10G Ethernet PCS/PMA v6.0

LogiCORE IP Product Guide

Vivado Design Suite

PG068 February 4, 2021

XILINX

Table of Contents

Introduction

The LogiCORE™ IP 10G Ethernet Physical Coding Sublayer/Physical Medium Attachment (PCS/PMA) core forms a seamless interface between the Xilinx 10G Ethernet Media Access Controller (MAC) core and a 10 Gb/s-capable PHY, enabling the design of high-speed Ethernet systems and subsystems.

Features

- Designed to 10 Gigabit Ethernet specification IEEE Standard 802.3-2012 clause 49, 72, 73, 74
- Optional Management Data Interface (MDIO) to manage PCS/PMA registers
- Supports 10GBASE-SR, -LR and -ER optical links in Zynq-7000, UltraScale™, Virtex-7, and Kintex-7 devices (LAN mode only)
- Supports 10GBASE-KR backplane links in UltraScale, and Virtex-7 devices, including Auto-Negotiation (AN), Training and Forward Error Correction (FEC)
- 10 Gigabit Ethernet Media Independent Interface (XGMII) connects seamlessly to the Xilinx 10 Gigabit Ethernet MAC
- A 64-bit or 32-bit data width option is available for the 10GBASE-R standard. The 10GBASE-KR standard is always provided with a 64-bit data width.

LogiCORE IP Facts Table	
Core Specifics	
Supported Device Family(1) (2)	**10GBASE-R**: UltraScale™ Zynq®-7000 SoC, Virtex®-7, Kintex®-7(3) **10GBASE-KR**: UltraScale™, Virtex-7(4)
Supported User Interfaces	MDIO, XGMII
Resources	Performance and Resource Utilization web page
Provided with Core	
Design Files	Encrypted RTL
Example Design	Verilog and VHDL
Test Bench	Verilog and VHDL
Constraints File	Xilinx Design Constraint (XDC)
Simulation Model	Verilog or VHDL source HDL Model
Supported S/W Driver	See the *10 Gigabit Ethernet Subsystem Product Guide* (PG157) [Ref 3]
Tested Design Flows(5)	
Design Entry	Vivado® Design Suite
Simulation	For supported simulators, see the Xilinx Design Tools: Release Notes Guide.
Synthesis	Vivado Synthesis
Support	
Release Notes and Known Issues	Master Answer Record: 54669
All Vivado IP Change Logs	Master Vivado IP Change Log: 72775
Xilinx Support web page	

Notes:

1. Some packages and speedgrades might not support the 10 Gb/s line rate. For a complete list of supported devices, see the Vivado IP catalog. For new designs in the UltraScale/ UltraScale+™ portfolio, see the 10G/25G Ethernet Subsystem webpage.
2. For 10GBASE-KR channel analysis, contact your local Xilinx representative.
3. -2, -2L or -3.
4. GTHE2 transceivers only.
5. For the supported versions of third-party tools, see the Xilinx Design Tools: Release Notes Guide.

Overview

Navigating Content by Design Process

Xilinx® documentation is organized around a set of standard design processes to help you find relevant content for your current development task. This document covers the following design processes:

- **Hardware, IP, and Platform Development**: Creating the PL IP blocks for the hardware platform, creating PL kernels, subsystem functional simulation, and evaluating the Vivado timing, resource and power closure. Also involves developing the hardware platform for system integration. Topics in this document that apply to this design process include:
  - Port Descriptions
  - Register Space
  - Clocking
  - Resets
  - Interfacing to the Core
  - Special Design Considerations
  - Customizing and Generating the Core
  - Detailed Example Design

Core Overview

10GBASE-R/KR is a 10 Gb/s serial interface. It is intended to provide the Physical Coding Sublayer (PCS) and Physical Medium Attachment (PMA) functionality between the 10 Gigabit Media Independent Interface (XGMII) interface on a Ten Gigabit Ethernet Media Access Controller (MAC) and a Ten Gigabit Ethernet network physical-side interface (PHY).

The 10GBASE-KR core is distinguished from the 10GBASE-R core by the addition of a Link Training block as well as optional Auto-Negotiation (AN) and Forward Error Correction (FEC) features, to support a 10 Gb/s data stream across a backplane.

10GBASE-R

For Zynq®-7000, UltraScale™, Virtex®-7, and Kintex®-7 devices, all of the PCS and management blocks illustrated are implemented in logic, except for part of the Gearbox and SERDES. Figure 1-1 shows the architecture.

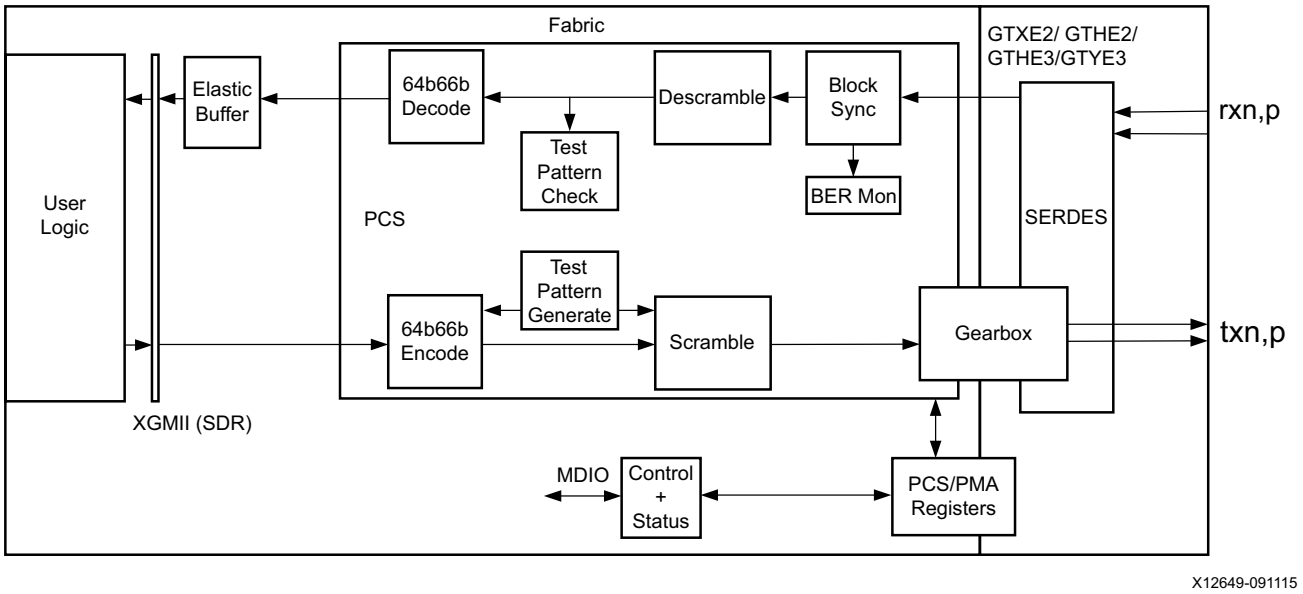

Figure 1-1: **Implementation of the 10GBASE-R Core**

The major functional blocks include the following:

- XGMII interface, designed for simple attachment of 10 Gigabit Ethernet MAC
- Transmit path, including scrambler, 64b/66b encoder and Gearbox
- Receive path, including block synchronization, descrambler, decoder and BER (Bit Error Rate) monitor
- Elastic buffer in the receive datapath.

  The elastic buffer is 32 words deep (1 word = 64bits data + 8 control). (For 32-bit 10GBASE-R cores, the elastic buffer is twice the depth and half the width, but has the same properties.) If the buffer empties, local fault codes are inserted instead of data. This allows you to collect up to 64 clock correction (CC) sequences before the buffer overflows (and words are dropped). The buffer normally fills up to one half and then deletes CC sequences when over half full, and inserts CC sequences when under one half full. So from a half-full state, you can (conservatively) accept an extra 360 KB of data (that is, receiving at +200 ppm) without dropping any. From a half-full state you can cope with another 360 KB of data without inserting local faults (for –200 ppm).

- Test pattern generation and checking
- Serial interface to optics

- Management registers (PCS/PMA) with optional MDIO interface

10GBASE-KR

Figure 1-2 illustrates a block diagram of the 10GBASE-KR core implementation. The major functional blocks include the following:

- XGMII interface, designed for simple attachment of 10 Gigabit Ethernet MAC
- Transmit path, including scrambler, 64b/66b encoder, FEC, AN and Training
- Receive path, including block synchronization, descrambler, decoder and BER (Bit Error Rate) monitor, FEC, AN and Training
- Elastic buffer in the receive datapath.

  The elastic buffer is 32 words deep (1 word = 64bits data + 8 control). If the buffer empties, local fault codes are inserted instead of data. This allows you to collect up to 64 clock correction (CC) sequences before the buffer overflows (and words are dropped). The buffer normally fills up to one half and then deletes CC sequences when over half full, and inserts CC sequences when under one half full. So from a half-full state, you can (conservatively) accept an extra 360 KB of data (that is, receiving at +200 ppm) without dropping any. From a half-full state you can cope with another 360 KB of data without inserting local faults (for –200 ppm).
- Test pattern generation and checking
- Serial interface to backplane connector
- Management registers (PCS/PMA) with optional MDIO interface

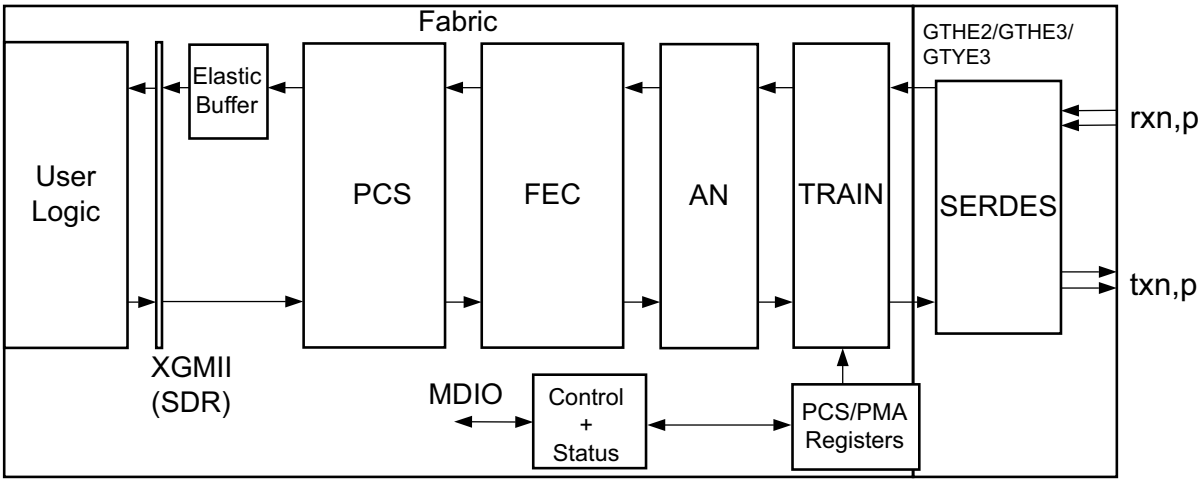

Figure 1-2: **Implementation of the BASE-KR Core**

Applications

Figure 1-3 shows a typical Ethernet system architecture and the core within it. The MAC and all the blocks to the right are defined in *IEEE Std 802.3* [Ref 1].

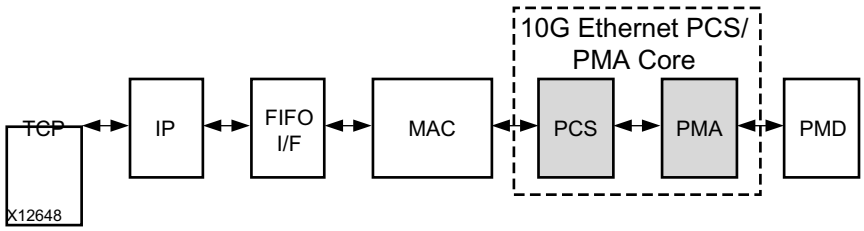

Figure 1-3: **Typical Ethernet System Architecture**

Figure 1-4 shows the 10G Ethernet PCS/PMA core connected on one side to a 10G Ethernet MAC and on the other to an optical module (BASE-R) or backplane (BASE-KR) using a serial interface. The optional WAN Interface Sublayer (WIS) part of the 10GBASE-R standard is not implemented in this core.

The 10G Ethernet PCS/PMA core is designed to be attached to the Xilinx IP 10G Ethernet MAC core over XGMII. More details are provided in Chapter 3, Designing with the Core.

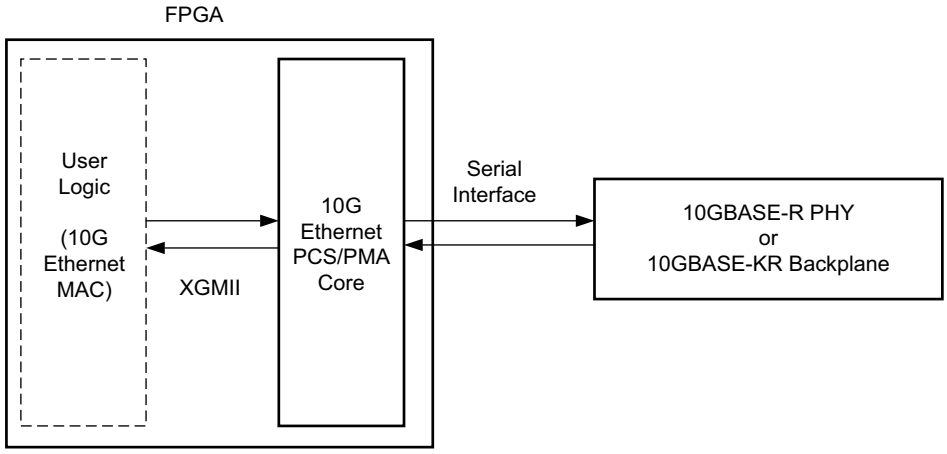

Figure 1-4: **Core Connected to MAC Core Using XGMII Interface**

Unsupported Features

The following features are not supported in this release of the core.

While the Training Protocol is supported natively by the core, no logic is provided that controls the far-end transmitter adaptation based on analysis of the received signal quality. This is because extensive testing has shown that to be unnecessary.

However, a training interface is provided on the core that allows access to all core registers and to the DRP port on the transceiver. You can employ this interface to implement your own Training Algorithm for 10GBASE-KR, if required.

Licensing and Ordering

License Checkers

If the IP requires a license key, the key must be verified. The Vivado design tools have several license check points for gating licensed IP through the flow. If the license check succeeds, the IP can continue generation. Otherwise, generation halts with error. License checkpoints are enforced by the following tools:

- Vivado Synthesis
- Vivado Implementation
- `write_bitstream` (Tcl command)

IMPORTANT: *IP license level is ignored at checkpoints. The test confirms a valid license exists. It does not check IP license level.*

License Type

10G Ethernet PCS/PMA (10GBASE-R)

This Xilinx LogiCORE™ IP module is provided at no additional cost with the Xilinx Vivado® Design Suite under the terms of the Xilinx End User License. Information about this and other Xilinx LogiCORE IP modules is available at the Xilinx Intellectual Property page. For information about pricing and availability of other Xilinx LogiCORE IP modules and tools, contact your local Xilinx sales representative.

For more information, visit the 10 Gigabit Ethernet PCS/PMA (10GBASE-R) product web page.

10G Ethernet PCS/PMA with FEC/Auto-Negotiation (10GBASE-KR)

This Xilinx LogiCORE IP module is provided under the terms of the Xilinx Core License Agreement. The module is shipped as part of the Vivado Design Suite. For full access to all core functionalities in simulation and in hardware, you must purchase a license for the core. Contact your local Xilinx sales representative for information about pricing and availability.

For more information, visit the 10 Gigabit Ethernet PCS/PMA with FEC/Auto-Negotiation (10GBASE-KR) product web page. The 10G/25GBASE-KR/CR license key is bundled with this product. For more information, visit the 10G/25G Ethernet Subsystem product web page.

Information about this and other Xilinx LogiCORE IP modules is available at the Xilinx Intellectual Property page. For information on pricing and availability of other Xilinx LogiCORE IP modules and tools, contact your local Xilinx sales representative.

Product Specification

Standards

The 10GBASE-R/KR core is designed to the standard specified in clauses 45, 49, 72, 73, and 74 of the 10 Gigabit Ethernet specification IEEE Std 802.3 [Ref 1].

Performance

Transceiver Latency

See the *7 Series Transceivers User Guide* (UG476) [Ref 4], the *UltraScale Architecture GTH Transceivers User Guide* (UG576) [Ref 5], and the *UltraScale Architecture GTY Transceivers User Guide* (UG578) [Ref 6] for information on the transceiver latency.

64-Bit Data Width

Latency

These measurements are for the core only; they do not include the latency through the transceiver. The latency through the transceiver can be obtained from the relevant user guide.

Transmit Path Latency

As measured from the input port `xgmii_txd[63:0]` of the transmitter side XGMII (until that data appears on `gt_txd[31:0]` on the transceiver interface), the latency through the 7 series core for the XGMII interface configuration in the transmit direction is 20 periods of `txoutclk`. When the optional FEC functionality is included in the core and enabled, this increases to 26 periods of `txoutclk`.

Measuring in the same way for an UltraScale™ device for 10GBASE-R, the transmit latency is six periods of the 156.25 MHz transmit clock, which increases to 12 periods when FEC is included and enabled. The latency for UltraScale devices for 10GBASE-KR is 20 periods of

txoutclk. When the optional FEC functionality is included in the core and enabled, this increases to 26 periods of txoutclk.

Receive Path Latency

Latency in the receive direction is variable and depends mainly on the fill level of the receive elastic buffer.

Measured from the input into the core on gt_rxd[31:0] until the data appears on xgmii_rxd[63:0] of the receiver side XGMII interface, the latency through the 7 series core in the receive direction is nominally equal to 1831 UI, or 27.75 cycles of coreclk, increasing to 2723 UI, or 41.26 cycles of coreclk when the elastic buffer is at its fullest possible level. The exact latency depends on sync bit alignment position and data positioning within the transceiver 4-byte interface. For UltraScale devices in 10GBASE-R configuration, excluding the elastic buffer, the latency through the core in the receive direction is nominally equal to seven cycles of the 156.25 MHz receive clock. The latency through the elastic buffer is the same as calculated for 7 series devices (the number of cycles is for the 156.25 MHz receive clock). For UltraScale devices in 10GBASE-KR configuration the latency is same as calculated for 7 series cores.

When the optional FEC functionality is included in the core the UltraScale core has a single extra cycle of the 156.25 MHz receive clock and this increases for all devices by 70 cycles of rxrecclk_out when FEC is enabled and if error reporting to the PCS layer is enabled, there is an extra 66 cycles of rxrecclk_out latency.

32-Bit Data Width

Latency

These measurements are for the core only; they do not include the latency through the transceiver. The latency through the transceiver can be obtained from the relevant user guide. There is a margin of error of 33 UI (a single 32-bit XGMII word) with these numbers.

Transmit Path Latency

As measured from the input port xgmii_txd[31:0] of the transmitter side XGMII (until that data appears on gt_txd[31:0] on the transceiver interface), the latency through the core for the XGMII interface configuration in the transmit direction for 7 series devices is 14 periods of txoutclk. For UltraScale devices this is eight periods of the 312.5 MHz transmit clock.

Receive Path Latency

Latency in the Receive direction is variable and depends mainly on the fill level of the receive elastic buffer.

Measured from the input into the core on `gt_rxd[31:0]` until the data appears on `xgmii_rxd[31:0]` of the receiver side XGMII interface, the latency through the core in the receive direction for 7 series devices is nominally equal to 1472 UI, or 44.6 cycles of `coreclk`, increasing to ~72 cycles of `coreclk` when the elastic buffer is at its fullest possible level. The exact latency depends on sync bit alignment position and data positioning within the transceiver 4-byte interface. For UltraScale devices, excluding the elastic buffer, the latency is eight cycles of the 312.5 MHz receive clock.

Resource Utilization

For details about resource utilization, visit Performance and Resource Utilization.

Port Descriptions

This section provides information about the ports for the XGMII interface and for the serial data interface. Additionally, information is provided about the ports for the management interface (MDIO) and its alternative, the vector-based configuration and status signals. Information is also provided about the clock and reset signals, the DRP training interface ports, the transceiver debug ports and miscellaneous core signals.

XGMII Interface Signals

For 10GBASE-R, the core provides the option of a 64-bit or a 32-bit XGMII data width (the selected data width also applies internally to the core). For the 10GBASE-KR option, only the 64-bit option is available.

