Virtex-6 FPGA GTX Transceivers

User Guide

UG366 (v2.6) July 27, 2011

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.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
06/24/09	1.0	Initial Xilinx release.
08/11/09	2.0	Chapter 2: • Added new sections: Using TXOUTCLK to Drive the GTX Transceiver TX, page 131, GTX Transceiver TX Reset in Response to Completion of Configuration, page 139, GTX Transceiver TX Reset in Response to GTXTXRESET Pulse, page 140, GTX Transceiver TX Component-Level Resets, page 140, After Power-up and Configuration, page 142, After Turning on a Reference Clock to the TX PLL, page 142, After Changing the Reference Clock to the TX PLL, page 142, After Assertion/Deassertion of TXPOWERDOWN, page 142, TX Rate Change with the TX Buffer Enabled, page 142, TX Rate Change with the TX Buffer Bypassed, page 143, TX Parallel Clock Source Reset, page 143, TX Phase Alignment after Rate Change Use Mode, page 160, and Rate Change Use Mode for PCI Express 2.0 Operation, page 172. • Added the RXPLLREFSELDY[2:0] port to Table 2-4, page 106. • Replaced first sentence of Single External Reference Clock Use Model, page 108. • Added new section Multiple External Reference Clocks Use Model, page 110. • Revised PLL nominal operating range and added Table 2-6, page 113. • Added the PMA_COM_CFG attribute to Table 2-9, page 115. • Replaced Table 2-10, page 117. • Added PCI Express mode power conditions to bulleted list in Power-Down Features for PCI Express Operation, page 123. • Added note 1 to Table 2-10, page 117 on P1 and P2 power state support. • In Dynamic Reconfiguration Port, page 125, revised occurrences of DO to DRPDO. • In Table 2-18, page 126, changed the bus width of the DRP address bus to DADDR[7:0]. Chapter 3: • Renamed TX Clock Divider Control block to TX Fabric Clock Output Control. • Revised "GTX Lanes in Channel" values for 2-byte and 4-byte rows in Table 3-3, page 129. • In the Functional Description of TX Initialization, page 136, revised #2 and added #3. Added Figure 3-8, page 137 showing the GTX TX reset hierarchy. • Revised the GTXTEST[12:0] and GTXTXRESET descriptions in Table 3-7, page 138. • Revised Ease of Use and TX Lane-to-Lane Deskew rows in Table 3-15, page 154. • Revised the TXDLYALIGNDISABLE, TXDLYALIGNMONITOR[7:0], and TXOUTCLK descriptions in Table 3-18, page 156. • Revised steps 2, 5, and 9 in Using the TX Phase-Alignment Circuit to Bypass the Buffer, page 159. • Changed the width of TXDLYALIGNRESET in Figure 3-21, page 160 to 16 TXUSRCLK2 cycles and revised caption. • Revised paragraph under Figure 3-23, page 161 on making phase alignment effective. • In Serial Clock Divider, page 169, provided more details on using the D divider in fixed line rate and multiple line rate applications. • In Table 3-28, page 169, removed TXPLL_DIVSEL_OUT = Ignored from all rows in the Dynamic Control via Ports column. • In Table 3-29, page 170, added the GTXTEST[1] port and revised the clock domain and description of TXRATEDONE. • In Table 3-30, page 171, revised the description of TRANS_TIME_RATE. • Revised PCI Express Clocking Use Mode, page 171 and added Figure 3-29, page 172 and Figure 3-30, page 173.

Date	Version	Revision
08/11/09 (*Cont'd*)	2.0	Chapter 3 (*Cont'd*): • Changed the widths of TXPREEMPHASIS, TXDIFFCTRL, and TXPOSTEMPHASIS in Figure 3-31, page 174. • Revised description of RXPOWERDOWN and TXPOWERDOWN in Table 3-33, page 180. • Added note to the Functional Description of TX Out-of-Band Signaling, page 181. • In Table 3-34, page 182, changed TXELECIDLE to one bit and added COMFINISH. • Updated descriptions of TXELECIDLE and TXPOWERDOWN ports in Table 3-34, page 182 Chapter 4: • Added new sections GTX Transceiver RX Reset in Response to Completion of Configuration, page 263, GTX Transceiver RX Reset in Response to GTXRXRESET Pulse, page 263, Link Idle Reset Support, page 263, GTX Transceiver RX Component-Level Resets, page 264, After Power-up and Configuration, page 266, After Turning on a Reference Clock to RX PLL, page 266, After Changing the Reference Clock to RX PLL, page 267, After Assertion/Deassertion of RXPOWERDOWN, page 267, RX Rate Change with RX Elastic Buffer Enabled, page 267, RX Rate Change with RX Elastic Buffer Bypassed, page 267, RX Parallel Clock Source Reset, page 267, After Remote Power-Up, page 267, Electrical Idle Reset, page 267, After Connecting RXN/RXP, page 268, After an RX Elastic Buffer Error, page 268, Before Channel Bonding, page 268, After Changing Channel Bonding Mode on the Fly, page 268, After a PRBS Error, page 268, After an Oversampler Error, page 269, and After Comma Realignment, page 269. • Added ESD Diodes label to Figure 4-2, page 184, Figure 4-3, page 186, Figure 4-4, page 187, Figure 4-5, page 188, Figure 4-6, page 189, and Figure 4-7, page 190. • Revised captions for Figure 4-9, page 193 and Figure 4-10, page 194. • In Table 4-2, page 185, added sentence about system evaluation purposes to the descriptions of TERMINATION_CTRL[4:0] and TERMINATION_OVRD. • Added GATERXELECIDLE and IGNORESIGDET ports to Table 4-9, page 191. • Added Figure 4-8, page 192. • In Serial Clock Divider, page 207, provided more details on using the D divider in fixed line rate and multiple line rate applications. • In Table 4-23, page 207, removed RXPLL_DIVSEL_OUT = Ignored from all rows in the Dynamic Control via Ports column. • In Table 4-24, page 208, revised the clock domain and description of RXRATEDONE. • In Table 4-25, page 208, revised the description of TRANS_TIME_RATE. • Added RX decoder port and attribute tables (Table 4-38, page 229 and Table 4-39, page 230, respectively). • Changed description of RXDLYALIGNMONITOR[7:0] to reserved in Table 4-40, page 232. • Moved description of RX CDR lock to RX CDR, page 203. • Revised descriptions of CLK_COR_ADJ_LEN, CLK_COR_DET_LEN, CLK_COR_MAX_LAT, and CLK_CORRECT_USE attributes in Table 4-47, page 241. • In the Functional Description of RX Initialization, page 260, revised #2 and added #3. Added Figure 4-49, page 260 showing the GTX receiver reset hierarchy. • In Table 4-52, page 261, revised the GTXTEST[12:0] description and added the PRBSCNTRESET port. • Added the RX_EN_REALIGN_RESET_BUF2 attribute to Table 4-53, page 261. • Revised "GTX Lanes in Channel" values for 2-byte and 4-byte rows in Table 4-58, page 270. Appendix B: • Added new appendix.

Date	Version	Revision
01/19/10	2.1	Updated width of TXBUFSTATUS port in Table 1-1. Updated Figure 1-4. Updated description of SIM_GTXRESET_SPEEDUP in Table 1-2. Added GTXE1_X0Y1 location for LX75T to Figure 1-9. Added new section Reference Clock Input Structure, page 101. Added note after Figure 2-4, Figure 2-5, Figure 2-6, and Figure 2-7. Updated PLL nominal operation range in Functional Description. Removed Line Rate Range column and added -1 Line Rate Range and -2/-3 Line Rate Range columns to Table 2-6. Added note after Figure 2-9. Added description of N1 divider setting after Table 2-7. Updated entries in and removed REFCLK Max and Min columns from Table 2-10. Removed Power Down Transition Times section. Updated Description column of Table 2-10. Moved Ports and Attributes, page 130 before Using TXOUTCLK to Drive the GTX Transceiver TX, page 131. Updated Using TXOUTCLK to Drive the GTX Transceiver TX, page 131. Added guideline for asynchronous GTXTXRESET pulse width in GTX Transceiver TX Reset in Response to GTXTXRESET Pulse, page 140. Added TXDLYALIGNMONENB and updated descriptions of TXDLYALIGNRESET, TXOUTCLK, and TXPMASETPHASE to Table 3-18. Updated steps 1d and 6 in Using the TX Phase-Alignment Circuit to Bypass the Buffer, page 159. Updated TX Oversampling, page 167. In Table 3-26, removed PMA_RX_CFG, updated description of TX_OVERSAMPLE_MODE, and added TXPLL_DIVSEL_OUT. Added note 5 to Figure 3-28. Updated line rate ranges in Table 3-28. Changed IBUFDS to IBUFDS_GTXE1 in Figure 3-29 and added a note after the figure. Replaced TXPREEMPHASIS with TXPOSTEMPHASIS in description of TXDEEMPH in Table 3-31. Changed PCI Express version from 3.0 to 2.0 in note for Table 3-31. Replaced TXPREEMPHASIS with TXPOSTEMPHASIS in descriptions of TX_DEEMPH_0/1 in Table 3-32. Replaced TXPREEMPHASIS with TXPOSTEMPHASIS in PCIe Mode, page 179 and Customizable User Presets, page 179. Added note after Figure 4-2 and Table 4-3. Updated Table 4-5 and Table 4-7. Added OOBDETECT_THRESHOLD_0/1 to and updated description of SATA_IDLE_VAL in Table 4-10. Updated descriptions of DFETAPOVRD and DFEDLYOVRD ports after Figure 4-12. Updated descriptions of DFECLKDLYADJ, DFECLKDLYADJMON, and DFEDLYOVRD in Table 4-11. Updated descriptions of DFE_CAL_TIME, DFE_CFG, and RX_EN_IDLE_HOLD_DFE attributes in Table 4-12. Renamed RX Clock Divider Control section as RX Fabric Clock Output Control, page 206. Updated MGTREFCLKFAB[1] bit in and added note 4 to Figure 4-15. Updated line rate ranges in Table 4-23. Updated RX Margin Analysis, page 209. Added DFEEYEDACMON port to Table 4-26. Replaced INTDATAWIDTH with RX_DATA_WIDTH in and added note to Figure 4-19. Changed RXOVERSAMPLER to RXOVERSAMPLEERR in Table 4-29. Updated description of RX_OVERSAMPLE_MODE in and added RXPLL_DIVSEL_OUT to Table 4-30. Swapped the order of the SIPO and Polarity Inversion blocks in Figure 4-20. Updated descriptions of RX_PRBS_ERR_CNT and RXPRBSERR_LOOPBACK attributes in Table 4-32. Replaced GTXRESET with GTXRXRESET in Use Models, page 215. Changed PCOMMA_ALIGN and MCOMMA_ALIGN to PCOMMA_DETECT and MCOMMA_DETECT, respectively, in Alignment Status Signals, page 218 and Table 4-34. Updated RX Buffer Bypass, page 230 with restrictions on RX buffer bypass operation. Updated descriptions of CHAN_BOND_1/2_MAX_SKEW a nd CHAN_BOND_SEQ_LEN attribute in Table 4-49. Added guideline for asynchronous GTXTXRESET pulse width in GTX Transceiver RX Reset in Response to GTXRXRESET Pulse, page 263. Added description of power supply regulators for MGTAVCC and VCCINT in Overview, page 286. In Table B-1, changed attribute encoding 3 in attribute bits 1:0 of DADDRs `7h`, `12h`, and `13h` to Reserved.
02/23/10	2.2	Updated descriptions of RXDLYALIGNOVERRIDE in Table 4-40 and RX_DLYALIGN_OVRDSETTING in Table 4-41. Updated Using the RX Phase Alignment Circuit to Bypass the Buffer, page 234, including Note 2 in Notes for Figure 4-32.. Updated Figure 4-33.

Date	Version	Revision
05/24/10	2.3	Added description of buffer bypass mode to Multiple External Reference Clocks Use Model. Added Power-Down Requirements for TX and RX Buffer Bypass. Added description of TX buffer bypass to Functional Description, page 136 and Functional Description, page 156. Added description of RX buffer bypass to Functional Description, page 230. Updated Functional Description, page 260 with description of buffer bypass mode. Removed GTXTEST[12:0] from Table 4-52. Updated Managing Unused GTX Transceivers. Replaced “group” with “bank” in Table 5-1, Analog Power Supply Pins for Virtex-6 LXT Devices, and Partially Unused Quad Column. Added Note 2 to Table 5-4 and Table 5-5. Added note about buffer bypass mode to Reference Clock Checklist. Added Reference Clock Toggling.
10/01/10	2.4	Updated Functional Description, GTX Transceiver TX Reset in Response to Completion of Configuration, and GTX Transceiver TX Reset in Response to GTXTXRESET Pulse.
01/17/11	2.5	Replaced PMA_COM_CFG with PMA_CFG in Table 2-9. Replaced RXRATE with RXRATE[1:0] in Chapter 4, Receiver. Added note before Table 1-1. Added TXDLYALIGNMONENB, RXDLYALIGNMONENB, PMA_RXSYNC_CFG, TXDRIVE_LOOPBACK_HIZ, and TXDRIVE_LOOPBACK_PD to Table 1-1. In Table 1-1, moved RX_PRBS_ERR_CNT from RX Pattern Checker to Status Registers (Read Only) section. Added FF1154 Package Placement Diagrams, FF1155 Package Placement Diagrams, FF1923 Package Placement Diagrams, and FF1924 Package Placement Diagrams. Updated Figure 2-1. Added RX_CLK25_DIVIDER and TX_CLK25_DIVIDER to Table 2-9 and Table 1-1. Updated description of TXPDOWNASYNCH in Table 2-11. Added BGTEST_CFG, BIAS_CFG, and PMA_TX_CFG to Table 2-12 and Table 1-1. Added Table 2-17. Added ACJTAG. Updated Table 3-10. Updated descriptions of TXDIFFCTRL[3:0], TXPDOWNASYNCH, TXPOSTEMPHASIS[4:0], and TXPREEMPHASIS[3:0] in Table 3-31. Updated description of TXPOWERDOWN[1:0] in Table 3-34. Updated description of IGNORESIGDET and changed direction of RXVALID from In to Out in Table 4-9. Updated OOBDETECT_THRESHOLD attribute in Table 4-10. Added Use Mode – Fixed Tap Mode and Use Mode – Auto-To-Fix and Use Mode – Auto. Updated descriptions of PMA_RX_CFG, RX_EN_IDLE_HOLD_CDR, RX_EN_IDLE_RESET_FR, and RX_EN_IDLE_RESET_PH in Table 4-22. Updated Eye Outline Scan Mode. Updated description of RX_EYE_OFFSET in Table 4-27. Updated description of PMA_RX_CFG in Table 4-30. Updated Figure 4-26. Added Manual Alignment, including Figure 4-27. Removed RX_PRBS_ERR_CNT from Table 4-32 and added it to Table 4-33. Added RXSLIDE to Table 4-34 and Table 1-1. Updated description of ALIGN_COMMA_WORD, and added MCOMMA_10B_VALUE, MCOMMA_DETECT, PCOMMA_10B_VALUE, PCOMMA_DETECT, SHOW_REALIGN_COMMA, RX_SLIDE_MODE, and RX_SLIDE_AUTO_WAIT to Table 4-35 and Table 1-1. Updated Figure 4-28 and description of RX_LOS_THRESHOLD in Functional Description, page 225. Changed RX_DATA_WIDTH attribute type in Table 4-39. Added RXDLYALIGNMONENB to Table 4-40. Added PMA_RXSYNC_CFG to Table 4-41. Changed direction of RXDATA[31:0] port from In to Out in Table 4-40. Updated Common Package Power Plane Prioritization. Added Hot Swapping Devices. Updated `2Ah` and `47h` rows of Table B-1. Added Table B-2. Added Appendix C, Low Latency Design.

Date	Version	Revision
01/17/11 *(Cont'd)*	2.5	Updated POWER_SAVE description in Table 2-12, Table 3-19, and Table 4-41. Updated embedded table title of TXPOSTEMPHASIS[4:0] port in Table 3-31. Updated description of "From TX Parallel Data" in Figure 4-1 and Figure C-3. Updated Table 4-2. Updated Use Mode – Auto-To-Fix and Use Mode – Auto. Updated Using the RX Phase Alignment Circuit to Bypass the Buffer. Updated Figure 4-32 and Figure 4-34. Updated DADDR in Table B-1. Per XCN11009: *Virtex-6: Data-Sheet, User Guides and JTAG ID Updates*: Updated TX Buffer Bypass: TX delay aligner bypassed, additional requirements on interconnect logic clocking use model; Updated RX Buffer Bypass: RX delay aligner bypassed for lower line rates, higher line rate support is an advanced feature.
07/27/11	2.6	Chapter 1, Transceiver and Tool Overview, added TX_IDLE_DEASSERT_DELAY and TX_IDLE_DEASSERT_DELAY to Table 1-1, updated SIM_GTXRESET_SPEEDUP description in Table 1-2. Replaced FF1154 package placement diagrams Figure 1-24 through Figure 1-35, FF1155 Figure 1-36 through Figure 1-41, FF1923 Figure 1-42 through Figure 1-51, and FF1924 Figure 1-42 through Figure 1-51. Chapter 2, Shared Transceiver Features, updated ODIV2 description in Table 2-1, TXPLLLKDET/RXPLLLKDET description in Table 2-8, and GigE standard in Table 2-10. Clarified far-end PMA loopback and far-end PMA loopback in Functional Description, page 123. Chapter 3, Transmitter, renamed Figure 3-9 and added paragraph below its notes to clarify TXRESET and RXRESET. Renamed Figure 3-10 and updated notes for Figure 3-10 and Figure 3-11. Updated link to Interlaken in TX Gearbox. Updated Figure 3-28. Updated TXOUTCLK description in Table 3-29. Added TX_IDLE_DEASSERT_DELAY and TX_IDLE_DEASSERT_DELAY to Table 3-35. Chapter 4, Receiver, updated use mode 3 max swing in Table 4-6. Revised descriptions of PMA_RX_CFG, RX_EN_IDLE_HOLD_CDR, RX_EN_IDLE_RESET_FR, and RX_EN_IDLE_RESET_PH in Table 4-22. Updated Figure 4-15. Revised description of RXRECCLK in Table 4-24. Added bus range to RXLOSSOFSYNC in Table 4-36. Updated range of RXDATA[7:0] in 8B/10B Decoder Bit and Byte Order. Updated RX Running Disparity. Updated and added note to Figure 4-30. Added note to RX_EN_IDLE_RESET_BUF and revised RX_FIFO_ADDR_MODE in Table 4-44. Added note to RX_EN_IDLE_HOLD_CDR, RX_EN_IDLE_HOLD_DFE, RX_EN_IDLE_RESET_BUF, RX_EN_IDLE_RESET_PH, and RX_EN_IDLE_RESET_FR in Table 4-53. Added note to Link Idle Reset Support. Revised After Remote Power-Up. Added note to Before Channel Bonding. Chapter 5, Board Design Guidelines, added XC6VSX315T-FF1156 and XC6VSX475T-FF1156 to Table 5-2. Updated title of Analog Power Supply Pins for Virtex-6 LXT Devices and added Analog Power Supply Pins for Virtex-6 HXT Devices. Appendix C, Low Latency Design, updated TX Buffer, PMA + Interface, and Total TX Latency in Table C-1. Updated PMA + Interface and Total RX Latency in Table C-2.

Table of Contents

Preface: About This Guide

Chapter 1: Transceiver and Tool Overview

Chapter 2: Shared Transceiver Features

Chapter 3: Transmitter

Chapter 4: Receiver

Chapter 5: Board Design Guidelines

About This Guide

This document shows how to use the GTX transceivers in Virtex®-6 FPGAs. In this document:

- Virtex-6 FPGA GTX transceiver is abbreviated as *GTX transceiver.*
- *GTXE1* is the name of the instantiation primitive that instantiates one Virtex-6 FPGA GTX transceiver.
- A *Quad* or *Q* is a cluster or set of four GTX transceivers that share two differential reference clock pin pairs and analog supply pins.

Guide Contents

This manual contains the following chapters:

- Chapter 1, Transceiver and Tool Overview
- Chapter 2, Shared Transceiver Features
- Chapter 3, Transmitter
- Chapter 4, Receiver
- Chapter 5, Board Design Guidelines
- Appendix A, 8B/10B Valid Characters
- Appendix B, DRP Address Map of the GTX Transceiver
- Appendix C, Low Latency Design

Additional Documentation

The following documents are also available for download at http://www.xilinx.com/products/virtex6.

- Virtex-6 Family Overview

  The features and product selection of the Virtex-6 family are outlined in this overview.
- Virtex-6 FPGA Data Sheet: DC and Switching Characteristics

  This data sheet contains the DC and Switching Characteristic specifications for the Virtex-6 family.
- Virtex-6 FPGA Packaging and Pinout Specifications

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

- Virtex-6 FPGA Configuration User Guide

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

- Virtex-6 FPGA SelectIO Resources User Guide

  This guide describes the SelectIO™ resources available in all Virtex-6 devices.

- Virtex-6 FPGA Clocking Resources User Guide

  This guide describes the clocking resources available in all Virtex-6 devices, including the MMCM and PLLs.

- Virtex-6 FPGA Memory Resources User Guide

  The functionality of the block RAM and FIFO are described in this user guide.

- Virtex-6 FPGA Configurable Logic Block User Guide

  This guide describes the capabilities of the configurable logic blocks (CLBs) available in all Virtex-6 devices.

- Virtex-6 FPGA DSP48E1 Slice User Guide

  This guide describes the architecture of the DSP48E1 slice in Virtex-6 FPGAs and provides configuration examples.

- Virtex-6 FPGA Embedded Tri-Mode Ethernet MAC User Guide

  This guide describes the dedicated Tri-Mode Ethernet Media Access Controller available in all Virtex-6 FPGAs except the XC6VLX760.

- Virtex-6 FPGA System Monitor User Guide

  The System Monitor functionality available in all Virtex-6 devices is outlined in this guide.

- Virtex-6 FPGA PCB Designer’s Guide

  This guide provides information on PCB design for Virtex-6 FPGA GTX transceivers, with a focus on strategies for making design decisions at the PCB and interface level.

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

The following documents provide additional information useful to this document:

1. *High-Speed Serial I/O Made Simple*

   http://www.xilinx.com/publications/books/serialio/index.htm

Chapter 1

Transceiver and Tool Overview

Overview

The Virtex®-6 FPGA GTX transceiver is a power-efficient transceiver. The GTX transceiver is highly configurable and tightly integrated with the programmable logic resources of the FPGA. It provides the following features to support a wide variety of applications:

- Current Mode Logic (CML) serial drivers/buffers with configurable termination, voltage swing
- Programmable TX pre-emphasis/post-emphasis, RX equalization, and linear and decision feedback equalization (DFE) for optimized signal integrity.
- Line rates from 600 Mb/s to 6.6 Gb/s, with optional 5X digital oversampling required for rates between 480 Mb/s and 600 Mb/s.
- Optional built-in PCS features, such as 8B/10B encoding, comma alignment, channel bonding, and clock correction.
- Fixed latency modes for minimized, deterministic datapath latency.
- Beacon signaling for PCI Express® designs and Out-of-Band signaling including COM signal support for SATA designs.
- RX/TX Gearbox provides header insertion and extraction support for 64B/66B and 64B/67B (Interlaken) protocols.
- Receiver eye scan

  Horizontal eye scan in the time domain for testing purposes

The first-time user is recommended to read *High-Speed Serial I/O Made Simple* [Ref 1], which discusses high-speed serial transceiver technology and its applications. The CORE Generator™ tool includes a Wizard to automatically configure GTX transceivers to support configurations for different protocols or perform custom configuration (see Virtex-6 FPGA GTX Transceiver Wizard). The GTX transceiver offers a data rate range and features that allow physical layer support for various protocols.

Figure 1-1 illustrates a block view of the Virtex-6 FPGA GTX transceiver.

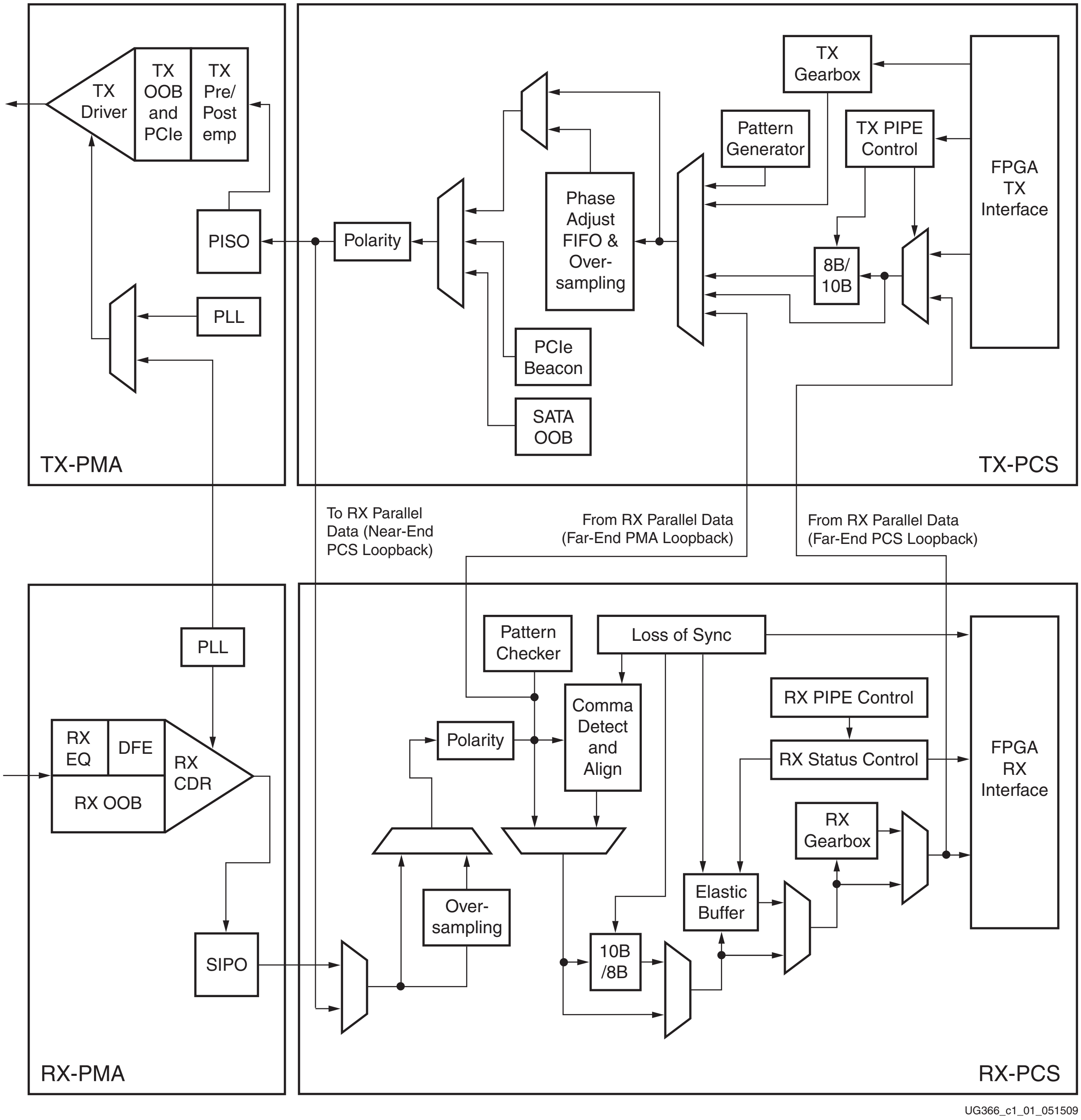


Figure 1-1: **Virtex-6 FPGA GTX Transceiver Simplified Block Diagram**

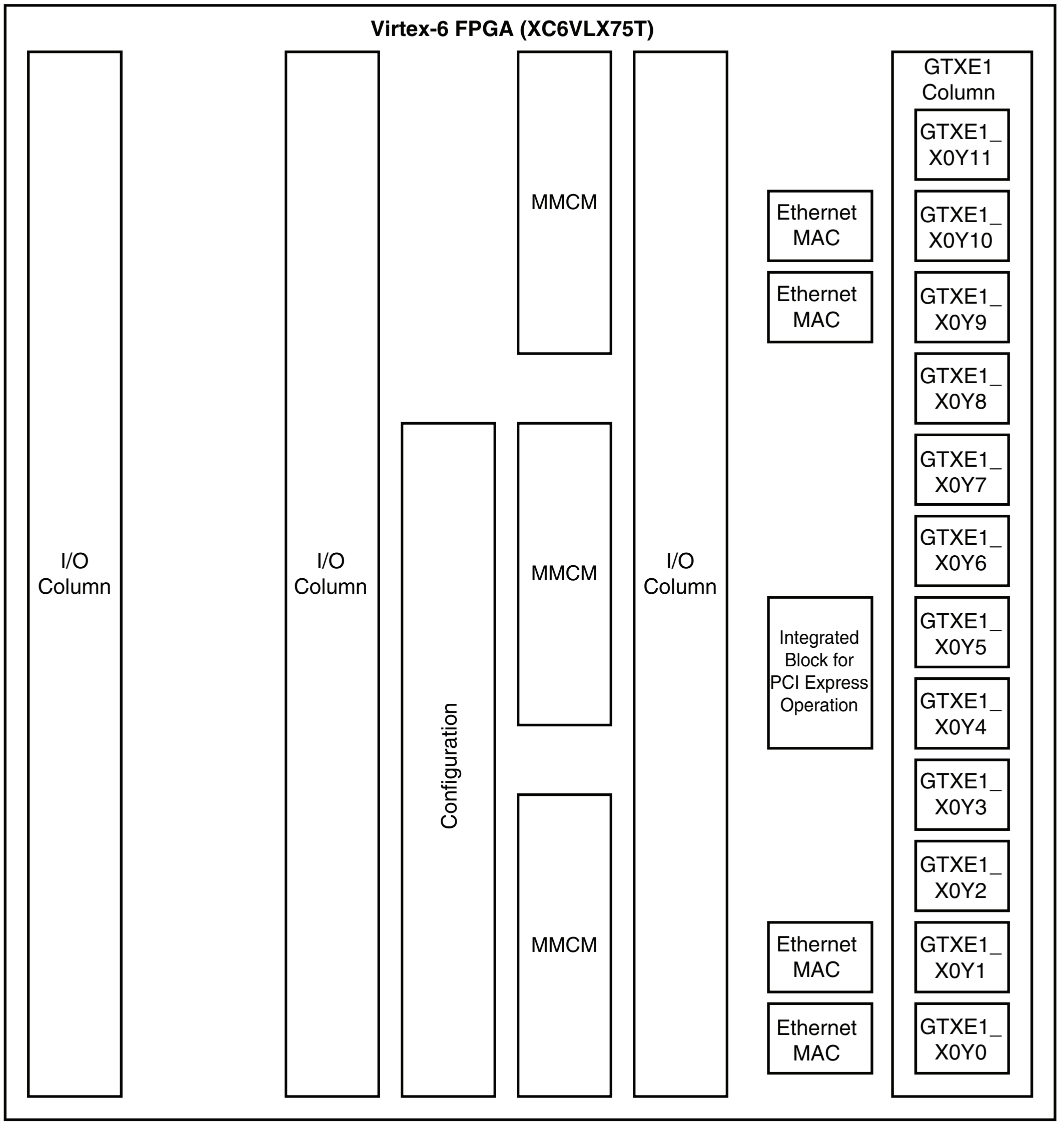
Details about the different functional blocks of the transmitter and receiver including their use models are described in Chapter 3, Transmitter, and Chapter 4, Receiver.

Figure 1-2 shows the GTX transceiver placement in an example Virtex-6 device (XC6VLX75T).

Additional information on the functional blocks in Figure 1-2 is available in the following locations:

- The *Virtex-6 FPGA Configuration User Guide* provides more information on the Configuration and Clock, MMCM, and I/O blocks.
- The *Virtex-6 FPGA Embedded Tri-Mode Ethernet MAC User Guide* provides detailed information on the Ethernet MAC.

Figure 1-2 illustrates the location of the GTX transceiver inside the Virtex-6 XC6VLX75T FPGA.



UG366_c1_02_051509

Figure 1-2: **GTX Transceiver Inside the Virtex-6 XC6VLX75T FPGA**

GTX transceivers are clustered together in a set of four called a *Quad* or *Q*. Figure 1-3 illustrates the clustering of four GTX transceivers to a Quad. Refer to Implementation, page 41 for placement information and the mapping of each transceiver into a specific Quad.

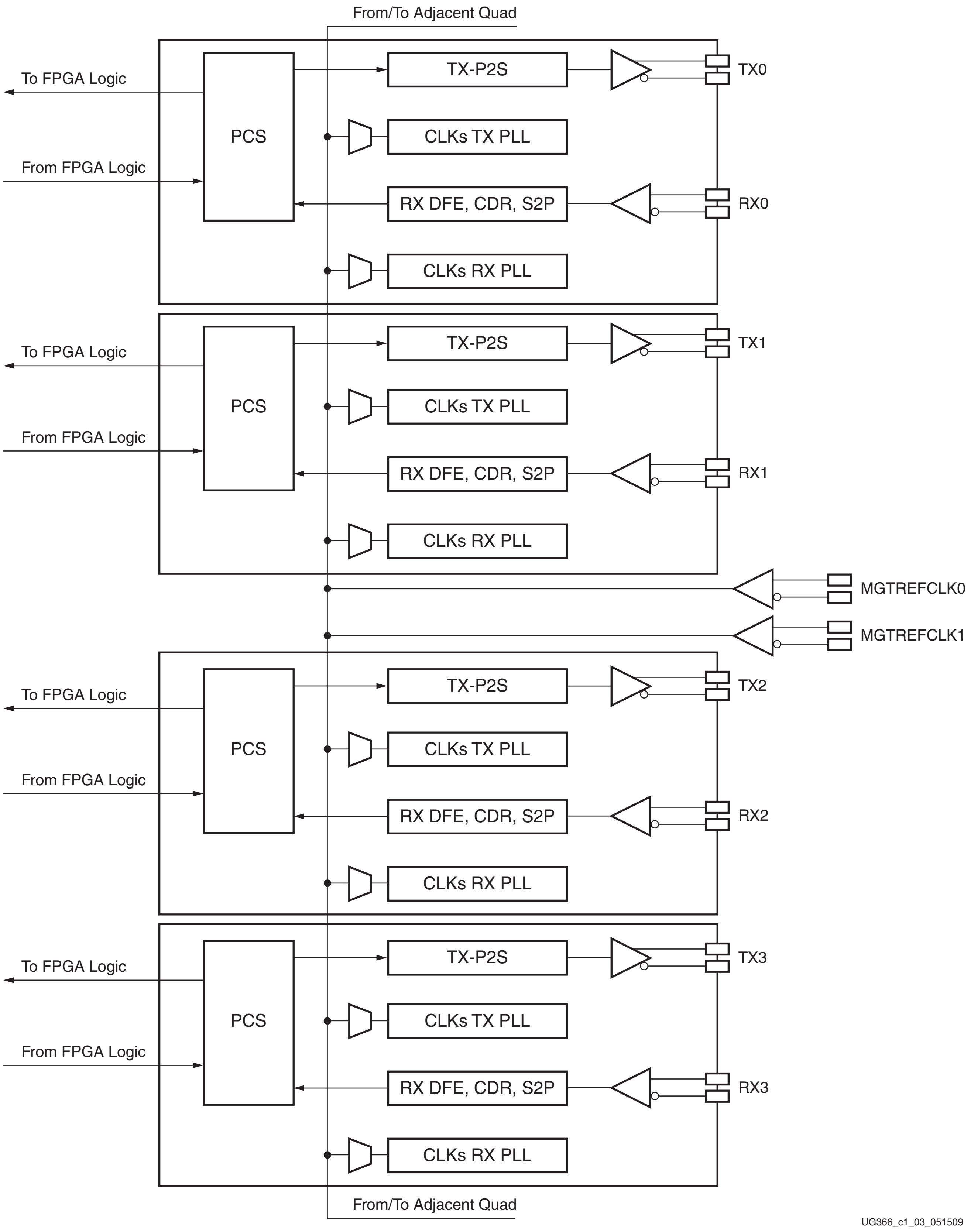


Figure 1-3: **Quad Configuration**

This cluster of four GTX transceivers share two differential reference clock pin pairs and clock routing. Chapter 2, Shared Transceiver Features, discusses details about reference clock sources and the routing.

Port and Attribute Summary

The ports and attributes are grouped in tables for each functionality group (e.g., reference clock selection). If a port or attribute appears in multiple chapters, it is listed in the group of its first appearance. Table 1-1 summarizes the ports and attributes according to functionality group.

Note: Table 1-1 lists all the ports and attributes covered in this user guide. Some ports or attributes are present in the instantiation primitive or are listed in Appendix B, DRP Address Map of the GTX Transceiver but not in Table 1-1.

Table 1-1: **Port and Attribute Summary**

Port/Attribute	Section, Page
Simulation	
Attributes:	
• SIM_GTXRESET_SPEEDUP	page 38
• SIM_RECEIVER_DETECT_PASS	page 38
• SIM_RXREFCLK_SOURCE	page 39
• SIM_TX_ELEC_IDLE_LEVEL	page 39
• SIM_TXREFCLK_SOURCE	page 39
• SIM_VERSION	page 39
Clocking	
Ports:	
• GREFCLKRX	page 106
• GREFCLKTX	page 106
• MGTREFCLKRX[1:0]	page 106
• MGTREFCLKTX[1:0]	page 106
• NORTHREFCLKRX[1:0]	page 106
• NORTHREFCLKTX[1:0]	page 106
• PERFCLKRX	page 106
• PERFCLKTX	page 106
• RXPLLREFSELDY[2:0]	page 107
• SOUTHREFCLKRX[1:0]	page 107
• SOUTHREFCLKTX[1:0]	page 107
• TXPLLREFSELDY[2:0]	page 107
Attributes:	
• PMA_CAS_CLK_EN	page 107
• SIM_RXREFCLK_SOURCE[2:0]	page 108
• SIM_TXREFCLK_SOURCE[2:0]	page 108

Table 1-1: **Port and Attribute Summary *(Cont'd)***

Port/Attribute	Section, Page
PLL	
Ports: • PLLTXRESET • PLLRXRESET • TXPLLLKDET • RXPLLLKDET • TXPLLLKDETEN • RXPLLLKDETEN • TXPLLPOWERDOWN • RXPLLPOWERDOWN	 page 115 page 115 page 115 page 115 page 115 page 115 page 115 page 115
Attributes: • PMA_CFG • TX_CLK_SOURCE • TX_TDCC_CFG • TXPLL_COM_CFG • RXPLL_COM_CFG • TXPLL_CP_CFG • RXPLL_CP_CFG • TXPLL_DIVSEL_FB • RXPLL_DIVSEL_FB • TXPLL_DIVSEL_OUT • RXPLL_DIVSEL_OUT • TXPLL_DIVSEL_REF • RXPLL_DIVSEL_REF • TXPLL_DIVSEL45_FB • RXPLL_DIVSEL45_FB • TXPLL_LKDET_CFG • RXPLL_LKDET_CFG • TXPLL_SATA • RX_CLK25_DIVIDER • TX_CLK25_DIVIDER	 page 115 page 115 page 116 page 116 page 116 page 116 page 116 page 116 page 116 page 116 page 116 page 116 page 116 page 116 page 116 page 116 page 116 page 116 page 117 page 117
Power Down	
Ports: • RXPLLPOWERDOWN • RXPOWERDOWN[1:0] • TXPDOWNASYNCH • TXPLLPOWERDOWN • TXPOWERDOWN[1:0]	 page 120 page 120 page 120 page 120 page 120

Table 1-1: **Port and Attribute Summary *(Cont'd)***

Port/Attribute	Section, Page
Attributes: • BGTEST_CFG • BIAS_CFG • PMA_TX_CFG • POWER_SAVE • TRANS_TIME_FROM_P2 • TRANS_TIME_NON_P2 • TRANS_TIME_RATE • TRANS_TIME_TO_P2	 page 121 page 121 page 121 page 121 page 121 page 121 page 121 page 121
Loopback	
Ports: • LOOPBACK[2:0] Attributes: • TXDRIVE_LOOPBACK_HIZ • TXDRIVE_LOOPBACK_PD	 page 125 page 125 page 125
DRP	
Ports: • DADDR[7:0] • DCLK • DEN • DI[15:0] • DRPDO[15:0] • DRDY • DWE	 page 126 page 126 page 126 page 126 page 126 page 126 page 126
FPGA TX Interface	
Ports: • MGTREFCLKFAB[1:0] • TXCHARDISPMODE[3:0] • TXCHARDISPVAL[3:0] • TXDATA[31:0] • TXUSRCLK • TXUSRCLK2	 page 130 page 130 page 130 page 130 page 130 page 130
Attributes: • GEN_TXUSRCLK • TX_DATA_WIDTH	 page 131 page 131

Table 1-1: **Port and Attribute Summary *(Cont'd)***

Port/Attribute	Section, Page
TX Initialization	
Ports:	
• GTXTEST[12:0]	page 138
• GTXTXRESET	page 138
• PLLTXRESET	page 138
• TSTIN[19:0]	page 138
• TXDLYALIGNRESET	page 138
• TXRESET	page 139
• TXRESETDONE	page 139
Attributes:	
• TX_EN_RATE_RESET_BUF	page 139
TX Encoder	
Ports:	
• TXBYPASS8B10B[3:0]	page 145
• TXCHARDISPMODE[3:0]	page 145
• TXCHARDISPVAL[3:0]	page 145
• TXCHARISK[3:0]	page 146
• TXENC8B10BUSE	page 146
• TXKERR[3:0]	page 146
• TXRUNDISP[3:0]	page 146
TX Gearbox	
Ports:	
• TXGEARBOXREADY	page 147
• TXHEADER[2:0]	page 147
• TXSEQUENCE[6:0]	page 147
• TXSTARTSEQ	page 147
Attributes:	
• GEARBOX_ENDEC	page 147
• TXGEARBOX_USE	page 147
TX Buffer	
Ports:	
• TXBUFSTATUS[1:0]	page 155
• TXRESET	page 155
Attributes:	
• TX_BUFFER_USE	page 155
• TX_OVERSAMPLE_MODE	page 155
TX Buffer Bypass	

Table 1-1: **Port and Attribute Summary *(Cont'd)***

Port/Attribute	Section, Page
Ports:	
• TXDLYALIGNDISABLE	page 156
• TXDLYALIGNMONENB	page 156
• TXDLYALIGNMONITOR[7:0]	page 156
• TXDLYALIGNOVERRIDE	page 157
• TXDLYALIGNRESET	page 157
• TXDLYALIGNUPDSW	page 157
• TXENPMAPHASEALIGN	page 157
• TXOUTCLK	page 157
• TXPLLLKDET	page 157
• TXPLLLKDETEN	page 157
• TXPMASETPHASE	page 157
• TXUSRCLK	page 157
Attributes:	
• TX_BUFFER_USE	page 158
• TX_BYTECLK_CFG[5:0]	page 158
• TX_DATA_WIDTH	page 158
• TX_DLYALIGN_CTRINC	page 158
• TX_DLYALIGN_LPFINC	page 158
• TX_DLYALIGN_MONSEL	page 158
• TX_DLYALIGN_OVRDSETTING	page 158
• TX_PMADATA_OPT	page 158
• TX_XCLK_SEL	page 159
• TXOUTCLK_CTRL	page 159
TX Pattern Generator	
Ports:	
• TXENPRBSTST[2:0]	page 165
• TXPRBSFORCEERR	page 165
Attributes:	
• RXPRBSERR_LOOPBACK	page 165
TX Oversampling	
Attributes:	
• TX_OVERSAMPLE_MODE	page 167
TX Polarity Control	
Ports:	
• TXPOLARITY	page 167
TX Fabric Clock Output Control	

Table 1-1: **Port and Attribute Summary *(Cont'd)***

Port/Attribute	Section, Page
Ports: • GTXTEST[1] • MGTREFCLKFAB[0] • O • ODIV2 • PHYSTATUS • TXOUTCLK • TXOUTCLKPCS • TXRATE • TXRATEDONE	 page 170 page 170 page 170 page 170 page 170 page 170 page 170 page 171 page 171
Attributes: • TRANS_TIME_RATE • TX_EN_RATE_RESET_BUF • TXOUTCLK_CTRL • TXPLL_DIVSEL_OUT	 page 171 page 171 page 171 page 171
TX Configurable Driver	
Ports: • TXBUFDIFFCTRL[2:0] • TXDEEMPH • TXDIFFCTRL[3:0] • TXELECIDLE • TXINHIBIT • TXMARGIN[2:0] • TXPDOWNASYNCH • TXPOSTEMPHASIS[4:0] • TXPREEMPHASIS[3:0] • TXP TXN • TXSWING	 page 174 page 174 page 175 page 175 page 175 page 175 page 176 page 176 page 177 page 177 page 177
Attributes: • TX_DEEMPH_0[4:0] • TX_DEEMPH_1[4:0] • TX_DRIVE_MODE • TX_MARGIN_FULL_0[6:0] • TX_MARGIN_FULL_1[6:0] • TX_MARGIN_FULL_2[6:0] • TX_MARGIN_FULL_3[6:0] • TX_MARGIN_FULL_4[6:0] • TX_MARGIN_LOW_0[6:0] • TX_MARGIN_LOW_1[6:0] • TX_MARGIN_LOW_2[6:0] • TX_MARGIN_LOW_3[6:0] • TX_MARGIN_LOW_4[6:0]	 page 177 page 177 page 178 page 178 page 178 page 178 page 178 page 178 page 178 page 178 page 179 page 179 page 179

Table 1-1: **Port and Attribute Summary *(Cont'd)***

Port/Attribute	Section, Page
TX Receiver Detect Support for PCI Express Designs	
Ports:	
• PHYSTATUS	page 180
• RXPOWERDOWN[1:0]	page 181
• TXPOWERDOWN[1:0]	page 181
• RXSTATUS[2:0]	page 181
• TXDETECTRX	page 181
TX OOB	
Ports:	
• COMFINISH	page 182
• TXCOMINIT	page 182
• TXCOMSAS	page 182
• TXCOMWAKE	page 182
• TXELECIDLE	page 182
• TXPOWERDOWN[1:0]	page 182
Attributes:	
• COM_BURST_VAL	page 182
• TXPLL_SATA	page 182
• TX_IDLE_DEASSERT_DELAY	page 182
• TX_IDLE_ASSERT_DELAY	page 182
RX AFE	
Ports:	
• RXN	page 185
• RXP	page 185
Attributes:	
• AC_CAP_DIS	page 185
• CM_TRIM[1:0]	page 185
• RCV_TERM_GND	page 185
• RCV_TERM_VTTRX	page 185
• TERMINATION_CTRL[4:0]	page 185
• TERMINATION_OVRD	page 185
RX OOB	
Ports:	
• COMINITDET	page 191
• COMSASDET	page 191
• COMWAKEDET	page 191
• GATERXELECIDLE	page 191
• IGNORESIGDET	page 191
• RXELECIDLE	page 191
• RXSTATUS[2:0]	page 192
• RXVALID	page 192

Table 1-1: **Port and Attribute Summary *(Cont'd)***

Port/Attribute	Section, Page
Attributes: • SAS_MAX_COMSAS • SAS_MIN_COMSAS • SATA_BURST_VAL • SATA_IDLE_VAL • SATA_MAX_BURST • SATA_MAX_INIT • SATA_MAX_WAKE • SATA_MIN_BURST • SATA_MIN_INIT • SATA_MIN_WAKE	 page 192 page 192 page 192 page 192 page 192 page 192 page 192 page 192 page 193 page 193
RX Equalizer	
Ports: • DFECLKDLYADJ[5:0] • DFECLKDLYADJMON[5:0] • DFEDLYOVRD • DFEEYEDACMON[4:0] • DFESENSCAL[2:0] • DFETAP1[4:0] • DFETAP1MONITOR[4:0] • DFETAP2[4:0] • DFETAP2MONITOR[4:0] • DFETAP3[3:0] • DFETAP3MONITOR[3:0] • DFETAP4[3:0] • DFETAP4MONITOR[3:0] • DFETAPOVRD • RXEQMIX[9:0]	 page 196 page 196 page 196 page 196 page 196 page 196 page 196 page 196 page 196 page 196 page 197 page 197 page 197 page 197 page 197
Attributes: • DFE_CAL_TIME[4:0] • DFE_CFG[7:0] • RX_EN_IDLE_HOLD_DFE	 page 197 page 197 page 197
RX CDR	
Ports: • RXCDRRESET • RXRATE[1:0]	 page 204 page 204

Table 1-1: **Port and Attribute Summary *(Cont'd)***

Port/Attribute	Section, Page
Attributes: • CDR_PH_ADJ_TIME • PMA_CDR_SCAN • PMA_RX_CFG • RX_EN_IDLE_HOLD_CDR • RX_EN_IDLE_RESET_FR • RX_EN_IDLE_RESET_PH • RX_EYE_SCANMODE • RXPLL_DIVSEL_OUT	page 204 page 204 page 204 page 204 page 205 page 205 page 205 page 205
RX Clock Divider Control	
Ports: • MGTREFCLKFAB[1] • O • ODIV2 • PHYSTATUS • RXRATE[1:0] • RXRATEDONE • RXRECCLK • RXRECCLKPCS	page 208 page 208 page 208 page 208 page 208 page 208 page 208 page 208
Attributes: • RX_EN_RATE_RESET_BUF • RXPLL_DIVSEL_OUT • RXRECCLK_CTRL • TRANS_TIME_RATE	page 208 page 208 page 209 page 209
RX Margin Analysis	
Ports: • RXDATA[31:0]	page 211
Attributes: • RX_EYE_OFFSET • RX_EYE_SCANMODE	page 212 page 212
RX Polarity Control	
Ports: • RXPOLARITY	page 212
RX Oversampling	
Ports: • RXENSAMPLEALIGN • RXOVERSAMPLEERR	page 214 page 214
Attributes: • PMA_RX_CFG • RX_OVERSAMPLE_MODE	page 214 page 214

Table 1-1: **Port and Attribute Summary *(Cont'd)***

Port/Attribute	Section, Page
RX Pattern Checker	
Ports:	
• PRBSCNTRESET	page 215
• RXENPRBSTST[2:0]	page 215
• RXPRBSERR	page 215
Attributes:	
• RXPRBSERR_LOOPBACK	page 215
Status Registers (Read Only):	
• RX_PRBS_ERR_CNT	page 215
RX Byte and Word Alignment	
Ports:	
• RXBYTEISALIGNED	page 221
• RXBYTEREALIGN	page 221
• RXCOMMADET	page 221
• RXCOMMADETUSE	page 221
• RXENMCOMMAALIGN	page 221
• RXENPCOMMAALIGN	page 221
• RXSLIDE	page 222
Attributes:	
• ALIGN_COMMA_WORD	page 222
• COMMA_10B_ENABLE	page 222
• COMMA_DOUBLE	page 223
• MCOMMA_10B_VALUE	page 223
• MCOMMA_DETECT	page 223
• PCOMMA_10B_VALUE	page 223
• PCOMMA_DETECT	page 223
• SHOW_REALIGN_COMMA	page 223
• RX_SLIDE_MODE	page 224
• RX_SLIDE_AUTO_WAIT	page 224
RX Loss-of-Sync State Machine	
Ports:	
• RXLOSSOFSYNC[1:0]	page 226
Attributes:	
• RX_LOS_INVALID_INCR	page 226
• RX_LOS_THRESHOLD	page 226
• RX_LOSS_OF_SYNC_FSM	page 226
RX 8B/10B Decoder	

Table 1-1: **Port and Attribute Summary *(Cont'd)***

Port/Attribute	Section, Page
Ports: • RXCHARISCOMMA[3:0] • RXCHARISK[3:0] • RXDEC8B10BUSE • RXDISPERR[3:0] • RXNOTINTABLE[3:0] • RXRUNDISP[3:0]	page 229 page 229 page 229 page 229 page 229 page 229
Attributes: • DEC_MCOMMA_DETECT • DEC_PCOMMA_DETECT • DEC_VALID_COMMA_ONLY • RX_DATA_WIDTH • RX_DECODE_SEQ_MATCH	page 230 page 230 page 230 page 230 page 230
RX Buffer Bypass	
Ports: • RXDLYALIGNDISABLE • RXDLYALIGNMONENB • RXDLYALIGNMONITOR[7:0] • RXDLYALIGNOVERRIDE • RXDLYALIGNRESET • RXDLYALIGNSWPPRECURB • RXDLYALIGNUPDSW • RXENPMAPHASEALIGN • RXPLLLKDET • RXPLLLKDETEN • RXPMASETPHASE • RXRECCLK • RXUSRCLK	page 232 page 232 page 232 page 232 page 232 page 232 page 232 page 232 page 232 page 232 page 232 page 232 page 232
Attributes: • RX_BUFFER_USE • RX_DATA_WIDTH • RX_DLYALIGN_CTRINC • RX_DLYALIGN_EDGESET • RX_DLYALIGN_LPFINC • RX_DLYALIGN_MONSEL • RX_DLYALIGN_OVRDSETTING • RX_XCLK_SEL • RXRECCLK_CTRL • RXUSRCLK_DLY • PMA_RXSYNC_CFG	page 233 page 233 page 233 page 233 page 233 page 233 page 233 page 233 page 234 page 234 page 234
RX Elastic Buffer	

Table 1-1: **Port and Attribute Summary *(Cont'd)***

Port/Attribute	Section, Page
Ports: • RXBUFRESET • RXBUFSTATUS[2:0]	 page 238 page 238
Attributes: • RX_BUFFER_USE • RX_EN_IDLE_RESET_BUF • RX_FIFO_ADDR_MODE • RX_IDLE_HI_CNT • RX_IDLE_LO_CNT • RX_XCLK_SEL	 page 238 page 238 page 238 page 238 page 239 page 239
RX Clock Correction	
Ports: • RXBUFRESET • RXBUFSTATUS[2:0] • RXCLKCORCNT[2:0]	 page 240 page 240 page 241
Attributes: • CLK_COR_ADJ_LEN • CLK_COR_DET_LEN • CLK_COR_INSERT_IDLE_FLAG • CLK_COR_KEEP_IDLE • CLK_COR_MAX_LAT • CLK_COR_MIN_LAT • CLK_COR_PRECEDENCE • CLK_COR_REPEAT_WAIT • CLK_COR_SEQ_1_1 • CLK_COR_SEQ_1_2 • CLK_COR_SEQ_1_3 • CLK_COR_SEQ_1_4 • CLK_COR_SEQ_1_ENABLE • CLK_COR_SEQ_2_1 • CLK_COR_SEQ_2_2 • CLK_COR_SEQ_2_3 • CLK_COR_SEQ_2_4 • CLK_COR_SEQ_2_ENABLE • CLK_COR_SEQ_2_USE • CLK_CORRECT_USE • RX_DATA_WIDTH • RX_DECODE_SEQ_MATCH	 page 241 page 241 page 241 page 241 page 241 page 242 page 242 page 242 page 242 page 242 page 242 page 242 page 242 page 243 page 243 page 243 page 243 page 243 page 243 page 243 page 243 page 243
RX Channel Bonding	

Table 1-1: **Port and Attribute Summary *(Cont'd)***

Port/Attribute	Section, Page
Ports: • RXCHANBONDSEQ • RXCHANISALIGNED • RXCHANREALIGN • RXCHBONDI[3:0] • RXCHBONDO[3:0] • RXCHBONDLEVEL[2:0] • RXCHBONDMASTER • RXCHBONDSLAVE • RXENCHANSYNC	 page 246 page 246 page 247 page 247 page 247 page 247 page 247 page 247 page 247
Attributes: • CHAN_BOND_1_MAX_SKEW • CHAN_BOND_2_MAX_SKEW • CHAN_BOND_KEEP_ALIGN • CHAN_BOND_SEQ_1_1 • CHAN_BOND_SEQ_1_2 • CHAN_BOND_SEQ_1_3 • CHAN_BOND_SEQ_1_4 • CHAN_BOND_SEQ_1_ENABLE • CHAN_BOND_SEQ_2_1 • CHAN_BOND_SEQ_2_2 • CHAN_BOND_SEQ_2_3 • CHAN_BOND_SEQ_2_4 • CHAN_BOND_SEQ_2_ENABLE • CHAN_BOND_SEQ_2_CFG • CHAN_BOND_SEQ_2_USE • CHAN_BOND_SEQ_LEN • PCI_EXPRESS_MODE • RX_DATA_WIDTH	 page 248 page 248 page 248 page 248 page 248 page 248 page 248 page 248 page 248 page 248 page 248 page 248 page 248 page 248 page 248 page 249 page 249 page 249
RX Gearbox	
Ports: • RXDATAVALID • RXGEARBOXSLIP • RXHEADER[2:0] • RXHEADERVALID • RXSTARTOFSEQ	 page 255 page 255 page 255 page 255 page 255
Attributes: • GEARBOX_ENDEC • RXGEARBOX_USE	 page 255 page 255
RX Initialization	

Table 1-1: **Port and Attribute Summary *(Cont'd)***

Port/Attribute	Section, Page
Ports:	
• GTXRXRESET	page 261
• PLLRXRESET	page 261
• PRBSCNTRESET	page 261
• RXBUFRESET	page 261
• RXCDRRESET	page 261
• RXDLYALIGNRESET	page 261
• RXRESET	page 261
• RXRESETDONE	page 261
• TSTIN[19:0]	page 261
Attributes:	
• CDR_PH_ADJ_TIME[4:0]	page 261
• RX_EN_IDLE_HOLD_CDR	page 261
• RX_EN_IDLE_HOLD_DFE	page 261
• RX_EN_IDLE_RESET_BUF	page 262
• RX_EN_IDLE_RESET_PH	page 262
• RX_EN_IDLE_RESET_FR	page 262
• RX_EN_MODE_RESET_BUF	page 262
• RX_EN_RATE_RESET_BUF	page 262
• RX_EN_REALIGN_RESET_BUF	page 262
• RX_IDLE_HI_CNT[3:0]	page 262
• RX_IDLE_LO_CNT[3:0]	page 262
FPGA RX Interface	
Ports:	
• MGTREFCLKFAB[1:0]	page 272
• RXCHARISK[3:0]	page 272
• RXDATA[31:0]	page 272
• RXDISPERR[3:0]	page 272
• RXUSRCLK	page 272
• RXUSRCLK2	page 272
Attributes:	
• GEN_RXUSRCLK	page 272
• RX_DATA_WIDTH	page 272

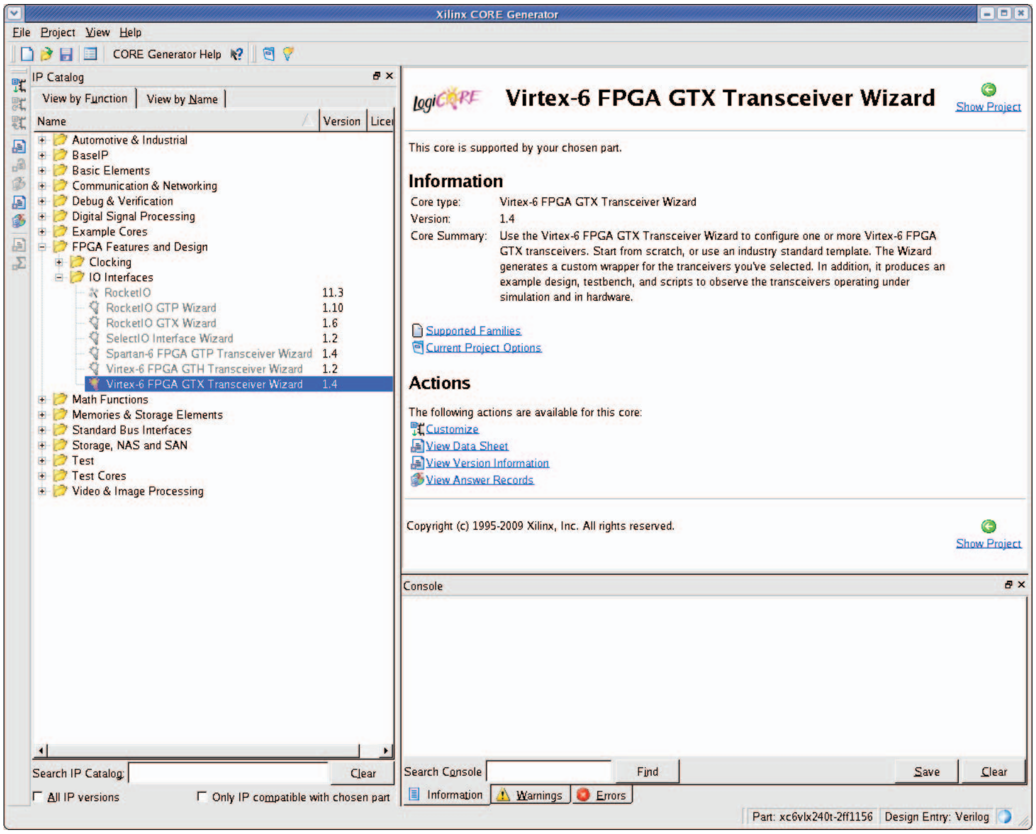
Virtex-6 FPGA GTX Transceiver Wizard

The Virtex-6 FPGA GTX Transceiver Wizard is the preferred tool to generate a wrapper to instantiate a GTX transceiver primitive called GTXE1. The Wizard can be found in the CORE Generator tool. Be sure to download the most up-to-date IP Update before using the Wizard. Details on how to use this Wizard can be found in UG516, *LogiCORE IP Virtex-6 FPGA GTX Transceiver Wizard User Guide*.

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

```
/FPGA Features & Design/IO Interfaces
```

See Figure 1-4.



- Automotive & Industrial
- BaseIP
- Basic Elements
- Communication & Networking
- Debug & Verification
- Digital Signal Processing
- Example Cores
- FPGA Features and Design
  - Clocking
  - IO Interfaces

Name	Version
RocketIO	11.3
RocketIO GTP Wizard	1.10
RocketIO GTX Wizard	1.6
SelectIO Interface Wizard	1.2
Spartan-6 FPGA GTP Transceiver Wizard	1.4
Virtex-6 FPGA GTH Transceiver Wizard	1.2
Virtex-6 FPGA GTX Transceiver Wizard	1.4

- Math Functions
- Memories & Storage Elements
- Standard Bus Interfaces
- Storage, NAS and SAN
- Test
- Test Cores
- Video & Image Processing

LogiCORE Virtex-6 FPGA GTX Transceiver Wizard

This core is supported by your chosen part.

Information

Core type: Virtex-6 FPGA GTX Transceiver Wizard
Version: 1.4
Core Summary: Use the Virtex-6 FPGA GTX Transceiver Wizard to configure one or more Virtex-6 FPGA GTX transceivers. Start from scratch, or use an industry standard template. The Wizard generates a custom wrapper for the tranceivers you've selected. In addition, it produces an example design, testbench, and scripts to observe the transceivers operating under simulation and in hardware.

Supported Families
Current Project Options

Actions

The following actions are available for this core:
Customize
View Data Sheet
View Version Information
View Answer Records

Copyright (c) 1995-2009 Xilinx, Inc. All rights reserved.

Part: xc6vlx240t-2ff1156 Design Entry: Verilog

UG366_c1_04_010710

Figure 1-4: **Virtex-6 FPGA GTX Transceiver Wizard**

3. Double-click **V6 GTX Wizard** to launch the Wizard.

Simulation

Functional Description

Simulations using GTX transceivers have specific prerequisites that the simulation environment and the test bench must fulfill.

The *Synthesis and Simulation Design Guide* explains how to set up the simulation environment for supported simulators depending on the used Hardware Description Language (HDL). This design guide can be downloaded from the Xilinx website.

The prerequisites for simulating a design with GTX transceivers are:

- Simulator with support for SecureIP models, which are encrypted versions of the Verilog HDL used for implementation of the modeled block.

  SecureIP is a new IP encryption methodology. To support SecureIP models, a Verilog LRM - IEEE Std 1364-2005 encryption compliant simulator is required.

- 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 capable of simulating VHDL and Verilog simultaneously.

- Installed GTX SecureIP model.
- Correct setup of the simulator for SecureIP use (initialization file, environment variable(s)).
- Running COMPXLIB (which compiles the simulation libraries (e.g. UNISIM, SIMPRIMS, etc.) in the correct order.
- Correct simulator resolution (Verilog)
- The user guide of the simulator and the *Synthesis and Simulation Design Guide* provide a detailed list of settings for SecureIP support.

Ports and Attributes

There are no simulation-only ports.

The `GTXE1` primitive has attributes intended only for simulation. Table 1-2 lists the *simulation-only* attributes of the `GTXE1` primitive. The names of these attributes start with *SIM_*.

Table 1-2: **GTXE1 Simulation-Only Attributes**

Attribute	Type	Description
SIM_GTXRESET_SPEEDUP	Integer	This attribute shortens the time it takes to finish the GTXRXRESET and GTXTXRESET sequence and lock the TX PMA PLL and RX PMA PLL during simulation. 0: The GTXRXRESET and GTXTXRESET sequence is simulated with its original duration (standard initialization is approximately 120 µs). 1: Shorten the GTXRXRESET and GTXTXRESET cycle time (fast initialization is approximately 300 ns).
SIM_RECEIVER_DETECT_PASS	Boolean	This attribute simulates the TXDETECTRX feature in the GTX transceiver. TRUE: Simulates an RX connection to the TX serial ports. TXDETECTRX initiates receiver detection, and RXSTATUS[2:0] = `011` reports that an RX port is connected. FALSE (default): Simulates a disconnected TX port. TXDETECTRX initiates receiver detection, and RXSTATUS[2:0] = `000` reports that an RX port is not connected.

Table 1-2: **GTXE1 Simulation-Only Attributes *(Cont'd)***

Attribute	Type	Description
SIM_RXREFCLK_SOURCE	3-Bit Binary	This attribute selects the reference clock source used to drive the RX PMA PLL in simulation for designs where the RX PMA PLL is always driven by the same reference clock source. The RXPLLREFSELDY port must be set to `000` for this attribute to select the reference clock source. For multi-rate designs that require the reference clock source to be changed on the fly, the RXPLLREFSELDY port is used to dynamically select the source instead. `000`: Selects the MGTREFCLKRX[0] port as the source `001`: Selects the MGTREFCLKRX[1] port as the source `010`: Selects the NORTHREFCLKRX[0] port as the source `011`: Selects the NORTHREFCLKRX[1] port as the source `100`: Selects the SOUTHREFCLKRX[0] port as the source `101`: Selects the SOUTHREFCLKRX[1] port as the source `110`: Reserved `111`: Selects a clock from the FPGA logic which can be either port GREFCLKRX or PERFCLKRX as the source
SIM_TX_ELEC_IDLE_LEVEL	1-Bit Binary	This attribute sets the value of TXN and TXP during simulation of electrical idle. This attribute can be set to 0, 1, x, or z. The default for this attribute is *x*.
SIM_TXREFCLK_SOURCE	3-Bit Binary	This attribute selects the reference clock source used to drive the TX PMA PLL in simulation for designs where the TX PMA PLL is always driven by the same reference clock source. The TXPLLREFSELDY port must be set to `000` for this attribute to select the reference clock source. For multi-rate designs that require the reference clock source to be changed on the fly, the TXPLLREFSELDY port is used to dynamically select the source instead. `000`: Selects the MGTREFCLKTX[0] port as the source `001`: Selects the MGTREFCLKTX[1] port as the source `010`: Selects the NORTHREFCLKTX[0] port as the source `011`: Selects the NORTHREFCLKTX[1] port as the source `100`: Selects the SOUTHREFCLKTX[0] port as the source `101`: Selects the SOUTHREFCLKTX[1] port as the source `110`: Selects the RX recovered clock from the RX channel as the source `111`: Selects a clock from the FPGA logic that can be either the GREFCLKTX or the PERFCLKTX port as the source
SIM_VERSION	Real	This attribute selects the simulation version to match different steppings of silicon. The default for this attribute is 1.0.

SIM_GTXRESET_SPEEDUP

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

If TXOUTCLK or RXRECCLK is used to generate clocks in the design, these clocks occasionally flatline while the GTX transceiver is locking. If an MMCM is used to divide TXOUTCLK or RXRECCLK, the final output clock is not ready until both the GTX transceiver and the MMCM have locked. Equation 1-1 provides an estimate of the time required before a stable source from TXOUTCLK or RXRECCLK is available in simulation, including the time required for any MMCMs used.

$$t_{USRCLKstable} \cong t_{GTXRESETsequence} + t_{locktimeMMCM} \qquad \text{Equation 1-1}$$

If the MMCM is not used, the term can be removed from the lock time equation.

SIM_RECEIVER_DETECT_PASS

The GTX transceiver includes a TXDETECTRX feature that allows the transmitter to detect whether its serial ports are currently connected to a receiver by measuring rise time on the TXP/TXN differential pin pair (see TX Receiver Detect Support for PCI Express Designs, page 180).

The GTXE1 SecureIP model includes an attribute for simulating TXDETECTRX called SIM_RECEIVER_DETECT_PASS. This attribute allows TXDETECTRX to be simulated for the GTX transceiver without modeling the measurement of rise time on the TXP/TXN differential pin pair.

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

SIM_RXREFCLK_SOURCE

The GTXE1 SecureIP model includes an attribute to select the reference clock source used to drive the RX PMA PLL in simulation called SIM_RXREFCLK_SOURCE. This attribute is to be used in designs where the RX PMA PLL's clock input is always driven by the same reference clock source.

Reference clock sources include the dedicated clock pins of the Quad that the transceiver belongs to, the north-running reference clocks, the south-running reference clocks, and a clock from the FPGA logic. Table 1-2 shows the possible settings for this attribute.

For multi-rate designs requiring the reference clock source driving the RX PMA PLL to be changed on the fly, the RXPLLREFSELDY port is used to dynamically select the reference clock source instead.

SIM_TXREFCLK_SOURCE

The GTXE1 SecureIP model includes an attribute to select the reference clock source used to drive the TX PMA PLL in simulation called SIM_TXREFCLK_SOURCE. This attribute is to be used in designs where the TX PMA PLL's clock input is always driven by the same reference clock source.

Reference clock sources include the dedicated clock pins of the Quad that the transceiver belongs to, the north-running reference clocks, the south-running reference clocks, and a clock from the FPGA logic. Table 1-2 shows the possible settings for this attribute.

For multi-rate designs requiring the reference clock source driving the TX PMA PLL to be changed on the fly, the TXPLLREFSELDY port is used to dynamically select the reference clock source instead.

SIM_VERSION

The SIM_VERSION attribute selects the simulation version to match different steppings of silicon. The default for this attribute is 1.0.

SIM_TX_ELEC_IDLE_LEVEL

The SIM_TX_ELEC_IDLE_LEVEL attribute sets the value of the transceiver's differential transmitter output pair TXN and TXP during simulation of electrical idle. This attribute can be set to 0, 1, x, or z. The default for this attribute is *x*.

Implementation

Functional Description

This section provides the information needed to map Virtex-6 FPGA GTX transceivers instantiated in a design to device resources, including:

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

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

While this section describes how to instantiate GTX clocking components, the details of the different GTX transceiver clocking options are discussed in Reference Clock Selection, page 102.

The position of the GTX transceiver is specified by an XY coordinate system that describes the column number and its relative position within that column. In current members of the Virtex-6 family, all GTX transceivers are located in a single column along one side of the die.

The transceiver with the coordinates "X0Y0" is for a given device/package combination always located at the lowest position of the lowest available bank. For the combination of a package with a large pin count (for example, 1759) and a smaller device (for example, XC6VLX240T), transceivers at higher or lower banks are not available.

There are two ways to create a UCF for designs that utilize the GTX transceiver. The preferred method is to use the Virtex-6 FPGA GTX Transceiver Wizard (see Virtex-6 FPGA GTX Transceiver Wizard, page 36). The Wizard automatically generates UCF templates that configure the transceivers and contain placeholders for GTX 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 transceiver are correctly entered.

Figure 1-5, page 42 through Figure 1-23, page 60 provide the GTX transceiver position information for all available device and package combinations along with the pad numbers for the external signals associated with each GTX transceiver.

FF484 Package Placement Diagrams

Figure 1-5 through Figure 1-6 show the placement diagrams for the FF484 package.

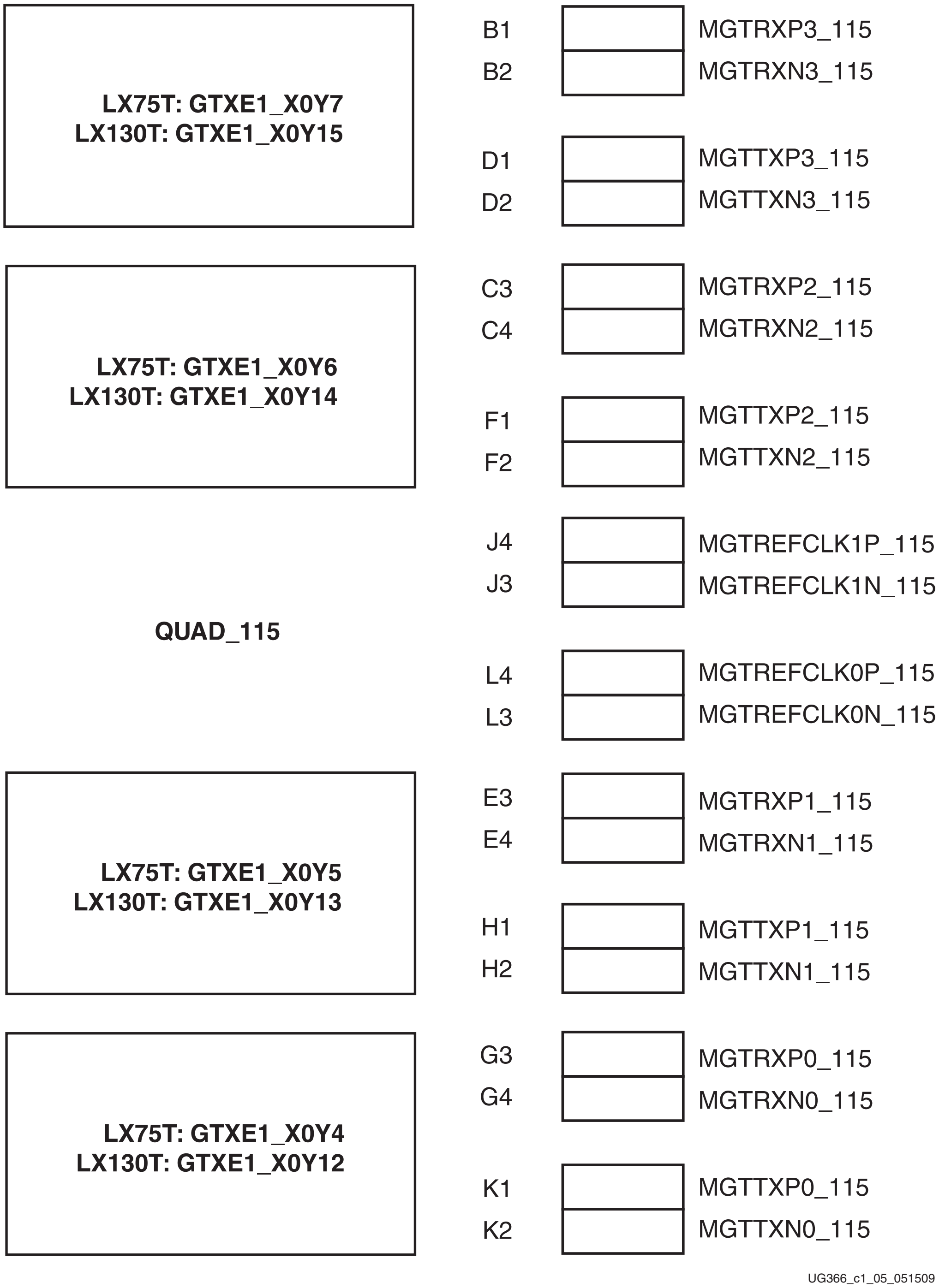


Figure 1-5: **Placement Diagram for the FF484 Package (1 of 2)**

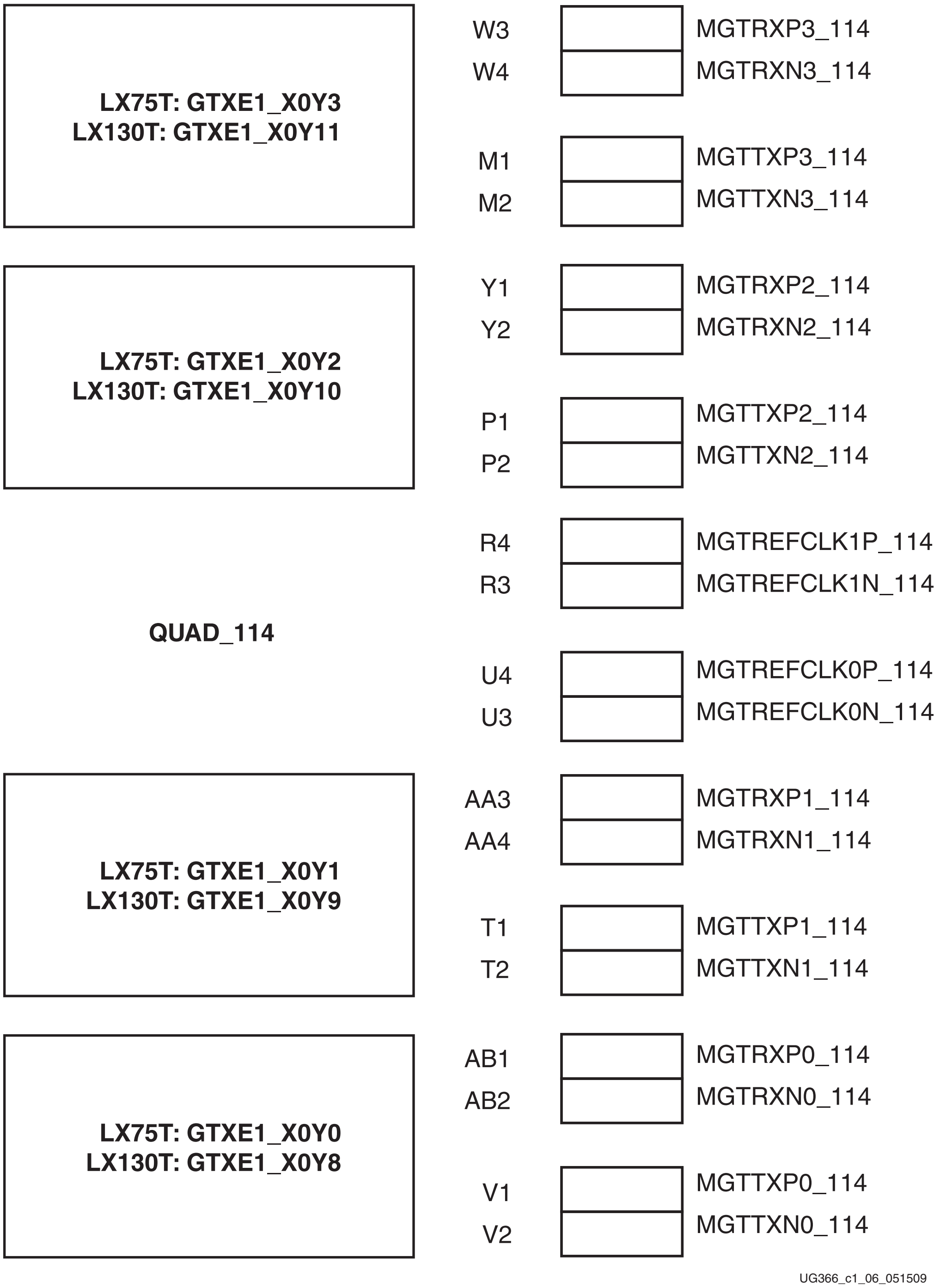


Figure 1-6: **Placement Diagram for the FF484 Package (2 of 2)**

FF784 Package Placement Diagrams

Figure 1-7 through Figure 1-9 show the placement diagrams for the FF784 package.

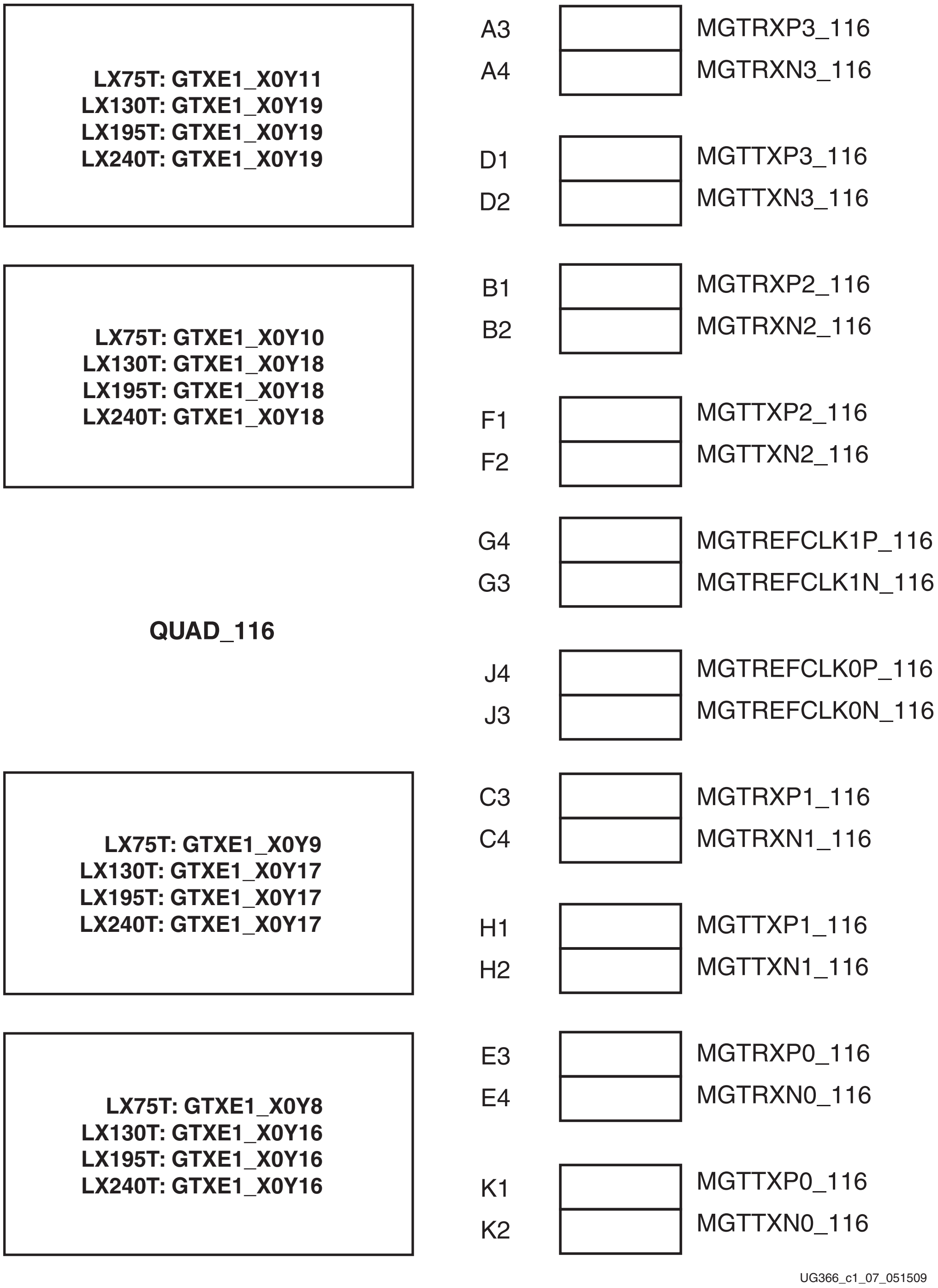


Figure 1-7: **Placement Diagram for the FF784 Package (1 of 3)**

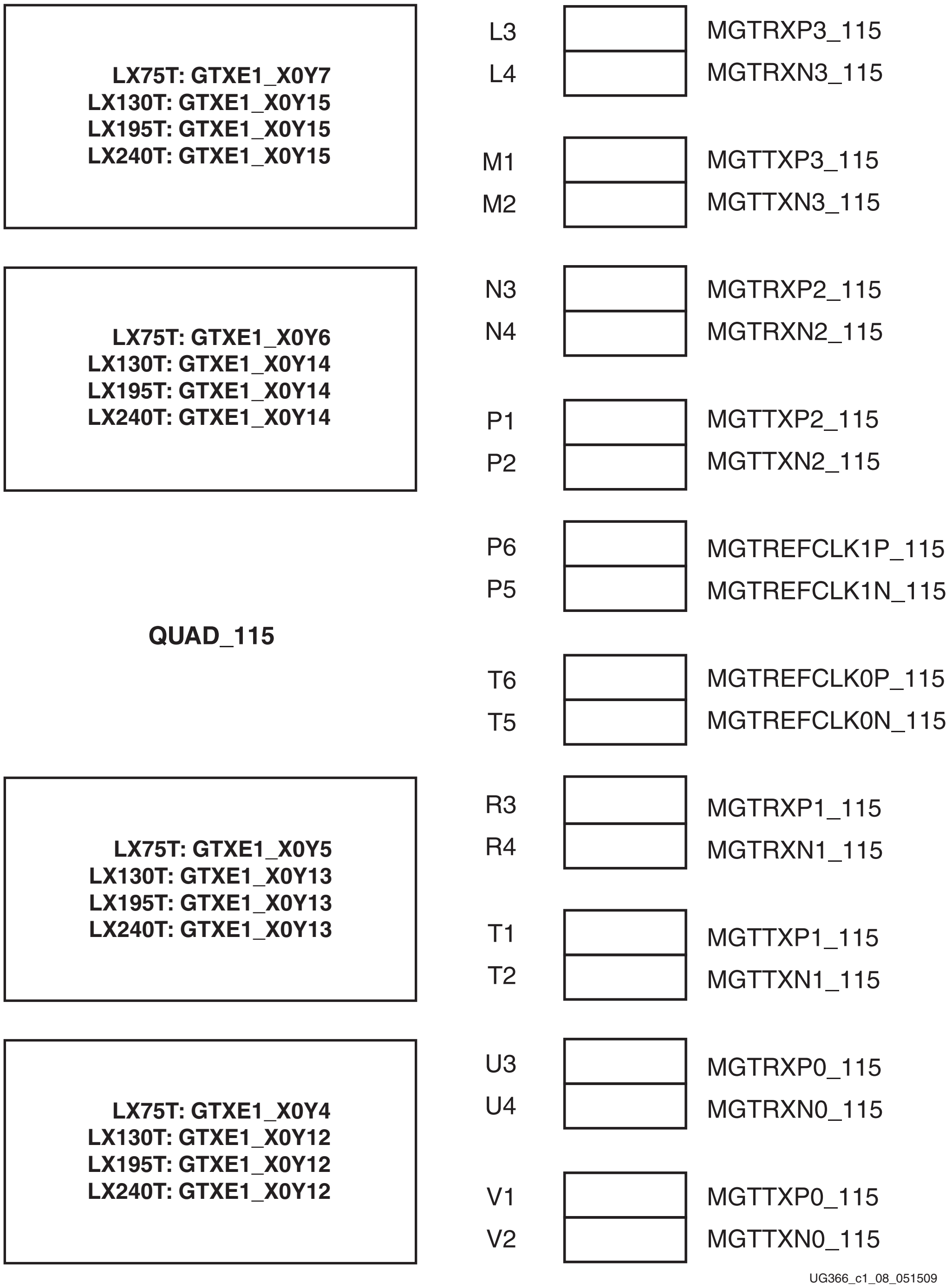
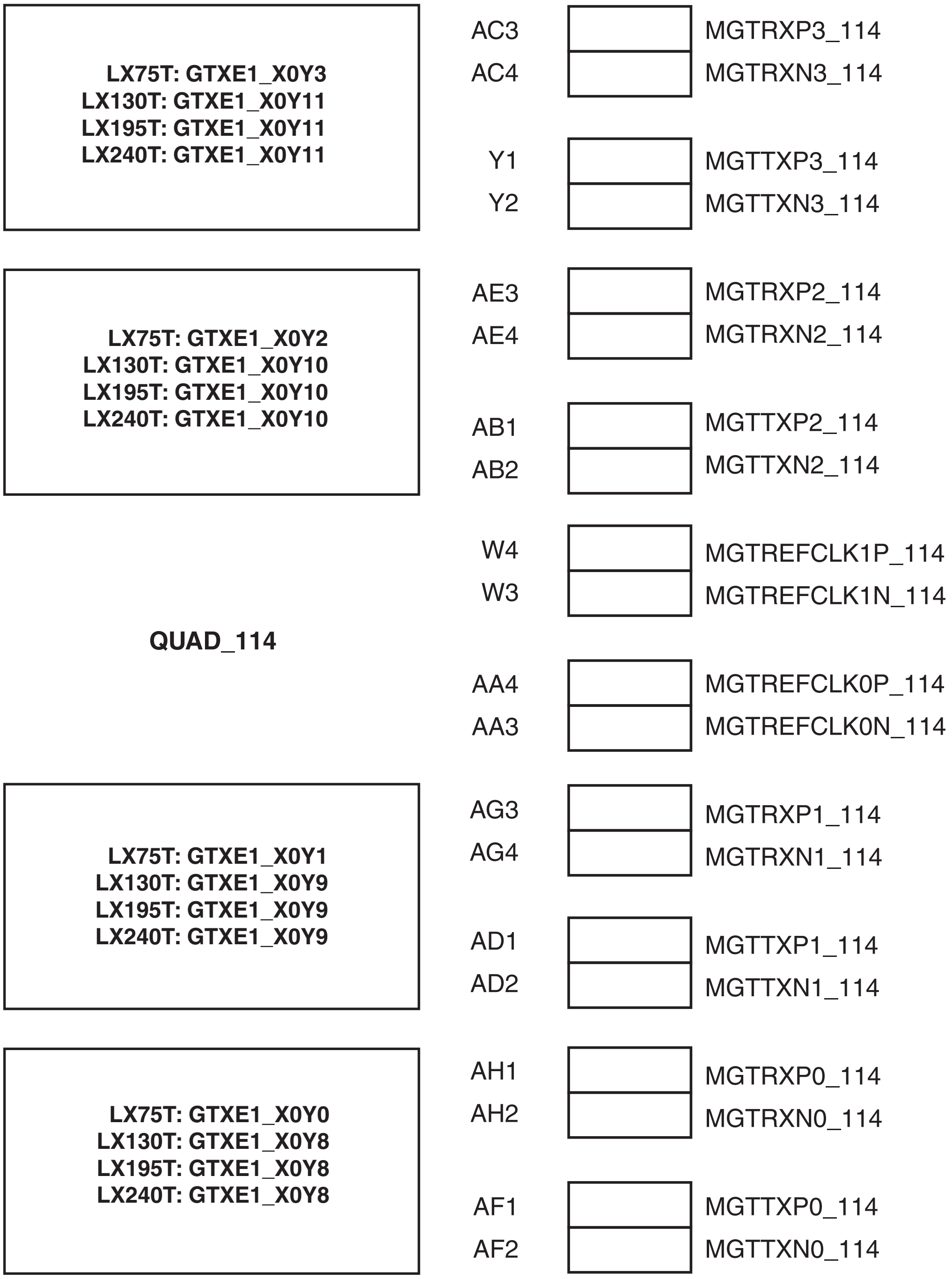


Figure 1-8: **Placement Diagram for the FF784 Package (2 of 3)**



UG366_c1_09_122109

Figure 1-9: **Placement Diagram for the FF784 Package (3 of 3)**

FF1156 Package Placement Diagrams

Figure 1-10 through Figure 1-14 show the placement diagrams for the FF1156 package.

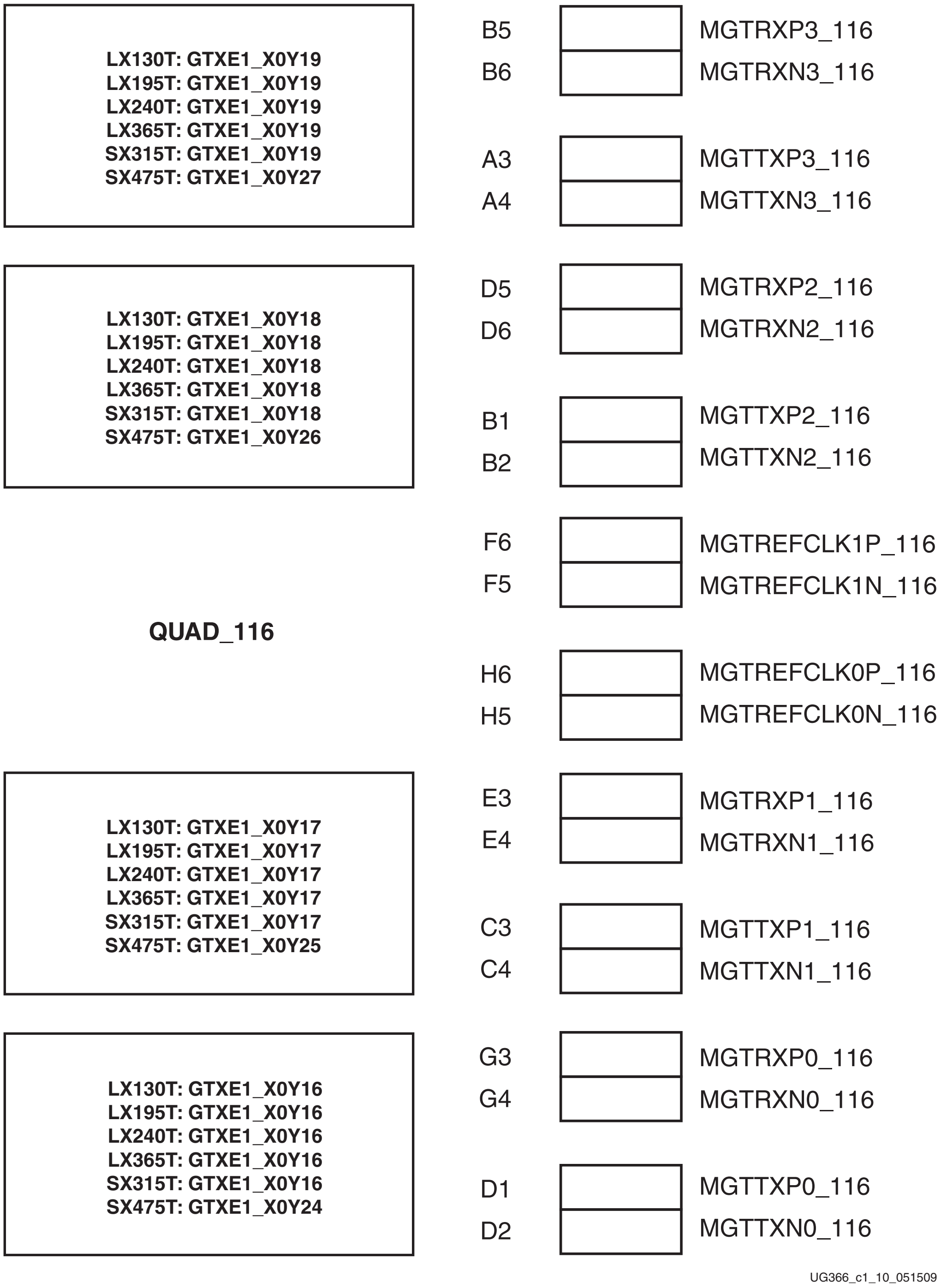
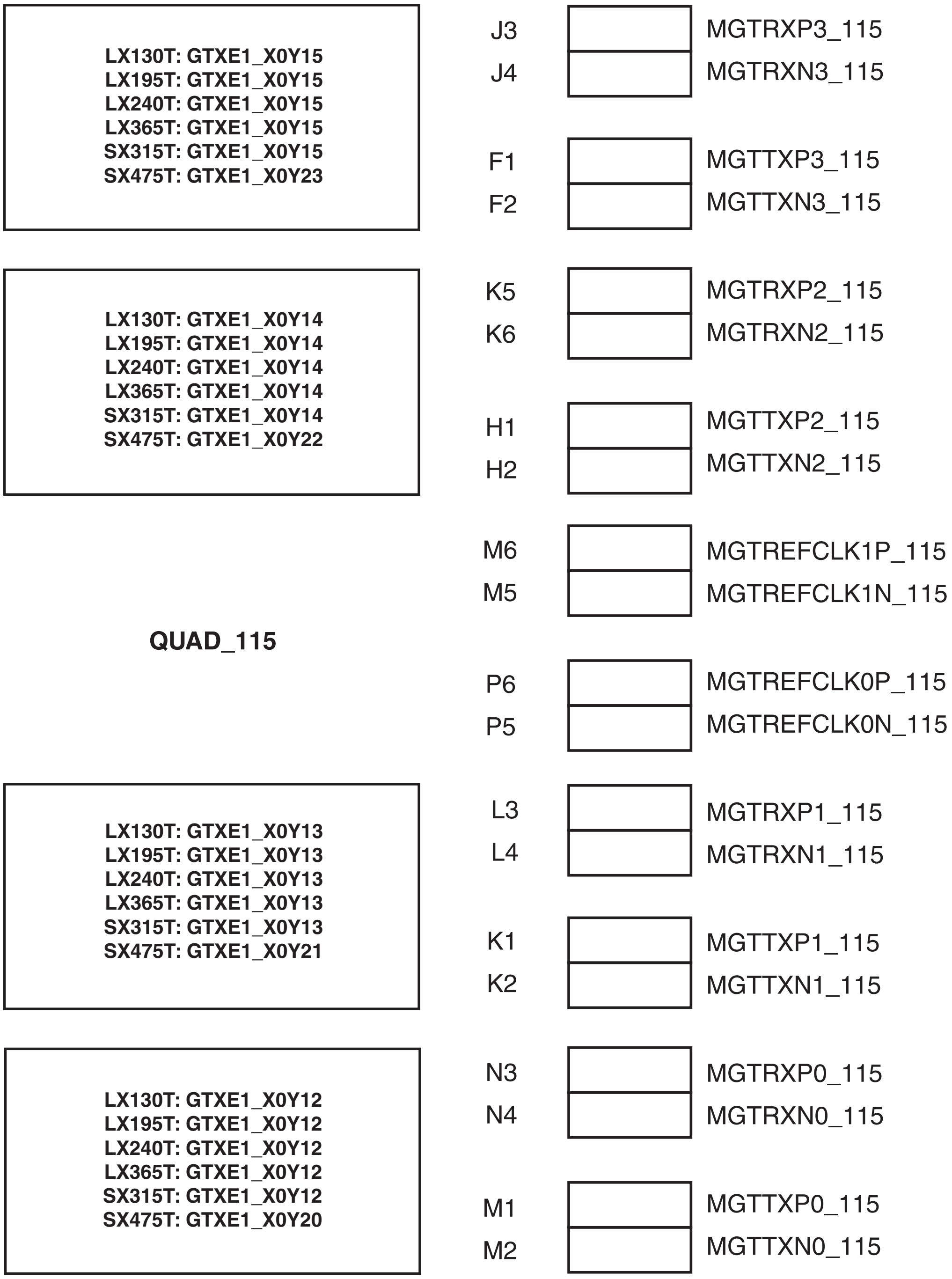


Figure 1-10: **Placement Diagram for the FF1156 Package (1 of 5)**



UG366_c1_11_051509

Figure 1-11: **Placement Diagram for the FF1156 Package (2 of 5)**

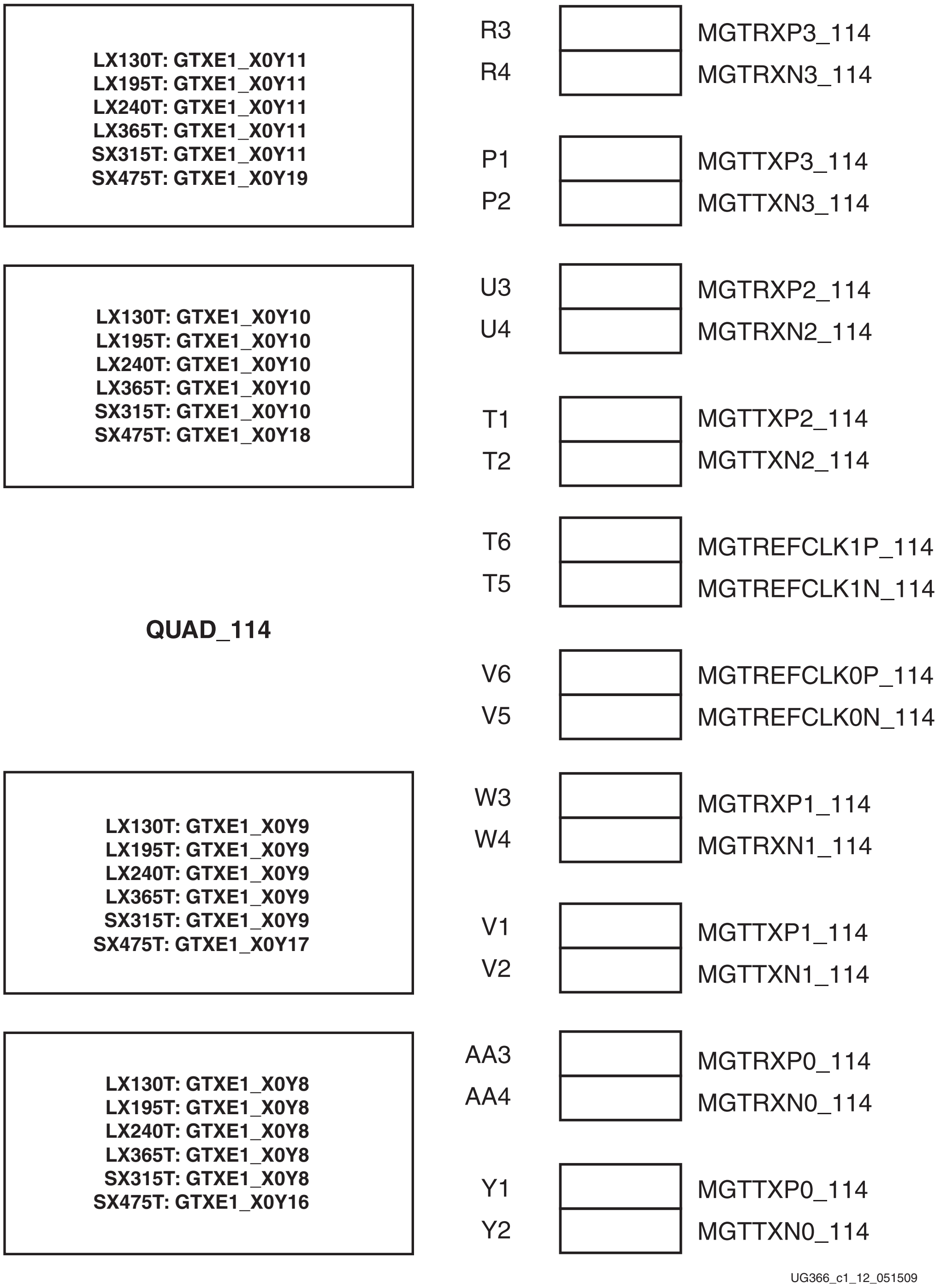


Figure 1-12: **Placement Diagram for the FF1156 Package (3 of 5)**

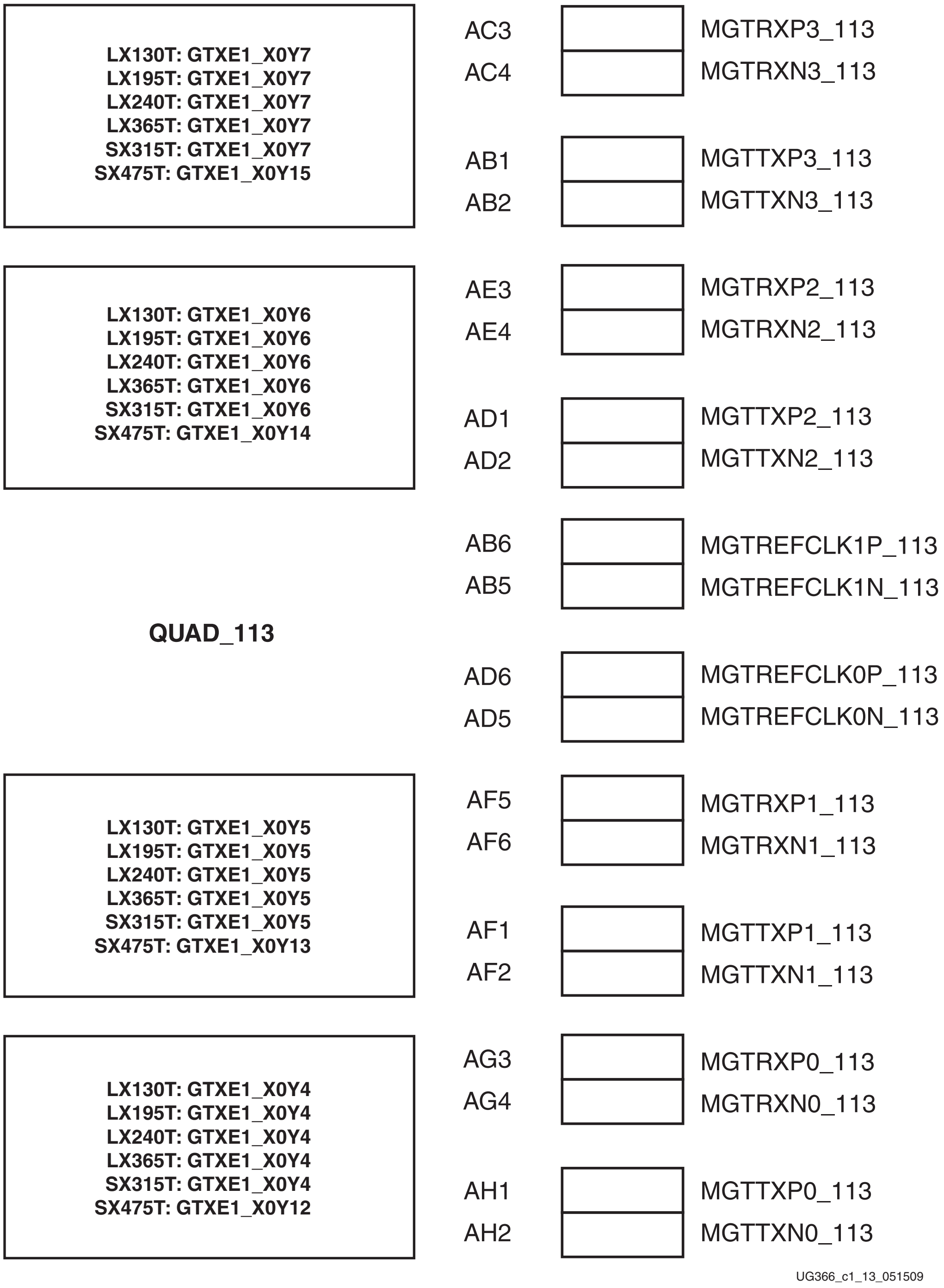


Figure 1-13: **Placement Diagram for the FF1156 Package (4 of 5)**

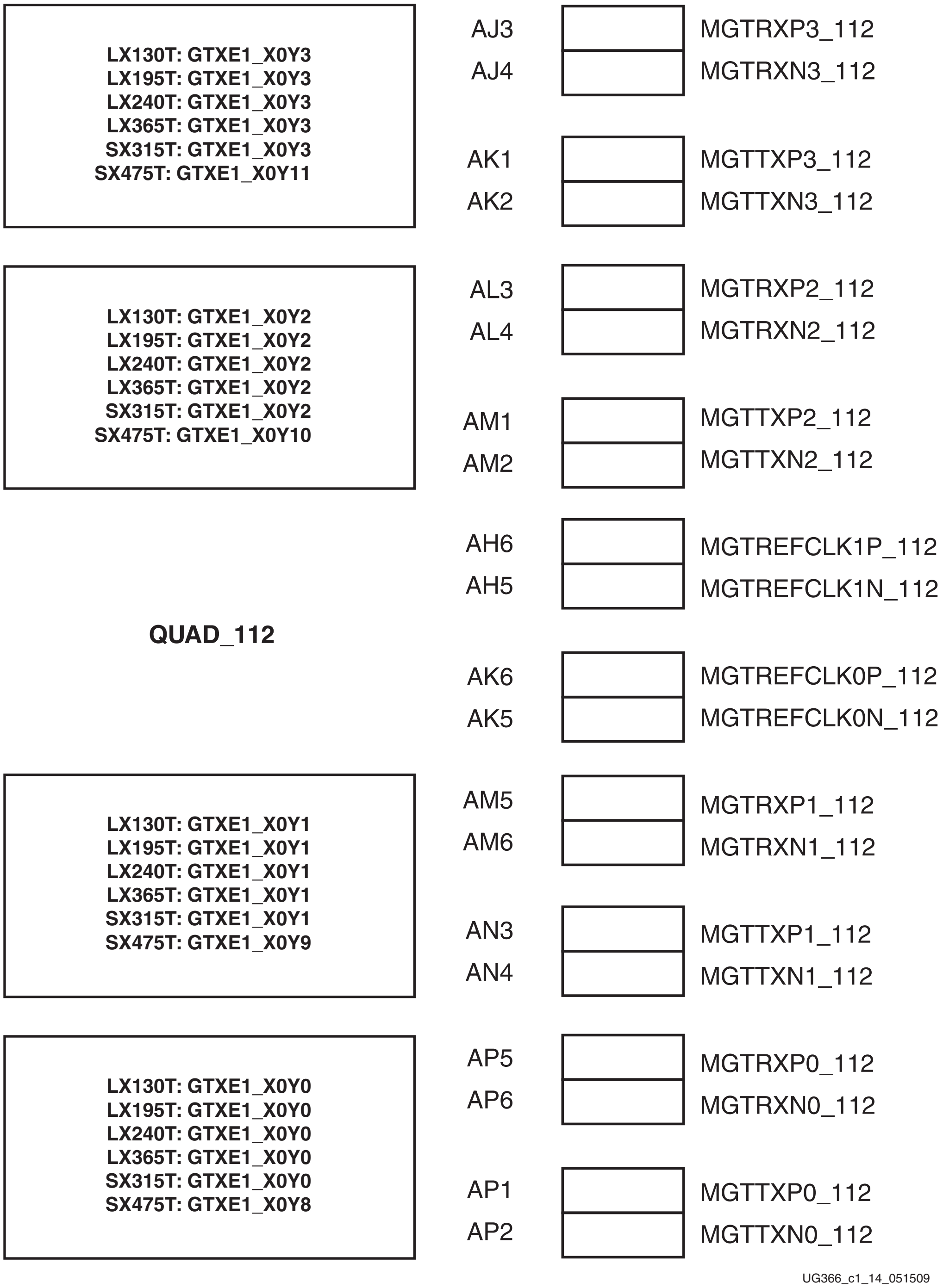


Figure 1-14: **Placement Diagram for the FF1156 Package (5 of 5)**

FF1759 Package Placement Diagrams

Figure 1-15 through Figure 1-23 show the placement diagrams for the FF1759 package.

