7 Series FPGAs GTX/GTH Transceivers

User Guide

UG476 (v1.9.1) April 22, 2013

XILINX®

Notice of Disclaimer
The information disclosed to you hereunder (the “Materials”) is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of the Limited Warranties which can be viewed at http://www.xilinx.com/warranty.htm; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in Critical Applications: http://www.xilinx.com/warranty.htm#critapps.

Automotive Applications Disclaimer
XILINX PRODUCTS ARE NOT DESIGNED OR INTENDED TO BE FAIL-SAFE, OR FOR USE IN ANY APPLICATION REQUIRING FAIL-SAFE PERFORMANCE, SUCH AS APPLICATIONS RELATED TO: (I) THE DEPLOYMENT OF AIRBAGS, (II) CONTROL OF A VEHICLE, UNLESS THERE IS A FAIL-SAFE OR REDUNDANCY FEATURE (WHICH DOES NOT INCLUDE USE OF SOFTWARE IN THE XILINX DEVICE TO IMPLEMENT THE REDUNDANCY) AND A WARNING SIGNAL UPON FAILURE TO THE OPERATOR, OR (III) USES THAT COULD LEAD TO DEATH OR PERSONAL INJURY. CUSTOMER ASSUMES THE SOLE RISK AND LIABILITY OF ANY USE OF XILINX PRODUCTS IN SUCH APPLICATIONS.

© Copyright 2011–2013 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, ISE, Kintex, Spartan, Virtex, Vivado, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. CPRI is a trademark of Siemens AG. PCI, PCIe and PCI Express are trademarks of PCI-SIG and used under license. All other trademarks are the property of their respective owners.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
03/01/11	1.0	Initial Xilinx release.
03/28/11	1.1	Chapter 1, Removed Table 1-4: GTX Transceiver Channels by Device/Package (Kintex-7 FPGAs) and added link to UG475: *7 Series FPGAs Packaging and Pinout Specifications*. Updated Table B-1.

Date	Version	Revision
07/08/11	1.2	Chapter 1, updated PCS and PMA features in Table 1-1. Chapter 2, revised ODIV2 attribute in Table 2-1 and removed REFCLK_CTRL from Table 2-2. Revised Reference Clock Selection and Distribution. Updated line rate and lock range in Channel PLL. Updated D factor in Table 2-8. Revised description of CPLLLOCKDETCLK in Table 2-9. Renamed CPLL_RXOUT_DIV to RXOUT_DIV and CPLL_TXOUT_DIV to TXOUT_DIV and updated their descriptions in Table 2-10. Updated line rate in Quad PLL. Revised VCO in Figure 2-11. Updated N valid setting and added D factor to Table 2-13. Updated QPLLLOCKDETCLK description in Table 2-14. Updated QPLL_CFG description in Table 2-15. Added RXOUT_DIV and TXOUT_DIV attributes to Table 2-15. Added CFGRESET and PCSRSVDOUT ports to Table 2-24. Corrected GTTXRESET name in Figure 2-16. Revised PLL Power Down. Chapter 3, updated line rate in TXUSRCLK and TXUSRCLK2 Generation. Added Using TXOUTCLK to Drive the TX Interface. Deleted TXRUNDISP[7:0] port from Table 3-7. Added RX to and updated description of GEARBOX_MODE in Table 3-9. Added Enabling the TX Gearbox, TX Gearbox Bit and Byte Ordering, TX Gearbox Operating Modes, External Sequence Counter Operating Mode, Internal Sequence Counter Operating Mode (GTX Transceiver Only), Table 3-10, and Table 3-11. Updated TXPHALIGNDONE description in Table 3-17. Updated Figure 3-19 and its relevant notes in Using TX Buffer Bypass in Single-Lane Auto Mode (GTX Transceiver Only). Added Using TX Buffer Bypass in Multi-Lane Manual Mode (GTX and GTH Transceivers). Updated TX Polarity Control. Updated Figure 3-28. Renamed CPLL_TXOUT_DIV to TXOUT_DIV in Serial Clock Divider, Table 3-25, and Table 3-26. Added TXDLYBYPASS to Table 3-26. Changed TXPOSTCURSOR range in Figure 3-29. Chapter 4, updated programmable voltage values in Table 4-2. Added GTX and GTH Use Modes—RX Termination. Updated RXOOBRESET and added RXELECIDLEMODE[1:0] to Table 4-7. Updated Figure 4-22. Renamed CPLL_RXOUT_DIV to RXOUT_DIV in Table 4-22. Updated bullets in Parallel Clock Divider and Selector. Added RXDLYBYPASS toTable 4-23. Renamed CPLL_RXOUT_DIV to RXOUT_DIV in Table 4-23 and Table 4-24. Added Eye Scan Architecture, Figure 4-25, Figure 4-26, and Figure 4-27. Added Ports and Attributes and Table 4-25, and Table 4-26. Updated Manual Alignment and Figure 4-35, and added Figure 4-36. Updated RXSLIDE description in Table 4-31. Updated description of SHOW_REALIGN_COMMA, RXSLIDE_MODE, and RXSLIDE_AUTO_WAI T, and renamed RXRECCLK to RXOUTCLK and SHOW_ALIGN_COMMA to SHOW_REALIGN_COMMA in Table 4-32. Revised RX Running Disparity. Replaced RX8B10BEN description in and deleted RXRUNDISP[7:0} from Table 4-33. Added RX CDR to Figure 4-39. Revised RXPHALIGNDONE description in Table 4-35. Revised Using RX Buffer Bypass in Single-Lane Auto Mode (GTX Transceiver Only) and updated Figure 4-40. Added Using RX Buffer Bypass in Multi-Lane Manual Mode (GTX and GTH Transceivers). Added RX CDR to Figure 4-48. Revised descriptions of CLK_COR_MAX_LAT, CLK_COR_MIN_LAT, and CLK_COR_SEQ_LEN in Table 4-43. Modified Using RX Clock Correction, added Enabling Clock Correction, modified Setting RX Elastic Buffer Limits and renamed CLK_COR_ADJ_LEN to CLK_COR_SEQ_LEN, added Setting Clock Correction Sequences, Clock Correction Options, and Monitoring Clock Correction. Modified RXCHBONDLEVEL description in Table 4-46. Added Using RX Channel Bonding, Enabling Channel Bonding. Added Setting Channel Bonding Sequences. Added Setting the Maximum Skew.

Date	Version	Revision
07/08/11	1.2 *(Continued)*	Added Precedence between Channel Bonding and Clock Correction. Revised description of RXGEARBOXSLIP in Table 4-48. Replaced description of GEARBOX_MODE in Table 4-49. Added Enabling the RX Gearbox. Added RX Gearbox Operating Modes. Added RX Gearbox Block Synchronization. Renamed to RXRECCLK to RXOUTCLK in RXUSRCLK and RXUSRCLK2 Generation. Updated description of RX_INT_DATAWIDTH in Table 4-54. Chapter 5, added Figure 5-2. Added Analog Power Supply Pins, Table 5-2, and Table 5-3. Updated Figure 5-9. Revised Unused Reference Clocks. Deleted LVDS section. Revised Printed Circuit Board and added Table 5-4. Added PCB Design Checklist and Table 5-5. Chapter 6, added Gen3 to RX Buffer in Table 6-1. Added TXCHARDISPMODE[0] to Table 6-2. Updated description of TXDEEMPH and RXELECIDLE in Table 6-2. Updated PCI Express Use Mode and Gen3 for RXBUF_EN and RX_XCLK_SEL in Table 6-4, added PIPE Control Signal and Table 6-5. Updated Reference Clock and Table 6-6, and added Table 6-7. Modified Parallel Clock (PCLK), added Figure 6-1, added introductory paragraph to and modified Figure 6-2. Revised Rate Change between Gen1 and Gen2 Speeds. Updated Figure 6-6. Revised Using DRP During Rate Change to Enter or Exit Gen3 Speed. Updated Gen3 for RXBUF_EN and RX_XCLK_SEL in Table 6-8. Updated PCI Express Channel Bonding, and added Binary-Tree Example. Added XAUI Use Model and Table 6-11 through Table 6-14. Appendix A, Placement Information by Package, updated content. Appendix B, Placement Information by Device, expanded Table B-1. Appendix D, DRP Address Map of the GTX/GTH Transceiver, removed CPLL_ from CPLL_RXOUT_DIV and CPLL_TXOUT_DIV in Table D-2. Renamed CPLL_TXOUT_DIV to TXOUT_DIV and CPLL_RXOUT_DIV to RXOUT_DIV throughout. Removed CPLL and CPLL_ prefix throughout. Renamed PLL to CPLL throughout.
07/28/11	1.2.1	Reformatted Table 3-17, Table 3-30, and Table 4-10.
11/16/11	1.3	Appendix A, Placement Information by Package, added Virtex-7 FPGA packages. Appendix B, Placement Information by Device, added Table B-2.
04/04/12	1.4	Added GTH transceivers throughout. Chapter 1: Updated Table 1-1. Chapter 2: Updated description of O and ODIV2 ports in Table 2-1. Updated CLKSWING_CFG type to Binary in Table 2-2. Updated CLKSWING_CFG setting in Table 2-3. Updated Channel PLL and Quad PLL. Updated Figure 2-15. Updated GTX/GTH Transceiver TX Reset in Response to Completion of Configuration. Updated Figure 2-20. Updated GTX/GTH Transceiver RX Reset in Response to Completion of Configuration. In Table 2-33, updated RXPD[1:0] clock domain to Async. Added Digital Monitor. Chapter 3: Updated Figure 3-1. Added BUFH to Figure 3-4 and Figure 3-5, and notes for each figure. Updated Running Disparity. Updated TX Gearbox. Updated Figure 3-14. In Table 3-17, updated description of TXPHDLYPD and added TXSYNCMODE, TXSYNCALLIN, TXSYNCIN, TXSYNCOUT, and TXSYNCDONE. Added TXSYNC_MULTI_LANE, TXSYNC_SKIP_DA, and TXSYNC_OVRD to Table 3-18. Updated notes after Figure 3-23. Added TX Phase Interpolator PPM Controller. Updated Figure 3-28, including notes 2 and 6. In Table 3-28, updated descriptions of TXDIFFCTRL[3:0], TXELECIDLE, TXINHIBIT, TXPOSTCURSOR[4:0], TXPRECURSOR[4:0], and GTXTXP/GTXTXN. In Table 3-32, updated RXPD[1:0] clock domain to Async.

Date	Version	Revision
04/04/12	1.4 *(Cont'd)*	Chapter 4: Updated Table 4-2. Updated GTX and GTH Use Modes—RX Termination and RX Equalizer (DFE and LPM). In Table 4-11 and Table 4-15, replaced RXDFEOSHOLD, RXDFEOSOVRDEN, and RX_DFE_OS_CFG with RXOSHOLD, RXOSOVRDEN, and RX_OS_CFG, respectively. In Table 4-11, merged RX_DFE_LPM and HOLD_DURING_EIDLE into RX_DFE_LPM_HOLD_DURING_EIDLE, and removed IAS_CFG. Added GTX Use Modes, including Figure 4-19, Table 4-12, and Table 4-13. Updated Figure 4-22, including note 2. In Table 4-25, added RXLPMEN. In Table 4-26, removed A_RXLPMEN, updated ES_EYE_SCAN_EN, and added PMA_RSV2[5], USE_PCS_CLK_PHASE_SEL, and ES_CLK_PHASE_SEL. Updated description of RX_PRBS_ERR_CNT in Table 4-30. Updated Alignment Status Signals and Manual Alignment. Updated description of RXBYTEISALIGNED in Table 4-31. Added COMMA_ALIGN_LATENCY to Table 4-32. Added sentence about use of RXSLIDE_MODE to RX Buffer Bypass. In Table 4-35, updated description of RXPHDLYPD, and added RXSYNCMODE, RXSYNCALLIN, RXSYNCIN, RXSYNCOUT, and RXSYNCDONE. In Table 4-36, added RXSYNC_MULTI_LANE, RXSYNC_SKIP_DA, and RXSYNC_OVRD. Updated Using RX Buffer Bypass in Single-Lane Auto Mode (GTX Transceiver Only) heading. Updated notes after Figure 4-44. Updated description of FTS_LANE_DESKEW_CFG in Table 4-47. Updated Functional Description, page 275. In Table 4-48, updated descriptions of RXDATAVALID, RXGEARBOXSLIP, RXHEADER, RXHEADERVALID, and RXSTARTOFSEQ, and added RXSLIDE. Updated description of GEARBOX_MODE in Table 4-49. Added normal mode to RX Gearbox Operating Modes. Updated Figure 4-64 and RX Gearbox Block (GTH Transceiver). Chapter 5: Updated Table 5-1, Table 5-4, and Table 5-5. Added LVDS, including Figure 5-8. In Figure 5-9, updated value of capacitors from 0.01 µF to 0.1 µF. Chapter 6: Updated Functional Description, including Table 6-1. Updated description of RXELECIDLE in Table 6-2. Updated [TX/RX]RATE[2:0] setting in Table 6-4. Updated [TX/RX]OUT_DIV setting in Table 6-6. Updated QPLL_FBDIV setting in Table 6-7. PCLK frequency in Parallel Clock (PCLK). Updated Rate Change to Enter or Exit Gen3 Speed. Updated PCI Express Clock Correction. In Table 6-12, updated value of [TX/RX]_CLK25_DIV from 2 to 7. Appendix A: Added sentence about leaded package options to first paragraph. Added GTH Transceiver Package Placement Diagrams. Appendix B: Added Table B-3. Appendix D: Updated Table D-1 and Table D-2. Added Table D-3 and Table D-4.
05/07/12	1.5	Chapter 5: Updated recommendations for MGTVCCAUX_G[N] in Table 5-5. Appendix A: Corrected labels in Figure A-50, Figure A-61, Figure A-63, Figure A-91, and Figure A-92.
09/11/12	1.6	Chapter 2: Updated Figure 2-1. Updated CLKSWING_CFG in Table 2-2. Added GTREFCLKMONITOR to Table 2-4. Added OUTREFCLK_SEL_INV to Table 2-5. Added REFCLKOUTMONITOR to Table 2-6. Added Single External Reference Clocks Use Model and Multiple External Reference Clocks Use Model. Updated Channel PLL and Quad PLL. Added note to Table 2-10. Added QPLL Settings for Common Protocols. Updated Table 2-12. Removed QPLL_CFG from Attribute column for Factor N in Table 2-13. Added Table 2-27 and descriptions of reset situations. Updated description of RXRESETDONE in Table 2-28. Replaced RXCDRRESET_TIME with RXCDRPHRESET_TIME in Table 2-29. Added Table 2-32 and descriptions of reset situations. Updated description of loopback modes in Loopback. Updated description of DMONITOR_CFG[23:0] and added RX_DEBUG_CFG, PCS_RSVD_ATTR[6], and CFOK_CFG[41] to Table 2-41. Updated Verilog code in Capturing the Digital Monitor Output. In Interpreting the Digital Monitor Output, added bullet for RXDFELF[3:0] and updated RXDFEAGC[3:0] for GTX and GTH transceivers.

Date	Version	Revision
09/11/12	1.6 *(Cont'd)*	Chapter 3: Updated Table 3-2. Replaced GTX with GTH in Figure 3-14. Updated Figure 3-15. Updated description of TXPHDLYRESET in Table 3-17. Updated description of TXSYNC_MULTILANE in and title of Table 3-18. Added TX Buffer Bypass Use Modes. Updated Using TX Buffer Bypass in Single-Lane Auto Mode (GTX Transceiver Only) heading. Updated title and notes for Figure 3-19. Added Using TX Buffer Bypass in Single-Lane Manual Mode (GTH Transceiver Only). Updated Using TX Buffer Bypass in Multi-Lane Manual Mode (GTX and GTH Transceivers). Added Using TX Buffer Bypass in Multi-Lane Auto Mode (GTH Transceiver Only). Updated Figure 3-28 and added note 5 after figure. In Table 3-27, replaced TX_EN_RATE_RESET_BUF with TXBUF_RESET_ON_RATE_CHANGE. In Table 3-29, updated description of TXPI_SYNFREQ_PPM[2:0]. In Table 3-30, updated PIPE version to 3.0, added TXDIFFPD and TXPISOPD, updated TXQPISENN and TXQPISENP, and removed preliminary from descriptions. Updated Table 3-31. Added bits to SATA_BURST_SEQ_LEN in Table 3-35. Chapter 4: Updated Figure 4-2. Updated descriptions of RXQPISENN and RXQPISENP in Table 4-1. In Table 4-2, updated description of TERM_RCAL_CFG. Removed DisplayPort from Table 4-4. In Table 4-5, added LPM mode, removed DisplayPort, and updated attribute settings. In Table 4-7, changed clock domain for RXELECIDLE from RXUSRCLK2 to Async. Updated Table 4-8. Added GTX/GTH Use Mode. Added table note 1 to Table 4-9. Updated Figure 4-18. In Table 4-10, updated RXLPMLFKLOVRDEN and added RXDFECM1EN, RXDFEXYDHOLD, RXDFEXYDOVRDEN, RXDFEXYDEN, RXMONITORSEL, and RXMONITOROUT. In Table 4-11, added RX_DFE_XYD_CFG and default values of other attributes. Removed paragraphs about CDR lock after Figure 4-21. Updated descriptions of RXRATE[2:0] and RXCDRLOCK in Table 4-15. Updated descriptions of RXCDR_HOLD_DURING_EIDLE, RXCDR_FR_RESET_ON_EIDLE, and RXCDR_PH_RESET_ON_EIDLE in Table 4-16. Updated Figure 4-22, and notes 3 and 6 after figure. In Table 4-24, replaced RX_EN_RATE_RESET_BUF with RXBUF_RESET_ON_RATE_CHANGE. In Table 4-25, updated description of RXRATE and added EYESCANMODE. In Table 4-26, added ES_PMA_CFG and updated descriptions of ES_SDATA_MASK, ES_QUALIFIER, and ES_QUAL_MASK. Added DRP Address Hex (GTH Transceiver) column to Table 4-27. Added SETERRSTATUS to Table 4-33. Added UCODEER_CLR to Table 4-34. Updated description of RXPHDLYRESET and RXPHOVRDEN in Table 4-35. Updated title of Table 4-36. Added RX Buffer Bypass Use Modes. Updated Using RX Buffer Bypass in Single-Lane Auto Mode (GTX Transceiver Only). Added Using RX Buffer Bypass in Single-Lane Auto Mode (GTH Transceiver Only). Updated Using RX Buffer Bypass in Multi-Lane Manual Mode (GTX and GTH Transceivers). Added Using RX Buffer Bypass in Multi-Lane Auto Mode (GTH Transceiver Only). Updated Figure 4-49. In Table 4-43, corrected CLK_COR_PRECEDENT with CLK_COR_PRECEDENCE and updated upper range of valid values for CLK_COR_MAX_LAT and CLK_COR_MIN_LAT from 48 to 60. In Setting RX Elastic Buffer Limits, changed upper value of CLK_COR_MAX_LAT and CLK_COR_MIN_LAT from 48 to 60. Chapter 5: Updated second paragraph of Termination Resistor Calibration Circuit. Updated first and third paragraphs of Analog Power Supply Pins. In Table 5-2, removed "Preliminary" after device numbers in first column, added "(RCAL)" to entries in MGT 115 column, and removed XC7K410T-FBG900, XC7K410T-FFG900, and XC7K420T-FFG901 rows. In Table 5-3, removed "Preliminary" after device numbers in first column and added "(RCAL)" to entries in MGT columns. Updated paragraph before Figure 5-4. In Figure 5-4, replaced V_{ISE} with "Single-Ended Voltage." Chapter 6: Added sentence about P2 state to PIPE Control Signal. Updated second row of Table 6-7. Updated first paragraph of PCI Express Channel Bonding.

Date	Version	Revision
09/11/12	1.6 *(Cont'd)*	Appendix A: Added FLG1761 package to GTX Transceiver Package Placement Diagrams. Added FFG1928, FLG1926, FLG1928, and FLG1930 package diagrams to GTH Transceiver Package Placement Diagrams. Appendix B: Updated Table B-2 and Table B-3. Appendix D: Updated Table D-2 and Table D-4.
10/18/12	1.7	Chapter 5: Removed XC7V1500T FLG1761 row from Table 5-3. Appendix A: Removed FLG1761 package placement diagrams. Appendix B: Removed XC7V1500T from Table B-2.
01/02/13	1.8	Chapter 2: Added TXPMARESETDONE to Table 2-24. Added RXOSCALRESET, RSOSINTDONE, and RXPMARESETDONE to Table 2-28. Added RXOSCALRESET_TIME and RXOSCALRESET_TIMEOUT to Table 2-29. Updated description of DMONITOR_CFG[23:0] in Table 2-41. Updated Interpreting the Digital Monitor Output. Chapter 3: Updated TXDATA signal in Figure 3-12 and Figure 3-13. Added note about external sequence counter to Table 3-12 and Table 3-13. Updated descriptions of ports for TXSYNCMODE, TXSYNCALLIN, TXSYNCIN, TXSYNCOUT, and TXSYNCDONE in Table 3-17. Updated description of TXSYNC_MULTILANE and TXSYNC_SKIP_DA, and added LOOPBACK_CFG to Table 3-18. Added note about multi-lane TX buffer bypass support to Table 3-19, Using TX Buffer Bypass in Multi-Lane Manual Mode (GTX and GTH Transceivers), and Using TX Buffer Bypass in Multi-Lane Auto Mode (GTH Transceiver Only). Chapter 4: Updated Suggested Protocols and Usage Notes in Table 4-3 to Table 4-6. Updated Figure 4-3 to Figure 4-6. Updated description of RXELECIDLE port in Table 4-7. Updated description of PCS_RSVD_ATTR[8] port in Table 4-8. Updated Using DFE Mode. Added "GTX" to titles of Table 4-17 through Table 4-19. Updated descriptions of RXSYNCMODE, RXSYNCALLIN, RXSYNCIN, RXSYNCOUT, and RXSYNCDONE in Table 4-35. Added note about multi-lane RX buffer bypass support to Table 4-37, Using RX Buffer Bypass in Multi-Lane Manual Mode (GTX and GTH Transceivers), and Using RX Buffer Bypass in Multi-Lane Auto Mode (GTH Transceiver Only). Updated description of RXBUF_ADDR_MODE in Table 4-40. Added paragraph about CLK_COR_MIN_LAT setting after Table 4-41. Chapter 5: Updated pin names in Table 5-1. Updated XC7K420T-FFG1156 for MGT 118 in Table 5-2. Updated Table 5-3. Updated pin names in Table 5-5.
04/04/13	1.9	Chapter 1: Updated text after Figure 1-2. Removed description of CPLL after Figure 1-5. Chapter 2: Removed GTX from CPLL Settings for Common Protocols and QPLL Settings for Common Protocols. In Table 2-15, changed type for QPLL_CLKOUT_CFG to 4-bit Binary. Added note 1 to Table 2-17. Added GTH to title of Figure 2-12. In Table 2-24, updated clock domain of TXRESETDONE to TXUSRCLK2. Added GTH transceiver to note 1 in Table 2-27. In Figure 2-21 and Figure 2-22, replaced RXISCANRESET with EYESCANRESET. Updated GTX/GTH Transceiver RX Reset in Response to Completion of Configuration and GTX/GTH Transceiver RX Reset in Response to GTRXRESET Pulse. Added GTH Transceiver RX PMA Reset. In Table 2-30 and Table 2-31, replaced RX ISCAN with RX EYESCAN. Updated title of After Changing Channel Bonding Mode During Run Time. Updated description of near-end PCS loopback after Figure 2-26. In Table 2-38 and Table 2-39, updated descriptions of DRPEN and DRPWE. Chapter 3: Updated width of TXDATA[39:32] in Figure 3-16. Updated Functional Description, page 127. Updated descriptions of TXSYNCMODE, TXSYNCALLIN, TXSYNCIN, TXSYNCOUT, and TXSYNCDONE in Table 3-17. Updated descriptions of TXSYNC_MULTILANE, TXSYNC_SKIP_DA, and TXSYNC_OVRD in Table 3-18.

Date	Version	Revision
04/04/13	1.9 *(Cont'd)*	Updated TX Buffer Bypass Use Modes, including Table 3-19. Added note 5 after Figure 3-19. Updated Using TX Buffer Bypass in Single-Lane Manual Mode (GTH Transceiver Only), including Figure 3-20. Updated second paragraph of Using TX Buffer Bypass in Multi-Lane Manual Mode (GTX and GTH Transceivers). Removed "Using TX Buffer Bypass in Multi-Lane Auto Mode (GTH Transceiver Only)" section. Updated Figure 3-22. Updated third bullet after Figure 3-24. In Table 3-26, updated descriptions of TXOUTCLKFABRIC and TXOUTCLKPCS. In Table 3-29, replaced GT Wizard with 7 Series FPGAs Transceivers Wizard. In Table 3-30, updated descriptions of TXDIFFCTRL[3:0], TXELECIDLE, TXINHIBIT, TXQPISENN, and TXQPISENP. Chapter 4: In Table 4-2, updated descriptions of TERM_RCAL_CFG and TERM_RCAL_OVRD. Corrected spelling of PMA_RSV2 in Table 4-3, Table 4-4, Table 4-5, and Figure 4-5. In Table 4-8, added PCS_RSVD_ATTR[3] and RXOOB_CLK_CFG. Updated Figure 4-7 and accompanying text. Removed CDR block from Figure 4-16, Figure 4-17, and Figure 4-18. Removed GTH transceivers from Choosing Between LPM and DFE Modes and Using LPM Mode. Updated Using DFE Mode. Added GTH Use Modes. Updated GTX/GTH Use Modes. Added Table 4-20. In Table 4-23, updated descriptions of RXOUTCLKFABRIC, RXOUTCLKPCS, and RXDLYBYPASS. Added Using RXRATE (GTH Transceiver Only). In Table 4-33, updated description of RXDISPERR[7:0]. In Table 4-35, updated description of RXPHALIGNDONE. In Table 4-36, updated type for RXPH_CFG. Added RXDDIEN to bulleted list after Figure 4-43. Updated width of RXDATA[39:32] in Figure 4-67. Appendix A: Removed discussion about leaded package options from second introductory paragraph.
04/22/13	1.9.1	Chapter 3: Removed repetitive rows in Table 3-30.

Table of Contents

Revision History 3

Preface: About This Guide

Guide Contents 19
Additional Resources 20
Additional References 20

Chapter 1: Transceiver and Tool Overview

Overview and 7 Series Features 21
7 Series FPGAs Transceivers Wizard 27
Simulation 27
Functional Description 27
Ports and Attributes 28
GTXE2_COMMON Attributes 28
GTXE2_CHANNEL/GTHE2_CHANNEL Attributes 29
Implementation 30
Functional Description 30
Serial Transceiver Channels by Device/Package 31

Chapter 2: Shared Features

Reference Clock Input Structure 33
Functional Description 33
Ports and Attributes 34
Use Modes: Reference Clock Termination 35
Reference Clock Selection and Distribution 35
Functional Description 35
Ports and Attributes 37
External Reference Clock Use Model 40
Single External Reference Clocks Use Model 40
Multiple External Reference Clocks Use Model 43
Channel PLL 46
Functional Description 46
Ports and Attributes 48
CPLL Settings for Common Protocols 50
Use Modes 52
Dynamically Changing CPLL settings 52
Dynamically switching from CPLL to QPLL 53
Quad PLL 53
Functional Description 53
Ports and Attributes 55
QPLL Settings for Common Protocols 59
Use Modes 59
Dynamically Changing QPLL settings 59
Dynamically Switching from QPLL to CPLL 60

Reset and Initialization 60
Reset Modes 62
CPLL Reset 63
QPLL Reset 63
TX Initialization and Reset 64
Ports and Attributes 65
GTX/GTH Transceiver TX Reset in Response to Completion of Configuration 66
GTX/GTH Transceiver TX Reset in Response to GTTXRESET Pulse 67
GTX/GTH Transceiver TX Component Reset 68
After Power-up and Configuration. 70
After Turning on a Reference Clock to the CPLL/QPLL Being Used 70
After Changing the Reference Clock to the CPLL/QPLL being used 70
After Assertion/Deassertion of C/QPLLPD, for the PLL being used. 70
After Assertion/Deassertion of TXPD[1:0] 70
TX Rate Change 70
TX Parallel Clock Source Reset 70
RX Initialization and Reset 71
Ports and Attributes 73
GTX/GTH Transceiver RX Reset in Response to Completion of Configuration 77
GTX/GTH Transceiver RX Reset in Response to GTRXRESET Pulse 79
GTH Transceiver RX PMA Reset 81
GTX/GTH Transceiver RX Component Resets 82
After Power-up and Configuration. 84
After Turning on a Reference Clock to the CPLL/QPLL Being Used 85
After Changing the Reference Clock to the CPLL/QPLL Being Used 85
After Assertion/Deassertion of CPLLD or QPLLPD for the PLL Being Used 85
After Assertion/Deassertion of RXPD[1:0] 85
RX Rate Change 85
RX Parallel Clock Source Reset 85
After Remote Power-Up 85
Electrical Idle Reset 85
After Connecting RXN/RXP. 86
After Recovered Clock Becomes Stable. 86
After an RX Elastic Buffer Error 86
After Changing Channel Bonding Mode During Run Time 86
After a PRBS Error 86
After Comma Realignment 86
Power Down 86
Functional Description 86
Ports and Attributes 87
Generic Power-Down Capabilities 89
PLL Power Down 89
TX and RX Power Down 89
Power-Down Features for PCI Express Operation 89
Loopback 90
Functional Description 90
Ports and Attributes 91
Dynamic Reconfiguration Port 91
Functional Description 91
Ports and Attributes 91
Usage Model 93
Write Operation 93
Read Operation 93

Digital Monitor 94
Functional Description 94
Ports and Attributes 94
Use Mode 1 96

Chapter 3: Transmitter

TX Overview 99
Functional Description 99
FPGA TX Interface 100
Functional Description 100
Interface Width Configuration 100
TXUSRCLK and TXUSRCLK2 Generation 101
Ports and Attributes 102
Using TXOUTCLK to Drive the TX Interface 103
TXOUTCLK Driving GTX/GTH Transceiver TX in 2-Byte or 4-Byte Mode 104
TXOUTCLK Driving GTX/GTH Transceiver TX in 4-Byte or 8-Byte Mode 106
TX 8B/10B Encoder 108
Functional Description 108
8B/10B Bit and Byte Ordering 108
K Characters 108
Running Disparity 109
Ports and Attributes 110
Enabling and Disabling 8B/10B Encoding 111
TX Gearbox 112
Functional Description 112
Ports and Attributes 112
Enabling the TX Gearbox 113
TX Gearbox Bit and Byte Ordering 114
TX Gearbox Operating Modes 115
External Sequence Counter Operating Mode 115
Internal Sequence Counter Operating Mode (GTX Transceiver Only) 119
CAUI Interface (GTH Transceiver) 121
Use Case 122
TX Gearbox Block (GTH Transceiver) 122
TX Buffer 125
Functional Description 125
Ports and Attributes 126
Using the TX Buffer 127
TX Buffer Bypass 127
Functional Description 127
Ports and Attributes 127
TX Buffer Bypass Use Modes 131
Using TX Buffer Bypass in Single-Lane Auto Mode (GTX Transceiver Only) 131
Using TX Buffer Bypass in Single-Lane Manual Mode (GTH Transceiver Only) . . 132
Using the TX Phase Alignment to Minimize the TX Lane-to-Lane Skew 133
Using TX Buffer Bypass in Multi-Lane Manual Mode (GTX and GTH Transceivers) 134
TX Pattern Generator 137
Functional Description 137
Ports and Attributes 139
Use Models 140

TX Polarity Control 141
Functional Description 141
Ports and Attributes 141
Using TX Polarity Control 141
TX Fabric Clock Output Control 141
Functional Description 141
Serial Clock Divider 143
Parallel Clock Divider and Selector 143
Ports and Attributes 144
TX Phase Interpolator PPM Controller 146
Functional Description 146
Ports and Attributes 146
TX Phase Interpolator PPM Controller Use Mode 147
TX Configurable Driver 148
Functional Description 148
Ports and Attributes 148
TX Receiver Detect Support for PCI Express Designs 155
Functional Description 155
Ports and Attributes 155
Using the TX Receiver Detection for PCI Express 156
TX Out-of-Band Signaling 157
Functional Description 157
Ports and Attributes 157

Chapter 4: Receiver

RX Overview 159
Functional Description 159
RX Analog Front End 160
Functional Description 160
Ports and Attributes 161
GTX and GTH Use Modes—RX Termination 163
RX Out-of-Band Signaling 168
Functional Description 168
Ports and Attributes 168
GTX/GTH Use Mode 169
Use Modes 171
RX Equalizer (DFE and LPM) 176
Functional Description 176
Ports and Attributes 179
GTX Use Modes 187
Choosing Between LPM and DFE Modes 187
Using LPM Mode 187
Using DFE Mode 187
GTH Use Modes 190
Choosing Between LPM and DFE Modes 190
Using LPM Mode 190
Using DFE Mode 191
GTX and GTH Transceivers: Switching Between LPM and DFE Modes at Run Time 191
RX CDR 191
Functional Description 191
Ports and Attributes 192

GTX/GTH Use Modes . . . 195
RX CDR Lock to Reference . . . 195
Dynamically Changing RX CDR Settings for Line Rate and Selected Protocol Changes 195
Dynamically Changing RX CDR Settings to Tune CDR Loop Filter Settings Only. . . . 195
RX Fabric Clock Output Control . . . 200
Functional Description . . . 200
Serial Clock Divider . . . 202
Parallel Clock Divider and Selector . . . 202
Ports and Attributes . . . 203
Using RXRATE (GTH Transceiver Only) . . . 204
RX Margin Analysis . . . 205
Functional Description . . . 205
Eye Scan Theory . . . 206
Eye Scan Architecture . . . 206
Ports and Attributes . . . 210
RX Polarity Control . . . 214
Functional Description . . . 214
Ports and Attributes . . . 214
Using RX Polarity Control . . . 214
RX Pattern Checker . . . 214
Functional Description . . . 214
Ports and Attributes . . . 215
Use Models . . . 216
RX Byte and Word Alignment . . . 216
Functional Description . . . 216
Enabling Comma Alignment . . . 217
Configuring Comma Patterns . . . 217
Activating Comma Alignment . . . 218
Alignment Status Signals . . . 219
Manual Alignment . . . 221
Ports and Attributes . . . 223
RX 8B/10B Decoder . . . 228
Functional Description . . . 228
8B/10B Bit and Byte Ordering . . . 228
RX Running Disparity . . . 230
Special Characters . . . 230
Ports and Attributes . . . 231
Enabling and Disabling 8B/10B Decoding . . . 233
RX Buffer Bypass . . . 233
Functional Description . . . 233
Ports and Attributes . . . 234
RX Buffer Bypass Use Modes . . . 238
Using RX Buffer Bypass in Single-Lane Auto Mode (GTX Transceiver Only) . . . 239
Using RX Buffer Bypass in Single-Lane Auto Mode (GTH Transceiver Only) . . . 240
Using RX Buffer Bypass in Multi-Lane Manual Mode (GTX and GTH Transceivers) 242
Using RX Buffer Bypass in Multi-Lane Auto Mode (GTH Transceiver Only) . . . 244
RX Elastic Buffer . . . 248
Functional Description . . . 248
Ports and Attributes . . . 249
Using the RX Elastic Buffer . . . 253
RX Clock Correction . . . 253

Functional Description . . . 253
Ports and Attributes . . . 255
Using RX Clock Correction . . . 260
Enabling Clock Correction . . . 260
Setting RX Elastic Buffer Limits . . . 261
Setting Clock Correction Sequences . . . 261
Clock Correction Options . . . 263
Monitoring Clock Correction . . . 263
RX Channel Bonding . . . 263
Functional Description . . . 263
Ports and Attributes . . . 265
Using RX Channel Bonding . . . 271
Enabling Channel Bonding . . . 271
Channel Bonding Mode . . . 271
Connecting Channel Bonding Ports . . . 271
Setting Channel Bonding Sequences . . . 273
Setting the Maximum Skew . . . 274
Precedence between Channel Bonding and Clock Correction . . . 275
RX Gearbox . . . 275
Functional Description . . . 275
Ports and Attributes . . . 276
Enabling the RX Gearbox . . . 278
RX Gearbox Operating Modes . . . 279
RX Gearbox Block Synchronization . . . 281
CAUI Interface (GTH Transceiver) . . . 285
Use Case . . . 285
RX Gearbox Block (GTH Transceiver) . . . 285
FPGA RX Interface . . . 287
Functional Description . . . 287
Interface Width Configuration . . . 287
RXUSRCLK and RXUSRCLK2 Generation . . . 288
Ports and Attributes . . . 289

Chapter 5: Board Design Guidelines

Overview . . . 291
Pin Description and Design Guidelines . . . 291
GTX/GTH Transceiver Pin Descriptions . . . 291
Termination Resistor Calibration Circuit . . . 294
Analog Power Supply Pins . . . 295
Reference Clock . . . 300
Overview . . . 300
GTX/GTH Transceiver Reference Clock Checklist . . . 302
Reference Clock Interface . . . 303
LVDS . . . 303
LVPECL . . . 303
AC Coupled Reference Clock . . . 304
Unused Reference Clocks . . . 304
Reference Clock Power . . . 304
Power Supply and Filtering . . . 304
Overview . . . 304
Power Supply Regulators . . . 305

Linear versus Switching Regulators . . . 305
Linear Regulator . . . 305
Switching Regulator . . . 306
Power Supply Distribution Network . . . 306
Staged Decoupling . . . 306
Die . . . 306
Package . . . 306
Printed Circuit Board . . . 306
PCB Design Checklist . . . 307

Chapter 6: Use Model

PCI Express . . . 311
Functional Description . . . 311
Ports and Attributes . . . 312
PCI Express Use Mode . . . 315
PIPE Control Signal . . . 316
PCI Express Clocking . . . 316
Reference Clock . . . 316
Parallel Clock (PCLK) . . . 317
PCI Express Reset . . . 319
PCI Express Power Management . . . 321
PCI Express Rate Change . . . 321
Rate Change between Gen1 and Gen2 Speeds . . . 321
Rate Change to Enter or Exit Gen3 Speed . . . 322
Using DRP During Rate Change to Enter or Exit Gen3 Speed . . . 324
PCI Express Channel Bonding . . . 325
One-Hop Example . . . 325
Daisy-Chain Example . . . 326
Binary-Tree Example . . . 326
Channel Bonding Attribute Settings . . . 328
PCI Express Clock Correction . . . 329
XAUI Use Model . . . 331
Functional Description . . . 331
XAUI Use Mode . . . 331
XAUI Clocking . . . 332
Reference Clock . . . 332
Parallel Clock . . . 332
XAUI Channel Bonding . . . 334
XAUI Clock Correction . . . 335

Appendix A: Placement Information by Package

GTX Transceiver Package Placement Diagrams . . . 337
FBG484 Package Placement Diagram . . . 338
FBG676 Package Placement Diagram . . . 339
FBG900 Package Placement Diagram . . . 340
FFG676 Package Placement Diagram . . . 342
FFG900 Package Placement Diagram . . . 343
FFG901 Package Placement Diagram . . . 345
FFG1156 Package Placement Diagram . . . 349
FFG1157 Package Placement Diagram . . . 353
FFG1158 Package Placement Diagram . . . 356

FFG1761 Package Placement Diagram 362
FFG1927 Package Placement Diagram 367
FFG1930 Package Placement Diagram 375
FLG1925 Package Placement Diagram 378
FHG1761 Package Placement Diagram 380
GTH Transceiver Package Placement Diagrams 385
FFG1157 Package Placement Diagram 386
FFG1158 Package Placement Diagram 389
FFG1761 Package Placement Diagram 395
FFG1926 Package Placement Diagram 400
FFG1927 Package Placement Diagram 408
FFG1928 Package Placement Diagram 418
FFG1930 Package Placement Diagram 428
FLG1926 Package Placement Diagram 431
FLG1928 Package Placement Diagram 439
FLG1930 Package Placement Diagram 451

Appendix B: Placement Information by Device

Appendix C: 8B/10B Valid Characters

Appendix D: DRP Address Map of the GTX/GTH Transceiver

About This Guide

Xilinx® 7 series FPGAs include three unified FPGA families that are all designed for lowest power to enable a common design to scale across families for optimal power, performance, and cost. The Artix™-7 family is optimized for lowest cost and absolute power for the highest volume applications. The Virtex®-7 family is optimized for highest system performance and capacity. The Kintex™-7 family is an innovative class of FPGAs optimized for the best price-performance. This guide serves as a technical reference describing the 7 series FPGAs GTX/GTH transceivers.

The 7 series FPGAs GTX/GTH transceivers user guide, part of an overall set of documentation on the 7 series FPGAs, is available on the Xilinx website at www.xilinx.com/7.

In this document:

- 7 series FPGAs GTX/GTH transceiver channel is abbreviated as GTX/GTH transceiver.
- GTXE2_CHANNEL/GTHE2_CHANNEL is the name of the instantiation primitive that instantiates one GTX/GTH transceiver channel.
- GTXE2_COMMON/GTHE2_COMMON is the name of the primitive that instantiates one Quad PLL (QPLL).
- A Quad or Q is a cluster or set of four GTX/GTH transceiver channels, one GTXE2_COMMON/GTHE2_COMMON primitive, two differential reference clock pin pairs, and analog supply pins.

Guide Contents

This manual contains:

- Chapter 1, Transceiver and Tool Overview
- Chapter 2, Shared Features
- Chapter 3, Transmitter
- Chapter 4, Receiver
- Chapter 5, Board Design Guidelines
- Chapter 6, Use Model
- Appendix A, Placement Information by Package
- Appendix B, Placement Information by Device
- Appendix C, 8B/10B Valid Characters
- Appendix D, DRP Address Map of the GTX/GTH Transceiver

Additional Resources

To find additional documentation, see the Xilinx website at:

http://www.xilinx.com/support/documentation/index.htm.

To search the Answer Database of silicon, software, and IP questions and answers, or to create a technical support WebCase, see the Xilinx website at:

http://www.xilinx.com/support.

Additional References

These documents provide additional information useful to this document:

1. *High-Speed Serial I/O Made Simple*
   http://www.xilinx.com/publications/archives/books/serialio.pdf

Chapter 1

Transceiver and Tool Overview

Overview and 7 Series Features

The 7 series FPGAs GTX and GTH transceivers are power-efficient transceivers, supporting line rates from 500 Mb/s to 12.5 Gb/s for GTX transceivers and 13.1 Gb/s for GTH transceivers. The GTX/GTH transceiver is highly configurable and tightly integrated with the programmable logic resources of the FPGA. Table 1-1 summarizes the features by functional group that support a wide variety of applications.

Table 1-1: **7 Series FPGAs GTX and GTH Transceiver Features**

Group	Feature	GTX	GTH
PCS	2-byte and 4-byte internal datapath to support different line rate requirements	X	X
	8B/10B encoding and decoding	X	X
	64B/66B and 64B/67B support	X	X
	Comma detection and byte and word alignment	X	X
	PRBS generator and checker	X	X
	FIFO for clock correction and channel bonding	X	X
	Programmable FPGA logic interface	X	X
	100 Gb Attachment Unit Interface (CAUI) support		X
	Native multi-lane support for buffer bypass		X
	TX Phase Interpolator PPM Controller for external voltage-controlled crystal oscillator (VCXO) replacement		X

Table 1-1: **7 Series FPGAs GTX and GTH Transceiver Features *(Cont'd)***

Group	Feature	GTX	GTH
PMA	Shared LC tank phase-locked loop (PLL) per Quad for best jitter performance	X	X
	One ring PLL per channel for best clocking flexibility	X	X
	Power-efficient adaptive linear equalizer mode called the low-power mode (LPM)	X	X
	5-tap decision feedback equalization (DFE)	X	
	7-tap DFE		X
	Reflection cancellation for enhanced backplane support		X
	TX Pre-emphasis	X	X
	Programmable TX output	X	X
	Beacon signaling for PCI Express® designs	X	X
	Out-of-band (OOB) signaling including COM signal support for Serial ATA (SATA) designs	X	X
	Line rate support up to 12.5 Gb/s	X	X
	Line rate support up to 13.1 Gb/s		X

The GTX/GTH transceiver supports these use modes:

- PCI Express, Revision 1.1/2.0/3.0
- 10GBASE-R
- Interlaken
- 10 Gb Attachment Unit Interface (XAUI), Reduced Pin eXtended Attachment Unit Interface (RXAUI), 100 Gb Attachment Unit Interface (CAUI), 40 Gb Attachment Unit Interface (XLAUI)
- Common Packet Radio Interface (CPRI™)/Open Base Station Architecture Initiative (OBSAI)
- OC-48/192
- OTU-1, OTU-2, OTU-3, OTU-4
- Serial RapidIO (SRIO)
- Serial Advanced Technology Attachment (SATA)/Serial Attached SCSI (SAS)
- Serial Digital Interface (SDI)

In comparison to prior generation transceivers in Virtex-6 FPGAs, the GTX/GTH transceiver in the 7 series FPGAs has the following new or enhanced features:

- 2-byte and 4-byte internal datapath to support different line rate requirements.
- Quad-based LC tank PLL (QPLL) for best jitter performance and channel-based ring oscillator PLL.
- Power-efficient, adaptive linear equalizer mode called the low-power mode (LPM) and a high-performance, adaptive decision feedback equalization (DFE) mode to compensate for high frequency losses in the channel while providing maximum flexibility.

- RX margin analysis feature to provide non-destructive, 2-D post-equalization eye scan.

The first-time user is recommended to read *High-Speed Serial I/O Made Simple* [Ref 1], which discusses high-speed serial transceiver technology and its applications. The CORE Generator™ tool includes a wizard to automatically configure GTX/GTH transceivers to support configurations for different protocols or perform custom configuration. The GTX/GTH transceiver offers a data rate range and features that allow physical layer support for various protocols.

Figure 1-1, page 24 shows the GTX transceiver placement in an example Kintex™-7 device (XC7K325T). This device has 16 GTX transceivers.

Figure 1-1: **GTX Transceiver Inside Kintex-7 XC7K325T FPGA**

Additional information on the functional blocks of 7 series FPGAs is available at:

UG470, *7 Series FPGAs Configuration User Guide* provides more information on the configuration.

UG471, *7 Series FPGAs SelectIO Resources User Guide* provides more information on the I/O blocks.

UG472, *7 Series FPGAs Clocking Resources User Guide* provides more information on the mixed mode clock manager (MMCM) and clocking.

Figure 1-2 illustrates the clustering of four GTXE2_CHANNEL primitives and one GTXE2_COMMON primitive to form a Quad.

Figure 1-2: **GTX Transceiver Quad Configuration**

Four GTXE2 channels clustered together with one GTXE2_COMMON primitive are called a *Quad* or *Q*. The GTXE2_COMMON primitive contains an LC-tank PLL (QPLL).

Each GTXE2_CHANNEL primitive consists of a channel PLL, a transmitter, and a receiver.

Figure 1-3 illustrates the topology of a GTXE2_CHANNEL primitive.

Figure 1-3: **GTXE2_CHANNEL Primitive Topology**

Refer to Figure 2-9, page 46 for the description of the channel clocking architecture, which provides clocks to the RX and TX clock dividers.

7 Series FPGAs Transceivers Wizard

The 7 Series FPGAs Transceivers Wizard (hereinafter called the Wizard) is the preferred tool to generate a wrapper to instantiate a GTX/GTH transceiver primitive called GTXE2 or GTHE2. The Wizard is located in the CORE Generator tool. The user is recommended to download the most up-to-date IP update before using the Wizard. Details on how to use this Wizard can be found in UG769, *LogiCORE IP 7 Series FPGAs Transceivers Wizard User Guide*.

Follow these steps to launch the Wizard:

1. Start the CORE Generator tool.
2. Locate the 7 Series FPGAs Transceivers Wizard in the taxonomy tree under:

   `/FPGA Features & Design/IO Interfaces`

   See Figure 1-4.

UG476_c1_04_011011

Figure 1-4: **7 Series FPGAs Transceivers Wizard**

3. Double-click 7 Series FPGAs Transceivers Wizard to launch the Wizard.

Simulation

Functional Description

Simulations using the GTXE2/GTHE2 channel and common primitives have specific prerequisites that the simulation environment and the test bench must fulfill. For instructions on how to set up the simulation environment for supported simulators depending on the used hardware description language (HDL), see the latest version of UG626, *Synthesis and Simulation Design Guide*. This design guide can be downloaded from the Xilinx website.

The prerequisites for simulating a design with the GTXE2/GTHE2 channel and common primitives are:

- A simulator with support for SecureIP models.

  SecureIP models are encrypted versions of the Verilog HDL used for implementation of the modeled block. SecureIP is an IP encryption methodology. To support SecureIP models, a Verilog LRM - IEEE Std 1364-2005 encryption compliant simulator is required.
- A mixed-language simulator for VHDL simulation.

  SecureIP models use a Verilog standard. To use them in a VHDL design, a mixed-language simulator is required. The simulator must be able to simulate VHDL and Verilog simultaneously.
- An installed GTX/GTH transceiver SecureIP model.
- The correct setup of the simulator for SecureIP use (initialization file, environment variables).
- The ability to run COMPXLIB, which compiles the simulation libraries (e.g., UNISIM, SIMPRIMS) in the correct order.
- The correct simulator resolution (Verilog).
- The user guide of the simulator and UG626, *Synthesis and Simulation Design Guide* provide a detailed list of settings for SecureIP support.

Ports and Attributes

There are no simulation-only ports on the GTXE2_COMMON and GTXE2_CHANNEL (or GTHE2_COMMON and GTHE2_CHANNEL) primitives.

GTXE2_COMMON Attributes

The GTXE2_COMMON/GTHE2_COMMON primitive has attributes intended only for simulation. Table 1-2 lists the simulation-only attributes of the GTXE2_COMMON/GTHE2_COMMON primitive. The names of these attributes start with *SIM_*.

Table 1-2: **GTXE2_COMMON//GTHE2_COMMON Simulation-Only Attributes**

Attribute	Type	Description
SIM_QPLLREFCLK_SOURCE	Binary	SIM_QPLLREFCLK_SOURCE allows for simulation before and after the pin swap changes. This allows the block to be simulated with the correct clock source both before and after the pin swap. SIM_QPLLREFCLK_SOURCE must be set to the same value as QPLLREFCLKSEL[2:0].
SIM_RESET_SPEEDUP	Boolean	If the SIM_RESET_SPEEDUP attribute is set to TRUE (default), an approximated reset sequence is used to speed up the reset time for simulations, where faster reset times and faster simulation times are desirable. If the SIM_RESET_SPEEDUP attribute is set to FALSE, the model emulates hardware reset behavior in detail.
SIM_VERSION	Real	This attribute selects the simulation version to match different steppings of silicon. The default for this attribute is 1.0.

GTXE2_CHANNEL/GTHE2_CHANNEL Attributes

The GTXE2_CHANNEL/GTHE2_CHANNEL primitive has attributes intended only for simulation. Table 1-3 lists the simulation-only attributes of the GTXE2_CHANNEL/GTHE2_CHANNEL primitive. The names of these attributes start with *SIM_*.

Table 1-3: **GTXE2_CHANNEL/GTHE2_CHANNEL Simulation-Only Attributes**

Attribute	Type	Description
SIM_CPLLREFCLK_SEL	Binary	SIM_CPLLREFCLK_SOURCE allows for simulation before and after the pin swap changes. This allows for the block to be simulated with the correct clock source both before and after the pin swap. SIM_CPLLREFCLK_SOURCE must be set to the same value as CPLLREFCLKSEL[2:0].
SIM_RESET_SPEEDUP	Boolean	If the SIM_RESET_SPEEDUP attribute is set to TRUE (default), an approximated reset sequence is used to speed up the reset time for simulations, where faster reset times and faster simulation times are desirable. If the SIM_RESET_SPEEDUP attribute is set to FALSE, the model emulates hardware reset behavior in detail.
SIM_RECEIVER_DETECT_PASS	Boolean	SIM_RECEIVER_DETECT_PASS is a string TRUE/FALSE attribute to determine if a receiver detect operation should indicate a pass or fail in simulation.
SIM_TX_EIDLE_DRIVE_LEVEL	String	SIM_TX_EIDLE_DRIVE_LEVEL can be set to 0, 1, X, or Z to allow for simulation of electrical idle and receiver detect operations using an external pull-up resistor. The default for this attribute is X.
SIM_VERSION	Real	This attribute selects the simulation version to match different steppings of silicon. The default for this attribute is 1.0.

Implementation

Functional Description

This section provides the information needed to map 7 series GTX/GTH transceivers instantiated in a design to device resources, including:

- The location of the GTX/GTH transceiver Quads on the available device and package combinations.
- The pad numbers of external signals associated with each GTX/GTH transceiver Quad.
- How GTX/GTH transceiver channels, the GTXE2_COMMON/GTHE2_COMMON primitive, and clocking resources instantiated in a design are mapped to available locations with a user constraints file (UCF).

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

This section describes how to instantiate GTX/GTH transceiver clocking components.

The position of each GTX/GTH transceiver channel and common primitive is specified by an XY coordinate system that describes the column number and the relative position within that column. In current members of the 7 series family, all GTX/GTH transceiver Quads are located in a single column along one side of the die.

For a given device/package combination, the transceiver with the coordinates *X0Y0* is always located at the lowest position of the lowest available bank.

There are two ways to create a UCF for designs that utilize the GTX/GTH transceiver. The preferred method is to use the 7 Series FPGAs Transceivers Wizard. The Wizard automatically generates UCF templates that configure the transceivers and contain placeholders for GTX/GTH transceiver placement information. The UCFs generated by the Wizard can then be edited to customize operating parameters and placement information for the application.

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

When an application requires an LC-tank PLL, a GTXE2_COMMON/GTHE2_COMMON primitive must be instantiated as shown in Figure 1-5. (The GTXE2_COMMON configuration is shown in the figure.)

Figure 1-5: **Four Channel Configuration (Reference Clock from the QPLL of GTXE2_COMMON)**

Serial Transceiver Channels by Device/Package

See UG475, *7 Series FPGAs Packaging and Pinout Specification*.

Chapter 2

Shared Features

Reference Clock Input Structure

Functional Description

The reference clock input structure is illustrated in Figure 2-1. The input is terminated internally with 50Ω on each leg to 4/5 MGTAVCC. The reference clock is instantiated in software with the IBUFDS_GTE2 software primitive. The ports and attributes controlling the reference clock input are tied to the IBUFDS_GTE2 software primitive.

Figure 2-1 shows the internal structure of the reference clock input buffer.

Figure 2-1: **Reference Clock Input Structure**

Ports and Attributes

Table 2-1 defines the reference clock input ports in the IBUFDS_GTE2 software primitive.

Table 2-1: **Reference Clock Input Ports (IBUFDS_GTE2)**

Port	Dir	Clock Domain	Description
I IB	In (pad)	N/A	These are the reference clock input ports that get mapped to GTREFCLK0P/ GTREFCLK0N and GTREFCLK1P/ GTREFCLK1P.
CEB	In	N/A	This is the active-Low asynchronous clock enable signal for the clock buffer. Setting this signal High powers down the clock buffer.
O	Out	N/A	This output drives the GTREFCLK[0/1] signals in the GTXE2_COMMON/ GTHE2_COMMON or GTXE2_CHANNEL/GTHE2_CHANNEL software primitives. It can also drive the CMT (PLL, MMCM, or BUFMRCE), BUFH, or BUFG via the HROW routing. However, either the O or ODIV2 port can be used in the design. Refer to Reference Clock Selection and Distribution, page 35 for more details.
ODIV2[(1)]	Out	N/A	This output is a divide-by-2 version of the O signal. It can drive the CMT (PLL, MMCM, or BUFMRCE), BUFH, or BUFG via the HROW routing. However, either the O or ODIV2 port can be used in the design. The selection is controlled automatically by the software depending on whether port O or ODIV2 is connected. Refer to Reference Clock Selection and Distribution, page 35 for more details.

Notes:

1. The O and ODIV2 outputs are not phase matched to each other.

Table 2-2 defines the attributes in the IBUFDS_GTE2 software primitive that configure the reference clock input.

Table 2-2: **Reference Clock Input Attributes (IBUFDS_GTE2)**

Attribute	Type	Description
CLK_RCV_TRST	Boolean	Reserved. This attribute switches the 50Ω termination resistors into the signal path. This attribute must always be set to TRUE.
CLKCM_CFG	Boolean	Reserved. This attribute switches in the termination voltage for the 50Ω termination. This attribute must always be set to TRUE.
CLKSWING_CFG[1:0]	Boolean	Reserved. This attribute controls the internal swing of the clock. This attribute must always be set to `2'b11`.

Use Modes: Reference Clock Termination

The reference clock input is to be externally AC coupled. Table 2-3 shows the pin and attribute settings required to achieve this.

Table 2-3: **Port and Attribute Settings**

Input Type	Settings
Ports	CEB = 0
Attributes	CLKRCV_TRST = TRUE CLKCM_CFG = TRUE CLKSWING_CFG = `2'b11`

Reference Clock Selection and Distribution

Functional Description

The GTX/GTH transceivers in 7 series FPGAs provide different reference clock input options. Clock selection and availability is similar to the Virtex-6 FPGA GTX/GTH transceivers, but the reference clock selection architecture supports both the LC tank (or QPLL) and ring oscillator (or CPLL) based PLLs.

Architecturally, the concept of a Quad (or Q), contains a grouping of four GTXE2_CHANNEL/GTHE2_CHANNEL primitives, one GTXE2_COMMON/GTHE2_CHANNEL primitive, two dedicated external reference clock pin pairs, and dedicated reference clock routing. The GTXE2_CHANNEL/GTHE2_CHANNEL primitive must be instantiated for each transceiver. If the high-performance QPLL is needed, the GTXE2_COMMON/GTHE2_COMMON primitive must also be instantiated. In general, the reference clock for a Quad (Q(n)) can also be sourced from the Quad below (Q(n–1)) via GTNORTHREFCLK or from the Quad above (Q(n+1)) via GTSOUTHREFCLK. For devices that support stacked silicon interconnect (SSI) technology, the reference clock sharing via GTNORTHREFCLK and GTSOUTREFCLK ports is limited within its own super logic region (SLR). See DS182, *Kintex-7 FPGAs Data Sheet: DC and Switching Characteristics* and DS183, *Virtex-7 FPGAs Data Sheet: DC and Switching Characteristics* for more information about SSI technology.

Reference clock features include:

- Clock routing for north and south bound clocks.
- Flexible clock inputs available for the QPLL or CPLL.
- Static or dynamic selection of the reference clock for the QPLL or CPLL.

Figure 1-1, page 24 shows the Quad architecture with four GTX/GTH transceivers, two dedicated reference clock pin pairs, and dedicated north or south reference clock routing. Each GTX/GTH transceiver channel in a Quad has six clock inputs available:

- Two local reference clock pin pairs, GTREFCLK0 or GTREFCLK1
- Two reference clock pin pairs from the Quads above, GTSOUTHREFCLK0 or GTSOUTHREFCLK1
- Two reference clocks pin pairs from the Quads below, GTNORTHREFCLK0 or GTNORTHREFCLK1

Figure 2-2 shows the detailed view of the reference clock multiplexer structure within a single GTXE2_COMMON/GTHE2_COMMON primitive. The QPLLREFCLKSEL port is required when multiple reference clock sources are connected to this multiplexer. A single reference clock is most commonly used. In this case, the QPLLREFCLKSEL port can be tied to `3'b001`, and the Xilinx software tools handle the complexity of the multiplexers and associated routing.

Figure 2-2: **QPLL Reference Clock Selection Multiplexer**

Similarly, Figure 2-3 shows the detailed view of the reference clock multiplexer structure within a single GTXE2_CHANNEL/GTHE2_CHANNEL primitive. The CPLLREFCLKSEL port is required when multiple reference clock sources are connected to this multiplexer. A single reference clock is most commonly used. In this case, the CPLLREFCLKSEL port can be tied to `3'b001`, and the Xilinx software tools handle the complexity of the multiplexers and associated routing.

Figure 2-3: **CPLL Reference Clock Selection Multiplexer**

Ports and Attributes

Table 2-4 through Table 2-7, page 40 define the clocking ports and attributes for GTXE2_CHANNEL/GTHE2_CHANNEL and GTXE2_COMMON/GTHE2_COMMON primitives.

Table 2-4: **GTXE2_CHANNEL/GTHE2_CHANNEL Clocking Ports**

Port	Direction	Clock Domain	Description
CPLLREFCLKSEL[2:0]	In	Async	Input to dynamically select the input reference clock to the Channel PLL. This input should be set to `3'b001` when only one clock source is connected to the Channel PLL reference clock selection multiplexer. Reset must be applied to the Channel PLL after changing the reference clock input. `000`: Reserved `001`: GTREFCLK0 selected `010`: GTREFCLK1 selected `011`: GTNORTHREFCLK0 selected `100`: GTNORTHREFCLK1 selected `101`: GTSOUTHREFCLK0 selected `110`: GTSOUTHREFCLK1 selected `111`: GTGREFCLK selected
GTGREFCLK	In	Clock	Reference clock generated by the internal FPGA logic. This input is reserved for internal testing purposes only.
GTNORTHREFCLK0	In	Clock	North-bound clock from the Quad below.
GTNORTHREFCLK1	In	Clock	North-bound clock from the Quad below.
GTREFCLK0	In	Clock	External clock driven by IBUFDS_GTE2 for the Channel PLL. For more information, refer to GTX/GTH Transceiver Reference Clock Checklist, page 302.
GTREFCLK1	In	Clock	External clock driven by IBUFDS_GTE2 for the Channel PLL. For more information, refer to GTX/GTH Transceiver Reference Clock Checklist, page 302.
GTSOUTHREFCLK0	In	Clock	South-bound clock from the Quad above.
GTSOUTHREFCLK1	In	Clock	South-bound clock from the Quad above.
QPLLCLK	In	Clock	Clock input from the high-performance Quad PLL. The user should connect QPLLOUTCLK from the GTXE2_COMMON/GTHE2_COMMON primitive to this port when the high-performance Quad PLL is used to drive the TX and/or RX channel(s).

Table 2-4: **GTXE2_CHANNEL/GTHE2_CHANNEL Clocking Ports *(Cont'd)***

Port	Direction	Clock Domain	Description
QPLLREFCLK	In	Clock	The user connects this port to the QPLLOUTREFCLK port of the GTX2_COMMON/GTH2_COMMON.
RXSYSCLKSEL[1:0]	In	Async	Selects the reference clock source to drive the RX datapath: RXSYSCLKSEL[0] = `1'b0` (CPLL) RXSYSCLKSEL[0] = `1'b1` (QPLL) Selects the reference clock source to drive RXOUTCLK: RXSYSCLKSEL[1] = `1'b0` (reference clock from GTXE2_CHANNEL or GTHE2_CHANNEL) RXSYSCLKSEL[1] = `1'b1` (reference clock from GTXE2_COMMON or GTHE2_COMMON)
TXSYSCLKSEL[1:0]	In	Async	Selects the reference clock source to drive the TX datapath: TXSYSCLKSEL[0] = `1'b0` (CPLL) TXSYSCLKSEL[0] = `1'b1` (QPLL) Selects the reference clock source to drive TXOUTCLK: TXSYSCLKSEL[1] = `1'b0` (reference clock from GTXE2_CHANNEL or GTHE2_CHANNEL) TXSYSCLKSEL[1] = `1'b1` (reference clock from GTXE2_COMMON or GTHE2_COMMON)
GTREFCLKMONITOR	Out	Clock	CPLL reference clock selection multiplexer output.

Table 2-5: **GTXE2_CHANNEL/GTHE2_CHANNEL Clocking Attribute**

Attribute	Type	Description
SIM_CPLLREFCLK_SEL	3-bit Binary	Simulation control for the channel PLL reference clock selection. This attribute must contain the same binary value as the CPLLREFCLKSEL[2:0] port.
OUTREFCLK_SEL_INV	1-bit Binary	Select signal for GTREFCLKMONITOR output. 0: Non-inverted GTREFCLKMONITOR output 1: Inverted GTREFCLKMONITOR output

Table 2-6: **GTXE2_COMMON/GTHE2_COMMON Clocking Ports**

Port	Direction	Clock Domain	Description
GTGREFCLK	In	Clock	Reference clock generated by the internal FPGA logic. This input is reserved for internal testing purposes only.
GTNORTHREFCLK0	In	Clock	North-bound clock from the Quad below.
GTNORTHREFCLK1	In	Clock	North-bound clock from the Quad below.
GTREFCLK0	In	Clock	External clock driven by IBUFDS_GTE2 for the Quad PLL. For more information, refer to GTX/GTH Transceiver Reference Clock Checklist, page 302.
GTREFCLK1	In	Clock	External clock driven by IBUFDS_GTE2 for the Quad PLL. For more information, refer to GTX/GTH Transceiver Reference Clock Checklist, page 302.
GTSOUTHREFCLK0	In	Clock	South-bound clock from the Quad above.
GTSOUTHREFCLK1	In	Clock	South-bound clock from the Quad above.
QPLLOUTCLK	Out	Clock	High-performance Quad PLL clock output. The user should connect this port to the QPLLCLK port of the GTXE2_CHANNEL/ GTHE2_CHANNEL primitive when the transmitter and/or receiver require using the high-performance Quad PLL clock source.
QPLLOUTREFCLK	Out	Clock	The user connects this port to the QPLLREFCLK port of the GTX2_CHANNEL/GTH2_CHANNEL.
QPLLREFCLKSEL[2:0]	In	Async	Input to dynamically select the input reference clock to the Quad PLL. This input should be set to `3'b001` when only one clock source is connected to the Quad PLL reference clock selection multiplexer. Reset must be applied to the Quad PLL after changing the reference clock input. `000`: Reserved `001`: GTREFCLK0 selected `010`: GTREFCLK1 selected `011`: GTNORTHREFCLK0 selected `100`: GTNORTHREFCLK1 selected `101`: GTSOUTHREFCLK0 selected `110`: GTSOUTHREFCLK1 selected `111`: GTGREFCLK selected
REFCLKOUTMONITOR	Out	Clock	QPLL reference clock selection multiplexer output.

Table 2-7: **GTXE2_COMMON/GTHE2_COMMON Clocking Attribute**

Attribute	Type	Description
SIM_QPLLREFCLK_SEL	3-bit Binary	Simulation control for the Quad PLL reference clock selection. This attribute must contain the same binary value as the QPLLREFCLKSEL[2:0] port.

External Reference Clock Use Model

Each Quad has two dedicated differential reference clock inputs that can be connected to the external clock sources. An IBUFDS_GTE2 primitive must be instantiated to use these dedicated reference clock pin pairs. The user design connects the IBUFDS_GTE2 output (O) to the GTREFCLK0 or GTREFCLK1 ports of the GTXE2_COMMON/GTHE2_COMMON or GTXE2_CHANNEL/GTHE2_CHANNEL primitive, where the reference clock selection multiplexer is located. Depending on the line rate requirement, the user design has the flexibility to use different combinations of QPLL or CPLL to drive the TX and/or RX datapath.

Figure 2-4: **External Reference Clock Use Case**

Single External Reference Clocks Use Model

Each Quad has two dedicated differential reference clock input pins (MGTREFCLK0[P/N] or MGTREFCLK1[P/N]) that can be connected to external clock sources. In a single external reference clock use model, an IBUFDS_GTE2 must be instantiated to use one of the dedicated differential reference clock sources.

Figure 2-5 shows a single external reference clock connected to multiple transceivers within a single Quad. The user design connects the IBUFDS_GTE2 output (O) to the GTREFCLK0 ports of the GTXE2_COMMON and GTXE2_CHANNEL primitives for the GTX transceiver, and GTHE2_COMMON and GTHE2_CHANNEL primitives for the GTH transceiver.

Figure 2-5: **Single External Reference Clock with Multiple Transceivers in a Single Quad**

Note: The IBUFDS_GTE2 diagram in Figure 2-5 is a simplification. The output port ODIV2 is left floating, and the input port CEB is set to logic 0.

Figure 2-6 shows a single external reference clock with multiple transceivers in multiple Quads. The user design connects the IBUFDS_GTE2 output (O) to the GTREFCLK0 ports of the GTXE2_COMMON and GTXE2_CHANNEL primitives for the GTX transceiver, and GTHE2_COMMON and GTHE2_CHANNEL primitives for the GTH transceiver. In this case, the Xilinx implementation tools make the necessary adjustments to the north/south routing as well as pin swapping necessary to route the reference clocks from one Quad to another when required.

Figure 2-6: **Single External Reference Clock with Multiple Transceivers in Multiple Quads**

Note: The IBUFDS_GTE2 diagram in Figure 2-6 is a simplification. The output port ODIV2 is left floating, and the input port CEB is set to logic 0.

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

- The number of Quads above the sourcing Quad must not exceed one.
- The number of Quads below the sourcing Quad must not exceed one.
- The total number of Quads sourced by an external clock pin pair (MGTREFCLKN/MGTREFCLKP) must not exceed three Quads (or 12 transceivers).

The maximum number of transceivers that can be sourced by a single clock pin pair is 12. Designs with more than 12 transceivers require the use of multiple external clock pins to ensure that the rules for controlling jitter are followed. When multiple clock pins are used, an external buffer can be used to drive them from the same oscillator.

Multiple External Reference Clocks Use Model

Each Quad has two dedicated differential reference clock input pins (MGTREFCLK0[P/N] or MGTREFCLK1[P/N]) that can be connected to external clock sources. In the multiple external reference clocks use model, each dedicated reference clock pin pair must instantiate its corresponding IBUFDS_GTE2 primitive to use these dedicated reference clock resources.

For the first external reference clock (MGTREFCLK0[P/N]), the user design connects the IBUFDS_GTE2 output (O) to the GTREFCLK0 ports of the GTXE2_COMMON and GTXE2_CHANNEL primitives for the GTX transceiver, and GTHE2_COMMON and GTHE2_CHANNEL primitives for the GTH transceiver. Similarly, for the second external reference clock (MGTREFCLK1[P/N]), the user design connects the IBUFDS_GTE2 output (O) to the GTREFCLK1 ports of the GTXE2_COMMON and GTXE2_CHANNEL primitives for the GTX transceiver, and GTHE2_COMMON and GTHE2_CHANNEL primitives for the GTH transceiver.

Figure 2-7 shows that the QPLL of each Quad and the CPLL of each transceiver can be sourced by either MGTREFCLK0[P/N] or MGTREFCLK1[P/N] within a single Quad. Users can set QPLLREFCLKSEL[2:0] and CPLLREFCLKSEL[2:0] to the corresponding values to select the source of the reference clock.

Figure 2-7: **Multiple GTX Transceivers with Multiple Reference Clocks in a single Quad**

Note: The IBUFDS_GTE2 diagram in Figure 2-7 is a simplification. The output port ODIV2 is left floating, and the input port CEB is set to logic 0.

The flexibility of the reference clock selection architecture allows each transceiver within a Quad to have access to the dedicated reference clocks from the Quad immediately above and below. Figure 2-8 shows an example of how one of the transceivers belonging to one Quad can access the dedicated reference clocks from another Quad by using the NORTHREFCLK and SOUTHREFCLK ports. In a situation where there are more than one reference clock options per GTX or GTH transceiver PLL, the user design is required to set the QPLLREFCLKSEL[2:0] and CPLLREFCLKSEL[2:0] based on the design requirements.

Figure 2-8: **Multiple GTX Transceivers with Multiple Reference Clocks in Different Quads**

Notes relevant to Figure 2-8:

1. QPLLREFCLKSEL[2:0]/CPLLREFCLKSEL[2:0] is used to select between GTREFCLK0/1, GTNORTHREFCLK0/1, and GTSOUTHREFCLK0/1.
2. The IBUFDS_GTE2 diagram is a simplification. The output port ODIV2 is left floating, and the input port CEB is set to logic 0.

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

- The number of Quads above the sourcing Quad must not exceed one.
- The number of Quads below the sourcing Quad must not exceed one.
- The total number of Quads sourced by an external clock pin pair (MGTREFCLKN/ MGTREFCLKP) must not exceed three Quads (or 12 transceivers).

The maximum number of transceivers that can be sourced by a single clock pin pair is 12. Designs with more than 12 transceivers require the use of multiple external clock pins to ensure that the rules for controlling jitter are followed. When multiple clock pins are used, an external buffer can be used to drive them from the same oscillator.

For multi-rate designs that require the reference clock source to be changed dynamically, the QPLLREFCLKSEL and CPLLREFCLKSEL ports are used to dynamically select the reference clock source. After the selection has been made, the user design is responsible for resetting the CPLL and QPLL via the active-High CPLLRESET and QPLLRESET ports and the subsequent initialization process described in Reset and Initialization, page 60.

Channel PLL

Functional Description

Each GTX/GTH transceiver channel contains one ring-based channel PLL (CPLL). The internal channel clocking architecture is shown in Figure 2-9. The TX and RX clock dividers can individually select the clock from the QPLL or CPLL to allow the TX and RX datapaths to operate at asynchronous frequencies using different reference clock inputs.

Figure 2-9: **Internal Channel Clocking Architecture**

The CPLL input clock selection is described in Reference Clock Selection and Distribution, page 35. The CPLL outputs feed the TX and RX clock divider blocks, which control the generation of serial and parallel clocks used by the PMA and PCS blocks. The CPLL can be shared between the TX and RX datapaths if they operate at line rates that are integral multiples of the same VCO frequency.

Figure 2-10 illustrates a conceptual view of the CPLL architecture. The input clock can be divided by a factor of M before feeding into the phase frequency detector. The feedback dividers, N1 and N2, determine the VCO multiplication ratio and the CPLL output frequency. A lock indicator block compares the frequencies of the reference clock and the VCO feedback clock to determine if a frequency lock has been achieved.

Figure 2-10: **CPLL Block Diagram**

The CPLL in the GTX transceiver has a nominal operating range between 1.6 GHz to 3.3 GHz. The CPLL in the GTH transceiver has a nominal operating range between 1.6 GHz to 5.16 GHz. The 7 Series FPGAs Transceivers Wizard chooses the appropriate CPLL settings based on application requirements.

Equation 2-1 shows how to determine the CPLL output frequency (GHz).

$$f_{PLLClkout} = f_{PLLClkin} \times \frac{N1 \times N2}{M} \qquad \textit{Equation 2-1}$$

Equation 2-2 shows how to determine the line rate (Gb/s). D represents the value of the TX or RX clock divider block in the channel.

$$f_{LineRate} = \frac{f_{PLLClkout} \times 2}{D} \qquad \textit{Equation 2-2}$$

Table 2-8 lists the allowable divider settings.

Table 2-8: **CPLL Divider Settings**

Factor	Attribute	Valid Settings
M	CPLL_REFCLK_DIV	1, 2
N2	CPLL_FBDIV	1, 2, 3, 4, 5
N1	CPLL_FBDIV_45	4, 5
D	RXOUT_DIV TXOUT_DIV	1, 2, 4, 8, 16[(1)]

1. TX/RXOUT_DIV = 16 is not supported when using CPLL.

Ports and Attributes

Table 2-9 and Table 2-10 defines the pins and attributes for the CPLL.

Table 2-9: **CPLL Ports**

Port	Direction	Clock Domain	Description
CPLLLOCKDETCLK	In	Clock	Stable reference clock for the detection of the feedback and reference clock signals to the CPLL. The input reference clock to the CPLL or any output clock generated from the CPLL (e.g., TXOUTCLK) must not be used to drive this clock. This clock is required only when using the CPLLFBCLKLOST and CPLLREFCLKLOST ports. It does not affect the CPLL lock detection, reset and power-down functions.
CPLLLOCKEN	In	Async	This port enables the CPLL lock detector. It must always be tied High.
CPLLPD	In	Async	Active-High signal that powers down the CPLL for power savings.
CPLLREFCLKSEL	In	Async	Input to dynamically select the input reference clock to the CPLL. This input should be set to `3'b001` when only one clock source is connected to the CPLL reference clock selection multiplexer. Reset must be applied to the CPLL after changing the reference clock input. `000`: Reserved `001`: GTREFCLK0 selected `010`: GTREFCLK1 selected `011`: GTNORTHREFCLK0 selected `100`: GTNORTHREFCLK1 selected `101`: GTSOUTHREFCLK0 selected `110`: GTSOUTHREFCLK1 selected `111`: GTGREFCLK selected

Table 2-9: **CPLL Ports *(Cont'd)***

Port	Direction	Clock Domain	Description
CPLLRESET	In	Async	This active-High port resets the dividers inside the PLL as well as the PLL lock indicator and status block.
CPLLFBCLKLOST	Out	CPLLLOCKDETCLK	A High on this signal indicates the feedback clock from the CPLL feedback divider to the phase frequency detector of the CPLL is lost.
CPLLLOCK	Out	Async	This active-High PLL frequency lock signal indicates that the PLL frequency is within predetermined tolerance. The transceiver and its clock outputs are not reliable until this condition is met.
CPLLREFCLKLOST	Out	CPLLLOCKDETCLK	A High on this signal indicates the reference clock to the phase frequency detector of the CPLL is lost.
TSTOUT	Out	Async	Reserved.
GTRSVD	In	Async	Reserved.
PCSRSVDIN	In	Async	Reserved.
PCSRSVDIN2	In	Async	Reserved.
PMARSVDIN	In	Async	Reserved.
PMARSVDIN2	In	Async	Reserved.
TSTIN	In	Async	Reserved.

Table 2-10: **CPLL Attributes**

Attribute	Type	Description
CPLL_CFG	24-bit Hex	Reserved. Configuration setting for the CPLL. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
CPLL_FBDIV	Integer	CPLL feedback divider N2 settings as shown in Figure 2-9, page 46. Valid settings are 1, 2, 3, 4, and 5.
CPLL_FBDIV_45	Integer	CPLL reference clock divider N1 settings as shown in Figure 2-9, page 46. Valid settings are 4 and 5.
CPLL_INIT_CFG	24-bit Hex	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.

Table 2-10: **CPLL Attributes *(Cont'd)***

Attribute	Type	Description
CPLL_LOCK_CFG	16-bit Hex	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
CPLL_REFCLK_DIV	Integer	CPLL reference clock divider M settings as shown in Figure 2-9, page 46. Valid settings are 1 and 2.
RXOUT_DIV(1)	Integer	CPLL/QPLL output clock divider D for the RX datapath as shown in Figure 2-9, page 46. Valid settings are 1, 2, 4, 8, and 16.
TXOUT_DIV(1)	Integer	CPLL/QPLL output clock divider D for the TX datapath as shown in Figure 2-9, page 46. Valid settings are 1, 2, 4, 8, and 16.
SATA_CPLL_CFG	String	Reserved. SATA application specific setting. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
SIM_CPLLREFCLK_SEL	3-bit Binary	Simulation control for the Quad PLL reference clock selection. This attribute must contain the same binary value as the CPLLREFCLKSEL[2:0] port.
PMA_RSV3	2-bit Binary	Reserved.

Notes:

1. TXOUT_DIV/RXOUT_DIV = 16 is not supported when using the CPLL.

CPLL Settings for Common Protocols

Table 2-11 shows example CPLL divider settings for several standard protocols. This table is not inclusive of every combination of REFCLK frequencies and divider settings.

