RocketIO™ X Transceiver User Guide

UG035 (v2.0) February 22, 2007

XILINX®

RocketIO™ X Transceiver User Guide
UG035 (v2.0) February 22, 2007

The following table shows the revision history for this document.

Date	Version	Revision
10/31/03	1.0	Xilinx initial release. (ADVANCE DRAFT)
11/14/03	1.1	Minor updates made throughout.
12/09/03	1.2	Additions to end of Chapter 2 and minor changes Appendix C.
12/16/03	1.2.1	Change made to "Status Indication."

Date	Version	Revision
03/09/04	1.3	Chapter 1, "RocketIO X Transceiver Overview": • Modified GT10 Primitive in "Definitions:," page 23. • Modified Table 1-2, page 25 and added Note 3. • Updated definitions in Primitive Ports, Table 1-4, page 26. Made changes to: BREFCLKNIN, BREFCLKPIN, PMAREGADDR[5:0], PMAREGDATAIN[7:0], PMAREGRW, RXBUFSTATUS[1:0], RXCHARISCOMMA, RXCHARISK[7:0], RXCLKCORCNT[2:0], RXCOMMADETUSE, RXDATA[63:0], RXDATAWIDTH[1:0], RXDEC8B10BUSE, RXDESCRAM64B66BUSE, RXINTDATAWIDTH[1:0], RXLOSSOFSYNC[1:0], RXNOTINTABLE[7:0], RXRUNDISP[7:0], RXUSRCLK, RXUSRCLK2, TXBYPASS8B10B[7:0], TXCHARDISPMODE[7:0], TXCHARDISPVAL[7:0], TXCHARISK[7:0], TXDATA[63:0], TXDATAWIDTH[1:0], TXGEARBOX64B66BUSE, TXINTDATAWIDTH[1:0], TXKERR[7:0], TXRUNDISP[7:0], TXSCRAM64B66BUSE, TXUSRCLK, and TXUSRCLK2. • Updated RocketIO X Transceiver Attributes, Table 1-5, page 32. Made changes to: ALIGN_COMMA_WORD, CHAN_BOND_MODE, CHAN_BOND_SEQ_1_*[10:0], CLK_COR_8B10B_DE, CLK_COR_MAX_LAT, CLK_COR_MIN_LAT, CLK_COR_SEQ_2_USE, CLK_COR_SEQ_LEN, and RX_LOSS_OF_SYNC_FSM. • Updated text under "Modifiable Attributes," page 36 and created new Appendix F, "Modifiable Attributes." Chapter 2, "Digital Design Considerations" • Changed last sentence of the last paragraph under 8B/10B Table , page 44. • Added sentence to "RXNOTINTABLE," page 47. • Modified Table 2-8, page 50. • Added new heading, "Setting MCOMMA_10B_VALUE, PCOMMA_10B_VALUE, and COMMA_10B_MASK (Special Note)," page 50. • Corrected column heading in Table 2-9, page 51. • Added small table to show byte alignment options under "ALIGN_COMMA_WORD," page 52. • Modified text under 64B/66B Encoder "Bypassing," page 53 and made changes to Table 2-10, page 53. • Added note to section, "Scrambler," page 55 (Normal Operation). • Updated "Functions Common to All Protocols" under section,"Clock Correction," page 59. • Added text to "Clock Correction Sequences," page 60 (relating to 11th bit format). • Updated Attribute Setting column in Table 2-13, page 61. • Added Channel Bonding match logic example under section "Channel Bonding," page 61. • Added new section, "Applications Without Channel Bonding." • Modified text under "Status Indication," page 64 and changed Table 2-15, page 64. • Modified text under "Event Indication," page 64 and changed Table 2-17, page 65. Chapter 3, "Clocking and Clock Domains": • Added note to "Clock Domain Architecture," page 69. • Modified Figure 3-1, page 69, Figure 3-2 and Figure 3-3. Minor editing change to Figure 3-4. • Modified Figure 3-6 through Figure 3-12.

Date	Version	Revision
03/09/04 (*Cont'd*)	1.3 (*Cont'd*)	• Added new section, "PMA," page 77 and Table 3-3, page 78. • Added new section, "Data Path Latency," page 80, and new tables, Table 3-4, page 80, and Table 3-5, page 81. • Modified text (at end of section) and added "LT1963A" to "Voltage Regulation," page 104. • Replaced Figure 4-23, page 107. • Modified first paragraph under "Routing Serial Traces," page 107. • Modified Figure 4-31, page 112 and Figure 4-32, page 112. • Deleted Figure 4-33. • Chapter 5, "Simulation and Implementation": • Modified Table 5-3, page 117 (Loopback Modes). • Modified section, "Parallel Loopback," page 118. • Added new section, "Post/Pre-Driver Serial Loopback," page 118. Appendix B, "8B/10B Valid Characters" • Updated Table B-1, page 127. Changed entries in "Current RD+" column for Data Byte D9.1, D9.2, D9.3, D9.4, D9.5, D9.6, and D9.7. Appendix C, "PMA Attribute Programming Bus" • Added table references in "Register Definition" section and changed the order of the register definitions. • Modified Table C-1, page 137. Appendix D, "Virtex-II Pro to Virtex-II Pro X FPGA Design Migration" • Added note directly above Figure D-2, page 159. • Added new section, "Migration Differences." Added new "Index."
06/29/04	1.4	• Changed the PMA_SPEED attribute description in Table 1-5. • Modified "Clock Correction Sequences" in Chapter 2 and Table 2-13. • Removed section, "Applications Without Channel Bonding" in Chapter 2. • Modified first paragraph after Figure 2-6, page 46. • Added illustration to section on "ALIGN_COMMA_WORD," page 52. • Added note to "Clock Correction Sequences," page 60. • Made change to Figure 2-12, page 63. • Added sample Verilog code after Table 2-17, page 65. • Added notes to Table 3-3, page 78. • Removed old Figure 3-2 through Figure 3-12 and related text. Replaced with new Figure 3-2 through Figure 3-15 and related text. • Added new section, "PMA" in Chapter 3 and modified Table 3-3, page 78. • Chapter 3, "Clocking and Clock Domains" text revised; new Use Model figures and Table 3-2 added. • Added note to "Simulation TX Emphasis and RX Equalization Settings," page 103. • Added "Conditions" to T_{LOCK} row of Table 4-7, page 104. • Updated Figure 4-32 and the paragraph above it.

Date	Version	Revision
06/29/04 (*Cont'd*)	1.4 (*Cont'd*)	• Removed sentence from the first paragraph of "Model Considerations" in Chapter 5. • Added paragraph to "Post/Pre-Driver Serial Loopback" in Chapter 5 and modified Table 5-4. • Made changes to Table A-3, page 124, Table A-4, page 125,and Table A-5, page 126. • Corrected Table B-1. • Added new Appendix G, "Related Online Documents." • Miscellaneous edits throughout.
11/22/04	1.5	• Changed RXCOMMADET definition in Table 1-4. • Modified graphic illustrating "ALIGN_COMMA_WORD," page 52. • Added text to TXKERR[3] in Table 2-10, page 53 and a note to the table. • Fixed error in Figure 3-10, page 74. • Edited Mode Number column in Table 3-3, page 78. • Edited Table 4-6, page 95 and added note. • Changed VCSO number to EV-2101CA in text and in Figure 4-32, page 112. • Added PMA attributes and updated Table C-2, page 138; added new PMA attribute definitions. • Added XAPP762 and XAPP767 to Appendix G, "Related Online Documents."Also added RPT007 to "Characterization Reports."
02/22/07	2.0	• "64B/66B," page 53: Corrected TXKERR bit that indicates even boundary for per-clock bypassing from bit 0 to bit 3. • "64B/66B," page 53: Simplified description of TXC bit tracking with TXDAT. • Appendix C, "PMA Attribute Programming Bus": Combined RXCPGAIN into RXCPI as "RXCPGAIN, RXCPI," page 148. Deleted separate RXCPGAIN entry. • Appendix G, "Related Online Documents": Removed XAPP752. • Removed all references to transfer rate of 10.3125 Gb/s and OC-192 protocol (not supported). • Removed all table rows/columns with unsupported "TBD" referenes. • Table 1-4: Corrected definitions of TXCHARDISPMODE[7:0] and TXCHARDISPVAL[7:0]. • Table 3-2, Table 3-3, Table F-1: Removed unsupported PMA modes, use models, and standards. Removed former Table F-3, Table F-4, and Table F-8 (unsupported standards). • Deleted former Table 4-7 and Table 4-11 and replaced them with reference links to the data sheet. • Table 4-9: Corrected signal names to RXLOOPFILTERC[1:0] and RXLOOPFILTERR[2:0]. • Table C-2: Corrections at addresses 0x0A, 0x0B, and 0x0F. • Table C-12: Corrected logic levels. • Figure 3-2 through Figure 3-15, inclusive: Corrected BREFCLK input to GT10 from single-ended to differential.

Table of Contents

Chapter 4: Analog Design Considerations

Chapter 5: Simulation and Implementation

Appendix A: RocketIO X Transceiver Timing Model

Schedule of Figures

Chapter 1: RocketIO X Transceiver Overview

Chapter 2: Digital Design Considerations

Chapter 3: Clocking and Clock Domains

Chapter 4: Analog Design Considerations

Chapter 5: Simulation and Implementation

Appendix A: RocketIO X Transceiver Timing Model

Appendix B: 8B/10B Valid Characters

Appendix C: PMA Attribute Programming Bus

Appendix D: Virtex-II Pro to Virtex-II Pro X FPGA Design Migration

Appendix E: Serial Backplane System Design

Appendix F: Modifiable Attributes

Appendix G: Related Online Documents

Schedule of Tables

Chapter 1: RocketIO X Transceiver Overview

Chapter 2: Digital Design Considerations

Chapter 3: Clocking and Clock Domains

Chapter 4: Analog Design Considerations

Chapter 5: Simulation and Implementation

Appendix A: RocketIO X Transceiver Timing Model

Appendix B: 8B/10B Valid Characters

Appendix C: PMA Attribute Programming Bus

Appendix D: Virtex-II Pro to Virtex-II Pro X FPGA Design Migration

Appendix E: Serial Backplane System Design

Appendix F: Modifiable Attributes

Appendix G: Related Online Documents

About This Guide

RocketIO X Features

RocketIO X transceivers have flexible, programmable features that allow a multi-gigabit serial transceiver (MGT) to be easily integrated into any Virtex-II Pro X design:

- Variable speed full-duplex transceiver, allowing 2.488 Gb/s to 6.25 Gb/s transfer rates, including specific baud rates used by various standards (listed in Table 1-2, page 25)
- Depending on the Virtex-II Pro X device, from 8 to 20 transceiver modules on an FPGA
- Monolithic clock synthesis and clock recovery system, eliminating the need for external components
- Automatic lock-to-reference function
- Serial output differential swing that can be programmed between 200 mV to 1600 mV (peak-peak), allowing compatibility with other serial system voltage levels
- Levels of programmable emphasis from 0 to 500% (not all emphasis and swing combinations can be attained)
- Receiver equalization
- AC and DC coupling
- On-chip termination of 50Ω (eliminating the need for external termination resistors)
- Pre and post driver serial and parallel TX to RX internal loopback modes for testing operability
- Programmable comma detection to allow for any protocol and detection of any 10-bit character
- 8B/10B and 64B/66B encoding blocks

User Guide Organization

This guide is organized as follows:

- Preface, "About This Guide" – Summary of RocketIO X transceiver features, which allow a multi-gigabit serial transceiver to be integrated easily into any Virtex-II Pro X design.
- Chapter 1, "RocketIO X Transceiver Overview" – RocketIO X transceiver basic architecture and capabilities. Includes instantiations, VHDL code examples, available ports, primitive and modifiable attributes, and byte mapping.
- Chapter 2, "Digital Design Considerations" – Ports and attributes for the provided communications protocol primitives; transceiver instantiation; 8B/10B encoding; 64B/66B encoding; channel bonding.
- Chapter 3, "Clocking and Clock Domains" – Clock domain architecture; clock ports, and examples for clocking and reset schemes.

- Chapter 4, "Analog Design Considerations" – RocketIO X serial overview; pre-emphasis; jitter; clock/data recovery; PCB design requirements.
- Chapter 5, "Simulation and Implementation" – Simulation models and considerations; implementation tools; and debugging and diagnostics.
- Appendix A, "RocketIO X Transceiver Timing Model" – Timing parameters associated with the RocketIO X transceiver core.
- Appendix B, "8B/10B Valid Characters" – Valid data and K characters table.
- Appendix C, "PMA Attribute Programming Bus" – RocketIO X transceiver simple, parallel programming bus for dynamically configuring the PMA attribute settings. For *Advanced Users Only.*
- Appendix D, "Virtex-II Pro to Virtex-II Pro X FPGA Design Migration" – Important differences regarding migration from Virtex-II Pro™ to the Virtex-II Pro X FPGAs. Highlights relevant PCB, power supply, and reference clock differences.
- Appendix E, "Serial Backplane System Design" – Additional PCB design guidelines to meet the demands of the RocketIO X transceiver for operation above 3.125 Gb/s.

Related Information

For a complete menu of online information resources available on the Xilinx website, visit **http://www.xilinx.com/virtex2pro x/**.

For a comprehensive listing of available tutorials and resources on network technologies and communications protocols, visit **http://www.iol.unh.edu/training/**.

Additional Resources

For additional information, go to http://support.xilinx.com. The following table lists some of the resources available on this website. Use the URLs to access these resources directly.

Resource	Description/URL
Tutorials	Tutorials covering Xilinx design flows, from design entry to verification and debugging http://support.xilinx.com/support/techsup/tutorials/index.htm
Answer Browser	Database of Xilinx solution records http://support.xilinx.com/xlnx/xil_ans_browser.jsp
Application Notes	Descriptions of device-specific design techniques and approaches http://support.xilinx.com/apps/appsweb.htm
Data Sheets	Device-specific information on Xilinx device characteristics, including readback, boundary scan, configuration, length count, and debugging http://support.xilinx.com/xlnx/xweb/xil_publications_index.jsp
Problem Solvers	Interactive tools that allow you to troubleshoot your design issues http://support.xilinx.com/support/troubleshoot/psolvers.htm
Tech Tips	Latest news, design tips, and patch information for the Xilinx design environment http://www.support.xilinx.com/xlnx/xil_tt_home.jsp

User Guide Conventions

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

Port and Attribute Names

Input and output ports of the RocketIO X transceiver primitives are denoted in upper-case letters. Attributes of the RocketIO X transceiver are denoted in upper-case letters with underscores. Trailing numbers in primitive names denote the byte width of the data path. These values are preset and not modifiable. When assumed to be the same frequency, RXUSRCLK and TXUSRCLK are referred to as USRCLK and can be used interchangeably. This also holds true for RXUSRCLK2, TXUSRCLK2, and USRCLK2.

Comma Definition

A comma is a "K" character used by the transceiver to align the serial data on a byte/half-word boundary (depending on the protocol used), so that the serial data is correctly decoded into parallel data.

Typographical

The following typographical conventions are used in this document:

Convention	Meaning or Use	Example
`Courier font`	Messages and prompts that the system displays	`speed grade: - 100`
`Courier bold`	Literal commands that you enter in a syntactical statement	**`ngdbuild`** *`design_name`*
Helvetica bold	Commands that you select from a menu	**File → Open**
	Keyboard shortcuts	**Ctrl+C**
Italic font	Variables in syntax statements for which you must supply values	**`ngdbuild`** *`design_name`*
	References to other manuals	See the *Virtex-II Pro User Guide* for more information.
	Emphasis in text	If a wire is drawn so that it overlaps the pin of a symbol, the two nets are *not* connected.
Square brackets [ ]	Optional entry / parameter; required in bus specifications, such as **`bus[7:0]`**	**`ngdbuild`** `[`*`option_name`*`]` *`design_name`*
Braces { }	A list of items from which you must choose one or more	**`lowpwr =`**`{`**`on`**`|`**`off`**`}`
Vertical bar |	Separates items in a list of choices	**`lowpwr =`**`{`**`on`**`|`**`off`**`}`
Ellipsis ...	Repetitive material that has been omitted	**`allow block`** *`block_name loc1 loc2 ... locn;`*

Chapter 1

RocketIO X Transceiver Overview

Basic Architecture and Capabilities

Note: **The definitions, descriptions, and recommendations in this user guide reflect Step 1 silicon. For Step 0 silicon, see the Errata for special considerations.**

The RocketIO X block diagram is illustrated in Figure 1-1. Depending on the device, a Virtex-II Pro X FPGA has between 8 and 20 transceiver modules, as shown in Table 1-1.

Table 1-1: **Number of RocketIO X Cores per Device Type**

Device	RocketIO X Cores
XC2VPX20	8
XC2VPX70	20

Definitions:

- Attribute – An attribute is a control parameter to configure the RocketIO X transceiver. There are both primitive ports (traditional I/O ports for control and status) and transceiver attributes. Transceiver attributes are also controls to the transceiver that regulate data widths and encoding rules, but controls that are configured as a group in "soft" form through the invocation of a primitive.
- GT10 Primitive – A primitive is a pre-designed collection of attribute values that accomplish a known data rate, encoding type, data width, etc. A single primitive invocation, for example, OC-48 mode which configures all the dozens of pertinent attributes to their correct values in a single step.

The transceiver module is designed to operate at any serial bit rate in the range of 2.488 Gb/s to 6.25 Gb/s per channel, including the specific bit rates used by the communications standards listed in Table 1-2, page 25. Data-rate specific attribute settings are set appropriately in the GT10 primitives.

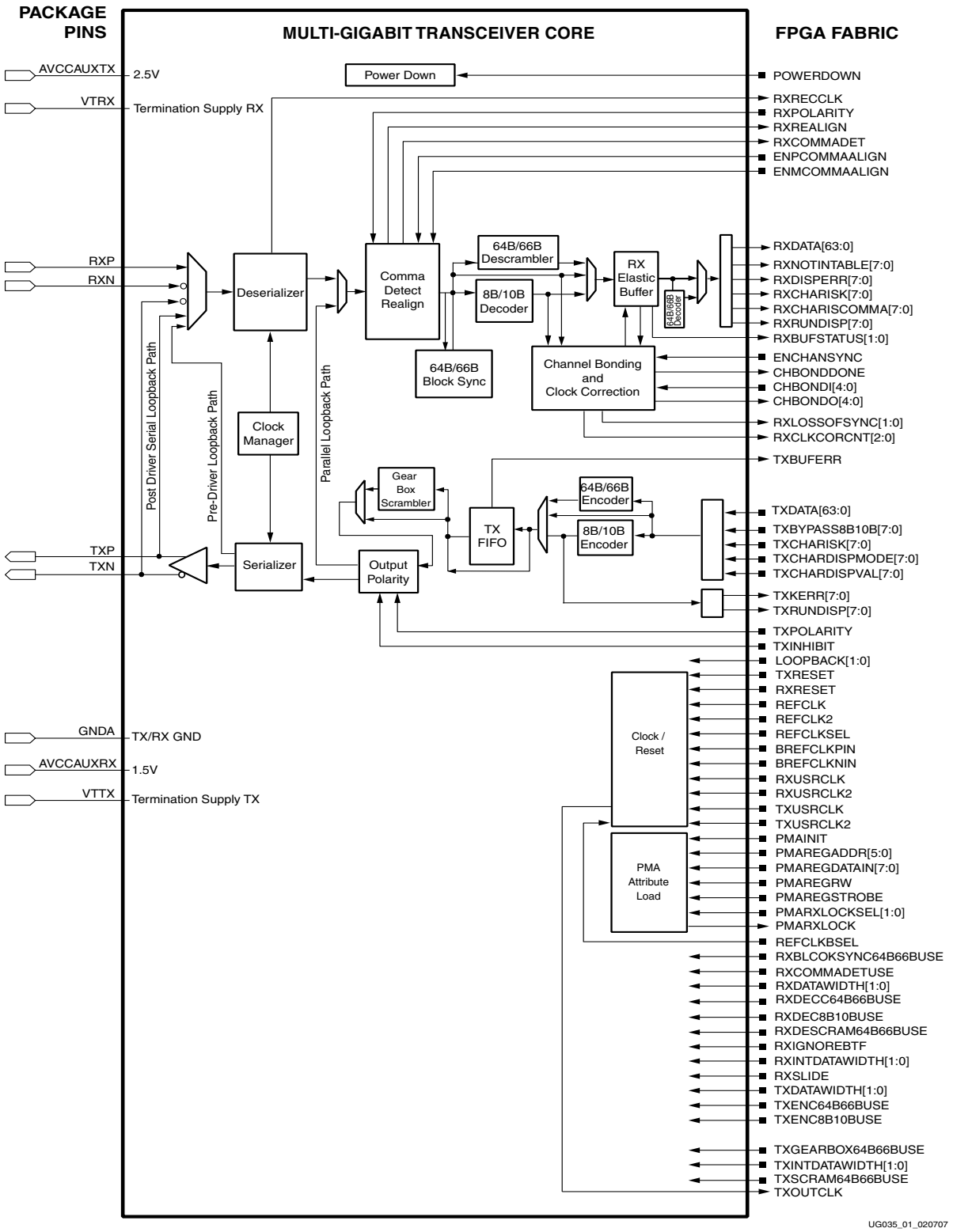


Figure 1-1: **RocketIO X Transceiver Block Diagram**

Table 1-2: **Communications Standards Supported by RocketIO X Transceiver**

Mode	Channels(1) (Lanes)	I/O Bit Rate (Gb/s)
SONET OC-48	1	2.488
PCI Express	1, 2, 4, 8, 16	2.5
Infiniband	1, 4, 12	2.5
XAUI (10-Gigabit Ethernet)	4	3.125
XAUI (10-Gigabit Fibre Channel)	4	3.1875
Aurora (Xilinx protocol)	1, 2, 3, 4, ...	2.488 – 6.25
Custom Mode	1, 2, 3, 4, ...	2.488 – 6.25

Notes:

1. One channel is considered to be one transceiver.

Table 1-3 lists the transceiver primitives provided. These primitives carry attributes set to default values for the communications protocols listed in Table 1-2. Data widths of one, two, and four bytes (lower speeds) or four and eight bytes (higher speeds) are selectable for the various protocols.

Table 1-3: **Supported RocketIO X Transceiver Primitives**

Primitive	Description	Primitive	Description
GT10_CUSTOM	Fully customizable by user	GT10_XAUI_2	10GE XAUI, 2-byte data path
GT10_OC48_1	SONET OC-48, 1-byte data path	GT10_XAUI_4	10GE XAUI, 4-byte data path
GT10_OC48_2	SONET OC-48, 2-byte data path	GT10_AURORA_1	Xilinx protocol, 1-byte data path
GT10_OC48_4	SONET OC-48, 4-byte data path	GT10_AURORA_2	Xilinx protocol, 2-byte data path
GT10_PCI_EXPRESS_1	PCI Express, 1-byte data path	GT10_AURORA_4	Xilinx protocol, 4-byte data path
GT10_PCI_EXPRESS_2	PCI Express, 2-byte data path	GT10_10GE_4	10Gbit Ethernet, 4-byte data path
GT10_PCI_EXPRESS_4	PCI Express, 4-byte data path	GT10_10GE_8	10Gbit Ethernet, 8-byte data path
GT10_INFINIBAND_1	Infiniband, 1-byte data path	GT10_10GFC_4	10Gbit Fibre Channel, 4-byte data path
GT10_INFINIBAND_2	Infiniband, 2-byte data path	GT10_10GFC_8	10Gbit Fibre Channel, 8-byte data path
GT10_INFINIBAND_4	Infiniband, 4-byte data path	GT10_AURORAX_4	Xilinx 10G protocol, 4-byte data path
GT10_XAUI_1	10GE XAUI, 1-byte data path	GT10_AURORAX_8	Xilinx 10G protocol, 8-byte data path

There are three ways to configure the RocketIO X transceiver:

- Static properties can be set through attributes in the HDL code. Use of attributes are covered in detail in "Primitive Attributes," page 32.
- Dynamic changes can be made to the attributes via the attribute programming bus. See Appendix C, "PMA Attribute Programming Bus"for details.
- Dynamic changes can be made through the ports of the primitives.

The RocketIO X transceiver consists of the Physical Media Attachment (PMA) and Physical Coding Sublayer (PCS). The PMA contains the serializer/deserializer (SERDES), TX and RX buffers, clock generator, and clock recovery circuitry. The PCS contains the 8B/10B encoder/decoder, 64B/66B encoder/decoder/scrambler/descrambler, and the

elastic buffer supporting channel bonding and clock correction. Refer again to Figure 1-1, page 24, showing the RocketIO X transceiver top-level block diagram and FPGA interface signals.

RocketIO X Transceiver Instantiations

For the different clocking schemes, several things must change, including the clock frequency for USRCLK and USRCLK2 discussed in Chapter 3, "Clocking and Clock Domains." The data and control ports for GT10_CUSTOM always use maximum bus widths. To implement the designs that do not take full advantage of the bus width, concatenate zeros onto inputs and the wires for outputs for Verilog designs, and set outputs to open and concatenate zeros on unused input bits for VHDL designs.

HDL Code Examples

The Architecture Wizard can be used to create instantiation templates. This wizard creates code and instantiation templates that define the attributes for a specific application.

Available Ports

Table 1-4 contains the port descriptions of all primitives. The RocketIO X transceiver primitives contain 72 ports. The differential serial data ports (RXN, RXP, TXN, and TXP) are connected directly to external pads; the remaining 68 ports are all accessible from the FPGA logic.

Table 1-4: **Primitive Ports**

Port	I/O	Port Size	Definition
BREFCLKNIN	I	1	Differential BREFCLK negative input from the BREFCLK pad. See Figure 4-31 and Figure 4-32 for analog considerations.
BREFCLKPIN	I	1	Differential BREFCLK positive input from the BREFCLK pad. See Figure 4-31 and Figure 4-32 for analog considerations.
CHBONDDONE	O	1	Indicates a receiver has successfully completed channel bonding when asserted High.
CHBONDI[4:0]	I	5	The channel bonding control that is used only by "slaves" which is driven by a transceiver's CHBONDO port. See Figure 2-10.
CHBONDO[4:0]	O	5	Channel bonding control that passes channel bonding and clock correction control to other transceivers. See Figure 2-10.
ENCHANSYNC	I	1	Control from the fabric to the transceiver enables the transceiver to perform channel bonding.
ENMCOMMAALIGN	I	1	Selects realignment of incoming serial bitstream on minus-comma. When asserted realigns serial bitstream byte boundary to where minus-comma is detected.
ENPCOMMAALIGN	I	1	Selects realignment of incoming serial bitstream on plus-comma. When reasserted realigns serial bitstream byte boundary to where plus-comma is detected.

Table 1-4: **Primitive Ports *(Continued)***

Port	I/O	Port Size	Definition
LOOPBACK[1:0]	I	2	Selects the three loopback test modes. These modes are internal parallel, pre-driver serial, and post-driver serial. See Table 5-3, page 117 for more information.)
PMAINIT	I	1	When asserted High and then deasserted Low, reloads the PMA coefficients into the PMA from the attribute PMA_SPEED and then resets the PCS.
PMAREGADDR[5:0]	I	6	PMA attribute bus address. This input is asynchronous.
PMAREGDATAIN[7:0]	I	8	PMA attribute bus data input. This input is asynchronous.
PMAREGRW	I	1	PMA attribute bus read/write control. This input is asynchronous.
PMAREGSTROBE	I	1	PMA attribute bus strobe. Note: This input is asynchronous.
PMARXLOCK	O	1	Indicates that the receive PLL has locked in the fine loop. When RXPLL is set to "Lock to Local Reference Clock, " this signal is always a logic 1.
PMARXLOCKSEL[1:0]	I	2	Selects determination of lock in the receive PLL. See Table 4-7, page 96.
POWERDOWN	I	1	Shuts down both the receiver and transmitter sides of the transceiver when asserted High. Note: This input is asynchronous.
REFCLK	I	1	The reference clock net that is embedded within the fabric.
REFCLK2	I	1	An alternative to REFCLK. Can be selected by the REFCLKSEL.
REFCLKBSEL	I	1	Selects between BREFCLK and REFCLK/REFCLK2 as reference clock. Asserted selects BREFCLK. Deasserted selects REFCLK or REFCLK2, depending on REFCLKSEL.
REFCLKSEL	I	1	Selects between REFCLK or REFCLK2 as reference clock. Deasserted selects REFCLK. Asserted selects REFCLK2.
RXBUFSTATUS[1:0]	O	2	Receiver elastic buffer status. Indicates the status of the receive FIFO pointers, channel bonding skew, and clock correction events. See "Status and Event Bus," page 64.
RXBLOCKSYNC64B66BUSE	I	1	If asserted, the block sync is used. If deasserted, the block sync logic is bypassed.
RXCHARISCOMMA[7:0]	O	1, 2, 4, 8(1)	Indicates the reception of K28.0, K28.5, K28.7, and some out of band commas (depending on the setting of DEC_VALID_COMMA_ONLY by the 8B/10B decoder.
RXCHARISK[7:0]	O	1, 2, 4, 8(1)	If 8B/10B decoding is enabled, it indicates that the received data is a "K" character when asserted. Included in Byte-mapping. If 8B/10B decoding is bypassed, it remains as the first bit received (Bit "a") of the 10-bit encoded data (see Figure 2-3).

Table 1-4: **Primitive Ports *(Continued)***

Port	I/O	Port Size	Definition
RXCLKCORCNT[2:0]	O	3	Status that denotes occurrence of clock correction, channel bonding, and receive FIFO pointer status. This status is synchronized on the incoming RXDATA. See "Clock Correction," page 59 and "Status and Event Bus," page 64.
RXCOMMADET	O	1	Indicates that the symbol defined by PCOMMA_10B_VALUE (IF PCOMMA_DETECT is asserted) and/or MCOMMA_10B_VALUE (if MCOMMA_DETECT is asserted) has been received.
RXCOMMADETUSE	I	1	If asserted High, the comma detect is used. If deasserted, the comma detect is bypassed.
RXDATA[63:0]	O	8, 16, 32, 64(2)	Up to eight bytes of decoded (8B/10B encoding) or encoded (8B/10B bypassed) received data at the user fabric.
RXDATAWIDTH[1:0]	I	2	(00, 01, 10, 11) Indicates width of FPGA parallel bus. See "Bus Interface" in Chapter 2.
RXDEC64B66BUSE	I	1	If asserted High, the 64B/B66B decoder is used. If deasserted, the 64B/66B decoder is bypassed.
RXDEC8B10BUSE	I	1	If asserted High, the 8B/10B decoder is used. If deasserted, the 8B/10B decoder is bypassed. CLK_COR_8B10B_DE = RXDEC8B10BUSE
RXDESCRAM64B66BUSE	I	1	If asserted High, the scrambler is used. If deasserted, the scrambler is bypassed.
RXDISPERR[7:0]	O	1, 2, 4, 8(1)	If 8B/10B encoding is enabled it indicates whether a disparity error has occurred on the serial line. Included in Byte-mapping scheme.
RXIGNOREBTF	I	1	If asserted High, the block type field (BTF) is ignored in the 64B/66B decoder. Instead of reporting an error, the block is passed on as is. If deasserted, unrecognized BTFs are marked as error blocks.
RXINTDATAWIDTH[1:0]	I	2	(00, 01, 10, 11) Sets the internal mode of the receive PCS, either 16, 20, 32, or 40 bit.
RXLOSSOFSYNC[1:0]	O	2	Bit 0 is always zero. Bit 1 indicates there is a 64B/66B Block Lock when deasserted to logic Low.
RXN	I	1	Serial differential port (FPGA external)
RXNOTINTABLE[7:0]	O	1, 2, 4, 8(1)	Status of encoded data when the data is not a valid character when asserted High. Applies to the byte-mapping scheme.
RXP	I	1	Serial differential port (FPGA external)
RXPOLARITY	I	1	Similar to TXPOLARITY, but for RXN and RXP. When deasserted, assumes regular polarity. When asserted, reverses polarity.
RXREALIGN	O	1	Signal from the PMA denoting that the byte alignment with the serial data stream changed due to a comma detection. Asserted High when alignment occurs.

Table 1-4: **Primitive Ports *(Continued)***

Port	I/O	Port Size	Definition
RXRECCLK	O	1	Clock recovered from the data stream and divided. Divide ratio depends on PMA_SPEED setting and/or PMA attributes. See Appendix C, "PMA Attribute Programming Bus."
RXRESET	I	1	Synchronous RX system reset that "recenters" the receive elastic buffer. It also resets 8B/10B decoder, comma detect, channel bonding, clock correction logic, and other internal receive registers. It does not reset the receiver PLL.
RXRUNDISP[7:0]	O	1, 2, 4, 8(1)	Signals the running disparity (`0` = negative, `1` = positive) in the received serial data. If 8B/10B encoding is bypassed, it remains as the second bit received (Bit "b") of the 10-bit encoded data.
RXSLIDE	I	1	Enables the "slip" of the detection block by 1 bit. To enable a slide of 1 bit, it increments from a lower bit to a higher bit. This signal must be asserted and then deasserted synchronous to RXUSRCKLK2. RXSLIDE must be held Low for at least two clock cycles before being asserted High again.
RXUSRCLK	I	1	Clock from a DCM or a BUFG that is used for reading the RX elastic buffer. It also clocks CHBONDI and CHBONDO in and out of the transceiver. Typically, the same as TXUSRCLK. ***Note:*** RXUSRCLK and RXUSRCLK2 should be 180° out of phase from each other.
RXUSRCLK2	I	1	Clock output from a DCM that clocks the receiver data and status between the transceiver and the FPGA fabric. Typically, the same as TXUSRCLK2. ***Note:*** RXUSRCLK and RXUSRCLK2 should be 180° out of phase from each other.
TXBUFERR	O	1	Provides status of the transmission FIFO. If asserted High, an overflow/underflow has occurred. When this bit becomes set, it can only be reset by asserting TXRESET.
TXBYPASS8B10B[7:0]	I	8	If TXENC8B10BUSE = 1 and TXENC64B66BUSE = 0 (8B/10B encoder enabled and 64B/66B encoder disabled), each bit of TXBYPASS8B10B[7:0] controls the bypass of the corresponding TXDATA byte; an asserted bit bypasses encoding for the data in the corresponding byte lane. If TXENC8B10BUSE = 0 and TXENC64B66BUSE = 1 (8B/10B encoder disabled and 64B/66B encoder enabled), TXBYPASS8B10B[2:0] bits are used for additional 64B / 66B encoder block bypass control. TXBYPASS8B10B[7:3] bits are not relevant in this particular configuration. Bits [2:1] carry the substitute sync header (SH[1:0]) for the block bypass operation; bit [0] is asserted for each block that the user wants to bypass.

Table 1-4: **Primitive Ports *(Continued)***

Port	I/O	Port Size	Definition
TXCHARDISPMODE[7:0]	I	1, 2, 4, 8(1)	If 8B/10B encoding is enabled, this bus determines what mode of disparity is to be sent. When 8B/10B is bypassed, this becomes the last bit transmitted (Bit "a") of the 10-bit encoded TXDATA bus section (see Table 2-6, page 45) for each byte specified by the byte-mapping. The bits have no meaning if TXENC8B10BUSE is deasserted.
TXCHARDISPVAL[7:0]	I	1, 2, 4, 8(1)	If 8B/10B encoding is enabled, this bus determines what type of disparity is to be sent. When 8B/10B is bypassed, this becomes the ninth bit transmitted (Bit "b") of the 10-bit encoded TXDATA bus section (see Table 2-6, page 45) for each byte specified by the byte-mapping section. The bits have no meaning if TXENC8B10BUSE is deasserted.
TXCHARISK[7:0]	I	1, 2, 4, 8(1)	If TXENC8B10BUSE = 1 (8B/10B encoder enable), then TXCHARISK[7:0] signals the K-definition of the TXDATA byte in the corresponding byte lane. (1 indicates that the byte is a K character; 0 indicates that the byte is a data character) If TXENC64B66BUSE = 1 (64B/66B encoder enable), then TXCHARISK[3:0] signals the block-formatting definitions of TXDATA (1 indicates that the byte is a control character; 0 indicates that the byte is a data character). TXCHARISK[7:4] bits are not relevant in this particular configuration.
TXDATA[63:0]	I	8, 16, 32, 64(2)	Transmit data from the FPGA user fabric that can be 1, 2, 4, or 8 bytes wide, depending on the primitive used. TXDATA[7:0] is always the first byte transmitted. The position of the first byte depends on selected TX data path width.
TXDATAWIDTH[1:0]	I	2	(00, 01, 10, 11) Indicates width of FPGA parallel bus. See "Bus Interface" in Chapter 2.
TXENC64B66BUSE	I	1	If asserted High, the 64B/66B encoder is used. If deasserted, the 64/66 encoder is bypassed.
TXENC8B10BUSE	I	1	If asserted High, the 8B/10B encoder is used. If deasserted, the 8B/10Bencoder is bypassed.
TXGEARBOX64B66BUSE	I	1	If asserted High, the 64B/66B gearbox is used. If deasserted, the 64/66 gearbox is bypassed. TXSCRAM64B66USE = TXGEARBOX64B66BUSE
TXINHIBIT	I	1	If asserted High, the TX differential pairs are forced to be a constant 1/0. TXN = 1, TXP = 0
TXINTDATAWIDTH[1:0]	I	2	(00, 01, 10, 11) Indicates internal data width (see Table 2-4, page 42).
TXKERR[7:0]	O	1, 2, 4, 8(1)	Indicates even boundary for bypassing in 64B/66B mode.
TXN	O	1	Transmit differential port (FPGA external)

Table 1-4: **Primitive Ports *(Continued)***

Port	I/O	Port Size	Definition
TXOUTCLK	O	1	Synthesized Clock from RocketIO X transmitter. This clock can be scaled (e.g., for 64B/66B) relative to BREFCLK, depending upon the specific operating mode of the transmitter.
TXP	O	1	Transmit differential port (FPGA external)
TXPOLARITY	I	1	Specifies whether or not to invert the final transmitter output. Able to reverse the polarity on the TXN and TXP lines. Deasserted sets regular polarity. Asserted reverses polarity.
TXRESET	I	1	Synchronous TX system reset that "recenters" the transmit elastic buffer. It also resets 8B/10B encoder and other internal transmission registers. It does not reset the transmission PLL.
TXRUNDISP[7:0]	O	1, 2, 4, 8(1)	Signals the running disparity for its corresponding byte, after that byte is encoded. Zero equals negative disparity and positive disparity for a one. This is also overloaded to be the data output bus of the PMA attribute bus. See Appendix C, "PMA Attribute Programming Bus"
TXSCRAM64B66BUSE	I	1	If asserted High, the 64B/66B scrambler is used. If deasserted, the 64B/66B scrambler is bypassed. TXSCRAM64B66USE = TXGEARBOX64B66BUSE
TXUSRCLK	I	1	Clock output from a DCM that is clocked with the REFCLK (or other reference clock). This clock is used for writing the TX buffer and is frequency-locked to the REFCLK. ***Note:*** TXUSRCLK and TXUSRCLK2 should be 180° out of phase from each other.
TXUSRCLK2	I	1	Clock output from a DCM that clocks transmission data and status and reconfiguration data between the transceiver an the FPGA fabric. The ratio between the TXUSRCLK and TXUSRCLK2 depends on the width of the TXDATA. ***Note:*** TXUSRCLK and TXUSRCLK2 should be 180° out of phase from each other.

Notes:

1. Port size depends on which primitive is used (1, 2, 4, 8 byte).
2. Port size depends on which primitive is used (8, 16, 32, 64 byte).

Primitive Attributes

The primitives also contain attributes set by default to specific values controlling each specific primitive's protocol parameters. Included are channel-bonding settings (for primitives supporting channel bonding), and clock correction sequences. Table 1-5 shows a brief description of each attribute. See Appendix F, "Modifiable Attributes" (Table F-1through Table F-6) for the default values of each primitive.

Table 1-5: **RocketIO X Transceiver Attributes**

Attribute	Type	Description
ALIGN_COMMA_WORD	Integer	Integer (1, 2, 4) controls the alignment of detected commas within the transceiver's 4-byte wide data path. See Table 2-9, page 51.
CHAN_BOND_64B66B_SV	Boolean	TRUE/FALSE. This signal is reserved for future use and must be held to FALSE.
CHAN_BOND_LIMIT	Integer	Integer 1-63 that defines maximum number of bytes a slave receiver can read following a channel bonding sequence and still successfully align to that sequence. ***Note:*** This attribute must be set to 16.
CHAN_BOND_MODE	String	STRING OFF, MASTER, SLAVE_1_HOP, SLAVE_2_HOPS **OFF:** No channel bonding involving this transceiver. **MASTER:** This transceiver is master for channel bonding. Its CHBONDO port directly drives CHBONDI ports on one or more SLAVE_1_HOP transceivers. **SLAVE_1_HOP:** This transceiver is a slave for channel bonding. SLAVE_1_HOP's CHBONDI is directly driven by a MASTER transceiver CHBONDO port. SLAVE_1_HOP's CHBONDO port can directly drive CHBONDI ports on one or more SLAVE_2_HOPS transceivers. **SLAVE_2_HOPS:** This transceiver is a slave for channel bonding. SLAVE_2_HOPS CHBONDI is directly driven by a SLAVE_1_HOP CHBONDO port.
CHAN_BOND_ONE_SHOT	Boolean	FALSE/TRUE that controls repeated execution of channel bonding. **FALSE:** Master transceiver initiates channel bonding whenever possible (whenever channel-bonding sequence is detected in the input) as long as input ENCHANSYNC is High and RXRESET is Low. **TRUE:** Master transceiver initiates channel bonding only the first time it is possible (channel bonding sequence is detected in input) following negated RXRESET and asserted ENCHANSYNC. After channel-bonding alignment is done, it does not occur again until RXRESET is asserted and negated, or until ENCHANSYNC is negated and reasserted. Slave transceivers should always have CHAN_BOND_ONE_SHOT set to FALSE.

Table 1-5: **RocketIO X Transceiver Attributes *(Continued)***

Attribute	Type	Description
CHAN_BOND_SEQ_1_*[10:0]	11-bit vector	These define the channel bonding sequence. The usage of these vectors also depends on CHAN_BOND_SEQ_LEN and CHAN_BOND_SEQ_2_USE. See "Transmitting Vitesse Channel Bonding Sequence," page 48, for format.
CHAN_BOND_SEQ_1_MASK[3:0]	4-bit vector	Each bit of the mask determines if that particular sequence is detected regardless of its value. If bit 0 is High, then CHAN_BOND_SEQ_1_1 is matched regardless of its value.
CHAN_BOND_SEQ_2_*[10:0]	11-bit vector	These define the channel bonding sequence: The usage of these vectors also depends on CHAN_BOND_SEQ_LEN and CHAN_BOND_SEQ_2_USE. See "Receiving Vitesse Channel Bonding Sequence," page 48, for format.
CHAN_BOND_SEQ_2_MASK[3:0]	4-bit vector	Each bit of the mask determines if that particular sequence is detected regardless of its value. If bit 0 is High, then CHAN_BOND_SEQ_2_1 is matched regardless of its value.
CHAN_BOND_SEQ_2_USE	Boolean	FALSE/TRUE that controls use of second channel bonding sequence. **FALSE:** Channel bonding uses only one channel bonding sequence defined by CHAN_BOND_SEQ_1_1 ... 4, **or** one 8-byte sequence defined by CHAN_BOND_SEQ_1_X and CHAN_BOND_SEQ_2_X in combination. **TRUE:** Channel bonding uses two channel bonding sequences defined by CHAN_BOND_SEQ_1_1 ... 4 and CHAN_BOND_SEQ_2_1 ... 4, as further constrained by CHAN_BOND_SEQ_LEN.
CHAN_BOND_SEQ_LEN	Integer	Integer (1, 2, 3, 4, 8) defines length in bytes of channel bonding sequence. This defines the length of the sequence the transceiver matches to detect opportunities for channel bonding.
CLK_COR_8B10B_DE	Boolean	This signal selects if clock correction occurs relative to the encoded or decoded version of the 8B/10B stream. If set to TRUE, the decoded version is used. If set to FALSE, the encoded version is used. Must be set in conjunction with RXDEC8B10USE. CLK_COR_8B10B_DE = RXDEC8B10BUSE
CLK_COR_MAX_LAT	Integer	(0-63) Integer defines the upper bound of the receive FIFO. ***Note:*** This attribute is recommended to be set to 48.
CLK_COR_MIN_LAT	Integer	(0-63) Integer defines the lower bound of the receive FIFO. ***Note:*** This attribute is recommended to be set to 32.
CLK_COR_SEQ_1_*[10:0]	11-bit vector	These define the sequence for clock correction. The attribute used depends on the CLK_COR_SEQ_LEN and CLK_COR_SEQ_2_USE.
CLK_COR_SEQ_1_MASK[3:0]	4-bit vector	Each bit of the mask determines if that particular sequence is detected regardless of its value. If bit 0 is High, then CLK_COR_SEQ_1_1 is matched regardless of its value.
CLK_COR_SEQ_2_*[10:0]	11-bit vector	These define the sequence for clock correction. The attribute used depends on the CLK_COR_SEQ_LEN and CLK_COR_SEQ_2_USE.

Table 1-5: **RocketIO X Transceiver Attributes *(Continued)***

Attribute	Type	Description
CLK_COR_SEQ_2_MASK[3:0]	4-bit vector	Each bit of the mask determines if that particular sequence is detected regardless of its value. If bit 0 is High, then CLK_COR_SEQ_2_1 is matched regardless of its value.
CLK_COR_SEQ_2_USE	Boolean	FALSE/TRUE Control use of second clock correction sequence. **FALSE:** Clock correction uses only one clock correction sequence defined by CLK_COR_SEQ_1_1 ... 4, **or** one 8-byte sequence defined by CLK_COR_SEQ_1_X and CLK_COR_SEQ_2_X in combination. **TRUE:** Clock correction uses two clock correction sequences defined by CLK_COR_SEQ_1_1 ... 4 and CLK_COR_SEQ_2_1 ... 4, as further constrained by CLK_COR_SEQ_LEN.
CLK_COR_SEQ_DROP	Boolean	TRUE/FALSE. When asserted TRUE, the clock correction mode is via idle removal. When FALSE, the clock correction mode is via idle removal or insertion. ***Note:*** This attribute must be set to FALSE.
CLK_COR_SEQ_LEN	Integer	Integer (1, 2, 3, 4, 8) that defines the length of the sequence the transceiver matches to detect opportunities for clock correction. It also defines the size of the correction, since the transceiver executes clock correction by repeating or skipping entire clock correction sequences.
CLK_CORRECT_USE	Boolean	TRUE/FALSE controls the use of clock correction logic. **FALSE:** Permanently disable execution of clock correction (rate matching). Clock RXUSRCLK must be frequency-locked with RXRECCLK in this case. **TRUE:** Enable clock correction (normal mode).
COMMA_10B_MASK[9:0]	10-bit vector	These define the mask that is ANDed with the incoming serial-bit stream before comparison against PCOMMA_10B_VALUE and MCOMMA_10B_VALUE.
DEC_MCOMMA_DETECT	Boolean	TRUE/FALSE controls the raising of per-byte flag RXCHARISCOMMA on minus-comma.
DEC_PCOMMA_DETECT	Boolean	TRUE/FALSE controls the raising of per-byte flag RXCHARISCOMMA on plus-comma.
DEC_VALID_COMMA_ONLY	Boolean	TRUE/FALSE controls the raising of RXCHARISCOMMA on an invalid comma. **FALSE:** Raise RXCHARISCOMMA on: `xxx1111100` (if DEC_PCOMMA_DETECT is TRUE) and/or on: `xxx0000011` (if DEC_MCOMMA_DETECT is TRUE) or on 8B/10B translation commas regardless of the settings of the `xxx` bits. **TRUE:** Raise RXCHARISCOMMA only on valid characters that are in the 8B/10B translation.