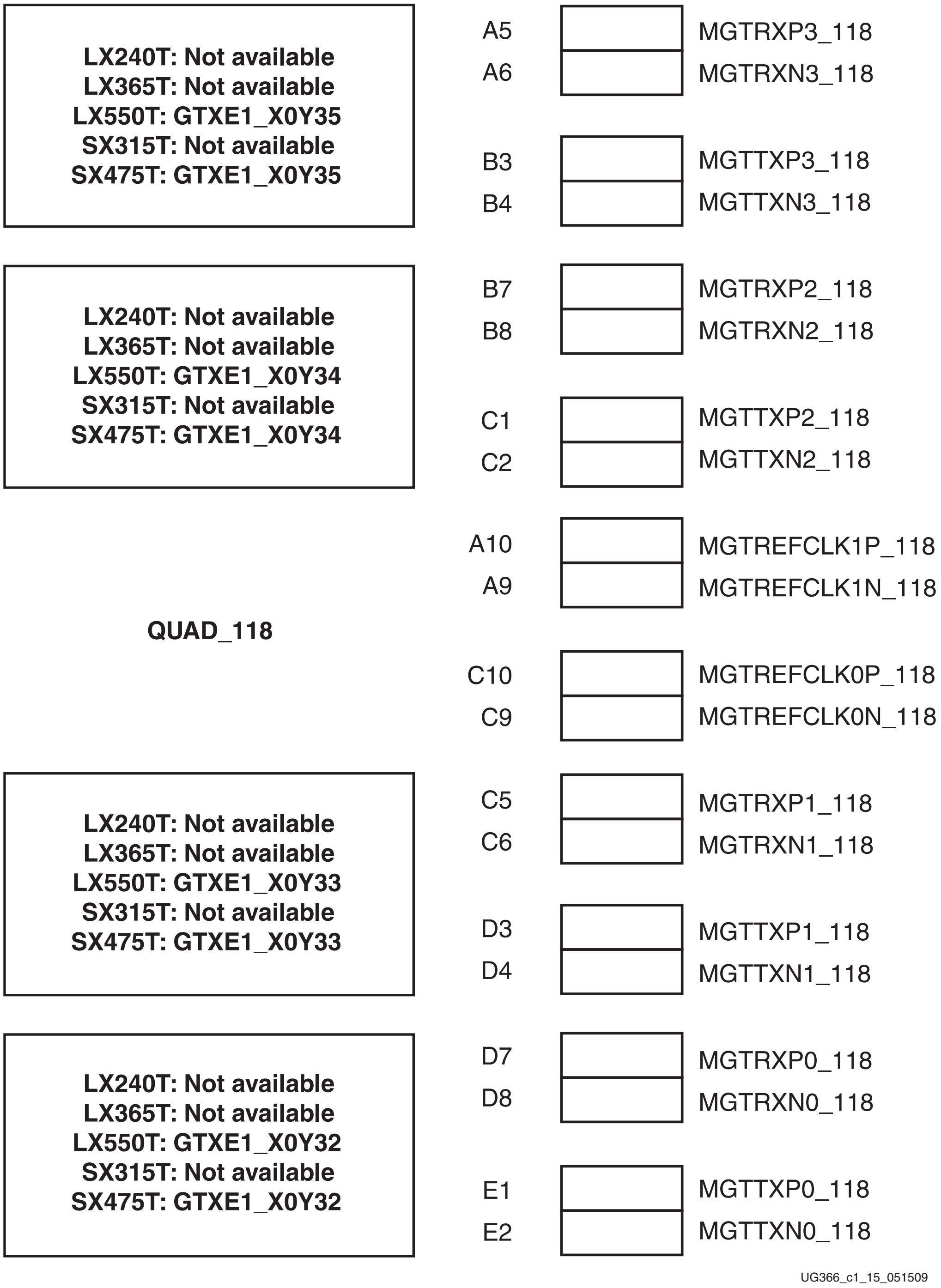


Figure 1-15: **Placement Diagram for the FF1759 Package (1 of 9)**

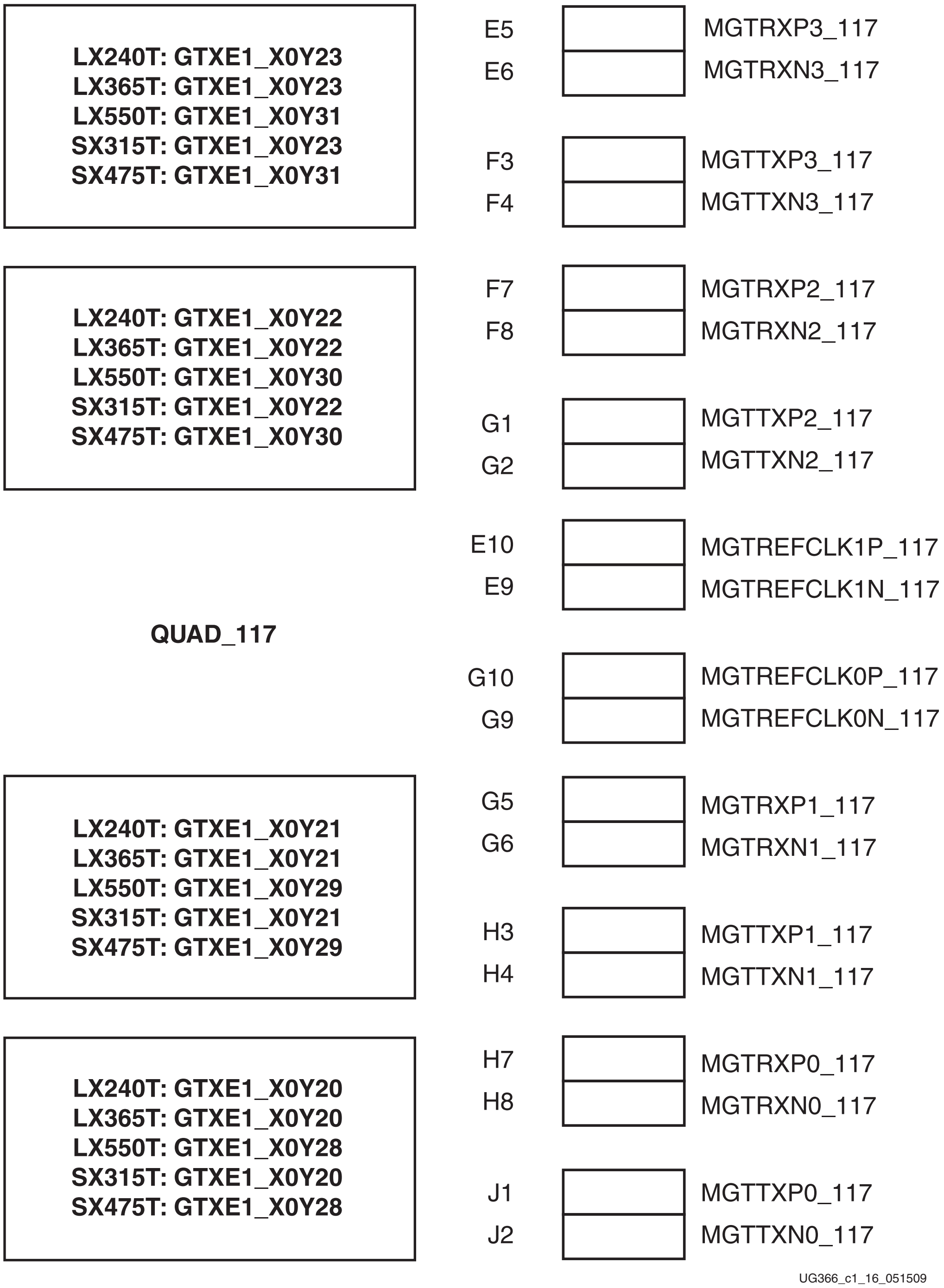


Figure 1-16: **Placement Diagram for the FF1759 Package (2 of 9)**

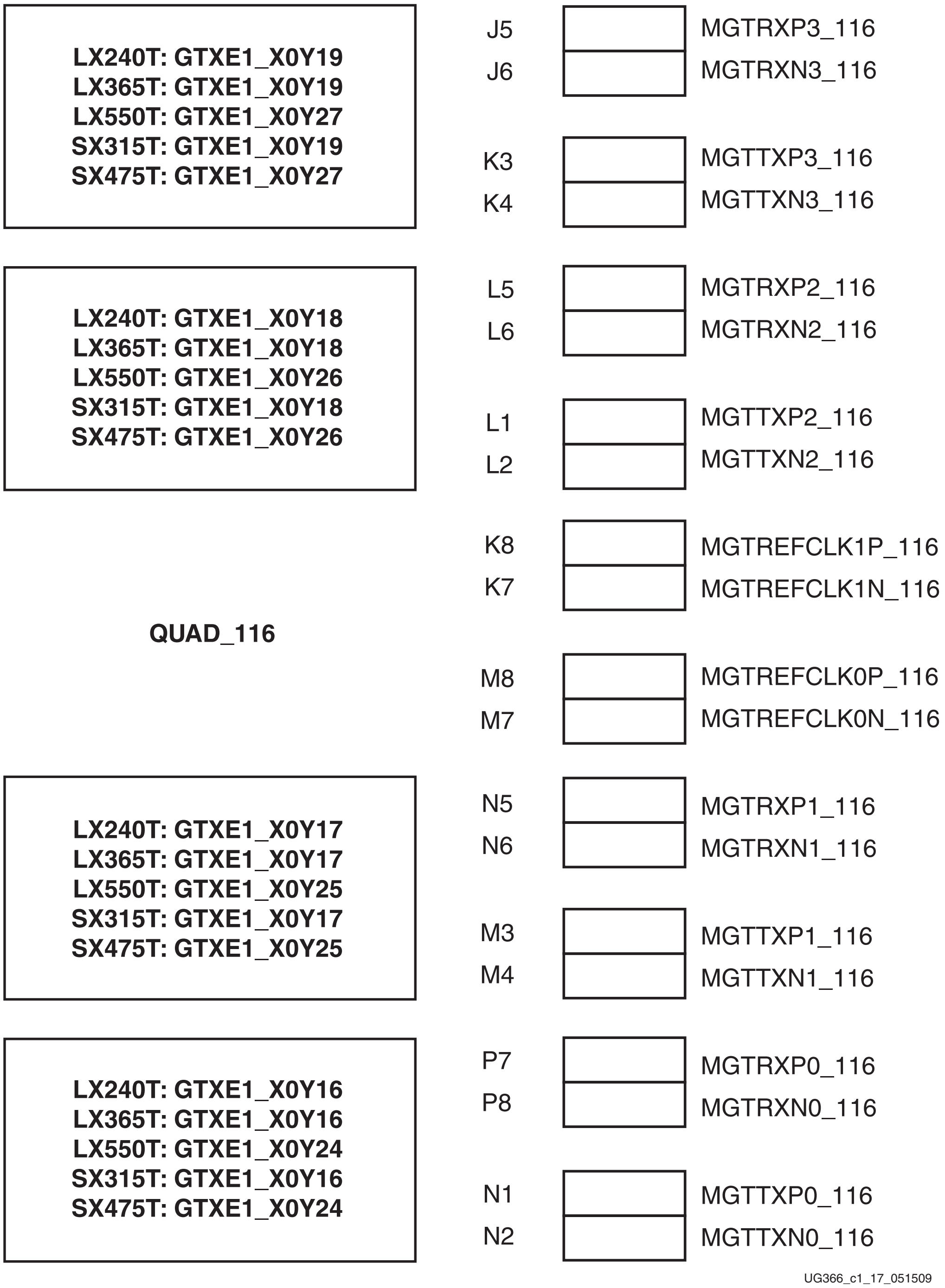


Figure 1-17: **Placement Diagram for the FF1759 Package (3 of 9)**

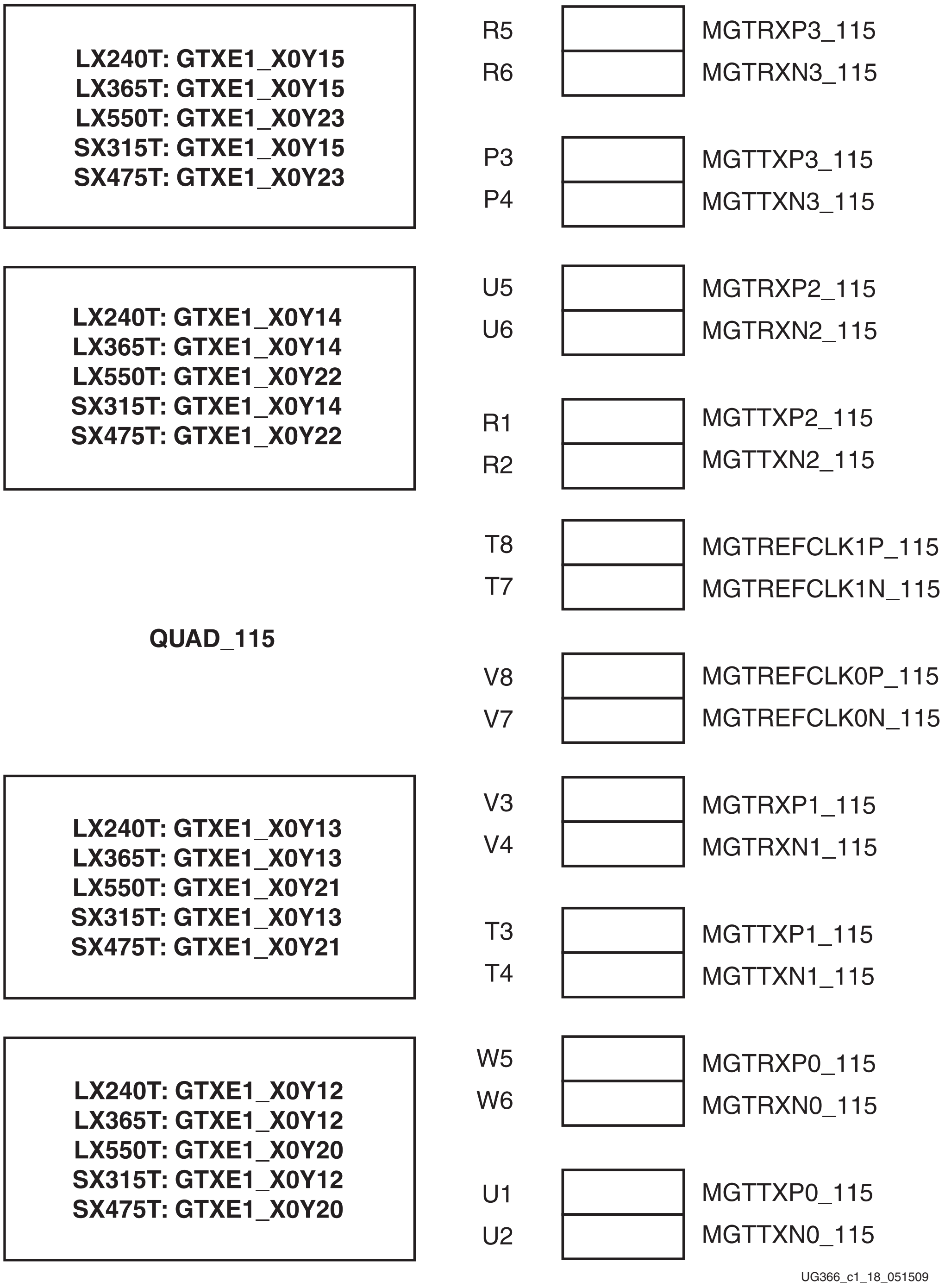


Figure 1-18: **Placement Diagram for the FF1759 Package (4 of 9)**

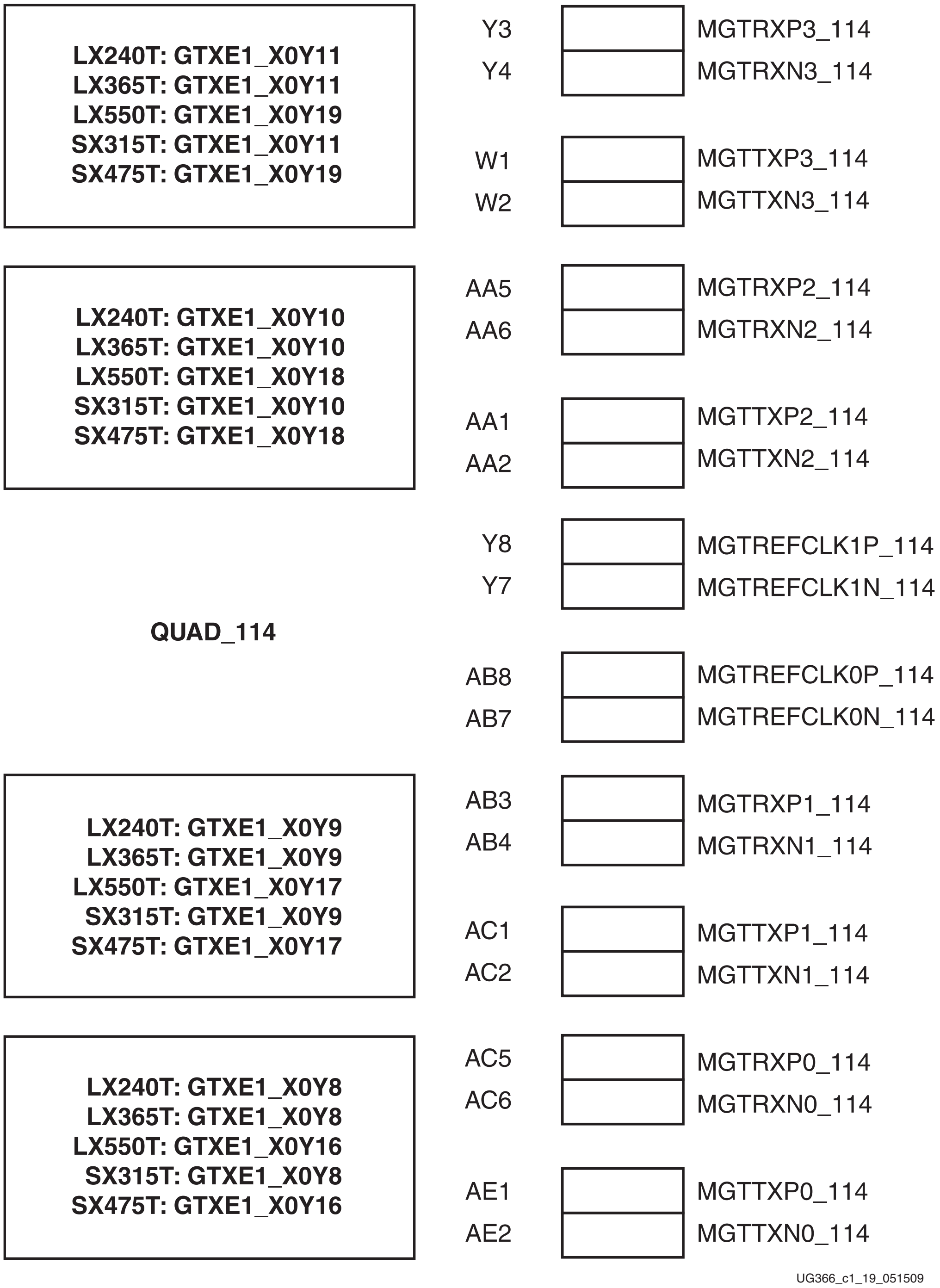


Figure 1-19: **Placement Diagram for the FF1759 Package (5 of 9)**

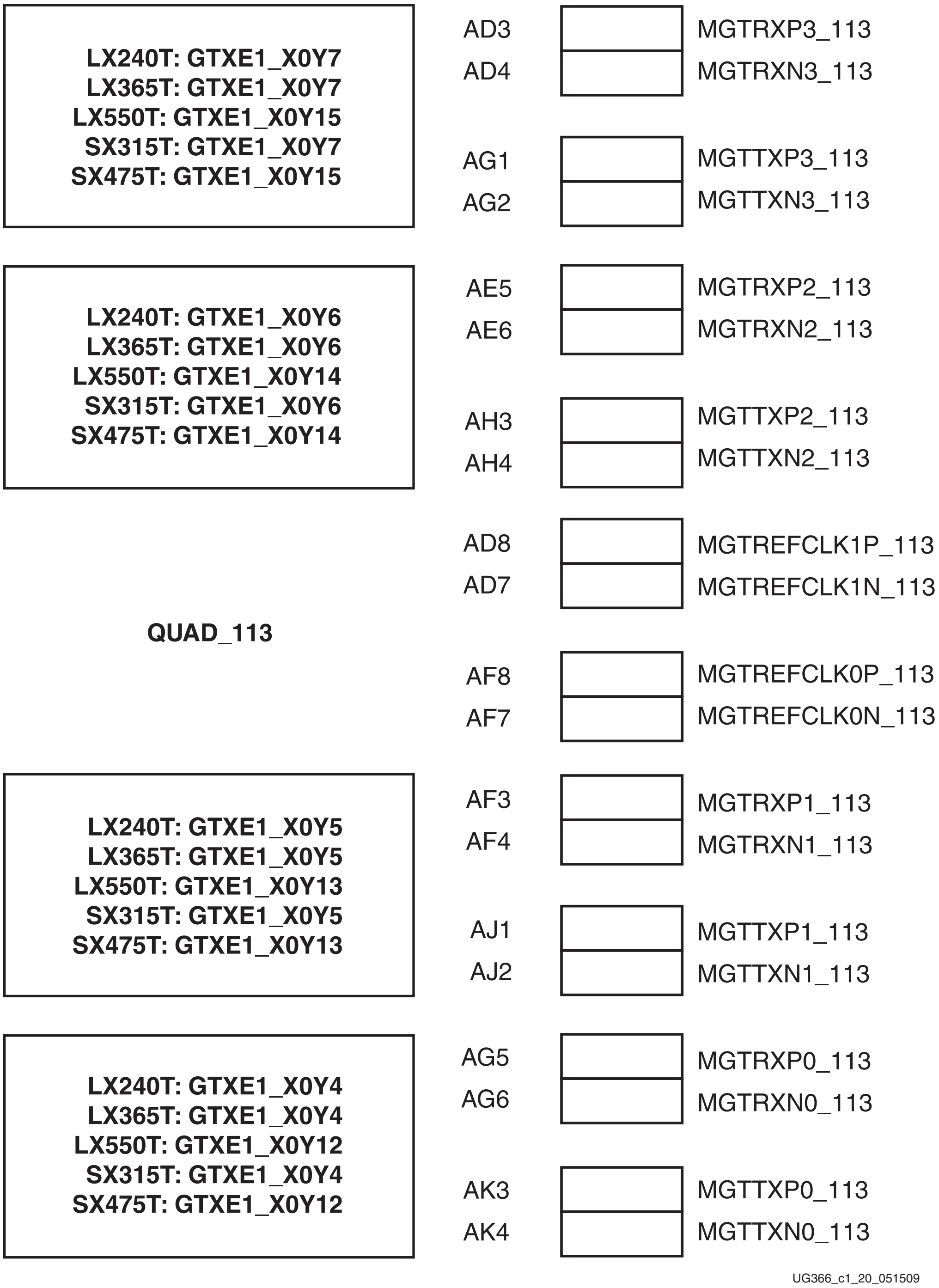


Figure 1-20: **Placement Diagram for the FF1759 Package (6 of 9)**

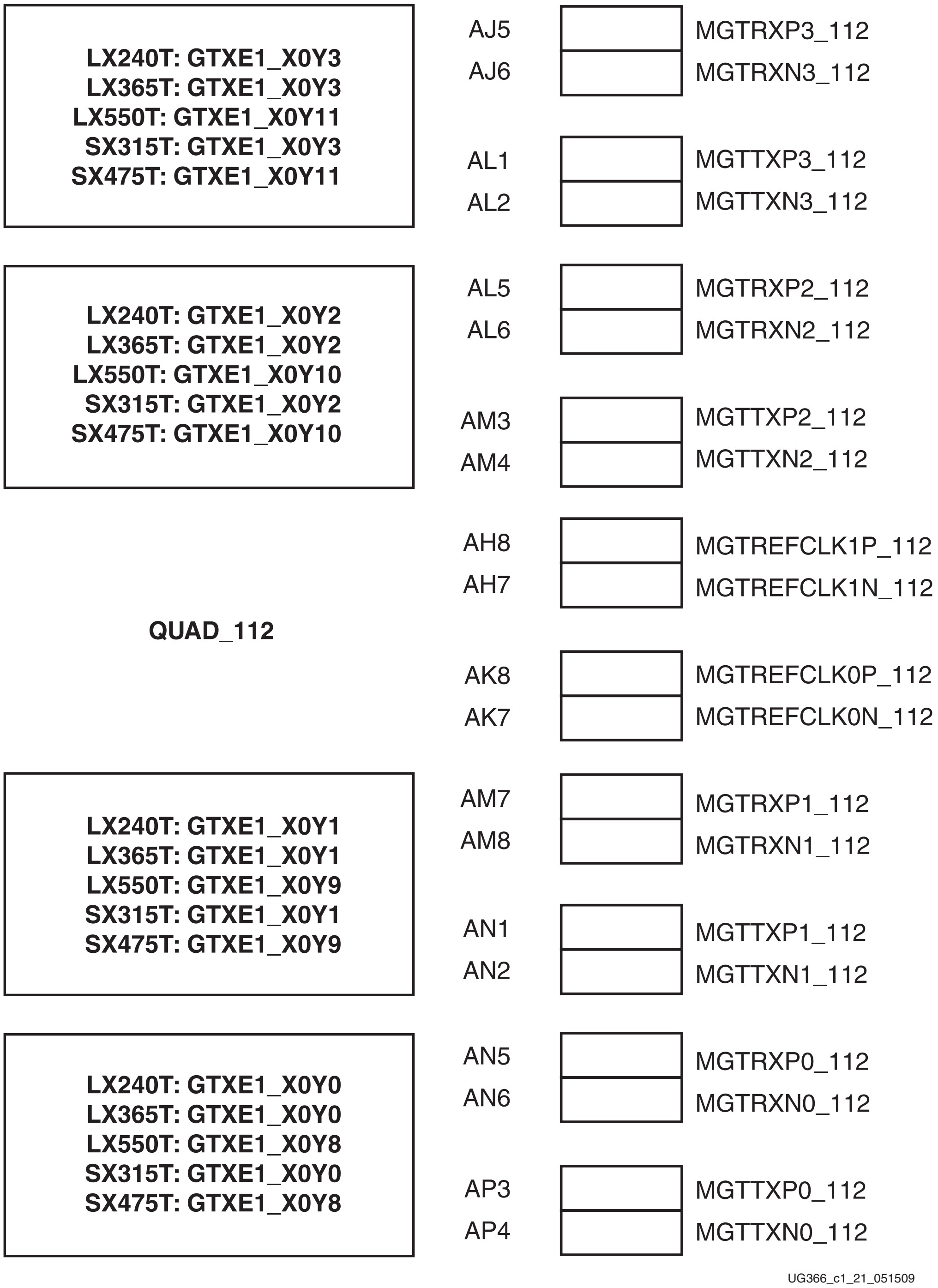


Figure 1-21: **Placement Diagram for the FF1759 Package (7 of 9)**

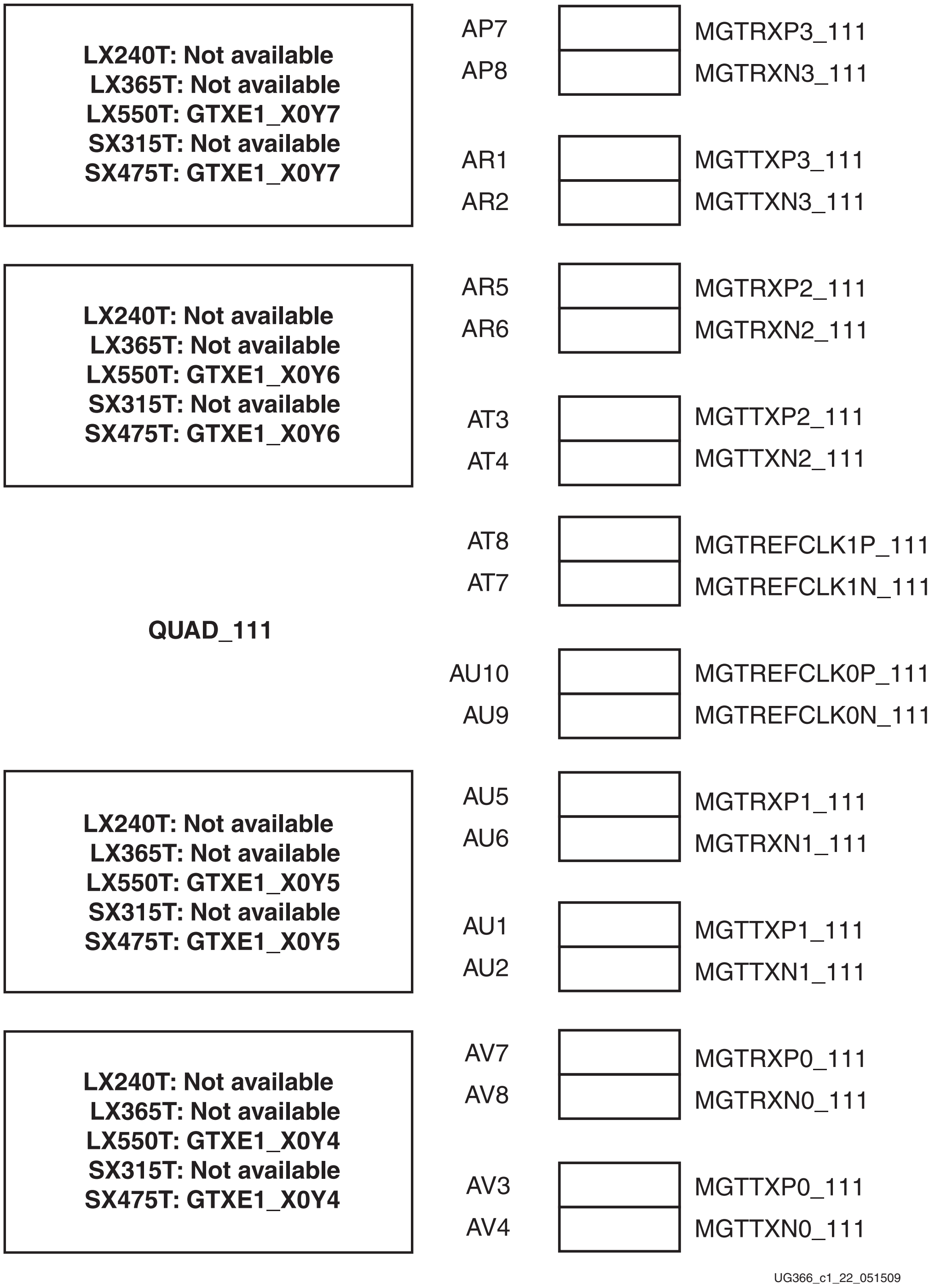


Figure 1-22: **Placement Diagram for the FF1759 Package (8 of 9)**

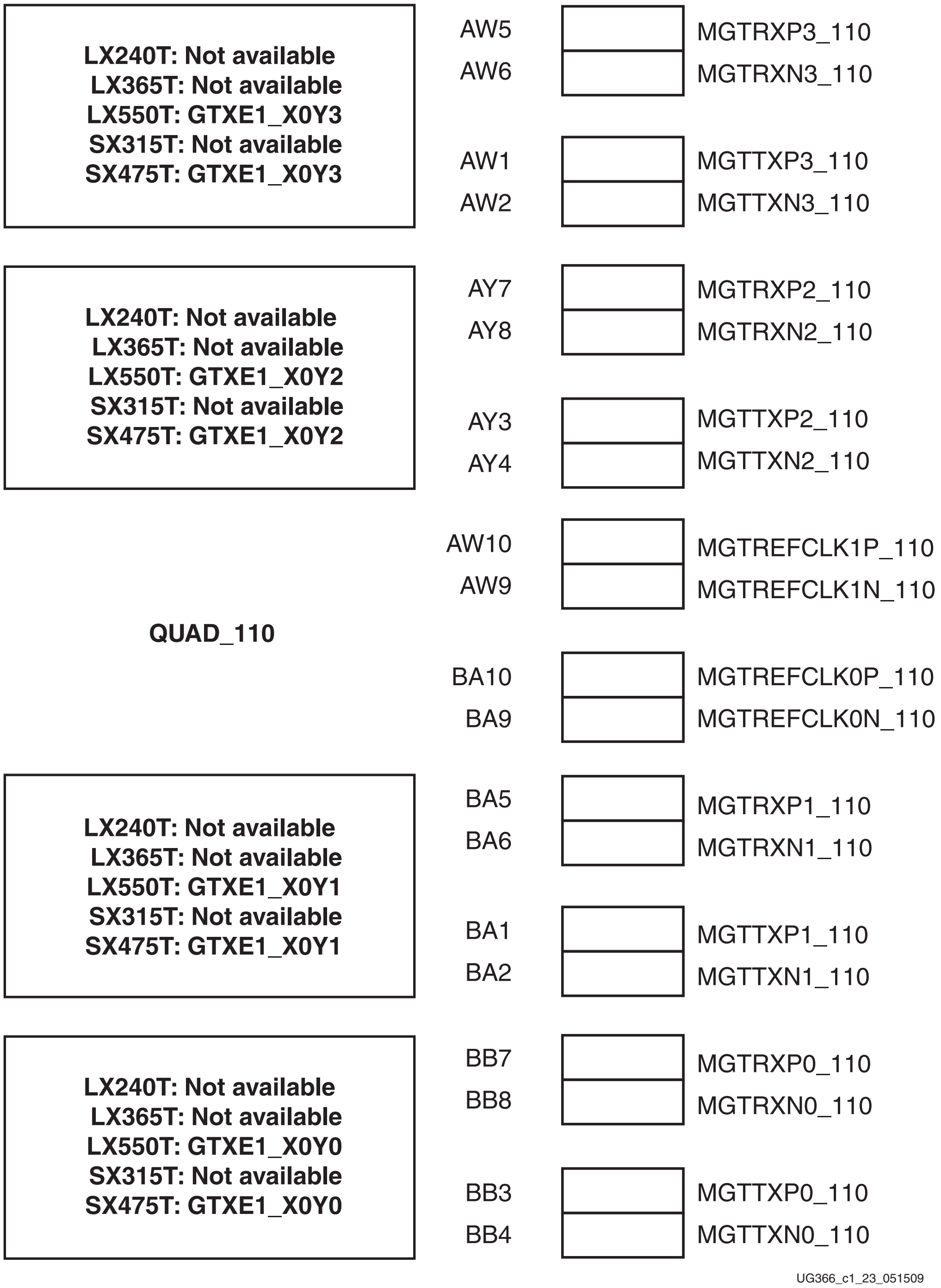


Figure 1-23: **Placement Diagram for the FF1759 Package (9 of 9)**

FF1154 Package Placement Diagrams

Figure 1-24 through Figure 1-35 show the placement diagrams for the FF1154 package.

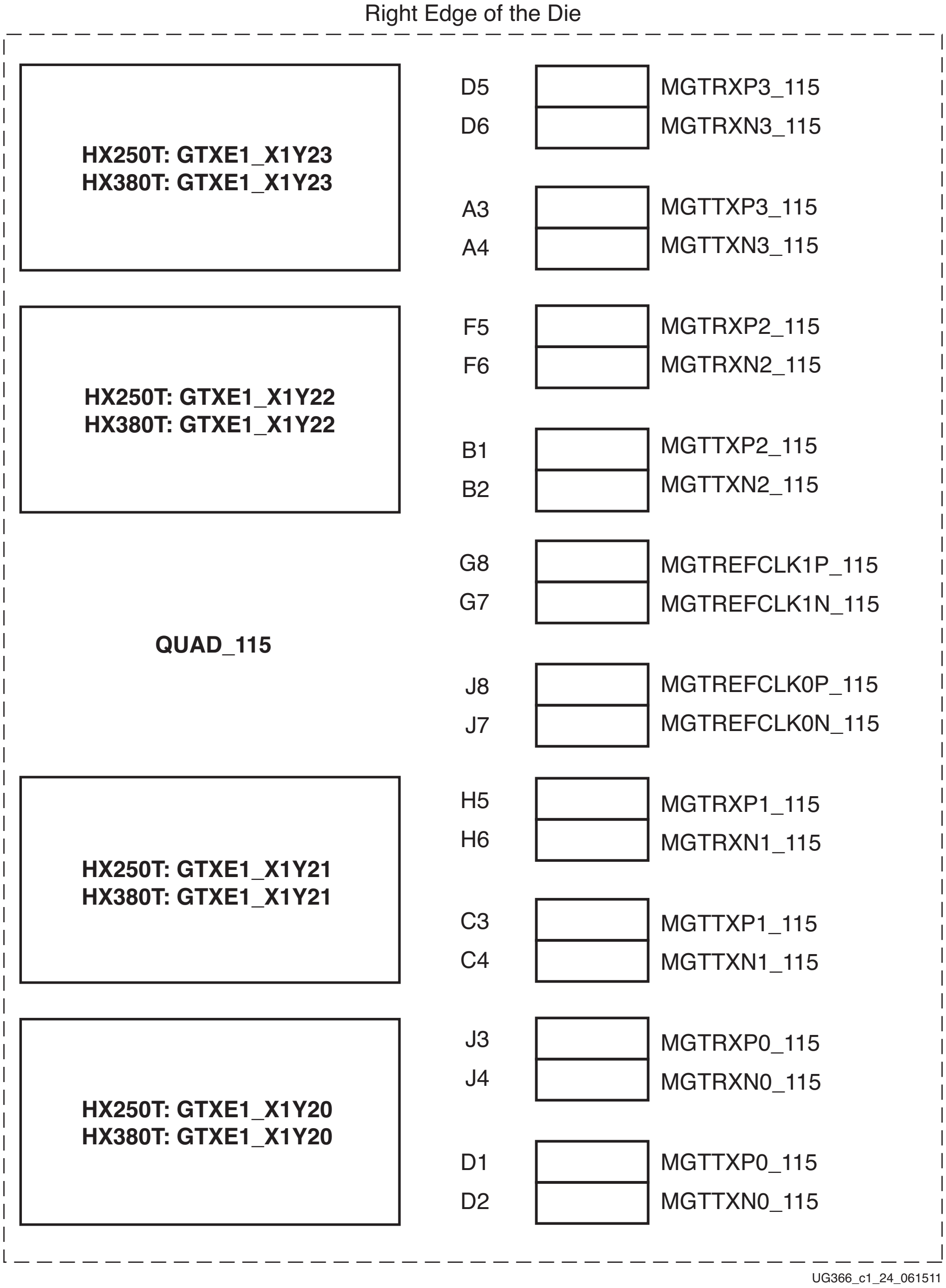


Figure 1-24: **Placement Diagram for the FF1154 Package (1 of 12)**

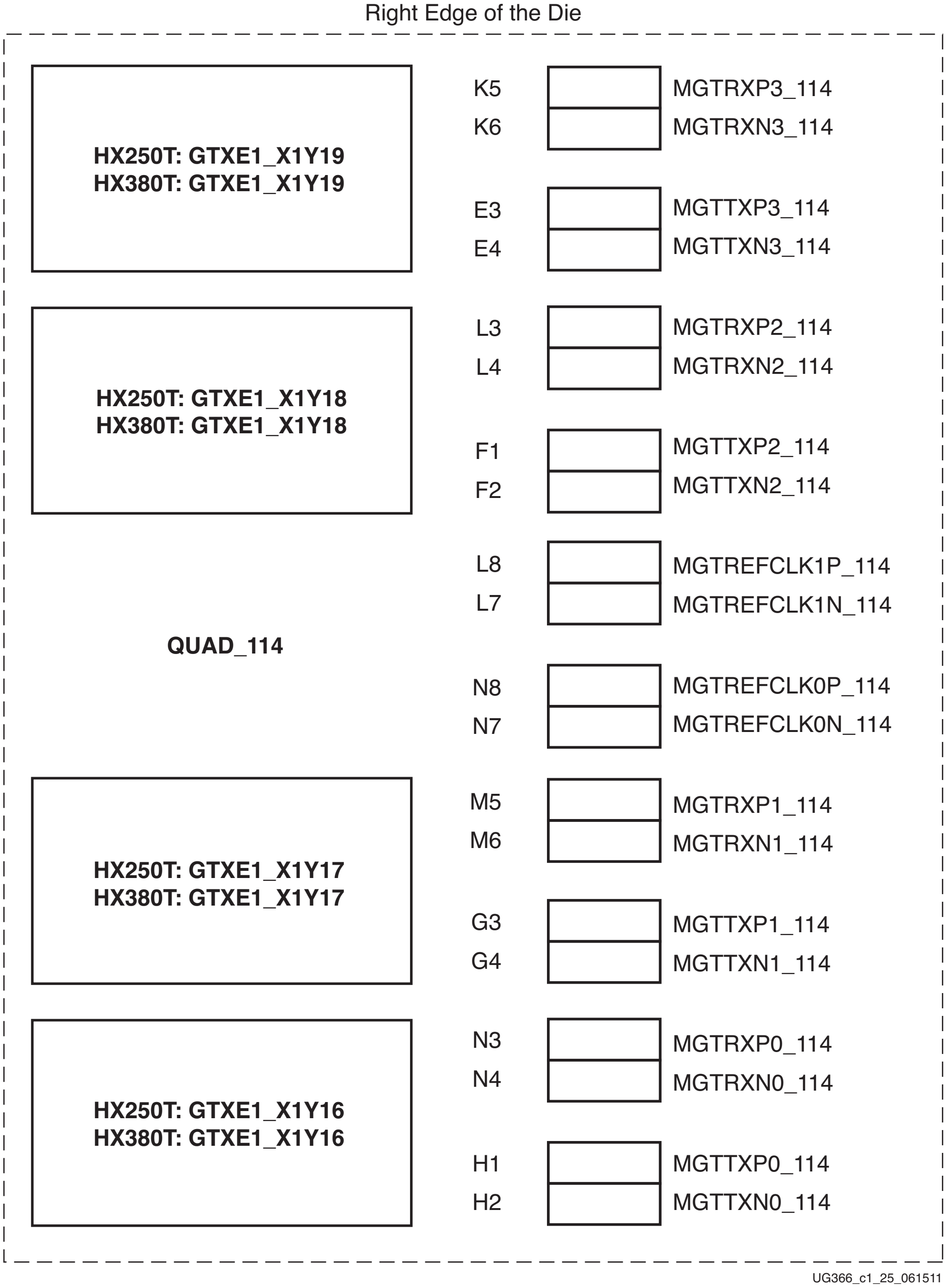


Figure 1-25: **Placement Diagram for the FF1154 Package (2 of 12)**

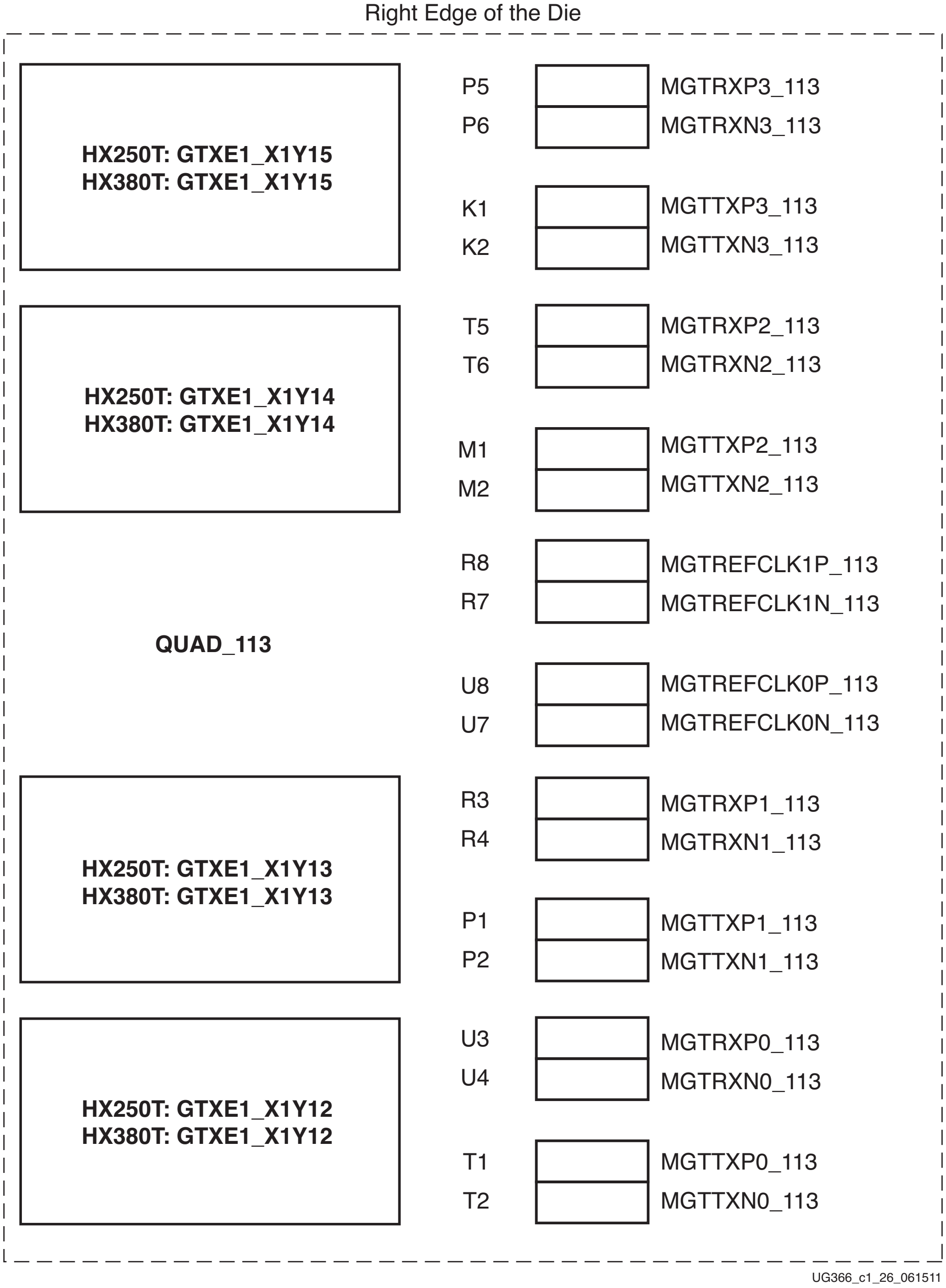


Figure 1-26: **Placement Diagram for the FF1154 Package (3 of 12)**

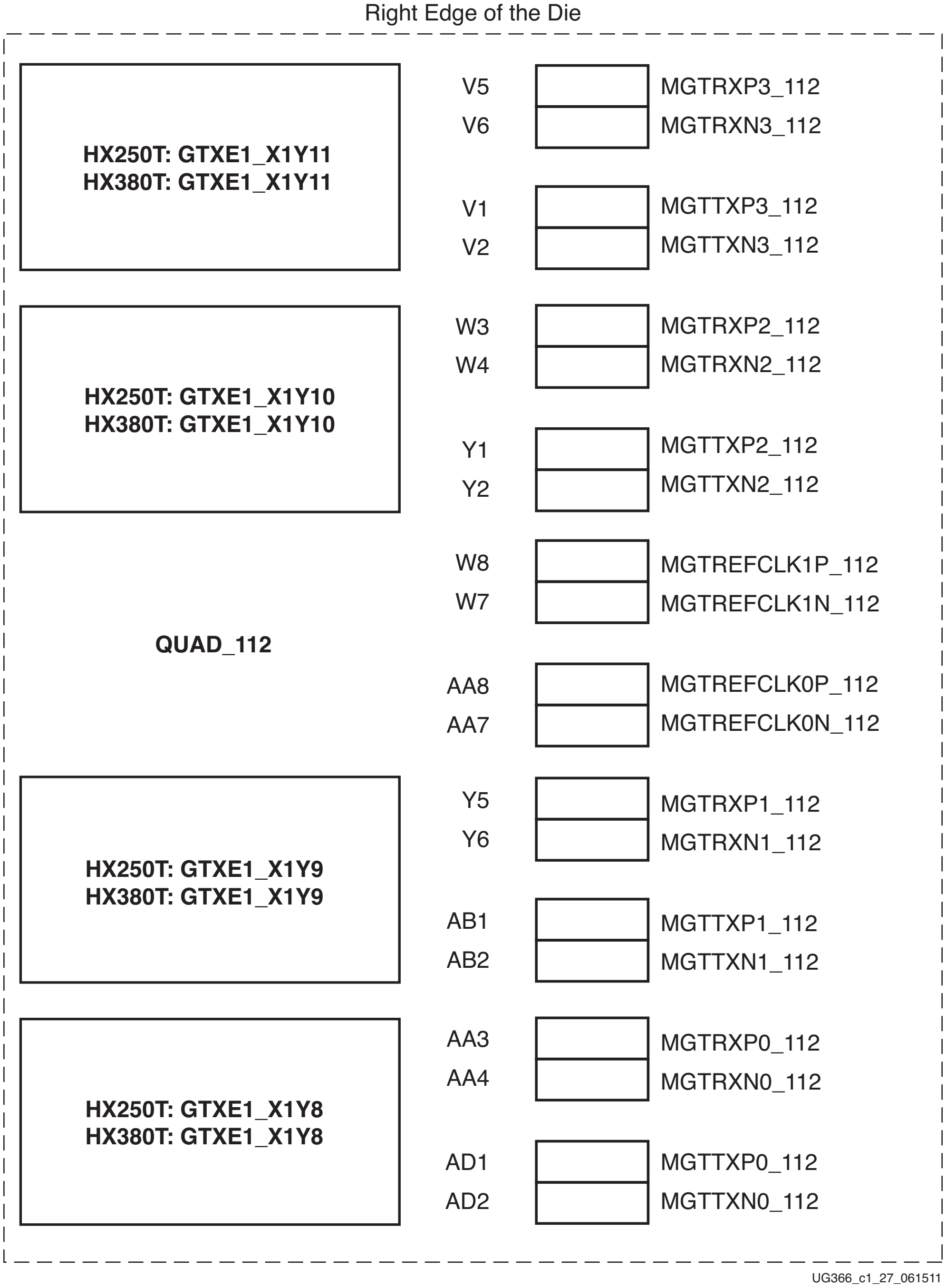


Figure 1-27: **Placement Diagram for the FF1154 Package (4 of 12)**

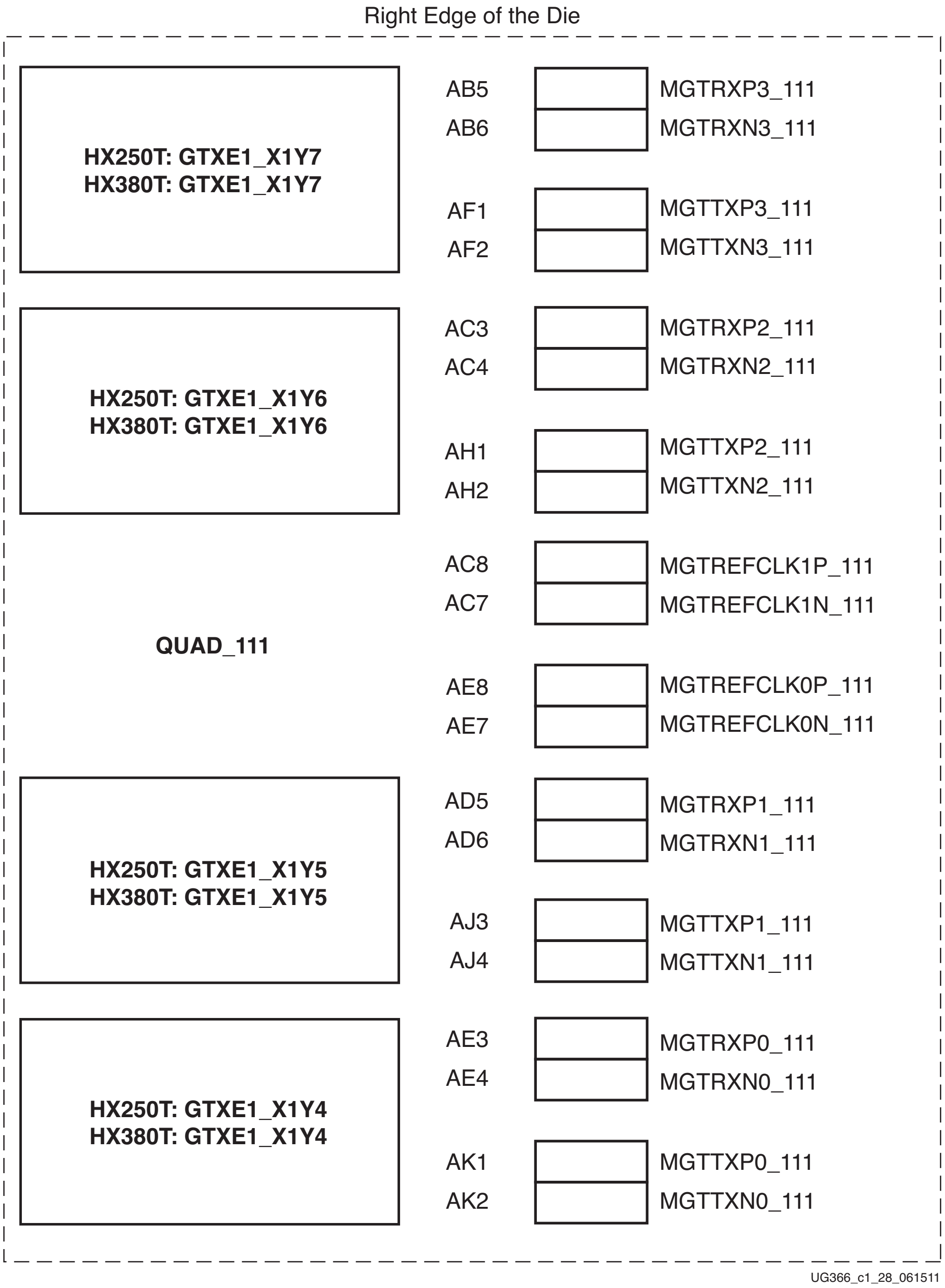


Figure 1-28: **Placement Diagram for the FF1154 Package (5 of 12)**

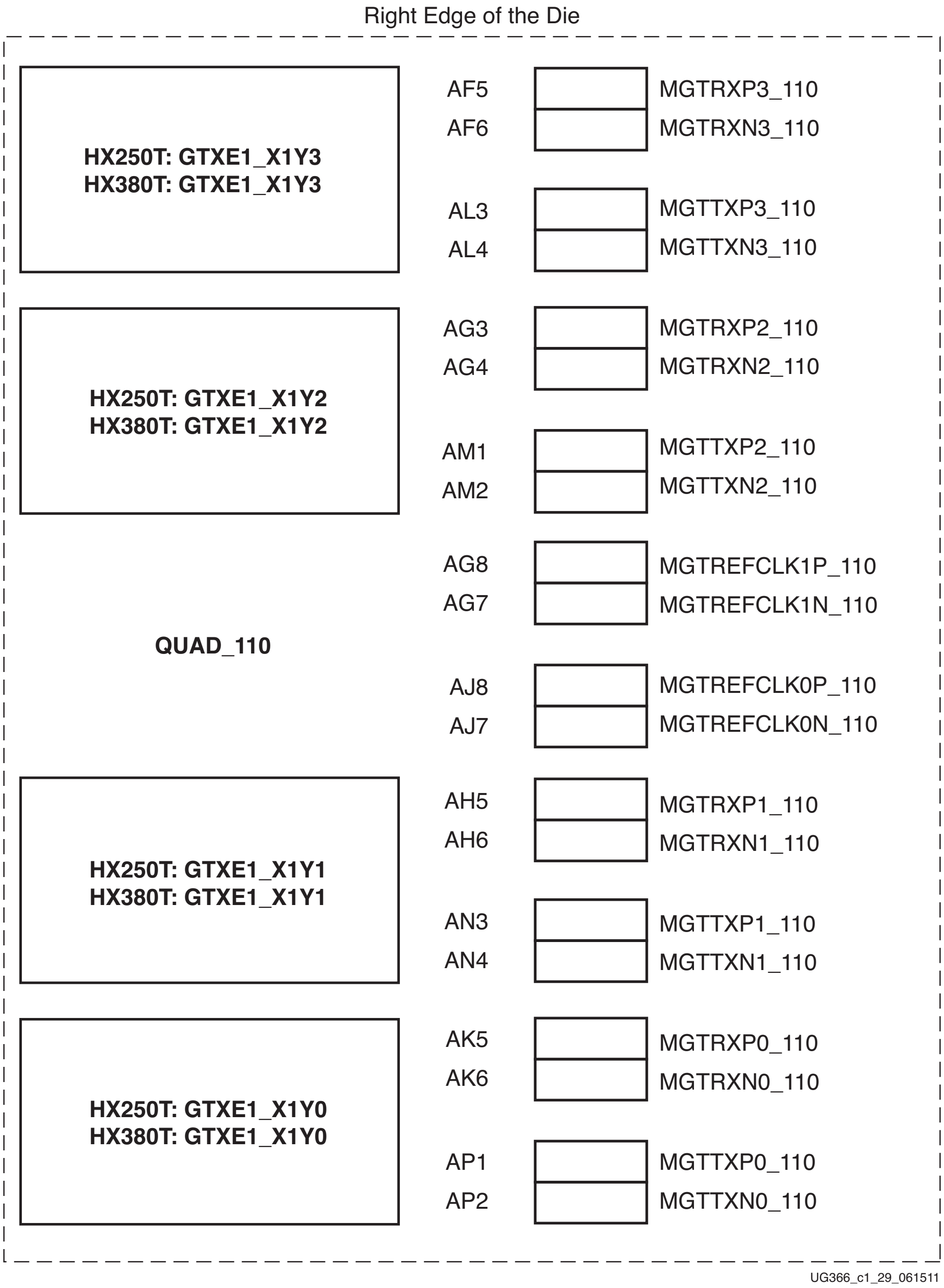


Figure 1-29: **Placement Diagram for the FF1154 Package (6 of 12)**

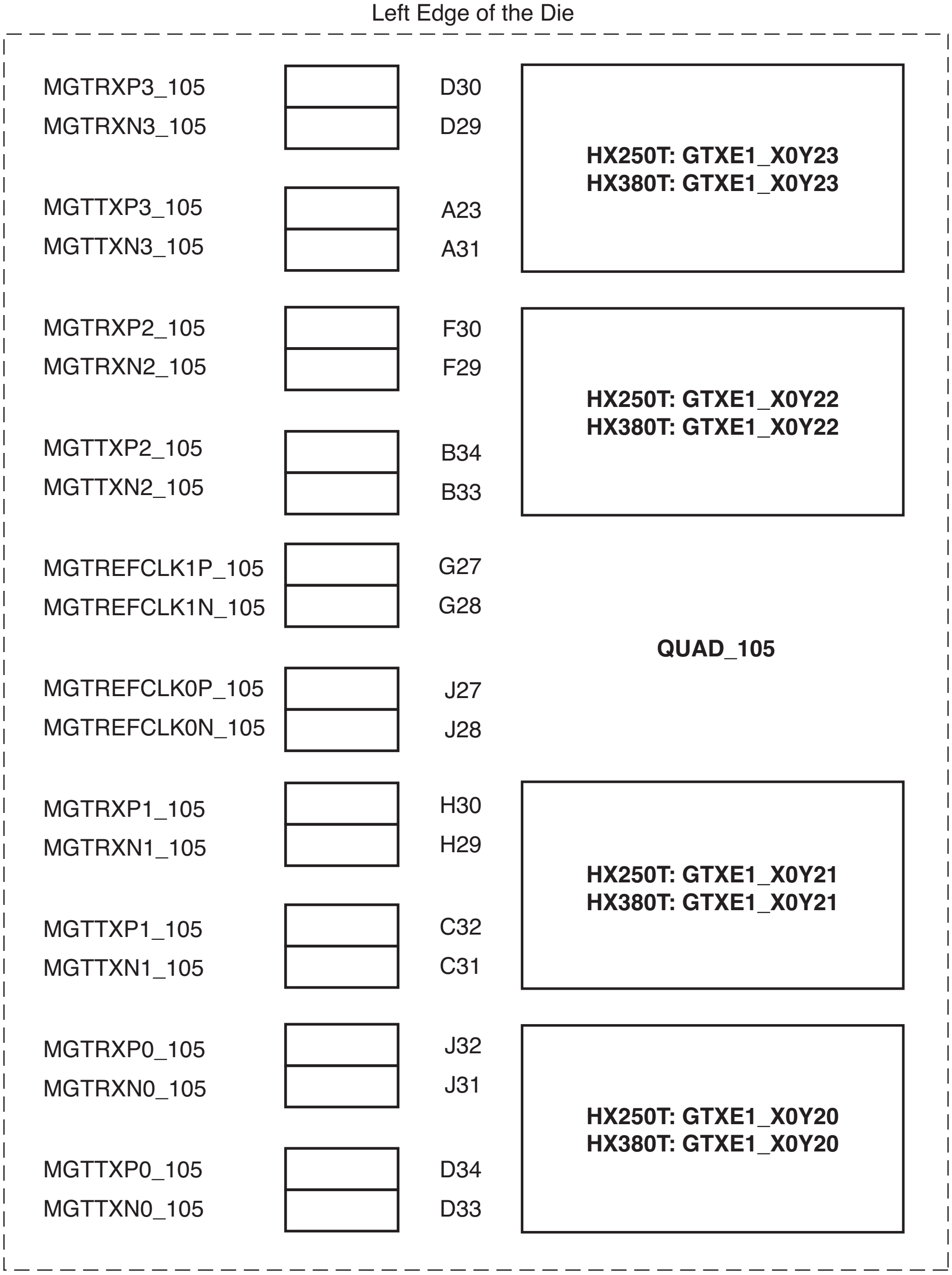


Figure 1-30: **Placement Diagram for the FF1154 Package (7 of 12)**

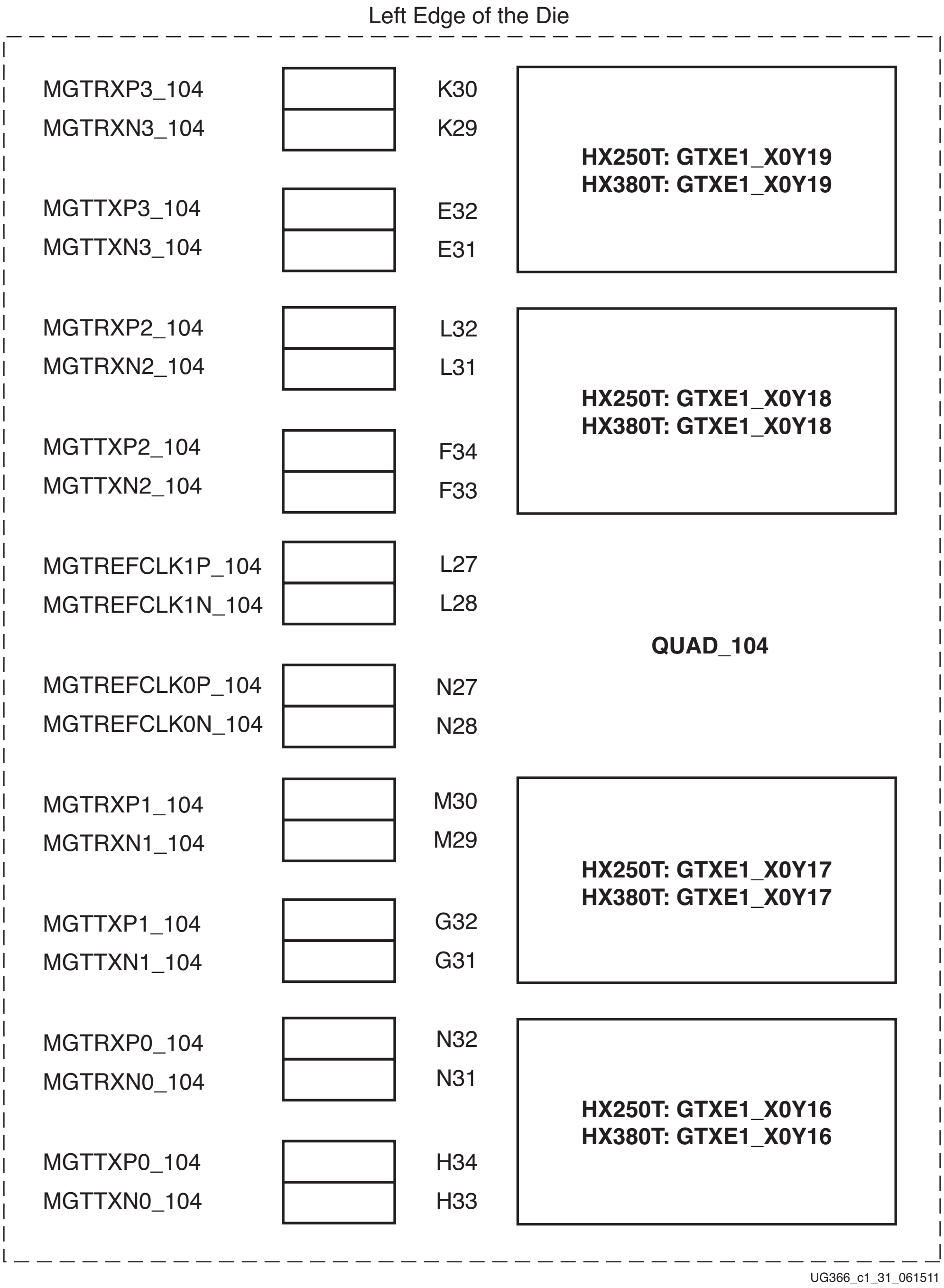


Figure 1-31: **Placement Diagram for the FF1154 Package (8 of 12)**

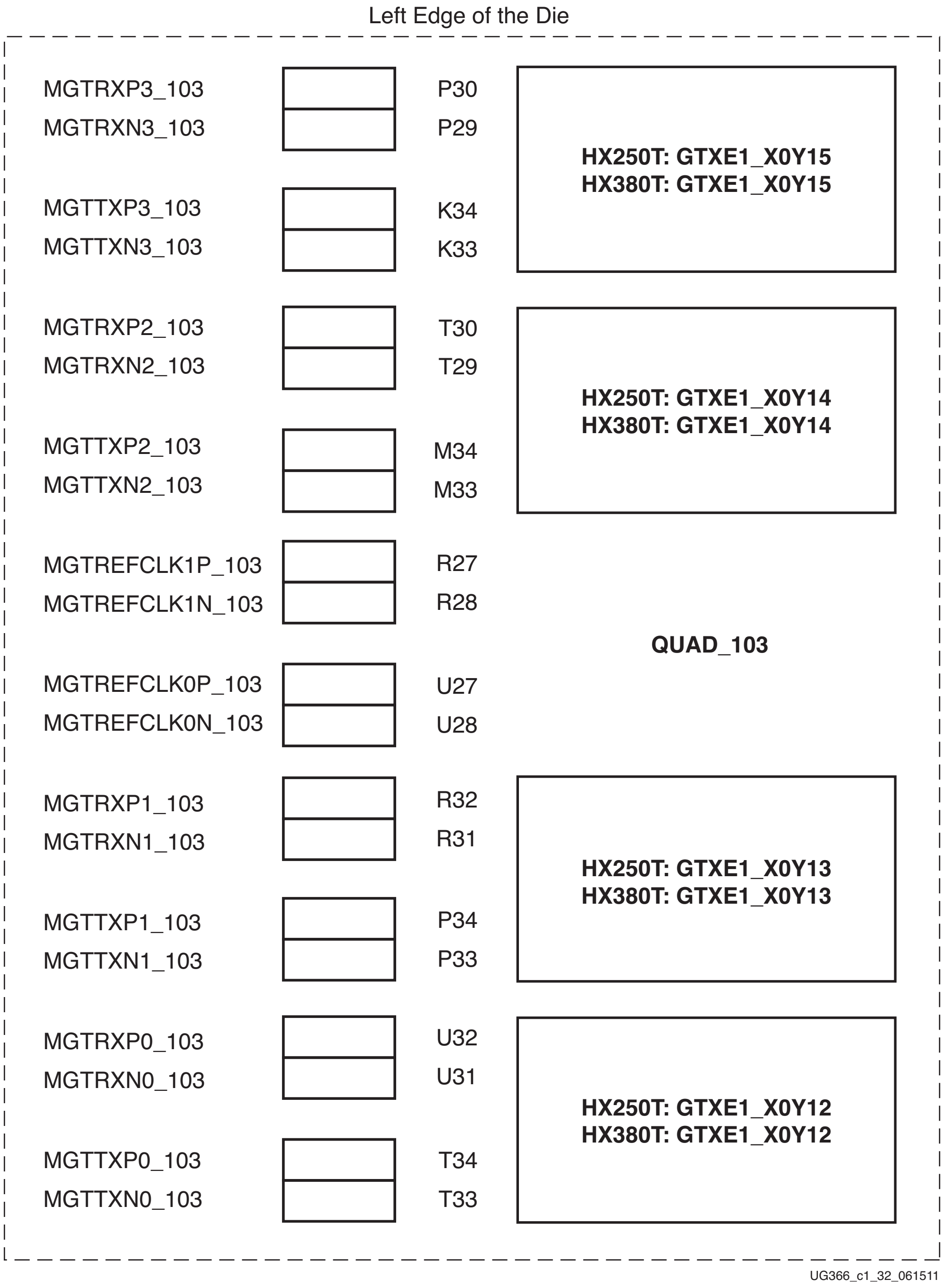


Figure 1-32: **Placement Diagram for the FF1154 Package (9 of 12)**

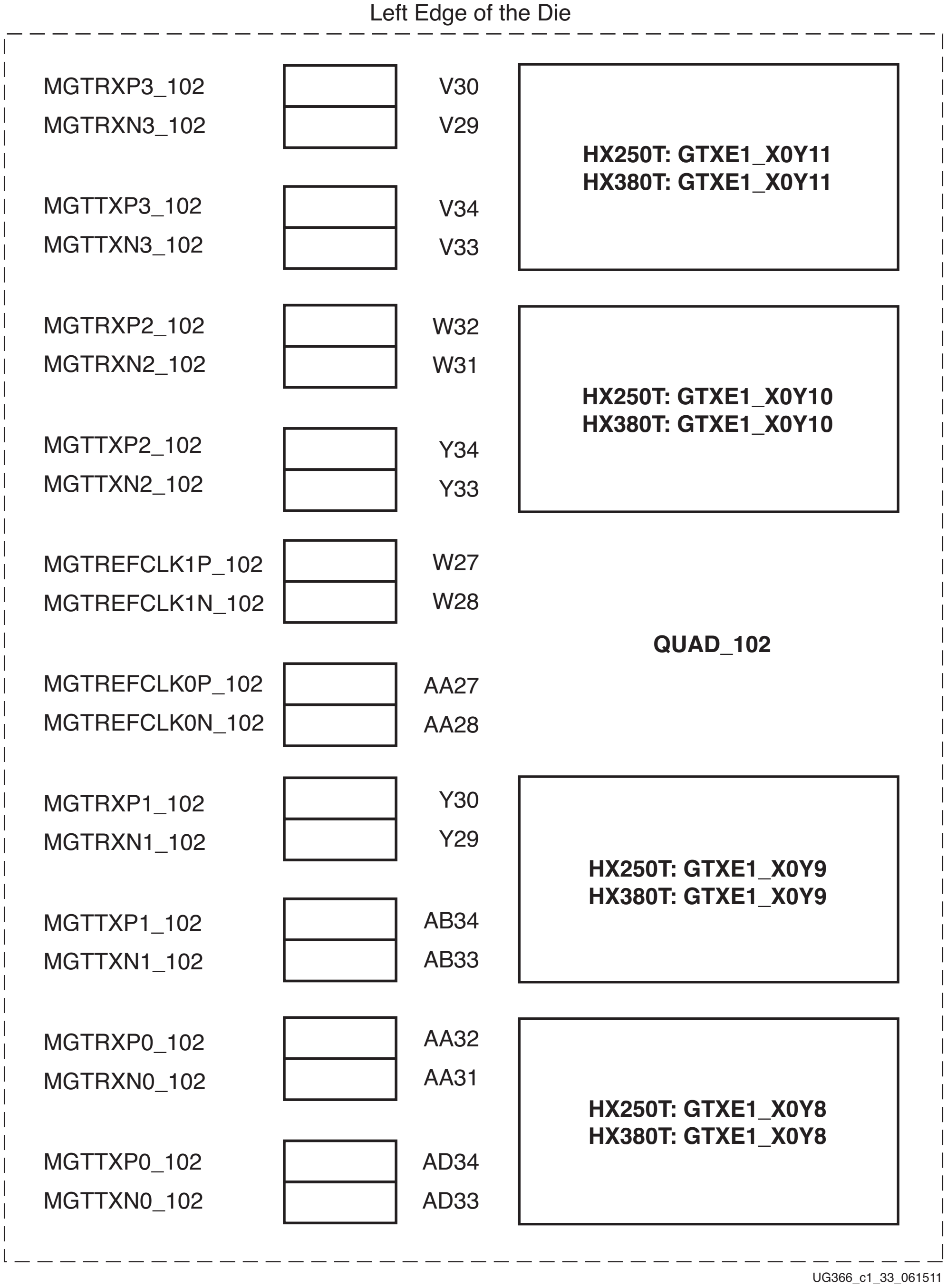
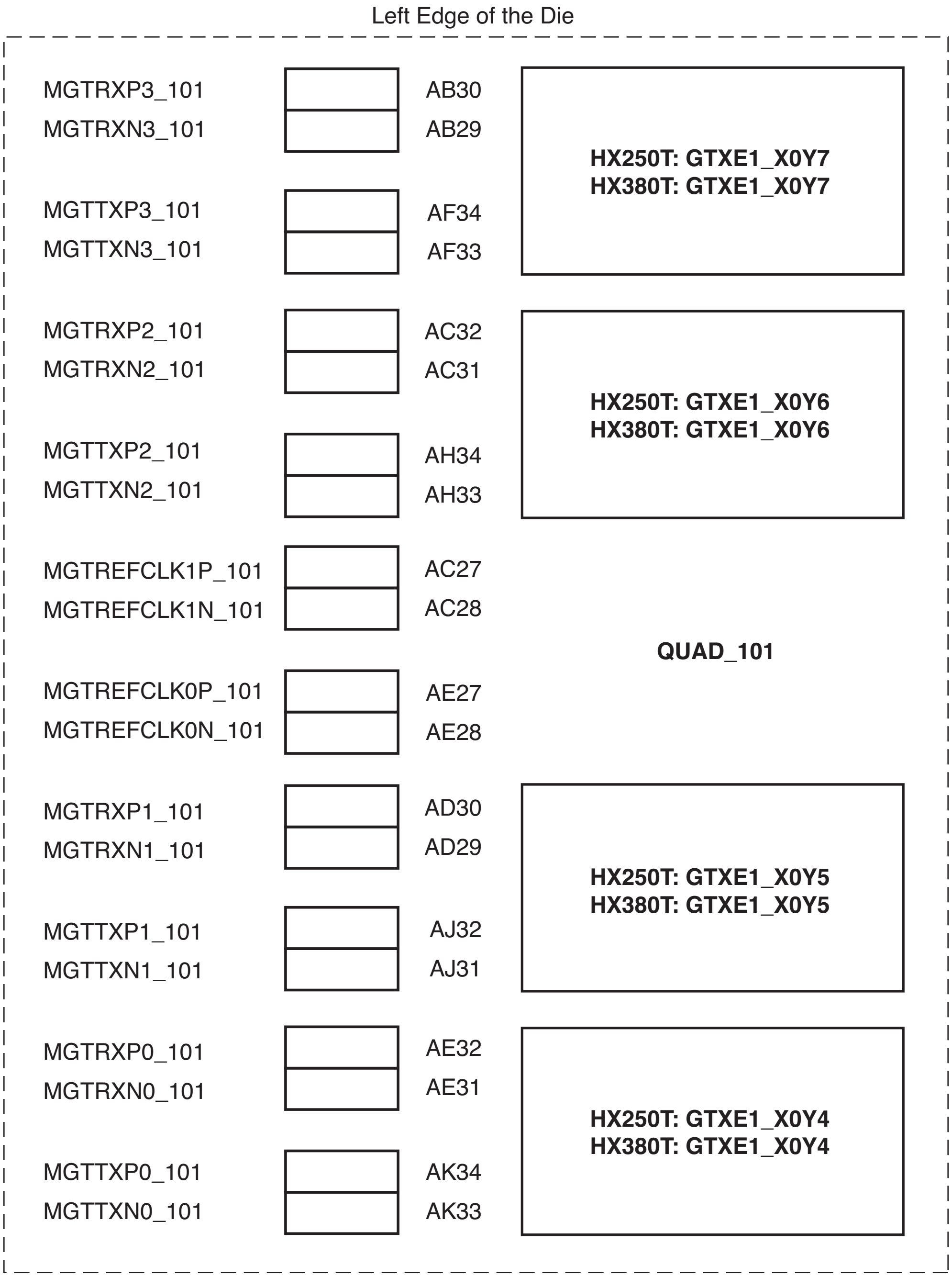


Figure 1-33: **Placement Diagram for the FF1154 Package (10 of 12)**



UG366_c1_34_061511

Figure 1-34: **Placement Diagram for the FF1154 Package (11 of 12)**

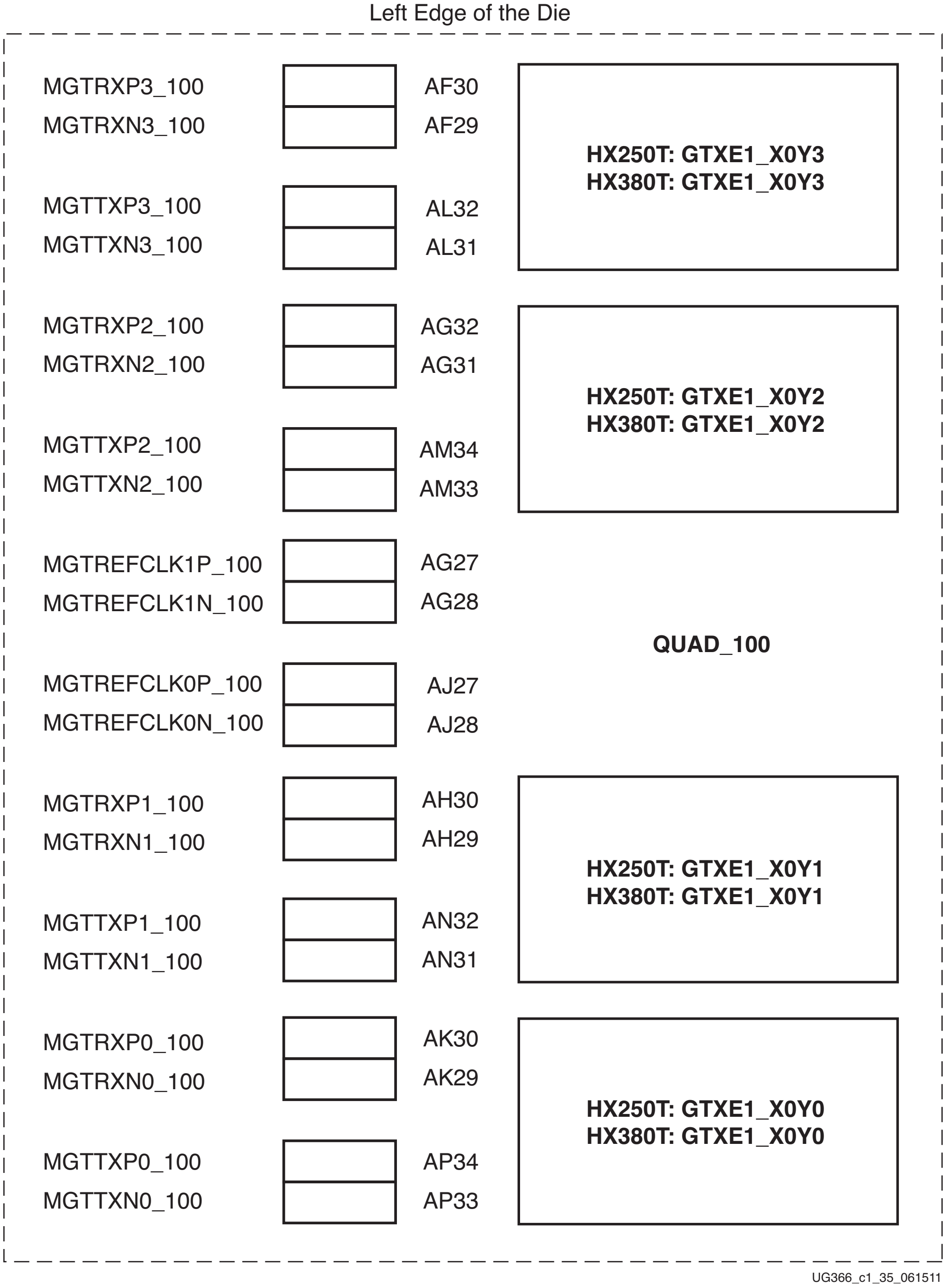


Figure 1-35: **Placement Diagram for the FF1154 Package (12 of 12)**

FF1155 Package Placement Diagrams

Figure 1-36 through Figure 1-41 show the placement diagrams for the FF1155 package.

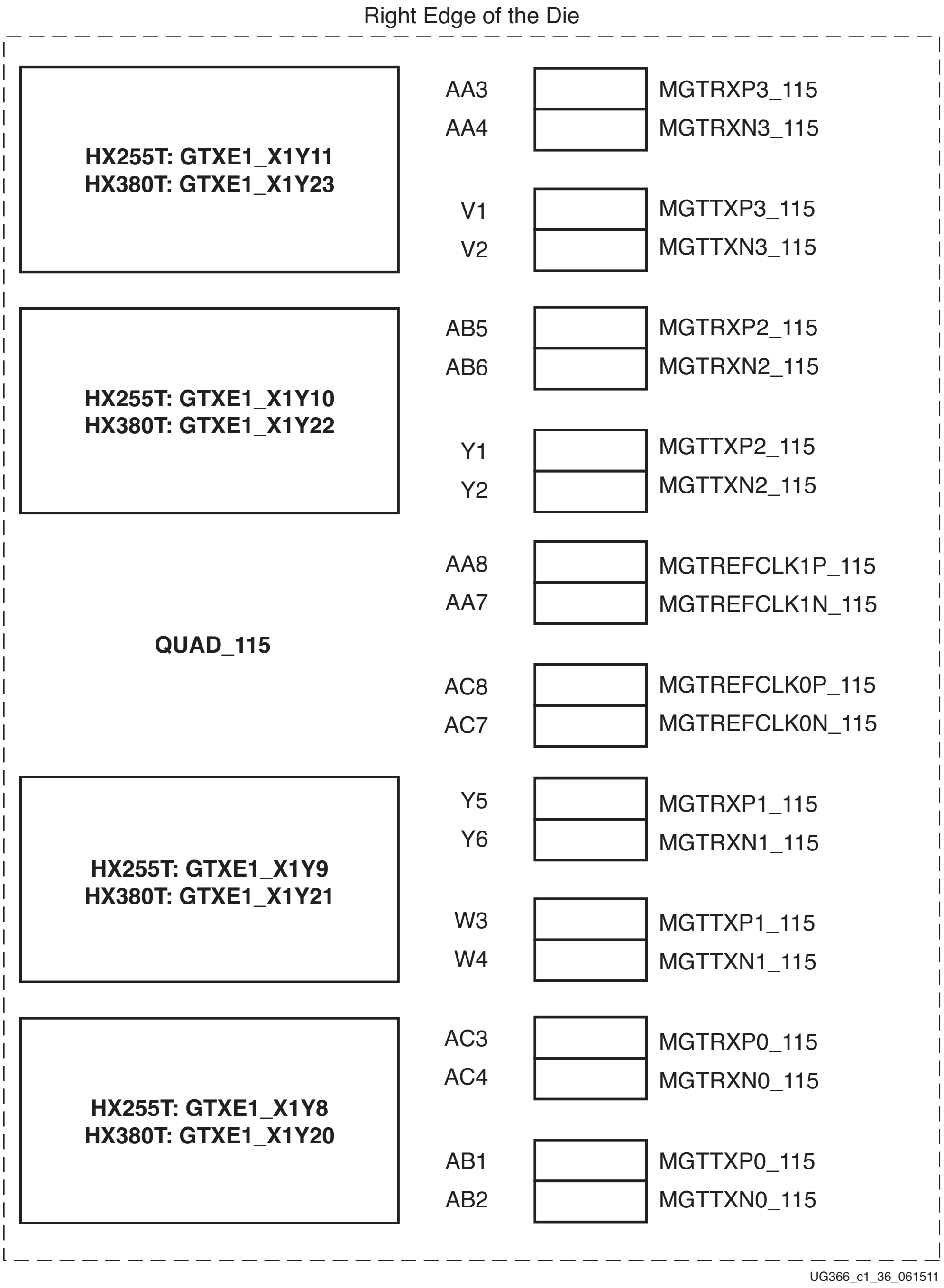


Figure 1-36: **Placement Diagram for the FF1155 Package (1 of 6)**

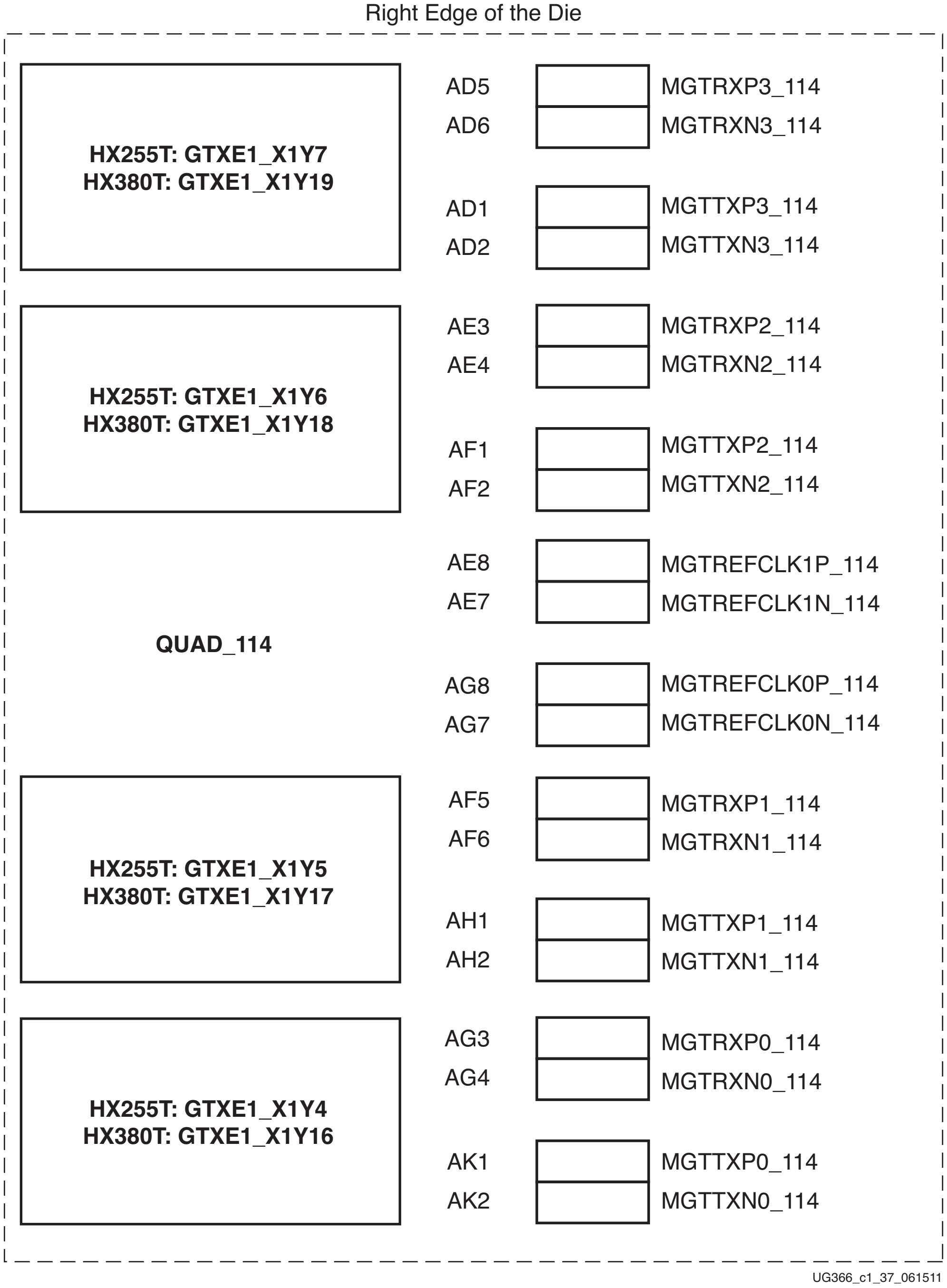


Figure 1-37: **Placement Diagram for the FF1155 Package (2 of 6)**

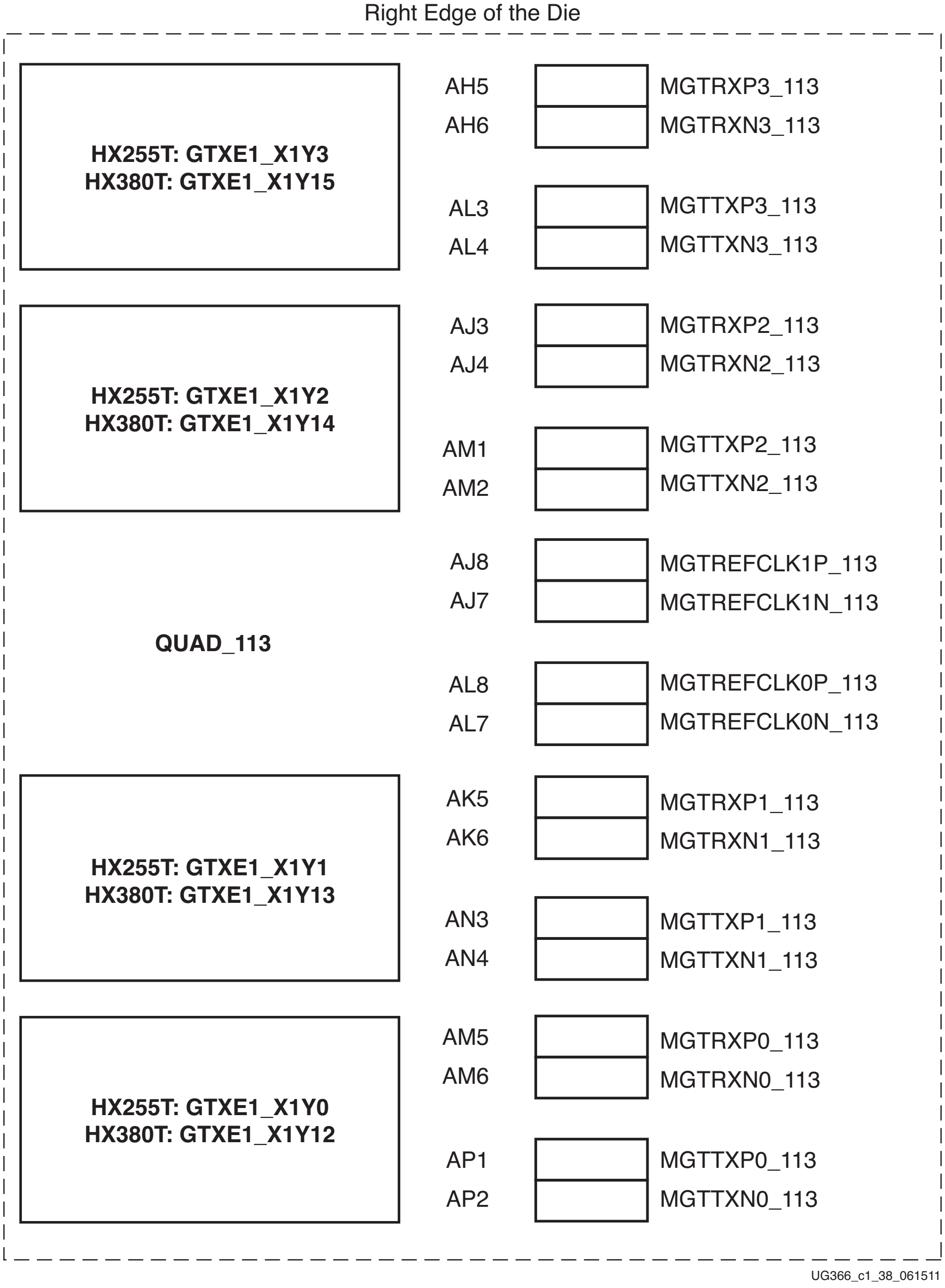


Figure 1-38: **Placement Diagram for the FF1155 Package (3 of 6)**

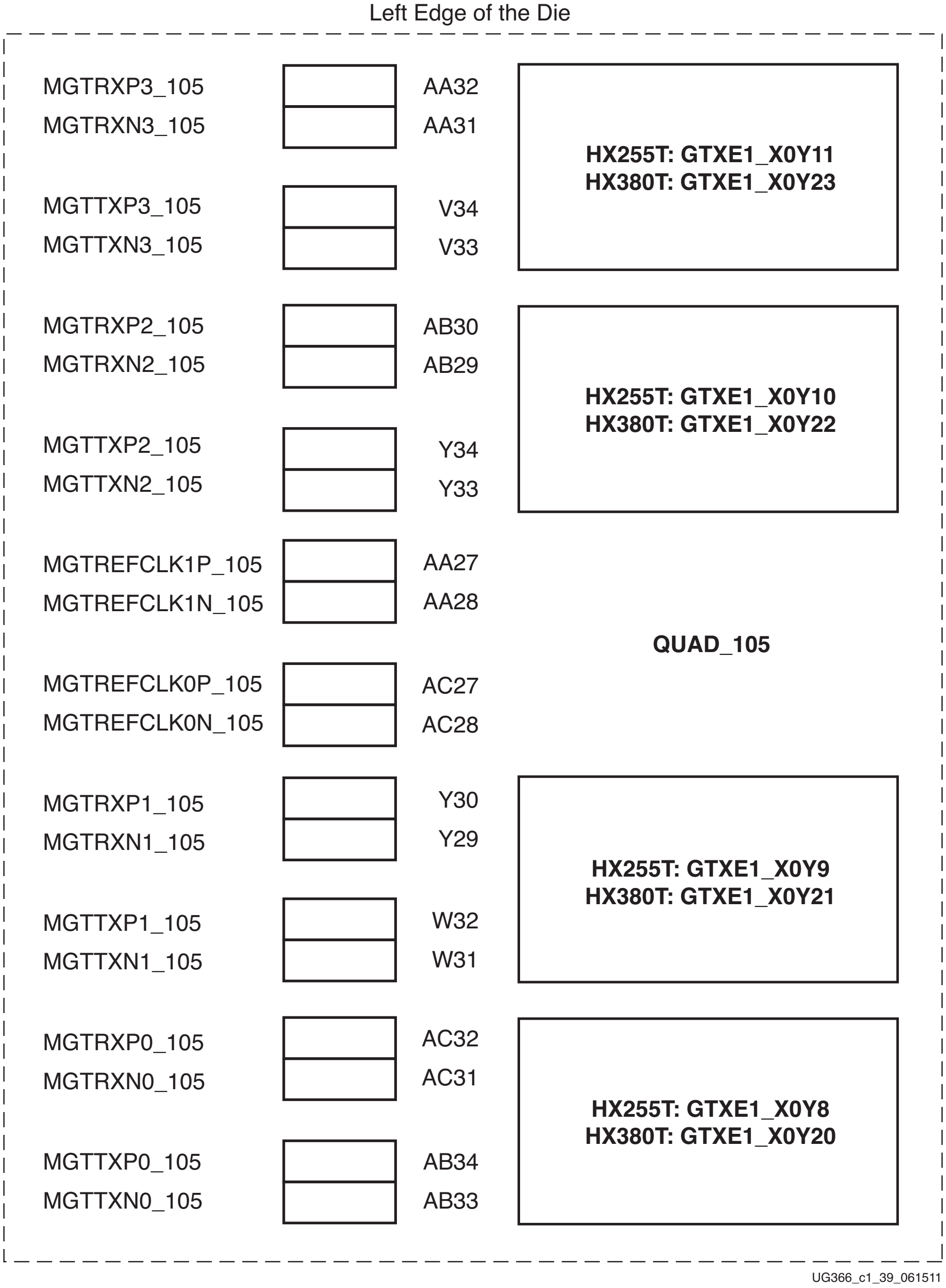


Figure 1-39: **Placement Diagram for the FF1155 Package (4 of 6)**

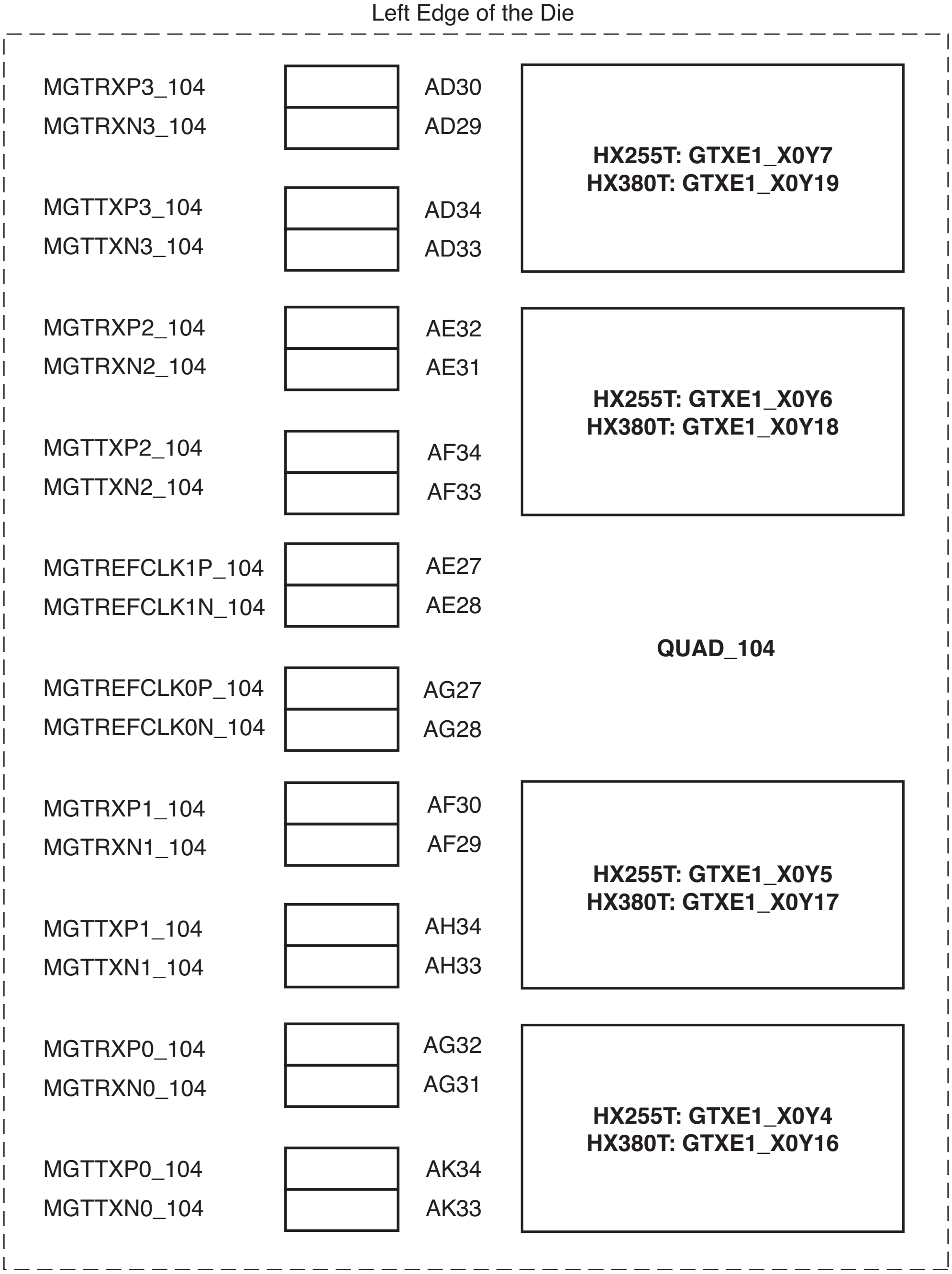


Figure 1-40: **Placement Diagram for the FF1155 Package (5 of 6)**

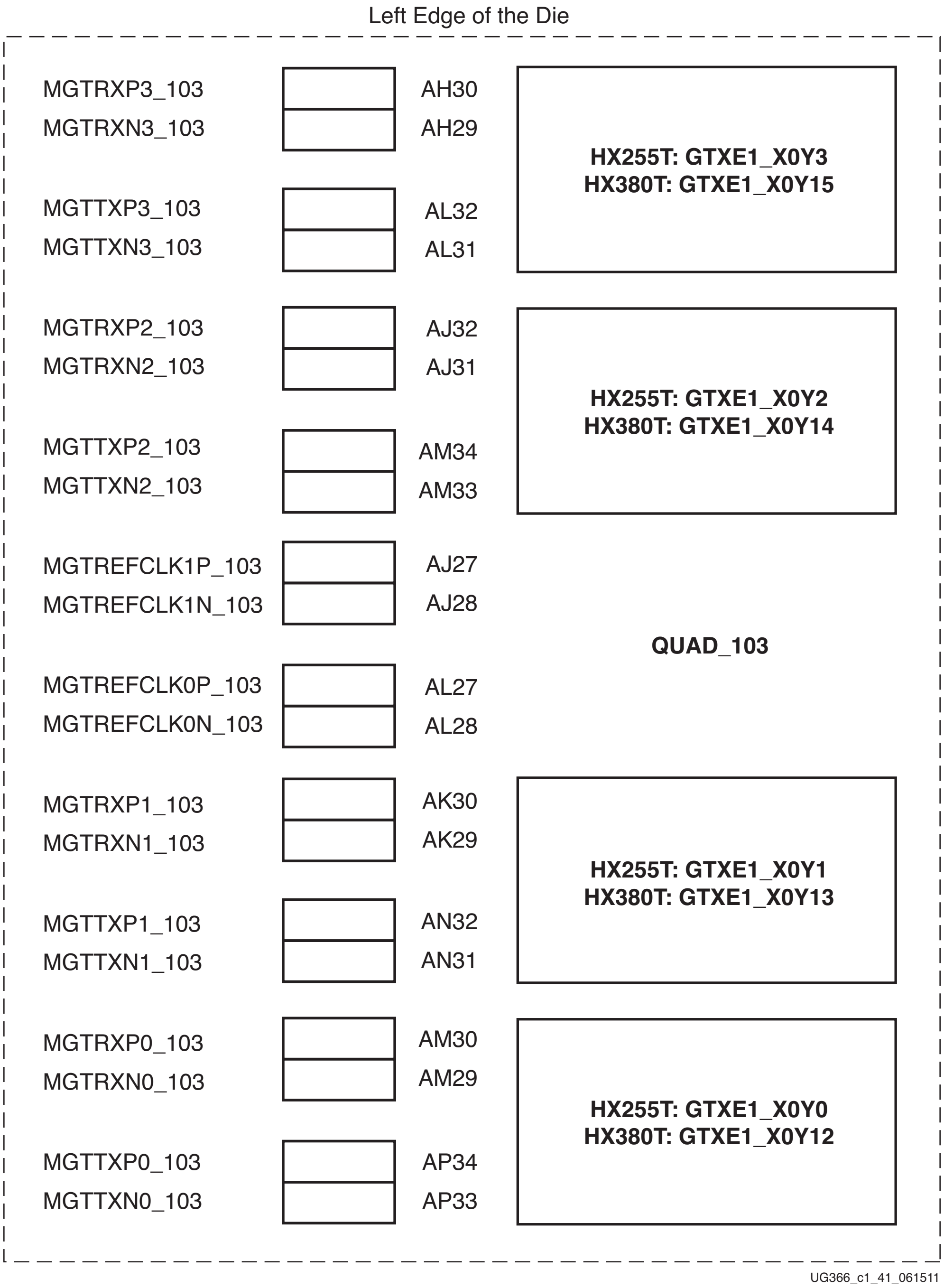


Figure 1-41: **Placement Diagram for the FF1155 Package (6 of 6)**

FF1923 Package Placement Diagrams

Figure 1-42 through Figure 1-51 show the placement diagrams for the FF1923 package.

