIGN1	In	RXUSRCLK2	Aligns the byte boundary when comma plus is detected.	Configurable Comma Alignment and Detection (page 191)
RXENPMAPHASEALIGN0 RXENPMAPHASEALIGN1	In	RXUSRCLK2	Enables the alignment of the XCLK with the RXURSCLK (independently) for each transceiver in the GTX_DUAL tile. Set High when bypassing the RX elastic buffer.	Configurable RX Elastic Buffer and Phase Alignment (page 204)
RXENPRBSTST0[1:0] RXENPRBSTST1[1:0]	In	RXUSRCLK2	Receiver test pattern checker control.	PRBS Detection (page 188)
RXENSAMPLEALIGN0 RXENSAMPLEALIGN1	In	RXUSRCLK2	When High, the 5x oversampler in the PCS continually adjusts its sample point. When Low, it samples only at the point that was active before the port went Low.	Oversampling (page 184)
RXEQMIX0[1:0] RXEQMIX1[1:0]	In	Async	Sets the wideband/high-pass mix ratio for the RX equalizer.	RX Termination and Equalization (page 158)
RXGEARBOXSLIP0 RXGEARBOXSLIP1	In	RXUSRCLK2	Slips the RX Gearbox position by one cycle.	RX Gearbox (page 226)
RXHEADER0[2:0] RXHEADER1[2:0]	Out	RXUSRCLK2	RX header bits from RX Gearbox.	RX Gearbox (page 226)
RXHEADERVALID0 RXHEADERVALID1	Out	RXUSRCLK2	Indicates when RX header bits from the RX Gearbox are valid.	RX Gearbox (page 226)
RXLOSSOFSYNC0[1:0] RXLOSSOFSYNC1[1:0]	Out	RXUSRCLK2	FPGA status related to byte stream synchronization, depending on the state of the RX_LOSS_OF_SYNC_FSM attribute.	Configurable Loss-of-Sync State Machine (page 195)
RXNOTINTABLE0[3:0] RXNOTINTABLE1[3:0]	Out	RXUSRCLK2	Indicates if RXDATA is the result of an 8B/10B code that is in error.	Configurable 8B/10B Decoder (page 199)
RXOVERSAMPLEERR0 RXOVERSAMPLEERR1	Out	RXUSRCLK2	Indicates the FIFO in oversampling circuit has either overflowed or underflowed.	Oversampling (page 184)

Table 1-3: **GTX_DUAL Port Summary *(Cont'd)***

Port	Dir	Domain	Description	Section (Page)
RXPMASETPHASE0 RXPMASETPHASE1	In	RXUSRCLK2	Aligns the PMA receiver recovered clock with the PCS user clocks, allowing the RX elastic buffer to be bypassed.	Configurable RX Elastic Buffer and Phase Alignment (page 204)
RXPOLARITY0 RXPOLARITY1	In	RXUSRCLK2	Inverts the polarity of incoming data.	RX Polarity Control (page 187)
RXPOWERDOWN0[1:0] RXPOWERDOWN1[1:0]	In	Async	Powers down the RX lanes.	Power Control (page 107), Receive Detect Support for PCI Express Operation (page 152)
RXPRBSERR0 RXPRBSERR1	Out	RXUSRCLK2	Indicates if the number of errors in PRBS testing exceeds the value set by the PRBS_ERR_THRESHOLD attribute.	PRBS Detection (page 188)
RXRECCLK0 RXRECCLK1	Out	N/A	Recovered clocks derived from the RX Clock Data Recovery circuit. Clocks the RX logic between the PMA and the RX elastic buffer.	FPGA RX Interface (page 231)
RXRESET0 RXRESET1	In	Async	Active-High reset for the RX PCS logic.	Reset (page 99), FPGA RX Interface (page 231)
RXRUNDISP0[3:0] RXRUNDISP1[3:0]	Out	RXUSRCLK2	Shows the running disparity of the 8B/10B encoder when RXDATA is received.	Configurable 8B/10B Decoder (page 199)
RXSLIDE0 RXSLIDE1	In	RXUSRCLK2	Implements a comma alignment bump control, allowing manual comma alignment. Both the PMA and PCS provide RXSLIDE functionality depending on the value of the RX_SLIDE_MODE_0 or RX_SLIDE_MODE_1 attribute.	Configurable Comma Alignment and Detection (page 191)
RXSTARTOFSEQ0 RXSTARTOFSEQ1	Out	RXUSRCLK2	Indicates when the internal sequence counter of the RX Gearbox is 0.	RX Gearbox (page 226)
RXSTATUS0[2:0] RXSTATUS1[2:0]	Out	RXUSRCLK2	Shows the status of PCI Express or SATA operations. The decoding depends on the setting of RX_STATUS_FMT.	TX Out-of-Band/Beacon Signaling (page 154), RX OOB/Beacon Signaling (page 171), Receive Detect Support for PCI Express Operation (page 151)
RXUSRCLK0 RXUSRCLK1	In	N/A	Input clock used for internal RX logic after the RX elastic buffer.	FPGA RX Interface (page 232)

Table 1-3: **GTX_DUAL Port Summary *(Cont'd)***

Port	Dir	Domain	Description	Section (Page)
RXUSRCLK20 RXUSRCLK21	In	N/A	Input clock used for the interface between the FPGA and the GTX transceiver.	FPGA RX Interface (page 232)
RXVALID0 RXVALID1	Out	RXUSRCLK2	Indicates symbol lock and valid data on RXDATA and RXCHARISK[3:0] for PCI Express operations.	RX OOB/Beacon Signaling (page 171)
TXBUFDIFFCTRL0[2:0] TXBUFDIFFCTRL1[2:0]	In	Async	Controls the strength of the TX pre-drivers.	Configurable TX Driver (page 148)
TXBUFSTATUS0[1:0] TXBUFSTATUS1[1:0]	Out	TXUSRCLK2	TX buffer status. Indicates TX buffer overflow or underflow.	TX Buffering, Phase Alignment, and TX Skew Reduction (page 140)
TXBYPASS8B10B0[3:0] TXBYPASS8B10B1[3:0]	In	TXUSRCLK2	Controls the operation of the TX 8B/10B encoder on a per-byte basis.	Configurable 8B/10B Encoder (page 126)
TXCHARDISPMODE0[3:0] TXCHARDISPMODE1[3:0]	In	TXUSRCLK2	TXCHARDISPMODE and TXCHARDISPVAL allow the 8B/10B disparity of outgoing data to be controlled when 8B/10B encoding is enabled. When 8B/10B encoding is disabled, TXCHARDISPMODE is used to extend the data bus for TX interfaces whose width is a multiple of 10.	FPGA TX Interface (page 116), Configurable 8B/10B Encoder (page 126)
TXCHARDISPVAL0[3:0] TXCHARDISPVAL1[3:0]	In	TXUSRCLK2	TXCHARDISPVAL and TXCHARDISPMODE allow the disparity of outgoing data to be controlled when 8B/10B encoding is enabled. When 8B/10B encoding is disabled, TXCHARDISPVAL is used to extend the data bus for 10-bit, 20-bit, or 40-bit TX interfaces.	FPGA TX Interface (page 117), Configurable 8B/10B Encoder (page 126)
TXCHARISK0[3:0] TXCHARISK1[3:0]	In	TXUSRCLK2	Set High to send TXDATA as an 8B/10B K character.	Configurable 8B/10B Encoder (page 127)
TXCOMSTART0 TXCOMSTART1	In	TXUSRCLK2	Initiates the transmission of the COM sequence selected by TXCOMTYPE (SATA only).	TX Out-of-Band/Beacon Signaling (page 154)
TXCOMTYPE0 TXCOMTYPE1	In	TXUSRCLK2	Selects the type of COM signal to send (SATA only).	TX Out-of-Band/Beacon Signaling (page 154)
TXDATA0[31:0] TXDATA1[31:0]	In	TXUSRCLK2	Transmitting data bus.	FPGA TX Interface (page 117)
TXDATAWIDTH0[1:0] TXDATAWIDTH1[1:0]	In	TXUSRCLK2	Selects the width of the transmitting data bus port.	FPGA TX Interface (page 117)

Table 1-3: **GTX_DUAL Port Summary *(Cont'd)***

Port	Dir	Domain	Description	Section (Page)
TXDETECTRX0 TXDETECTRX1	In	TXUSRCLK2	Activates the receiver detection feature for PCI Express operations and also initiates the loopback as described in the PIPE specification.	Power Control (page 107), Receive Detect Support for PCI Express Operation (page 151)
TXDIFFCTRL0[2:0] TXDIFFCTRL1[2:0]	In	Async	Controls the transmitter differential output swing.	Configurable TX Driver (page 148)
TXELECIDLE0 TXELECIDLE1	In	TXUSRCLK2	Drives TXN and TXP to the same voltage to perform electrical idle/beaconing for PCI Express operations.	Power Control (page 107), TX Out-of-Band/Beacon Signaling (page 154)
TXENC8B10BUSE0 TXENC8B10BUSE1	In	TXUSRCLK2	Enables the 8B/10B encoder.	Configurable 8B/10B Encoder (page 127), FPGA TX Interface (page 117)
TXENPMAPHASEALIGN0 TXENPMAPHASEALIGN1	In	Async	Allows both GTX transceivers in a GTX_DUAL tile to align their XCLKs with their TXUSRCLKs and allows the XCLKs in multiple GTX transceivers to be synchronized.	TX Buffering, Phase Alignment, and TX Skew Reduction (page 140)
TXENPRBSTST0[1:0] TXENPRBSTST1[1:0]	In	TXUSRCLK2	Transmitter test pattern generation control.	TX PRBS Generator (page 145)
TXGEARBOXREADY0 TXGEARBOXREADY1	Out	TXUSRCLK2	Indicates when data must be applied to the TX Gearbox.	TX Gearbox (page 131)
TXHEADER0[2:0] TXHEADER1[2:0]	In	TXUSRCLK2	TX header input data to TX Gearbox.	TX Gearbox (page 131)
TXINHIBIT0 TXINHIBIT1	In	TXUSRCLK2	Inhibits data transmission.	Configurable TX Driver (page 148)
TXKERR0[3:0] TXKERR1[3:0]	Out	TXUSRCLK2	Indicates if an invalid code for a K character was specified.	Configurable 8B/10B Encoder (page 127)
TXOUTCLK0 TXOUTCLK1	Out	N/A	Provides a parallel clock generated by the internal dividers of the GTX transceiver. Note: When INTDATAWIDTH is High, the duty cycle is 60/40 instead of 50/50. TXOUTCLK cannot drive TXUSRCLK when the TX phase-alignment circuit is used.	FPGA TX Interface (page 117), TX Buffering, Phase Alignment, and TX Skew Reduction (page 140)
TXPMASETPHASE0 TXPMASETPHASE1	In	Async	Aligns XCLK with TXUSRCLK for both GTX transceivers in the GTX_DUAL tile.	TX Buffering, Phase Alignment, and TX Skew Reduction (page 140)
TXPOLARITY0 TXPOLARITY1	In	TXUSRCLK2	Specifies if the final transmitter output is inverted.	TX Polarity Control (page 144)

Table 1-3: **GTX_DUAL Port Summary *(Cont'd)***

Port	Dir	Domain	Description	Section (Page)
TXPOWERDOWN0[1:0] TXPOWERDOWN1[1:0]	In	TXUSRCLK2(1)	Powers down the TX lanes.	Power Control (page 108), Receive Detect Support for PCI Express Operation (page 152), TX Out-of-Band/Beacon Signaling (page 154)
TXPREEMPHASIS0[3:0] TXPREEMPHASIS1[3:0]	In	Async	Controls the pre-emphasis.	Configurable TX Driver (page 148)
TXRESET0 TXRESET1	In	Async	Resets the PCS of the GTX transmitter, including the phase adjust FIFO, the 8B/10B encoder, and the FPGA TX interface.	Reset (page 99), FPGA TX Interface (page 118)
TXRUNDISP0[3:0] TXRUNDISP1[3:0]	Out	TXUSRCLK2	Indicates the current running disparity of the 8B/10B encoder.	Configurable 8B/10B Encoder (page 127)
TXSEQUENCE0[6:0] TXSEQUENCE1[6:0]	In	TXUSRCLK2	Input to the TX Gearbox from a sequence counter implemented in the FPGA logic.	TX Gearbox (page 131)
TXSTARTSEQ0 TXSTARTSEQ1	In	TXUSRCLK2	Input to the TX Gearbox from the FPGA logic indicating the start of a TX sequence.	TX Gearbox (page 131)
TXUSRCLK0 TXUSRCLK1	In	N/A	Provides a clock for the internal TX PCS datapath.	FPGA TX Interface (page 118), TX Buffering, Phase Alignment, and TX Skew Reduction (page 140)
TXUSRCLK20 TXUSRCLK21	In	N/A	Synchronizes the FPGA logic with the TX interface.	FPGA TX Interface (page 118)

Notes:

1. TXPOWERDOWN0[1:0] and TXPOWERDOWN1[1:0] of the GTX_DUAL tile belong to the TXUSRCLK2 clock domain. This is different from the GTP_DUAL tile implementation where TXPOWERDOWN0[1:0] and TXPOWERDOWN1[1:0] are asynchronous.

Table 1-4 lists alphabetically the signal names, clock domains, directions, and descriptions for the CRC ports, and provides links to their detailed descriptions.

Table 1-4: **CRC Port Summary**

Port	Dir	Domain	Description	Section (Page)
CRCCLK	In	N/A	CRC clock.	Cyclic Redundancy Check (page 236)
CRCDATAVALID	In	CRCCLK	Indicates valid data on 32-bit CRCIN inputs.	Cyclic Redundancy Check (page 237)
CRCDATAVALIDA	In	CRCCLK	Indicates valid data on 64-bit CRCIN inputs.	Cyclic Redundancy Check (page 236)
CRCDATAWIDTH[2:0]	In	CRCCLK	Indicates how many input data bytes are valid.	Cyclic Redundancy Check (page 236)

Table 1-4: **CRC Port Summary *(Cont'd)***

Port	Dir	Domain	Description	Section (Page)
CRCIN[63:0]	In	CRCCLK	CRC input data. The maximum datapath width is eight bytes.	Cyclic Redundancy Check (page 236)
CRCOUT[31:0]	Out	CRCCLK	32-bit CRC output. CRCOUT is the byte-reversed, bit-inverted CRC value corresponding to the CRC calculation on valid bytes from the previous clock cycle and the previous CRC value. CRCDATAVALIDA must be driven High.	Cyclic Redundancy Check (page 236)
CRCRESET	In	CRCCLK	Synchronous reset of CRC registers.	Cyclic Redundancy Check (page 236)

Table 1-5 lists alphabetically the attribute names, default values, and directions of the GTX_DUAL attributes, and provides links to their detailed descriptions.

Table 1-5: **GTX_DUAL Attribute Summary**

Attribute	Type	Description	Section (Page)
AC_CAP_DIS_0 AC_CAP_DIS_1	Boolean	Disables built-in AC coupling capacitors on receiver inputs when set to TRUE.	RX Termination and Equalization (page 159)
ALIGN_COMMA_WORD_0 ALIGN_COMMA_WORD_1	Integer	Controls alignment of detected commas within a multi-byte datapath.	Configurable Comma Alignment and Detection (page 191)
CB2_INH_CC_PERIOD_0 CB2_INH_CC_PERIOD_1	Integer	Specific for PCI Express designs, allows removal or retaining of control characters during channel bonding and clock correction.	Configurable Channel Bonding (Lane Deskew) (page 217)
CDR_PH_ADJ_TIME	5-bit Binary	Defines the waiting time after deassertion of the CDR phase reset before the optional reset sequence for PCI Express designs is complete during electrical idle.	Reset (page 100), RX Clock Data Recovery (page 177)
CHAN_BOND_1_MAX_SKEW_0 CHAN_BOND_1_MAX_SKEW_1 CHAN_BOND_2_MAX_SKEW_0 CHAN_BOND_2_MAX_SKEW_1	Integer	Sets the maximum amount of lane skew allowed when using channel bonding. Must be set less than one-half the minimum distance between channel bonding sequences.	Configurable Channel Bonding (Lane Deskew) (page 217)
CHAN_BOND_KEEP_ALIGN_0 CHAN_BOND_KEEP_ALIGN_1	Boolean	Specific for PCI Express designs, allows lanes to remember the previous skew during periods of electrical idle.	Configurable Channel Bonding (Lane Deskew) (page 217)
CHAN_BOND_LEVEL_0 CHAN_BOND_LEVEL_1	Integer	Indicates the amount of internal pipelining used for the elastic buffer control signals.	Configurable Channel Bonding (Lane Deskew) (page 218)

Table 1-5: **GTX_DUAL Attribute Summary *(Cont'd)***

Attribute	Type	Description	Section (Page)
CHAN_BOND_MODE_0 CHAN_BOND_MODE_1	String	Defines the channel bonding mode of operation for the transceiver.	Configurable Channel Bonding (Lane Deskew) (page 218)
CHAN_BOND_SEQ_1_1_0 CHAN_BOND_SEQ_1_1_1 CHAN_BOND_SEQ_1_2_0 CHAN_BOND_SEQ_1_2_1 CHAN_BOND_SEQ_1_3_0 CHAN_BOND_SEQ_1_3_1 CHAN_BOND_SEQ_1_4_0 CHAN_BOND_SEQ_1_4_1	10-bit Binary	Used in conjunction with CHAN_BOND_SEQ_1_ENABLE to define channel bonding sequence 1.	Configurable Channel Bonding (Lane Deskew) (page 218)
CHAN_BOND_SEQ_1_ENABLE_0 CHAN_BOND_SEQ_1_ENABLE_1	4-bit Binary	Sets which parts of channel bonding sequence 1 are don't cares.	Configurable Channel Bonding (Lane Deskew) (page 218)
CHAN_BOND_SEQ_2_1_0 CHAN_BOND_SEQ_2_1_1 CHAN_BOND_SEQ_2_2_0 CHAN_BOND_SEQ_2_2_1 CHAN_BOND_SEQ_2_3_0 CHAN_BOND_SEQ_2_3_1 CHAN_BOND_SEQ_2_4_0 CHAN_BOND_SEQ_2_4_1	10-bit Binary	Used in conjunction with CHAN_BOND_SEQ_2_ENABLE to define the second channel bonding sequence.	Configurable Channel Bonding (Lane Deskew) (page 219)
CHAN_BOND_SEQ_2_ENABLE_0 CHAN_BOND_SEQ_2_ENABLE_1	4-bit Binary	Sets which parts of channel bonding sequence 2 are don't cares.	Configurable Channel Bonding (Lane Deskew) (page 219)
CHAN_BOND_SEQ_2_USE_0 CHAN_BOND_SEQ_2_USE_1	Boolean	Determines if the second channel bonding sequence is to be used.	Configurable Channel Bonding (Lane Deskew) (page 219)
CHAN_BOND_SEQ_LEN_0 CHAN_BOND_SEQ_LEN_1	Integer	Defines the length in bytes of the channel bonding sequence that the transceiver matches to detect opportunities for channel bonding.	Configurable Channel Bonding (Lane Deskew) (page 219)
CLK_COR_ADJ_LEN_0 CLK_COR_ADJ_LEN_1	Integer	Defines the size of the adjustment (number of bytes repeated or skipped) in a clock correction.	Configurable Clock Correction (page 211)
CLK_COR_DET_LEN_0 CLK_COR_DET_LEN_1	Integer	Defines the length of the sequence that the transceiver matches to detect opportunities for clock correction.	Configurable Clock Correction (page 212)
CLK_COR_INSERT_IDLE_FLAG_0 CLK_COR_INSERT_IDLE_FLAG_1	Boolean	Controls whether RXRUNDISP input status indicates running disparity or inserted-idle (clock correction sequence) flag.	Configurable Clock Correction (page 212)

Table 1-5: **GTX_DUAL Attribute Summary *(Cont'd)***

Attribute	Type	Description	Section (Page)
CLK_COR_KEEP_IDLE_0 CLK_COR_KEEP_IDLE_1	Boolean	Controls whether the elastic buffer must retain at least one clock correction sequence in the byte stream.	Configurable Clock Correction (page 212)
CLK_COR_MAX_LAT_0 CLK_COR_MAX_LAT_1	Integer	Specifies the maximum elastic buffer latency.	Configurable Clock Correction (page 212)
CLK_COR_MIN_LAT_0 CLK_COR_MIN_LAT_1	Integer	Specifies the minimum elastic buffer latency.	Configurable Clock Correction (page 212)
CLK_COR_PRECEDENCE_0 CLK_COR_PRECEDENCE_1	Boolean	Determines whether clock correction or channel bonding takes precedence when both operations are triggered at the same time. Set to TRUE to give clock correction precedence.	Configurable Clock Correction (page 212)
CLK_COR_REPEAT_WAIT_0 CLK_COR_REPEAT_WAIT_1	Integer	Specifies the minimum number of RXUSRCLK cycles without clock correction that must occur between successive clock corrections.	Configurable Clock Correction (page 212)
CLK_COR_SEQ_1_1_0 CLK_COR_SEQ_1_1_1 CLK_COR_SEQ_1_2_0 CLK_COR_SEQ_1_2_1 CLK_COR_SEQ_1_3_0 CLK_COR_SEQ_1_3_1 CLK_COR_SEQ_1_4_1	10-bit Binary	The CLK_COR_SEQ_1 attributes are used in conjunction with CLK_COR_SEQ_1_ENABLE to define clock correction sequence 1.	Configurable Clock Correction (page 213)
CLK_COR_SEQ_1_ENABLE_0 CLK_COR_SEQ_1_ENABLE_1	4-bit Binary	Sets which parts of clock correction sequence 1 are don't cares.	Configurable Clock Correction (page 213)
CLK_COR_SEQ_2_1_0 CLK_COR_SEQ_2_1_1 CLK_COR_SEQ_2_2_0 CLK_COR_SEQ_2_2_1 CLK_COR_SEQ_2_3_0 CLK_COR_SEQ_2_3_1 CLK_COR_SEQ_2_4_0 CLK_COR_SEQ_2_4_1	10-bit Binary	Used in conjunction with CLK_COR_SEQ_2_ENABLE to define the second clock correction sequence.	Configurable Clock Correction (page 213)
CLK_COR_SEQ_2_ENABLE_0 CLK_COR_SEQ_2_ENABLE_1	4-bit Binary	Sets which parts of clock correction sequence 2 are don't cares.	Configurable Clock Correction (page 213)
CLK_COR_SEQ_2_USE_0 CLK_COR_SEQ_2_USE_1	Boolean	Determines if the second clock correction sequence is to be used.	Configurable Clock Correction (page 213)
CLK_CORRECT_USE_0 CLK_CORRECT_USE_1	Boolean	Set to TRUE to enable clock correction.	Configurable Clock Correction (page 211)

Table 1-5: **GTX_DUAL Attribute Summary *(Cont'd)***

Attribute	Type	Description	Section (Page)
CLK25_DIVIDER	Integer	Sets the divider used to divide CLKIN down to an internal rate close to 25 MHz.	Clocking (page 95), Power Control (page 108)
CLKINDC_B	Boolean	Must be set to TRUE. Oscillators driving the dedicated reference clock inputs must be AC coupled.	Clocking (page 95), Analog Design Guidelines (page 250)
CLKRCV_TRST	Boolean	When set to TRUE, switches on the differential clock input pair's internal termination resistors.	Clocking (page 95), Analog Design Guidelines (page 250)
CM_TRIM_0 CM_TRIM_1	2-bit Binary	Adjusts the input common mode values.	RX Termination and Equalization (page 159)
COM_BURST_VAL_0[3:0] COM_BURST_VAL_1[3:0]	4-bit Binary	Number of bursts transmitted for a SATA COM sequence.	TX Out-of-Band/Beacon Signaling (page 155)
COMMA_10B_ENABLE_0 COMMA_10B_ENABLE_1	10-bit Binary	Sets which bits of MCOMMA/PCOMMA must be matched to incoming data and which bits are don't cares.	Configurable Comma Alignment and Detection (page 191)
COMMA_DOUBLE_0 COMMA_DOUBLE_1	Boolean	When TRUE, a PCOMMA match followed immediately by an MCOMMA match is required for comma detection. Used to detect A1/A2 framing characters for SONET.	Configurable Comma Alignment and Detection (page 191)
DEC_MCOMMA_DETECT_0 DEC_MCOMMA_DETECT_1	Boolean	Enables detection of negative 8B/10B commas.	Configurable 8B/10B Decoder (page 199)
DEC_PCOMMA_DETECT_0 DEC_PCOMMA_DETECT_1	Boolean	Enables detection of positive 8B/10B commas.	Configurable 8B/10B Decoder (page 199)
DEC_VALID_COMMA_ONLY_0 DEC_VALID_COMMA_ONLY_1	Boolean	Limits the set of commas to which RXCHARISCOMMA responds.	Configurable 8B/10B Decoder (page 199)
DFE_CAL_TIME	5-bit Binary	DFE calibration time.	Decision Feedback Equalization (page 164)
DFE_CFG_0[9:0] DFE_CFG_1[9:0]	10-bit Binary	DFE configuration settings.	Decision Feedback Equalization (page 164)
GEARBOX_ENDEC_0 GEARBOX_ENDEC_1	3-bit Binary	Selects Gearbox mode.	TX Gearbox (page 132), RX Gearbox (page 227)
MCOMMA_10B_VALUE_0 MCOMMA_10B_VALUE_1	10-bit Binary	Defines comma minus to raise RXCOMMADET and align the parallel data.	Configurable Comma Alignment and Detection (page 192)
MCOMMA_DETECT_0 MCOMMA_DETECT_1	Boolean	Set to TRUE to allow minus comma detection and alignment.	Configurable Comma Alignment and Detection (page 192)
OOB_CLK_DIVIDER	Integer	Sets the squelch clock rate based on CLKIN.	RX OOB/Beacon Signaling (page 172)

Table 1-5: **GTX_DUAL Attribute Summary *(Cont'd)***

Attribute	Type	Description	Section (Page)
OOBDETECT_THRESHOLD_0 OOBDETECT_THRESHOLD_1	3-bit Binary	Sets the minimum differential voltage between RXN and RXP before a signal is recognized as a valid electrical idle for a PCI Express operation or a SATA OOB signal.	RX OOB/Beacon Signaling (page 172)
OVERSAMPLE_MODE	Boolean	Enables 5x oversampling.	Shared PMA PLL (page 85), TX Buffering, Phase Alignment, and TX Skew Reduction (page 141), Parallel In to Serial Out (page 146), Serial In to Parallel Out (page 181), Oversampling (page 184), Configurable RX Elastic Buffer and Phase Alignment (page 204)
PCI_EXPRESS_MODE_0 PCI_EXPRESS_MODE_1	Boolean	Enables specific PCI Express operations.	Power Control (page 108), Configurable Channel Bonding (Lane Deskew) (page 219)
PCOMMA_10B_VALUE_0 PCOMMA_10B_VALUE_1	10-bit Binary	Defines comma plus to raise RXCOMMADET and align the parallel data.	Configurable Comma Alignment and Detection (page 192)
PCOMMA_DETECT_0 PCOMMA_DETECT_1	Boolean	Set to TRUE to allow plus comma detection and alignment.	Configurable Comma Alignment and Detection (page 192)
PLL_COM_CFG	24-bit Hex	Controls PLL configuration.	
PLL_CP_CFG	8-bit Hex	Controls PLL feedback loop.	
PLL_DIVSEL_FB	Integer	Controls the feedback divider of the shared PMA PLL.	Shared PMA PLL (page 85)
PLL_DIVSEL_REF	Integer	Controls the reference clock divider of the shared PMA PLL.	Shared PMA PLL (page 85)
PLL_FB_DCCEN	Boolean	PLL DCC enable. Set to FALSE (default) to disable. Set to TRUE to enable.	
PLL_LKDET_CFG	3-bit Binary	Configuration for the shared PMA PLL lock detect circuit.	
PLL_RXDIVSEL_OUT_0 PLL_RXDIVSEL_OUT_1	Integer	Sets the divider for the RX line rate for each GTX transceiver.	Shared PMA PLL (page 85), Serial In to Parallel Out (page 181)
PLL_SATA_0 PLL_SATA_1	Boolean	Tie to FALSE. When FALSE, allows TX SATA operations to work at the SATA Generation 1 (1.5 Gb/s) or SATA Generation 2 (3 Gb/s) rate.	TX Out-of-Band/Beacon Signaling (page 155)

Table 1-5: **GTX_DUAL Attribute Summary *(Cont'd)***

Attribute	Type	Description	Section (Page)
PLL_TDCC_CFG	3-bit Binary	Configuration for the shared PMA PLL duty-cycle correction circuit.	
PLL_TXDIVSEL_OUT_0 PLL_TXDIVSEL_OUT_1	Integer	Sets the divider for the TX line rate for each GTX transceiver.	Shared PMA PLL (page 85), TX Buffering, Phase Alignment, and TX Skew Reduction (page 141), Parallel In to Serial Out (page 146), TX Out-of-Band/Beacon Signaling (page 155)
PMA_CDR_SCAN_0 PMA_CDR_SCAN_1	27-bit Hex	Allows direct control of the CDR sampling point.	RX Clock Data Recovery (page 177)
PMA_COM_CFG	69-bit Hex	Common configuration attribute for the PMA.	
PMA_RX_CFG_0 PMA_RX_CFG_1	25-bit Hex	Adjusts CDR operation for oversampling and PLL_RXDIVSEL_OUT settings.	RX Clock Data Recovery (page 177)
PMA_RXSYNC_CFG_0 PMA_RXSYNC_CFG_1	7-bit Hex	Configuration for the PMA RX low latency mode.	Oversampling (page 184)
PMA_TX_CFG_0 PMA_TX_CFG_1	20-bit Hex	TX channel specific settings.	TX Buffering, Phase Alignment, and TX Skew Reduction (page 141)
PRBS_ERR_THRESHOLD_0 PRBS_ERR_THRESHOLD_1	32-bit Hex	Sets the error threshold for the PRBS checker.	PRBS Detection (page 188)
RCV_TERM_GND_0 RCV_TERM_GND_1	Boolean	Sets the RX termination voltage to GND. Used with internal and external AC coupling to support TXDETECTRX functionality for PCI Express operation.	RX Termination and Equalization (page 159)
RCV_TERM_VTTRX_0 RCV_TERM_VTTRX_1	Boolean	Sets the RX termination voltage to MGTAVTTRX.	RX Termination and Equalization (page 159)
RX_BUFFER_USE_0 RX_BUFFER_USE_1	Boolean	Set to TRUE to use the RX elastic buffer.	Configurable RX Elastic Buffer and Phase Alignment (page 204)
RX_DECODE_SEQ_MATCH_0 RX_DECODE_SEQ_MATCH_1	Boolean	Determines whether sequences are matched against 8B/10B decoded data or undecoded data.	Configurable Clock Correction (page 213)
RX_EN_IDLE_HOLD_CDR	Boolean	Enables the CDR to hold data during an optional reset sequence of an electrical idle state for PCI Express designs.	Reset (page 100), RX Clock Data Recovery (page 177)

Table 1-5: **GTX_DUAL Attribute Summary *(Cont'd)***

Attribute	Type	Description	Section (Page)
RX_EN_IDLE_HOLD_DFE_0 RX_EN_IDLE_HOLD_DFE_1	Boolean	When TRUE, restores the DFE contents from internal registers after termination of an electrical idle state for PCI Express designs.	Reset (page 100), Decision Feedback Equalization (page 164)
RX_EN_IDLE_RESET_BUF_0 RX_EN_IDLE_RESET_BUF_1	Boolean	When TRUE, and no valid signal is present at the RX inputs, resets the RX elastic buffer.	Reset (page 100), Configurable RX Elastic Buffer and Phase Alignment (page 204)
RX_EN_IDLE_RESET_FR	Boolean	When TRUE, enables the reset of the CDR frequency circuits by using an optional reset sequence while in an electrical idle state for PCI Express designs.	Reset (page 100), RX Clock Data Recovery (page 178)
RX_EN_IDLE_RESET_PH	Boolean	When TRUE, enables the reset of the CDR phase circuits by using an optional reset sequence while in an electrical idle state for PCI Express designs.	Reset (page 100), RX Clock Data Recovery (page 178)
RX_IDLE_HI_CNT_0 RX_IDLE_HI_CNT_1	4-bit Binary	Determines how long valid data on the RX inputs must be absent before asserting the reset signal to the RX elastic buffer.	Reset (page 100), Configurable RX Elastic Buffer and Phase Alignment (page 204)
RX_IDLE_LO_CNT_0 RX_IDLE_LO_CNT_1	4-bit Binary	Determines how long valid data on the RX inputs must be present before deasserting the reset signal to the RX elastic buffer.	Reset (page 100), Configurable RX Elastic Buffer and Phase Alignment (page 204)
RX_LOS_INVALID_INCR_0 RX_LOS_INVALID_INCR_1	Integer	Defines the number of valid characters required to decrement the error count by 1 for the purpose of loss-of-sync determination.	Configurable Loss-of-Sync State Machine (page 196)
RX_LOS_THRESHOLD_0 RX_LOS_THRESHOLD_1	Integer	Defines the error count required to move the Loss of Sync state machine from the SYNC_ACQUIRED to the SYNC_LOST state.	Configurable Loss-of-Sync State Machine (page 196)
RX_LOSS_OF_SYNC_FSM_0 RX_LOSS_OF_SYNC_FSM_1	Boolean	Defines the behavior of the RXLOSSOFSYNC outputs.	Configurable Loss-of-Sync State Machine (page 196)
RX_SLIDE_MODE_0 RX_SLIDE_MODE_1	String	Selects between sliding in the PMA or in the PCS.	Configurable Comma Alignment and Detection (page 192)
RX_STATUS_FMT_0 RX_STATUS_FMT_1	String	Sets whether the RX_STATUS port is used to report the status of PCI Express or SATA features.	RX OOB/Beacon Signaling (page 172)

Table 1-5: **GTX_DUAL Attribute Summary *(Cont'd)***

Attribute	Type	Description	Section (Page)
RX_XCLK_SEL_0 RX_XCLK_SEL_1	String	Selects which clock is used on the PMA side of the RX elastic buffer. The default setting is RXREC (RX recovered clock). Use RXUSR (RXUSRCLK) when bypassing the RX elastic buffer.	Configurable RX Elastic Buffer and Phase Alignment (page 204)
RXGEARBOX_USE_0 RXGEARBOX_USE_1	Boolean	Enables the RX Gearbox.	RX Gearbox (page 227)
SATA_BURST_VAL_0 SATA_BURST_VAL_1	3-bit Binary	Number of bursts required for the SATA OOB detector to declare a COM match.	RX OOB/Beacon Signaling (page 172)
SATA_IDLE_VAL_0 SATA_IDLE_VAL_1	3-bit Binary	Number of idles required for the SATA OOB detector to declare a COM match.	RX OOB/Beacon Signaling (page 172)
SATA_MAX_BURST_0 SATA_MAX_BURST_1	Integer	Sets the threshold for the SATA detector to reject a burst in terms of squelch clock cycles.	RX OOB/Beacon Signaling (page 172)
SATA_MAX_INIT_0 SATA_MAX_INIT_1	Integer	Sets the maximum time allowed for a COMINIT/COMRESET idle for the SATA detector in terms of squelch clock cycles.	RX OOB/Beacon Signaling (page 172)
SATA_MAX_WAKE_0 SATA_MAX_WAKE_1	Integer	Sets the maximum time allowed for a COMWAKE idle for the SATA detector in terms of squelch clock cycles.	RX OOB/Beacon Signaling (page 172)
SATA_MIN_BURST_0 SATA_MIN_BURST_1	Integer	Sets the threshold for the SATA detector to reject a burst in terms of squelch clock cycles.	RX OOB/Beacon Signaling (page 173)
SATA_MIN_INIT_0 SATA_MIN_INIT_1	Integer	Sets the minimum time allowed for a COMINIT/COMRESET idle for the SATA detector in terms of squelch clock cycles.	RX OOB/Beacon Signaling (page 173)
SATA_MIN_WAKE_0 SATA_MIN_WAKE_1	Integer	Sets the minimum time allowed for a COMWAKE idle for the SATA detector in terms of squelch clock cycles.	RX OOB/Beacon Signaling (page 173)
SIM_GTXRESET_SPEEDUP	Integer	Shortens the time it takes to finish the GTXRESET sequence and PLL lock during simulation.	Simulation (page 52)
SIM_MODE	String	This simulation-only attribute chooses between FAST and LEGACY simulation models.	Simulation (page 52)
SIM_PLL_PERDIV2	9-bit Hex	Specifies the length of one symbol in picoseconds for simulation.	Simulation (page 52)
SIM_RECEIVER_DETECT_PASS0 SIM_RECEIVER_DETECT_PASS1	Boolean	Controls the receiver detect modeling in simulation and is intended for PCI Express designs only.	Simulation (page 52)

Table 1-5: **GTX_DUAL Attribute Summary *(Cont'd)***

Attribute	Type	Description	Section (Page)
TERMINATION_CTRL[4:0]	5-bit Binary	Controls internal termination calibration circuit.	Analog Design Guidelines (page 250)
TERMINATION_IMP_0 TERMINATION_IMP_1	Integer	Set to 50. Implies 50Ω terminated inputs and outputs with a differential impedance of 100Ω.	RX Termination and Equalization (page 159), Analog Design Guidelines (page 251)
TERMINATION_OVRD	Boolean	Selects whether the external 59Ω(2) precision resistor connected to the MGTRREF pin or an override value is used, as defined by TERMINATION_CTRL.	Analog Design Guidelines (page 251)
TRANS_TIME_FROM_P2_0 TRANS_TIME_FROM_P2_1	12-bit Hex	Transition time from the P2 power-down state in internal 25 MHz clock cycles. The exact time depends on the CLKIN rate and the setting of CLK25_DIVIDER.	Power Control (page 108)
TRANS_TIME_NON_P2_0 TRANS_TIME_NON_P2_1	8-bit Hex	Transition time to or from any power-down state except P2 in internal 25 MHz clock cycles. The exact time depends on the CLKIN rate and the setting of CLK25_DIVIDER.	Power Control (page 108)
TRANS_TIME_TO_P2_0 TRANS_TIME_TO_P2_1	10-bit Hex	Transition time to the P2 power-down state in internal 25 MHz clock cycles. The exact time depends on the CLKIN rate and the setting of CLK25_DIVIDER.	Power Control (page 108)
TX_BUFFER_USE_0 TX_BUFFER_USE_1	Boolean	Indicates whether or not the TX buffer is used.	TX Buffering, Phase Alignment, and TX Skew Reduction (page 141)
TX_DETECT_RX_CFG_0 TX_DETECT_RX_CFG_1	14-bit Hex	Configuration for the transmitter detect remote receiver circuit.	
TXGEARBOX_USE_0 TXGEARBOX_USE_1	Boolean	Enables TX Gearbox.	TX Gearbox (page 132)
TX_IDLE_DELAY_0 TX_IDLE_DELAY_1	3-bit Binary	Sets the idle delay.	
TXRX_INVERT0 TXRX_INVERT1	3-bit Binary	Controls inverters that optimize the clock paths within the GTX transceiver. When bypassing the TX buffer, set to `111`. Otherwise, set to `011`.	TX Buffering, Phase Alignment, and TX Skew Reduction (page 141)

Table 1-5: **GTX_DUAL Attribute Summary *(Cont'd)***

Attribute	Type	Description	Section (Page)
TX_XCLK_SEL_0 TX_XCLK_SEL_1	String	Selects the clock used to drive the clock domain in the PCS following the TX buffer. Set to TXOUT (TXOUTCLK) when using the TX buffer. Set to TXUSR (TXUSRCLK) when bypassing the TX buffer.	TX Buffering, Phase Alignment, and TX Skew Reduction (page 141)

Notes:

1. Refer to Appendix D for information about the mapping of these attributes to DRP binary values.
2. The nominal value of the external precision resistor RREF connected to the MGTRREF pin is different for LXT/SXT devices and FXT/TXT devices. For LXT/SXT devices with GTP_DUAL tiles, RREF is 50Ω For FXT/TXT devices with GTX_DUAL tiles, RREF is 59Ω TXT devices need an RREF resistor for each column of GTX_DUAL tiles.

Table 1-6 lists the attribute name, default value, and direction of the CRC attribute, and provides a link to the detailed description.

Table 1-6: **CRC Attribute Summary**

Attribute	Type	Description	Section (Page)
CRC_INIT[31:0]	32-bit Hex	32-bit value for initial state of CRC internal registers in the CRC32/CRC64 block.	Cyclic Redundancy Check (page 237)

Chapter 2

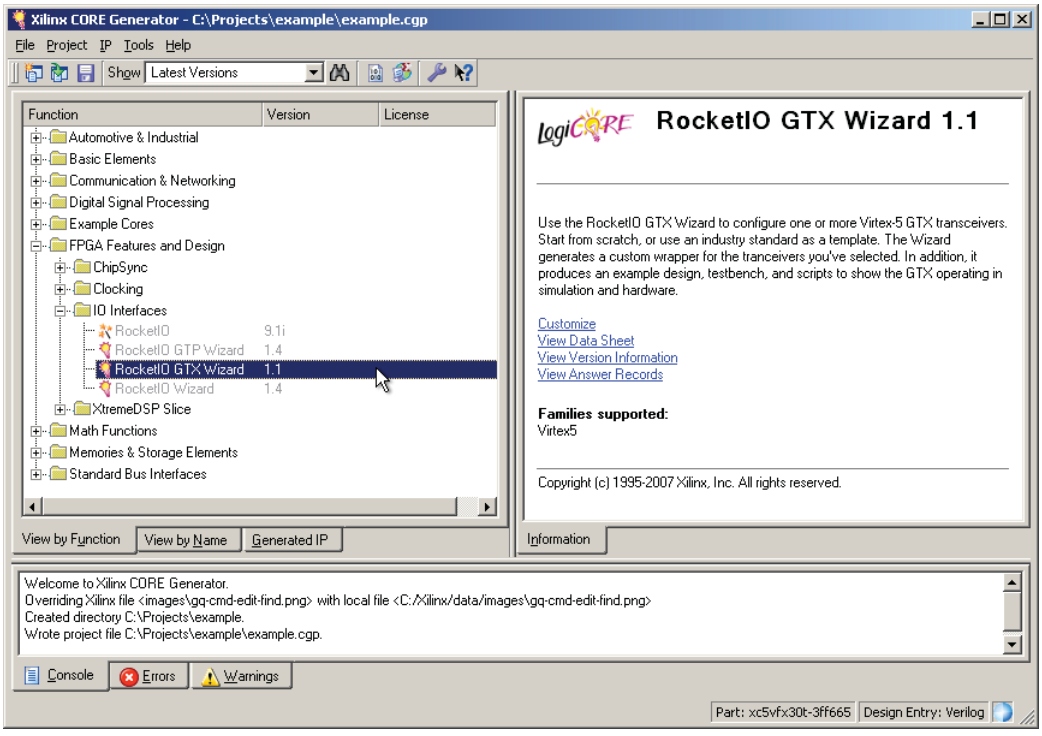
RocketIO GTX Transceiver Wizard

The RocketIO GTX Transceiver Wizard is the preferred tool to generate a wrapper to instantiate a GTX_DUAL primitive. The Wizard can be found in the Xilinx CORE Generator™ tool. Be sure to download the most up-to-date IP Update before using the Wizard. Details on how to use this wizard can be found in UG204, *RocketIO GTX Transceiver Getting Started Guide*.

1. Start the Xilinx CORE Generator tool.
2. Locate the RocketIO GTX Wizard in the taxonomy tree under:

   `/FPGA Features & Design/IO Interfaces`

   See Figure 2-1.



UG198_c2_01_070507

Figure 2-1: **Locating the RocketIO GTX Wizard**

3. Double click **`RocketIO GTX Wizard`** to launch the Wizard.

XILINX®

Chapter 3

Simulation

Overview

Simulations using GTX_DUAL tiles have specific prerequisites that the simulation environment and the test bench must fulfill.

The *Synthesis and Simulation Design Guide* [Ref 3] explains how to set up the simulation environment for supported simulators depending on the used Hardware Description Language (HDL). This design guide can be downloaded from the Xilinx website at: http://www.xilinx.com/support/software_manuals.htm

The prerequisites for simulating a design with GTX transceivers are:

- Simulator with a SWIFT interface to support *SmartModels*, which are encrypted versions of the HDL used for implementation of the modeled block
- Installed GTX_DUAL SmartModel
- Correct setting of the environment variable that points to the SmartModel installation directory
- Correct setup of the simulator for SmartModel use (initialization file, environment variable(s))
- Compilation of the SmartModel wrapper files into the *UNISIM* and *SIMPRIM* libraries
- Compilation of the GTX_DUAL SmartModel into a simulation library
- Correct simulator resolution (Verilog)
- Correct compilation order of simulation libraries

The user guide of the simulator and the *Synthesis and Simulation Design Guide* provide a detailed list of settings for SmartModel support. The COMPXLIB tool with sl_admin facilitates the setup of the supported simulator.

Ports and Attributes

The GTX_DUAL primitive has attributes intended only for simulation. Table 3-1 lists the *simulation-only* attributes of the GTX_DUAL tile. The names of these attributes start with *SIM_*.

Table 3-1: **GTX_DUAL Simulation-Only Attributes**

Attribute	Type	Description
SIM_GTXRESET_SPEEDUP	Integer	This attribute shortens the time it takes to finish the GTXRESET sequence and lock the shared PMA PLL during simulation. Must be used together with the correct setting of SIM_PLL_PERDIV2. `1`: Shorten the GTXRESET cycle time (fast initialization is approximately 300 ns). The value of SIM_PLL_PERDIV2 defines the PLL frequency in this mode. Because SIM_PLL_PERDIV2 cannot be changed on the fly during simulation, this mode cannot be used for multirate designs. `0`: The GTXRESET sequence is simulated with its original duration (standard initialization is approximately 160 µs). This mode must be used for multirate designs.
SIM_MODE	String	This simulation-only attribute chooses between two available UNISIM/SIMPRIM simulation models. FAST: When this attribute is set to FAST, a faster simulation model of the PMA is used to cut simulation run time. LEGACY: When this attribute is set to LEGACY, the legacy simulation model of the PMA is used, which results in a longer simulation run time.
SIM_PLL_PERDIV2	9-bit Hex	This attribute specifies a 9-bit hex value equal to half the period of the PLL clock frequency in picoseconds. For example, 400 ps (decimal) is equal to `0x190` (hexadecimal), which is the default value. If SIM_PLL_PERDIV2 is not set correctly, poor locking behavior and incorrect clock frequencies occur in simulation.
SIM_RECEIVER_DETECT_PASS0 SIM_RECEIVER_DETECT_PASS1	Boolean	This attribute is used to simulate the TXDETECTRX feature in each GTX transceiver. TRUE (default): Simulates an RX connection to the TX serial ports. TXDETECTRX reports that an RX port is connected. FALSE: Simulates a disconnected TX port. TXDETECTRX reports that the RX port is not detected.

There are no simulation-only ports.

Description

The behavior of the GTX_DUAL tile is modeled using a SmartModel. The SmartModel allows the design containing GTX_DUAL tiles to be simulated in the following design phases:

- Register Transfer Level (RTL)/Pre-Synthesis Simulation
- Post-Synthesis Simulation/Pre-NGDBuild Simulation

- Post-NGDBuild/Pre-Map Simulation
- Post-Map/Partial Timing Simulation
- Post-Place and Route/Timing Simulation

Limitations

The analog nature of some blocks inside the GTX_DUAL tile generates some restrictions when simulated using an HDL simulator. Receiver detection and OOB/beacon signaling are analog features of the GTX_DUAL tile that can only be modeled in a limited way with an HDL simulator. The shared PMA PLL is another analog block in the GTX_DUAL tile that is difficult to model precisely. The simulation-only attributes SIM_GTXRESET_SPEEDUP and SIM_PLL_PERDIV2 speed up the simulation by shortening the locking time of the shared PMA PLL.

SmartModel Attributes

SIM_GTXRESET_SPEEDUP

The SIM_GTXRESET_SPEEDUP attribute can be used to shorten the simulated lock time of the shared PMA PLL.

If TXOUTCLK or RXRECCLK is used to generate clocks in the design, these clocks occasionally flatline while the GTX_DUAL tile is locking. If a PLL or a digital clock manager (DCM) is used to divide TXOUTCLK or RXRECCLK, the final output clock is not ready until both the GTX_DUAL tile and the PLL or DCM have locked. Equation 3-1 provides an estimate of the time required before a stable source from TXOUTCLK or RXRECCLK is available in simulation, including the time required for any PLLs or DCMs used.

$$t_{USRCLKstable} \cong t_{GTXRESETsequence} + t_{locktimePLL} + t_{locktimeDCM} \quad \textit{Equation 3-1}$$

If either the PLL or the DCM is not used, the respective term can be removed from the lock time equation. When simulating multirate designs where the shared PMA PLL frequency or REFCLK frequency changes, SIM_GTXRESET_SPEEDUP must be set to FALSE. Appendix F, "Advanced Clocking" illustrates multirate design examples.

SIM_MODE

This simulation-only attribute chooses between two available UNISIM/SIMPRIM simulation models. The LEGACY setting selects the legacy simulation model of the PMA of the GTX transceiver. The FAST setting selects a faster simulation model of the PMA of the GTX transceiver to cut simulation run time.

The Legacy model is available for existing users who have been using simulation models with ISE 11.1 and older software for their designs; however, this legacy model will be phased out after ISE® 11.1 software. The FAST setting is highly recommended for new customer designs.

This attribute can be used independently of the SIM_GTXRESET_SPEEDUP attribute.

SIM_PLL_PERDIV2

The GTX_DUAL tile contains an analog PLL to generate the transmit and receive clocks out of a reference clock. Because HDL simulators do not fully model the analog PLL, the GTX_DUAL Smartmodel includes an equivalent behavioral model to simulate the PLL

output. The SIM_PLL_PERDIV2 attribute is used by the behavioral model to generate the PLL output as accurately as possible. It must be set to one-half the period of the shared PMA PLL. See "Examples," page 56 for how to calculate SIM_PLL_PERDIV2 for a given rate.

SIM_RECEIVER_DETECT_PASS

The GTX_DUAL includes a TXDETECTRX feature that allows the transmitter to detect whether its serial ports are currently connected to a receiver by measuring rise time on the TXP/TXN differential pin pair (see "Receive Detect Support for PCI Express Operation," page 151).

The GTX_DUAL SmartModel includes an attribute for simulating TXDETECTRX called SIM_RECEIVER_DETECT_PASS. This attribute allows TXDETECTRX to be simulated for each GTX transceiver without modelling the measurement of rise time on the TXP/TXN differential pin pair.

By default, SIM_RECEIVER_DETECT_PASS is set to TRUE. When TRUE, the attribute models a connected receiver, and TXDETECTRX operations indicate a receiver is connected. To model a disconnected receiver, SIM_RECEIVER_DETECT_PASS for the transceiver is set to FALSE.

Power-Up and Reset

Link Idle Reset

To simulate correctly, the Link Idle Reset circuit, described in"Link Idle Reset Support," page 102, must be implemented and connected to each GTX_DUAL instance. This circuit is included automatically when the Wizard is used to configure the GTX_DUAL instance.

Toggling GSR

The GSR signal is a global routing of nets in the design that provide a means of setting or resetting applicable components in the device during configuration.

The simulation behavior of this signal is modeled using the glbl module in Verilog and the ROC/ROCBUF components in VHDL.

Providing Clocks in Simulation

In simulation, the clocks inside the PMA are generated using the SIM_PLL_PERDIV2 parameter (in picoseconds). Any other clocks driven into the user clock must have the same level of precision, or TX buffer errors (and RX buffer errors in systems without clock correction) can result. When generating USRCLK, USRCLK2, or reference clock signals in the test bench, the clock periods must be related to SIM_PLL_PERDIV2 and also be a round number (in picoseconds). In some cases, the simulation a clock rate is slightly different from the clock rate used in the actual design.

Simulating in Verilog

The GSR signal is defined in the `$XILINX/verilog/src/glbl.v` module. Because the `glbl.v` module connects the global signal to the design, it is necessary to compile this module with the other design files and load it along with the `design.v` and `testfixture.v` files for simulation.

Note: There is an important difference between simulating GTP_DUAL tiles and GTX_DUAL tiles in Verilog. The simulation model of a GTP_DUAL tile requires an additional global 3-state signal (GTS). A GTX_DUAL tile simulation model does not use this additional GTS signal.

Defining GSR in a Test Bench

There are two ways to handle GSR in a test bench:

1. In most cases, GSR do not need to be defined in the test bench. The `glbl.v` file declares the GSR signals and automatically pulses GSR for 100 ns. This handling is sufficient for back-end simulations and functional simulations as well. The simulation model of the GTX_DUAL does not use the GTS signal.
2. If GSR needs to be emulated in the test bench, the following snippet of code must be added to the `testfixture.v` file:

```
assign glbl.GSR = gsr_r;
initial
begin
gsr_r = 1'b1;
#(16*CLOCKPERIOD);
gsr_r = 1'b0;
end
```

Simulating in VHDL

The `ROCBUF` cell controls the emulated GSR signal in a test bench. This component creates a buffer for the GSR signal and provides an input port on the buffer to drive GSR. This port must be declared in the entity list and driven through the test bench.

The VHDL code for this cell, located in `EX_ROCBUF.vhd`, is listed here:

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
library UNISIM;
use UNISIM.all;
entity EX_ROCBUF is
port (
CLOCK, ENABLE, SRP,RESET : in std_logic;
C_OUT: out std_logic_vector (3 downto 0)
);
end EX_ROCBUF;

architecture A of EX_ROCBUF is
signal GSR : std_logic;
signal COUNT : std_logic_vector (3 downto 0);
component ROCBUF
port (
I : in std_logic;
O : out std_logic
);
end component;
```

```
begin
U1 : ROCBUF port map (I => SRP, O => GSR);

//dummy process
COUNTER : process (CLOCK, ENABLE, RESET)
begin
....
end process COUNTER;
end A
```

The VHDL code for this test bench, located in `EX_ROCBUF_tb.vhd`, is listed here:

```
entity EX_ROCBUF_tb is
end EX_ROCBUF_tb;
architecture behavior of EX_ROCBUF_tb is
declare component EX_ROCBUF
declare signals
begin
EX_ROCBUF_inst: EX_ROCBUF PORT MAP(
CLOCK => CLOCK,
ENABLE => ENABLE,
SRP => SRP,
RESET => RESET,
COUT => COUT
);
Clk_generation: process
Begin
....
End process
reset <= '1', '0' after CLK_PERIOD * 30;
SRP <= '1', '0' after CLK_PERIOD * 25;
end
```

Further details can be found in the *Synthesis and Simulation Design Guide* [Ref 3].

Examples

Simulation Environment Setup Example (ModelSim SE 6.1e on Linux)

This section provides an example how to set up a simulation environment for SmartModel support. This is a prerequisite for simulating designs containing GTX_DUAL tile(s).

This example uses ModelSim SE 6.1e, the HDL simulator from Mentor Graphics, with RedHat Enterprise Linux 3.0 as the operating system and version 9.1i of the Xilinx ISE® development system. The *Synthesis and Simulation Guide* provides guidelines and examples for a different HDL simulator or ISE development system(1).

1. If there is a contradiction between this example and the documentation of your simulator, the simulator documentation has precedence. If a newer version of the Xilinx ISE development system is used, check the Xilinx website for additional information.

Use **setenv** to set these environment variables:

- XILINX — Location of the installed Xilinx ISE system (for example, /opt/Xilinx/ise_9_1_i)
- MODEL_TECH — Location of the installed ModelSim simulator (for example, /edatools/mentor/modelsim/6.1e/)
- LMC_HOME — $XILINX/smartmodel/lin/installed_lin
- LMC_CONFIG — $LMC_HOME/data/linux.lmc
- LD_LIBRARY_PATH — $LMC_HOME/lib/linux.lib:$LD_LIBRARY_PATH

The initialization file (Modelsim.ini) contains these settings:

```
libsm = $MODEL_TECH/libsm.sl
libswift = $LMC_HOME/lib/linux.lib/libswift.so
Resolution = 1ps ;(one picosecond simulator resolution)
```

The location of SmartModel in the ISE directory tree is:

```
$XILINX/virtex5/smartmodel/lin/image
```

The selected options for the COMPXLIB tool are:

```
compxlib -s mti_se -l all -arch all -smartmodel_setup
```

These options use the COMPXLIB tool to compile all libraries for all languages for the ModelSim SE 6.1e HDL simulator. The default output directory is $XILINX/language/target_simulator. The compiled libraries are specified to be written to $XILINX/vhdl/mti_se and $XILINX/verilog/mti_se.

SIM_PLL_PERDIV2 Calculation Example

This section provides examples of how to calculate the correct value for the simulation-only attribute SIM_PLL_PERDIV2.

The period of the PLL can be calculated using Equation 3-2 and Equation 3-3.

$$\text{PLL SPEED} = \left(\frac{\text{REFCLK}}{\text{PLL\_DIVSEL\_REF}}\right) \times \text{DIV} \times (\text{PLL\_DIVSEL\_FB}) \qquad \textit{Equation 3-2}$$

$$\text{SIM\_PLL\_PERDIV2} = \frac{(1/\ \text{PLL SPEED})}{2} \qquad \textit{Equation 3-3}$$

The terms used in Equation 3-2 and Equation 3-3 are defined as follows:

- REFCLK is the speed of the clock tied to the CLKIN input of the GTX_DUAL tile in MHz.
- PLL_DIVSEL_REF is an attribute that defines the dividing factor of the reference clock divider of the shared PMA PLL.
- PLL_DIVSEL_FB is an attribute that defines the dividing factor of the feedback divider (which acts like a multiplication factor) of the shared PMA PLL.
- DIV = 5 when INTDATAWIDTH = 1 (20-bit mode) OR when OVERSAMPLE_MODE = TRUE
- DIV = 4 when INTDATAWIDTH = 0 (16-bit mode) AND OVERSAMPLE_MODE = FALSE

Example for PCI Express Design

To calculate PLL SPEED and SIM_PLL_PERDIV2 for the PCI Express example, the following values are assigned:

- REFCLK = 100 MHz
- PLL_DIVSEL_REF = 1
- DIV = 5
- PLL_DIVSEL_FB = 5

Using Equation 3-2, PLL SPEED is 2.5 GHz, meaning that the period is 400 ps. Using Equation 3-3, SIM_PLL_PERDIV2 is 400 divided by 2 equal to 200 decimal (`C8` hexadecimal).

Example for Gigabit Ethernet Design

To calculate PLL SPEED and SIM_PLL_PERDIV2 for the Gigabit Ethernet example, the following values are assigned:

- REFCLK = 125 MHz
- PLL_DIVSEL_REF = 1
- DIV = 5
- PLL_DIVSEL_FB = 4

Using Equation 3-2, PLL SPEED is 2.5 GHz, meaning that the period is 400 ps. Using Equation 3-3, SIM_PLL_PERDIV2 is 400 divided by 2 or 200 decimal (`C8` hexadecimal).

Example for XAUI Design

To calculate PLL SPEED and SIM_PLL_PERDIV2 for the XAUI example, the following values are assigned:

- REFCLK = 312.5 MHz
- PLL_DIVSEL_REF = 1
- DIV = 5
- PLL_DIVSEL_FB = 1

Using Equation 3-2, PLL SPEED is 1.5625 GHz, meaning that the period is 640 ps. Using Equation 3-3, SIM_PLL_PERDIV2 is 640 divided by 2 or 320 decimal (`140` hexadecimal).

XILINX®

Chapter 4

Implementation

Overview

This chapter provides the information needed to map GTX_DUAL tiles instantiated in a design to device resources, including:

- The location of the GTX_DUAL tiles on the available device and package combinations.
- The pad numbers of external signals associated with each GTX_DUAL tile.
- How GTX_DUAL tiles and clocking resources instantiated in a design are mapped to available locations with a user constraints file (UCF).

It is a common practice to define the location of GTX transceivers early in the design process to ensure correct usage of clock resources and to facilitate signal integrity analysis during board design. The implementation flow facilitates this practice through the use of location constraints in the UCF.

While this chapter describes how to instantiate GTX_DUAL clocking components, the details of the different GTX_DUAL tile clocking options are discussed in "Clocking," page 93.

Ports and Attributes

Table 4-1 shows the external ports associated with each GTX_DUAL tile.

Table 4-1: **GTX_DUAL Tile External Ports**

Port	Dir	Domain	Description
MGTTXP0 MGTTXN0 MGTTXP1 MGTTXN1	Out	Embedded TX Clock	Differential transmit data pairs for GTX transceivers 0 and 1
MGTRXP0 MGTRXN0 MGTRXP1 MGTRXN1	In	Embedded RX Clock	Differential receive data pairs for GTX transceivers 0 and 1
MGTREFCLKP MGTREFCLKN	In	N/A	Differential reference clock input pair
MGTAVCCPLL	Analog	Analog	Pad for 1.0V supply for PLL

Table 4-1: **GTX_DUAL Tile External Ports *(Cont'd)***

Port	Dir	Domain	Description
MGTAVCC	Analog	Analog	Two pads for 1.0V supply for transceiver mixed signal circuitry
MGTAVTTRX	Analog	Analog	Pad for 1.2V supply for RX circuitry
MGTAVTTTX	Analog	Analog	Two pads for 1.2V supply for TX circuitry

Notes:

1. These port names have the prefix *MGT* to identify them easily in a pad file that is often used to create symbols for board design schematics. In this document, the MGT prefix was removed from those names; however, names with and without the MGT prefix are synonymous with each other.

There are no attributes for this section.

Description

The position of GTX_DUAL tiles is specified by an XY coordinate system that describes the column number and its relative position within that column. In current members of the Virtex-5 FXT platform, all GTX_DUAL tiles are located in a single column along one side of the die. TXT devices have one column of GTX_DUAL tiles on the right side of the die and one column of GTX_DUAL tiles on the left side of the device. As a result the X coordinate for all of the GTX_DUAL tiles is 0 on all FXT devices. On TXT devices, the left column is X0 and the right column is X1. "Package Placement Information," page 62 lists the GTX_DUAL tile position information for all available device and package combinations along with the pad numbers for the external signals associated with each tile. The Virtex-5 TXT devices have two columns of GTX_DUAL tiles. The left column on the die has the indices X0 and the right column of the die has the indices X1.

There are two ways to create a UCF for designs that utilize GTX_DUAL tiles. The preferred method is by using the RocketIO GTX Wizard (see Chapter 2, "RocketIO GTX Transceiver Wizard"). The Wizard automatically generates UCF templates that configure the transceivers and contain placeholders for GTX_DUAL placement information. The UCFs generated by the Wizard can then be edited to customize operating parameters and placement information for the application.

The second approach is to create the UCF by hand. When using this approach, the designer must enter both configuration attributes that control transceiver operation as well as tile location parameters. Care must be taken to ensure that all of the parameters needed to configure the GTX_DUAL tile are correctly entered.

Example of a UCF for GTX_DUAL Placement

This section shows key elements of a UCF that instantiates seven GTX_DUAL tiles. The file implements the example configuration shown in Figure 5-5, page 97. The device and package combination chosen in this example is an XC5VFX100T-FF1136.

```
;
; Instantiate the GTX_DUAL tiles in locations X0Y7 to X0Y1
;
INST design_root/gtx_dual[1]/gtx_dual   LOC=GTX_DUAL_X0Y1;
INST design_root/gtx_dual[2]/gtx_dual   LOC=GTX_DUAL_X0Y2;
INST design_root/gtx_dual[3]/gtx_dual   LOC=GTX_DUAL_X0Y3;
INST design_root/gtx_dual[4]/gtx_dual   LOC=GTX_DUAL_X0Y4;
INST design_root/gtx_dual[5]/gtx_dual   LOC=GTX_DUAL_X0Y5;
INST design_root/gtx_dual[6]/gtx_dual   LOC=GTX_DUAL_X0Y6;
INST design_root/gtx_dual[7]/gtx_dual   LOC=GTX_DUAL_X0Y7;
;
; Connect the REFCLK_PAD_(N/P) differential pair to the middle
; GTX_DUAL tile (GTX_DUAL_X0Y4)
;
NET refclk_pad_n LOC=P3;
NET refclk_pad_p LOC=P4;
```

The instantiation of the GTX_DUAL tiles and the IBUFDS primitive is typically done in HDL code within the design hierarchy. That code also connects the output of the IBUFDS primitive to the CLKIN inputs of the GTX_DUAL tiles, as illustrated by the following Verilog code fragment:

```
//
// Instantiate the GTX_DUAL tiles
//
genvar tile_num;

generate for (tile_num = 1; tile_num <= 7; ++tile_num)

  begin: gtx_dual

    GTX_DUAL gtx_dual
    (
      .CLKIN(refclk),

      … The remaining GTX_DUAL ports are not shown
    )