Table 1-5: **RocketIO X Transceiver Attributes *(Continued)***

Attribute	Type	Description
MCOMMA_10B_VALUE[9:0]	10-bit vector	These define minus-comma for the purpose of raising RXCOMMADET and realigning the serial bit stream byte boundary. This definition does not affect 8B/10B encoding or decoding. Also see COMMA_10B_MASK.
MCOMMA_DETECT	Boolean	TRUE/FALSE indicates whether to raise or not raise the RXCOMMADET when minus-comma is detected.
PCOMMA_10B_VALUE[9:0]	10-bit vector	These define plus-comma for the purpose of raising RXCOMMADET and realigning the serial bit stream byte boundary. This definition does not affect 8B/10B encoding or decoding. Also see COMMA_10B_MASK.
PCOMMA_DETECT	Boolean	TRUE/FALSE indicates whether to raise or not raise the RXCOMMADET when plus-comma is detected.
PMA_PWR _CNTRL	Integer	This masks the startup sequence of the PMA and must always be set to all ones.
PMA_SPEED	String	(13_40) Selects the mode of the PMA. Refer to PMA section for the proper mode selection.
RX_BUFFER_USE	Boolean	TRUE/FALSE. Recommended to always be set to TRUE. Enables the use of the receive side buffer. When set to TRUE, the buffer is enabled.
RX_LOS_INVALID_INCR[7:0]	Integer	Power of two in a range of 1 to 128 that denotes the number of valid characters required to "cancel out" appearance of one invalid character for loss of sync determination.
RX_LOS_THRESHOLD	Integer	Power of two in a range of 4 to 512. When divided by RX_LOS_INVALID_INCR, denotes the number of invalid characters required to cause FSM transition to “sync lost” state.
RX_LOSS_OF_SYNC_FSM	Boolean	Undefined.
SH_CNT_MAX[7:0]	8-bit vector	8-bit binary; controls when the 64B/66B synchronization state machine enters synchronization. (max sync header count)
SH_INVALID_CNT_MAX[7:0]	8-bit vector	8-bit binary; controls when the 64B/66B synchronization state machine leaves synchronization. (max invalid sync header count)
TX_BUFFER_USE	Boolean	When set to TRUE, this enables the use of the transmit buffer.

Modifiable Attributes

As shown in Appendix F, "Modifiable Attributes" (Table F-1 through Table F-6) only certain attributes are modifiable for any primitive. These attributes help to define the protocol used by the primitive. Only the GT10_CUSTOM primitive allows the user to modify all of the attributes to a protocol not supported by another transceiver primitive. This allows for complete flexibility. The other primitives allow modification of the analog attributes of the serial data lines and several channel-bonding values.

Byte Mapping

Most of the 8-bit wide status and control buses correlate to a specific byte of the TXDATA or RXDATA. This scheme is shown in Table 1-6. This creates a way to tie all the signals together regardless of the data path width needed for the GT10_CUSTOM. All other primitives with specific data width paths and all byte-mapped ports are affected by this situation. For example, a 1-byte wide data path has only 1-bit control and status bits (TXCHARISK[0]) correlating to the data bits TXDATA[7:0].

Table 1-6: **Control/Status Bus Association to Data Bus Byte Paths**

Control/Status Bit	Data Bits
[0]	[7:0]
[1]	[15:8]
[2]	[23:16]
[3]	[31:24]
[4]	[39:32]
[5]	[47:40]
[6]	[55:48]
[7]	[64:56]

Chapter 2

Digital Design Considerations

The Physical Coding Sublayer (PCS) portion of the RocketIO X transceiver has been significantly updated relative to the RocketIO. The RocketIO X PCS supports 8B/10B and 64B/66B encode/decode, SONET compatibility, and generic data modes. The RocketIO X transceiver operates in four basic internal modes: 16 bit, 20 bit, 32 bit, and 40 bit. When accompanied by the predefined modes of the Physical Media Attachment (PMA), the user has a large combination of protocols and data rates from which to choose. With the custom RocketIO X transceiver, the user has an almost infinite amount of possibilities from which to choose in constructing the most advanced and easily configurable communication paths in the history of communication ICs.

The RocketIO X PCS also represents a shift in the configurability of transceivers. This allows the user to change not only speeds of the PMA in real time, but also protocols within the PCS. Internal data width, external data width, and data routing can all be configured on a clock-by-clock basis. With this advancement, users can initialize a communication channel at a low speed (for example, 2.5 Gb/s using 8B/10B (20 bit internal) and then auto-negotiate after the channel is stable to a 6.25 Gb/s speed using 64B/66B (32-bit internal).

Note: The information in this chapter is provided to RocketIO X users as a reference for understanding the individual attribute and control port settings within a primitive. Users have the choice of using the supported primitives in Table 1-3, page 25, and ignoring this chapter, or using this chapter to better understand PCS configuration and/or to modify attribute and port values to create a user transceiver configuration.

Top-Level Architecture

Transmit Architecture

The transmit architecture for the PCS is shown in Figure 2-1. For information about bypassing particular blocks, consult the block function section for that particular block.

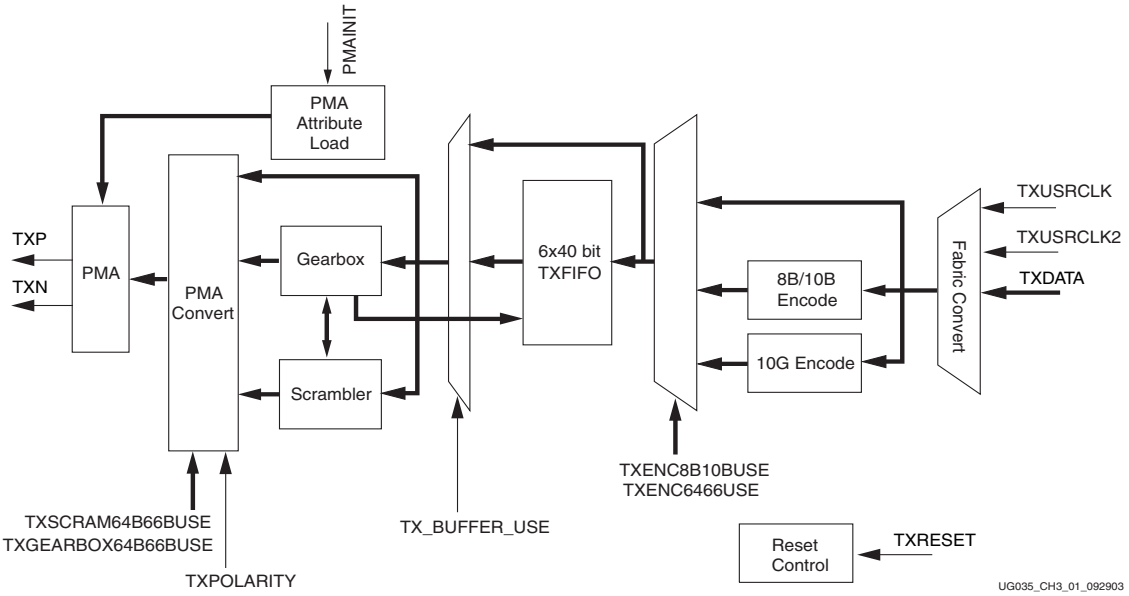


Figure 2-1: **Transmit Architecture**

Receive Architecture

The receive architecture for the PCS is shown in Figure 2-2. For information about bypassing particular blocks, consult the block function section for that particular block.

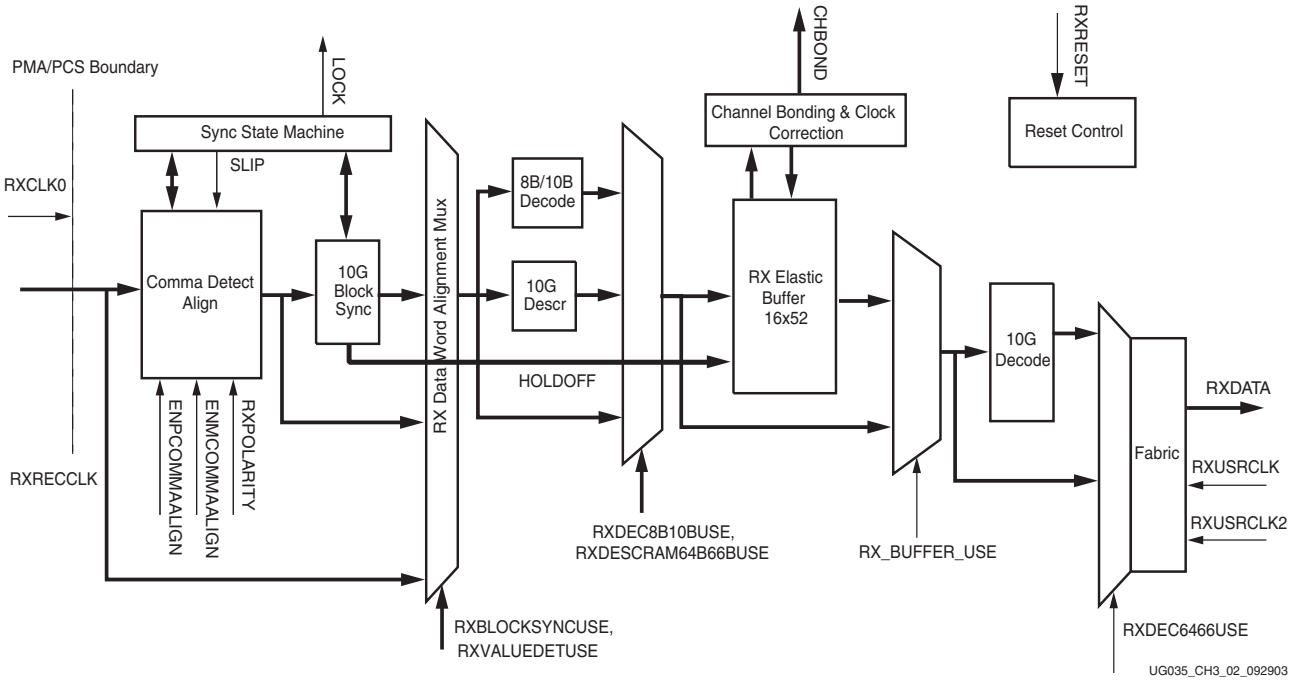


Figure 2-2: **Receive Architecture**

Operation Modes

Internally, there are four modes of operation within the PCS: 16 bit, 20 bit, 32 bit, and 40 bit.

The PCS fundamentally operates in either 2-byte mode, or 4-byte mode, with 2-byte mode corresponding to 16- and 20-bit mode, and with 4-byte mode corresponding to 32- and 40-bit mode. When in 2-byte mode, the external interface can either be one, two, or four bytes wide. When in 4-byte mode, the external interface can either be 4 or 8 bytes wide. It is not possible to have an internal 2-byte width and an 8-byte external interface. It is also not possible to have an internal 4-byte interface, along with a 1-byte external interface. See Table 2-1.

Table 2-1: **PCS Interface Choice**

Speed	2 Byte (internal mode)	4 Byte (internal mode)
2.488 Gb/s	recommended	do not use
5 - 6.25 Gb/s	do not use	recommended

A general guide to use is that 2-byte mode should be used in the PCS when the serial speed is below 5 Gb/s, and the 4-byte mode should be used when the serial speed is greater than 5 Gb/s. In 2-byte mode, the PCS processes 4-byte data every other byte. This is transparent to the user, but skews between transceivers result in larger bit skews at the transmit interface as compared to Virtex-II Pro transceivers. Any one of the three encoding schemes (8B/10B and 64B/66B encode/decode, SONET, and generic data modes) can be used in either 2- or 4-byte mode, with each block having a bypass ability.

For more information on setting the PCS mode, refer to the block functional definition of the bus interface in this guide.

Block Level Functions

Classification of Signals and Overloading

This section describes the pertinent signals at the interface of the PCS and how to prioritize them. For more information about a particular signal, refer to the I/O specification, or the particular block function of interest.

Static Signals (Control Inputs)

The following static signals are inputs that control the internal and external mode of operation in the PCS. Typically, these signals would be the first consideration after the mode of operation has been selected:

- RXDATAWIDTH[1:0]
- RXINTDATAWIDTH[1:0]
- TXDATAWIDTH[1:0]
- TXINTDATAWIDTH[1:0]

The following static signals are inputs that control the PCS interblock routing and bypass for particular blocks, which adjust the architecture of the PCS for the user's particular application:

- RXBLOCKSYNC64B66BUSE
- RXDEC64B66BUSE
- RXDEC8B10BUSE
- RXDESCRAM64B66BUSE
- RXCOMMADETUSE
- TXENC64B66BUSE
- TXENC8B10BUSE
- TXGEARBOX64B66BUSE
- TXSCRAM64B66BUSE

The following static signals are inputs that control various functions, but are usually set once at the beginning of a state machine, or after an auto-negotiation sequence. They are typically not altered on a clock-by-clock basis:

- ALIGN_COMMA_WORD
- ENMCOMMAALIGN
- ENPCOMMAALIGN
- RXPOLARITY
- TXPOLARITY
- PMAINIT
- RXIGNOREBTF
- PMARXLOCKSEL[1:0]

The following static signals are inputs that cause either major functional resets or are used in troubleshooting. These signals are mostly used at initialization, not during the functional operation of the circuit:

- LOOPBACK[1:0] (***Note***: *This signal can also be a dynamic signal*.)
- POWERDOWN
- RXRESET
- TXRESET
- TXINHIBIT

Dynamic Signals

The following dynamic signals indicate data received on the receive bus, along with status signals that indicate specific information about RXDATA. The set values of these signals define the application setup by the user and are the most important after the static signals are allocated:

- MCOMMA_10B_VALUE
- PCOMMA_10B_VALUE
- COMMA_10B_MASK
- RXCHARISCOMMA[7:0]
- RXCHARISK[7:0]

- RXDISPERR[7:0]
- RXNOTINTABLE[7:0]
- RXDATA[63:0]

The following dynamic signals indicate data to be transmitted on the transmit bus, along with status signals that indicate specific information about how TXDATA is to be handled while passing through the PCS. The set values of these signals define the application setup by the user and are the most important after the static signals are allocated:

- TXBYPASS8B10B[7:0]
- TXCHARDISPMODE[7:0]
- TXCHARDISPVAL[7:0]
- TXCHARISK[7:0]
- TXDATA[63:0]

The following dynamic signals indicate various status information about the current state or prior state of the PCS:

- CHBONDDONE, RXBUFSTATUS[1:0], RXCLKCORCNT[2:0]
- CHBONDO[4:0]
- PMARXLOCK
- RXLOSSOFSYNC[1:0]
- RXREALIGN
- RXCOMMADET
- TXBUFERR
- TXKERR[7:0]
- TXRUNDISP[7:0]
- RXRUNDISP[7:0]

The following dynamic signals control internal states of the PCS:

- RXSLIDE
- CHBONDI[4:0]

The following dynamic signals affect the control registers of the PMA:

- PMAREGADDR[5:0]
- PMAREGDATAIN[7:0]
- PMAREGRW
- PMAREGSTROBE

Bus Interface

Selecting the External Configuration (Fabric Interface)

By using the signals TXDATAWIDTH[1:0] and RXDATAWIDTH[1:0], the fabric interface can be determined.

Table 2-2: **Selecting the External Configuration**

RXDATAWIDTH/TXDATAWIDTH	Data Width	Internal Bus Requirements
2'b00	8/10 bit (1 byte)	16, 20 bit mode
2'b01	16/20 bit (2 byte)	16, 20 bit mode
2'b10	32/40 bit (4 byte)	16, 20, 32, 40 bit mode
2'b11	64/80 bit (8 byte)	32, 40 bit mode

Selecting the Internal Configuration

Table 2-3: **Selecting the Internal Configuration**

RXINTDATAWIDTH/TXINTDATAWIDTH	Internal Data Width
2'b00	16 bit
2'b01	20 bit
2'b10	32 bit
2'b11	40 bit

Clock Ratio

USRCLK2 clocks the data buffers. The ability to send parallel data to the transceiver at four different widths requires the user to change the frequency of USRCLK2. This creates a frequency ratio between USRCLK and USRCLK2. The falling edges of the clocks must align. See Table 2-4.

Table 2-4: **Data Width Clock Ratios**

Fabric Data Width	Frequency Ratio of USRCLK\USRCLK2	
	2-Byte Internal Data Width	4-Byte Internal Data Width
1 byte	1:2 (1)	N/A
2 byte	1:1	N/A
4 byte	2:1 (1)	1:1
8 byte	N/A	2:1 (1)

Notes:

1. Each edge of slower clock must align with falling edge of faster clock.

8B/10B

Note: In the RocketIO transceiver, the most significant byte was sent first; in the RocketIO X transceiver the least significant byte is sent first.

The following sections categorize the ports and attributes of the transceiver according to specific functionality including 8B/10B encoding/decoding, 64B/66B encoding/decoding, SERDES alignment, clock correction (clock recovery), channel bonding, fabric interface, and other signals.

The 8B/10B encoding translates an 8-bit parallel data byte to be transmitted into a 10-bit serial data stream. This conversion and data alignment are shown in Figure 2-3. The serial port transmits the least significant bit of the 10-bit data, "a" first and proceeds to "j". This allows data to be read and matched to the form shown in Appendix B, "8B/10B Valid Characters." .

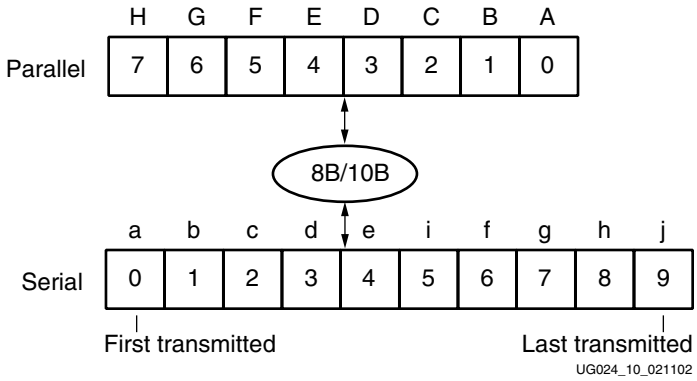


Figure 2-3: **8B/10B Parallel-to-Serial Conversion**

The serial data bit sequence is dependent on the width of the parallel data. The least significant byte is always sent first regardless of the whether 1-byte, 2-byte, 4-byte or 8-byte paths are used. The most significant byte is always last. Figure 2-4 shows a case when the serial data corresponds to each byte of the parallel data. TXDATA[7:0] is serialized and sent out first followed by TXDATA[15:8], TXDATA[23:16], and finally TXDATA[31:24]. The 2-byte path transmits TXDATA[7:0] and then TXDATA[15:8].

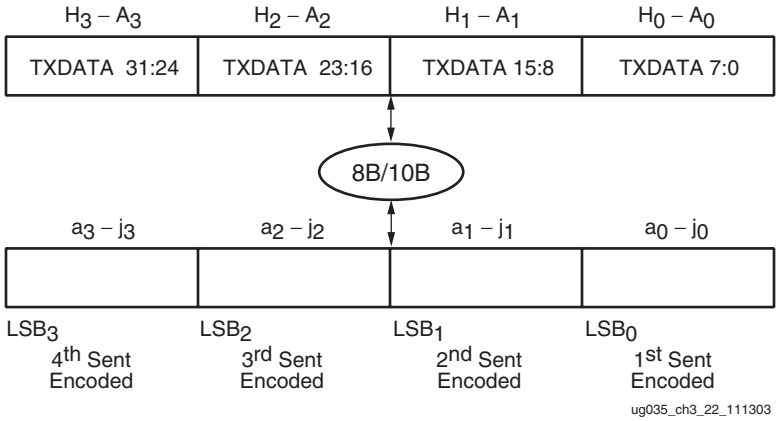


Figure 2-4: **4-Byte Serial Structure**

Encoder

A bypassable 8B/10B encoder is included in the transmitter. The encoder uses the same 256 data characters and 12 control characters (shown in Appendix B, "8B/10B Valid Characters") that are used for Gigabit Ethernet, XAUI, Fibre Channel, and InfiniBand.

The encoder accepts 8 bits of data along with a K-character signal for a total of 9 bits per character applied. If the K-character signal is High, the data is encoded into one of the 12 possible K-characters available in the 8B/10B code. If the K-character input is Low, the 8 bits are encoded as standard data.

There are two ports that enable the 8B/10B encoding in the transceiver. The TXBYPASS8B10B is a byte-mapped port that is 1, 2, 4 or 8 bits depending on the data width of the transceiver primitive being used. These bits correlate to each byte of the data path. To enable the 8B/10B encoding of the transmitter, these bits should be set to a logic 0. In this mode, the transmit data that is input to the TXDATA port is non-encoded data of either 8, 16, 32, or 64 bits wide. However, if other encoding schemes are preferred, the encoder capabilities are bypassed by setting all bits to a logic 1. The extra bits are fed through the TXCHARDISPMODE and TXCHARDISPVAL buses.

TXCHARDISPVAL and TXCHARDISPMODE

TXCHARDISPVAL and TXCHARDISPMODE are dual-purpose ports for the transmitter depending whether 8B/10B encoding is done. Table 2-6 shows this dual functionality. When encoding is enabled, these ports function as byte-mapped control ports controlling the running disparity of the transmitted serial data (Table 2-5).

Table 2-5: **Running Disparity Control**

{txchardispmode, txchardispval}	Function
00	Maintain running disparity normally
01	Invert normally generated running disparity before encoding this byte
10	Set negative running disparity before encoding this byte
11	Set positive running disparity before encoding this byte

In the encoding configuration, the disparity of the serial transmission can be controlled with the TXCHARDISPVAL and TXCHARDISPMODE ports. When TXCHARDISPMODE is set to a logic 1, the running disparity is set before encoding the specific byte. TXCHARDISPVAL determines if the disparity is negative (set to a logic 0) or positive (set to a logic 1).

When TXCHARDSIPMODE is set to a logic 0, the running disparity is maintained if TXCHARDISPVAL is also set to a logic 0. However, the disparity is inverted before encoding the byte when the TXCHARDISPVAL is set to a logic 1.

Most applications use the mode where both TXCHARDISPMODE and TXCHARDISPVAL are set to logic 0. Some applications can use other settings if special running disparity configurations are required, such as in the "Vitesse Disparity Example," page 48.

In the bypassed configuration, TXCHARDISPMODE[0] becomes bit 9 of the 10 bits of encoded data (TXCHARDISPMODE[1:7] are bits 19, 29, 39, 49, 59, 69, and 79 in the 20-bit and 40-bit and 80-bit wide buses). TXCHARDISPVAL becomes bits 8, 18, 28, 38, 48, 58, 68, and 78 of the transmit data bus while the TXDATA bus completes the bus. See Table 2-6.

Table 2-6: **8B/10B Bypassed Signal Significance**

Signal		Function	
TXBYPASS8B10B(1)	0	8B/10B encoding is enabled (not bypassed)	
	1	8B/10B encoding bypassed (disabled)	
TXCHARDISPMODE, TXCHARDISPVAL		**Function, 8B/10B Enabled**	**Function, 8B/10B Bypassed**
	00	Maintain running disparity normally	Part of 10-bit encoded byte (see Figure 2-5): TXCHARDISPMODE[0], (or: [1] / [2] / [3] /[4]/[5]/[6]/[7]) TXCHARDISPVAL[0], (or: [1] / [2] / [3] /[4]/[5]/[6]/[7]) TXDATA[7:0] (or: [15:8] / [23:16] / [31:24]/ [39:32]/[47:40]/[55:48]/[63:56]))
	01	Invert the normally generated running disparity before encoding this byte.	
	10	Set negative running disparity before encoding this byte.	
	11	Set positive running disparity before encoding this byte.	
TXCHARISK		Received byte is a K-character	Unused

Notes:
1. If 8B/10B is bypassed, this port can be defined if 64B/66B encoding is used.

During transmit, while 8B/10B encoding is enabled, the disparity of the serial transmission can be controlled with the TXCHARDISPVAL and TXCHARDISPMODE ports. When 8B/10B encoding is bypassed, these bits become Bits “b” and “a,” respectively, of the 10-bit encoded data that the transceiver must transmit to the receiving terminal. Figure 2-5 illustrates the TX data map during 8B/10B bypass.

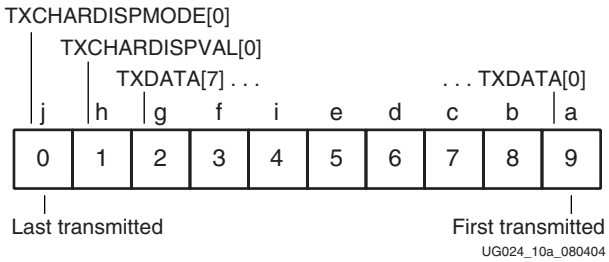


Figure 2-5: **10-Bit TX Data Map with 8B/10B Bypassed**

TXCHARISK

TXCHARISK is a byte-mapped control port that is only used when the 8B/10B encoder is implemented. This port indicates whether the byte of TXDATA is to be encoded as a control (K) character when asserted and data character when de-asserted. When 8B/10B encoding is bypassed this port is undefined.

TXRUNDISP

TXRUNDISP is a status port that is byte-mapped to the TXDATA. This port indicates the running disparity after this byte of TXDATA is encoded. When asserted, the disparity is positive. When de-asserted, the disparity is negative.

Decoder

An optional 8B/10B decoder is included in the receiver. A programmable option allows the decoder to be bypassed. When the 8B/10B decoder is bypassed, the 10-bit character order is shown in Figure 2-6 for a graphical representation of the received 10-bit character.

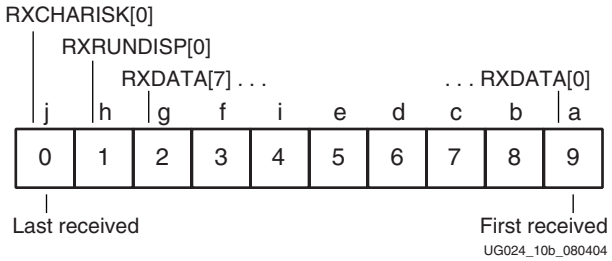


Figure 2-6: **10-Bit RX Data Map with 8B/10B Bypassed**

The decoder uses the same table (see Appendix B, "8B/10B Valid Characters") that is used for Gigabit Ethernet, Fibre Channel, and InfiniBand. In addition to decoding all data and K-characters, the decoder has several extra features. The decoder separately detects both "disparity errors" and "out-of-band" errors. A *disparity error* occurs when a 10-bit character is received that exists within the 8B/10B table, but has an incorrect disparity. An *out-of-band error* occurs when a 10-bit character is received that does not exist within the 8B/10B table. It is possible to obtain an out-of-band error without having a disparity error, or more commonly, a disparity error is possible without an out-of-band error. The proper disparity is always computed for both legal and illegal characters. The current running disparity is available at the RXRUNDISP signal.

The 8B/10B decoder performs a unique operation if out-of-band data is detected. If out-of-band data is detected, the decoder signals the error and passes the illegal 10-bits through and places them on the outputs. This can be used for debugging purposes if desired.

The decoder also signals reception of one of the 12 valid K-characters. In addition, a programmable comma detect is included. The comma detect signal registers a comma on the receipt of any comma+, comma–, or both. Since the comma is defined as a 7-bit character, this includes several out-of-band characters. Another option allows the decoder to detect only the three defined commas (K28.1, K28.5, and K28.7) as comma+, comma–, or both. In total, there are six possible options, three for valid commas and three for "any comma."

Note that all bytes (1, 2, 4 or 8) at the RX FPGA interface each have their own individual 8B/10B indicators (K-character, disparity error, out-of-band error, current running disparity, and comma detect).

During receive, while 8B/10B decoding is enabled, the running disparity of the serial transmission can be read by the transceiver from the RXRUNDISP port, while the RXCHARISK port indicates presence of a K-character. When 8B/10B decoding is bypassed, these bits remain as Bits "b" and "a," respectively, of the 10-bit encoded data that the transceiver passes on to the user logic. Table 2-7 illustrates the RX data map during 8B/10B bypass.

Table 2-7: **8B/10B Bypassed Signal Significance**

Signal		Function	
RXCHARISK		Received byte is a K-character	Part of 10-bit encoded byte (see Figure 2-6): RXCHARISK[0], (or: [1] / [2] / [3] /[4]/[5]/[6]/[7]) RXRUNDISP[0], (or: [1] / [2] / [3] /[4]/[5]/[6]/[7]) RXDATA[7:0] (or: [15:8] / [23:16] / [31:24]/ [39:32]/[47:40]/[55:48]/[63:56])
RXRUNDISP	0	Indicates running disparity is NEGATIVE	
	1	Indicates running disparity is POSITIVE	
RXDISPERR		Disparity error occurred on current byte	Unused

RXCHARISK and RXRUNDISP

RXCHARISK and RXRUNDISP are dual-purpose ports for the receiver depending whether 8B/10B decoding is enabled. Figure 2-8 shows this dual functionality. When decoding is enabled, these ports function as byte-mapped status ports of the received data.

In the encoding configuration, when RXCHARISK is asserted that byte of the received data is a control (K) character. Otherwise, the received byte of data is a data character. (See Appendix B, "8B/10B Valid Characters"). The RXRUNDISP port indicates the disparity of the received byte is either negative or positive. RXRUNDISP is asserted to indicate positive disparity. This is used in cases like the "Vitesse Disparity Example," page 48.

In the bypassed configuration, RXCHARISK and RXRUNDISP are additional data bits for the 10-, 20-, 40-, or 80-bit buses. This is similar to the transmit side. RXCHARISK[0:7] relates to bits 9, 19, 29, 39, 49, 59, 69, and 79 while RXRUNDISP pertains to bits 8, 18, 28, 38, 48, 58, 68, and 78 of the data bus. See Figure 2-8.

RXDISPERR

RXDISPERR is a status port for the receiver that is byte-mapped to the RXDATA. When a bit is asserted, a disparity error occurred on the received data. This usually indicated that the data is corrupt by bit errors, transmission of an invalid control character, or for cases when normal disparity is not required such as in the "Vitesse Disparity Example," page 48.

RXNOTINTABLE

RXNOTINTABLE is asserted whenever the received data is not in the 8B/10B tables. The data received on bytes marked by RXNOTINTABLE are invalid. This port is also byte-mapped to RXDATA and is only used when the 8B/10B decoder is enabled.

Vitesse Disparity Example

To support other protocols, the transceiver can affect the disparity mode of the serial data transmitted. For example, Vitesse channel-to-channel alignment protocol sends out:

```
K28.5+ K28.5+ K28.5- K28.5-
```

or

```
K28.5- K28.5- K28.5+ K28.5+
```

Instead of:

```
K28.5+ K28.5- K28.5+ K28.5-
```

or

```
K28.5- K28.5+ K28.5- K28.5+
```

The logic must assert TXCHARDISPVAL to cause the serial data to send out two negative running disparity characters.

Transmitting Vitesse Channel Bonding Sequence

```
TXBYPASS8B10B
| TXCHARISK
| | TXCHARDISPMODE
| | | TXCHARDISPVAL
| | | | TXDATA
| | | | |
0 1 0 0 10111100    K28.5+ (or K28.5-)
0 1 0 1 10111100    K28.5+ (or K28.5-)
0 1 0 0 10111100    K28.5- (or K28.5+)
0 1 0 1 10111100    K28.5- (or K28.5+)
```

The RocketIO X core receives this data but must have the CHAN_BOND_SEQ set with the `disp_err` bit set High for the cases when TXCHARDISPVAL is set High during data transmission.

Receiving Vitesse Channel Bonding Sequence

On the RX side, the definition of the channel bonding sequence uses the `disp_err` bit to specify the flipped disparity.

```
                  10-bit literal value
                  | disp_err
                  | | char_is_k
                  | | | 8-bit_byte_value
                  | | | |
CHAN_BOND_SEQ_1_1 = 0 0 1 10111100   matches K28.5+ (or K28.5-)
CHAN_BOND_SEQ_1_2 = 0 1 1 10111100   matches K28.5+ (or K28.5-)
CHAN_BOND_SEQ_1_3 = 0 0 1 10111100   matches K28.5- (or K28.5+)
CHAN_BOND_SEQ_1_4 = 0 1 1 10111100   matches K28.5- (or K28.5+)
CHAN_BOND_SEQ_LEN = 4
CHAN_BOND_SEQ_2_USE = FALSE
```

Comma Detection

Summary

Comma detection has been expanded beyond 10-bit symbol detection and alignment to include 8-bit symbol detection and alignment for 16-, 20-, 32-, and 40-bit paths. The ability to detect symbols, and then either align to 1-word, 2-word, or 4-word boundaries is included. The RXSLIDE input allows the user to “slide” or “slip” the alignment by one bit in each 16-, 20-, 32- and 40-bit mode at any time for SONET applications.

The following signals/attributes affect the function of the comma detection block:

- RXCOMMADETUSE
- ENMCOMMAALIGN
- ENPCOMMAALIGN
- ALIGN_COMMA_WORD[1:0]
- MCOMMA_10B_VALUE[9:0]
- DEC_MCOMMA_DETECT
- PCOMMA_10B_VALUE[9:0]
- DEC_PCOMMA_DETECT
- COMMA_10B_MASK[9:0]
- RXSLIDE
- RXINTDATAWIDTH[1:0]

Bypass

By deasserting RXCOMMADETUSE Low, symbol detection is not enabled. If RXCOMMADETUSE is asserted High, symbol detection takes place.

Symbol Detection

By using the signals MCOMMA_10B_VALUE, DEC_MCOMMA_DETECT, PCOMMA_10B_VALUE, DEC_PCOMMA_DETECT, and COMMA_10B_MASK any 8-bit or 10-bit symbol detection can take place for two different symbol values.

To detect a 10-bit symbol COMMA_10B_MASK[9:0] should initially be set to 10’b11111_11111. Any bit can be changed to further affect the masking capability.

To detect an 8-bit symbol, the COMMA_10B_MASK[9:0] should be set to 10’b00_1111_1111. The first two bits must be set to zero. Any of the last 8 bits can be altered to change the mask further.

The MCOMMA_10B_VALUE[9:0] and PCOMMA_10B_VALUE[9:0] fields indicate the comma symbol definitions to be used by the comparison logic, i.e., the templates against which incoming data is compared in the search for commas to establish alignment.

The DEC_MCOMMA_DETECT and DEC_PCOMMA_DETECT indicate which symbol should be compared to the incoming data for alignment. See Table 2-8.

Table 2-8: **Symbol Detection**

MCOMMA_DETECT	PCOMMA_DETECT	Function
0	0	No symbol detection takes place.
0	1	RXCOMMADET is asserted if the incoming data is compared and aligned to the symbol defined by PCOMMA_10B_VALUE.
1	0	RXCOMMADET is asserted if the incoming data is compared and aligned to the symbol defined by MCOMMA_10B_VALUE.
1	1	RXCOMMADET is asserted if the incoming data is compared and aligned to the symbol defined by PCOMMA_10B_VALUE or MCOMMA_10B_VALUE.

Setting MCOMMA_10B_VALUE, PCOMMA_10B_VALUE, and COMMA_10B_MASK (Special Note)

The attributes, MCOMMA_10B_VALUE, PCOMMA_10B_VALUE, and COMMA_10B_MASK are used by the MGT to indicate to the comma detection block the values to which the block should be aligned. Once set to a value, the comma detection block searches the data stream for these values and aligns the pipeline to the position where the value was detected in the data stream. Virtex-II Pro X users need to note that these values are reversed relative to Virtex-II Pro devices. The reason for this is that while Virtex-II Pro devices support mainly 8B/10B applications, Virtex-II Pro X devices can support many applications and use a more general approach.

Figure 2-7 shows a Virtex-II Pro X 8B/10B comma detection example relative to the data stream received at the PCS/PMA interface on the receive side. Note that with Virtex-II Pro devices, the M/PCOMMA_10B_VALUE[9:0] is set to 10'b0011111010, whereas in Virtex-II Pro X devices the value is set to 10'b0101111100. This also follows for the COMMA_10B_MASK, which in Virtex-II Pro devices is set to 10'b1111111000, whereas in Virtex-II Pro X devices, it is set to 10'b0001111111.

With this change, the block can be considered more of a value detection block, rather than a comma detection block. To detect values listed in the 8B/10B8B/10B tables, simply reverse the values in the tables. To detect SONET type values, the exact value can be used without reversal.

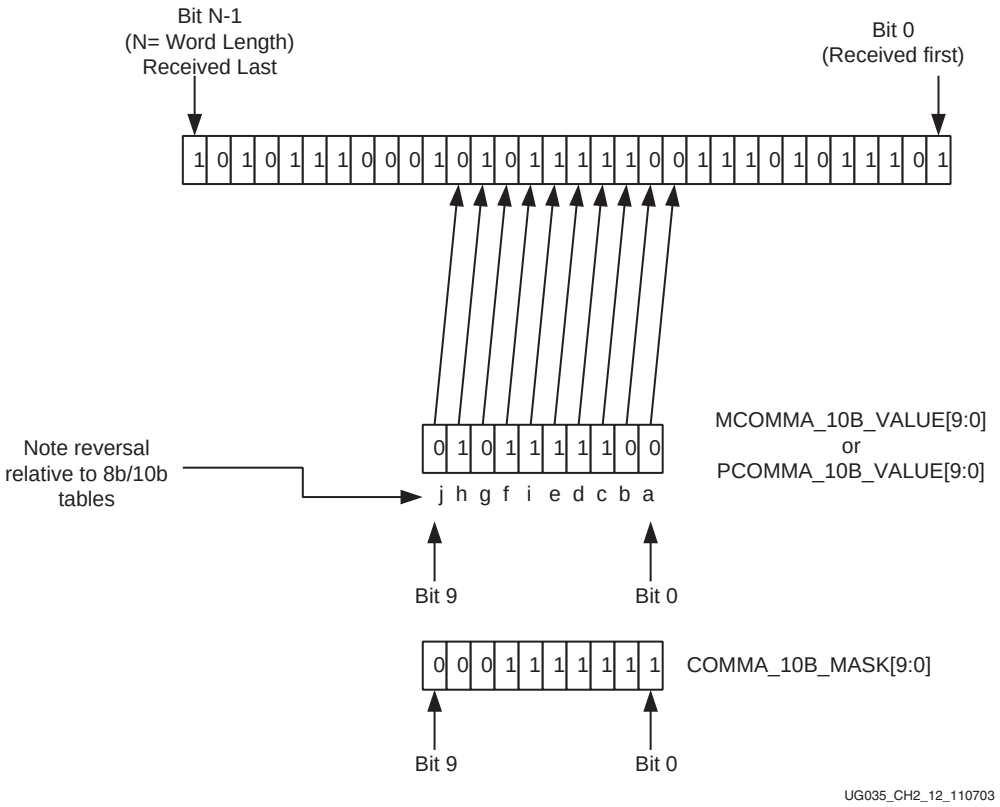


Figure 2-7: **8B/10B Comma Detection Example**

Alignment

After the positive symbol or the negative symbol is detected, the data is aligned to that symbol. By using the signals ENMCOMMAALIGN, ENPCOMMAALIGN, ALIGN_COMMA_WORD, and RXSLIDE, alignment can be completely controlled for all data pipeline configurations. See Table 2-9.

Table 2-9: **Data Alignment**

ENMCOMMAALIGN	ENPCOMMAALIGN	Function(1)
0	0	No alignment takes place.
0	1	If a positive symbol is detected, alignment takes place at that symbol location.
1	0	If a negative symbol is detected, alignment takes place at that symbol location.

Table 2-9: **Data Alignment**

ENMCOMMAALIGN	ENPCOMMAALIGN	Function(1)
1	1	If a negative or positive symbol is detected, alignment takes place at that symbol location.

Notes:

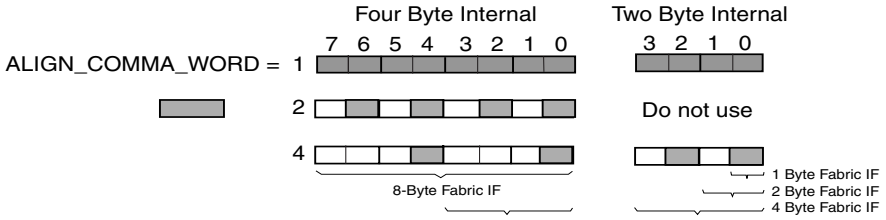
1. The symbol mentioned is defined by P/MCOMMA_10B_VALUE.

ALIGN_COMMA_WORD

The attribute ALIGN_COMMA_WORD controls when realignment takes place when the difference between symbols is on a byte-by-byte basis. If the current position of the symbol detected is some fraction of a byte different than the previous symbol position, alignment takes place regardless of the setting of ALIGN_COMMA_WORD.

- There are three options for ALIGN_COMMA_WORD: 1 byte, 2 byte, and 4 byte. When ALIGN_COMMA_WORD is set to a 1, the detection circuit allows detection symbols in contiguous bytes. When ALIGN_COMMA_WORD is set to a 2, the detection circuit allows detection symbols every other byte. When ALIGN_COMMA_WORD is set to a 4, the detection circuit allows detection symbols every fourth byte.

	16/20	32/40
1	byte alignment	byte alignment
2	N/A	2-byte alignment
4	2-byte alignment	4-byte alignment



Note: Shaded blocks indicate where the comma can align to.

RXSLIDE

RXSLIDE can be used to "slide" the aligned data by one bit. The RXSLIDE function when asserted High, increments the alignment by one bit, until it reaches the most significant bit, equal to the maximum word length –1. When RXSLIDE is asserted High, it must be asserted Low for two clock periods before it can be asserted High again. This functionality can be used for applications such as SONET.

64B/66B

Encoder

Bypassing

There are two types of bypassing regarding the 64B/66B encoder. The encoder block can either be entirely bypassed, or the 64B/66B encoder can be used and can be bypassed on a clock-by-clock basis.

If TXENC64B66BUSE is deasserted Low, the entire 64B/66B encoder is not used. If encoding is done in the fabric, the sync header [0:1] must be placed at TXCHARDISPVAL[0] and TXCHARDISPMODE[0] with the 32 TXDATA bits.

If TXENC64B66BUSE is asserted High, the TXBYPASS8B10B bit 0 signal bypasses the 64B/66B encoder on a clock basis, which means that two clock cycles are needed to do a full bypass of a block. The Sync Header is taken from the TXCHARDISPMODE[0:1]. To bypass on a block basis, the even boundary needs to be indicated at the fabric interface, which is contained in TXKERR bit 3. The TXCHARISK signal performs the function of TXC.

Table 2-10: **64B/66B Bypassing**

Signal	Function	
TXENC64B66BUSE	0 entire 64B/66B encoder bypassed	
	1 bypass on a clock-to-clock basis	
TXBYPASS8B10B[0]	**Function 64B/66B clock-to-clock bypass**	**Function 64B/66B entirely bypassed**
	0 indicates no bypass	defined by Table 2-12(1)
	1 indicates bypass this block	
TXCHARDISPMODE[0:1]	sync header shown in Figure 2-9 (same as SH[0:1])	
TXKERR[3]	indicates even boundary for bypassing on block basis	indicates which byte contains the sync header
TXCHARISK[3:0]	performs function of TXC	indicates character is a (K) control character

Notes:

1. TX sync header [0] = TXCHARDISPMODE[0]
   TX sync header [1] = TXCHARDISPVAL[0]

The transmit 64B/66B encoder borrows four bits of the TXCHARISK bus (bits [3:0]) to convey the control signaling to the 64B/66B encoder. The four TXC bits track with the TXDATA to signal data block formatting. The transmit fabric interface logic (which first monitors transmit data as it travels from the fabric interface to the PMA) drives the encoder with the four TXC bits as follows:

Table 2-11: **Transmit 64B/66B Encoder Control Mapping**

TXC[3:0] (TXCHARISK[3:0])	Block Formatting
1111	Idles OR terminate-with-idles
0001	Start-of-frame OR ordered-set
1110	Terminate in second position
1100	Terminate in third position

Table 2-11: **Transmit 64B/66B Encoder Control Mapping**

TXC[3:0] (TXCHARISK[3:0])	Block Formatting
1000	Terminate in fourth position
0000	Data OR error (no k-chars)

Each "one" in the TXC span represents a control-character-match -- recognition that the associated byte is a special control character of some type (idle, start, terminate, or ordered-set).

Normal Operation

The 64B/66B encoder implements the Encoding Block Format function shown in Figure 2-9.

Input Data	Sync	Block Payload							
Bit Position / Data Block Format	01	2 ... 65							
D_0 D_1 D_2 D_3 D_4 D_5 D_6 D_7	01	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
Control Block Formats		Block Type Field							
C_0 C_1 C_2 C_3 C_4 C_5 C_6 C_7	10	0x1e	C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7
C_0 C_1 C_2 C_3 O_4 D_5 D_6 D_7	10	0x2d	C_0	C_1	C_2	C_3	O_4	D_5	D_6	D_7
C_0 C_1 C_2 C_3 S_4 D_5 D_6 D_7	10	0x33	C_0	C_1	C_2	C_3		D_5	D_6	D_7
O_0 D_1 D_2 D_3 S_4 D_5 D_6 D_7	10	0x66	D_1	D_2	D_3	O_0		D_5	D_6	D_7
O_0 D_1 D_2 D_3 O_4 D_5 D_6 D_7	10	0x55	D_1	D_2	D_3	O_0	O_4	D_5	D_6	D_7
S_0 D_1 D_2 D_3 D_4 D_5 D_6 D_7	10	0x78	D_1	D_2	D_3	D_4		D_5	D_6	D_7
O_0 D_1 D_2 D_3 C_4 C_5 C_6 C_7	10	0x4b	D_1	D_2	D_3	O_0	C_4	C_5	C_6	C_7
T_0 C_1 C_2 C_3 C_4 C_5 C_6 C_7	10	0x87		C_1	C_2	C_3	C_4	C_5	C_6	C_7
D_0 T_1 C_2 C_3 C_4 C_5 C_6 C_7	10	0x99	D_0		C_2	C_3	C_4	C_5	C_6	C_7
D_0 D_1 T_2 C_3 C_4 C_5 C_6 C_7	10	0xaa	D_0	D_1		C_3	C_4	C_5	C_6	C_7
D_0 D_1 D_2 T_3 C_4 C_5 C_6 C_7	10	0xb4	D_0	D_1	D_2		C_4	C_5	C_6	C_7
D_0 D_1 D_2 D_3 T_4 C_5 C_6 C_7	10	0xcc	D_0	D_1	D_2	D_3		C_5	C_6	C_7
D_0 D_1 D_2 D_3 D_4 T_5 C_6 C_7	10	0xd2	D_0	D_1	D_2	D_3	D_4		C_6	C_7
D_0 D_1 D_2 D_3 D_4 D_5 T_6 C_7	10	0xe1	D_0	D_1	D_2	D_3	D_4	D_5		C_7
D_0 D_1 D_2 D_3 D_4 D_5 D_6 T_7	10	0xff	D_0	D_1	D_2	D_3	D_4	D_5	D_6	

UG035_ch3_23_091103

Figure 2-8: **Block Format Function**

The control codes are specified as follows in Table 2-12:

Table 2-12: **Control Codes**

Control Character	Notation	XGMII Control Code	10GBASE-R Control Code	10GBASE-R 0 Code	8B/10B Code
idle	/I/	0x07	0x00		K28.0 or K28.3 or K28.5
start	/S/	0xfb	encoded by block type field		K27.7
terminate	/T/	0xfd	encoded by block type field		K29.7
error	/E/	0xfe	0x1e		K30.7
Sequence ordered_set	/Q/	0x9c	encoded by block type field plus O mode	0x0	K28.4
reserved0	/R/	0x1c	0x2d		K28.0
reserved1		0x3c	0x33		K28.1
reserved2	/N/	0x7c	0x4b		K28.3
reserved3	/K/	0xbc	0x55		K28.5
reserved4		0xdc	0x66		K28.6
reserved5		0xf7	0x78		K23.7
Signal ordered_set	/Fsig/	0x5c	encoded by block type field plus O mode	0xF	K28.2

Scrambler

Bypassing

If the signal TXSCRAM64B66BUSE is deasserted Low, the scrambler is not used. Note that the scrambler operates on the read side of the transmit FIFO.

Normal Operation

If the signal TXSCRAM64B66BUSE is asserted High, the scrambler is enabled for use. The scrambler uses the polynomial:

$$G(x) = 1 + x^{39} + x^{58}$$

to scramble 64B/66B payload data. The scrambler works in conjunction with the gearbox to scramble and format data correctly.

TXSCRAM64B66BUSE	0 scrambler not used
	1 scrambler enabled

Note: When using the 64B/66B scrambler, the Gearbox must also be enabled (Always set to TXSCRAM64BB66USE = TXGEARBOX64B66BUSE)

Gearbox

Bypassing

If the signal TXGEARBOX64B66BUSE is deasserted Low, the gearbox is not used. The gearbox should always be enabled when using the 64/66 protocol.

Normal Operation

If the signal TXGEARBOX64B66BUSE is asserted High, the gear box is enabled. The gearbox frames 64B/66B data for the PMA.

TXGEARBOX64B66BUSE	0
	1 always set to '1' when scrambler and descrambler are enabled.

Decoder

Bypassing

If RXDEC64B66BUSE is deasserted Low, the entire 64B/66B decoder is not used.

Normal Operation

If RXDEC64B66BUSE is asserted High, the 64B/66B decoder decodes according to the 64B/66B block format table shown in Figure 2-6.

If the signal RXIGNOREBTF is asserted High, block type fields not recognized are passed on, whereas if the signal is asserted Low, the error block /E/ is passed on. RXCHARISK is equivalent to RXC when the decoder is enabled.

RXDEC64B66BUSE	0 decoder not used	
	1 decoder used	
RXIGNOREBTF	**Function 64B/66B decoder used**	**Function 64B/66B decoder bypassed**
	0 unrecognized field types cause /E/ passed on	undefined
	1 unrecognized field types passed on	
RXCHARISK	equivalent to RXC	defined by 8B/10B decoder use

Descrambler

Bypassing

If the signal RXDESCRAM64B66BUSE is deasserted Low, the descrambler is not used.