Table 2-11: **CPLL Divider Settings for Common Protocols**

Standard	Line Rate [Gb/s]	Internal Data Width [16b/20b/32b/40b]	PLL Frequency [GHz]	REFCLK (Typical) [MHz]	Using Typical REFCLK Frequency			
					N1	N2	D	M
Fibre Channel (Single Rate)	4.25	20b	2.125	212.5	5	2	1	1
	2.125	20b	2.125	106.25	5	4	2	1
	1.0625	20b	2.125	106.25	5	4	4	1
Fibre Channel (Multi-Rate)	4.25	20b	2.125	212.5	5	2	1	1
	2.125	20b	2.125	212.5	5	2	2	1
	1.0625	20b	2.125	212.5	5	2	4	1
XAUI	3.125	20b	3.125	156.25	5	4	2	1
RXAUI	6.25	20b	3.125	156.25	5	4	1	1
GigE	1.25	20b	2.5	125	5	4	4	1

Table 2-11: **CPLL Divider Settings for Common Protocols *(Cont'd)***

Standard	Line Rate [Gb/s]	Internal Data Width [16b/20b/32b/40b]	PLL Frequency [GHz]	REFCLK (Typical) [MHz]	Using Typical REFCLK Frequency			
					N1	N2	D	M
Aurora (Single Rate)	6.25	20b	3.125	312.5	5	2	1	1
	5	20b	2.5	250	5	2	1	1
	3.125	20b	3.125	156.25	5	4	2	1
	2.5	20b	2.5	125	5	4	2	1
	1.25	20b	2.5	125	5	4	4	1
Aurora (Multi-Rate)	6.25	20b	3.125	312.5	5	2	1	1
	5	20b	2.5	312.5	4	2	1	1
	3.125	20b	3.125	312.5	5	2	2	1
	2.5	20b	2.5	312.5	4	2	2	1
	1.25	20b	2.5	312.5	4	2	4	1
Aurora 64B/66B	3.125	32b	3.125	156.25	5	4	2	1
Serial RapidIO (Single Rate)	3.125	20b	3.125	156.25	5	4	2	1
	2.5	20b	2.5	125	5	4	2	1
	1.25	20b	2.5	125	5	4	4	1
Serial RapidIO (Multi-Rate)	3.125	20b	3.125	156.25	5	4	2	1
	2.5	20b	2.5	156.25	4	4	2	1
	1.25	20b	2.5	156.25	4	4	4	1
SATA	3	20b	3	150	5	4	2	1
	1.5	20b	3	150	5	4	4	1
PCIe Optimal Jitter	5	20b	2.5	250	5	2	1	1
	2.5	20b	2.5	250	5	2	2	1
	5	20b	2.5	125	5	4	1	1
	2.5	20b	2.5	125	5	4	2	1
PCIe 100 MHz REFCLK	5	20b	2.5	100	5	5	1	1
	2.5	20b	2.5	100	5	5	2	1
CPRI (Multi-Rate)	3.072	20b	3.072	122.88	5	5	2	1
	2.4576	20b	2.4576	122.88	5	4	2	1
	1.2288	20b	2.4576	122.88	5	4	4	1
	0.6144	20b	2.4576	122.88	5	4	8	1

Table 2-11: **CPLL Divider Settings for Common Protocols *(Cont'd)***

Standard	Line Rate [Gb/s]	Internal Data Width [16b/20b/32b/40b]	PLL Frequency [GHz]	REFCLK (Typical) [MHz]	Using Typical REFCLK Frequency			
					N1	N2	D	M
OBSAI (Multi-Rate)	6.144	20b	3.072	153.6	5	4	1	1
	3.072	20b	3.072	153.6	5	4	2	1
	1.536	20b	3.072	153.6	5	4	4	1
	0.768	20b	3.072	153.6	5	4	8	1
3G-SDI HD-SDI (Multi-Rate)	2.97	20b	2.97	148.5	5	4	2	1
	1.485	20b	2.97	148.5	5	4	4	1
Interlaken	6.25	16b	3.125	312.5	5	2	1	1
	4.25	16b	2.125	212.5	5	2	1	1
	3.125	16b	3.125	156.25	5	4	2	1
SFI-5	3.125	16b	3.125	195.3125	4	4	2	1
OC-48	2.48832	16b	2.48832	155.52	4	4	2	1
OC-12	0.62208	16b	2.48832	155.52	4	4	8	1
OTU-1	2.666057	16b	2.666057	166.6286	4	4	2	1
CEI 6.25	6.25	20b	3.125	390.625	4	2	1	1

Some protocols are shown twice as a single-rate configuration and a multi-rate configuration. In single-rate configurations, only one line rate is required, and the reference clock is optimized for that particular line rate. In multi-rate configurations, a reference clock is selected for the highest line rate, and the appropriate dividers are selected to support the lower line rates.

The general guidelines for the maximum, typical, and minimum frequencies for a given protocol and line rate are:

- Maximum VCO frequency is selected to use the minimum PLL feedback divider settings. This option usually provides the highest jitter performance.
- Typical reference clock frequency is selected to limit the PLL multiplication to either 8 or 10 depending on the protocol.
- For lower line rate operation, the minimum frequency is selected to allow for a PLL multiplication of 16 or 20.
- Performance impact needs to be carefully considered if a reference clock below the typical recommended frequency is used. Refer to DS182, *Kintex-7 FPGAs Data Sheet: DC and Switching Characteristics* and DS183, *Virtex-7 FPGAs Data Sheet: DC and Switching Characteristics* for the minimum and maximum reference clock frequencies.

Use Modes

Dynamically Changing CPLL settings

The following describes the sequence of events to dynamically change CPLL settings. It pertains only to changes for the CPLL:

1. When ready (all valid data is transmitted or received), provide changes via port CPLLREFCLKSEL and/or DRP to the attributes listed in Table 2-10.
2. Follow the reset guidelines as detailed in CPLL Reset, page 63.
3. When the CPLL has locked, assert GTTXRESET and/or GTRXRESET and follow the guidelines as detailed in GTX/GTH Transceiver TX Reset in Response to GTTXRESET Pulse, page 67 and GTX/GTH Transceiver RX Reset in Response to GTRXRESET Pulse, page 79.
4. Continue with transceiver operation.

Dynamically switching from CPLL to QPLL

The following describes the sequence of events to dynamically change from using a CPLL to a QPLL:

1. Power up the QPLL by deasserting the ports QPLLPD and QPLLRESET. Wait until the port QPLLLOCK = 1.
2. Assert ports GTTXRESET and/or GTRXRESET. Set port TXSYSCLKSEL[0] = 1'b1 and RXSYSCLKSEL[0] = 1'b1. Assert port [TX/RX]USERRDY.
3. Deassert GTTXRESET and/or GTRXRESET. Wait for TXRESETDONE = 1'b1 and RXRESETDONE = 1'b1.
4. Power down the CPLL by asserting CPLLRESET and CPLLPD to save power.
5. Continue with transceiver operation.

Quad PLL

Functional Description

Each Quad contains one LC-based PLL, referred to as the Quad PLL (QPLL). The QPLL can be shared by the serial transceiver channels within the same Quad, but cannot be shared by channels in other Quads. Use of the QPLL is required when operating the channels at line rates above the CPLL operating range. The GTXE2_COMMON primitive encapsulates the GTX QPLL and must be instantiated when the GTX QPLL is used. Similarly, the GTHE2_COMMON primitive encapsulates the GTH QPLL and must be instantiated when the GTH QPLL is used.

The QPLL input reference clock selection is described in Reference Clock Selection and Distribution, page 35. The QPLL outputs feed the TX and RX clock divider blocks of each serial transceiver channel within the same Quad, which control the generation of serial and parallel clocks used by the PMA and PCS blocks. Figure 2-9, page 46 shows the internal channel clocking architecture.

Figure 2-11 illustrates a conceptual view of the QPLL architecture. The input clock can be divided by a factor of M before it is fed into the phase frequency detector. The feedback divider N determines the VCO multiplication ratio. The QPLL output frequency is half of the VCO frequency. A lock indicator block compares the frequencies of the reference clock and the VCO feedback clock to determine if a frequency lock has been achieved.

Figure 2-11: **QPLL Detail**

The QPLL VCO operates within two different frequency bands. Table 2-12 describes the nominal operating range for these bands. For more information, see the specific device data sheet at http://www.xilinx.com/support/documentation/7_series.htm#156339.

Table 2-12: **QPLL Nominal Operating Range**

Transceiver	Frequency (GHz)	
GTX	Lower Band	5.93–8.0
	Upper Band	9.8–12.5
GTH	8.0–13.1	

When the lower band VCO is selected, the upper band VCO is automatically powered down and vice versa. The 7 Series FPGAs Transceivers Wizard chooses the appropriate band and QPLL settings based on application requirements.

Equation 2-3 shows how to determine the PLL output frequency (GHz).

$$f_{PLLClkout} = f_{PLLClkin} \times \frac{N}{M \times 2} \qquad \textit{Equation 2-3}$$

Equation 2-4 shows how to determine the line rate (Gb/s). D represents the value of the TX or RX clock divider block in the channel. See Table 2-8, page 47 for the valid settings for D.

$$f_{LineRate} = \frac{f_{PLLClkout} \times 2}{D} \qquad \textit{Equation 2-4}$$

Table 2-13 lists the allowable divider values.

Table 2-13: **QPLL Divider Settings**

Factor	Attribute	Valid Settings
M	QPLL_REFCLK_DIV	1, 2, 3, 4
N	QPLL_FBDIV QPLL_FBDIV_RATIO	16, 20, 32, 40, 64, 66, 80, 100 (see Table 2-16)
D	RXOUT_DIV TXOUT_DIV	1, 2, 4, 8, 16

Ports and Attributes

Table 2-14 and Table 2-15, page 57 define the pins and attributes for the QPLL.

Table 2-14: **QPLL Ports**

Port	Direction	Clock Domain	Description
QPLLDMONITOR	Out	Async	Reserved.
QPLLFBCLKLOST	Out	QPLLLOCKDETCLK	A High on this signal indicates the feedback clock from the QPLL feedback divider to the phase frequency detector of the QPLL is lost.
QPLLLOCK	Out	Async	This active-High PLL frequency lock signal indicates that the PLL frequency is within predetermined tolerance. The transceiver and its clock outputs are not reliable until this condition is met.
QPLLLOCKDETCLK	In	Clock	Stable reference clock for the detection of the feedback and reference clock signals to the QPLL. The input reference clock to the QPLL or any output clock generated from the QPLL (e.g., TXOUTCLK) must not be used to drive this clock. This clock is required only when using the QPLLFBCLKLOST and QPLLREFCLKLOST ports. It does not affect the QPLL lock detection, reset, and power-down functions.
QPLLLOCKEN	In	Async	This port enables the QPLL lock detection circuitry. It must always be tied High.
QPLLOUTCLK	Out	N/A	QPLL output clock. The user should connect this clock signal to QPLLCLK in the GTXE2_CHANNEL primitive (or GTHE2_CHANNEL primitive) when using the GTX (or GTH) transceiver.
QPLLOUTRESET	In	Async	Reserved. Tied Low.
QPLLPD	In	Async	Active-High signal that powers down the QPLL for power savings.

Table 2-14: **QPLL Ports *(Cont'd)***

Port	Direction	Clock Domain	Description
QPLLREFCLKLOST	Out	QPLLLOCKDETCLK	A High on this signal indicates the reference clock to the phase frequency detector of the QPLL is lost.
QPLLREFCLKSEL	In	Async	Input to dynamically select the input reference clock to the QPLL. This input should be set to `3'b001` when only one clock source is connected to the QPLL reference clock selection multiplexer. Reset must be applied to the QPLL after changing the reference clock input. `000`: Reserved `001`: GTREFCLK0 selected `010`: GTREFCLK1 selected `011`: GTNORTHREFCLK0 selected `100`: GTNORTHREFCLK1 selected `101`: GTSOUTHREFCLK0 selected `110`: GTSOUTHREFCLK1 selected `111`: GTGREFCLK selected
QPLLRESET	In	Async	This active-High port resets the dividers inside the PLL as well as the PLL lock indicator and status block.
QPLLRSVD1[15:0]	In	-	Reserved.
QPLLRSVD2[4:0]	In	-	Reserved.
BGBYPASSB	In	Async	Reserved. This port must be set to `1'b1`. This value should not be modified.
BGMONITORENB	In	Async	Reserved. This port must be set to `1'b1`. This value should not be modified.
BGPDB	In	Async	Reserved. This port must be set to `1'b1`. This value should not be modified.
BGRCALOVRD[4:0]	In	Async	Reserved. This port must be set to `5'b11111`. This value should not be modified.

Table 2-14: **QPLL Ports *(Cont'd)***

Port	Direction	Clock Domain	Description
RCALENB	In	Async	Reserved. This port must be set to `1'b1`. This value should not be modified.
PMARSVD	In	Async	Reserved.

Table 2-15: **QPLL Attributes**

Attribute	Type	Description
QPLL_CFG	27-bit Hex	Reserved. This attribute is the configuration setting for the QPLL. QPLL_CFG[6] selects the QPLL frequency band. 0: Upper band 1: Lower band The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
QPLL_CLKOUT_CFG	4-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
QPLL_COARSE_FREQ_OVRD	6-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
QPLL_COARSE_FREQ_OVRD_EN	1-bit Binary	Reserved. This attribute must be set to 0.
QPLL_CP	10-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
QPLL_CP_MONITOR_EN	1-bit Binary	Reserved. This attribute must be set to 0.
QPLL_DMONITOR_SEL	1-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
QPLL_FBDIV	10-bit Binary	QPLL feedback divider N settings as shown in Figure 2-11, page 54. Refer to Table 2-16 for its configuration.
QPLL_FBDIV_MONITOR_EN	1-bit Binary	Reserved. This attribute must be set to 0.
QPLL_FBDIV_RATIO	1-bit Binary	Refer to Table 2-16.
QPLL_INIT_CFG	23-bit Hex	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.

Table 2-15: **QPLL Attributes *(Cont'd)***

Attribute	Type	Description
QPLL_LOCK_CFG	16-bit Hex	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
QPLL_LPF	4-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
QPLL_REFCLK_DIV	Integer	QPLL reference clock divider M settings as shown in Figure 2-11. Valid settings are 1, 2, 3, and 4.
SIM_QPLLREFCLK_SEL	3-bit Binary	Simulation control for the Quad PLL reference clock selection. This attribute must contain the same binary value as the QPLLREFCLKSEL[2:0] port.
RXOUT_DIV	Integer	QPLL/CPLL output clock divider D for the RX datapath as shown in Figure 2-9. Valid settings are 1, 2, 4, 8, and 16.
TXOUT_DIV	Integer	QPLL/CPLL output clock divider D for the TX datapath as shown in Figure 2-9. Valid settings are 1, 2, 4, 8, and 16.
COMMON_CFG	32-bit Binary	Reserved.

Table 2-16: **N Divider Configuration**

N	QPLL_FBDIV_RATIO	QPLL_FBDIV[9:0]
16	1	0000100000
20	1	0000110000
32	1	0001100000
40	1	0010000000
64	1	0011100000
66	0	0101000000
80	1	0100100000
100	1	0101110000

QPLL Settings for Common Protocols

Table 2-17 shows example QPLL divider settings for several standard protocols. This table is not inclusive of every combination of REFCLK frequencies and divider settings.

Table 2-17: **QPLL Divider Settings for Common Protocols**

Standard	Line Rate [Gb/s]	Internal Data Width [16b/20b/32b/40b]	PLL Frequency [GHz]	QPLL [Upper/Lower Band]	REFCLK (Typical) [MHz]	Using Typical REFCLK Frequency			
						N (QPLL_FBDIV, QPLL_FBDIV_RATIO)	RXOUT_DIV (D)	TXOUT_DIV (D)	M (QPLL_REFCLK_DIV)
10GBASE-R (156.25 MHz)	10.3125	32b	10.3125	Upper	156.25	66	1	1	1
Interlaken 10.3125 (161.13 MHz)	10.3125	32b	10.3125	Upper	161.13	64	1	1	1
OC-192 (9.953 Gb/s, 155.516 MHz)	9.953024	32b	9.953024	Upper	155.516	64	1	1	1
PCIe Gen3	8	32b	8	Lower	100	80	1	1	1
CEI 6.25[(1)]	6.25	20b	6.25	Lower	390.625	16	1	1	1
	6.25	20b	6.25	Lower	156.25	40	1	1	1
Standard: SFP+ (SFF-8431, SFI)	9.8304[(2)]	32b	9.8304	Upper	122.88	80	1	1	1
	9.95328	32b	9.95328	Upper	155.52	64	1	1	1
	10.3125	32b	10.3125	Upper	156.25	66	1	1	1
	10.5187	32b	10.5187	Upper	164.355	64	1	1	1
	11.1	32b	11.1	Upper	173.4375	64	1	1	1

Notes:

1. The divider settings for CEI 6.25 in this table apply to GTX transceivers only.
2. Line rate used for CPRI over SFP+ applications.

Use Modes

Dynamically Changing QPLL settings

The following describes the sequence of events to dynamically change QPLL settings. This pertains only to changes for the QPLL:

1. When ready (all valid data is transmitted or received), provide changes via port QPLLREFCLKSEL and/or DRP to the attributes listed in Table 2-10.
2. Follow the reset guidelines as detailed in QPLL Reset, page 63.
3. When the QPLL has locked, assert GTTXRESET and/or GTRXRESET and follow the guidelines as detailed in GTX/GTH Transceiver TX Reset in Response to GTTXRESET Pulse, page 67 and GTX/GTH Transceiver RX Reset in Response to GTRXRESET Pulse, page 79.
4. Continue with transceiver operation.

Dynamically Switching from QPLL to CPLL

The following describes the sequence of events to dynamically change from using a QPLL to a CPLL:

1. Power up the CPLL by deasserting the ports CPLLPD and CPLLRESET. Wait until the port CPLLLOCK = 1.
2. Assert ports GTTXRESET and/or GTRXRESET. Set port TXSYSCLKSEL[0] = 1'b0 and RXSYSCLKSEL[0] = 1'b0. Assert port [TX/RX]USERRDY.
3. Deassert GTTXRESET and/or GTRXRESET. Wait for TXRESETDONE = 1'b1 and RXRESETDONE = 1'b1.
4. Power down the QPLL by asserting QPLLRESET and QPLLPD to save power.
5. Continue with transceiver operation.

Reset and Initialization

The GTX/GTH transceiver must be initialized after FPGA device power-up and configuration before it can be used. The GTX/GTH transmitter (TX) and receiver (RX) can be initialized independently and in parallel as shown in Figure 2-12. The GTX/GTH transceiver TX and RX initialization comprises two steps:

1. Initializing the associated PLL driving TX/RX
2. Initializing the TX and RX datapaths (PMA + PCS)

The GTX/GTH transceiver TX and RX can receive a clock from either the QPLL or the CPLL. The associated PLL (QPLL/CPLL) used by the TX and RX must be initialized first before TX and RX initialization. Any PLL used by the TX and RX is reset individually and its reset operation is completely independent from all TX and RX resets. The TX and RX datapaths must be initialized only after the associated PLL is locked.

Figure 2-12: **GTX/GTH Transceiver Initialization Overview**

The GTX/GTH transceiver TX and RX use a state machine to control initialization process. They are partitioned into a few reset regions. The partition allows the reset state machine to control the reset process in a sequence that the PMA can be reset first and the PCS can be reset after the assertion of the TXUSERRDY or RXUSERRDY. It also allows the PMA, the PCS, and functional blocks inside them to be reset individually when needed during normal operation.

The GTX/GTH transceiver offers two types of reset: initialization and component.

- Initialization Reset: This reset is used for complete GTX/GTH transceiver initialization. It must be used after device power-up and configuration. During normal operation, when necessary, GTTXRESET and GTRXRESET can also be used to reinitialize the GTX/GTH transceiver TX and RX. GTTXRESET is the initialization reset port for the GTX/GTH transceiver TX. GTRXRESET is the initialization reset port for the GTX/GTH transceiver RX.
- Component Reset: This reset is used for special cases and specific subsection resets while the GTX/GTH transceiver is in normal operation. TX component reset ports include TXPMARESET and TXPCSRESET. RX component reset ports include RXPMARESET, RXDFELPMRESET, EYESCANRESET, RXPCSRESET, RXBUFRESET, and RXOOBRESET.

For major coverage differences between initialization and component resets, refer to Table 2-26 for the GTX/GTH transceiver TX and Table 2-30 and Table 2-31 for the GTX/GTH transceiver RX.

All reset ports described in this section initiate the internal reset state machine when driven High. The internal reset state machines are held in the reset state until these same

reset ports are driven Low. These resets are all asynchronous. The guideline for the pulse width of these asynchronous resets is one period of the reference clock, unless otherwise noted.

Note: Reset ports should not be used for the purpose of power down. For details on proper power down usage, refer to Power Down, page 86.

Reset Modes

The GTX/GTH transceiver RX resets can operate in two different modes: Sequential mode and single mode. The GTX/GTH transceiver TX resets can operate only in sequential mode.

- Sequential mode: The reset state machine starts with an initialization or component reset input driven High and proceeds through all states after the requested reset states in the reset state machine, as shown in Figure 2-15 for the GTX/GTH transceiver TX or Figure 2-20 for the GTX/GTH transceiver RX until completion. The completion of sequential mode reset flow is signaled when (TX/RX)RESETDONE transitions from Low to High.
- Single mode: The reset state machine only executes the requested component reset independently for a predetermined time set by its attribute. It does not process any state after the requested state, as shown in Figure 2-20 for the GTX/GTH transceiver RX. The requested reset can be any component reset to reset the PMA, the PCS, or functional blocks inside them. The completion of a single mode reset is signaled when RXRESETDONE transitions from Low to High.

The GTX/GTH transceiver initialization reset must use sequential mode. All component resets can be operated in either sequential mode or single mode, except for TX resets, which can only operate in sequential mode.

The GTX/GTH transceiver uses GTRESETSEL to select between sequential reset mode and single reset mode. Table 2-18 provides configuration details that apply to both the GTX/GTH transceiver TX and GTX/GTH transceiver RX. Reset modes have no impact on CPLL and QPLL resets. During normal operation, the GTX/GTH transceiver TX or GTX/GTH transceiver RX can be reset by applications in either sequential mode or single mode (GTX/GTH transceiver RX only), which provides flexibility to reset a portion of the GTX/GTH transceiver. When using either sequential mode or single mode, RESETOVRD must be driven Low, as shown in Table 2-18. RESETOVRD and GTRESETSEL must be set to the desired value 300–500 ns before the assertions of any reset.

Table 2-18: **GTX/GTH Transceiver Reset Modes Operation**

Operation Mode	RESETOVRD	GTRESETSEL
Sequential Mode	0	0
Single Mode	0	1

Table 2-19: **GTX/GTH Transceiver Reset Mode Ports**

Port	Dir	Clock Domain	Description
GTRESETSEL	In	Async	Reset mode enable port. Low: Sequential mode (recommended). High: Single mode.
RESETOVRD	In	Async	Reserved. Must be tied to ground.

CPLL Reset

The CPLL must be reset before it can be used. Each GTX/GTH transceiver channel has three dedicated ports for CPLL reset. As shown in Figure 2-13, CPLLRESET is an input that resets the CPLL. CPLLLOCK is an output that indicates the reset process is done. The guideline for this asynchronous CPLLRESET pulse width is one period of the reference clock. The real CPLL reset generated by the internal GTX/GTH transceiver circuit is much longer than the CPLLRESET High pulse duration. The time required for the CPLL to lock is affected by a few factors, such as bandwidth setting and clock frequency.

Figure 2-13: **CPLL Reset Timing Diagram**

Table 2-20: **CPLL Reset Port**

Port	Dir	Clock Domain	Description
CPLLRESET	In	Async	This port is driven High and then deasserted to start the CPLL reset.
CPLLLOCK	Out	Async	This active-High CPLL frequency lock signal indicates that the CPLL frequency is within a predetermined tolerance. The GTX/GTH transceiver and its clock outputs are not reliable until this condition is met.
CPLLLOCKEN	In	Async	This active-High signal enables the CPLL lock detector.

Table 2-21: **CPLL Reset Attributes**

Attribute	Type	Description
CPLLRESET_TIME (CPLL_INIT_CFG[9:0])	10-bit Binary	Reserved. Represents the time duration to apply internal CPLL reset. Must be a non-zero value. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.

QPLL Reset

The QPLL must be reset before it can be used. Each GTX/GTH transceiver Quad has three dedicated ports for the QPLL reset. As shown in Figure 2-14, QPLLRESET is an input that resets the QPLL. QPLLLOCK is an output that indicates the reset process is done. The guideline for this asynchronous QPLLRESET pulse width is one period of the reference clock. The real QPLL reset generated by the internal GTX/GTH transceiver circuit is much

longer than the QPLLRESET High pulse duration. The time required for the QPLL to lock is affected by a few factors, such as bandwidth setting and clock frequency.

Figure 2-14: **QPLL Reset Timing Diagram**

Table 2-22: **QPLL Reset Port**

Port	Dir	Clock Domain	Description
QPLLRESET	In	Async	This port is driven High and then deasserted to start the QPLL reset.
QPLLLOCK	Out	Async	This active-High QPLL frequency lock signal indicates that the QPLL frequency is within a predetermined tolerance. The GTX/GTH transceiver and its clock outputs are not reliable until this condition is met.
QPLLLOCKEN	In	Async	This active-High signal enables the QPLL lock detector.

Table 2-23: **QPLL Reset Attributes**

Attribute	Type	Description
QPLLRESET_TIME (QPLL_INIT_CFG[9:0])	10-bit Binary	Reserved. Represents the time duration to apply internal QPLL reset. Must be a non-zero value. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.

TX Initialization and Reset

The GTX/GTH transceiver TX uses a reset state machine to control the reset process. The GTX/GTH transceiver TX is partitioned into two reset regions, TX PMA and TX PCS. The partition allows TX initialization and reset to be operated only in sequential mode, as shown in Figure 2-15.

The initializing TX must use GTTXRESET in sequential mode. Activating GTTXRESET input can automatically trigger a full asynchronous TX reset. The reset state machine executes the reset sequence, as shown in Figure 2-15, covering the whole TX PMA and TX PCS. During normal operation, when needed, sequential mode allows the user to reset TX from activating TXPMARESET and continue the reset state machine until TXRESETDONE transitions from Low to High.

The TX reset state machine does not reset the PCS until TXUSERRDY is detected High. The user should drive TXUSERRDY High after these conditions are met:

1. All clocks used by the application including TXUSRCLK/TXUSRCLK2 are shown as stable or locked when the PLL or MMCM is used.
2. The user interface is ready to transmit data to the GTX/GTH transceiver.

Figure 2-15: **GTX/GTH Transceiver TX Reset State Machine Sequence**

Ports and Attributes

Table 2-24 lists ports required by TX initialization process.

Table 2-24: **TX Initialization and Reset Ports**

Port	Dir	Clock Domain	Description
GTTXRESET	In	Async	This port is driven High and then deasserted to start the full TX reset sequence. The time required for the reset sequence is to be determined.
TXPMARESET	In	Async	This port is used to reset the TX PMA. It is driven High and then deasserted to start the TX PMA reset process. In sequential mode, activating this port resets both the TX PMA and the TX PCS.
TXPCSRESET	In	Async	This port is used to reset the TX PCS. It is driven High and then deasserted to start the PCS reset process. In sequential mode, activating this port only resets the TX PCS.
TXUSERRDY	In	Async	This port is driven High from the user's application when TXUSRCLK and TXUSRCLK2 are stable. For example, if an MMCM is used to generate both TXUSRCLK and TXUSRCLK2, then the MMCM lock signal can be used here.

Table 2-24: **TX Initialization and Reset Ports *(Cont'd)***

Port	Dir	Clock Domain	Description
TXRESETDONE	Out	TXUSRCLK2	This active-High signal indicates the GTX/GTH transceiver TX has finished reset and is ready for use. This port is driven Low when GTTXRESET goes High and is not driven High until the GTX/GTH transceiver TX detects TXUSERRDY High.
CFGRESET	In	Async	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
TXPMARESETDONE	Out	Async	GTH transceiver: This active-high signal indicates GTH TX PMA reset is complete. This port is driven Low when GTTXRESET or TXPMARESET is asserted.
PCSRSVDOUT	Out	Async	Reserved.

Table 2-25 lists attributes required by GTX/GTH transceiver TX initialization. In general cases, the reset time required by the TX PMA or the TX PCS varies depending on line rate. The factor affecting PMA reset time and PCS reset time are user-configurable attributes TXPMARESET_TIME and TXPCSRESET_TIME.

Table 2-25: **TX Initialization and Reset Attributes**

Attribute	Type	Description
TXPMARESET_TIME	5-bit Binary	Reserved. Represents the time duration to apply a TX PMA reset. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used. Must be a non-zero value when GTTXRESET or TXPMARESET is used to initiate the reset process.
TXPCSRESET_TIME	5-bit Binary	Reserved. Represents the time duration to apply a TX PCS reset. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used. Must be a non-zero value when TXPCSRESET is used to initiate the reset process.

GTX/GTH Transceiver TX Reset in Response to Completion of Configuration

The TX reset sequence shown in Figure 2-15 is not automatically started to follow global GSR. It must meet these conditions:

1. GTRESETSEL must be Low to use sequential mode.
2. GTTXRESET must be used.
3. TXPMARESET and TXPCSRESET must be constantly driven Low during the entire reset process before TXRESETDONE is detected High.
4. GTTXRESET cannot be driven Low until the associated PLL is locked.

If the reset mode is defaulted to sequential mode upon configuration, then C/QPLLRESET and GTTXRESET can be asserted after waiting for a minimum of 500 ns after configuration is complete.

If the reset mode is defaulted to single mode, then the user must:

1. Wait a minimum of 500 ns after configuration is complete.
2. Change reset mode to Sequential mode.
3. Wait another 300-500 ns.
4. Assert C/QPLLRESET and GTTXRESET.

It is recommended to use the associated PLLLOCK from either CPLL or QPLL to release GTTXRESET from High to Low as shown in Figure 2-16. The TX reset state machine waits when GTTXRESET is detected High and starts the reset sequence until GTTXRESET is released Low.

Figure 2-16: **GTX/GTH Transmitter Initialization after FPGA Configuration**

GTX/GTH Transceiver TX Reset in Response to GTTXRESET Pulse

The GTX/GTH transceiver allows the user to reset the entire TX completely at any time by sending GTTXRESET an active-High pulse. TXPMARESET_TIME and TXPCSRESET_TIME can be set statically or reprogrammed through DRP ports to adjust the required reset time before applying GTTXRESET. These conditions must be met when using GTTXRESET:

1. GTRESETSEL must be driven Low to use sequential mode.
2. TXPMARESET and TXPCSRESET must be driven constantly Low during the entire reset process before TXRESETDONE is detected High.
3. The associated PLL must indicate locked.
4. The guideline for this asynchronous GTTXRESET pulse width is one period of the reference clock.

Figure 2-17: **GTX/GTH Transmitter Reset after GTTXRESET Pulse**

GTX/GTH Transceiver TX Component Reset

TX PMA and TX PCS can be reset individually. GTTXRESET must be driven constantly Low during the TXPMARESET or TXPCSRESET process before finish.

Driving TXPMARESET from High to Low starts the PMA reset process. TXPCSRESET must be driven constantly Low during the TXPMARESET process. In sequential mode (Figure 2-18), the reset state machine automatically starts the PCS reset after finishing the PMA reset, if TXUSERRDY is High.

Figure 2-18: **TXPMARESET in Sequential Mode**

Driving TXPCSRESET from High to Low starts the PCS reset process when TXUSERRDY is High. TXPMARESET must be driven constantly Low when the PCS is in reset process. In sequential mode, the reset state machine only resets the PCS (see Figure 2-19).

Figure 2-19: **TXPCSRESET in Sequential Mode**

Table 2-26 summarizes all resets available to the GTX/GTH transceiver TX and components affected by them in sequential mode. Using TXPMARESET in sequential mode resets everything covered by GTTXRESET except the TX reset state machine.

Table 2-26: **TX Initialization Reset and Component Reset Coverage in Sequential Mode**

	Functional Blocks	GTTXRESET	TXPMARESET	TXPCSRESET
TX PCS	FPGA TX Fabric Interface	✓	✓	✓
	TX 8B/10B Encoder	✓	✓	✓
	TX Gearbox	✓	✓	✓
	TX Buffer	✓	✓	✓
	TX Pattern Generator	✓	✓	✓
	TX Polarity Control	✓	✓	✓
	TX Out-of-Band Signaling	✓	✓	✓
	TX Reset FSM	✓		
TX PMA	TX Configuration Driver	✓	✓	
	TX Receiver Detect for PCI Express Designs	✓	✓	
	TX PISO	✓	✓	

Table 2-27 lists the recommended resets for various situations.

Table 2-27: **Recommended Resets for Common Situations**

Situation	Components to be Reset	Recommended Reset[(1)]
After power up and configuration	Entire TX	GTTXRESET
After turning on a reference clock to the CPLL/QPLL being used	Entire TX	GTTXRESET
After changing the reference clock to the CPLL/QPLL being used	Entire TX	GTTXRESET
After assertion/deassertion of CPLLPD or QPLLPD for the PLL being used	Entire TX	GTTXRESET
After assertion/deassertion of TXPD[1:0]	Entire TX	GTTXRESET
TX rate change	TX PCS	Reset performed automatically
TX parallel clock source reset	TX PCS	TXPCSRESET

Notes:

1. The recommended reset has the smallest impact on the other components of the GTX/GTH transceiver.

After Power-up and Configuration

The entire GTX/GTH TX requires a reset after configuration. See GTX/GTH Transceiver TX Reset in Response to Completion of Configuration, page 66.

After Turning on a Reference Clock to the CPLL/QPLL Being Used

If the reference clock(s) changes or the GTX/GTH transceiver(s) are powered up after configuration, GTTXRESET should be toggled after the PLL fully completes its reset procedure.

After Changing the Reference Clock to the CPLL/QPLL being used

Whenever the reference clock input to the PLL is changed, the PLL must be reset afterwards to ensure that it locks to the new frequency. The GTTXRESET should be toggled after the PLL fully completes its reset procedure.

After Assertion/Deassertion of C/QPLLPD, for the PLL being used

When the CPLL or QPLL being used goes back to normal operation after power down, the PLL must be reset. The GTTXRESET should be toggled after the PLL fully completes its reset procedure.

After Assertion/Deassertion of TXPD[1:0]

After the TXPD signal is deasserted, GTTXRESET must be toggled.

TX Rate Change

When a rate change is performed, the required reset sequence is performed automatically. When TXRATEDONE is asserted, it indicates that both a rate change and the necessary reset sequence have been applied and completed.

If the TX buffer is enabled, the TXBUF_RESET_ON_RATE_CHANGE attribute should be set to TRUE to allow the TX buffer to reset automatically after a rate change. If TX buffer bypass mode is used, alignment must be repeated after TXRATEDONE is asserted.

TX Parallel Clock Source Reset

The clocks driving TXUSRCLK and TXUSRCLK2 must be stable for correct operation. These clocks are often driven from an MMCM in the FPGA to meet phase and frequency requirements. If the MMCM loses lock and begins producing incorrect output, TXPCSRESET should be toggled after the clock source re-locks.

If TX buffer bypass mode is used, alignment must be repeated after the completion of the reset procedure.

RX Initialization and Reset

The GTX/GTH transceiver RX uses a reset state machine to control the reset process. Due to its complexity, the GTX/GTH transceiver RX is partitioned into more reset regions than the GTX/GTH transceiver TX. The partition allows RX initialization and reset to be operated in either sequential mode or single mode as shown in Figure 2-20:

1. RX in Sequential Mode

   To initialize the GTX/GTH transceiver RX, GTRXRESET must be used in sequential mode. Activating the GTRXRESET input can automatically trigger a full asynchronous RX reset. The reset state machine executes the reset sequence as shown in Figure 2-20, covering the entire RX PMA and RX PCS. During normal operation, sequential mode also allows the user to initiate a reset by activating any of these resets including RXPMARESET, RXDFELPMRESET, EYESCANRESET, RXPCSRESET, and RXBUFRESET, and continue the reset state machine until RXRESETDONE transitions from Low to High.

2. RX in Single Mode

   When the GTX/GTH transceiver RX is in single mode, RXPMARESET, RXDFELPMRESET, EYESCANRESET, RXPCSRESET, and RXBUFRESET in the reset sequence can be executed individually and independently without triggering a reset on other reset regions.

In either sequential mode or single mode, the RX reset state machine does not reset the PCS until RXUSERRDY goes High. The user should drive RXUSERRDY High after these conditions are met:

1. All clocks used by the application, including RXUSRCLK and RXUSRCLK2, are shown to be stable or locked when the PLL or the MMCM is used.
2. The user interface is ready to receive data from the GTX/GTH transceiver.

Figure 2-20: **GTX/GTH Transceiver RX Reset State Machine Sequence**

Ports and Attributes

Table 2-28 lists the ports required by the GTX/GTH transceiver RX initialization process.

Table 2-28: **RX Initialization and Reset Ports**

Port	Dir	Clock Domain	Description
GTRXRESET	In	Async	This port is driven High and then deasserted to start the full Channel RX reset sequence.
RXOSCALRESET	In	Async	GTH transceiver: Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
RSOSINTDONE	Out	Async	GTH transceiver: Reserved.
RXPMARESET	In	Async	This port is driven High and then deasserted to start RX PMA reset process. In single mode, activating RXPMARESET resets only the RX PMA blocks not including CDR and DFE. In sequential mode, activating RXPMARESET starts the RX reset process as shown in Figure 2-20 from RXPMARESET and followed by RXCDRPHASERESET, RXCDRFREQRESET, RXDFELPMRESET, EYESCANRESET, RXPCSRESET, and RXBUFRESET. Detailed coverage on sequential mode is listed in Table 2-30.
RXCDRRESET	In	Async	Reserved. Tied Low.
RXCDRFREQRESET	In	Async	Reserved. Tied Low.
RXDFELPMRESET	In	Async	This port is driven High and then deasserted to start the DFE reset process. In single mode, activating RXDFELPMRESET resets only the RX DFE circuits. In sequential mode, activating RXDFELPMRESET starts the RX reset process as shown in Figure 2-20 from RXDFELPMRESET and followed by EYESCANRESET, RXPCSRESET, and RXBUFRESET. Detailed coverage in sequential mode is listed in Table 2-30.

Table 2-28: **RX Initialization and Reset Ports *(Cont'd)***

Port	Dir	Clock Domain	Description
EYESCANRESET	In	Async	This port is driven High and then deasserted to start the EYESCAN reset process. In single mode, activating EYESCANRESET resets only the RX Eye Scan circuits. In sequential mode, activating EYESCANRESET starts the RX reset process as shown in Figure 2-20 from EYESCANRESET and followed by RXPCSRESET, and RXBUFRESET. Detailed coverage in sequential mode is listed in Table 2-30.
RXPCSRESET	In	Async	This port is driven High and then deasserted to start the PCS reset process. In single mode, activating RXPCSRESET resets only the RX PCS circuits. In sequential mode, activating RXPCSRESET starts the RX reset process as shown in Figure 2-20 from RXPCSRESET and followed by RXBUFRESET. Detailed coverage in sequential mode is listed in Table 2-30. In both modes, RXPCSRESET does not start the reset process until RXUSERRDY is High.
RXBUFRESET	In	Async	This port is driven High and then deasserted to start the RX elastic buffer reset process. In either single mode or sequential mode, activating RXBUFRESET resets the RX elastic buffer only.
RXUSERRDY	In	Async	This port is driven High from the user's application when RXUSRCLK and RXUSRCLK2 are stable. For example, if an MMCM is used to generate both RXUSRCLK and RXUSRCLK2, then the MMCM lock signal can be used here.
RXRESETDONE	Out	RXUSRCLK2	When asserted, this active-High signal indicates the GTX/GTH transceiver RX has finished reset and is ready for use. In sequential mode, this port is driven Low when GTRXRESET is driven High. This signal is not driven High until RXUSERRDY goes High. In single mode, this port is driven Low when any of the RX resets are asserted. This signal is not asserted until all RX resets are deasserted and RXUSERRDY is asserted.

Table 2-28: **RX Initialization and Reset Ports *(Cont'd)***

Port	Dir	Clock Domain	Description
RXPMARESETDONE	Out	Async	GTH transceiver: This active-High signal indicates GTH RX PMA reset is complete. This port is driven Low when GTRXRESET or RXPMARESET is asserted.
RXOOBRESET	In	Async	This port can be used to reset the OOB individually. It should be tied Low if the OOB function is not used or the OOB single reset is not required. RXOOBRESET is independent from the GTX/GTH transceiver RX reset state machine sequence as shown in Figure 2-20. Sequential mode and single mode do not apply to RXOOBRESET. Activating RXOOBRESET does not cause RXRESETDONE to transition from Low to High or High to Low.

Table 2-29 lists the attributes required by GTX/GTH transceiver RX initialization. In general cases, the reset time required by each reset on the RX datapath varies depending on line rate and function. The factors affecting each reset time are user-configurable attributes listed in Table 2-29.

Table 2-29: **RX Initialization and Reset Attributes**

Attribute	Type	Description
RXOSCALRESET_TIME	5-bit Binary	GTH transceiver: Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used. Must be a non-zero value when GTRXRESET is used to initiate the reset process.
RXOSCALRESET_TIMEOUT	5-bit Binary	GTH transceiver: Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used. Should be set to zero for normal operation.
RXPMARESET_TIME	5-bit Binary	Reserved. Represents the time duration to apply the RX PMA reset. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used. Must be a non-zero value when using GTRXRESET or RXPMARESET to initiate reset process.
RXCDRPHRESET_TIME	5-bit Binary	Reserved. Represents the time duration to apply RX CDR Phase reset. Must be a non-zero value when using RXCDRRESET to initialize the reset process.

Table 2-29: **RX Initialization and Reset Attributes *(Cont'd)***

Attribute	Type	Description
RXCDRFREQRESET_TIME	5-bit Binary	Reserved. Represents the time duration to apply the RX CDRFREQ reset. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used. Must be a non-zero value when using RXCDRFREQRESET to initiate the reset process.
RXDFELPMRESET_TIME	7-bit Binary	Reserved. Represents the time duration to apply the RX DFE reset. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used. Must be a non-zero value when using RXDFELPMRESET to initiate the reset process.
RXISCANRESET_TIME	5-bit Binary	Reserved. Represents the time duration to apply the RX EYESCAN reset. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used. Must be a non-zero value when using RXISCANRESET_TIME to initiate the reset process.
RXPCSRESET_TIME	5-bit Binary	Reserved. Represents the time duration to apply the RX PCS reset. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used. Must be a non-zero value when using RXPCSRESET to initiate the reset process.
RXBUFRESET_TIME	5-bit Binary	Reserved. Represents the time duration to apply the RX BUFFER reset. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used. Must be a non-zero value when using RXBUFRESET to initiate the reset process.

GTX/GTH Transceiver RX Reset in Response to Completion of Configuration

The RX reset sequence shown in Figure 2-20 is not automatically started to follow the global GSR. For the following transceivers and configurations:

- All GTX transceivers
- GTH transceivers configured with RXOUT_DIV = 1 and/or RX_DATA_WIDTH = 16, 32, or 64

These conditions must be met:

1. GTRESETSEL must be driven Low to use the sequential mode.
2. GTRXRESET must be used.
3. All single reset inputs including RXPMARESET, RXCDRRESET, RXCDRFREQRESET, RXDFELPMRESET, EYESCANRESET, RXPCSRESET, and RXBUFRESET must be constantly held Low during the entire reset process before RXRESETDONE goes High.
4. GTRXRESET cannot be driven Low until the associated PLL is locked.

If the reset mode is defaulted to sequential mode upon configuration, then C/QPLLRESET and GTRXRESET can be asserted after waiting for a minimum of 500 ns after configuration is complete.

If the reset mode is defaulted to single mode, then the user must:

1. Wait a minimum of 500 ns after configuration is complete.
2. Change reset mode to Sequential mode.
3. Wait another 300-500 ns.
4. Assert C/QPLLRESET and GTRXRESET.

It is recommended to use the associated PLLLOCK from either the CPLL or QPLL to release GTRXRESET from High to Low as shown in Figure 2-21. The RX reset state machine waits when GTRXRESET is High and starts the reset sequence until GTRXRESET is released Low.

Figure 2-21: **GTX/GTH Receiver after FPGA Configuration**

For a GTH transceiver configured as:

- RXOUT_DIV != 1 and
- RX internal data width is 20- or 40-bit (RX_DATA_WIDTH = 20, 40, or 80)

These conditions must be met:

1. GTRESETSEL must be driven Low to use the sequential mode.
2. GTRXRESET must be used.
3. All single reset inputs including RXPMARESET, RXCDRRESET, RXCDRFREQRESET, RXLPMRESET, EYESCANRESET, RXPCSRESET, and RXBUFRESET must constantly be held Low during the entire reset process before RXRESETDONE goes High.
4. GTRXRESET cannot be driven Low until the associated PLL is locked.

If the reset mode is defaulted to sequential mode upon configuration, C/QPLLRESET and GTRXRESET can be asserted after waiting for a minimum of 500 ns after configuration is complete.

If the reset mode is defaulted to single mode, the user must:

1. Wait a minimum of 500 ns after configuration is complete.
2. Change reset mode to Sequential mode.
3. Wait another 300–500 ns.

To issue a GTRXRESET upon configuration, the steps in Figure 2-22 should be performed.

Figure 2-22: **GTH Receiver after FPGA Configuration**

Notes relevant to Figure 2-22:

1. DRP wr denotes the function of performing a DRP write to address `9'h011`. The exact DRP transaction is not shown.
2. The sequence of events in Figure 2-22 is not drawn to scale.
3. To trigger RX reset upon configuration, C/QPLLRESET are asserted and released while GTRXRESET is kept asserted. The assertion of GTRXRESET causes RXPMARESETDONE to go Low.

4. A DRP write is issued to the GTHE2_CHANNEL primitive, DRPADDR 9'h011, setting bit[11] to 1'b0.

   To ensure only bit[11] of DRPADDR 9'h011 is modified, it is best to perform a read-modify-write function.

5. Upon DRP write completion, the user can set and hold GTRXRESET Low as desired. The user can extend the assertion of GTRXRESET as long as GTRXRESET is held High until the DRP write is completed.

   Note: It is recommended to use the associated PLLLOCK from either the CPLL or QPLL to release GTRXRESET from High to Low as shown in Figure 2-19.

6. The user should wait for falling edge of RXPMARESETDONE.
7. A DRP write is issued to the GTHE2_CHANNEL primitive, DRPADDR 9'h011, restoring the original setting for bit[11]. The completion of this DRP write must occur before RXPMARESETDONE switches from Low to High. RXPMARESETDONE stays Low for a minimum of 0.66 µs.
8. GTRXRESET should be driven with an output of a register to avoid glitches.
9. RXPMARESET_TIME should be set to `5'h3`. This should be the default setting.
10. The sequence above simulates correctly if SIM_GTRESET_SPEEDUP is set to FALSE and the GT functional simulation model in the UniSims library is used. If SIM_GTRESET_SPEEDUP is set to TRUE or if the GT functional simulation model in the unifast library is used, the above sequence should be bypassed.

    Note: For GTH transceivers configured as RXOUT_DIV = 1 or RX_DATA_WIDTH = 16, 32, or 64, the steps above are allowed but not required. Refer to Figure 2-23 for details on requirements.

GTX/GTH Transceiver RX Reset in Response to GTRXRESET Pulse

The GTX/GTH transceiver allows the user to completely reset the entire GTX/GTH transceiver RX at any time when needed by sending GTRXRESET an active High pulse. All RX reset attributes listed in Table 2-28 can be set statically or reprogrammed through DRP ports to adjust the required reset time before applying GTRXRESET. These conditions must be met to use GTRXRESET:

1. GTRESETSEL must be driven Low to use sequential mode.
2. All reset inputs shown on the left of Figure 2-20 including RXPMARESET, RXCDRRESET, RXCDRFREQRESET, RXDFELPMRESET, EYESCANRESET, RXPCSRESET, and RXBUFRESET must be constantly driven Low during the entire reset process before RXRESETDONE is detected High.
3. The associated PLL must indicate locked.
4. For the following transceivers and configurations:
   - All GTX transceivers
   - GTH transceivers configured with RXOUT_DIV = 1 and/or RX_DATA_WIDTH = 16, 32, or 64

   The guideline for this asynchronous GTRXRESET pulse width is one period of the reference clock.

Figure 2-23: **GTX/GTH Receiver Reset after GTRXRESET Pulse**

For GTH transceiver configured as:

- RXOUT_DIV != 1 and
- RX internal data width is 20 or 40-bit (RX_DATA_WIDTH = 20, 40, or 80)

These conditions must be met to use GTRXRESET:

1. GTRESETSEL must be driven Low to use sequential mode.
2. All reset inputs shown on the left of Figure 2-18, page 68 including RXPMARESET, RXCDRRESET, RXCDRFREQRESET, RXDFELPMRESET, EYESCANRESET, RXPCSRESET, and RXBUFRESET must be constantly driven Low during the entire reset process before RXRESETDONE is detected High.
3. The associated PLL must indicate locked.
4. The steps for issuing GTRXRESET are illustrated in Figure 2-24.

Figure 2-24: **GTH Receiver Reset for GTRXRESET**

Notes relevant to Figure 2-24:

1. DRP wr denotes the function of performing a DRP write to address `9'h011`. The exact DRP transaction is not shown.
2. The sequence of events in Figure 2-24 is not drawn to scale.

3. To trigger GTRXRESET, GTRXRESET should be set and held High. This causes RXPMARESETDONE to go Low.
4. A DRP write should be issued to the GTHE2_CHANNEL primitive, DRPADDR `9'h011`, bit[11] should be set to `1'b0`.

   To ensure only bit[11] of DRPADDR `9'h011` is modified, it is best to perform a read-modify-write function.
5. Upon DRP write completion, GTRXRESET can be set and held Low as desired. The user can extend the assertion of GTRXRESET as long as GTRXRESET is held High until the DRP write is completed.
6. The user should wait for the falling edge of RXPMARESETDONE.
7. A DRP write should be issued to the GTHE2_CHANNEL primitive, DRPADDR `9'h011`, restoring the original setting for bit[11]. The completion of this DRP write must occur before RXPMARESETDONE switches from Low to High. RXPMARESETDONE stays Low for a minimum of 0.66 µs.
8. GTRXRESET should be driven with an output of a register to avoid glitches.
9. RXPMARESET_TIME should be set to `5'h3`. This should be the default setting.
10. The sequence above simulates correctly if SIM_GTRESET_SPEEDUP is set to FALSE and the GT functional simulation model in the UniSims library is used. If SIM_GTRESET_SPEEDUP is set to TRUE or if the GT functional simulation model in the unifast library is used, the above sequence should be bypassed.

    Note: For GTH transceivers configured as RXOUT_DIV = 1 or RX_DATA_WIDTH = 16, 32, or 64, the steps above are allowed but not required. Refer to Figure 2-23 for details on requirements.

GTH Transceiver RX PMA Reset

For GTH transceivers configured as:

- RXOUT_DIV != 1 and
- RX internal data width is 20- or 40-bit (RX_DATA_WIDTH = 20, 40, or 80)

When users want to issue an RXPMARESET, the steps in Figure 2-25 should be performed.

Figure 2-25: **GTH Transceiver RXPMARESET Sequence**

Notes relevant to Figure 2-25:

1. DRP wr denotes the function of performing a DRP write to address `9'h011`. The exact DRP transaction is not shown.
2. The sequence of events in Figure 2-25 is not drawn to scale.

3. To trigger an RXPMARESET, a DRP write should be issued to the GTHE2_CHANNEL primitive, DRPADDR `9'h011`, and bit[11] should be set to `1'b0`.

   To ensure only bit[11] of DRPADDR `9'h011` is modified, it is best to perform a read-modify-write function.
4. Upon DRP write completion, RXPMARESET should be set and held High.
5. The user should wait for RXPMARESETDONE to be detected Low.
6. A DRP write should be issued to the GTHE2_CHANNEL primitive, DRPADDR `9'h011`, restoring the original setting for bit[11].
7. Upon DRP write completion, RXPMARESET can be set and held Low as desired. The user can extend the assertion of RXPMARESET, as long as RXPMARESET is held High until the DRP write is completed.
8. RXPMARESET should be driven with an output of a register to avoid glitches.

   Note: For GTH transceivers configured as RXOUT_DIV = 1 or RX_DATA_WIDTH = 16, 32, or 64, the steps above are allowed but not required.

GTX/GTH Transceiver RX Component Resets

GTX/GTH transceiver RX component resets can operate in either sequential mode or single mode. They are primarily used for special cases. These resets are needed when only a specific subsection needs to be reset. Table 2-30 and Table 2-31 also summarize all resets available to the GTX/GTH transceiver RX and components affected by them in both sequential mode and single mode. These resets are all asynchronous.

Table 2-30: **RX Component Reset Coverage in Sequential Mode**

	Functional Blocks	GTRX RESET	RXPMA RESET	RXDFE RESET	EYESCAN RESET	RXPCS RESET	RXBUF RESET
RX PCS	FPGA RX Fabric Interface	✓	✓	✓	✓	✓	
	RX Gearbox	✓	✓	✓	✓	✓	
	RX Status Control	✓	✓	✓	✓	✓	
	RX Elastic Buffer Delay Aligner	✓	✓	✓	✓	✓	
	RX 8B/10B Encoder	✓	✓	✓	✓	✓	
	RX Comma Detect and Alignment	✓	✓	✓	✓	✓	
	RX Polarity	✓	✓	✓	✓	✓	
	PRBS Checker	✓	✓	✓	✓	✓	
	RX Elastic Buffer	✓	✓	✓	✓	✓	✓
	RX Reset FSM	✓					

Table 2-30: **RX Component Reset Coverage in Sequential Mode *(Cont'd)***

	Functional Blocks	GTRX RESET	RXPMA RESET	RXDFE RESET	EYESCAN RESET	RXPCS RESET	RXBUF RESET
RX PMA	RX Analog Front End	✓	✓				
	RX Out-of-Band Signaling	✓	✓				
	RX SIPO	✓	✓				
	RX CDR Phase Path	✓	✓				
	RX CDR Frequency Path	✓	✓				
	RX DFE	✓	✓	✓			
	RX EYESCAN	✓	✓	✓	✓		

Table 2-31: **RX Component Reset Coverage in Single Mode**

	Functional Blocks	GTRX RESET	RXPMA RESET	RXDFE RESET	EYESCAN RESET	RXPCS RESET	RXBUF RESET	RXOOB RESET
RX PCS	FPGA RX Fabric Interface					✓		
	RX Gearbox					✓		
	RX Status Control					✓		
	RX Delay Aligner					✓		
	RX 8B/10B Encoder					✓		
	RX Comma Detect and Alignment					✓		
	RX Polarity					✓		
	PRBS Checker					✓		
	RX Elastic Buffer						✓	
	RX Reset FSM							
RX PMA	RX Analog Front End		✓					
	RX Out-of-Band Signaling		✓					✓
	RX SIPO		✓					
	RX CDR Phase Path							
	RX CDR Frequency Path							
	RX DFE			✓				
	RX EYESCAN				✓			

Table 2-32 lists the recommended resets for various situations.

Table 2-32: **Recommended Resets for Common Situations**

Situation	Components to be Reset	Recommended Reset[(1)]
After power up and configuration	Entire RX	GTRXRESET
After turning on a reference clock to the CPLL/QPLL being used	Entire RX	GTRXRESET
After changing the reference clock to the CPLL/QPLL being used	Entire RX	GTRXRESET
After assertion/deassertion of CPLLPD or QPLLPD for the PLL being used	Entire RX	GTRXRESET
After assertion/deassertion of RXPD[1:0]	Entire RX	GTRXRESET
RX rate change	RX PCS	Reset performed automatically
RX parallel clock source reset	RX PCS	RXPCSRESET
After remote power up	Entire RX	GTRXRESET
Electrical idle	Entire RX	Handled automatically with appropriate attribute settings
After connecting RXN/RXP[(2)]	Entire RX	GTRXRESET
After recovered clock becomes stable	RX Elastic Buffer	RXBUFRESET
After an RXBUFFER error	RX Elastic Buffer	RXBUFRESET
After changing channel bonding mode in real time	RX Elastic Buffer	RX elastic buffer is reset automatically after change in channel bonding mode by setting RXBUF_RESET_ON_CB_CHANGE to TRUE
After PRBS error	PRBS Error Counter	PRBSCNTRESET
After comma realignment	RX Elastic Buffer (optional)	RX elastic buffer is reset automatically after comma realignment by setting RXBUF_RESET_ON_COMMA ALIGN to TRUE

Notes:

1. The recommended reset has the smallest impact on the other components of the GTX transceiver.
2. It is assumed that RXN/RXP are connected simultaneously.

After Power-up and Configuration

The entire GTX/GTH TX requires a reset after configuration. See GTX/GTH Transceiver RX Reset in Response to Completion of Configuration, page 77.

After Turning on a Reference Clock to the CPLL/QPLL Being Used

If the reference clock(s) changes or GTX/GTH transceiver(s) are powered up after configuration, GTRXRESET should be toggled after the PLL fully completes its reset procedure.

After Changing the Reference Clock to the CPLL/QPLL Being Used

Whenever the reference clock input to the PLL is changed, the PLL must be reset afterwards to ensure that it locks to the new frequency. The GTRXRESET should be toggled after the PLL fully completes its reset procedure.

After Assertion/Deassertion of CPLLD or QPLLPD for the PLL Being Used

When the CPLL or QPLL being used goes back to normal operation after power down, the PLL must be reset. The GTRXRESET should be toggled after the PLL fully completes its reset procedure.

After Assertion/Deassertion of RXPD[1:0]

After the RXPD signal is deasserted, GTRXRESET must be toggled.

RX Rate Change

When a rate change is performed, the required reset sequence is performed automatically. When RXRATEDONE is asserted, it indicates that both a rate change and the necessary reset sequence have been applied and completed.

If the RX buffer is enabled, the RXBUF_RESET_ON_RATE_CHANGE attribute should be set to TRUE to allow the RX buffer to reset automatically after a rate change. If RX buffer bypass mode is used, alignment must be repeated after RXRATEDONE is asserted.

RX Parallel Clock Source Reset

The clocks driving RXUSRCLK and RXUSRCLK2 must be stable for correct operation. These clocks are often driven from an MMCM in the FPGA to meet phase and frequency requirements. If the MMCM loses lock and begins producing incorrect output, RXPCSRESET should be toggled after the clock source re-locks. If RX buffer bypass mode is used, alignment must be repeated after the completion of the reset procedure.

After Remote Power-Up

If the source of incoming data is powered up after the GTX/GTH transceiver that is receiving its data has begun operating, the RX side must be reset to ensure a clean lock to the incoming data.

Electrical Idle Reset

For protocols that support OOB and electrical idle, when the differential voltage of the RX input to the transceiver drops to OOB or electrical idle levels, the RX CDR is managed automatically when the attributes associated with electrical idle are set to appropriate values. Recommended values from the 7 Series FPGAs Transceivers Wizard should be used.

After Connecting RXN/RXP

When the RX data to the GTX/GTH transceiver comes from a connector that can be plugged in and unplugged, the RX side must be reset when the data source is plugged in to ensure that it can lock to incoming data.

After Recovered Clock Becomes Stable

Depending on the design of the clocking scheme, it is possible for the RX reset sequence to be completed before the CDR is locked to the incoming data. In this case, the recovered clock might not be stable when RXRESETDONE is asserted.

When the RX buffer is used, RXBUFRESET should be triggered after the recovered clock becomes stable. When RX buffer bypass is used, the alignment procedure should not start until the recovered clock becomes stable.

Refer to DS182, *Kintex-7 FPGAs Data Sheet: DC and Switching Characteristics* and DS183, *Virtex-7 FPGAs Data Sheet: DC and Switching Characteristics* for successful CDR lock-to-data criteria.

After an RX Elastic Buffer Error

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

After Changing Channel Bonding Mode During Run Time

When set to TRUE, RXBUF_RESET_ON_CB_CHANGE enables automatic reset of the RX elastic buffer when RXCHANBONDMASTER, RXCHANBONDSLAVE, or RXCHANBONDLEVEL change.

After a PRBS Error

PRBSCNTRESET is asserted to reset the PRBS error counter.

After Comma Realignment

When set to TRUE, RXBUF_RESET_ON_COMMAALIGN enables automatic reset of the RX elastic buffer during comma realignment.

Power Down

Functional Description

The GTX/GTH transceiver supports a range of power-down modes. These modes support both generic power management capabilities as well as those defined in the PCI Express® and SATA standards.

The GTX/GTH transceiver offers different levels of power control. Each channel in each direction can be powered down separately using TXPD and RXPD. The CPLLPD port directly affects the Channel PLL while the QPLLPD port directly affects the Quad PLL.

Ports and Attributes

Table 2-33 defines the power-down ports.

Table 2-33: **Power-Down Ports**

Port	Dir	Clock Domain	Description
CPLLPD	In	Async	This active-High signal powers down the Channel PLL.
QPLLPD	In	Async	This active-High signal powers down the Quad PLL.
RXPD[1:0]	In	Async	Powers down the RX lane according to the PCI Express PIPE protocol encoding. `00`: P0 (normal operation) `01`: P0s (low recovery time power down) `10`: P1 (longer recovery time) `11`: P2 (lowest power state)
TXPD[1:0]	In	TXUSRCLK2 (TXPDELECIDLEMODE makes this port asynchronous)	Powers down the TX lane according to the PCI Express PIPE protocol encoding. `00`: P0 (normal operation) `01`: P0s (low recovery time power down) `10`: P1 (longer recovery time; Receiver Detection still on) `11`: P2 (lowest power state) Attributes can control the transition times between these power-down states.
TXPDELECIDLEMODE	In	Async	Determines if TXELECIDLE and TXPD should be treated as synchronous or asynchronous signals.
TXPHDLYPD	In	Async	TX phase and delay alignment circuit power down. It is set to `1'b0` in TX buffer bypass mode. 0: Power up the TX phase and delay alignment circuit. 1: Power down the TX phase and delay alignment circuit.

Table 2-33: **Power-Down Ports *(Cont'd)***

Port	Dir	Clock Domain	Description
RXPHDLYPD	In	Async	RX phase and delay alignment circuit power down. It is set to `1'b0` in RX buffer bypass mode. 0: Power up the RX phase and delay alignment circuit. 1: Power down the RX phase and delay alignment circuit.

Table 2-34 defines the power-down attributes.

Table 2-34: **Power-Down Attributes**

Attribute	Type	Description
PD_TRANS_TIME_FROM_P2	12-bit Hex	Counter settings for programmable transition time from P2 state for PCIe. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
PD_TRANS_TIME_NONE_P2	8-bit Hex	Counter settings for programmable transition time to/from all states except P2 for PCIe. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
PD_TRANS_TIME_TO_P2	8-bit Hex	Counter settings for programmable transition time to P2 state for PCIe. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
TRANS_TIME_RATE	8-bit Hex	Counter settings for programmable transition time when the rate is changed using the [TX/RX]RATE pins for all protocols including the PCIe protocol (Gen2/Gen1 data rates). The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
RX_CLKMUX_PD	1-bit Binary	The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
TX_CLKMUX_PD	1-bit Binary	The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.

Generic Power-Down Capabilities

The GTX/GTH transceiver provides several power-down features that can be used in a wide variety of applications. Table 2-35 summarizes these capabilities.

Table 2-35: **Basic Power-Down Functions Summary**

Function	Controlled By	Affects
Quad PLL Control	QPLLPD	Powers down the Quad PLL.
Channel PLL Control	CPLLPD	Powers down the Channel PLL.
TX Power Control	TXPD[1:0]	The TX of the GTX/GTH transceiver.
RX Power Control	RXPD[1:0]	The RX of the GTX/GTH transceiver.

PLL Power Down

To activate the Quad PLL power-down mode, the active-High QPLLPD signal is asserted. Similarly, to activate the Channel PLL power-down mode, the active-High CPLLPD signal is asserted. When either QPLLPD or CPLLPD is asserted, the corresponding PLL is powered down. As a result, all clocks derived from the respective PLL are stopped.

Recovery from this power state is indicated by the assertion of the corresponding PLL lock signal that is either the QPLLLOCK signal of the Quad PLL or the CPLLLOCK signal of the GTX/GTH transceiver of the Quad PLL or the CPLLLOCK signal of *the respective channel*.

TX and RX Power Down

When the TX and RX power control signals are used in non PCI Express implementations, TXPD and RXPD can be used independently. Also, when these interfaces are used in non PCI Express applications, only two power states are supported, as shown in Table 2-36. When using this power-down mechanism, these must be true:

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

Table 2-36: **TX and RX Power States for Operation that are not for PCI Express Designs**

TXPD[1:0] or RXPD[1:0]	Description
00	Normal mode. Transceiver TX or RX is active sending or receiving data.
11	Power-down mode. Transceiver TX or RX is idle.

Power-Down Features for PCI Express Operation

Refer to PCI Express Power Management, page 321 for more details.

Loopback

Functional Description

Loopback modes are specialized configurations of the transceiver datapath where the traffic stream is folded back to the source. Typically, a specific traffic pattern is transmitted and then compared to check for errors. Figure 2-26 illustrates a loopback test configuration with four different loopback modes.

Figure 2-26: **Loopback Testing Overview**

Loopback test modes fall into two broad categories:

- Near-end loopback modes loop transmit data back in the transceiver closest to the traffic generator.
- Far-end loopback modes loop received data back in the transceiver at the far end of the link.

Loopback testing can be used either during development or in deployed equipment for fault isolation. The traffic patterns used can be either application traffic patterns or specialized pseudo-random bit sequences. Each GTX/GTH transceiver has a built-in PRBS generator and checker.

Each GTX/GTH transceiver features several loopback modes to facilitate testing:

- Near-End PCS Loopback (path 1 in Figure 2-26)

  The RX elastic buffer must be enabled and RX_XCLK_SEL must be set to RXREC for Near-End PCS loopback to function properly. While in Near-End PCS loopback, the RX XCLK domain is clocked by the TX PMA parallel clock (TX XCLK). If the RXOUTCLK is used to clock FPGA logic and RXOUTCLKSEL is set to RXOUTCLKPMA during normal operation, one of these two items must be changed when placing the GTX/GTH transceiver into near-end PCS loopback:

  - Set RXOUTCLKSEL to select RXOUTCLKPCS, or
  - Set RXCDRHOLD = `1'b1`
- Near-End PMA Loopback (path 2 in Figure 2-26)
- Far-End PMA Loopback (path 3 in Figure 2-26)

The TX buffer must be enabled and TX_XCLK_SEL must be set to TXOUT for Far-End PMA loopback to function properly. While in Far-End PMA loopback, the write side of the TX buffer is clocked by the RX PMA parallel clock (RX XCLK).

- Far-End PCS Loopback (path 4 in Figure 2-26)

  If clock correction is not used, a transceiver in Far-end PCS loopback must use the same reference clock used by the transceiver that is the source of the loopback data. Regardless of whether or not clock correction is used, the TXUSRCLK and RXUSRCLK ports must be driven by the same clocking resource (BUFG, BUFR, or BUFH).

Ports and Attributes

Table 2-37 defines the loopback ports.

Table 2-37: **Loopback Ports**

Port	Dir	Clock Domain	Description
LOOPBACK[2:0]	In	Async	`000`: Normal operation `001`: Near-End PCS Loopback `010`: Near-End PMA Loopback `011`: Reserved `100`: Far-End PMA Loopback `101`: Reserved `110`: Far-End PCS Loopback

There are no loopback attributes.

Dynamic Reconfiguration Port

Functional Description

The dynamic reconfiguration port (DRP) allows the dynamic change of parameters of the GTXE2_CHANNEL/GTHE2_CHANNEL and GTXE2_COMMON/ GTHE2_COMMON primitives. The DRP interface is a processor-friendly synchronous interface with an address bus (DRPADDR) and separated data buses for reading (DRPDO) and writing (DRPDI) configuration data to the primitives. An enable signal (DRPEN), a read/write signal (DRPWE), and a ready/valid signal (DRPRDY) are the control signals that implement read and write operations, indicate operation completion, or indicate the availability of data.

Ports and Attributes

Table 2-38 shows the DRP related ports for GTXE2_CHANNEL/GTHE2_CHANNEL.

Table 2-38: **DRP Ports of GTXE2_CHANNEL/GTHE2_CHANNEL**

Port	Dir	Clock Domain	Description
DRPADDR[8:0]	In	DRPCLK	DRP address bus.
DRPCLK	In	N/A	DRP interface clock.

Table 2-38: **DRP Ports of GTXE2_CHANNEL/GTHE2_CHANNEL *(Cont'd)***

Port	Dir	Clock Domain	Description
DRPEN	In	DRPCLK	DRP enable signal. 0: No read or write operation performed. 1: Enables a read or write operation. For write operations, DRPWE and DRPEN should be driven High for one DRPCLK cycle only. See Figure 2-27 for correct operation.
DRPDI[15:0]	In	DRPCLK	Data bus for writing configuration data from the FPGA logic resources to the transceiver.
DRPRDY	Out	DRPCLK	Indicates operation is complete for write operations and data is valid for read operations.
DRPDO[15:0]	Out	DRPCLK	Data bus for reading configuration data from the GTX/GTH transceiver to the FPGA logic resources.
DRPWE	In	DRPCLK	DRP write enable. 0: Read operation when DRPEN is 1. 1: Write operation when DRPEN is 1. For write operations, DRPWE and DRPEN should be driven High for one DRPCLK cycle only. See Figure 2-27 for correct operation.

Table 2-39 shows the DRP related ports for GTXE2_COMMON/GTHE2_COMMON.

Table 2-39: **DRP Ports of GTXE2_COMMON/GTHE2_COMMON**

Port	Dir	Clock Domain	Description
DRPADDR[7:0]	In	DRPCLK	DRP address bus.
DRPCLK	In	N/A	DRP interface clock.
DRPEN	In	DRPCLK	DRP enable signal. 0: No read or write operation performed. 1: Enables a read or write operation. For write operations, DRPWE and DRPEN should be driven High for one DRPCLK cycle only. See Figure 2-27 for correct operation.
DRPDI[15:0]	In	DRPCLK	Data bus for writing configuration data from the FPGA logic resources to the transceiver.
DRPRDY	Out	DRPCLK	Indicates operation is complete for write operations and data is valid for read operations.
DRPDO[15:0]	Out	DRPCLK	Data bus for reading configuration data from the GTX/GTH transceiver to the FPGA logic resources.

Table 2-39: **DRP Ports of GTXE2_COMMON/GTHE2_COMMON *(Cont'd)***

Port	Dir	Clock Domain	Description
DRPWE	In	DRPCLK	DRP write enable. 0: Read operation when DRPEN is 1. 1: Write operation when DRPEN is 1. For write operations, DRPWE and DRPEN should be driven High for one DRPCLK cycle only. See Figure 2-27 for correct operation.

Usage Model

Write Operation

Figure 2-27 shows the DRP write operation timing. New DRP operation can be initiated when DRPRDY is asserted.

Figure 2-27: **DRP Write Timing**

Read Operation

Figure 2-28 shows the DRP read operation timing. New DRP operation can be initiated when DRPRDY is asserted.

Figure 2-28: **DRP Read Timing**

Digital Monitor

Functional Description

The digital monitor can be used to monitor the state of the receiver LPM and DFE mode adaptation loops, such as the adapted continuous time linear equalizer (CTLE) gains and DFE taps.

Ports and Attributes

Table 2-40 shows the digital monitor ports.

Table 2-40: **Digital Monitor Ports**

Port	Dir	Clock Domain	Description
GTX transceiver: DMONITOROUT[7:0] GTH transceiver: DMONITOROUT[14:0]	Out	Async/Local Clock	Digital Monitor Output Bus

Table 2-41 shows the digital monitor attributes.

Table 2-41: **Digital Monitor Attributes**

<table>
<tr><th>Attribute</th><th>Type</th><th>Description</th></tr>
<tr><td>GTX transceiver:
RX_DEBUG_CFG[11:0]</td><td>12-bit Binary</td><td>[11:6] - Reserved. Set to 6'h00
[5:2] - Select adaptation loop:
<table><tr><th>[5:2]</th><th>Description</th></tr><tr><td>4'b0000</td><td>Reserved</td></tr><tr><td>4'b0001</td><td>RXDFEOS[6:0]</td></tr><tr><td>4'b0010</td><td>RXLPMHF[6:0]</td></tr><tr><td>4'b0011</td><td>RXLPMLF[6:0]</td></tr><tr><td>4'b0100</td><td>RXDFELF[3:0]</td></tr><tr><td>4'b0101</td><td>RXDFEVP[6:0]</td></tr><tr><td>4'b0110</td><td>RXDFEUT[6:0]</td></tr><tr><td>4'b0111</td><td>RXDFETAP2[5:0]</td></tr><tr><td>4'b1000</td><td>RXDFETAP3[5:0]</td></tr><tr><td>4'b1001</td><td>RXDFETAP4[4:0]</td></tr><tr><td>4'b1010</td><td>RXDFETAP5[4:0]</td></tr><tr><td>4'b1011</td><td>RXDFEAGC[4:0]</td></tr><tr><td>4'b11XX</td><td>Reserved</td></tr></table>
[1:0] - Reserved. Set to 2'b00</td></tr>
<tr><td>GTH transceiver:
RX_DEBUG_CFG[13:0]</td><td>14-bit Binary</td><td>[13:8] - Reserved. Set to 6'h00
[7:6] - Reserved. Set to 2'b11.
[5] - Reserved. Set to 1'b0.
[4:0] - Select adaptation loop:
<table><tr><th>[4:0]</th><th>Description</th></tr><tr><td>5'b00000</td><td>Reserved</td></tr><tr><td>5'b00001</td><td>Reserved</td></tr><tr><td>5'b00010</td><td>RXDFEOS[6:0]</td></tr><tr><td>5'b00011</td><td>RXLPMHF[6:0]</td></tr><tr><td>5'b00100</td><td>RXLPMLF[6:0]</td></tr><tr><td>5'b00101</td><td>RXDFELF[3:0]</td></tr><tr><td>5'b00110</td><td>RXDFEAGC[3:0]</td></tr><tr><td>5'b00111</td><td>RXDFEVP[6:0]</td></tr><tr><td>5'b01000</td><td>RXDFEUT[6:0]</td></tr><tr><td>5'b01001</td><td>RXDFETAP2[5:0]</td></tr><tr><td>5'b01010</td><td>RXDFETAP3[5:0]</td></tr><tr><td>5'b01011</td><td>RXDFETAP4[4:0]</td></tr><tr><td>5'b01100</td><td>RXDFETAP5[4:0]</td></tr><tr><td>5'b01101</td><td>RXDFETAP6[4:0]</td></tr><tr><td>5'b01110</td><td>RXDFETAP7[4:0]</td></tr></table></td></tr>
</table>

Table 2-41: **Digital Monitor Attributes *(Cont'd)***

Attribute	Type	Description
GTH Transceiver: CFOK_CFG[41]	1-bit Binary	Reserved. Set to `1'b1`.
GTX transceiver: PCS_RSVD_ATTR[6]	1-bit Binary	Reserved. Set to `1'b1`.
DMONITOR_CFG[23:0]	24-bit Binary	[23:16] - Reserved. Set to `8'h00`. [15] - Reserved. Set to `1'b1`. [14:1] - Reserved. Set to `14'h0000`. [0] - DMONITOR SELECT: • Set to `1'b1` to monitor Digital Monitor signals • Set to `1'b0` when not monitoring

Use Mode 1

Set the DMONITOR_CFG attribute via the DRP port to the appropriate loop for monitoring. The DRP location of DMONITOR_CFG is:

`0x086`[15:0] = DMONITOR_CFG[15:0]

`0x087`[7:0] = DMONITOR_CFG[23:16]

The output can be observed on DMONITOROUT. The signals from the digital monitor are LSB aligned and asynchronous.

Capturing the Digital Monitor Output

The DMONITOROUT signals change slowly in comparison to the USRCLK2s. One way to capture the DMONITOROUT is described here.

The USRCLK2 or a system clock that is comparable in frequency can be used to register the DMONITOROUT. A second set of flip-flops takes the output of the first set of flip-flops as its input.

If the output of the first and second set of flip-flops is the same, a good value was captured and can be thus be propagated to the downstream logic.

Example Verilog code is provided here as an illustration:

```
reg [7:0] compare1, compare2, dmonitorout_sync;
always@ (posedge TXUSRCLK2)
begin
  if (reset)
  begin
    compare1 <= 8'd0;
    compare2 <= 8'd0;
    dmonitorout_sync <= 8'd0;
  end
  else
  begin
    compare1 <= DMONITOROUT;
    compare2 <= compare1;

    if (compare1 == compare2)
      dmonitorout_sync <= compare2;
    else
```

```
        dmonitorout_sync<=dmonitorout_sync;

    end //else
  end //always
```

Any method that captures the information successfully is valid.

Interpreting the Digital Monitor Output

This section describes which bits of the DMONITOROUT bus are relevant for the appropriate selection of DMON_CFG and the manner of interpreting the output.

- RXDFEOS[6:0] = DMONITOROUT[6:0]

  `7'd0` = –Full scale

  `7'd63`, `7'd64` = 0

  `7'd127` = +Full scale

- RXLPMHF [6:0] = RXLPMLF [6:0] = DMONITOROUT[6:0]

  `7'd0` = 0

  `7'd127` = Full scale

- RXDFELF [3:0] = DMONITOR [6:3]

  `4'd0` = 0

  `4'd15` = Full Scale

- RXDFEVP [6:0] = DMONITOROUT[6:0]

  `7'd0` = 0

  `7'd127` = Full scale

- RXDFEUT [6:0] = DMONITOROUT[6:0]

  `7'd0` = 0

  `7'd127` = Full scale

- RXDFETAP2 [5:0] = DMONITOROUT[5:0]

  `6'd0` = 0

  `6'd63` = Full scale

- RXDFETAP3 [5:0] = DMONITOROUT[5:0]

  `6'd0` = –Full scale

  `6'd31`, `6'd32` = 0

  `6'd63` = +Full scale

- RXDFETAP4 [4:0] = RXDFETAP5 [4:0] = DMONITOROUT[4:0]

  `5'd0` = –Full scale

  `5'd15`, `5'd16` = 0

  `5'd31` = +Full scale

- GTX transceiver:

  RXDFEAGC [4:0] = DMONITOROUT[4:0]

  `5'd0` = 0

`5'd31` = Full scale

- GTH transceiver:
  RXDFEAGC [3:0] = DMONITOROUT [4:1]

  `4'd0` = 0

  `4'd15` = Full scale

- GTH transceiver:
  RXDFETAP6 [4:0] = RXDFETAP7 [4:0] = DMONITOROUT [4:0]

  `5'd0` = –Full scale

  `5'd15`, `5'd16` = 0

  `5'd31` = +Full scale

Chapter 3

Transmitter

TX Overview

Functional Description

This chapter shows how to configure and use each of the functional blocks inside the transmitter (TX). Each transceiver includes an independent transmitter, which consists of a PCS and a PMA. Figure 3-1 shows the functional blocks of the transmitter. Parallel data flows from the FPGA logic into the FPGA TX interface, through the PCS and PMA, and then out the TX driver as high-speed serial data.