  end

endgenerate


//
// Instantiate the IBUFDS for the reference clock
//
IBUFDS ref_clk_buffer
(
  .O (refclk),
  .I (refclk_pad_p),
  .IB (refclk_pad_n)
)
```

Package Placement Information

The diagrams in this section illustrate GTX_DUAL placement for the following packages:

- FXT packages:
  - XC5VFX30T-FF665
  - XC5VFX70T-FF665
  - XC5VFX70T-FF1136
  - XC5VFX100T-FF1136
  - XC5VFX100T-FF1738
  - XC5VFX130T-FF1738
  - XC5VFX200T-FF1738
- TXT packages:
  - XC5VTX150T-FF1156
  - XC5VTX150T-FF1759
  - XC5VTX240T-FF1759

Figure 4-1 illustrates the nomenclature used in each of these diagrams. The GTX_DUAL placement name is the name used in the UCF to map GTX_DUAL tiles instantiated in the design to specific tiles on the device. The board-level pin names and numbers are the names placed in the PKG file generated by the ISE design flow. This file is typically used by board-level schematic capture and layout tools to create component symbols and layout footprints.

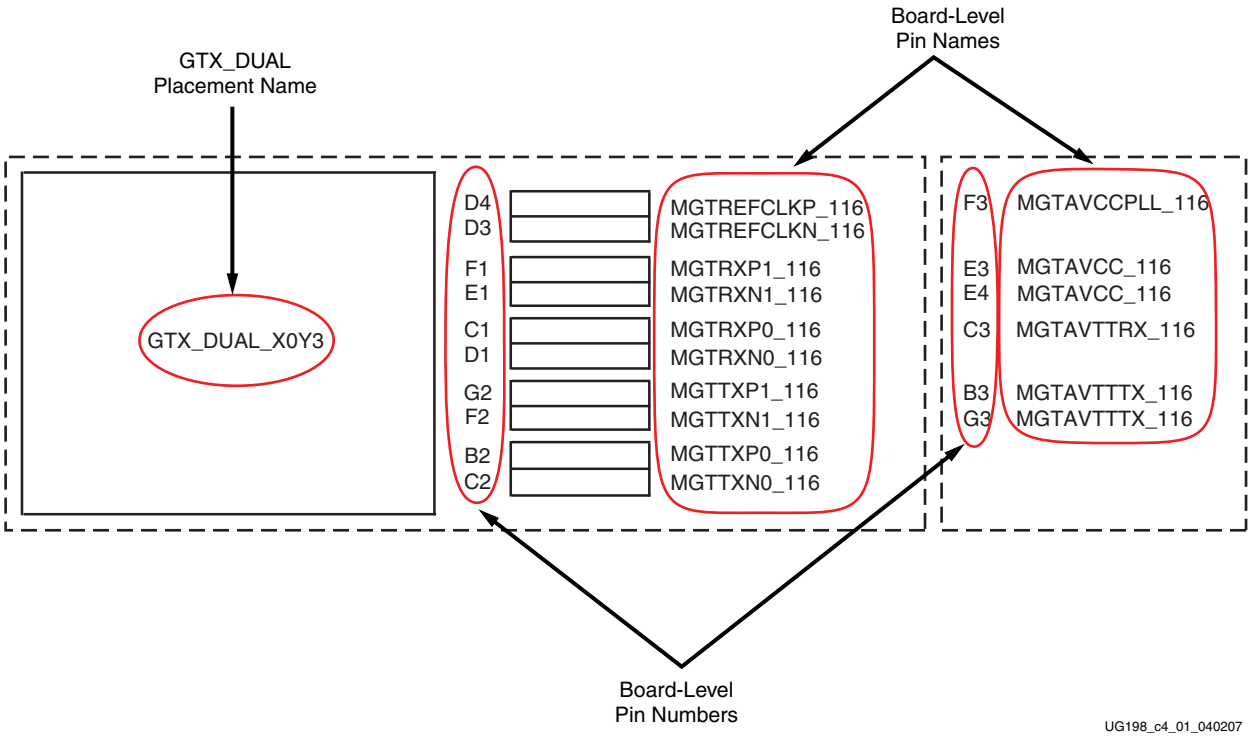


Figure 4-1: **Placement Diagram Nomenclature**

Table 4-2 defines the location of MGTRREF, MGTAVTTRXC, MGTRREF_R (TXT only), and MGTAVTTRXC_R (TXT only) for the different packages.

Table 4-2: **GTX_DUAL Analog Pin Placement**

Package	MGTRREF (FXT only)	MGTAVTTRXC (FXT only)	MGTRREF_L (TXT only)	MGTAVTTRXC_L (TXT only)	MGTRREF_R (TXT only)	MGTAVTTRXC_R (TXT only)
665	P4	P5	–	–	–	–
1136	V4	V5	–	–	–	–
1156	–	–	V31	U31	V4	U4
1738	AB4	AA5	–	–	–	–
1759	–	–	AB39	AA39	AB4	AA4

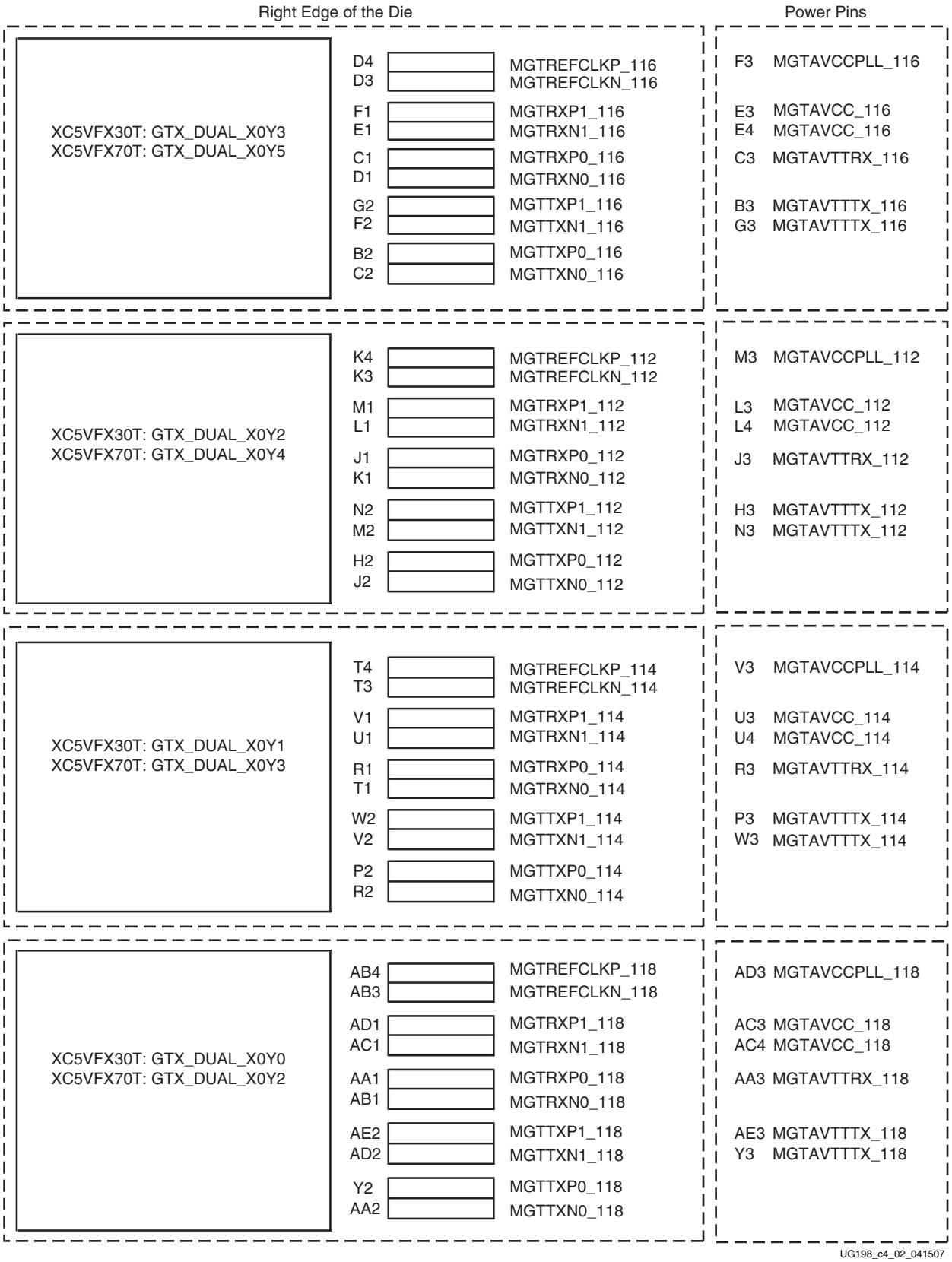


Figure 4-2: **XC5VFX30T-FF665, XC5VFX70T-FF665 GTX Placement**

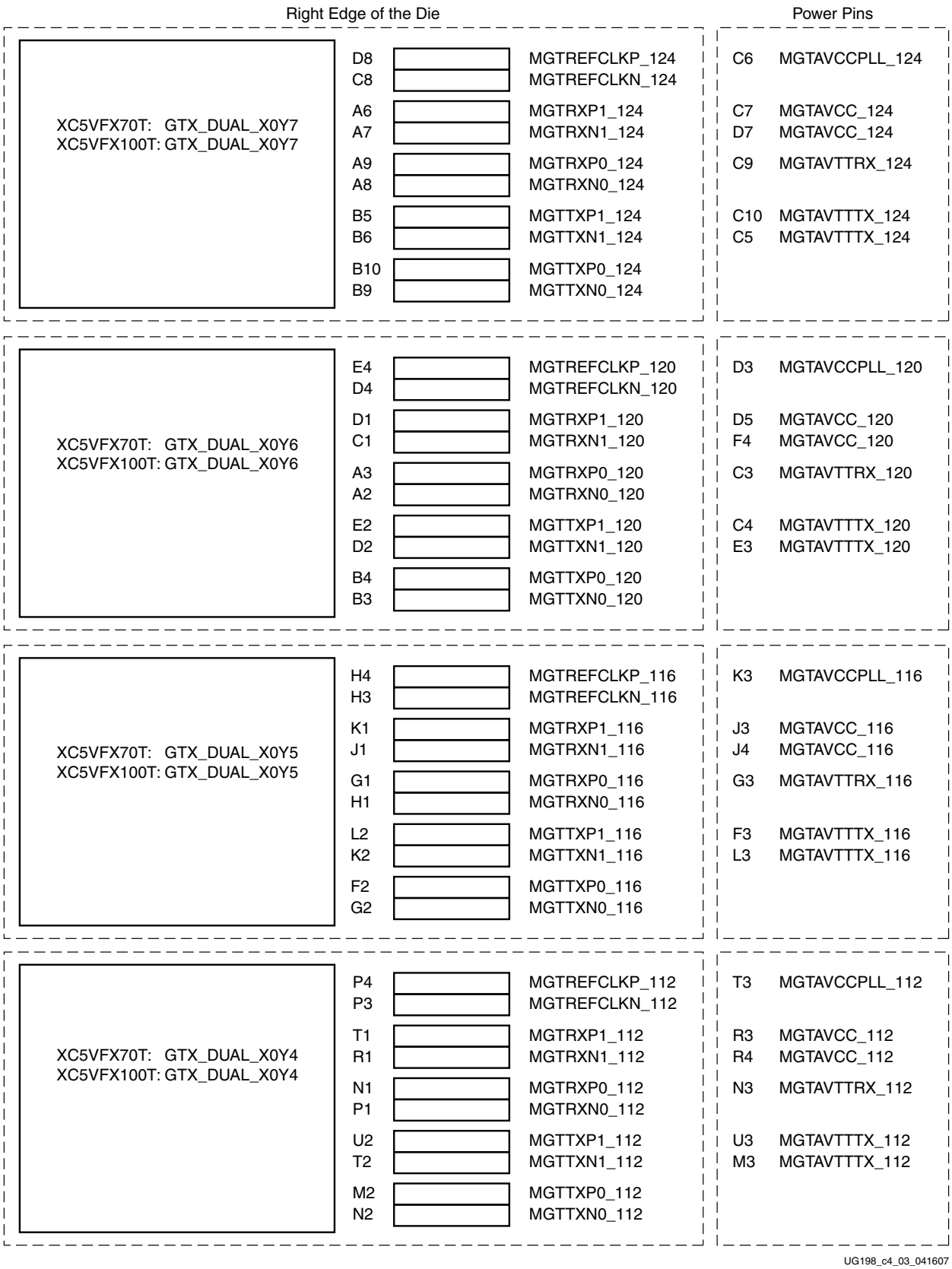


Figure 4-3: **XC5VFX70T-FF1136, XC5VFX100T-FF1136 GTX Placement (1 of 2)**

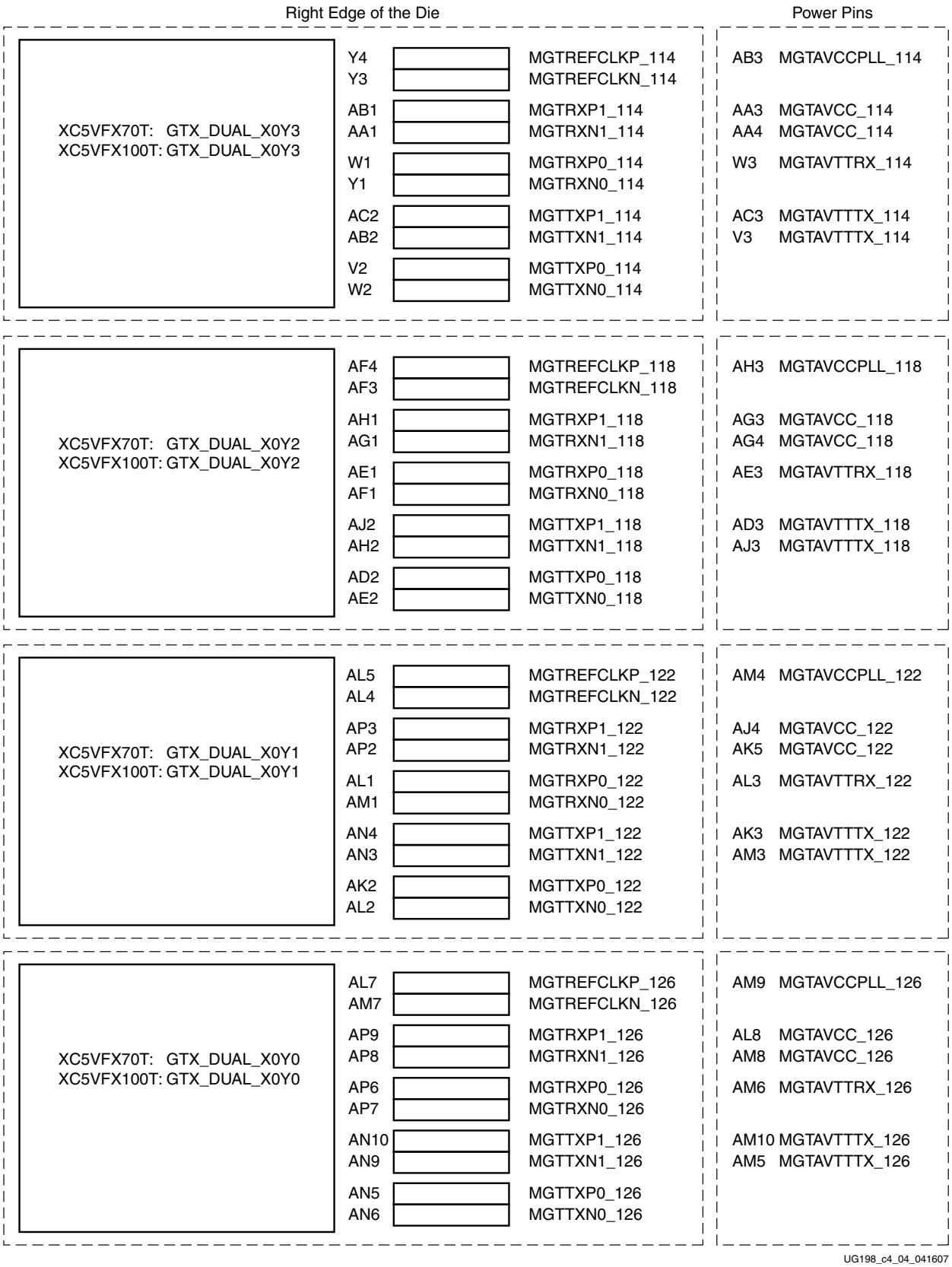


Figure 4-4: **XC5VFX70T-FF1136, XC5VFX100T-FF1136 GTX Placement (2 of 2)**

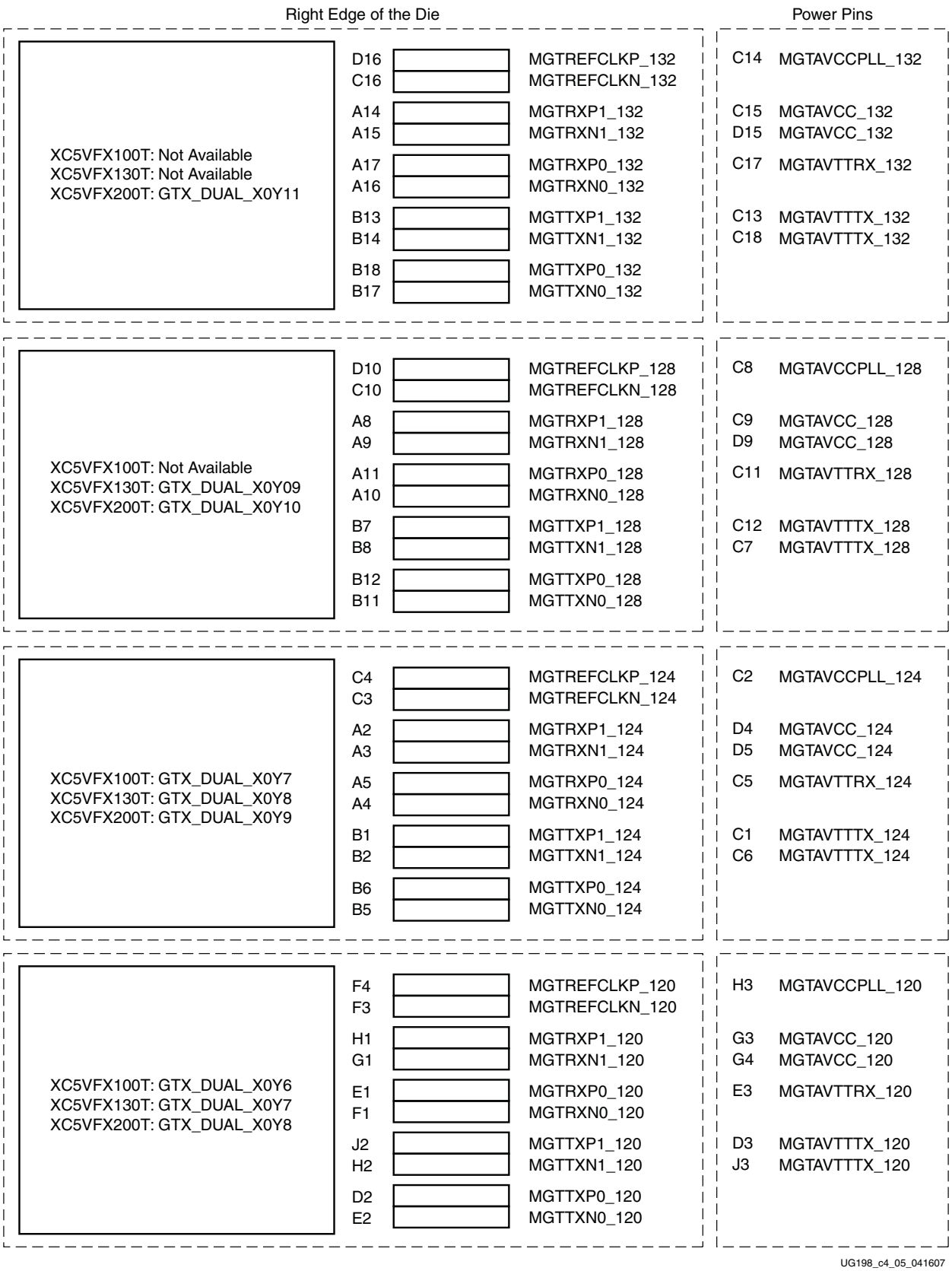


Figure 4-5: **XC5VFX100T-FF1738, XC5VFX130T-FF1738, and XC5VFX200T-FF1738 GTX Placement (1 of 3)**

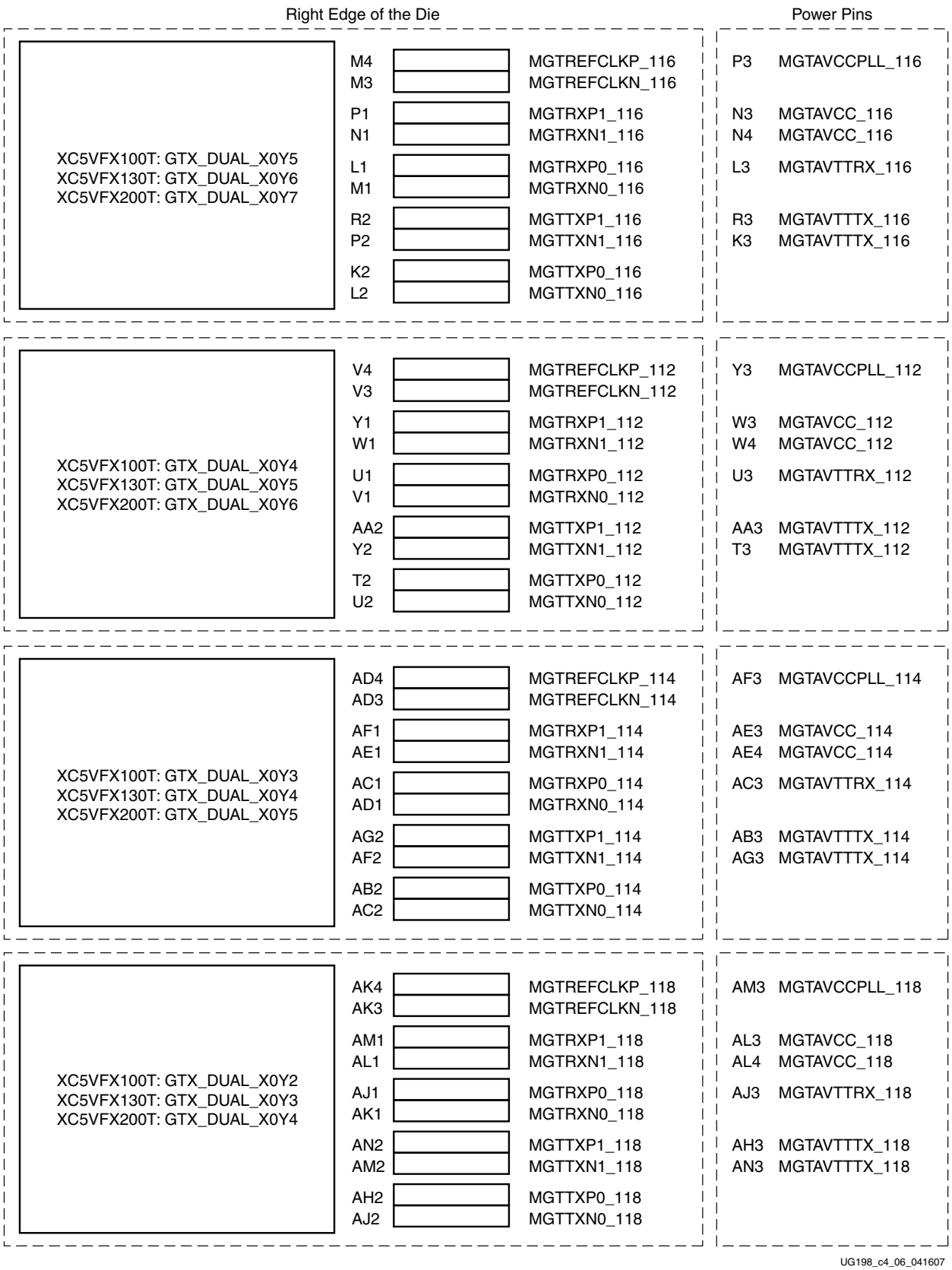


Figure 4-6: **XC5VFX100T-FF1738, XC5VFX130T-FF1738, and XC5VFX200T-FF1738 GTX Placement (2 of 3)**

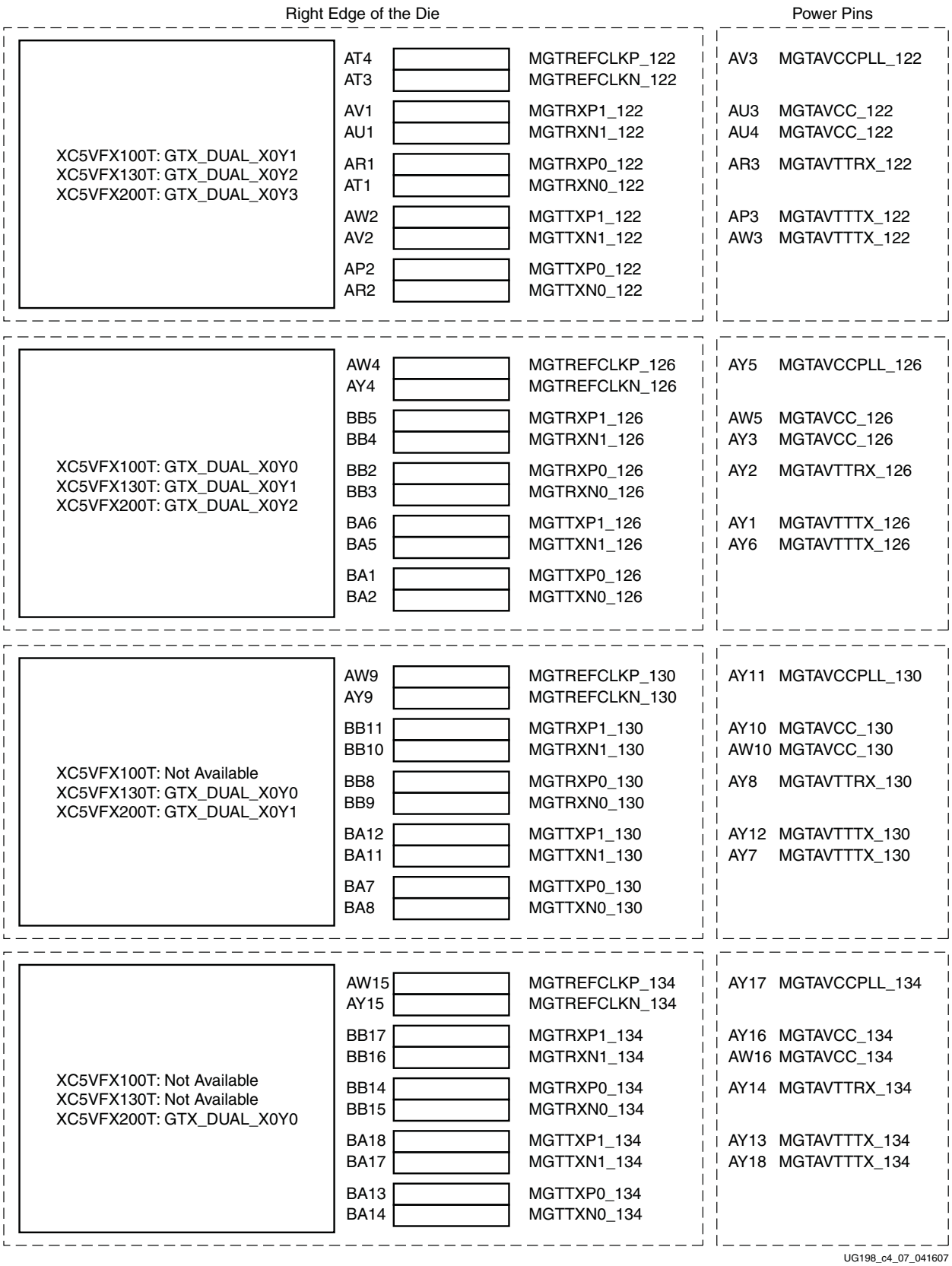


Figure 4-7: **XC5VFX100T-FF1738, XC5VFX130T-FF1738, and XC5VFX200T-FF1738 GTX Placement (3 of 3)**

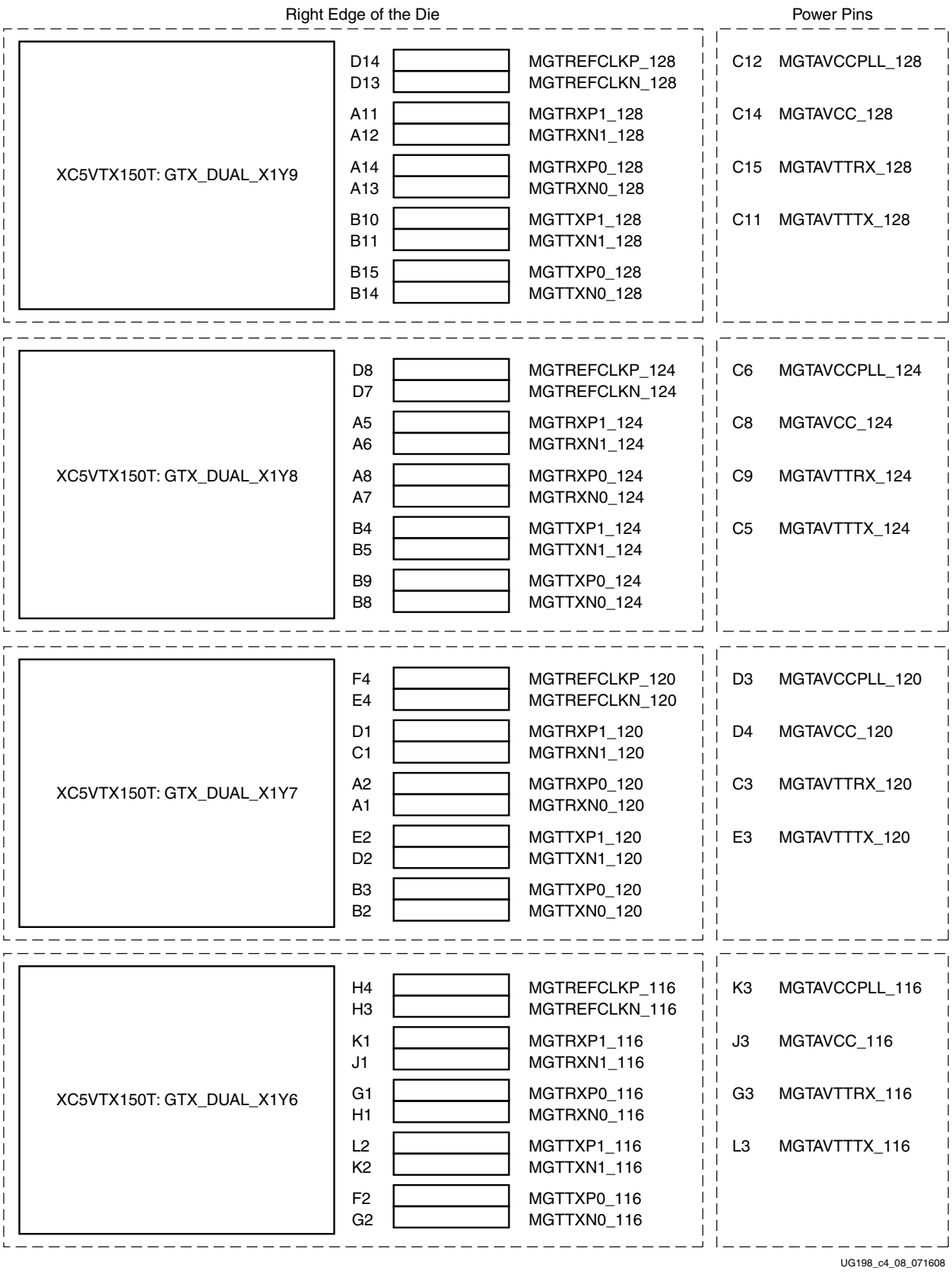


Figure 4-8: **XC5VTX150T-FF1156 GTX Placement (1 of 6)**

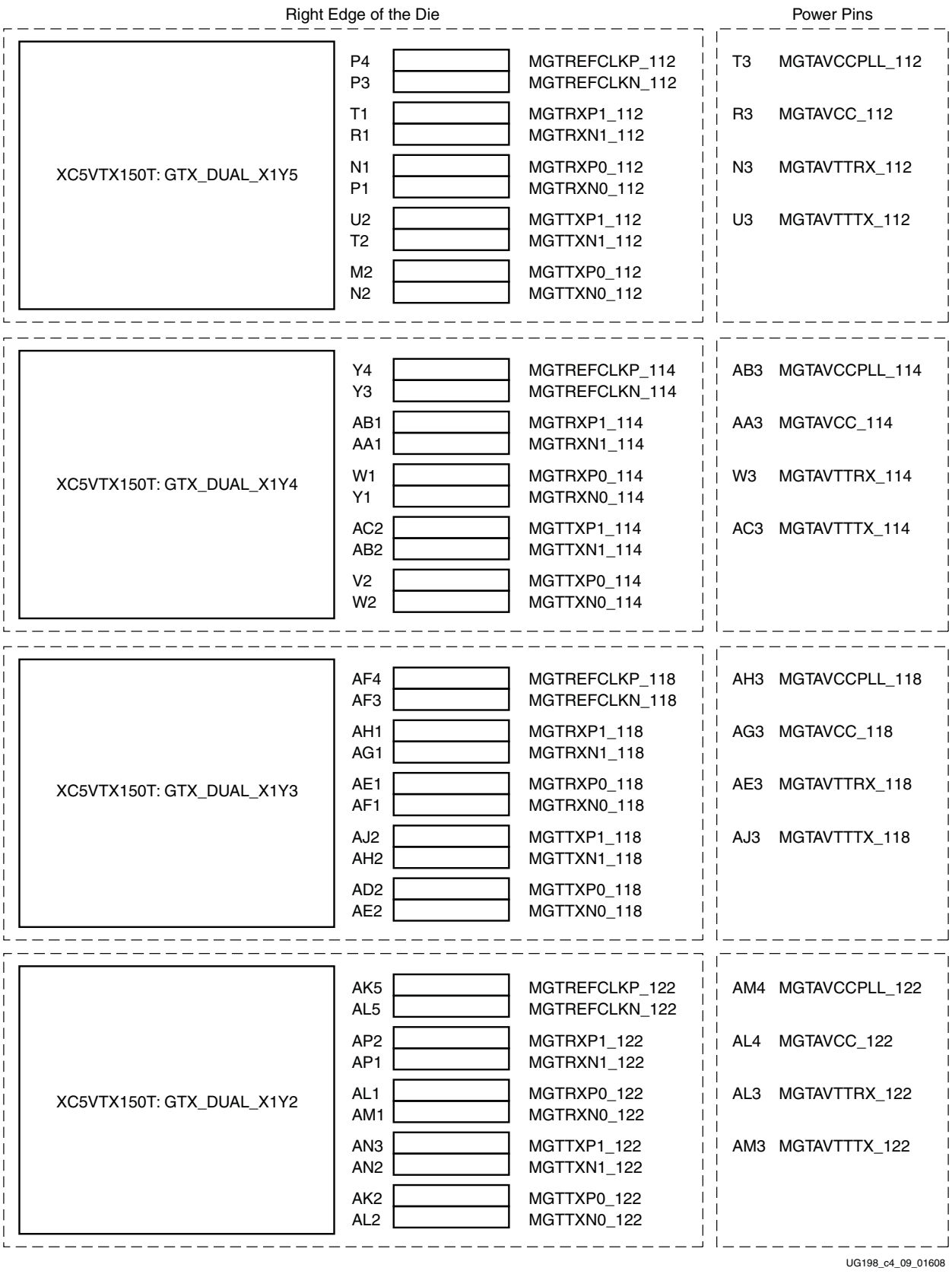


Figure 4-9: **XC5VTX150T-FF1156 GTX Placement (2 of 6)**

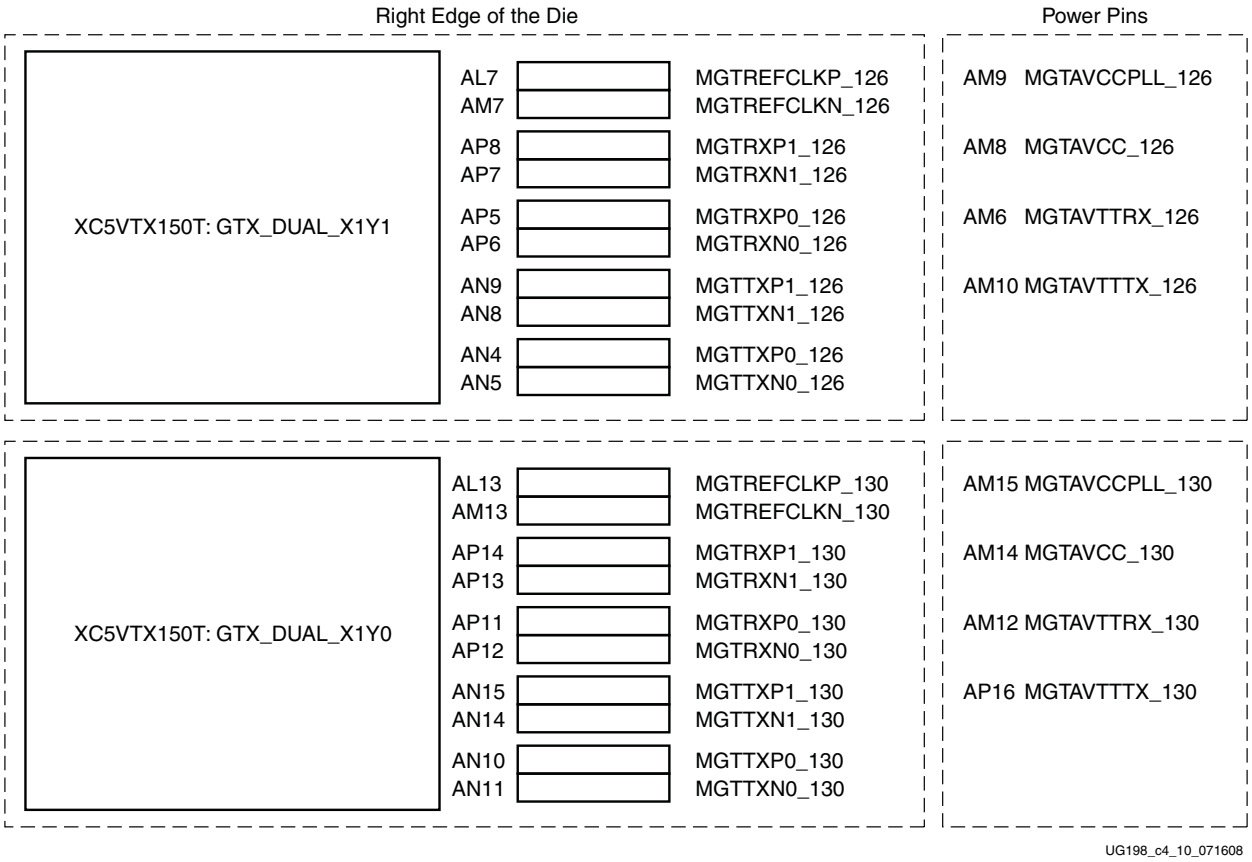


Figure 4-10: **XC5VTX150T-FF1156 GTX Placement (3 of 6)**

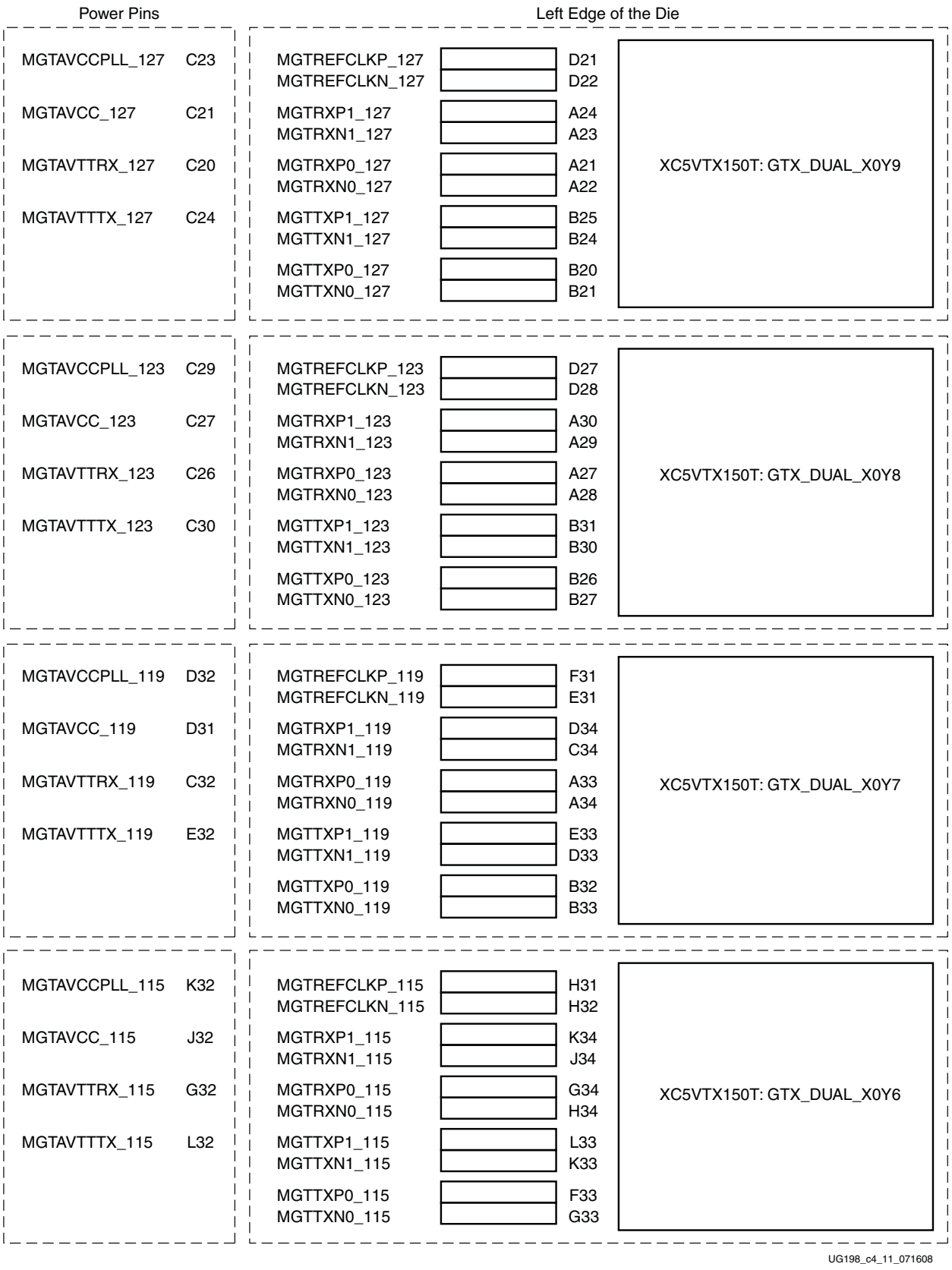


Figure 4-11: **XC5VTX150T-FF1156 GTX Placement (4 of 6)**

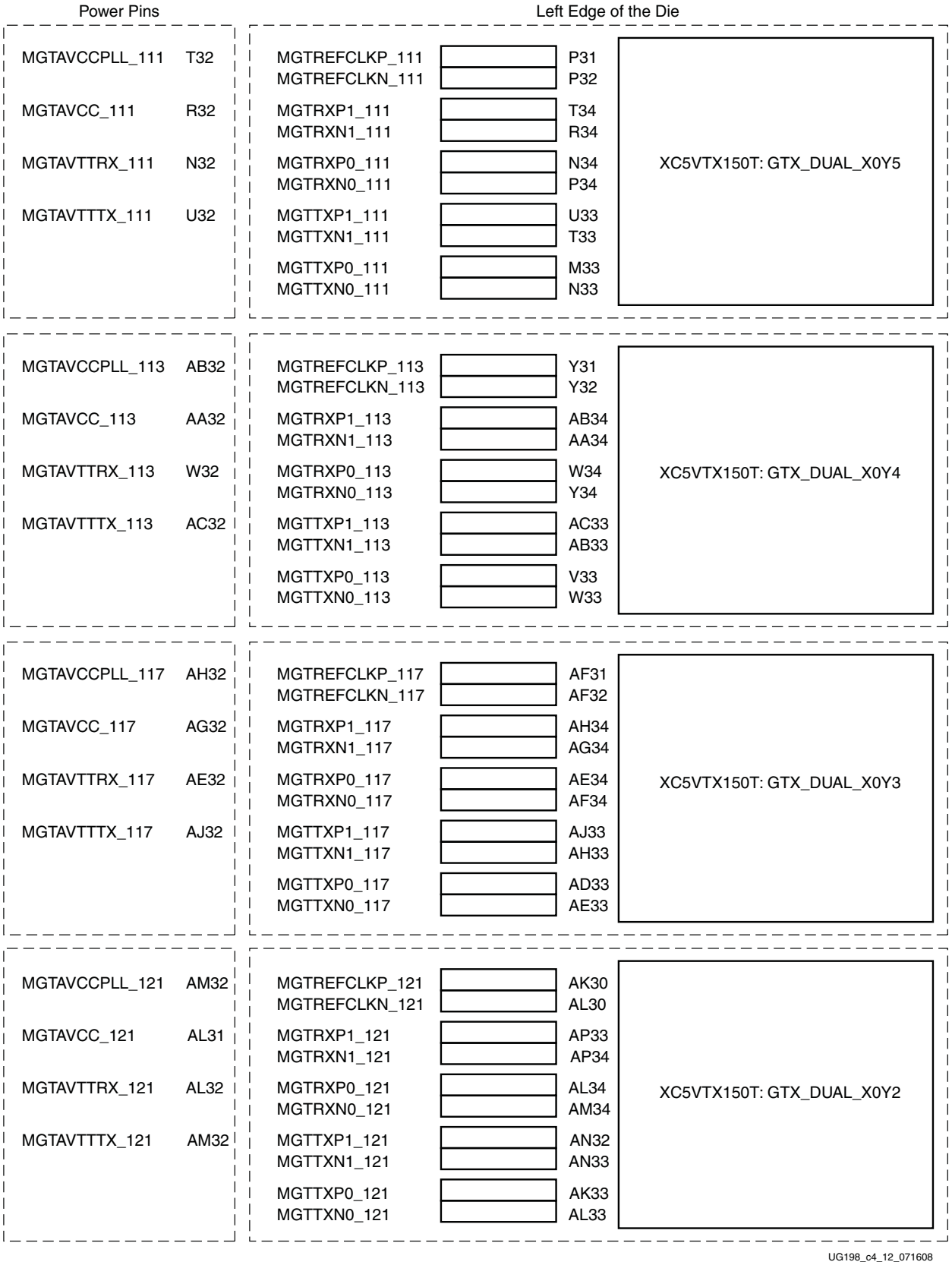


Figure 4-12: **XC5VTX150T-FF1156 GTX Placement (5 of 6)**

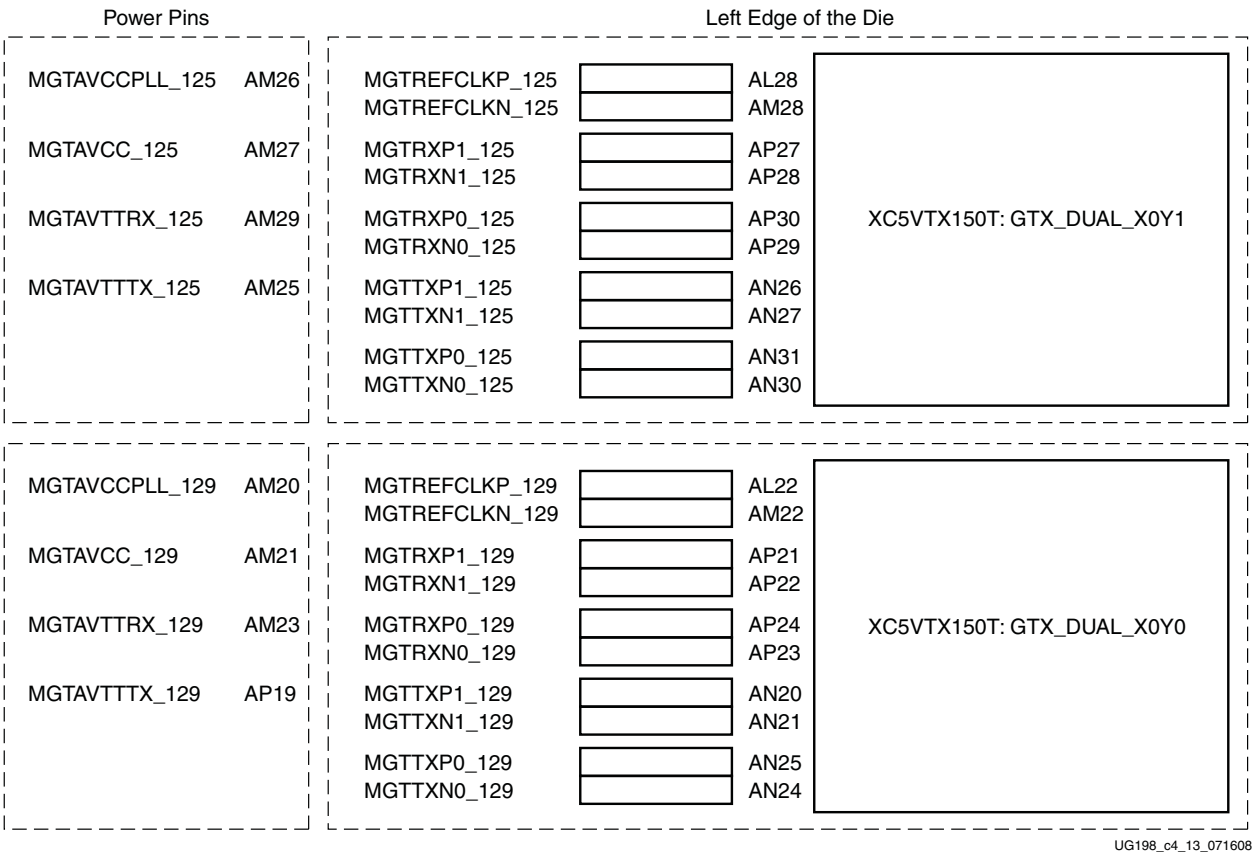


Figure 4-13: **XC5VTX150T-FF1156 GTX Placement (6 of 6)**

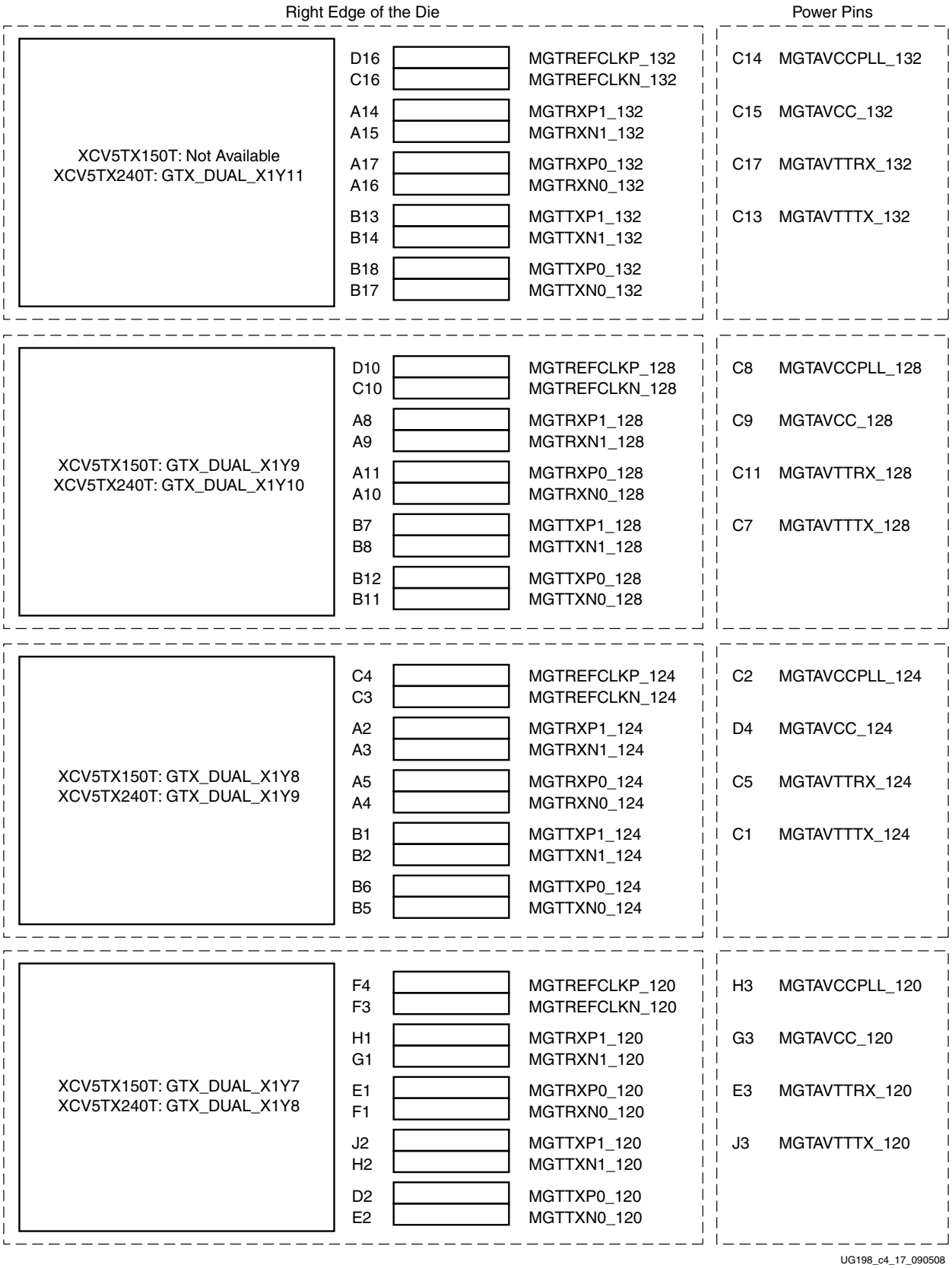


Figure 4-14: **XC5VTX150T-FF1759 and XC5VTX240T-FF1759 GTX Placement (1 of 6)**

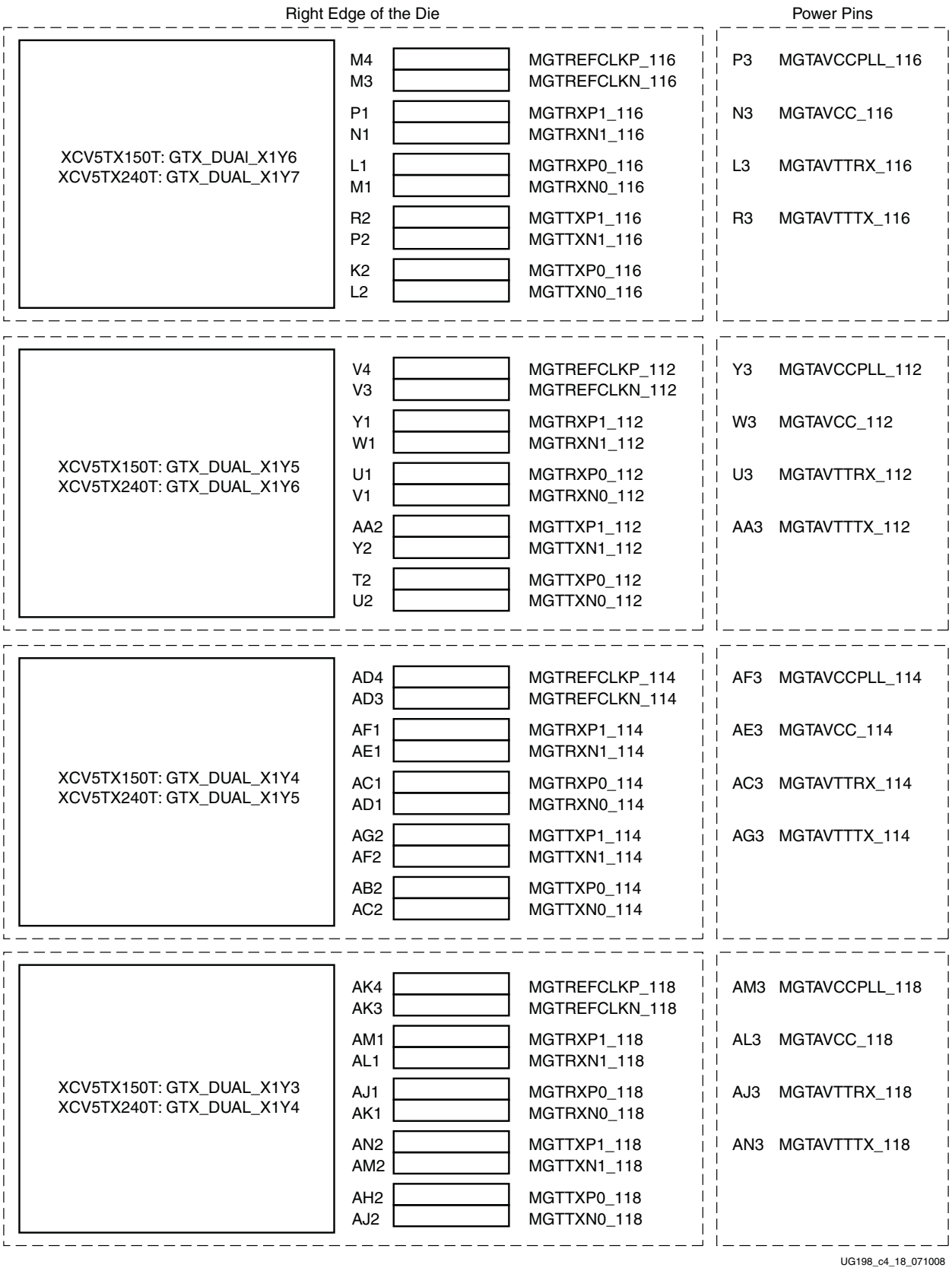


Figure 4-15: **XC5VTX150T-FF1759 and XC5VTX240T-FF1759 GTX Placement (2 of 6)**

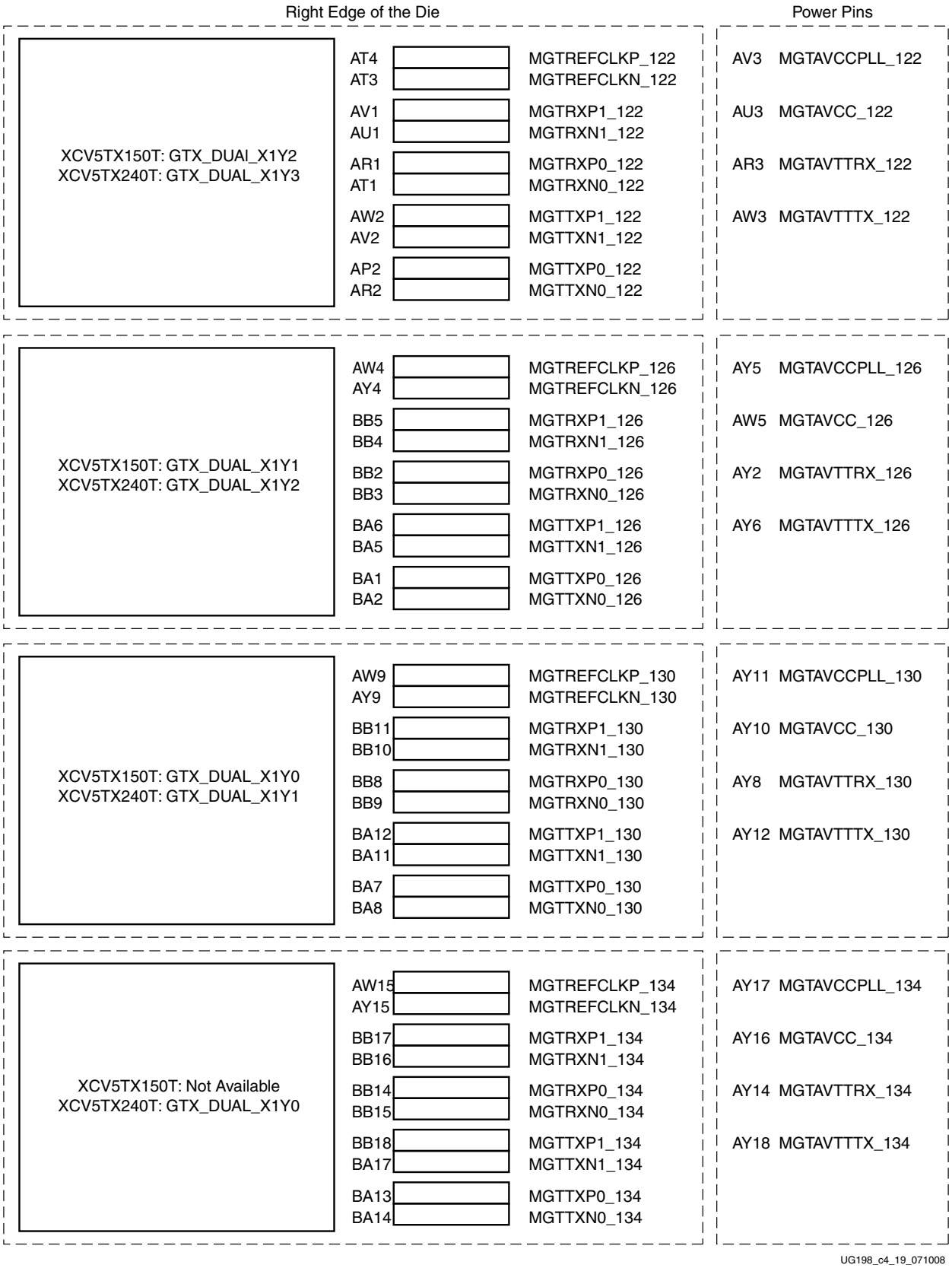


Figure 4-16: **XC5VTX150T-FF1759 and XC5VTX240T-FF1759 GTX Placement (3 of 6)**

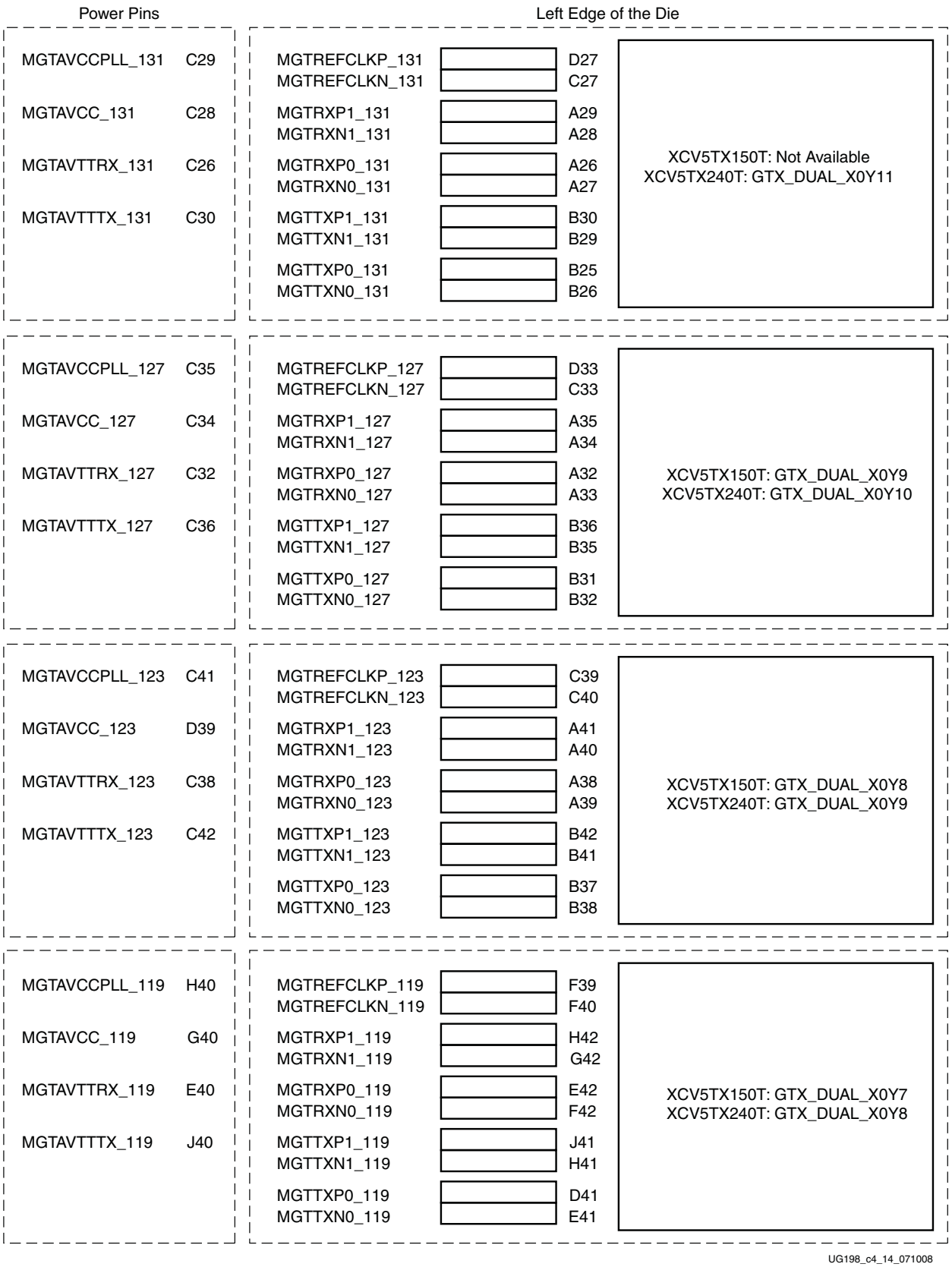


Figure 4-17: **XC5VTX150T-FF1759 and XC5VTX240T-FF1759 GTX Placement (4 of 6)**

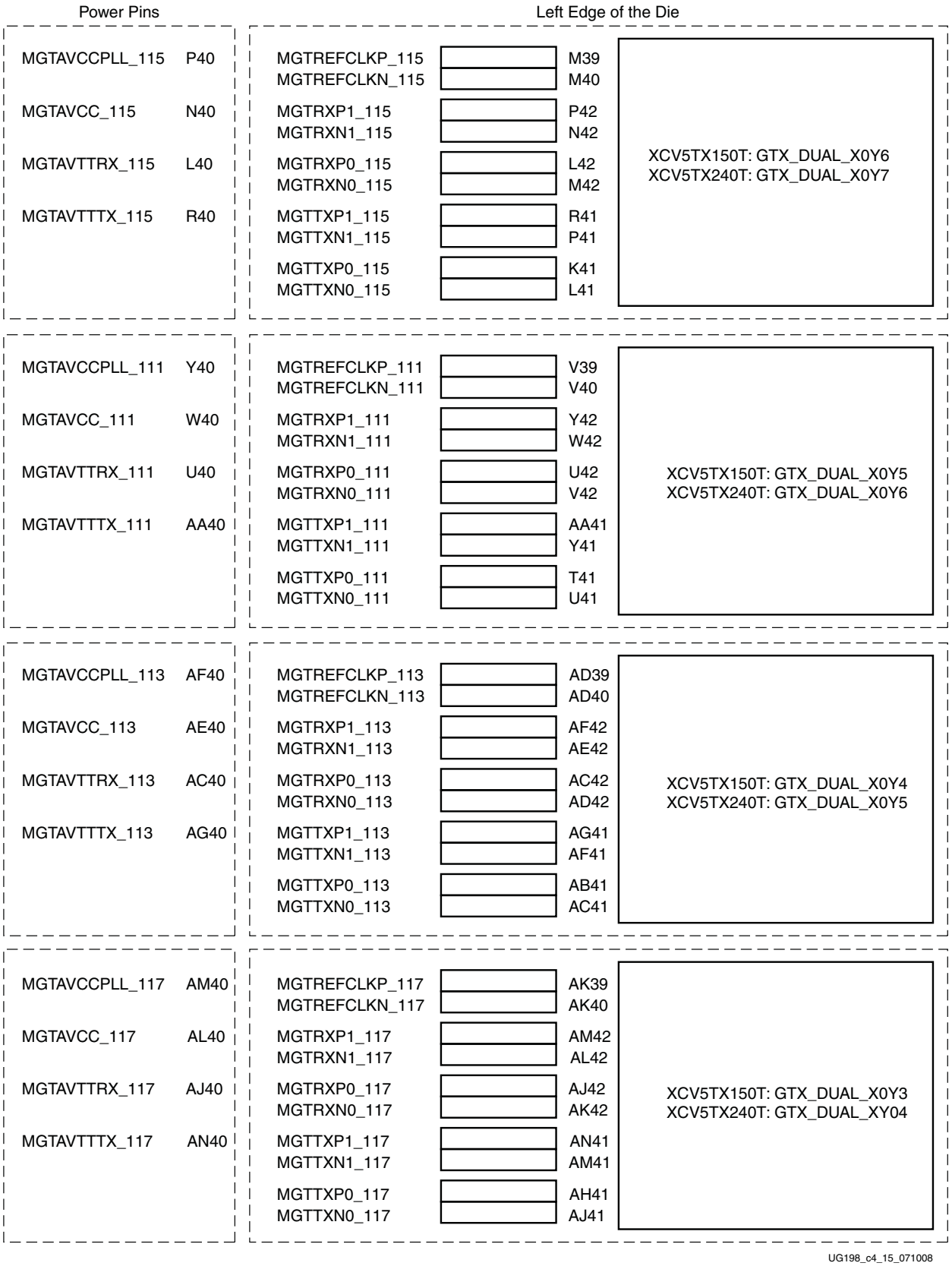


Figure 4-18: **XC5VTX150T-FF1759 and XC5VTX240T-FF1759 GTX Placement (5 of 6)**

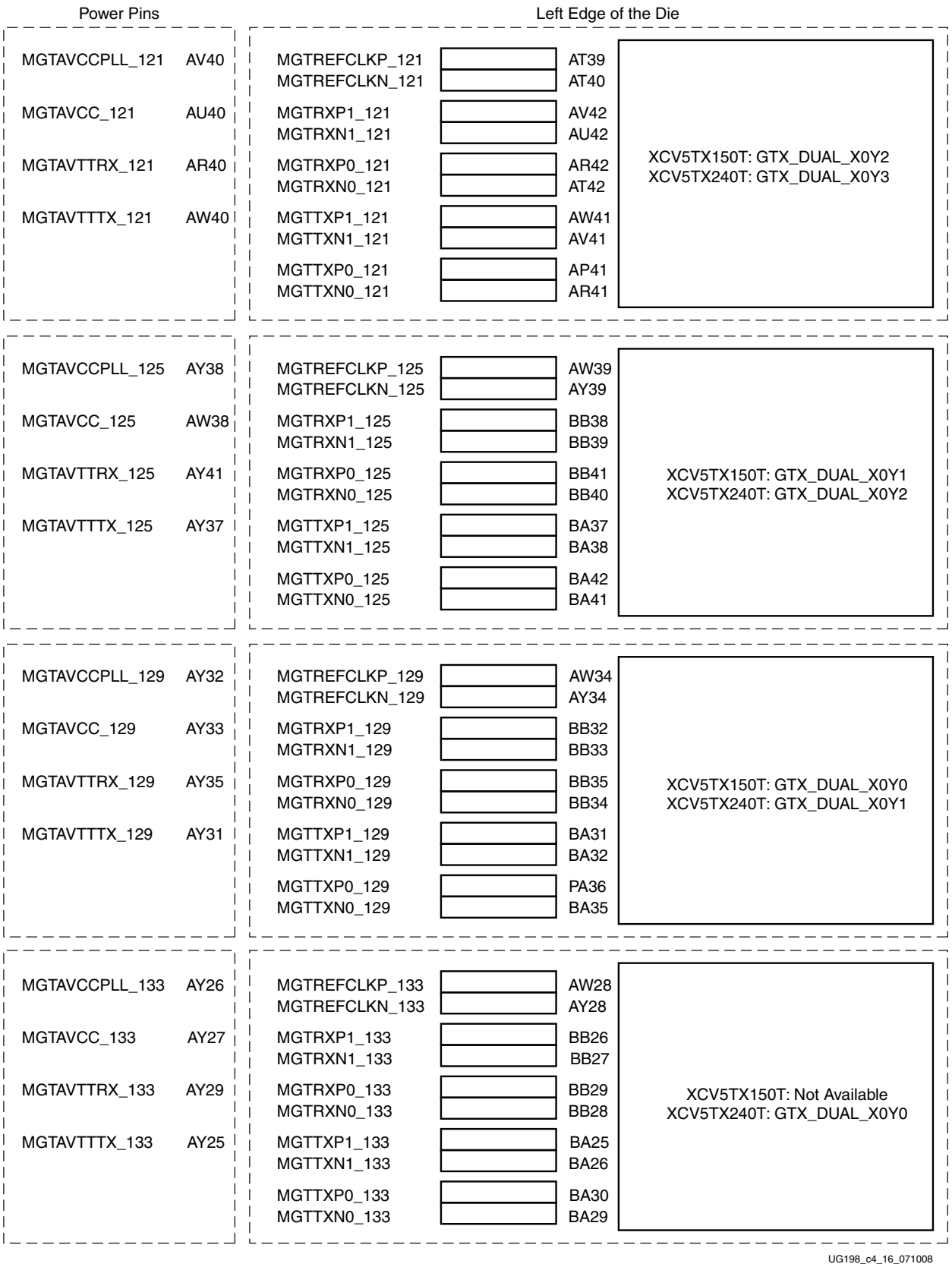


Figure 4-19: **XC5VTX150T-FF1759 and XC5VTX240T-FF1759 GTX Placement (6 of 6)**

Chapter 5

Tile Features

Tile Features Overview

To minimize power consumption and area, many important GTX functions are shared between two transceivers. These functions include the generation of a high-speed serial clock, resets, power control, and dynamic reconfiguration.

Correct clocking and reset behavior is critical for any GTX transceiver design. This chapter describes the following steps that must be performed when configuring the GTX_DUAL tile:

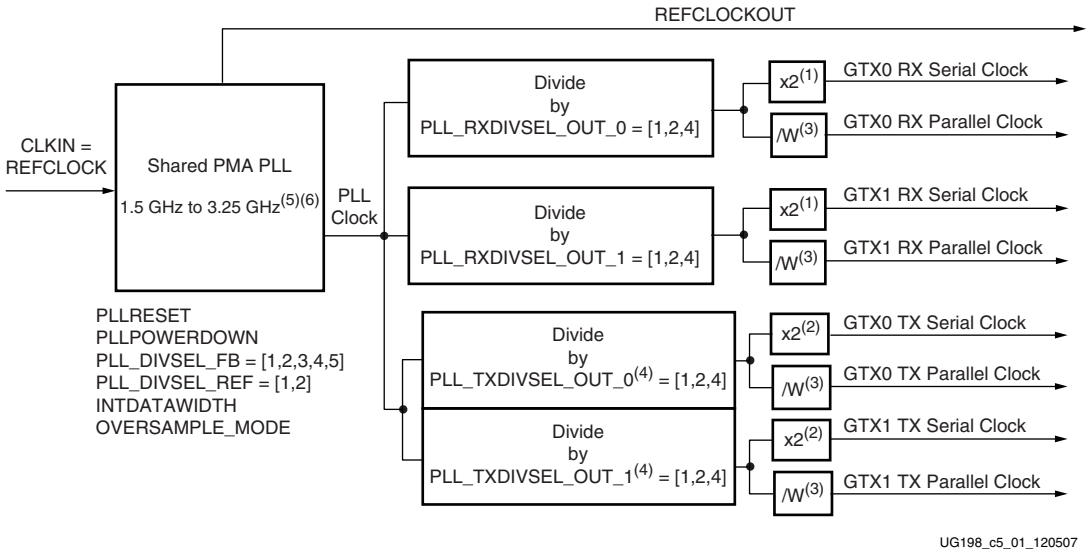
- Set the shared PMA PLL rate
- Set the reference clock source
- Set the Reset options for specific protocols

These steps are performed automatically when the Wizard is used to configure the GTX_DUAL tile.

Shared PMA PLL

Overview

This section describes the shared PMA PLL of the GTX_DUAL tile (Figure 5-1). Each GTX_DUAL tile includes one shared PMA PLL used to generate a high-speed serial clock from a high-quality reference clock (CLKIN). The high-speed clock from this block drives the TX and RX PMA blocks for both GTX transceivers in the tile.



Notes:

1. The Serial In Parallel Out (SIPO) block in each receiver uses both edges of the high-speed clock. As a result, the effective RX serial clock rate is 2 x PLL Clock/PLL_RXDIVSEL_OUT_n.
2. The Parallel In Serial Out (PISO) block in each transmitter uses both edges of the high-speed clock. As a result, the effective TX serial clock rate is 2 x PLL Clock/PLL_TXDIVSEL_OUT_n.
3. The parallel clock rate is divided to match the internal datapath width. When INTDATAWIDTH = 0 (16-bit internal width), W = 8. When INTDATAWIDTH = 1 (20-bit internal width), W = 10.
4. Refer to Chapter 9, "Loopback," about the correct setting of these attributes for specific loopback modes.
5. Nominal operating range.
6. The nominal operating range of the shared PMA PLL in a GTX_DUAL tile is 1.5 GHz to 3.25 GHz. The nominal operation range of the shared PMA PLL in a GTP_DUAL tile is 1.0 GHz to 2.0 GHz.

Figure 5-1: **Shared PMA PLL Detail**

The shared PMA PLL generates the high-speed clock (PLL clock) used by both transceivers in the GTX_DUAL tile. After the shared PMA PLL rate is set (PLL clock), the TX and RX output dividers (dividers ending with _OUT) are set to determine the TX and RX line rates for each transceiver.

Ports and Attributes

Table 5-1 defines the shared PMA PLL ports.

Table 5-1: **Shared PMA PLL Ports**

Port	Dir	Domain	Description
CLKIN	In	Async	Reference clock input to the shared PMA PLL. See "Clocking," page 93 for more information about the different ways this port can be driven.
INTDATAWIDTH	In	Async	Sets the internal datapath width for the GTX_DUAL tile. If Low, the internal datapath width is set to 16 bits. If High, the internal datapath width is set to 20 bits.
PLLLKDET	Out	Async	This port indicates that the VCO rate is within acceptable tolerances of the desired rate when High. Neither GTX transceiver in the tile operates reliably until this condition is met.
PLLLKDETEN	In	Async	This port enables the PLL lock detector and must always be tied High.
REFCLKOUT	Out	Async	The REFCLKOUT port from each GTX_DUAL tile provides direct access to the reference clock provided to the shared PMA PLL (CLKIN). It can be routed for use in the FPGA logic.

Table 5-2 defines the shared PMA PLL attributes.

Table 5-2: **Shared PMA PLL Attributes**

Attribute	Type	Description
OVERSAMPLE_MODE	Boolean	When OVERSAMPLE_MODE is TRUE, then DIV = 5 regardless of the setting of INTDATAWIDTH. When OVERSAMPLE_MODE is FALSE, and INTDATAWIDTH is Low, then DIV = 4 or INTDATAWIDTH is High, then DIV = 5
PLL_DIVSEL_FB	Integer	Controls the feedback divider. Valid settings for PLL_DIVSEL_FB are 1, 2, 3, 4, and 5. PLL_DIVSEL_FB is multiplied by 4 or 5, depending on the width of the internal datapath as set by INTDATAWIDTH. If INTDATAWIDTH is Low, the feedback divider N is set to PLL_DIVSEL_FB x 4. If INTDATAWIDTH is High, the feedback divider N is set to PLL_DIVSEL_FB x 5.
PLL_DIVSEL_REF	Integer	Controls the reference clock divider. Valid settings for PLL_DIVSEL_REF are 1 and 2.
PLL_RXDIVSEL_OUT_0 PLL_RXDIVSEL_OUT_1	Integer	Divides the PLL clock to produce a high-speed RX clock. Because both edges of the clock are used, the divided clock must run at one-half the desired RX line rate. Permitted divider settings are 1, 2, and 4. See "Serial In to Parallel Out," page 181 for details.
PLL_TXDIVSEL_OUT_0 PLL_TXDIVSEL_OUT_1	Integer	Divides the PLL clock to produce a high-speed TX clock. Because both edges of the clock are used, the divided clock must run at one-half the desired TX line rate. Divider settings are 1, 2, and 4. Each GTX transceiver has its own PLL_TXDIVSEL_OUT. See "Parallel In to Serial Out," page 146 for details.

Description

The first step to using the GTX_DUAL tile is to set the output of the shared PMA PLL (PLL clock) to a rate that can be used by each GTX transceiver to generate an appropriate serial line rate and the corresponding parallel clock rate. Both GTX TX and RX blocks are equipped with an independent divider that can divide the PLL clock by a factor of 1, 2, or 4. This divider allows the TX and RX blocks of each GTX transceiver to run at different rates, related by an integer multiple.

The PLL clock rate must be set to one-half the required line rate after the independent dividers. For example, if an RX line rate of 5 Gb/s is desired in GTX0, and the independent divider in the GTX RX block is set to 1, the PLL clock must be set to 2.5 GHz.

The shared PMA PLL has a nominal operation range between 1.5 GHz and 3.25 GHz. The PLL clock must be set within this range. Equation 5-1 shows how to set the PLL clock based on CLKIN (the reference clock), PLL_DIVSEL_FB, PLL_DIVSEL_REF, and INTDATAWIDTH. PLL_DIVSEL_FB and PLL_DIVSEL_REF control dividers inside the PLL. INTDATAWIDTH controls the internal parallel data width of the entire GTX_DUAL tile.

Equation 5-1 shows how to set the rate of the PLL clock.

$$f_{PLL\ Clock} = f_{CLKIN} \times \frac{PLL\_DIVSEL\_FB \times DIV}{PLL\_DIVSEL\_REF} \qquad \textit{Equation 5-1}$$

The following conditions apply to Equation 5-1:

- PLL_DIVSEL_REF = {1, 2}.
- PLL_DIVSEL_FB = {1, 2, 3, 4, 5}.
- DIV = 5, when:
  - OVERSAMPLEMODE = TRUE
  - OVERSAMPLEMODE = FALSE AND INTDATAWIDTH is HIGH
- DIV = 4, when:
  - OVERSAMPLEMODE = FALSE AND INTDATAWIDTH is LOW
- The PLL clock must be operated in its nominal frequency range.
- The smallest possible divider values must be selected.

The programmable dividers allow support for various standards.

Figure 5-2 shows a conceptual view of the shared PMA PLL from which the PLL clock is generated.

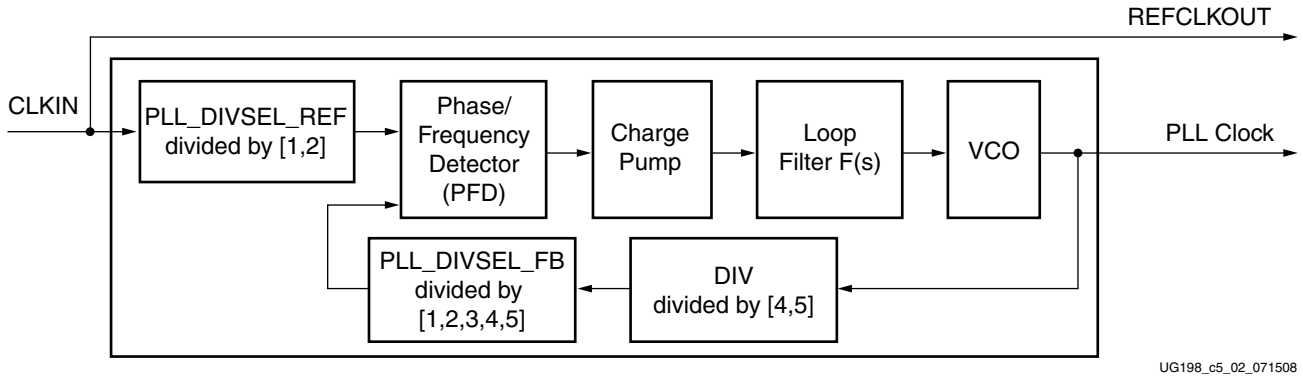


Figure 5-2: **Shared PMA PLL Conceptual View**

Table 5-3 shows example settings for several standard protocols.

Table 5-3: **Communication Standards**

Standard	Line Rate [Gb/s]	TX/RX USRCLK Frequency [MHz]	TX/RX USRCLK2 Frequency [MHz]			Reference Clock Frequency REFCLK [MHz]	PLL Clock Frequency [GHz]	Reference Clock Divider Setting PLL_DIVSEL_REF ε{1,2}	Feedback Loop Divider Setting PLL_DIVSEL_FB ε{1,2,3,4,5}	Divider Settings: PLL_RXDIVSEL_OUT_(0/1)(1) PLL_TXDIVSEL_OUT_(0/1)(1) ε{1,2,4}
			1-byte Logic Interface(2,10)	2-byte Logic Interface(3)	4-byte Logic Interface(4)					
INTDATAWIDTH = 1 (20-bit Internal Datapath) →(DIV = 5)										
FC4	4.25	212.5	NA	212.5	106.25	212.5	2.125	1	2	1
FC2	2.125	106.25	212.5	106.25	53.125	212.5	2.125	1	2	2
FC1	1.0625	53.125	106.25	53.125	26.5625	212.5	2.125	1	2	4
XAUI	3.125	156.25	312.5	156.25	78.125	312.5	1.5625	1	1	1
						156.25			2	
10GBASE-CX4	3.125	156.25	312.5	156.25	78.125	312.5	1.5625	1	1	1
GigE	1.25	62.5	125	62.5	31.25	125	2.5	1	4	4
Aurora	6.25	312.5	NA	312.5	156.25	156.25	3.125	1	4	1
	3.125	156.25	312.5	156.25	78.125	156.25	1.5625	1	2	1
	2.5	125	250	125	62.5	250	2.5	1	2	2
	1.25	62.5	125	62.5	31.25	250	2.5	1	2	4
Serial RapidIO	3.125	156.25	312.5	156.25	78.125	156.25	1.5625	1	2	1
	2.5	125	250	125	62.5	125	2.5	1	4	2
	1.25	62.5	125	62.5	31.25	125	2.5	1	4	4
SATA III	6.0	300	NA	300	150	150	3.0	1	4	1
SATA Generation 2	3	150	300	150	75	150	1.5	1	2	1
SATA Generation 1	1.5	75	150	75	37.5	150	1.5	1	2	2
PCI Express, Rev. 2	5.0	250	NA	250	125	250	2.5	1	2	1
PCI Express, Rev. 1.0a, Rev. 1.1	2.5	125	250	125	62.5	100	2.5	1	5	2
Infiniband	2.5	125	250	125	62.5	125	2.5	1	4	1
CPRI(5)	2.4576	122.88	245.76	122.88	61.44	122.88	2.4576	1	4	2
	1.2288	61.44	122.88	61.44	30.72	122.88	2.4576	1	4	4
OBSAI(5)	3.072	153.6	307.2	153.6	76.8	153.6	3.072	1	4	2
	1.536	76.8	153.6	76.8	38.4	153.6	3.072	1	4	4
HD-SDI(7)	1.485	74.25	148.5	74.25	37.125	148.5	2.97	1	4	4
3G-SDI	2.970	148.5	297.0	148.5	74.25	148.5	2.97	1	4	2

Table 5-3: **Communication Standards *(Cont'd)***

Standard	Line Rate [Gb/s]	TX/RX USRCLK Frequency [MHz]	TX/RX USRCLK2 Frequency [MHz]			Reference Clock Frequency REFCLK [MHz]	PLL Clock Frequency [GHz]	Reference Clock Divider Setting PLL_DIVSEL_REF ε{1,2}	Feedback Loop Divider Setting PLL_DIVSEL_FB ε{1,2,3,4,5}	Divider Settings: PLL_RXDIVSEL_OUT_(0/1)(1) PLL_TXDIVSEL_OUT_(0/1)(1) ε{1,2,4}
			1-byte Logic Interface(2,10)	2-byte Logic Interface(3)	4-byte Logic Interface(4)					
INTDATAWIDTH = 0 (16-bit Internal Datapath) →(DIV = 4)										
Interlaken	6.25	390.625	NA(9)	390.625	195.3125	390.625	3.125	1	2	1
	3.125	195.3125	NA(9)	195.3125	97.65625	390.625	3.125	1	2	2
OC48	2.488	155.5	311	155.5	77.75	311.04	2.488.32	1	2	2
SFI-5(8)	3.125	195.3125	390.625	195.3125	97.65625	195.313	3.125	1	4	2
	2.488	155.5	311	155.5	77.75	155.52	2.48832	1	4	2
SPI-5(8)	2.488	155.5	311	155.5	77.75	155.52	2.48832	1	4	2
TFI-5(8)	3.1104	194.4	388.8	194.4	97.2	194.4	3.1104	1	4	2
	2.488	155.5	311	155.5	77.75	155.52	2.48832	1	4	2

Notes:

1. See "Parallel In to Serial Out," page 146 and "Serial In to Parallel Out," page 181 for more details about the divider setting.
2. 1-byte (8/10-bit) user interface. Set RXDATAWIDTH = TXDATAWIDTH = 0, RX/TXUSRCLK2 rate = 2 x RX/TXUSRCLK rate.
3. 2-byte (16/20-bit) user interface. Set RXDATAWIDTH = TXDATAWIDTH = 1, RX/TXUSRCLK2 rate = 1 x RX/TXUSRCLK rate.
4. 4-byte (32/40-bit) user interface. Set RXDATAWIDTH = TXDATAWIDTH = 2, RX/TXUSRCLK2 rate = 0.5 x RX/TXUSRCLK rate.
5. Synchronous system.
6. This data rate is supported using 5x digital oversampling. Set OVERSAMPLE_MODE = TRUE.
7. Other frequency is 0.1% lower.
8. Maximum data rate.
9. When using the internal Gearbox, a 1-byte interface is not supported. A 1-byte interface can be implemented by using an external Gearbox in the FPGA logic.
10. A 1-byte interface is NOT supported with data rates of 4.25 Gb/s or higher.

See "Clocking," page 93 for details on supplying CLKIN to the shared PMA PLL.

Depending on the width of the RX and TX interface to the FPGA logic resources, the following fixed ratios exist between RXUSRCLK and RXUSRCLK2 frequencies and between TXUSRCLK and TXUSRCLK2 frequencies:

1. 1-byte RX/TX data interface to the fabric: TX/RXUSRCLK2 = 2 x TX/RXUSRCLK
2. 2-byte RX/TX data interface to the fabric: TX/RXUSRCLK2 = TX/RXUSRCLK
3. 4-byte RX/TX data interface to the fabric: TX/RXUSRCLK2 = 0.5 x TX/RXUSRCLK

"FPGA TX Interface" in Chapter 6 contains details on generating TXUSRCLK and TXUSRCLK2.

"FPGA RX Interface" in Chapter 7 contains details on generating RXUSRCLK and RXUSRCLK2.

Examples

Configuring the Shared PMA PLL for XAUI Operation

The three methods to configure the shared PMA PLL for XAUI are described below:

1. Use the RocketIO GTX Transceiver Wizard.

   The wizard includes a protocol file for XAUI that allows it to automatically configure the GTX_DUAL primitive for use in a XAUI design.

2. Use the settings from Table 5-3.

   Table 5-3 includes the settings for common configurations of popular protocols. XAUI settings are included in Table 5-3, along with other protocols that use 8B/10B encoding.

3. Use Equation 5-1 as described in the following steps:

   a. Determine the required line rates.

      For XAUI operation, both TX and RX use a line rate of 3.125 Gb/s.

   b. Determine the internal datapath width.

      Because XAUI is an 8B/10B-encoded standard, an internal datapath width of 20 bits is required. See "Configurable 8B/10B Encoder," page 125 and "Configurable 8B/10B Decoder," page 198 for more information about encoding and internal datapath width requirements.

   c. Determine the desired reference clock rate.

      This example uses a reference clock running at 156.25 MHz, a common rate for XAUI.

   d. Calculate the required PLL clock rate.

      Because the SIPO block utilizes a Double Date Rate (DDR) architecture where both edges of the clock are used to deserialize data, it must be fed a clock from the PLL that is one-half the magnitude of the desired line rate. In this XAUI example, the calculation is 3.125 Gb/s line rate divided by 2 equaling 1.5625 GHz. Because this RX rate of 1.5625 GHz is within the nominal PLL operation range, the external divider (PLL_RXDIVSEL_OUT) must be one to allow the PLL to run at 1.5625 GHz. The PLL clock rate is thus 1.5625 x 1 = 1.5625 GHz.

   e. Calculate the required DIV value.

      Because the internal datapath width must be 20 bits and INTDATAWIDTH is High, DIV = 5.

   f. Calculate the required PLL divider ratio.

      Using the values f_{CLKIN}, DIV, and f_{PLL_CLOCK} determined above, rearrange Equation 5-1 to calculate the divider ratio as shown in Equation 5-2. The result is a ratio of two.

$$\frac{PLL\_DIVSEL\_FB}{PLL\_DIVSEL\_REF} = \frac{f_{PLL\_Clock}}{f_{CLKIN} \times DIV} = \frac{1.5625\ GHz}{156.25\ MHz \times 5} = 2 \qquad \textit{Equation 5-2}$$

   g. Select the PLL divider values.

      Select the smallest divider values that result in the required PLL divider ratio. In this case, using PLL_DIVSEL_FB = 2 and PLL_DIVSEL_REF = 1 results in a ratio of 2.

Configuring the Shared PMA PLL for OC-48 Operation

This example shows how to set the shared PMA PLL divider settings for OC-48 using Equation 5-1. The RocketIO GTX Transceiver Wizard and Table 5-3 are simpler alternatives. This example is provided only to illustrate the process with Equation 5-1.

Use Equation 5-1 as described in the following steps:

1. Determine the required line rates.

   For OC-48, both TX and RX use a line rate of 2.488 Gb/s.

2. Determine the internal datapath width.

   Because OC-48 uses no encoding and a datapath that is a multiple of 8 bits, an internal datapath width of 16 bits is required.

3. Determine the desired reference clock rate.

   This example uses a reference clock running at 155.5 MHz.

4. Calculate the required PLL clock rate.

   The SIPO block utilizes a DDR architecture where both edges of the clock are used to deserialize data. The SIPO block must be fed a clock from the PLL that is one-half the magnitude of the desired line rate. In this OC-48 example, a 2.488 Gb/s line rate is divided by 2, or 1.244 GHz. Because this RX rate of 1.244 GHz is below the nominal PLL operation range, the external divider (PLL_RXDIVSEL_OUT) must be two to allow the PLL to run two times as fast (2.488 GHz). The PLL clock rate is thus 1.244 x 2 = 2.488 GHz.

5. Calculate the required DIV value.

   Because the internal datapath width must be 16 bits and INTDATAWIDTH is Low, DIV = 4.

6. Calculate the required PLL divider ratio.

   Using the values f_{CLKIN}, DIV, and f_{PLL_CLOCK} determined above, rearrange Equation 5-1 to calculate the divider ratio as shown in Equation 5-3. The result is a ratio of 2.

$$\frac{PLL\_DIVSEL\_FB}{PLL\_DIVSEL\_REF} = \frac{f_{PLL\_Clock}}{f_{CLKIN} \times DIV} = \frac{2.488\,GHz}{155.5\ MHz \times 4} = 2 \qquad \textit{Equation 5-3}$$

7. Select the PLL divider values.

   Select the smallest divider values that result in the required PLL divider ratio. In this case, using PLL_DIVSEL_FB = 2 and PLL_DIVSEL_REF = 1 results in a ratio of 2.

Configuring the Shared PMA PLL for Gigabit Ethernet Operation

This example shows how to set the shared PMA PLL divider settings for Gigabit Ethernet using Equation 5-1. The RocketIO GTX Transceiver Wizard and Table 5-3 are simpler alternatives. This example is provided only to illustrate the process with Equation 5-1.

Use Equation 5-1 as described in the following steps:

1. Determine the required line rates.

   For Gigabit Ethernet operation, both TX and RX use a line rate of 1.25 Gb/s.

2. Determine the internal datapath width.

   Because Gigabit Ethernet uses 8B/10B encoding, an internal datapath width of 20 bits is required.

3. Determine the desired reference clock rate.

   This example uses a reference clock running at 125 MHz.

4. Calculate the required PLL clock rate.

   The SIPO block utilizes a DDR architecture where both edges of the clock are used to deserialize data. The SIPO block must be fed a clock from the PLL that is one-half the magnitude of the desired line rate. In this Gigabit Ethernet example, a 1.25 Gb/s line rate is divided by 2, or 0.625 GHz. Because this RX rate of 0.625 GHz is below the nominal PLL operation range, the external divider (PLL_RXDIVSEL_OUT) must be four to allow the PLL to run four times as fast (2.5 GHz). The PLL clock rate is thus 0.625 x 4 = 2.5 GHz.

5. Calculate the required DIV value.

   Because the internal datapath width must be 20 bits and INTDATAWIDTH is High, DIV = 5.

6. Calculate the required PLL divider ratio.

   Using the values f_{CLKIN}, DIV, and f_{PLL_CLOCK} determined above, rearrange Equation 5-1 to calculate the divider ratio as shown in Equation 5-4. The result is a ratio of 2.

$$\frac{PLL\_DIVSEL\_FB}{PLL\_DIVSEL\_REF} = \frac{f_{PLL\_Clock}}{f_{CLKIN} \times DIV} = \frac{2.5\ GHz}{125\ MHz \times 5} = 4 \qquad \text{Equation 5-4}$$

7. Select the PLL divider values.

   Select the smallest divider values that result in the required PLL divider ratio. In this case, using PLL_DIVSEL_FB = 4 and PLL_DIVSEL_REF = 1 results in a ratio of 4.

Configuring the Shared PMA PLL for PCI Express Operation

This example shows how to set the shared PMA PLL divider settings for compliance with PCI Express specification, Rev. 2.0 (5 Gb/s) using Equation 5-1, page 86. The RocketIO GTX Transceiver Wizard and Table 5-3 are simpler alternatives. This example is provided only to illustrate the process with Equation 5-1.

Use Equation 5-1 as described in the following steps:

1. Determine the required line rates.

   For PCI Express operation, both TX and RX use a line rate of 5 Gb/s.

2. Determine the internal datapath width.

   Because PCI Express operation uses 8B/10B encoding, an internal datapath width of 20 bits is required.

3. Determine the desired reference clock rate.

   This example uses a reference clock running at 100 MHz.

4. Calculate the required PLL clock rate.

   The SIPO block utilizes a DDR architecture where both edges of the clock are used to deserialize data. The SIPO block must be fed a clock from the PLL that is one-half the magnitude of the desired line rate. In this PCI Express example, a 5 Gb/s line rate is divided by 2, or 2.5 GHz. Because this RX rate of 2.5 GHz is within the nominal PLL operation range, the external divider (PLL_RXDIVSEL_OUT) must be 1. The PLL clock rate is thus 2.5 GHz x 1 = 2.5 GHz.

5. Calculate the required DIV value.

   Because the internal datapath width must be 20 bits and INTDATAWIDTH is High, DIV = 5.

6. Calculate the required PLL divider ratio.

   Using the values f_{CLKIN}, DIV, and f_{PLL_CLOCK} determined above, rearrange Equation 5-1 to calculate the divider ratio as shown in Equation 5-5. The result is a ratio of 5.

$$\frac{PLL\_DIVSEL\_FB}{PLL\_DIVSEL\_REF} = \frac{f_{PLL\_Clock}}{f_{CLKIN} \times DIV} = \frac{2.5\ GHz}{100\ MHz \times 5} = 5 \qquad \textit{Equation 5-5}$$

7. Select the PLL divider values.

   Select the smallest divider values that result in the required PLL divider ratio. In this case, using PLL_DIVSEL_FB = 5 and PLL_DIVSEL_REF = 1 results in a ratio of 5.

Clocking

Overview

For proper high-speed operation, the GTX transceiver requires a high-quality, low-jitter, reference clock. Because of the shared PMA PLL architecture inside the GTX_DUAL tile, each reference clock sources both channels. The reference clock is used to produce the PLL clock, which is divided by one, two, or four to make individual TX and RX serial clocks and parallel clocks for each GTX transceiver. See "Shared PMA PLL," page 84 for details.

The GTX_DUAL reference clock is provided through the CLKIN port. There are three ways to drive the CLKIN port (see Figure 5-3):

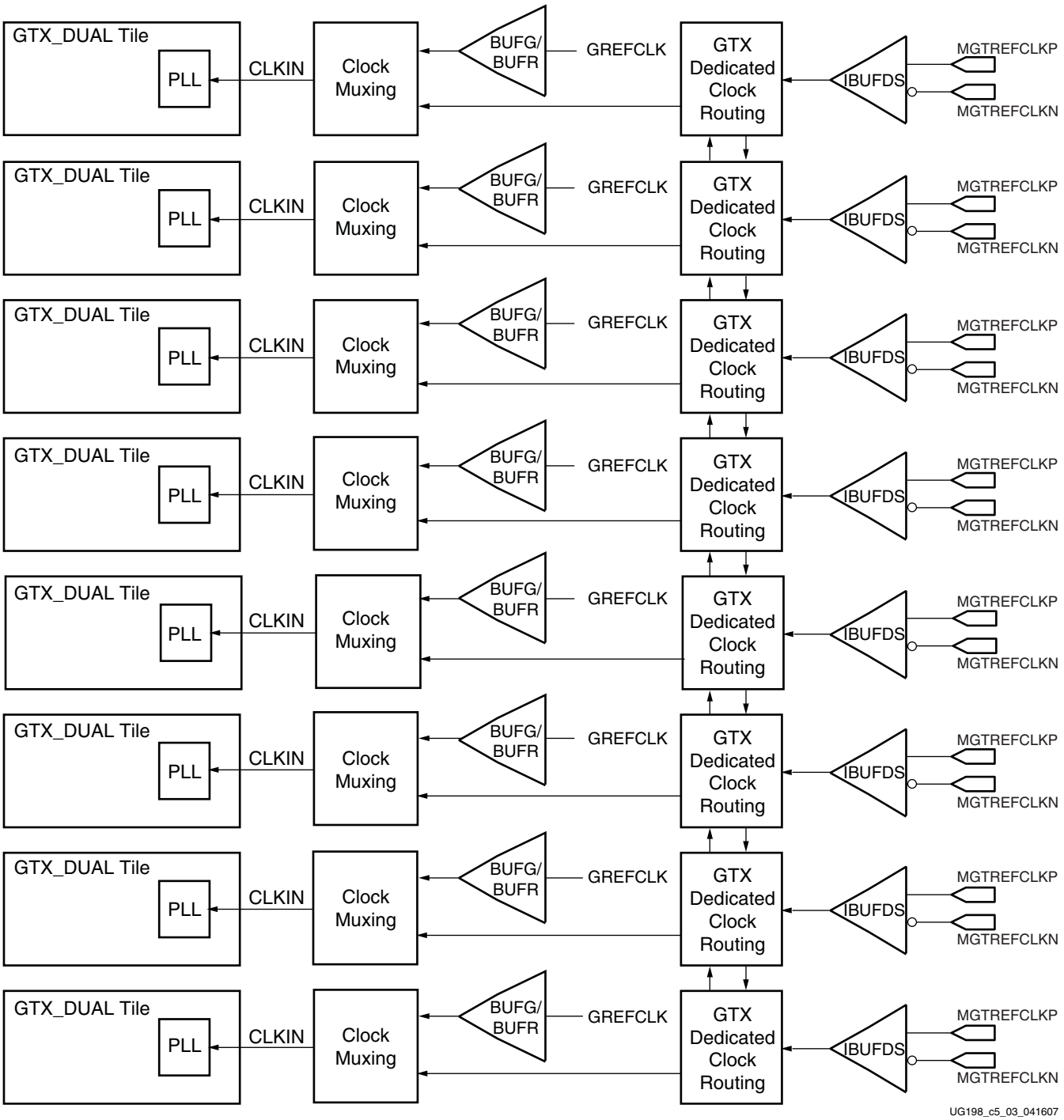
- Using an external oscillator to drive GTX dedicated clock routing
- Using a clock from a neighboring GTX_DUAL tile through GTX dedicated clock routing
- Using a clock from inside the FPGA (GREFCLK)

Using the dedicated clock routing provides the best possible clock to the GTX_DUAL tiles. Each GTX_DUAL tile has a pair of dedicated clock pins, represented by IBUFDS primitives, that can be used to drive the dedicated clock routing. Refer to "REFCLK Guidelines" in Chapter 10 for IBUFDS details.

This section shows how to select the dedicated clocks for use by one or more GTX_DUAL tiles. Guidelines for driving these pins on the board are discussed in Chapter 10, "GTX-to-Board Interface."

When GREFCLK clocking is used for a specific GTX_DUAL tile, the dedicated clock routing is not used. Instead, the global clock resources of the FPGA are connected to the shared PMA PLL. GREFCLK clocking is not recommended for most designs because of the increased jitter introduced by the FPGA clock nets.

The implementation of REFCLKPWRDNB is different between the GTP_DUAL tiles and GTX_DUAL tiles. REFCLKPWRDNB powers down the entire reference clock circuit on the GTP_DUAL tile but only powers down the part of the reference clock circuit that brings in CLKP and CLKN on the GTX_DUAL tile. All other clocks are free to flow, including NORTH, SOUTH, and REFCLK as long as power is applied to the GTX_DUAL tile.



Notes:

1. Refer to "REFCLK Guidelines" in Chapter 10 for details on the IBUFDS primitive.
2. Refer to Appendix F, "Advanced Clocking," for advanced clocking scenarios.

Figure 5-3: **GTX Transceiver Clocking**

Ports and Attributes

Table 5-4 defines the shared clocking ports.

Table 5-4: **Shared Clocking Ports**

Port	Dir	Clock Domain	Description
CLKIN	In	N/A	Reference clock input to the shared PMA PLL.
REFCLKOUT	Out	N/A	The REFCLKOUT port from each GTX_DUAL tile provides access to the reference clock provided to the shared PMA PLL (CLKIN). It can be routed for use in the FPGA logic.

Table 5-5 defines the shared clocking attributes.

Table 5-5: **Shared Clocking Attributes**

Attribute	Type	Description
CLK25_DIVIDER	Integer	CLK25_DIVIDER is set to get an internal clock for the tile. 1: CLKIN $\leq$ 25 MHz 2: 25 MHz < CLKIN $\leq$ 50 MHz 3: 50 MHz < CLKIN $\leq$ 75 MHz 4: 75 MHz < CLKIN $\leq$ 100 MHz 5: 100 MHz < CLKIN $\leq$ 125 MHz 6: 125 MHz < CLKIN $\leq$ 150 MHz 10: 150 MHz < CLKIN $\leq$ 250 MHz 12: CLKIN > 250 MHz
CLKINDC_B	Boolean	Must be set to TRUE. Oscillators driving the dedicated reference clock inputs must be AC coupled. When set to FALSE for testing, the common mode voltage of the driving circuit must match the common mode voltage of the differential clock input pair (MGTCLKP, MGTCLKN). The differential swing must not exceed the maximum differential swing of the clock input pair.(1, 2)
CLKRCV_TRST	Boolean	When set to FALSE, switches off the internal termination resistors of the differential clock input pair. This results in a high impedance input characteristic that is only intended for testing. When set to TRUE, the differential clock input is nominally terminated with a 100 Ω differential impedance. Each clock input in (MGTCLKP, MGTCKLKN) is contacted via a 50 Ωresistor to a midterm nominal voltage of 0.8V.(1, 2)

Notes:

1. Violation of the rules outlined in this section result in a marginal or dysfunctional design, device degradation in the future, or device damage.
2. Consult DS202: *Virtex-5 FPGA Data Sheet* for the common mode voltage values and the associated differential swing and operating conditions.

Description

Clocking from an External Source

Each GTX_DUAL tile has a pair of dedicated pins that can be connected to an external clock source. To use these pins, an IBUFDS primitive is instantiated. In the User Constraints file, the IBUFDS input pins are constrained to the locations of the dedicated clock pins for the GTX_DUAL tile. In the design, the output of the IBUFDS is connected to the CLKIN port. The locations of the dedicated pins for all the GTX_DUAL tiles are documented in Chapter 4, "Implementation." Chapter 10, "GTX-to-Board Interface" provides a selection of suitable external oscillators and describes the board-level requirements for the dedicated reference clock. Figure 5-4 shows a differential GTX clock pin pair sourced by an external oscillator on the board.

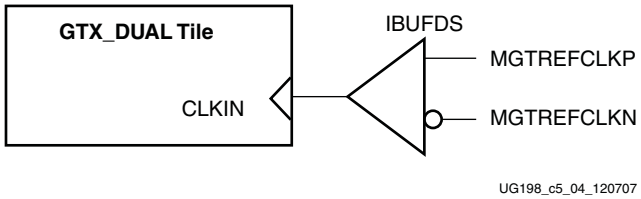


Figure 5-4: **Single GTX_DUAL Tile Clocked Externally**

Clocking from a Neighboring GTX_DUAL Tile

The external clock from one GTX_DUAL tile can be used to drive the CLKIN ports of neighboring tiles. The example in Figure 5-5 uses the clock from one GTX_DUAL tile to clock six neighboring tiles. A GTX_DUAL tile shares its clock with its neighbors using the dedicated clock routing resources.

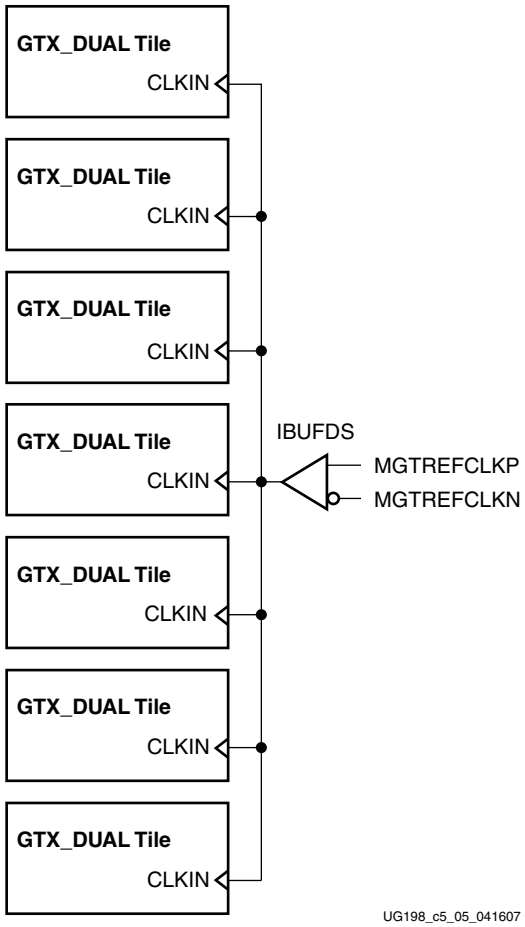


Figure 5-5: **Multiple GTX_DUAL Tiles with Shared Reference Clock**

Note: The following rules must be observed when sharing a reference clock to ensure that jitter margins for high-speed designs are met:

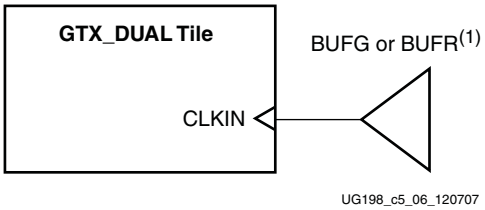
1. The number of GTX_DUAL tiles *above* the sourcing GTX_DUAL tile must *not* exceed three.
2. The number of GTX_DUAL tiles *below* the sourcing GTX_DUAL tile must *not* exceed three.
3. The total number of GTX_DUAL tiles sourced by the external clock pin pair (MGTREFCLKN/MGTREFCLKP) must *not* exceed seven.

The maximum number of GTX transceivers that can be sourced by a single clock pin pair is 14. Designs with more than 14 transceivers require the use of multiple external clock pins to ensure that the rules for controlling jitter are followed. When multiple clock pins are used, an external buffer can be used to drive them from the same oscillator. The same oscillator must be used when the GTX transceivers are combined to form a single channel using channel bonding (see "Configurable Channel Bonding (Lane Deskew)," page 216).

Clocking using GREFCLK

The internal clock nets of the FPGA can provide the reference clock for the GTX_DUAL tile by connecting the output of a global clock buffer (BUFG) or a regional clock buffer (BUFR) to the CLKIN port. This type of clocking, called GREFCLK clocking, has the lowest performance of any of the three clocking methods, because FPGA clocking resources introduce too much jitter for operation at high rates. Avoid GREFCLK clocking. See the *Virtex-5 FPGA Data Sheet* for the jitter margins at different speeds.

Figure 5-6 shows how a GTX_DUAL tile connects to a BUFR or a BUFG. If a BUFR is used, it must be located in the same region as the GTX_DUAL tile.



Notes:

1. Refer to the *Virtex-5 FPGA Data Sheet* and the *Virtex-5 FPGA Configuration Guide* for the maximum clock frequency and jitter limitations of BUFR.

Figure 5-6: **Single GTX_DUAL Tile Clocked from the FPGA**

Reset

Overview

The GTX_DUAL tile must be reset before any of the GTX transceivers can be used. There are three ways to reset a GTX_DUAL tile:

1. Power up and configure the FPGA. Power-up reset is covered in this section.
2. Drive the GTXRESET port High to trigger a full asynchronous reset of the GTX_DUAL tile. GTXRESET is covered in this section.
3. Assert one or more of the individual reset signals on the block to reset a specific subcomponent of the tile. These resets are covered in detail in the sections for each subcomponent (Table 5-8, page 103).

The GTX_DUAL reset hierarchy is shown in Figure 5-7, page 99.

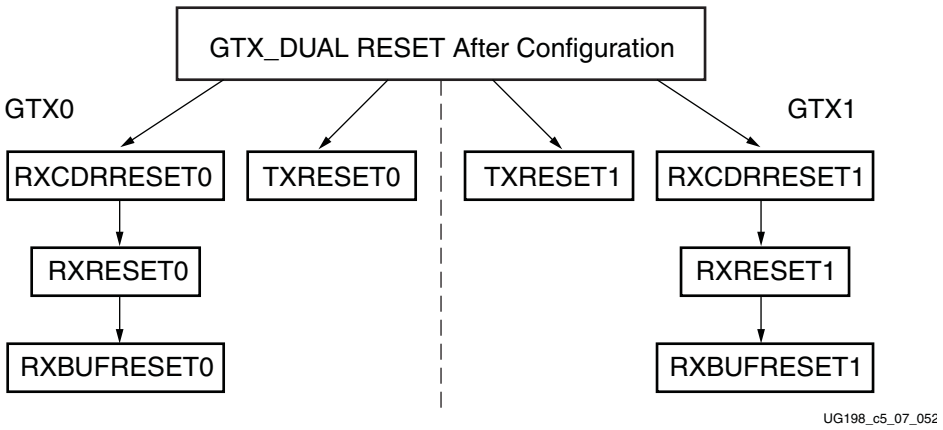


Figure 5-7: **GTX_DUAL Reset Hierarchy**

Ports and Attributes

Table 5-6 defines the shared tile reset ports.

Table 5-6: **Shared Tile Reset Ports**

Port	Dir	Domain	Description
GTXRESET(1)(2)	In	Async	This port is driven High and then deasserted to start the full GTX_DUAL reset sequence. This sequence takes about 120 µs to complete, and systematically resets all subcomponents of the GTX_DUAL tile.
RESETDONE0 RESETDONE1	Out	Async	This port goes High when the GTX transceiver has finished reset and is ready for use. For this signal to work correctly, CLKIN and all clock inputs on the individual GTX transceiver (TXUSRCLK, TXUSRCLK2, RXUSRCLK, RXUSRCLK2) must be driven.
RXBUFRESET0(1) RXBUFRESET1(2)	In	Async	This active-High signal resets the RX elastic buffer logic.
RXCDRRESET0(1) RXCDRRESET1(2)	In	RXUSRCLK2	Individual reset signal for the RX CDR and the RX part of the PCS for this channel. This signal is driven High to cause the CDR to give up its current lock and return to the shared PMA PLL frequency.
RXRESET0(1) RXRESET1(2)	In	Async	Active-High reset for the RX PCS logic.
TXRESET0(1) TXRESET1(2)	In	Async	Resets the PCS of the GTX transmitter, including the phase adjust FIFO, the 8B/10B encoder, and the FPGA TX interface.

Notes:

1. When these resets are active, then RESETDONE0 is driven Low. All resets are asynchronous, positive-edge triggered, and synchronized internally to a specific clock domain.
2. When these resets are active, then RESETDONE1 is driven Low. All resets are asynchronous, positive-edge triggered, and synchronized internally to a specific clock domain.

Table 5-7 defines the shared tile reset attributes.

Table 5-7: **Shared Tile Reset Attributes**

Attribute	Type	Description
CDR_PH_ADJ_TIME[4:0]	5-bit Binary	Sets time to wait after deassertion of CDR phase reset before completion of optional reset sequence for PCI Express operation during electrical idle.
RX_EN_IDLE_HOLD_CDR	Boolean	Enables CDR to hold its data during optional reset sequence for PCI Express operation during electrical idle.
RX_EN_IDLE_HOLD_DFE_0 RX_EN_IDLE_HOLD_DFE_1	Boolean	When TRUE, enables ability to restore DFE contents from internal registers after termination of an electrical idle state for PCI Express operation.
RX_EN_IDLE_RESET_BUF_0 RX_EN_IDLE_RESET_BUF_1	Boolean	When TRUE, the elastic buffer is reset if a valid signal is not present on the RX inputs. Works in conjunction with attributes RX_IDLE_HI_CNT_0, RX_IDLE_HI_CNT_1, RX_IDLE_LO_CNT_0, and RX_IDLE_LO_CNT_1.
RX_EN_IDLE_RESET_PH	Boolean	Enables reset of CDR phase circuits during optional reset sequence for PCI Express operation during electrical idle.
RX_EN_IDLE_RESET_FR	Boolean	Enables reset of CDR frequency circuits during optional reset sequence for PCI Express operation during electrical idle.
RX_IDLE_HI_CNT_0[3:0] RX_IDLE_HI_CNT_1[3:0]	4-bit Binary	Programmable counters used in association with resetting the elastic buffer in response to the absence of valid data on the RX inputs. Determines how long the RX inputs must remain in electrical idle before the elastic buffer reset is asserted.
RX_IDLE_LO_CNT_0[3:0] RX_IDLE_LO_CNT_1[3:0]	4-bit Binary	Programmable counters associated with deasserting the reset condition of the elastic buffer in response to the detection of valid data on the RX inputs. Determines how long the RX inputs must have good data (not be in electrical idle) before elastic buffer reset is deasserted.

Description

GTX Reset in Response to Completion of Configuration

Figure 5-8 shows the GTX_DUAL reset sequence following completion of configuration of a powered-up GTX_DUAL tile. The same sequence is activated any time PLLPOWERDOWN goes from High to Low during normal operation.

Refer to "Power Control," page 107 on power-down for details about PLLPOWERDOWN.

Notes:

1. The timing of the reset sequencer inside the GTX_DUAL tile depends on the frequency of CLK25. The estimates given in this figure assume that the frequency of CLK25 is 25 MHz.
2. The GTX_DUAL tile reset sequence responds when the completion of configuration occurs only if PLLPOWERDOWN is Low when configuration finishes.

Figure 5-8: **GTX_DUAL Reset Sequence Following Configuration**

The following GTX_DUAL sections are affected by the reset sequence after configuration:

- Shared PMA PLL
- GTX0 transmit section (PMA and PCS)
- GTX0 receive section (PMA and PCS)
- GTX1 transmit section (PMA and PCS)
- GTX1 receive section (PMA and PCS)

GTX Reset When the GTXRESET Port is Asserted

Figure 5-9 is similar to Figure 5-8, showing the full reset sequence occurring in response to a pulse on GTXRESET. GTXRESET acts as an asynchronous reset signal.

Figure 5-9: **Reset Sequence Triggered by the GTXRESET Pulse**

The following GTX_DUAL sections are affected by the GTXRESET sequence:

- Shared PMA PLL
- GTX0 transmit section (PMA and PCS)
- GTX0 receive section (PMA and PCS)
- GTX1 transmit section (PMA and PCS)
- GTX1 receive section (PMA and PCS)

GTX Component-Level Resets

Component resets are primarily used for special cases. These resets are needed when only the reset of a specific GTX_DUAL subsection is required. Each of the component-level reset signals is described in Table 5-6, page 99.

All component resets are asynchronous with the exception of PRBSCNTRESET, which is synchronous to RXUSRCLK2.

Link Idle Reset Support

The Link Idle Reset circuit used with the GTP_DUAL tiles in Virtex-5 LXT and SXT devices is not necessary when using the GTX_DUAL tiles in Virtex-5 FXT and TXT devices. The RX elastic buffer reset sequence during an electrical idle condition is available for additional functionality.

During an electrical idle condition, the Clock Data Recovery (CDR) circuit in the receiver can lose lock ("RX Clock Data Recovery," page 176). To restart the CDR after an electrical idle condition, set the RX_EN_IDLE_RESET_PH, RX_EN_IDLE_RESET_FR, and RX_EN_IDLE_HOLD_CDR attributes to TRUE. These attributes affect both GTX transceivers in a GTX_DUAL tile. The CDR_PH_ADJ_TIME[4:0] attribute sets the wait time before deassertion of the CDR phase reset and must be left at the default value. The RX_EN_IDLE_RESET_BUF_0 attribute enables a reset sequence for the GTX0 transceiver's RX elastic buffer and RX_EN_IDLE_RESET_BUF_1 does the same for the GTX1 transceiver. The RX elastic buffer of a GTX transceiver is automatically held in reset and

reinitialized after the end of an electrical idle condition on the RX pin pair if the RX_EN_IDLE_RESET_BUF_(0/1) attributes are set to TRUE. The RX_IDLE_HI_CNT_(0/1) and RX_IDLE_LO_CNT_(0/1) attributes control the timing of the RX elastic buffer reset sequence and must be left at the default values.

Resetting the GTX_DUAL Tile

Each GTX_DUAL tile offers several ways to reset its subcomponents. Table 5-8 shows all the different ways of resetting a GTX_DUAL tile, and the subcomponents that are affected by each type of reset.

Table 5-8: **Available Resets Pins and the Components Reset by These Reset Pins**

	Component	Configuration	GTXRESET	PLLPOWERDOWN (Falling Edge)	TXRESET0 TXRESET1	RXCDRRESET0 RXCDRRESET1	RXRESET0 RXRESET1	RXBUFRESET0 RXBUFRESET1	PRBSCNTRESET0 PRBSCNTRESET1
GTX-to-Board Interface	Termination Resistor Calibration	✓							
Shared Resources	Shared PMA PLL	✓	✓	✓					
	PLL Lock Detection	✓	✓	✓					
	Reset Control	✓	✓	✓					
	Power Control	✓	✓	✓					
	Clocking	✓	✓	✓					
	DRP	✓							
TX PCS	FPGA TX Interface	✓	✓	✓	✓				
	8B/10B Encoder	✓	✓	✓	✓				
	TX Buffer	✓	✓	✓	✓				
	PRBS Generator	✓	✓	✓	✓				
	Polarity Control	✓	✓	✓	✓				
TX PMA	PISO	✓	✓	✓					
	TX Pre-emphasis	✓	✓	✓					
	TX OOB & PCI	✓	✓	✓					
	TX Driver	✓	✓	✓					

Table 5-8: **Available Resets Pins and the Components Reset by These Reset Pins *(Cont'd)***

	Component	Configuration	GTXRESET	PLLPOWERDOWN (Falling Edge)	TXRESET0 TXRESET1	RXCDRRESET0 RXCDRRESET1	RXRESET0 RXRESET1	RXBUFRESET0 RXBUFRESET1	PRBSCNTRESET0 PRBSCNTRESET1
RX PCS	FPGA RX Interface	✓	✓	✓		✓	✓		
	RX Elastic Buffer	✓	✓	✓		✓	✓	✓	
	RX Status Control	✓	✓	✓		✓	✓		
	8B/10B Decoder	✓	✓	✓		✓	✓		
	Comma Detect and Align	✓	✓	✓		✓	✓		
	RX LOS State Machine	✓	✓	✓		✓	✓		
	RX Polarity	✓	✓	✓		✓	✓		
	PRBS Checker	✓	✓	✓		✓	✓		✓
	5x Oversampler	✓	✓	✓		✓	✓		
RX PMA	SIPO	✓	✓	✓		✓			
	RX CDR	✓	✓	✓		✓			
	RX Termination and Equalization	✓	✓	✓					
	RX OOB	✓	✓	✓		✓			
Loopback	Loopback paths	✓	✓	✓					

The reset that occurs after configuration and the GTXRESET port are the most common ways to prepare GTX_DUAL(s) for operation, but certain situations can require the use of other reset ports. Table 5-9 outlines some of these situations and indicates the recommended resets.

Table 5-9: **Recommended Resets for Common Situations**

Situation	Components to be Reset	Recommended Reset[1]
After power up and configuration	Entire GTX_DUAL tile	Reset after configuration is automatic
After turning on a reference clock	Shared PMA PLL	GTXRESET
After changing a reference clock	Shared PMA PLL	GTXRESET
Parallel clock source reset	TX PCS, RX PCS, Phase Alignment	TXRESET0, TXRESET1, RXRESET0, RXRESET1
After remote power up	RX CDR	A built-in reset sequencer automatically sets these situations by setting RX_EN_IDLE_RESET_PH, RX_EN_IDLE_RESET_FR, RX_EN_IDLE_HOLD_CDR to TRUE.
After PCI Express electrical idle condition	RX CDR	
After connecting RXN/RXP	RX CDR	
After a TX buffer error	TX Buffer	TXRESET0, TXRESET1
After an RX buffer error	RX Elastic Buffer	RXBUFRESET0, RXBUFRESET1

Table 5-9: **Recommended Resets for Common Situations**

Situation	Components to be Reset	Recommended Reset[1]
Before channel bonding	RX CDR, then RXBUFFER after CDR is locked	Either assert RXBUFRESET, or automatically reset by setting RX_EN_IDLE_RESET_BUF = TRUE to enable the RXBUFRESET0/RXBUFRESET1 sequence
After PRBS error	PRBS Error counter	PRBSCNTRESET0, PRBSCNTRESET1
After oversampler error	Oversampler	RXRESET0, RXRESET1

Notes:

1. The recommended reset has the smallest impact on the other components of the GTX_DUAL tile.

Examples

Power-up and Configuration

All GTX_DUAL tiles are reset automatically after configuration. The supplies for the calibration resistor and calibration resistor reference must be powered up before configuration to ensure correct calibration of the termination impedance of all transceivers.

After Turning on a Reference Clock

The reference clock source(s) and the power to the GTX_DUAL tile must be available before configuring the FPGA. The reference clock must be stable before configuration especially when using PLL based clock sources (e.g., voltage controlled crystal oscillators). If the reference clock(s) or GTX_DUAL tile(s) are powered up after configuration, apply GTXRESET to allow the shared PMA PLL(s) to lock.

After Changing a Reference Clock

Whenever the reference clock input to a GTX_DUAL tile is changed, the shared PMA PLL must be reset afterwards to ensure that it locks to the new frequency. The GTXRESET port must be used for this purpose.

Parallel Clock Source Reset

The clocks driving TXUSRCLK, RXUSRCLK, TXUSRCLK2, and RXUSRCLK2 must be stable for correct operation. These clocks are often driven from a PLL or DCM in the FPGA to meet phase and frequency requirements. If the DCM or PLL loses lock, and begins producing incorrect output, TXRESET and RXRESET must be used to hold transceiver PCS in reset until the clock source is locked again.

If the TX or RX buffer is bypassed and phase alignment is in use, phase alignment must be performed again after the clock source relocks.

Note: Bypassing the TX or RX buffer is an advanced feature and is not recommended for normal operation. TX or RX buffer bypass operation can be guaranteed only under certain system-level conditions and data rates.

After Remote Power-up

If the remote source of incoming data is powered up after the GTX transceiver receiving its data is operating, the RX CDR must be reset to ensure a clean lock to the incoming data. By

following the guidelines in "Link Idle Reset Support," page 102, the electrical idle reset situation is automatically managed.

Electrical Idle Reset

When the differential voltage of the RX input to a GTX transceiver drops to OOB or electrical idle levels, the RX CDR can be pulled out of lock by the apparent sudden change in frequency. By following the guidelines in "Link Idle Reset Support," page 102, the electrical idle reset situation is automatically managed.

After Connecting RXP/RXN

When the RX data to the GTX transceiver comes from a connector that can be plugged in and unplugged, the RX CDR must be reset when the data source is plugged in to ensure that it can lock to incoming data. By following the guidelines in "Link Idle Reset Support," page 102, the electrical idle reset situation is automatically managed.

After a TX Buffer Error

When the TX buffer overflows or underflows, it must be reset using TXRESET to ensure correct behavior.

After an RX Buffer Error

After an RX buffer overflow or underflow, the RX elastic buffer must be reset using the RXBUFRESET port to ensure correct behavior.

Before Channel Bonding

For successful channel bonding, the RX elastic buffers of all the bonded transceivers must be written using the same recovered frequency, and read using the same RXUSRCLK frequency.

To provide the same RXUSRCLK frequency to all bonded transceivers, use a low skew clock buffer (for example, a BUFG) to drive all the RXUSRCLK ports from the same clock source. Bonding should not be attempted until the clock source is stable.

To provide the same recovered clock to all bonded transceivers:

- All of the TX data sources must be locked to the same reference clock.
- All of the bonded transceivers must have CDR lock to the incoming data.

The required reset for channel bonding is as follows:

- To automatically reset the CDR of all bonded transceivers, set RX_EN_IDLE_RESET_PH, RX_EN_IDLE_RESET_FR, RX_EN_IDLE_HOLD_CDR to TRUE.
- Wait for CDR lock and bit alignment on all bonded transceivers.
- Either assert RXBUFRESET to all bonded transceivers, or automatically reset by setting RX_EN_IDLE_RESET_BUF = TRUE to enable the RXBUFRESET sequence.
- Attempt channel bonding.

See "RX Clock Data Recovery" in Chapter 7 for recommended methods of detecting CDR lock.

After a PRBS Error

To clear the RXPRBSERR signal after the PRBS error threshold is exceeded, assert PRBSERRRESET.

After an Oversampler Error

If RXOVERSAMPLEERR goes High to indicate an overflow or underflow in the oversampling block, asserting RXRESET clears it.

Power Control

Overview

The GTX_DUAL tiles support a range of power-down modes. These modes support both generic power management capabilities as well as those defined in the PCI Express and SATA standards.

Ports and Attributes

Table 5-10 defines the power ports.

Table 5-10: **Power Ports**

Port	Dir	Domain	Description
CLKIN	In	N/A	Reference clock input to the shared PMA PLL. The CLKIN rate in conjunction with CLK25_DIVIDER determines the timing of power-down state transitions for PCI Express designs.
PLLPOWERDOWN(1)	In	Async	Powers down the shared PMA PLL: 0: Shared PMA PLL is powered up 1: Shared PMA PLL is powered down
REFCLKPWRDNB(2)	In	Async	Powers down the part of the GTX reference clock circuit between the differential clock pair input pin and the dedicated clock routing circuit: 0: Circuit used to bring in CLKP and CLKN is powered down 1: Circuit used to bring in CLKP and CLKN is powered up
RXPOWERDOWN0[1:0] RXPOWERDOWN1[1:0]	In	Async	Powers down the RX lanes. The encoding complies with PCI Express encoding. TX and RX can be powered down separately. However, for PCI Express compliance, TXPOWERDOWN and RXPOWERDOWN must be used together. 00: P0 (normal operation) 01: P0s (low recovery time power down) 10: P1 (longer recovery time; Receiver Detection is still on) 11: P2 (lowest power state)
TXDETECTRX0 TXDETECTRX1	In	TXUSRCLK2	Activates the receive detection sequence. The sequence ends when PHYSTATUS is asserted to indicate that the results of the test are ready on RXSTATUS.
TXELECIDLE0 TXELECIDLE1	In	TXUSRCLK2	Drives TXN and TXP to the same voltage to perform electrical idle/beaconing for PCI Express designs.

Table 5-10: **Power Ports *(Cont'd)***

Port	Dir	Domain	Description
TXPOWERDOWN0[1:0](3) TXPOWERDOWN1[1:0](3)	In	TXUSRCLK2	Powers down the TX lanes. The encoding complies with PCI Express encoding. TX and RX can be powered down separately, however, for PCI Express compliance, TXPOWERDOWN and RXPOWERDOWN must be used together. 00: P0 (normal operation) 01: P0s (low recovery time power down) 10: P1 (longer recovery time; Receiver Detection is still on) 11: P2 (lowest power state)

Notes:

1. Because of the shared PMA PLL, a powerdown via PLLPOWERDOWN or REFCLKPWRDNB affects both channels.
2. REFCLKPWRDNB only powers down the part of the reference clock circuit that brings in CLKP and CLKN on the GTX_DUAL tile. In the GTP_DUAL tile implementation, REFCLKPWRDNB powers down the entire reference clock circuit.
3. The TXPOWERDOWN(0/1) ports in the GTX_DUAL tiles are in the TXUSRCLK2 clock domain. This is different from the GTP_DUAL tile's TXPOWERDOWN(0/1) ports and the GTX_DUAL tile's RXPOWERDOWN(0/1) ports which are all asynchronous.

Table 5-11 defines the power attributes.

Table 5-11: **Power Attributes**

Attribute	Type	Description
CLK25_DIVIDER	Integer	The internal digital logic for GTX_DUAL tile management runs at about 25 MHz. CLK25_DIVIDER is set to get an internal clock for the tile. The CLK25_DIVIDER in conjunction with CLKIN determines the timing of powerdown state transitions for PCI Express by adjusting the internal 25 MHz clock rate.
PCI_EXPRESS_MODE_0 PCI_EXPRESS_MODE_1	Boolean	Setting this attribute to TRUE enables certain operations specific to PCI Express designs, specifically, recognizing TXELECIDLE = 1, TXCHARDISPMODE = 1, TXCHARDISPVAL = 0 as a request to power down the channel. TXCHARDISPMODE = 1 and TXCHARDISPVAL = 0 encode the PIPE interface signal TXCompliance = 1 of the PIPE specification. The default for this attribute is TRUE for PCI Express operation. For all other protocols, the default setting is FALSE.
TRANS_TIME_FROM_P2_0 TRANS_TIME_FROM_P2_1	12-bit Hex	Transition time from the P2 state in internal 25 MHz clock cycles. The exact time depends on the CLKIN rate and the setting of CLK25_DIVIDER. The P2 state is related to the PCI Express power state definition.
TRANS_TIME_NON_P2_0 TRANS_TIME_NON_P2_1	8-bit Hex	Transition time to or from any state except P2 in internal 25 MHz clock cycles. The exact time depends on the CLKIN rate and the setting of CLK25_DIVIDER. This setting is related to the PCI Express power state definition.
TRANS_TIME_TO_P2_0 TRANS_TIME_TO_P2_1	10-bit Hex	Transition time to the P2 state in internal 25 MHz clock cycles. The exact time depends on the CLKIN rate and the setting of CLK25_DIVIDER. This setting is related to the PCI Express power state definition.

Description

The GTX_DUAL tile offers different levels of power control. Each channel in each direction can be powered down separately using TXPOWERDOWN and RXPOWERDOWN. Additionally, the part of the reference clock circuit between the differential clock input pair pin and dedicated clock routing circuit can be powered down. The PLLPOWERDOWN port directly affects the shared PMA PLL and therefore both channels of the GTX_DUAL tile.

Generic GTX Power-Down Capabilities

The GTX_DUAL tile provides several power-down features that can be used in a wide variety of applications. Table 5-12 summarizes these capabilities.

Table 5-12: **Basic Power-Down Functions Summary**

Function	Controlled By	Affects
REFCLK Power Down	REFCLKPWRDNB	The GTX_DUAL tile (TX and RX for both transceivers) when using an external oscillator to drive the dedicated clock routing, and all downstream GTX_DUAL tiles sharing that REFCLK.
PLL Power Down	PLLPOWERDOWN	TX and RX for both transceivers in a GTX_DUAL tile. Powers down the shared PMA PLL as well as the RX and TX PMA circuits.
TX Power Down	TXPOWERDOWN[1:0]	TX in a single transceiver.
RX Power Down	RXPOWERDOWN[1:0]	RX in a single transceiver.

REFCLK Power Down

To activate the REFCLK power-down mode, the active-Low REFCLKPWRDNB signal is asserted. When REFCLKPWRDNB is asserted, toggling of all circuitry between the differential clock pair input pins and the dedicated clock routing circuitry is suppressed. As long as power is applied to a GTX_DUAL tile, all other circuitry including the dedicated clock routing and shared PMA PLL continue to toggle even when REFCLKPWRDNB is asserted. This is a different implementation from the GTP_DUAL tile where when REFCLKPWRDNB is asserted, the dedicated clocking routing circuity is disabled and all circuitry clocked by the REFCLK input is suppressed, including the shared PMA PLL and all clocks derived from it. If the GTX_DUAL tiles share a common reference, REFCLK is suppressed to tiles that are downstream in the clock routing chain. Figure 5-3, page 94 illustrates how the dedicated clock routing blocks forward REFCLKs between GTX_DUAL tiles.

Deactivation of the REFCLK power-down mode is indicated by the PLLLKDET signal, which is asserted on the tiles sourcing the affected reference clocks.

PLL Power Down

To activate the PLL power-down mode, the active-High PLLPOWERDOWN signal is asserted. When PLLPOWERDOWN is asserted, the shared PMA PLL and both the PMA TX and PMA RX circuits are powered down. As a result, all clocks derived from the PMA PLL are stopped. This is a different implementation than for the GTP_DUAL tile where

when PLLPOWERDOWN is asserted, only the shared PMA PLL and all clocks derived from it are stopped.

Recovery from this power state is indicated by the assertion of the PLLLKDET signal on the tile whose REFCLKPWRDNB signal is asserted.

TX and RX Power Down

When the TX and RX power-down signals are used in non PCI Express implementations, TXPOWERDOWN and RXPOWERDOWN can be used independently. However, when these interfaces are used in non PCI Express applications, only two power states are supported, as shown in Table 5-13. When using this power-down mechanism, the following must be True:

- TXPOWERDOWN[1] and TXPOWERDOWN[0] are connected together.
- RXPOWERDOWN[1] and RXPOWERDOWN[0] are connected together.
- TXDETECTRX must be strapped Low.
- TXELECIDLE must be strapped to TXPOWERDOWN[1] and TXPOWERDOWN[0].

Table 5-13: **TX and RX Power States for Non PCI Express Operation**

TXPOWERDOWN[1:0] or RXPOWERDOWN[1:0]	Description
00	P0 mode. Transceiver TX or RX is active sending or receiving data.
11	P2 mode. Transceiver TX or RX is idle.

Power Control Features for PCI Express Operation

The GTX_DUAL tile implements all of the functions needed for power control states compatible with those defined in the PCI Express and PIPE specifications. When implementing PCI Express compatible power control, the following conditions must be met:

- The TXPOWERDOWN and RXPOWERDOWN signals on each GTX transceiver must be connected together to ensure that they are in the same state at all times.
- The REFCLKPWRDNB and PLLPOWERDOWN signals must be held in inactive states.

Table 5-14: **TX and RX Power States for PCI Express Operation**

TXPOWERDOWN[1:0] and RXPOWERDOWN[1:0]	TXDETECTRX	TXELECIDLE	Description
00 (P0 state)	0	0	The PHY is transmitting data. The MAC provides data bytes to be sent every clock cycle.
	0	1	The PHY is not transmitting and is in the electrical idle state.
	1	0	The PHY goes into loopback mode.
	1	1	Not permitted.

Table 5-14: **TX and RX Power States for PCI Express Operation *(Cont'd)***

TXPOWERDOWN[1:0] and RXPOWERDOWN[1:0]	TXDETECTRX	TXELECIDLE	Description
01 (P0s state)	Don't Care	0	The MAC must always put the PHY into the electrical idle state while in P0s state. The PHY behavior is undefined if TXELECIDLE is deasserted while in P0s or P1.
		1	The PHY is not transmitting and is in the electrical idle state.
10 (P1 state)	Don't Care	0	Not permitted. The MAC must always put the PHY into the electrical idle state while in P1. The PHY behavior is undefined if TXELECIDLE is deasserted while in P0s or P1.
	0	1	The PHY is idle.
	1	1	The PHY does a receiver detection operation.
11 (P2 state)	Don't Care	0	The PHY transmits beacon signaling
		1	The PHY is idle.

The GTX_DUAL acknowledges changes in the PCI Express power mode by asserting the PHYSTATUS signals for one clock cycle independently for each transceiver.

Power-Down Transition Times

The delays between changes in the power-down state when TXPOWERDOWN and RXPOWERDOWN are changed are controlled by the TRANS_TIME_FROM_P2, TRANS_TIME_NON_P2, and TRANS_TIME_TO_P2 attributes as described in Table 5-11.

Each TRANS_TIME delay is set in terms of internal 25 MHz clock cycles. The internal 25 MHz clock rate is set using the CLK25_DIVIDER attribute and the reference clock rate. Equation 5-6 is used to determine the actual rate.

$$\text{Transition time [ns]} = \left(\frac{\text{CLK25\_DIVIDER}}{\text{CLKIN}}\right) \times \text{TRANS\_TIME attribute} \qquad \textit{Equation 5-6}$$

Examples

The example in Figure 5-10 shows the recommended method to power down an unused tile or an unused transceiver in a tile. In the example, an uninstantiated GTX_DUAL tile has a 1 assigned to all power control ports to allow forwarding of a reference clock from GTX_DUAL neighbor tiles.

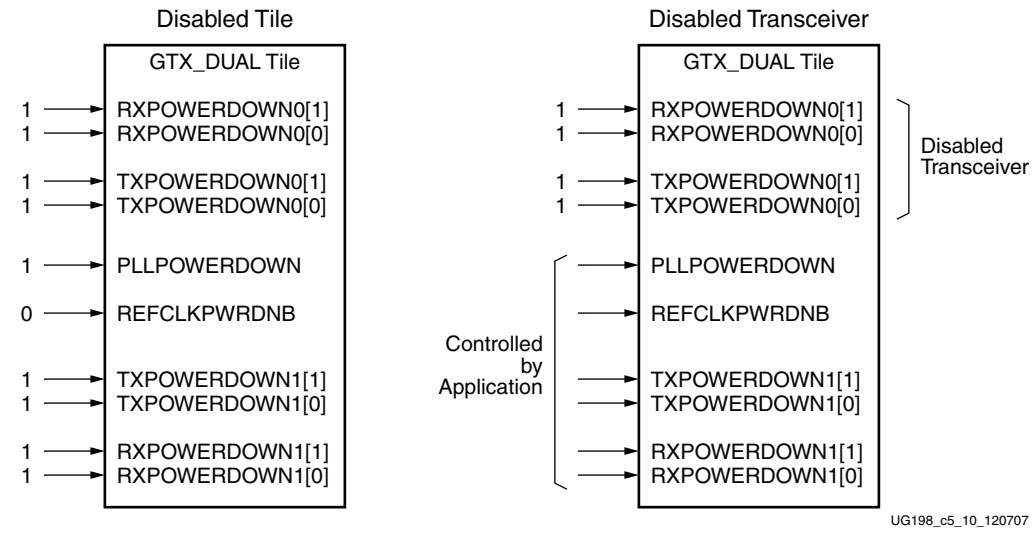


Figure 5-10: **Powering Down an Unused Tile**

Figure 5-11 shows how to connect power-down signals for a four-lane PIPE compatible configuration.

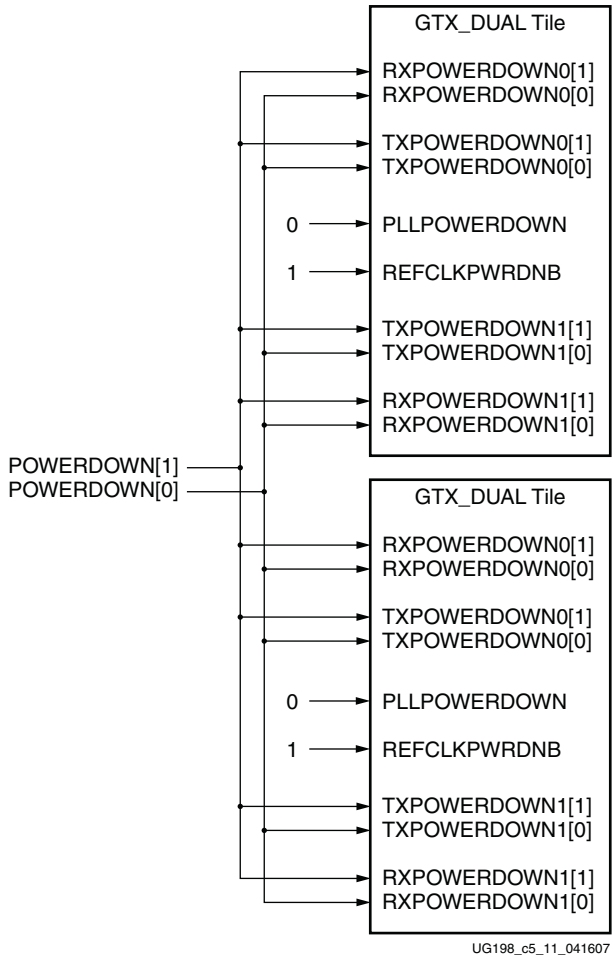


Figure 5-11: **4x PIPE Compatible Configuration**

Dynamic Reconfiguration Port

Overview

The dynamic reconfiguration port (DRP) allows the dynamic change of parameters of the GTX_DUAL tile. The DRP interface is a processor-friendly synchronous interface with an address bus (DADDR) and separated data buses for reading (DO) and writing (DI) configuration data to the GTX_DUAL tile. An enable signal (DEN), a read/write signal (DWE), and a ready/valid signal (DRDY) are the control signals that implement read and write operations, indicate operation completion, or indicate the availability of data.

Ports and Attributes

Table 5-15 defines the DRP ports.

Table 5-15: **DRP Ports**

Port	Dir	Clock Domain	Description
DADDR[6:0]	In	DCLK	DRP address bus
DCLK	In	N/A	DRP interface clock
DEN	In	DCLK	Set to 1 to enable a read or write operation. Set to 0 on DCLK cycles where no operation is required.
DI[15:0]	In	DCLK	Data bus for writing configuration data from the FPGA logic resources to the GTX_DUAL tile.
DO[15:0]	Out	DCLK	Data bus for reading configuration data from the GTX_DUAL tile to the FPGA logic resources.
DRDY	Out	DCLK	Indicates operation is complete for write operations and data is valid for read operations.
DWE	In	DCLK	Set to 0 for read operations. Set to 1 for write operations.

There are no attributes in this section.

Description

The *Virtex-5 FPGA Configuration Guide* provides detailed information on the DRP interface. Refer to Appendix D, "DRP Address Map of the GTX_DUAL Tile," for a map of GTX_DUAL DRP attributes sorted alphabetically by name and by address.

Stopping the reference clock during a DRP operation can prevent the correct termination of the operation.

Chapter 6

GTX Transmitter (TX)

This chapter shows how to configure and use each of the functional blocks inside the GTX transmitter.

Transmitter Overview

Each GTX transceiver in the GTX_DUAL tile includes an independent transmitter, which consists of a PCS and a PMA. Figure 6-1 shows the functional blocks of the transmitter. Parallel data flows from the FPGA into the FPGA TX interface, through the PCS and PMA, and then out the TX driver as high-speed serial data. Refer to Appendix E, "Low Latency Design," for latency information on this block diagram.

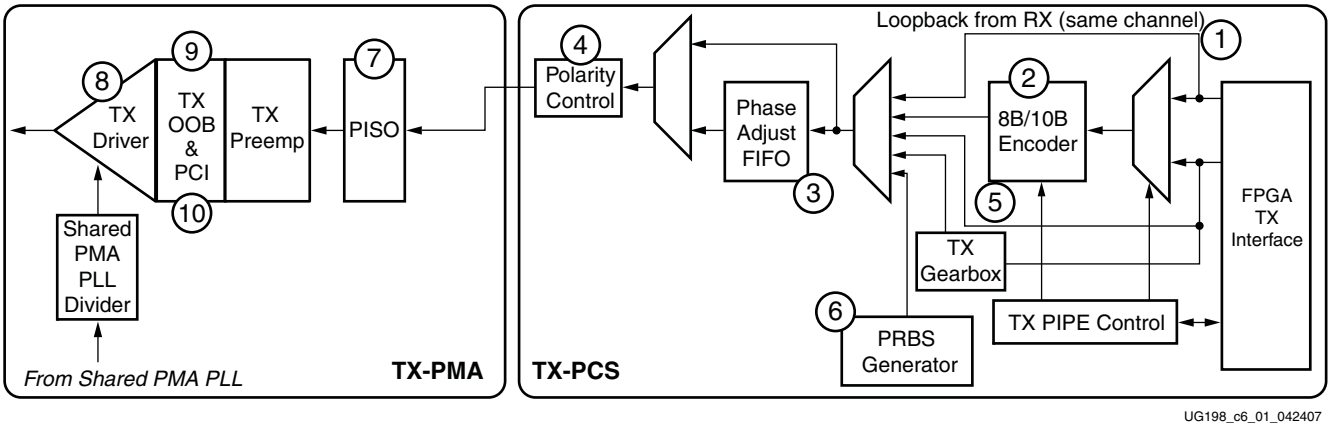


Figure 6-1: **GTX TX Block Diagram**

The key elements within the GTX transmitter are:

1. "FPGA TX Interface," page 116
2. "Configurable 8B/10B Encoder," page 125
3. "TX Buffering, Phase Alignment, and TX Skew Reduction," page 138
4. "TX Polarity Control," page 144
5. "TX Gearbox," page 131
6. "TX PRBS Generator," page 145
7. "Parallel In to Serial Out," page 146
8. "Configurable TX Driver," page 147
9. "Receive Detect Support for PCI Express Operation," page 151
10. "TX Out-of-Band/Beacon Signaling," page 154

FPGA TX Interface

Overview

The FPGA TX interface is the FPGA's gateway to the TX datapath of the GTX transceiver. Applications transmit data through the GTX transceiver by writing data to the TXDATA port on the positive edge of TXUSRCLK2.

The width of the port can be configured to be one, two, or four bytes wide. The actual width of the port depends on the GTX_DUAL tile's INTDATAWIDTH setting (controls the width of the internal datapath), and whether or not the 8B/10B encoder is enabled. Port widths can be 8 bits, 10 bits, 16 bits, 20 bits, 32 bits, and 40 bits.

The rate of the parallel clock (TXUSRCLK2) at the interface is determined by the TX line rate, the width of the TXDATA port, and whether or not 8B/10B encoding is enabled. A second parallel clock (TXUSRCLK) must be provided for the internal PCS logic in the transmitter. This chapter shows how to drive the parallel clocks and explains the constraints on those clocks for correct operation. The highest transmitter data rates require a 4-byte interface to achieve a TXUSRCLK2 rate in the specified operating range.

Ports and Attributes

Table 6-1 defines the FPGA TX interface ports.

Table 6-1: **FPGA TX Interface Ports**

Port	Direction	Clock Domain	Description
INTDATAWIDTH	In	Async	Specifies the width of the internal datapath for the entire GTX_DUAL tile. This shared port is also described in "Shared PMA PLL," page 84. • 0: Internal datapath is 16 bits wide • 1: Internal datapath is 20 bits wide
REFCLKOUT	Out	N/A	The REFCLKOUT port from each GTX_DUAL tile provides direct access to the reference clock provided to the shared PMA PLL (CLKIN). It can be routed for use in the FPGA logic.
TXCHARDISPMODE0[3:0] TXCHARDISPMODE1[3:0]	In	TXUSRCLK2	TXCHARDISPMODE and TXCHARDISPVAL allow control of the 8B/10B outgoing data disparity when 8B/10B encoding is enabled. When 8B/10B encoding is disabled, TXCHARDISPMODE is used to extend the data bus for TX interfaces with a width that is a multiple of 10. See "FPGA TX Interface," page 116. TXCHARDISPMODE[3] corresponds to TXDATA[31:24] TXCHARDISPMODE[2] corresponds to TXDATA[23:16] TXCHARDISPMODE[1] corresponds to TXDATA[15:8] TXCHARDISPMODE[0] corresponds to TXDATA[7:0] Table 6-5, page 130 shows how to use TXCHARDISPMODE to control the disparity of outgoing data when 8B/10B encoding is enabled.

Table 6-1: **FPGA TX Interface Ports *(Cont'd)***

Port	Direction	Clock Domain	Description
TXCHARDISPVAL0[3:0] TXCHARDISPVAL1[3:0]	In	TXUSRCLK2	TXCHARDISPVAL and TXCHARDISPMODE allow control of the 8B/10B outgoing data disparity when 8B/10B encoding is enabled. When 8B/10B encoding is disabled, TXCHARDISPVAL is used to extend the data bus for 10- and 20-bit TX interfaces. See "FPGA TX Interface," page 116. TXCHARDISPVAL[3] corresponds to TXDATA[31:24] TXCHARDISPVAL[2] corresponds to TXDATA[23:16] TXCHARDISPVAL[1] corresponds to TXDATA[15:8] TXCHARDISPVAL[0] corresponds to TXDATA[7:0] Table 6-5, page 130 shows how to use TXCHARDISPVAL to control the disparity of outgoing data when 8B/10B encoding is enabled.
TXDATA0[31:0] TXDATA1[31:0]	In	TXUSRCLK2	The bus for transmitting data. The width of this port depends on TXDATAWIDTH: • TXDATAWIDTH = 0: TXDATA[7:0] = 8 bits wide • TXDATAWIDTH = 1: TXDATA[15:0] = 16 bits wide • TXDATAWIDTH = 2: TXDATA[31:0] = 32 bits wide When a 10-bit, 20-bit, or 40-bit bus is required, the TXCHARDISPVAL and TXCHARDISPMODE ports from the 8B/10B encoder are concatenated with the TXDATA port. See Figure 6-3, page 119.
TXDATAWIDTH0[1:0] TXDATAWIDTH1[1:0]	In	TXUSRCLK2	Selects the width of the TXDATA port. • 0: TXDATA is 8 bits or 10 bits wide • 1: TXDATA is 16 bits or 20 bits wide • 2: TXDATA is 32 bits or 40 bits wide • 3: Reserved
TXENC8B10BUSE0 TXENC8B10BUSE1	In	TXUSRCLK2	TXENC8B10BUSE is set High to enable the 8B/10B encoder. INTDATAWIDTH must also be High. 0: 8B/10B encoder bypassed. This option reduces latency. 1: 8B/10B encoder enabled. INTDATAWIDTH must be High.