64-Bit XGMII

When the 64-bit datapath is selected, the MAC (or client) side of the core has a 64-bit datapath plus eight control bits implementing an XGMII interface. Table 2-1 defines the signals, which are all synchronous to a 156.25 MHz clock source; the relevant clock port is dependent upon the family and core permutation. It is designed to be connected to either user logic within the FPGA or, by using SelectIO™ technology Double Data Rate (DDR) registers in your own top-level design, to provide an external 32-bit DDR XGMII, defined in clause 46 of *IEEE Std 802.3*. TX clock source and RX clock source are defined in Table 3-1.

Table 2-1: **MAC-Side Interface Ports**

Signal Name	Direction	Clock Domain	Description
xgmii_txd[63:0]	In	TX clock source	64-bit transmit data word
xgmii_txc[7:0]	In	TX clock source	8-bit transmit control word

Table 2-1: **MAC-Side Interface Ports**

Signal Name	Direction	Clock Domain	Description
xgmii_rxd[63:0]	Out	RX clock source	64-bit receive data word
xgmii_rxc[7:0]	Out	RX clock source	8-bit receive control word

32-bit XGMII

When the 32-bit datapath is selected, the MAC (or client) side of the core has a 32-bit datapath plus four control bits implementing an XGMII interface. Table 2-2 defines the signals, which are all synchronous to a 312.5 MHz clock source; the relevant clock port is dependent upon the family and core permutation. It is designed to be connected to user logic within the FPGA. TX clock source and RX clock source are defined in Table 3-1.

Table 2-2: **MAC-Side Interface Ports**

Signal Name	Direction	Clock Domain	Description
xgmii_txd[31:0]	In	TX clock source	32-bit transmit data word
xgmii_txc[3:0]	In	TX clock source	4-bit transmit control word
xgmii_rxd[31:0]	Out	RX clock source	32-bit receive data word
xgmii_rxc[3:0]	Out	RX clock source	4-bit receive control word

Serial Data Ports

The serial data ports should be connected to the PMD which is either an optical module or a backplane.

Table 2-3: **Serial Data Ports**

Signal Name	Direction	Description
txn, txp	Out	Serial data to optics/backplane
rxn, rxp	In	Serial data from optics/backplane

Optical Module Interface Ports

The status and control interface to an attached optical module is a simple pin-to-pin interface on those pins that need to be connected. The signals are described in Table 2-4. See Chapter 3, Designing with the Core for details on connecting an optical module to the 10GBASE-R core. For 10GBASE-KR, it is recommended to tie `signal_detect` to 1, `tx_fault` to 0, and leave `tx_disable` unconnected.

Table 2-4: **Optical Module Interface Ports**

Signal Name	Direction	Description
signal_detect	In	Status signal from attached optical module.(1)
tx_fault	In	Status signal from attached optical module.(2)(3)
tx_disable	Out	Control signal to attached optical module

Notes:

1. When an optical module is present, the logical NOR of MODDEF0 and LOS (Loss of Signal) outputs should be used to create the `signal_detect` input to the core.
2. This signal is not connected inside this version of the core. You should handle these inputs and reset your design as required.
3. Connect to `SFP+ tx_fault` signal, or `XFP MOD_NR` signal, depending on which is present.

Management Interface (MDIO) Ports

The optional MDIO interface is a simple low-speed two-wire interface for management of the 10G Ethernet PCS/PMA core, consisting of a clock signal and a bidirectional data signal. The interface is defined in clause 45 of the *IEEE 802.3-2012* standard.

In this core, the MDIO interface is an optional block. If implemented, the bidirectional data signal MDIO is implemented as three unidirectional signals. These can be used to drive a 3-state buffer either in the FPGA IOB or in a separate device.

Where a single point-to-point connection to a MAC is required, connect `mdio_in` on the 10G Ethernet PCS/PMA core to `mdio_out` on the MAC and vice versa, leaving the `mdio_tri` output unconnected. The `mdio_tri` signal is not required for a point-to-point connection.

Table 2-5: **MDIO Management Interface Ports**

Signal Name	Direction	Description
mdc	In	Management clock
mdio_in	In	MDIO Input
mdio_out	Out	MDIO Output
mdio_tri	Out	MDIO 3-state control. 1 disconnects the output driver from the MDIO bus.
prtad[4:0]	In	MDIO port address. When multiple MDIO-managed ports appear on the same bus, this input can be used to set the address of each port.

Clock and Reset Ports

The clock and reset ports are described in this section for both Shared Logic options.

Shared Logic Included in Example Design

If **Include Shared Logic in example design** is selected during the core customization, circuits for clock and reset management are included in the top-level example design sources. These can include clock generators, reset synchronizers, or other circuits that can be useful in your particular application. Table 2-6 shows the ports on the core that are associated with clocks and resets.

Table 2-6: **Clock and Reset Ports—Shared Logic in Example Design**

Signal Name	Direction	Description
coreclk	In	This clock is used to clock the TX datapath and management logic in 7 series devices and for 10GBASE-KR in UltraScale devices. In UltraScale devices for 10GBASE-R this clock is used only as a free running clock source for reset logic associated with the transceiver.
txusrclk, txusrclk2	In	Connected to the TXUSRCLK and TXUSRCLK2 input ports of the transceiver, these clocks are derived from the TXOUTCLK from the transceiver. In UltraScale devices for 10GBASE-R txusrclk2 is used to clock the transmit datapath and management logic.
dclk(1)	In	Management/DRP clock: this clock can be any rate that is valid for the applicable transceiver DRPCLK.
areset	In	Asynchronous (master) reset(2)
txoutclk	Out	TXOUTCLK from the transceiver used in the shared clock generation logic
rxrecclk_out	Out	RXOUTCLK from the transceiver
areset_coreclk	In	Synchronous reset in the coreclk domain
gttxreset	In	Transceiver TX reset signal in the coreclk domain
gtrxreset	In	Transceiver RX reset signal in the coreclk domain
qplllock	In	Transceiver QPLL Lock signal for 7 series devices
qplloutclk	In	Transceiver QPLL clock for 7 series devices
qplloutrefclk	In	Transceiver QPLL refclk for 7 series devices
qpll0lock	In	Transceiver QPLL lock signal for UltraScale devices
qpll0outclk	In	Transceiver QPLL clock for UltraScale devices
qpll0outrefclk	In	Transceiver QPLL refclk for UltraScale devices
reset_counter_done	In	Indication that 500 ns have passed after configuration was complete
tx_resetdone	Out	Transceiver TX reset-done
rx_resetdone	Out	Transceiver RX reset-done
reset_tx_bufg_gt	Out	Control from core to the BUFG_GT in the shared logic. Only for 64-bit datapath cores on UltraScale devices

Table 2-6: **Clock and Reset Ports—Shared Logic in Example Design *(Cont'd)***

Signal Name	Direction	Description
qpll0reset	Out	For UltraScale devices, a reset signal from the core to the QPLL located in the shared logic.

Notes:

1. For UltraScale devices the DCLK must be free-running and the frequency must be kept less than or equal to the maximum DRPCLK frequency specified for the transceiver type or the TXUSRCLK2 frequency, which is 156.25 MHz for 64-bit datapaths.
2. This reset also resets all management registers.

Shared Logic Included in Core

If **Include Shared Logic in core** is selected during core customization, most of the clocking and reset blocks are included within the core. Table 2-7 shows the ports on the core that are associated with these clocks and resets, which can be reused by other user logic or IP cores.

Table 2-7: **Clock and Reset Ports—Shared Logic in Core**

Signal Name	Direction	Description
refclk_p, refclk_n	In	Differential clock input (for transceiver)
dclk(1)	In	Management/DRP clock: this clock can be any rate that is valid for the applicable transceiver drpclk
reset	In	Asynchronous master reset(2)
resetdone_out	Out	Combined transceiver reset-done indication (in the coreclk_out domain)
coreclk_out	Out	This is used to clock the TX datapath and management logic in 7 series devices and for 10GBASE-KR in UltraScale devices. In UltraScale devices for 10GBASE-R this clock is used only as a free running clock source for reset logic associated with the transceiver.
qplllock_out	Out	Lock indication from QPLL block in core for 7 series devices
qplloutclk_out	Out	QPLL output clock from QPLL block in core for 7 series devices
qplloutrefclk_out	Out	QPLL output reference clock from QPLL block in core for 7 series devices
qpll0lock_out	Out	Lock indication from QPLL block in core for UltraScale architecture
qpll0outclk_out	Out	QPLL output clock from QPLL in core for UltraScale architecture
qpll0outrefclk_out	Out	QPLL output reference clk from QPLL in core for UltraScale architecture
txusrclk_out	Out	txusrclk from shared logic block in core
txusrclk2_out	Out	txusrclk2 from shared logic block in core. This is used to clock the TX datapath and management logic in UltraScale devices for 10GBASE-R designs.
rxrecclk_out	Out	RXOUTCLK from the GT output which is used to clock the receive datapath in the core.

Table 2-7: **Clock and Reset Ports—Shared Logic in Core *(Cont'd)***

Signal Name	Direction	Description
areset_datapathclk_out	Out	reset signal synchronized to the TX datapath clock which is the free-running coreclk_out for 7 series devices and 10GBASE-KR designs for UltraScale devices or the txusrclk2_out for UltraScale devices in 10GBASE-R designs.
areset_coreclk_out	Out	reset signal synchronized to the free-running coreclk_out; UltraScale devices only
gttxreset_out	Out	Signal that is used to reset the TX side of the transceiver, synchronized to coreclk_out
gtrxreset_out	Out	Signal that is used to reset the RX side of the transceiver, synchronized to coreclk_out
txuserrdy_out	Out	Transceiver control signal equivalent to the QPLLLOCK signal, synchronized to txusrclk2_out
reset_counter_done_out	Out	Indication that 500 ns have passed after configuration or 'master' reset, synchronized to coreclk_out

Notes:

1. For UltraScale devices the DCLK must be free-running and the frequency must be kept less than or equal to the maximum DRPCLK frequency specified for the transceiver type or the TXUSRCLK2 frequency, which is 156.25 MHz for 64-bit datapaths.
2. This reset also resets all management registers.

10GBASE-KR Training Interface

In Virtex-7, and UltraScale devices, in 10GBASE-KR only, an external training algorithm can optionally be connected to the training interface, which allows access to both the 802.3 registers in the core and the DRP registers in the transceiver. Table 2-8 shows the ports on the core that are associated with that interface.

Table 2-8: **Training Interface Ports**

Signal Name	Direction	Description
training_enable	In	Signal from external training algorithm to enable the training interface. This should not be confused with the IEEE register 1.150.1–Training Enable. A rising edge on training_enable initiates a register access.
training_addr[20:0]	In	Register address from training algorithm – bits [20:16] are the DEVAD for 802.3 registers
training_rnw	In	Read/Write_bar signal from training algorithm
training_ipif_cs	In	Select access to 802.3 registers in the core(1)
training_drp_cs	In	Select access to DRP registers in the transceiver
training_rddata[15:0]	Out	Read data from DRP or 802.3 registers
training_rdack	Out	Read Acknowledge signal to external training algorithm

Table 2-8: **Training Interface Ports *(Cont'd)***

Signal Name	Direction	Description
training_wrack	Out	Write Acknowledge signal to external training algorithm

Notes:

1. This signal has no meaning or effect when the core is created without an MDIO interface because all registers are exposed through the configuration and status vectors. This should be tied to 0 in that case. Access to transceiver DRP registers through the training interface is unaffected.

Figure 2-1 and Figure 2-2 show the timing diagrams for using the training interface to access internal core registers and transceiver registers through the DRP port. As shown, `training_drp_cs`, `training_ipif_cs`, and `training_enable` should be brought Low between read or write accesses. The clock for Figure 2-1 and Figure 2-2 can be determined from Table 3-1.

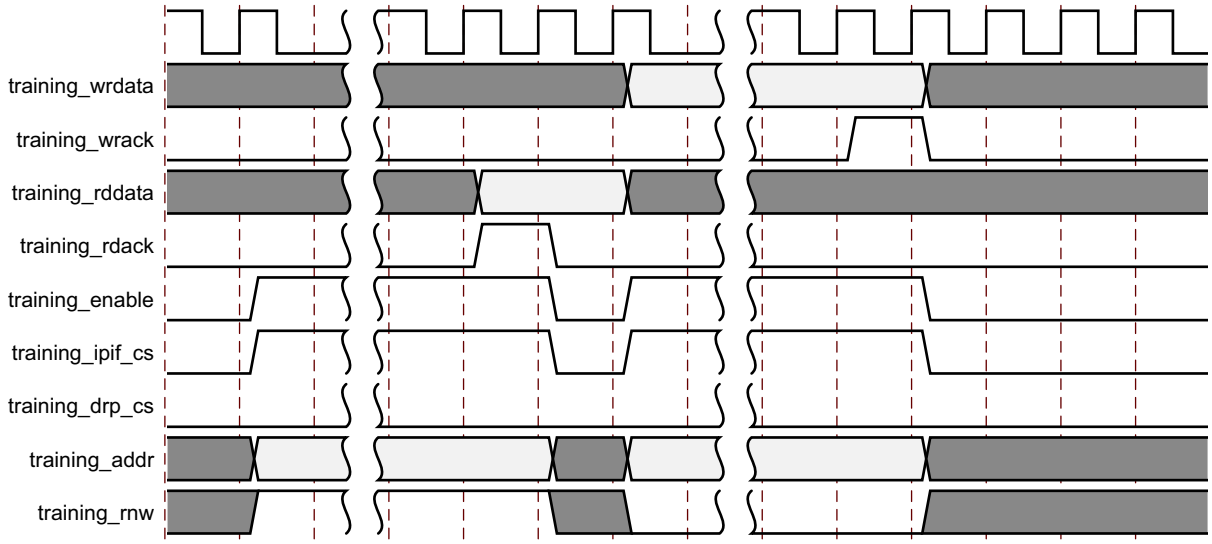

Figure 2-1: **Using the Training Interface to Access Internal Core Registers**

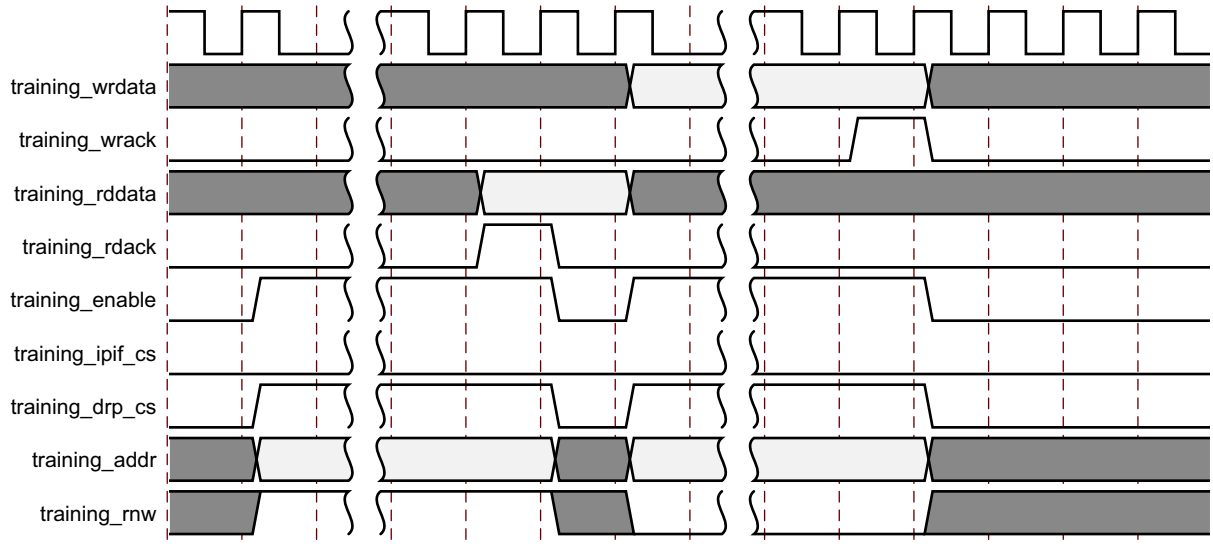

Figure 2-2: **Using the Training Interface to Access Transceiver Registers through the DRP Port**

DRP Interface Ports

Zynq-7000, Virtex-7, and Kintex-7 Devices

To facilitate the connection of user logic to the DRP interface of the transceiver, the interface between the core logic and the transceiver is brought out to an interface which can be connected to an external Arbiter block. The interface direct to the transceiver DRP is also provided.

If no user logic or arbiter is required, the `core_gt_drp_interface` can be connected directly to the `user_gt_drp_interface` and `drp_req` can be connected directly to `drp_gnt`.

All signals in Table 2-9 are synchronous to the `dclk` input of the core.

Table 2-9: **DRP Interface Ports**

Signal Name	Direction	Interface	Description
drp_req	Out	N/A	This active-High signal can be used on an external arbiter, to request and hold onto access to the DRP.
drp_gnt	In	N/A	This signal should be driven High when access is granted to the DRP by an external arbiter. If no external arbiter is present, connect this directly to the `drp_req` signal.
drp_daddr_o [15:0]	Out	core_gt_drp_interface	This vector is driven by the core and is eventually used on the `DADDR` port on the transceiver.

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

Signal Name	Direction	Interface	Description
drp_den_o	Out	core_gt_drp_interface	This signal is driven by the core and is eventually used on the DEN port on the transceiver.
drp_di_o[15:0]	Out	core_gt_drp_interface	This vector is driven by the core and is eventually used on the DI port on the transceiver.
drp_dwe_o	Out	core_gt_drp_interface	This signal is driven by the core and is eventually used on the DWE port on the transceiver.
drp_drpdo_i[15:0]	In	core_gt_drp_interface	This vector is driven by an external arbiter, or by drp_drpdo_o and is eventually used by the core.
drp_drdy_i	In	core_gt_drp_interface	This signal is driven by an external arbiter, or by drp_drdy_o and is eventually used by the core.
drp_daddr_i[15:0]	In	user_gt_drp_interface	This vector is driven by an external arbiter or by drp_daddr_o and is eventually used on the DADDR port on the transceiver.
drp_den_i	In	user_gt_drp_interface	This signal is driven by an external arbiter or by drp_den_o and is eventually used on the DEN port on the transceiver.
drp_di_i[15:0]	In	user_gt_drp_interface	This vector is driven by an external arbiter or by drp_di_o and is eventually used on the DI port on the transceiver.
drp_dwe_i	In	user_gt_drp_interface	This signal is driven by an external arbiter or by drp_dwe_o and is eventually used on the DWE port on the transceiver.
drp_drpdo_o[15:0]	Out	user_gt_drp_interface	This vector is driven by the DO port on the transceiver.
drp_drdy_o	Out	user_gt_drp_interface	This signal is driven by the DRDY port on the transceiver.

UltraScale Architecture

To facilitate the connection of user logic to the DRP interface of the transceiver, the interface between the core logic and the transceiver is brought out to an interface that can be connected to an external Arbiter block. The interface directly to the transceiver DRP is also provided.

If no user logic or arbiter is required, the core_to_gt_drp interface can be connected directly to the gt_drp interface and the drp_req can be connected directly to drp_gnt.

All signals in Table 2-10 are synchronous to the dclk input of the core.

Table 2-10: **DRP Interface Signals**

Signal Name	Direction	Interface	Description
drp_req	Out	N/A	This active-High signal can be used on an external arbiter to request and hold onto access to the DRP.
drp_gnt	In	N/A	This signal should be driven High when access is granted to the DRP by an external arbiter. If no external arbiter is present, connect this directly to the drp_req signal.
core_to_gt_drpaddr[15:0]	Out	core_to_gt_drp	This vector is driven by the core and is eventually used on the DADDR port on the transceiver.
core_to_gt_drpen	Out	core_to_gt_drp	This signal is driven by the core and is eventually used on the DEN port on the transceiver.
core_to_gt_drpdi[15:0]	Out	core_to_gt_drp	This vector is driven by the core and is eventually used on the DI port on the transceiver.
core_to_gt_drpwe	Out	core_to_gt_drp	This signal is driven by the core and is eventually used on the DWE port on the transceiver.
core_to_gt_drpdo[15:0]	In	core_to_gt_drp	This vector is driven by an external arbiter, or by gt_drpdo and is eventually used by the core.
core_to_gt_drprdy	In	core_to_gt_drp	This signal is driven by an external arbiter, or by gt_drprdy and is eventually used by the core.
gt_drpaddr[15:0]	In	gt_drp	This vector is driven by an external arbiter or by core_to_gt_drpaddr and is eventually used on the DADDR port on the transceiver.
gt_drpen	In	gt_drp	This signal is driven by an external arbiter or by core_to_gt_drpen and is eventually used on the DEN port on the transceiver.
gt_drpdi[15:0]	In	gt_drp	This vector is driven by an external arbiter or by gt_drpdi and is eventually used on the DI port on the transceiver.
gt_drpwe	In	gt_drp	This signal is driven by an external arbiter or by core_to_gt_drpwe and is eventually used on the DWE port on the transceiver.
gt_drpdo[15:0]	Out	gt_drp	This vector is driven by the DO port on the transceiver.
gt_drprdy	Out	gt_drp	This signal is driven by the DRDY port on the transceiver.