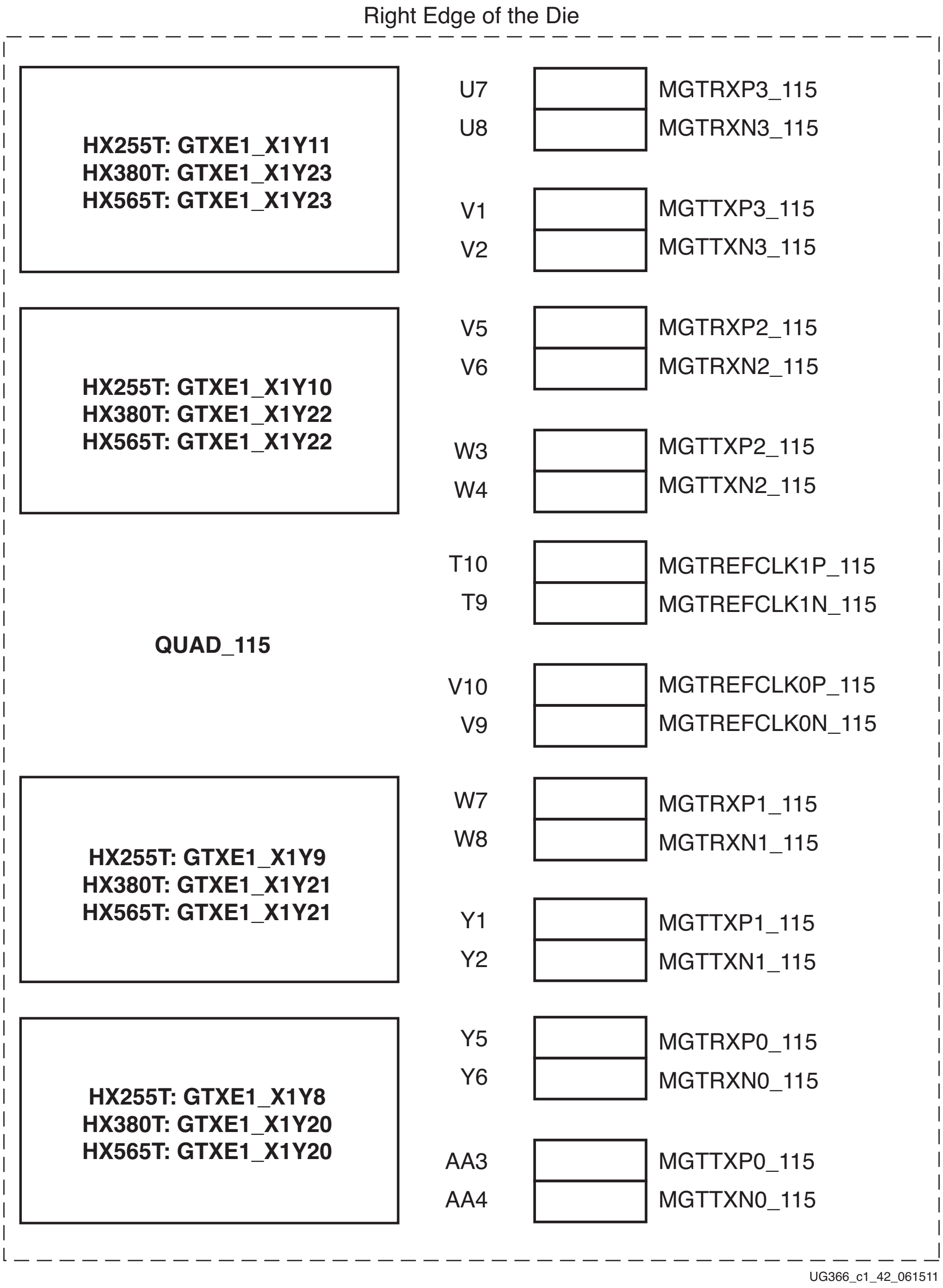


Figure 1-42: **Placement Diagram for the FF1923 Package (1 of 10)**

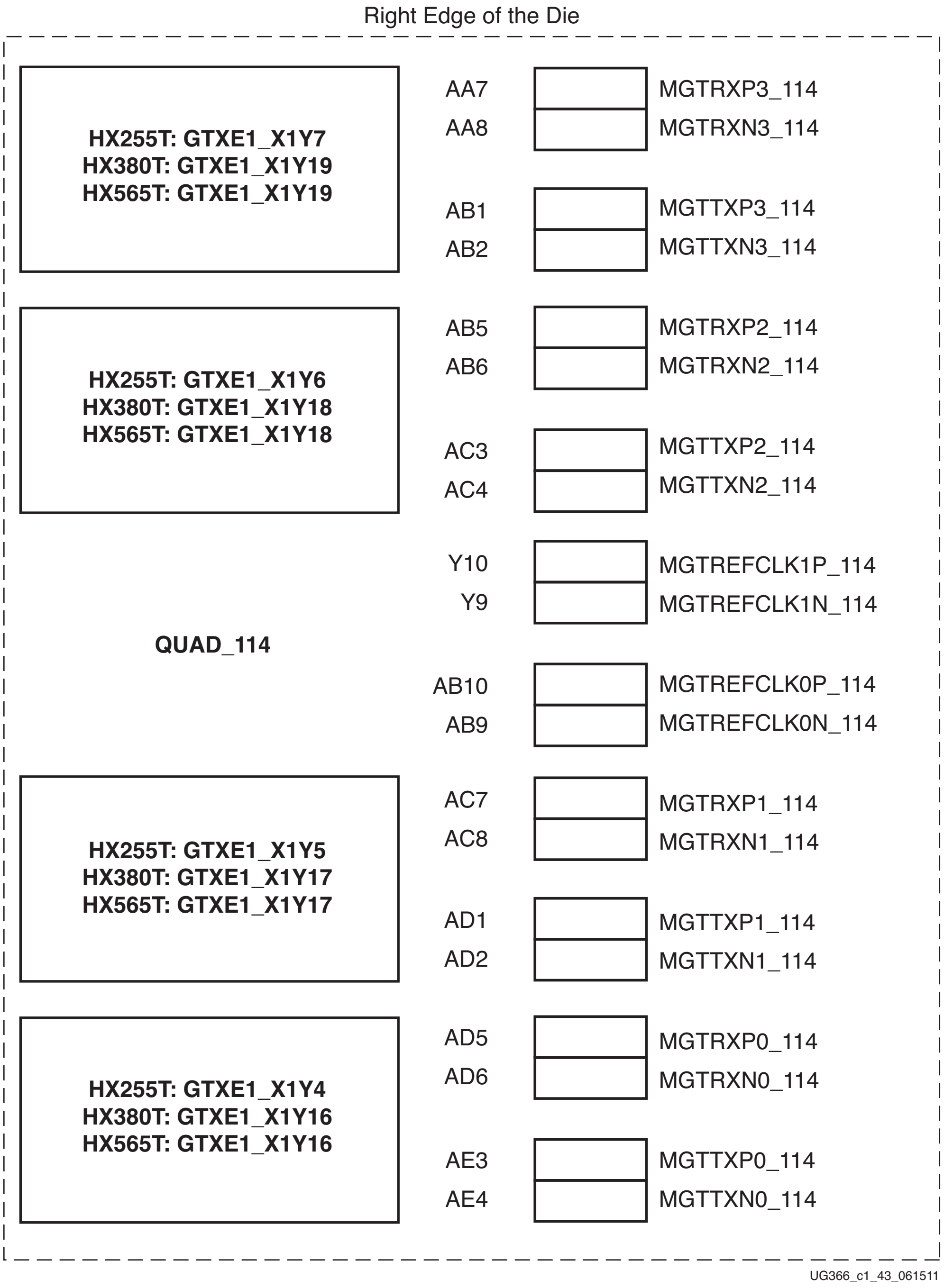


Figure 1-43: **Placement Diagram for the FF1923 Package (2 of 10)**

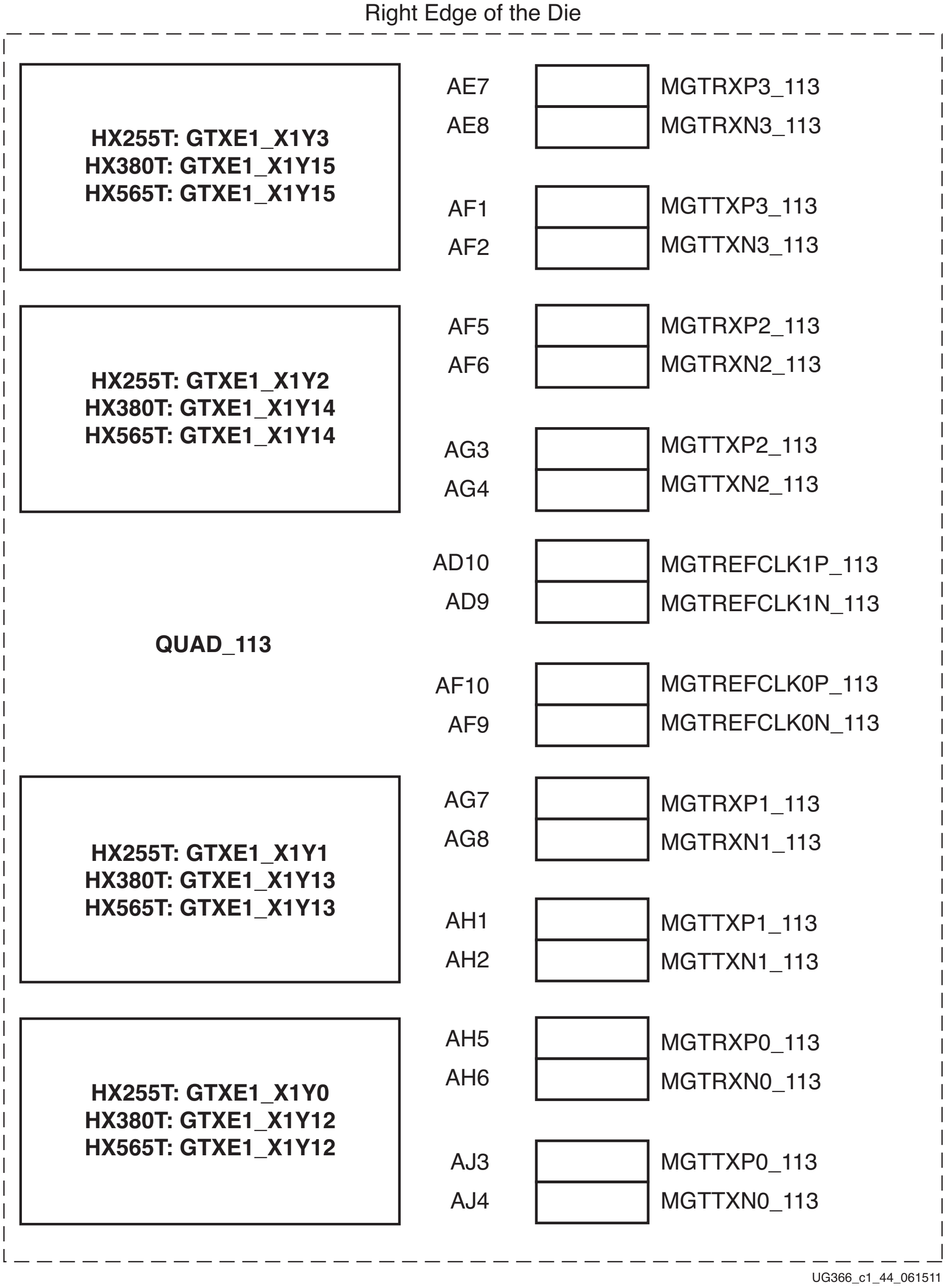


Figure 1-44: **Placement Diagram for the FF1923 Package (3 of 10)**

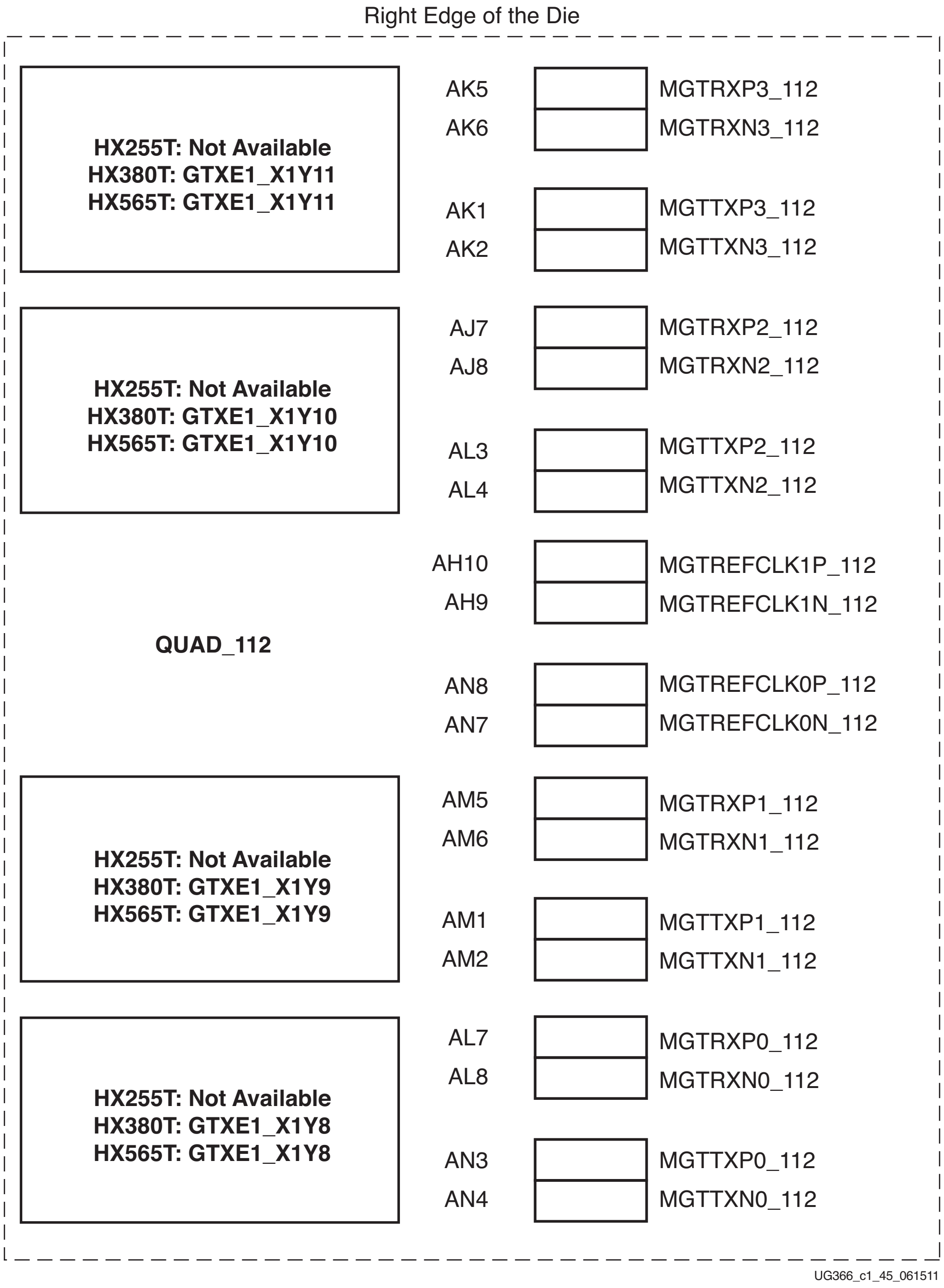


Figure 1-45: **Placement Diagram for the FF1923 Package (4 of 10)**

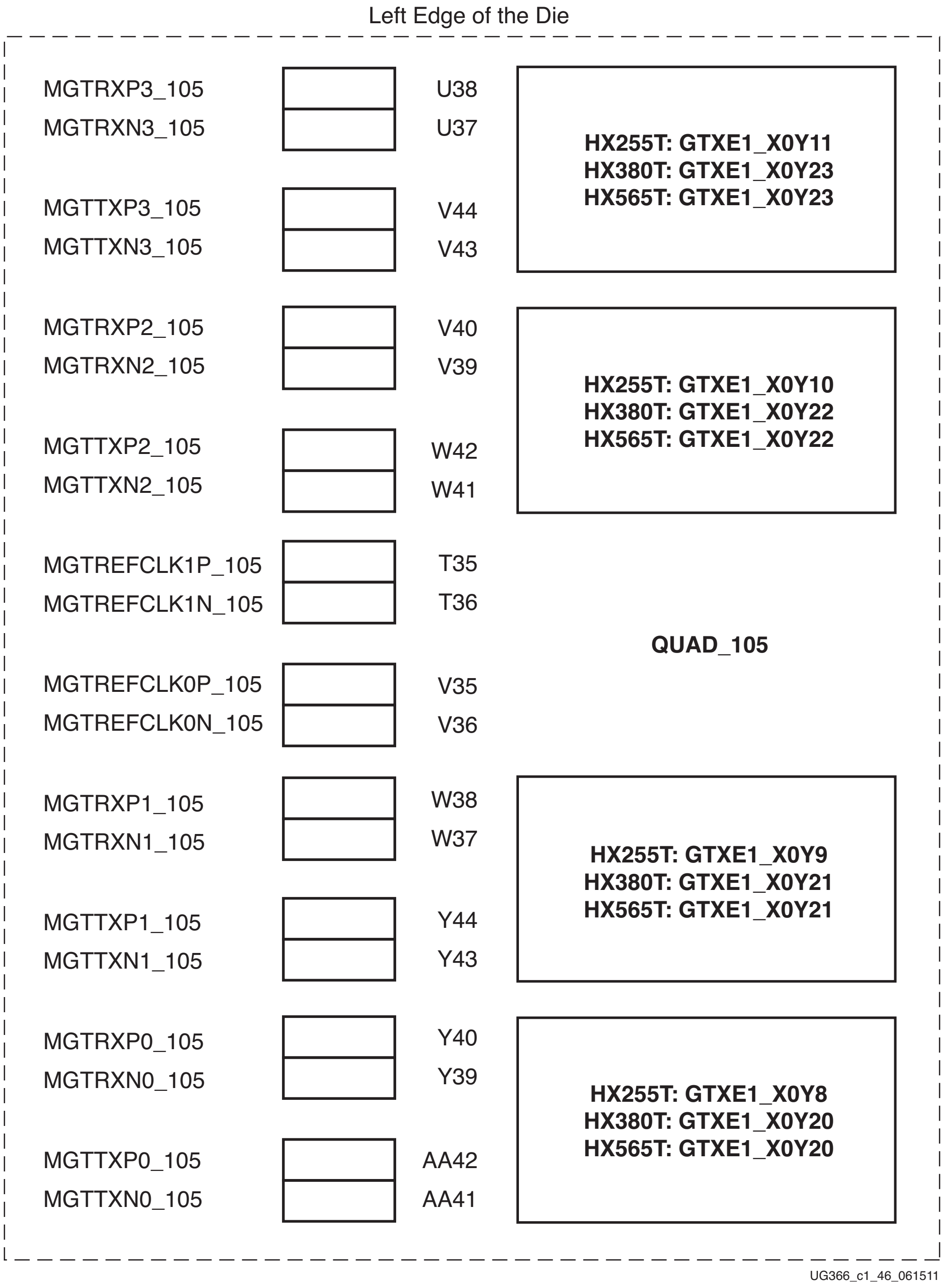
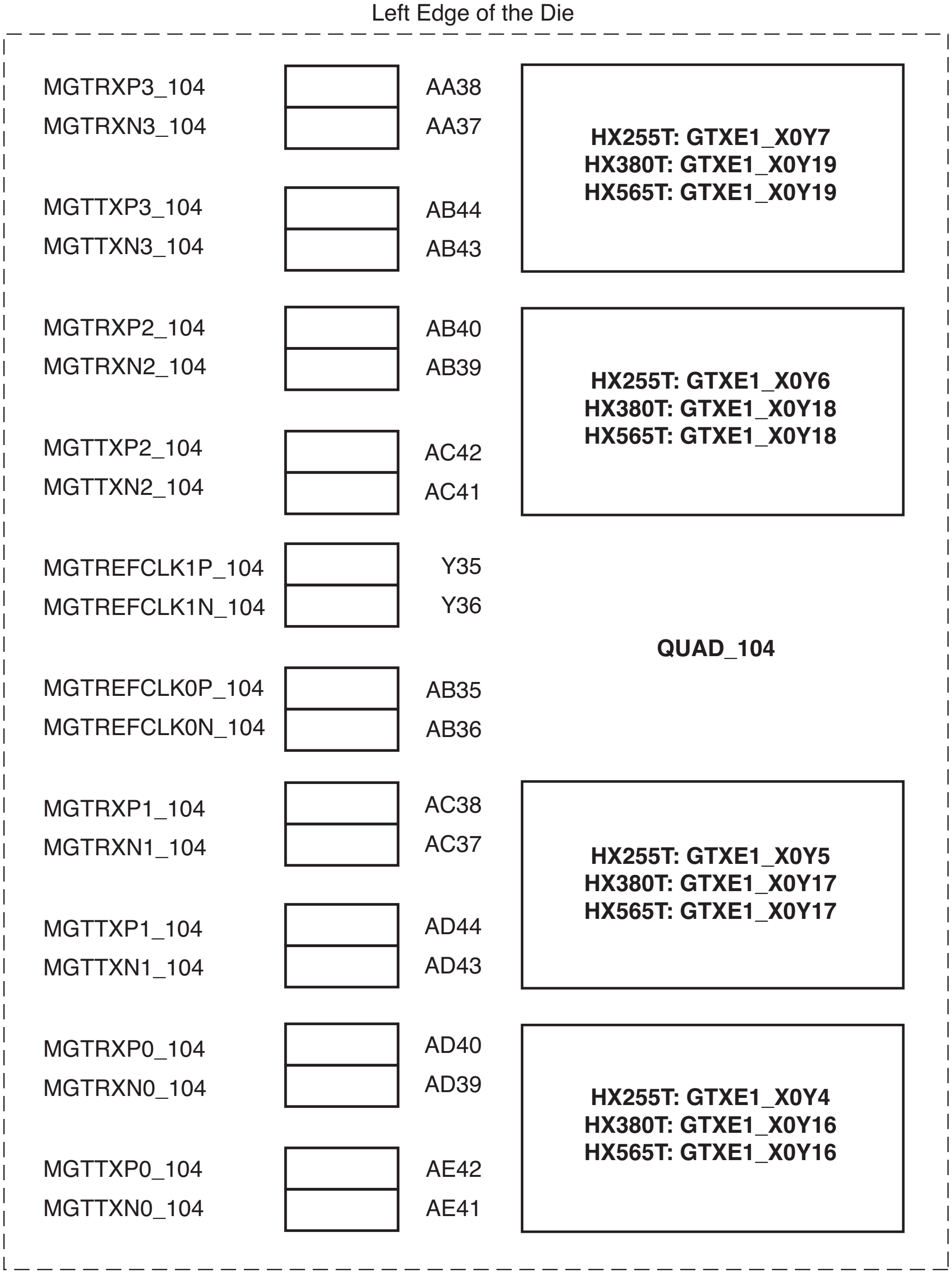


Figure 1-46: **Placement Diagram for the FF1923 Package (5 of 10)**



UG366_c1_47_061511

Figure 1-47: **Placement Diagram for the FF1923 Package (6 of 10)**

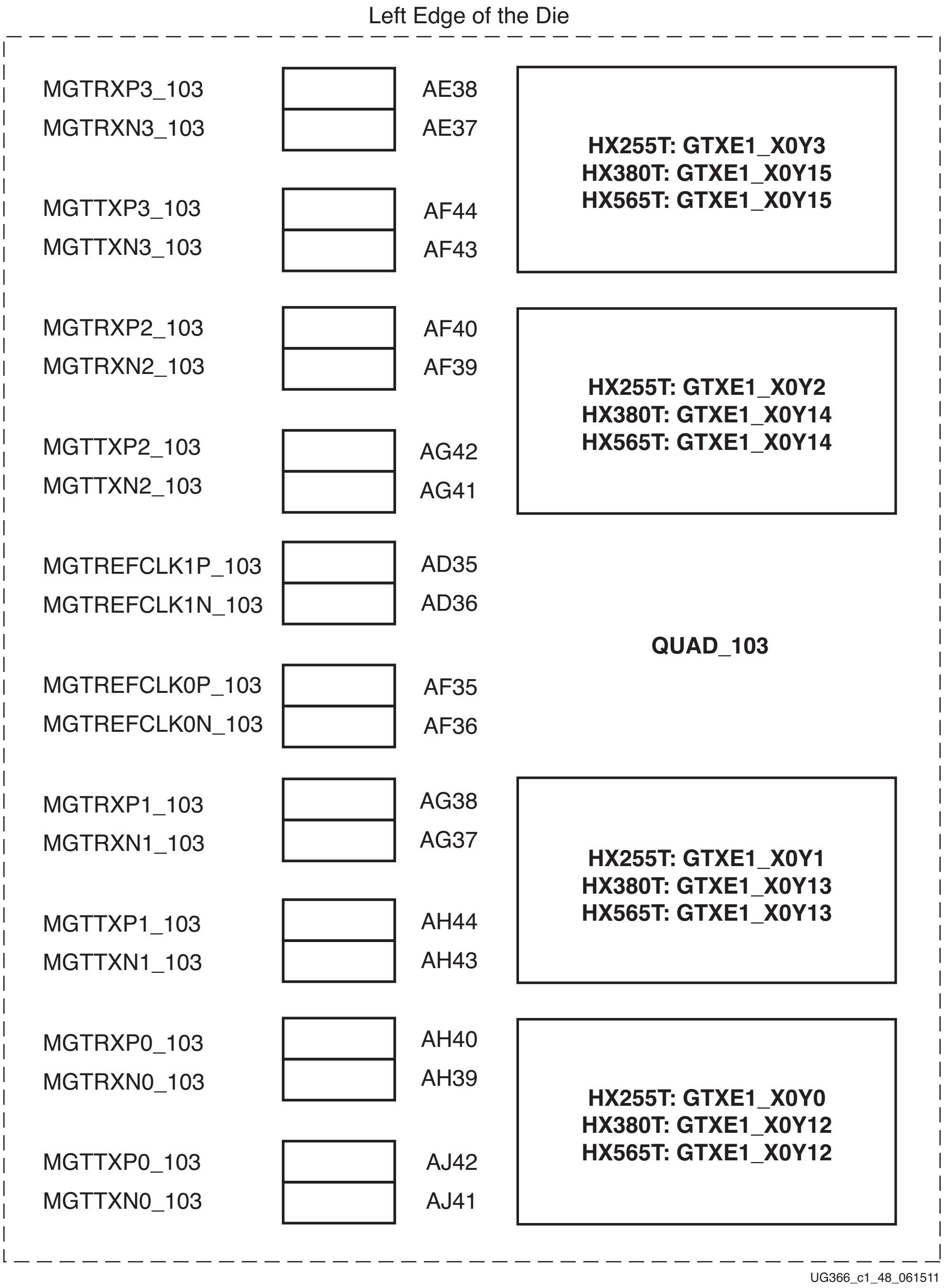


Figure 1-48: **Placement Diagram for the FF1923 Package (7 of 10)**

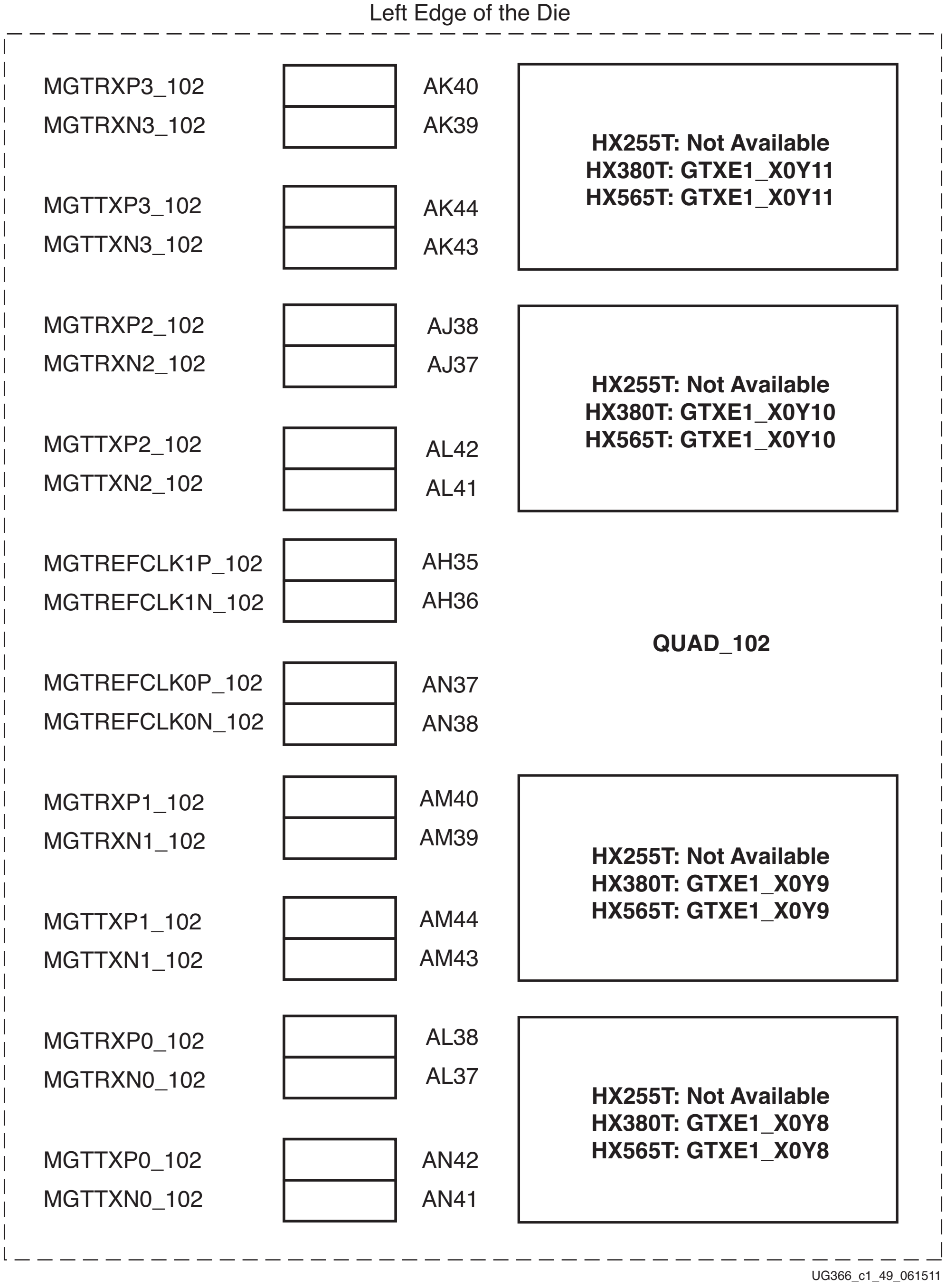


Figure 1-49: **Placement Diagram for the FF1923 Package (8 of 10)**

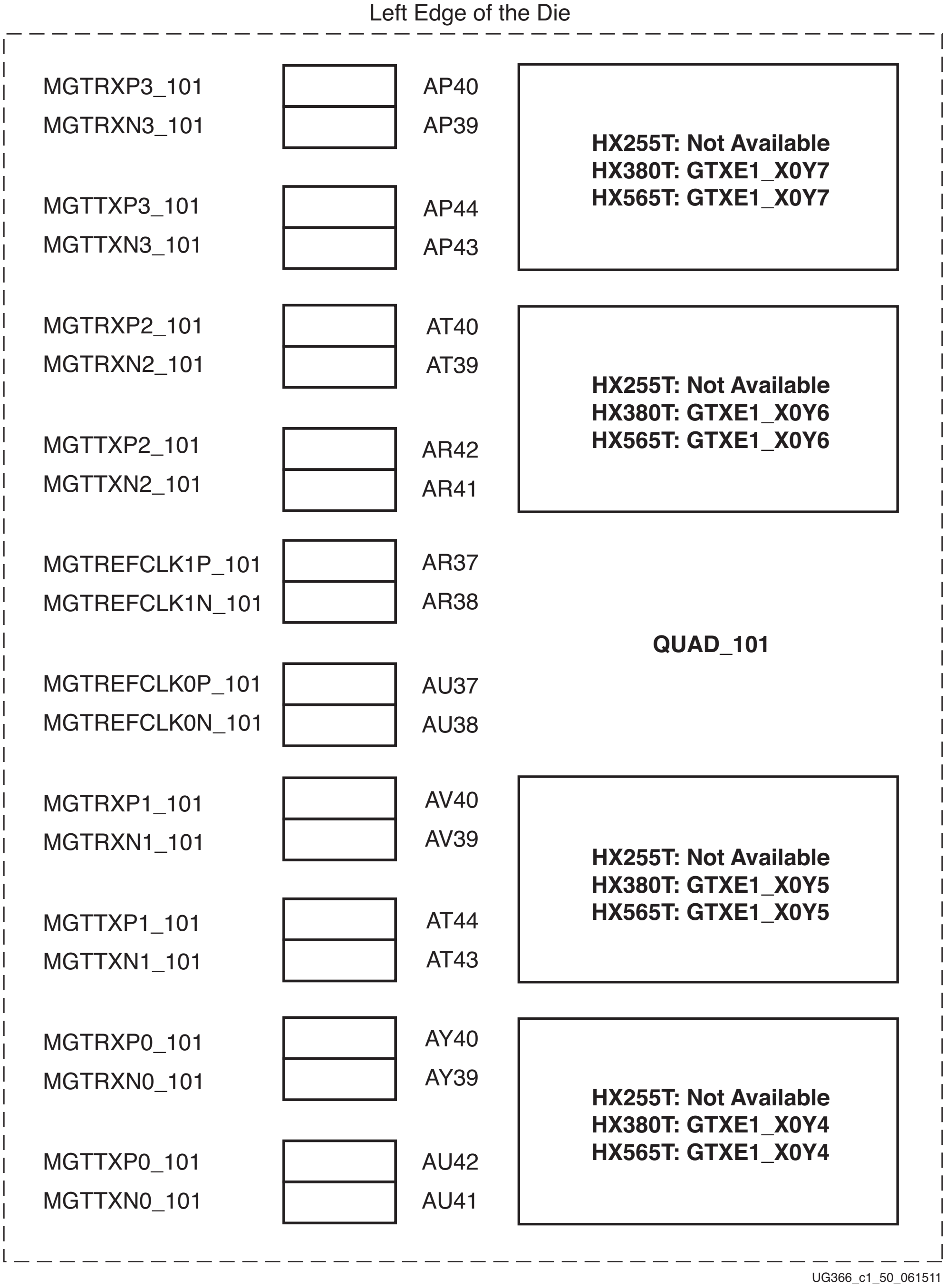
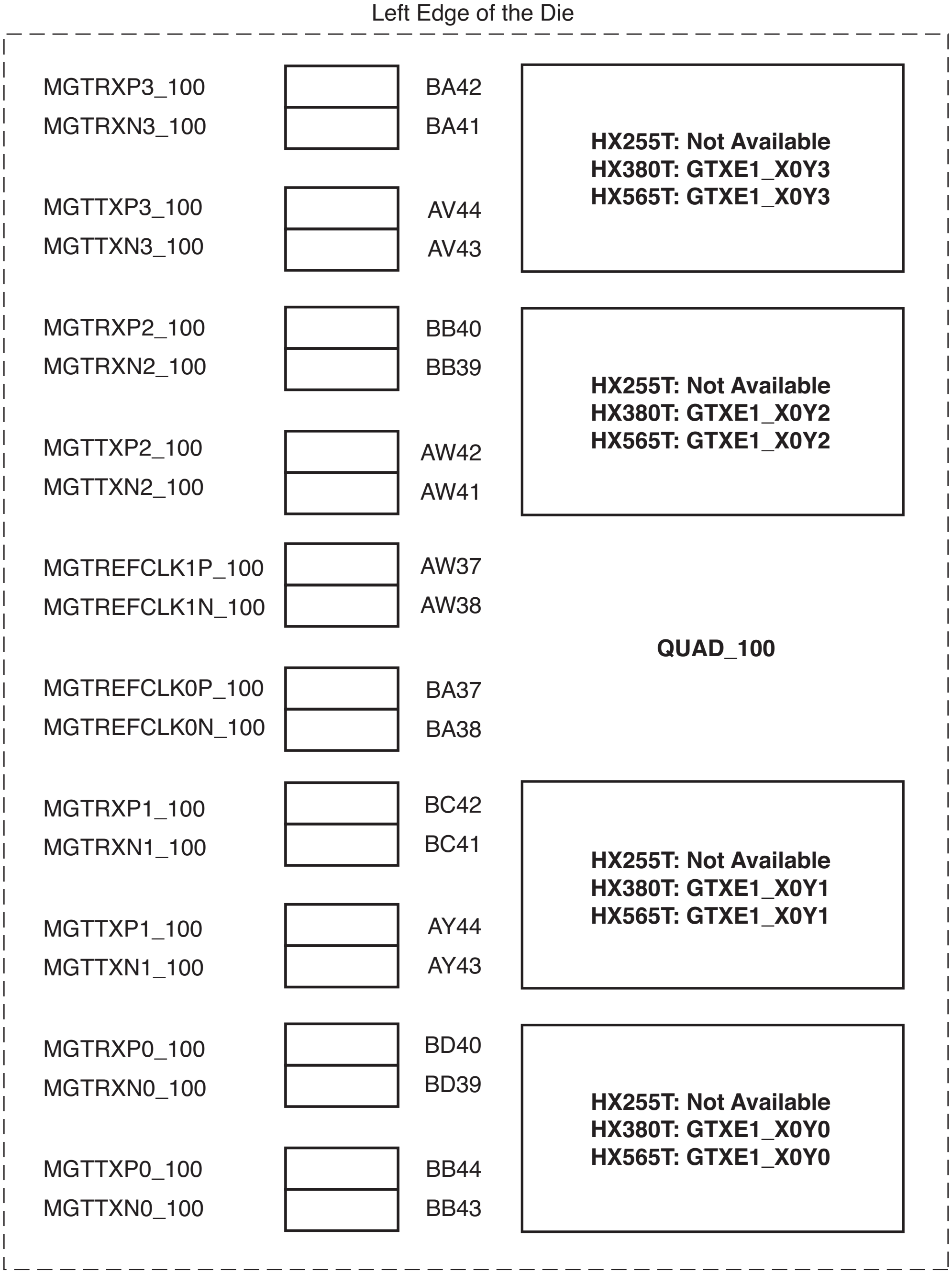


Figure 1-50: **Placement Diagram for the FF1923 Package (9 of 10)**



UG366_c1_51_061511

Figure 1-51: **Placement Diagram for the FF1923 Package (10 of 10)**

FF1924 Package Placement Diagrams

Figure 1-42 through Figure 1-51 show the placement diagrams for the FF1924 package.

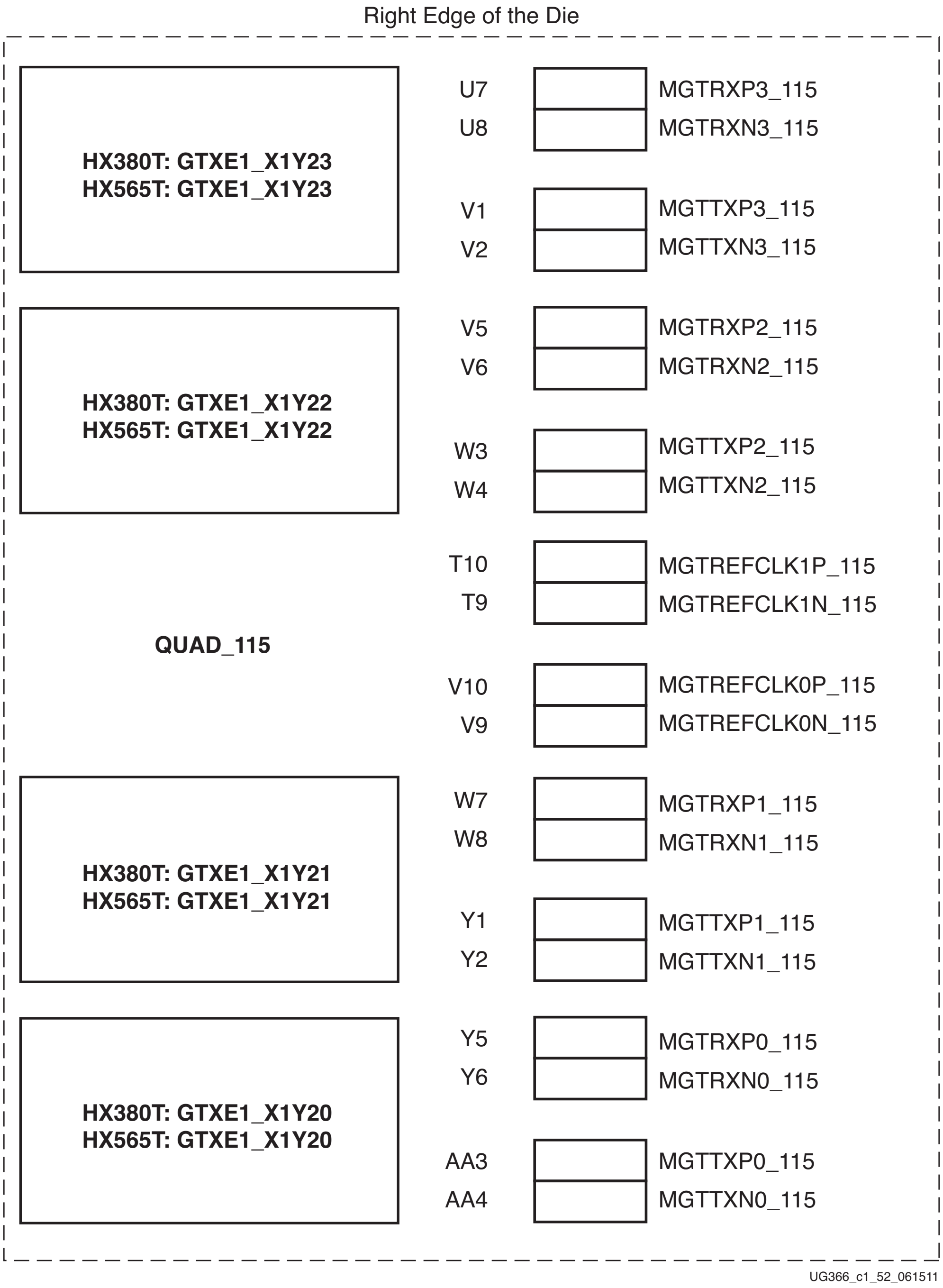


Figure 1-52: **Placement Diagram for the FF1924 Package (1 of 12)**

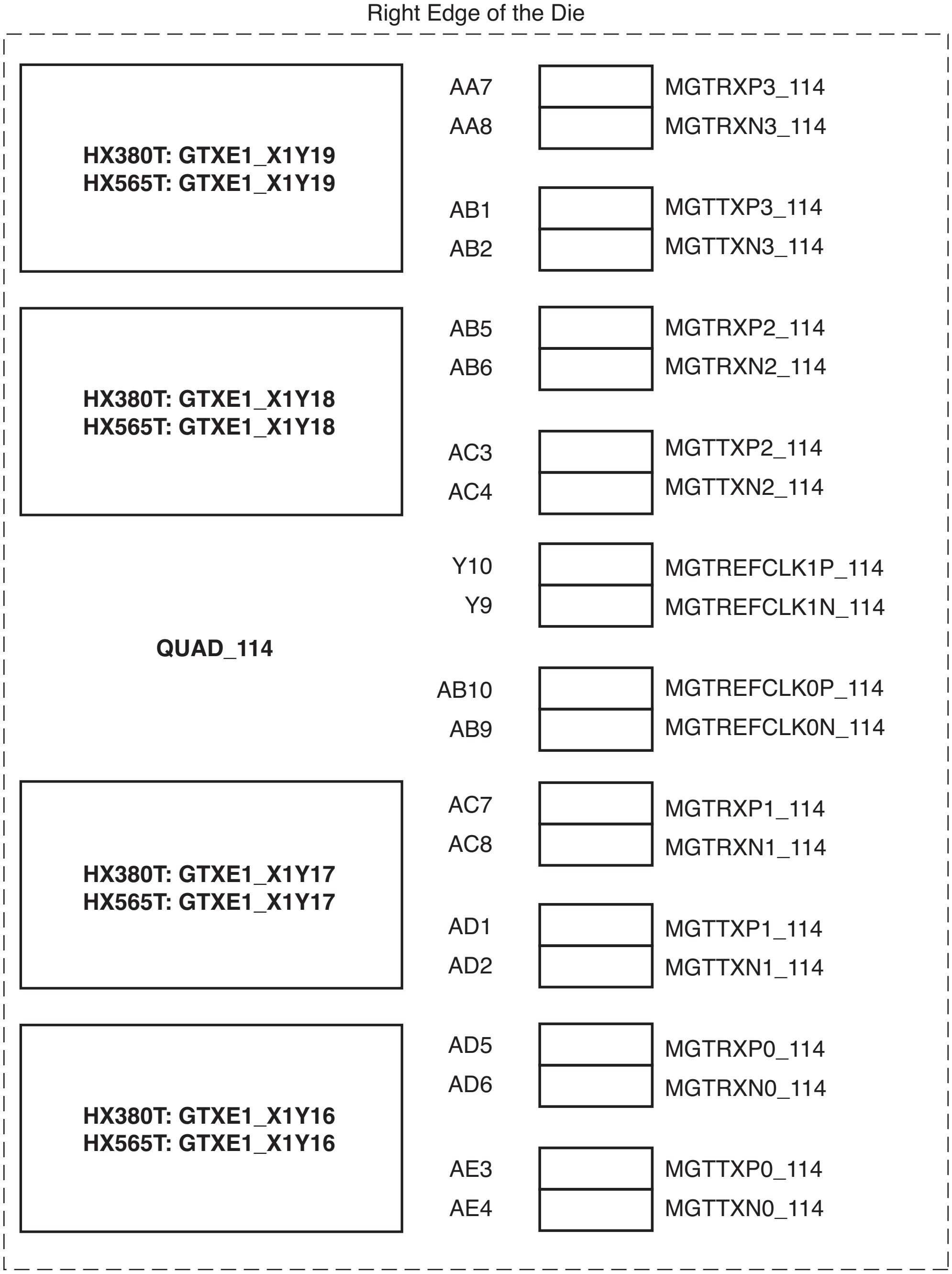


Figure 1-53: **Placement Diagram for the FF1924 Package (2 of 12)**

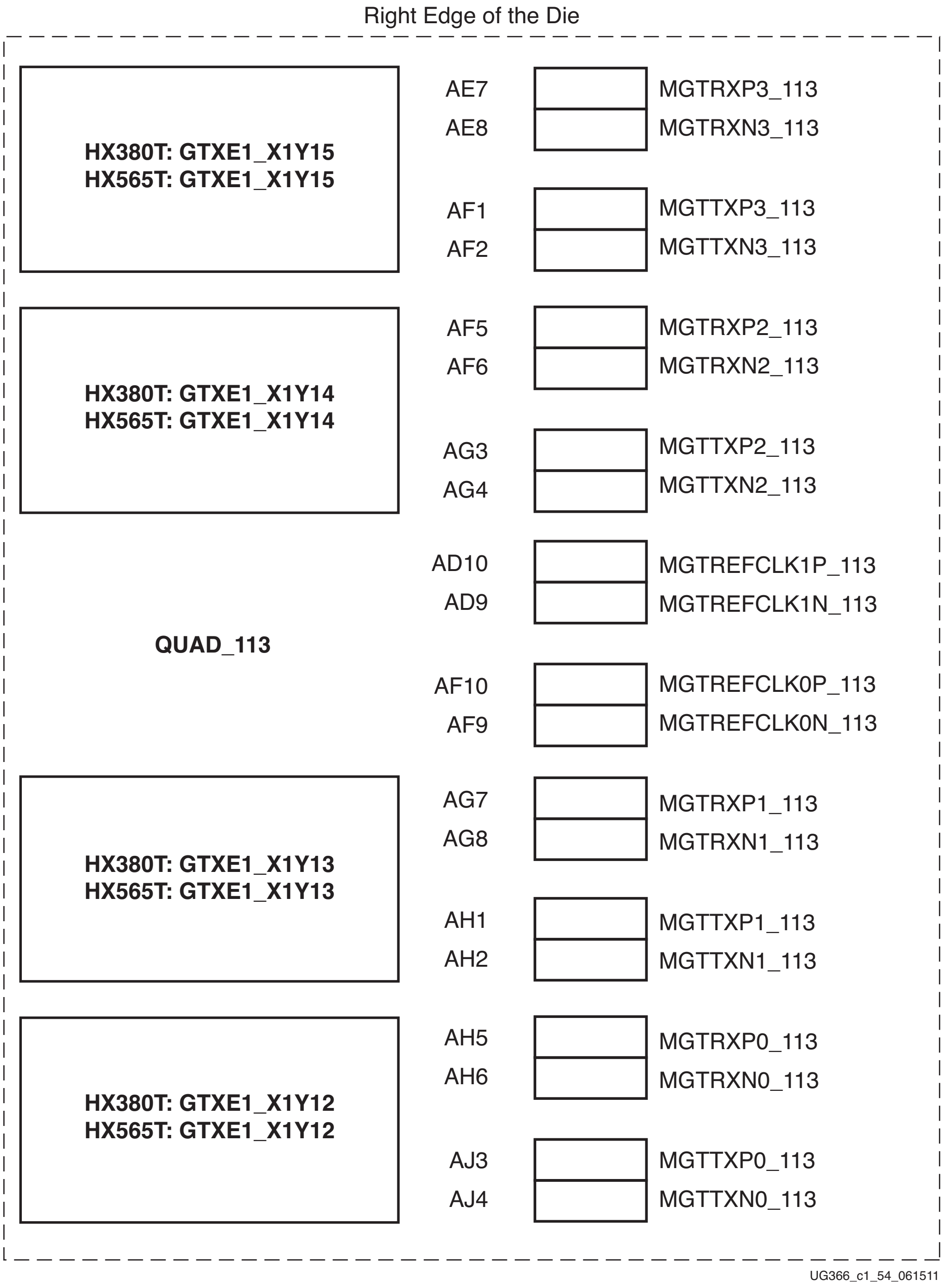


Figure 1-54: **Placement Diagram for the FF1924 Package (3 of 12)**

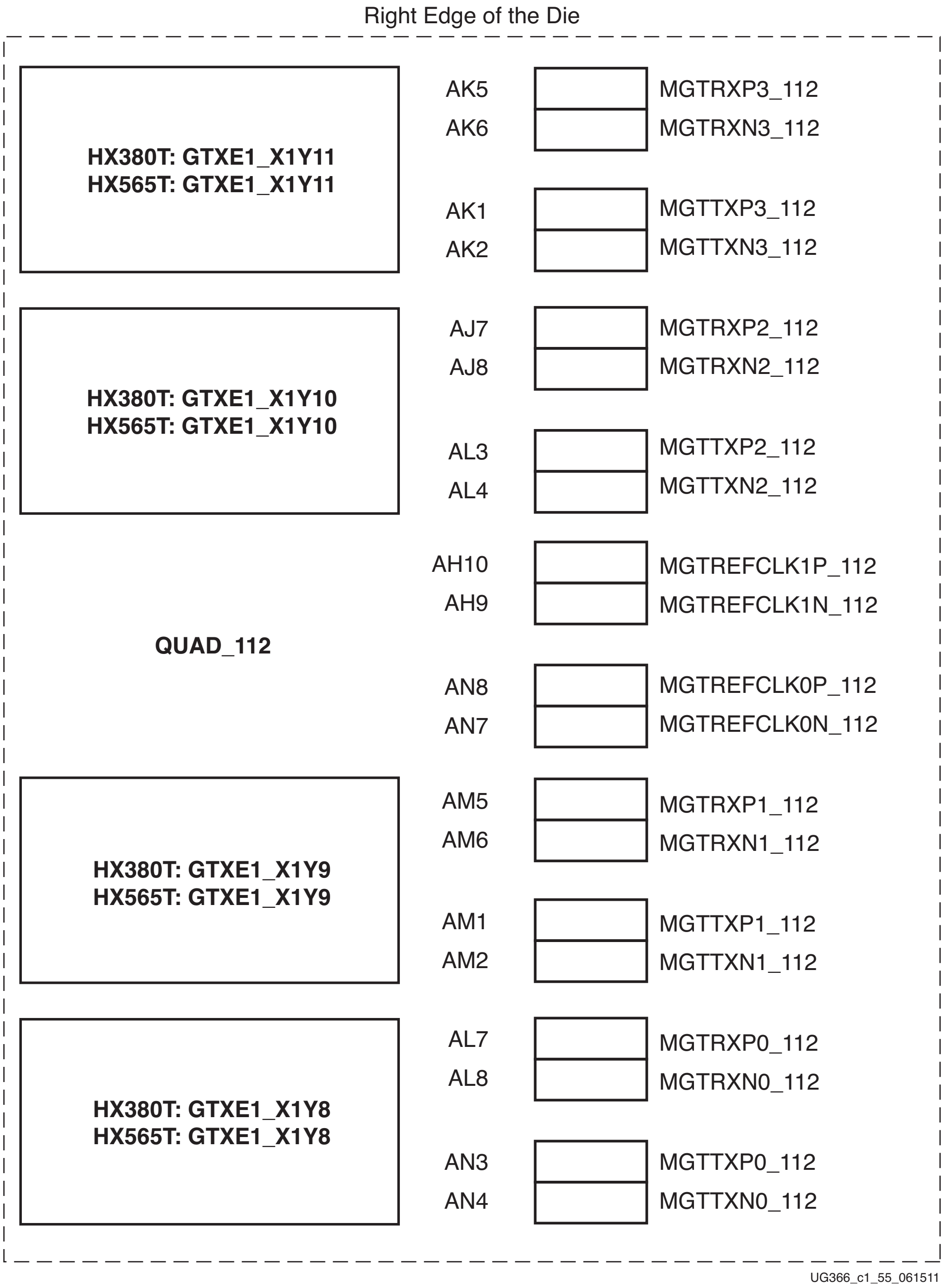


Figure 1-55: **Placement Diagram for the FF1924 Package (4 of 12)**

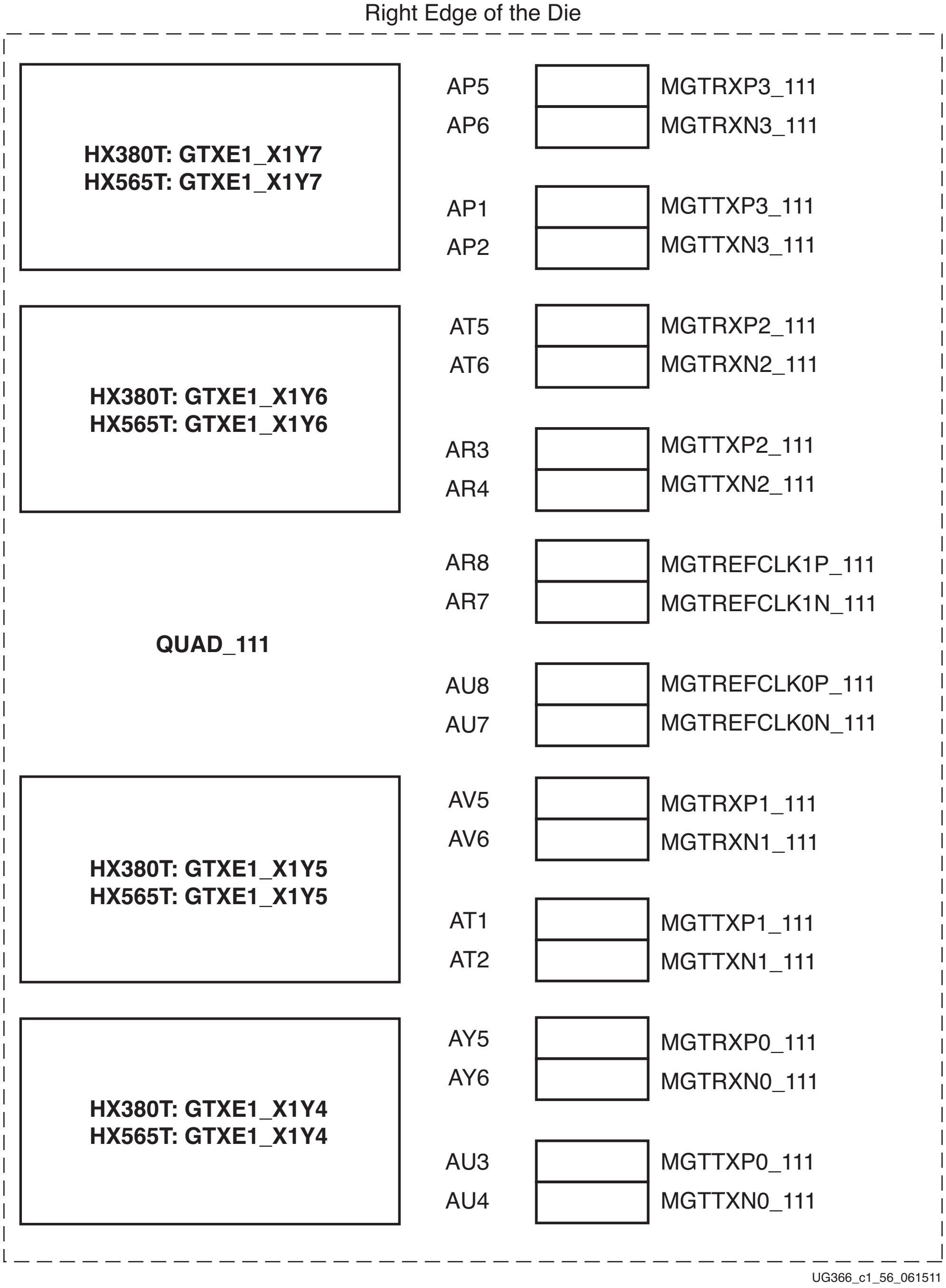
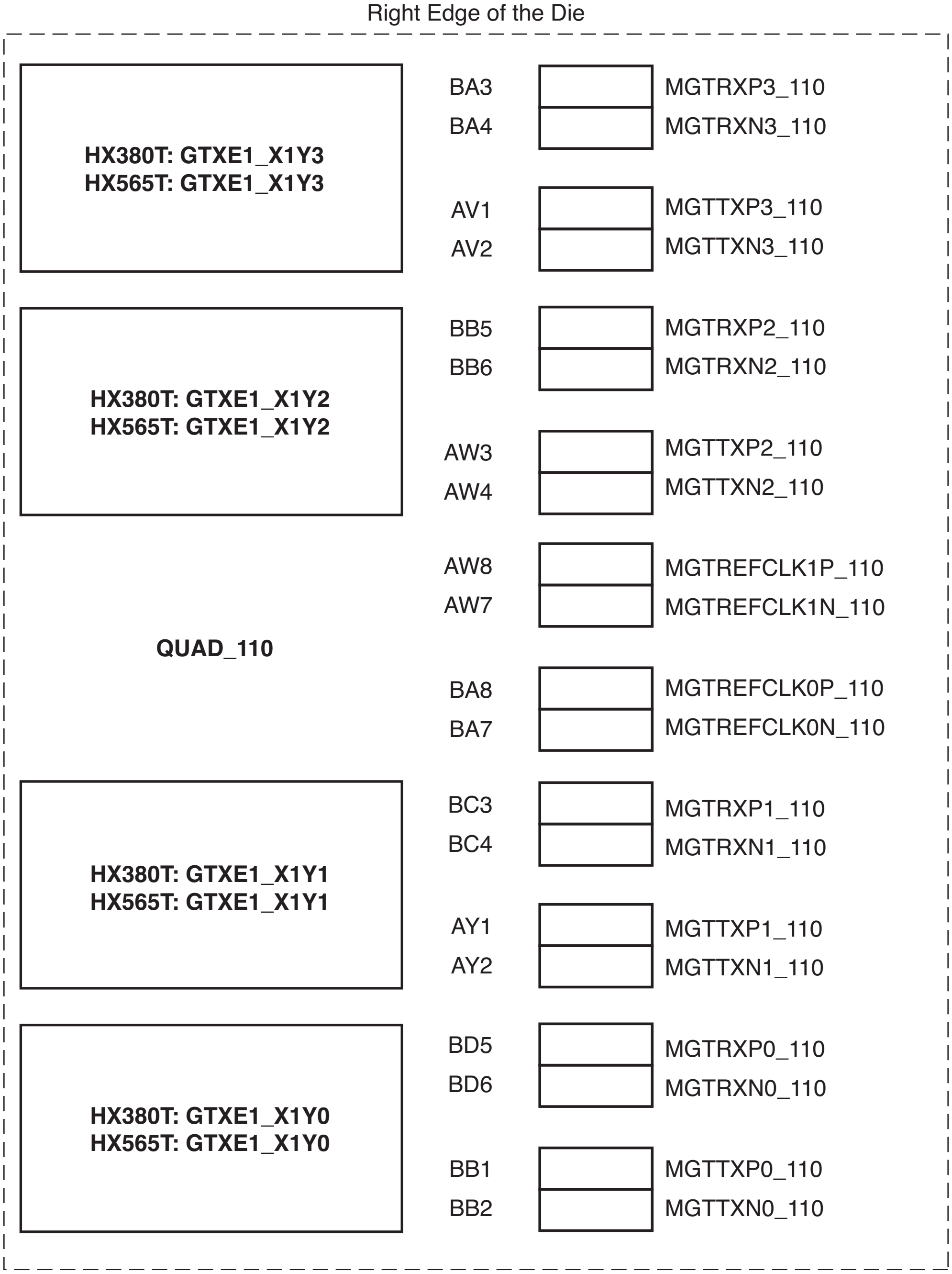


Figure 1-56: **Placement Diagram for the FF1924 Package (5 of 12)**



UG366_c1_57_061511

Figure 1-57: **Placement Diagram for the FF1924 Package (6 of 12)**

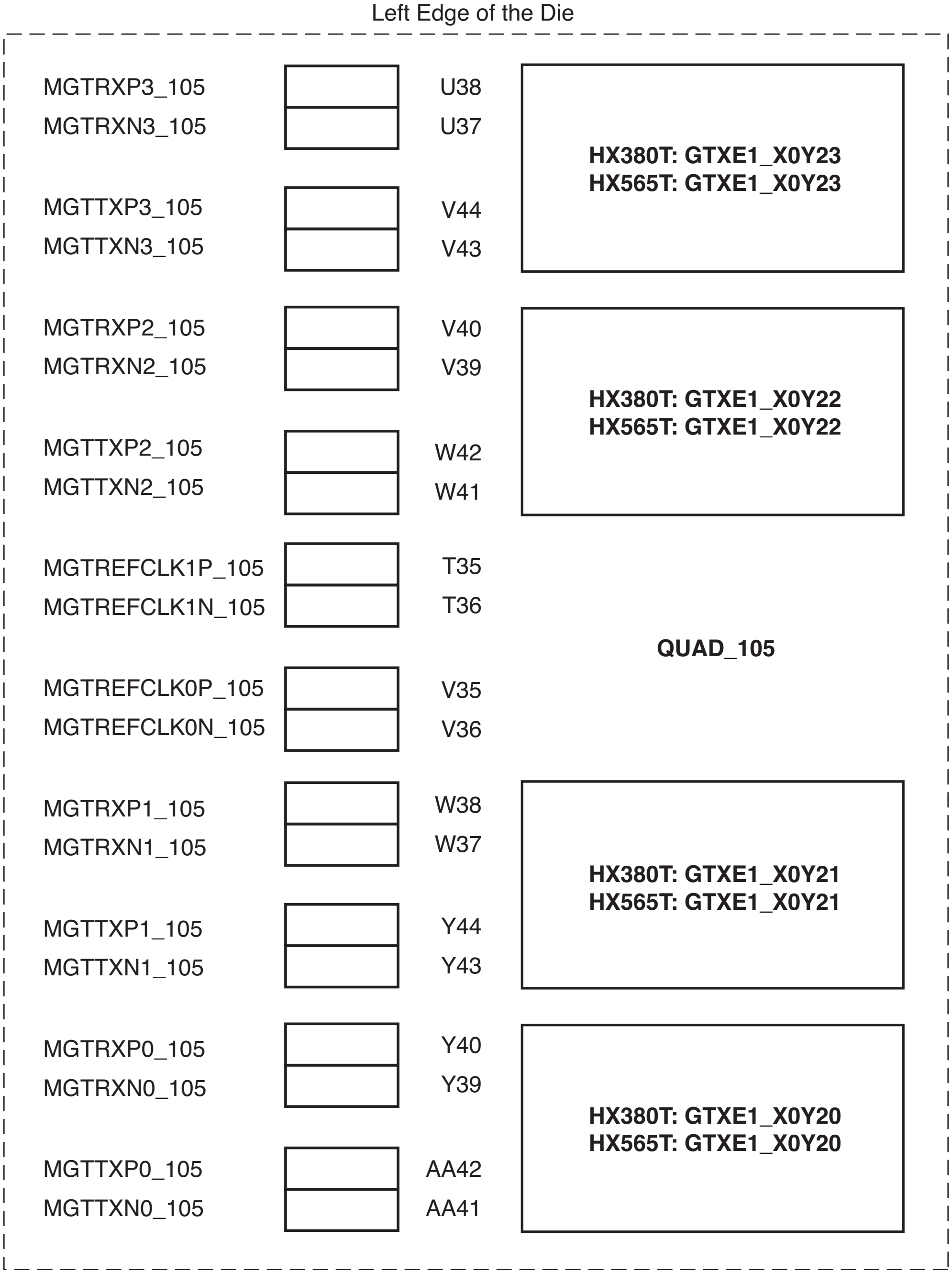


Figure 1-58: **Placement Diagram for the FF1924 Package (7 of 12)**

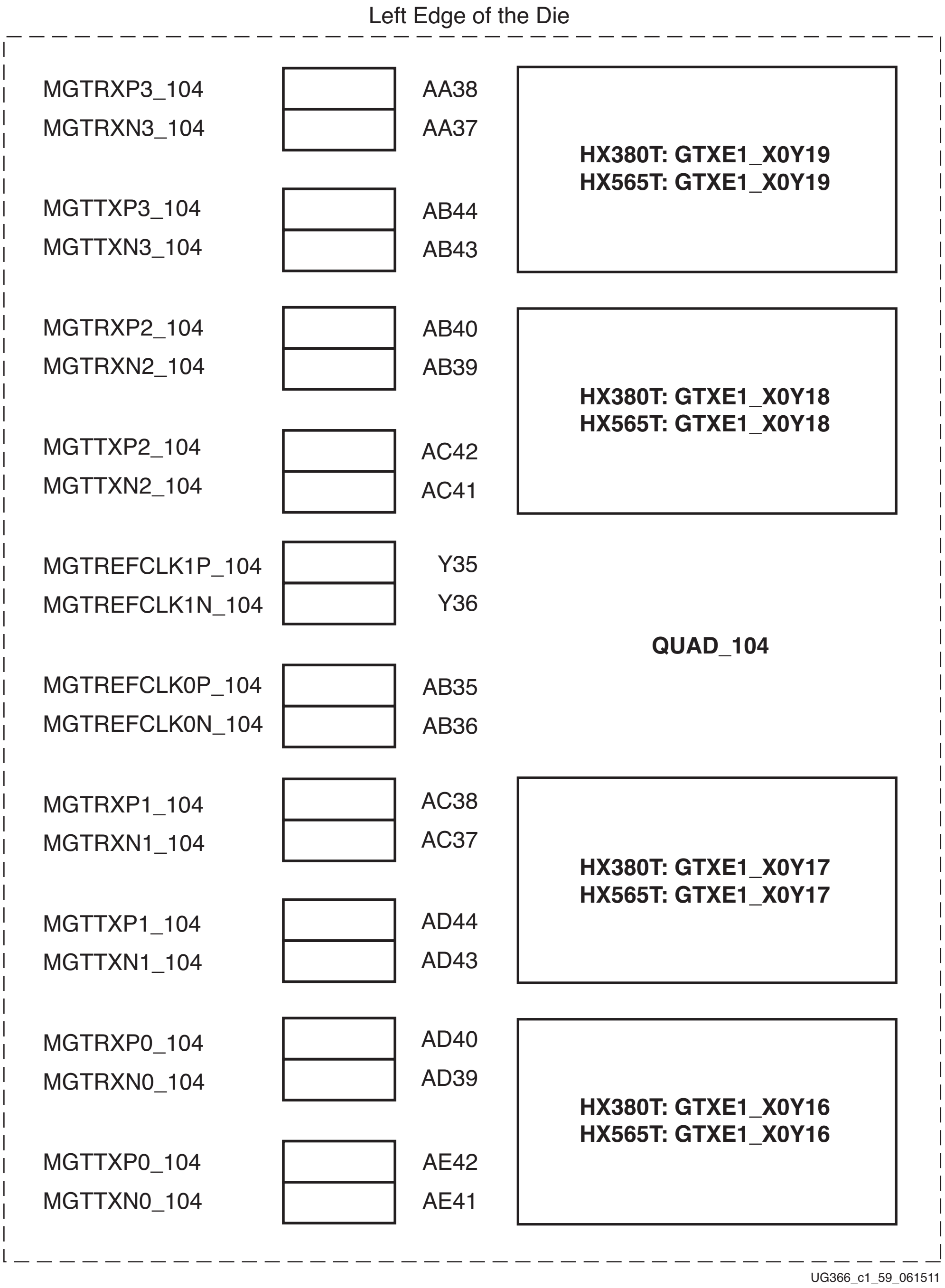
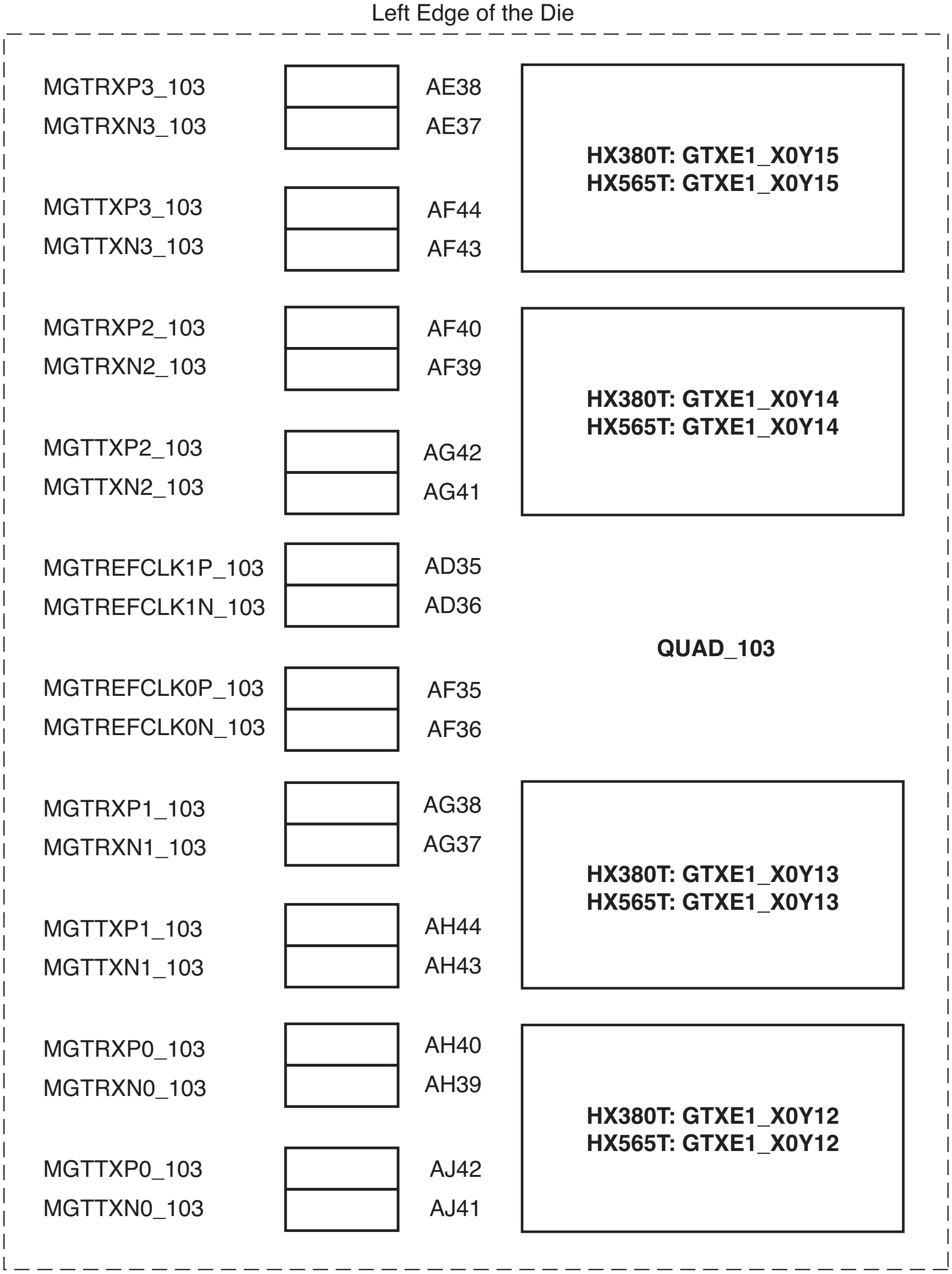


Figure 1-59: **Placement Diagram for the FF1924 Package (8 of 12)**



UG366_c1_60_061511

Figure 1-60: **Placement Diagram for the FF1924 Package (9 of 12)**

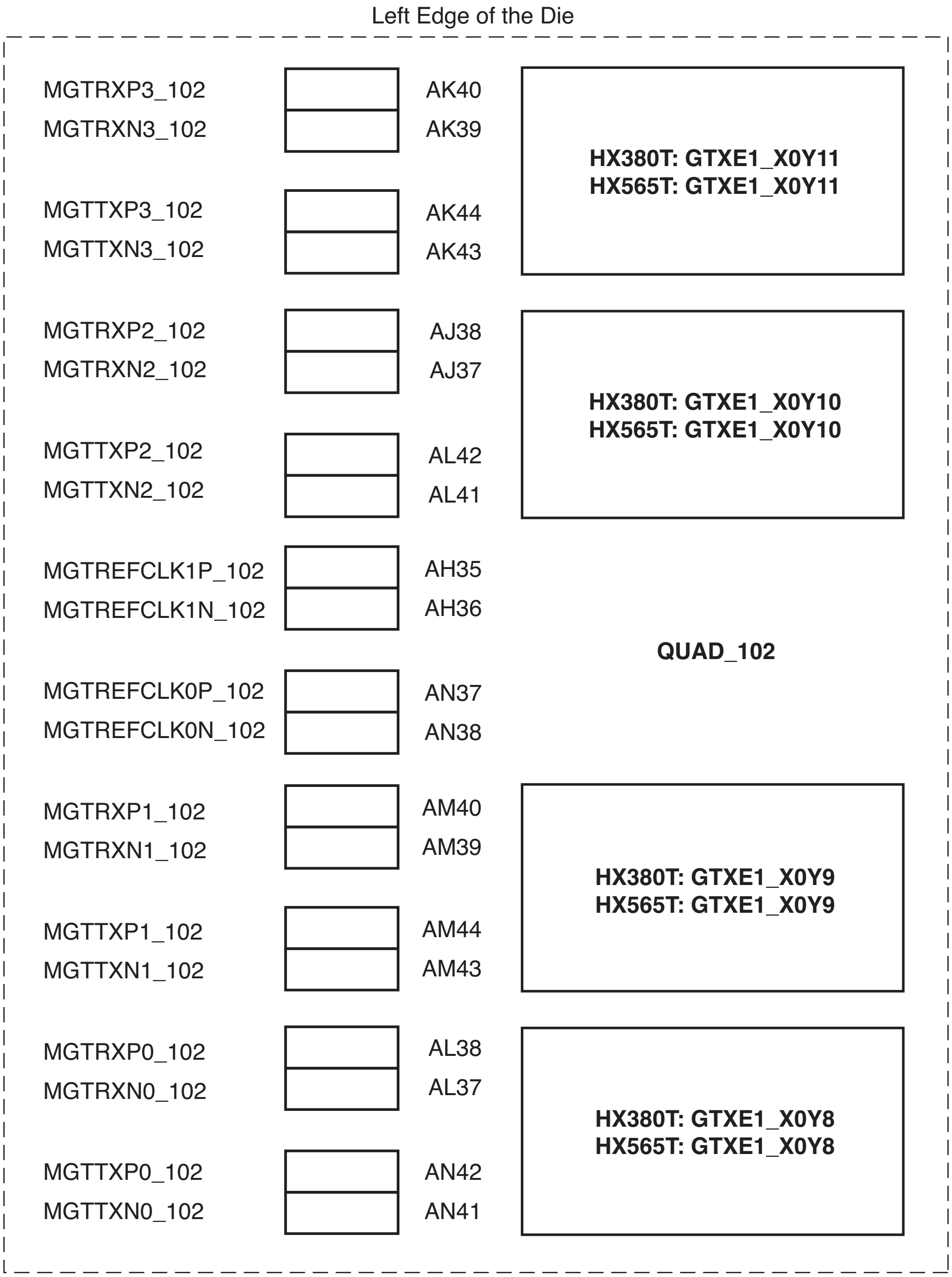
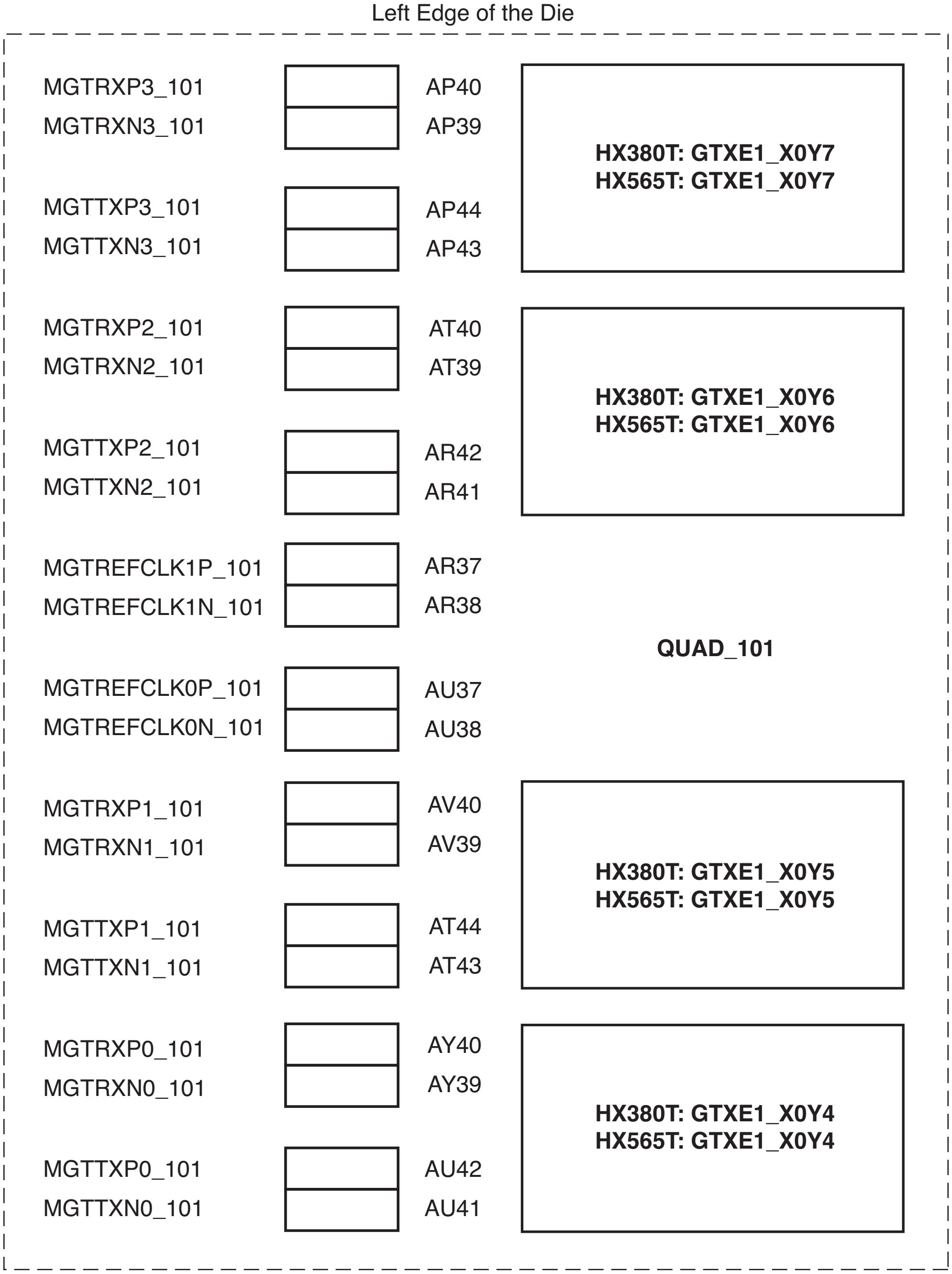


Figure 1-61: **Placement Diagram for the FF1924 Package (10 of 12)**



UG366_c1_62_061511

Figure 1-62: **Placement Diagram for the FF1924 Package (11 of 12)**

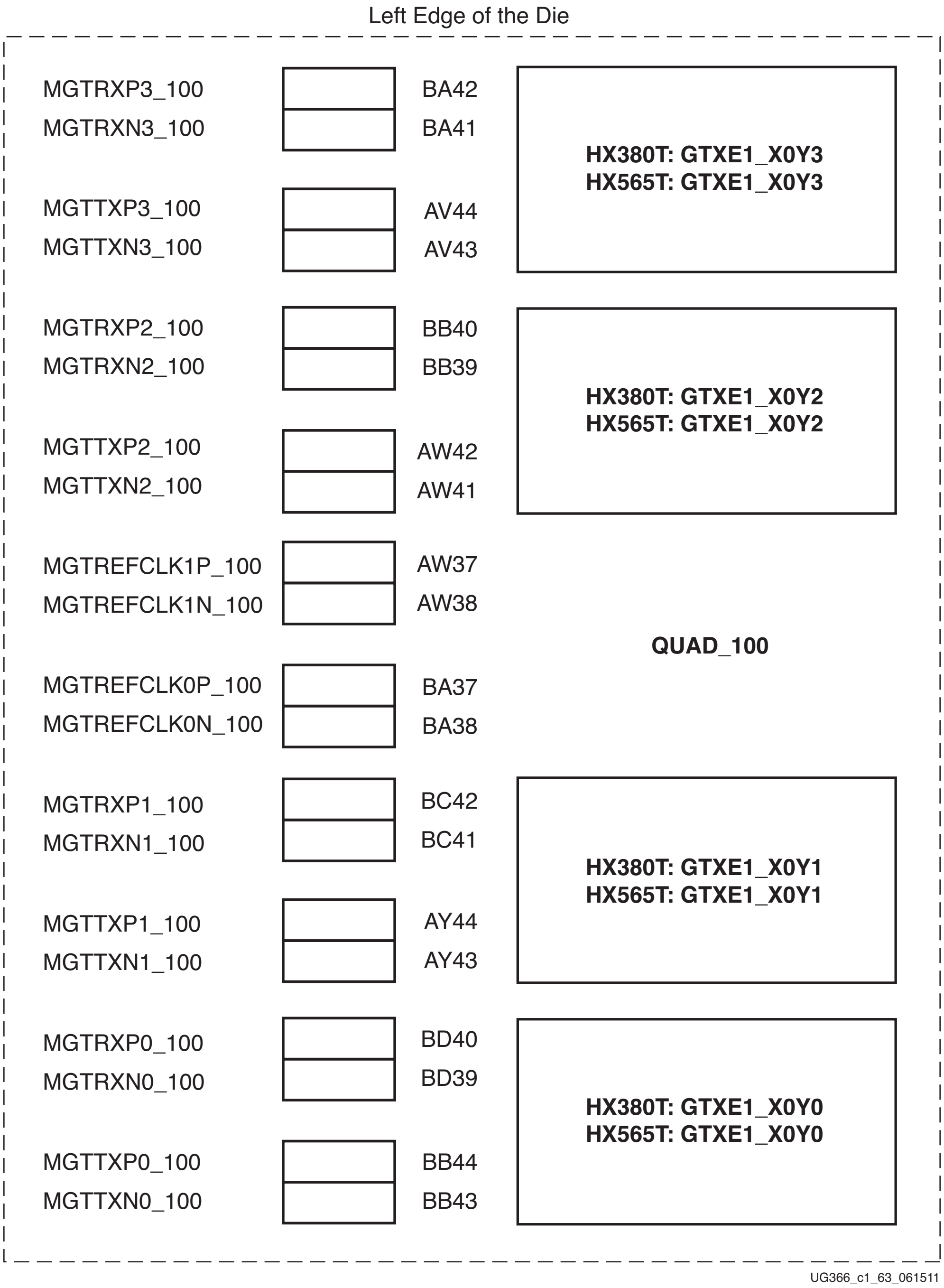


Figure 1-63: **Placement Diagram for the FF1924 Package (12 of 12)**

Shared Transceiver 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 4/5MGTAVCC. The reference clock is instantiated in software with the IBUFDS_GTXE1 software primitive. The ports and attributes controlling the reference clock input are tied to the IBUFDS_GTXE1 software primitive.

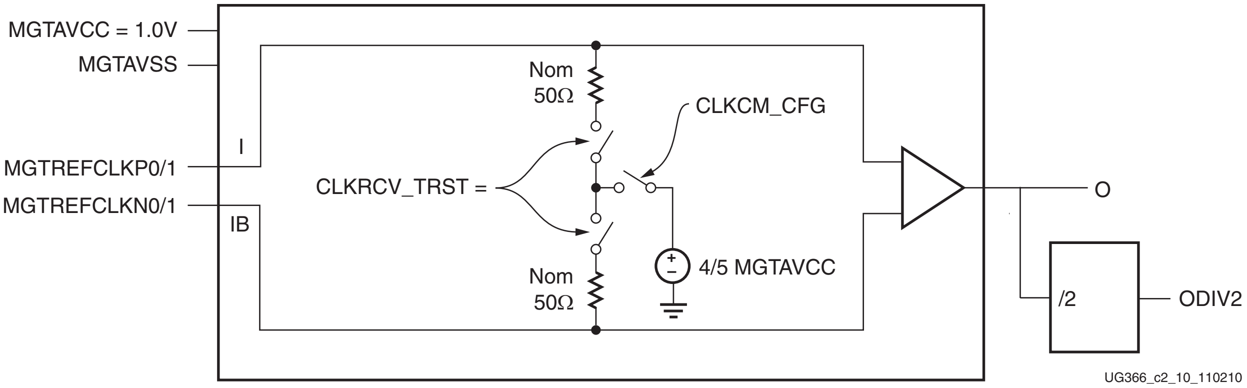


Figure 2-1: **Reference Clock Input Structure**

Ports and Attributes

Table 2-1 defines the reference clock input ports in the IBUFDS_GTXE1 software primitive.

Table 2-1: **Reference Clock Input Ports (IBUFDS_GTXE1)**

Port	Dir	Clock Domain	Description
I IB	In (Pad)	N/A	These are the reference clock input ports that get mapped to MGTREFCLK0P/MGTREFCLK0N and MGTREFCLK1P/MGTREFCLK1P.
CEB	In	N/A	This is the active-Low asynchronous clock enable signal for the clock buffer. Pulling this signal High powers down the clock buffer.

Table 2-1: **Reference Clock Input Ports (IBUFDS_GTXE1) *(Cont'd)***

Port	Dir	Clock Domain	Description
O	Out	N/A	This output drives the MGTREFCLKTX[0/1] and MGTREFCLKRX[0/1] signals in the GTXE1 software primitive. Refer to Reference Clock Selection, page 102 for more details.
ODIV2[1]	Out	N/A	This output is a divide-by-2 version of the O. This signal can be used to drive clock sources in FPGA logic. It can't be used to drive the MGTREFCLKTX[0/1] and MGTREFCLKRX[0/1] signals in the GTXE1 software primitive. Refer to Reference Clock Selection, page 102 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_GTXE1 software primitive that configure the reference clock input.

Table 2-2: **Reference Clock Input Attributes (IBUFDS_GTXE1)**

Attribute	Type	Description
CLKRCV_TRST	Boolean	RESTRICTED. This attribute switches in the 50Ω termination resistors into the signal path. This attribute must always be set to TRUE.
CLKCM_CFG	Boolean	RESTRICTED. This attribute switches in the termination voltage for the 50Ω termination. This attribute must always be set to TRUE.

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

Reference Clock Selection

Functional Description

GTX transceivers provide several available reference clock inputs. Clock selection and availability changed slightly across the first three generations of Virtex® FPGA transceivers. The Virtex-6 FPGA GTX transceiver significantly enhances reference clock capabilities by adding dedicated clock routing and multiplexer resources. Architecturally, the concept of a Quad (or Q), contains a grouping of four `GTXE1` primitives, two dedicated reference clock pin pairs, and dedicated reference clock routing. The term Quad in this document describes the reference clocking architecture of the Virtex-6 FPGA GTX transceivers.

Reference clock features include:

- Clock routing for north and south bound clocks.
- Clock inputs available per GTX transceiver PLL.
- Static or dynamic selection of the reference clock for the transmitter and receiver PLLs.

Figure 2-2 shows the Quad architecture with four GTX transceivers, two dedicated reference clock pin pairs, and dedicated north/south reference clock routing. Each GTX transceiver in a Quad has seven clock inputs available:

- Two local reference clock pin pairs, MGTREFCLK[0/1]
- Two reference clock pin pairs from the Quads above, SOUTHREFCLK[0/1]
- Two reference clocks pin pairs or below, NORTHREFCLK[0/1]
- Internal to each GTX transceiver, the clock from the receiver can be forwarded to the transmit PLL reference clock, CAS_CLK. CAS_CLK must only be used for diagnostics purposes.

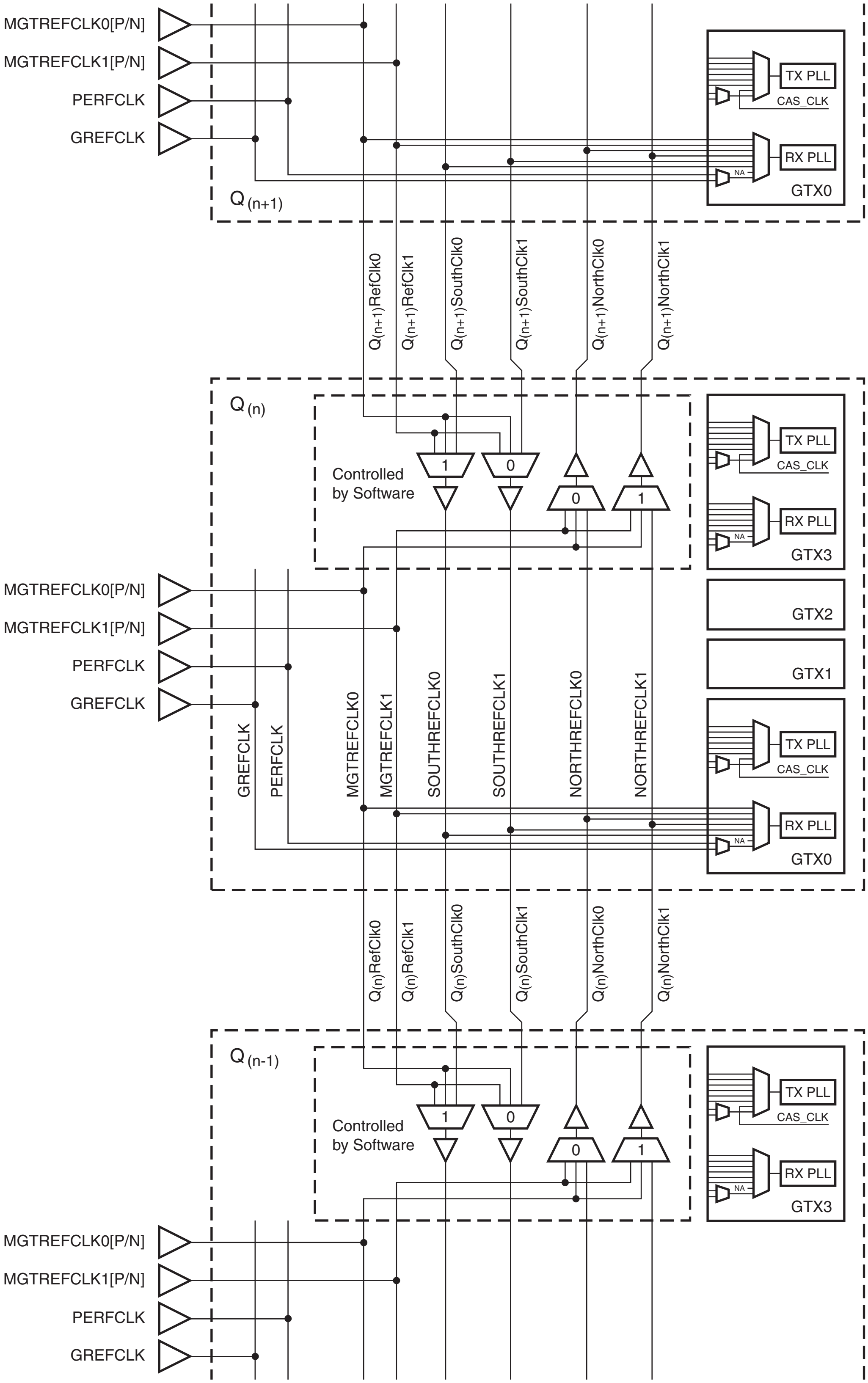
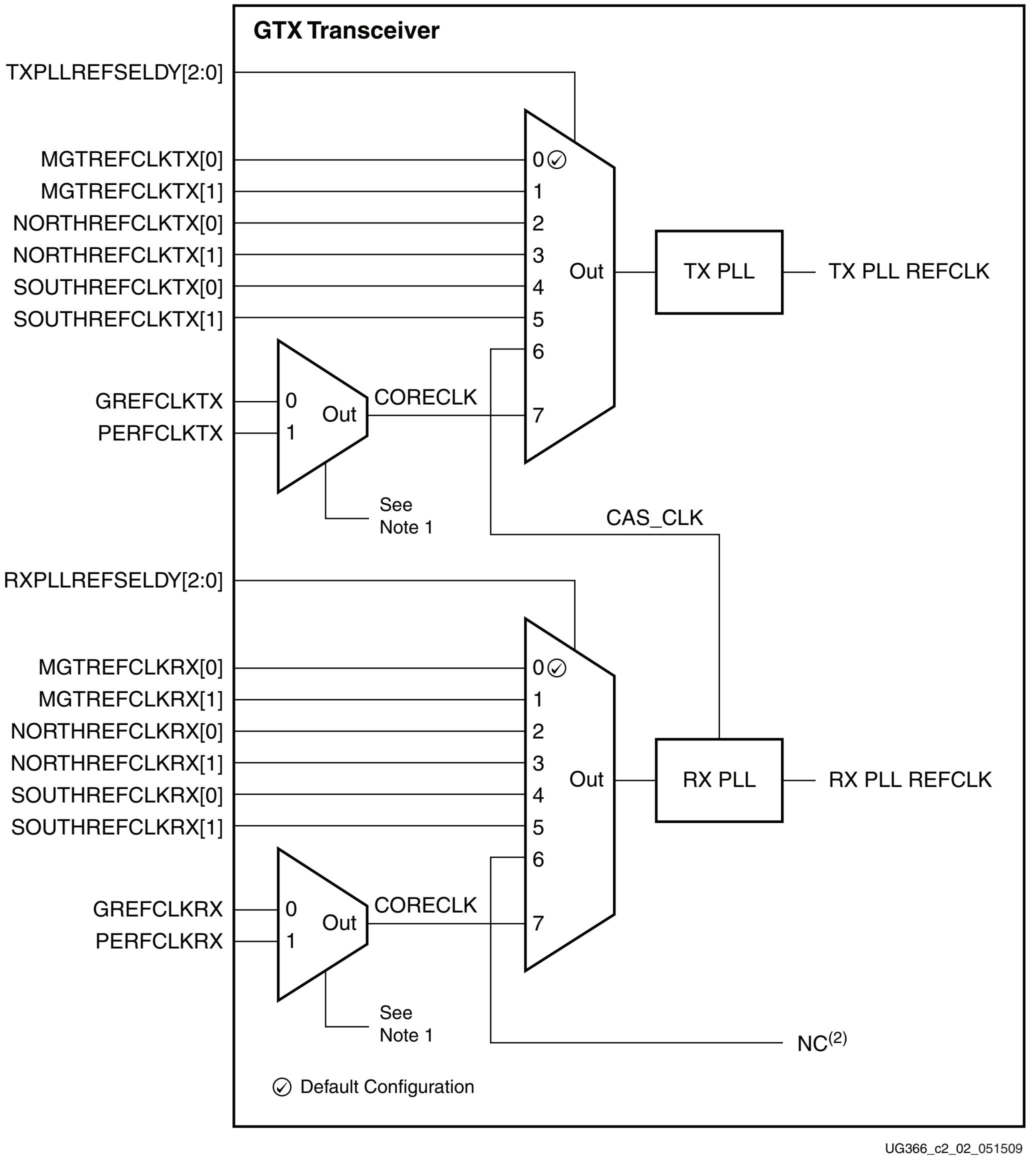


Figure 2-2: **Conceptual View of GTX Transceiver Reference Clocking**

Figure 2-3 shows the detailed view of the reference clock multiplexer structure within a single GTXE1 primitive. The TXPLLREFSELDY and RXPLLREFSELDY ports are required when multiple reference clocks are used. A single reference clock is most commonly used. In this case, the TXPLLREFSELDY and RXPLLREFSELDY ports can be connected to 000, and the Xilinx software tools handle the complexity of the multiplexers and associated routing. See Single External Reference Clock Use Model for more information.



Notes:

1. The CORECLK multiplexer is controlled by software. If GREFCLK is connected, software configures the multiplexer to use GREFCLK. If the PERFCLK is connected, software configures the multiplexer to use PERFCLK. There is no user-controllable attribute to switch the multiplexer. Only one of the inputs can be connected at a time.
2. The CAS_CLK input to the RX PLL is not used or configured.

Figure 2-3: **GTX Transceiver Detailed Diagram**

The four GTX transceivers that make up a Quad share two dedicated reference clock pin pairs. The user design accesses these reference clocks by instantiating IBUFDS_GTXE1 primitives. These reference clocks can be used locally by any of the four GTX transceivers within the Quad. In addition, they can be routed to the GTX transceivers in the north or south neighboring Quads using the dedicated reference clock routing shown in Figure 2-2.

Each GTX transceiver can also select reference clocks from the Quad below ($Q_{(n-1)}$) sourced from the NORTHREFCLKTX[0/1] and NORTHREFCLKRX[0/1] ports; reference clocks from the Quad above ($Q_{(n+1)}$) sourced from the SOUTHREFCLKTX[0/1] and SOUTHREFCLKRX[0/1] ports; reference clocks from the FPGA logic sourced from PERFCLKTX and PERFCLKRX, or GREFCLKTX and GREFCLKRX.

The Xilinx software tools handle the complexity of the multiplexers and associated routing for designs that require a single reference clock per GTX transceiver PLL. If dynamic switching of reference clocks is required, the user must set the reference clock multiplexers using the GTX transceiver TXPLLREFSELDY and RXPLLREFSELDY ports. The dedicated reference clock routing between Quads is set by the Xilinx software tools in both single and multiple reference clock modes.

Internal clock nets of the FPGA can provide reference clocks for the GTX transceiver by connecting the output of a global clocking resource to the GTX transceiver PERFCLK or GREFCLK port. Only one of these inputs can be connected at a time. These reference clock ports have the lowest performance of the available clocking methods because FPGA clocking resources can introduce jitter for operation at high data rates. Use of PERFCLK and GREFCLK is reserved for internal test purposes only.

Ports and Attributes

Table 2-4 defines the GTX transceiver clocking ports.

Table 2-4: **GTX Transceiver Clocking Ports**

Port	Dir	Clock Domain	Description
GREFCLKRX	In	Clock	Internal FPGA logic clock. Reserved for internal testing purposes only.
GREFCLKTX	In	Clock	Internal FPGA logic clock. Reserved for internal testing purposes only.
MGTREFCLKRX[1:0]	In	Clock	External jitter stable clock driven by IBUFDS_GTXE1 for the RX PLL.
MGTREFCLKTX[1:0]	In	Clock	External jitter stable clock driven by IBUFDS_GTXE1 for the TX PLL.
NORTHREFCLKRX[1:0]	In	Clock	North-bound clocks from the Quad below.
NORTHREFCLKTX[1:0]	In	Clock	North-bound clocks from the Quad below.
PERFCLKRX	In	Clock	Internal FPGA logic clock. Reserved for internal testing purposes only.
PERFCLKTX	In	Clock	Internal FPGA logic clock. Reserved for internal testing purposes only.