Normal Operation

If the signal RXDESCRAM64B66BUSE is asserted High, the descrambler is enabled for use. The descrambler uses the polynomial:

$$G(x) = 1 + x^{39} + x^{58}$$

Block Sync

Normal Operation

This block sync design works hand-in-hand with the commaDet block. The commaDet takes as input 32 bits of scrambled and unaligned data from the PMA. It then sends to the block sync the 2-bit sync header, or what it thinks is the sync header based on the current tag value. It asserts `test_sh` which tells the block sync to test the value of the sync header. The block sync analyzes the sync header and if it is valid, increments the `sh_cnt` counter. If the sync header is not a legal value, `sh_cnt` is incremented as well as the counter `sh_invalid_cnt`, and then `bit_slip` is asserted for one clock. The bit slip signal feeds back to the commaDet block and tells it to shift the barrel shifter by one bit. This process of slipping and testing the sync header repeats until block lock is achieved.

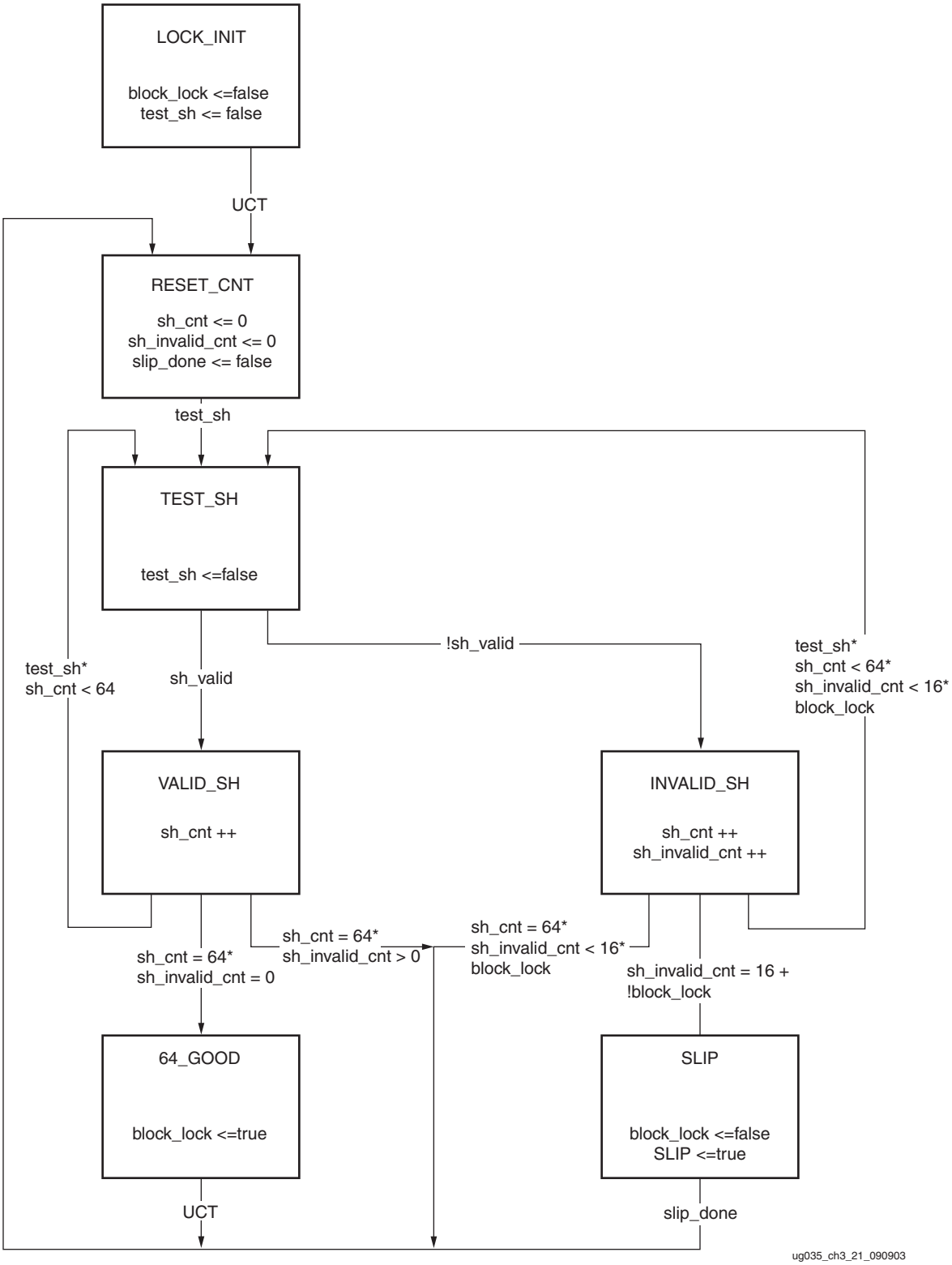


Figure 2-9: **Block Sync State Machine**

The state machine works by keeping track of valid and invalid sync 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. When test_sh is asserted, the next state is TEST_SH, which checks the validity of the sync header. If it is valid, the next state is VALID_SH, if not, the state changes to INVALID_SH.

From VALID_SH, if sh_cnt is less than the attribute 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 zeroed.

If at TEST_SH sh_cnt equals sh_cnt_max, but sh_invalid_cnt is greater than zero, then the next state is RESET_CNT. From INVALID_SH, if sh_invalid_cnt equals sh_invalid_cnt_max, or if block_lock is not asserted, the next state is SLIP, where bit_slip is asserted, and then on to RESET_CNT. If sh_cnt equals sh_cnt_max and sh_invalid_cnt is less than sh_invalid_cnt_max and block_lock is asserted, then go back to RESET_CNT without changing block_lock or bit_slip.

Finally, if test_sh is High and sh_cnt is less than sh_cnt_max, and sh_invalid_cnt is less than sh_invalid_cnt_max and block_lock is asserted, go back to the TEST_SH state. **The main thing to note with this state machine is that to achieve block lock, one must receive sh_cnt_max number of valid sync headers in a row without getting an invalid sync header.** However, once block lock is achieved, sh_invalid_cnt_max -1 number of invalid sync headers can be received within sh_cnt_max number of valid sync headers. Thus, once locked, it is harder to break lock.

Functions Common to All Protocols

Clock Correction

Clock correction is needed when the rate that data is fed into the write side of the receive FIFO is either slower or faster than the rate that data is retrieved from the read side of the receive FIFO. The rate of write data entering the FIFO is determined by the frequency of RXRECCLK. The rate of read data retrieved from the read side of the FIFO is determined by the frequency of RXUSRCLK.

There is one clock correction mode: Append/Remove Idle Clock Correction.

Append/Remove Idle Clock Correction

When the attribute CLK_COR_SEQ_DROP is asserted Low and CLK_CORRECT_USE is asserted hIgh, the Append/remove Idle Clock Correction mode is enabled.

The Append/remove Idle Clock Correction mode corrects for differing clock rates by finding idles in the bitstream, and then either appending or removing idles at the point where the idles were found.

There are a few attributes that need to be set by the user so that the append/remove function can be used correctly. The attribute CLK_COR_MAX_LAT sets the maximum latency through the receive FIFO. If the latency through the receive FIFO exceeds this value, idles are removed so that latency through the receive FIFO is less than CLK_COR_MAX_LAT.

The attribute CLK_COR_MIN_LAT sets the minimum latency through the receive FIFO. If the latency through the receive FIFO is less than this value, idles are inserted so that the latency through the receive FIFO are greater than CLK_COR_MIN_LAT. A correction to the latency due to a CLK_COR_MAX_LAT violation is never less than

CLK_COR_MIN_LAT. This is also true for a correction to the latency due to a CLK_COR_MIN_LAT violation; the resulting latency after the correction is greater than CLK_COR_MAX_LAT.

Clock Correction Sequences

Searching within the bitstream for an idle is the core function of the clock correction circuit. The detection of idles starts the correction procedure.

Idles the clock correction circuit should detect are specified by the lower 10 bits of the attributes:

- CLK_COR_SEQ_1_1
- CLK_COR_SEQ_1_2
- CLK_COR_SEQ_1_3
- CLK_COR_SEQ_1_4
- CLK_COR_SEQ_2_1
- CLK_COR_SEQ_2_2
- CLK_COR_SEQ_2_3
- CLK_COR_SEQ_2_4

The 11th bit of each clock correction sequence attribute determines either an 8- or 10-bit compare.

Detection of the clock correction sequence in the bitstream is specified by eight words consisting of 10 bits each. Clock correction sequences can have lengths of 1, 2, 3, 4 or 8 bytes.

When the length specified by the user is between 1 and 4, CLK_COR_SEQ_1_* holds the first pattern to be searched for. CLK_COR_SEQ_1_1 is the least significant byte, which is transmitted first from the transmitter and detected first in the receiver. If CLK_COR_SEQ_2_USE is asserted High when the length is between 1 and 4, the sequence specified by CLK_COR_SEQ_2_* is specified as a second pattern to match. In that case, the pattern specified by sequence 1 *or* sequence 2 matches as a clock correction sequence.

Note: The CLK_COR_SEQ_MASK must have the bits set to a logic 1 mask off the 2 or 3 unused bytes.

When the length specified by the user is eight, CLK_COR_SEQ_1_* holds the first four bytes, while CLK_COR_SEQ_2_* holds the last four bytes. CLK_COR_SEQ_1_1 is the least significant byte, which is transmitted first from the transmitter and detected first in the receiver. CLK_COR__SEQ_2_USE must be asserted High.

The clock correction sequence is a special sequence to accommodate frequency differences between the received data (as reflected in RXRECCLK) and RXUSRCLK. Most of the primitives have these defaulted to the respective protocols. Only the GT_CUSTOM allows this sequence to be set to any specific protocol. The sequence contains 11 bits including the 10 bits of serial data. The 11th bit has two different formats. The typical usage is:

- 0, disparity error required, char is K, 8-bit data value (after 8B/10B decoding, depends on CLK_COR_8B10B_DE)
- 0, 10-bit data value (without 8B/10B decoding, depends on CLK_COR_8B10B_DE)
- 1, xx, sync character (with 64B/66B encoding
- 1, xx, 8-bit data value

Table 2-13 is an example of data 11-bit attribute setting, the character value, CHARISK value, and the parallel data interface, and how each corresponds with the other.

Table 2-13: **Clock Correction Sequence/Data Correlation for 16-Bit Data Port**

Attribute Setting	Character	CHARISK	TXDATA (hex)
CLK_COR_SEQ_1_1 = 00110111100	K28.5	*1*	*BC*
CLK_COR_SEQ_1_2 = 00010010101	D21.4	*0*	*95*
CLK_COR_SEQ_1_3 = 00010110101	D21.5	*0*	*B5*
CLK_COR_SEQ_1_4 = 00010110101	D21.5	*0*	*B5*

Notes:

1. CLK_COR_8B10B_DE = TRUE.

Determining Correct CLK_COR_MIN_LAT

To determine the correct CLK_COR_MIN_LAT value, several requirements must be met.

- CLK_COR_MIN_LAT must be less than or equal to 12.
- CLK_COR_MIN_LAT and CLK_COR_MAX_LAT must be multiples of CCS/CBS lengths and ALIGN_COMMA_WORD.
- For symbols less than 8 bytes, (CLK_COR_MIN_LAT – CHAN_BOND_LIMIT) > 12. For symbols of 8 bytes, (CLK_COR_MIN_LAT – CHAN_BOND_LIMIT) > 16.

Channel Bonding

Channel bonding is the technique of tying several serial channels together to create one aggregate channel. Several channels are fed on the transmit side by one parallel bus and reproduced on the receive side as the identical parallel bus. The maximum number of serial differential pairs that can be bonded is 20. Channel bonding is supported by several primitives including GT10_CUSTOM, GT10_INFINIBAND, GT10_XAUI, and GT10_AURORA.

The channel bonding match logic finds CB characters across word boundaries and performs a “comma” style realignment of the data. The data path is byte scrambled until reset as shown below in the example (additional comma alignments will not realign the data). As a result, users should be careful when picking channel bonding characters and should use, in general, special characters that cannot appear in the normal data stream.

Example:

The channel bond character is 0x000000FF. If this sequence of data is sent:

```
000000FF
01020304
05060708
09000000
FF010203
04050607
```

The result is:

```
000000FF
01020304
05060708
000000FF
01020304
```

050607xx

The bonded channels consist of one master transceiver and 1 to 19 slave transceivers. The CHBONDI/CHBONDO buses of the transceivers are daisy-chained together as shown in Figure 2-10.

When the master transceiver detects a channel bond alignment sequence in its data stream, it signals the slave to perform channel bonding by driving its CHBONDO bus as follows in Table 2-14:

Table 2-14: **Channel Bond Alignment Sequence**

Detected	CHBONDO Bus
No Channel Bond	$XX000_2$
Channel Bond - Byte 0	$XX100_2$
Channel Bond - Byte 1	$XX101_2$
Channel Bond - Byte 2	$XX110_2$
Channel Bond - Byte 3	$XX111_2$

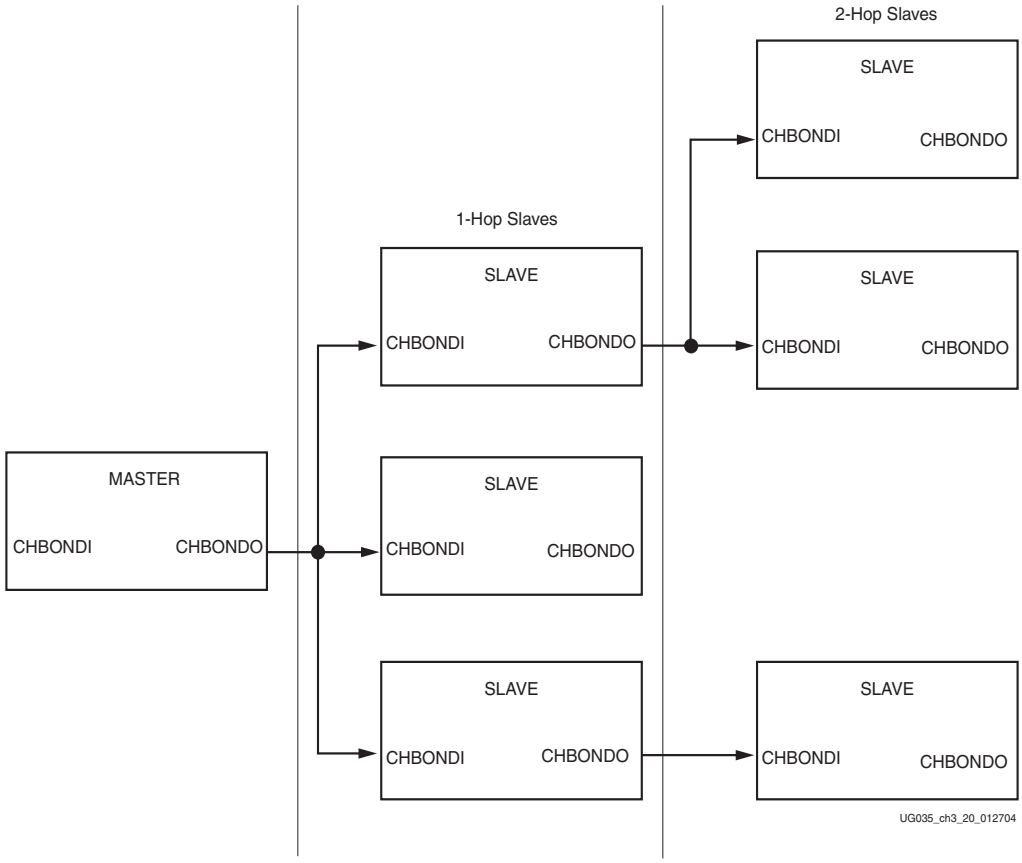


Figure 2-10: **Daisy-Chained Transceiver CHBONDI/CHBONDO Buses**

Whether a slave is a 1-hop or 2-hop slave, internal logic causes the data driven on the CHBONDO bus from the master to be recognized by the slaves at the same time and must

be deterministic. Therefore, it is important that the interconnect of CHBONDO-to-CHBONDI not contain any pipeline stages. The data must transfer from CHBONDO to CHBONDI in one clock.

The data streams input to the channel bonded transceivers can be skewed in time from each other. The maximum byte skew that the channel bond logic should allow is set by the attribute MC_CHAN_BOND_LIMIT. During the channel bond operation, the slave receives notification of the master's alignment code location via the CHBONDO bus. If a slave detects the position of its alignment code to be outside the window of CHAN_BOND_LIMIT from the master, then the slave does not perform the channel bond and sets a channel bond error flag. If the channel bond is successful, the slave outputs its skew relative to the master. The skew and channel bond error flag are available on the RXBUFSTATUS bus.

For place and route, the transceiver has one restriction. This is required when channel bonding is implemented. Because of the delay limitations on the CHBONDO to CHBONDI ports, linking of the Master to a Slave_1_hop must run either in the X or Y direction, but not both.

In Figure 2-11, the two Slave_1_hops are linked to the master in only one direction. To navigate to the other slave (a Slave_2_hops), both X and Y displacement is needed. This slave needs one level of daisy-chaining, which is the basis of the Slave_2_hops setting.

Figure 2-11 and Figure 2-12 show the channel bonding mode and linking for an XC2VPX20 and XC2VPX70 devices, which (optionally) contain more transceivers (20) per chip. To ensure the timing is met on the link between the CHBONDO and CHBONDI ports, a constraint must be added to check the time delay.

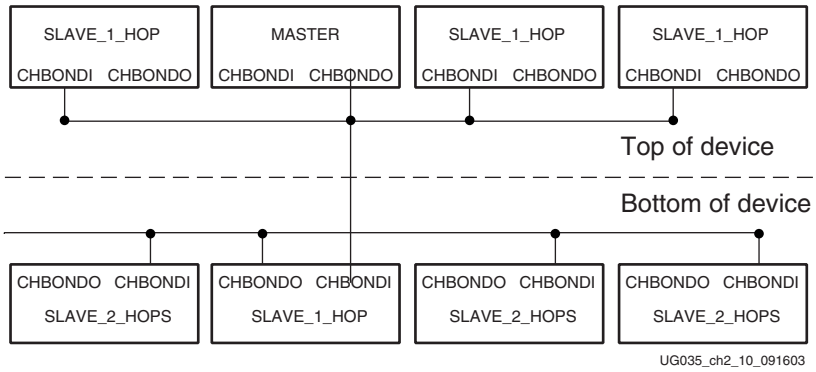


Figure 2-11: **XC2VPX20 Device Implementation**

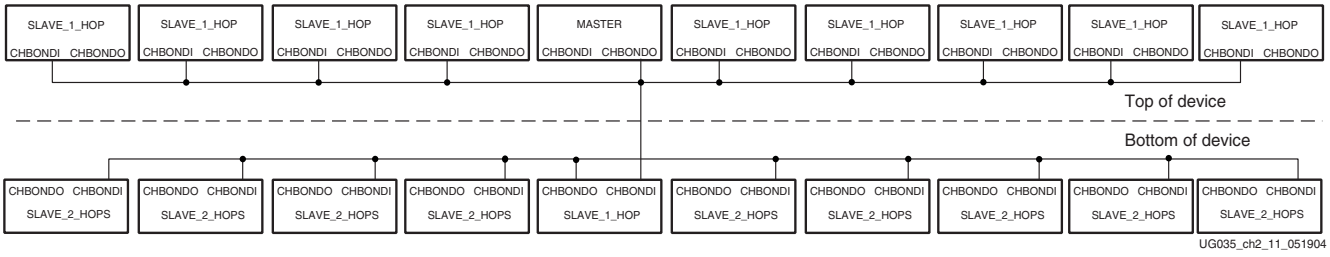


Figure 2-12: **XC2VPX70 Device Implementation**

Status and Event Bus

The Virtex-II Pro X design has merged several signals together to provide extra functionality over the Virtex-II Pro™ design. The signals CHBONDDONE, RXBUFSTATUS, and RXCLKCORCNT were previously used independently of each other to indicate status. In the Virtex-II Pro X design, these signals are concatenated together to provide a status and event bus.

There are two modes of this concatenated bus, status mode and event mode. In status mode, the bus indicates either the difference between the read and write pointers of the receive side FIFO or the skew of the last channel bond event.

Status Indication

In status mode, the RXBUFSTATUS and RXCLKCORCNT pins alternate between the buffer pointer difference and channel bonding skew. The protocol is described by three sequential clocks (STATUS and DATA are one clock in duration) when operating with a 32-bit or 40-bit internal data-width, or six sequential clocks (STATUS and DATA are two clocks in duration) when operating with a 16-bit or 20-bit internal data width:

```
<STATUS INDICATOR> <DATA0><DATA1>
```

where

STATUS INDICATOR can indicate either pointer difference or channel bond skew, DATA0 indicates status data 5:3, and DATA1 indicates status data 2:0.

Table 2-15 shows the signal values for a pointer difference status where the variable pointerDiff[5:0] holds the pointer difference between the receive write and read pointers. If the pointerDiff[5:0] is < 6'b000110, then RXFIFO is almost under flown. If the pointerDiff[5:0] is > 6'b111001, then the RXFIFO is almost over flown.

Table 2-15: **Signal Values for a Pointer Difference Status**

Status	CHBONDDONE	RXBUFSTATUS	RXCLKCORCNT
STATUS INDICATOR	1'b0	2'b01	3'b000
DATA0	1'b0	2'b00	pointerDiff[5:3]
DATA1	1'b0	2'b00	pointerDiff[2:0]

Table 2-16 shows the signal values for a channel bonding skew where the variable cbSkew[5:0] holds the pointer difference between the receive write and read pointers:

Table 2-16: **Signal Values for a Channel Bonding Skew**

Status	CHBONDDONE	RXBUFSTATUS	RXCLKCORCNT
STATUS INDICATOR	1'b0	2'b01	3'b001
DATA0	1'b0	2'b00	cbSkew[5:3]
DATA1	1'b0	2'b00	cbSkew[2:0]

Event Indication

Two types of events can occur. See Table 2-17. When an event occurs, it can override a status indication. An event can only last for one clock and can be signaled by CHBONDDONE asserting High, or RXBUFSTATUS equating to 2'b10.

Table 2-17: **Signal Values for Event Indication**

Event	CHBONDDONE	RXBUFSTATUS	RXCLKCORCNT
Channel Bond Load	1'b1	2'b00	3'b111
Clock Correction	1'b0	2'b10	3'bxxx

Note: An event will always override status, but after an event is completed, status will continue to alternate between the pointer difference and the channel bond skew.

Sample Verilog

The following sample code is to determine underflow or overflow of the RX buffer when 32-bit or 40-bit internal data path is selected.

```
module   status_decoder (
     RXUSRCLK2,
     DCM_LOCKED_N,
     PMARXLOCK,
     CHBONDDONE,
     RXBUFSTATUS,
     RXCLKCORCNT,

     cc_event_insert,      // Clock Correction Insertion Event
     cc_event_remove,      // Clock Correction Removal Event
     cb_event_load,        // Channel Bonding Load Event
     err_event_cc,         // Clock Correction Error Event
     err_event_cb,         // Channel Bonding Error Event

     pointerDiff,          // RX Elastic Buffer Pointer Difference
     rxbuf_almost_err,     // RX Elastic Buffer Almost Error

     cbSkew);

     input        RXUSRCLK2;
     input        DCM_LOCKED_N;
     input        PMARXLOCK;
     input        CHBONDDONE;
     input  [1:0] RXBUFSTATUS;
     input  [2:0] RXCLKCORCNT;
     output       cc_event_insert;
     output       cc_event_remove;
     output       cb_event_load;
     output       err_event_cc;
     output       err_event_cb;
     output [5:0] pointerDiff;
     output       rxbuf_almost_err;
     output [5:0] cbSkew;

////////////////////////////////////////////////////////////////////////
//Signal declaration
////////////////////////////////////////////////////////////////////////
reg         cc_event_insert;
reg         cc_event_remove;
reg         cb_event_load;
reg         err_event_cc;
reg         err_event_cb;
reg [5:0]   pointerDiff;
```

```
reg [2:0]   pointerDiff_hi;
reg [1:0]   pointerDiff_valid;
reg [5:0]   cbSkew;
reg [2:0]   cbSkew_hi;
reg [1:0]   cbSkew_valid;
reg         rxbuf_almost_err;

wire [5:0] status_event_bus;
wire [2:0] status_bus;

parameter CC_EVENT_INSERT_C = 6'b010001;
parameter CC_EVENT_REMOVE_C = 6'b010000;
parameter CB_EVENT_LOAD_C   = 6'b100111;
parameter ERR_EVENT_CC_C    = 6'b011000;
parameter ERR_EVENT_CB_C    = 6'b011001;

parameter STATUS_INDICATOR_C= 3'b001;
parameter STATUS_DATA_C     = 3'b000;

assign status_event_bus = {CHBONDDONE, RXBUFSTATUS[1], RXBUFSTATUS[0],
RXCLKCORCNT[2],
RXCLKCORCNT[1], RXCLKCORCNT[0]};
assign status_bus      = {CHBONDDONE, RXBUFSTATUS[1], RXBUFSTATUS[0]};


//////////////////////////////////////////////////////////////////////
//Logic to decode events
//////////////////////////////////////////////////////////////////////
always @(posedge RXUSRCLK2 or posedge DCM_LOCKED_N)
  begin
    if (DCM_LOCKED_N) begin
      cc_event_insert <= 1'b0;
      cc_event_remove <= 1'b0;
      cb_event_load   <= 1'b0;
      err_event_cc    <= 1'b0;
      err_event_cb    <= 1'b0;
     end
    else begin
      cc_event_insert <= status_event_bus == CC_EVENT_INSERT_C;
      cc_event_remove <= status_event_bus == CC_EVENT_REMOVE_C;
      cb_event_load   <= status_event_bus == CB_EVENT_LOAD_C;
      err_event_cc    <= status_event_bus == ERR_EVENT_CC_C;
      err_event_cb    <= status_event_bus == ERR_EVENT_CB_C;

    end
 end


//////////////////////////////////////////////////////////////////////
// Logic to decode the cbSkew value and pointerDiff value
//////////////////////////////////////////////////////////////////////
always @(posedge RXUSRCLK2 or posedge DCM_LOCKED_N)
  begin
    if (DCM_LOCKED_N) begin
       pointerDiff_valid <=  2'b00;
       cbSkew_valid      <=  2'b00;
      end
    else if ((status_bus == STATUS_INDICATOR_C) & ~RXCLKCORCNT[2] &
~RXCLKCORCNT[1] ) begin
```

```
       pointerDiff_valid <=  {1'b0, ~RXCLKCORCNT[0]};
       cbSkew_valid      <=  {1'b0,  RXCLKCORCNT[0]};
      end
    else if (status_bus == STATUS_DATA_C) begin
       pointerDiff_valid[1] <=  pointerDiff_valid[0];
       pointerDiff_valid[0] <=  1'b0;
       cbSkew_valid[1]      <=  cbSkew_valid[0];
       cbSkew_valid[0]      <=  1'b0;
     end
    else begin // clear the valid signal if the status is interrupted
by an event.
       pointerDiff_valid <=  2'b00;
       cbSkew_valid      <=  2'b00;
     end
  end


always @(posedge RXUSRCLK2 or posedge DCM_LOCKED_N)
  begin
   if (DCM_LOCKED_N || ~PMARXLOCK) begin // reset the value to neutral
position
       pointerDiff <=  32;
       pointerDiff_hi <= 4;
       cbSkew      <=  32;
       cbSkew_hi   <=  4;
      end
   else if (status_bus == STATUS_DATA_C) begin

       if (pointerDiff_valid[0])  // register higher 3 bits
         pointerDiff_hi   <=  RXCLKCORCNT;
      else if (pointerDiff_valid[1]) // update entire register when all
6 bits are acquired.
         pointerDiff      <=  {pointerDiff_hi , RXCLKCORCNT};

       if (cbSkew_valid[0])      // register higher 3 bits
         cbSkew_hi        <=  RXCLKCORCNT;
       else if (cbSkew_valid[1]) // update entire register when all 6
bits are acquired.
         cbSkew           <=  {cbSkew_hi , RXCLKCORCNT};
   end
 end

//////////////////////////////////////////////////////////////////////
// Generate RX Elastic Buffer almost error
//////////////////////////////////////////////////////////////////////
always @(posedge RXUSRCLK2 or posedge DCM_LOCKED_N)
 begin
    if (DCM_LOCKED_N)
      rxbuf_almost_err <= 1'b0;
    else
      rxbuf_almost_err <= (pointerDiff < 6) | (pointerDiff > 57);
end

endmodule
```

Chapter 3

Clocking and Clock Domains

Clock Domain Architecture

There are seven clock inputs into each RocketIO X transceiver instantiation. REFCLK, REFCLK2, and BREFCLK are clocks generated from an external source. BREFCLK is a set of differential inputs into the FPGA that can create a clock tree for all MGTs on one side of the device. See Figure 3-1. The reference clocks connect to the REFCLK, REFCLK2, or BREFCLK of the RocketIO X Multi-Gigabit Transceiver (MGT). While only one of these reference clocks is needed to drive the MGT, **BREFCLK inputs for the reference clock are recommended for the best operation. All characterization and data sheet numbers use the BREFCLK. Therefore, REFCLK usage results in performance degradation from the published performance numbers.** BREFCLK also clocks a Digital Clock Manager (DCM) to generate all of the other clocks for the MGT.

Note: Do not run a reference clock through a DCM; jitter control is optimized on reference clock nets without the use of a DCM.

Note: BREFCLK inputs can only be used to drive the MGTs and DCMs.

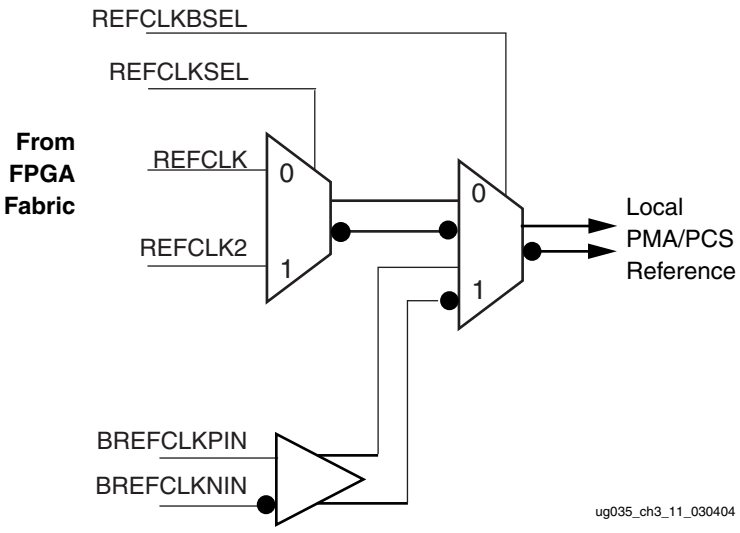


Figure 3-1: **Reference Clock Selection**

Clock Ports

Table 3-1: **Clock Ports**

Clock	I/Os	Description
BREFCLKNIN BREFCLKPIN	Input	Reference clock used for generating high-frequency timing in the TX and RX PLLs. The multiplication ratio for parallel-to-serial conversion is mode/protocol dependent.
REFCLK	Input	Reference clock used for generating high-frequency timing in the TX and RX PLLs. The multiplication ratio for parallel-to-serial conversion is mode/protocol dependent. This clock is a higher jitter clock. If used, performance based on data sheet numbers will not be met.
REFCLK2	Input	Reference clock used for generating high-frequency timing in the TX and RX PLLs. The multiplication ratio for parallel-to-serial conversion is mode/protocol dependent. This clock is a higher jitter clock. If used, performance based on data sheet numbers will not be met.
RXUSRCLK	Input	Clock from FPGA used for reading the RX Elastic Buffer. Clock signals CHBONDI and CHBONDO into and out of the transceiver. This clock is typically the same as TXUSRCLK, but if RXRECCLK is used to source a DCM, this clock can be frequency locked to the recovered clock.
RXUSRCLK2	Input	Clock from FPGA used to clock RX data and status between the transceiver and FPGA fabric. The relationship between RXUSRCLK2 and RXUSRCLK depends on the width of the receiver data path. RXUSRCLK2 is typically the same as TXUSRCLK2, but if RXRECCLK is used to source a DCM, this clock can be frequency locked to the recovered clock.
TXUSRCLK	Input	Clock from FPGA used for writing the TX Buffer. This clock must be frequency locked to TXOUTCLK for proper operation.
TXUSRCLK2	Input	Clock from FPGA used to clock TX data and status between the transceiver and FPGA fabric. The relationship between TXUSRCLK2 and TXUSRCLK depends on the width of the transmission data path.
RXRECCLK	Output	Recovered clock from serial data stream. This clock is scaled based upon the specific mode/protocol.
TXOUTCLK	Output	Scaled transmit clock generated within TX clock management unit. This clock is scaled based upon the specific mode/protocol.
REFCLKBSEL	Input	Selects between REFCLK/REFCLK2 and BREFCLK. 0 selects REFCLK/REFCLK2 (based on REFCLKSEL); 1 selects BREFCLK.
REFCLKSEL	Input	Selects which reference clock is used (when REFCLKBSEL=0). 0 selects REFCLK; 1 selects REFCLK2.

Use Models

Virtex-II Pro X MGTs offer considerable flexibility of clocking schemes. Please see Table 3-2 and Table 3-3 for a complete summary of supported PMA modes, use models, and clocking speeds. The PMA modes are set by the PMA_SPEED attribute.

Note: The examples shown in Figure 3-2 through Figure 3-12 are valid for BREFCLK configurations. These examples can be used for REFCLK configurations, but published performance cannot be met.

The use models discussed below the clock ratio terminology discussed in table 2-4 in which the USRCLK:USRCLK2 is in terms of frequency. All the use models have USRCLK or USRCLK2 as the base frequency (1); other frequencies can be 2, 4, or 0 (half the base frequency). The models use an X:Y:Z format: X = reference clock, Y = USRCLK, and Z = USRCLK2. Table 3-2, page 76 shows which use model can be used for the base serial rate of that mode.

1:1 Use models

The use models in this section represent when the external and internal data widths are the same.

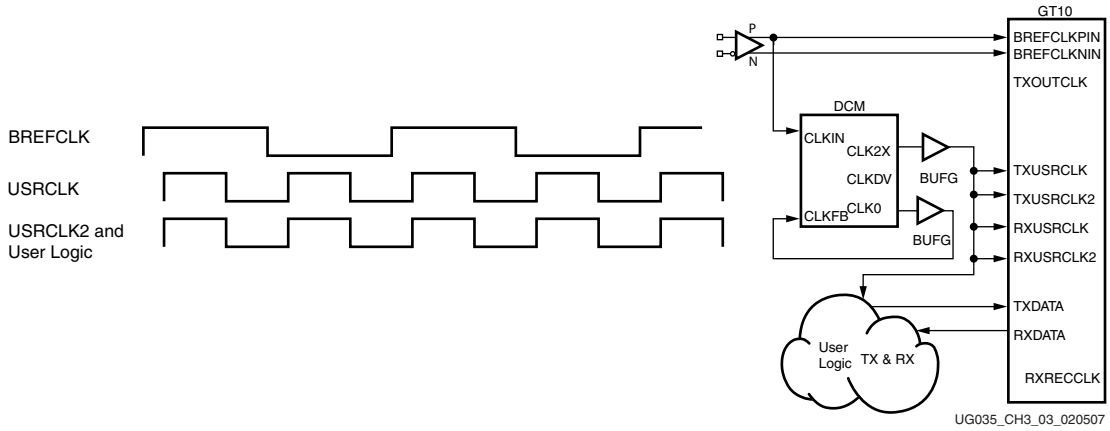


Figure 3-2: **BREFCLK 0:1:1**

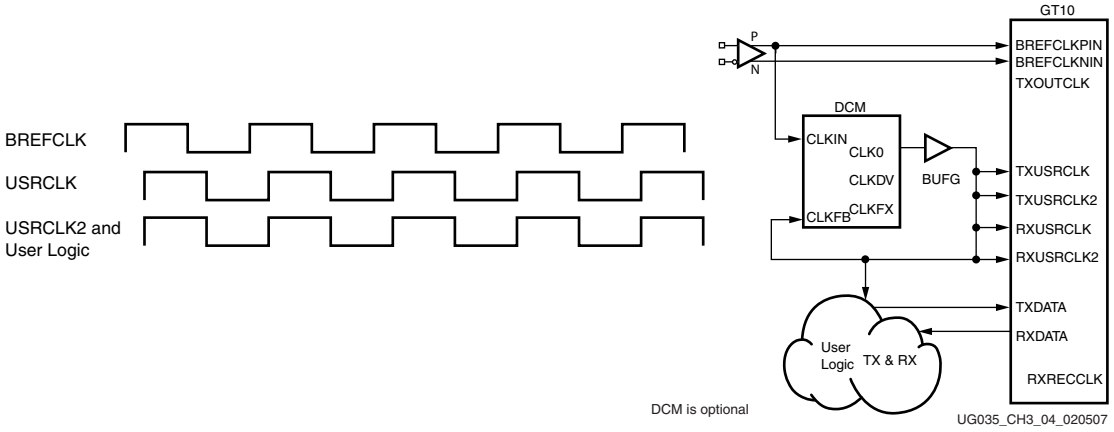


Figure 3-3: **BREFCLK 1:1:1**

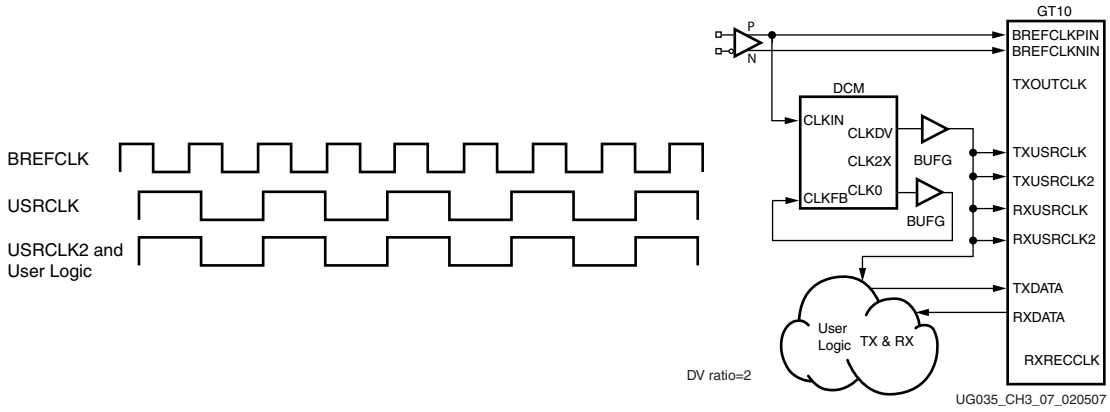


Figure 3-4: **BREFCLK 2:1:1**

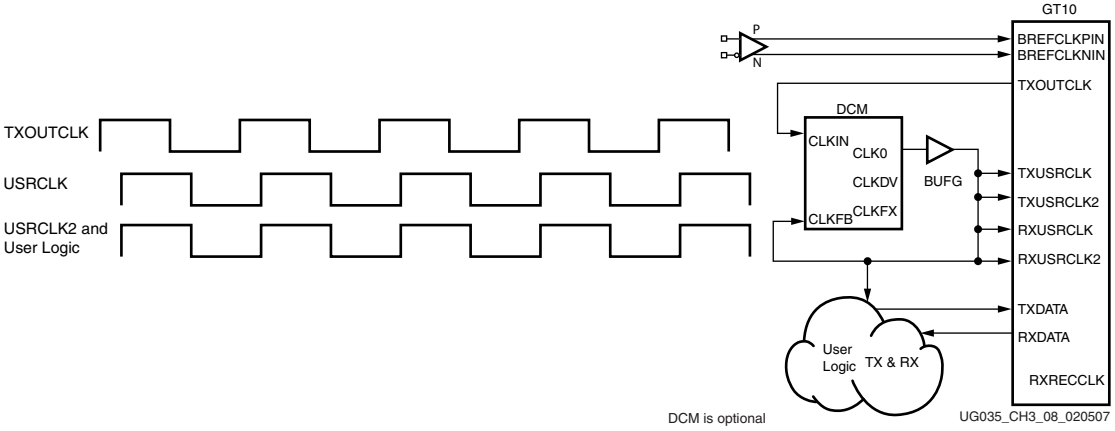


Figure 3-5: **TXOUTCLK 1:1:1**

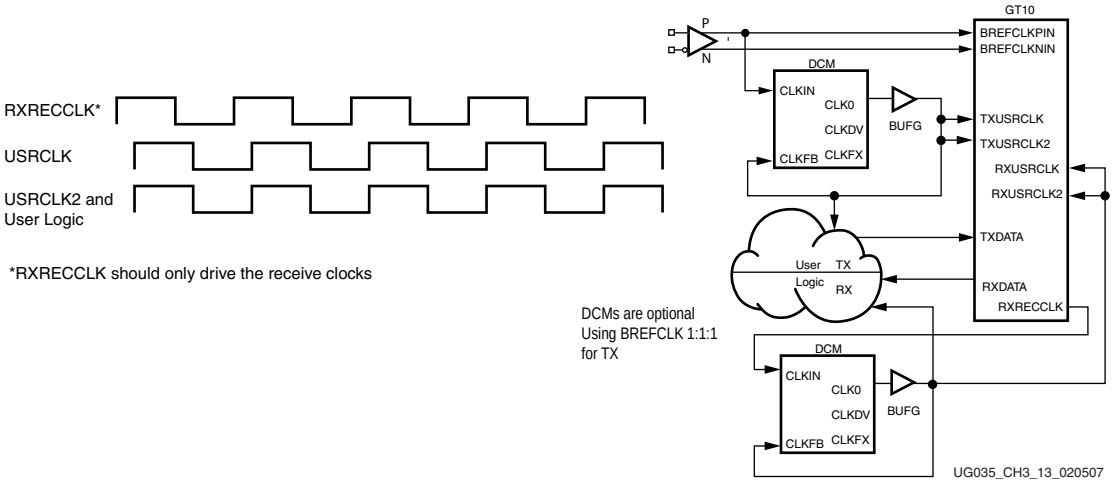


Figure 3-6: **RXRECCLK 1:1:1**

2:1 Use Models

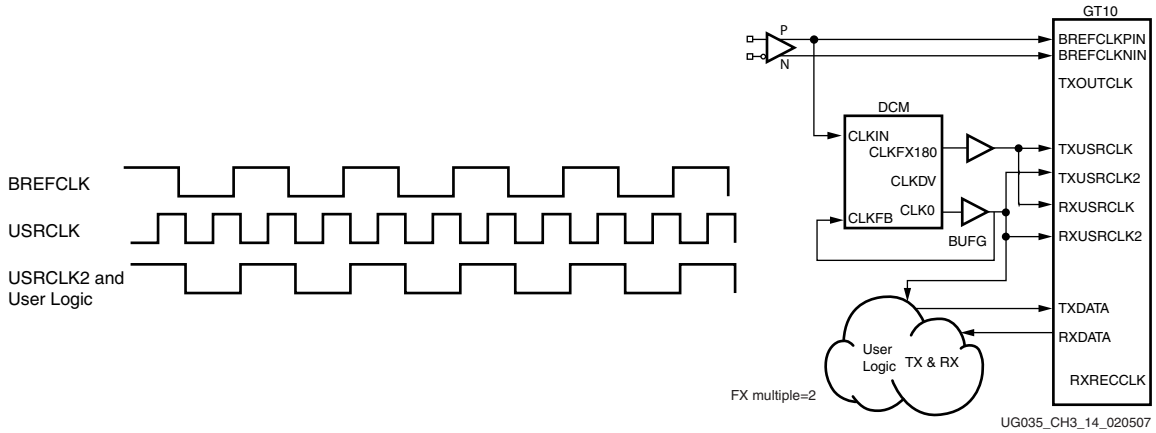


Figure 3-7: **BREFCLK 1:2:1**

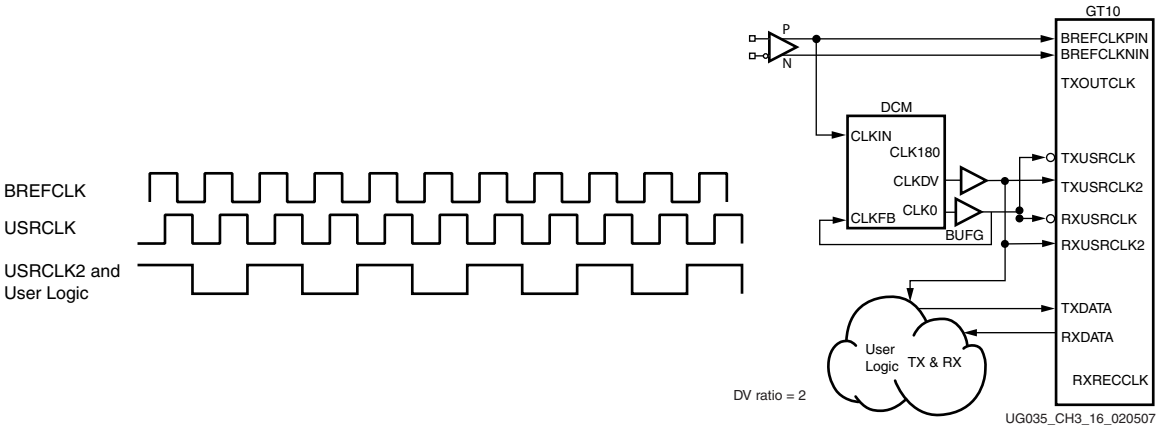


Figure 3-8: **BREFCLK 2:2:1**

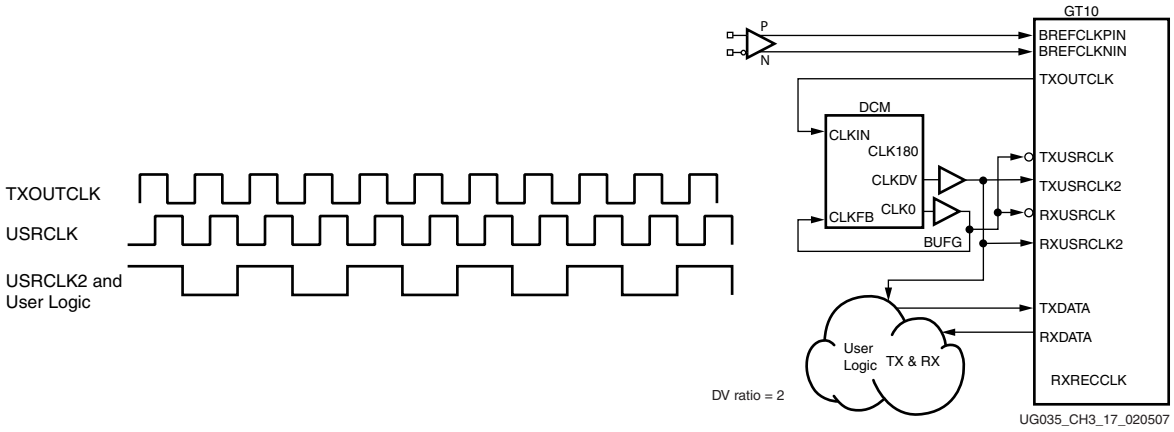


Figure 3-9: **TXOUTCLK 2:2:1**

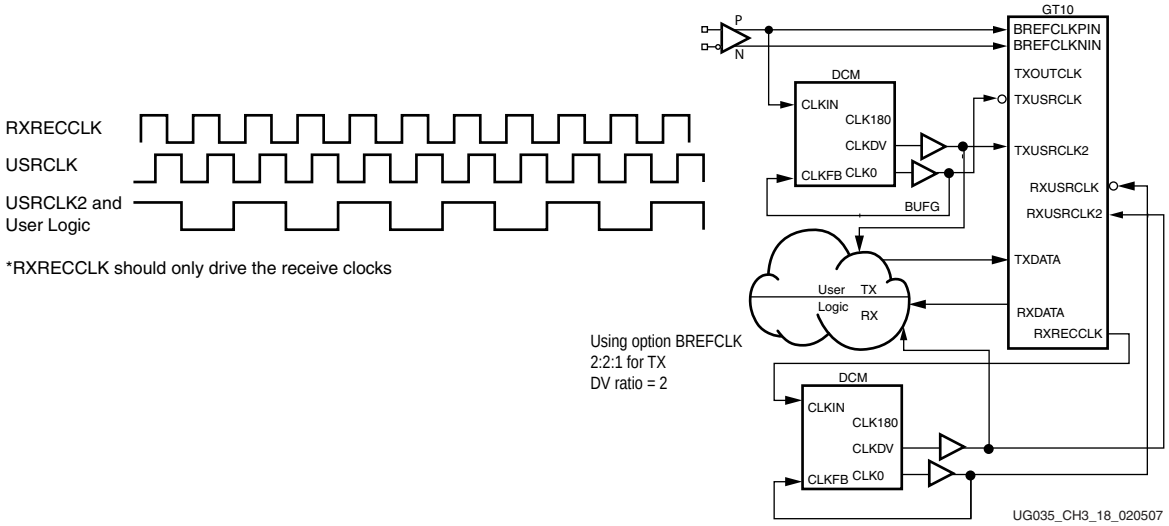


Figure 3-10: **RXRECCLK 2:2:1**

1:2 Use Models

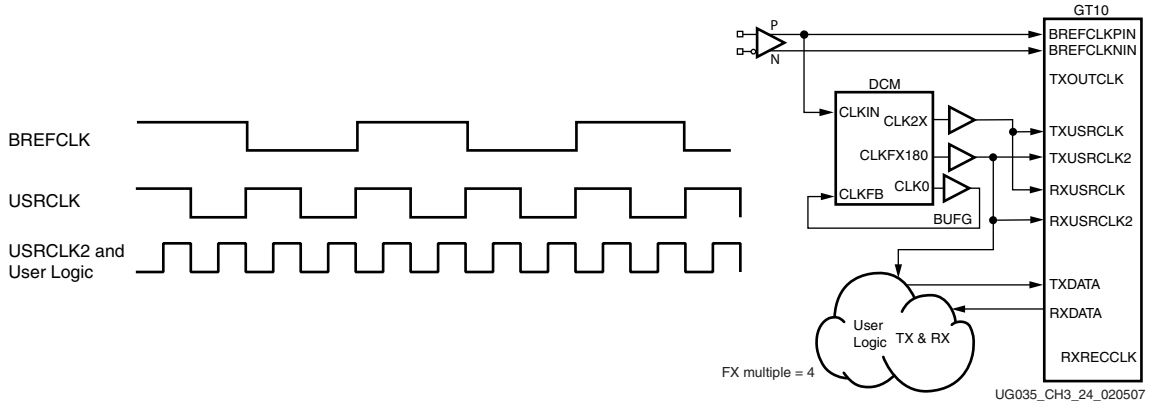


Figure 3-11: **BREFCLK 0:1:2**

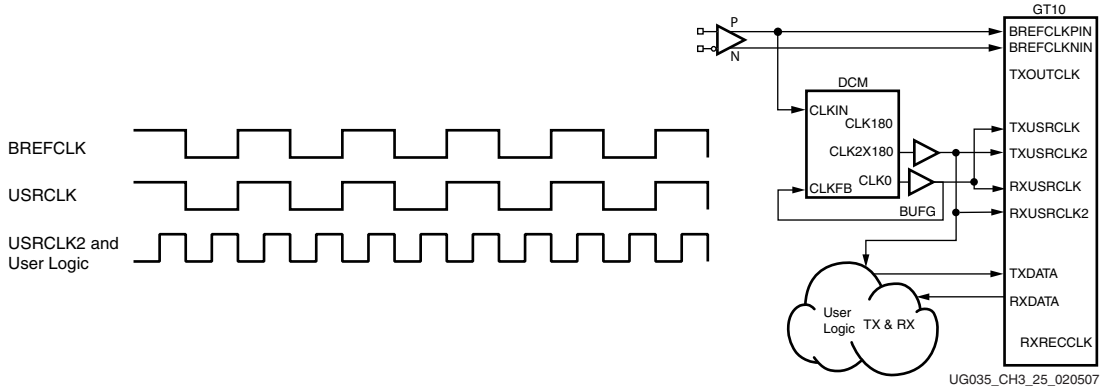


Figure 3-12: **BREFCLK 1:1:2**

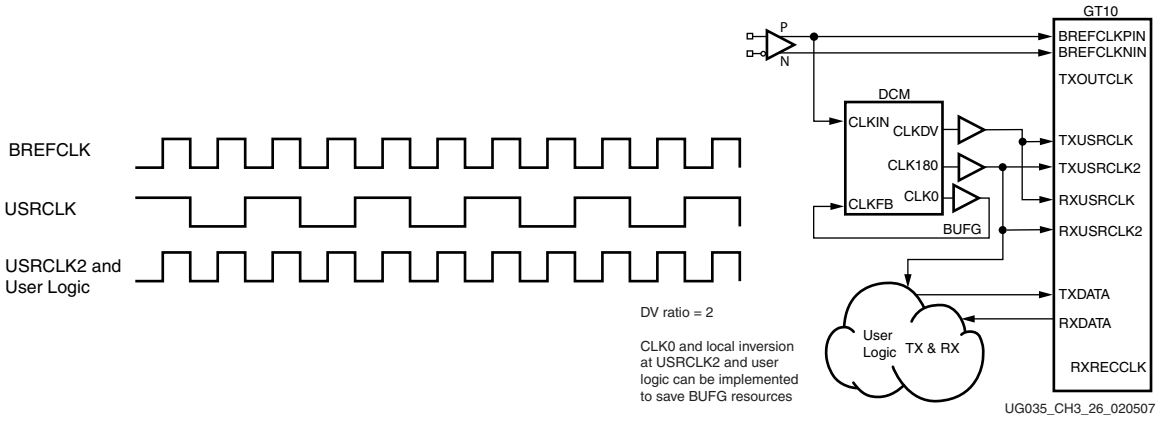


Figure 3-13: **BREFCLK 2:1:2**

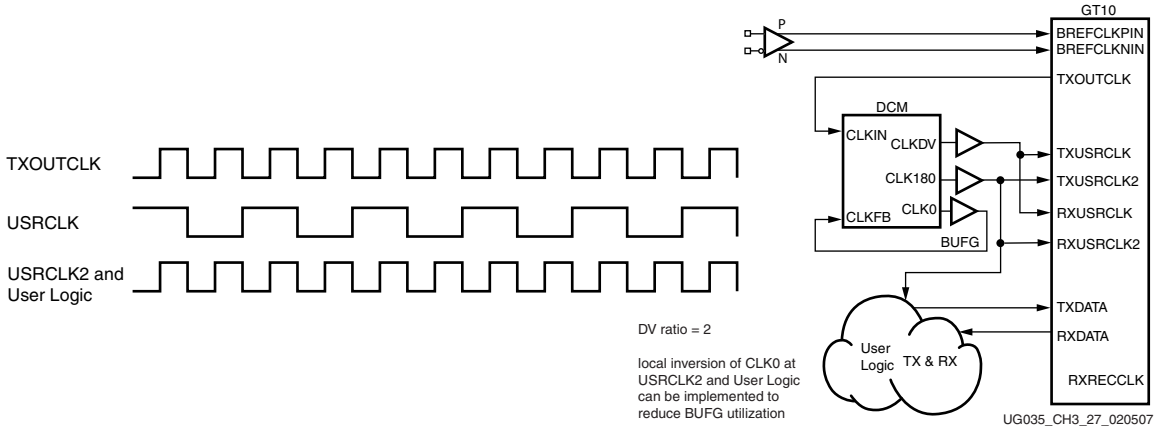


Figure 3-14: **TXOUTCLK 2:1:2**

Note: Figure 3-15 shows the RocketIO X transceiver instantiated using the recovered clock to clock in the FPGA fabric on the receive side. This can be used to avoid clock correction schemes. The TX can have any of the other 1:2 use models. Also, the waveform only indicates the receive clocks.