Miscellaneous Ports

The signals in Table 2-11 apply to all supported devices.

Table 2-11: **Miscellaneous Ports**

Signal Name	Direction	Description
core_status[7:0]	Out	Bit 0 = PCS Block Lock **10GBASE-R cores**: Bits [7:1] are reserved **10GBASE-KR cores**: Bit 1 = FEC Signal OK(1), Bit 2 = pmd_signal_detect (Training Done) (2) Bit 3 = AN Complete Bit 4 = AN Enable Bit 5 = an_link_up (3). Bits[7:6] are reserved All bits are synchronous to the TX clock source as defined in Table 3-1.
is_eval	Out	**10GBASE-KR only**: Constant output which is 1 if this is an Evaluation Licensed core
an_enable	In	**10GBASE-KR only**: Used to disable Auto-negotiation during simulation – normally tie this to 1. Only for cores with Optional Auto-negotiation block.
sim_speedup_control	In	Use this signal during simulation to short-cut through some timers. See Speeding up Simulation for more information
pma_pmd_type[2:0]	In	**10GBASE-R only:** Set this to a constant to define the PMA/PMD type as described in IEEE 802.3 section 45.2.1.6: 111 = 10GBASE-SR 110= 10GBASE-LR 101 = 10GBASE-ER

Notes:

1. This bit is equivalent to the FEC block lock if FEC is included in the core *and* FEC is enabled AND Training Done AND `signal_detect` AND `an_link_up`.
   If FEC is not included or is not enabled, this bit is equivalent to Training Done AND `signal_detect` AND `an_link_up`.
2. This is equivalent to Training Done AND `signal_detect`.
3. The latter two signals are required in the core to enable a switching of transceiver RX modes during auto-negotiation. When the optional auto-negotiation block is not included with the core, or is included but disabled by either the `an_enable` pin on the core (simulation-only) or by the management register 7.0.12, `an_link_up` (bit 5) is fixed to a constant 1 and bits 3 and 4 is a constant 0.

Speeding up Simulation

Direct control of some timers in the core is provided for use before and after implementation. To use the shorter timer values, drive `sim_speedup_control` Low until after GSR has fallen (typically after 100 ns of simulation time) and then High and hold it High.

To remove short-cut logic automatically, tie the port to either 0 or 1 before the final implementation stage. This allows the optimization step to remove the logic.

While tying the port off for final implementation is recommended, you can leave it connected to a pin on the device. As long as that pin is not driven Low and then High, the speedup values for the timers will never be used.

The timer that is speeded up with this control is the transceiver RX reset timer. This delays the assertion of RXUSERRDY which is reduced from 37 million UI to 50,000 UI. Also, for BASE-KR cores, the auto-negotiation Break Link Timer value is reduced from around 67 ms to just 6.4 µs.

Transceiver Debug Ports

If you select **Additional transceiver control and status ports** during core customization, the ports in Table 2-12 are available for Zynq-7000, Virtex-7, and Kintex-7 devices; for UltraScale devices the ports are listed in Table 2-13. Consult the relevant transceiver user guide or product guide for more information.

IMPORTANT: *The ports in the transceiver Control And Status Interface must be driven in accordance with the appropriate GT user guide. Using the input signals listed in Table 2-12 might result in unpredictable behavior of the core.*

Table 2-12: **Transceiver Debug Signals**

Signal Name	Direction	Clock Domain	Description
gt0_eyescanreset	In	Async	Eye Scan Reset control
gt0_eyescantrigger	In	rxrecclk_out	Eye Scan Trigger control
gt0_rxcdrhold	In	Async	CDR Hold control
gt0_txprbsforceerr	In	txusrclk2/ txusrclk2_out	Force a single TXPRBS error
gt0_txpolarity	In	txusrclk2/ txusrclk2_out	Switch the sense of txn and txp
gt0_rxpolarity	In	rxrecclk_out	Switch the sense of rxn and rxp
gt0_rxrate[2:0]	In	rxrecclk_out	RX Rate control
gt0_txprecursor [4:0]	In	Async	TX Precursor control (BASE-R only)
gt0_txpostcursor [4:0]	In	Async	TX Postcursor control (BASE-R only)
gt0_txdiffctrl [3:0]	In	Async	TX Differential Drive control (BASE-R only)
gt0_eyescandataerror	Out	Async	Eye Scan Data Error indication
gt0_txbufstatus[1:0]	Out	txusrclk2/ txusrclk2_out	Transceiver TX Buffer Status

Table 2-12: **Transceiver Debug Signals *(Cont'd)***

Signal Name	Direction	Clock Domain	Description
gt0_txpmareset	In	Async	Reset the transceiver TX PMA block
gt0_rxpmareset	In	Async	Reset the transceiver RX PMA block
gt0_txresetdone	Out	txusrclk2/ txusrclk2_out	Indication from transceiver TX side
gt0_rxresetdone	Out	rxrecclk_out	Indication from transceiver RX side
gt0_rxbufstatus[2:0]	Out	txusrclk2/ txusrclk2_out	Transceiver RX Buffer status indication
gt0_rxdfelpmreset	In	Async	Reset control for transceiver Equalizer
gt0_rxprbserr	Out	Async	Indication from transceiver PRBS Checker
gt0_dmonitorout[7:0](1)	Out	Async	Transceiver Digital Monitor outputs
gt0_rxpmaresetdone	Out	Async	Indication from transceiver (GTHE2 transceiver only)
gt0_rxlpmen	In	Async	Transceiver RX Equalizer control (BASE-R only)

Notes:
1. This output is 8-bits wide for the GTXE2 transceiver and 15 bits for the GTHE2 transceiver.

Table 2-13: **Transceiver Debug Signals - UltraScale Devices**

Signal Name	Direction	Clock Domain	Description
gt_txpcsreset	In	Async	Reset the transceiver TX PCS block
gt_txpmareset	In	Async	Reset the transceiver TX PMA block
gt_rxpmareset	In	Async	Reset the transceiver RX PMA block
gt_txresetdone	Out	txusrclk2/ txusrclk2_out	Indication from the transceiver TX side
gt_rxresetdone	Out	rxrecclk_out	Indication from the transceiver RX side
gt_rxpmaresetdone	Out	Async	Indication from the transceiver
gt_txbufstatus[1:0]	Out	txusrclk2/ txusrclk2_out	Transceiver TX Buffer status indication
gt_rxbufstatus[2:0]	Out	rxrecclk_out	Transceiver RX Buffer status indication

Table 2-13: **Transceiver Debug Signals - UltraScale Devices *(Cont'd)***

Signal Name	Direction	Clock Domain	Description
gt_rxrate[2:0]	In	rxrecclk_out	Transceiver RX Rate control
gt_eyescantrigger	In	rxrecclk_out	Eye Scan Trigger control
gt_eyescanreset	In	Async	Eye Scan Reset control
gt_eyescandataerror	Out	Async	Transceiver Eye Scan Data Error indication
gt_rxpolarity	In	rxrecclk_out	Transceiver RX Polarity control
gt_txpolarity	In	txusrclk2/ txusrclk2_out	Transceiver TX Polarity control
gt_rxdfelpmreset	In	Async	Transceiver Equalizer Reset control
gt_txprbsforceerr	In	txusrclk2/ txusrclk2_out	Transceiver PRBS Generation control
gt_rxprbserr	Out	rxrecclk_out	Indication from transceiver PRBS Checker
gt_rxcdrhold	In	Async	Transceiver RX CDR control
gt_dmonitorout[17:0]	Out	Async	Transceiver Digital Monitor outputs
gt_rxlpmen	In	Async	Transceiver RX Equalizer control (BASE-R only)
gt_txprecursor[4:0]	In	Async	Transceiver Precursor control (BASE-R only)
gt_txpostcursor[4:0]	In	Async	Transceiver Postcursor control (BASE-R only)
gt_txdiffctrl[3:0](1)	In	Async	Transceiver Output Level control (BASE-R only)
gt_pcsrsvdin[15:0]	In	Async	Bit2 of this vector can be used to asynchronously reset the DRP bus on UltraScale device GT_CHANNEL blocks. All other bits are tied to 0.
gt_txoutclksel[2:0]	In	Async	Transceiver TXOUTCLK select
gt_loopback	In	Async	Transceiver loopback selection
gt_rxprbslocked	Out	rxrecclk_out	Indication from transceiver PRBS Checker
gt_txpd[1:0] txusrclk2	In	Async based on gt_txpdelecidlemode	Transceiver TX Power-down
gt_rxpd[1:0]	In	Async	Transceiver RX Power-down

Table 2-13: **Transceiver Debug Signals - UltraScale Devices *(Cont'd)***

Signal Name	Direction	Clock Domain	Description
gt_txelecidle	In	txusrclk2/ Async based on gt_txpdelecidlemode	Forces TXP and TXN to Common mode
gt_txpdelecidlemode	In	Async	Determines if gt_txelecidle and gt_txpd should be synchronous or asynchronous signals
gt_cpllpd	In	Async	Powers down and resets CPLL for power savings
gt_qpll0pd/gt_qpll1_pd	In	Async	Active-High signal that powers down QPLL0 and QPLL1 for power savings

Notes:
1. This input is 4 bits wide for the GTHE3 transceiver but 5 bits wide for the GTYE3/GTHE4/GTYE4 transceiver.

Configuration and Status Signals

If the 10GBASE-R/KR core is generated without an MDIO interface, the key configuration and status information is carried on simple bit vectors, described in Table 2-14. Neither vector is completely populated so the actual number of pins required is much lower than the maximum widths of the vectors. For the status vector, correct default values are provided for all bits in the associated IEEE registers. See Table 2-15 to Table 2-18.

Table 2-14: **Configuration and Status Vectors**

Signal Name	Direction	Description
configuration_vector[535:0]	In	Configures the PCS/PMA registers
status_vector[447:0]	Out	Reflects recent status of PCS/PMA registers

See the Register Space section for information about the registers emulated with these configuration and status vectors.

Some IEEE registers are defined as set/clear-on-read, and because there is no read when using the configuration and status vectors, special controls have been provided to imitate that behavior. See Figure 2-3 and Figure 2-4.

BASE-R

Table 2-15 shows the breakdown of the 10GBASE-R-specific configuration vector and Table 2-16 shows the breakdown of the status vector. Any bits not mentioned are assumed to be 0s. The TX clock source is defined in Table 3-1.

Table 2-15: **Configuration Vector - BASE-R**

Bit	IEEE Register Bit	Description	Clock Domain
0	1.0.0	PMA Loopback Enable	Async
15	1.0.15	PMA Reset (1)	TX clock source
16	1.9.0	Global PMD TX Disable	Async
110	3.0.14	PCS Loopback Enable	TX clock source
111	3.0.15	PCS Reset(1)	TX clock source
169:112	3.37–3.34	MDIO Register 3.34–37: 10GBASE-R Test Pattern Seed A0–3	TX clock source
233:176	3.41–3.38	MDIO Register 3.38–41: 10GBASE-R Test Pattern Seed B0–3	TX clock source
240	3.42.0	Data Pattern Select	TX clock source
241	3.42.1	Test Pattern Select	TX clock source
242	3.42.2	RX Test Pattern Checking Enable	TX clock source
243	3.42.3	TX Test Pattern Enable	TX clock source
244	3.42.4	PRBS31 TX Test Pattern Enable	TX clock source
245	3.42.5	PRBS31 RX Test Pattern Checking Enable	TX clock source
399:384	3.65535.15:0	125 µs timer control	Async(3)
512	(1.1.2) (2)	Set PMA Link Status	TX clock source
513	(1.8.11) (2) (1.8.10) (2)	Clear PMA/PMD Link Faults	TX clock source
516	(3.1.2) (2)	Set PCS Link Status	TX clock source
517	(3.8.11) (2) (3.8.10) (2)	Clear PCS Link Faults	TX clock source
518	(3.33) (2)	MDIO Register 3.33: 10GBASE-R Status 2	TX clock source
519	(3.43) (2)	MDIO Register 3.43: 10GBASE-R Test Pattern Error Counter	TX clock source

Notes:

1. These reset signals should be asserted for a single clock tick only.
2. Reset controls for the given registers. These control signals should be at least 2 cycles wide.
3. Typically constant.

Table 2-16: **Status Vector - BASE-R**

Bit	IEEE Register Bit	Description	Clock Domain
15	1.0.15	PMA Reset(1)	TX clock source
18	1.1.2	PMA/PMD RX Link Status (Latching Low)	TX clock source
23	1.1.7	PMA/PMD Fault(2)	TX clock source
32	1.8.0	PMA Loopback Ability	N/A(3)
39:37(4)	1.8.7:5	10GBASE-R Ability	N/A
40	1.8.8	Transmit Disable Ability	N/A
42	1.8.10	PMA/PMD RX Fault (Latching High)	TX clock source
43	1.8.11	PMA/PMD TX Fault (Latching High)	TX clock source
44	1.8.12	PMA/PMD RX Fault Ability	N/A
45	1.8.13	PMA/PMD TX Fault Ability	N/A
47	1.8.15	Device Responding (MDIO Register 1.8: 10G PMA/PMD Status 2)	N/A
48	1.10.0	Global PMD RX Signal Detect	TX clock source
207:192	1.65535	MDIO Register 1.65535: Core Version Info	N/A
223	3.0.15	PCS Reset(1)	TX clock source
226	3.1.2	PCS RX Link Status (Latching Low)	TX clock source
231	3.1.7	PCS Fault(2)	TX clock source
240	3.8.0	10GBASE-R Ability	N/A
250	3.8.10	PCS RX Fault (Latching High)	TX clock source
251	3.8.11	PCS TX Fault (Latching High)	TX clock source
255	3.8.15	Device Responding (MDIO Register 3.8: 10G PCS Status 2)	N/A
256	3.32.0	10GBASE-R PCS RX Locked	TX clock source
257	3.32.1	10GBASE-R PCS high BER	TX clock source
258	3.32.2	PRBS31 Support	N/A
268	3.32.12	10GBASE-R PCS RX Link Status	TX clock source
279:272	3.33.7:0	10GBASE-R PCS Errored Blocks Counter	TX clock source
285:280	3.33.13:8	10GBASE-R PCS BER Counter	TX clock source
286	3.33.14	Latched High RX high BER	TX clock source
287	3.33.15	Latched Low RX Block Lock	TX clock source
303:288	3.43.15:0	10GBASE-R Test Pattern Error Counter	TX clock source

Notes:

1. This signal should be asserted for at most 3 cycles of the associated clock.
2. This bit is a logical OR of two latching bits and so will exhibit latching behavior without actually being latching itself.
3. This bit is a constant and clock domain is not applicable.
4. These bits are only valid for 10GBASE-R cores.

BASE-KR

Table 2-17 shows the *additional* signals in the configuration vector which are specific to BASE-KR functionality. TX clock source is defined in Table 3-1.

Table 2-17: **Configuration Vector - BASE-KR**

Bit	IEEE Register Bit	Description	Clock Domain
32	1.150.0	Restart Training	TX clock source
33	1.150.1	Enable Training	Async
53:48	1.152.5:0	MDIO Register 1.152: 10GBASE-KR LP Coefficient Update (valid when 1.150.1 = 0)	Async
60	1.152.12	LP Coefficient Initialize (valid when 1.150.1 = 0)	Async
61	1.152.13	LP Coefficient Preset (valid when 1.150.1 = 0)	Async
64	1.171.0	Enable FEC(1)(2)	TX clock source
65	1.171.1	FEC signal errors to PCS(1)(6)	TX clock source
281	7.0.9	Restart Auto-negotiation(3)	TX clock source
284	7.0.12	Enable Auto-negotiation(3)	TX clock source
285	7.0.13	Extended Next Page Support(3)	TX clock source
287	7.0.15	Reset Auto-negotiation(3)	TX clock source
300:293	7.16.12:5	AN Advertisement Data D12..D5	TX clock source
301	7.16.13	AN Advertisement Data – Remote fault	TX clock source
303	7.16.15	AN Advertisement Data – Next Page	TX clock source
319:304	7.17.15:0	AN Advertisement Data – D31..D16	TX clock source
335:320	7.18.15:0	AN Advertisement Data – D47..D32	TX clock source
346:336	7.22.10:0	AN XNP – Message Unformatted Code Field	TX clock source
348	7.22.12	AN XNP Acknowledge 2	TX clock source
349	7.22.13	AN XNP Message Page	TX clock source
351	7.22.15	AN XNP Next Page	TX clock source
367:352	7.23.15:0	AN XNP Unformatted Code Field 1	TX clock source
383:368	7.24.15:0	AN XNP Unformatted Code Field 2	TX clock source
405:400	1.65520.5:0	MDIO Register: 1.65520: Vendor-Specific LD Training	TX clock source
412	1.65520.12	LD Training Initialize	TX clock source
413	1.65520.13	LD Training Preset	TX clock source
415	1.65520.15	Training Done	TX clock source
514	(1.173:172)(4)	MDIO Register 1.173: 10GBASE-R FEC Corrected Blocks (Upper) MDIO Register 1.172: 10GBASE-R FEC Corrected Blocks (Lower)	TX clock source

Table 2-17: **Configuration Vector - BASE-KR *(Cont'd)***

Bit	IEEE Register Bit	Description	Clock Domain
515	(1.175:174)(4)	MDIO Register 1.175: 10GBASE-R FEC Uncorrected Blocks (Upper) MDIO Register 1.174: 10GBASE-R FEC Uncorrected Blocks (Lower)	TX clock source
520	(7.1.2)(4)	Set AN Link Up/Down	TX clock source
521	(7.1.4)(4)	Clear AN Remote Fault	TX clock source
522	(7.1.6)(4)	Clear AN Page Received	TX clock source
523	(7.18:16)(5)	MDIO Register 7.16:17:18: AN Advertisement	TX clock source
524	(7.24:22)(5)	MDIO Register 7.22, 23, 24: AN XNP Transmit	TX clock source

Notes:

1. Only valid when the optional FEC block is included.
2. If FEC is enabled during auto-negotiation then this register bit is overridden by the auto-negotiation control of FEC. So even with this bit set to 0, if auto-negotiation results in FEC being enabled, FEC is enabled and cannot be disabled except by changing the auto-negotiation Base Page Ability bits 46 and 47, and re-negotiating the link. FEC can still be enabled explicitly by setting this bit to 1 which overrides whatever auto-negotiation decides.
3. Only valid when the optional AN block is included.
4. Reset controls for the given registers. These controls should be at least 2 cycles wide.
5. Toggle to load the AN Page data from the associated configuration vector bits.
6. If FEC Error Passing is enabled while FEC is enabled, errors will be seen temporarily. To avoid this, only enable Error Passing while FEC is disabled.

Table 2-18 shows the *additional* signals in the status vector which are specific to BASE-KR functionality.