Figure 3-1: **GTX/GTH Transceiver TX Block Diagram**

The key elements within the GTX/GTH transceiver TX are:

1. FPGA TX Interface, page 100
2. TX 8B/10B Encoder, page 108
3. TX Gearbox, page 112
4. TX Buffer, page 125
5. TX Buffer Bypass, page 127
6. TX Pattern Generator, page 137
7. TX Polarity Control, page 141

8. TX Fabric Clock Output Control, page 141
9. TX Phase Interpolator PPM Controller, page 146
10. TX Configurable Driver, page 148
11. TX Receiver Detect Support for PCI Express Designs, page 155
12. TX Out-of-Band Signaling, page 157

FPGA TX Interface

Functional Description

The FPGA TX interface is the FPGA's gateway to the TX datapath of the GTX/GTH transceiver. Applications transmit data through the GTX/GTH transceiver by writing data to the TXDATA port on the positive edge of TXUSRCLK2. The width of the port can be configured to be two, four, or eight bytes wide. The actual width of the port depends on the TX_DATA_WIDTH and TX_INT_DATAWIDTH attributes and TX8B10BEN port setting. Port widths can be 16, 20, 32, 40, 64, and 80 bits. The rate of the parallel clock (TXUSRCLK2) at the interface is determined by the TX line rate, the width of the TXDATA port, and whether or not 8B/10B encoding is enabled. A second parallel clock (TXUSRCLK) must be provided for the internal PCS logic in the transmitter. This section shows how to drive the parallel clocks and explains the constraints on those clocks for correct operation. The highest transmitter data rates require an 8-byte interface to achieve a TXUSRCLK2 rate in the specified operating range.

Interface Width Configuration

The 7 series FPGA GTX/GTH transceiver contains 2-byte and 4-byte internal datapath and is configurable by setting the TX_INT_DATAWIDTH attribute. The FPGA interface width is configurable by setting the TX_DATA_WIDTH attribute. When the 8B/10B encoder is enabled, the TX_DATA_WIDTH attribute must be configured to 20 bits, 40 bits, or 80 bits, and in this case, the FPGA TX interface only uses the TXDATA ports. For example, TXDATA[15:0] is used when the FPGA interface width is 16. When the 8B/10B encoder is bypassed, the TX_DATA_WIDTH attribute can be configured to any of the available widths: 16, 20, 32, 40, 64 or 80 bits.

Table 3-1 shows how the interface width for the TX datapath is selected. 8B/10B encoding is described in more detail in TX 8B/10B Encoder, page 108.

Table 3-1: **FPGA TX Interface Datapath Configuration**

TX8B10BEN	TX_DATA_WIDTH	TX_INT_DATAWIDTH	FPGA Interface Width	Internal Data Width
1	20	0	16	20
	40	0	32	20
	40	1	32	40
	80	1	64	40

Table 3-1: **FPGA TX Interface Datapath Configuration *(Cont'd)***

TX8B10BEN	TX_DATA_WIDTH	TX_INT_DATAWIDTH	FPGA Interface Width	Internal Data Width
0	16	0	16	16
	20	0	20	20
	32	0	32	16
	32	1	32	32
	40	0	40	20
	40	1	40	40
	64	1	64	32
	80	1	80	40

When the 8B/10B encoder is bypassed and the TX_DATA_WIDTH is 20, 40, or 80, the TXCHARDISPMODE and TXCHARDISPVAL ports are used to extend the TXDATA port from 16 to 20 bits, 32 to 40 bits, or 64 to 80 bits. Table 3-2 shows the data transmitted when the 8B/10B encoder is disabled. When the TX gearbox is used, refer to TX Gearbox, page 112 for data transmission order.

Table 3-2: **TX Data Transmitted when 8B/10B Encoder Bypassed**

	< < < Data Transmission Order is Right to Left (LSB to MSB) < < <																																																	
	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Data Transmitted	TXCHARDISPMODE[3]	TXCHARDISPVAL[3]	TXDATA[31:24]								TXCHARDISPMODE[2]	TXCHARDISPVAL[2]	TXDATA[23:16]								TXCHARDISPMODE[1]	TXCHARDISPVAL[1]	TXDATA[15:8]								TXCHARDISPMODE[0]	TXCHARDISPVAL[0]	TXDATA[7:0]					

	< < < Data Transmission Order is Right to Left (LSB to MSB) < < <																																																	
	79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40
Data Transmitted	TXCHARDISPMODE[7]	TXCHARDISPVAL[7]	TXDATA[63:56]								TXCHARDISPMODE[6]	TXCHARDISPVAL[6]	TXDATA[55:48]								TXCHARDISPMODE[5]	TXCHARDISPVAL[5]	TXDATA[47:40]								TXCHARDISPMODE[4]	TXCHARDISPVAL[4]	TXDATA[39:30]					

TXUSRCLK and TXUSRCLK2 Generation

The FPGA TX interface includes two parallel clocks: TXUSRCLK and TXUSRCLK2. TXUSRCLK is the internal clock for the PCS logic in the GTX/GTH transmitter. The required rate for TXUSRCLK depends on the internal datapath width of the GTXE2_CHANNEL/GTHE2_CHANNEL primitive and the TX line rate of the GTX/GTH transmitter. Equation 3-1 shows how to calculate the required rate for TXUSRCLK.

$$TXUSRCLK\ Rate = \frac{Line\ Rate}{Internal\ Datapath\ Width} \qquad \textit{Equation 3-1}$$

TXUSRCLK2 is the main synchronization clock for all signals into the TX side of the GTX/GTH transceiver. Most signals into the TX side of the GTX/GTH transceiver are sampled on the positive edge of TXUSRCLK2. TXUSRCLK2 and TXUSRCLK have a fixed-rate relationship based on the TX_DATA_WIDTH and TX_INT_DATAWIDTH settings. Table 3-3 shows the relationship between TXUSRCLK2 and TXUSRCLK per TX_DATA_WIDTH and TX_INT_DATAWIDTH values. A line rate greater than 6.6 Gb/s requires a 4-byte internal datapath by setting TX_INT_DATAWIDTH to 1.

Table 3-3: **TXUSRCLK2 Frequency Relationship to TXUSRCLK**

FPGA Interface Width	TX_DATA_WIDTH	TX_INT_DATAWIDTH	TXUSRCLK2 Frequency
2-Byte	16, 20	0	$F_{TXUSRCLK2} = F_{TXUSRCLK}$
4-Byte	32, 40	0	$F_{TXUSRCLK2} = F_{TXUSRCLK}/2$
4-Byte	32, 40	1	$F_{TXUSRCLK2} = F_{TXUSRCLK}$
8-Byte	64, 80	1	$F_{TXUSRCLK2} = F_{TXUSRCLK}/2$

These rules about the relationships between clocks must be observed for TXUSRCLK and TXUSRCLK2:

- TXUSRCLK and TXUSRCLK2 must be positive-edge aligned, with as little skew as possible between them. As a result, low-skew clock resources (BUFGs and BUFRs) should be used to drive TXUSRCLK and TXUSRCLK2.
- Even though they might run at different frequencies, TXUSRCLK, TXUSRCLK2, and the transmitter reference clock must have the same oscillator as their source. Thus TXUSRCLK and TXUSRCLK2 must be multiplied or divided versions of the transmitter reference clock.

Ports and Attributes

Table 3-4 defines the FPGA TX Interface ports.

Table 3-4: **FPGA TX Interface Ports**

Port	Dir	Clock Domain	Description
TXCHARDISPMODE[7:0]	In	TXUSRCLK2	When 8B/10B encoding is disabled, TXCHARDISPMODE is used to extend the data bus for 20-, 40- and 80-bit TX interfaces.
TXCHARDISPVAL[7:0]	In	TXUSRCLK2	When 8B/10B encoding is disabled, TXCHARDISPVAL is used to extend the data bus for 20-, 40- and 80-bit TX interfaces.

Table 3-4: **FPGA TX Interface Ports *(Cont'd)***

Port	Dir	Clock Domain	Description
TXDATA[63:0]	In	TXUSRCLK2	The bus for transmitting data. The width of this port depends on TX_DATA_WIDTH: TX_DATA_WIDTH = 16, 20: TXDATA[15:0] = 16 bits wide TX_DATA_WIDTH = 32, 40: TXDATA[31:0] = 32 bits wide TX_DATA_WIDTH = 64, 80: TXDATA[63:0] = 64 bits wide When a 20-bit, 40-bit, or 80-bit bus is required, the TXCHARDISPVAL and TXCHARDISPMODE ports from the 8B/10B encoder is concatenated with the TXDATA port. See Table 3-2, page 101.
TXUSRCLK	In	Clock	This port is used to provide a clock for the internal TX PCS datapath.
TXUSRCLK2	In	Clock	This port is used to synchronize the FPGA logic with the TX interface. This clock must be positive-edge aligned to TXUSRCLK when TXUSRCLK is provided by the user.

Table 3-5 defines the FPGA TX interface attributes.

Table 3-5: **FPGA TX Interface Attributes**

Attribute	Type	Description
TX_DATA_WIDTH	Integer	Sets the bit width of the TXDATA port. When 8B/10B encoding is enabled, TX_DATA_WIDTH must be set to 20, 40, or 80. Valid settings are 16, 20, 32, 40, 64, and 80. See Interface Width Configuration, page 100 for more information.
TX_INT_DATAWIDTH	Integer	Controls the width of the internal datapath. 0: 2-byte internal datapath 1: 4-byte internal datapath. Set to 1 if a line rate is greater than 6.6 Gb/s.

Using TXOUTCLK to Drive the TX Interface

Depending on the TXUSRCLK and TXUSRCLK2 frequencies, there are different ways FPGA clock resources can be used to drive the parallel clock for the TX interface. Figure 3-2 through Figure 3-5 show different ways FPGA clock resources can be used to drive the parallel clocks for the TX interface. In these examples, the TXOUTCLK is derived from the MGTREFCLK0[P/N] or MGTREFCLK1[P/N] and the TXOUTCLKSEL = 011 to select the TXPLLREFCLK_DIV1 path as indicated in Figure 3-28, page 142.

- Depending on the input reference clock frequency and the required line rate, an MMCM and the appropriate TXOUTCLKSEL port setting is required. The

CORE Generator™ tool creates a sample design based on different design requirements for most cases.

- In use models where TX buffer is bypassed, there are additional restrictions on the clocking resources. Refer to TX Buffer Bypass, page 127 for more information.

TXOUTCLK Driving GTX/GTH Transceiver TX in 2-Byte or 4-Byte Mode

In Figure 3-2, TXOUTCLK is used to drive TXUSRCLK and TXUSRCLK2 for 2-byte mode (TX_DATA_WIDTH = 16 or 20 and TX_INT_DATWIDTH = 0) or 4-byte mode (TX_DATA_WIDTH = 32 or 40 and TX_INT_DATWIDTH = 1) in a single-lane configuration. In both cases, the frequency of TXUSRCLK2 is equal to TXUSRCLK.

Figure 3-2: **Single Lane—TXOUTCLK Drives TXUSRCLK2 (2-Byte or 4-Byte Mode)**

Notes relevant to Figure 3-2:

1. In Virtex®-7devices, BUFR via BUFMR can be used with certain limitations. For details about placement constraints and restrictions on clocking resources (MMCM, BUFR, BUFG, etc.), refer to UG472, *7 Series FPGAs Clocking Resources User Guide*.
2. $F_{TXUSRCLK2} = F_{TXUSRCLK}$.

Similarly, Figure 3-3 shows the shows the same settings in multiple lanes configuration.

Figure 3-3: **Multiple Lanes—TXOUTCLK Drives TXUSRCLK2 (2-Byte or 4-Byte Mode)**

Notes relevant to Figure 3-3:

1. In Virtex-7 devices, BUFR via BUFMR can be used with certain limitations. For details about placement constraints and restrictions on clocking resources (MMCM, BUFR, BUFG, etc.), refer to UG472, *7 Series FPGAs Clocking Resources User Guide*.
2. $F_{TXUSRCLK2} = F_{TXUSRCLK}$.

TXOUTCLK Driving GTX/GTH Transceiver TX in 4-Byte or 8-Byte Mode

In Figure 3-4, TXOUTCLK is used to drive TXUSRCLK2 for 4-byte mode (TX_DATA_WIDTH = 32 or 40 and TX_INT_DATWIDTH = 0) or 8-byte mode (TX_DATA_WIDTH = 64 or 80 and TX_INT_DATWIDTH = 1). In both cases, the frequency of TXUSRCLK2 is equal to half of the frequency of TXUSRCLK. MMCM, which is part of the Clock Management Tile (CMT) located in the top half of the device, can only drive the BUFGs in the top half of the devices. Similarly, MMCM located in the bottom half can only drive BUFGs in the bottom half.

Figure 3-4: **Single Lane—TXOUTCLK Drives TXUSRCLK2 (4-Byte or 8-Byte Mode)**

Notes relevant to Figure 3-4:

1. In Kintex™-7 devices, BUFH/BUFG is required. In Virtex-7 devices, BUFH/BUFG is not required.
2. $F_{TXUSRCLK2} = F_{TXUSRCLK}/2$
3. In Virtex-7 devices, BUFR can be used with certain limitations. For details about placement constraints and restrictions on clocking resources (MMCM, BUFR, BUFG, etc.), refer to UG472, *7 Series FPGAs Clocking Resources User Guide*.

Similarly, Figure 3-5 shows the shows the same settings in multiple lanes configuration.

Figure 3-5: **Multiple Lanes—TXOUTCLK Drives TXUSRCLK2 (4-Byte or 8-Byte Mode)**

Notes relevant to Figure 3-5:

1. In Kintex-7 devices, BUFH/BUFG is required. In Virtex-7 devices, BUFH/BUFG is not required.
2. $F_{TXUSRCLK2} = F_{TXUSRCLK}/2$.
3. In Virtex-7 devices, BUFR can be used with certain limitations. For details about placement constraints and restrictions on clocking resources (MMCM, BUFR, BUFG, etc.), refer to UG472, *7 Series FPGAs Clocking Resources User Guide*.

TX 8B/10B Encoder

Functional Description

Many protocols use 8B/10B encoding on outgoing data. 8B/10B is an industry standard encoding scheme that trades two bits overhead per byte for achieved DC-balance and bounded disparity to allow reasonable clock recovery. The GTX/GTH transceiver has a built-in 8B/10B TX path to encode TX data without consuming FPGA resources. Enabling the 8B/10B encoder increases latency through the TX path. The 8B/10B encoder can be disabled or bypassed to minimize latency, if not needed.

8B/10B Bit and Byte Ordering

The order of the bits after the 8B/10B encoder is the opposite of the order shown in Appendix C, 8B/10B Valid Characters, because 8B/10B encoding requires bit a0 to be transmitted first, and the GTX/GTH transceiver always transmits the right-most bit first. To match with 8B/10B, the 8B/10B encoder in the GTX/GTH transceiver automatically reverses the bit order. Figure 3-6 shows data transmitted by the GTX/GTH transceiver when TX_DATA_WIDTH = 20, 40, and 80. The number of bits used by TXDATA and corresponding byte orders are determined by TX_DATA_WIDTH.

- Only use TXDATA[15:0] if TX_DATA_WIDTH = 20
- Only use TXDATA[31:0] if TX_DATA_WIDTH = 40
- Use full TXDATA[63:0] if TX_DATA_WIDTH = 80

When the 8B/10B encoder is bypassed and TX_DATA_WIDTH is set to a multiple of 10, 10-bit characters are passed to TX data interface with this format:

- The corresponding TXCHARDISPMODE represents the 9th bit
- The corresponding TXCHARDISPVAL represents the 8th bit
- The corresponding TXDATA byte represents [7:0] bits

K Characters

The 8B/10B table includes special characters (K characters) that are often used for control functions. TXCHARISK ports are used to indicate if data on TXDATA are K characters or regular data. The 8B/10B encoder checks received TXDATA byte to match any K character if corresponding TXCHARISK bit is driven High.

Figure 3-6: **8B/10B Bit and Byte Ordering**

Running Disparity

8B/10B coding is DC-balanced, meaning that the long-term ratio of 1s and 0s transmitted should be exactly 50%. To achieve this, the encoder always calculates the difference between the number of 1s transmitted and the number of 0s transmitted, and at the end of each character transmitted, makes the difference either +1 or –1. This difference is known as the *running disparity*.

To accommodate protocols that use disparity to send control information, the running disparity not only can be generated by the 8B/10B encoder but is also controllable through TXCHARDISPMODE and TXCHARDISPVAL as shown in Table 3-6. For example, an Idle character sent with reversed disparity might be used to trigger clock correction.

Table 3-6: **TXCHARDISPMODE and TXCHARDISPVAL versus Outgoing Disparity**

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

Ports and Attributes

Table 3-7 lists the ports required by the TX 8B/10B encoder.

Note: There are no TX encoder attributes.

Table 3-7: **TX 8B/10B Encoder Ports**

Port	Dir	Clock Domain	Description
TX8B10BBYPASS[7:0]	In	TXUSRCLK2	This active-High port allows byte-interleaved data to bypass 8B/10B on a per-byte basis. TX8B10BEN must be High to use this per-byte bypass mode. TX8B10BBYPASS [7] corresponds to TXDATA[63:56] TX8B10BBYPASS [6] corresponds to TXDATA[55:48] TX8B10BBYPASS [5] corresponds to TXDATA[47:40] TX8B10BBYPASS [4] corresponds to TXDATA[39:32] TX8B10BBYPASS [3] corresponds to TXDATA[31:24] TX8B10BBYPASS [2] corresponds to TXDATA[23:16] TX8B10BBYPASS [1] corresponds to TXDATA[15:8] TX8B10BBYPASS [0] corresponds to TXDATA[7:0] TXBYPASS8B10B[x] = 1, encoder for byte x is bypassed. TXBYPASS8B10B[x] = 0, encoder for byte x is used.
TX8B10BEN	In	TXUSRCLK2	TX8B10BEN is set High to enable the 8B/10B encoder. TX_DATA_WIDTH must be set to 20, 40, or 80 when the 8B/10B encoder is enabled. 0: 8B/10B encoder bypassed. This option reduces latency. 1: 8B/10B encoder enabled.

Table 3-7: **TX 8B/10B Encoder Ports *(Cont'd)***

Port	Dir	Clock Domain	Description
TXCHARDISPMODE[7:0]	In	TXUSRCLK2	Set High to work with TXCHARDISPVAL to force running disparity negative or positive when encoding TXDATA. Set Low to use normal running disparity. Refer to Table 3-6 for a detailed definition. TXCHARDISPMODE[7] corresponds to TXDATA[63:56] TXCHARDISPMODE[6] corresponds to TXDATA[55:48] TXCHARDISPMODE[5] corresponds to TXDATA[47:40] TXCHARDISPMODE[4] corresponds to TXDATA[39:32] TXCHARDISPMODE[3] corresponds to TXDATA[31:24] TXCHARDISPMODE[2] corresponds to TXDATA[23:16] TXCHARDISPMODE[1] corresponds to TXDATA[15:8] TXCHARDISPMODE[0] corresponds to TXDATA[7:0]
TXCHARDISPVAL[7:0]	In	TXUSRCLK2	Work with TXCHARDISPMODE to provide running disparity control. Refer to Table 3-6 for detailed information. TXCHARDISPVAL[7] corresponds to TXDATA[63:56] TXCHARDISPVAL[6] corresponds to TXDATA[55:48] TXCHARDISPVAL[5] corresponds to TXDATA[47:40] TXCHARDISPVAL[4] corresponds to TXDATA[39:32] TXCHARDISPVAL[3] corresponds to TXDATA[31:24] TXCHARDISPVAL[2] corresponds to TXDATA[23:16] TXCHARDISPVAL[1] corresponds to TXDATA[15:8] TXCHARDISPVAL[0] corresponds to TXDATA[7:0]
TXCHARISK[7:0]	In	TXUSRCLK2	When High, indicates the corresponding data byte on TXDATA is a valid K character. TXCHARISK[7] corresponds to TXDATA[63:56] TXCHARISK[6] corresponds to TXDATA[55:48] TXCHARISK[5] corresponds to TXDATA[47:40] TXCHARISK[4] corresponds to TXDATA[39:32] TXCHARISK[3] corresponds to TXDATA[31:24] TXCHARISK[2] corresponds to TXDATA[23:16] TXCHARISK[1] corresponds to TXDATA[15:8] TXCHARISK[0] corresponds to TXDATA[7:0] A TXCHARISK bit should be driven Low when the corresponding data byte from TXDATA is set to bypass the 8B/10B encoder.

Enabling and Disabling 8B/10B Encoding

To enable the 8B/10B encoder, TX8B10BEN must be driven High. The TX 8B/10B encoder allows byte interleaved data to bypass the encoder on a per-byte basis. When TX8B10BEN is driven Low, all encoders are turned off and no data from TXDATA can be encoded. When TX8B10BEN is High, driving a bit from TX8B10BBYPASS High can make the corresponding byte channel from TXDATA bypass 8B/10B encoding. When the encoder is turned off, the operation of the TXDATA port is as described in the FPGA TX interface.

TX Gearbox

Functional Description

Some high-speed data rate protocols use 64B/66B encoding to reduce the overhead of 8B/10B encoding while retaining the benefits of an encoding scheme. The TX gearbox provides support for 64B/66B and 64B/67B header and payload combining. The Interlaken interface protocol specification uses the 64B/67B encoding scheme. Refer to the Interlaken specification for further information. The Interlaken specification can be downloaded from: http://www.cortina-systems.com/news/interlaken.

The TX gearbox supports 2-byte, 4-byte and 8-byte interfaces. Scrambling of the data is done in the FPGA logic. In the GTH transceiver, a CAUI interface mode is also supported in addition to the normal gearbox mode.

Ports and Attributes

Table 3-8 defines the TX gearbox ports.

Table 3-8: **TX Gearbox Ports**

Port Name	Dir	Clock Domain	Description
TXGEARBOXREADY	Out	TXUSRCLK2	This output indicates if data can be applied to the 64B/66B or 64B/67B gearbox when GEARBOX_MODE is set to use the gearbox. 0: No data can be applied 1: Data must be applied
TXHEADER[2:0]	In	TXUSRCLK2	These ports are the header inputs. [1:0] are used for the 64B/66B gearbox, and [2:0] are used for the 64B/67B gearbox. GTH transceiver: TXHEADER[2:0] are used for the header in normal mode or for datastream A in CAUI interface mode.
TXCHARISK[2:0] (GTH transceiver only)	In	TXUSRCLK2	Used as TXHEADER[2:0] for datastream B in CAUI interface mode.

Table 3-8: **TX Gearbox Ports *(Cont'd)***

Port Name	Dir	Clock Domain	Description
TXSEQUENCE[6:0]	In	TXUSRCLK2	These inputs are used for the fabric sequence counter when the TX gearbox is used. [5:0] are used for the 64B/66 B gearbox, and [6:0] are used for the 64B/67B gearbox. This port is shared by both PCS lanes (PCSLs) in CAUI interface mode in the GTH transceiver.
TXSTARTSEQ	In	TXUSRCLK2	This input indicates the first word to be applied after reset for the 64B/66B or 64B/67B gearbox. The internal sequencer counter must be enabled by the GEARBOX_MODE attribute. This port is shared by both PCSLs in CAUI interface mode in the GTH transceiver.

Table 3-9 defines the TX gearbox attributes.

Table 3-9: **TX Gearbox Attributes**

Attribute	Type	Description
GEARBOX_MODE	3-bit Binary	This attribute indicates the TX and RX gearbox modes: • Bit 2: Unused and set to 0 in the GTX transceiver, and in normal mode in the GTH transceiver. Set to 1 when the CAUI interface is engaged in the GTH transceiver. • Bit 1: 0: Use the external sequence counter and apply inputs to TXSEQUENCE. 1: Use the internal sequence counter, gate the input header and data with the TXGEARBOXREADY output. GTH transceiver: Unused and set to 1. Internal sequence counter is not supported. • Bit 0: 0: 64B/67B gearbox mode for Interlaken. 1: 64B/66B gearbox.
TXGEARBOX_EN	Boolean	When TRUE, this attribute enables the TX gearbox.

Enabling the TX Gearbox

To enable the TX gearbox for the GTX/GTH transceiver, the TXGEARBOX_EN attribute should be set to TRUE.

Bit 2 of the GEARBOX_MODE attribute is unused and must be set to 0 in the GTX transceiver and when in normal operating mode in the GTH transceiver. It is set to 1 to

engage the CAUI interface in the GTH transceiver. The GTX/GTH transceiver's TX gearbox and RX gearbox use the same mode.

TX Gearbox Bit and Byte Ordering

Figure 3-7 shows an example of the first four cycles of data entering and exiting the TX gearbox for 64B/66B encoding when using a 4-byte logic interface (TX_DATA_WIDTH = 32 (4-byte), TX_INT_DATAWIDTH = 1 (4-byte)) in normal mode (GEARBOX_MODE[2] = `1'b0`). The input consists of a 2-bit header and 32 bits of data. On the first cycle, the header and 30 bits of data exit the TX gearbox. On the second cycle, the remaining two data bits from the previous cycle's TXDATA input along with 30 data bits from the current TXDATA input exit the TX gearbox. On the third cycle, the output of the TX gearbox contains two remaining data bits from the first 66-bit block, the header of the second 66-bit block, and 28 data bits from the second 66-bit block.

Figure 3-7: **TX Gearbox Bit Ordering in Normal Mode (GEARBOX_MODE[2] = `1'b0`)**

Note relevant to Figure 3-7:

1. Per IEEE802.3ae nomenclature, H1 corresponds to TxB<0>, H0 to TxB<1>, etc.

TX Gearbox Operating Modes

The TX gearbox has two operating modes. The external sequence counter operating mode must be implemented in user logic. The second mode uses an internal sequence counter. The internal sequence counter mode is not supported in the GTH transceiver. The TX gearbox supports 2-byte, 4-byte, and 8-byte interfaces to the FPGA logic.

External Sequence Counter Operating Mode

As shown in Figure 3-8, the external sequence counter operating mode uses the TXSEQUENCE [6:0], TXDATA[63:0], and TXHEADER[2:0] inputs when in normal mode (GEARBOX_MODE[2] = `1'b0`). (TXCHARISK[2:0] is also used when the CAUI interface is used in the GTH transceiver (GEARBOX_MODE[2] = `1'b1`).) A binary counter must exist in the user logic to drive the TXSEQUENCE port. For 64B/66B encoding, the counter increments from 0 to 32 and repeats from 0. For 64B/67B encoding, the counter increments from 0 to 66 and repeats from 0. When using 64B/66B encoding, tie TXSQUENCE [6] to logic 0 and tie the unused TXHEADER [2] to logic 0. (The unused TXCHARISK[2] should be tied to logic 0 when using the CAUI interface in the GTH transceiver for GEARBOX_MODE[2] = `1'b1`). The sequence counter increment ranges ({0 to 32}, {0 to 66}) are identical for 2-byte, 4-byte and 8-byte interfaces. However, the counter must increment once every two TXUSRCLK2 cycles when using a mode where TX_DATA_WIDTH is the same as TX_INT_DATAWIDTH (e.g., a 4-byte fabric interface (TX_DATA_WIDTH = 32) and a 4-byte internal data width (TX_INT_DATAWIDTH= 1)).

Figure 3-8: **TX Gearbox in External Sequence Counter Operating Mode in Normal Mode (GEARBOX_MODE[2] = `1'b0`)**

Due to the nature of the 64B/66B and 64B/67B encoding schemes, user data is held (paused) during various sequence counter values. Data is paused for two TXUSRCLK2 cycles in modes with the same TX_DATA_WIDTH and TX_INT_DATAWIDTH, and for one TXUSRCLK2 cycle in modes where TX_DATA_WIDTH is twice the TX_INT_DATAWIDTH. Valid data transfer is resumed on the next TXUSRCLK2 cycle. The data pause only applies to TXDATA and not to TXHEADER. The TXSEQUENCE pause locations for various modes are described in Table 3-10 and Table 3-11.

Table 3-10: **64B/66B Encoding Frequency of TXSEQUENCE and Pause Locations in Normal Mode (GEARBOX_MODE[2] = `1'b0`)**

TX_DATA_WIDTH	TX_INT_DATAWIDTH	Frequency of TXSEQUENCE	TXSEQUENCE PAUSE
64 (8-byte)	1 (4-byte)	1 X TXUSRCLK2	32
32 (4-byte)	1 (4-byte)	2 X TXUSRCLK2	32
32 (4-byte)	0 (2-byte)	1 X TXUSRCLK2	31
16 (2-byte)	0 (2-byte)	2 X TXUSRCLK2	31

Table 3-11: **64B/67B Encoding Frequency of TXSEQUENCE and Pause Locations in Normal Mode (GEARBOX_MODE[2] = `1'b0`)**

TX_DATA_WIDTH	TX_INT_DATAWIDTH	Frequency of TXSEQUENCE	TXSEQUENCE PAUSE
64 (8-byte)	1 (4-byte)	1 X TXUSRCLK2	22, 44, 66
32 (4-byte)	1 (4-byte)	2 X TXUSRCLK2	22, 44, 66
32 (4-byte)	0 (2-byte)	1 X TXUSRCLK2	21, 44, 65
16 (2-byte)	0 (2-byte)	2 X TXUSRCLK2	21, 44, 65

Figure 3-9 shows how a pause occurs at counter value 32 when using an 8-byte fabric interface and a 4-byte internal datapath in external sequence counter mode with 64B/66B encoding in normal mode (GEARBOX_MODE[2] = `1'b0`).

Figure 3-9: **Pause at Sequence Counter Value 32 in Normal Mode (GEARBOX_MODE[2] = `1'b0`)**

Figure 3-10 shows how a pause occurs at counter value 44 when using a 2-byte fabric interface with a 2-byte internal datapath in external sequence counter mode with 64B/67B encoding in normal mode (GEARBOX_MODE[2] = `1'b0`).

Figure 3-10: **Pause at Sequence Counter Value 44 in Normal Mode (GEARBOX_MODE[2] = `1'b0`)**

The sequence of transmitting 64/67 data for the external sequence counter mode using a 2-byte internal datapath (TX_INT_DATAWIDTH = 0) in normal mode (GEARBOX_MODE[2] = `1'b0`) is:

1. Apply GTTXRESET and wait until the reset cycle is completed.
2. During reset, apply `7'h00` to TXSEQUENCE, header information to TXHEADER, and initial data to TXDATA. This state can be held indefinitely until data transmission is ready.
3. On count 0, apply data to TXDATA and header information to TXHEADER. For a 2-byte interface (TX_DATA_WIDTH = 16), drive the second 2 bytes to TXDATA while still on count 0.
4. The sequence counter increments to 1 while data is driven on TXDATA.

5. After applying 4 bytes of data, the counter increments to 2. Apply data on TXDATA and header information on TXHEADER.
6. On count 21, stop data pipeline.
7. On count 22, drive data on TXDATA.
8. On count 44, stop data pipeline.
9. On count 45, drive data on TXDATA.
10. On count 65, stop data pipeline.
11. On count 66, drive data on TXDATA.

The sequence of transmitting 64/67 data for the external sequence counter mode using the 4-byte internal datapath (TX_INT_DATAWIDTH = 1) in normal mode (GEARBOX_MODE[2] = `1'b0`) is as follows:

1. Apply GTTXRESET and wait until the reset cycle is completed.
2. During reset, apply `7'h00` to TXSEQUENCE, header information to TXHEADER, and initial data to TXDATA. This state can be held indefinitely until data transmission is ready.
3. On count 0, apply data to TXDATA and header information to TXHEADER. For a 4-byte interface (TX_DATA_WIDTH = 32), drive the second 4 bytes to TXDATA while still on count 0.
4. Sequence counter increments to 1 while data is driven on TXDATA.
5. After applying 8 bytes of data, the counter increments to 2. Drive data on TXDATA and header information on TXHEADER.
6. On count 22, stop data pipeline.
7. On count 23, drive data on TXDATA.
8. On count 44, stop data pipeline.
9. On count 45, drive data on TXDATA.
10. On count 66, stop data pipeline.

The sequence of transmitting 64/66 data for the external sequence counter mode using the 2-byte internal datapath (TX_INT_DATAWIDTH = 0) in normal mode (GEARBOX_MODE[2] = `1'b0`) is as follows:

1. Apply GTTXRESET and wait until the reset cycle is completed.
2. During reset, apply `6'h00` to TXSEQUENCE, the appropriate header data to TXHEADER, and initial data to TXDATA. This state can be held indefinitely until data transmission is ready.
3. On count 0, apply data to TXDATA and header information to TXHEADER. For a 2-byte interface (TX_DATA_WIDTH = 16), drive the second 2 bytes to TXDATA while still on count 0.
4. The sequence counter increments to 1 while data is driven on TXDATA.
5. After applying 4 bytes of data, the counter increments to 2. Drive data on TXDATA and header information on TXHEADER.
6. On count 31, stop data pipeline.
7. On count 32, drive data on TXDATA.

The sequence of transmitting 64/66 data for the external sequence counter mode using a 4-byte internal datapath (TX_INT_DATAWIDTH = 1) in normal mode (GEARBOX_MODE[2] = `1'b0`) is as follows:

1. Apply GTTXRESET and waits until the reset cycle is completed.
2. During reset, apply `6'h00` to TXSEQUENCE, the appropriate header data to TXHEADER, and initial data to TXDATA. This state can be held indefinitely until data transmission is ready.
3. On count 0, drive data to TXDATA and header information to TXHEADER. For a 4-byte interface (TX_DATA_WIDTH = 32), drive the second 4 bytes to TXDATA while still on count 0.
4. The sequence counter increments to 1 while data is driven on TXDATA.
5. After applying 8 bytes of data, the counter increments to 2. Drive data on TXDATA and header information on TXHEADER.
6. On count 32, stop data pipeline.

Internal Sequence Counter Operating Mode (GTX Transceiver Only)

As shown in Figure 3-11, the internal sequence counter operating mode uses the TXSTARTSEQ input and the TXGEARBOXREADY output, in addition to the TXDATA data inputs and the TXHEADER header inputs in the GTX transceiver. In this use model, the TXSEQUENCE inputs are not used. The use model is similar to the previous use model, except that the TXGEARBOXREADY output is used.

Note: This mode is not supported in the GTH transceiver. Thus, it is also not supported in the CAUI interface mode (GEARBOX_MODE[2] = `1'b1`) which is only available in the GTH transceiver.

Figure 3-11: **TX Gearbox in Internal Sequence Counter Mode in GTX Transceiver**

The TXSTARTSEQ input indicates to the TX gearbox when the first byte of data after a reset is valid. TXSTARTSEQ is asserted High when the first byte of valid data is applied after a reset condition. The TXDATA and TXHEADER inputs must be held stable after reset, and TXSTARTSEQ must be held Low until data can be applied continuously. There are no requirements on how long a user can wait before starting to transmit data. TXSTARTSEQ is asserted High along with the first 2, 4, or 8 bytes of valid data and not before. After the first bytes of data, TXSTARTSEQ can be held at any value that is convenient.

After data is driven, TXGEARBOXREADY is deasserted Low for either two TXUSRCLK2 cycles or three TXUSRCLK2 cycles, depending on the TX_DATA_WIDTH and

TX_INT_DATAWIDTH. TXGEARBOXREADY is deasserted for two TXUSRCLK2 cycles for modes where TX_DATA_WIDTH is twice that of TX_INT_DATAWIDTH. It is deasserted for three TXUSRCLK2 cycles for modes with the same TX_DATA_WIDTH and TX_INT_DATAWIDTH. Figure 3-12 and Figure 3-13 show the behavior of TXGEARBOXREADY for an 8-byte fabric interface and a 2-byte fabric interface in the GTX transceiver, respectively. When TXGEARBOXREADY is deasserted Low, only one TXUSRCLK2 cycle remains before the data pipe must be stopped. The 1-cycle latency is fixed and cannot be changed. After one cycle of latency, data must be held through until TXGEARBOXREADY transitions High, where new data must be driven. For this mode of operation, the number of hold points is identical to when using the external sequence counter mode for 64B/67B and 64B/66B.

Figure 3-12: **TX Gearbox Internal Sequence Mode in GTX Transceiver, TX_DATA_WIDTH = 64 (8-Byte), TX_INT_DATAWIDTH = 1 (4-Byte), 64B/66B**

Figure 3-13: **TX Gearbox Internal Sequence Mode in GTX Transceiver, TX_DATA_WIDTH = 16 (2-Byte), TX_INT_DATAWIDTH = 0 (2-Byte), 64B/67B**

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

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

CAUI Interface (GTH Transceiver)

The CAUI interface requires two data interfaces on the transceiver. This section describes the design of the CAUI interface block on the TX that is implemented in the GTH transceiver. This supports a dual data interface in the 64B/66B and 64B/67B modes (datastream A and datastream B). CAUI interface mode can be selected by setting the attribute GEARBOX_MODE[2] to `1'b1`. When in CAUI interface mode, the only allowed settings are TX_INT_DATAWIDTH = 1 (4-byte) and TX_DATA_WDTH = 64 (8-byte) or 32 (4-byte).

Use Case

Two PCSLs are expected to feed data into each PCS across the CAUI interface. Each PCSL connects to half of the TXDATA ports. Figure 3-14 shows the expected connections between PCSLs and the PCS.

Figure 3-14: **CAUI Interface - TX Use Case**

In Figure 3-14, x is the width of the PCSL data bus. Allowed values are 16 and 32. TXSEQUENCE and TXSTARTSEQ are shared by the two PCSLs.

TX Gearbox Block (GTH Transceiver)

The top level of the TX gearbox in the GTX transceiver has one instance of each of these components:

- 64B/66B 4-byte gearbox
- 64B/66B 2-byte gearbox
- 64B/67B 4-byte gearbox
- 64B/67B 2-byte gearbox
- Sequence counter

To support the CAUI interface, the GTH transceiver has an additional instance of each of the 2-byte gearboxes. Two instances (one for 64B/66B and one for 64B/67B mode) of the Bit Mux block is also added to merge the two data streams. The input TXHEADER[2:0] is used for header bits of datastream A. Input signal TXCHARISK[2:0] is used for the header bits of datastream B.

Figure 3-15 shows the CAUI interface (TX path) of the GTH transceiver.

Figure 3-15: **CAUI Interface (TX Datapath)**

When in CAUI interface mode and the PCSL data width is 32 bits each (TX_DATA_WIDTH = 64 (8-byte)), the 8-byte to 4-byte converter splits the data into two streams in such a way that datastream A and datastream B reach the corresponding gearbox as shown in Figure 3-16 and Figure 3-17.

Figure 3-16: **Input to the 8-Byte to 4-Byte Converter (TX_DATA_WIDTH = 64 (8-Byte), TX_INT_DATAWIDTH =1 (4-Byte), GEARBOX_MODE[2] = `1'b1`)**

Figure 3-17: **Output of the 8-Byte to 4-Byte Converter (TX_DATA_WIDTH = 64 (8-Byte), TX_INT_DATAWIDTH = 1 (4-Byte), GEARBOX_MODE[2] = `1'b1`)**

The Bit Mux block interleaves two bitstreams (two 16-bit inputs) to form one merged bitstream that is twice the width. The Bit Mux function is as described in clause 83.5.2 of IEEE Std 802.3ba-2010.

Although TX_INT_DATAWIDTH = 1 (4-byte) is used in CAUI interface mode, two 2-byte gearboxes are used to realize the functionality, as shown in Figure 3-15. The functionality of these 2-byte gearboxes is the same as described in External Sequence Counter Operating Mode, page 115, for the case when TX_INT_DATAWIDTH = 0 (2-byte). Internal sequence counter mode is not supported in the GTH transceiver. TXSEQUENCE pause locations for various modes are described in Table 3-12 and Table 3-13 when using the external sequence counter mode.

Table 3-12: **64B/66B Encoding Frequency of TXSEQUENCE and Pause Locations in CAUI Interface Mode in GTH Transceiver (GEARBOX_MODE[2] = `1'b1`)**

TX_DATA_WIDTH	TX_INT_DATAWIDTH	Frequency of TXSEQUENCE	TXSEQUENCE PAUSE (1)
64 (8-Byte)	1 (4-byte)	1 x TXUSRCLK2	31
32 (4-byte)	1 (4-byte)	2 x TXUSRCLK2	31

Notes:

1. Although the TX sequence pause location is 31, the external sequence counter should cycle through from 0–32 for proper operation as described in the external sequence counter operating sequence for 64B/66B for the case when TX_INT_DATAWIDTH = 0 (2-byte) on page 118.

Table 3-13: **64B/67B Encoding Frequency of TXSEQUENCE and Pause Locations in CAUI Interface Mode in GTH Transceiver (GEARBOX_MODE[2] = `1'b1`)**

TX_DATA_WIDTH	TX_INT_DATAWIDTH	Frequency of TXSEQUENCE	TXSEQUENCE PAUSE (1)
64 (8-Byte)	1 (4-byte)	1 x TXUSRCLK2	21, 44, 65
32 (4-byte)	1 (4-byte)	2 x TXUSRCLK2	21, 44, 65

Notes:

1. Although the TX sequence pause location stops at 65, the external sequence counter should cycle through from 0–66 for proper operation as described in the external sequence counter operating sequence for 64B/67B for the case when TX_INT_DATAWIDTH = 0 (2-byte) on page 117.

Due to additional functionality, latency through the gearbox block is expected to be longer in the GTH transceiver compared to the GTX transceiver.

TX Buffer

Functional Description

The GTX/GTH transceiver TX datapath has two internal parallel clock domains used in the PCS: the PMA parallel clock domain (XCLK) and the TXUSRCLK domain. To transmit data, the XCLK rate must match the TXUSRCLK rate, and all phase differences between the two domains must be resolved. Figure 3-18 shows the XCLK and TXUSRCLK domains.

Figure 3-18: **TX Clock Domains**

The GTX/GTH transmitter includes a TX buffer and a TX phase alignment circuit to resolve phase differences between the XCLK and TXUSRCLK domains. The TX phase alignment circuit is used when TX buffer is bypassed (see TX Buffer Bypass, page 127). All TX datapaths must use either the TX buffer or the TX phase alignment circuit. Table 3-14 shows trade-offs between buffering and phase alignment.

Table 3-14: **TX Buffering versus Phase Alignment**

	TX Buffer	TX Phase Alignment
Ease of Use	The TX buffer is the recommended default to use when possible. It is robust and easier to operate.	Phase alignment is an advanced feature that requires extra logic and additional constraints on clock sources. TXOUTCLKSEL must select the GTX/GTH transceiver reference clock as the source of TXOUTCLK to drive TXUSRCLK.
Latency	If low latency is critical, the TX buffer must be bypassed.	Phase alignment uses fewer register in the TX datapath to achieve lower and deterministic latency.
TX Lane-to-Lane Deskew		The TX phase alignment circuit can be used to reduce the lane skew between separate GTX/GTH transceivers. All GTX/GTH transceivers involved must use the same line rate.

Ports and Attributes

Table 3-15 defines the TX buffer ports.

Table 3-15: **TX Buffer Ports**

Port	Dir	Clock Domain	Description
TXBUFSTATUS[1:0]	Out	TXUSRCLK2	TX buffer status. TXBUFSTATUS[1]: TX buffer overflow or underflow status. When TXBUFSTATUS[1] is set High, it remains High until the TX buffer is reset. 1: TX FIFO has overflow or underflow. 0: No TX FIFO overflow or underflow error. TXBUFSTATUS[0]: TX buffer fullness. 1: TX FIFO is at least half full. 0: TX FIFO is less than half full.

Table 3-16 defines the TX buffer attributes.

Table 3-16: **TX Buffer Attributes**

Attribute	Type	Description
TXBUF_EN	Boolean	Use or bypass the TX buffer. TRUE: Uses the TX buffer (default). FALSE: Bypasses the TX buffer (advanced feature).

Table 3-16: **TX Buffer Attributes *(Cont'd)***

Attribute	Type	Description
TX_XCLK_SEL	String	Selects the clock source used to drive the PMA parallel clock domain (XCLK). TXOUT: Selects TXOUTCLK as source of XCLK. Use when using the TX buffer. TXUSR: Selects TXUSRCLK as source of XCLK. Used when bypassing the TX buffer.
TXBUF_RESET_ON_RATE_CHANGE	Boolean	GTX/GTH transceiver internally generated TX buffer reset on rate change. TRUE: Enables auto TX buffer reset on rate change. FALSE: Disables auto TX buffer reset on rate change.

Using the TX Buffer

The TX buffer should be reset whenever TXBUFSTATUS indicates an overflow or underflow condition. The TX buffer can be reset by using GTTXRESET, TXPCSRESET, or the GTX/GTH transceiver internally generated TX buffer reset on rate change when TXBUF_RESET_ON_RATE_CHANGE = TRUE (see TX Initialization and Reset, page 64). Assertion of GTTXRESET triggers a sequence that resets the entire transmitter of the GTX/GTH transceiver. These settings are use to enable the TX buffer to resolve phase differences between the XCLK and TXUSRCLK domains:

- TXBUF_EN = TRUE
- TX_XCLK_SEL = TXOUT

TX Buffer Bypass

Functional Description

Bypassing the TX buffer is an advanced feature of the 7 series GTX/GTH transceiver. The TX phase alignment circuit is used to adjust the phase difference between the PMA parallel clock domain (XCLK) and the TXUSRCLK domain when the TX buffer is bypassed. It also performs the TX delay alignment by adjusting the TXUSRCLK to compensate for the temperature and voltage variations. The combined TX phase and delay alignments can be automatically performed by the GTX transceiver or manually controlled by the user. For the GTH transceiver, this functionality must be controlled manually by the user. Refer to Figure 3-18, page 125 for the XCLK and TXUSRCLK domains and Table 3-14, page 126 for trade-offs between buffering and phase alignment.

Ports and Attributes

Table 3-17 defines the TX buffer bypass ports.

Table 3-17: **TX Buffer Bypass Ports**

Port	Dir	Clock Domain	Description
TXPHDLYRESET	In	Async	TX phase alignment hard reset to force TXUSRCLK to the center of the delay alignment tap. The delay alignment tap has a full range of ±4 ns and a half range of ±2 ns. This hard reset can be used to initiate the GTX transceiver to perform the TX phase and delay alignment automatically when all other TX buffer bypass input ports are set Low. The user is recommended to use TXDLYSRESET only for phase and delay alignment.
TXPHALIGN	In	Async	Sets the TX phase alignment. Tied Low when using the auto alignment mode.
TXPHALIGNEN	In	Async	Enables the TX phase alignment in manual mode. Tied Low when using the auto mode.
TXPHDLYPD	In	Async	TX phase and delay alignment circuit power down. Tied High when a) TX buffer bypass is not in use; b) TXPD is asserted, or c) TXOUTCLKSEL is set to `3'b011` or `3'b100` but the reference clock is not connected. Tied Low during TX buffer bypass mode normal operation. 0: Power-up the TX phase and delay alignment circuit. 1: Power-down the TX phase and delay alignment circuit.
TXPHINIT	In	Async	TX phase alignment initialization. Reserved. Tied Low when using the auto alignment mode.
TXPHOVRDEN	In	Async	TX phase alignment counter override enable. Tied Low when not in use. 0: Normal operation. 1: Enables TX phase alignment counter override with the value from TXPH_CFG[10:0].

Table 3-17: **TX Buffer Bypass Ports *(Cont'd)***

Port	Dir	Clock Domain	Description
TXDLYSRESET	In	Async	TX delay alignment soft reset to gradually shift TXUSRCLK to the center of the delay alignment tap. The delay alignment tap has a full range of ±4 ns and a half range of ±2 ns. This soft reset can be used to initiate the GTX transceiver to perform the TX phase and delay alignment automatically when all other TX buffer bypass input ports are set Low.
TXDLYBYPASS	In	Async	TX delay alignment bypass. 0: Uses the TX delay alignment circuit. 1: Bypasses the TX delay alignment circuit.
TXDLYEN	In	Async	Enables the TX delay alignment in manual mode. Tied Low when using the auto mode.
TXDLYOVRDEN	In	Async	TX delay alignment counter override enable. Tied Low when not in use. 0: Normal operation. 1: Enables TX delay alignment counter override with the value from TXDLY_CFG[14:6].
TXPHDLYTSTCLK	In	Async	TX phase and delay alignment test clock. Used with TXDLYHOLD and TXDLYUPDOWN.
TXDLYHOLD	In	TXPHDLYTSTCLK	TX delay alignment hold. Used as a hold override when TXPHDLY_CFG[1] = 1 to bypass the TX phase and delay alignment voter. Tied Low when not in use.
TXDLYUPDOWN	In	TXPHDLYTSTCLK	TX delay alignment up or down. Used as an up or down override when TXPHDLY_CFG[1] = 1 to bypass the TX phase and delay alignment voter. Tied Low when not in use.
TXPHALIGNDONE	Out	Async	TX phase alignment done. When the auto TX phase and delay alignment is used, the second rising edge of TXPHALIGNDONE detected after TXDLYSRESETDONE assertion indicates TX phase and delay alignment are done.
TXPHINITDONE	Out	Async	Indicates that TX phase alignment initialization is done.

Table 3-17: **TX Buffer Bypass Ports *(Cont'd)***

Port	Dir	Clock Domain	Description
TXDLYSRESETDONE	Out	Async	Indicates that TX delay alignment soft reset is done.
TXSYNCMODE	In	Async	Reserved. Tie to GND.
TXSYNCALLIN	In	Async	Reserved. Tie to GND.
TXSYNCIN	In	Async	Reserved. Tie to GND.
TXSYNCOUT	Out	Async	Reserved.
TXSYNCDONE	Out	Async	Reserved.

Table 3-18: **TX Buffer Bypass Attributes**

Attribute	Type	Description
TXBUF_EN	Boolean	Use or bypass the TX buffer. TRUE: Uses the TX buffer (default). FALSE: Bypasses the TX buffer (advanced feature).
TX_XCLK_SEL	String	Selects the clock source used to drive the PMA parallel clock domain (XCLK). TXOUT: Selects TXOUTCLK as the source of XCLK. Used when using the TX buffer. TXUSR: Selects TXUSRCLK as the source of XCLK. Used when bypassing the TX buffer.
TXPH_CFG	16-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
TXPH_MONITOR_SEL	5-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
TXPHDLY_CFG	24-bit Binary	TX phase and delay alignment configuration. TXPHDLY_CFG[19] = `1` is used to set the TX delay alignment tap to the full range of ±4 ns. TXPHDLY_CFG[19] = `0` is used to set the TX delay alignment tap to the half range of ±2 ns. Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
TXDLY_CFG	16-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
TXDLY_LCFG	9-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.

Table 3-18: **TX Buffer Bypass Attributes *(Cont'd)***

Attribute	Type	Description
TXDLY_TAP_CFG	16-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
TXSYNC_MULTILANE	1-bit Binary	Reserved. Tie to `1'b0`.
TXSYNC_SKIP_DA	1-bit Binary	Reserved. Tie to `1'b0`.
TXSYNC_OVRD	1-bit Binary	Reserved. Tie to `1'b1`.
LOOPBACK_CFG	1-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.

TX Buffer Bypass Use Modes

TX phase alignment can be performed on one channel (single lane) or a group of channels sharing a single TXOUTCLK (multi-lane). For GTX transceivers, TX buffer bypass supports single-lane auto mode and multi-lane manual mode. For GTH transceivers, TX buffer bypass supports manual mode for both single-lane and multi-lane applications (see Table 3-19).

Table 3-19: **TX Buffer Bypass Use Modes**

TX Buffer Bypass	GTX Transceivers	GTH Transceivers
Single Lane	Auto	Manual
Multi-lane	Manual[(1)]	Manual[(1)]

Notes:

1. Multi-lane TX buffer bypass support crossing the SLR boundary in stacked silicon interconnect (SSI) technology devices is an advanced feature and is not recommended for normal operation. This feature can be guaranteed only under certain system-level conditions and data rates.

Using TX Buffer Bypass in Single-Lane Auto Mode (GTX Transceiver Only)

These GTX transceiver settings should be used to bypass the TX buffer:

- TXBUF_EN = FALSE.
- TX_XCLK_SEL = TXUSR.
- TXOUTCLKSEL = `011b` or `100b` to select the GTX transceiver reference clock as the source of TXOUTCLK.

With the GTX transceiver reference clock selected, TXOUTCLK is used as the source of the TXUSRCLK. The user must ensure that TXOUTCLK and the selected GTX transceiver reference clock are operating at the desired frequency. When the TX buffer is bypassed, the TX phase alignment procedure must be performed after these conditions:

- Resetting or powering up the GTX transceiver TX.
- Resetting or powering up the CPLL and/or QPLL.
- Change of the GTX transceiver reference clock source or frequency.
- Change of the TX line rate.

Figure 3-19 shows the required steps to perform the auto TX phase alignment and use the TX delay alignment to adjust TXUSRCLK to compensate for temperature and voltage variations.

Figure 3-19: **TX Buffer Bypass—Single-Lane Auto Mode (GTX Transceiver Only)**

Notes relevant to Figure 3-19:

1. The sequence of events in Figure 3-19 is not drawn to scale.
2. After conditions such as a GTX transmitter reset or TX rate change, TX phase alignment must be performed to align XCLK and TXUSRCLK. The TX phase and delay alignments are initiated by asserting TXDLYSRESET. The assertion of TXDLYSRESET should be less than 50 ns.
3. Wait until TXDLYSRESETDONE is High. TXDLYSRESETDONE will remain asserted for a minimum of 100 ns.
4. TX phase alignment is done when the second rising edge of TXPHALIGNDONE is detected. The first assertion of TXPHALIGNDONE will have a minimum pulse width of 100 ns. Upon the second rising edge of TXPHALIGNDONE, this signal should remain asserted until another alignment procedure is initiated.
5. An assertion/deassertion of GTTXRESET is required if TXPHALIGNDONE does not follow the sequence shown in Figure 3-19.
6. TX delay alignment continues to adjust TXUSRCLK to compensate for temperature and voltage variations.

Using TX Buffer Bypass in Single-Lane Manual Mode (GTH Transceiver Only)

These GTH transceiver settings should be used to bypass the TX buffer:

- TXBUF_EN = FALSE.
- TX_XCLK_SEL = TXUSR.
- TXOUTCLKSEL = `011b` or `100b` to select the GTH transceiver reference clock as the source of TXOUTCLK.

With the GTH transceiver reference clock selected, TXOUTCLK is used as the source of the TXUSRCLK. The user must ensure that TXOUTCLK and the selected GTH transceiver reference clock are operating at the desired frequency. When the TX buffer is bypassed, the TX phase alignment procedure must be performed after these conditions:

- Resetting or powering up the GTH transceiver TX.
- Resetting or powering up the CPLL and/or QPLL.
- Change of the GTH transceiver reference clock source or frequency.
- Change of the TX line rate.

Figure 3-20 shows the required steps to perform the manual TX phase alignment and use the TX delay alignment to adjust TXUSRCLK to compensate for temperature and voltage variations.

Figure 3-20: **TX Buffer Bypass Example—Single-Lane Manual Mode (GTH Transceiver Only)**

Notes relevant to Figure 3-20:

1. The sequence of events in Figure 3-20 is not drawn to scale.
2. Set the TXSYNC_OVRD attribute to 1'b1.
3. Set TXPHDLYRESET and TXDLYBYPASS to Low for all lanes.
4. Set TXPHALIGNEN to High.
5. Assert TXDLYSRESET. Hold this signal High until TXDLYSRESETDONE is asserted.
6. Deassert TXDLYSRESET after TXDLYSRESETDONE is asserted.
7. When TXDLYSRESET is deasserted, assert TXPHINIT. Hold this signal High until the rising edge of TXPHINITDONE is observed.
8. Deassert TXPHINIT.
9. Assert TXPHALIGN. Hold this signal High until the rising edge of TXPHALIGNDONE is observed.
10. Deassert TXPHALIGN.
11. Assert TXDLYEN. This causes TXPHALIGNDONE to be deasserted.
12. Hold TXDLYEN until the rising edge of TXPHALIGNDONE is observed.
13. TX delay alignment continues to adjust TXUSRCLK to compensate for temperature and voltage variations.

Using the TX Phase Alignment to Minimize the TX Lane-to-Lane Skew

The TX phase alignment circuit can also be used to minimize skew between GTX/GTH transceivers. Figure 3-21 shows how the TX phase alignment circuit can reduce lane skew by aligning the XCLK domains of multiple GTX/GTH transceivers to a common clock source. Figure 3-21 shows multiple GTX/GTH transceiver lanes running before and after TX phase is aligned to a common clock. Before the TX phase alignment, all XCLKs have an

arbitrary phase difference. After TX phase alignment, the only phase difference is the skew from the common clock, and all lanes transmit data simultaneously as long as the datapath latency is matched. TXUSRCLK and TXUSRCLK2 for all GTX/GTH transceivers must come from the same source and must be routed through a low skew clocking resource such as a BUFG for the TX phase alignment circuit to be effective.

Figure 3-21: **TX Phase Alignment to Minimize TX Lane-to-Lane Skew**

Using TX Buffer Bypass in Multi-Lane Manual Mode (GTX and GTH Transceivers)

Multi-lane TX buffer bypass support crossing the SLR boundary in SSIT devices is an advanced feature and is not recommended for normal operation. This feature can be guaranteed only under certain system-level conditions and data rates.

For GTX and GTH transceivers, when a multi-lane application requires TX buffer bypass, phase alignment needs to be performed manually. This section describes the steps required to perform the multi-lane TX buffer bypass alignment procedure manually.

- Master: In a multi-lane application, the buffer bypass master is the lane that is the source of TXOUTCLK.
- Slave: All the lanes that share the same TXUSRCLK/TXUSRCLK2, which is generated from the TXOUTCLK of the buffer bypass master.

Figure 3-22 shows an example of buffer bypass master versus slave lanes.

Figure 3-22: **Example of Buffer Bypass Master versus Slave Lanes**

The following GTX/GTH transceiver settings are used to bypass the TX buffer:

- TXBUF_EN = FALSE.
- TX_XCLK_SEL = TXUSR.
- TXOUTCLKSEL = `3'b011` or `3'b100` to select the GTX/GTH transceiver reference clock as the source of TXOUTCLK.

With the GTX/GTH transceiver reference clock selected, TXOUTCLK is used as the source of the TXUSRCLK. The user must ensure that TXOUTCLK and the selected GTX/GTH transceiver reference clock is running and operating at the desired frequency. When the TX buffer is bypassed, the TX phase alignment procedure must be performed after these conditions:

- Resetting or powering up the GTX/GTH transmitter.
- Resetting or powering up the CPLL, QPLL, or both.
- Change of the GTX/GTH transceiver reference clock source or frequency.
- Change of the TX line rate.

Figure 3-23 shows the required steps to perform manual TX phase and delay alignment.

Figure 3-23: **TX Phase and Delay Alignment in Manual Mode**

Notes relevant to Figure 3-23:

1. The sequence of events shown in Figure 3-23 is not drawn to scale.
2. M_* denotes ports related to the master lane.
3. S_* denotes ports related to the slave lane(s).
4. GTX transceiver: Set the PCS_RSVD_ATTR[1] attribute to `1'b1`.

   GTH transceiver: Set the TXSYNC_OVRD attribute to `1'b1`.
5. Set TXPHDLYRESET and TXDLYBYPASS to Low for all lanes.
6. Set TXPHALIGNEN to High for all lanes.
7. Assert TXDLYSRESET for all lanes. Hold this signal High until TXDLYSRESETDONE of the respective lane is asserted.
8. Deassert TXDLYSRESET for the lane in which the TXDLYSRESETDONE is asserted.
9. When TXDLYSRESET of all lanes are deasserted, assert TXPHINIT for the master lane. Hold this signal High until the rising edge of TXPHINITDONE of the master lane is observed.
10. Deassert TXPHINIT for the master lane.
11. Assert TXPHALIGN for the master lane. Hold this signal High until the rising edge of TXPHALIGNDONE of the master lane is observed.
12. Deassert TXPHALIGN for the master lane.

13. Assert TXDLYEN for the master lane. This causes TXPHALIGNDONE to be deasserted.
14. Hold TXDLYEN for the master lane High until the rising edge of TXPHALIGNDONE of the master lane is observed.
15. Deassert TXDLYEN for the master lane.
16. Assert TXPHINIT for all slave lane(s). Hold this signal High until the rising edge of TXPHINITDONE of the respective slave lane is observed.
17. Deassert TXPHINIT for the slave lane in which the TXPHINITDONE is asserted.
18. When TXPHINIT for all slave lane(s) are deasserted, assert TXPHALIGN for all slave lane(s). Hold this signal High until the rising edge of TXPHALIGNDONE of the respective slave lane is observed.
19. Deassert TXPHALIGN for the slave lane in which the TXPHALIGNDONE is asserted.
20. When TXPHALIGN for all slave lane(s) are deasserted, assert TXDLYEN for the master lane. This causes TXPHALIGNDONE of the master lane to be deasserted.
21. Wait until TXPHALIGNDONE of the master lane reasserts. Phase and delay alignment for the multi-lane interface is complete. Continue to hold TXDLYEN for the master lane High to adjust TXUSRCLK to compensate for temperature and voltage variations.

TX Pattern Generator

Functional Description

Pseudo-random bit sequences (PRBS) are commonly used to test the signal integrity of high-speed links. These sequences appear random but have specific properties that can be used to measure the quality of a link. The GTX/GTH transceiver pattern generator block can generate several industry-standard PRBS patterns listed in Table 3-20.

Table 3-20: **Supported PRBS Patterns**

Name	Polynomial	Length of Sequence	Description
PRBS-7	$1 + X^6 + X^7$	2^7 - 1 bits	Used to test channels with 8B/10B.
PRBS-15	$1 + X^{14} + X^{15}$	2^{15} - 1 bits	ITU-T Recommendation O.150, Section 5.3. PRBS-15 is often used for jitter measurement because it is the longest pattern the Agilent DCA-J sampling scope can handle.
PRBS-23	$1 + X^{18} + X^{23}$	2^{23} - 1 bits	ITU-T Recommendation O.150, Section 5.6. PRBS-23 is often used for non-8B/10B encoding schemes. It is one of the recommended test patterns in the SONET specification.
PRBS-31	$1 + X^{28} + X^{31}$	2^{31} - 1 bits	ITU-T Recommendation O.150, Section 5.8. PRBS-31 is often used for non-8B/10B encoding schemes. It is a recommended PRBS test pattern for 10 Gigabit Ethernet. See IEEE 802.3ae-2002.

In addition to PRBS patterns, the GTX/GTH transceiver supports 16-UI, 20-UI, 32-UI, or 40-UI square wave test patterns, depending on data width as well as a 2-UI square wave

test pattern and PCI Express compliance pattern generation. Clocking patterns are usually used to check PLL random jitter often done with a spectrum analyzer.

Table 3-21: **PCI Express Compliance Pattern**

Symbol	K28.5	D21.5	K28.5	D10.2
Disparity	0	1	1	0
Pattern	0011111010	1010101010	1100000101	0101010101

Figure 3-24: **20-UI Square Wave**

The error insertion function is supported to verify link connection and also for jitter tolerance tests. When an inverted PRBS pattern is necessary, TXPOLARITY signal is used to control polarity.

Figure 3-25: **TX Pattern Generator Block**

Ports and Attributes

Table 3-22 defines the pattern generator ports.

Table 3-22: **Pattern Generator Ports**

Port Name	Dir	Clock Domain	Description
TXPRBSSEL[2:0]	In	TXUSRCLK2	Transmitter PRBS generator test pattern control. `000`: Standard operation mode (test pattern generation is off) `001`: PRBS-7 `010`: PRBS-15 `011`: PRBS-23 `100`: PRBS-31 `101`: PCI Express compliance pattern. Only works with 20-bit and 40-bit modes `110`: Square wave with 2 UI (alternating 0s/1s) `111`: Square wave with 16 UI, 20 UI, 32 UI, or 40 UI period (based on data width)
TXPRBSFORCEERR	In	TXUSRCLK2	When this port is driven High, errors are forced in the PRBS transmitter. While this port is asserted, the output data pattern contains errors. When TXPRBSSEL is set to `000`, this port does not affect TXDATA.

Table 3-23 defines the pattern generator attribute.

Table 3-23: **Pattern Generator Attribute**

Attribute	Type	Description
RXPRBS_ERR_LOOPBACK	1-bit Binary	When set to 1, causes RXPRBSERR bit to be internally looped back to TXPRBSFORCEERR of the same GTX/GTH transceiver. This allows synchronous and asynchronous jitter tolerance testing without worrying about data clock domain crossing. When set to 0, TXPRBSFORCEERR forces onto the TX PRBS.

Use Models

The pattern generation and check function are usually used for verifying link quality tests and also for jitter tolerance tests. For link quality testing, the test pattern is chosen by setting TXPRBSSEL and RXPRBSSEL to a non-000 value, and RXPRBS_ERR_LOOPBACK is set to 0 (Figure 3-26). Only the PRBS pattern is recognized by the RX pattern checker.

Figure 3-26: **Link Test Mode with a PRBS-7 Pattern**

To calculate accurately the receiver's bit error rate (BER), an external jitter tolerance tester should be used. For the test, the GTX/GTH transceiver should loop the received error status back through the transmitter by setting RXPRBS_ERR_LOOPBACK to 1 (Figure 3-27). The same setting should be applied to RXPRBSSEL and TXPRBSSEL.

Figure 3-27: **Jitter Tolerance Test Mode with a PRBS-7 Pattern**

TX Polarity Control

Functional Description

If TXP and TXN differential traces are accidentally swapped on the PCB, the differential data transmitted by the GTX/GTH transceiver TX is reversed. One solution is to invert the parallel data before serialization and transmission to offset the reversed polarity on the differential pair. The TX polarity control can be accessed through the TXPOLARITY input from the fabric user interface. It is driven High to invert the polarity of outgoing data.

Ports and Attributes

Table 3-24 defines the ports required for TX polarity control.

Table 3-24: **TX Polarity Control Ports**

Port	Dir	Clock Domain	Description
TXPOLARITY	In	TXUSRCLK2	The TXPOLARITY port is used to invert the polarity of outgoing data. 0: Not inverted. TXP is positive, and TXN is negative. 1: Inverted. TXP is negative, and TXN is positive.

Using TX Polarity Control

TXPOLARITY can be tied High if the polarity of TXP and TXN needs to be reversed.

TX Fabric Clock Output Control

Functional Description

The TX Clock Divider Control block has two main components: serial clock divider control and parallel clock divider and selector control. The clock divider and selector details are illustrated in Figure 3-28.

Figure 3-28: **TX Serial and Parallel Clock Divider**

Notes relevant to Figure 3-28:

1. TXOUTCLKPCS and TXOUTCLKFABRIC are redundant outputs. Use TXOUTCLK for new designs.
2. The REFCLK_CTRL option is controlled automatically by software and is not user selectable. The user can only route one of the IBUFDS_GTE2's O or ODIV2 outputs to the FPGA logic via the CMT (PLL, MMCM or BUFMRCE), BUFH, or BUFG.
3. IBUFDS_GTE2 is a redundant output for additional clocking scheme flexibility.
4. There is only one CPLL in the GTXE2_CHANNEL/GTHE2_CHANNEL. The QPLL from the GTXE2_COMMON/GTHE2_COMMON can also be used, when applicable.
5. The selection of the /2 or /4 divider block is controlled by the TX_INT_DATAWIDTH attribute from the GTXE2_CHANNEL/GTHE2_CHANNEL primitive. /2 is selected when TX_INT_DATAWIDTH = 0 (2-byte internal datapath) and /4 is selected when TX_INT_DATAWIDTH = 1 (4-byte internal datapath).
6. The selection of the /4 or /5 divider block is controlled by the TX_DATA_WIDTH attribute from the GTXE2_CHANNEL/GTHE2_CHANNEL primitive. /4 is selected when TX_DATA_WIDTH = 16, 32, or 64. /5 is selected when TX_DATA_WIDTH = 20, 40, or 80.

7. For details about placement constraints and restrictions on clocking resources (MMCM, BUFGCTRL, IBUFDS_GTE2, BUFG, etc.), refer to the UG472, *7 Series FPGAs Clocking Resources User Guide*.

Serial Clock Divider

Each transmitter PMA module has a D divider that divides down the clock from the PLL for lower line rate support. This serial clock divider, D, can be set statically for applications with a fixed line rate or it can be changed dynamically for protocols with multiple line rates.

To use the D divider in fixed line rate applications, the TXOUT_DIV attribute must be set to the appropriate value, and the TXRATE port needs to be tied to `3'b000`. Refer to the Static Setting via Attribute column in Table 3-25 for details.

To use the D divider in multiple line rate applications, the TXRATE port is used to dynamically select the D divider value. The TXOUT_DIV attribute and the TXRATE port must select the same D divider value upon device configuration. After device configuration, the TXRATE is used to dynamically change the D divider value. Refer to the Dynamic Control via Ports column in Table 3-25 for details.

The control for the serial divider is shown in Table 3-25. For details about the line rate range per speed grade, refer to the 7 series FPGAs documentation page for the appropriate data sheet.

Table 3-25: **TX PLL Output Divider Setting**

D Divider Value	Static Setting via Attribute	Dynamic Control via Ports
1	TXOUT_DIV = 1 TXRATE = `3'b000`	TXOUT_DIV = Ignored TXRATE = `3'b001`
2	TXOUT_DIV = 2 TXRATE = `3'b000`	TXOUT_DIV = Ignored TXRATE = `3'b010`
4	TXOUT_DIV = 4 TXRATE = `3'b000`	TXOUT_DIV = Ignored TXRATE = `3'b011`
8	TXOUT_DIV = 8 TXRATE = `3'b000`	TXOUT_DIV = Ignored TXRATE = `3'b100`
16	TXOUT_DIV = 16 TXRATE = `3'b000`	TXOUT_DIV = Ignored TXRATE = `3'b101`

Parallel Clock Divider and Selector

The parallel clock outputs from the TX clock divider control block can be used as a fabric logic clock, depending on the line rate requirement.

The recommended clock for the fabric is the TXOUTCLK from one of the GTX/GTH transceivers. It is also possible to bring the MGTREFCLK directly to the FPGA logic and use as the fabric clock. TXOUTCLK is preferred for general applications as it has an output delay control used for applications that bypass the TX buffer for output lane deskewing or constant datapath delay. Refer to TX Buffer Bypass, page 127 for more details.

The TXOUTCLKSEL port controls the input selector and allows these clocks to be output via the TXOUTCLK port:

- TXOUTCLKSEL = `3'b001`: The TXOUTCLKPCS path is not recommended for use because it incurs extra delay from the PCS block.
- TXOUTCLKSEL = `3'b010`: TXOUTCLKPMA is the divided down PLL clock after the TX phase interpolator and is used by the TX PCS block. This clock is interrupted when the PLL is reset by one of the related reset signals.
- TXOUTCLKSEL = `3'b011` or `3'b100`: TXPLLREFCLK_DIV1 or TXPLLREFCLK_DIV2 is the input reference clock to the CPLL or QPLL, depending on the TXSYSCLKSEL[1] setting. TXPLLREFCLK is the recommended clock for general usage and is required for the TX buffer bypass mode.

Ports and Attributes

Table 3-26 defines the ports required for TX fabric clock output control.

Table 3-26: **TX Fabric Clock Output Control Ports**

Port	Dir	Clock Domain	Description
TXOUTCLKSEL[2:0]	In	Async	This port controls the multiplexer select signal in Figure 3-28. `3'b000`: Static 1 `3'b001`: TXOUTCLKPCS path `3'b010`: TXOUTCLKPMA path `3'b011`: TXPLLREFCLK_DIV1 path `3'b100`: TXPLLREFCLK_DIV2 path Others: Reserved.
TXRATE[2:0]	In	TXUSRCLK2	This port dynamically controls the setting for the TX serial clock divider D (see Table 3-25), and it is used with the TXOUT_DIV attribute. `3'b000`: Use the TXOUT_DIV divider value `3'b001`: Set the D divider to 1 `3'b010`: Set the D divider to 2 `3'b011`: Set the D divider to 4 `3'b100`: Set the D divider to 8 `3'b101`: Set the D divider to 16
TXOUTCLKFABRIC	Out	Clock	TXOUTCLKFABRIC is a redundant output reserved for testing. TXOUTCLK with TXOUTCLKSEL = `3'b011` should be used instead.
TXOUTCLK	Out	Clock	TXOUTCLK is the recommended clock output to the FPGA logic. The TXOUTCLKSEL port is the input selector for TXOUTCLK and allows the PLL input reference clock to the FPGA logic.
TXOUTCLKPCS	Out	Clock	TXOUTCLKPCS is a redundant output. TXOUTCLK with TXOUTCLKSEL = `3'b001` should be used instead.

Table 3-26: **TX Fabric Clock Output Control Ports *(Cont'd)***

Port	Dir	Clock Domain	Description
TXRATEDONE	Out	TXUSRCLK2	The TXRATEDONE port is asserted High for one TXUSRCLK2 cycle in response to a change on the TXRATE port. The TRANS_TIME_RATE attribute defines the period of time between a change on the TXRATE port and the assertion of TXRATEDONE.
TXDLYBYPASS	In	Async	TX delay alignment bypass: 0: Uses the TX delay alignment circuit. Set to `1'b0` when the TX buffer is bypassed. 1: Bypasses the TX delay alignment circuit. Set to `1'b1` when the TX buffer is used.

Table 3-27 defines the attributes required for TX fabric clock output control.

Table 3-27: **TX Fabric Clock Output Control Attributes**

Attribute	Type	Description
TRANS_TIME_RATE	8-bit Hex	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used. This attribute determines when PHYSTATUS and TXRATEDONE are asserted after a rate change.
TXBUF_RESET_ON_RATE_CHANGE	Boolean	When set to TRUE, this attribute enables an automatic TX buffer reset during a rate change event initiated by a change in TXRATE.
TXOUT_DIV	Integer	This attribute controls the setting for the TX serial clock divider. This attribute is only valid when TXRATE = `3'b000`. Otherwise the D divider value is controlled by TXRATE. Valid settings are 1, 2, 4, 8, and 16.

TX Phase Interpolator PPM Controller

Functional Description

The TX Phase Interpolator Parts Per Million (TXPIPPM) Controller module provides support for dynamically controlling the TX phase interpolator (TX PI). Located in the TX PCS, its inputs come from the FPGA TX interface and it outputs to the TX PMA. Applications exist that require fine-tune control of the data in the TX PMA. Control of the output clock from the PLL is achieved through a TX PI, which in turn can be controlled by the TX phase interpolator PPM controller module. The FPGA logic can control the TX PI in the TX PMA through the use of the TX Phase Interpolator PPM Controller module in the PCS.

Ports and Attributes

Table 3-28 defines the ports required for the TX Phase Interpolator PPM Controller. These ports and attributes are relevant for GTH transceivers only.

Table 3-28: **TX Phase Interpolator PPM Controller Ports**

Port	Dir	Clock Domain	Description
TXPIPPMEN	In	TXUSRCLK2	`1'b0`: Disables the TX Phase Interpolator PPM Controller block. The TX PI is not updated with a PI code and retains the previous PI code. `1'b1`: Enables the TX Phase Interpolator PPM Controller block. The TX PI is updated with a PI code every TXPI_SYNFREQ_PPM[2:0] cycles.
TXPIPPMOVRDEN	In	TXUSRCLK2	1'b0: Normal operation 1'b1: Enables direct control of the PI Code output to the TX PI in the TX PMA. Use with TXPPMOVRD_VALUE[6:0] to program the value of PI code.
TXPIPPMSEL	In	TXUSRCLK2	Reserved. This should always be tied to `1'b1`.
TXPIPPMPD	In	Async	`1'b0`: Does not power down the TX Phase Interpolator PPM Controller module. `1'b1`: Powers down the TX Phase Interpolator PPM Controller module.
TXPIPPMSTEPSIZE[4:0]	In	TXUSRCLK2	TXPIPPMSTEPSIZE[4]: `1'b1`: Increments PI Code. `1'b0`: Decrements PI Code. TXPIPPMSTEPSIZE[3:0] is the amount to increment or decrement PI code. Its values range from 0 to 15.

Table 3-29 defines the TX Phase Interpolator PPM Controller attributes.

Table 3-29: **TX Phase Interpolator PPM Controller Attributes**

Attribute	Type	Description
TXPI_SYNFREQ_PPM[2:0]	3-bit Binary	This attribute specifies how often PI Code to the TX PI is updated. It is updated every (TXPI_SYNFREQ_PPM[2:0] + 1) cycles. All values are valid except for `3'b000`. The 7 Series FPGAs Transceivers Wizard's default value should be used for this attribute.
TXPI_PPM_CFG[7:0]	8-bit Binary	When TXPIPPMOVRDEN = `1'b1`, the lower 7 bits of this attribute should be programmed to one of the 128 values output to the TX PI. The most significant bit needs to be pulsed (asserted High and then Low) for the TX PI to register the new 7-bit value of TXPI_PPM_CFG[6:0].
TXPI_INVSTROBE_SEL	1-bit Binary	Reserved. Tied to `1'b0`.
TXPI_GREY_SEL	1-bit Binary	`1'b0`: TXPIPPMSTEPSIZE[3:0] is binary encoded. `1'b1`: TXPIPPMSTEPSIZE[3:0] is grey encoded.
TXPI_PPMCLK_SEL	String	Reserved. The 7 Series FPGAs Transceivers Wizard's default value should be used for this attribute.

TX Phase Interpolator PPM Controller Use Mode

The following describes a sample use case:

1. A frequency counter in the fabric determines the lead/lag relationship between the two clocks of interest and increments or decrements (TXPIPPMSTEPSIZE[4]) the PI Code by a certain step size (TXPIPPMSTEPSIZE[3:0]).
2. A sampler and lock detect circuit in the fabric determines when the two clocks are phase aligned. When not phase aligned, the user asserts a signal to the TX Phase Interpolator PPM Controller with the desirable PI Code.

This continual phase shifting (fine-tuning) occurs when the lock detect circuit deems the two clocks out of phase and enables the TX Phase Interpolator PPM Controller.

TX Configurable Driver

Functional Description

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

- Differential voltage control
- Pre-cursor and post-cursor transmit pre-emphasis
- Calibrated termination resistors

Figure 3-29: **TX Configurable Driver Block Diagram**

Ports and Attributes

Table 3-30 defines the TX configurable driver ports.

Table 3-30: **TX Configurable Driver Ports**

Port	Dir	Clock Domain	Description
TXBUFDIFFCTRL[2:0]	In	TXUSRCLK2	Pre-driver Swing Control. The default is `3'b100` (nominal value). Do *not* modify this value.
TXDEEMPH	In	TXUSRCLK2	TX de-emphasis control for PCI Express PIPE 2.0 interface. This signal is mapped internally to TXPOSTCURSOR via attributes. 0: 6.0 dB de-emphasis (TX_DEEMPH_0[4:0] attribute) 1: 3.5 dB de-emphasis (TX_DEEMPH_1[4:0] attribute)

Table 3-30: **TX Configurable Driver Ports *(Cont'd)***

Port	Dir	Clock Domain	Description
TXDIFFCTRL[3:0]	In	TXUSRCLK2	Driver Swing Control. The default is user specified. All listed values are in mV_{PPD}. [3:0] / mV_{PPD} 4'b0000 / 0.269 4'b0001 / 0.336 4'b0010 / 0.407 4'b0011 / 0.474 4'b0100 / 0.543 4'b0101 / 0.609 4'b0110 / 0.677 4'b0111 / 0.741 4'b1000 / 0.807 4'b1001 / 0.866 4'b1010 / 0.924 4'b1011 / 0.973 4'b1100 / 1.018 4'b1101 / 1.056 4'b1110 / 1.092 4'b1111 / 1.119 **Note:** The peak-to-peak differential voltage is defined when TXPOSTCURSOR = 5'b00000 and TXPRECURSOR = 5'b00000.
TXELECIDLE	In	TXUSRCLK2	When High, this signal forces MGTXTXP/MGTHTXP and MGTXTXN/MGTHTXN both to Common mode, creating an electrical idle signal.
TXINHIBIT	In	TXUSRCLK2	When High, this signal blocks transmission of TXDATA and forces MGTXTXP/MGTHTXP to 0 and MGTXTXN/MGTHTXN to 1.
TXMAINCURSOR[6:0]	In	Async	Allows the main cursor coefficients to be directly set if the TX_MAINCURSOR_SEL attribute is set to 1'b1. 51 – TXPOSTCURSOR coefficient units – TXPRECURSOR coefficient units $\leq$ TXMAINCURSOR coefficient units $\leq$ 80 –TXPOSTCURSOR coefficient units – TXPRECURSOR coefficient units.