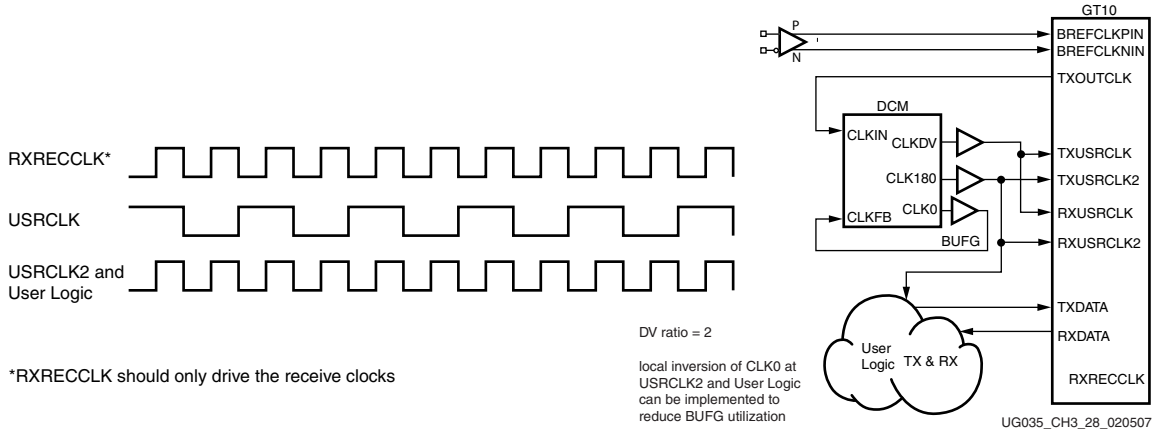


Figure 3-15: **RXRECCLK 2:1:2**

Supported Use Models for Each PMA Mode

The use models discussed in the previous section work for certain PMA modes. In some PMA modes, a BREFCLK use model is not available because the reference clock exceeds the DCM CLKIN maximum frequency. Other use models can only be run with the DCM in high frequency mode, and in some cases the DCM can only meet the speed in higher speed grades when the serial rate can be met in a lower speed grade. These restrictions are shown in Table 3-2. Note that these use models and DCM specifications were selected for the optimum serial rate for that specific mode. There might be further restrictions if the serial rates are run at rates other than the optimum rate (e.g., PMA Mode 11_32 optimum rate is 10.313 Gb/s but it can be run at 8 Gb/s).

Table 3-2: **Supported Use Models**

PMA MODE	Clocking Use Model Name	Comment/Restrictions
13_40	BREFCLK 1:1:1, TXOUTCLK 1:1:1, RXRECCLK 1:1:1	
13_80	BREFCLK 2:2:1, TXOUTCLK 2:2:1, RXRECCLK 2:2:1	
20_40	BREFCLK 2:1:1*, TXOUTCLK 1:1:1, RXRECCLK 1:1:1	DCM must be in HF mode
20_80	TXOUTCLK 2:2:1, RXRECCLK 2:2:1	
21_40	BREFCLK 2:1:1*, TXOUTCLK 1:1:1, RXRECCLK 1:1:1	DCM must be in HF mode
21_80	TXOUTCLK 2:2:1, RXRECCLK 2:2:1	
24_10	BREFCLK 2:1:2*, TXOUTCLK 2:1:2, RXRECCLK 2:1:2	DCM must be in HF mode and -7,-6 only
24_20	BREFCLK 2:1:1*,TXOUTCLK 1:1:1, RXRECCLK 1:1:1	DCM must be in HF mode and -7,-6 only
24_40	TXOUTCLK 2:2:1, RXRECCLK 2:2:1	
25_10	BREFCLK 1:1:2, TXOUTCLK 2:1:2, RXRECCLK 2:1:2	
25_20	BREFCLK 1:1:1, TXOUTCLK 1:1:1, RXRECCLK 1:1:1	
25_40	BREFCLK 2:2:1, TXOUTCLK 2:2:1, RXRECCLK 2:2:1	
26_10	BREFCLK 0:1:2*, TXOUTCLK 2:1:2, RXRECCLK 2:1:2	DCM must be in HF mode and -7,-6 only
26_20	BREFCLK 0:1:1, TXOUTCLK 1:1:1, RXRECCLK 1:1:1	
26_40	BREFCLK 1:2:1, TXOUTCLK 2:2:1, RXRECCLK 2:2:1	
27_10	BREFCLK 2:1:2*, TXOUTCLK 2:1:2, RXRECCLK 2:1:2	DCM must be in HF mode and -7,-6 only
27_20	BREFCLK 2:1:1*,TXOUTCLK 1:1:1, RXRECCLK 1:1:1	DCM must be in HF mode and -7,-6 only
27_40	TXOUTCLK 2:2:1, RXRECCLK 2:2:1	
28_10	BREFCLK 1:1:2, TXOUTCLK 2:1:2, RXRECCLK 2:1:2	
28_20	BREFCLK 1:1:1, TXOUTCLK 1:1:1, RXRECCLK 1:1:1	
28_40	BREFCLK 2:2:1, TXOUTCLK 2:2:1, RXRECCLK 2:2:1	
29_10	BREFCLK 0:1:2, TXOUTCLK 2:1:2, RXRECCLK 2:1:2	
29_20	BREFCLK 0:1:1, TXOUTCLK 1:1:1, RXRECCLK 1:1:1	
29_40	BREFCLK 1:2:1, TXOUTCLK 2:2:1, RXRECCLK 2:2:1	

Table 3-2: **Supported Use Models *(Continued)***

PMA MODE	Clocking Use Model Name	Comment/Restrictions
30_8	BREFCLK 1:1:2, TXOUTCLK 2:1:2, RXRECCLK 2:1:2	
30_16	BREFCLK 1:1:1, TXOUTCLK 1:1:1, RXRECCLK 1:1:1	
30_32	BREFCLK 2:2:1, TXOUTCLK 2:2:1, RXRECCLK 2:2:1	
31_8	BREFCLK 0:1:2, TXOUTCLK 2:1:2, RXRECCLK 2:1:2	
31_16	BREFCLK 0:1:1, TXOUTCLK 1:1:1, RXRECCLK 1:1:1	
31_32	BREFCLK 1:2:1, TXOUTCLK 2:2:1, RXRECCLK 2:2:1	

PMA

The PMA uses the PMA_SPEED attribute to set many aspects of the RocketIO X transceiver for a given serial rate. Many of the aspect set includes analog voltages, biasing, and drive strength optimized for a serial rate range. Other settings define the default equalization and pre-emphasis settings, plus the internal clocks for different internal and external bus widths.

The mode number (shown in Table 3-3) is comprised of two parts: the first number (rate number) and the second number (external bus width). The rate number (e.g., 30 in 30_16) does not have any direct correlation to serial speed. However, a rate number of 6 is for the fastest, 10.313 Gb/s. As the rate number increases the serial speed supported decreases to rate number 30 and 31 supporting 2.488 Gb/s.

The bus width number (e.g., 16 in 30_16) indicates the possible external data width. The definition of the values are as follows:

- Rate modes 15, 16, 17, 30, and 31 (such as 30_8 or 17_32) – 8, 16, 32, 64 support encoding bypass for SONET applications with the 16X-clock multiply and the corresponding 8, 16, 32, and 64 bit fabric bus widths.
- Rate modes 12, 13, 14, 18 to 29 – 10, 20, 40, 80 support 8B/10B encoding (internal to the MGT) to support 8-, 16-, 32- and 64-bit fabric bus widths or external 8B/10B encoding and other 40-bit compatible encoding schemes using 10-, 20-, 40- or 80-bit fabric bus widths.
- Rate modes 6 to 11 – 32 and 64 support 64B/66B encoding with fabric bus widths of 32 or 64

  Note: Rate modes 6 to 11 are the only modes that support 64B/66B encoding.

All modes supporting 5 Gb/s or greater support the 40/32-bit internal bus widths only, while all modes supporting speeds under 5 Gb/s only support 20/16-bit internal bus widths.

Table 3-3 shows the standard supported for any given mode, along with supported encoding, serial rate, bus width, and frequencies for reference clocks, TXOUTCLK, XRECCLK, USRCLK and USRCLK2.

Table 3-3: **Supported Standards, Speeds, Bus Widths, and Frequencies for Reference Clocks**

Mode Number	Supported Standard (1)	Serial Speed	REFCLK or BREFCLK (MHz)	TXOUTCLK (MHz)	RXRECCLK (MHz)	USRCLK (MHz)	USRCLK2 (MHz)	Internal(2) BUSWIDTH (bits)	External(2) BUSWIDTH (bits)	Encoding (3)
13_40	N/A(1)	6.25	156.25	156.25	156.25	156.25	156.25	40	40/32	None/ 8B/10B
13_80	N/A(1)	6.25	156.25	156.25	156.25	156.25	78.125	40	80/64	None/ 8B/10B
20_40	N/A	6.25	312.5	156.25	156.25	156.25	156.25	40	40/32	None/ 8B/10B
20_80	N/A	6.25	312.5	156.25	156.25	156.25	78.125	40	80/64	None/ 8B/10B
21_40	N/A	5	250	125	125	125	125	40	40/32	None/ 8B/10B
21_80	N/A	5	250	125	125	125	62.5	40	80/64	None/ 8B/10B
24_10	XAUI	3.125	312.5	312.5	312.5	156.25	312.5	20	10/8	None/ 8B/10B
24_20	XAUI	3.125	312.5	156.25	156.25	156.25	156.25	20	20/16	None/ 8B/10B
24_40	XAUI	3.125	312.5	156.25	156.25	156.25	78.125	20	40/32	None/ 8B/10B
25_10	XAUI	3.125	156.25	312.5	312.5	156.25	312.5	20	10/8	None/ 8B/10B
25_20	XAUI	3.125	156.25	156.25	156.25	156.25	156.25	20	20/16	None/ 8B/10B
25_40	XAUI	3.125	156.25	156.25	156.25	156.25	78.125	20	40/32	None/ 8B/10B
26_10	XAUI	3.125	78.125	312.5	312.5	156.25	312.5	20	10/8	None/ 8B/10B
26_20	XAUI	3.125	78.125	156.25	156.25	156.25	156.25	20	20/16	None/ 8B/10B
26_40	XAUI	3.125	78.125	156.25	156.25	156.25	78.125	20	40/32	None/ 8B/10B
27_10	PCI Express/ Infiniband	2.5	250	250	250	125	250	20	10/8	None/ 8B/10B
27_20	PCI Express/ Infiniband	2.5	250	125	125	125	125	20	20/16	None/8B /10B

Table 3-3: **Supported Standards, Speeds, Bus Widths, and Frequencies for Reference Clocks *(Continued)***

Mode Number	Supported Standard (1)	Serial Speed	REFCLK or BREFCLK (MHz)	TXOUTCLK (MHz)	RXRECCLK (MHz)	USRCLK (MHz)	USRCLK2 (MHz)	Internal(2) BUSWIDTH (bits)	External(2) BUSWIDTH (bits)	Encoding (3)
27_40	PCI Express/ Infiniband	2.5	250	125	125	125	62.5	20	40/32	None/ 8B/10B
28_10	PCI Express/ Infiniband	2.5	125	250	250	125	250	20	10/8	None/ 8B/10B
28_20	PCI Express/ Infiniband	2.5	125	125	125	125	125	20	20/16	None/ 8B/10B
28_40	PCI Express/ Infiniband	2.5	125	125	125	125	62.5	20	40/32	None/ 8B/10B
29_10	PCI Express/ Infiniband	2.5	62.5	250	250	125	250	20	10/8	None/8B /10B
29_20	PCI Express/ Infiniband	2.5	62.5	125	125	125	125	20	20/16	None/ 8B/10B
29_40	PCI Express/ Infiniband	2.5	62.5	125	125	125	62.5	20	40/32	None/ 8B/10B
30_8	OC48	2.4883	155.52	311.04	311.04	155.52	311.04	16	8	None
30_16	OC48	2.4883	155.52	155.52	155.52	155.52	155.52	16	16	None
30_32	OC48	2.4883	155.52	155.52	155.52	155.52	77.76	16	32	None
31_8	OC48	2.4883	77.76	311.04	311.04	155.52	311.04	16	8	None
31_16	OC48	2.4883	77.76	155.52	155.52	155.52	155.52	16	16	None
31_32	OC48	2.4883	77.76	155.52	155.52	155.52	77.76	16	32	None

Notes:

1. Settings optimized for that speed or standard.
2. See Table 2-2, page 42 and Table 2-3, page 42 on how to configure the data width for the internal and external bus widths.
3. Supported internally to the transceiver; other encode/decode schemes supported in the fabric.

Clock Dependency

All signals used by the FPGA fabric to interact between user logic and the transceiver depend on an edge of USRCLK2 (PMA attribute bus signals are asynchronous). These signals all have setup and hold times with respect to this clock. For specific timing values, see Module 3 of the Virtex-II Pro data sheet. The timing relationships are further discussed and illustrated in Appendix A, "RocketIO X Transceiver Timing Model."

Data Path Latency

With the many configurations of the Virtex-II Pro X transceiver, both the transmit and receive latency of the data path varies. Table 3-4 and Table 3-5 provide approximate latencies for common configurations.

Table 3-4: **Latency Through Various Transmitter Components/Processes**

Block	1 Byte	2 Byte	4 Byte	8 Byte
Fabric Interface(1)	2 TXUSRCLK2 + 2 TXUSRCLK	1 TXUSRCLK2 + 2 TXUSRCLK	1 TXUSRCLK2 + 1 TXUSRCLK	2 TXUSRCLK2 + 1 TXUSRCLK
Encoding: **8B/10B** **64B/66B** **Bypass**	 3 TXUSRCLK 2-3 TXUSRCLK 1 TXUSRCLK			
TXFIFO	4 TXUSRCLK (±.5) + 1 TXCLK0(3)			
64B/66B Scrambling	1-2 TXCLK0			
PMA Convert	1 TXCLK0 (approx.)			
Worst Case Total(2)	4 TXCLK0, 9.5 TXUSRCLK, 2 TXUSRCLK2	4 TXCLK0, 9.5 TXUSRCLK, 1 TXUSRCLK2	4 TXCLK0, 8.5 TXUSRCLK, 1 TXUSRCLK2	4 TXCLK0, 8.5 TXUSRCLK, 2 TXUSRCLK2
Best Case Total (2)	3 TXCLK0, 6.5 TXUSRCLK, 2 TXUSRCLK2	4 TXCLK0, 6.5 TXUSRCLK, 1 TXUSRCLK2	4 TXCLK0, 5.5 TXUSRCLK, 1 TXUSRCLK2	4 TXCLK0, 5.5 TXUSRCLK, 2 TXUSRCLK2

Notes:

1. Fabric interface has delays in both clock domains. The ratio between these two is shown in Table 3-3.
2. Approximate latency when worst/best case latency for each block is used.
3. TXCLK0 is equal in frequency to the TXUSRCLK.

Table 3-5: **Latency Through Various Receiver Components/Processes**

Block	1 Byte	2 Byte	4 Byte	8 Byte
PMA_PCS Interface(1)	3 RXCLK0(2)	3 RXCLK0	2 or 3 RXCLK0	2 RXCLK0
CommaDET/Align	2-3 RXCLK0			
Bypass	1 RXCLK0			
Decoding: **8B/10B** **64B/66B** **Bypass**	 2 RXCLK0 + 2 RXUSRCLK 2 RXCLK0 + 3 RXUSRCLK 1 RXCLK0 + 2 RXUSRCLK			
RXFIFO				
No Clock Correction	3 RXCLK0 + 10 RXUSRCLK			
Min/Max Used(3)	3 RXCLK0 + 1 RXUSRCLK + (MIN_LAT/4) < latency < 3 RXCLK0 + 1 RXUSRCLK + (MAX_LAT/4)			
Fabric Interface(4)	1 RXUSRCLK + 2 RXUSRCLK2	1 RXUSRCLK + 1 RXUSRCLK2	1-2 RXUSRCLK + 1 RXUSRCLK2	2 RXUSRCLK + 1 RXUSRCLK2
Worst Case Total(5)	12 RXCLK0, (5 + MAX_LAT/4) RXUSRCLK, 2 RXUSRCLK2	12 RXCLK0, (5 + MAX_LAT/4) RXUSRCLK, 1 RXUSRCLK2	12 RXCLK0, (6 + MAX_LAT/4) RXUSRCLK, 1 RXUSRCLK2	12 RXCLK0, (6 + MAX_LAT/4) RXUSRCLK, 1 RXUSRCLK2
Best Case Total (5)	9 RXCLK0, (4 + MIN_LAT/4) RXUSRCLK, 2 RXUSRCLK2	9 RXCLK0, (4 + MAX_LAT/4) RXUSRCLK, 1 RXUSRCLK2	9 RXCLK0, (6 + MAX_LAT/4) RXUSRCLK, 1 RXUSRCLK2	9 RXCLK0, (4 + MAX_LAT/4) RXUSRCLK, 1 RXUSRCLK2

Notes:

1. 4-byte mode can be run with internal data width of 2 or 4 byte. If 2 byte (lower serial speeds), the 3-clock delay must be used; otherwise 2-clock delay can be used (approx. clock cycles).
2. RXCLK0 is equal in frequency to the RXUSRCLK.
3. Clock correction must be enabled to use this feature.
4. Fabric interface has delays in both clock domains. The ratio between these two is shown in Table 3-3.
5. Approximate latency when worst/best case latency for each block is used.

Resets and Power Down

In the PCS, there are several reset and power down functions that have different effects. It is possible to only reset the PCS, which brings every flop in the PCS to a known value, but does not affect the PMA configuration. It is also possible to reset the PCS and at the same time re-initialize the PMA function, which would reload the PMA coefficients.

PCS Reset

When the signals TXRESET or RXRESET are asserted High, the PCS is considered in reset. After the signal that caused the reset is asserted Low, the PCS takes five clocks to come out of reset for each clock domain.

The PMA configuration vector is not affected during this reset, so the PMA speed, filter settings, and so on, all remain intact. Also, the PMA internal pipeline is not affected and continues to operate in normal fashion.

PCS/PMA Power Down

The POWERDOWN signal controls power down within the PCS and PMA. The following table describes the function of the POWERDOWN bus.

Table 3-6: **Power Control Descriptions**

POWERDOWN	Function
0	PCS and PMA function normally
1	PMA and PCS in powerdown mode

Chapter 4

Analog Design Considerations

Serial I/O Description

The RocketIO X transceiver transmits and receives serial differential signals, using a nominal supply voltage of 1.5 VDC. A serial differential pair consists of a true (V_P) and a complement (V_N) set of signals. The voltage difference represents the transferred data. Thus: $V_P - V_N = V_{DATA}$. Differential switching is performed at the crossing of the two complementary signals so that no separate reference level is needed.

A graphical representation of this concept is shown in Figure 4-1.

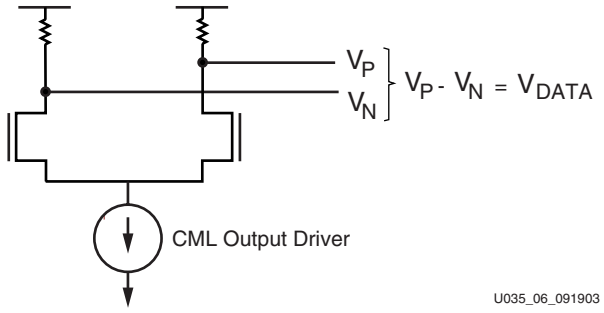


Figure 4-1: **Differential Amplifier**

Differential Transmitter

The RocketIO X transceiver is implemented in Current Mode Logic (CML). A CML transmitter output consists of transistors configured as shown in Figure 4-1. CML uses a positive supply and offers easy interface requirements. In this configuration, both legs of the driver, V_P and V_N, sink current, with one leg always sinking more current than its complement. The CML output consists of a differential pair with 50Ω source resistors. The signal swing is created by switching the current in a common-source differential pair. The differential transmitter specification is shown in Table 4-1.

Table 4-1: **Differential Transmitter Parameters**

Parameter		Min	Typ	Max	Units	Conditions
V_{OUT}	Serial output differential peak to peak (TXP/TXN)	200		1600	mV	Output differential voltage is programmable
V_{TTX}	Output termination voltage supply		1.5		V	
V_{TCM}	Common mode output voltage range	0.9		1.4	V	

Output Swing and Emphasis

The output swing and emphasis levels of the RocketIO X MGTs are fully programmable. Each is controlled via attributes at configuration, but can be modified via partial reconfiguration or the PMA attribute programming bus (Appendix C, "PMA Attribute Programming Bus").

Note: Table 4-2 through Table 4-5 are based on simulation only. See the Characterization Report for silicon-based tables.

Emphasis

With emphasis, the initial differential voltage swing is boosted to create a stronger rising or falling waveform. This method compensates for high frequency loss in the transmission media that would otherwise limit the magnitude of this waveform. The effects of emphasis are shown in four scope screen captures, Figure 4-2 through Figure 4-5. (Also, see "Register Definition" in Appendix C for the signal or attribute Effects.) Note that Figure 4-2 through Figure 4-5 are for illustration purposes only and do not correspond to actual RocketIO X data. The STRONG notation in Figure 4-3 is used to show that the waveform is greater in voltage magnitude, at this point, than the LOGIC or normal level (LOGIC level is the level with no emphasis).

A second characteristic of RocketIO X transceiver emphasis is that the STRONG level is reduced after some time to the LOGIC level, thereby minimizing the voltage swing necessary to switch the differential pair into the opposite state.

Lossy transmission lines cause the dissipation of electrical energy. This emphasis technique extends the distance that signals can be driven down lossy line media and increases the signal-to-noise ratio at the receiver.

Emphasis can be described from two perspectives, additive to the smaller voltage (V_{SM}) (pre-emphasis) or subtractive from the larger voltage (V_{LG}) (de-emphasis). The resulting benefits in compensating for channel loss are identical. It is simply a relative way of specifying the effect at the transmitter.

The equations for calculating Pre-Emphasis as a percentage and dB are as follows:

$$\text{Pre-Emphasis}_{\%} = ((V_{LG}-V_{SM}) / V_{SM}) \times 100$$
$$\text{Pre-Emphasis}_{dB} = 20 \log(V_{LG}/V_{SM})$$

The equations for calculating De-Emphasis as a percentage and dB are as follows:

$$\text{De-Emphasis}_{\%} = (V_{LG} - V_{SM}) / V_{LG}) \times 100$$
$$\text{De-Emphasis}_{dB} = 20 \log(V_{SM}/V_{LG})$$

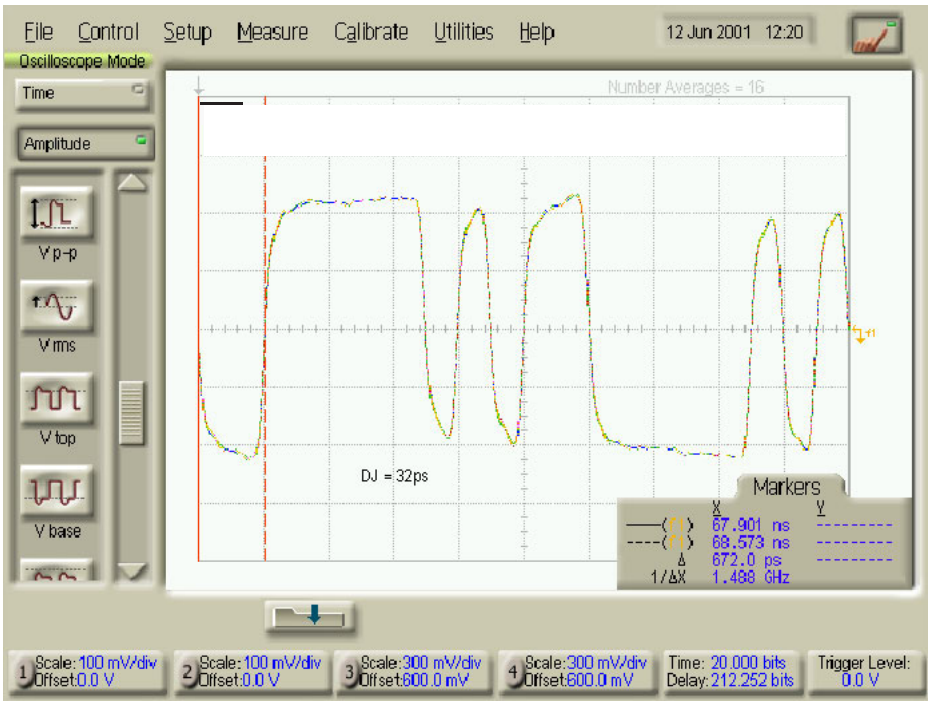


Figure 4-2: **Alternating K28.5+ Without Pre-Emphasis**

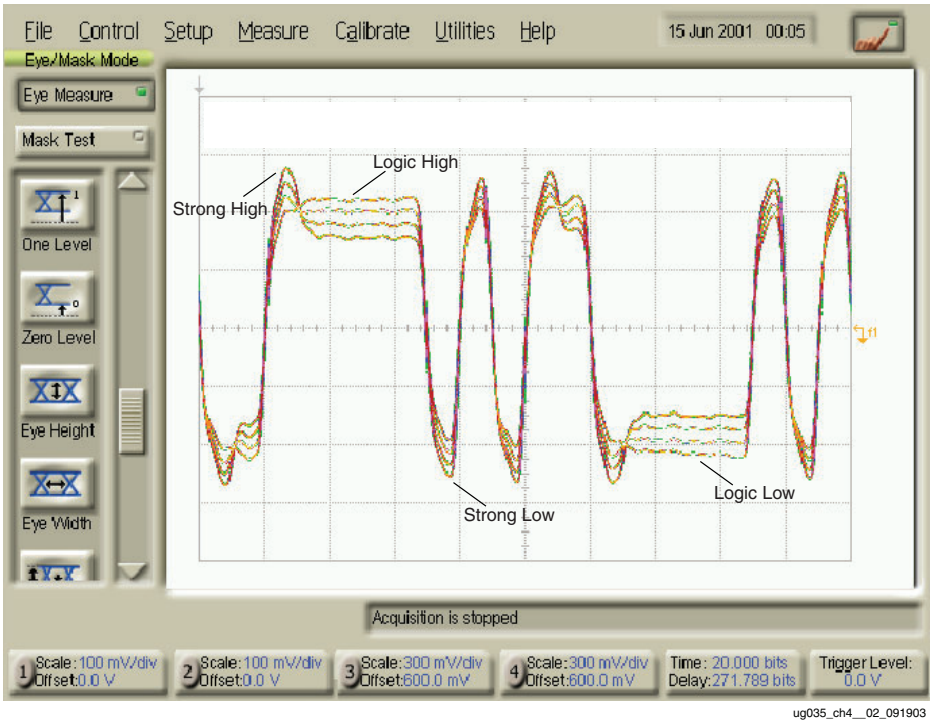


Figure 4-3: **K28.5+ With Pre-Emphasis**

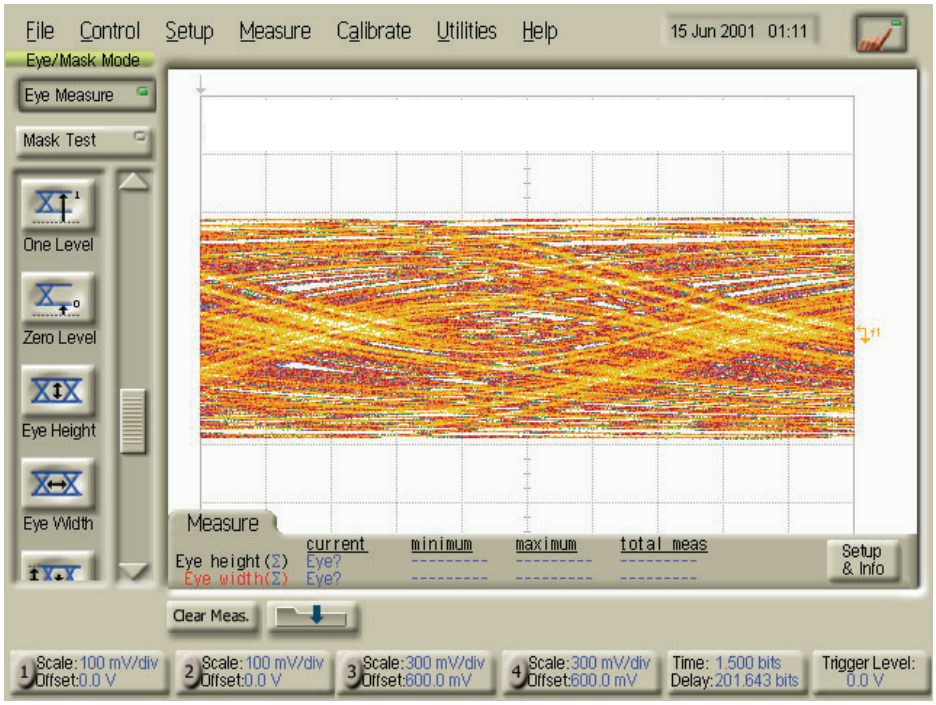


Figure 4-4: **Eye Diagram: Without Pre-Emphasis**

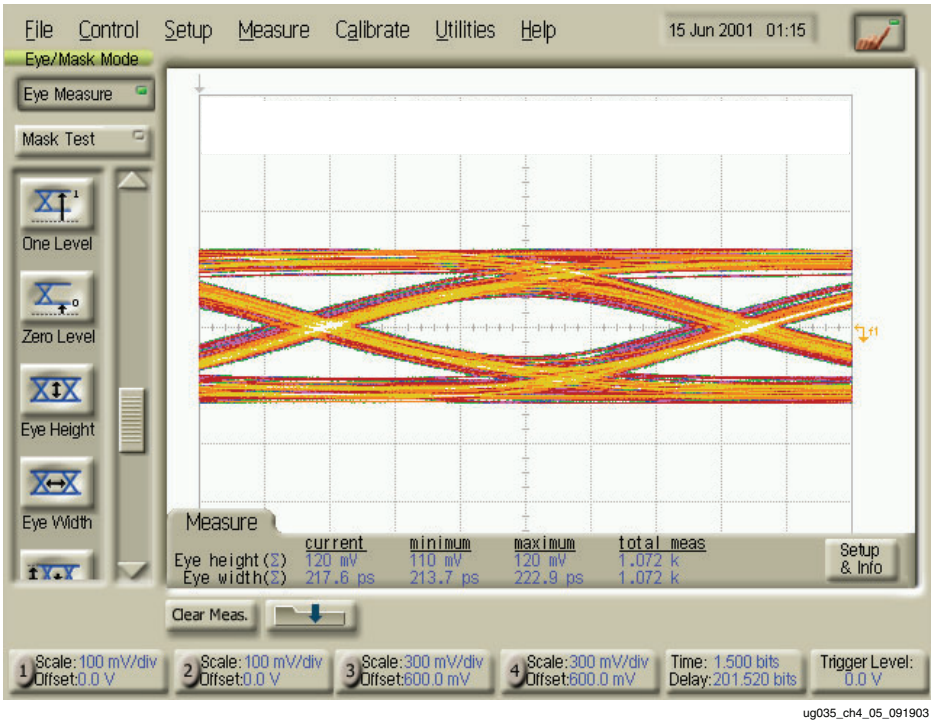


Figure 4-5: **Eye Diagram: With Pre-Emphasis**

DC Coupled

When the TX output and RX input of two MGTs are DC coupled, the transmitter sees both an AC and DC impedance of 25Ω (50Ω||50Ω). The drive/emphasis settings for supported single-ended output swing (Vos) and pre-emphasis levels (%, dB) when DC coupled are shown in Table 4-2. The unused (grey) combinations are not supported and should not be used.

Table 4-2: **Output Swing versus Pre-Emphasis (DC Coupled)**

TXEMPHLEVEL	TXDOWNLEVEL 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0			120mV 0% 0.0dB	180mV 0% 0.0dB	240mV 0% 0.0dB	300mV 0% 0.0dB	360mV 0% 0.0dB	420mV 0% 0.0dB	480mV 0% 0.0dB	540mV 0% 0.0dB	600mV 0% 0.0dB	660mV 0% 0.0dB	720mV 0% 0.0dB	780mV 0% 0.0dB		
1				150mV 40% 2.9dB	210mV 29% 2.2dB	270mV 22% 1.7dB	330mV 18% 1.5dB	390mV 15% 1.2dB	450mV 13% 1.1dB	510mV 12% 1.0dB	570mV 11% 0.9dB	630mV 10% 0.8dB	690mV 9% 0.7dB			
2				120mV 100% 6.0dB	180mV 67% 4.4dB	240mV 50% 3.5dB	300mV 40% 2.9dB	360mV 33% 2.5dB	420mV 29% 2.2dB	480mV 25% 1.9dB	540mV 22% 1.7dB	600mV 20% 1.6dB	660mV 18% 1.5dB			
3					150mV 120% 6.8dB	210mV 86% 5.4dB	270mV 67% 4.4dB	330mV 55% 3.8dB	390mV 46% 3.3dB	450mV 40% 2.9dB	510mV 35% 2.6dB	570mV 32% 2.4dB				
4					120mV 100% 6.0dB	180mV 133% 7.4dB	240mV 100% 6.0dB	300mV 80% 5.1dB	360mV 67% 4.4dB	420mV 57% 3.9dB	480mV 50% 3.5dB	540mV 44% 3.2dB				
5						150mV 200% 9.5dB	210mV 143% 7.7dB	270mV 111% 6.5dB	330mV 91% 5.6dB	390mV 77% 5.0dB	450mV 67% 4.4dB					
6						120mV 300% 12.0dB	180mV 200% 9.5dB	240mV 150% 8.0dB	300mV 120% 6.8dB	360mV 100% 6.0dB	420mV 86% 5.4dB					
7							150mV 280%1 11.6dB	210mV 200% 9.5dB	270mV 156% 8.1dB	330mV 127% 7.1dB						
8							120mV 400% 14.0dB	180mV 267% 11.3dB	240mV 200% 9.5dB	300mV 160% 8.3dB						
9								150mV 360% 13.3dB	210mV 257% 11.1dB							
10								120mV 500% 15.6dB	180mV 333% 12.7dB							
11																
12																
13																
14																
15																

Figure 4-6 and Figure 4-7 illustrate the above DC coupled Vos versus Pre-Emphasis in percent and dB, respectively.

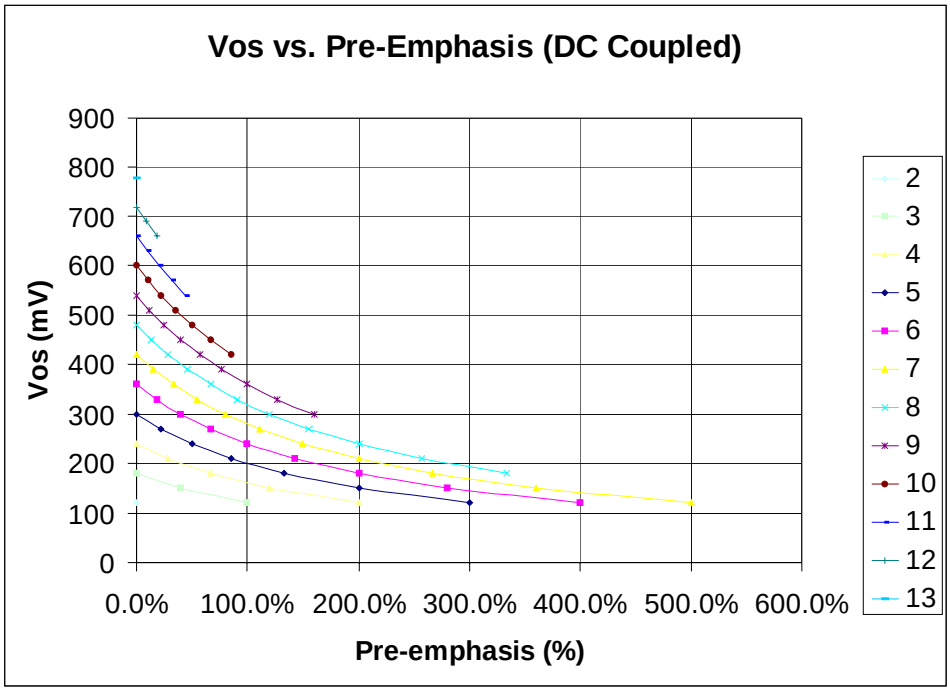


Figure 4-6: **Output Swing versus Pre-Emphasis (%) When DC Coupled**

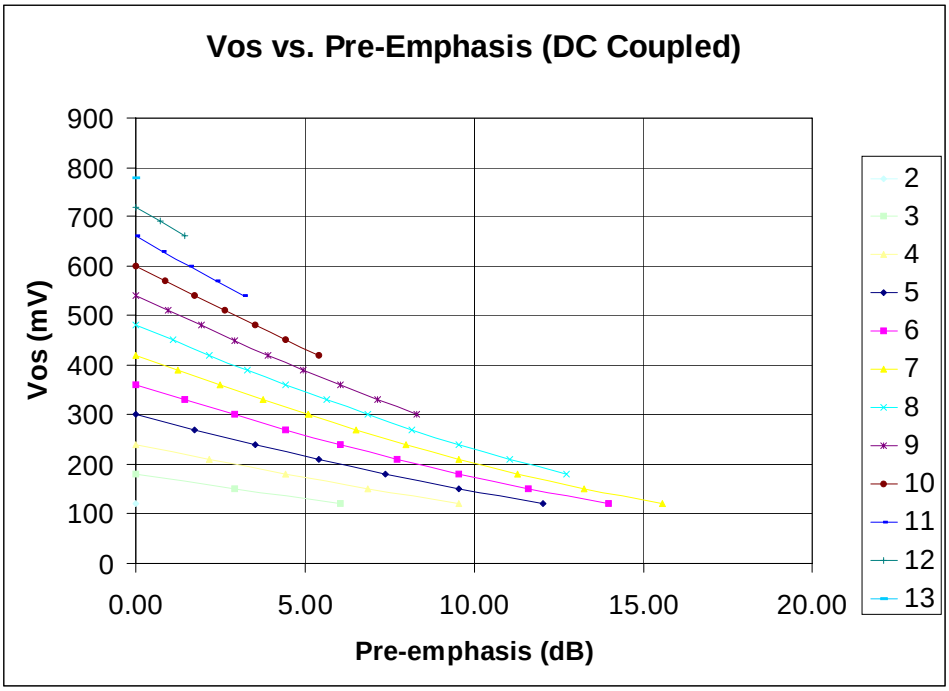


Figure 4-7: **Output Swing versus Pre-Emphasis (dB) When DC Coupled**

The drive/emphasis settings for supported single-ended output swing (Vos) and de-emphasis levels (%, dB) when DC coupled are shown in Table 4-3. The unused (grey) combinations are not supported and should not be used.

Table 4-3: **Output Swing versus De-Emphasis (DC Coupled)**

TXEMPHLEVEL	TXDOWNLEVEL 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0			120mV 0% 0.0dB	180mV 0% 0.0dB	240mV 0% 0.0dB	300mV 0% 0.0dB	360mV 0% 0.0dB	420mV 0% 0.0dB	480mV 0% 0.0dB	540mV 0% 0.0dB	600mV 0% 0.0dB	660mV 0% 0.0dB	720mV 0% 0.0dB	780mV 0% 0.0dB		
1				210mV 29% -2.9dB	270mV 22% -2.2dB	330mV 18% -1.7dB	390mV 15% -2.9dB	450mV 13% -1.2dB	510mV 12% -1.1dB	570mV 11% -1.0dB	630mV 10% -0.9dB	690mV 9% -0.8dB	750mV 8% -0.7dB			
2				240mV 50% -6.0dB	300mV 40% -4.4dB	360mV 33% -3.5dB	420mV 29% -2.9dB	480mV 25% -2.5dB	540mV 22% -2.2dB	600mV 20% -1.9dB	660mV 18% -1.7dB	720mV 17% -1.6dB	780mV 15% -1.5dB			
3					330mV 55% -6.8dB	390mV 46% -5.4dB	450mV 40% -4.4dB	510mV 35% -3.8dB	570mV 32% -3.3dB	630mV 29% -2.9dB	690mV 26% -2.6dB	750mV 15% -2.4dB				
4					360mV 67% -9.5dB	420mV 57% -7.4dB	480mV 50% -6.0dB	540mV 44% -5.1dB	600mV 40% -4.4dB	660mV 36% -3.9dB	720mV 33% -3.5dB	780mV 31% -3.2dB				
5						450mV 67% -9.5dB	510mV 59% -7.7dB	570mV 53% -6.5dB	630mV 48% -5.6dB	690mV 43% -5.0dB	750mV 40% -4.4dB					
6						480mV 75% -12.0dB	540mV 67% -9.5dB	600mV 60% -8.0dB	660mV 55% -6.8dB	720mV 50% -6.0dB	780mV 46% -5.4dB					
7							570mV 74%1 -11.6dB	630mV 67% -9.5dB	690mV 61% -8.1dB	750mV 56% -7.1dB						
8							600mV 80% -14.0dB	660mV 73% -11.3dB	720mV 67% -9.5dB	780mV 62% -8.3dB						
9								690mV 78% -13.3dB	750mV 72% -11.1dB							
10								720mV 83% -15.6dB	780mV 77% -12.7dB							
11																
12																
13																
14																
15																

Figure 4-8 and Figure 4-9 illustrate the above DC coupled Vos versus De-Emphasis in percent and dB, respectively.