TXOUTCLK0 TXOUTCLK1	Out	N/A	This port provides a parallel clock generated by the GTX transceiver. This clock can be used to drive TXUSRCLK for one or more GTX transceivers. The rate of the clock depends on INTDATAWIDTH: • INTDATAWIDTH is Low: $F_{TXOUTCLK}$ = Line Rate/16 • INTDATAWIDTH is High: $F_{TXOUTCLK}$ = Line Rate/20 ***Note:*** • When INTDATAWIDTH is High, the duty cycle is 60/40 instead of 50/50. • When oversampling is enabled, the line rate in the calculation of $F_{TXOUTCLK}$ is equal to the oversampled line rate, not the PMA line rate.

Table 6-1: **FPGA TX Interface Ports *(Cont'd)***

Port	Direction	Clock Domain	Description
TXRESET0 TXRESET1	In	Async	Resets the PCS of the GTX transmitter, including the phase adjust FIFO, the 8B/10B encoder, and the FPGA TX interface.
TXUSRCLK0 TXUSRCLK1	In	N/A	Use this port to provide a clock for the internal TX PCS datapath. This clock must always be provided. The rate depends on INTDATAWIDTH: • INTDATAWIDTH is Low: $F_{TXUSRCLK}$ = Line Rate/16 • INTDATAWIDTH is High: $F_{TXUSRCLK}$ = Line Rate/20
TXUSRCLK20 TXUSRCLK21	In	N/A	Use this port to synchronize the FPGA logic with the TX interface. This clock must be positive-edge aligned to TXUSRCLK. The rate of this clock depends on $F_{TXUSRCLK}$ and TXDATAWIDTH: • TXDATAWIDTH = 0: $F_{TXUSRCLK2} = 2 \times F_{TXUSRCLK}$ • TXDATAWIDTH = 1: $F_{TXUSRCLK2} = F_{TXUSRCLK}$ • TXDATAWIDTH = 2: $F_{TXUSRCLK2} = F_{TXUSRCLK}/2$

There are no attributes in this section.

Description

The FPGA TX interface allows parallel data to be written to the GTX transceiver for transmission as serial data. To use the interface:

- The width of the data interface must be configured
- TXUSRCLK2 and TXUSRCLK must be connected to clocks running at the correct rate

Configuring the Width of the Interface

Table 6-2 shows how the interface width for the TX datapath is selected. 8B/10B encoding is discussed in more detail in "Configurable 8B/10B Encoder," page 125.

Table 6-2: **TX Datapath Width Configuration**

INTDATAWIDTH	TXDATAWIDTH	TXENC8B10BUSE	FPGA TX Interface Width
0	0	N/A	8 bits
0	1	N/A	16 bits
0	2	N/A	32 bits
1	0	0	10 bits
1	1	0	20 bits
1	2	0	40 bits
1	0	1	8 bits
1	1	1	16 bits
1	2	1	32 bits

Figure 6-2 shows how TXDATA is transmitted serially when the internal datapath is 16 bits (INTDATAWIDTH is Low) and 8B/10B encoding is disabled.

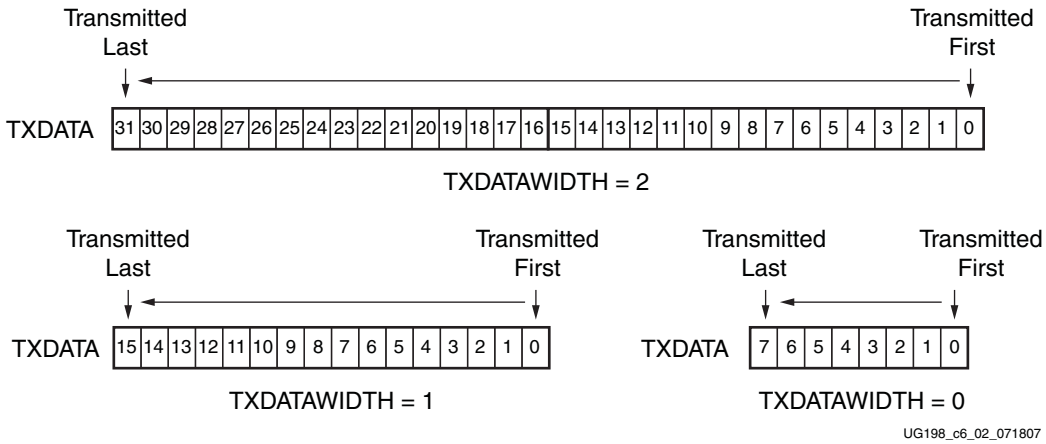


Figure 6-2: **8B/10B Bypassed, 16-Bit Internal Datapath**

Figure 6-3 and Figure 6-4 show how TXDATA is transmitted serially when the internal datapath is 20 bits (INTDATAWIDTH is High) and 8B/10B encoding is disabled. When TXDATA is 10 bits, 20 bits, or 40 bits wide, the TXCHARDISPMODE and TXCHARDISPVAL ports are taken from the 8B/10B encoder interface and used to send the extra bits.

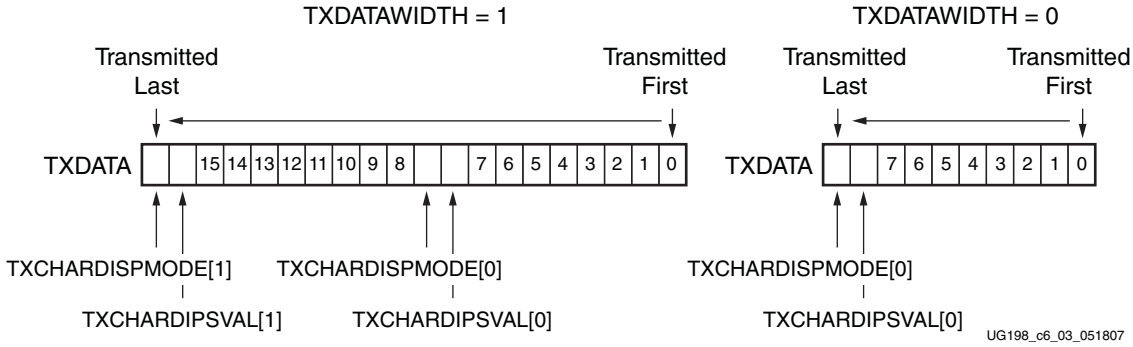


Figure 6-3: **8B/10B Bypassed, 20-Bit Internal Datapath for TXDATAWIDTH = 0 or 1**

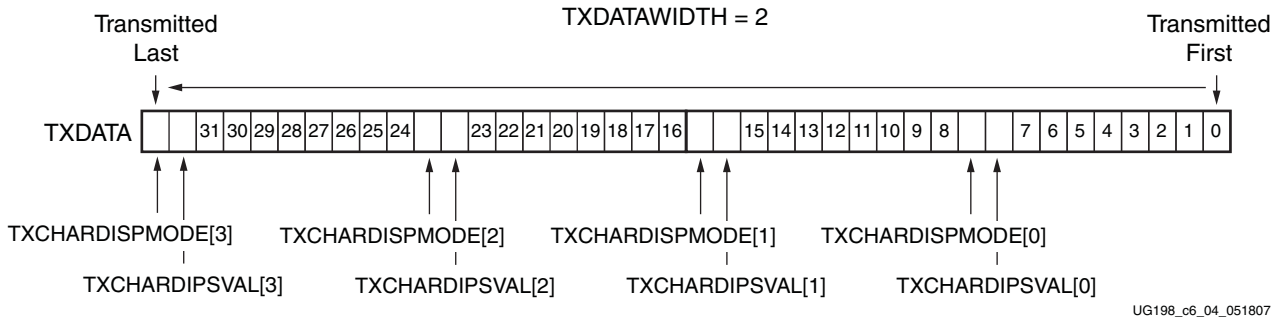


Figure 6-4: **8B/10B Bypassed, 20-Bit Internal Datapath for TXDATAWIDTH = 2**

When 8B/10B encoding is used, the width of the data interface is 8 bits, 16 bits, or 32 bits (Figure 6-2), and the data is encoded before it is transmitted serially. "Configurable 8B/10B Encoder," page 125 provides more details about bit ordering when using 8B/10B encoding.

Connecting TXUSRCLK and TXUSRCLK2

The FPGA TX interface includes two parallel clocks: TXUSRCLK and TXUSRCLK2. TXUSRCLK is the internal clock for the PCS logic in the GTX transmitter. The required rate for TXUSRCLK depends on the internal datapath width of the GTX_DUAL tile (INTDATAWIDTH), and the TX line rate of the GTX transmitter ("Parallel In to Serial Out," page 146 describes how the TX line rate is determined). Equation 6-1 shows how to calculate the required rate for TXUSRCLK.

$$\text{TXUSRCLK Rate} = \frac{\text{Line Rate}}{\text{Internal Datapath Width}} \qquad \textit{Equation 6-1}$$

TXUSRCLK2 is the main synchronization clock for all signals into the TX side of the GTX transceiver. Most signals into the TX side of the GTX transceiver are sampled on the positive edge of TXUSRCLK2. TXUSRCLK2 and TXUSRCLK have a fixed-rate relationship based on the TXDATAWIDTH setting. Equation 6-2 through Equation 6-4 show how to calculate the required rate for TXUSRCLK2 based on TXUSRCLK for TXDATAWIDTH = {0,1,2}.

$$\text{TXDATAWIDTH} = 0: F_{TXUSRCLK2} = 2 \times F_{TXUSRCLK} \qquad \textit{Equation 6-2}$$

$$\text{TXDATAWIDTH} = 1: F_{TXUSRCLK2} = F_{TXUSRCLK} \qquad \textit{Equation 6-3}$$

$$\text{TXDATAWIDTH} = 2: F_{TXUSRCLK2} = F_{TXUSRCLK}/2 \qquad \textit{Equation 6-4}$$

The following rules about the relationships between clocks must be observed for TXUSRCLK, TXUSRCLK2, and CLKIN:

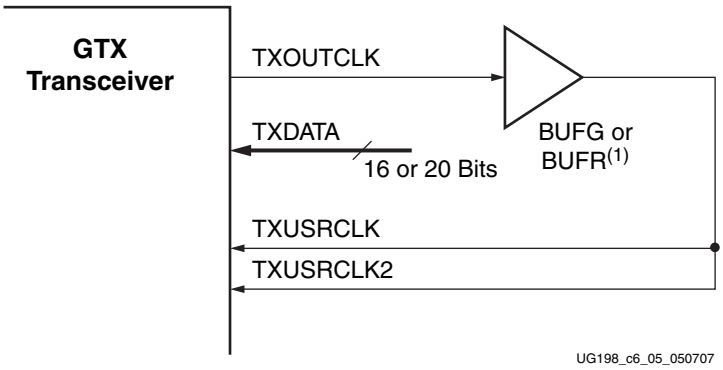
- TXUSRCLK and TXUSRCLK2 must be positive-edge aligned, with as little skew as possible between them. As a result, low-skew clock resources (BUFGs and BUFRs) should be used to drive TXUSRCLK and TXUSRCLK2. When TXUSRCLK and TXUSRCLK2 have the same frequency, the same clock resource is used to drive both. When the two clocks have different frequencies, TXUSRCLK is used to derive TXUSRCLK2 through the multiplication or division of TXUSRCLK. The designer must ensure that the two are positive-edge aligned. The "Examples" section shows various clock configurations that meet this requirement.
- Even though they might run at different frequencies, TXUSRCLK, TXUSRCLK2, and CLKIN must have the same oscillator as their source. Thus TXUSRCLK and TXUSRCLK2 must be multiplied or divided versions of CLKIN. The GTX transceiver provides access to CLKIN in two ways: the REFCLKOUT pin (shared by both GTX transceivers in the GTX_DUAL tile) and the TXOUTCLK pin. The "Examples" section shows several clock configurations with each pin.
- REFCLKOUT is the same as CLKIN. It is free-running and operates even before the shared PMA PLL is locked. However, because REFCLKOUT uses the CLKIN rate, it can require multiplication and division to produce the required rates for TXUSRCLK and TXUSRCLK2.
- TXOUTCLK provides a copy of CLKIN already divided to the TXUSRCLK rate, potentially requiring fewer dividers. However, TXOUTCLK is not free-running. It is only valid after the shared PMA PLL is locked and cannot be used when TX phase alignment is turned on (see "TX Buffering, Phase Alignment, and TX Skew Reduction," page 138).

Examples

Figure 6-5 through Figure 6-9 show different ways FPGA clock resources can be used to drive the parallel clocks for the TX interface.

TXOUTCLK Driving a GTX TX in 2-Byte Mode

In Figure 6-5, TXOUTCLK is used to drive TXUSRCLK and TXUSRCLK2 for 2-byte mode (TXDATAWIDTH = 1).



Notes:

1. Refer to the *Virtex-5 FPGA Data Sheet* and the *Virtex-5 FPGA Configuration Guide* for the maximum clock frequency and jitter limitations of BUFR.

Figure 6-5: **TXOUTCLK Drives TXUSRCLK and TXUSRCLK2 (2-Byte Mode)**

TXOUTCLK Driving GTX TX in 4-Byte Mode

The example in Figure 6-6 uses 4-byte wide datapaths (TXDATAWIDTH = 2). TXOUTCLK drives TXUSRCLK, and TXOUTCLK is divided by two using a DCM or PLL to drive TXUSRCLK2.

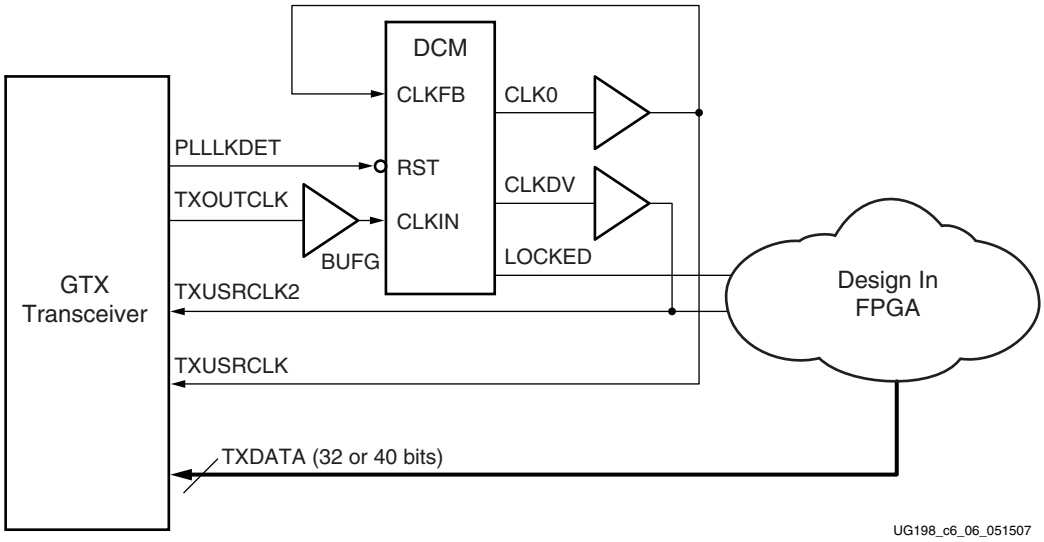


Figure 6-6: **DCM Provides Clocks for 4-Byte Wide Datapath**

TXOUTCLK Driving a GTX TX in 1-Byte Mode

In Figure 6-7, a one-byte datapath is used (TXDATAWIDTH = 0). TXOUTCLK drives TXUSRCLK. TXOUTCLK is also multiplied by two using a PLL to drive TXUSRCLK2. Because the internal datapath of the GTX_DUAL tile is two bytes wide (16 or 20 bits) and the datapath to the FPGA logic in this configuration is only one byte wide (8 or 10 bits), the TXUSRCLK2 must be twice as fast as TXUSRCLK.

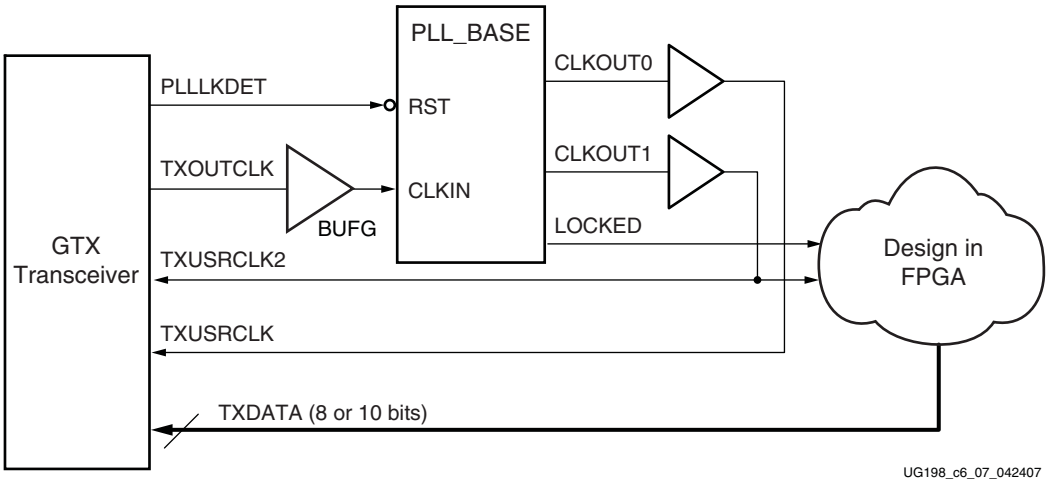


Figure 6-7: **PLL Provides Clocks for 1-Byte Datapath**

TXOUTCLK Driving Multiple Transceivers for a 4-Byte Interface

Figure 6-8 shows TXOUTCLK driving multiple GTX user clocks. In this situation, the frequency must be correct for all GTX transceivers, and they must share the same reference clock. In Figure 6-8, because the top GTX transceiver uses a 4-byte interface, it requires a divide-by-2 clock for TXUSRCLK2.

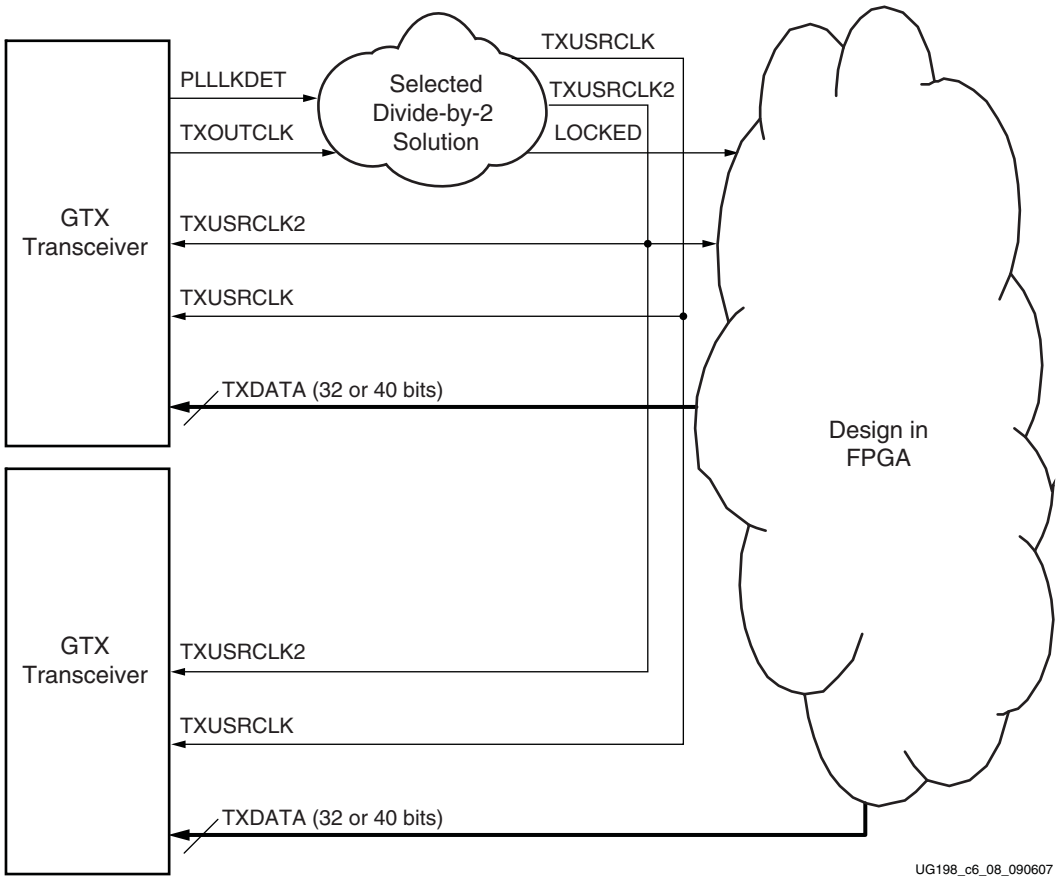


Figure 6-8: **TXOUTCLK Drives Multiple GTX Transceivers with a 4-Byte Interface**

REFCLKOUT Driving Multiple Transceivers with a 4-Byte Interface

Figure 6-9 shows how REFCLKOUT can be used to generate USRCLK signals. REFCLKOUT runs continuously, even when the GTX_DUAL tile is reset. However, extra clocking resources can be used to generate the correct USRCLK frequency. In Figure 6-9, a PLL is used to generate the TXUSRCLK and TXUSRCLK2 frequencies from REFCLKOUT. A DCM can be used instead of the PLL, but the PLL is more convenient when the REFCLKOUT rate is not an integer multiple of the required TXUSRCLK rates.

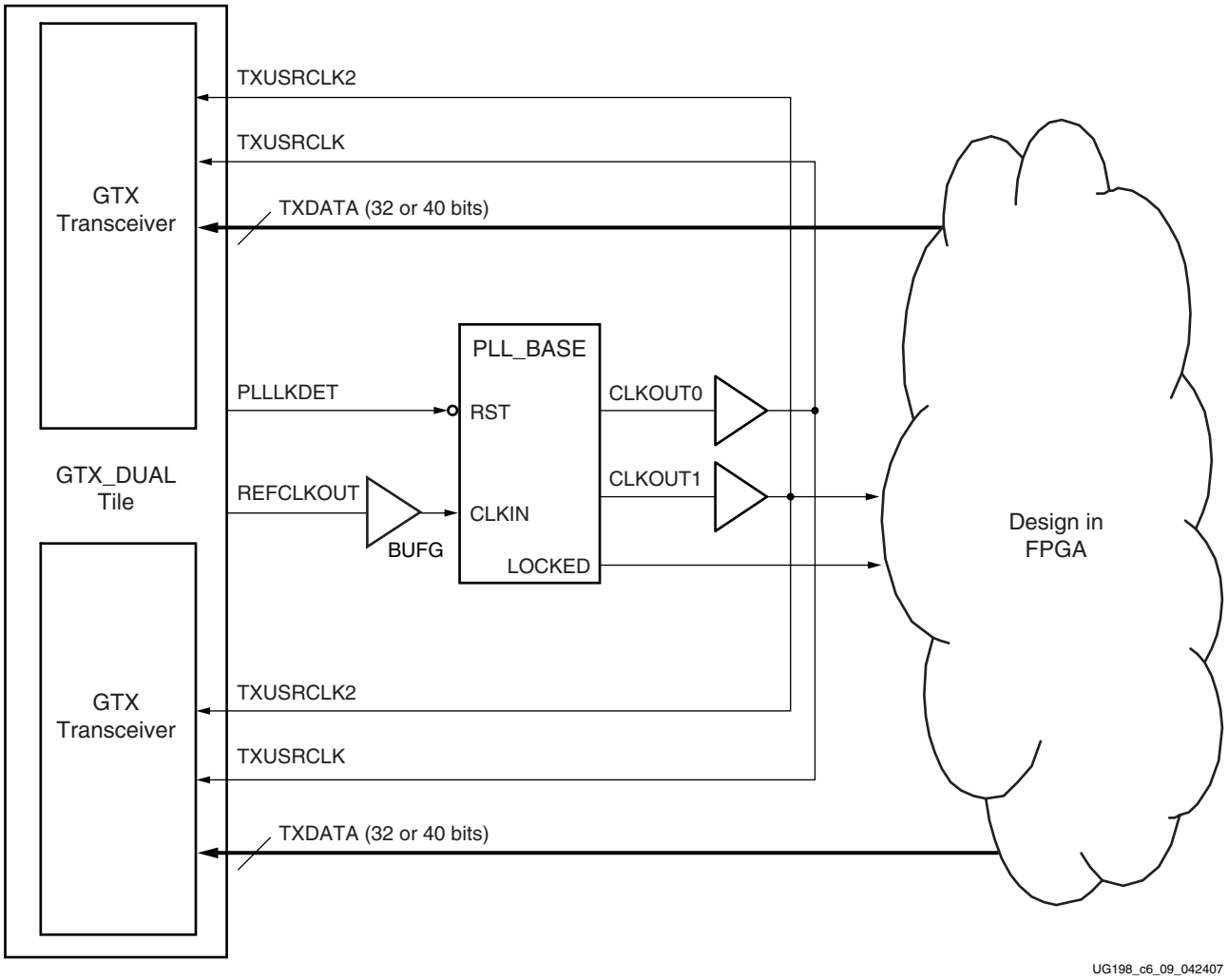


Figure 6-9: **REFCLKOUT Driving Multiple GTX Transceivers in 4-Byte Mode**

Configurable 8B/10B Encoder

Overview

Many protocols use 8B/10B encoding on outgoing data. 8B/10B is an industry-standard encoding scheme that trades two bits of overhead per byte for improved performance. Table 6-3 outlines the benefits and costs of 8B/10B. Appendix C shows how 8-bit values are mapped to 10-bit data and control sequences in 8B/10B.

Table 6-3: **8B/10B Trade-Offs**

8B/10B Benefits	8B/10B Costs
DC Balanced: No increase in bit errors due to line charging on AC-coupled channels.	Two-bit overhead per byte: Every byte transmitted is mapped to a 10-bit character. As a result, 20% of the channel bandwidth is consumed for overhead.
Limited Run Lengths: The maximum number of bits without a transition is 5, making it easy for receivers to achieve and maintain lock.	Both sides of the channel must use 8B/10B: 8B/10B data must be decoded before it can be used.
Error Detection: All single-bit errors and many multi-bit errors can be detected using disparity and out-of-table error checking.	
Control Characters: 8B/10B allows bytes to be marked as control characters. This feature is heavily used in many standard protocols.	

The GTX transceiver includes an 8B/10B encoder to encode TX data without consuming FPGA resources. If encoding is not needed, the block can be disabled to minimize latency.

Ports and Attributes

Table 6-4 defines the TX encoder ports.

Table 6-4: **TX Encoder Ports**

Port	Direction	Clock Domain	Description
TXBYPASS8B10B0[3:0] TXBYPASS8B10B1[3:0]	In	TXUSRCLK2	TXBYPASS8B10B controls the operation of the TX 8B/10B encoder on a per-byte basis. It is only effective when both TXENC8B10B and INTDATAWIDTH are High (8B/10B is enabled). TXBYPASS8B10B[3] corresponds to TXDATA[31:24] TXBYPASS8B10B[2] corresponds to TXDATA[23:16] TXBYPASS8B10B[1] corresponds to TXDATA[15:8] TXBYPASS8B10B[0] corresponds to TXDATA[7:0] TXBYPASS8B10B[*x*] = 1, encoder for byte *x* is bypassed TXBYPASS8B10B[*x*] = 0, encoder for byte *x* is used
TXCHARDISPMODE0[3:0] TXCHARDISPMODE1[3:0]	In	TXUSRCLK2	TXCHARDISPMODE and TXCHARDISPVAL allow the 8B/10B disparity of outgoing data to be controlled when 8B/10B encoding is enabled. When 8B/10B encoding is disabled, TXCHARDISPMODE is used to extend the data bus for TX interfaces with a width that is a multiple of 10. TXCHARDISPMODE[3] corresponds to TXDATA[31:24] TXCHARDISPMODE[2] corresponds to TXDATA[23:16] TXCHARDISPMODE[1] corresponds to TXDATA[15:8] TXCHARDISPMODE[0] corresponds to TXDATA[7:0] Table 6-5, page 130 shows how TXCHARDISPMODE is used to control the disparity of outgoing data when 8B/10B encoding is enabled.
TXCHARDISPVAL0[3:0] TXCHARDISPVAL1[3:0]	In	TXUSRCLK2	TXCHARDISPVAL and TXCHARDISPMODE allow the 8B/10B disparity of outgoing data disparity to be controlled when 8B/10B encoding is enabled. When 8B/10B encoding is disabled, TXCHARDISPVAL is used to extend the data bus for 10- and 20-bit TX interfaces (see "FPGA TX Interface," page 116). TXCHARDISPVAL[3] corresponds to TXDATA[31:24] TXCHARDISPVAL[2] corresponds to TXDATA[23:16] TXCHARDISPVAL[1] corresponds to TXDATA[15:8] TXCHARDISPVAL[0] corresponds to TXDATA[7:0] Table 6-5, page 130 shows how TXCHARDISPVAL is used to control the disparity of outgoing data when 8B/10B encoding is enabled.

Table 6-4: **TX Encoder Ports *(Cont'd)***

Port	Direction	Clock Domain	Description
TXCHARISK0[3:0] TXCHARISK1[3:0]	In	TXUSRCLK2	TXCHARISK is set High to send TXDATA as an 8B/10B K character. TXCHARISK should only be asserted for TXDATA values in the K-character table of the 8B/10B table in Appendix C, "8B/10B Valid Characters." TXCHARISK[3] corresponds to TXDATA[31:24] TXCHARISK[2] corresponds to TXDATA[23:16] TXCHARISK[1] corresponds to TXDATA[15:8] TXCHARISK[0] corresponds to TXDATA[7:0] TXCHARISK is undefined for bytes that bypass 8B/10B encoding.
TXENC8B10BUSE0 TXENC8B10BUSE1	In	TXUSRCLK2	TXENC8B10BUSE is set High to enable the 8B/10B encoder. INTDATAWIDTH must also be High. 0: 8B/10B encoder bypassed. This option reduces latency. 1: 8B/10B encoder enabled. INTDATAWIDTH must be 1.
TXKERR0[3:0] TXKERR1[3:0]	Out	TXUSRCLK2	TXKERR indicates if an invalid code for a K character was specified. For 1-byte interfaces: TXKERR[0] corresponds to TXDATA[7:0] For 2-byte interfaces: TXKERR[0] corresponds to TXDATA[15:8] TXKERR[1] corresponds to TXDATA[7:0] For 4-byte interfaces: TXKERR[3] corresponds to TXDATA[15:8] TXKERR[2] corresponds to TXDATA[7:0] TXKERR[1] corresponds to TXDATA[31:24] TXKERR[0] corresponds to TXDATA[23:16]
TXRUNDISP0[3:0] TXRUNDISP1[3:0]	Out	TXUSRCLK2	TXRUNDISP indicates the current running disparity of the 8B/10B encoder. This disparity corresponds to TXDATA clocked in several cycles earlier. For 1-byte interfaces: TXRUNDISP[0] corresponds to previous TXDATA[7:0] data For 2-byte interfaces: TXRUNDISP[0] corresponds to previous TXDATA[15:8] data TXRUNDISP[1] corresponds to previous TXDATA[7:0] data For 4-byte interfaces: TXRUNDISP[3] corresponds to previous TXDATA[15:8] data TXRUNDISP[2] corresponds to previous TXDATA[7:0] data TXRUNDISP[1] corresponds to previous TXDATA[31:24] data TXRUNDISP[0] corresponds to previous TXDATA[23:16] data

There are no attributes in this section.

Description

Enabling 8B/10B Encoding

To disable the 8B/10B encoder on a given GTX transceiver, TXENC8B10BUSE must be driven Low. To enable the 8B/10B encoder, TXENC8B10BUSE must be driven High. When the encoder is turned off, the operation of the TXDATA port is as described in "FPGA TX Interface."

8B/10B Bit and Byte Ordering

The order of the bits after the 8B/10B encoder is the opposite of the order shown in Appendix C, "8B/10B Valid Characters," because 8B/10B encoding requires bit a0 to be transmitted first, and the GTX transceiver always transmits the right-most bit first. To match with 8B/10B, the 8B/10B encoder in the GTX transceiver automatically reverses the bit order (Figure 6-10).

For the same reason, when a 2-byte interface is used, the first byte to be transmitted (byte 0) must be placed on TXDATA[7:0], and the second placed on TXDATA[15:8]. When a 4-byte interface is used, byte 0 must be placed on TXDATA[7:0], byte 1 must be placed on TXDATA[15:8], byte 2 must be placed on TXDATA[23:16], and byte 3 must be placed on TXDATA[31:24]. This placement ensures that the byte 0 bits are all sent before the byte 1 bits, as required by 8B/10B encoding.

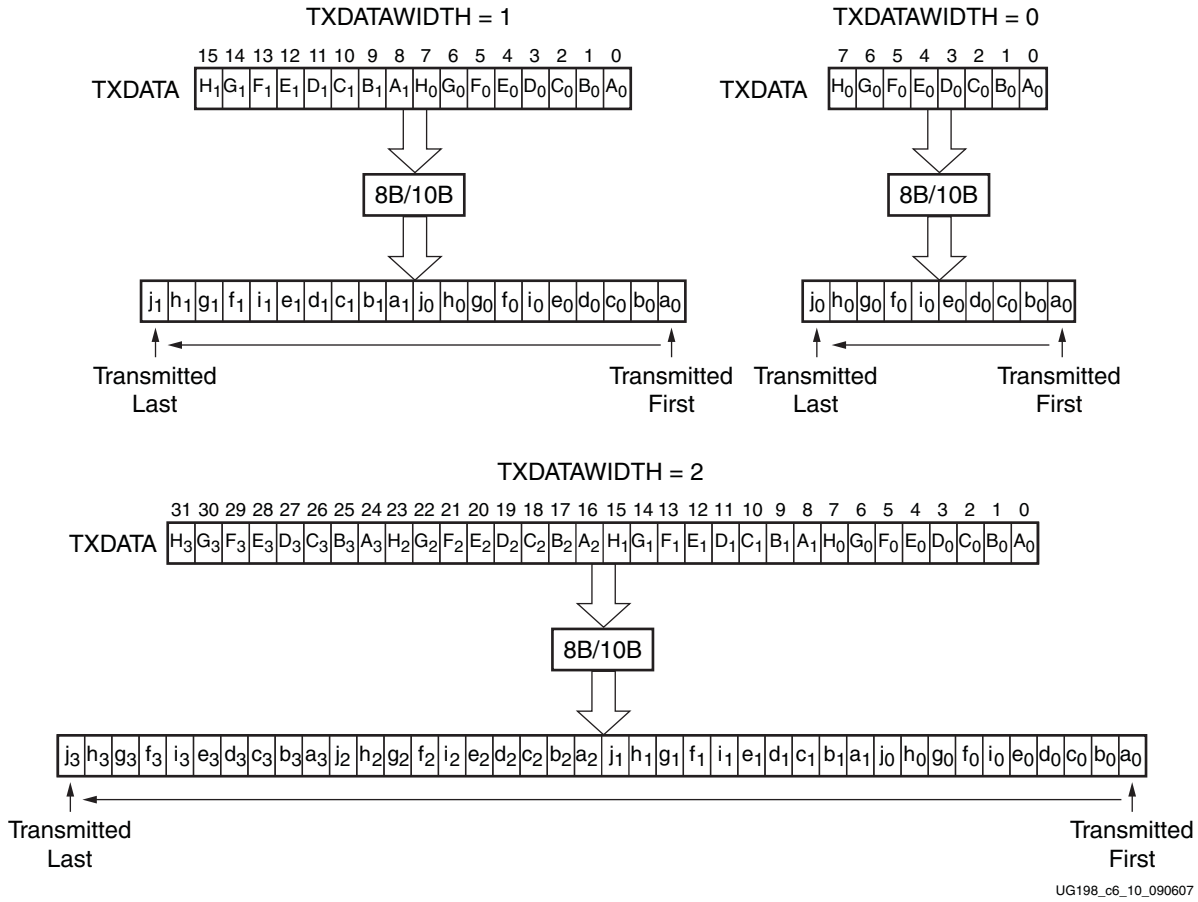


Figure 6-10: **8B/10B Encoding**

K Characters

The 8B/10B table includes special characters (K characters) that are often used for control functions. To transmit TXDATA as a K character instead of regular data, the TXCHARISK port must be driven High. If TXDATA is not a valid K character, the encoder drives TXKERR High.

Running Disparity

8B/10B uses running disparity to balance the number of ones and zeros transmitted. Whenever a character is transmitted, the encoder recalculates the running disparity. The current TX running disparity can be read from the TXCHARDISP port. This running disparity is calculated several cycles after the TXDATA is clocked into the FPGA TX interface, so it cannot be used to decide the next value to send, as required in some protocols.

Normally, running disparity is used to determine whether a positive or negative 10-bit code is transmitted next. The encoder allows the next disparity value to be controlled directly as well, to accommodate protocols that use disparity to send control information. For example, an Idle character sent with reversed disparity might be used to trigger clock correction. Table 6-5 shows how the TXCHARDISPMODE and TXCHARDISPVAL ports are used to control outgoing disparity values.

Table 6-5: **TXCHARDISPMODE and TXCHARDISPVAL vs. Outgoing Disparity**

TXCHARDISPMODE	TXCHARDISPVAL	Outgoing Disparity
0	0	Calculated normally by the 8B/10B encoder
0	1	Inverts normal running disparity when encoding TXDATA
1	0	Forces running disparity negative when encoding TXDATA
1	1	Forces running disparity positive when encoding TXDATA

8B/10B Bypass

The encoder offers total control of outgoing data using the TXBYPASS8B10B signal. To bypass the 8B/10B encoding and write the outgoing 10-bit code directly, TXBYPASS8B10B must be driven High. When TXBYPASS8B10B is High, the TX interface for the byte is the same as in Figure 6-2, page 119. Bypassing the encoder using TXBYPASS8B10B does not reduce latency, but it does allow each byte of the TX interface to be bypassed individually on a cycle-by-cycle basis.

TX Gearbox

Overview

Some high-speed data rate protocols use 64B/66B encoding to reduce the overhead of 8B/10B encoding while retaining the benefits of an encoding scheme. The TX Gearbox provides support for 64B/66B and 64B/67B header and payload combining. The Interlaken interface protocol specification uses the 64B/67B encoding scheme. Refer to the Interlaken specification for further information. The Interlaken specification can be downloaded from: http://www.cortina-systems.com/news/interlaken. The TX Gearbox only supports 2-byte and 4-byte interfaces. A 1-byte interface is not supported.

Scrambling of the data is done in the FPGA logic. The RocketIO GTX Transceiver Wizard has example code for the scrambler.

Ports and Attributes

Table 6-6 defines the TX Gearbox ports.

Table 6-6: **TX Gearbox Ports**

Port	Direction	Clock Domain	Description
TXGEARBOXREADY0 TXGEARBOXREADY1	Out	TXUSRCLK2	Output indicating how data is applied to TX Gearbox. 0: No data can be applied 1: Data must be applied Use with attributes GEARBOX_ENCDEC_0 and GEARBOX_ENCDEC_1.
TXHEADER0[2:0] TXHEADER1[2:0]	In	TXUSRCLK2	Input for header bits. Bit[2]: Indicates data inverted for 64B/67B encoding Bit[1:0]: The encoding for these bits is: 01: Data header 10: Control header
TXSEQUENCE0[6:0] TXSEQUENCE1[6:0]	In	TXUSRCLK2	Input from 7-bit/6-bit counter in FPGA logic to the TX Gearbox. Time to data for 64B/67B and 64B/66B.
TXSTARTSEQ0 TXSTARTSEQ1	In	TXUSRCLK2	Input to TX Gearbox indicating the first character in the data sequence for 64B/67B and 64B/66B encoding.

Table 6-7 defines the TX Gearbox attributes.

Table 6-7: **TX Gearbox Attributes**

Attribute	Type	Description
GEARBOX_ENDEC_0[2:0] GEARBOX_ENDEC_1[2:0]	3-bit Binary	Indicates TX Gearbox modes: Bit [2]: Always set to 0 to enable the Gearbox decoder Bit [1]: The encoding for this bit is: 0: Use external sequence counter and apply inputs to TXSEQUENCE 1: Use internal sequence counter, gate input header, and data with output TXGEARBOXREADY0 and TXGEARBOXREADY1 Bit [0]: The encoding for this bit is: 0: 64B/67B Gearbox mode for Interlaken 1: 64B/66B Gearbox
TXGEARBOX_USE_0 TXGEARBOX_USE_1	Boolean	When TXGEARBOX_USE = TRUE, TX Gearbox is enabled. The TX Gearbox only supports 2-byte and 4-byte logic interfaces. When using the internal TX Gearbox, a 1-byte logic interface is not supported. A 1-byte logic interface can be implemented by using an external Gearbox in the FPGA logic. INTDATAWIDTH must be 0 (16-bit internal data width mode) when the TX Gearbox is enabled.

Description

Enabling the TX Gearbox

To enable the TX Gearbox for the GTX transceiver 0, set the attribute TXGEARBOX_USE_0 to TRUE. To enable the TX Gearbox for the GTX transceiver 1, set the attribute TXGEARBOX_USE_1 to TRUE.

Bit 2 of the GEARBOX_ENDEC attribute must be set to 0 to enable the Gearbox decoder. The decoder controls the GTX transceiver's TX Gearbox and RX Gearbox. The GTX transceiver's TX Gearbox and RX Gearbox use the same mode.

TX Gearbox Bit and Byte Ordering

Figure 6-11 shows an example of the first five cycles of data entering and data exiting the TX Gearbox for 64B/66B encoding when using a two-byte logic interface. The input consists of a 2-bit header and 16 bits of data. On the first cycle, the header and 14 bits of data exit the TX Gearbox. On the second cycle, the remaining two data bits from the previous cycle's TXDATA input along with 14 data bits from the current TXDATA input exit the TX Gearbox. This continues for the third and fourth cycle. On the fifth cycle, the output of the TX Gearbox contains two remaining data bits from the first 66-bit block, the header of the second 66-bit block, and 12 data bits from the second 66-bit block. As shown in Figure 6-11, the header bits are serialized first followed by the data bits.

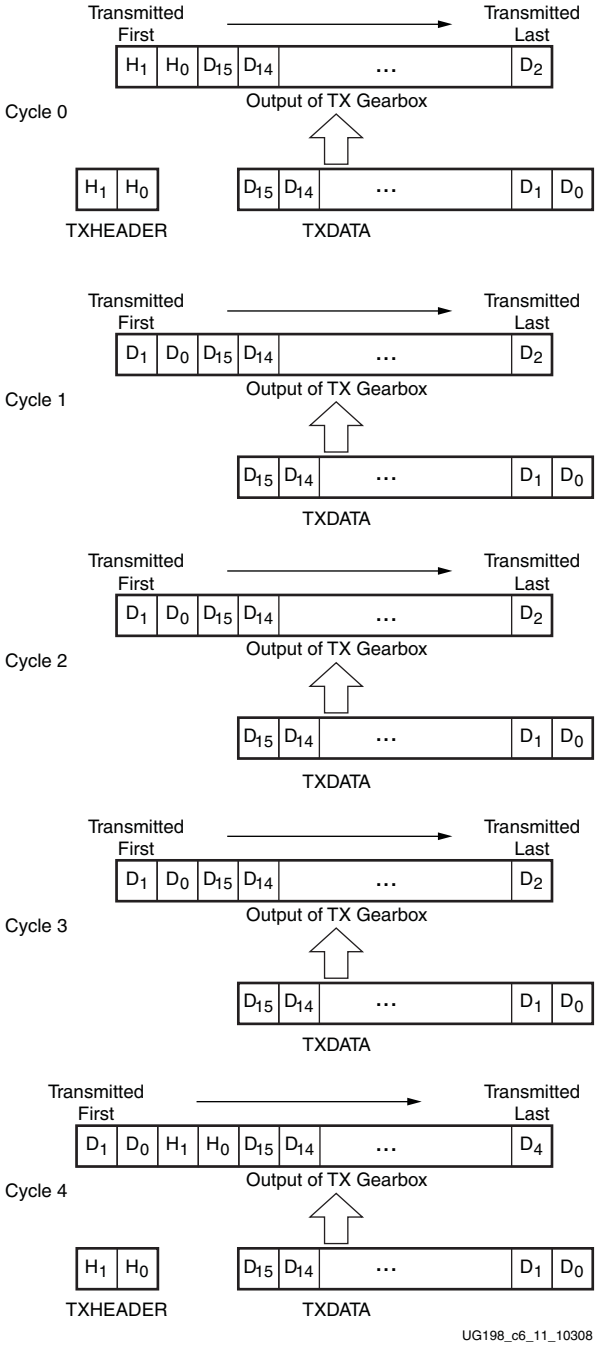


Figure 6-11: **TX Gearbox Bit Ordering**

TX Gearbox Operating Modes

The TX Gearbox has two operating modes. The external sequence counter operating mode must be implemented in user logic. The second mode uses an internal sequence counter. The TX Gearbox only supports 2-byte and 4-byte interfaces to the FPGA logic.

External Sequence Counter Operating Mode

As shown in Figure 6-12, the external sequence counter operating mode uses the TXSEQUENCE(0/1)[6:0], TXDATA(0/1)[31:0], and TXHEADER(0/1)[2:0] inputs. A binary counter must exist in the user logic to drive the TXSEQUENCE(0/1) input port. For 64B/66B encoding, the counter increments from 0 to 32 and repeats from 0. For 64B/67B encoding, the counter increments from 0 to 66 and repeats from 0. When using 64B/66B encoding, tie TXSQUENCE(0/1)[6] to a logic 0 and tie the unused TXHEADER(0/1)[2] to a logic 0. The sequence counter increment ranges ({0 to 32}, {0 to 66}) are identical for both the 2-byte and 4-byte interfaces. However, the counter must increment once every two TXUSRCLK2 cycles when using a 2-byte interface and every TXUSRCLK2 cycle when using a 4-byte interface.

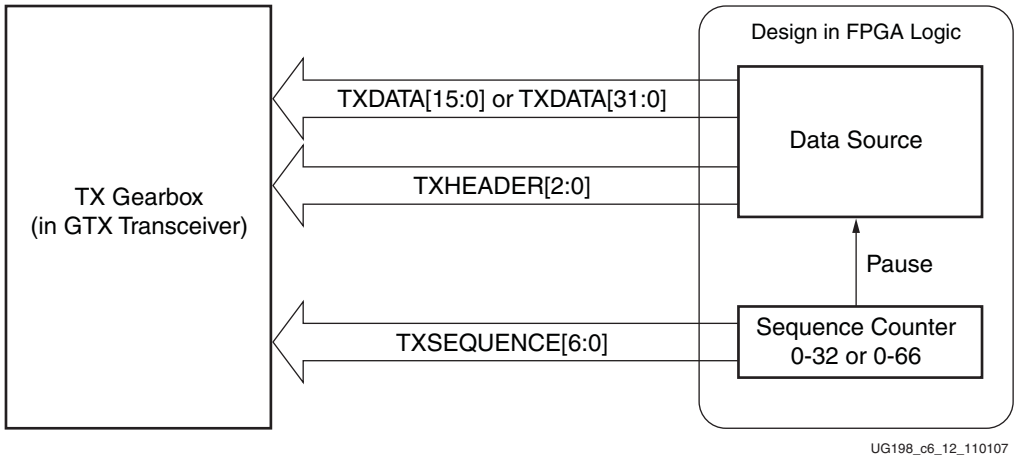


Figure 6-12: **TX Gearbox in External Sequence Counter Mode**

Due to the nature of the 64B/66B and 64B/67B encoding schemes, user data is held (paused) during various sequence counter values. Data is then paused for two TXUSRCLK2 cycles in 2-byte mode and for one TXUSRCLK2 cycle in 4-byte mode. Valid data transfer is resumed on the next TXUSRCLK2 cycle. The data pause only applies to TXDATA(0/1) and not to TXHEADER(0/1).

- 64B/67B encoding: data is held (paused) for sequence counter values of 21, 44, and 65.
- 64B/66B encoding, data is held (paused) at counter value 31.

Figure 6-13 shows how a pause occurs at counter value 31 when using a 4-byte interface, external sequence counter mode, and 64B/66B encoding.

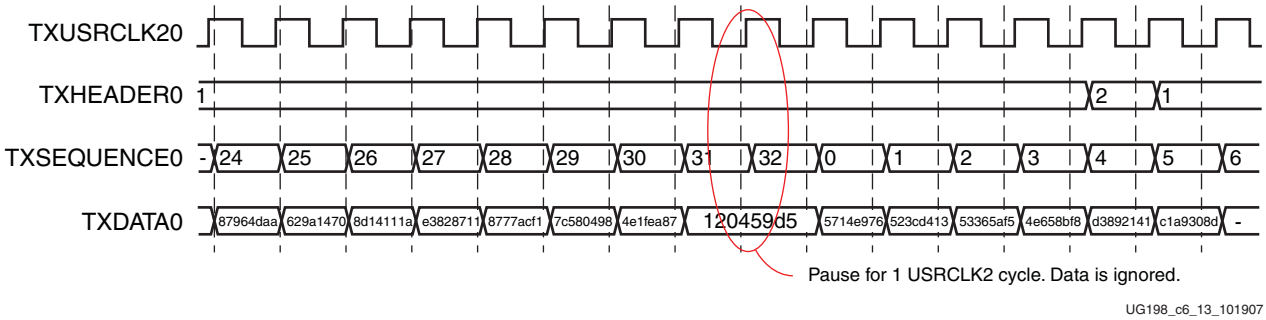


Figure 6-13: **Pause at Sequence Counter Value 31**

Figure 6-14 shows how a pause occurs at counter value 44 when using a 2-byte interface, external sequence counter mode, and 64B/67B encoding.

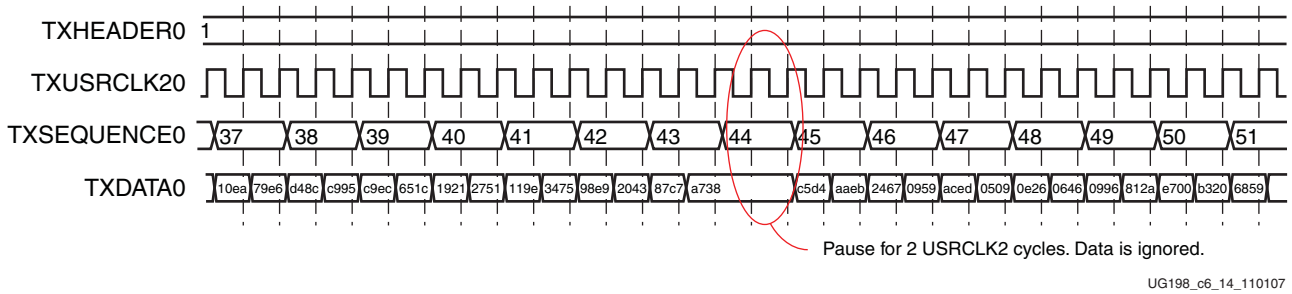


Figure 6-14: **Pause at Sequence Counter Value 44**

The sequence of transmitting 64B/67B data for the external sequence counter mode is:

1. Assert TXRESET and wait until the reset cycle is completed.
2. During reset, drive `7'h00` on TXSEQUENCE, header information on TXHEADER[2:0], and initial data on TXDATA. This state can be held indefinitely in readiness for data transmission.
3. On count 0, drive data to TXDATA and header information to TXHEADER. For a 2-byte interface, drive a second 2 bytes to TXDATA while still on count 0.
4. The sequence counter increments to 1 while driving data on TXDATA.
5. After applying 4 bytes of data, the counter increments to 2 and drives data on TXDATA and header information on TXHEADER[2:0].
6. On count 21: stop the data pipeline.
7. On count 22: drive data on TXDATA.
8. On count 44: stop the data pipeline.
9. On count 45: drive data on TXDATA.
10. On count 65: stop the data pipeline.
11. On count 66: drive data on TXDATA.

The sequence of transmitting 64B/66B data for the external sequence counter mode is:

1. Assert TXRESET and wait until the reset cycle is completed.
2. During reset, drive `6'h00` on TXSEQUENCE, header information on TXHEADER[1:0], and initial data on TXDATA. This state can be held indefinitely in readiness for data transmission.
3. On count 0, drive data to TXDATA and header information to TXHEADER[1:0]. For a 2-byte interface, drive a second 2 bytes to TXDATA while still on count 0.
4. The sequence counter increments to 1 while driving data on TXDATA.
5. After applying 4 bytes of data, the counter increments to 2 and drives data on TXDATA and header information on TXHEADER.
6. On count 31: stop the data pipeline.
7. On count 32: drive data on TXDATA.

Internal Sequence Counter Operating Mode

As shown in Figure 6-15, the internal sequence counter operating mode uses the TXSTARTSEQ input and the TXGEARBOXREADY output in addition to the TXDATA data

inputs and the TXHEADER header inputs. In this use model, the TXSEQUENCE inputs are not used. The use model is similar to the previous use model except that the TXGEARBOXREADY output is not used.

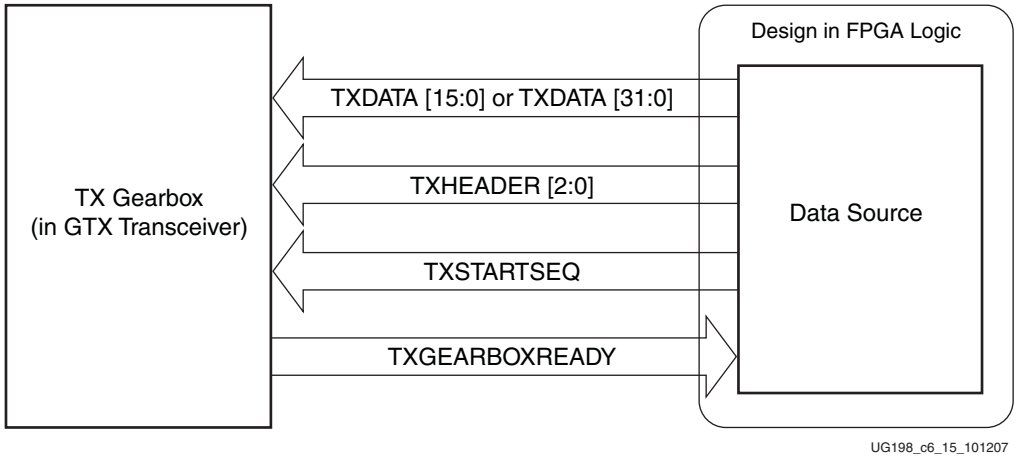


Figure 6-15: **TX Gearbox in Internal Sequence Counter Mode**

The TXSTARTSEQ input indicates to the TX Gearbox when the first byte of data after a reset is valid. TXSTARTSEQ is asserted High when the first byte of valid data is applied after a reset condition. The TXDATA and TXHEADER inputs must be held stable after reset, and TXSTARTSEQ must be held Low until data can be applied continuously.

There are no requirements on how long a user can wait before starting to transmit data. TXSTARTSEQ is asserted High along with the first two bytes/four bytes of valid data and not before. After the first bytes of data, TXSTARTSEQ can be held at any value that is convenient.

After data is driven, TXGEARBOXREADY is deasserted Low for either two TXUSRCLK2 cycles (4-byte mode) or three TXUSRCLK2 cycles (2-byte mode). Figure 6-16 and Figure 6-17 show the behavior of TXGEARBOXREADY for a 4-byte interface and a 2-byte interface, respectively. When TXGEARBOXREADY is deasserted Low, only one TXUSRCLK2 cycle remains before the data pipe must be stopped. The one-cycle latency is fixed and cannot be changed. After one cycle of latency, data must be held through 4 bytes (one TXUSRCLK2 cycle for 4-byte mode or two TXUSRCLK2 cycles for 2-byte mode) and then data is continued to be driven. Only data must be held. TXGEARBOXREADY transitions High on the cycle where new data must be driven. For this mode of operation, the number of hold points is identical to when using the external sequence counter mode for 64B/67B and 64B/66B.

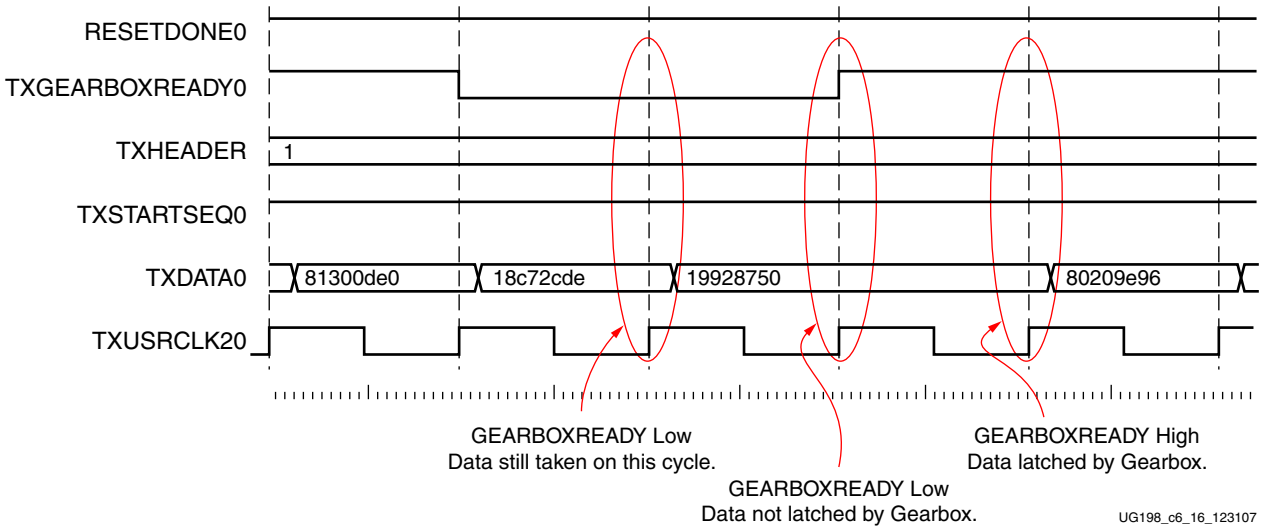


Figure 6-16: **TX Gearbox Internal Sequence Mode, 4-Byte Interface, 64B/67B**

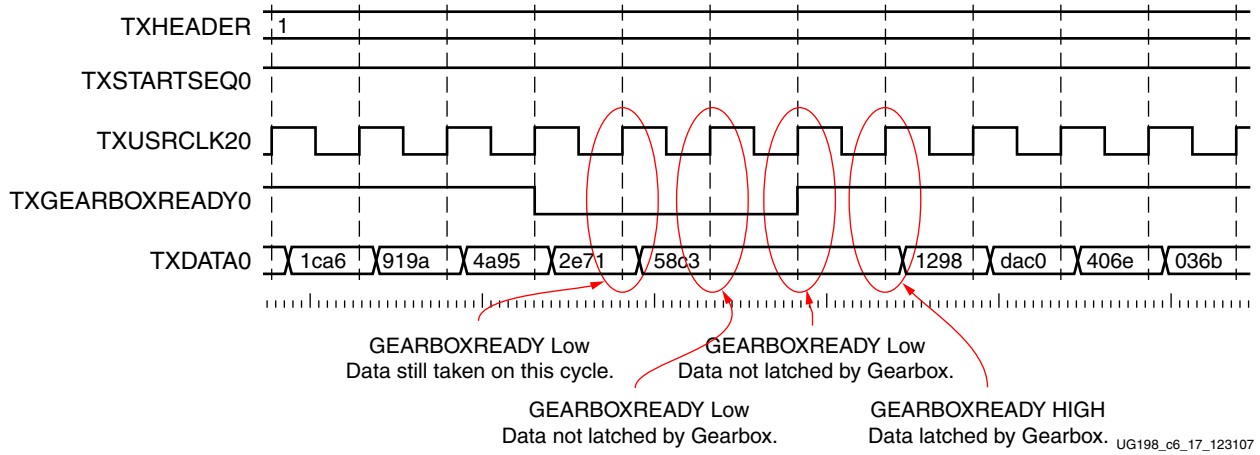


Figure 6-17: **TX Gearbox Internal Sequence Mode, 2-Byte Interface, 64B/66B**

The sequence of transmitting data for the internal sequence counter mode is:

1. Hold TXSTARTSEQ Low.
2. Assert TXRESET and wait until the reset cycle is completed.
3. TXGEARBOXREADY goes High.
4. During reset, place the appropriate header data on TXHEADER and the initial data on TXDATA. This state can be held indefinitely in readiness for data transmission.
5. Drive TXSTARTSEQ High and place the first valid header information on TXHEADER and data on TXDATA.
6. Continue to drive header information and data until TXGEARBOXREADY goes Low.
7. When TXGEARBOXREADY goes Low, drive the last 2 (or 4) bytes of data and the header information (4-byte input mode).
8. Hold the data pipeline for four bytes of data (one TXUSRCLK2 cycle for a 4-byte input or two TXUSRCLK2 cycles for a 2-byte input).
9. On the next TXUSRCLK2 cycle, drive data on the TXDATA inputs. TXGEARBOXREADY is asserted High on the previous TXUSRCLK2 cycle.

TX Buffering, Phase Alignment, and TX Skew Reduction

Overview

The GTX TX datapath has two internal parallel clock domains used in the PCS: the PMA parallel clock domain (XCLK) and the TXUSRCLK domain. To transmit data, the XCLK rate must match the TXUSRCLK rate, and all phase differences between the two domains must be resolved. Figure 6-18 shows the XCLK and USRCLK domains.

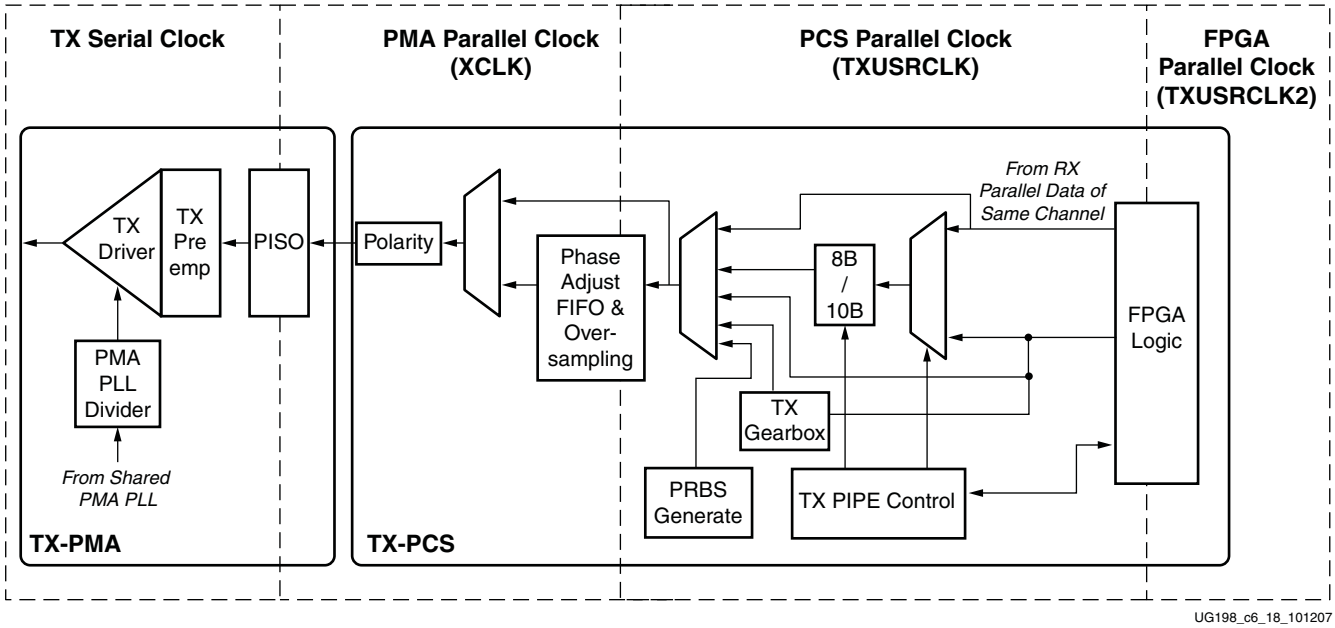


Figure 6-18: **Clock Domains and Alignment Logic**

The GTX transmitter includes a TX buffer and a TX phase-alignment circuit to resolve phase differences between the PMACLK and TXUSRCLK domains. All TX datapaths must use these circuits. Table 6-8 shows trade-offs between buffering and phase alignment.

Table 6-8: **Buffering and Phase-Alignment Trade-Offs**

	TX Buffer	TX Phase Alignment
Ease of Use	The TX buffer is used when possible. It is robust and easy to operate.	Phase alignment requires extra logic and additional constraints on clock sources. TXOUTCLK cannot be used.
Latency	If low latency is critical, the TX buffer must be bypassed.(1)	Phase alignment uses fewer registers in the datapath.
Skew Reduction	The TX buffer is required for skew reduction.	The phase-alignment circuit can be used to reduce the skew between separate GTX transceivers. All GTX transceivers involved must use the same line rate.
Oversampling	The TX buffer is required for oversampling.	

Notes:

1. Bypassing the TX buffer is an advanced feature and is not recommended for normal operation. TX buffer bypass operation can be guaranteed only under certain system-level conditions and data rates.

The TX phase-alignment circuit can also be used to minimize skew between GTX transceivers. Figure 6-19 shows how the phase-alignment circuit can reduce lane skew by aligning the PMACLK domains of multiple GTX transceivers to a common clock.

Figure 6-19 shows multiple lanes running before and after phase alignment to a common clock. Before phase alignment, all PMACLKs have an arbitrary phase difference, but after alignment, the only phase difference is the skew for the common clock, and all data is transmitted simultaneously as long as the datapath latency is matched.

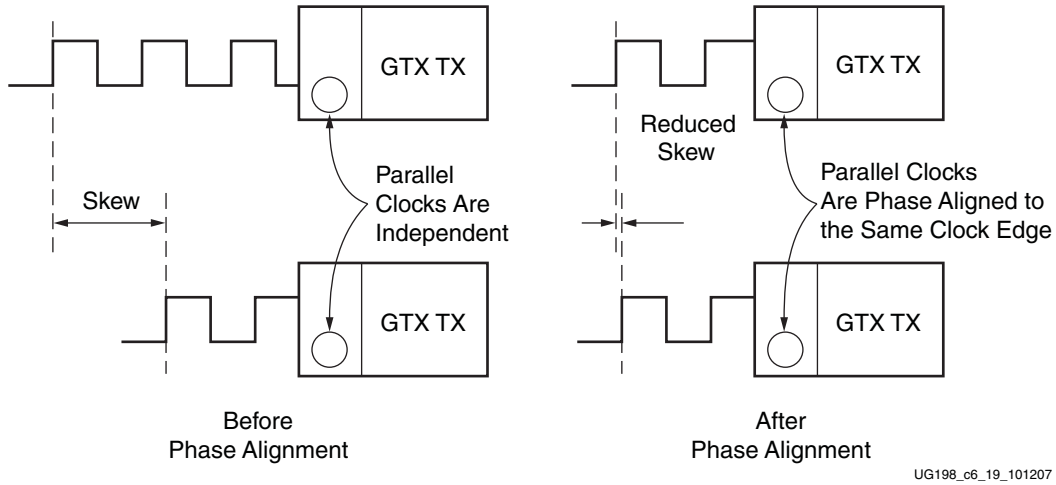


Figure 6-19: **Phase-Alignment Detail**

When oversampling is enabled (OVERSAMPLE_MODE = TRUE), the TX buffer is used for bit interpolation and must always be active. See "Oversampling," page 183 for more information about built-in 5x oversampling.

Ports and Attributes

Table 6-9 defines the signals comprising the TX buffering and phase-alignment ports.

Table 6-9: **TX Buffering and Phase-Alignment Ports**

Port	Direction	Clock Domain	Description
PLLLKDET	Out	Async	This port indicates that the VCO rate is within acceptable tolerances of the desired rate when High. Neither GTX transceiver in the tile operates reliably until this condition is met.
REFCLKOUT	Out	N/A	The REFCLKOUT port from each GTX_DUAL tile provides direct access to the reference clock provided to the shared PMA PLL (CLKIN). It can be routed for use in the FPGA logic.
TXBUFSTATUS0[1:0] TXBUFSTATUS1[1:0]	Out	TXUSRCLK2	TX buffer status. TXBUFSTATUS[1]: TX buffer overflow or underflow 1: FIFO has overflowed or underflowed 0: No overflow/underflow error TXBUFSTATUS[0]: TX buffer fullness 1: FIFO is at least half full 0: FIFO is less than half full If TXBUFSTATUS[1] goes High, it remains High until TXRESET is asserted.
TXENPMAPHASEALIGN0 TXENPMAPHASEALIGN1	In	Async	When activated, both GTX transmitters in a GTX_DUAL tile can align their XCLKs with their TXUSRCLKs. This also allows the XCLKs in multiple GTX transmitters to be synchronized to reduce TX skew between them. This implementation is different from the GTP_DUAL tile because each GTX transceiver has an independent TXENPMAPHASEALIGN input.
TXOUTCLK0 TXOUTCLK1	Out	N/A	This port provides a parallel clock generated by the GTX transceiver. This clock can be used to drive TXUSRCLK for one or more GTX transceivers. The clock rate depends on INTDATAWIDTH: • INTDATAWIDTH is Low: $F_{TXOUTCLK}$ = Line Rate/16 • INTDATAWIDTH is High: $F_{TXOUTCLK}$ = Line Rate/20 ***Note:*** • When INTDATAWIDTH is High, the duty cycle is 60/40 instead of 50/50. TXOUTCLK cannot drive TXUSRCLK when the TX phase-alignment circuit is used. • When oversampling is enabled, the line rate in the calculation of $F_{TXOUTCLK}$ is equal to the oversampled line rate, not the PMA line rate.
TXPMASETPHASE0 TXPMASETPHASE1	In	Async	When activated, TXPMASETPHASE aligns XCLK with TXUSRCLK for both GTX transmitters in the GTX_DUAL tile.
TXUSRCLK0 TXUSRCLK1	In	N/A	Use this port to provide a clock for the internal TX PCS datapath. This clock must always be provided. Its rate depends on INTDATAWIDTH: • INTDATAWIDTH is Low: $F_{TXUSRCLK}$ = Line Rate/16 • INTDATAWIDTH is High: $F_{TXUSRCLK}$ = Line Rate/20

Table 6-10 defines the TX buffering and phase-alignment attributes.

Table 6-10: **TX Buffering and Phase-Alignment Attributes**

Attribute	Type	Description
OVERSAMPLE_MODE	Boolean	This shared attribute activates the built-in 5x digital oversampling circuits in both GTX_DUAL transceivers. Oversampling is supported between $1/10^{th}$ of the lower border of the shared PMA PLL operating range and $4/10^{th}$ of the upper border of the shared PMA PLL operating range. For data rates that need a PLL clock without oversampling that is below one-half of the lower border of the shared PMA PLL, oversampling is mandatory to ensure that the shared PMA PLL operates in its frequency range. TRUE: Built-in 5x digital oversampling enabled for both GTX transceivers on the tile FALSE: Digital oversampling disabled See "Oversampling," page 183 for more details about 5x digital oversampling.
PLL_TXDIVSEL_OUT_0 PLL_TXDIVSEL_OUT_1	Integer	Divides the PLL clock to produce a high-speed TX clock. Because both edges of the clock are used, the divided clock must run at one-half the desired TX line rate. Available divider settings are 1, 2, and 4. Each GTX transceiver has a separate PLL_TXDIVSEL_OUT. See "Parallel In to Serial Out," page 146.
PMA_TX_CFG_0 PMA_TX_CFG_1	20-bit Hex	TX channel specific settings. The default value is `20'h80082`.
TX_BUFFER_USE_0 TX_BUFFER_USE_1	Boolean	This attribute must be set to TRUE when the TX buffer is used. TRUE: Use the TX buffer. FALSE: Bypass the TX buffer. The phase-alignment circuit must be used when TX_BUFFER_USE is FALSE.(1)
TX_XCLK_SEL0 TX_XCLK_SEL1	String	Selects the clock used to drive the clock domain in the PCS following the TX buffer. When using the TX buffer, this attribute is set to TXOUT. The attribute must be set as follows: TXOUT: Use when TX_BUFFER_USE = TRUE TXUSR: Use when TX_BUFFER_USE = FALSE(1)
TXRX_INVERT0 TXRX_INVERT1	3-bit Binary	Controls inverters that optimize the clock paths within the GTX transceiver for different modes of operation. When using the TX buffer, this attribute must be set to `3'b011`. The attribute must be set as follows: `011`: Use when TX_BUFFER_USE = TRUE `111`: Use when TX_BUFFER_USE = FALSE(1)

Notes:

1. Bypassing the TX buffer is an advanced feature and is not recommended for normal operation. TX buffer bypass operation can be guaranteed only under certain system-level conditions and data rates.

Description

Using the TX Buffer

To use the TX buffer to resolve phase differences between the domains, TX_BUFFER_USE must be set to TRUE. The buffer should be reset whenever TXBUFSTATUS indicates an overflow or an underflow. The buffer can be reset using GTXRESET (see "Reset," page 98) or TXRESET (see "FPGA TX Interface," page 116). Assertion of GTXRESET triggers a sequence that resets the entire GTX_DUAL tile.

Using the TX Phase-Alignment Circuit to Minimize TX Skew

To use the phase-alignment circuit to force the XCLK phase of multiple lanes to match the common TXUSRCLK phase, follow these steps.

Initial conditions when TX_BUFFER_USE is TRUE:

- Set TX_BUFFER_USE_0 and TX_BUFFER_USE_1 to TRUE.
- Set TXRX_INVERT0 and TXRX_INVERT1 to 011.
- Set TX_XCLK_SEL0 and TX_XCLK_SEL1 to TXOUT.
- Set PMA_TX_CFG0 and PMA_TX_CFG1 to `20'h80082`.

1. Set TX_XCLK_SEL0 and TX_XCLK_SEL1 to *TXUSR*.
2. Wait for all clocks to stabilize, then drive TXENPMAPHASEALIGN High.

   Keep TXENPMAPHASEALIGN High unless the phase-alignment procedure must be repeated. Driving TXENPMAPHASEALIGN Low causes phase alignment to be lost.
3. Wait 32 TXUSRCLK2 clock cycles, and then drive TXPMASETPHASE High.
4. Wait the number of required TXUSRCLK2 clock cycles as specified in Table 6-11, and then drive TXPMASETPHASE Low. The phase of the PMACLK is now aligned with TXUSRCLK.
5. Set TX_XCLK_SEL0 and TX_XCLK_SEL1 back to *TXOUT.*
6. *Assert and deassert TXRESET synchronously to TXUSRCLK. In this use mode, TXRESET must be deasserted simultaneously to all GTX Transceivers on which the deskew operation is being performed.*

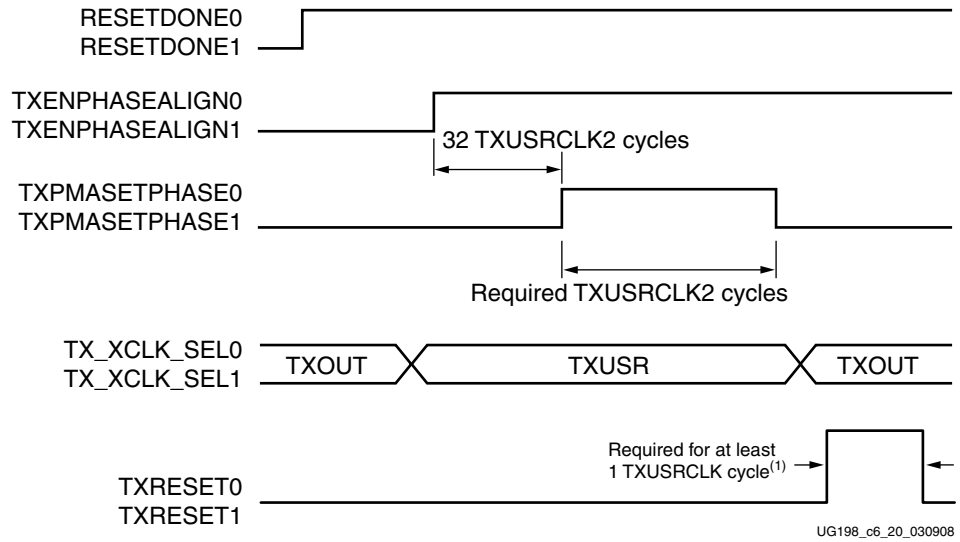
Table 6-11: **Number of Required TXUSRCLK2 Clock Cycles**

PLL_DIVSEL_OUT_0 PLL_DIVSEL_OUT_1	TXUSRCLK2 Wait Cycles
1	8,192
2	16,384
4	32,767

The phase-alignment procedure must be redone if any of the following conditions occur:

- GTXRESET is asserted
- PLLPOWERDOWN is deasserted
- The clocking source changed

Figure 6-20 shows the TX phase-alignment procedure. TXENPMAPHASEALIGN(0/1) and TXPMASETPHASE(0/1) are independent for each GTX transceiver. This implementation is different from the GTP_DUAL tile where TXENPHASEALIGN and TXPMASETPHASE are shared tile pins. The procedure is always applied to each GTX transceiver's TXENPMAPHASEALIGN(0/1) signal on the tile. TXOUTCLK cannot be the source for TXUSRCLK when the TX phase-alignment circuit is used. See "FPGA TX Interface," page 116 for details.



Notes:

1. Assert and deassert TXRESET synchronously to TXUSRCLK. In this use mode, TXRESET must be deasserted simultaneously to all GTX Transceivers on which the deskew operation is being performed.

Figure 6-20: **TX PMACLK Phase-Alignment Procedure**

It is critical that the shared PMA PLL clock and TXUSRCLK both be stable before phase alignment is attempted:

- If REFCLKOUT drives TXUSRCLK directly, wait for PLLLKDET to be asserted before phase aligning.
- If TXUSRCLK comes from a DCM or PLL, wait for PLLLKDET and the LOCKED signal from the DCM or PLL before phase aligning.

Clocking for the TX Phase-Alignment Circuit to Minimize TX Skew

For phase alignment to be effective, TXUSRCLK for all the GTX transceivers must come from the same source and must be routed through a low-skew clocking resource (a BUFG or a BUFR). Figure 6-21 shows how the TXUSRCLK signals must be driven for low-skew operation.

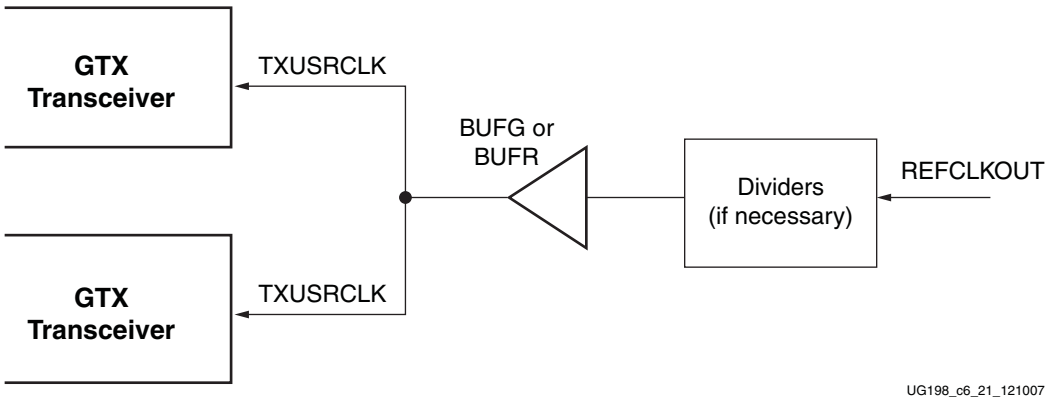


Figure 6-21: **TX Low-Skew Phase-Alignment Configuration**

TX Polarity Control

Overview

The GTX transceiver includes a TX polarity control function to invert outgoing data from the PCS before serialization and transmission. The TXPOLARITY port is driven High to invert the polarity of outgoing data.

Ports and Attributes

Table 6-12 defines the TX polarity control ports.

Table 6-12: **TX Polarity Control Ports**

Port	Direction	Clock Domain	Description
TXPOLARITY0 TXPOLARITY1	In	TXUSRCLK2	Specifies whether the final transmitter output is inverted or not. 0: Not inverted. TXP is positive, and TXN is negative. 1: Inverted: TXP is negative, and TXN is positive.

There are no attributes in this section.

Description

The GTX transceiver can invert the polarity of its TX data before it is transmitted. This feature can be used to avoid hardware fixes for swapped TXP/TXN differential traces on a board.

TX PRBS Generator

Overview

Pseudo-random bit sequences (PRBS) are commonly used to test the signal integrity of high-speed links. These sequences appear random but have specific properties that can be used to measure the quality of a link.

The GTX PRBS block can generate several industry-standard PRBS patterns. Table 6-13 lists the available PRBS patterns and their typical uses.

Table 6-13: **Pseudo-Random Bit Sequences**

Name	Polynomial	Length of Sequence (bits)	Consecutive Zeros	Typical Use
PRBS-7	$1 + X^6 + X^7$ (inverted)	$2^7 - 1$	7	Used to test channels with 8B/10B.
PRBS-23	$1 + X^{18} + X^{23}$ (inverted)	$2^{23} - 1$	23	ITU-T Recommendation O.150, Section 5.6. One of the recommended test patterns in the SONET specification.
PRBS-31	$1 + X^{28} + X^{31}$ (inverted)	$2^{31} - 1$	31	ITU-T Recommendation O.150, Section 5.8. A recommended PRBS test pattern for 10 Gigabit Ethernet. See IEEE 802.3ae-2002.

Ports and Attributes

Table 6-14 defines the TX PRBS generator ports.

Table 6-14: **TX PRBS Generator Ports**

Port	Direction	Clock Domain	Description
TXENPRBSTST0[1:0] TXENPRBSTST1[1:0]	In	TXUSRCLK2	Transmitter test pattern generation control. A pseudo-random bit sequence (PRBS) is generated by enabling the test pattern generation circuit. `00`: Test pattern generation off (standard operation mode) `01`: Enable $2^7 - 1$ PRBS generation `10`: Enable $2^{23} - 1$ PRBS generation `11`: Enable $2^{31} - 1$ PRBS generation Because PRBS patterns are deterministic, the receiver can check the received data against a sequence of its own PRBS generator.

There are no attributes in this section.

Description

Each GTX transceiver includes a built-in PRBS generator. This feature can be used in conjunction with other test features, such as loopback and the built-in PRBS checker, to run tests on a given channel.

To use the PRBS generator, the PRBS test mode is selected using the TXENPRBSTST port. Table 6-14 lists the available settings.

Parallel In to Serial Out

Overview

The Parallel In to Serial Out (PISO) block is the heart of the GTX TX datapath. It serializes parallel data from the PCS using a high-speed clock from the shared PMA PLL.

The PISO block serializes 16 or 20 bits per parallel clock cycle, depending on the internal data width for the tile (INTDATAWIDTH). The clock rate is determined by the shared PMA PLL rate, divided by a local TX divider.

Ports and Attributes

Table 6-15 defines the TX PISO ports.

Table 6-15: **TX PISO Ports**

Port	Direction	Clock Domain	Description
INTDATAWIDTH	In	Async	Specifies the width of the internal datapath for the entire GTX_DUAL tile. This shared port is also described in "Shared PMA PLL," page 84. 0: Internal datapath is 16 bits wide 1: Internal datapath is 20 bits wide

Table 6-16 defines the TX PISO attributes.

Table 6-16: **TX PISO Attributes**

Attribute	Type	Description
OVERSAMPLE_MODE	Boolean	This shared attribute activates the built-in 5x digital oversampling circuits in both GTX transceivers. Oversampling must be enabled when running the GTX transceivers at line rates between 150 Mb/s and 750 Mb/s. TRUE: Built-in 5x digital oversampling enabled for both GTX transceivers on the tile FALSE: Digital oversampling disabled See "Oversampling," page 183 for more details about 5x digital oversampling.
PLL_TXDIVSEL_OUT_0 PLL_TXDIVSEL_OUT_1	Integer	Sets the divider for the TX line rate for the individual GTX transceiver. The divider can be set to 1, 2, or 4.

Description

Equation 6-5 shows how to calculate the TX line rate when operating without oversampling (OVERSAMPLE_MODE = FALSE).

$$\text{Tx Line Rate} = \frac{\text{PLL Clock Rate} \times 2}{\text{PLL\_TXDIVSEL\_OUT}} \quad \textit{Equation 6-5}$$

When oversampling is activated, use Equation 6-6 to calculate the line rate.

$$\text{Tx Line Rate} = \frac{\text{PLL Clock Rate} \times 2}{\text{PLL\_TXDIVSEL\_OUT} \times 5} \quad \textit{Equation 6-6}$$

See "Oversampling," page 183 for more information about oversampling.

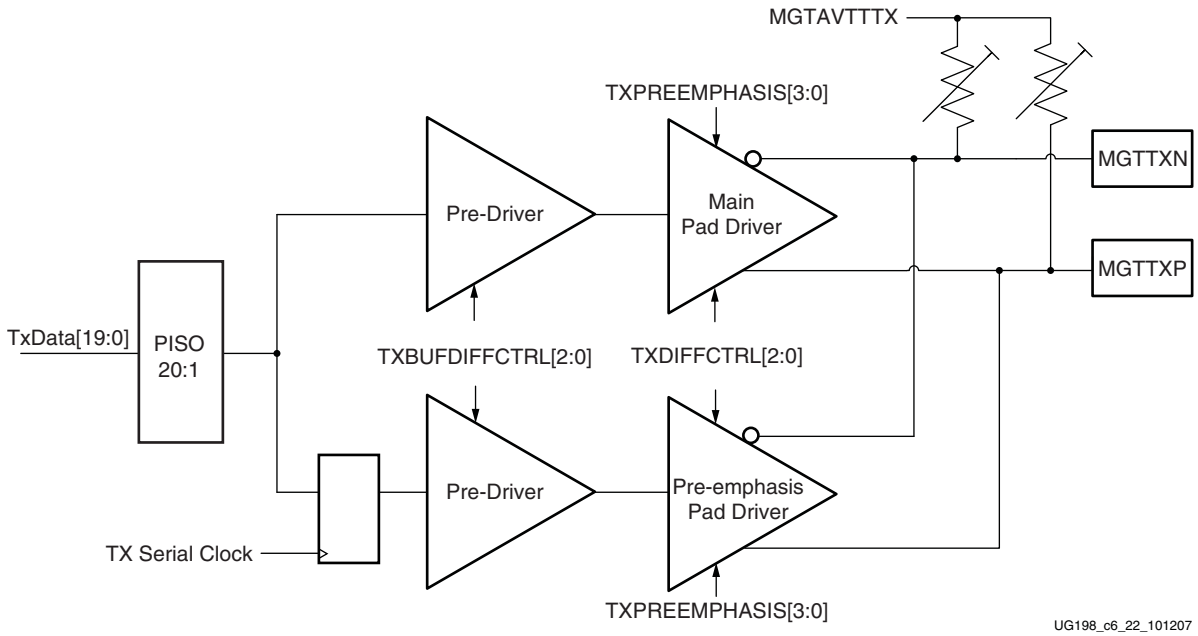
Configurable TX Driver

Overview

The GTX TX driver is a high-speed current-mode differential output buffer. To maximize signal integrity, it includes these features:

- Differential voltage control
- Pre-emphasis
- Configurable termination impedance

The impact of each of these features on signal integrity depends on the board and the receiver. See Chapter 10, "GTX-to-Board Interface," for a detailed discussion of how to use them to maximize signal integrity. Figure 6-22 shows the different segments of the TX driver.



Notes:

1. TXPREEMPHASIS[3] is always 0.

Figure 6-22: **TX Driver Segments**

Ports and Attributes

Table 6-17 defines the TX driver ports.

Table 6-17: **TX Driver Ports**

Ports	Direction	Clock Domain	Description
TXBUFDIFFCTRL0[2:0] TXBUFDIFFCTRL1[2:0]	In	Async	Controls the strength of the pre-drivers. Leave at the default value.
TXDIFFCTRL0[2:0] TXDIFFCTRL1[2:0]	In	Async	Controls the transmitter differential output swing. Table 6-18 shows the approximate differential voltage swing that results from each setting.
TXPREEMPHASIS0[3:0] TXPREEMPHASIS1[3:0]	In	Async	Controls the relative strength of the pre-emphasis as shown in Table 6-18. TXPREEMPHASIS[3] is always 0.

There are no TX driver attributes. The GTX_DUAL tile does not have a TX_DIFF_BOOST attribute.

Description

Differential Voltage Control

The amplitude of the TX driver's differential swing can be controlled using the TXDIFFCTRL and TXBUFDIFFCTRL ports. As shown in Figure 6-22, page 147, TXDIFFCTRL controls the drive strength of the main pad driver and the pre-emphasis pad driver. Table 6-18 shows the differential output voltage that results from each of the possible settings for the port.

Table 6-18: **Transmitter Output Swing and Pre-Emphasis Settings(1)**

Port	Value	Description
TXDIFFCTRL0[2:0] TXDIFFCTRL1[2:0]		Transmitter Differential Output Swing (mV)(2)
	000	500
	001	700
	010	800
	011	900
	100	1000
	101	1100
	110	1200
	111	1300

Table 6-18: **Transmitter Output Swing and Pre-Emphasis Settings(1) *(Cont'd)***

Port	Value	Description
TXBUFDIFFCTRL0[2:0] TXBUFDIFFCTRL1[2:0]		**Transmitter Pre-Driver Swing (mV)(2)**
	000	600
	001	700
	010	800
	011	900
	100	1000
	101(3)	1100
	110	1200
	111	1300
TXPREEMPHASIS0[3:0](5) TXPREEMPHASIS1[3:0](5)		**Transmitter Pre-Emphasis (%)(2,4)**
	000	0
	001	8
	010	17
	011	25
	100	33
	101	42
	110	50
	111	58

Notes:

1. The encoding and values are not the same as those for the GTP_DUAL tile.
2. Nominal values. Refer to the *Virtex-5 FPGA Data Sheet* for the exact values.
3. Recommended default value.
4. Refer to *Handbook of Digital Techniques for High-Speed Design* [Ref 4] and *High-Speed Signal Propagation: Advanced Black Magic* [Ref 7] for detailed explanations of pre-emphasis.
5. TXPREEMPHASIS[3] is always 0.

Each of the pad drivers is driven by a pre-driver whose output amplitude is controlled by TXBUFDIFFCTRL.

Pre-emphasis

Serial traces tend to attenuate high frequencies more than low frequencies. Pre-emphasis is a technique used to pre-equalize transmitted data. It compensates for excess high-frequency loss by transmitting high-frequency signal components with more power than low-frequency signal components.

The pre-emphasis technique used by the GTX transmitter de-emphasizes the low-frequency signal components and emphasizes the high-frequency signal components. When a serial bit is repeated, the power of the repeated bit is reduced. For example, if two ones are sent in a row, the power of the second one is reduced compared to the first. The power of slowly changing signals (the low-frequency components) is reduced relative to quickly changing signals (the high-frequency components). Refer to *Handbook of Digital Techniques for High-Speed Design*, pages 465–471 [Ref 4] and *High-Speed Signal Propagation: Advanced Black Magic*, pages 211–215 [Ref 7] for more detailed explanations of pre-emphasis and de-emphasis.

Each GTX transceiver has a TXPREEMPHASIS port for controlling pre-emphasis. Table 6-18 shows the percentage decrease of signal amplitude for de-emphasized bits at each TXPREEMPHASIS level. The higher the percentage, the more de-emphasis is applied.

Overcompensation of high-frequency losses when using too much pre-emphasis causes signal distortion and increases crosstalk. Combining pre-emphasis with RX equalization (see "RX Termination and Equalization," page 158) can over-emphasize the high frequency signal components at the receiver.

Configurable Termination Impedance

The termination impedance of each transmitter must be matched as closely as possible to the impedance of the serial trace it is driving to minimize distortion due to reflections. See Chapter 10, "GTX-to-Board Interface," for more information about how the termination impedance is calibrated.

TXINHIBIT

The TX driver includes the TXINHIBIT port. When TXINHIBIT is driven High, the TX driver stops sending data and transmits only a differential 0 value (TXP High, TXN Low).

Receive Detect Support for PCI Express Operation

Overview

The GTX transceiver supports receiver detect functionality as defined by the *PHY Interface for PCI Express (PIPE) Specification*. This function drives the TX link to one state followed by a second state and then measures the time required for the link voltage to change. The PIPE specification provides details about the receiver detection mechanism.

Ports and Attributes

Table 6-19 defines the receive detect support ports.

Table 6-19: **Receive Detect Support Ports**

Port	Direction	Domain	Description
PHYSTATUS0 PHYSTATUS1	Out	Async	This signal is asserted High to indicate completion of several PHY functions, including power management state transitions and receiver detection. When this signal transitions during entry and exit from P2 and RXUSRCLK2 is not running, the signaling is asynchronous.
RXSTATUS0[2:0] RXSTATUS1[2:0]	Out	RXUSRCLK2	The decoding of RXSTATUS[2:0] depends on the setting of RX_STATUS_FMT. When RX_STATUS_FMT = PCIE: 000: Receiver not present (when in receiver detection sequence)/Received data OK (during normal operation) 001: Reserved 010: Reserved 011: Receiver present (when in receiver detection sequence) 100: 8B/10B decode error 101: Elastic Buffer Overflow. Different than defined in the PIPE specification. 110: Elastic Buffer Underflow. Different than defined in the PIPE specification. 111: Receive Disparity Error When RX_STATUS_FMT = SATA: RXSTATUS[0]: TXCOMSTART operation complete RXSTATUS[1]: COMWAKE signal received RXSTATUS[2]: COMRESET/COMINIT signal received
TXDETECTRX0 TXDETECTRX1	In	TXUSRCLK2	Activates the receive detection sequence. The sequence ends when PHYSTATUS is asserted to indicate that the results of the test are ready on RXSTATUS.

Table 6-19: **Receive Detect Support Ports *(Cont'd)***

Port	Direction	Domain	Description
RXPOWERDOWN0[1:0] RXPOWERDOWN1[1:0]	In	Async	Controls the power state of the TX and RX links. The encoding complies with the PCI Express specification encoding. TX and RX can be powered down separately. However, for PCI Express compliance, TXPOWERDOWN and RXPOWERDOWN must be used together. 00: P0 (normal operation) 01: P0s (low recovery time power down) 10: P1 (longer recovery time/Receiver Detection still on) 11: P2 (lowest power state)
TXPOWERDOWN0[1:0](1) TXPOWERDOWN1[1:0]		TXUSRCLK2	

Notes:

1. The GTX_DUAL tile's TXPOWERDOWN(0/1) ports are synchronous to the TXUSRCLK2 domain. This is different from the GTP_DUAL function in the Virtex-5 FPGA LXT and SXT platforms.

There are no attributes in this section.

Description

The PCI Express specification includes a feature that allows the transmitter on a given link to detect if a receiver is present. The decision if a receiver is present is based on the rise time of TXP/TXN. Figure 6-23 shows the circuit model used for receive detection. The GTX transceiver must be in the P1 power-down state to perform receiver detection. Also receiver detection requires a 75 to 200 nF external coupling capacitor between the transmitter and receiver, and the receiver must be terminated to GND.

The detection sequence starts with the assertion of TXDETECTRX. In response, the receive detect logic drives TXN and TXP to $V_{DD} - V_{SWING}/2$ and then releases them. After a programmable interval, the levels of TXN and TXP are compared with a threshold voltage. At the end of the sequence, RXSTATUS and PHYSTATUS reflect the results of the receiver detection.

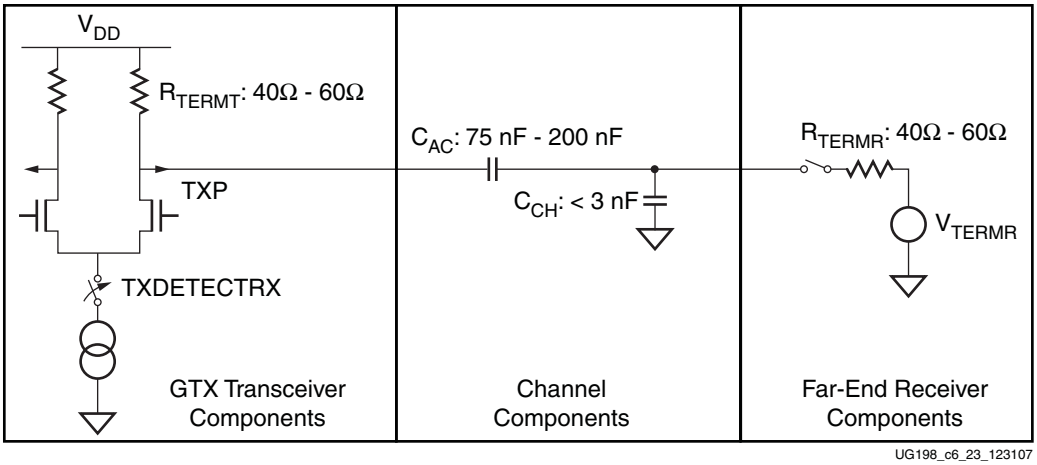


Figure 6-23: **Receive Detection Circuit Model**

Figure 6-24 illustrates the rise times associated with the receiver present and receiver absent conditions, along with the detection threshold.

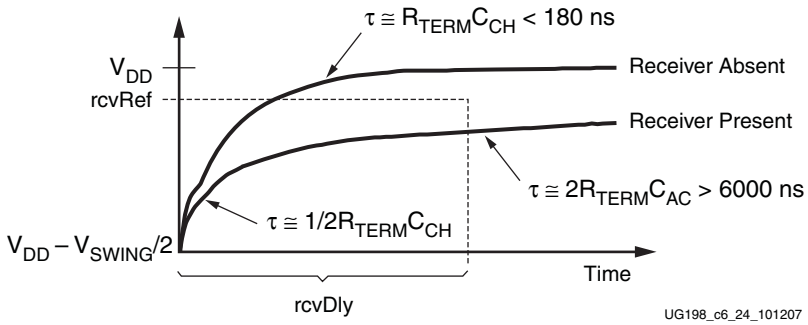


Figure 6-24: **Receive Detection Thresholds**

When the PHY has completed the receiver detect sequence, it asserts PHYSTATUS for one clock and drives the RXSTATUS signals to the appropriate code. After the receiver detection is completed (as signaled by the assertion of PHYSTATUS), TXDETECTRX must be deasserted. Figure 6-25 shows this process.

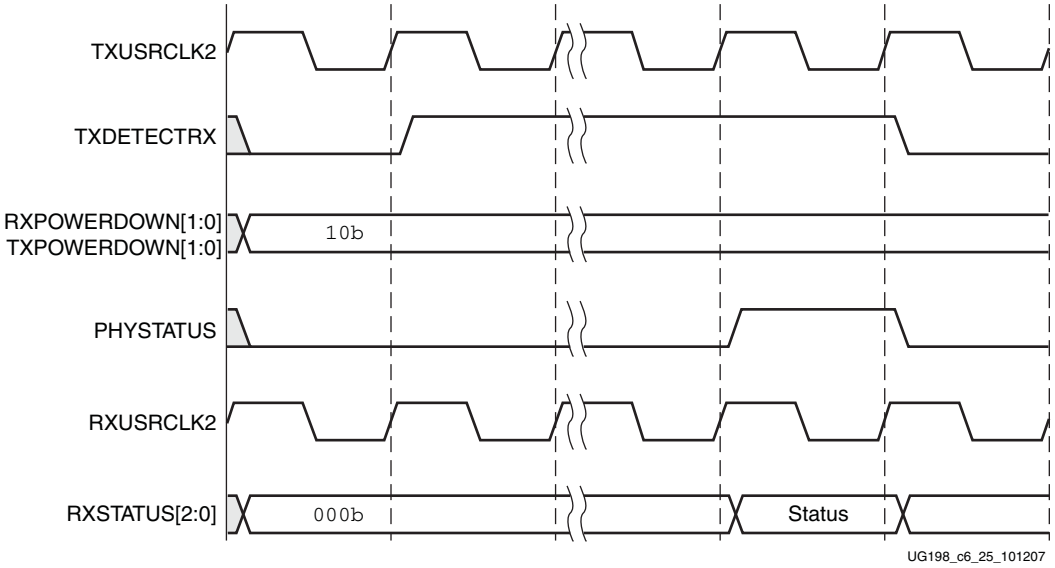


Figure 6-25: **Receiver Detect Waveforms**

TX Out-of-Band/Beacon Signaling

Overview

Each GTX transceiver provides support for generating the Out-of-Band (OOB) sequences described in the Serial ATA (SATA) specification and beaconing described in the PCI Express specification. See Appendix B, "OOB/Beacon Signaling," for an overview of OOB signaling and how it is used in these protocols.

GTX_DUAL support for SATA OOB signaling consists of the analog circuitry required to encode the OOB signal state and state machines to format bursts of OOB signals for SATA COM sequences (COMRESET, COMWAKE, and COMINIT). Each GTX transceiver also supports SATA auto-negotiation by allowing the timing of the COM sequences to be changed based on the divider settings used for the TX line rate.

GTX_DUAL supports beaconing as described in the *PHY Interface for the PCI Express (PIPE) Specification*. The format of the beacon sequence is controlled by the FPGA logic.

Ports and Attributes

Table 6-20 describes the ports that control OOB/beacon signaling.

Table 6-20: **TX OOB/Beacon Signaling Ports**

Port	Direction	Domain	Description
RXSTATUS0[2:0] RXSTATUS1[2:0]	Out	RXUSRCLK2	The decoding of RXSTATUS[2:0] depends on the setting of RX_STATUS_FMT: • When RX_STATUS_FMT = PCIE: RXSTATUS is not used for PCIe TXELECIDLE • When RX_STATUS_FMT = SATA: RXSTATUS[0]: TXCOMSTART operation complete RXSTATUS[1]: COMWAKE signal received RXSTATUS[2]: COMRESET/COMINIT signal received
TXCOMSTART0 TXCOMSTART1	In	TXUSRCLK2	Initiates the transmission of the COM* sequence selected by TXCOMTYPE (SATA only).
TXCOMTYPE0 TXCOMTYPE1	In	TXUSRCLK2	Selects the type of COM signal to send (SATA only): 0: COMRESET/COMINIT 1: COMWAKE
TXELECIDLE0 TXELECIDLE1	In	TXUSRCLK2	When in the P2 power state, this signal controls whether an electrical idle or a beacon indication is driven out onto the TX pair.
TXPOWERDOWN0[1:0] TXPOWERDOWN1[1:0]	In	TXUSRCLK2	Powers down the TX lanes. The GTX_DUAL tile must be in the P2 power state (TXPOWERDOWN = 11) to generate beacon signaling.

Table 6-21 defines the OOB/beacon signaling attributes.

Table 6-21: **TX OOB/Beacon Signaling Attributes**

Attribute	Type	Description
COM_BURST_VAL_0 COM_BURST_VAL_1	4-bit Binary	Number of bursts in a COM sequence.
PLL_SATA_0 PLL_SATA_1	Boolean	Tie to FALSE. When FALSE, PLL_SATA allows TX SATA operations to work at the SATA Generation 1 (1.5 Gb/s) or SATA Generation 2 (3 Gb/s) rate.
PLL_TXDIVSEL_OUT_0 PLL_TXDIVSEL_OUT_1	Integer	Sets the divider for the TX line rate for the individual GTX transceiver. Can be set to 1, 2, or 4.

Description

The GTX_DUAL tile supports two signaling modes: one for SATA operations and one for PCI Express operations. The use of these two mechanisms is mutually exclusive.

Beacon Signaling for PCI Express Operations

Beacon signaling for PCI Express operations is performed when the GTX_DUAL tile is in the P2 power state. Transmission of a beacon is initiated by the deassertion of TXELECIDLE, as shown in Figure 6-26. FPGA control logic controls beacon timing by the sequencing of TXELECIDLE.

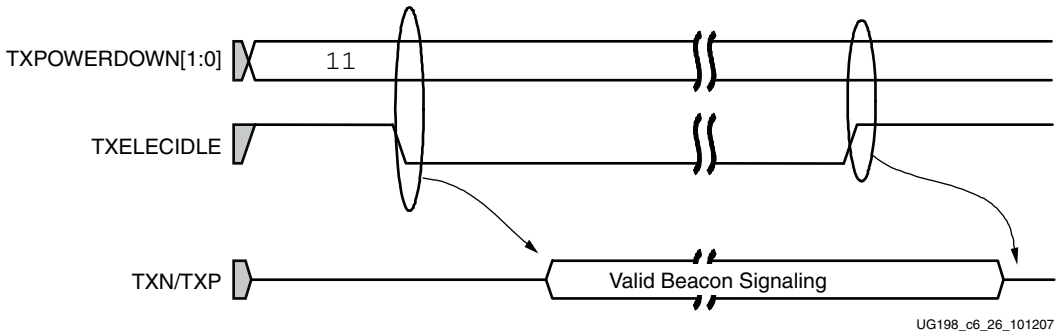


Figure 6-26: **Beacon Generation for PCI Express Operations**

OOB Signaling for SATA Operations

OOB signaling for SATA operation is initiated through the use of the TXELECIDLE, TXCOMSTART, and TXCOMTYPE ports. When TXELECIDLE is held High, assertion of TXCOMSTART for one TXUSRCLK2 cycle initiates the transmission of a COM sequence. The type of COM sequence generated is controlled by the TXCOMTYPE port as shown in Table 6-20.

The number of bursts in the COM sequence is controlled by the COM_BURST_VAL attribute. The timing of the COM sequences transmitted is correct as long as the PLL clock (see "Shared PMA PLL," page 84) is set to 1.5 GHz, and PLL_TXDIVSEL_OUT for each channel is set to either 1 (for a 3.0 Gb/s SATA Generation 2 rate) or 2 (for a 1.5 Gb/s SATA Generation 1 rate).

Note: If the SATA rate needs to be changed at run time (e.g., for SATA auto-negotiation), the GTX_DUAL DRP (see "Dynamic Reconfiguration Port," page 113) can be used to change the appropriate PLL_TXDIVSEL_OUT attribute. The COM sequence timing is adjusted automatically.

When a TXCOMSTART operation completes, RXSTATUS[0] is driven High for one RXUSRCLK2 cycle. Do *not* use the output of RXSTATUS[0] without synchronizing it to the TXUSRCLK2 domain, if TXUSRCLK2 and RXUSRCLK2 are driven by separate clocks.

Chapter 7

GTX Receiver (RX)

This chapter shows how to configure and use each of the functional blocks inside the GTX receiver.

Receiver Overview

Each GTX transceiver includes an independent receiver, made up of a PCS and a PMA. Figure 7-1 shows the functional blocks of the receiver (RX). High-speed serial data flows from traces on the board into the PMA of the RX, into the PCS, and finally into the FPGA logic. Refer to Appendix E, "Low Latency Design," for latency information on this block diagram.

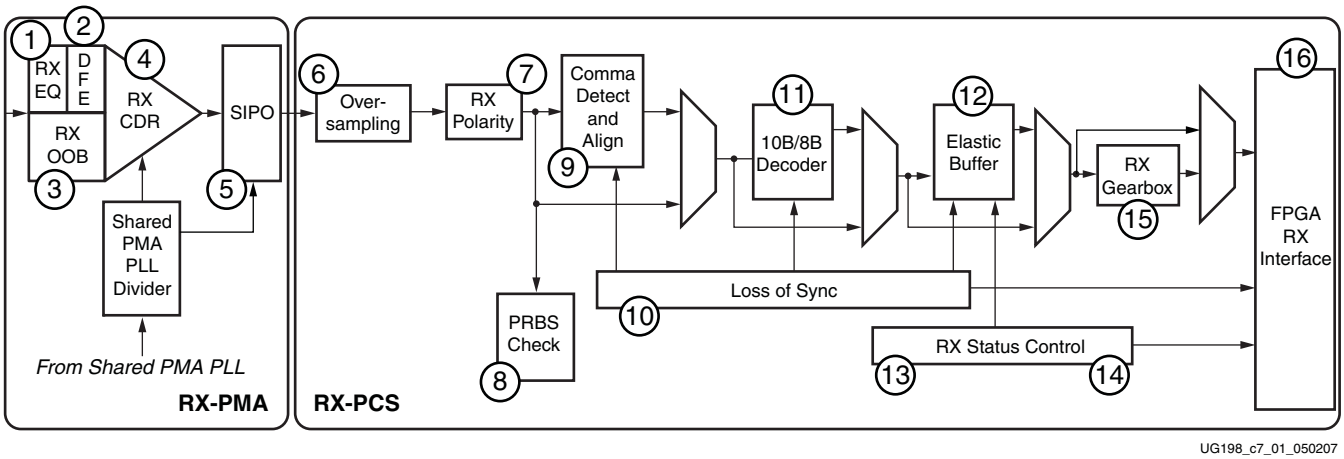


Figure 7-1: **GTX RX Block Diagram**

The key elements within the GTX receiver are:

1. "RX Termination and Equalization," page 158
2. "Decision Feedback Equalization," page 163
3. "RX OOB/Beacon Signaling," page 170
4. "RX Clock Data Recovery," page 176
5. "Serial In to Parallel Out," page 181
6. "Oversampling," page 183
7. "RX Polarity Control," page 187
8. "PRBS Detection," page 188
9. "Configurable Comma Alignment and Detection," page 189

RX Termination and Equalization

Overview

The first stage of the RX datapath in each GTX transceiver is the RX current mode logic (CML) receiver. This block determines the value of the incoming high-speed differential signal.

At high speeds, error-free data reception requires good signal integrity. The CML receiver includes circuits to allow the termination of the channel to be optimized for the best possible signal integrity.

The receiver also features an RX equalization circuit that allows it to compensate for high-frequency losses in the channel, improving the quality of the received signal. This circuit can be tuned to meet the specific requirements of the physical channel used in the design.

Ports and Attributes

The parameters of the 4-mode linear equalizer are described in Table 7-1.

Table 7-1: **RX Termination and Equalization Ports**

Port	Dir	Clock Domain	Description
MGTRXN0 MGTRXN1 MGTRXP0 MGTRXP1	In (Pad)	RX Serial Clock	RXN and RXP are the differential complements of each other forming a differential receiver input pair. These ports represent pads. The location of these ports must be constrained (see Chapter 4, "Implementation") and brought to the top level of the design.
RXEQMIX0[1:0](1) RXEQMIX1[1:0]	In	Async	This port controls the 4-mode linear RX equalization circuit. The following ratios are available: 00: Large high-frequency boost (recommended mode for DFE) 01: Small high-frequency boost 10: Moderate high-frequency boost (recommended mode for DFE) 11: Bypass with gain

Notes:

1. See "Optional 4-Mode Active Linear Equalization," page 161 for details on how to set RXEQMIX[1:0].

Table 7-2 describes the attributes and settings regarding termination for both receivers of the GTX_DUAL tile.

Table 7-2: **RX Termination and Equalization Attributes**

Attribute	Type	Description
AC_CAP_DIS_0 AC_CAP_DIS_1	Boolean	Bypasses the built-in AC coupling in the receiver. Use this attribute when DC coupling is required. The default for this attribute is FALSE for PCI Express designs. For all other protocols, the default setting is TRUE. TRUE: Built-in AC coupling capacitors are bypassed. DC coupling to the receiver is possible. FALSE: Built-in AC coupling capacitors are enabled. See Chapter 10, "GTX-to-Board Interface," for details about when it is appropriate to add an additional external AC coupling capacitor based on data rate or protocol.
CM_TRIM_0 CM_TRIM_1	2-bit Binary	Adjusts the input common mode levels. These levels are automatically set in the RocketIO GTX Transceiver Wizard.
RCV_TERM_GND_0 RCV_TERM_GND_1	Boolean	Activates the Ground reference for the receiver termination network. The default for this attribute is TRUE for PCI Express designs. For all other protocols, the default setting is FALSE. TRUE: Ground reference for receiver termination activated. FALSE: Ground reference for receiver termination disabled. See Table 7-4, page 161 for valid combinations of RX termination attributes. Must be set to TRUE for PCI Express designs.
RCV_TERM_VTTRX_0 RCV_TERM_VTTRX_1	Boolean	Activates MGTAVTTRX reference for receiver termination network. The default for this attribute is FALSE for PCI Express designs. For all other protocols, the default setting is TRUE. Set to FALSE when using AC coupling. TRUE: MGTAVTTRX reference for receiver termination activated. FALSE: MGTAVTTRX reference for receiver termination disabled. See Table 7-4, page 161 for valid combinations of RX termination attributes.
TERMINATION_IMP_0 TERMINATION_IMP_1	Integer	Selects the termination impedance for the TX driver and RX CML receiver. See Chapter 10, "GTX-to-Board Interface," for details on calibration of impedance values. Always set to 50 to select 50Ω termination impedance. The RocketIO GTX Transceiver Wizard automatically sets this parameter to 50.

Description

The GTX_DUAL receivers are connected via differential pad pairs RXN and RXP to the transmission line on the board. The receiver includes four main features for optimizing signal integrity:

- Optional Built-in AC Coupling
- Configurable Termination Impedance
- Configurable Termination Voltage
- Optional 4-Mode Active Linear Equalization

Optional Built-in AC Coupling

The GTX receiver includes small AC-coupling capacitors that isolate the receiver from the line. These capacitors are required when the RX termination is set to GND. When GND termination is not used, these capacitors can be disabled to allow for internal DC coupling. See Chapter 10, "GTX-to-Board Interface," for more details about how to pick an appropriately sized external AC coupling capacitor. Some applications with low data rates, long run lengths, or both require an external AC coupling capacitor.

The AC_CAP_DIS attribute controls the AC-coupling bypass switch.

To turn on built-in AC coupling, AC_CAP_DIS is set to FALSE. To disable the built-in AC coupling, AC_CAP_DIS is set to TRUE.

Configurable Termination Impedance

To minimize distortion due to reflections, the termination impedance of each receiver must be matched as closely as possible to the impedance of the serial trace to which it is connected. See Chapter 10, "GTX-to-Board Interface," for more information about how the termination impedance is calibrated.

Configurable Termination Voltage

The line termination circuit connects the RXP and RXN pads through the calibrated line termination impedances to one of three termination voltages: GND, MGTAVTTRX, or 2/3 MGTAVTTRX (from the GTX receiver). At any given time, only one of these connections is active.

The appropriate RX termination voltage is selected based on the type of coupling used, the TX termination voltage, and the requirements of the protocol being used. Table 7-3 summarizes the termination options available in the GTX receiver. Table 7-4 shows how to activate the different RX termination options.

Table 7-3: **Recommended RX Termination Settings**

Coupling Type	Valid RX Termination Settings	Notes