Table 2-18: **Status Vector - BASE-KR**

Bit	IEEE Register Bit	Description	Clock Domain
41	1.8.9	Extended Abilities	N/A(1)
67:64	1.151.3:0	MDIO Register 1.151: 10GBASE-KR PMD Status	TX clock source
85:80	1.153.5:0	MDIO Register 1.153: 10GBASE-KR LP Status	TX clock source
95	1.153.15	LP Status Report Training Complete	TX clock source
101:96	1.152.5:0	MDIO Register 1.152: 10GBASE-KR LP Coefficient Update	TX clock source
108	1.152.12	LP Coefficient Initialize	TX clock source
109	1.152.13	LP Coefficient Preset	TX clock source
117:112	1.155.5:0	MDIO Register 1.155: 10GBASE-KR LD Status	TX clock source
127	1.155.15	LD Status Report Training Complete	TX clock source
159:128	1.173:172	MDIO Register 1.173: 10GBASE-R FEC Corrected Blocks (Upper)(2) MDIO Register 1.172: 10GBASE-R FEC Corrected Blocks (Lower)(2)	TX clock source

Table 2-18: **Status Vector - BASE-KR *(Cont'd)***

Bit	IEEE Register Bit	Description	Clock Domain
191:160	1.175:174	MDIO Register 1.175: 10GBASE-R FEC Uncorrected Blocks (Upper)(2) MDIO Register 1.174: 10GBASE-R FEC Uncorrected Blocks (Lower)(2)	TX clock source
319	7.0.15	AN Reset(3)	TX clock source
320	7.1.0	LP AN Capable(3)	TX clock source
322	7.1.2	AN Link Up/Down (latching Low)(3)	TX clock source
323	7.1.3	AN Ability(3)	N/A
324	7.1.4	AN Remote Fault (latching High)(3)	TX clock source
325	7.1.5	AN Complete(3)	TX clock source
326	7.1.6	AN Page Received (latching High)(3)	TX clock source
327	7.1.7	AN Extended Next Page Used(3)	TX clock source
383:336	7.21:19	MDIO Register 7.19, 20, 21: AN LP Base Page Ability(3)	TX clock source
394:384	7.25.10:0	AN LP XNP – Message Unformatted Code Field(3)	TX clock source
395	7.25.11	AN LP XNP – Toggle(3)	TX clock source
396	7.25.12	AN LP XNP – Acknowledge2(3)	TX clock source
397	7.25.13	AN LP XNP – Message Page(3)	TX clock source
399	7.25.15	AN LP XNP – Next Page(3)	TX clock source
415:400	7.26.15:0	AN LP XNP – Unformatted Code Field 1(3)	TX clock source
431:416	7.27.15:0	AN LP XNP – Unformatted Code Field 2(3)	TX clock source
432	7.48.0	Backplane AN Ability(3)	TX clock source
435	7.48.3	Backplane Ethernet Status – KR negotiated(3)	TX clock source
436	7.48.4	Backplane Ethernet Status – FEC negotiated(3)	TX clock source

Notes:

1. This bit is a constant and clock domain is not applicable.
2. Only valid when the optional FEC block is included.
3. Only valid when the optional AN block is included.

Bit 286 of the status vector is latching-High and is cleared Low by bit 518 of the `configuration_vector` port. Figure 2-3 shows how the status bit is cleared.

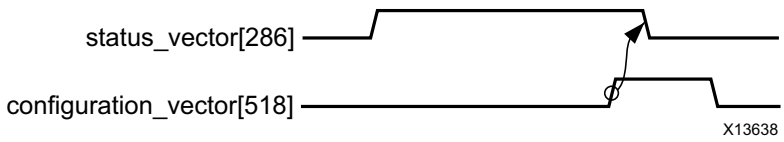

Figure 2-3: **Clearing the Latching-High Bits**

Bits 18, 226, and 287 of the `status_vector` port are latching-Low and set High by bits 512, 516, and 518 of the configuration vector. Figure 2-4 shows how the status bits are set.

Figure 2-4: **Setting the Latching-Low Bits**

Similarly, latching-High status vector bits 42 and 43 can be reset with configuration vector bit 513, and bits 250 and 251 can be reset with configuration vector bit 517.

- Status bits 285:272 are also reset using configuration vector bit 518
- Status bits 303:288 are reset using configuration vector bit 519

For Base-KR cores, similar reset behaviors exist for the following status vector bits:

- status vector bits 159:128 – cleared with configuration vector bit 514
- status vector bits 191:160 – cleared with configuration vector bit 515
- status vector bit 322 – set with configuration vector bit 520
- status vector bit 324 – cleared with configuration vector bit 521
- status vector bit 326 – cleared with configuration vector bit 522

Finally, configuration vector bits 335:293 and 383:336 each implement three registers which are normally latched into the core when the lower register is written, keeping the data coherent. Because there is no need for this behavior when the entire vector is exposed, these bits are latched into the core whenever configuration register bits 523 and 524 respectively are toggled High. The configuration vector bits used to set/reset the status vectors should be at least 2 clock cycles wide.

Register Space

This core implements registers which are further described in 802.3 Clause 45. If the core is generated without an MDIO interface, these registers are still implemented but accessed using configuration or status pins on the core. For example, register 1.0, bit 15 (PMA Reset) is implemented as bit 15 of the configuration vector and register 1.1, bit 7 (PMA/PMD Fault) is implemented as status vector bit 23. These mappings are described in Configuration and Status Signals.

If the core is configured as a 10GBASE-R PCS/PMA, it occupies MDIO Device Addresses 1 and 3 in the MDIO register address map, as shown in Table 2-19.

Table 2-19: **10GBASE-R PCS/PMA MDIO Registers**

Register Address	Register Name
1.0	MDIO Register 1.0: PMA/PMD Control 1
1.1	MDIO Register 1.1: PMA/PMD Status 1
1.4	MDIO Register 1.4: PMA/PMD Speed Ability
1.5, 1.6	MDIO Registers 1.5 and 1.6: PMA/PMD Devices in Package
1.7	MDIO Register 1.7: 10G PMA/PMD Control 2
1.8	MDIO Register 1.8: 10G PMA/PMD Status 2
1.9	MDIO Register 1.9: 10G PMD Transmit Disable
1.10	MDIO Register 1.10: 10G PMD Signal Receive OK
1.11 to 1.65534	Reserved
1.65535	MDIO Register 1.65535: Core Version Info
3.0	MDIO Register 3.0: PCS Control 1
3.1	MDIO Register 3.1: PCS Status 1
3.4	MDIO Register 3.4: PCS Speed Ability
3.5, 3.6	MDIO Registers 3.5 and 3.6: PCS Devices in Package
3.7	MDIO Register 3.7: 10G PCS Control 2
3.8	MDIO Register 3.8: 10G PCS Status 2
3.9 to 3.31	Reserved
3.32	MDIO Register 3.32: 10GBASE-R Status 1
3.33	MDIO Register 3.33: 10GBASE-R Status 2
3.34–37	MDIO Register 3.34–37: 10GBASE-R Test Pattern Seed A0–3
3.38–41	MDIO Register 3.38–41: 10GBASE-R Test Pattern Seed B0–3
3.42	MDIO Register 3.42: 10GBASE-R Test Pattern Control
3.43	MDIO Register 3.43: 10GBASE-R Test Pattern Error Counter
3.44 to 3.65534	Reserved
3.65535	MDIO Register 3.65535: 125 Microsecond Timer Control

If the core is configured as a 10GBASE-KR PCS/PMA, it occupies MDIO Device Addresses 1, 3 and optionally 7 in the MDIO register address map, as shown in Table 2-20.

Table 2-20: **10GBASE-KR PCS/PMA Registers**

Register Address	Register Name
1.0	MDIO Register 1.0: PMA/PMD Control 1
1.1	MDIO Register 1.1: PMA/PMD Status 1
1.4	MDIO Register 1.4: PMA/PMD Speed Ability

Table 2-20: **10GBASE-KR PCS/PMA Registers *(Cont'd)***

Register Address	Register Name
1.5, 1.6	MDIO Registers 1.5 and 1.6: PMA/PMD Devices in Package
1.7	MDIO Register 1.7: 10G PMA/PMD Control 2
1.8	MDIO Register 1.8: 10G PMA/PMD Status 2
1.9	MDIO Register 1.9: 10G PMD Transmit Disable
1.10	MDIO Register 1.10: 10G PMD Signal Receive OK
1.11 to 1.149	Reserved
1.150	MDIO Register 1.150: 10GBASE-KR PMD Control
1.151	MDIO Register 1.151: 10GBASE-KR PMD Status
1.152	MDIO Register 1.152: 10GBASE-KR LP Coefficient Update
1.153	MDIO Register 1.153: 10GBASE-KR LP Status
1.154	MDIO Register 1.154: 10GBASE-KR LD Coefficient Update
1.155	MDIO Register 1.155: 10GBASE-KR LD Status
1.170	MDIO Register 1.170: 10GBASE-R FEC Ability(1)
1.171	MDIO Register 1.171: 10GBASE-R FEC Control(1)
1.172 to 1.173	MDIO Register 1.172: 10GBASE-R FEC Corrected Blocks (Lower)(1) MDIO Register 1.173: 10GBASE-R FEC Corrected Blocks (Upper)(1)
1.174 to 1.175	MDIO Register 1.174: 10GBASE-R FEC Uncorrected Blocks (Lower)(1) MDIO Register 1.175: 10GBASE-R FEC Uncorrected Blocks (Upper)(1)
1.176 to 1.65519	Reserved
1.65520	MDIO Register: 1.65520: Vendor-Specific LD Training (vendor-specific register where Local Device Coefficient Updates are to be written by Training Algorithm)
1.65521 to 1.65534	Reserved
1.65535	MDIO Register 1.65535: Core Version Info
3.0	MDIO Register 3.0: PCS Control 1
3.1	MDIO Register 3.1: PCS Status 1
3.4	MDIO Register 3.4: PCS Speed Ability
3.5, 3.6	MDIO Registers 3.5 and 3.6: PCS Devices in Package
3.7	MDIO Register 3.7: 10G PCS Control 2
3.8	MDIO Register 3.8: 10G PCS Status 2
3.9 to 3.31	Reserved
3.32	MDIO Register 3.32: 10GBASE-R Status 1
3.33	MDIO Register 3.33: 10GBASE-R Status 2
3.34–37	MDIO Register 3.34–37: 10GBASE-R Test Pattern Seed A0–3
3.38–41	MDIO Register 3.38–41: 10GBASE-R Test Pattern Seed B0–3
3.42	MDIO Register 3.42: 10GBASE-R Test Pattern Control

Table 2-20: **10GBASE-KR PCS/PMA Registers *(Cont'd)***

Register Address	Register Name
3.43	MDIO Register 3.43: 10GBASE-R Test Pattern Error Counter
3.44 to 3.65534	Reserved
3.65535	MDIO Register 3.65535: 125 Microsecond Timer Control
7.0	MDIO Register 7.0: AN Control(2)
7.1	MDIO Register 7.1: AN Status(2)
7.16, 17, 18	MDIO Register 7.16:17:18: AN Advertisement(2)
7.19, 20, 21	MDIO Register 7.19, 20, 21: AN LP Base Page Ability(2)
7.22, 23, 24	MDIO Register 7.22, 23, 24: AN XNP Transmit(2)
7.25, 26, 27	MDIO Register 7.25, 26, 27: AN LP XNP Ability(2)
7.48	MDIO Register 7.48: Backplane Ethernet Status(2)

Notes:
1. For cores with optional FEC block.
2. For cores with optional AN block.

MDIO Register 1.0: PMA/PMD Control 1

Figure 2-5 shows the MDIO register 1.0: Physical Medium Attachment/Physical Medium Dependent (PMA/PMD) Control 1.

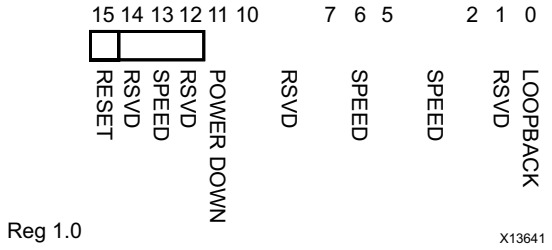

Figure 2-5: **PMA/PMD Control 1 Register**

Table 2-21 shows the PMA Control 1 register bit definitions.

Table 2-21: **PMA/PMD Control 1 Register**

Bits	Name	Description	Attributes	Default Value
1.0.15	Reset	1 = Block reset 0 = Normal operation The 10GBASE-R/KR block is reset when this bit is set to 1. It returns to 0 when the reset is complete.	R/W Self-clearing	0
1.0.14	Reserved	The block always returns 0 for this bit and ignores writes.	R/O	0

Table 2-21: **PMA/PMD Control 1 Register *(Cont'd)***

Bits	Name	Description	Attributes	Default Value
1.0.13	Speed Selection	The block always returns 1 for this bit and ignores writes.	R/O	1
1.0.12	Reserved	The block always returns 0 for this bit and ignores writes.	R/O	0
1.0.11	Power down	This bit has no effect.	R/W	0
1.0.10:7	Reserved	The block always returns 0 for these bits and ignores writes.	R/O	All 0s
1.0.6	Speed Selection	The block always returns 1 for this bit and ignores writes.	R/O	1
1.0.5:2	Speed Selection	The block always returns 0s for these bits and ignores writes.	R/O	All 0s
1.0.1	Reserved	The block always returns 0 for this bit and ignores writes.	R/O	All 0s
1.0.0	Loopback	1 = Enable PMA loopback mode (Near-end) 0 = Disable PMA loopback mode	R/W	0

MDIO Register 1.1: PMA/PMD Status 1

Figure 2-6 shows the MDIO register 1.1: PMA/PMD Status 1.

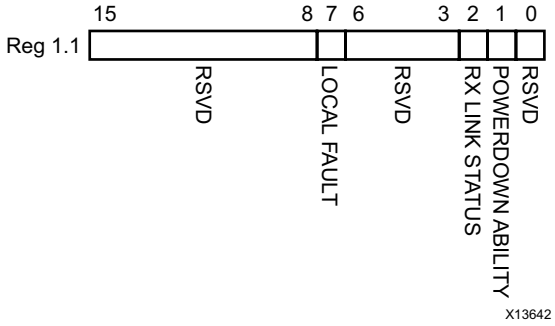

Figure 2-6: **PMA/PMD Status 1 Register**

Table 2-22 shows the PMA/PMD Status 1 register bit definitions.

Table 2-22: **PMA/PMD Status 1 Register**

Bits	Name	Description	Attributes	Default Value
1.1.15:8	Reserved	The block always returns 0 for this bit.	R/O	0
1.1.7	Local Fault	1 = Local Fault detected	R/O	0
1.1.6:3	Reserved	The block always returns 0 for this bit.	R/O	0

Table 2-22: **PMA/PMD Status 1 Register *(Cont'd)***

Bits	Name	Description	Attributes	Default Value
1.1.2	Receive Link Status	1 = Receive Link UP	R/O Latches Low	1
1.1.1	Power Down Ability	The block always returns 1 for this bit.	R/O	1
1.1.0	Reserved	The block always returns 0 for this bit.	R/O	0

MDIO Register 1.4: PMA/PMD Speed Ability

Figure 2-7 shows the MDIO register 1.4: PMA/PMD Speed Ability.

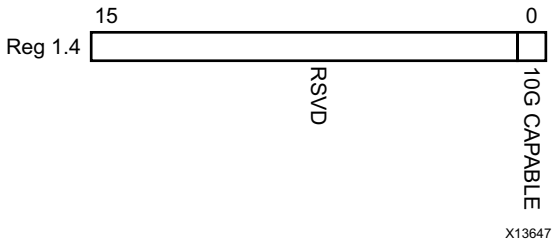

Figure 2-7: **PMA/PMD Speed Ability Register**

Table 2-23 shows the PMA/PMD Speed Ability register bit definitions.

Table 2-23: **PMA/PMD Speed Ability Register**

Bits	Name	Description	Attribute	Default Value
1.4.15:1	Reserved	The block always returns 0 for these bits and ignores writes.	R/O	All 0s
1.4.0	10G Capable	The block always returns 1 for this bit and ignores writes.	R/O	1

MDIO Registers 1.5 and 1.6: PMA/PMD Devices in Package

Figure 2-8 shows the MDIO registers 1.5 and 1.6: PMA/PMD Devices in Package.

Figure 2-8: **PMA/PMD Devices in Package Registers**

Table 2-24 shows the PMA/PMD Devices in Package registers bit definitions.

Table 2-24: **PMA/PMD Devices in Package Registers**

Bits	Name	Description	Attributes	Default Value
1.6.15	Vendor- specific Device 2 Present	The block always returns 0 for this bit.	R/O	0
1.6.14	Vendor-specific Device 1 Present	The block always returns 0 for this bit.	R/O	0
1.6.13	Clause 22 Extension Present	The block always returns 1 for this bit.	R/O	1
1.6.12:0	Reserved	The block always returns 0 for these bits.	R/O	All 0s
1.5.15:8	Reserved	The block always returns 0 for these bits.	R/O	All 0s
1.5.7	Auto-negotiation present	1 = optional AN block is included	R/O	1
1.5.6	TC Present	The block always returns 0 for this bit	R/O	0
1.5.5	DTE XS Present	The block always returns 0 for this bit.	R/O	0
1.5.4	PHY XS Present	The block always returns 0 for this bit.	R/O	0
1.5.3	PCS Present	The block always returns 1 for this bit.	R/O	1

Table 2-24: **PMA/PMD Devices in Package Registers *(Cont'd)***

Bits	Name	Description	Attributes	Default Value
1.5.2	WIS Present	The block always returns 0 for this bit.	R/O	0
1.5.1	PMA/PMD Present	The block always returns 1 for this bit.	R/O	1
1.5.0	Clause 22 Device Present	The block always returns 0 for this bit.	R/O	0

MDIO Register 1.7: 10G PMA/PMD Control 2

Figure 2-9 shows the MDIO register 1.7: 10G PMA/PMD Control 2.

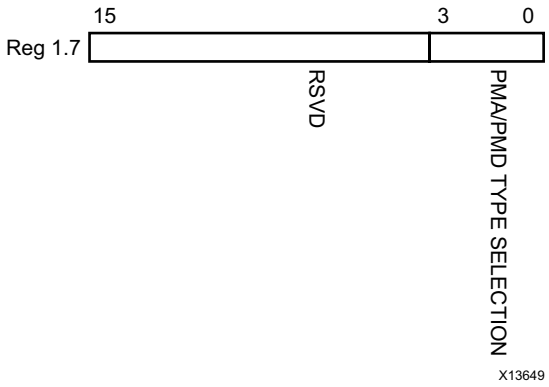

Figure 2-9: **10G PMA/PMD Control 2 Register**

Table 2-25 shows the PMA/PMD Control 2 register bit definitions.

Table 2-25: **10G PMA/PMD Control 2 Register**

Bits	Name	Description	Attributes	Default Value
1.7.15:4	Reserved	The block always returns 0 for these bits and ignores writes.	R/O	All 0s
1.7.3:0	PMA/PMD Type Selection	This returns the value 0xyz, where xyz is set from the top level core port pma_pmd_type vector.	R/W	(1)

Notes:

1. BASE-R: Set from pma_pmd_type port.
   BASE-KR: returns 0xB.

MDIO Register 1.8: 10G PMA/PMD Status 2

Figure 2-10 shows the MDIO register 1.8: 10G PMA/PMD Status 2.

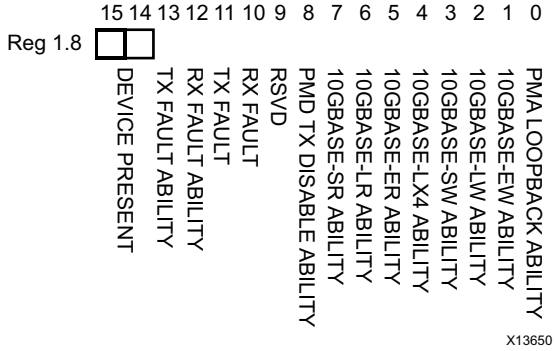

Figure 2-10: **10G PMA/PMD Status 2 Register**

Table 2-26 shows the PMA/PMD Status 2 register bit definitions.

Table 2-26: **10G PMA/PMD Status 2 Register**

Bits	Name	Description	Attributes	Default Value
1.8.15:14	Device Present	The block always returns 10 for these bits.	R/O	10
1.8.13	Transmit Local Fault Ability	The block always returns a 1 for this bit.	R/O	1
1.8.12	Receive Local Fault Ability	The block always returns a 1 for this bit.	R/O	1
1.8.11	Transmit Fault	1 = Transmit Fault detected	Latches High	0
1.8.10	Receive Fault	1 = Receive Fault detected	Latches High	0
1.8.9	Extended abilities	The block always returns 1 for this bit.	R/O	1
1.8.8	PMD Transmit Disable Ability	The block always returns 1 for this bit.	R/O	1
1.8.7	10GBASE-SR Ability	10GBASE-R only: Returns a 1 if pma_pmd_type port is set to 111	R/O	(1)
1.8.6	10GBASE-LR Ability	10GBASE-R only: Returns a 1 if pma_pmd_type port is set to 110	R/O	(1)
1.8.5	10GBASE-ER Ability	10GBASEe-R only: Returns a 1 if the pma_pmd_type port is set to 101	R/O	(1)
1.8.4	10GBASE-LX4 Ability	The block always returns 0 for this bit.	R/O	0
1.8.3	10GBASE-SW Ability	The block always returns 0 for this bit.	R/O	0

Table 2-26: **10G PMA/PMD Status 2 Register *(Cont'd)***

Bits	Name	Description	Attributes	Default Value
1.8.2	10GBASE-LW Ability	The block always returns 0 for this bit.	R/O	0
1.8.1	10GBASE-EW Ability	The block always returns 0 for this bit.	R/O	0
1.8.0	PMA Loopback Ability	The block always returns 1 for this bit.	R/O	1

Notes:
1. Depends on pma_pmd_type port.