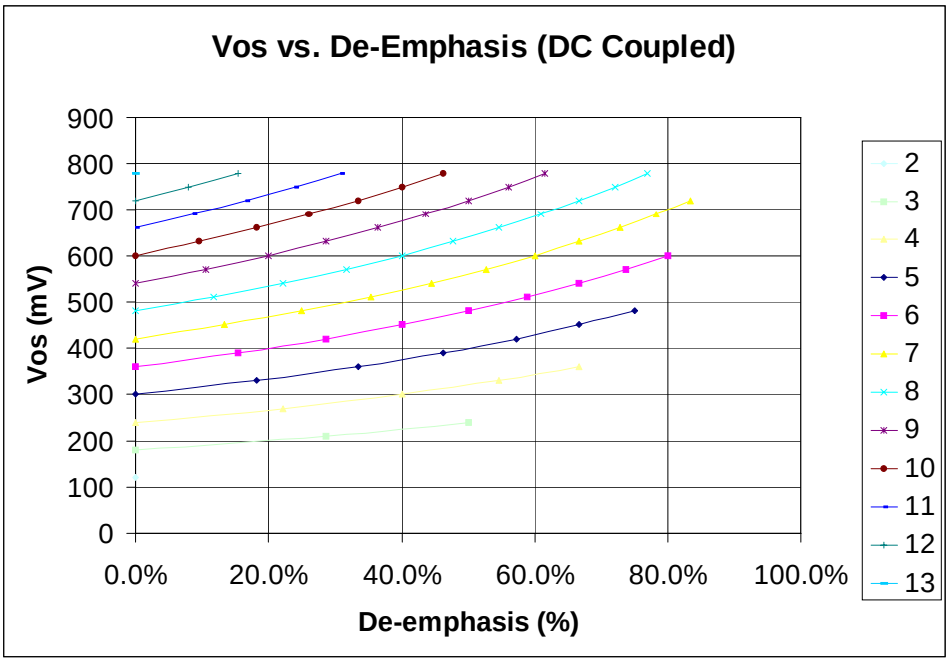


Figure 4-8: **Output Swing versus De-Emphasis (%) When DC Coupled**

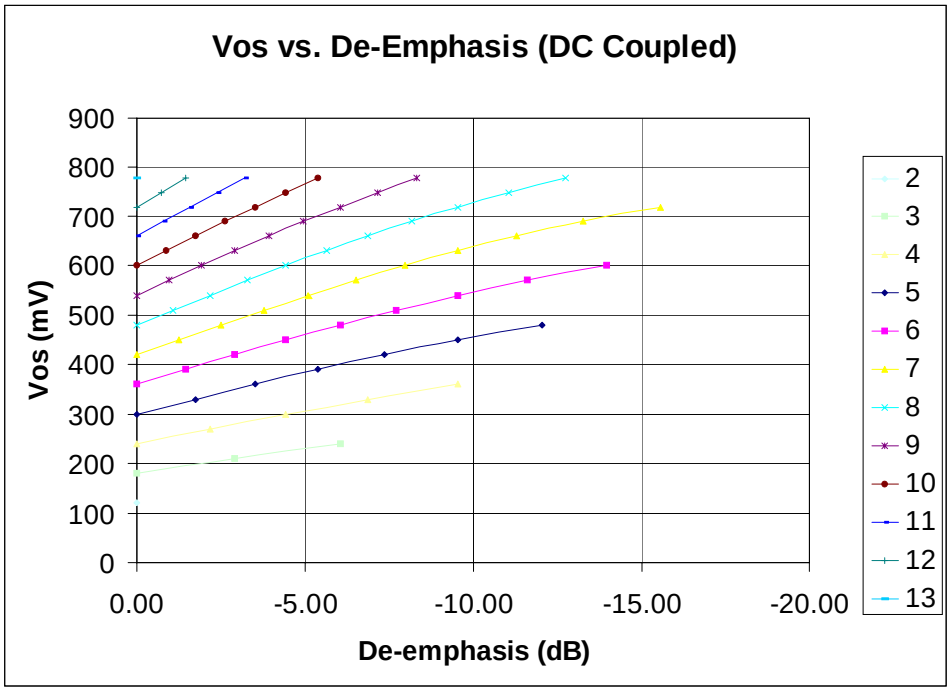


Figure 4-9: **Output Swing versus De-Emphasis (dB) When DC Coupled**

AC Coupled

When the transmit output and receive input of two MGTs are AC coupled and/or the receiver is differentially terminated, the transmitter sees an AC impedance of 25Ω (50Ω ||50Ω). However, the DC impedance is now 50Ω, thus changing the common mode voltage from the DC coupled case. The drive/emphasis settings for supported single-ended output swing (Vos) and pre-emphasis levels (%, dB) when AC coupled are shown in Table 4-4. The unused (grey) combinations are not supported and should not be used.

Table 4-4: **Output Swing versus Pre-Emphasis (AC Coupled)**

TXEMPHLEVEL	TXDOWNLEVEL 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0			120mV 0% 0.0dB	180mV 0% 0.0dB	240mV 0% 0.0dB	300mV 0% 0.0dB	360mV 0% 0.0dB	420mV 0% 0.0dB	480mV 0% 0.0dB							
1				150mV 40% 2.9dB	210mV 29% 2.2dB	270mV 22% 1.7dB	330mV 18% 2.9dB	390mV 15% 1.2dB	450mV 13% 1.1dB							
2				120mV 100% 6.0dB	180mV 67% 4.4dB	240mV 50% 3.5dB	300mV 40% 2.9dB	360mV 33% 2.5dB								
3					150mV 120% 6.8dB	210mV 86% 5.4dB	270mV 67% 4.4dB	330mV 55% 3.8dB								
4					120mV 200% 9.5dB	180mV 133% 7.4dB	240mV 100% 6.0dB									
5						150mV 200% 9.5dB	210mV 143% 7.7dB									
6						120mV 300% 12.0dB										
7																
8																
9																
10																
11																
12																
13																
14																
15																

Figure 4-10 and Figure 4-11 illustrate the above AC couple Vos versus Pre-Emphasis in percent and dB, respectively.

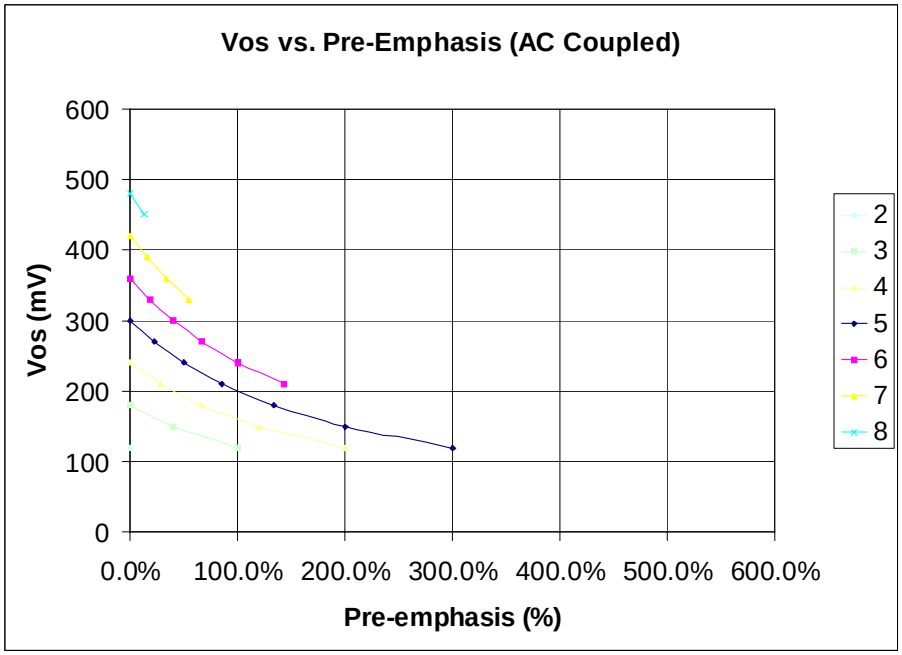


Figure 4-10: **Output Swing versus Pre-Emphasis (%) When AC Coupled**

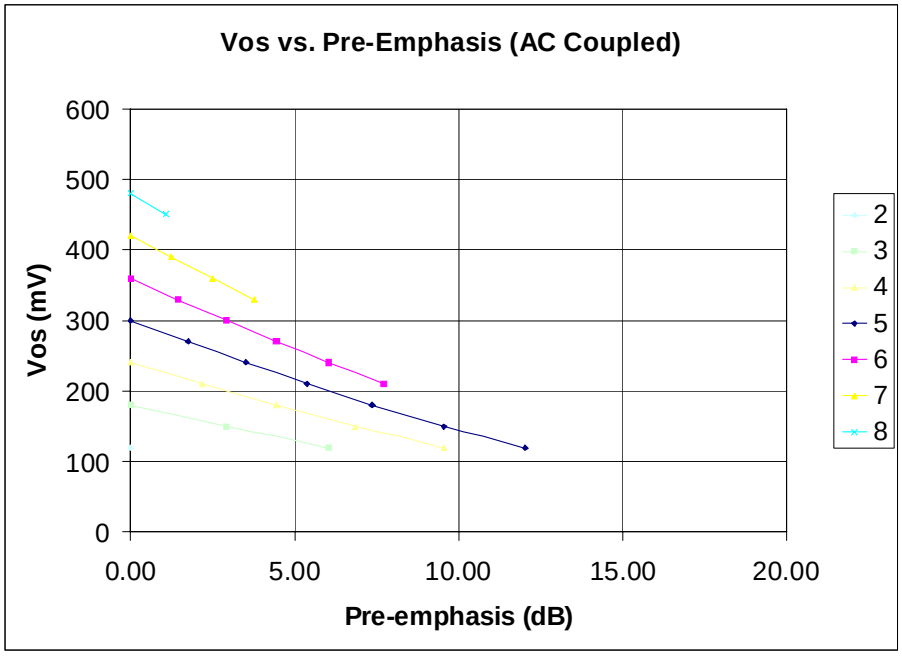


Figure 4-11: **Output Swing versus Pre-Emphasis (dB) When AC Coupled**

The drive emphasis settings for supported single-ended output swing (Vos) and de-emphasis levels (%, dB) when AC coupled are shown in Table 4-5. The unused (grey combinations are not supported and should not be used.

Table 4-5: **Output Swing versus De-Emphasis (AC Coupled)**

TXEMPHLEVEL	TXDOWNLEVEL 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0			120mV 0% 0.0dB	180mV 0% 0.0dB	240mV 0% 0.0dB	300mV 0% 0.0dB	360mV 0% 0.0dB	420mV 0% 0.0dB	480mV 0% 0.0dB							
1				210mV 29% -2.9dB	270mV 22% -2.2dB	330mV 18% -1.7dB	390mV 15% -1.5dB	450mV 13% -1.2dB	510mV 12% -1.1dB							
2				240mV 50% -6.0dB	300mV 40% -4.4dB	360mV 33% -3.5dB	420mV 29% -2.9dB	480mV 25% -2.5dB								
3					330mV 55% -6.8dB	390mV 46% -5.4dB	450mV 40% -4.4dB	510mV 35% -3.8dB								
4					360mV 67% -9.5dB	420mV 57% -7.4dB	480mV 50% -6.0dB									
5						450mV 67% -9.5dB	510mV 59% -7.7dB									
6						480mV 75% -12.0dB										
7																
8																
9																
10																
11																
12																
13																
14																
15																

Figure 4-12 and Figure 4-13 illustrate the above AC coupled Vos versus De-Emphasis in percent and dB, respectively.

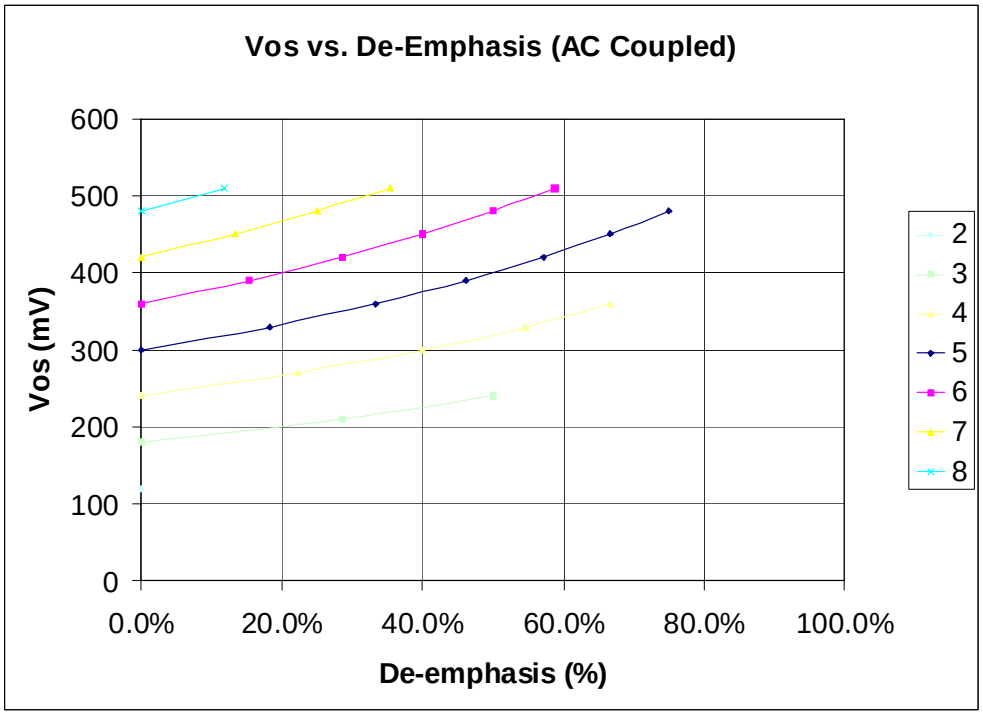


Figure 4-12: **Output Swing versus De-Emphasis (%) When AC Coupled**

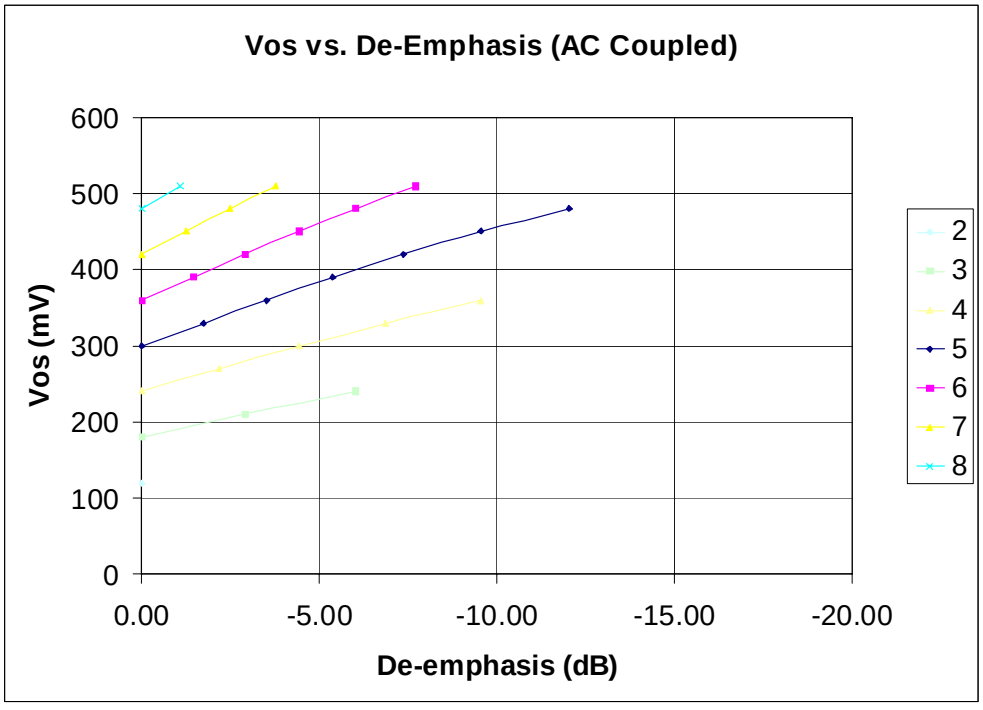


Figure 4-13: **Output Swing versus De-Emphasis (dB) When AC Coupled**

Differential Receiver

The differential receiver accepts the V_P and V_N signals, carrying out the difference calculation V_P - V_N electronically.

All input data must be differential and nominally biased to a common mode voltage of 0.25 V – 2.5 V, or AC coupled. Internal terminations provide for simple 50Ω transmission line connection.

The differential receiver parameters are shown in Table 4-6.

Table 4-6: **Differential Receiver Parameters**

Parameter		Min	Typ	Max	Units	Conditions
V_{IN}	Serial input differential peak to peak (RXP/RXN)			2000	mV	
V_{ICM}	Common mode input voltage range	250		2500	mV	
T_{JTOL}	Receive data total jitter tolerance (peak to peak)				UI(1)	
T_{DJTOL}	Receive data deterministic jitter tolerance (peak to peak)				UI(1)	

Notes:

1. UI = Unit Interval = 1 bit period.
2. See DS083 for jitter tolerance values.

Jitter

Jitter is defined as the short-term variations of significant instants of a signal from their ideal positions in time (ITU). Jitter is typically expressed in a decimal fraction of Unit Interval (UI), e.g., 0.3 UI.

Total Jitter (DJ + RJ)

Deterministic Jitter (DJ)

DJ is data pattern dependant jitter, attributed to a unique source (e.g., Inter Symbol Interference (ISI) due to loss effects of the media). DJ is linearly additive.

Random Jitter (RJ)

RJ is due to stochastic sources, such as substrate, power supply, etc. RJ is additive as the sum of squares, and follows a bell curve.

Clock and Data Recovery

The serial transceiver input is locked to the input data stream through Clock and Data Recovery (CDR), a built-in feature of the RocketIO X transceiver. CDR keys off of the rising and falling edges of incoming data and derives a clock that is representative of the incoming data rate.

The derived clock, RXRECCLK, is generated and locked to as long as it remains within the specified component range. This clock is presented to the FPGA fabric at 1/16th to 1/40th the incoming data rate depending on mode. Refer to the data sheet (DS083) for CDR timing specifications.

A sufficient number of transitions must be present in the data stream for CDR to work properly. The CDR circuit is guaranteed to work with 8B/10B and 64B/66B encoding. Further, CDR requires approximately 5,000 transitions upon power-up to guarantee locking to the incoming data rate. Once lock is achieved, up to 75 missing transitions can be tolerated before lock to the incoming data stream is lost.

Another feature of CDR is its ability to accept an external precision clock, REFCLK, which either acts to clock incoming data or to assist in synchronizing the derived RXRECCLK. REFCLK acts either to clock incoming data or to assist in synchronizing the derived RXRECCLK.

For further clarity, the TXUSRCLK is used to clock data from the FPGA fabric to the TX FIFO. The FIFO depth accounts for the slight phase difference between these two clocks. If the clocks are locked in frequency, then the FIFO acts much like a pass-through buffer.

Receiver Lock Control

During normal operation, the receiver PLL automatically locks to incoming data (when present) or to the local reference clock (when data is not present). This is the default configuration for all primitives. This function can be overridden via the PMARXLOCKSEL[1:0] port, as defined in Table 4-7.

Table 4-7: **PMARXLOCKSEL[1:0] Definition**

PMARXLOCKSEL[1:0]	Description
00	Automatic (Default)
01	Lock to local reference
10	Lock to receive data
11	Reserved (do not use)

When receive PLL lock is forced to the local reference, phase information from the incoming data stream is ignored. Data continues to be sampled, but synchronous to the local reference rather than relative to edges in the data stream.

When receive PLL lock is forced to the incoming data, the local reference clock is ignored. The recovered clock is not guaranteed to be within any tolerance of the local reference and might “wander” in the absence of data and/or transitions.

Receive Equalization

In addition to transmit emphasis, the RocketIO X transceiver provides a programmable receive equalization feature to further compensate the effects of channel attenuation at high frequencies.

Copper traces on printed circuit boards, including backplanes, have low-pass filter characteristics which severely attenuate signals in the GHz and above range. Reflections at the impedance mismatch boundaries can also cause signal degradation. The RocketIO X transceiver has an equalizer in the receiver which can essentially null the lossy attenuation effects of the PCB at GHz frequencies.

RXFER[9:0] is a 10-bit register that adjusts this receive equalization. By adjusting these bits, the right amount of equalization can be added to reverse the signal degradation caused by a PCB. Users are encouraged to calculate or measure the loss function/curve of their channel and then try to fit an equalizer function which is inverse of the channel loss function. The closer the fit, the better the effect is; in essence, it is possible to completely nullify the PCB effects with the right bit settings.

RXFER[9:0] can be set through software configuration or the PMA Attribute Bus. Refer to Appendix C, "PMA Attribute Programming Bus."

Other registers that control certain aspects of the receiver are also described in Appendix C, "PMA Attribute Programming Bus." Among them, the following registers are important:

- RXFEI[1:0] controls the receiver analog front end (AFE)/equalizer current
- RXLOOPFILTERC[1:0] selects the receiver PLL filter capacitor setting
- RXLOOPFILTERR[2:0] selects the receiver PLL filter resistor setting.

Figure 4-14 is a plot of the magnitude frequency response for all 1024 states of RXFER[9:0].

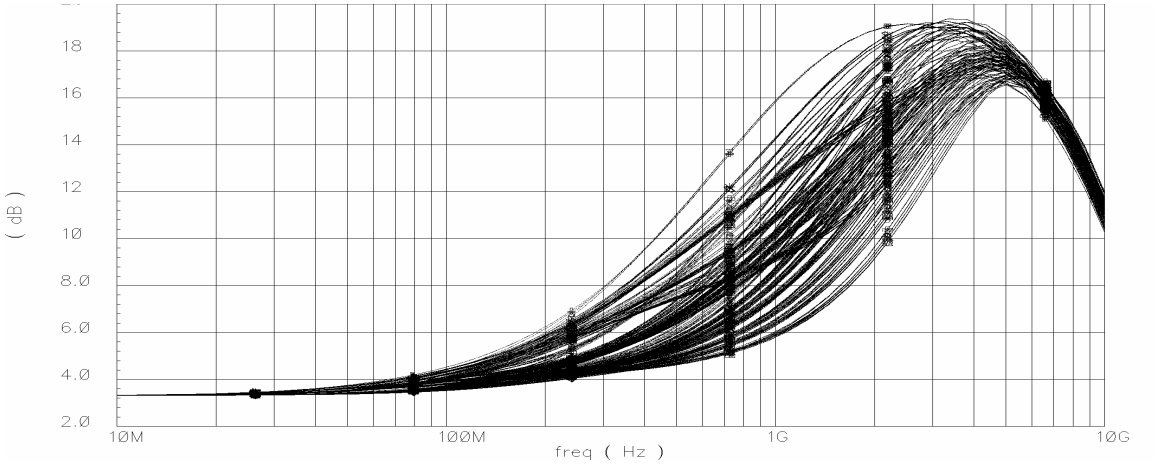


Figure 4-14: **Magnitude (dB) vs. Frequency (Hz) Plot for all 1024 states of RXFER[9:0]**

RXFER[9:0] can be broken out into four groups from the lowest to the highest frequency ranges. Depending on the bit settings of RXFER[9:0], certain frequency ranges are boosted in magnitude:

- RXFER[3:2] adjusts a boost in the 50 MHz to 200 MHz range (low frequency)
- RXFER[1:0] adjusts a boost in the 200 MHz to 1 GHz range (mid frequency)

- RXFER[6:4] adjusts a boost in the 500 MHz to 2 GHz range (high frequency)
- RXFER[9:7] adjusts a boost in the 500 MHz to 2 GHz range (high frequency)

These groupings represent cascaded boosting stages; hence, adjusting more than one stage results in a cumulative response. RXFER[6:4] and RXFER[9:7] control identical cascaded stages.

Note: The frequency ranges do not follow LSB to MSB with regard to which frequency range is boosted.

Low Frequency Boosting

RXFER[3:2] adjusts a boost in the 50 MHz to 200 MHz range. Figure 4-15 is a plot of the magnitude frequency response for RXFER[3:2], while the other 8 bits are all 1s. RXFER[9:0] = `111111XX11`.

The frequency steps here do not follow LSB to MSB. RXFER[2] is higher weight than RXFER[3]. Hence, the boost of the transfer function is set at low to high frequencies by the bits RXFER[3:2] = 00, 10, 01 and 11, respectively.

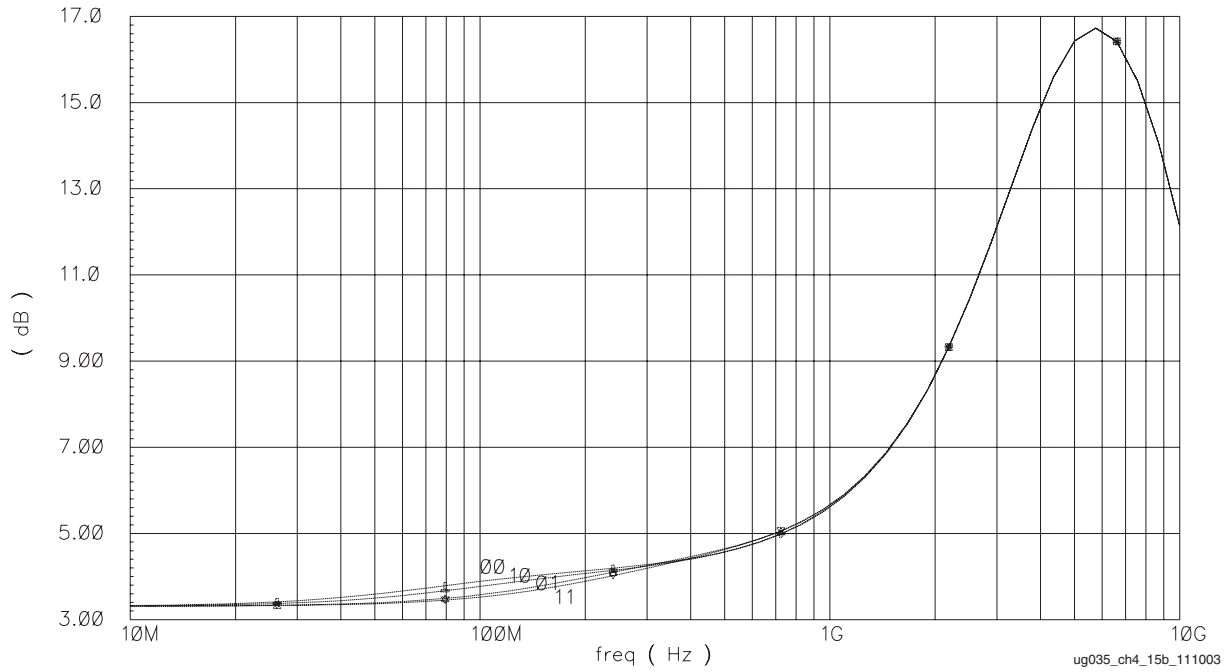


Figure 4-15: **Magnitude (dB) vs. Frequency (Hz) Response for Four Settings of RXFER[3:2]**

Mid Frequency Boosting

RXFER[1:0] adjust a boost in the 200 MHz to 1 GHz range.

Figure 4-16 is a plot of the magnitude frequency response for RXFER[1:0], while the other 8 bits are all 1s. RXFER[9:0] = `1111111XX`

In this case, however, the frequency steps follow LSB to MSB, i.e., RXFER[1] is higher weight than RXFER[0]. Hence, the boost of the transfer function is set at low to high frequencies by the bits RXFER[1:0] = `00`, `01`, `10`, and `11` respectively.

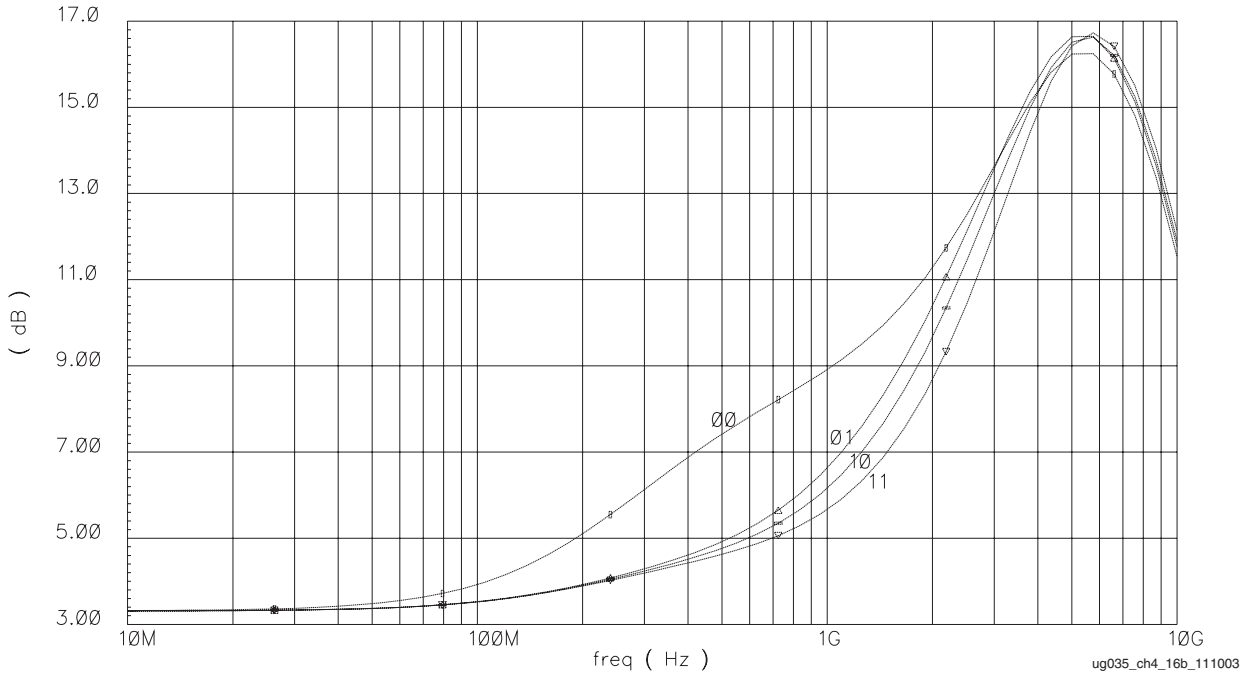


Figure 4-16: **Magnitude (dB) vs. Frequency (Hz) Response for Four Settings of RXFER[1:0]**

High Frequency Boosting

RXFER[6:4] adjusts a boost in the 500 MHz to 2 GHz range.

Figure 4-17 is a plot of the magnitude frequency response for RXFER[6:4], while the other 7 bits are all 1s. RXFER[9:0] = `111XXX1111`

The frequency steps here do not follow LSB to MSB. The order of the lowest weight bit to the highest weight bit is RXFER[5], RXFER[6] and then RXFER[4]. Hence, the boost of the transfer function is set at low to high frequencies by the bits RXFER[6:4] = `000`, `010`, `100`, `110`, `001`, `011`, `101`, and `111` respectively.

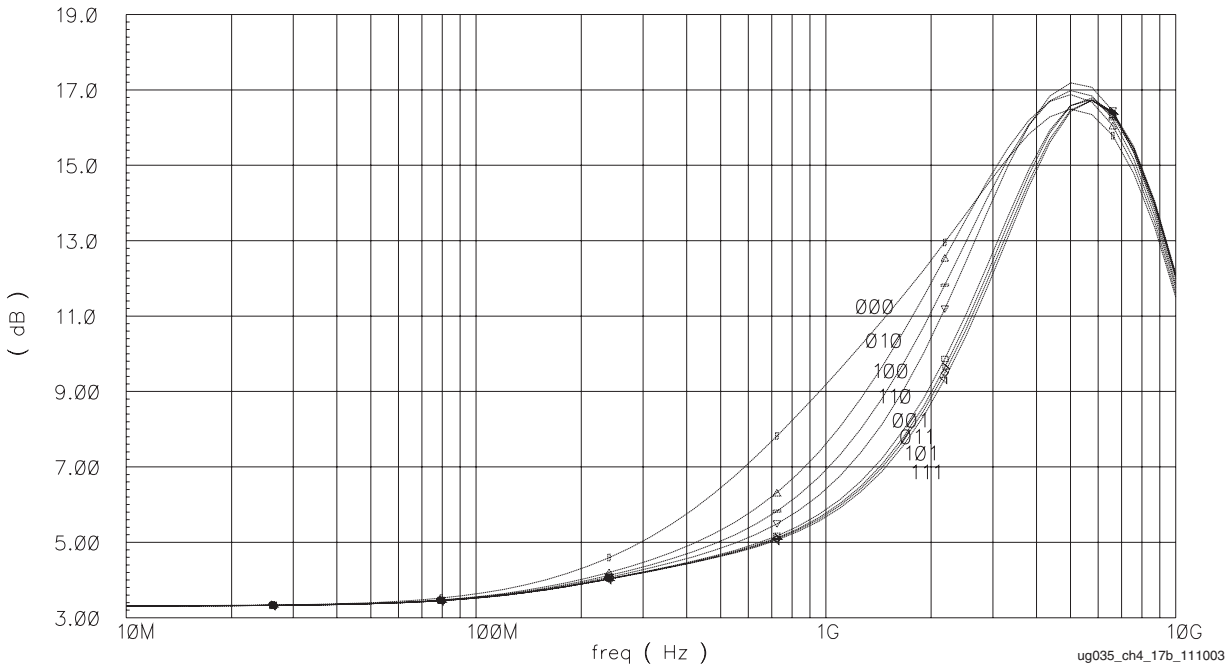


Figure 4-17: **Magnitude (dB) vs. Frequency (Hz) Response for Eight Settings of RXFER[6:4]**

RXFER[9:7] also adjusts a boost in the 500 MHz to 2 GHz range.

Figure 4-18 is a plot of the magnitude frequency response for RXFER[9:7], while the other 7 bits are all 1s. RXFER[9:0] = XXX1111111.

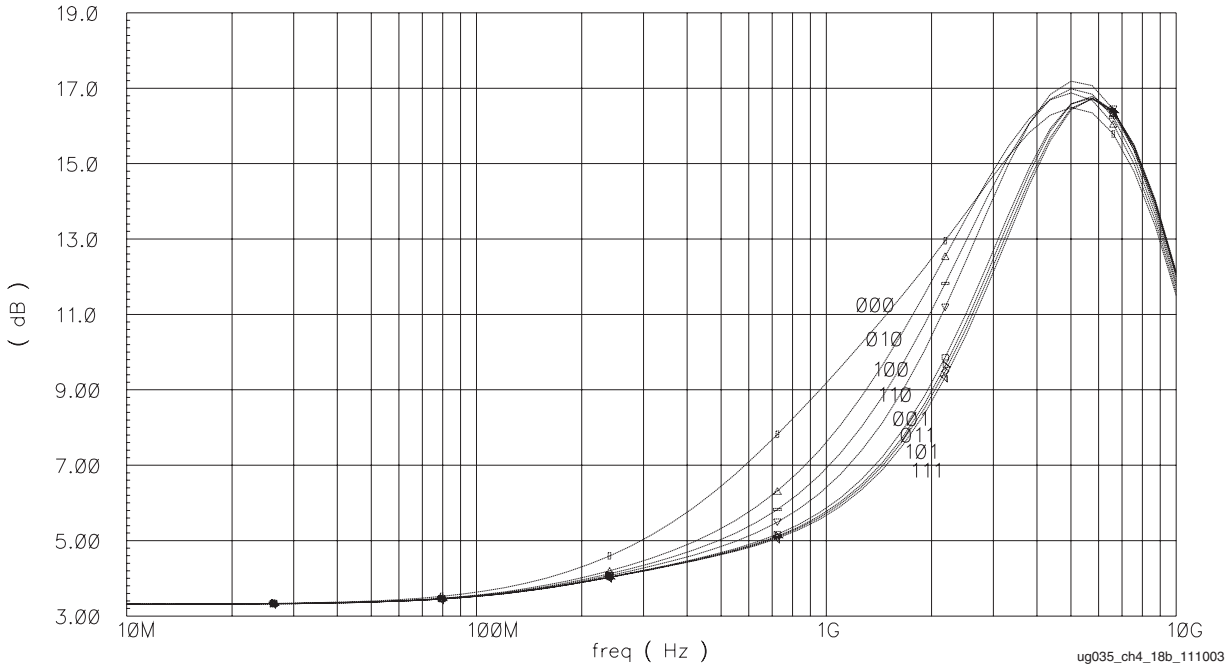


Figure 4-18: **Magnitude (dB) vs. Frequency (Hz) Response for Eight Settings of RXFER[9:7]**

The frequency steps in this case do not follow LSB to MSB. The order of lowest weight bit to highest weight bit is RXFER[8], RXFER[9] then RXFER[7]. Hence, the boost of the transfer function is set at low to high frequencies by the bits RXFER[9:7] = 000, 010, 100, 110, 001, 011, 101, and 111 respectively.

It can be easily observed from Figure 4-17 and Figure 4-18 that the magnitude vs. frequency curves for RXFER[9:7] are very similar to those curves for RXFER[6:4]. They represent two identical cascaded stages. Changing all six bits results in a cumulative change in the boost frequency.

Figure 4-19 shows the results for 8 of the 64 states represented by RXFER[9:4].

Figure 4-19 shows a plot of the eight states where:

- RXFER[9] = RXFER[6]
- RXFER[8] = RXFER[5]
- RXFER[7] = RXFER[4]
- RXFER[3:0] = 1111

As stated before, the frequency steps do not follow LSB to MSB here. The order of the lowest weight bit to the highest weight bit is RXFER[8], RXFER[9], RXFER[7], and RXFER[5], RXFER[6], RXFER[4].

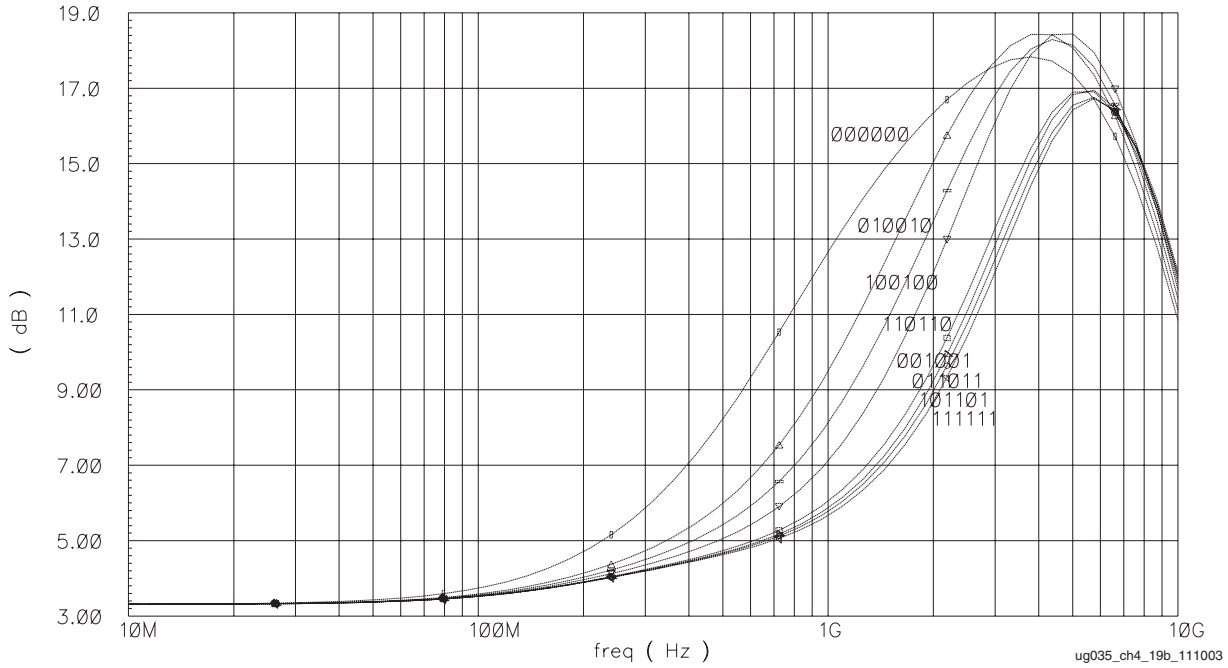


Figure 4-19: **Magnitude (dB) vs. Frequency (Hz) Response for Eight Settings (out of 64) of RXFER[9:4]**

Alternatively, the two groupings can be skewed to give very different frequency responses. For example, setting RXFER[9:7] to the lowest frequency and RXFER[6:4] to the highest frequency results in a flattened boost response.

Figure 4-20 shows such an example where the frequency response for RXFER[9:0] = 0001111111 is compared with RXFER[9:0] = 1101101111 for reference.

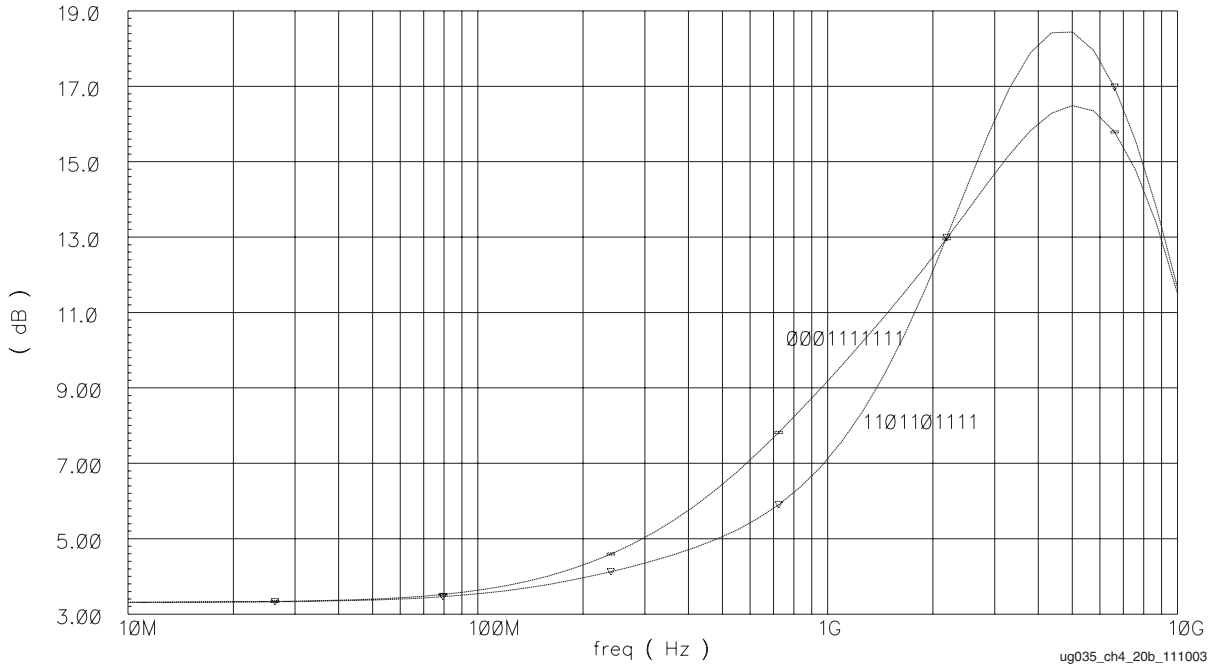


Figure 4-20: **Magnitude (dB) vs. Frequency (Hz) Response for RXFER[9:0] = 0001111111, 110110111**

Simulation TX Emphasis and RX Equalization Settings

Note: This section uses simulation numbers only.

The RocketIO X transceiver incorporates pre-emphasis in the transmitter which preemptively corrects for the PCB loss. In addition, there is an equalizer in the receiver which can post-process the signal to correct for the same loss. Proper transmitter and receiver settings can essentially null the lossy attenuation effects of a PCB completely.

This section provides some good estimates for settings to transfer signals over long, medium, and short backplane lengths, including different lengths of FR4 as well as two backplane connectors. The signal path represents the path from one line/switch card to another through a backplane. Users must be sure to use a connector that reliably passes a signal at operation speed. See Table 4-8.

Table 4-8: **Example Signal Paths**

Trace	Signal Path
Long trace for 6.25 Gb/s and 3.125 Gb/s	package -|- 3" FR4 -|- VHDM-HSD connector -|- 40" FR4 -|- VHDM-HSD connector -|- 3" FR4 -|- package
Medium trace for 6.25 Gb/s and 3.125 Gb/s	package -|- 3" FR4 -|- VHDM-HSD connector -|- 20" FR4 -|- VHDM-HSD connector -|- 3" FR4 -|- package
Short trace for 6.25 Gb/s and 3.125 Gb/s	package -|- 3" FR4 -|- VHDM-HSD connector -|- 7" FR4 -|- VHDM-HSD connector -|- 3" FR4 -|- package
"None" trace for 6.25 Gb/s and 3.125 Gb/s	package -|- package

Many different PCB materials and connectors are available in the market, and using different PCB material or a different connector will definitely change these settings in Table 4-9. Even if the same PCB material and connector are used as described here, the optimum settings might be different depending on the trace width, PCB stack-up (layer thicknesses), and so on. However, these numbers can be a good start. These numbers can change depending upon semiconductor process corners, temperature, supply voltage, etc.

Note: These numbers are based on simulation only, and not based on actual characterization. Therefore, Xilinx emphasizes that **these numbers are only estimates** (i.e., *ballpark* numbers); hence, Xilinx is not responsible if these are not the optimum settings for a particular application.

More detailed descriptions of the transmit pre-emphasis and receive equalization are given in this chapter, and users are encouraged to read those sections to understand the pre-emphasis and equalization features before trying them out.

Table 4-9: **Settings and Results**

Parameter	None 10	None 6	None 3	Short 10	Short 6	Short 3	Med. 10	Med. 6	Med. 3	Long 6	Long 3
TXDOWNLEVEL[3:0]	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
TXEMPHLEVEL[3:0]	0000	0000	0000	0000	0000	0000	0010	0100	0000	0100	0100
RXFER[9:0]	11 1111 0111	11 1111 1111	11 1111 1111	11 1111 0111	11 1111 1001	11 1111 1001	11 1111 0111	11 1111 1001	11 1111 1001	11 0110 0001	11 1110 0001
RXLOOPFILTERC[1:0]	01	01	01	01	01	01	01	01	01	01	01
RXLOOPFILTERR[2:0]	101	100	010	101	100	010	101	100	010	100	010

Notes:
1. The eye described here is the simulated eye after the equalizer.
2. Measured as a pk-pk differential RX voltage.
3. All other control signals are set to their default settings.

PCB Design Requirements

To ensure reliable operation of theRocketIO X transceivers, certain requirements must be met by the designer. This section outlines these requirements governing power filtering networks, high-speed differential signal traces, and reference clocks. Any designs that do not adhere to these requirements are not supported by Xilinx, Inc.

Power Conditioning

Each RocketIO X transceiver has five power supply pins, all of which are sensitive to noise. To operate properly, RocketIO X transceivers require a certain level of noise isolation from surrounding noise sources. For this reason, it is required that both dedicated voltage regulators and passive high-frequency filtering be used to power RocketIO X circuitry.

The power requirements for the various line rates are listed in the Virtex-II Pro / Virtex-II Pro X Data Sheet (DS083, Module 3).

Voltage Regulation

The transceiver voltage regulator circuits must not be shared with any other supplies (including FPGA supplies V_{CCINT}, V_{CCO}, V_{CCAUX}, and V_{REF}). Voltage regulators can be

shared among transceiver power supplies of the same voltage; however, each supply pin must still have its own separate passive filtering network.

One of the following devices are required as voltage regulators:

- Linear Technology LT1963 (LT1963A) 1.5A low-dropout (LDO) (For more information about this device, visit http://www.linear-tech.com/prod/datasheet.html?datasheet=886.)
- Texas Instruments TPS795xx 500mA RF LDO (For more information about this device, visit http://focus.ti.com/docs/prod/productfolder.jhtml?genericPartNumber=TPS79518.)
- Texas Instruments TPS796xx 1A RF LDO (For more information about this device, visit http://focus.ti.com/docs/prod/productfolder.jhtml?genericPartNumber=TPS79618.)
- Texas Instruments TPS786xx 1.5A RF LDO (For more information about this device, visit http://focus.ti.com/docs/prod/productfolder.jhtml?genericPartNumber=TPS78618.)

The LT1963 (LT1963A) regulator must be used with the circuit specified by the manufacturer. Figure 4-21 shows the schematic for the adjustable version of the LT1963 (LT1963A) with values for a 2.5V supply, as would be used for AVCCAUXTX. Alternatively, fixed output voltage devices in the same series may be used, such as the LT1963 (LT1963A)-2.5. If the fixed version is used, SENSE should be connected to OUT. A similar solution must be used for generating 1.5V for AVCCAUXRX and VTTX.

The specified Texas Instruments regulators require a similar solution.