Table 3-30: **TX Configurable Driver Ports *(Cont'd)***

<table>
<tr><th>Port</th><th>Dir</th><th>Clock Domain</th><th>Description</th></tr>
<tr><td>TXMARGIN[2:0]</td><td>In</td><td>Async</td><td>TX Margin control for PCI Express PIPE 3.0 Interface. These signals are mapped internally to TXDIFFCTRL/TXBUFDIFFCTRL via attributes.
<table>
<tr><th>[2:0]</th><th>Full Range</th><th>Half Range</th><th>Full Range Attribute</th><th>Half Range Attribute</th></tr>
<tr><td>000</td><td>800-1200</td><td>400-1200</td><td>TX_MARGIN_FULL_0</td><td>TX_MARGIN_LOW_0</td></tr>
<tr><td>001</td><td>800-1200</td><td>400-700</td><td>TX_MARGIN_FULL_1</td><td>TX_MARGIN_LOW_1</td></tr>
<tr><td>010</td><td>800-1200</td><td>400-700</td><td>TX_MARGIN_FULL_2</td><td>TX_MARGIN_LOW_2</td></tr>
<tr><td>011</td><td>200-400</td><td>100-200</td><td>TX_MARGIN_FULL_3</td><td>TX_MARGIN_LOW_3</td></tr>
<tr><td>100</td><td>100-200</td><td>100-200</td><td>TX_MARGIN_FULL_4</td><td>TX_MARGIN_LOW_4</td></tr>
<tr><td>101</td><td colspan="4" rowspan="3">default to “DIRECT” mode</td></tr>
<tr><td>110</td></tr>
<tr><td>111</td></tr>
</table></td></tr>
<tr><td>TXQPIBIASEN</td><td>In</td><td>Async</td><td>Enables the GND bias on the TX output as required by the QPI specification.</td></tr>
<tr><td>TXQPISENN</td><td>Out</td><td>Async</td><td>Sense output that registers a 1 or 0 on the MGTXTXN/MGTHTXN pin.</td></tr>
<tr><td>TXQPISENP</td><td>Out</td><td>Async</td><td>Sense output that registers a 1 or 0 on the MGTXTXP/MGTHTXP pin.</td></tr>
<tr><td>TXQPISTRONGPDOWN</td><td>In</td><td>Async</td><td>Pulls the TX output strongly to GND to enable handshaking as required by the QPI protocol.</td></tr>
<tr><td>TXQPIWEAKPUP</td><td>In</td><td>Async</td><td>Pulls the TX output weakly to MGTAVTT to enable handshaking as required by the QPI protocol.</td></tr>
</table>

Table 3-30: **TX Configurable Driver Ports *(Cont'd)***

Port	Dir	Clock Domain	Description
TXPOSTCURSOR[4:0]	In	Async	Transmitter post-cursor TX pre-emphasis control. The default is user specified. All listed values (dB) are typical.

[4:0]	Emphasis (dB)	|Coefficient Units|
5'b00000	0.00	0
5'b00001	0.22	1
5'b00010	0.45	2
5'b00011	0.68	3
5'b00100	0.92	4
5'b00101	1.16	5
5'b00110	1.41	6
5'b00111	1.67	7
5'b01000	1.94	8
5'b01001	2.21	9
5'b01010	2.50	10
5'b01011	2.79	11
5'b01100	3.10	12
5'b01101	3.41	13
5'b01110	3.74	14
5'b01111	4.08	15
5'b10000	4.44	16
5'b10001	4.81	17
5'b10010	5.19	18
5'b10011	5.60	19
5'b10100	6.02	20
5'b10101	6.47	21
5'b10110	6.94	22
5'b10111	7.43	23
5'b11000	7.96	24
5'b11001	8.52	25
5'b11010	9.12	26
5'b11011	9.76	27
5'b11100	10.46	28
5'b11101	11.21	29
5'b11110	12.04	30
5'b11111	12.96	31

Note: The TXPOSTCURSOR values are defined when the TXPRECURSOR =5'b00000
Emphasis = $20\log10(V_{high}/V_{low}) = |20\log10\ (V_{low}/V_{high})|$

TXPOSTCURSORINV	In	Async	When set to 1'b1, inverts the polarity of the TXPOSTCURSOR coefficient. The default is 1'b0.

Table 3-30: **TX Configurable Driver Ports *(Cont'd)***

Port	Dir	Clock Domain	Description
TXPRECURSOR[4:0]	In	Async	Transmitter pre-cursor TX pre-emphasis control. The default is user specified. All listed values (dB) are typical.

[4:0]	Emphasis (dB)	|Coefficient Units|
5'b00000	0.00	0
5'b00001	0.22	1
5'b00010	0.45	2
5'b00011	0.68	3
5'b00100	0.92	4
5'b00101	1.16	5
5'b00110	1.41	6
5'b00111	1.67	7
5'b01000	1.94	8
5'b01001	2.21	9
5'b01010	2.50	10
5'b01011	2.79	11
5'b01100	3.10	12
5'b01101	3.41	13
5'b01110	3.74	14
5'b01111	4.08	15
5'b10000	4.44	16
5'b10001	4.81	17
5'b10010	5.19	18
5'b10011	5.60	19
5'b10100	6.02	20
5'b10101	6.02	20
5'b10110	6.02	20
5'b10111	6.02	20
5'b11000	6.02	20
5'b11001	6.02	20
5'b11010	6.02	20
5'b11011	6.02	20
5'b11100	6.02	20
5'b11101	6.02	20
5'b11110	6.02	20
5'b11111	6.02	20

Note: The TXPRECURSOR values are defined when the TXPOSTCURSOR =5'b00000
Emphasis = $20\log10(V_{high}/V_{low})$ = |20log10 (Vlow/Vhigh)|

TXPRECURSORINV	In	Async	When set to 1'b1, inverts the polarity of the TXPRECURSOR coefficient. The default is 1'b0.

Table 3-30: **TX Configurable Driver Ports *(Cont'd)***

Port	Dir	Clock Domain	Description
MGTXTXP/MGTHTXP MGTXTXN/MGTHTXN	Out (Pad)	TX Serial Clock	Differential complements of one another forming a differential transmit output pair. These ports represent the pads. The locations of these ports must be constrained (see Implementation, page 30) and brought to the top level of the design.
TXSWING	In	Async	TX swing control for PCI Express PIPE 3.0 Interface. This signal is mapped internally to TXDIFFCTRL/TXBUFDIFFCTRL. 0: Full swing 1: Low swing
TXDIFFPD	In	Async	Reserved.
TXPISOPD	In	Async	Reserved.

Table 3-31 defines the TX configurable driver attributes.

Table 3-31: **TX Configurable Driver Attributes**

Attribute	Type	Description
TX_DEEMPH0[4:0]	5-bit Binary	This attribute has the value of TXPOSTCURSOR[4:0] that has to be mapped when TXDEEMPH = 0. TX_DEEMPH0[4:0] = TXPOSTCURSOR[4:0]. The default is `5'b10100`. Do not modify this value.
TX_DEEMPH1[4:0]	5-bit Binary	This attribute has the value of TXPOSTCURSOR[4:0] that has to be mapped when TXDEEMPH = 1. TX_DEEMPH1[4:0] = TXPOSTCURSOR[4:0]. The default is `5'b01101`. Do not modify this value.
TX_DRIVE_MODE	String	This attribute selects whether PCI Express PIPE 2.0 pins, PCI Express PIPE 3.0 extension pins, or TX Drive Control pins control the TX driver. The default is "DIRECT." DIRECT: TXBUFDIFFCRL, TXDIFFCTRL, TXPOSTCURSOR, TXPRECURSOR and TXMAINCURSOR (If TX_MAINCURSOR_SEL = 1'b1) control the TX driver settings. PIPE: TXDEEMPH, TXMARGIN, TXSWING, TXPRECURSOR and TXMAINCURSOR (If TX_MAINCURSOR_SEL = 1'b1) control the TX driver settings. PIPEGEN3: TXMARGIN, TXSWING, TXPOSTCURSOR, TXPRECURSOR and TXMAINCURSOR (If TX_MAINCURSOR_SEL = 1'b1) control the TX driver settings.
TX_MAINCURSOR_SEL	1-bit Binary	Allows independent control of the main cursor. `1'b0`: The TXMAINCURSOR coefficient is automatically determined by the equation: 80 – TXPOSTCURSOR coefficient – TXPRECURSOR coefficient `1'b1`: TXMAINCURSOR coefficient can be independently set by the TXMAINCURSOR pins within the range specified in the pin description.

Table 3-31: **TX Configurable Driver Attributes *(Cont'd)***

Attribute	Type	Description
TX_MARGIN_FULL_0[6:0]	7-bit Binary	This attribute has the value of TXBUFDIFFCTRL[2:0] and TXDIFFCTRL[3:0] that has to be mapped when TXMARGIN = `000` and TXSWING = `0`. TX_MARGIN_FULL_0 = TXBUFDIFFCTRL[2:0], TXDIFFCTRL[3:0].
TX_MARGIN_FULL_1[6:0]	7-bit Binary	This attribute has the value of TXBUFDIFFCTRL[2:0] and TXDIFFCTRL[3:0] that has to be mapped when TXMARGIN = `001` and TXSWING = `0`. TX_MARGIN_FULL_1 = TXBUFDIFFCTRL[2:0], TXDIFFCTRL[3:0].
TX_MARGIN_FULL_2[6:0]	7-bit Binary	This attribute has the value of TXBUFDIFFCTRL[2:0] and TXDIFFCTRL[3:0] that has to be mapped when TXMARGIN = `010` and TXSWING = `0`. TX_MARGIN_FULL_2 = TXBUFDIFFCTRL[2:0], TXDIFFCTRL[3:0].
TX_MARGIN_FULL_3[6:0]	7-bit Binary	This attribute has the value of TXBUFDIFFCTRL[2:0] and TXDIFFCTRL[3:0] that has to be mapped when TXMARGIN = `011` and TXSWING = `0`. TX_MARGIN_FULL_3 = TXBUFDIFFCTRL[2:0], TXDIFFCTRL[3:0].
TX_MARGIN_FULL_4[6:0]	7-bit Binary	This attribute has the value of TXBUFDIFFCTRL[2:0] and TXDIFFCTRL[3:0] that has to be mapped when TXMARGIN = `100` and TXSWING = `0`. TX_MARGIN_FULL_4 = TXBUFDIFFCTRL[2:0], TXDIFFCTRL[3:0].
TX_MARGIN_LOW_0[6:0]	7-bit Binary	This attribute has the value of TXBUFDIFFCTRL[2:0] and TXDIFFCTRL[3:0] that has to be mapped when TXMARGIN = `000` and TXSWING = `1`. TX_MARGIN_LOW_0 = TXBUFDIFFCTRL[2:0], TXDIFFCTRL[3:0].
TX_MARGIN_LOW_1[6:0]	7-bit Binary	This attribute has the value of TXBUFDIFFCTRL[2:0] and TXDIFFCTRL[3:0] that has to be mapped when TXMARGIN = `001` and TXSWING = `1`. TX_MARGIN_LOW_1 = TXBUFDIFFCTRL[2:0], TXDIFFCTRL[3:0].
TX_MARGIN_LOW_2[6:0]	7-bit Binary	This attribute has the value of TXBUFDIFFCTRL[2:0] and TXDIFFCTRL[3:0] that has to be mapped when TXMARGIN = `010` and TXSWING = `1`. TX_MARGIN_LOW_2 = TXBUFDIFFCTRL[2:0], TXDIFFCTRL[3:0].
TX_MARGIN_LOW_3[6:0]	7-bit Binary	This attribute has the value of TXBUFDIFFCTRL[2:0] and TXDIFFCTRL[3:0] that has to be mapped when TXMARGIN = `011` and TXSWING = `1`. TX_MARGIN_LOW_3 = TXBUFDIFFCTRL[2:0], TXDIFFCTRL[3:0].
TX_MARGIN_LOW_4[6:0]	7-bit Binary	This attribute has the value of TXBUFDIFFCTRL[2:0] and TXDIFFCTRL[3:0] that has to be mapped when TXMARGIN = `100` and TXSWING = `1`. TX_MARGIN_LOW_4 = TXBUFDIFFCTRL[2:0], TXDIFFCTRL[3:0].
TX_PREDRIVER_MODE	1-bit Binary	This is a restricted attribute. Always set this to `1'b0`. Do not modify this attribute.
TX_QPI_STATUS_EN	1-bit Binary	Enables the QPI signals to be passed into the fabric.

Table 3-31: **TX Configurable Driver Attributes *(Cont'd)***

Attribute	Type	Description
TX_EIDLE_ASSERT_DELAY	3-bit Binary	Programmable delay between TXELECIDLE de-assertion to TXP/N exiting electrical idle. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
TX_EIDLE_DEASSERT_DELAY	3-bit Binary	Programmable delay between TXELECIDLE de-assertion to TXP/N exiting electrical idle. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
TX_LOOPBACK_DRIVE_HIZ	1-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.

TX Receiver Detect Support for PCI Express Designs

Functional Description

The PCI Express specification includes a feature that allows the transmitter on a given link to detect if a receiver is present. The decision if a receiver is present is based on the rise time of TXP/TXN. Figure 3-30 shows the circuit model used for receive detection. The GTX/GTH transceiver must be in the P1 power down state to perform receiver detection. Receiver detection requires a 75 nF to 200 nF external coupling capacitor between the transmitter and receiver, and the receiver must be terminated to GND. The receiver detection sequence starts with the assertion of TXDETECTRX. In response, the receiver detection logic drives TXN and TXP to (V_{DD} - V_{SWING}/2) and then releases them. After a programmable interval, the levels of TXN and TXP are compared with a threshold voltage. At the end of the sequence, the receiver detection status is presented on RXSTATUS when PHYSTATUS is asserted High for one cycle.

Figure 3-30: **Receiver Detection Circuit Model**

Ports and Attributes

Table 3-32 describes the TX receiver detection ports.

Table 3-32: **TX Receiver Detection Ports**

Port	Dir	Clock Domain	Description
TXDETECTRX	In	TXUSRCLK2	Used to tell the GTX/GTH transceiver to begin a receiver detection operation. 0: Normal operation. 1: Receiver detection.
TXPD[1:0]	In	TXUSRCLK2	Power up or down the TX and RX of the GTX/GTH transceiver. In PCI Express mode, TXPD and RXPD should be tied to the same source. To perform receiver detection, set these signals to the P1 power saving state. 00: P0 power state for normal operation. 01: P0s power saving state with low recovery time latency. 10: P1 power saving state with longer recovery time latency. 11: P2 power saving state with lowest power.
RXPD[1:0]	In	Async	
PHYSTATUS	Out	RXUSRCLK2	In PCI Express mode, this signal is used to communicate completion of several GTX/GTH transceiver functions, including power management state transitions, rate change, and receiver detection. During receiver detection, this signal is asserted High to indicate receiver detection completion.
RXSTATUS[2:0]	Out	RXUSRCLK2	During receiver detection, this signal is read when PHYSTATUS is asserted High. Only these encodings are valid during receiver detection: 000: Receiver not present. 011: Receiver present.

Table 3-33: **TX Receiver Detection Attributes**

Attribute	Type	Description
TX_RXDETECT_CFG	14-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
TX_RXDETECT_REF	3-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.

Using the TX Receiver Detection for PCI Express

While in the P1 power state, the GTX/GTH transceiver can be instructed to perform a receiver detection operation to determine if there is a receiver at the other end of the link. Figure 3-31 shows an example use mode on how to perform receiver detection in PCI Express mode.

Figure 3-31: **PCI Express Receiver Detection**

Note: Figure 3-31 shows the sequence of events for the receiver present case and is not drawn to scale.

Notes relevant to Figure 3-31:

1. Ensure that the GTX/GTH transceiver has successfully entered the P1 power state with [TX/RX]PD = 2'd2 before receiver detection is performed by asserting TXDETECTRX.
2. Wait for PHYSTATUS = 1'd1 to read RXSTATUS on the same PCLK cycle. In PCI Express mode, PCLK is [TX/RX]USRCLK. If RXSTATUS = 3'd3, then the receiver is present. If RXSTATUS = 3'd0, then the receiver is not present. Deassert TXDETECTRX to exit receiver detection.

TX Out-of-Band Signaling

Functional Description

Each GTX/GTH transceiver provides support for generating the out-of-band (OOB) sequences described in the Serial ATA (SATA), Serial Attach SCSI (SAS) specification, and beaconing described in the PCI Express specification.

Ports and Attributes

Table 3-34 shows the OOB signaling related ports.

Table 3-34: **TX OOB Signaling Ports**

Port	Dir	Clock Domain	Description
TXCOMFINISH	Out	TXUSRCLK2	Indicates completion of transmission of the last SAS or SATA COM beacon.
TXCOMINIT	In	TXUSRCLK2	Initiates transmission of the COMINIT sequence for SATA/SAS.
TXCOMSAS	In	TXUSRCLK2	Initiates transmission of the COMSAS sequence for SAS.
TXCOMWAKE	In	TXUSRCLK2	Initiates transmission of the COMWAKE sequence for SATA/SAS.

Table 3-34: **TX OOB Signaling Ports *(Cont'd)***

Port	Dir	Clock Domain	Description
TXPDELECIDLEMODE	In	TXUSRCLK2	Determines if TXELECIDLE and TXPOWERDOWN should be treated as synchronous or asynchronous signals. Enables compliance during cold and warm PCI Express resets. 1: Asynchronous 0: Synchronous
TXPD[1:0]	In	TXUSRCLK2	Powers down the TX lane according to the PCI Express encoding. `00`: P0 normal operation `01`: P0s low recovery time power down `10`: P1 longer recovery time, RecDet still on `11`: P2 lowest power state. Attributes can control the transition times between these power down mode (PD_TRANS_TIME_FROM_P2, PD_TRANS_TIME_NONE_P2, PD_TRANS_TIME_TO_P2).

Table 3-35 shows the OOB signaling attributes.

Table 3-35: **TX OOB Signaling Attributes**

Attribute	Type	Description
SATA_CPLL_CFG	String	Configuration bits for the CPLL setting related to SAS/SATA. VCO_3000MHZ = Full rate mode VCO_1500MHZ = ½ rate mode VCO_750MHZ = ¼ rate mode
SATA_BURST_SEQ_LEN[3:0]	4-bit Binary	Number of bursts in a COM sequence for SAS/SATA.

Chapter 4

Receiver

RX Overview

Functional Description

This section shows how to configure and use each of the functional blocks inside the receiver (RX). Each GTX/GTH transceiver includes an independent receiver, made up of a PCS and a PMA. Figure 4-1 shows the blocks of the GTX/GTH transceiver RX. High-speed serial data flows from traces on the board into the PMA of the GTX/GTH transceiver RX, into the PCS, and finally into the FPGA logic. Refer to Figure 2-9, page 46 for the description of the channel clocking architecture, which provides clocks to the RX and TX clock dividers.

Figure 4-1: **GTX/GTH Transceiver RX Block Diagram**

The key elements within the GTX/GTH transceiver RX are:

1. RX Analog Front End, page 160
2. RX Out-of-Band Signaling, page 168
3. RX Equalizer (DFE and LPM), page 176
4. RX CDR, page 191
5. RX Fabric Clock Output Control, page 200
6. RX Margin Analysis, page 205
7. RX Polarity Control, page 214

8. RX Pattern Checker, page 214
9. RX Byte and Word Alignment, page 216
10. RX 8B/10B Decoder, page 228
11. RX Buffer Bypass, page 233
12. RX Elastic Buffer, page 248
13. RX Clock Correction, page 253
14. RX Channel Bonding, page 263
15. RX Gearbox, page 275
16. FPGA RX Interface, page 287

RX Analog Front End

Functional Description

The RX analog front end (AFE) is a high-speed current-mode input differential buffer (see Figure 4-1). It has these features:

- Configurable RX termination voltage
- Calibrated termination resistors

Figure 4-2: **RX Analog Front End**

Ports and Attributes

Table 4-1 defines the RX AFE ports.

Table 4-1: **RX AFE Ports**

Port	Dir	Clock Domain	Description
GTXRXN/GTHRXN, GTXRXP/GTHRXP	In (Pad)	RX Serial Clock	Differential complements of one another forming a differential receiver input pair. These ports represent pads. The location of these ports must be constrained (see Implementation, page 30) and brought to the top level of the design.
RXQPISENN	Out	Async	Sense output that registers a 1 or 0 on the GTXRXN/GTHRXN pin.
RXQPISENP	Out	Async	Sense Output that registers a 1 or 0 on the GTXRXP/GTHRXP pin.
RXQPIEN	In	Async	Disables the RX termination for the QPI protocol.

Table 4-2 defines the RX AFE attributes.

Table 4-2: **RX AFE Attributes**

Attribute	Type	Description
RX_CM_SEL [1:0]	2-bit Binary	Controls the mode for the RX termination voltage. `2'b00` - AVTT `2'b01` - GND `2'b10` - Floating `2'b11` - Programmable

Table 4-2: **RX AFE Attributes *(Cont'd)***

Attribute	Type	Description
GTX transceiver: (PMA_RSV2[4], RX_CM_TRIM [2:0]) GTH transceiver: RX_CM_TRIM [3:0]	4-bit Binary	GTX/GTH transceivers: Controls the Common mode in Programmable mode. `4'b0000` – 100 mV `4'b0001` – 200 mV `4'b0010` – 250 mV `4'b0011` – 300 mV `4'b0100` – 350 mV `4'b0101` – 400 mV `4'b0110` – 500 mV `4'b0111` – 550 mV `4'b1000` – 600 mV `4'b1001` – 700 mV `4'b1010` – 800 mV `4'b1011` – 850 mV `4'b1100` – 900 mV `4'b1101` – 950 mV `4'b1110` – 1000 mV `4'b1111` – 1100 mV
TERM_RCAL_CFG	GTX transceiver: 5-bit Binary GTH transceiver: 15-bit binary.	GTX transceiver bits [4:0] Controls the internal termination calibration circuit. Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used. GTH transceiver bits [14:0] Controls the internal termination calibration circuit. Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
TERM_RCAL_OVRD	GTX transceiver: 1-bit Binary GTH transceiver: 3-bit Binary	GTX transceiver: Selects whether the external 100Ω precision resistor is connected to the MGTRREF pin or a value defined by TERM_RCAL_CFG [4:0]. Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used. GTH transceiver bits [2:0]: Selects whether the external 100Ω precision resistor is connected to the MGTRREF pin or a value defined by TERM_RCAL_CFG [14:0]. Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.

GTX and GTH Use Modes—RX Termination

Table 4-3: **Use Mode 1—RX Termination**

Use Mode	External AC Coupling	Term Voltage	Maximum Swing mV_{DPP}	Suggested Protocols and Usage Notes
1	On	Gnd	1200	GTX transceiver: Attribute Settings: • RX_CM_SEL[1:0] = 2'b01 • PMA_RSV2[7:6] = 2'b10 GTH transceiver: Port Settings: • RXDFEAGCTRL[4:3] = 2'b01 Attribute settings: • RX_CM_SEL[1:0] = 2'b01

Figure 4-3: **Use Mode 1**

Table 4-4: **Use Mode 2—RX Termination**

Use Mode	External AC Coupling	Term Voltage	Max Swing mV_{DPP}	Suggested Protocols and Usage Notes
2	On	AVTT	1200	GTX Transceiver: Protocol: • Backplane in LPM mode • CEI-6 (1200 mV_{DPP}) in LPM mode • Wireless in LPM mode • Serial RapidIO in LPM mode Attribute Settings: • RX_CM_SEL[1:0] = `2'b00` • PMA_RSV2[7:6] = `2'b01` GTH Transceiver: Protocol: • Backplane in LPM mode • CEI-6 (1200 mV_{DPP}) in LPM mode • Wireless in LPM mode • Serial RapidIO in LPM mode Port Settings: • RXDFEAGCTRL[4:3] = `2'b10` Attribute Settings: • RX_CM_SEL[1:0] = `2'b00`

Figure 4-4: **Use Mode 2**

Table 4-5: **Use Mode 3—RX Termination**

Use Mode	External AC Coupling	Term Voltage	Max Swing mV_{DPP}	Suggested Protocols and Usage Notes
3	On	800 mV	2000	GTX Transceiver: Protocol: • Optical IF (SONET/SDH/OTU) • SFP+, HD/SD-SDI • XAUI (1600 mV_{DPP}), GbE • PCIe® in DFE and LPM modes • Backplane in DFE mode • CEI-6 (1200 mV_{DPP}) in DFE mode • Wireless in DFE mode • Serial RapidIO in DFE mode • Interlaken for DFE and LPM mode Attribute Settings: • RX_CM_SEL[1:0] = `2'b11` • RX_CM_TRIM [3:0] = `4'b1010` • PMA_RSV2[7:6] = `2'b01`

Table 4-5: **Use Mode 3—RX Termination**

Use Mode	External AC Coupling	Term Voltage	Max Swing mV_{DPP}	Suggested Protocols and Usage Notes
3	On	800 mV	2000	GTH Transceiver: Protocols: • Optical IF (SONET/SDH/OTU) • SFP+, HD/SD-SDI • XAUI (1600 mVdpp), GbE • PCIe in DFE and LPM modes • Backplane in DFE mode • CEI-6 (1200 mVDPP) in DFE mode • Wireless in DFE mode • Serial RapidIO in DFE mode • Interlaken in DFE and LPM modes Port Settings: • RXDFEAGCTRL[4:3] = `2'b10` Attribute Settings: • RX_CM_SEL [1:0] = `2'b11` • RX_CM_TRIM[3:0] = `4'b1010`

Figure 4-5: **Use Mode 3**

Table 4-6: **Use Mode 4—RX Termination**

Use Mode	External AC Coupling	Term Voltage	Max Swing mV_{DPP}	Suggested Protocols and Usage Notes
4	Off	Float	2000	GTX Transceiver: Protocol: • GPON Attribute Settings: • RX_CM_SEL[1:0] = `2'b10` GTH Transceiver: Protocol: • GPON Port Settings: • Depends on circuit implementation. Likely High Common mode (RXDFEAGCTRL[4:3] = `2'b10`) Attribute Settings: • RX_CM_SEL[1:0] = `2'b10` **Note:** This only works in LPM mode.

Figure 4-6: **Use Mode 4**

RX Out-of-Band Signaling

Functional Description

The GTX/GTH receiver provides support for decoding the out-of-band (OOB) sequences described in the Serial ATA (SATA) and Serial Attach SCSI (SAS) specifications and supports beaconing described in the PCI Express specification. GTX/GTH receiver support for SATA/SAS OOB signaling consists of the analog circuitry required to decode the OOB signal state and state machines to decode bursts of OOB signals for SATA/SAS COM sequences.

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

Ports and Attributes

Table 4-7 defines the OOB signaling related ports.

Table 4-7: **RX OOB Signaling Ports**

Port	Dir	Clock Domain	Description
RXOOBRESET	In	Async	Reserved. Tie to GND.
RXELECIDLEMODE[1:0]	In	Async	Input signal to control the behavior of RXELECIDLE. `2'b00` = RXELECIDLE indicates the status of the OOB signal detection circuit. Use this setting for PCIe, SATA/SAS, and protocols/applications using OOB. In these cases, the OOB circuit must be powered on. `2'b11` = RXELECIDLE outputs a static `1'b0`. Use this setting for non-OOB protocols.
RXELECIDLE	Out	Async	This output indicates the status of OOB signal detection and is only valid for PCIe, SATA/SAS, and protocols/applications using OOB. In these cases, the OOB circuit must be powered on. `0` = Activity is seen on the receiver `1` = No activity is seen For non-OOB protocols, RXELECIDLEMODE[1:0] must be set to `2'b11`. RXELECIDLE outputs a static `1'b0` and in this case does not indicate signal detection status.
RXCOMINITDET	Out	RXUSRCLK2	Indicates reception of the COMINIT sequence for SATA/SAS.
RXCOMSASDET	Out	RXUSRCLK2	Indicates reception of the COMSAS sequence for SAS.
RXCOMWAKEDET	Out	RXUSRCLK2	Indicates reception of the COMWAKE sequence for SATA/SAS.

Table 4-8 defines the OOB signaling attributes.

Table 4-8: **RX OOB Signaling Attributes**

Attribute	Type	Description
PCS_RSVD_ATTR[8]	1-bit Binary	OOB power up. The OOB circuit can be optionally powered down when not being used. `1'b0` = Circuit powered down `1'b1` = Circuit powered up (PCIe, SATA/SAS, protocols/applications using OOB)
PCS_RSVD_ATTR[3]	1-bit Binary	`1'b0` = Selects sysclk. `1'b1` = Selects port CLKRSVD[0].
GTH Transceiver: RXOOB_CLK_CFG	1-bit Binary	`1'b0` = Selects sysclk. `1'b1` = Selects port sigvalidclk.
RXOOB_CFG[6:0]	7-bit Binary	OOB block configuration. The default value is `7'b0000110`.
SATA_BURST_VAL[2:0]	3-bit Binary	Number of bursts to declare a COM match for SAS/SATA. The default value is `3'b100`.
SATA_EIDLE_VAL[2:0]	3-bit Binary	Number of idles to declare a COM match for SAS/SATA. The default value is `3'b100`.
SAS_MIN_COM	Integer	1-63. Lower bound on activity burst for COM FSM for SAS/SATA. The default value is 36.
SATA_MIN_INIT	Integer	1-63. Lower bound on idle count during COMSAS for SAS. The default value is 12.
SATA_MIN_WAKE	Integer	1-63. Lower bound on idle count during COMINIT/COMRESET for SAS/SATA. The default value is 4.
SATA_MAX_BURST	Integer	1-63. Upper bound on activity burst for COM FSM for SAS/SATA. The default value is 8.
SATA_MIN_BURST	Integer	1-61. Lower bound on activity burst for COM FSM for SAS/SATA. The default value is 8.
SAS_MAX_COM	Integer	1-127. Upper bound on idle count during COMSAS for SAS. The default value is 64.
SATA_MAX_INIT	Integer	1-63. Upper bound on idle count during COMINIT/COMRESET for SAS/SATA. The default value is 21.
SATA_MAX_WAKE	Integer	1-63. Upper bound on idle count during COMWAKE for SAS/SATA. The default value is 7.

GTX/GTH Use Mode

To use OOB, the following RX termination conditions need to be applied:

- AC-coupled case: Termination voltage should be 800 mV or greater
- DC-coupled case: Termination voltage should be 900 mV or greater

Also, the attribute PCS_RSVD_ATTR[8] should be set to `1'b1`. The OOB circuit has two possible sources from which it can receive a clock, as shown in Figure 4-7.

Figure 4-7: **Clocking Mechanism for the OOB Detect Circuit**

The attribute PCS_RSVD_ATTR[3] (GTX)/RXOOB_CLK_CFG (GTH) controls the source of oobclk. Setting PCS_RSVD_ATTR[3] (GTX)/RXOOB_CLK_CFG (GTH) to 0 selects the sysclk. Setting PCS_RSVD_ATTR[3] (GTX)/RXOOB_CLK_CFG (GTH) to 1 selects an alternative clock source from CLKRSVD[0] (GTX)/SIGVALIDCLK (GTH). A divided down reference clock can be connected to the CLKRSVD[0] (GTX)/SIGVALIDCLK (GTH) pin, providing an alternative clock for the OOB circuit.

The port that controls the sysclk source is TXSYSCLKSEL. Setting this port to `1'b0` selects the reference clock from the channel PLL, and setting this port to `1'b1` selects the reference clock from the common PLL.

The divided down clock(s) requires no special phase relationships between other clocks in the SERDES. However, there is a requirement of a 50% duty cycle. Figure 4-8 and Figure 4-9 show the method for clock division. Figure 4-8 shows how a simple toggle flip-flop can be used to divide the REFCLK.

Figure 4-8: **Toggle Flip-Flop to Divide REFCLK**

Figure 4-9 shows how cascading several divide-by-two circuits produces higher order clock dividers such as divide-by-4 and divide-by-8.

Figure 4-9: **Clock Dividers**

Use Modes

For OOB operating at a line rate of 1.5 Gb/s or below, see the flowchart in Figure 4-10 to determine the frequency *f* of the OOB clock.

Figure 4-10: **Flowchart for Protocols with Line Rates ≤ 1.5G**

The requirement in Equation 4-1 must be satisfied for the OOB to work correctly.

$$f \leq \text{linerate}/(3 \times \text{runlength}) \qquad \textit{Equation 4-1}$$

OOB operating at line rates > 1.5 Gb/s is an advanced feature. Operation for certain protocols at higher line rates such as PCIe (Gen1 and Gen2) and SATA are addressed in Table 4-9.

Table 4-9: **OOB Guidelines for Operating Rates above 1.5 Gb/s**

Protocol	Operation
PCIe Gen1	See Figure 4-11 for the algorithm to determine whether the RX is in electrical idle. If a scrambler has not been used, the RX electrical idle should not be used for internal detect logic of the hold/reset logic of the DFE, LPM, or CDR[(1)]. The user needs to verify received data to decide whether or not an electrical idle state is present i.e., qualification using incoming data is essential in this mode of operation. If a scrambler is used, electrical idle may solely be used to determine whether the RX is in electrical idle.
PCIe Gen2	See Figure 4-12 for the algorithm to determine whether the RX is in electrical idle. Other methods that can be used for this are shown in Figure 4-13 and Figure 4-14. RX electrical idle should not be used for internal detect logic of the hold/reset logic of the DFE, LPM, or CDR[(1)]. The user needs to verify received data to decide whether or not an electrical idle state is present i.e., qualification using incoming data is essential in this mode of operation.
SATA 1.5 Gb/s	Use Equation 4-1 to derive the appropriate OOB clock (see Figure 4-10).
SATA 3 Gb/s	See Figure 4-15 for the algorithm to determine whether the RX is in electrical idle. RX electrical idle should not be used for internal detect logic of the hold/reset logic of the DFE, LPM, or CDR[(1)]. The user needs to verify received data to decide whether or not an electrical idle state is present i.e., qualification using incoming data is essential in this mode of operation.
SATA 6 Gb/s	See Figure 4-15 for the algorithm to determine whether the RX is in electrical idle. RX electrical idle should not be used for internal detect logic of the hold/reset logic of the DFE, LPM, or CDR[(1)]. The user needs to verify received data to decide whether or not an electrical idle state is present i.e., qualification using incoming data is essential in this mode of operation.
PCIe Gen 3 or Gen2	See Figure 4-13 and Figure 4-14 for the algorithm to determine whether the RX is in electrical idle. While entering and exiting electrical idle, the EIOS detection must be used along with RXELECIDLE assertion to determine whether the RX is in electrical idle. RX electrical idle should not be used for internal detect logic of the hold/reset logic of the DFE, LPM, or CDR[(1)]. The user needs to verify received data to decide whether or not an electrical idle state is present i.e., qualification using incoming data is essential in this mode of operation.

Table 4-9: **OOB Guidelines for Operating Rates above 1.5 Gb/s *(Cont'd)***

Protocol	Operation

Notes:

1. The attributes pertaining to DFE, LPM, and CDR are:
- RXCDR_HOLD_DURING_EIDLE
- RXCDR_FR_RESET_ON_EIDLE
- RXCDR_PH_RESET_ON_EIDLE
- RX_DFE_LPM_HOLD_DURING_EIDLE
- RXBUF_RESET_ON_EIDLE
- RXBUF_EIDLE_HI_CNT
- RXBUF_EIDLE_LO_CNT

Figure 4-11: **Flowchart for PCIe Gen1**

Figure 4-12: **Flowchart for PCIe Gen2**

Figure 4-13: **Flowchart for Entry to RX Electrical Idle for PCIe Gen2 or Gen3**

Figure 4-14: **Flowchart for Exit from RX Electrical Idle for PCIe Gen2 or Gen3**

Figure 4-15: **Flowchart for SATA 3G or SATA 6G**

RX Equalizer (DFE and LPM)

Functional Description

A serial link bit error rate (BER) performance is a function of the transmitter, the transmission media, and the receiver. The transmission media or channel is bandwidth-limited and the signal traveling through it is subjected to attenuation and distortion.

There are two types of adaptive filtering available to the GTX/GTH receiver depending on system level trade-offs between power and performance. Optimized for power with lower channel loss, the GTX/GTH receiver has a power-efficient adaptive mode named the low-power mode (LPM), see Figure 4-16. For equalizing lossier channels, the DFE mode is available. See Figure 4-17 for the GTX transceiver and Figure 4-18 for the GTH transceiver.

The DFE allows better compensation of transmission channel losses by providing a closer adjustment of filter parameters than when using a linear equalizer. However, a DFE cannot remove the pre-cursor of a transmitted bit; it only compensates for the post cursors. A linear equalizer allows pre-cursor and post-cursor gain. The GTX/GTH RX DFE mode is a discrete-time adaptive high-pass filter. The TAP values of the DFE are the coefficients of this filter that are set by the adaptive algorithm.

Figure 4-16: **LPM Mode**

Figure 4-17: **GTX DFE Mode**

Figure 4-18: **GTH DFE Mode**

Ports and Attributes

Table 4-10 defines the RX equalizer ports.

Table 4-10: **RX Equalizer Ports**

Port	Dir	Clock Domain	Description
RXLPMEN	In	RXUSRCLK2	RX datapath 0: DFE 1: LPM
RXDFELPMRESET	In	RXUSRCLK2	Reset for LPM and DFE datapath. Must be toggled after switching between modes to initialize adaptation.
{RXOSHOLD, RXOSOVRDEN}	In	RXUSRCLK2	{HOLD,OVRDEN} RX LPM or DFE `2'b00`: OS Offset cancelation loop adapt `2'b10`: Freeze current adapt value `2'bx1`: Override OS value according to attribute RX_OS_CFG The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
{RXLPMLFHOLD, RXLPMLFKLOVRDEN}	In	RXUSRCLK2	{HOLD,OVRDEN} RX LPM `2'b00`: KL Low frequency loop adapt `2'b10`: Freeze current adapt value `2'bx1`: Override KL value according to attribute RXLPM_LF_CFG The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
{RXLPMHFHOLD, RXLPMHFOVRDEN}	In	RXUSRCLK2	{HOLD,OVRDEN} RX LPM `2'b00`: KH High frequency loop adapt `2'b10`: Freeze current adapt value `2'bx1`: Override KH value according to attribute RXLPM_HF_CFG The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
{RXDFEAGCHOLD, RXDFEAGCOVRDEN}	In	RXUSRCLK2	{HOLD,OVRDEN} RX DFE `2'b00`: Automatic gain control (AGC) loop adapt `2'b10`: Freeze current AGC adapt value `2'bx1`: Override AGC value according to attribute RX_DFE_GAIN_CFG The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
{RXDFELFHOLD, RXDFELFOVRDEN}	In	RXUSRCLK2	{HOLD,OVRDEN} RX DFE `2'b00`: KL Low frequency loop adapt `2'b10`: Freeze current KL adapt value `2'bx1`: Override KL value according to attribute RX_DFE_KL_CFG The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.

Table 4-10: **RX Equalizer Ports *(Cont'd)***

Port	Dir	Clock Domain	Description
{RXDFEUTHOLD, RXDFEUTOVRDEN}	In	RXUSRCLK2	{HOLD,OVRDEN} RX DFE `2'b00`: UT Unrolled threshold loop adapt `2'b10`: Freeze current UT adapt value `2'bx1`: Override UT value according to attribute RX_DFE_UT_CFG The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
{RXDFEVPHOLD, RXDFEVPOVRDEN}	In	RXUSRCLK2	{HOLD,OVRDEN} RX DFE `2'b00`: VP Voltage peak loop adapt `2'b10`: Freeze current VP adapt value `2'bx1`: Override VP value according to attribute RX_DFE_VP_CFG The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
{RXDFETAP2HOLD, RXDFETAP2OVRDEN}	In	RXUSRCLK2	{HOLD,OVRDEN} RX DFE `2'b00`: TAP2 loop adapt `2'b10`: Freeze current TAP2 adapt value `2'bx1`: Override TAP2 value according to attribute RX_DFE_H2_CFG The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
{RXDFETAP3HOLD, RXDFETAP3OVRDEN}	In	RXUSRCLK2	{HOLD,OVRDEN} RX DFE `2'b00`: TAP3 loop adapt `2'b10`: Freeze current TAP3 adapt value `2'bx1`: Override TAP3 value according to attribute RX_DFE_H3_CFG The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
{RXDFETAP4HOLD, RXDFETAP4OVRDEN}	In	RXUSRCLK2	{HOLD,OVRDEN} RX DFE `2'b00`: TAP4 loop adapt `2'b10`: Freeze current TAP4 adapt value `2'bx1`: Override TAP4 value according to attribute RX_DFE_H4_CFG The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
{RXDFETAP5HOLD, RXDFETAP5OVRDEN}	In	RXUSRCLK2	{HOLD,OVRDEN} RX DFE `2'b00`: TAP5 loop adapt `2'b10`: Freeze current TAP5 adapt value `2'bx1`: Override TAP5 value according to attribute RX_DFE_H5_CFG The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
RXDFECM1EN	In	RXUSRCLK2	Reserved.
RXDFEXYDHOLD	In	RXUSRCLK2	Reserved.
RXDFEXYDOVRDEN	In	RXUSRCLK2	Reserved.

Table 4-10: **RX Equalizer Ports *(Cont'd)***

Port	Dir	Clock Domain	Description
RXDFEXYDEN	In	RXUSRCLK2	Reserved. Set to `1'b1`.
RXMONITORSEL[1:0]	In	Async	Select signal for RXMONITOROUT[6:0] `2'b00`: Reserved `2'b01`: Select AGC loop `2'b10`: Select UT loop `2'b11`: Select VP loop
RXMONITOROUT[6:0]	Out	Async	GTX transceiver: RXDFEVP[6:0] = RXMONITOROUT[6:0] RXDFEUT[6:0] = RXMONITOROUT[6:0] RXDFEAGC[4:0] = RXMONITOROUT[4:0] GTH transceiver: RXDFEVP[6:0] = RXMONITOROUT[6:0] RXDFEUT[6:0] = RXMONITOROUT[6:0] RXDFEAGC[3:0] = RXMONITOROUT[4:1]
GTH transceiver: {RXDFETAP6HOLD, RXDFETAP6OVRDEN}	In	RXUSRCLK2	{HOLD,OVRDEN} RX DFE `2'b00`: TAP6 loop adapt `2'b10`: Freeze current TAP6 adapt value `2'bx1`: Override TAP6 value according to the reserved attribute The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
GTH transceiver: {RXDFETAP7HOLD, RXDFETAP7OVRDEN}	In	RXUSRCLK2	{HOLD,OVRDEN} RX DFE `2'b00`: TAP7 loop adapt `2'b10`: Freeze current `2'bx1`: Override TAP7 value according to the reserved attribute TAP7 adapt value The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
GTH transceiver: {RXDFESLIDETAPHOLD, RXDFESLIDETAPINITOVRDEN}	In	RXUSRCLK2	{HOLD,OVRDEN} RX DFE `2'b00`: Sliding TAP loop adapt `2'b10`: Freeze current sliding TAP adapt value `2'bx1`: Override sliding TAP value according to port RXDFESLIDETAP The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
GTH transceiver: RXDFESLIDETAPADAPTEN	In	RXUSRCLK2	RX DFE: `1'b0`: Disable sliding TAP adaptation `1'b1`: Enable sliding TAP adaptation
GTH transceiver: RXDFESLIDETAP[4:0]	In	RXUSRCLK2	RX DFE: Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.

Table 4-10: **RX Equalizer Ports *(Cont'd)***

Port	Dir	Clock Domain	Description
GTH transceiver: RXDFEAGCTRL[4:0]	In	RXUSRCLK2	RX DFE: Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
GTH transceiver: RXDFESLIDETAPSTROBE	In	RXUSRCLK2	RX DFE: Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
GTH transceiver: RXDFESLIDETAPID[5:0]	In	RXUSRCLK2	RX DFE: Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
GTH transceiver: RXDFESLIDETAPONLYADAPTEN	In	RXUSRCLK2	RX DFE: Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
GTH transceiver: RXOSINTEN	In	RXUSRCLK2	RX LPM & DFE: Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
GTH transceiver: RXOSINTNTRLEN	In	RXUSRCLK2	RX LPM & DFE: Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
GTH transceiver: RXOSINTCFG[3:0]	In	RXUSRCLK2	RX LPM & DFE: Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
GTH transceiver: RXOSINTID0[3:0]	In	RXUSRCLK2	RX LPM & DFE: Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
GTH transceiver: RXOSINTOVRDEN	In	RXUSRCLK2	RX LPM & DFE: Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
GTH transceiver: RXOSINTSTROBE	In	RXUSRCLK2	RX LPM & DFE: Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
GTH transceiver: RXOSINTSTROBESTARTED	In	RXUSRCLK2	RX LPM & DFE: Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
GTH transceiver: {RXOSINTHOLD,RXOSINTTESTOVRDEN}	In	RXUSRCLK2	{HOLD,OVRDEN} `2'b00`: Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used. `2'b10`: Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used. `2'bx1`: Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.

Table 4-10: **RX Equalizer Ports *(Cont'd)***

Port	Dir	Clock Domain	Description
GTH transceiver: RXADAPTSELTEST[13:0]	In	RXUSRCLK2	RX LPM & DFE: Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
GTH transceiver: RXDFEVSEN	In	RXUSRCLK2	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
GTH transceiver: RXDFEXYDEN	In	RXUSRCLK2	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
GTH transceiver: RXDFESTADAPTDONE	Out	RXUSRCLK2	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
GTH transceiver: RXDFESLIDETAPSTROBESTARTED	Out	RXUSRCLK2	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
GTH transceiver: RXDFESLIDETAPSTROBEDONE	Out	RXUSRCLK2	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
GTH transceiver: RXDFESLIDETAPSTARTED	Out	RXUSRCLK2	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
GTH transceiver: RXOSINTDONE	Out	RXUSRCLK2	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.

Table 4-11 defines the RX equalizer attributes.

Table 4-11: **RX Equalizer Attributes**

Attribute	Type	Description
RX_OS_CFG[12:0]	13-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used. The default value is `13'h0080`.
RXLPM_LF_CFG[13:0]	14-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used. The default value is `14'h00F0`.
RXLPM_HF_CFG[13:0]	14-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used. The default value is `14'h00F0`.
RX_DFE_LPM_CFG[15:0]	16-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used. Also refer to GTX Use Modes, page 187.

Table 4-11: **RX Equalizer Attributes *(Cont'd)***

Attribute	Type	Description
RX_DFE_GAIN_CFG[22:0]	23-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used. The default value is `23'h020FEA`.
RX_DFE_H2_CFG[11:0]	12-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used. The default value is `12'h000`.
RX_DFE_H3_CFG[11:0]	12-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used. The default value is `12'h040`.
RX_DFE_H4_CFG[10:0]	11-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used. The default value is `11'h0E0`.
RX_DFE_H5_CFG[10:0]	11-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used. The default value is `11'h0E0`.
PMA_RSV[31:0]	32-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used. These bits relate to RXPI and are line rate dependent: • `32'h0001_8480`: Lower line rates, CPLL full range, 6 GHz ≤ QPLL VCO rate < 6.6 GHz • `32'h001E_7080`: Higher line rates, QPLL > 6.6 GHz
RX_DFE_LPM_HOLD_DURING_EIDLE	1-bit Binary	`1'b0`: Default setting. `1'b1`: Restores the DFE contents from internal registers after termination of an electrical idle state for PCI Express operation. Holds the DFE circuit in reset when an electrical idle condition is detected. ***Note:*** For channels with large attenuation (lossy channels typically exceeding 15 dB at Nyquist), it is recommended that RX_DFE_LPM_HOLD_DURING_EIDLE be set to `1'b0` because fast transitioning data patterns like the 101010 sequence in CJPAT/CJTPAT can accidentally trigger an electrical idle.
RX_DFE_XYD_CFG	13-bit Binary	Reserved. This attribute should be set to `13'h0000`, which is also the default value.
GTH transceiver: RX_DFE_H6_CFG[10:0]	11-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.

Table 4-11: **RX Equalizer Attributes *(Cont'd)***

Attribute	Type	Description
GTH transceiver: RX_DFE_H7_CFG[10:0]	11-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
GTH transceiver: RX_DFE_ST_CFG[53:0]	54-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
GTH transceiver: RX_DFELPM_KLKH_AGC_STUP_EN	1-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
GTH transceiver: RX_DFELPM_CFG0[3:0]	4-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
GTH transceiver: RX_DFELPM_CFG1	1-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
GTH transceiver: RX_DFE_KL_LPM_KH_CFG0[1:0]	2-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
GTH transceiver: RX_DFE_KL_LPM_KH_CFG1[2:0]	3-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
GTH transceiver: RX_DFE_KL_LPM_KH_CFG2[3:0]	4-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
GTH transceiver: RX_DFE_KL_LPM_KL_CFG0[1:0]	2-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
GTH transceiver: RX_DFE_KL_LPM_KL_CFG1[2:0]	3-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
GTH transceiver: RX_DFE_KL_LPM_KL_CFG2[3:0]	4-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
GTH transceiver: RX_DFE_AGC_CFG0[1:0]	2-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
GTH transceiver: RX_DFE_AGC_CFG1[2:0]	3-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
GTH transceiver: RX_DFE_AGC_CFG2[3:0]	4-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.

Table 4-11: **RX Equalizer Attributes *(Cont'd)***

Attribute	Type	Description
GTH transceiver: RX_DFE_KL_LPM_KH_OVRDEN	1-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
GTH transceiver: RX_DFE_KL_LPM_KL_OVRDEN	1-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
GTH transceiver: ADAPT_CFG0[19:0]	20-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
GTX transceiver: PMA_RSV4[31:0] (no connect) GTH transceiver: PMA_RSV4[14:0]	32-bit Binary 15-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
GTX transceiver: PMA_RSV2[15:0] GTH transceiver: PMA_RSV2[31:0]	16-bit Binary 32-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used. The default value is `16'h0050`.
GTX transceiver: RX_BIAS_CFG[11:0] GTH transceiver: RX_BIAS_CFG[23:0]	12-bit Binary 24-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used. The default value is `12'h040`.
GTX transceiver: RX_DEBUG_CFG[11:0] GTH transceiver: RX_DEBUG_CFG[13:0]	12-bit Binary 14-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used. Also refer to Digital Monitor, page 94.
GTX transceiver: RX_DFE_KL_CFG[12:0] RX_DFE_KL_CFG2[31:0] GTH transceiver: RX_DFE_KL_CFG2[31:0]	13-bit Binary 32-bit Binary 32-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used. Also refer to GTX Use Modes, page 187. Default values are: • RX_DFE_KL_CFG = `13'h00F0` • RX_DFE_KL_CFG2 = `32'h3010_D90C`

Table 4-11: **RX Equalizer Attributes *(Cont'd)***

Attribute	Type	Description
RX_DFE_UT_CFG[16:0]	17-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used. The default value is `17'h11E00`.
RX_DFE_VP_CFG[16:0]	17-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used. The default value is `17'h03F03`.

GTX Use Modes

Choosing Between LPM and DFE Modes

LPM mode is recommended for applications with line rates up to 11.2 Gb/s for short reach applications, with channel losses of 12 dB or less at the Nyquist frequency.

DFE mode is recommended for medium- to long-reach applications, with channel losses of 8 dB and above at the Nyquist frequency. A DFE has the advantage of equalizing a channel without amplifying noise and crosstalk. DFE can also correct reflections caused by channel discontinuities within the first five post cursors in GTX transceivers. DFE mode is the best choice when crosstalk is a concern or when reflections are identified in a single-bit response analysis.

Both LPM and DFE modes must be carefully considered in 8B/10B applications or where data scrambling is not employed. To properly adapt to data, the auto adaptation in both LPM and DFE modes requires incoming data to be random. For example, in a XAUI application, the user payload data is non-scrambled and 8B/10B encoded. While the user payload is generally random, the frequency content of the data is inherently limited by the encoding, and there is nothing defined in the protocol to prevent repeated patterns from occurring. These repeated patterns can cause the auto adapting algorithms to drift away from the ideal equalization setting. Patterns with characteristics similar to PRBS7 (or higher polynomial) are sufficiently random for auto adaptation to properly choose the correct equalization setting.

Using LPM Mode

The GTX LPM and DFE modes employ different CTLE blocks. In addition, GTX transceivers have a baseline wander cancellation circuit. In GTX transceivers, the CTLE and baseline wander cancellation in LPM mode is fully adapting and requires no manual tuning of the gain attribute settings.

LPM mode is selected by setting the RXLPMEN port to `1'b1`.

Using DFE Mode

The DFE mode utilizes an AGC, CTLE, DFE, and baseline wander cancellation to equalize the effects of the channel.

DFE mode is selected by setting the RXLPMEN port to `1'b0`.

The AGC, DFE, and baseline wander cancellation are auto adapting. The adaptation can be held by asserting the HOLD port for the particular adaptation loop. For example, to hold the current adaptation value of DFE Tap 2, the RXDFETAP2HOLD port should be set to 1. The adapted value is held for as long as this port is asserted or until GTRXRESET, RXPMARESET, or RXDFELPMRESET is pulsed. When GTRXRESET, RXPMARESET, or

RXDFELPMRESET is pulsed, the held value is overwritten by the initial value set in the RX equalizer attributes. To allow the adaptation to continue, RXDFETAP2HOLD should be set back to 0. Table 4-10 contains all the HOLD ports for the various adapting loops.

DFE Training/Initialization

RXDFEAGCHOLD and RXDFELFHOLD should be asserted after training to freeze the AGC adapt value. (Refer to DS183, *Virtex-7 FPGAs Data Sheet: DC and Switching Characteristics* for T_{DLOCK}, the time it takes to lock to the data present at the input.)

GTX Transceivers Only: DFE Mode—CTLE Use Model

The CTLE in GTX DFE mode can be used in two ways: auto adapting and fixed. Channel analysis is required to use the auto adapting CTLE mode and is considered an advanced use mode.

GTX Transceivers Only: CTLE Auto Adapting Mode (Advanced Use Mode)

The CTLE in the GTX DFE mode can be used in auto adapting mode when the channel insertion loss deviation is minimal.

To use the CTLE in Auto Adapting mode, these attributes must be set:

- RX_BIAS_CFG[5:4] = `2'b11`
- RX_DFE_KL_CFG2[26:23] = `4'b0111`
- RX_DFE_LPM_CFG[5:2] = `4'b0010`

GTX Transceivers Only: CTLE Fixed Mode

To use the CTLE in fixed mode, the insertion loss at the Nyquist frequency or line rate divided by two must be known. The CTLE is made up of two components: a mid/high frequency boosting component followed by a wide-band gain component.

The mid/high frequency boosting component is controlled by the RX_DFE_KL_CFG2 attribute, bit locations [8:5] and [17:14]. These two attribute fields must remain equal to one another.

The wide-band gain component is controlled by the RX_DFE_KL_CFG2[26:23] attribute. This CTLE component can be used to boost or attenuate all frequencies within the bandwidth shown in Figure 4-19.

Figure 4-19: **GTX DFE Mode CTLE Frequency Response**

The allowable range for the CTLE attributes and the typical mid/high frequency boost is shown in Table 4-12. The mid/high frequency boost is the relative difference in gain between DC and the mid/high frequency peaking.

Table 4-12: **GTX CTLE Attribute Ranges and Typical Mid/High Frequency Boost**

Mid/High Frequency Boost		Wide-Band Gain	Typical Boost (dB)
RX_DFE_KL_CFG2[8:5]	RX_DFE_KL_CFG2[17:14]	RX_DFE_KL_CFG2[26:23]	
0	0	0–7	2
1	1	0–7	3
2	2	0–7	5
3	3	0–7	6
4	4	0–7	8
5	5	0–7	9

Table 4-13 shows typical use models for different channel insertion losses at the Nyquist frequency.

Table 4-13: **GTX Use Models for Channel Insertion Losses at Nyquist Frequency**

Loss at Nyquist (dB)	TX FIR Emphasis (dB)	Mid/High Frequency Boost RX_DFE_KL_CFG2		Wide-Band Gain RX_DFE_KL_CFG2
		Bits [8:5]	Bits [17:14]	Bits [26:23]
<15	0	0	0	3–6
		1	1	
15–25	6	1	1	(1)
		2	2	
		3	3	
25–30	6	4	4	0–3
		5	5	

Notes:

1. General guidelines for wide-band gain:
   If launch amplitude ≥ 1000 mVppd, wide-band gain = 7.
   If launch amplitude ≤ 300 mVppd, wide-band gain = 0.
   Otherwise, wide-band gain = round (launch amplitude – 300)/100.

GTH Use Modes

Choosing Between LPM and DFE Modes

LPM mode is recommended for short reach applications with channel losses of 14 dB or less at the Nyquist frequency.

DFE mode is recommended for medium- to long-reach applications with channel losses of 8 dB and above at the Nyquist frequency. DFE mode has the advantage of equalizing a channel without amplifying noise and crosstalk. DFE can also correct reflections caused by channel discontinuities within the first 63 post cursors in GTH transceivers. DFE mode is the best choice when crosstalk is a concern or when reflections are identified in a single-bit response analysis.

Both LPM and DFE modes must be carefully considered in 8B/10B applications or where data scrambling is not employed. To properly adapt to data, the auto adaptation in both LPM and DFE modes requires incoming data to be random. For example, in a XAUI application, the user payload data is non-scrambled and 8B/10B encoded. While the user payload is generally random, the frequency content of the data is inherently limited by the encoding, and there is nothing defined in the protocol to prevent repeated patterns from occurring. These repeated patterns can cause the auto adapting algorithms to drift away from the ideal equalization setting. Patterns with characteristics similar to PRBS7 (or higher polynomials) are sufficiently random for auto adaptation to properly choose the correct equalization setting.

Using LPM Mode

The GTH LPM and DFE modes share the same CTLE blocks. In addition, GTH transceivers have a baseline wander cancellation circuit. In the GTH transceiver, the CTLE and baseline

wander cancellation in LPM mode is fully adapting and requires no manual tuning of the gain attribute settings. LPM mode is selected by setting the RXLPMEN port to `1'b1`.

Using DFE Mode

Full-auto adapt is the general use mode for GTH DFE. The CTLE and AGC are fully adapting and require no manual tuning of gain attribute settings. For channel loss and TX finite impulse response (FIR) guidelines, refer to Table 4-14.

In both LPM and DFE modes, these attributes must be set to configure the CTLE and AGC in full auto adapting mode:

- RX_DFE_KL_LPM_KH_OVRDEN = `1'b1`
- RX_DFE_KL_LPM_KL_OVRDEN = `1'b1`
- RX_DFE_AGC_OVRDEN = `1'b1`

Table 4-14: **GTH DFE Channel Loss Guide Line**

RX Modes	TX FIR Emphasis (dB)	Channel Loss @ Nyquist (dB)	Nyquist Frequency (GHz)	Line Rate (Gb/s)
DFE	0	Short Reach 15 dB and less	Up to 6.55	Up to 13.1
	6	Mid Reach 15–20 dB	Up to 6.25	Up to 13.1
	6	Long Reach 20 dB+	Up to 6.55	Up to 13.1

Sliding taps are an advanced use mode.

GTX and GTH Transceivers: Switching Between LPM and DFE Modes at Run Time

In multi-rate applications, it may be required to switch between LPM (in lower line rates) to DFE (in higher line rates). These steps should be followed to switch between LPM and DFE modes:

1. Invert the current value of RXLPMEN (RXLPMEN = ~RXLPMEN).
2. Reset the receiver's PMA by asserting RXPMARESET.

See RX Initialization and Reset, page 71 for more information regarding RXPMARESET.

RX CDR

Functional Description

The RX clock data recovery (CDR) circuit in each GTXE2_CHANNEL/ GTHE2_CHANNEL transceiver extracts the recovered clock and data from an incoming

data stream. Figure 4-20 illustrates the architecture of the CDR block. Clock paths are shown with dotted lines for clarity.

Figure 4-20: **CDR Detail**

The GTXE2_CHANNEL/GTHE2_CHANNEL transceiver employs phase rotator CDR architecture. Incoming data first goes through receiver equalization stages. The equalized data is captured by an edge and a data sampler. The data captured by the data sampler is fed to the CDR state machine and the downstream transceiver blocks.

The CDR state machine uses the data from both the edge and data samplers to determine the phase of the incoming data stream and to control the phase interpolators (PIs). The phase for the edge sampler is locked to the transition region of the data stream while the phase of the data sampler is positioned in the middle of the data eye.

Figure 4-21: **CDR Sampler Positions**

The CPLL or QPLL provides a base clock to the phase interpolator. The phase interpolator in turn produces fine, evenly spaced sampling phases to allow the CDR state machine to have fine phase control. The CDR state machine can track incoming data streams that can have a frequency offset from the local PLL reference clock.

Ports and Attributes

Table 4-15 defines the CDR ports.

Table 4-15: **CDR Ports**

Port	Dir	Clock Domain	Description
RXCDRFREQRESET	In	Async	CDR frequency detector reset.
RXCDRHOLD	In	Async	Hold the CDR control loop frozen.
RXCDROVRDEN	In	Async	Reserved.
RXCDRRESET	In	Async	CDR phase detector reset.
RXCDRRESETRSV	In	Async	Reserved.
RXRATE[2:0]	In	RXUSRCLK2	Dynamic pins to automatically change effective PLL dividers in the GTX/GTH transceiver RX. These ports are used for PCI Express and other standards. `000`: Use RXOUT_DIV attributes `001`: Divide by 1 `010`: Divide by 2 `011`: Divide by 4 `100`: Divide by 8 `101`: Divide by 16 `110`: Divide by 1 `111`: Divide by 1 RXBUF_RESET_ON_RATE_CHANGE attribute enables optional automatic reset.
RXCDRLOCK	Out	Async	Reserved.
RXOSHOLD	In	Async	When set to `1'b1`, the current value of the offset cancellation is held. When set to `1'b0`, the offset cancellation is adapted.
RXOSOVRDEN	In	Async	When set to `1'b1`, the Offset Cancellation is controlled by the RX_OS_CFG attribute. When set to `1'b0`, the AGC is controlled by the RXOSHOLD signal.

Table 4-16 defines the CDR related attributes.

Table 4-16: **CDR Attributes**

Attribute	Type	Description
RXCDR_CFG	72-bit Hex	CDR configuration. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
RXCDR_LOCK_CFG	6-bit Binary	CDR Lock loop configuration. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.

Table 4-16: **CDR Attributes *(Cont'd)***

Attribute	Type	Description
RXCDR_HOLD_DURING_EIDLE	Binary	`1'b0`: Default setting. `1'b1`: Enables the CDR to hold its internal states during an optional reset sequence of an electrical idle state as used in PCI Express operation. ***Note:*** For channels with large attenuation (lossy channels typically exceeding 15 dB at Nyquist), it is recommended that RXCDR_HOLD_DURING_EIDLE be set to `1'b0` because fast transitioning data patterns like the 101010 sequence in CJPAT/CJTPAT can accidentally trigger an electrical idle.
RXCDR_FR_RESET_ON_EIDLE	Binary	`1'b0`: Default setting. `1'b1`: Enables automatic reset of CDR frequency during an optional reset sequence of an electrical idle state as used in PCI Express operation. ***Note:*** For channels with large attenuation (lossy channels typically exceeding 15 dB at Nyquist), it is recommended that RXCDR_FR_RESET_ON_EIDLE should be set to `1'b0` because fast transitioning data patterns like the 101010 sequence in CJPAT/CJTPAT can accidentally trigger an electrical idle.
RXCDR_PH_RESET_ON_EIDLE	Binary	`1'b0`: Default setting. `1'b1`: Enables automatic reset of CDR phase during an optional reset sequence of an electrical idle state as used in PCI Express operation. ***Note:*** For channels with large attenuation (lossy channels typically exceeding 15 dB at Nyquist), it is recommended that RXCDR_PH_RESET_ON_EIDLE should be set to `1'b0` because fast transitioning data patterns like the 101010 sequence in CJPAT/CJTPAT can accidentally trigger an electrical idle.
RX_OS_CFG	13-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.

RXMONITORSEL[1:0] selects outputting adaptation values (AGC, UT, and VP) on RXMONITOROUT[6:0]. Refer to RX Equalizer (DFE and LPM), page 176 for more details.

GTX/GTH Use Modes

RX CDR Lock to Reference

To get the CDR to lock to reference, set RXCDRHOLD = `1'b1` and RXCDROVRDEN = `1'b0`.

Dynamically Changing RX CDR Settings for Line Rate and Selected Protocol Changes

The following describes the sequence of events to dynamically change the RX CDR settings. It pertains only to changes for the CDR:

1. When ready (all valid data flushed out of receiver datapath), use the DRP to implement changes to the CDR loop filter settings with the attribute RXCDR_CFG[71:0] (GTX)/RXCDR_CFG[83:0] (GTH). Recommended settings for this attribute are provided in Table 4-17, Table 4-18, and Table 4-19.
2. Provide changes via ports QPLLREFCLKSEL/CPLLREFCLKSEL and/or the DRP to the attributes listed in Table 2-10 and Table 2-14.
3. Follow the reset guidelines as detailed in CPLL Reset, page 63 and QPLL Reset, page 63.
4. When the CPLL/QPLL has locked, assert GTRXRESET and follow the guidelines detailed in GTX/GTH Transceiver RX Reset in Response to GTRXRESET Pulse, page 79.
5. After the RXRESETDONE signal goes High, correct data must be verified before continuing with the operation of the transceiver (i.e., check a known data pattern).

Dynamically Changing RX CDR Settings to Tune CDR Loop Filter Settings Only

1. When ready (all valid data flushed out of receiver datapath), use the DRP to implement changes to the CDR loop filter settings with the attribute RXCDR_CFG[71:0] (GTX)/RXCDR_CFG[83:0] (GTH). Recommended settings for this attribute are provided in Table 4-17, Table 4-18, and Table 4-19.
2. Assert the GTRXRESET port and follow the guidelines detailed in GTX/GTH Transceiver RX Reset in Response to GTRXRESET Pulse, page 79.
3. After the RXRESETDONE signal goes High, correct data must be verified before continuing with the operation of the transceiver (i.e., check a known data pattern).

Table 4-17: **GTX CDR Recommended Settings for Scrambled/PRBS Data[(1)] (No SSC[(2)])**

PLL	LPM/DFE	Data Rate	RXOUT_DIV	REFCLK PPM	RXCDR_CFG
CPLL/QPLL	LPM	Greater than 6.6G	1	±200	72'h0B_0000_23FF_1040_0020
		Less than or equal to 6.6G	1		72'h03_0000_23FF_1020_0020
		Greater than 6.6G	1	±700	72'h0B_8000_23FF_1040_0020
		Less than or equal to 6.6G	1		72'h03_8000_23FF_1020_0020
		Greater than 6.6G	1	±1250	72'h0B_8000_23FF_1020_0020
		Less than or equal to 6.6G	1		72'h03_8000_23FF_1020_0020
	DFE	Greater than 6.6G	1	±200	72'h0B_0000_23FF_1040_0020
		Less than or equal to 6.6G	1		72'h03_0000_23FF_2040_0020
		Greater than 6.6G	1	±700	72'h0B_8000_23FF_1040_0020
		Less than or equal to 6.6G	1		72'h03_8000_23FF_2040_0020
		Greater than 6.6G	1	±1250	72'h0B_8000_23FF_1020_0020
		Less than or equal to 6.6G	1		72'h03_8000_23FF_1020_0020
CPLL/QPLL	LPM/DFE	1.6G - 6.25G	2	±200	72'h03_0000_23FF_4020_0020
			2		
			2	±700	72'h03_8000_23FF_4020_0020
			2		
			2	±1250	
			2		
CPLL/QPLL	LPM/DFE	0.8G-3.125G	4	±200	72'h03_0000_23FF_4010_0020
			4		
			4	±700	72'h03_8000_23FF_4010_0020
			4		
			4	±1250	
			4		

Table 4-17: **GTX CDR Recommended Settings for Scrambled/PRBS Data[1] (No SSC[2]) *(Cont'd)***

PLL	LPM/DFE	Data Rate	RXOUT_DIV	REFCLK PPM	RXCDR_CFG
CPLL/QPLL	LPM/DFE	0.5G-1.5625G	8	±200	72'h03_0000_23FF_4008_0020
			8		
			8	±700	72'h03_8000_23FF_4008_0020
			8		
			8	±1250	
			8		

Notes:

1. For protocol-specific settings, use the recommended value from the 7 Series FPGAs Transceivers Wizard and/or the protocol characterization reports.
2. Spread-spectrum clocking (SSC) is used to reduce the spectral density of electromagnetic interference (EMI).

Table 4-18: **GTX CDR Recommended Settings for Protocols with SSC**

PLL	LPM/DFE	Data Rate	RXOUT_DIV	REFCLK PPM with SSC	RXCDR_CFG
CPLL/QPLL	LPM/DFE	Less than or equal to 6.6G	1	±700 FM 33 KHz Triangular	72'h03_8000_23FF_1040_0008

Table 4-19: **GTX CDR Recommended Settings for 8B/10B Encoded Data[1] (No SSC[2])**

PLL	LPM/DFE	Data Rate	RXOUT_DIV	REFCLK PPM	RXCDR_CFG
CPLL/QPLL	LPM	Less than or equal to 6.6G	1	±200	72'h03_0000_23FF_1040_0020
			1		
			1	±700	72'h03_8000_23FF_1040_0020
			1		
			1	±1250	
			1		
CPLL/QPLL	LPM	1.6G - 6.25G	2	±200	72'h03_0000_23FF_1020_0020
			2		
			2	±700	72'h03_8000_23FF_1020_0020
			2		
			2	±1250	
			2		

Table 4-19: **GTX CDR Recommended Settings for 8B/10B Encoded Data[1] (No SSC[2]) *(Cont'd)***

PLL	LPM/DFE	Data Rate	RXOUT_DIV	REFCLK PPM	RXCDR_CFG
CPLL/QPLL	LPM	0.8G-3.125G	4	±200	72'h03_0000_23FF_1010_0020
			4		
			4	±700	72'h03_8000_23FF_1010_0020
			4		
			4	±1250	
			4		
CPLL/QPLL	LPM	0.5G-1.5625G	8	±200	72'h03_0000_23FF_1008_0020
			8		
			8	±700	72'h03_8000_23FF_1008_0020
			8		
			8	±1250	
			8		

Notes:

1. For protocol-specific settings, use the recommended value from the 7 Series FPGAs Transceivers Wizard and/or the protocol characterization reports.
2. Spread-spectrum clocking (SSC) is used to reduce the spectral density of electromagnetic interference (EMI).
3. RX_DEBUG_CFG is `12'h000` for all settings.

Table 4-20: **GTH CDR Recommended Settings for Scrambled/PRBS Data[1] (No SSC[2])**

PLL	LPM/DFE	Data Rate	RXOUT_DIV	REFCLK PPM	RXCDR_CFG
CPLL/QPLL	LPM	Greater than 8G	1	±200	83'h0_0020_07FE_2000_C208_001A
		Less than or equal to 8G	1		83'h0_0020_07FE_2000_C208_0018
		Greater than 8G	1	±700	83'h0_0020_07FE_2000_C208_801A
		Less than or equal to 8G	1		83'h0_0020_07FE_2000_C208_8018
		Greater than 8G	1	±1250	83'h0_0020_07FE_1000_C208_801A
		Less than or equal to 8G	1		83'h0_0020_07FE_1000_C208_8018
	DFE	Greater than 8G	1	±200	83'h0_0020_07FE_2000_C208_001A
		Less than or equal to 8G	1		83'h0_0020_07FE_2000_C208_0018
		Greater than 8G	1	±700	83'h0_0020_07FE_2000_C208_801A
		Less than or equal to 8G	1		83'h0_0020_07FE_2000_C208_8018
		Greater than 8G	1	±1250	83'h0_0020_07FE_1000_C208_801A
		Less than or equal to 8G	1		83'h0_0020_07FE_1000_C208_8018

Table 4-20: **GTH CDR Recommended Settings for Scrambled/PRBS Data[1] (No SSC[2]) *(Cont'd)***

PLL	LPM/DFE	Data Rate	RXOUT_DIV	REFCLK PPM	RXCDR_CFG
CPLL/QPLL	LPM/DFE	1.6G–6.55G	2	±200	83'h0_0020_07FE_1000_C220_0018
			2		
			2	±700	83'h0_0020_07FE_1000_C220_8018
			2		
			2	±1250	
			2		
CPLL/QPLL	LPM/DFE	0.8G–3.275G	4	±200	83'h0_0020_07FE_0800_C220_0018
			4		
			4	±700	83'h0_0020_07FE_0800_C220_8018
			4		
			4	±1250	
			4		
CPLL/QPLL	LPM/DFE	0.5G–1.6375G	8	±200	83'h0_0020_07FE_0400_C220_0018
			8		
			8	±700	83'h0_0020_07FE_0400_C220_8018
			8		
			8	±1250	
			8		

Notes:

1. For protocol-specific settings, use the recommended value from the 7 Series FPGAs Transceivers Wizard and/or the protocol characterization reports.
2. Spread-spectrum clocking (SSC) is used to reduce the spectral density of electromagnetic interference (EMI).

Table 4-21: **GTH CDR Recommended Settings for 8B/10B Encoded Data[1] (No SSC[2])**

PLL	LPM/DFE	Data Rate	RXOUT_DIV	REFCLK PPM	RX_CDR_CFG	RX_DEBUG_CFG
CPLL/QPLL	LPM	Less than or equal to 6.6G	1	±200	83'h0_0020_07FE_2000_C208_0018	14'h000
			1			14'h000
			1	±700	83'h0_0020_07FE_2000_C208_8018	14'h000
			1			14'h000
			1	±1250		14'h000
			1			14'h000
CPLL/QPLL	LPM	1.6G–6.55G	2	±200	83'h0_0020_07FE_1000_C208_0018	14'h000
			2			14'h000
			2	±700	83'h0_0020_07FE_1000_C208_8018	14'h000
			2			14'h000
			2	±1250		14'h000
			2			14'h000
CPLL/QPLL	LPM	0.8G–3.275G	4	±200	83'h0_0020_07FE_0800_C208_0018	14'h000
			4			14'h000
			4	±700	83'h0_0020_07FE_0800_C208_8018	14'h000
			4			14'h000
			4	±1250		14'h000
			4			14'h000
CPLL/QPLL	LPM	0.5G – 1.6375G	8	±200	83'h0_0020_07FE_0400_C208_0018	14'h000
			8			14'h000
			8	±700	83'h0_0020_07FE_0400_C208_8018	14'h000
			8			14'h000
			8	±1250		14'h000
			8			14'h000

Notes:

1. For protocol-specific settings, use the recommended value from the 7 Series FPGAs Transceivers Wizard and/or the protocol characterization reports.
2. Spread-spectrum clocking (SSC) is used to reduce the spectral density of electromagnetic interference (EMI).

RX Fabric Clock Output Control

Functional Description

The RX clock divider control block has two main components: serial clock divider control and parallel clock divider and selector control. The clock divider and selector details are

illustrated in Figure 4-22.

Figure 4-22: **RX Serial and Parallel Clock Divider**

Note relevant to Figure 4-22:

1. RXOUTCLKPCS and RXOUTCLKFABRIC are redundant outputs. RXOUTCLK should be used for new designs.
2. The REFCLK_CTRL option is controlled automatically by software and is not user selectable. The user can only route one of IBUFDS_GTE2's O or ODIV2 outputs to the FPGA logic via the CMT (PLL, MMCM or BUFMRCE), BUFH, or BUFG.
3. IBUFDS_GTE2 is a redundant output for additional clocking scheme flexibility.
4. There is only one CPLL in the GTXE2_CHANNEL/GTHE2_CHANNEL. QPLL from the GTXE2_COMMON/GTHE2_COMMON can also be used when applicable.
5. The selection of the /4 or /5 divider block is controlled by the RX_DATA_WIDTH attribute from the GTXE2_CHANNEL/GTHE2_CHANNEL primitive. /4 is selected when RX_DATA_WIDTH = 16, 32, or 64. /5 is selected when RX_DATA_WIDTH = 20, 40, or 80.
6. The selection of the /2 or /4 divider block is controlled by the RX_INT_DATAWIDTH attribute from the GTXE2_CHANNEL/GTHE2_CHANNEL primitive. /2 is selected when RX_INT_DATAWIDTH = 0 (2-byte internal datapath) and /4 is selected when RX_INT_DATAWIDTH = 1 (4-byte internal datapath).

7. For details about placement constraints and restrictions on clocking resources (MMCM, IBUFDS_GTE2, BUFG, etc.), refer to UG472, *7 Series FPGAs Clocking Resources User Guide*.

Serial Clock Divider

Each transmitter PMA module has a D divider that divides down the clock from the PLL for lower line rate support. This serial clock divider, D, can be set statically for applications with a fixed line rate or it can be changed dynamically for protocols with multiple line rates. The control for the serial divider is described in Table 4-22. For details about the line rate range per speed grade, refer to the 7 series FPGAs documentation page for the appropriate data sheet.

To use the D divider in fixed line rate applications, the RXOUT_DIV attribute must be set to the appropriate value, and the RXRATE port needs to be tied to `3'b000`. Refer to the Static Setting via Attribute column in Table 4-22 for details.

To use the D divider in multiple line rate applications, the RXRATE port is used to dynamically select the D divider value. The RXOUT_DIV attribute and the RXRATE port must select the same D divider value upon device configuration. After device configuration, the RXRATE is used to dynamically change the D divider value. Refer to the Dynamic Control via Ports column in Table 4-22 for details.

Table 4-22: **RX PLL Output Divider Setting**

D Divider Value	Static Setting via Attribute	Dynamic Control via Ports
1	RXOUT_DIV = 1 RXRATE = `3'b000`	RXOUT_DIV = Ignored RXRATE = `3'b001`
2	RXOUT_DIV = 2 RXRATE = `3'b000`	RXOUT_DIV = Ignored RXRATE = `3'b010`
4	RXOUT_DIV = 4 RXRATE = `3'b000`	RXOUT_DIV = Ignored RXRATE = `3'b011`
8	RXOUT_DIV = 8 RXRATE = `3'b000`	RXOUT_DIV = Ignored RXRATE = `3'b100`
16	RXOUT_DIV = 16 RXRATE = `3'b000`	RXOUT_DIV = Ignored RXRATE = `3'b101`

Parallel Clock Divider and Selector

The parallel clock outputs from the RX clock divider control block can be used as a fabric logic clock depending on the line rate and protocol requirements.

The recommended clock for the FPGA logic is the RXOUTCLK from one of the GTX/GTH transceivers. It is also possible to bring the MGTREFCLK directly to the fabric and use as the fabric clock. RXOUTCLK is preferred for general applications because it has an output delay control used for applications that bypass the RX buffer for constant datapath delay. Refer to RX Buffer Bypass, page 233 for more details.