DC coupled	2/3 MGTAVTTRX, MGTAVTTRX	The TX termination (and TX termination voltage) must match the selected RX termination (and RX termination voltage) to prevent DC currents. 2/3 MGTAVTTRX is optimal for the receiver.
External AC coupling only	2/3 MGTAVTTRX	Recommended configuration because it permits connections to the widest variety of transceivers including hot-swap applications.

Table 7-3: **Recommended RX Termination Settings *(Cont'd)***

Coupling Type	Valid RX Termination Settings	Notes
Built-in AC coupling only	MGTAVTTRX, 2/3 MGTAVTTRX, GND	The RX termination (and RX termination voltage) must match the TX termination (and TX termination voltage) to prevent DC currents. The built-in AC-coupling capacitors do not block a DC current path between the RX termination of the receiver and the TX termination of the transmitter. Only an external AC-coupling capacitor prevents a DC current path between the transmitter's TX termination and the receiver's RX termination. See Chapter 10, "GTX-to-Board Interface," for more details about how to pick an appropriately sized external AC-coupling capacitor.
Built-in and external AC coupling	MGTAVTTRX, 2/3 MGTAVTTRX, GND	Use GND to support the TXDETECTRX feature for PCI Express designs. See "Receive Detect Support for PCI Express Operation," page 151.

Table 7-4: **RX Termination Attribute Settings**

RX Termination Voltage	RCV_TERM_GND	RCV_TERM_VTTRX
MGTAVTTRX	FALSE	TRUE
2/3 MGTAVTTRX	FALSE	FALSE
GND	TRUE	FALSE

Optional 4-Mode Active Linear Equalization

High-speed serial traces and connections typically attenuate high-frequency signals more than low-frequency signals. As a result, high-frequency data passing through a channel tends to get distorted as the high-frequency components of the signal lose more power than the low-frequency components.

The GTX receiver includes an active linear equalizer circuit that can be used to compensate for signal distortion on the line due to high-frequency attenuation.

The equalizer offers four different modes of equalization with different frequency response characteristics (Figure 7-2). Modes with a higher amplification of high-frequency signal components are intended for transmission channels with a strong attenuation of high-frequency signal components. The mode of the equalizer is controlled by the RXEQMIX port. Table 7-1, page 158 shows the four settings.

The receiver termination and equalization attributes and settings for the GTX_DUAL tile are shown in Table 7-2, page 159.

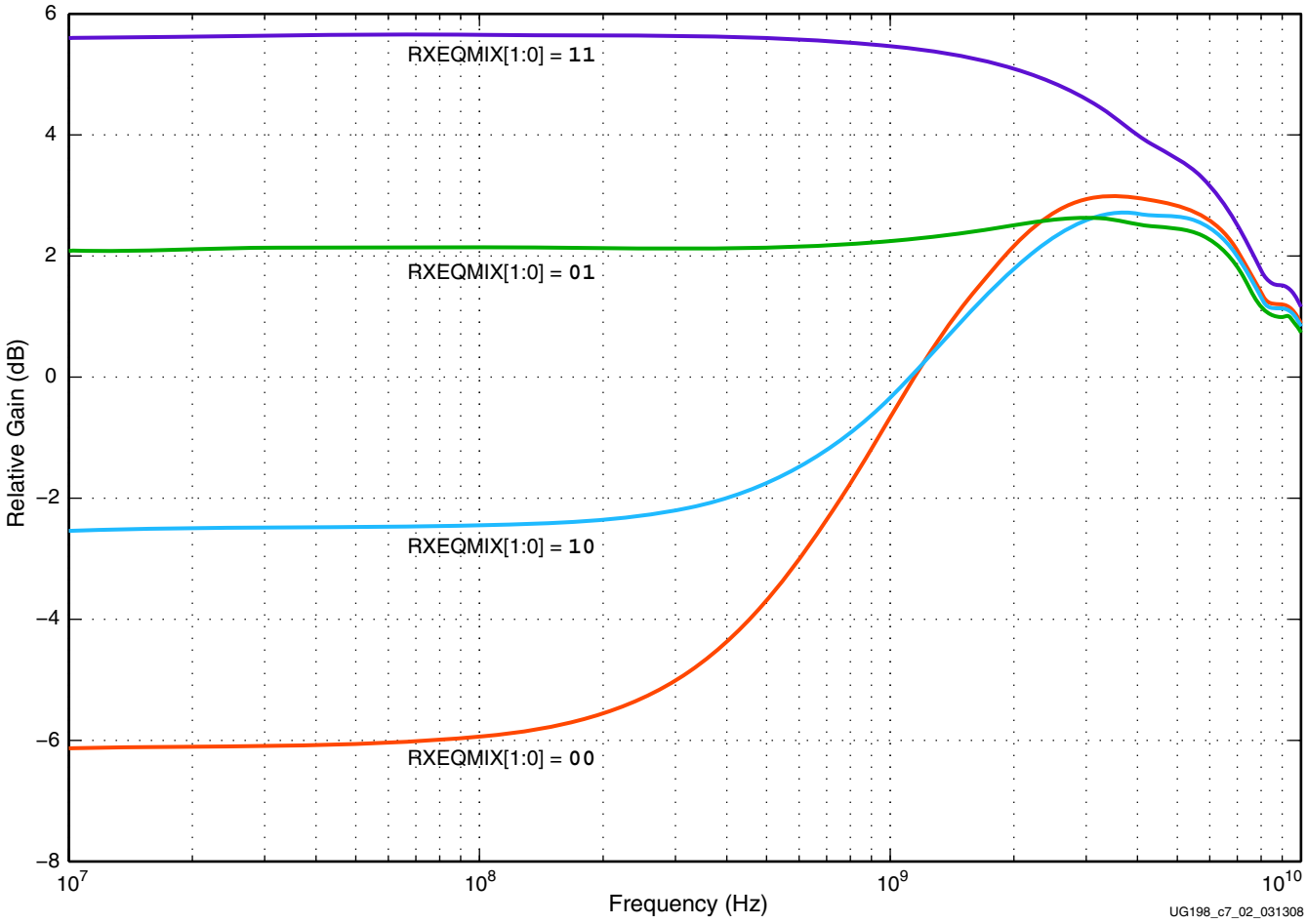


Figure 7-2: **RX Equalizer**

The recommended settings of RXEQMIX[1:0] when using the Decision Feedback Equalizer (DFE) are `00` and `10`. Another possibility is to use IBERT to find the EQ setting that delivers the best possible bit error rate (BER).

Decision Feedback Equalization

Overview

Note: This feature is currently undergoing characterization. The information provided in this section will be updated when characterization is completed.

The Decision Feedback Equalizer (DFE) block in the RX data path follows the linear equalization block. This block provides an efficient reduction to the tail of a transmitted bit or post-cursor. Attenuation of the pre-cursor or post-cursor of a transmitted bit reduces the Inter-Symbol-Inference (ISI) and improves system margin.

Ports and Attributes

Table 7-5 defines the DFE ports.

Table 7-5: **DFE Ports**

Port	Direction	Clock Domain	Description
DFECLKDLYADJ0[5:0] DFECLKDLYADJ1[5:0]	In (Pad)	RXUSRCLK2	DFE clock delay adjust override for each transceiver. `00000`: Phase difference between the bit serial sample clock and the DFECLK is 0°. `11111`: Phase difference between the bit serial sample clock and the DFECLK is 90°. This DFE is automatically calibrated for optimal performance by a built-in state machine. This override value is only accepted when bit 8 of DFE_CFG_(0/1) is `1`.
DFECLKDLYADJMONITOR0[5:0] DFECLKDLYADJMONITOR1[5:0]	Out	RXUSRCLK2	DFE clock delay calibration result monitor for each transceiver.
DFEEYEDACMONITOR0[4:0] DFEEYEDACMONITOR1[4:0]	Out	RXUSRCLK2	Averaged Vertical Eye Height (voltage domain) used by the DFE as an optimization criterion. `11111`: indicates approximately 180 mV_{PPD} of internal eye opening.
DFESENSCAL0[2:0] DFESENSCAL1[2:0]	Out	RXUSRCLK2	Sampler sensitivity self-calibration after the reset. `000`: Lowest offset `111`: Highest offset
DFETAP10[4:0] DFETAP11[4:0]	In	RXUSRCLK2	DFE tap 1 weight value control for each transceiver (5-bit resolution).
DFETAP1MONITOR0[4:0] DFETAP1MONITOR1[4:0]	Out	RXUSRCLK2	DFE tap 1 weight value monitor for each transceiver (5-bit resolution).
DFETAP20[4:0] DFETAP21[4:0]	In	RXUSRCLK2	DFE tap 2 weight value control for each transceiver (4-bit resolution plus 1-bit sign). For example, –2 is represented as `1 0010`.
DFETAP2MONITOR0[4:0] DFETAP2MONITOR1[4:0]	Out	RXUSRCLK2	DFE tap 2 weight value monitor for each transceiver (4-bit resolution plus 1-bit sign). For example, –2 is represented as `1 0010`.
DFETAP30[3:0] DFETAP31[3:0]	In	RXUSRCLK2	DFE tap 3 weight value control for each transceiver (3-bit resolution plus 1-bit sign). For example, –2 is represented as `1 010`.

Table 7-5: **DFE Ports *(Cont'd)***

Port	Direction	Clock Domain	Description
DFETAP3MONITOR0[3:0] DFETAP3MONITOR1[3:0]	Out	RXUSRCLK2	DFE tap 3 weight value monitor for each transceiver (3-bit resolution plus 1-bit sign). For example, –2 is represented as `1 010`.
DFETAP40[3:0] DFETAP41[3:0]	In	RXUSRCLK2	DFE tap 4 weight value control for each transceiver (3-bit resolution plus 1-bit sign). For example, –2 is represented as `1 010`.
DFETAP4MONITOR0[3:0] DFETAP4MONITOR1[3:0]	Out	RXUSRCLK2	DFE tap 4 weight value monitor for each transceiver (3-bit resolution plus 1-bit sign). For example, –2 is represented as `1 010`.

Table 7-6 defines the DFE attributes.

Table 7-6: **DFE Attributes**

<table>
<tr><th>Attribute</th><th>Type</th><th colspan="2">Description</th></tr>
<tr><td>DFE_CAL_TIME</td><td>5-bit Binary</td><td colspan="2">DFE calibration time. Set to <code>00110</code> for normal operation (default).</td></tr>
<tr><td rowspan="5">DFE_CFG_0[9:0]
DFE_CFG_1[9:0]</td><td rowspan="5">10-bit Binary</td><td>DFE_CFG</td><td>Description</td></tr>
<tr><td>9</td><td>Override enable for the DFE tap values.
<code>0</code>: Use the optimized DFE tap self-adapting values.
<code>1</code>: Override the DFE self-adapting values (recommended) with the values provided by DFETAP1, DFETAP2, DFETAP3, and DFETAP4.</td></tr>
<tr><td>8</td><td>Override enable for the DFE clock delay adjustment.
<code>0</code>: Use the optimized DFE clock delay calibration value (recommended).
<code>1</code>: Override the DFE clock delay calibration state machine with the value provided by DFECLKDLYADJ0[5:0] or DFECLKDLYADJ1[5:0]. The optimized DFE clock delay calibration is done when GTXRESET is deasserted.
When switching from override mode to the optimized mode dynamically, always toggle GTX_RESET.</td></tr>
<tr><td>[7:3]</td><td>Limiter/EQAMP gain and power control
Set to <code>01111</code> for high-speed operation (default)
Set to <code>10011</code> for low-speed operation</td></tr>
<tr><td>[2:0]</td><td>Vertical Eye measure AMP gain and power control.
Set to <code>011</code> for high-speed operation (default)
Set to <code>101</code> for low-speed operation</td></tr>
<tr><td>RX_EN_IDLE_HOLD_DFE_0
RX_EN_IDLE_HOLD_DFE_1</td><td>Boolean</td><td colspan="2">TRUE: Restores the DFE contents from internal registers after termination of an electrical idle state for PCI Express operation. Holds the DFE circuit in reset when an electrical idle condition is detected.
FALSE: (default)
Note: For channels with large attenuation, it is recommended that RX_EN_IDLE_HOLD_DFE_0/1 should be set to FALSE because fast transitioning data patterns like the <code>101010</code> sequence in CJPAT/CJTPAT can accidentally trigger an electrical idle.</td></tr>
</table>

Description

Figure 7-3 illustrates the location of the DFE in the RX.

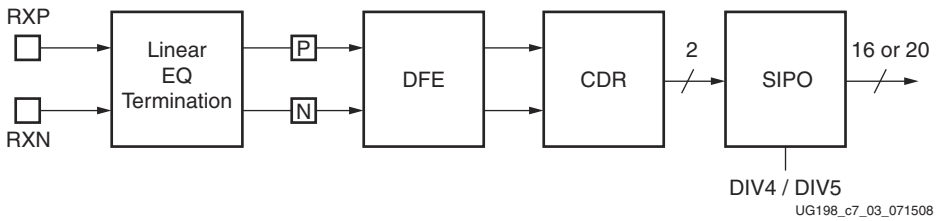


Figure 7-3: **DFE in the GTX RX**

Figure 7-4 illustrates a conceptual view of the DFE.

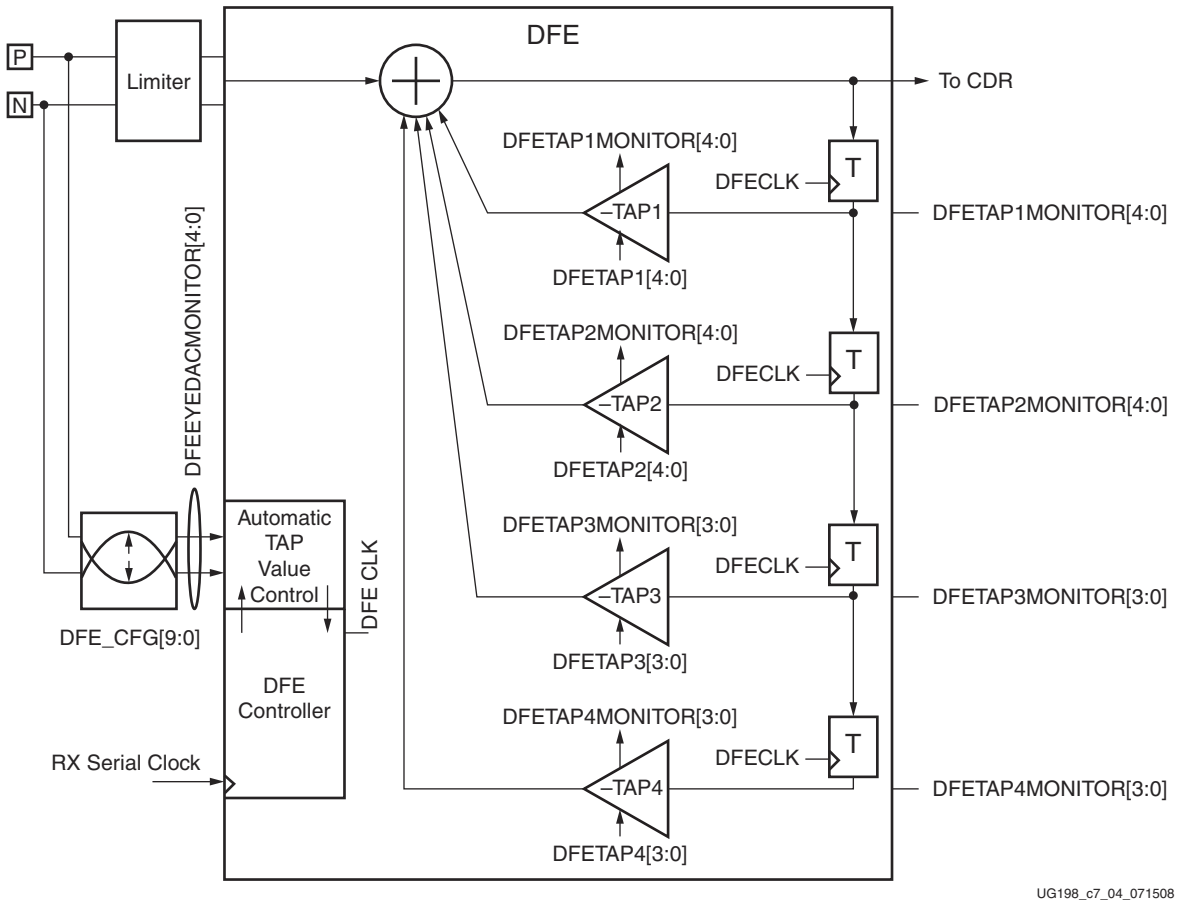


Figure 7-4: **DFE Conceptual View**

The DFE allows better compensation of transmission channel losses by providing a closer adjustment of filter parameters than when using a linear equalizer. However, a DFE cannot remove the pre-cursor of a transmitted bit. A linear equalizer allows pre-cursor and post-cursor attenuation, but has only a coarse adaptor to the transmission channel characteristic. The GTX_DUAL DFE in the GTX RX is a time-discrete adaptive high-pass filter(1). The TAP values of the DFE are the coefficients of this filter that are set by the automatic TAP value controller. The optimization criteria for the TAP values and the DFECLK delay is the vertical eye opening. The DFE_CFG_(0/1) attribute switches off auto-calibration and overrides the TAP values and the DFECLK delay.

Channel BER Optimization Approach

The TX pre-emphasis, the RX equalization, and the DFE work together to optimize the BER for a channel. In general, the steps for optimizing the BER are:

1. Determining adequate transmitter swing.
   - For chip-to-chip applications, a swing level (outer eye amplitude) of 600 mV_{PPD} is a good starting point.
   - For backplane applications, a swing level (outer eye amplitude) of 800 mV_{PPD} to 1000 mV_{PPD} is a good starting point.
2. Tuning the RX continuous-time linear equalizer.
3. Tuning the RX DFE.
4. Tuning the TX pre-emphasis.

Figure 7-5 illustrates the channel BER optimization flow.

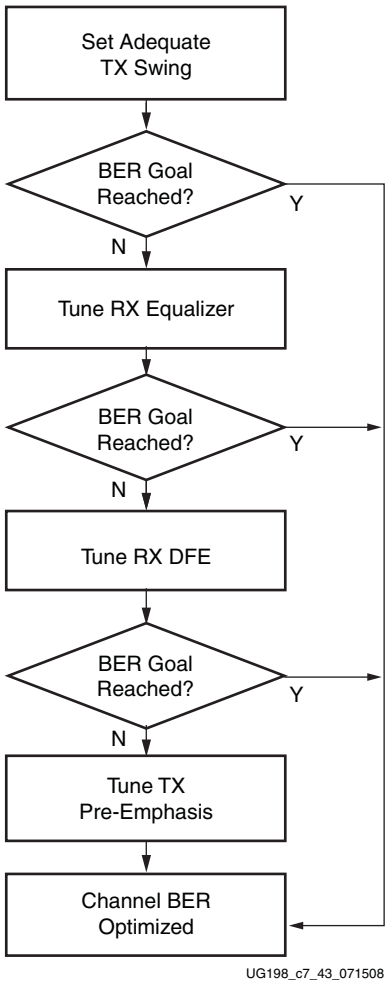


Figure 7-5: **General Channel BER Optimization Flow**

1. Refer to *Digital Filters - Basics and Design*, page 95ff [Ref 11] for a block diagram of an IIR filter. The DFE can be seen as an IIR filter with only a recursive part, which implies that the coefficients for the non-recursive part are all equal to zero with the exception of b0, which is one.

Channel equalization mechanisms like transmit pre-emphasis and receiver equalizers reduce ISI at the expense of voltage swing. Therefore, adequate swing level is important. However, an overly high swing level can increase ISI.

The RX linear equalizer can reduce ISI and has a power advantage. The RX DFE can provide fine tuning of internal eye height optimization. If RX linear equalization is insufficient to fully compensate for the channel ISI, transmitter pre-emphasis can provide additional ISI reduction.

Use Mode – Fixed Tap Mode

This mode requires that DFE_CFG[9] = 1 (DFETAPx value override). It is recommended that DFE_CFG[8] = 0 (optimized DFE clock delay calibration). The remaining DFE_CFG bits must be held at their default values.

The values to be written to the DFE taps are applied to DFETAPx0/1; DFETAPxMONITOR0/1 reflects the value entered.

The chip-to-chip and backplane application examples provide starting points for setting DFETAP1 and RXEQMIX0/1 based on the trace length and the associated loss of the channel. DFETAP2, DFETAP3, and DFETAP4 are set to 0 in these examples. For channels with different characteristics or lengths than given in the examples, the designer should use the example with the closest match as a starting point.

Either the GTX Wizard example design or IBERT can be used to fine-tune the settings to the user's particular channel.

Example RX Linear Equalizer and DFE Settings for Chip-to-Chip Applications

Table 7-7 provides DFETAP1 and RXEQMIX settings for 4.25 Gb/s operation for chip-to-chip applications.

Table 7-7: **DFETAP1 and RXEQMIX at 4.25 Gb/s for Chip-to-Chip Applications**

Nominal Trace Length on FR4 Substrate (inches [mm])	Loss (dB) @ 2.125 GHz	DFETAP1	
		RXEQMIX = 10	RXEQMIX = 00
18 [457.2]	5	0	0
28 [711.2]	7.5	10	0
38 [965.2]	10.5	N/A	0
48 [1219.2]	13.25	N/A	0

Notes:

1. GTX TXDIFFCTRL = 111 and TXPREEMPHASIS = 000 (maximum TX swing and no TX pre-emphasis).

In most chip-to-chip applications operating at 4.25 Gb/s with a nominal trace length of up to 48 inches and around 13.25 dB of attenuation at 2.125 GHz, RX linear equalization is sufficient without any need for DFE.

Table 7-8 provides DFETAP1 and RXEQMIX settings for 5 Gb/s operation for chip-to-chip applications.

Table 7-8: **DFETAP1 and RXEQMIX at 5 Gb/s for Chip-to-Chip Applications**

Nominal Trace Length on FR4 Substrate (inches [mm])	Loss (dB) @ 2.5 GHz	DFETAP1	
		RXEQMIX = 10	RXEQMIX = 00
18 [457.2]	5.75	0	0
28 [711.2]	8.75	10	0
38 [965.2]	12.25	N/A	0
48 [1219.2]	15.5	N/A	5

Notes:

1. GTX TXDIFFCTRL = `111` and TXPREEMPHASIS = `000` (maximum TX swing and no TX pre-emphasis).

In most chip-to-chip applications operating at 5 Gb/s with up to a nominal trace length of 48 inches with approximately 15.5 dB of attenuation at 2.5 GHz, RX linear equalization with a slight boost on DFETAP1 is sufficient. TX pre-emphasis can also reduce ISI and replace the function of DFE.

Table 7-9 provides DFETAP1 and RXEQMIX settings for 6.5 Gb/s operation for chip-to-chip applications.

Table 7-9: **DFETAP1 and RXEQMIX at 6.5 Gb/s for Chip-to-Chip Applications**

Nominal Trace Length on FR4 Substrate (inches [mm])	Loss (dB) @ 3.25 GHz	DFETAP1	
		RXEQMIX = 10	RXEQMIX = 00
18 [457.2]	8	10	5
28 [711.2]	11.5	31	5
38 [965.2]	15.5	N/A	25
48 [1219.2]	20	N/A	31

Notes:

1. GTX TXDIFFCTRL = `111` and TXPREEMPHASIS = `000` (maximum TX swing and no TX pre-emphasis).

In most chip-to-chip applications, where the nominal trace length is 10 inches and shorter at 6.5 Gb/s, TX pre-emphasis and continuous time linear equalization are sufficient. For longer chip-to-chip applications, the DFE should be used.

Example RX Linear Equalizer and DFE Settings for Backplane Applications

Table 7-10 provides DFETAP1 and RXEQMIX settings for 4.25 Gb/s operation for backplane applications.

Table 7-10: **DFETAP1 and RXEQMIX at 4.25 Gb/s for Backplane Applications**

Nominal Trace Length on FR4 Substrate (inches [mm])	Loss (dB) @ 2.125 GHz	DFETAP1	
		RXEQMIX = 10	RXEQMIX = 00
30 [762](2)	7.75 - 8.25	5	0
44 [1117.6](3)	9.5 - 11	N/A	0
64 [1625.6](4)	13.5 - 17	N/A	23

Notes:

1. GTX TXDIFFCTRL = 111 and TXPREEMPHASIS = 000 (maximum TX swing and no TX pre-emphasis).
2. Nominal 30 inches [762 mm] trace length = 6 inches [152.4 mm] backplane + 2 HmZD or eHSD connectors (trace length neglected) + 24 inches [609.6 mm] on line cards.
3. Nominal 44 inches [1117.6 mm] trace length = 20 inches [508 mm] backplane + 2 HmZD or eHSD connectors (trace length neglected) + 24 inches [609.6 mm] on line cards.
4. Nominal 64 inches [1625.6 mm] trace length = 40 inches [1016 mm] backplane + 2 HmZD or eHSD connectors (trace length neglected) + 24 inches [609.6 mm] on line cards.

In most backplane applications operating at 4.25 Gb/s with up to a total nominal trace length of 64 inches and an end-to-end attenuation of 17 dB at 2.125 GHz, RX linear equalization alone is sufficient.

Table 7-11 provides DFETAP1 and RXEQMIX settings for 5 Gb/s operation for backplane applications.

Table 7-11: **DFETAP1 and RXEQMIX at 5 Gb/s for Backplane Applications**

Nominal Trace Length on FR4 Substrate (inches [mm])	Loss (dB) @ 2.5 GHz	DFETAP1	
		RXEQMIX = 10	RXEQMIX = 00
30 [762](2)	8.5 - 9.5	15	0
44 [1117.6](3)	11.5 - 13	-	5

Notes:

1. GTX TXDIFFCTRL = 111 and TXPREEMPHASIS = 000 (maximum TX swing and no TX pre-emphasis).
2. Nominal 30 inches [762 mm] trace length = 6 inches [152.4 mm] backplane + 2 HmZD or eHSD connectors (trace length neglected) + 24 inches [609.6 mm] on line cards.
3. Nominal 44 inches [1117.6 mm] trace length = 20 inches [508 mm] backplane + 2 HmZD or eHSD connectors (trace length neglected) + 24 inches [609.6 mm] on line cards.

In most backplane applications operating at 5 Gb/s with a nominal trace length of 44 inches and approximately 13 dB of attenuation at 2.5 GHz, RX linear equalization with a slight boost on DFETAP1 is sufficient. TX pre-emphasis can also reduce ISI and replace the function of DFE.

Table 7-12 provides DFETAP1 and RXEQMIX settings for 6.5 Gb/s operation for backplane applications.

Table 7-12: **DFETAP1 and RXEQMIX at 6.5 Gb/s for Backplane Applications**

Nominal Trace Length on FR4 Substrate (inches [mm])	Loss (dB) @ 3.25 GHz	DFETAP1
		RXEQMIX = 00
30 [762](2)	11.8 - 15	15
44 [1117.6](3)	15.5 - 19.25	31

Notes:

1. GTX TXDIFFCTRL = `111` and TXPREEMPHASIS = `000` (maximum TX swing and no TX pre-emphasis).
2. Nominal 30 inches [762 mm] trace length = 6 inches [152.4 mm] backplane + 2 HmZD or eHSD connectors (trace length neglected) + 24 inches [609.6 mm] on line cards.
3. Nominal 44 inches [1117.6 mm] trace length = 20 inches [508 mm] backplane + 2 HmZD or eHSD connectors (trace length neglected) + 24 inches [609.6 mm] on line cards.

RX OOB/Beacon Signaling

Overview

The GTX_DUAL tile provides support for decoding the OOB sequences described in the Serial ATA (SATA) specification and supports beaconing described in the PCI Express specification. An overview of OOB signaling and how it is used in these protocols can be found in Appendix B, "OOB/Beacon Signaling."

Support for SATA OOB signaling consists of the analog circuitry required to decode the OOB signal state and state machines to decode bursts of OOB signals for SATA COM sequences (COMRESET, COMWAKE, and COMINIT).

The GTX_DUAL tile supports beacons that are PCI Express compliant by using interface signals defined in the *PHY Interface for the PCI Express (PIPE) Specification*. The FPGA logic decodes the beacon sequence.

Ports and Attributes

Table 7-13 defines the RX OOB/beacon signaling ports.

Table 7-13: **RX OOB/Beacon Signaling Ports**

Port	Direction	Clock Domain	Description
RXELECIDLE0 RXELECIDLE1	Out	Async	Indicates the differential voltage between RXN and RXP dropped below the minimum threshold (OOBDETECT_THRESHOLD). Signals below this threshold are OOB signals. 1: OOB signal detected. The differential voltage is below the minimum threshold. 0: OOB signal not detected. The differential voltage is above the minimum threshold. This port is intended for PCI Express and SATA standards.
RXSTATUS0[2:0] RXSTATUS1[2:0]	Out	RXUSRCLK2	The decoding of RXSTATUS[2:0] depends on the setting of RX_STATUS_FMT. When RX_STATUS_FMT = PCIE: 000: Receiver not present (when in receiver detection sequence)/Received data OK (during normal operation) 001: Reserved 010: Reserved 011: Receiver present (when in receiver detection sequence) 100: 8B/10B decode error. The PIPE requirement to send out an EDB (EndBad, K30.7) character on the RX output is supported. 101: RX Elastic Buffer Underflow. Stays asserted until cleared (different from that defined in the PIPE specification). 110: RX Elastic Buffer Overflow. Stays asserted until cleared (different from that defined in the PIPE specification). 111: Receive Disparity Error When RX_STATUS_FMT = SATA: RXSTATUS[0]: TXCOMSTART operation complete RXSTATUS[1]: COMWAKE signal received RXSTATUS[2]: COMRESET/COMINIT signal received
RXVALID0 RXVALID1	In	RXUSRCLK2	Indicates symbol lock and valid data on RXDATA and RXCHARISK[3:0] when High, as defined in the PIPE specification.

Table 7-14 defines the RX OOB/beacon signaling attributes.

Table 7-14: **RX OOB/Beacon Signaling Attributes**

Attribute	Type	Description
OOB_CLK_DIVIDER	Integer	Sets the squelch clock rate. The squelch clock must be set between 25 MHz and 37.5 MHz, as close to 25 MHz as possible for the SATA OOB detector to work correctly. Squelch Clock rate = CLKIN/OOB_CLK_DIVIDER Valid divider settings are 1, 2, 4, 6, 8, 10, 12, and 14.
OOBDETECT_THRESHOLD_0 OOBDETECT_THRESHOLD_1	3-bit Binary	Sets the minimum differential voltage between RXN and RXP. When the differential voltage drops below this level, the incoming signal is considered an OOB signal. This 3-bit binary encoded attribute has the following nominal values of the OOB threshold voltage(1):
		Value / **OOB Nominal Threshold Voltage [mV]**
		`000 – 101` / Not supported
		`110` (default) / 90
		`111` / 95
RX_STATUS_FMT_0 RX_STATUS_FMT_1	String	Defines which status encoding is used: PCIE: PCI Express encoding SATA: SATA encoding
SATA_BURST_VAL_0 SATA_BURST_VAL_1	3-bit Binary	Number of bursts required to declare a COM match. The default for SATA_BURST_VAL is 4, which is the burst count specified in SATA for COMINIT, COMRESET, and COMWAIT.
SATA_IDLE_VAL_0 SATA_IDLE_VAL_1	3-bit Binary	Number of idles required to declare a COM match. Each idle is an OOB signal with a length that matches either COMINIT/COMRESET or COMWAIT. When the SATA detector starts to count one type of idle (for example, COMRESET/COMINIT), it resets the count if it receives the other type. This value defaults to 3 to match the SATA specification.
SATA_MAX_BURST_0 SATA_MAX_BURST_1	Integer	Sets the threshold for the SATA detector to reject a burst in terms of squelch clock cycles. SATA_MAX_BURST has valid values between 1 and 61 (the default is 7) and must be greater than SATA_MIN_BURST. See the "Description" section to learn how to calculate the best value for a given squelch clock rate.
SATA_MAX_INIT_0 SATA_MAX_INIT_1	Integer	Sets the maximum time allowed for a COMINIT/COMRESET idle for the SATA detector in terms of squelch clock cycles. SATA_MAX_INIT has valid values between 1 and 61 (the default is 22) and must be greater than SATA_MIN_INIT. See the "Description" section to learn how to calculate the best value for a given squelch clock rate.
SATA_MAX_WAKE_0 SATA_MAX_WAKE_1	Integer	Sets the maximum time allowed for a COMWAKE idle for the SATA detector in terms of squelch clock cycles. SATA_MAX_WAKE has valid values between 1 and 61 (the default is 7) and must be greater than SATA_MIN_WAKE. See the "Description" section to learn how to calculate the best value for a given squelch clock rate.

Table 7-14: **RX OOB/Beacon Signaling Attributes *(Cont'd)***

Attribute	Type	Description
SATA_MIN_BURST_0 SATA_MIN_BURST_1	Integer	Sets the threshold for the SATA detector to reject a burst in terms of squelch clock cycles. SATA_MIN_BURST has valid values between 1 and 61 (the default is 4) and must be less than SATA_MAX_BURST. See the "Description" section to learn how to calculate the best value for a given squelch clock rate.
SATA_MIN_INIT_0 SATA_MIN_INIT_1	Integer	In SATA, OOB signals are used as idles in COMINIT, COMRESET, and COMWAKE. The minimum length of an idle that must be accepted for COMINIT/COMRESET signals in SATA is 304 ns. If the idle is shorter than 175 ns, it cannot be used for COMINIT/COMRESET. SATA_MIN_INIT is used to set the minimum time allowed for a COMINIT/COMRESET Idle for the SATA detector in terms of squelch clock cycles. SATA_MIN_INIT has valid values between 1 and 61 (the default is 12) and must be less than SATA_MAX_INIT. See the "Description" section to learn how to calculate the best value for a given squelch clock rate. The squelch clock is set based using OOB_CLK_DIVIDER.
SATA_MIN_WAKE_0 SATA_MIN_WAKE_1	Integer	In SATA, OOB signals are used as idles in COMINIT, COMRESET, and COMWAKE. The minimum length of an idle that must be accepted for COMWAKE signals in SATA is 101 ns. If the idle is shorter than 55 ns, it cannot be used for COMWAKE. SATA_MIN_WAKE is used to set the minimum time allowed for a COMWAKE idle for the SATA detector in terms of squelch clock cycles. SATA_MIN_WAKE has valid values between 1 and 61 (the default is 4) and must be less than SATA_MAX_WAKE. See the "Description" section to learn how to calculate the best value for a given squelch clock rate. The squelch clock is set based using OOB_CLK_DIVIDER.

Notes:

1. OOB nominal values are shown, consult DS202: *Virtex-5 FPGA Data Sheet* for OOB specifications.

Description

The GTX_DUAL tile supports two RX status modes: one for SATA operation and one for PCI Express operation. The decoding mode is configured by the RX_STATUS_FMT attribute.

Detecting Electrical Idle for PCI Express Operation

Regardless of the state of the RX_STATUS_FMT attribute, the RXELECIDLE port indicates if the differential voltage between the RXN and RXP serial I/O pins falls within the OOB threshold defined by the OOBDETECT_THRESHOLD attribute. This signal can be used to decode PCI Express beacon sequences. The latency between the arrival of an OOB signal and the assertion of RXELECIDLE is found in the *Virtex-5 FPGA Data Sheet*.

Detecting OOB for SATA Operation

Each GTX transceiver includes a SATA OOB detector to decode SATA COM sequences. When RX_STATUS_FMT is set to *SATA*, the pins of the RXSTATUS port are used to indicate the arrival of COM sequences. Figure 7-6 shows the SATA OOB detector. It divides CLKIN down to create a squelch clock that runs at approximately 25 MHz. This clock is used to sample the output of the block that detects OOB signals to look for transitions between regular data and OOB signals. Both edges of the squelch clock are used. The squelch detector FSM uses the transitions to calculate the length of each burst and each idle. It uses this information to drive the RXSTATUS port, indicating which COM sequences have been found.

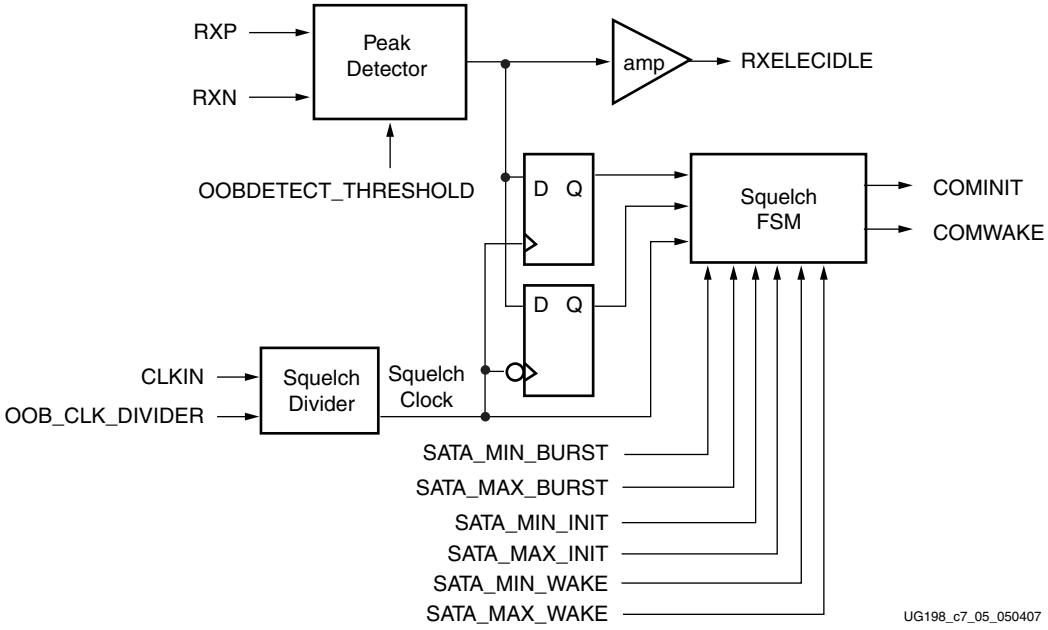


Figure 7-6: **SATA OOB Detector Block Diagram**

Before the SATA OOB detector can work, it must be configured to use the reference clock provided (CLKIN). The OOB_CLK_DIVIDER attribute must be set to produce a squelch clock between 25 MHz and 37.5 MHz, but as close to 25 MHz as possible. In addition, the MIN and MAX times for bursts and idles must be set based on the squelch clock rate. The formula to set the squelch clock is shown in Equation 7-1.

$$\text{Parameter in squelch cycles} = \left(\frac{\text{Parameter in ns}}{1000}\right) \times \text{squelch clock frequency in MHz} \times 2 \qquad \textit{Equation 7-1}$$

All the minimum values are defined with a minimum time below which the signal must be rejected, and a minimum time above which the signal always meets the minimum time requirement. After calculating each of these in terms of squelch clock cycles, the appropriate MIN parameter is set to an integer value between these two numbers.

Similarly, for all maximum values, there is a maximum time above which the signal must be rejected, and a maximum time below which the signal always meets the maximum time requirement. After calculating these in terms of squelch clock, the appropriate MAX parameter is set to an integer value between these two numbers.

Example

Table 7-15 shows some example settings for the squelch clock divider.

Note: The RocketIO GTX Transceiver Wizard automatically sets the OOB_CLK_DIVIDER attribute based on the provided reference clock frequency.

Table 7-15: **Example Clock Generation Settings for SATA**

Parameter	Example 1	Example 2	Example 3	Example 4	Example 5	Example 6
CLKIN (MHz)	25	75	100	150	250	300
OOB_CLK_DIVIDER	1	2	4	6	10	12
Squelch Clock (MHz)	25	37.5	25	25	25	25
Effective Sample Period (ns)	20	13.33333333	20	20	20	20

Table 7-16 shows all the minimum and maximum values defined in SATA for burst and idle lengths, along with example calculations based on the squelch clock frequencies computed in Table 7-15.

Table 7-16: **Example SATA Attribute Settings**

Parameter	ns	Cycles	Cycles	Cycles	Cycles	Cycles	Cycles
Shortest burst width that must be rejected	55	2.8	4.1	2.8	2.8	2.8	2.357142857
MinBurstWidth		**4**	**6**	**4**	**4**	**4**	**3**
Shortest burst width that must be accepted	101	5.1	7.6	5.1	5.1	5.1	4.341428571
Nominal burst length	107	5.3	8.0	5.3	5.3	5.3	4.572857143
Longest burst width that must be accepted	112	5.6	8.4	5.6	5.6	5.6	4.8
MaxBurstWidth		**7**	**11**	**7**	**7**	**7**	**6**
Longest burst width that must be rejected	175	8.8	13.1	8.8	8.8	8.8	7.5
Shortest idle width that must be rejected for COMINIT	175	8.8	13.1	8.8	8.8	8.8	7.5
MinInitWidth		**12**	**18**	**12**	**12**	**12**	**10**
Shortest idle width that must be accepted for COMINIT	304	15.2	22.8	15.2	15.2	15.2	13.02857143
Nominal idle width for COMINIT	320	16.0	24.0	16.0	16.0	16.0	13.71428571
Longest idle width that must be accepted for COMINIT	336	16.8	25.2	16.8	16.8	16.8	14.4
MaxInitWidth		**22**	**32**	**22**	**22**	**22**	**18**
Longest idle width that must be rejected for COMINIT	525	26.3	39.4	26.3	26.3	26.3	22.5
Shortest idle width that must be rejected for COMWAKE	55	2.8	4.1	2.8	2.8	2.8	2.357142857
MinWakeWidth		**4**	**6**	**4**	**4**	**4**	**3**
Shortest idle width that must be accepted for COMWAKE	101.3	5.1	7.6	5.1	5.1	5.1	4.341428571

Table 7-16: **Example SATA Attribute Settings *(Cont'd)***

Parameter	ns	Cycles	Cycles	Cycles	Cycles	Cycles	Cycles
Nominal idle width for COMWAKE	106.7	5.3	8.0	5.3	5.3	5.3	4.572857143
Longest idle width that must be accepted for COMWAKE	112	5.6	8.4	5.6	5.6	5.6	4.8
MaxWakeWidth		**7**	**11**	**7**	**7**	**7**	**6**
Longest idle width that must be rejected for COMWAKE	175	8.8	13.1	8.8	8.8	8.8	7.5

RX Clock Data Recovery

Overview

The RX Clock Data Recovery (CDR) circuit in each GTX transceiver extracts a recovered clock from incoming data. As long as the line rate of the recovered clock matches the line rate of the receiver within a tolerance band as specified in the *Virtex-5 FPGA Data Sheet*, and there are sufficient transitions in the data, the CDR can extract a clock. The CDR has advanced features, including a horizontal sample point shift feature that can be used to evaluate the horizontal eye opening of the received signal.

Ports and Attributes

Table 7-17 defines RX CDR signaling ports.

Table 7-17: **RX CDR Signaling Ports**

Port	Dir	Clock Domain	Description
RESETDONE0 RESETDONE1	Out	Async	This port goes High when the GTX transceiver has finished reset and is ready for use. For this signal to work correctly, CLKIN and all clock inputs on the individual GTX transceiver (TXUSRCLK, TXUSRCLK2, RXUSRCLK, RXUSRCLK2) must be driven.
RXCDRRESET0(1) RXCDRRESET1(2)	In	RXUSRCLK2	Individual reset signal for the RX CDR and the RX part of the PCS for this channel. This signal is driven High to cause the CDR to give up its current lock and return to the shared PMA PLL frequency.

Notes:

1. When this reset is active, then RESETDONE0 is driven Low. All resets are asynchronous, positive-edge triggered, and synchronized internally to a specific clock domain.
2. When this reset is active, then RESETDONE1 is driven Low. All resets are asynchronous, positive-edge triggered, and synchronized internally to a specific clock domain.

Table 7-18 defines the RX CDR attributes.

Table 7-18: **RX CDR Attributes**

Attribute	Type	Description
CDR_PH_ADJ_TIME	5-bit Binary	Defined the delay after deassertion of the CDR phase reset before the optional reset sequence of PCI Express operation is complete during electrical idle.
PMA_CDR_SCAN_0 PMA_CDR_SCAN_1	27-bit Hex	This 27-bit attribute allows direct control of the CDR sampling point. • When OVERSAMPLE_MODE = TRUE, set PMA_CDR_SCAN to `27'h6404040`. • When OVERSAMPLE_MODE = FALSE, set PMA_CDR_SCAN as follows: ♦ When PLL_RXDIVSEL_OUT = 2 or 4, set PMA_CDR_SCAN to `27'h6404035` ♦ When PLL_RXDIVSEL_OUT = 1, set PMA_CDR_SCAN per the following equations: PMA_CDR_SCAN[26:8] = `19'h64040` PMA_CDR_SCAN[7:0] = 64 – Round(1.792 * RX Line Rate)(1)
PMA_RX_CFG_0 PMA_RX_CFG_1	25-bit Hex	This 25-bit attribute allows the operation of the CDR to be adjusted. In normal operation, set this attribute to its default value. • For 5x Oversampling operation, set PMA_RX_CFG = `25'0F44000`. • For lock-to-reference operation, set PMA_RX_CFG = `25'0F44000`. • For synchronous operation, set PMA_RX_CFG = `25'0F44088`. • For ±100 PPM operation, set PMA_RX_CFG = `25'0F44088`. • For ±1000 PPM operation, set PMA_RX_CFG = `25'0F44089`.
RX_EN_IDLE_HOLD_CDR	Boolean	Enables the CDR to hold data during an optional reset sequence of an electrical idle state for PCI Express operation.

Table 7-18: **RX CDR Attributes *(Cont'd)***

Attribute	Type	Description
RX_EN_IDLE_RESET_FR	Boolean	When TRUE, enables the reset of the CDR frequency circuits during an optional reset sequence of an electrical idle state for PCI Express operation.
RX_EN_IDLE_RESET_PH	Boolean	When TRUE, enables the reset of the CDR phase circuits during an optional reset sequence of an electrical idle state for PCI Express operation.

Notes:

1. RX Line Rate in Gb/s.

Description

Before serial data received from the line can be used, the embedded clock in the signal must be recovered. The CDR circuit in each GTX transceiver is responsible for this function. It takes a divided, high-speed serial clock from the shared PMA PLL and adjusts its phase and frequency until its transitions match the incoming data. As shown in Figure 7-7, the result is a clock that matches the clock originally used to generate the serial stream.

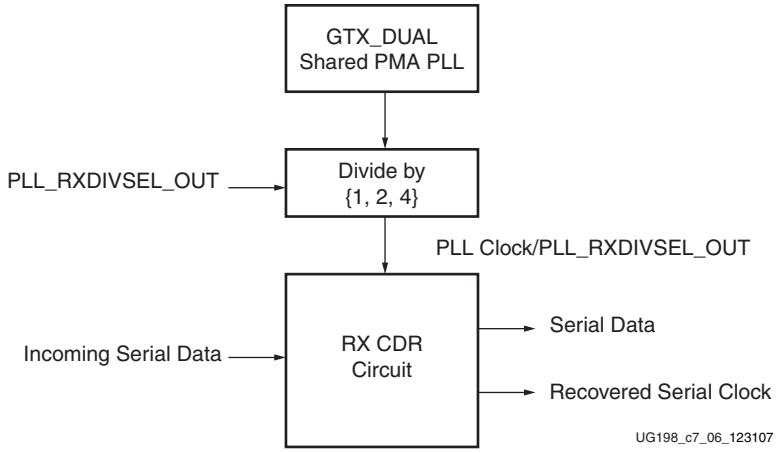


Figure 7-7: **Conceptual View of RX CDR Circuit**

Because transitions in the incoming data are used to recover the serial clock, long runs without transitions can introduce error.

CDR Reset

The CDR must be reset before it can operate on incoming data. There are several ways to reset the CDR:

- Use the GTXRESET port to reset all components in the GTX_DUAL tile, including the CDR in each transceiver. See "Reset," page 98 for more details.
- Use the RXCDRRESET port to reset the CDR block, the OOB circuits for SATA (see "RX OOB/Beacon Signaling," page 170), the RX elastic buffer (see "Configurable RX Elastic Buffer and Phase Alignment," page 202), and the remaining sections of the RX PCS.

Figure 7-8 shows the timing of the internal reset signals when RXCDRRESET is asserted. RXCDRRESET can be asserted asynchronously. When it is asserted, an internal CDR reset pulse, synchronized to an internally generated 1 MHz clock, resets the CDR. Similarly, a reset pulse is generated for the SATA OOB circuit (internal SATA reset), the RX PCS datapath (internal RXRESET), and the RX elastic buffer (internal RXBUFRESET). The entire sequence completes in approximately 5 µs.

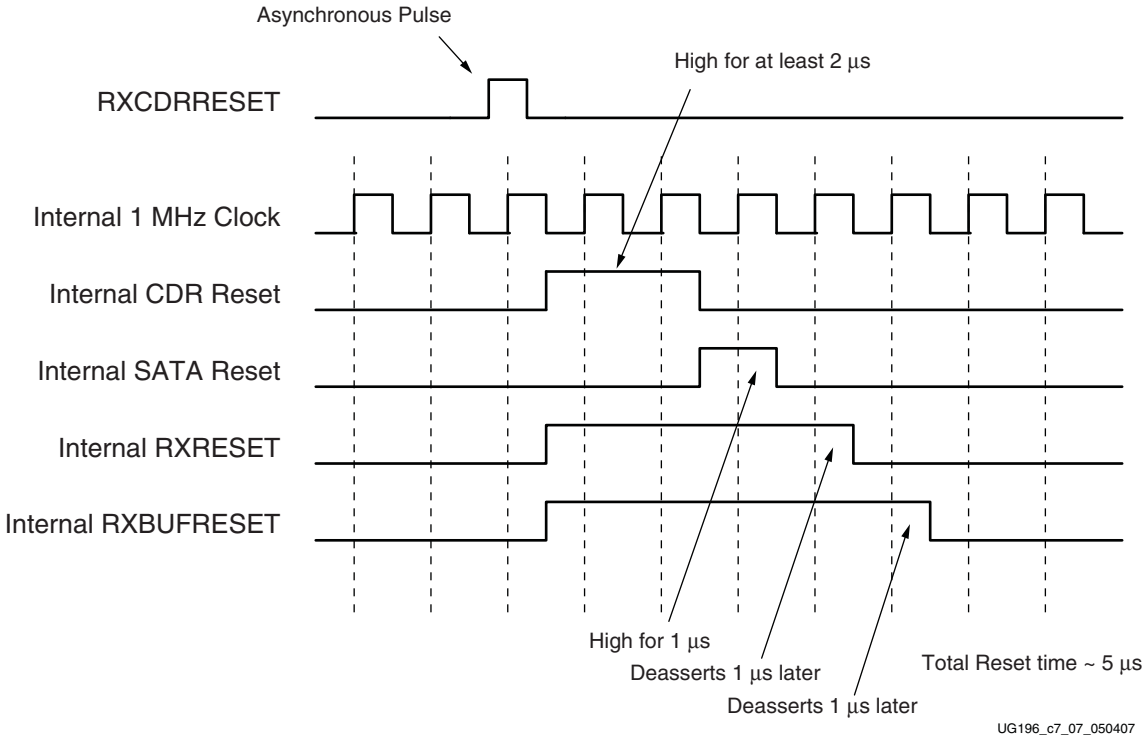


Figure 7-8: **Reset Sequence Triggered by RXCDRRESET**

Horizontal Sample Point Shift

One advanced feature of the CDR of the GTX transceiver is the built-in horizontal sample point shift. During normal operation, the CDR finds the transition points in incoming data and uses them to recover the frequency of the incoming clock.

The transition points are also used to select the optimal time to sample data. To minimize the chance of error, the CDR attempts to sample data as far from transition points as possible (that is, at the time when the bit value is most stable). This position is the center of the data eye (see Figure 7-9).

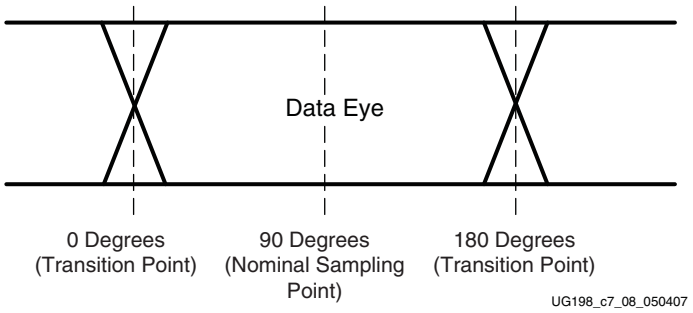


Figure 7-9: **Normal CDR Operation**

A rough measure of the quality of the incoming signal, as seen by the CDR, can be obtained by determining how close to the transition points of the eye the sampling point can be before the number of received bit errors increases significantly. Figure 7-10 shows an example of a possible scan with an eye from clean data (good signal integrity) versus a scan of bad data.

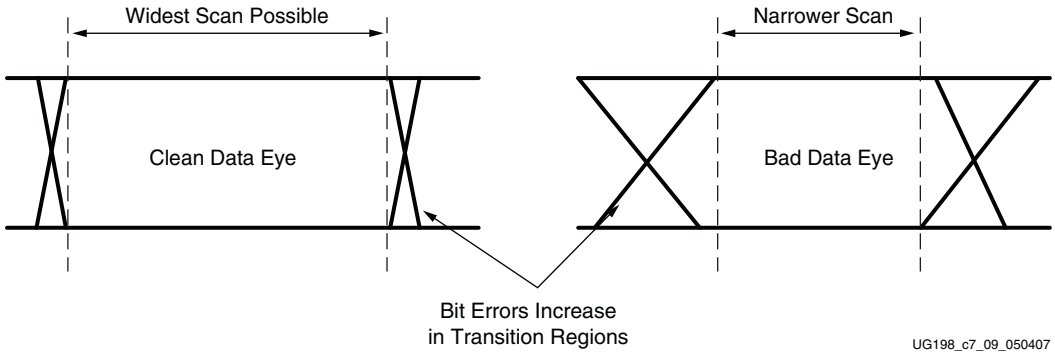


Figure 7-10: **Scanning to Evaluate the Data Eye**

This functionality is supported through the PMA_CDR_SCAN attribute. Dynamic scans are performed by changing PMA_CDR_SCAN using the DRP while receiving known data, such as a PRBS pattern to determine the number of bit errors.

PMA_CDR_SCAN has an 8-bit field that controls the position of the data sampling point relative to the first transition point in the eye. Resetting the field to 0 puts the sampling point at the first transition point. Setting the field to 127 puts the data sampling point at the 180° point, the second transition point of the eye (see Figure 7-9). The default value for each field is 64, the nominal 90° sampling point.

There are three fields to accommodate the different possible PLL_RXDIVSEL_OUT settings. Table 7-19 shows how to set PMA_CDR_SCAN for PLL_RXDIVSEL_OUT = 1.

Note: The horizontal sample point shift is not supported for PLL_RXDIVSEL_OUT = 2 or PLL_RXDIVSEL_OUT = 4.

Table 7-19: **Settings for PMA_CDR_SCAN based on PLL_RXDIVSEL_OUT**

PLL_RXDIVSEL_OUT	PMA_CDR_SCAN					
	[26]	[25]	[24]	[23:16]	[15:8]	[7:0]
1	1	1	0	8'h40 (1)	8'h40 (1)	8'h00 - 8'h80 (2)
2	1	1	0	8'h40 (1)	8'h40 (1)	8'h35 (1)
4	1	1	0	8'h40 (1)	8'h40 (1)	8'h35 (1)

Notes:

1. Must be left at the RocketIO GTX Transceiver Wizard default setting.
2. See Table 7-18 for normal operation (when not using the built-in horizontal sample point shift).

Serial In to Parallel Out

Overview

The Serial In to Parallel Out (SIPO) block deserializes serial data from the GTX receiver PMA and presents it as parallel data to the PCS.

Ports and Attributes

Table 7-20 defines the SIPO ports.

Table 7-20: **SIPO Ports**

Port	Dir	Clock Domain	Description
INTDATAWIDTH	In	Async	Specifies the width of the internal datapath for the entire GTX_DUAL tile. This shared port is also described in "Shared PMA PLL," page 84. 0: Internal datapath is 16 bits wide 1: Internal datapath is 20 bits wide

Table 7-21 defines the SIPO attributes.

Table 7-21: **SIPO Attributes**

Attribute	Type	Description
OVERSAMPLE_MODE	Boolean	This shared attribute activates the built-in 5x digital oversampling circuits in both GTX_DUAL transceivers. Oversampling is supported between $1/10^{th}$ of the lower border of the shared PMA PLL operating range and $4/10^{th}$ of the upper border of the shared PMA PLL operating range. For data rates using a PLL clock without oversampling and are below one-half of the lower border of the shared PMA PLL, oversampling is mandatory to ensure that the shared PMA PLL operated in its frequency range. Oversampling must be enabled when running the GTX transceivers at line rates between 150 Mb/s and 750 Mb/s. TRUE: Built-in 5x digital oversampling enabled for both GTX transceivers on the tile FALSE: Digital oversampling disabled See "Oversampling," page 183 for more details about 5x digital oversampling.
PLL_RXDIVSEL_OUT_0 PLL_RXDIVSEL_OUT_1	Integer	This divider defines the nominal line rate for the receiver. It can be set to 1, 2, or 4. RX Line Rate = PLL Clock * 2/PLL_RXDIVSEL_OUT

Description

The SIPO block is the heart of the RX datapath. It uses both edges of a high-speed clock to deserialize incoming data and present it to the PCS.

The serial clock rate to the SIPO, which is the RX line rate of the GTX transceiver, depends on the PLL clock rate and the setting of PLL_RXDIVSEL_OUT in the GTX transceiver. Equation 7-2 shows how to set the RX line rate. See "Shared PMA PLL," page 84 for more information on how to set the PLL clock rate.

$$\text{RX Line Rate} = \frac{\text{PLL Clock} \times 2}{\text{PLL\_RXDIVSEL\_OUT}} \qquad \textit{Equation 7-2}$$

The parallel clock rate to the parallel side of the SIPO is the XCLK rate of the GTX transceiver. This rate matches the USRCLK rate of the GTX transceiver, which is used internally in the PCS. See "Configurable RX Elastic Buffer and Phase Alignment," page 202 for more details about the clock domains in the RX side of the GTX transceiver. The XCLK rate depends on the internal datapath width used in the tile, because this value is the width of the parallel data the SIPO produces. Equation 7-3 shows how to calculate the parallel (XCLK) rate of the SIPO.

$$\text{RX XCLK Rate} = \frac{\text{RX Line Rate}}{\text{Internal Datapath Width}} \qquad \textit{Equation 7-3}$$

When OVERSAMPLE_MODE is FALSE, the internal datapath width is 16 bits when INTDATAWIDTH is Low and is 20 bits when INTDATAWIDTH is High. When OVERSAMPLE_MODE is TRUE, the internal datapath width is 20 bits. See "Oversampling," page 183 for more details about receiver operation when oversampling is activated.

Both the serial and parallel clocks for the SIPO are generated from the recovered clock in the CDR circuit.

Oversampling

Overview

Each GTX transceiver includes built-in 5x oversampling to enable serial rates from $1/10^{th}$ of the lower border of the frequency range of the shared PMA PLL up to $4/10^{th}$ of the upper border of the shared PMA PLL(1). At these low rates, the regular CDR must operate at five times the desired line rate to stay within its operating limits.

The digital oversampling circuit takes parallel data from the SIPO at five times the desired line rate and uses the position of bit value transitions to recover a clock. The transition points are also used to pick an optimal sampling point to recover 4 bits of data from each set of 20 bits presented.

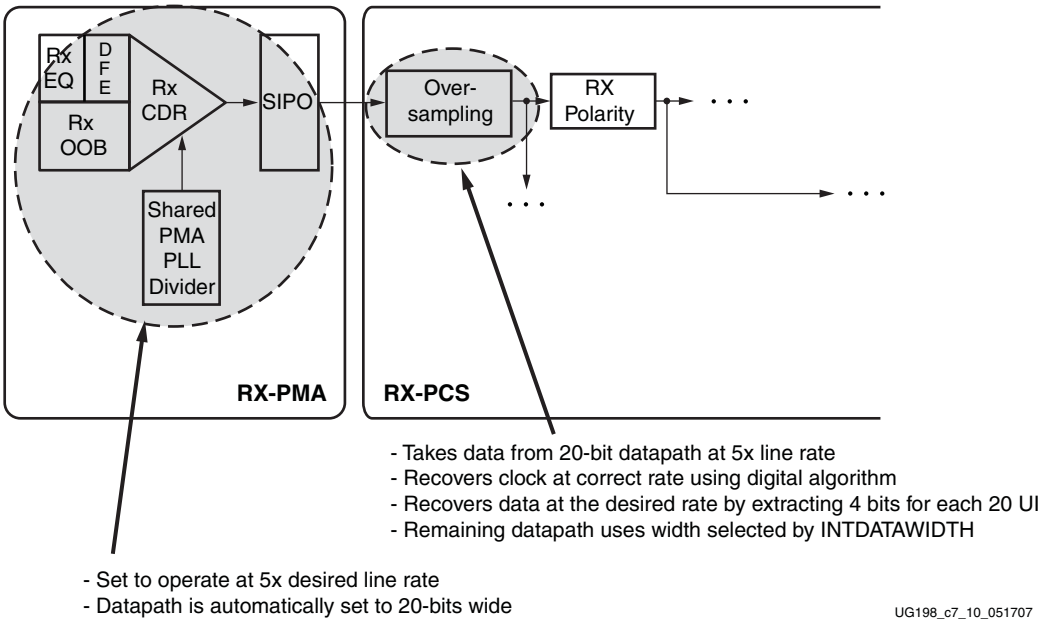


Figure 7-11: **GTX RX Block Diagram - Oversampling**

1. Lower border calculation: 2/(4 x 5) = 1/10. Upper border calculation: 2/(1 x 5) = 2/5 = 4/10. Calculations are based on Equation 7-4 and Equation 7-5 using a maximum divider setting (4) for the lower border and a minimal divider setting for the upper border.

Ports and Attributes

Table 7-22 defines the ports for built-in digital oversampling.

Table 7-22: **RX DCDR Ports**

Port	Dir	Clock Domain	Description
RXENSAMPLEALIGN0 RXENSAMPLEALIGN1	In	RXUSRCLK2	When High, the 5x oversampler in the PCS continually adjusts its sample point. When Low, it samples only at the point that was active before the port went Low.
RXOVERSAMPLEERR0 RXOVERSAMPLEERR1	Out	RXUSRCLK2	When High, indicates the FIFO in the oversampling circuit has either overflowed or underflowed. The PCS must be reset to resume proper operation.

Table 7-23 defines the attributes for built-in digital oversampling.

Table 7-23: **RX DCDR Attributes**

Attribute	Type	Description
OVERSAMPLE_MODE	Boolean	This shared attribute is also defined in "Shared PMA PLL," page 84 and "TX Buffering, Phase Alignment, and TX Skew Reduction," page 138. It applies to both sides of both GTX transceivers on the GTX_DUAL tile. When TRUE, 5X oversampling is On.
PMA_RX_CFG_0 PMA_RX_CFG_1	25-bit Hex	This 25-bit attribute allows the operation of the CDR to be adjusted. In normal operation, set this attribute to its default value. In oversampling mode, set PMA_RX_CFG = `25'h0F44000`.

Description

Each GTX transceiver includes a built-in 5x digital oversampling circuit to enable serial rates from $1/10^{th}$ of the lower border of the frequency range of the shared PMA PLL up to $4/10^{th}$ of the upper border of the shared PMA PLL(1). Oversampling applies to both transceivers in a GTX_DUAL tile. If oversampling is activated for one transceiver, it is activated for both.

Configuring the GTX transceiver to use oversampling requires the following steps:

- Configuring the 5x line rate
- Configuring the PCS internal datapath and clocks
- Activating and operating the oversampling block

The RocketIO GTX Transceiver Wizard automatically configures the GTX_DUAL tile and makes the oversampling ports available when generating a GTX wrapper with oversampling enabled.

1. Lower border calculation: 2/(4 x 5) = 1/10. Upper border calculation: 2/(1 x 5) = 2/5 = 4/10. Calculations are based on Equation 7-4 and Equation 7-5 using a maximum divider setting (4) for the lower border and a minimal divider setting for the upper border.

Configuring the 5x Line Rate

The SIPO in the RX PMA must provide the oversampling block with 20 bits per parallel cycle, sampled at a rate five times faster than the desired line rate. The required line rate for the PMA is given by Equation 7-4.

$$PMALineRate = 5 \times DesiredLineRate \quad \text{Equation 7-4}$$

To achieve the required line rate, the RX divider for the transceiver must be set so that the resulting required PLL clock rate is within the PLL operating range. Refer to the *Virtex-5 FPGA Data Sheet* for the specified PLL operating range. Equation 7-5 shows the relationship between the required rate and the PLL clock. "Serial In to Parallel Out," page 181 provides more information about the local RX divider.

$$f_{PLLClock} = \frac{PMALineRate \times PLL\_RXDIVSEL\_OUT}{2} \quad \text{Equation 7-5}$$

The shared PMA PLL, which is discussed in detail in "Shared PMA PLL," page 84, must be set to produce the required PLL clock frequency. Equation 7-6 shows how the PLL clock frequency is related to the frequency of CLKIN (the reference clock to the tile). Oversampling mode automatically uses a 20-bit internal datapath in the PMA, regardless of the INTDATAWIDTH setting. The lowest possible ratio of PLL_DIVSEL_FB to PLL_DIVSEL_REF is recommended.

$$f_{PLLClock} = f_{CLKIN} \times \frac{5 \times PLL\_DIVSEL\_FB}{PLL\_DIVSEL\_REF} \quad \text{Equation 7-6}$$

Configuring the PCS Internal Datapath and Clocks

Figure 7-12 shows the clock domains of the GTX RX datapath when oversampling is used. The RX serial clock runs at the PMA line rate calculated previously. The XCLK runs at the resulting parallel rate for a 20-bit datapath, PMALineRate/20. The oversampled data from the SIPO is fed into the oversampling block at the XCLK rate.

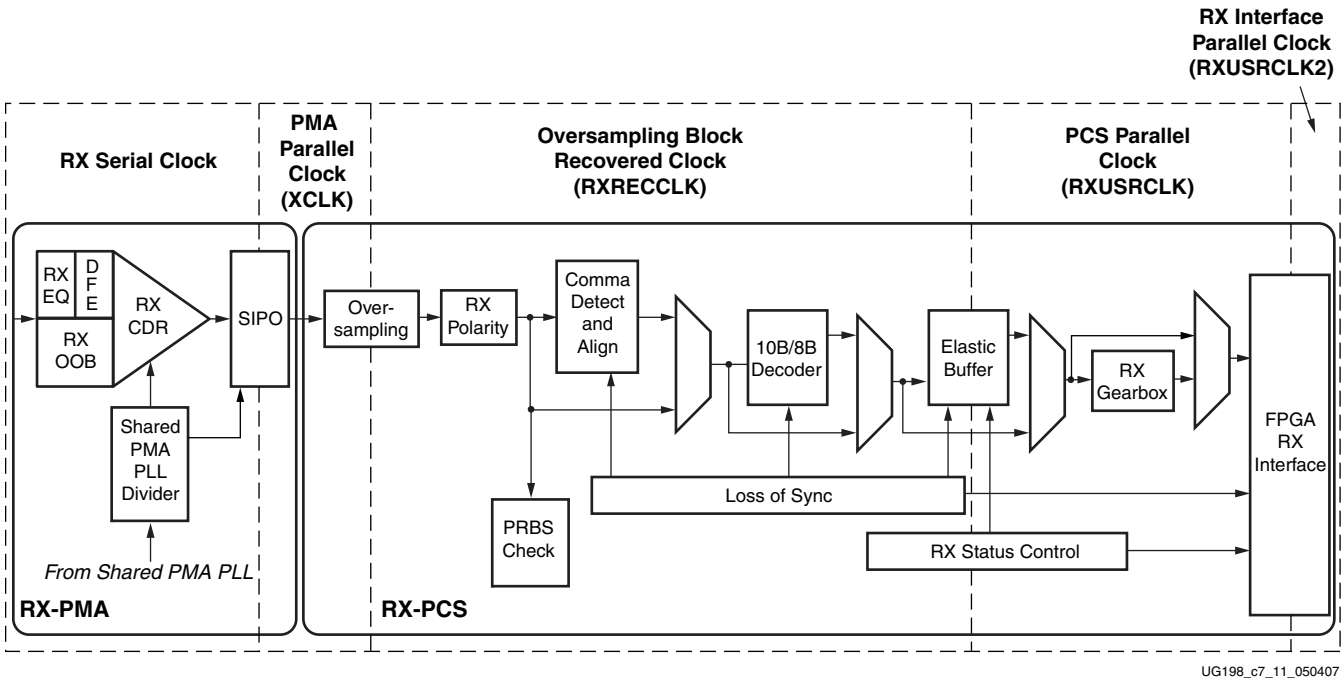


Figure 7-12: **RX Clock Domains When Using Built-In Oversampling**

The oversampling block produces 4 bits of data for every 20 bits received. This data is fed into the remaining PCS datapath at the required clock rate for the desired line rate (given by Equation 7-5). This PCS rate is used for RXUSRCLK. A multiple of the rate is used for RXUSRCLK2, depending on the selected RX datapath width.

When oversampling is used, the oversampled line rate, not the PMA line rate, and the PCS internal datapath width, which is set by INTDATAWIDTH, must be used for RXUSRCLK calculations. "FPGA RX Interface," page 231 provides more details about the relationship between RXUSRCLK and RXUSRCLK2.

Activating and Operating the Oversampling Block

After the PMA line rate and PCS datapath are set, the oversampling block can be enabled by setting OVERSAMPLING_MODE to TRUE. This attribute affects both transceivers in the GTX_DUAL tile.

The oversampling block includes a small buffer to hold data before passing it to the PCS. This buffer can overflow/underflow if the PMA and PCS frequencies are different, such as if RXUSRCLK stopped temporarily due to other events in the system. If an error in the oversampling block buffer occurs, the transceiver asserts the RXOVERSAMPLEERR signal. This error can be cleared by asserting RXRESET or RXCDRRESET.

RXENSAMPLEALIGN is tied High so the oversampling block always attempts to find the best possible recovered clock and sample point in the incoming data.

RX Polarity Control

Overview

The GTX RX can invert incoming data using the RX polarity control function. This function is useful in designs where the RXP and RXN signals can be accidentally connected in reverse. The RXPOLARITY port is driven High to invert the polarity of incoming data.

Ports and Attributes

Table 7-24 defines the RX polarity ports.

Table 7-24: **RX Polarity Ports**

Port	Dir	Clock Domain	Description
RXPOLARITY0 RXPOLARITY1	In	RXUSRCLK2	The RX polarity port is used to invert the polarity of incoming data. 1: Invert polarity 0: Regular polarity

There are no attributes in this section.

Description

The RX polarity port is used to invert the polarity of incoming data. Driving this port High causes the RXN pin to be treated as RXP and the RXP pin to be treated as RXN.

PRBS Detection

Overview

The GTX receiver includes a built-in PRBS checker. This checker can be set to check for one of three industry-standard PRBS patterns. The checker is self-synchronizing and works on the incoming data before comma alignment or decoding. This function can be used to test the signal integrity of the channel.

Ports and Attributes

Table 7-25 defines the PRBS detection ports.

Table 7-25: **PRBS Detection Ports**

Port	Dir	Clock Domain	Description
INTDATAWIDTH	In	Async	Specifies the width of the internal datapath for the entire GTX_DUAL tile.
PRBSCNTRESET0 PRBSCNTRESET1	In	RXUSRCLK2	Resets the PRBS error counter. PRBSCNTRESET is applied synchronously.
RXENPRBSTST0[1:0] RXENPRBSTST1[1:0]	In	RXUSRCLK2	Receiver test pattern checker control: 00: Disable PRBS checkers 01: Enable 2^7-1 PRBS checker 10: Enable 2^{23}-1 PRBS checker 11: Enable 2^{31}-1 PRBS checker
RXPRBSERR0 RXPRBSERR1	Out	RXUSRCLK2	RXPRBSERR goes High when the number of errors in PRBS testing exceeds the value set by the PRBS_ERR_THRESHOLD attribute.

Table 7-26 defines the PRBS detection attributes.

Table 7-26: **PRBS Detection Attributes**

Attribute	Type	Description
PRBS_ERR_THRESHOLD0 PRBS_ERR_THRESHOLD1	32-bit Hex	Sets the error threshold for the PRBS checker. If PRBS testing is enabled, a counter counts the number of errors. If the number of errors exceeds the value of PRBS_ERR_THRESHOLD, the output RXPRBSERR goes High. This attribute is set as a 32-bit hex value.

Description

To use the built-in PRBS checker, RXENPRBSTST is set to match the PRBS pattern being sent to the receiver. The RXENPRBSTST entry in Table 7-25 shows the available settings.

When the PRBS checker is running, it attempts to find the selected PRBS pattern in the incoming data. When it finds the pattern, it can detect PRBS errors by comparing the incoming pattern with the expected pattern.

The checker counts the number of errors it sees and compares it with PRBS_ERR_THRESHOLD. When the error count exceeds the threshold, RXPRBSERR is asserted. Asserting PRBSCNTRESET clears RXPRBSERR. GTXRESET, RXCDRRESET, and RXRESET also reset the count.

Configurable Comma Alignment and Detection

Overview

Serial data must be aligned to symbol boundaries before it can be used as parallel data. To make alignment possible, transmitters send a recognizable sequence, usually called a *comma*. The receiver searches for the comma in the incoming data. When it finds a comma, it moves the comma to a byte boundary so the received parallel words match the transmitted parallel words.

Figure 7-13 shows the alignment to a 10-bit comma. The TX parallel data is on the left. The serial data with the comma is highlighted in the middle. The RX receiving unaligned bits are on the right side.

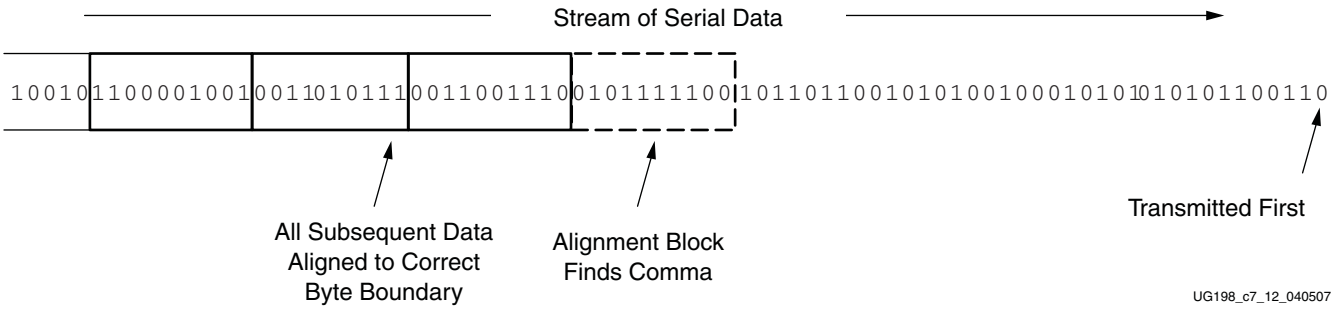


Figure 7-13: **Conceptual View of Comma Alignment (Aligning to a 10-Bit Comma)**

Figure 7-14 shows the TX parallel data is on the left side, and the RX receiving recognizable parallel data is on the right side.

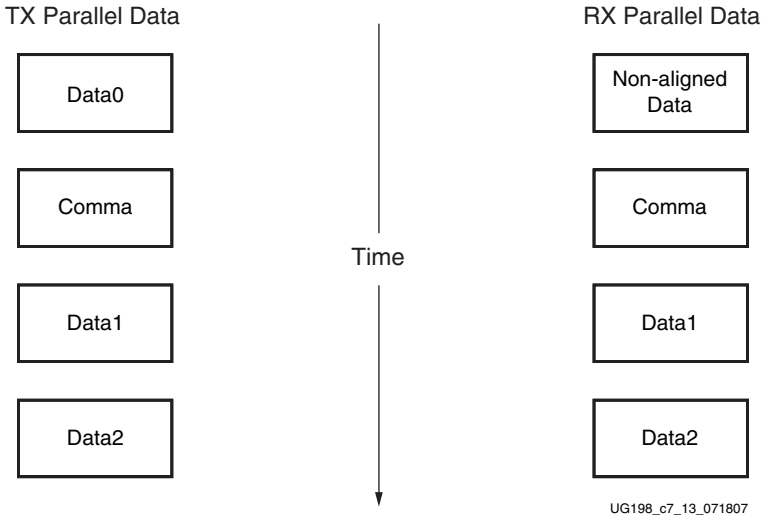


Figure 7-14: **Parallel Data View of Comma Alignment**

The GTX transceiver includes an alignment block that can be programmed to align specific commas to various byte boundaries, or to manually align data using attribute settings (see Table 7-28, page 191). SONET A1/A2 alignment is possible using comma double mode. The block can be bypassed to reduce latency if it is not needed.

Ports and Attributes

Table 7-27 defines the RX comma alignment and detection ports.

Table 7-27: **RX Comma Alignment and Detection Ports**

Port	Dir	Clock Domain	Description
RXBYTEISALIGNED0 RXBYTEISALIGNED1	Out	RXUSRCLK2	This signal from the comma detection and realignment circuit is High to indicate that the parallel data stream is properly aligned on byte boundaries according to comma detection. 0: Parallel data stream not aligned to byte boundaries 1: Parallel data stream aligned to byte boundaries There are several cycles after RXBYTEISALIGNED is asserted before aligned data is available at the FPGA RX interface. RXBYTEISALIGNED responds to plus comma alignment when PCOMMA_ALIGN is TRUE. RXBYTEISALIGNED responds to minus comma alignment when MCOMMA_ALIGN is TRUE.
RXBYTEREALIGN0 RXBYTEREALIGN1	Out	RXUSRCLK2	This signal from the comma detection and realignment circuit indicates that the byte alignment within the serial data stream has changed due to comma detection. 0: Byte alignment has not changed 1: Byte alignment has changed Data can be lost when alignment occurs, which can cause data errors (and disparity errors when the 8B/10B decoder is used).
RXCOMMADET0 RXCOMMADET1	Out	RXUSRCLK2	This signal is asserted when the comma alignment block detects a comma. The assertion occurs several cycles before the comma is available at the FPGA RX interface. 0: Comma not detected 1: Comma detected
RXCOMMADETUSE0 RXCOMMADETUSE1	In	RXUSRCLK2	RXCOMMADETUSE activates the comma detection and alignment circuit. 0: Bypass the circuit 1: Use the comma detection and alignment circuit Bypassing the comma and alignment circuit reduces RX datapath latency.
RXENMCOMMAALIGN0 RXENMCOMMAALIGN1	In	RXUSRCLK2	Aligns the byte boundary when *comma minus* is detected. 0: Disabled 1: Enabled

Table 7-27: **RX Comma Alignment and Detection Ports *(Cont'd)***

Port	Dir	Clock Domain	Description
RXENPCOMMAALIGN0 RXENPCOMMAALIGN1	In	RXUSRCLK2	Aligns the byte boundary when *comma plus* is detected. 0: Disabled 1: Enabled
RXSLIDE0 RXSLIDE1	In	RXUSRCLK2	RXSLIDE implements a comma alignment *bump* control. When RXSLIDE is asserted, the byte alignment is adjusted by one bit, which permits determination and control of byte alignment by the FPGA logic. Each assertion of RXSLIDE causes just one adjustment. RXSLIDE must be deasserted for two RXUSRCLK2 cycles before it can be reasserted to cause another adjustment. When asserted, RXSLIDE takes precedence over normal comma alignment.

Table 7-28 defines the RX comma alignment and detection attributes.

Table 7-28: **RX Comma Alignment and Detection Attributes**

Attribute	Type	Description
ALIGN_COMMA_WORD_0 ALIGN_COMMA_WORD_1	Integer	Controls alignment of detected commas within a multi-byte datapath. 1: Align comma to either byte within a two-byte datapath. The comma can be aligned to either the *even* byte [9:0] or the *odd* byte [19:10] of RXDATA at the FPGA when the two-byte RX interface is selected. 2: Align comma to the *even* byte within a two-byte datapath. The aligned comma is guaranteed to be aligned to byte RXDATA [9:0]. For ALIGN_COMMA_WORD = 2 to work properly in conjunction with the elastic buffer, both CLK_COR_ADJ_LEN and CLK_COR_MIN_LAT must be even. Protocols that send commas in even and odd positions must set ALIGN_COMMA_WORD to 1. If RXDATAWIDTH is driven Low, ALIGN_COMMA_WORD must be set to 1.
COMMA_10B_ENABLE_0 COMMA_10B_ENABLE_1	10-bit Binary	Sets which bits of MCOMMA/PCOMMA must be matched to incoming data and which bits can be any value. This attribute is a 10-bit mask with a default value of 1111111111. Any bit in the mask that is reset to 0 effectively turns the corresponding bit in MCOMMA or PCOMMA to a don't care bit.
COMMA_DOUBLE_0 COMMA_DOUBLE_1	Boolean	Specifies whether a comma match consists of either a comma plus or a comma minus alone, or whether both are required in the sequence. FALSE: The plus comma (PCOMMA) and minus comma (MCOMMA) are handled separately. An individual match for either can lead to comma detection and alignment. TRUE: A comma match consists of a comma plus followed immediately by a comma minus. The match pattern is 20 or 16 bits (as determined by INTDATAWIDTH). When COMMA_DOUBLE is TRUE, PCOMMA_DETECT must be the same as MCOMMA_DETECT, and RXENPCOMMAALIGN must be the same as RXENMCOMMAALIGN.

Table 7-28: **RX Comma Alignment and Detection Attributes *(Cont'd)***

Attribute	Type	Description
MCOMMA_10B_VALUE_0 MCOMMA_10B_VALUE_1	10-bit Binary	Defines comma minus to raise RXCOMMADET and aligns the parallel data. Reception order is right to left. (MCOMMA_10B_VALUE[0] is received first.) The default value is `1010000011` (K28.5). This definition does not affect 8B/10B encoding or decoding.
MCOMMA_DETECT_0 MCOMMA_DETECT_1	Boolean	Controls the raising of RXCOMMADET on comma minus. FALSE: Do not raise RXCOMMADET when comma minus is detected. TRUE: Raise RXCOMMADET when comma minus is detected. (This setting does not affect comma alignment.)
PCOMMA_10B_VALUE_0 PCOMMA_10B_VALUE_1	10-bit Binary	Defines comma plus to raise RXCOMMADET and aligns the parallel data. Reception order is right to left. (PCOMMA_10B_VALUE[0] is received first.) The default value is `0101111100` (K28.5). This definition does not affect 8B/10B encoding or decoding.
PCOMMA_DETECT_0 PCOMMA_DETECT_1	Boolean	Controls the raising of RXCOMMADET on comma plus. FALSE: Do not raise RXCOMMADET when comma plus is detected. TRUE: Raise RXCOMMADET when comma plus is detected. (This setting does not affect comma alignment.)
RX_SLIDE_MODE_0 RX_SLIDE_MODE_1	String	Selects between sliding in the PMA or in the PCS. Permitted values are PCS (default) and PMA.

Description

Enabling Comma Alignment

To enable the comma alignment block, the RXCOMMADETUSE port is driven High. RXCOMMADETUSE is driven Low to bypass the block completely for minimum latency.

Configuring Comma Patterns

To set the comma pattern that the block searches for in the incoming data stream, the MCOMMA_10B_VALUE, PCOMMA_10B_VALUE, and COMMA_10B_ENABLE attributes are used. The comma lengths depend on the INTDATAWIDTH of the tile (see "Shared PMA PLL," page 84). Figure 7-15 shows how the COMMA_10B_ENABLE masks each of the comma values to allow partial pattern matching.

Figure 7-15 shows how a COMMA is combined with COMMA_ENABLE to make a wildcarded comma for a 20-bit internal comma.

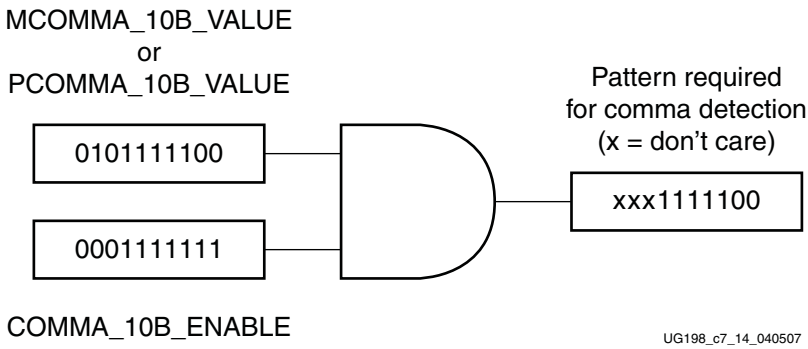


Figure 7-15: **Comma Pattern Masking**

If COMMA_DOUBLE is TRUE, the MCOMMA and PCOMMA patterns are combined so that the block searches for two commas in a row. The number of bits in the comma depends on INTDATAWIDTH (see "Shared PMA PLL," page 84). Figure 7-16 shows how the commas are combined when COMMA_DOUBLE is TRUE.

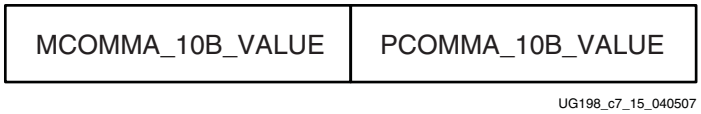


Figure 7-16: **Extended Comma Pattern Definition**

Figure 7-17 shows how COMMA_10B_ENABLE and wildcarding work for a double-width comma.

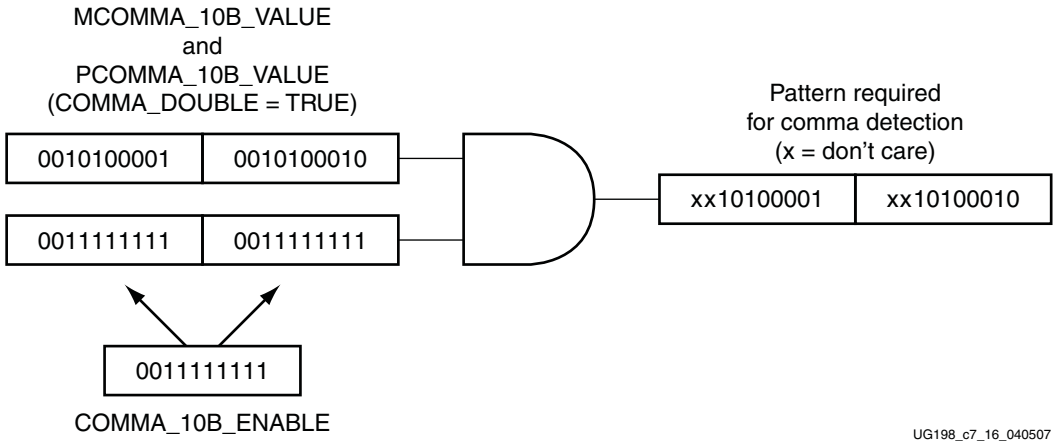


Figure 7-17: **Extended Comma Pattern Masking**

Activating Comma Alignment

Commas are aligned to the closest boundary providing they are found while comma alignment is active. RXENMCOMMAALIGN is driven High to align on the MCOMMA pattern. RXENPCOMMAALIGN is driven High to activate alignment on the PCOMMA pattern. Both enable ports are driven to align to either pattern. When COMMA_DOUBLE is TRUE, both enable ports must always be driven to the same value.

Alignment Status Signals

While MCOMMA or PCOMMA alignment is active, any matching comma pattern causes the block to realign to the closest boundary. After successful alignment, the block holds RXBYTEISALIGNED High. At this time, RXENMCOMMAALIGN and RXENPCOMMAALIGN can be driven Low to turn off alignment and keep the current alignment position. PCOMMA_ALIGN must be TRUE for PCOMMAs to cause RXBYTEISALIGNED to go High. Similarly, MCOMMA_ALIGN must be TRUE for MCOMMAs to cause RXBYTEISALIGNED to go High.

Commas can arrive while RXBYTEISALIGNED is High. If the commas arrive aligned to boundaries, there is no change. If the commas arrive out of position, the block deasserts RXBYTEISALIGNED until the commas are aligned again. If alignment is still activated for the comma that arrives, the block automatically aligns the new comma to the closest boundary and drives RXBYTEREALIGN High for one RXUSRCLK2 cycle.

Alignment Boundaries

The legal boundaries for alignment are defined by ALIGN_COMMA_WORD. The spacing of the legal boundaries is determined by INTDATAWIDTH, and the number of legal boundary positions is determined by the number of bytes in the RXDATA interface. Figure 7-18 shows the boundaries that can be selected.

RXDATAWIDTH	ALIGN_COMMA_WORD	Possible RX Alignments (Grey = Comma Can Appear on Byte)
0 (1-byte)	1 (Any Boundary)	RXDATA Byte 0
0 (1-byte)	2 (Even Boundary Only)	Invalid Configuration
1 (2-byte)	1 (Any Boundary)	RXDATA Byte 1 | RXDATA Byte 0
1 (2-byte)	2 (Even Boundaries Only)	RXDATA Byte 1 | RXDATA Byte 0
2 (4-byte)	1 (Any Boundariy)	RXDATA Byte 3 | RXDATA Byte 2 | RXDATA Byte 1 | RXDATA Byte 0
2 (4-byte)	2 (Even Boundaries Only)	RXDATA Byte 3 | RXDATA Byte 2 | RXDATA Byte 1 | RXDATA Byte 0

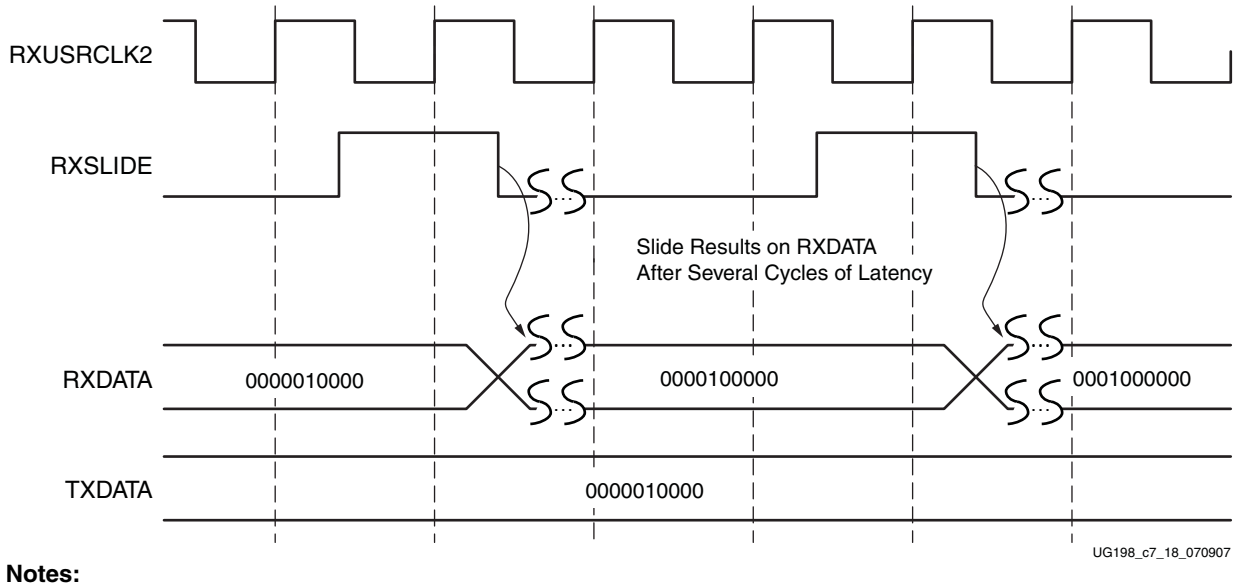
UG198_c7_17_070907

Figure 7-18: **Comma Alignment Boundaries**

Manual Alignment

RXSLIDE can be used to override the automatic comma alignment and to shift the parallel data. RXSLIDE is driven High for one RXUSRCLK2 cycle to shift the parallel data to the left by one bit. RXSLIDE must be Low for at least two RXUSRCLK2 cycles before it can be used again.

Figure 7-19 shows the waveforms for manual alignment using RXSLIDE, before and after the data shift.



Notes:

1. Latency between the slide and the slide result at RXDATA depends on the number of active RX PCS blocks in the datapath

Figure 7-19: **Manual Data Alignment Using RXSLIDE for RXDATAWIDTH = 1 (16-bit)**

Configurable Loss-of-Sync State Machine

Overview

Several 8B/10B protocols make use of a standard Loss-of-Sync (LOS) state machine to detect when the channel is malfunctioning. Each GTX receiver includes a LOS state machine that can be activated for protocols requiring it. When the state machine is not used, the LOS state machine's ports can be re-used to monitor the condition of incoming data.

Ports and Attributes

Table 7-29 defines the RX loss-of-sync state machine ports.

Table 7-29: **RX Loss-of-Sync State Machine Ports**

Port	Dir	Clock Domain	Description
RXLOSSOFSYNC0[1:0] RXLOSSOFSYNC1[1:0]	Out	RXUSRCLK2	FPGA status related to byte stream synchronization. The meaning depends on the state of the RX_LOSS_OF_SYNC_FSM attribute. If RX_LOSS_OF_SYNC_FSM = TRUE, this output reflects the state of an internal Loss-of-Sync FSM as follows: [1] = 1: Sync lost due to either sequence of invalid characters or reset [0] = 1: In the resync state due to a channel bonding sequence or realignment If RX_LOSS_OF_SYNC_FSM = FALSE, this output presents the following information about incoming data: [1] = 1: Received data is not an 8B/10B character or has a disparity error [0] = 1: Channel bonding sequence detected in data

Table 7-30 defines the RX loss-of-sync state machine attributes.

Table 7-30: **RX Loss-of-Sync State Machine Attributes**

Attribute	Type	Description
RX_LOS_INVALID_INCR_0 RX_LOS_INVALID_INCR_1	Integer	Defines the number of valid characters required to *cancel out* the appearance of one invalid character for the purpose of loss-of-sync determination. Valid settings are 1, 2, 4, 8, 16, 32, 64, and 128.
RX_LOS_THRESHOLD_0 RX_LOS_THRESHOLD_1	Integer	When divided by RX_LOS_INVALID_INCR_(0/1), defines the number of invalid characters required to cause an FSM transition to the *sync lost* state. Valid settings are 4, 8, 16, 32, 64, 128, 256, and 512.
RX_LOSS_OF_SYNC_FSM_0 RX_LOSS_OF_SYNC_FSM_1	Boolean	RX_LOSS_OF_SYNC_FSM defines the behavior of the RXLOSSOFSYNC[1:0] outputs. FALSE (default): RXLOSSOFSYNC[1] goes High when invalid data (not in table or disparity error) is found in 8B/10B decoding. RXLOSSOFSYNC[0] goes High when a channel bonding sequence has been written into the RX elastic buffer. TRUE: Loss of sync FSM is in operation and its state is reflected on RXLOSSOFSYNC[1].

Description

Figure 7-20 shows the standard LOS state machine, used in several 8B/10B protocols (for example, XAUI) to detect problems in the incoming data stream.

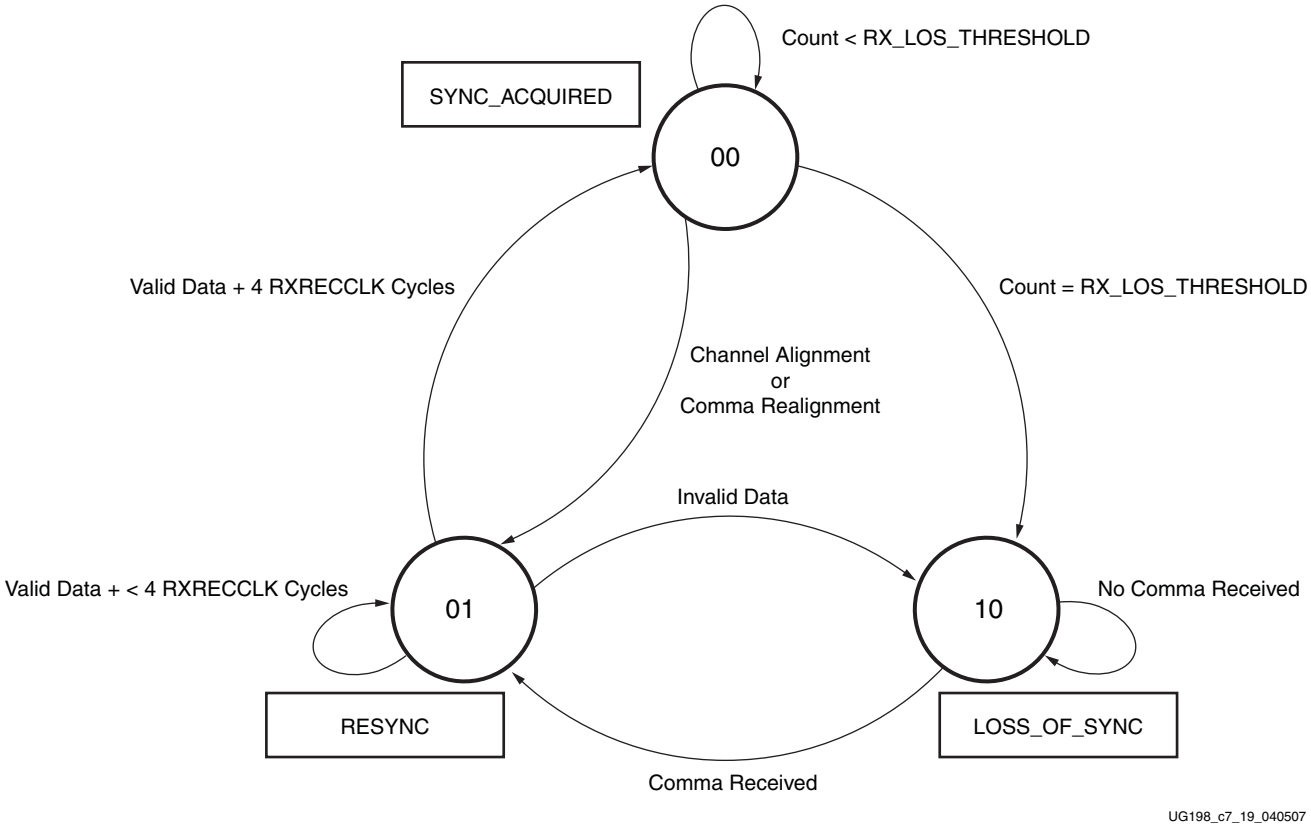


Figure 7-20: **LOS State Machine**

To activate the LOS state machine in the GTX transceiver, RX_LOSS_OF_SYNC_FSM is set to TRUE. While the state machine is active, the RXLOSSOFSYNC port presents its current state.

If the LOS state machine is inactive (RX_LOSS_OF_SYNC_FSM = FALSE), the RXLOSSOFSYNC port presents information about the received data. The RXLOSSOFSYNC entry in Table 7-29 shows the meaning of the RXLOSSOFSYNC port in this case.

The operation of the LOS state machine can be tuned using the RX_LOS_INVALID_INCR and RX_LOS_THRESHOLD attributes. RX_LOS_THRESHOLD adjusts how sensitive the LOS state machine is to bad characters by adjusting the number of characters required to force the machine from the SYNC_ACQUIRED state to the LOSS_OF_SYNC state. The RX_LOS_THRESHOLD entry in Table 7-30 shows the valid settings for this attribute.

The LOS state machine allows the error count in the SYNC_ACQUIRED state to decrease over time, so that sparse errors are eventually discarded. The rate that the error count is decreased is controlled by the RX_LOS_INVALID_INCR attributes, as defined in Table 7-30.