Table 2-4: **GTX Transceiver Clocking Ports *(Cont'd)***

Port	Dir	Clock Domain	Description
RXPLLREFSELDY[2:0]	In	Async	Receiver PLL reference clock dynamic selection. Set to 000 when one reference clock is used. When multiple reference clocks are connected, RXPLLREFSELDY provides dynamic selection as follows: 000: MGTREFCLKRX[0] selected 001: MGTREFCLKRX[1] selected 010: NORTHREFCLKRX[0] selected 011: NORTHREFCLKRX[1] selected 100: SOUTHREFCLKRX[0] selected 101: SOUTHREFCLKRX[1] selected 110: No connect 111: GREFCLKRX or PERFCLKRX selected (only one of these can be used at a time)
SOUTHREFCLKRX[1:0]	In	Clock	South-bound clocks from the Quad above.
SOUTHREFCLKTX[1:0]	In	Clock	South-bound clocks from the Quad above.
TXPLLREFSELDY[2:0]	In	Async	Transmitter PLL reference clock dynamic selection. Set to 000 when one reference clock is used. When multiple reference clocks are connected, TXPLLREFSELDY provides dynamic selection as follows: 000: MGTREFCLKTX[0] selected 001: MGTREFCLKTX[1] selected 010: NORTHREFCLKTX[0] selected 011: NORTHREFCLKTX[1] selected 100: SOUTHREFCLKTX[0] selected 101: SOUTHREFCLKTX[1] selected 110: CAS_CLK (Internal clock generated from the RX PLL) 111: GREFCLKTX or PERFCLKTX selected (only one of these can be used at a time)

Table 2-5 defines the GTX transceiver clocking attributes.

Table 2-5: **GTX Transceiver Clocking Attributes**

Attribute	Type	Description
PMA_CAS_CLK_EN	Boolean	This attribute is the enable for CAS_CLK from the receiver forwarded to the transmitter PLL. TRUE: Enables CAS_CLK. TXPLLREFSELDY[2:0] is unused in this case. FALSE: Disables CAS_CLK.

Table 2-5: **GTX Transceiver Clocking Attributes *(Cont'd)***

Attribute	Type	Description
SIM_RXREFCLK_SOURCE[2:0]	3-bit Binary	Simulation control for GTX transceiver reference clock selection. This attribute must contain the same binary value as the RXPLLREFSELDY port.
SIM_TXREFCLK_SOURCE[2:0]	3-bit Binary	Simulation control for the GTX transceiver reference clock selection. This attribute is set to the same binary value as the TXPLLREFSELDY port.

Single External Reference Clock Use Model

Each Quad has two dedicated differential reference clock inputs (MGTREFCLK0[P/N] or MGTREFCLK1[P/N]) that can be connected to external clock sources. An IBUFDS_GTXE1 primitive must be instantiated to use these dedicated reference clock pin pairs. The user design connects the IBUFDS_GTXE1 output (O) to the MGTREFCLKRX[0] and MGTREFCLKTX[0] ports of the `GTXE1` primitive. MGTREFCLKTX[0] must be connected even if the TX PLL is not used in the design. The IBUFDS_GTXE1 input pins are constrained in the User Constraints File (UCF). For details about placement constraints and restrictions on clocking resources (MMCM, BUFGCTRL, IBUFDS_GTXE1, BUFG, etc.), refer to the *Virtex-6 FPGA Clocking Resources User Guide*.

The simulation-only attributes must be set on the `GTXE1` primitive to match the clock input used. For the single external reference clock use model, the following settings must be applied (these are the default settings):

- SIM_RXREFCLK_SOURCE = `000`
- SIM_TXREFCLK_SOURCE = `000`

Figure 2-4 shows a single reference clock connected to a single GTX transceiver.

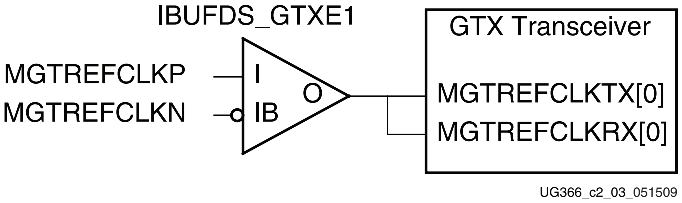


Figure 2-4: **Single External Reference Clock**

Note: The IBUFDS_GTXE1 diagram in Figure 2-4 is a simplification. The output port ODIV2 is left floating, and the input port CEB is set to logic 0.

Figure 2-5 shows a single reference clock connected to multiple GTX transceivers.

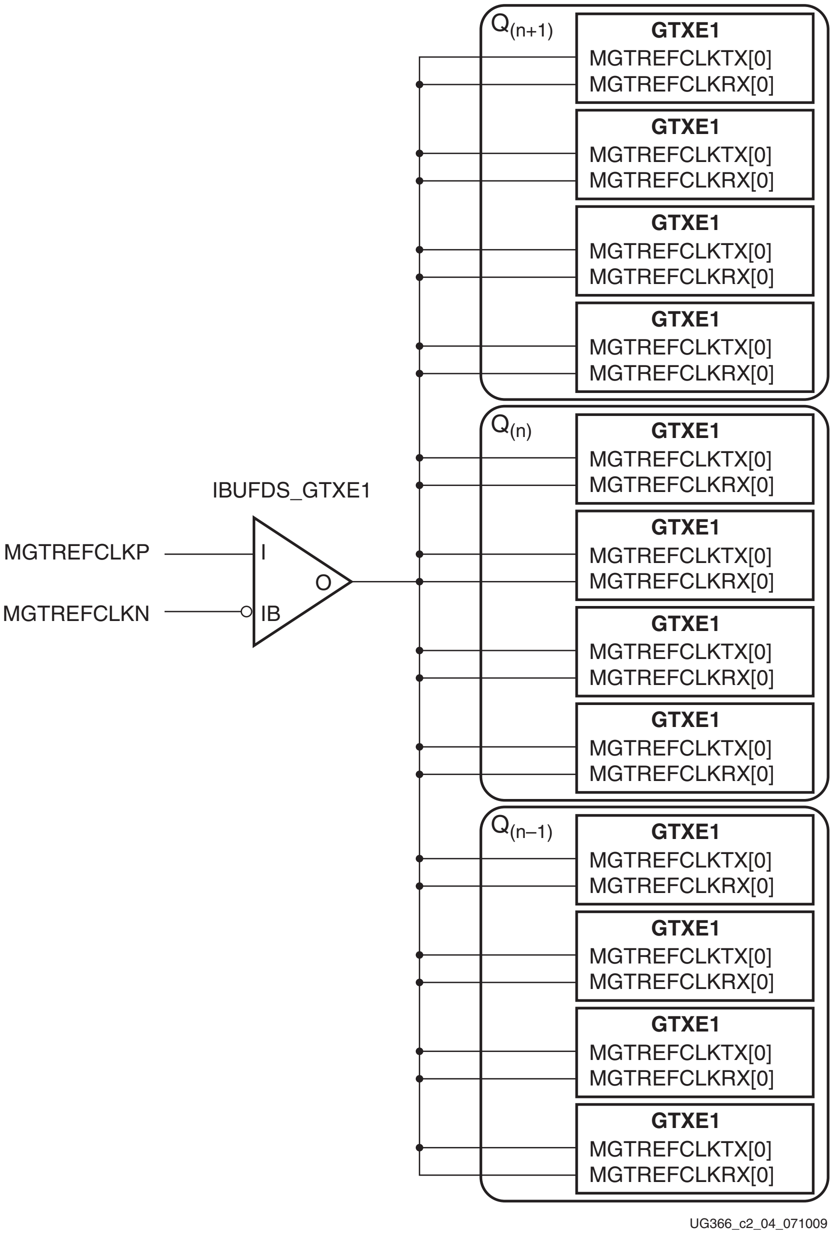


Figure 2-5: **Multiple GTX Transceivers with Shared Reference Clock**

Note: The IBUFDS_GTXE1 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.

The Xilinx implementation tools make the necessary adjustments to the north/south routing shown in Figure 2-2 as well as pin swapping necessary to the GTX transceiver clock inputs to route clocks from one Quad to another when required.

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

1. The number of Quads *above* the sourcing Quad must *not* exceed one.
2. The number of Quads *below* the sourcing Quad must *not* exceed one.

3. The total number of Quads sourced by an external clock pin pair (MGTREFCLKN/MGTREFCLKP) must *not* exceed 3 Quads (or 12 GTX transceivers).

The maximum number of GTX 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 inputs (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_GTXE1 primitive to use these dedicated reference clock resources. For details about placement constraints and restrictions on clocking resources (MMCM, BUFGCTRL, IBUFDS_GTXE1, BUFG, etc.), refer to the *Virtex-6 FPGA Clocking Resources User Guide*. For the first external reference clock, the user design connects the MGTREFCLK0[P/N] IBUFDS_GTXE1 output (O) to the MGTREFCLKRX[0] and MGTREFCLKTX[0] ports of the `GTXE1` primitive. For the second external reference clock, the user design connects the MGTREFCLK1[P/N] IBUFDS_GTXE1 output (O) to the MGTREFCLKRX[1] and MGTREFCLKTX[1] ports of the `GTXE1` primitive. Even if the TX PLL is not used, the corresponding reference clock ports that are used to drive the TX PLL are still required to be connected to ensure the correct reset sequence of the GTX transceiver. The TX reference clock ports can be tied to the clock driving the RX reference clock ports.

Figure 2-6 shows two external reference clock sources connected to multiple GTX transceivers within the same Quad.

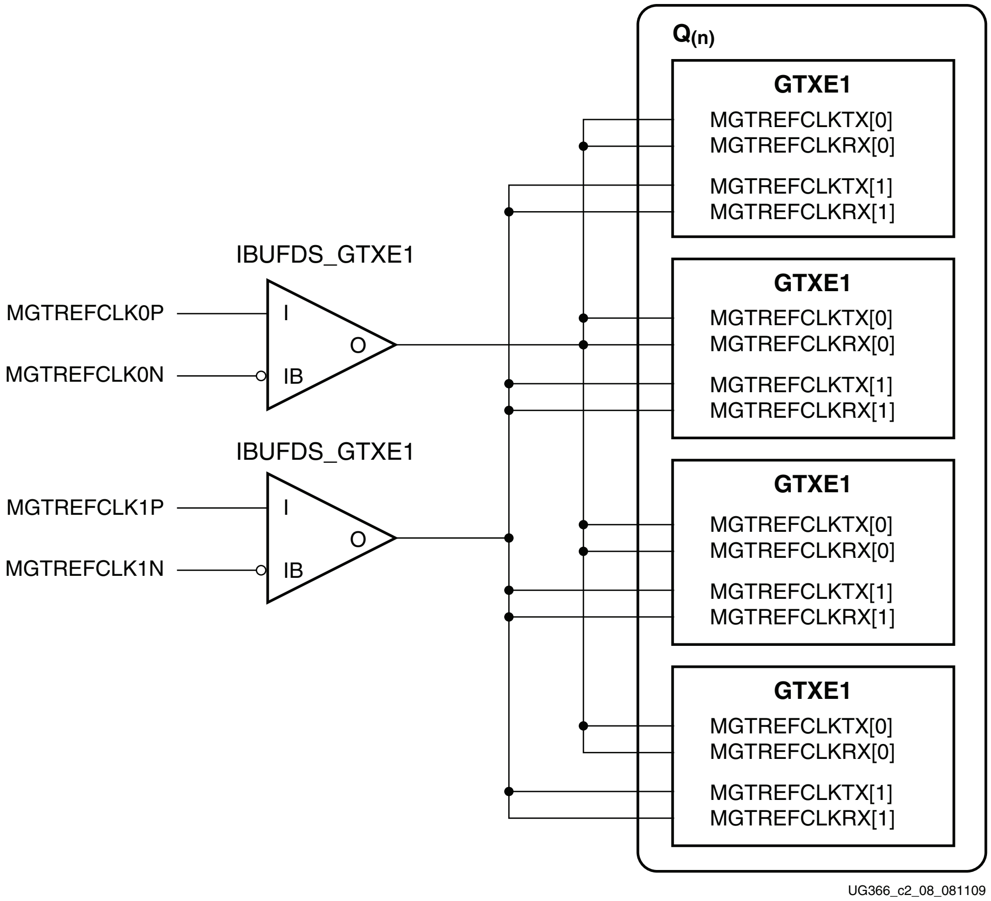


Figure 2-6: **Multiple GTX Transceivers with Multiple Reference Clocks**

Note: The IBUFDS_GTXE1 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.

The User Constraints File (UCF) can be used to constrain the location of the transceivers and the corresponding reference clock source location. For example, if Q(n) in Figure 2-6 belongs to QUAD_114 of an LX75T-FF484 device (as shown in Figure 1-6, page 43), these constraints can be applied:

```
## Set placement for the corresponding GTXE1 instances
INST rocketio_wrapper_i/gtx0_rocketio_wrapper_i/gtxe1_i  LOC=GTXE1_X0Y0;
INST rocketio_wrapper_i/gtx1_rocketio_wrapper_i/gtxe1_i  LOC=GTXE1_X0Y1;
INST rocketio_wrapper_i/gtx2_rocketio_wrapper_i/gtxe1_i  LOC=GTXE1_X0Y2;
INST rocketio_wrapper_i/gtx3_rocketio_wrapper_i/gtxe1_i  LOC=GTXE1_X0Y3;
## Reference clock constraints. Assign the IBUFDS_GTXE1 input nets to the
## package pins for the corresponding dedicated clock sources
## MGTREFCLK0[P/N]_114 or MGTREFCLK1[P/N]_114.

NET MGTREFCLK0N  LOC=U3;
NET MGTREFCLK0P  LOC=U4;
NET MGTREFCLK1N  LOC=R3;
NET MGTREFCLK1P LOC=R4;
```

Figure 2-6 shows that the TX PLL and RX PLL of each transceiver can be sourced by either MGTREFCLK0[P/N] or MGTREFCLK1[P/N]. Users can set TXPLLREFSELDY[2:0] and RXPLLREFSELDY[2:0] to the corresponding value as shown in Figure 2-3, page 105.

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 2-7 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. The Xilinx software tools handle the complexity of the multiplexers and associated routing for designs that require a single reference clock per GTX transceiver PLL. In situations where there is more than one reference clock option per GTX transceiver PLL (Figure 2-7), the user design is required to connect the corresponding ports and set TXPLLREFSELDY[2:0] and RXPLLREFSELDY[2:0] based on the design requirements.

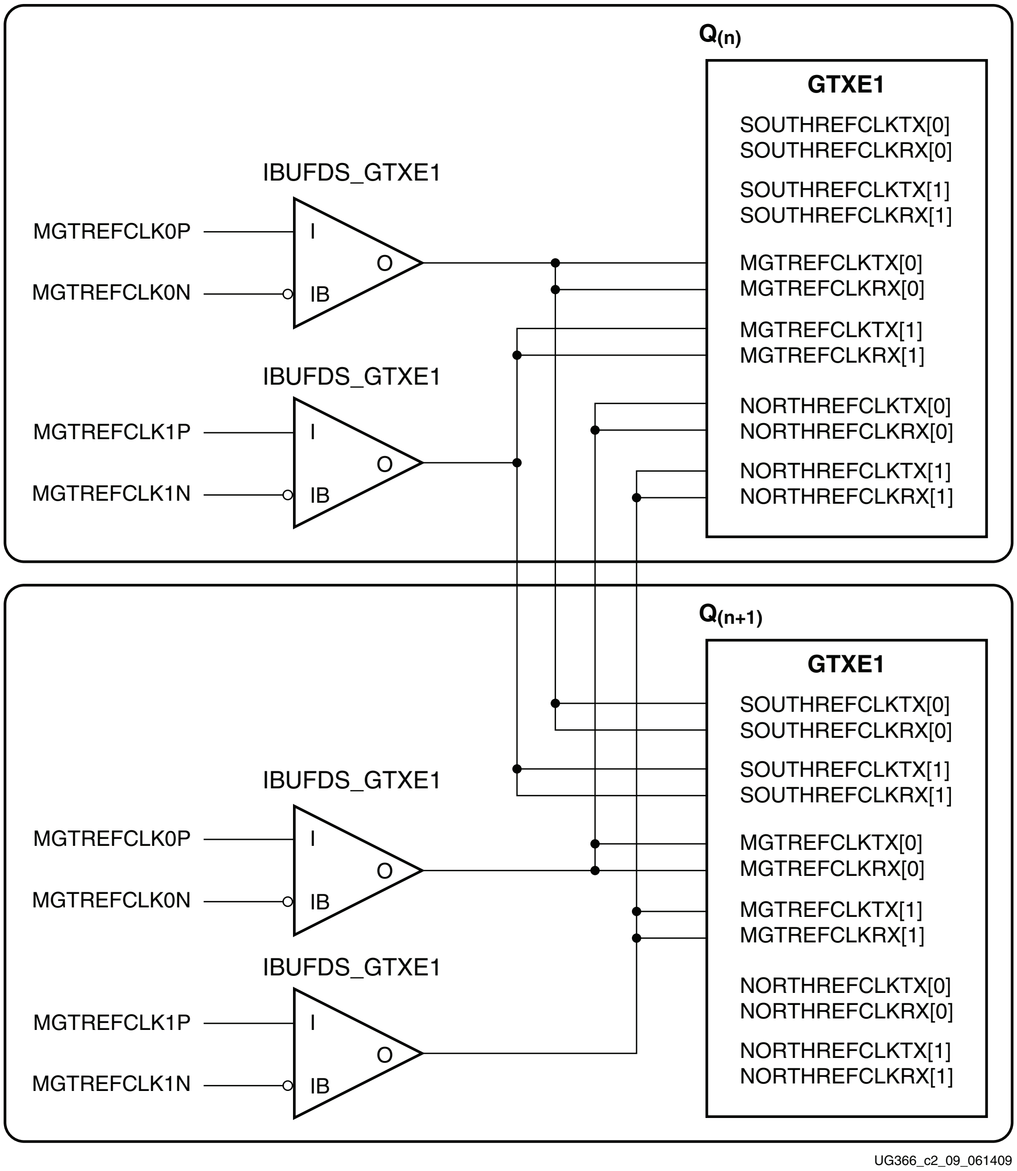


Figure 2-7: **Multiple GTX Transceivers with Multiple Reference Clocks in Different Quads**

Note: The IBUFDS_GTXE1 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.

For multi-rate designs that require the reference clock source to be changed on the fly, the TXPLLREFSELDY and RXPLLREFSELDY ports are used to dynamically select the reference clock source. When the selection has been made, the user design is responsible for resetting the TX PLL and RX PLL via the active-high GTXTXRESET and GTXRXRESET ports. Because GTXTXRESET and GTXRXRESET ports are asynchronous, the user design can simply provide a pulse as short as one clock cycle of the reference clock supported by the FPGA logic. Refer to the PLL section to confirm if the PLL divider settings are optimal for both reference clocks.

When bypassing the TX buffer, all of these requirements must be met:

- If the TX PLL supplies the clock for the TX datapath (TX_CLK_SOURCE = “TXPLL”), the transmitter reference clock must always be toggling.
- If the RX PLL supplies the clock for the TX datapath (TX_CLK_SOURCE = “RXPLL”), the receiver reference clock must always be toggling.

Refer to TX Buffer Bypass, page 156 for more information.

When bypassing the RX buffer, the receiver reference clock must always be toggling. Refer to RX Buffer Bypass, page 230 for more information.

PLL

Functional Description

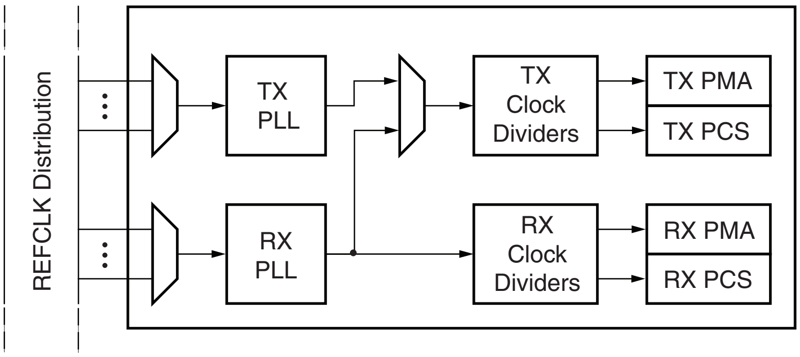
Each GTX transceiver contains one TX PLL and one RX PLL, which allows the TX and RX datapaths to operate in asynchronous frequencies using different reference clock inputs. For applications where the TX and RX datapaths operate in the same line rate range, the RX PLL can be shared between the TX and RX datapaths and the TX PLL can be powered down to conserve power. The TX or RX PLL in one GTX transceiver cannot be shared with other GTX transceivers, only within the same transceiver.

The PLL has a nominal operation range between 1.2 to 2.7 GHz for -1 speed grade devices and 1.2 to 3.3 GHz for -2 and -3 speed -grade devices. Refer to the *Virtex-6 FPGA Data Sheet* for the operating limits. The PLL output has a divider that can divide the output frequency by one, two, or four. Table 2-6 shows the line rates typically supported by each divider setting. An overlapping frequency range exists between PLL output divider settings. The overlapping range provides flexibility with different PLL frequency options in multi-rate applications.

Table 2-6: **Supported Line Rates per Divider Setting**

PLL Output Divider	-1 Line Rate Range (Gb/s)	-2/-3 Line Rate Range (Gb/s)
1	2.4 to 5.0	2.4 to 6.6
2	1.2 to 2.7	1.2 to 3.3
4	0.6 to 1.35	0.6 to 1.65

Lower line rate support requires either fabric-based oversampling circuits or the built-in 5X oversampling block.

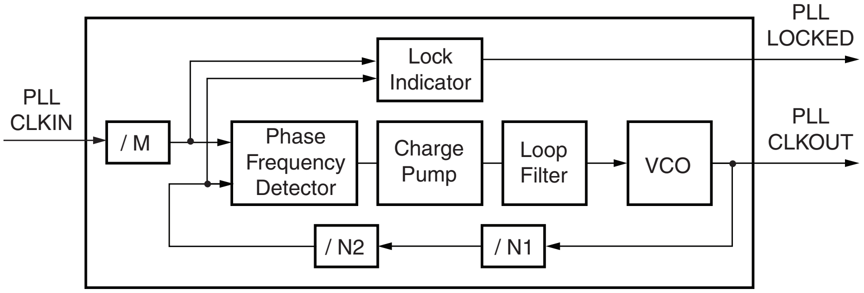


UG366_c2_05_051509

Figure 2-8: **Top-Level PLL Architecture**

The PLL input clock selection is described in Reference Clock Selection, page 102. The PLL 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. These blocks are described in TX Fabric Clock Output Control, page 168 and RX Fabric Clock Output Control, page 206.

Figure 2-9 illustrates a conceptual view of the PLL architecture. A low phase noise PLL input clock is recommended for the best jitter performance. 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 PLL 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.



UG366_c2_06_051509

Figure 2-9: **PLL Detail**

Note: In Figure 2-9, a value of 4 or 5 for the feedback divider (N1*N2) of the PLL is not supported.

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

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

Equation 2-2 shows how to determine the line rate (Gb/s). D is the PLL output divider that resides in the clock divider block.

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

Table 2-7 lists the actual attribute and commonly used divider values.

Table 2-7: **PLL Divider Attribute and Common Values**

Factor	Attribute Name	Valid Settings
M	TXPLL_DIVSEL_REF RXPLL_DIVSEL_REF	1, 2
N1	TXPLL_DIVSEL45_FB RXPLL_DIVSEL45_FB	4, 5
N2	TXPLL_DIVSEL_FB RXPLL_DIVSEL_FB	2, 4, 5
D	TXPLL_DIVSEL_OUT RXPLL_DIVSEL_OUT	1, 2, 4

The Virtex-6 FPGA GTX transceiver allows the N1 divider to be set independently from the PCS internal datapath width. This allows additional flexibility in reference clock selection.

Ports and Attributes

Table 2-8 defines the PLL ports.

Table 2-8: **PLL Ports**

Port	Dir	Clock Domain	Description
PLLTXRESET PLLRXRESET	In	Async	These active-High PLL ports reset the dividers inside the PLL as well as the PLL lock indicator block.
TXPLLLKDET RXPLLLKDET	Out	Async	This active-High PLL frequency lock signal indicates that the PLL has achieved coarse lock.(Frequencies of reference clock and feedback clock are within 25%). The GTX transceiver and its clock outputs are not reliable until this condition is met.
TXPLLLKDETEN RXPLLLKDETEN	In	Async	This port enables the PLL lock detector and must always be tied High.
TXPLLPOWERDOWN RXPLLPOWERDOWN	In	Async	These active-High PLL signals provide power down.

Table 2-9 defines the PLL attributes.

Table 2-9: **PLL Attributes**

Attribute	Type	Description
PMA_CFG	76-bit Binary	Reserved. Use only recommended values from the Virtex-6 FPGA GTX Transceiver Wizard.
TX_CLK_SOURCE	String	This attribute is the multiplexer select signal in Figure 2-8, which determines whether the TX PLL or the RX PLL supplies the clock for the TX datapath. For applications where TX and RX have the same line rate with a small frequency offset, using the RX PLL to supply both the TX and RX datapaths allows some power savings. Valid values are “TXPLL” and “RXPLL”.

Table 2-9: **PLL Attributes *(Cont'd)***

Attribute	Type	Description
TX_TDCC_CFG	2-bit Binary	Reserved. Use only recommended values from the Virtex-6 FPGA GTX Transceiver Wizard.
TXPLL_COM_CFG RXPLL_COM_CFG	24-bit Hex	Reserved. Use only recommended values from the Virtex-6 FPGA GTX Transceiver Wizard.
TXPLL_CP_CFG RXPLL_CP_CFG	8-bit Hex	Reserved. Use only recommended values from the Virtex-6 FPGA GTX Transceiver Wizard.
TXPLL_DIVSEL_FB RXPLL_DIVSEL_FB	Integer	This attribute is N2 in Figure 2-9. This attribute specifies one of the two PLL feedback dividers. Common settings are 1, 2, 4, and 5.
TXPLL_DIVSEL_OUT RXPLL_DIVSEL_OUT	Integer	This attribute is D in Equation 2-2. It specifies the value of the PLL output divider, which resides in the clock divider block. Valid settings are 1, 2, and 4.
TXPLL_DIVSEL_REF RXPLL_DIVSEL_REF	Integer	This attribute is M in Figure 2-9. It specifies the value for the reference clock input divider. Common settings are 1 and 2.
TXPLL_DIVSEL45_FB RXPLL_DIVSEL45_FB	Integer	This attribute is N1 in Figure 2-9. It specifies one of the two PLL feedback dividers. Valid settings are 4 and 5.
TXPLL_LKDET_CFG RXPLL_LKDET_CFG	3-bit Binary	Reserved. Use only recommended values from the Virtex-6 FPGA GTX Transceiver Wizard.
TXPLL_SATA	2-bit Binary	Reserved. Use only recommended values from the Virtex-6 FPGA GTX Transceiver Wizard.

Table 2-9: **PLL Attributes *(Cont'd)***

Attribute	Type	Description
RX_CLK25_DIVIDER TX_CLK25_DIVIDER	Integer	This attribute is set to get an internal clock for the GTX transceiver channel: 1: CLKIN ≤ 25 MHz 2: 25 MHz < CLKIN ≤ 50 MHz 3: 50 MHz < CLKIN ≤ 75 MHz 4: 75 MHz < CLKIN ≤ 100 MHz 5: 100 MHz < CLKIN ≤ 125 MHz 6: 125 MHz < CLKIN ≤ 150 MHz 7: 150 MHz < CLKIN ≤ 175 MHz 8: 175 MHz < CLKIN ≤ 200 MHz 9: 200 MHz < CLKIN ≤ 225 MHz 10: 225 MHz < CLKIN ≤ 250 MHz 11: 250 MHz < CLKIN ≤ 275 MHz 12: 275 MHz < CLKIN ≤ 300 MHz 13: 300 MHz < CLKIN ≤ 325 MHz 14: 325 MHz < CLKIN ≤ 350 MHz 15: 350 MHz < CLKIN ≤ 375 MHz 16: 375 MHz < CLKIN ≤ 400 MHz 17: 400 MHz < CLKIN ≤ 425 MHz 18: 425 MHz < CLKIN ≤ 450 MHz 19: 450 MHz < CLKIN ≤ 475 MHz 20: 475 MHz < CLKIN ≤ 500 MHz 21: 500 MHz < CLKIN ≤ 525 MHz 22: 525 MHz < CLKIN ≤ 550 MHz 23: 550 MHz < CLKIN ≤ 575 MHz 24: 575 MHz < CLKIN ≤ 600 MHz 25: 600 MHz < CLKIN ≤ 625 MHz 26-32: Reserved

PLL Settings for Common Protocols

Table 2-10 shows example PLL divider settings for several standard protocols.

Table 2-10: **PLL Divider Settings for Common Protocols**

Standard	Line Rate [Gb/s]	Internal Data Width [16b/20b]	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

Table 2-10: **PLL Divider Settings for Common Protocols *(Cont'd)***

Standard	Line Rate [Gb/s]	Internal Data Width [16b/20b]	PLL Frequency [GHz]	REFCLK (Typical) [MHz]	Using Typical REFCLK Frequency			
					N1	N2	D	M
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	1.5625	156.25	5	2	1	1
GigE	1.25	20b	1.25	125	5	2	2	1
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	1.5625	156.25	5	2	1	1
	2.5	20b	2.5	125	5	4	2	1
	1.25	20b	1.25	125	5	2	2	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
Serial RapidIO (Single Rate)	3.125	20b	1.5625	156.25	5	2	1	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	1.5625	156.25	5	2	1	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	1.5	150	5	2	1	1
	1.5	20b	1.5	150	5	2	2	1
PCIe Optimal Jitter	5	20b	2.5	250	5	2	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 1-4X (Multi-Rate)	2.4576	20b	1.2288	122.88	5	2	1	1
	1.2288	20b	1.2288	122.88	5	2	2	1
	0.6144	20b	1.2288	122.88	5	2	4	1

Table 2-10: **PLL Divider Settings for Common Protocols *(Cont'd)***

Standard	Line Rate [Gb/s]	Internal Data Width [16b/20b]	PLL Frequency [GHz]	REFCLK (Typical) [MHz]	Using Typical REFCLK Frequency			
					N1	N2	D	M
CPRI 1-10X (Multi-Rate)	6.144	20b	3.072	307.2	5	2	1	1
	4.9152	20b	2.4576	307.2	4	2	1	1
	3.072	20b	3.072	307.2	5	2	2	1
	2.4576	20b	2.4576	307.2	4	2	2	1
	1.2288	20b	2.4576	307.2	4	2	4	1
	0.6144	20b	1.2288	307.2	4	2	4	2
OBSAI (Multi-Rate)	3.072	20b	1.536	153.6	5	2	1	1
	1.536	20b	1.536	153.6	5	2	2	1
	0.768	20b	1.536	153.6	5	2	4	1
3G-SDI HD-SDI (Multi-Rate)	2.97	20b	1.485	148.5	5	2	1	1
	1.485	20b	1.485	148.5	5	2	2	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	1.24416	155.52	4	2	4	1
OTU-1	2.666057	16b	2.666057	166.6286	4	4	2	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 frequency is selected to use the minimum PLL multiplication ratio. 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 the *Virtex-6 FPGA Data Sheet* for the minimum and maximum reference clock frequencies.

Power Down

Functional Description

The GTX 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 transceiver offers different levels of power control. Each channel in each direction can be powered down separately using TXPOWERDOWN and RXPOWERDOWN. The TXPLLPOWERDOWN and RXPLLPOWERDOWN port directly affects the shared.

Ports and Attributes

Table 2-11 defines the power-down ports.

Table 2-11: **Power-Down Ports**

Port	Dir	Clock Domain	Description
RXPLLPOWERDOWN	In	Async	Input to power down the RX PLL.
RXPOWERDOWN[1:0]	In	Async	Powers down the RX lane according to the PCIe® protocol encoding. `00`: P0 (normal operation) `01`: P0s (low recovery time power down) `10`: P1 (longer recovery time) `11`: P2 (lowest power state)
TXPDOWNASYNCH	In	Async	Determines whether TXELECIDLE and TXPOWERDOWN should be treated as synchronous or asynchronous signals. `0`: Sets TXELECIDLE and TXPOWERDOWN to synchronous mode. `1`: Sets TXELECIDLE and TXPOWERDOWN to asynchronous mode.
TXPLLPOWERDOWN	In	Async	Input to power down the TX PLL.
TXPOWERDOWN[1:0]	In	TXUSRCLK2 (TXPDOWNASYNCH makes this pin asynchronous)	Powers down the TX lane according to the PCIe 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.

Table 2-12 defines the power-down attributes.

Table 2-12: **Power-Down Attributes**

Attribute	Type	Description
BGTEST_CFG	2-bit Hex	Reserved. Use only recommended values from the Virtex-6 FPGA GTX Transceiver Wizard.
BIAS_CFG	17-bit Hex	Reserved. Use only recommended values from the Virtex-6 FPGA GTX Transceiver Wizard.
PMA_TX_CFG	20-bit Hex	Reserved. Use only recommended values from the Virtex-6 FPGA GTX Transceiver Wizard.
POWER_SAVE	10-bit Binary	POWER_SAVE[4]: Mux select for the TXOUTCLK output clock. Must be tied to 1′b1. 1'b0: Use the TX Delay Aligner 1'b1: Bypass the TX Delay Aligner POWER_SAVE[5]: Mux select for the RXRECCLK output clock. Must be tied to 1′b1 when RX buffer is used (RX_BUFFER_USE = TRUE). When RX buffer is bypassed, refer to Using the RX Phase Alignment Circuit to Bypass the Buffer, page 234. 1'b0: Use the RX Delay Aligner 1'b1: Bypass the RX Delay Aligner All other bits are reserved. Use recommended values from the Virtex-6 FPGA GTX Transceiver Wizard.
TRANS_TIME_FROM_P2	12-bit Hex	Counter settings for programmable transition time from P2 state for PCIe operation. Use recommended values from the Virtex-6 FPGA GTX Transceiver Wizard.
TRANS_TIME_NON_P2	8-bit Hex	Counter settings for programmable transition time to/from all states except P2 for PCIe operation. Use recommended values from the Virtex-6 FPGA GTX Transceiver Wizard.
TRANS_TIME_RATE	8-bit Hex	Counter settings for programmable transition time when rate is changed using RATE pins for all protocols including the PCIe protocol (Gen2/Gen1 data rates). Set to the maximum value for non PCIe modes. Use recommended values from the Virtex-6 FPGA GTX Transceiver Wizard.
TRANS_TIME_TO_P2	10-bit Hex	Counter settings for programmable transition time to the P2 state for PCIe operation. Use recommended values from the Virtex-6 FPGA GTX Transceiver Wizard.

Generic Power-Down Capabilities

The GTX transceiver provides several power-down features that can be used in a wide variety of applications. Table 2-13 summarizes these capabilities.

Table 2-13: **Basic Power-Down Functions Summary**

Function	Controlled By	Affects
TX PLL Power Down	TXPLLPOWERDOWN	The TX of the GTX transceiver. Powers down the TX PLL as well as some of the TX PMA circuits.
RX PLL Power Down	RXPLLPOWERDOWN	The RX of the GTX transceiver. Powers down the RX PLL as well as some of the RX PMA circuits.
TX Power Down	TXPOWERDOWN[1:0]	The TX of the GTX transceiver.
RX Power Down	RXPOWERDOWN[1:0]	The RX of the GTX transceiver.

PLL Power Down

To activate the PLL power-down mode, the active-High TXPLLPOWERDOWN or RXPLLPOWER DOWN signal is asserted. When either PLLPOWERDOWN is asserted, the corresponding PMA PLL and some part of PMA circuits are powered down. As a result, all clocks derived from the PMA PLL are stopped.

Recovery from this power state is indicated by the assertion of corresponding PLL lock signal that is either TXPLLLKDET or RXPLLLKDET signal on the GTX transceiver.

TX and RX Power Down

When the TX and RX power control signals are used in non PCI Express implementations, TXPOWERDOWN and RXPOWERDOWN can be used independently. However, when these interfaces are used in non PCI Express applications, only two power states are supported, as shown in Table 2-14. When using this power-down mechanism, the following must be TRUE:

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

Table 2-14: **TX and RX Power States for Operation that are not for PCI Express Designs**

TXPOWERDOWN[1:0] or RXPOWERDOWN[1:0]	Description
`00`	Normal mode. The TX or RX is active sending or receiving data.
`11`	Power-down mode. The TX or RX is idle.

Power-Down Requirements for TX and RX Buffer Bypass

When bypassing the TX buffer, all of these requirements must be met:

- If the TX PLL supplies the clock for the TX datapath (TX_CLK_SOURCE = “TXPLL”), TXPLLPOWERDOWN must be tied Low.
- If the RX PLL supplies the clock for the TX datapath (TX_CLK_SOURCE = “RXPLL”), RXPLLPOWERDOWN must be tied Low.

Refer to TX Buffer Bypass, page 156 for more information.

When bypassing the RX buffer, all of these requirements must be met:

- RXPLLPOWERDOWN must be tied Low.
- RXPOWERDOWN[0] and RXPOWERDOWN[1] must be tied Low.

Refer to RX Buffer Bypass, page 230 for more information.

Power-Down Features for PCI Express Operation

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

- RXPOWERDOWN[1] must be tied Low.
- RXPOWERDOWN[0] must be tied to TXPOWERDOWN[0].
- The POWERDOWN[1:0] signal defined by the PIPE 2.0 Specification must be connected to TXPOWERDOWN[1:0].
- The TXPLLPOWERDOWN and RXPLLPOWERDOWN ports must be tied Low.

Table 2-15: **TX and RX Power States for PCI Express Operation**

TXPOWERDOWN[1:0] and RXPOWERDOWN[1:0]	TXDETECTRX	TXELECIDLE	Description
`00` (P0 State)	0	0	The PHY is transmitting data. The Media Access Layer (MAC) provides data bytes to be sent every clock cycle.
	0	1	The PHY is not transmitting and is in the electrical idle state.
	1	0	The PHY goes into loopback mode.
	1	1	Not permitted.
`01` (P0s state)	Don't Care	0	The MAC must always put the PHY into the electrical idle state while in P0s state. The PHY behavior is undefined if TXELECIDLE is deasserted while in P0s or P1.
		1	The PHY is not transmitting and is in the electrical idle state.
`10` (P1 state)(1)	Don't Care	0	Not permitted. The MAC must always put the PHY into the electrical idle state while in P1. The PHY behavior is undefined if TXELECIDLE is deasserted while in P0s or P1.
	0	1	The PHY is idle.
	1	1	The PHY does a receiver detection operation.
`11` (P2 state)(1)	Don't Care	0	The PHY transmits beacon signaling
		1	The PHY is idle.

Notes:

1. Transmitter only. The P1 and P2 power states are not supported by the receiver.

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 2-10 illustrates a loopback test configuration with four different loopback modes.

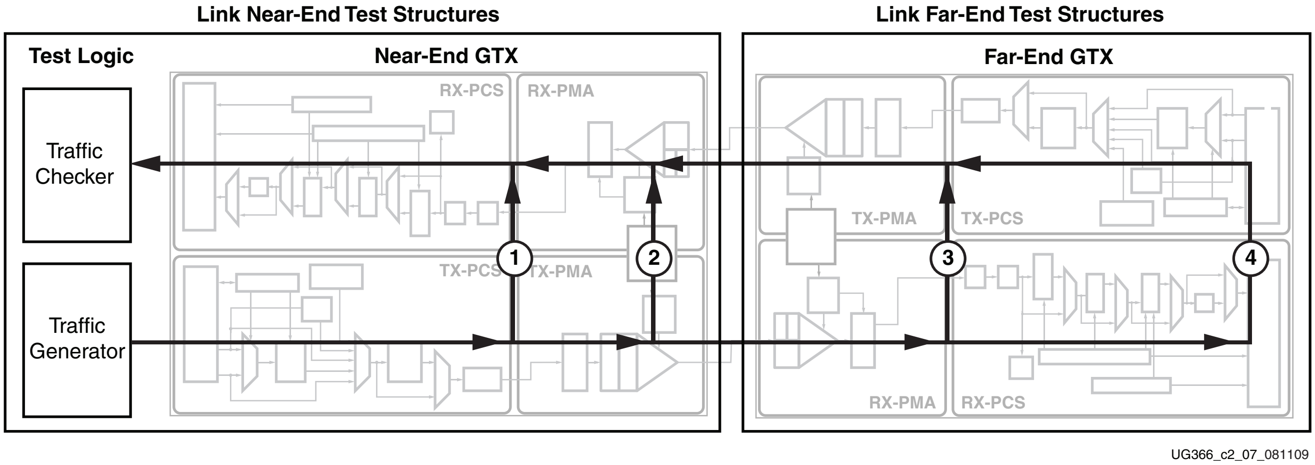


Figure 2-10: **Loopback Testing Overview**

Loopback test modes fall into two broad categories:

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

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

Each GTX transceiver features several loopback modes to facilitate testing:

- Near-End PCS Loopback (path 1 in Figure 2-10)
- Near-End PMA Loopback (path 2 in Figure 2-10)
- Far-End PMA Loopback (path 3 in Figure 2-10)
    - When in far-end PMA loopback mode, the receiver's recovered clock is used to clock the transmitter.
- Far-End PCS Loopback (path 4 in Figure 2-10)
    - When in far-end PCS loopback mode, the user must ascertain the transmitter is clocked by the same clock as the receiver, especially in asynchronous configurations.

Ports and Attributes

Table 2-16 defines the loopback ports.

Table 2-16: **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

Table 2-17 defines the loopback attributes.

Table 2-17: **Loopback Attributes**

Port	Type	Description
TXDRIVE_LOOPBACK_HIZ	Boolean	Reserved. Use only recommended values from the Virtex-6 FPGA GTX Transceiver Wizard.
TXDRIVE_LOOPBACK_PD	Boolean	Reserved. Use only recommended values from the Virtex-6 FPGA GTX Transceiver Wizard.

ACJTAG

Functional Description

The Virtex-6 FPGA GTX transceiver supports ACJTAG, as specified by IEEE Std 1149.6. To ensure reliable ACJTAG operation, the GTX transceiver RX expects the swing coming in to be 800 mV_{PPD} (400 mV_{PPSE}) or higher. In ACJTAG mode, the GTX transceiver TX swing is nominally 800 mV_{PPD}. For JTAG clock operating frequencies specifically in ACJTAG mode, refer to the *Virtex-6 FPGA Data Sheet*.

Dynamic Reconfiguration Port

Functional Description

The dynamic reconfiguration port (DRP) allows the dynamic change of parameters of the GTXE1 primitive. The DRP interface is a processor-friendly synchronous interface with an address bus (DADDR) and separated data buses for reading (DRPDO) and writing (DI) configuration data to the GTXE1 primitive. An enable signal (DEN), a read/write signal (DWE), and a ready/valid signal (DRDY) are the control signals that implement read and write operations, indicate operation completion, or indicate the availability of data. Refer to the *Virtex-6 FPGA Configuration User Guide* for detailed descriptions and timing diagrams of the DRP operations. Refer to Appendix B, DRP Address Map of the GTX Transceiver, for a DRP map of the GTX transceiver attributes.

Ports and Attributes

Table 2-18 defines the DRP ports.

Table 2-18: **DRP Ports**

Port	Dir	Clock Domain	Description
DADDR[7:0]	In	DCLK	DRP address bus.
DCLK	In	N/A	DRP interface clock.
DEN	In	DCLK	DRP enable signal. 0: No read or write operation performed. 1: Enables a read or write operation.
DI[15:0]	In	DCLK	Data bus for writing configuration data from the FPGA logic resources to the GTX transceiver.
DRDY	Out	DCLK	Indicates operation is complete for write operations and data is valid for read operations.
DRPDO[15:0]	Out	DCLK	Data bus for reading configuration data from the GTX transceiver to the FPGA logic resources.
DWE	In	DCLK	DRP write enable. 0: Read operation when DEN is 1. 1: Write operation when DEN is 1.

There are no DRP attributes.

Note: Attributes that have an impact on the entire Quad (the cluster of four GTX transceivers) are set by writing to the DRP of the first transceiver in the Quad. The first transceiver in the Quad has the lowest Y coordinates. Refer to Implementation, page 41 for details on transceiver placement and numbering.

Chapter 3

Transmitter

TX Overview

This chapter shows how to configure and use each of the functional blocks inside the GTX transceiver transmitter (TX). Each GTX 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 into the FPGA TX interface, through the PCS and PMA, and then out the TX driver as high-speed serial data.

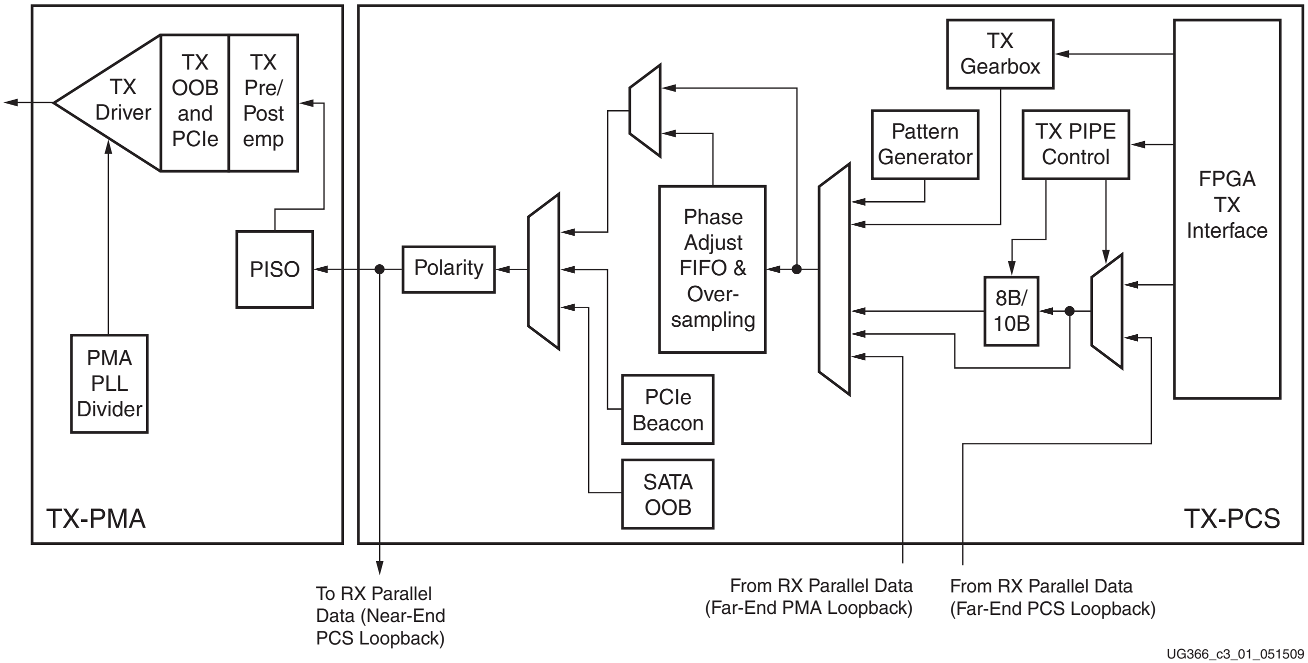


Figure 3-1: **GTX Transceiver TX Block Diagram**

The key elements of the GTX transceiver TX are:

1. FPGA TX Interface, page 128
2. TX Initialization, page 136
3. TX 8B/10B Encoder, page 143
4. TX Gearbox, page 146
5. TX Buffer, page 154
6. TX Buffer Bypass, page 156
7. TX Pattern Generator, page 163

8. TX Oversampling, page 167
9. TX Polarity Control, page 167
10. TX Fabric Clock Output Control, page 168
11. TX Configurable Driver, page 173
12. TX Receiver Detect Support for PCI Express Designs, page 180
13. TX Out-of-Band Signaling, page 181

FPGA TX Interface

Functional Description

The FPGA TX interface is the FPGA's gateway to the TX datapath of the GTX transceiver. Applications transmit data through the GTX transceiver by writing data to the TXDATA port on the positive edge of TXUSRCLK2. The width of the port can be configured to be one, two, or four bytes wide. The actual width of the port depends on the TX_DATA_WIDTH attribute and TXENC8B10BUSE port settings. Port widths can be 8, 10, 16, 20, 32, and 40 bits.

The rate of the parallel clock (TXUSRCLK2) at the interface is determined by the TX line rate, the width of the TXDATA port, and whether or not 8B/10B encoding is enabled. In some operating modes, 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 a 4-byte interface to achieve a TXUSRCLK2 rate in the specified operating range.

Interface Width Configuration

The Virtex®-6 FPGA GTX transceiver contains an internal 2-byte datapath. The FPGA interface width is configurable by setting the TX_DATA_WIDTH attribute. When the 8B/10B encoder is enabled, the FPGA interface must be configured to 10 bits, 20 bits, or 40 bits. When the 8B/10B encoder is bypassed, the FPGA interface is configured to any of the available widths: 8, 10, 16, 20, 32, or 40 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 143.

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

TXENC8B10BUSE	TX_DATA_WIDTH	FPGA Interface Width	Internal Data Width
1	10	8 bits	20 bits
	20	16 bits	20 bits
	40	32 bits	20 bits
0	8	8 bits	16 bits
	10	10 bits	20 bits
	16	16 bits	16 bits
	20	20 bits	20 bits
	32	32 bits	16 bits
	40	40 bits	20 bits

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

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

	< < Data transmission order is right to left < <																																							
	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]	TXCHARDIPSVAL[3]	TXDATA[31:24]								TXCHARDISPMODE[2]	TXCHARDIPSVAL[2]	TXDATA[23:16]								TXCHARDISPMODE[1]	TXCHARDIPSVAL[1]	TXDATA[15:8]								TXCHARDISPMODE[0]	TXCHARDIPSVAL[0]	TXDATA[7:0]							

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 transceiver transmitter. The required rate for TXUSRCLK depends on the internal datapath width of the `GTXE1` primitive and the TX line rate of the GTX transceiver transmitter. Equation 3-1 shows how to calculate the required rate for TXUSRCLK.

$$TXUSRCLK\ Rate = \frac{Line\ Rate}{Internal\ Datapath\ Width}$$

Equation 3-1

TXUSRCLK can be generated internally to the GTX transceiver. This functionality is controlled by the GEN_TXUSRCLK attribute. Table 3-3 describes the situations in which the TXUSRCLK can be generated internally by the GTX transceiver. In these cases, the TXUSRCLK port must be tied Low.

Table 3-3: **TXUSRCLK Internal Generation Configurations**

	TX_DATA_WIDTH	GTX Lanes in Channel[(1)]	GEN_TXUSRCLK
1-Byte	8, 10	1	TRUE
		2 or more	FALSE
2-Byte	16, 20	1 or more	TRUE
4-Byte	32, 40	1 or more	FALSE

Notes:

1. For single lane protocols such as 1 Gb/s Ethernet, “GTX Lanes in Channel” is 1. For multiple lane protocols like XAUI, “GTX Lanes in Channel” is 2 or more.

TXUSRCLK2 is the main synchronization clock for all signals into the TX side of the GTX transceiver. Most signals into the TX side of the GTX transceiver are sampled on the positive edge of TXUSRCLK2. TXUSRCLK2 and TXUSRCLK have a fixed-rate relationship based on the TX_DATA_WIDTH setting. Table 3-4 shows the relationship between TXUSRCLK2 and TXUSRCLK per TX_DATA_WIDTH values.

Table 3-4: **TXUSRCLK2 Frequency Relationship to TXUSRCLK**

	TX_DATA_WIDTH	TXUSRCLK2 Frequency
1-Byte	8, 10	$F_{TXUSRCLK2} = 2 \times F_{TXUSRCLK}$
2-Byte	16, 20	$F_{TXUSRCLK2} = F_{TXUSRCLK}$
4-Byte	32, 40	$F_{TXUSRCLK2} = F_{TXUSRCLK} / 2$

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. Table 3-3 and Table 3-4 describe the appropriate GEN_TXUSRCLK setting and TXUSRCLK2 frequency requirements. In cases where TXUSRCLK is generated by the user, the designer must ensure that TXUSRCLK and TXUSRCLK2 are positive-edge aligned.
- 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 3-5 defines the FPGA TX interface ports.

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

Port	Dir	Clock Domain	Description
MGTREFCLKFAB[1:0]	Out	Clock	Reserved. Do not use this port.
TXCHARDISPMODE[3:0]	In	TXUSRCLK2	When 8B/10B encoding is disabled, TXCHARDISPMODE is used to extend the data bus for 10- and 20-bit TX interfaces.
TXCHARDISPVAL[3:0]	In	TXUSRCLK2	When 8B/10B encoding is disabled, TXCHARDISPVAL is used to extend the data bus for 10- and 20-bit TX interfaces.

TXDATA[31:0]	In	TXUSRCLK2	The bus for transmitting data. The width of this port depends on TX_DATA_WIDTH:
			TXDATAWIDTH = 8,10: TXDATA[7:0] = 8 bits wide
			TXDATAWIDTH = 16,20: TXDATA[15:0] = 16 bits wide
			TXDATAWIDTH = 32,40: TXDATA[31:0] = 32 bits wide
			When a 10-bit, 20-bit, or 40-bit bus is required, the TXCHARDISPVAL and TXCHARDISPMODE ports from the 8B/10B encoder are concatenated with the TXDATA port. See Table 3-2.

TXUSRCLK	In	Clock	This port is used to provide a clock for the internal TX PCS datapath. In some use cases, this clock is internally generated. See Table 3-3.
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 3-6 defines the FPGA TX interface attributes.

Table 3-6: **FPGA TX Interface Attributes**

Attribute	Type	Description
GEN_TXUSRCLK	Boolean	Controls internal generation of TXUSRCLK available in certain modes of operation. See TXUSRCLK and TXUSRCLK2 Generation, page 129 for more details. TRUE: TXUSRCLK internally generated. TXUSRCLK must be tied Low. FALSE: TXUSRCLK must be provided by user.
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 10, 20, or 40. Valid settings are 8, 10, 16, 20, 32, and 40. See Interface Width Configuration, page 128 for more details.

Using TXOUTCLK to Drive the GTX Transceiver TX

Figure 3-2 through Figure 3-7 show different ways that FPGA clock resources can be used to drive the parallel clocks for the TX interface. In these examples, the TXOUTCLK port is derived from MGTREFCLK0[P/N] or MGTREFCLK1[P/N], and the TXOUTCLK_CTRL attribute is set to “TXPLLREFCLK_DIV1”.

- Depending on the input reference clock frequency and the required line rate, an MMCM and the appropriate TXOUTCLK_CTRL attribute setting is required. The CORE Generator™ tool creates a sample design based on different design requirements for most cases.
- When the TXOUTCLK_CTRL attribute is set to “TXOUTCLKPMA_DIV1”, the MMCM is required.
- In use models where the TX buffer is bypassed, there are additional restrictions on the clocking resources. Refer to TX Buffer Bypass, page 156 for more information.

TXOUTCLK Driving a GTX Transceiver TX in 2-Byte Mode (Single Lane)

In Figure 3-2, TXOUTCLK is used to drive TXUSRCLK2 for 2-byte mode (TX_DATA_WIDTH = 16 or 20). The GEN_TXUSRCLK attribute is set to “TRUE”, and the TXUSRCLK input port is tied to ground. TXUSRCLK is internally generated for the internal TX PCS datapath.

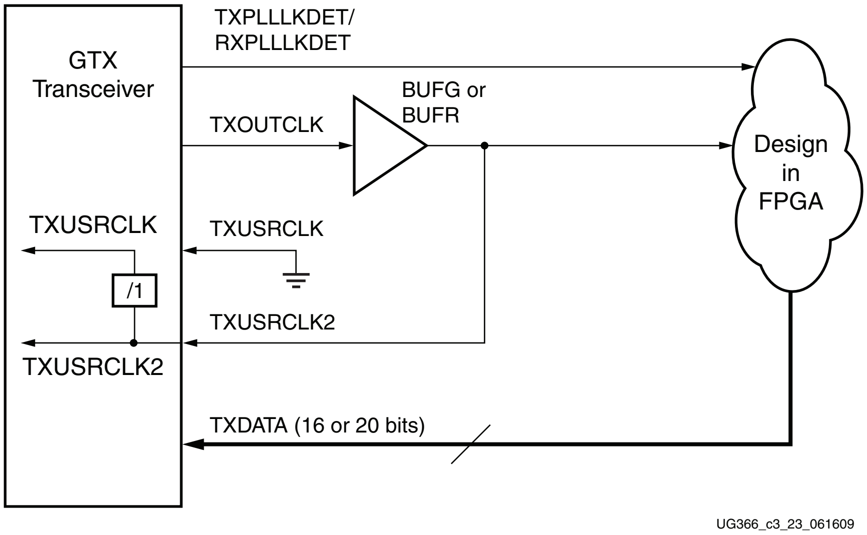


Figure 3-2: **TXOUTCLK Drives TXUSRCLK2 (2-Byte Mode)**

Refer to the *Virtex-6 FPGA Data Sheet* for the maximum clock frequency and jitter limitations of BUFR. For details about placement constraints and restrictions on clocking resources (MMCM, BUFGCTRL, IBUFDS_GTXE1, BUFG, etc.), refer to the *Virtex-6 FPGA Clocking Resources User Guide*.

TXOUTCLK Driving a GTX Transceiver TX in 4-Byte Mode (Single Lane)

Figure 3-3 uses 4-byte wide datapaths (TX_DATA_WIDTH = 32 or 40). TXOUTCLK is used to drive the CLKIN of the MMCM to derive two positive-edge aligned CLKOUT0 and CLKOUT1 signals, where the CLKOUT1 frequency is equal to the CLKOUT0 frequency divided by 2. If the TX PLL is not used and is derived from the RX PLL, the active-High RXPLLLKDET signal should be used to deassert the RST signal of the MMCM. TXOUTCLK can be used to drive CLKIN directly without using the BUFG resources. In the use models where TX Buffer is bypassed, TXOUTCLK must drive CLKIN directly. This requires the MMCM to be placed in the same clock region as the driving GTX transceiver.

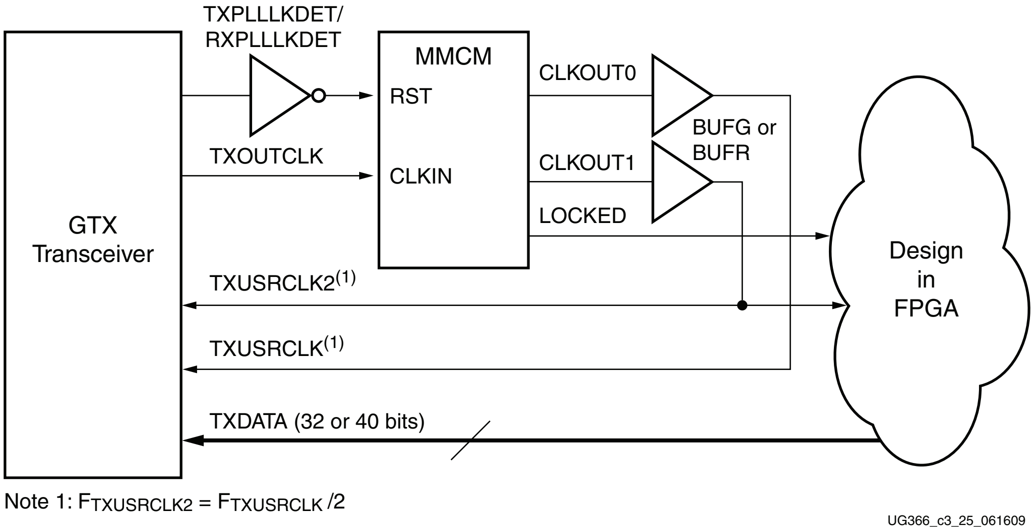
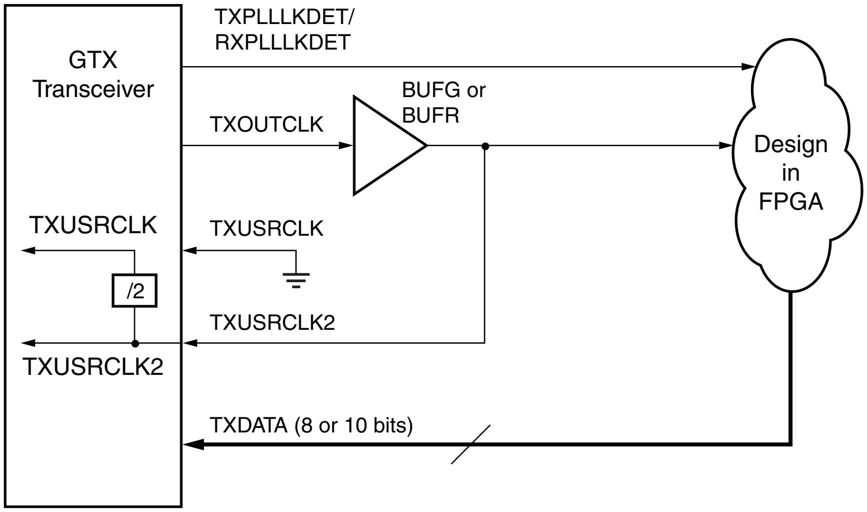


Figure 3-3: **MMCM Provides Clocks for 4-Byte Wide Datapath**

Refer to the *Virtex-6 FPGA Data Sheet* for the maximum clock frequency and jitter limitations of BUFR. For details about placement constraints and restrictions on clocking resources (MMCM, BUFGCTRL, IBUFDS_GTXE1, BUFG, etc.), refer to the *Virtex-6 FPGA Clocking Resources User Guide*.

TXOUTCLK Driving a GTX Transceiver TX in 1-Byte Mode (Single Lane)

In Figure 3-4, TXOUTCLK is used to drive TXUSRCLK2 for 1-byte mode (TX_DATA_WIDTH = 8 or 10). The GEN_TXUSRCLK attribute is set to “TRUE”, and the TXUSRCLK input port is tied to ground. TXUSRCLK is internally generated by dividing TXUSRCLK2 by 2 for the internal TX PCS datapath.



UG366_c3_21_061609

Figure 3-4: **TXOUTCLK Drives TXUSRCLK2 (1-Byte Mode)**

Refer to the *Virtex-6 FPGA Data Sheet* for the maximum clock frequency and jitter limitations of BUFR. For details about placement constraints and restrictions on clocking resources (MMCM, BUFGCTRL, IBUFDS_GTXE1, BUFG, etc.), refer to the *Virtex-6 FPGA Clocking Resources User Guide*.

TXOUTCLK Driving More Than One GTX Transceiver TX in 2-Byte Mode (Multiple Lanes)

In Figure 3-5, TXOUTCLK is used to drive multiple GTX transceiver user clocks. In this situation, the frequency must be correct for all the GTX transceivers, and they must share the same reference clock. In 2-byte mode (TX_DATA_WIDTH = 16 or 20), the GEN_TXUSRCLK attribute is set to “TRUE”, and the TXUSRCLK input port is tied to ground. TXUSRCLK is internally generated for the internal TX PCS datapath. The user can use either TXPLLKDET or RXPLLLKDET as a reset signal for the design in the FPGA. If the TX PLL is not used and is derived from the RX PLL, the active-High RXPLLLKDET signal should be used instead.

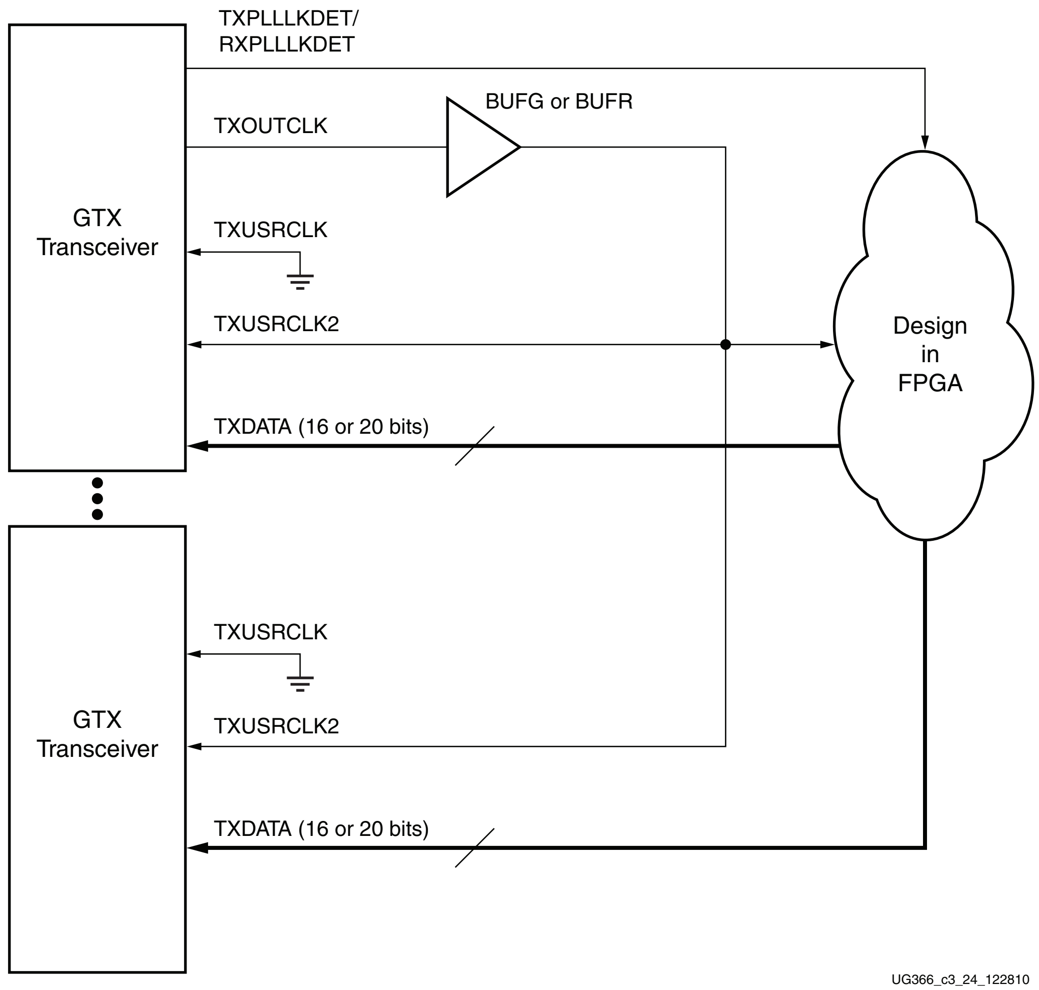


Figure 3-5: **TXOUTCLK Drives TXUSRCLK2 (2-Byte Mode)**

For details about placement constraints and restrictions on clocking resources (MMCM, BUFGCTRL, IBUFDS_GTXE1, BUFG, etc.), refer to the *Virtex-6 FPGA Clocking Resources User Guide*.

TXOUTCLK Driving More Than One GTX Transceiver TX in 4-Byte Mode (Multiple Lanes)

In Figure 3-6, TXOUTCLK is used to drive multiple GTX transceiver user clocks. In this case, the frequency must be correct for all the GTX transceivers, and they must share the same reference clock. In 4-byte mode (TX_DATA_WIDTH = 32 or 40), the GEN_TXUSRCLK attribute is set to “FALSE”, and TXOUTCLK is used as the reference clock for the MMCM.

TXOUTCLK is used to drive the CLKIN signal of the MMCM to derive two positive-edge aligned CLKOUT0 and CLKOUT1 signals, where the CLKOUT1 frequency is equal to the CLKOUT0 frequency divided by 2. The user can use either TXPLLKDET or RXPLLLKDET as a reset signal for the MMCM. If the TX PLL for each transceiver is not used and is derived from the RX PLL, the active-High RXPLLLKDET signal should be used to deassert the RST signal of the MMCM. TXOUTCLK can be used to drive CLKIN directly without using the BUFG resources. In the use models where TX Buffer is bypassed, TXOUTCLK must drive CLKIN directly. This requires the MMCM to be placed in the same clock region as the driving GTX transceiver.

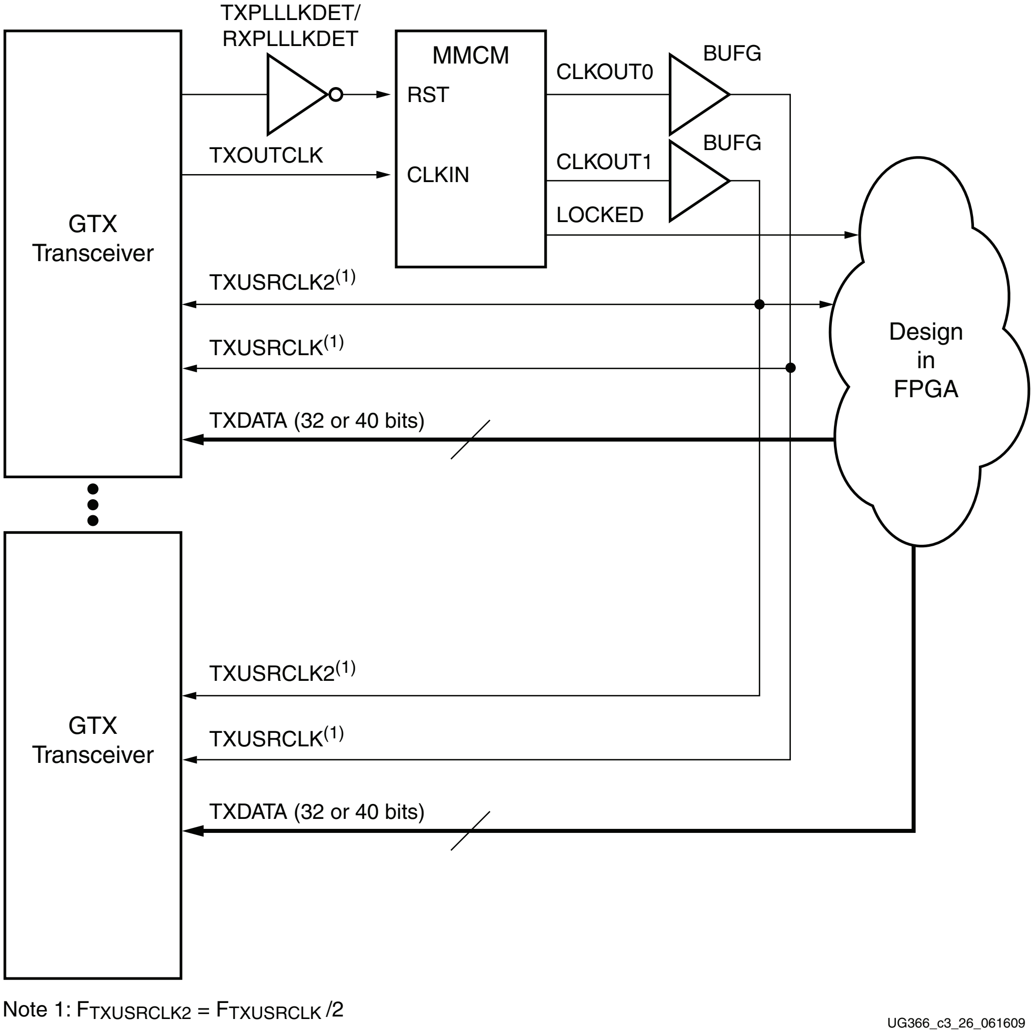


Figure 3-6: **TXOUTCLK Driving More Than One GTX Transceiver TX in 4-Byte Mode**

For details about placement constraints and restrictions on clocking resources (MMCM, BUFGCTRL, IBUFDS_GTXE1, BUFG, etc.), refer to the *Virtex-6 FPGA Clocking Resources User Guide*.

TXOUTCLK Driving More Than One GTX Transceiver TX in 1-Byte Mode (Multiple Lanes)

In Figure 3-7, TXOUTCLK is used to drive multiple GTX transceiver user clocks. In this situation, the frequency must be correct for all the GTX transceivers, and they must share the same reference clock. The example shows 1-byte mode (TX_DATA_WIDTH = 8 or 10), the GEN_TXUSRCLK attribute is set to "FALSE", and TXOUTCLK is used as the reference clock for the MMCM.

TXOUTCLK is used to drive the CLKIN signal of the MMCM to derive two positive-edge aligned CLKOUT0 and CLKOUT1 signals, where the CLKOUT1 frequency is equal to CLKOUT0 frequency multiplied by 2. If the TX PLL for each transceiver is not used and derived from the RX PLL, the active-High RXPLLLKDET signal should be used to deassert the RST signal of the MMCM. TXOUTCLK can be used to drive CLKIN directly without using the BUFG resources.

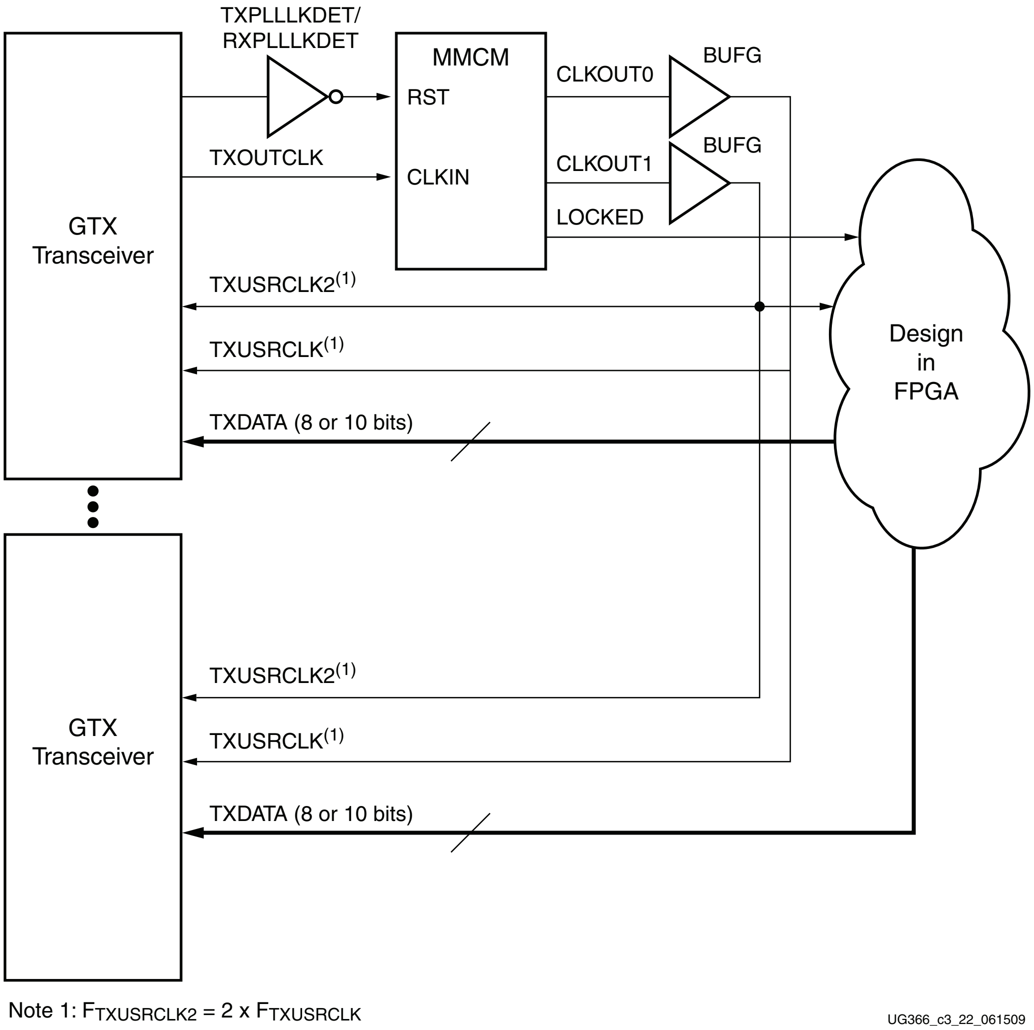


Figure 3-7: **TXOUTCLK Driving More Than One GTX Transceiver TX in 1-Byte Mode**

For details about placement constraints and restrictions on clocking resources (MMCM, BUFGCTRL, IBUFDS_GTXE1, BUFG, etc.), refer to the *Virtex-6 FPGA Clocking Resources User Guide*.

TX Initialization

Functional Description

The GTX transceiver TX must be reset before it can be used. There are three ways to reset the GTX transceiver TX:

1. Power up and configure the FPGA. Power-up reset is covered in this section.
2. Drive the GTXTXRESET port High to trigger a full asynchronous reset of the GTX transceiver TX.

3. Assert one or more of the individual reset signals on the block to reset a specific subcomponent of the transmitter. These resets are covered in detail in the sections for each subcomponent (refer to Table 3-9, page 141 for a list of the available transmitter resets).

All reset ports described in this section initiate the internal transmitter state machines when driven High. The internal reset state machines are held in the reset state until these same reset ports are driven Low. The completion of these state machines is signaled through the TXRESETDONE port.

Figure 3-8 shows the GTX transceiver TX reset hierarchy.

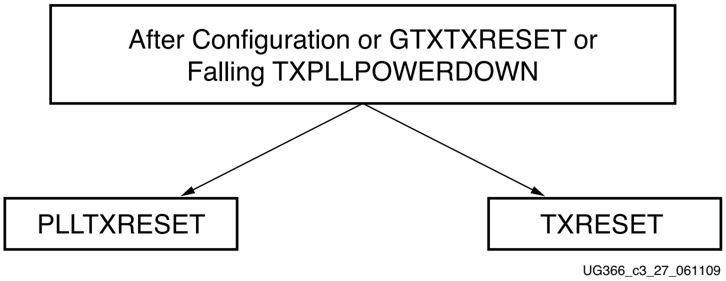


Figure 3-8: **GTX Transceiver TX Reset Hierarchy**

When bypassing the TX buffer, GTXTXRESET, GTXRXRESET, PLLTXRESET, and PLLRXRESET must not be tied High. Refer to TX Buffer Bypass, page 156 for more information.

The GTX transceiver TX can use either TX PLL or RX PLL. Whichever PLL is used, if TXPLL_DIVSEL_OUT is set to /2 or /4, the TX output clock divider must be reset twice when the associated PLLLKDET signal goes from Low to High. Resetting the TX output clock divider twice (double reset) is achieved through the GTXTEST[1] port, and is required under these conditions:

- When the TX output clock divider, TXPLL_DIVSEL_OUT, is set to /2 or /4
- After FPGA power-up and configuration
- After the associated reset of the PLL used by the TX is toggled
- After turning on a reference clock to the PLL used by the TX
- After changing the reference clock to the PLL used by the TX
- After assertion/deassertion of TXPOWERDOWN
- After assertion/deassertion of GTXTXRESET

The circuit that implements the double reset through GTXTEST[1] must use a free running clock. This clock can be sourced from:

- IBUFDS output. This also provides the MGTREFCLK to the GTX transceiver.
- DRP clock.
- Any other free running clock from the user design.

The circuit must follow the timing diagram shown in Figure 3-9.

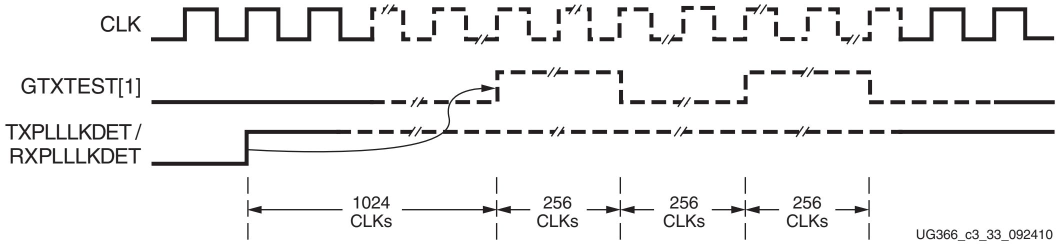


Figure 3-9: **GTXTEST[1] Double Reset Timing Diagram**

Notes relevant to Figure 3-9:

- The minimum wait time from the rising edge of the TXPLLLKDET/RXPLLLKDET signals to the first GTXTEST[1] reset pulse is 1,024 clock cycles.
- The minimum GTXTEST[1] pulse duration is 256 clock cycles.
- The minimum wait time between two GTXTEST[1] High pulses is 256 clock cycles.

If the transmitter uses TXPLL, the rising edge of TXPLLLKDET must be used to trigger GTXTEST[1]. If the transmitter uses RXPLL, RXPLLLKDET must be used to trigger GTXTEST[1].

TXRESET and RXRESET must be applied after GTXTEST[1] double reset. The guideline for the asynchronous TXRESET and RXRESET pulse width is one period of the reference clock.

Ports and Attributes

Table 3-7 defines the TX initialization ports.

Table 3-7: **TX Initialization Ports**

Port	Dir	Clock Domain	Description
GTXTEST[12:0]	In	Async	GTXTEST[0]: Reserved. Tied to `0`. GTXTEST[1]: The default is `0`. When this bit is set to `1`, the TX output clock dividers are reset. GTXTEST[12:2]: Reserved. Tied to `10000000000`.
GTXTXRESET	In	Async	This port is driven High and then deasserted to start the full TX GTX transceiver reset sequence. This sequence takes about 120 µs to complete and systematically resets all subcomponents of the GTX transceiver TX. If the RX PLL is supplying the clock for the TX datapath, GTXTXRESET and GTXRXRESET must be tied together. In addition, the transmitter reference clock must also be supplied (see Reference Clock Selection, page 102).
PLLTXRESET	In	Async	This port resets the TX PLL of the GTX transceiver when driven High. It affects the clock generated from the TX PMA. When this reset is asserted or deasserted, TXRESET must also be asserted or deasserted.
TSTIN[19:0]	In	Async	Reserved. Must be tied to `11111111111111111111`.
TXDLYALIGNRESET	In	Async	This port resets the TX delay aligner for the TX buffer bypass mode. See TX Buffer Bypass, page 156.

Table 3-7: **TX Initialization Ports *(Cont'd)***

Port	Dir	Clock Domain	Description
TXRESET	In	Async	PCS TX system reset. Resets the TX FIFO, 8B/10B encoder and other transmitter registers. This reset is a subset of GTXTXRESET.
TXRESETDONE	Out	Async	This port goes High when the GTX transceiver TX has finished reset and is ready for use. For this signal to work correctly, the TX reference clock and all clock inputs on the individual GTX transceiver (TXUSRCLK, TXUSRCLK2) must be driven.

Table 3-8 defines the TX initialization attributes.

Table 3-8: **TX Initialization Attributes**

Attribute	Type	Description
TX_EN_RATE_RESET_BUF	Boolean	When set to TRUE, this attribute enables automatic TX buffer reset during a rate change event initiated by a change in TXRATE[1:0].

GTX Transceiver TX Reset in Response to Completion of Configuration

Figure 3-10 shows the GTX transceiver TX reset following the completion of configuration of a powered-up GTX transceiver. The same sequence is activated any time TXPLLPOWERDOWN goes from High to Low during normal operation.

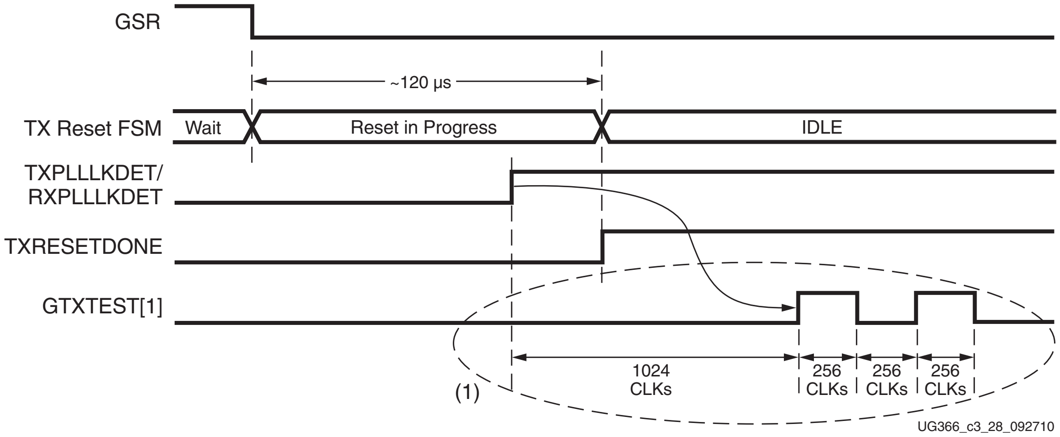


Figure 3-10: **Transmitter Reset After Completion of Configuration**

Notes relevant to Figure 3-10:

1. GTXTEST[1] is only required when the TX output clock divider, TXPLL_DIVSEL_OUT, is set to /2 or /4.
2. The timing of the reset sequencer inside the GTX transceiver TX depends on the frequency of an internal clock and certain configuration attributes. The estimate given in Figure 3-10 assumes that the frequency of the internal clock is 50 MHz with default values for the configuration attributes.
3. TXRESET and RXRESET must be applied after GTXTEST[1] double reset. The guideline for the asynchronous TXRESET and RXRESET pulse width is one period of the reference clock.

GTX Transceiver TX Reset in Response to GTXTXRESET Pulse

Figure 3-11, similar to Figure 3-10, shows the reset occurring in response to a pulse on GTXTXRESET. GTXTXRESET acts as an asynchronous reset signal. The guideline for the asynchronous GTXTXRESET pulse width is one period of the reference clock.

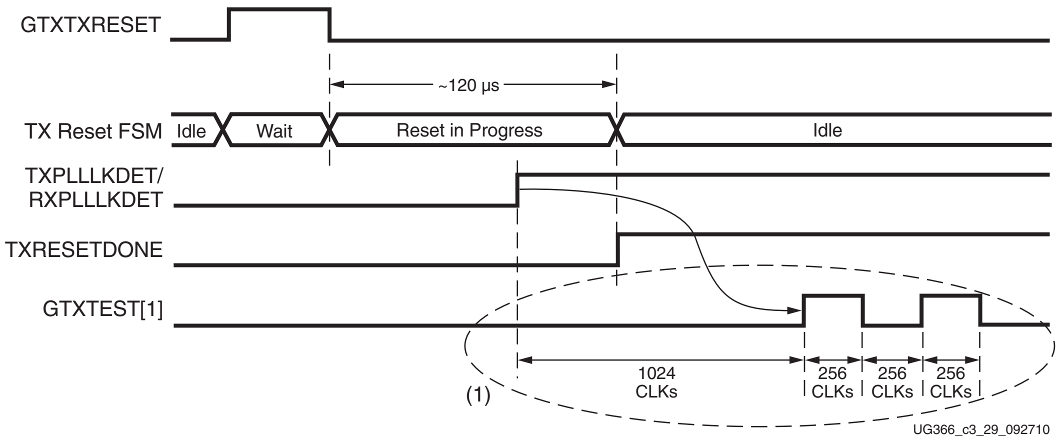


Figure 3-11: **Transmitter Reset After GTXTXRESET Pulse**

Notes relevant to Figure 3-11:

1. GTXTEST[1] is only required when the TX output clock divider, TXPLL_DIVSEL_OUT, is set to /2 or /4.
2. The timing of the reset sequencer inside the GTX transceiver TX depends on the frequency of an internal clock and certain configuration attributes. The estimate given in Figure 3-11 assumes that the frequency of the internal clock is 50 MHz with default values for the configuration attributes.
3. The entire GTX transceiver TX is affected by GTXTXRESET. If the RX PLL is supplying the clock for the TX datapath, GTXTXRESET and GTXRXRESET must be tied together.
4. TXRESET and RXRESET must be applied after GTXTEST[1] double reset. The guideline for the asynchronous TXRESET and RXRESET pulse width is one period of the reference clock.

GTX Transceiver TX Component-Level Resets

GTX transceiver TX component resets are primarily used for special cases. These resets are needed when only the reset of a specific subsection is required. Each of the component-level reset signals is described in Table 4-52, page 261.

All transmitter component resets are asynchronous. Table 3-9 summarizes the transmitter resets and the components that they reset.

Table 3-9: **Available Transmitter Resets and the Components Reset by Them**

	Component	Configuration	GTXTXRESET	TXPLLPOWERDOWN (Falling Edge)	PLLTXRESET	TXRESET	TXDLYALIGNRESET
TX PCS	FPGA TX Interface	✓	✓	✓		✓	
	TX Gearbox	✓	✓	✓		✓	
	TX Buffer	✓	✓	✓		✓	
	8B/10B Encoder	✓	✓	✓		✓	
	TX Polarity	✓	✓	✓		✓	
	Pattern Generator	✓	✓	✓		✓	
	5x Oversampler	✓	✓	✓		✓	
	TX Delay Aligner	✓					✓
TX PMA	TX Driver	✓	✓	✓			
	TX OOB	✓	✓	✓			
	TX Receiver Detect for PCIe Designs	✓	✓	✓			
	TX PLL	✓	✓	✓	✓		
	PISO	✓	✓	✓			
Loopback	Loopback Paths	✓	✓	✓			

Table 3-10 lists the recommended resets for various situations.

Table 3-10: **Recommended Resets for Common Situations**

Situation	Components to be Reset	Recommended Reset[1]
After power up and configuration	Entire GTX transceiver TX	After configuration, GTX transceiver TX is reset automatically
After turning on a reference clock to the TX PLL	Entire GTX transceiver TX	GTXTXRESET
After changing the reference clock to the TX PLL	Entire GTX transceiver TX	GTXTXRESET

Table 3-10: **Recommended Resets for Common Situations *(Cont'd)***

Situation	Components to be Reset	Recommended Reset[1]
After assertion/deassertion of TXPOWERDOWN	Entire GTX transceiver TX	GTXTXRESET
TX rate change with the TX buffer bypassed	TX PCS, TX Phase Alignment	GTXTEST[1], TXRESET
TX rate change with TX buffer enabled	TX PLL Output Clock Dividers, TX PCS	GTXTEST[1], TXRESET
TX parallel clock source reset	TX PLL Output Clock Dividers, TX Delay Aligner, TX Phase Alignment, TX PCS	GTXTEST[1], TXDLYALIGNRESET, TXRESET

Notes:

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

See TX Buffer Bypass, page 156 for details on the rate change procedure.

After Power-up and Configuration

The entire GTX transceiver TX is reset automatically after configuration-provided TXPLLPOWERDOWN is Low. The supplies for the calibration resistor and calibration resistor reference must be powered up before configuration to ensure correct calibration of the termination impedance of all transceivers.

After Turning on a Reference Clock to the TX PLL

The reference clock source(s) and the power to the GTX transceiver(s) must be available before configuring the FPGA. The reference clock must be stable before configuration especially when using PLL-based clock sources (e.g., voltage controlled crystal oscillators). If the reference clock(s) changes or GTX transceiver(s) are powered up after configuration, GTXTXRESET is asserted to allow the TX PLL(s) to lock.

After Changing the Reference Clock to the TX PLL

Whenever the reference clock input to the TX PLL is changed, the TX PLL must be reset afterwards to ensure that it locks to the new frequency. The GTXTXRESET port must be used for this purpose. Refer to Reference Clock Selection, page 102 for more details.

After Assertion/Deassertion of TXPOWERDOWN

After the TXPOWERDOWN signal is deasserted, GTXTXRESET must be asserted.

TX Rate Change with the TX Buffer Enabled

After TXRATEDONE is asserted, indicating the rate change has completed, the TX PLL output clock dividers must be reset followed by a TX PCS reset. To reset the clock dividers, GTXTEST[1] is asserted for at least 16 TXUSRCLK2 cycles. To reset the TX PCS, TXRESET is asserted.

To automatically reset the TX buffer after the rate change, the TX_EN_RATE_RESET_BUF attribute is set to "TRUE."

TX Rate Change with the TX Buffer Bypassed

After TXRATEDONE is asserted, indicating the rate change has completed, the TX PLL output clock dividers must be reset. Phase alignment must be performed again followed by reset of the TX PCS. See TX Buffer Bypass, page 156 for details on the rate change procedure.

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, TXRESET must be used to hold TX PCS in reset until the clock source is locked again.

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

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

8B/10B Bit and Byte Ordering

8B/10B encoding requires bit a0 to be transmitted first, and the GTX transceiver always transmits the right-most bit first. To match with 8B/10B, the 8B/10B encoder in the GTX transceiver automatically reverses the bit order (Figure 3-12).

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

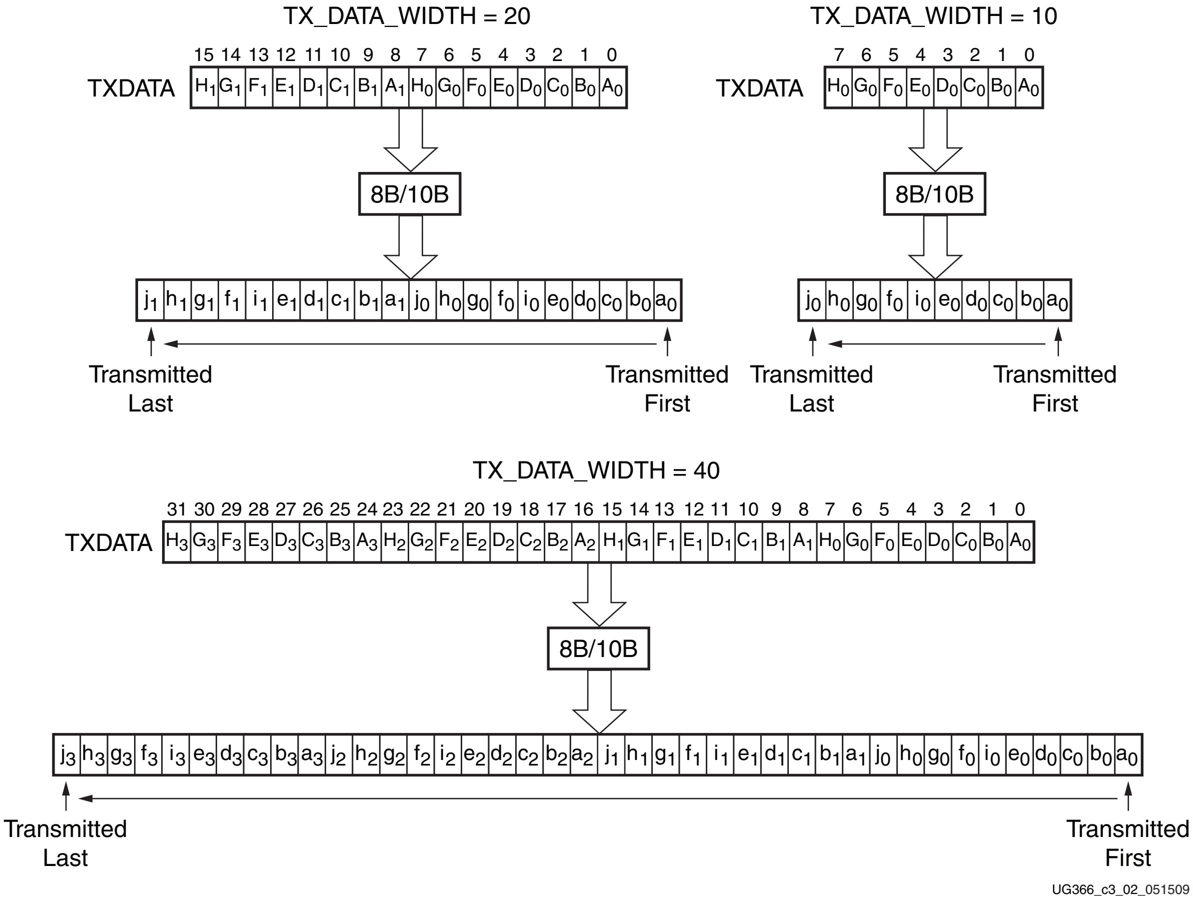


Figure 3-12: **8B/10B Encoding**

K Characters

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

Running Disparity

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

Normally, running disparity is used to determine whether a positive or negative 10-bit code is transmitted next. The encoder allows the next disparity value to be controlled directly as well, to accommodate protocols that use disparity to send control information. For example, an Idle character sent with reversed disparity might be used to trigger clock correction. Table 3-11 shows how the TXCHARDISPMODE and TXCHARDISPVAL ports are used to control outgoing disparity values.

Table 3-11: **TXCHARDISPMODE and TXCHARDISPVAL vs. Outgoing Disparity**

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

Ports and Attributes

Table 3-12 defines the TX encoder ports.

Table 3-12: **TX Encoder Ports**

Port	Dir	Clock Domain	Description
TXBYPASS8B10B[3:0]	In	TXUSRCLK2	TXBYPASS8B10B controls the operation of the TX 8B/10B encoder on a per-byte basis. It is only effective when TXENC8B10B is High (8B/10B is enabled) and RX_DATA_WIDTH is {10, 20, 40}. See Table 4-56, page 269 for additional information on RX_DATA_WIDTH. TXBYPASS8B10B[3] corresponds to TXDATA[31:24] TXBYPASS8B10B[2] corresponds to TXDATA[23:16] TXBYPASS8B10B[1] corresponds to TXDATA[15:8] TXBYPASS8B10B[0] corresponds to TXDATA[7:0] TXBYPASS8B10B[x] = 1, encoder for byte x is bypassed TXBYPASS8B10B[x] = 0, encoder for byte x is used
TXCHARDISPMODE[3:0]	In	TXUSRCLK2	TXCHARDISPMODE and TXCHARDISPVAL allow the 8B/10B disparity of outgoing data to be controlled when 8B/10B encoding is enabled. When 8B/10B encoding is disabled, TXCHARDISPMODE is used to extend the data bus for TX interfaces with a width that is a multiple of 10. TXCHARDISPMODE[3] corresponds to TXDATA[31:24] TXCHARDISPMODE[2] corresponds to TXDATA[23:16] TXCHARDISPMODE[1] corresponds to TXDATA[15:8] TXCHARDISPMODE[0] corresponds to TXDATA[7:0]
TXCHARDISPVAL[3:0]	In	TXUSRCLK2	TXCHARDISPVAL and TXCHARDISPMODE allow the 8B/10B disparity of outgoing data disparity to be controlled when 8B/10B encoding is enabled. When 8B/10B encoding is disabled, TXCHARDISPVAL is used to extend the data bus for 10- and 20-bit TX interfaces (see FPGA TX Interface, page 128). TXCHARDISPVAL[3] corresponds to TXDATA[31:24] TXCHARDISPVAL[2] corresponds to TXDATA[23:16] TXCHARDISPVAL[1] corresponds to TXDATA[15:8] TXCHARDISPVAL[0] corresponds to TXDATA[7:0]

Table 3-12: **TX Encoder Ports *(Cont'd)***

Port	Dir	Clock Domain	Description
TXCHARISK[3:0]	In	TXUSRCLK2	TXCHARISK is set High to send TXDATA as an 8B/10B K character. TXCHARISK should only be asserted for TXDATA values representing valid K-characters. TXCHARISK[3] corresponds to TXDATA[31:24] TXCHARISK[2] corresponds to TXDATA[23:16] TXCHARISK[1] corresponds to TXDATA[15:8] TXCHARISK[0] corresponds to TXDATA[7:0] TXCHARISK is undefined for bytes that bypass 8B/10B encoding.
TXENC8B10BUSE	In	TXUSRCLK2	TXENC8B10BUSE is set High to enable the 8B/10B encoder. TX_DATA_WIDTH must be set to 10, 20, or 40 when the 8B/10B encoder is enabled. `0`: 8B/10B encoder bypassed. This option reduces latency. `1`: 8B/10B encoder enabled.
TXKERR[3:0]	Out	TXUSRCLK2	TXKERR indicates if an invalid code for a K character was specified. TXKERR[3] corresponds to TXDATA[31:24] TXKERR[2] corresponds to TXDATA[23:16] TXKERR[1] corresponds to TXDATA[15:8] TXKERR[0] corresponds to TXDATA[7:0]
TXRUNDISP[3:0]	Out	TXUSRCLK2	TXRUNDISP indicates the current running disparity of the 8B/10B encoder. This disparity corresponds to TXDATA clocked in several cycles earlier. TXRUNDISP[3] corresponds to previous TXDATA[31:24] data TXRUNDISP[2] corresponds to previous TXDATA[23:16] data TXRUNDISP[1] corresponds to previous TXDATA[15:8] data TXRUNDISP[0] corresponds to previous TXDATA[7:0] data

There are no TX encoder attributes.

Enabling and Disabling 8B/10B Encoding

To enable the 8B/10B encoder, TXENC8B10BUSE must be driven High. To disable the 8B/10B encoder on a given GTX transceiver, TXENC8B10BUSE must be driven Low. When the encoder is turned off, the operation of the TXDATA port is as described in FPGA TX Interface, page 128.

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.interlakenalliance.com/. The TX gearbox only supports 2-byte and 4-byte interfaces. A 1-byte interface is not supported.

Scrambling of the data is done in the FPGA logic. The Virtex-6 FPGA GTX Transceiver Wizard has example code for the scrambler.

Ports and Attributes

Table 3-13 defines the TX gearbox ports.

Table 3-13: **TX Gearbox Ports**

Port	Dir	Clock Domain	Description
TXGEARBOXREADY	Out	TXUSRCLK2	This output indicates if data can be applied to the 64/66 or 64/67 gearbox when GEARBOX_ENDEC 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 64/66 gearbox, and [2:0] are used for the 64/67 gearbox.
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 64/66 gearbox, and [6:0] are used for the 64/67 gearbox.
TXSTARTSEQ	In	TXUSRCLK2	This input indicates the first word to be applied after reset for the 64/66 or 64/67 gearbox. The internal sequencer counter must be enabled by the GEARBOX_ENDEC attribute.

Table 3-14 defines the TX gearbox attributes.

Table 3-14: **TX Gearbox Attributes**

Attribute	Type	Description
GEARBOX_ENDEC	3-Bit Binary	This attribute indicates the TX gearbox modes: Bit 2: Always set to 0 to enable the gearbox decoder Bit 1: The encoding for this bit is: 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 Bit 0: The encoding for this bit is: 0: 64B/67B Gearbox mode for Interlaken 1: 64B/66B Gearbox
TXGEARBOX_USE	Boolean	When TRUE, this attribute enables the TX gearbox.

Enabling the TX Gearbox

To enable the TX gearbox for the GTX transceiver, the TXGEARBOX_USE attribute is set to TRUE.

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

TX Gearbox Bit and Byte Ordering

Figure 3-13 shows an example of the first five cycles of data entering and data exiting the TX gearbox for 64B/66B encoding when using a two-byte logic interface. The input consists of a 2-bit header and 16 bits of data. On the first cycle, the header and 14 bits of data exit the TX gearbox. On the second cycle, the remaining two data bits from the previous cycle's TXDATA input along with 14 data bits from the current TXDATA input exit the TX gearbox. This continues for the third and fourth cycle. On the fifth cycle, the output of the TX gearbox contains two remaining data bits from the first 66-bit block, the header of the second 66-bit block, and 12 data bits from the second 66-bit block. As shown in Figure 3-13, the header bits are serialized first followed by the data bits.

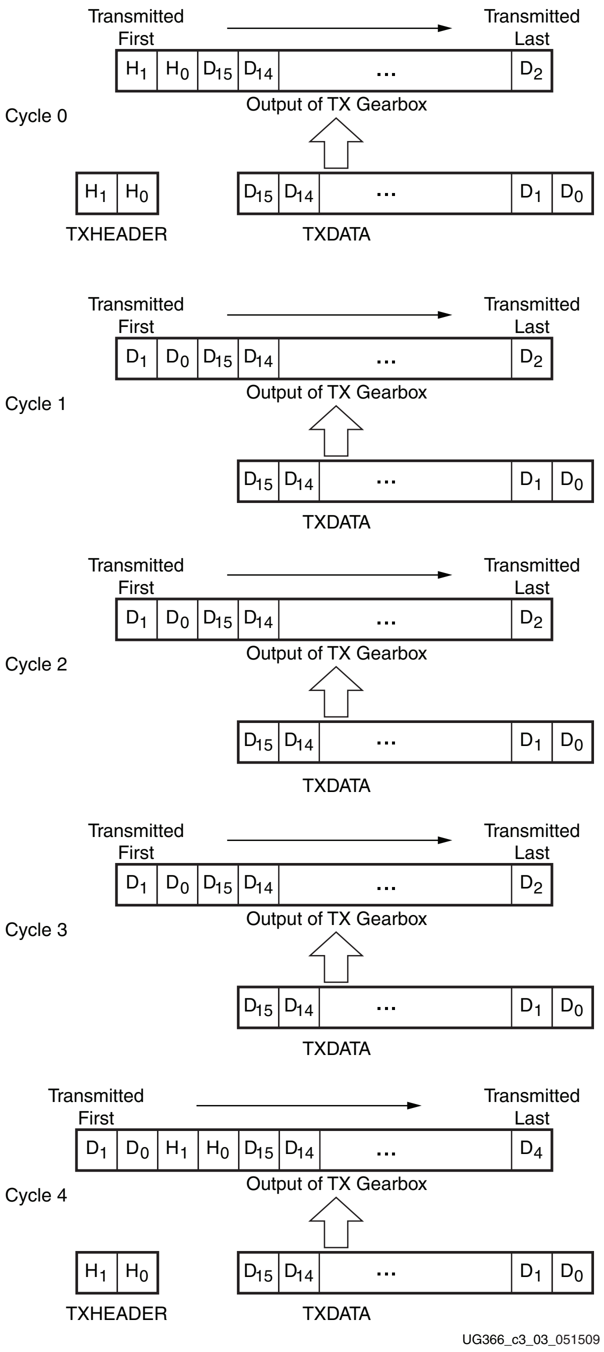


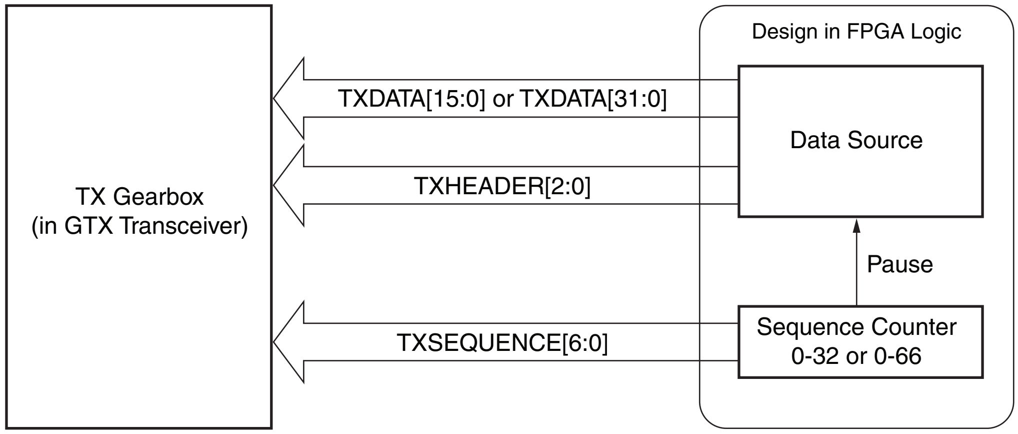
Figure 3-13: **TX Gearbox Bit Ordering**

TX Gearbox Operating Modes

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

External Sequence Counter Operating Mode

As shown in Figure 3-14, the external sequence counter operating mode uses the TXSEQUENCE[6:0], TXDATA[31:0], and TXHEADER[2:0] inputs. A binary counter must exist in the user logic to drive the TXSEQUENCE input port. For 64B/66B encoding, the counter increments from 0 to 32 and repeats from 0. For 64B/67B encoding, the counter increments from 0 to 66 and repeats from 0. When using 64B/66B encoding, tie TXSEQUENCE[6] to a logic 0 and tie the unused TXHEADER[2] to a logic 0. The sequence counter increment ranges ({0 to 32}, {0 to 66}) are identical for both the 2-byte and 4-byte interfaces. However, the counter must increment once every two TXUSRCLK2 cycles when using a 2-byte interface and every TXUSRCLK2 cycle when using a 4-byte interface.