MDIO Register 1.9: 10G PMD Transmit Disable

Figure 2-11 shows the MDIO 1.9 register: 10G PMD Transmit Disable.

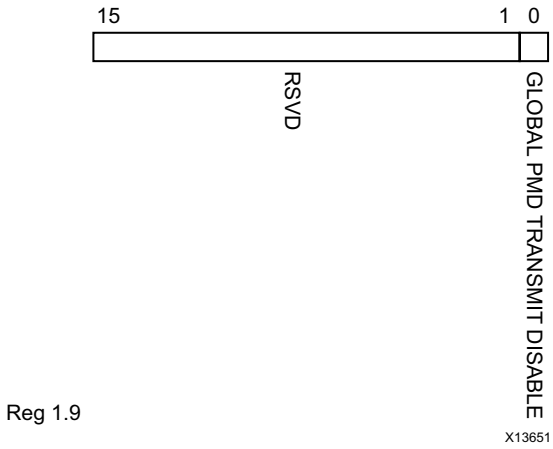

Figure 2-11: **10G PMD Transmit Disable Register**

Table 2-27: **10G PMD Transmit Disable Register**

Bits	Name	Description	Attributes	Default Value
1.9.15:1	Reserved	The block always returns 0 for these bits and ignores writes.	R/O	All 0s
1.9.0	Global PMD Transmit Disable	1 = Disable Transmit path (also sets transmit_disable pin) 0 = Enable Transmit path	R/W	0

MDIO Register 1.10: 10G PMD Signal Receive OK

Figure 2-12 shows the MDIO 1.10 register: 10G PMD Signal Receive OK.

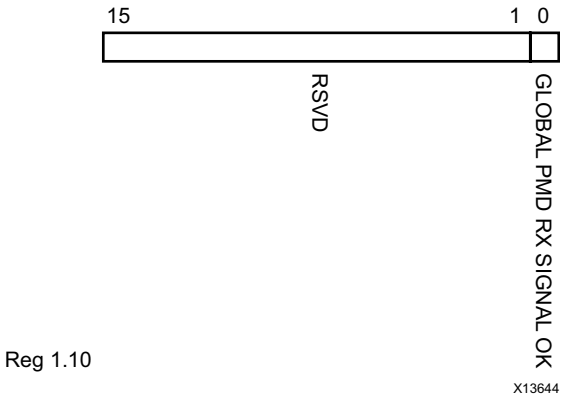

Figure 2-12: **10G PMD Signal Receive OK Register**

Table 2-28 shows the PMD Signal Receive OK register bit definitions.

Table 2-28: **10G PMD Signal Receive OK Register**

Bits	Name	Description	Attributes	Default Value
1.10.15:1	Reserved	The block always returns 0 for these bits.	R/O	0s
1.10.0	Global PMD receive signal detect	1 = Signal detected on receive 0 = Signal not detected on receive	R/O	N/A

MDIO Register 1.150: 10GBASE-KR PMD Control

Figure 2-13 shows the MDIO register 1.150: 10GBASE-KR PMD Control.

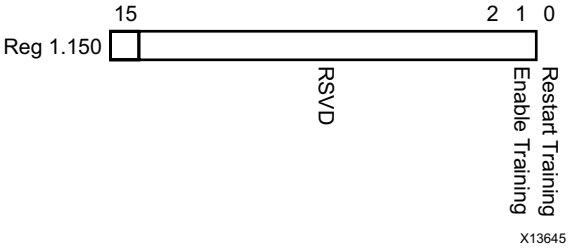

Figure 2-13: **10GBASE-KR PMD Control Register**

Table 2-29 shows the 10GBASE-KR PMD Control register bit definitions.

Table 2-29: **10GBASE-KR PMD Control Register**

Bits	Name	Description	Attributes	Default Value
1.150.15:2	Reserved	The block always returns 0 for this bit and ignores writes.	R/O	0
1.150.1	Training enable	1 = Enable the 10GBASE-KR start-up protocol 0 = Disable	R/W	0
1.150.0	Restart Training	1 = Reset the 10GBASE-KR start-up protocol 0 = Normal operation	R/W Self-clearing	0

MDIO Register 1.151: 10GBASE-KR PMD Status

Figure 2-14 shows the MDIO register 1.151: 10GBASE-KR PMD Status.

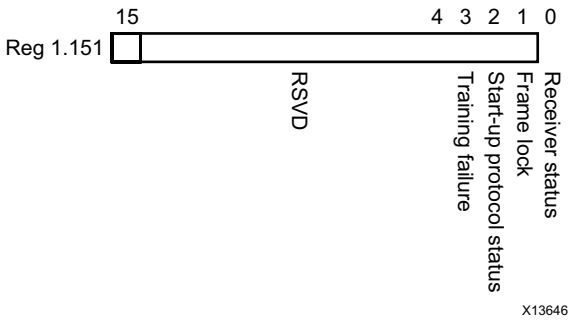

Figure 2-14: **10GBASE-KR PMD Status Register**

Table 2-30 shows the 10GBASE-KR PMD Status register bit definitions.

Table 2-30: **10GBASE-KR PMD Status Register**

Bits	Name	Description	Attributes	Default Value
1.151.15:4	Reserved	The block always returns 0 for this bit and ignores writes.	R/O	0
1.151.3	Training Failure	1 = Training Failure has been detected 0 = Not detected	R/O	0
1.151.2	Start-up Protocol status	1 = Start-up protocol in progress 0 = Protocol complete	R/O	0
1.151.1	Frame Lock	1 = Training frame delineation detected 0 = Not detected	R/O	0
1.151.0	Receiver status	1 = Receiver trained and ready to receive data 0 = Receiver training	R/O	0

MDIO Register 1.152: 10GBASE-KR LP Coefficient Update

Figure 2-15 shows the MDIO register 1.152: 10GBASE-KR LP Coefficient Update.

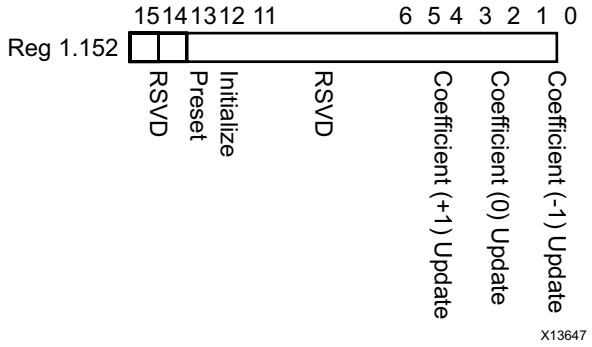

Figure 2-15: **10GBASE-KR LP Coefficient Update Register**

Table 2-31 shows the 10GBASE-KR LP coefficient update register bit definitions.

Table 2-31: **10GBASE-KR LP Coefficient Update Register**

Bits	Name	Description	Attributes	Default Value
1.152.15:14	Reserved	The block always returns 0 for this bit and ignores writes.	R/O	0
1.152.13	Preset	1 = Preset coefficients 0 = Normal operation	R/W(1)	0
1.152.12	Initialize	1 = Initialize coefficients 0 = Normal operation	R/W(1)	0
1.152.11:6	Reserved	The block always returns 0 for this bit and ignores writes.	R/O	0s
1.152.5:4	Coefficient (+1) update	5:4 = 11 = reserved 10 = decrement 01 = increment 00 = hold	R/W(1)	00
1.152.3:2	Coefficient (0) update	3:2 = 11 = reserved 10 = decrement 01 = increment 00 = hold	R/W(1)	00
1.152.1:0	Coefficient (–1) update	1:0 = 11 = reserved 10 = decrement 01 = increment 00 = hold	R/W(1)	00

Notes:

1. Writable only when register 1.150.1 = 0.

MDIO Register 1.153: 10GBASE-KR LP Status

Figure 2-16 shows the MDIO register 1.153: 10GBASE-KR LP status.

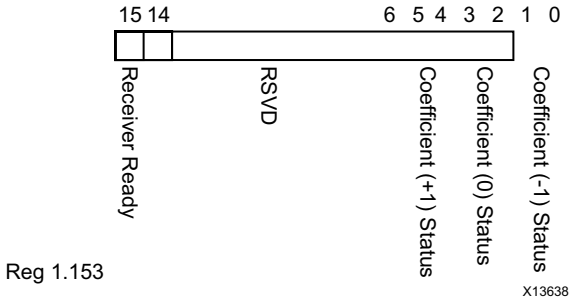

Figure 2-16: **10GBASE-KR LP Status Register**

Table 2-32 shows the 10GBASE-KR LP status register bit definitions.

Table 2-32: **10GBASE-KR LP Status Register**

Bits	Name	Description	Attributes	Default Value
1.153.15	Receiver Ready	1 = The LP receiver has determined that training is complete and is prepared to receive data 0 = The LP receiver is requesting that training continue	R/O	0
1.153.14:6	Reserved	The block always returns 0 for this bit and ignores writes.	R/O	0s
1.153.5:4	Coefficient (+1) status	5:4 = 11 = maximum 10 = minimum 01 = updated 00 = not updated	R/O	00
1.153.3:2	Coefficient (0) status	3:2 = 11 = maximum 10 = minimum 01 = updated 00 = not updated	R/O	00
1.153.1:0	Coefficient (–1) status	1:0 = 11 = maximum 10 = minimum 01 = updated 00 = not updated	R/O	00

MDIO Register 1.154: 10GBASE-KR LD Coefficient Update

Figure 2-17 shows the MDIO register 1.154: 10GBASE-KR LD coefficient update.

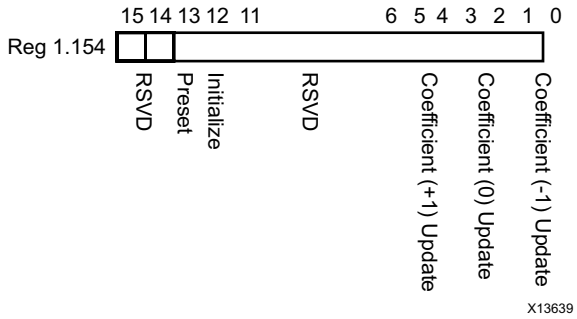

Figure 2-17: **10GBASE-KR LD Coefficient Update Register**

Table 2-33 shows the 10GBASE-KR LD coefficient update register bit definitions.

Table 2-33: **10GBASE-KR LD Coefficient Update Register**

Bits	Name	Description	Attributes	Default Value
1.154.15:14	Reserved	The block always returns 0 for this bit and ignores writes.	R/O	0
1.154.13	Preset	1 = Preset coefficients 0 = Normal operation	R/O(1)	0
1.154.12	Initialize	1 = Initialize coefficients 0 = Normal operation	R/O(1)	0
1.154.11:6	Reserved	The block always returns 0 for this bit and ignores writes.	R/O	0s
1.154.5:4	Coefficient (+1) update	5:4 = 11 = reserved 10 = decrement 01 = increment 00 = hold	R/O(1)	00
1.154.3:2	Coefficient (0) update	3:2 = 11 = reserved 10 = decrement 01 = increment 00 = hold	R/O(1)	00
1.154.1:0	Coefficient (–1) update	1:0 = 11 = reserved 10 = decrement 01 = increment 00 = hold	R/O(1)	00

Notes:

1. These registers are programmed by writing to register 1.65520.

MDIO Register 1.155: 10GBASE-KR LD Status

Figure 2-18 shows the MDIO register 1.155: 10GBASE-KR LD status.

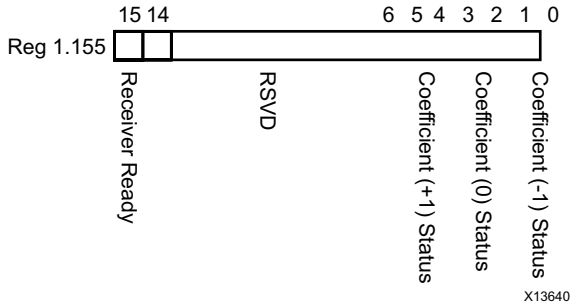

Figure 2-18: **10GBASE-KR LD Status Register**

Table 2-34 shows the 10GBASE-KR LD status register bit definitions.

Table 2-34: **10GBASE-KR LD Status Register**

Bits	Name	Description	Attributes	Default Value
1.155.15	Receiver Ready	1 = The LD receiver has determined that training is complete and is prepared to receive data. 0 = The LD receiver is requesting that training continue.	R/O	0
1.155.14:6	Reserved	The block always returns 0 for this bit and ignores writes.	R/O	0s
1.155.5:4	Coefficient (+1) status	5:4 = 11 = maximum 10 = minimum 01 = updated 00 = not updated	R/O	00
1.155.3:2	Coefficient (0) status	3:2 = 11 = maximum 10 = minimum 01 = updated 00 = not updated	R/O	00
1.155.1:0	Coefficient (–1) status	1:0 = 11 = maximum 10 = minimum 01 = updated 00 = not updated	R/O	00

MDIO Register 1.170: 10GBASE-R FEC Ability

Figure 2-19 shows the MDIO register 1.170: 10GBASE-R FEC Ability.

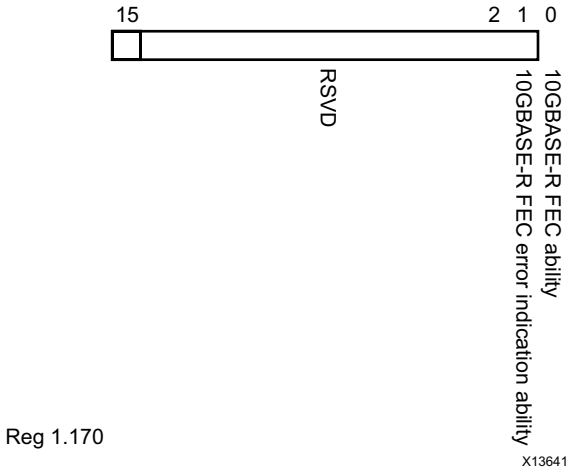

Figure 2-19: **10GBASE-R FEC Ability Register**

Table 2-35 shows the 10GBASE-R FEC Ability register bit definitions.

Table 2-35: **10GBASE-R FEC Ability Register**

Bits	Name	Description	Attributes	Default Value
1.170.15:2	Reserved	The block always returns 0 for this bit and ignores writes.	R/O	0s
1.170.1	10GBASE-R FEC error indication ability	1 = the PHY is able to report FEC decoding errors to the PCS layer	R/O	1
1.170.0	10GBASE-R FEC ability	1 = the PHY supports FEC	R/O	1

MDIO Register 1.171: 10GBASE-R FEC Control

Figure 2-20 shows the MDIO register 1.170: 10GBASE-R FEC Control.

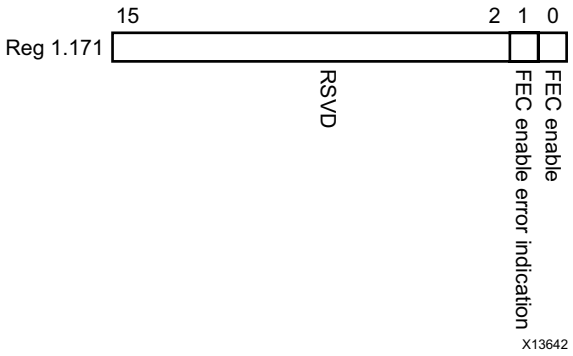

Figure 2-20: **10GBASE-R FEC Control Register**

Table 2-36 shows the 10GBASE-R FEC Control register bit definitions.

Table 2-36: **10GBASE-R FEC Control Register**

Bits	Name	Description	Attributes	Default Value
1.171.15:2	Reserved	The block always returns 0 for this bit and ignores writes.	R/O	0s
1.171.1	10GBASE-R FEC error indication ability(1)	1 = Configure the PHY to report FEC decoding errors to the PCS layer.	R/W	0
1.171.0	10GBASE-R FEC ability(2)	1 = enable FEC 0 = disable FEC	R/W	0

Notes:

1. If FEC Error Passing is enabled while FEC is enabled, errors will be seen temporarily. To avoid this, only enable Error Passing while FEC is disabled.
2. If FEC is enabled during auto-negotiation then this register bit is overridden by the auto-negotiation control of FEC. So even with this bit set to 0, if auto-negotiation results in FEC being enabled, FEC is enabled and cannot be disabled except by changing the auto-negotiation Base Page Ability bits 46 and 47, and re-negotiating the link. FEC can still be enabled explicitly by setting this bit to 1 which overrides whatever auto-negotiation decides.

MDIO Register 1.172: 10GBASE-R FEC Corrected Blocks (Lower)

Figure 2-21 shows the MDIO register 1.172: 10GBASE-R FEC Corrected Blocks (lower).

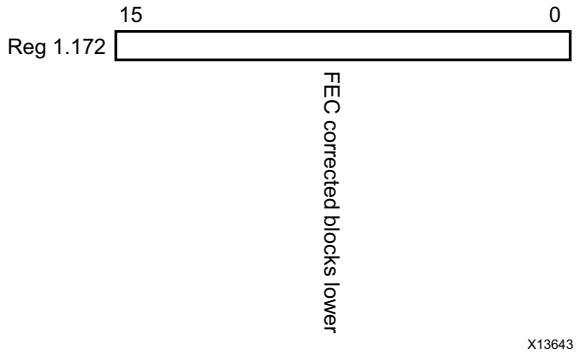

Figure 2-21: **10GBASE-R FEC Corrected Blocks (Lower) Register**

Table 2-37 shows the 10GBASE-R FEC Corrected Blocks (lower) register bit definitions.

Table 2-37: **10GBASE-R FEC Corrected Blocks (Lower) Register**

Bits	Name	Description	Attributes	Default Value
1.172.15:0	FEC Corrected blocks	Bits 15:0 of the Corrected Blocks count	R/O(1)	0s

Notes:
1. Cleared when read.

MDIO Register 1.173: 10GBASE-R FEC Corrected Blocks (Upper)

Figure 2-22 shows the MDIO register 1.173: 10GBASE-R FEC Corrected Blocks (upper).

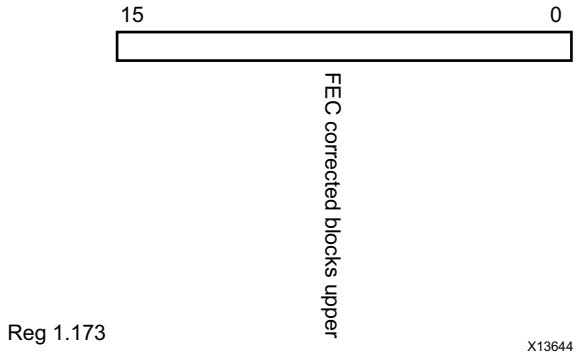

Figure 2-22: **10GBASE-R FEC Corrected Blocks (Upper) Register**

Table 2-38 shows the 10GBASE-R FEC Corrected Blocks (upper) register bit definitions.

Table 2-38: **10GBASE-R FEC Corrected Blocks (Upper) Register**

Bits	Name	Description	Attributes	Default Value
1.173.15:0	FEC Corrected blocks	Bits 31:16 of the Corrected Blocks count	R/O(1)	0s

Notes:
1. Latched when 1.172 is read. Cleared when read.

MDIO Register 1.174: 10GBASE-R FEC Uncorrected Blocks (Lower)

Figure 2-23 shows the MDIO register 1.174: 10GBASE-R FEC Uncorrected Blocks (lower).

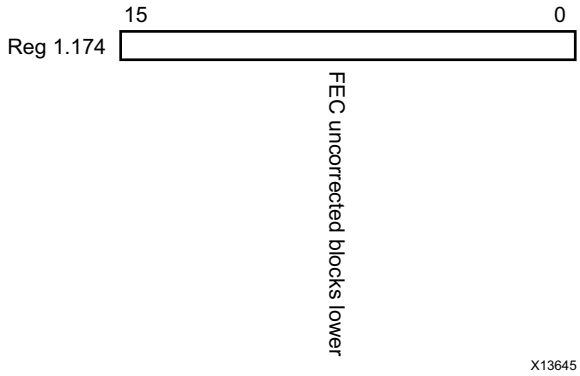

Figure 2-23: **10GBASE-R FEC Uncorrected Blocks (Lower) Register**

Table 2-39 shows the 10GBASE-R FEC Uncorrected Blocks (lower) register bit definitions.