The RXOUTCLKSEL port controls the input selector and allows these clocks to be output via TXOUTCLK port:

- RXOUTCLKSEL = `3'b001`: RXOUTCLKPCS path is not recommended to be used as it incurs extra delay from the PCS block.

- RXOUTCLKSEL = `3'b010`: RXOUTCLKPMA is the recovered clock that can be brought out to the FPGA logic. The recovered clock is used by protocols that do not have a clock compensation mechanism and require to use a clock synchronous to the data (the recovered clock), to clock the downstream fabric logic. It is also used by the RX PCS block. This clock is interrupted when the PLL or CDR is reset by one of the related reset signals.
- RXOUTCLKSEL = `3'b011` or `3'b100`: RXPLLREFCLK_DIV1 or RXPLLREFCLK_DIV2 is the input reference clock to the CPLL or QPLL depending on the RXSYSCLKSEL[1] setting. For usages that do not require outputting a recovered clock to the fabric, RXPLLREFCLK_DIV1 or RXPLLREFCLK_DIV2 can be used as the system clock. However, TXOUTCLK is usually used as system clock.

Ports and Attributes

Table 4-23 defines the ports required for RX fabric clock output control.

Table 4-23: **RX Fabric Clock Output Control Ports**

Port	Dir	Clock Domain	Description
RXOUTCLKSEL[2:0]	In	Async	This port controls the multiplexer select signal in Figure 4-22. `3'b000`: Static 1 `3'b001`: RXOUTCLKPCS path `3'b010`: RXOUTCLKPMA path `3'b011`: RXPLLREFCLK_DIV1 path `3'b100`: RXPLLREFCLK_DIV2 path Others: Reserved.
RXRATE[2:0]	In	RXUSRCLK2	This port dynamically controls the setting for the RX serial clock divider D (see Table 4-22) and it is used with RXOUT_DIV attribute. `3'b000`: Use RXOUT_DIV divider value `3'b001`: Set D divider to 1 `3'b010`: Set D divider to 2 `3'b011`: Set D divider to 4 `3'b100`: Set D divider to 8 `3'b101`: Set D divider to 16
RXOUTCLKFABRIC	Out	Clock	RXOUTCLKFABRIC is a redundant output reserved for testing. RXOUTCLK with RXOUTCLKSEL = `3'b011` should be used instead.
RXOUTCLK	Out	Clock	RXOUTCLK is the recommended clock output to the FPGA logic. The RXOUTCLKSEL port is the input selector for RXOUTCLK and allows the PLL input reference clock to the FPGA logic.

Table 4-23: **RX Fabric Clock Output Control Ports *(Cont'd)***

Port	Dir	Clock Domain	Description
RXOUTCLKPCS	Out	Clock	RXOUTCLKPCS is a redundant output. RXOUTCLK with RXOUTCLKSEL = `3'b001` should be used instead.
RXRATEDONE	Out	RXUSRCLK2	The RXRATEDONE port is asserted High for one RXUSRCLK2 cycle in response to a change on the RXRATE port. The TRANS_TIME_RATE attribute defines the period of time between a change on the RXRATE port and the assertion of RXRATEDONE.
RXDLYBYPASS	In	Async	RX delay alignment bypass: 0: Uses the RX delay alignment circuit. Set to `1'b0` when the RX buffer is bypassed. 1: Bypasses the RX delay alignment circuit. Set to `1'b1` when the RX buffer is used.

Table 4-24 defines the attributes required for RX fabric clock output control.

Table 4-24: **RX Fabric Clock Output Control Attributes**

Attribute	Type	Description
TRANS_TIME_RATE	8-bit Hex	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used. This attribute determines when PHYSTATUS and RXRATEDONE are asserted after a rate change.
RXBUF_RESET_ON_RATE_CHANGE	Boolean	When set to TRUE, this attribute enables automatic RX buffer reset during a rate change event initiated by a change in RXRATE.
RXOUT_DIV	Integer	This attribute controls the setting for the RX serial clock divider. This attribute is only valid when RXRATE = `3'b000`. Otherwise the D divider value is controlled by RXRATE. Valid settings are 1, 2, 4, 8, and 16.

Using RXRATE (GTH Transceiver Only)

When users want to change the divider D setting via RXRATE, the steps in Figure 4-23 should be performed if the GTH transceiver is changing into a configuration as follows:

- RXOUT_DIV != 1 and
- RX internal data width is 20 or 40-bit (RX_DATA_WIDTH = 20, 40, or 80)

Figure 4-23: **RXRATE Change Example**

Notes relevant to Figure 4-23:

1. DRP wr denotes the function of performing a DRP write to address `9'h011`. The exact DRP transaction is not shown.
2. The sequence of events in Figure 4-23 is not drawn to scale.
3. To change RXRATE, a DRP write should be issued to the GTHE2_CHANNEL primitive, DRPADDR `9'h011`, and bit[11] should be set to `1'b0`.

   To ensure only bit[11] of DRPADDR `9'h011` is modified, it is best to perform a read-modify-write function.
4. Upon DRP write completion, the value of RXRATE should be changed to the new desired setting.
5. The user should wait for RXPMARESETDONE to be detected Low.
6. A DRP write should be issued to the GTHE2_CHANNEL primitive, DRPADDR `9'h011`, restoring the original setting for bit[11]. The completion of this DRP write must occur before RXPMARESETDONE switches from Low to High. RXPMARESETDONE stays Low for a minimum of 0.66 us.
7. The sequence above simulates correctly if SIM_GTRESET_SPEEDUP is set to FALSE and the GT functional simulation model in the UniSims library is used. If SIM_GTRESET_SPEEDUP is set to TRUE or if the GT functional simulation model in the unifast library is used, the above sequence should be bypassed.

   Note: For GTH transceivers changing into a configuration of RXOUT_DIV = 1 or RX_DATA_WIDTH = 16, 32, or 64, the steps above are allowed but not required.

RX Margin Analysis

Functional Description

As line rates and channel attenuation increase, the receiver equalizers are more often enabled to overcome channel attenuation. This poses a challenge to system bring-up because the quality of the link cannot be determined by measuring the far-end eye opening at the receiver pins. At high line rates, the received eye measured on the printed circuit board can appear to be completely closed even though the internal eye after the receiver equalizer is open.

The 7 series FPGAs GTX/GTH transceivers RX eye scan provides a mechanism to measure and visualize the receiver eye margin after the equalizer. Additional use modes enable several other methods to determine and diagnose the effects of equalization settings.

Figure 4-24: **Offset Sample and Data Sample to Calculate BER as a Function of Offset—the Statistical Eye**

Eye Scan Theory

RXDATA is recovered from the equalized differential waveform by sampling after the RX equalizer. The horizontal sampling position is determined by the CDR function and the vertical position is differential zero. This is indicated as data sample in Figure 4-24.

To enable eye scan functionality, an additional sampler is provided with programmable (horizontal and vertical) offsets from the data sample point. This is indicated as offset sample in Figure 4-24.

A single eye scan measurement consists of accumulating the number of data samples (sample count) and the number of times that the offset sample disagreed with the data sample (error count). The bit error ratio (BER) at the programmed vertical and horizontal offset is the ratio of the error count to the sample count. The sample count can range from tens of thousands to greater than 10^{14}.

Repeating such BER measurements for the full array of horizontal and vertical offsets (or a subsampled set of offsets) produces a BER map as shown in Figure 4-24, commonly referred to as a *statistical eye*, where the color map represents $\log_{10}$(BER). In this view, the eye is apparently smaller than a traditional oscilloscope view (as in Figure 4-24) because it has been closed by very low probability jitter and noise that does not show up in the much lower number of samples of an oscilloscope.

Because this functionality puts no restrictions on the data patterns being received nor requires any changes in the RX settings, it can be performed while application data is being received without error. Furthermore, no FPGA logic is required—only the ability to read and write attributes.

Eye Scan Architecture

The blocks with shaded gray in Figure 4-25 describe the portion of the PMA architecture that supports eye scan. The horizontal offset (HORZ_OFFSET) advances or delays the sampling time of the offset samples relative to the data samples. The vertical offset

(VERT_OFFSET) raises or lowers the differential voltage threshold to which the equalized waveform is compared. The data samples are deserialized into the Rdata bus, and the offset samples are deserialized into the Sdata bus.

When in DFE mode (RXLPMEN=0), due to the *unrolled* first DFE tap, two separate eye scan measurements are needed, one at +UT and one at –UT, to measure the TOTAL BER at a given vertical and horizontal offset.

Figure 4-25: **PMA Architecture to Support Eye Scan**

Figure 4-26 describes the portion of the PCS architecture that supports eye scan. The 40-bit Rdata bus contains the data samples, and each bit of the 40-bit Sdata bus is one if and only if the corresponding data sample and offset sample are not equal. (See ES_ERRDET_EN in Table 4-26, page 210.)

Figure 4-26: **PCS Architecture to Support Eye Scan**

Two consecutive cycles of Sdata are masked by ES_SDATA_MASK[79:0] (i.e., bit-by-bit Sdata[i] AND NOT mask[i]). The algebraic sum of bits [39:0] of this result is the number of errors to be added in the error counter.

Two consecutive cycles of Rdata are compared with the pattern in ES_QUALIFIER[79:0], and that result is masked by (i.e., bit-by-bit ORed with) ES_QUAL_MASK[79:0]. The logical AND of this result determines whether the prescaler/sample counter is incremented and the errors added to the error counter. For a statistical eye, ES_QUAL_MASK is `80'b1`, so the sample counter and error counter accumulate on every cycle. ES_SDATA_MASK unmasks only the current data (bit 39 and below; see the description of RX_INT_DATAWIDTH) to avoid double counting errors because they appear first in the lower 40 bits and then in the upper 40 bits on the next cycle.

Alternate use modes produce scope-like displays by unmasking a sequence of Rdata bits (up to 40), causing error and sample accumulation only if Rdata matches ES_QUALIFIER in that range of bits. In these use modes, only one Sdata bit per measurement is unmasked. In diagnostic use modes, Rdata and Sdata are *frozen* and can be read out through the DRP interface when:

- An error occurs,
- A count qualifier occurs,

- A fabric port causes a trigger, or
- A trigger is forced via an attribute write.

The diagnostic use modes could be used, for example, to examine the pattern of burst errors due to DFE behavior.

Figure 4-27 documents the state transitions in the eye scan state machine.

Figure 4-27: **Eye Scan State Machine**

ES_CONTROL[1:0] are the signals arm and run, respectively. From the WAIT state, run initiates the BER measurement loop (left) and arm starts the diagnostic loop (right).

The RESET state zeros the error and sample counters, then enters the COUNT state or the ARMED state (depending on whether run or arm is active).

In the COUNT state, samples and errors are accumulated in the counters. When either counter is saturated, both counters stop and transition to the END state. This transition to the END state is detected by polling es_control_status[3:0]. Bit 0 (done) is set active only in the END, READ, and WAIT states. Bits [3:1] display the current state of the state machine.

The END state transitions to the WAIT state when run is set back to zero. The es_sample_count[15:0] and es_error_count[15:0] can be read either in the END or WAIT state.

In the ARMED state, the FIFOs (successive cycles of Rdata and of Sdata) are stopped when a trigger event occurs. The trigger event is either the count qualifier pulse, the logical OR of all bits into the error counter, or a manual trigger provided from a DRP data input or from a port. One of these four options is selected by trig[3:0] = ES_CONTROL[5:2].

In the READ state, the last two cycles of Rdata can be read from the COE status register, es_rdata[79:0], and the last two cycles of Sdata can be read from the COE status register, es_sdata[79:0].

Ports and Attributes

Table 4-25 defines ports related to the RX eye scan function.

Table 4-25: **RX Margin Analysis Ports**

Port	Dir	Domain	Description
EYESCANDATAERROR	Out	Async	Asserts high for one REC_CLK cycle when an (unmasked) error occurs while in the COUNT or ARMED state.
EYESCANTRIGGER	In	RXUSRCLK2	Causes a trigger event. See ES_CONTROL[4] below.
RXRATE	In	RXUSRCLK2	Dynamic pins to automatically change effective PLL dividers in RX. Used for PCIe and other standards. `00` will use RXOUT_DIV attributes `01` implies /4 `10` implies /2 `11` implies /1
RXLPMEN	In	Async	When set to `1'b1`, the LPM mode with the adaptive linear equalizer is enabled. When set to 1'b0, the high-performance DFE mode is enabled.
EYESCANMODE	In	Async	Reserved.

Table 4-26 defines RX eye scan attributes. Lower case attribute names indicate R/O.

Table 4-26: **RX Margin Analysis Attributes**

Attribute	Type	Description
ES_VERT_OFFSET	9-bit Binary	Controls the vertical (differential voltage) offset of the scan sample: [6:0]: Offset magnitude (centered on ±UT, the unwrapped threshold). [7]: Offset sign (1 is negative, 0 is positive). [8]: UT sign (1 selects the negative unwrapped threshold, 0 the positive threshold).

ES_HORZ_OFFSET	12-bit Hex	Controls the horizontal (phase) offset of the scan sample.
		[10:0]: Phase offset (two's complement). The center of data eye (0 UI) corresponds to a count of `11'd0` for all data rates. The table below lists the minimum count (representing -0.5 UI) and maximum count (representing +0.5 UI) for each data rate.
		`Rate  min count [dec(bin)]  eye center [dec(bin)]  max count [dec(bin)]`
		`Full  -32 (11'b11111100000)  +0(11'b00000000000)  +32(11'b00000100000)`
		`Half  -64 (11'b11111000000)  +0(11'b00000000000)  +64(11'b00001000000)`
		`Qrtr  -128 (11'b11110000000)  +0(11'b00000000000)  +128(11'b00010000000)`
		`Octal  -256 (11'b11100000000)  +0(11'b00000000000)  +256(11'b00100000000)`
		`Hex  -512 (11'b11000000000)  +0(11'b00000000000)  +512(11'b01000000000)`
		[11]: Phase unification. Must be set to 0 for all positive counts (including zero) and to 1 for all negative counts.

Table 4-26: **RX Margin Analysis Attributes *(Cont'd)***

Attribute	Type	Description
ES_PRESCALE	5-bit Binary	Controls the pre-scaling of the sample count to keep both sample count and error count in reasonable precision within the 16-bit register range. Prescale = $2^{(1 + \text{register value})}$, so minimum prescale is $2^{(1+0)}$ = 2 and maximum prescale is $2^{(1+31)}$ = 4,284,967,296.
ES_SDATA_MASK	80-bit Hex	This attribute masks up to two cycles of the 40-bit Sdata bus. Binary 1 causes the corresponding bus bit to be masked and binary 0 leaves it unmasked. To support the statistical eye view, the error counter accumulates the total number of unmasked 1s on the most recent cycle of the Sdata bus (masked by ES_SDATA_MASK[39:0]). To support the scope and waveform views, the error counter increments by only one for any non-zero number of unmasked 1s on the previous cycle of the Sdata bus (masked by ES_SDATA_MASK[79:40]). This attribute and ES_QUAL_MASK must also mask out unused bits for bus widths narrower than 40 bits. For the statistical eye view, this attribute would assume the following values as a function of bus width: 40-bit width: ES_SDATA_MASK = (`{40{1'b1}}, {40{1'b0}}`) 32-bit width: ES_SDATA_MASK = (`{40{1'b1}}, {32{1'b0}}, {8{1'b1}}`) 20-bit width: ES_SDATA_MASK = (`{40{1'b1}}, {20{1'b0}}, {20{1'b1}}`) 16-bit width: ES_SDATA_MASK = (`{40{1'b1}}, {16{1'b0}}, {24{1'b1}}`) Scope and waveform views require a sequence of measurements, unmasking only a single bit per measurement.
ES_QUALIFIER	80-bit Hex	Eye scan can qualify BER measurements based on patterns up to 40 contiguous bits long in any position in the input data. Because the data, and therefore the qualifier pattern, is not aligned, the position of the pattern must be discovered by a barrel-shifting search. For example, looking for the pattern `10'b0011111010` (K28.5 in 8B/10B code) with a 20-bit data width would require a sequence of measurements such as the following, searching for a non-zero sample count at the correct alignment: ES_QUALIFIER = (`{50{1'b?}}, 10'b0011111010, {20{1'b?}}`) ES_QUALIFIER = (`{49{1'b?}}, 10'b0011111010, {21{1'b?}}`) ES_QUALIFIER = (`{48{1'b?}}, 10'b0011111010, {22{1'b?}}`) …etc… (where ? represents a DON'T CARE bit that will be masked) The qualifier pattern is shifted only over the valid bits for the bus width (40, 32, 20, or 16). See the description of RX_INT_DATAWIDTH.
ES_QUAL_MASK	80-bit Hex	This attribute masks those bits not included in the qualifier pattern. For example, the corresponding values for the K28.5 example above would be: ES_QUAL_MASK = (`{50{1'b1}}, {10{1'b0}}, {20{1'b1}}`) ES_QUAL_MASK = (`{49{1'b1}}, {10{1'b0}}, {21{1'b1}}`) ES_QUAL_MASK = (`{48{1'b1}}, {10{1'b0}}, {22{1'b1}}`) …etc…
PMA_RSV2[5]	1-bit Binary	GTX transceiver: This bit should always be 1 when using Eye Scan. Setting this bit to 0 powers down the Eye Scan circuitry in the PMA and forces the Eye Scan state to WAIT. Re-enabling Eye Scan functionality requires reasserting this bit and asserting/deasserting PMA reset. GTH transceiver: Reserved. Unused.

Table 4-26: **RX Margin Analysis Attributes *(Cont'd)***

Attribute	Type	Description
ES_EYE_SCAN_EN	1-bit Binary	GTX transceiver: Reserved. This bit should always be asserted. GTH transceiver: This bit should always be 1 when using Eye Scan. Setting this bit to 0 powers down the Eye Scan circuitry in the PMA and forces the Eye Scan state to WAIT. Re-enabling Eye Scan functionality requires reasserting this bit and asserting/deasserting PMA reset.
ES_ERRDET_EN	1-bit Binary	1: Each bit of the Sdata bus is 1 if and only if the corresponding offset data sample does not agree with the recovered data sample. This is used for the statistical eye view. 0: Each bit of the Sdata bus is the recovered data sample. Therefore, if no errors occurred, the Sdata bus would be identical to the Rdata bus. This is used for the scope and waveform views.
ES_CONTROL	6-bit Binary	[0]: RUN. Asserting this bit causes a state transition from the WAIT state to the RESET state, initiating a BER measurement sequence. [1]: ARM Asserting this bit causes a state transition from the WAIT state to the RESET state, initiating a diagnostic sequence. In the ARMED state, deasserting this bit causes a state transition to the READ state if one of the states of bits [5:2] below is not met. [5:2]: `0001` In the ARMED state, causes a trigger event (transition to the READ state) when an error is detected (i.e., an unmasked 1 on the Sdata bus). `0010` In the ARMED state, causes a trigger event (transition to the READ state) when the qualifier pattern is detected in Rdata. `0100` In the ARMED state, causes a trigger event (transition to the READ state) when the eye_scan_trigger port asserts High. `1000` In the ARMED state, causes a trigger event (transition to the READ state) immediately.

es_control_status	4-bit Binary	[0]: DONE. Asserted High only in the WAIT, END, or READ states.
		[3:1]: Current state of the state machine:
		`WAIT 000`
		`RESET 001`
		`COUN 011`
		`END 010`
		`ARMED 101`
		`READ 100`

es_rdata	80-bit Binary	When a trigger event occurs in the ARMED state, es_rdata[39:0] is the present state of the Rdata bus and es_rdata[79:40] is the previous state of the Rdata bus.
es_sdata	80-bit Binary	When a trigger event occurs in the ARMED state, es_sdata[39:0] is the present state of the Sdata bus and es_sdata[79:40] is the previous state of the Sdata bus.
es_error_count	16-bit Hex	In END and WAIT states, contains the final error count for the preceding BER measurement.
es_sample_count	16-bit Hex	In END and WAIT states, contains the final sample count for the preceding BER measurement.

Table 4-26: **RX Margin Analysis Attributes *(Cont'd)***

Attribute	Type	Description
RX_DATA_WIDTH	Integer	Sets the bit width of the RXDATA port. When 8B/10B encoding is enabled, RX_DATA_WIDTH must be set to 20, 40, or 80. Valid settings are 16, 20, 32, 40, 64, and 80. See Interface Width Configuration, page 287 for more details.
GTH transceiver: USE_PCS_CLK_PHASE_SEL	1-bit Binary	If set to 1, the Eye Scan 4T clock phase is determined by ES_CLK_PHASE_SEL. If set to 0, the deserializer phase detector determines the phase of the Eye Scan 4T clock.
GTH transceiver: ES_CLK_PHASE_SEL	1-bit Binary	If USE_PCS_CLK_PHASE_SEL is asserted, setting this bit to 1 selects one phase of the Eye Scan 4T clock. Setting it to 0 selects the other phase.

RX_INT_DATAWIDTH	Integer	1: 32- or 40-bit interface
		0: 16- or 20-bit interface
		
		Width of valid data on Rdata and Sdata buses is RX fabric data width (see RX_DATA_WIDTH) divided by $2^{(1-RX_INT_DATAWIDTH)}$.
		For the different possible bus widths, the previous and current valid Rdata and Sdata bits correspond to the following indices in ES_SDATA_MASK, ES_QUALIFIER, ES_QUAL_MASK, es_rdata, and es_sdata:
		
		`valid data width  previous data  current data`
		`16  [79:64]  [39:24]`
		`20  [79:60]  [39:20]`
		`32  [79:48]  [39: 8]`
		`40  [79:40]  [39: 0]`

RXOUT_DIV	Integer	QPLL/CPLL output clock divider D for the RX datapath as shown in Figure 2-9, page 46. Valid settings are 1, 2, 4, 8, and 16. This attribute sets the divider only if the RXRATE port is set to `3'b000`.
ES_PMA_CFG	1-bit Binary	Reserved.

Table 4-27: **DRP Address Map for Eye Scan Read-Only (R) Registers**

DRP Address Hex (GTX Transceiver)	DRP Address Hex (GTH Transceiver)	DRP Bits	R/W	Name	Attribute Bit
14F	151	15:0	R	es_error_count	15:0
150	152	15:0	R	es_sample_count	15:0
151	153	3:0	R	es_control_status	3:0
152	154	15:0	R	es_rdata	79:64
153	155	15:0	R	es_rdata	63:48
154	156	15:0	R	es_rdata	47:32
155	157	15:0	R	es_rdata	31:16
156	158	15:0	R	es_rdata	15:0
157	159	15:0	R	es_sdata	79:64
158	15A	15:0	R	es_sdata	63:48

Table 4-27: **DRP Address Map for Eye Scan Read-Only (R) Registers *(Cont'd)***

DRP Address Hex (GTX Transceiver)	DRP Address Hex (GTH Transceiver)	DRP Bits	R/W	Name	Attribute Bit
159	15B	15:0	R	es_sdata	47:32
15A	15C	15:0	R	es_sdata	31:16
15B	15D	15:0	R	es_sdata	15:0

RX Polarity Control

Functional Description

If RXP and RXN differential traces are accidentally swapped on the PCB, the differential data received by the GTX/GTH transceiver RX are reversed. The GTX/GTH transceiver RX allows inversion to be done on parallel bytes in the PCS after the SIPO to offset reversed polarity on differential pair. Polarity control function uses the RXPOLARITY input, which is driven High from the fabric user interface to invert the polarity.

Ports and Attributes

Table 4-28 defines the ports required by the RX polarity control function.

Table 4-28: **RX Polarity Control Ports**

Port	Dir	Clock Domain	Description
RXPOLARITY	In	RXUSRCLK2	The RXPOLARITY port can invert the polarity of incoming data: 0: Not inverted. RXP is positive and RXN is negative. 1: Inverted. RXP is negative and RXN is positive.

Using RX Polarity Control

RXPOLARITY can be tied High if the polarity of RXP and RXN needs to be reversed.

RX Pattern Checker

Functional Description

The GTX/GTH receiver includes a built-in PRBS checker (see Figure 4-28). This checker can be set to check for one of four industry-standard PRBS patterns. The checker is self-synchronizing and works on the incoming data before comma alignment or decoding. This function can be used to test the signal integrity of the channel.

Figure 4-28: **RX Pattern Checker Block**

Ports and Attributes

Table 4-29 defines the pattern checker ports.

Table 4-29: **Pattern Checker Ports**

Port	Dir	Clock Domain	Description
RXPRBSCNTRESET	In	RXUSRCLK2	Resets the PRBS error counter.
RXPRBSSEL[2:0]	In	RXUSRCLK2	Receiver PRBS checker test pattern control. Only these settings are valid: `000`: Standard operation mode. (PRBS check is off) `001`: PRBS-7 `010`: PRBS-15 `011`: PRBS- 23 `100`: PRBS-31 No checking is done for non-PRBS patterns. Single bit errors cause bursts of PRBS errors because the PRBS checker uses data from the current cycle to generate next cycle's expected data.
RXPRBSERR	Out	RXUSRCLK2	This non-sticky status output indicates that PRBS errors have occurred.

Table 4-30 defines the pattern checker attributes.

Table 4-30: **Pattern Checker Attributes**

Attribute	Type	Description
RX_PRBS_ERR_CNT	16-bit Binary	PRBS error counter. This counter can be reset by asserting RXPRBSCNTRESET. When an error(s) in incoming parallel data occurs, this counter increments by 1 and counts up to `0xFFFF`. This error counter can only be accessed via the DRP. GTX transceiver: The address for this counter is `0x15C`. GTH transceiver: The address for this counter is `0x15E`.
RXPRBS_ERR_LOOPBACK	1-bit Binary	When this attribute is set to 1, the RXPRBSERR bit is internally looped back to TXPRBSFORCEERR of the same GTX/GTH transceiver. This allows synchronous and asynchronous jitter tolerance testing without worrying about data clock domain crossing. When this attribute is set to 0, TXPRBSFORCEERR is forced onto the TX PRBS.

Use Models

To use the built-in PRBS checker, RXPRBSSEL must be set to match the PRBS pattern being sent to the receiver. The RXPRBSSEL entry in Table 4-29 shows the available settings. When the PRBS checker is running, it attempts to find the selected PRBS pattern in the incoming data. If the incoming data is inverted by the transmitter or reversed RXP/RXN, the received data should also inverted by controlling RXPOLARITY. Otherwise, the PRBS checker does not lock. When it finds the pattern, it can detect PRBS errors by comparing the incoming pattern with the expected pattern. The expected pattern is generated from the previous incoming data. The checker counts the number of word (20 bits per word) errors and increments the word error counter by 1 when an error(s) is found in the incoming parallel data. Thus the word error counter might not match the actual number of bit errors if the incoming parallel data contains two or more bit errors. The error counter stops counting when reaching `0xFFFF`.

When the error occurs, RXPRBSERR is asserted. When no error is found in the following incoming data, RXPRBSERR is cleared. Asserting RXPRBSCNTRESET clears the error counter. GTRXRESET and RXPCSRESET also reset the count.

Refer to TX Pattern Generator, page 137 for more information about use models.

RX Byte and Word Alignment

Functional Description

Serial data must be aligned to symbol boundaries before it can be used as parallel data. To make alignment possible, transmitters send a recognizable sequence, usually called a

comma. The receiver searches for the comma in the incoming data. When it finds a comma, it moves the comma to a byte boundary so the received parallel words match the transmitted parallel words.

Figure 4-29 shows the alignment to a 10-bit comma. The RX receiving unaligned bits are on the right side. The serial data with the comma is highlighted in the middle. Byte aligned RX parallel data is on the left.

Figure 4-29: **Conceptual View of Comma Alignment (Aligning to a 10-Bit Comma)**

Figure 4-30 shows TX parallel data on the left side, and RX receiving recognizable parallel data after comma alignment on the right side.

Figure 4-30: **Parallel Data View of Comma Alignment**

Enabling Comma Alignment

To enable the comma alignment block, the RXCOMMADETEN port is driven High. RXCOMMADETEN is driven Low to bypass the block completely for minimum latency.

Configuring Comma Patterns

To set the comma pattern that the block searches for in the incoming data stream, the ALIGN_MCOMMA_VALUE, ALIGN_PCOMMA_VALUE, and ALIGN_COMMA_ENABLE attributes are used. The comma lengths depend on RX_DATA_WIDTH (see Table 4-54, page 290). Figure 4-31 shows how the ALIGN_COMMA_ENABLE masks each of the comma values to allow partial pattern matching.

Figure 4-31: **Comma Pattern Masking**

Figure 4-32 shows how the commas are combined when ALIGN_COMMA_DOUBLE is TRUE.

Figure 4-32: **Extended Comma Pattern Definition**

Figure 4-33 shows how a comma is combined with ALIGN_COMMA_ENABLE to make a wild-carded comma for a 20-bit internal comma. If ALIGN_COMMA_DOUBLE is TRUE, the MCOMMA and PCOMMA patterns are combined so that the block searches for two commas in a row. The number of bits in the comma depends on RX_DATA_WIDTH. Either a 16-bit or a 20-bit comma alignment mode is possible. A double comma is only detected when the received data has a PCOMMA defined by ALIGN_PCOMMA_VALUE followed by an MCOMMA defined by ALIGN_MCOMMA_VALUE with no extra bits in between.

Figure 4-33: **Extended Comma Pattern Masking**

Activating Comma Alignment

Commas are aligned to the closest boundary providing they are found while comma alignment is active. RXMCOMMAALIGNEN is driven High to align on the MCOMMA pattern. RXPCOMMAALIGNEN is driven High to activate alignment on the PCOMMA

pattern. Both enable ports are driven to align to either pattern. When ALIGN_COMMA_DOUBLE is TRUE, both enable ports must always be driven to the same value.

Alignment Status Signals

While MCOMMA or PCOMMA alignment is active, any matching comma pattern causes the block to realign to the closest boundary. After successful alignment, the block holds RXBYTEISALIGNED High. At this time, RXMCOMMAALIGNEN and RXPCOMMAALIGNEN can be driven Low to turn off alignment and keep the current alignment position. RXPCOMMAALIGNEN must be TRUE for PCOMMAs to cause RXBYTEISALIGNED to go High. Similarly, RXMCOMMAALIGNEN must be TRUE for MCOMMAs to cause RXBYTEISALIGNED to go High. Commas can arrive while RXBYTEISALIGNED is High. If the commas arrive aligned to boundaries, there is no change. If the commas arrive out of position, the block deasserts RXBYTEISALIGNED until the commas are aligned again. If alignment is still activated for the comma that arrives, the block automatically aligns the new comma to the closest boundary and drives RXBYTEREALIGN High for one RXUSRCLK2 cycle.

In applications that operate at a line rate greater than 5 Gb/s and have excessive noise in the system, the byte align block might falsely align to a wrong byte boundary and falsely assert the RXBYTEISALIGNED signal when no valid data is present. In such applications, a system-level check should be in place for checking the validity of the RXBYTEISALIGNED indicator and data.

In systems that use the RX OOB block, such as PCIe and SATA, after locking to a valid byte boundary and asserting the RXBYTEISALIGNED signal, the byte align block might occasionally deassert the RXBYTEISALIGNED signal even when there is no change in the byte boundary. In such applications, RXBYTEISALIGNED should not be used as a valid indicator of the change in byte boundary after the first assertion.

Alignment Boundaries

The allowed boundaries for alignment are defined by ALIGN_COMMA_WORD and RX_INT_DATAWIDTH. The spacing of the possible boundaries is determined by RX_DATA_WIDTH, and the number of boundary positions is determined by the number of bytes in the RXDATA interface (refer to Table 4-50, page 288 for RX_DATA_WIDTH and RX_INT_DATAWIDTH settings). Figure 4-34 shows the boundaries that can be selected.

RX_DATA_WIDTH	RX_INT_DATAWIDTH	ALIGN_COMMA_WORD	Possible RX Alignments (Grey = Comma Can Appear on Byte)
16/20 (2-byte)	0 (2-byte)	1	Byte1 Byte0
16/20 (2-byte)	0 (2-byte)	2	Byte1 Byte0
16/20 (2-byte)	0 (2-byte)	4	Invalid Configuration
32/40 (4-byte)	0 (2-byte)	1	Byte3 Byte2 Byte1 Byte0
32/40 (4-byte)	0 (2-byte)	2	Byte3 Byte2 Byte1 Byte0
32/40 (4-byte)	0 (2-byte)	4	Invalid Configuration
32/40 (4-byte)	1 (4-byte)	1	Byte3 Byte2 Byte1 Byte0
32/40 (4-byte)	1 (4-byte)	2	Byte3 Byte2 Byte1 Byte0
32/40 (4-byte)	1 (4-byte)	4	Byte3 Byte2 Byte1 Byte0
64/80 (8-byte)	1 (4-byte)	1	Byte7 Byte6 Byte5 Byte4 Byte3 Byte2 Byte1 Byte0
64/80 (8-byte)	1 (4-byte)	2	Byte7 Byte6 Byte5 Byte4 Byte3 Byte2 Byte1 Byte0
64/80 (8-byte)	1 (4-byte)	4	Byte7 Byte6 Byte5 Byte4 Byte3 Byte2 Byte1 Byte0

UG476_c4_15_091610

Figure 4-34: **Comma Alignment Boundaries**

Manual Alignment

RXSLIDE can be used to override the automatic comma alignment and to shift the parallel data. RXSLIDE is driven High for one RXUSRCLK2 cycle to shift the parallel data by one bit. RXSLIDE must be Low for at least 32 RXUSRCLK2 cycles before it can be used again.

Figure 4-35 shows the waveforms for manual alignment using RXSLIDE in RXSLIDE_MODE = PCS, before and after the data shift. When RXSLIDE_MODE = PCS is used, the number of bit shift positions when consecutive RXSLIDE pulses are issued is also determined by the comma alignment boundary set by ALIGN_COMMA_WORD, RX_DATA_WIDTH, and RX_INT_DATAWIDTH. For example, if the RX_DATA_WIDTH is 20 bits and ALIGN_COMMA_WORD is 1, after the 9th slide operation, the slide position returns back to 0. For the same RX_DATA_WIDTH setting, for an ALIGN_COMMA_WORD setting of 2, the slide position returns to 0 after the 19th slide operation. Thus in RXSLIDE_MODE = PCS, a maximum of 40 bits of sliding is possible when RX_INT_DATAWIDTH= 1 (4-byte) and ALIGN_COMMA_WORD = 4.

Figure 4-35: **Manual Data Alignment Using RXSLIDE for RX_DATA_WIDTH = 20 Bits and RXSLIDE_MODE = PCS**

Note relevant to Figure 4-35:

1. Latency between the slide and the slide result at RXDATA depends on the number of active RX PCS blocks in the datapath.

Figure 4-36 shows the waveforms for manual alignment using RXSLIDE in RXSLIDE_MODE = PMA before and after the data shift. In this mode, the data is shifted right by one bit for every RXSLIDE pulse issued, but there is some intermediate data with the bits shifted left before the final data appears on the bus. When RXSLIDE_MODE = PMA is used, the RX recovered clock phase is shifted by 2 UI for every alternate RXSLIDE pulse.

Figure 4-36: **Manual Data Alignment Using RXSLIDE for RX_DATA_WIDTH = 20 Bits and RXSLIDE_MODE = PMA**

Note relevant to Figure 4-36:

1. Latency between the slide and the slide result at RXDATA depends on the number of active RX PCS blocks in the datapath.

Ports and Attributes

Table 4-31 defines the RX byte and word alignment ports.

Table 4-31: **RX Byte and Word Alignment Ports**

Port Name	Dir	Clock Domain	Description
RXBYTEISALIGNED	Out	RXUSRCLK2	This signal from the comma detection and realignment circuit is High to indicate that the parallel data stream is properly aligned on byte boundaries according to comma detection. 0: Parallel data stream not aligned to byte boundaries 1: Parallel data stream aligned to byte boundaries There are several cycles after RXBYTEISALIGNED is asserted before aligned data is available at the FPGA RX interface. RXBYTEISALIGNED responds to plus comma alignment when RXPCOMMAALIGNEN is TRUE. RXBYTEISALIGNED responds to minus comma alignment when RXMCOMMAALIGNEN is TRUE. Alignment Status Signals, page 219 describes some conditions when this signal could deviate from the expected behavior.
RXBYTEREALIGN	Out	RXUSRCLK2	This signal from the comma detection and realignment circuit indicates that the byte alignment within the serial data stream has changed due to comma detection. 0: Byte alignment has not changed 1: Byte alignment has changed Data can be lost or repeated when alignment occurs, which can cause data errors (and disparity errors when the 8B/10B decoder is used).
RXCOMMADET	Out	RXUSRCLK2	This signal is asserted when the comma alignment block detects a comma. The assertion occurs several cycles before the comma is available at the FPGA RX interface. 0: Comma not detected 1: Comma detected

Table 4-31: **RX Byte and Word Alignment Ports *(Cont'd)***

Port Name	Dir	Clock Domain	Description
RXCOMMADETEN	In	RXUSRCLK2	RXCOMMADETEN activates the comma detection and alignment circuit. 0: Bypass the circuit 1: Use the comma detection and alignment circuit Bypassing the comma and alignment circuit reduces RX datapath latency.
RXPCOMMAALIGNEN	In	RXUSRCLK2	Aligns the byte boundary when comma plus is detected. 0: Disabled 1: Enabled.
RXMCOMMAALIGNEN	In	RXUSRCLK2	Aligns the byte boundary when comma minus is detected. 0: Disabled 1: Enabled.
RXSLIDE	In	RXUSRCLK2	RXSLIDE implements a comma alignment bump control. When RXSLIDE is asserted, the byte alignment is adjusted by one bit, which permits determination and control of byte alignment by the FPGA logic. Each assertion of RXSLIDE causes just one adjustment. RXSLIDE must be deasserted for more than 32 RXUSRCLK2 cycles before it can be reasserted to cause another adjustment. When asserted, RXSLIDE takes precedence over normal comma alignment. For proper operation, the user should set these values: RXPCOMMAALIGNEN = 0; RXMCOMMAALIGNEN = 0; RXCOMMADETEN = 1; SHOW_REALIGN_COMMA = FALSE

Table 4-32 defines the RX byte and word alignment attributes.

Table 4-32: **RX Byte and Word Alignment Attributes**

Attribute	Type	Description
ALIGN_COMMA_WORD	Integer	This attribute controls the alignment of detected commas within a multi-byte datapath. 1: Align comma to either of the 2 bytes for a 2-byte interface, any of the 4 bytes for a 4-byte interface, any of the 8 bytes for an 8-byte interface. The comma can be aligned to either the even bytes or the odd bytes of RXDATA output. 2: Align comma to the even bytes only. The aligned comma is guaranteed to be aligned to even bytes RXDATA[9:0] for a 2-byte interface, RXDATA[9:0]/RXDATA[29:20] for a 4-byte interface, RXDATA[9:0]/RXDATA[29:20]/RX[49:40]/RX[69:60] for an 8-byte interface 4: Align comma to a 4-byte boundary. This setting is not allowed for RX_INT_DATAWIDTH = 0. The aligned comma is guaranteed to be aligned to RXDATA[9:0] for a 4-byte interface, and RXDATA[9:0]/RXDATA[49:40] for an 8-byte interface Refer to Figure 4-34, page 220 for comma alignment boundaries allowed for the different ALIGN_COMMA_WORD, RX_DATA_WIDTH and RX_INT_DATAWIDTH settings. Protocols that send commas in even and odd positions must set ALIGN_COMMA_WORD to 1.
ALIGN_COMMA_ENABLE	10-bit Binary	Sets which bits in MCOMMA/PCOMMA must be matched to incoming data and which bits can be of any value. This attribute is a 10-bit mask with a default value of `1111111111`. Any bit in the mask that is reset to 0 effectively turns the corresponding bit in MCOMMA or PCOMMA to a don't care bit.

Table 4-32: **RX Byte and Word Alignment Attributes *(Cont'd)***

Attribute	Type	Description
ALIGN_COMMA_DOUBLE	Boolean	Specifies whether a comma match consists of either a comma plus or a comma minus alone, or if both are required in the sequence. FALSE: The plus comma (PCOMMA) and minus comma (MCOMMA) are handled separately. An individual match for either can lead to comma detection and alignment. TRUE: A comma match consists of a comma plus followed immediately by a comma minus. The match pattern is 20 or 16 bits (as determined by RX_DATA_WIDTH). When ALIGN_COMMA_DOUBLE is TRUE, ALIGN_PCOMMA_DET must be the same as ALIGN_MCOMMA_DET, and RXPCOMMAALIGNEN must be the same as RXMCOMMAALIGNEN.
ALIGN_MCOMMA_VALUE	10-bit Binary	Defines comma minus to raise RXCOMMADET and align the parallel data. The reception order is right to left. (ALIGN_MCOMMA_VALUE [0] is received first.) The default value is `10'b1010000011` (K28.5). This definition does not affect 8B/10B encoding or decoding.
ALIGN_MCOMMA_DET	Boolean	Controls the raising of RXCOMMADET on comma minus. FALSE: Do not raise RXCOMMADET when comma minus is detected. TRUE: Raise RXCOMMADET when comma minus is detected. (This setting does not affect comma alignment.)
ALIGN_PCOMMA_VALUE	10-bit Binary	Defines comma plus to raise RXCOMMADET and align parallel data. The reception order is right to left. (ALIGN_PCOMMA_VALUE [0] is received first.) The default value is `10'b0101111100` (K28.5). This definition does not affect 8B/10B encoding or decoding.
ALIGN_PCOMMA_DET	Boolean	Controls the raising of RXCOMMADET on comma plus. FALSE: Do not raise RXCOMMADET when comma plus is detected. TRUE: Raise RXCOMMADET when comma plus is detected. (This setting does not affect comma alignment.)

Table 4-32: **RX Byte and Word Alignment Attributes *(Cont'd)***

Attribute	Type	Description
SHOW_REALIGN_COMMA	Boolean	Defines if a comma that caused realignment is brought out to the FPGA RX. FALSE: Do not bring the comma that causes realignment to the FPGA RX. This setting reduces RX datapath latency TRUE: Bring the realignment comma to the FPGA RX. SHOW_REALIGN_COMMA = TRUE should not be used when ALIGN_COMMA_DOUBLE = TRUE or when manual alignment is used.
RXSLIDE_MODE	String	Defines the RXSLIDE mode. OFF: Default setting. The RXSLIDE feature is not used. PCS: PCS is used to perform the bit-slipping function. RXSLIDE is driven High for one RXUSRCLK2 cycle to shift the parallel data (RXDATA) to the left by one bit within the comma alignment boundary determined by the ALIGN_COMMA_WORD, RX_DATA_WIDTH, and RX_INT_DATAWIDTH settings. In this mode, even if RXOUTCLK is sourcing from the RX PMA, the clock phase remains the same. This option requires SHOW_REALIGN_COMMA to be FALSE. PMA: PMA is used to perform the bit-slipping function. RXSLIDE is driven High for one RXUSRCLK2 cycle to shift the parallel data (RXDATA) to the right by one bit. If RXOUTCLK is sourcing from the RX PMA, its phase might be changed. This mode provides minimum variation of latency compared to PCS mode. This option requires SHOW_REALIGN_COMMA to be FALSE. AUTO: This is an automated PMA mode without using the FPGA logic to monitor the RXDATA and issue RXSLIDE pulses. In this mode, RXSLIDE is ignored. In PCIe® applications, this setting is used for FTS lane deskew. This option requires SHOW_ALIGN_COMMA to be FALSE. When RX buffer bypass is used, RXSLIDE_MODE cannot be set to AUTO or PMA.

Table 4-32: **RX Byte and Word Alignment Attributes *(Cont'd)***

Attribute	Type	Description
RXSLIDE_AUTO_WAIT	Integer	Defines how long the PCS (in terms of RXUSRCLK clock cycle) waits for the PMA to auto slide before checking the alignment again. Valid settings are from 0 to 15. The default value is 7. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
RX_SIG_VALID_DLY	Integer	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
COMMA_ALIGN_LATENCY	7-bit Binary	Current alignment that is used by the byte align block to align the incoming data based on the comma location locked. This register is only accessible via the DRP. GTX transceiver: Bits[6:0] of DRP address `0x14E`. GTH transceiver: Bits[6:0] of DRP address `0x150`.

RX 8B/10B Decoder

Functional Description

If RX received data is 8B/10B encoded, it must be decoded. The GTX/GTH transceiver has a built-in 8B/10B encoder in the GTX/GTH transceiver TX and an 8B/10B decoder in the GTX/GTH transceiver RX, which includes four one-byte 8B/10B decoder modules on the datapath to decode data without consuming FPGA resources. The RX 8B/10B decoder has these features:

1. Supports 2-byte, 4-byte, and 8-byte datapath operation
2. Provides daisy-chained hookup of running disparity for proper disparity
3. Generates K characters and status outputs
4. Can be bypassed if incoming data is not 8B/10B encoded
5. Pipes out 10-bit literal encoded values when encountering a not-in-table error

8B/10B Bit and Byte Ordering

The order of the bits into the 8B/10B decoder is the opposite of the order shown in Appendix C, 8B/10B Valid Characters. 8B/10B decoding requires bit a0 to be received first, but the GTX/GTH transceiver always receives the right-most bit first. Consequently, the 8B/10B decoder automatically reverses the bit order of received data before decoding it. Decoded data is available on RXDATA ports. Figure 4-37 shows data received by the GTX/GTH transceiver RX when RX_DATA_WIDTH = 20, 40, or 80. Data is reconstructed into bytes and sent to the RXDATA interface after the 8B/10B decoder. The number of bits used by RXDATA and corresponding byte orders are determined by RX_DATA_WIDTH.

- Only use RXDATA[15:0] if RX_DATA_WIDTH = 20
- Only use RXDATA[31:0] if RX_DATA_WIDTH = 40

- Use full RXDATA[63:0] if RX_DATA_WIDTH = 80

When the 8B/10B decoder is bypassed but RX_DATA_WIDTH is set to multiple of 10, 10-bit characters are passed to the RX data interface with this format:

- The corresponding RXDISPERR represents the 9th bit
- The corresponding RXCHARISK represents the 8th bit
- The corresponding RXDATA byte represents the [7:0] bits

Figure 4-37: **8B/10B Decoder Bit and Byte Order**

RX Running Disparity

Disparity check is performed and the decoder drives the corresponding RXDISPERR High when the data byte on RXDATA arrives with the wrong disparity. In addition to disparity errors, the 8B/10B decoder detects 20-bit out-of-table error codes. The decoder drives the RXNOTINTABLE port High when decoder is enabled but a received 10-bit character cannot be mapped into a valid 8B/10B character listed in Appendix C, 8B/10B Valid Characters. The non-decoded 10-bit character is piped out of the decoder through the RX data interface with this format:

- The corresponding RXDISPERR represents the 9th bit
- The corresponding RXCHARISK represents the 8th bit
- The corresponding RXDATA byte represents the [7:0] bits

Figure 4-38 shows a waveform at the RX data interface when the decoder receives good data (A), data with disparity error (B), an out-of-table character (C), and an out-of-table character with disparity error (D).

Figure 4-38: **RX Data with 8B/10B Errors**

Special Characters

8B/10B decoding includes special characters (K characters) that are often used for control functions. When RXDATA is a K character, the decoder drives RXCHARISK High.

If DEC_PCOMMA_DETECT is set to TRUE, the decoder drives the corresponding RXCHARISCOMMA High whenever RXDATA is a positive 8B/10B comma. If DEC_MCOMMA_DETECT is TRUE, the decoder drives the corresponding RXCHARISCOMMA bit High whenever RXDATA is a negative 8B/10B comma.

Ports and Attributes

Table 4-33 defines the ports required by RX 8B/10B decoder.

Table 4-33: **RX 8B/10B Decoder Ports**

Port	Dir	Clock Domain	Description
RX8B10BEN	In	RXUSRCLK2	RX8B10BEN selects the use of the 8B/10B decoder in the RX datapath, just after the comma detection/realignment block. If this input is Low, the literal 10-bit data comes out as {RXDISPERR, RXCHARISK, RXDATA<8 bits>}. 1: 8B/10B decoder enabled 0: 8B/10B decoder bypassed (reduces latency)
RXCHARISCOMMA[7:0]	Out	RXUSRCLK2	Active High indicates the corresponding byte shown on RXDATA is a comma character. RXCHARISCOMMA[7] corresponds to RXDATA[63:56] RXCHARISCOMMA[6] corresponds to RXDATA[55:48] RXCHARISCOMMA[5] corresponds to RXDATA[47:40] RXCHARISCOMMA[4] corresponds to RXDATA[39:32] RXCHARISCOMMA[3] corresponds to RXDATA[31:24] RXCHARISCOMMA[2] corresponds to RXDATA[23:16] RXCHARISCOMMA[1] corresponds to RXDATA[15:8] RXCHARISCOMMA[0] corresponds to RXDATA[7:0]
RXCHARISK[7:0]	In	RXUSRCLK2	Active High indicates the corresponding byte shown on RXDATA is a K character when 8B/10B decoding is enabled. RXCHARISK[7] corresponds to RXDATA[63:56] RXCHARISK[6] corresponds to RXDATA[55:48] RXCHARISK[5] corresponds to RXDATA[47:40] RXCHARISK[4] corresponds to RXDATA[39:32] RXCHARISK[3] corresponds to RXDATA[31:24] RXCHARISK[2] corresponds to RXDATA[23:16] RXCHARISK[1] corresponds to RXDATA[15:8] RXCHARISK[0] corresponds to RXDATA[7:0] This is bit 8 of non-decoded data if the 8B/10B decoder is bypassed or the corresponding bit of RXNOTINTABLE is High. Refer to FPGA RX Interface, page 287.

Table 4-33: **RX 8B/10B Decoder Ports *(Cont'd)***

Port	Dir	Clock Domain	Description
RXDISPERR[7:0]	Out	RXUSRCLK2	Active High indicates the corresponding byte shown on RXDATA has a disparity error. RXDISPERR[7] corresponds to RXDATA[63:56] RXDISPERR[6] corresponds to RXDATA[55:48] RXDISPERR[5] corresponds to RXDATA[47:40] RXDISPERR[4] corresponds to RXDATA[39:32] RXDISPERR[3] corresponds to RXDATA[31:24] RXDISPERR[2] corresponds to RXDATA[23:16] RXDISPERR[1] corresponds to RXDATA[15:8] RXDISPERR[0] corresponds to RXDATA[7:0] This is bit 9 of non-decoded data if the 8B/10B decoder is bypassed or the corresponding bit of RXNOTINTABLE is High. Refer to FPGA RX Interface, page 287.
RXNOTINTABLE[7:0]	Out	RXUSRCLK2	Active High indicates the corresponding byte shown on RXDATA was not a valid character in the 8B/10B table. RXNOTINTABLE[7] corresponds to RXDATA[63:56] RXNOTINTABLE[6] corresponds to RXDATA[55:48] RXNOTINTABLE[5] corresponds to RXDATA[47:40] RXNOTINTABLE[4] corresponds to RXDATA[39:32] RXNOTINTABLE[3] corresponds to RXDATA[31:24] RXNOTINTABLE[2] corresponds to RXDATA[23:16] RXNOTINTABLE[1] corresponds to RXDATA[15:8] RXNOTINTABLE[0] corresponds to RXDATA[7:0]
SETERRSTATUS	In	Async	Reserved.

Table 4-34: **RX 8B/10B Decoder Attributes**

Attribute	Type	Description
RX_DISPERR_SEQ_MATCH	Boolean	Specifies whether the disparity error status of a decoded byte must match the indicator in the channel bonding and clock correction sequence. When TRUE, indicates the disparity error status must be matched. When FALSE, ignores the disparity error status.
DEC_MCOMMA_DETECT	Boolean	When set to TRUE, drives the per byte flag RXCHARISCOMMA High when an MCOMMA is detected. When set to FALSE, RXCHARISCOMMA is Low when a negative comma detected.

Table 4-34: **RX 8B/10B Decoder Attributes *(Cont'd)***

Attribute	Type	Description
DEC_PCOMMA_DETECT	Boolean	When set to TRUE, drives the per byte flag RXCHARISCOMMA High when a PCOMMA is detected. When set to FALSE, RXCHARISCOMMA is Low when a positive comma detected.
DEC_VALID_COMMA_ONLY	Boolean	When set to TRUE, drives the per byte flag RXCHARISCOMMA High when only IEEE 802.3 valid commas K28.1, K28.5, and K28.7 are detected. When set to FALSE, RXCHARISCOMMA is for positive or negative 8B/10B commas, depending how the user sets DEC_PCOMMA_DETECT and DEC_MCOMMA_DETECT.
RX_DATA_WIDTH	3-bit Binary	The PCS data width is set at the Fabric user interface with values of 16, 32, or 64 (if 8B/10B decoding is not used) or 20, 40, 80 (if 8B/10B decoding is used).
UCODEER_CLR	1-bit Binary	Reserved.

Enabling and Disabling 8B/10B Decoding

To enable the 8B/10B decoder, RX8B10BEN must be driven High. RX_DATA_WIDTH must be set to a multiple of 8 (8, 16, 32, 64) when the 8B/10B decoder enabled.

To disable the 8B/10B decoder on the GTX/GTH receiver path, RX8B10BEN must be driven Low. When the encoder is disabled, RX_DATA_WIDTH can be set to a multiple of 10 (10, 20, 40, 80). The operation of the RXDATA port with 8B/10B decoding bypassed is described in FPGA RX Interface, page 287.

RX Buffer Bypass

Functional Description

Bypassing the RX elastic buffer is an advanced feature of the 7 series GTX/GTH transceiver. The RX phase alignment circuit is used to adjust the phase difference between the PMA parallel clock domain (XCLK) and the RXUSRCLK domain when the RX elastic buffer is bypassed. It also performs the RX delay alignment by adjusting the RXUSRCLK to compensate for the temperature and voltage variations. The combined RX phase and delay alignments can be automatically performed by the GTX/GTH transceiver or manually controlled by the user. Figure 4-48 shows the XCLK and RXUSRCLK domains, and Table 4-38 shows trade-offs between buffering and phase alignment.

The RX elastic buffer can be bypassed to reduce latency when the RX recovered clock is used to source RXUSRCLK and RXUSRCLK2. When the RX elastic buffer is bypassed, latency through the RX datapath is low and deterministic, but clock correction and channel bonding are not available.

Figure 4-39 shows how RX phase alignment allows the RX elastic buffer to be bypassed. Before RX phase alignment, there is no guaranteed phase relationship between the PMA parallel clock domain (XCLK) and the RXUSRCLK domain. RX phase alignment selects a phase shifted version of the RX recovered clock from the CDR (XCLK) so that there is no significant phase difference between XCLK and RXUSRCLK.

When RX buffer bypass is used, RXSLIDE_MODE cannot be set to AUTO or PMA.

Figure 4-39: **Using RX Phase Alignment**

Ports and Attributes

Table 4-35 defines the RX buffer bypass ports.

Table 4-35: **RX Buffer Bypass Ports**

Port	Dir	Clock Domain	Description
RXPHDLYRESET	In	Async	RX phase alignment hard reset to force RXUSRCLK to the center of the delay alignment tap. The delay alignment tap has a full range of ±4 ns and a half range of ±2 ns. This hard reset can be used to initiate the GTX/GTH transceiver to perform the RX phase and delay alignment automatically when all other RX buffer bypass input ports are set Low. It is recommended to use RXDLYSRESET only for phase and delay alignment.
RXPHALIGN	In	Async	Sets the RX phase alignment. Tied Low when using the auto alignment mode.

Table 4-35: **RX Buffer Bypass Ports *(Cont'd)***

Port	Dir	Clock Domain	Description
RXPHALIGNEN	In	Async	RX phase alignment enable. Tied Low when using the auto alignment mode.
RXPHDLYPD	In	Async	RX phase and delay alignment circuit power down. Tied High when a) RX buffer bypass is not in use; b) RXPD is asserted or c) RXOUTCLKSEL is set to `3'b010` but the recovered clock is not available. Tied Low during RX buffer bypass mode normal operation. 0: Power-up the RX phase and delay alignment circuit. 1: Power-down the RX phase and delay alignment circuit.
RXPHOVRDEN	In	Async	RX phase alignment counter override enable. Tied Low when not in use. 0: Normal operation. 1: Enables the RX phase alignment counter override with the RXPH_CFG[10:6] value.
RXDLYSRESET	In	Async	RX delay alignment soft reset to gradually shift RXUSRCLK to the center of the delay alignment tap. The delay alignment tap has a full range of ±4 ns and a half range of ±2 ns. This soft reset can be used to initiate the GTX/GTH transceiver to perform the RX phase and delay alignment automatically when all other RX bypass buffer input ports are Low.
RXDLYBYPASS	In	Async	RX delay alignment bypass. 0: Uses the RX delay alignment circuit. 1: Bypasses the RX delay alignment circuit.
RXDLYEN	In	Async	RX delay alignment enable. Tied Low when not in use.
RXDLYOVRDEN	In	Async	RX delay alignment counter override enable. Tied Low when not in use. 0: Normal operation. 1: Enables the RX delay alignment counter override with the RXDLY_CFG[14:6] value.

Table 4-35: **RX Buffer Bypass Ports *(Cont'd)***

Port	Dir	Clock Domain	Description
RXDDIEN	In	Async	RX data delay insertion enable in the deserializer. Set High in RX buffer bypass mode.
RXPHALIGNDONE	Out	Async	RX phase alignment done. When the auto RX phase and delay alignment are used, the second rising edge of RXPHALIGNDONE detected after RXDLYSRESETDONE assertion indicates RX phase and delay alignment are done. The alignment of data in RXDATA can change after the second rising edge of RXPHALIGNDONE.
RXPHMONITOR	Out	Async	RX phase alignment monitor.
RXPHSLIPMONITOR	Out	Async	RX phase alignment slip monitor.
RXDLYSRESETDONE	Out	Async	RX delay alignment soft reset done.
RXSYNCMODE	In	Async	GTH transceiver: 0: RX Buffer Bypass Slave lane 1: RX Buffer Bypass Master lane This input is not used in multi-lane manual mode.
RXSYNCALLIN	In	Async	GTH transceiver: Single-lane auto mode: Connect this input to its own RXPHALIGNDONE. Multi-lane auto mode: Connect this input to the ANDed signal of RXPHALIGNDONE of the master and all slave lanes. Multi-lane manual mode: This input is not used in multi-lane manual mode.
RXSYNCIN	In	Async	GTH transceiver: Only valid in multi-lane auto mode applications. Connect this input to RXSYNCOUT from RX buffer bypass master lane.

Table 4-35: **RX Buffer Bypass Ports *(Cont'd)***

Port	Dir	Clock Domain	Description
RXSYNCOUT	Out	Async	GTH transceiver: Only valid for RX buffer bypass master lane in multi-lane auto mode applications. Connect this signal to the RXSYNCIN of each lane within the multi-lane application.
RXSYNCDONE	Out	Async	GTH transceiver: Indicates RX Buffer Bypass alignment procedure completion. Only valid for RX buffer bypass master lane in auto mode operation.

Table 4-36 defines the RX buffer attributes.

Table 4-36: **RX Buffer Bypass Attributes**

Attribute	Type	Description
RXBUF_EN	Boolean	Use or bypass the RX elastic buffer. TRUE: Uses the RX elastic buffer (default). FALSE: Bypasses the RX elastic buffer (advanced feature).
RX_XCLK_SEL	String	Selects the clock source used to drive the RX parallel clock domain (XCLK). RXREC: Selects the RX recovered clock as source of XCLK. Used when using the RX elastic buffer. RXUSR: Selects RXUSRCLK as the source of XCLK. Used when bypassing the RX elastic buffer.
RXPH_CFG	24-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
RXPH_MONITOR_SEL	5-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
RXPHDLY_CFG	24-bit Binary	RX phase and delay alignment configuration. RXPHDLY_CFG[19] = 1 is used to set the RX delay alignment tap to the full range of ±4 ns. RXPHDLY_CFG[19] = 0 is used to set the RX delay alignment tap to the half range of ±2 ns. Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
RXDLY_CFG	16-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.

Table 4-36: **RX Buffer Bypass Attributes *(Cont'd)***

Attribute	Type	Description
RXDLY_LCFG	9-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
RXDLY_TAP_CFG	16-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
RX_DDI_SEL	6-bit Binary	RX data delay insertion select. Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
RXSYNC_MULTILANE	1-bit Binary	GTH transceiver: Indicates whether the lane is used as part of a multi-lane interface. Only valid on RX buffer bypass master lane in auto mode. 0: This lane is used in single-lane mode. 1: This lane is used in multi-lane mode.
RXSYNC_SKIP_DA	1-bit Binary	GTH transceiver: Control to skip delay alignment procedure. Only valid on RX buffer bypass master lane in auto mode. 0: RX delay alignment procedure occurs. 1: RX delay alignment procedure is skipped.
RXSYNC_OVRD	1-bit Binary	GTH transceiver: Manual mode override. 0: RX Buffer bypass auto mode is enabled. 1: RX Buffer bypass manual mode is used. RX Buffer bypass control is implemented in fabric logic.
TST_RSV[0]	1-bit Binary	0: Normal. 1: Override data delay insertion (DDI) delay setting with RX_DDI_SEL attribute.

RX Buffer Bypass Use Modes

RX phase alignment can be performed on one channel (single lane) or a group of channels sharing a single RXOUTCLK (multi-lane). For GTX transceivers, RX buffer bypass supports single-lane auto mode and multi-lane manual mode. For GTH transceivers, RX buffer bypass supports single-lane auto mode, and multi-lane applications in manual and auto mode (Table 4-37).

Table 4-37: **RX Buffer Bypass Use Modes**

Rx Buffer Bypass	GTX Transceiver	GTH Transceiver
Single-Lane	Auto	Auto[(1)]
Multi-Lane	Manual[(2)]	Manual or Auto[(2)]

Notes:

1. Single-lane auto mode in GTX transceivers is not compatible with single-lane mode in GTH transceivers.
2. Multi-lane RX buffer bypass support crossing the SLR boundary in SSI-based devices is an advanced feature and is not recommended for normal operation. This feature can be guaranteed only under certain system-level conditions and data rates.

Using RX Buffer Bypass in Single-Lane Auto Mode (GTX Transceiver Only)

These GTX transceiver settings should be used to bypass the RX elastic buffer:

- RXBUF_EN = FALSE
- RX_XCLK_SEL = RXUSR
- RXOUTCLKSEL = `010b` to select the RX recovered clock as the source of RXOUTCLK
- RXDDIEN = 1

With the RX recovered clock selected, RXOUTCLK is to be used as the source of RXUSRCLK. The user must ensure that RXOUTCLK and the selected RX recovered clock are running and operating at the desired frequency. When the RX elastic buffer is bypassed, the RX phase alignment procedure must be performed after these conditions:

- Resetting or powering up the GTX receiver
- Resetting or powering up the CPLL and/or QPLL
- Changing the RX recovered clock source or frequency
- Changing the GTX transceiver RX line rate

Figure 4-40 shows the required steps to perform the auto RX phase alignment and use the RX delay alignment to adjust RXUSRCLK to compensate for temperature and voltage variations.

Figure 4-40: **RX Buffer Bypass—Single-Lane Auto mode (GTX Transceiver Only)**

Notes relevant to Figure 4-40:

1. The sequence of events in Figure 4-40 is not drawn to scale.
2. After conditions such as a GTX receiver reset or RX rate change, RX phase alignment must be performed to align XCLK and RXUSRCLK. Wait until exiting RXELECIDLE

and RX CDR is locked before asserting RXDLYSRESET to start the RX phase and delay alignments. The assertion of RXDLYSRESET should be less than 50 ns.

3. Wait until RXDLYSRESETDONE = 1. RXDLYSRESETDONE will stay asserted for a minimum of 100 ns.
4. RX phase alignment is done when the second rising edge of RXPHALIGNDONE is detected. The first assertion of RXPHALIGNDONE will have a minimum pulse width of 100 ns. Upon the second rising edge of RXPHALIGNDONE, this signal should remain asserted until another alignment procedure is initiated.
5. RX delay alignment continues to adjust RXUSRCLK to compensate for temperature and voltage variations.

It is necessary to start the RX phase alignment after RXELECIDLE is deasserted and RX CDR is lock to ensure that the RX recovered clock and RXUSRCLK are stable and ready to be used for alignment. When the RX elastic buffer is bypassed, data received from the PMA can be distorted due to phase differences after conditions such as a GTX transceiver reset or rate change. If the received data evaluated at the fabric interface is invalid, the RX phase alignment needs to be repeated while the RX CDR is locked.

Using RX Buffer Bypass in Single-Lane Auto Mode (GTH Transceiver Only)

These GTH transceiver settings should be used to bypass the RX buffer:

- RXBUF_EN = FALSE.
- RX_XCLK_SEL = RXUSR.
- RXOUTCLKSEL = `010b` to select the RX recovered clock as the source of RXOUTCLK.
- RXDDIEN = 1.

With the RX recovered clock selected, RXOUTCLK is to be used as the source of RXUSRCLK. The user must ensure that RXOUTCLK and the selected RX recovered clock are running and operating at the desired frequency. When the RX elastic buffer is bypassed, the RX phase alignment procedure must be performed after these conditions:

- Resetting or powering up the GTH receiver.
- Resetting or powering up the CPLL and/or QPLL.
- Changing the RX recovered clock source or frequency.
- Changing the GTH RX line rate.

To set up RX buffer bypass in single-lane auto mode, these attributes should be set:

- RXSYNC_MULTILANE = 0
- RXSYNC_OVRD = 0

Set the ports as per Figure 4-41.

Figure 4-41: **RX Buffer Bypass—Single-Lane, Auto Mode Port Connection (GTH Transceiver Only)**

Figure 4-42 shows the required steps to perform the auto RX phase alignment and use the RX delay alignment to adjust RXUSRCLK to compensate for temperature and voltage variations.

Figure 4-42: **RX Buffer Bypass Example—Single-Lane Auto Mode (GTH Transceiver Only)**

Notes relevant to Figure 4-42:

1. The sequence of events in Figure 4-42 is not drawn to scale.
2. After conditions such as a GTH receiver reset or RX rate change, RX phase alignment must be performed to align XCLK and RXUSRCLK. Wait until exiting RXELECIDLE and RX CDR is locked before asserting RXDLYSRESET to start the RX phase and delay alignments. The assertion of RXDLYSRESET should be less than 50 ns.
3. Wait until RXDLYSRESETDONE is High. RXDLYSRESETDONE will stay asserted for a minimum of 100 ns.
4. When RXSYNCDONE is asserted, the alignment procedure is completed. This signal will remain asserted until the alignment procedure is re-initiated.
5. Upon the assertion of RXSYNCDONE, RXPHALIGNDONE indicates whether alignment is achieved and maintained.
6. RX delay alignment continues to adjust RXUSRCLK to compensate for temperature and voltage variations.

It is necessary to start the RX phase alignment after RX CDR is locked to ensure that the RX recovered clock and RXUSRCLK are stable and ready to be used for alignment. When the RX elastic buffer is bypassed, data received from the PMA can be distorted due to phase differences after conditions such as a GTH transceiver reset or rate change. If the received

data evaluated at the fabric interface is invalid, the RX phase alignment needs to be repeated while the RX CDR is locked.

Using RX Buffer Bypass in Multi-Lane Manual Mode (GTX and GTH Transceivers)

Multi-lane RX buffer bypass support crossing the SLR boundary in SSI-based devices is an advanced feature and is not recommended for normal operation. This feature can be guaranteed only under certain system-level conditions and data rates.

For GTX transceivers, when a multi-lane application requires RX buffer bypass, phase alignment should be performed manually. For GTH transceivers, phase alignment can be performed manually or automatically.

This section describes the steps required to perform the multi-lane RX buffer bypass alignment procedure manually:

- Master: In a multi-lane application, the buffer bypass master is the lane that is the source of RXOUTCLK.
- Slave: All the lanes that share the same RXUSRCLK/RXUSRCLK2, which is generated from the RXOUTCLK of the buffer bypass master.

Figure 4-43 shows an example of buffer bypass master versus slave lanes.

Figure 4-43: **Example of RX Buffer Bypass Master versus Slave Lanes**

These GTX/GTH transceiver settings should be used to bypass the RX elastic buffer:

- RXBUF_EN = FALSE
- RX_XCLK_SEL = RXUSR

- RXOUTCLKSEL = 010 to select the RX recovered clock as the source of RXOUTCLK
- RXDDIEN = 1

With the RX recovered clock selected, RXOUTCLK is to be used as the source of RXUSRCLK. The user must ensure that RXOUTCLK and the selected RX recovered clock are operating at the desired frequency. When the RX elastic buffer is bypassed, the RX phase alignment procedure must be performed after these conditions:

- Resetting or powering up the GTX/GTH receiver
- Resetting or powering up the CPLL and/or QPLL
- Changing the RX recovered clock source or frequency
- Changing the GTX/GTH transceiver RX line rate

Figure 4-44 shows the required steps to perform manual RX phase and delay alignment.

Figure 4-44: **RX Phase and Delay Alignment in Manual Mode**

Notes relevant to Figure 4-44:

1. The sequence of events shown in Figure 4-44 is not drawn to scale.
2. M_* denotes ports related to the master lane.
3. S_* denotes ports related to the slave lane(s).
4. GTX transceiver: Set the PCS_RSVD_ATTR[2] attribute to `1'b1`.
   GTH transceiver: Set the RXSYNC_OVRD attribute to `1'b1`.
5. Set RXPHDLYRESET and RXDLYBYPASS to Low for all lanes.
6. Set RXPHALIGNEN and RXDDIEN to High for all lanes.
7. Assert RXDLYSRESET for all lanes. Hold this signal High until RXDLYSRESETDONE of the respective lane is asserted.

8. Deassert RXDLYSRESET for the lane in which the RXDLYSRESETDONE is asserted.
9. When RXDLYSRESET of all lanes are deasserted, assert RXPHALIGN for the master lane. Hold this signal High until the rising edge of RXPHALIGNDONE of the master lane is observed.
10. Deassert RXPHALIGN for the master lane.
11. Assert RXDLYEN for the master lane. This causes RXPHALIGNDONE to be deasserted.
12. Hold RXDLYEN for the master lane High until the rising edge of RXPHALIGNDONE of the master lane is observed.
13. Deassert RXDLYEN for the master lane.
14. Assert RXPHALIGN for all slave lane(s). Hold this signal High until the rising edge of RXPHALIGNDONE of the respective slave lane is observed.
15. Deassert RXPHALIGN for the slave lane in which the RXPHALIGNDONE is asserted.
16. When RXPHALIGN for all slave lane(s) are deasserted, assert RXDLYEN for the master lane. This causes RXPHALIGNDONE of the master lane to be deasserted.
17. Wait until RXPHALIGNDONE of the master lane reasserts. Phase and delay alignment for the multi-lane interface is complete. Continue to hold RXDLYEN for the master lane High to adjust RXUSRCLK to compensate for temperature and voltage variations.

In a multi-lane application, it is necessary to start the RX alignment procedure on the interface after RXELECIDLE is deasserted on any lane. The RX CDR of all lanes should be locked before starting the RX alignment procedure. This requirement is to ensure that the RX recovered clocks and RXUSRCLK are stable and ready before alignment.

When the RX elastic buffer is bypassed, data received from the PMA might be distorted due to phase differences after conditions such as a GTX/GTH transceiver reset or rate change. If the received data evaluated at the fabric interface is invalid on any lane, the RX alignment procedure should be repeated for the interface after the RX CDR is locked on all lanes.

Using RX Buffer Bypass in Multi-Lane Auto Mode (GTH Transceiver Only)

Multi-lane TX buffer bypass support crossing the SLR boundary in SSI-based devices is an advanced feature and is not recommended for normal operation. This feature can be guaranteed only under certain system-level conditions and data rates.

For GTH transceivers, when a multi-lane application requires RX buffer bypass, phase alignment can be performed manually or automatically. This section describes the steps required to perform the multi-lane RX buffer bypass alignment procedure automatically:

- Master: In a multi-lane application, the buffer bypass master is the lane that is the source of RXOUTCLK.
- Slave: These are all the lanes that share the same RXUSRCLK/RXUSRCLK2, which is generated from the RXOUTCLK of the buffer bypass master.

Figure 4-45 shows an example of buffer bypass master versus slave lanes.

Figure 4-45: **Example of Buffer Bypass Master versus Slave Lanes**

These GTH transceiver settings should be used to bypass the RX buffer:

- RXBUF_EN = FALSE.
- RX_XCLK_SEL = RXUSR.
- RXOUTCLKSEL = `010` to select the RX recovered clock as the source of RXOUTCLK.
- RXDDIEN = 1.

With the RX recovered clock selected, RXOUTCLK is to be used as the source of RXUSRCLK. The user must ensure that RXOUTCLK and the selected RX recovered clock are running and operating at the desire frequency. When the RX elastic buffer is bypassed, the RX phase alignment procedure must be performed after these conditions:

- Resetting or powering up the GTH receiver.
- Resetting or powering up the CPLL and/or QPLL.
- Changing the RX recovered clock source or frequency.
- Changing the GTH RX line rate.

To set up RX buffer bypass in multi-lane auto mode, the following attributes should be set:

- RXSYNC_MULTILANE = 1
- RXSYNC_OVRD = 0

The ports should be set as shown in Figure 4-46.

Figure 4-46: **RX Buffer Bypass—Multi-Lane Auto Mode Port Connection (GTH Transceiver Only)**

Figure 4-47 shows the required steps to perform auto RX phase and delay alignment.

Figure 4-47: **RX Buffer Bypass Example—Multi-Lane Auto Mode (GTH Transceiver Only)**

Notes relevant to Figure 4-47:

1. The sequence of events shown in Figure 4-47 is not drawn to scale.
2. M_* denotes ports related to the master lane.
3. S_* denotes ports related to the slave lane(s).
4. After conditions such as a GTH receiver reset or RX rate change, RX phase alignment must be performed to align XCLK and RXUSRCLK. Wait until exiting RXELECIDLE and RX CDR is locked before asserting RXDLYSRESET to start the RX phase and delay alignments. The assertion of RXDLYSRESET should be less than 50 ns.
5. Wait until RXDLYSRESETDONE is High. RXDLYSRESETDONE will stay asserted for a minimum of 100 ns.
6. When RXSYNCDONE of the master lane is asserted, the alignment procedure is completed. This signal will remain asserted until alignment procedure is re-initiated.
7. Upon the assertion of RXSYNCDONE of the master lane, RXPHALIGNDONE of the master lane indicates whether alignment is achieved and maintained.
8. RX delay alignment continues to adjust RXUSRCLK to compensate for temperature and voltage variations.

In a multi-lane application, it is necessary to start the RX alignment procedure on the interface after RXELECIDLE is deasserted on any lane. RX CDR of all lanes needs to be locked before starting the RX alignment procedure. This requirement is to make sure the RX recovered clocks and RXUSRCLK are stable and ready before alignment.

When the RX elastic buffer is bypassed, data received from the PMA can be distorted due to phase differences after conditions such as a GTH transceiver reset or rate change. If the received data evaluated at the fabric interface is invalid on any lane, the RX alignment procedure needs to be repeated for the interface after RX CDR is locked on all lanes.

RX Elastic Buffer

Functional Description

The GTX/GTH transceiver RX datapath has two internal parallel clock domains used in the PCS: The PMA parallel clock domain (XCLK) and the RXUSRCLK domain. To receive data, the PMA parallel rate must be sufficiently close to the RXUSRCLK rate, and all phase differences between the two domains must be resolved. Figure 4-48 shows the two parallel clock domains: XCLK and RXUSRCLK.

Figure 4-48: **RX Clock Domains**

The GTX/GTH transceiver includes an RX elastic buffer to resolve differences between the XCLK and RXUSRCLK domains. The phase of the two domains can also be matched by using the RX recovered clock from the transceiver to drive RXUSRCLK and adjusting its phase to match XCLK when the RX buffer is bypassed (see RX Buffer Bypass, page 233). All RX datapaths must use one of these approaches. The costs and benefits of each approach are shown in Table 4-38.

Table 4-38: **RX Buffering versus Phase Alignment**

	RX Elastic Buffer	RX Phase Alignment
Ease of Use	The RX buffer is the recommended default to use when possible. It is robust and easier to operate.	Phase alignment is an advanced feature that requires extra logic and additional constraints on clock sources. RXOUTCLKSEL must select the RX recovered clock as the source of RXOUTCLK to drive RXUSRCLK.
Clocking Options	Can use RX recovered clock or local clock (with clock correction).	Must use the RX recovered clock.

Table 4-38: **RX Buffering versus Phase Alignment *(Cont'd)***

	RX Elastic Buffer	RX Phase Alignment
Initialization	Works immediately.	Must wait for all clocks to stabilize before performing the RX phase and delay alignment procedure.
Latency	Buffer latency depends on features use, such as clock correction and channel bonding.	Lower deterministic latency.
Clock Correction and Channel Bonding	Required for clock correction and channel bonding.	Not performed inside the transceiver. Required to be implemented in user logic.

Ports and Attributes

Table 4-39 defines the RX buffer ports.

Table 4-39: **RX Buffer Ports**

Port	Dir	Clock Domain	Description
RXBUFRESET	In	Async	Resets and reinitializes the RX elastic buffer.
RXBUFSTATUS[2:0]	Out	RXUSRCLK2	RX buffer status. `000b`: Nominal condition. `001b`: Number of bytes in the buffer are less than CLK_COR_MIN_LAT `010b`: Number of bytes in the buffer are greater than CLK_COR_MAX_LAT `101b`: RX elastic buffer underflow `110b`: RX elastic buffer overflow

Table 4-40 defines the RX buffer attributes.

Table 4-40: **RX Buffer Attributes**

Attribute	Type	Description
RXBUF_EN	Boolean	Use or bypass the RX elastic buffer. TRUE: Uses the RX elastic buffer (default). FALSE: Bypasses the RX elastic buffer (advanced feature).
RX_XCLK_SEL	String	Selects the clock source used to drive the RX parallel clock domain (XCLK). RXREC: Selects the RX recovered clock as the source of XCLK. Used when using the RX elastic buffer. RXUSR: Selects RXUSRCLK as the source of XCLK. Used when bypassing the RX elastic buffer.
RX_BUFFER_CFG	6-bit Binary	RX elastic buffer configuration. Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.

Table 4-40: **RX Buffer Attributes *(Cont'd)***

Attribute	Type	Description
RX_DEFER_RESET_BUF_EN	Boolean	Defer RX elastic buffer reset on comma realignment. The time deferred is controlled by RXBUF_EIDLE_HI_CNT. TRUE: Enables deferral of RX elastic buffer reset on comma realignment. FALSE: Disables deferral of RX elastic buffer reset on comma realignment.
RXBUF_ADDR_MODE	String	RX elastic buffer address mode. FULL: Enables the RX elastic buffer for clock correction and channel bonding support. FAST: Enables the RX elastic buffer for phase compensation without clock correction and channel bonding support. This mode is recommended for high line rates.
RXBUF_EIDLE_HI_CNT	4-bit Binary	Controls the timing of asserting the GTX/GTH transceiver internally generated RX elastic buffer reset on electrical idle when valid data is not present on the RXP/RXN serial lines. Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
RXBUF_EIDLE_LO_CNT	4-bit Binary	Controls the timing of deasserting the GTX/GTH transceiver internally generated RX elastic buffer reset on electrical idle when valid data is present on the RXP/RXN serial lines. Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
RXBUF_RESET_ON_CB_CHANGE	Boolean	GTX/GTH transceiver internally generated RX elastic buffer reset on channel bonding change. TRUE: Enables auto RX elastic buffer reset on channel bonding change. FALSE: Disables auto RX elastic buffer reset on channel bonding change.
RXBUF_RESET_ON_COMMAALIGN	Boolean	GTX/GTH transceiver internally generated RX elastic buffer reset on comma realignment. TRUE: Enables auto RX elastic buffer reset on comma alignment. FALSE: Disables auto RX elastic buffer reset on comma alignment.