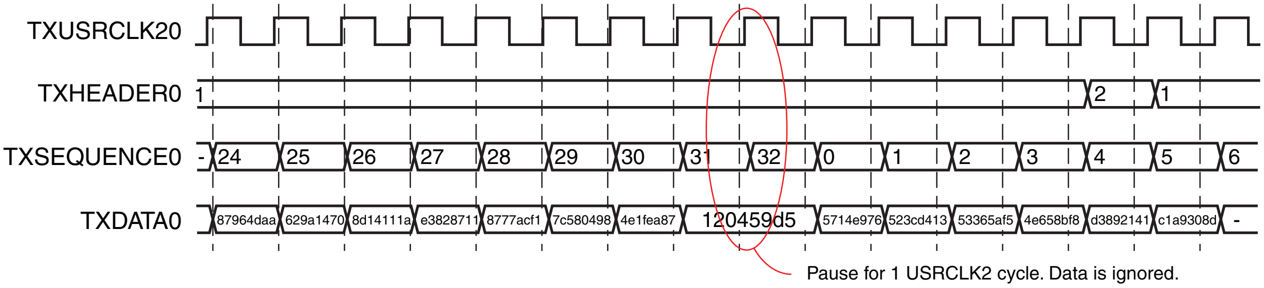
UG366_c3_04_051509

Figure 3-14: **TX Gearbox in External Sequence Counter Mode**

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

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

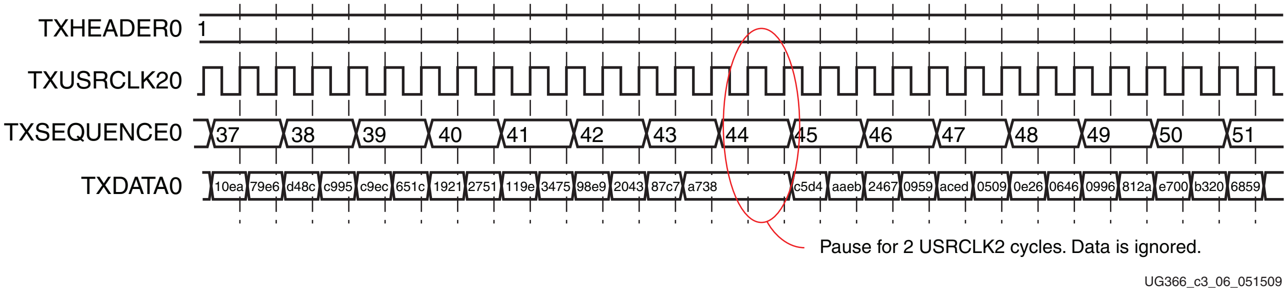
Figure 3-15 shows how a pause occurs at counter value 31 when using a 4-byte interface, external sequence counter mode, and 64B/66B encoding.



UG366_c3_05_051509

Figure 3-15: **Pause at Sequence Counter Value 31**

Figure 3-16 shows how a pause occurs at counter value 44 when using a 2-byte interface, external sequence counter mode, and 64B/67B encoding.



Pause for 2 USRCLK2 cycles. Data is ignored.

Figure 3-16: **Pause at Sequence Counter Value 44**

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

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

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

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

Internal Sequence Counter Operating Mode

As shown in Figure 3-17, the internal sequence counter operating mode uses the TXSTARTSEQ input and the TXGEARBOXREADY output in addition to the TXDATA data inputs and the TXHEADER header inputs. In this use model, the TXSEQUENCE inputs are not used. The use model is similar to the previous use model except that the TXGEARBOXREADY output is not used.

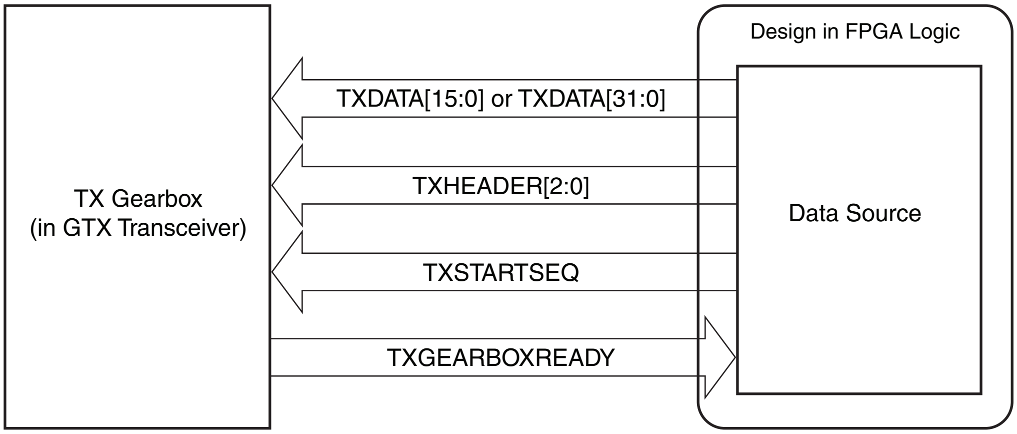


Figure 3-17: **TX Gearbox in Internal Sequence Counter Mode**

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

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

After data is driven, TXGEARBOXREADY is deasserted Low for either two TXUSRCLK2 cycles (4-byte mode) or three TXUSRCLK2 cycles (2-byte mode). Figure 3-18 and Figure 3-19 show the behavior of TXGEARBOXREADY for a 4-byte interface and a 2-byte interface, respectively. When TXGEARBOXREADY is deasserted Low, only one TXUSRCLK2 cycle remains before the data pipe must be stopped. The one-cycle latency is fixed and cannot be changed. After one cycle of latency, data must be held through four bytes (one TXUSRCLK2 cycle for 4-byte mode or two TXUSRCLK2 cycles for 2-byte mode) and then data is continued to be driven. Only data must be held. TXGEARBOXREADY transitions High on the cycle where new data must be driven. For this mode of operation, the number of hold points is identical to when using the external sequence counter mode for 64B/67B and 64B/66B.

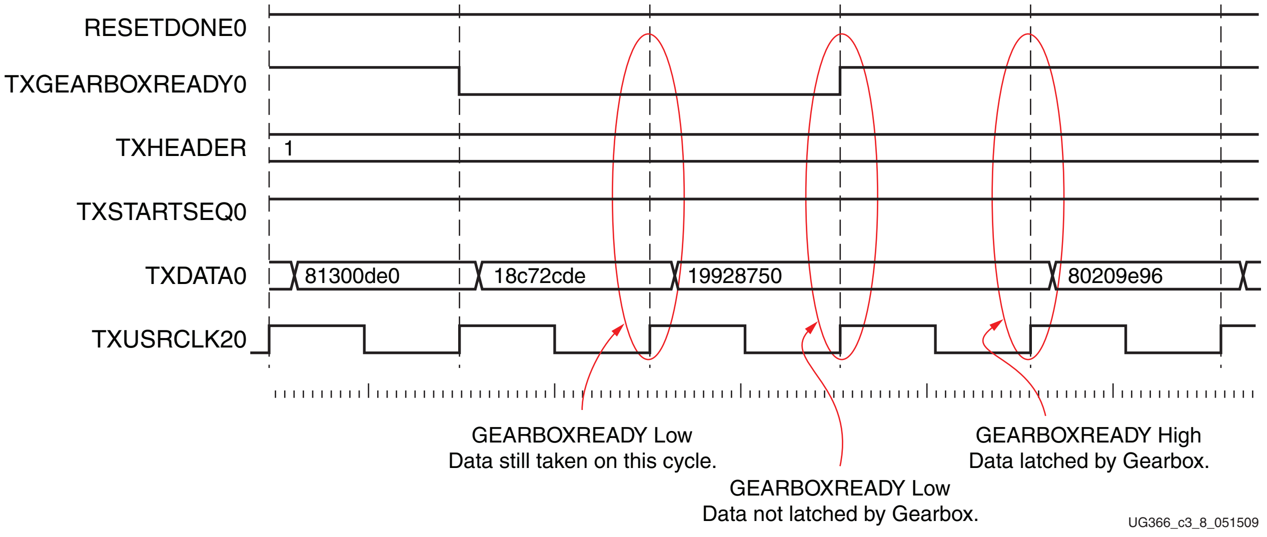


Figure 3-18: **TX Gearbox Internal Sequence Mode, 4-Byte Interface, 64B/67B**

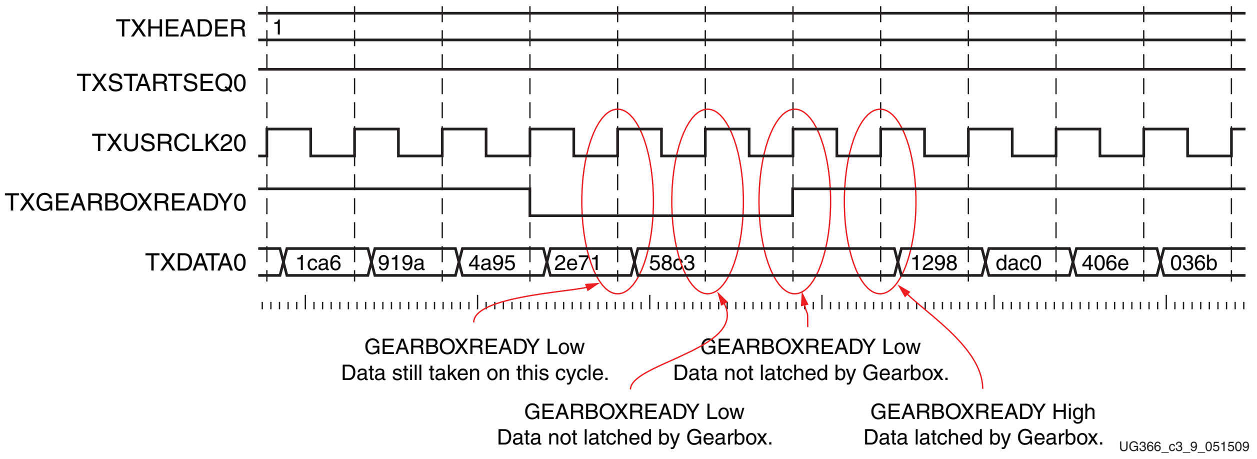


Figure 3-19: **TX Gearbox Internal Sequence Mode, 2-Byte Interface, 64B/66B**

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

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

TX Buffer

Functional Description

The GTX 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 3-20 shows the XCLK and TXUSRCLK domains.

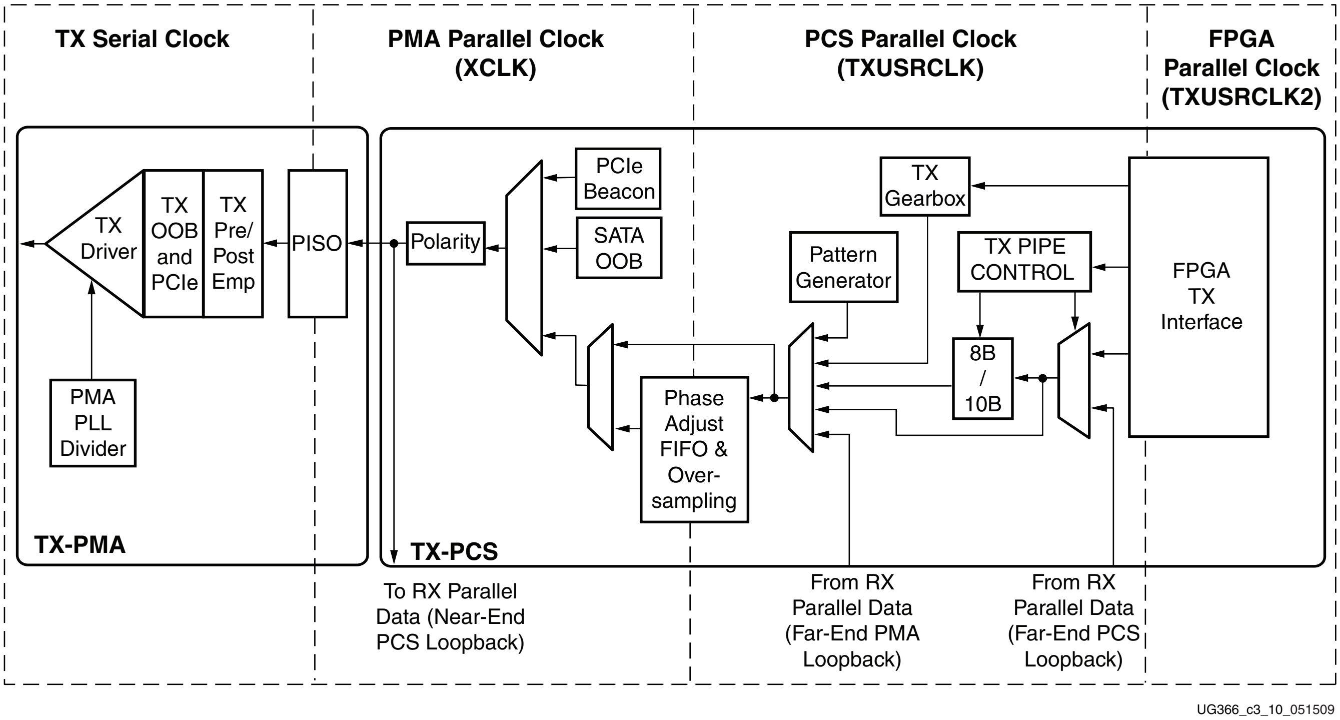

TX Serial Clock
PMA Parallel Clock (XCLK)
PCS Parallel Clock (TXUSRCLK)
FPGA Parallel Clock (TXUSRCLK2)
TX Driver
TX OOB and PCIe
TX Pre/ Post Emp
PISO
PMA PLL Divider
TX-PMA
Polarity
PCIe Beacon
SATA OOB
Phase Adjust FIFO & Over-sampling
TX Gearbox
Pattern Generator
TX PIPE CONTROL
8B / 10B
FPGA TX Interface
TX-PCS
To RX Parallel Data (Near-End PCS Loopback)
From RX Parallel Data (Far-End PMA Loopback)
From RX Parallel Data (Far-End PCS Loopback)
UG366_c3_10_051509

Figure 3-20: **TX Clock Domains**

The GTX transceiver transmitter includes a TX buffer and a TX phase-alignment circuit to resolve phase differences between the PMACLK and TXUSRCLK domains. All TX datapaths must use these circuits. Table 3-15 shows trade-offs between buffering and phase alignment.

Table 3-15: **Buffering vs. Phase Alignment**

	TX Buffer	TX Phase Alignment
Ease of Use	The TX buffer is used when possible. It is robust and easy to operate.	Phase alignment requires extra logic and additional constraints on clock sources. TXOUTCLK cannot be used unless the TXOUTCLK_CTRL attribute is set to “TXPLLREFCLK_DIV1” or “TXPLLREFCLK_DIV2”.
Latency	If low latency is critical, the TX buffer must be bypassed.	Phase alignment uses fewer registers in the datapath.

Table 3-15: **Buffering vs. Phase Alignment *(Cont'd)***

	TX Buffer	TX Phase Alignment
TX Lane-to-Lane Deskew		The phase-alignment circuit can be used to reduce the skew between separate GTX transceivers. All GTX transceivers involved must use the same line rate.
Oversampling	The TX buffer is required for oversampling.	

Ports and Attributes

Table 3-16 defines the TX buffer ports.

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

Port	Dir	Clock Domain	Description
TXBUFSTATUS[1:0]	Out	TXUSRCLK2	TX buffer status. TXBUFSTATUS[1]: TX buffer overflow or underflow 1: FIFO has overflowed or underflowed 0: No overflow/underflow error TXBUFSTATUS[0]: TX buffer fullness 1: FIFO is at least half full 0: FIFO is less than half full When TXBUFSTATUS[1] goes High, it remains High until TXRESET is asserted.
TXRESET	In	Async	PCS TX system reset. This input resets the TX FIFO, 8B/10B encoder, and other transmitter registers. This reset is a subset of GTXTXRESET.

Table 3-16 defines the TX buffer attributes.

Table 3-17: **TX Buffer Attributes**

Attribute	Type	Description
TX_BUFFER_USE	Boolean	Use or bypass the TX buffer. TRUE: Use the TX buffer. FALSE: Bypass the TX buffer (advanced feature).
TX_OVERSAMPLE_MODE	Boolean	Enables/disables the built-in 5X digital oversampling. TRUE: Built-in 5X digital oversampling is enabled. This option is used for slow line rates. FALSE: Built-in 5X digital oversampling is disabled. This option is used for TX normal mode.

Using the TX Buffer

To use the TX buffer to resolve phase differences between the domains, TX_BUFFER_USE must be set to TRUE. The buffer should be reset whenever TXBUFSTATUS indicates an overflow or an underflow. The buffer can be reset using GTXTXRESET or TXRESET. Assertion of GTXTXRESET triggers a sequence that resets the entire TX of the GTX transceiver.

Using the TX Buffer for Oversampling Mode

When oversampling is enabled (OVERSAMPLE_MODE = TRUE), the TX buffer is used for bit interpolation and must always be active. See TX Oversampling, page 167 for more information about built-in 5X oversampling.

TX Buffer Bypass

Functional Description

Bypassing the TX buffer is an advanced feature of the Virtex-6 FPGA GTX transceivers. 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. Figure 3-20, page 154 shows the XCLK and USRCLK domains. Table 3-15, page 154 shows the trade-offs between the buffer and the buffer bypass modes.

The system margin in TX Buffer Bypass mode depends on the TXUSRCLK frequency and the model. To enhance the system margin for better compensation to temperature and/or voltage variation, additional requirements on the clocking use model must be met. These requirements are described in the section Transmit Fabric Clocking Use Model for TX Buffer Bypass, page 162.

To ensure that the TXOUTCLK output port operates at the desired frequency in TX buffer bypass mode, all of the following conditions must be met:

- When the TX PLL supplies the clock for the TX datapath (TX_CLK_SOURCE = “TXPLL”):
  - The transmitter reference clock must always be toggling
  - TXPLLPOWERDOWN must be tied Low
- When the RX PLL supplies the clock for the TX datapath (TX_CLK_SOURCE = “RXPLL”):
  - The receiver reference clock must always be toggling
  - RXPLLPOWERDOWN must be tied Low
- GTXTXRESET, GTXRXRESET, PLLTXRESET, and PLLRXRESET must not be tied High.

For transceivers that are not instantiated in the user design, the ISE® software, version 12.1 or later, automatically ensures that the TXOUTCLK performance is preserved for future use. MGTAVCC must be supplied to these transceivers. Refer to Managing Unused GTX Transceivers, page 276 for more information.

Ports and Attributes

Table 3-18 defines the TX buffer bypass ports.

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

Port	Dir	Clock Domain	Description
TXDLYALIGNDISABLE	In	Async	Reserved. Must be tied High.
TXDLYALIGNMONENB	In	Async	Reserved. Must be tied High.
TXDLYALIGNMONITOR[7:0]	Out	Async	Reserved.

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

Port	Dir	Clock Domain	Description
TXDLYALIGNOVERRIDE	In	Async	Reserved. Must be tied Low.
TXDLYALIGNRESET	In	Async	Reserved. Must be tied High.
TXDLYALIGNUPDSW	In	Async	Reserved. Use recommended values from the Virtex-6 FPGA GTX Transceiver Wizard.
TXENPMAPHASEALIGN	In	Async	When activated, the GTX transceiver transmitter can align its XCLK with its TXUSRCLK. This also allows the XCLKs in multiple GTX transceiver transmitters to be synchronized to reduce TX skew between them.
TXOUTCLK	Out	N/A	This output is the recommended clock to the FPGA logic. The TXOUTCLK_CTRL attribute is the input selector for TXOUTCLK. When TXOUTCLK is used as the clock source for TXUSRCLK in TX buffer bypass mode, TXOUTCLK_CTRL must select either TXPLLREFCLK_DIV1 or TXPLLREFCLK_DIV2 for TX phase alignment to be effective.
TXPLLLKDET	Out	Async	Indicates that the VCO rate is within acceptable tolerances of the desired rate when High. The GTX transceiver does not operate reliably until this condition is met.
TXPLLLKDETEN	In	Async	Enables the TX PLL lock detector. It must be tied High.
TXPMASETPHASE	In	Async	When activated, TXPMASETPHASE aligns XCLK with TXUSRCLK allowing the TX buffer to be bypassed. While TXPMASETPHASE is High, the phase of XCLK generally changes to affect its alignment with TXUSRCLK. This in turn implies a change in phase on the TX serial lines, which appears as temporary, occasional changes in the nominal UI. Thus, the UI might be observed to vary by several ps when TXPMASETPHASE is High.
TXUSRCLK	In	N/A	Use this port to provide a clock for the internal TX PCS datapath. The rate for TXUSRCLK depends on TX_DATA_WIDTH.

Table 3-19 defines the TX buffer bypass attributes.

Table 3-19: **TX Buffer Bypass Attributes**

Attribute	Type	Description
POWER_SAVE	10-bit Binary	POWER_SAVE[4]: Mux select for the TXOUTCLK output clock. Must be tied to 1’b1. 1'b0: Use the TX Delay Aligner 1'b1: Bypass the TX Delay Aligner POWER_SAVE[5]: Mux select for the RXRECCLK output clock. Must be tied to 1’b1 when RX buffer is used (RX_BUFFER_USE = TRUE). When RX buffer is bypassed, refer to Using the RX Phase Alignment Circuit to Bypass the Buffer, page 234. 1'b0: Use the RX Delay Aligner 1'b1: Bypass the RX Delay Aligner All other bits are reserved. Use recommended values from the Virtex-6 FPGA GTX Transceiver Wizard.
TX_BUFFER_USE	Boolean	Use or bypass the TX buffer. TRUE: Use the TX buffer (normal mode). FALSE: Bypass the TX buffer (advanced feature).
TX_BYTECLK_CFG[5:0]	6-bit Hex	Reserved. Use only recommended values from the Virtex-6 FPGA GTX Transceiver Wizard.
TX_DATA_WIDTH	Integer	Sets the transmitter external data width 8/10: 1-byte interface 16/20: 2-byte interface 32/40: 4-byte interface If 8B10B is used, this attribute must be a multiple of 10.
TX_DLYALIGN_CTRINC	4-bit Binary	Reserved. Use only recommended values from the Virtex-6 FPGA GTX Transceiver Wizard.
TX_DLYALIGN_LPFINC	4-bit Binary	Reserved. Use only recommended values from the Virtex-6 FPGA GTX Transceiver Wizard.
TX_DLYALIGN_MONSEL	3-bit Binary	Reserved. Use only recommended values from the Virtex-6 FPGA GTX Transceiver Wizard.
TX_DLYALIGN_OVRDSETTING	8-bit Binary	Sets the overdrive value for the TX delay aligner. This attribute takes effect when TXDLYALIGNOVERRIDE is driven High.
TX_PMADATA_OPT	1-bit Binary	This attribute controls the inverter that optimizes the clock path within the GTX transceiver for different mode of operations. The attribute must be set as follows: 0: Use when TX_BUFFER_USE = TRUE 1: Use when TX_BUFFER_USE = FALSE

Table 3-19: **TX Buffer Bypass Attributes *(Cont'd)***

Attribute	Type	Description
TX_XCLK_SEL	String	Selects the clock used to drive the clock domain in the PCS following the TX buffer. When using the TX buffer, this attribute is set to TXOUT. The attribute must be set as follows: TXOUT: Use when TX_BUFFER_USE = TRUE TXUSR: Use when TX_BUFFER_USE = FALSE
TXOUTCLK_CTRL	String	This attribute is the multiplexer select signal from the TX Fabric Clock Output Control block (see Figure 3-28). TXOUTCLKPCS (DRP value `000`) TXOUTCLKPMA_DIV1 (DRP value `001`) TXOUTCLKPMA_DIV2 (DRP value `010`) TXPLLREFCLK_DIV1 (DRP value `011`) TXPLLREFCLK_DIV2 (DRP value `100`) OFF_LOW (DRP value `101`) OFF_HIGH (DRP value `110`)

Using the TX Phase-Alignment Circuit to Bypass the Buffer

To use the TX phase-alignment circuit, follow these steps:

1. Set the following attributes with their values as follows:
   a. Set TXOUTCLK_CTRL to use either TXPLLREFCLK_DIV2 or TXPLLREFCLK_DIV1
   b. Set TX_XCLK_SEL to TXUSR
   c. Set TX_BUFFER_USE to FALSE
   d. Set TX_PMADATA_OPT to `1'b1`
2. After power-on, make sure TXPMASETPHASE and TXENPMAPHASEALIGN are driven Low.
3. Apply GTXTXRESET and wait for TXRESETDONE to go High.
4. Drive TXENPMAPHASEALIGN High.

   Keep TXENPMAPHASEALIGN High unless the phase-alignment procedure must be repeated. Driving TXENPMAPHASEALIGN Low causes phase alignment to be lost.
5. Wait 32 TXUSRCLK2 clock cycles and then drive TXPMASETPHASE High.
6. Wait the number of required TXUSRCLK2 clock cycles as specified in Table 3-20, and then drive TXPMASETPHASE Low. The phase of the PMACLK is now aligned with TXUSRCLK.

Table 3-20: **Number of Required TXUSRCLK2 Clock Cycles for Driving TXPMASETPHASE High**

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

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

- GTXTXRESET is asserted
- TXPLLPOWERDOWN is deasserted
- The clocking source changed
- The line rate of the GTX transceiver TX changed

Figure 3-21 shows the TX phase-alignment procedure. See Table 3-20 for the required TXUSRCLK2 clock cycles in asserting TXPMASETPHASE High.

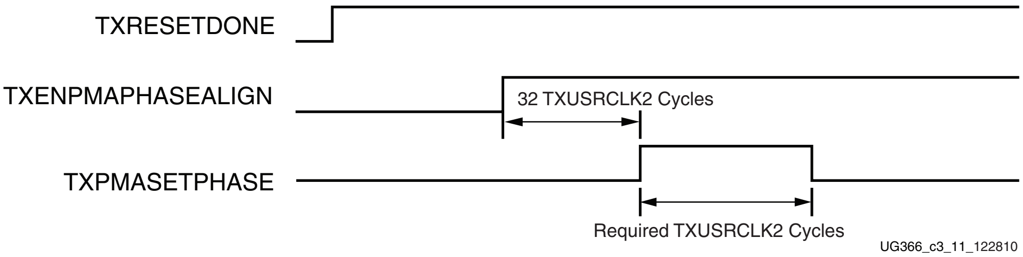


Figure 3-21: **TX Phase Alignment Procedure after GTXTXRESET**

TX Phase Alignment after Rate Change Use Mode

After a line rate change and if the TX buffer is bypassed, it is necessary to perform TX phase alignment and reset the TX PLL output clock divider. The main difference between the TX phase-alignment procedure after a reset and after a rate change is holding TXENPMAPHASEALIGN asserted after the rate change.

The steps for an example TX phase-alignment after a rate change are:

1. In non PCI Express mode, a TX rate change completion occurs when a TXRATEDONE pulse is detected.

   In PCI Express mode, a TX rate change completion occurs when a PHYSTATUS pulse is detected following a TXRATEDONE pulse.

   After PHYSTATUS pulse is detected, assert GTXTEST[1] for 16 TXUSRCLK2 cycles to reset the TX PLL output clock divider.
2. Hold TXENPMAPHASEALIGN asserted.
3. Wait for at least 32 TXUSRCLK2 cycles.
4. Assert TXPMASETPHASE for the required TXUSRCLK2 cycles specified in Table 3-20.
5. Assert the TXRESET pulse and wait for TXRESETDONE to be asserted.
6. For PCI Express mode, the “USER_PHYSTATUS” user signal must be generated and used as PIPE PHYSTATUS. Under normal operation, USER_PHYSTATUS follows the GTX transceiver PHYSTATUS signal. During rate change and subsequent TX phase alignment, USER_PHYSTATUS must gate the GTX transceiver PHYSTATUS and delay its assertion until after the TX phase alignment sequence is completed.

   For non PCI Express mode, a “SYNC_DONE” user signal, which behaves like USER_PHYSTATUS, can be asserted to indicate TX phase-alignment completion.

Figure 3-22 shows the TX phase alignment after a rate change.

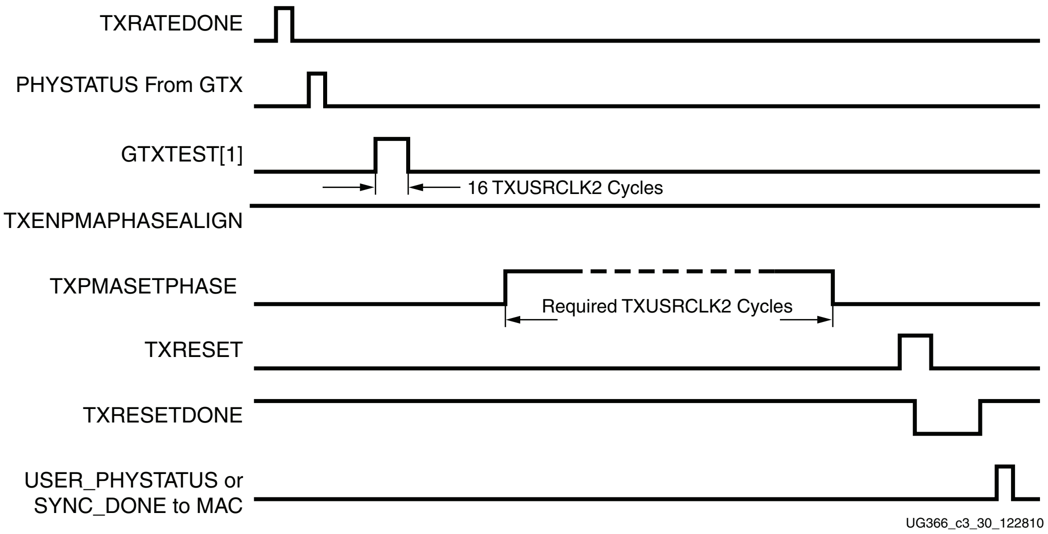


Figure 3-22: **TX Phase Alignment After Rate Change**

Note: PHYSTATUS and USER_PHYSTATUS are used in PCI Express mode. USER_PHYSTATUS is a gated version of PHYSTATUS from the GTX transceiver TX. It is recommended that PHYSTATUS indicating a rate change completion is gated to the Media Access Layer (MAC) during TX phase alignment. TXENPMAPHASEALIGN must remain asserted after a rate change.

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

The TX phase-alignment circuit can also be used to minimize skew between GTX transceivers. Figure 3-23 shows how the phase-alignment circuit can reduce lane skew by aligning the PMACLK domains of multiple GTX transceivers to a common clock.

Figure 3-23 shows multiple lanes running before and after phase alignment to a common clock. Before phase alignment, all PMACLKs have an arbitrary phase difference, but after alignment, the only phase difference is the skew for the common clock, and all data is transmitted simultaneously as long as the datapath latency is matched.

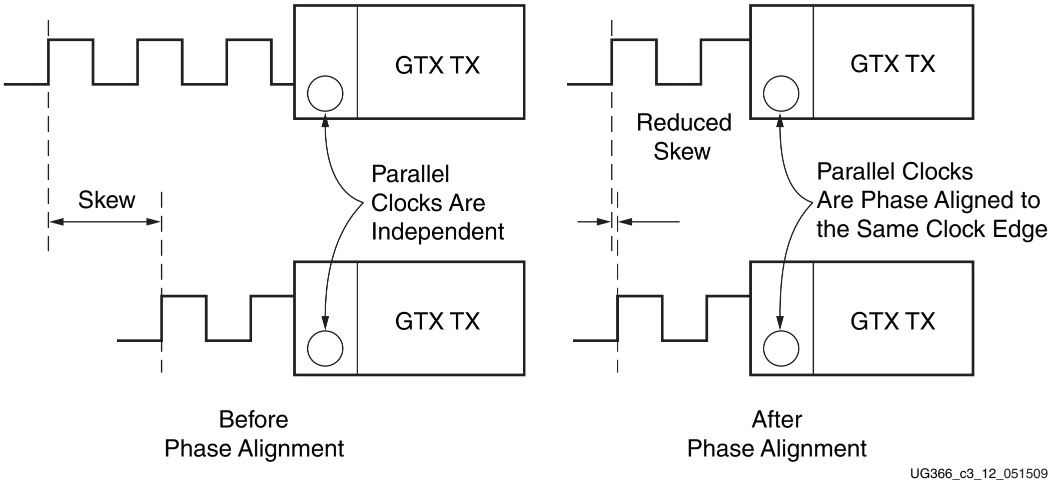


Figure 3-23: **TX Phase Alignment Circuit to Reduce Lane Skew**

For phase alignment to be effective, TXUSRCLK and TXUSRCLK2 for all GTX transceivers must come from the same source and must be routed through a low-skew clocking resource (such as BUFG,BUFR or MMCM). When TXOUTCLK is used as the clock source in TX buffer bypass mode, TXOUTCLK_CTRL must select either TXPLLREFCLK_DIV1 or TXPLLREFCLK_DIV2. Refer to Figure 3-5, page 134, Figure 3-6, page 135, and Figure 3-7, page 136 for additional examples on how TXUSRCLK and TXUSRCLK2 signals should be driven for low-skew operation.

Transmit Fabric Clocking Use Model for TX Buffer Bypass

The system margin in TX Buffer Bypass mode depends on the following:

- TXUSRCLK frequency (See Equation 3-1)
- Clocking resources used to generate TXUSRCLK from TXOUTCLK (BUFG, BUFR or MMCM)

Table 3-21 describes the required clocking use model when bypassing the TX Buffer as a function of the line rate and the internal Data Width. To enhance the system margin for better compensation to Temperature and/or Voltage drift, these requirements must be met.

Table 3-21: **Required Clocking Use Model When bypassing the TX Buffer as a Function of the Line Rate and the internal Data Width**

Line Rate (Gb/s)	Internal Data Width	Device	Clocking Use Model
≤ 4.2	20-bit	≤ 240T	BUFG, BUFR, MMCM[1]
≤ 5.8	20-bit	≤ 240T	BUFR, MMCM[1]
≤ 6.2	20-bit	≤ 240T	MMCM [1]
≤ 6.6	20-bit	≤ 240T	MMCM[1]
≤ 4.2	16-bit	≤ 240T	BUFG, BUFR, MMCM[1]
≤ 4.6	16-bit	≤ 240T	BUFR, MMCM[1]
≤ 4.9	16-bit	≤ 240T	MMCM[1]
≤ 6.6	16-bit	≤ 240T	MMCM[1]

1. MMCM must be placed in the same clock region as the driving GTX transceiver. For details about placement constraints and restrictions on clocking resources (BUFG, BUFR, MMCM, etc.), refer to the *Virtex-6 FPGA Clocking Resources User Guide*.

FPGA TX Interface describes these clocking use models in detail. Figures Figure 3-2 and Figure 3-5 describe the clocking use model with BUFG and BUFR. Figures Figure 3-3 and Figure 3-6 describe the clocking use model with MMCM.

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 transceiver pattern generator block can generate several industry-standard PRBS patterns listed in Table 3-22.

Table 3-22: **Supported PRBS Pattern**

Name	Polynomial	Length of Sequence	Descriptions
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 as 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 scheme. 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 scheme. A recommended PRBS test pattern for 10 Gigabit Ethernet. See IEEE 802.3ae-2002.

In addition to PRBS patterns, the GTX transceiver supports 20 UI (or 16 UI) and 2 UI square wave test patterns and PCI Express compliant pattern generation. Clocking pattern is usually used to check PLL random jitter often done with a spectrum analyzer.

Table 3-23: **PCI Express Compliance Pattern**

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

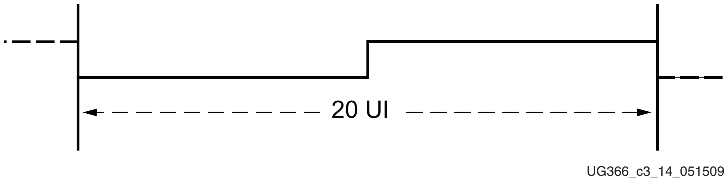


Figure 3-24: **20 UI Square Wave**

Error insertion function is supported to verify link connection and also for jitter tolerance test. When inverted PRBS pattern is necessary, use TXPOLARITY signal to control polarity.

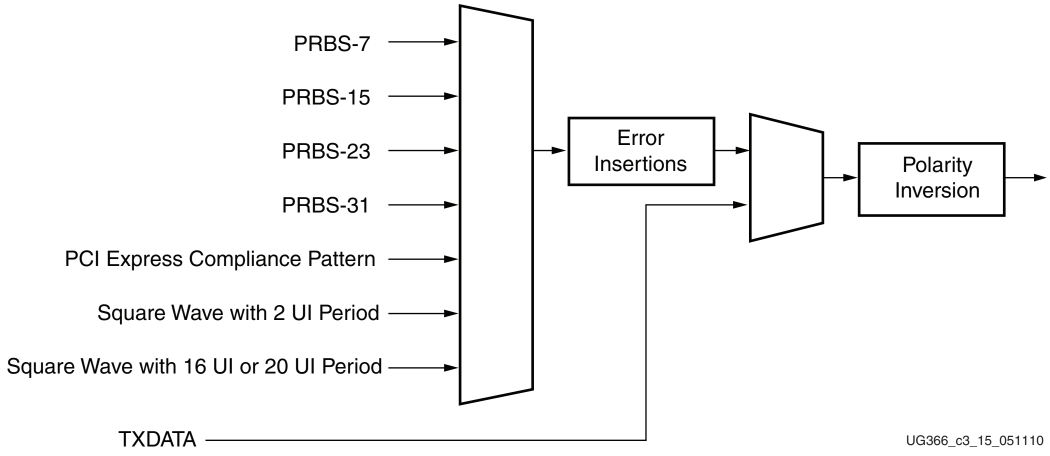

PRBS-7
PRBS-15
PRBS-23
PRBS-31
PCI Express Compliance Pattern
Square Wave with 2 UI Period
Square Wave with 16 UI or 20 UI Period
TXDATA
Error Insertions
Polarity Inversion
UG366_c3_15_051110

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

Ports and Attributes

Table 3-24 defines the pattern generator ports.

Table 3-24: **Pattern Generator Ports**

Port	Dir	Clock Domain	Description
TXENPRBSTST[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 mode `110`: Square wave with 2 UI (alternating 0's/1's) `111`: Square wave with 16 UI or 20 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 TXENPRBSTST is set to `000`, this does not affect TXDATA.

Table 3-25 defines the pattern generator attribute.

Table 3-25: **Pattern Generator Attribute**

Attribute	Type	Description
RXPRBSERR_LOOPBACK	1-bit Binary	When set to 1, causes RXPRBSERR bit to be internally looped back to TXPRBSFORCEERR of the same GTX 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 test and also for jitter tolerance test. For link quality testing, choose test pattern by setting TXENPRBSTST and RXENPRBSTST to non-`000`, and set RXPRBSERR_LOOPBACK to `0` (Figure 3-26). Only the PRBS pattern is recognized by the RX pattern checker.

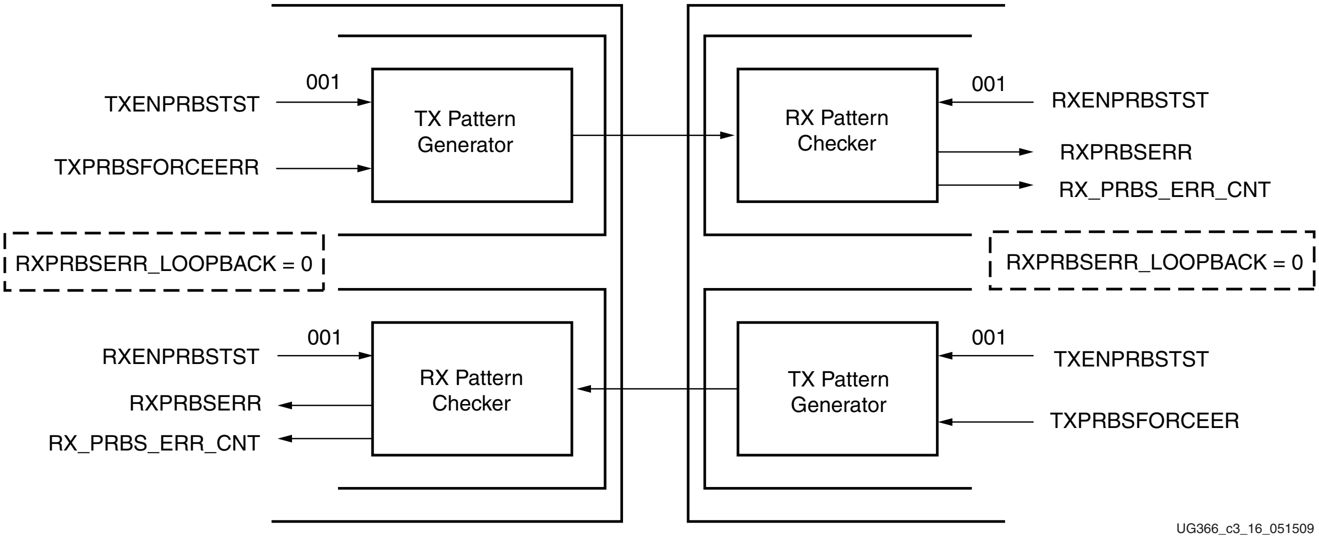


Figure 3-26: **Link Test Mode with a PRBS-7 Pattern**

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

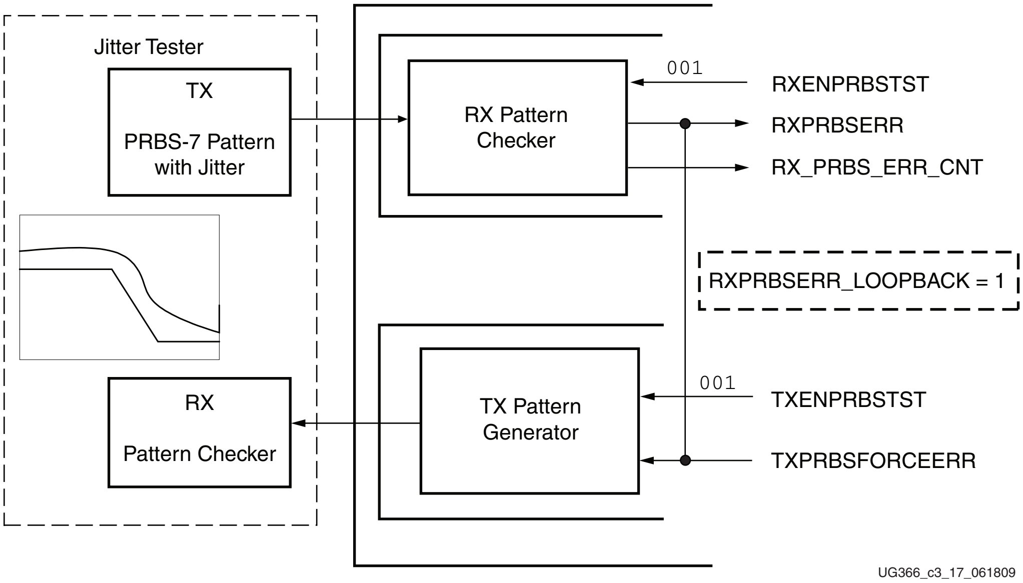


Figure 3-27: **Jitter Tolerance Test Mode with a PRBS-7 Pattern**

TX Oversampling

Functional Description

Each GTX transceiver includes built-in 5X oversampling to enable serial rates from 1/10th of the lower border of the frequency range of the TX PMA PLL up to 4/10th of the TX PMA PLL. The digital oversampling circuit takes parallel data from the user interface at five times the desired line rate and replicates the data bit five times before advancing to the next bit. The internal data width is automatically set to 20 bits when TX oversampling is enabled.

Ports and Attributes

There are no TX oversampling ports.

Table 3-26 defines the TX oversampling attributes.

Table 3-26: **TX Oversampling Attributes**

Attribute	Type	Description
TX_OVERSAMPLE_MODE	Boolean	This enables transmitter oversampling when TRUE, and 5X oversampling is On.
TXPLL_DIVSEL_OUT	Integer	TXPLL_DIVSEL_OUT must be set to 1 when using oversampling mode.

TX Polarity Control

Functional Description

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

Ports and Attributes

Table 3-27 defines the TX polarity control ports.

Table 3-27: **TX Polarity Control Ports**

Port	Dir	Clock Domain	Description
TXPOLARITY	In	TXUSRCLK2	The TX polarity 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.

There are no TX polarity control attributes.

Using TX Polarity Control

If the TXP/TXN differential traces are swapped on a board, tie TXPOLARITY High.

TX Fabric Clock Output Control

Functional Description

The TX Fabric Clock Output 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.

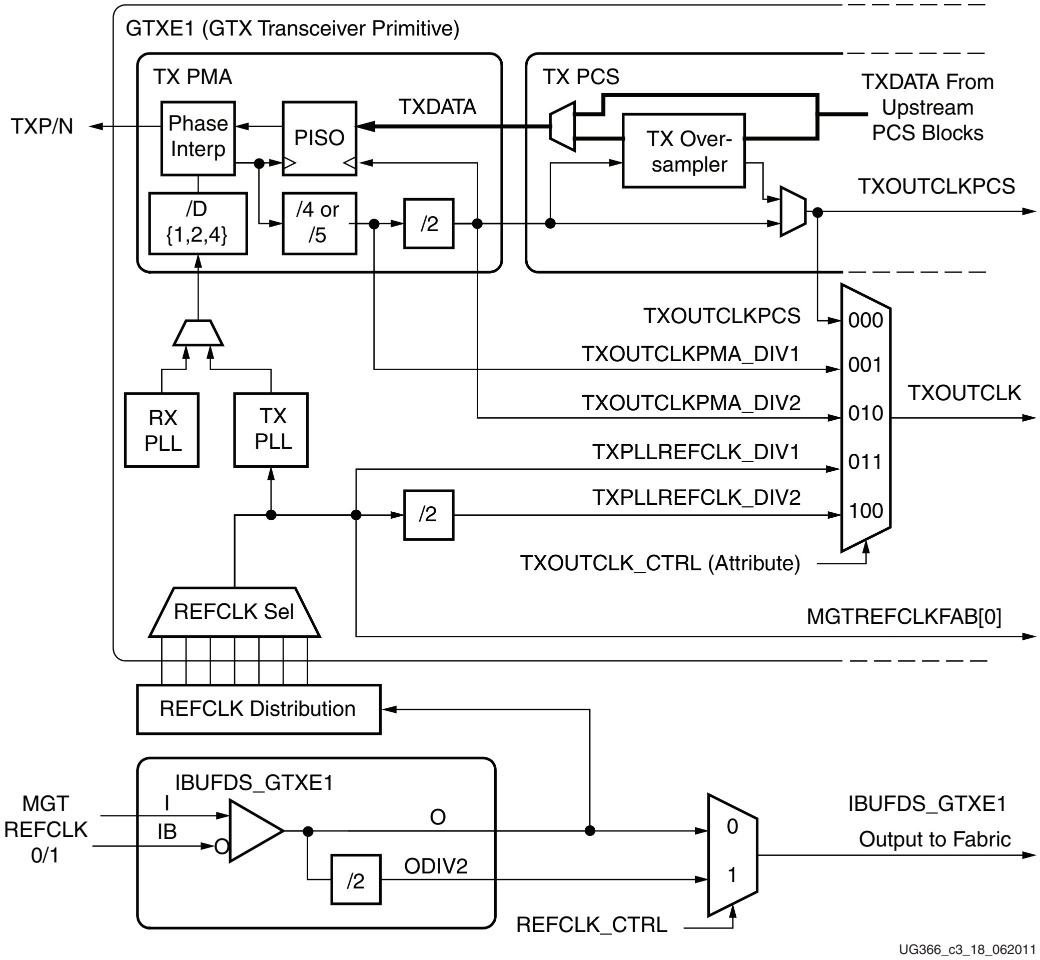


Figure 3-28: **TX Serial and Parallel Clock Divider Detail**

Notes relevant to Figure 3-28:

1. TXOUTCLKPCS and MGTREFCLKFAB[0] 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 IBUFDS_GTXE1's O or ODIV2 outputs to the fabric.
3. IBUFDS_GTXE1 is a redundant output for additional clocking scheme flexibility.
4. The RX PLL resides in the RX portion of the same GTX transceiver. It can be used in place of the TX PLL for low-power operation.

5. The selection of the /4 or /5 divider block is dependent on TX_DATA_WIDTH (see Table 3-1, page 128):
    - /4 is selected when the internal data width is 16
    - /5 is selected when the internal data width is 20

For details about placement constraints and restrictions on clocking resources (MMCM, BUFGCTRL, IBUFDS_GTXE1, BUFG, etc.), refer to the *Virtex-6 FPGA 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 divider 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 TXPLL_DIVSEL_OUT attribute must be set to the appropriate value, and the TXRATE port needs to be tied to 00.

To use the D divider in multiple line rate applications, the TXRATE port is used to dynamically select the D divider value. The TXPLL_DIVSEL_OUT 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.

The control for the serial divider is described in Table 3-28. For details about the line rate range per speed grade, refer to the *Virtex-6 FPGA Data Sheet*.

Table 3-28: **TX PLL Output Divider Setting**

Line Rate Range (Gb/s)	D Divider Value	Static Setting via Attribute	Dynamic Control via Ports

2.40 to 6.60	1	TXPLL_DIVSEL_OUT = 1
		TXRATE = 00
			TXRATE = 11

1.20 to 3.3	2	TXPLL_DIVSEL_OUT = 2
		TXRATE = 00
			TXRATE = 10

0.60 to 1.65	4	TXPLL_DIVSEL_OUT = 4
		TXRATE = 00
			TXRATE = 01

Parallel Clock Divider and Selector

The parallel clock outputs from the TX Fabric Clock Output Control block can be used as a fabric logic clock. The parallel clock divider block can output a 1-byte or 2-byte data width clock.

The recommended clock for the fabric is the TXOUTCLK from one of the GTX transceivers. It is also possible to bring the MGTREFCLK directly to the fabric 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 156 for more details.

The TXOUTCLK_CTRL attribute controls the input selector and allows the following clocks to be output via TXOUTCLK port:

- TXOUTCLKPCS: This clock should only be used when the TX Oversampling block is enabled. The TX Oversampling block divides down the TXOUTCLKPMA_DIV2 clock to match the 5X oversampled data rate.

- TXOUTCLKPMA_DIV1/DIV2: This is the divided down PLL clock after the TX phase interpolator and is used by the TX PCS block. The TX phase interpolator is used to match the phase of the internal clock to the FPGA logic clock in TX buffer bypass mode.
- TXPLLREFCLK_DIV1/DIV2: This is the input reference clock to the TX PLL. TXPLLREFCLK is the recommended clock for general usage and is required for the TX buffer bypass mode.

The TXOUTCLKPMA_DIV2 output is the 2-byte datapath frequency and is used when TXDATA is 2 bytes. The TXOUTCLKPMA_DIV1 output is the 1-byte datapath frequency and is used when TXDATA is 1 byte.

Ports and Attributes

Table 3-29 defines the TX Fabric Clock Output Control block ports.

Table 3-29: **TX Fabric Clock Output Control Ports**

Port	Dir	Clock Domain	Description
GTXTEST[1]	In	Async	This port resets the TX PLL output divider (controlled via the TXPLL_DIVSEL_OUT attribute or the TXRATE port). The TX PLL output clock divider must be reset whenever the line rate is changed by modifying the output dividers at run time. To perform this reset, GTXTEST[1] is asserted for 16 TXUSRCLK2 cycles after the line rate change is completed.
MGTREFCLKFAB[0]	Out	Async	MGTREFCLKFAB[0] is a redundant output. TXOUTCLK with TXOUTCLK_CTRL = “TXPLLREFCLK_DIV1” should be used instead.
O ODIV2	Out	Async	The IBUFDS_GTXE1 primitive allows MGTREFCLK to be output to the FPGA logic directly.
PHYSTATUS	Out	RXUSRCLK2 /Async	After TXRATE is changed to initiate a rate change, PHYSTATUS is output Low and toggles for one TXUSRCLK2 cycle at the conclusion of the rate change as defined by TRANS_TIME_RATE. PHYSTATUS is intended for PCI Express protocol operation. Non PCI Express protocol operation should monitor TXRATEDONE. Refer to Table 3-33, page 180 for more information on PHYSTATUS.
TXOUTCLK	Out	Async	TXOUTCLK is the recommended clock output to the FPGA logic. The TXOUTCLK_CTRL attribute is the input selector for TXOUTCLK and allows the TX PLL input reference clock or other internal clocks to be output to the FPGA logic. It can only be connected directly to the clock pin of the MMCM in the same clocking region or the input of the BUFR / BUFG.
TXOUTCLKPCS	Out	Async	TXOUTCLKPCS is a redundant output. TXOUTCLK with TXOUTCLK_CTRL = “TXOUTCLKPCS” should be used instead.

Table 3-29: **TX Fabric Clock Output Control Ports *(Cont'd)***

Port	Dir	Clock Domain	Description
TXRATE	In	TXUSRCLK2	This port controls the setting for the TX serial clock divider for low line rate support (see Table 3-28). This input port is used in combination with the TXPLL_DIVSEL_OUT attribute. 00: Let TXPLL_DIVSEL_OUT determine the D divider value 01: Set D divider to 4 10: Set D divider to 2 11: Set D divider to 1
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.

Table 3-30 defines the TX Fabric Clock Output Control block attributes.

Table 3-30: **TX Fabric Clock Output Control Attributes**

Attribute	Type	Description
TRANS_TIME_RATE	8-bit Hex	Reserved. Use only recommended values from the Virtex-6 FPGA GTX Transceiver Wizard. This attribute determines when PHYSTATUS and TXRATEDONE are asserted after a rate change.
TX_EN_RATE_RESET_BUF	Boolean	When set to TRUE, this attribute enables automatic TX buffer reset during a rate change event initiated by a change in TXRATE.
TXOUTCLK_CTRL	String	This attribute is the multiplexer select signal in Figure 3-28, page 168. It determines which GTX transceiver internal clock is output to the FPGA logic via the TXOUTCLK port. Valid settings are: "TXOUTCLKPCS" "TXOUTCLKPMA_DIV1" "TXOUTCLKPMA_DIV2" "TXPLLREFCLK_DIV1" "TXPLLREFCLK_DIV2"
TXPLL_DIVSEL_OUT	Integer	This attribute controls the setting for the TX serial clock divider for low line rate support (see Table 3-28, page 169). This attribute is only valid when TXRATE = 00. Otherwise the D divider value is controlled by TXRATE. Valid settings are: 1: Set the D divider to 1 2: Set the D divider to 2 4: Set the D divider to 4

PCI Express Clocking Use Mode

In most applications, TXOUTCLK port is used to clock the FPGA logic. TXOUTCLKPCS via TXOUTCLK is used for the oversampling mode, and TXPLLREFCLK via TXOUTCLK is used for the rest of the applications.

For PCI Express clocking, the use case is to route the reference clock from IBUFDS_GTXE1 directly to the FPGA user logic instead of through TXOUTCLK if the TX buffer is used, as shown in Figure 3-29. However, if TX buffer is bypassed and compensation for voltage and/or temperature variations from the global clock trees is necessary, TXOUTCLK must be used as the user clock source. See TX Buffer Bypass, page 156 for more details. In Figure 3-29, TXRATE_SEL is an FPGA user logic control signal that selects a new PCLK frequency from the BUFGMUX (125 MHz or 250 MHz) for a rate change. PCLK is the PIPE clock for the FPGA user logic and is the parallel interface clock used to synchronize data transfers across the parallel interface.

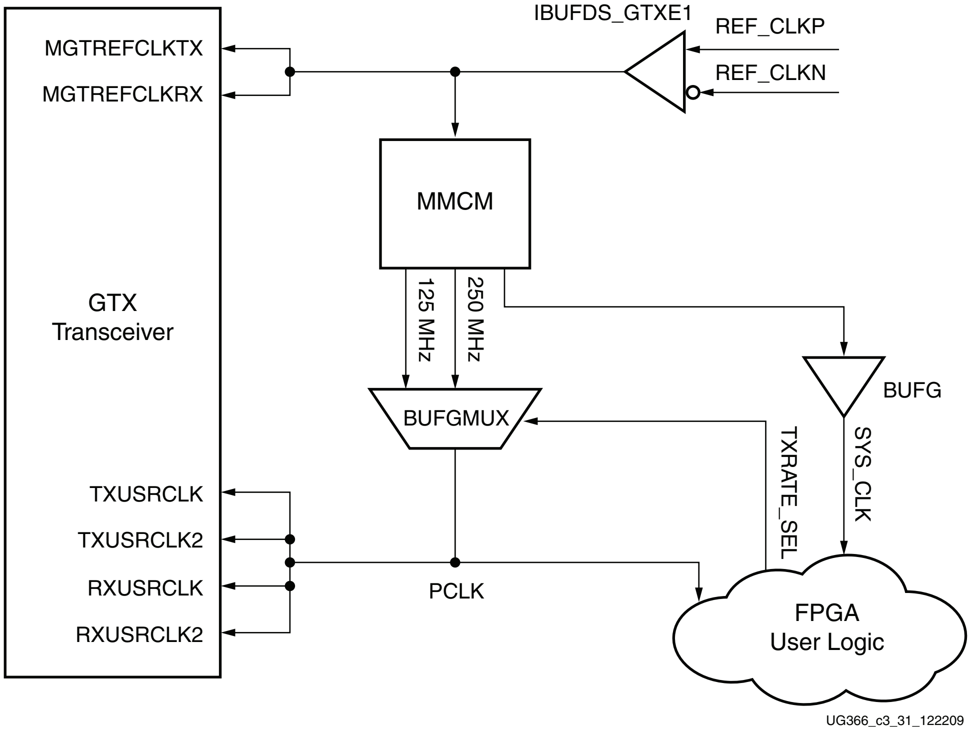


Figure 3-29: **PCI Express Clocking**

Note: The IBUFDS_GTXE1 diagram in Figure 3-29 is a simplification. The output port ODIV2 is left floating, and the input port CEB is set to logic 0.

For details about placement constraints and restrictions on clocking resources (MMCM, BUFGCTRL, IBUFDS_GTXE1, BUFG, etc.), refer to the *Virtex-6 FPGA Clocking Resources User Guide*.

Rate Change Use Mode for PCI Express 2.0 Operation

In PCI Express mode, TXRATE[1] must be tied to 1, and TXRATE[0] is used for line rate change. RXRATE[1:0] must be tied to TXRATE[1:0]. The TXPLL_DIVSEL_OUT attribute must be set to 2.

To perform a PCI Express line rate change, TXPOWERDOWN must be in the P0 power state, and TXELECIDLE must be asserted. A change in TXRATE[0] initiates a PCI Express line rate change. If an MMCM is used to generate the 125 MHz and 250 MHz user clocks, a BUFGMUX is recommended to be used. The TXRATEDONE single cycle pulse can be used as an timing indicator to change the clock frequency from BUFGMUX. The PHYSTATUS single cycle pulse indicates that the PCI Express line rate change is completed.

After a PCI Express line rate change and if the TX buffer is bypassed, it is necessary to perform a TX phase alignment and reset the TX PLL output clock divider. Refer to TX Buffer Bypass, page 156 for more information on TX phase alignment.

The rate change procedure for PCI Express 2.0 operation is:

1. Set TXPOWERDOWN[1:0] = 00.
2. Assert TXELECIDLE.
3. Set a new TXRATE[1:0] rate:
   a. Tie TXRATE[1] = 1 in PCIe mode
   b. Set TXRATE[0] = 0 (2.5 Gb/s line rate)
   c. Set TXRATE[0] = 1 (5.0 Gb/s line rate)
4. Wait for the TXRATEDONE pulse.
5. Select a new TXUSRCLK2 or user clock frequency from BUFGMUX:
   a. Select TXUSRCLK2 = 125 MHz (2.5 Gb/s line rate)
   b. Select TXUSRCLK2 = 250 MHz (5.0 Gb/s line rate)
6. Wait for the PHYSTATUS pulse.
7. Deassert TXELECIDLE. TXELECIDLE can instead be deasserted after TX phase alignment.
8. Perform TX phase alignment after the rate change (see TX Phase Alignment after Rate Change Use Mode, page 160 for more information).

Figure 3-30 shows the rate change timing waveforms for PCI Express operation. In Figure 3-30, TXRATE_SEL is a FPGA user logic control signal that selects a new TXUSRCLK2 frequency (125 MHz or 250 MHz) from the BUFGMUX for a rate change.

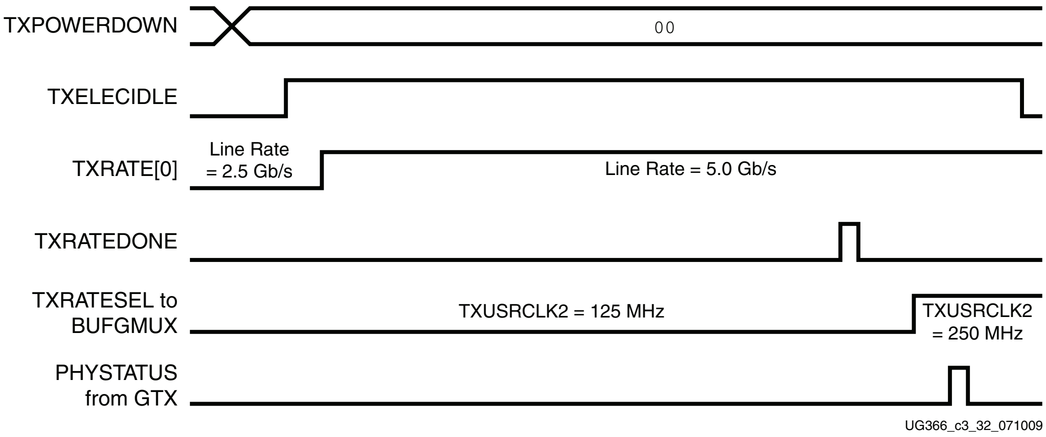


Figure 3-30: **Rate Change Timing for PCI Express Operation**

TX Configurable Driver

Functional Description

The GTX 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

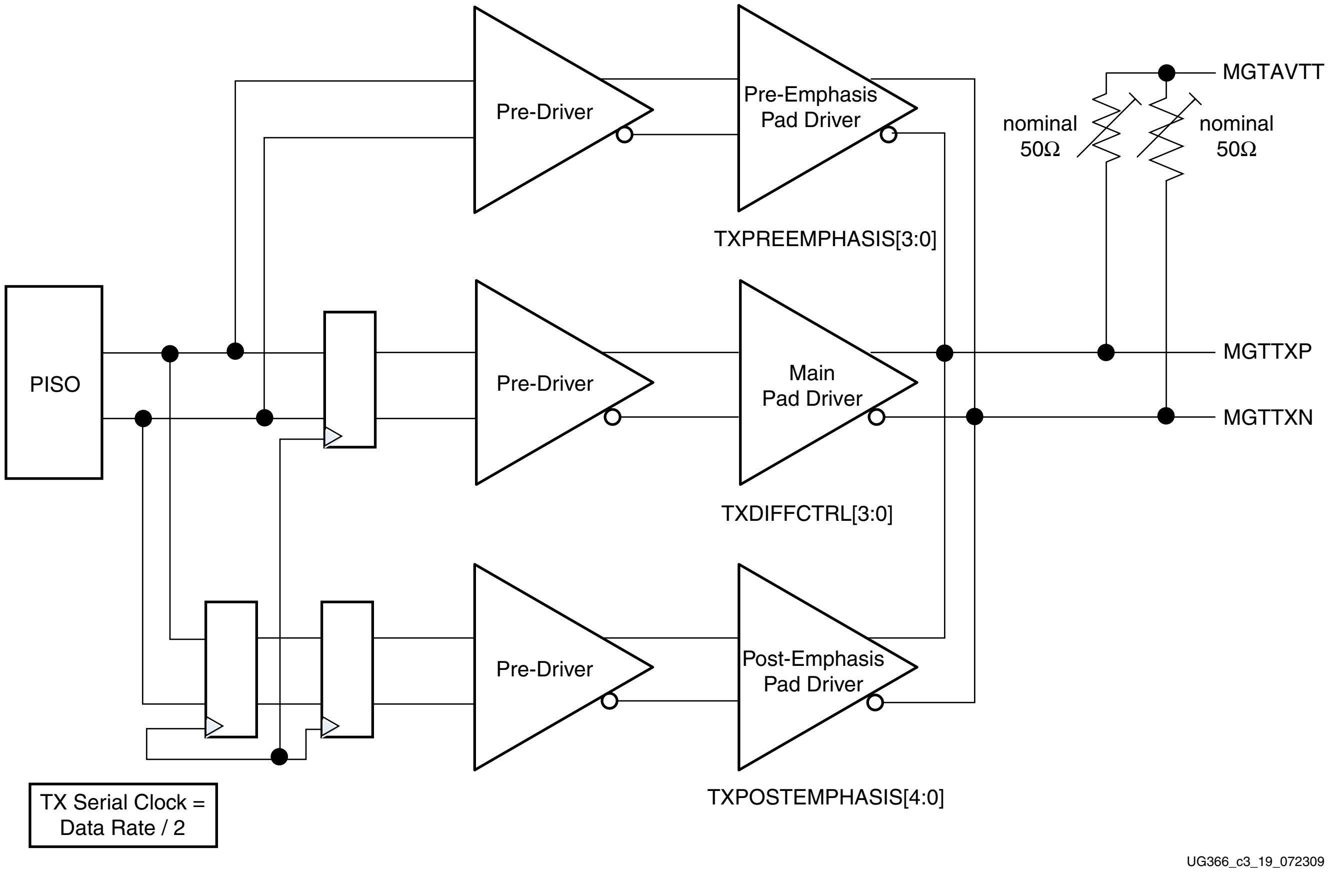


Figure 3-31: **TX Driver Block Diagram**

Ports and Attributes

Table 3-31 defines the TX configurable driver ports.

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

Port	Dir	Clock Domain	Description
TXBUFDIFFCTRL[2:0]	In	Async	Pre-driver Swing Control. The default is `3'b100` (nominal value). Do *not* modify this value.
TXDEEMPH	In	Async	TX de-emphasis control for PCI Express PIPE interface. This signal is mapped internally to TXPOSTEMPHASIS 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)[1]

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

<table>
<tr><th>Port</th><th>Dir</th><th>Clock Domain</th><th>Description</th></tr>
<tr><td>TXDIFFCTRL[3:0]</td><td>In</td><td>Async</td><td>Driver Swing Control. The default is user specified. All listed values (mV_{PPD}) are typical.
<table>
<tr><th>[3:0]</th><th>mV_{PPD}</th></tr>
<tr><td>0000</td><td>110</td></tr>
<tr><td>0001</td><td>210</td></tr>
<tr><td>0010</td><td>310</td></tr>
<tr><td>0011</td><td>400</td></tr>
<tr><td>0100</td><td>480</td></tr>
<tr><td>0101</td><td>570</td></tr>
<tr><td>0110</td><td>660</td></tr>
<tr><td>0111</td><td>740</td></tr>
<tr><td>1000</td><td>810</td></tr>
<tr><td>1001</td><td>880</td></tr>
<tr><td>1010</td><td>940</td></tr>
<tr><td>1011</td><td>990</td></tr>
<tr><td>1100</td><td>1040</td></tr>
<tr><td>1101</td><td>1080</td></tr>
<tr><td>1110</td><td>1110</td></tr>
<tr><td>1111</td><td>1130</td></tr>
</table>
</td></tr>
<tr><td>TXELECIDLE</td><td>In</td><td>TXUSRCLK2, Async</td><td>TXPDOWNASYNCH makes this pin asynchronous.</td></tr>
<tr><td>TXINHIBIT</td><td>In</td><td>TXUSRCLK2</td><td>When High, this signal blocks transmission of TXDATA and forces TXP to 0 and TXN to 1.</td></tr>
<tr><td>TXMARGIN[2:0]</td><td>In</td><td>Async</td><td>TX Margin control for PCI Express PIPE Interface[1]. 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>
</table>

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

<table>
<tr><th>Port</th><th>Dir</th><th>Clock Domain</th><th>Description</th></tr>
<tr><td>TXPDOWNASYNCH</td><td>In</td><td>Async</td><td>Determines if TXELECIDLE and TXPOWERDOWN should be treated as synchronous or asynchronous signals. Enables compliance during cold and warm PCI Express resets.
0: Sets TXELECIDLE and TXPOWERDOWN to synchronous mode.
1: Sets TXELECIDLE and TXPOWERDOWN to asynchronous mode.</td></tr>
<tr><td>TXPOSTEMPHASIS[4:0]</td><td>In</td><td>Async</td><td>Transmitter Post-Cursor TX Pre-Emphasis Control. The default is user specified. All listed values (dB) are typical.
<table>
<tr><th>[4:0]</th><th>dB (Post-Emphasis Magnitude)</th></tr>
<tr><td>00000</td><td>0.18</td></tr>
<tr><td>00001</td><td>0.19</td></tr>
<tr><td>00010</td><td>0.18</td></tr>
<tr><td>00011</td><td>0.18</td></tr>
<tr><td>00100</td><td>0.18</td></tr>
<tr><td>00101</td><td>0.18</td></tr>
<tr><td>00110</td><td>0.18</td></tr>
<tr><td>00111</td><td>0.18</td></tr>
<tr><td>01000</td><td>0.19</td></tr>
<tr><td>01001</td><td>0.2</td></tr>
<tr><td>01010</td><td>0.39</td></tr>
<tr><td>01011</td><td>0.63</td></tr>
<tr><td>01100</td><td>0.82</td></tr>
<tr><td>01101</td><td>1.07</td></tr>
<tr><td>01110</td><td>1.32</td></tr>
<tr><td>01111</td><td>1.6</td></tr>
<tr><td>10000</td><td>1.65</td></tr>
<tr><td>10001</td><td>1.94</td></tr>
<tr><td>10010</td><td>2.21</td></tr>
<tr><td>10011</td><td>2.52</td></tr>
<tr><td>10100</td><td>2.76</td></tr>
<tr><td>10101</td><td>3.08</td></tr>
<tr><td>10110</td><td>3.41</td></tr>
<tr><td>10111</td><td>3.77</td></tr>
<tr><td>11000</td><td>3.97</td></tr>
<tr><td>11001</td><td>4.36</td></tr>
<tr><td>11010</td><td>4.73</td></tr>
<tr><td>11011</td><td>5.16</td></tr>
<tr><td>11100</td><td>5.47</td></tr>
<tr><td>11101</td><td>5.93</td></tr>
<tr><td>11110</td><td>6.38</td></tr>
<tr><td>11111</td><td>6.89</td></tr>
</table>
</td></tr>
</table>

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

<table>
<tr><th>Port</th><th>Dir</th><th>Clock Domain</th><th>Description</th></tr>
<tr><td>TXPREEMPHASIS[3:0]</td><td>In</td><td>Async</td><td>Transmitter Pre-Cursor TX Pre-Emphasis Control· The default is user specified. All listed values (dB) are typical.
<table>
<tr><th>[3:0]</th><th>dB (Pre-Emphasis Magnitude)</th></tr>
<tr><td>0000</td><td>0.15</td></tr>
<tr><td>0001</td><td>0.3</td></tr>
<tr><td>0010</td><td>0.45</td></tr>
<tr><td>0011</td><td>0.61</td></tr>
<tr><td>0100</td><td>0.74</td></tr>
<tr><td>0101</td><td>0.91</td></tr>
<tr><td>0110</td><td>1.07</td></tr>
<tr><td>0111</td><td>1.25</td></tr>
<tr><td>1000</td><td>1.36</td></tr>
<tr><td>1001</td><td>1.55</td></tr>
<tr><td>1010</td><td>1.74</td></tr>
<tr><td>1011</td><td>1.94</td></tr>
<tr><td>1100</td><td>2.11</td></tr>
<tr><td>1101</td><td>2.32</td></tr>
<tr><td>1110</td><td>2.54</td></tr>
<tr><td>1111</td><td>2.77</td></tr>
</table></td></tr>
<tr><td>TXP
TXN</td><td>Out (Pad)</td><td>TX Serial Clock</td><td>TXP and TXN are 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 41) and brought to the top level of the design.</td></tr>
<tr><td>TXSWING</td><td>In</td><td>Async</td><td>TX swing control for PCI Express PIPE Interface[1]. This signal is mapped internally to TXDIFFCTRL/TXBUFDIFFCTRL.
0: Full Swing
1: Low Swing</td></tr>
</table>

Notes:

[1] As per PHY Interface for the PCI Express Architecture, PCI Express 2.0, Revision 0.5, August 2008.

Table 3-32 defines the TX configurable driver attributes.

Table 3-32: **TX Configurable Driver Attributes**

Attribute	Type	Description
TX_DEEMPH_0[4:0]	5-bit Binary	This attribute has the value of TXPOSTEMPHASIS[4:0] that has to be mapped when TXDEEMPH = 0. TX_DEEMPH_0[4:0] = TXPOSTEMPHASIS[4:0]. The default is `11010` (nominal value). Do *not* modify this value.
TX_DEEMPH_1[4:0]	5-bit Binary	This attribute has the value of TXPOSTEMPHASIS[4:0] that has to be mapped when TXDEEMPH = 1. TX_DEEMPH_1[4:0] = TXPOSTEMPHASIS[4:0]. The default is `10000` (nominal value). Do *not* modify this value.

Table 3-32: **TX Configurable Driver Attributes *(Cont'd)***

Attribute	Type	Description
TX_DRIVE_MODE	String	This attribute selects whether PCI Express PIPE Spec pins or TX Drive Control pins control the TX driver. The default is "DIRECT". DIRECT: TXBUFFDIFFCTRL, TXDIFFCTRL, and TXPOSTEMPHASIS control the TX driver settings. PIPE: TXDEEMPH, TXMARGIN, and TXSWING control the TX driver settings.
TX_MARGIN_FULL_0[6:0]	7-bit Binary	This attribute has the value of TXBUFFDIFFCTRL[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]. The default is `7'b1001110` (1040 mV_{PPD} typical). Do *not* modify this value.
TX_MARGIN_FULL_1[6:0]	7-bit Binary	This attribute has the value of TXBUFFDIFFCTRL[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]. The default is `7'b1001001` (850 mV_{PPD} typical). Do *not* modify this value.
TX_MARGIN_FULL_2[6:0]	7-bit Binary	This attribute has the value of TXBUFFDIFFCTRL[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]. The default is `7'b1000101` (515 mV_{PPD} typical). Do *not* modify this value.
TX_MARGIN_FULL_3[6:0]	7-bit Binary	This attribute has the value of TXBUFFDIFFCTRL[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]. The default is `7'b1000010` (290 mV_{PPD} typical). Do *not* modify this value.
TX_MARGIN_FULL_4[6:0]	7-bit Binary	This attribute has the value of TXBUFFDIFFCTRL[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]. The default is `7'b1000000` (130 mV_{PPD} typical). Do *not* modify this value.
TX_MARGIN_LOW_0[6:0]	7-bit Binary	This attribute has the value of TXBUFFDIFFCTRL[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]. The default is `7'b1000110` (590 mV_{PPD} typical). Do *not* modify this value.
TX_MARGIN_LOW_1[6:0]	7-bit Binary	This attribute has the value of TXBUFFDIFFCTRL[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]. The default is `7'b1000100` (445 mV_{PPD} typical). Do *not* modify this value.

Table 3-32: **TX Configurable Driver Attributes *(Cont'd)***

Attribute	Type	Description
TX_MARGIN_LOW_2[6:0]	7-bit Binary	This attribute has the value of TXBUFFDIFFCTRL[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]. The default is `7'b1000010` (290 mV_{PPD} typical). Do *not* modify this value.
TX_MARGIN_LOW_3[6:0]	7-bit Binary	This attribute has the value of TXBUFFDIFFCTRL[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]. The default is `7'b1000000` (130 mV_{PPD} typical). Do *not* modify this value.
TX_MARGIN_LOW_4[6:0]	7-bit Binary	This attribute has the value of TXBUFFDIFFCTRL[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]. The default is `7'b1000000` (130 mV_{PPD} typical). Do *not* modify this value.

Use Modes – TX Driver

General

TX_DRIVE_MODE is set to "DIRECT".

Based on the application requirement, TXDIFFCTRL, TXPREEMPHASIS, and TXPOSTEMPHASIS values are set to the appropriate values.

PCIe Mode

TX_DRIVE_MODE is set to "PIPE".

Then the PHY Interface for the PCI Express Architecture, PCI Express 3.0 Document, Revision 0.5, August 2008 is followed to use the TXMARGIN, TXSWING, and TXDEEMPH signals. See Table 3-31, page 174 for TXMARGIN, TXSWING, TXDEEMPH, and TXPOSTEMPHASIS mappings.

Customizable User Presets

TX_DRIVE_MODE is set to "PIPE".

To make the interface easier to use, the TX_MARGIN_*_*, TX_DEEMPH_* attributes can be set to the user defaults and then TXSWING, TXMARGIN, and TXDEEMP can be used to control the TX driver. See Table 3-31, page 174 for TXMARGIN, TXSWING, TXDEEMPH, and TXPOSTEMPHASIS mappings.

This is used in a backplane situation where the presets (TX_MARGIN_*_*, TXSWING) correspond to different slot numbers.

Use Mode – Resistor Calibration

For more information on the on-chip resistor calibration, refer to Termination Resistor Calibration Circuit, page 274.

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 3-32 shows the circuit model used for receive detection. The GTX transceiver must be in the P1 power-down state to perform receiver detection. Also receiver detection requires a 75 to 200 nF external coupling capacitor between the transmitter and receiver, and the receiver must be terminated to GND. The detection sequence starts with the assertion of TXDETECTRX. In response, the Receiver Detect logic drives TXN and TXP to $V_{DD} - V_{SWING}/2$ and then releases them. At the end of the sequence, RXSTATUS and PHYSTATUS reflect the results of the receiver detection.

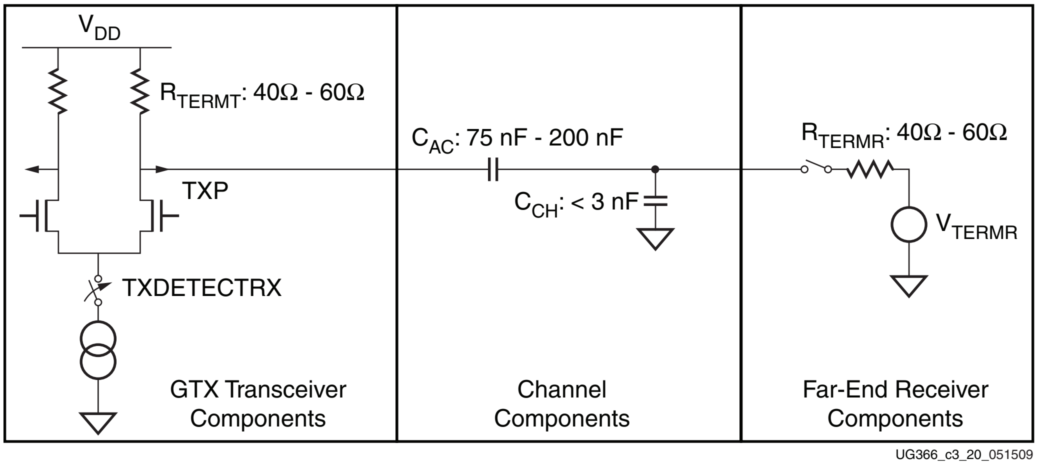


Figure 3-32: **Receiver Detection Circuit Model**

Ports and Attributes

Table 3-33 defines the TX receiver detect support ports.

Table 3-33: **TX Receiver Detect Support Ports**

Port	Dir	Clock Domain	Description
PHYSTATUS	Out	RXUSRCLK2/ Async	This signal is asserted High to indicate completion of several PHY functions, including power management state transitions and receiver detection. When this signal transitions during entry and exit from P2 and RXUSRCLK2 is not running, the signaling is asynchronous.

Table 3-33: **TX Receiver Detect Support Ports *(Cont'd)***

Port	Dir	Clock Domain	Description
RXPOWERDOWN[1:0]	In	Async	These inputs control the power state of the TX and RX links. The encoding complies with the PCI Express specification encoding. TX and RX can be powered down separately. `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)
TXPOWERDOWN[1:0]		TXUSRCLK2	
RXSTATUS[2:0]	Out	RXUSRCLK2	PCI Only usage. `000`: Receiver not present (when in receiver detection sequence)/Received data OK (during normal operation). `001`: Reserved. `010`: Reserved. `011`: Receiver present (when in receiver detection sequence). `100`: 8B/10B decode error. `101`: Elastic buffer overflow. Different than defined in the PIPE specification. `110`: Elastic buffer underflow. Different than defined in the PIPE specification. `111`: Receive disparity error.
TXDETECTRX	In	TXUSRCLK2	This input activates the receive detection sequence. The sequence ends when PHYSTATUS is asserted to indicate that the results of the test are ready on RXSTATUS.

There are no TX receiver detect support attributes.

TX Out-of-Band Signaling

Functional Description

Each GTX 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. GTX transceiver support for SATA/SAS OOB signaling consists of the analog circuitry required to encode the OOB signal state and state machines to format bursts of OOB signals for SATA/SAS COM sequences.

Each GTX transceiver also supports SATA and SAS auto-negotiation by allowing the timing of the COM sequences to be changed based on the divider settings used for the TX line rate.

Note: The GTX transceiver transmits the ALIGNp primitive character within each OOB burst pulse at the current line rate operation.

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

Ports and Attributes

Table 3-34 defines the TX OOB ports.

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

Port	Direction	Clock Domain	Description
COMFINISH	Out	TXUSRCLK2	This output indicates completion of transmission of the last SAS or SATA TXCOM sequence.
TXCOMINIT	In	TXUSRCLK2	This input initiates the transmission of the TXCOMINIT sequence.
TXCOMSAS	In	TXUSRCLK2	This input initiates the transmission of the TXCOMSAS sequence.
TXCOMWAKE	In	TXUSRCLK2	This input initiates the transmission of the TXCOMWAKE sequence.
TXELECIDLE	In	TXUSRCLK2	When in the PCIe P2 power state, this signal controls whether an electrical idle or a beacon indication is driven onto the TX pair. When in SATA mode, keep TXELECIDLE High for generating SATA/SAS OOB COM signaling.
TXPOWERDOWN[1:0]	In	TXUSRCLK2	This input powers down the TX lanes. It is primarily used for PCIe designs. Use TXPOWERDOWN = 00 for operating SATA OOB signaling.

Table 3-35 defines the TX OOB attributes.

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

Attribute	Type	Description
COM_BURST_VAL	4-bit Binary	This attribute determines the number of bursts in a COM sequence.
TXPLL_SATA	2-bit Binary	Reserved. Use only recommended values from the Virtex-6 FPGA GTX Transceiver Wizard.
TX_IDLE_DEASSERT_DELAY	3-bit Binary	Programmable delay between TXELECIDLE de-assertion to TXP/N exiting electrical idle. Use only recommended values from the Virtex-6 FPGA GTX Transceiver Wizard.
TX_IDLE_ASSERT_DELAY	3-bit Binary	Programmable delay between TXELECIDLE assertion to TXP/N entering electrical idle. Use only recommended values from the Virtex-6 FPGA GTX Transceiver Wizard.

The GTX transceiver supports four signaling modes: three for SATA/SAS operations and one for PCI Express operations. The use of these mechanisms is mutually exclusive.

Chapter 4

Receiver

RX Overview

This chapter shows how to configure and use each of the functional blocks inside the GTX transceiver receiver (RX). Each GTX transceiver includes an independent receiver, made up of a PCS and a PMA. Figure 4-1 shows the blocks of the RX. High-speed serial data flows from traces on the board into the PMA of the RX, into the PCS, and finally into the FPGA logic.

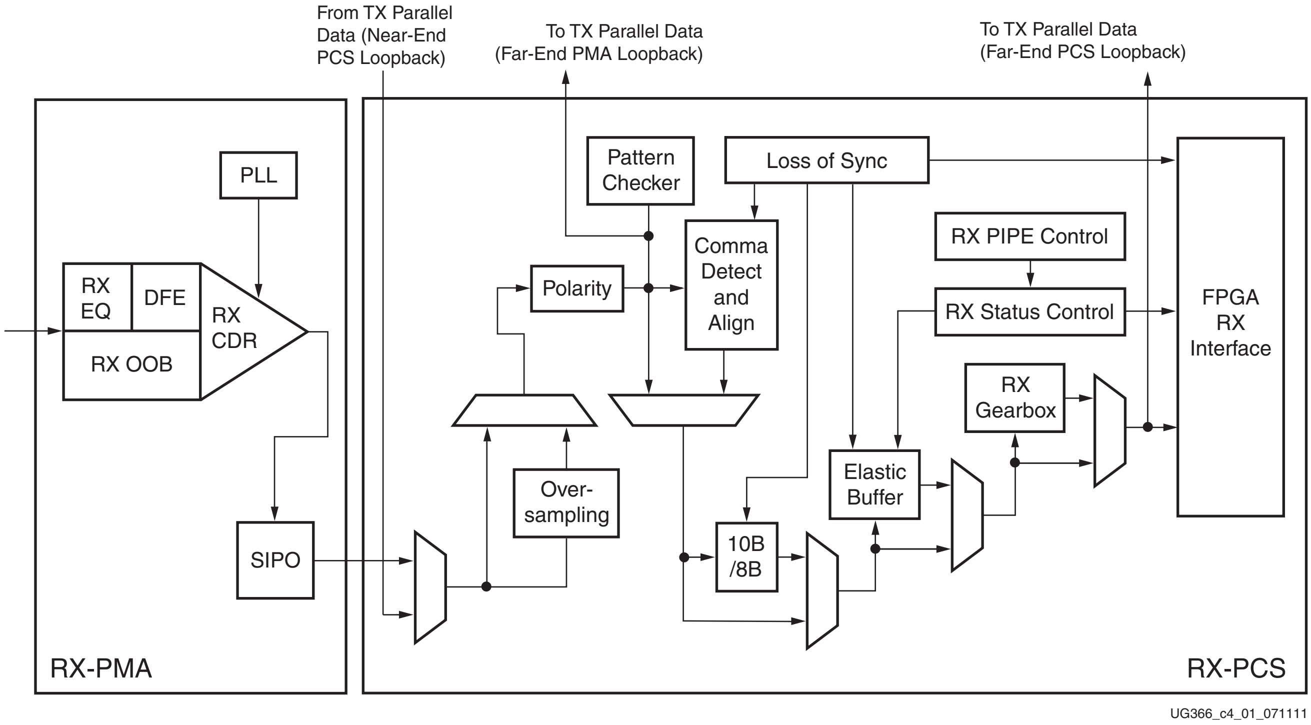


Figure 4-1: **GTX Transceiver RX Block Diagram**

The key elements within the GTX transceiver RX are:

1. RX Analog Front End, page 184
2. RX Out-of-Band Signaling, page 191
3. RX Equalizer, page 193
4. RX CDR, page 203
5. RX Fabric Clock Output Control, page 206
6. RX Margin Analysis, page 209

RX Analog Front End

Functional Description

The RX analog front end (AFE) is a high-speed current-mode input differential buffer. It has the following features

- Configurable RX termination voltage
- Bypassable on-chip coupling capacitors
- Calibrated termination resistors

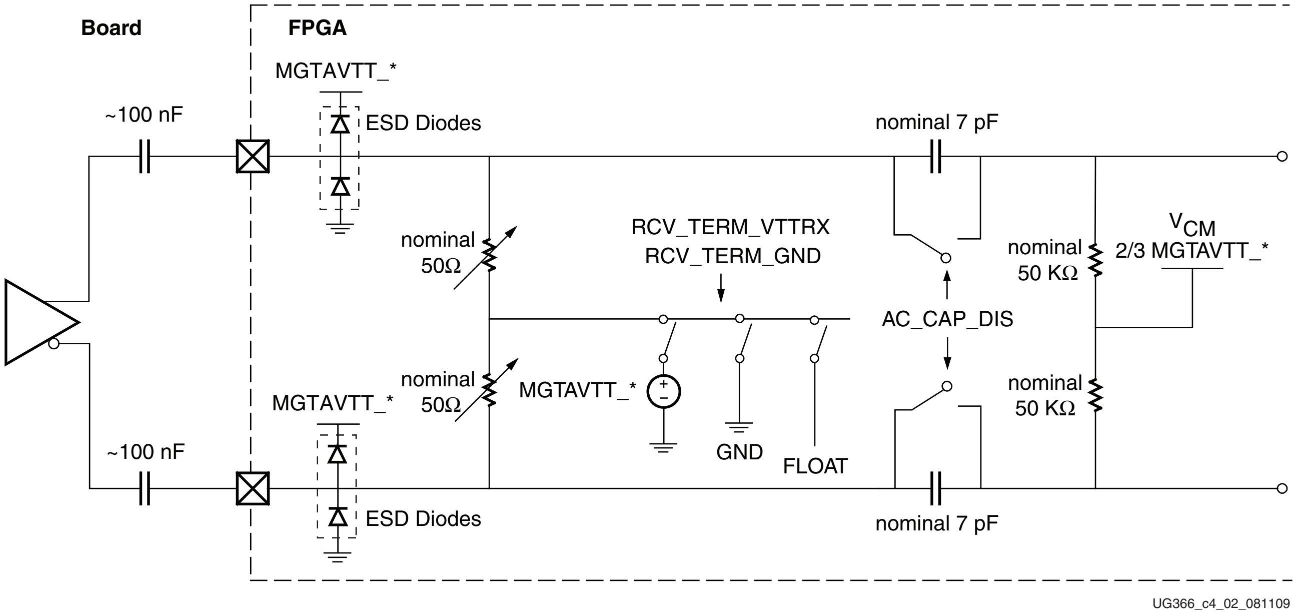


Figure 4-2: **RX AFE Block Diagram**

Note: MGTAVTT_* refers to MGTAVTT_S of the south package power plane and MGTAVTT_N of the north package power plane, as outlined in Figure 5-4, page 277.

Ports and Attributes

Table 4-1 defines the RX AFE ports.

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

Port	Dir	Clock Domain	Description
RXN RXP	In (Pad)	RX Serial Clock	RXN and RXP are 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 41) and brought to the top level of the design.

Table 4-2 defines the RX analog front end attributes.

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

Attribute	Type	Description
AC_CAP_DIS	Boolean	Bypasses the built-in AC coupling in the receiver. TRUE: Built-in AC coupling capacitors are bypassed. DC coupling to the receiver is possible. FALSE: Built-in AC coupling capacitors are enabled. See Chapter 5, Board Design Guidelines, for details about when it is appropriate to add an additional external AC coupling capacitor based on data rate or protocol. See Use Modes – RX Termination for valid RX Termination combinations.
CM_TRIM[1:0]	2-bit Binary	Adjusts the input common mode levels. These levels are automatically set in the Virtex®-6 FPGA GTX Transceiver Wizard.
RCV_TERM_GND	Boolean	Activates the Ground reference for the receiver termination network. The default for this attribute is TRUE for PCI Express designs. Generally, for all other protocols, the default setting is FALSE. TRUE: Ground reference for receiver termination activated. FALSE: Ground reference for receiver termination disabled. See Use Modes – RX Termination for valid RX Termination combinations.
RCV_TERM_VTTRX	Boolean	Activates the MGTAVTT_* reference for the receiver termination network. The default for this attribute is FALSE for PCI Express designs. For all other protocols, the default setting is TRUE. The setting is FALSE when using AC coupling. TRUE: MGTAVTT_* reference for receiver termination activated. FALSE: MGTAVTT_* reference for receiver termination disabled. See Use Modes – RX Termination for valid RX Termination combinations.
TERMINATION_CTRL[4:0]	5-bit Binary	Controls the internal termination calibration circuit. This is an advance feature.
TERMINATION_OVRD	Boolean	Selects whether the external 100Ω precision resistor connected to the MGTRREF pin or an override value is used, as defined by TERMINATION_CTRL[4:0]. This is an advance feature.

Use Modes – RX Termination

Table 4-3 lists the possible settings for RCV_TERM_GND and RCV_TERM_VTTRX.

Table 4-3: **RX Termination Voltage and Attribute Mapping**

RCV_TERM_GND	RCV_TERM_VTTRX	RX Termination Voltage
FALSE	FALSE	FLOAT
FALSE	TRUE	MGTAVTT_*
TRUE	FALSE	GND
TRUE	TRUE	RESERVED / NOT ALLOWED

Note: MGTAVTT_* refers to MGTAVTT_S of the south package power plane and MGTAVTT_N of the north package power plane as outlined in Figure 5-4, page 277.

Table 4-4 outlines the recommended settings for RX termination in Use Mode 1. Figure 4-3 shows the Use Mode 1 configuration.

Table 4-4: **RX Termination Use Mode 1 Configuration**

Use Mode	External AC Coupling	Term Voltage	Internal AC Coupling	Internal Bias	Recommended Max Swing mV_{DPP}	Suggested Protocols and Usage Notes
1	On	GND	On	800 mV	1200	**Protocol**: PCIe **Attribute Settings:** AC_CAP_DIS = FALSE RCV_TERM_GND = TRUE RCV_TERM_VTTRX = FALSE

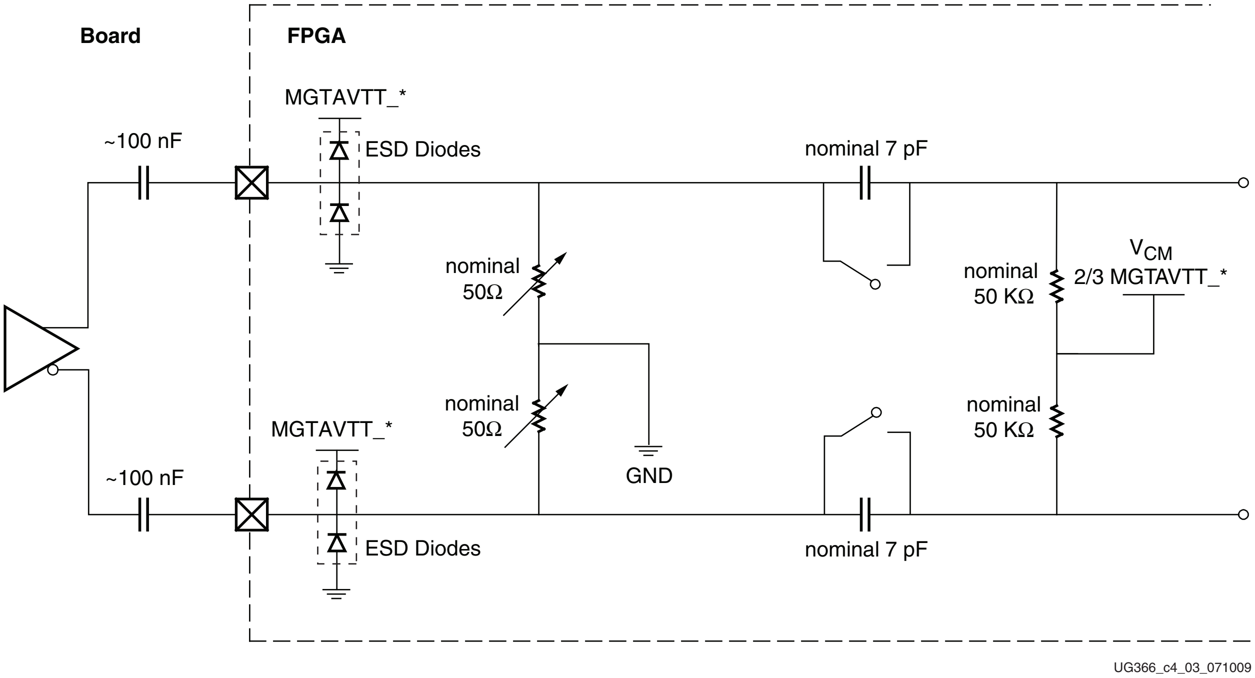


Figure 4-3: **RX Termination Use Mode 1 Configuration**

Table 4-5 outlines the recommended settings for RX termination in Use Mode 2. Figure 4-4 shows the Use Mode 2 configuration.

Table 4-5: **RX Termination Use Mode 2 Configuration and Notes**

Use Mode	External AC Coupling	Term Voltage	Internal AC Coupling	Internal Bias	Recommended Max Swing mV_{DPP}	Suggested Protocols and Usage Notes
2	On	VTT	Off	800 mV	1200	**Protocol**: Backplane, CEI-6 (1200 mV_{DPP}), Wireless, Serial RapidIO, DisplayPort (0.4V/0.6V/0.8V option). Protocols such as HD-SDI and SD-SDI can also use this termination mode if cable equalizer devices are used between SDI cable and the Xilinx FPGA SerDes interface. In this configuration, the signal swing from the cable equalizer device is usually less than 1200 mV. **Attribute Settings:** AC_CAP_DIS = TRUE RCV_TERM_GND = FALSE RCV_TERM_VTTRX = TRUE

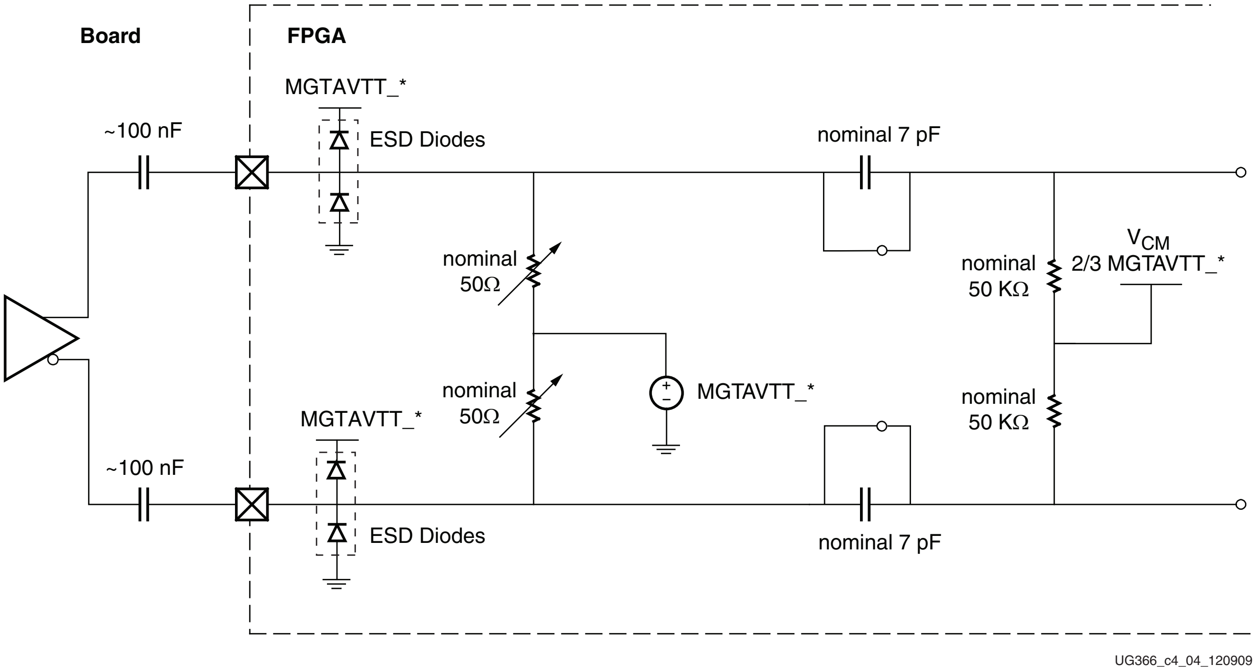


Figure 4-4: **RX Termination Use Mode 2 Configuration**

Table 4-6 outlines the recommended settings for RX termination in Use Mode 3. Figure 4-5 shows the Use Mode 3 configuration.

Table 4-6: **RX Termination Use Mode 3 Configuration and Notes**

Use Mode	External AC Coupling	Term Voltage	Internal AC Coupling	Internal Bias	Recommended Max Swing mV_{DPP}	Suggested Protocols and Usage Notes
3	ON	Float	OFF	800 mV	2000	**Protocol**: Optical IF (SONET/SDH/OTU), SFP+, HD/SD-SDI, XAUI (1600 mVdpp), GbE, DisplayPort (1.2V option) **Attribute Settings:** AC_CAP_DIS = TRUE RCV_TERM_GND = FALSE RCV_TERM_VTTRX = FALSE **Notes:** Similar to Case 2, but for interfaces with > 1200 mV swing. This option has a lower bandwidth than Case 2.

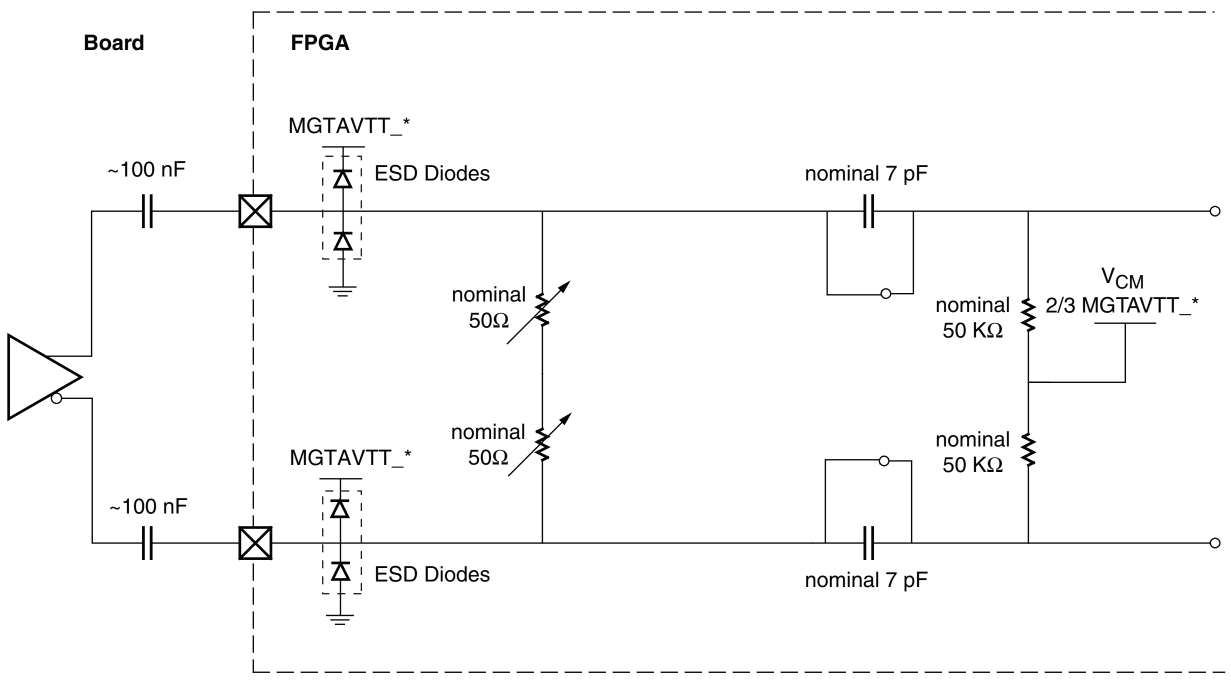


Figure 4-5: **RX Termination Use Mode 3 Configuration**

Table 4-7 outlines the recommended settings for RX termination in Use Mode 4. Figure 4-6 shows the Use Mode 4 configuration.

Table 4-7: **RX Termination Use Mode 4 Configuration and Notes**

Use Mode	External AC Coupling	Term Voltage	Internal AC Coupling	Internal Bias	Recommended Max Swing mV_{DPP}	Suggested Protocols and Usage Notes
4	OFF	V_{TT}	OFF	800 mV	1200	**Protocol**: Custom GTX-GTX chip-to-chip interface **Attribute Settings:** AC_CAP_DIS = TRUE RCV_TERM_GND = FALSE RCV_TERM_VTTRX = TRUE **Notes:** Recommended for use if the TX termination voltage is 1.2V. If the TX termination voltage is not 1.2V, DC current will result with possible signal distortion. This allows for DC coupling on the board. Not recommended for the backplane due to interoperability with SerDes where the TX termination voltage is not 1.2V.