Table 2-39: **10GBASE-R FEC Uncorrected Blocks (Lower) Register**

Bits	Name	Description	Attributes	Default Value
1.174.15:0	FEC Uncorrected blocks	Bits 15:0 of the Uncorrected Blocks count	R/O(1)	0s

Notes:
1. Cleared when read.

MDIO Register 1.175: 10GBASE-R FEC Uncorrected Blocks (Upper)

Figure 2-24 shows the MDIO register 1.175: 10GBASE-R FEC Uncorrected Blocks (upper).

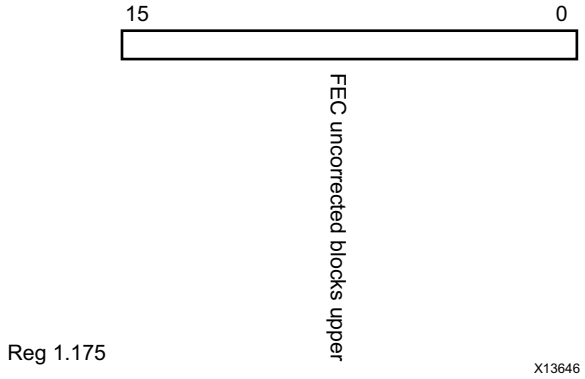

Figure 2-24: **10GBASE-R FEC Uncorrected Blocks (Upper) Register**

Table 2-40 shows the 10GBASE-R FEC Uncorrected Blocks (upper) register bit definitions.

Table 2-40: **10GBASE-R FEC Uncorrected Blocks (Upper) Register**

Bits	Name	Description	Attributes	Default Value
1.175.15:0	FEC Uncorrected blocks	Bits 31:16 of the Uncorrected Blocks count	R/O(1)	0s

Notes:

1. Latched when 1.174 is read. Cleared when read.

MDIO Register: 1.65520: Vendor-Specific LD Training

Figure 2-25 shows the MDIO register 1.65520: Vendor-specific LD Training.

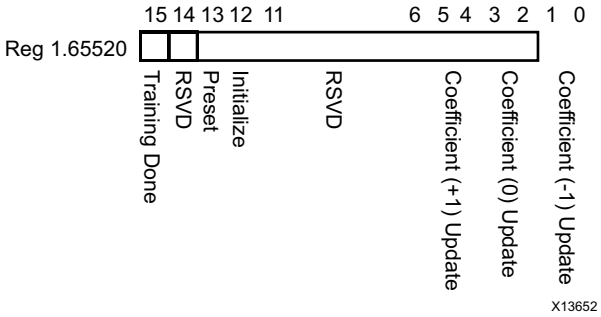

Figure 2-25: **Vendor-specific LD Training Register**

Table 2-41 shows the Vendor-specific LD Training register bit definitions.

Table 2-41: **Vendor-Specific LD Training Register**

Bits	Name	Description	Attributes	Default Value
1.65520.15	Training Done	1 = Training Algorithm has determined that the LP transmitter has been successfully trained.	R/W(1)	0
1.65520.14	Reserved	The block always returns 0 for this bit and ignores writes.	R/O	0
1.65520.13	Preset	1 = Preset coefficients 0 = Normal operation	R/O(2)	0
1.65520.12	Initialize	1 = Initialize coefficients 0 = Normal operation	R/O(2)	0
1.65520.5:4	Coefficient (+1) update	5:4 = 11 = reserved 10 = decrement 01 = increment 00 = hold	R/O(2)	00
1.65520.3:2	Coefficient (0) update	3:2 = 11 = reserved 10 = decrement 01 = increment 00 = hold	R/O(2)	00
1.65520.1:0	Coefficient (–1) update	1:0 = 11 = reserved 10 = decrement 01 = increment 00 = hold	R/O(2)	00

Notes:
1. This register will be transferred automatically to register 1.155.15.
2. These registers will be transferred automatically to register 1.154.

MDIO Register 1.65535: Core Version Info

Figure 2-26 shows the MDIO 1.65535 register: Core Version Info.

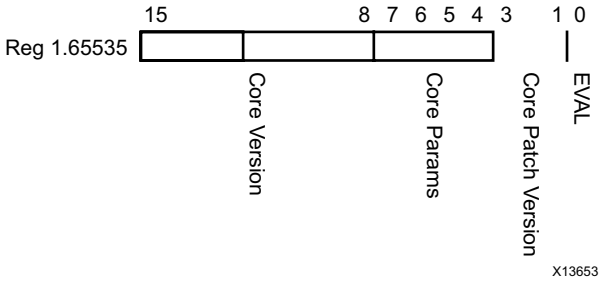

Figure 2-26: **Core Version Info Register**

Table 2-42 shows the core version information register bit definitions.

Table 2-42: **Core Version Information**

Bits	Name	Description	Attributes	Default Value
1.65535.15:8	Core Version	Bits 15..12 give the major core version and bits 11..8 give the minor core version.	R/O	(1)
1.65535.7:4	Core parameters	Bit 7 = 1 = KR included Bit 6 – reserved Bit 5 = 1 = AN included Bit 4 = 1 = FEC included	R/O	(2)
1.65535.3:1	Core Patch Version	Bits 3..1 give the patch (revision) number, if any, for the core.	R/O	000
1.65535.0	BASE-KR only: EVAL	1 = This core was generated using a Hardware Evaluation license	R/O	0

Notes:
1. x'60' for version 6.0 of core.
2. Depends on core generation parameters.

MDIO Register 3.0: PCS Control 1

Figure 2-27 shows the MDIO register 3.0: PCS Control 1.

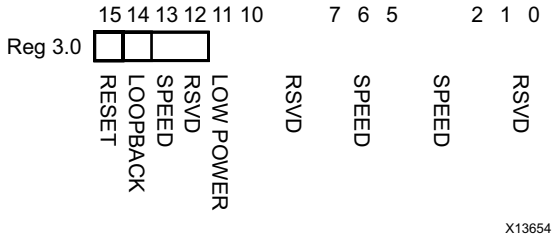

Figure 2-27: **PCS Control 1 Register**

Table 2-43 shows the PCS Control 1 register bit definitions.

Table 2-43: **PCS Control 1 Register**

Bits	Name	Description	Attributes	Default Value
3.0.15	Reset	1 = Block reset 0 = Normal operation The 10GBASE-R/KR block is reset when this bit is set to 1. It returns to 0 when the reset is complete. Note: After writing to this bit, wait for at least nine cycles of the management clock (mdc) at 2.5 MHz before starting MDIO transactions.	R/W Self-clearing	0
3.0.14	10GBASE-R/KR Loopback	1 = Use PCS Loopback (Near-end) 0 = Do not use PCS Loopback	R/W	0
3.0.13	Speed Selection	The block always returns 1 for this bit. 1 (and bit 6 = 1) = bits 5:2 select the speed	R/O	1
3.0.12	Reserved	The block always returns 0 for this bit and ignores writes.	R/O	0
3.0.11	Power down	This bit has no effect.	R/W	0
3.0.10:7	Reserved	The block always returns 0 for these bits and ignores writes.	R/O	All 0s
3.0.6	Speed Selection	The block always returns 1 for this bit.	R/O	1
3.0.5:2	Speed Selection	The block always returns 0000 = 10Gb/s	R/O	All 0s
3.0.1:0	Reserved	The block always returns 0 for this bit and ignores writes.	R/O	All 0s

MDIO Register 3.1: PCS Status 1

Figure 2-28 shows the MDIO register 3.1: PCS Status 1.

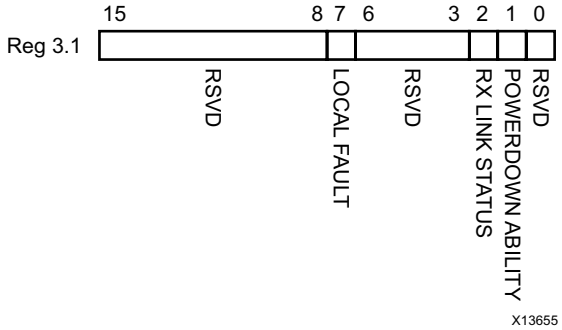

Figure 2-28: **PCS Status 1 Register**

Table 2-44 show the PCS 1 register bit definitions.

Table 2-44: **PCS Status 1 Register Bit Definition**

Bits	Name	Description	Attributes	Default Value
3.1.15:8	Reserved	The block always returns 0s for these bits and ignores writes.	R/O	All 0s
3.1.7	Local Fault	1 = Local Fault detected	R/O	0
3.1.6:3	Reserved	The block always returns 0s for these bits and ignores writes.	R/O	All 0s
3.1.2	PCS Receive Link Status	1 = The PCS receive link is up 0 = The PCS receive link is down This is a latching Low version of bit 3.32.12.	R/O Self-setting	-
3.1.1	Power Down Ability	The block always returns 1 for this bit.	R/O	1
3.1.0	Reserved	The block always returns 0 for this bit and ignores writes.	R/O	0

MDIO Register 3.4: PCS Speed Ability

Figure 2-29 shows the MDIO register 3.4: PCS Speed Ability.

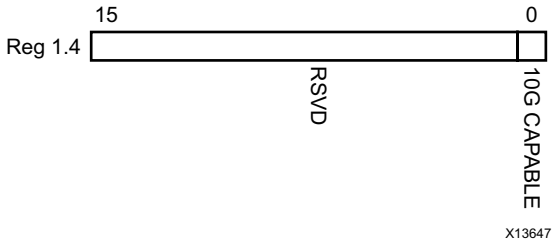

Figure 2-29: **PCS Speed Ability Register**

Table 2-45 shows the PCS Speed Ability register bit definitions.

Table 2-45: **PCS Speed Ability Register**

Bits	Name	Description	Attributes	Default Value
3.4.15:1	Reserved	The block always returns 0 for these bits and ignores writes.	R/O	All 0s
3.4.0	10G Capable	The block always returns 1 for this bit and ignores writes.	R/O	1

MDIO Registers 3.5 and 3.6: PCS Devices in Package

Figure 2-30 shows the MDIO registers 3.5 and 3.6: PCS Devices in Package.

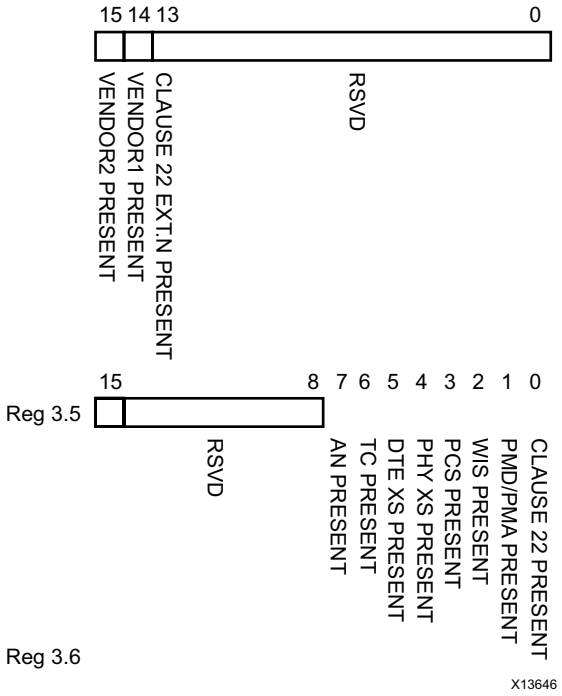

Figure 2-30: **PCS Devices in Package Registers**

Table 2-46 shows the PCS Devices in Package registers bit definitions.

Table 2-46: **PCS Devices in Package Registers**

Bits	Name	Description	Attributes	Default Value
3.6.15	Vendor-specific Device 2 Present	The block always returns 0 for this bit.	R/O	0
3.6.14	Vendor- specific Device 1 Present	The block always returns 0 for this bit.	R/O	0
3.6.13	Clause 22 extension present	The block always returns 0 for this bit.	1/0	1
3.6.12:0	Reserved	The block always returns 0 for these bits.	R/O	All 0s
3.5.15:8	Reserved	The block always returns 0 for these bits.	R/O	All 0s
3.5.7	Auto Negotiation Present	1 = AN Block included	1/0	1
3.5.6	TC present	The block always returns 0 for this bit.	R/O	0
3.5.5	PHY XS Present	The block always returns 0 for this bit.	R/O	0
3.5.4	PHY XS Present	The block always returns 0 for this bit.	R/O	0
3.5.3	PCS Present	The block always returns 1 for this bit.	R/O	1

Table 2-46: **PCS Devices in Package Registers *(Cont'd)***

Bits	Name	Description	Attributes	Default Value
3.5.2	WIS Present	The block always returns 0 for this bit.	R/O	0
3.5.1	PMA/PMD Present	The block always returns 1 for this bit.	R/O	1
3.5.0	Clause 22 device present	The block always returns 0 for this bit.	R/O	0

MDIO Register 3.7: 10G PCS Control 2

Figure 2-31 shows the MDIO register 3.7: 10G PCS Control 2.

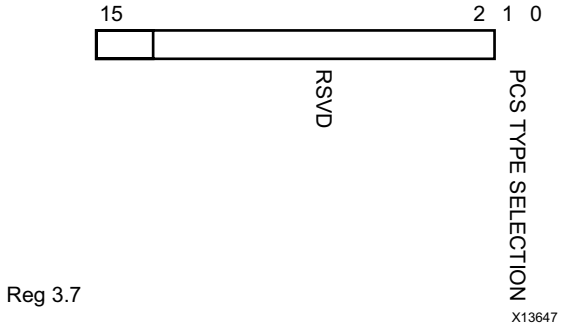

Figure 2-31: **10G PCS Control 2 Register**

Table 2-47 shows the 10 G PCS Control 2 register bit definitions.

Table 2-47: **10G PCS Control 2 Register**

Bits	Name	Description	Attributes	Default Value
3.7.15:2	Reserved	The block always returns 0 for these bits and ignores writes.	R/O	All 0s
3.7.1:0	PCS Type Selection	00 = Select 10GBASE-R PCS type. Any other value written to this register is ignored.	R/W	00

MDIO Register 3.8: 10G PCS Status 2

Figure 2-32 shows the MDIO register 3.8: 10G PCS Status 2.

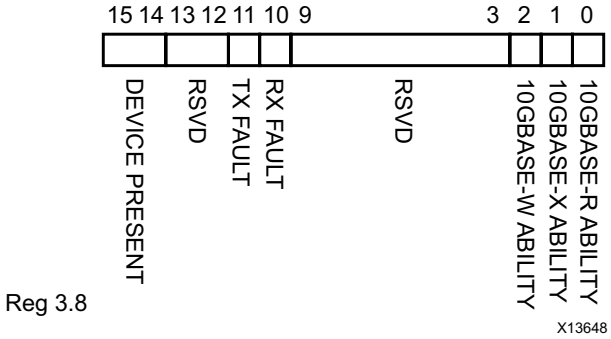

Figure 2-32: **10G PCS Status 2 Register**

Table 2-48 shows the 10G PCS Status 2 register bit definitions.

Table 2-48: **10G PCS Status 2 Register**

Bits	Name	Description	Attributes	Default Value
3.8.15:14	Device present	The block always returns 10.	R/O	10
3.8.13:12	Reserved	The block always returns 0 for these bits.	R/O	All 0s
3.8.11	Transmit local fault	1 = Transmit Fault detected	R/O	0
3.8.10	Receive local fault	1 = Receive Fault detected	R/O	0
3.8.9:3	Reserved	The block always returns 0 for these bits.	R/O	All 0s
3.8.2	10GBASE-W Capable	The block always returns 0 for this bit.	R/O	0
3.8.1	10GBASE-X Capable	The block always returns 0 for this bit.	R/O	0
3.8.0	10GBASE-R Capable	The block always returns 1 for this bit.	R/O	1

MDIO Register 3.32: 10GBASE-R Status 1

Figure 2-33 shows the MDIO register 3.32: 10GBASE-R Status 1.

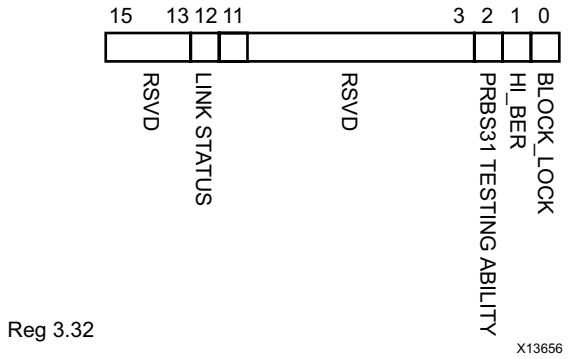

Figure 2-33: **10GBASE-R Status Register 1**

Table 2-49 shows the 10GBASE-R Status register bit definitions.

Table 2-49: **10GBASE-R Status Register 1**

Bits	Name	Description	Attributes	Default Value
3.32.15:13	Reserved	The block always returns 0 for these bits.	R/O	All 0s
3.32.12	10GBASE-R Link Status	1 = 10GBASE-R receive is aligned 0 = 10GBASE-R receive is not aligned.	RO	0
3.32.11:3	Reserved	The block always returns 0 for these bits.	R/O	0s
3.32.2	PRBS31 Pattern Testing Ability	The block always returns 1 for this bit.	R/O	1
3.32.1	Hi BER	1 = RX showing hi-ber 0 = RX not showing hi ber	R/O	0
3.32.0	Block Lock	1 = RX is synchronized 0 = RX is not synchronized.	R/O	0

MDIO Register 3.33: 10GBASE-R Status 2

Figure 2-34 shows the MDIO register 3.33: 10GBASE-R Status 2.

Figure 2-34: **10GBASE-R Status Register 2**

Table 2-50 shows the 10GBASE-R Status register bit definition. All bits are cleared when read.

Table 2-50: **10GBASE-R Status Register 2**

Bits	Name	Description	Attributes	Default Value
3.33.15	Latched Block Lock	Latch-Low version of block lock	R/O	0
3.33.14	Latched HiBER	Latch-High version of Hi BER	R/O	1
3.33.13:8	BER	BER Counter	R/O	0s
3.33.7:0	Errored Blocks Count	Counter for Errored Blocks	R/O	0s

MDIO Register 3.34–37: 10GBASE-R Test Pattern Seed A0–3

Figure 2-35 shows the MDIO register 3.34–37 10GBASE-R Test Pattern Seed A.

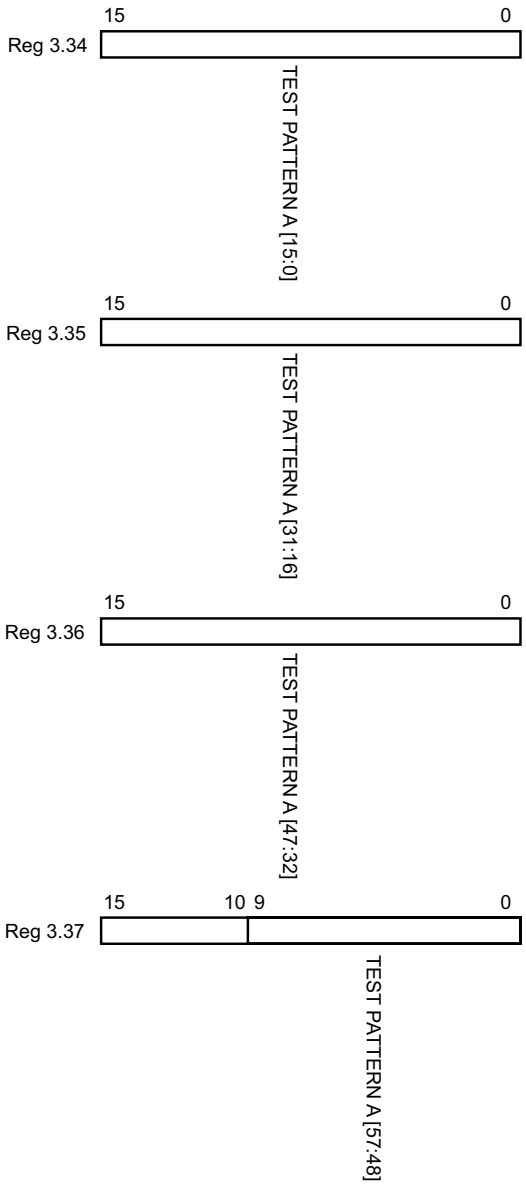

Figure 2-35: **10GBASE-R Test Pattern Seed A0-3 Registers**

Table 2-51 shows the 10GBASE-R Test Pattern Seed A0–2 register bit definitions.