Table 4-40: **RX Buffer Attributes *(Cont'd)***

Attribute	Type	Description
RX_DEFER_RESET_BUF_EN	Boolean	Defer RX elastic buffer reset on comma realignment. The time deferred is controlled by RXBUF_EIDLE_HI_CNT. TRUE: Enables deferral of RX elastic buffer reset on comma realignment. FALSE: Disables deferral of RX elastic buffer reset on comma realignment.
RXBUF_ADDR_MODE	String	RX elastic buffer address mode. FULL: Enables the RX elastic buffer for clock correction and channel bonding support. FAST: Enables the RX elastic buffer for phase compensation without clock correction and channel bonding support. This mode is recommended for high line rates.
RXBUF_EIDLE_HI_CNT	4-bit Binary	Controls the timing of asserting the GTX/GTH transceiver internally generated RX elastic buffer reset on electrical idle when valid data is not present on the RXP/RXN serial lines. Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
RXBUF_EIDLE_LO_CNT	4-bit Binary	Controls the timing of deasserting the GTX/GTH transceiver internally generated RX elastic buffer reset on electrical idle when valid data is present on the RXP/RXN serial lines. Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
RXBUF_RESET_ON_CB_CHANGE	Boolean	GTX/GTH transceiver internally generated RX elastic buffer reset on channel bonding change. TRUE: Enables auto RX elastic buffer reset on channel bonding change. FALSE: Disables auto RX elastic buffer reset on channel bonding change.
RXBUF_RESET_ON_COMMAALIGN	Boolean	GTX/GTH transceiver internally generated RX elastic buffer reset on comma realignment. TRUE: Enables auto RX elastic buffer reset on comma alignment. FALSE: Disables auto RX elastic buffer reset on comma alignment.

Table 4-40: **RX Buffer Attributes *(Cont'd)***

Attribute	Type	Description
RXBUF_RESET_ON_EIDLE	Boolean	GTX/GTH transceiver internally generated RX elastic buffer reset on electrical idle. TRUE: Enables auto reset of RX elastic buffer during an optional reset sequence of an electrical idle state as used in PCI Express operation. FALSE: Disables auto RX elastic buffer reset on electrical idle. This should be the default setting. **Note:** For channels with large attenuation (lossy channels typically exceeding 15 dB at Nyquist), it is recommended that RXBUF_RESET_ON_EIDLE should be set to FALSE because fast transitioning data patterns like the `101010` sequence in CJPAT/CJTPAT can accidentally trigger an electrical idle.
RXBUF_RESET_ON_RATE_CHANGE	Boolean	GTX/GTH transceiver internally generated RX elastic buffer reset on rate change. TRUE: Enables auto RX elastic buffer reset on rate change. FALSE: Disables auto RX elastic buffer reset on rate change.
RXBUF_THRESH_OVRD	Boolean	RX elastic buffer threshold override. TRUE: Use the RXBUF_THRESH_OVFLW and RXBUF_THRESH_UNDFLW attributes to set the RX elastic buffer overflow and underflow thresholds, respectively. FALSE: Automatically calculates the RX elastic buffer overflow and underflow thresholds. This is the recommended default setting.
RXBUF_THRESH_OVFLW	Integer	RX elastic buffer overflow threshold specified as the number of bytes. If the data latency through the RX elastic buffer is at or above this threshold, the buffer is considered to be in an overflow condition. Used when RXBUF_THRESH_OVRD = TRUE. Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
RXBUF_THRESH_UNDFLW	Integer	RX elastic buffer underflow threshold specified as number of bytes. If the data latency through the RX elastic buffer is at or below this threshold, the buffer is consider to be in underflow condition. Used when RXBUF_THRESH_OVRD = TRUE. Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
RXBUFRESET_TIME	5-bit Binary	RX elastic buffer reset time. Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.

Using the RX Elastic Buffer

These settings are used to enable the RX elastic buffer to resolve phase differences between the XCLK and RXUSRCLK domains:

- RXBUF_EN = TRUE
- RX_XCLK_SEL = RXREC

The content of the RX elastic buffer becomes invalid if an RX elastic buffer overflow or underflow condition occurs. When any of these conditions occur, the RX elastic buffer should be reset and reinitialized by using GTRXRESET, RXPCSRESET, RXBUFRESET, or the GTX/GTH transceiver internally generated RX elastic buffer reset (see RX Initialization and Reset, page 71). The internally generated RX elastic buffer reset can occur on channel bonding change, comma realignment, electrical idle, or rate change conditions.

The RX elastic buffer is also used for clock correction (see RX Clock Correction) and channel bonding (see RX Channel Bonding, page 263). Clock correction is used in cases where XCLK and RXUSRCLK are not frequency matched. Table 4-41 lists common clock configurations and shows whether they require clock correction.

Table 4-41: **Common Clock Configurations**

Types of Clocking	Require Clock Correction?
Synchronous system where both sides uses the reference clock from the same physical oscillator.	No
Asynchronous system when separate reference clocks are used and the GTX/GTH receiver uses an RX recovered clock.	No
Asynchronous system when separate reference clocks are used and the GTX/GTH receiver uses a local clock.	Yes

When the RX elastic buffer is used, the setting of CLK_COR_MIN_LAT affects the latency through the buffer, regardless of whether clock correction is used.

RX Clock Correction

Functional Description

The RX elastic buffer is designed to bridge between two different clock domains, RXUSRCLK and XCLK, which is the recovered clock from CDR. Even if RXUSRCLK and XCLK are running at same clock frequency, there is always a small frequency difference. Because XCLK and RXUSRCLK are not exactly the same, the difference can be accumulated to cause the RX elastic buffer to eventually overflow or underflow unless it is corrected. To allow correction, each GTX/GTH transceiver TX periodically transmits one or more special characters that the GTX/GTH transceiver RX is allowed to remove or replicate in the RX elastic buffer as necessary. By removing characters when the RX elastic buffer is too full and replicating characters when the RX elastic buffer is too empty, the receiver can prevent overflow or underflow.

Figure 4-49: **Clock Correction Conceptual View**

Ports and Attributes

Table 4-42 defines the ports required by RX clock correction functions.

Table 4-42: **RX Clock Correction Ports**

Port	Dir	Clock Domain	Description
RXBUFRESET	In	Async	Resets the RX elastic buffer and related logic.
RXBUFSTATUS[2:0]	Out	RXUSRCLK2	Indicates the status of the RX elastic buffer: `000`: In nominal operating range where the buffer occupancy is within the CLK_COR_MIN_LAT and CLK_COR_MAX_LAT range `001`: RX elastic buffer occupancy is less than CLK_COR_MIN_LAT `010`: RX elastic buffer occupancy is greater than CLK_COR_MAX_LAT `101`: RX elastic buffer underflow `110`: RX elastic buffer overflow
RXCLKCORCNT[1:0]	Out	RXUSRCLK2	Reports the clock correction status of the RX elastic buffer when the first byte of a clock correction sequence is shown in RXDATA. `00`: No clock correction `01`: One sequence skipped `10`: Two sequences skipped `11`: One sequence added
RX8B10BEN	In	RXUSRCLK2	Active High to enable the 8B/10B decoder in the GTX/GTH transceiver RX. If 8B/10B decoding is enabled, RX_DATA_WIDTH must be a multiple of 10 (20, 40, 80). If 8B/10B decoding is not enabled, RX_DATA_WIDTH must be a multiple of 8 (16, 32, 64).

Table 4-43 defines the attributes required by RX channel bonding.

Table 4-43: **RX Clock Correction Attributes**

Attribute	Type	Description
CBCC_DATA_SOURCE_SEL	String	This attribute is used together with RX8B10BEN to select the data source for clock correction and channel bonding. When RX8B10BEN is High, CBCC_DATA_SOURCE_SEL = DECODED, the clock correction sequence matches the data decoded after the 8B/10B decoder. CBCC_DATA_SOURCE_SEL = ENCODED, the clock correction sequence matches the raw data from the comma detection and realignment block before the 8B/10B decoder. When RX8B10BEN is Low, CBCC_DATA_SOURCE_SEL = DECODED is not supported. CBCC_DATA_SOURCE_SEL = ENCODED, the clock correction sequence matches the raw data from the comma detection and realignment block.
CLK_CORRECT_USE	Boolean	Set TRUE to enable the clock correction function. Set FALSE to disable the clock correction function. These attributes need to be set while clock correction disabled: CLK_COR_SEQ_1_1 = `10'b0100000000` CLK_COR_SEQ_2_1 = `10'b0100000000` CLK_COR_SEQ_1_ENABLE = `4'b1111` CLK_COR_SEQ_2_ENABLE = `4'b1111`
CLK_COR_KEEP_IDLE	Boolean	Set TRUE to keep at least one clock correction sequence in the data stream for every continuous stream of clock correction sequences received. Set FALSE to remove all clock correction sequences from the byte stream if needed to recenter the RX elastic buffer range.
CLK_COR_MAX_LAT	Integer	Specifies the maximum RX elastic buffer latency. If the RX elastic buffer exceeds CLK_COR_MAX_LAT, the clock correction circuit removes incoming clock correction sequences to prevent overflow. Valid values for this attribute range from 3 to 60. The 7 Series FPGAs Transceivers Wizard chooses an optimal CLK_COR_MAX_LAT value based on application requirements.

Table 4-43: **RX Clock Correction Attributes *(Cont'd)***

Attribute	Type	Description
CLK_COR_MIN_LAT	Integer	Specifies the minimum RX elastic buffer latency. If the RX elastic buffer drops below CLK_COR_MIN_LAT, the clock correction circuit replicates incoming clock correction sequences to prevent underflow. When the RX elastic buffer is reset, its pointers are set so that there are CLK_COR_MIN_LAT unread (and uninitialized) data bytes in the buffer. Valid values for this attribute range from 3 to 60. Refer to Table 4-44 for the restriction. The 7 Series FPGAs Transceivers Wizard chooses a CLK_COR_MIN_LAT value based on application requirements.
CLK_COR_PRECEDENCE	Boolean	Determines whether clock correction or channel bonding takes precedence when both operations are triggered at the same time. TRUE: Clock correction takes precedence over channel bonding if there is opportunity for both FALSE: Channel bonding takes precedence over clock correction if there is opportunity for both
CLK_COR_REPEAT_WAIT	Integer	This attribute specifies the minimum number of RXUSRCLK cycles between two successive clock corrections being placed. If this attribute is 0, no limit is placed on how frequently the clock correction character can be placed. Valid values for this attribute range from 0 to 31.
CLK_COR_SEQ_LEN	Integer	Defines the length of the sequence in bytes that has to match to detect opportunities for clock correction. This attribute also defines the size of the adjustment (number of bytes repeated or skipped) in a clock correction. Valid lengths are 1, 2, and 4 bytes.

Table 4-43: **RX Clock Correction Attributes *(Cont'd)***

Attribute	Type	Description
CLK_COR_SEQ_1_ENABLE	4-bit Binary	Mask enable bit for the first clock correction sequence. CLK_FOR_SEQ_1_ENABLE[0] is the mask bit for CLK_COR_SEQ_1_1. CLK_FOR_SEQ_1_ENABLE[1] is the mask bit for CLK_COR_SEQ_1_2. CLK_FOR_SEQ_1_ENABLE[2] is the mask bit for CLK_COR_SEQ_1_3. CLK_FOR_SEQ_1_ENABLE[3] is the mask bit for CLK_COR_SEQ_1_4. When CLK_FOR_SEQ_1_ENABLE[*] is 0, the corresponding CLK_COR_SEQ_1_* is either considered as a don't care or is matched automatically without a comparison. When CLK_FOR_SEQ_1_ENABLE[*] is 1, the corresponding CLK_COR_SEQ_1_* is compared for a match.
CLK_COR_SEQ_1_1	10-bit Binary	First clock correction sequence 1 to be compared when CLK_FOR_SEQ_1_ENABLE[0] = 1.
CLK_COR_SEQ_1_2	10-bit Binary	First clock correction sequence 2 to be compared when CLK_FOR_SEQ_1_ENABLE[1] = 1.
CLK_COR_SEQ_1_3	10-bit Binary	First clock correction sequence 3 to be compared when CLK_FOR_SEQ_1_ENABLE[2] = 1.
CLK_COR_SEQ_1_4	10-bit Binary	First clock correction sequence 4 to be compared when CLK_FOR_SEQ_1_ENABLE[3] = 1.
CLK_COR_SEQ_2_USE	Boolean	Set to TRUE if the second clock correction sequence (CLK_COR_SEQ_2_*) is used in addition to the CLK_COR_SEQ_1_* that is always used.

Table 4-43: **RX Clock Correction Attributes *(Cont'd)***

Attribute	Type	Description
CLK_COR_SEQ_2_ENABLE	4-bit Binary	Mask enable bit for the second clock correction sequence. CLK_FOR_SEQ_2_ENABLE[0] is the mask bit for CLK_COR_SEQ_2_1. CLK_FOR_SEQ_2_ENABLE[1] is the mask bit for CLK_COR_SEQ_2_2. CLK_FOR_SEQ_2_ENABLE[2] is the mask bit for CLK_COR_SEQ_2_3. CLK_FOR_SEQ_2_ENABLE[3] is the mask bit for CLK_COR_SEQ_2_4. When CLK_FOR_SEQ_2_ENABLE[*] is 0, the corresponding CLK_COR_SEQ_2_* is either considered as a don't care or is matched automatically without a comparison. When CLK_FOR_SEQ_2_ENABLE[*] is 1, the corresponding CLK_COR_SEQ_2_* is compared for a match.
CLK_COR_SEQ_2_1	10-bit Binary	Second clock correction sequence 1 to be compared when CLK_FOR_SEQ_2_ENABLE[0] = 1
CLK_COR_SEQ_2_2	10-bit Binary	Second clock correction sequence 2 to be compared when CLK_FOR_SEQ_2_ENABLE[1] = 1
CLK_COR_SEQ_2_3	10-bit Binary	Second clock correction sequence 3 to be compared when CLK_FOR_SEQ_2_ENABLE[2] = 1
CLK_COR_SEQ_2_4	10-bit Binary	Second clock correction sequence 4 to be compared when CLK_FOR_SEQ_2_ENABLE[3] = 1
RX_DATA_WIDTH	Integer	Sets the bit width of the RXDATA port. When 8B/10B encoding is enabled, RX_DATA_WIDTH must be set to 20, 40, or 80. Valid settings are 16, 20, 32, 40, 64, and 80. See Interface Width Configuration, page 287 for more details.
RX_DISPERR_SEQ_MATCH	Boolean	Specifies whether the disparity error status of a decoded byte must match the indicator in the channel bonding and clock correction sequence. TRUE: The disparity error status must be matched. FALSE: The disparity error status is ignored.

Table 4-43: **RX Clock Correction Attributes *(Cont'd)***

Attribute	Type	Description
RX_INT_DATAWIDTH	Integer	Controls the width of the internal datapath. 0: 2-Byte internal datapath 1: 4-Byte internal datapath
ALIGN_COMMA_WORD	Integer	This attribute controls the alignment of detected commas within a multi-byte datapath. 1: Align the comma to either of the 2 bytes for a 2-byte interface, any of the 4 bytes for a 4-byte interface, and any of the 8 bytes for an 8-byte interface. The comma can be aligned to either the even bytes or the odd bytes of the RXDATA output. 2: Align the comma to the even bytes only. The aligned comma is guaranteed to be aligned to even bytes RXDATA[9:0] for a 2-byte interface, RXDATA[9:0]/RXDATA[29:20] for a 4-byte interface, and RXDATA[9:0]/RXDATA[29:20]/RX[49:40/RX[69:60] for an 8-byte interface 4: Align the comma to a 4-byte boundary. This setting is not allowed for RX_INT_DATAWIDTH = 0. The aligned comma is guaranteed to be aligned to RXDATA[9:0] for a 4-byte interface and RXDATA[9:0]/RXDATA[49:40] for an 8-byte interface. Refer to Figure 4-34 for comma alignment boundaries that are allowed for the different ALIGN_COMMA_WORD, RX_DATA_WIDTH, and RX_INT_DATAWIDTH settings. Protocols that send commas in even and odd positions must set ALIGN_COMMA_WORD to 1.

Using RX Clock Correction

The user must follow the steps described in this section to use the receiver's clock correction feature.

Enabling Clock Correction

Each GTX/GTH transceiver includes a clock correction circuit that performs clock correction by controlling the pointers of the RX elastic buffer. To use clock correction, RXBUF_EN is set to TRUE to turn on the RX elastic buffer, and CLK_CORRECT_USE is set to TRUE to turn on the clock correction circuit.

Clock correction is triggered when the RX elastic buffer latency is too high or too low, and the clock correction circuit detects a match sequence. To use clock correction, the clock correction circuit must be configured to set these items:

- RX elastic buffer limits
- Clock correction sequence

Setting RX Elastic Buffer Limits

The RX elastic buffer limits are set using CLK_COR_MIN_LAT (minimum latency) and CLK_COR_MAX_LAT (maximum latency). When the number of bytes in the RX elastic buffer drops below CLK_COR_MIN_LAT, the clock correction circuit writes an additional CLK_COR_SEQ_LEN byte from the first clock correction sequence it matches to prevent buffer underflow. Similarly, when the number of bytes in the RX elastic buffer exceeds CLK_COR_MAX_LAT, the clock correction circuit deletes CLK_COR_SEQ_LEN bytes from the first clock correction sequence it matches, starting with the first byte of the sequence. The 7 Series FPGAs Transceivers Wizard chooses an optimal setting for CLK_COR_MIN_LAT and CLK_COR_MAX_LAT based on application requirements.

Because CLK_COR_MIN_LAT is used to set the initial RX elastic buffer latency, it must be divisible by the ALIGN_COMMA_WORD setting to preserve the comma alignment through the elastic buffer. The value for CLK_COR_MIN_LAT must comply with RX_INT_DATAWIDTH and ALIGN_COMMA_WORD as shown in Table 4-44.

The CLK_COR_MAX_LAT setting has no impact on the RX elastic buffer latency that is established, so it can be set any value from 3 to 60.

Table 4-44: **CLK_COR_MIN_LAT Setting Restriction**

ALIGN_COMMA_WORD	RX_INT_DATAWIDTH	CLK_COR_MIN_LAT
1	0 (16/20)	No restriction, any value from 3 to 60.
1	1 (32/40)	No restriction, any value from 3 to 60.
2	0 (16/20)	Must be divisible by 2
2	1 (32/40)	Must be divisible by 2
4[(1)]	0 (16/20)	Not supported
4	1 (32/40)	Must be divisible by 4

Notes:

1. ALIGN_COMMA_WORD = 4 and RX_INT_DATAWIDTH = 0 (16/20) is not a valid configuration. Refer to Figure 4-34 for more information.

Setting Clock Correction Sequences

The clock correction sequences are programmed using the CLK_COR_SEQ_1_* attributes and CLK_COR_SEQ_LEN. Each CLK_COR_SEQ_1_* attribute corresponds to one subsequence in clock correction sequence 1. CLK_COR_SEQ_LEN is used to set the number of subsequences to be matched. If the 40-bit or 20-bit internal datapaths are used, the clock correction circuit matches all 10 bits of each subsequence. If the 16-bit or 32-bit internal datapaths are used, only the right-most eight bits of each subsequence are used.

A second, alternate clock correction sequence can be activated by setting CLK_COR_SEQ_2_USE to TRUE. The first and second sequences share length settings, but

use different subsequence values for matching. Set the CLK_COR_SEQ_2_* attributes to define the subsequence values for the second sequence.

When using 8B/10B decoding (RX8B10BEN is High), CBCC_DATA_SOURCE_SEL is set to DECODED to search the output of the 8B/10B decoder for sequence matches instead of non-decoded data. This allows the circuit to look for 8-bit values with either positive or negative disparity, and to distinguish K characters from regular characters (see TX 8B/10B Encoder, page 108 and RX 8B/10B Decoder, page 228 for details). Figure 4-50 shows how to set a clock correction sequence byte when RX8B10BEN is High and CBCC_DATA_SOURCE_SEL is set to DECODED.

When CBCC_DATA_SOURCE_SEL is set to ENCODED, the sequence must exactly match incoming raw data. When RX_DISPERR_SEQ_MATCH is set to FALSE, CLK_COR_SEQ_x_y[9] is not used for matching.

Figure 4-50: **Clock Correction Subsequence Settings with CBCC_DATA_SOURCE_SEL = DECODED**

Some protocols use clock correction sequences with don't care subsequences. The clock correction circuit can be programmed to recognize these sequences using CLK_COR_SEQ_1_ENABLE and CLK_COR_SEQ_2_ENABLE. When the enable bit for a sequence is Low, that byte is considered matched no matter what the value is. Figure 4-51 shows the mapping between the clock correction sequences and the clock correction sequence enable bits.

Figure 4-51: **Clock Correction Sequence Mapping**

To preserve comma alignment through the elastic buffer, CLK_COR_SEQ_LEN and ALIGN_COMMA_WORD must be selected such that they comply with Table 4-45.

Table 4-45: **Valid ALIGN_COMMA_WORD/CLK_COR_SEQ_LEN Combinations**

ALIGN_COMMA_WORD	CLK_COR_SEQ_LEN
1	1, 2, 4
2	2, 4
4	4

Clock Correction Options

CLK_COR_REPEAT_WAIT is used to control the clock correction frequency. This value is set to the minimum number of RXUSRCLK cycles required between clock correction events. This attribute is set to 0 to allow clock correction to at occur any time. Some protocols allow clock correction to occur at any time, but require that if the clock correction circuit removes sequences, at least one sequence stays in the stream. For protocols with this requirement, CLK_COR_KEEP_IDLE is set to TRUE.

Monitoring Clock Correction

The clock correction circuit can be monitored using the RXCLKCORCNT and RXBUFSTATUS ports. The RXCLKCORCNT entry in Table 4-42 shows how to decode the values of RXCLKCORCNT to determine the status of the clock correction circuit. The RXBUFSTATUS entry in Table 4-42 shows how to decode the values of RXBUFSTATUS to determine how full the RX elastic buffer is.

RX Channel Bonding

Functional Description

Protocols such as XAUI and PCI Express combine multiple serial transceiver connections to create a single higher throughput channel. Each serial transceiver connection is called one lane. Unless each of the serial connections is exactly the same length, skew between the lanes can cause data to be transmitted at the same time but arrive at different times. Channel bonding cancels out the skew between GTX/GTH transceiver lanes by using the RX elastic buffer as a variable latency block. Channel bonding is also called channel deskew or lane-to-lane deskew. GTX/GTH transmitters used for a bonded channel all transmit a channel bonding character (or a sequence of characters) simultaneously. When the sequence is received, the GTX/GTH receiver can determine the skew between each lane and adjust the latency of RX elastic buffers, so that data is presented without skew at the RX fabric user interface.

Figure 4-52: **Channel Bonding Conceptual View**

RX channel bonding supports 8B/10B encoded data but does not support these encoded data types:

- 64B/66B
- 64B/67B
- 128B/130B
- Scrambled data

Ports and Attributes

Table 4-46 defines the ports required by RX channel bonding functions.

Table 4-46: **RX Channel Bonding Ports**

Port	Dir	Clock Domain	Description
RXCHANBONDSEQ	Out	RXUSRCLK2	This port goes High when RXDATA contains the start of a channel bonding sequence.
RXCHANISALIGNED	Out	RXUSRCLK2	This signal from the RX elastic buffer goes High to indicate that the channel is properly aligned with the master transceiver according to observed channel bonding sequences in the data stream. This signal goes Low if an unaligned channel bonding sequence is detected, indicating that channel alignment was lost.
RXCHANREALIGN	Out	RXUSRCLK2	This signal from the RX elastic buffer is held High for at least one cycle when the receiver has changed the alignment between this transceiver and the master.
RXCHBONDI[4:0]	In	RXUSRCLK	Channel bonding control ports used by slaves only. These ports are used to receive channel bonding and clock correction control information from master GTX/GTH transceiver RXCHBONDO ports or from daisy-chained slave GTX/GTH transceiver RXCHBONDO ports, which are concatenated from the master GTX/GTH transceiver.
RXCHBONDO[4:0]	Out	RXUSRCLK	Channel bonding control ports used to propagate channel bonding and clock correction information to the slave GTX/GTH transceiver from the master or a daisy-chained slave concatenated from the master. The master RXCHBONDO can be tied to one or multiple slave RXCHBONDI ports. The slave RXCHBONDO should be tied to the next level slave RXCHBONDI to form a daisy chain and pass information from the master to each slave.

Table 4-46: **RX Channel Bonding Ports *(Cont'd)***

Port	Dir	Clock Domain	Description
RXCHBONDLEVEL[2:0]	In	RXUSRCLK2	Indicates the amount of internal pipelining used for the RX elastic buffer control signals. A higher value permits more daisy chaining of RXCHBONDO and RXCHBONDI to ease placement and routing constraints. To minimize required latency through the RX elastic buffer, CHAN_BOND_LEVEL in the master is set to the smallest value possible for the required amount of daisy-chaining. When using a 4-byte internal datapath (RX_INT_DATAWIDTH = 1), the master should not exceed RXCHANBONDLEVEL = 3.
RXCHBONDMASTER	In	RXUSRCLK2	Indicates that the transceiver is the master for channel bonding. Its RXCHBONDO port directly drives the RXCHBONDI ports on one or more slave transceivers. This port cannot be driven High at the same time as RXCHBONDSLAVE.
RXCHBONDSLAVE	In	RXUSRCLK2	Indicates that this transceiver is a slave for channel bonding. Its RXCHBONDI port is directly driven by the RXCHBONDO port of another slave or master transceiver. If its RXCHBONDLEVEL[2:0] setting is greater than 0, its RXCHBONDO port can directly drive the RXCHBONDI ports on one or more other slave transceivers. This port cannot be driven High at the same time as RXCHBONDMASTER.
RXCHBONDEN	In	RXUSRCLK2	This port enables channel bonding (from the FPGA logic to both the master and slaves).

Table 4-47 defines the attributes required by RX channel bonding.

Table 4-47: **RX Channel Bonding Attributes**

Attribute	Type	Description
CHAN_BOND_MAX_SKEW	Integer	This attribute controls the number of USRCLK cycles that the master waits before ordering the slaves to execute channel bonding. This attribute determines the maximum skew that can be handled by channel bonding. It must always be less than one-half the minimum distance (in bytes or 10-bit codes) between channel bonding sequences. Valid values range from 1 to 14.
CHAN_BOND_KEEP_ALIGN	Boolean	Allows preservation of ALIGN characters during channel bonding for PCI Express.
CHAN_BOND_SEQ_1_1 CHAN_BOND_SEQ_1_2 CHAN_BOND_SEQ_1_3 CHAN_BOND_SEQ_1_4	10-bit Binary	The CHAN_BOND_SEQ_1 attributes are used in conjunction with CHAN_BOND_SEQ_1_ENABLE to define channel bonding sequence 1. Each subsequence is 10 bits long. The rules for setting the subsequences depend on RX_DATA_WIDTH and CBCC_DATA_SOURCE_SEL. Not all subsequences need to be used. CHAN_BOND_SEQ_LEN determines how much of the sequence is used for a match. If CHAN_BOND_SEQ_LEN = 1, only CHAN_BOND_SEQ_1_1 is used. CHAN_BOND_SEQ_1_ENABLE can be used to make parts of the sequence don't care. If CHAN_BOND_SEQ_1_ENABLE[k] is 0, CHAN_BOND_SEQ_1_k is a don't-care subsequence and is always considered to be a match.
CHAN_BOND_SEQ_1_ENABLE	4-bit Binary	

Table 4-47: **RX Channel Bonding Attributes *(Cont'd)***

Attribute	Type	Description
CHAN_BOND_SEQ_2_1 CHAN_BOND_SEQ_2_2 CHAN_BOND_SEQ_2_3 CHAN_BOND_SEQ_2_4	10-bit Binary	The CHAN_BOND_SEQ_2 attributes are used in conjunction with CHAN_BOND_SEQ_2_ENABLE to define the second channel bonding sequence. When CHAN_BOND_SEQ_2_USE is TRUE, the second sequence is used as an alternate sequence to trigger channel bonding. Each subsequence is 10 bits long. The rules for setting the subsequence depend on RX_DATA_WIDTH and CBCC_DATA_SOURCE_SEL. Not all subsequences need to be used. CHAN_BOND_SEQ_LEN determines how many of the subsequences are used for a match. If CHAN_BOND_SEQ_LEN = 1, only CHAN_BOND_SEQ_2_1 is used. CHAN_BOND_SEQ_2_ENABLE can be used to make parts of the sequence don't care. If CHAN_BOND_SEQ_2_ENABLE[k] is 0, CHAN_BOND_SEQ_2_k is a don't-care subsequence and is always considered to be a match.
CHAN_BOND_SEQ_2_ENABLE	4-bit Binary	
CHAN_BOND_SEQ_2_USE	Boolean	Determines if the two-channel bonding sequence is to be used. TRUE: Channel bonding can be triggered by channel bonding sequence 1 or 2. FALSE: Channel bonding is only triggered by sequence 1.
CHAN_BOND_SEQ_LEN	Integer	Defines the length in bytes of the channel bonding sequence that the GTX/GTH transceiver has to match to find skew. Valid lengths are 1, 2, and 4 bytes.
CBCC_DATA_SOURCE_SEL	String	This attribute is used to select the data source for clock correction and channel bonding. When set to DECODED, selects data from the 8B/10B decoder when RX8B10BEN is High. When set to ENCODED, selects data from the comma detection and realignment block.

Table 4-47: **RX Channel Bonding Attributes *(Cont'd)***

Attribute	Type	Description
FTS_DESKEW_SEQ_ENABLE	4-bit Binary	Enable mask for FTS_LANE_DESKEW_CFG. FTS_DESKEW_SEQ_ENABLE[0] is for FTS_LANE_DESKEW_CFG[0] FTS_DESKEW_SEQ_ENABLE[1] is for FTS_LANE_DESKEW_CFG[1] FTS_DESKEW_SEQ_ENABLE[2] is for FTS_LANE_DESKEW_CFG[2] FTS_DESKEW_SEQ_ENABLE[3] is for FTS_LANE_DESKEW_CFG[3] The default value is `1111`.
FTS_LANE_DESKEW_CFG	4-bit Binary	Bit 3: This bit is set to `1'b1` on a slave to freeze the alignment to prevent spurious misalignments or modified alignments that can occur following slip-4, snap-4, or clock correction when good channel alignment is still maintained. This bit is set to `1'b0` on a slave to unfreeze the alignment. Bit 2: Specifies whether a "master" channel doing FTS lane deskew that just reached the end of an FTS OS in its lookahead control logic inhibits its own generation of clock correction commands for a brief time. The purpose is to prevent clock correction commands from interfering with operation of the slave's slip-4 and snap-4 logic. The logic guarantees that clock correction can still occur if a full SKP OS is present. Bit 1: Specifies whether a "slave" channel doing FTS lane deskew is permitted (`1'b1`) or inhibited (`1'b0`) from performing an immediate backward alignment adjustment by four bytes (slip-4) if the slave is found to have reached the SKP OS following FTS before the master. Bit 0: Specifies whether a "slave" channel doing FTS lane deskew is permitted (`1'b1`) or inhibited (`1'b0`) from performing an immediate forward alignment adjustment by four bytes (snap-4) if the master is found to have reached the SKP OS following FTS before the slave.

Table 4-47: **RX Channel Bonding Attributes *(Cont'd)***

Attribute	Type	Description
FTS_LANE_DESKEW_EN	Boolean	This attribute is set to TRUE to enable channel bonding logic for FTS lane deskew. FTS lane deskew is separate from the standard algorithm using channel bonding sequences 1 and 2, and it operates in parallel with the standard algorithm. FTS lane deskew operates only in two-byte mode.
PCS_PCIE_EN	Boolean	This attribute is set to TRUE when the GTX/GTH transceiver is used for PCI Express and set to FALSE for all other protocols. The channel bonding function requires this attribute together with TXCHARDISPMODE and TXCHARDISPVAL to support PIPE encode and FTS lane deskew. It also works together with TXELECIDLE to match a shorter sequence from reusing prior channel bonding information after the GTX/GTH transceiver returns from electrical idle. Refer to PCI Express in Chapter 6 for details.
RX_DATA_WIDTH	Integer	Sets the bit width of the RXDATA port. When 8B/10B encoding is enabled, RX_DATA_WIDTH must be set to 20, 40, or 80. Valid settings are 16, 20, 32, 40, 64, and 80. See Interface Width Configuration, page 287 for more details.
RX_DISPERR_SEQ_MATCH	Boolean	Specifies whether the disparity error status of a decoded byte must match the indicator in the channel bonding and clock correction sequence. TRUE: The disparity error must be matched. FALSE: The disparity error status is ignored.

Using RX Channel Bonding

The user must follow the steps described below to use the receiver's channel bonding feature.

Enabling Channel Bonding

Each GTX/GTH transceiver includes a circuit that performs channel bonding by controlling the pointers of the RX elastic buffer. Because channel bonding requires the use of the RX buffer, the RXBUF_EN attribute must be set to TRUE.

Each GTX/GTH transceiver has a channel bonding circuit. Configuring a GTX/GTH transceiver for channel bonding requires these steps:

1. Set the channel bonding mode for each GTX/GTH transceiver.
2. Tie the RXCHBONDMASTER of the master transceiver High.
3. Tie the RXCHBONDSLAVE of the slave transceiver(s) High.
4. Connect the channel bonding port from the master to each slave, either directly or by daisy chaining.
5. Set the channel bonding sequence and detection parameters.

Channel Bonding Mode

The channel bonding mode for each GTX/GTH transceiver determines whether channel bonding is active and whether the GTX/GTH transceiver is the master or a slave. Each set of channel bonded GTX/GTH transceivers must have one master and any number of slaves. To turn on channel bonding for a group of GTX/GTH transceivers, one transceiver is set to master. The remaining GTX/GTH transceivers in the group are set to slaves.

Connecting Channel Bonding Ports

The channel bonding operation requires connecting the master GTX/GTH transceiver RXCHBONDO port to the RXCHBONDI port of all slaves in the group. Only GTX/GTH transceivers belonging to the same column can be channel bonded together. A direct connection is required for adjacent GTX/GTH transceivers. To directly connect a master to a slave:

1. Connect the RXCHBONDO port of the master to the RXCHBONDI port of the slave.
2. Tie the RXCHBONDMASTER of the master transceiver High.
3. Tie the RXCHBONDSLAVE of each slave transceiver High.

When GTX/GTH transceivers are directly connected, meeting the timing constraints becomes difficult as the transceivers get further apart. The solution to this problem is to connect the transceivers in a daisy chain. Daisy chaining is performed using the RXCHBONDLEVEL[2:0] ports to allow additional pipeline stages between the master and the slave. The RXCHBONDO port of each slave is used as a pipeline stage in the RXCHBONDO path from the master. Figure 4-53 and Figure 4-54 show two daisy-chain examples.

Figure 4-53: **Channel Bonding Daisy Chain Example 1**

Figure 4-54: **Channel Bonding Daisy Chain Example 2**

To set up a daisy chain, the GTX/GTH transceivers are first connected using RXCHBONDO and RXCHBONDI to create a path from the RXCHBONDI port of each slave to the RXCHBONDO port of the master. The following steps describe how to set the RXCHANBONDLEVEL for the GTX/GTH transceivers in the chain:

1. Set the RXCHANBONDLEVEL of the master to 7.
2. Set the RXCHANBONDLEVEL of each slave to the RXCHANBONDLEVEL of the GTX/GTH transceiver driving the slave's RXCHBONDI port minus 1.
3. Find the slave with the lowest level. Subtract this level from the RXCHANBONDLEVEL of all GTX/GTH transceivers so that the lowest slave has level 0 and the master has the minimum level required to service all the slaves. When using a 4-byte internal datapath (RX_INT_DATAWIDTH = 1), do not have the master exceed RXCHANBONDLEVEL = 3.

When the connections between channel bonding ports among GTX/GTH transceivers are being decided, the designer must remember that RXCHBONDI and RXCHBONDO belong to the RXUSRCLK clock domain. Meeting the timing constraint of RXUSRCLK becomes increasingly difficult as RXUSRCLK increases in frequency and as directly connected transceivers get further apart. As long as timing constraints are met, channel bonding transceivers together in adjacent SLRs is possible.

Selecting a GTX/GTH transceiver in the middle of the GTX/GTH transceiver column to be the master for channel bonding allows for the most flexibility when connecting channel bonding ports. When the channel bonding master is in the middle of the GTX/GTH transceiver column, connections can be made to GTX/GTH transceivers north and south of the master. Because of the GTX/GTH transceiver dedicated clock routing structure, an additional benefit of having the channel bonding master at the center of the GTX/GTH transceiver column is that up to 12 GTX/GTH transceivers can be channel bonded together using a single clock pin pair.

As long as timing constraints are met, there is no limit to the number of GTX/GTH transceivers that can be on a particular RXCHANBONDLEVEL.

Setting Channel Bonding Sequences

The channel bonding sequence is programmed in the same way as the clock correction sequence. CHAN_BOND_SEQ_LEN sets the length of the sequence, and CHAN_BOND_SEQ_1_* sets the values of the sequence. If CHAN_BOND_SEQ_2_USE is TRUE, CHAN_BOND_SEQ_2_* sets the values for the alternate second sequence. The number of active bits in each subsequence depends on RX_DATA_WIDTH and CBCC_DATA_SOURCE_SEL (see RX Clock Correction, page 253). When RX_DISPERR_SEQ_MATCH is set to FALSE, CHAN_BOND_SEQ_x_y[9] is not used for matching.

Figure 4-55 shows how the subsequence bits are mapped.

Figure 4-55: **Channel Bonding Sequence Settings**

As with clock correction sequences, channel bonding sequences can have don't care subsequences. CHAN_BOND_SEQ_1_ENABLE and CHAN_BOND_SEQ_2_ENABLE set these bytes. Figure 4-56 shows the mapping of the enable attributes for the channel bonding subsequences.

Figure 4-56: **Channel Bonding Sequence Mapping**

Setting the Maximum Skew

When the master receives a channel bonding sequence, it does not trigger channel bonding immediately. Several more bytes must arrive if the slaves have more latency. This wait time effectively becomes the maximum skew that the RX elastic buffer can handle. If the skew is greater than this wait time, the slaves might not receive the sequence by the time the master triggers channel bonding.

Figure 4-57 shows two FIFOs, one for the master and one for the slave. If the slave is behind the master, the master must wait several cycles before triggering channel bonding, otherwise the slow slave does not have the channel bonding sequence in its buffer.

Figure 4-57: **Channel Bonding Example (CHAN_BOND_MAX_SKEW = 2 and Master RXCHANBONDLEVEL[2:0] = 1)**

CHAN_BOND_MAX_SKEW is used to set the maximum skew allowed for channel bonding sequences 1 and 2. The maximum skew range is 1 to 14. This range must always be less than one-half the minimum distance (in bytes or 10-bit codes) between channel bonding sequences. This minimum distance is determined by the protocol being used.

Precedence between Channel Bonding and Clock Correction

The clock correction (see RX Clock Correction, page 253) and channel bonding circuits both perform operations on the pointers of the RX elastic buffer. Normally, the two circuits work together without conflict, except when clock correction events and channel bonding events occur simultaneously. In this case, one of the two circuits must take precedence. To make clock correction a higher priority than channel bonding, CLK_COR_PRECEDENCE must be set to TRUE. To make channel bonding a higher priority, CLK_COR_PRECEDENCE must be set to FALSE.

RX Gearbox

Functional Description

The RX gearbox provides support for 64B/66B and 64B/67B header and payload separation. The gearbox uses output pins RXDATA[63:0] and RXHEADER[2:0] for the payload and header of the received data in normal mode. Similar to TX Gearbox, page 112, the RX gearbox operates with the PMA using a single clock. Because of this, occasionally, the output data is invalid. Output pins RXHEADERVALID and RXDATAVALID determine

if the appropriate header and data are valid. The RX gearbox supports 2-byte, 4-byte, and 8-byte interfaces.

The data out of the RX gearbox is not necessarily aligned. Alignment is done in the FPGA logic. The RXGEARBOXSLIP port can be used to slip the data from the gearbox cycle-by-cycle until correct alignment is reached. It takes a specific number of cycles before the bitslip operation is processed and the output data is stable. Descrambling of the data and block synchronization is done in the FPGA logic. In the GTH transceiver, a CAUI interface mode is also supported besides the normal gearbox mode.

Ports and Attributes

Table 4-48 defines the RX gearbox ports.

Table 4-48: **RX Gearbox Ports**

Port Name	Dir	Clock Domain	Description
RXDATAVALID/ RXDATAVALID[1:0] (GTH transceiver only)	Out	RXUSRCLK2	Status output when Gearbox 64B/66B or 64B/67B is used, which indicates that the data appearing on RXDATA is valid. For example, during 64B/66B encoding, this signal is deasserted every 32 cycles for the 8-byte interface (and 4-byte interface with RX_INT_DATAWIDTH= 0) and every 64 cycles for the 2-byte interface (and 4-byte interface with RX_INT_DATAWIDTH = 1). GTH transceiver: RXDATAVALID[0] indicates that the data appearing on RXDATA is valid in normal mode. The current RXDATA for datastream A is valid in CAUI interface mode. RXDATAVALID[1] indicates that the current RXDATA is valid for datastream B in CAUI interface mode.

Table 4-48: **RX Gearbox Ports *(Cont'd)***

Port Name	Dir	Clock Domain	Description
RXGEARBOXSLIP	In	RXUSRCLK2	When High, this port causes the gearbox contents to slip to the next possible alignment. This port is used to achieve alignment with the FPGA logic. Asserting this port for one RXUSRCLK2 cycle changes the data alignment coming out of the gearbox. RXGEARBOXSLIP must be deasserted for at least one cycle and then reasserted to cause a new realignment of the data. If multiple realignments occur in rapid succession, it is possible to pass the proper alignment point without recognizing the correct alignment point in the FPGA logic. GTH transceiver: RXGEARBOXSLIP for datastream A in CAUI interface mode.
RXHEADER[2:0]/ RXHEADER[5:0] (GTH transceiver only)	Out	RXUSRCLK2	GTX transceiver: Header outputs for 64B/66B (1:0) and 64B/67B (2:0). GTH transceiver: RXHEADER[2:0]: Header output in normal mode and for datastream A in CAUI interface mode. RXHEADER[5:3]: Header output for datastream B in CAUI interface mode.
RXHEADERVALID/ RXHEADERVALID[1:0] (GTH transceiver only)	Out	RXUSRCLK2	Indicates that the RXHEADER is valid when using the gearbox. GTH transceiver: RXHEADERVALID[0]: Indicates that RXHEADER is valid for the current data in normal mode and for datastream A in CAUI interface mode. RXHEADERVALID[1]: Indicates that RXHEADER is valid for datastream B in CAUI interface mode.
RXSLIDE (GTH transceiver only)	In	RXUSRCLK2	Used as RXGEARBOXSLIP for datastream B in CAUI interface mode.

Table 4-48: **RX Gearbox Ports *(Cont'd)***

Port Name	Dir	Clock Domain	Description
RXSTARTOFSEQ/ RXSTARTOFSEQ[1:0] (GTH transceiver only)	Out	RXUSRCLK2	When the gearbox 64B/66B or 64B/67B is enabled, this output indicates when the sequence counter is 0 for the present RXDATA outputs. GTH transceiver: RXSTARTOFSEQ[0]: This output indicates when the sequence counter is 0 for the present RXDATA in normal mode, and for datastream A in CAUI interface mode. RXSTARTOFSEQ[1]: This output indicates when the sequence counter is 0 for datastream B in CAUI interface mode.

Table 4-49 defines the RX gearbox attributes.

Table 4-49: **RX Gearbox Attributes**

Attribute	Type	Description

GEARBOX_MODE	3-bit Binary	This attribute indicates the TX and RX gearbox modes:
		• Bit 2:
		  • GTX transceiver: Unused. Set to 0 in normal operating mode.
		  • GTH transceiver: Set to 1 when the CAUI interface is engaged.
		• Bit 1:
		0: Use the external sequence counter and apply inputs to TXSEQUENCE in the TX gearbox.
		1: Use the internal sequence counter and gate the input header and data with the TXGEARBOXREADY output in the TX gearbox.
		  • GTH transceiver: Unused and set to 1. Internal sequence counter is not supported in the GTH transceiver.
		• Bit 0:
		0: 64B/67B gearbox mode for Interlaken
		1: 64B/66B gearbox

RXGEARBOX_EN	Boolean	When TRUE, this attribute enables the RX gearbox.

Enabling the RX Gearbox

To enable the RX gearbox for the GTX/GTH transceiver, set the attribute RXGEARBOX_EN to TRUE. Bit 2 of the GEARBOX_MODE attribute is unused and must be set to `0` in the GTX transceiver and when in normal operating mode in the GTH transceiver. It is set to `1` to engage the CAUI interface in the GTH transceiver. The GEARBOX_MODE attribute controls the GTX/GTH transceiver's TX and RX gearbox use modes.

RX Gearbox Operating Modes

The RX gearbox operates the same in either external sequence counter mode or internal sequence counter mode. The RX gearbox only supports 2-byte, 4-byte and 8-byte logic interfaces to the FPGA logic.

As shown in Figure 4-58, either mode uses the RXDATA, RXHEADER, RXDATAOUTVALID, and RXHEADEROUTVALID outputs in addition to the RXGEARBOXSLIP input in normal mode (GEARBOX_MODE[2] = `1'b0`).

Figure 4-58: **Gearbox in Either Internal or External Sequence Mode in Normal Mode (GEARBOX_MODE[2] = `1'b0`)**

Figure 4-59 shows an example of four cycles of data entering and exiting the RX gearbox for 64B/66B encoding when using a 4-byte logic interface (RX_DATA_WIDTH = 32 (4-byte), RX_INT_DATAWIDTH = 1 (4-byte)) in normal mode (GEARBOX_MODE[2] = `1'b0`).

Figure 4-59: **RX Gearbox Operation in Normal Mode (GEARBOX_MODE[2] = `1'b0`)**

Note relevant to Figure 4-59:

1. As per IEEE Std 802.3ae-2002 nomenclature, H1 corresponds to RxB<0>, H0 to RxB<1>, etc.

The RX gearbox internally manages all sequencing, which differs from the TX gearbox option of either internal or external sequencing. Depending on whether a 2-byte, 4-byte, or 8-byte interface is used, RXDATAOUTVALID and RXHEADEROUTVALID assert and deassert for different periods of length. The RX gearbox encounters similar data and

header pauses found in the TX gearbox. Figure 4-60 shows such a pause in addition to RXHEADERVALID and RXDATAVALID being deasserted for one cycle. Figure 4-61 shows the operation for 64B/67B encoding when RX_DATA_WIDTH = 16 (2-byte) and RX_INT_DATAWIDTH = 0 (2-byte) in normal mode (GEARBOX_MODE[2] = `1'b0`).

Figure 4-60: **RX Gearbox When Using 64B/66B Encoding and RX_DATA_WIDTH = 64 (8-Byte) and RX_INT_DATAWIDTH= 1 (4-Byte) in Normal Mode (GEARBOX_MODE[2] = `1'b0`)**

Figure 4-61: **RX Gearbox When Using 64B/67B Encoding and RX_DATA_WIDTH = 16 (2-Byte) and RX_INT_DATAWIDTH = 0 (2-Byte)**

RX Gearbox Block Synchronization

The 64B/66B and 64B/67B protocols depend on block synchronization to determine their block boundaries. Block synchronization is required because all incoming data is unaligned before block lock is achieved. The goal is to search for the valid synchronization header by changing the data alignment. The RXGEARBOXSLIP input port is used to change the gearbox data alignment so that all possible alignments can be checked in normal mode (GEARBOX_MODE[2] = 1'b0). (RXSLIDE is used as RXGEARBOXSLIP for

the second datastream in the CAUI interface mode in the GTH transceiver (GEARBOX_MODE[2] = 1'b1).) The RXGEARBOXSLIP signal feeds back from the block synchronization state machine to the RX gearbox and tells it to slip the data alignment. This process of slipping and testing the synchronization header repeats until block lock is achieved. When using the RX gearbox, a block synchronization state machine is required in the FPGA logic. Figure 4-62 shows the operation of a block synchronization state machine. The 7 Series FPGAs Transceivers Wizard has example code for this type of module.

Figure 4-62: **Block Synchronization State Machine**

The state machine works by keeping track of valid and invalid synchronization headers. Upon reset, block lock is deasserted, and the state is LOCK_INIT. The next state is RESET_CNT where all counters are zeroed out. The synchronization header is analyzed in

the TEST_SH state. If the header is valid, sh_cnt is incremented in the VALID_SH state, otherwise sh_count and sh_invalid_count are incremented in the INVALID_SH state.

For the block synchronization state machine shown in Figure 4-62, sh_cnt_max and sh_invalid_cnt_max are both constants that are set to 64 and 16, respectively. From the VALID_SH state, if sh_cnt is less than the value sh_cnt_max and test_sh is High, the next state is TEST_SH. If sh_cnt is equal to sh_cnt_max and sh_invalid_cnt equals 0, the next state is GOOD_64 and from there block_lock is asserted. Then the process repeats again and the counters are cleared to zeros. To achieve block lock, the state machine must receive sh_cnt_max number of valid synchronization headers in a row without getting an invalid synchronization header. However, when block lock is achieved sh_invalid_cnt_max – 1, the number of invalid synchronization headers can be received within sh_cnt_max number of valid synchronization headers. Thus, once locked, it is harder to break lock.

Figure 4-63 shows a waveform of the block synchronization state machine asserting RXGEARBOXSLIP numerous times because of invalid synchronization headers before achieving data alignment. After the RXGEARBOXSLIP is issued, the state machine waits 32 RXUSRCLK2 cycles before checking for valid synchronization headers.

Figure 4-63: **RX Gearbox with Block Synchronization in Normal Mode (GEARBOX_MODE[2] = `1'b0`)**

CAUI Interface (GTH Transceiver)

The CAUI interface requires two data interfaces on the transceiver. This section describes the design of the CAUI interface block on the RX that is implemented in the GTH transceiver. This supports a dual data interface in 64/66 and 64/67 modes (datastream A and datastream B). The CAUI interface mode can be selected by setting the attribute GEARBOX_MODE[2] to `1'b1`. When in CAUI interface mode, the only allowed settings are RX_INT_DATAWIDTH = 1 (4-byte) and RX_DATA_WDTH = 64 (8-byte) or 32 (4-byte).

Use Case

Two PCSLs are expected to interface with PCS across the CAUI interface. Each PCSL performs its own block alignment. Figure 4-64 shows the expected connections between PCSLs and the PCS.

Figure 4-64: **CAUI Interface RX Use Case**

In Figure 4-64, x is the width of PCSL data bus. Allowed values are 16 and 32.

RX Gearbox Block (GTH Transceiver)

The top-level RX gearbox in the GTX transceiver has one instance of each of the following components:

1. 64/66 4-byte gearbox
2. 64/66 2-byte gearbox
3. 64/67 4-byte gearbox
4. 64/67 2-byte gearbox
5. Sequence detector

To support the CAUI interface in the GTH transceiver, one instance of each 2-byte gearbox is added. One instance of the bit demux block is also added. The RXGEARBOXSLIP input signal is used for datastream A, while the RXSLIDE input signal is used as a gearbox slip input for datastream B. The widths of the RXDATAVALID, RXHEADER, RXHEADERVALID and RXSTARTOFSEQ outputs are doubled to accommodate the second data stream.

Figure 4-65 shows the CAUI interface (RX path) of the GTH transceiver.

Figure 4-65: **CAUI Interface (RX Datapath)**

In CAUI interface mode, the bit demux block splits the incoming data stream from the PMA into A and B streams. The block receives 32 bits of encoded data every cycle. All even bits are assigned to datastream A and all odd bits are assigned to datastream B.

Though RX_INT_DATAWIDTH = 1 (4-byte) is used in this mode, two 2-byte gearboxes are used to realize the functionality shown in Figure 4-65. The functionality of these 2-byte gearboxes are the same as described in the previous sections for the case when RX_INT_DATAWIDTH = 0 (2-byte).

If the PCSL data width is 32 bits each (RX_DATA_WIDTH = 64 (8-byte)), the 4-byte to 8-byte converter combines the data streams in such a way that datastreams A and B reach the corresponding PCSLs as shown in Figure 4-66 and Figure 4-67.

Figure 4-66: **Input to the 4-Byte to 8-Byte Converter (RX_DATA_WIDTH = 64 (8-Byte), RX_INT_DATAWIDTH = 1 (4-Byte), GEARBOX_MODE[2] = `1'b1`)**

Figure 4-67: **Output of the 4-Byte to 8-Byte Converter (RX_DATA_WIDTH = 64 (8-Byte), RX_INT_DATAWIDTH = 1 (4-Byte), GEARBOX_MODE[2] = `1'b1`)**

Due to the additional functionality, latency through the gearbox block is expected to be longer in the GTH transceiver when compared to the GTX transceiver.

FPGA RX Interface

Functional Description

The FPGA RX interface is the FPGA's gateway to the RX datapath of the GTX/GTH transceiver. Applications transmit data through the GTX/GTH transceiver by writing data to the RXDATA port on the positive edge of RXUSRCLK2. The width of the port can be configured to be two, four, or eight bytes wide. The actual width of the port depends on the RX_DATA_WIDTH and RX_INT_DATAWIDTH attributes and RX8B10BEN port setting. Port widths can be 16, 20, 32, 40, 64, and 80 bits. The rate of the parallel clock (RXUSRCLK2) at the interface is determined by the RX line rate, the width of the RXDATA port, and whether or not 8B/10B decoding is enabled. In some operating modes, a second parallel clock (RXUSRCLK) must be provided for the internal PCS logic in the transmitter. This section shows how to drive the parallel clocks and explains the constraints on those clocks for correct operation. The highest transmitter data rates require an 8-byte interface to achieve a RXUSRCLK2 rate in the specified operating range.

Interface Width Configuration

The 7 series GTX/GTH transceiver contains 2-byte and 4-byte internal datapaths and is configurable by setting the RX_INT_DATAWIDTH attribute. The FPGA interface width is configurable by setting the RX_DATA_WIDTH attribute. When the 8B/10B decoder is enabled, RX_DATA_WIDTH must be configured to 20 bits, 40 bits, or 80 bits, and in this case, the FPGA RX interface only uses the RXDATA ports. For example, RXDATA[15:0] is used when the FPGA interface width is 16. When the 8B/10B decoder is bypassed, RX_DATA_WIDTH can be configured to any of the available widths: 16, 20, 32, 40, 64, or 80 bits.

Table 4-50 shows how the interface width for the RX datapath is selected. 8B/10B decoding is described in more detail in RX 8B/10B Decoder, page 228.

Table 4-50: **FPGA RX Interface Datapath Configuration**

RX8B10BEN	RX_DATA_WIDTH	RX_INT_DATAWIDTH	FPGA Interface Width	Internal Data Width
1	20	0	16	20
	40	0	32	20
	40	1	32	40
	80	1	64	40
0	16	0	16	16
	20	0	20	20
	32	0	32	16
	32	1	32	32
	40	0	40	20
	40	1	40	40
	64	1	64	32
	80	1	80	40

When the 8B/10B decoder is bypassed and RX_DATA_WIDTH is 20, 40, or 80, the RXDISPERR and RXCHARISK ports are used to extend the RXDATA port from 16 to 20 bits, 32 to 40 bits, or 64 to 80 bits. Table 4-51 shows the data received when the 8B/10B decoder is disabled. When the RX gearbox is used, refer to RX Gearbox, page 275 for data transmission order.

Table 4-51: **RX Data Received When the 8B/10B Decoder is Bypassed**

| | < < < Data Reception is Right to Left (LSB to MSB) < < < | | |

$$RXUSRCLK\ Rate = \frac{Line\ Rate}{Internal\ Datapath\ Width} \qquad \text{Equation 4-2}$$

RXUSRCLK2 is the main synchronization clock for all signals into the RX side of the GTX/GTH transceiver. Most signals into the RX side of the GTX/GTH transceiver are sampled on the positive edge of RXUSRCLK2. RXUSRCLK2 and RXUSRCLK have a fixed-rate relationship based on the RX_DATA_WIDTH and RX_INT_DATAWIDTH settings. Table 4-52 shows the relationship between RXUSRCLK2 and RXUSRCLK per RX_DATA_WIDTH and RX_INT_DATAWIDTH values. A line rate greater than 6.6 Gb/s requires a 4-byte internal datapath to be used by setting RX_INT_DATAWIDTH to 1.

Table 4-52: **RXUSRCLK2 Frequency Relationship to RXUSRCLK**

FPGA Interface Width	RX_DATA_WIDTH	RX_INT_DATAWIDTH	RXUSRCLK2 Frequency
2-Byte	16, 20	0	$F_{RXUSRCLK2} = F_{RXUSRCLK}$
4-Byte	32, 40	0	$F_{RXUSRCLK2} = F_{RXUSRCLK} / 2$
4-Byte	32, 40	1	$F_{RXUSRCLK2} = F_{RXUSRCLK}$
8-Byte	64, 80	1	$F_{RXUSRCLK2} = F_{RXUSRCLK} / 2$

These rules about the relationships between clocks must be observed for RXUSRCLK and RXUSRCLK2:

- RXUSRCLK and RXUSRCLK2 must be positive-edge aligned, with as little skew as possible between them. As a result, low-skew clock resources (BUFGs and BUFRs) should be used to drive RXUSRCLK and RXUSRCLK2.
- If the channel is configured so the same oscillator drives the reference clock for the transmitter and the receiver, TXOUTCLK can be used to drive RXUSRCLK and RXUSRCLK2 in the same way that they are used to drive TXUSRCLK and TXUSRCLK2. When clock correction is turned off or the RX buffer is bypassed, RX phase alignment must be used to align the serial clock and the parallel clocks.
- If separate oscillators are driving the reference clocks for the transmitter and receiver on the channel, and clock correction is not used, RXUSRCLK and RXUSRCLK2 must be driven by RXOUTCLK (RXOUTCLKSEL = `3'b010` for RXOUTCLKPMA), and the phase-alignment circuit must be used.
- If clock correction is used, RXUSRCLK and RXUSRCLK2 can be sourced by RXOUTCLK or TXOUTCLK.

Ports and Attributes

Table 4-53 defines the FPGA RX interface ports.

Table 4-53: **FPGA RX Interface Ports**

Port	Dir	Clock Domain	Description
RXDISPERR[7:0]	Out	RXUSRCLK2	When 8B/10B decoding is disabled, RXDISPERR is used to extend the data bus for 20-bit, 40-bit and 80-bit RX interfaces.
RXCHARISK[7:0]	Out	RXUSRCLK2	When 8B/10B decoding is disabled, RXCHARISK is used to extend the data bus for 20-bit, 40-bit and 80-bit RX interfaces.
RXDATA[63:0]	Out	RXUSRCLK2	The bus for transmitting data. The width of this port depends on RX_DATA_WIDTH: RX_DATA_WIDTH = 16, 20: RXDATA[15:0] = 16 bits wide RX_DATA_WIDTH = 32, 40: RXDATA[31:0] = 32 bits wide RX_DATA_WIDTH = 64, 80: RXDATA[63:0] = 64 bits wide When a 20-bit, 40-bit, or 80-bit bus is required, the RXCHARISK and RXDISPERR ports from the 8B/10B encoder are concatenated with the RXDATA port. See Table 4-51, page 288.
RXUSRCLK	In	Clock	This port is used to provide a clock for the internal RX PCS datapath.
RXUSRCLK2	In	Clock	This port is used to synchronize the FPGA logic with the RX interface. This clock must be positive-edge aligned to RXUSRCLK when RXUSRCLK is provided by the user.

Table 4-54 defines the FPGA RX interface attributes.

Table 4-54: **FPGA RX Interface Attributes**

Attribute	Type	Description
RX_DATA_WIDTH	Integer	Sets the bit width of the RXDATA port. When 8B/10B encoding is enabled, RX_DATA_WIDTH must be set to 20, 40, or 80. Valid settings are 16, 20, 32, 40, 64, and 80. See Interface Width Configuration, page 287 for more details.
RX_INT_DATAWIDTH	Integer	Controls the width of the internal datapath. 0: 2-byte internal datapath 1: 4-byte internal datapath. Set to 1 if a line rate is greater than 6.6 Gb/s.

Chapter 5

Board Design Guidelines

Overview

Topics related to implementing a design on a printed circuit board that uses the 7 series FPGAs GTX/GTH transceivers are presented in this chapter. The GTX/GTH transceivers are analog circuits that require special consideration and attention when designing and implementing them on a printed circuit board. Besides an understanding of the functionality of the device pins, a design that performs optimally requires attention to issues such as device interfacing, transmission line impedance and routing, power supply design filtering and distribution, component selection, and PCB layout and stackup design.

Pin Description and Design Guidelines

GTX/GTH Transceiver Pin Descriptions

Table 5-1 defines the GTX/GTH transceiver Quad pins.

Table 5-1: **GTX/GTH Transceiver Quad Pin Descriptions**

Pins	Dir	Description
MGTREFCLK0P MGTREFCLK0N	In (Pad)	Differential clock input pin pair for the reference clock of the GTX/GTH transceiver Quad.
MGTREFCLK1P MGTREFCLK1N	In (Pad)	Differential clock input pin pair for the reference clock of the GTX/GTH transceiver Quad.
MGTXRXP[3:0]/MGTXRXN[3:0] MGTHRXP[3:0]/MGTHRXN[3:0]	In (Pad)	RXP and RXN are the differential input pairs for each of the receivers in the GTX/GTH transceiver Quad.
MGTXTXP[3:0]/MGTXTXN[3:0] MGTHTXP[3:0]/MGTHTXN[3:0]	Out (Pad)	TXP and TXN are the differential output pairs for each of the transmitters in the GTX/GTH transceiver Quad.
MGTAVTTRCAL	In (Pad)	Bias current supply for the termination resistor calibration circuit. See Termination Resistor Calibration Circuit.
MGTRREF	In (Pad)	Calibration resistor input pin for the termination resistor calibration circuit. See Termination Resistor Calibration Circuit.

Table 5-1: **GTX/GTH Transceiver Quad Pin Descriptions *(Cont'd)***

Pins	Dir	Description
MGTAVCC	In (Pad)	MGTAVCC is the analog supply for the internal analog circuits of the GTX/GTH transceiver Quad tile. This includes the analog circuits for the PLLs, transmitters, and receivers. Most packages have multiple groups of power supply connections in the package for MGTAVCC. Refer to the package pin definitions to identify in which power supply group a specific GTX/GTH transceiver Quad is located. The nominal voltage is 1.0 V_{DC}.
MGTAVTT	In (Pad)	MGTAVTT is the analog supply for the Transmitter and Receiver termination circuits of the GTX/GTH transceiver Quad tile. Most packages have multiple groups of power supply connections in the package for MGTAVTT. Refer to the package pin definitions to identify in which power supply group a specific GTX/GTH transceiver Quad is located. The nominal voltage is 1.2 V_{DC}.
MGTVCCAUX	In (Pad)	MGTVCCAUX is the auxiliary analog QPLL voltage supply for the transceivers. Most packages have multiple groups of power supply connections in the package for MGTVCCAUX. Refer to the package pin definitions to identify in which power supply group a specific GTX/GTH transceiver Quad is located. The nominal voltage is 1.8 V_{DC}.

Figure 5-1 shows the external power supply connections with the 7 series FPGA GTX/GTH transceivers and Figure 5-2, page 294 shows the internal power supply connections of the GTX/GTH transceiver.

Figure 5-1: **7 Series FPGAs GTX/GTH Transceivers External Power Supply Connections**

Note relevant to Figure 5-1:

1. The voltage values are nominal. See the respective 7 series FPGAs data sheet for values and tolerances.

Figure 5-2: **7 Series FPGAs GTX/GTH Transceivers Internal Power Supply Connections**

Termination Resistor Calibration Circuit

There is one resistor calibration circuit (RCAL) shared between all GTX/GTH transceiver Quad primitives in a GTX/GTH transceiver Quad column. The MGTAVTTRCAL and MGTRREF pins connect the bias circuit power and the external calibration resistor to the RCAL circuit. The RCAL circuit performs the resistor calibration only during configuration of the FPGA. Prior to configuration, all analog supply voltages must be present and within the proper tolerance as specified in the respective 7 series FPGAs data sheet.

The RCAL circuit is associated with the GTX/GTH transceiver Quad that is the RCAL master. The RCAL master performs the termination resistor calibration during configuration of the FPGA and then distributes the calibrated values to all of the GTX/GTH transceiver Quads in the column. The Quad in which the RCAL circuit is located must be powered on. See Table 5-2 for the Quad location of the MGTAVTTRCAL pin. For Stacked Silicon Interconnect (SSI) technology devices, each slice to be used (that contains multiple Quads) must be powered on.

The MGTAVTTRCAL pin should be connected to the MGTAVTT supply and to a pin on the 100Ω precision external resistor. The other pin of the resistor is connected to the MGTRREF pin. The resistor calibration circuit provides a controlled current load to the resistor connected to the MGTRREF pin. It then senses the voltage drop across the external calibration resistor and uses that value to adjust the internal resistor calibration setting.

The quality of the resistor calibration is dependent on the accuracy of the voltage measurement at the MGTAVTTRCAL and MGTRREF pins. To eliminate errors due to the voltage drop across the traces that lead from the resistor and to the FPGA pins, the trace from the MGTAVTTRCAL pin to the resistor should have the same length and geometry as the trace that connects the other pin of the resistor to the MGTRREF pin. (See the suggested layout in Figure 5-3.)

Figure 5-3: **PCB Layout for the RCAL Resistor**

Analog Power Supply Pins

The GTX/GTH transceiver Quad analog power supplies (MGTAVCC, MGTAVTT, and MGTVCCAUX) have planes inside the package. For some of the packages, there are multiple planes for each analog power supply. It there is more than one power supply group in the package, the power supply pin names have a _G# suffix that identifies which pins are associated with which power supply group. If all of the Quads in a power supply group are not used, the associated power pins can be left unconnected or tied to ground (unless the RCAL circuit is in that Quad).

For each GTX/GTH transceiver analog power supply group there are three power supplies (MGTAVCC, MGTAVTT, and MGTVCCAUX). If there are two power supply groups in a package, then there are a total of six power supply planes in the package for these groups, with three planes in the package for each power supply group.

Table 5-2 shows the power supply grouping in each of the packages for Kintex™-7 devices with GTX transceivers. It also specifies which Quad contains the RCAL calibration circuit.

Table 5-2: **Kintex-7 FPGA GTX Transceiver Power Supply Grouping Per Package**

	Kintex-7 FPGA GTX Transceiver Quads							
	MGT 111	MGT 112	MGT 113	MGT 114	MGT 115	MGT 116	MGT 117	MGT 118
XC7K70T-FBG484					Single Plane (RCAL)			
XC7K70T-FBG676					Single Plane (RCAL)	Single Plane		

Table 5-2: **Kintex-7 FPGA GTX Transceiver Power Supply Grouping Per Package *(Cont'd)***

	Kintex-7 FPGA GTX Transceiver Quads							
	MGT 111	**MGT 112**	**MGT 113**	**MGT 114**	**MGT 115**	**MGT 116**	**MGT 117**	**MGT 118**
XC7K160T-FBG484					Single Plane (RCAL)			
XC7K160T-FBG676					Single Plane (RCAL)	Single Plane		
XC7K160T-FFG676					Single Plane (RCAL)	Single Plane		
XC7K325T-FBG676					Single Plane (RCAL)	Single Plane		
XC7K325T-FBG900					Single Plane (RCAL)	Single Plane	Single plane	Single Plane
XC7K325T-FFG676					Single Plane (RCAL)	Single Plane		
XC7K325T-FFG900					Single Plane (RCAL)	Single Plane	Single plane	Single Plane
XC7K355T-FFG901		G10	G10	G10	G11 (RCAL)	G11	G11	
XC7K410T-FBG676					Single Plane (RCAL)	Single Plane		
XC7K410T-FBG900					Single Plane (RCAL)	Single Plane	Single Plane	Single Plane
XC7K410T-FFG676					Single Plane (RCAL)	Single Plane		
XC7K410T-FFG900					Single Plane (RCAL)	Single Plane	Single Plane	Single Plane
XC7K420T-FFG901	G10	G10	G10	G10	G11 (RCAL)	G11	G11	
XC7K420T-FFG1156	G10	G10	G10	G10	G11 (RCAL)	G11	G11	G11

Table 5-2: **Kintex-7 FPGA GTX Transceiver Power Supply Grouping Per Package *(Cont'd)***

	Kintex-7 FPGA GTX Transceiver Quads							
	MGT 111	**MGT 112**	**MGT 113**	**MGT 114**	**MGT 115**	**MGT 116**	**MGT 117**	**MGT 118**
XC7K480T-FFG901	G10	G10	G10	G10	G11 (RCAL)	G11	G11	
XC7K480T-FFG1156	G10	G10	G10	G10	G11 (RCAL)	G11	G11	G11

Table 5-3 shows the power supply grouping in each of the packages for Virtex®-7 devices with GTX/GTH transceivers.

Table 5-3: **Virtex-7 FPGA GTX/GTH Transceiver Power Supply Grouping Per Package**

		Virtex-7 FPGA GTX/GTH Transceiver Quads											
		MGT 210	**MGT 211**	**MGT 212**	**MGT 213**	**MGT 214**	**MGT 215**	**MGT 216**	**MGT 217**	**MGT 218**	**MGT 219**	**MGT 220**	**MGT 221**
		MGT 110	**MGT 111**	**MGT 112**	**MGT 113**	**MGT 114**	**MGT 115**	**MGT 116**	**MGT 117**	**MGT 118**	**MGT 119**	**MGT 120**	**MGT 121**
XC7V585T-FFG1157						G10	G10 (RCAL)	G10	G11	G11			
XC7V585T-FFG1761			G10	G10	G10	G10	G10 (RCAL)	G10	G11	G11	G11		
XC7V2000T-FHG1761			G10	G10 (RCAL)	G10	G10	G10 (RCAL)	G10	G11	G11 (RCAL)	G11		
XC7V2000T-FLG1925				G10 (RCAL)	G10	G11	G11 (RCAL)						
XC7VX330T-FFG1157						G10	G10 (RCAL)	G10	G11	G11			
XC7VX330T-FFG1761					G10	G10	G10 (RCAL)	G10	G11	G11	G11		
XC7VX415T-FFG1157						G10	G10 (RCAL)	G10	G11	G11			
XC7VX415T-FFG1158	Left-side					G20	G20 (RCAL)	G20	G21	G21	G21		
	Right-side					G10	G10 (RCAL)	G10	G11	G11	G11		
XC7VX415T-FFG1927	Left-side					G20	G20 (RCAL)	G20	G21	G21	G21		
	Right-side					G10	G10 (RCAL)	G10	G11	G11	G11		
XC7VX485T-FFG1157	Left-side												
	Right-side					G10	G10 (RCAL)	G10	G11	G11			

Table 5-3: **Virtex-7 FPGA GTX/GTH Transceiver Power Supply Grouping Per Package *(Cont'd)***

		Virtex-7 FPGA GTX/GTH Transceiver Quads											
		MGT 210	MGT 211	MGT 212	MGT 213	MGT 214	MGT 215	MGT 216	MGT 217	MGT 218	MGT 219	MGT 220	MGT 221
		MGT 110	MGT 111	MGT 112	MGT 113	MGT 114	MGT 115	MGT 116	MGT 117	MGT 118	MGT 119	MGT 120	MGT 121
XC7VX485T-FFG1761	Left-side												
	Right-side				G10	G10	G10 (RCAL)	G10	G11	G11	G11		
XC7VX485T-FFG1158	Left-side					G20	G20 (RCAL)	G20	G21	G21	G21		
	Right-side					G10	G10 (RCAL)	G10	G11	G11	G11		
XC7VX485T-FFG1927	Left-side				G20	G20	G20 (RCAL)	G20	G21	G21	G21		
	Right-side				G10	G10	G10 (RCAL)	G10	G11	G11	G11		
XC7VX485T-FFG1930	Left-side												
	Right-side				G10	G10	G10	G11 (RCAL)	G11	G11			
XC7VX550T-FFG1158	Left-side					G20	G20 (RCAL)	G20	G21	G21	G21		
	Right-side					G10	G10 (RCAL)	G10	G11	G11	G11		
XC7VX550T-FFG1927	Left-side	G19	G19	G19	G20	G20	G20 (RCAL)	G20	G21	G21	G21		
	Right-side	G9	G9	G9	G10	G10	G10 (RCAL)	G10	G11	G11	G11		
XC7VX690T-FFG1157	Left-side												
	Right-side					G10	G10 (RCAL)	G10	G11	G11			
XC7VX690T-FFG1761	Left-side												
	Right-side		G10	G10	G10	G10	G10 (RCAL)	G10	G11	G11	G11		
XC7VX690T-FFG1158	Left-side					G20	G20 (RCAL)	G20	G21	G21	G21		
	Right-side					G10	G10 (RCAL)	G10	G11	G11	G11		
XC7VX690T-FFG1926	Left-side		G20	G20	G21	G21	G21 (RCAL)	G22	G22	G22			
	Right-side		G10	G10	G11	G11	G11 (RCAL)	G12	G12	G12			

Table 5-3: **Virtex-7 FPGA GTX/GTH Transceiver Power Supply Grouping Per Package *(Cont'd)***

		Virtex-7 FPGA GTX/GTH Transceiver Quads											
		MGT 210	MGT 211	MGT 212	MGT 213	MGT 214	MGT 215	MGT 216	MGT 217	MGT 218	MGT 219	MGT 220	MGT 221
		MGT 110	MGT 111	MGT 112	MGT 113	MGT 114	MGT 115	MGT 116	MGT 117	MGT 118	MGT 119	MGT 120	MGT 121
XC7VX690T-FFG1927	Left-side	G19	G19	G19	G20	G20	G20 (RCAL)	G20	G21	G21	G21		
	Right-side	G9	G9	G9	G10	G10	G10 (RCAL)	G10	G11	G11	G11		
XC7VX690T-FFG1930	Left-side												
	Right-side				G10	G10	G10 (RCAL)	G11	G11	G11			
XC7VX980T-FFG1926	Left-side		G20	G20	G21	G21	G21 (RCAL)	G22	G22	G22			
	Right-side		G10	G10	G11	G11	G11 (RCAL)	G12	G12	G12			
XC7VX980T-FFG1928	Left-side	G20	G20	G20 (RCAL)	G21	G21	G21 (RCAL)	G22	G22	G22 (RCAL)			
	Right-side	G10	G10	G10 (RCAL)	G11	G11	G11 (RCAL)	G12	G12	G12 (RCAL)			
XC7VX980T-FFG1930	Left-side												
	Right-side				G10	G10	G10 (RCAL)	G11	G11	G11			
XC7VX1140T-FLG1926	Left-side		G20	G20 (RCAL)	G21	G21	G21 (RCAL)	G22	G22	G22 (RCAL)			
	Right-side		G10	G10 (RCAL)	G11	G11	G11 (RCAL)	G12	G12	G12 (RCAL)			
XC7VX1140T-FLG1928	Left-side	G20	G20	G20 (RCAL)	G21	G21	G21 (RCAL)	G22	G22	G22 (RCAL)	G23	G23	G23 (RCAL)
	Right-side	G10	G10	G10 (RCAL)	G11	G11	G11 (RCAL)	G12	G12	G12 (RCAL)	G13	G13	G13 (RCAL)
XC7VX1140T-FLG1930	Left-side												
	Right-side				G10	G10	G10 (RCAL)	G11	G11	G11 (RCAL)			
XCV7H580T-HCG1155	Left-side				G20	G20	G20 (RCAL)						
	Right-side				G10	G10	G10 (RCAL)						
XCV7H580T-HCG1931	Left-side				G21	G21	G21 (RCAL)	G22	G22	G22 (RCAL)			
	Right-side				G11	G11	G11 (RCAL)	G12	G12	G12 (RCAL)			

Table 5-3: **Virtex-7 FPGA GTX/GTH Transceiver Power Supply Grouping Per Package *(Cont'd)***

		Virtex-7 FPGA GTX/GTH Transceiver Quads											
		MGT 210	MGT 211	MGT 212	MGT 213	MGT 214	MGT 215	MGT 216	MGT 217	MGT 218	MGT 219	MGT 220	MGT 221
		MGT 110	MGT 111	MGT 112	MGT 113	MGT 114	MGT 115	MGT 116	MGT 117	MGT 118	MGT 119	MGT 120	MGT 121
XCV7H580T-HCG1932	Left-side				G21	G21	G21 (RCAL)	G22	G22	G22 (RCAL)			
	Right-side				G11	G11	G11 (RCAL)	G12	G12	G12 (RCAL)			
XCV7H870T-HCG1931	Left-side				G21	G21	G21 (RCAL)	G22	G22	G22 (RCAL)			
	Right-side				G11	G11	G11 (RCAL)	G12	G12	G12 (RCAL)			
XCV7H870T-HCG1932	Left-side	G20	G20	G20 (RCAL)	G21	G21	G21 (RCAL)	G22	G22	G22 (RCAL)			
	Right-side	G10	G10	G10 (RCAL)	G11	G11	G11 (RCAL)	G12	G12	G12 (RCAL)			

Reference Clock

Overview

This section focuses on the selection of the reference clock source or oscillator. An oscillator is characterized by:

- Frequency range
- Output voltage swing
- Jitter (deterministic, random, peak-to-peak)
- Rise and fall times
- Supply voltage and current
- Noise specification
- Duty cycle and duty-cycle tolerance
- Frequency stability

These characteristics are selection criteria when choosing an oscillator for a GTX/GTH transceiver design. Figure 5-4 illustrates the convention for the single-ended clock input voltage swing, peak-to-peak as used in the GTX/GTH transceiver portion of the respective 7 series FPGAs data sheet. This figure is provided to show the contrast to the differential clock input voltage swing calculation shown in Figure 5-5.

Figure 5-4: **Single-Ended Clock Input Voltage Swing, Peak-to-Peak**

Figure 5-5 illustrates the differential clock input voltage swing, peak-to-peak, which is defined as MGTREFCLKP – MGTREFCLKN.

Figure 5-5: **Differential Clock Input Voltage Swing, Peak-to-Peak**

Figure 5-6 shows the rise and fall time convention of the reference clock.

Figure 5-6: **Rise and Fall Times**

Figure 5-7 illustrates the internal details of the IBUFDS. The dedicated differential reference clock input pair MGTREFCLKP/MGTREFCLKN is internally terminated with 100Ω differential impedance. The common mode voltage of this differential reference clock input pair is 4/5 of MGTAVCC, or nominal 0.8V. See the respective 7 series FPGAs data sheet for exact specifications.