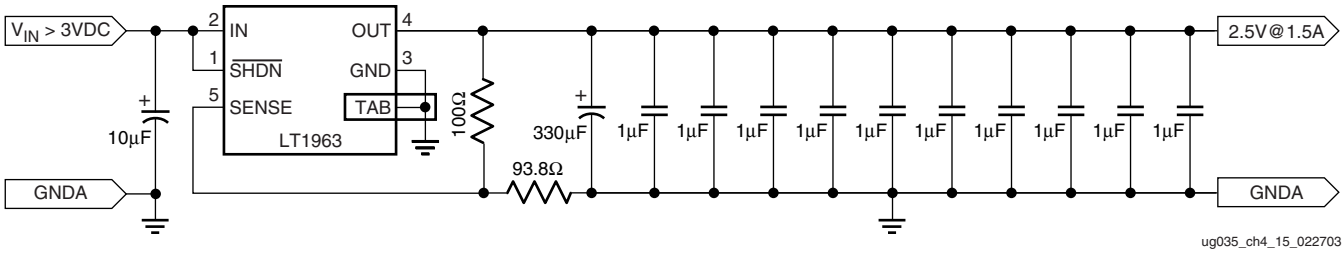


Figure 4-21: **Power Supply Circuit Using LT1963 (LT1963A) Regulator**

Termination voltage can be of any value in the range of 0.25V to 2.5V. In cases where the RocketIO X transceiver is interfacing with a transceiver from another vendor, termination voltage can be dictated by the specifications of the other transceiver. In cases where the RocketIO X transceiver is interfacing with another RocketIO X transceiver, a 1.5V termination voltage is recommended. The LT1963 (LT1963A) circuit's output capacitors (330 µF and 1 µF) can be placed anywhere on the board, preferably close to the output of the LT1963 device.

All characterization work was done using the LT1963 device. If another part is used instead of the LT1963, it must meet the following requirements:

- Must be a linear regulator.
- Must have output noise no greater than 40 µV RMS from 10 Hz to 100 kHz.
- Must regulate to within 2% of the nominal output voltage (±50 mV).

Passive Filtering

To achieve the necessary isolation from high-frequency power supply noise, passive filter networks are required on the power supply pins. The topology of these ferrite bead circuits is given in Figure 4-22.

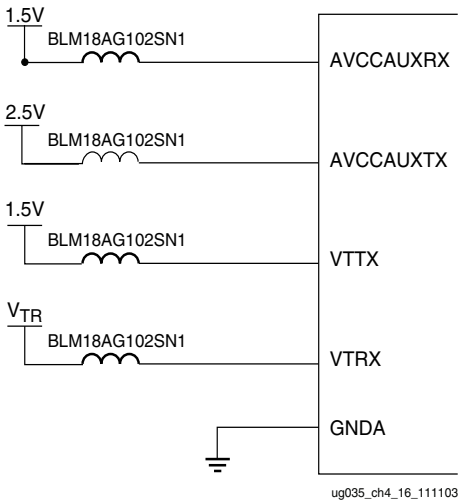


Figure 4-22: **Power Filtering Network for One Transceiver**

Each transceiver power pin requires one ferrite bead. Like the Virtex-II Pro MGT flip chip devices, all Virtex-II Pro X MGTs contain power-filtering capacitors in the package to reduce component count on the PCB, as well as improve the effectiveness of these capacitors. The value of this internal capacitor is 0.22 µF. The ferrite bead is the Murata BLM 18AG102SN1 and must be used as shown in Figure 4-22. These components may not be shared among multiple RocketIO power supply pins under any circumstances.

Figure 4-23 shows an example with eight transceivers total–four on the top edge (rows A/B) and four on the bottom edge (rows AE/AF). Figure 4-23 shows the bottom PCB layer, with lands for the ferrite beads of all supplies: AVCCAUXTX, AVCCAUXRX, VTTX, and

VTRX. The ferrite beads are mounted at the 16 "L[n]" locations (highlighted by the colored circles).

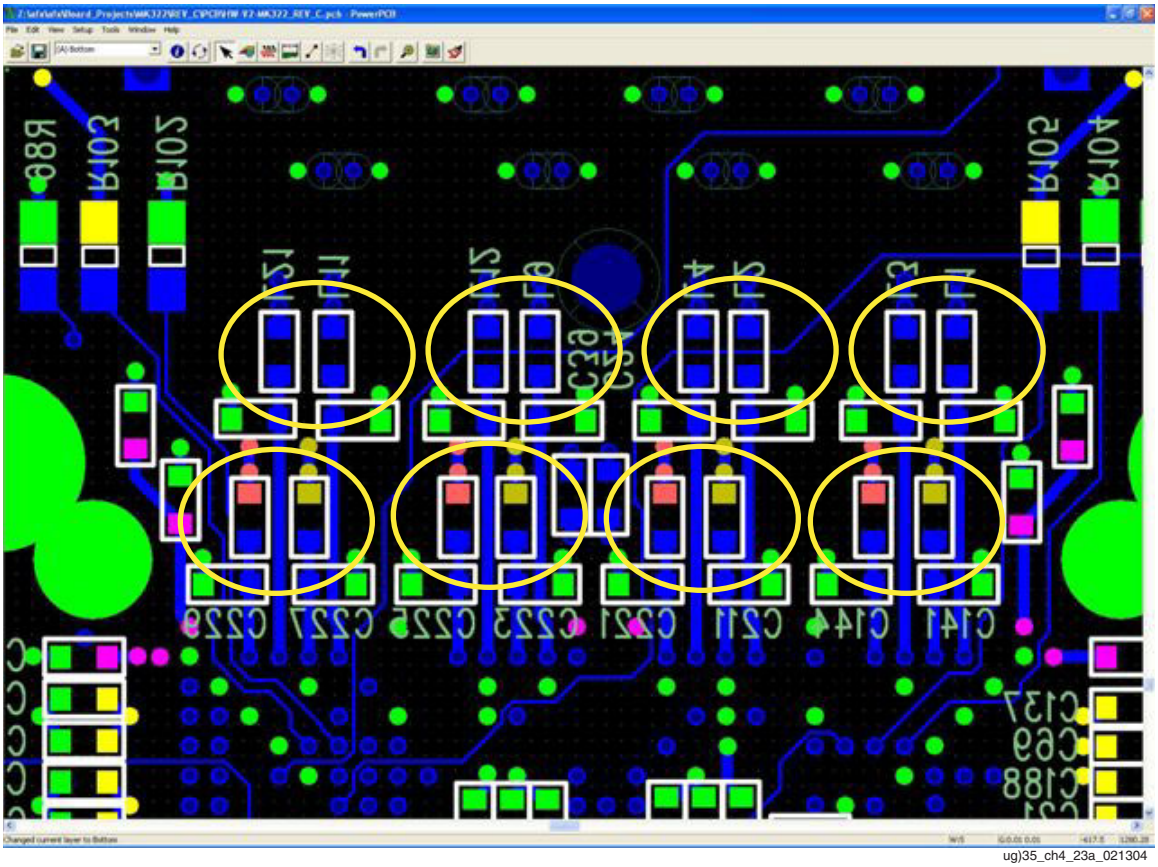


Figure 4-23: **Example Power Filtering PCB Layout for Four MGTs (In Device With Internal Capacitors), Bottom Layer**

All AVCCAUXTX and AVCCAUXRX pins in a Virtex-II Pro X device must be connected to 2.5V/1.5V regardless of whether or not they are used. See "Powering the RocketIO X Transceivers," page 111 and "POWERDOWN Port," page 111 for details.

High-Speed Serial Trace Design

Routing Serial Traces

All RocketIO X transceiver I/Os are placed on the periphery of the BGA package to facilitate routing and inspection (since JTAG is not available on serial I/O pins). RocketIO X transceivers have a 50Ω output/input impedance. Controlled impedance traces should be used to connect the RocketIO X transceiver to other compatible transceivers.

When routing a differential pair, the complementary traces must be matched in length to as close a tolerance as is feasible. Length mismatches produce common mode noise and radiation. Severe length mismatches produce jitter and unpredictable timing problems at the receiver. Matching the differential traces to within 50 mils (1.27 mm) produces a robust design. Since signals propagate in FR4 PCB traces at approximately 180 ps per inch, a difference of 50 mils produces a timing skew of roughly 9 ps. Use SI CAD tools to confirm these assumptions on specific board designs.

All signal traces must have an intact reference plane beneath them. Stripline and microstrip geometries may be used. The reference plane should extend no less than five trace widths to either side of the trace to ensure predictable transmission line behavior.

Routing of a differential pair is optimally done in a point-to-point fashion, ideally remaining on the same PCB routing layer. As vias represent an impedance discontinuity, layer-to-layer changes should be avoided wherever possible. It is acceptable to traverse the PCB stackup to reach the transmitter and receiver package pins. If serial traces must change layers, care must be taken to ensure an intact current return path. For this reason, routing of high-speed serial traces should be on signal layers that share a reference plane. If the signal layers do not share a reference plane, a capacitor of value 0.01 µF should be connected across the two reference layers close to the vias where the signals change layers. If both of the reference layers are DC coupled (if they are both ground), they can be connected with vias close to where the signals change layers.

To control crosstalk, serial differential traces should be spaced at least five trace separation widths from all other PCB routes, including other serial pairs. A larger spacing is required if other PCB routes carry noisy signals, such as TTL and other similarly noisy standards.

The RocketIO X transceiver is designed to reliably transmit data at 3.125 and 6.25 Gb/s over more than 46 inches of FR4 and two high-bandwidth connectors.

Differential Trace Design

The characteristic impedance of a pair of differential traces depends not only on the individual trace dimensions, but also on the spacing between them. The RocketIO X transceivers require a 100Ω differential trace impedance. To achieve this differential impedance requirement, the characteristic impedance of each individual trace must be slightly higher than half of the target differential impedance. A field solver should be used to determine the exact trace geometry suited to the specific application (Figure 4-24). This task should not be left up to the PCB vendor.

Differential impedance of traces on the finished PCB should be verified with Time Domain Reflectometry (TDR) measurements.

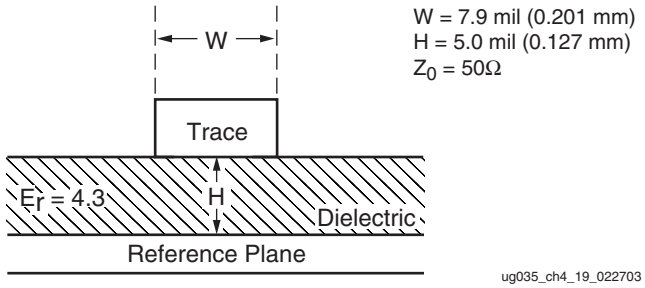


Figure 4-24: **Single-Ended Trace Geometry**

Tight coupling of differential traces is recommended. Tightly coupled traces (as opposed to loosely coupled) maintain a very close proximity to one another along their full length. Since the differential impedance of tightly coupled traces depends heavily on their proximity to each other, it is imperative that they maintain constant spacing along their full length, without deviation. If it is necessary to separate the traces in order to route through a pin field or other PCB obstacle, it can be helpful to modify the trace geometry in the vicinity of the obstacle to correct for the impedance discontinuity (increase the individual trace width where trace separation occurs).

Figure 4-25 and Figure 4-26 show examples of PCB geometries that result in 100Ω differential impedance.

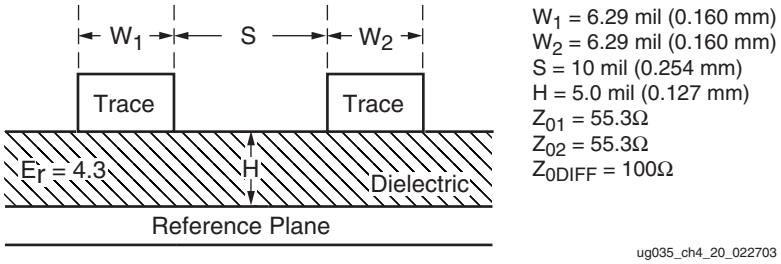


Figure 4-25: **Microstrip Edge-Coupled Differential Pair**

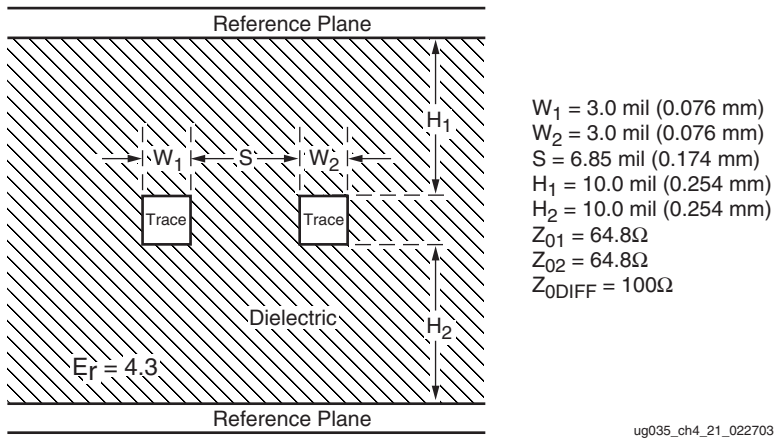


Figure 4-26: **Stripline Edge-Coupled Differential Pair**

Note: For more information about serial backplane traces, see Appendix E, "Serial Backplane System Design."

Termination

The RocketIO X transceiver implements on-chip 50Ω termination in both the transmitter (TXP/TXN) and receiver (RXP/RXN). The output driver and termination are powered by VTTX at 1.5V. This configuration uses a CML approach with 50Ω to TXP and TXN as shown in Figure 4-27.

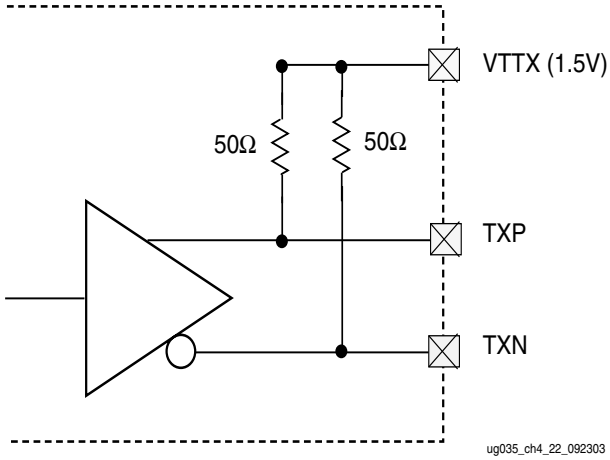


Figure 4-27: **Transmit Termination**

The receiver termination supply (VTRX) is the center tap of differential termination to RXP and RXN as shown in Figure 4-28. This supports multiple termination styles, including high-side, low-side, and differential (floating or active). This configuration supports receiver termination compatible to Virtex-II Pro devices, using a CML (high-side) termination to an active supply of 1.8V – 2.5V. For DC coupling of two Virtex-II Pro X devices, a 1.5V CML termination for VTRX is recommended.

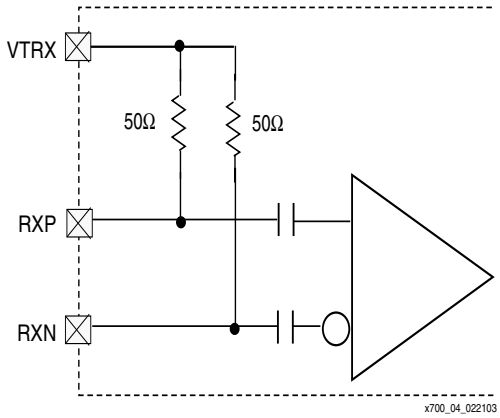


Figure 4-28: **Receive Termination**

AC and DC Coupling

AC coupling (use of DC blocking capacitors in the signal path) should be used in cases where transceiver differential voltages are compatible, but common mode voltages are not. Some designs require AC coupling to accommodate hot plug-in, and/or differing power supply voltages at different transceivers. This is illustrated in Figure 4-29. The RocketIO X transceiver has on-chip AC coupling caps in the receiver after the receive termination, thus supporting a wide common mode input range (0.25V – 2.5V). For common mode voltages outside of this range, external AC coupling should be used. The on-chip AC coupling supports coded or scrambled data with run lengths of up to 72 bits for the entire range of data rates (2.488 – 6.25 Gb/s). This AC coupling provides a high-pass filter with a corner frequency of 100 kHz.

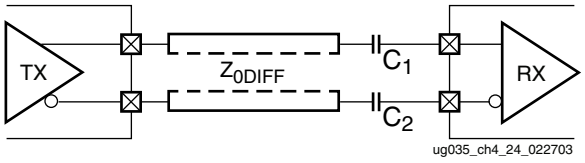


Figure 4-29: **AC-Coupled Serial Link**

Capacitors of value 0.01 µF in a 0402 package are suitable for AC coupling up to 6.25 Gb/s when 8B/10B or 64B/66B encoding is used. Different data rates and different encoding schemes may require a different value.

DC coupling (direct connection) is preferable in cases where RocketIO X transceivers are interfaced with other RocketIO X transceivers. Passive components are not required when DC coupling is used. This is illustrated in Figure 4-30.

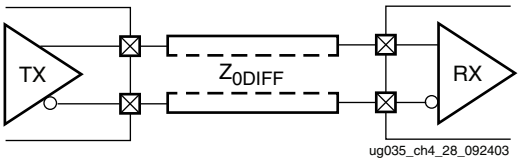


Figure 4-30: **DC-Coupled Serial Link**

Other Important Design Notes

Powering the RocketIO X Transceivers

IMPORTANT! *All* RocketIO X transceivers in the FPGA, whether instantiated in the design or not, must be connected to power and ground. Unused transceivers can be powered by any 2.5V/1.5V source, and passive filtering is not required. Refer to "Power Conditioning," page 104 for detailed information on powering instantiated RocketIO X transceivers.

Maximum power consumption (PMA + PCS) per MGT is about 600 mW at 6.25 Gb/s.

POWERDOWN Port

POWERDOWN is a single-bit primitive port (see Table 3-1, page 70) that allows shutting off the transceiver, in case it is not needed for the design or will not be transmitting or receiving for a long period of time. When POWERDOWN is asserted, the transceiver does not use any power. The clocks are disabled and do not propagate through the core. The 3-state TXP and TXN pins are set to high-Z, while the outputs to the fabric are frozen but *not* set to high-Z.

Any given transceiver that is *not* instantiated in the design is automatically set to the POWERDOWN state by the Xilinx ISE development software and consumes no power. An instantiated transceiver, however, consumes some power, even if it is not engaged in transmitting or receiving. Therefore, when a transceiver is not to be used for an extended period of time, the POWERDOWN port should be asserted High to reduce overall power consumption by the Virtex-II Pro X FPGA. Deasserting the POWERDOWN port restores the transceiver to normal functional status.

Reference Clock

A high degree of accuracy is required from the reference clock. For this reason, it is required that an EPSON EG2121CA 2.5V oscillator be used. (Visit the Epson Electronics America website for detailed information about this device.) The power supply circuit specified by the manufacturer must be used, and the circuit in Figure 4-31 must be used to interface the LVPECL outputs of the oscillator with the inputs of the transceiver reference clock.

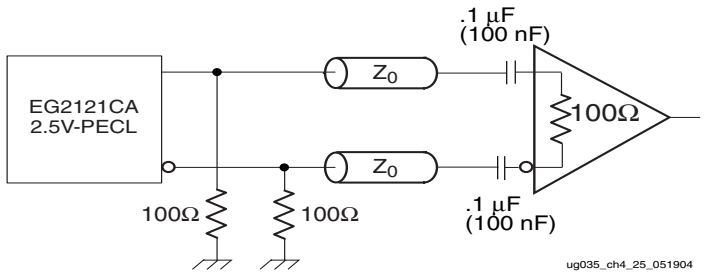


Figure 4-31: **Reference Clock Oscillator Interface up to 400 MHz**

The EG2121CA can be used for frequencies up to 400 MHz. For applications beyond this frequency (e.g., 622.08 MHz, 644.53125 MHz), the EG2121CA has to be replaced with EV-2101CA, which is a Voltage Controlled Saw Oscillator (VCSO). Note that the VCSO requires a control voltage in addition to the supply voltage. Also note that the supply voltage is 3.3V and not 2.5V (as in the EG2121CA). See Figure 4-32.

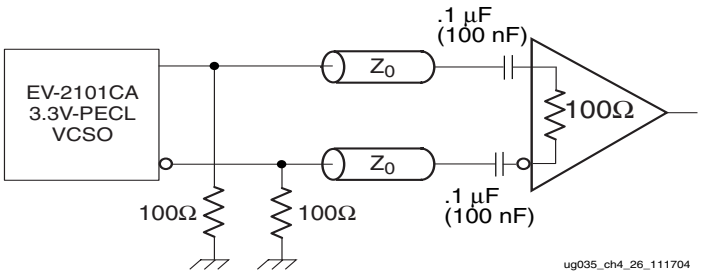


Figure 4-32: **Reference Clock Oscillator Interface above 400 MHz**

Chapter 5

Simulation and Implementation

PMA Initialization

When the PMAINIT signal is asserted High, the PCS resets the PMA and the PCS, and loads coefficients from the PMA_SPEED attribute into the PMA. The PCS is held in reset during this time. Figure 5-1 shows the steps that take place once the PMAINIT signal has been asserted and deasserted.

If the PMAINIT signal is asserted and deasserted, the PMARXLOCK signal goes Low and stays Low until configuration is complete and the receive PLL has regained lock. Note that the PMARXLOCK signal can assert High and Low while the receive PLL is locking.

The addresses 0-14 in the PMA are loaded sequentially first, with address 15 loaded last. Address 15 is loaded four times, in the order of the power-up sequence defined by the PMA. The attribute PMA_PWR_MASK defines which sections of the PMA are powered up. By default, the entire PMA powers "Model Considerations" up, including the line driver.

After the initialization sequence is completed, the initialization function relinquishes control of the PMA bus to the fabric. At this time, the user can load custom values into the PMA configuration registers.

If the user would not like to depend on PMARXLOCK to indicate when the function is completed, a timer can be used. The period of the clock used is the reference clock divided by two. This clock is connected internally. Refer to for start-up times.

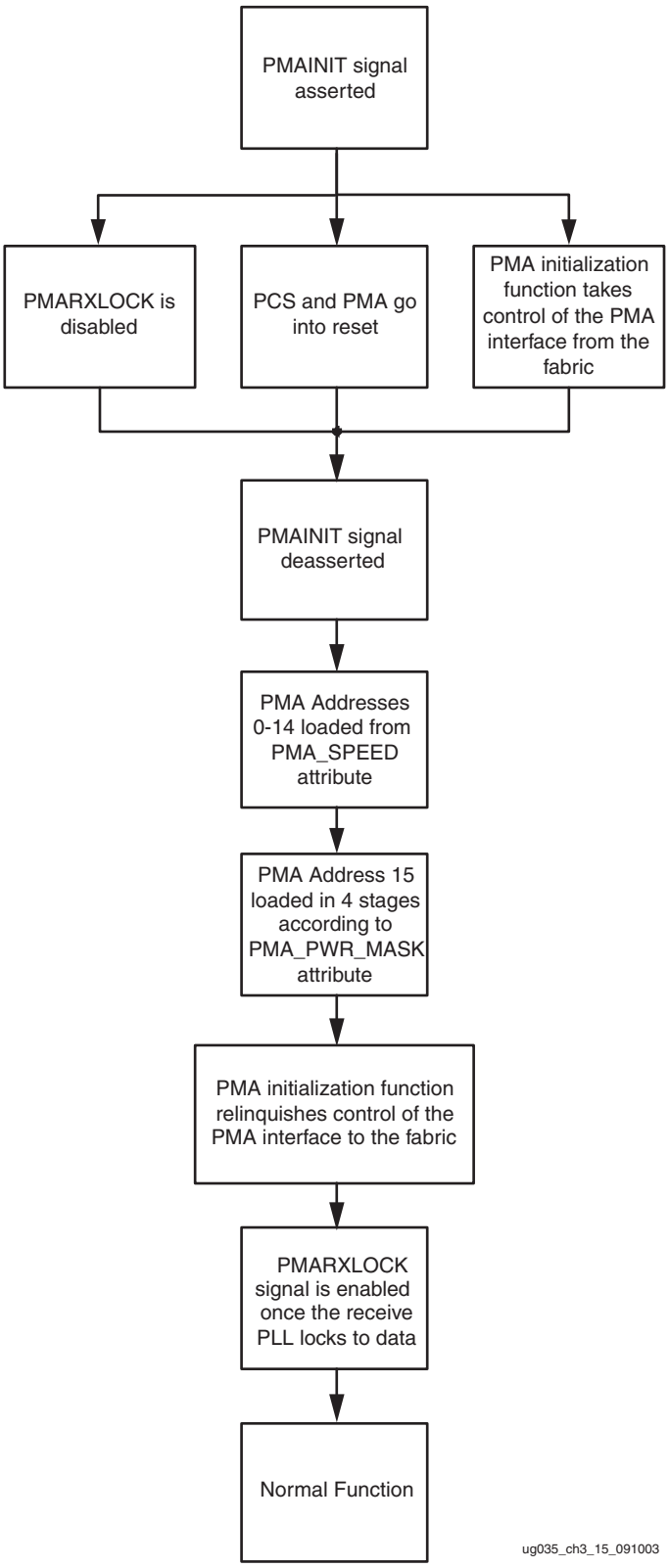


Figure 5-1: **PMA Initialization**

Model Considerations

When running with the functional SWIFT model, users need to note that the model correctly mimics the startup times required for the PMA. However, since the startup time is in most cases greater than 10 µs, it is recommended that the complete model be used to ensure correct system operation.

There are two components that delay the startup of the PMA. The first is the power-up sequence within the PCS, which brings up PMA bias circuits in an orderly and predictable manner. The second is the PMA itself, which has a long power-up cycle. For more information on PMA power up, refer to Appendix C, "PMA Attribute Programming Bus."

Given a reference clock with period `t`, the equation for computing the startup time is:

```
Time to startup = 10000*t +5000
```

where the result and `t` are in nanoseconds.

When working with the model, observe the following suggestions:

1. After applying the reference clock and deasserting resets, TXOUTCLK should be defined and oscillating. If TXOUTCLK is undefined, most likely the circuit is within the instability period.
2. Note that in 64/66 protocols some periods for the reference clock results in a non-integer multiple between the two clock domains on either side of the FIFOs. The ratio must be exactly 64/66, or data is "lost" in the simulation due to rounding error. An example of a good period for the reference clock in mode 6_32 would be 1.6 ns, which results in a fabric period of 3.3 ns (the average is really three pulses of 3.2 ns followed by one of 3.6 ns) and a PMA interface period of 3.2 ns. This ratio of 3.3/3.2 is exactly proportional to 66/64, which meets the requirement.
3. If you are having problems passing data, try to put the model into loopback. Make sure that TXOUTCLK and RXRECCLK are not undefined. If RXRECCLK is undefined, either the clock data recovery circuit has not locked, or the input data is undefined.
4. Many signals in the design must be configured properly for the design to work. Be sure to double check all resets to make sure they are in the correct state.

Simulation Models

SmartModels

SmartModels are encrypted versions of the actual HDL code. These models allow the user to simulate the actual functionality of the design without having access to the code itself. A simulator with SmartModel capability is required to use SmartModels. See **Solution Record 15501** for information on how to install the SmartModels.

HSPICE

HSPICE is an analog design model that allows simulation of the RX and TX high-speed transceiver. To obtain these HSPICE models, go to the SPICE Suite Access web page at: **http://support.xilinx.com/support/software/spice/spice-request.htm**.

MGT Package Pins

The MGTs make up a hard core placed in the FPGA fabric; all package pins for the MGTs are dedicated on the Virtex-II Pro X device. This is shown in the package pin diagrams in the Virtex-II Pro data sheet. When creating a design, LOC constraints must be used to implement a specific MGT on the die. This LOC constraint also determines which package pins are used. Table 5-1 and Table 5-2 show the correlation between the LOC grid and the package pins themselves. The pin numbers are the TXNPAD, TXPPAD, RXPPAD, and RXNPAD, respectively. The power pins are adjacent to these pins in the package pin diagrams of the handbook.

Table 5-1: **LOC Grid and Package Pins Correlation for FF896Package**

LOC Constraints	FF896
	XC2VPX20
GT_X0_Y0	AK27, AK26, AK25, AK24
GT_X0_Y1	A27, A26, A25, A24
GT_X1_Y0	AK20, AK19, AK18, AK17
GT_X1_Y1	A20, A19, A18, A17
GT_X2_Y0	AK14, AK13, AK12, AK11
GT_X2_Y1	A14, A13, A12, A11
GT_X3_Y0	AK7, AK6, AK5, AK4
GT_X3_Y1	A7, A6, A5, A4

Table 5-2: **LOC Grid and Package Pins Correlation for FF1704 Packages**

LOC Constraints	FF1704
	XC2VPX70
GT_X0_Y0	BB41, BB40, BB39, BB38
GT_X0_Y1	A41, A40, A39, A38
GT_X1_Y0	BB37, BB36, BB35, BB34
GT_X1_Y1	A37, A36, A35, A34
GT_X2_Y0	BB33, BB32, BB31, BB30
GT_X2_Y1	A33, A32, A31, A30
GT_X3_Y0	BB29, BB28, BB27, BB26
GT_X3_Y1	A29, A28, A27, A26
GT_X4_Y0	BB25, BB24, BB23, BB22
GT_X4_Y1	A25, A24, A23, A22
GT_X5_Y0	BB21, BB20, BB19, BB18
GT_X5_Y1	A21, A20, A19, A18

Table 5-2: **LOC Grid and Package Pins Correlation for FF1704 Packages**

LOC Constraints	FF1704
	XC2VPX70
GT_X6_Y0	BB17, BB16, BB15, BB14
GT_X6_Y1	A17, A16, A15, A14
GT_X7_Y0	BB13, BB12, BB11, BB10
GT_X7_Y1	A13, A12, A11, A10
GT_X8_Y0	BB9, BB8, BB7, BB6
GT_X8_Y1	A9, A8, A7, A6
GT_X9_Y0	BB5, BB4, BB3, BB2
GT_X9_Y1	A5, A4, A3, A2

Diagnostic Signals

Often a diagnostic check is needed upon power-up. RocketI/O X transceivers have several inputs and outputs to run these checks.

Loopback

LOOPBACK allows the user to send the data that is being transmitted directly to the receiver of the transceiver. Table 5-3 shows the four modes for loopback.

Table 5-3: **LOOPBACK Modes**

LOOPBACK[1:0]	Description
00	Normal operation. Not in loopback. The transmitted data is sent out from the differential transmit ports (TXN, TXP) and is sent to another transceiver without being sent to its own receiver logic. During normal operation, the LOOPBACK should be set to 00.
01	Internal parallel loopback. This mode is used to check the PCS function. The PMA is not included.
10	Post-driver serial loopback (after the TX output buffer). This mode is used to check that the entire transceiver is working properly. This includes testing both the PCS and PMA functions. This emulates what another transceiver would receive as data from this specific transceiver design
11	Pre-driver serial loopback (before the TX output buffer). Same as above, but not including the TX output buffer. The output buffer can be disabled in this loopback mode.

Parallel Loopback

In parallel loopback mode, data is looped at the PCS/PMA interface and clocked via the synthesized clock from the RocketIO X transmitter. A stable REFCLK and valid PMA initialization are required.

In parallel loopback mode, TXINHIBIT can not be used to suppress the transmit data from being sent onto the TXN/TXP pins, because TXINHIBIT also inhibits the data being looped back to the receiver. The pre-driver serial loopback mode should be used for applications that do not require any data to be driven onto the transmit pins.

Note: TXPOLARITY/RXPOLARITY have no effect on the parallel loopback operation.

Post/Pre-Driver Serial Loopback

Since the serial-loopback mode test-data paths are internal to the silicon, certain equalization default settings optimized for a complete system can cause improper operation when using serial loopback. The modes that need different settings for serial loopback are shown in Table 5-4, along with the recommended settings for specific addresses of the PMA Attribute Bus (see Appendix C, "PMA Attribute Programming Bus").

Pre-driver serial loopback does not switch the TXN and TXP. It also does not require TXN and TXP to be terminated. However, post-driver serial loopback does require proper termination of TXN/TXP for proper operation

Table 5-4: **Recommended Settings for Serial Loopback**

Mode	PMA Address	Recommended Value
20_XX (6.25G) 30_XX (2/48G) 28_XX (2.56G) 25_XX (3.125G) 13_XX (10G)	0 x OC	OD
	0 x OD	FC

XILINX®

Appendix A

RocketIO X Transceiver Timing Model

This appendix explains the timing parameters associated with the RocketIO X™ transceiver core. It is intended to be used in conjunction with Module 3 of the Virtex-II Pro data sheet (DS083) and the Timing Analyzer (TRCE) report from Xilinx software. For specific timing parameter values, refer to the data sheet.

There are many signals entering and exiting the RocketIO X core. (Refer to Figure A-2.) The model presented in this section treats the RocketIO X core as a "black box." Propagation delays internal to the RocketIO X core logic are ignored. Signals are characterized with setup and hold times for inputs, and with clock to valid output times for outputs.

There are seven clocks associated with the RocketIO X core, but only three of these clocks—RXUSRCL, RXUSRCLK2, and TXUSRCLK2—have I/Os that are synchronous to them. The following table gives a brief description of all of these clocks. For an in-depth discussion of clocking the RocketIO X core, refer to Chapter 2, "Digital Design Considerations."

Table A-1: **RocketIO X Clock Descriptions**

CLOCK SIGNAL	DESCRIPTION
BREFCLKPIN & BREFCLKNIN	Main reference clock for RocketIO X transceiver.
REFCLK	High-quality reference clock driving transmission (reading TX FIFO, and multiplied for parallel/serial conversion) and clock recovery. REFCLK frequency is accurate to ± 100 ppm. This clock originates off the device, is routed through fabric interconnect, and is selected by the REFCLKSEL.
TXOUTCLK	Synthesized Clock from RocketIO X transmitter. This clock can be scaled (e.g., for 64B/66B) relative to BREFCLK, depending upon the specific operating mode of the transmitter.
TXUSRCLK	Clock used for writing the TX buffer. Frequency-locked to REFCLK.
TXUSRCLK2	Clocks transmission data and status and reconfiguration data between the transceiver and the FPGA fabric. Relationship between TXUSRCLK2 and TXUSRCLK depends on width of transmission data path.

Table A-1: **RocketIO X Clock Descriptions**

CLOCK SIGNAL	DESCRIPTION
RXRECCLK	Recovered Clock from RocketIO X receiver, locked to incoming data stream. This clock can be scaled (e.g., 64/66) relative to incoming data rate, depending upon the specific operating mode of the receiver.
RXUSRCLK	Clock used for reading the RX elastic buffer. Clocks CHBONDI and CHBONO into and out of the transceiver. Typically the same as TXUSRCLK.
RXUSRCLK2	Clocks receiver data and status between the transceiver and the FPGA fabric. Typically the same as TXUSRCLK2. Relationship between RXUSRCLK2 and RXUSRCLK depends on width of receiver data path.

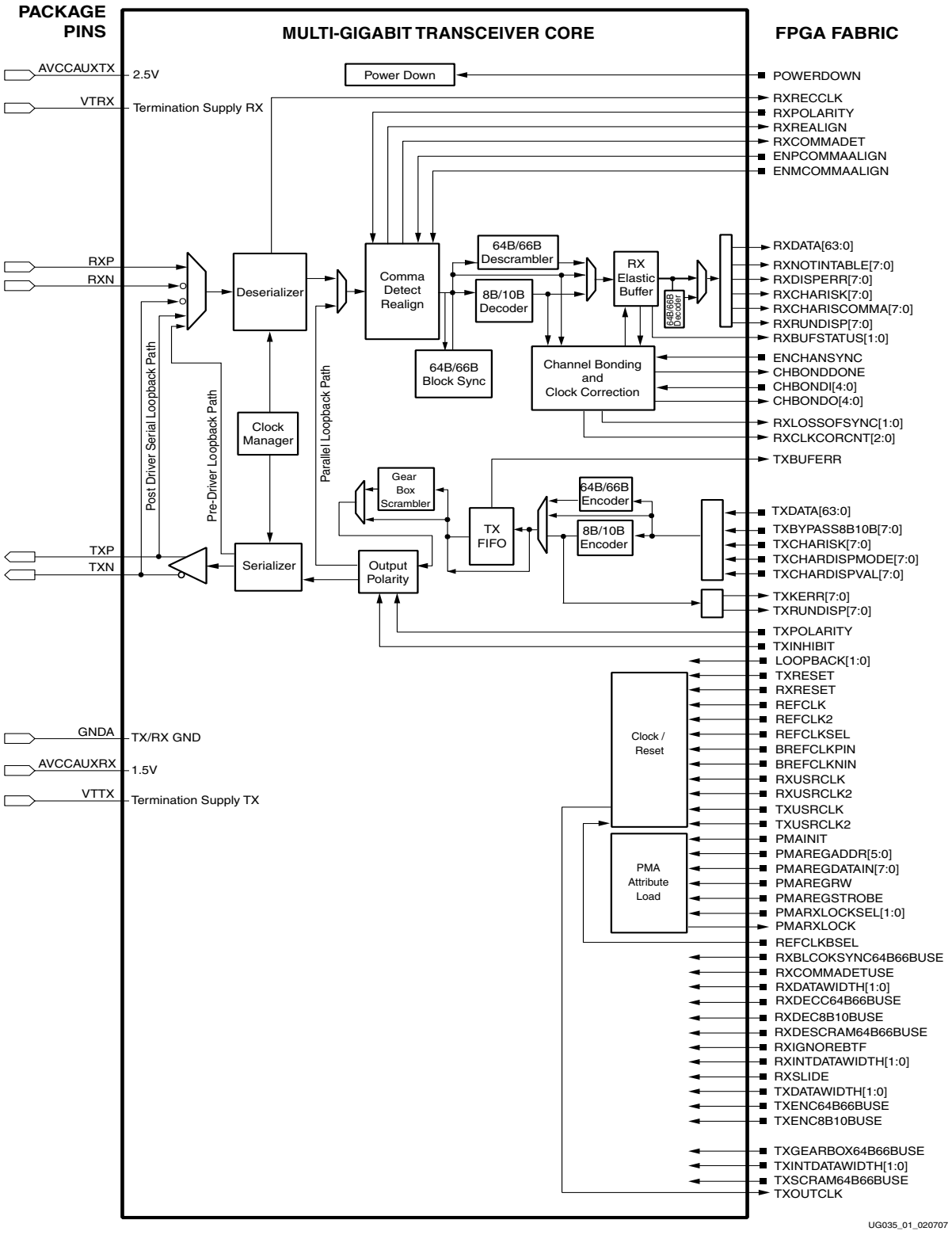


Figure A-1: **RocketIO X Transceiver Block Diagram**

Timing Parameters

Parameter designations are constructed to reflect the functions they perform, as well as the I/O signals to which they are synchronous. The following subsections explain the meaning of each of the basic timing parameter designations used in the tables.

Input Setup/Hold Times Relative to Clock

Basic Format:

ParameterName_SIGNAL

where

ParameterName = T with subscript string defining the timing relationship

SIGNAL = name of RocketIO X signal synchronous to the clock

ParameterName Format:

T_{GxCK} = Setup time before clock edge

T_{GCKx} = Hold time after clock edge

where

x = C (Control inputs)
D (Data inputs)

Setup/Hold Time (Examples):

T_{GCCK}_RRST/T_{GCKC}_PLB Setup/hold times of RX Reset input relative to rising edge of RXUSRCLK2

T_{GDCK}_TDAT/T_{GCKD}_TDAT Setup/hold times of TX Data inputs relative to rising edge of TXUSRCLK2

Clock to Output Delays

Basic Format:

ParameterName_SIGNAL

where

ParameterName = T with subscript string defining the timing relationship

SIGNAL = name of RocketIO X signal synchronous to the clock

ParameterName Format:

T_{GCKx} = Delay time from clock edge to output

where

x = CO (Control outputs)
DO (Data outputs)
ST (Status outputs)

Output Delay Time (Examples):

T_{GCKCO}_CHBO Rising edge of RXUSRCLK to Channel Bond outputs

T_{GCKDO}_RDAT Rising edge of RXUSRCLK2 to RX Data outputs

T_{GCKST}_TBERR Rising edge of TXUSRCLK2 to TX Buffer Err output

Clock Pulse Width

ParameterName Format:

T_{xPWH} = Minimum pulse width, High state

T_{xPWL} = Minimum pulse width, Low state

where

x = REF (REFCLK)
TX (TXUSRCLK)
TX2 (TXUSRCLK2)
RX (RXUSRCLK)
RX2 (RXUSRCLK2)

Pulse Width (Examples):

T_{TX2PWL} Minimum pulse width, TX2 clock, Low state
T_{REFPWH} Minimum pulse width, Reference clock, High state

Timing Diagram and Timing Parameter Tables

A timing diagram (Figure A-2) illustrates the timing relationships.

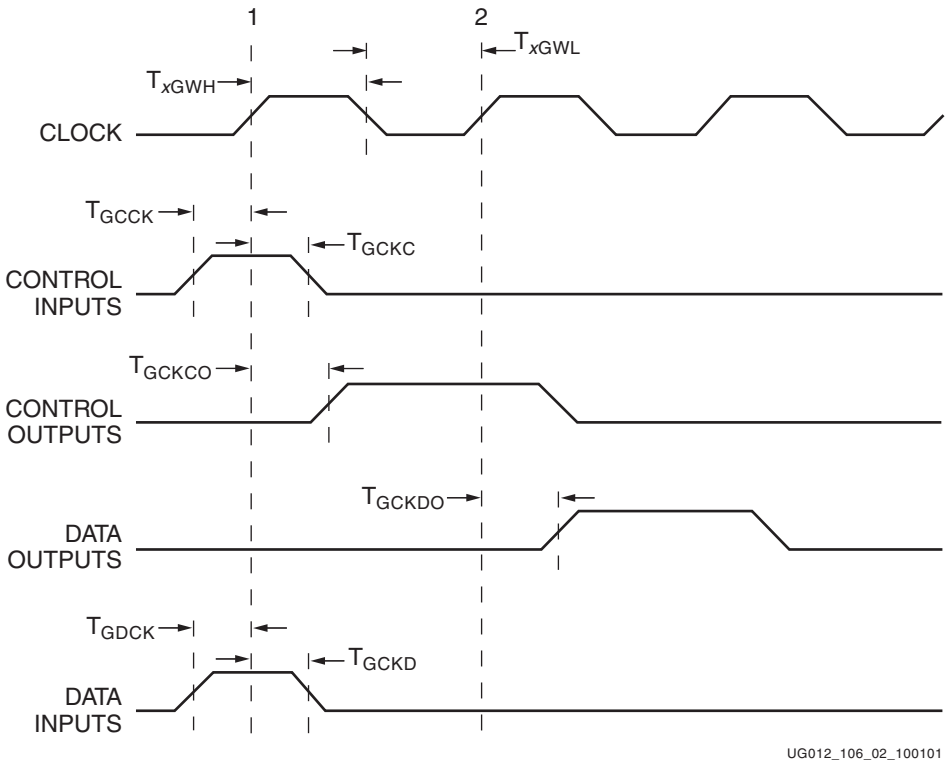


Figure A-2: **RocketIO X Transceiver Timing Relative to Clock Edge**

The following four tables list the timing parameters as reported by the implementation tools relative to the clocks given in Table A-1, page 119, along with the RocketIO X signals that are synchronous to each clock. (No signals are synchronous to REFCLK or TXUSRCLK.)

- Table A-2, "Parameters Relative to RX User Clock (RXUSRCLK)," page 124
- Table A-3, "Parameters Relative to RX User Clock2 (RXUSRCLK2)," page 124
- Table A-4, "Parameters Relative to TX User Clock2 (TXUSRCLK2)," page 125
- Table A-5, "PMA Clock Parameters," page 126
- Table A-6, "Miscellaneous Clock Parameters," page 126

Table A-2: **Parameters Relative to RX User Clock (RXUSRCLK)**

Parameter	Function	Signals
Setup/Hold:		
$T_{GCCK_}$CHBI/$T_{GCKC_}$CHBI	Control inputs	CHBONDI[4:0]
Clock to Out:		
$T_{GCKCO_}$CHBO	Control outputs	CHBONDO[4:0]
Clock:		
T_{RXPWH}	Clock pulse width, High state	RXUSRCLK
T_{RXPWL}	Clock pulse width, Low state	RXUSRCLK

Table A-3: **Parameters Relative to RX User Clock2 (RXUSRCLK2)**

Parameter	Function	Signals
Setup/Hold:		
$T_{GCCK_}$RRST/$T_{GCKC_}$RRST	Control input	RXRESET
$T_{GCCK_}$RPOL/$T_{GCKC_}$RPOL	Control input	RXPOLARITY
$T_{GCCK_}$ECSY/$T_{GCKC_}$ECSY	Control input	ENCHANSYNC
$T_{GCCK_}$BLKSNC/$T_{GCKC_}$BLKSNC	Control input	RXBLOCKSYNC64B66EUSE
$T_{GCCK_}$CMDT/$T_{GCCK_}$CMDT	Control inputs	RXCOMMADETUSE
$T_{GCCK_}$IBTF/$T_{GCCK_}$IBTF	Control inputs	RXIGNOREBTF
$T_{GCCK_}$RDATW/$T_{GCCK_}$RDATW	Control inputs	RXDATAWIDTH[1:0]
$T_{GCCK_}$RDEC/$T_{GCCK_}$RDEC	Control inputs	RXDEC64B66BUSE RXDEC8B/10BUSE
$T_{GCCK_}$RDES/$T_{GCCK_}$RDES	Control inputs	RXDESCRAM64B66BUSE
$T_{GCCK_}$RIDATW/$T_{GCCK_}$RIDATW	Control inputs	RXINTDATAWIDTH[1:0]
$T_{GCCK_}$RXSLIDE/$T_{GCCK_}$RXSLIDE	Control inputs	RXSLIDE
Clock to Out:		
$T_{GCKST_}$PLCK	Status output	PMARXLOCK
$T_{GCKST_}$RNIT	Status outputs	RXNOTINTABLE[7:0]
$T_{GCKST_}$RDERR	Status outputs	RXDISPERR[7:0]
$T_{GCKST_}$RCMCH	Status outputs	RXCHARISCOMMA[7:0]
$T_{GCKST_}$ALIGN	Status output	RXREALIGN
$T_{GCKST_}$CMDT	Status output	RXCOMMADET
$T_{GCKST_}$RLOS	Status outputs	RXLOSSOFSYNC[1:0]
$T_{GCKST_}$RCCCNT	Status outputs	RXCLKCORCNT[2:0]
$T_{GCKST_}$RBSTA	Status outputs	RXBUFSTATUS[1:0]
$T_{GCKST_}$CHBD	Status output	CHBONDDONE
$T_{GCKST_}$RKCH	Status outputs	RXCHARISK[7:0]
$T_{GCKST_}$RRDIS	Status outputs	RXRUNDISP[7:0]
$T_{GCKDO_}$RDAT	Data outputs	RXDATA[63:0]

Table A-3: **Parameters Relative to RX User Clock2 (RXUSRCLK2) *(Continued)***

Parameter	Function	Signals
Clock:		
T_{RX2PWH}	Clock pulse width, High state	RXUSRCLK2
T_{RX2PWL}	Clock pulse width, Low state	RXUSRCLK2

Table A-4: **Parameters Relative to TX User Clock2 (TXUSRCLK2)**

Parameter	Function	Signals
Setup/Hold:		
$T_{GCCK_}$TBYP/$T_{GCKC_}$TBYP	Control inputs	TXBYPASS8B10B[7:0]
$T_{GCCK_}$TPOL/$T_{GCKC_}$TPOL	Control inputs	TXPOLARITY
$T_{GCCK_}$TINH/$T_{GCKC_}$TINH	Control inputs	TXINHIBIT
$T_{GCCK_}$LBK/$T_{GCKC_}$LBK	Control inputs	LOOPBACK[1:0]
$T_{GCCK_}$TRST/$T_{GCKC_}$TRST	Control inputs	TXRESET
$T_{GCCK_}$TKCH/$T_{GCKC_}$TKCH	Control inputs	TXCHARISK[7:0]
$T_{GCCK_}$TCDM/$T_{GCKC_}$TCDM	Control inputs	TXCHARDISPMODE[7:0]
$T_{GCCK_}$TCDV/$T_{GCKC_}$TCDV	Control inputs	TXCHARDISPVAL[7:0]
$T_{GCCK_}$TDATW/$T_{GCCK_}$TDATW	Control inputs	TXDATAWIDTH[1:0]
$T_{GCCK_}$TENC/$T_{GCCK_}$TENC	Control inputs	TXENC64B66BUSE TXENC8B10BUSE
$T_{GCCK_}$TIDATW/$T_{GCCK_}$TIDATW	Control inputs	TXINTDATAWIDTH[1:0]
$T_{GCCK_}$TXGEAR/$T_{GCCK_}$TXGEAR	Control inputs	TXGEARBOX64B66BUSE
$T_{GCCK_}$TXSCBL/$T_{GCCK_}$TXSCBL	Control inputs	TXSCRAM64B66BUSE
$T_{GCCK_}$RFCKSL/$T_{GCCK_}$RFCKSL	Control inputs	REFCLKSEL REFCLKBSEL
$T_{GDCK_}$TDAT/$T_{GCKD_}$TDAT	Data inputs	TXDATA[63:0]
Clock to Out:		
$T_{GCKST_}$TBERR	Status outputs	TXBUFERR
$T_{GCKST_}$TKERR	Status outputs	TXKERR[7:0]
$T_{GCKDO_}$TRDIS	Data outputs	TXRUNDDISP[7:0]
Clock:		
T_{TX2PWH}	Clock pulse width, High state	TXUSRCLK2
T_{TX2PWL}	Clock pulse width, Low state	TXUSRCLK2

Table A-5: **PMA Clock Parameters**

Parameter	Function	Signals
$T_{GCCK\_}PADDR/T_{GCCK\_}PADDR$	Control inputs	PMAREGADDR[5:0]
$T_{GCCK\_}PINIT/T_{GCCK\_}PINIT$	Control inputs	PMAINIT
$T_{GCCK\_}PLKSEL/T_{GCCK\_}PLKSEL$	Control inputs	PMARXLOCKSEL
$T_{GCCK\_}PRW/T_{GCCK\_}PRW$	Control inputs	PMAREGRW
$T_{GCCK\_}PSTRB/T_{GCCK\_}PSTRB$	Control inputs	PMAREGSTROBE
$T_{GDCK\_}PDAT/T_{GCCK\_}PDAT$	Data inputs	PMAREGDATAIN[7:0]

Notes:

1. See Appendix C, "PMA Attribute Programming Bus" for additional timing information.

Table A-6: **Miscellaneous Clock Parameters**

Parameter	Function	Signals
T_{REFPWH}	Clock pulse width, High state	REFCLK(1)
T_{REFPWL}	Clock pulse width, Low state	REFCLK(1)
$T_{BREFNPWH}$	Clock pulse width, High state	BREFCLKNIN(1)
$T_{BREFPPWH}$	Clock pulse width, High state	BREFCLKPIN(1)
$T_{BREFNPWL}$	Clock pulse width, Low state	BREFCLKNIN(1)
$T_{BREFPPWL}$	Clock pulse width, Low state	BREFCLKPIN(1)
T_{TXPWH}	Clock pulse width, High state	TXUSRCLK(2)
T_{TXPWL}	Clock pulse width, Low state	TXUSRCLK(2)

Notes:

1. REFCLK is not synchronous to any RocketIO X signals.
2. TXUSRCLK is not synchronous to any RocketIO X signals.

XILINX®

Appendix B

8B/10B Valid Characters

Valid Data and Control 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 B-1 and Table B-2 show the Data and K tables of valid characters.