Table 2-51: **10GBASE-R Test Pattern Seed A0-2 Register**

Bits	Name	Description	Attributes	Default Value
3.34–36.15:0 3.37.9:0	Seed A bits 15:0, 31:16, 47:32, 57:48 resp	Seed for Pseudo-Random Test Pattern	R/W	all 0s

MDIO Register 3.38–41: 10GBASE-R Test Pattern Seed B0–3

Figure 2-36 shows the MDIO register 3.38–41: 10GBASE-R Test Pattern Seed B.

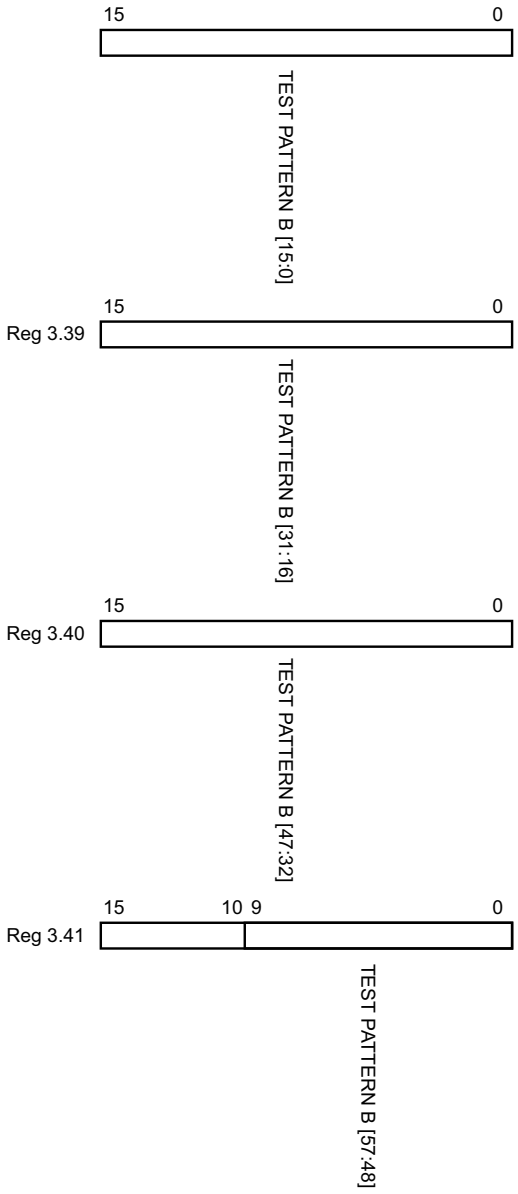

Figure 2-36: **10GBASE-R Test Pattern Seed B0-3 Registers**

Table 2-52 shows the 10GBASE-R Test Pattern Seed B0–3 register bit definitions.

Table 2-52: **10GBASE-R Test Pattern Seed B0-3 Register**

Bits	Name	Description	Attributes	Default Value
3.38–40.15:0 3.41.9:0	Seed B bits 15:0, 31:16, 47:32, 57:48 resp	Seed for Pseudo-Random Test Pattern	R/W	all 0s

MDIO Register 3.42: 10GBASE-R Test Pattern Control

Figure 2-37 shows the MDIO register 3.42: 10GBASE-R Test Pattern Control.

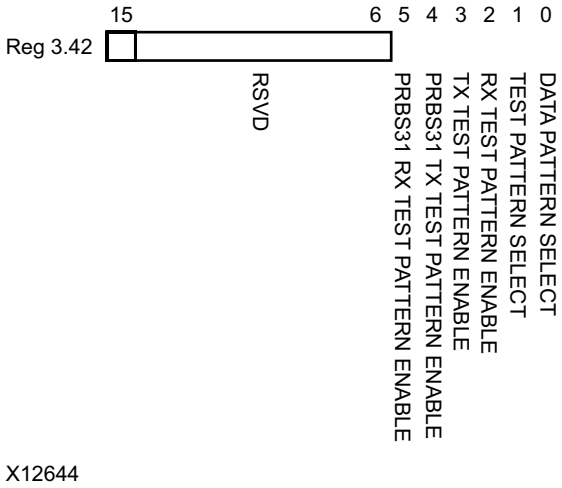

Figure 2-37: **10GBASE-R Test Pattern Control Register**

Table 2-53 shows the 10GBASE-R Test Pattern Control register bit definitions.

Table 2-53: **10GBASE-R Test Pattern Control Register**

Bits	Name	Description	Attributes	Default Value
3.42.15:6	Reserved	The block always returns 0s for these bits.	R/O	All 0s
3.42.5	PRBS31 RX test pattern enable	1 = Enable PRBS RX tests 0 = Disable PRBS RX tests	R/W	0
3.42.4	PRBS31 TX test pattern enable	1 = Enable PRBS TX tests 0 = Disable PRBS TX tests	R/W	0
3.42.3	TX test pattern enable	Enables the TX Test Pattern which has been selected with bits [1:0].	R/W	0
3.42.2	RX test pattern enable	Enables the RX Test Pattern Checking which has been selected with bits [1:0]	R/W	0
3.42.1	Test pattern select	1 = Square wave 0 = Pseudo-Random	R/W	0
3.42.0	Data pattern select	1 = Zeros pattern 0 = LF Data pattern	R/W	0

Notes:

1. PRBS31 test pattern generation and checking is implemented in the transceiver and the error count is read by the 10GBASE-R/KR core through the transceiver DRP interface. All other test pattern generation and checking where applicable is implemented in the PCS logic in the core.

MDIO Register 3.43: 10GBASE-R Test Pattern Error Counter

Due to the special implementation of this register for cores with the MDIO interface, when the MDIO PCS Address is set to point to this register and PRBS31 RX error checking is

enabled in register 3.42.5, no other MDIO commands are accepted until a different PCS address is selected with an MDIO ADDRESS command.

For 7 series devices, the number of errors is equal to the number of 20-bit words received that included errors, rather than the actual number of bit errors.

For UltraScale devices, the number of errors is equal to the number of single bit errors received where there are fewer than 64K bit errors. UltraScale device transceivers use a 32-bit counter for this feature, but it is only possible to read back 16 bits from the error counter register in the core. The lower 16 bits from the transceiver counter register are used as the value for the core register. Each read operation clears the transceiver counter register so, as long as there are fewer than 64K bit errors between each successive read, the values returned are valid.

Figure 2-38 shows the MDIO register 3.43: 10GBASE-R Test Pattern Error Counter.

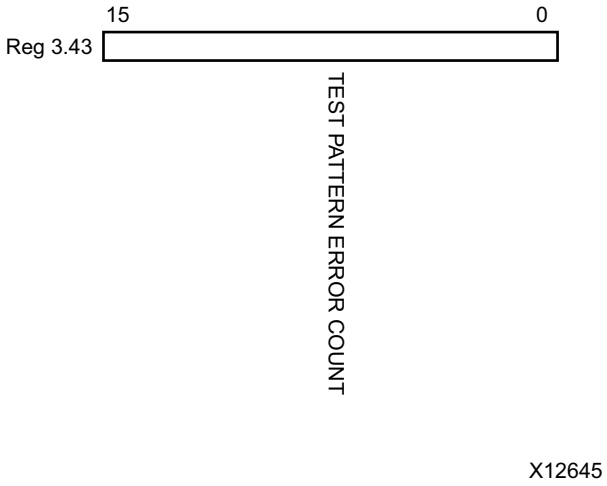

Figure 2-38: **10GBASE-R Test Pattern Error Counter Register**

Table 2-54 shows the 10GBASE-R Test Pattern Error Counter register bit definitions. This register is cleared when read.

Table 2-54: **10GBASE-R Test Pattern Error Counter Register**

Bits	Name	Description	Attributes	Default Value
3.43.15:0	Test pattern error counter	Count of errors	R/O	All 0s

MDIO Register 3.65535: 125 Microsecond Timer Control

Figure 2-39 shows the MDIO 3.65535 register: 125 microsecond timer control.

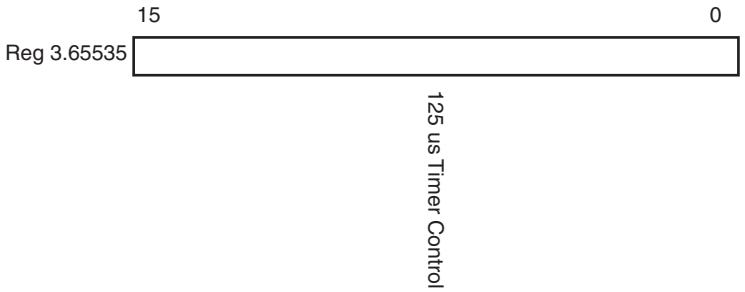

Figure 2-39: **125 Microsecond Timer Control Register**

Table 2-55: **125 µs Timer Control**

Bits	Name	Description	Attributes	Default Value
3.65535.15:0	125 µs timer control	Bits 15..0 set the number of clock cycles at 156.25 MHz to be used to measure the 125 µs timer in the BER monitor state machine. Useful for debug purposes (simulation speedup).	R/W	x'4C4B'

MDIO Register 7.0: AN Control

Figure 2-40 shows the MDIO register 7.0: AN Control.

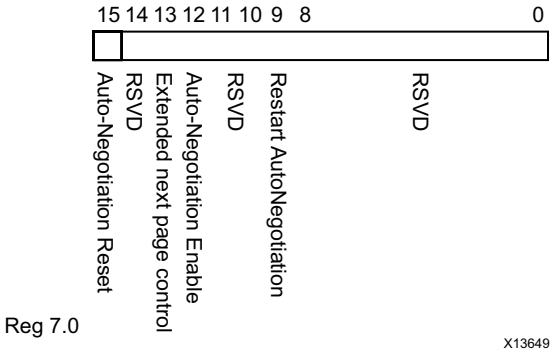

Figure 2-40: **AN Control Register**

Table 2-56 shows the AN Control register bit definitions.

Table 2-56: **AN Control Register**

Bits	Name	Description	Attributes	Default Value
7.0.15	AN Reset	1 = AN Reset 0 = AN normal operation	R/W Self-clearing	0
7.0.14	Reserved	The block always returns 0 for this bit and ignores writes.	R/O	0
7.0.13	Extended Next Page control	1 = Extended Next Pages are supported 0 = Not supported	R/W	0
7.0.12	AN Enable	1 = Enable AN Process 0 = Disable	R/W(1)	1
7.0.11:10	Reserved	The block always returns 0 for this bit and ignores writes.	R/O	00
7.0.9	Restart AN	1 = Restart AN process 0 = Normal operation	R/W Self-clearing	0
7.0.8:0	Reserved	The block always returns 0 for this bit and ignores writes.	R/O	0s

Notes:

1. For simulation purposes only, to disable AN at start-up, the external core pin 'an_enable' should be tied Low.

MDIO Register 7.1: AN Status

Figure 2-41 shows the MDIO register 7.1: AN Status.

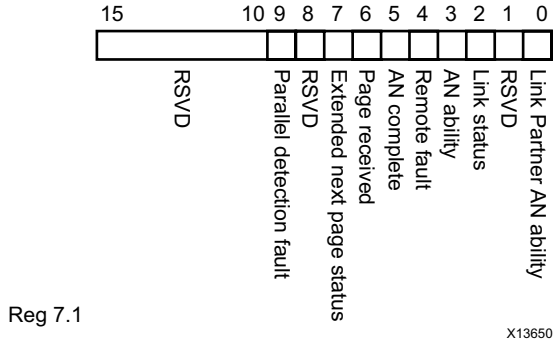

Figure 2-41: **AN Status Register**

Table 2-57 shows the AN Status register bit definitions.

Table 2-57: **AN Status Register**

Bits	Name	Description	Attributes	Default Value
7.1.15:10	Reserved	The block always returns 0 for this bit and ignores writes.	R/O	0s
7.1.9	Parallel Detection Fault	1 = A fault has been detected through the parallel detection function 0 = no fault detected	R/O Latches High	0
7.1.8	Reserved	The block always returns 0 for this bit and ignores writes.	R/O	0
7.1.7	Extended Next Page status	1 = XNP format is used 0 = XNP format is not allowed	R/O	0
7.1.6	Page Received	1 = A page has been received 0 = No page received	R/O Latches High	0
7.1.5	AN Complete	1 = AN process has completed 0 = not completed	R/O	0
7.1.4	Remote fault	1 = remote fault condition detected 0 = not detected	R/O Latches High	0
7.1.3	AN ability	1 = PHY supports auto-negotiation 0 = PHY does not support auto-negotiation	R/O	1
7.1.2	Link status	1 = Link is up 0 = Link is down	R/O Latches Low	0
7.1.1	Reserved	The block always returns 0 for this bit and ignores writes.	R/O	0
7.1.0	LP AN ability	1 = LP is able to perform AN 0 = not able	R/O	0

MDIO Register 7.16:17:18: AN Advertisement

Figure 2-42 shows the MDIO register 7.16: AN Advertisement.

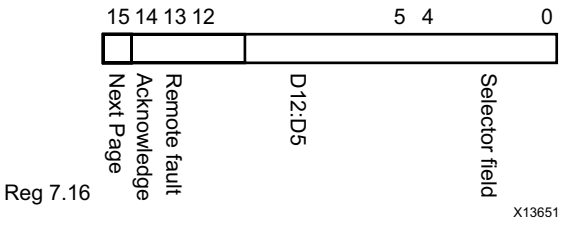

Figure 2-42: **AN Advertisement Register 0**

Table 2-58 shows the AN Advertisement register bit definitions.

Table 2-58: **AN Advertisement Register 0**

Bits	Name	Description	Attributes	Default Value
7.16.15	Next Page	Consult IEEE802.3	R/W	0
7.16.14	Acknowledge	The block always returns 0 for this bit and ignores writes.	R/O	0
7.16.13	Remote Fault	Consult IEEE802.3	R/W	0
7.16.12:5	D12:D5	Consult IEEE802.3	R/W	0s
7.16.4:0	Selector Field	Consult IEEE802.3	R/W	00001

Figure 2-43 shows the MDIO register 7.17: AN Advertisement.

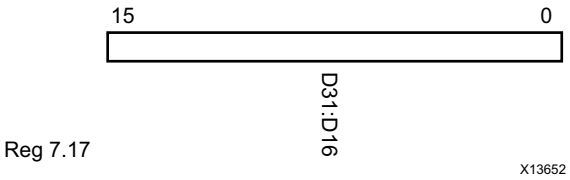

Figure 2-43: **AN Advertisement Register 1**

Table 2-59 shows the AN Advertisement register bit definitions.

Table 2-59: **AN Advertisement Register 1**

Bits	Name	Description	Attributes	Default Value
7.17.15:0	D31:D16	Consult IEEE802.3	R/W	0

Figure 2-44 shows the MDIO register 7.18: AN Advertisement.

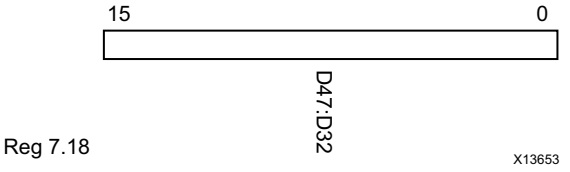

Figure 2-44: **AN Advertisement Register 2**

Table 2-60 shows the AN Advertisement register bit definitions.

Table 2-60: **AN Advertisement Register 2**

Bits	Name	Description	Attributes	Default Value
7.18.15:0	D47:D32	Consult IEEE802.3	R/W	0

MDIO Register 7.19, 20, 21: AN LP Base Page Ability

Figure 2-45 shows the MDIO register 7.19: AN LP Base Page Ability.

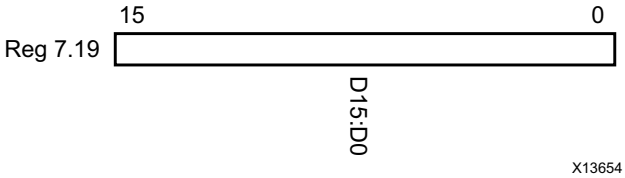

Figure 2-45: **AN LP Base Page Ability Register 0**

Table 2-61 shows the AN LP Base Page Ability register bit definitions.

Table 2-61: **AN LP Base Page Ability Register 0**

Bits	Name	Description	Attributes	Default Value
7.19.15:0	D15:D0	Consult IEEE802.3	R/O	0

Figure 2-46 shows the MDIO register 7.20: AN LP Base Page Ability.

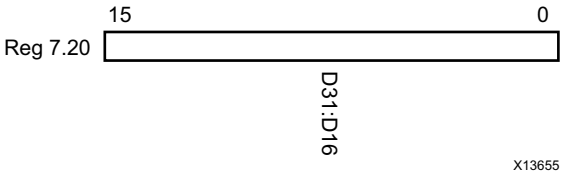

Figure 2-46: **AN LP Base Page Ability Register 1**

Table 2-62 shows the AN LP Base Page Ability register bit definitions.

Table 2-62: **AN LP Base Page Ability Register 1**

Bits	Name	Description	Attributes	Default Value
7.20.15:0	D31:D16	Consult IEEE802.3	R/W	0

Figure 2-47 shows the MDIO register 7.21: AN LP Base Page Ability.

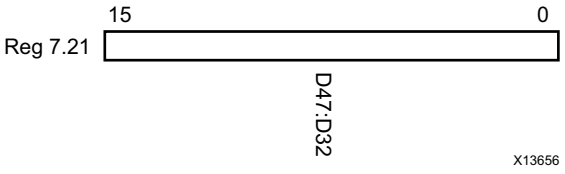

Figure 2-47: **AN LP Base Page Ability Register 2**

Table 2-63 shows the AN LP Base Page Ability register bit definitions.

Table 2-63: **AN LP Base Page Ability Register 2**

Bits	Name	Description	Attributes	Default Value
7.21.15:0	D47:D32	Consult IEEE802.3	R/W	0

MDIO Register 7.22, 23, 24: AN XNP Transmit

Figure 2-48 shows the MDIO register 7.22: AN XNP Transmit.

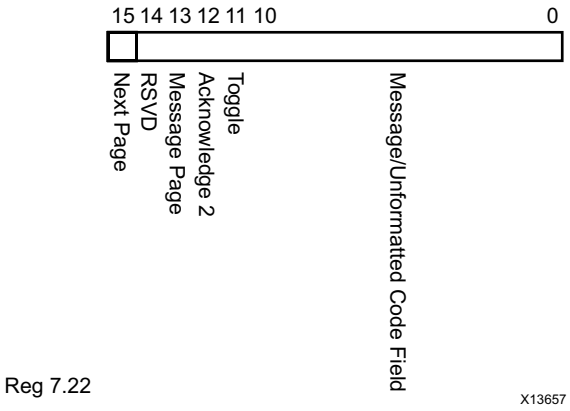

Figure 2-48: **AN XNP Transmit Register 0**

Table 2-64 shows the AN XNP Transmit register bit definitions.

Table 2-64: **AN XNP Transmit Register 0**

Bits	Name	Description	Attributes	Default Value
7.22.15	Next Page	Consult IEEE802.3	R/W	0
7.22.14	Reserved	The block always returns 0 for this bit and ignores writes.	R/O	0
7.22.13	Message Page	Consult IEEE802.3	R/W	0
7.22.12	Acknowledge 2	Consult IEEE802.3	R/W	0
7.22.11	Toggle	Consult IEEE802.3	R/O	0
7.22.10:0	Message/ Unformatted Code Field	Consult IEEE802.3	R/W	0s

Figure 2-49 shows the MDIO register 7.23: AN XNP Transmit.

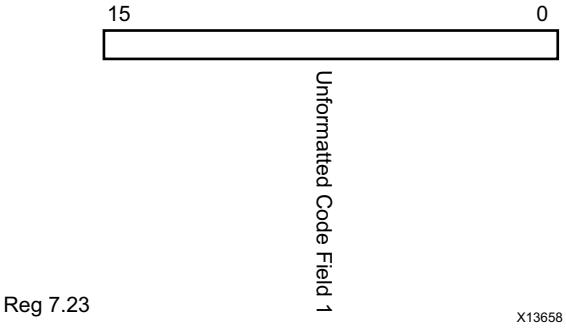

Figure 2-49: **AN XNP Transmit Register 1**

Table 2-65 shows the AN XNP Transmit register bit definitions.

Table 2-65: **AN XNP Transmit Register 1**

Bits	Name	Description	Attributes	Default Value
7.23.15:0	Unformatted Code Field 1	Consult IEEE802.3	R/W	0s

Figure 2-50 shows the MDIO register 7.24: AN XNP Transmit.