Figure 5-7: **MGTREFCLK Input Buffer Details**

Note relevant to Figure 5-7:

1. The resistor values are nominal. See the respective 7 series FPGAs data sheet for exact specifications.

GTX/GTH Transceiver Reference Clock Checklist

This criteria must be met when choosing an oscillator for a design with GTX/GTH transceivers:

- Provide AC coupling between the oscillator output pins and the dedicated GTX/GTH transceiver Quad clock input pins.
- Ensure that the differential voltage swing of the reference clock is the range as specified in DS182, *Kintex-7 FPGAs Data Sheet: DC and Switching Characteristics* and DS183, *Virtex-7 FPGAs Data Sheet: DC and Switching Characteristics*. The nominal range is 250 mV–2000 mV and the nominal value is 1200 mV).
- Meet or exceed the reference clock characteristics as specified in *Kintex-7 FPGAs Data Sheet: DC and Switching Characteristics* and *Virtex-7 FPGAs Data Sheet: DC and Switching Characteristics*.
- Meet or exceed the reference clock characteristics as specified in the standard for which the GTX/GTH transceiver provides physical layer support.
- Fulfill the oscillator vendor's requirement regarding power supply, board layout, and noise specification.
- Provide a dedicated point-to-point connection between the oscillator and GTX/GTH transceiver Quad clock input pins.
- Keep impedance discontinuities on the differential transmission lines to a minimum (impedance discontinuities generate jitter).

Reference Clock Interface

LVDS

Figure 5-8 shows how an LVDS oscillator is connected to a reference clock input of a GTX/GTH transceiver. (The GTX transceiver configuration is shown in the figure.)

Figure 5-8: **Interfacing an LVDS Oscillator to the 7 Series FPGAs GTX/GTH Transceiver Reference Clock Input**

LVPECL

Figure 5-9 shows how an LVPECL oscillator is connected to a reference clock input of a GTX/GTH transceiver. (The GTX configuration is shown in the figure.)

Figure 5-9: **Interfacing an LVPECL Oscillator to the 7 Series FPGAs GTX Transceiver Reference Clock Input**

Note relevant to Figure 5-9:

1. The resistor values are nominal. Refer to the oscillator data sheet for actual bias resistor requirement.

AC Coupled Reference Clock

AC coupling of the oscillator reference clock output to the GTX/GTH transceiver Quad reference clock inputs serves multiple purposes:

- Blocking a DC current between the oscillator and the GTX/GTH transceiver Quad dedicated clock input pins (which reduces the power consumption of both parts as well)
- Common mode voltage independence
- The AC coupling capacitor forms a high-pass filter with the on-chip termination that attenuates a wander of the reference clock

To minimize noise and power consumption, external AC coupling capacitors between the sourcing oscillator and the GTX/GTH transceiver Quad dedicated reference clock input pins are required.

Unused Reference Clocks

If the reference clock input is not used, the reference clock input pins should be left unconnected (both MGTREFCLKP and MGTREFCLKN).

Reference Clock Power

The GTX/GTH transceiver reference clock input circuit is powered by MGTAVCC. Excessive noise on this supply has a negative impact on the performance of any GTX/GTH transceiver Quad that uses the reference clock from this circuit.

Power Supply and Filtering

Overview

The 7 series FPGAs GTX/GTH transceiver Quad requires three analog power supplies: MGTAVCC at a nominal voltage level of 1.0 V_{DC}, MGTVCCAUX at a nominal voltage level of 1.8 V_{DC}, and MGTAVTT at a nominal voltage level of 1.2 V_{DC}. The pins for each of these analog power supplies are tied to a plane in the package. In some packages, there are two planes (a north plane and a south plane) for each of the analog power supplies. See Overview, page 291 for a discussion of the internal power planes in the 7 series FPGAs GTX/GTH transceiver packages.

Noise on the GTX/GTH transceiver analog power supplies can cause degradation in the performance of the transceivers. The most likely form of degradation is an increase in jitter at the output of the GTX/GTH transmitter and reduced jitter tolerance in the receiver. Sources of power supply noise are:

- Power supply regulator noise
- Power distribution network
- Coupling from other circuits

Each of these noise sources must be considered in the design and implementation of the GTX/GTH transceiver analog power supplies. The total peak-to-peak noise as measured at the input pin of the FPGA should not exceed 10 mVpk-pk.

Power Supply Regulators

Normally, the GTX/GTH transceiver analog voltage supplies have local power supply regulators that provide a final stage of voltage regulation. Preferably these regulators are placed as close as is feasible to the GTX/GTH transceiver power supply pins. Minimizing the distance between the analog voltage regulators and the GTX/GTH transceiver power supply pins reduces the opportunity for noise coupling into the supply after the regulator and for noise generated by current transients caused by load dynamics.

Linear versus Switching Regulators

The type of power supply regulator can have a significant impact on the complexity, cost, and performance of the power supply circuit. A power supply regulator must provide adequate power to the GTX/GTH transceiver with a minimum amount of noise while meeting the overall system thermal and efficiency requirements. There are two major types of power supply voltage regulators available for regulating the GTX/GTH transceiver analog voltage rails, linear regulators, and switching regulators. Each of these types of regulators has advantages and disadvantages. The optimal choice of regulator type depends on system requirements such as:

- Physical size
- Thermal budget
- Power efficiency
- Cost

Linear Regulator

A linear regulator is usually the simplest means to provide voltage regulation for the GTX/GTH transceiver analog supply rails. Inherently, a linear regulator does not inject significant noise into the regulated output voltage. In fact, some, not all, linear regulators provide noise rejection at the output from noise present on the voltage input. Another advantage of the linear regulator is that it usually requires a minimal number of external components to realize a circuit on the printed circuit board.

There are potentially two major disadvantages to linear regulators, minimum dropout voltage, and limited efficiency. Linear regulators require an input voltage that is higher than the output voltage. This minimum dropout voltage often is dependent on the load current. Even low dropout linear regulators require a minimum difference between the input voltage and the output voltage of the regulator. The system power supply design must consider the minimum dropout voltage requirements of the linear regulators.

The efficiency of a linear regulator is dependent on the voltage difference between the input and output of the linear regulator. For instance, if the input voltage of the regulator is 2.5 V_{DC} and the output voltage of the regulator is 1.2 V_{DC}, the voltage difference is 1.3 V_{DC}. Assuming that the current into the regulator is essentially equal to the current out of the regulator, the maximum efficiency of the regulator is 48%. This means that for every watt delivered to the load, the system must consume an additional watt for regulation. This power consumed by the regulator generates heat that must be dissipated by the system. Providing a means to dissipate the heat generated by the linear regulator can drive up the system cost. So even though from a simple component count and complexity cost, the linear regulator appears to have an advantage over the switching regulator, if the overall system cost is considered, including power consumption and heat dissipation, in high current applications, the linear regulator can actually be at a disadvantage.

Switching Regulator

A switching regulator can provide an efficient means to deliver a well-regulated voltage for the GTX/GTH transceiver analog power supply. Unlike the linear regulator, the switching regulator does not depend on the voltage drop between the input voltage of the regulator and the output voltage to provide regulation. Therefore the switching regulator can supply large amounts of current to the load while maintaining high power efficiency. It is not uncommon for a switching regulator to maintain efficiencies of 95% or greater. This efficiency is not severely impacted by the voltage drop between the input of the regulator and the output. It is impacted by the load current in a much lesser degree than that of the linear regulator. Because of the efficiency of the switching regulator, the system does not need to supply as much power to the circuit, and it does not need to provide a means to dissipate power consumed by the regulator.

The disadvantages to the switching regulator are complexity of the circuit and noise generated by the regulator switching function. Switching regulator circuits are usually more complex than linear regulator circuits. This shortcoming in switching regulators has recently been addressed by several switching regulator component vendors. Normally, a switching power supply regulation circuit requires a switching transistor element, an inductor, and a capacitor. Depending on the required efficiency and load requirements, a switching regulator circuit might require external switching transistors and inductors. Besides the component count, these switching regulators require very careful placement and routing on the printed circuit board to be effective.

Switching regulators generate significant noise and therefore usually require additional filtering before the voltage is delivered to the GTX/GTH transceiver analog power supply input of the 7 series FPGAs GTX/GTH transceiver. As the amplitude of the noise should be limited to less than 10 mVpp, the power supply filter should be designed to attenuate the noise from the switching regulator to meet this requirement.

Power Supply Distribution Network

Staged Decoupling

Die

The decoupling capacitance on the die filters the highest frequency noise components on the power supplies. The source for this very high frequency noise is the internal on-die circuits.

Package

The 7 series FPGAs package has additional decoupling. Decoupling capacitors in the package provide attenuation for noise in the package power plane, thereby reducing the interaction between GTX/GTH transceiver Quads. These capacitors in the package also aid in maintaining a low-impedance, high-frequency path between the power supply, MGTAVCC MGTVCCAUX, or MGTAVTT, and ground.

Printed Circuit Board

Because the impedance between the power planes and ground has been kept low on the die and in the package, the board design has a much more relaxed requirement for decoupling on the printed circuit board. The primary purpose of the PCB decoupling

capacitors is to provide noise isolation between the transceiver power supply pins and the external noise sources. Some examples of external noise sources are:

- Power supply regulator circuits
- On board digital switching circuits
- SelectIO signals from the FPGA

Decoupling capacitors should be provided on the PCB near the GTX/GTH transceiver power pins. These capacitors reduce the impedance of the PCB power distribution network. The reduced impedance of the PDN provides a means to attenuate noise from external sources before it can get into the device package power planes. The noise at the power pins should be less than 10 mVpp over the band from 10 kHz to 80 MHz.

The decoupling capacitor guidelines for the GTX/GTH transceivers are shown in Table 5-4. The GTX/GTH transceiver Quads are organized into power supply groups in the package. See Analog Power Supply Pins for the package being used.

Table 5-4: **GTX/GTH Transceiver PCB Capacitor Recommendations**

Quantity Per Group			Capacitance (µF)	Tolerance	Type
MGTAVCC	MGTAVTT	MGTVCCAUX			
1	1	1	4.70	10%	Ceramic

PCB Design Checklist

Table 5-5 is a checklist of items that can be used to design and review any 7 series FPGAs GTX/GTH transceiver PCB schematic and layout.

Table 5-5: **GTX/GTH Transceiver PCB Design Checklist**

Pins	Recommendations
MGTREFCLK0P MGTREFCLK0N MGTREFCLK1P MGTREFCLK1N	• Use AC coupling capacitors for connection to oscillator. • For AC coupling capacitors, see Reference Clock Interface, page 303. • Reference clock traces should be provided enough clearance to eliminate crosstalk from adjacent signals. • Reference clock oscillator output must comply with the minimum and maximum input amplitude requirements for these input pins. See DS182, *Kintex-7 FPGAs Data Sheet: DC and Switching Characteristics* or DS183, *Virtex-7 FPGAs Data Sheet: DC and Switching Characteristics*. • If reference clock input is not used, leave the associated pin pair unconnected.
MGTXRXP[3:0]/MGTXRXN[3:0] MGTHRXP[3:0]/MGTHRXN[3:0]	• Use AC coupling capacitors for connection to transmitter. The recommended value for AC coupling capacitors is 100 nF. • Receiver data traces should be provided enough clearance to eliminate crosstalk from adjacent signals. • If a receiver is not used, connect the associated pin pair to ground. • See RX Analog Front End, page 160.
MGTXTXP[3:0]/MGTXTXN[3:0] MGTHTXP[3:0]/MGTHTXN[3:0]	• Transmitter should be AC coupled to the receiver. The recommended value for the AC coupling capacitors is 100 nF. • Transmitter data traces should be provided enough clearance to eliminate crosstalk from adjacent signals. • If a transmitter is not used, leave the associated pin pair unconnected.

Table 5-5: **GTX/GTH Transceiver PCB Design Checklist *(Cont'd)***

Pins	Recommendations
MGTAVTTRCAL	• Connect to MGTAVTT and to a 100Ω resistor that is also connected to MGTRREF. Use identical trace geometry for the connection between the resistor and this pin and for the connection from the other pin of the resistor to MGTRREF. • See Termination Resistor Calibration Circuit, page 294.
MGTRREF	• Connect to a 100Ω resistor that is also connected to MGTAVTTRCAL. Use identical trace geometry for the connection between the resistor to this pin and for the connection from the other pin of the resistor to MGTAVTTRCAL. • See Termination Resistor Calibration Circuit, page 294.
MGTAVCC[N]	• The nominal voltage is 1.0 VDC. • See DS182, *Kintex-7 FPGAs Data Sheet: DC and Switching Characteristics* or DS183, *Virtex-7 FPGAs Data Sheet: DC and Switching Characteristics* for power supply voltage tolerances. • The power supply regulator for this voltage should not be shared with non-transceiver loads. • Many packages have multiple groups of power supply connections in the package for MGTAVCC. Refer to Table 5-2, page 295 to identify in which power supply group a specific GTX/GTH transceiver Quad is located. Information on pin locations for each package can be found in UG475, *7 Series FPGAs Packaging and Pinout Specifications*. • The following filter capacitor is recommended: • 1 of 4.7 µF 10% • For optimal performance, power supply noise must be less than 10 mVpp. • If all of the Quads in a power supply group are not used, the associated power pins can be left unconnected or tied to ground. • For power consumption, refer to the XPower Estimator (XPE) for 7 series devices at www.xilinx.com/power.

Table 5-5: **GTX/GTH Transceiver PCB Design Checklist *(Cont'd)***

Pins	Recommendations
MGTAVTT[N]	• The nominal voltage is 1.2 VDC. • See DS182, *Kintex-7 FPGAs Data Sheet: DC and Switching Characteristics* or DS183, *Virtex-7 FPGAs Data Sheet: DC and Switching Characteristics* for power supply voltage tolerances. • The power supply regulator for this voltage should not be shared with non-MGT loads. • Many packages have multiple groups of power supply connections in the package for MGTAVTT. Refer to Table 5-2, page 295 to identify in which power supply group a specific GTX/GTH transceiver Quad is located. Information on pin locations for each package can be found in UG475, *7 Series FPGAs Packaging and Pinout Specifications*. • The following ceramic filter capacitor is recommended: • 1 of 4.7 µF 10% • For optimal performance, power supply noise must be less than 10 mVpp. • If all of the Quads in a power supply group are not used, the associated power pins can be left unconnected or tied to ground. • For power consumption, refer to the XPower Estimator (XPE) for 7 series devices at www.xilinx.com/power.
MGTVCCAUX[N]	• The nominal voltage is 1.8 VDC. • See DS182, *Kintex-7 FPGAs Data Sheet: DC and Switching Characteristics* or DS183, *Virtex-7 FPGAs Data Sheet: DC and Switching Characteristics* for power supply voltage tolerances. • The power supply regulator for this voltage should not be shared with non-MGT loads. • Many packages have multiple groups of power supply connections in the package for MGTAVTT. Refer to Table 5-2, page 295 to identify in which power supply group a specific GTX/GTH transceiver Quad is located. For information on pin locations for each package, see UG475, *7 Series FPGAs Packaging and Pinout Specifications*. • The following filter capacitor is recommended: • 1 of 4.7 µF 10% • For optimal performance, power supply noise must be less than 10 mVpp. • If all of the QPLLs in this power supply group are not used but the Quads are used, the filter capacitors are not necessary and these pins can be connected to VCCAUX. • If all of the Quads in a power supply group are not used, the associated pins can be left unconnected or tied to ground.

Chapter 6

Use Model

PCI Express

This section provides the recommended guidelines to configure and use the GTX/GTH transceiver for PCI Express® applications. These recommended guidelines do not represent all possible PCI Express use modes. The use model described in this section is based on the PHY Interface for the PCI Express Architecture (PIPE).

Functional Description

The GTX/GTH transceiver supports PCI Express Gen1, Gen2, and Gen3 applications in x1 or multi-lane configurations. In PCI Express mode, the Gen1 line rate is 2.5 Gb/s, the Gen2 line rate is 5.0 Gb/s, and the Gen3 line rate is 8.0 Gb/s. The Channel PLL (CPLL) or Quad PLL (QPLL) can be used for Gen1 and Gen2 speeds. The Quad PLL (QPLL) is required for Gen3 speeds. The GTX/GTH transceiver includes PCS features, such as 8B/10B, comma alignment, channel bonding, and clock correction to support Gen1 and Gen2 applications. A custom soft Gen3 PCS block is required to support Gen3, which allows these GTX/GTH transceiver PCS features to be bypassed or disabled. A PIPE compatible wrapper can be generated from the 7 Series FPGAs Transceivers Wizard to configure the GTX/GTH transceiver for PCI Express applications. Table 6-1 shows the recommended GTX/GTH transceiver features for PCI Express applications. Refer to the specific sections of this user guide for all supported configurations of each feature. The PCI Express use modes described in this section are based on these recommendations.

Table 6-1: **Recommended GTX/GTH Transceiver Features for PCI Express**

Features	Gen1	Gen2	Gen3
CPLL	✓	✓	
QPLL	✓	✓	✓
2-byte Internal and 2-byte External Data Widths	✓	✓	
4-byte Internal and 4-byte External Data Widths			✓
TX Buffer			
RX Buffer	✓	✓	✓
Comma Alignment	✓	✓	
Channel Bonding	✓	✓	
Clock Correction	✓	✓	
8B/10B Encoder and Decoder	✓	✓	

Ports and Attributes

Table 6-2 and Table 6-3, page 314 show the commonly used GTX/GTH transceiver ports and attributes, respectively, for PCI Express applications.

Table 6-2: **PCI Express Ports**

Port	Dir	Clock Domain	Description
TXDETECTRX	In	TXUSRCLK2	Initiates the GTX/GTH transceiver to begin a receiver detection operation. Refer to the TX Receiver Detect Support for PCI Express Designs, page 155 for additional details on receiver detection. 0: Normal operation. 1: Receiver detection.
TXELECIDLE	In	TXUSRCLK2	Forces the TXP/TXN output to electrical idle when asserted in all power states. During TX electrical idle, both TXP and TXN are driven to the DC common mode voltage. 0: Normal operation. 1: TX electrical idle.
TXCHARDISPMODE[0]	In	TXUSRCLK2	Sets the running disparity to negative. Used when transmitting the PCI Express compliance pattern. In PCI Express applications, this signal is equivalent to TXCOMPLIANCE of the PIPE interface. 0: Normal operation. 1: TX compliance.
RXPOLARITY	In	RXUSRCLK2	Initiates the GTX/GTH transceiver to do a polarity inversion on the received data. 0: Normal operation. 1: Invert received data.
TXPD[1:0]	In	TXUSRCLK2	Powers up or down the TX and RX of the GTX/GTH transceiver. In PCI Express applications, TXPD and RXPD should be tied to the same source. `00b`: P0 power state for normal operation. `01b`: P0s power saving state with low recovery time latency. `10b`: P1 power saving state with longer recovery time latency. `11b`: P2 power saving state with lowest power and longest recovery time latency.
RXPD[1:0]	In	Async	

Table 6-2: **PCI Express Ports *(Cont'd)***

Port	Dir	Clock Domain	Description
TXRATE[2:0]	In	TXUSRCLK2	Dynamically controls the link signaling rate. In PCI Express applications, TXRATE and RXRATE must be tied to the same source. The QPLL must be used for Gen3 operation. It is recommended to set [TX/RX]RATE = `000b` and [TX/RX]OUT_DIV = 1 to enter Gen3 operation. `000b`: Divide by [TX/RX]OUT_DIV ([TX/RX]OUT_DIV is set to 2 to achieve divide by 2 for Gen1). `001b`: Divide by 1 (recommended for Gen2). `010b`: Divide by 2. `011b`: Divide by 4. `100b`: Divide by 8. `101b`: Divide by 16. `110b`: Divide by 1. `111b`: Divide by 1.
RXRATE[2:0]	In	RXUSRCLK2	
TXDEEMPH	In	Async	Selects TX de-emphasis when PCI Express is in Gen1 or Gen2 modes. 0: 6.0 dB de-emphasis. 1: 3.5 dB de-emphasis.
TXMARGIN[2:0]	In	Async	Selects TX voltage levels. Refer to the TX Configurable Driver Port section of this user guide for TXMARGIN settings and mappings in PCI Express mode. `000b`: Programmable. Normal operating range. `001b`: Programmable. `010b`: Programmable. `011b`: Programmable. `100b`: Programmable. `101b`: Programmable. `110b`: Programmable. `111b`: Programmable.
TXSWING	In	Async	Controls TX voltage swing level in Gen1 or Gen2 modes. 0: Full swing. 1: Low or half swing.
RXVALID	Out	RXUSRCLK2	Indicates symbol lock and valid data on RXDATA and RXCHARISK in Gen1 or Gen2 modes.
PHYSTATUS	Out	RXUSRCLK2	Used to communicate completion of several GTX/GTH transceiver functions including power management state transitions, rate change, and receiver detection.

Table 6-2: **PCI Express Ports *(Cont'd)***

Port	Dir	Clock Domain	Description
RXELECIDLE	Out	Async	Indicates RX detection of an electrical idle. In Gen3 line rate, the PCI Express MAC must use logic to detect and infer entering RX electrical idle instead of relying on the RXELECIDLE signal. While in RX electrical idle, use RXELECIDLE de-assertion to exit electrical idle. It is also recommended to infer entering RX electrical idle in Gen2. 0: Normal operation. 1: RX electrical idle.
RXSTATUS[2:0]	Out	RXUSRCLK2	Encodes RX status and error codes for the RX data stream when receiving data in Gen1 or Gen2 modes. `000b`: Received data OK. `001b`: 1 SKP added. `010b`: 1 SKP removed. `011b`: Receiver detected. `100b`: 8B/10B decode error. `101b`: Elastic buffer overflow. `110b`: Elastic buffer underflow. `111b`: Receive disparity error.

Table 6-3: **PCI Express Attributes**

Attribute	Type	Description
PCS_PCIE_EN	Boolean	Enables the PCS to PCI Express mode. FALSE: Non PCI Express mode. TRUE: PCI Express mode.
TX_DRIVE_MODE	String	Sets the TX drive mode. DIRECT: Non PCI Express applications. PIPE: Used for PCI Express applications in Gen1 or Gen2 modes. TX driver settings are controlled by TXDEEMPH, TXMARGIN, and TXSWING. PIPEGEN3: Used for PCI Express applications in Gen3 mode.

PCI Express Use Mode

The recommended GTX/GTH transceiver power-on default setting for PCI Express applications is configured for Gen1. PCI Express powers up in Gen1 before entering Gen2 or Gen3 mode. To reduce latency and minimize TX lane-to-lane skew, the TX buffer should be bypassed. In Gen3 speeds, some PCS features, such as comma alignment, channel bonding, clock correction, and 8B/10B can be bypassed or disabled when a custom soft Gen3 PCS block is used. Table 6-4 shows the recommended GTX/GTH transceiver settings for PCI Express applications at Gen1, Gen2, and Gen3 speeds.

Table 6-4: **Recommended GTX/GTH Transceiver Settings for PCI Express**

GTX/GTH Transceiver Setting	Gen1	Gen2	Gen3
PCS_PCIE_EN	TRUE	TRUE	TRUE
TX_DRIVE_MODE	PIPE	PIPE	PIPEGEN3
[TX/RX]RATE[2:0] (setting when CPLL is used for Gen1 and Gen2)	000b (must set [TX/RX]OUT_DIV to 2)	001b (divide by 1)	000b (must set [TX/RX]OUT_DIV to divide by 1)
[TX/RX]RATE[2:0] (setting when QPLL is use for Gen1 and Gen2)	000b (must set [TX/RX]OUT_DIV to divide by 4)	010b (divide by 2)	000b (must set [TX/RX]OUT_DIV to divide by 1)
[TX/RX]USRCLK	125 MHz	250 MHz	250 MHz
[TX/RX]USRCLK2	125 MHz	250 MHz	250 MHz
[TX/RX]_DATA_WIDTH	20	20	32
[TX/RX]_INT_DATAWIDTH	0	0	1
TXBUF_EN	FALSE	FALSE	FALSE
RXBUF_EN	TRUE	TRUE	TRUE
TX_XCLK_SEL	TXUSR	TXUSR	TXUSR
RX_XCLK_SEL	RXREC	RXREC	RXREC
TXOUTCLKSEL[2:0]	011b	011b	011b
RXOUTCLKSEL[2:0]	010b	010b	010b
CLK_CORRECT_USE	TRUE	TRUE	FALSE
RXCOMMADETEN	1	1	0
RXCHBONDEN	1 (set to 0 for x1 configurations)	1 (set to 0 for x1 configurations)	0
[TX/RX]8B10BEN	1	1	0

PIPE Control Signal

Table 6-5 is a look-up table for the [TX/RX]PD, TXDETECTRX, and TXELECIDLE control signals. The look-up table describes the GTX/GTH transceiver's mode or behavior based on the decoding of these signals. TXELECIDLE should always be asserted in P0s and P1 states. During TX electrical idle, the GTX/GTH transceiver is not transmitting data. The P2 power saving state is not a recommended use mode.

Table 6-5: **PIPE Control Signal Look-up Table**

[TX/RX]PD	TXDETECTRX	TXELECIDLE	Description
00b (P0 power state)	0	0	Normal operation.
	0	1	TX electrical idle.
	1	0	Loopback mode.
	1	1	Not supported for PIPE. Illegal condition.
01b (P0s power saving state)	Don't Care	0	Not supported for PIPE. PHY behavior is undefined if TXELECIDLE is deasserted while in P0s or P1 power state.
	Don't Care	1	TX electrical idle.
10b (P1 power saving state)	Don't Care	0	Not supported for PIPE. PHY behavior is undefined if TXELECIDLE is deasserted while in P0s or P1 power state.
	0	1	Electrical idle.
	1	1	Receiver detection.
11b (P2 power saving state)	Don't Care	0	Transmit beacon.
		1	Electrical idle.

PCI Express Clocking

Reference Clock

The GTX/GTH transceiver uses the reference clock to generate internal bit rate clocks for transmitting and receiving data. In PCI Express mode, the recommended reference clock is 100 MHz. The 125 MHz or 250 MHz reference clocks are also supported. The reference clock feeds into an IBUFDS_GTE2 that drives the GTX/GTH transceiver's reference clock. If the TX buffer is bypassed, TXOUTCLKSEL must select the GTX/GTH transceiver's reference clock as the source of TXOUTCLK. The reference clock must be stable and free running after power-on. For asynchronous clocking applications, its worst-case frequency offset must be within 600 ppm or ±300 ppm. The GTX/GTH transceiver has an internal 25 MHz clock derived from the reference clock from the [TX/RX]_CLK25_DIV setting. The 25 MHz clock is used as a synchronizer and timer for various PCI Express operations, such as reset, power management, rate change, and beacon. Table 6-6 shows the recommended CPLL and clock divider settings for PCI Express mode.

Table 6-6: **Recommended CPLL Divider and Clock Settings for PCI Express Reference Clocks**

GTX/GTH Transceiver Settings	100 MHz	125 MHz	250 MHz
CPLL_REFCLK_DIV	1	1	1
CPLL_FBDIV_45	5	5	5
CPLL_FBDIV	5	4	2
[TX/RX]OUT_DIV[(1)]	2 (Gen1) 1 (Gen2)	2 (Gen1) 1 (Gen2)	2 (Gen1) 1 (Gen2)
[TX/RX]OUT_DIV (setting when QPLL is used for Gen1 and Gen2)	4 (Gen1) 2 (Gen2)	4 (Gen1) 2 (Gen2)	4 (Gen1) 2 (Gen2)
[TX/RX]_CLK25_DIV	4	5	10

Notes:

1. [TX/RX]OUT_DIV is used to select between 2.5 Gb/s or 5. Gb/s line rate.

For Gen3 applications, QPLL must be used. Table 6-7 shows the recommended QPLL divider setting for PCI Express mode.

Table 6-7: **Recommended QPLL Divider Settings for PCI Express Reference Clocks**

QPLL Dividers	100 MHz	125 MHz	250 MHz
QPLL_FBDIV (setting when QPLL at 8 Gb/s is used for Gen3 only)	`10'b0100100000`	`10'b0011100000`	`10'b0001100000`
QPLL_FBDIV (setting when QPLL at 10Gb/s is used for Gen1 and Gen2)	`10'b0101110000`	`10'b0100100000`	`10'b0010000000`
QPLL_FBDIV_RATIO	1	1	1

Parallel Clock (PCLK)

PCLK is an FPGA logic (fabric) interface. In PCI Express mode, PCLK is the parallel interface clock used to synchronize data transfer across the parallel interface. The recommended PCLK frequency is 125 MHz in Gen1 and 250 MHz in Gen2 and Gen3. A MMCME2_ADV can be used to generate the 125 MHz and 250 MHz clock from the reference clock. When the MMCME2_ADV is used, the user can select the desire PCLK frequency using a BUFGMUX. It is recommended that the internal and external data widths be the same in PCI Express mode. When the internal and external data widths are the same, the [TX/RX]USRCLK and [TX/RX]USRCLK2 frequencies are equivalent.

It is recommended that the *TX* buffer is *bypassed* in Gen1, Gen2, and Gen3. When the TX buffer is bypassed, TXOUTCLKSEL must select the GTX/GTH transceiver reference clock as the source of TXUSRCLK2 through TXOUTCLK. To minimize TX lane-to-lane skew in multi-lane applications, the source of TXUSRCLK and TXUSRCLK2 for all lanes must come from the same source. Refer to TX Buffer Bypass, page 127 for additional details.

It is recommended that the *RX* buffer is *used* in Gen1, Gen2, and Gen3. When the RX buffer is used in *synchronous* clocking applications, the GTX/GTH transceiver reference clock is the source for both RXUSRCLK and RXUSRCLK2. Figure 6-1 shows a PCI Express

synchronous clocking architecture example. For Kintex-7 devices, it is recommended that a BUFG be used between the TXOUTCLK and MMCME2_ADV path. For Virtex-7 devices, this BUFG is not necessary.

Figure 6-1: **PCI Express Synchronous Clocking Architecture Example**

When the RX buffer is used in *asynchronous* clocking applications, the RX recovered clock from the master lane 0 is the source for both RXUSRCLK and RXUSRCLK2 when entering Gen3. Figure 6-2 shows a PCI Express asynchronous clocking architecture example. A BUFG should be inserted in the PCLK path to balance the timing if cascaded BUFGMUXes are used. Refer to RX Buffer Bypass, page 233 for additional details.

Figure 6-2: **PCI Express Asynchronous Clocking Architecture Example**

PCI Express Reset

After initial configuration and power on, the user should hold the GTX/GTH transceiver in reset until power and the reference clock are stable. PCI Express reset should be

performed in TX electrical idle and P1 power-down state. These are the recommended GTX/GTH transceiver settings during reset:

- TXELECIDLE = 1
- [TX/RX]PD[1:0] = `10b`
- TXDETECTRX = 0
- RXPOLARITY = 0
- TXMARGIN[2:0] = `000b`
- TXDEEMPH = 1
- TXSWING = 0

The GTX/GTH transceiver powers up in Gen1. After reset, PCLK must be stable. The timing diagram in Figure 6-3 shows a PCI Express reset example. In the figure, user reset, TX sync done, and gated PHYSTATUS are example fabric user signals.

Figure 6-3: **PCI Express Reset Example**

Notes relevant to Figure 6-3:

1. The sequence of events in Figure 6-3 is not drawn to scale.
2. When a user reset is detected, reset the CPLL and GTX/GTH transceiver by asserting CPLLRESET and GTX/GTH[TX/RX]RESET, respectively. Continue to hold CPLLRESET High for at least one reference clock cycle and until CPLLLOCK goes Low. Continue to hold the GTX/GTH transceiver in reset.
3. Wait until CPLLLOCK = 1 before releasing the GTX/GTH transceiver reset.
4. If the MMCM is used, ensure that the MMCM is locked before asserting [TX/RX]USERRDY.
5. Wait for [TX/RX]RESETDONE = 1 and PHYSTATUS = 0.
6. Perform TX sync (phase and delay alignment) if the TX buffer is bypassed.
7. Deassert the gated PHYSTATUS after the TX sync is done. The gated PHYSTATUS is a delayed version of the GTX/GTH transceiver's raw PHYSTATUS. After the gated PHYSTATUS is deasserted, the GTX/GTH transceiver successfully completes reset and is ready for normal operation.

PCI Express Power Management

To minimize power consumption, the GTX/GTH transceivers can enter PCI Express power-down states using the [TX/RX]PD ports. In PCI Express mode, TXPD and RXPD are recommended to be tied together. The available power states defined by the PIPE specification are P0, P0s, P1, and P2. P0 is the normal power state. PCLK is operational in all states, except in P2. PHYSTATUS assertion for a PCLK cycle indicates a successful power state transition in response to a change in [TX/RX]PD, and is asynchronous only in the P2 state. The valid power transitions for PCIe are:

- P0 to P0s
- P0s to P0
- P0 to P1
- P1 to P0
- P0 to P2
- P2 to P0

For PCI Express applications in Gen1 and Gen2 modes, the CPLL is used, and the QPLL is recommended to be powered down to save power. In Gen3 mode, the QPLL is required and the CPLL is recommended to be powered down to save power. The use mode timing diagram in Figure 6-4 shows an example of a PCI Express power down from the P0 to the P0s state.

Figure 6-4: **PCI Express P0 to P0s Power Transition Example**

Notes relevant to Figure 6-4:

1. The sequence of events in Figure 6-4 is not drawn to scale.
2. Change [TX/RX]PD from `00b` (P0) to `01b` (P0s).
3. Wait for PHYSTATUS = 1 pulse to indicate successful power transition.
4. Continue to hold [TX/RX]PD until another power change is requested.

PCI Express Rate Change

Rate Change between Gen1 and Gen2 Speeds

After powering up the GTX/GTH transceiver in Gen1 operation, the user can perform a PCI Express rate change to enter Gen2. PCI Express rate changes between Gen1 and Gen2 speeds are performed by changing the PCLK frequency and dynamically changing the GTX/GTH transceiver output divider. The datapath width is fixed for rate changes between Gen1 and Gen2 speeds. All PCI Express rate changes should be performed after entering TX electrical idle, and the P0 or P1 power down states. These are the recommended GTX/GTH transceiver settings before entering rate change:

- TXELECIDLE = 1
- [TX/RX]PD = `00b` (P0) or `10b` (P1)

The use mode timing diagram in Figure 6-5 shows an example of a PCI Express rate change from Gen1 to Gen2 speed. In the figure, user rate, PCLK, TX sync done, and gated PHYSTATUS are example fabric user signals. PCLK is the source of [TX/RX]USRCLK2 and [TX/RX]USRCLK. It should be changed back to 125 MHz when entering Gen1 from Gen2 speed.

Figure 6-5: **PCI Express Gen1 to Gen2 Rate Change Example**

Notes relevant to Figure 6-5:

1. The sequence of events in Figure 6-5 is not drawn to scale.
2. After a user rate change is detected, wait at least 16 PCLK cycles to flush out any valid TXDATA before changing the PCLK frequency to 250 MHz.
3. Change the GTX/GTH transceiver's rate to the Gen2 speed by setting [TX/RX]RATE = `001b`. This changes the CPLL output divider to 1 for the Gen2 speed. The CPLL output divider is 2 for the Gen1 speed.
4. Wait for [TX/RX]RATEDONE = 1 and PHYSTATUS = 1 pulses.
5. Perform TX sync (phase and delay alignment) if the TX buffer is bypassed.
6. Assert gated PHYSTATUS after the TX sync (phase and delay alignment) is done. The gated PHYSTATUS is a delayed version of the GTX/GTH transceiver's raw PHYSTATUS. After the gated PHYSTATUS asserts to indicate a successful rate change completion, RXDATA, RXSTATUS, and RXVALID should not be used until RXELECIDLE exit and RX CDR lock conditions are achieved.

Rate Change to Enter or Exit Gen3 Speed

In Gen1 or Gen2, the user can perform a PCI Express rate change to enter Gen3. The PCI Express rate change to enter or exit Gen3 is an advanced feature that requires bypassing the GTX/GTH transceiver PCS using DRP, changing the data width, and switching between the CPLL and QPLL. The use mode timing diagram in Figure 6-6 shows an example of a PCI Express rate change from Gen2 to Gen3 when the CPLL is used in Gen1 and Gen2. When entering Gen3, the CPLL should be powered down to save power. To power down the CPLL, it is recommended to assert CPLLRESET before asserting CPLLPD after successful Gen3 entry. When exiting Gen3, the QPLL should be powered down to save power if it is not used in Gen1 and Gen2. To power down the QPLL, it is recommend to assert QPLLRESET before asserting QPLLPD after successful Gen3 exit. The CPLL should be powered down when QPLL is used for Gen1, Gen2, and Gen3.

These are the recommended settings to select CPLL and QPLL for PCI Express applications:

- [TX/RX]SYSCLKSEL = `00b` selects the CPLL and uses the reference clock routed from the GTX/GTH transceiver channel.
- [TX/RX]SYSCLKSEL = `01b` selects the QPLL and uses the reference clock routed from the GTX/GTH transceiver channel.

In Figure 6-6, user rate, channel DRP done, PCLK, TX sync done, gated PHYSTATUS, and RX sync done are example fabric user signals. The PCLK frequency should be changed back to 125 MHz when entering Gen1 from Gen3 speed. It should be kept at 250 MHz when entering Gen2 from Gen3 speed.

Figure 6-6: **PCI Express Gen2 to Gen3 Rate Change Example**

Notes relevant to Figure 6-6:

1. The sequence of events in Figure 6-6 is not drawn to scale.
2. After a user rate change is detected, power up the QPLL by deasserting QPLLPD and QPLLRESET and wait until QPLLLOCK = 1.
3. Reset the GTX/GTH transceiver's PMA by asserting [TX/RX]PMARESET before switching from CPLL to QPLL by setting [TX/RX]SYSCLKSEL = `01b`. Hold the PMA in reset.
4. Use the DRP to bypass the PCS to configure the GTX/GTH transceiver for raw mode. Release the GTX/GTH transceiver reset and wait until [TX/RX]RESETDONE = 1.

5. Keep PCLK = 250 MHz when entering Gen3 from Gen2 speed. Change the GTX/GTH transceiver's rate to Gen3 speed by setting [TX/RX]RATE to `000b` and wait for the [TX/RX]RATEDONE = 1 and PHYSTATUS = 1 pulses.
6. Power down the CPLL by asserting CPLLRESET and CPLLPD to save power. Perform TX sync (phase and delay alignment) and assert the gated PHYSTATUS when the TX sync is done. The gated PHYSTATUS is a delayed version of the GTX/GTH transceiver's raw PHYSTATUS.
7. If the RX buffer is bypassed, the RX sync (phase and delay alignment) must be performed. Wait until RX electrical idle has exited and RX CDR is locked before performing the RX sync alignment. RXDATA is ready for processing after RX sync is done. It is recommended to perform a rate change to Gen3 in P0 power state when the RX buffer is bypassed.

Using DRP During Rate Change to Enter or Exit Gen3 Speed

To enter Gen3 operation, the GTXE2_CHANNEL's or GTHE2_CHANNEL's DRP must be used to bypass the PCS. To exit Gen3, the DRP must be used to re-enable the PCS features. When reset occurs in Gen3, DRP access is required to re-enable the PCS for Gen1. DRP should be performed while the PMA is in reset during rate change. Table 6-8 shows the DRP address and data for the GTX/GTH transceiver attributes that should be updated by the DRP. The user should perform read-modify-write DRP operations to ensure that only the targeted attributes are changed.

Table 6-8: **DRP Look-up Table to Enter or Exit Gen3 Speeds**

GTX/GTH Transceiver Attributes	Description	DRP Address	DRP Data	Gen1 and Gen2	Gen 3
TXOUT_DIV	TX Output Divider	`088h`	[6:4]	`001b` Divide by 2(1)	`000b` Divide by 1(1)
RXOUT_DIV	RX Output Divider	`088h`	[2:0]	`001b` Divide by 2(1)	`000b` Divide by 1(1)
TX_DATA_WIDTH	TX External Data Width	`06Bh`	[2:0]	`011b` 2-byte	`100b` 4-byte
TX_INT_DATAWIDTH	TX Internal Data Width	`06Bh`	[4]	0 2-byte	1 4-byte
RX_DATA_WIDTH	RX External Data Width	`011h`	[13:11]	`011b` 2-byte	`100b` 4-byte
RX_INT_DATAWIDTH	RX External Data Width	`011h`	[14]	0 2-byte	1 4-byte
TXBUF_EN	TX Buffer Enable	`01Ch`	[14]	0 Bypass TX Buffer	0 Bypass TX Buffer
RXBUF_EN	RX Buffer Enable	`09Dh`	[1]	1 Use RX Buffer	1 Use RX Buffer
TX_XCLK_SEL	TX XCLK Select	`059h`	[7]	1 TXUSR	1 TXUSR
RX_XCLK_SEL	RX XCLK Select	`059h`	[6]	0 RXREC	0 RXREC

Table 6-8: **DRP Look-up Table to Enter or Exit Gen3 Speeds *(Cont'd)***

GTX/GTH Transceiver Attributes	Description	DRP Address	DRP Data	Gen1 and Gen2	Gen 3
CLK_CORRECT_USE	Use Clock Correction	044h	[14]	1 TRUE	0 FALSE
TX_DRIVE_MODE	TX Drive Mode	019h	[4:0]	00001b PIPE	00010b PIPEGEN3

1. The GTX/GTH transceiver internally selects the TX and RX output dividers to use divide by 1 when the [TX/RX]RATE port is used to change the rate to Gen2 speed. The recommended setting in the table is used to enter and exit Gen3 speed.

PCI Express Channel Bonding

To perform RX lane-to-lane deskew, the GTX/GTH transceiver's channel bonding feature is required for multi-lane PCI Express applications in Gen1 and Gen2 modes. In Gen3 mode, the channel bonding feature can be disabled when a custom soft Gen3 PCS block is used. RXCHBONDEN should be set to 0 to disable the channel bonding feature in Gen3 mode and for x1 PCI Express applications. RXCHANBONDMASTER and RXCHANBONDSLAVE should be set to 0 in Gen3 when channel bonding is disabled. The GTX/GTH channel bonding input ports can be tied to zeros when channel bonding is always disabled.

The three channel bonding examples are described in:

- One-Hop Example
- Daisy-Chain Example
- Binary-Tree Example

One-Hop Example

The one-hop channel bonding structure is recommended to reduce RX latency. In one-hop channel bonding, lane 0 is configured as the master and is directly connected to each slave. Figure 6-7 shows a PCI Express one-hop channel bonding example.

Figure 6-7: **PCI Express One-Hop Channel Bonding Example**

Daisy-Chain Example

The daisy-chain channel bonding structure is recommended to improve timing. In daisy-chain channel bonding, lane 0 is configured as the master, and each slave is pipelined into a daisy chained structure. Figure 6-8 shows a PCI Express daisy-chain channel bonding example, where N represents the total number of PCI Express lanes.

Figure 6-8: **PCI Express Daisy-Chain Channel Bonding Example**

Binary-Tree Example

To balance between reducing RX latency and improving timing, a binary-tree channel bonding structure can be used. In binary-tree channel bonding, each GTX/GTH transceiver channel is connected to at most to two PCI Express lanes. Figure 6-9 shows a PCI Express binary-tree channel bonding example.

Figure 6-9: **PCI Express Binary-Tree Channel Bonding Example**

Channel Bonding Attribute Settings

In PCI Express applications, channel bonding is used to perform RX lane-to-lane deskew with the TS1 and TS2 ordered set during link training, or with the fast training sequence (FTS) ordered set during L0s exit. Table 6-9 shows the recommended channel bonding attribute settings for PCI Express. Refer to RX Channel Bonding, page 263 for additional details.

Table 6-9: **Channel Bonding Attributes for PCI Express**

Attribute	Type	Description
CHAN_BOND_KEEP_ALIGN	Boolean	Channel bonding keep align. Set to TRUE to preserve alignment for PCI Express applications.
CHAN_BOND_MAX_SKEW	Integer	Channel bonding max skew. Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
CHAN_BOND_SEQ_LEN	Integer	Channel bonding sequence length. Set to 4 for PCI Express applications.
CHAN_BOND_SEQ_1_ENABLE	4-bit Binary	Channel bonding sequence 1 enable for CHAN_BOND_SEQ_1_[1/2/3/4]. Set to `1111b` for PCI Express applications.
CHAN_BOND_SEQ_1_1	10-bit Binary	Set to `0001001010b` to use training sequence 1 (TS1) for PCI Express applications.
CHAN_BOND_SEQ_1_2	10-bit Binary	Set to `0001001010b` to use TS1 for PCI Express applications.
CHAN_BOND_SEQ_1_3	10-bit Binary	Set to `0001001010b` to use TS1 for PCI Express applications.
CHAN_BOND_SEQ_1_4	10-bit Binary	Set to `0110111100b` to use comma (COM) for PCI Express applications.
CHAN_BOND_SEQ_2_USE	Boolean	Use channel bonding sequence 2. Set to TRUE to enable training sequence 2 (TS2) for PCI Express applications.
CHAN_BOND_SEQ_2_ENABLE	4-bit Binary	Channel bonding sequence 2 enable for CHAN_BOND_SEQ_2_[1/2/3/4]. Set to `1111b` for PCI Express applications.
CHAN_BOND_SEQ_2_1	10-bit Binary	Set to `0001000101b` to use TS2 for PCI Express applications.
CHAN_BOND_SEQ_2_2	10-bit Binary	Set to `0001000101b` to use TS2 for PCI Express applications.
CHAN_BOND_SEQ_2_3	10-bit Binary	Set to `0001000101b` to use TS2 for PCI Express applications.

Table 6-9: **Channel Bonding Attributes for PCI Express *(Cont'd)***

Attribute	Type	Description
CHAN_BOND_SEQ_2_4	10-bit Binary	Set to `0110111100b` to use COM for PCI Express applications.
FTS_DESKEW_SEQ_ENABLE	4-bit Binary	FTS deskew sequence enable for FTS_LANE_DESKEW_CFG. Set to `1111b` for PCI Express applications.
FTS_LANE_DESKEW_EN	Boolean	FTS lane deskew enable. Set to TRUE to enable FTS for PCI Express applications.
FTS_LANE_DESKEW_CFG	4-bit Binary	FTS lane deskew configuration. Set to `1111b` for PCI Express applications.

PCI Express Clock Correction

To perform RX clock compensation, the GTX/GTH transceiver's clock correction feature is recommended for PCI Express applications in Gen1 and Gen2 modes. In Gen3 mode, the clock correction feature can be disabled when a custom soft Gen3 PCS block is used. In PCI Express applications, clock correction is used perform RX clock compensation with the SKP ordered set. The clock correction feature inserts or removes an SKP symbol from the RX elastic buffer to compensate for clock differences up to 600 ppm or ±300 ppm. Table 6-10 shows the recommended clock correction attribute settings for PCI Express. Refer to RX Clock Correction, page 253 for additional details.

Table 6-10: **Clock Correction Attributes for PCI Express**

Attribute	Type	Description
CBCC_DATA_SOURCE_SEL	String	Channel bonding and clock correction data source select. Set to DECODED to use data from the 8B/10B decoder for PCI Express applications.
CLK_CORRECT_USE	Boolean	Use clock correction. Set to TRUE to enable clock correction for PCI Express applications in Gen1 and Gen2.
CLK_CORRECT_KEEP_IDLE	Boolean	Keep clock correction idle. Set to TRUE to keep at least one clock correction sequence for every continuous stream of clock correction sequences received for PCI Express applications.
CLK_COR_MAX_LAT	Integer	Clock correction max latency. Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
CLK_COR_MIN_LAT	Integer	Clock correction min latency. Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.

Table 6-10: **Clock Correction Attributes for PCI Express *(Cont'd)***

Attribute	Type	Description
CLK_COR_PRECEDENCE	Boolean	Clock correction precedence. Set to TRUE to set higher priority on clock correction over channel bonding for PCI Express applications.
CLK_COR_REPEAT_WAIT	Integer	Wait cycles to repeat clock correction. Set to 0 for continuous clock correction for PCI Express applications.
CLK_COR_SEQ_LEN	Integer	Set to 1 for PCI Express applications.
CLK_COR_SEQ_1_ENABLE	4-bit Binary	Set to `1111b` to use skip (SKP) ordered set for PCI Express applications.
CLK_COR_SEQ_1_1	10-bit Binary	Set to `0100011100b` for PCI Express applications.
CLK_COR_SEQ_1_2	10-bit Binary	Set to `0000000000b` for PCI Express applications.
CLK_COR_SEQ_1_3	10-bit Binary	Set to `0000000000b` for PCI Express applications.
CLK_COR_SEQ_1_4	10-bit Binary	Set to `0000000000b` for PCI Express applications.
CLK_COR_SEQ_2_ENABLE	4-bit Binary	Set to `0000b` for PCI Express applications.
CLK_COR_SEQ_2_USE	Boolean	Set to FALSE to disable clock correction sequence 2 for PCI Express applications.
CLK_COR_SEQ_2_1	10-bit Binary	Set to `0000000000b` for PCI Express applications.
CLK_COR_SEQ_2_2	10-bit Binary	Set to `0000000000b` for PCI Express applications.
CLK_COR_SEQ_2_3	10-bit Binary	Set to `0000000000b` for PCI Express applications.
CLK_COR_SEQ_2_4	10-bit Binary	Set to `0000000000b` for PCI Express applications.

XAUI Use Model

This section provides the recommended guidelines to configure and use the GTX/GTH transceiver for XAUI applications.

Functional Description

In a XAUI application, four GTX/GTH transceivers, each operating at a line rate of 3.125 Gb/s, are channel bonded together. A wrapper containing properly configured GTX/GTH transceivers for XAUI applications can be generated from the 7 Series FPGAs Transceivers Wizard.

Recommended GTX/GTH transceiver features used in a XAUI application are:

- QPLL
- 2-byte internal and 2-byte FPGA interface width
- TX buffer bypass
- RX buffer
- Comma alignment
- Channel bonding
- Clock correction
- 8B/10B encoder and decoder

XAUI Use Mode

Table 6-11 shows the recommended GTX/GTH transceiver settings for several key attributes and ports.

Table 6-11: **Recommended GTX/GTH Transceiver Settings for XAUI Applications**

GTX/GTH Transceiver Attribute	Value
[TX/RX]RATE[2:0]	`3'b000` (Must set [TX/RX]OUT_DIV to 2)
[TX/RX]USRCLK	156.25 MHz
[TX/RX]USRCLK2	156.25 MHz
[TX/RX]_DATA_WIDTH	20
[TX/RX]_INT_DATAWIDTH	0
TXBUF_EN	FALSE
RXBUF_EN	TRUE
TX_XCLK_SEL	TXUSR
RX_XCLK_SEL	RXREC
TXOUTCLKSEL[2:0]	`3'b011`
CLK_CORRECT_USE	True
RXCOMMADETEN	1
RXCHBONDEN	1
[TX/RX]8B10BEN	1

XAUI Clocking

Reference Clock

For XAUI operation, a single 156.25 MHz reference clock is utilized. The reference clock feeds into an IBUFDS_GTE2 that drives the GTXE2_COMMON's or GTHE2_COMMON's GTREFCLK0 port as shown in Figure 6-10. The GTX/GTH transceiver uses the reference clock to generate the high speed serial clocks for transmitting and receiving. Table 6-12 shows the recommended PLL and clock divider settings for XAUI operation.

Table 6-12: **Recommended QPLL and Clock Settings for XAUI Applications**

GTX/GTH Transceiver Attribute	Value
QPLL_REFCLK_DIV	1
QPLL_FBDIV	`10'b0010000000`
QPLL_FBDIV_RATIO	1
[TX/RX]OUT_DIV	2
[TX/RX]_CLK25_DIV	7

Parallel Clock

The GTX/GTH transceiver internal data width and the FPGA interface are both two bytes in XAUI operation. Specifically, the internal data width is 20 bits and the FPGA interface width is 16 bits. Because the internal and FPGA interface width are both two bytes, the [TX/RX]USRCLK and [TX/RX]USRCLK2 frequencies are equivalent. In XAUI operation, the [TX/RX]USRCLK and [TX/RX]USRCLK2 frequencies are all 156.25 MHz.

For XAUI operation, the TX buffer is bypassed. Because the TX buffer is bypassed, TXOUTCLKSEL must be set to `3'b011` so that the GTX/GTH transceiver reference clock is used as the source of TXUSRCLK and TXUSRCLK2 through TXOUTCLK. Figure 6-10 shows a XAUI clocking architecture example.

Figure 6-10: **XAUI Clocking Architecture Example**

XAUI Channel Bonding

XAUI uses four channel bonded GTX/GTH transceivers. There are multiple ways to properly connect the channel bonding ports when channel bonding four GTX/GTH transceivers. Figure 6-11 shows one such channel bonding example.

Figure 6-11: **XAUI Channel Bonding Example**

Table 6-13 shows the recommended channel bonding attribute settings for XAUI. See RX Channel Bonding, page 263 for additional details.

Table 6-13: **Channel Bonding Attributes for XAUI**

Attribute	Type	Description
CHAN_BOND_KEEP_ALIGN	Boolean	Channel bonding keep align. Set to FALSE for XAUI applications.
CHAN_BOND_MAX_SKEW	Integer	Channel bonding maximum skew. Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
CHAN_BOND_SEQ_LEN	Integer	Channel bonding sequence length. Set to 1 for XAUI applications.
CHAN_BOND_SEQ_1_ENABLE	4-bit Binary	Channel bonding sequence 1 enable. For CHAN_BOND_SEQ_1_[1/2/3/4]. Set to `4'b1111` for XAUI applications.
CHAN_BOND_SEQ_1_1	10-bit Binary	Set to `10'b0101111100` for XAUI applications.
CHAN_BOND_SEQ_1_2	10-bit Binary	Set to `10'b0000000000` for XAUI applications.
CHAN_BOND_SEQ_1_3	10-bit Binary	Set to `10'b0000000000` for XAUI applications.
CHAN_BOND_SEQ_1_4	10-bit Binary	Set to `10'b0000000000` for XAUI applications.

Table 6-13: **Channel Bonding Attributes for XAUI *(Cont'd)***

Attribute	Type	Description
CHAN_BOND_SEQ_2_USE	Boolean	Use channel bonding sequence 2. Set to FALSE for XAUI applications.
CHAN_BOND_SEQ_2_ENABLE	4-bit Binary	Channel bonding sequence 2 enable. For CHAN_BOND_SEQ_2_[1/2/3/4]. Set to `4'b1111` for XAUI applications.
CHAN_BOND_SEQ_2_1	10-bit Binary	Set to `10'b0000000000` for XAUI applications.
CHAN_BOND_SEQ_2_2	10-bit Binary	Set to `10'b0000000000` for XAUI applications.
CHAN_BOND_SEQ_2_3	10-bit Binary	Set to `10'b0000000000` for XAUI applications.
CHAN_BOND_SEQ_2_4	10-bit Binary	Set to `10'b0000000000` for XAUI applications.
FTS_LANE_DESKEW_EN	Boolean	FTS lane deskew enable. Set to FALSE for XAUI applications.

XAUI Clock Correction

Table 6-14 shows the recommended clock correction attribute settings for XAUI. Refer to RX Clock Correction, page 253 for additional details.

Table 6-14: **Clock Correction Attributes for XAUI**

Attribute	Type	Description
CBCC_DATA_SOURCE_SEL	String	Channel bonding and clock correction data source select. Set to DECODED to use data from the output of the 8B/10B decoder for XAUI applications.
CLK_CORRECT_USE	Boolean	Use clock correction. Set to TRUE to enable clock correction for XAUI applications.
CLK_CORRECT_KEEP_IDLE	Boolean	Keep clock correction idle. Set to FALSE for XAUI applications.
CLK_COR_MAX_LAT	Integer	Clock correction maximum latency. Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
CLK_COR_MIN_LAT	Integer	Clock correction minimum latency. Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
CLK_COR_PRECEDENCE	Boolean	Clock correction precedence. Set to TRUE to set higher priority on clock correction over channel bonding for XAUI applications.

Table 6-14: **Clock Correction Attributes for XAUI *(Cont'd)***

Attribute	Type	Description
CLK_COR_REPEAT_WAIT	Integer	Wait cycles to repeat clock correction. Set to 0 for continuous clock correction for XAUI applications.
CLK_COR_SEQ_LEN	Integer	Set to 1 for XAUI applications.
CLK_COR_SEQ_1_ENABLE	4-bit Binary	Set to `4'b1111` for XAUI applications.
CLK_COR_SEQ_1_1	10-bit Binary	Set to `10'b0100011100` for XAUI applications.
CLK_COR_SEQ_1_2	10-bit Binary	Set to `10'b0100000000` for XAUI applications.
CLK_COR_SEQ_1_1	10-bit Binary	Set to `10'b0100000000` for XAUI applications.
CLK_COR_SEQ_1_4	10-bit Binary	Set to `10'b0100000000` for XAUI applications.
CLK_COR_SEQ_2_ENABLE	4-bit Binary	Set to `4'b1111` for XAUI applications.
CLK_COR_SEQ_2_USE	Boolean	Set to FALSE to disable clock correction sequence 2 for XAUI applications.
CLK_COR_SEQ_2_1	10-bit Binary	Set to `10'b0100000000` for XAUI applications.
CLK_COR_SEQ_2_2	10-bit Binary	Set to `10'b0100000000` for XAUI applications.
CLK_COR_SEQ_2_3	10-bit Binary	Set to `10'b0100000000` for XAUI applications.
CLK_COR_SEQ_2_4	10-bit Binary	Set to `10'b0100000000` for XAUI applications.

Appendix A

Placement Information by Package

This appendix provides the Quad position information for available device and package combinations along with the pad numbers for the external signals associated with each serial transceiver channel and the associated primitive.

Some devices are available in both Pb and Pb-free (additional G) packages as standard ordering options.

GTX Transceiver Package Placement Diagrams

- FBG484 Package Placement Diagram, page 338
- FBG676 Package Placement Diagram, page 339
- FBG900 Package Placement Diagram, page 340
- FFG676 Package Placement Diagram, page 342
- FFG900 Package Placement Diagram, page 343
- FFG901 Package Placement Diagram, page 345
- FFG1156 Package Placement Diagram, page 349
- FFG1157 Package Placement Diagram, page 353
- FFG1158 Package Placement Diagram, page 356
- FFG1761 Package Placement Diagram, page 362
- FFG1927 Package Placement Diagram, page 367
- FFG1930 Package Placement Diagram, page 375
- FLG1925 Package Placement Diagram, page 378
- FHG1761 Package Placement Diagram, page 380

FBG484 Package Placement Diagram

Figure A-1 shows the placement diagram for the FBG484 package.

Figure A-1: **Placement Diagram for the FBG484 Package**

FBG676 Package Placement Diagram

Figure A-2 shows the placement diagram for the FBG676 package.

Figure A-2: **Placement Diagram for the FBG676 Package**

FBG900 Package Placement Diagram

Figure A-3 and Figure A-4 show the placement diagram for the FBG900 package.

Figure A-3: **Placement Diagram for the FBG900 Package (1 of 2)**

Figure A-4: **Placement Diagram for the FBG900 Package (2 of 2)**

FFG676 Package Placement Diagram

Figure A-5 shows the placement diagram for the FFG676 package.

Figure A-5: **Placement Diagram for the FFG676 Package**

FFG900 Package Placement Diagram

Figure A-6 and Figure A-7 show the placement diagram for the FFG900 package.

Figure A-6: **Placement Diagram for the FFG900 Package (1 of 2)**

Figure A-7: **Placement Diagram for the FFG900 Package (2 of 2)**

FFG901 Package Placement Diagram

Figure A-8 through Figure A-11 show the placement diagram for the FFG901 package.

Figure A-8: **Placement Diagram for the FFG901 Package (1 of 4)**

Figure A-9: **Placement Diagram for the FFG901 Package (2 of 4)**

Figure A-10: **Placement Diagram for the FFG901 Package (3 of 4)**

Figure A-11: **Placement Diagram for the FFG901 Package (4 of 4)**

FFG1156 Package Placement Diagram

Figure A-12 through Figure A-15 show the placement diagram for the FFG1156 package.

Figure A-12: **Placement Diagram for the FFG1156 Package (1 of 4)**

Figure A-13: **Placement Diagram for the FFG1156 Package (2 of 4)**

Figure A-14: **Placement Diagram for the FFG1156 Package (3 of 4)**

Figure A-15: **Placement Diagram for the FFG1156 Package (4 of 4)**

FFG1157 Package Placement Diagram

Figure A-16 through Figure A-18 show the placement diagram for the FFG1157 package.

Figure A-16: **Placement Diagram for the FFG1157 Package (1 of 3)**

Figure A-17: **Placement Diagram for the FFG1157 Package (2 of 3)**

Figure A-18: **Placement Diagram for the FFG1157 Package (3 of 3)**

FFG1158 Package Placement Diagram

Figure A-19 through Figure A-24 show the placement diagram for the FFG1158 package.

Figure A-19: **Placement Diagram for the FFG1158 Package (1 of 6)**

Figure A-20: **Placement Diagram for the FFG1158 Package (2 of 6)**

Figure A-21: **Placement Diagram for the FFG1158 Package (3 of 6)**

Figure A-22: **Placement Diagram for the FFG1158 Package (4 of 6)**

Figure A-23: **Placement Diagram for the FFG1158 Package (5 of 6)**

Figure A-24: **Placement Diagram for the FFG1158 Package (6 of 6)**

FFG1761 Package Placement Diagram

Figure A-25 through Figure A-29 show the placement diagram for the FFG1761 package.

Figure A-25: **Placement Diagram for the FFG1761 Package (1 of 5)**

Figure A-26: **Placement Diagram for the FFG1761 Package (2 of 5)**

Figure A-27: **Placement Diagram for the FFG1761 Package (3 of 5)**

Figure A-28: **Placement Diagram for the FFG1761 Package (4 of 5)**

Figure A-29: **Placement Diagram for the FFG1761 Package (5 of 5)**

FFG1927 Package Placement Diagram

Figure A-30 through Figure A-37 show the placement diagram for the FFG1927 package.

Figure A-30: **Placement Diagram for the FFG1927 Package (1 of 8)**

Figure A-31: **Placement Diagram for the FFG1927 Package (2 of 8)**

Figure A-32: **Placement Diagram for the FFG1927 Package (3 of 8)**

Figure A-33: **Placement Diagram for the FFG1927 Package (4 of 8)**

Figure A-34: **Placement Diagram for the FFG1927 Package (5 of 8)**

Figure A-35: **Placement Diagram for the FFG1927 Package (6 of 8)**

Figure A-36: **Placement Diagram for the FFG1927 Package (7 of 8)**

Figure A-37: **Placement Diagram for the FFG1927 Package (8 of 8)**

FFG1930 Package Placement Diagram

Figure A-38 through Figure A-40 show the placement diagram for the FFG1930 package.

Figure A-38: **Placement Diagram for the FFG1930 Package (1 of 3)**

Figure A-39: **Placement Diagram for the FFG1930 Package (2 of 3)**

Figure A-40: **Placement Diagram for the FFG1930 Package (3 of 3)**

FLG1925 Package Placement Diagram

Figure A-41 through Figure A-42 show the placement diagram for the FLG1925 package.

Figure A-41: **Placement Diagram for the FLG1925 Package (1 of 2)**

Figure A-42: **Placement Diagram for the FLG1925 Package (2 of 2)**

FHG1761 Package Placement Diagram

Figure A-43 through Figure A-47 show the placement diagram for the FHG1761 package.

Figure A-43: **Placement Diagram for the FHG1761 Package (1 of 5)**

Figure A-44: **Placement Diagram for the FHG1761 Package (2 of 5)**

Figure A-45: **Placement Diagram for the FHG1761 Package (3 of 5)**

Figure A-46: **Placement Diagram for the FHG1761 Package (4 of 5)**

Figure A-47: **Placement Diagram for the FHG1761 Package (5 of 5)**

GTH Transceiver Package Placement Diagrams

- FFG1157 Package Placement Diagram, page 386
- FFG1158 Package Placement Diagram, page 389
- FFG1761 Package Placement Diagram, page 395
- FFG1926 Package Placement Diagram, page 400
- FFG1927 Package Placement Diagram, page 408
- FFG1928 Package Placement Diagram, page 418
- FFG1930 Package Placement Diagram, page 428
- FLG1926 Package Placement Diagram, page 431
- FLG1928 Package Placement Diagram, page 439
- FLG1930 Package Placement Diagram, page 451

FFG1157 Package Placement Diagram

Figure A-48 through Figure A-50 show the placement diagram for the FFG1157 package.

Figure A-48: **Placement Diagram for the FFG1157 Package (1 of 3)**

Figure A-49: **Placement Diagram for the FFG1157 Package (2 of 3)**

Figure A-50: **Placement Diagram for the FFG1157 Package (3 of 3)**

FFG1158 Package Placement Diagram

Figure A-51 through Figure A-56 show the placement diagram for the FFG1158 package.

Figure A-51: **Placement Diagram for the FFG1158 Package (1 of 6)**

Figure A-52: **Placement Diagram for the FFG1158 Package (2 of 6)**

Figure A-53: **Placement Diagram for the FFG1158 Package (3 of 6)**

Figure A-54: **Placement Diagram for the FFG1158 Package (4 of 6)**

Figure A-55: **Placement Diagram for the FFG1158 Package (5 of 6)**

Figure A-56: **Placement Diagram for the FFG1158 Package (6 of 6)**

FFG1761 Package Placement Diagram

Figure A-57 through Figure A-61 show the placement diagram for the FFG1761 package.

Figure A-57: **Placement Diagram for the FFG1761 Package (1 of 5)**

Figure A-58: **Placement Diagram for the FFG1761 Package (2 of 5)**

Figure A-59: **Placement Diagram for the FFG1761 Package (3 of 5)**

Figure A-60: **Placement Diagram for the FFG1761 Package (4 of 5)**

Figure A-61: **Placement Diagram for the FFG1761 Package (5 of 5)**

FFG1926 Package Placement Diagram

Figure A-62 through Figure A-69 show the placement diagram for the FFG1926 package.

Figure A-62: **Placement Diagram for the FFG1926 Package (1 of 8)**

Figure A-63: **Placement Diagram for the FFG1926 Package (2 of 8)**

Figure A-64: **Placement Diagram for the FFG1926 Package (3 of 8)**

Figure A-65: **Placement Diagram for the FFG1926 Package (4 of 8)**

Figure A-66: **Placement Diagram for the FFG1926 Package (5 of 8)**

Figure A-67: **Placement Diagram for the FFG1926 Package (6 of 8)**

Figure A-68: **Placement Diagram for the FFG1926 Package (7 of 8)**

Figure A-69: **Placement Diagram for the FFG1926 Package (8 of 8)**

FFG1927 Package Placement Diagram

Figure A-70 through Figure A-79 show the placement diagram for the FFG1927 package.

Figure A-70: **Placement Diagram for the FFG1927 Package (1 of 10)**

Figure A-71: **Placement Diagram for the FFG1927 Package (2 of 10)**

Figure A-72: **Placement Diagram for the FFG1927 Package (3 of 10)**

Figure A-73: **Placement Diagram for the FFG1927 Package (4 of 10)**

Figure A-74: **Placement Diagram for the FFG1927 Package (5 of 10)**

Figure A-75: **Placement Diagram for the FFG1927 Package (6 of 10)**

Figure A-76: **Placement Diagram for the FFG1927 Package (7 of 10)**

Figure A-77: **Placement Diagram for the FFG1927 Package (8 of 10)**

Figure A-78: **Placement Diagram for the FFG1927 Package (9 of 10)**

Figure A-79: **Placement Diagram for the FFG1927 Package (10 of 10)**

FFG1928 Package Placement Diagram

Figure A-90 through Figure A-89 show the placement diagram for the FFG1928 package.

Figure A-80: **Placement Diagram for the FFG1928 Package (1 of 10)**

Figure A-81: **Placement Diagram for the FFG1928 Package (2 of 10)**

Figure A-82: **Placement Diagram for the FFG1928 Package (3 of 10)**

Figure A-83: **Placement Diagram for the FFG1928 Package (4 of 10)**

Figure A-84: **Placement Diagram for the FFG1928 Package (5 of 10)**

Figure A-85: **Placement Diagram for the FFG1928 Package (6 of 10)**

Figure A-86: **Placement Diagram for the FFG1928 Package (7 of 10)**

Figure A-87: **Placement Diagram for the FFG1928 Package (8 of 10)**

Figure A-88: **Placement Diagram for the FFG1928 Package (9 of 10)**

Figure A-89: **Placement Diagram for the FFG1928 Package (10 of 10)**

FFG1930 Package Placement Diagram

Figure A-90 through Figure A-92 show the placement diagram for the FFG1930 package.

Figure A-90: **Placement Diagram for the FFG1930 Package (1 of 3)**

Figure A-91: **Placement Diagram for the FFG1930 Package (2 of 3)**

Figure A-92: **Placement Diagram for the FFG1930 Package (3 of 3)**

FLG1926 Package Placement Diagram

Figure A-93 through Figure A-100 show the placement diagram for the FLG1926 package.

Figure A-93: **Placement Diagram for the FLG1926 Package (1 of 8)**

Figure A-94: **Placement Diagram for the FLG1926 Package (2 of 8)**

Figure A-95: **Placement Diagram for the FLG1926 Package (3 of 8)**

Figure A-96: **Placement Diagram for the FLG1926 Package (4 of 8)**

Figure A-97: **Placement Diagram for the FLG1926 Package (5 of 8)**

Figure A-98: **Placement Diagram for the FLG1926 Package (6 of 8)**

Figure A-99: **Placement Diagram for the FLG1926 Package (7 of 8)**

Figure A-100: **Placement Diagram for the FLG1926 Package (8 of 8)**

FLG1928 Package Placement Diagram

Figure A-101 through Figure A-112 show the placement diagram for the FLG1928 package.

Figure A-101: **Placement Diagram for the FLG1928 Package (1 of 12)**

Figure A-102: **Placement Diagram for the FLG1928 Package (2 of 12)**

Figure A-103: **Placement Diagram for the FLG1928 Package (3 of 12)**

Figure A-104: **Placement Diagram for the FLG1928 Package (4 of 12)**

Figure A-105: **Placement Diagram for the FLG1928 Package (5 of 12)**

Figure A-106: **Placement Diagram for the FLG1928 Package (6 of 12)**

Figure A-107: **Placement Diagram for the FLG1928 Package (7 of 12)**

Figure A-108: **Placement Diagram for the FLG1928 Package (8 of 12)**

Figure A-109: **Placement Diagram for the FLG1928 Package (9 of 12)**

Figure A-110: **Placement Diagram for the FLG1928 Package (10 of 12)**

Figure A-111: **Placement Diagram for the FLG1928 Package (11 of 12)**

Figure A-112: **Placement Diagram for the FLG1928 Package (12 of 12)**

FLG1930 Package Placement Diagram

Figure A-113 through Figure A-115 show the placement diagram for the FLG1930 package.

Figure A-113: **Placement Diagram for the FLG1930 Package (1 of 3)**

Figure A-114: **Placement Diagram for the FLG1930 Package (2 of 3)**

Figure A-115: **Placement Diagram for the FLG1930 Package (3 of 3)**

Appendix B

Placement Information by Device

Table B-1 defines the Kintex™-7 FPGA device-package combinations and the available GTX transceiver banks. For transceiver location, refer to Appendix A, Placement Information by Package.

Table B-1: **Kintex-7 FPGA Device-Package Combinations and GTX Transceiver Banks**

Package	FBG484	FBG676	FBG900	FFG676	FFG900	FFG901	FFG1156
XC7K70T	MGT_BANK_115	MGT_BANK_115, MGT_BANK_116					
XC7K160T	MGT_BANK_115	MGT_BANK_115, MGT_BANK_116		MGT_BANK_115, MGT_BANK_116			
XC7K325T		MGT_BANK_115, MGT_BANK_116	MGT_BANK_115, MGT_BANK_116, MGT_BANK_117, MGT_BANK_118	MGT_BANK_115, MGT_BANK_116	MGT_BANK_115, MGT_BANK_116, MGT_BANK_117, MGT_BANK_118		
XC7K355T						MGT_BANK_112, MGT_BANK_113, MGT_BANK_114, MGT_BANK_115, MGT_BANK_116, MGT_BANK_117	
XC7K410T		MGT_BANK_115, MGT_BANK_116	MGT_BANK_115, MGT_BANK_116, MGT_BANK_117, MGT_BANK_118	MGT_BANK_115, MGT_BANK_116	MGT_BANK_115, MGT_BANK_116, MGT_BANK_117, MGT_BANK_118		
XC7K420T						MGT_BANK_111, MGT_BANK_112, MGT_BANK_113, MGT_BANK_114, MGT_BANK_115, MGT_BANK_116, MGT_BANK_117	MGT_BANK_111, MGT_BANK_112, MGT_BANK_113, MGT_BANK_114, MGT_BANK_115, MGT_BANK_116, MGT_BANK_117, MGT_BANK_118
XC7K480T						MGT_BANK_111, MGT_BANK_112, MGT_BANK_113, MGT_BANK_114, MGT_BANK_115, MGT_BANK_116, MGT_BANK_117	MGT_BANK_111, MGT_BANK_112, MGT_BANK_113, MGT_BANK_114, MGT_BANK_115, MGT_BANK_116, MGT_BANK_117, MGT_BANK_118

Table B-2 defines the Virtex®-7 FPGA device-package combinations and the available GTX transceiver banks. For transceiver location, refer to Appendix A, Placement Information by Package.

Table B-2: **Virtex-7 FPGA Device-Package Combinations and GTX Transceiver Banks**

Package	FFG1157	FFG1158	FFG1761	FFG1927	FFG1930	FLG1925	FHG1761
XC7V585T	MGT_BANK_114, MGT_BANK_115, MGT_BANK_116, MGT_BANK_117, MGT_BANK_118		MGT_BANK_111, MGT_BANK_112, MGT_BANK_113, MGT_BANK_114, MGT_BANK_115, MGT_BANK_116, MGT_BANK_117, MGT_BANK_118, MGT_BANK_119				
XC7V2000T						MGT_BANK_112, MGT_BANK_113, MGT_BANK_114, MGT_BANK_115	MGT_BANK_111, MGT_BANK_112, MGT_BANK_113, MGT_BANK_114, MGT_BANK_115, MGT_BANK_116, MGT_BANK_117, MGT_BANK_118, MGT_BANK_119
XC7VX485T	MGT_BANK_114, MGT_BANK_115, MGT_BANK_116, MGT_BANK_117, MGT_BANK_118	MGT_BANK_114, MGT_BANK_115, MGT_BANK_116, MGT_BANK_117, MGT_BANK_118, MGT_BANK_119, MGT_BANK_214, MGT_BANK_215, MGT_BANK_216, MGT_BANK_217, MGT_BANK_218, MGT_BANK_219	MGT_BANK_113, MGT_BANK_114, MGT_BANK_115, MGT_BANK_116, MGT_BANK_117, MGT_BANK_118, MGT_BANK_119	MGT_BANK_113, MGT_BANK_114, MGT_BANK_115, MGT_BANK_116, MGT_BANK_117, MGT_BANK_118, MGT_BANK_119, MGT_BANK_213, MGT_BANK_214, MGT_BANK_215, MGT_BANK_216, MGT_BANK_217, MGT_BANK_218, MGT_BANK_219	MGT_BANK_113, MGT_BANK_114, MGT_BANK_115, MGT_BANK_116, MGT_BANK_117, MGT_BANK_118		

Table B-3 defines the Virtex-7 FPGA device-package combinations and the available GTH transceiver banks. For transceiver location, refer to Appendix A, Placement Information by Package.

Table B-3: **Virtex-7 FPGA Device-Package Combinations and GTH Transceiver Banks**

Package	FFG1157	FFG1158	FFG1761	FFG1926	FFG1927	FFG1928	FFG1930	FLG1926	FLG1928	FLG1930
XC7VX330T	MGT_BANK_114, MGT_BANK_115, MGT_BANK_116, MGT_BANK_117, MGT_BANK_118		MGT_BANK_113, MGT_BANK_114, MGT_BANK_115, MGT_BANK_116, MGT_BANK_117, MGT_BANK_118, MGT_BANK_119							
XC7VX415T	MGT_BANK_114, MGT_BANK_115, MGT_BANK_116, MGT_BANK_117, MGT_BANK_118	MGT_BANK_114, MGT_BANK_115, MGT_BANK_116, MGT_BANK_117, MGT_BANK_118, MGT_BANK_119, MGT_BANK_214, MGT_BANK_215, MGT_BANK_216, MGT_BANK_217, MGT_BANK_218, MGT_BANK_219			MGT_BANK_114, MGT_BANK_115, MGT_BANK_116, MGT_BANK_117, MGT_BANK_118, MGT_BANK_119, MGT_BANK_214, MGT_BANK_215, MGT_BANK_216, MGT_BANK_217, MGT_BANK_218, MGT_BANK_219					
XC7VX550T		MGT_BANK_114, MGT_BANK_115, MGT_BANK_116, MGT_BANK_117, MGT_BANK_118, MGT_BANK_119, MGT_BANK_214, MGT_BANK_215, MGT_BANK_216, MGT_BANK_217, MGT_BANK_218, MGT_BANK_219			MGT_BANK_110, MGT_BANK_111, MGT_BANK_112, MGT_BANK_113, MGT_BANK_114, MGT_BANK_115, MGT_BANK_116, MGT_BANK_117, MGT_BANK_118, MGT_BANK_119, MGT_BANK_210, MGT_BANK_211, MGT_BANK_212, MGT_BANK_213, MGT_BANK_214, MGT_BANK_215, MGT_BANK_216, MGT_BANK_217, MGT_BANK_218, MGT_BANK_219					

Table B-3: **Virtex-7 FPGA Device-Package Combinations and GTH Transceiver Banks *(Cont'd)***

Package	FFG1157	FFG1158	FFG1761	FFG1926	FFG1927	FFG1928	FFG1930	FLG1926	FLG1928	FLG1930
XC7VX690T	MGT_BANK_114, MGT_BANK_115, MGT_BANK_116, MGT_BANK_117, MGT_BANK_118	MGT_BANK_114, MGT_BANK_115, MGT_BANK_116, MGT_BANK_117, MGT_BANK_118, MGT_BANK_119, MGT_BANK_214, MGT_BANK_215, MGT_BANK_216, MGT_BANK_217, MGT_BANK_218, MGT_BANK_219	MGT_BANK_111, MGT_BANK_112, MGT_BANK_113, MGT_BANK_114, MGT_BANK_115, MGT_BANK_116, MGT_BANK_117, MGT_BANK_118, MGT_BANK_119	MGT_BANK_111, MGT_BANK_112, MGT_BANK_113, MGT_BANK_114, MGT_BANK_115, MGT_BANK_116, MGT_BANK_117, MGT_BANK_118, MGT_BANK_211, MGT_BANK_212, MGT_BANK_213, MGT_BANK_214, MGT_BANK_215, MGT_BANK_216, MGT_BANK_217, MGT_BANK_218	MGT_BANK_110, MGT_BANK_111, MGT_BANK_112, MGT_BANK_113, MGT_BANK_114, MGT_BANK_115, MGT_BANK_116, MGT_BANK_117, MGT_BANK_118, MGT_BANK_119, MGT_BANK_210, MGT_BANK_211, MGT_BANK_212, MGT_BANK_213, MGT_BANK_214, MGT_BANK_215, MGT_BANK_216, MGT_BANK_217, MGT_BANK_218 MGT_BANK_219		MGT_BANK_113, MGT_BANK_114, MGT_BANK_115, MGT_BANK_116, MGT_BANK_117, MGT_BANK_118			
XC7VX980T				MGT_BANK_111, MGT_BANK_112, MGT_BANK_113, MGT_BANK_114, MGT_BANK_115, MGT_BANK_116, MGT_BANK_117, MGT_BANK_118, MGT_BANK_211, MGT_BANK_212, MGT_BANK_213, MGT_BANK_214, MGT_BANK_215, MGT_BANK_216, MGT_BANK_217, MGT_BANK_218		MGT_BANK_110, MGT_BANK_111, MGT_BANK_112, MGT_BANK_113, MGT_BANK_114, MGT_BANK_115, MGT_BANK_116, MGT_BANK_117, MGT_BANK_118, MGT_BANK_210, MGT_BANK_211, MGT_BANK_212, MGT_BANK_213, MGT_BANK_214, MGT_BANK_215, MGT_BANK_216, MGT_BANK_217, MGT_BANK_218	MGT_BANK_113, MGT_BANK_114, MGT_BANK_115, MGT_BANK_116, MGT_BANK_117, MGT_BANK_118			
XC7VX1140T								MGT_BANK_111, MGT_BANK_112, MGT_BANK_113, MGT_BANK_114, MGT_BANK_115, MGT_BANK_116, MGT_BANK_117, MGT_BANK_118, MGT_BANK_211, MGT_BANK_212, MGT_BANK_213, MGT_BANK_214, MGT_BANK_215, MGT_BANK_216, MGT_BANK_217, MGT_BANK_218	MGT_BANK_110, MGT_BANK_111, MGT_BANK_112, MGT_BANK_113, MGT_BANK_114, MGT_BANK_115, MGT_BANK_116, MGT_BANK_117, MGT_BANK_118, MGT_BANK_119, MGT_BANK_120, MGT_BANK_121, MGT_BANK_210, MGT_BANK_211, MGT_BANK_212, MGT_BANK_213, MGT_BANK_214, MGT_BANK_215, MGT_BANK_216, MGT_BANK_217, MGT_BANK_218, MGT_BANK_219, MGT_BANK_220, MGT_BANK_221	MGT_BANK_113, MGT_BANK_114, MGT_BANK_115, MGT_BANK_116, MGT_BANK_117, MGT_BANK_118

XILINX

Appendix C

8B/10B Valid Characters

8B/10B encoding includes a set of Data characters and K characters. Eight-bit values are coded into 10-bit values, keeping the serial line DC balanced. K characters are special Data characters designated with a CHARISK. K characters are used for specific informative designations. Table C-1 shows the valid Data characters. Table C-2, page 467 shows the valid K characters.