Table B-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

Table B-1: **Valid Data Characters *(Continued)***

Data Byte Name	Bits HGF EDCBA	Current RD – abcdei fghj	Current RD + abcdei fghj
D20.0	000 10100	001011 1011	001011 0100
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

Table B-1: **Valid Data Characters *(Continued)***

Data Byte Name	Bits HGF EDCBA	Current RD – abcdei fghj	Current RD + abcdei fghj
D20.1	001 10100	001011 1001	001011 1001
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

Table B-1: **Valid Data Characters *(Continued)***

Data Byte Name	Bits HGF EDCBA	Current RD – abcdei fghj	Current RD + abcdei fghj
D20.2	010 10100	001011 0101	001011 0101
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

Table B-1: **Valid Data Characters *(Continued)***

Data Byte Name	Bits HGF EDCBA	Current RD – abcdei fghj	Current RD + abcdei fghj
D20.3	011 10100	001011 1100	001011 0011
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

Table B-1: **Valid Data Characters *(Continued)***

Data Byte Name	Bits HGF EDCBA	Current RD – abcdei fghj	Current RD + abcdei fghj
D20.4	100 10100	001011 1101	001011 0010
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

Table B-1: **Valid Data Characters *(Continued)***

Data Byte Name	Bits HGF EDCBA	Current RD – abcdei fghj	Current RD + abcdei fghj
D20.5	101 10100	001011 1010	001011 1010
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

Table B-1: **Valid Data Characters *(Continued)***

Data Byte Name	Bits HGF EDCBA	Current RD – abcdei fghj	Current RD + abcdei fghj
D20.6	110 10100	001011 0110	001011 0110
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

Table B-1: **Valid Data Characters *(Continued)***

Data Byte Name	Bits HGF EDCBA	Current RD – abcdei fghj	Current RD + abcdei fghj
D20.7	111 10100	001011 0111	001011 0001
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 B-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 C

PMA Attribute Programming Bus

The RocketIO X transceivers provide a simple, parallel programming bus for dynamically configuring the PMA attribute settings. This gives the end user real-time control of PMA features without the need to use partial reconfiguration or to bring out discrete control ports to the fabric for each and every attribute (not feasible).

Note:

1. This feature is for ADVANCED USERS ONLY. Direct modification of these attributes should only be done with a thorough understanding of the capabilities, performance, and side-effects of the resulting settings. For most applications, the default PMA attribute settings provided in the software primitives should be adequate.
2. The attribute bus is not user accessible during normal FPGA configuration or a PMAINIT cycle (internally controlled for PMA initialization). Control is passed back to the fabric after initialization completes the updating of all registers.
3. The previous bitstream configuration settings of the PMA attributes are restored after assertion of PMAINIT and the subsequent initialization cycle, thus overwriting any modifications via the attribute programming bus.

Interface Description

The PMA attribute programming bus consists of a simple, strobed, byte-wide interface. The user accessible ports are defined in Table C-1. During FPGA configuration or a PMAINIT cycle, these ports have no effect on PMA attributes.

Table C-1: **PMA Attribute Bus Ports**

Port	Description
PMAREGADDR[5:0]	Register Address
PMAREGDATAIN[7:0]	Register Data In (from FPGA to PMA register)
PMAREGSTROBE	Strobe Asserted Low
PMAREGRW	Read Asserted High/Write Asserted Low
TXRUNDISP(1)(2)	Register Data Out (from PMA register to FPGA)

Notes:

1. TXRUNDISP is the Register Data Out when PMAREGADDR[5] is asserted; otherwise, it indicates the TX running disparity.
2. All read data must be inverted.

Register access is via a simple, active-low strobe cycle. A write cycle followed by a read cycle is illustrated in Figure C-1. When PMAREGSTROBE is unasserted (High), both read and write access are inhibited. PMAREGADDR[5], PMAREGRW, and PMAREGDATAIN[7:0] must be stable during PMAREGSTROBE asserted (Low) and also meet setup and hold timing relative to the falling and rising edges of PMAREGSTROBE, respectively.

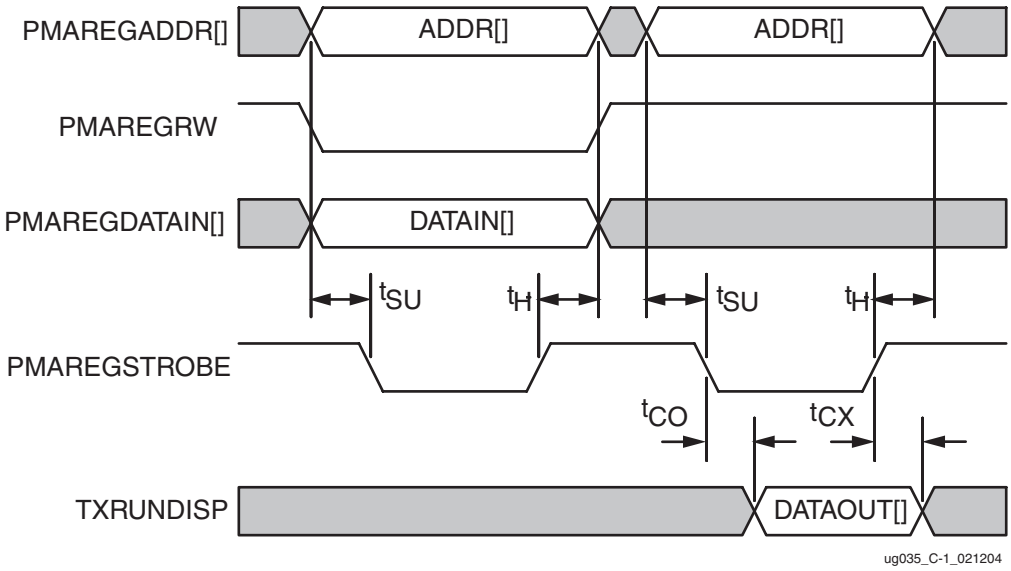


Figure C-1: **PMA Attribute Bus Waveform**

Memory Map

The PMA registers are byte-wide (8-bit) and memory mapped, based on PMAREGADDR[5:0]. The memory assignment and register layout are shown in Table C-2. Reserved bits are shown in grey and should be set to 0. Reserved registers are also shown in grey and should not be written.

Table C-2: **PMA Attribute Memory Map**

Address	Register Name	7	6	5	4	3	2	1	0
0x00	MASTERBIAS	VCODAC[5:0]						MASTERBIAS[1:0]	
0x01	TXDIVRATIOLO	TXDIVRATIO[7:0]							
0x02	TXDIVRATIOHI	SEL_DAC_FIX[3:2]		SEL_DAC_TRAN[3:2]		ENDCD	TXBUSWID	TXDIVRATIO[9:8]	
0x03	TXLOOPFILTER	IBOOST				TXLOOPFILTERR[1:0]		TXLOOPFILTERC[1:0]	
0x04	TXMODECONTROL	TXREG[1:0]		TXVSEL[1:0]		TXVCOGAIN	TXVCODAC		TXCPI
0x05	TXOUTPUTLEVEL		SLEW	EMPOFF	PRDRVOFF	TXDOWNLEVEL[3:0]			
0x06	TXOUTPUTMODE			TXANASW	TXDIGSW	TXEMPHLEVEL[3:0]			
0x07	RXDIVRATIOLO	RXDIVRATIO[7:0]							
0x08	RXDIVRATIOHI	SEL_DAC_TRAN[1:0]		RXDIVRATIO[13:8]					
0x09	RXLOOPFILTER	SEL_DAC_FIX[1:0]		AFE_FLAT_ENABLE	RXLOOPFILTERR[2:0]			RXLOOPFILTERC[1:0]	
0x0A	RXMODE0	RXREG[1:0]		RXVSEL[1:0]		RXVCOGAIN	RXVCODAC	RXCPI[0]	RXVCOSW

Table C-2: **PMA Attribute Memory Map *(Continued)***

Address	Register Name	7	6	5	4	3	2	1	0
0x0B	RXMODE1	RXCPI[1] (RXCPGAIN)	RXVSELCP[1:0]		RXFLTCPT[4:0]				
0x0C	RXFEICONTROL0	RXFER[1:0]		RXFLCPI[1:0]		RXFEI[1:0]		VSELAFE[1:0]	
0x0D	RXFEICONTROL1	RXFER[9:2]							
0x0E	<reserved>								
0x0F	POWERCONTROL	TXDRVEN	RXEN	TXEN		RXANAEN	TXDIGEN	TXANAEN	BIASEN
0x10-0x3F	<reserved>								

Register Definition

This section defines the individual PMA attribute vectors, as presented in Table C-2.

MASTERBIAS[1:0]

MASTERBIAS[1:0] selects the reference voltage used to generate bias currents throughout the Multi-Gigabit Transceiver (MGT). The default for all primitives is 00 (Bandgap). The voltage selection is as follows:

Table C-3: **MASTERBIAS[1:0] Definition**

MASTERBIAS[1:0]	Voltage Reference
00	Bandgap, Nominal Voltage (Default)
01	Bandgap, Higher Current
10	Bandgap, Lower Current
11	Voltage Divider, Bandgap Unused

VCODAC[5:0]

VCODAC[5:0] is a binary weighted current used to set the VCO center frequency in the transmit or receive VCO, when TXVCODAC or RXVCODAC, respectively, are Low or High. When in automatic mode (default), this value is unused.

TXDIVRATIO[9:0]

TXDIVRATIO[9:0] controls the divider ratios for TXCLK0, TXOUTCLK, and the PLL clock multiplier ratio of the transmitter relative. The defaults for these are primitive dependent, based on reference clock frequency and encoding.

Table C-4: **TX Clock Multiplier Ratio Definition**

TXDIVRATIO[3:0]	Divider
0000	÷ 4
0001	÷ 4.125
0010	÷ 5

Table C-4: **TX Clock Multiplier Ratio Definition *(Continued)***

TXDIVRATIO[3:0]	Divider
0011	Reserved
0100	÷ 8
0101	÷ 8.25
0110	÷ 10
0111	Reserved
1000	÷ 16
1001	÷ 16.5
1010	÷ 20
1011	Reserved
1100	Reserved
1101	Reserved
1110	Reserved
1111	Reserved

Table C-5: **TXCLK0 Divider Ratio Definition**

TXDIVRATIO[5:4]	Divider
00	÷ 8
01	÷ 8.25
10	÷ 10
11	Reserved

Table C-6: **TXOUTCLK Divider Ratio Definition**

TXDIVRATIO[9:6]	Divider
0000	÷ 4
0001	÷ 4.125
0010	÷ 5
0011	Reserved
0100	÷ 8
0101	÷ 8.25
0110	÷ 10
0111	Reserved
1000	÷ 16

Table C-6: **TXOUTCLK Divider Ratio Definition *(Continued)***

TXDIVRATIO[9:6]	Divider
1001	÷ 16.5
1010	÷ 20
1011	Reserved
1100	Reserved
1101	Reserved
1110	Reserved
1111	Reserved

TXBUSWID

TXBUSWID selects between wide and narrow internal parallel transmit data bus. The default is 1 (Wide). The bus width settings are defined as follows:

Table C-7: **TXBUSWID Definition**

TXBUSWID	TX Parallel Bus Width
0	16/20 bit
1	32/40 bit (default)

TXLOOPFILTERC[1:0]

TXLOOPFILTERC[1:0] selects the transmit PLL filter capacitor setting. The default is primitive dependent. The loop filter capacitor selection is as follows:

Table C-8: **TXLOOPFILTERC[1:0] Definition**

TXLOOPFILTERC[1:0]	TX Filter Capacitor
00	15 pF
01	30 pF
10	45 pF
11	60 pF

TXLOOPFILTERR[1:0]

TXLOOPFILTERR[1:0] selects the transmit PLL filter resistor setting. The default is primitive dependent. The loop filter resistor selection is as follows:

Table C-9: **TXLOOPFILTERR[1:0] Definition**

TXLOOPFILTERR[1:0]	TX Filter Resistor
00	12 kΩ
01	6 kΩ

Table C-9: **TXLOOPFILTERR[1:0] Definition *(Continued)***

TXLOOPFILTERR[1:0]	TX Filter Resistor
10	4 kΩ
11	3 kΩ

IBOOST

IBOOST selects between nominal and increased bias current in all regulators. The default is 0 (Nominal). The current is defined as follows:

Table C-10: **IBOOST Definition**

IBOOST	Regulator Bias Current
0	Nominal
1	Nominal + 37.5µA

TXCPI

TXCPI sets the transmit charge pump bias current. The default is primitive dependent. The transmit charge pump bias current is defined as follows:

Table C-11: **TXCPI Definition**

TXCPI	TX Charge Pump Bias Current
0	150 µA
1	300 µA

TXVCODAC

TXVCODAC selects between automatic and DAC control of transmit VCO center frequency. The default is 1 (Automatic, i.e., independent of VCODAC[5:0]). When set to DAC mode, the center frequency is controlled by the value of VCODAC[5:0]. The transmit center frequency control is defined as follows:

Table C-12: **TXVCODAC Definition**

TXVCODAC	TX VCO Center Frequency
1	DAC Controlled
0	Automatic (Default)

TXVCOGAIN

TXVCOGAIN selects between low and high gain of the transmit VCO. The default is 1 (High). The transmit VCO gain control is defined as follows:

Table C-13: **TXVCOGAIN Definition**

TXVCOGAIN	TX VCO Gain
0	Low
1	High (Default)

TXVSEL[1:0]

TXVSEL[1:0] selects the transmit regulator voltage. The default is 00 (Nominal). The regulator voltage selection is defined as follows:

Table C-14: **TXVSEL[1:0] Definition**

TXVSEL[1:0]	TX Regulator
00	Nominal (Default)
01	HIGH Vdd
10	Low Vdd
11	Voltage Divider

TXREG[1:0]

TXREG[1:0] selects the transmit regulator output current. The default is primitive dependent. The transmit regulator current is defined as follows:

Table C-15: **TXREG[1:0] Definition**

TXREG[1:0]	TX Regulator Current
00	15 mA
01	30 mA
10	45 mA
11	60 mA

TXDOWNLEVEL[3:0]

TXDOWNLEVEL[3:0] selects the transmit line driver current (and, thus, output voltage swing). The output swing is defined in the ***RocketIO X Transceiver User Guide***.

PRDRVOFF

PRDRVOFF enables and disables the line driver. The default is 0 (Enabled). When disabled, the output is kept at the common mode voltage. The line driver control is defined as follows:

Table C-16: **PRDRVOFF Definition**

PRDRVOFF	Line Driver
0	Enabled (Default)
1	Disabled

EMPOFF

EMPOFF enables and disables the emphasis feature in the line driver. The default is 0 (Enabled). When enabled, the emphasis level is set by TXEMPHLEVEL[3:0]. The emphasis control is defined as follows:

Table C-17: **EMPOFF Definition**

EMPOFF	Emphasis
0	Enabled (Default)
1	Disabled

SLEW

SLEW sets the slew rate of the line driver. The default is primitive dependent. The slew rate control is defined as follows:

Table C-18: **SLEW Definition**

SLEW	Slew Rate
0	Fast
1	Slow

TXEMPHLEVEL[3:0]

TXEMPHLEVEL[3:0] selects the transmit line driver emphasis current level (and, thus, emphasis voltage level). The emphasis levels are defined in the ***RocketIO X Transceiver User Guide***.

TXDIGSW

TXDIGSW selects the source of 1.5V transmit digital power supply. The default is primitive dependent. The transmit digital power supply selection is defined as follows:

Table C-19: **TXDIGSW Definition**

TXDIGSW	Transmit Digital Supply
0	Regulated (from 2.5V)
1	Unregulated (from 1.5V)

TXANASW

TXANASW selects the source of 1.5V transmit analog power supply. The default is primitive dependent. The transmit analog power supply selection is defined as follows:

Table C-20: **TXANASW Definition**

TXANASW	Transmit Analog Supply
0	Regulated (from 2.5V)
1	Unregulated (from 1.5V)

RXDIVRATIO[13:0]

RXDIVRATIO[13:0] controls the divider ratios for RXCLK0, RXRECCLK, and the PLL clock multiplier ratio of the receiver relative to the receiver bit rate. In addition, RXDIVRATIO[13] controls a programmable divide by 2 on the BREFCLK input to the phase detector. The defaults for these are primitive dependent, based on reference clock frequency and encoding.

Table C-21: **RX Clock Multiplier Ratio Definition**

RXDIVRATIO[3:0]	Divider
0000	÷ 8
0001	÷ 8.25
0010	÷ 10
0011	Reserved
0100	÷ 16
0101	÷ 16.5
0110	÷ 20
0111	Reserved
1000	÷ 32
1001	÷ 33
1010	÷ 40
1011	Reserved

Table C-21: **RX Clock Multiplier Ratio Definition *(Continued)***

RXDIVRATIO[3:0]	Divider
1100	Reserved
1101	Reserved
1110	Reserved
1111	Reserved

Table C-22: **RXCLK0 Divider Ratio Definition**

RXDIVRATIO[11:8]	Divider
0000	÷ 8
0001	÷ 8.25
0010	÷ 10
0011	Reserved
0100	÷ 16
0101	÷ 16.5
0110	÷ 20
0111	Reserved
1000	÷ 32
1001	÷ 33
1010	÷ 40
1011	Reserved
1100	Reserved
1101	Reserved
1110	Reserved
1111	Reserved

Table C-23: **RXRECCLK Divider Ratio Definition**

RXDIVRATIO[7:4]	Divider
0000	÷ 4
0001	÷ 4.125
0010	÷ 5
0011	Reserved
0100	÷ 8
0101	÷ 8.25
0110	÷ 10

Table C-23: **RXRECCLK Divider Ratio Definition *(Continued)***

RXDIVRATIO[7:4]	Divider
0111	Reserved
1000	÷ 16
1001	÷ 16.5
1010	÷ 20
1011	Reserved
1100	Reserved
1101	Reserved
1110	Reserved
1111	Reserved

Table C-24: **VCO Divider Ratio Definition**

RXDIVRATIO[12]	VCO Divider
0	÷ 1
1	÷ 2

Table C-25: **BREFCLK Divider Ratio Definition**

RXDIVRATIO[13]	BREFCLK Divider (Phase Detector)
0	÷ 1
1	÷ 2

RXLOOPFILTERC[1:0]

RXLOOPFILTERC[1:0] selects the receiver PLL filter capacitor setting. The default is primitive dependent. The receiver loop filter capacitor selection is as follows:

Table C-26: **RXLOOPFILTERC[1:0] Definition**

RXLOOPFILTERC[1:0]	RX Filter Capacitor
00	15 pF
01	30 pF
10	30 pF
11	45 pF

RXLOOPFILTERR[2:0]

RXLOOPFILTERR[2:0] selects the receive PLL filter resistor setting. The default is primitive dependent. The receiver loop filter resistor selection is as follows:

Table C-27: **RXLOOPFILTERR[2:0] Definition**

RXLOOPFILTERR[2:0]	RX Filter Resistor
000	12 kΩ
001	6 kΩ
010	5 kΩ
011	4 kΩ
100	3 kΩ
101	2.25 kΩ
110	1.5 kΩ
111	0.75 kΩ

RXVCOSW

RXVCOSW selects source of 1.5V receiver VCO power supply. The default is primitive dependent. The receiver VCO power supply selection is defined as follows:

Table C-28: **RXVCOSW Definition**

RXVCOSW	Receiver VCO Supply
0	Regulated (from 2.5V)
1	Unregulated (from 1.5V)

RXCPGAIN, RXCPI

RXCPGAIN/RXCPI sets the receiver charge pump bias current. The default is primitive dependent. The receiver charge pump bias current is defined as follows:

Table C-29: **RXCPGAIN/RXCPI[1:0] Definition**

RXCPGAIN/RXCPI[1:0]	RX Charge Pump Bias Current
00	150 µA
01	300 µA
10	300 µA
11	600 µA

RXVCODAC

RXVCODAC selects between automatic and DAC control of receiver VCO center frequency. The default is 0 (Automatic, i.e., independent of VCODAC[5:0]). When set to

DAC mode, the center frequency is controlled by the value of VCODAC[5:0]. The receiver center frequency control is defined as follows:

Table C-30: **RXVCODAC Definition**

RXVCODAC	RX VCO Center Frequency
0	Automatic (Default)
1	DAC Controlled

RXVCOGAIN

RXVCOGAIN selects between low and high gain of the receiver VCO. The default is 1 (High). The receiver VCO gain control is defined as follows:

Table C-31: **RXVCOGAIN Definition**

RXVCOGAIN	RX VCO Gain
0	Low
1	High (Default)

RXVSEL[1:0]

RXVSEL[1:0] selects the receiver regulator voltage. The default is 00 (Nominal). The receiver regulator voltage selection is defined as follows:

Table C-32: **RXVSEL[1:0] Definition**

RXVSEL[1:0]	RX Regulator
00	Nominal (Default)
01	High Vdd
10	Low Vdd
11	Voltage Divider

RXREG[1:0]

RXREG[1:0] sets the receiver VCO regulator output current. The default is primitive dependent. The receiver regulator current is defined as follows:

Table C-33: **RXREG[1:0] Definition**

RXREG[1:0]	RX Regulator Current
00	13 mA
01	27 mA
10	40 mA
11	53 mA

RXVSELCP[1:0]

RXVSELCP[1:0] sets the receiver charge pump common mode voltage. The default is primitive dependent. The receiver charge pump common mode voltage is defined as follows:

Table C-34: **RXVSELCP[1:0] Definition**

RXVSELCP[1:0]	RX Charge Pump Common Mode
00	600 mV
01	720 mV
10	840 mV
11	960 mV

RXFLTCPT[4:0]

RXFLTCPT[4:0] controls the phase adjustment of the receiver sample clock relative to the data. RXFLTCPT[4] is the sign and RXFLTCPT[3:0] are binary weighted magnitude. The magnitude is relative to the data unit interval (UI), with a range of ~±0.3UI.

Note: Need to determine early/late relationship of the sign.

VSELAFE[1:0]

VSELAFE[1:0] sets the receiver analog front end (AFE) common mode input voltage. The default is primitive dependent. The receiver AFE common mode voltage is defined as follows:

Table C-35: **VSELAFE[1:0] Definition**

VSELAFE[1:0]	RX AFE Common Mode Input
00	720 mV
01	840 mV
10	960 mV
11	1080 mV

RXFEI[1:0]

RXFEI[1:0] sets the receiver front end/equalizer current. The default is 11 (24 mA). The receiver front-end current is defined as follows:

Table C-36: **RXFEI[1:0] Definition**

RXFEI[1:0]	RX Front End Current
00	6 mA
01	12 mA

Table C-36: **RXFEI[1:0] Definition *(Continued)***

RXFEI[1:0]	RX Front End Current
10	18 mA
11	24 mA (Default)

RXFER[9:0]

RXFER[9:0] sets the equalization in the receiver front end/equalizer, based on the adjustment of four basic boosts in four different frequency ranges. For details, see "Receive Equalization" in Chapter 4 in the ***RocketIO X Transceiver User Guide.***

RXFLCPI[1:0]

RXFLCPI[1:0] sets the fine loop charge pump current. The default is 00 (40 µA). The receiver fine loop charge pump current is defined as follows:

Table C-37: **RXFLCPI[1:0] Definition**

RXFLCPI[1:0]	RX Fine Loop CP Current
00	40 µA (Default)
01	80 µA
10	20 µA
11	60 µA

BIASEN

BIASEN enables and disables the MGT bias, including bandgap, V-to-I and clock management. The default is 1 (Enabled). The bias control is defined as follows:

Table C-38: **BIASEN Definition**

BIASEN	Bias
0	Disabled
1	Enabled (Default)

TXANAEN

TXANAEN enables and disables the transmit analog voltage regulator. The default is 1 (Enabled). For proper operation, the bias must also be enabled. The transmit analog regulator control is defined as follows:

Table C-39: **TXANAEN Definition**

TXANAEN	TX Analog Regulator
0	Disabled
1	Enabled (Default)

TXDIGEN

TXDIGEN enables and disables the transmit digital voltage regulator. The default is 1 (Enabled). For proper operation, the bias must also be enabled. The transmit digital regulator control is defined as follows:

Table C-40: **TXDIGEN Definition**

TXDIGEN	TX Digital Regulator
0	Disabled
1	Enabled (Default)

RXANAEN

RXANAEN enables and disables the receiver analog voltage regulator. The default is 1 (Enabled). For proper operation, the bias must also be enabled. The receiver analog regulator control is defined as follows:

Table C-41: **RXANAEN Definition**

RXANAEN	RX Analog Regulator
0	Disabled
1	Enabled (Default)

TXEN

TXEN enables and disables transmit operation. The default is 1 (Enabled). For proper operation, the bias must also be enabled. The transmit control is defined as follows:

Table C-42: **TXEN Definition**

TXEN	TX Operation
0	Disabled
1	Enabled (Default)

RXEN

RXEN enables and disables receiver operation. The default is 1 (Enabled). For proper operation, the bias must also be enabled. The receiver control is defined as follows:

Table C-43: **RXEN Definition**

RXEN	RX Operation
0	Disabled
1	Enabled (Default)

TXDRVEN

TXDRVEN enables and disables transmit line driver operation. The default is 1 (Enabled). For proper operation, the bias and transmitter must also be enabled. The transmit line driver control is defined as follows:

Table C-44: **TXDRVEN Definition**

TXDRVEN	TX Line Driver Operation
0	Disabled
1	Enabled (Default)

PMAINIT

When PMAINIT is asserted (High), the FPGA configuration attributes are automatically loaded into the PMA. This overwrites any PMA attribute modifications made via the PMA attribute programming bus. This feature can be used to restore the FPGA bitstream configuration from memory or to load newly configured attributes following a partial reconfiguration. Because of the automatic configuration process used to load PMA attributes after assertion of PMAINIT, the PMA attribute bus has no operational effect until after initialization has completed the updating of all registers.

SEL_DAC_TRAN[3:0]

This attribute is used to program the transition current in the fine loop charge pump. The default setting sets the current to its nominal value and should be set to 0000. See "Data-Density Independent Phase Adjustment for CDR."

SEL_DAC_FIX[3:0]

This attribute is used to program the fixed current in the fine loop charge pump. The default setting set the current to its nominal value and should be set to 0000. See "Data-Density Independent Phase Adjustment for CDR."

ENDCD

This attribute enables better transmitter jitter performance. The default value is mode-dependent. It is 1 for high-speed operation (greater than or equal to 5 Gb/s) or 0 for low-speed operation (less than 5 Gb/s).

AFE_FLAT_ENABLE

This attribute makes the AFE response flat (no peaking). The default is 0.

Data-Density Independent Phase Adjustment for CDR

This section describes the programmability of the fine loop charge pump (CP) to achieve offset adjustment that is data-density independent. The fine loop CP is made of a Phase CP, Transition CP, and DC current sources as shown in Figure C-2.

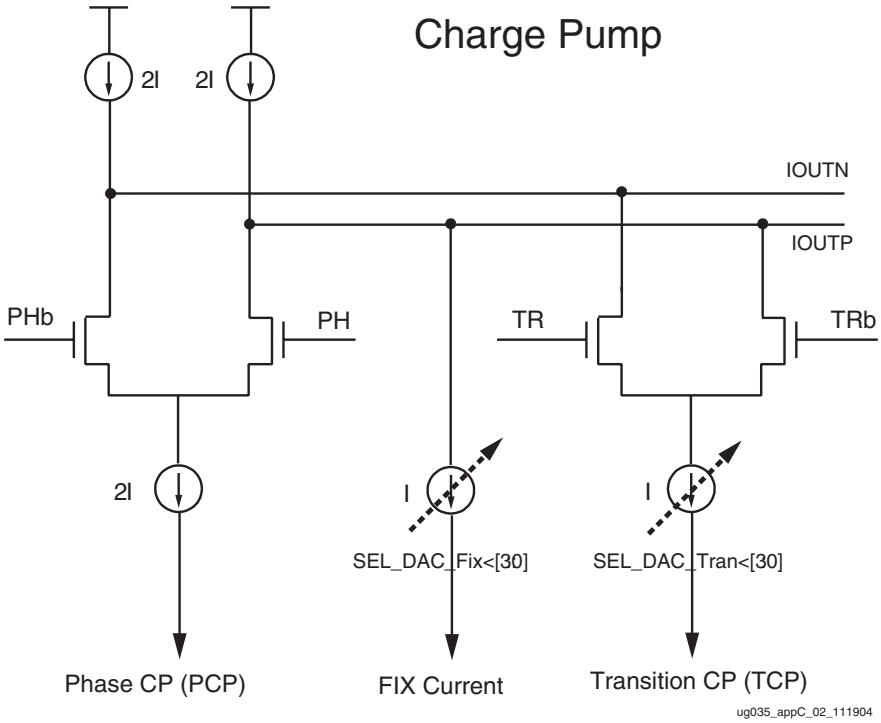


Figure C-2: **Fine Loop Charge Pump**

Data-density independent sampling point control is achieved by forcing the fine loop CP to source/sink charge on the loop filter, only when there is a transition in the data under locking conditions, i.e., when the loop is locked. This can be done by making the tail currents of TRAN_CP and DC_CP currents programmable in a complementary fashion. Each tail current is made programmable by a 4-bit bus according to Table C-45.

Table C-45: **Tail Current Value Vs. Programmability Code**

Code	Tail Current	Code	Tail Current
0	I_n *(1 + 0 * 0.125)	8	I_n *(1 - 8 * 0.125)
1	I_n *(1 + 1 * 0.125)	9	I_n *(1 - 7 * 0.125)
2	I_n *(1 + 2 * 0.125)	10	I_n *(1 - 6 * 0.125)
3	I_n *(1 + 3 * 0.125)	11	I_n *(1 - 5 * 0.125)
4	I_n *(1 + 4 * 0.125)	12	I_n *(1 - 4 * 0.125)
5	I_n *(1 + 5 * 0.125)	13	I_n *(1 - 3 * 0.125)
6	I_n *(1 + 6 * 0.125)	14	I_n *(1 - 2 * 0.125)
7	I_n *(1 + 7 * 0.125)	15	I_n *(1 - 1 * 0.125)

Where I_n is the nominal value of the tail current of DC_CP or Tran_CP, and the code is evaluated in decimal.

The default code for DC_CP current is 0 and for the Tran_CP code is 0. This will set the two currents to their nominal values. DC_CP bus is accessed through SEL_DAC_FIX [3:0]

attributes from the FPGA and Tran_CP is accessed though SEL_DAC_TRAN[3:0] from the FPGA.

The programmability is always written in the form of (SEL_DAC_TRAN[3:0], SEL_DAC_FIX[3:0]). For example, (0,0) is the default as explained above. Table C-46 shows the codes that achieve data-density independent control with their corresponding sampling point offset. Code (5,11) moves the sampling point to right of the ideal sampling point by
5*0.125*0.5UI = 0.3125UI.

Figure C-3 shows examples of the sampling point offsets.

Table C-46: **Allowed Programmable Codes**

Code	Resulting Offset (UI)	Code	Resulting Offset (UI)
(0, 0)	0	(8, 8)	± 0.5
(1, 15)	+ 0.0625	(9, 7)	- 0.4375
(2, 14)	+ 0.125	(10, 6)	- 0.375
(3, 13)	+ 0.1875	(11, 5)	- 0.3125
(4, 12)	+ 0.25	(12, 4)	- 0.25
(5, 11)	+ 0.3125	(13, 3)	- 0.1875
(6, 10)	+ 0.375	(14, 2)	- 0.125
(7, 9)	+ 0.4375	(15, 1)	- 0.0625

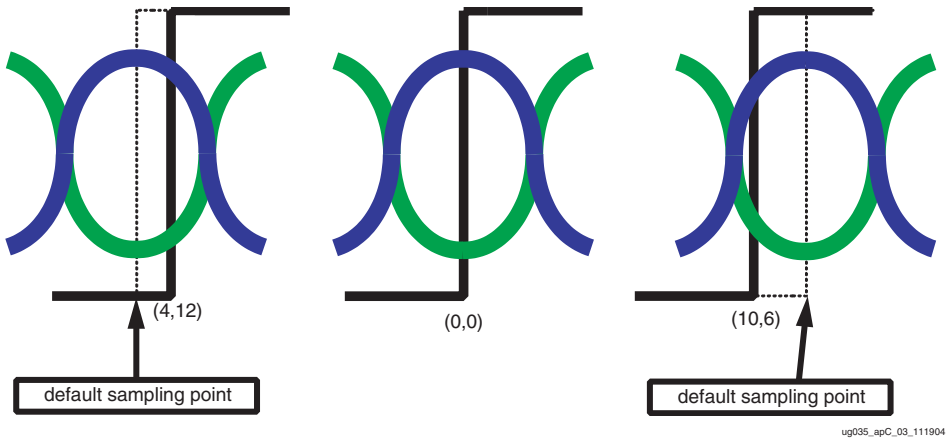


Figure C-3: **Sampling Point Offset**

Users should be advised that depending on the real sampling point location, some of the codes above can cause either loss of lock or higher BER. A recommended approach to enhance the performance is always to start from the default and then select a different code by gradually stepping through Table C-46.

Although not recommended, the user can use different codes than those appearing in Table C-46. For example, a code of (1, 1) can be used. This code causes the TRAN_CP current and the DC_CP current to be 12.5% higher than the nominal value, which will

cause an offset in the sampling point. However, this offset is data-density dependent and can eventually result in a loss of lock.

XILINX®

Appendix D

Virtex-II Pro to Virtex-II Pro X FPGA Design Migration

Introduction

This appendix describes important differences regarding migration from Virtex-II Pro™ to the Virtex-II Pro™ X FPGAs. Note that 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-II Pro X FPGAs, refer to the *Virtex-II Pro Data Sheet* (DS083) and the RocketIO *User Guide* (UG024).

Primary Differences

Virtex-II Pro X FPGAs are pin compatible with corresponding Virtex-II Pro family members. The primary differences between Virtex-II Pro and Virtex-II Pro X FPGAs are:

- BREFCLK - High-speed, lower jitter upgrade supporting up to 6.25 Gb/s I/O
- Multi-Gigabit Transceiver (MGT) - supporting 2.488 Gb/s to 6.25 Gb/s I/O

BREFCLK

As with Virtex-II Pro FPGAs, at speeds of 2.5 Gb/s or greater, the REFCLK configuration introduces more than the maximum allowable jitter to the RocketIO X transceiver. For this reason, the BREFCLK configuration is required.

The BREFCLK configuration uses dedicated routing resources that reduce jitter. BREFCLK differences between Virtex-II Pro and Virtex-II Pro X FPGAs are summarized in Table D-1.

Table D-1: **BREFCLK Differences Summary**

Differences	Virtex-II Pro FPGAs	Virtex-II Pro X FPGAs
BREFCLK Inputs	2 top, 2 bottom	1 top, 1 bottom
Reference Frequency	Up to 156.25 MHz	Up to 645 MHz
Termination	On-chip or Off-chip 100Ω Differential	On-chip 100Ω Differential
BREFCLK Selection	Attribute (REF_CLK_V_SEL) and Port (REFCLKSEL)	Port (REFCLKBSEL)

BREFCLK must enter the FPGA through dedicated differential clock I/O. This BREFCLK can connect to the BREFCLK input of the transceiver and the CLKIN input of the DCM for creation of USRCLKs. If transceivers on both the top and bottom of the FPGA are to be used, two BREFCLKs must be created: one for the top of the chip and one for the bottom. These dedicated clocks use the same clock inputs for all packages, as shown in Table D-2.

Table D-2: **BREFCLK Inputs**

BREFCLK			
Top		Bottom	
P	GCLK4S	P	GCLK6P
N	GCLK5P	N	GCLK7S

Figure D-1 shows how REFCLK and BREFCLK are selected through use of REFCLKSEL and REFCLKBSEL.

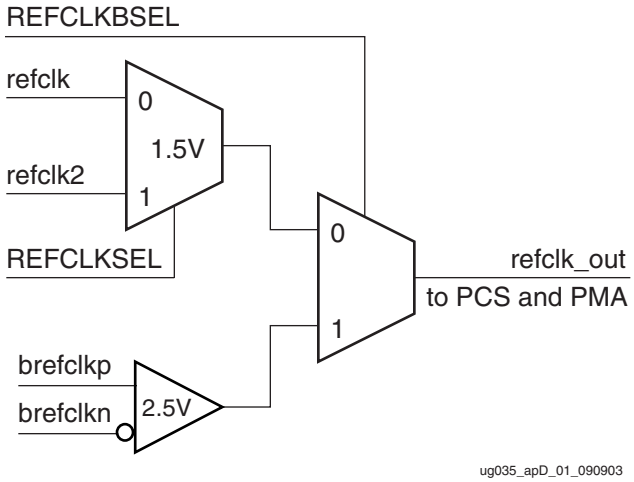


Figure D-1: **REFCLK/BREFCLK Selection Logic**

Although the BREFCLK2 pins from Virtex-II Pro FPGA are still available as global clock resources, they do not support a dedicated, low-jitter reference clock to the Virtex-II Pro X MGTs.

Table D-3 shows the BREFCLK pin numbers for Virtex-II Pro X packages. These are compatible with corresponding Virtex-II Pro package pins. Note that these pads must be used for BREFCLK operations.

Table D-3: **Virtex-II Pro X BREFCLK Pin Numbers**

Package	BREFCLK Pin Numbers	
	Top (P/N)	Bottom (P/N)
FF896	F16/G16	AH16/AJ16
FF1704	G22/F22	AU22/AT22

The Virtex-II Pro X BREFCLK inputs are LVDS compatible with on-chip, 100Ω differential termination. Reference oscillators similar to the ones used with Virtex-II Pro FPGAs can be used with Virtex-II Pro X FPGAs. These include the Epson EG2121CA and EG2101CA series, which support the full range of BREFCLK frequencies (up to 645 MHz). For more information, see the reference clock sections of this guide and the RocketIO *Transceiver User Guide* (**UG024**).

Power Regulation and Filtering

As with Virtex-II Pro FPGAs, each Virtex-II Pro X transceiver has five power supply pins, all of which are sensitive to noise. Table D-4 summarizes the Virtex-II Pro X power supply pins (relative to Virtex-II Pro), including their names, associated voltages, and power requirements.

To operate properly, RocketIO and RocketIO X transceivers require a certain level of noise isolation from surrounding noise sources. For this reason, it is required that both dedicated voltage regulators and passive high-frequency filtering be used to power RocketIO and RocketIO X circuitry. Voltage regulation requirements for Virtex-II Pro X FPGAs are identical to Virtex-II Pro FPGAs. Two Linear Technology LT1963 regulator circuits are needed to generate the required 1.5V and 2.5V supplies. For more information, see the voltage regulation sections of this guide and the *RocketIO Transceiver User Guide* (**UG024**).

Table D-4: **Voltage Changes for Virtex-II Pro X FPGA Power Regulation**

Virtex-II Pro		Virtex-II Pro X(1)	
Supply	**Voltage**	**Supply**	**Voltage**
AVCCAUXTX	2.5V	AVCCAUXTX	2.5V
AVCCAUXRX	2.5V	AVCCAUXRX	1.5V
VTTX	1.8V–2.625V	VTTX	1.5V
VTRX	1.8V–2.625V	VTRX	0.25–2.5V
GNDA	0V	GNDA	0V

Notes:
1. Power listed is the estimated power per MGT for the individual supply over all MGT operating modes. The total power per MGT is mode specific and less than the sum total of these values.

Figure D-2 shows the power filtering network for one transceiver. Each transceiver power pin requires one ferrite bead. The ferrite bead is the Murata BLM18AG102SN1.

Note: These components must not be shared under any circumstances.

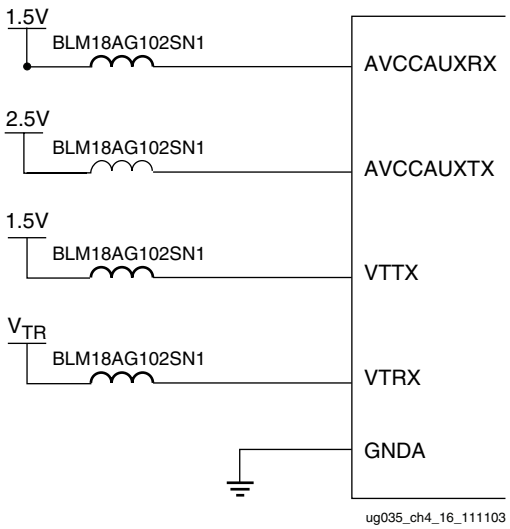


Figure D-2: **Power Filtering Network for One Transceiver**

High-Speed Serial I/O Termination

Virtex-II Pro on-chip termination is programmable to 50Ω or 75Ω. Virtex-II Pro X transceivers support only a fixed 50Ω on-chip termination.

The output driver and termination are powered by VTTX at 1.5V. This configuration uses a CML approach with 50Ω to TXP and TXN as shown in Figure D-3.

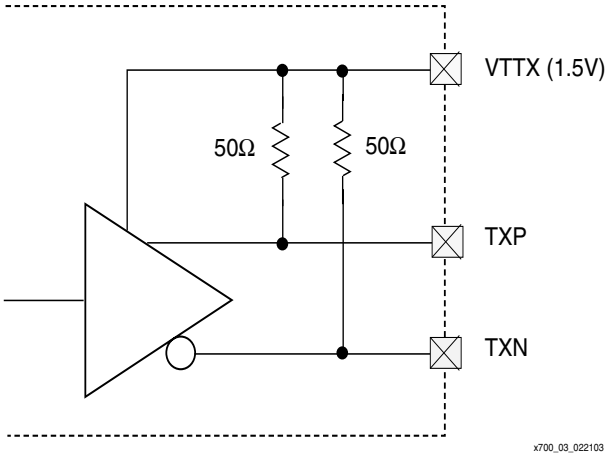


Figure D-3: **Transmit Termination**

The receiver termination supply (VTRX) is the center tap of differential termination to RXP and RXN as shown in Figure D-4. This supports multiple termination styles, including high-side, low-side, and differential (floating or active). This configuration supports receiver termination compatible to Virtex-II Pro devices, using a CML (high-side) termination to an active supply of 1.8-2.5V. For DC coupling of two Virtex-II Pro X devices, a 1.5V CML termination for VTRX is recommended.

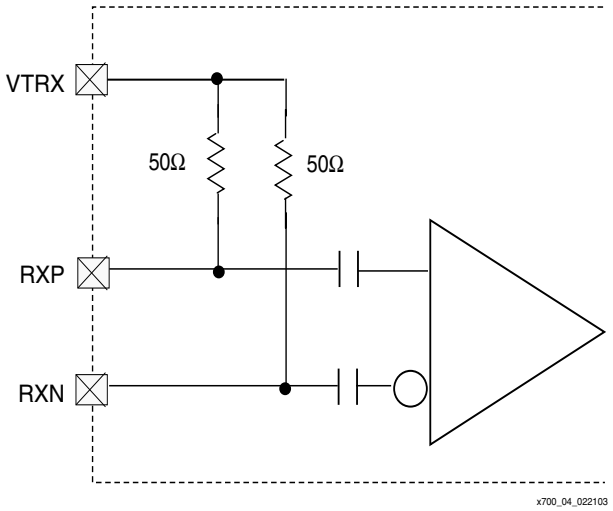


Figure D-4: **Receive Termination**

Migration Differences

The following is a list of differences that must be considered when migrating a design from using RocketIO transceivers to using RocketIO X transceivers.

Port Widths and Byte Mapping

All ports that are byte mapped are 8 bits wide instead of 4 bits wide as in RocketIO X transceivers to support up to 8 bytes or eight 10-bit words at the data interface. The mapping is the same. For instance, RXCHARISK[0] denotes if RXDATA[7:0] is a K-character; RXCHARISK[1] denotes if RXDATA[15:8] is a K-character, etc.

CRC

There is no CRC functionality built into the RocketIO X transceiver. CRC must be done in the fabric if required. Xilinx Application Note, XAPP209 describes how to implement CRC in the FPGA fabric.

Comma Alignment

The attribute ALIGN_COMMA_MSB is replaced by the attribute ALIGN_COMMA_WORD. It can be set to 1, 2, and 4. The RocketIO transceiver required a circuit for aligning the RXDATA on 4-byte boundaries. With the RocketIO X transceiver that functionality can be accomplished by setting ALIGN_COMMA_WORD to 4.

The ENPCOMMAALIGN and ENMCOMMAALIGN are now synchronous to RXUSRCLK2. There is no need to clock them with a register in the RXRECCLK domain as there was with the RocketIO transceiver.

The PCOMMA_10B_VALUE, MCOMMA_10B_VALUE, and COMMA_10B_MASK attributes are bit swapped. For instance, with the RocketIO transceiver, to set the comma value to positive K28.5 one sets PCOMMA_10B_VALUE = 0011111010. With the RocketIO X transceiver one sets PCOMMA_10B_VALUE = 0101111100.

Port RXCOMMADETUSE is added and must be asserted High to enable comma alignment.

Reference Clocking

BREFCLKPIN and BREFCLKNIN pins can only be routed to all transceivers on one side of the chip.

Clocking and Data Width

TXUSRCLK/TXUSERCLK2 and RXUSRCLK/RXUSRCLK2 have the same phase and ratio dependence on the data path width as in the RocketIO transceiver.

TXOUTCLK is a new port that is derived from the BREFCLK or REFCLK input. This signal can be routed to a DCM to produce the USRCLKs. The TXOUTCLK can be set to be a divided version of the reference clock for use in 64B/66B applications. BREFCLK or REFCLK can still be routed directly to a DCM or BUFGMUX as with the RocketIO transceiver.

Ports for data width have been added. RXDATAWIDTH/TXDATAWIDTH replace the TX_DATA_WIDTH and RX_DATA_WIDTH attributes from the RocketIO transceiver.

Table 2-2, page 42 shows the settings for RXDATAWIDTH and TXDATAWIDTH and the corresponding data widths. It also shows the corresponding internal bus width described below.

Only the first 3 modes (1-byte, 2-byte, and 4-byte) apply to designs migrating from the RocketIO transceiver.

Table 2-3, page 42 shows the settings for RXINTDATAWIDTH and TXINTDATAWIDTH and the corresponding internal bus widths.

Most Virtex-II Pro designs will use internal data widths of 01 corresponding to a 20-bit internal data bus, which the RocketIO transceiver has. The wider internal buses are used for higher data rates.

Channel Bonding

The CHBONDDONE signal operation has changed. See CHBONDDONE, RXCLKCORCNT, and RXBUFSTATUS in Table 1-4, page 26.