Configurable 8B/10B Decoder

Overview

Many protocols require receivers to decode 8B/10B data. 8B/10B is an industry standard encoding scheme that trades two bits of overhead per byte for improved performance. Table 6-3, page 125 outlines the benefits and costs of 8B/10B. Appendix C shows how 8B/10B maps 10-bit sequences to 8-bit data and control values.

The GTX transceiver includes an 8B/10B decoder to decode RX data without consuming FPGA resources. The decoder includes status signals to indicate errors and incoming control sequences. If decoding is not needed, the block can be disabled to minimize latency.

Ports and Attributes

Table 7-31 defines the RX decoder ports.

Table 7-31: **RX Decoder Ports**

Port	Dir	Clock Domain	Description
RXCHARISCOMMA0[3:0] RXCHARISCOMMA1[3:0]	Out	RXUSRCLK2	RXCHARISCOMMA is asserted when RXDATA is an 8B/10B comma. This signal, which depends on DEC_MCOMMA_DETECT and DEC_PCOMMA_DETECT, is always Low when RXDEC8B10BUSE is Low. RXCHARISCOMMA[3] corresponds to RXDATA[31:24] RXCHARISCOMMA[2] corresponds to RXDATA[23:16] RXCHARISCOMMA[1] corresponds to RXDATA[15:8] RXCHARISCOMMA[0] corresponds to RXDATA[7:0]
RXCHARISK0[3:0] RXCHARISK1[3:0]	Out	RXUSRCLK2	RXCHARISK is asserted when RXDATA is an 8B/10B K character. This signal is always Low when RXDEC8B10BUSE is Low. RXCHARISK[3] corresponds to RXDATA[31:24] RXCHARISK[2] corresponds to RXDATA[23:16] RXCHARISK[1] corresponds to RXDATA[15:8] RXCHARISK[0] corresponds to RXDATA[7:0]
RXDATAWIDTH0[1:0] RXDATAWIDTH1[1:0]	In	RXUSRCLK2	Selects the width of the RXDATA port. 0: RXDATA is 8 bits or 10 bits wide 1: RXDATA is 16 bits or 20 bits wide 2: RXDATA is 32 bits or 40 bits wide 3: Reserved
RXDEC8B10BUSE0 RXDEC8B10BUSE1	In	RXUSRCLK2	RXDEC8B10BUSE enables the 8B/10B decoder. 1: 8B/10B decoder enabled 0: 8B/10B decoder bypassed (reduces latency)
RXDISPERR0[3:0] RXDISPERR1[3:0]	Out	RXUSRCLK2	When High, RXDISPERR indicates that RXDATA was received with a disparity error. RXDISPERR[3] corresponds to RXDATA[31:24] RXDISPERR[2] corresponds to RXDATA[23:16] RXDISPERR[1] corresponds to RXDATA[15:8] RXDISPERR[0] corresponds to RXDATA[7:0]

Table 7-31: **RX Decoder Ports *(Cont'd)***

Port	Dir	Clock Domain	Description
RXNOTINTABLE0[3:0] RXNOTINTABLE1[3:0]	Out	RXUSRCLK2	RXNOTINTABLE indicates that RXDATA is the result of an erroneous 8B/10B code. RXNOTINTABLE[3] corresponds to RXDATA[31:24] RXNOTINTABLE[2] corresponds to RXDATA[23:16] RXNOTINTABLE[1] corresponds to RXDATA[15:8] RXNOTINTABLE[0] corresponds to RXDATA[7:0]
RXRUNDISP0[3:0] RXRUNDISP1[3:0]	Out	RXUSRCLK2	RXRUNDISP shows the running disparity of the 8B/10B encoder when RXDATA is received. RXRUNDISP[3] corresponds to RXDATA[31:24] RXRUNDISP[2] corresponds to RXDATA[23:16] RXRUNDISP[1] corresponds to RXDATA[15:8] RXRUNDISP[0] corresponds to RXDATA[7:0]

Table 7-32 defines the RX decoder attributes.

Table 7-32: **RX Decoder Attributes**

Attributes	Type	Description
DEC_MCOMMA_DETECT_0 DEC_MCOMMA_DETECT_1	Boolean	Enables detection of negative 8B/10B commas: TRUE: RXCHARISCOMMA is asserted when RXDATA is a negative 8B/10B comma FALSE: RXCHARISCOMMA does not respond to negative 8B/10B commas
DEC_PCOMMA_DETECT_0 DEC_PCOMMA_DETECT_1	Boolean	Enables detection of positive 8B/10B commas: TRUE: RXCHARISCOMMA is asserted when RXDATA is a positive 8B/10B comma FALSE: RXCHARISCOMMA does not respond to positive 8B/10B commas
DEC_VALID_COMMA_ONLY_0 DEC_VALID_COMMA_ONLY_1	Boolean	Limits the set of commas to which RXCHARISCOMMA responds. TRUE: RXCHARISCOMMA is asserted only for K28.1, K28.5, and K28.7 (see 8B/10B K character table in Appendix C) FALSE: RXCHARISCOMMA responds to any positive or negative 8B/10B comma, depending on the settings for DEC_MCOMMA_DETECT and DEC_PCOMMA_DETECT

Description

Enabling the 8B/10B Decoder

To disable the 8B/10B decoder, RXDEC8B10BUSE is driven Low. With the decoder off, the number of bits per byte on the RX interface is determined by the internal data width of the tile. See "FPGA RX Interface," page 231 for details about the deserialization order and parallel bitmapping when 8B/10B decoding is bypassed.

To enable the 8B/10B decoder, RXDEC8B10BUSE is driven High. INTDATAPATH must also be High (20-bit internal datapath) to enable the 8B/10B decoder.

8B/10B Decoder Bit and Byte Order

The order of the bits into the 8B/10B decoder is the opposite of the order shown in the 8B/10B table in Appendix C. 8B/10B requires bit a0 to be received first, but the GTX transceiver always receives the right-most bit first. Consequently, the 8B/10B decoder is designed to automatically reverse the bit order of received data before decoding it.

Similarly, because the GTX transceiver receives the right-most bit first, when a 2-byte interface is used, the first byte received (byte 0) is presented on RXDATA[7:0], and the second byte is presented on RXDATA[15:8]. When a 4-byte interface is used, the first received byte is presented on RXDATA[23:16], and the fourth byte is presented on RXDATA[31:24]. Figure 7-21 shows how the decoder maps 10-bit data to 8-bit values.

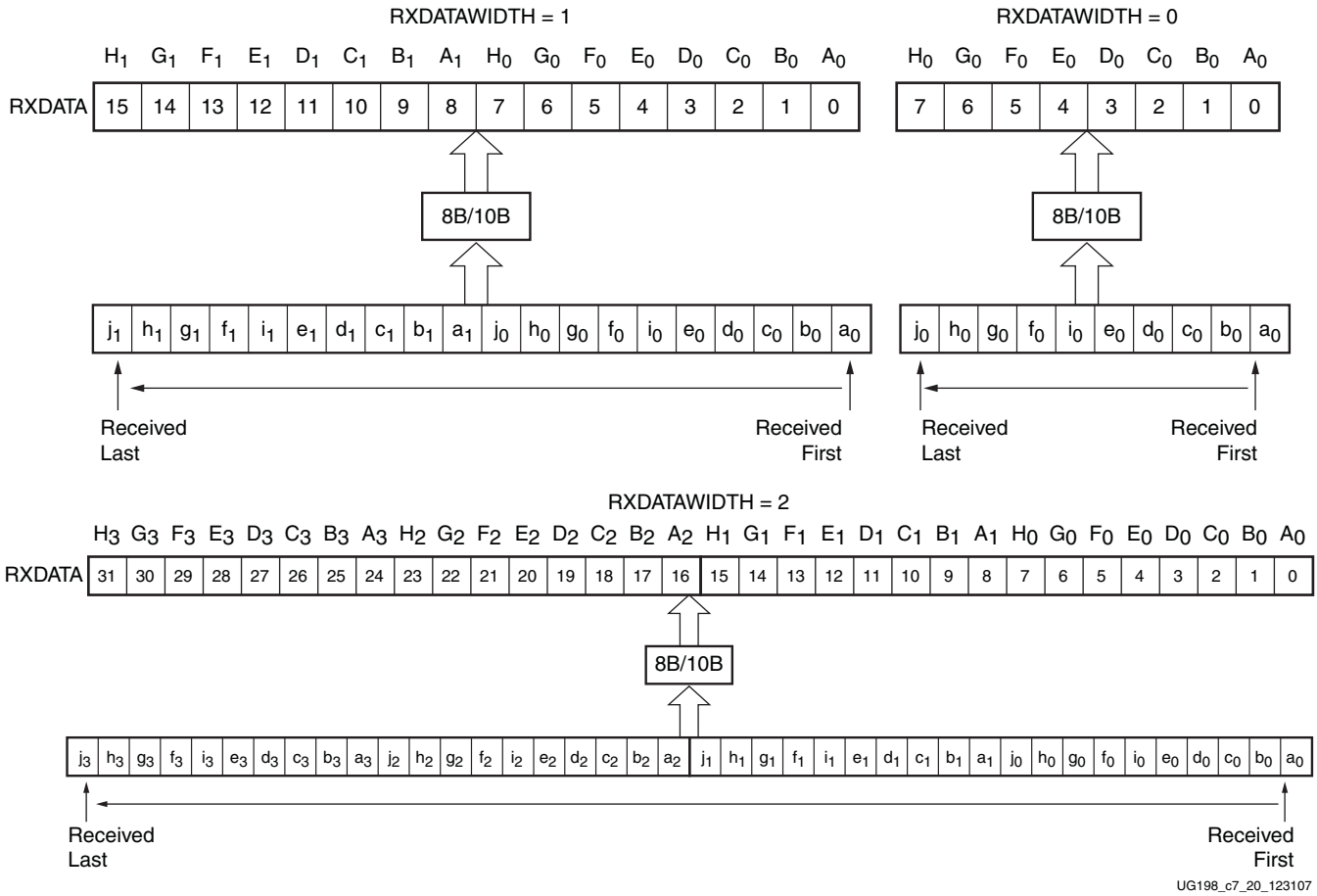


Figure 7-21: **RX Interface with 8B/10B Decoding**

K Characters and 8B/10B Commas

The 8B/10B table (shown in Appendix C) includes special characters (K characters) that are often used for control functions. When RXDATA is a K character, the decoder drives RXCHARISK High.

If DEC_PCOMMA_DETECT is TRUE, the decoder drives RXCHARISCOMMA High whenever RXDATA is a positive 8B/10B comma. Similarly, if DEC_MCOMMA_DETECT is TRUE, the decoder drives RXCHARISCOMMA High whenever RXDATA is a negative 8B/10B comma.

To limit the set of commas that trigger RXCHARISCOMMA to K28.1, K28.5, and K28.7, DEC_VALID_COMMA_ONLY is set to TRUE. This setting is typically used for Ethernet-based applications. RXCHARISCOMMA does not depend on MCOMMA_10B_VALUE or PCOMMA_10B_VALUE.

RX Running Disparity

The 8B/10B decoder uses a running disparity system to balance the number of 1s and 0s transmitted. The 8B/10B decoder tracks the running disparity of incoming data to detect errors. Monitor the RXRUNDISP port to see the current running disparity.

Disparity Errors and Not-in-Table Errors

The decoder drives RXDISPERR High when RXDATA arrives with the wrong disparity. In addition to disparity errors, the 8B/10B decoder detects 20-bit out-of-table error codes. The decoder drives the RXNOTINTABLE port High when RXDATA is not a valid 8B/10B character.

Figure 7-22 shows a waveform with a few error bytes arriving on RXDATA and the RXNOTINTABLE and RXDISPERR ports indicating the error.

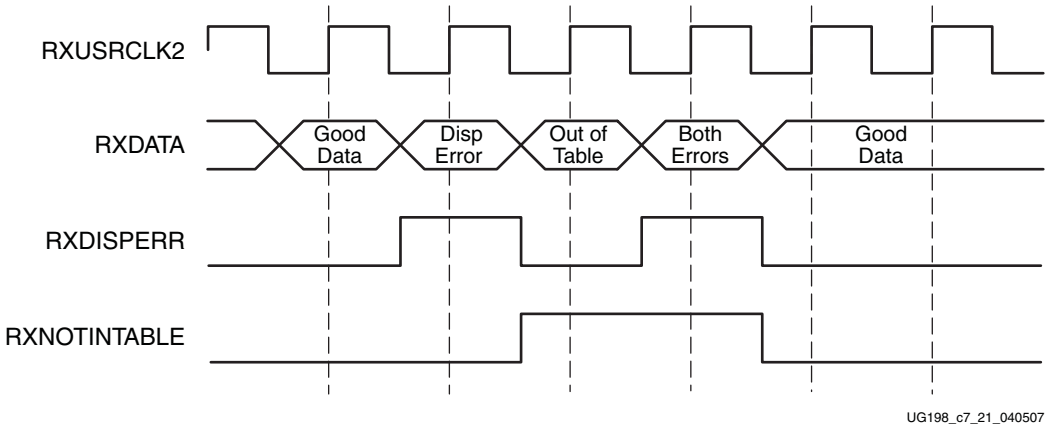


Figure 7-22: **RX Data with 8B/10B Errors**

Configurable RX Elastic Buffer and Phase Alignment

Overview

The GTX RX datapath has two internal parallel clock domains used in the PCS: the PMA parallel clock domain (XCLK) and the RXUSRCLK domain. To receive data, the PMA parallel rate must be sufficiently close to the RXUSRCLK rate, and all phase differences between the two domains must be resolved. Figure 7-23 shows the two parallel clock domains, XCLK and RXUSRCLK.

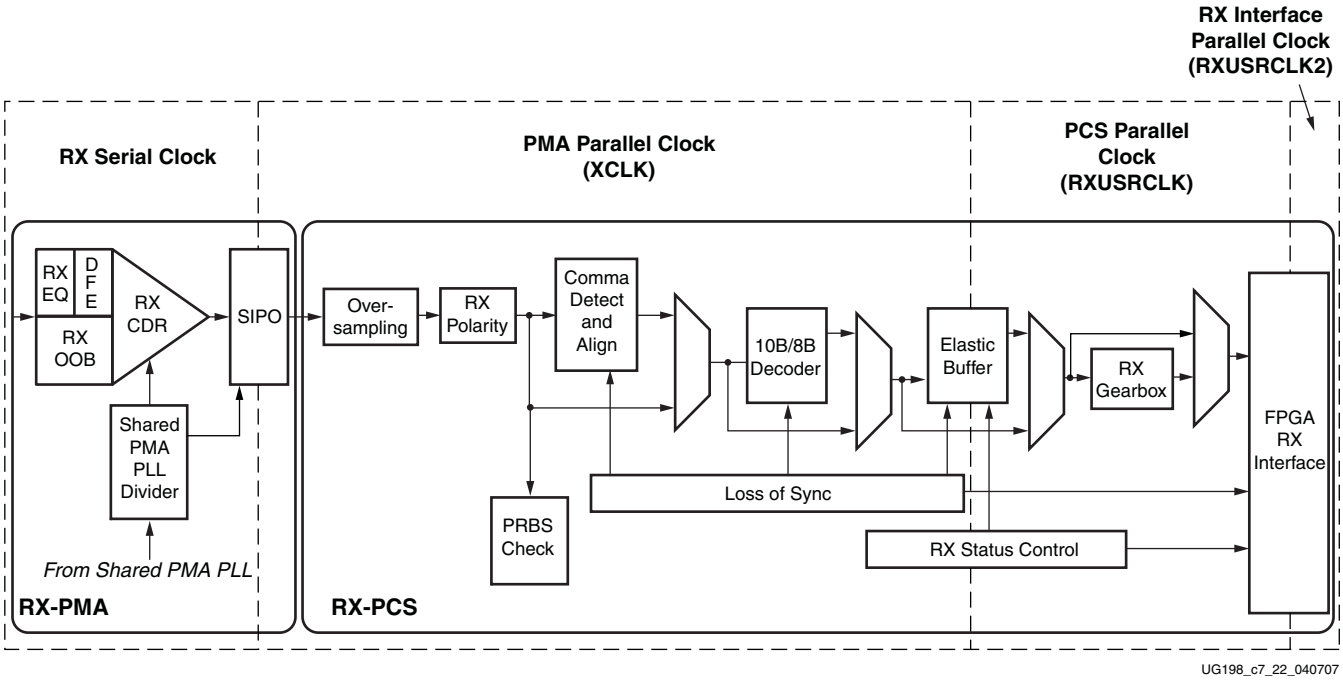


Figure 7-23: **Receiver Parallel Clock Domains**

The GTX transceiver includes an RX elastic buffer to resolve differences between the PMACLK and RXUSRCLK domain. The phase of the two domains can also be matched by using the recovered clock from the transceiver to drive RXUSRCLK and adjusting its phase to match XCLK. All RX datapaths must use one of these approaches. The costs and benefits of each approach are shown in Table 7-33.

Table 7-33: **Buffering vs. Phase Alignment**

	RX Elastic Buffer	RX Phase Alignment
Clocking Options	Can use recovered clock or local clock (with clock correction)	Must use recovered clock
Initialization	Works immediately	Must wait for all clocks to stabilize then perform alignment procedure
Latency	Buffer latency depends on features used (clock correction and channel bonding)	Lower latency than using the RX elastic buffer(1)

Table 7-33: **Buffering vs. Phase Alignment *(Cont'd)***

	RX Elastic Buffer	RX Phase Alignment
Clock Correction/ Channel Bonding	Required for clock correction/channel bonding	
Internal Data Width	Can be 16 or 20 bits wide	Can be 16 or 20 bits wide(2)

Notes:

1. Bypassing the RX buffer is an advanced feature and is not recommended for normal operation. RX buffer bypass operation can be guaranteed only under certain system-level conditions and data rates.
2. There is an important difference between the GTP_DUAL and GTX_DUAL tiles. The GTP_DUAL internal data width must be 10 bits wide when using RX phase alignment. The GTX_DUAL internal data width can be either 16 or 20 bits wide.

The RX elastic buffer is also used for clock correction (see "Configurable Clock Correction," page 210) and channel bonding (see "Configurable Channel Bonding (Lane Deskew)," page 216). Clock correction is used in cases where PMACLK and RXUSRCLK are not frequency matched. Table 7-34 lists common clock configurations and shows whether they require clock correction.

Table 7-34: **Common Clock Configurations**

	Needs Clock Correction?
Synchronous System (both sides use same physical oscillator for REFCLK)	No
Separate Reference Clocks, RX uses RXRECCLK	No
Separate Reference Clocks, RX uses Local Clock	Yes

Ports and Attributes

Table 7-35 defines the RX elastic buffer and phase-alignment ports.

Table 7-35: **RX Elastic Buffer and Phase-Alignment Ports**

Port	Dir	Clock Domain	Description
INTDATAWIDTH	In	Async	Specifies the width of the internal datapath for the entire GTX_DUAL tile. 0: Internal datapath is 16 bits wide 1: Internal datapath is 20 bits wide
RXBUFRESET0 RXBUFRESET1	In	Async	Resets the RX elastic buffer logic and re-initializes the RX elastic buffer.
RXBUFSTATUS0[2:0] RXBUFSTATUS1[2:0]	Out	RXUSRCLK2	Indicates the status of the RX elastic buffer as follows: 000: Nominal condition 001: Number of bytes in the buffer are less than CLK_COR_MIN_LAT 010: Number of bytes in the buffer are greater than CLK_COR_MAX_LAT 101: RX elastic buffer underflow(1) 110: RX elastic buffer overflow(1)

Table 7-35: **RX Elastic Buffer and Phase-Alignment Ports *(Cont'd)***

Port	Dir	Clock Domain	Description
RXENPMAPHASEALIGN0 RXENPMAPHASEALIGN1	In	RXUSRCLK2	Enables the alignment of the XCLK with the RXURSCLK (independently) for each transceiver in the GTX_DUAL tile. Set High when bypassing the RX elastic buffer.(2)
RXPMASETPHASE0 RXPMASETPHASE1	In	RXUSRCLK2	Used to align the XCLK and RXUSRCLK domains when RXUSRCLK is driven by RXRECCLK. Allows the RX elastic buffer to be bypassed.(2)

Notes:

1. If an RX elastic buffer overflow or an RX elastic buffer underflow condition occurs, the content of the RX elastic buffer becomes invalid, and the RX elastic buffer needs re-initialization by asserting RXBUFRESET.
2. Bypassing the RX buffer is an advanced feature and is not recommended for normal operation. RX buffer bypass operation can be guaranteed only under certain system-level conditions and data rates.

Table 7-36 defines the RX elastic buffer and phase-alignment attributes.

Table 7-36: **RX Elastic Buffer and Phase-Alignment Attributes**

Attribute	Type	Description
OVERSAMPLE_MODE	Boolean	Enables built-in 5x digital oversampling. Applies to both transceivers in the GTX_DUAL tile. TRUE: Built-in 5x oversampling enabled FALSE: Built-in 5x oversampling disabled TX_BUFFER_USE must be true when OVERSAMPLE_MODE is TRUE. See "Oversampling," page 183 for the remaining configuration steps required to use oversampling.
RX_BUFFER_USE_0 RX_BUFFER_USE_1	Boolean	Use or bypass the RX elastic buffer. TRUE: Use the RX elastic buffer (normal mode). FALSE: Permanently bypass the RX elastic buffer. If OVERSAMPLE_MODE is FALSE, RX phase alignment must be used whenever the RX elastic buffer is bypassed.(1)
RX_EN_IDLE_RESET_BUF_0 RX_EN_IDLE_RESET_BUF_1	Boolean	When TRUE, resets the RX elastic buffer when no valid signal is on the RX inputs.
RX_XCLK_SEL_0 RX_XCLK_SEL_1	String	Selects the clock used to drive the RX parallel clock domain (XCLK). "RXREC": (default) XCLK domain driven by recovered clock from CDR. When OVERSAMPLE_MODE is TRUE, the recovered clock is sourced from the oversampling block. "RXUSR": RXUSRCLK port drives RX parallel clock domain. Use this mode when bypassing the RX elastic buffer.(1)

Notes:

1. Bypassing the RX buffer is an advanced feature and is not recommended for normal operation. RX buffer bypass operation can be guaranteed only under certain system-level conditions and data rates.

Description

Using the RX Elastic Buffer

Figure 7-24 shows how the RX elastic buffer bridges the PMA parallel clock domain (XCLK) and the PCS parallel clock domain (RXUSRCLK) in the RX datapath. This bridging is necessary because there is no guaranteed frequency or phase relationship between the parallel clock from the SIPO (XCLK) and the parallel clocks from the FPGA logic (RXUSRCLK and RXUSRCLK2).

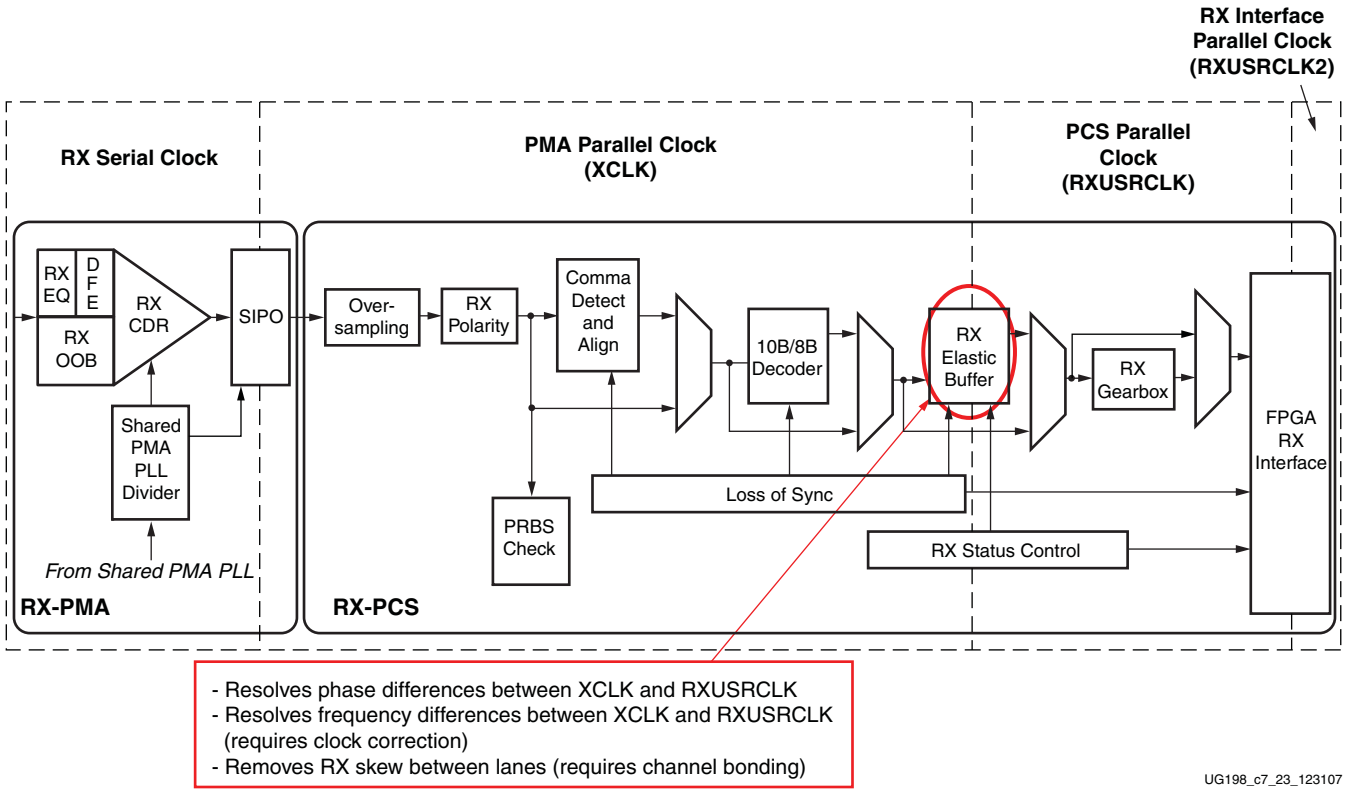


Figure 7-24: **Using the RX Elastic Buffer**

The RX elastic buffer can also be used when OVERSAMPLE_MODE is TRUE.

To use the RX elastic buffer to resolve phase differences between the domains:

- Set RX_BUFFER_USE to TRUE.
- Reset the buffer whenever RXBUFSTATUS indicates an overflow or an underflow.
- The buffer can be reset using GTXRESET (see "Reset," page 98), RXRESET, or RXBUFRESET.

Using RX Phase Alignment

The RX elastic buffer can be bypassed to reduce latency when RXRECCLK is used to source RXUSRCLK and RXUSRCLK2. When the RX elastic buffer is bypassed, latency through the RX datapath is low and deterministic, but clock correction and channel bonding are not available.

Note: Bypassing the RX buffer is an advanced feature and is not recommended for normal operation. RX buffer bypass operation can be guaranteed only under certain system-level conditions and data rates.

Figure 7-25 shows how phase alignment allows the RX elastic buffer to be bypassed. Before phase alignment, there is no guaranteed phase relationship between the parallel clock generated from the recovered clock in the CDR circuit (XCLK) and the parallel clocks from the FPGA logic (RXUSRCLK and RXUSRCLK2). Phase alignment causes RXRECCLK from the CDR to be adjusted so that there is no significant phase difference between XCLK and RXUSRCLK.

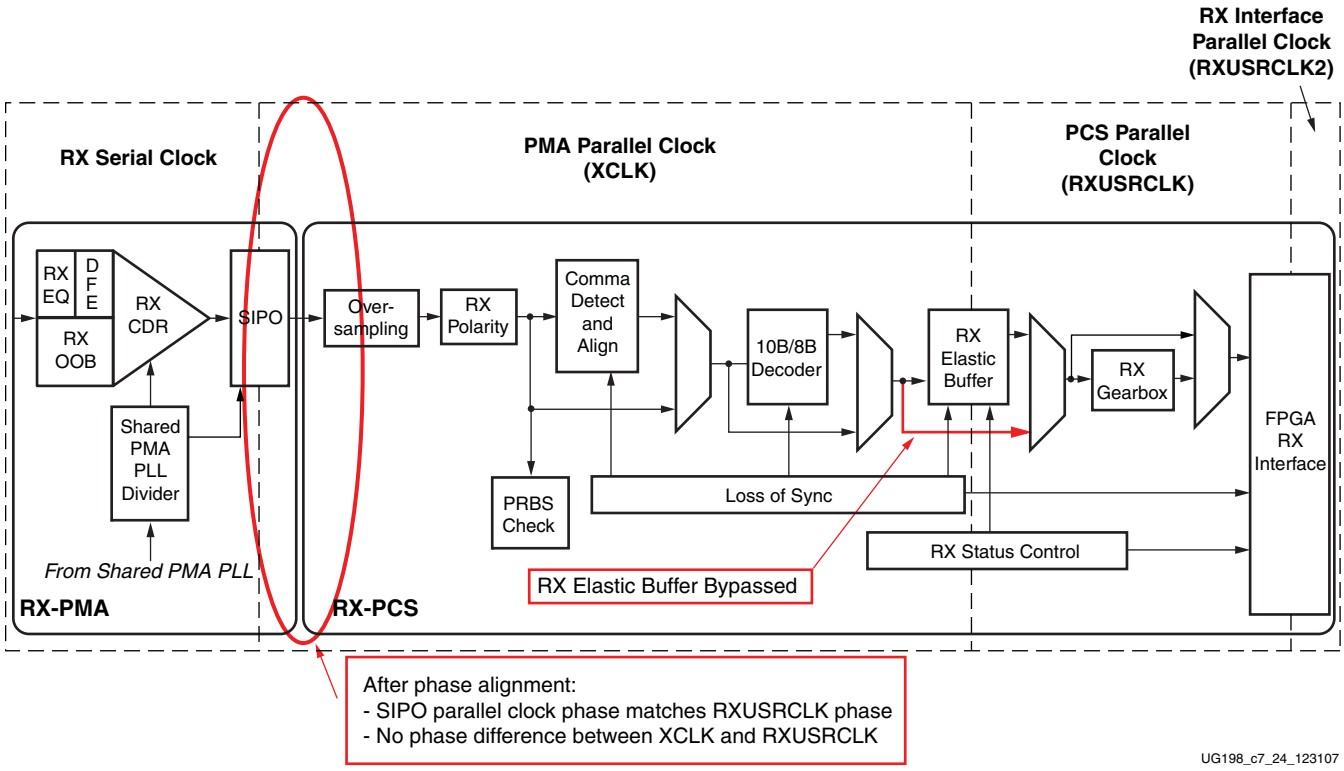


Figure 7-25: **Using Phase Alignment**

To use RX phase alignment:

1. Set RX_BUFFER_USE to FALSE to bypass the RX elastic buffer (optional).

   Note: Bypassing the RX buffer is an advanced feature and is not recommended for normal operation. RX buffer bypass operation can be guaranteed only under certain system-level conditions and data rates.
2. Set RX_XCLK_SEL to *RXUSR*.
3. Source RXUSRCLK and RXUSRCLK2 with the RXRECCLK output. Divide RXRECCLK by 2 if necessary to provide RXUSRCLK2 (see "FPGA RX Interface," page 231 for details).
4. Reset the RX datapath using GTXRESET or one of the CDR resets.
5. Wait for the shared PMA PLL and any DCM or PLL used for RXUSRCLK2 to lock.
6. Wait for the CDR to lock and provide a stable RXRECCLK.
7. Drive RXPMASETPHASE High for 32 RXUSRCLK2 cycles and then deassert it.

Step 6 requires careful guidance. Normally, CDR lock is detected by measuring the quality of incoming data. Methods for detecting CDR lock include:

- Finding known data in the incoming data stream (for example, commas or A1/A2 framing characters). In general, several consecutive known data patterns must be received without error to indicate a CDR lock.
- Using the LOS state machine (see "Configurable Loss-of-Sync State Machine," page 195). If incoming data is 8B/10B encoded and the CDR is locked, the LOS state machine moves to the SYNC_ACQUIRED state and stays there.

When the RX elastic buffer is bypassed, data received from the PMA might be distorted due to phase differences as it passes to the PCS. This makes it difficult to determine whether or not bad data is received because the CDR is not locked, or the CDR *is* locked and the phase alignment has not yet been attempted. To work around this problem, RX phase alignment must be attempted several times and the output data evaluated after each attempt. Good data is received if the phase is aligned while the RX CDR is locked.

The flow diagram in Figure 7-26 shows the series of steps required for successful RX phase alignment. Any number of clock cycles can be used for the CDR lock time, but using a larger number decreases the number of cycles through the states.

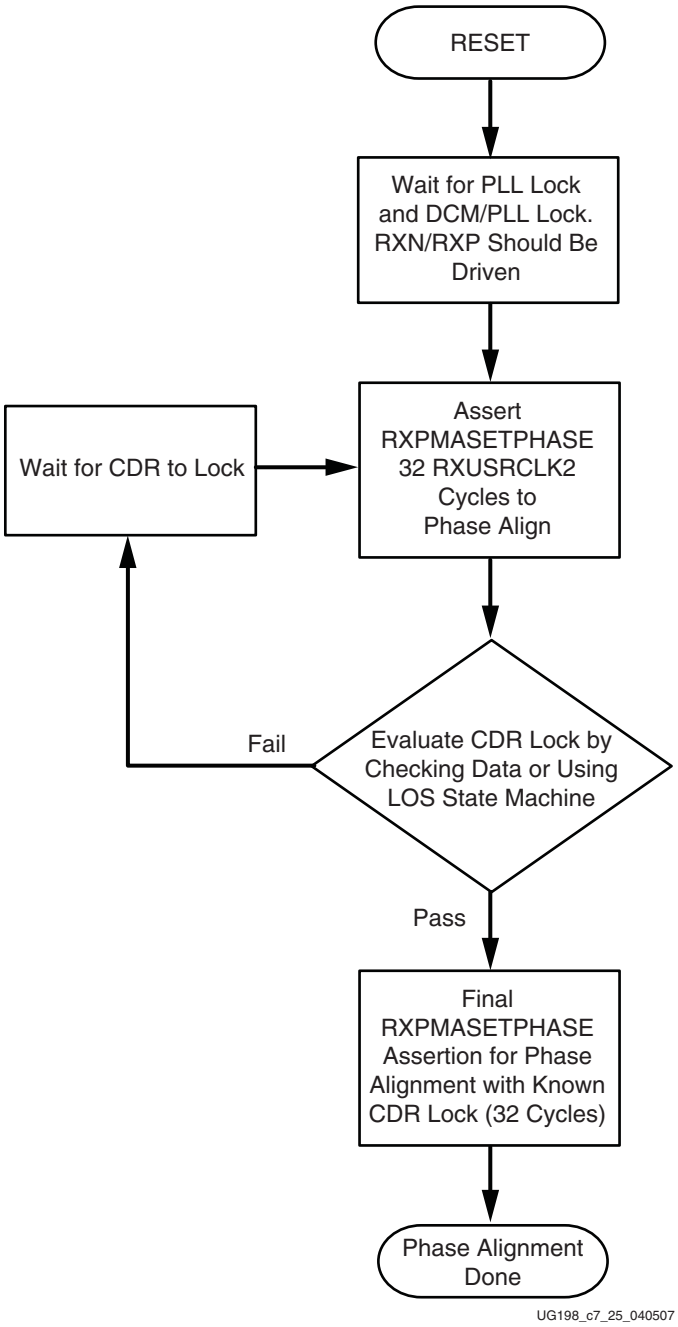


Figure 7-26: **Steps Required for Successful RX Phase Alignment**

Bypassing the RX Elastic Buffer while Using Built-in Oversampling

If OVERSAMPLE_MODE is TRUE to activate built-in oversampling, the RX elastic buffer is bypassed without the use of phase alignment. Instead, a shallow buffer inside the oversampling block is used to resolve phase differences between RXUSRCLK and RXRECCLK. See "Oversampling," page 183 for information about monitoring the status of the Oversampling block.

Note: Bypassing the RX buffer is an advanced feature and is not recommended for normal operation. RX buffer bypass operation can be guaranteed only under certain system-level conditions and data rates.

Figure 7-27 shows how the RX elastic buffer is bypassed with oversampling enabled. The shallow buffer in the oversampling block resolves any phase differences between RXUSRCLK and the recovered clock it generates.

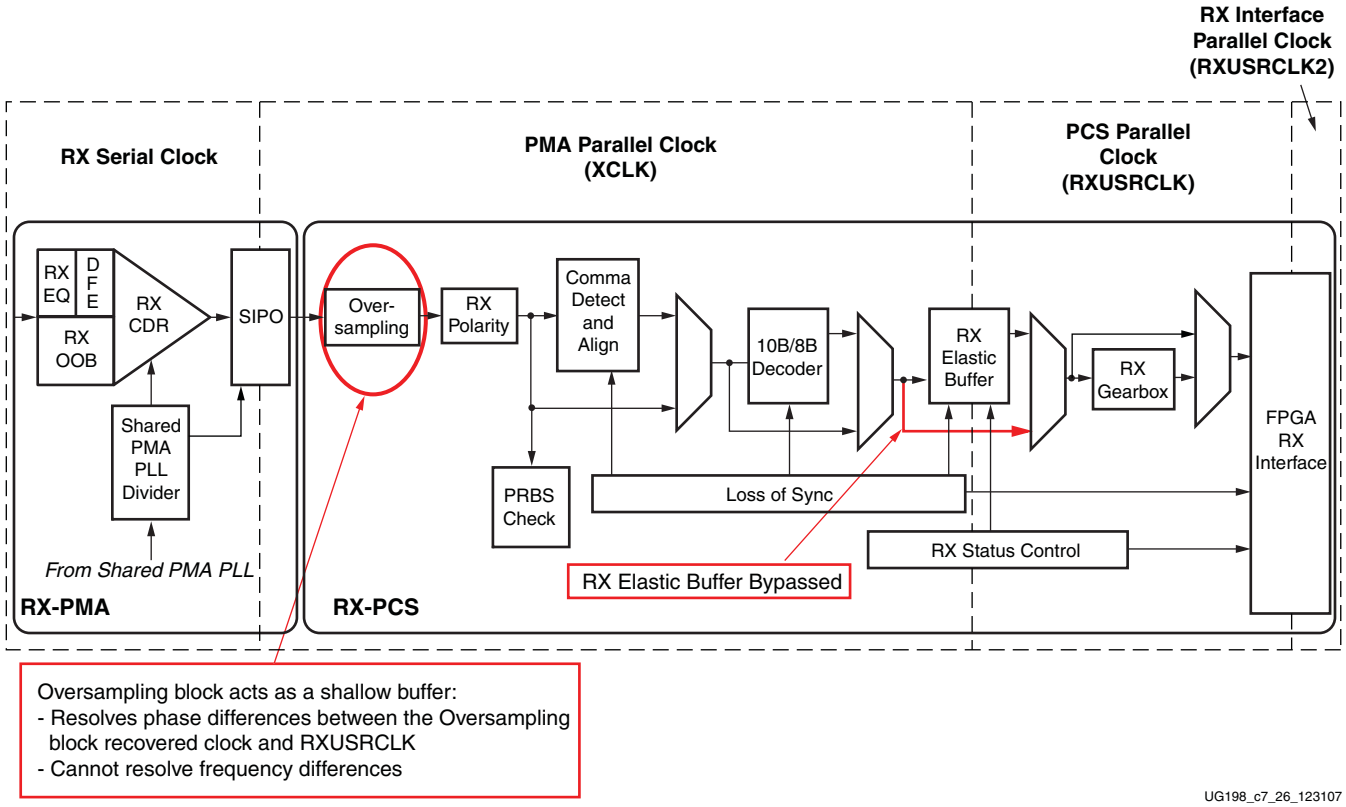


Figure 7-27: **Buffer Bypass with Oversampling Enabled**

To bypass the RX elastic buffer when OVERSAMPLING_MODE is TRUE:

1. Set RX_BUFFER_USE to FALSE to bypass the RX elastic buffer (optional).

   Note: Bypassing the RX buffer is an advanced feature and is not recommended for normal operation. RX buffer bypass operation can be guaranteed only under certain system-level conditions and data rates.

2. Set RX_XCLK_SEL to *RXUSR*.
3. Source RXUSRCLK and RXUSRCLK2 with the RXRECCLK output. Divide RXRECCLK by 2 if necessary to provide RXUSRCLK2 (see "FPGA RX Interface," page 231 for details).

Configurable Clock Correction

Overview

The RX elastic buffer has an additional benefit: it can tolerate frequency differences between the XCLK and RXUSRCLK domains by performing clock correction. Clock correction actively prevents the RX elastic buffer from getting too full or too empty by deleting or replicating special idle characters in the data stream.

Figure 7-28 shows a conceptual view of clock correction.

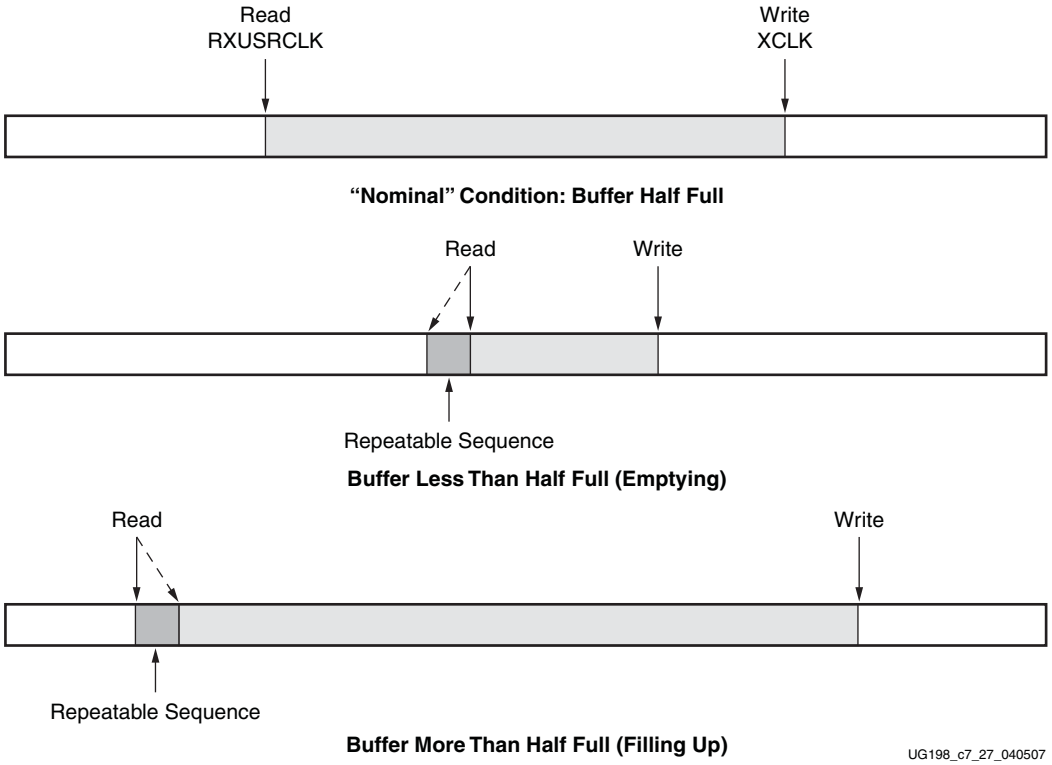


Figure 7-28: **Clock Correction**

Clock correction should be used whenever there is a frequency difference between XCLK and RXUSRCLK. It can be avoided by using the same frequency source for TX and RX, or by using the recovered clock to drive RXUSRCLK. The "Configurable RX Elastic Buffer and Phase Alignment" section has more details about the steps required if clock correction is not used.

Ports and Attributes

Table 7-37 defines the clock correction ports.

Table 7-37: **Clock Correction Ports**

Port	Dir	Clock Domain	Description
INTDATAWIDTH	In	Async	Specifies the bit width for the TX and RX internal datapaths. This port controls both transceivers on the tile: 0: 16-bit width 1: 20-bit width
RXBUFRESET0 RXBUFRESET1	In	Async	Resets the RX elastic buffer logic and re-initializes the RX elastic buffer.
RXBUFSTATUS0[2:0] RXBUFSTATUS1[2:0]	Out	RXUSRCLK2	Indicates the status of the RX elastic buffer as follows: 000: Nominal condition 001: Number of bytes in buffer is less than CLK_COR_MIN_LAT 010: Number of bytes in buffer is greater than CLK_COR_MAX_LAT 101: RX elastic buffer underflow(1) 110: RX elastic buffer overflow(1)
RXCLKCORCNT0[2:0] RXCLKCORCNT1[2:0]	Out	RXUSRCLK2	Reports the clock correction status of the RX elastic buffer: 000: No clock correction 001: 1 sequence skipped 010: 2 sequences skipped 011: 3 sequences skipped 100: 4 sequences skipped 101: Reserved 110: 2 sequences added 111: 1 sequence added

Notes:

1. If an RX elastic buffer overflow or an RX elastic buffer underflow condition occurs, the content of the RX elastic buffer becomes invalid, and the RX elastic buffer needs re-initialization by asserting RXBUFRESET.

Table 7-38 defines the clock correction attributes.

Table 7-38: **Clock Correction Attributes**

Attribute	Type	Description
CLK_CORRECT_USE_0 CLK_CORRECT_USE_1	Boolean	Enables clock correction. FALSE: Clock correction disabled TRUE: Clock correction enabled
CLK_COR_ADJ_LEN_0 CLK_COR_ADJ_LEN_1	Integer	This attribute defines the size of the adjustment (number of bytes repeated or skipped) in a clock correction. The bytes skipped or repeated always start from the beginning of the clock correction sequence to allow more bytes to be replaced than in the specified clock correction sequence. Valid lengths are from one to four bytes.

Table 7-38: **Clock Correction Attributes *(Cont'd)***

Attribute	Type	Description
CLK_COR_DET_LEN_0 CLK_COR_DET_LEN_1	Integer	This attribute defines the length of the sequence that the transceiver matches to detect opportunities for clock correction. Valid lengths are from one to four bytes.
CLK_COR_INSERT_IDLE_FLAG_0 CLK_COR_INSERT_IDLE_FLAG_1	Boolean	Controls whether the RXRUNDISP input status indicates running disparity or inserted-idle (clock correction sequence) flag. FALSE: RXRUNDISP indicates running disparity when RXDATA is decoded data. TRUE: RXRUNDISP is raised for the first byte of each inserted (repeated) clock correction ("Idle") sequence (when RXDATA is decoded data).
CLK_COR_KEEP_IDLE_0 CLK_COR_KEEP_IDLE_1	Boolean	Controls whether the RX elastic buffer must retain at least one clock correction sequence in the byte stream. FALSE: The transceiver can remove all clock correction sequences to further re-center the RX elastic buffer during clock correction. TRUE: In the final RXDATA stream, the transceiver must leave at least one clock correction sequence per continuous stream of clock correction sequences.
CLK_COR_MAX_LAT_0 CLK_COR_MAX_LAT_1	Integer	Specifies the maximum RX elastic buffer latency. If the RX elastic buffer exceeds CLK_COR_MAX_LAT, the clock correction circuit replicates incoming clock correction sequences to prevent overflow. Valid values for this attribute range from 3 to 48.
CLK_COR_MIN_LAT_0 CLK_COR_MIN_LAT_1	Integer	Specifies the minimum RX elastic buffer latency. If the RX elastic buffer drops below CLK_COR_MIN_LAT, the clock correction circuit replicates incoming clock correction sequences to prevent underflow. When the RX elastic buffer is reset, its pointers are set so there are CLK_COR_MIN_LAT unread (and uninitialized) data bytes in the buffer. Valid values for this attribute range from 3 to 48.
CLK_COR_PRECEDENCE_0 CLK_COR_PRECEDENCE_1	Boolean	Determines whether clock correction or channel bonding takes precedence when both operations are triggered at the same time. TRUE: Clock correction takes precedence over channel bonding if there is opportunity for both FALSE: Channel bonding takes precedence over clock correction if there is opportunity for both
CLK_COR_REPEAT_WAIT_0 CLK_COR_REPEAT_WAIT_1	Integer	This attribute specifies the minimum number of RXUSRCLK cycles without clock correction that must occur between successive clock corrections. If this attribute is zero, no limit is placed on how frequently clock correction can occur. Valid values for this attribute range from 0 to 31.

Table 7-38: **Clock Correction Attributes *(Cont'd)***

Attribute	Type	Description
CLK_COR_SEQ_1_1_0 CLK_COR_SEQ_1_1_1	10-bit Binary	The CLK_COR_SEQ_1 attributes are used in conjunction with CLK_COR_SEQ_1_ENABLE to define clock correction sequence 1. The sequence is made up of four subsequences. Each subsequence is 10 bits long. The rules for setting the subsequences depend on INTDATAWIDTH and RX_DECODE_SEQ_MATCH. See the "Description" section to learn how to set clock correction subsequences. Not all subsequences need to be used. CLK_COR_DET_LEN determines how many of the sequence are used for a match. If CLK_COR_DET_LEN = 1, only CLK_COR_SEQ_1_1 is used. CLK_COR_SEQ_1_ENABLE can be used to make parts of the sequence don't cares. If CLK_COR_SEQ_1_ENABLE[k] is 0, CLK_COR_SEQ_1_k is a don't care subsequence and is always considered to be a match.
CLK_COR_SEQ_1_2_0 CLK_COR_SEQ_1_2_1		
CLK_COR_SEQ_1_3_0 CLK_COR_SEQ_1_3_1		
CLK_COR_SEQ_1_4_0 CLK_COR_SEQ_1_4_1		
CLK_COR_SEQ_1_ENABLE_0 CLK_COR_SEQ_1_ENABLE_1	4-bit Binary	
CLK_COR_SEQ_2_1_0 CLK_COR_SEQ_2_1_1	10-bit Binary	The CLK_COR_SEQ_2 attributes are used in conjunction with CLK_COR_SEQ_2_ENABLE to define the second clock correction sequence. This second sequence is used as an alternate sequence for clock correction when CLK_COR_SEQ_2_USE is TRUE: if either sequence 1 or sequence 2 arrives, clock correction is performed. The sequence is made up of four subsequences. Each subsequence is 10 bits long. The rules for setting the subsequences depend on INTDATAWIDTH and RX_DECODE_SEQ_MATCH. See the "Description" section to learn how to set clock correction subsequences. Not all subsequences need to be used. CLK_COR_DET_LEN determines how much of the sequence is used for a match. If CLK_COR_DET_LEN = 1, only CLK_COR_SEQ_2_1 is used. CLK_COR_SEQ_2_ENABLE can be used to make parts of the sequence don't care. If CLK_COR_SEQ_2_ENABLE[k] is 0, CLK_COR_SEQ_2_k is a don't care byte subsequence and is always considered to be a match.
CLK_COR_SEQ_2_2_0 CLK_COR_SEQ_2_2_1		
CLK_COR_SEQ_2_3_0 CLK_COR_SEQ_2_3_1		
CLK_COR_SEQ_2_4_0 CLK_COR_SEQ_2_4_1		
CLK_COR_SEQ_2_ENABLE_0 CLK_COR_SEQ_2_ENABLE_1	4-bit Binary	
CLK_COR_SEQ_2_USE_0 CLK_COR_SEQ_2_USE_1	Boolean	Determines if the second clock correction sequence is to be used. When set to TRUE, the second clock correction sequence also triggers clock correction.
RX_DECODE_SEQ_MATCH_0 RX_DECODE_SEQ_MATCH_1	Boolean	Determines whether sequences are matched against the input to the 8B/10B decoder or the output. Used for the clock correction circuit and the channel bonding circuit. TRUE: Sequences are matched against the output of the 8B/10B decoder. K characters and disparity information is used. Bit ordering of the 8B/10B output is used. FALSE: Sequences are matched against unencoded data. Bit ordering is as for an unencoded parallel interface.

Description

Enabling Clock Correction

Each GTX transceiver includes a clock correction circuit that performs clock correction by controlling the pointers of the RX elastic buffer. To use clock correction, RX_BUFFER_USE is set to TRUE to turn on the RX elastic buffer, and CLK_CORRECT_USE is set to TRUE to turn on the clock correction circuit.

Clock correction is triggered when the RX elastic buffer latency is too high or too low, and the clock correction circuit detects a match sequence. To use clock correction, the clock correction circuit must be configured to set the following items:

- RX elastic buffer limits
- Clock correction sequence

Setting RX Elastic Buffer Limits

The RX elastic buffer limits are set using CLK_COR_MIN_LAT (minimum latency) and CLK_COR_MAX_LAT (maximum latency). When the number of bytes in the RX elastic buffer drops below CLK_COR_MIN_LAT, the clock correction circuit writes an additional CLK_COR_ADJ_LEN bytes from the first clock correction sequence it matches to prevent the buffer from underflowing. Similarly, when the number of bytes in the RX elastic buffer exceeds CLK_COR_MAX_LAT, the clock correction circuit deletes CLK_COR_ADJ_LEN bytes from the first clock correction sequence it matches, starting with the first byte of the sequence.

Setting Clock Correction Sequences

The clock correction sequences are programmed using the CLK_COR_SEQ_1_* attributes and CLK_COR_ADJ_LEN. Each CLK_COR_SEQ_1_* attribute corresponds to one subsequence in clock correction sequence 1. CLK_COR_ADJ_LEN is used to set the number of subsequences to be matched. If INTDATAWIDTH is High (20-bit internal datapath), the clock correction circuit matches all 10 bits of each subsequence. If INTDATAWIDTH is Low (16-bit internal datapath), only the right-most eight bits of each subsequence are used.

A second, alternate clock correction sequence can be activated by setting CLK_COR_SEQ_2_USE to TRUE. The first and second sequences share length settings, but use different subsequence values for matching. Set the CLK_COR_SEQ_2_* attributes to define the subsequence values for the second sequence.

When using 8B/10B decoding (RXDEC8B10BUSE and INTDATAWIDTH are High), RX_DECODE_SEQ_MATCH is set to TRUE to search the output of the 8B/10B decoder for sequence matches instead of non-decoded data. This allows the circuit to look for 8-bit values with either positive or negative disparity, and to distinguish K characters from regular characters (see "Configurable 8B/10B Encoder," page 125 and "Configurable 8B/10B Decoder," page 198 for details). Figure 7-29 shows how to set a clock correction sequence byte when RX_DECODE_SEQ_MATCH is TRUE.

When RX_DECODE_SEQ_MATCH is FALSE, the sequence must exactly match non-decoded incoming data. The bit order of the data matches the bit order shown in Figure 7-42 and Figure 7-43 for RXDATA with no 8B/10B decoding.

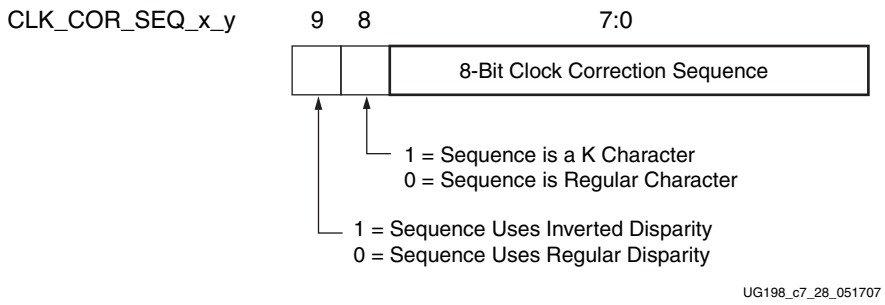


Figure 7-29: **Clock Correction Subsequence Settings with RX_DECODE_SEQ_MATCH = TRUE**

Some protocols use clock correction sequences with *don't care* subsequences. The clock correction circuit can be programmed to recognize these sequences using CLK_COR_SEQ_1_ENABLE and CLK_COR_SEQ_2_ENABLE. When the enable bit for a sequence is Low, that byte is considered matched no matter what the value. Figure 7-30 shows the mapping between the clock correction sequences and the clock correction sequence enable bits.

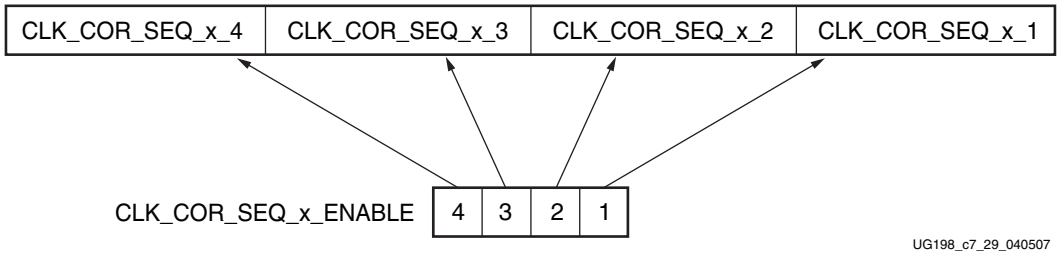


Figure 7-30: **Clock Correction Sequence Mapping**

Clock Correction Options

CLK_COR_REPEAT_WAIT is used to control the clock correction frequency. This value is set to the minimum number of RXUSRCLK cycles required between clock correction events. This attribute is set to 0 to allow clock correction to occur any time.

Some protocols allow clock correction to occur at any time, but require that if the clock correction circuit removes sequences, at least one sequence stays in the stream. For protocols with this requirement, CLK_COR_KEEP_IDLE is set to TRUE.

Monitoring Clock Correction

The clock correction circuit can be monitored using the RXCLKCORCNT and RXBUFSTATUS ports. The RXCLKCORCNT entry in Table 7-37 shows how to decode the values of RXCLKCORCNT to determine the status of the clock correction circuit. The RXBUFSTATUS entry in Table 7-37 shows how to decode the values of RXBUFSTATUS to determine how full the RX elastic buffer is.

In addition to RXCLKCORCNT and RXBUFSTATUS, RXRUNDISP can be taken from the 8B/10B decoder interface (see "Configurable 8B/10B Decoder," page 198) and used to indicate when RXDATA has the first byte of a clock correction sequence that was replicated and added to the RX elastic buffer. To use the RXRUNDISP port to indicate inserted idles instead of the current RX running disparity, CLK_COR_INSERT_IDLE_FLAG is set to TRUE.

Configurable Channel Bonding (Lane Deskew)

Overview

The RX elastic buffer can also be used for channel bonding. Channel bonding cancels out the skew between GTX transceiver lanes of the same column by using the RX elastic buffer as a variable latency block. The transmitter sends a pattern simultaneously on all lanes, which the channel bonding circuit uses to set the latency for each lane so that data is presented without skew at the FPGA RX interface.

Figure 7-31 shows a conceptual view of channel bonding.

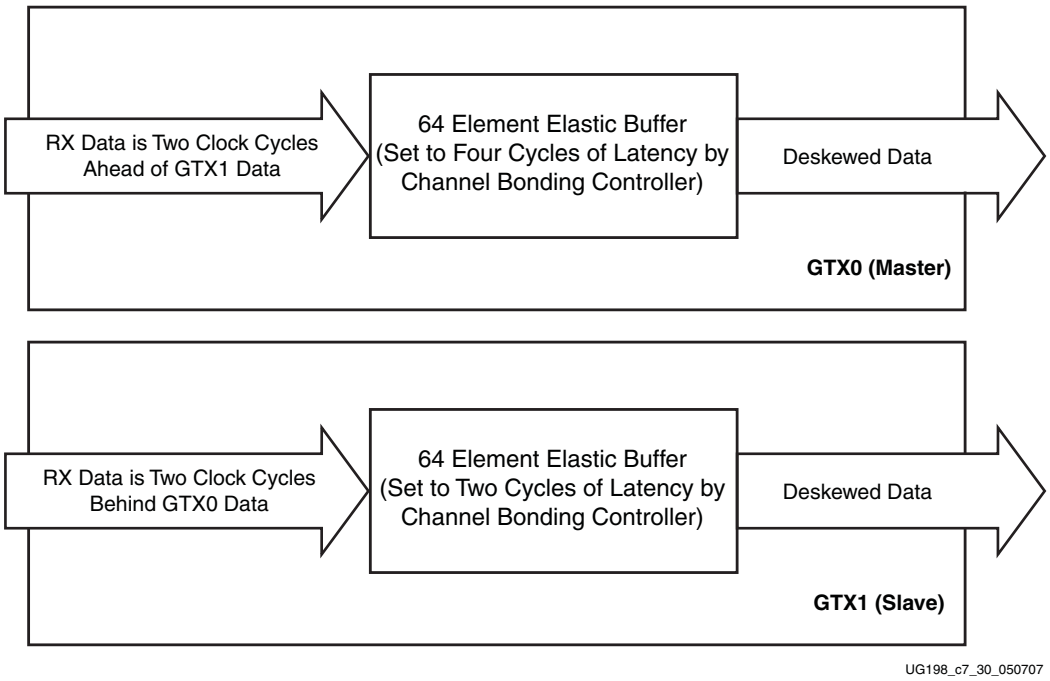


Figure 7-31: **Channel Bonding Conceptual View**

Ports and Attributes

Table 7-39 defines the channel bonding ports.

Table 7-39: **Channel Bonding Ports**

Port	Dir	Clock Domain	Description
INTDATAWIDTH	In	Async	Specifies the bit width for the TX and RX internal datapaths. This port controls both transceivers on the GTX_DUAL tile. 0: 16-bit width 1: 20-bit width
RXCHANBONDSEQ0 RXCHANBONDSEQ1	Out	RXUSRCLK2	Goes High when RXDATA contains the start of a channel bonding sequence.
RXCHANISALIGNED0 RXCHANISALIGNED1	Out	RXUSRCLK2	This signal from the RX elastic buffer goes High to indicate that the channel is properly aligned with the master transceiver according to observed channel bonding sequences in the data stream. This signal goes Low if an unaligned channel bonding sequence is detected, indicating that channel alignment was lost.
RXCHANREALIGN0 RXCHANREALIGN1	Out	RXUSRCLK2	This signal from the RX elastic buffer is held High for at least one cycle when the receiver has changed the alignment between this transceiver and the master.
RXCHBONDI0[3:0] RXCHBONDI1[3:0]	In	RXUSRCLK	FPGA channel bonding control. This signal is used only by *slaves*. It is driven from another transceiver's RXCHBONDO port that is the *master* in this configuration.
RXCHBONDO0[3:0] RXCHBONDO1[3:0]	Out	RXUSRCLK	FPGA channel bonding control. This signal is used by the *master* and *slaves* to pass channel bonding and clock correction control to other transceivers' RXCHBONDI ports.
RXENCHANSYNC0 RXENCHANSYNC1	In	RXUSRCLK2	Enable channel bonding (from the FPGA to the master). Tie this port High for slaves.

Table 7-40 defines the channel bonding attributes.

Table 7-40: **Channel Bonding Attributes**

Attribute	Type	Description
CB2_INH_CC_PERIOD_0 CB2_INH_CC_PERIOD_1	Integer	Allows retention or removal of control characters during channel bonding and clock correction for PCI Express designs.
CHAN_BOND_1_MAX_SKEW_0 CHAN_BOND_1_MAX_SKEW_1	Integer	These attributes control the number of USRCLK cycles that the master waits before ordering the slaves to execute channel bonding. This attribute determines the maximum skew that can be handled by channel bonding. It must always be less than the minimum distance (in bytes or 20-bit codes) between channel bonding sequences. Valid values range from 1 to 14.
CHAN_BOND_2_MAX_SKEW_0 CHAN_BOND_2_MAX_SKEW_1	Integer	
CHAN_BOND_KEEP_ALIGN	Boolean	Allows preservation of ALIGN characters during channel bonding for PCI Express designs.

Table 7-40: **Channel Bonding Attributes *(Cont'd)***

Attribute	Type	Description
CHAN_BOND_LEVEL_0 CHAN_BOND_LEVEL_1	Integer	CHAN_BOND_LEVEL indicates the amount of internal pipelining used for the elastic buffer control signals. A higher value permits more daisy-chaining of RXCHBONDO and RXCHBONDI to ease placement and routing constraints. To minimize required latency through the elastic buffer, CHAN_BOND_LEVEL in the master set to the smallest value possible for the required amount of daisy-chaining. See the "Description" section for channel bonding to learn how to set the channel bonding level. Valid values range from 0 to 7.
CHAN_BOND_MODE_0 CHAN_BOND_MODE_1	String	Defines the channel bonding mode of operation for this transceiver. OFF: No channel bonding. MASTER: This transceiver is master for channel bonding. Its RXCHBONDO port directly drives RXCHBONDI ports on one or more SLAVE transceivers. SLAVE: This transceiver is a slave for channel bonding. Its RXCHBONDI port is directly driven by the RXCHBONDO port of another SLAVE or MASTER transceiver. If its CHAN_BOND_LEVEL setting is greater than 0, its RXCHBONDO port may directly drive RXCHBONDI ports on one or more other SLAVE transceivers.
CHAN_BOND_SEQ_1_1_0 CHAN_BOND_SEQ_1_1_1	10-bit Binary	The CHAN_BOND_SEQ_1 attributes are used in conjunction with CHAN_BOND_SEQ_1_ENABLE to define channel bonding sequence 1. Each subsequence is 10 bits long. The rules for setting the subsequences depend on INTDATAWIDTH and RX_DECODE_SEQ_MATCH. See the "Description" section to learn how to set channel bonding subsequences. Not all subsequences need to be used. CHAN_BOND_SEQ_LEN determines how much of the sequence is used for a match. If CHAN_BOND_SEQ_LEN = 1, only CHAN_BOND_SEQ_1_1 is used. CHAN_BOND_SEQ_1_ENABLE can be used to make parts of the sequence don't care. If CHAN_BOND_SEQ_1_ENABLE[k] is 0, CHAN_BOND_SEQ_1_k is a don't-care subsequence and is always considered to be a match.
CHAN_BOND_SEQ_1_2_0 CHAN_BOND_SEQ_1_2_1		
CHAN_BOND_SEQ_1_3_0 CHAN_BOND_SEQ_1_3_1		
CHAN_BOND_SEQ_1_4_0 CHAN_BOND_SEQ_1_4_1		
CHAN_BOND_SEQ_1_ENABLE_0 CHAN_BOND_SEQ_1_ENABLE_1	4-bit Binary	

Table 7-40: **Channel Bonding Attributes *(Cont'd)***

Attribute	Type	Description
CHAN_BOND_SEQ_2_1_0 CHAN_BOND_SEQ_2_1_1	10-bit Binary	The CHAN_BOND_SEQ_2 attributes are used in conjunction with CHAN_BOND_SEQ_2_ENABLE to define the second channel bonding sequence. When CHAN_BOND_SEQ_2_USE is TRUE, the second sequence is used as an alternate sequence to trigger channel bonding. Each subsequence is 10 bits long. The rules for setting the subsequence depend on INTDATAWIDTH and RX_DECODE_SEQ_MATCH. See the "Description" section to learn how to set channel bonding sequences. Not all subsequences need to be used. CHAN_BOND_SEQ_LEN determines how many of the subsequences are used for a match. If CHAN_BOND_SEQ_LEN = 1, only CHAN_BOND_SEQ_2_1 is used. CHAN_BOND_SEQ_2_ENABLE can be used to make parts of the sequence don't care. If CHAN_BOND_SEQ_2_ENABLE[k] is 0, CHAN_BOND_SEQ_2_k is a don't care subsequence and is always considered to be a match.
CHAN_BOND_SEQ_2_2_0 CHAN_BOND_SEQ_2_2_1		
CHAN_BOND_SEQ_2_3_0 CHAN_BOND_SEQ_2_3_1		
CHAN_BOND_SEQ_2_4_0 CHAN_BOND_SEQ_2_4_1		
CHAN_BOND_SEQ_2_ENABLE_0 CHAN_BOND_SEQ_2_ENABLE_1	4-bit Binary	
CHAN_BOND_SEQ_2_USE_0 CHAN_BOND_SEQ_2_USE_1	Boolean	Determines if the second channel bonding sequence is to be used. TRUE: Channel bonding can be triggered by channel bonding sequence 1 or 2. FALSE: Channel bonding is only triggered by sequence 1.
CHAN_BOND_SEQ_LEN_0 CHAN_BOND_SEQ_LEN_1	Integer	Defines the length in bytes of the channel bonding sequence (1 to 4 bytes) that the transceiver matches to detect opportunities for channel bonding.
PCI_EXPRESS_MODE_0 PCI_EXPRESS_MODE_1	Boolean	The default for this attribute is TRUE for PCI Express designs. For all other protocols, the default setting is FALSE. Setting this attribute to TRUE enables certain operations specific to PCI Express operation, specifically, recognizing TXELECIDLE = 1, TXCHARDISPMODE = 1, TXCHARDISPVAL = 0 as a request to power down the channel. TXCHARDISPMODE = 1 and TXCHARDISPVAL = 0 encode the PIPE interface signal TXCompliance = 1 of the PIPE specification. The TXCHARDISPMODE and TXCHARDISPVAL settings encode for PIPE and enable special support for FTS lane deskew.

Description

Enabling Channel Bonding

Each GTX transceiver includes a circuit that performs channel bonding by controlling the pointers of the RX elastic buffer. To use channel bonding, the RX_BUFFER_USE attribute must be TRUE to turn on the elastic buffer.

Each GTX transceiver has a channel bonding circuit. Configuring a GTX transceiver for channel bonding requires the following steps:

1. Set the channel bonding mode for each GTX transceiver.
2. Set master to CHAN_BOND_MODE = MASTER.
3. Set slave to CHAN_BOND_MODE = SLAVE.

4. Connect the channel bonding port from the master to each slave, either directly or by daisy chaining.
5. Set the channel bonding sequence and detection parameters.

Channel Bonding Mode

The channel bonding mode for each GTX transceiver determines whether channel bonding is active and whether the GTX transceiver is the master or a slave. Each set of channel-bonded GTX transceivers must have one master and can have any number of slaves. To turn on channel bonding for a group of GTX transceivers, one transceiver is set to *Master*. The remaining GTX transceivers in the group are set to *Slaves*.

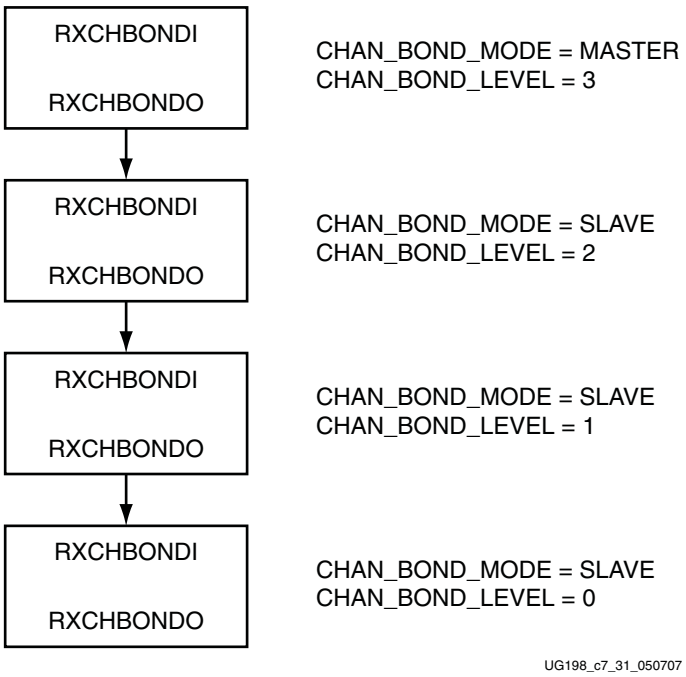
In the general use case, the master and the slaves of a channel bonding group are located in the same column (for example, the X0 column of a TXT device). An advanced case is when the master and the slaves of a channel bonding group are located in different columns. Contact your System I/O specialist for details on the advanced case.

Connecting Channel Bonding Ports

The channel bonding operation requires connecting the master GTX RXCHBONDO port to the RXCHBONDI port of all slaves in the group. A direct connection is required for adjacent GTX transceivers. To directly connect a master to a slave:

1. Connect the RXCHBONDO port of the master to the RXCHBONDI port of the slave.
2. Set the CHAN_BOND_LEVEL of the master to 1.
3. Set the CHAN_BOND_LEVEL of each slave to 0.

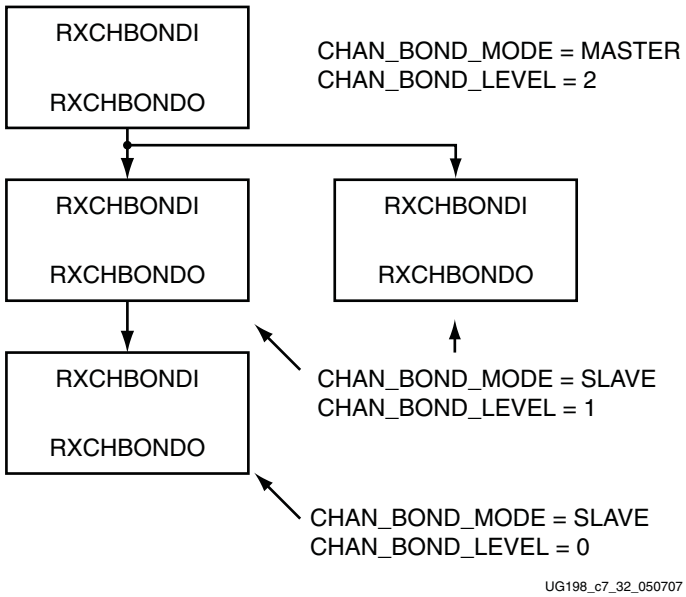
When GTX transceivers are directly connected, meeting the timing constraints becomes difficult as the transceivers get further apart. The solution to this problem is to connect the transceivers in a daisy chain. Daisy chaining is performed using the CHAN_BOND_LEVEL signal to allow additional pipeline stages between the master and the slave. The RXCHBONDO port of each slave is used as a pipeline stage in the RXCHBONDO path from the master. Figure 7-32 and Figure 7-33 show two daisy-chain examples.



Notes:

1. Each box can represent either transceiver in a GTX_DUAL tile.
2. This channel bonding daisy-chain example can be implemented using from two to four GTX_DUAL tiles.

Figure 7-32: **Channel Bonding Daisy Chain Example 1**



Notes:

1. Each box can represent either transceiver in a GTX_DUAL tile.
2. This channel bonding daisy-chain example can be implemented using from two to four GTX_DUAL tiles.

Figure 7-33: **Channel Bonding Daisy Chain Example 2**

To set up a daisy chain, first connect the GTX transceivers using RXCHBONDO and RXCHBONDI to create a path from the RXCHBONDI port of each slave to the RXCHBONDO port of the master. Use the following steps to set the CHAN_BOND_LEVEL for the GTX transceivers in the chain:

1. Set the CHAN_BOND_LEVEL of the master to 7.
2. Set the CHAN_BOND_LEVEL of each slave to the CHAN_BOND_LEVEL of the GTX transceiver driving the slave's RXCHBONDI port minus 1.
3. Find the slave with the lowest level. Subtract this level from the CHAN_BOND_LEVEL of all GTX transceivers so that the lowest slave has level 0 and the master has the minimum level required to service all the slaves.

Any number of GTX transceivers can be channel bonded together as long as the timing constraints of the design are met and all clocking guidelines are followed (see "Clocking from a Neighboring GTX_DUAL Tile," page 96).

When the connections between channel bonding ports among GTX transceivers are being decided, the designer must remember that RXCHBONDI and RXCHBONDO belong to the RXUSRCLK clock domain. Meeting the timing constraint of RXUSRCLK becomes increasingly difficult as RXUSRCLK increases in frequency and as directly connected transceivers get further apart.

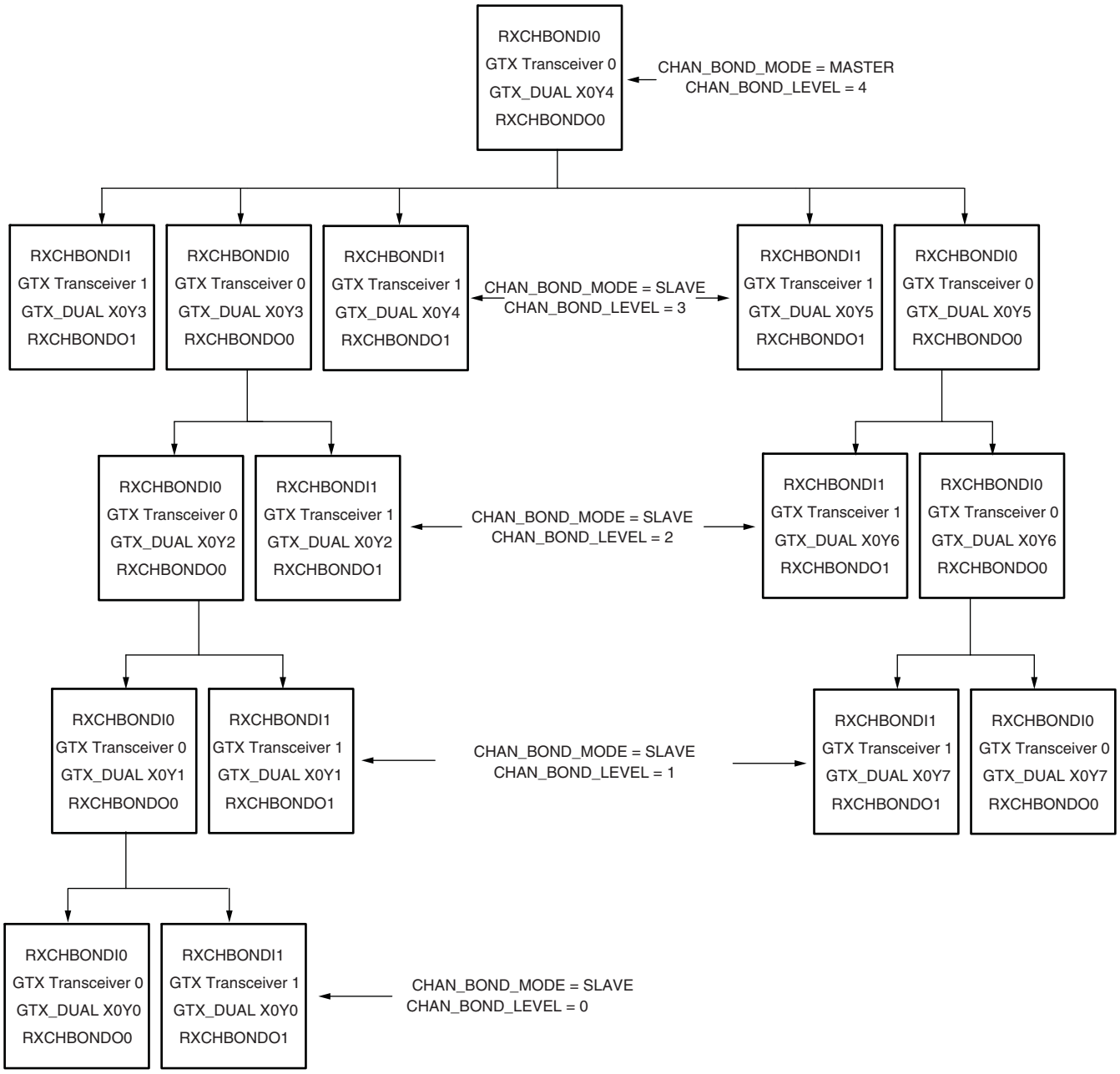
Selecting a GTX transceiver in the middle of the GTX_DUAL column to be the master for channel bonding allows for the most flexibility when connecting channel bonding ports. When the channel bonding master is in the middle of the GTX_DUAL column, connections can be made to GTX transceivers north and south of the master. This configuration allows for daisy chaining of up to seven levels north and seven levels south of the channel bonding master. Because of the GTX dedicated clock routing structure, an additional benefit of having the channel bonding master at the center of the GTX_DUAL column is that up to 14 GTX transceivers can be channel bonded together using a single clock pin pair. If more than 14 GTX transceivers are channel bonded together, the use of additional clock pins is required as well as using the same oscillator (see "Clocking from a Neighboring GTX_DUAL Tile," page 96).

Note: GTX transceivers that are channel bonded together *must* belong to the same column of the device.

As long as timing constraints are met, the following items apply:

- There is no limit to the number of GTX transceivers that can be on a particular CHAN_BOND_LEVEL.
- GTX transceivers from the same GTX_DUAL tile can be on the same CHAN_BOND_LEVEL.
- GTX transceivers from different GTX_DUAL tiles can be on the same CHAN_BOND_LEVEL.

Figure 7-34 shows an example for channel bonding 16 GTX transceivers. This example is only one of many ways to channel bond 16 GTX transceivers. In this example, transceivers of the X0 column are channel-bonded together.



Note: In this example, the GTX transceivers that are channel bonded together belong to the same column of the device. Contact your System I/O specialist for details on an advanced configuration, where the master and the slaves of a channel bonding group are located in different columns.

UG198_c7_33_072708

Figure 7-34: **Channel Bonding Example using 16 GTX Transceivers**

Setting the Channel Bonding Sequence

The channel bonding sequence is programmed in the same way as the clock correction sequence. CHAN_BOND_SEQ_LEN sets the length of the sequence from one to four subsequences. CHAN_BOND_SEQ_1_* sets the values of the sequence. If CHAN_BOND_SEQ_2_USE is TRUE, CHAN_BOND_SEQ_2_* sets the values for the alternate second sequence.

The number of active bits in each subsequence depends on INTDATAWIDTH (see Chapter 5, "Tile Features") and RX_DECODE_SEQ_MATCH (see "Configurable Clock Correction," page 210). Figure 7-35 shows how the subsequence bits are mapped.

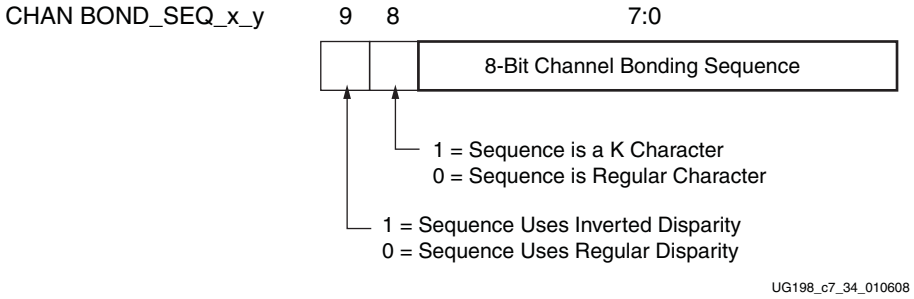


Figure 7-35: **Channel Bonding Sequence Settings**

As with clock correction sequences, channel bonding sequences can have *don't care* subsequences. CHAN_BOND_SEQ_1_ENABLE and CHAN_BOND_SEQ_2_ENABLE set these bytes. Figure 7-36 shows the mapping of the enable attributes for the channel bonding subsequences.

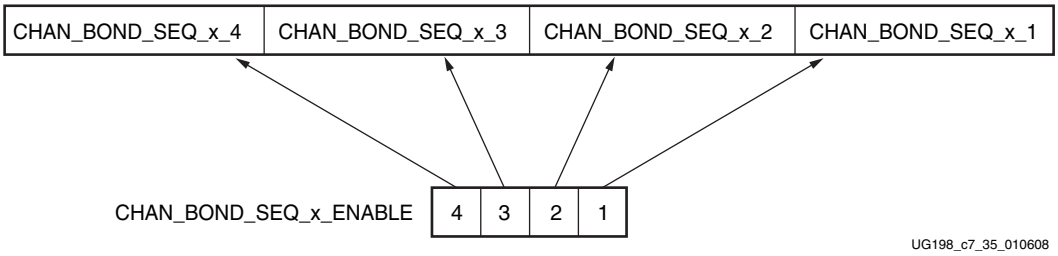


Figure 7-36: **Channel Bonding Sequence Mapping**

Setting the Maximum Skew

When the master receives a channel bonding sequence, it does not trigger channel bonding immediately. Several more bytes must arrive in case the slaves have more latency. This wait time effectively becomes the maximum skew that the RX elastic buffer can handle. If the skew is greater than this wait time, the slaves might not receive the sequence by the time the master triggers channel bonding (see Figure 7-37).

Figure 7-37 shows two FIFOs, one for the master and one for the slave. If the slave is behind the master, the master must wait several cycles before triggering channel bonding, otherwise the slow slave will not have the channel bonding sequence in its buffer.

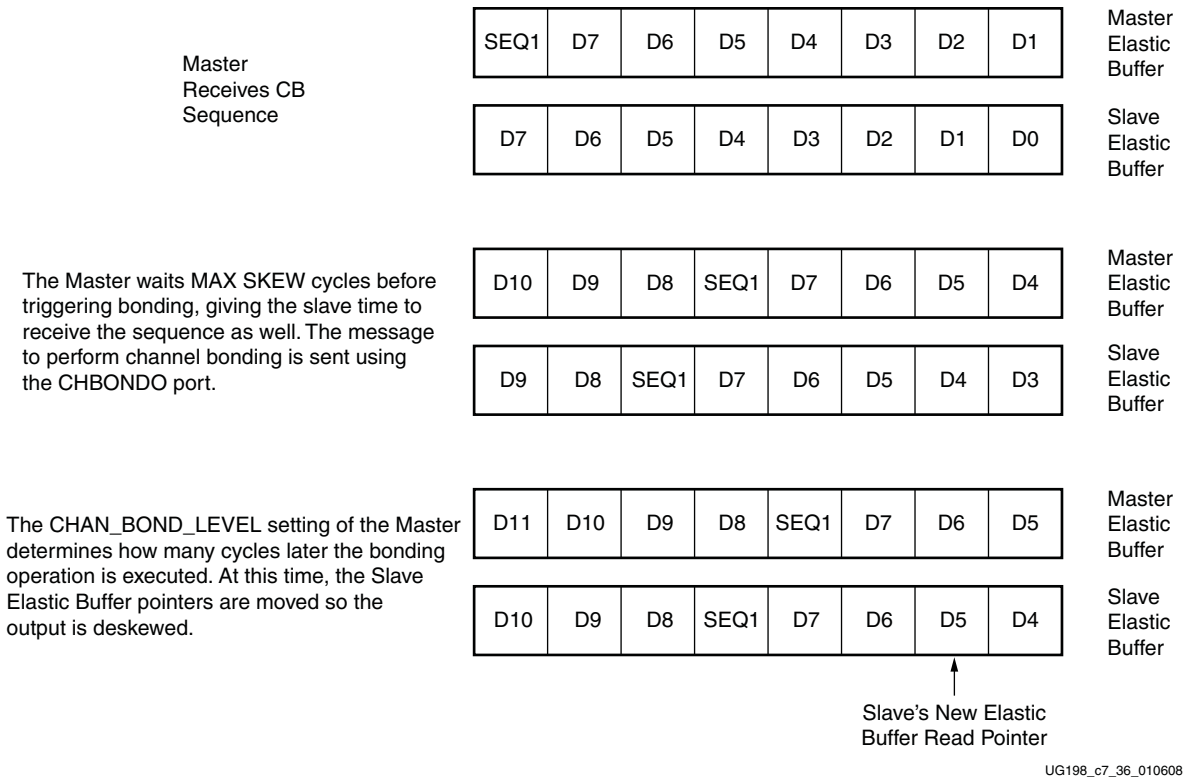


Figure 7-37: **Channel Bonding Example (MAX_SKEW = 2 and Master CHAN_BOND_LEVEL = 1)**

CHAN_BOND_1_MAX_SKEW and CHAN_BOND_2_MAX_SKEW are used to set the maximum skew allowed for channel bonding sequences 1 and 2, respectively. The maximum skew range is 1 to 14. The channel bond skew must be set no higher than the minimum distance allowed between channel bonding sequences in the data stream. This minimum distance is determined by the protocol being used.

Precedence between Channel Bonding and Clock Correction

The clock correction (see "Configurable Clock Correction," page 210) and channel bonding circuits both perform operations on the pointers of the RX elastic buffer. Normally, the two circuits work together without conflict, except when clock correction events and channel bonding events occur simultaneously. In this case, one of the two circuits must take precedence. To make clock correction a higher priority than channel bonding, CLK_COR_PRECEDENCE must be set to TRUE. To make channel bonding a higher priority, CLK_COR_PRECEDENCE must be set to FALSE.

RX Gearbox

Overview

The RX Gearbox uses output pins RXDATA[31:0] and RXHEADER[2:0] for receiving data. Similar to the"TX Gearbox," page 131, the RX Gearbox operates with the PMA using a single clock. Because of this, occasionally, the output data is invalid. Output pins RXHEADERVALID and RXDATAVALID determine if the appropriate header and data are valid. The RX Gearbox only supports 2-byte and 4-byte interfaces. A 1-byte interface is not supported.

The data out of the RX Gearbox is not necessarily aligned. Alignment is done in the FPGA logic. The RXGEARBOXSLIP(0/1) port can be used to slip the data from the Gearbox cycle-by-cycle until the correct alignment is reached. It takes a specified amount of cycles before the bitslip operation is processed and the output data is stable.

Descrambling of the data and block synchronization is done in the FPGA logic. The RocketIO GTX Transceiver Wizard has example code for each module.

Ports and Attributes

Table 7-41 defines the RX Gearbox ports.

Table 7-41: **RX Gearbox Ports**

Port	Direction	Clock Domain	Description
RXDATAVALID0 RXDATAVALID1	Out	RXUSRCLK2	Indicates a valid RXDATA. For example, during 64B/66B encoding, this signal is deasserted every 32 cycles for the 4-byte interface, and every 64 cycles for the 2-byte interface.
RXGEARBOXSLIP0 RXGEARBOXSLIP1	In	RXUSRCLK2	Asserting this signal slips the RX Gearbox position. It is used to achieve alignment with the FPGA logic. Asserting this port for one RXUSRCLK2 cycle changes the data alignment coming out of the gearbox. 4-byte interface: When RXGEARBOXSLIP is asserted for more than one cycle, the gearbox realigns the data once for each RXUSRCLK2 cycle that it is held High. 2-byte interface: RXGEARBOXSLIP must be deasserted for at least one cycle and then reasserted to cause a new realignment of the data. If multiple realignments occur in rapid succession, it is possible to pass the proper alignment point without recognizing the correct alignment point in the FPGA logic.
RXHEADER0[2:0] RXHEADER1[2:0]	Out	RXUSRCLK2	RX header bits.
RXHEADERVALID0 RXHEADERVALID1	Out	RXUSRCLK2	Indicates valid RX header bits. For example, during 64B/67B encoding, this signal is asserted every other cycle for the 4-byte interface and every 4 cycles for the 2-byte interface.
RXSTARTOFSEQ0 RXSTARTOFSEQ1	Out	RXUSRCLK2	Indicates that the RX Gearbox's internal counter is at 0.

Table 7-42 defines the RX Gearbox attributes.

Table 7-42: **RX Gearbox Attributes**

Attribute	Type	Description
GEARBOX_ENDEC_0[2:0] GEARBOX_ENDEC_1[2:0]	3-bit Binary	This attribute is shared between the RX and TX Gearboxes. The RX and TX Gearboxes cannot use different schemes. `000`: 64B/67B using external sequence counter `001`: 64B/66B using external sequence counter `010`: 64B/67B using internal sequence counter `011`: 64B/66B using internal sequence counter
RXGEARBOX_USE_0 RXGEARBOX_USE_1	Boolean	TRUE: Use the RX Gearbox. INTDATAWIDTH must be Low (16-bit internal data width mode) when the RX Gearbox is enabled. FALSE: Bypass the RX Gearbox. The RX Gearbox only supports 2-byte and 4-byte logic interfaces. When using the internal RX Gearbox, a 1-byte logic interface is not supported. A 1-byte logic interface can be implemented by using an external gearbox in the FPGA logic.

Description

Enabling the RX Gearbox

To enable the RX Gearbox for GTX transceiver 0, the RXGEARBOX_USE_0 attribute is set to TRUE. To enable the RX Gearbox for GTX transceiver 1, the RXGEARBOX_USE_1 attribute is set to TRUE.

Bit 2 of the GEARBOX_ENDEC attribute must be set to 0 to enable the Gearbox decoder. The decoder controls the GTX transceiver 's TX Gearbox and RX Gearbox. The TX Gearbox and RX Gearbox use the same mode.

RX Gearbox Operating Modes

The RX Gearbox operates the same in either external sequence counter mode or internal sequence counter mode. The RX Gearbox only supports 2-byte and 4-byte logic interfaces to the FPGA logic. A 1-byte logic interface is not supported.

As shown in Figure 7-38, either mode uses the RXDATA, RXHEADER, RXDATAOUTVALID, and RXHEADEROUTVALID outputs in addition to the RXGEARBOXSLIP input.

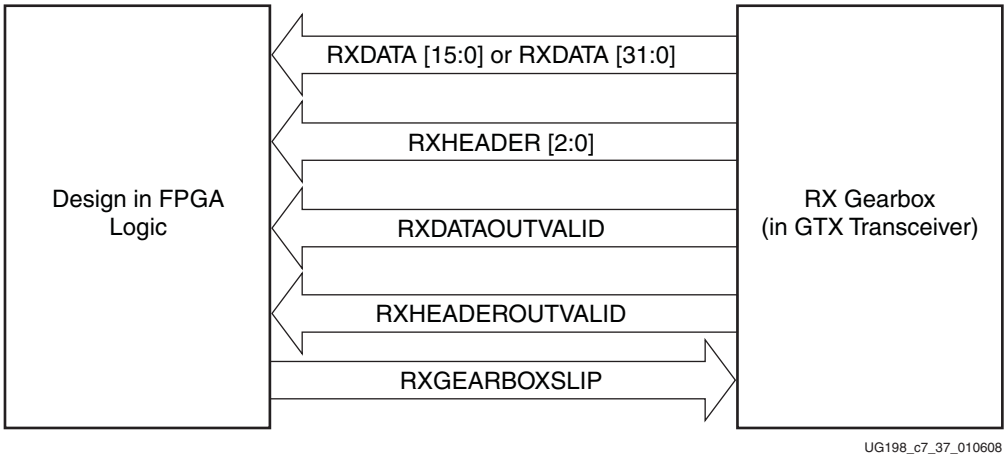


Figure 7-38: **RX Gearbox in Either Internal or External Sequence Mode**

The RX Gearbox internally manages all sequencing, which differs from the TX Gearbox option of either internal or external sequencing. Depending on whether a 2-byte or a 4-byte interface is used, RXDATAOUTVALID and RXHEADEROUTVALID assert and deassert for different periods of length. The RX Gearbox encounters similar data and header pauses found in the TX Gearbox. Figure 7-39 shows such a pause in addition to RXHEADERVALID asserting every other cycle and RXDATAVALID being deasserted for one cycle.

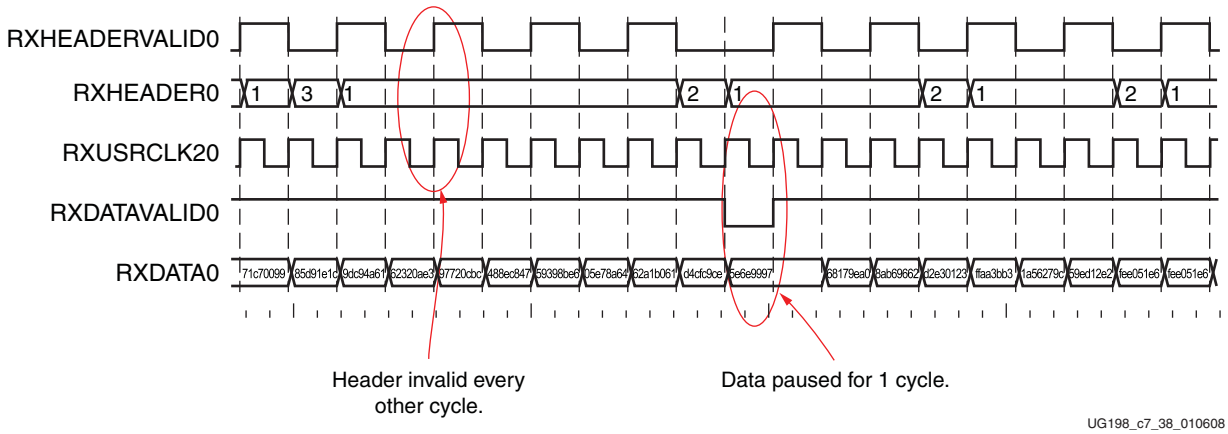


Figure 7-39: **RX Gearbox when using 64B/66B Encoding and a 4-Byte Interface**

Block Synchronization

The 64B/66B and 64B/67B protocols depend on block synchronization to determine their block boundaries. Block synchronization is required because all incoming data is unaligned before block lock is achieved. The goal is to search for the valid synchronization header by changing the data alignment. The RXGEARBOXSLIP input port is used to change the gearbox data alignment so that all possible alignments can be checked. The RXGEARBOXSLIP signal feeds back from the block synchronization state machine to the RX Gearbox and tells it to slip the data alignment. This process of slipping and testing the synchronization header repeats until block lock is achieved. When using the RX Gearbox, a block synchronization state machine is required in the FPGA logic. Figure 7-40 shows the operation of a block synchronization state machine. The RocketIO GTX Transceiver Wizard has example code for this type of module.

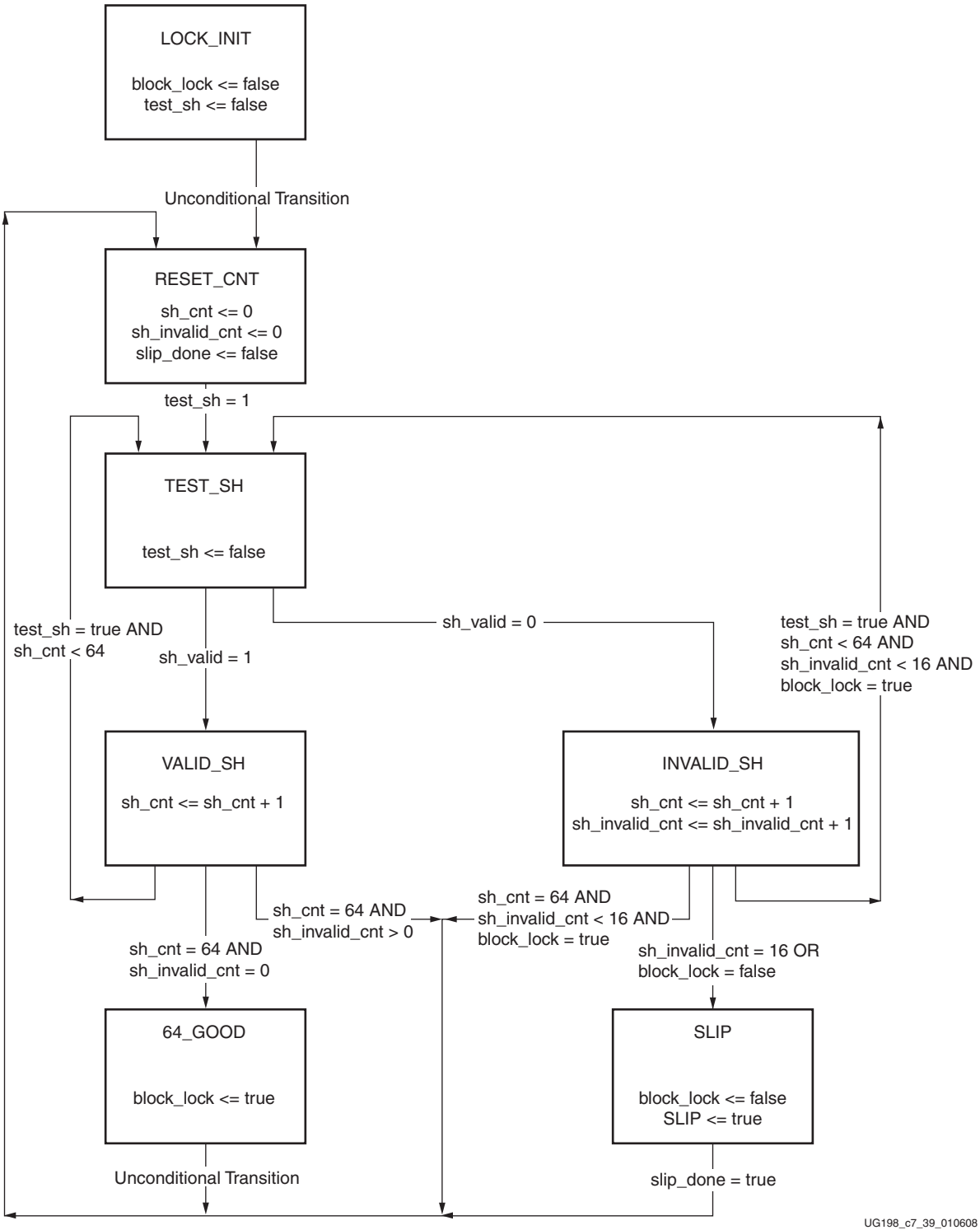


Figure 7-40: **Block Synchronization State Machine**

The state machine works by keeping track of valid and invalid synchronization headers. Upon reset, block lock is deasserted, and the state is LOCK_INIT. The next state is RESET_CNT where all counters are zeroed out. The synchronization header is analyzed in the TEST_SH state. If the header is valid, `sh_cnt` is incremented in the VALID_SH state,

otherwise sh_count and sh_invalid_count are incremented in the INVALID_SH state.

For the block synchronization state machine shown in Figure 7-40, sh_cnt_max and sh_invalid_cnt_max are both constants that are set to 64 and 16, respectively. From the VALID_SH state, if sh_cnt is less than the value sh_cnt_max and test_sh is High, the next state is TEST_SH. If sh_cnt is equal to sh_cnt_max and sh_invalid_cnt equals 0, the next state is GOOD_64 and from there block_lock is asserted. Then the process repeats again and the counters are cleared to zeros. To achieve block lock, the state machine must receive sh_cnt_max number of valid synchronization headers in a row without getting an invalid synchronization header. However, when block lock is achieved, sh_invalid_cnt_max – 1 number of invalid synchronization headers can be received within sh_cnt_max number of valid synchronization headers. Thus, once locked, it is harder to break lock.

Figure 7-41 shows a waveform of the block synchronization state machine asserting RXGEARBOXSLIP numerous times because of invalid synchronization headers before achieving data alignment.

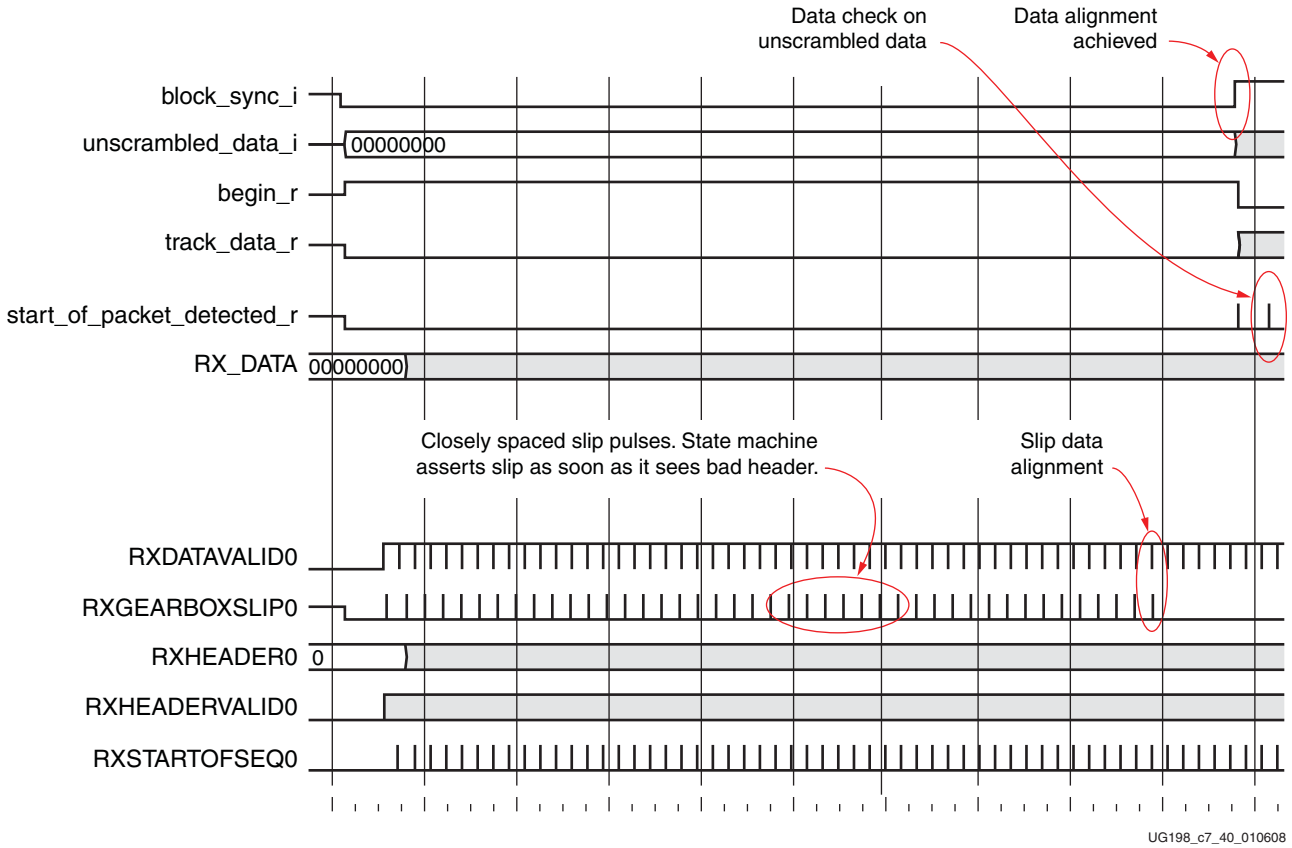


Figure 7-41: **RX Gearbox with Block Synchronization**

FPGA RX Interface

Overview

The FPGA receives RX data from the GTX receiver through the FPGA RX interface. Data is read from the RXDATA port on the positive edge of RXUSRCLK2.

The width of RXDATA can be configured to be one or two bytes wide. The actual width of the port depends on the internal data width of the GTX_DUAL tile, and whether or not the 8B/10B decoder is enabled. Ports widths of 8 bits, 10 bits, 16 bits, 20 bits, 32 bits, and 40 bits are possible.

The rate of the parallel clock (RXUSRCLK2) at the interface is determined by the RX line rate, the width of the RXDATA port, and whether or not 8B/10B decoding is enabled. RXUSRCLK must be provided for the internal PCS logic in the receiver. This section shows how to drive the parallel clocks and explains the constraints on those clocks for correct operation.

Ports and Attributes

Table 7-43 defines the FPGA RX interface ports.

Table 7-43: **FPGA RX Interface Ports**

Port	Dir	Clock Domain	Description
INTDATAWIDTH	In	Async	Specifies the bit width for the TX and RX paths. The bit width of TX and RX must be identical for both channels. 0: 16-bit width 1: 20-bit width
REFCLKOUT	Out	N/A	The REFCLKOUT port from each GTX_DUAL tile provides access to the reference clock provided to the shared PMA PLL (CLKIN). It can be routed for use in the FPGA logic.
RXDATA0[31:0] RXDATA1[31:0]	Out	RXUSRCLK2	Receive data bus of the receive interface to the FPGA. The width of RXDATA(0/1) depends on the setting of RXDATAWIDTH(0/1).
RXDATAWIDTH0 RXDATAWIDTH1	In	RXUSRCLK2	Selects the width of the RXDATA(0/1) receive data connection to the FPGA. 0: One-byte interface => RXDATA(0/1)[7:0] 1: Two-byte interface => RXDATA(0/1)[15:0] 2: Four-byte interface => RXDATA(0/1)[31:0] The clock domain depends on the selected clock (RXRECCLK(0/1), RXUSRCLK(0/1), and RXUSRCLK2(0/1)) for this interface.
RXRECCLK0 RXRECCLK1	Out	N/A	Recovered clock from the CDR. Clocks the RX logic between the PMA and the RX elastic buffer. Can be used to drive RXUSRCLK synchronously with incoming data. When RXPOWERDOWN[1:0] is set to 11, which is P2 the lowest power state, then RXRECCLK of this transceiver is indeterminate. RXRECCLK of this GTX transceiver is either a static 1 or a static 0.
RXRESET0 RXRESET1	In	Async	PCS RX system reset. Resets the RX elastic buffer, 8B/10B decoder, comma detect, and other RX registers. This is a per channel subset of GTXRESET.

Table 7-43: **FPGA RX Interface Ports *(Cont'd)***

Port	Dir	Clock Domain	Description
RXUSRCLK0 RXUSRCLK1	In	N/A	This port provides a clock for the internal RX PCS datapath. This clock must always be provided. The rate depends on INTDATAWIDTH where: INTDATAWIDTH is Low; $F_{RXUSRCLK}$ = Line Rate/16 INTDATAWIDTH is High; $F_{RXUSRCLK}$ = Line Rate/20
RXUSRCLK20 RXUSRCLK21	In	N/A	This port synchronizes the FPGA logic with the RX interface. This clock must be positive-edge aligned to RXUSRCLK. The clock rate depends on $F_{RXUSRCLK}$ and RXDATAWIDTH: RXDATAWIDTH = 0; $F_{RXUSRCLK2}$ = 2 x $F_{RXUSRCLK}$ RXDATAWIDTH = 1; $F_{RXUSRCLK2}$ = $F_{RXUSRCLK}$ RXDATAWIDTH = 2; $F_{RXUSRCLK2}$ = $F_{RXUSRCLK}/2$

There are no attributes in this section.

Description

The FPGA RX interface allows parallel received data to be read from the GTX transceiver. For this interface to be used, the following must be done:

- The width of the RXDATA port must be configured
- RXUSRCLK2 and RXUSRCLK must be connected to clocks running at the correct rate.

Configuring the Width of the Interface

Table 7-44 shows how to select the interface width for the RX datapath. 8B/10B decoding is discussed in more detail in "Configurable 8B/10B Decoder," page 198.

Table 7-44: **RX Datapath Width Configuration**

INTDATAWIDTH(1)	RXDATAWIDTH(2)	RXDEC8B10BUSE	FPGA RX Interface Width (bits)
0	0	N/A	8
0	1	N/A	16
0	2	N/A	32
1	0	0	10
1	1	0	20
1	2	0	40
1	0	1	8
1	1	1	16
1	2	1	32

Notes:

1. The internal datapath is 16 bits when INTDATAWIDTH is Low and 20 bits when INTDATAWIDTH is High.
2. The RXDATA interface is one byte wide when RXDATAWIDTH = 0, two bytes wide when RXDATAWIDTH = 1, and four bytes when RXDATAWIDTH = 2.

Figure 7-42 shows how RXDATA is received serially when the internal datapath is 16 bits (INTDATAWIDTH is Low) and 8B/10B decoding is disabled.

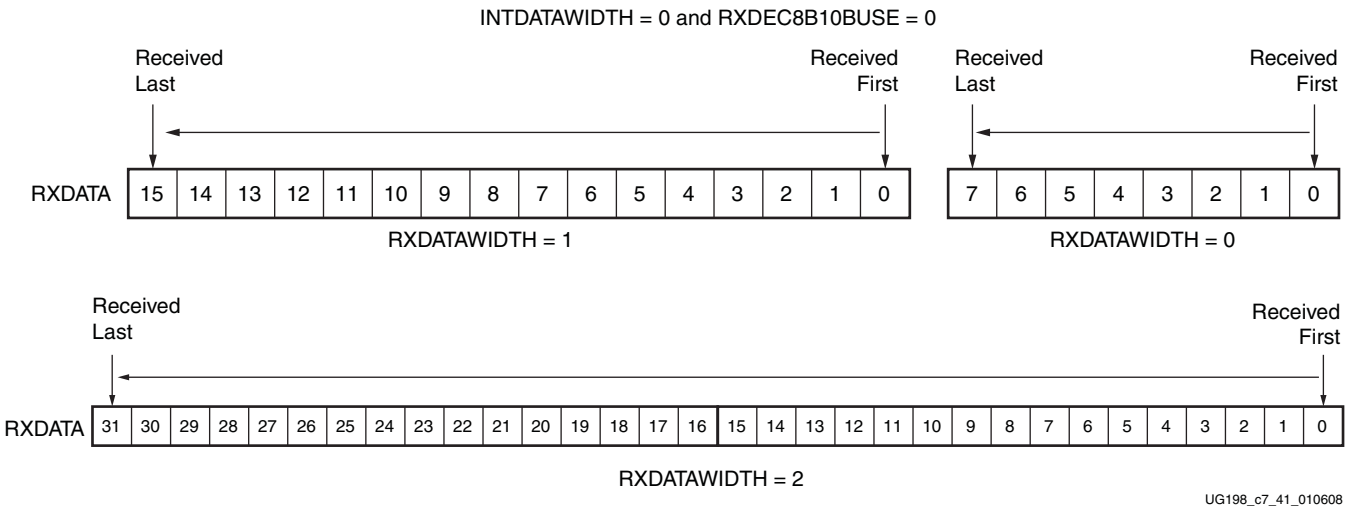


Figure 7-42: **RX Interface with 8B/10B Bypassed (16-Bit Internal Datapath)**

Figure 7-43 shows how RXDATA is received serially when the internal datapath is 20 bits (INTDATAWIDTH is High) and 8B/10B decoding is disabled. When RXDATA is 10 bits or 20 bits wide, the RXDISPERR and RXCHARISK ports are taken from the 8B/10B decoder interface and are used to present the extra bits.

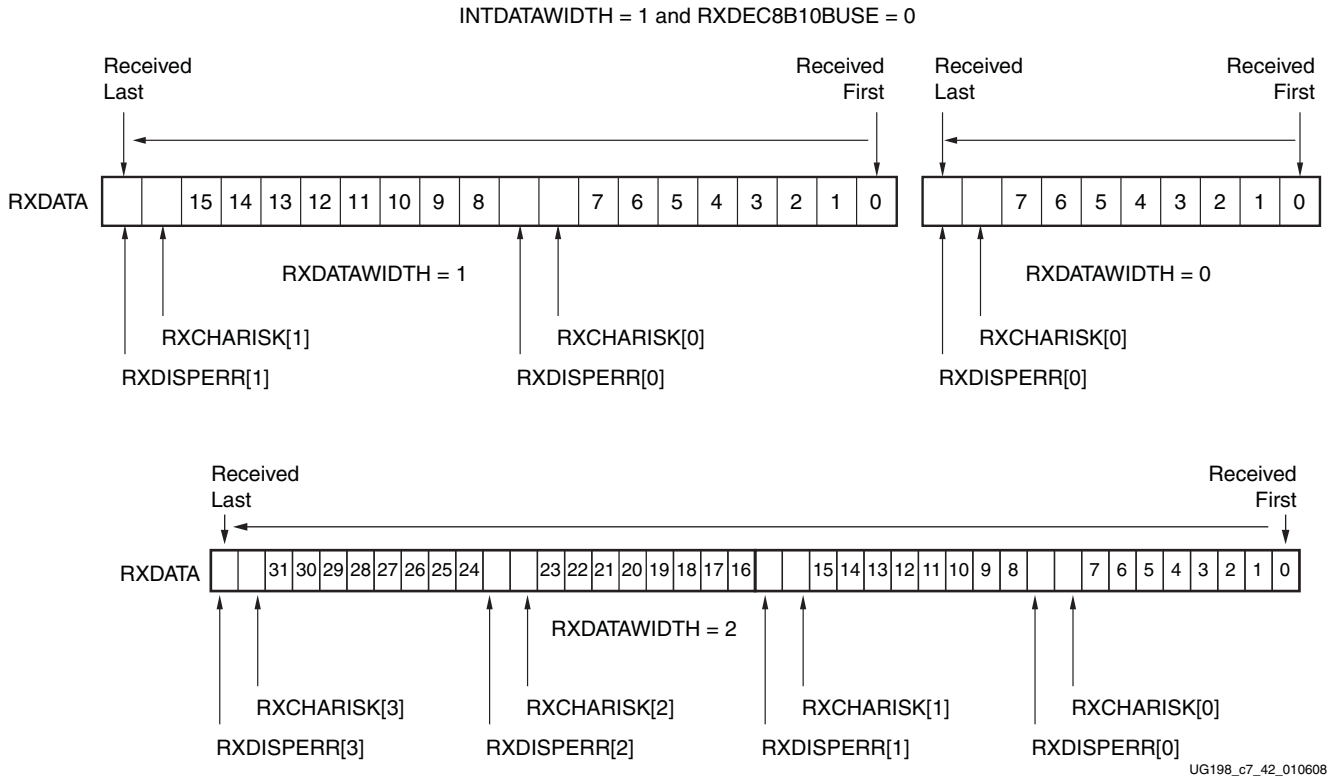


Figure 7-43: **RX Interface with 8B/10B Bypassed (20-Bit Internal Datapath)**

When 8B/10B decoding is used, the data interface is a multiple of 8 bits like in Figure 7-42, but the data is decoded before it is presented at the RXDATA port. Refer to "Configurable 8B/10B Decoder," page 198 for more details about bit ordering when using 8B/10B decoding.

Connecting RXUSRCLK and RXUSRCLK2

The FPGA RX interface includes two parallel clocks: RXUSRCLK and RXUSRCLK2. RXUSRCLK is the internal clock for the PCS logic in the GTX receiver. The required rate for RXUSRCLK depends on the internal datapath width of the GTX_DUAL tile (INTDATAWIDTH) and the RX line rate of the GTX receiver (see "Serial In to Parallel Out," page 181 to see how RX line rate is determined). Equation 7-7 shows how to calculate the required rate for RXUSRCLK.

$$\text{RXUSRCLK Rate} = \frac{\text{Line Rate}}{\text{Internal Datapath Width}} \qquad \textit{Equation 7-7}$$

RXUSRCLK2 is the main synchronization clock for all signals into the RX side of the GTX transceiver. Most signals into the RX side of the GTX receiver are sampled on the positive edge of RXUSRCLK2. RXUSRCLK2 is twice the rate of RXUSRCLK when RXDATAWIDTH is 0, the same rate as RXUSRCLK when RXDATAWIDTH = 1, and one half the rate of RXUSRCLK when RXDATAWIDTH is 2. Equation 7-8 through Equation 7-10 show how to calculate the required rate for RXUSRCLK2 based on RXDATAWIDTH.

$$\text{RXDATAWIDTH} = 0\text{: } F_{RXUSRCLK2} = 2 \times F_{RXUSRCLK} \qquad \textit{Equation 7-8}$$

$$\text{RXDATAWIDTH} = 1\text{: } F_{RXUSRCLK2} = F_{RXUSRCLK} \qquad \textit{Equation 7-9}$$

$$\text{RXDATAWIDTH} = 2\text{: } F_{RXUSRCLK2} = F_{RXUSRCLK}/2 \qquad \textit{Equation 7-10}$$

There are some rules about the relationships between clocks that must be observed for RXUSRCLK, RXUSRCLK2, and CLKIN.

- RXUSRCLK and RXUSRCLK2 must be positive-edge aligned, with as little skew as possible between them. Use low-skew clock resources (BUFGs and BUFRs) to drive RXUSRCLK and RXUSRCLK2. When the two are the same frequency, the same clock resource drives both. When the two are different frequencies, RXUSRCLK is divided to get RXUSRCLK2. The designer must ensure that the two are positive-edge aligned.
- If the channel is configured so the same oscillator drives the reference clock for the transmitter and the receiver, REFCLKOUT or TXOUTCLK can be used to drive RXUSRCLK and RXUSRCLK2 in the same way as they are used to drive TXUSRCLK and TXUSRCLK2. When clock correction is turned off, RX phase alignment must be used to align the serial clock and the parallel clocks. See "Configurable RX Elastic Buffer and Phase Alignment," page 202 for details about enabling phase alignment.
- If separate oscillators are driving the reference clocks for the transmitter and receiver on the channel, and clock correction is not used, RXUSRCLK and RXUSRCLK2 must be driven by RXRECCLK, and the phase-alignment circuit must be used.
- If clock correction is used, RXUSRCLK and RXUSRCLK2 can be sourced by RXRECCLK, REFCLKOUT, or TXOUTCLK.

Chapter 8

Cyclic Redundancy Check

Overview

In Virtex-5 devices, each high-speed transceiver tile is paired with two cyclic redundancy check (CRC) integrated blocks. Each CRC block can operate independently as two 32-bit input CRC modules (CRC32) or can be combined into a single 64-bit input CRC module (CRC64). The CRC modules use the standard 32-bit Ethernet polynomial for CRC calculation. The CRC integrated blocks are independent of the transceiver blocks.

Figure 8-1 shows the basic port interface of the CRC block.

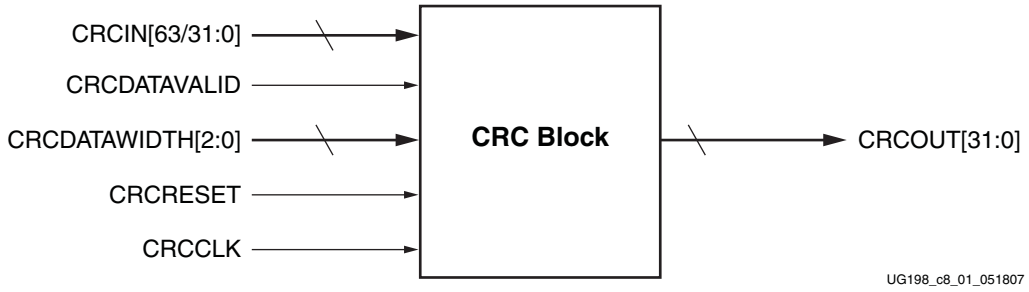


Figure 8-1: **CRC Integrated Block**

For clarification:

- Each GTX_DUAL tile is paired with two integrated CRC blocks.
- Each integrated CRC block can either operate as one 64-bit wide CRC module or as two independent 32-bit wide CRC modules.
- For a given GTX_DUAL tile, four independent 32-bit wide CRC modules are only possible when a 64-bit wide CRC module is not used.

Figure 8-2 shows how CRC modules are typically used in an application.

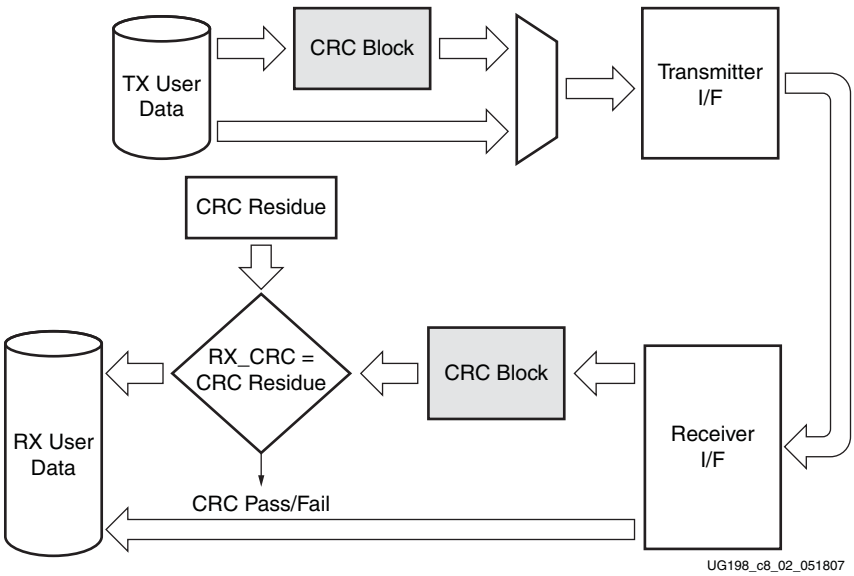


Figure 8-2: **CRC Application**