Table C-1: **Valid Data Characters**

Data Byte Name	Bits HGF EDCBA	Current RD – abcdei fghj	Current RD + abcdei fghj
D0.0	000 00000	100111 0100	011000 1011
D1.0	000 00001	011101 0100	100010 1011
D2.0	000 00010	101101 0100	010010 1011
D3.0	000 00011	110001 1011	110001 0100
D4.0	000 00100	110101 0100	001010 1011
D5.0	000 00101	101001 1011	101001 0100
D6.0	000 00110	011001 1011	011001 0100
D7.0	000 00111	111000 1011	000111 0100
D8.0	000 01000	111001 0100	000110 1011
D9.0	000 01001	100101 1011	100101 0100
D10.0	000 01010	010101 1011	010101 0100
D11.0	000 01011	110100 1011	110100 0100
D12.0	000 01100	001101 1011	001101 0100
D13.0	000 01101	101100 1011	101100 0100
D14.0	000 01110	011100 1011	011100 0100
D15.0	000 01111	010111 0100	101000 1011
D16.0	000 10000	011011 0100	100100 1011
D17.0	000 10001	100011 1011	100011 0100
D18.0	000 10010	010011 1011	010011 0100
D19.0	000 10011	110010 1011	110010 0100
D20.0	000 10100	001011 1011	001011 0100

Table C-1: **Valid Data Characters *(Cont'd)***

Data Byte Name	Bits HGF EDCBA	Current RD – abcdei fghj	Current RD + abcdei fghj
D21.0	000 10101	101010 1011	101010 0100
D22.0	000 10110	011010 1011	011010 0100
D23.0	000 10111	111010 0100	000101 1011
D24.0	000 11000	110011 0100	001100 1011
D25.0	000 11001	100110 1011	100110 0100
D26.0	000 11010	010110 1011	010110 0100
D27.0	000 11011	110110 0100	001001 1011
D28.0	000 11100	001110 1011	001110 0100
D29.0	000 11101	101110 0100	010001 1011
D30.0	000 11110	011110 0100	100001 1011
D31.0	000 11111	101011 0100	010100 1011
D0.1	001 00000	100111 1001	011000 1001
D1.1	001 00001	011101 1001	100010 1001
D2.1	001 00010	101101 1001	010010 1001
D3.1	001 00011	110001 1001	110001 1001
D4.1	001 00100	110101 1001	001010 1001
D5.1	001 00101	101001 1001	101001 1001
D6.1	001 00110	011001 1001	011001 1001
D7.1	001 00111	111000 1001	000111 1001
D8.1	001 01000	111001 1001	000110 1001
D9.1	001 01001	100101 1001	100101 1001
D10.1	001 01010	010101 1001	010101 1001
D11.1	001 01011	110100 1001	110100 1001
D12.1	001 01100	001101 1001	001101 1001
D13.1	001 01101	101100 1001	101100 1001
D14.1	001 01110	011100 1001	011100 1001
D15.1	001 01111	010111 1001	101000 1001
D16.1	001 10000	011011 1001	100100 1001
D17.1	001 10001	100011 1001	100011 1001
D18.1	001 10010	010011 1001	010011 1001
D19.1	001 10011	110010 1001	110010 1001
D20.1	001 10100	001011 1001	001011 1001

Table C-1: **Valid Data Characters *(Cont'd)***

Data Byte Name	Bits HGF EDCBA	Current RD – abcdei fghj	Current RD + abcdei fghj
D21.1	001 10101	101010 1001	101010 1001
D22.1	001 10110	011010 1001	011010 1001
D23.1	001 10111	111010 1001	000101 1001
D24.1	001 11000	110011 1001	001100 1001
D25.1	001 11001	100110 1001	100110 1001
D26.1	001 11010	010110 1001	010110 1001
D27.1	001 11011	110110 1001	001001 1001
D28.1	001 11100	001110 1001	001110 1001
D29.1	001 11101	101110 1001	010001 1001
D30.1	001 11110	011110 1001	100001 1001
D31.1	001 11111	101011 1001	010100 1001
D0.2	010 00000	100111 0101	011000 0101
D1.2	010 00001	011101 0101	100010 0101
D2.2	010 00010	101101 0101	010010 0101
D3.2	010 00011	110001 0101	110001 0101
D4.2	010 00100	110101 0101	001010 0101
D5.2	010 00101	101001 0101	101001 0101
D6.2	010 00110	011001 0101	011001 0101
D7.2	010 00111	111000 0101	000111 0101
D8.2	010 01000	111001 0101	000110 0101
D9.2	010 01001	100101 0101	100101 0101
D10.2	010 01010	010101 0101	010101 0101
D11.2	010 01011	110100 0101	110100 0101
D12.2	010 01100	001101 0101	001101 0101
D13.2	010 01101	101100 0101	101100 0101
D14.2	010 01110	011100 0101	011100 0101
D15.2	010 01111	010111 0101	101000 0101
D16.2	010 10000	011011 0101	100100 0101
D17.2	010 10001	100011 0101	100011 0101
D18.2	010 10010	010011 0101	010011 0101
D19.2	010 10011	110010 0101	110010 0101
D20.2	010 10100	001011 0101	001011 0101

Table C-1: **Valid Data Characters *(Cont'd)***

Data Byte Name	Bits HGF EDCBA	Current RD – abcdei fghj	Current RD + abcdei fghj
D21.2	010 10101	101010 0101	101010 0101
D22.2	010 10110	011010 0101	011010 0101
D23.2	010 10111	111010 0101	000101 0101
D24.2	010 11000	110011 0101	001100 0101
D25.2	010 11001	100110 0101	100110 0101
D26.2	010 11010	010110 0101	010110 0101
D27.2	010 11011	110110 0101	001001 0101
D28.2	010 11100	001110 0101	001110 0101
D29.2	010 11101	101110 0101	010001 0101
D30.2	010 11110	011110 0101	100001 0101
D31.2	010 11111	101011 0101	010100 0101
D0.3	011 00000	100111 0011	011000 1100
D1.3	011 00001	011101 0011	100010 1100
D2.3	011 00010	101101 0011	010010 1100
D3.3	011 00011	110001 1100	110001 0011
D4.3	011 00100	110101 0011	001010 1100
D5.3	011 00101	101001 1100	101001 0011
D6.3	011 00110	011001 1100	011001 0011
D7.3	011 00111	111000 1100	000111 0011
D8.3	011 01000	111001 0011	000110 1100
D9.3	011 01001	100101 1100	100101 0011
D10.3	011 01010	010101 1100	010101 0011
D11.3	011 01011	110100 1100	110100 0011
D12.3	011 01100	001101 1100	001101 0011
D13.3	011 01101	101100 1100	101100 0011
D14.3	011 01110	011100 1100	011100 0011
D15.3	011 01111	010111 0011	101000 1100
D16.3	011 10000	011011 0011	100100 1100
D17.3	011 10001	100011 1100	100011 0011
D18.3	011 10010	010011 1100	010011 0011
D19.3	011 10011	110010 1100	110010 0011
D20.3	011 10100	001011 1100	001011 0011

Table C-1: **Valid Data Characters *(Cont'd)***

Data Byte Name	Bits HGF EDCBA	Current RD – abcdei fghj	Current RD + abcdei fghj
D21.3	011 10101	101010 1100	101010 0011
D22.3	011 10110	011010 1100	011010 0011
D23.3	011 10111	111010 0011	000101 1100
D24.3	011 11000	110011 0011	001100 1100
D25.3	011 11001	100110 1100	100110 0011
D26.3	011 11010	010110 1100	010110 0011
D27.3	011 11011	110110 0011	001001 1100
D28.3	011 11100	001110 1100	001110 0011
D29.3	011 11101	101110 0011	010001 1100
D30.3	011 11110	011110 0011	100001 1100
D31.3	011 11111	101011 0011	010100 1100
D0.4	100 00000	100111 0010	011000 1101
D1.4	100 00001	011101 0010	100010 1101
D2.4	100 00010	101101 0010	010010 1101
D3.4	100 00011	110001 1101	110001 0010
D4.4	100 00100	110101 0010	001010 1101
D5.4	100 00101	101001 1101	101001 0010
D6.4	100 00110	011001 1101	011001 0010
D7.4	100 00111	111000 1101	000111 0010
D8.4	100 01000	111001 0010	000110 1101
D9.4	100 01001	100101 1101	100101 0010
D10.4	100 01010	010101 1101	010101 0010
D11.4	100 01011	110100 1101	110100 0010
D12.4	100 01100	001101 1101	001101 0010
D13.4	100 01101	101100 1101	101100 0010
D14.4	100 01110	011100 1101	011100 0010
D15.4	100 01111	010111 0010	101000 1101
D16.4	100 10000	011011 0010	100100 1101
D17.4	100 10001	100011 1101	100011 0010
D18.4	100 10010	010011 1101	010011 0010
D19.4	100 10011	110010 1101	110010 0010
D20.4	100 10100	001011 1101	001011 0010

Table C-1: **Valid Data Characters *(Cont'd)***

Data Byte Name	Bits HGF EDCBA	Current RD – abcdei fghj	Current RD + abcdei fghj
D21.4	100 10101	101010 1101	101010 0010
D22.4	100 10110	011010 1101	011010 0010
D23.4	100 10111	111010 0010	000101 1101
D24.4	100 11000	110011 0010	001100 1101
D25.4	100 11001	100110 1101	100110 0010
D26.4	100 11010	010110 1101	010110 0010
D27.4	100 11011	110110 0010	001001 1101
D28.4	100 11100	001110 1101	001110 0010
D29.4	100 11101	101110 0010	010001 1101
D30.4	100 11110	011110 0010	100001 1101
D31.4	100 11111	101011 0010	010100 1101
D0.5	101 00000	100111 1010	011000 1010
D1.5	101 00001	011101 1010	100010 1010
D2.5	101 00010	101101 1010	010010 1010
D3.5	101 00011	110001 1010	110001 1010
D4.5	101 00100	110101 1010	001010 1010
D5.5	101 00101	101001 1010	101001 1010
D6.5	101 00110	011001 1010	011001 1010
D7.5	101 00111	111000 1010	000111 1010
D8.5	101 01000	111001 1010	000110 1010
D9.5	101 01001	100101 1010	100101 1010
D10.5	101 01010	010101 1010	010101 1010
D11.5	101 01011	110100 1010	110100 1010
D12.5	101 01100	001101 1010	001101 1010
D13.5	101 01101	101100 1010	101100 1010
D14.5	101 01110	011100 1010	011100 1010
D15.5	101 01111	010111 1010	101000 1010
D16.5	101 10000	011011 1010	100100 1010
D17.5	101 10001	100011 1010	100011 1010
D18.5	101 10010	010011 1010	010011 1010
D19.5	101 10011	110010 1010	110010 1010
D20.5	101 10100	001011 1010	001011 1010

Table C-1: **Valid Data Characters *(Cont'd)***

Data Byte Name	Bits HGF EDCBA	Current RD – abcdei fghj	Current RD + abcdei fghj
D21.5	101 10101	101010 1010	101010 1010
D22.5	101 10110	011010 1010	011010 1010
D23.5	101 10111	111010 1010	000101 1010
D24.5	101 11000	110011 1010	001100 1010
D25.5	101 11001	100110 1010	100110 1010
D26.5	101 11010	010110 1010	010110 1010
D27.5	101 11011	110110 1010	001001 1010
D28.5	101 11100	001110 1010	001110 1010
D29.5	101 11101	101110 1010	010001 1010
D30.5	101 11110	011110 1010	100001 1010
D31.5	101 11111	101011 1010	010100 1010
D0.6	110 00000	100111 0110	011000 0110
D1.6	110 00001	011101 0110	100010 0110
D2.6	110 00010	101101 0110	010010 0110
D3.6	110 00011	110001 0110	110001 0110
D4.6	110 00100	110101 0110	001010 0110
D5.6	110 00101	101001 0110	101001 0110
D6.6	110 00110	011001 0110	011001 0110
D7.6	110 00111	111000 0110	000111 0110
D8.6	110 01000	111001 0110	000110 0110
D9.6	110 01001	100101 0110	100101 0110
D10.6	110 01010	010101 0110	010101 0110
D11.6	110 01011	110100 0110	110100 0110
D12.6	110 01100	001101 0110	001101 0110
D13.6	110 01101	101100 0110	101100 0110
D14.6	110 01110	011100 0110	011100 0110
D15.6	110 01111	010111 0110	101000 0110
D16.6	110 10000	011011 0110	100100 0110
D17.6	110 10001	100011 0110	100011 0110
D18.6	110 10010	010011 0110	010011 0110
D19.6	110 10011	110010 0110	110010 0110
D20.6	110 10100	001011 0110	001011 0110

Table C-1: **Valid Data Characters *(Cont'd)***

Data Byte Name	Bits HGF EDCBA	Current RD – abcdei fghj	Current RD + abcdei fghj
D21.6	110 10101	101010 0110	101010 0110
D22.6	110 10110	011010 0110	011010 0110
D23.6	110 10111	111010 0110	000101 0110
D24.6	110 11000	110011 0110	001100 0110
D25.6	110 11001	100110 0110	100110 0110
D26.6	110 11010	010110 0110	010110 0110
D27.6	110 11011	110110 0110	001001 0110
D28.6	110 11100	001110 0110	001110 0110
D29.6	110 11101	101110 0110	010001 0110
D30.6	110 11110	011110 0110	100001 0110
D31.6	110 11111	101011 0110	010100 0110
D0.7	111 00000	100111 0001	011000 1110
D1.7	111 00001	011101 0001	100010 1110
D2.7	111 00010	101101 0001	010010 1110
D3.7	111 00011	110001 1110	110001 0001
D4.7	111 00100	110101 0001	001010 1110
D5.7	111 00101	101001 1110	101001 0001
D6.7	111 00110	011001 1110	011001 0001
D7.7	111 00111	111000 1110	000111 0001
D8.7	111 01000	111001 0001	000110 1110
D9.7	111 01001	100101 1110	100101 0001
D10.7	111 01010	010101 1110	010101 0001
D11.7	111 01011	110100 1110	110100 1000
D12.7	111 01100	001101 1110	001101 0001
D13.7	111 01101	101100 1110	101100 1000
D14.7	111 01110	011100 1110	011100 1000
D15.7	111 01111	010111 0001	101000 1110
D16.7	111 10000	011011 0001	100100 1110
D17.7	111 10001	100011 0111	100011 0001
D18.7	111 10010	010011 0111	010011 0001
D19.7	111 10011	110010 1110	110010 0001
D20.7	111 10100	001011 0111	001011 0001

Table C-1: **Valid Data Characters *(Cont'd)***

Data Byte Name	Bits HGF EDCBA	Current RD – abcdei fghj	Current RD + abcdei fghj
D21.7	111 10101	101010 1110	101010 0001
D22.7	111 10110	011010 1110	011010 0001
D23.7	111 10111	111010 0001	000101 1110
D24.7	111 11000	110011 0001	001100 1110
D25.7	111 11001	100110 1110	100110 0001
D26.7	111 11010	010110 1110	010110 0001
D27.7	111 11011	110110 0001	001001 1110
D28.7	111 11100	001110 1110	001110 0001
D29.7	111 11101	101110 0001	010001 1110
D30.7	111 11110	011110 0001	100001 1110
D31.7	111 11111	101011 0001	010100 1110

Table C-2: **Valid Control K Characters**

Special Code Name	Bits HGF EDCBA	Current RD – abcdei fghj	Current RD + abcdei fghj
K28.0	000 11100	001111 0100	110000 1011
K28.1	001 11100	001111 1001	110000 0110
K28.2	010 11100	001111 0101	110000 1010
K28.3	011 11100	001111 0011	110000 1100
K28.4	100 11100	001111 0010	110000 1101
K28.5	101 11100	001111 1010	110000 0101
K28.6	110 11100	001111 0110	110000 1001
K28.7[(1)]	111 11100	001111 1000	110000 0111
K23.7	111 10111	111010 1000	000101 0111
K27.7	111 11011	110110 1000	001001 0111
K29.7	111 11101	101110 1000	010001 0111
K30.7	111 11110	011110 1000	100001 0111

Notes:

1. Used for testing and characterization only.

Appendix D

DRP Address Map of the GTX/GTH Transceiver

Table D-1 lists the DRP map of GTX2_COMMON primitive sorted by address.

Note: The reserved bits should NOT be modified. Attributes that are not described explicitly are set automatically by the 7 Series FPGAs Transceivers Wizard. These attributes must be left at their defaults, except for use cases that explicitly request different values.

Table D-1: **DRP Map of GTX2_COMMON Primitive**

DRP Address (Hex)	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Encoding
0030	15:0	R/W	QPLL_INIT_CFG	15:0	0–65535	0–65535
0031	14:11	R/W	QPLL_LPF	3:0	0–15	0–15
0031	7:0	R/W	QPLL_INIT_CFG	23:16	0–255	0–255
0032	15:0	R/W	QPLL_CFG	15:0	0–65535	0–65535
0033	15:11	R/W	QPLL_REFCLK_DIV	4:0	1	16
					2	0
					3	1
					4	2
					5	3
					6	5
					8	6
					10	7
					12	13
					16	14
					20	15
0033	10:0	R/W	QPLL_CFG	26:16	0–2047	0–2047
0034	15:0	R/W	QPLL_LOCK_CFG	15:0	0–65535	0–65535
0035	15:10	R/W	QPLL_COARSE_FREQ_OVRD	5:0	0–63	0–63
0035	9:0	R/W	QPLL_CP	9:0	0–1023	0–1023
0036	15	R/W	QPLL_DMONITOR_SEL	0	0–1	0–1

Table D-1: **DRP Map of GTX2_COMMON Primitive *(Cont'd)***

DRP Address (Hex)	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Encoding
0036	14	R/W	QPLL_FBDIV_MONITOR_EN	0	0–1	0–1
0036	13	R/W	QPLL_CP_MONITOR_EN	0	0–1	0–1
0036	11	R/W	QPLL_COARSE_FREQ_OVRD_EN	0	0–1	0–1
0036	9:0	R/W	QPLL_FBDIV	9:0	0–1023	0–1023
0037	6	R/W	QPLL_FBDIV_RATIO	0	0–1	0–1
0037	5:2	R/W	QPLL_CLKOUT_CFG	3:0	0–15	0–15
003E	15:0	R/W	BIAS_CFG	15:0	0–65535	0–65535
003F	15:0	R/W	BIAS_CFG	31:16	0–65535	0–65535
0040	15:0	R/W	BIAS_CFG	47:32	0–65535	0–65535
0041	15:0	R/W	BIAS_CFG	63:48	0–65535	0–65535
0043	15:0	R/W	COMMON_CFG	15:0	0–65535	0–65535
0044	15:0	R/W	COMMON_CFG	31:16	0–65535	0–65535

Table D-2 lists the DRP map of GTX2_CHANNEL primitive sorted by address.

Note: The reserved bits should NOT be modified. Attributes that are not described explicitly are set automatically by the 7 Series FPGAs Transceivers Wizard. These attributes must be left at their defaults, except for use cases that explicitly request different values.

Table D-2: **DRP Map of GTX2_CHANNEL Primitive**

DRP Address (Hex)	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Encoding
000	1	R/W	UCODEER_CLR	0	0–1	0–1
00D	15	R/W	RXDFELPMRESET_TIME	0	0–1	0–1
00D	14:10	R/W	RXCDRPHRESET_TIME	4:0	0–31	0–31
00D	9:5	R/W	RXCDRFREQRESET_TIME	4:0	0–31	0–31
00D	4:0	R/W	RXBUFRESET_TIME	4:0	0–31	0–31
00E	15:11	R/W	RXPCSRESET_TIME	4:0	0–31	0–31
00E	10:6	R/W	RXPMARESET_TIME	4:0	0–31	0–31
00E	5:0	R/W	RXDFELPMRESET_TIME	6:1	0–63	0–63
00F	14:10	R/W	RXISCANRESET_TIME	4:0	0–31	0–31
00F	9:5	R/W	TXPCSRESET_TIME	4:0	0–31	0–31
00F	4:0	R/W	TXPMARESET_TIME	4:0	0–31	0–31
011	14	R/W	RX_INT_DATAWIDTH	0	0–1	0–1
011	13:11	R/W	RX_DATA_WIDTH	2:0	16	2
					20	3
					32	4
					40	5
					64	6
					80	7

Table D-2: **DRP Map of GTX2_CHANNEL Primitive *(Cont'd)***

DRP Address (Hex)	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Encoding
011	10:6	R/W	RX_CLK25_DIV	4:0	1	0
					2	1
					3	2
					4	3
					5	4
					6	5
					7	6
					8	7
					9	8
					10	9
					11	10
					12	11
					13	12
					14	13
					15	14
					16	15
					17	16
					18	17
					19	18
					20	19
					21	20
					22	21
					23	22
					24	23
					25	24
					26	25
					27	26
					28	27
					29	28
					30	29
					31	30
					32	31
011	5:4	R/W	RX_CM_SEL	1:0	0–3	0–3
011	3:1	R/W	RX_CM_TRIM	2:0	0–7	0–7
011	0	R/W	RXPRBS_ERR_LOOPBACK	0	0–1	0–1
012	15:12	R/W	SATA_BURST_SEQ_LEN	3:0	0–15	0–15
012	11:10	R/W	OUTREFCLK_SEL_INV	1:0	0–3	0–3
012	9:7	R/W	SATA_BURST_VAL	2:0	0–7	0–7
012	6:0	R/W	RXOOB_CFG	6:0	0–127	0–127
013	14:9	R/W	SAS_MIN_COM	5:0	1–63	1–63

Table D-2: **DRP Map of GTX2_CHANNEL Primitive *(Cont'd)***

DRP Address (Hex)	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Encoding
013	8:3	R/W	SATA_MIN_BURST	5:0	1–61	1–61
013	2:0	R/W	SATA_EIDLE_VAL	2:0	0–7	0–7
014	11:6	R/W	SATA_MIN_WAKE	5:0	1–63	1–63
014	5:0	R/W	SATA_MIN_INIT	5:0	1–63	1–63
015	12:6	R/W	SAS_MAX_COM	6:0	1–127	1–127
015	5:0	R/W	SATA_MAX_BURST	5:0	1–63	1–63
016	11:6	R/W	SATA_MAX_WAKE	5:0	1–63	1–63
016	5:0	R/W	SATA_MAX_INIT	5:0	1–63	1–63
018	7:0	R/W	TRANS_TIME_RATE	7:0	0–255	0–255
019	12	R/W	TX_PREDRIVER_MODE	0	0–1	0–1
019	11:9	R/W	TX_EIDLE_DEASSERT_ DELAY	2:0	0–7	0–7
019	8:6	R/W	TX_EIDLE_ASSERT_ DELAY	2:0	0–7	0–7
019	5	R/W	TX_LOOPBACK_DRIVE_HIZ	0	FALSE	0
					TRUE	1
019	4:0	R/W	TX_DRIVE_MODE	4:0	DIRECT	0
					PIPE	1
					PIPEGEN3	2
01A	15:8	R/W	PD_TRANS_TIME_TO_P2	7:0	0–255	0–255
01A	7:0	R/W	PD_TRANS_TIME_NONE_P2	7:0	0–255	0–255
01B	12:1	R/W	PD_TRANS_TIME_FROM_P2	11:0	0–4095	0–4095
01B	0	R/W	PCS_PCIE_EN	0	FALSE	0
					TRUE	1
01C	15	R/W	TXBUF_RESET_ON_RATE_ CHANGE	0	FALSE	0
					TRUE	1
01C	14	R/W	TXBUF_EN	0	FALSE	0
					TRUE	1
01C	5	R/W	TXGEARBOX_EN	0	FALSE	0
					TRUE	1
01C	2:0	R/W	GEARBOX_MODE	2:0	0–7	0–7
01D	15:0	R/W	RX_DFE_GAIN_CFG	15:0	0–65535	0–65535
01E	6:0	R/W	RX_DFE_GAIN_CFG	22:16	0–127	0–127
01E	14	R/W	RX_DFE_LPM_HOLD_DURING _EIDLE	0	0–1	0–1
01F	11:0	R/W	RX_DFE_H2_CFG	11:0	0–4095	0–4095
020	11:0	R/W	RX_DFE_H3_CFG	11:0	0–4095	0–4095
021	10:0	R/W	RX_DFE_H4_CFG	10:0	0–2047	0–2047
022	10:0	R/W	RX_DFE_H5_CFG	10:0	0–2047	0–2047
023	12:0	R/W	RX_DFE_KL_CFG	12:0	0–8191	0–8191
024	15	R/W	RX_DFE_UT_CFG	0	0–1	0–1

Table D-2: **DRP Map of GTX2_CHANNEL Primitive *(Cont'd)***

DRP Address (Hex)	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Encoding
025	15:0	R/W	RX_DFE_UT_CFG	16:1	0–65535	0–65535
024	12:0	R/W	RX_OS_CFG	12:0	0–8191	0–8191
026	15:0	R/W	RX_DFE_VP_CFG	15:0	0–65535	0–65535
027	0	R/W	RX_DFE_VP_CFG	16	0–1	0–1
028	12:0	R/W	RX_DFE_XYD_CFG	12:0	0–8191	0–8191
029	15:0	R/W	RX_DFE_LPM_CFG	15:0	0–65535	0–65535
02A	13:0	R/W	RXLPM_HF_CFG	13:0	0–16383	0–16383
02B	13:0	R/W	RXLPM_LF_CFG	13:0	0–16383	0–16383
02C	15:0	R/W	ES_QUALIFIER	15:0	0–65535	0–65535
02D	15:0	R/W	ES_QUALIFIER	31:16	0–65535	0–65535
02E	15:0	R/W	ES_QUALIFIER	47:32	0–65535	0–65535
02F	15:0	R/W	ES_QUALIFIER	63:48	0–65535	0–65535
030	15:0	R/W	ES_QUALIFIER	79:64	0–65535	0–65535
031	15:0	R/W	ES_QUAL_MASK	15:0	0–65535	0–65535
032	15:0	R/W	ES_QUAL_MASK	31:16	0–65535	0–65535
033	15:0	R/W	ES_QUAL_MASK	47:32	0–65535	0–65535
034	15:0	R/W	ES_QUAL_MASK	63:48	0–65535	0–65535
035	15:0	R/W	ES_QUAL_MASK	79:64	0–65535	0–65535
036	15:0	R/W	ES_SDATA_MASK	15:0	0–65535	0–65535
037	15:0	R/W	ES_SDATA_MASK	31:16	0–65535	0–65535
038	15:0	R/W	ES_SDATA_MASK	47:32	0–65535	0–65535
039	15:0	R/W	ES_SDATA_MASK	63:48	0–65535	0–65535
03A	15:0	R/W	ES_SDATA_MASK	79:64	0–65535	0–65535
03B	15:11	R/W	ES_PRESCALE	4:0	0–31	0–31
03B	8:0	R/W	ES_VERT_OFFSET	8:0	0–511	0–511
03C	11:0	R/W	ES_HORZ_OFFSET	11:0	0–4095	0–4095
03D	15	R/W	RX_DISPERR_SEQ_MATCH	0	FALSE	0
					TRUE	1
03D	14	R/W	DEC_PCOMMA_DETECT	0	FALSE	0
					TRUE	1
03D	13	R/W	DEC_MCOMMA_DETECT	0	FALSE	0
					TRUE	1
03D	12	R/W	DEC_VALID_COMMA_ONLY	0	FALSE	0
					TRUE	1
03D	9	R/W	ES_ERRDET_EN	0	FALSE	0
					TRUE	1
03D	8	R/W	ES_EYE_SCAN_EN	0	FALSE	0
					TRUE	1
03D	5:0	R/W	ES_CONTROL	5:0	0–63	0–63
03E	9:0	R/W	ALIGN_COMMA_ENABLE	9:0	0–1023	0–1023

Table D-2: **DRP Map of GTX2_CHANNEL Primitive *(Cont'd)***

DRP Address (Hex)	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Encoding
03F	9:0	R/W	ALIGN_MCOMMA_VALUE	9:0	0–1023	0–1023
040	15:14	R/W	RXSLIDE_MODE	1:0	OFF	0
					AUTO	1
					PCS	2
					PMA	3
040	9:0	R/W	ALIGN_PCOMMA_VALUE	9:0	0–1023	0–1023
041	15:13	R/W	ALIGN_COMMA_WORD	2:0	1	1
					2	2
					4	4

Table D-2: **DRP Map of GTX2_CHANNEL Primitive *(Cont'd)***

DRP Address (Hex)	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Encoding
041	12:8	R/W	RX_SIG_VALID_DLY	4:0	1	0
					2	1
					3	2
					4	3
					5	4
					6	5
					7	6
					8	7
					9	8
					10	9
					11	10
					12	11
					13	12
					14	13
					15	14
					16	15
					17	16
					18	17
					19	18
					20	19
					21	20
					22	21
					23	22
					24	23
					25	24
					26	25
					27	26
					28	27
					29	28
					30	29
					31	30
					32	31
041	7	R/W	ALIGN_PCOMMA_DET	0	FALSE	0
					TRUE	1
041	6	R/W	ALIGN_MCOMMA_DET	0	FALSE	0
					TRUE	1
041	5	R/W	SHOW_REALIGN_COMMA	0	FALSE	0
					TRUE	1
041	4	R/W	ALIGN_COMMA_DOUBLE	0	FALSE	0
					TRUE	1

Table D-2: **DRP Map of GTX2_CHANNEL Primitive *(Cont'd)***

DRP Address (Hex)	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Encoding
041	3:0	R/W	RXSLIDE_AUTO_WAIT	3:0	0–15	0–15
044	14	R/W	CLK_CORRECT_USE	0	FALSE	0
					TRUE	1
044	13:10	R/W	CLK_COR_SEQ_1_ENABLE	3:0	0–15	0–15
044	9:0	R/W	CLK_COR_SEQ_1_1	9:0	0–1023	0–1023
045	15:10	R/W	CLK_COR_MAX_LAT	5:0	3–60	3–60
045	9:0	R/W	CLK_COR_SEQ_1_2	9:0	0–1023	0–1023
046	15:10	R/W	CLK_COR_MIN_LAT	5:0	3–60	3–60
046	9:0	R/W	CLK_COR_SEQ_1_3	9:0	0–1023	0–1023
047	14:10	R/W	CLK_COR_REPEAT_WAIT	4:0	0–31	0–31
047	9:0	R/W	CLK_COR_SEQ_1_4	9:0	0–1023	0–1023
048	14	R/W	CLK_COR_SEQ_2_USE	0	FALSE	0
					TRUE	1
048	13:10	R/W	CLK_COR_SEQ_2_ENABLE	3:0	0–15	0–15
048	9:0	R/W	CLK_COR_SEQ_2_1	9:0	0–1023	0–1023
049	13	R/W	CLK_COR_KEEP_IDLE	0	FALSE	0
					TRUE	1
049	12	R/W	CLK_COR_PRECEDENCE	0	FALSE	0
					TRUE	1
049	11:10	R/W	CLK_COR_SEQ_LEN	1:0	1	0
					2	1
					3	2
					4	3
049	9:0	R/W	CLK_COR_SEQ_2_2	9:0	0–1023	0–1023
04A	9:0	R/W	CLK_COR_SEQ_2_3	9:0	0–1023	0–1023
04B	15	R/W	RXGEARBOX_EN	0	FALSE	0
					TRUE	1
04B	9:0	R/W	CLK_COR_SEQ_2_4	9:0	0–1023	0–1023
04C	15:12	R/W	CHAN_BOND_SEQ_1_ ENABLE	3:0	0–15	0–15
04C	9:0	R/W	CHAN_BOND_SEQ_1_1	9:0	0–1023	0–1023
04D	15:14	R/W	CHAN_BOND_SEQ_LEN	1:0	1	0
					2	1
					3	2
					4	3
04D	9:0	R/W	CHAN_BOND_SEQ_1_2	9:0	0–1023	0–1023
04E	15	R/W	CHAN_BOND_KEEP_ALIGN	0	FALSE	0
					TRUE	1
04E	9:0	R/W	CHAN_BOND_SEQ_1_3	9:0	0–1023	0–1023
04F	9:0	R/W	CHAN_BOND_SEQ_1_4	9:0	0–1023	0–1023

Table D-2: **DRP Map of GTX2_CHANNEL Primitive *(Cont'd)***

DRP Address (Hex)	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Encoding
050	15:12	R/W	CHAN_BOND_SEQ_2_ ENABLE	3:0	0–15	0–15
050	11	R/W	CHAN_BOND_SEQ_2_USE	0	FALSE	0
					TRUE	1
050	9:0	R/W	CHAN_BOND_SEQ_2_1	9:0	0–1023	0–1023
051	15:12	R/W	FTS_LANE_DESKEW_CFG	3:0	0–15	0–15
051	11	R/W	FTS_LANE_DESKEW_EN	0	FALSE	0
					TRUE	1
051	9:0	R/W	CHAN_BOND_SEQ_2_2	9:0	0–1023	0–1023
052	15:12	R/W	FTS_DESKEW_SEQ_ENABLE	3:0	0–15	0–15
052	11	R/W	CBCC_DATA_SOURCE_SEL	0	ENCODED	0
					DECODED	1
052	9:0	R/W	CHAN_BOND_SEQ_2_3	9:0	0–1023	0–1023
053	15:12	R/W	CHAN_BOND_MAX_SKEW	3:0	1–14	1–14
053	9:0	R/W	CHAN_BOND_SEQ_2_4	9:0	0–1023	0–1023
054	15:0	R/W	RXDLY_TAP_CFG	15:0	0–65535	0–65535
055	15:0	R/W	RXDLY_CFG	15:0	0–65535	0–65535
057	12:8	R/W	RXPH_MONITOR_SEL	4:0	0–31	0–31
057	5:0	R/W	RX_DDI_SEL	5:0	0–63	0–63
059	7	R/W	TX_XCLK_SEL	0	TXOUT	0
					TXUSR	1
059	6	R/W	RX_XCLK_SEL	0	RXREC	0
					RXUSR	1
05B	15:0	R/W	CPLL_INIT_CFG	15:0	0–65535	0–65535
05C	7:0	R/W	CPLL_INIT_CFG	23:16	0–255	0–255
05C	15:8	R/W	CPLL_CFG	7:0	0–255	0–255
05D	15:0	R/W	CPLL_CFG	23:8	0–65535	0–65535
05E	15:14	R/W	SATA_CPLL_CFG	1:0	VCO_ 3000MHZ	0
					VCO_ 1500MHZ	1
					VCO_ 750MHZ	2

Table D-2: **DRP Map of GTX2_CHANNEL Primitive *(Cont'd)***

DRP Address (Hex)	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Encoding
05E	12:8	R/W	CPLL_REFCLK_DIV	4:0	1	16
					2	0
					3	1
					4	2
					5	3
					6	5
					8	6
					10	7
					12	13
					16	14
					20	15
05E	7	R/W	CPLL_FBDIV_45	0	4	0
					5	1
05E	6:0	R/W	CPLL_FBDIV	6:0	1	16
					2	0
					3	1
					4	2
					5	3
					6	5
					8	6
					10	7
					12	13
					16	14
					20	15
05F	15:0	R/W	CPLL_LOCK_CFG	15:0	0–65535	0–65535
060	15:0	R/W	TXPHDLY_CFG	15:0	0–65535	0–65535
061	7:0	R/W	TXPHDLY_CFG	23:16	0–255	0–255
062	15:0	R/W	TXDLY_CFG	15:0	0–65535	0–65535
063	15:0	R/W	TXDLY_TAP_CFG	15:0	0–65535	0–65535
064	15:0	R/W	TXPH_CFG	15:0	0–65535	0–65535
065	12:8	R/W	TXPH_MONITOR_SEL	4:0	0–31	0–31
066	11:0	R/W	RX_BIAS_CFG	11:0	0–4095	0–4095
068	1	R/W	TX_CLKMUX_PD	0	0–1	0–1
068	0	R/W	RX_CLKMUX_PD	0	0–1	0–1
069	8	R/W	TERM_RCAL_OVRD	0	0–1	0–1
069	4:0	R/W	TERM_RCAL_CFG	4:0	0–31	0–31

Table D-2: **DRP Map of GTX2_CHANNEL Primitive *(Cont'd)***

DRP Address (Hex)	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Encoding
06A	4:0	R/W	TX_CLK25_DIV	4:0	1	0
					2	1
					3	2
					4	3
					5	4
					6	5
					7	6
					8	7
					9	8
					10	9
					11	10
					12	11
					13	12
					14	13
					15	14
					16	15
					17	16
					18	17
					19	18
					20	19
					21	20
					22	21
					23	22
					24	23
					25	24
					26	25
					27	26
					28	27
					29	28
					30	29
					31	30
					32	31
06B	15	R/W	TX_QPI_STATUS_EN	0	0–1	0–1
06B	4	R/W	TX_INT_DATAWIDTH	0	0	0
					1	1

Table D-2: **DRP Map of GTX2_CHANNEL Primitive *(Cont'd)***

DRP Address (Hex)	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Encoding
06B	2:0	R/W	TX_DATA_WIDTH	2:0	16	2
					20	3
					32	4
					40	5
					64	6
					80	7
06F	15:0	R/W	PCS_RSVD_ATTR	15:0	0–65535	0–65535
070	15:0	R/W	PCS_RSVD_ATTR	31:16	0–65535	0–65535
071	15:0	R/W	PCS_RSVD_ATTR	47:32	0–65535	0–65535
074	15:11	R/W	RX_DFE_KL_CFG2	8:4	0–31	0–31
074	3:0	R/W	RX_DFE_KL_CFG2	3:0	0–15	0–15
075	14:8	R/W	TX_MARGIN_FULL_1	6:0	0–127	0–127
075	6:0	R/W	TX_MARGIN_FULL_0	6:0	0–127	0–127
076	14:8	R/W	TX_MARGIN_FULL_3	6:0	0–127	0–127
076	6:0	R/W	TX_MARGIN_FULL_2	6:0	0–127	0–127
077	14:8	R/W	TX_MARGIN_LOW_0	6:0	0–127	0–127
077	6:0	R/W	TX_MARGIN_FULL_4	6:0	0–127	0–127
078	14:8	R/W	TX_MARGIN_LOW_2	6:0	0–127	0–127
078	6:0	R/W	TX_MARGIN_LOW_1	6:0	0–127	0–127
079	14:8	R/W	TX_MARGIN_LOW_4	6:0	0–127	0–127
079	6:0	R/W	TX_MARGIN_LOW_3	6:0	0–127	0–127
07A	12:8	R/W	TX_DEEMPH1	4:0	0–31	0–31
07A	4:0	R/W	TX_DEEMPH0	4:0	0–31	0–31
07C	10:8	R/W	TX_RXDETECT_REF	2:0	0–7	0–7
07C	3	R/W	TX_MAINCURSOR_SEL	0	0–1	0–1
07C	1:0	R/W	PMA_RSV3	1:0	0–3	0–3
07D	13:0	R/W	TX_RXDETECT_CFG	13:0	0–16383	0–16383
07F	14:10	R/W	RX_DFE_KL_CFG2	17:13	0–31	0–31
07F	3:0	R/W	RX_DFE_KL_CFG2	12:9	0–15	0–15
082	15:0	R/W	PMA_RSV2	15:0	0–65535	0–65535
083	15:7	R/W	RX_DFE_KL_CFG2	26:18	0–511	0–511
086	15:0	R/W	DMONITOR_CFG	15:0	0–65535	0–65535
087	7:0	R/W	DMONITOR_CFG	23:16	0–255	0–255
088	6:4	R/W	TXOUT_DIV	2:0	1	0
					2	1
					4	2
					8	3
					16	4

Table D-2: **DRP Map of GTX2_CHANNEL Primitive *(Cont'd)***

DRP Address (Hex)	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Encoding
088	2:0	R/W	RXOUT_DIV	2:0	1	0
					2	1
					4	2
					8	3
					16	4
08C	7:3	R/W	RX_DFE_KL_CFG2	31:27	0–31	0–31
091	15:0	R/W	PMA_RSV4	15:0	0–65535	0–65535
092	15:0	R/W	PMA_RSV4	31:16	0–65535	0–65535
097	15:0	R/W	TST_RSV	15:0	0–65535	0–65535
098	15:0	R/W	TST_RSV	31:16	0–65535	0–65535
099	15:0	R/W	PMA_RSV	15:0	0–65535	0–65535
09A	15:0	R/W	PMA_RSV	31:16	0–65535	0–65535
09B	5:0	R/W	RX_BUFFER_CFG	5:0	0–63	0–63
09C	13:8	R/W	RXBUF_THRESH_OVFLW	5:0	0–63	0–63
09C	5:0	R/W	RXBUF_THRESH_UNDFLW	5:0	0–63	0–63
09D	15:12	R/W	RXBUF_EIDLE_HI_CNT	3:0	0–15	0–15
09D	11:8	R/W	RXBUF_EIDLE_LO_CNT	3:0	0–15	0–15
09D	7	R/W	RXBUF_ADDR_MODE	0	FULL	0
					FAST	1
09D	6	R/W	RXBUF_RESET_ON_EIDLE	0	FALSE	0
					TRUE	1
09D	5	R/W	RXBUF_RESET_ON_CB_CHANGE	0	FALSE	0
					TRUE	1
09D	4	R/W	RXBUF_RESET_ON_RATE_CHANGE	0	FALSE	0
					TRUE	1
09D	3	R/W	RXBUF_RESET_ON_COMMAALIGN	0	FALSE	0
					TRUE	1
09D	2	R/W	RXBUF_THRESH_OVRD	0	FALSE	0
					TRUE	1
09D	1	R/W	RXBUF_EN	0	FALSE	0
					TRUE	1
09D	0	R/W	RX_DEFER_RESET_BUF_EN	0	FALSE	0
					TRUE	1
09F	8:0	R/W	TXDLY_LCFG	8:0	0–511	0–511
0A0	8:0	R/W	RXDLY_LCFG	8:0	0–511	0–511
0A1	15:0	R/W	RXPH_CFG	15:0	0–65535	0–65535
0A2	7:0	R/W	RXPH_CFG	23:16	0–255	0–255
0A3	15:0	R/W	RXPHDLY_CFG	15:0	0–65535	0–65535
0A4	7:0	R/W	RXPHDLY_CFG	23:16	0–255	0–255
0A5	11:0	R/W	RX_DEBUG_CFG	11:0	0–2047	0–2047

Table D-2: **DRP Map of GTX2_CHANNEL Primitive *(Cont'd)***

DRP Address (Hex)	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Encoding
0A6	9:0	R/W	ES_PMA_CFG	9:0	0–1023	0–1023
0A7	13	R/W	RXCDR_PH_RESET_ON_EIDLE	0	0–1	0–1
0A7	12	R/W	RXCDR_FR_RESET_ON_EIDLE	0	0–1	0–1
0A7	11	R/W	RXCDR_HOLD_DURING_ EIDLE	0	0–1	0–1
0A7	5:0	R/W	RXCDR_LOCK_CFG	5:0	0–63	0–63
0A8	15:0	R/W	RXCDR_CFG	15:0	0–65535	0–65535
0A9	15:0	R/W	RXCDR_CFG	31:16	0–65535	0–65535
0AA	15:0	R/W	RXCDR_CFG	47:32	0–65535	0–65535
0AB	15:0	R/W	RXCDR_CFG	63:48	0–65535	0–65535
0AC	7:0	R/W	RXCDR_CFG	71:64	0–255	0–255

Table D-3 lists the DRP map of GTH2_COMMON primitive sorted by address.

Note: The reserved bits should NOT be modified. Attributes that are not described explicitly are set automatically by the 7 Series FPGAs Transceivers Wizard. These attributes must be left at their defaults, except for use cases that explicitly request different values.

Table D-3: **DRP Map of GTH2_COMMON Primitive**

DRP Address (Hex)	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Encoding
0030	15:0	R/W	QPLL_INIT_CFG	15:0	0–65535	0–65535
0031	14:11	R/W	QPLL_LPF	3:0	0–15	0–15
0031	7:0	R/W	QPLL_INIT_CFG	23:16	0–255	0–255
0032	15:0	R/W	QPLL_CFG	15:0	0–65535	0–65535
0033	15:11	R/W	QPLL_REFCLK_DIV	4:0	1	16
					2	0
					3	1
					4	2
					5	3
					6	5
					8	6
					10	7
					12	13
					16	14
					20	15
0033	10:0	R/W	QPLL_CFG	26:16	0–2047	0–2047
0034	15:0	R/W	QPLL_LOCK_CFG	15:0	0–65535	0–65535
0035	15:10	R/W	QPLL_COARSE_FREQ_OVRD	5:0	0–63	0–63

Table D-3: **DRP Map of GTH2_COMMON Primitive *(Cont'd)***

DRP Address (Hex)	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Encoding
0035	9:0	R/W	QPLL_CP	9:0	0–1023	0–1023
0036	15	R/W	QPLL_DMONITOR_SEL	0	0–1	0–1
0036	14	R/W	QPLL_FBDIV_MONITOR_EN	0	0–1	0–1
0036	13	R/W	QPLL_CP_MONITOR_EN	0	0–1	0–1
0036	11	R/W	QPLL_COARSE_FREQ_OVRD_EN	0	0–1	0–1
0036	9:0	R/W	QPLL_FBDIV	9:0	0–1023	0–1023
0037	6	R/W	QPLL_FBDIV_RATIO	0	0–1	0–1
0037	5:2	R/W	QPLL_CLKOUT_CFG	3:0	0–15	0–15
003D	15:0	R/W	RSVD_ATTR1	15:0	0–65535	0–65535
003E	15:0	R/W	BIAS_CFG	15:0	0–65535	0–65535
003F	15:0	R/W	BIAS_CFG	31:16	0–65535	0–65535
0040	15:0	R/W	BIAS_CFG	47:32	0–65535	0–65535
0041	15:0	R/W	BIAS_CFG	63:48	0–65535	0–65535
0042	15:0	R/W	RSVD_ATTR0	15:0	0–65535	0–65535
0043	15:0	R/W	COMMON_CFG	15:0	0–65535	0–65535
0044	15:0	R/W	COMMON_CFG	31:16	0–65535	0–65535
0047	14:13	R/W	RCAL_CFG	1:0	0–3	0–3
0047	12	R/W	QPLL_RP_COMP	0	0–1	0–1
0047	11:10	R/W	QPLL_VTRL_RESET	1:0	0–3	0–3

Table D-4 lists the DRP map of GTH2_CHANNEL primitive sorted by address.

Note: The reserved bits should NOT be modified. Attributes that are not described explicitly are set automatically by the 7 Series FPGAs Transceivers Wizard. These attributes must be left at their defaults, except for use cases that explicitly request different values.

Table D-4: **DRP Map of GTH2_CHANNEL Primitive**

DRP Address (Hex)	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Binary Encoding
0000	15	R/W	ACJTAG_RESET	0	0–1	0–1
0000	14	R/W	ACJTAG_DEBUG_MODE	0	0–1	0–1
0000	13	R/W	ACJTAG_MODE	0	0–1	0–1
0000	1	R/W	UCODEER_CLR	0	0–1	0–1
000A	5	R/W	A_RXOSCALRESET	0	0–1	0–1
000C	15:9	R/W	RXDFELPMRESET_TIME	6:0	0–127	0–127
000D	14:10	R/W	RXCDRPHRESET_TIME	4:0	0–31	0–31
000D	9:5	R/W	RXCDRFREQRESET_TIME	4:0	0–31	0–31
000D	4:0	R/W	RXBUFRESET_TIME	4:0	0–31	0–31

Table D-4: **DRP Map of GTH2_CHANNEL Primitive *(Cont'd)***

DRP Address (Hex)	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Binary Encoding
000E	15:11	R/W	RXPCSRESET_TIME	4:0	0–31	0–31
000E	10:6	R/W	RXPMARESET_TIME	4:0	0–31	0–31
000F	14:10	R/W	RXISCANRESET_TIME	4:0	0–31	0–31
000F	9:5	R/W	TXPCSRESET_TIME	4:0	0–31	0–31
000F	4:0	R/W	TXPMARESET_TIME	4:0	0–31	0–31
0010	15	R/W	RXSYNC_OVRD	0	0–1	0–1
0010	14	R/W	TXSYNC_OVRD	0	0–1	0–1
0010	13	R/W	TXSYNC_SKIP_DA	0	0–1	0–1
0010	12	R/W	RXSYNC_SKIP_DA	0	0–1	0–1
0010	11	R/W	TXSYNC_MULTILANE	0	0–1	0–1
0010	10	R/W	RXSYNC_MULTILANE	0	0–1	0–1
0011	14	R/W	RX_INT_DATAWIDTH	0	0–1	0–1
0011	13:11	R/W	RX_DATA_WIDTH	2:0	16	2
					20	3
					32	4
					40	5
					64	6
					80	7

Table D-4: **DRP Map of GTH2_CHANNEL Primitive *(Cont'd)***

DRP Address (Hex)	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Binary Encoding
0011	10:6	R/W	RX_CLK25_DIV	4:0	1	0
					2	1
					3	2
					4	3
					5	4
					6	5
					7	6
					8	7
					9	8
					10	9
					11	10
					12	11
					13	12
					14	13
					15	14
					16	15
					17	16
					18	17
					19	18
					20	19
					21	20
					22	21
					23	22
					24	23
					25	24
					26	25
					27	26
					28	27
					29	28
					30	29
					31	30
					32	31
0011	5:4	R/W	RX_CM_SEL	1:0	0–3	0–3
0011	0	R/W	RXPRBS_ERR_LOOPBACK	0	0–1	0–1
0012	15:12	R/W	SATA_BURST_SEQ_LEN	3:0	0–15	0–15
0012	11:10	R/W	OUTREFCLK_SEL_INV	1:0	0–3	0–3
0012	9:7	R/W	SATA_BURST_VAL	2:0	0–7	0–7
0012	6:0	R/W	RXOOB_CFG	6:0	0–127	0–127
0013	14:9	R/W	SAS_MIN_COM	5:0	1–63	1–63
0013	8:3	R/W	SATA_MIN_BURST	5:0	1–61	1–61

Table D-4: **DRP Map of GTH2_CHANNEL Primitive *(Cont'd)***

DRP Address (Hex)	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Binary Encoding
0013	2:0	R/W	SATA_EIDLE_VAL	2:0	0–7	0–7
0014	11:6	R/W	SATA_MIN_WAKE	5:0	1–63	1–63
0014	5:0	R/W	SATA_MIN_INIT	5:0	1–63	1–63
0015	12:6	R/W	SAS_MAX_COM	6:0	1–127	1–127
0015	5:0	R/W	SATA_MAX_BURST	5:0	1–63	1–63
0016	11:6	R/W	SATA_MAX_WAKE	5:0	1–63	1–63
0016	5:0	R/W	SATA_MAX_INIT	5:0	1–63	1–63
0017	15:11	R/W	RXOSCALRESET_TIMEOUT	4:0	0–31	0–31
0017	10:6	R/W	RXOSCALRESET_TIME	4:0	0–31	0–31
0018	7:0	R/W	TRANS_TIME_RATE	7:0	0–255	0–255
0019	11:9	R/W	TX_EIDLE_DEASSERT_DELAY	2:0	0–7	0–7
0019	8:6	R/W	TX_EIDLE_ASSERT_DELAY	2:0	0–7	0–7
0019	5	R/W	TX_LOOPBACK_DRIVE_HIZ	0	FALSE	0
					TRUE	1
0019	4:0	R/W	TX_DRIVE_MODE	4:0	DIRECT	0
					PIPE	1
					PIPEGEN3	2
001A	15:8	R/W	PD_TRANS_TIME_TO_P2	7:0	0–255	0–255
001A	7:0	R/W	PD_TRANS_TIME_NONE_P2	7:0	0–255	0–255
001B	12:1	R/W	PD_TRANS_TIME_FROM_P2	11:0	0–4095	0–4095
001B	0	R/W	PCS_PCIE_EN	0	FALSE	0
					TRUE	1
001C	15	R/W	TXBUF_RESET_ON_RATE_CHANGE	0	FALSE	0
					TRUE	1
001C	14	R/W	TXBUF_EN	0	FALSE	0
					TRUE	1
001C	5	R/W	TXGEARBOX_EN	0	FALSE	0
					TRUE	1
001C	2:0	R/W	GEARBOX_MODE	2:0	0–7	0–7
001D	15:0	R/W	RX_DFE_GAIN_CFG	15:0	0–65535	0–65535
001E	14	R/W	RX_DFE_LPM_HOLD_DURING_EIDLE	0	0–1	0–1
001E	6:0	R/W	RX_DFE_GAIN_CFG	22:16	0–127	0–127
001F	11:0	R/W	RX_DFE_H2_CFG	11:0	0–4095	0–4095
0020	11:0	R/W	RX_DFE_H3_CFG	11:0	0–4095	0–4095
0021	10:0	R/W	RX_DFE_H4_CFG	10:0	0–2047	0–2047
0022	10:0	R/W	RX_DFE_H5_CFG	10:0	0–2047	0–2047
0024	15	R/W	RX_DFE_UT_CFG	16	0–1	0–1
0024	12:0	R/W	RX_OS_CFG	12:0	0–8191	0–8191
0025	15:0	R/W	RX_DFE_UT_CFG	15:0	0–65535	0–65535

Table D-4: **DRP Map of GTH2_CHANNEL Primitive *(Cont'd)***

DRP Address (Hex)	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Binary Encoding
0026	15:0	R/W	RX_DFE_VP_CFG	15:0	0–65535	0–65535
0027	0	R/W	RX_DFE_VP_CFG	16	0–1	0–1
0029	15:0	R/W	RX_DFE_LPM_CFG	15:0	0–65535	0–65535
002A	15:14	R/W	RXLPM_LF_CFG	17:16	0–3	0–3
002A	13:0	R/W	RXLPM_HF_CFG	13:0	0–16383	0–16383
002B	15:0	R/W	RXLPM_LF_CFG	15:0	0–65535	0–65535
002C	15:0	R/W	ES_QUALIFIER	15:0	0–65535	0–65535
002D	15:0	R/W	ES_QUALIFIER	31:16	0–65535	0–65535
002E	15:0	R/W	ES_QUALIFIER	47:32	0–65535	0–65535
002F	15:0	R/W	ES_QUALIFIER	63:48	0–65535	0–65535
0030	15:0	R/W	ES_QUALIFIER	79:64	0–65535	0–65535
0031	15:0	R/W	ES_QUAL_MASK	15:0	0–65535	0–65535
0032	15:0	R/W	ES_QUAL_MASK	31:16	0–65535	0–65535
0033	15:0	R/W	ES_QUAL_MASK	47:32	0–65535	0–65535
0034	15:0	R/W	ES_QUAL_MASK	63:48	0–65535	0–65535
0035	15:0	R/W	ES_QUAL_MASK	79:64	0–65535	0–65535
0036	15:0	R/W	ES_SDATA_MASK	15:0	0–65535	0–65535
0037	15:0	R/W	ES_SDATA_MASK	31:16	0–65535	0–65535
0038	15:0	R/W	ES_SDATA_MASK	47:32	0–65535	0–65535
0039	15:0	R/W	ES_SDATA_MASK	63:48	0–65535	0–65535
003A	15:0	R/W	ES_SDATA_MASK	79:64	0–65535	0–65535
003B	15:11	R/W	ES_PRESCALE	4:0	0–31	0–31
003B	8:0	R/W	ES_VERT_OFFSET	8:0	0–511	0–511
003C	11:0	R/W	ES_HORZ_OFFSET	11:0	0–4095	0–4095
003D	15	R/W	RX_DISPERR_SEQ_MATCH	0	FALSE	0
					TRUE	1
003D	14	R/W	DEC_PCOMMA_DETECT	0	FALSE	0
					TRUE	1
003D	13	R/W	DEC_MCOMMA_DETECT	0	FALSE	0
					TRUE	1
003D	12	R/W	DEC_VALID_COMMA_ONLY	0	FALSE	0
					TRUE	1
003D	9	R/W	ES_ERRDET_EN	0	FALSE	0
					TRUE	1
003D	8	R/W	ES_EYE_SCAN_EN	0	FALSE	0
					TRUE	1
003D	5:0	R/W	ES_CONTROL	5:0	0–63	0–63
003E	9:0	R/W	ALIGN_COMMA_ENABLE	9:0	0–1023	0–1023
003F	9:0	R/W	ALIGN_MCOMMA_VALUE	9:0	0–1023	0–1023

Table D-4: **DRP Map of GTH2_CHANNEL Primitive *(Cont'd)***

DRP Address (Hex)	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Binary Encoding
0040	15:14	R/W	RXSLIDE_MODE	1:0	OFF	0
					AUTO	1
					PCS	2
					PMA	3
0040	9:0	R/W	ALIGN_PCOMMA_VALUE	9:0	0–1023	0–1023
0041	15:13	R/W	ALIGN_COMMA_WORD	2:0	1	1
					2	2
					4	4
0041	12:8	R/W	RX_SIG_VALID_DLY	4:0	1	0
					2	1
					3	2
					4	3
					5	4
					6	5
					7	6
					8	7
					9	8
					10	9
					11	10
					12	11
					13	12
					14	13
					15	14
					16	15
					17	16
					18	17
					19	18
					20	19
					21	20
					22	21
					23	22
					24	23
					25	24
					26	25
					27	26
					28	27
					29	28
					30	29
					31	30
					32	31

Table D-4: **DRP Map of GTH2_CHANNEL Primitive *(Cont'd)***

DRP Address (Hex)	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Binary Encoding
0041	7	R/W	ALIGN_PCOMMA_DET	0	FALSE	0
					TRUE	1
0041	6	R/W	ALIGN_MCOMMA_DET	0	FALSE	0
					TRUE	1
0041	5	R/W	SHOW_REALIGN_COMMA	0	FALSE	0
					TRUE	1
0041	4	R/W	ALIGN_COMMA_DOUBLE	0	FALSE	0
					TRUE	1
0041	3:0	R/W	RXSLIDE_AUTO_WAIT	3:0	0–15	0–15
0044	14	R/W	CLK_CORRECT_USE	0	FALSE	0
					TRUE	1
0044	13:10	R/W	CLK_COR_SEQ_1_ENABLE	3:0	0–15	0–15
0044	9:0	R/W	CLK_COR_SEQ_1_1	9:0	0–1023	0–1023
0045	15:10	R/W	CLK_COR_MAX_LAT	5:0	3–60	3–60
0045	9:0	R/W	CLK_COR_SEQ_1_2	9:0	0–1023	0–1023
0046	15:10	R/W	CLK_COR_MIN_LAT	5:0	3–60	3–60
0046	9:0	R/W	CLK_COR_SEQ_1_3	9:0	0–1023	0–1023
0047	14:10	R/W	CLK_COR_REPEAT_WAIT	4:0	0–31	0–31
0047	9:0	R/W	CLK_COR_SEQ_1_4	9:0	0–1023	0–1023
0048	14	R/W	CLK_COR_SEQ_2_USE	0	FALSE	0
					TRUE	1
0048	13:10	R/W	CLK_COR_SEQ_2_ENABLE	3:0	0–15	0–15
0048	9:0	R/W	CLK_COR_SEQ_2_1	9:0	0–1023	0–1023
0049	13	R/W	CLK_COR_KEEP_IDLE	0	FALSE	0
					TRUE	1
0049	12	R/W	CLK_COR_PRECEDENCE	0	FALSE	0
					TRUE	1
0049	11:10	R/W	CLK_COR_SEQ_LEN	1:0	1	0
					2	1
					3	2
					4	3
0049	9:0	R/W	CLK_COR_SEQ_2_2	9:0	0–1023	0–1023
004A	9:0	R/W	CLK_COR_SEQ_2_3	9:0	0–1023	0–1023
004B	15	R/W	RXGEARBOX_EN	0	FALSE	0
					TRUE	1
004B	9:0	R/W	CLK_COR_SEQ_2_4	9:0	0–1023	0–1023
004C	15:12	R/W	CHAN_BOND_SEQ_1_ ENABLE	3:0	0–15	0–15
004C	9:0	R/W	CHAN_BOND_SEQ_1_1	9:0	0–1023	0–1023

Table D-4: **DRP Map of GTH2_CHANNEL Primitive *(Cont'd)***

DRP Address (Hex)	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Binary Encoding
004D	15:14	R/W	CHAN_BOND_SEQ_LEN	1:0	1	0
					2	1
					3	2
					4	3
004D	9:0	R/W	CHAN_BOND_SEQ_1_2	9:0	0–1023	0–1023
004E	15	R/W	CHAN_BOND_KEEP_ALIGN	0	FALSE	0
					TRUE	1
004E	9:0	R/W	CHAN_BOND_SEQ_1_3	9:0	0–1023	0–1023
004F	9:0	R/W	CHAN_BOND_SEQ_1_4	9:0	0–1023	0–1023
0050	15:12	R/W	CHAN_BOND_SEQ_2_ ENABLE	3:0	0–15	0–15
0050	11	R/W	CHAN_BOND_SEQ_2_USE	0	FALSE	0
					TRUE	1
0050	9:0	R/W	CHAN_BOND_SEQ_2_1	9:0	0–1023	0–1023
0051	15:12	R/W	FTS_LANE_DESKEW_CFG	3:0	0–15	0–15
0051	11	R/W	FTS_LANE_DESKEW_EN	0	FALSE	0
					TRUE	1
0051	9:0	R/W	CHAN_BOND_SEQ_2_2	9:0	0–1023	0–1023
0052	15:12	R/W	FTS_DESKEW_SEQ_ENABLE	3:0	0–15	0–15
0052	11	R/W	CBCC_DATA_SOURCE_SEL	0	ENCODED	0
					DECODED	1
0052	9:0	R/W	CHAN_BOND_SEQ_2_3	9:0	0–1023	0–1023
0053	15:12	R/W	CHAN_BOND_MAX_SKEW	3:0	1–14	1–14
0053	9:0	R/W	CHAN_BOND_SEQ_2_4	9:0	0–1023	0–1023
0054	15:0	R/W	RXDLY_TAP_CFG	15:0	0–65535	0–65535
0055	15:0	R/W	RXDLY_CFG	15:0	0–65535	0–65535
0057	12:8	R/W	RXPH_MONITOR_SEL	4:0	0–31	0–31
0057	5:0	R/W	RX_DDI_SEL	5:0	0–63	0–63
0059	7	R/W	TX_XCLK_SEL	0	TXOUT	0
					TXUSR	1
0059	6	R/W	RX_XCLK_SEL	0	RXREC	0
					RXUSR	1
005A	9	R/W	TXOOB_CFG	0	0–1	0–1
005A	8	R/W	LOOPBACK_CFG	0	0–1	0–1
005B	15:0	R/W	CPLL_INIT_CFG	15:0	0–65535	0–65535
005C	15:14	R/W	RXPI_CFG3	1:0	0–3	0–3
005C	13:12	R/W	RXPI_CFG0	1:0	0–3	0–3
005C	7:0	R/W	CPLL_INIT_CFG	23:16	0–255	0–255
005D	14:13	R/W	RXPI_CFG2	1:0	0–3	0–3
005D	12:11	R/W	RXPI_CFG1	1:0	0–3	0–3

Table D-4: **DRP Map of GTH2_CHANNEL Primitive *(Cont'd)***

DRP Address (Hex)	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Binary Encoding
005D	10:8	R/W	TXPI_CFG5	2:0	0–7	0–7
005D	7	R/W	TXPI_CFG4	0	0–1	0–1
005D	6	R/W	TXPI_CFG3	0	0–1	0–1
005D	5:4	R/W	TXPI_CFG2	1:0	0–3	0–3
005D	3:2	R/W	TXPI_CFG1	1:0	0–3	0–3
005D	1:0	R/W	TXPI_CFG0	1:0	0–3	0–3
005E	15:14	R/W	SATA_CPLL_CFG	1:0	VCO_3000MHZ	0
					VCO_1500MHZ	1
					VCO_750MHZ	2
005E	12:8	R/W	CPLL_REFCLK_DIV	4:0	1	16
					2	0
					3	1
					4	2
					5	3
					6	5
					8	6
					10	7
					12	13
					16	14
					20	15
005E	7	R/W	CPLL_FBDIV_45	0	4	0
					5	1
005E	6:0	R/W	CPLL_FBDIV	6:0	1	16
					2	0
					3	1
					4	2
					5	3
					6	5
					8	6
					10	7
					12	13
					16	14
					20	15
005F	15:0	R/W	CPLL_LOCK_CFG	15:0	0–65535	0–65535
0060	15:0	R/W	TXPHDLY_CFG	15:0	0–65535	0–65535
0061	15	R/W	RXPI_CFG5	0	0–1	0–1
0061	14	R/W	RXPI_CFG4	0	0–1	0–1
0061	7:0	R/W	TXPHDLY_CFG	23:16	0–255	0–255

Table D-4: **DRP Map of GTH2_CHANNEL Primitive *(Cont'd)***

DRP Address (Hex)	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Binary Encoding
0062	15:0	R/W	TXDLY_CFG	15:0	0–65535	0–65535
0063	15:0	R/W	TXDLY_TAP_CFG	15:0	0–65535	0–65535
0064	15:0	R/W	TXPH_CFG	15:0	0–65535	0–65535
0065	12:8	R/W	TXPH_MONITOR_SEL	4:0	0–31	0–31
0065	2:0	R/W	RXPI_CFG6	2:0	0–7	0–7
0066	15:0	R/W	RX_BIAS_CFG	15:0	0–65535	0–65535
0067	7:0	R/W	RX_BIAS_CFG	23:16	0–255	0–255
0068	3	R/W	RXOOB_CLK_CFG	0	PMA	0
					FABRIC	1
0068	1	R/W	TX_CLKMUX_EN	0	0–1	0–1
0068	0	R/W	RX_CLKMUX_EN	0	0–1	0–1
0069	14:0	R/W	TERM_RCAL_CFG	14:0	0–32767	0–32767
006A	15:13	R/W	TERM_RCAL_OVRD	2:0	0–7	0–7
006A	4:0	R/W	TX_CLK25_DIV	4:0	1	0
					2	1
					3	2
					4	3
					5	4
					6	5
					7	6
					8	7
					9	8
					10	9
					11	10

Table D-4: **DRP Map of GTH2_CHANNEL Primitive *(Cont'd)***

DRP Address (Hex)	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Binary Encoding
006A	4:0	R/W	TX_CLK25_DIV *(Cont'd)*	4:0	12	11
					13	12
					14	13
					15	14
					16	15
					17	16
					18	17
					19	18
					20	19
					21	20
					22	21
					23	22
					24	23
					25	24
					26	25
					27	26
					28	27
					29	28
					30	29
					31	30
					32	31
006B	15	R/W	TX_QPI_STATUS_EN	0	0–1	0–1
006B	11:8	R/W	PMA_RSV5	3:0	0–15	0–15
006B	4	R/W	TX_INT_DATAWIDTH	0	0	0
					1	1
006B	2:0	R/W	TX_DATA_WIDTH	2:0	16	2
					20	3
					32	4
					40	5
					64	6
					80	7
006F	15:0	R/W	PCS_RSVD_ATTR	15:0	0–65535	0–65535
0070	15:0	R/W	PCS_RSVD_ATTR	31:16	0–65535	0–65535
0071	15:0	R/W	PCS_RSVD_ATTR	47:32	0–65535	0–65535
0072	15:0	R/W	RX_DFE_KL_CFG	15:0	0–65535	0–65535
0073	15:0	R/W	RX_DFE_KL_CFG	31:16	0–65535	0–65535
0074	12:9	R/W	RX_DFE_AGC_CFG2	3:0	0–15	0–15
0074	8:6	R/W	RX_DFE_AGC_CFG1	2:0	0–7	0–7
0074	5:4	R/W	RX_DFE_AGC_CFG0	1:0	0–3	0–3
0074	0	R/W	RX_DFE_KL_CFG	32	0–1	0–1

Table D-4: **DRP Map of GTH2_CHANNEL Primitive *(Cont'd)***

DRP Address (Hex)	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Binary Encoding
0075	14:8	R/W	TX_MARGIN_FULL_1	6:0	0–127	0–127
0075	6:0	R/W	TX_MARGIN_FULL_0	6:0	0–127	0–127
0076	14:8	R/W	TX_MARGIN_FULL_3	6:0	0–127	0–127
0076	6:0	R/W	TX_MARGIN_FULL_2	6:0	0–127	0–127
0077	14:8	R/W	TX_MARGIN_LOW_0	6:0	0–127	0–127
0077	6:0	R/W	TX_MARGIN_FULL_4	6:0	0–127	0–127
0078	14:8	R/W	TX_MARGIN_LOW_2	6:0	0–127	0–127
0078	6:0	R/W	TX_MARGIN_LOW_1	6:0	0–127	0–127
0079	14:8	R/W	TX_MARGIN_LOW_4	6:0	0–127	0–127
0079	6:0	R/W	TX_MARGIN_LOW_3	6:0	0–127	0–127
007A	13:8	R/W	TX_DEEMPH1	5:0	0–63	0–63
007A	5:0	R/W	TX_DEEMPH0	5:0	0–63	0–63
007B	15:0	R/W	TX_RXDETECT_PRECHARGE_TIME	15:0	0–65535	0–65535
007C	15	R/W	TX_RXDETECT_PRECHARGE_TIME	16	0–1	0–1
007C	13:11	R/W	RX_DFE_KL_LPM_KL_CFG1	2:0	0–7	0–7
007C	10:8	R/W	TX_RXDETECT_REF	2:0	0–7	0–7
007C	7:4	R/W	RX_DFE_KL_LPM_KL_CFG2	3:0	0–15	0–15
007C	3	R/W	TX_MAINCURSOR_SEL	0	0–1	0–1
007C	1:0	R/W	PMA_RSV3	1:0	0–3	0–3
007D	13:0	R/W	TX_RXDETECT_CFG	13:0	0–16383	0–16383
007E	14:13	R/W	RX_DFE_KL_LPM_KL_CFG0	1:0	0–3	0–3
007E	12:9	R/W	RX_DFE_KL_LPM_KH_CFG2	3:0	0–15	0–15
007E	8:6	R/W	RX_DFE_KL_LPM_KH_CFG1	2:0	0–7	0–7
007E	5:4	R/W	RX_DFE_KL_LPM_KH_CFG0	1:0	0–3	0–3
007E	3:0	R/W	RX_CM_TRIM	3:0	0–15	0–15
007F	15:0	R/W	CPLL_CFG	15:0	0–65535	0–65535
0080	13	R/W	RX_DFELPM_KLKH_AGC_STUP_EN	0	0–1	0–1
0080	12:0	R/W	CPLL_CFG	28:16	0–8191	0–8191
0081	7	R/W	RX_DFE_KL_LPM_KH_OVRDEN	0	0–1	0–1
0081	6	R/W	RX_DFE_KL_LPM_KL_OVRDEN	0	0–1	0–1
0081	5	R/W	RX_DFE_AGC_OVRDEN	0	0–1	0–1
0081	4	R/W	RX_DFELPM_CFG1	0	0–1	0–1
0081	3:0	R/W	RX_DFELPM_CFG0	3:0	0–15	0–15
0082	15:0	R/W	PMA_RSV2	15:0	0–65535	0–65535
0083	15:0	R/W	PMA_RSV2	31:16	0–65535	0–65535
0084	10	R/W	RESET_POWERSAVE_DISABLE	0	0–1	0–1

Table D-4: **DRP Map of GTH2_CHANNEL Primitive *(Cont'd)***

DRP Address (Hex)	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Binary Encoding
0086	15:0	R/W	DMONITOR_CFG	15:0	0–65535	0–65535
0087	7:0	R/W	DMONITOR_CFG	23:16	0–255	0–255
0088	6:4	R/W	TXOUT_DIV	2:0	1	0
					2	1
					4	2
					8	3
					16	4
0088	2:0	R/W	RXOUT_DIV	2:0	1	0
					2	1
					4	2
					8	3
					16	4
0089	15:0	R/W	CFOK_CFG	15:0	0–65535	0–65535
008A	15:0	R/W	CFOK_CFG	31:16	0–65535	0–65535
008B	15:10	R/W	CFOK_CFG3	5:0	0–63	0–63
008B	9:0	R/W	CFOK_CFG	41:32	0–1023	0–1023
008C	10:0	R/W	RX_DFE_H6_CFG	10:0	0–2047	0–2047
008D	10:0	R/W	RX_DFE_H7_CFG	10:0	0–2047	0–2047
008E	15:0	R/W	RX_DFE_ST_CFG	15:0	0–65535	0–65535
008F	15:0	R/W	RX_DFE_ST_CFG	31:16	0–65535	0–65535
0090	15:0	R/W	RX_DFE_ST_CFG	47:32	0–65535	0–65535
0091	15	R/W	ES_CLK_PHASE_SEL	0	0–1	0–1
0091	14	R/W	USE_PCS_CLK_PHASE_SEL	0	0–1	0–1
0091	11:6	R/W	CFOK_CFG2	5:0	0–63	0–63
0091	5:0	R/W	RX_DFE_ST_CFG	53:48	0–63	0–63
0092	15:0	R/W	ADAPT_CFG0	15:0	0–65535	0–65535
0093	3:0	R/W	ADAPT_CFG0	19:16	0–15	0–15
0094	14:0	R/W	PMA_RSV4	14:0	0–32767	0–32767
0095	7:0	R/W	TXPI_PPM_CFG	7:0	0–255	0–255
0096	5	R/W	TXPI_GREY_SEL	0	0–1	0–1
0096	4	R/W	TXPI_INVSTROBE_SEL	0	0–1	0–1
0096	3	R/W	TXPI_PPMCLK_SEL	0	TXUSRCLK	0
					TXUSRCLK2	1
0096	2:0	R/W	TXPI_SYNFREQ_PPM	2:0	1–7	1–7
0097	15:0	R/W	TST_RSV	15:0	0–65535	0–65535
0098	15:0	R/W	TST_RSV	31:16	0–65535	0–65535
0099	15:0	R/W	PMA_RSV	15:0	0–65535	0–65535
009A	15:0	R/W	PMA_RSV	31:16	0–65535	0–65535
009B	5:0	R/W	RX_BUFFER_CFG	5:0	0–63	0–63
009C	13:8	R/W	RXBUF_THRESH_OVFLW	5:0	0–63	0–63

Table D-4: **DRP Map of GTH2_CHANNEL Primitive *(Cont'd)***

DRP Address (Hex)	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Binary Encoding
009C	5:0	R/W	RXBUF_THRESH_UNDFLW	5:0	0–63	0–63
009D	15:12	R/W	RXBUF_EIDLE_HI_CNT	3:0	0–15	0–15
009D	11:8	R/W	RXBUF_EIDLE_LO_CNT	3:0	0–15	0–15
009D	7	R/W	RXBUF_ADDR_MODE	0	FULL	0
					FAST	1
009D	6	R/W	RXBUF_RESET_ON_EIDLE	0	FALSE	0
					TRUE	1
009D	5	R/W	RXBUF_RESET_ON_CB_ CHANGE	0	FALSE	0
					TRUE	1
009D	4	R/W	RXBUF_RESET_ON_RATE_ CHANGE	0	FALSE	0
					TRUE	1
009D	3	R/W	RXBUF_RESET_ON_ COMMAALIGN	0	FALSE	0
					TRUE	1
009D	2	R/W	RXBUF_THRESH_OVRD	0	FALSE	0
					TRUE	1
009D	1	R/W	RXBUF_EN	0	FALSE	0
					TRUE	1
009D	0	R/W	RX_DEFER_RESET_BUF_EN	0	FALSE	0
					TRUE	1
009F	8:0	R/W	TXDLY_LCFG	8:0	0–511	0–511
00A0	8:0	R/W	RXDLY_LCFG	8:0	0–511	0–511
00A1	15:0	R/W	RXPH_CFG	15:0	0–65535	0–65535
00A2	7:0	R/W	RXPH_CFG	23:16	0–255	0–255
00A3	15:0	R/W	RXPHDLY_CFG	15:0	0–65535	0–65535
00A4	7:0	R/W	RXPHDLY_CFG	23:16	0–255	0–255
00A5	13:0	R/W	RX_DEBUG_CFG	13:0	0–16383	0–16383
00A6	9:0	R/W	ES_PMA_CFG	9:0	0–1023	0–1023
00A7	13	R/W	RXCDR_PH_RESET_ON_EIDLE	0	0–1	0–1
00A7	12	R/W	RXCDR_FR_RESET_ON_EIDLE	0	0–1	0–1
00A7	11	R/W	RXCDR_HOLD_DURING_ EIDLE	0	0–1	0–1
00A7	5:0	R/W	RXCDR_LOCK_CFG	5:0	0–63	0–63
00A8	15:0	R/W	RXCDR_CFG	15:0	0–65535	0–65535
00A9	15:0	R/W	RXCDR_CFG	31:16	0–65535	0–65535
00AA	15:0	R/W	RXCDR_CFG	47:32	0–65535	0–65535
00AB	15:0	R/W	RXCDR_CFG	63:48	0–65535	0–65535
00AC	15:0	R/W	RXCDR_CFG	79:64	0–65535	0–65535
00AD	2:0	R/W	RXCDR_CFG	82:80	0–7	0–7