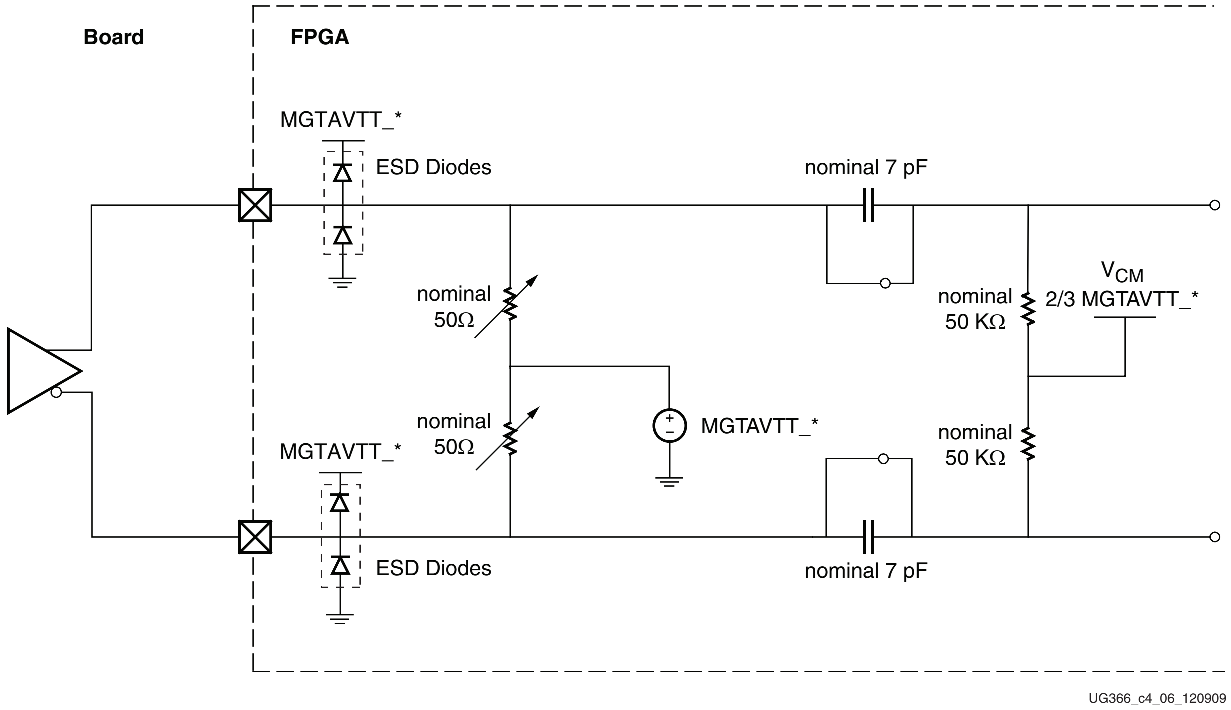


Figure 4-6: **RX Termination Use Mode 4 Configuration**

Table 4-8 outlines the recommended settings for RX termination in Use Mode 5. Figure 4-7 shows the Use Mode 5 configuration.

Table 4-8: **RX Termination Use Mode 5 Configuration and Notes**

Use Mode	External AC Coupling	Term Voltage	Internal AC Coupling	Internal Bias	Recommended Max Swing mV_{DPP}	Suggested Protocols and Usage Notes
5	OFF	Float/ GND	OFF	800 mV	1600	**Protocol**: GPON **Attribute Settings:** AC_CAP_DIS = TRUE RCV_TERM_GND = TRUE / FALSE RCV_TERM_VTTRX = FALSE **Notes:** True DC mode. A DC common mode of 2/3 MGTAVTT_* (800 mV) is required. To achieve this, an external level shifting network might be required.

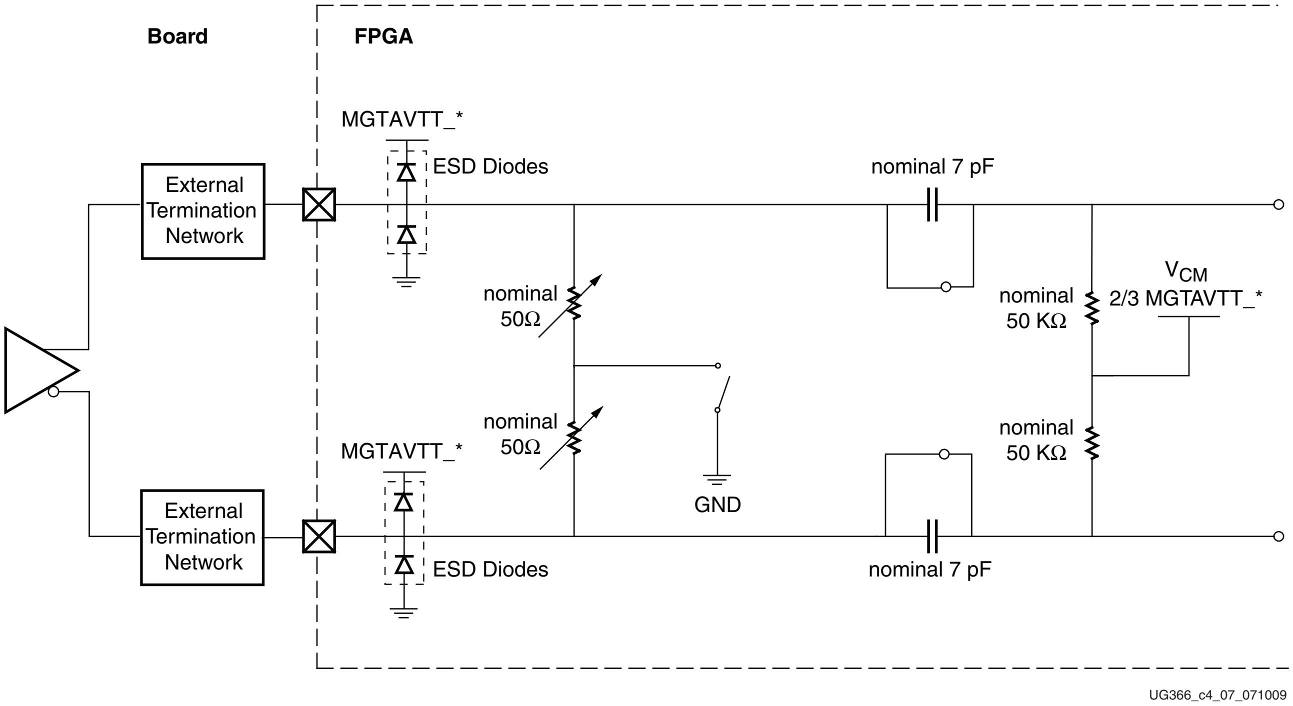


Figure 4-7: **RX Termination Use Mode 5 Configuration**

Use Mode – Resistor Calibration

For more information on the on-chip resistor calibration, refer to Termination Resistor Calibration Circuit, page 274.

RX Out-of-Band Signaling

Functional Description

The GTX transceiver 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 transceiver 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 transceiver 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 4-9 defines the RX OOB ports.

Table 4-9: **RX OOB Ports**

Port	Direction	Clock Domain	Description
COMINITDET	Out	RXUSRCLK2	Indicates detection of a COMINIT sequence.
COMSASDET	Out	RXUSRCLK2	Indicates detection of a COMSAS sequence.
COMWAKEDET	Out	RXUSRCLK2	Indicates detection of a COMWAKE sequence.
GATERXELECIDLE	In	Async	Optional port. This port is tied to zero for PCIe and SATA modes (default). For other usages, this port is asserted High to gate the RXELECIDLE output (see Figure 4-8).
IGNORESIGDET	In	Async	Optional port. This port is tied to zero for PCIe and SATA modes (default). For other usages, this port is asserted High to disable RX signal electrical idle detection logic from resetting other GTX transceivers logic, including comma detection, RX elastic buffer logic, and DFE logic (see Figure 4-8). When IGNORESIGDET is set High, it prevents the following attributes from automatically triggering an internal reset or hold on RX electrical idle: • RX_EN_IDLE_RESET_BUF • RX_EN_IDLE_HOLD_CDR • RX_EN_IDLE_RESET_FR • RX_EN_IDLE_RESET_PH • RX_EN_IDLE_HOLD_DFE
RXELECIDLE	Out	Async	Indicates the differential voltage between RXN and RXP dropped below the minimum threshold (OOBDETECT_THRESHOLD). Signals below this threshold are OOB signals. 1: OOB signal detected. The differential voltage is below the minimum threshold. 0: OOB signal not detected. The differential voltage is above the minimum threshold. This port is intended for PCI Express and SATA standards.

Table 4-9: **RX OOB Ports**

Port	Direction	Clock Domain	Description
RXSTATUS[2:0]	Out	RXUSRCLK2	RXSTATUS[2:0] are used only for PCIe mode, as defined by the PIPE specification.
RXVALID	Out	RXUSRCLK2	Indicates symbol lock and valid data on RXDATA and RXCHARISK[3:0] when High, as defined in the PIPE specification.

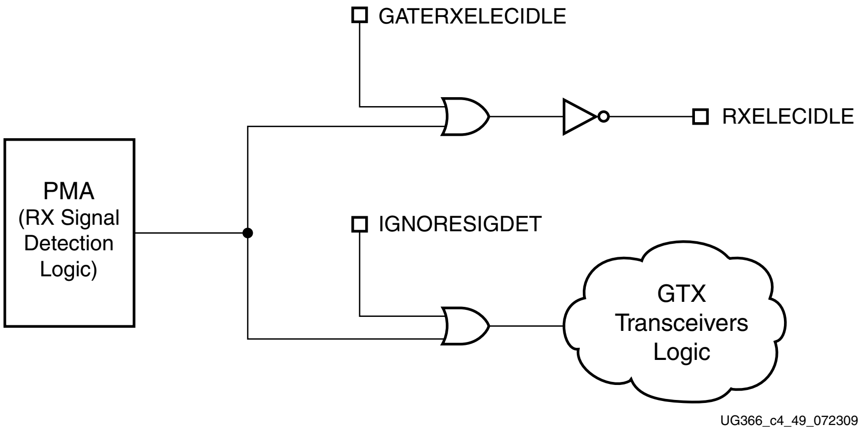


Figure 4-8: **Optional Ports GATERXELECIDLE and IGNORESIGDET**

Table 4-10 defines the RX OOB attributes.

Table 4-10: **RX OOB Attributes**

Attribute	Type	Description	
OOBDETECT_THRESHOLD	3-bit Binary	Sets the minimum differential voltage between RXN and RXP. When the differential voltage drops below this level, the incoming signal is considered an OOB signal. This 3-bit binary encoded attribute has the following nominal values of the OOB threshold voltage[1]:	
		Value	**OOB Nominal Threshold Voltage (mV)**
		000 – 010	Reserved
		011 (default)	100
		100 – 111	Reserved
SAS_MAX_COMSAS	6-bit Hex	Upper bound on the idle count during COMSAS for SAS	
SAS_MIN_COMSAS	6-bit Hex	Lower bound on the idle count during COMSAS for SAS	
SATA_BURST_VAL	3-bit Binary	Number of bursts to declare a COM match for SAS/SATA	
SATA_IDLE_VAL	3-bit Binary	Number of idles to declare a COM match for SAS/SATA	
SATA_MAX_BURST	6-bit Hex	Upper bound on an activity burst for COM FSM for SAS/SATA	
SATA_MAX_INIT	6-bit Hex	Upper bound on idle count during COMINIT/COMRESET for SAS/SATA	
SATA_MAX_WAKE	6-bit Hex	Upper bound on idle count during COMWAKE for SAS/SATA	
SATA_MIN_BURST	6-bit Hex	Lower bound on an activity burst for COM FSM for SAS/SATA	

Table 4-10: **RX OOB Attributes *(Cont'd)***

Attribute	Type	Description
SATA_MIN_INIT	6-bit Hex	Lower bound on idle count during COMINIT/COMRESET for SAS/SATA
SATA_MIN_WAKE	6-bit Hex	Lower bound on idle count during COMWAKE for SAS/SATA

Notes:

1. These are OOB nominal values. Consult the *Virtex-6 FPGA Data Sheet* for OOB specifications.

RX Equalizer

Functional Description

The RX has a continuous time RX equalization circuit and a decision feedback equalization circuit to compensate for high-frequency losses in the channel.

This continuous time RX equalization circuit can be tuned to meet the specific requirements of the physical channel used in the design by compensating for signal distortion due to high-frequency attenuation.

It is a 3-stage amplifier with the ability to boost the input signal at low, intermediate, and high frequencies. There are eight different frequency responses to accommodate several channels.

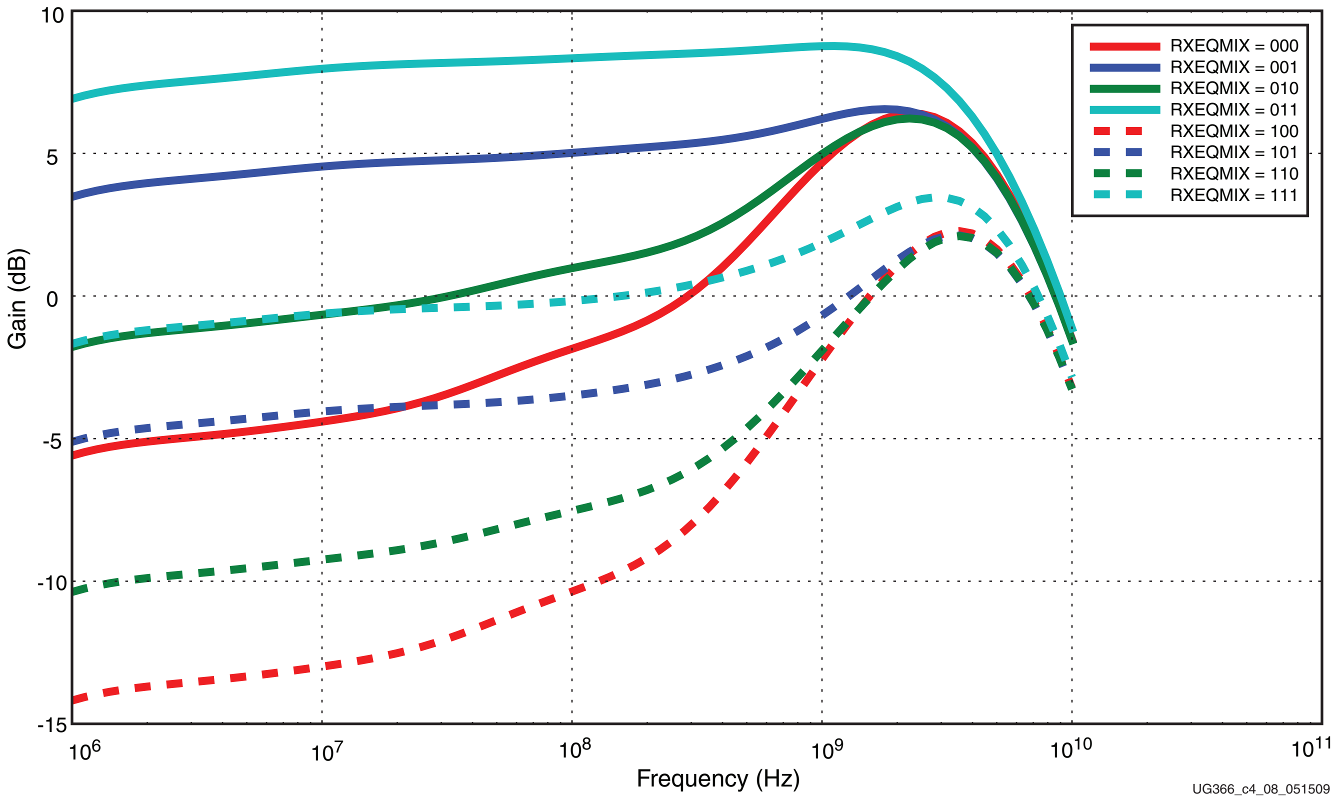


Figure 4-9: **Absolute Gain (Voltage Transfer Function)**

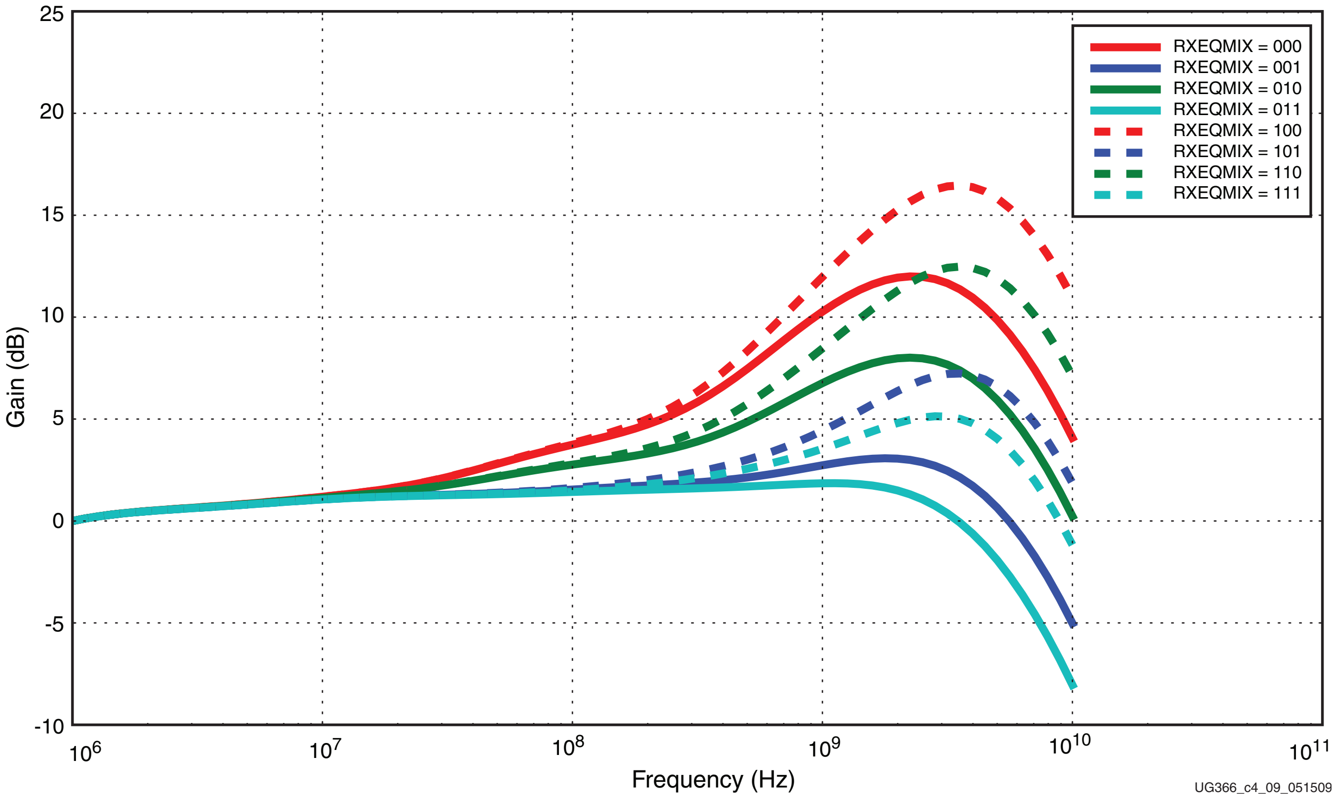


Figure 4-10: **Relative Gain Normalized to Low Frequency (Voltage Transfer Function)**

The Decision Feedback Equalizer enhances the internal eye by reducing the post-cursor tail of the transmitted bit. It is a 4-tap architecture.

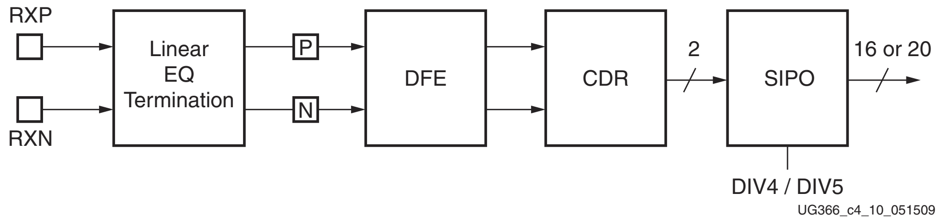


Figure 4-11: **DFE in the GTX Transceiver RX**

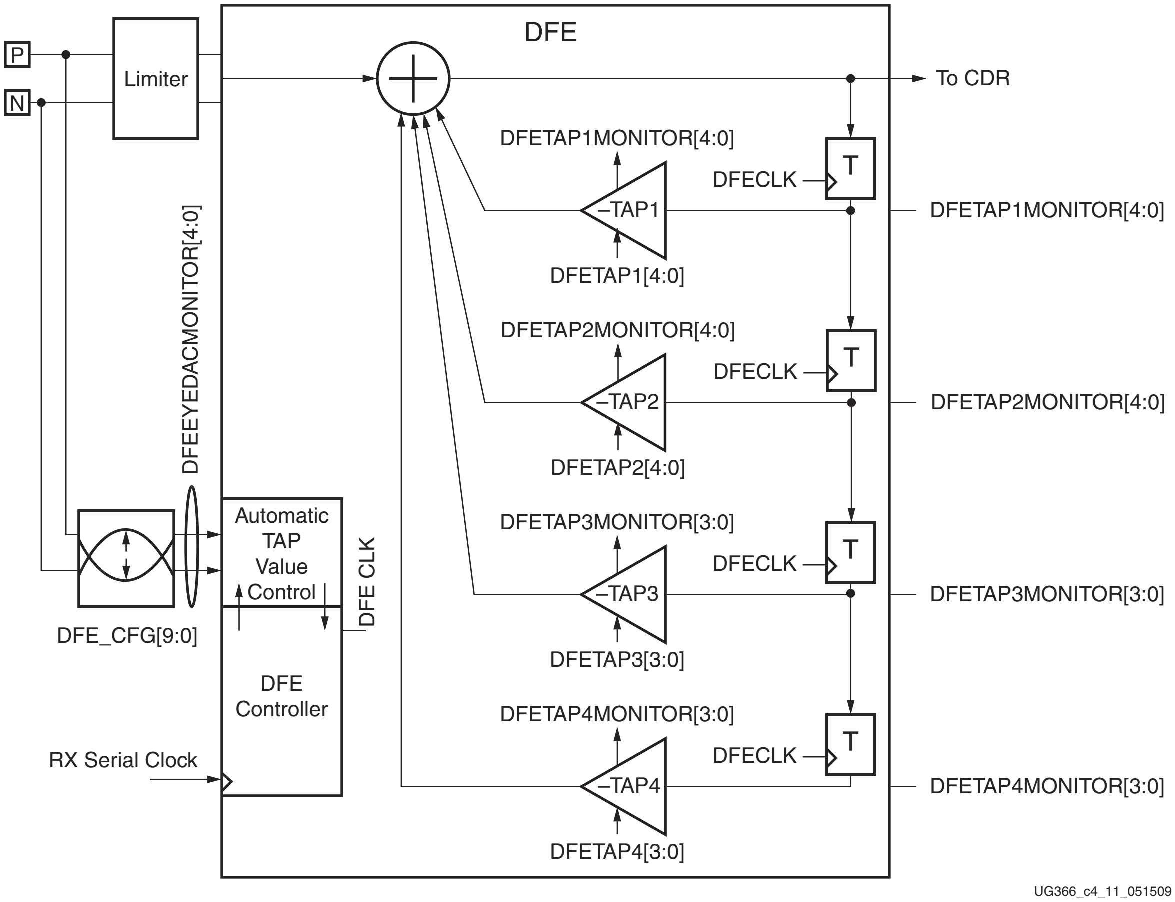


Figure 4-12: **DFE Conceptual View**

The DFE allows better compensation of transmission channel losses by providing a closer adjustment of filter parameters than when using a linear equalizer. However, a DFE cannot remove the pre-cursor of a transmitted bit. A linear equalizer allows pre-cursor and post-cursor attenuation, but has only a coarse adaptor to the transmission channel characteristic. The GTX transceiver DFE in the GTX transceiver RX is a discrete-time adaptive high-pass filter. The TAP values of the DFE are the coefficients of this filter can be either set manually or by the auto-calibration logic. The optimization criteria for the TAP values and the DFECLK delay is the vertical eye opening. The DFETAPOVRD port switches off auto-calibration and overrides the TAP values, and the DFEDLYOVRD port overrides the DFECLK delay.

Ports and Attributes

Table 4-11 defines the RX equalizer ports.

Table 4-11: **RX Equalizer Ports**

Port	Dir	Clock Domain	Description
DFECLKDLYADJ[5:0]	In	RXUSRCLK2	DFE clock delay adjust override for each transceiver. `000000`: Phase difference between the bit serial sample clock and the DFECLK is 0°. `111111`: Phase difference between the bit serial sample clock and the DFECLK is 90°. This DFE is automatically calibrated for optimal performance by a built-in state machine. This override value is only accepted when DFEDLYOVRD is 1.
DFECLKDLYADJMON[5:0]	Out	RXUSRCLK2	DFE clock delay calibration result monitor for each transceiver.
DFEDLYOVRD	In	RXUSRCLK2	Override enable for the DFE clock delay adjustment. `0`: Use the optimized DFE clock delay calibration value (recommended). `1`: Override the DFE clock delay calibration state machine with the value provided by DFECLKDLYADJ[5:0]. The optimized DFE clock delay calibration is done when GTXRXRESET is deasserted. When switching from override mode to the optimized mode dynamically, always toggle GTXRXRESET.
DFEEYEDACMON[4:0]	Out	RXUSRCLK2	Averaged Vertical Eye Height (voltage domain) used by the DFE as an optimization criterion. `11111`: Indicates approximately 200 mV$_{PPD}$ of internal eye opening.
DFESENSCAL[2:0]	Out	RXUSRCLK2	Sampler sensitivity self-calibration after the reset. `000`: Lowest offset `111`: Highest offset
DFETAP1[4:0]	In	RXUSRCLK2	DFE tap 1 weight value control for each transceiver (5-bit resolution).
DFETAP1MONITOR[4:0]	Out	RXUSRCLK2	DFE tap 1 weight value monitor for each transceiver (5-bit resolution).
DFETAP2[4:0]	In	RXUSRCLK2	DFE tap 2 weight value control for each transceiver (4-bit resolution plus 1-bit sign). For example, –2 is represented as `1 0010`.
DFETAP2MONITOR[4:0]	Out	RXUSRCLK2	DFE tap 2 weight value monitor for each transceiver (4-bit resolution plus 1-bit sign). For example, –2 is represented as `1 0010`.
DFETAP3[3:0]	In	RXUSRCLK2	DFE tap 3 weight value control for each transceiver (3-bit resolution plus 1-bit sign). For example, –2 is represented as `1 010`.

Table 4-11: **RX Equalizer Ports *(Cont'd)***

Port	Dir	Clock Domain	Description
DFETAP3MONITOR[3:0]	Out	RXUSRCLK2	DFE tap 3 weight value monitor for each transceiver (3-bit resolution plus 1-bit sign). For example, –2 is represented as `1 010`.
DFETAP4[3:0]	In	RXUSRCLK2	DFE tap 4 weight value control for each transceiver (3-bit resolution plus 1-bit sign). For example, –2 is represented as `1 010`.
DFETAP4MONITOR[3:0]	Out	RXUSRCLK2	DFE tap 4 weight value monitor for each transceiver (3-bit resolution plus 1-bit sign). For example, –2 is represented as `1 010`.
DFETAPOVRD	In	RXUSRCLK2	Override enable for the DFE tap values. 0: Use the optimized DFE tap self-adapting values. 1: Override the DFE self-adapting values (recommended) with the values provided by DFETAP1, DFETAP2, DFETAP3, and DFETAP4.
RXEQMIX[9:0]	In	Async	Receiver Equalization Control. Bits [9:3] are reserved. Only bits [2:0] should be used. The default is `3'b000`, user specific.

Table 4-12 defines the RX equalization attributes.

Table 4-12: **RX Equalization Attributes**

<table>
<tr><th>Attribute</th><th>Type</th><th colspan="2">Description</th></tr>
<tr><td>DFE_CAL_TIME[4:0]</td><td>5-bit Binary</td><td colspan="2">DFE calibration time. Set to 01100 for normal operation (default).</td></tr>
<tr><td rowspan="3">DFE_CFG[7:0]</td><td rowspan="3">8-bit Binary</td><td><b>DFE_CFG</b></td><td><b>Description</b></td></tr>
<tr><td>[7:3]</td><td>Limiter/EQAMP gain and power control
Set to 00011 (default)</td></tr>
<tr><td>[2:0]</td><td>Vertical Eye measure AMP gain and power control.
Set to 011 (default)</td></tr>
<tr><td>RX_EN_IDLE_HOLD_DFE</td><td>Boolean</td><td colspan="2">TRUE: Restores the DFE contents from internal registers after termination of an electrical idle state for PCI Express operation. Holds the DFE circuit in reset when an electrical idle condition is detected.
FALSE: (default)
<b>Note</b>: For channels with large attenuation (lossy channels typically exceeding 15 dB at Nyquist), it is recommended that RX_EN_IDLE_HOLD_DFE should be set to FALSE because fast transitioning data patterns like the 101010 sequence in CJPAT/CJTPAT can accidentally trigger an electrical idle</td></tr>
</table>

Use Mode – Continuous Time RX Linear Equalizer Only

Modes with greater gain at high frequencies are intended for lossy (usually longer) channels. Following is a simple way to determine how to use the RX equalizer:

1. Determine the operating data rate.
2. Determine the channel loss (board) in dB at data rate/2. This is the differential insertion loss from measured or extracted S-parameter data commonly referred to as Sdd21.
3. Pick the appropriate RXEQMIX setting from the relative gain plot.

   Always make sure that the transmit amplitude is sufficient when picking modes with a higher gain because there is DC attenuation of the signal. Reference the absolute gain plot.

Based on these results, the appropriate setting of RXEQMIX can be picked.

These settings must always be verified in hardware because there are several considerations, such as discontinuities in the system, jitter on the TX clocks, which cannot be compensated for by the linear equalizer.

Example

Data Rate = 6.25 Gb/s, which means the fundamental frequency is 3.125 GHz.

Channel Loss in dB at 3.125 GHz = 12 dB

RXEQMIX[2:0] = `000` or `110` is sufficient.

This setting allows compensation of 12 dB at 3.125 GHz hence providing the needed gain.

Use Mode – Fixed Tap Mode

This mode requires that DFETAPOVRD = 1 (DFETAPx value override). It is recommended that DFEDLYOVRD = 0 (optimized DFE clock delay calibration). DFE_CFG bits should be at their default values. The values to be written to the DFE taps are applied to DFETAPx. DFETAPxMONITOR reflects the value entered.

The chip-to-chip and backplane application examples in this section provide starting points for setting DFETAP1 and RXEQMIX based on the trace length and the associated loss of the channel. DFETAP2, DFETAP3, and DFETAP4 are set to 0 in these examples. For channels with different characteristics or lengths than given in the examples, the designer should use the example with the closest match as a starting point. Either the Virtex-6 FPGA GTX Transceiver Wizard example design or IBERT can be used to fine tune the settings to the user’s particular channel.

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

Table 4-13 provides DFETAP1 and RXEQMIX settings for 3.125 Gb/s operation for chip-to-chip applications.

Table 4-13: **DFETAP1 and RXEQMIX at 3.125 Gb/s for Chip-to-Chip Applications**

Nominal Trace Length on FR4 Substrate (inches [mm])	Loss (dB) @ 1.5625 GHz	TAP1				
		RXEQMIX = 000	RXEQMIX = 010	RXEQMIX = 101	RXEQMIX = 110	RXEQMIX = 111
13 [330.2]	2.8		0	0		
20 [508.0]	4.7		0	0		
30 [762.0]	8.9	0	0			
42 [1066.8]	11.7	0	0			
50 [1270.0]	13.0	0	0			
70 [1778.0]	17.8	0			0	

Notes:

1. GTX transceiver TXDIFFCTRL = `4'b1010`, TXPOSTEMPHASIS = `5'b00000`, and TXPREEMPHASIS = `4'b0000`.

Table 4-14 provides DFETAP1 and RXEQMIX settings for 4.25 Gb/s operation for chip-to-chip applications.

Table 4-14: **DFETAP1 and RXEQMIX at 4.25 Gb/s for Chip-to-Chip Applications**

Nominal Trace Length on FR4 Substrate (inches [mm])	Loss (dB) @ 2.125 GHz	TAP1				
		RXEQMIX = 000	RXEQMIX = 010	RXEQMIX = 101	RXEQMIX = 110	RXEQMIX = 111
13 [330.2]	3.4			0		0
20 [508.0]	5.7		0	0		
30 [762.0]	11.7		0		0	
42 [1066.8]	14.8		0		0	
50 [1270.0]	16.1		10		0	
70 [1778.0]	22.3	5			5	

Notes:

1. GTX transceiver TXDIFFCTRL = `4'b1010`, TXPOSTEMPHASIS = `5'b00000`, and TXPREEMPHASIS = `4'b0000`.

Table 4-15 provides DFETAP1 and RXEQMIX settings for 5 Gb/s operation for chip-to-chip applications.

Table 4-15: **DFETAP1 and RXEQMIX at 5 Gb/s for Chip-to-Chip Applications**

Nominal Trace Length on FR4 Substrate (inches [mm])	Loss (dB) @ 2.5 GHz	TAP1				
		RXEQMIX = 000	RXEQMIX = 010	RXEQMIX = 101	RXEQMIX = 110	RXEQMIX = 111
13 [330.2]	3.8			0		0
20 [508.0]	6.4		0	0		
30 [762.0]	13.0		5		0	

Table 4-15: **DFETAP1 and RXEQMIX at 5 Gb/s for Chip-to-Chip Applications *(Cont'd)***

Nominal Trace Length on FR4 Substrate (inches [mm])	Loss (dB) @ 2.5 GHz	TAP1				
		RXEQMIX = 000	RXEQMIX = 010	RXEQMIX = 101	RXEQMIX = 110	RXEQMIX = 111
42 [1066.8]	16.5		5		0	
50 [1270.0]	18.0		10		5	
70 [1778.0]	25.1	10			5	

Notes:

1. GTX transceiver TXDIFFCTRL = `4'b1010`, TXPOSTEMPHASIS = `5'b00000`, and TXPREEMPHASIS = `4'b0000`.

Table 4-16 provides DFETAP1 and RXEQMIX settings for 6.25 Gb/s operation for chip-to-chip applications.

Table 4-16: **DFETAP1 and RXEQMIX at 6.25 Gb/s for Chip-to-Chip Applications**

Nominal Trace Length on FR4 Substrate (inches [mm])	Loss (dB) @ 3.125 GHz	TAP1				
		RXEQMIX = 000	RXEQMIX = 010	RXEQMIX = 101	RXEQMIX = 110	RXEQMIX = 111
13 [330.2]	4.5			0		0
20 [508.0]	7.9		0	0		
30 [762.0]	15.4		10		0	
42 [1066.8]	20.1		10		5	
50 [1270.0]	22.7		15		5	
70 [1778.0]	30.2	15			15	

Notes:

1. GTX transceiver TXDIFFCTRL = `4'b1010`, TXPOSTEMPHASIS = `5'b00000`, and TXPREEMPHASIS = `4'b0000`.

Example – RX Linear Equalizer and DFE Settings for Backplane Applications

Table 4-17 provides DFETAP1 and RXEQMIX settings for 3.125 Gb/s operation for backplane applications.

Table 4-17: **DFETAP1 and RXEQMIX at 3.125 Gb/s for Backplane Applications**

Nominal Trace Length on FR4 Substrate (inches [mm])	Loss (dB) @ 1.5625 GHz	TAP1		
		RXEQMIX = 000	RXEQMIX = 010	RXEQMIX = 110
24 [609.6](2)	8.1		0	0
40 [1016.0](3)	10.3		0	0

Table 4-17: **DFETAP1 and RXEQMIX at 3.125 Gb/s for Backplane Applications *(Cont'd)***

Nominal Trace Length on FR4 Substrate (inches [mm])	Loss (dB) @ 1.5625 GHz	TAP1		
		RXEQMIX = 000	RXEQMIX = 010	RXEQMIX = 110
60 [1524.0][4]	13.6	0		0

Notes:

1. GTX transceiver TXDIFFCTRL = 4'b1010, TXPOSTEMPHASIS = 5'b00000, and TXPREEMPHASIS = 4'b0000.
2. Nominal 24 inches [609.6 mm] trace length = 6 inches [152.4 mm] backplane + 2 HmZD or eHSD connectors (trace length neglected) + 18 inches [457.2 mm] on line cards.
3. Nominal 40 inches [1016.0 mm] trace length = 20 inches [508 mm] backplane + 2 HmZD or eHSD connectors (trace length neglected) + 20 inches [508 mm] on line cards.
4. Nominal 60 inches [1524.0 mm] trace length = 40 inches [1016 mm] backplane + 2 HmZD or eHSD connectors (trace length neglected) + 20 inches [508 mm] on line cards.

Table 4-18 provides DFETAP1 and RXEQMIX settings for 4.25 Gb/s operation for backplane applications.

Table 4-18: **DFETAP1 and RXEQMIX at 4.25 Gb/s for Backplane Applications**

Nominal Trace Length on FR4 Substrate (inches [mm])	Loss (dB) @ 2.125 GHz	TAP1		
		RXEQMIX = 000	RXEQMIX = 010	RXEQMIX = 110
24 [609.6][2]	9.9		0	0
40 [1016.0][3]	12.7		5	0
60 [1524.0][4]	17.2	5		5

Notes:

1. GTX transceiver TXDIFFCTRL = 4'b1010, TXPOSTEMPHASIS = 5'b00000, and TXPREEMPHASIS = 4'b0000.
2. Nominal 24 inches [609.6 mm] trace length = 6 inches [152.4 mm] backplane + 2 HmZD or eHSD connectors (trace length neglected) + 18 inches [457.2 mm] on line cards.
3. Nominal 40 inches [1016.0 mm] trace length = 20 inches [508 mm] backplane + 2 HmZD or eHSD connectors (trace length neglected) + 20 inches [508 mm] on line cards.
4. Nominal 60 inches [1524.0 mm] trace length = 40 inches [1016 mm] backplane + 2 HmZD or eHSD connectors (trace length neglected) + 20 inches [508 mm] on line cards.

Table 4-19 provides DFETAP1 and RXEQMIX settings for 5 Gb/s operation for backplane applications.

Table 4-19: **DFETAP1 and RXEQMIX at 5 Gb/s for Backplane Applications**

Nominal Trace Length on FR4 Substrate (inches [mm])	Loss (dB) @ 2.5 GHz	TAP1		
		RXEQMIX = 000	RXEQMIX = 010	RXEQMIX = 110
24 [609.6][2]	12.2		0	0
40 [1016.0][3]	15.1		10	0

Table 4-19: **DFETAP1 and RXEQMIX at 5 Gb/s for Backplane Applications *(Cont'd)***

Nominal Trace Length on FR4 Substrate (inches [mm])	Loss (dB) @ 2.5 GHz	TAP1		
		RXEQMIX = 000	RXEQMIX = 010	RXEQMIX = 110
60 [1524.0](4)	20.4	10		5

Notes:

1. GTX transceiver TXDIFFCTRL = `4'b1010`, TXPOSTEMPHASIS = `5'b00000`, and TXPREEMPHASIS = `4'b0000`.
2. Nominal 24 inches [609.6 mm] trace length = 6 inches [152.4 mm] backplane + 2 HmZD or eHSD connectors (trace length neglected) + 18 inches [457.2 mm] on line cards.
3. Nominal 40 inches [1016.0 mm] trace length = 20 inches [508 mm] backplane + 2 HmZD or eHSD connectors (trace length neglected) + 20 inches [508 mm] on line cards.
4. Nominal 60 inches [1524.0 mm] trace length = 40 inches [1016 mm] backplane + 2 HmZD or eHSD connectors (trace length neglected) + 20 inches [508 mm] on line cards.

Table 4-20 provides DFETAP1 and RXEQMIX settings for 6.25 Gb/s operation for backplane applications.

Table 4-20: **DFETAP1 and RXEQMIX at 6.25 Gb/s for Backplane Applications**

Nominal Trace Length on FR4 Substrate (inches [mm])	Loss (dB) @ 3.125 GHz	TAP1		
		RXEQMIX = 000	RXEQMIX = 010	RXEQMIX = 110
24 [609.6](2)	15.8		10	0
40 [1016.0](3)	19.9		15	5
60 [1524.0](4)	25.8	15		10

Notes:

1. GTX transceiver TXDIFFCTRL = `4'b1010`, TXPOSTEMPHASIS = `5'b00000`, and TXPREEMPHASIS = `4'b0000`.
2. Nominal 24 inches [609.6 mm] trace length = 6 inches [152.4 mm] backplane + 2 HmZD or eHSD connectors (trace length neglected) + 18 inches [457.2 mm] on line cards.
3. Nominal 40 inches [1016.0 mm] trace length = 20 inches [508 mm] backplane + 2 HmZD or eHSD connectors (trace length neglected) + 20 inches [508 mm] on line cards.
4. Nominal 60 inches [1524.0 mm] trace length = 40 inches [1016 mm] backplane + 2 HmZD or eHSD connectors (trace length neglected) + 20 inches [508 mm] on line cards.

Use Mode – Auto-To-Fix

This is an advanced feature.

Use Mode – Auto

This is an advanced feature.

RX CDR

Functional Description

The RX Clock Data Recovery (CDR) circuit in each GTX transceiver extracts the recovered clock and data from an incoming data stream. Figure 4-13 illustrates the architecture of the CDR block. Clock paths are shown with dotted lines for clarity.

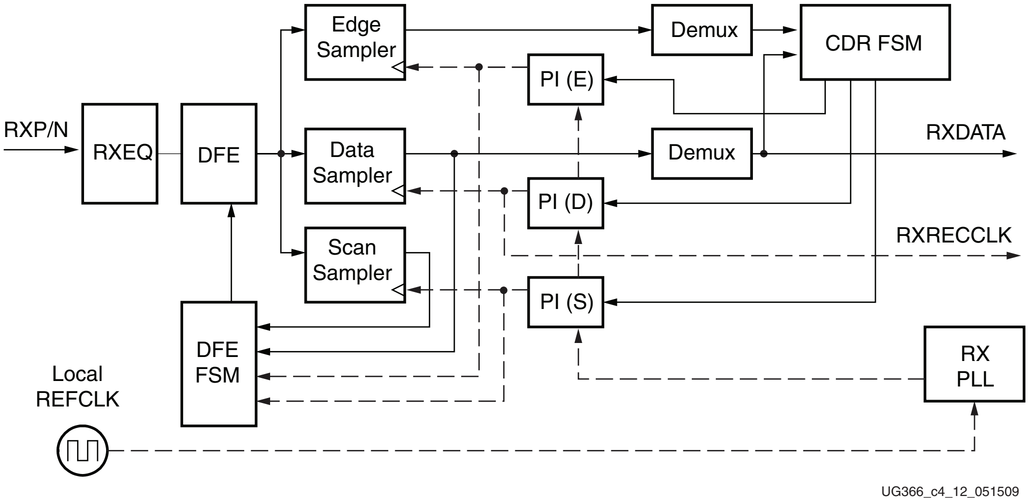


Figure 4-13: **CDR Detail**

The GTX 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 DFE 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 (PI). 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. A third sampler, the scan sampler, usually uses the same phase as the data sampler and is used by the DFE block.

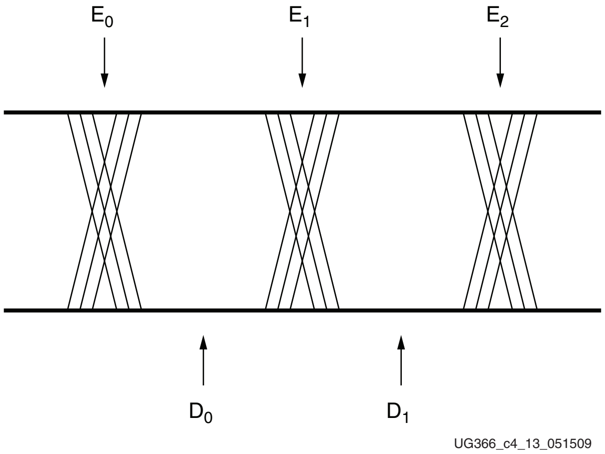


Figure 4-14: **CDR Sampler Positions**

The RX PLL 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 stream that can have a frequency offset, usually no more than ±1000 PPM, from the local PLL reference clock.

Methods for detecting CDR lock include:

- Finding known data in the incoming data stream (for example, commas or A1/A2 framing characters). In general, several consecutive known data patterns must be received without error to indicate a CDR lock.
- Using the LOS state machine (see RX Loss-of-Sync State Machine, page 225). If incoming data is 8B/10B encoded and the CDR is locked, the LOS state machine moves to the SYNC_ACQUIRED state and stays there.

Ports and Attributes

Table 4-21 defines the RX CDR ports.

Table 4-21: **RX CDR Ports**

Port	Dir	Clock Domain	Description
RXCDRRESET	In	Async	Active-High CDR reset that resets the CDR logic and the RX part of the PCS for this channel. This signal needs to be asserted whenever the frequency of the RX PLL changes.
RXRATE[1:0]	In	RXUSRCLK2	This port and RX PLL output divider attribute, RXPLL_DIVSEL_OUT, defines the line rate for the receiver based on the RX PLL frequency. Refer to RX Fabric Clock Output Control, page 206 for more details.

Table 4-22 defines the RX CDR attributes.

Table 4-22: **RX CDR Attributes**

Attribute	Type	Description
CDR_PH_ADJ_TIME	5-bit Binary	Reserved. Use only recommended values from the Virtex-6 FPGA GTX Transceiver Wizard. This attribute defines the delay after deassertion of the CDR phase reset before the optional reset sequence of PCI Express operation is complete during electrical idle.
PMA_CDR_SCAN	27-bit Hex	Reserved. Use only recommended values from the Virtex-6 FPGA GTX Transceiver Wizard.
PMA_RX_CFG	25-bit Hex	Reserved. Use only recommended values from the Virtex-6 FPGA GTX Transceiver Wizard. For lock-to-reference operation, set PMA_RX_CFG to `25'h05CE000`. In normal operation, set PMA_RX_CFG to the default value generated by the Virtex-6 FPGA GTX Transceiver Wizard.
RX_EN_IDLE_HOLD_CDR	Boolean	When set to TRUE, it 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 RX_EN_IDLE_HOLD_CDR should be set to FALSE because fast transitioning data patterns like the `101010` sequence in CJPAT/CJTPAT can accidentally trigger an electrical idle.

Table 4-22: **RX CDR Attributes *(Cont'd)***

Attribute	Type	Description
RX_EN_IDLE_RESET_FR	Boolean	When set to TRUE, it 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 RX_EN_IDLE_RESET_FR should be set to FALSE because fast transitioning data patterns like the `101010` sequence in CJPAT/CJTPAT can accidentally trigger an electrical idle.
RX_EN_IDLE_RESET_PH	Boolean	When set to TRUE, it 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 RX_EN_IDLE_RESET_PH should be set to FALSE because fast transitioning data patterns like the `101010` sequence in CJPAT/CJTPAT can accidentally trigger an electrical idle.
RX_EYE_SCANMODE	2-bit Binary	This attribute should be set to `00` for normal operation. Refer to RX Margin Analysis, page 209 for detailed information.
RXPLL_DIVSEL_OUT	Integer	This divider defines the nominal line rate for the receiver. It can be set to 1, 2, or 4. RX Line Rate = RX PLL Clock * 2/PLL_RXDIVSEL_OUT

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 4-15.

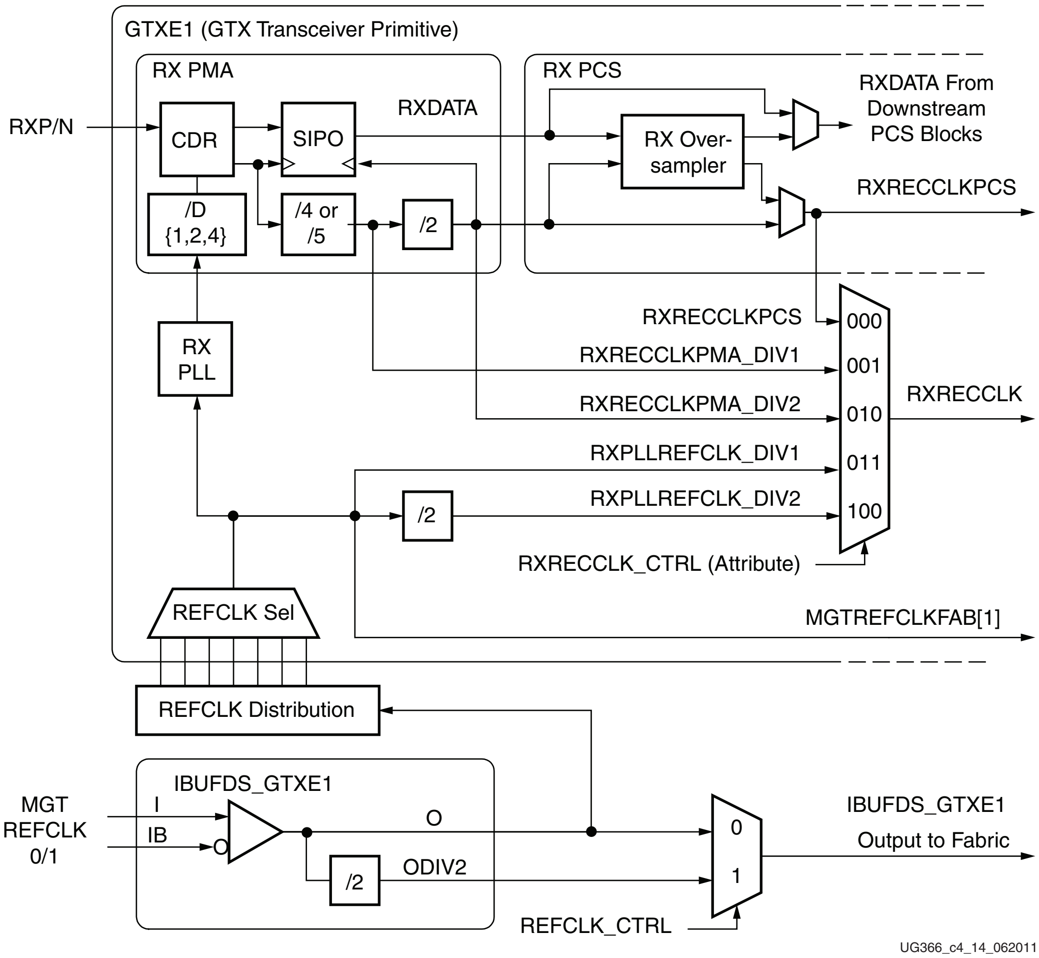


Figure 4-15: **RX Serial and Parallel Clock Divider Detail**

Notes relevant to Figure 4-15:

1. RXRECCLKPCS and MGTREFCLKFAB[1] are redundant outputs. Use RXRECCLK 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_GTXE1's O or ODIV2 outputs to the fabric.
3. IBUFDS_GTXE1 is a redundant output for additional clocking scheme flexibility.
4. The selection of the /4 or /5 divider block is dependent on RX_DATA_WIDTH (see Table 4-56, page 269):
   - /4 is selected when the internal data width is 16
   - /5 is selected when the internal data width is 20

Serial Clock Divider

Each receiver PMA module has a D divider that divides down the clock from the PLL for lower line rate support. This divider 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 RXPLL_DIVSEL_OUT attribute must be set to the appropriate value, and the RXRATE[1:0] port needs to be tied to 00.

To use the D divider in multiple line rate applications, the RXRATE[1:0] port is used to dynamically select the D divider value. The RXPLL_DIVSEL_OUT attribute and the RXRATE[1:0] port must select the same D divider value upon device configuration. After device configuration, the RXRATE[1:0] is used to dynamically change the D divider value.

The control for the serial divider is described in Table 4-23. For details about the line rate range per speed grade, refer to the *Virtex-6 FPGA Data Sheet*.

Table 4-23: **RX PLL Output Divider Setting**

Line Rate Range (Gb/s)	D Divider Value	Static Setting via Attribute	Dynamic Control via Ports
2.40 to 6.60	1	RXPLL_DIVSEL_OUT = 1 RXRATE[1:0] = 00	RXRATE[1:0] = 11
1.20 to 3.3	2	RXPLL_DIVSEL_OUT = 2 RXRATE[1:0] = 00	RXRATE[1:0] = 10
0.60 to 1.65	4	RXPLL_DIVSEL_OUT = 4 RXRATE[1:0] = 00	RXRATE[1:0] = 01

Parallel Clock Divider and Selector

The recovered clock 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. The parallel clock divider block can output a 1-byte or 2-byte data width recovered clock.

When the recovered clock is needed, the recommended clock output is RXRECCLK. The attribute RXRECCLK_CTRL controls the input selector and allows the following clocks to be output via RXRECCLK port:

- RXRECCLKPCS: This clock should only be used when the RX Oversampling block is enabled. The RX Oversampling block divides down the RXRECCLKPMA_DIV2 clock to match the 5X oversampled data rate.
- RXRECCLKPMA_DIV1/DIV2: This is the recovered clock after the CDR and is used when the RX Oversampling block is not used.
- RXPLLREFCLK_DIV1/DIV2: This is the input reference clock to the RX PLL and is typically not used. For usages that do not require outputting a recovered clock to the fabric, RXPLLREFCLK_DIV1/DIV2 can be used as the system clock. However, TXOUTCLK is usually used as system clock.

The RXRECCLKPMA_DIV2 output is the 2-byte datapath frequency and is used when RXDATA is 2 bytes. The RXRECCLKPMA_DIV1 output is the 1-byte datapath frequency and is used when RXDATA is 1 byte.

RXRECCLK output has an output delay control used for applications that bypass the RX buffer for constant datapath delay. Refer to RX Buffer Bypass, page 230 for more details.

Ports and Attributes

Table 4-24 defines the RX clock divider control block ports.

Table 4-24: **RX Clock Divider Control Ports**

Port	Dir	Clock Domain	Description
MGTREFCLKFAB[1]	Out	Async	This is a redundant output. RXRECCLK with RXRECCLK_CTRL = "RXPLLREFCLK_DIV1" should be used instead.
O ODIV2	Out	Async	The IBUFDS_GTXE1 primitive allows the MGTREFCLK to be output to the FPGA logic directly.
PHYSTATUS	Out	RXUSRCLK2	After RXRATE[1:0] is changed to initiate a rate change, PHYSTATUS goes Low and toggles for one RXUSRCLK2 cycle at the conclusion of the rate change as defined by TRANS_TIME_RATE. PHYSTATUS is intended for PCI Express protocol. Non PCI Express protocol should monitor RXRATEDONE.
RXRATE[1:0]	In	RXUSRCLK2	Controls the setting for the RX serial clock divider for low line rate support (see Table 4-23). This input port is used in combination with the RXPLL_DIVSEL_OUT attribute. `00`: Let RXPLL_DIVSEL_OUT determine the D divider value `01`: Set the D divider to 4 `10`: Set the D divider to 2 `11`: Set the D divider to 1
RXRATEDONE	Out	RXUSRCLK2	The RXRATEDONE port is asserted High for one RXUSRCLK2 cycle in response to a change on the RXRATE[1:0] port. The TRANS_TIME_RATE attribute determines the period of time between a change on the RXRATE[1:0] port and the assertion of RXRATEDONE.
RXRECCLK	Out	Async	This is the recommended clock output to the fabric. The attribute RXRECCLK_CTRL is the input selector for RXRECCLK and allows the RX PLL input reference clock or the recovered clocks to be output to fabric. The RXRECCLK output to the fabric can only be connected directly to the clock pin of the MMCM in the same clocking region or the input of the BUFR / BUFG.
RXRECCLKPCS	Out	Async	This is a redundant output. RXRECCLK with RXRECCLK_CTRL = "RXRECCLKPCS" should be used instead.

Table 4-25 defines the RX Clock Divider Control block attributes.

Table 4-25: **RX Clock Divider Control Attributes**

Attribute	Type	Description
RX_EN_RATE_RESET_BUF	Boolean	When set to TRUE, this attribute enables automatic RX buffer reset during a rate change event initiated by a change in RXRATE[1:0].
RXPLL_DIVSEL_OUT	Integer	This controls the setting for the RX serial clock divider for low line rate support (see Table 4-23). This attribute is only valid when RXRATE[1:0] = `00`. Otherwise the D divider value is controlled by RXRATE[1:0]. Valid settings are: 1: Set the D divider to 1 2: Set the D divider to 2 4: Set the D divider to 4

Table 4-25: **RX Clock Divider Control Attributes *(Cont'd)***

Attribute	Type	Description
RXRECCLK_CTRL	String	This attribute is the multiplexer select signal shown in Figure 4-15. It determines which GTX transceiver internal clock is output to fabric via RXRECCLK signal port. Valid settings are: "RXRECCLKPCS" "RXRECCLKPMA_DIV1" "RXRECCLKPMA_DIV2" "RXPLLREFCLK_DIV1" "RXPLLREFCLK_DIV2"
TRANS_TIME_RATE	8-bit Hex	Reserved. Use only recommended values from the Virtex-6 FPGA GTX Transceiver Wizard. This attribute determines when PHYSTATUS and RXRATEDONE are asserted after a rate change.

RX Margin Analysis

Functional Description

As line rates and channel attenuation increase, the receiver equalizers are often enabled to overcome channel attenuation. This posts a challenge to system debug as the quality of the link cannot be determined by measuring the far-end eye diagram. 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.

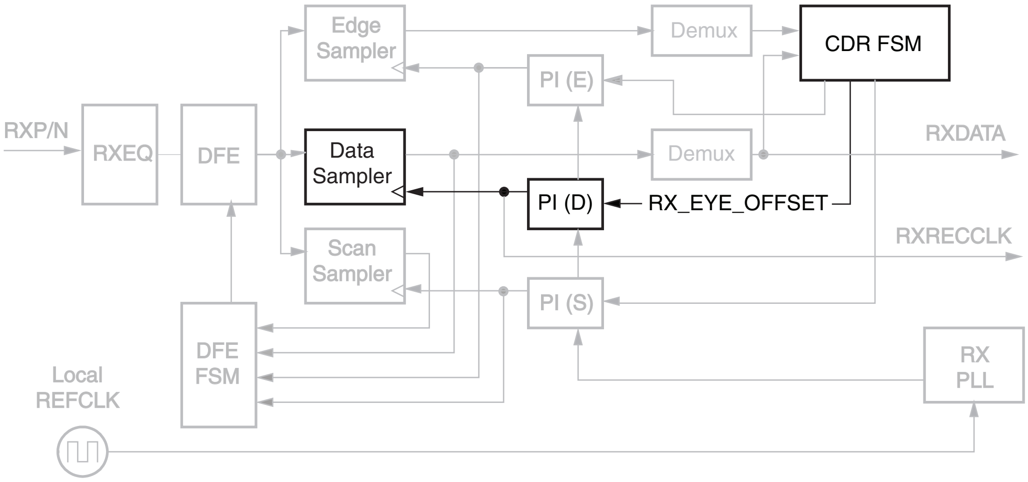
The RX CDR block provides a diagnostic mechanism to estimate the receiver eye margin after the equalizer when in the 1x divider setting (RXPLL_DIVSEL_OUT = 1).

Horizontal Eye Margin Scan

In the regular operational mode where RX_EYE_SCANMODE is set to `00`, the CDR decision making state machine centers the edge sampler to the data transition region. The phase of the data sampler has a constant 0.5 UI offset from the edge sampler to allow the data sampler to stay in the middle of the data eye.

In the horizontal eye margin scan mode where RX_EYE_SCANMODE is set to `10`, the CDR allows the user to control the phase offset between the data sampler and the edge sampler via the attribute RX_EYE_OFFSET. This is illustrated in Figure 4-16. As the CDR state machine operates normally, the scanned margin is the true receiver margin, incorporating both receiver equalizer and CDR effects. Refer to RX Equalizer, page 193 for more details on the DFE.

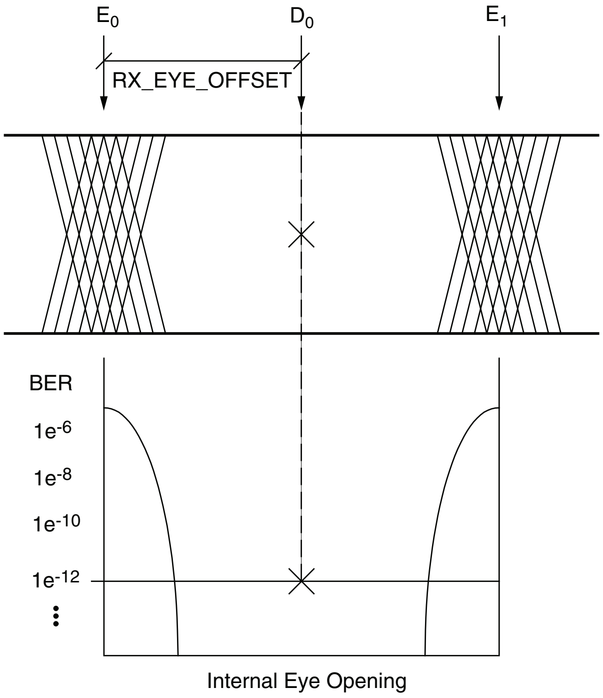
At the beginning of a horizontal eye margin scan, the CDR state machine should be frozen because measuring the bit error ratio (BER) requires sampling the equalized waveform far from the optimal points. This sampling could cause CDR errors, resulting in slipped cycles and inducing unrecoverable, high error rates. (The requirement to freeze the CDR carries the penalty that the input RX data must be synchronous to the RX REFCLK.) The CDR can be frozen by setting the lowest 11 bits of the PMA_RX_CFG attribute to zero (PMA_RX_CFG[10:0] = `11'b0`).



UG366_c4_15_051509

Figure 4-16: **Horizontal Eye Margin Scan Detail**

As illustrated in Figure 4-17, when the data sampling phase approaches the edge transition region, user logic observes a corresponding increased in bit error rate on the received user data.



UG366_c4_16_051509

Figure 4-17: **Data Sampling Position to Bit Error Rate**

This scan mode provides only the physical mechanism to offset the data sampling position. It does not provide the actual scanning and bit error rate monitoring functionalities. These functionalities need to be implemented in either FPGA user logic or user software. This mode is only recommended for diagnostic purposes because the received user data is corrupted due to the non-optimal sampling position.

Eye Outline Scan Mode

This method provides diagnostic information related to the deterministic eye shape, particularly the vertical eye opening. Setting RX_EYE_SCANMODE to `01` suspends the DFE adaptation (if enabled). In this mode, DFEEYEDACMON provides a value between 0 and 31 that is proportional to the eye height at the phase offset determined by RX_EYE_OFFSET.

This method measures the vertical opening corresponding to a high BER using the Scan Sampler (see Figure 4-16) so that no additional eye closure due to random jitter or noise is taken into account. However, this method does not corrupt received data that continues to be detected using the data sampler. Eye outline scan mode can thus be employed while data continues to be received without error.

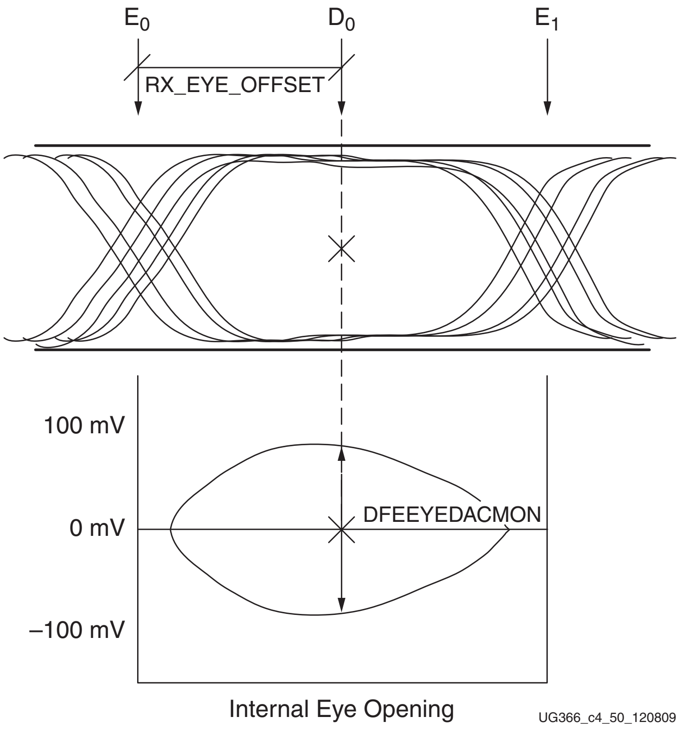


Figure 4-18: **Vertical Eye Height vs. RX_EYE_OFFSET**

Ports and Attributes

Table 4-26 defines the RX margin analysis ports.

Table 4-26: **RX Margin Analysis Ports**

Port	Dir	Clock Domain	Description
RXDATA[31:0]	Out	RXUSRCLK2	The user needs to detect data errors on RXDATA in order to monitor the bit error rate of the link.
DFEEYEDACMON[4:0]	Out	RXUSRCLK2	Average vertical eye height (voltage domain) used by the DFE as an optimization criterion. `11111`: Indicates approximately 200 mV_{PPD} of internal eye opening.

Table 4-27 defines the RX margin analysis attributes.

Table 4-27: **RX Margin Analysis Attributes**

Attribute	Type	Description
RX_EYE_OFFSET	8-bit Hex	When RX_EYE_SCANMODE = 10 or 01, RX_EYE_OFFSET determines the offset between the edge sampler, controlled by the CDR, and the data sampler. The valid range is 0 to 127 (0x00 to 0x7F) which corresponds to the 0 to 1.0 UI position of a serial data bit.
RX_EYE_SCANMODE	2-bit Binary	RX_EYE_SCANMODE determines if the receiver should work in normal operation of in the scan operation. 00: Regular data operation. RX_EYE_OFFSET is ignored. 01: Eye outline scan mode. 10: Horizontal eye margin scan mode. 11: Reserved.

RX Polarity Control

Functional Description

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

Ports and Attributes

Table 4-28 defines the RX polarity control ports.

Table 4-28: **RX Polarity Control Ports**

Port	Dir	Clock Domain	Description
RXPOLARITY	In	RXUSRCLK2	The RX polarity 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.

There are no RX polarity control attributes.

Using RX Polarity Control

If the polarity of RXP/RXN needs to be inverted, RXPOLARITY must be tied High.

RX Oversampling

Feature Description

Each GTX transceiver includes built-in 5X oversampling to enable serial rates from 1/10th of the lower border of the frequency range of the RX PMA PLL up to 4/10th of the PLL. At these low rates, the regular CDR must operate at five times the desired line rate to stay within its operating limits. The digital oversampling circuit takes parallel data from the SIPO at five times the desired line rate and uses the position of bit value transitions to recover a clock. The transition points are also used to pick an optimal sampling point to recover 4 bits of data from each set of 20 bits presented.

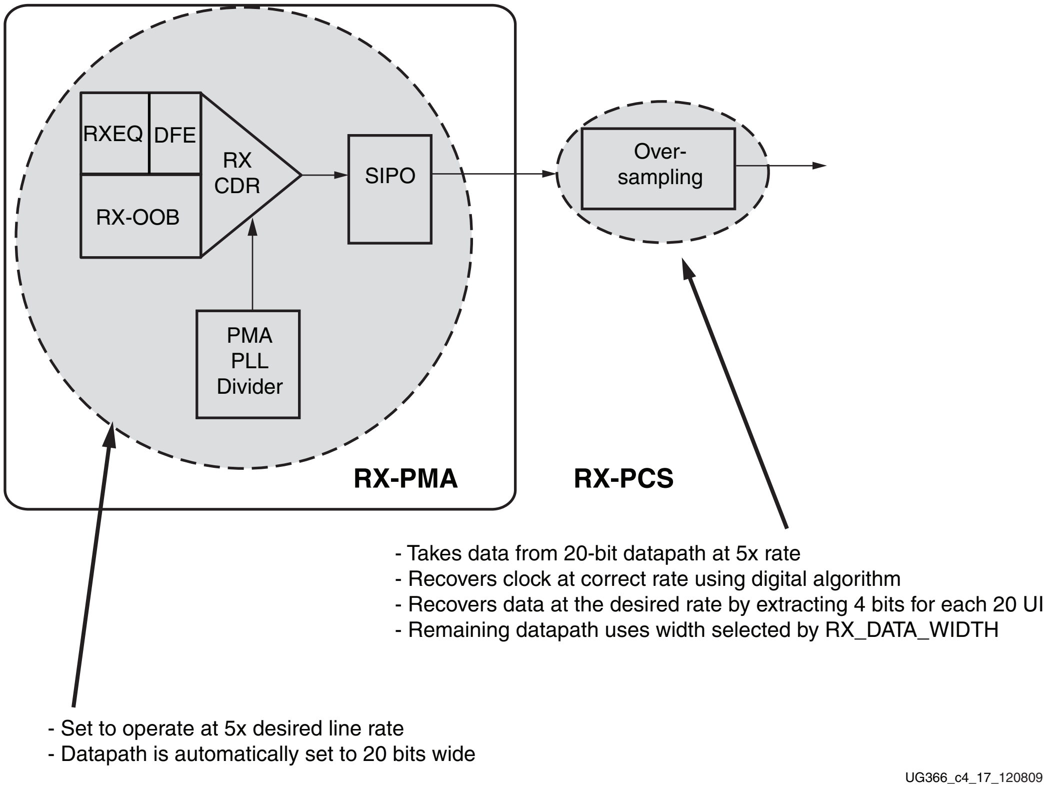

RXEQ
DFE
RX CDR
RX-OOB
SIPO
PMA PLL Divider
Over-sampling
RX-PMA
RX-PCS
- Takes data from 20-bit datapath at 5x rate
- Recovers clock at correct rate using digital algorithm
- Recovers data at the desired rate by extracting 4 bits for each 20 UI
- Remaining datapath uses width selected by RX_DATA_WIDTH
- Set to operate at 5x desired line rate
- Datapath is automatically set to 20 bits wide
UG366_c4_17_120809

Figure 4-19: **GTX Transceiver RX Block Diagram – Oversampling**

Note relevant to Figure 4-19:

1. For more details about RX_DATA_WIDTH, refer to Table 4-56, page 269.

Each GTX transceiver can be configured for oversampling independent of the other transceivers in a Quad. Configuring the GTX transceiver to use oversampling requires the following steps:

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

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

Ports and Attributes

Table 4-29 defines the RX oversampling ports.

Table 4-29: **RX Oversampling Ports**

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

Table 4-30 defines the RX oversampling attributes.

Table 4-30: **RX Oversampling Attributes**

Attribute	Type	Description
PMA_RX_CFG	25-bit Hex	This 25-bit attribute allows the operation of the CDR to be adjusted. In normal operation, set the attribute PMA_RX_CFG to the default value generated by the wizard. In oversampling mode, PMA_RX_CFG is set to `25'h0F44000`.
RX_OVERSAMPLE_MODE	Boolean	This attribute enables receiver oversampling when TRUE and when 5X oversampling is On.
RXPLL_DIVSEL_OUT	Integer	RXPLL_DIVSEL_OUT must be set to 1 when using oversampling mode.

RX Pattern Checker

Functional Description

The GTX transceiver receiver includes a built-in PRBS checker. 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.

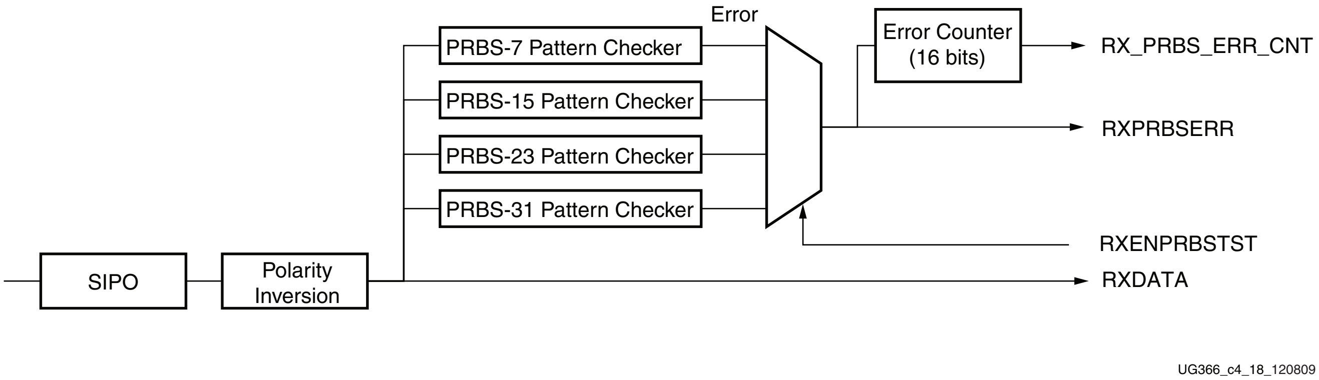


Figure 4-20: **RX Pattern Generator Block**

Ports and Attributes

Table 4-31 defines the pattern checker ports.

Table 4-31: **Pattern Checker Ports**

Port	Dir	Clock Domain	Description
PRBSCNTRESET	In	RXUSRCLK2	Reset PRBS error counter
RXENPRBSTST[2:0]	In	RXUSRCLK2	Receiver PRBS checker test pattern control. Only the following 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 as 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 4-32 defines the pattern checker attributes.

Table 4-32: **Pattern Checker Attributes**

Attribute	Type	Description
RXPRBSERR_LOOPBACK	1-bit Binary	This attribute can only be controlled via the DRP. The address location for this attribute is bit 8 of `0x2A`. When this attribute is set to `1`, the RXPRBSERR bit is internally looped back to TXPRBSFORCEERR of the same GTX 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.

Table 4-33 defines the RX pattern checker registers.

Table 4-33: **Pattern Checker Registers (Read Only)**

Attribute	Type	Description
RX_PRBS_ERR_CNT	16-bit Binary	PRBS error counter. This counter can be reset by asserting PRBSCNTRESET. When there is an error(s) in incoming parallel data, this counter increments by 1 and counts up to `0xFFFF`. This error counter can only be accessed via the DRP. The address for this counter is `0x82`.

Use Models

To use the built-in PRBS checker, set RXENPRBSTST to match the PRBS pattern being sent to the receiver. The RXENPRBSTST entry in Table 4-31 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 transmitter or reversed RXP/RXN, the received data should also inverted by controlling RXPOLARITY. Otherwise, the PRBS checker will 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 increment the word error counter by 1 when an error(s) is found in the incoming parallel data. This means that the word error counter may not match to the actual number of bit errors if the incoming parallel data contains two or more bit errors. The error counter stops counting when achieving `0xFFFF`.

When the error occurs, RXPRBSERR is asserted. When no error is found in the following incoming data, RXPRBSERR is cleared. Asserting PRBSCNTRESET clears the error counter. GTXRXRESET, RXCDRRESET, and RXRESET also reset the count.

Refer to TX Pattern Generator, page 163 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 4-21 shows the alignment to a 10-bit comma. The TX parallel data is on the left. The serial data with the comma is highlighted in the middle. The RX receiving unaligned bits are on the right side.

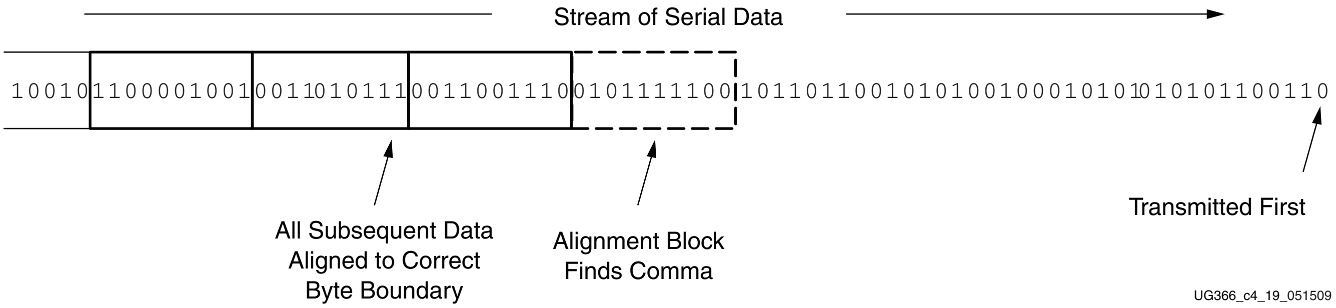


Figure 4-21: **Conceptual View of Alignment (Aligning to a 10-Bit Comma)**

Figure 4-22 shows the TX parallel data is on the left side, and the RX receiving recognizable parallel data is on the right side.

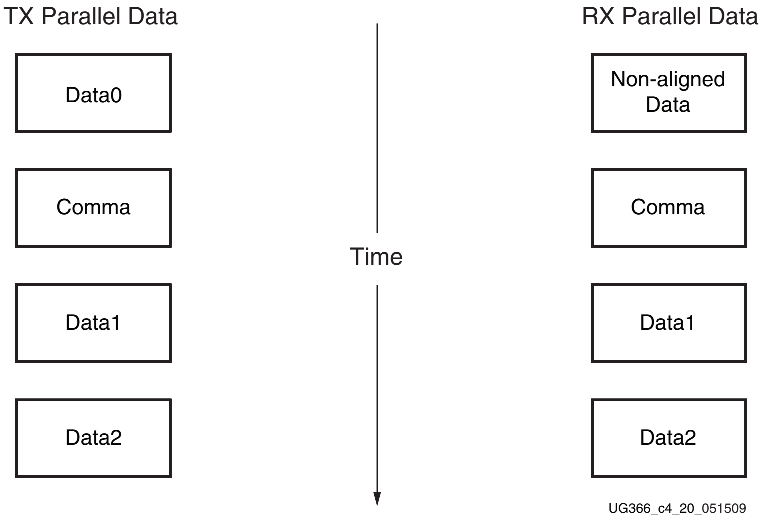


Figure 4-22: **Parallel Data View of Comma Alignment**

The GTX transceiver includes an alignment block that can be programmed to align specific commas to various byte boundaries, or to manually align data using attribute settings (see Figure 4-22) SONET A1/A2 alignment is possible using comma double mode. The block can be bypassed to reduce latency if it is not needed.

Enabling Comma Alignment

To enable the comma alignment block, the RXCOMMADETUSE port is driven High. RXCOMMADETUSE is driven Low to bypass the block completely for minimum latency.

Configuring Comma Patterns

To set the comma pattern that the block searches for in the incoming data stream, the MCOMMA_10B_VALUE, PCOMMA_10B_VALUE, and COMMA_10B_ENABLE attributes are used. The comma lengths depend on RX_DATA_WIDTH (see Table 4-56, page 269). Figure 4-23 shows how the COMMA_10B_ENABLE masks each of the comma values to allow partial pattern matching. Figure 4-23 shows how a COMMA is combined with COMMA_ENABLE to make a wildcarded comma for a 20-bit internal comma.

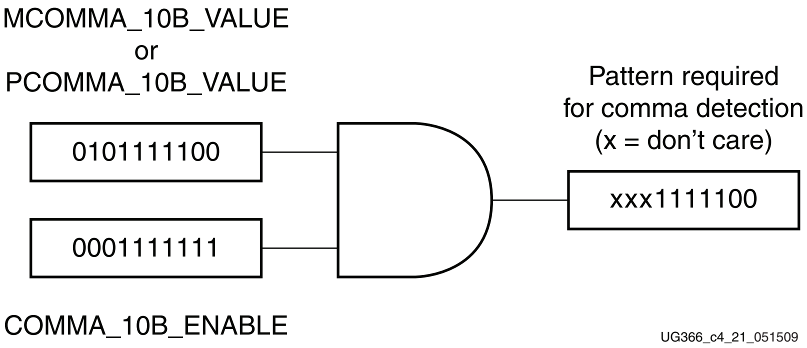


Figure 4-23: **Comma Pattern Masking**

If COMMA_DOUBLE is TRUE, the MCOMMA and PCOMMA patterns are combined so that the block searches for two commas in a row. The number of bits in the comma depends on RX_DATA_WIDTH. Either a 16-bit or a 20-bit comma alignment mode is possible. Figure 4-24 shows how the commas are combined when COMMA_DOUBLE is TRUE.

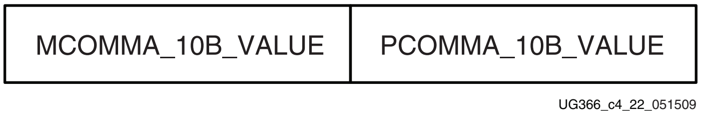


Figure 4-24: **Extended Comma Pattern Definition**

Figure 4-25 shows how COMMA_10B_ENABLE and wildcarding work for a double-width comma.

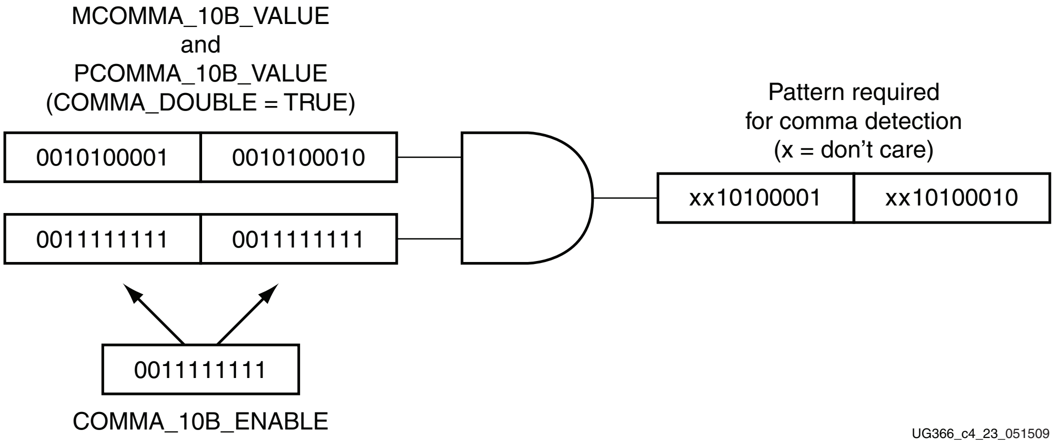


Figure 4-25: **Extended Comma Pattern Masking**

Activating Comma Alignment

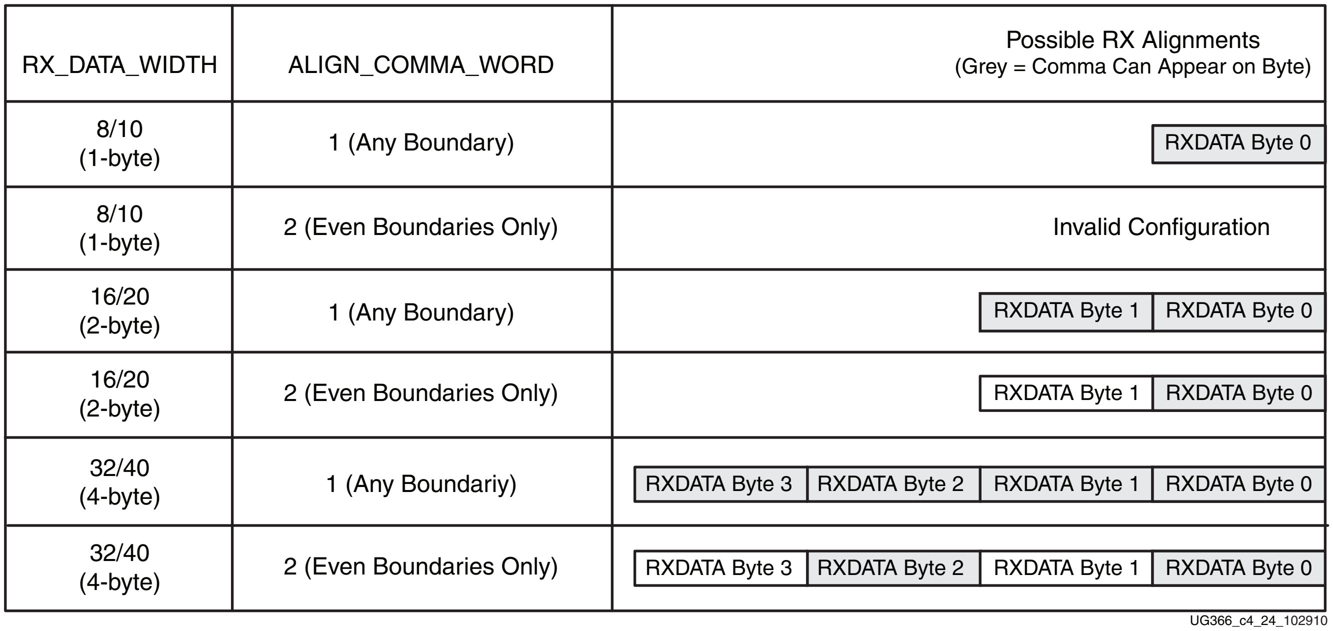
Commas are aligned to the closest boundary providing they are found while comma alignment is active. RXENMCOMMAALIGN is driven High to align on the MCOMMA pattern. RXENPCOMMAALIGN is driven High to activate alignment on the PCOMMA pattern. Both enable ports are driven to align to either pattern. When COMMA_DOUBLE is TRUE, both enable ports must always be driven to the same value.

Alignment Status Signals

While MCOMMA or PCOMMA alignment is active, any matching comma pattern causes the block to realign to the closest boundary. After successful alignment, the block holds RXBYTEISALIGNED High. At this time, RXENMCOMMAALIGN and RXENPCOMMAALIGN can be driven Low to turn off alignment and keep the current alignment position. PCOMMA_DETECT must be TRUE for PCOMMAs to cause RXBYTEISALIGNED to go High. Similarly, MCOMMA_DETECT must be TRUE for MCOMMAs to cause RXBYTEISALIGNED to go High. Commas can arrive while RXBYTEISALIGNED is High. If the commas arrive aligned to boundaries, there is no change. If the commas arrive out of position, the block deasserts RXBYTEISALIGNED until the commas are aligned again. If alignment is still activated for the comma that arrives, the block automatically aligns the new comma to the closest boundary and drives RXBYTEREALIGN High for one RXUSRCLK2 cycle.

Alignment Boundaries

The allowed boundaries for alignment are defined by ALIGN_COMMA_WORD. 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 4-56, page 269 for RX_DATA_WIDTH settings). Figure 4-21 shows the boundaries that can be selected.



RX_DATA_WIDTH	ALIGN_COMMA_WORD	Possible RX Alignments (Grey = Comma Can Appear on Byte)			
8/10 (1-byte)	1 (Any Boundary)				▨ RXDATA Byte 0
8/10 (1-byte)	2 (Even Boundaries Only)			Invalid Configuration	
16/20 (2-byte)	1 (Any Boundary)			▨ RXDATA Byte 1	▨ RXDATA Byte 0
16/20 (2-byte)	2 (Even Boundaries Only)			RXDATA Byte 1	▨ RXDATA Byte 0
32/40 (4-byte)	1 (Any Boundariy)	▨ RXDATA Byte 3	▨ RXDATA Byte 2	▨ RXDATA Byte 1	▨ RXDATA Byte 0
32/40 (4-byte)	2 (Even Boundaries Only)	RXDATA Byte 3	▨ RXDATA Byte 2	RXDATA Byte 1	▨ RXDATA Byte 0

Figure 4-26: **Comma Alignment Boundaries**

Manual Alignment

RXSLIDE can be used to override the automatic comma alignment and 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 16 RXUSRCLK2 cycles before it can be used again. Slide results on RXDATA will appear after several cycles of latency through the PCS. The actual latency depends on the active blocks in the PCS path and maximum latency can be up to 64 cycles. Figure 4-27 shows the waveforms for manual alignment using RXSLIDE in RX_SLIDE_MODE = PCS before and after the data shift.

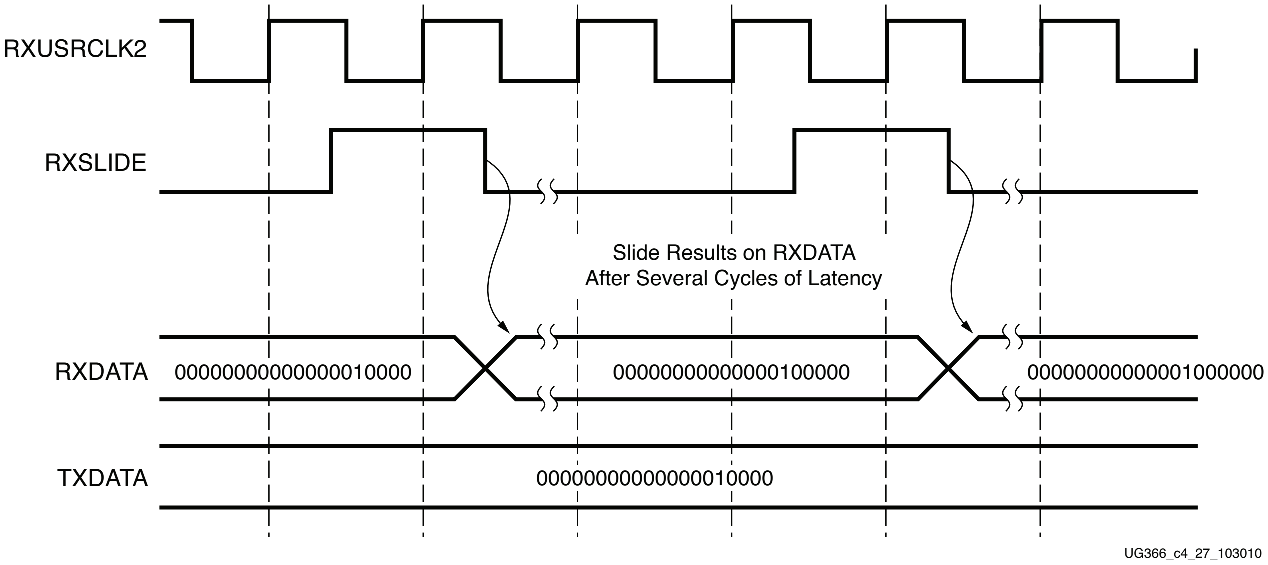


Figure 4-27: **Manual Data Alignment Using RXSLIDE for RX_DATA_WIDTH = 20 Bits**

Ports and Attributes

Table 4-34 defines the RX comma alignment and detection ports.

Table 4-34: **RX Comma Alignment and Detection Ports**

Port	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 PCOMMA_DETECT is TRUE. RXBYTEISALIGNED responds to minus comma alignment when MCOMMA_DETECT is TRUE.
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 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
RXCOMMADETUSE	In	RXUSRCLK2	RXCOMMADETUSE activates the comma detection and alignment circuit. 0: Bypass the circuit 1: Use the comma detection and alignment circuit Bypassing the comma and alignment circuit reduces RX datapath latency.
RXENMCOMMAALIGN	In	RXUSRCLK2	Aligns the byte boundary when *comma minus* is detected. 0: Disabled 1: Enabled
RXENPCOMMAALIGN	In	RXUSRCLK2	Aligns the byte boundary when *comma plus* is detected. 0: Disabled 1: Enabled

Table 4-34: **RX Comma Alignment and Detection Ports *(Cont'd)***

Port	Dir	Clock Domain	Description

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 16 RXUSRCLK2 cycles before it can be reasserted to cause another adjustment. When asserted, RXSLIDE takes precedence over normal comma alignment. For proper operation, the following settings should be made:
			RXENPCOMMALIGN = 0
			RXENMCOMMALIGN = 0
			RXCOMMADETUSE = 1

Table 4-35 defines the RX comma alignment attributes.

Table 4-35: **RX Comma 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 byte within a two-byte or four-byte datapath. The comma can be aligned to either the even byte [9:0] or the odd byte [19:10] for a two-byte RX interface, or the even bytes [9:0]/[29:20] or odd bytes [19:10]/[39:30] for a four-byte RX interface. 2: Align comma to the even bytes within the datapath. The aligned comma is guaranteed to be aligned to byte RXDATA[9:0] in a two-byte datapath, or bytes [9:0]/[29:20] in a four-byte datapath. For ALIGN_COMMA_WORD = 2 to work properly in conjunction with the RX elastic buffer, both CLK_COR_ADJ_LEN and CLK_COR_MIN_LAT must be even. Protocols that send commas in even and odd positions must set ALIGN_COMMA_WORD to 1. If RX_DATA_WIDTH is set to 8 or 10, ALIGN_COMMA_WORD must be set to 1.
COMMA_10B_ENABLE	10-bit Binary	Sets which bits of MCOMMA/PCOMMA must be matched to incoming data and which bits can be any value. This attribute is a 10-bit mask with a default value of `1111111111`. Any bit in the mask that is reset to 0 effectively turns the corresponding bit in MCOMMA or PCOMMA to a don't care bit.

Table 4-35: **RX Comma Alignment Attributes *(Cont'd)***

Attribute	Type	Description
COMMA_DOUBLE	Boolean	Specifies whether a comma match consists of either a comma plus or a comma minus alone, or whether both are required in the sequence. FALSE: The plus comma (PCOMMA) and minus comma (MCOMMA) are handled separately. An individual match for either can lead to comma detection and alignment. TRUE: A comma match consists of a comma plus followed immediately by a comma minus. The match pattern is 20 or 16 bits (as determined by RX_DATA_WIDTH). When COMMA_DOUBLE is TRUE, PCOMMA_DETECT must be the same as MCOMMA_DETECT, and RXENPCOMMAALIGN must be the same as RXENMCOMMAALIGN.
MCOMMA_10B_VALUE	10-bit Binary	Defines comma minus to raise RXCOMMADET and align the parallel data. Reception order is right to left. (MCOMMA_10B_VALUE [0] is received first.) The default value is `10'b1010000011` (K28.5). This definition does not affect 8B/10B encoding or decoding.
MCOMMA_DETECT	Boolean	Controls the raising of RXCOMMADET on comma minus. FALSE: Do not raise RXCOMMADET when comma minus is detected. TRUE: Raise RXCOMMADET when comma minus is detected. (This setting does not affect comma alignment.)
PCOMMA_10B_VALUE	10-bit Binary	Defines comma plus to raise RXCOMMADET and align parallel data. Reception order is right to left. (PCOMMA_10B_VALUE [0] is received first.) The default value is `10'b0101111100` (K28.5). This definition does not affect 8B/10B encoding or decoding.
PCOMMA_DETECT	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.)
SHOW_REALIGN_COMMA	Boolean	Defines whether a comma that causes realignment is brought out to the FPGA RX. FALSE: Do not bring the realignment comma to the FPGA RX. This setting reduces the RX datapath latency. TRUE: Bring the realignment comma to the FPGA RX. SHOW_REALIGN_COMMA = TRUE should not be used when COMMA_DOUBLE = TRUE.

Table 4-35: **RX Comma Alignment Attributes *(Cont'd)***

Attribute	Type	Description
RX_SLIDE_MODE	String	Defines the RXSLIDE mode: OFF: This is the default setting. The RXSLIDE feature is not used. PCS: The 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. In this mode, even if the RXRECCLK is sourcing from the RX PMA, the clock phase remains the same. PMA: The 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 RXRECCLK 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_REALIGN_COMMA to be FALSE.
RX_SLIDE_AUTO_WAIT	Integer	Defines how long the PCS waits for the PMA to auto slide (in terms of RXUSRCLK clock cycles) before checking the alignment again. Valid settings are from 0 to 15. The default value is 5. Only recommended values from the Virtex-6 FPGA GTX Transceiver Wizard should be used.

RX Loss-of-Sync State Machine

Functional Description

Several 8B/10B protocols make use of a standard Loss-of-Sync (LOS) state machine to detect when the channel is malfunctioning. Each GTX transceiver receiver includes a LOS state machine that can be activated for protocols requiring it. When the state machine is not used, the LOS state machine's ports can be re-used to monitor the condition of incoming data.

Figure 4-28 shows the standard LOS state machine, used in several 8B/10B protocols (for example, XAUI) to detect problems in the incoming data stream.

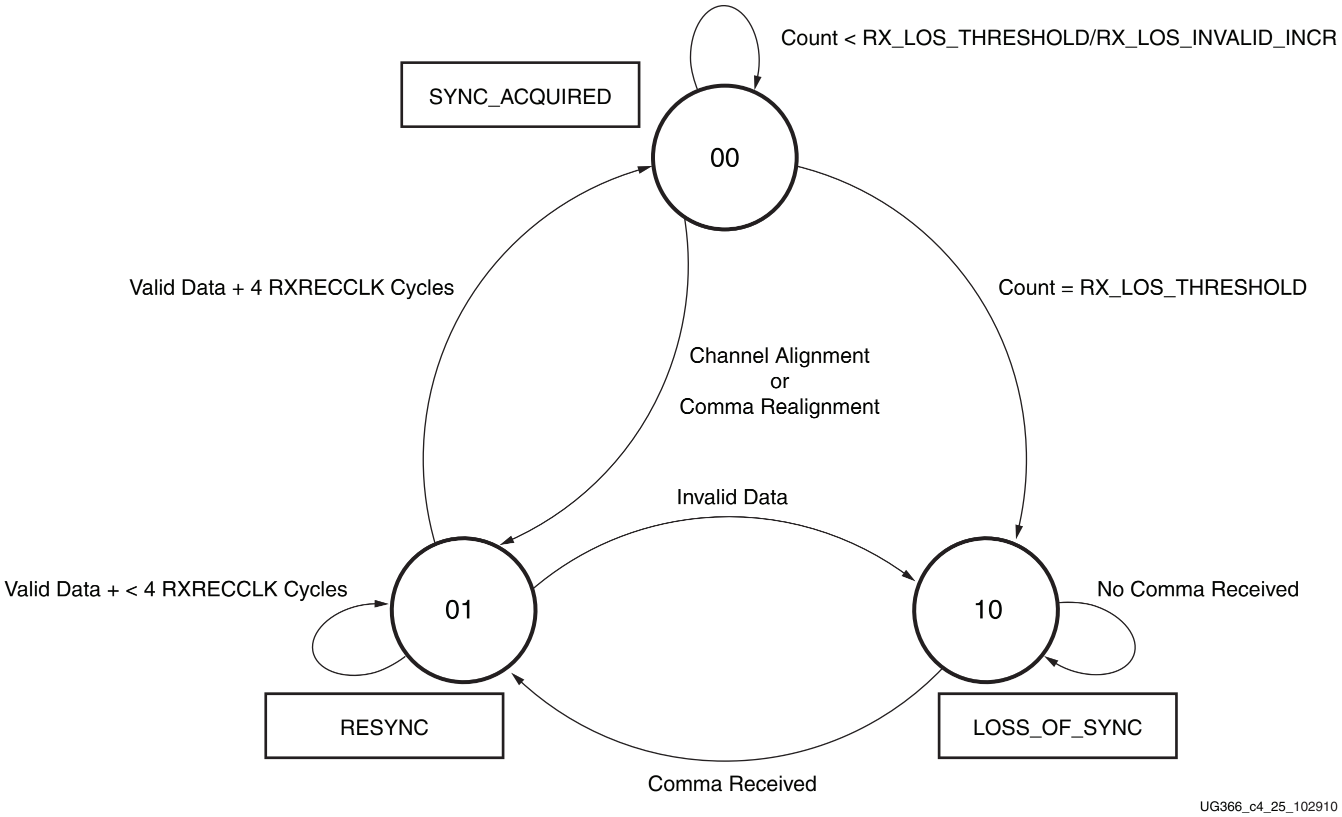


Figure 4-28: **LOS State Machine**

To activate the LOS state machine in the GTX transceiver, RX_LOSS_OF_SYNC_FSM is set to TRUE. While the state machine is active, the RXLOSSOFSYNC port presents its current state.

If the LOS state machine is inactive (RX_LOSS_OF_SYNC_FSM = FALSE), the RXLOSSOFSYNC port presents information about the received data. The RXLOSSOFSYNC entry in Table 4-36 shows the meaning of the RXLOSSOFSYNC port in this case.

The operation of the LOS state machine can be tuned using the RX_LOS_INVALID_INCR and RX_LOS_THRESHOLD attributes. RX_LOS_THRESHOLD adjusts how sensitive the LOS state machine is to bad characters (RX_LOS_THRESHOLD divided by RX_LOS_INVALID_INCR) to change the machine from the SYNC_ACQUIRED state to the

LOSS_OF_SYNC state. The RX_LOS_THRESHOLD entry in Table 4-37 shows the valid settings for this attribute.

The LOS state machine allows the error count in the SYNC_ACQUIRED state to decrease over time, so that sparse errors are eventually discarded. The rate that the error count is decreased is controlled by the RX_LOS_THRESHOLD and RX_LOS_INVALID_INCR attributes, as defined in Table 4-37.

Ports and Attributes

Table 4-36 defines the RX Loss-of-Sync State Machine ports.

Table 4-36: **RX Loss-Of-Sync State Machine Ports**

Port	Dir	Clock Domain	Description
RXLOSSOFSYNC[1:0]	Out	RXUSRCLK2	FPGA status related to byte stream synchronization. The meaning depends on the state of the RX_LOSS_OF_SYNC_FSM attribute. If RX_LOSS_OF_SYNC_FSM = TRUE, this output reflects the state of an internal Loss-of-Sync FSM as follows: [1] = 1: Sync lost due to either sequence of invalid characters or reset [0] = 1: In the resync state due to a channel bonding sequence or realignment If RX_LOSS_OF_SYNC_FSM = FALSE, this output presents the following information about incoming data: [1] = 1: Received data is not an 8B/10B character or has a disparity error [0] = 1: Channel bonding sequence detected in data

Table 4-37 defines the RX Loss-of-Sync State Machine attributes.

Table 4-37: **RX Loss-Of-Sync State Machine Attributes**

Attribute	Type	Description
RX_LOS_INVALID_INCR	Integer	Defines the number of valid characters required to *cancel* out the appearance of one invalid character for the purpose of loss-of-sync determination. Valid settings are 1, 2, 4, 8, 16, 32, 64, and 128.
RX_LOS_THRESHOLD	Integer	When divided by RX_LOS_INVALID_INCR, defines the number of invalid characters required to cause an FSM transition to the sync lost state. Valid settings are 4, 8, 16, 32, 64, 128, 256, and 512.
RX_LOSS_OF_SYNC_FSM	Integer	RX_LOSS_OF_SYNC_FSM defines the behavior of the RXLOSSOFSYNC[1:0] outputs. FALSE (default): RXLOSSOFSYNC[1] goes High when invalid data (not in table or disparity error) is found in 8B/10B decoding. RXLOSSOFSYNC[0] goes High when a channel bonding sequence has been written into the RX elastic buffer. TRUE: Loss of sync FSM is in operation and its state is reflected on RXLOSSOFSYNC[1].

RX 8B/10B Decoder

Functional Description

Many protocols require receivers to decode 8B/10B data. 8B/10B is an industry standard encoding scheme that trades two bits of overhead per byte for improved performance.

The GTX transceiver includes an 8B/10B decoder to decode RX data without consuming FPGA resources. The decoder includes status signals to indicate errors and incoming control sequences. If decoding is not needed, the block can be disabled to minimize latency.

8B/10B Decoder Bit and Byte Order

8B/10B requires bit a0 to be received first, but the GTX transceiver always receives the right-most bit first. Consequently, the 8B/10B decoder is designed to automatically reverse the bit order of received data before decoding it. Similarly, because the GTX transceiver receives the right-most bit first, when a 2-byte interface is used, the first byte received (byte 0) is presented on RXDATA[7:0], and the second byte is presented on RXDATA[15:8]. When a 4-byte interface is used, the first received byte is presented on RXDATA[7:0], and the fourth byte is presented on RXDATA[31:24]. Figure 4-29 shows how the decoder maps 10-bit data to 8-bit values.