The CRC block only calculates the CRC for the input data stream using the standard polynomial. The CRC blocks do not perform any data framing. The application is responsible for appending CRC values to outgoing frames and validating the CRC on the RX side.

Ports and Attributes

Table 8-1 defines the CRC 64-bit I/O ports, and Table 8-2 defines the CRC 32-bit I/O ports.

Table 8-1: **CRC 64-Bit I/O Ports**

Port	Dir	Port Size	Clock Domain	Description
CRCCLK	In	1	N/A	CRC clock
CRCDATAVALID	In	1	CRCCLK	Indicates valid data on the CRCIN inputs. 1: Data valid 0: Data invalid Deasserting this signal causes the CRC value to be held for the number of cycles that this signal is deasserted.
CRCDATAWIDTH[2:0]	In	3	CRCCLK	Indicates how many input data bytes are valid. Refer to Table 8-4, page 238 and Table 8-5, page 238 for input byte ordering for CRC32 and CRC64, respectively.
CRCIN[63:0]	In	64	CRCCLK	CRC input data. The maximum datapath width is eight bytes.
CRCOUT[31:0]	Out	32	CRCCLK	32-bit CRC output. CRCOUT is the byte-reversed, bit-inverted CRC value corresponding to the CRC calculation on valid bytes from the previous clock cycle and the previous CRC value. **Note:** CRCDATAVALID must be driven High.
CRCRESET	In	1	CRCCLK	Synchronous reset of CRC registers. When CRCRESET is asserted, the CRC block is initialized to the CRC_INIT value.

Table 8-2: **CRC 32-Bit I/O Ports**

Port	Dir	Port Size	Clock Domain	Description
CRCCLK	In	1	N/A	CRC clock
CRCDATAVALID	In	1	CRCCLK	Indicates valid data on the CRCIN inputs. `1'b1`: Data valid `1'b0`: Data invalid Deasserting this signal causes the CRC value to be held for the number of cycles that this signal is deasserted.
CRCDATAWIDTH[2:0]	In	3	CRCCLK	Indicates how many input data bytes are valid. Refer to Table 8-4, page 238 and Table 8-5, page 238 for input byte ordering for CRC32 and CRC64, respectively.
CRCIN[31:0]	In	32	CRCCLK	CRC input data. The maximum datapath width is four bytes.
CRCOUT[31:0]	Out	32	CRCCLK	32-bit CRC output. CRCOUT is the byte-reversed, bit inverted CRC value corresponding to the CRC calculation on valid bytes from the previous clock cycle and the previous CRC value. **Note:** CRCDATAVALID must be driven High.
CRCRESET	In	1	CRCCLK	Synchronous reset of CRC registers. When CRCRESET is asserted, the CRC block is initialized to the CRC_INIT value.

Table 8-3 defines the CRC64 and CRC32 attributes.

Table 8-3: **CRC64/CRC32 Attributes**

Attribute	Type	Description
CRC_INIT[31:0]	32-bit Hex	32-bit value for initial state of CRC internal registers in the CRC32/CRC64 block. When CRCRESET is applied, the CRC registers are synchronously initialized to this value. The default value is `0xFFFFFFFF`.

Description

Using CRC for Error Checking

A CRC is an error-checking mechanism for a block of data, such as a frame of network traffic. The calculated CRC is used to detect errors after transmission or storage and is calculated on a per clock cycle basis.

A CRC is computed for the contents of a frame and appended to the end of the frame before transmission or storage. When the frame is received or retrieved, it is verified by recalculating the CRC for the contents to confirm that no changes occurred.

CRCs are simple to implement in digital hardware, easy to analyze mathematically, and good at detecting common errors caused by noise in transmission channels.

The CRC Primitive

Each CRC block computes a 32-bit CRC using the CRC32 polynomial specified for PCI Express, Gigabit Ethernet, and other common protocols. The CRC32 polynomial is:

$$G(X) = X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1 \quad \textit{Equation 8-1}$$

There are two primitives for instantiating CRC integrated blocks. The 32-bit CRC primitive (CRC32) can process 8, 16, 24, or 32-bit input data and generates a 32-bit CRC. The 64-bit primitive (CRC64) can process 8, 16, 24, 32, 40, 48, 56, or 64-bit input data and also generates a 32-bit CRC. Using the CRC64 primitive consumes both CRC integrated blocks paired with a given transceiver tile.

Table 8-4: **CRC32 – Valid Data Widths**

CRCDATAWIDTH[2:0]	Data Width	CRC Data Bus Bits
000	8-bit	CRCIN[31:24]
001	16-bit	CRCIN[31:16]
010	24-bit	CRCIN[31:8]
011	32-bit	CRCIN[31:0]

Notes:

1. CRCDATAWIDTH[2] must always be driven Low for the CRC32 primitive.

For CRC64, CRCDATAWIDTH is interpreted as indicated in Table 8-5.

Table 8-5: **CRC64 – Valid Data Widths**

CRCDATAWIDTH[2:0]	Data Width	CRC Data Bus Bits
000	8-bit	CRCIN[63:56]
001	16-bit	CRCIN[63:48]
010	24-bit	CRCIN[63:40]
011	32-bit	CRCIN[63:32]
100	40-bit	CRCIN[63:24]
101	48-bit	CRCIN[63:16]
110	56-bit	CRCIN[63:8]
111	64-bit	CRCIN[63:0]

Using the CRC Blocks

Figure 8-3 shows a CRC block calculating the CRC for input data. Also shown in this figure is the CRC32 primitive. This operation is performed when the CRC is being generated or checked. CRC_POLY is the fixed CRC32 polynomial used for all calculations.

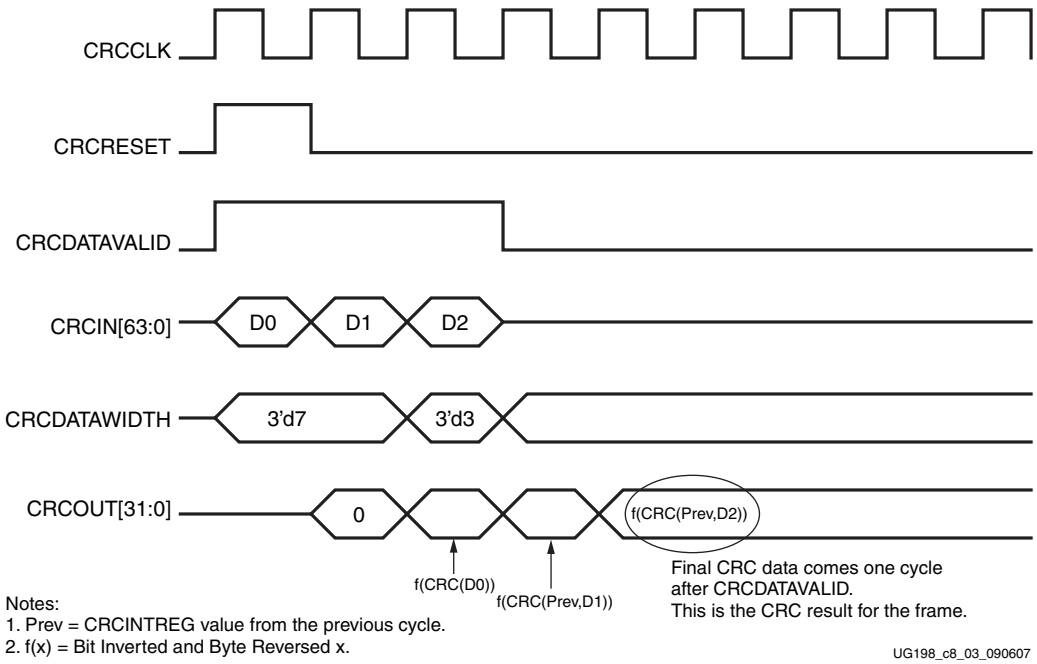


Figure 8-3: **CRC Block Timing**

At the start of each frame, CRCRESET must be applied to set the initial CRC value to CRC_INIT. CRC calculations are cumulative, so this step is required to start the CRC calculation at a known value. CRC_INIT is a 32-bit value for the initial state of the CRC internal register. Its default value is `0xFFFFFFFF`. When CRCRESET is driven Low, on the first cycle the CRC block outputs `0x00000000` on the CRCOUT port. The following cycles will have the calculated CRC value for the data on the CRCIN port. The CRC_INIT value required for a given protocol is specified as part of that protocol's CRC algorithm. Table 8-6 shows the CRC_INIT values for some common protocols that use the CRC32 polynomial.

Table 8-6: **CRC_INIT Values for Some Common Protocols**

Protocol	CRC_INIT
Ethernet	`32'hFFFF_FFFF`
PCI Express	`32'hFFFF_FFFF`
Infiniband	`32'hFFFF_FFFF`
Fibre Channel	`32'hFFFF_FFFF`
SATA	`32'h5232_5032`

The CRCDATAVALID port acts as a clock enable for the CRC block. If CRCDATAVALID is High and CRCRESET has been deasserted, a new CRC value is calculated every clock cycle, and the result appears on CRCOUT after one clock cycle. If CRCDATAVALID is Low, all CRC registers hold their values from the previous cycle.

CRCDATAWIDTH determines how many input data bytes are valid. Table 8-4, page 238 shows how to use CRCDATAWIDTH to set the number of valid input bytes on the CRC32 primitive. Table 8-5, page 238 shows the same for the CRC64 primitive. CRCDATAWIDTH is usually used at the end of a frame, when not all the bytes in a data word contain valid data. CRCDATAWIDTH can also be tied off to accommodate datapaths smaller than 32 bits wide.

To achieve faster throughput using the CRC block, apply the first data byte(s) and a reset on the same clock cycle. As shown in Figure 8-3, the CRC result appears on the output CRCOUT[31:0] one clock cycle after the last valid data byte is applied. If the fastest throughput is not required, the CRC module can be reset on any prior cycle as long as the CRCDATAVALID input is 0.

In Figure 8-4, the CRCRESET is asserted one CRCLK cycle after CRCDATAVALID. This allows back-to-back CRC frame calculations.

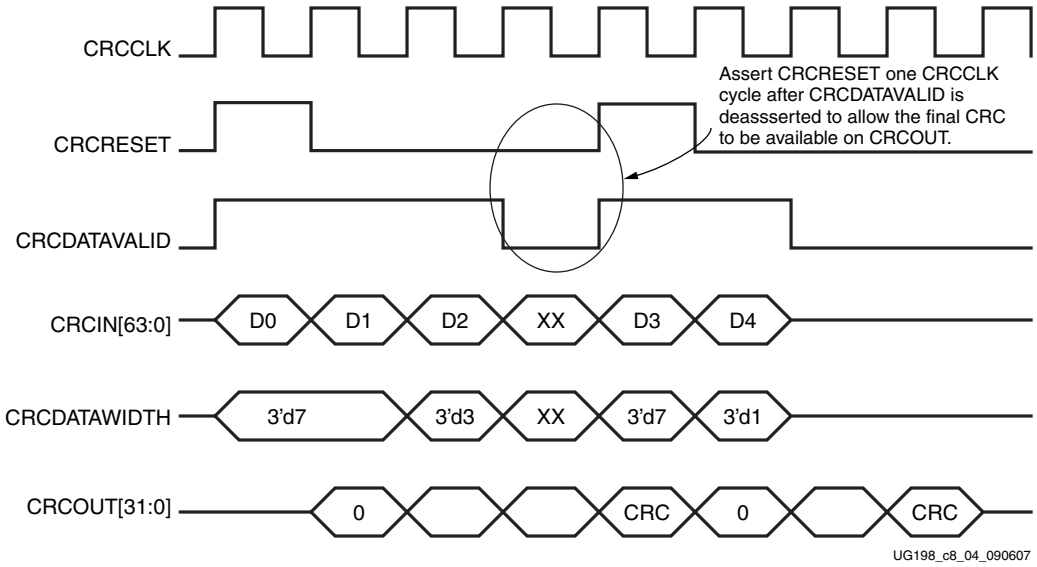


Figure 8-4: **CRCRESET Assertion Timing for Subsequent Frames**

In Figure 8-3, the internal CRCINTREG register shows the raw result of the CRC calculation. CRCOUT provides the bit-inverted, byte-reversed version of the CRC output at the same time. Figure 8-5 shows an example of the byte rotation and bit inversion operations.

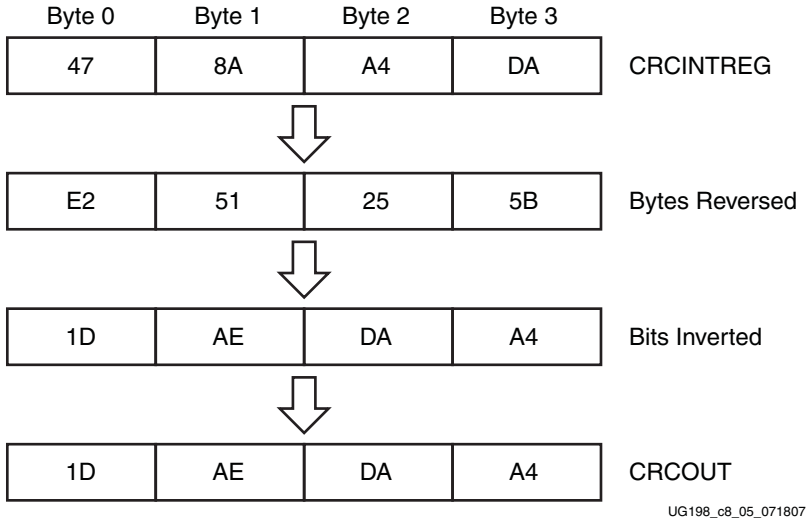


Figure 8-5: **Byte Rotation and Bit Inversion**

Byte rotation and bit inversion of the CRC output value are requirements for many common protocols, so CRCOUT is provided with these operations already performed to save FPGA resources. In designs that use a protocol that does not specify these operations in the CRC algorithm, the user must undo these operations before using the CRC value.

The CRC result for a given frame is the CRCOUT value corresponding to the final byte(s) of the frame. This value appears one cycle after the final byte(s) of the frame are presented to the CRCIN port with CRCDATAVALID High.

Integrating the CRC Blocks for TX

To use a CRC32 or CRC64 primitive to add CRC to frames for transmission or storage, the following logic needs to be built into the FPGA and connected to the CRC block:

- The Start of Frame (SOF) must trigger CRCRESET.
- If the datapath is not as wide as the maximum data width of the CRC primitive, CRCDATAWIDTH must be tied off so that only the bytes in the datapath are valid.
- The CRC output from the CRC block must be added to the frame, usually after the last data byte and before an End of Frame (EOF) character.

Integrating the CRC Blocks for RX

To use a CRC32 or CRC64 primitive to check the CRC on incoming data, the following logic needs to be built into the FPGA and connected to the CRC block:

- The SOF must trigger CRCRESET. This usually requires a decoder of some sort to find the SOF character in the incoming data stream.
- The EOF must trigger a CRC check.
  - If fixed length frames are used, this can be done with a counter.

- If variable length frames are used, this typically requires setting a count based on a length value in the frame, or decoding an EOF character in the incoming data stream.

- The CRC check must be performed. There are two common methods of CRC checking:
  - The compare method

    Calculate CRC over all the data in the frame, then extract the CRC from the frame and compare it to the calculated CRC. The EOF trigger is used to indicate which CRCOUT value should be used and which bytes of the frame contain the transmitted CRC.
  - The residue method

    Calculate CRC over all the data in the frame *and* the transmitted CRC, and then compare the result to the residue for the CRC32 polynomial. If the two values match, the CRC check passes. The residue value for CRC32, after bit inversion and byte reversal, is `32'h1CDF4421`.
- Remove the transmitted CRC from the frame, if necessary.

Implementation of the CRC Block

Figure 8-6 shows an implementation of the CRC.

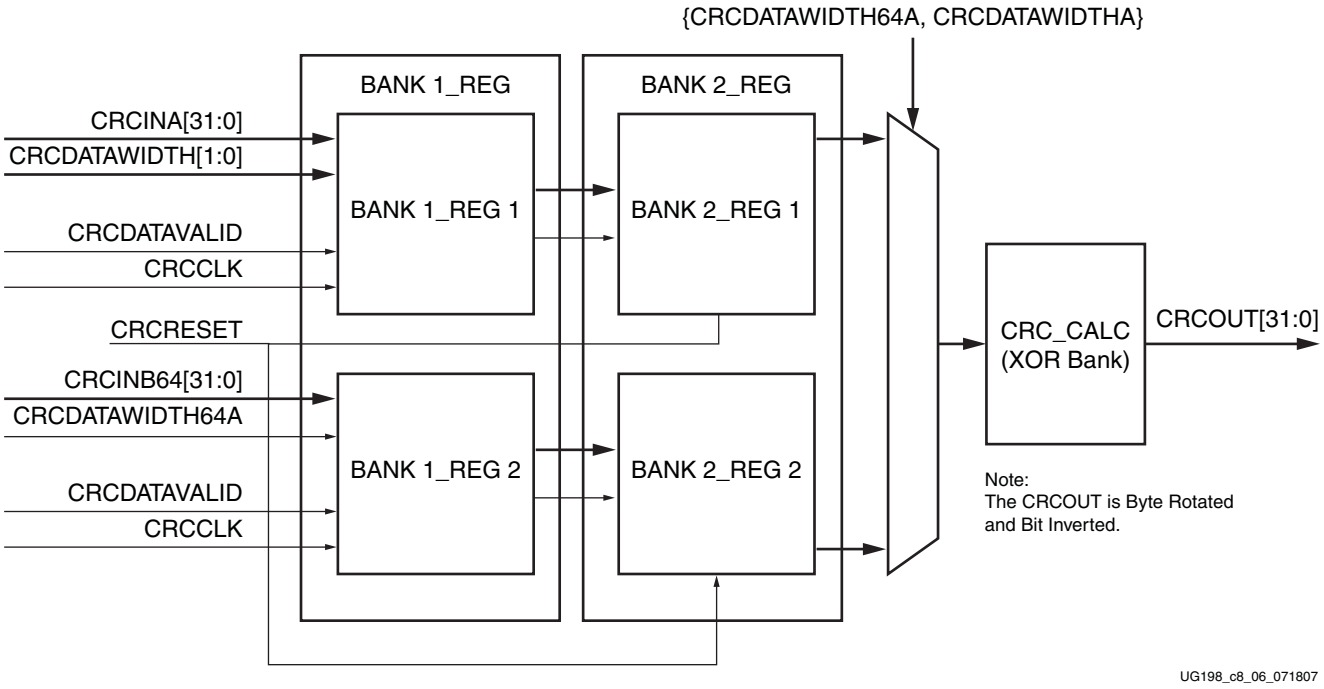


Figure 8-6: **CRC Implementation**

References

Refer to *IEEE 802.3 Cyclic Redundancy Check* [Ref 15] and *Configurable LocalLink CRC Reference Design* [Ref 16] for more information on the CRC.

Chapter 9

Loopback

Overview

Loopback modes are specialized configurations of the transceiver datapath where the traffic stream is folded back to the source. Typically, a specific traffic pattern is transmitted and then compared to check for errors. Figure 9-1 illustrates a loopback test configuration with four different loopback modes.

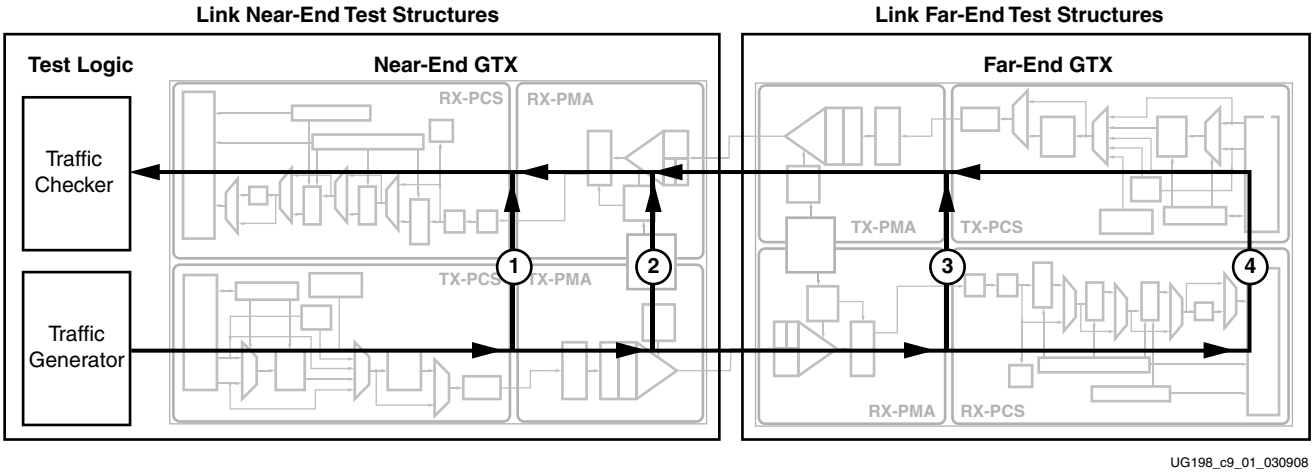


Figure 9-1: **Loopback Testing Overview**

Loopback test modes fall into two broad categories:

- Near-end loopback modes loop transmit data back in the transceiver closest to the traffic generator.
- Far-end loopback modes loop received data back in the transceiver at the far end of the link.

Loopback testing can be used either during development or in deployed equipment for fault isolation. The traffic patterns used can be either application traffic patterns, or specialized pseudo-random bit sequences. Each GTX transceiver has a built-in PRBS generator and checker.

Each channel of the GTX_DUAL tile features several loopback modes to facilitate testing:

- Near-End PCS Loopback (path 1 in Figure 9-1)
- Near-End PMA Loopback (path 2 in Figure 9-1)
- Far-End PMA Loopback (path 3 in Figure 9-1)
- Far-End PCS Loopback (path 4 in Figure 9-1)

The LOOPBACK[2:0] port selects between the normal operation mode and the different loopback modes.

Ports and Attributes

Table 9-1 defines the loopback ports.

Table 9-1: **Loopback Ports**

Port	Dir	Clock Domain	Description
LOOPBACK0[2:0] LOOPBACK1[2:0]	In	Async	000: Normal operation 001: Near-End PCS Loopback 010: Near-End PMA Loopback 011: Reserved 100: Far-End PMA Loopback 101: Reserved 110: Far-End PCS Loopback(1) 111: Reserved

Notes:

1. PCI Express compliant.

There are no attributes in this section.

Description

Near-End PCS Loopback

The test data is generated and checked by user logic and then looped back in the PCS. The difference compared to the Near-End PMA Loopback mode is that the PMA section is not involved. The test data is looped back before passing the parallel-to-serial and the serial-to-parallel converter. All analog high-speed circuits in the PMA section can be completely powered down. Figure 9-2 illustrates this configuration.

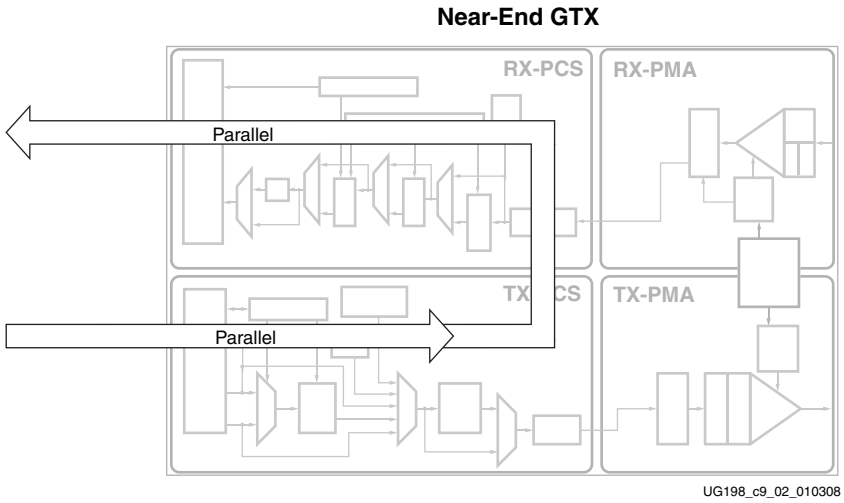


Figure 9-2: **Near-End PCS Loopback**

Near-End PMA Loopback

This mode uses the Near-End source for generating and checking the test data. The loopback occurs in the serial section of the PMA before the line drivers. The test data can be generated and checked by the built-in PRBS block inside the PCS or by user logic. The test data also appears on the package pins. Figure 9-3 shows the configuration without using the built-in PRBS.

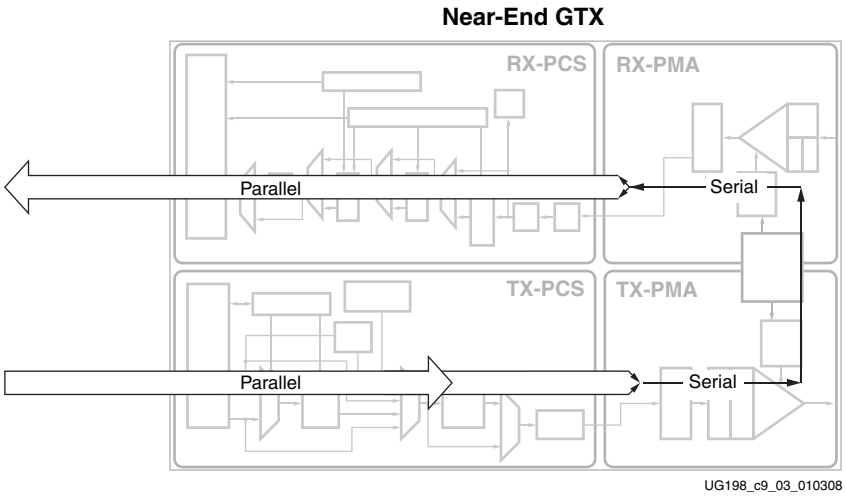


Figure 9-3: **Near-End PMA Loopback**

Marginal Conditions and Limitations

In the Near-End PMA loopback mode, the TXDATA input is routed through the PMA, serialized, and then looped back in through the RX side of the PMA and back out to RXDATA. When this mode is in use, there must be nothing driving the RXP/RXN serial inputs of the channel. The inputs can be left unconnected.

When the device is on a board, the remote transmitter should be 3-stated.

Far-End PMA Loopback

This mode uses the Near-End data source for generating and checking test data. The loopback occurs after passing the serial-to-parallel converter of the PMA. This mode tests the complete PMA section including the serial-to-parallel and parallel-to-serial conversion. Almost the entire PCS section is bypassed because the RX serial inputs are looped back to the TX serial outputs. Figure 9-4 illustrates this mode.

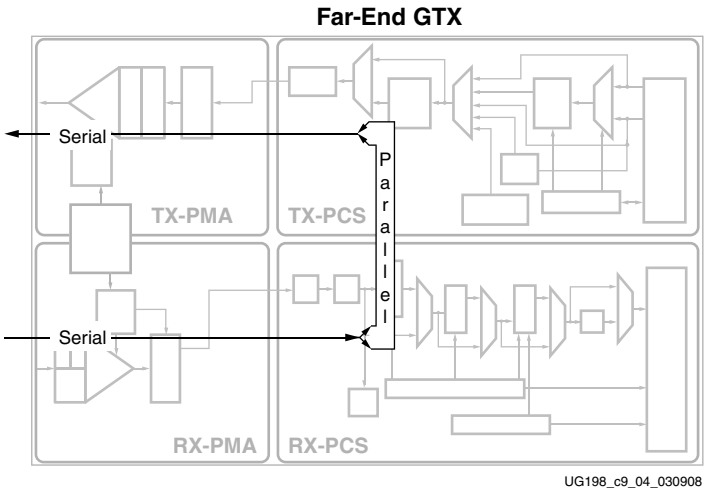


Figure 9-4: **Far-End PMA Loopback**

Marginal Conditions and Limitations

When the Far-End PMA loopback mode is used, disregarding PLL_RXDIVSEL_OUT, PLL_TXDIVSEL_OUT must be set to `1`. Refer to "Shared PMA PLL," page 84 for details about the TX divider settings.

In Far-End PMA loopback mode, the TX buffer is borrowed for the parallel loopback path to compensate for possible phase differences between the TX and RX parallel clocks. For this reason, TX_BUFFER_USE must be TRUE when using this mode. This mode can only be used for one transceiver at a time on any given tile due to clocking restrictions.

When one channel of a GTX_DUAL tile is placed in the Far-End PMA loopback mode, the TX side of the other channel no longer operates reliably. Do not attempt nor expect reliable use of the second channel while the first is in `100` loopback.

Far-End PCS Loopback

This mode uses an external source to generate and check the test data. The loopback occurs in the PCS section and the received data is presented to the user logic.

The transmit data of the user logic is not transmitted when this mode is activated. This mode is a PCI Express compliant loopback mode that echoes the received data back to the sender. Figure 9-5 illustrates this configuration.

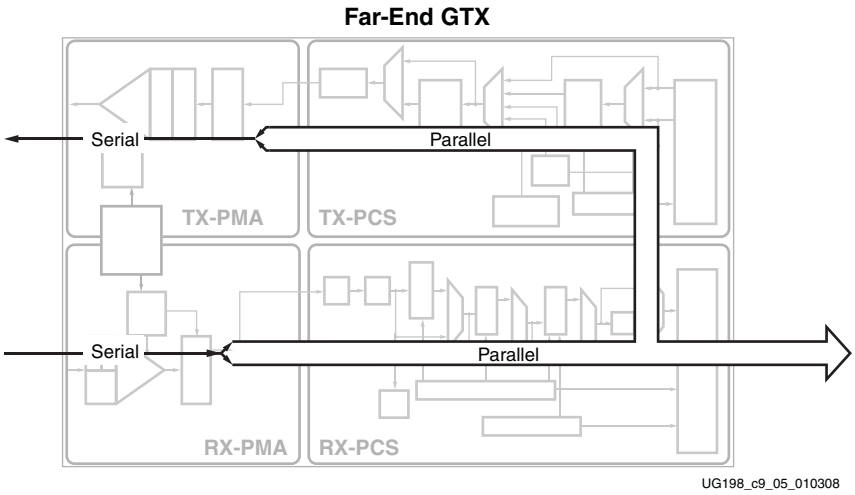


Figure 9-5: **Far-End PCS Loopback**

Chapter 10

GTX-to-Board Interface

Analog Design Guidelines

Overview

In designs with FPGAs that contain GTX transceivers, the overall system performance of a communication link is highly dependent on the characteristics of the power supply and clocking design on both endpoints. This section discusses guidelines and recommendations for these topics.

As a prerequisite, the design guidelines outlined in the *Virtex-5 FPGA PCB Designer's Guide* must be observed to keep the power supply and switching noise on the board to a minimum. Additionally, it is highly recommended to use Point-of-Load (POL) power distribution techniques as outlined in:
http://www.xilinx.com/publications/xcellonline/xcell_57/xc_pdf/p105-107_57-bellinix.pdf

Refer to the book *EMC and the Printed Circuit Board* [Ref 8] by Mark I. Montrose, sponsored by the IEEE Electromagnetic Compatibility Society, for additional guidelines.

Implementation of these guidelines not only improves system margins but is a prerequisite for compliance to regulations as defined by Federal Communications Commission (FCC) and the Verband Deutscher Elektrotechniker (VDE) regarding Electromagnetic Compatibility (EMC), Electromagnetic Interference (EMI), and Radio Frequency Interference (RFI).

Ports and Attributes

Table 10-1 defines the analog pins.

Table 10-1: **Analog Pins**

Pins	Dir	Description
MGTAVCC	In (Pad)	MGTAVCC is the analog supply for the internal analog circuits of the GTX_DUAL tile.
MGTAVCCPLL	In (Pad)	MGTAVCCPLL is the analog supply for the shared PMA PLL of the GTX_DUAL tile.
MGTAVTTRX	In (Pad)	MGTAVTTRX is the analog supply for the receiver circuits and termination of the GTX_DUAL tile.
MGTAVTTRXC	In (Pad)	FXT only: MGTAVTTRXC is the analog supply for resistor calibration and standby circuit of the entire device.

Table 10-1: **Analog Pins *(Cont'd)***

Pins	Dir	Description
MGTAVTTRXC_R	In (Pad)	TXT only: MGTAVTTRXC_R is the analog supply for resistor calibration and standby circuit of the right transceiver column.
MGTAVTTRXC_L	In (Pad)	TXT only: MGTAVTTRXC_L is the analog supply for resistor calibration and standby circuit of the left transceiver column.
MGTAVTTTX	In (Pad)	MGTAVTTTX is the analog supply for the transmitter termination and driver circuits and reference clock routing of the GTX_DUAL tile.
MGTREFCLKP MGTREFCLKN	In (Pad)	Differential clock input pin pair(1) for the reference clock of the GTX_DUAL tile.
MGTRREF	In (Pad)	FXT only: MGTRREF is the reference resistor input for the entire device.
MGTRREF_R	In (Pad)	TXT only: MGTRREF_R is reference resistor input for the right transceiver column.
MGTRREF_L	In (Pad)	TXT only: MGTRREF_L is reference resistor input for the left transceiver column.

Notes:

1. This clock can only be accessed by the FPGA logic through the REFCLKOUT port.

Table 10-2 defines the analog attributes.

Table 10-2: **Analog Attributes**

Attribute	Type	Description
CLKINDC_B	Boolean	Must be set to TRUE. Oscillators driving the dedicated reference clock inputs must be AC coupled. When set to FALSE for testing, the common mode voltage of the driving circuit must match the common mode voltage of the differential clock input pair (MGTCLKP, MGTCLKN). The differential swing must not exceed the maximum differential swing of the clock input pair.(1, 2)
CLKRCV_TRST	Boolean	When set to FALSE, switches off the internal termination resistors of the differential clock input pair. This results in a high-impedance input characteristic that is only intended for testing. When set to TRUE, the differential clock input pair is terminated with a 100Ω differential impedance. Each clock input in (MGTCLKP, MGTCLKN) is contacted via a 50Ω resistor to a midterm nominal voltage of 0.8V.(1, 2)
TERMINATION_CTRL[4:0]	5-bit Binary	Controls the internal termination calibration circuit. Refer to Table 10-3, page 257 for encoding.

Table 10-2: **Analog Attributes *(Cont'd)***

Attribute	Type	Description
TERMINATION_IMP_0 TERMINATION_IMP_1	Integer	Selects the termination impedance for the TX driver and receiver. See Figure 10-7 calibrating the impedance values. Always set to 50, which selects the 50Ω termination impedance. The RocketIO GTX Transceiver Wizard automatically sets the TERMINATION_IMP_(0/1) attributes to 50.
TERMINATION_OVRD	Boolean	Selects whether the external 59Ω(3) precision resistor connected to the MGTRREF pin or an override value is used, as defined by TERMINATION_CTRL[4:0].

Notes:

1. Violation of the rules outlined in this section result in a marginal or dysfunctional design, device degradation in the future, or device damage.
2. Consult DS202: *Virtex-5 FPGA Data Sheet* for the common mode voltage values and the associated differential swing and operating conditions.
3. The nominal value of the external precision resistor RREF connected to the MGTRREF pin is different for LXT/SXT devices and FXT/TXT devices. For LXT/SXT devices with GTP_DUAL tiles, RREF is 50Ω nominal. For FXT/TXT devices with GTX_DUAL tiles, RREF is 59Ω nominal.

Description

Each Virtex-5 FXT device requires one 59Ω external precision (1%) resistor on the PCB (connected directly to the MGTRREF pin and to the closest MGTAVTTTX pin). A TXT device requires two external precision resistors, one resistor for each column of GTX_DUAL tiles.

Note: The nominal value of the external precision resistor RREF connected to the MGTRREF pin is different for LXT/SXT devices and FXT/TXT devices. For LXT/SXT devices with GTP_DUAL tiles, RREF is 50Ω nominal. For FXT/TXT devices with GTX_DUAL tiles, RREF is 59Ω nominal.

Each used GTX_DUAL tile requires a filter circuit on the following analog power supply pins:

- MGTAVCCPLL
- MGTAVTTTX
- MGTAVTTRX
- MGTAVCC

The correct implementation and placement of these filter circuits is important for suppressing high-frequency noise.

Figure 10-1 illustrates the connection of the external precision resistor to the MGTRREF and MGTAVTTTX pins of the device. Each Virtex-5 FXT device requires a filter circuit on the MGTAVTTRXC pin, which powers the resistor calibration circuit of the device. Each TXT device has two GTX_DUAL columns and therefore two resistor calibration circuits (one for each GTX_DUAL column).

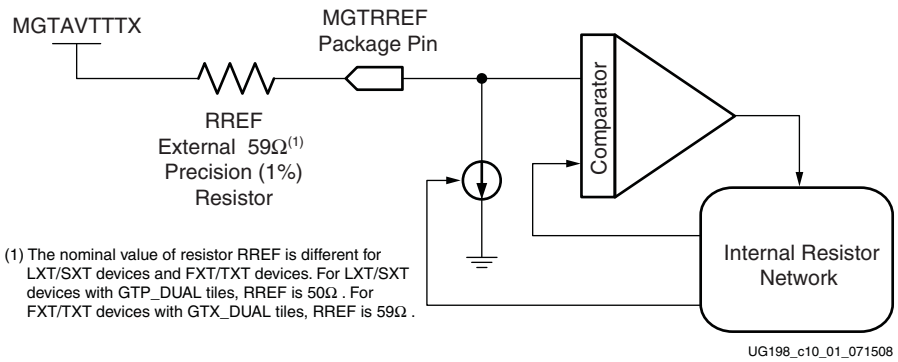


Figure 10-1: **Resistor Calibration Circuit**

Figure 10-2 illustrates the view from the schematic design perspective.

Note:

- FXT only: If any GTX transceivers are used in a column, the GTX_DUAL 112 tile that contains the calibration block *must* be connected as illustrated in Figure 10-2.
- TXT only: If any GTX transceivers are used in a right column, the GTX_DUAL 112 tile that contains the calibration block for the right column *must* be connected as illustrated in Figure 10-2. If any GTX transceivers are used in a left column, the GTX_DUAL 111 tile that contains the calibration block for the left column *must* be connected as illustrated in Figure 10-2.

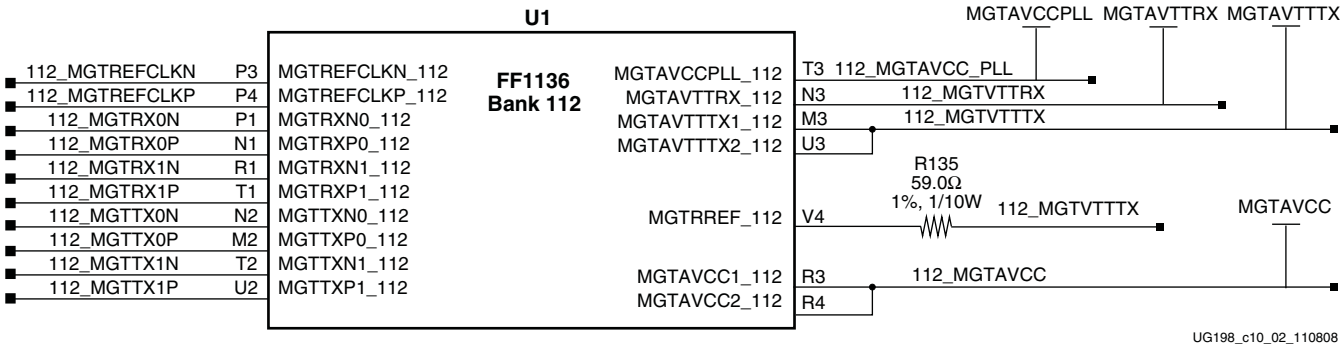
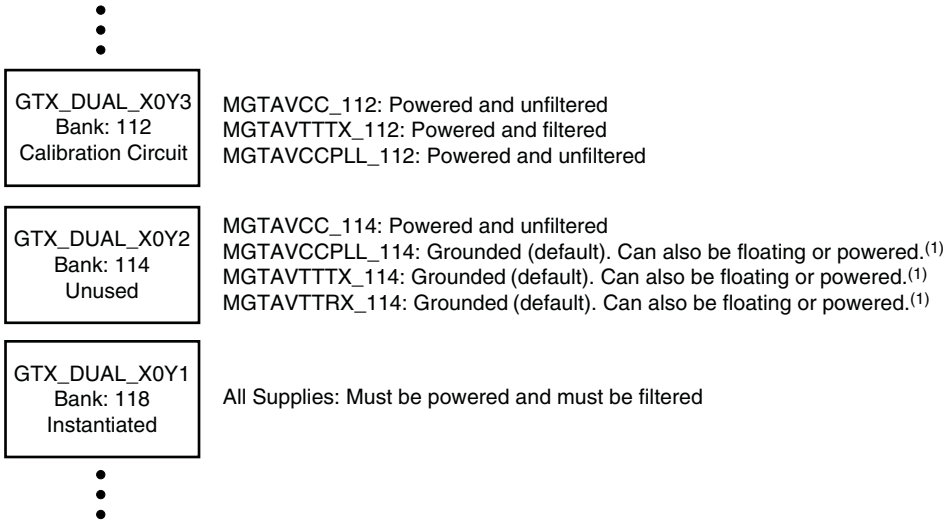


Figure 10-2: **RREF Resistor Schematic Design Perspective**

Figure 10-3 illustrates the scenario of a partially used GTX_DUAL column with an unused GTX_DUAL 112 tile.

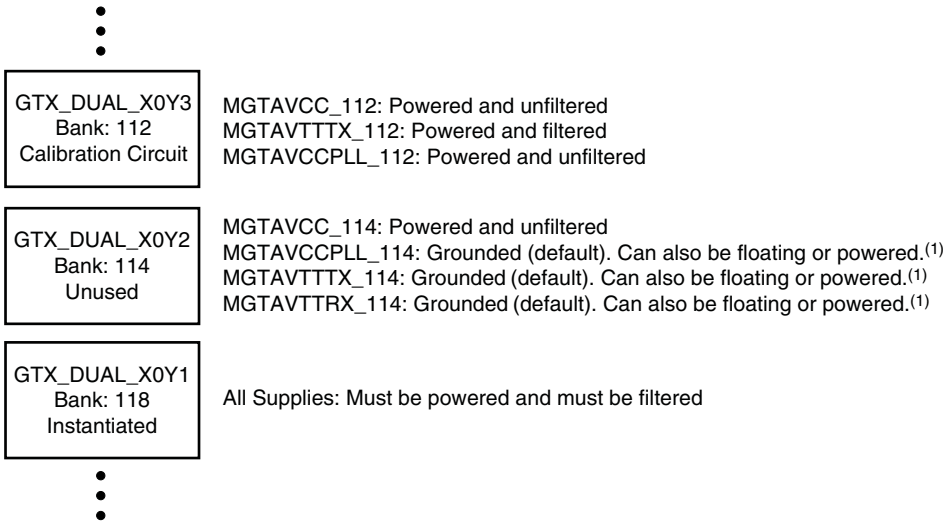


Note:
1. Grounding the supplies of unused GTX_DUAL tiles further improves the noise immunity of the device. Powering the supplies of unused GTX_DUAL tiles increases the power consumption of the device compared to leaving them floating or connecting them to ground. However, leaving the supply pins of unused GTX_DUAL tiles floating or powered is less optimal, but acceptable because they do not affect operation.

UG198_c10_12_110808

Figure 10-3: **Partially Used GTX_DUAL Column (FXT Devices and Right Column in TXT Devices)**

Figure 10-4 illustrates the scenario of a partially used GTX_DUAL column with an unused GTX_DUAL 111 tile.



Note:
1. Grounding the supplies of unused GTX_DUAL tiles further improves the noise immunity of the device. Powering the supplies of unused GTX_DUAL tiles increases the power consumption of the device compared to leaving them floating or connecting them to ground. However, leaving the supply pins of unused GTX_DUAL tiles floating or powered is less optimal, but acceptable because they do not affect operation.

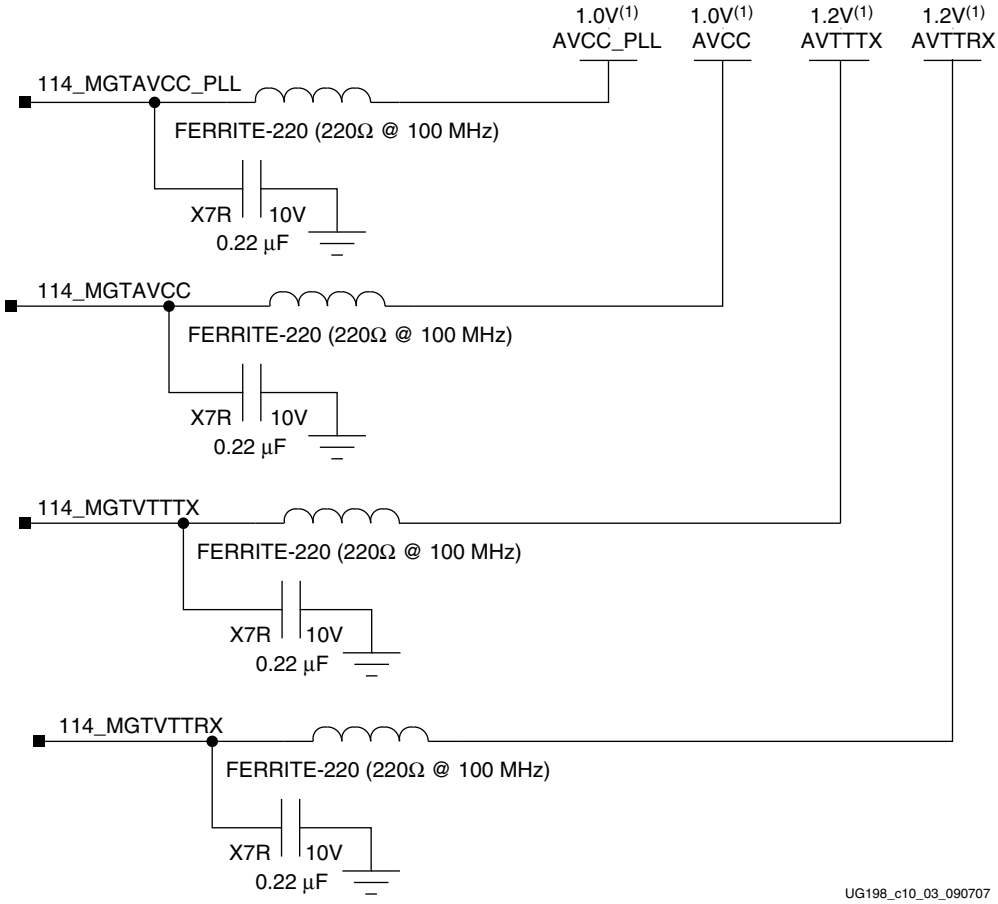
UG198_c10_12_110808

Figure 10-4: **Partially Used GTX_DUAL Column (Left Column in TXT Devices)**

The *Virtex-5 FPGA Data Sheet* provides the required exact voltage level and tolerance ranges of these analog supplies. Adequate filtering must be provided as illustrated in Figure 10-5. Xilinx recommends the use of separate (adjustable) voltage regulators for each supply circuit to facilitate the migration between LXT, SXT, FXT, and TXT platforms.

The voltage levels in Figure 10-5 are *nominal* values. Refer to the GTX transceiver portion of the *Virtex-5 FPGA Data Sheet* for the exact values based on operating conditions, especially voltage and temperature.

There is an important difference between GTP_DUAL and GTX_DUAL tiles: MGTAVCCPLL is nominal 1.2V on the GTP_DUAL tile and nominal 1.0V on the GTX_DUAL tile.

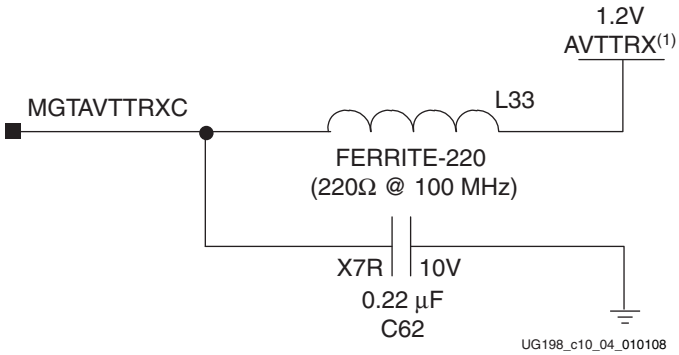


Notes:

1. The analog supplies shown in Figure 10-5 must be sourced directly from a dedicated regulator. Derived voltages from other supplies or resistor voltage dividers are not permitted.

Figure 10-5: **Power Filtering Schematic**

Figure 10-6 illustrates the power filter network for the MGTAVTTRXC pin.



Notes:

1. The analog supplies shown in Figure 10-6 must be sourced directly from a dedicated regulator. Derived voltages from other supplies or resistor voltage dividers are not permitted.

Figure 10-6: **Power Filter Network for MGTAVTTRXC**

The calculated calibration value of the resistor calibration circuit is shared between all GTX_DUAL primitives of a GTX_DUAL column (see Figure 10-7). For FXT devices, the resistor calibration circuit is located in the MGT112 GTX_DUAL tile. A TXT device has two resistor calibration circuits. One is located in the MGT111 GTX_DUAL tile and the other is located in the MGT112 GTX_DUAL tile.

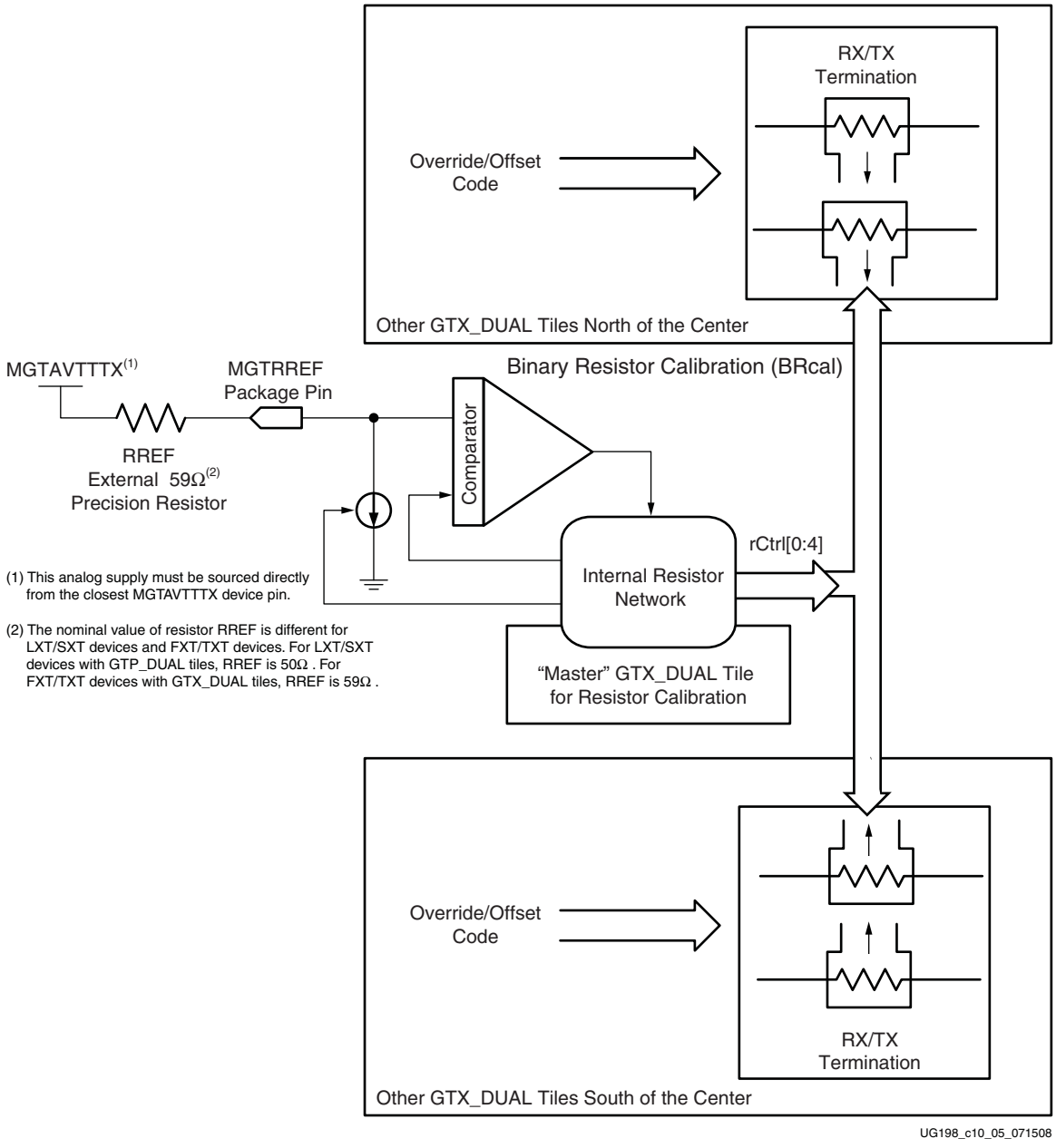


Figure 10-7: **Calibration Result Sharing Between All GTX_DUAL Tiles of a Column**

The resistor calibration is performed automatically one time during the configuration process. All analog supply voltages must be present and within the proper tolerance as specified in the *Virtex-5 FPGA Data Sheet*.

Note:

- FXT only:
  - The GTX_DUAL 112 tile contains the calibration circuit for the entire right GTX_DUAL column.
  - The MGTAVCC supply must be powered on all GTX_DUAL tiles between the GTX_DUAL 112 tile and any instantiated tile.
  - Filtering of MGTAVCC on unused GTX_DUAL tiles is *not* required.
- TXT only:
  - The GTX_DUAL 112 tile contains the calibration circuit for the entire right GTX_DUAL column.
  - The GTX_DUAL 111 tile contains the calibration circuit for the entire left GTX_DUAL column.
  - The MGTAVCC supply must be powered on all GTX_DUAL tiles in the right column between the GTX_DUAL 112 tile and any instantiated tile.
  - The MGTAVCC supply must be powered on all GTX_DUAL tiles in the left column between the GTX_DUAL 111 tile and any instantiated tile.
  - Filtering of MGTAVCC on unused GTX_DUAL tiles in the right column is not required.
  - Filtering of MGTAVCC on unused GTX_DUAL tiles in the left column is not required.

This value can be overwritten independently for each transceiver by using the TERMINATION_CTRL and TERMINATION_OVRD attributes. *This feature is intended for system evaluation purposes only.*

When overriding the control value based on the calibration to an external precision reference resistor, the process independence is lost. As a consequence, the termination value has a tolerance of ± 25% given by the resistor tolerance of the semiconductor process.

Table 10-3 shows the encoding of the TERMINATION_CTRL attribute.

Table 10-3: **TERMINATION_CTRL Attribute Encoding**

TERMINATION_CTRL[4:0]					Nominal Resistance [Ω]	+25% Resistance [Ω]	–25% Resistance [Ω]
0	0	0	0	0	103.0	128.8	77.3
0	0	0	0	1	97.8	122.3	73.4
0	0	0	1	0	93.2	116.4	69.9
0	0	0	1	1	88.9	111.1	66.7
0	0	1	0	0	85.0	106.3	63.8
0	0	1	0	1	81.5	101.9	61.1
0	0	1	1	0	78.2	97.8	58.7
0	0	1	1	1	75.2	94.0	56.4
0	1	0	0	0	72.4	90.5	54.3
0	1	0	0	1	69.8	87.3	52.4
0	1	0	1	0	67.4	84.2	50.5

Table 10-3: **TERMINATION_CTRL Attribute Encoding *(Cont'd)***

TERMINATION_CTRL[4:0]					Nominal Resistance [Ω]	+25% Resistance [Ω]	–25% Resistance [Ω]
0	1	0	1	1	65.1	81.4	48.9
0	1	1	0	0	63.0	78.8	47.3
0	1	1	0	1	61.1	76.3	45.8
0	1	1	1	0	59.2	74.0	44.4
0	1	1	1	1	57.5	71.8	43.1
1	0	0	0	0	55.8	69.8	41.9
1	0	0	0	1	54.3	67.8	40.7
1	0	0	1	0	52.8	66.0	39.6
1	0	0	1	1	51.4	64.3	38.6
1	0	1	0	0	50.1	62.6	37.6
1	0	1	0	1	48.8	61.0	36.6
1	0	1	1	0	47.6	59.5	35.7
1	0	1	1	1	46.5	58.1	34.9
1	1	0	0	0	45.4	56.8	34.1
1	1	0	0	1	44.4	55.5	33.3
1	1	0	1	0	43.4	54.2	32.5
1	1	0	1	1	42.5	53.1	31.8
1	1	1	0	0	41.5	51.9	31.2
1	1	1	0	1	40.7	50.9	30.5
1	1	1	1	0	39.9	49.8	29.9
1	1	1	1	1	39.1	48.8	29.3

REFCLK Guidelines

Overview

This section focuses on the selection of the reference clock source or oscillator. An oscillator is characterized by:

- Frequency range
- Output voltage swing
- Jitter (deterministic, random, peak-to-peak)
- Rise and fall times
- Supply voltage and current
- Noise specification
- Duty cycle and duty-cycle tolerance

- Frequency stability

These characteristics are selection criteria when choosing an oscillator for a GTX transceiver design.

Figure 10-8 illustrates the convention for the single-ended clock input voltage swing, peak-to-peak as used in the GTX transceiver portion of the *Virtex-5 FPGA Data Sheet*.

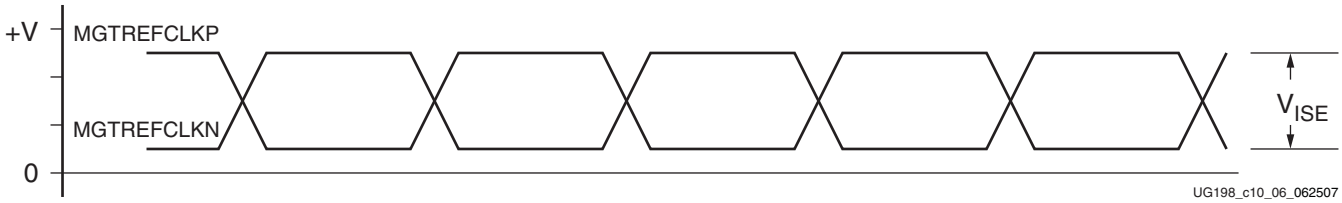


Figure 10-8: **Single-Ended Clock Input Voltage Swing, Peak-to-Peak**

Figure 10-9 illustrates the differential clock input voltage swing, peak-to-peak, which is defined as MGTREFCLKP – MGTREFCLKN.

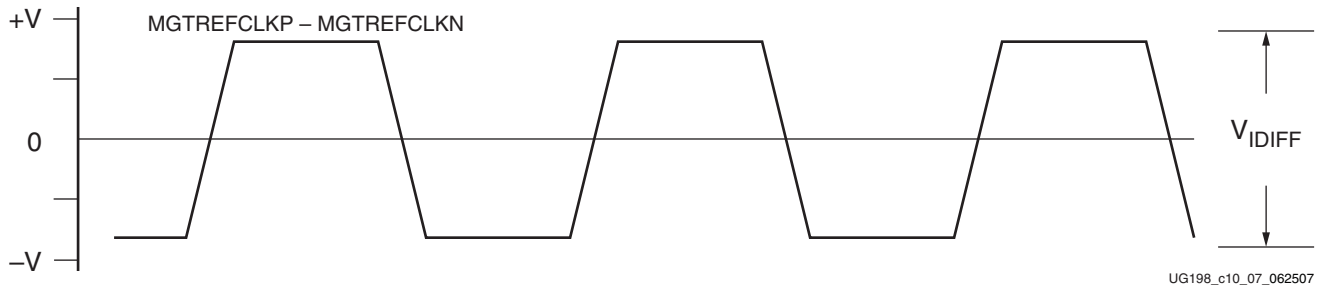


Figure 10-9: **Differential Clock Input Voltage Swing, Peak-to-Peak**

Figure 10-10 shows the rise and fall time convention of the reference clock.

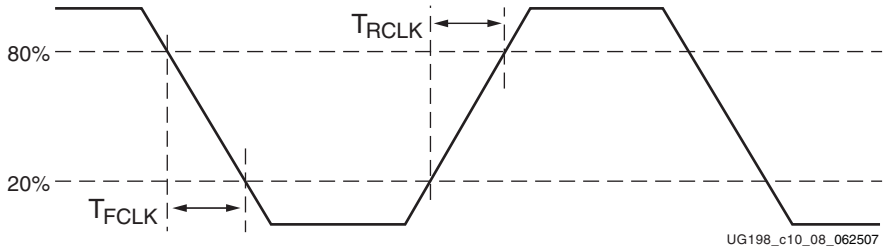
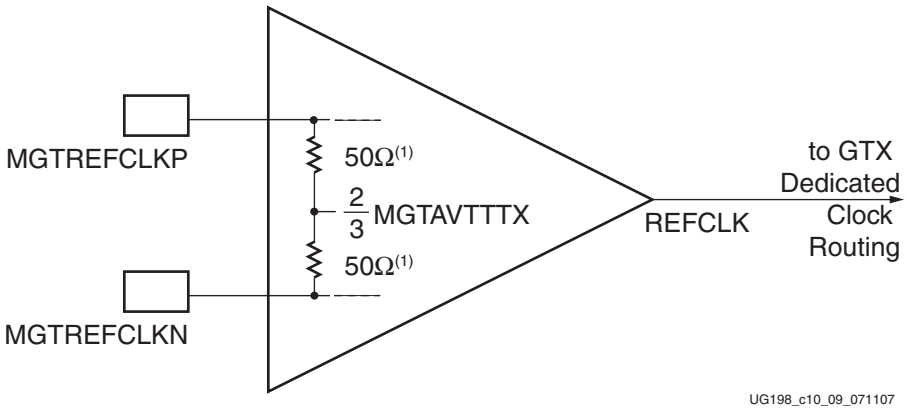


Figure 10-10: **Rise and Fall Time**

Figure 10-11 illustrates the internal details of the IBUFDS. The dedicated differential reference clock input pair MGTREFCLKP/MGTREFCLKN is internally terminated with 100Ωdifferential impedance. The common mode voltage of this differential reference clock input pair is 2/3 of MGTAVCCPLL, or nominal 0.8V. Refer to the *Virtex-5 FPGA Data Sheet* for exact specifications.



Notes:

1. Nominal values. Refer to the *Virtex-5 FPGA Data Sheet* for exact specifications.

Figure 10-11: **IBUFDS Details**

If the common mode voltage of the driving clock source is different from the common mode voltage of the differential reference clock input pair, then AC coupling capacitors are mandatory to prevent device degradation and/or other damage.

GTX Reference Clock Checklist

The following criteria must be met when choosing an oscillator for a design with GTX transceivers:

- Provide AC coupling between the oscillator output pins and the dedicated GTX_DUAL clock input pins.
- Ensure that the differential voltage swing of the reference clock is the range as specified in the *Virtex-5 FPGA Data Sheet* (the nominal range is 200 mV – 2000 mV, and the nominal typical value is 1200 mV).
- Meet or exceed the reference clock characteristics as specified in the *Virtex-5 FPGA Data Sheet*.
- Meet or exceed the reference clock characteristics as specified in the standard for which the GTX transceiver provides physical layer support.
- Fulfill the oscillator vendor's requirement regarding power supply, board layout, and noise specification.
- Provide a dedicated point-to-point connection between the oscillator and GTX_DUAL clock input pins.
- Keep impedance discontinuities on the differential transmission lines to a minimum (impedance discontinuities generate jitter).
- Any GTX_DUAL tile that sources a reference clock must be instantiated and REFCLKPWRDNB must be asserted High.

Description

Oscillator Selection

Selecting an oscillator and designing a clock distribution system requires a careful selection of components as well as a proper board layout to ensure that overall system requirements are met.

When designing a clocking system for a design with a GTX transceiver, the requirements of the implemented standard (Ethernet, OC-48, SDI, etc.) as well as the requirements given in the GTX Transceiver section of the *Virtex-5 FPGA Data Sheet* must be fulfilled. However, the requirements for the GTX transceiver reference clock as specified in the *Virtex-5 FPGA Data Sheet* must be met or exceeded. Under these conditions, the GTX transceiver was characterized as specified in the *Virtex-5 FPGA Data Sheet*.

The differential clock input of the GTX_DUAL primitive requires AC coupling capacitors between the oscillator output pins and the dedicated clock input pins of the Virtex-5 device.

Sourcing More Than One Differential Clock Input Pair from One Oscillator

If a clock needs to be shared between more than seven GTX_DUAL primitives of a Virtex-5 device, more than one differential clock input pair is required. Either an oscillator with multiple outputs or a single output oscillator and a multi-output clock buffer is required.

The connection between the dedicated clock input pin pair of a GTX_DUAL primitive and the outputs of the oscillator or buffer MUST be a point-to-point connection. Bifurcated transmission lines, “T-stubs”, branches, and daisy chaining are *not* permitted.

Switching between Two Different Reference Clocks

There are two ways to implement a design that needs to operate at two different clock rates (for example, in an HD-SDI video application):

1. Use the DRP to switch between two different clocks that are sourced from two different GTX_DUAL reference clock pins to the dedicated clock routing of the GTX_DUAL column.
2. Use an external clock multiplexer with one or multiple outputs.

The first solution is limited to up to four GTX_DUAL primitives and therefore to a maximum of eight GTX transceivers. In this configuration, one clock is sourced from the top and one clock is sourced from the bottom of four GTX_DUAL primitives, which are direct neighbors to each other.

The second solution can be used for up to 7 GTX_DUAL primitives and therefore to a maximum of 14 GTX transceivers, if an external multiplexer with one output is used. In this configuration, the GTX_DUAL primitive that sources the clock is located in the middle of seven GTX_DUAL primitives that are direct neighbors to each other. If a clock multiplexer with *n* outputs is used, this solution can be expanded up to *n* times 7 GTX_DUAL primitives and therefore up to *n* times 14 GTX transceivers.

AC Coupling

AC coupling of the oscillator reference clock output to the GTX_DUAL reference clock inputs serves multiple purposes:

- Blocking a DC current between the oscillator and the GTX_DUAL dedicated clock input pins (which reduces the power consumption of both parts as well)
- Common mode voltage independence
- The AC coupling capacitor forms a high-pass filter with the on-chip termination that attenuates a wander(1) of the reference clock

To minimize noise and power consumption, external AC coupling capacitors between the sourcing oscillator and the GTX_DUAL dedicated clock reference clock input pins are required.

Unused Reference Clock Inputs of GTX_DUAL Tiles for Clock Forwarding

It is recommended to connect the unused differential input pin clock pair to ground or leave it floating (both MGTREFCLKP and MGTREFCLKN).

Examples of Vendors and Devices

The alphabetical list of vendors and devices in Table 10-4 is intended to facilitate the search for parts.

Note: This table does *not* recommend, endorse, or verify the parts list!

1. A wander is low-frequency jitter.

Table 10-4: **Vendor and Device Examples**

Vendor	Website	Products	Examples	Remarks
Analog Devices	http://www.analog.com	Voltage regulators		
Epson	http://www.eea.epson.com	Oscillators	EG-2121CA (53.125 - 500 MHz oscillator)	
Integrated Systems	http://www.icst.com	Oscillators, buffers, PLLs	HiPer clock family	
Linear Technology	http://www.linear.com	Voltage regulators	LTC3026 (1.5A, 0.4 - 2.6V adjustable)	Provides SPICE models for some regulators, free simulation tools
Maxim	http://www.maxim-ic.com	Voltage regulators		
Micrel	http://www.micrel.com	Oscillators, clock buffers	SY100EP14U (1:5 driver with 2:1 MUX)	
Murata	http://www.murata.com	EMI suppression, ferrite beads, chip capacitors		Provides S parameters, design libraries for Signal integrity tools, and free simulation tools for their components (http://www.murata.com/designlib/index.html)
National Semiconductor	http://www.national.com	Voltage regulators	LP-3878-ADJ	
ON Semiconductor	http://www.onsemi.com	Voltage regulators	NCP5663 (3.0 A, 0.9Vmin, adjustable)	
Silicon Laboratories	http://www.silabs.com	Oscillators, clock multipliers, jitter attenuators	Si530 family (10 - 1400 MHz)	
TDK	http://www.component.tdk.com	Ferrite beads, capacitors		Provides component S parameters for simulation support, design libraries for Signal Integrity tools (http://www.component.tdk.com/tvcl_sparam.php)
Texas Instruments	http://www.ti.com	Voltage regulators, PLLs, buffer		
Vectron	http://www.vectron.com	Oscillators		
X2Y Attenuators, LLC	http://www.x2y.com	Low inductance capacitors		

Providing Power

Overview

This section focuses on the voltage regulators that directly source each dedicated filter network connected to one of the analog power supply pins of the GTX_DUAL primitive.

A voltage regulator is characterized by:

- Input voltage range
- Output voltage range
- Output voltage current
- Output voltage tolerance
- Output noise voltage
- Power supply ripple rejection (PSRR)

These characteristics are the selection criteria when choosing a voltage regulator for a design with GTX transceivers. The output voltage noise and the PSRR over frequency are often neglected but are very important selection criteria.

As a rule-of-thumb guideline, any substantial noise in the frequency range of 1 MHz and above on the power supply lines contributes to jitter. Depending on the frequency range and amplitude, this noise can degrade the overall system performance. The AVCC supply pins, which source the internal analog circuits of the transceivers, and AVCC_PLL, which sources the shared PMA PLL of the GTX_DUAL primitive, are especially sensitive to power supply noise.

When designing a complete Power Distribution System (PDS), the PSRR of the whole system and of each regulator is load-current and frequency dependent.

Description

Linear Regulator Selection Criteria

The selection criteria for the linear regulator are:

- Meet or exceed the characteristics specified in the *Virtex-5 FPGA Data Sheet*.
- The PSRR of the linear regulator must provide attenuation in a frequency range where the sourcing power supply or regulator emits noise. Use an adjustable regulator so that the voltage can be changed, if necessary.

The PSRR over frequency of a linear regulator (see Figure 10-12) is temperature and load-current dependent. Because the PSRR of the regulator in this example has a local minimum of rejection around 300 KHz, extra care must be taken if the sourcing power supply has spurs or high-amplitude noise in this frequency range.

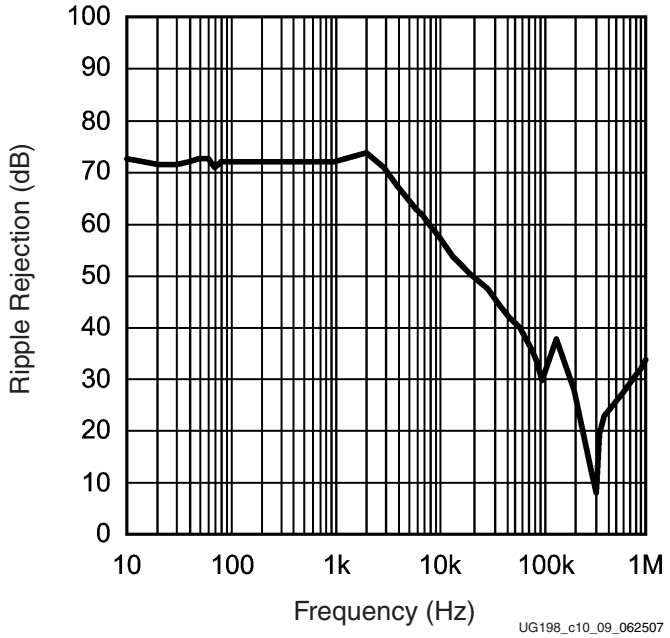


Figure 10-12: **PSRR over the Frequency of a Linear Regulator**

If the sourcing power supply cannot be changed or a different power supply cannot be selected, an additional filter network between the output of the sourcing power supply and the input of the linear regulator is required to prevent substantial noise from passing through the linear regulator at the minimum of attenuation. Because the capacitor on the output of the regulator is part of the regulator control loop, this capacitor not only impacts the regulator stability but the PSSR as well.

Regulator Design Guidelines

The selection criteria for the regulator design are:

- Do not operate the regulator with a V_{IN} that is just slightly over $V_{OUT} + V_{DROPOUT}$.
- Keep in mind that the dropout voltage is highly load dependent.
- Remember that the regulator stability and performance are only guaranteed. When using the correct decoupling capacitors (value, ESR, dielectric material) on input, output, and bypass pins of the regulator.
- Strictly follow the layout rules of the regulator data sheet.
- Do not operate the regulator at its maximum rated output current.
- If possible, place the regulator close to the filter network.
- Place the filter network as close as possible to the analog supply pin that it sources.
- Remember that the PSRR of the regulator is output load current and frequency dependent.

Ferrite Selection Guidelines

The selection criteria for the ferrite are:

- Choose a ferrite with a low DC resistance.
- Do not operate a ferrite at its maximum current rating(1).
- Choose a ferrite with a high impedance in a frequency range where you expect or measure the highest spurs or noise levels.

Figure 10-13 illustrates the impedance over frequency characteristics of different ferrites.

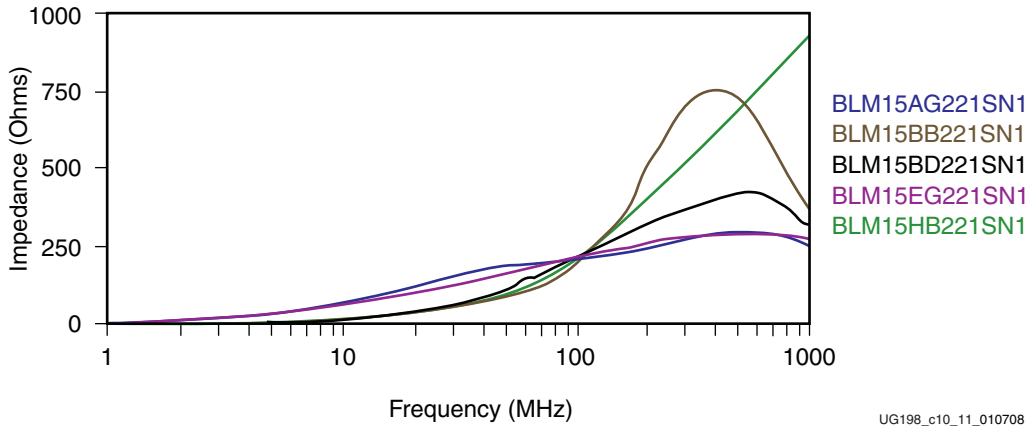


Figure 10-13: **Impedance over Frequency Characteristics of Different Ferrites**

The ferrites are from the same manufacturer, have the same footprint, and have a nominal impedance of 220Ω at 100 MHz. A low DC impedance is required to keep the influence of the load current change to the supply voltage change on the GTX_DUAL power pin to a minimum. Some manufacturers offer S parameters, proprietary-free simulation tools, or third-party simulation tool support via model libraries for their ferrites and capacitors. These allow the optimization of the filter circuit.

Depending on the spectrum of the noise still on the output pin of the linear regulator, the ferrite with the optimal characteristics must be selected. High-amplitude spurs in the frequency range of 1 MHz and above require special attention.

Capacitor Selection Guidelines

The selection criteria for the capacitor are:

- Low-inductance capacitor
- Dielectric material with a low-temperature coefficient
- Dielectric material with a low-frequency coefficient

1. If a ferrite is operating at its maximum current rating, the magnetic material of the ferrite body is in saturation, which impacts its ability to suppress high frequencies.

Filter Network Design Guidelines

The selection criteria for the filter network are:

- Place the filter network as close as possible to the device power pin.
- Ensure a low-inductance connection between the capacitor and the power pin.
- Simulate the filter circuit and optimize it, if possible.
- Isolate the analog supply plane between the filter and the FPGA pin. Make sure that no signals can capacitively or inductively couple into this supply.

Boundary-Scan Testing Guidelines

If Boundary-Scan[1] is to be used as part of the product verification, make sure that the analog supply voltage pin MGTAVCC of all GTX_DUAL tiles of the device are powered. The analog supply voltage pin MGTAVCC of all unused GTX_DUAL tiles must be connected to the same supply that supplies V_{CCINT}. V_{CCINT} is the power supply pin for the internal core logic.

Special Conditions: Unused GTX_DUAL Column

The following conditions apply to completely or partially unused GTX_DUAL columns:

- **Completely unused GTX_DUAL column**

  There are rare cases when an entire GTX_DUAL column of a device is never going to be used. In this case, the following pins or pin pairs of all GTX_DUAL tiles of this column must be connected as indicated in Table 10-5.

Table 10-5: **Unused GTX_DUAL Column Connections**

Pin or Pin Pair of the Unused Column	Connect To
MGTRXP/MGTRXN	GND
MGTTXP/MGTTXN	Floating, No Connection
MGTREFCLKP/MGTREFCLKN	Floating, No Connection
MGTAVTTTX	GND
MGTAVTTRX	GND
MGTAVTTRXC	Floating, No Connection
MGTRREF	Floating, No Connection
MGTAVCCPLL	GND
MGTAVCC	V_{CCINT}[1] or GND[2]

Notes:

1. If Boundary-Scan is part of the product verification, make sure that the analog supply voltage pin MGTAVCC of all GTX_DUAL tiles is connected to the same supply that supplies V_{CCINT}. V_{CCINT} is the power supply pin for the internal core logic.
2. If Boundary-Scan is *not* part of the product verification, connect MGTAVCC to GND.

1. Using any of the following JTAG operations (INTEST, EXTEST, SAMPLE, or PRELOAD), configuration via JTAG or ChipScope™ tool operations is NOT affected.

- **Partially used GTX_DUAL column**

  When unused GTX_DUAL tiles of a column are used to forward a clock, MGTAVCC must be powered but filtering is not required. For all other pins, the guidelines given in Table 10-6 must be followed.

  If the unused GTX_DUAL tiles of a column are not used for clock forwarding, the pin or pin pairs of the unused GTX_DUAL tiles must be connected as indicated in Table 10-6.

Table 10-6: **Partially Used GTX_DUAL Column Connections**

Pin or Pin Pair	Connect To
MGTRXP/MGTRXN	GND
MGTTXP/MGTTXN	Floating, No Connection
MGTREFCLKP/MGTREFCLKN	Floating, No Connection
MGTAVTTTX	GND
MGTAVTTRX	GND
MGTAVCCPLL	GND
MGTAVCC(1, 2, 3)	V_{CCINT}(1)

Notes:

1. Connect the analog supply voltage pin MGTAVCC of all unused GTX_DUAL tiles to the same supply that supplies V_{CCINT}. V_{CCINT} is the power supply pin for the internal core logic. For V_{CCINT}, follow the guidelines given in the *Virtex-5 FPGA PCB Designer's Guide*.
2. For FXT devices, if any GTX transceivers are used in a column, then the GTX_DUAL 112 tile that contains the calibration block *must* be connected as illustrated in Figure 10-2, page 252.
3. For TXT devices, if any GTX transceivers are used in a right column, then the GTX_DUAL 112 tile that contains the calibration block *must* be connected as illustrated in Figure 10-2, page 252. If any GTX transceivers are used in a left column, then the GTX_DUAL 111 tile that contains the calibration block *must* be connected as illustrated in Figure 10-2, page 252.

SelectIO to GTX Crosstalk Guidelines

Because a GTX transceiver's performance can degrade in an environment flooded with SelectIO™ activity, it is important to have guidelines for SelectIO usage that minimize the impact on GTX transceiver performance.

Although the Virtex-5 FPGA package exhibits little package-related crosstalk issues, the pinout of the device might lead to customer designs becoming susceptible to PCB-via related crosstalk issues. The near proximity of SelectIO signals (aggressor) to GTX transceiver analog supplies (victim) results in their PCB via structures being placed in close proximity as well. This ball adjacency and resulting via adjacency creates a via-coupling region between the SelectIO signals and the GTX transceiver analog supplies that is not filtered by on-board power supply filtering. The amount of crosstalk voltage induced on the victim circuit by the aggressor circuit is equal to the rate of change of current in the aggressor times the mutual inductance shared between the two circuits. For an in-depth discussion on via crosstalk and calculations of mutual inductance for various via configurations, refer to *High-Speed Signal Propagation: Advanced Black Magic* by Howard Johnson and Martin Graham [Ref 7]. The sensitivity of the GTX transceiver analog supplies to coupled noise from the PCB results in a degradation of GTX transceiver performance. The MGTAVCC and MGTAVCCPLL supplies are most sensitive to coupled noise. The MGTAVTTTX pins are also sensitive to coupled noise, because they power the reference clock circuits.

To minimize the impact on GTX transceiver performance, these BGA adjacency guidelines must be followed:

Note: The BGA adjacency guidelines must be followed as well for every package device combination not listed in Table 10-7 through Table 10-13.

- TXT devices only: No SelectIO pins are 1.0 mm (horizontal or vertical) or 1.4 mm (diagonal) away from GTX transceiver supply pins in TXT devices as implied by the package design.
- Avoid utilizing SelectIO nets 1.0 mm (horizontal or vertical) or 1.4 mm (diagonal) away from GTX transceiver analog power supply pins. Ground these SelectIO locations in the PCB, and set the SelectIO output to the highest drive and a forced-Low setting. If these SelectIO outputs must be used, use them either in differential signaling applications or for static control/status signals with low speed and low drive.
- Avoid using a large number of SelectIO signals in I/O banks near GTX transceivers. See Table 10-7 for specific aggressive I/O bank to GTX transceiver pairing.

Table 10-7: **Aggressive I/O Banks**

GTX_DUAL	FF665	FF1136	FF1738
MGT112	12	12	12
MGT114	12	18	18
MGT116	12, 16	12	12
MGT118	12, 18	18	18, 26
MGT120		12, 20	20
MGT122		22	18, 26
MGT124		20	
MGT126		22	26

Table 10-7: **Aggressive I/O Banks *(Cont'd)***

GTX_DUAL	FF665	FF1136	FF1738
MGT128			20, 24
MGT130			34
MGT132			24
MGT134			34

- If these SelectIO pins must be used for higher drive/higher speed applications, apply power to the GTX transceiver analog supplies with a plane or wide buses a few layers below the top of the board. Using a blind via to the GTX transceiver analog supplies is better than using a through via. Shield these supply planes or buses with ground planes above and below.
- If a through via to supply the GTX transceiver analog supply pins must be used, use a layer closest to the FPGA for routing signals to these vias. Route SelectIO nets in the uppermost layer available after GTX transceiver high-speed signal and GTX transceiver analog supply routing is implemented.
- If supplying GTX transceiver power from the bottom of the board, route these SelectIO nets in the highest available routing layer.

The absolute worst-case scenario is to supply GTX transceiver analog supplies from the bottom of the board and have all adjacent SelectIO outputs running at high drive and high speed and routed to lower routing layers. For more information, refer to "BGA Escape Example," page 304 for information on escaping of SelectIO nets adjacent to GTX transceiver analog supply pins.

The SelectIO signals that have the largest impact on GTX transceiver performance are those whose solder balls are adjacent to GTX transceiver analog supply solder balls (BGA adjacency). Table 10-8 through Table 10-12 provide the GTX transceiver user with pin-specific guidance recommendations to optimize GTX transceiver performance in the presence of SelectIO switching. Specifically, the tables identify those pins which are either 1.0 mm or 1.4 mm away from an GTX transceiver analog supply pin and REFCLK pins. Table 10-9 and Table 10-11 specifically identify signals that can couple into either the MGTREFCLK pin pair or the MGTAVTTTX supply pins, resulting in noise coupling onto the reference clock signal. If the MGTREFCLK input pin pair is used or a reference clock that was forwarded from a GTX_DUAL neighbor tile is used, it is important to follow the guidelines given for the identified pins. The identified pins can only be used if the reference clock is not routed through this neighbor GTX_DUAL tile.

Additionally, it is important to properly shield the traces or planes connecting the analog supply vias to the supply filter network of the analog supplies from SelectIO signals escaping from the BGA field. This is best done by placing GND planes above and below the layer containing the analog traces or planes.

Note: For device/package combination not listed in Table 10-7 through Table 10-13, the BGA adjacency guidelines *must* be followed.

Table 10-8: **SelectIO Net Adjacent to Analog Supplies (FF665 Packages)**

GTX_DUAL Tile	1 mm	1.4 mm
MGT116	E5	D5, F5, G4
MGT112	L5	K5

Table 10-8: **SelectIO Net Adjacent to Analog Supplies (FF665 Packages) *(Cont'd)***

GTX_DUAL Tile	1 mm	1.4 mm
MGT114	U5	T5
MGT118	AD4	AB5, AD5

Table 10-9: **SelectIO Net Adjacent to MGTCLK (FF665 Packages)**

GTX_DUAL Tile	1 mm	1.4 mm
116_REFCLK	D5, A3, B4, G4	E5, A4, H4
112_REFCLK	K5, H4	J5, L5, G4
114_REFCLK	T5, W4	R5, U5, Y4
118_REFCLK	AB5, Y4, AF3	AA5, W4, AF4

Table 10-10: **SelectIO Net Adjacent to Analog Supplies (FF1136 Packages)**

GTX_DUAL Tile	1 mm	1.4 mm
MGT124	E7	E8, E6
MGT120	F5	E6, G5
MGT116	J5	H5, L4
MGT112		P5
MGT114	AA5	AB5, AC4
MGT118	AG5	AF5, AH5
MGT122	AK6	AH5, AJ6
MGT126	AK8	AK7, AK9, AL10

Table 10-11: **SelectIO Net Adjacent to MGTCLK (FF1136 Packages)**

GTX_DUAL Tile	1 mm	1.4 mm
124_REFCLK	E8, D10	E7, E9, D11
120_REFCLK	F5	
116_REFCLK	H5, L4	G5, J5
112_REFCLK	P5	N5, L4
114_REFCLK	AC4	AA5, AD4
118_REFCLK	AF5, AD4	AG5, AC4
122_REFCLK		AK6
126_REFCLK	AK7, AL10, AM11	AK6, AK8, AL11

Table 10-12: **SelectIO Net Adjacent to Analog Supplies (FF1738 Packages)**

GTX_DUAL Tile	1 mm	1.4 mm
MGT132(1)	E15	E14, D13
MGT128(2)	E9	E10, E8, D7
MGT124	E5	
MGT120		F5, H5
MGT116	N5	P5, R4
MGT112	W5	V5
MGT114	AE5	AD5, AF5, AG4
MGT118	AL5	AK5, AN4
MGT122		AT5, AV5
MGT126	AV5	AV6
MGT130(2)	AV10	AV9, AV11, AW12
MGT134(1)	AV16	AV15, AW18

Notes:

1. The GTX_DUAL tile is only available on XC5VFX200T devices.
2. The GTX_DUAL tile is available on XC5VFX200T and XC5VFX130T devices.

Table 10-13: **SelectIO Net Adjacent to MGTCLK (FF1738 Packages)**

GTX_DUAL Tile	1 mm	1.4 mm
132_REFCLK(1)	D18, D13	E17, E15, D12
128_REFCLK(2)	E10, D12, D7	E9, D13
124_REFCLK		D7
120_REFCLK	F5	E5, K4
116_REFCLK	K4, R4	L5, N5, T4
112_REFCLK	V5, T4	W5, R4
114_REFCLK	AD5, AG4	AC5, AE5, AH4
118_REFCLK	AK5, AH4, AN4	AJ5, AL5, AG4
122_REFCLK	AT5	AR5, AN4
126_REFCLK		AV5, AW7
130_REFCLK(2)	AV9, AW7, AW12	AV8, AV10, AW13
134_REFCLK(1)	AV15, AW13, AW18, AY19	AV14, AV16, AW12, BA19

Notes:

1. The GTX_DUAL tile is only available on XC5VFX200T devices.
2. The GTX_DUAL tile is available on XC5VFX200T and XC5VFX130T devices.

Section 2: Board Level Design

This section describes general design guidelines when designing systems and boards where the interconnect exhibits transmission line behavior. This situation occurs when signals have rise and/or fall times smaller than 2.5 times the flight time from one end of the interconnection to the other end. These guidelines also apply to all designs with high-speed transceivers.

These guidelines have been used to create boards that have successfully operated at serial transmission rates in excess of 10 Gb/s. Although designing for 10 Gb/s operation can seem excessive when the application calls for slower speeds, it is better to design knowing that constraints can be relaxed for slower speeds if needed. Other interfaces with challenging signal integrity demands, such as high-speed memory interfaces, also benefit from the approach used for 10 Gb/s design.

This section includes the following chapters:

Chapter 11

Design Constraints Overview

Figure 11-1 shows a typical physical interconnect topology between two transceivers. Any physical link connecting two point-to-point high-speed serial transceivers is defined as a channel. A channel begins at the die solder bumps of the transmitter and ends at the die solder bumps of the receiver.

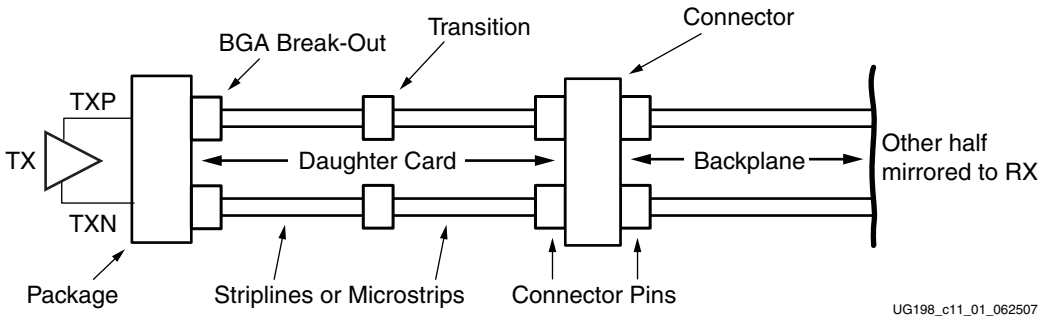


Figure 11-1: **Two Connected Transceivers Forming a Link**

In Figure 11-1, the channel consists of the FPGA package, transmission lines, connectors, and transitions.

Transitions are defined as any section along a multi-gigabit channel where the signal must go from a transmission line to a three-dimensional structure or vice-versa. Vias, connectors, and coupling capacitors are examples of these structures.

While a more comprehensive list of transitions is discussed in Chapter 13, "Design of Transitions," some common transitions are:

- Ball grid array (BGA) to PCB microstrip
- Microstrip to stripline vias
- DC blocking capacitors
- Connectors
- Bends and turns in a trace

For optimal performance, the following must be minimized in a given channel:

- Signal attenuation due to losses in the transmission medium
- Impedance transitions at each transition can lead to reflections, ringing, and other artifacts in the signal

At gigahertz signaling speeds, losses due to the transmission medium become significant due to greater signal attenuation with increasing frequency. The attenuation of the high-frequency components slows down the edge and reduces voltage swing, resulting in eye

closure. See Chapter 12, "PCB Materials and Traces," for guidelines on stackup and characteristic trace impedance design.

While there are several transitions in each channel, most or all transitions can be designed for the least negative impact on performance. Because standard non-optimized PCB structures tend to be capacitive at gigahertz frequencies, a convenient figure of merit for transitions is excess capacitance. An ideal transition does not have excess capacitance or inductance.

For each typical transition, techniques to limit excess capacitance and excess inductance are provided to build a robust channel on the first pass. These design rules, techniques, and examples are presented in Chapter 13, "Design of Transitions." The goal is to ensure tight control of any impedance variation along the entire channel.

In general, minimizing the number of components and layer changes in the high-speed serial PCB traces brings about the most benefit. Careful design of the traces, vias, and even connector pads is required for gigahertz speeds.

Powering Transceivers

Supplying noise-free power to the transceivers is a critical factor in achieving low link error rates and reliable system-level operation. This section discusses general principles that should be applied to transceiver power supply designs.

Linear regulators are required for directly sourcing the power supplies. Although switching regulators are an appealing option in applications requiring high-power efficiency, they are unsuitable for directly sourcing transceivers because they can introduce switching noise, even if care is taken to eliminate the noise.

Power Distribution Architecture

In most system-level designs, multiple voltage levels are required for powering devices on the board. The supplies for devices that use leading-edge process technologies are typically low voltages in the region of 1V. At these low voltages, it is important that noise levels on these supplies be kept at a minimum.

For this reason, Xilinx recommends the use of point-of-load (POL) power distribution techniques. The POL approach places the power supplies right at the device being powered, hence the name. Background information on this approach can found in http://www.xilinx.com/publications/xcellonline/xcell_57/xc_pdf/p105-107_57-bellinix.pdf. This approach can be extended to the use of separate linear regulators for groups of transceivers. This has several advantages including:

- Increased system reliability by eliminating a single point of failure.
- The ability to independently adjust supply voltages for transceiver groups to support the requirements of different link interfaces.
- Reduction of power requirements from each regulator, which reduces the physical size, simplifies board layout, and eliminates board hot spots.

Figure 11-2 shows how POL power distribution can be applied to powering transceivers.

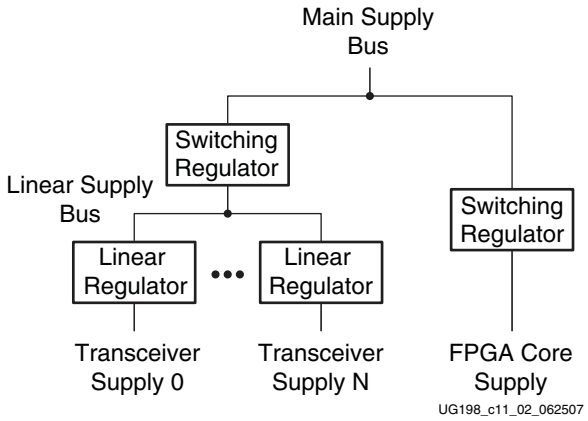


Figure 11-2: **POL Power Distribution Architecture**

In most cases, the linear regulator is driven by a switching power supply. Care must be taken to ensure that ripple and other switching noise artifacts are filtered out by the regulator and/or filtering circuitry. Extra care must be taken when using low drop out (LDO) linear regulators because limited voltage margin at the input reduces the ability of the regulator to filter out noise components.

Regulator Selection

The designer must follow the voltage regulator selection guidelines for the specific receiver. A majority of transceiver performance issues are traced to power-supply noise issues.

Filtering

The power distribution system guidelines for the specific receiver must be strictly followed to achieve the specified performance.

Reference Clock

Clock Sources

A high-quality crystal oscillator is essential for good performance. The oscillator manufacturer's power supply design guide must be followed.

When choosing alternate clock sources, the alternate oscillators must meet or exceed the specifications as required by the transceiver data sheet.

Depending on the application and its performance goals, it is possible to stray from the clock source specifications. In that case, the specified performance of the transceiver is not guaranteed.

Clock Traces

Because performance of the transceiver is directly related to the quality of its reference clock, every effort must be made to ensure the signal integrity of the clock traces from oscillator to FPGA. Apply the same techniques for 10 Gb/s trace design in Chapter 13, "Design of Transitions," to these clock traces.

If more than one clock source is driving a single reference clock differential pair, a high-speed switch should be used. When multiple clock sources are bused together on a single connection, the signal integrity of the clock is not optimal due to the presence of stubs. Using a high-speed switch makes every clock path point-to-point with one driver and one receiver.

One example is when an unpopulated oscillator shares the same trace to the clock input pin as another clock source. The trace segment from the pad to the junction is a stub. Any signal travelling down the segment is reflected due to the different impedance presented by the open end at the pad.

For applications where a single reference clock source must drive multiple inputs, high-speed clock buffers should be used for clock distribution to eliminate stubs and reduce reflections on the clock lines.

Coupling

DC Coupling

DC coupling can be used when the common mode ranges of the interconnecting devices are the same. Any discrepancy in the common mode not only takes margin away from the differential voltage swing but can also damage the device.

AC Coupling

AC coupling isolates the common modes of the two devices and is the preferred configuration in hot-plug applications. The capacitor prevents any DC current from flowing between connected devices.

Some transceivers have built-in DC blocking capacitors with programmable bypass. In most cases, an external DC blocking capacitor is needed to provide adequate system-level performance.

External Capacitor Value Selection

If an external DC blocking capacitor is needed, it is important to select an appropriate value. The selection of a capacitor value is a trade-off between the following contradictory criteria:

- Encoding schemes with longer run lengths require larger capacitance values to reduce pattern dependent jitter (PDJ).
- Higher data rates require smaller capacitor values to reduce edge rate degradation.

PDJ is not an issue in protocols using line codings that preserve DC balance. DC balance is the property where the average number of 1s and 0s transmitted are equal. 8B/10B is an example of a line-coding scheme that provides DC balance. When 8B/10B encoding is used, 0.01 µF capacitors in a 0402 (EIA) package are suitable for external AC coupling at 3.125 Gb/s.

Line coding schemes that do not guarantee DC balance require more careful analysis. An example includes SONET, which uses scrambling to ensure adequate symbol transitions but does not provide DC balance. The remainder of this section provides the theory needed to select a blocking capacitor value appropriate for the application.

Several protocols, including the PCI Express and SATA protocols, specify ranges for blocking capacitors in applications. This is done not only to simplify compliance but also to ensure that the link presence detection features included in these specifications work correctly.

Table 11-1: **PCI Express and SATA Blocking Capacitor Values**

Specification	Required Range
PCI Express Base Specification, Revision 1.1	75 to 200 nF
Serial ATA Specification, Revision 2.5	0 to 12 nF

The blocking capacitor when combined with the termination resistance acts as a high-pass filter. Figure 11-3 shows a simplified circuit model for the link. The internal blocking capacitors are not shown in this model because they do not play a significant role in blocking DC currents from the external link as described in "RX Termination and Equalization," page 158.

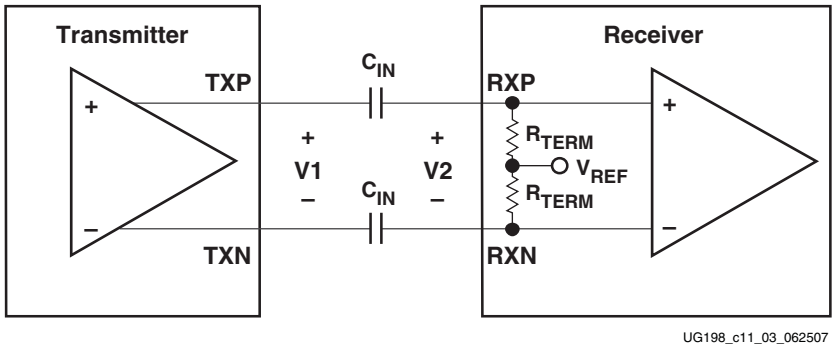


Figure 11-3: **Simplified Link Circuit Model**

Problems occur when the line is held in the on state for an extended period of time. When this happens, charge accumulates on the blocking capacitors and a DC offset is added or subtracted from V2. This offset results in what is known as baseline wander (see Figure 11-4). In Figure 11-4, VTH is the threshold voltage.