The CHANNEL_BOND_WAIT and CHANNEL_BOND_OFFSET are gone. For the RocketIO X transceiver, they are derived from CHANNEL_BOND_LIMIT, which should be set to 16. CHAN_BOND_SEQ_*_MASK has been added. It is a 4-bit signal that allows the user to mask out any part of the channel bonding sequence (make it a *don't care*) by setting any bit of the mask to 1 (all zeros only bond on the exact sequence; a mask 0100 ignores the third byte).

CHAN_BOND_64B66B_SV is unused and should always be set to FALSE.

Clock Correction

New attributes CLK_COR_MAX_LAT and CLK_COR_MIN_LAT allow the latency through the elastic buffer to be set. This replaces the function of XAPP 670. CLK_COR_8B10B_DE sets whether the CLK_COR_SEQ_* attributes are set using 8-bit (CLK_COR_8B10B_DE=TRUE) or 10-bit (CLK_COR_8B10B_DE=FALSE).

Drop mode clock correction is added. CLK_COR_SEQ_DROP should be FALSE to be compatible with the RocketIO designs.

64B/66B

The following ports control 64B/66B and should be tied to GND since RocketIO did not have this feature:

- RXDESCRAM64B66BUSE
- RXIGNOREBTF
- RXBLOCKSYNC64B66BUSE
- RXDEC64B66BUSE
- TXENC64B66BUSE
- TXGEARBOX64B66BUSE
- TXSCRAM64B66BUSE

The following attributes control 64B/66B and the values do not matter:

- SH_CNT_MAX
- SH_INVALID_CNT_MAX

8B/10B

Basic operation is the same. Attribute RX_DECODE_USE is replaced by a port RXDEC8B10BUSE. The new port TXENC8B10BUSE enables the 8B10B encoder, and port TXBYPASS8B10B allows bypassing of the encoder on a per byte basis. Port TXKERR does not function in 8B10B mode and is only used for 64B/66B encoding.

PMA

The following ports allow access to the Attribute Programming Bus. This accomplishes the same function as Xilinx Application Note XAPP660 by allowing the user to dynamically change the differential swing, pre-emphasis, and equalization settings.

- PMAREGADDR
- PMAREGDATAIN
- PMAREGRW
- PMAREGSTROBE

See the Appendix C, "PMA Attribute Programming Bus" for details on how to use these signals.

- PMA_SPEED – This attribute sets the clock multiplier and internal PLL settings to work with that clock. This effectively sets the data rate. For supported PMA_SPEED mode settings, see "PMA," page 77. Also, see Appendix C, "PMA Attribute Programming Bus" for other details.
- PMA_PWR_CNTRL is a new attribute and must always be set all 1s.

PMA new and useful ports:

- PMAINIT – This port reinitializes the PMA with the original values set by the PMA_SPEED attribute.
- PMARXLOCKSEL – This port allows the user to select the behavior of the RX PLL. See Table D-4, page 159 for details.
- PMARXLOCK - This port indicates that the RX PLL has locked in the fine loop.

Serialization

In the RocketIO transceiver, the most significant byte is sent first; in the RocketIO X transceiver, the least significant byte is sent first. Within a byte, the bits are still sent in the same order, i.e., bit 9 is sent first and bit 0 is sent last. See section "8B/10B," page 43 for details.

Miscellaneous

The LOOPBACK port supports four modes. The first three are identical to the RocketIO transceiver.

00 – no loopback

01 – internal parallel loopback

10 – post-driver serial loopback (loopback after the output driver)

11 – pre-driver serial loopback (loopback before the output driver)

TX_BUFFER_USE and RX_BUFFER_USE must still be set to TRUE.

Appendix E

Serial Backplane System Design

Additional PCB design guidelines are required to meet the demands of the RocketIO X transceiver for operation above 3.125 Gb/s. Backplane system design guidelines can be divided into three separate components. One component is directed toward the launch from the package onto the line card or switch card PCB. A second component is focused on the launch from the backplane connector onto either the line card or the backplane. A third component covers the transmission lines that link the backplane connectors to each other as well as the RocketIO X transceiver package.

Transmission Lines

The line card or switch card can be designed using microstrip or stripline technology. When using microstrip lines, care must be taken to assure that a clean path exists from the package to the backplane connector. Microstrip lines are ideal from the standpoint that they eliminate the need for a via to internal signal layers. Unfortunately, microstrip lines are not normally conducive to the design of high-density backplane systems.

Careful consideration should be given to the size of transmission line structures. Both the line card and the backplane should avoid the use of minimal width transmission lines.

Note: A transmission line width of **no less than 8 mil** is recommended to minimize skin effect and dielectric losses.

Connector to PCB Launch

Care must be taken in choosing the appropriate connector solution for the given data rate of operation. Traditional connector technology relies on the use of large plated through hole (PTH) vias to provide a robust mechanical mating interface between the backplane and line card. High-density backplane applications require the use of multiple signal routing layers, resulting in via stubs of varying depths. Optimizing via effects is therefore the key to maximizing overall backplane transmission distance.

One solution is backdrilling or counter-boring. Counter-boring is accomplished by drilling out any unused portion of the PTH via from the underside of the backplane. Figure E-1 shows an illustration of how backdrilling is achieved.

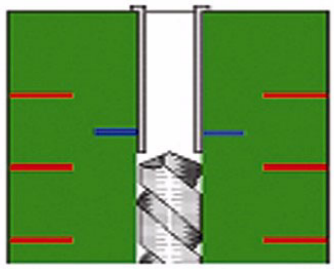


Figure E-1: **Backdrilling Process**

Removing the unused part of the via can lead to vast improvements in the quality of the data being transmitted and received. The plot shown in Figure E-2 illustrates the improvement that can be achieved by carefully removing most of the via stub. Because backdrilling is not an exact science, reliable design dictates that there always remain a small amount of unused via that cannot be backdrilled. The smaller the via stub that remains, the farther the resonance frequency of the via can be pushed out.

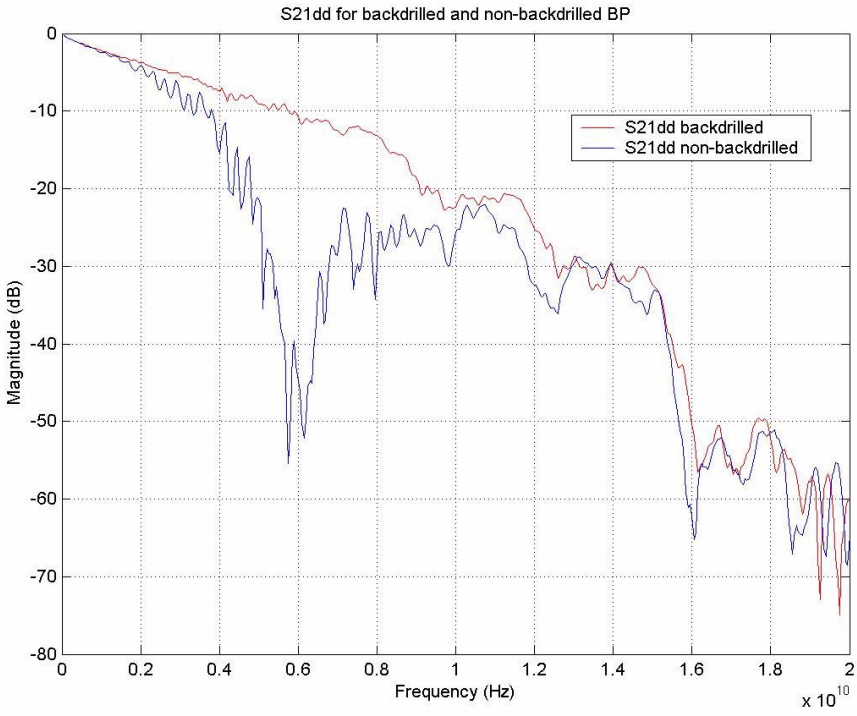


Figure E-2: **Backdrilled vs Non-Backdrilled Channel Characteristics**

A second way to reduce via effects is to use a blind via or a buried via. Building a backplane with blind and/or buried vias requires the PCB manufacturer to build and drill the backplane in multiple stages. First, a standard PTH via is used to connect the signal from the top layer of a board to the bottom of a board. A second PCB is then attached to the first to complete the backplane, effectively moving the signal layer from the bottom layer

to an inner layer. This technique completely eliminates the via stub; however, it is often difficult to design boards with multiple blind via depths.

A third and final way to reduce via stub effects is to route signals on the lower layers within the backplane. Various combinations of these techniques can be used to improve backplane performance. Regardless of the method, via stubs on the backplane should be limited to 50 mil or less to ensure that the resonance point of the via gets pushed out past 10 GHz. Larger via stubs can be used, but they directly impact the backplane length that can be achieved.

Not all signal layers require a unique solution. Signals running on routing layers that are near each other might be able to share the same backdrill or blind via depth. Signals running on the lower layers of the backplane do not need to be backdrilled if they are already less than 50 mil from the bottom.

Antipad design can also have an effect on the performance of the backplane. The use of rectangular antipads placed around each signal pair is recommended to reduce via capacitance. Reducing via capacitance with larger antipads helps to limit the amount of signal loss through the via. However, larger antipads can also produce additional crosstalk between signal pairs. The balance between signal loss and crosstalk must be determined based on the backplane application.

Crosstalk should also be considered when developing the backplane routing configuration. Receiver input pairs should not be surrounded by transmitter output pairs within the backplane connector. Routing a single transmit pair next to a single receive pair is acceptable, but ideally the backplane should be designed so that all transmitter output pairs are routed through one half of the connector, and all receiver input pairs are routed through the other half of the connector. Careful attention should also be paid to where grounding shields are located within the connector.

Alternative connector technologies should be considered for driving longer distances across the backplane as well as reducing crosstalk within the connector. Surface mount (SMT) connectors and compression-fit connectors are two examples. Backplanes designed for these connector types can be further optimized to reduce overall via size, resulting in less coupling between vias and less signal loss through the vias.

Package to PCB Launch

The MGT pins are located on the periphery of the Virtex-II Pro X device. It is recommended that microstrip traces be used to route signals to and from MGT pins for a short distance (a few millimeters) before routing them through differential vias to lower signal layers. Differential stripline traces should be used for routing in inner layers.

Many of the design techniques used at the backplane connector can be used for optimizing the launch from the package onto the PCB line card or switch card. Signals should be routed on layers near the bottom of the board whenever possible. If signals are routed on the upper layers of the board, backdrilling, blind vias, or buried vias should be used to reduce stubs to less than 50 mil. Rectangular antipads should be utilized in a similar fashion as discussed in the previous section. Finally, pads should be removed from all unused signal layers.

XILINX®

Appendix F

Modifiable Attributes

Table F-1: **Default Attribute Values: GT10_CUSTOM**

Attribute	GT10_CUSTOM
ALIGN_COMMA_WORD	1, 2, 4
CHAN_BOND_64B66B_SV	FALSE, TRUE
CHAN_BOND_LIMIT	16, 0…31
CHAN_BOND_MODE	OFF, MASTER, SLAVE_1_HOP, SLAVE_2_HOPS
CHAN_BOND_ONE_SHOT	FALSE, TRUE
CHAN_BOND_SEQ_1_1	11'b00000000000, 11-bit binary
CHAN_BOND_SEQ_1_2	11'b00000000000, 11-bit binary
CHAN_BOND_SEQ_1_3	11'b00000000000, 11-bit binary
CHAN_BOND_SEQ_1_4	11'b00000000000, 11-bit binary
CHAN_BOND_SEQ_1_MASK	4'b0000, 4-bit binary
CHAN_BOND_SEQ_2_1	11'b00000000000, 11-bit binary
CHAN_BOND_SEQ_2_2	11'b00000000000, 11-bit binary
CHAN_BOND_SEQ_2_3	11'b00000000000, 11-bit binary
CHAN_BOND_SEQ_2_4	11'b00000000000, 11-bit binary
CHAN_BOND_SEQ_2_MASK	4'b0000, 4-bit binary
CHAN_BOND_SEQ_2_USE	FALSE, TRUE
CHAN_BOND_SEQ_LEN	1, 1, 2, 3, 4, 8
CLK_COR_8B10B_DE	FALSE, TRUE
CLK_COR_ADJ_MAX	UNUSED
CLK_COR_INSERT_IDLE_FLAG	UNUSED
CLK_COR_KEEP_IDLE	UNUSED
CLK_COR_MAX_LAT	36, 0…63
CLK_COR_MIN_LAT	28, 0…63
CLK_COR_REPEAT_WAIT	UNUSED

Table F-1: **Default Attribute Values: GT10_CUSTOM** ***(Continued)***

Attribute	GT10_CUSTOM
CLK_COR_SEQ_1_1	11'b00000000000, 11-bit binary
CLK_COR_SEQ_1_2	11'b00000000000, 11-bit binary
CLK_COR_SEQ_1_3	11'b00000000000, 11-bit binary
CLK_COR_SEQ_1_4	11'b00000000000, 11-bit binary
CLK_COR_SEQ_1_MASK	4'b0000, 4-bit binary
CLK_COR_SEQ_2_1	11'b00000000000, 11-bit binary
CLK_COR_SEQ_2_2	11'b00000000000, 11-bit binary
CLK_COR_SEQ_2_3	11'b00000000000, 11-bit binary
CLK_COR_SEQ_2_4	11'b00000000000, 11-bit binary
CLK_COR_SEQ_2_MASK	4'b0000, 4-bit binary
CLK_COR_SEQ_2_USE	FALSE, TRUE
CLK_COR_SEQ_DROP	FALSE, TRUE
CLK_COR_SEQ_LEN	1, 1, 2, 3, 4, 8
CLK_CORRECT_USE	TRUE, FALSE
COMMA_10B_MASK	10'b0001111111, 10-bit binary
DEC_MCOMMA_DETECT	TRUE, FALSE
DEC_PCOMMA_DETECT	TRUE, FALSE
DEC_VALID_COMMA_ONLY	TRUE, FALSE
LANE_ID	UNUSED
MCOMMA_10B_VALUE	10'b1010000011,10-bit binary
MCOMMA_DETECT	TRUE, FALSE
PCOMMA_10B_VALUE	10'b0101111100,10-bit binary
PCOMMA_DETECT	TRUE, FALSE
PMA_PWR_CNTRL	8'b11111111, 8-bit binary
PMA_SPEED	25_10, 25_20, 25_40, 28_10, 28_20, 28_40, 31_16, 31_32, 31_8
RX_BUFFER_USE	TRUE, FALSE
RX_LOS_INVALID_INCR	1, 2, 4, 8, 16, 32, 64, 128
RX_LOS_THRESHOLD	4, 8, 16, 32, 64, 128, 256, 512
RX_LOSS_OF_SYNC_FSM	TRUE, FALSE
SH_CNT_MAX	64, 1…255

Table F-1: **Default Attribute Values: GT10_CUSTOM *(Continued)***

Attribute	GT10_CUSTOM
SH_INVALID_CNT_MAX	16, 1…255
TX_BUFFER_USE	TRUE, FALSE

Table F-2: **Default Attribute Values: GT10_AURORA_1, GT10_AURORA_2, and GT10_AURORA_4**

Attribute	GT10_AURORA_1	GT10_AURORA_2	GT10_AURORA_4
ALIGN_COMMA_WORD	1, 2	1, 2	1, 2
CHAN_BOND_64B66B_SV	UNUSED	UNUSED	UNUSED
CHAN_BOND_LIMIT	16, 0…31	16, 0…31	16, 0…31
CHAN_BOND_MODE	OFF, MASTER, SLAVE_1_HOP, SLAVE_2_HOPS	OFF, MASTER, SLAVE_1_HOP, SLAVE_2_HOPS	OFF, MASTER, SLAVE_1_HOP, SLAVE_2_HOPS
CHAN_BOND_ONE_SHOT	FALSE, TRUE	FALSE, TRUE	FALSE, TRUE
CHAN_BOND_SEQ_1_1	11'b0010111110	11'b0010111110	11'b0010111110
CHAN_BOND_SEQ_1_2	11'b0010111110	11'b0010111110	11'b0010111110
CHAN_BOND_SEQ_1_3	11'b0010111110	11'b0010111110	11'b0010111110
CHAN_BOND_SEQ_1_4	11'b0010111110	11'b0010111110	11'b0010111110
CHAN_BOND_SEQ_1_MASK	4'b0000, 4-bit binary	4'b0000, 4-bit binary	4'b0000, 4-bit binary
CHAN_BOND_SEQ_2_1	11'b00000000000	11'b00000000000	11'b00000000000
CHAN_BOND_SEQ_2_2	11'b00000000000	11'b00000000000	11'b00000000000
CHAN_BOND_SEQ_2_3	11'b00000000000	11'b00000000000	11'b00000000000
CHAN_BOND_SEQ_2_4	11'b00000000000	11'b00000000000	11'b00000000000
CHAN_BOND_SEQ_2_MASK	4'b0000, 4-bit binary	4'b0000, 4-bit binary	4'b0000, 4-bit binary
CHAN_BOND_SEQ_2_USE	FALSE	FALSE	FALSE
CHAN_BOND_SEQ_LEN	1	1	1
CLK_COR_8B10B_DE	FALSE, TRUE	FALSE, TRUE	FALSE, TRUE
CLK_COR_ADJ_MAX	UNUSED	UNUSED	UNUSED
CLK_COR_INSERT_IDLE_FLAG	FALSE	FALSE	FALSE
CLK_COR_KEEP_IDLE	FALSE	FALSE	FALSE
CLK_COR_MAX_LAT	36, 0…63	36, 0…63	36, 0…63
CLK_COR_MIN_LAT	28, 0…63	28, 0…63	28, 0…63
CLK_COR_REPEAT_WAIT	8	8	8
CLK_COR_SEQ_1_1	11'b00111110111	11'b00111110111	11'b00111110111

Table F-2: **Default Attribute Values: GT10_AURORA_1, GT10_AURORA_2, and GT10_AURORA_4** ***(Continued)***

Attribute	GT10_AURORA_1	GT10_AURORA_2	GT10_AURORA_4
CLK_COR_SEQ_1_2	11'b00111110111	11'b00111110111	11'b00111110111
CLK_COR_SEQ_1_3	11'b00111110111	11'b00111110111	11'b00111110111
CLK_COR_SEQ_1_4	11'b00111110111	11'b00111110111	11'b00111110111
CLK_COR_SEQ_1_MASK	4'b0000, 4-bit binary	4'b0000, 4-bit binary	4'b0000, 4-bit binary
CLK_COR_SEQ_2_1	11'b00000000000	11'b00000000000	11'b00000000000
CLK_COR_SEQ_2_2	11'b00000000000	11'b00000000000	11'b00000000000
CLK_COR_SEQ_2_3	11'b00000000000	11'b00000000000	11'b00000000000
CLK_COR_SEQ_2_4	11'b00000000000	11'b00000000000	11'b00000000000
CLK_COR_SEQ_2_MASK	4'b0000, 4-bit binary	4'b0000, 4-bit binary	4'b0000, 4-bit binary
CLK_COR_SEQ_2_USE	FALSE	FALSE	FALSE
CLK_COR_SEQ_DROP	FALSE, TRUE	FALSE, TRUE	FALSE, TRUE
CLK_COR_SEQ_LEN	2	2	4
CLK_CORRECT_USE	TRUE	TRUE	TRUE
COMMA_10B_MASK	10'b1111111111	10'b1111111111	10'b1111111111
DEC_MCOMMA_DETECT	TRUE	TRUE	TRUE
DEC_PCOMMA_DETECT	TRUE	TRUE	TRUE
DEC_VALID_COMMA_ONLY	TRUE	TRUE	TRUE
LANE_ID	UNUSED	UNUSED	UNUSED
MCOMMA_10B_VALUE	10'b1100000101	10'b1100000101	10'b1100000101
MCOMMA_DETECT	TRUE	TRUE	TRUE
PCOMMA_10B_VALUE	10'b0011111010	10'b0011111010	10'b0011111010
PCOMMA_DETECT	TRUE	TRUE	TRUE
PMA_PWR_CNTRL	8'b11111111, 8-bit binary	8'b11111111, 8-bit binary	8'b11111111, 8-bit binary
PMA_SPEED	25_10	25_20	25_40
RX_BUFFER_USE	TRUE	TRUE	TRUE
RX_LOS_INVALID_INCR	1	1	1
RX_LOS_THRESHOLD	4	4	4
RX_LOSS_OF_SYNC_FSM	FALSE	FALSE	FALSE
SH_CNT_MAX	UNUSED	UNUSED	UNUSED
SH_INVALID_CNT_MAX	UNUSED	UNUSED	UNUSED
TX_BUFFER_USE	TRUE, FALSE	TRUE, FALSE	TRUE, FALSE

Table F-3: **Default Attribute Values: GT10_PCI_EXPRESS_1, GT10_PCI_EXPRESS_2, and GT10_PCI_EXPRESS_4**

Attribute	GT10_PCI_EXPRESS_1	GT10_PCI_EXPRESS_2	GT10_PCI_EXPRESS_4
ALIGN_COMMA_WORD	2, 1	2, 1	2, 1
CHAN_BOND_64B66B_SV	UNUSED	UNUSED	UNUSED
CHAN_BOND_LIMIT	16, 0…31	16, 0…31	16, 0…31
CHAN_BOND_MODE	OFF, MASTER, SLAVE_1_HOP, SLAVE_2_HOPS	OFF, MASTER, SLAVE_1_HOP, SLAVE_2_HOPS	OFF, MASTER, SLAVE_1_HOP, SLAVE_2_HOPS
CHAN_BOND_ONE_SHOT	FALSE, TRUE	FALSE, TRUE	FALSE, TRUE
CHAN_BOND_SEQ_1_1	11'b00110111100	11'b00110111100	11'b00110111100
CHAN_BOND_SEQ_1_2	11'b00000000000	11'b00000000000	11'b00000000000
CHAN_BOND_SEQ_1_3	11'b00001001010	11'b00001001010	11'b00001001010
CHAN_BOND_SEQ_1_4	11'b00001001010	11'b00001001010	11'b00001001010
CHAN_BOND_SEQ_1_MASK	4'b0000, 4-bit binary	4'b0000, 4-bit binary	4'b0000, 4-bit binary
CHAN_BOND_SEQ_2_1	11'b00110111100	11'b00110111100	11'b00110111100
CHAN_BOND_SEQ_2_2	11'b00000000000	11'b00000000000	11'b00000000000
CHAN_BOND_SEQ_2_3	11'b00001000101	11'b00001000101	11'b00001000101
CHAN_BOND_SEQ_2_4	11'b00001000101	11'b00001000101	11'b00001000101
CHAN_BOND_SEQ_2_MASK	4'b0000, 4-bit binary	4'b0000, 4-bit binary	4'b0000, 4-bit binary
CHAN_BOND_SEQ_2_USE	FALSE, TRUE	FALSE, TRUE	FALSE, TRUE
CHAN_BOND_SEQ_LEN	2, 1, 2, 3, 4	2, 1, 2, 3, 4	2, 1, 2, 3, 4
CLK_COR_8B10B_DE	FALSE, TRUE	FALSE, TRUE	FALSE, TRUE
CLK_COR_ADJ_MAX	UNUSED	UNUSED	UNUSED
CLK_COR_INSERT_IDLE_FLAG	UNUSED	UNUSED	UNUSED
CLK_COR_KEEP_IDLE	UNUSED	UNUSED	UNUSED
CLK_COR_MAX_LAT	36, 0…63	36, 0…63	36, 0…63
CLK_COR_MIN_LAT	28, 0…63	28, 0…63	28, 0…63
CLK_COR_REPEAT_WAIT	UNUSED	UNUSED	UNUSED
CLK_COR_SEQ_1_1	11'b00100011100	11'b00100011100	11'b00100011100
CLK_COR_SEQ_1_2	11'b00000000000	11'b00000000000	11'b00000000000
CLK_COR_SEQ_1_3	11'b00000000000	11'b00000000000	11'b00000000000
CLK_COR_SEQ_1_4	11'b00000000000	11'b00000000000	11'b00000000000
CLK_COR_SEQ_1_MASK	4'b0000, 4-bit binary	4'b0000, 4-bit binary	4'b0000, 4-bit binary

Table F-3: **Default Attribute Values: GT10_PCI_EXPRESS_1, GT10_PCI_EXPRESS_2, and GT10_PCI_EXPRESS_4 *(Continued)***

Attribute	GT10_PCI_EXPRESS_1	GT10_PCI_EXPRESS_2	GT10_PCI_EXPRESS_4
CLK_COR_SEQ_2_1	11'b00000000000	11'b00000000000	11'b00000000000
CLK_COR_SEQ_2_2	11'b00000000000	11'b00000000000	11'b00000000000
CLK_COR_SEQ_2_3	11'b00000000000	11'b00000000000	11'b00000000000
CLK_COR_SEQ_2_4	11'b00000000000	11'b00000000000	11'b00000000000
CLK_COR_SEQ_2_MASK	4'b0000, 4-bit binary	4'b0000, 4-bit binary	4'b0000, 4-bit binary
CLK_COR_SEQ_2_USE	FALSE, TRUE	FALSE, TRUE	FALSE, TRUE
CLK_COR_SEQ_DROP	FALSE, TRUE	FALSE, TRUE	FALSE, TRUE
CLK_COR_SEQ_LEN	2, 1, 2, 3, 4	2, 1, 2, 3, 4	2, 1, 2, 3, 4
CLK_CORRECT_USE	TRUE, FALSE	TRUE, FALSE	TRUE, FALSE
COMMA_10B_MASK	10'b0001111111	10'b0001111111	10'b0001111111
DEC_MCOMMA_DETECT	TRUE, FALSE	TRUE, FALSE	TRUE, FALSE
DEC_PCOMMA_DETECT	TRUE, FALSE	TRUE, FALSE	TRUE, FALSE
DEC_VALID_COMMA_ONLY	TRUE, FALSE	TRUE, FALSE	TRUE, FALSE
LANE_ID	UNUSED	UNUSED	UNUSED
MCOMMA_10B_VALUE	10'b1010000011	10'b1010000011	10'b1010000011
MCOMMA_DETECT	TRUE	TRUE	TRUE
PCOMMA_10B_VALUE	10'b0101111100	10'b0101111100	10'b0101111100
PCOMMA_DETECT	TRUE	TRUE	TRUE
PMA_PWR_CNTRL	8'b11111111, 8-bit binary	8'b11111111, 8-bit binary	8'b11111111, 8-bit binary
PMA_SPEED	28_10	28_20	28_40
RX_BUFFER_USE	TRUE, FALSE	TRUE, FALSE	TRUE, FALSE
RX_LOS_INVALID_INCR	1, 2, 4, 8, 16, 32, 64, 128	1, 2, 4, 8, 16, 32, 64, 128	1, 2, 4, 8, 16, 32, 64, 128
RX_LOS_THRESHOLD	4, 8, 16, 32, 64, 128, 256, 512	4, 8, 16, 32, 64, 128, 256, 512	4, 8, 16, 32, 64, 128, 256, 512
RX_LOSS_OF_SYNC_FSM	TRUE, FALSE	TRUE, FALSE	TRUE, FALSE
SH_CNT_MAX	UNUSED	UNUSED	UNUSED
SH_INVALID_CNT_MAX	UNUSED	UNUSED	UNUSED
TX_BUFFER_USE	TRUE, FALSE	TRUE, FALSE	TRUE, FALSE

Table F-4: **Default Attribute Values: GT10_INFINIBAND_1, GT10_INFINIBAND_2, and GT10_INFINIBAND_4**

Attribute	GT10_INFINIBAND_1	GT10_INFINIBAND_2	GT10_INFINIBAND_4
ALIGN_COMMA_WORD	2, 1	2, 1	2, 1
CHAN_BOND_64B66B_SV	UNUSED	UNUSED	UNUSED
CHAN_BOND_LIMIT	16, 0…31	16, 0…31	16, 0…31
CHAN_BOND_MODE	OFF, MASTER, SLAVE_1_HOP, SLAVE_2_HOPS	OFF, MASTER, SLAVE_1_HOP, SLAVE_2_HOPS	OFF, MASTER, SLAVE_1_HOP, SLAVE_2_HOPS
CHAN_BOND_ONE_SHOT	FALSE, TRUE	FALSE, TRUE	FALSE, TRUE
CHAN_BOND_SEQ_1_1	11'b00110111100	11'b00110111100	11'b00110111100
CHAN_BOND_SEQ_1_2	LANE_ID	LANE_ID	LANE_ID
CHAN_BOND_SEQ_1_3	11'b00001001010	11'b00001001010	11'b00001001010
CHAN_BOND_SEQ_1_4	11'b00001001010	11'b00001001010	11'b00001001010
CHAN_BOND_SEQ_1_MASK	4'b0000, 4-bit binary	4'b0000, 4-bit binary	4'b0000, 4-bit binary
CHAN_BOND_SEQ_2_1	11'b00110111100	11'b00110111100	11'b00110111100
CHAN_BOND_SEQ_2_2	LANE_ID	LANE_ID	LANE_ID
CHAN_BOND_SEQ_2_3	11'b00001000101	11'b00001000101	11'b00001000101
CHAN_BOND_SEQ_2_4	11'b00001000101	11'b00001000101	11'b00001000101
CHAN_BOND_SEQ_2_MASK	4'b0000, 4-bit binary	4'b0000, 4-bit binary	4'b0000, 4-bit binary
CHAN_BOND_SEQ_2_USE	TRUE	TRUE	TRUE
CHAN_BOND_SEQ_LEN	2, 1, 2, 3, 4	2, 1, 2, 3, 4	2, 1, 2, 3, 4
CLK_COR_8B10B_DE	FALSE, TRUE	FALSE, TRUE	FALSE, TRUE
CLK_COR_ADJ_MAX	UNUSED	UNUSED	UNUSED
CLK_COR_INSERT_IDLE_FLAG	UNUSED	UNUSED	UNUSED
CLK_COR_KEEP_IDLE	UNUSED	UNUSED	UNUSED
CLK_COR_MAX_LAT	36, 0…63	36, 0…63	36, 0…63
CLK_COR_MIN_LAT	28, 0…63	28, 0…63	28, 0…63
CLK_COR_REPEAT_WAIT	UNUSED	UNUSED	UNUSED
CLK_COR_SEQ_1_1	11'b00100011100	11'b00100011100	11'b00100011100
CLK_COR_SEQ_1_2	11'b00000000000	11'b00000000000	11'b00000000000
CLK_COR_SEQ_1_3	11'b00000000000	11'b00000000000	11'b00000000000
CLK_COR_SEQ_1_4	11'b00000000000	11'b00000000000	11'b00000000000
CLK_COR_SEQ_1_MASK	4'b0000, 4-bit binary	4'b0000, 4-bit binary	4'b0000, 4-bit binary
CLK_COR_SEQ_2_1	11'b00000000000	11'b00000000000	11'b00000000000

Table F-4: **Default Attribute Values: GT10_INFINIBAND_1, GT10_INFINIBAND_2, and GT10_INFINIBAND_4 *(Continued)***

Attribute	GT10_INFINIBAND_1	GT10_INFINIBAND_2	GT10_INFINIBAND_4
CLK_COR_SEQ_2_2	11'b00000000000	11'b00000000000	11'b00000000000
CLK_COR_SEQ_2_3	11'b00000000000	11'b00000000000	11'b00000000000
CLK_COR_SEQ_2_4	11'b00000000000	11'b00000000000	11'b00000000000
CLK_COR_SEQ_2_MASK	4'b0000, 4-bit binary	4'b0000, 4-bit binary	4'b0000, 4-bit binary
CLK_COR_SEQ_2_USE	FALSE, TRUE	FALSE, TRUE	FALSE, TRUE
CLK_COR_SEQ_DROP	FALSE, TRUE	FALSE, TRUE	FALSE, TRUE
CLK_COR_SEQ_LEN	2, 1, 2, 3, 4	2, 1, 2, 3, 4	2, 1, 2, 3, 4
CLK_CORRECT_USE	TRUE, FALSE	TRUE, FALSE	TRUE, FALSE
COMMA_10B_MASK	10'b0001111111	10'b0001111111	10'b0001111111
DEC_MCOMMA_DETECT	TRUE, FALSE	TRUE, FALSE	TRUE, FALSE
DEC_PCOMMA_DETECT	TRUE, FALSE	TRUE, FALSE	TRUE, FALSE
DEC_VALID_COMMA_ONLY	TRUE, FALSE	TRUE, FALSE	TRUE, FALSE
LANE_ID	11'b00000000000, 11-bit binary	11'b00000000000, 11-bit binary	11'b00000000000, 11-bit binary
MCOMMA_10B_VALUE	10'b1010000011	10'b1010000011	10'b1010000011
MCOMMA_DETECT	TRUE	TRUE	TRUE
PCOMMA_10B_VALUE	10'b0101111100	10'b0101111100	10'b0101111100
PCOMMA_DETECT	TRUE	TRUE	TRUE
PMA_PWR_CNTRL	8'b11111111, 8-bit binary	8'b11111111, 8-bit binary	8'b11111111, 8-bit binary
PMA_SPEED	28_10	28_20	28_40
RX_BUFFER_USE	TRUE, FALSE	TRUE, FALSE	TRUE, FALSE
RX_LOS_INVALID_INCR	1, 2, 4, 8, 16, 32, 64, 128	1, 2, 4, 8, 16, 32, 64, 128	1, 2, 4, 8, 16, 32, 64, 128
RX_LOS_THRESHOLD	4, 8, 16, 32, 64, 128, 256, 512	4, 8, 16, 32, 64, 128, 256, 512	4, 8, 16, 32, 64, 128, 256, 512
RX_LOSS_OF_SYNC_FSM	TRUE, FALSE	TRUE, FALSE	TRUE, FALSE
SH_CNT_MAX	UNUSED	UNUSED	UNUSED
SH_INVALID_CNT_MAX	UNUSED	UNUSED	UNUSED
TX_BUFFER_USE	TRUE, FALSE	TRUE, FALSE	TRUE, FALSE

Table F-5: **Default Attribute Values: GT10_XAUI_1, GT10_XAUI_2, and GT10_XAUI_4**

Attribute	GT10_XAUI_1	GT10_XAUI_2	GT10_XAUI_4
ALIGN_COMMA_WORD	2, 1	2, 1	2, 1
CHAN_BOND_64B66B_SV	UNUSED	UNUSED	UNUSED
CHAN_BOND_LIMIT	16, 0…31	16, 0…31	16, 0…31
CHAN_BOND_MODE	OFF, MASTER, SLAVE_1_HOP, SLAVE_2_HOPS	OFF, MASTER, SLAVE_1_HOP, SLAVE_2_HOPS	OFF, MASTER, SLAVE_1_HOP, SLAVE_2_HOPS
CHAN_BOND_ONE_SHOT	FALSE, TRUE	FALSE, TRUE	FALSE, TRUE
CHAN_BOND_SEQ_1_1	11'b00101111100	11'b00101111100	11'b00101111100
CHAN_BOND_SEQ_1_2	11'b00000000000	11'b00000000000	11'b00000000000
CHAN_BOND_SEQ_1_3	11'b00000000000	11'b00000000000	11'b00000000000
CHAN_BOND_SEQ_1_4	11'b00000000000	11'b00000000000	11'b00000000000
CHAN_BOND_SEQ_1_MASK	4'b0000, 4-bit binary	4'b0000, 4-bit binary	4'b0000, 4-bit binary
CHAN_BOND_SEQ_2_1	11'b00000000000	11'b00000000000	11'b00000000000
CHAN_BOND_SEQ_2_2	11'b00000000000	11'b00000000000	11'b00000000000
CHAN_BOND_SEQ_2_3	11'b00000000000	11'b00000000000	11'b00000000000
CHAN_BOND_SEQ_2_4	11'b00000000000	11'b00000000000	11'b00000000000
CHAN_BOND_SEQ_2_MASK	4'b0000, 4-bit binary	4'b0000, 4-bit binary	4'b0000, 4-bit binary
CHAN_BOND_SEQ_2_USE	FALSE, TRUE	FALSE, TRUE	FALSE, TRUE
CHAN_BOND_SEQ_LEN	2, 1, 2, 3, 4	2, 1, 2, 3, 4	2, 1, 2, 3, 4
CLK_COR_8B10B_DE	FALSE, TRUE	FALSE, TRUE	FALSE, TRUE
CLK_COR_ADJ_MAX	UNUSED	UNUSED	UNUSED
CLK_COR_INSERT_IDLE_FLAG	UNUSED	UNUSED	UNUSED
CLK_COR_KEEP_IDLE	UNUSED	UNUSED	UNUSED
CLK_COR_MAX_LAT	36, 0…63	36, 0…63	36, 0…63
CLK_COR_MIN_LAT	28, 0…63	28, 0…63	28, 0…63
CLK_COR_REPEAT_WAIT	UNUSED	UNUSED	UNUSED
CLK_COR_SEQ_1_1	11'b00100011100	11'b00100011100	11'b00100011100
CLK_COR_SEQ_1_2	11'b00000000000	11'b00000000000	11'b00000000000
CLK_COR_SEQ_1_3	11'b00000000000	11'b00000000000	11'b00000000000
CLK_COR_SEQ_1_4	11'b00000000000	11'b00000000000	11'b00000000000
CLK_COR_SEQ_1_MASK	4'b0000, 4-bit binary	4'b0000, 4-bit binary	4'b0000, 4-bit binary

Table F-5: **Default Attribute Values: GT10_XAUI_1, GT10_XAUI_2, and GT10_XAUI_4 *(Continued)***

Attribute	GT10_XAUI_1	GT10_XAUI_2	GT10_XAUI_4
CLK_COR_SEQ_2_1	11'b00000000000	11'b00000000000	11'b00000000000
CLK_COR_SEQ_2_2	11'b00000000000	11'b00000000000	11'b00000000000
CLK_COR_SEQ_2_3	11'b00000000000	11'b00000000000	11'b00000000000
CLK_COR_SEQ_2_4	11'b00000000000	11'b00000000000	11'b00000000000
CLK_COR_SEQ_2_MASK	4'b0000, 4-bit binary	4'b0000, 4-bit binary	4'b0000, 4-bit binary
CLK_COR_SEQ_2_USE	FALSE, TRUE	FALSE, TRUE	FALSE, TRUE
CLK_COR_SEQ_DROP	FALSE, TRUE	FALSE, TRUE	FALSE, TRUE
CLK_COR_SEQ_LEN	2, 1, 2, 3, 4	2, 1, 2, 3, 4	2, 1, 2, 3, 4
CLK_CORRECT_USE	TRUE, FALSE	TRUE, FALSE	TRUE, FALSE
COMMA_10B_MASK	10'b0001111111	10'b0001111111	10'b0001111111
DEC_MCOMMA_DETECT	TRUE, FALSE	TRUE, FALSE	TRUE, FALSE
DEC_PCOMMA_DETECT	TRUE, FALSE	TRUE, FALSE	TRUE, FALSE
DEC_VALID_COMMA_ONLY	TRUE, FALSE	TRUE, FALSE	TRUE, FALSE
LANE_ID	UNUSED	UNUSED	UNUSED
MCOMMA_10B_VALUE	10'b1010000011	10'b1010000011	10'b1010000011
MCOMMA_DETECT	TRUE	TRUE	TRUE
PCOMMA_10B_VALUE	10'b0101111100	10'b0101111100	10'b0101111100
PCOMMA_DETECT	TRUE	TRUE	TRUE
PMA_PWR_CNTRL	8'b11111111, 8-bit binary	8'b11111111, 8-bit binary	8'b11111111, 8-bit binary
PMA_SPEED	25_10	25_20	25_40
RX_BUFFER_USE	TRUE, FALSE	TRUE, FALSE	TRUE, FALSE
RX_LOS_INVALID_INCR	1, 2, 4, 8, 16, 32, 64, 128	1, 2, 4, 8, 16, 32, 64, 128	1, 2, 4, 8, 16, 32, 64, 128
RX_LOS_THRESHOLD	4, 8, 16, 32, 64, 128, 256, 512	4, 8, 16, 32, 64, 128, 256, 512	4, 8, 16, 32, 64, 128, 256, 512
RX_LOSS_OF_SYNC_FSM	TRUE, FALSE	TRUE, FALSE	TRUE, FALSE
SH_CNT_MAX	UNUSED	UNUSED	UNUSED
SH_INVALID_CNT_MAX	UNUSED	UNUSED	UNUSED
TX_BUFFER_USE	TRUE, FALSE	TRUE, FALSE	TRUE, FALSE

Table F-6: **Default Attribute Values: GT10_OC48_1, GT10_OC48_2, and GT10_OC48_4**

Attribute	GT10_OC48_1	GT10_OC48_2	GT10_OC48_4
ALIGN_COMMA_WORD	1, 2	1, 2	1, 2
CHAN_BOND_64B66B_SV	UNUSED	UNUSED	UNUSED
CHAN_BOND_LIMIT	UNUSED	UNUSED	UNUSED
CHAN_BOND_MODE	UNUSED	UNUSED	UNUSED
CHAN_BOND_ONE_SHOT	UNUSED	UNUSED	UNUSED
CHAN_BOND_SEQ_1_1	UNUSED	UNUSED	UNUSED
CHAN_BOND_SEQ_1_2	UNUSED	UNUSED	UNUSED
CHAN_BOND_SEQ_1_3	UNUSED	UNUSED	UNUSED
CHAN_BOND_SEQ_1_4	UNUSED	UNUSED	UNUSED
CHAN_BOND_SEQ_1_MASK	UNUSED	UNUSED	UNUSED
CHAN_BOND_SEQ_2_1	UNUSED	UNUSED	UNUSED
CHAN_BOND_SEQ_2_2	UNUSED	UNUSED	UNUSED
CHAN_BOND_SEQ_2_3	UNUSED	UNUSED	UNUSED
CHAN_BOND_SEQ_2_4	UNUSED	UNUSED	UNUSED
CHAN_BOND_SEQ_2_MASK	UNUSED	UNUSED	UNUSED
CHAN_BOND_SEQ_2_USE	UNUSED	UNUSED	UNUSED
CHAN_BOND_SEQ_LEN	UNUSED	UNUSED	UNUSED
CLK_COR_8B10B_DE	UNUSED	UNUSED	UNUSED
CLK_COR_ADJ_MAX	UNUSED	UNUSED	UNUSED
CLK_COR_INSERT_IDLE_FLAG	UNUSED	UNUSED	UNUSED
CLK_COR_KEEP_IDLE	UNUSED	UNUSED	UNUSED
CLK_COR_MAX_LAT	UNUSED	UNUSED	UNUSED
CLK_COR_MIN_LAT	UNUSED	UNUSED	UNUSED
CLK_COR_REPEAT_WAIT	UNUSED	UNUSED	UNUSED
CLK_COR_SEQ_1_1	UNUSED	UNUSED	UNUSED
CLK_COR_SEQ_1_2	UNUSED	UNUSED	UNUSED
CLK_COR_SEQ_1_3	UNUSED	UNUSED	UNUSED
CLK_COR_SEQ_1_4	UNUSED	UNUSED	UNUSED
CLK_COR_SEQ_1_MASK	UNUSED	UNUSED	UNUSED
CLK_COR_SEQ_2_1	UNUSED	UNUSED	UNUSED
CLK_COR_SEQ_2_2	UNUSED	UNUSED	UNUSED

Table F-6: **Default Attribute Values: GT10_OC48_1, GT10_OC48_2, and GT10_OC48_4 *(Continued)***

Attribute	GT10_OC48_1	GT10_OC48_2	GT10_OC48_4
CLK_COR_SEQ_2_3	UNUSED	UNUSED	UNUSED
CLK_COR_SEQ_2_4	UNUSED	UNUSED	UNUSED
CLK_COR_SEQ_2_MASK	UNUSED	UNUSED	UNUSED
CLK_COR_SEQ_2_USE	UNUSED	UNUSED	UNUSED
CLK_COR_SEQ_DROP	UNUSED	UNUSED	UNUSED
CLK_COR_SEQ_LEN	UNUSED	UNUSED	UNUSED
CLK_CORRECT_USE	UNUSED	UNUSED	UNUSED
COMMA_10B_MASK	10'b0011111111	10'b0011111111	10'b0011111111
DEC_MCOMMA_DETECT	TRUE, FALSE	TRUE, FALSE	TRUE, FALSE
DEC_PCOMMA_DETECT	TRUE, FALSE	TRUE, FALSE	TRUE, FALSE
DEC_VALID_COMMA_ONLY	UNUSED	UNUSED	UNUSED
LANE_ID	UNUSED	UNUSED	UNUSED
MCOMMA_10B_VALUE	10'b0010101010, 10-bit binary	10'b0010101010, 10-bit binary	10'b0010101010, 10-bit binary
MCOMMA_DETECT	TRUE, FALSE	TRUE, FALSE	TRUE, FALSE
PCOMMA_10B_VALUE	10'b0010101010, 10-bit binary	10'b0010101010, 10-bit binary	10'b0010101010, 10-bit binary
PCOMMA_DETECT	TRUE, FALSE	TRUE, FALSE	TRUE, FALSE
PMA_PWR_CNTRL	8'b11111111, 8-bit binary	8'b11111111, 8-bit binary	8'b11111111, 8-bit binary
PMA_SPEED	31_8	31_16	31_32
RX_BUFFER_USE	TRUE, FALSE	TRUE, FALSE	TRUE, FALSE
RX_LOS_INVALID_INCR	1, 2, 4, 8, 16, 32, 64, 128	1, 2, 4, 8, 16, 32, 64, 128	1, 2, 4, 8, 16, 32, 64, 128
RX_LOS_THRESHOLD	4, 8, 16, 32, 64, 128, 256, 512	4, 8, 16, 32, 64, 128, 256, 512	4, 8, 16, 32, 64, 128, 256, 512
RX_LOSS_OF_SYNC_FSM	TRUE, FALSE	TRUE, FALSE	TRUE, FALSE
SH_CNT_MAX	UNUSED	UNUSED	UNUSED
SH_INVALID_CNT_MAX	UNUSED	UNUSED	UNUSED
TX_BUFFER_USE	TRUE, FALSE	TRUE, FALSE	TRUE, FALSE

XILINX®

Appendix G

Related Online Documents

The documents described in this Appendix are accessible on the Xilinx website at www.xilinx.com. Document links shown in blue are clickable in this PDF file, providing easy access to the most current revision of each document.

Application Notes

XAPP752: Virtex-II Pro X OC-48 Jitter Compliance Test Results

This application note explains the results of the Virtex-II Pro X MGT tests against the ITU/Bellcore GR-235 Specification for Jitter Tolerance, Jitter Transfer, and Jitter Generation. The ITU/Bellcore GR-235 Specification has three jitter specifications that a transceiver must meet to claim OC-48 jitter compliance. This compliance ensures that the transceiver interoperates with any other transceiver that also meets these jitter specifications and controls the accumulation of timing jitter within the connection to acceptable levels.

XAPP762: RocketIO X Bit-Error Rate Tester Reference Design

This application note describes the implementation of a RocketIO™ X bit-error rate tester (XBERT) reference design. The reference design generates and verifies non-encoded high-speed serial data on one or multiple point-to-point links (2.5 Gb/s to 6.25 Gb/s) between RocketIO X multi-gigabit transceiver (MGT) ports, embedded within a single Virtex-II Pro™ X FPGA. This high-speed serial data is constructed in FPGA fabric using a pseudo-random bit sequence (PRBS) pattern, a clock pattern, or a user-defined pattern. The reference design provides access to the PMA attribute programming bus on the RocketIO X MGT, which enables real-time control of PMA features, such as the TX output swing, TX pre-emphasis, and RX equalization. The reference design utilizes the UltraController™ embedded processor—a lightweight PowerPC™ microcontroller solution. The embedded PPC405 processor transfers control and status between the XBERT module and the UART core through the General-Purpose Input/Output (GPIO) interface of the UltraController block, and enables a user interface through the software and an external RS-232 serial port. The reference design is built using the Embedded Development Kit (EDK), and it can be easily modified or extended.

XAPP767: RocketIO X Transceiver Clock Mode Switcher for Virtex-II Pro X FPGAs

RocketIO™ X Multi-Gigabit Transceivers (MGTs) operating at 6.25 Gb/s offer great flexibility, providing programmability of serial rates, datapath widths, encoding schemes, and reference clock to serial rate ratios (multipliers). Most of these options are modified by

the PMA_SPEED attribute. Some applications require multiple rates supported by a single MGT, requiring the mode of the MGT to be changed dynamically. The mode can be changed via the PMA attribute bus using multiple read-modify-write operations. This application note demonstrates a way to switch clocking modes without knowledge of a PMA attribute bus interface.

Characterization Reports

RPT007: RocketIO™ Transceiver Characterization Report for Virtex-II Pro X FPGAs

This report discusses the performance of the RocketIO Transceiver for the Virtex-II Pro X FPGAs. To access this report, go to:

http://www.xilinx.com/xlnx/xweb/xil_publications_index.jsp?category=Reports

Index

Numerics

A

B

C

D

R

S

T

U

V