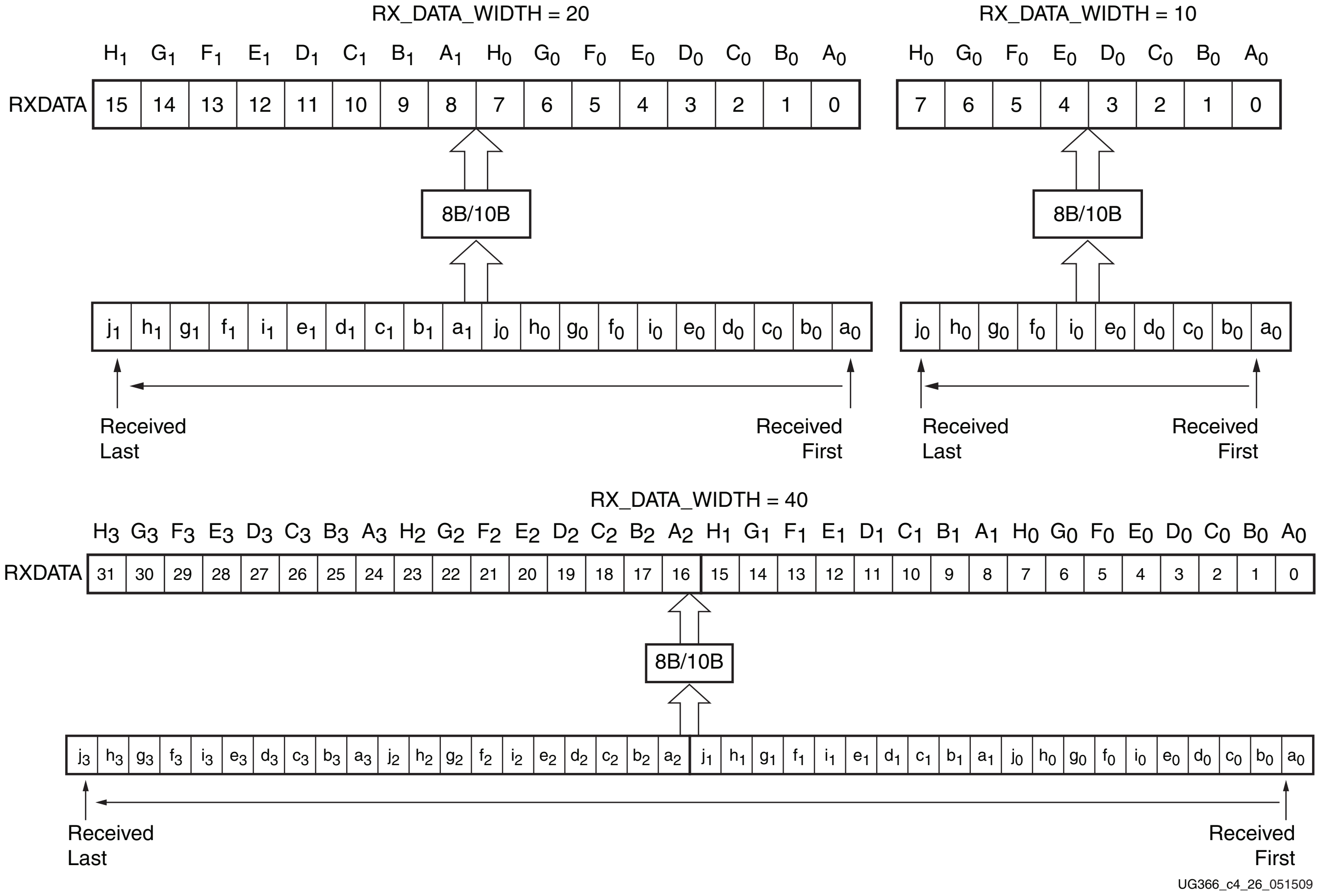

RX_DATA_WIDTH = 20
H1 G1 F1 E1 D1 C1 B1 A1 H0 G0 F0 E0 D0 C0 B0 A0
RXDATA
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
8B/10B
j1 h1 g1 f1 i1 e1 d1 c1 b1 a1 j0 h0 g0 f0 i0 e0 d0 c0 b0 a0
Received Last
Received First
RX_DATA_WIDTH = 10
H0 G0 F0 E0 D0 C0 B0 A0
7 6 5 4 3 2 1 0
8B/10B
j0 h0 g0 f0 i0 e0 d0 c0 b0 a0
Received Last
Received First
RX_DATA_WIDTH = 40
H3 G3 F3 E3 D3 C3 B3 A3 H2 G2 F2 E2 D2 C2 B2 A2 H1 G1 F1 E1 D1 C1 B1 A1 H0 G0 F0 E0 D0 C0 B0 A0
RXDATA
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
8B/10B
j3 h3 g3 f3 i3 e3 d3 c3 b3 a3 j2 h2 g2 f2 i2 e2 d2 c2 b2 a2 j1 h1 g1 f1 i1 e1 d1 c1 b1 a1 j0 h0 g0 f0 i0 e0 d0 c0 b0 a0
Received Last
Received First
UG366_c4_26_051509

Figure 4-29: **RX Interface with 8B/10B Decoding**

RX Running Disparity

8B/10B includes special characters (K characters) that are often used for control functions. When RXDATA is a K character, the decoder drives RXCHARISK High.

If DEC_PCOMMA_DETECT is TRUE, the decoder drives RXCHARISCOMMA High whenever RXDATA is a positive 8B/10B comma. Similarly, if DEC_MCOMMA_DETECT is TRUE, the decoder drives RXCHARISCOMMA High whenever RXDATA is a negative 8B/10B comma.

The decoder drives RXDISPERR High when RXDATA arrives with the wrong disparity. In addition to disparity errors, the 8B/10B decoder detects 20-bit out-of-table error codes. The decoder drives the RXNOTINTABLE port High when RXDATA is not a valid 8B/10B character. In this case, the aligned and not decoded data is present on RXDATA, RXCHARISK, and RXDISPERR. This behavior is identical to disabling the decoder for bypassing as shown in Table 4-57.

Figure 4-30 shows a waveform with a few error bytes arriving on RXDATA and the RXNOTINTABLE and RXDISPERR ports indicating the error.

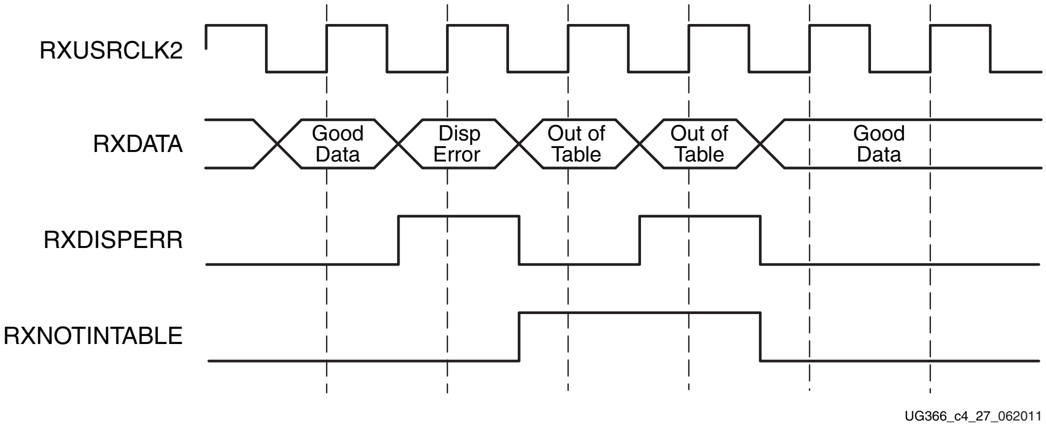


Figure 4-30: **RX Data with 8B/10B Errors**

Note relevant to Figure 4-30:

1. When RXNOTINTABLE is flagged, RXDISPERR is used to output the most significant bit of non-decoded data.

Ports and Attributes

Table 4-38 defines the RX decoder ports.

Table 4-38: **RX Decoder Ports**

Port	Dir	Clock Domain	Description
RXCHARISCOMMA[3:0]	Out	RXUSRCLK2	RXCHARISCOMMA indicates that the corresponding byte of RXDATA is a comma character. 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[3:0]	Out	RXUSRCLK2	RXCHARISK indicates that the corresponding byte of RXDATA is a K character when 8B/10B is used. These pins are reused as bit 8 of each 10-bit word when 8B/10B is bypassed and the transceiver data width is a multiple of 10. 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]
RXDEC8B10BUSE	In	RXUSRCLK2	RXDEC8B10BUSE 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)
RXDISPERR[3:0]	Out	RXUSRCLK2	When High, RXDISPERR indicates that the corresponding byte of RXDATA has a disparity error. These pins are reused as bit 9 of each 10-bit word when 8B/10B is bypassed and the transceiver data width is a multiple of 10. 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]
RXNOTINTABLE[3:0]	Out	RXUSRCLK2	RXNOTINTABLE indicates that the corresponding byte of RXDATA was decoded from a 10-bit value that was not a valid character in the 8B/10B table. 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]
RXRUNDISP[3:0]	Out	RXUSRCLK2	RXRUNDISP shows the running disparity of the 8B/10B encoder when RXDATA is received. RXRUNDISP[3] corresponds to RXDATA[31:24] RXRUNDISP[2] corresponds to RXDATA[23:16] RXRUNDISP[1] corresponds to RXDATA[15:8] RXRUNDISP[0] corresponds to RXDATA[7:0]

Table 4-39 defines the RX decoder attributes.

Table 4-39: **RX Decoder Attributes**

Attributes	Type	Description
DEC_MCOMMA_DETECT	Boolean	Enables detection of negative 8B/10B commas: TRUE: RXCHARISCOMMA is asserted when RXDATA is a negative 8B/10B comma FALSE: RXCHARISCOMMA does not respond to negative 8B/10B commas
DEC_PCOMMA_DETECT	Boolean	Enables detection of positive 8B/10B commas: TRUE: RXCHARISCOMMA is asserted when RXDATA is a positive 8B/10B comma FALSE: RXCHARISCOMMA does not respond to positive 8B/10B commas
DEC_VALID_COMMA_ONLY	Boolean	Limits the set of commas to which RXCHARISCOMMA responds: TRUE: RXCHARISCOMMA is asserted only for K28.1, K28.5, and K28.7 (see 8B/10B K character table in Appendix A) FALSE: RXCHARISCOMMA responds to any positive or negative 8B/10B comma, depending on the settings for DEC_MCOMMA_DETECT and DEC_PCOMMA_DETECT
RX_DATA_WIDTH	Integer	Selects PCS data width mode between the values of 8, 16, 32, 10, 20, and 40. If 8B/10B is used, this attribute must be set to a multiple of 10 {10, 20, 40} even though only a multiple of 8 {8, 16, 32} RXDATA bits come out to the FPGA logic. RXUSRCLK and RXUSRCLK2 clock frequencies also need to be set correctly.
RX_DECODE_SEQ_MATCH	Boolean	Selects whether a datastream is coming out of the Comma Detection and Realign block (FALSE) or the 8B/10B decoder output (provided that RXDEC8B10BUSE is also High) should be used for channel bonding and clock correction sequences (TRUE).

RX Buffer Bypass

Functional Description

Bypassing the RX buffer is an advanced feature of the Virtex-6 FPGA GTX transceivers. RX buffer bypass can operate only under certain system-level conditions. 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 buffer is bypassed. Figure 4-34, page 237 shows the XCLK and USRCLK domains. Table 4-42, page 237 shows the trade-offs between the buffer and the buffer bypass modes.

The RX elastic buffer can be bypassed to reduce latency when RXRECCLK is used to source RXUSRCLK and RXUSRCLK2. When the RX elastic buffer is bypassed, latency through the RX datapath is low and deterministic, but clock correction and channel bonding are not available.

Figure 4-31 shows how phase alignment allows the RX elastic buffer to be bypassed. Before phase alignment, there is no guaranteed phase relationship between the parallel clock generated from the recovered clock in the CDR circuit (XCLK) and the parallel clocks from the FPGA logic (RXUSRCLK and RXUSRCLK2). Phase alignment causes RXRECCLK from the CDR to be adjusted so that there is no significant phase difference between XCLK and RXUSRCLK.

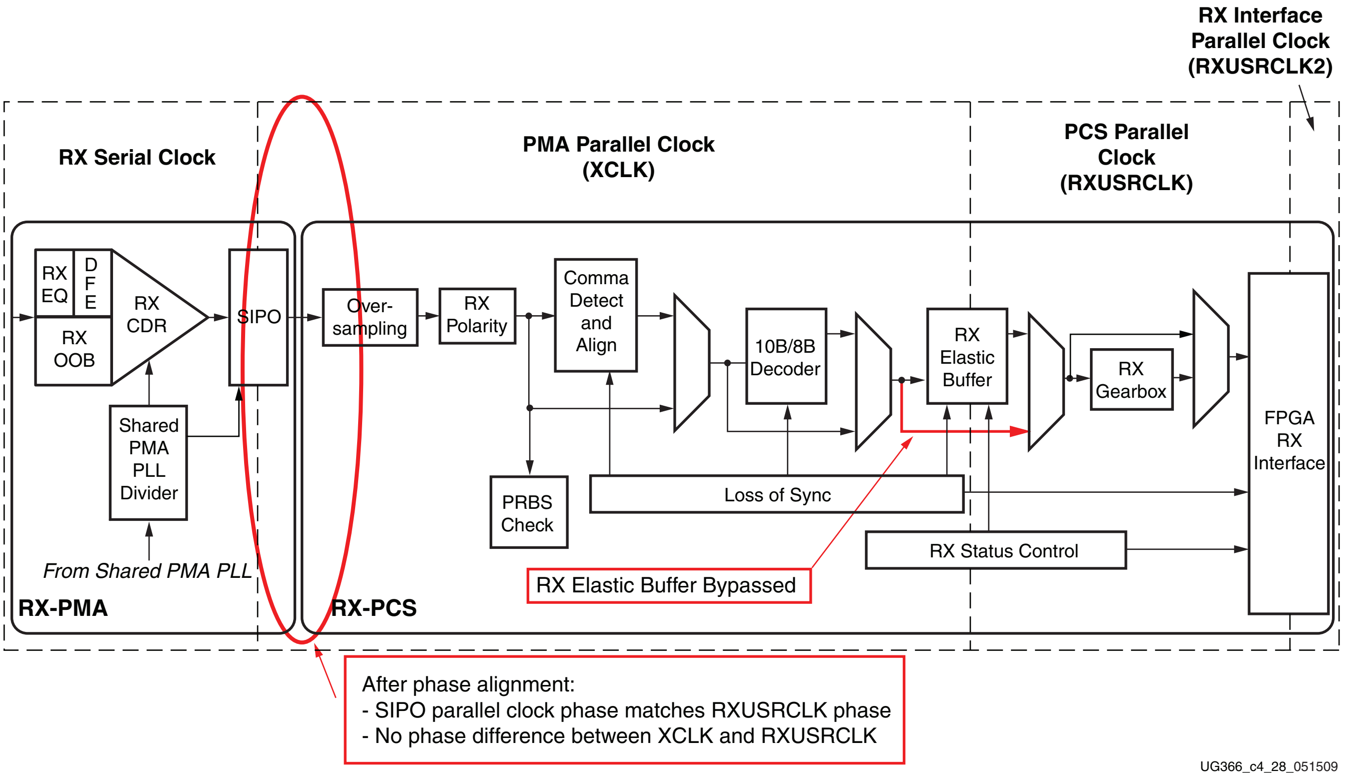


Figure 4-31: **Using Phase Alignment**

Note: Bypassing the RX buffer is an advanced feature. RX buffer bypass can operate only under certain system-level conditions.

To ensure that the RXRECCLK output port operates at the desired frequency in RX buffer bypass mode, all of these conditions must be met:

- Receiver reference clock must always be toggling
- RXPOWERDOWN[0] and RXPOWERDOWN[1] must be tied Low
- RXPLLPOWERDOWN must be tied Low
- GTXRXRESET and PLLRXRESET must not be tied High

For transceivers that are not instantiated in the user design, the ISE software, version 12.1 or later, automatically ensures that the RXRECCLK performance is preserved for future use. MGTAVCC must be supplied to these transceivers. Refer to Managing Unused GTX Transceivers, page 276 for more information.

Ports and Attributes

Table 4-40 defines the RX buffer bypass ports.

Table 4-40: **RX Buffer Bypass Ports**

Port	Dir	Clock Domain	Description
RXDLYALIGNDISABLE	In	Async	RX delay aligner enable/disable signal. When driven Low, it enables the delay phase aligner.
RXDLYALIGNMONENB	In	Async	RX delay aligner monitor enable/disable. When driven Low, this port enables the RX delay aligner monitor.
RXDLYALIGNMONITOR[7:0]	Out	Async	Reserved.
RXDLYALIGNOVERRIDE	In	Async	RX delay aligner override. This port must always be tied High. The delay value is set manually from the RX_DLYALIGN_OVRDSETTING[7:0] attribute.
RXDLYALIGNRESET	In	Async	Resets the RX delay aligner.
RXDLYALIGNSWPPRECURB	In	Async	Reserved. It must be tied High.
RXDLYALIGNUPDSW	In	Async	Reserved. It must be tied Low.
RXENPMAPHASEALIGN	In	Async	When activated, the GTX transceiver receiver can align its XCLK with its RXUSRCLK.
RXPLLLKDET	Out	Async	Indicates that the VCO rate is within acceptable tolerances of the desired rate when High. The GTX transceiver does not operate reliably until this condition is met.
RXPLLLKDETEN	In	Async	Enables the RX PLL lock detector. It must be tied High.
RXPMASETPHASE	In	Async	When activated this pin aligns the PMA receiver recovered clock with the PCS RXUSRCLK, allowing the RX elastic buffer to be bypassed.
RXRECCLK	Out	N/A	This is the recommended clock output to the fabric. The attribute RXRECCLK_CTRL is the input selector for RXRECCLK and allows the RX PLL input reference clock or the recovered clocks to be output to fabric.
RXUSRCLK	In	N/A	Use this port to provide a clock for the internal RX PCS datapath. The rate for RXUSRCLK depends on RX_DATA_WIDTH.

Table 4-41 defines the RX buffer bypass attributes.

Table 4-41: **RX Buffer Bypass Attributes**

Attribute	Type	Description
POWER_SAVE	10-bit Binary	POWER_SAVE[4]: Mux select for the TXOUTCLK output clock. Must be tied to 1'b1. 1'b0: Use the TX Delay Aligner 1'b1: Bypass the TX Delay Aligner POWER_SAVE[5]: Mux select for the RXRECCLK output clock. Must be tied to 1'b1 when RX buffer is used (RX_BUFFER_USE = TRUE). When RX buffer is bypassed, refer to Using the RX Phase Alignment Circuit to Bypass the Buffer, page 234. 1'b0: Use the RX Delay Aligner 1'b1: Bypass the RX Delay Aligner All other bits are reserved. Use recommended values from the Virtex-6 FPGA GTX Transceiver Wizard.
RX_BUFFER_USE	Boolean	Use or bypass the RX elastic buffer. TRUE: Use the RX elastic buffer (normal mode). FALSE: Permanently bypass the RX elastic buffer (advanced feature).
RX_DATA_WIDTH	Integer	Sets the receiver external data width. 8/10: 1 byte interface 16/20: 2 byte interface 32/40: 4 byte interface If 8B10B is used, this attribute must be a multiple of 10.
RX_DLYALIGN_CTRINC	4-bit Binary	Reserved. Use only recommended values from the Virtex-6 FPGA GTX Transceiver Wizard.
RX_DLYALIGN_EDGESET	5-bit Binary	Reserved. Use only recommended values from the Virtex-6 FPGA GTX Transceiver Wizard.
RX_DLYALIGN_LPFINC	4-bit Binary	Reserved. Use only recommended values from the Virtex-6 FPGA GTX Transceiver Wizard.
RX_DLYALIGN_MONSEL	3-bit Binary	Reserved. Use only recommended values from the Virtex-6 FPGA GTX Transceiver Wizard.
RX_DLYALIGN_OVRDSETTING	8-bit Binary	Sets the overdrive value for the RX delay aligner. This attribute takes effect when RXDLYALIGNOVERRIDE is driven High. This attribute must be set to `8'b10000000`.
RX_XCLK_SEL	String	Selects the clock used to drive the RX parallel clock domain (XCLK). "RXREC": (default) XCLK domain driven by recovered clock from CDR. When OVERSAMPLE_MODE is TRUE, the recovered clock is sourced from the oversampling block. "RXUSR": RXUSRCLK port drives RX parallel clock domain. Use this mode when bypassing the RX elastic buffer.

Table 4-41: **RX Buffer Bypass Attributes *(Cont'd)***

Attribute	Type	Description
RXRECCLK_CTRL	String	This attribute is the multiplexer select signal from the RX Clock Divider Control block (see Figure 4-15, page 206). RXRECCLKPCS (DRP value 000) RXRECCLKPMA_DIV1 (DRP value 001) RXRECCLKPMA_DIV2 (DRP value 010) RXPLLREFCLK_DIV1 (DRP value 011) RXPLLREFCLK_DIV2 (DRP value 100) OFF_LOW (DRP value 101) OFF_HIGH (DRP value 110)
RXUSRCLK_DLY	16-bit Hex	Reserved. Use only recommended values from the Virtex-6 FPGA GTX Transceiver Wizard.
PMA_RXSYNC_CFG	7-bit Hex	Reserved. Use only recommended values from the Virtex-6 FPGA GTX Transceiver Wizard.

Using the RX Phase Alignment Circuit to Bypass the Buffer

Bypassing the RX buffer is an advanced feature. RX buffer bypass can operate only under certain system-level conditions. To use the RX phase-alignment circuit, follow these steps:

1. Set the following attributes with their values as follows:
   a. Set RXRECCLK_CTRL:
      - 2 byte or 4 byte – use RXRECCLKPMA_DIV2
      - 1 byte – use RXRECCLKPMA_DIV1

   b. Set RX_BUFFER_USE to FALSE to bypass the RX elastic buffer.

   c. Set RX_XCLK_SEL to RXUSR.

   d. Set RX_DLYALIGN_OVRDSETTING to `8'b10000000`.
2. Make sure all the input ports RXENPMAPHASEALIGN and RXPMASETPHASE are driven Low.
3. Make sure that the ports RXDLYALIGNOVERRIDE and RXDLYALIGNDISABLE are driven High.
4. Reset the RX datapath using GTXRXRESET or the RXCDRRESET.
5. If an MMCM is used to generate RXUSRCLK/RXUSRCLK2 clocks, wait for the MMCM to lock.
6. Wait for the CDR to lock and provide a stable RXRECCLK.
7. Assert RXDLYALIGNRESET for 20 RXUSRCLK2 clock cycles.
8. Drive RXENPMAPHASEALIGN High. Keep RXENPMAPHASEALIGN High unless the phase-alignment procedure must be repeated. Driving RXENPMAPHASEALIGN Low causes phase align to be lost.
9. Wait 32 RXRUSCLK2 clock cycles and then drive RXPMASETPHASE High for 32 RXUSRCLK2 cycles and then deassert it.
10. Drive RXDLYALIGNDISABLE High.

Step 6 requires careful guidance. Normally, CDR lock is detected by measuring the quality of incoming data.

Figure 4-32 shows the RX phase alignment procedure.

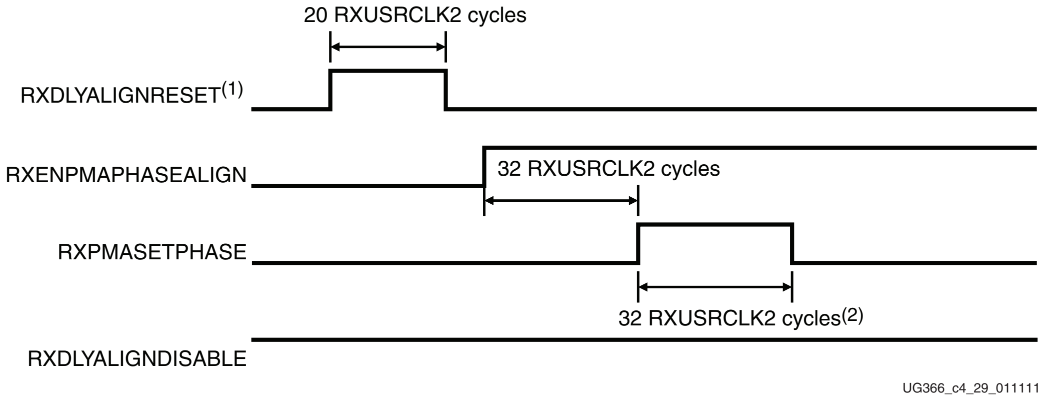


Figure 4-32: **RX Phase Alignment Procedure**

Notes for Figure 4-32.

1. Assert this signal when RXUSRCLK/RXUSRCLK2 clocks are stabilized.
2. Repeat the RXPMASETPHASE procedure if bad data is received (see Figure 4-33)

When the RX elastic buffer is bypassed, data received from the PMA might be distorted due to phase differences as it passes to the PCS. This makes it difficult to determine whether or not bad data is received because the CDR is not locked, or the CDR is locked and the phase alignment has not yet been attempted. To work around this, RX phase alignment must be repeated multiple times and the output data evaluated after each attempt. Good data is received if the phase is aligned while the RX CDR is locked. This process must be repeated until valid data is received at the fabric interface.

The flow diagram in Figure 4-33 shows the series of steps required for successful RX phase alignment. Any number of clock cycles can be used for the CDR lock time, but using a larger number decreases the number of cycles through the states.

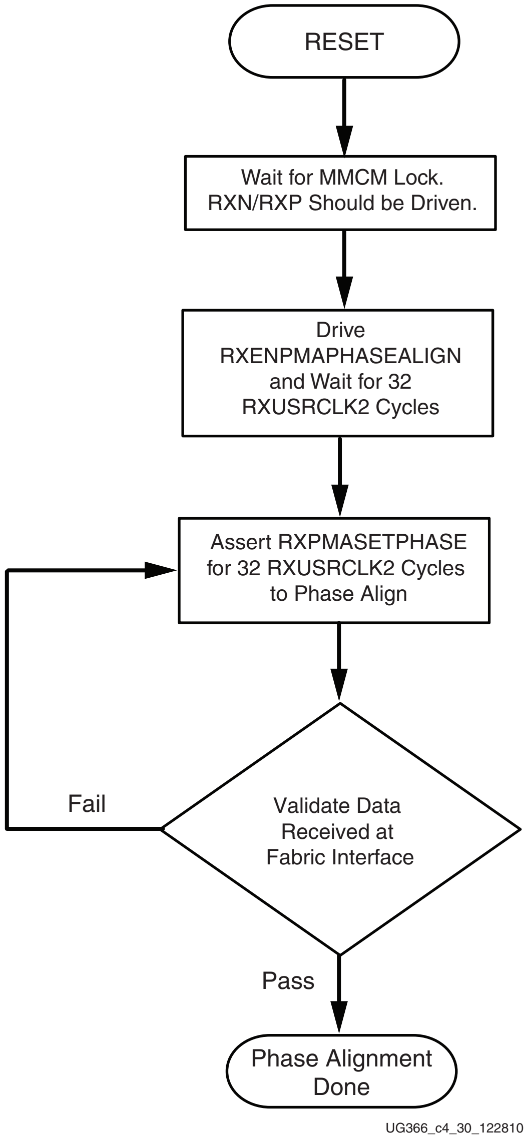


Figure 4-33: **Steps Required for Successful RX Phase Alignment**

RX Elastic Buffer

Functional Description

The GTX 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 4-34 shows the two parallel clock domains, XCLK and RXUSRCLK.

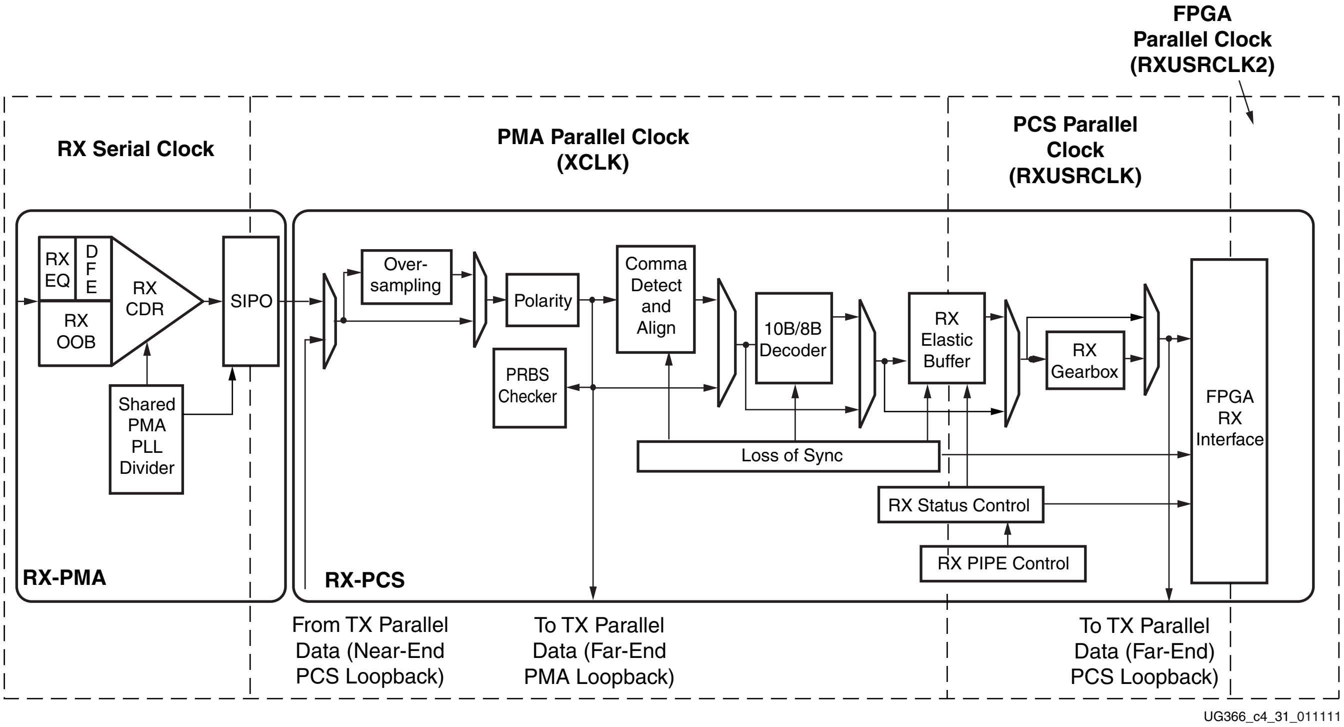


Figure 4-34: **RX Clock Domain**

The GTX transceiver includes an RX elastic buffer to resolve differences between the PMACLK and RXUSRCLK domains. The phase of the two domains can also be matched by using the recovered clock from the transceiver to drive RXUSRCLK and adjusting its phase to match XCLK (see RX Buffer Bypass, page 230). All RX datapaths must use one of these approaches. The costs and benefits of each approach are shown in Table 4-42.

Table 4-42: **Buffering vs. Phase Alignment**

	RX Elastic Buffer	RX Phase Alignment
Clocking Options	Can use recovered clock or local clock (with clock correction)	Must use recovered clock
Initialization	Works immediately	Must wait for all clocks to stabilize then perform alignment procedure
Latency	Buffer latency depends on features used (clock correction and channel bonding)	Lower deterministic latency than using the RX elastic buffer
Clock Correction / Channel Bonding	Required for clock correction / channel bonding	

Ports and Attributes

Table 4-43 defines the RX elastic buffer ports.

Table 4-43: **RX Elastic Buffer Ports**

Port	Dir	Clock Domain	Description
RXBUFRESET	In	Async	Resets the RX elastic buffer logic and re-initializes the RX elastic buffer.
RXBUFSTATUS[2:0]	Out	RXUSRCLK2	Indicates the status of the RX elastic buffer as follows: `000`: Nominal condition `001`: Number of bytes in the buffer are less than CLK_COR_MIN_LAT `010`: Number of bytes in the buffer are greater than CLK_COR_MAX_LAT `101`: RX elastic buffer underflow[1] `110`: RX elastic buffer overflow[1]

Notes:

1. If an RX elastic buffer overflow or an RX elastic buffer underflow condition occurs, the content of the RX elastic buffer becomes invalid, and the RX elastic buffer needs re-initialization by asserting/deasserting RXBUFRESET.

Table 4-44 defines the RX elastic buffer attributes.

Table 4-44: **RX Elastic Buffer Attributes**

Attribute	Type	Description
RX_BUFFER_USE	Boolean	Use or bypass the RX elastic buffer. TRUE: Use the RX elastic buffer (normal mode). FALSE: Permanently bypass the RX elastic buffer (advanced feature). If OVERSAMPLE_MODE is FALSE, RX phase alignment must be used whenever the RX elastic buffer is bypassed.
RX_EN_IDLE_RESET_BUF	Boolean	Enable or disable the automatic RX elastic buffer reset. TRUE: Enable the automatic RX elastic buffer reset when valid signals are not present on the RXN/RXP inputs. FALSE: Disable the automatic RX elastic buffer reset when valid signals are not present on the RXN/RXP inputs. ***Note:*** For channels with large attenuation (lossy channels typically exceeding 15 dB at Nyquist), it is recommended that RX_EN_IDLE_RESET_BUF should be set to FALSE because fast transitioning data patterns like the `101010` sequence in CJPAT/CJTPAT can accidentally trigger an electrical idle.
RX_FIFO_ADDR_MODE	String	FULL: Enable the RX elastic buffer for clock correction and channel bonding support. FAST: Enable the RX elastic buffer for phase compensation without clock correction and channel bonding support.
RX_IDLE_HI_CNT	4-bit Binary	Determines count value after which an assertion of reset due to RX_EN_IDLE_RESET_BUF is triggered after valid data is no longer present on the RXP/RXN lines. Use the Virtex-6 FPGA GTX Transceiver Wizard default.

Table 4-44: **RX Elastic Buffer Attributes *(Cont'd)***

Attribute	Type	Description
RX_IDLE_LO_CNT	4-bit Binary	Determines count value after which a deassertion of reset due to RX_EN_IDLE_RESET_BUF is triggered after valid data once again present on the RXP/RXN lines. Use the Virtex-6 FPGA GTX Transceiver Wizard default.
RX_XCLK_SEL	String	Selects the clock used to drive the RX parallel clock domain (XCLK). "RXREC": (default) XCLK domain driven by recovered clock from CDR. When RX_OVERSAMPLE_MODE is TRUE, the recovered clock is sourced from the oversampling block. "RXUSR": RXUSRCLK port drives RX parallel clock domain. Use this mode when bypassing the RX elastic buffer.

Using the RX Elastic Buffer for Channel Bonding or Clock Correction

The RX elastic buffer is also used for clock correction (see RX Clock Correction) and channel bonding (see RX Channel Bonding, page 246). Clock correction is used in cases where PMACLK and RXUSRCLK are not frequency matched. Table 4-45 lists common clock configurations and shows whether they require clock correction.

Table 4-45: **Common Clock Configurations**

	Needs Clock Correction?
Synchronous System (both sides use same physical oscillator for REFCLK)	No
Separate Reference Clocks, RX uses recovered clock	No
Separate Reference Clocks, RX uses local clock	Yes

To use the RX elastic buffer for channel bonding or clock correction:

- Set RX_BUFFER_USE to TRUE.
- Reset the buffer whenever RXBUFSTATUS indicates an overflow or an underflow.
- The buffer can be reset using GTXRXRESET, RXRESET, or RXBUFRESET (see RX Initialization, page 260).

RX Clock Correction

Functional Description

The RX elastic buffer has an additional benefit: it can tolerate frequency differences between the XCLK and RXUSRCLK domains by performing clock correction. Clock correction actively prevents the RX elastic buffer from getting too full or too empty by deleting or replicating special idle characters in the data stream.

Figure 4-35 shows a conceptual view of clock correction.

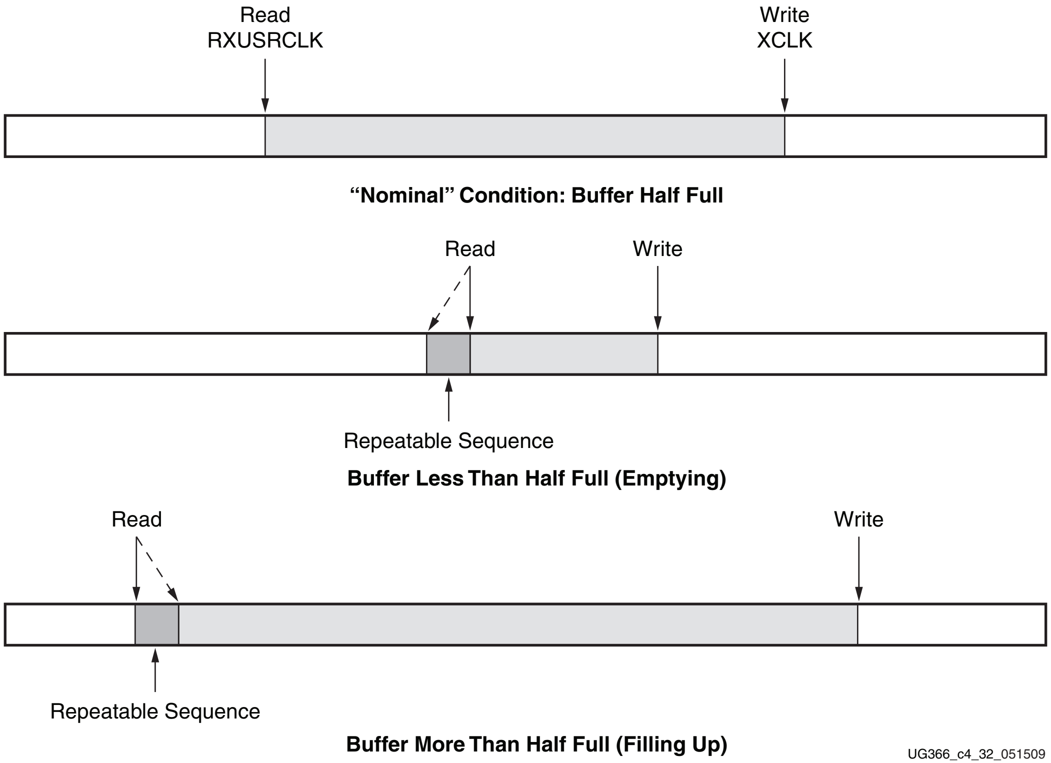

Read
RXUSRCLK
Write
XCLK
"Nominal" Condition: Buffer Half Full
Read
Write
Repeatable Sequence
Buffer Less Than Half Full (Emptying)
Read
Write
Repeatable Sequence
Buffer More Than Half Full (Filling Up)
UG366_c4_32_051509

Figure 4-35: **Clock Correction**

Clock correction should be used whenever there is a frequency difference between XCLK and RXUSRCLK. It can be avoided by using the same frequency source for TX and RX, or by using the recovered clock to drive RXUSRCLK. The RX Elastic Buffer section has more details about the steps required if clock correction is not used.

Ports and Attributes

Table 4-46 defines the RX clock correction ports.

Table 4-46: **RX Clock Correction Ports**

Port	Dir	Clock Domain	Description
RXBUFRESET	In	Async	Resets the RX elastic buffer logic and re-initializes the RX elastic buffer.
RXBUFSTATUS[2:0]	Out	RXUSRCLK2	Indicates the status of the RX elastic buffer as follows: 000: Nominal condition 001: Number of bytes in the buffer are less than CLK_COR_MIN_LAT 010: Number of bytes in the buffer are greater than CLK_COR_MAX_LAT 101: RX elastic buffer underflow[1] 110: RX elastic buffer overflow[1]

Table 4-46: **RX Clock Correction Ports *(Cont'd)***

Port	Dir	Clock Domain	Description
RXCLKCORCNT[2:0]	Out	RXUSRCLK2	Reports the clock correction status of the RX elastic buffer: 000: No clock correction 001: 1 sequence skipped 010: 2 sequences skipped 011: 3 sequences skipped 100: 4 sequences skipped 101: Reserved 110: 2 sequences added 111: 1 sequence added

Notes:

1. If an RX elastic buffer overflow or an RX elastic buffer underflow condition occurs, the content of the RX elastic buffer becomes invalid, and the RX elastic buffer needs re-initialization by asserting/deasserting RXBUFRESET.

Table 4-47 defines the RX clock correction attributes.

Table 4-47: **RX Clock Correction Attributes**

Attribute	Type	Description
CLK_COR_ADJ_LEN	Integer	This attribute defines the size of the adjustment (number of bytes repeated or skipped) in a clock correction. The bytes skipped or repeated always start from the beginning of the clock correction sequence to allow more bytes to be replaced than in the specified clock correction sequence. Valid lengths are 1, 2, and 4 bytes.
CLK_COR_DET_LEN	Integer	This attribute defines the length of the sequence that the transceiver matches to detect opportunities for clock correction. Valid lengths are 1, 2, and 4 bytes.
CLK_COR_INSERT_IDLE_FLAG	Boolean	Controls whether the RXRUNDISP input status indicates running disparity or inserted-idle (clock correction sequence) flag. FALSE: RXRUNDISP indicates running disparity when RXDATA is decoded data. TRUE: RXRUNDISP is raised for the first byte of each inserted (repeated) clock correction ("Idle") sequence (when RXDATA is decoded data).
CLK_COR_KEEP_IDLE	Boolean	Controls whether the RX elastic buffer must retain at least one clock correction sequence in the byte stream. FALSE: The transceiver can remove all clock correction sequences to further re-center the RX elastic buffer during clock correction. TRUE: In the final RXDATA stream, the transceiver must leave at least one clock correction sequence per continuous stream of clock correction sequences.
CLK_COR_MAX_LAT	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 48.

Table 4-47: **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 un-initialized) data bytes in the buffer. Valid values for this attribute range from 3 to 48.
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 without clock correction that must occur between successive clock corrections. If this attribute is zero, no limit is placed on how frequently clock correction can occur. Valid values for this attribute range from 0 to 31.
CLK_COR_SEQ_1_1	10-bit Binary	The CLK_COR_SEQ_1 attributes are used in conjunction with CLK_COR_SEQ_1_ENABLE to define clock correction sequence 1. The sequence is made up of four subsequences. Each subsequence is 10 bits long. The rules for setting the subsequences depend on RX_DATA_WIDTH and RX_DECODE_SEQ_MATCH. See Setting Clock Correction Sequences, page 244 to learn how to set clock correction subsequences. Not all subsequences need to be used. CLK_COR_DET_LEN determines how many of the sequence are used for a match. If CLK_COR_DET_LEN = 1, only CLK_COR_SEQ_1_1 is used. CLK_COR_SEQ_1_ENABLE can be used to make parts of the sequence don't cares. If CLK_COR_SEQ_1_ENABLE[k] is 0, CLK_COR_SEQ_1_k is a don't care subsequence and is always considered to be a match.
CLK_COR_SEQ_1_2		
CLK_COR_SEQ_1_3		
CLK_COR_SEQ_1_4		
CLK_COR_SEQ_1_ENABLE	4-bit Binary	

Table 4-47: **RX Clock Correction Attributes *(Cont'd)***

Attribute	Type	Description
CLK_COR_SEQ_2_1	10-bit Binary	The CLK_COR_SEQ_2 attributes are used in conjunction with CLK_COR_SEQ_2_ENABLE to define the second clock correction sequence. This second sequence is used as an alternate sequence for clock correction when CLK_COR_SEQ_2_USE is TRUE: if either sequence 1 or sequence 2 arrives, clock correction is performed. The sequence is made up of four subsequences. Each subsequence is 10 bits long. The rules for setting the subsequences depend on RX_DATA_WIDTH and RX_DECODE_SEQ_MATCH. See Setting Clock Correction Sequences, page 244 to learn how to set clock correction subsequences. Not all subsequences need to be used. CLK_COR_DET_LEN determines how much of the sequence is used for a match. If CLK_COR_DET_LEN = 1, only CLK_COR_SEQ_2_1 is used. CLK_COR_SEQ_2_ENABLE can be used to make parts of the sequence don't care. If CLK_COR_SEQ_2_ENABLE[k] is 0, CLK_COR_SEQ_2_k is a don't care byte subsequence and is always considered to be a match.
CLK_COR_SEQ_2_2		
CLK_COR_SEQ_2_3		
CLK_COR_SEQ_2_4		
CLK_COR_SEQ_2_ENABLE	4-bit Binary	
CLK_COR_SEQ_2_USE	Boolean	Determines if the second clock correction sequence is to be used. When set to TRUE, the second clock correction sequence also triggers clock correction.

CLK_CORRECT_USE	Boolean	Enables clock correction.
		TRUE: Clock correction enabled
		FALSE: Clock correction disabled. In this case, set:
		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`

RX_DATA_WIDTH	Integer	Sets the receiver external data width: 8/10: 1-byte interface 16/20: 2-byte interface 32/40: 4-byte interface If 8B10B is used, this attribute must be a multiple of 10.
RX_DECODE_SEQ_MATCH	Boolean	Determines whether sequences are matched against the input to the 8B/10B decoder or the output. Used for the clock correction circuit and the channel bonding circuit. TRUE: Sequences are matched against the output of the 8B/10B decoder. K characters and disparity information is used. Bit ordering of the 8B/10B output is used. FALSE: Sequences are matched against non-encoded data. Bit ordering is as for an non-encoded parallel interface.

Using RX Clock Correction

The user must follow the steps described in this section to use the receiver clock correction.

Enabling Clock Correction

Each GTX transceiver includes a clock correction circuit that performs clock correction by controlling the pointers of the RX elastic buffer. To use clock correction, RX_BUFFER_USE is set to TRUE to turn on the RX elastic buffer, and CLK_CORRECT_USE is set to TRUE to turn on the clock correction circuit.

Clock correction is triggered when the RX elastic buffer latency is too high or too low, and the clock correction circuit detects a match sequence. To use clock correction, the clock correction circuit must be configured to set the following items:

- RX elastic buffer limits
- Clock correction sequence

Setting RX Elastic Buffer Limits

The RX elastic buffer limits are set using CLK_COR_MIN_LAT (minimum latency) and CLK_COR_MAX_LAT (maximum latency). When the number of bytes in the RX elastic buffer drops below CLK_COR_MIN_LAT, the clock correction circuit writes an additional CLK_COR_ADJ_LEN bytes from the first clock correction sequence it matches to prevent the buffer from underflowing. Similarly, when the number of bytes in the RX elastic buffer exceeds CLK_COR_MAX_LAT, the clock correction circuit deletes CLK_COR_ADJ_LEN bytes from the first clock correction sequence it matches, starting with the first byte of the sequence.

Setting Clock Correction Sequences

The clock correction sequences are programmed using the CLK_COR_SEQ_1_* attributes and CLK_COR_ADJ_LEN. Each CLK_COR_SEQ_1_* attribute corresponds to one subsequence in clock correction sequence 1. CLK_COR_ADJ_LEN is used to set the number of subsequences to be matched. If the 20-bit internal datapath is used, the clock correction circuit matches all 10 bits of each subsequence. If the 16-bit internal datapath is 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 (RXDEC8B10BUSE is High), RX_DECODE_SEQ_MATCH is set to TRUE to search the output of the 8B/10B decoder for sequence matches instead of non-decoded data. This allows the circuit to look for 8-bit values with either positive or negative disparity, and to distinguish K characters from regular characters (see TX 8B/10B Encoder, page 143 and RX 8B/10B Decoder, page 227 for details). Figure 4-36 shows how to set a clock correction sequence byte when RX_DECODE_SEQ_MATCH is TRUE.

When RX_DECODE_SEQ_MATCH is FALSE, the sequence must exactly match non-encoded incoming data.

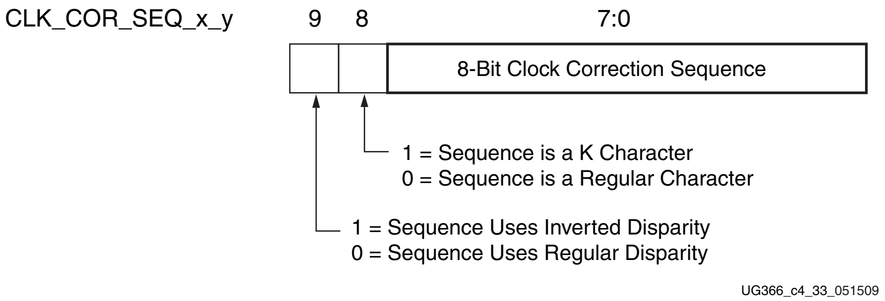

CLK_COR_SEQ_x_y
9
8
7:0
8-Bit Clock Correction Sequence
1 = Sequence is a K Character
0 = Sequence is a Regular Character
1 = Sequence Uses Inverted Disparity
0 = Sequence Uses Regular Disparity
UG366_c4_33_051509

Figure 4-36: **Clock Correction Subsequence Settings with RX_DECODE_SEQ_MATCH = TRUE**

Some protocols use clock correction sequences with don't care subsequences. The clock correction circuit can be programmed to recognize these sequences using CLK_COR_SEQ_1_ENABLE and CLK_COR_SEQ_2_ENABLE. When the enable bit for a sequence is Low, that byte is considered matched no matter what the value. Figure 4-37 shows the mapping between the clock correction sequences and the clock correction sequence enable bits.

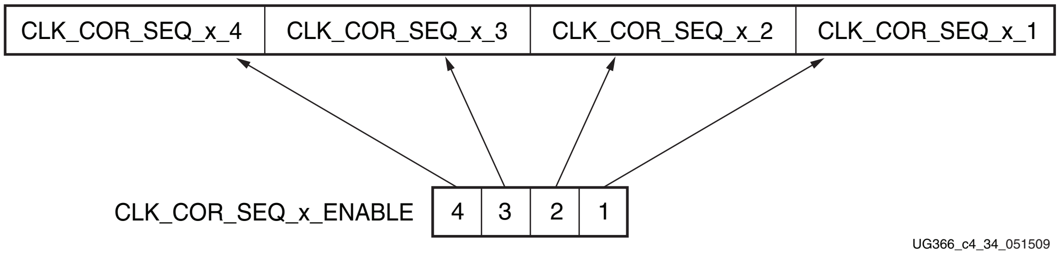


Figure 4-37: **Clock Correction Sequence Mapping**

Clock Correction Options

CLK_COR_REPEAT_WAIT is used to control the clock correction frequency. This value is set to the minimum number of RXUSRCLK cycles required between clock correction events. This attribute is set to 0 to allow clock correction to occur any time.

Some protocols allow clock correction to occur at any time, but require that if the clock correction circuit removes sequences, at least one sequence stays in the stream. For protocols with this requirement, CLK_COR_KEEP_IDLE is set to TRUE.

Monitoring Clock Correction

The clock correction circuit can be monitored using the RXCLKCORCNT and RXBUFSTATUS ports. The RXCLKCORCNT entry in Table 4-46 shows how to decode the values of RXCLKCORCNT to determine the status of the clock correction circuit. The RXBUFSTATUS entry in Table 4-46 shows how to decode the values of RXBUFSTATUS to determine how full the RX elastic buffer is.

In addition to RXCLKCORCNT and RXBUFSTATUS, RXRUNDISP can be taken from the 8B/10B decoder interface (see RX 8B/10B Decoder, page 227) and used to indicate when RXDATA has the first byte of a clock correction sequence that was replicated and added to the RX elastic buffer. To use the RXRUNDISP port to indicate inserted idles instead of the current RX running disparity, CLK_COR_INSERT_IDLE_FLAG is set to TRUE.

RX Channel Bonding

Functional Description

The RX elastic buffer can also be used for channel bonding. Channel bonding cancels out the skew between GTX transceiver lanes by using the RX elastic buffer as a variable latency block. The transmitter sends a pattern simultaneously on all lanes, which the channel bonding circuit uses to set the latency for each lane so that data is presented without skew at the FPGA RX interface.

Figure 4-38 shows a conceptual view of channel bonding.

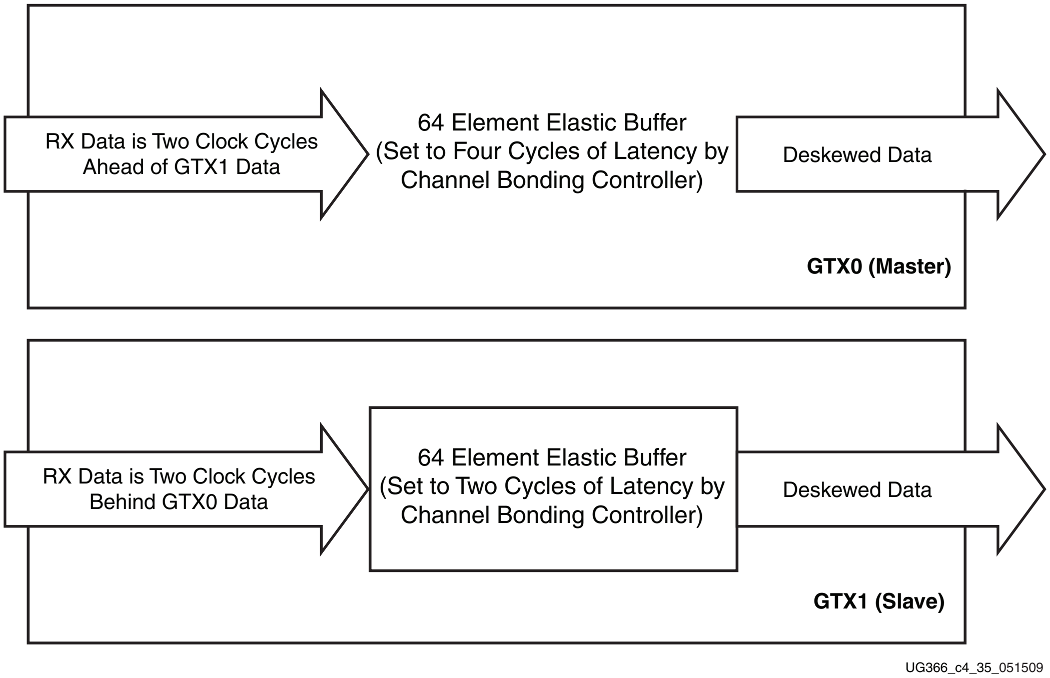

RX Data is Two Clock Cycles Ahead of GTX1 Data
64 Element Elastic Buffer (Set to Four Cycles of Latency by Channel Bonding Controller)
Deskewed Data
GTX0 (Master)
RX Data is Two Clock Cycles Behind GTX0 Data
64 Element Elastic Buffer (Set to Two Cycles of Latency by Channel Bonding Controller)
Deskewed Data
GTX1 (Slave)
UG366_c4_35_051509

Figure 4-38: **Channel Bonding Conceptual View**

Ports and Attributes

Table 4-48 defines the RX channel bonding ports.

Table 4-48: **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.

Table 4-48: **RX Channel Bonding Ports *(Cont'd)***

Port	Dir	Clock Domain	Description
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[3:0]	In	RXUSRCLK / RXUSRCLK2	FPGA channel bonding control. This signal is used only by slaves. It is driven from another transceiver's RXCHBONDO port that is the master in this configuration. If GEN_RXUSRCLK is set to TRUE, the timing of RXCHBONDI port is synchronous to RXUSRCLK2.
RXCHBONDO[3:0]	Out	RXUSRCLK / RXUSRCLK2	FPGA channel bonding control. This signal is used by the master and slaves to pass channel bonding and clock correction control to other transceivers' RXCHBONDI ports. If GEN_RXUSRCLK is set to TRUE, the timing of RXCHBONDI port is synchronous to RXUSRCLK2.
RXCHBONDLEVEL[2:0]	In	RXUSRCLK2	Indicates the amount of internal pipelining used for the elastic buffer control signals. A higher value permits more daisy-chaining of RXCHBONDO and RXCHBONDI to ease placement and routing constraints. To minimize required latency through the RX elastic buffer, CHAN_BOND_LEVEL in the master set to the smallest value possible for the required amount of daisy-chaining. See Connecting Channel Bonding Ports, page 250 for channel bonding to learn how to set the channel bonding level.
RXCHBONDMASTER	In	RXUSRCLK2	Indicates that the transceiver is master for channel bonding. Its RXCHBONDO port directly drives 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 may directly drive RXCHBONDI ports on one or more other SLAVE transceivers. This port cannot be driven High at the same time as RXCHBONDMASTER.
RXENCHANSYNC	In	RXUSRCLK2	This port enables channel bonding (from the FPGA logic to both the master and slaves).

Table 4-49 defines the RX channel bonding attributes.

Table 4-49: **RX Channel Bonding Attributes**

Attribute	Type	Description
CHAN_BOND_1_MAX_SKEW	Integer	These attributes control the number of USRCLK cycles that the master waits before ordering the slaves to execute channel bonding. This attribute determines the maximum skew that can be handled by channel bonding. It must always be less than one-half the minimum distance (in bytes or 10-bit codes) between channel bonding sequences. Valid values range from 1 to 14.
CHAN_BOND_2_MAX_SKEW	Integer	
CHAN_BOND_KEEP_ALIGN	Boolean	Allows preservation of ALIGN characters during channel bonding for PCI Express designs.
CHAN_BOND_SEQ_1_1	10-bit Binary	The CHAN_BOND_SEQ_1 attributes are used in conjunction with CHAN_BOND_SEQ_1_ENABLE to define channel bonding sequence 1. Each subsequence is 10 bits long. The rules for setting the subsequences depend on RX_DATA_WIDTH and RX_DECODE_SEQ_MATCH. See Setting Channel Bonding Sequences, page 253 to learn how to set channel bonding subsequences. Not all subsequences need to be used. CHAN_BOND_SEQ_LEN determines how much of the sequence is used for a match. If CHAN_BOND_SEQ_LEN = 1, only CHAN_BOND_SEQ_1_1 is used. CHAN_BOND_SEQ_1_ENABLE can be used to make parts of the sequence don't care. If CHAN_BOND_SEQ_1_ENABLE[k] is 0, CHAN_BOND_SEQ_1_k is a don't-care subsequence and is always considered to be a match.
CHAN_BOND_SEQ_1_2		
CHAN_BOND_SEQ_1_3		
CHAN_BOND_SEQ_1_4		
CHAN_BOND_SEQ_1_ENABLE	4-bit Binary	
CHAN_BOND_SEQ_2_1	10-bit Binary	The CHAN_BOND_SEQ_2 attributes are used in conjunction with CHAN_BOND_SEQ_2_ENABLE to define the second channel bonding sequence. When CHAN_BOND_SEQ_2_USE is TRUE, the second sequence is used as an alternate sequence to trigger channel bonding. Each subsequence is 10 bits long. The rules for setting the subsequence depend on RX_DATA_WIDTH and RX_DECODE_SEQ_MATCH. See Setting Channel Bonding Sequences, page 253 to learn how to set channel bonding sequences. Not all subsequences need to be used. CHAN_BOND_SEQ_LEN determines how many of the subsequences are used for a match. If CHAN_BOND_SEQ_LEN = 1, only CHAN_BOND_SEQ_2_1 is used. CHAN_BOND_SEQ_2_ENABLE can be used to make parts of the sequence don't care. If CHAN_BOND_SEQ_2_ENABLE[k] is 0, CHAN_BOND_SEQ_2_k is a don't-care subsequence and is always considered to be a match.
CHAN_BOND_SEQ_2_2		
CHAN_BOND_SEQ_2_3		
CHAN_BOND_SEQ_2_4		
CHAN_BOND_SEQ_2_ENABLE	4-bit Binary	
CHAN_BOND_SEQ_2_CFG	5-bit Binary	Attributes to control channel bonding for PCIe FTS. The attribute should be set as follows: `00000`: For protocols other than PCIe `11111`: For PCIe protocol
CHAN_BOND_SEQ_2_USE	Boolean	Determines if the second channel bonding sequence is to be used. TRUE: Channel bonding can be triggered by channel bonding sequence 1 or 2. FALSE: Channel bonding is only triggered by sequence 1.

Table 4-49: **RX Channel Bonding Attributes *(Cont'd)***

Attribute	Type	Description
CHAN_BOND_SEQ_LEN	Integer	Defines the length in bytes of the channel bonding sequence that the transceiver matches to detect opportunities for channel bonding. Valid lengths are 1, 2, and 4 bytes.
PCI_EXPRESS_MODE	Boolean	The default for this attribute is TRUE for PCI Express designs. For all other protocols, the default setting is FALSE. Setting this attribute to TRUE enables certain operations specific to PCI Express operation, specifically, recognizing TXELECIDLE = 1, TXCHARDISPMODE = 1, TXCHARDISPVAL = 0 as a request to power down the channel. TXCHARDISPMODE = 1 and TXCHARDISPVAL = 0 encode the PIPE interface signal TXCompliance = 1 of the PIPE specification. The TXCHARDISPMODE and TXCHARDISPVAL settings encode for PIPE and enable special support for FTS lane deskew. For channel bonding, setting this attribute to TRUE allows channel bonding on a shorter sequence with the reuse of prior channel bonding information.
RX_DATA_WIDTH	Integer	Sets the receiver external data width: 8/10: 1-byte interface 16/20: 2-byte interface 32/40: 4-byte interface If 8B10B is used, this attribute must be a multiple of 10.

Using RX Channel Bonding

The user must follow the steps described below in order to use the receiver channel bonding.

Enabling Channel Bonding

Each GTX transceiver includes a circuit that performs channel bonding by controlling the pointers of the RX elastic buffer. To use channel bonding, the RX_BUFFER_USE attribute must be TRUE to turn on the elastic buffer.

Each GTX transceiver has a channel bonding circuit. Configuring a GTX transceiver for channel bonding requires the following steps:

1. Set the channel bonding mode for each GTX transceiver.
2. 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 transceiver determines whether channel bonding is active and whether the GTX transceiver is the master or a slave. Each set of channel-bonded GTX transceivers must have one master and can have any number of slaves. To turn on channel bonding for a group of GTX transceivers, one transceiver is set to Master. The remaining GTX transceivers in the group are set to Slaves.

Connecting Channel Bonding Ports

The channel bonding operation requires connecting the master GTX transceiver RXCHBONDO port to the RXCHBONDI port of all slaves in the group. A direct connection is required for adjacent GTX transceivers. To directly connect a master to a slave:

1. Connect the RXCHBONDO port of the master to the RXCHBONDI port of the slave.
2. Tie the RXCHBONDMASTER of the master transceiver High.
3. Tie the RXCHBONDSLAVE of each slave transceiver High.

When GTX transceivers are directly connected, meeting the timing constraints becomes difficult as the transceivers get further apart. The solution to this problem is to connect the transceivers in a daisy chain. Daisy chaining is performed using the 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 4-39 and Figure 4-40 show two daisy-chain examples.

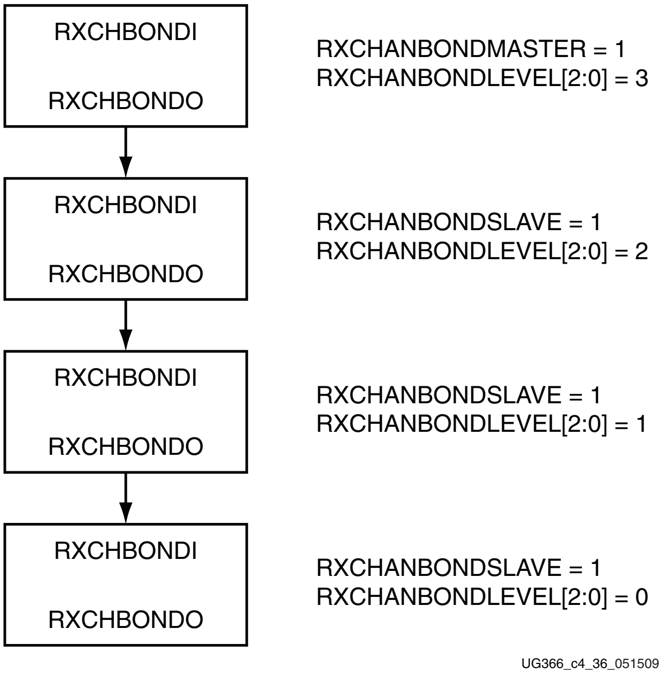


Figure 4-39: **Channel Bonding Daisy Chain Example 1**

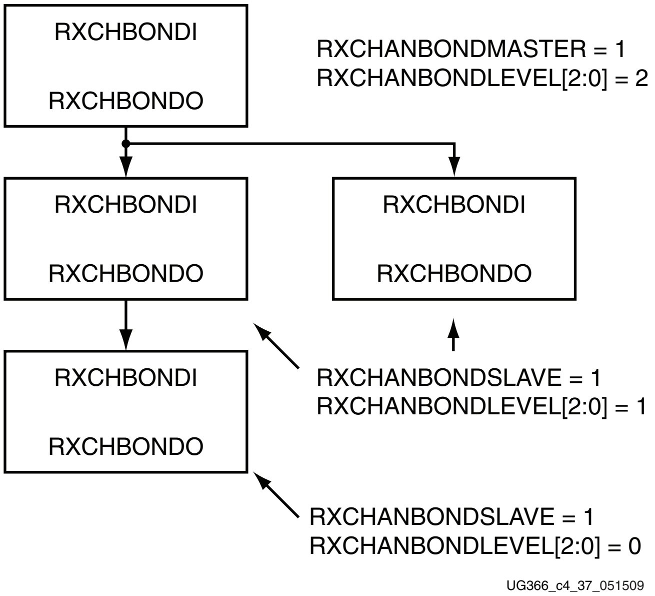


Figure 4-40: **Channel Bonding Daisy Chain Example 2**

To set up a daisy chain, first connect the GTX transceivers using RXCHBONDO and RXCHBONDI to create a path from the RXCHBONDI port of each slave to the RXCHBONDO port of the master. Use the following steps to set the RXCHANBONDLEVEL[2:0] for the GTX transceivers in the chain:

1. Set the RXCHANBONDLEVEL[2:0] of the master to 7.
2. Set the RXCHANBONDLEVEL[2:0] of each slave to the RXCHANBONDLEVEL[2:0] of the GTX transceiver driving the slave's RXCHBONDI port minus 1.
3. Find the slave with the lowest level. Subtract this level from the RXCHANBONDLEVEL[2:0] of all GTX transceivers so that the lowest slave has level 0 and the master has the minimum level required to service all the slaves.

Any number of GTX transceivers can be channel bonded together as long as the timing constraints of the design are met and all clocking guidelines are followed (see Figure 2-5, page 109).

When the connections between channel bonding ports among GTX transceivers are being decided, the designer must remember that RXCHBONDI and RXCHBONDO belong to the RXUSRCLK clock domain. Meeting the timing constraint of RXUSRCLK becomes increasingly difficult as RXUSRCLK increases in frequency and as directly connected transceivers get further apart.

Selecting a GTX transceiver in the middle of the GTX 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 transceiver column, connections can be made to GTX transceivers north and south of the master. This configuration allows for daisy chaining of up GTX transceivers from one Quad above and one Quad below the channel bonding master. Because of the GTX transceiver dedicated clock routing structure, an additional benefit of having the channel bonding master at the center of the GTX transceiver column is that up to 12 GTX transceivers can be channel bonded together using a single clock pin pair. If more than 12 GTX transceivers are channel bonded together, the use of additional clock pins is required as well as using the same oscillator (see Figure 2-5, page 109).

As long as timing constraints are met, the following items apply:

- There is no limit to the number of GTX transceivers that can be on a particular RXCHANBONDLEVEL[2:0].
- GTX transceivers can be on the same RXCHANBONDLEVEL[2:0].

Figure 4-41 shows an example for channel bonding 16 GTX transceivers. This example is only one of many ways to channel bond 16 GTX transceivers. In this example, transceivers of the X0 column are channel bonded together.

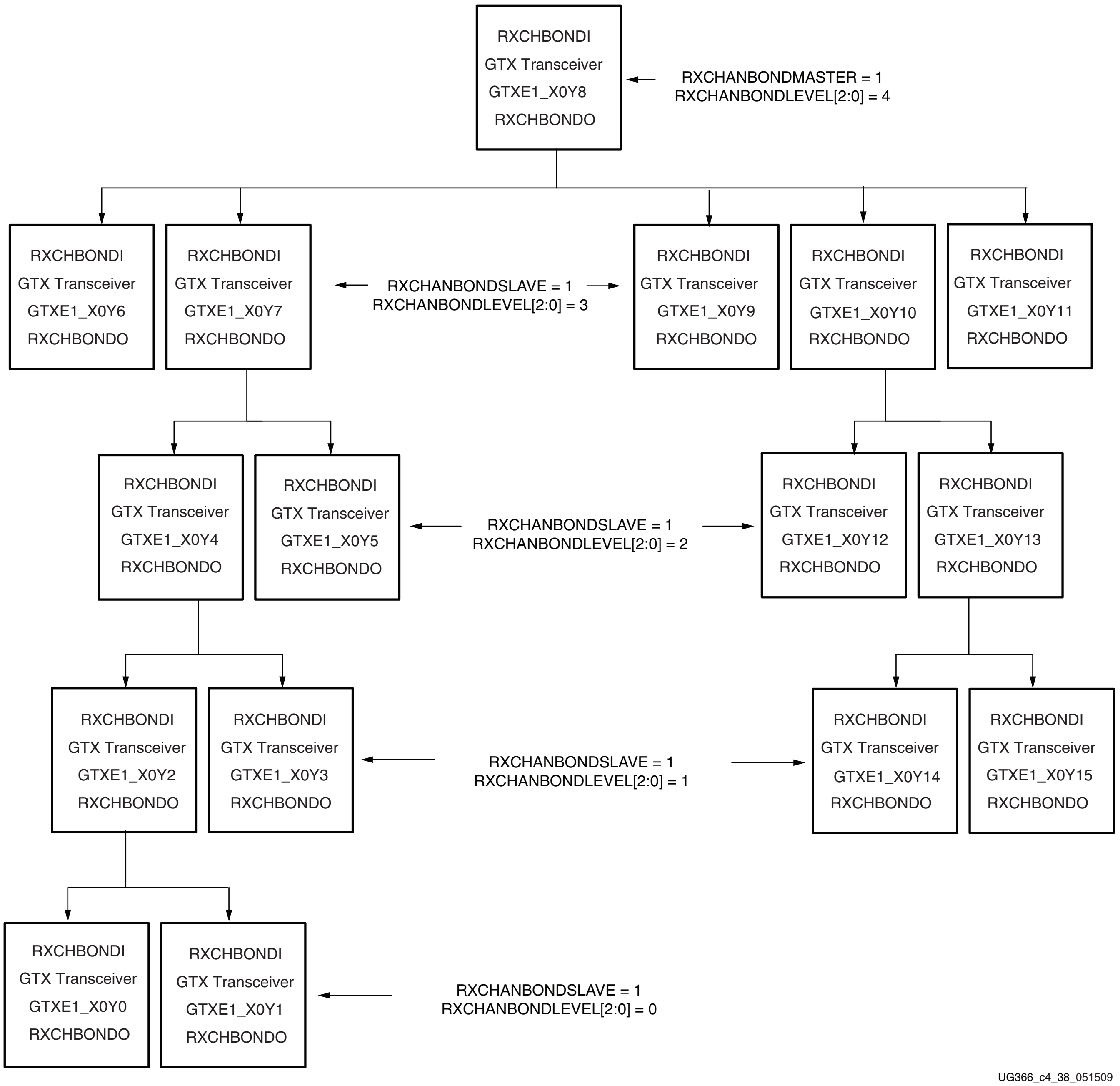


Figure 4-41: **Channel Bonding Example Using 16 GTX Transceivers**

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 RX_DECODE_SEQ_MATCH (see RX Clock Correction, page 239). Figure 4-42 shows how the subsequence bits are mapped.

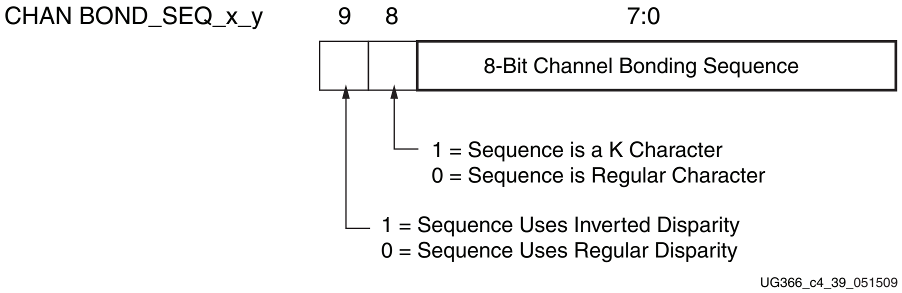

CHAN BOND_SEQ_x_y
9
8
7:0
8-Bit Channel Bonding Sequence
1 = Sequence is a K Character
0 = Sequence is Regular Character
1 = Sequence Uses Inverted Disparity
0 = Sequence Uses Regular Disparity
UG366_c4_39_051509

Figure 4-42: **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 4-43 shows the mapping of the enable attributes for the channel bonding subsequences.

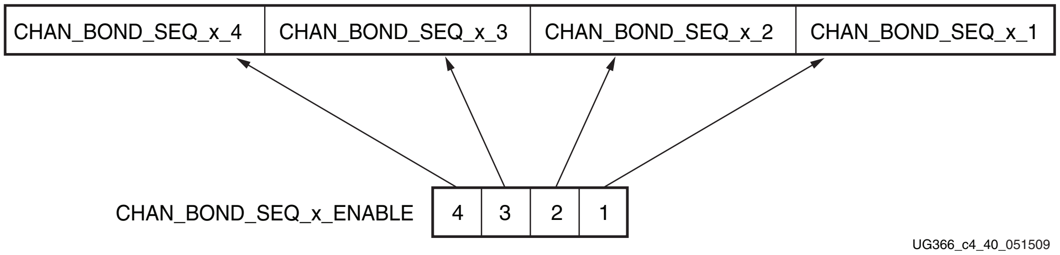


Figure 4-43: **Channel Bonding Sequence Mapping**

Setting the Maximum Skew

When the master receives a channel bonding sequence, it does not trigger channel bonding immediately. Several more bytes must arrive in case the slaves have more latency. This wait time effectively becomes the maximum skew that the RX elastic buffer can handle. If the skew is greater than this wait time, the slaves might not receive the sequence by the time the master triggers channel bonding (see Figure 4-44).

Figure 4-44 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.

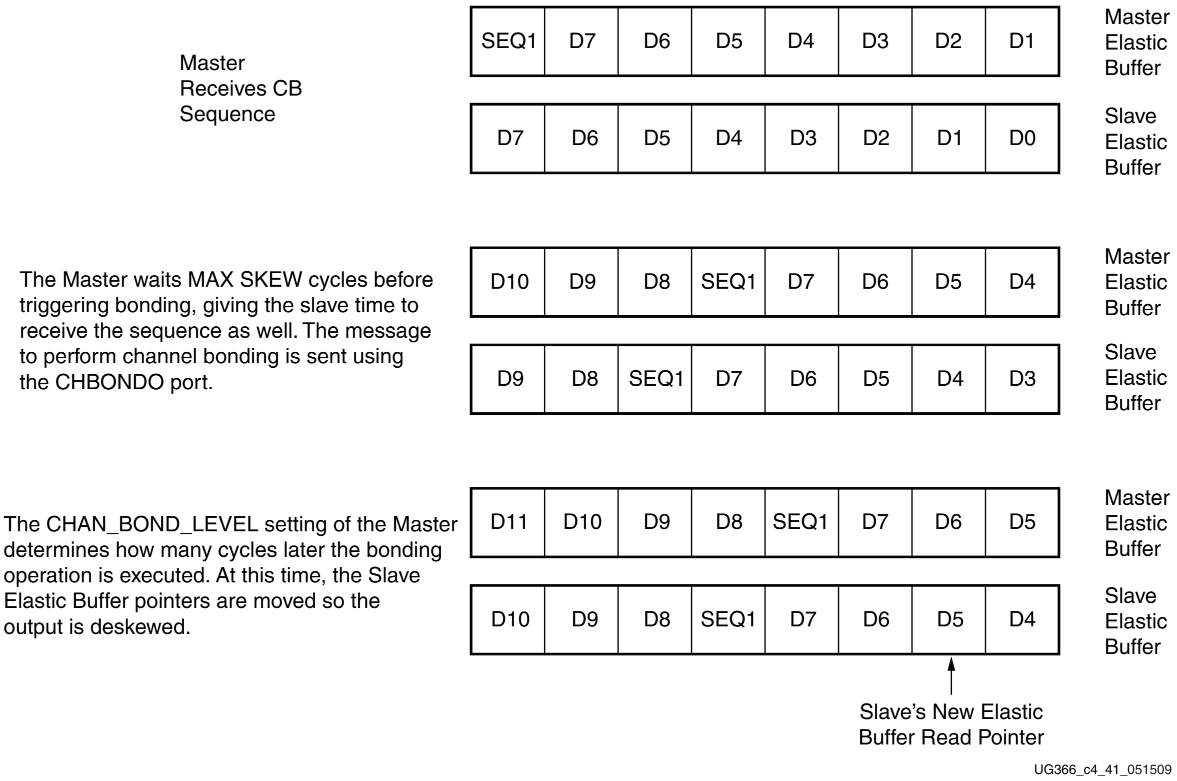

Master
Receives CB
Sequence
SEQ1 | D7 | D6 | D5 | D4 | D3 | D2 | D1
Master Elastic Buffer
D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0
Slave Elastic Buffer
The Master waits MAX SKEW cycles before triggering bonding, giving the slave time to receive the sequence as well. The message to perform channel bonding is sent using the CHBONDO port.
D10 | D9 | D8 | SEQ1 | D7 | D6 | D5 | D4
Master Elastic Buffer
D9 | D8 | SEQ1 | D7 | D6 | D5 | D4 | D3
Slave Elastic Buffer
The CHAN_BOND_LEVEL setting of the Master determines how many cycles later the bonding operation is executed. At this time, the Slave Elastic Buffer pointers are moved so the output is deskewed.
D11 | D10 | D9 | D8 | SEQ1 | D7 | D6 | D5
Master Elastic Buffer
D10 | D9 | D8 | SEQ1 | D7 | D6 | D5 | D4
Slave Elastic Buffer
Slave's New Elastic Buffer Read Pointer
UG366_c4_41_051509

Figure 4-44: **Channel Bonding Example (CHAN_BOND_*_MAX_SKEW = 2 and Master RXCHANBONDLEVEL[2:0] = 1)**

CHAN_BOND_1_MAX_SKEW and CHAN_BOND_2_MAX_SKEW are used to set the maximum skew allowed for channel bonding sequences 1 and 2, respectively. The maximum skew range is 1 to 14. 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 239) 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 uses output pins RXDATA[31:0] and RXHEADER[2:0] for receiving data. Similar to TX Gearbox, page 146, the RX gearbox operates with the PMA using a single clock. Because of this, occasionally, the output data is invalid. Output pins RXHEADERVALID and RXDATAVALID determine if the appropriate header and data are valid. The RX gearbox only supports 2-byte and 4-byte interfaces. A 1-byte interface is not supported.

The data out of the RX gearbox is not necessarily aligned. Alignment is done in the FPGA logic. The RXGEARBOXSLIP port can be used to slip the data from the gearbox cycle-by-cycle until the correct alignment is reached. It takes a specified amount of cycles before the bitslip operation is processed and the output data is stable.

Descrambling of the data and block synchronization is done in the FPGA logic.

Ports and Attributes

Table 4-50 defines the RX gearbox ports.

Table 4-50: **RX Gearbox Ports**

Port	Dir	Clock Domain	Description
RXDATAVALID	Out	RXUSRCLK2	Status output when Gearbox 64/66 or 64/67 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 4-byte interface and every 64 cycles for the 2-byte interface.
RXGEARBOXSLIP	In	RXUSRCLK2	When High, this port causes the gearbox contents to slip by one bit. It is used to achieve alignment with the FPGA logic. Asserting this port for one RXUSRCLK2 cycle changes the data alignment coming out of the gearbox. When RXGEARBOXSLIP is asserted for more than one cycle, the gearbox realigns the data once for each RXUSRCLK2 cycle that it is held High.
RXHEADER[2:0]	Out	RXUSRCLK2	Header outputs for 64/66 (1:0) and 64/67 (2:0).
RXHEADERVALID	Out	RXUSRCLK2	Indicates that the RXHEADER is valid when using the gearbox.
RXSTARTOFSEQ	Out	RXUSRCLK2	When Gearbox 64/66 or 64/67 is enabled, this output indicates when the sequence counter is 0 for the present RXDATA outputs.

Table 4-51 defines the RX gearbox attributes.

Table 4-51: **RX Gearbox Attributes**

Attribute	Type	Description
GEARBOX_ENDEC	3-bit Binary	Gearbox Modes: 000: 64B/67B using external sequence counter 001: 64B/66B using external sequence counter 010: 64B/67B using internal sequence counter 011: 64B/66B using internal sequence counter
RXGEARBOX_USE	Boolean	When TRUE, this attribute enables the RX gearbox.

Enabling the RX Gearbox

To enable the RX gearbox for the GTX transceiver, the RXGEARBOX_USE attribute is set to TRUE. Bit 2 of the GEARBOX_ENDEC attribute must be set to 0 to enable the gearbox decoder. The decoder controls the GTX transceiver's TX gearbox and RX gearbox. The TX gearbox and RX gearbox use the same mode.

RX Gearbox Operating Modes

The RX gearbox operates the same in either external sequence counter mode or internal sequence counter mode. The RX gearbox only supports 2-byte and 4-byte logic interfaces to the FPGA logic. A 1-byte logic interface is not supported.

As shown in Figure 4-45, either mode uses the RXDATA, RXHEADER, RXDATAOUTVALID, and RXHEADEROUTVALID outputs in addition to the RXGEARBOXSLIP input.

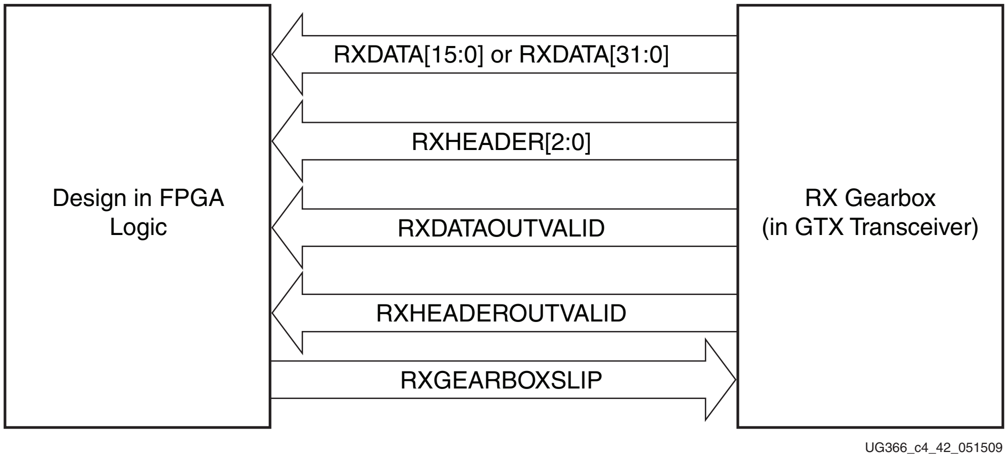


Figure 4-45: **RX Gearbox in Either Internal or External Sequence Mode**

The RX gearbox internally manages all sequencing, which differs from the TX gearbox option of either internal or external sequencing. Depending on whether a 2-byte or a 4-byte interface is used, RXDATAOUTVALID and RXHEADEROUTVALID assert and deassert for different periods of length. The RX gearbox encounters similar data and header pauses found in the TX gearbox. Figure 4-46 shows such a pause in addition to RXHEADERVALID asserting every other cycle and RXDATAVALID being deasserted for one cycle.

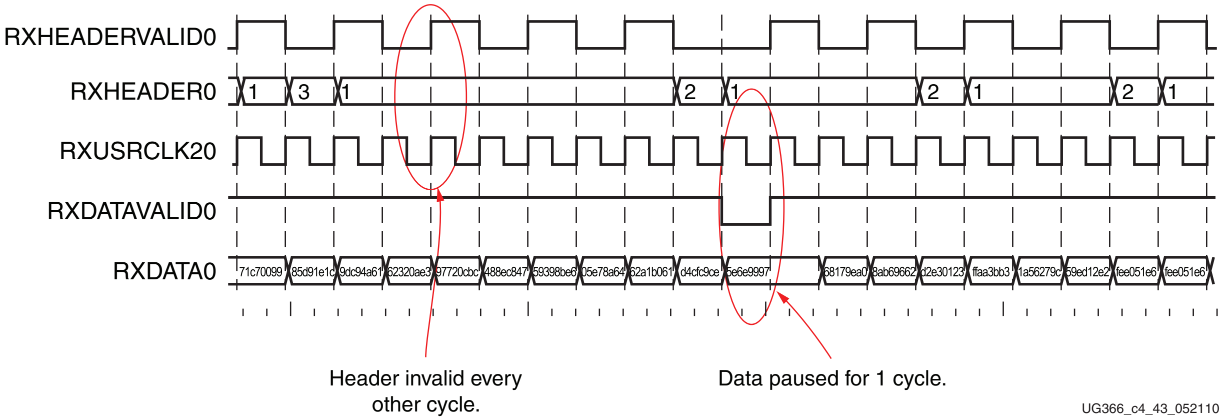


Figure 4-46: **RX Gearbox When Using 64B/66B Encoding and a 4-Byte Interface**

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. 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 4-47 shows the operation of a block synchronization state machine. The Virtex-6 FPGA GTX Transceiver Wizard has example code for this type of module.

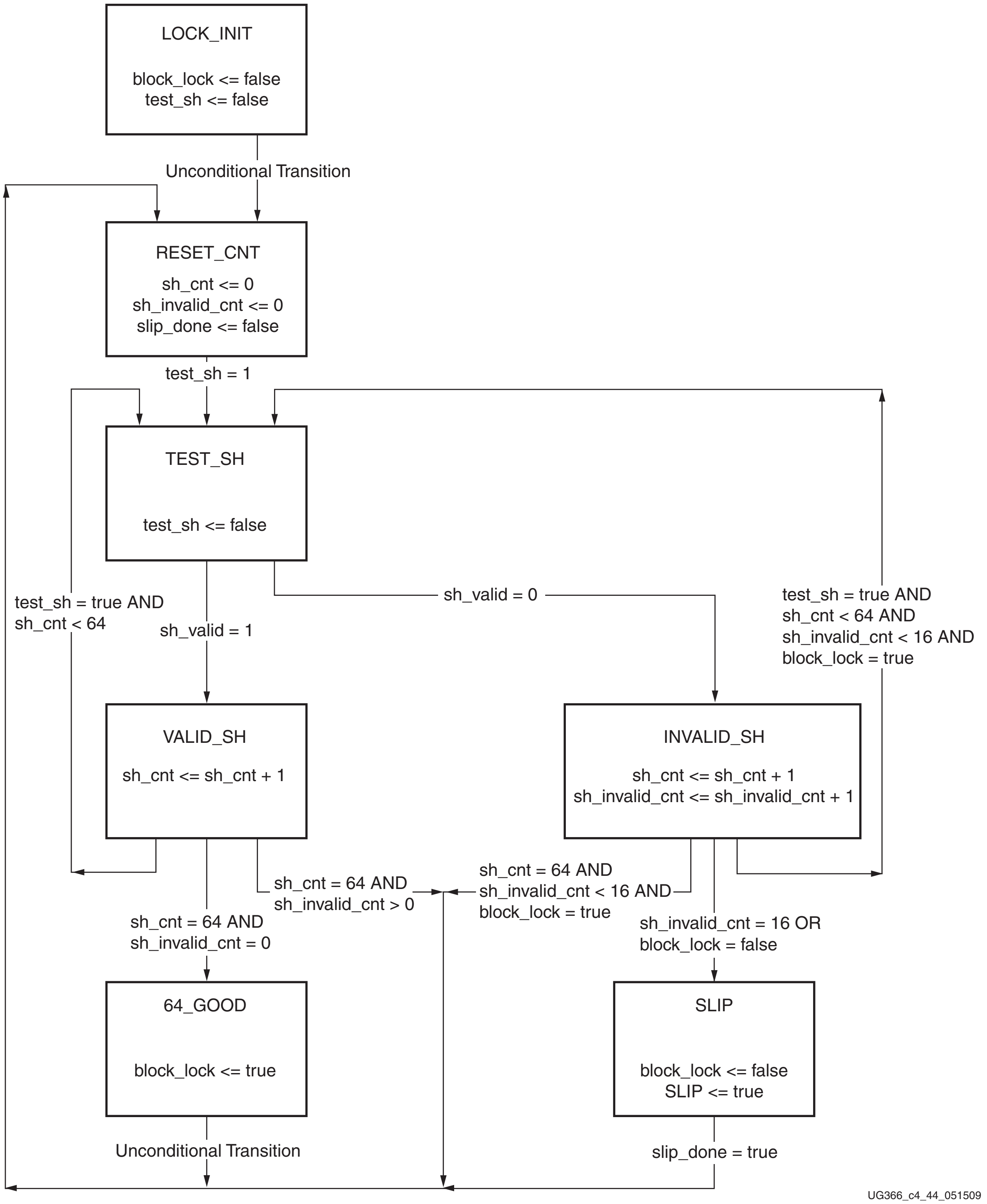


Figure 4-47: **Block Synchronization State Machine**

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 4-47, sh_cnt_max and sh_invalid_cnt_max are both constants that are set to 64 and 16, respectively. From the VALID_SH state, if sh_cnt is less than the value sh_cnt_max and test_sh is High, the next state is TEST_SH. If sh_cnt is equal to sh_cnt_max and sh_invalid_cnt equals 0, the next state is GOOD_64 and from there block_lock is asserted. Then the process repeats again and the counters are cleared to zeros. To achieve block lock, the state machine must receive sh_cnt_max number of valid synchronization headers in a row without getting an invalid synchronization header. However, when block lock is achieved, sh_invalid_cnt_max – 1 number of invalid synchronization headers can be received within sh_cnt_max number of valid synchronization headers. Thus, once locked, it is harder to break lock.

Figure 4-48 shows a waveform of the block synchronization state machine asserting RXGEARBOXSLIP numerous times because of invalid synchronization headers before achieving data alignment.

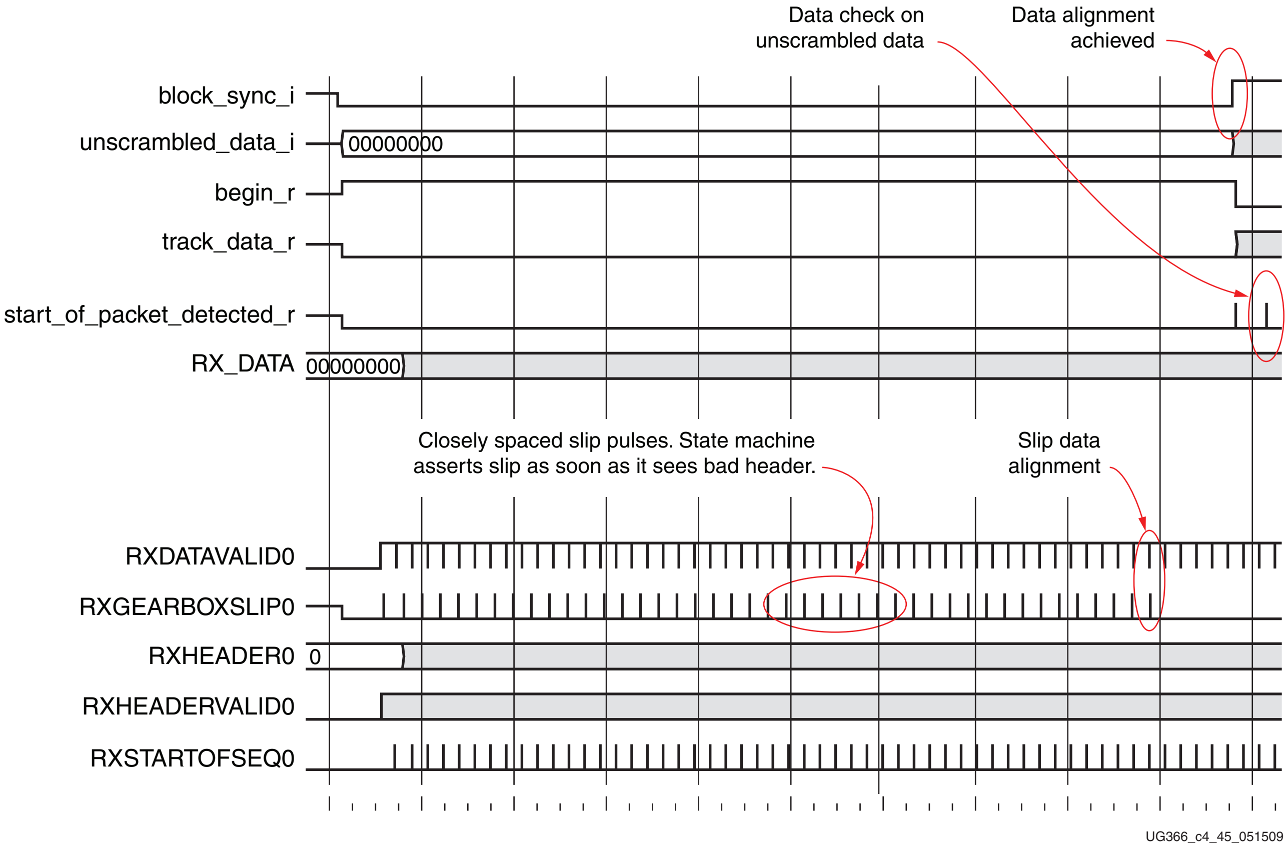


Figure 4-48: **RX Gearbox with Block Synchronization**

RX Initialization

Functional Description

The GTX transceiver RX must be reset before it can be used. There are three ways to reset the GTX transceiver RX:

1. Power up and configure the FPGA. Power-up reset is covered in this section.
2. Drive the GTXRXRESET port High to trigger a full asynchronous reset of the GTX transceiver RX.
3. Assert one or more of the individual reset signals on the block to reset a specific subcomponent of the receiver. These resets are covered in detail in the sections for each subcomponent (refer to Table 4-54, page 265).

All the reset ports described in this section initiate the internal receiver reset state machines when driven High. The internal reset state machines are held in the reset state until these same reset ports are driven Low. The completion of these state machines is signaled through the RXRESETDONE port.

Figure 4-49 shows the GTX transceiver RX reset hierarchy.

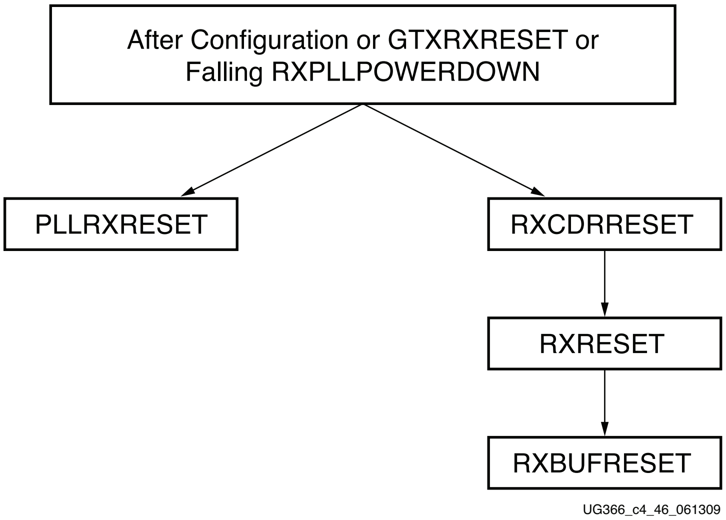


Figure 4-49: **GTX Transceiver RX Reset Hierarchy**

When bypassing the TX buffer or the RX buffer, GTXRXRESET and PLLRXRESET must not be tied High. Refer to TX Buffer Bypass, page 156 and RX Buffer Bypass, page 230 for more information.

Ports and Attributes

Table 4-52 defines the RX initialization ports.

Table 4-52: **RX Initialization Ports**

Port	Dir	Clock Domain	Description
GTXRXRESET	In	Async	This port is asserted High and then deasserted to start the full GTX transceiver RX reset sequence. This sequence takes about 120 µs to complete, and systematically resets all subcomponents of the GTX transceiver RX.
PLLRXRESET	In	Async	This port resets the RX PLL of the GTX transceiver RX.
PRBSCNTRESET	In	RXUSRCLK2	This port resets the PRBS error counter.
RXBUFRESET	In	Async	PCS RX elastic buffer reset. This active-High signal resets the RX elastic buffer logic; it resets the pointers in the buffer and flushes the data out of the buffer.
RXCDRRESET	In	Async	This port resets the CDR and the RX part of the PCS. This signal is driven High to cause the CDR to give up its current lock and return to the PMA PLL frequency. This port does not reset the PLL nor the PMA low latency clock alignment.
RXDLYALIGNRESET	In	Async	This port resets the RX delay aligner for the RX buffer bypass mode. See RX Buffer Bypass, page 230.
RXRESET	In	Async	PCS RX system reset. Resets receiver elastic buffer, 8B/10B decoder, comma detect and other receiver registers. This reset is a subset of GTXRXRESET and RXCDRRESET.
RXRESETDONE	Out	Async	This port goes High when the GTX transceiver RX has finished reset and is ready for use. For this signal to work correctly, the RX reference clock and all clock inputs on the individual GTX transceiver RX (RXUSRCLK, RXUSRCLK2) must be driven.
TSTIN[19:0]	In	Async	Reserved. Must be tied to `11111111111111111111`.

Table 4-53 defines the RX initialization attributes.

Table 4-53: **RX Initialization Attributes**

Attribute	Type	Description
CDR_PH_ADJ_TIME[4:0]	5-bit Binary	Sets time to wait after deassertion of CDR phase reset before completion of optional reset sequence for PCI Express operation during electrical idle.
RX_EN_IDLE_HOLD_CDR	Boolean	Enables CDR to hold its data during optional reset sequence for PCI Express operation during electrical idle. ***Note:*** For channels with large attenuation (lossy channels typically exceeding 15 dB at Nyquist), it is recommended that RX_EN_IDLE_HOLD_CDR should be set to FALSE because fast transitioning data patterns like the `101010` sequence in CJPAT/CJTPAT can accidentally trigger an electrical idle.
RX_EN_IDLE_HOLD_DFE	Boolean	Enables the ability to restore DFE contents from internal registers after termination of an electrical idle state for PCI Express operation. ***Note:*** For channels with large attenuation (lossy channels typically exceeding 15 dB at Nyquist), it is recommended that RX_EN_IDLE_HOLD_DFE should be set to FALSE because fast transitioning data patterns like the `101010` sequence in CJPAT/CJTPAT can accidentally trigger an electrical idle.

Table 4-53: **RX Initialization Attributes *(Cont'd)***

Attribute	Type	Description
RX_EN_IDLE_RESET_BUF	Boolean	Enables the reset of the receiver elastic buffer if a valid signal is not present on the serial RX inputs. Works in conjunction with attributes RX_IDLE_HI_CNT and RX_IDLE_LO_CNT. ***Note:*** For channels with large attenuation (lossy channels typically exceeding 15 dB at Nyquist), it is recommended that RX_EN_IDLE_RESET_BUF should be set to FALSE because fast transitioning data patterns like the `101010` sequence in CJPAT/CJTPAT can accidentally trigger an electrical idle.
RX_EN_IDLE_RESET_PH	Boolean	Enables the reset of the CDR phase circuits during optional reset sequence for PCI Express operation during electrical idle. ***Note:*** For channels with large attenuation (lossy channels typically exceeding 15 dB at Nyquist), it is recommended that RX_EN_IDLE_RESET_PH should be set to FALSE because fast transitioning data patterns like the `101010` sequence in CJPAT/CJTPAT can accidentally trigger an electrical idle.
RX_EN_IDLE_RESET_FR	Boolean	Enables the reset of the CDR frequency circuits during optional reset sequence for PCI Express operation during electrical idle. ***Note:*** For channels with large attenuation (lossy channels typically exceeding 15 dB at Nyquist), it is recommended that RX_EN_IDLE_RESET_FR should be set to FALSE because fast transitioning data patterns like the `101010` sequence in CJPAT/CJTPAT can accidentally trigger an electrical idle.
RX_EN_MODE_RESET_BUF	Boolean	Enables automatic reset of the RX elastic buffer when the RXCHBONDMASTER, RXCHBONDSLAVE or RXCHBONDLEVEL[2:0] ports change. See RX Channel Bonding, page 246.
RX_EN_RATE_RESET_BUF	Boolean	Enables automatic reset of the RX elastic buffer when the RXRATE[1:0] ports change.
RX_EN_REALIGN_RESET_BUF	Boolean	Enables automatic reset of the RX elastic buffer during comma alignment.
RX_EN_REALIGN_RESET_BUF2	Boolean	If this attribute is FALSE, the RX elastic buffer reset upon realignment occurs only when the received signal from RXP/RXN is determined valid and RXELECIDLE = `0`.
RX_IDLE_HI_CNT[3:0]	4-bit Binary	Programmable counters used in association with resetting the RX elastic buffer in response to the absence of valid data on the serial RX inputs. Determines how long the serial RX inputs must remain in electrical idle before the RX elastic buffer reset is asserted.
RX_IDLE_LO_CNT[3:0]	4-bit Binary	Programmable counters associated with deasserting the reset condition of the elastic buffer in response to the detection of valid data on the serial RX inputs. Determines how long the serial RX inputs must have good data (not be in electrical idle) before the RX elastic buffer reset is deasserted.

GTX Transceiver RX Reset in Response to Completion of Configuration

Figure 4-50 shows the GTX transceiver RX reset following the completion of configuration of a powered-up GTX transceiver. The same sequence is activated any time RXPLLPOWERDOWN goes from High to Low during normal operation.

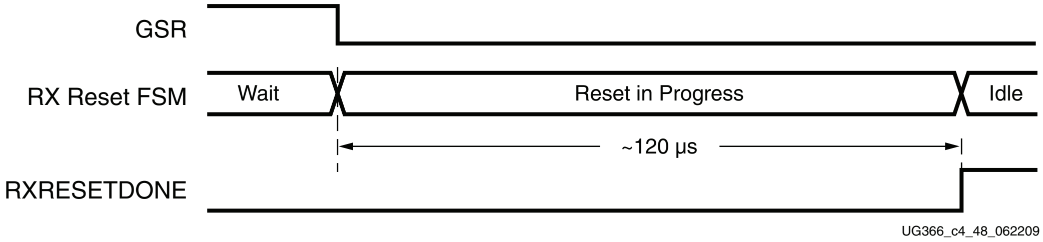


Figure 4-50: **Receiver Reset After Configuration**

Note: The timing of the reset sequencer inside the GTX transceiver receiver depends on the frequency of an internal clock and certain configuration attributes. The estimate given in Figure 4-50 assumes that the frequency of the internal clock is 50 MHz with default values for the configuration attributes.

GTX Transceiver RX Reset in Response to GTXRXRESET Pulse

Figure 4-51 is similar to Figure 4-50, showing the reset occurring in response to a pulse on GTXRXRESET. GTXRXRESET acts as an asynchronous reset signal. The guideline for the asynchronous GTXRXRESET pulse width is one period of the reference clock.

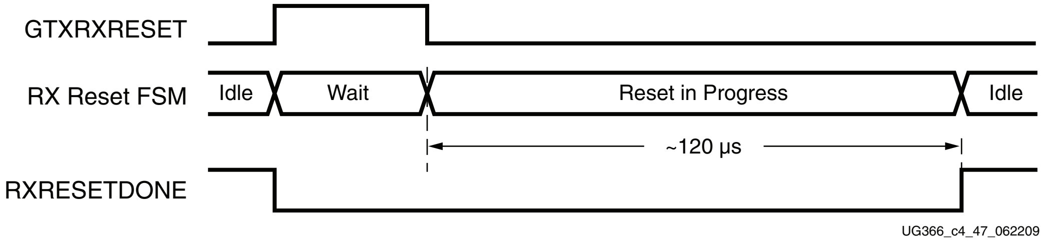


Figure 4-51: **Receiver Reset after GTXRXRESET Pulse**

Note: The timing of the reset sequencer inside the GTX transceiver RX depends on the frequency of an internal clock and certain configuration attributes. The estimate given in Figure 4-51 assumes that the frequency of the internal clock is 50 MHz with default values for the configuration attributes.

The entire GTX transceiver RX is affected by GTXRXRESET.

Link Idle Reset Support

The Link Idle Reset circuitry is built into the GTX transceiver receiver. The RX elastic buffer reset sequence during an electrical idle condition is available for additional functionality.

During an electrical idle condition, the CDR circuit in the receiver can lose lock (see RX CDR, page 203). To restart the CDR after an electrical idle condition, set the RX_EN_IDLE_RESET_PH, RX_EN_IDLE_RESET_FR, and RX_EN_IDLE_HOLD_CDR attributes to TRUE. The CDR_PH_ADJ_TIME[4:0] attribute sets the wait time before deassertion of the CDR phase reset and must be left at the default value.

The RX_EN_IDLE_RESET_BUF attribute enables a reset sequence for the GTX transceiver's RX elastic buffer. The RX elastic buffer of a GTX transceiver is automatically held in reset and reinitialized after the end of an electrical idle condition on the RX pin pair if the RX_EN_IDLE_RESET_BUF attribute is set to TRUE. The RX_IDLE_HI_CNT and RX_IDLE_LO_CNT attributes control the timing of the RX elastic buffer reset sequence and must be left at the default values.

Note: For channels with large attenuation (lossy channels typically exceeding 15 dB at Nyquist), it is recommended the RX_EN_IDLE_* reset attributes be set to FALSE because fast transitioning data patterns like the `101010` sequence in CJPAT/CJTPAT can accidentally trigger an electrical idle.

GTX Transceiver RX Component-Level Resets

GTX transceiver RX component resets are primarily used for special cases. These resets are needed when only the reset of a specific subsection is required. Each component-level reset signal is described in Table 4-54, page 265.

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

Table 4-54: **Available Receiver Resets and the Components Reset by Them**

	Component	**Configuration**	**GTXRXRESET**	**RXPLLPOWERDOWN (Falling Edge)**	**PLLRXRESET**	**RXCDRRESET**	**RXRESET**	**RXBUFRESET**	**PRBSCNTRESET**	**RXDLYALIGNRESET**
RX PCS	FPGA RX Interface		*	*		*	*			
	RX Gearbox	*	*	*		*	*			
	RX Status Control	*	*	*		*	*			
	RX Elastic Buffer	*	*	*		*	*	*		
	8B/10B Decoder	*	*	*		*	*			
	Comma Detect and Align	*	*	*		*	*			
	RX LOS State Machine	*	*	*		*	*			
	RX Polarity	*	*	*		*	*			
	PRBS Checker	*	*	*		*	*		*	
	5X Oversampler	*	*	*		*	*			
	RX Delay Aligner	*								*
RX PMA	RX Analog Front End	*	*	*						
	RX OOB	*	*	*		*				
	RX Equalizer	*	*	*						
	RX PLL	*	*	*	*					
	RX CDR	*	*	*		*				
	SIPO	*	*	*		*				
Loopback	Loopback Paths	*	*	*						

Table 4-55 lists the recommended resets for various situations.