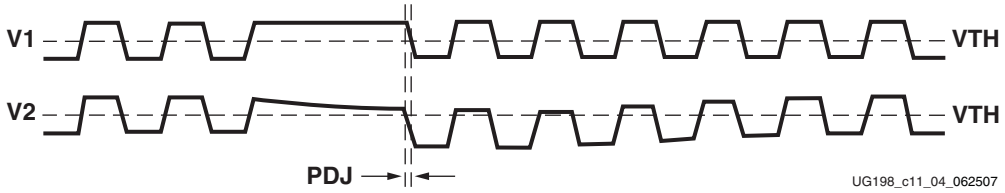


Figure 11-4: **Baseline Wander and PDJ**

The effect of baseline wander is to shift the signal with respect to the threshold points in the receiver. This in turn skews the time at which transitions within the signal are recognized. PDJ is the result of this skew. Figure 11-5 shows an overlay of V1 and V2 in the region of Figure 11-4 where the jitter is greatest and shows several key parameters.

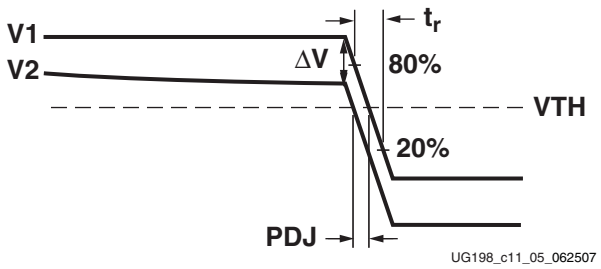


Figure 11-5: **PDJ Detail**

To calculate the blocking capacitor value, several factors must be known:

- t_r: The rise time of the signal
- T: The bit period
- N_{CID}: The maximum number of consecutive identical digits (CIDs)
- PDJ: The amount of pattern dependent jitter that can be tolerated by the system

From Figure 11-5 it can be seen that PDJ can be estimated by:

$$PDJ = \frac{\Delta V}{slope} \quad \textit{Equation 11-1}$$

The voltage drop can be calculated using Equation 11-2:

$$\Delta V = 0.5 V_{PP}(1 - e^{-t/\tau}) \quad \textit{Equation 11-2}$$

where:

- τ is the RC time constant (C is the AC coupling capacitor, R = 2 x R_{TERM}).
- t is the total discharge time, which is equal to N_{CID}T.

The slope is defined by Equation 11-3:

$$slope = V_{PP} \times \frac{0.6}{t_r} \quad \textit{Equation 11-3}$$

Substituting Equation 11-2 and Equation 11-3 into Equation 11-1 and solving for C gives:

$$C = \frac{-T \times N_{CID}}{2 \times R_{TERM} \times \ln\left(1 - \frac{1.2PDJ}{t_r}\right)} \quad \textit{Equation 11-4}$$

To demonstrate the use of Equation 11-4, calculate the blocking capacitor value needed for a serial link running at 3.125 Gb/s using 8B/10B line coding. This example uses the following assumptions:

- Bit period (T) = 3.200 x 10^{-10} (3.125 Gb/s)
- Signal rise time (t_r) = 6.400 x 10-11 (0.2 UI)
- Pattern Dependent Jitter (PDJ) = 3.200 x 10-12 (0.01 UI)
- Consecutive Identical Digits (N_{CID}) = 5 (guaranteed by 8B/10B)
- Termination Resistance (R_{TERM}) = 50Ω

Plugging these values into Equation 11-4 gives:

$$C = \frac{-(3.20 \times 10^{-10}) \times 5}{2 \times 50 \times \ln\left(1 - \frac{1.2(3.20 \times 10^{-12})}{6.40 \times 10^{-11}}\right)} = 0.26nF \qquad \text{Equation 11-5}$$

Equation 11-4 is only valid for cases where the problem data pattern consists of a single sequence of consecutive identical digits. More complex pathological cases can occur in protocols using block coding schemes and scrambling.

One example of such a case occurs in the serial digital interface (SDI) used to transmit digital video. Figure 11-6 illustrates this pattern, which is called an equalization test pattern.

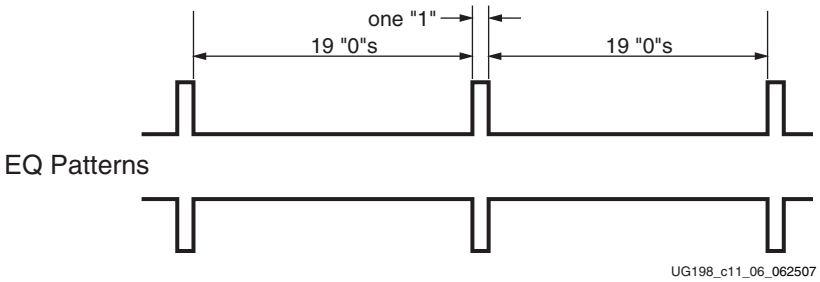


Figure 11-6: **SDI EQ Pathological Waveform**

The waveform is 20 bits long and consists of a single "1" bit followed by 19 "0" bits. The complementary waveform, a single "0" bit followed by 19 "1" bits is also equally likely. The EQ waveform can repeat across the entire length of the active portion of a video line. For standard definition video (SD-SDI), this waveform can consist of up to 720 consecutive repetitions of the 20-bit pattern. For high definition video (HD-SDI), this pattern can repeat up to 1920 consecutive times.

For this case, the blocking capacitor value cannot be calculated using Equation 11-4 because an N_{CID} value of 19 does not reflect the total time that a DC imbalance is being applied on the line. To properly analyze charge accumulation on blocking capacitors for this case, more extensive analysis beyond the scope of this document is required.

SelectIO to Serial Transceiver Crosstalk Guidelines

The breakout of SelectIO signals adjacent to transceiver analog supply pins is also important. As noted in "SelectIO to GTX Crosstalk Guidelines," page 269, these SelectIO requirements, if not taken into account, can have an effect on transceiver performance. This impact occurs when SelectIO solder balls are adjacent to transceiver analog supply or REFCLK solder balls and their corresponding PCB vias are adjacent as well, creating both package and board coupling mechanisms. The solder balls, which are part of the package, offer some coupling, and the adjacent PCB vias offer two to four times more coupling. Simulation suggests that the amount of coupling due to adjacent PCB vias is affected by which layer the SelectIO escape is located and on how the analog supplies are delivered to the transceivers. Simulation predicts that coupling can be reduced by using the upper PCB routing layers to route SelectIO signals and/or by using a higher layer to distribute transceiver analog power supplies. When a design dictates that SelectIO signals with BGA adjacency to transceiver analog supply pins are to be used for high-drive/high-speed applications, the following guidelines apply:

- Apply power to the transceiver analog supplies with a plane or wide buses a few layers below the top of the board. Using a blind via to the transceiver analog supplies is better than using a through via. Shield the supply plane with GND planes above and below.
- If a through via to supply the transceiver analog supply pins must be used, use an upper layer to supply analog power to these vias. Route SelectIO nets in the uppermost layer available after transceiver signal and transceiver analog supply routing is implemented.
- If supplying transceiver power from the bottom of the board, route these SelectIO nets in the highest available routing layer.
- Do not use SelectIO blocks adjacent to REFCLK pins because the REFCLK pins are a reference clock source to the transceivers either in the same tile or in other tiles.

Figure 11-7 depicts the coupling regions for BGA adjacent SelectIO signals. The primary coupling mechanism is mutual inductive coupling, which occurs in the area between the active signal path and the power via. The secondary coupling mechanism, also shown in Figure 11-7, is capacitive. The primary coupling mechanism is much larger, and the recommendations are designed to minimize this effect.

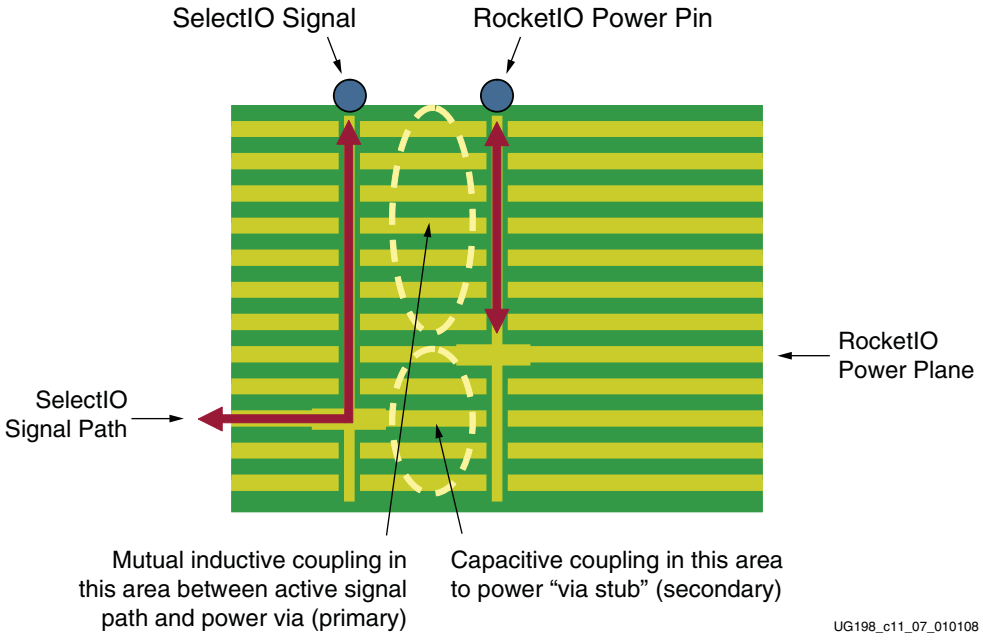


Figure 11-7: **Via Structures for BGA Adjacent SelectIO Signals**

Chapter 12

PCB Materials and Traces

The choice of PCB materials and cable type can have a large impact on system performance. Although any transmission medium is lossy at gigahertz frequencies, this chapter provides some guidelines on managing signal attenuation so as to obtain optimal performance for a given application.

How Fast is Fast?

Signal edges contain frequency components called harmonics. Each harmonic is a multiple of the signal frequency and has significant amplitude up to a frequency determined by Equation 12-1:

$$f \approx 0.35 / T \qquad \text{Equation 12-1}$$

Where:

f = Frequency in GHz
T = The smaller of signal rise (T_r) or fall (T_f) time in ns

Because dielectric losses in a PCB are frequency dependent, a bandwidth of concern must be determined to find the total loss the PCB. Frequencies must start at the operation frequency and extend to the frequency in Equation 12-1. For example, a 10 Gb/s signal with a 10 ps rise time has a bandwidth from 10 GHz to 35 GHz.

Dielectric Losses

The amount of signal energy lost into the dielectric is a function of the materials characteristics. Some parameters used to describe the material include relative permittivity ε_r (also known as the dielectric constant) and loss tangent. Skin effect is also a contributor to energy loss at line speeds in the gigahertz range.

Relative Permittivity

Relative permittivity is a measure of the effect of the dielectric on the capacitance of a conductor. The higher the relative permittivity, the slower a signal travels on a trace and the lower the impedance of a given trace geometry. A lower ε_r is almost always preferred.

Although the relative permittivity varies with frequency in all materials, FR4 exhibits wide variations in ε_r with frequency. Because ε_r affects impedance directly, FR4 traces can have a spread of impedance values with increasing frequency. While this spread can be less significant at 3.125 Gb/s, it can be a concern at 10 Gb/s operation.

Loss Tangent

Loss tangent is a measure of how much electromagnetic energy is lost to the dielectric as it propagates down a transmission line. A lower loss tangent allows more energy to reach its destination with less signal attenuation.

As frequency increases, the magnitude of energy loss increases as well, causing the highest frequency harmonics in the signal edge to suffer the most attenuation. This appears as a degradation in the rise and fall times.

Skin Effect and Resistive Losses

Xilinx, in partnership with Dr. Howard Johnson, has developed a two-part DVD tutorial on signal integrity techniques and loss budgeting for transceivers [Ref 6].

The skin effect is the tendency for current to flow preferentially near the outer surface of a conductor. This is mainly due to the larger magnetic fields in higher frequency signals pushing current flow in the perpendicular direction towards the perimeter of the conductor.

As current density near the surface increases, the effective cross-sectional area through which current flows decreases. Resistance increases because the effective cross-sectional area of the conductor is now smaller. Because this skin effect is more pronounced as frequency increases, resistive losses increase with signaling rates.

Resistive losses have a similar effect on the signal as loss tangent. Rise and fall times increase due to the decreased amplitude of the higher harmonics, with the highest frequency harmonics being most affected. In the case of 10 Gb/s signals, even the fundamental frequency can be attenuated to some degree when using FR4.

For example, an 8 mil wide trace at 1 MHz has a resistance on the order of 0.06Ω/inch, while the same trace at 10 Gb/s has a resistance of just over 1Ω/inch. Given a 10 inch trace and 1.6V voltage swing, a voltage drop of 160 mV occurs from resistive losses of the fundamental frequency, not including the losses in the harmonics and dielectric loss.

Choosing the Substrate Material

The goal in material selection is to optimize both performance and cost for a particular application.

FR4, the most common substrate material, provides good performance with careful system design. For long trace lengths or high signaling rates, a more expensive substrate material with lower dielectric loss must be used.

Substrates, such as Nelco, have lower dielectric loss and exhibit significantly less attenuation in the gigahertz range, thus increasing the maximum bandwidth of PCBs. At 3.125 Gb/s, the advantages of Nelco over FR4 are added voltage swing margin and longer trace lengths. At 10 Gb/s, Nelco is necessary unless high-speed traces are kept very short.

The choice of substrate material depends on the total length of the high-speed trace and also the signaling rate.

What-if analysis can be done in HSPICE simulation to evaluate various substrate materials. By varying the dielectric constant, loss tangent, and other parameters of the PCB substrate material. The impact on eye quality can be simulated to justify the use of higher cost materials. The impact of other parameters such as copper thickness can also be explored.

Traces

Trace Geometry

For any trace, its characteristic impedance is dependent on its stackup geometry as well as the trace geometry. In the case of differential traces, the inductive and capacitive coupling between the tightly coupled pair also determines the characteristic impedance of the traces.

The impedance of a trace is determined by its inductive and capacitive coupling to nearby conductors. For example, these conductors can be planes, vias, pads, connectors, and other traces, including the other closely coupled trace in a differential pair. The substrate properties, conductor properties, flux linkage area, and distance to a nearby conductor determine the amount of coupling and hence, the contribution to the final impedance.

2D field solvers are necessary in resolving these complex interactions and contribute to the calculation of the final impedance of the trace. They are also a useful tool to verify existing trace geometries.

A common misconception is that two 50Ω single-ended traces can be routed side-by-side to give a pair with 100Ω differential impedance. While this approximation might be true if the traces are loosely coupled, routing differential traces in a loosely coupled fashion does not maximize the noise immunity of differential mode signaling.

Tightly coupled differential pairs are required for all high-speed transceiver traces because they are more sensitive to noise than slower signals. As a general rule of thumb, tight coupling within a differential pair is achieved by spacing them no more than four trace widths apart.

Wider traces create a larger cross-sectional area for current to flow and reduce resistive losses. Use the widest traces that space constraints allow. Because trace width tolerances are expressed in absolute terms, a wider trace also minimizes the percentage variation of the manufactured trace, resulting in tighter impedance control along the length of the transmission line.

Striplines are preferred over microstrips because the reference planes on both sides of the trace provide radiation shielding. Microstrips are shielded on only one side (by the reference plane) because they run on the top-most or bottom-most layers, leaving the other side exposed to the environment.

For best results, the use of a 2D field solver is recommended for verification.

Trace Characteristic Impedance Design

Because the transceivers use differential signaling, the most useful trace configurations are differential edge-coupled center stripline and differential microstrip. While some backplanes use the differential broadside-coupled stripline configuration, it is not recommended for 10 Gb/s operation, because the P and N vias are asymmetrical and introduce common-mode non-idealities.

With few exceptions, 50Ωcharacteristic impedance (Z_0) is used for transmission lines in the channel. In general, when the width/spacing (W/S) ratio is greater than 0.4 (8 mil wide traces with 20 mil separation), coupling between the P and N signals affects the trace impedance. In this case, the differential traces must be designed to have an odd mode impedance (Z_{0O}) of 50Ω, resulting in a differential impedance (Z_{DIFF}) of 100Ω, because $Z_{DIFF} = 2 \times Z_{0O}$.

The same W/S ratio also must be less than 0.8, otherwise strong coupling between the traces requires narrower, lossier traces for a Z_{0O} of 50Ω To clarify, with Z_{0O} at 50Ω, an even mode impedance (Z_{0E}) of 60Ω or below is desired.

Figure 12-1 through Figure 12-4 show example cross sections of differential structures.

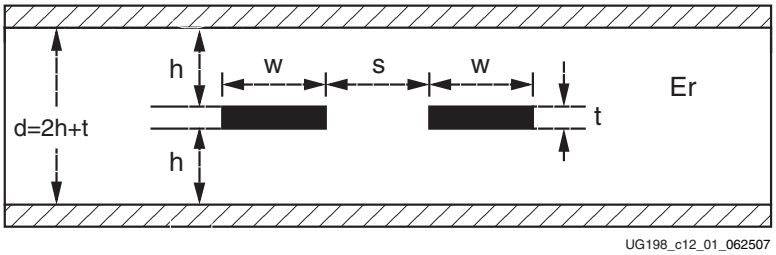


Figure 12-1: **Differential Edge-Coupled Centered Stripline**

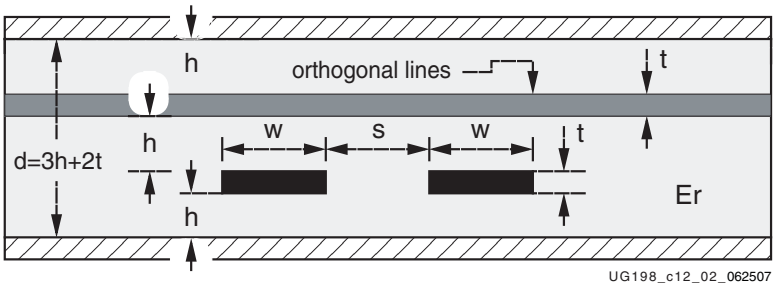


Figure 12-2: **Differential Edge-Coupled Offset Stripline**

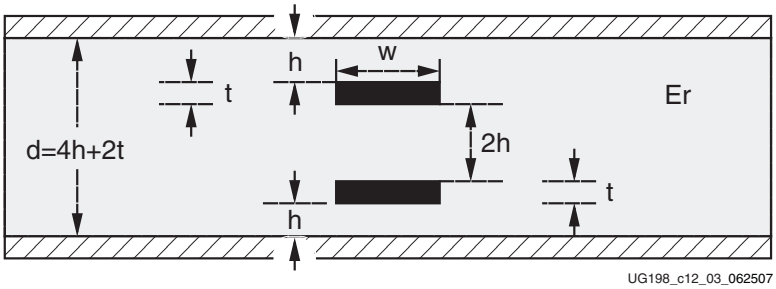


Figure 12-3: **Centered Broadside-Coupled Stripline**

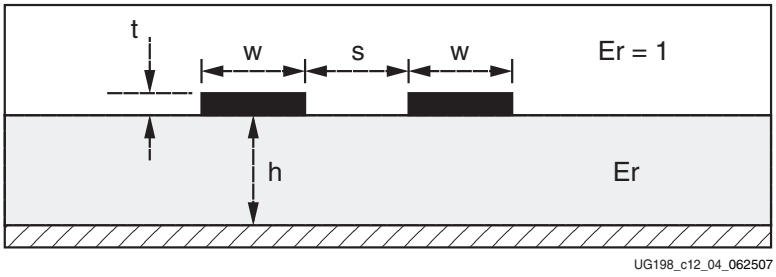


Figure 12-4: **Differential Microstrip**

A good PCB manufacturer understands controlled impedance and allows fine adjustments for line widths to produce a Z_{0O} of 50Ω The PCB manufacturer also provides the parameters necessary for the specific PCB layout. Some parameters can be calculated or simulated from the guideline outlined in the example. Although ±10% tolerance on Z_{0O} is typical and can provide adequate performance, the additional cost of a tighter tolerance results in better channel performance.

Trace Routing

High-speed serial differential traces are routed with the highest priority to ensure that the optimal path is available to these critical traces. This reduces the need for bends and vias and minimizes the potential for impedance transitions. Traces must be kept straight, short, and with as few layer changes as possible. The impact of vias is discussed in "Differential Vias," page 295.

Routing of high-speed traces must be avoided near other traces or other potential sources of noise. Traces on neighboring signal planes should run perpendicular to minimize crosstalk.

Striplines are to be used whenever possible, as are the uppermost and lowermost stripline layers to minimize via stubs. When the stackup is being planned, these layers must be placed as close to the top and bottom layers whenever possible.

Design constraints might require microstrips for the BGA exit path or from via to connector launch or SMT pads. In such cases, the microstrip trace must be kept as short as possible.

Right-angled bends must not be used. Mitered 45-degree bends are to be used instead. At a 90-degree bend, the effective width of the trace changes, causing an impedance discontinuity due to the capacitive coupling of the additional conductor area to the reference plane.

The two traces of a differential pair must be length-matched to eliminate skew. Skew creates mismatches in the common mode and reduces the differential voltage swing as a result.

Plane Splits

Ground planes should be used as reference planes for signals, as opposed to noisier power planes. Each reference plane should be contiguous for the length of the trace, because routing over plane splits creates an impedance discontinuity. In this case, the impedance of the trace changes because its coupling to the reference plane is changed abruptly at the plane split.

Return Currents

Routing over plane splits also creates issues with the return current. High-speed signals travel near the surface of the trace due to the skin effect mentioned in "Dielectric Losses." Meanwhile, the return current also travels near the surface of the tightly coupled reference plane.

Because of the tight coupling, the return current has the tendency to travel close to the original signal-carrying trace. At the plane split, the return current can no longer follow the same path parallel to the trace, but must instead find an alternative route.

A plane split causes a suboptimal current return path and increases the current loop area, thereby increasing the inductance of the trace at the plane split, changing the impedance of the trace.

Simulating Lossy Transmission Lines

Due to the different modeling implementations used by various circuit simulators (frequency-domain versus time-domain techniques), it is important to check that the models accurately reflect actual losses. One method is to compare the models against known published configurations.

Cable

Cables are controlled-impedance transmission lines due to the constant physical dimensions of conductor and dielectric along the length of the cable. The highest quality cable shows little variation in these dimensions and also has a wide bandwidth with low loss at high frequencies.

Connectors

The connectors attached to cables should exhibit low parasitic inductance and capacitance for high bandwidth operation.

Skew Between Conductors

When selecting a cable, look for a specification of the skew between the conductors in a cable. If the conductors are not length matched, the skew appears in the common mode and directly reduces the eye height.

Chapter 13

Design of Transitions

Each transition in the channel must be designed to minimize any negative impact on the link performance. This section addresses the interface at either ends of a transmission line.

Transmission lines have defined and controlled characteristic impedance along their length by definition. However, the three-dimensional structures that they interface do not have easily defined or constant impedance along the signal path. Software tools such as 3D field solvers are necessary for computing the impedance that a 10 Gb/s signal sees as it passes through these structures, while 2D field solvers are sufficient for computing transmission line characteristic impedance.

PCB designers can use the analyses and examples in this section to greatly accelerate the design of such a channel. Cases not covered in this section might need further simulation and board iterations.

Excess Capacitance and Inductance

Most differential transitions are overly capacitive. The P and N paths couple to each other, increasing capacitance. Many transitions have a frequency response identical to that of a lumped capacitor over a wide frequency band.

By design, adding inductance cancels this excess capacitance in many cases except when impacted by density concerns and physical limitations. While techniques such as blind vias, solder balls on a larger pitch, and very small via pads reduce capacitance, they are not always feasible in a design.

Time domain reflectometry (TDR) techniques, either through simulation or measurement, allow the designer to identify excess capacitance or excess inductance in a transition.

Time Domain Reflectometry

To make TDR measurements, a step input is applied to the interconnect. The location and magnitude of the excess capacitance or inductance that the voltage step experiences as it traverses the interconnect can be determined through observing the reflected signal.

A shunt capacitance (see Figure 13-1) causes a momentary dip in the impedance, while a series inductance (see Figure 13-2) causes an impedance discontinuity in the opposite direction. Td is assumed to be the propagation delay through the first transmission line segment on the left. The reflected wave due to the impedance discontinuity takes 2 * Td to return to the TDR port. If the signal propagation speed through the transmission line is known, the location of the excess capacitance or inductance along the channel can be calculated.

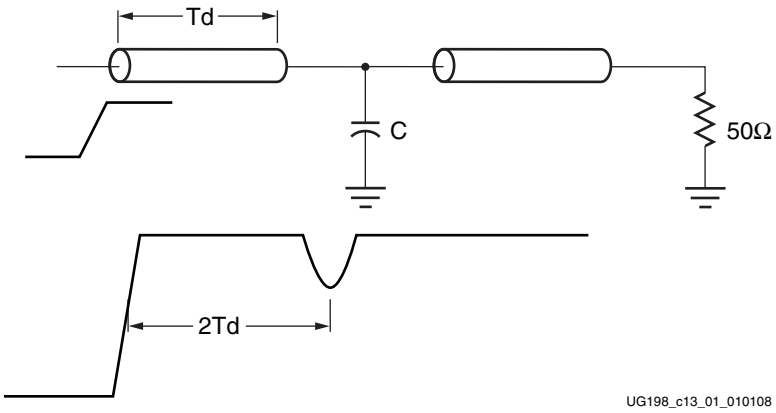


Figure 13-1: **TDR Signature of Shunt Capacitance**

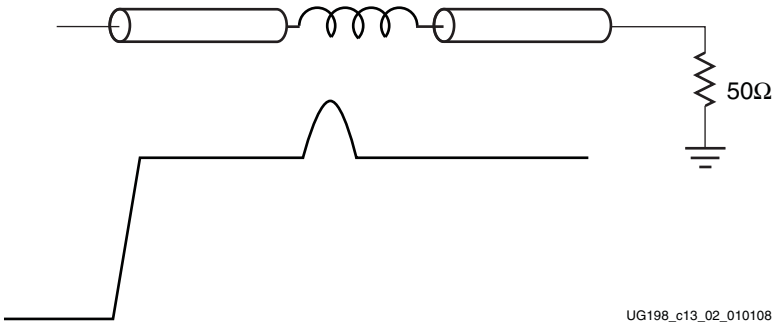


Figure 13-2: **TDR Signature of Series Inductance**

The magnitude of this excess capacitance (C) or inductance (L) can also be extracted from the TDR waveform by integrating the normalized area of the transition's TDR response. The respective equations for capacitance and inductance are:

$$C = -\frac{2}{Z_0}\int_{t1}^{t2}\frac{V_{tdr}(t) - V_{step}}{V_{step}}\,dt \qquad \textit{Equation 13-1}$$

$$L = 2Z_0\int_{t1}^{t2}\frac{V_{tdr}(t) - V_{step}}{V_{step}}\,dt \qquad \textit{Equation 13-2}$$

Figure 13-3 shows the integration of the normalized TDR area.

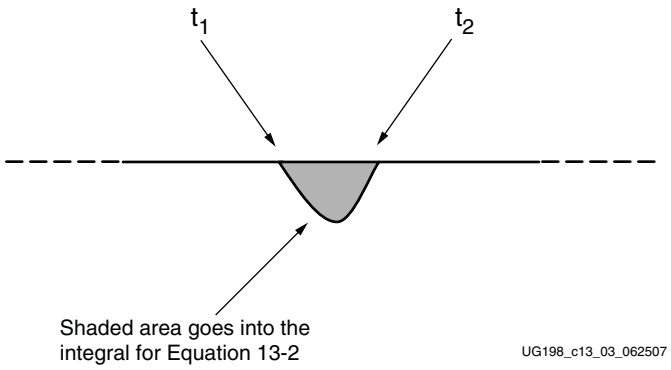


Figure 13-3: **Integration of Normalized TDR Area**

The results using these equations are not sensitive to rise time variation and are valid for simulated TDR measurements provided that the leading and trailing transmission lines are very close to 50Ω However, for actual measurements, accuracy is very dependent on Z_0.

BGA Package

The transceiver signal paths within the BGA package are optimized using a 3D full-wave solver. Package traces are designed to be 50Ω high-speed transmission lines, while solder ball and bump regions are tuned to 50Ω

Flip-chip package transitions are effectively invisible to 10 Gb/s signals. The longest package paths have some insertion loss, less than 1 dB worst-case.

SMT Pads

For applications that require AC coupling between transmitter and receiver, SMT pads are introduced in the channel to allow coupling capacitors to be mounted. Standard SMT pads have excess capacitance due to plate capacitance to a nearby reference plane. In the Figure 13-4 example, a 5 mil trace with a Z_0 of 50Ω transitions to an 0402 SMT pad that is 28 mils wide, all over 3 mils of FR4.

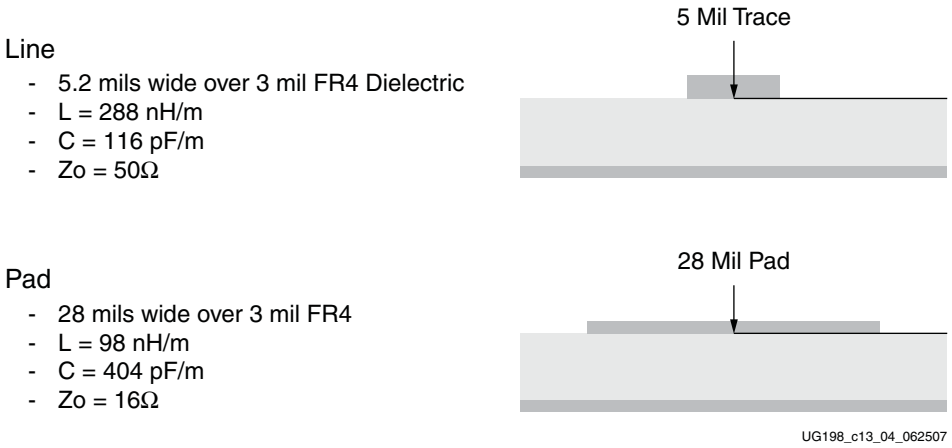


Figure 13-4: **2D Field-Solver Analysis of 5 Mil Trace and 28 Mil Pad**

Using a 2D field solver on these dimensions yields a Z_0 of 50Ω for the 5 mil trace. The Z_0 for the 0402 pad is 16Ω because the pad has too much capacitance and too little inductance, resulting in an impedance of less than 50Ω. Performance of this transition can be optimized in one of two ways.

The first method makes the trace the same width as the pad and moves the ground plane deeper into the stackup to maintain the Z_0 of the transition at 50Ω. This method does not require any special analysis, but there might be some error due to the fringing capacitance of the SMT capacitor body. Trace density is limited because traces are now 28 mils wide.

The second method, shown in Figure 13-5, clears the ground plane underneath the pad, which removes much of the excess capacitance caused by the plate capacitance between the pad and the ground plane. This technique allows for greater trace density than the first method, but requires 3D field-solver analysis or measurement along with several board iterations to get the desired performance.

Figure 13-5: **Transition Optimization**

The 2D field-solver example shows that close to 50Ω can be achieved if the ground plane under the pad footprint is cleared out. A 3D field solver is then used to verify this result to a greater degree of accuracy.

Figure 13-6 shows the ground plane cleared away exactly as it was for the 2D simulation. Using frequency domain analysis within HFSS, there is a 20 dB (10x) improvement in return loss using this technique.

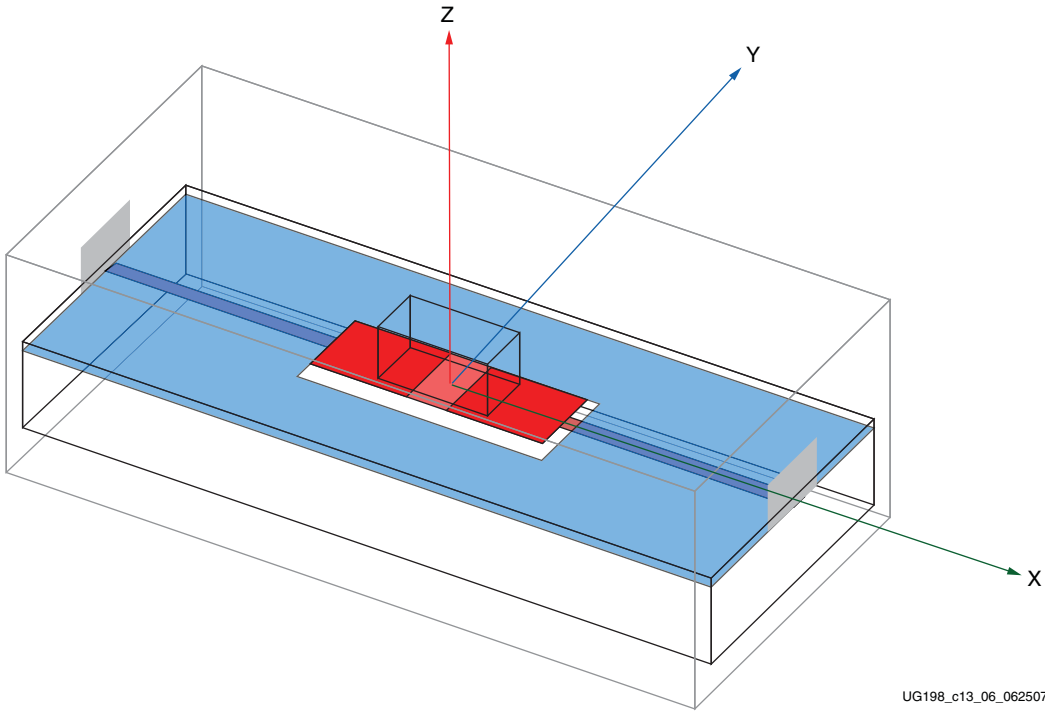


Figure 13-6: **Ansoft HFSS Model of Pad Clear-Out**

Figure 13-7 shows the return loss comparison between 0402 pad structures with linear scale.

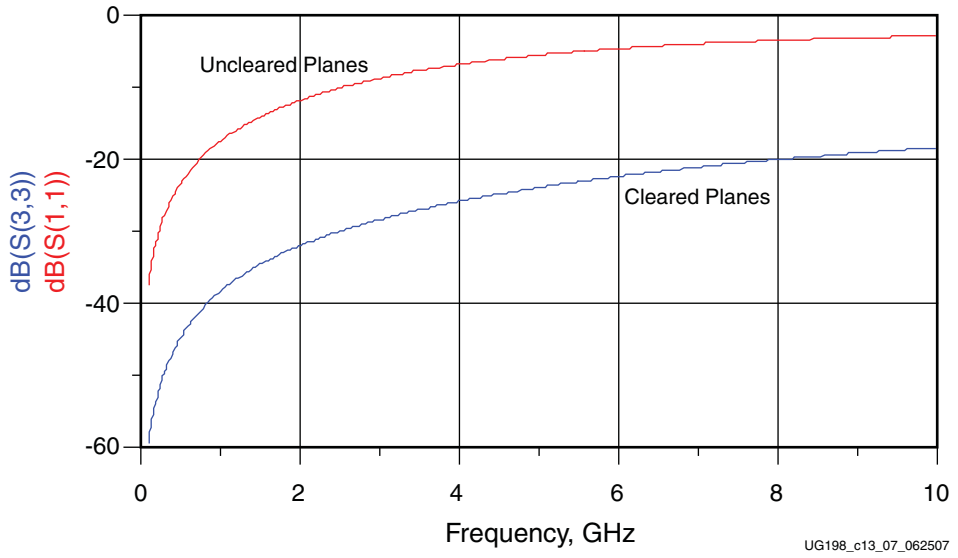


Figure 13-7: **Return Loss Comparison Between 0402 Pad Structures**

The approximately –40 dB/decade slope in Figure 13-8 shows good fit to the frequency response of a lumped capacitor.

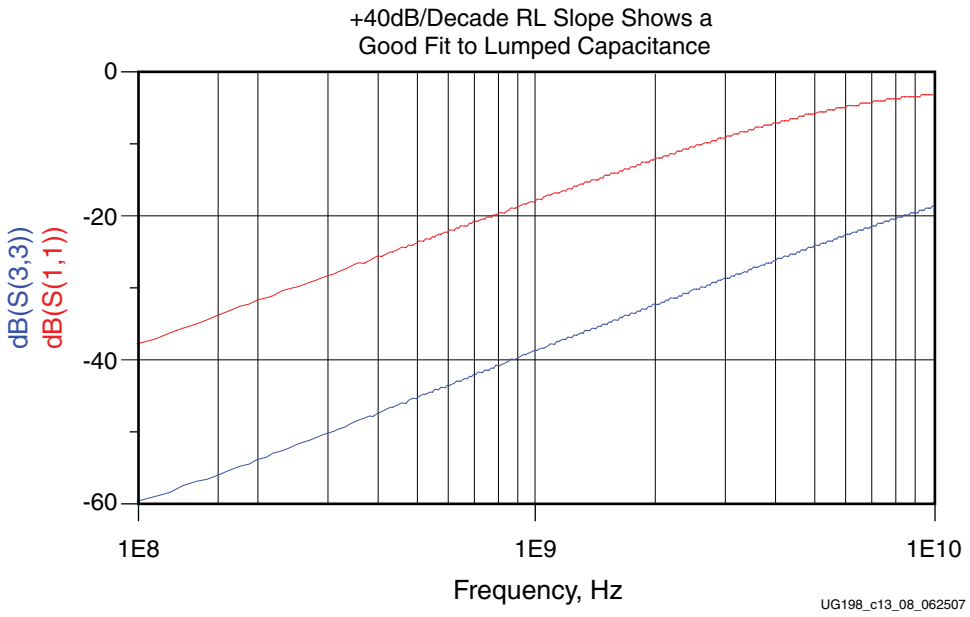


Figure 13-8: **Return Loss Comparison Between 0402 Pad Structures on Log (Frequency) Scale**

Next, using simulated measurements on the same transition modeled in HFSS, the time-domain performance of this transition can be measured by doing a TDR on the S-parameter results from the earlier frequency domain analysis.

In Figure 13-9 and Figure 13-10, the red curve with the large capacitive dip corresponds to the SMT pad without the ground plane cleared from underneath. The blue curve shows that clearing out the ground plane removes much of the excess capacitance. This improvement can be quantified using Equation 13-1 and Equation 13-2.

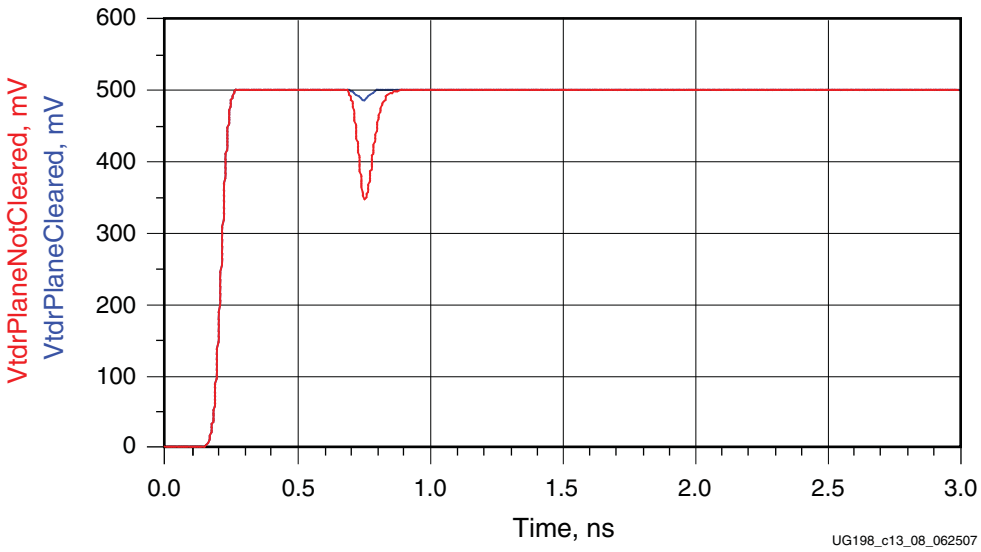


Figure 13-9: **TDR Results Comparing 0402 Pad Structures**

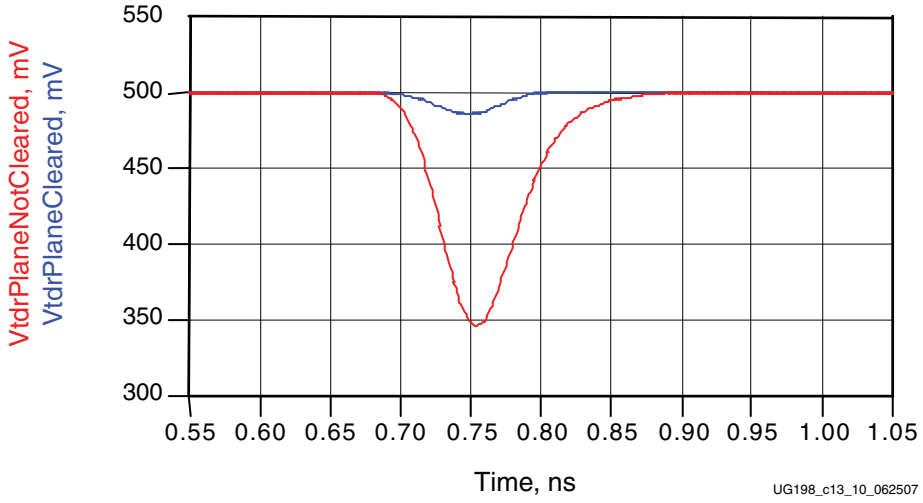


Figure 13-10: **TDR Results Comparing 0402 Pad Structures**

As shown from Figure 13-11 and Figure 13-12, clearing the ground plane under SMT pads yields a significant improvement in the performance of an SMT pad transition. Excess capacitance is reduced by 15x, and return loss is improved by 20 dB.

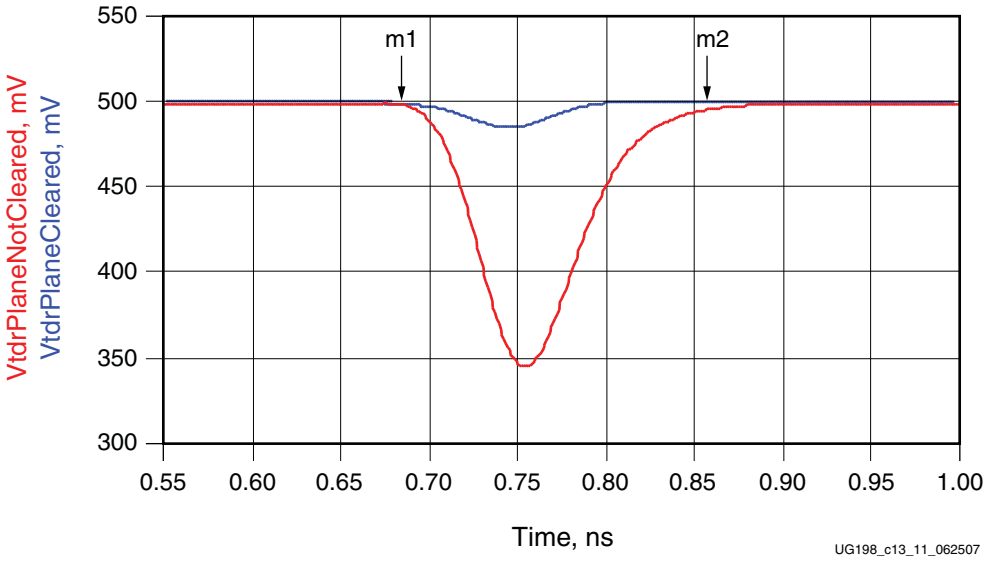


Figure 13-11: **840 fF Excess Capacitance with Ground Plane Intact**

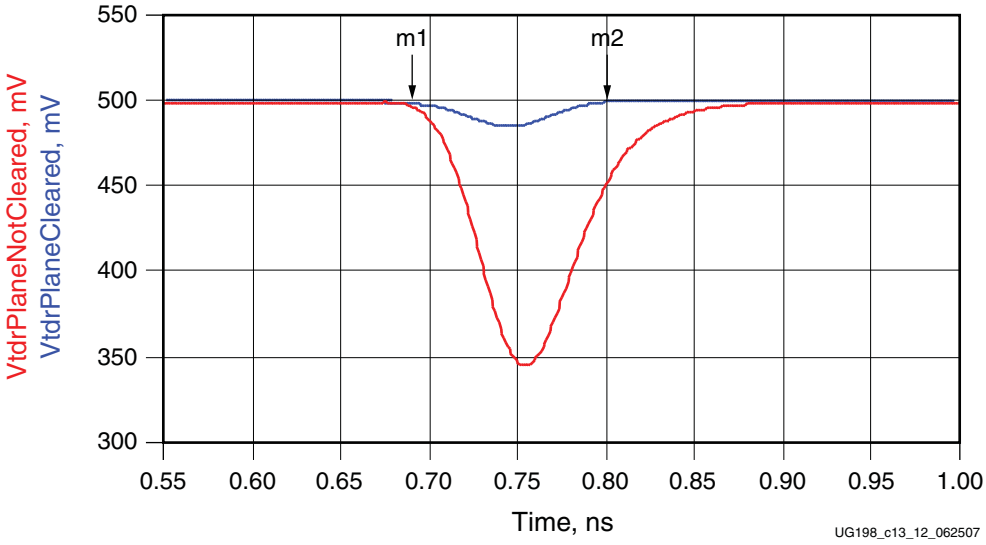


Figure 13-12: **57 fF Excess Capacitance with Ground Plane Intact**

Differential Vias

The most common transition is the differential via where the signal pair must transition from an upper stripline layer or top microstrip to a lower stripline layer or bottom microstrip.

Figure 13-13 shows a Ground-Signal-Signal-Ground (GSSG) type differential via. Ground vias are connected to each ground plane in the stackup, while signal layers only contain pads for the entry and exit layers.

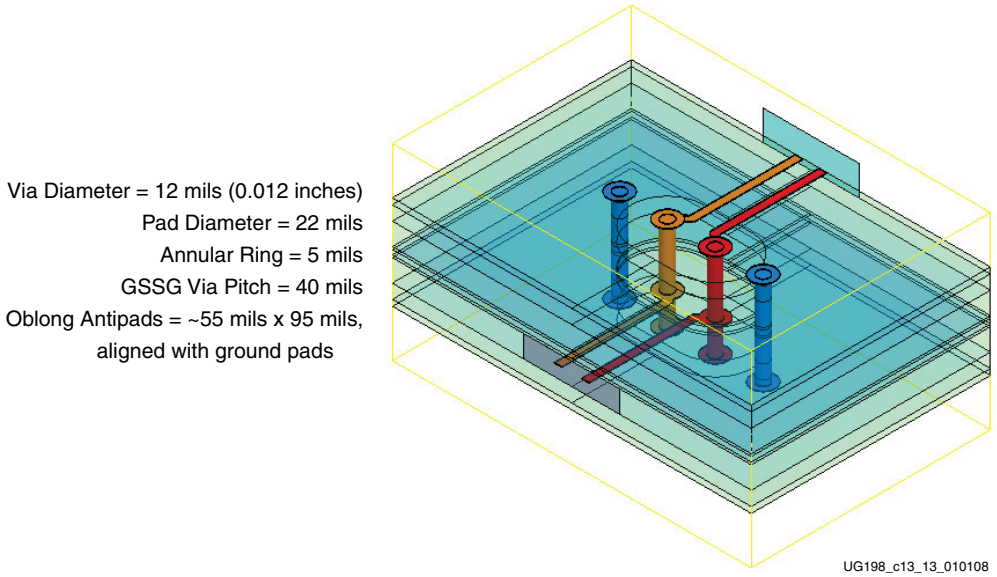


Figure 13-13: **Differential Via Design Example**

A key advantage of a GSSG via is that it allows for the signal's return current to flow in the ground via near the corresponding signal via, reducing excess inductance. The signal path is also symmetrical between the P and N halves of the differential signal, which is critical in controlling common-mode artifacts due to P/N imbalance.

The larger oblong antipads reduce excess fringing capacitance between the via body and the surrounding planes edges. Unused pads are also removed.

A good starting point is to use the dimensions shown in Figure 13-13 as an example differential via design for an 80 mil board. To accommodate density constraints or the lack thereof, the dimensions can be scaled accordingly to preserve the ratios of each dimension relative to the others. Such scaling preserves the impedance performance of the differential via while allowing variation in overall size to better suit specific applications. These final dimensions are limited by manufacturability and density constraints.

While the via length can be varied by a small amount to suit boards that are thicker or thinner than the 80 mil example, changing the ratio of the via length relative to other dimensions affects the via's impedance. For this and other configurations of differential vias, it is best to simulate a model using 3D field-solver tools to ensure that performance targets are met.

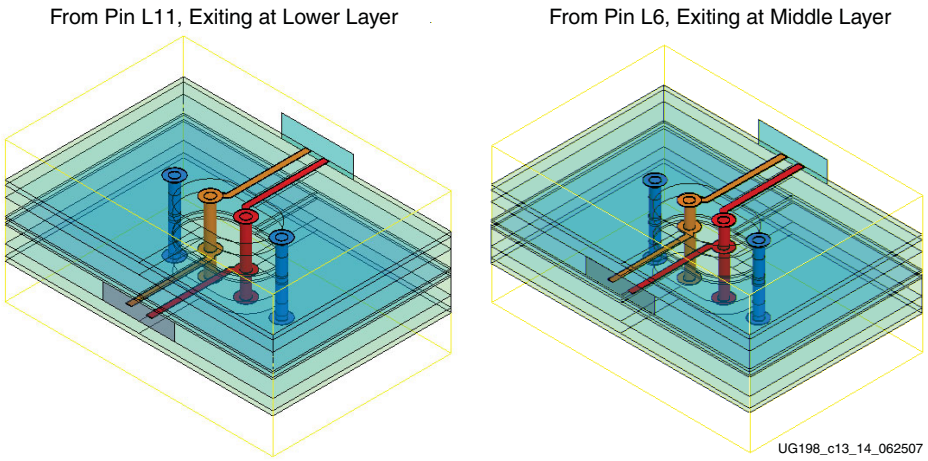


Figure 13-14: **Differential GSSG Via in 16-Layer PCB from Pins L11 and L6**

As a general rule, the P and N paths need to be kept at equal lengths through a transition. Where possible, via stub length should be kept to a minimum by traversing the signal through the entire length of the vias. The analysis shown in Figure 13-15 compares the S-parameter return loss for common-mode (SCC11) and differential (SDD11) responses.

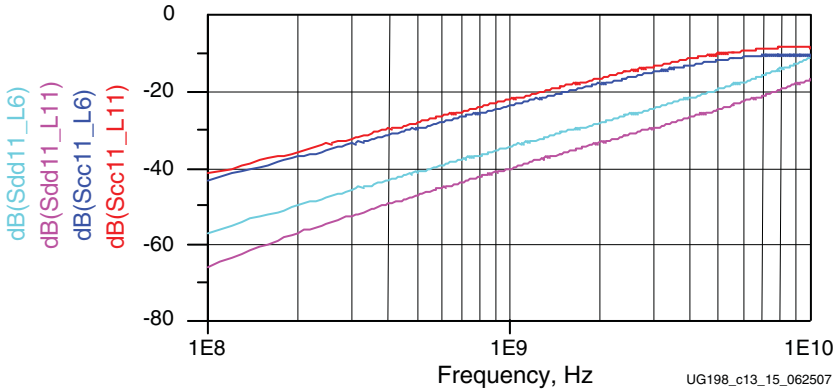


Figure 13-15: **Simulated Return Loss Comparing Differential and Common-Mode Losses for L11 and L6 GSSG Vias**

From the graph in Figure 13-15, the common-mode response is 20 dB worse in terms of return loss. The much worse common-mode response relative to the differential response is the reason why it is a good idea to reduce P/N skew as much as possible before entering a transition. The 60/40 rule of thumb is 40 dB of return loss at 1 GHz, which implies 60 fF of excess capacitance. Because excess capacitance is a single pole response, simple extrapolation rules can be used. For example, a shift to 34 dB return loss doubles the excess capacitance. Due to the excellent performance characteristics of GSSG vias, even long via stubs only double the differential via's capacitance at the most.

Chapter 14, "Guidelines and Examples," provides additional examples of differential vias.

P/N Crossover Vias

Some transceivers offer the ability to independently switch the polarity of the transmit and receive signal pairs. This functionality eliminates the need to cross over the P/N signals at the board level, which in turn significantly enhances signal integrity. If possible, P/N crossover vias are to be avoided and the polarity switch of the transceiver should be used.

SMA Connectors

Well-designed SMA connectors can reduce debugging time and allow a high-performance channel to be designed correctly on the first pass. SMA connectors that perform well at 10 Gb/s need to be simulated, designed, and manufactured to meet this performance target. Vendors can also offer design services that ensure that the connector works well on a specific board. Assembly guidelines are crucial in ensuring that the process of mating the connector to the board is well-controlled to give the specified performance.

Xilinx uses Rosenberger SMA connectors almost exclusively because of their excellent performance and because of the points listed in the previous paragraph.

Backplane Connectors

There are numerous signal integrity issues associated with backplane connectors including:

- P/N signal skew
- Crosstalk
- Stubs due to connector pins

Chapter 14, "Guidelines and Examples," provides a design example based on the popular HM-Zd connector.

Some connector manufacturers offer not only S parameters, models, and layout guidelines for their connectors but also design support, seminars, and tutorials.

Microstrip/Stripline Bends

A bend in a PCB trace is a transition. When routing differential traces through a 90° corner, the outer trace is longer than the inner trace, which introduces P/N imbalance. Even within a single trace, signal current has the tendency to hug the inside track of a corner, further reducing the actual delay through a bend.

To minimize skew between the P and N paths, 90° turns in microstrips or striplines are routed as two 45° bends to give mitered corners. The addition of a jog-out also allows the trace lengths to be matched. Figure 13-16 shows example bends in traces.

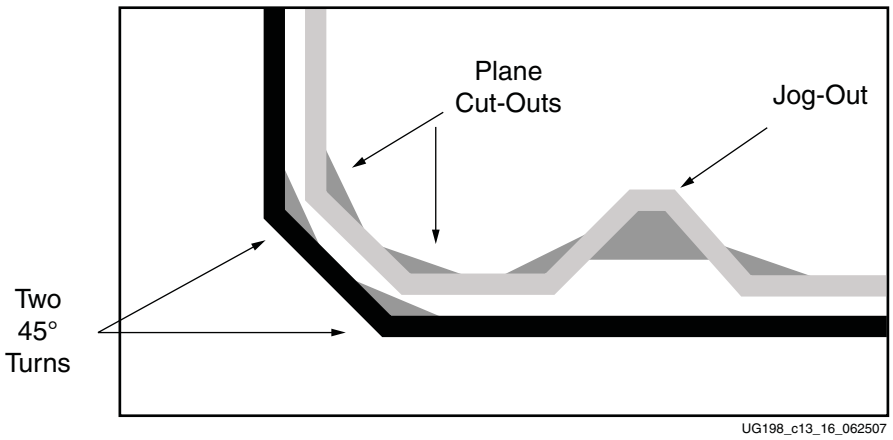


Figure 13-16: **Example Design for 90 Degree Bends in Traces**

Turns add capacitance because the trace at a 90° corner is 41% wider. That difference is reduced to 8% with a 45° turn. The addition of plane cutouts to a depth of 30 mils act to reduce this amount of excess capacitance. The trace was not widened to maintain 50Ωwith the plane cutouts in place.

When this mitered bend is simulated with the jog-out and plane cutouts, excess capacitance is reduced and P/N length and phase matching is improved. Without jog-outs, the P/N length mismatch is 16 mils. Given FR4 material, the 16 mil difference translates to a phase mismatch of 4.8° at 5 GHz, or 2.68 ps (0.0268 UI) at 10 Gb/s.

Figure 13-17 through Figure 13-19 show that phase mismatch is reduced to 0.75° with jog-outs and 0.3° with jog-outs and plane cutouts. The combination of jog-outs and plane cutouts yields simulation results that show the excess capacitance of the structure is reduced to 65 fF.

Designers are tempted to widen lines to compensate for the characteristic impedance increase as the lines are separated and couple less strongly. However, even without widening the lines, the combined capacitance of the corners and jog-outs is still overly capacitive, and therefore the uncoupled section of the jog-out must not be widened.

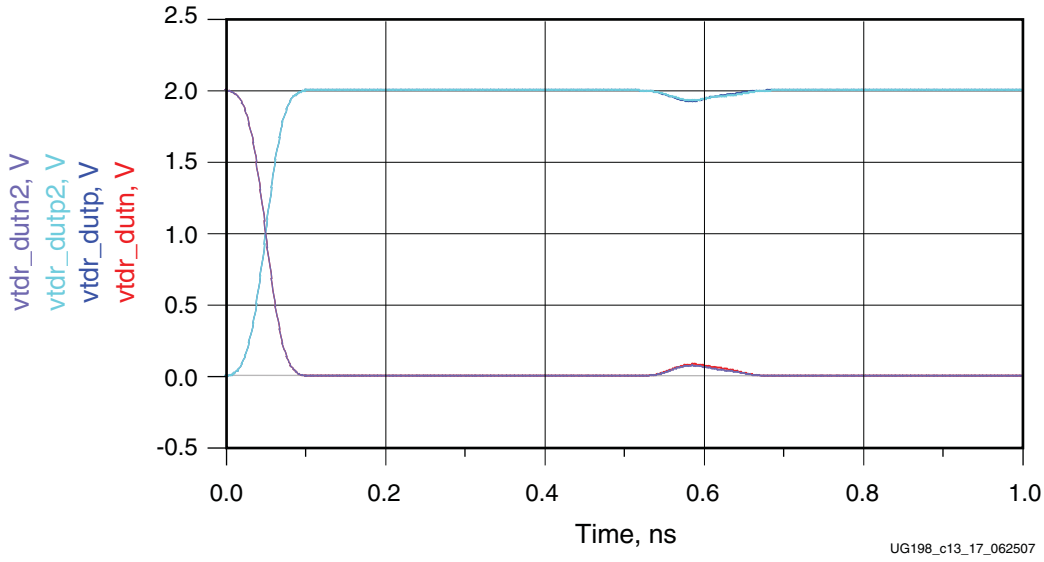


Figure 13-17: **Simulated TDR of 45 Degree Bends with Jog-Outs**

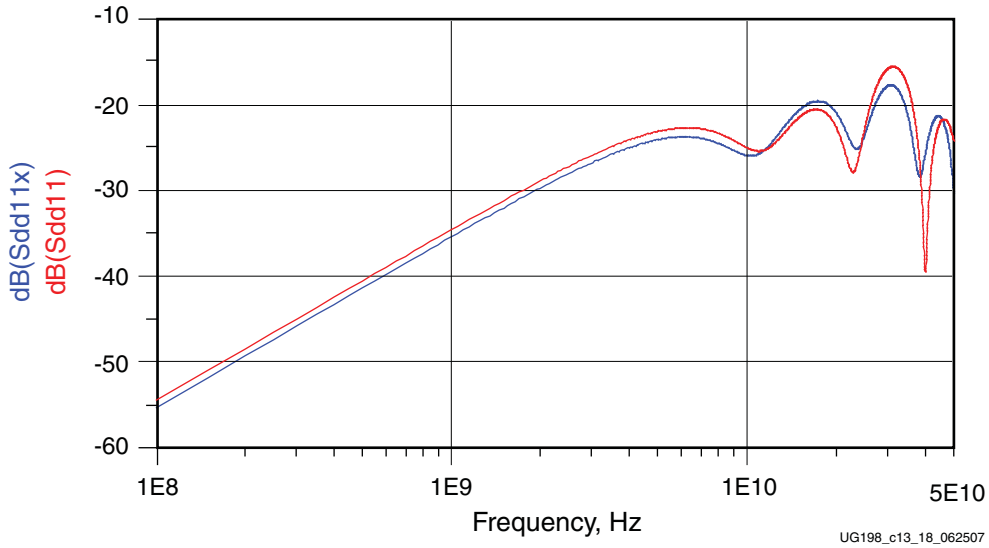


Figure 13-18: **Simulated Return Loss of 45 Degree Bends with Jog-Outs**

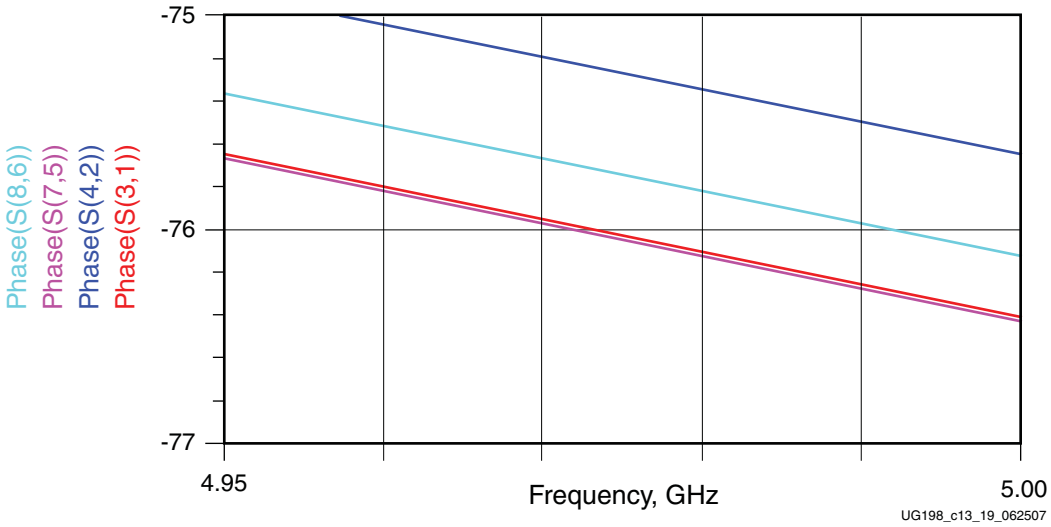


Figure 13-19: **Simulated Phase Response of 45 Degree Bends with Jog-Outs**

For wide traces, curved routing can also be helpful as shown in Figure 13-20.

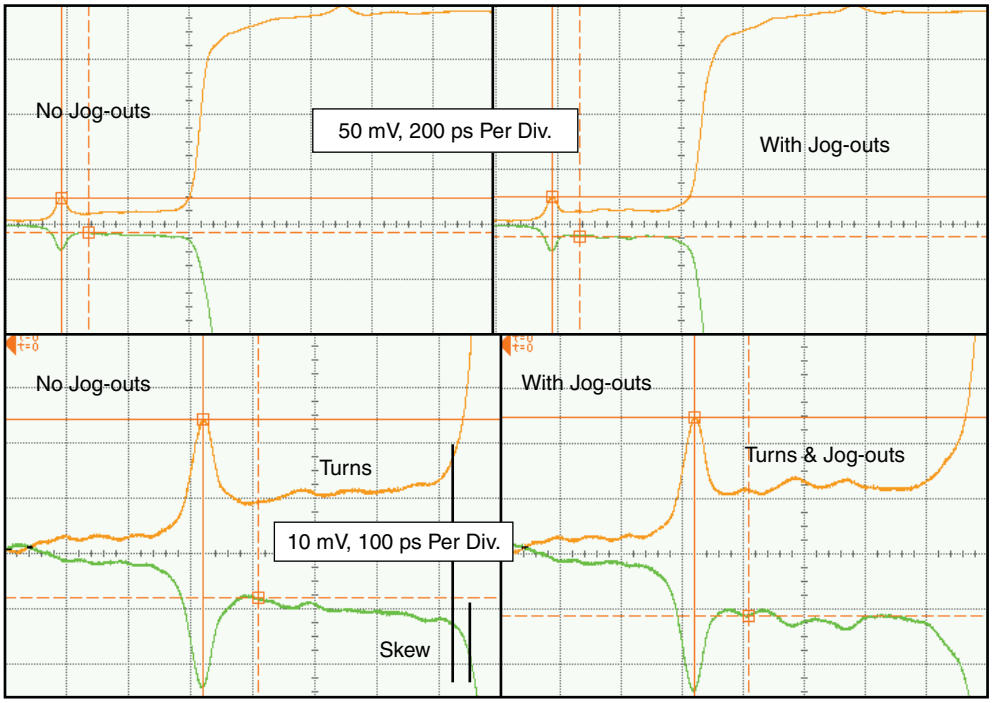


Figure 13-20: **Measured TDR of 45 Degree Bends with and without Jog-Outs**

XILINX®

Chapter 14

Guidelines and Examples

This chapter discusses high-level PCB guidelines and strategies. Design examples are provided that show how these guidelines are applied how transitions can be modified to accommodate specific applications.

Summary of Guidelines

This high-level summary provides a quick reference to some of the guidelines already covered in previous sections, and also introduces some general strategies when designing high-speed serial channels.

When defining the stack-up, high-speed stripline layers are kept near the bottom of the board. If all high-speed traces can be routed on the top and/or the bottom microstrip layers, there is no need for a stripline layer. Wider traces are preferred and widths of 6 mils to 12 mils are typical.

Unless there are tight space constraints, the differential trace pairs do not need to be coupled closely. For example, instead of using a 5 mil width with 5 mil spacing, the same characteristic trace impedance can be obtained using a 7 mil trace width with 12 mil spacing.

High-speed differential pairs and transitions must be spread apart on adjacent channels generously to limit crosstalk, even if the paths become longer. In most cases, they eventually have to be spread out to match connector pin spacings.

For transitions, large clearances of planes must be provided around and below transitions to limit excess capacitance. Transitions are spaced apart within the same channel. For example, differential vias typically are not placed next to DC blocking capacitors or connectors. However, in some specific cases, performance was acceptable with this placement.

To further limit excess capacitance in vias, the unused pads on vias should be removed and the via stub length is kept to a minimum. By routing from the top microstrip to the bottom microstrip, the via stub can be eliminated. Routing from the top microstrip to the bottom-most stripline layer results in a negligible via stub. If the lowest layers are not available for high-speed striplines, other striplines can be used. However, the via stub should be removed by back-drilling the vias.

Use of minimum spacing and clearance design rules is to be avoided, such as 5 mil pad clearances. These clearances can be detrimental to performance even at lower multi-gigabit rates due to the excess capacitance from the tight spacing.

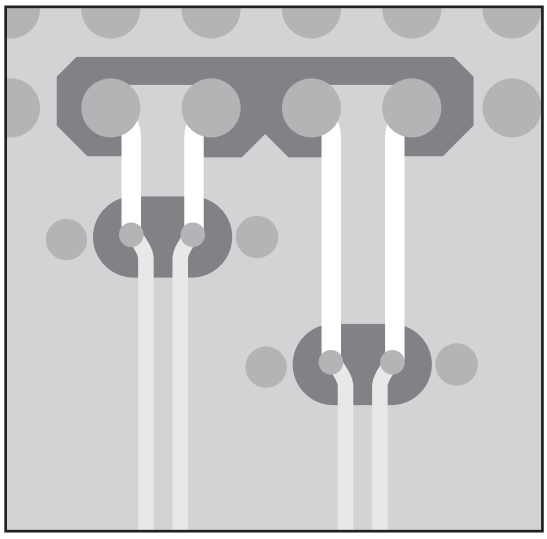
Most transitions shown in this document have 40 fF to 200 fF of excess capacitance. One exception is a press-fit connector with the PCB pin array having about 500 fF to 800 fF of excess capacitance using these guidelines, with a via stub less than 10 mils. With smaller antipads or longer via stubs, the excess capacitance is much greater. Because the

transceivers have a die capacitance of 500 fF to 600 fF, most transitions can be designed to have a very small impact on performance up to speeds of 10 Gb/s.

These guidelines are recommended to be followed even for designs slower than 10 Gb/s, allowing for more margin at lower speeds such that a smaller output signal swing can be used. Having a 10 Gb/s capable channel also provides the option to upgrade the bandwidth of the system for the next generation product.

BGA Escape Example

The transceiver signal pairs are routed along the edges of the flip-chip BGA. A microstrip is used to escape. When there is adequate spacing from the BGA, the optimized GSSG differential vias are used to change layers, if needed. It is recommended that these vias be staggered, as shown in Figure 14-1, to minimize the formation of slots in the power and ground planes.



UG198_c14_01_062507

Figure 14-1: **BGA Escape Design Example**

The round BGA pads for the transceiver signals present a small amount of capacitance to a solid PCB ground below. Therefore one option is to open a void in the ground plane below the signal pads with the same diameter as the signal pads. However, simulations show that the void only removes 30 fF of capacitance.

HM-Zd Design Example

For backplane applications, in-line connectors such as the one shown in Figure 14-2, are the most common. Of these connectors, the most common mounting method is press-fit, although SMT connectors offer much better performance.

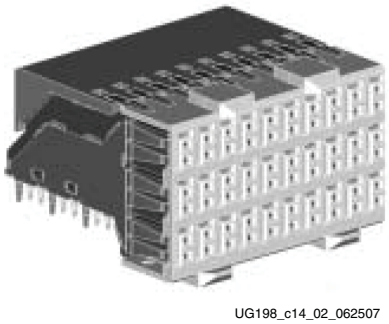


Figure 14-2: **Tyco Z-PACK HM-Zd Press-Fit Connector**

The right-angle connectors have P/N length differences in the signal paths, as shown in Figure 14-3, that require PCB trace lengths to be adjusted to compensate for the skew.

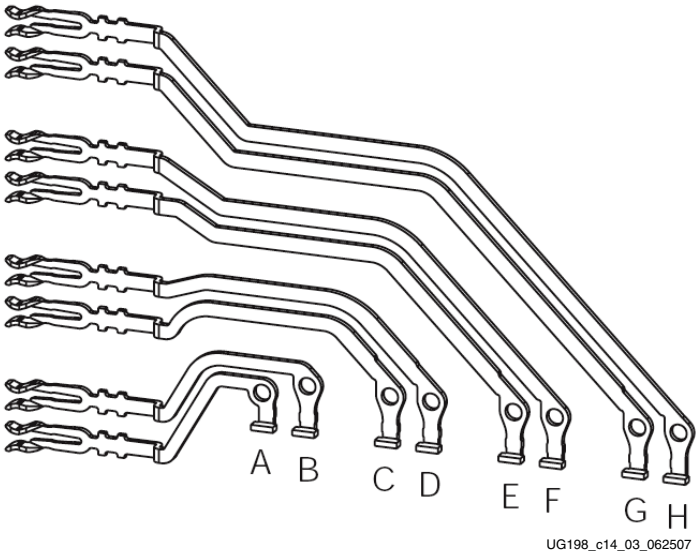


Figure 14-3: **Tyco Z-PACK HM-Zd Press-Fit Connector Internals**

Figure 14-4 shows an example design where the traces are preskewed to compensate for P/N length mismatches within the connector body.

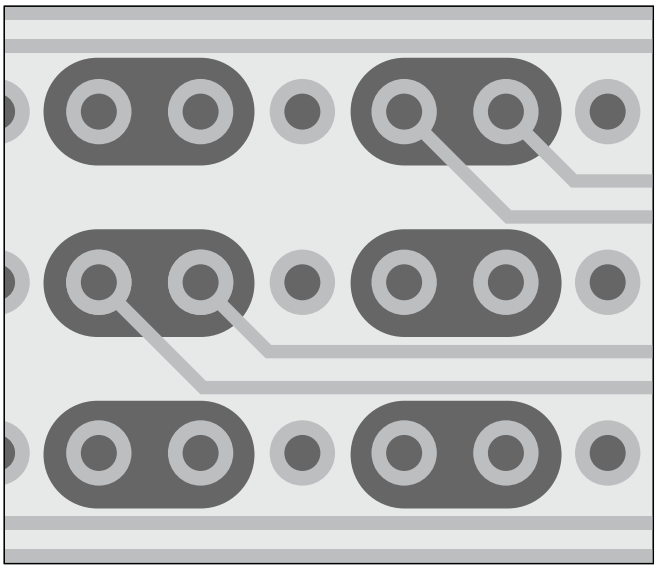


Figure 14-4: **Tyco Z-PACK HM-Zd Press-Fit Connector Design Example**

Press-fit connectors require large vias that allow the connector pins to be inserted. These vias are on a fixed pitch to match the pitch of the connector pins. Having large vias on a tight pitch results in excess capacitance.

To mitigate this excess capacitance, via stubs must be kept short. Because the connector pin is around 95 mils, backdrilling can only be done to that depth. Routing on the lower layers helps to reduce the via stub length.

Making the antipads around the differential vias as large as possible minimizes capacitance. As shown in Figure 14-4, the antipad size is maximized such that the ground reference for the traces extends beyond the edge of the striplines by about 3 mils.

All power and ground planes that do not provide an impedance reference to the striplines or microstrips should be removed. Vias need to be distributed around the periphery of the connector to stitch the ground planes together.

The designer is recommended to taper the wide microstrips or striplines as they enter the connector footprint. However, this technique has not yet been fully validated. The reduced trace width causes additional line loss and inter-symbol interference (ISI) effects from the greater impedance variation. These effects can be offset by the additional performance gained from larger antipads with less excess capacitance.

Section 3: Appendices

XILINX®

Appendix A

MGT to GTX Transceiver Design Migration

Overview

This appendix describes important differences regarding migration from the Virtex-II Pro and Virtex-4 multi-gigabit transceivers (MGTs) to the Virtex-5 FPGA RocketIO GTX transceivers. This appendix does not describe all of the features and capabilities of these devices but only highlights relevant PCB, power supply, and reference clock differences. For more information on Virtex-II Pro and Virtex-4 FPGAs, refer to *Virtex-II Pro and Virtex-II Pro X Complete Data Sheet* [Ref 12], *RocketIO Transceiver User Guide* [Ref 13], and *Virtex-4 RocketIO Multi-Gigabit Transceiver User Guide* [Ref 14].

Primary Differences

Virtex-5 FXT/TXT FPGAs are a different family from the Virtex-II Pro and Virtex-4 families. The Virtex-5 FXT/TXT devices are not pin compatible with these previous generation devices. However, many aspects of the MGTs and GTX transceivers are the same between families. The primary differences are:

- Number of MGTs and GTX transceivers per device
- Clocking
- Serial rates and ranges
- Encoding standards – 8B/10B, 64B/66B, SONET, and others
- Clock multiplier settings and PLL ranges
- Flexibility due to partial reconfiguration, PMA programming bus, dynamic reconfiguration port (DRP)
- Board design guidelines

MGTs per Device

Virtex-5 FPGAs allow for a large range of GTP and GTX transceivers per device. Table A-1 shows the number of transceivers available for each family.

Table A-1: **Transceivers per Device**

Virtex Device	# of Transceivers
Virtex-II Pro FPGA	4, 8, 12, 16, 20
Virtex-4 FPGA	8, 12, 16, 20, 24

Table A-1: **Transceivers per Device *(Cont'd)***

Virtex Device	# of Transceivers
Virtex-5 LXT and SXT FPGA(1)	8, 12, 16, 24
Virtex-5 FXT FPGA(2)	8, 12, 16, 24
Virtex-5 TXT FPGA(2)	40, 48

Notes:

1. Because two GTP transceivers use shared PMA PLL resources in a GTP_DUAL tile, applications where transceivers do not have common clock settings may not be able to use both transceivers in a tile. This will reduce the total number of available transceivers in these applications.
2. Because two GTX transceivers use shared PMA PLL resources in a GTX_DUAL tile, applications where transceivers do not have common clock settings may not be able to use both transceivers in a tile. This will reduce the total number of available transceivers in these applications.

Clocking

Virtex devices provide several available clock inputs. Table A-2 shows the clocks for each family and their respective serial speeds.

Table A-2: **Available Clock Inputs**

Family	Clock Names	Differential (Internal)	Dedicated Routes	Max Serial Speeds (Gb/s)	Dynamic Switching	Package Input Voltage (V)(1)	Inputs per Device	Clocks per Device
Virtex-II Pro FPGA	BREFCLK		Yes	3.125	Yes(2)	2.5	8(3)	2(3)
	BREFCLK2		Yes	3.125	Yes(2)	2.5	8(3)	2(3)
	REFCLK			2.5	Yes(2)	2.5	8(3)	2(3)
	REFCLK2			2.5	Yes(2)	2.5	8(3)	2(3)
Virtex-4 FPGA	GREFCLK	Yes	Yes	1.0	Yes(4)		Note 5	Note 5
	REFCLK1	Yes	Yes	6.5	Yes(4)		8	4
	REFCLK2	Yes	Yes	6.5	Yes(4)		8	4
Virtex-5 LXT and SXT FPGA	GREFCLK	Yes	Yes		Yes		1 per GTP_DUAL tile	1 per GTP_DUAL tile
	REFCLK	Yes	Yes	3.75	Yes		1 per GTP_DUAL tile	1 per GTP_DUAL tile

Table A-2: **Available Clock Inputs**

Family	Clock Names	Differential (Internal)	Dedicated Routes	Max Serial Speeds (Gb/s)	Dynamic Switching	Package Input Voltage (V)(1)	Inputs per Device	Clocks per Device
Virtex-5 FXT and TXT FPGA	GREFCLK	Yes	Yes		Yes		1 per GTX_DUAL tile	1 per GTX_DUAL tile
	REFCLK	Yes	Yes	6.5	Yes		1 per GTX_DUAL tile	1 per GTX_DUAL tile

Notes:

1. Nominal values. Refer to the specific data sheet for the exact values.
2. Dynamic selection between the REFCLKs or the BREFCLKs. To switch from REFCLK to BREFCLK or vice versa requires reconfiguration.
3. BREFCLK should use dedicated GCLK I/O, which decreases GCLK I/O resources for other logic (also two pins per clock).
4. Reference clock switching is done via an attribute and the DRP using the RXAPMACLKSEL, RXBPMACLKSEL, and TXABPMACLKSEL attributes. These attributes are located at DRP address `0x5D` on bits [13:12], [11:10], and [9:8], respectively.
5. GREFCLK comes from the global clock tree and can come from any FPGA clock input. It should only be used for serial rates under 1.0 Gb/s.

Clock selection changed slightly across the first three generations of MGTs. In contrast, the GTP_DUAL or GTX_DUAL tiles significantly enhance clocking capabilities by adding dedicated clocks routing and MUXing resources. Figure A-1 shows how the reference clocks are selected for each device.

There is an important difference between the GTP_DUAL and GTX_DUAL tiles: REFCLKPWRDNB powers down the entire reference clock circuit on the GTP_DUAL tile but only powers down the IBUFDS circuit that brings in CLKP and CLKN on the GTX_DUAL tile. All other clocks are free to flow, including CLKOUTNORTH, CLKINNORTH, CLKOUTSOUTH, CLKINSOUTH, and REFCLK as long as power is applied to the GTX_DUAL tile.

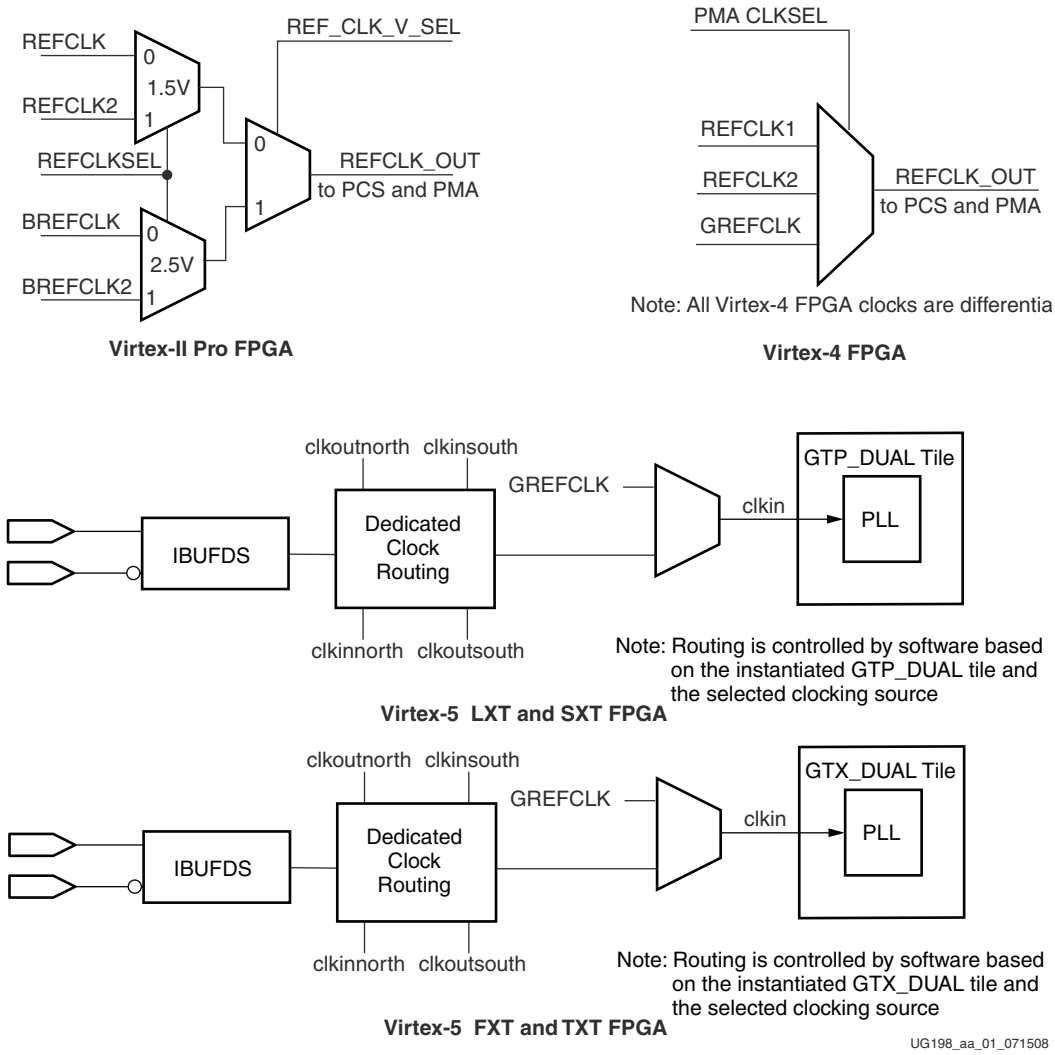


Figure A-1: **Reference Clock Selection for Each Device**

Serial Rate Support

As the Xilinx transceivers continue to migrate, so do the supported serial rates. Table A-3 shows the rates supported by each MGT, GTP, or GTX transceiver.

Table A-3: **Serial Rate Support**

Virtex-II Pro MGT	Virtex-4 MGT	Virtex-5 FPGA (LXT or SXT) GTP Transceiver	Virtex-5 FPGA (FXT or TXT) GTX Transceiver
0.622 – 3.125 Gb/s	0.622 – 6.5 Gb/s	0.100(1) – 3.75 Gb/s	150 Mb/s(2) to 6.5 Gb/s

Notes:

1. Use of the built-in 5x digital oversampling block required for rates from 100 Mb/s to 500 Mb/s.
2. Use of the built-in 5x digital oversampling block required for rates from 150 Mb/s to 750 Mb/s.

Encoding Support and Clock Multipliers

Protocol encoding support and clock multiplier support vary depending on the transceiver generation. Table A-4 shows the encoding support available in each MGT, GTP, or GTX transceiver.

Table A-4: **Encoding Support**

Encoding Schemes	Virtex-II Pro MGT	Virtex-4 MGT	Virtex-5 FPGA (LXT and SXT) GTP Transceiver	Virtex-5 FPGA (FXT and TXT) GTX Transceiver
8B/10B	Yes	Yes	Yes	Yes
64B/66B	Yes(1)	Yes	Yes(1)	Yes(1)
SONET	Yes(1)	Yes	Yes	Yes
Others	Yes(2)	Yes(2)	Yes(2)	Yes(2)

Notes:

1. Encoding and clocks must be done in the FPGA logic.
2. Depending on encode, some functionality must be done in the FPGA logic.

Reference clock multiplication has also evolved over the transceiver generations from a limited selection of multiplier values to a fully programmable solution starting with Virtex-4 devices. Table A-5 shows the clock multiplier values supported in the Virtex-II Pro devices.

Table A-5: **Virtex-II Pro Clock Multipliers**

	Supported Clock Multiplier Values
Virtex-II Pro FPGA	20

The Virtex-4 and Virtex-5 devices use the circuitry shown in Figure A-2 to multiply the reference clock.

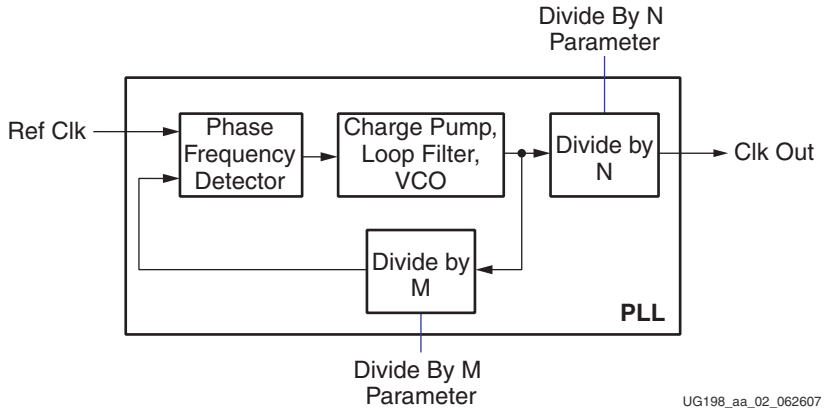


Figure A-2: **Virtex-4 and Virtex-5 FPGA Clock Multiplication Circuitry**

Table A-6 shows the parameters used to configure the operation of the clock multiplication circuitry as well as the supported divide values. While Virtex-4 devices support separate multiply ratios for transmit and receive operations, one multiply ratio is used for both in Virtex-5 devices.

Table A-6: **Virtex-4 and Virtex-5 FPGA Clock Multiplication Parameters**

	Virtex-4 MGT	Virtex-5 FPGA (LXT and SXT) GTP Transceiver	Virtex-5 FPGA (FXT and TXT) GTX Transceiver
Divide by M Parameter	TXPLLNDIVSEL, RXPLLNDIVSEL	PLL_DIVSEL_FB	PLL_DIVSEL_FB
Divide by M Values	8, 10, 16, 20, 32, 40	1, 2, 3, 4, 5	1, 2, 3, 4, 5
Divide by N Parameter	TXOUTDIV2SEL, RXOUTDIV2SEL	PLL_DIVSEL_REF	PLL_DIVSEL_REF
Divide by N Values	1, 2, 4, 8, 16, 32	1, 2	1, 2

Flexibility

In Virtex-II Pro devices, changing of attributes requires partial reconfiguration. Virtex-4 and Virtex-5 devices allow all attribute changes from the DRP, and any default values can be set in the HDL, the User Constraint File (UCF), or both.

Board Guidelines

Power Supply Filtering

The Virtex-5 FPGA GTP and GTX transceivers simplify power supply design over previous generations in two ways:

1. Only two different supply voltages are needed to power the transceivers.
2. Because the two transceivers that share a tile have common power pins, the number of power supply filtering components is reduced.

Note: The power supply design guidelines in Chapter 10, "GTX-to-Board Interface," must be strictly followed.

Refer to the MGT-to-board section of the specific transceiver's user guide for guidelines about power supply design. Some transceivers require independent voltage regulators and filter circuits even when the voltages of different analog supplies are identical.

Table A-7 shows the power pin voltages for all Virtex families, and Figure A-3 shows the power supply filtering for all Virtex families.

Table A-7: **Power Pin Voltages**

Pin	Virtex-II Pro MGT	Virtex-4 MGT	Virtex-5 FPGA (LXT and SXT) GTP Transceiver(1)	Virtex-5 FPGA (FXT and TXT) GTX Transceiver(1)
AVCCAUXRX	2.5V	1.2V	-	-
AVCCAUXTX	2.5V	1.2V	-	-
AVCCAUXMGT	N/A	2.5V	-	-
VTTX	1.8 - 2.5V(2)	1.5V	-	-
VTRX	1.5 - 2.5V(2)	0.25 - 2.5V(2)	-	-
MGTAVCCPLL	-	-	1.2V	1.0V
MGTAVCC	-	-	1.0V	1.0V
MGTAVTTTX	-	-	1.2V	1.2V
MGTAVTTRX	-	-	1.2V	1.2V
MGTAVTTRXC	-	-	1.2V	1.2V

Notes:

1. Nominal values. Refer to the device data sheets for values and operating conditions.
2. Depends on AC/DC coupling or termination options. See the device user guides for more details.

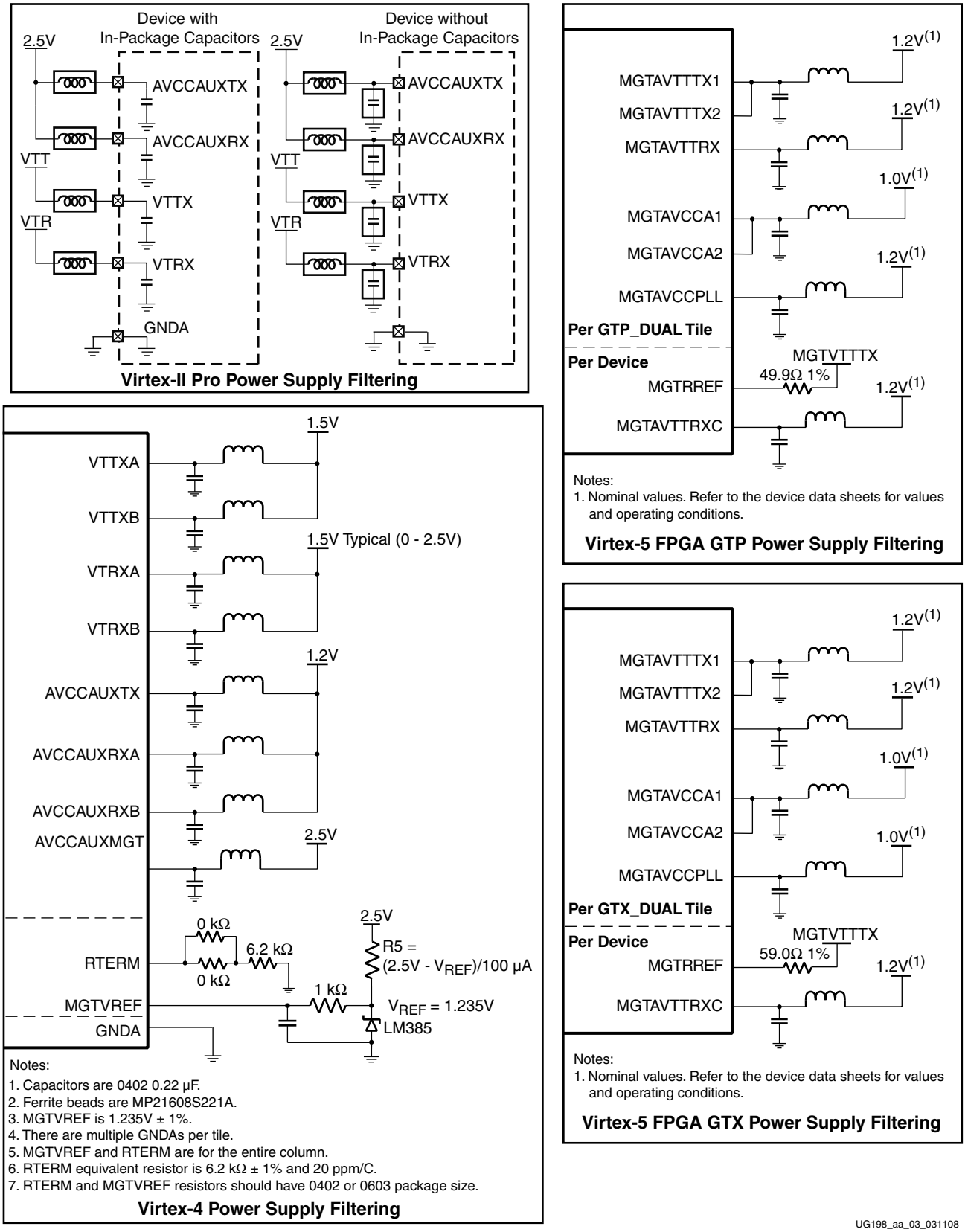


Figure A-3: **Virtex-II, Virtex-II Pro, Virtex-4, Virtex-5 FPGA Power Supply Filtering**

Other Minor Differences

Termination

In Virtex-II Pro devices, the transceivers contain on-chip termination and reference voltages for VTTX and VTRX. In Virtex-4 devices, the MGTs use a reference resistor to create the termination circuitry for each MGT column. The Virtex-5 FPGA GTP and GXT transceivers simplify the termination circuitry by providing a calibrated 50Ω termination. Table A-8 shows the termination options for each FPGA generation.

Table A-8: **Termination Options**

Termination	Virtex-II Pro MGT	Virtex-4 MGT	Virtex-5 FPGA (LXT and SXT) GTP Transceiver	Virtex-5 FPGA (FXT) GTX Transceiver
Value	50/75 Ω	50Ω	50Ω	50Ω
Voltage Pins	VTTX/VTRX	VTTX/VTRX	MGTAVTTTX/MGTAVTTRX	MGTAVTTTX/MGTAVTTRX

FPGA Logic Interface

FPGA logic interface width support varies depending on the transceiver. Table A-9 shows the widths supported in all MGT, GTP, and GTX transceivers.

Table A-9: **FPGA Logic Interface Support**

Virtex-II Pro MGT	Virtex-4 MGT	Virtex-5 FPGA GTP Transceiver	Virtex-5 FPGA GTX Transceiver
1-byte, 2-byte, 4-byte, 8-byte	1-byte, 2-byte, 4-byte, 8-byte	1-byte, 2-byte	1-byte, 2-byte, 4-byte

CRC

Table A-10 shows the CRC support provided by all four transceiver families.

Table A-10: **CRC Transceiver Support**

Virtex-II Pro MGT	Virtex-4 MGT	Virtex-5 FPGA GTP Transceiver	Virtex-5 FPGA GTX Transceiver
CRC-32	Independent CRC-32 block supports datapath from 8 to 64 bits wide.	Independent CRC block supports two CRC-32 calculations over 32-bit datapaths or a single CRC-32 calculation over a 64-bit datapath.	Independent CRC block supports two CRC-32 calculations over 32-bit datapaths or a single CRC-32 calculation over a 64-bit datapath.

Loopback

The loopback options of the Virtex transceivers have evolved to improve flexibility. Virtex-II Pro MGTs have two loopback modes, and Virtex-4 MGTs have four loopback modes. Table A-11 shows this evolution.

Table A-11: **Loopback Options**

Mode	Virtex-II Pro MGT	Virtex-4 MGT	Virtex-5 FPGA (LXT and SXT) GTP Transceiver	Virtex-5 FPGA (FXT and TXT) GTX Transceiver
Parallel Loopback (Tx →RX)	Yes	Yes	Yes	Yes
Serial Pre-Driver	–	Yes	Yes	Yes
Serial Post-Driver	Yes	–	–	–
Serial Digital Receiver	–	Yes	–	–
External Data PMA-Only Parallel Loopback	–	–	Yes	Yes
PCI Express Repeater	–	–	Yes	Yes

Serialization

As in Virtex-4 devices, Virtex-5 FPGA GTP and GTX transceivers serialize and send the least significant byte first. Virtex-II Pro devices send the most-significant byte first.

Defining Clock Correction and Channel Bonding Sequences

The bit definitions of CLK_COR_SEQ and CHAN_BOND_SEQ have changed to support more encoding functionality. Table A-12 illustrates the differences.

Table A-12: **CLK_COR_SEQ and CHAN_BOND_SEQ Sequences**

Bit Definition	Virtex-II Pro MGT	Virtex-4 MGT	Virtex-5 FPGA (LXT and SXT) GTP Transceiver	Virtex-5 FPGA (FXT and TXT) GTP Transceiver
8B/10B encoded definition	00110111100(1,4)	00110111100(2,4)	0110111100(3,5)	0110111100(3,5)
10-bit literal value	10011111010(1,4)	10011111010(2,4)	0011111010(1,5)	0011111010(1,5)
64B/66B encoding (sync character)	N/A	1XX (sync header)	N/A	N/A
8-bit literal value (for 64B/66B and other encodings)	N/A	1XX (8-bit data)	N/A	N/A

Notes:

1. Defines K28.5.
2. Defines K28.5 and depends on CLK_COR_8B10B_DE (plus all 10 bits are defined).
3. Defines K28.5 and regular disparity.
4. For Virtex-II Pro and Virtex-4 MGTs, the 11th bit (left-most bit) determines either an 8-bit or 10-bit compare.
5. For Virtex-5 FPGA GTP and GTX transceivers, the RX_DECODE_SEQ_MATCH attribute determines if matches occur against the output of the 8B/10B decoder (RX_DECODE_SEQ_MATCH = TRUE) or against the undecoded incoming data (RX_DECODE_SEQ_MATCH = FALSE).

RXSTATUS Bus

Several buses have changed over the FPGA generations to improve the information that is indicated. Table A-13 shows the migration from Virtex-II Pro to Virtex-5 devices.

Table A-13: **Status Bus Changes**

Description	Virtex-II Pro MGT	Virtex-4 MGT	Virtex-5 FPGA (LXT and SXT) GTP Transceiver(1)	Virtex-5 FPGA (FXT and TXT) GTX Transceiver(2)
Indicates channel bonding complete	CHBONDONE(3)	RXSTATUS[5]	RXCHANISALIGNED0 RXCHANISALIGNED1	RXCHANISALIGNED0 RXCHANISALIGNED1
Indicates status bus is status, data, event	N/A	RXSTATUS[4:3]	RXSTATUS[2:0]	RXSTATUS[2:0]
Indicates channel bonding or clock correction pointers change	RXCLKCORCNT0[2:0] RXCLKCORCNT0[2:0]	RXSTATUS[2:0]	RXCHANREALIGN0 RXCHANREALIGN1	RXCHANREALIGN0 RXCHANREALIGN1
Indicates that an RX buffer has underflow/overflow	RXBUFSTATUS[1]	RXBUFERR	RXBUFSTATUS[2:0]	RXBUFSTATUS[2:0]

Notes:

1. Signal optimization settings are independent between both GTP transceivers of a GTP_DUAL tile. GTP0 is indicated by the suffix "0" after the signal name, and GTP1 is indicated by the suffix "1" (for example, RXEQMIX0 or RXEQMIX1).
2. Signal optimization settings are independent between both GTX transceivers of a GTX_DUAL tile. GTX0 is indicated by the suffix "0" after the signal name, and GTX1 is indicated by the suffix "1" (for example, RXEQMIX0 or RXEQMIX1).
3. RXCLKCORCNT must go to `3'b101` before channel bonding is complete.

Pre-emphasis, Differential Swing, and Equalization

The differential signaling techniques are more robust in recent Xilinx transceivers. The Virtex-5 FPGA GTP and GTX transceivers add ports to control TX characteristics to simplify reconfiguration. Table A-14 shows the migration of attributes from Virtex-II Pro and Virtex-4 MGTs to Virtex-5 FPGA GTP and GTX transceivers.

Table A-14: **Signal Optimization Attributes and Ports**

Description	Virtex-II Pro MGT	Virtex-4 MGT	Virtex-5 FPGA GTP Transceivers(1)		Virtex-5 FPGA GTX Transceivers(1)	
			Ports	Attributes	Ports	Attributes
Controls TX pre-emphasis and edge rate	TX_PREMPHASIS	TXPRE_PRDRV_DAC TXPRE_TAP_PD TXSLEWRATE TXPOST_PRDRV_DAC TXDAT_PRDRV_DAC TXPOST_TAP_PD	TXPREEMPHASIS[3:0]		TXPREEMPHASIS[3:0]	
Controls differential amplitude of the transmitted signal	TX_DIFF_CTRL	TXPRE_TAP_DAC TXPOST_TAP_DAC TXDAT_TAP_DAC	TXBUFDIFFCTRL0[2:0] TXBUFDIFFCTRL1[2:0] TXDIFFCTRL0[2:0] TXDIFFCTRL1[2:0]	TX_DIFF_ BOOST_0 TX_DIFF_ BOOST_1	TXBUFDIFFCTRL0[2:0] TXBUFDIFFCTRL1[2:0] TXDIFFCTRL0[2:0] TXDIFFCTRL1[2:0]	

Table A-14: **Signal Optimization Attributes and Ports *(Cont'd)***

Description	Virtex-II Pro MGT	Virtex-4 MGT	Virtex-5 FPGA GTP Transceivers(1)		Virtex-5 FPGA GTX Transceivers(1)	
			Ports	Attributes	Ports	Attributes
Active equalization	N/A	RXAFEEQ	RXENEQB0 RXENEQB1 RXEQMIX0[1:0] RXEQMIX1[1:0] RXEQPOLE0[3:0] RXEQPOLE1[3:0]		RXEQMIX0[1:0] RXEQMIX1[1:0]	
Discrete equalization	N/A	RXEQ	N/A	N/A	Refer to DFE related ports in Table 7-5, page 163.	DFE_CFG_0[9:0] DFE_CFG_1[9:0]

Notes:

1. Signal optimization settings are independent between both GTP transceivers of a GTP_DUAL tile and GTX transceivers of a GTX_DUAL tile. GTP0 and GTX0 are indicated by the suffix "0" after the signal name, and GTP1 and GTX1 are indicated by the suffix "1" (for example RXEQMIX0, RXEQMIX1).

Appendix B

OOB/Beacon Signaling

The GTX transceiver supports Out-of-Band (OOB) signaling for conformance with standards such as SATA and supports beaconing for conformance to the PCI Express specification. OOB signaling mechanisms are used to send low-speed signals between the transmitter and receiver when high-speed serial data transmission is not active, typically when the link is in a power-down state or has not been initialized.

OOB signaling uses non-differential transitions on the differential inputs of the receiver to send information. To send an OOB signal, transmitters drive their serial differential output pins to the same voltage, resulting in a reduced differential voltage between the pins. When the absolute differential voltage drops below a preset threshold level, the receiver detects the signal as an OOB signal. Figure B-1 illustrates this concept.

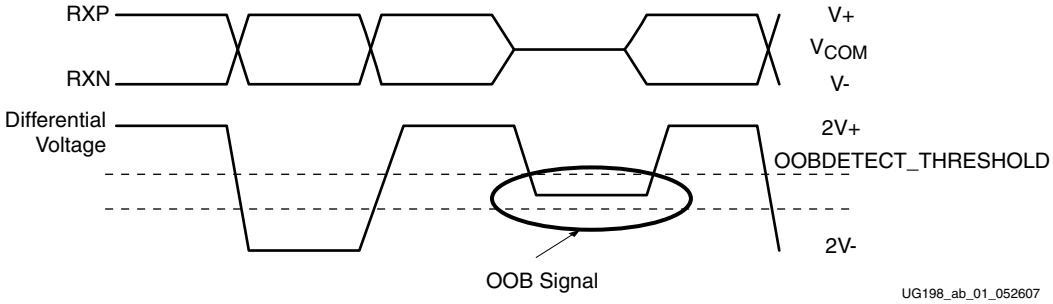


Figure B-1: **OOB Signaling**

"TX Out-of-Band/Beacon Signaling," page 154 and "RX OOB/Beacon Signaling," page 170 provide details of the support for OOB signaling in the GTX transceiver TX and RX logic, respectively. The remainder of this section summarizes the use of OOB signaling in SATA operation and beaconing for PCI Express operation.

OOB Signaling for SATA Operation

SATA operation uses OOB signals as part of its COMWAKE, COMINIT, and COMRESET sequences as shown in Figure B-2. These sequences consist of a fixed length burst of non-OOB data followed by an OOB signal (called an idle signal in SATA). The length of the idle defines the type of COM sequence being received: COMWAKE sequences use 106.7 ns idles, and COMINIT/COMRESET sequences use 320 ns idles. A COM sequence is valid when it is received four times consecutively.

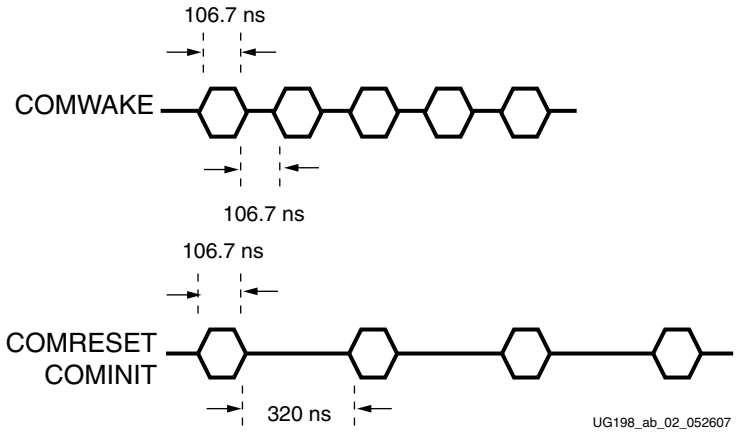


Figure B-2: **SATA COM Sequences**

In addition to the analog circuitry required to encode and detect the OOB signal state, the GTX transceiver includes state machines to format and decode bursts of OOB/beacon signals for SATA (COMRESET, COMWAKE, and COMINIT).

Beacon Signaling for PCI Express Operation

PCI Express operation uses sequences called *beacons* to wake endpoints from power-down states. A beacon is the transmission of K28.5 (COM) characters. A beacon sequence can have a frequency anywhere from 30 KHz to 500 MHz.

The PCI Express specification describes the beacon mechanism as an inband wake-up indication, and defines the use of a discrete wake signal (WAKE#) as an out-of-band mechanism.

GTX transceiver support for PCI Express beacons uses interface signals defined in the *PHY Interface for the PCI Express (PIPE) Specification*. Control logic in the FPGA manages the format of the beacon sequence.

XILINX®

Appendix C

8B/10B Valid Characters

8B/10B encoding includes a set of Data characters and K characters. Eight-bit values are coded into 10-bit values, keeping the serial line DC balanced. K characters are special Data characters designated with a CHARISK. K characters are used for specific informative designations. Table C-1 shows the valid Data characters. Table C-2, page 331 shows the valid K characters.

Table C-1: **Valid Data Characters**

Data Byte Name	Bits HGF EDCBA	Current RD – abcdei fghj	Current RD + abcdei fghj
D0.0	000 00000	100111 0100	011000 1011
D1.0	000 00001	011101 0100	100010 1011
D2.0	000 00010	101101 0100	010010 1011
D3.0	000 00011	110001 1011	110001 0100
D4.0	000 00100	110101 0100	001010 1011
D5.0	000 00101	101001 1011	101001 0100
D6.0	000 00110	011001 1011	011001 0100
D7.0	000 00111	111000 1011	000111 0100
D8.0	000 01000	111001 0100	000110 1011
D9.0	000 01001	100101 1011	100101 0100
D10.0	000 01010	010101 1011	010101 0100
D11.0	000 01011	110100 1011	110100 0100
D12.0	000 01100	001101 1011	001101 0100
D13.0	000 01101	101100 1011	101100 0100
D14.0	000 01110	011100 1011	011100 0100
D15.0	000 01111	010111 0100	101000 1011
D16.0	000 10000	011011 0100	100100 1011
D17.0	000 10001	100011 1011	100011 0100
D18.0	000 10010	010011 1011	010011 0100
D19.0	000 10011	110010 1011	110010 0100
D20.0	000 10100	001011 1011	001011 0100

Table C-1: **Valid Data Characters *(Cont'd)***

Data Byte Name	Bits HGF EDCBA	Current RD – abcdei fghj	Current RD + abcdei fghj
D21.0	000 10101	101010 1011	101010 0100
D22.0	000 10110	011010 1011	011010 0100
D23.0	000 10111	111010 0100	000101 1011
D24.0	000 11000	110011 0100	001100 1011
D25.0	000 11001	100110 1011	100110 0100
D26.0	000 11010	010110 1011	010110 0100
D27.0	000 11011	110110 0100	001001 1011
D28.0	000 11100	001110 1011	001110 0100
D29.0	000 11101	101110 0100	010001 1011
D30.0	000 11110	011110 0100	100001 1011
D31.0	000 11111	101011 0100	010100 1011
D0.1	001 00000	100111 1001	011000 1001
D1.1	001 00001	011101 1001	100010 1001
D2.1	001 00010	101101 1001	010010 1001
D3.1	001 00011	110001 1001	110001 1001
D4.1	001 00100	110101 1001	001010 1001
D5.1	001 00101	101001 1001	101001 1001
D6.1	001 00110	011001 1001	011001 1001
D7.1	001 00111	111000 1001	000111 1001
D8.1	001 01000	111001 1001	000110 1001
D9.1	001 01001	100101 1001	100101 1001
D10.1	001 01010	010101 1001	010101 1001
D11.1	001 01011	110100 1001	110100 1001
D12.1	001 01100	001101 1001	001101 1001
D13.1	001 01101	101100 1001	101100 1001
D14.1	001 01110	011100 1001	011100 1001
D15.1	001 01111	010111 1001	101000 1001
D16.1	001 10000	011011 1001	100100 1001
D17.1	001 10001	100011 1001	100011 1001
D18.1	001 10010	010011 1001	010011 1001
D19.1	001 10011	110010 1001	110010 1001
D20.1	001 10100	001011 1001	001011 1001

Table C-1: **Valid Data Characters *(Cont'd)***

Data Byte Name	Bits HGF EDCBA	Current RD – abcdei fghj	Current RD + abcdei fghj
D21.1	001 10101	101010 1001	101010 1001
D22.1	001 10110	011010 1001	011010 1001
D23.1	001 10111	111010 1001	000101 1001
D24.1	001 11000	110011 1001	001100 1001
D25.1	001 11001	100110 1001	100110 1001
D26.1	001 11010	010110 1001	010110 1001
D27.1	001 11011	110110 1001	001001 1001
D28.1	001 11100	001110 1001	001110 1001
D29.1	001 11101	101110 1001	010001 1001
D30.1	001 11110	011110 1001	100001 1001
D31.1	001 11111	101011 1001	010100 1001
D0.2	010 00000	100111 0101	011000 0101
D1.2	010 00001	011101 0101	100010 0101
D2.2	010 00010	101101 0101	010010 0101
D3.2	010 00011	110001 0101	110001 0101
D4.2	010 00100	110101 0101	001010 0101
D5.2	010 00101	101001 0101	101001 0101
D6.2	010 00110	011001 0101	011001 0101
D7.2	010 00111	111000 0101	000111 0101
D8.2	010 01000	111001 0101	000110 0101
D9.2	010 01001	100101 0101	100101 0101
D10.2	010 01010	010101 0101	010101 0101
D11.2	010 01011	110100 0101	110100 0101
D12.2	010 01100	001101 0101	001101 0101
D13.2	010 01101	101100 0101	101100 0101
D14.2	010 01110	011100 0101	011100 0101
D15.2	010 01111	010111 0101	101000 0101
D16.2	010 10000	011011 0101	100100 0101
D17.2	010 10001	100011 0101	100011 0101
D18.2	010 10010	010011 0101	010011 0101
D19.2	010 10011	110010 0101	110010 0101
D20.2	010 10100	001011 0101	001011 0101

Table C-1: **Valid Data Characters *(Cont'd)***

Data Byte Name	Bits HGF EDCBA	Current RD – abcdei fghj	Current RD + abcdei fghj
D21.2	010 10101	101010 0101	101010 0101
D22.2	010 10110	011010 0101	011010 0101
D23.2	010 10111	111010 0101	000101 0101
D24.2	010 11000	110011 0101	001100 0101
D25.2	010 11001	100110 0101	100110 0101
D26.2	010 11010	010110 0101	010110 0101
D27.2	010 11011	110110 0101	001001 0101
D28.2	010 11100	001110 0101	001110 0101
D29.2	010 11101	101110 0101	010001 0101
D30.2	010 11110	011110 0101	100001 0101
D31.2	010 11111	101011 0101	010100 0101
D0.3	011 00000	100111 0011	011000 1100
D1.3	011 00001	011101 0011	100010 1100
D2.3	011 00010	101101 0011	010010 1100
D3.3	011 00011	110001 1100	110001 0011
D4.3	011 00100	110101 0011	001010 1100
D5.3	011 00101	101001 1100	101001 0011
D6.3	011 00110	011001 1100	011001 0011
D7.3	011 00111	111000 1100	000111 0011
D8.3	011 01000	111001 0011	000110 1100
D9.3	011 01001	100101 1100	100101 0011
D10.3	011 01010	010101 1100	010101 0011
D11.3	011 01011	110100 1100	110100 0011
D12.3	011 01100	001101 1100	001101 0011
D13.3	011 01101	101100 1100	101100 0011
D14.3	011 01110	011100 1100	011100 0011
D15.3	011 01111	010111 0011	101000 1100
D16.3	011 10000	011011 0011	100100 1100
D17.3	011 10001	100011 1100	100011 0011
D18.3	011 10010	010011 1100	010011 0011
D19.3	011 10011	110010 1100	110010 0011
D20.3	011 10100	001011 1100	001011 0011

Table C-1: **Valid Data Characters *(Cont'd)***

Data Byte Name	Bits HGF EDCBA	Current RD – abcdei fghj	Current RD + abcdei fghj
D21.3	011 10101	101010 1100	101010 0011
D22.3	011 10110	011010 1100	011010 0011
D23.3	011 10111	111010 0011	000101 1100
D24.3	011 11000	110011 0011	001100 1100
D25.3	011 11001	100110 1100	100110 0011
D26.3	011 11010	010110 1100	010110 0011
D27.3	011 11011	110110 0011	001001 1100
D28.3	011 11100	001110 1100	001110 0011
D29.3	011 11101	101110 0011	010001 1100
D30.3	011 11110	011110 0011	100001 1100
D31.3	011 11111	101011 0011	010100 1100
D0.4	100 00000	100111 0010	011000 1101
D1.4	100 00001	011101 0010	100010 1101
D2.4	100 00010	101101 0010	010010 1101
D3.4	100 00011	110001 1101	110001 0010
D4.4	100 00100	110101 0010	001010 1101
D5.4	100 00101	101001 1101	101001 0010
D6.4	100 00110	011001 1101	011001 0010
D7.4	100 00111	111000 1101	000111 0010
D8.4	100 01000	111001 0010	000110 1101
D9.4	100 01001	100101 1101	100101 0010
D10.4	100 01010	010101 1101	010101 0010
D11.4	100 01011	110100 1101	110100 0010
D12.4	100 01100	001101 1101	001101 0010
D13.4	100 01101	101100 1101	101100 0010
D14.4	100 01110	011100 1101	011100 0010
D15.4	100 01111	010111 0010	101000 1101
D16.4	100 10000	011011 0010	100100 1101
D17.4	100 10001	100011 1101	100011 0010
D18.4	100 10010	010011 1101	010011 0010
D19.4	100 10011	110010 1101	110010 0010
D20.4	100 10100	001011 1101	001011 0010

Table C-1: **Valid Data Characters *(Cont'd)***

Data Byte Name	Bits HGF EDCBA	Current RD – abcdei fghj	Current RD + abcdei fghj
D21.4	100 10101	101010 1101	101010 0010
D22.4	100 10110	011010 1101	011010 0010
D23.4	100 10111	111010 0010	000101 1101
D24.4	100 11000	110011 0010	001100 1101
D25.4	100 11001	100110 1101	100110 0010
D26.4	100 11010	010110 1101	010110 0010
D27.4	100 11011	110110 0010	001001 1101
D28.4	100 11100	001110 1101	001110 0010
D29.4	100 11101	101110 0010	010001 1101
D30.4	100 11110	011110 0010	100001 1101
D31.4	100 11111	101011 0010	010100 1101
D0.5	101 00000	100111 1010	011000 1010
D1.5	101 00001	011101 1010	100010 1010
D2.5	101 00010	101101 1010	010010 1010
D3.5	101 00011	110001 1010	110001 1010
D4.5	101 00100	110101 1010	001010 1010
D5.5	101 00101	101001 1010	101001 1010
D6.5	101 00110	011001 1010	011001 1010
D7.5	101 00111	111000 1010	000111 1010
D8.5	101 01000	111001 1010	000110 1010
D9.5	101 01001	100101 1010	100101 1010
D10.5	101 01010	010101 1010	010101 1010
D11.5	101 01011	110100 1010	110100 1010
D12.5	101 01100	001101 1010	001101 1010
D13.5	101 01101	101100 1010	101100 1010
D14.5	101 01110	011100 1010	011100 1010
D15.5	101 01111	010111 1010	101000 1010
D16.5	101 10000	011011 1010	100100 1010
D17.5	101 10001	100011 1010	100011 1010
D18.5	101 10010	010011 1010	010011 1010
D19.5	101 10011	110010 1010	110010 1010
D20.5	101 10100	001011 1010	001011 1010

Table C-1: **Valid Data Characters *(Cont'd)***

Data Byte Name	Bits HGF EDCBA	Current RD – abcdei fghj	Current RD + abcdei fghj
D21.5	101 10101	101010 1010	101010 1010
D22.5	101 10110	011010 1010	011010 1010
D23.5	101 10111	111010 1010	000101 1010
D24.5	101 11000	110011 1010	001100 1010
D25.5	101 11001	100110 1010	100110 1010
D26.5	101 11010	010110 1010	010110 1010
D27.5	101 11011	110110 1010	001001 1010
D28.5	101 11100	001110 1010	001110 1010
D29.5	101 11101	101110 1010	010001 1010
D30.5	101 11110	011110 1010	100001 1010
D31.5	101 11111	101011 1010	010100 1010
D0.6	110 00000	100111 0110	011000 0110
D1.6	110 00001	011101 0110	100010 0110
D2.6	110 00010	101101 0110	010010 0110
D3.6	110 00011	110001 0110	110001 0110
D4.6	110 00100	110101 0110	001010 0110
D5.6	110 00101	101001 0110	101001 0110
D6.6	110 00110	011001 0110	011001 0110
D7.6	110 00111	111000 0110	000111 0110
D8.6	110 01000	111001 0110	000110 0110
D9.6	110 01001	100101 0110	100101 0110
D10.6	110 01010	010101 0110	010101 0110
D11.6	110 01011	110100 0110	110100 0110
D12.6	110 01100	001101 0110	001101 0110
D13.6	110 01101	101100 0110	101100 0110
D14.6	110 01110	011100 0110	011100 0110
D15.6	110 01111	010111 0110	101000 0110
D16.6	110 10000	011011 0110	100100 0110
D17.6	110 10001	100011 0110	100011 0110
D18.6	110 10010	010011 0110	010011 0110
D19.6	110 10011	110010 0110	110010 0110
D20.6	110 10100	001011 0110	001011 0110

Table C-1: **Valid Data Characters *(Cont'd)***

Data Byte Name	Bits HGF EDCBA	Current RD – abcdei fghj	Current RD + abcdei fghj
D21.6	110 10101	101010 0110	101010 0110
D22.6	110 10110	011010 0110	011010 0110
D23.6	110 10111	111010 0110	000101 0110
D24.6	110 11000	110011 0110	001100 0110
D25.6	110 11001	100110 0110	100110 0110
D26.6	110 11010	010110 0110	010110 0110
D27.6	110 11011	110110 0110	001001 0110
D28.6	110 11100	001110 0110	001110 0110
D29.6	110 11101	101110 0110	010001 0110
D30.6	110 11110	011110 0110	100001 0110
D31.6	110 11111	101011 0110	010100 0110
D0.7	111 00000	100111 0001	011000 1110
D1.7	111 00001	011101 0001	100010 1110
D2.7	111 00010	101101 0001	010010 1110
D3.7	111 00011	110001 1110	110001 0001
D4.7	111 00100	110101 0001	001010 1110
D5.7	111 00101	101001 1110	101001 0001
D6.7	111 00110	011001 1110	011001 0001
D7.7	111 00111	111000 1110	000111 0001
D8.7	111 01000	111001 0001	000110 1110
D9.7	111 01001	100101 1110	100101 0001
D10.7	111 01010	010101 1110	010101 0001
D11.7	111 01011	110100 1110	110100 1000
D12.7	111 01100	001101 1110	001101 0001
D13.7	111 01101	101100 1110	101100 1000
D14.7	111 01110	011100 1110	011100 1000
D15.7	111 01111	010111 0001	101000 1110
D16.7	111 10000	011011 0001	100100 1110
D17.7	111 10001	100011 0111	100011 0001
D18.7	111 10010	010011 0111	010011 0001
D19.7	111 10011	110010 1110	110010 0001
D20.7	111 10100	001011 0111	001011 0001

Table C-1: **Valid Data Characters *(Cont'd)***

Data Byte Name	Bits HGF EDCBA	Current RD – abcdei fghj	Current RD + abcdei fghj
D21.7	111 10101	101010 1110	101010 0001
D22.7	111 10110	011010 1110	011010 0001
D23.7	111 10111	111010 0001	000101 1110
D24.7	111 11000	110011 0001	001100 1110
D25.7	111 11001	100110 1110	100110 0001
D26.7	111 11010	010110 1110	010110 0001
D27.7	111 11011	110110 0001	001001 1110
D28.7	111 11100	001110 1110	001110 0001
D29.7	111 11101	101110 0001	010001 1110
D30.7	111 11110	011110 0001	100001 1110
D31.7	111 11111	101011 0001	010100 1110

Table C-2: **Valid Control K Characters**

Special Code Name	Bits HGF EDCBA	Current RD – abcdei fghj	Current RD + abcdei fghj
K28.0	000 11100	001111 0100	110000 1011
K28.1	001 11100	001111 1001	110000 0110
K28.2	010 11100	001111 0101	110000 1010
K28.3	011 11100	001111 0011	110000 1100
K28.4	100 11100	001111 0010	110000 1101
K28.5	101 11100	001111 1010	110000 0101
K28.6	110 11100	001111 0110	110000 1001
K28.7(1)	111 11100	001111 1000	110000 0111
K23.7	111 10111	111010 1000	000101 0111
K27.7	111 11011	110110 1000	001001 0111
K29.7	111 11101	101110 1000	010001 0111
K30.7	111 11110	011110 1000	100001 0111

Notes:

1. Used for testing and characterization only.

XILINX®

Appendix D

DRP Address Map of the GTX_DUAL Tile

All attributes are stored as binary values in the DRP table. Some attributes use special mappings from their UCF/HDL values to their binary values. The mappings for these values are shown in Table D-1.

For attributes not listed in Table D-1, use the following rules to determine the binary mapping:

- Attributes with TRUE/FALSE values use "1" to represent TRUE and "0" to represent FALSE.
- Convert integer values to binary.

Table D-1: **Special Attribute Mappings**

Attribute	UCF/HDL Attribute Value	DRP Binary Value
CHAN_BOND_MODE	OFF	00
	MASTER	01
	SLAVE	10
CLK25_DIVIDER	1	000
	2	001
	3	010
	4	011
	5	100
	6	101
	10	110
	12	111

Table D-1: **Special Attribute Mappings *(Cont'd)***

Attribute	UCF/HDL Attribute Value	DRP Binary Value
OOB_CLK_DIVIDER	1	000
	2	001
	3	010
	4	011
	5	100
	6	101
	10	110
	12	111
PLL_DIVSEL_FB	1	10000
	2	00000
	3	00001
	4	00010
	5	00011
PLL_DIVSEL_REF	1	010000
	2	000000
PLL_RXDIVSEL_OUT	1	00
	2	01
	4	10
PLL_TXDIVSEL_OUT	1	00
	2	01
	4	10
RX_LOS_INVALID_INCR	1	000
	2	001
	4	010
	8	011
	16	100
	32	101
	64	110
	128	111

Table D-1: **Special Attribute Mappings *(Cont'd)***

Attribute	UCF/HDL Attribute Value	DRP Binary Value
RX_LOS_THRESHOLD	4	000
	8	001
	16	010
	32	011
	64	100
	128	101
	256	110
	512	111
RX_SLIDE_MODE	PCS	0
	PMA	1
RX_STATUS_FMT	PCIE	0
	SATA	1
RX_XCLK_SEL	RXREC	0
	RXUSR	1
TX_XCLK_SEL	TXOUT	0
	TXUSR	1

DRP Address by Attribute

Table D-2 and Table D-3 list the DRP addresses according to attribute name.

Table D-2: **DRP Address by Attribute**

Attribute	Bit																															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
AC_CAP_DIS_0	49<14>																															
AC_CAP_DIS_1	6<1>																															
ALIGN_COMMA_WORD_0	2b<12>																															
ALIGN_COMMA_WORD_1	24<3>																															
CB2_INH_CC_PERIOD_0	4d<0>	4d<1>	4d<2>	4d<3>																												
CB2_INH_CC_PERIOD_1	3<0>	2<15>	2<14>	2<13>																												
CDR_PH_ADJ_TIME	3c<15>	3d<0>	3d<1>	3d<2>																												
CHAN_BOND_1_MAX_SKEW_0	2c<0>	2b<15>	2b<14>	2b<13>																												
CHAN_BOND_1_MAX_SKEW_1	23<15>	24<0>	24<1>	24<2>																												
CHAN_BOND_2_MAX_SKEW_0	2c<4>	2c<3>	2c<2>	2c<1>																												
CHAN_BOND_2_MAX_SKEW_1	23<11>	23<12>	23<13>	23<14>																												
CHAN_BOND_KEEP_ALIGN_0	4d<4>																															
CHAN_BOND_KEEP_ALIGN_1	2<12>																															

Table D-2: **DRP Address by Attribute *(Cont'd)***

Attribute	Bit																															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
CHAN_BOND_LEVEL_0	2c<7>	2c<6>	2c<5>																													
CHAN_BOND_LEVEL_1	23<8>	23<9>	23<10>																													
CHAN_BOND_MODE_0	2c<9>	2c<8>																														
CHAN_BOND_MODE_1	23<6>	23<7>																														
CHAN_BOND_SEQ_1_1_0	2d<3>	2d<2>	2d<1>	2d<0>	2c<15>	2c<14>	2c<13>	2c<12>	2c<11>	2c<10>																						
CHAN_BOND_SEQ_1_1_1	22<12>	22<13>	22<14>	22<15>	23<0>	23<1>	23<2>	23<3>	23<4>	23<5>																						
CHAN_BOND_SEQ_1_2_0	2d<13>	2d<12>	2d<11>	2d<10>	2d<9>	2d<8>	2d<7>	2d<6>	2d<5>	2d<4>																						
CHAN_BOND_SEQ_1_2_1	22<2>	22<3>	22<4>	22<5>	22<6>	22<7>	22<8>	22<9>	22<10>	22<11>																						
CHAN_BOND_SEQ_1_3_0	2e<7>	2e<6>	2e<5>	2e<4>	2e<3>	2e<2>	2e<1>	2e<0>	2d<15>	2d<14>																						
CHAN_BOND_SEQ_1_3_1	21<8>	21<9>	21<10>	21<11>	21<12>	21<13>	21<14>	21<15>	22<0>	22<1>																						
CHAN_BOND_SEQ_1_4_0	2f<1>	2f<0>	2e<15>	2e<14>	2e<13>	2e<12>	2e<11>	2e<10>	2e<9>	2e<8>																						
CHAN_BOND_SEQ_1_4_1	20<14>	20<15>	21<0>	21<1>	21<2>	21<3>	21<4>	21<5>	21<6>	21<7>																						
CHAN_BOND_SEQ_1_ENABLE_0		2f<5>	2f<4>	2f<3>	2f<2>																											

Table D-2: **DRP Address by Attribute *(Cont'd)***

Attribute	Bit																															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
CHAN_BOND_SEQ_1_ENABLE_1		20<10>	20<11>	20<12>	20<13>																											
CHAN_BOND_SEQ_2_1_0	2f<15>	2f<14>	2f<13>	2f<12>	2f<11>	2f<10>	2f<9>	2f<8>	2f<7>	2f<6>																						
CHAN_BOND_SEQ_2_1_1	20<0>	20<1>	20<2>	20<3>	20<4>	20<5>	20<6>	20<7>	20<8>	20<9>																						
CHAN_BOND_SEQ_2_2_0	30<10>	30<9>	30<8>	30<7>	30<6>	30<5>	30<4>	30<3>	30<1>	30<0>																						
CHAN_BOND_SEQ_2_2_1	1f<5>	1f<6>	1f<7>	1f<8>	1f<9>	1f<10>	1f<11>	1f<12>	1f<14>	1f<15>																						
CHAN_BOND_SEQ_2_3_0	47<9>	47<8>	47<7>	47<6>	47<5>	47<4>	47<3>	47<2>	47<1>	30<11>																						
CHAN_BOND_SEQ_2_3_1	8<6>	8<7>	8<8>	8<9>	8<10>	8<11>	8<12>	8<13>	8<14>	1f<4>																						
CHAN_BOND_SEQ_2_4_0	48<3>	48<2>	48<1>	48<0>	47<15>	47<14>	47<13>	47<12>	47<11>	47<10>																						
CHAN_BOND_SEQ_2_4_1	7<12>	7<13>	7<14>	7<15>	8<0>	8<1>	8<2>	8<3>	8<4>	8<5>																						
CHAN_BOND_SEQ_2_ENABLE_0		39<14>	39<15>	3a<0>	3a<1>																											
CHAN_BOND_SEQ_2_ENABLE_1		16<1>	16<0>	15<15>	15<14>																											
CHAN_BOND_SEQ_2_USE_0	39<13>																															
CHAN_BOND_SEQ_2_USE_1	16<2>																															

Table D-2: **DRP Address by Attribute *(Cont'd)***

Attribute	Bit																															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
CHAN_BOND_SEQ_LEN_0	39<11>	39<12>																														
CHAN_BOND_SEQ_LEN_1	16<4>	16<3>																														
CLK_COR_ADJ_LEN_0	39<9>	39<10>																														
CLK_COR_ADJ_LEN_1	16<6>	16<5>																														
CLK_COR_DET_LEN_0	39<7>	39<8>																														
CLK_COR_DET_LEN_1	16<8>	16<7>																														
CLK_COR_INSERT_IDLE_FLAG_0	39<6>																															
CLK_COR_INSERT_IDLE_FLAG_1	16<9>																															
CLK_COR_KEEP_IDLE_0	39<5>																															
CLK_COR_KEEP_IDLE_1	16<10>																															
CLK_COR_MAX_LAT_0	38<15>	39<0>	39<1>	39<2>	39<3>	39<4>																										
CLK_COR_MAX_LAT_1	17<0>	16<15>	16<14>	16<13>	16<12>	16<11>																										
CLK_COR_MIN_LAT_0	38<9>	38<10>	38<11>	38<12>	38<13>	38<14>																										

Table D-2: **DRP Address by Attribute *(Cont'd)***

Attribute	Bit 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLK_COR_MIN_LAT_1																											17<1>	17<2>	17<3>	17<4>	17<5>	17<6>
CLK_COR_PRECEDENCE_0																																38<8>
CLK_COR_PRECEDENCE_1																																17<7>
CLK_COR_REPEAT_WAIT_0																												38<6>	38<5>	38<4>	38<3>	38<2>
CLK_COR_REPEAT_WAIT_1																												17<9>	17<10>	17<11>	17<12>	17<13>
CLK_COR_SEQ_1_1_0																							38<1>	38<0>	37<15>	37<14>	37<13>	37<12>	37<11>	37<10>	37<9>	37<8>
CLK_COR_SEQ_1_1_1																							17<14>	17<15>	18<0>	18<1>	18<2>	18<3>	18<4>	18<5>	18<6>	18<7>
CLK_COR_SEQ_1_2_0																							37<7>	37<6>	37<5>	37<4>	37<3>	37<2>	37<1>	37<0>	36<15>	36<14>
CLK_COR_SEQ_1_2_1																							18<8>	18<9>	18<10>	18<11>	18<12>	18<13>	18<14>	18<15>	19<0>	19<1>
CLK_COR_SEQ_1_3_0																							36<13>	36<12>	36<11>	36<10>	36<9>	36<8>	36<7>	36<6>	36<5>	36<4>
CLK_COR_SEQ_1_3_1																							19<2>	19<3>	19<4>	19<5>	19<6>	19<7>	19<8>	19<9>	19<10>	19<11>
CLK_COR_SEQ_1_4_0																							36<3>	36<2>	36<1>	36<0>	35<15>	35<14>	35<13>	35<12>	35<11>	35<10>

Table D-2: **DRP Address by Attribute *(Cont'd)***

Attribute	Bit																															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
CLK_COR_SEQ_1_4_1	1a<5>	1a<4>	1a<3>	1a<2>	1a<1>	1a<0>	19<15>	19<14>	19<13>	19<12>																						
CLK_COR_SEQ_1_ENABLE_0		35<6>	35<7>	35<8>	35<9>																											
CLK_COR_SEQ_1_ENABLE_1		1a<9>	1a<8>	1a<7>	1a<6>																											
CLK_COR_SEQ_2_1_0	34<12>	34<13>	34<14>	34<15>	35<0>	35<1>	35<2>	35<3>	35<4>	35<5>																						
CLK_COR_SEQ_2_1_1	1b<3>	1b<2>	1b<1>	1b<0>	1a<15>	1a<14>	1a<13>	1a<12>	1a<11>	1a<10>																						
CLK_COR_SEQ_2_2_0	34<2>	34<3>	34<4>	34<5>	34<6>	34<7>	34<8>	34<9>	34<10>	34<11>																						
CLK_COR_SEQ_2_2_1	1b<13>	1b<12>	1b<11>	1b<10>	1b<9>	1b<8>	1b<7>	1b<6>	1b<5>	1b<4>																						
CLK_COR_SEQ_2_3_0	33<8>	33<9>	33<10>	33<11>	33<12>	33<13>	33<14>	33<15>	34<0>	34<1>																						
CLK_COR_SEQ_2_3_1	1c<7>	1c<6>	1c<5>	1c<4>	1c<3>	1c<2>	1c<1>	1c<0>	1b<15>	1b<14>																						
CLK_COR_SEQ_2_4_0	32<14>	32<15>	33<0>	33<1>	33<2>	33<3>	33<4>	33<5>	33<6>	33<7>																						
CLK_COR_SEQ_2_4_1	1d<1>	1d<0>	1c<15>	1c<14>	1c<13>	1c<12>	1c<11>	1c<10>	1c<9>	1c<8>																						
CLK_COR_SEQ_2_ENABLE_0		32<10>	32<11>	32<12>	32<13>																											

Table D-2: **DRP Address by Attribute *(Cont'd)***

Attribute	Bit																															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
CLK_COR_SEQ_2_ENABLE_1		1d<5>	1d<4>	1d<3>	1d<2>																											
CLK_COR_SEQ_2_USE_0	32<9>																															
CLK_COR_SEQ_2_USE_1	1d<6>																															
CLK_CORRECT_USE_0	38<7>																															
CLK_CORRECT_USE_1	17<8>																															
CLK25_DIVIDER	26<11>	26<10>	26<9>																													
CLKINDC_B	4<3>																															
CLKNORTH_SEL	4<8>																															
CLK_RCV_TRST	9<12>																															
CLKSOUTH_SEL	4<7>																															
CM_TRIM_0	4e<4>	4e<5>																														
CM_TRIM_1	1<12>	1<11>																														
COM_BURST_VAL_0	32<5>	32<6>	32<7>	32<8>																												
COM_BURST_VAL_1	1d<10>	1d<9>	1d<8>	1d<7>																												

Table D-2: **DRP Address by Attribute *(Cont'd)***

Attribute	Bit																															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
COMMA_10B_ENABLE_0	31<11>	31<12>	31<13>	31<14>	31<15>	32<0>	32<1>	32<2>	32<3>	32<4>																						
COMMA_10B_ENABLE_1	1e<4>	1e<3>	1e<2>	1e<1>	1e<0>	1d<15>	1d<14>	1d<13>	1d<12>	1d<11>																						
COMMA_DOUBLE_0	31<10>																															
COMMA_DOUBLE_1	1e<5>																															
DEC_MCOMMA_DETECT_0	31<9>																															
DEC_MCOMMA_DETECT_1	1e<6>																															
DEC_PCOMMA_DETECT_0	31<8>																															
DEC_PCOMMA_DETECT_1	1e<7>																															
DEC_VALID_COMMA_ONLY_0	31<7>																															
DEC_VALID_COMMA_ONLY_1	1e<8>																															
DFE_CAL_TIME	3b<14>	3b<15>	3c<0>	3c<1>	3c<2>																											
DFE_CFG_0	4d<10>	4d<11>	4d<12>	4d<13>	4d<14>	4d<15>	4e<0>	4e<1>	4e<2>	4e<3>																						
DFE_CFG_1	2<6>	2<5>	2<4>	2<3>	2<2>	2<1>	2<0>	1<15>	1<14>	1<13>																						

Table D-2: **DRP Address by Attribute *(Cont'd)***

Attribute	Bit																															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
GEARBOX_ENDEC_0	4d<5>	4d<6>	4d<7>																													
GEARBOX_ENDEC_1	2<11>	2<10>	2<9>																													
MCOMMA_10B_VALUE_0	30<13>	30<14>	30<15>	31<0>	31<1>	31<2>	31<3>	31<4>	31<5>	31<6>																						
MCOMMA_10B_VALUE_1	1f<2>	1f<1>	1f<0>	1e<15>	1e<14>	1e<13>	1e<12>	1e<11>	1e<10>	1e<9>																						
MCOMMA_DETECT_0	30<12>																															
MCOMMA_DETECT_1	1f<3>																															
OOB_CLK_DIVIDER	26<14>	26<13>	26<12>																													
OOBDETECT_THRESHOLD_0	3a<3>	3a<4>	3a<5>																													
OOBDETECT_THRESHOLD_1	15<12>	15<11>	15<10>																													
OVERSAMPLE_MODE	26<15>																															
PCI_EXPRESS_MODE_0	46<15>																															
PCI_EXPRESS_MODE_1	9<0>																															
PCOMMA_10B_VALUE_0	46<5>	46<6>	46<7>	46<8>	46<9>	46<10>	46<11>	46<12>	46<13>	46<14>																						

Table D-2: **DRP Address by Attribute *(Cont'd)***

Attribute	Bit																															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
PCOMMA_10B_VALUE_1	9<10>	9<9>	9<8>	9<7>	9<6>	9<5>	9<4>	9<3>	9<2>	9<1>																						
PCOMMA_DETECT_0	46<4>																															
PCOMMA_DETECT_1	9<11>																															
PLL_COM_CFG	28<3>	28<2>	28<1>	28<0>	27<15>	27<14>	27<13>	27<12>	27<11>	27<10>	27<9>	27<8>	27<7>	27<6>	27<5>	27<4>	27<3>	27<2>	27<1>	27<0>	13<15>	14<0>	14<1>	14<2>								
PLL_CP_CFG	28<11>	28<10>	28<9>	28<8>	28<7>	28<6>	28<5>	28<4>																								
PLL_DIVSEL_FB	29<0>	28<15>	28<14>	28<13>	28<12>																											
PLL_DIVSEL_REF	4<9>	4<10>	4<11>	4<12>	4<13>	4<14>																										
PLL_FB_DCCEN	30<2>																															
PLL_RXDIVSEL_OUT_0	46<2>	46<3>																														
PLL_RXDIVSEL_OUT_1	0a<0>	9<15>																														
PLL_SATA_0	46<1>																															
PLL_SATA_1	9<14>																															
PLL_STARTUP_EN	4a<10>																															

Table D-2: **DRP Address by Attribute *(Cont'd)***

Bit	Attribute	PLL_TDCC_CFG[2:0]	PLL_TXDIVSEL_OUT_0	PLL_TXDIVSEL_OUT_1	PLL_LKDET_CFG	PMA_CDR_SCAN_0	PMA_CDR_SCAN_1	PMA_COM_CFG	PMA_RX_CFG_0	PMA_RX_CFG_1	PMA_RXSYNC_CFG_0	PMA_RXSYNC_CFG_1	PMA_TX_CFG_0	PMA_TX_CFG_1
31								4a<6>						
30								4a<5>						
29								4a<4>						
28								4a<3>						
27								4a<2>						
26						45<14>	0a<1>	5<9>						
25						45<13>	0a<2>	5<10>						
24						45<12>	0a<3>	5<11>	6<3>	49<12>				
23						45<11>	0a<4>	5<12>	7<6>	48<9>				
22						45<10>	0a<5>	5<13>	49<8>	6<7>				
21						45<9>	0a<6>	5<8>	49<7>	6<8>				
20						45<8>	0a<7>	4a<7>	49<6>	6<9>				
19						45<7>	0a<8>	3a<6>	49<5>	6<10>			4f<9>	0<7>
18						45<6>	0a<9>	24<4>	49<4>	6<11>			4f<8>	0<8>
17						45<5>	0a<10>	24<5>	49<3>	6<12>			4f<7>	0<9>
16						45<4>	0a<11>	24<6>	49<2>	6<13>			4f<6>	0<10>
15						45<3>	0a<12>	24<7>	49<1>	6<14>			4f<5>	0<11>
14						45<2>	0a<13>	24<8>	49<0>	6<15>			4f<4>	0<12>
13						45<1>	0a<14>	24<9>	48<15>	7<0>			4f<3>	0<13>
12						45<0>	0a<15>	24<10>	6<2>	49<13>			4f<2>	0<14>
11						44<15>	0b<0>	24<11>	6<6>	49<9>			4f<1>	0<15>
10						44<14>	0b<1>	24<12>	48<8>	7<7>			4f<0>	1<0>
9						44<13>	0b<2>	2b<11>	48<7>	7<8>			4e<15>	1<1>
8						44<12>	0b<3>	15<13>	48<6>	7<9>			4e<14>	1<2>
7						44<11>	0b<4>	2b<3>	48<5>	7<10>			4e<13>	1<3>
6						44<10>	0b<5>	2b<4>	48<4>	7<11>	4b<15>	0<0>	4e<12>	1<4>
5						44<9>	0b<6>	2b<5>	48<13>	7<2>	4b<14>	0<1>	4e<11>	1<5>
4						44<8>	0b<7>	2b<7>	48<12>	7<3>	4b<13>	0<2>	4e<10>	1<6>
3						44<7>	0b<8>	2b<8>	48<11>	7<4>	4b<12>	0<3>	4e<9>	1<7>
2	4b<0>			4<15>	44<6>	0b<9>	2b<9>	48<10>	7<5>	4b<11>	0<4>	4e<8>	1<8>	
1	4a<14>	46<0>	5<3>	5<0>	44<5>	0b<10>	2b<10>	6<4>	49<11>	4b<10>	0<5>	4e<7>	1<9>	
0	4a<15>	45<15>	5<4>	5<1>	44<4>	0b<11>	2b<6>	6<5>	49<10>	4b<9>	0<6>	4e<6>	1<10>	

Table D-2: **DRP Address by Attribute *(Cont'd)***

Attribute	Bit																															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
PRBS_ERR_THRESHOLD_0	42<4>	42<5>	42<6>	42<7>	42<8>	42<9>	42<10>	42<11>	42<12>	42<13>	42<14>	42<15>	43<0>	43<1>	43<2>	43<3>	43<4>	43<5>	43<6>	43<7>	43<8>	43<9>	43<10>	43<11>	43<12>	43<13>	43<14>	43<15>	44<0>	44<1>	44<2>	44<3>
PRBS_ERR_THRESHOLD_1	0d<11>	0d<10>	0d<9>	0d<8>	0d<7>	0d<6>	0d<5>	0d<4>	0d<3>	0d<2>	0d<1>	0d<0>	0c<15>	0c<14>	0c<13>	0c<12>	0c<11>	0c<10>	0c<9>	0c<8>	0c<7>	0c<6>	0c<5>	0c<4>	0c<3>	0c<2>	0c<1>	0c<0>	0b<15>	0b<14>	0b<13>	0b<12>
RCV_TERM_GND_0	4a<0>																															
RCV_TERM_GND_1	5<15>																															
RCV_TERM_VTTRX_0	4a<1>																															
RCV_TERM_VTTRX_1	5<14>																															
REFCLK_SEL[2:0]	4<6>	4<5>	4<4>																													
RX_BUFFER_USE_0	42<3>																															
RX_BUFFER_USE_1	0d<12>																															
RX_CDR_FORCE_ROTATE_0	48<14>																															
RX_CDR_FORCE_ROTATE_1	7<1>																															
RX_DECODE_SEQ_MATCH_0	42<2>																															
RX_DECODE_SEQ_MATCH_1	0d<13>																															

Table D-2: **DRP Address by Attribute *(Cont'd)***

Attribute	Bit																															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
RX_EN_IDLE_HOLD_CDR	3a<9>																															
RX_EN_IDLE_HOLD_DFE_0	4c<1>																															
RX_EN_IDLE_HOLD_DFE_1	3<14>																															
RX_EN_IDLE_RESET_BUF_0	4c<3>																															
RX_EN_IDLE_RESET_BUF_1	3<12>																															
RX_EN_IDLE_RESET_FR	3a<10>																															
RX_EN_IDLE_RESET_PH	3a<2>																															
RX_IDLE_HI_CNT_0	4c<6>	4c<7>	4c<8>	4c<9>																												
RX_IDLE_HI_CNT_1	3<9>	3<8>	3<7>	3<6>																												
RX_IDLE_LO_CNT_0	4c<12>	4c<13>	4c<14>	4c<15>																												
RX_IDLE_LO_CNT_13	3<4>	3<3>	3<2>	3<1>																												
RX_LOS_INVALID_INCR_0	41<15>	42<0>	42<1>																													
RX_LOS_INVALID_INCR_1	0e<0>	0d<15>	0d<14>																													

Table D-2: **DRP Address by Attribute *(Cont'd)***

Attribute	Bit																															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
RX_LOS_THRESHOLD_0	41<11>	41<12>	41<13>																													
RX_LOS_THRESHOLD_1	0e<4>	0e<3>	0e<2>																													
RX_LOSS_OF_SYNC_FSM_0	41<14>																															
RX_LOSS_OF_SYNC_FSM_1	0e<1>																															
RX_SLIDE_MODE_0	41<10>																															
RX_SLIDE_MODE_1	0e<5>																															
RX_STATUS_FMT_0	41<9>																															
RX_STATUS_FMT_1	0e<6>																															
RX_XCLK_SEL_0	41<8>																															
RX_XCLK_SEL_1	0e<7>																															
RXGEARBOX_USE_0	4d<8>																															
RXGEARBOX_USE_1	2<8>																															
SATA_BURST_VAL_0	41<5>	41<6>	41<7>																													

Table D-2: **DRP Address by Attribute *(Cont'd)***

Attribute	Bit																															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SATA_BURST_VAL_1	0e<10>	0e<9>	0e<8>																													
SATA_IDLE_VAL_0	41<2>	41<3>	41<4>																													
SATA_IDLE_VAL_1	0e<13>	0e<12>	0e<11>																													
SATA_MAX_BURST_0	40<12>	40<13>	40<14>	40<15>	41<0>	41<1>																										
SATA_MAX_BURST_1	0f<3>	0f<2>	0f<1>	0f<0>	0e<15>	0e<14>																										
SATA_MAX_INIT_0	40<6>	40<7>	40<8>	40<9>	40<10>	40<11>																										
SATA_MAX_INIT_1	0f<9>	0f<8>	0f<7>	0f<6>	0f<5>	0f<4>																										
SATA_MAX_WAKE_0	40<0>	40<1>	40<2>	40<3>	40<4>	40<5>																										
SATA_MAX_WAKE_1	0f<15>	0f<14>	0f<13>	0f<12>	0f<11>	0f<10>																										
SATA_MIN_BURST_0	3f<10>	3f<11>	3f<12>	3f<13>	3f<14>	3f<15>																										
SATA_MIN_BURST_1	10<5>	10<4>	10<3>	10<2>	10<1>	10<0>																										
SATA_MIN_INIT_0	3f<4>	3f<5>	3f<6>	3f<7>	3f<8>	3f<9>																										
SATA_MIN_INIT_1	10<11>	10<10>	10<9>	10<8>	10<7>	10<6>																										

Table D-2: **DRP Address by Attribute *(Cont'd)***

Attribute	Bit																															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SATA_MIN_WAKE_0	3e<14>	3e<15>	3f<0>	3f<1>	3f<2>	3f<3>										3e<12>																
SATA_MIN_WAKE_1	11<1>	11<0>	10<15>	10<14>	10<13>	10<12>										3e<11>																
TERMINATION_CTRL	29<5>	29<4>	29<3>	29<2>	29<1>											3e<10>																
TERMINATION_IMP_0	3e<13>															3e<9>																
TERMINATION_IMP_1	11<2>															3e<8>																
TERMINATION_OVRD	29<6>															3e<7>																
TRANS_TIME_FROM_P2_0	3d<13>	3d<14>	3d<15>	3e<0>	3e<1>	3e<2>	3e<3>	3e<4>	3e<5>	3e<6>	3e<7>	3e<8>	3e<9>	3e<10>	3e<11>	3e<12>																
TRANS_TIME_FROM_P2_1	12<2>	12<1>	12<0>	11<15>	11<14>	11<13>	11<12>	11<11>	11<10>	11<9>	11<8>	11<7>	11<6>	11<5>	11<4>	11<3>																
TRANS_TIME_NON_P2_0	3c<13>	3c<14>	3c<15>	3d<0>	3d<1>	3d<2>	3d<3>	3d<4>	3d<5>	3d<6>	3d<7>	3d<8>	3d<9>	3d<10>	3d<11>	3d<12>																
TRANS_TIME_NON_P2_1	13<2>	13<1>	13<0>	12<15>	12<14>	12<13>	12<12>	12<11>	12<10>	12<9>	12<8>	12<7>	12<6>	12<5>	12<4>	12<3>																
TRANS_TIME_TO_P2_0	3b<13>	3b<14>	3b<15>	3c<0>	3c<1>	3c<2>	3c<3>	3c<4>	3c<5>	3c<6>	3c<7>	3c<8>	3c<9>	3c<10>	3c<11>	3c<12>																
TRANS_TIME_TO_P2_1	14<2>	14<1>	14<0>	13<15>	13<14>	13<13>	13<12>	13<11>	13<10>	13<9>	13<8>	13<7>	13<6>	13<5>	13<4>	13<3>																

Table D-2: **DRP Address by Attribute *(Cont'd)***

Attribute	Bit																															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
TX_BUFFER_USE_0	3b<12>																															
TX_BUFFER_USE_1	14<3>																															
TX_DETECT_RX_CFG_0	3a<14>	3a<15>	3b<0>	3b<1>	3b<2>	3b<3>	3b<4>	3b<5>	3b<6>	3b<7>	3b<8>	3b<9>	3b<10>	3b<11>																		
TX_DETECT_RX_CFG_1	15<1>	15<0>	14<15>	14<14>	14<13>	14<12>	14<11>	14<10>	14<9>	14<8>	14<7>	14<6>	14<5>	14<4>																		
TX_IDLE_DELAY_0	4b<6>	4b<7>	4b<8>																													
TX_IDLE_DELAY_1	4<1>	4<0>	3<15>																													
TX_XCLK_SEL_0	3a<8>																															
TX_XCLK_SEL_1	15<7>																															
TXGEARBOX_USE_0	4d<9>																															
TXGEARBOX_USE_1	2<7>																															
TXOUTCLK_SEL_0	3a<7>																															
TXOUTCLK_SEL_1	15<8>																															

Table D-2: **DRP Address by Attribute *(Cont'd)***

Attribute	Bit																														
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
TXRX_INVERT_0	3a<11>	3a<12>	3a<13>																													
TXRX_INVERT_1	15<4>	15<3>	15<2>																													

Table D-3: **DRP Address by Attribute (Bits 32:68)**

Attribute	Bit																																			
	32	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**	**45**	**46**	**47**	**48**	**49**	**50**	**51**	**52**	**53**	**54**	**55**	**56**	**57**	**58**	**59**	**60**	**61**	**62**	**63**	**64**	**65**	**66**	**67**	**68**
PMA_COM_CFG	29<7>	29<14>	26<1>	29<15>	26<2>	29<13>	26<4>	29<12>	26<3>	29<11>	26<5>	29<10>	26<6>	29<9>	26<7>	29<8>	26<8>	2a<15>	25<2>	2a<14>	25<1>	2a<0>	25<15>	2a<1>	26<0>	2a<4>	2a<3>	2a<5>	2a<2>	25<12>	25<13>	25<11>	25<14>	2a<7>	25<9>	2a<6>	25<10>

DRP Address by Bit Location

This section lists the attributes according to DRP address and bit location:

Table D-4: **DRP Addresses 0x00 through 0x07**

Bit	Address							
	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07
0	PMA_RXSYNC_CFG_1[6]	PMA_TX_CFG_1[10]	DFE_CFG_1[6]	CB2_INH_CC_PERIOD_1[0]	TX_IDLE_DELAY_1[1]	PLL_LKDET_CFG[1]	Do Not Modify	PMA_RX_CFG_1[13]
1	PMA_RXSYNC_CFG_1[5]	PMA_TX_CFG_1[9]	DFE_CFG_1[5]	RX_IDLE_LO_CNT_1[3]	TX_IDLE_DELAY_1[0]	PLL_LKDET_CFG[0]	AC_CAP_DIS_1	RX_CDR_FORCE_ROTATE_1
2	PMA_RXSYNC_CFG_1[4]	PMA_TX_CFG_1[8]	DFE_CFG_1[4]	RX_IDLE_LO_CNT_1[2]	Do Not Modify	Do Not Modify	PMA_RX_CFG_0[12]	PMA_RX_CFG_1[5]
3	PMA_RXSYNC_CFG_1[3]	PMA_TX_CFG_1[7]	DFE_CFG_1[3]	RX_IDLE_LO_CNT_1[1]	CLKINDC_B	PLL_TXDIVSEL_OUT_1[1]	PMA_RX_CFG_0[24]	PMA_RX_CFG_1[4]
4	PMA_RXSYNC_CFG_1[2]	PMA_TX_CFG_1[6]	DFE_CFG_1[2]	RX_IDLE_LO_CNT_1[0]	REFCLK_SEL[2]	PLL_TXDIVSEL_OUT_1[0]	PMA_RX_CFG_0[1]	PMA_RX_CFG_1[3]
5	PMA_RXSYNC_CFG_1[1]	PMA_TX_CFG_1[5]	DFE_CFG_1[1]	Do Not Modify	REFCLK_SEL[1]	Do Not Modify	PMA_RX_CFG_0[0]	PMA_RX_CFG_1[2]
6	PMA_RXSYNC_CFG_1[0]	PMA_TX_CFG_1[4]	DFE_CFG_1[0]	RX_IDLE_HI_CNT_1[3]	REFCLK_SEL[0]	Do Not Modify	PMA_RX_CFG_0[11]	PMA_RX_CFG_0[23]
7	PMA_TX_CFG_1[19]	PMA_TX_CFG_1[3]	TXGEARBOX_USE_1	RX_IDLE_HI_CNT_1[2]	CLKSOUTH_SEL	Do Not Modify	PMA_RX_CFG_1[22]	PMA_RX_CFG_1[10]
8	PMA_TX_CFG_1[18]	PMA_TX_CFG_1[2]	RXGEARBOX_USE_1	RX_IDLE_HI_CNT_1[1]	CLKNORTH_SEL	PMA_COM_CFG[21]	PMA_RX_CFG_1[21]	PMA_RX_CFG_1[9]
9	PMA_TX_CFG_1[17]	PMA_TX_CFG_1[1]	GEARBOX_ENDEC_1[2]	RX_IDLE_HI_CNT_1[0]	PLL_DIVSEL_REF[0]	PMA_COM_CFG[26]	PMA_RX_CFG_1[20]	PMA_RX_CFG_1[8]
10	PMA_TX_CFG_1[16]	PMA_TX_CFG_1[0]	GEARBOX_ENDEC_1[1]	Do Not Modify	PLL_DIVSEL_REF[1]	PMA_COM_CFG[25]	PMA_RX_CFG_1[19]	PMA_RX_CFG_1[7]
11	PMA_TX_CFG_1[15]	CM_TRIM_1[1]	GEARBOX_ENDEC_1[0]	Do Not Modify	PLL_DIVSEL_REF[2]	PMA_COM_CFG[24]	PMA_RX_CFG_1[18]	PMA_RX_CFG_1[6]
12	PMA_TX_CFG_1[14]	CM_TRIM_1[0]	CHAN_BOND_KEEP_ALIGN_1	RX_EN_IDLE_RESET_BUF_1	PLL_DIVSEL_REF[3]	PMA_COM_CFG[23]	PMA_RX_CFG_1[17]	CHAN_BOND_SEQ_2_4_1[0]
13	PMA_TX_CFG_1[13]	DFE_CFG_1[9]	CB2_INH_CC_PERIOD_1[3]	Do Not Modify	PLL_DIVSEL_REF[4]	PMA_COM_CFG[22]	PMA_RX_CFG_1[16]	CHAN_BOND_SEQ_2_4_1[1]
14	PMA_TX_CFG_1[12]	DFE_CFG_1[8]	CB2_INH_CC_PERIOD_1[2]	RX_EN_IDLE_HOLD_DFE_1	PLL_DIVSEL_REF[5]	RCV_TERM_VTTRX_1	PMA_RX_CFG_1[15]	CHAN_BOND_SEQ_2_4_1[2]
15	PMA_TX_CFG_1[11]	DFE_CFG_1[7]	CB2_INH_CC_PERIOD_1[1]	TX_IDLE_DELAY_1[2]	PLL_LKDET_CFG[2]	RCV_TERM_GND_1	PMA_RX_CFG_1[14]	CHAN_BOND_SEQ_2_4_1[3]

Table D-5: **DRP Addresses 0x08 through 0x0F**

Bit	Address							
	0x08	0x09	0x0A	0x0B	0x0C	0x0D	0x0E	0x0F
0	CHAN_BOND_SEQ_2_4_1[4]	PCI_EXPRESS_MODE_1	PLL_RXDIVSEL_OUT_1[0]	PMA_CDR_SCAN_1[11]	PRBS_ERR_THRESHOLD_1[27]	PRBS_ERR_THRESHOLD_1[11]	RX_LOS_INVALID_INCR_1[0]	SATA_MAX_BURST_1[3]
1	CHAN_BOND_SEQ_2_4_1[5]	PCOMMA_10B_VALUE_1[9]	PMA_CDR_SCAN_1[26]	PMA_CDR_SCAN_1[10]	PRBS_ERR_THRESHOLD_1[26]	PRBS_ERR_THRESHOLD_1[10]	RX_LOSS_OF_SYNC_FSM_1	SATA_MAX_BURST_1[2]
2	CHAN_BOND_SEQ_2_4_1[6]	PCOMMA_10B_VALUE_1[8]	PMA_CDR_SCAN_1[25]	PMA_CDR_SCAN_1[9]	PRBS_ERR_THRESHOLD_1[25]	PRBS_ERR_THRESHOLD_1[9]	RX_LOS_THRESHOLD_1[2]	SATA_MAX_BURST_1[1]
3	CHAN_BOND_SEQ_2_4_1[7]	PCOMMA_10B_VALUE_1[7]	PMA_CDR_SCAN_1[24]	PMA_CDR_SCAN_1[8]	PRBS_ERR_THRESHOLD_1[24]	PRBS_ERR_THRESHOLD_1[8]	RX_LOS_THRESHOLD_1[1]	SATA_MAX_BURST_1[0]
4	CHAN_BOND_SEQ_2_4_1[8]	PCOMMA_10B_VALUE_1[6]	PMA_CDR_SCAN_1[23]	PMA_CDR_SCAN_1[7]	PRBS_ERR_THRESHOLD_1[23]	PRBS_ERR_THRESHOLD_1[7]	RX_LOS_THRESHOLD_1[0]	SATA_MAX_INIT_1[5]
5	CHAN_BOND_SEQ_2_4_1[9]	PCOMMA_10B_VALUE_1[5]	PMA_CDR_SCAN_1[22]	PMA_CDR_SCAN_1[6]	PRBS_ERR_THRESHOLD_1[22]	PRBS_ERR_THRESHOLD_1[6]	RX_SLIDE_MODE_1	SATA_MAX_INIT_1[4]
6	CHAN_BOND_SEQ_2_3_1[0]	PCOMMA_10B_VALUE_1[4]	PMA_CDR_SCAN_1[21]	PMA_CDR_SCAN_1[5]	PRBS_ERR_THRESHOLD_1[21]	PRBS_ERR_THRESHOLD_1[5]	RX_STATUS_FMT_1	SATA_MAX_INIT_1[3]
7	CHAN_BOND_SEQ_2_3_1[1]	PCOMMA_10B_VALUE_1[3]	PMA_CDR_SCAN_1[20]	PMA_CDR_SCAN_1[4]	PRBS_ERR_|THRESHOLD_1[20]	PRBS_ERR_THRESHOLD_1[4]	RX_XCLK_SEL_1	SATA_MAX_INIT_1[2]
8	CHAN_BOND_SEQ_2_3_1[2]	PCOMMA_10B_VALUE_1[2]	PMA_CDR_SCAN_1[19]	PMA_CDR_SCAN_1[3]	PRBS_ERR_THRESHOLD_1[19]	PRBS_ERR_THRESHOLD_1[3]	SATA_BURST_VAL_1[2]	SATA_MAX_INIT_1[1]
9	CHAN_BOND_SEQ_2_3_1[3]	PCOMMA_10B_VALUE_1[1]	PMA_CDR_SCAN_1[18]	PMA_CDR_SCAN_1[2]	PRBS_ERR_THRESHOLD_1[18]	PRBS_ERR_THRESHOLD_1[2]	SATA_BURST_VAL_1[1]	SATA_MAX_INIT_1[0]
10	CHAN_BOND_SEQ_2_3_1[4]	PCOMMA_10B_VALUE_1[0]	PMA_CDR_SCAN_1[17]	PMA_CDR_SCAN_1[1]	PRBS_ERR_THRESHOLD_1[17]	PRBS_ERR_THRESHOLD_1[1]	SATA_BURST_VAL_1[0]	SATA_MAX_WAKE_1[5]
11	CHAN_BOND_SEQ_2_3_1[5]	PCOMMA_DETECT_1	PMA_CDR_SCAN_1[16]	PMA_CDR_SCAN_1[0]	PRBS_ERR_THRESHOLD_1[16]	PRBS_ERR_THRESHOLD_1[0]	SATA_IDLE_VAL_1[2]	SATA_MAX_WAKE_1[4]
12	CHAN_BOND_SEQ_2_3_1[6]	CLKRCV_TRST	PMA_CDR_SCAN_1[15]	PRBS_ERR_THRESHOLD_1[31]	PRBS_ERR_THRESHOLD_1[15]	RX_BUFFER_USE_1	SATA_IDLE_VAL_1[1]	SATA_MAX_WAKE_1[3]
13	CHAN_BOND_SEQ_2_3_1[7]	Do Not Modify	PMA_CDR_SCAN_1[14]	PRBS_ERR_THRESHOLD_1[30]	PRBS_ERR_THRESHOLD_1[14]	RX_DECODE_SEQ_MATCH_1	SATA_IDLE_VAL_1[0]	SATA_MAX_WAKE_1[2]
14	CHAN_BOND_SEQ_2_3_1[8]	PLL_SATA_1	PMA_CDR_SCAN_1[13]	PRBS_ERR_THRESHOLD_1[29]	PRBS_ERR_THRESHOLD_1[13]	RX_LOS_INVALID_INCR_1[2]	SATA_MAX_BURST_1[5]	SATA_MAX_WAKE_1[1]
15	Do Not Modify	PLL_RXDIVSEL_OUT_1[1]	PMA_CDR_SCAN_1[12]	PRBS_ERR_THRESHOLD_1[28]	PRBS_ERR_THRESHOLD_1[12]	RX_LOS_INVALID_INCR_1[1]	SATA_MAX_BURST_1[4]	SATA_MAX_WAKE_1[0]

Table D-6: **DRP Addresses 0x10 through 0x17**

Bit	Address							
	0x10	0x11	0x12	0x13	0x14	0x15	0x16	0x17
0	SATA_MIN_BURST_1[5]	SATA_MIN_WAKE_1[1]	Do Not Modify	Do Not Modify	PLL_COM_CFG[21]	TX_DETECT_RX_CFG_1[1]	CHAN_BOND_SEQ_2_ENABLE_1[2]	CLK_COR_MAX_LAT_1[0]
1	SATA_MIN_BURST_1[4]	SATA_MIN_WAKE_1[0]	Do Not Modify	Do Not Modify	PLL_COM_CFG[22]	TX_DETECT_RX_CFG_1[0]	CHAN_BOND_SEQ_2_ENABLE_1[1]	CLK_COR_MIN_LAT_1[5]
2	SATA_MIN_BURST_1[3]	TERMINATION_IMP_1	Do Not Modify	Do Not Modify	PLL_COM_CFG[23]	TXRX_INVERT_1[2]	CHAN_BOND_SEQ_2_USE_1	CLK_COR_MIN_LAT_1[4]
3	SATA_MIN_BURST_1[2]	TRANS_TIME_FROM_P2_1[11]	TRANS_TIME_NON_P2_1[7]	TRANS_TIME_TO_P2_1[9]	TX_BUFFER_USE_1	TXRX_INVERT_1[1]	CHAN_BOND_SEQ_LEN_1[1]	CLK_COR_MIN_LAT_1[3]
4	SATA_MIN_BURST_1[1]	TRANS_TIME_FROM_P2_1[10]	TRANS_TIME_NON_P2_1[6]	TRANS_TIME_TO_P2_1[8]	TX_DETECT_RX_CFG_1[13]	TXRX_INVERT_1[0]	CHAN_BOND_SEQ_LEN_1[0]	CLK_COR_MIN_LAT_1[2]
5	SATA_MIN_BURST_1[0]	TRANS_TIME_FROM_P2_1[9]	TRANS_TIME_NON_P2_1[5]	TRANS_TIME_TO_P2_1[7]	TX_DETECT_RX_CFG_1[12]	Do Not Modify	CLK_COR_ADJ_LEN_1[1]	CLK_COR_MIN_LAT_1[1]
6	SATA_MIN_INIT_1[5]	TRANS_TIME_FROM_P2_1[8]	TRANS_TIME_NON_P2_1[4]	TRANS_TIME_TO_P2_1[6]	TX_DETECT_RX_CFG_1[11]	Do Not Modify	CLK_COR_ADJ_LEN_1[0]	CLK_COR_MIN_LAT_1[0]
7	SATA_MIN_INIT_1[4]	TRANS_TIME_FROM_P2_1[7]	TRANS_TIME_NON_P2_1[3]	TRANS_TIME_TO_P2_1[5]	TX_DETECT_RX_CFG_1[10]	TX_XCLK_SEL_1	CLK_COR_DET_LEN_1[1]	CLK_COR_PRECEDENCE_1
8	SATA_MIN_INIT_1[3]	TRANS_TIME_FROM_P2_1[6]	TRANS_TIME_NON_P2_1[2]	TRANS_TIME_TO_P2_1[4]	TX_DETECT_RX_CFG_1[9]	TXOUTCLK_SEL_1	CLK_COR_DET_LEN_1[0]	CLK_CORRECT_USE_1
9	SATA_MIN_INIT_1[2]	TRANS_TIME_FROM_P2_1[5]	TRANS_TIME_NON_P2_1[1]	TRANS_TIME_TO_P2_1[3]	TX_DETECT_RX_CFG_1[8]	Do Not Modify	CLK_COR_INSERT_IDLE_FLAG_1	CLK_COR_REPEAT_WAIT_1[4]
10	SATA_MIN_INIT_1[1]	TRANS_TIME_FROM_P2_1[4]	TRANS_TIME_NON_P2_1[0]	TRANS_TIME_TO_P2_1[2]	TX_DETECT_RX_CFG_1[7]	OOBDETECT_THRESHOLD_1[2]	CLK_COR_KEEP_IDLE_1	CLK_COR_REPEAT_WAIT_1[3]
11	SATA_MIN_INIT_1[0]	TRANS_TIME_FROM_P2_1[3]	Do Not Modify	TRANS_TIME_TO_P2_1[1]	TX_DETECT_RX_CFG_1[6]	OOBDETECT_THRESHOLD_1[1]	CLK_COR_MAX_LAT_1[5]	CLK_COR_REPEAT_WAIT_1[2]
12	SATA_MIN_WAKE_1[5]	TRANS_TIME_FROM_P2_1[2]	Do Not Modify	TRANS_TIME_TO_P2_1[0]	TX_DETECT_RX_CFG_1[5]	OOBDETECT_THRESHOLD_1[0]	CLK_COR_MAX_LAT_1[4]	CLK_COR_REPEAT_WAIT_1[1]
13	SATA_MIN_WAKE_1[4]	TRANS_TIME_FROM_P2_1[1]	Do Not Modify	Do Not Modify	TX_DETECT_RX_CFG_1[4]	PMA_COM_CFG[8]	CLK_COR_MAX_LAT_1[3]	CLK_COR_REPEAT_WAIT_1[0]
14	SATA_MIN_WAKE_1[3]	TRANS_TIME_FROM_P2_1[0]	Do Not Modify	Do Not Modify	TX_DETECT_RX_CFG_1[3]	CHAN_BOND_SEQ_2_ENABLE_1[4]	CLK_COR_MAX_LAT_1[2]	CLK_COR_SEQ_1_1_1[9]
15	SATA_MIN_WAKE_1[2]	Do Not Modify	Do Not Modify	PLL_COM_CFG[20]	TX_DETECT_RX_CFG_1[2]	CHAN_BOND_SEQ_2_ENABLE_1[3]	CLK_COR_MAX_LAT_1[1]	CLK_COR_SEQ_1_1_1[8]

Table D-7: **DRP Addresses 0x18 through 0x1F**

Bit	Address							
	0x18	0x19	0x1A	0x1B	0x1C	0x1D	0x1E	0x1F
0	CLK_COR_SEQ_1_1_1[7]	CLK_COR_SEQ_1_2_1[1]	CLK_COR_SEQ_1_4_1[5]	CLK_COR_SEQ_2_1_1[3]	CLK_COR_SEQ_2_3_1[7]	CLK_COR_SEQ_2_4_1[1]	COMMA_10B_ENABLE_1[4]	MCOMMA_10B_VALUE_1[2]
1	CLK_COR_SEQ_1_1_1[6]	CLK_COR_SEQ_1_2_1[0]	CLK_COR_SEQ_1_4_1[4]	CLK_COR_SEQ_2_1_1[2]	CLK_COR_SEQ_2_3_1[6]	CLK_COR_SEQ_2_4_1[0]	COMMA_10B_ENABLE_1[3]	MCOMMA_10B_VALUE_1[1]
2	CLK_COR_SEQ_1_1_1[5]	CLK_COR_SEQ_1_3_1[9]	CLK_COR_SEQ_1_4_1[3]	CLK_COR_SEQ_2_1_1[1]	CLK_COR_SEQ_2_3_1[5]	CLK_COR_SEQ_2_ENABLE_1[4]	COMMA_10B_ENABLE_1[2]	MCOMMA_10B_VALUE_1[0]
3	CLK_COR_SEQ_1_1_1[4]	CLK_COR_SEQ_1_3_1[8]	CLK_COR_SEQ_1_4_1[2]	CLK_COR_SEQ_2_1_1[0]	CLK_COR_SEQ_2_3_1[4]	CLK_COR_SEQ_2_ENABLE_1[3]	COMMA_10B_ENABLE_1[1]	MCOMMA_DETECT_1
4	CLK_COR_SEQ_1_1_1[3]	CLK_COR_SEQ_1_3_1[7]	CLK_COR_SEQ_1_4_1[1]	CLK_COR_SEQ_2_2_1[9]	CLK_COR_SEQ_2_3_1[3]	CLK_COR_SEQ_2 ENABLE_1[2]	COMMA_10B_ENABLE_1[0]	CHAN_BOND_SEQ_2_3_1[9]
5	CLK_COR_SEQ_1_1_1[2]	CLK_COR_SEQ_1_3_1[6]	CLK_COR_SEQ_1_4_1[0]	CLK_COR_SEQ_2_2_1[8]	CLK_COR_SEQ_2_3_1[2]	CLK_COR_SEQ_2_ENABLE_1[1]	COMMA_DOUBLE_1	CHAN_BOND_SEQ_2_2_1[0]
6	CLK_COR_SEQ_1_1_1[1]	CLK_COR_SEQ_1_3_1[5]	CLK_COR_SEQ_1_ENABLE_1[4]	CLK_COR_SEQ_2_2_1[7]	CLK_COR_SEQ_2_3_1[1]	CLK_COR_SEQ_2_USE_1	DEC_MCOMMA_DETECT_1	CHAN_BOND_SEQ_2_2_1[1]
7	CLK_COR_SEQ_1_1_1[0]	CLK_COR_SEQ_1_3_1[4]	CLK_COR_SEQ_1_ ENABLE_1[3]	CLK_COR_SEQ_2_2_1[6]	CLK_COR_SEQ_2_3_1[0]	COM_BURST_VAL_1[3]	DEC_PCOMMA_DETECT_1	CHAN_BOND_SEQ_2_2_1[2]
8	CLK_COR_SEQ_1_2_1[9]	CLK_COR_SEQ_1_3_1[3]	CLK_COR_SEQ_1_ ENABLE_1[2]	CLK_COR_SEQ_2_2_1[5]	CLK_COR_SEQ_2_4_1[9]	COM_BURST_VAL_1[2]	DEC_VALID_COMMA_ONLY_1	CHAN_BOND_SEQ_2_2_1[3]
9	CLK_COR_SEQ_1_2_1[8]	CLK_COR_SEQ_1_3_1[2]	CLK_COR_SEQ_1_ ENABLE_1[1]	CLK_COR_SEQ_2_2_1[4]	CLK_COR_SEQ_2_4_1[8]	COM_BURST_VAL_1[1]	MCOMMA_10B_VALUE_1[9]	CHAN_BOND_SEQ_2_2_1[4]
10	CLK_COR_SEQ_1_2_1[7]	CLK_COR_SEQ_1_3_1[1]	CLK_COR_SEQ_2_1_1[9]	CLK_COR_SEQ_2_2_1[3]	CLK_COR_SEQ_2_4_1[7]	COM_BURST_VAL_1[0]	MCOMMA_10B_VALUE_1[8]	CHAN_BOND_SEQ_2_2_1[5]
11	CLK_COR_SEQ_1_2_1[6]	CLK_COR_SEQ_1_3_1[0]	CLK_COR_SEQ_2_1_1[8]	CLK_COR_SEQ_2_2_1[2]	CLK_COR_SEQ_2_4_1[6]	COMMA_10B_ENABLE_1[9]	MCOMMA_10B_VALUE_1[7]	CHAN_BOND_SEQ_2_2_1[6]
12	CLK_COR_SEQ_1_2_1[5]	CLK_COR_SEQ_1_4_1[9]	CLK_COR_SEQ_2_1_1[7]	CLK_COR_SEQ_2_2_1[1]	CLK_COR_SEQ_2_4_1[5]	COMMA_10B_ENABLE_1[8]	MCOMMA_10B_VALUE_1[6]	CHAN_BOND_SEQ_2_2_1[7]
13	CLK_COR_SEQ_1_2_1[4]	CLK_COR_SEQ_1_4_1[8]	CLK_COR_SEQ_2_1_1[6]	CLK_COR_SEQ_2_2_1[0]	CLK_COR_SEQ_2_4_1[4]	COMMA_10B_ENABLE_1[7]	MCOMMA_10B_VALUE_1[5]	Do Not Modify
14	CLK_COR_SEQ_1_2_1[3]	CLK_COR_SEQ_1_4_1[7]	CLK_COR_SEQ_2_1_1[5]	CLK_COR_SEQ_2_3_1[9]	CLK_COR_SEQ_2_4_1[3]	COMMA_10B_ENABLE_1[6]	MCOMMA_10B_VALUE_1[4]	CHAN_BOND_SEQ_2_2_1[8]
15	CLK_COR_SEQ_1_2_1[2]	CLK_COR_SEQ_1_4_1[6]	CLK_COR_SEQ_2_1_1[4]	CLK_COR_SEQ_2_3_1[8]	CLK_COR_SEQ_2_4_1[2]	COMMA_10B_ENABLE_1[5]	MCOMMA_10B_VALUE_1[3]	CHAN_BOND_SEQ_2_2_1[9]

Table D-8: **DRP Addresses 0x20 through 0x27**

Bit	Address							
	0x20	0x21	0x22	0x23	0x24	0x25	0x26	0x27
0	CHAN_BOND_SEQ_2_1_1[0]	CHAN_BOND_SEQ_1_4_1[2]	CHAN_BOND_SEQ_1_3_1[8]	CHAN_BOND_SEQ_1_1_1[4]	CHAN_BOND_1_MAX_SKEW_1[1]	Do Not Modify	PMA_COM_CFG[56]	PLL_COM_CFG[19]
1	CHAN_BOND_SEQ_2_1_1[1]	CHAN_BOND_SEQ_1_4_1[3]	CHAN_BOND_SEQ_1_3_1[9]	CHAN_BOND_SEQ_1_1_1[5]	CHAN_BOND_1_MAX_SKEW_1[2]	PMA_COM_CFG[52]	PMA_COM_CFG[34]	PLL_COM_CFG[18]
2	CHAN_BOND_SEQ_2_1_1[2]	CHAN_BOND_SEQ_1_4_1[4]	CHAN_BOND_SEQ_1_2_1[0]	CHAN_BOND_SEQ_1_1_1[6]	CHAN_BOND_1_MAX_SKEW_1[3]	PMA_COM_CFG[50]	PMA_COM_CFG[36]	PLL_COM_CFG[17]
3	CHAN_BOND_SEQ_2_1_1[3]	CHAN_BOND_SEQ_1_4_1[5]	CHAN_BOND_SEQ_1_2_1[1]	CHAN_BOND_SEQ_1_1_1[7]	ALIGN_COMMA_WORD_1	Do Not Modify	PMA_COM_CFG[40]	PLL_COM_CFG[16]
4	CHAN_BOND_SEQ_2_1_1[4]	CHAN_BOND_SEQ_1_4_1[6]	CHAN_BOND_SEQ_1_2_1[2]	CHAN_BOND_SEQ_1_1_1[8]	PMA_COM_CFG[18]	Do Not Modify	PMA_COM_CFG[38]	PLL_COM_CFG[15]
5	CHAN_BOND_SEQ_2_1_1[5]	CHAN_BOND_SEQ_1_4_1[7]	CHAN_BOND_SEQ_1_2_1[3]	CHAN_BOND_SEQ_1_1_1[9]	PMA_COM_CFG[17]	Do Not Modify	PMA_COM_CFG[42]	PLL_COM_CFG[14]
6	CHAN_BOND_SEQ_2_1_1[6]	CHAN_BOND_SEQ_1_4_1[8]	CHAN_BOND_SEQ_1_2_1[4]	CHAN_BOND_MODE_1[0]	PMA_COM_CFG[16]	Do Not Modify	PMA_COM_CFG[44]	PLL_COM_CFG[13]
7	CHAN_BOND_SEQ_2_1_1[7]	CHAN_BOND_SEQ_1_4_1[9]	CHAN_BOND_SEQ_1_2_1[5]	CHAN_BOND_MODE_1[1]	PMA_COM_CFG[15]	Do Not Modify	PMA_COM_CFG[46]	PLL_COM_CFG[12]
8	CHAN_BOND_SEQ_2_1_1[8]	CHAN_BOND_SEQ_1_3_1[0]	CHAN_BOND_SEQ_1_2_1[6]	CHAN_BOND_LEVEL_1[0]	PMA_COM_CFG[14]	Do Not Modify	PMA_COM_CFG[48]	PLL_COM_CFG[11]
9	CHAN_BOND_SEQ_2_1_1[9]	CHAN_BOND_SEQ_1_3_1[1]	CHAN_BOND_SEQ_1_2_1[7]	CHAN_BOND_LEVEL_1[1]	PMA_COM_CFG[13]	PMA_COM_CFG[66]	CLK25_DIVIDER[2]	PLL_COM_CFG[10]
10	CHAN_BOND_SEQ_1_ENABLE_1[1]	CHAN_BOND_SEQ_1_3_1[2]	CHAN_BOND_SEQ_1_2_1[8]	CHAN_BOND_LEVEL_1[2]	PMA_COM_CFG[12]	PMA_COM_CFG[68]	CLK25_DIVIDER[1]	PLL_COM_CFG[9]
11	CHAN_BOND_SEQ_1_ENABLE_1[2]	CHAN_BOND_SEQ_1_3_1[3]	CHAN_BOND_SEQ_1_2_1[9]	CHAN_BOND_2_MAX_SKEW_1[0]	PMA_COM_CFG[11]	PMA_COM_CFG[63]	CLK25_DIVIDER[0]	PLL_COM_CFG[8]
12	CHAN_BOND_SEQ_1_ENABLE_1[3]	CHAN_BOND_SEQ_1_3_1[4]	CHAN_BOND_SEQ_1_1_1[0]	CHAN_BOND_2_MAX_SKEW_1[1]	PMA_COM_CFG[10]	PMA_COM_CFG[61]	OOB_CLK_DIVIDER[2]	PLL_COM_CFG[7]
13	CHAN_BOND_SEQ_1_ENABLE_1[4]	CHAN_BOND_SEQ_1_3_1[5]	CHAN_BOND_SEQ_1_1_1[1]	CHAN_BOND_2_MAX_SKEW_1[2]	Do Not Modify	PMA_COM_CFG[62]	OOB_CLK_DIVIDER[1]	PLL_COM_CFG[6]
14	CHAN_BOND_SEQ_1_4_1[0]	CHAN_BOND_SEQ_1_3_1[6]	CHAN_BOND_SEQ_1_1_1[2]	CHAN_BOND_2_MAX_SKEW_1[3]	Do Not Modify	PMA_COM_CFG[64]	OOB_CLK_DIVIDER[0]	PLL_COM_CFG[5]
15	CHAN_BOND_SEQ_1_4_1[1]	CHAN_BOND_SEQ_1_3_1[7]	CHAN_BOND_SEQ_1_1_1[3]	CHAN_BOND_1_MAX_SKEW_1[0]	Do Not Modify	PMA_COM_CFG[54]	OVERSAMPLE_MODE	PLL_COM_CFG[4]

Table D-9: **DRP Addresses 0x28 through 0x2F**

Bit	Address							
	0x28	0x29	0x2A	0x2B	0x2C	0x2D	0x2E	0x2F
0	PLL_COM_CFG[3]	PLL_DIVSEL_FB[0]	PMA_COM_CFG[53]	Do Not Modify	CHAN_BOND_1_MAX_SKEW_0[0]	CHAN_BOND_SEQ_1_1_0[3]	CHAN_BOND_SEQ_1_3_0[7]	CHAN_BOND_SEQ_1_4_0[1]
1	PLL_COM_CFG[2]	TERMINATION_CTRL[4]	PMA_COM_CFG[55]	Do Not Modify	CHAN_BOND_2_MAX_SKEW_0[3]	CHAN_BOND_SEQ_1_1_0[2]	CHAN_BOND_SEQ_1_3_0[6]	CHAN_BOND_SEQ_1_4_0[0]
2	PLL_COM_CFG[1]	TERMINATION_CTRL[3]	PMA_COM_CFG[60]	Do Not Modify	CHAN_BOND_2_MAX_SKEW_0[2]	CHAN_BOND_SEQ_1_1_0[1]	CHAN_BOND_SEQ_1_3_0[5]	CHAN_BOND_SEQ_1_ENABLE_0[4]
3	PLL_COM_CFG[0]	TERMINATION_CTRL[2]	PMA_COM_CFG[58]	PMA_COM_CFG[7]	CHAN_BOND_2_MAX_SKEW_0[1]	CHAN_BOND_SEQ_1_1_0[0]	CHAN_BOND_SEQ_1_3_0[4]	CHAN_BOND_SEQ_1_ENABLE_0[3]
4	PLL_CP_CFG[7]	TERMINATION_CTRL[1]	PMA_COM_CFG[57]	PMA_COM_CFG[6]	CHAN_BOND_2_MAX_SKEW_0[0]	CHAN_BOND_SEQ_1_2_0[9]	CHAN_BOND_SEQ_1_3_0[3]	CHAN_BOND_SEQ_1_ENABLE_0[2]
5	PLL_CP_CFG[6]	TERMINATION_CTRL[0]	PMA_COM_CFG[59]	PMA_COM_CFG[5]	CHAN_BOND_LEVEL_0[2]	CHAN_BOND_SEQ_1_2_0[8]	CHAN_BOND_SEQ_1_3_0[2]	CHAN_BOND_SEQ_1_ENABLE_0[1]
6	PLL_CP_CFG[5]	TERMINATION_OVRD	PMA_COM_CFG[67]	PMA_COM_CFG[0]	CHAN_BOND_LEVEL_0[1]	CHAN_BOND_SEQ_1_2_0[7]	CHAN_BOND_SEQ_1_3_0[1]	CHAN_BOND_SEQ_2_1_0[9]
7	PLL_CP_CFG[4]	PMA_COM_CFG[32]	PMA_COM_CFG[65]	PMA_COM_CFG[4]	CHAN_BOND_LEVEL_0[0]	CHAN_BOND_SEQ_1_2_0[6]	CHAN_BOND_SEQ_1_3_0[0]	CHAN_BOND_SEQ_2_1_0[8]
8	PLL_CP_CFG[3]	PMA_COM_CFG[47]	Do Not Modify	PMA_COM_CFG[3]	CHAN_BOND_MODE_0[1]	CHAN_BOND_SEQ_1_2_0[5]	CHAN_BOND_SEQ_1_4_0[9]	CHAN_BOND_SEQ_2_1_0[7]
9	PLL_CP_CFG[2]	PMA_COM_CFG[45]	Do Not Modify	PMA_COM_CFG[2]	CHAN_BOND_MODE_0[0]	CHAN_BOND_SEQ_1_2_0[4]	CHAN_BOND_SEQ_1_4_0[8]	CHAN_BOND_SEQ_2_1_0[6]
10	PLL_CP_CFG[1]	PMA_COM_CFG[43]	Do Not Modify	PMA_COM_CFG[1]	CHAN_BOND_SEQ_1_1_0[9]	CHAN_BOND_SEQ_1_2_0[3]	CHAN_BOND_SEQ_1_4_0[7]	CHAN_BOND_SEQ_2_1_0[5]
11	PLL_CP_CFG[0]	PMA_COM_CFG[41]	Do Not Modify	PMA_COM_CFG[9]	CHAN_BOND_SEQ_1_1_0[8]	CHAN_BOND_SEQ_1_2_0[2]	CHAN_BOND_SEQ_1_4_0[6]	CHAN_BOND_SEQ_2_1_0[4]
12	PLL_DIVSEL_FB[4]	PMA_COM_CFG[39]	Do Not Modify	ALIGN_COMMA_WORD_0	CHAN_BOND_SEQ_1_1_0[7]	CHAN_BOND_SEQ_1_2_0[1]	CHAN_BOND_SEQ_1_4_0[5]	CHAN_BOND_SEQ_2_1_0[3]
13	PLL_DIVSEL_FB[3]	PMA_COM_CFG[37]	Do Not Modify	CHAN_BOND_1_MAX_SKEW_0[3]	CHAN_BOND_SEQ_1_1_0[6]	CHAN_BOND_SEQ_1_2_0[0]	CHAN_BOND_SEQ_1_4_0[4]	CHAN_BOND_SEQ_2_1_0[2]
14	PLL_DIVSEL_FB[2]	PMA_COM_CFG[33]	PMA_COM_CFG[51]	CHAN_BOND_1_MAX_SKEW_0[2]	CHAN_BOND_SEQ_1_1_0[5]	CHAN_BOND_SEQ_1_3_0[9]	CHAN_BOND_SEQ_1_4_0[3]	CHAN_BOND_SEQ_2_1_0[1]
15	PLL_DIVSEL_FB[1]	PMA_COM_CFG[35]	PMA_COM_CFG[49]	CHAN_BOND_1_MAX_SKEW_0[1]	CHAN_BOND_SEQ_1_1_0[4]	CHAN_BOND_SEQ_1_3_0[8]	CHAN_BOND_SEQ_1_4_0[2]	CHAN_BOND_SEQ_2_1_0[0]

Table D-10: **DRP Addresses 0x30 through 0x37**

Bit	Address							
	0x30	0x31	0x32	0x33	0x34	0x35	0x36	0x37
0	CHAN_BOND_SEQ_2_2_0[9]	MCOMMA_10B_VALUE_0[3]	COMMA_10B_ENABLE_0[5]	CLK_COR_SEQ_2_4_0[2]	CLK_COR_SEQ_2_3_0[8]	CLK_COR_SEQ_2_1_0[4]	CLK_COR_SEQ_1_4_0[6]	CLK_COR_SEQ_1_2_0[2]
1	CHAN_BOND_SEQ_2_2_0[8]	MCOMMA_10B_VALUE_0[4]	COMMA_10B_ENABLE_0[6]	CLK_COR_SEQ_2_4_0[3]	CLK_COR_SEQ_2_3_0[9]	CLK_COR_SEQ_2_1_0[5]	CLK_COR_SEQ_1_4_0[7]	CLK_COR_SEQ_1_2_0[3]
2	PLL_FB_DCCEN	MCOMMA_10B_VALUE_0[5]	COMMA_10B_ENABLE_0[7]	CLK_COR_SEQ_2_4_0[4]	CLK_COR_SEQ_2_2_0[0]	CLK_COR_SEQ_2_1_0[6]	CLK_COR_SEQ_1_4_0[8]	CLK_COR_SEQ_1_2_0[4]
3	CHAN_BOND_SEQ_2_2_0[7]	MCOMMA_10B_VALUE_0[6]	COMMA_10B_ENABLE_0[8]	CLK_COR_SEQ_2_4_0[5]	CLK_COR_SEQ_2_2_0[1]	CLK_COR_SEQ_2_1_0[7]	CLK_COR_SEQ_1_4_0[9]	CLK_COR_SEQ_1_2_0[5]
4	CHAN_BOND_SEQ_2_2_0[6]	MCOMMA_10B_VALUE_0[7]	COMMA_10B_ENABLE_0[9]	CLK_COR_SEQ_2_4_0[6]	CLK_COR_SEQ_2_2_0[2]	CLK_COR_SEQ_2_1_0[8]	CLK_COR_SEQ_1_3_0[0]	CLK_COR_SEQ_1_2_0[6]
5	CHAN_BOND_SEQ_2_2_0[5]	MCOMMA_10B_VALUE_0[8]	COM_BURST_VAL_0[0]	CLK_COR_SEQ_2_4_0[7]	CLK_COR_SEQ_2_2_0[3]	CLK_COR_SEQ_2_1_0[9]	CLK_COR_SEQ_1_3_0[1]	CLK_COR_SEQ_1_2_0[7]
6	CHAN_BOND_SEQ_2_2_0[4]	MCOMMA_10B_VALUE_0[9]	COM_BURST_VAL_0[1]	CLK_COR_SEQ_2_4_0[8]	CLK_COR_SEQ_2_2_0[4]	CLK_COR_SEQ_1_ENABLE_0[1]	CLK_COR_SEQ_1_3_0[2]	CLK_COR_SEQ_1_2_0[8]
7	CHAN_BOND_SEQ_2_2_0[3]	DEC_VALID_COMMA_ONLY_0	COM_BURST_VAL_0[2]	CLK_COR_SEQ_2_4_0[9]	CLK_COR_SEQ_2_2_0[5]	CLK_COR_SEQ_1_ENABLE_0[2]	CLK_COR_SEQ_1_3_0[3]	CLK_COR_SEQ_1_2_0[9]
8	CHAN_BOND_SEQ_2_2_0[2]	DEC_PCOMMA_DETECT_0	COM_BURST_VAL_0[3]	CLK_COR_SEQ_2_3_0[0]	CLK_COR_SEQ_2_2_0[6]	CLK_COR_SEQ_1_ENABLE_0[3]	CLK_COR_SEQ_1_3_0[4]	CLK_COR_SEQ_1_1_0[0]
9	CHAN_BOND_SEQ_2_2_0[1]	DEC_MCOMMA_DETECT_0	CLK_COR_SEQ_2_USE_0	CLK_COR_SEQ_2_3_0[1]	CLK_COR_SEQ_2_2_0[7]	CLK_COR_SEQ_1_ENABLE_0[4]	CLK_COR_SEQ_1_3_0[5]	CLK_COR_SEQ_1_1_0[1]
10	CHAN_BOND_SEQ_2_2_0[0]	COMMA_DOUBLE_0	CLK_COR_SEQ_2_ENABLE_0[1]	CLK_COR_SEQ_2_3_0[2]	CLK_COR_SEQ_2_2_0[8]	CLK_COR_SEQ_1_4_0[0]	CLK_COR_SEQ_1_3_0[6]	CLK_COR_SEQ_1_1_0[2]
11	CHAN_BOND_SEQ_2_3_0[9]	COMMA_10B_ENABLE_0[0]	CLK_COR_SEQ_2_ENABLE_0[2]	CLK_COR_SEQ_2_3_0[3]	CLK_COR_SEQ_2_2_0[9]	CLK_COR_SEQ_1_4_0[1]	CLK_COR_SEQ_1_3_0[7]	CLK_COR_SEQ_1_1_0[3]
12	MCOMMA_DETECT_0	COMMA_10B_ENABLE_0[1]	CLK_COR_SEQ_2_ENABLE_0[3]	CLK_COR_SEQ_2_3_0[4]	CLK_COR_SEQ_2_1_0[0]	CLK_COR_SEQ_1_4_0[2]	CLK_COR_SEQ_1_3_0[8]	CLK_COR_SEQ_1_1_0[4]
13	MCOMMA_10B_VALUE_0[0]	COMMA_10B_ENABLE_0[2]	CLK_COR_SEQ_2_ENABLE_0[4]	CLK_COR_SEQ_2_3_0[5]	CLK_COR_SEQ_2_1_0[1]	CLK_COR_SEQ_1_4_0[3]	CLK_COR_SEQ_1_3_0[9]	CLK_COR_SEQ_1_1_0[5]
14	MCOMMA_10B_VALUE_0[1]	COMMA_10B_ENABLE_0[3]	CLK_COR_SEQ_2_4_0[0]	CLK_COR_SEQ_2_3_0[6]	CLK_COR_SEQ_2_1_0[2]	CLK_COR_SEQ_1_4_0[4]	CLK_COR_EQ_1_2_0[0]	CLK_COR_SEQ_1_1_0[6]
15	MCOMMA_10B_VALUE_0[2]	COMMA_10B_ENABLE_0[4]	CLK_COR_SEQ_2_4_0[1]	CLK_COR_SEQ_2_3_0[7]	CLK_COR_SEQ_2_1_0[3]	CLK_COR_SEQ_1_4_0[5]	CLK_COR_SEQ_1_2_0[1]	CLK_COR_SEQ_1_1_0[7]

Table D-11: **DRP Addresses 0x38 through 0x3F**

Bit	Address							
	0x38	0x39	0x3A	0x3B	0x3C	0x3D	0x3E	0x3F
0	CLK_COR_SEQ_1_1_0[8]	CLK_COR_MAX_LAT_0[1]	CHAN_BOND_SEQ_2_ENABLE_0[3]	TX_DETECT_RX_CFG_0[2]	DFE_CAL_TIME[2]	CDR_PH_ADJ_TIME[1]	Do Not Modify	SATA_MIN_WAKE_0[2]
1	CLK_COR_SEQ_1_1_0[9]	CLK_COR_MAX_LAT_0[2]	CHAN_BOND_SEQ_2_ENABLE_0[4]	TX_DETECT_RX_CFG_0[3]	DFE_CAL_TIME[3]	CDR_PH_ADJ_TIME[2]	TRANS_TIME_FROM_P2_0[0]	SATA_MIN_WAKE_0[3]
2	CLK_COR_REPEAT_WAIT_0[0]	CLK_COR_MAX_LAT_0[3]	RX_EN_IDLE_RESET_PH	TX_DETECT_RX_CFG_0[4]	DFE_CAL_TIME[4]	CDR_PH_ADJ_TIME[3]	TRANS_TIME_FROM_P2_0[1]	SATA_MIN_WAKE_0[4]
3	CLK_COR_REPEAT_WAIT_0[1]	CLK_COR_MAX_LAT_0[4]	OOBDETECT_THRESHOLD_0[0]	TX_DETECT_RX_CFG_0[5]	TRANS_TIME_TO_P2_0[0]	CDR_PH_ADJ_TIME[4]	TRANS_TIME_FROM_P2_0[2]	SATA_MIN_WAKE_0[5]
4	CLK_COR_REPEAT_WAIT_0[2]	CLK_COR_MAX_LAT_0[5]	OOBDETECT_THRESHOLD_0[1]	TX_DETECT_RX_CFG_0[6]	TRANS_TIME_TO_P2_0[1]	Do Not Modify	TRANS_TIME_FROM_P2_0[3]	SATA_MIN_INIT_0[0]
5	CLK_COR_REPEAT_WAIT_0[3]	CLK_COR_KEEP_IDLE_0	OOBDETECT_THRESHOLD_0[2]	TX_DETECT_RX_CFG_0[7]	TRANS_TIME_TO_P2_0[2]	TRANS_TIME_NON_P2_0[0]	TRANS_TIME_FROM_P2_0[4]	SATA_MIN_INIT_0[1]
6	CLK_COR_REPEAT_WAIT_0[4]	CLK_COR_INSERT_IDLE_FLAG_0	PMA_COM_CFG[19]	TX_DETECT_RX_CFG_0[8]	TRANS_TIME_TO_P2_0[3]	TRANS_TIME_NON_P2_0[1]	TRANS_TIME_FROM_P2_0[5]	SATA_MIN_INIT_0[2]
7	CLK_CORRECT_USE_0	CLK_COR_DET_LEN_0[0]	TXOUTCLK_SEL_0	TX_DETECT_RX_CFG_0[9]	TRANS_TIME_TO_P2_0[4]	TRANS_TIME_NON_P2_0[2]	TRANS_TIME_FROM_P2_0[6]	SATA_MIN_INIT_0[3]
8	CLK_COR_PRECEDENCE_0	CLK_COR_DET_LEN_0[1]	TX_XCLK_SEL_0	TX_DETECT_RX_CFG_0[10]	TRANS_TIME_TO_P2_0[5]	TRANS_TIME_NON_P2_0[3]	TRANS_TIME_FROM_P2_0[7]	SATA_MIN_INIT_0[4]
9	CLK_COR_MIN_LAT_0[0]	CLK_COR_ADJ_LEN_0[0]	RX_EN_IDLE_HOLD_CDR	TX_DETECT_RX_CFG_0[11]	TRANS_TIME_TO_P2_0[6]	TRANS_TIME_NON_P2_0[4]	TRANS_TIME_FROM_P2_0[8]	SATA_MIN_INIT_0[5]
10	CLK_COR_MIN_LAT_0[1]	CLK_COR_ADJ_LEN_0[1]	RX_EN_IDLE_RESET_FR	TX_DETECT_RX_CFG_0[12]	TRANS_TIME_TO_P2_0[7]	TRANS_TIME_NON_P2_0[5]	TRANS_TIME_FROM_P2_0[9]	SATA_MIN_BURST_0[0]
11	CLK_COR_MIN_LAT_0[2]	CHAN_BOND_SEQ_LEN_0[0]	TXRX_INVERT_0[0]	TX_DETECT_RX_CFG_0[13]	TRANS_TIME_TO_P2_0[8]	TRANS_TIME_NON_P2_0[6]	TRANS_TIME_FROM_P2_0[10]	SATA_MIN_BURST_0[1]
12	CLK_COR_MIN_LAT_0[3]	CHAN_BOND_SEQ_LEN_0[1]	TXRX_INVERT_0[1]	TX_BUFFER_USE_0	TRANS_TIME_TO_P2_0[9]	TRANS_TIME_NON_P2_0[7]	TRANS_TIME_FROM_P2_0[11]	SATA_MIN_BURST_0[2]
13	CLK_COR_MIN_LAT_0[4]	CHAN_BOND_SEQ_2_USE_0	TXRX_INVERT_0[2]	Do Not Modify	Do Not Modify	Do Not Modify	TERMINATION_IMP_0	SATA_MIN_BURST_0[3]
14	CLK_COR_MIN_LAT_0[5]	CHAN_BOND_SEQ_2_ENABLE_0[1]	TX_DETECT_RX_CFG_0[0]	DFE_CAL_TIME[0]	Do Not Modify	Do Not Modify	SATA_MIN_WAKE_0[0]	SATA_MIN_BURST_0[4]
15	CLK_COR_MAX_LAT_0[0]	CHAN_BOND_SEQ_2_ENABLE_0[2]	TX_DETECT_RX_CFG_0[1]	DFE_CAL_TIME[1]	CDR_PH_ADJ_TIME[0]	Do Not Modify	SATA_MIN_WAKE_0[1]	SATA_MIN_BURST_0[5]

Table D-12: **DRP Addresses 0x40 through 0x47**

Bit	Address							
	0x40	0x41	0x42	0x43	0x44	0x45	0x46	0x47
0	SATA_MAX_WAKE_0[0]	SATA_MAX_BURST_0[4]	RX_LOS_INVALID_INCR_0[1]	PRBS_ERR_THRESHOLD_0[12]	PRBS_ERR_THRESHOLD_0[28]	PMA_CDR_SCAN_0[12]	PLL_TXDIVSEL_OUT_0[1]	Do Not Modify
1	SATA_MAX_WAKE_0[1]	SATA_MAX_BURST_0[5]	RX_LOS_INVALID_INCR_0[2]	PRBS_ERR_THRESHOLD_0[13]	PRBS_ERR_THRESHOLD_0[29]	PMA_CDR_SCAN_0[13]	PLL_SATA_0	CHAN_BOND_SEQ_2_3_0[8]
2	SATA_MAX_WAKE_0[2]	SATA_IDLE_VAL_0[0]	RX_DECODE_SEQ_MATCH_0	PRBS_ERR_THRESHOLD_0[14]	PRBS_ERR_THRESHOLD_0[30]	PMA_CDR_SCAN_0[14]	PLL_RXDIVSEL_OUT_0[0]	CHAN_BOND_SEQ_2_3_0[7]
3	SATA_MAX_WAKE_0[3]	SATA_IDLE_VAL_0[1]	RX_BUFFER_USE_0	PRBS_ERR_THRESHOLD_0[15]	PRBS_ERR_THRESHOLD_0[31]	PMA_CDR_SCAN_0[15]	PLL_RXDIVSEL_OUT_0[1]	CHAN_BOND_SEQ_2_3_0[6]
4	SATA_MAX_WAKE_0[4]	SATA_IDLE_VAL_0[2]	PRBS_ERR_THRESHOLD_0[0]	PRBS_ERR_THRESHOLD_0[16]	PMA_CDR_SCAN_0[0]	PMA_CDR_SCAN_0[16]	PCOMMA_DETECT_0	CHAN_BOND_SEQ_2_3_0[5]
5	SATA_MAX_WAKE_0[5]	SATA_BURST_VAL_0[0]	PRBS_ERR_THRESHOLD_0[1]	PRBS_ERR_THRESHOLD_0[17]	PMA_CDR_SCAN_0[1]	PMA_CDR_SCAN_0[17]	PCOMMA_10B_VALUE_0[0]	CHAN_BOND_SEQ_2_3_0[4]
6	SATA_MAX_INIT_0[0]	SATA_BURST_VAL_0[1]	PRBS_ERR_THRESHOLD_0[2]	PRBS_ERR_THRESHOLD_0[18]	PMA_CDR_SCAN_0[2]	PMA_CDR_SCAN_0[18]	PCOMMA_10B_VALUE_0[1]	CHAN_BOND_SEQ_2_3_0[3]
7	SATA_MAX_INIT_0[1]	SATA_BURST_VAL_0[2]	PRBS_ERR_THRESHOLD_0[3]	PRBS_ERR_THRESHOLD_0[19]	PMA_CDR_SCAN_0[3]	PMA_CDR_SCAN_0[19]	PCOMMA_10B_VALUE_0[2]	CHAN_BOND_SEQ_2_3_0[2]
8	SATA_MAX_INIT_0[2]	RX_XCLK_SEL_0	PRBS_ERR_THRESHOLD_0[4]	PRBS_ERR_THRESHOLD_0[20]	PMA_CDR_SCAN_0[4]	PMA_CDR_SCAN_0[20]	PCOMMA_10B_VALUE_0[3]	CHAN_BOND_SEQ_2_3_0[1]
9	SATA_MAX_INIT_0[3]	RX_STATUS_FMT_0	PRBS_ERR_THRESHOLD_0[5]	PRBS_ERR_THRESHOLD_0[21]	PMA_CDR_SCAN_0[5]	PMA_CDR_SCAN_0[21]	PCOMMA_10B_VALUE_0[4]	CHAN_BOND_SEQ_2_3_0[0]
10	SATA_MAX_INIT_0[4]	RX_SLIDE_MODE_0	PRBS_ERR_THRESHOLD_0[6]	PRBS_ERR_THRESHOLD_0[22]	PMA_CDR_SCAN_0[6]	PMA_CDR_SCAN_0[22]	PCOMMA_10B_VALUE_0[5]	CHAN_BOND_SEQ_2_4_0[9]
11	SATA_MAX_INIT_0[5]	RX_LOS_THRESHOLD_0[0]	PRBS_ERR_THRESHOLD_0[7]	PRBS_ERR_THRESHOLD_0[23]	PMA_CDR_SCAN_0[7]	PMA_CDR_SCAN_0[23]	PCOMMA_10B_VALUE_0[6]	CHAN_BOND_SEQ_2_4_0[8]
12	SATA_MAX_BURST_0[0]	RX_LOS_THRESHOLD_0[1]	PRBS_ERR_THRESHOLD_0[8]	PRBS_ERR_THRESHOLD_0[24]	PMA_CDR_SCAN_0[8]	PMA_CDR_SCAN_0[24]	PCOMMA_10B_VALUE_0[7]	CHAN_BOND_SEQ_2_4_0[7]
13	SATA_MAX_BURST_0[1]	RX_LOS_THRESHOLD_0[2]	PRBS_ERR_THRESHOLD_0[9]	PRBS_ERR_THRESHOLD_0[25]	PMA_CDR_SCAN_0[9]	PMA_CDR_SCAN_0[25]	PCOMMA_10B_VALUE_0[8]	CHAN_BOND_SEQ_2_4_0[6]
14	SATA_MAX_BURST_0[2]	RX_LOSS_OF_SYNC_FSM_0	PRBS_ERR_THRESHOLD_0[10]	PRBS_ERR_THRESHOLD_0[26]	PMA_CDR_SCAN_0[10]	PMA_CDR_SCAN_0[26]	PCOMMA_10B_VALUE_0[9]	CHAN_BOND_SEQ_2_4_0[5]
15	SATA_MAX_BURST_0[3]	RX_LOS_INVALID_INCR_0[0]	PRBS_ERR_THRESHOLD_0[11]	PRBS_ERR_THRESHOLD_0[27]	PMA_CDR_SCAN_0[11]	PLL_TXDIVSEL_OUT_0[0]	PCI_EXPRESS_MODE_0	CHAN_BOND_SEQ_2_4_0[4]

Table D-13: **DRP Addresses 0x48 through 0x4F**

Bit	Address							
	0x48	0x49	0x4A	0x4B	0x4C	0x4D	0x4E	0x4F
0	CHAN_BOND_SEQ_2_4_0[3]	PMA_RX_CFG_0[14]	RCV_TERM_GND_0	PLL_TDCC_CFG[2]	Do Not Modify	CB2_INH_CC_PERIOD_0[0]	DFE_CFG_0[6]	PMA_TX_CFG_0[10]
1	CHAN_BOND_SEQ_2_4_0[2]	PMA_RX_CFG_0[15]	RCV_TERM_VTTRX_0	Do Not Modify	RX_EN_IDLE_HOLD_DFE_0	CB2_INH_CC_PERIOD_0[1]	DFE_CFG_0[7]	PMA_TX_CFG_0[11]
2	CHAN_BOND_SEQ_2_4_0[1]	PMA_RX_CFG_0[16]	PMA_COM_CFG[27]	Do Not Modify	Do Not Modify	CB2_INH_CC_PERIOD_0[2]	DFE_CFG_0[8]	PMA_TX_CFG_0[12]
3	CHAN_BOND_SEQ_2_4_0[0]	PMA_RX_CFG_0[17]	PMA_COM_CFG[28]	Do Not Modify	RX_EN_IDLE_RESET_BUF_0	CB2_INH_CC_PERIOD_0[3]	DFE_CFG_0[9]	PMA_TX_CFG_0[13]
4	PMA_RX_CFG_0[6]	PMA_RX_CFG_0[18]	PMA_COM_CFG[29]	Do Not Modify	Do Not Modify	CHAN_BOND_KEEP_ALIGN_0	CM_TRIM_0[0]	PMA_TX_CFG_0[14]
5	PMA_RX_CFG_0[7]	PMA_RX_CFG_0[19]	PMA_COM_CFG[30]	Do Not Modify	Do Not Modify	GEARBOX_ENDEC_0[0]	CM_TRIM_0[1]	PMA_TX_CFG_0[15]
6	PMA_RX_CFG_0[8]	PMA_RX_CFG_0[20]	PMA_COM_CFG[31]	TX_IDLE_DELAY_0[0]	RX_IDLE_HI_CNT_0[0]	GEARBOX_ENDEC_0[1]	PMA_TX_CFG_0[0]	PMA_TX_CFG_0[16]
7	PMA_RX_CFG_0[9]	PMA_RX_CFG_0[21]	PMA_COM_CFG[20]	TX_IDLE_DELAY_0[1]	RX_IDLE_HI_CNT_0[1]	GEARBOX_ENDEC_0[2]	PMA_TX_CFG_0[1]	PMA_TX_CFG_0[17]
8	PMA_RX_CFG_0[10]	PMA_RX_CFG_0[22]	Do Not Modify	TX_IDLE_DELAY_0[2]	RX_IDLE_HI_CNT_0[2]	RXGEARBOX_USE_0	PMA_TX_CFG_0[2]	PMA_TX_CFG_0[18]
9	PMA_RX_CFG_1[23]	PMA_RX_CFG_1[11]	Do Not Modify	PMA_RXSYNC_CFG_0[0]	RX_IDLE_HI_CNT_0[3]	TXGEARBOX_USE_0	PMA_TX_CFG_0[3]	PMA_TX_CFG_0[19]
10	PMA_RX_CFG_0[2]	PMA_RX_CFG_1[0]	PLL_STARTUP_EN	PMA_RXSYNC_CFG_0[1]	Do Not Modify	DFE_CFG_0[0]	PMA_TX_CFG_0[4]	Do Not Modify
11	PMA_RX_CFG_0[3]	PMA_RX_CFG_1[1]	Do Not Modify	PMA_RXSYNC_CFG_0[2]	Do Not Modify	DFE_CFG_0[1]	PMA_TX_CFG_0[5]	Do Not Modify
12	PMA_RX_CFG_0[4]	PMA_RX_CFG_1[24]	Do Not Modify	PMA_RXSYNC_CFG_0[3]	RX_IDLE_LO_CNT_0[0]	DFE_CFG_0[2]	PMA_TX_CFG_0[6]	Do Not Modify
13	PMA_RX_CFG_0[5]	PMA_RX_CFG_1[12]	Do Not Modify	PMA_RXSYNC_CFG_0[4]	RX_IDLE_LO_CNT_0[1]	DFE_CFG_0[3]	PMA_TX_CFG_0[7]	Do Not Modify
14	RX_CDR_FORCE_ROTATE_0	AC_CAP_DIS_0	PLL_TDCC_CFG[1]	PMA_RXSYNC_CFG_0[5]	RX_IDLE_LO_CNT_0[2]	DFE_CFG_0[4]	PMA_TX_CFG_0[8]	Do Not Modify
15	PMA_RX_CFG_0[13]	Do Not Modify	PLL_TDCC_CFG[0]	PMA_RXSYNC_CFG_0[6]	RX_IDLE_LO_CNT_0[3]	DFE_CFG_0[5]	PMA_TX_CFG_0[9]	Do Not Modify

Appendix E

Low Latency Design

This appendix illustrates the latency of the different functional blocks inside the TX and the RX sections of the GTX transceiver. Figure E-1 shows a pictorial definition of the TX and RX latencies.

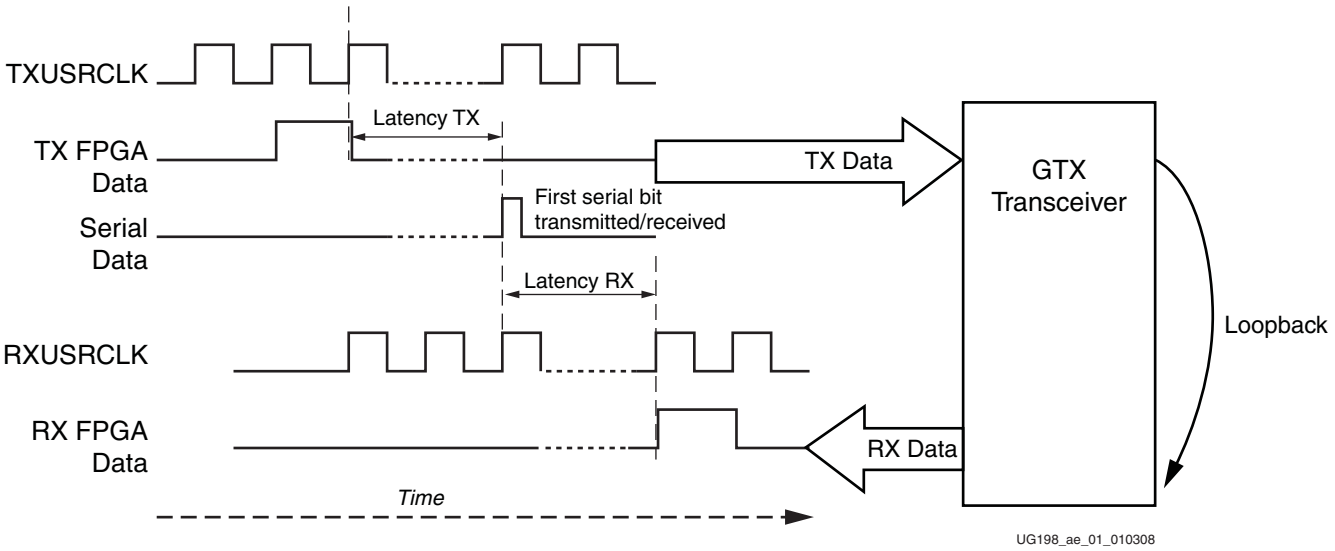


Figure E-1: **Latency Definition**

Each functional block has a latency defined as the time difference between the inputs and the outputs of the specific block. Some blocks in the GTX transceiver can be bypassed, reducing the latency of the datapath through the transmitter or the receiver. The latency of the blocks is deterministic with the exception of the RX elastic buffer (64-element FIFO) and the TX buffer (4-element FIFO). Bypassing buffers requires marginal conditions to be met, for example, phase alignment procedures or USRCLK requirements.

Refer to "TX Buffering, Phase Alignment, and TX Skew Reduction," page 138, "Configurable RX Elastic Buffer and Phase Alignment," page 202, "Connecting TXUSRCLK and TXUSRCLK2," page 120, and "Connecting RXUSRCLK and RXUSRCLK2," page 234 for the implications and marginal conditions on bypassing buffers.

GTX TX Latency

Figure E-2 shows a detailed block diagram of the GTX TX. Refer to Chapter 6, "GTX Transmitter (TX)," and Figure 6-1, page 115 for more details on this figure and the GTX TX blocks.

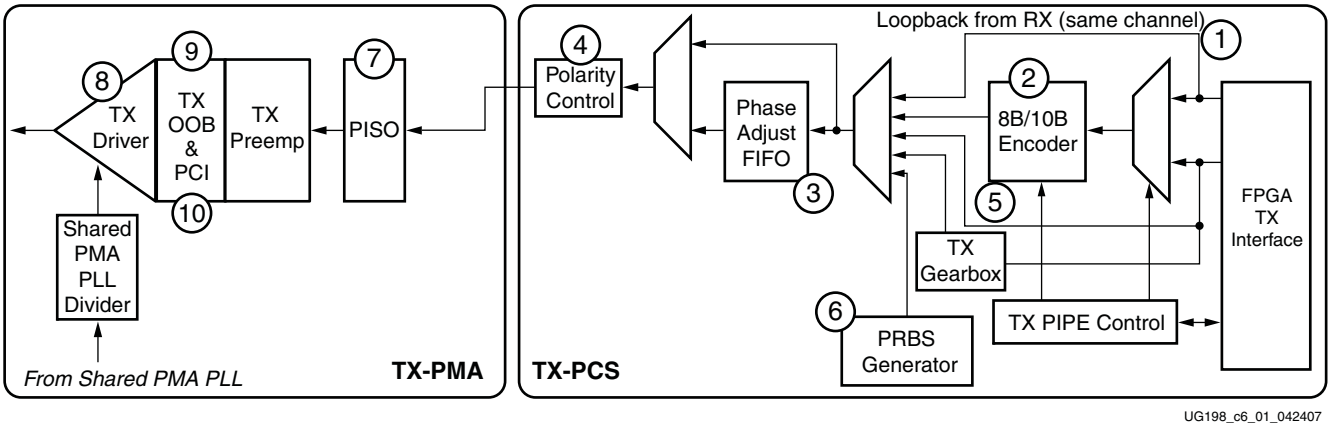


Figure E-2: **GTX TX Block Diagram**

Table E-1 defines the latency for the specific functional blocks or group of functional blocks of the transmitter section of the GTX transceiver. The values in the Block Number column correspond to the circled numbers in Figure E-2.

Table E-1: **GTX TX Latency**

Block Number	Block Name	Attribute Setting		Latency Contribution (TXUSRCLK cycles)	
				Min	Max
1	FPGA TX Interface	TXDATAWIDTH	0	0.5	0.5
			1	1	1
			2	2	2
2	8B/10B Encoder	TXENC8B10BUSE	0	0	0
			1	1	1
3	TX Buffer	TX_BUFFER_USE	FALSE	1	1
			TRUE	2	4
4+6+7+8+9	PMA + Interface	-	-	2	2

GTX RX Latency

Figure E-3 shows a detailed block diagram of the GTX RX. Refer to Chapter 7, "GTX Receiver (RX)," and Figure 7-1, page 157 for more details on this figure and the GTX RX blocks.

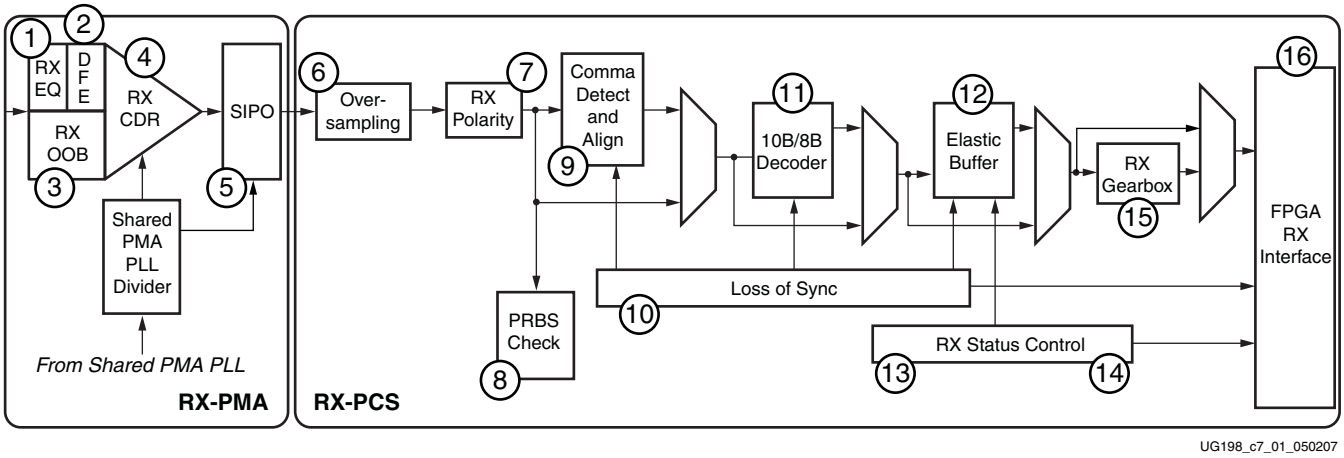


Figure E-3: **GTX RX Block Diagram**

Table E-2 defines the latency for the specific functional blocks or group of functional blocks of the receiver section of the GTX transceiver. The values in the Block Number column correspond to the circled numbers in Figure E-3.

Table E-2: **GTX RX Latency**

Block Number	Block Name	Attribute Setting		Latency Contribution (RXUSRCLK cycles)	
				Min	Max
1+2+3+4	PMA + Interface	-	-	4	4
5+6	Over-sampling	OVERSAMPLE_MODE	FALSE	0	0
			TRUE		
9	Comma Alignment	RXCOMMADETUSE	0	1	1
			1	2.5	3.5
11	10B/8B Decoder	RXDEC8B10BUSE	0	0	0
			1	1	1
12	RX Elastic Buffer	RX_BUFFER_USE	FALSE	2	2
			TRUE	2 cycles + CLK_COR_MIN_LAT/2	2 cycles + CLK_COR_MAX_LAT/2
16	FPGA RX Interface	RXDATAWIDTH	0	1.5	1.5
			1	2	2
			2	3	3

Appendix F

Advanced Clocking

Each GTX_DUAL primitive contains a reference clock multiplexing structure that is addressable using the dynamic reconfiguration port (DRP). This structure can connect one out of four different reference clock sources to the CLKIN port of the shared PMA PLL in the GTX_DUAL tile.

Direct manipulation of the reference clock multiplexers using the DRP produces flexible reference clocking arrangements instead of clocking with assignments in HDL. The reference clock applied to a particular tile can be changed at run-time. GTXRESET is applied after the clocking arrangement is changed.

There are several rules to keep in mind when designing a multi-clock scheme:

- Only one clock can be forwarded northbound through a tile at a time.
- Only one clock can be forwarded southbound through a tile at a time.
- A clock cannot be forwarded more than three tiles from its tile of origin in either direction.
- A clock can be forwarded northbound or southbound through a tile, even if the GTX transceivers in that tile are not using the forwarded clock.

Overlapping clock regions can be constructed when using advanced clocking, a practice that is not allowed when using HDL to connect the reference clock.

The reference clock multiplexing structure is shown in Figure F-1. The X's in the REFCLK MUX are don't cares. Refer to "REFCLK Guidelines" in Chapter 10 for IBUFDS details.

Note: Refer to Chapter 3, "Simulation" for the correct simulation-only attribute settings, SIM_PLL_PERDIV2 and SIM_GTPRESET_SPEEDUP, for simulating multirate designs. Refer to the "Clocking", "Reset", and "Power Control" sections of Chapter 5 for additional information.

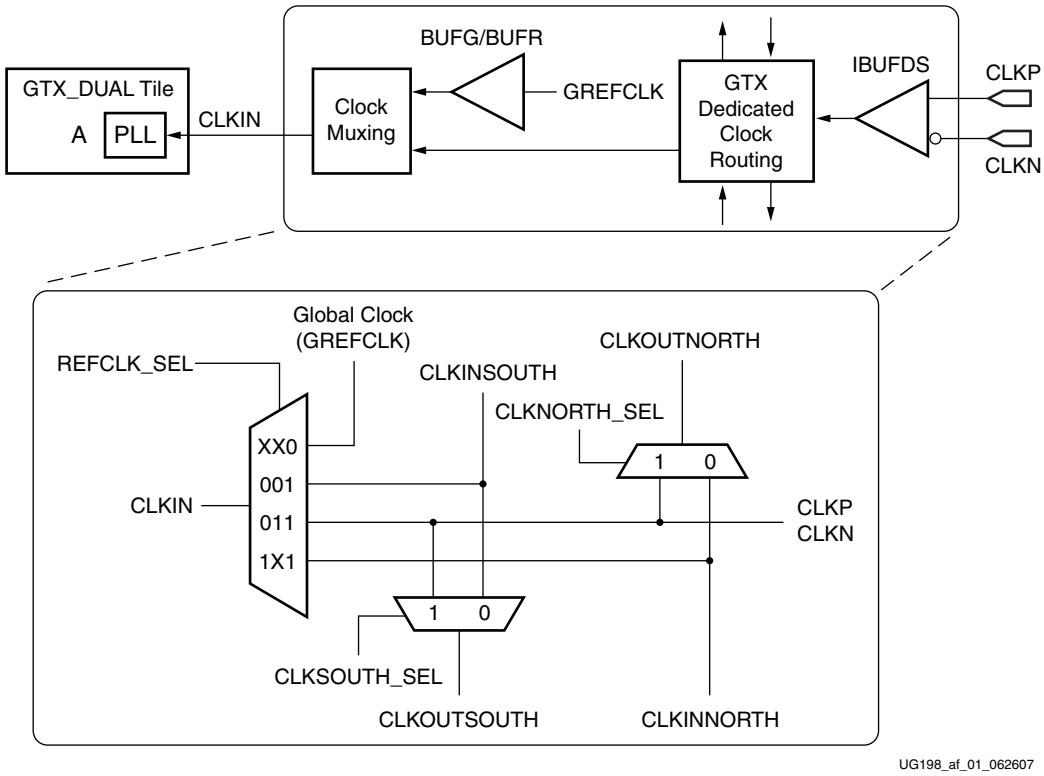


Figure F-1: **Reference Clock Multiplexing Structure**

All the MUX selectors reside in address `0x04` of the DRP and are mapped as shown in Table F-1.

Table F-1: **MUX Selector**

REFCLK_SEL	Bit	Address
REFCLK_SEL[0]	6	`0x04`
REFCLK_SEL[1]	5	`0x04`
REFCLK_SEL[2]	4	`0x04`
CLKSOUTH_SEL	7	`0x04`
CLKNORTH_SEL	8	`0x04`

To ensure that other attributes in DRP address `0x04` are not accidentally changed, a read/modify/write procedure should be used to change the MUX selectors.

Example

The example system shown in Figure F-2 has six GTX_DUAL tiles with four reference clocks. It demonstrates several different clocking schemes.

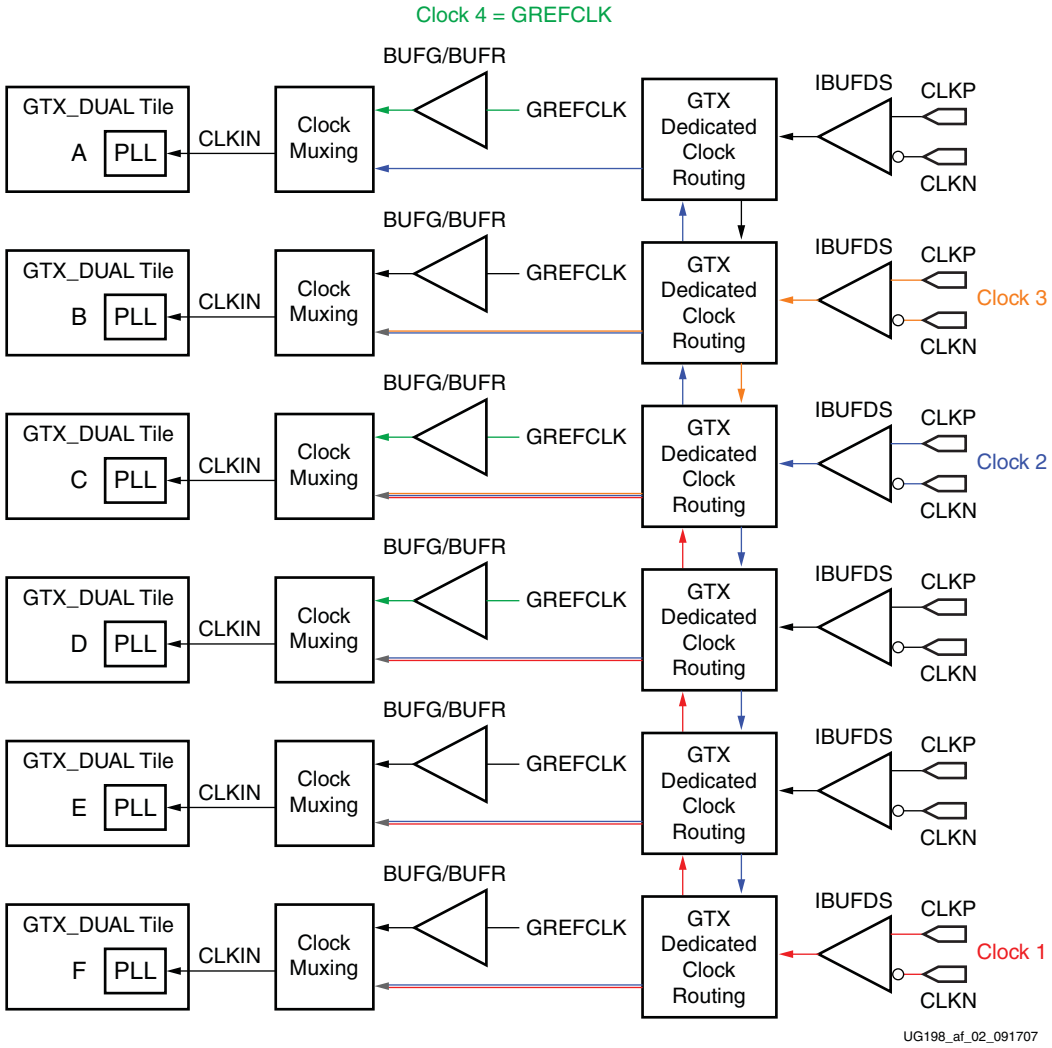


Figure F-2: **Example System**

Note: Any GTX_DUAL tile that sources a reference clock must be instantiated, and REFCLKPWRDNB must be asserted High.

Table F-2 describes the example in Figure F-2. When the GTX_DUAL tile D is used to select clock 1 as the CLKIN source, REFCLK_SEL is set to `1x1`.

Table F-2: **GTX Tile Utilization**

GTX_DUAL Tile	CLKIN Options	REFCLK_SEL	CLKNORTH_SEL	CLKSOUTH_SEL
A	Clock 2	`1x1`	Don't Care: Nowhere to send the northbound clock.	Don't Care: No southbound clock to forward.
	Clock 4	`xx0`		
B	Clock 2	`1x1`	0: Forward clock 2 northbound to tile A.	1: Drive clock 3 southbound to tile C.
	Clock 3	`011`		
C	Clock 1	`1x1`	1: Drive clock 2 northbound to tile B.	1: Drive clock 2 southbound to tile D.
	Clock 2	`011`		
	Clock 3	`001`		
	Clock 4	`xx0`		
D	Clock 1	`1x1`	0: Forward clock 1 northbound to tile C.	0: Forward clock 2 southbound to tile E.
	Clock 2	`001`		
	Clock 4	`xx0`		
E	Clock 1	`1x1`	0: Forward clock 1 northbound to tile D.	0: Forward clock 2 southbound to tile F.
	Clock 2	`001`		
F	Clock 1	`011`	1: Drive clock 1 northbound to tile E.	Don't Care: Clock 2 cannot be sent more than three tiles from where it originates.
	Clock 2	`001`		

Index

Numerics

A

B

C

D

E

F

G

H

I

J

R

S

T

U

V

W

X