Table 4-55: **Recommended Resets for Common Situations**

Situation	Components to be Reset	Recommended Reset[1]
After power up and configuration	Entire GTX transceiver RX	After configuration, the GTX transceiver RX is reset automatically
After turning on a reference clock to RX PLL	Entire GTX transceiver RX	GTXRXRESET
After changing the reference clock to RX PLL	Entire GTX transceiver RX	GTXRXRESET
After assertion/deassertion of RXPOWERDOWN	Entire GTX transceiver RX	GTXRXRESET
RX rate change with RX elastic buffer bypassed	RX PCS, RX Phase Alignment	RXRESET
RX rate change with RX elastic buffer enabled	RX PCS	RX RESET
RX parallel clock source reset	RX PCS, RX Phase Alignment	RXRESET
After remote power up	RX CDR	A built-in reset sequencer automatically sets these situations by setting RX_EN_IDLE_RESET_PH, RX_EN_IDLE_RESET_FR, RX_EN_IDLE_HOLD_CDR to TRUE
Electrical idle reset	RX CDR	
After connecting RXN/RXP[2]	RX CDR	
After an RXBUFFER error	RX Elastic Buffer	RXBUFRESET
Before channel bonding	RX CDR, then RXBUFFER after CDR is locked	Either assert RXBUFRESET, or automatically reset by setting RX_EN_IDLE_RESET_BUF = TRUE to enable the RXBUFRESET sequence
After changing channel bonding mode on the fly	RX Elastic Buffer	RX elastic buffer is reset automatically after change in channel bonding mode by setting RX_EN_MODE_RESET_BUF to TRUE
After PRBS error	PRBS Error Counter	PRBSCNTRESET
After an oversampler error	Oversampler	RXRESET
After comma realignment	RX Elastic Buffer (optional)	RX elastic buffer is reset automatically after comma realignment by setting RX_EN_REALIGN_RESET_BUF 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 transceiver RX is reset automatically after the configuration-provided RXPLLPOWERDOWN is Low. The supplies for the calibration resistor and calibration resistor reference must be powered up before configuration to ensure correct calibration of the termination impedance of all transceivers.

After Turning on a Reference Clock to RX PLL

The reference clock source(s) and the power to the GTX transceiver(s) must be available before configuring the FPGA. The reference clock must be stable before configuration especially when using PLL-based clock sources (e.g., voltage controlled crystal oscillators). If the reference clock(s) or GTX transceiver(s) are powered up after configuration, GTXRXRESET must be asserted to allow the RX PLL(s) to lock.

After Changing the Reference Clock to RX PLL

Whenever the reference clock input to GTX transceiver RX PLL is changed, the RX PLL must be reset afterwards to ensure that it locks to the new frequency. The GTXRXRESET port must be used for this purpose. Refer to Reference Clock Selection, page 102 for more details.

After Assertion/Deassertion of RXPOWERDOWN

After deassertion of the RXPOWERDOWN signal, GTXRXRESET must be asserted.

RX Rate Change with RX Elastic Buffer Enabled

After the completion of a rate change initiated via the RXRATE[1:0] port, the RX PCS must be reset using RXRESET. To automatically reset the RX elastic buffer after a rate change, RX_EN_RATE_RESET_BUF must be set to TRUE.

RX Rate Change with RX Elastic Buffer Bypassed

After the completion of a rate change initiated via the RXRATE[1:0] port, the RX PCS must be reset using RXRESET. Phase alignment must be performed again. RXRATEDONE or PHYSTATUS can be used to detect the completion of the rate change (see RX Fabric Clock Output Control, page 206). See RX Buffer Bypass, page 230 for details on the phase alignment procedure.

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, RXRESET must be used to hold the RX PCS in reset until the clock source is locked again.

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

See RX Buffer Bypass, page 230 for details on the phase alignment procedure.

After Remote Power-Up

If the incoming data becomes available after the GTX transceiver receiver is powered up, the RX CDR must be reset to ensure a clean lock to the incoming data. When the guidelines in Link Idle Reset Support, page 263 are followed, the electrical idle reset situation is automatically managed.

Electrical Idle Reset

When the differential voltage of the RX input to a GTX transceiver drops to OOB or electrical idle levels, the RX CDR can be pulled out of lock by the apparent sudden change in frequency. When the guidelines in Link Idle Reset Support, page 263 are followed, the electrical idle reset situation is automatically managed.

After Connecting RXN/RXP

When the RX data to the GTX transceiver comes from a connector that can be plugged in and unplugged, the RX CDR must be reset when the data source is plugged in to ensure that it can lock to incoming data. When the guidelines in Link Idle Reset Support, page 263 are followed, the electrical idle reset situation is automatically managed.

After an RX Elastic Buffer Error

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

Before Channel Bonding

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

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

To provide the same recovered clock to all bonded transceivers:

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

The required reset for channel bonding is:

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

See RX CDR, page 203 for recommendations on how to detect CDR lock.

Note: For channels with large attenuation (lossy channels typically exceeding 15 dB at Nyquist), it is recommended that RX_EN_IDLE_* reset attributes be set to FALSE because fast transitioning data patterns like the `101010` sequence in CJPAT/CJTPAT can accidentally trigger an electrical idle.

After Changing Channel Bonding Mode on the Fly

When set to TRUE, RX_EN_MODE_RESET_BUF enables automatic reset of the RX elastic buffer when the RXCHANBONDMASTER, RXCHANBONDSLAVE, or RXCHANBONDLEVEL ports change. See RX Channel Bonding, page 246.

After a PRBS Error

PRBSCNTRESET is asserted to reset the PRBS error counter.

After an Oversampler Error

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

After Comma Realignment

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

FPGA RX Interface

Functional Description

The FPGA receives RX data from the GTX transceiver RX through the FPGA RX interface. Data is read from the RXDATA port on the positive edge of RXUSRCLK2. RXDATA can be configured to be one, two, or four bytes wide. The actual width of the port depends on the RX_DATA_WIDTH attribute and RXDEC8B10BUSE port settings. Port widths can be 8, 10, 16, 20, 32, and 40 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 receiver. This section shows how to drive the parallel clocks and explains the constraints on those clocks for correct operation. The highest receiver data rates require a 4-byte interface to achieve an RXUSRCLK2 rate in the specified operating range.

Interface Width Configuration

The Virtex-6 FPGA GTX transceiver contains an internal 2-byte datapath. The FPGA interface width is configurable by setting the RX_DATA_WIDTH attribute. When the 8B/10B decoder is enabled, the FPGA interface must be configured to 10, 20, or 40 bits. When the 8B/10B decoder is bypassed, the FPGA interface is configured to any of the available widths: 8, 10, 16, 20, 32, and 40 bits. Table 4-56 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 227.

Table 4-56: **FPGA RX Interface Datapath Configuration**

RXDEC8B10BUSE	RX_DATA_WIDTH	FPGA Interface Width	Internal Data Width
1	10	8 bits	20 bits
	20	16 bits	20 bits
	40	32 bits	20 bits

Table 4-56: **FPGA RX Interface Datapath Configuration *(Cont'd)***

RXDEC8B10BUSE	RX_DATA_WIDTH	FPGA Interface Width	Internal Data Width
0	8	8 bits	16 bits
	10	10 bits	20 bits
	16	16 bits	16 bits
	20	20 bits	20 bits
	32	32 bits	16 bits
	40	40 bits	20 bits

When the 8B/10B decoder is bypassed and the RX_DATA_WIDTH is 10, 20, or 40, the RXDISPERR and RXCHARISK ports are used to extend the RXDATA port from 8 to 10 bits, 16 to 20 bits, or 32 to 40 bits. Table 4-57 shows the data received when the 8B/10B decoder is disabled. When RX gearbox is used, data reception order is left to right.

Table 4-57: **RX Data Received with 8B/10B Decoder Bypassed**

	< < Data reception order is right to left < <																																							
	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 Received	RXDISPERR[3]	RXCHARISK[3]	RXDATA[31:24]								RXDISPERR[2]	RXCHARISK[2]	RXDATA[23:16]								RXDISPERR[1]	RXCHARISK[1]	RXDATA[15:8]								RXDISPERR[0]	RXCHARISK[0]	RXDATA[7:0]							

RXUSRCLK and RXUSRCLK2 Generation

The FPGA RX interface includes two parallel clocks: RXUSRCLK and RXUSRCLK2. RXUSRCLK is the internal clock for the PCS logic in the GTX transceiver receiver. The required rate for RXUSRCLK depends on the internal datapath width of the `GTXE1` primitive and the RX line rate of the GTX transceiver receiver. Equation 4-1 shows how to calculate the required rate for RXUSRCLK.

$$RXUSRCLK\ Rate\ =\ \frac{Line\ Rate}{Internal\ Datapath\ Width}$$

Equation 4-1

RXUSRCLK can be generated internally to the GTX transceiver. This functionality is controlled by the GEN_RXUSRCLK attribute. Table 4-58 describes the situations in which RXUSRCLK can be generated internally by the GTX transceiver. In these cases, the RXUSRCLK port must be tied Low.

Table 4-58: **RXUSRCLK Internal Generation Configurations**

	RX_DATA_WIDTH	GTX Lanes in Channel[(1)]	GEN_RXUSRCLK
1-Byte	8, 10	1	TRUE
		2 or more	FALSE
2-Byte	16, 20	1 or more	TRUE

Table 4-58: **RXUSRCLK Internal Generation Configurations**

	RX_DATA_WIDTH	GTX Lanes in Channel[1]	GEN_RXUSRCLK
4-Byte	32, 40	1 or more	FALSE

Notes:

1. For single lane protocols such as 1 Gb/s Ethernet, "GTX Lanes in Channel" is 1. For multiple lane protocols like XAUI, "GTX Lanes in Channel" is 2 or more.

RXUSRCLK2 is the main synchronization clock for all signals into the RX side of the GTX transceiver. Most signals into the RX side of the GTX transceiver are sampled on the positive edge of RXUSRCLK2. RXUSRCLK2 and RXUSRCLK have a fixed-rate relationship based on the RX_DATA_WIDTH setting. Table 4-59 shows the relationship between RXUSRCLK2 and RXUSRCLK per RX_DATA_WIDTH values.

Table 4-59: **RXUSRCLK2 Frequency Relationship to RXUSRCLK**

	RX_DATA_WIDTH	RXUSRCLK2 Frequency
1-Byte	8, 10	$F_{RXUSRCLK2} = 2 \times F_{RXUSRCLK}$
2-Byte	16, 20	$F_{RXUSRCLK2} = F_{RXUSRCLK}$
4-Byte	32, 40	$F_{RXUSRCLK2} = F_{RXUSRCLK} / 2$

The following 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. Table 4-58 and Table 4-59 describe the appropriate GEN_RXUSRCLK setting and RXUSRCLK2 frequency requirements. In cases where RXUSRCLK is generated by the user, the designer must ensure that RXUSRCLK and RXUSRCLK2 are positive-edge aligned.
- 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 RXRECCLK, and the phase-alignment circuit must be used.
- If clock correction is used, RXUSRCLK and RXUSRCLK2 can be sourced by RXRECCLK or TXOUTCLK.

For details about placement constraints and restrictions on clocking resources (MMCM, BUFGCTRL, IBUFDS_GTXE1, BUFG, etc.), refer to the *Virtex-6 FPGA Clocking Resources User Guide*.

Ports and Attributes

Table 4-60 defines the FPGA RX ports.

Table 4-60: **FPGA RX Ports**

Port	Dir	Clock Domain	Description
MGTREFCLKFAB[1:0]	Out	Clock	Reserved. Do not use this port.
RXCHARISK[3:0]	Out	RXUSRCLK2	When 8B/10B decoding is disabled, RXCHARISK is used to extend the data bus for 10- and 20-bit RX interfaces.

RXDATA[31:0]	Out	RXUSRCLK2	The bus for transmitting data. The width of this port depends on RX_DATA_WIDTH:
			RX_DATA_WIDTH = 8, 10: RXDATA[7:0] = 8 bits wide
			RX_DATA_WIDTH = 16, 20: RXDATA[15:0] = 16 bits wide
			RX_DATA_WIDTH = 32, 40: RXDATA[31:0] = 32 bits wide

RXDISPERR[3:0]	Out	RXUSRCLK2	When 8B/10B decoding is disabled, RXDISPERR is used to extend the data bus for 10- and 20-bit RX interfaces.
RXUSRCLK	In	Clock	This port provides a clock for the internal RX PCS datapath. In some use cases, this clock is internally generated. See Table 4-58.
RXUSRCLK2	In	Clock	This port synchronizes the FPGA logic with the RX interface. This clock must be positive-edge aligned to RXUSRCLK when RXUSRCLK is provided by the user.

Table 4-61 defines the FPGA RX attributes.

Table 4-61: **FPGA RX Attributes**

Attribute	Type	Description
GEN_RXUSRCLK	Boolean	Controls internal generation of RXUSRCLK available in certain modes of operation. See RXUSRCLK and RXUSRCLK2 Generation, page 270. TRUE: RXUSRCLK internally generated. RXUSRCLK must be tied Low. FALSE: RXUSRCLK must be provided by user.
RX_DATA_WIDTH	Integer	Sets the bit width of the RXDATA port. When 8B/10B decoding is enabled, RX_DATA _WIDTH must be set to 10, 20, or 40. Valid settings are 8, 10, 16, 20, 32, and 40. See Interface Width Configuration, page 269 for more details.

Chapter 5

Board Design Guidelines

Overview

This chapter discusses topics related to implementing a design that uses the Virtex®-6 FPGA GTX transceiver on a printed circuit board (PCB). The GTX transceivers are analog circuits that require special consideration and attention when designing and implementing them on a PCB. For a design to perform optimally, the designer requires an understanding of the functionality of the device pins and needs to attend 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 Transceiver Pin Descriptions

Table 5-1 defines the pins in a Quad.

Table 5-1: **Quad Pin Descriptions**

Pin	Dir	Description
MGTAVCC_N MGTAVCC_S	In (Pad)	MGTAVCC is the analog supply for the internal analog circuits of the Quad. This includes the analog circuits for the PLLs, transmitters, and receivers. Most packages have a north and south bank in the package for MGTAVCC. Refer to the package pin definitions to identify the power supply bank that is associated with the specific Quad. The nominal voltage is 1.0 V_{DC}.
MGTAVTT_N MGTAVTT_S	In (Pad)	MGTAVTT is the analog supply for the transmitter and receiver termination circuits of the Quad. Most packages have a north and south bank in the package for MGTAVTT. Refer to the package pin definitions to identify the power supply bank that is associated with the specific Quad. The nominal voltage is 1.2 V_{DC}.
MGTAVTTRCAL	In (Pad)	Bias current supply for the termination resistor calibration circuit. See Termination Resistor Calibration Circuit, page 274 for more information.
MGTREFCLK0P MGTREFCLK0N	In (Pad)	Differential clock input pin pair for the reference clock of the Quad.
MGTREFCLK1P MGTREFCLK1N	In (Pad)	Differential clock input pin pair for the reference clock of the Quad.
MGTRREF	In (Pad)	Calibration resistor input pin for the termination resistor calibration circuit. See Termination Resistor Calibration Circuit, page 274 for more information.

Table 5-1: **Quad Pin Descriptions *(Cont'd)***

Pin	Dir	Description
MGTRXP0/MGTRXN0 MGTRXP1/MGTRXN1 MGTRXP2/MGTRXN2 MGTRXP3/MGTRXN3	In (Pad)	RXP and RXN are the differential input pairs for each of the receivers in the Quad.
MGTTXP0/MGTTXN0 MGTTXP1/MGTTXN1 MGTTXP2/MGTTXN2 MGTTXP3/MGTTXN3	Out (Pad)	TXP and TXN are the differential output pairs for each of the transmitters in the Quad.

Figure 5-1 shows the connections of the power supply pins for the GTX transceiver. The listed voltages are nominal values. Refer to the *Virtex-6 FPGA Data Sheet* for values and operating conditions.

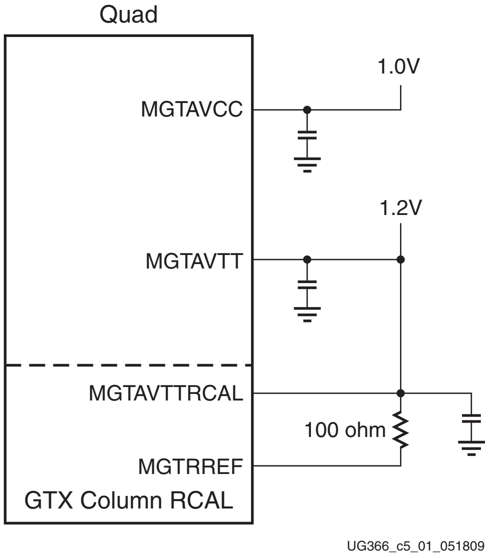


Figure 5-1: **Virtex-6 FPGA GTX Transceiver Power Supply Connections**

Termination Resistor Calibration Circuit

One resistor calibration circuit (RCAL) is shared between all of the Quad primitives in a Quad column (see Figure 5-2). The MGTAVTTRCAL and MGTRREF pins are used to 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 *Virtex-6 FPGA Data Sheet*.

The RCAL circuit is associated with the MGT115 Quad. It is referred to as 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 Quads in the column.

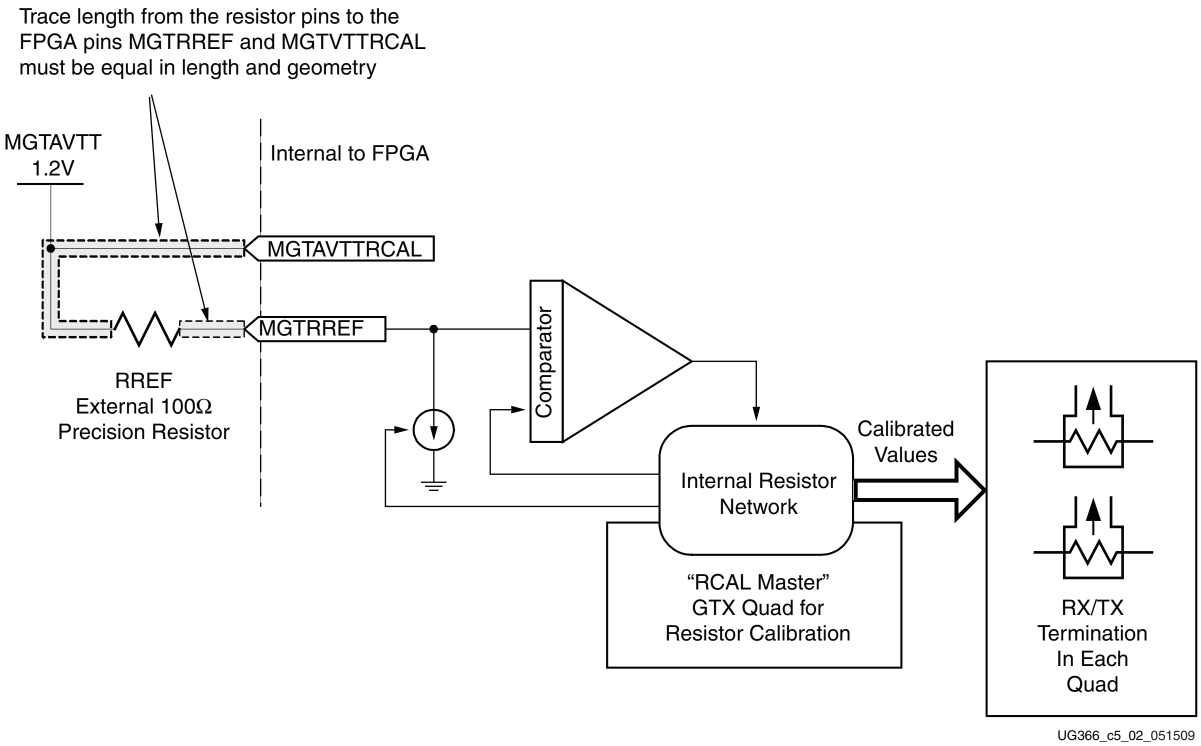

Trace length from the resistor pins to the FPGA pins MGTRREF and MGTVTTRCAL must be equal in length and geometry
MGTAVTT
1.2V
Internal to FPGA
MGTAVTTRCAL
MGTRREF
RREF
External 100Ω
Precision Resistor
Comparator
Internal Resistor Network
Calibrated Values
"RCAL Master" GTX Quad for Resistor Calibration
RX/TX Termination In Each Quad
UG366_c5_02_051509

Figure 5-2: **Termination Resistor Calibration Circuit**

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 that is 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 depends 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. Figure 5-3 shows a recommended layout.

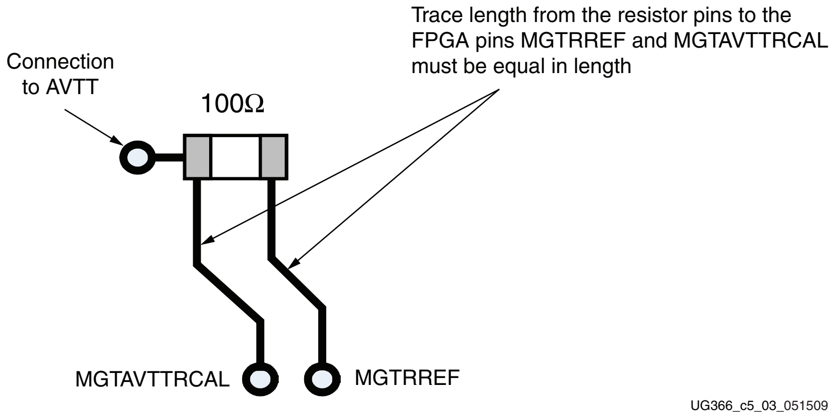

Trace length from the resistor pins to the FPGA pins MGTRREF and MGTAVTTRCAL must be equal in length
Connection to AVTT
100Ω
MGTAVTTRCAL
MGTRREF
UG366_c5_03_051509

Figure 5-3: **PCB Layout for the RCAL Resistor**

Managing Unused GTX Transceivers

In many applications, only a portion of the GTX transceivers are required. There are considerations for managing the unused GTX transceivers that affect such things as power consumption of the Virtex-6 FPGA. When considering which Quads to use in an application, the organization of the package power planes needs to be taken into account. Power can be utilized efficiently when there are multiple analog power planes in the package. If only a small number of the Quads are to be used, it might be possible to leave some of them completely unpowered.

The criteria for powering a GTX Quad is based on whether the GTX transceivers will be permanently unused or whether they will be used at a later date. If a GTX transceiver is ever intended to be used, it must be powered whenever the FPGA is powered up. If the GTX transceiver will never be used, the transceiver bank can be unpowered. Connections to unused GTX transceivers (including power supplies) are covered in the following sections.

Analog Power Supply Pins for Virtex-6 LXT Devices

For Virtex-6 LXT FPGAs, the Quad analog power supply pins, MGTAVCC and MGTAVTT, are connected to planes inside the package. Some packages have two planes for each of these analog power supplies, designated as north and south. Therefore, the Quads in a column are grouped by their common analog power supply connections into banks. There are north and south banks of Quads. For each of the analog power supplies, pins for all of the north Quads are connected to the same plane inside the package. Similarly, for each of the analog power supplies, the pins for the south bank of Quads are connected to the same plane inside the package. As a result, there are four Quad analog power supply planes inside the package. Inside the package are two MGTAVCC power planes and two MGTAVTT power planes. Refer to Table 5-2 for a description of the Virtex-6 LXT FPGA connected Quads.

In Table 5-2, there are two types of groupings of the Quads in a column:

- Devices with two banks of Quads in a column:
  - North bank of Quads
  - South bank of Quads
- Devices with one bank of Quads in a column

Table 5-2: **Connected Quads**

Device	Quad[1]								
	MGT110	MGT111	MGT112	MGT113	MGT114	MGT115	MGT116	MGT117	MGT118
XC6VLX75T-FF484[2]					Common[3]	Common			
XC6VLX75T-FF784[2]					Common	Common	Common		
XC6VLX130T-FF484[2]					Common	Common			
XC6VLX130T-FF784[2]					Common	Common	Common		
XC6VLX130T-FF1156			South[4,6]	South	South	North[5,6]	North		
XC6VLX195T-FF784[2]					Common	Common	Common		
XC6VLX195T-FF1156			South	South	South	North	North		
XC6VLX240T-FF784[2]					Common	Common	Common		
XC6VLX240T-FF1156			South	South	South	North	North		
XC6VLX240T-FF1759			South	South	South	North	North	North	
XC6VLX365T-FF1156			South	South	South	North	North		
XC6VLX365T-FF1759			South	South	South	North	North	North	
XC6VLX550T-FF1759	South	South	South	South	South	North	North	North	North
XC6VSX315T-FF1156			South	South	South	North	North		

Table 5-2: **Connected Quads *(Cont'd)***

Device	Quad[1]								
	MGT110	MGT111	MGT112	MGT113	MGT114	MGT115	MGT116	MGT117	MGT118
XC6VSX315T-FF1759			South	South	South	North	North	North	
XC6VSX475T-FF1156			South	South	South	North	North		
XC6VSX475T-FF1759	South	South	South	South	South	North	North	North	North

Notes:

1. Each Quad contains four transceivers.
2. FF484 and FF784 packages only have one common internal power plane for each of the Quad analog power supplies.
3. Common: The Quad is powered by a common set of power planes (MGTAVCC, MGTAVTT).
4. South: The Quad is powered by South package power planes (MGTAVCC_S, MGTAVTT_S).
5. North: The Quad is powered by North package power planes (MGTAVCC_N, MGTAVTT_N).
6. North and South represent the two planes for each of the analog power supplies, MGTAVCC and MGTAVTT.

Figure 5-4 shows the orientation in the device of the Quad power banks within a device. The type of grouping for the Quads in a column depends on the device package. The FF484 and FF784 packages have a single common power plane for each of the GTX transceiver supplies. The FF1156 and FF1759 packages have two power planes for each of the GTX transceiver power supplies.

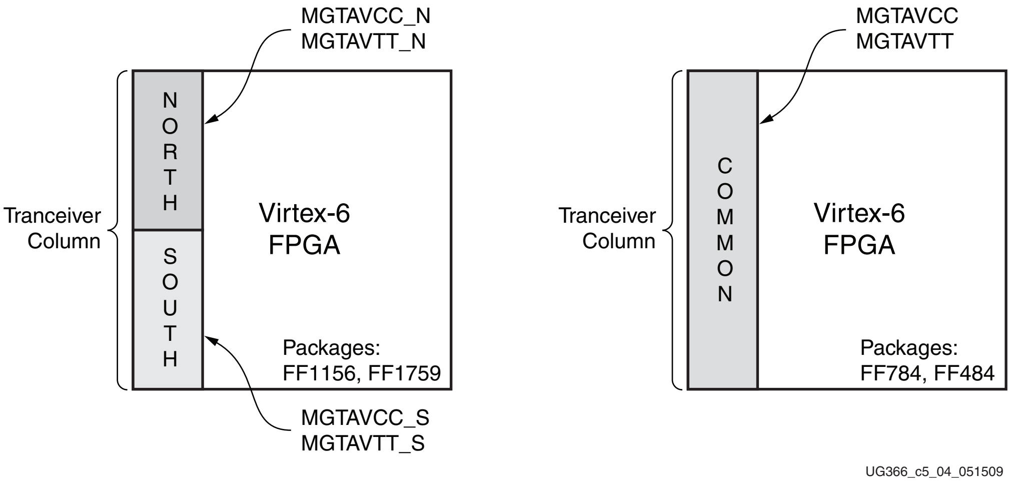


Figure 5-4: **GTX Transceiver Power Plane Bank Orientation in Virtex-6 FPGA Packages**

Analog Power Supply Pins for Virtex-6 HXT Devices

Virtex-6 HXT FPGAs contain Virtex-6 GTX transceivers and GTH transceivers. Detailed information on using the GTH transceivers can be found in UG371, *Virtex-6 FPGA GTH Transceivers User Guide*. The GTX and GTH transceivers are oriented in columns on one or both sides of the device. As in the case of LXT devices, the GTX transceiver QUADs are organized into power supply groups with each group having a common set of power supply planes. Table 5-3 shows the connections of the transceivers in the Virtex-6 FPGAs.

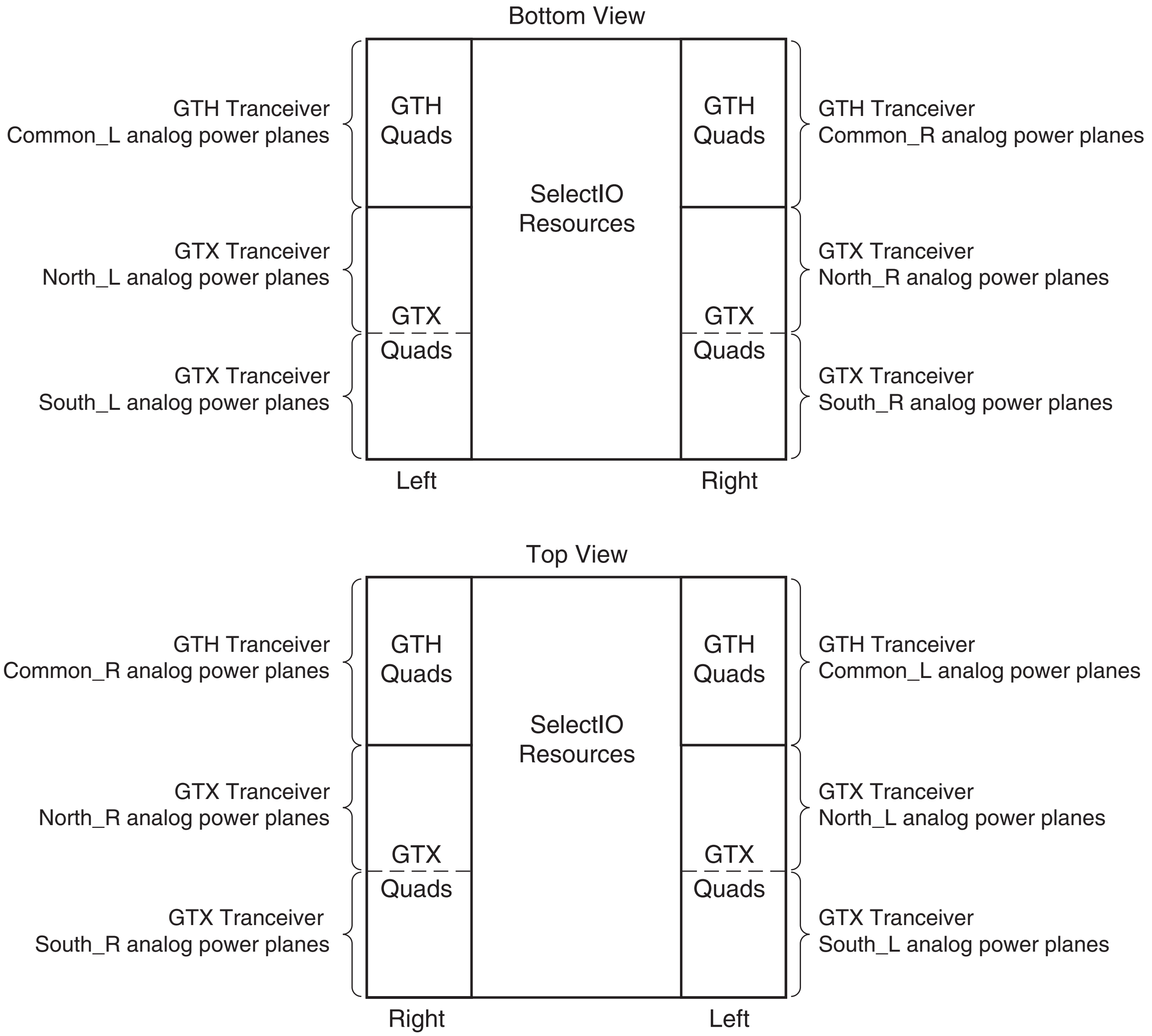
Table 5-3: **Virtex-6 FPGAs GTX and GTH Transceivers Grouped by Devices, Packages, and Power Planes**

Device/ Package	GTX Transceivers							GTH Transceivers		
	Left Side	**MGT100**	**MGT101**	**MGT102**	**MGT103**	**MGT104**	**MGT105**	**MGT106**	**MGT107**	**MGT108**
	Right Side	**MGT110**	**MGT111**	**MGT112**	**MGT113**	**MGT114**	**MGT115**	**MGT116**	**MGT117**	**MGT118**
HX250T-FF1154	Left Side	South_L	South_L	South_L	North_L	North_L	North_L			
	Right side	South_R	South_R	South_R	North_N	North_N	North_N			
HX255T-FF1155	Left Side				North_L	North_L	North_L			
	Right side				North_R	North_R	North_R	Common_R	Common_R	Common_R
HX255T-FF1923	Left Side				North_L	North_L	North_L	Common_L	Common_L	Common_L
	Right side				North_R	North_R	North_R	Common_R	Common_R	Common_R
HX380T-FF1154	Left Side	South_L	South_L	South_L	North_L	North_L	North_L			
	Right side	South_R	South_R	South_R	North_R	North_R	North_R			
HX380T-FF1155	Left Side				North_L	North_L	North_L			
	Right side				North_R	North_R	North_R	Common_R	Common_R	Common_R
HX380T-FF1923	Left Side	South_L	South_L	South_L	North_L	North_L	North_L	Common_L	Common_L	Common_L
	Right side			North_R	North_R	North_R	North_R	Common_R	Common_R	Common_R
HX380T-FF1924	Left Side	South_L	South_L	South_L	North_L	North_L	North_L	Common_L	Common_L	Common_L
	Right side	South_R	South_R	South_R	North_R	North_R	North_R	Common_R	Common_R	Common_R
HX565T-FF1923	Left Side	South_L	South_L	South_L	North_L	North_L	North_L	Common_L	Common_L	Common_L
	Right side			North_R	North_R	North_R	North_R	Common_R	Common_R	Common_R
HX565T-FF1924	Left Side	South_L	South_L	South_L	North_L	North_L	North_L	Common_L	Common_L	Common_L
	Right side	South_L	South_L	South_L	North_R	North_R	North_R	Common_R	Common_R	Common_R

Notes:

1. Right and left are oriented to the bottom view of the package.
2. South_L: South bank GTX transceiver power planes on left side of package.
3. North_L: North bank GTX transceiver power planes on left side of package.
4. South_R: South bank GTX transceiver power planes on right side of package.
5. North_R: North bank GTX transceiver power planes on right side of package.
6. Common_L: GTH transceiver power planes on left side of package.
7. Common_R: GTH transceiver power planes on right side of package.

Figure 5-5 shows the relative orientation of the transceivers (MGT100, MGT101, etc.) package power supply planes in the Virtex-6 HXT FPGA.


Bottom View
GTH Tranceiver Common_L analog power planes
GTX Tranceiver North_L analog power planes
GTX Tranceiver South_L analog power planes
GTH Quads
GTX Quads
SelectIO Resources
GTH Quads
GTX Quads
GTH Tranceiver Common_R analog power planes
GTX Tranceiver North_R analog power planes
GTX Tranceiver South_R analog power planes
Left
Right
Top View
GTH Tranceiver Common_R analog power planes
GTX Tranceiver North_R analog power planes
GTX Tranceiver South_R analog power planes
GTH Quads
GTX Quads
SelectIO Resources
GTH Quads
GTX Quads
GTH Tranceiver Common_L analog power planes
GTX Tranceiver North_L analog power planes
GTX Tranceiver South_L analog power planes
Right
Left

UG366_c5_20_062411

Figure 5-5: **GTX and GTH Transceivers Power Supply Bank Orientation in Virtex-6 HXT FPGA Devices**

Unused Quad Column

If none of the Quads in a column is used in the application, the Quad device pins can be connected as shown in Table 5-4.

Table 5-4: **Unused Quad Column Connections**

Quad Pin or Pin Pair of the Unused Column	Connection
MGTAVCC	GND
MGTAVTT	GND
MGTREFCLKP/MGTREFCLKN	FLOAT

Table 5-4: **Unused Quad Column Connections *(Cont'd)***

Quad Pin or Pin Pair of the Unused Column	Connection
MGTRXP/MGTRXN	GND
MGTTXP/MGTTXN	FLOAT
MGTAVTTRCAL(1)	GND
MGTRREF(1)	GND

Notes:

1. This is the only scenario when the RCAL pins can be connected to ground. In all other scenarios, these pins must be connected for normal operation.
2. The GTX transceiver banks can only be left unpowered if the transceivers in the bank will never be used, or if the transceivers in the bank will never bypass the TX or RX buffers. Refer to TX Buffer Bypass, page 156 and RX Buffer Bypass, page 230 for more information.

Partially Unused Quad Column

If only a portion of the Quads is used, there are two possible scenarios that affect how the Quad pins can be connected.

For the first scenario, if the Virtex-6 FPGA is one of the devices with a north bank and a south bank of Quads and none of the devices in the south bank are used in the application, the unused Quad pins should be connected as shown in Table 5-5. The analog power supply pins for the south bank are connected to ground. In this scenario, grounding the analog power supply pins applies only to the south bank of Quads because none of the Quads in the south group are used.

Table 5-5: **Unused Quad Pin Connections for a Partially Unused Column**

Pin or Pin Pair of the Unused Quads	Bank	Connection
MGTAVCC	North	MGTAVCC
MGTAVTT	North	MGTAVTT
MGTREFCLKP/MGTREFCLKN	North	FLOAT
MGTRXP/MGTRXN	North	GND
MGTTXP/MGTTXN	North	FLOAT
MGTAVTTRCAL	North	AVTT and RCAL precision resistor
MGTRREF	North	RCAL precision resistor
MGTAVCC(1)	South	GND
MGTAVTT(1)	South	GND
MGTREFCLKP/MGTREFCLKN	South	FLOAT
MGTRXP/MGTRXN	South	GND
MGTTXP/MGTTXN	South	FLOAT

Notes:

1. Connection if all Quads in the south bank are not used.
2. The south bank can only be left unpowered if the transceivers in the south bank will never be used, or if the transceivers in the south bank will never bypass the TX or RX buffers. Refer to TX Buffer Bypass, page 156 and RX Buffer Bypass, page 230 for more information.

For the second scenario, Table 5-6 shows the connections to the unused Quads when no Quad bank is completely unused.

Note: The analog power supplies must be connected.

Table 5-6: **Partially Unused Quad Bank**

Pin or Pin Pair of the Unused Quads	Connection
MGTAVCC	AVCC
MGTAVTT	AVTT
MGTREFCLKP/MGTREFCLKN	FLOAT
MGTRXP/MGTRXN	GND
MGTTXP/MGTTXN	FLOAT
MGTAVTTRCAL	AVTT and RCAL precision resistor
MGTRREF	RCAL precision resistor

Partially Used Quad

There are four GTX transceivers in a Quad. For a partially used Quad where one or more transceivers (but not all) are unused, the analog power supplies (MGTAVCC and MGTAVTT) need to be connected. Table 5-7 shows the connections for the GTX transceivers in the Quad that are not used.

Table 5-7: **Partially Used Quad Connections**

Pin or Pin Pair of the Unused Column	Connection
MGTAVCC	AVCC
MGTAVTT	AVTT
MGTREFCLKP/MGTREFCLKN	FLOAT
MGTRXP/MGTRXN	GND
MGTTXP/MGTTXN	FLOAT
MGTAVTTRCAL	AVTT and RCAL precision resistor
MGTRREF	RCAL precision resistor

Quad Usage Priority

If only a portion of the Quads in a Virtex-6 FPGA are used, they should be used based on their priority. The higher priority Quads should be used before the lower priority Quads.

The prioritization of the Quads is determined by package size. Because the smaller packages have a single internal power plane for each of the two power supply rails, MGTAVCC and MGTAVTT, their rules for prioritization are different than for the other packages in the Virtex-6 family of FPGAs.

Common Package Power Plane Prioritization

- Priority 1: MGT115

  This Quad, and specifically GTX0 within this Quad, must be instantiated if any of the GTX transceivers in the device are used in the application. It contains the RCAL circuit that is required for the RX and TX internal termination resistors for all of the transceivers in a column.

- Priority 2: MGT114/116

  Depending on availability in the package, these Quads have equal priority.

North/South Package Power Plane Prioritization

- Priority 1: MGT115

  This Quad should be used if any of the GTX transceivers in the device are used in the application. It contains the RCAL circuit that is required for the RX and TX internal termination resistors.

- Priority 2: MGT116/117/118

  If present in the Virtex-6 device, these Quads are connected in the package to the same power planes as MGT115, the north power plane bank. Therefore they have equal priority. Because the north power planes need to be powered for MGT115, these Quads are also powered; therefore they can be used without additional power supply connections.

- Priority 3: MGT110/111/112/113/114

  These transceivers are connected to the south power planes. They should be used if all Quads on the north power planes have already been utilized. If any of these Quads are used, then all MGTAVCC_N and MGTAVTT_S pins need to be connected to the appropriate power supply voltage.

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 transceiver design. Figure 5-6 illustrates the convention for the single-ended clock input voltage swing, peak- to-peak as used in the GTX transceiver portion of the *Virtex-6 FPGA Data Sheet*.

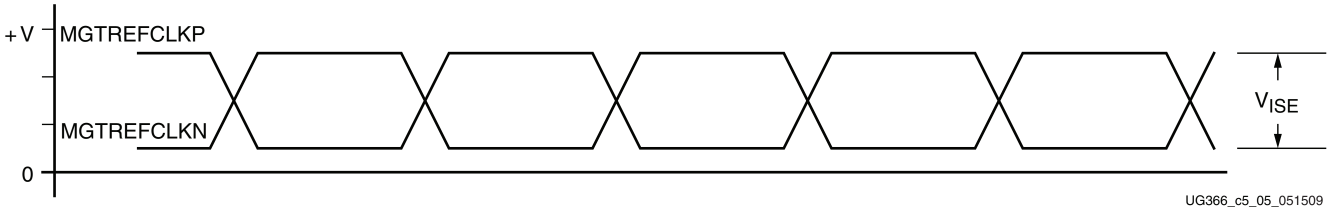


Figure 5-6: **Single-Ended Clock Input Voltage Swing, Peak-to-Peak**

Figure 5-7 illustrates the differential clock input voltage swing, peak-to-peak, which is defined as MGTREFCLKP – MGTREFCLKN.

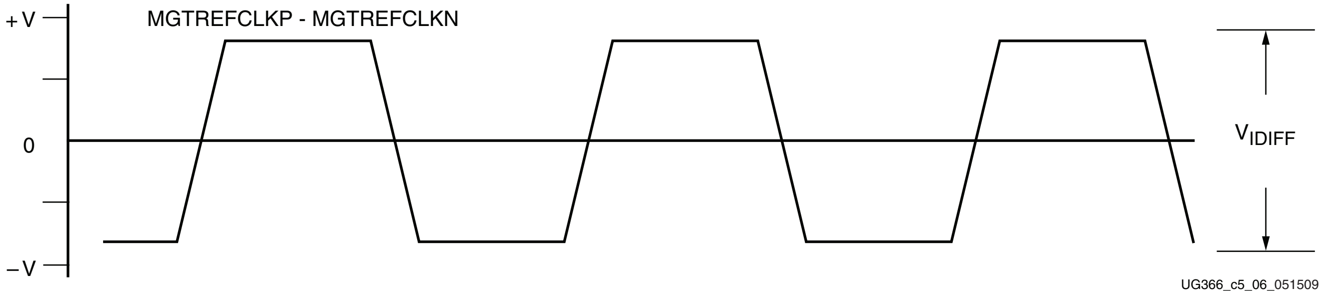


Figure 5-7: **Differential Clock Input Voltage Swing, Peak-to-Peak**

Figure 5-8 shows the rise and fall time convention of the reference clock.

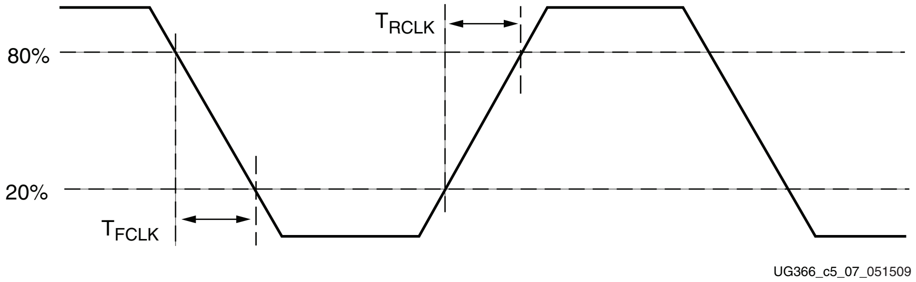


Figure 5-8: **Rise and Fall Times**

Figure 5-9 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. MGTAVCC is nominally 1.0V, hence the common mode voltage is nominally 800 mV. The resistor values given in Figure 5-9 are nominal. Refer to the *Virtex-6 FPGA Data Sheet* for exact specifications.

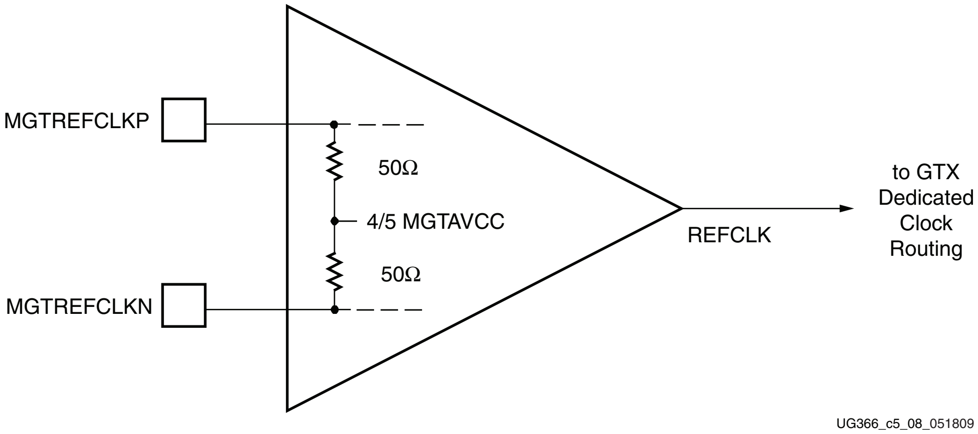


Figure 5-9: **MGTREFCLK Input Details**

Reference Clock Checklist

The following criteria must be met when choosing an oscillator for a design with GTX transceivers:

- Provide AC coupling between the oscillator output pins and the dedicated Quad clock input pins.
- Ensure that the differential voltage swing of the reference clock is the range as specified in the *Virtex-6 FPGA Data Sheet* (the nominal range is 200 mV – 2000 mV, and the typical value is 1200 mV).

  Note: These are nominal values. Refer to the *Virtex-6 FPGA Data Sheet* for exact values and ranges based on marginal conditions.
- Meet or exceed the reference clock characteristics as specified in the *Virtex-6 FPGA Data Sheet*.
- Meet or exceed the reference clock characteristics as specified in the standard for which the GTX transceiver provides physical layer support.

  Note: An actively toggling reference clock must be supplied to any transceiver that bypasses the TX or RX buffer at any time. The reference clock can be connected to an internal or external reference clock source. Refer to TX Buffer Bypass, page 156 and RX Buffer Bypass, page 230 for more information.
- 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 Quad clock input pins.
- Keep impedance discontinuities on the differential transmission lines to a minimum (impedance discontinuities generate jitter).

Reference Clock Interface

LVDS

Figure 5-10 shows how an LVDS oscillator is connected to a reference clock input of a GTX transceiver.

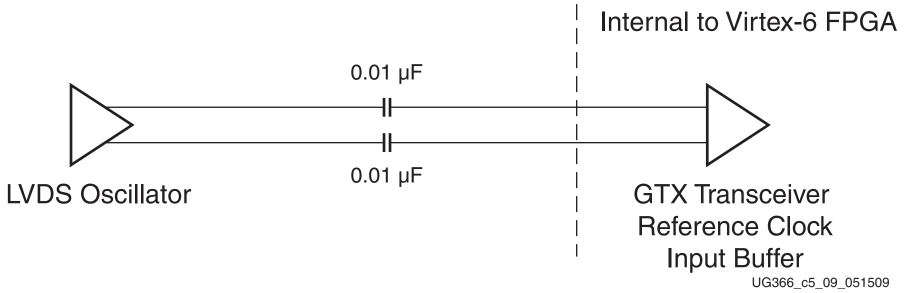


Figure 5-10: **Interfacing an LVDS Oscillator to a GTX Transceiver Reference Clock Input**

LVPECL

Figure 5-11 shows how an LVPECL oscillator is connected to a reference clock input of a GTX transceiver. The resistor values given in Figure 5-11 are nominal. Refer to the oscillator vendor data sheet for actual bias resistor requirement.

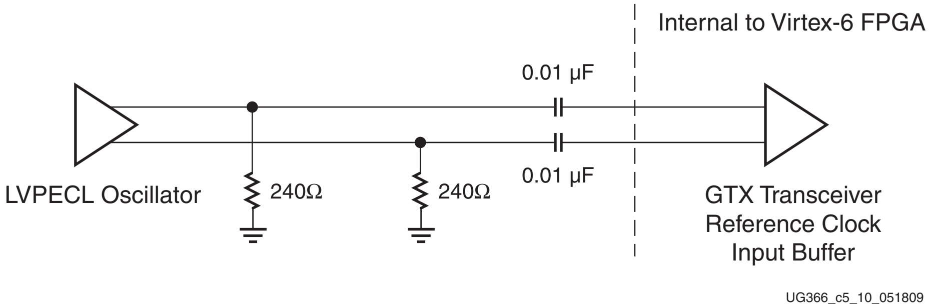


Figure 5-11: **Interfacing an LVPECL Oscillator to a GTX Transceiver Reference Clock Input**

AC Coupled Reference Clock

AC coupling of the oscillator reference clock output to the Quad reference clock inputs serves multiple purposes:

- DC current is blocked between the oscillator and the 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 Quad dedicated clock reference clock input pins are required.

Unused Reference Clocks

It is recommended to connect the unused differential input pin clock pair to ground or leave both MGTREFCLKP and MGTREFCLKN floating.

Reference Clock Power

MGTAVCC powers the GTX transceiver reference clock input circuit. Excessive noise on this supply has a negative impact on the performance of any Quad that uses the reference clock from this circuit.

Reference Clock Toggling

An actively toggling reference clock must be supplied to any transceiver that bypasses the TX or RX buffer at any time. Refer to TX Buffer Bypass, page 156 and RX Buffer Bypass, page 230 for more information.

Power Supply and Filtering

Overview

The Quad requires two analog power supplies: MGTAVCC at a nominal voltage level of 1.0 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. Some packages contain two planes (a north plane and a south plane) for each of the analog power supplies. See Analog Power Supply Pins for Virtex-6 LXT Devices, page 276 for a discussion of the internal power planes in the Virtex-6 FPGA packages.

Noise on the GTX 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 transceiver transmitter and reduced jitter tolerance in the GTX transceiver 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 transceiver analog power supplies. The total peak-to-peak noise as measured at the input pin of the FPGA should not exceed 10 mV_{PK-PK}.

Although the nominal voltage for MGTAVCC and VCCINT are both 1.0V, a separate power supply regulator should be used to power each of them. The reason for this recommendation is that MGTAVCC is very sensitive to noise and VCCINT has the potential to generate significant noise spikes well above the 10 mV_{PK-PK} maximum noise level recommended for the MGTAVCC power supply. Keeping the two power supply networks separate minimizes noise coupling between them.

Power Supply Regulators

Normally, the GTX 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 transceiver power supply pins. Minimizing the distance between the analog voltage regulators and the GTX 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 vs. 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 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 transceiver analog voltage rails: linear regulators and switching regulators. Each regulator type 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 transceiver analog supply rails. Inherently, a linear regulator does not inject significant noise into the regulated output voltage. Some, not all, linear regulators provide noise rejection at the output from noise present on the voltage input. The linear regulator usually requires a minimal number of external components to realize a circuit on the PCB.

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 depends 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 depends 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%. Thus 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, the linear regulator can be at a disadvantage in high current applications.

Switching Regulator

A switching regulator can provide a very efficient means to deliver a well-regulated voltage for the GTX 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, and it is impacted by the load current in a much lesser degree than 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 PCB to be effective.

Switching regulators generate significant noise and therefore usually require additional filtering before the voltage is delivered to the GTX transceiver analog power supply input of the Virtex-6 FPGA. The amplitude of the noise should be limited to less than 10 mV_{PK-PK}. Therefore the power supply filter should be designed to attenuate the noise from the switching regulator so that it meets this requirement.

Power Supply Distribution Network

Staged Decoupling

Die

There is decoupling capacitance on the die to filter the highest frequency noise components on the power supplies. The internal on-die circuits are the source for this very high frequency noise.

Package

Additional decoupling is in the Virtex-6 FPGA packages. Decoupling capacitors in the package provide attenuation for noise in the package power plane, thereby reducing the interaction between Quads. These capacitors in the package also help to maintain a low-impedance, high-frequency path between the power supply (MGTAVCC 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 requirement for decoupling on the PCB is more relaxed. Because the power supplies of the Quads are common and decoupled in the package, filtering is not necessary on the PCB to isolate the Quad power supply connections.

Decoupling capacitors provide two basic functions:

1. They help to isolate one circuit from another so that noise induced on the power supply by one circuit does not induce noise on the power supply of another circuit. In this case, the concern is noise coupling between Quads in the same FPGA.
2. They provide isolation between the power supply source and the load circuit.

Power Supply Decoupling Capacitors

For the GTX transceiver analog power supplies, the primary purpose of decoupling capacitors is to reduce the noise amplitude from the power supply source and other circuits on the PCB. The suggested filtering for the MGTAVCC and MGTAVTT power supplies is:

- One 0.22 µF, size 0402, ceramic capacitor per power supply pin
- One 4.7 µF, size 0402, ceramic capacitor per two Quads
- One 330 µF bulk capacitor for each power supply

Printed Circuit Board Design

Optimal performance from the GTX transceivers requires careful consideration in the design of the PCB. The areas of PCB design that must be considered are board stackup, component placement, and signal routing. The PCB design includes:

- Power distribution network for MGTAVTT and MGTAVCC
- Data lines for the receiver and transmitter
- Reference clock connections between the source oscillator and the GTX transceiver reference clock input
- Termination calibration resistor (see Termination Resistor Calibration Circuit, page 274)

This subsection discusses some issues regarding the implementation of these design issues on the PCB.

Board Stackup

For Virtex-6 FPGA GTX transceivers, the board stackup layers can be grouped into power distribution layers and signal routing layers. The power distribution layer group connects the power supply sources for MGTAVCC and MGTAVTT to the power supply pins on the Virtex-6 FPGA. Circuit board traces for receiver and transmitter data and the reference clock are provided in the signal routing layer group. These two layer groups can be considered separately within the stackup because it is the relative position of the layers within each group that is important. Figure 5-12 shows how the groups can be incorporated into an overall PCB stackup.

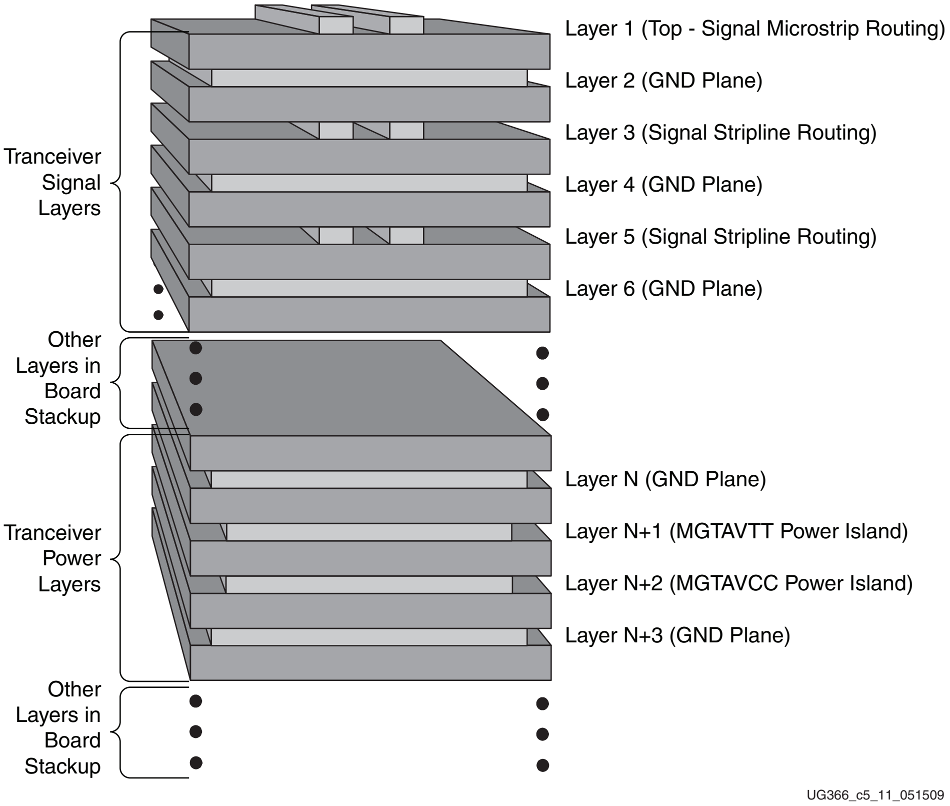

Layer 1 (Top - Signal Microstrip Routing)
Layer 2 (GND Plane)
Layer 3 (Signal Stripline Routing)
Layer 4 (GND Plane)
Layer 5 (Signal Stripline Routing)
Layer 6 (GND Plane)
Tranceiver Signal Layers
Other Layers in Board Stackup
Layer N (GND Plane)
Layer N+1 (MGTAVTT Power Island)
Layer N+2 (MGTAVCC Power Island)
Layer N+3 (GND Plane)
Tranceiver Power Layers
Other Layers in Board Stackup
UG366_c5_11_051509

Figure 5-12: **Stackup for GTX Transceiver Power and Signal Layers**

In this stackup, the GTX transceiver signal layers are at the top of the stackup. This group is composed of three signal routing layers and three plane layers. The planes provide a return current path for the transmission lines on the signal layers. Each of the signal routing layers is shielded from adjacent layers by a ground plane. Because of this shielding, the traces on each signal layer can be routed without having to consider the routing on an adjacent layer. This increases the routing channels on each signal layer, giving the layout designer more options for achieving an optimal signal breakout.

The GTX transceiver power layer group is treated as an autonomous group of layers that can be placed within the overall board stackup. This group of layers consists of a layer for each of the GTX transceiver power supplies (MGTAVCC and MGTAVTT) sandwiched between two ground layers. The ground layers provide shielding to the power planes from signals routed on layers above and below the power planes. Because of their low impedance, power planes are often prime candidates for providing return current paths for signals routed above or below them, even if the path is not intended. The ground planes also provide a means to connect the ground pins in the GTX transceiver region of the Virtex-6 FPGA.

GTX Transceiver Power Connections

Connections between the GTX transceiver power pins and the power distribution network are critical to the overall transceiver performance. The interface between the power distribution network and FPGA must be low impedance and low noise. The maximum noise allowed on the GTX transceiver power supplies at the FPGA is 10 mV_{PP} from 10 kHz to 80 MHz.

As mentioned in Board Stackup, the GTX transceiver power can be supplied by power islands.

Figure 5-13 shows the orientation of the power islands to the GTX transceiver region of the Virtex-6 package pinout. The power islands do not protrude into the SelectIO interface region of the FPGA.

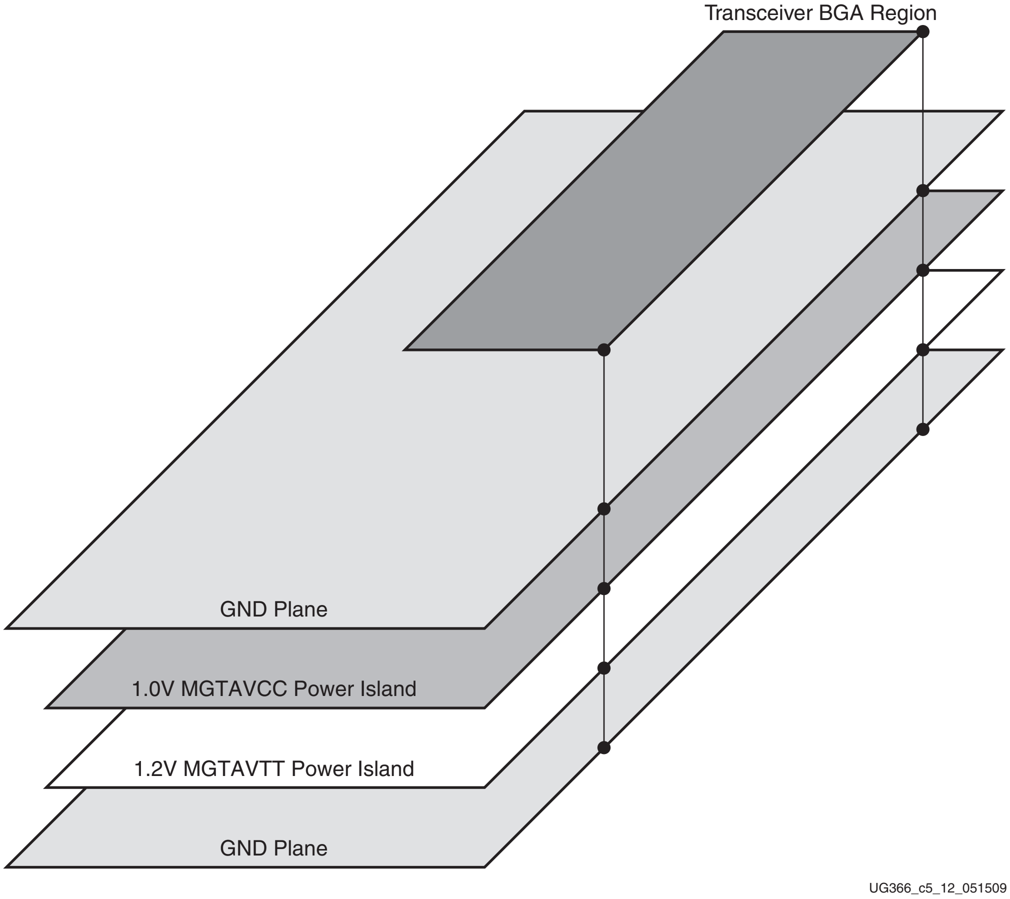


Figure 5-13: **GTX Transceiver Power Supply Islands**

Figure 5-14 shows how the power islands are oriented under the GTX transceiver region of the Virtex-6 FPGA and how to avoid exposure to the SelectIO interface region of the Virtex-6 FPGA BGA pin field. It also shows how filter capacitors discussed in Power Supply Decoupling Capacitors, page 288 can be oriented on the power planes to provide adequate noise filtering.

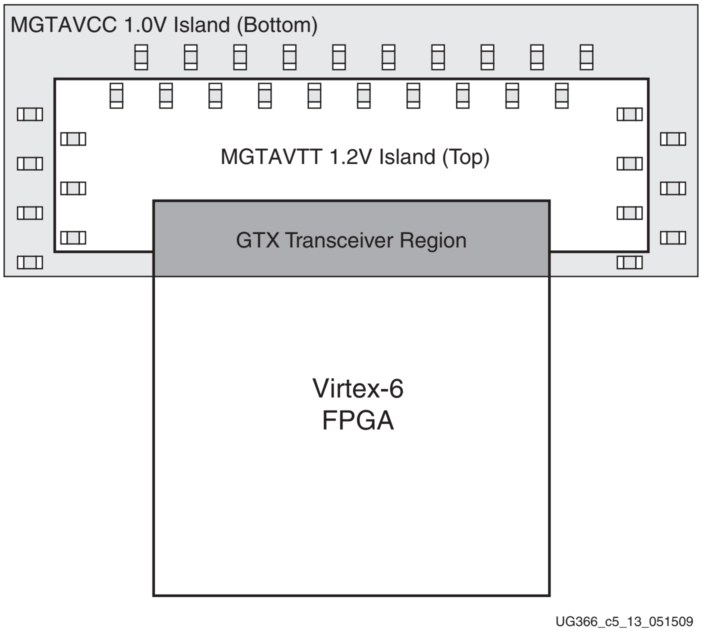


Figure 5-14: **Orientation between GTX Transceiver Power Islands and the Virtex-6 FPGA**

Signal BGA Breakout

The receiver, transmitter, and reference clock signals must be routed from the BGA pin field to destinations on the PCB. The signal routing layers provide the routing resources for this signal breakout. As shown in Figure 5-15, the signals on the outer rows of BGA pins can be routed using a microstrip on the top layer. These signals are routed to vias where the signal is transitioned from the top microstrip layer to striplines on Layer 3. An advantage to routing the signals from Layer 1 to Layer 3 is that the traces on both layers use the plane on Layer 2 as the return current reference plane.

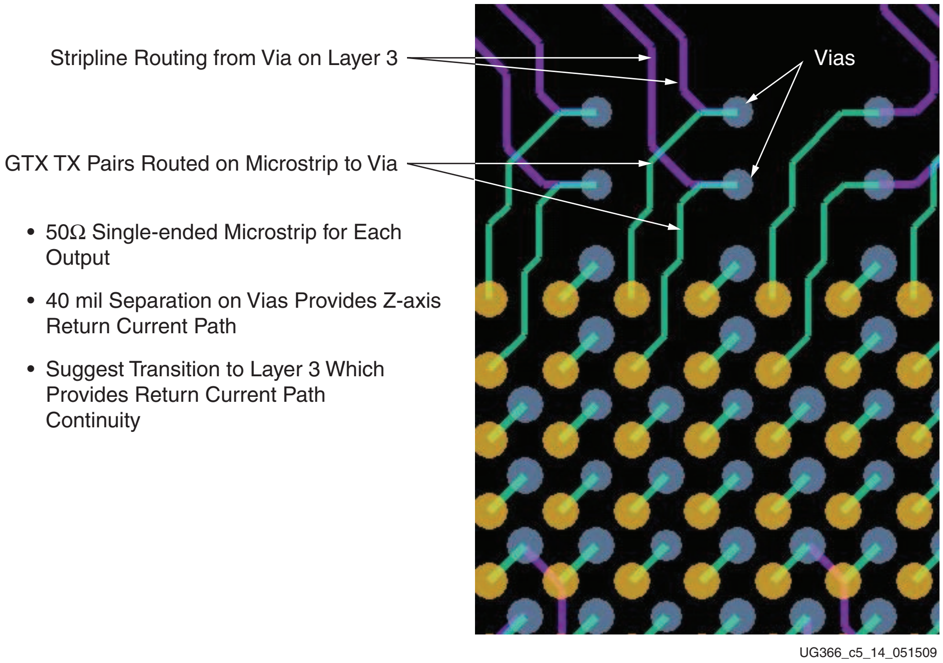

Stripline Routing from Via on Layer 3
Vias
GTX TX Pairs Routed on Microstrip to Via
• 50Ω Single-ended Microstrip for Each Output
• 40 mil Separation on Vias Provides Z-axis Return Current Path
• Suggest Transition to Layer 3 Which Provides Return Current Path Continuity
UG366_c5_14_051509

Figure 5-15: **TX Microstrip Breakout**

For signals that are on the inner rows of pins, it is necessary to route from the BGA pin pad on top of the board to a via. The signal pair is routed from each via by striplines on Layer 5 as shown in Figure 5-16. The Virtex-6 FPGA packages have been designed with grounds adjacent to all of the GTX transceiver signal pins. Having adjacent ground pins leads to adjacent BGA breakout vias. The adjacent ground vias provide a return current path for the signal via as the signal propagates from one layer to another layer in the stackup. If the two layers that are connected to the signal via have separate ground planes, the adjacent ground via provides a return current path in the Z-axis and thereby reduces the inductance of the via.

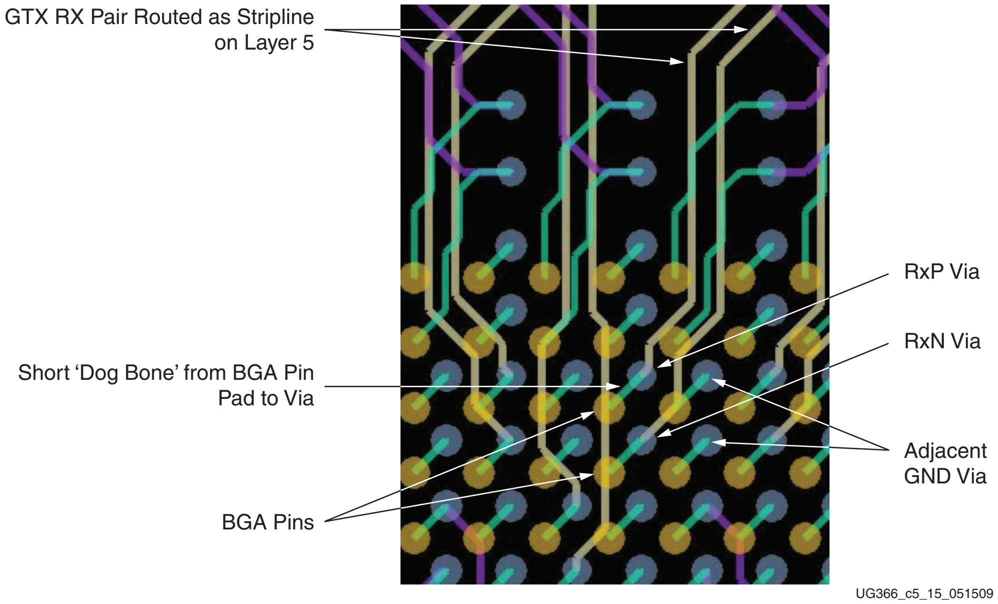

GTX RX Pair Routed as Stripline on Layer 5
RxP Via
RxN Via
Short 'Dog Bone' from BGA Pin Pad to Via
Adjacent GND Via
BGA Pins
UG366_c5_15_051509

Figure 5-16: **RX BGA Breakout to Stripline**

Crosstalk

Crosstalk is a major contributor to degradation in the performance of a GTX transceiver. The mechanisms for crosstalk are aggressor signals coupling into signal traces and/or coupling into the GTX transceiver power supplies. The latter is the most common mechanism and often the most damaging. Noise coupled into the power supply can corrupt the entire transceiver circuit rather than just a single lane, as in the case of coupling into signal traces. Also, the effect of noise coupled into the power supply is that the symptoms are often more difficult to interpret because the noise is integrated with the normal signals in the transceiver. The result is degradation in the performance of the transceiver that reveals itself as noise in the transmitter output and reduced jitter tolerance in the receiver.

To avoid performance degradation from crosstalk:

- Monitor the exposure of power planes to other circuits on the board, such as data lines for memory interfaces and processor buses.
- Provide adequate filtering to the GTX transceiver power supplies near the point of the load. The amount of filtering should be determined by the magnitude and frequency of the signal from potential noise sources. Noise on the GTX transceiver power supplies should be kept below 10 mV_{PP} from 10 kHz to 80 MHz.
- Be aware of the return current paths of signal traces in the vicinity of the GTX transceiver power distribution network. Besides broadband and edge coupling of traces on the same or adjacent layers, coupling from aggressor traces can occur if the aggressor signal is propagating from one layer to another, where each layer has a different reference plane. As the signal propagates through the portion of the via that does not have a return current path, it generates return currents in the next lowest impedance structure on the board. That victim can be a signal or power via for the GTX transceivers.

Hot Swapping Devices

In many applications, the device connected to the transceiver data pins is pluggable, and the device must often be installed while the FPGA is already powered and configured, i.e., hot-swappable or hot-pluggable. The Virtex-6 FPGA GTX transceivers support hot swapping. The Virtex-6 FPGA transceivers continue to function if the far-end receiver or transmitter is removed or installed. A recommended practice for hot-pluggable devices is to use a connector with staged pins so that the ground pins make contact prior to the other connector pins. This ensures that a reliable path to ground is present when the other device pins make contact.

SelectIO Usage Guidelines

Because a GTX transceiver's performance can degrade in an environment flooded with SelectIO™ interface activity, it is important to have guidelines for SelectIO interface usage that minimize the impact on GTX transceiver performance.

The pinout for the Virtex-6 FPGA package maintains a physical separation between the GTX transceiver pins and the SelectIO interface pins. Because of this separation in the package pinout, no SelectIO interface pins must be excluded when using the GTX transceivers.

Even though the Virtex-6 FPGA package pinout eliminates the crosstalk between SelectIO interface and the GTX transceiver pins due to pin adjacency, it is still possible to induce crosstalk on the PCB. Therefore, when routing signals on the PCB:

- Eliminate routing of GTX transceiver signals and SelectIO interface signals on adjacent layers. Be aware of the potential of broadside coupling if these signals are routed on adjacent layers.
- Maintain isolation of the return current paths for both the SelectIO interface signals and the GTX transceiver signals including both traces and vias.
- The power islands for the GTX transceivers are also a potential source for SelectIO interface induced noise. SelectO interface signals should not be routed over the GTX transceiver power islands.

Appendix A

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 A-1 shows the valid Data characters. Table A-2, page 305 shows the valid K characters.

Table A-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 A-1: **Valid Data Characters *(Cont'd)***

Data Byte	Bits	Current RD –	Current RD +
Name	HGF EDCBA	abcdei fghj	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 A-1: **Valid Data Characters *(Cont'd)***

Data Byte Name	Bits	Current RD –	Current RD +
	HGF EDCBA	abcdei fghj	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 A-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 A-1: **Valid Data Characters *(Cont'd)***

Data Byte	Bits	Current RD –	Current RD +
Name	HGF EDCBA	abcdei fghj	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 A-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 A-1: **Valid Data Characters *(Cont'd)***

Data Byte Name	Bits	Current RD –	Current RD +
	`HGF EDCBA`	`abcdei fghj`	`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 A-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 A-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 A-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 B

DRP Address Map of the GTX Transceiver

Table B-1 and Table B-2 list the DRP map sorted by address.

Note: Do NOT modify the Reserved bits! Attributes that are not described explicitly are set automatically by the Virtex-6 FPGA GTX Transceiver Wizard. These attributes must be left at their defaults, except for use cases that explicitly request different values.

Table B-1: **Attributes DRP Address Map**

DADDR	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Binary Encoding
0h	15:0	R/W	PMA_RX_CFG	15:0		
1h	15:9	R/W	PMA_RXSYNC_CFG	6:0		
	8:0		PMA_RX_CFG	24:16		
2h	15:0	R/W	RXUSRCLK_DLY	15:0		
3h	15:0	R/W	BIAS_CFG	15:0		
4h	15	R/W	RX_LOSS_OF_SYNC_FSM		FALSE	0
					TRUE	1
	14		RX_BUFFER_USE		FALSE	0
					TRUE	1
	13:10		CHAN_BOND_SEQ_1_ENABLE	3:0	0-15	1[1]
	9:0		CHAN_BOND_SEQ_1_1	9:0	0-1023	1[1]
5h	15:14	R/W	Reserved	1:0		
	13:10		CHAN_BOND_1_MAX_SKEW	3:0	1-14	1[1]
	9:0		CHAN_BOND_SEQ_1_2	9:0	0-1023	1[1]

Table B-1: **Attributes DRP Address Map *(Cont'd)***

DADDR	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Binary Encoding
6h	15:13	R/W	RX_LOS_INVALID_INCR	2:0	1	000
					2	001
					4	010
					8	011
					16	100
					32	101
					64	110
					128	111
	12:10		RX_LOS_THRESHOLD	2:0	4	000
					8	001
					16	010
					32	011
					64	100
					128	101
					256	110
					512	111
	9:0		CHAN_BOND_SEQ_1_3	9:0	0-1023	1[1]
7h	15:12	R/W	RX_IDLE_LO_CNT	3:0	0-15	1[1]
	11:10		CHAN_BOND_SEQ_LEN	1:0	1	00
					2	01
					Reserved	
					4	11
	9:0		CHAN_BOND_SEQ_1_4	9:0	0-1023	1[1]
8h	15	R/W	RX_EN_RATE_RESET_BUF		FALSE	0
					TRUE	1
	14		RX_EN_REALIGN_RESET_BUF		FALSE	0
					TRUE	1
	13:10		CHAN_BOND_SEQ_2_ENABLE	3:0	0-15	1[1]
	9:0		CHAN_BOND_SEQ_2_1	9:0	0-1023	1[1]
9h	15	R/W	RX_EN_MODE_RESET_BUF		FALSE	0
					TRUE	1
	14		CHAN_BOND_KEEP_ALIGN		FALSE	0
					TRUE	1
	13:10		CHAN_BOND_2_MAX_SKEW	3:0	1-14	1[1]
	9:0		CHAN_BOND_SEQ_2_2	9:0	0-1023	1[1]
Ah	15	R/W	RX_EN_IDLE_RESET_PH		FALSE	0
					TRUE	1
	14		CHAN_BOND_SEQ_2_USE		FALSE	0
					TRUE	1
	13:10		CHAN_BOND_SEQ_2_CFG	3:0	<4:0> 0-31	1[1]
	9:0		CHAN_BOND_SEQ_2_3	9:0	0-1023	1[1]

Table B-1: **Attributes DRP Address Map *(Cont'd)***

DADDR	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Binary Encoding
Bh	15:12	R/W	RX_IDLE_HI_CNT	3:0	0-15	1[1]
	11		RX_XCLK_SEL		RXREC	0
					RXUSR	1
	10		RX_EN_IDLE_RESET_BUF		FALSE	0
					TRUE	1
	9:0		CHAN_BOND_SEQ_2_4	9:0	0-1023	1[1]
Ch	15:14	R/W	Reserved	1:0		
	13		RX_FIFO_ADDR_MODE		FALSE	0
					TRUE	1
	12:0		Reserved	12:0		
Dh	15	R/W	CLK_COR_PRECEDENCE		FALSE	0
					TRUE	1
	14		CLK_CORRECT_USE		FALSE	0
					TRUE	1
	13:10		CLK_COR_SEQ_1_ENABLE	3:0	0-15	1[1]
	9:0		CLK_COR_SEQ_1_1	9:0	0-1023	1[1]
Eh	15	R/W	CLK_COR_KEEP_IDLE		FALSE	0
					TRUE	1
	14:10		CLK_COR_REPEAT_WAIT	4:0	0-31	1[1]
	9:0		CLK_COR_SEQ_1_2	9:0	0-1023	1[1]
Fh	15:10	R/W	CLK_COR_MIN_LAT	5:0	3-48	1[1]
	9:0		CLK_COR_SEQ_1_3	9:0	0-1023	1[1]
10h	15:10	R/W	CLK_COR_MAX_LAT	5:0	3-48	1[1]
	9:0		CLK_COR_SEQ_1_4	9:0	0-1023	1[1]
11h	15	R/W	CLK_COR_INSERT_IDLE_FLAG		FALSE	0
					TRUE	1
	14		CLK_COR_SEQ_2_USE		FALSE	0
					TRUE	1
	13:10		CLK_COR_SEQ_2_ENABLE	3:0	0-15	1[1]
	9:0		CLK_COR_SEQ_2_1	9:0	0-1023	1[1]
12h	15	R/W	Reserved			
	14		SHOW_REALIGN_COMMA		FALSE	0
					TRUE	1
	13:12		RX_SLIDE_MODE	1:0	OFF	00
					AUTO	01
					PCS	10
					PMA	11
	11:10		CLK_COR_ADJ_LEN	1:0	1	00
					2	01
					Reserved	
					4	11
	9:0		CLK_COR_SEQ_2_2	9:0	0-1023	1[1]

Table B-1: **Attributes DRP Address Map *(Cont'd)***

DADDR	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Binary Encoding
13h	15:12	R/W	RX_SLIDE_AUTO_WAIT	3:0	0-15	1[(1)]
	11:10		CLK_COR_DET_LEN	1:0	1	00
					2	01
					Reserved	
					4	11
	9:0		CLK_COR_SEQ_2_3	9:0	0-1023	1[(1)]
14h	15	R/W	DEC_VALID_COMMA_ONLY		FALSE	0
					TRUE	1
	14		ALIGN_COMMA_WORD		1	0
					2	1
	13		RX_DECODE_SEQ_MATCH		FALSE	0
					TRUE	1
	12		Reserved			
	11		DEC_MCOMMA_DETECT		FALSE	0
					TRUE	1
	10		DEC_PCOMMA_DETECT		FALSE	0
					TRUE	1
	9:0		CLK_COR_SEQ_2_4	9:0	0-1023	1[(1)]
15h	15:0	R/W	PMA_CDR_SCAN	15:0		
16h	15:11	R/W	CDR_PH_ADJ_TIME	4:0	0-31	1[(1)]
	10:0		PMA_CDR_SCAN	26:16		

Table B-1: **Attributes DRP Address Map *(Cont'd)***

DADDR	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Binary Encoding
17h	15	R/W	GEN_RXUSRCLK		FALSE	0
					TRUE	1
	14:12		RX_DATA_WIDTH	2:0	20	011
					8	000
					10	001
					16	010
					32	100
					40	101
	11		CHAN_BOND_SEQ_2_CFG	4	<4:0> 0-31	1[1]
	10		BIAS_CFG	16		
	9:5		RX_CLK25_DIVIDER	4:0	6	00101
					1	00000
					2	00001
					3	00010
					4	00011
					5	00100
					7	00110
					8	00111
					9	01000
					10	01001
					11	01010
					12	01011
					13	01100
					14	01101
					15	01110
					16	01111
					17	10000
					18	10001
					19	10010
					20	10011
					21	10100
					22	10101
					23	10110
					24	10111
					25	11000
					26	11001
					27	11010
					28	11011
					29	11100
					30	11101
					31	11110
					32	11111
	4		AC_CAP_DIS		FALSE	0
					TRUE	1
	3		GTX_CFG_PWRUP		FALSE	0
					TRUE	1
	2:0		OOBDETECT_THRESHOLD	2:0	0-7	1[1]

Table B-1: **Attributes DRP Address Map *(Cont'd)***

DADDR	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Binary Encoding
18h	15:4	R/W	Reserved	11:0		
	3		RXGEARBOX_USE		FALSE	0
					TRUE	1
	2:0		GEARBOX_ENDEC	2:0	0-7	1[1]
19h	15:0	R/W	RXPLL_COM_CFG	15:0		
1Ah	15:8	R/W	RXPLL_CP_CFG	7:0		
	7:0		RXPLL_COM_CFG	23:16		
1Bh	15:14	R/W	RXPLL_DIVSEL_OUT	1:0	1	00
					2	01
					4	10
	13:11		RXPLL_LKDET_CFG	2:0	0-7	1[1]
	10:7		Reserved	3:0		
	6		RXPLL_DIVSEL45_FB		5	1
					4	0
	5:1		RXPLL_DIVSEL_FB	4:0	2	00000
					4	00010
					5	00011
	0		Reserved			
1Ch	15	R/W	RX_OVERSAMPLE_MODE		FALSE	0
					TRUE	1
	14:6		Reserved	8:0		
	5:1		RXPLL_DIVSEL_REF	4:0	1	10000
					2	00000
	0		Reserved			
1Dh	15:0	R/W	TXPLL_COM_CFG	15:0		
1Eh	15:8	R/W	TXPLL_CP_CFG	7:0		
	7:0		TXPLL_COM_CFG	23:16		
1Fh	15:14	R/W	TXPLL_DIVSEL_OUT	1:0	1	00
					2	01
					4	10
	13:11		TXPLL_LKDET_CFG	2:0	0-7	1[1]
	10:9		Reserved	1:0		
	8		TX_CLK_SOURCE		RXPLL	1
					TXPLL	0
	7		Reserved			
	6		TXPLL_DIVSEL45_FB		5	1
					4	0
	5:1		TXPLL_DIVSEL_FB	4:0	2	00000
					4	00010
					5	00011
	0		Reserved			

Table B-1: **Attributes DRP Address Map *(Cont'd)***

DADDR	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Binary Encoding
20h	15	R/W	TX_OVERSAMPLE_MODE		FALSE	0
					TRUE	1
	14:6		Reserved	8:0		
	5:1		TXPLL_DIVSEL_REF	4:0	1	10000
					2	00000
	0		Reserved			
21h	15	R/W	PCI_EXPRESS_MODE		FALSE	0
					TRUE	1
	14		Reserved			
	13:0		TX_DETECT_RX_CFG	13:0		
22h	15		PMA_CAS_CLK_EN		FALSE	0
					TRUE	1
	14:0		Reserved	14:0		

Table B-1: **Attributes DRP Address Map *(Cont'd)***

DADDR	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Binary Encoding
23h	15	R/W	Reserved			
	14:10		TX_CLK25_DIVIDER	4:0	6	00101
					1	00000
					2	00001
					3	00010
					4	00011
					5	00100
					7	00110
					8	00111
					9	01000
					10	01001
					11	01010
					12	01011
					13	01100
					14	01101
					15	01110
					16	01111
					17	10000
					18	10001
					19	10010
					20	10011
					21	10100
					22	10101
					23	10110
					24	10111
					25	11000
					26	11001
					27	11010
					28	11011
					29	11100
					30	11101
					31	11110
					32	11111
	9:0		TRANS_TIME_TO_P2	9:0		
24h	15:12	R/W	COM_BURST_VAL	3:0	0-15	1[1]
	11:0		TRANS_TIME_FROM_P2	11:0		
25h	15	R/W	TX_PMADATA_OPT		0-1	1[1]
	14:10		Reserved	4:0		
	9:8		CM_TRIM	1:0	0-3	1[1]
	7:0		TRANS_TIME_NON_P2	7:0		
26h	15:14	R/W	BGTEST_CFG	1:0	0-3	1[1]
	13:12		TXPLL_SATA	1:0	0-3	1[1]
	11:6		SATA_MIN_WAKE	5:0	1-61	1[1]
	5:0		SATA_MAX_WAKE	5:0	1-61	1[1]

Table B-1: **Attributes DRP Address Map *(Cont'd)***

DADDR	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Binary Encoding
27h	15	R/W	TX_EN_RATE_RESET_BUF		FALSE	0
					TRUE	1
	14:12		SATA_IDLE_VAL	2:0	0-7	1[1]
	11:6		SATA_MIN_INIT	5:0	1-61	1[1]
	5:0		SATA_MAX_INIT	5:0	1-61	1[1]
28h	15	R/W	Reserved			
	14:12		SATA_BURST_VAL	2:0	0-7	1[1]
	11:6		SATA_MIN_BURST	5:0	1-61	1[1]
	5:0		SATA_MAX_BURST	5:0	1-61	1[1]
29h	15:12	R/W	Reserved	3:0		
	11:6		SAS_MIN_COMSAS	5:0	1-61	1[1]
	5:0		SAS_MAX_COMSAS	5:0	1-61	1[1]
2Ah	15:9	R/W	Reserved	6:0		
	8		RXPRBSERR_LOOPBACK		0-1	1[1]
	7:0		Reserved	7:0		
2Bh	15:12	R/W	Reserved	3:0		
	11		RX_EN_IDLE_HOLD_DFE		FALSE	0
					TRUE	1
	10		RX_EN_IDLE_RESET_FR		FALSE	0
					TRUE	1
	9		RX_EN_IDLE_HOLD_CDR		FALSE	0
					TRUE	1
	8:0		Reserved	8:0		
2Ch	15:0	R/W	Reserved	15:0		
2Dh	15:8	R/W	RX_EYE_OFFSET	7:0		
	7:0		DFE_CFG	7:0	0-255	1[1]
2Eh	15:11	R/W	DFE_CAL_TIME	4:0	0-31	1[1]
	10:9		RX_EYE_SCANMODE	1:0	0-3	1[1]
	8		RCV_TERM_VTTRX		FALSE	0
					TRUE	1
	7		RCV_TERM_GND		FALSE	0
					TRUE	1
	6		TERMINATION_OVRD		FALSE	0
					TRUE	1
	5		Reserved			
	4:0		TERMINATION_CTRL	4:0	0-31	1[1]
2Fh	15	R/W	TXGEARBOX_USE		FALSE	0
					TRUE	1
	14		TX_XCLK_SEL		TXUSR	1
					TXOUT	0
	13:11		TX_IDLE_ASSERT_DELAY	2:0	0-7	1[1]
	10		COMMA_DOUBLE		FALSE	0
					TRUE	1
	9:0		COMMA_10B_ENABLE	9:0	0-1023	1[1]

Table B-1: **Attributes DRP Address Map *(Cont'd)***

DADDR	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Binary Encoding
30h	15:14	R/W	Reserved	1:0		
	13:11		TX_IDLE_DEASSERT_DELAY	2:0	0-7	1[1]
	10		MCOMMA_DETECT		FALSE	0
					TRUE	1
	9:0		MCOMMA_10B_VALUE	9:0	0-1023	1[1]
31h	15	R/W	GEN_TXUSRCLK		FALSE	0
					TRUE	1
	14:12		TX_DATA_WIDTH	2:0	20	011
					8	000
					10	001
					16	010
					32	100
					40	101
	11		TX_BUFFER_USE		FALSE	0
					TRUE	1
	10		PCOMMA_DETECT		FALSE	0
					TRUE	1
	9:0		PCOMMA_10B_VALUE	9:0	0-1023	1[1]
32h	15:0	R/W	PMA_CFG	15:0		
33h	15:0	R/W	PMA_CFG	31:16		
34h	15:0	R/W	PMA_CFG	47:32		
35h	15:0	R/W	PMA_CFG	63:48		
36h	15:4	R/W	PMA_CFG	75:64		
	3:0		PMA_TX_CFG	19:16		
37h	15:0	R/W	PMA_TX_CFG	15:0		
38h	15:14	R/W	Reserved	1:0		
	13:7		TX_MARGIN_FULL_0	6:0	0-127	1[1]
	6:0		TX_MARGIN_LOW_0	6:0	0-127	1[1]
39h	15:14	R/W	TX_TDCC_CFG	1:0	0-3	1[1]
	13:7		TX_MARGIN_FULL_1	6:0	0-127	1[1]
	6:0		TX_MARGIN_LOW_1	6:0	0-127	1[1]
3Ah	15	R/W	TXDRIVE_LOOPBACK_PD		FALSE	0
					TRUE	1
	14		TXDRIVE_LOOPBACK_HIZ		FALSE	0
					TRUE	1
	13:7		TX_MARGIN_FULL_2	6:0	0-127	1[1]
	6:0		TX_MARGIN_LOW_2	6:0	0-127	1[1]
3Bh	15:14		Reserved	1:0		
	13:7		TX_MARGIN_FULL_3	6:0	0-127	1[1]
	6:0		TX_MARGIN_LOW_3	6:0	0-127	1[1]
3Ch	15	R/W	TX_DRIVE_MODE		DIRECT	0
					PIPE	1
	14		Reserved			
	13:7		TX_MARGIN_FULL_4	6:0	0-127	1[1]
	6:0		TX_MARGIN_LOW_4	6:0	0-127	1[1]
3Dh	15:0	R/W	Reserved	15:0		

Table B-1: **Attributes DRP Address Map *(Cont'd)***

DADDR	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Binary Encoding
3Eh	15:10	R/W	Reserved	5:0		
	9:5		TX_DEEMPH_1	4:0	0-31	1[1]
	4:0		TX_DEEMPH_0	4:0	0-31	1[1]
3Fh	15:8	R/W	Reserved	7:0		
	7:0		TRANS_TIME_RATE	7:0		
40h	15:0	R/W	Reserved	15:0		
41h	31:16	R/W	TST_ATTR	31:16		
42h	15:0		TST_ATTR	15:0		
43h	15:6	R/W	Reserved	9:0		
	5:3		RXRECCLK_CTRL	2:0	RXRECCLKPCS	000
					OFF_HIGH	110
					OFF_LOW	101
					RXPLLREFCLK_DIV1	011
					RXPLLREFCLK_DIV2	100
					RXRECCLKPMA_DIV1	001
					RXRECCLKPMA_DIV2	010
	2:0		TXOUTCLK_CTRL	2:0	TXOUTCLKPCS	000
					OFF_HIGH	110
					OFF_LOW	101
					TXOUTCLKPMA_DIV1	001
					TXOUTCLKPMA_DIV2	010
					TXPLLREFCLK_DIV1	011
					TXPLLREFCLK_DIV2	100
44h	15:10	R/W	Reserved	5:0		
	9:0		POWER_SAVE	9:0	0-1023	1[1]
45h	15:12	R/W	Reserved	3:0		
	11:6		TX_USRCLK_CFG	5:0		
	5:0		TX_BYTECLK_CFG	5:0		
46h	15:10	R/W	Reserved	5:0		
	9:0		RXRECCLK_DLY	9:0	0-1023	1[1]
47h	15:10	R/W	Reserved	5:0		
	9:0		Reserved	9:0		
48h	15:0	R/W	Reserved	15:0		
49h	15:0	R/W	Reserved	15:0		
4Ah	15:0	R/W	Reserved	15:0		
4Bh	15	R/W	Reserved			
	14		RX_EN_REALIGN_RESET_BUF2		FALSE	0
					TRUE	1
	13:11		Reserved	2:0		
	10:6		RX_DLYALIGN_EDGESET	4:0	0-31	1[1]
	5:3		TX_DLYALIGN_MONSEL	2:0	0-7	1[1]
	2:0		RX_DLYALIGN_MONSEL	2:0	0-7	1[1]

Table B-1: **Attributes DRP Address Map *(Cont'd)***

DADDR	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Binary Encoding
4Ch	15:8	R/W	TX_DLYALIGN_OVRDSETTING	7:0	0-255	1[1]
	7:4		TX_DLYALIGN_LPFINC	3:0	0-15	1[1]
	3:0		TX_DLYALIGN_CTRINC	3:0	0-15	1[1]
4Dh	15:8		RX_DLYALIGN_OVRDSETTING	7:0	0-255	1[1]
	7:4		RX_DLYALIGN_LPFINC	3:0	0-15	1[1]
	3:0		RX_DLYALIGN_CTRINC	3:0	0-15	1[1]
4Eh	15:0	R/W	Reserved	15:0		
4Fh	15:0	R/W	Reserved	15:0		

Notes:

1. The DRP has the same binary encoding value as the attribute encoding value.

Table B-2: **Status Registers DRP Address Map**

DADDR	DRP Bits	R/W	Register Name	Register Bits	Register Encoding	DRP Binary Encoding
82h[2]	15:0	R	RX_PRBS_ERR_CNT	15:0	0-65535	1[1]

Notes:

1. The DRP has the same binary encoding value as the attribute encoding value.
2. The receiver has to be operational for this DRP register to take effect.

Appendix C

Low Latency Design

This appendix illustrates the latency of the different functional blocks inside the TX and the RX sections of the GTX transceiver. Figure C-1 shows a pictorial definition of the TX and RX latencies.

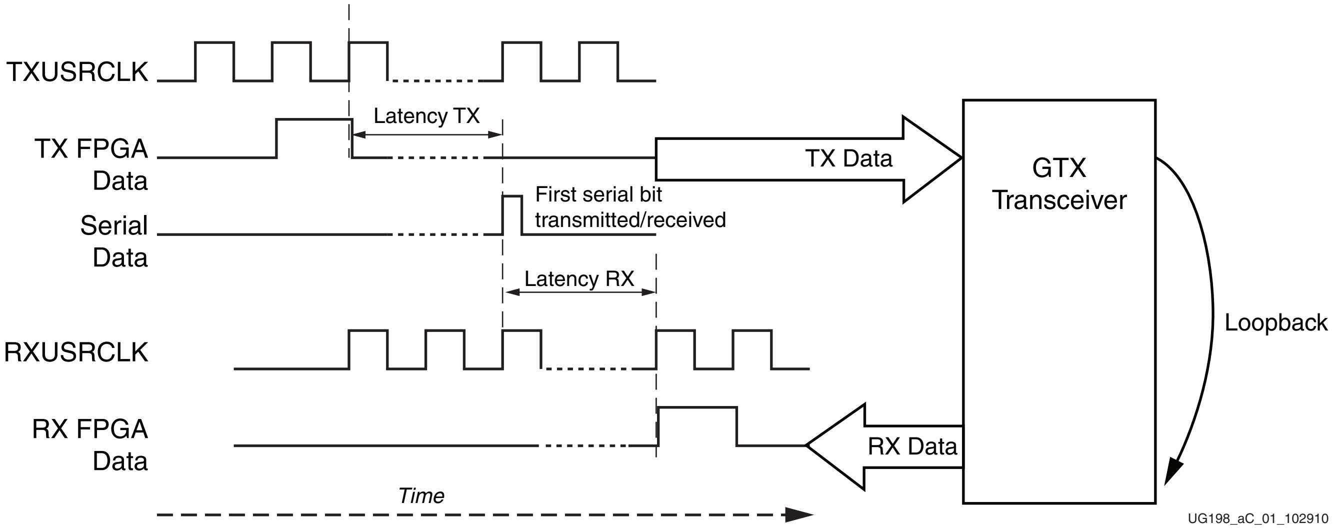


Figure C-1: **Latency Definition**

Each functional block has a latency defined as the time difference between the inputs and the outputs of the specific block. Some blocks in the GTX transceiver can be bypassed, reducing the latency of the datapath through the transmitter or the receiver. The latency of the blocks is deterministic with the exception of the RX elastic buffer and the TX buffer. Bypassing buffers requires the phase alignment procedure. Refer to TX Buffer Bypass, page 156 and RX Buffer Bypass, page 230 for more details.

GTX Transceiver TX Latency

Figure C-2 shows a detailed block diagram of the GTX transceiver TX. Refer to TX Overview, page 127 for more details on the GTX transceiver TX blocks.

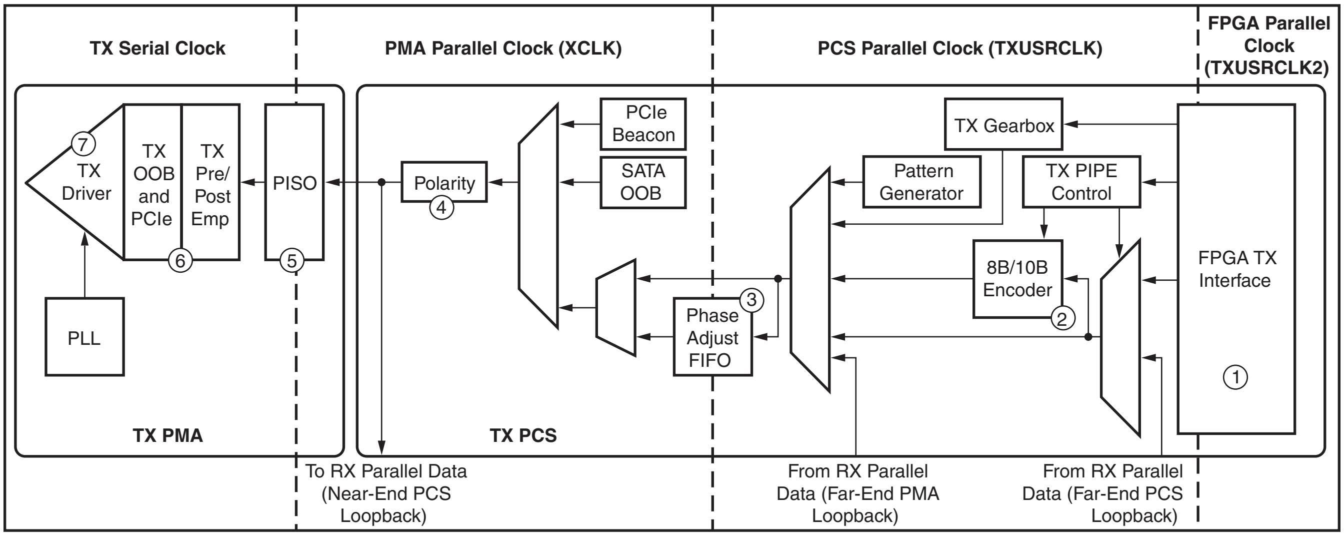

TX Serial Clock
PMA Parallel Clock (XCLK)
PCS Parallel Clock (TXUSRCLK)
FPGA Parallel Clock (TXUSRCLK2)
7
TX Driver
TX OOB and PCIe
TX Pre/ Post Emp
6
PISO
5
PLL
TX PMA
Polarity
4
PCIe Beacon
SATA OOB
Phase Adjust FIFO
3
TX PCS
Pattern Generator
TX Gearbox
TX PIPE Control
8B/10B Encoder
2
FPGA TX Interface
1
To RX Parallel Data (Near-End PCS Loopback)
From RX Parallel Data (Far-End PMA Loopback)
From RX Parallel Data (Far-End PCS Loopback)
UG366_aC_02_110110

Figure C-2: **GTX Transceiver TX Block Diagram**

Table C-1 defines the latency for the specific functional blocks or group of functional blocks of the transmitter section of the GTX transceiver. The values in the Block Number column correspond to the circled numbers in Figure C-2.

Table C-1: **GTX Transceiver TX Latency**

Block Number	Block Name	TX Latency (TXUSRCLK)		
1	FPGA TX Interface	TX_DATA_WIDTH = 8/10	TX_DATA_WIDTH = 16/20	TX_DATA_WIDTH = 32/40
		0.5 cycle	1 cycle	2 cycles
2	8B/10B Encoder	TXENC8B10BUSE = 0	TXENC8B10BUSE = 1	
		0 cycles	1 cycle	
3	TX Buffer	TX_BUFFER_USE = FALSE	TX_BUFFER_USE = TRUE	
		1 cycle	3 cycles	
4+5+6+7	PMA + Interface	3 cycles		
Total TX Latency		Minimum	Maximum	
		4.5 cycles	9 cycles	

GTX Transceiver RX Latency

Figure C-3 shows a detailed block diagram of the GTX transceiver RX. Refer to RX Overview, page 183 for more details on the GTX transceiver RX blocks.

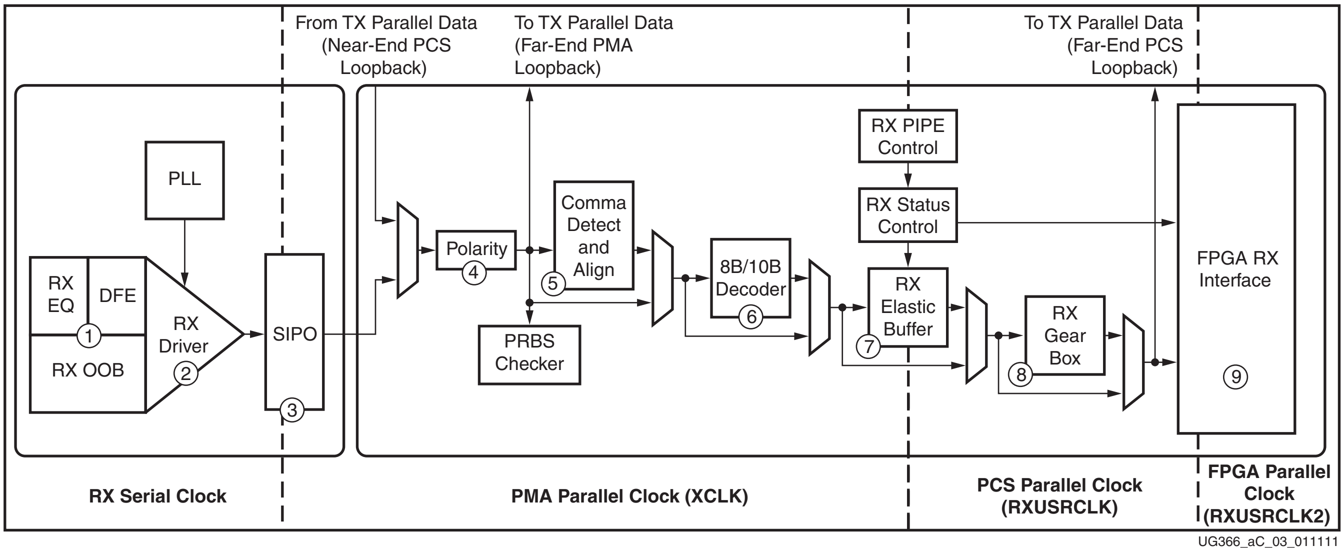


Figure C-3: **GTX Transceiver RX Block Diagram**

Table C-2 defines the latency for the specific functional blocks or group of functional blocks of the receiver section of the GTX transceiver. The values in the Block Number column correspond to the circled numbers in Figure C-3.

Table C-2: **GTX Transceiver RX Latency**

<table>
<tr><th>Block Number</th><th>Block Name</th><th colspan="4">RX Latency (RXUSRCLK)</th></tr>
<tr><td rowspan="2">9</td><td rowspan="2">FPGA RX Interface</td><td>RX_DATA_WIDTH = 8/10</td><td colspan="2">RX_DATA_WIDTH = 16/20</td><td>RX_DATA_WIDTH = 32/40</td></tr>
<tr><td>1.5 cycle</td><td colspan="2">2 cycles</td><td>3 cycles</td></tr>
<tr><td>1+2+3+4</td><td>PMA + Interface</td><td colspan="4">4 cycles + 7 UI ± 1 UI</td></tr>
<tr><td rowspan="3">5</td><td rowspan="3">Comma Detect</td><td>RXCOMMADETUSE = 0</td><td colspan="3">RXCOMMADETUSE = 1</td></tr>
<tr><td rowspan="2">1 cycle</td><td colspan="2">SHOW_REALIGN_COMMA = TRUE</td><td>SHOW_REALIGN_COMMA = FALSE</td></tr>
<tr><td colspan="2">2.5 to 3.5 cycles</td><td>2 to 3 cycles</td></tr>
<tr><td rowspan="2">6</td><td rowspan="2">8B/10B Decoder</td><td colspan="2">RXDEC8B10BUSE = 0</td><td colspan="2">RXDEC8B10BUSE = 1</td></tr>
<tr><td colspan="2">0 cycles</td><td colspan="2">1 cycle</td></tr>
<tr><td rowspan="2">7</td><td rowspan="2">RX Elastic Buffer</td><td colspan="2">RX_BUFFER_USE = FALSE</td><td colspan="2">RX_BUFFER_USE = TRUE</td></tr>
<tr><td colspan="2">0 cycles</td><td colspan="2">1.5 to 2.5 cycles +
(CLK_COR_MIN_LAT/2)</td></tr>
<tr><td colspan="2" rowspan="2">Total RX Latency</td><td colspan="2">Minimum</td><td colspan="2">Maximum</td></tr>
<tr><td colspan="2">6.5 cycles + 6 UI</td><td colspan="2">14 + (CLK_COR_MIN_LAT/2) cycles + 8 UI</td></tr>
</table>