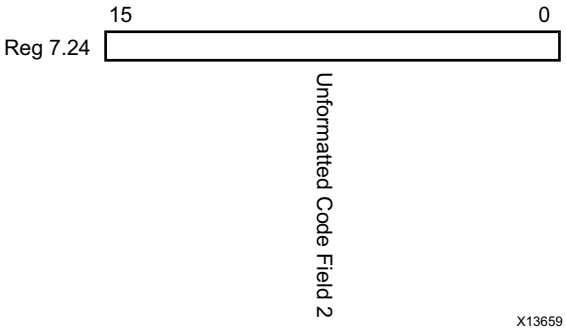

Figure 2-50: **AN XNP Transmit Register 2**

Table 2-66 shows the AN XNP Transmit register bit definitions.

Table 2-66: **AN XNP Transmit Register 2**

Bits	Name	Description	Attributes	Default Value
7.24.15:0	Unformatted Code Field 2	Consult IEEE802.3	R/W	0s

MDIO Register 7.25, 26, 27: AN LP XNP Ability

Figure 2-51 shows the MDIO register 7.25: AN LP XNP Ability.

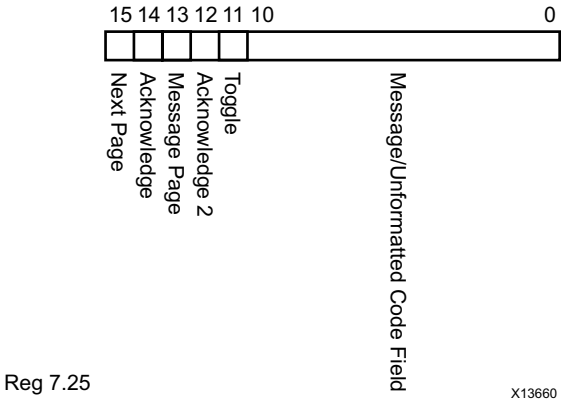

Figure 2-51: **AN LP XNP Ability Register 0**

Table 2-67 shows the AN LP XNP Ability register bit definitions.

Table 2-67: **AN LP XNP Ability Register 0**

Bits	Name	Description	Attributes	Default Value
7.25.15	Next Page	Consult IEEE802.3	R/O	0
7.25.14	Acknowledge	Consult IEEE802.3	R/O	0
7.25.13	Message Page	Consult IEEE802.3	R/O	0
7.25.12	Acknowledge 2	Consult IEEE802.3	R/O	0
7.25.11	Toggle	Consult IEEE802.3	R/O	0
7.25.10:0	Message/ Unformatted Code Field	Consult IEEE802.3	R/O	0s

Figure 2-52 shows the MDIO register 7.26: AN LP XNP Ability.

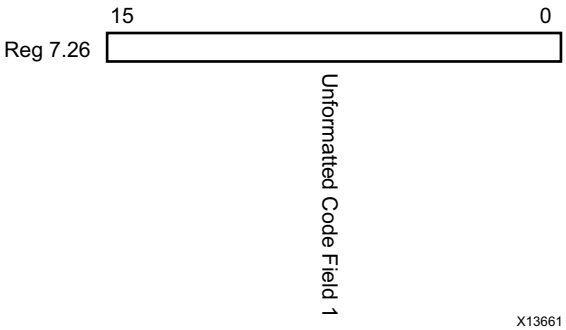

Figure 2-52: **AN LP XNP Ability Register 1**

Table 2-68 shows the AN LP XNP Ability register bit definitions.

Table 2-68: **AN LP XNP Ability Register 1**

Bits	Name	Description	Attributes	Default Value
7.26.15:0	Unformatted Code Field 1	Consult IEEE802.3	R/O	0s

Figure 2-53 shows the MDIO register 7.27: AN LP XNP Ability.

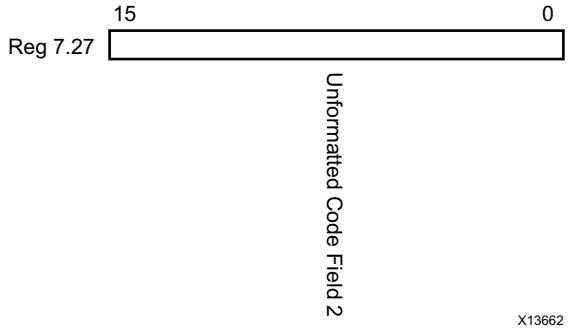

Figure 2-53: **AN LP XNP Ability Register 2**

Table 2-69 shows the AN LP XNP Ability register bit definitions.

Table 2-69: **AN LP XNP Ability Register 2**

Bits	Name	Description	Attributes	Default Value
7.27.15:0	Unformatted Code Field 2	Consult IEEE802.3	R/O	0s

MDIO Register 7.48: Backplane Ethernet Status

Figure 2-54 shows the MDIO register 7.48: Backplane Ethernet Status.

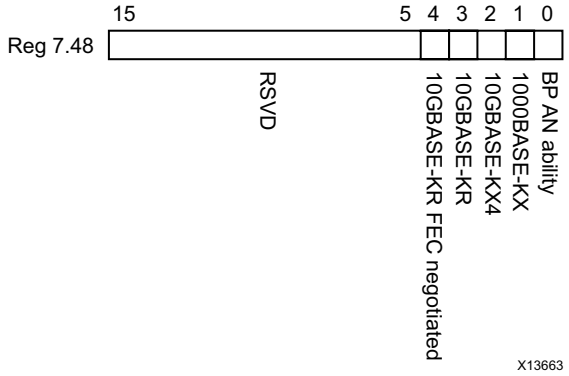

Figure 2-54: **Backplane Ethernet Status Register**

Table 2-70 shows the Backplane Ethernet Status register bit definitions.

Table 2-70: **Backplane Ethernet Status Register**

Bits	Name	Description	Attributes	Default Value
7.48.15:5	Reserved	The block always returns 0 for this bit and ignores writes.	R/O	0
7.48.4	10GBASE-KR FEC negotiated	1 = PMA/PMD is negotiated to perform 10GBASE-KR FEC 0 = not negotiated	R/O	0
7.48.3	10GBASE-KR	1 = PMA/PMD is negotiated to perform 10GBASE-KR 0 = not negotiated	R/O	0
7.48.2	10GBASE-KX4	1 = PMA/PMD is negotiated to perform 10GBASE-KX4 0 = not negotiated	R/O	0
7.48.1	1000GBASE-KX	1 = PMA/PMD is negotiated to perform 1000GBASE-KX 0 = not negotiated	R/O	0
7.48.0	BP AN ability	1 = PMA/PMD is able to perform one of the preceding protocols 0 = not able	R/O	1

Designing with the Core

This chapter provides a general description of how to use the 10GBASE-R/KR core in your designs as well as describing specific core interfaces.

This chapter also describes the steps required to turn a 10GBASE-R/KR core into a fully-functioning design with user-application logic. It is important to realize that not all implementations require all of the design steps listed in this chapter. Follow the logic design guidelines in this manual carefully.

General Design Guidelines

Use the Example Design as a Starting Point

Each instance of the 10GBASE-R/KR core created by the Vivado® design tool is delivered with an example design that can be implemented in an FPGA and simulated.

This design can be used as a starting point for your own design or can be used to sanity-check your application in the event of difficulty.

See Chapter 5, Detailed Example Design, for information about using and customizing the example designs for the 10GBASE-R/KR core.

Know the Degree of Difficulty

10GBASE-R/KR designs are challenging to implement in any technology, and the degree of difficulty is further influenced by:

- Maximum system clock frequency
- Targeted device architecture
- Nature of your application

All 10GBASE-R/KR implementations need careful attention to system performance requirements. Pipelining, logic mapping, placement constraints, and logic duplication are all methods that help boost system performance.

Keep It Registered

To simplify timing and increase system performance in an FPGA design, keep all inputs and outputs registered between your application and the core. This means that all inputs and outputs from your application should come from, or connect to a flip-flop. While registering signals is not possible for all paths, it simplifies timing analysis and makes it easier for the Xilinx® tools to place and route the design.

Recognize Timing Critical Signals

The timing constraint file that is provided with the example design for the core identifies the critical signals and the timing constraints that should be applied. See Constraining the Core for further information.

Use Supported Design Flows

See Chapter 4, Design Flow Steps for more information.

Make Only Allowed Modifications

The 10GBASE-R/KR core is not user-modifiable. Do not make modifications as they can have adverse effects on system timing and protocol functionality. Supported user configurations of the 10GBASE-R/KR core can only be made by selecting the options from within the IP catalog when the core is generated. See Chapter 4, Design Flow Steps.

Clocking

The clocking schemes in this section are illustrative only and might require customization for a specific application.

Transmit and Receive Clocks

The transmit and receive datapath clocks of this core, can have different sources, depending on the core options and device family as shown in Table 3-1.

Table 3-1: **Transmit and Receive Clock Sources**

Family		Shared Logic	TX Clock Source	RX Clock Source
7 Series		In core	coreclk_out	coreclk_out
		In example design	coreclk	coreclk

Table 3-1: **Transmit and Receive Clock Sources *(Cont'd)***

Family		Shared Logic	TX Clock Source	RX Clock Source
UltraScale Devices	10GBASE-R	In core	txusrclk2_out	txusrclk2_out when the RX elastic buffer is present.
		In example design	txusrclk2	txusrclk2 when the RX elastic buffer is present.
		In either core or example design	N/A	rxrecclk_out when the RX elastic buffer is omitted.
	10GBASE-KR	In core	coreclk_out	coreclk_out
		In example design	coreclk	coreclk

Reference Clock

For Zynq-7000, Virtex-7, and Kintex-7 devices, the transceiver differential reference clock (`refclk_p/refclk_n` ports) must run at 156.25 MHz, with the exception that, for 32-bit 10GBASE-R cores, the differential reference clock must run at 312.5 MHz.

For UltraScale devices, `refclk` can run at 156.25, 161.13, 312.5, and 322.26 MHz.

Transceiver Placement

For 7 series, a single IBUFDS_GTE2 block is used to feed the reference clocks for up to 12 GTXE2_CHANNEL and GTHE2_CHANNEL transceivers, through GTXE2_COMMON or GTHE2_COMMON blocks. The COMMON blocks can each be shared by up to 4 CHANNEL blocks in the same quad.

For details about Zynq-7000, Virtex-7, and Kintex-7 device transceiver clock distribution, see the *7 Series Transceivers User Guide* (UG476) [Ref 4].

The same scheme is also valid in UltraScale devices using IBUFDS_GTE3, GTHE3_CHANNEL, GTYE3_CHANNEL, GTHE4_CHANNEL, GTYE4_CHANNEL, GTHE3_COMMON, and GTYE3_COMMON, GTHE4_COMMON and GTYE4_COMMON blocks. On UltraScale devices, it is possible to feed the reference clocks to up to 20 CHANNEL transceivers (2 QUADs above and 2 QUADs below). For details, see the *UltraScale Architecture GTH Transceivers User Guide* (UG576) [Ref 5] and the *UltraScale Architecture GTY Transceivers User Guide* (UG578) [Ref 6].

For further notes on instantiating multiple cores, see Multiple Core Instances.

XGMII 64-Bit Interface —7 Series

The clocking scheme for the XGMII interface for 7 series devices is shown in Figure 3-1.

The GT*_CHANNEL primitives require a 156.25 MHz differential reference clock, as well as 322.26 MHz TX and RX user clocks. These user clocks must be created from the `TXOUTCLK` and `RXOUTCLK` outputs respectively.

The 156.25 MHz core clock (`coreclk`) must be created from the transceiver differential reference clock to keep the user logic and transceiver interface synchronous.
A management/configuration clock, `dclk`, is used by the core and the transceiver and can be any rate that is supported for the transceiver DRPCLK.

Figure 3-1 shows the clocking architecture with a single GTXE2_CHANNEL or GTHE2_CHANNEL block. The hierarchy shown is for shared logic included in the example design (see Shared Logic).

The XGMII interface is synchronous to `coreclk`.

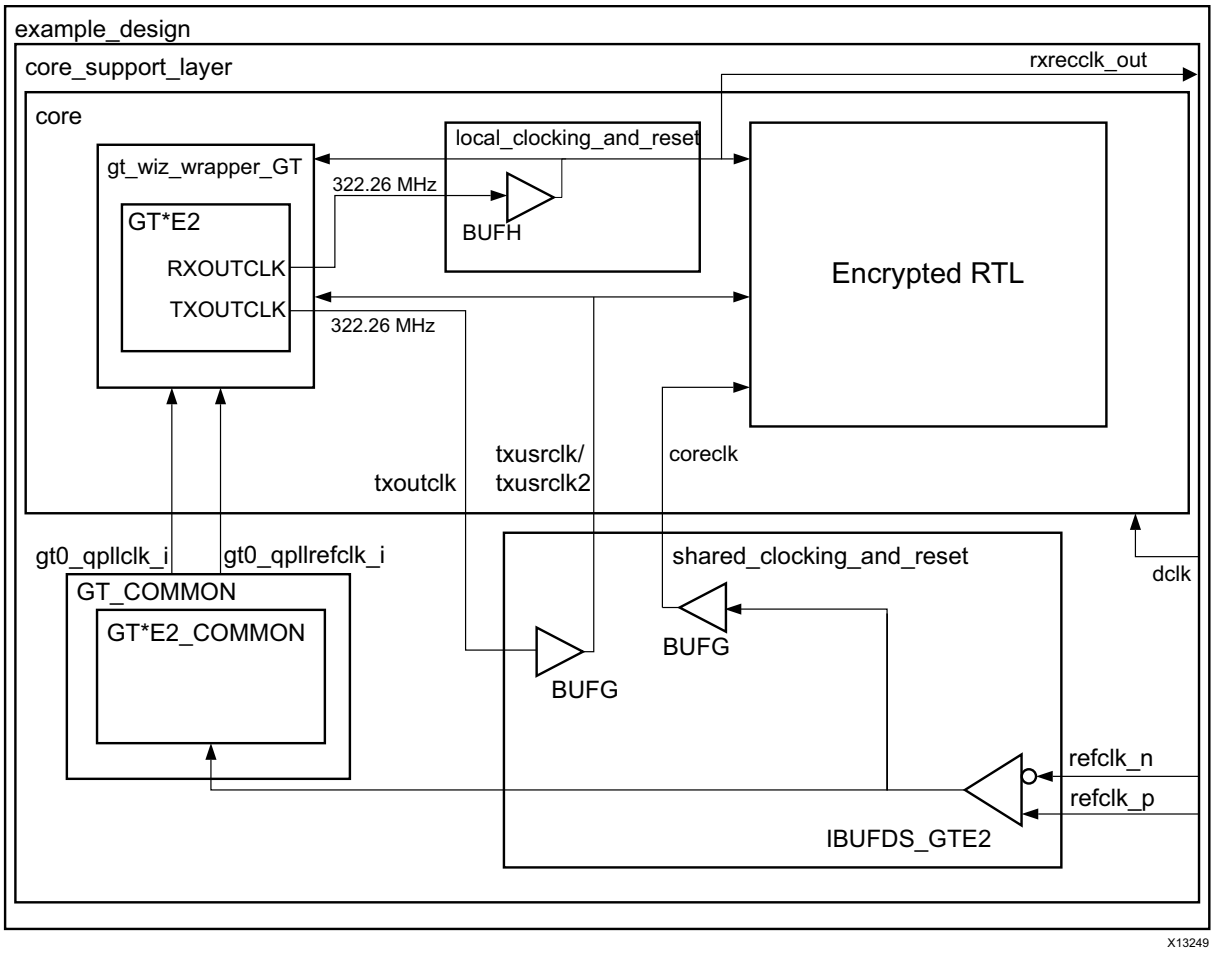

Figure 3-1: **Clocking Scheme for XGMII Interface: 7 Series FPGAs**

XGMII 64-Bit Interface—UltraScale Architecture

The clocking scheme for the XGMII interface for UltraScale™ devices is shown in Figure 3-2. The GT*_CHANNEL primitives require a differential reference clock which can be one of several frequencies, selectable at core generation time, as well as TX and RX user clocks. These user clocks must be created from the TXOUTCLK and RXOUTCLK outputs respectively.

A management/configuration clock, `dclk`, is used by the core and the transceiver and can be any rate that is supported for the transceiver DRPCLK.

In 10GBASE-KR designs, the 156.25 MHz core clock (`coreclk`) must be created from the transceiver differential reference clock to keep the user logic and transceiver interface synchronous. An MMCM can be used for generation of `coreclk` if the reference clock frequency is not 156.25 MHz or 312.5 MHz. The **Exclude RX Elastic Buffer** option is always set to False and is not editable for 10GBASE-KR designs.

In 10GBASE-R designs, the transmitter signals of the XGMII interface are synchronous to `txusrclk2`. The receiver signals of the XGMII are also synchronous to `txusrclk2` unless the **Exclude RX Elastic Buffer** option is selected, in which case they are synchronous to `rxrecclk_out`. Note that these XGMII clock sources are a change in version 6.0 of the core. Prior to version 6.0, the equivalent of `coreclk` was used instead. This change was made to reduce utilization and latency.

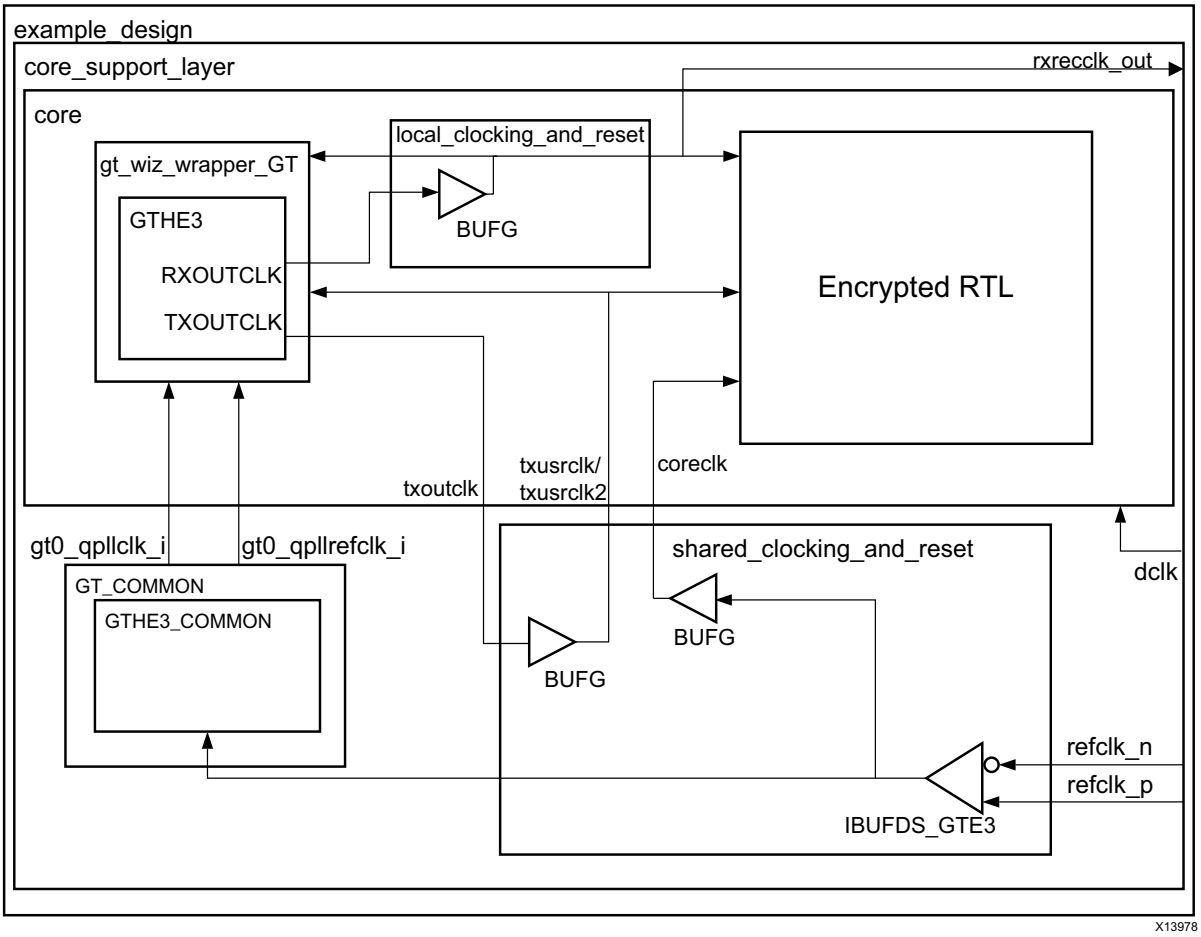

Figure 3-2: **Clocking Scheme for XGMII Interface: UltraScale Devices**

XGMII 32-Bit Interface—7 Series

The clocking scheme for 32-bit datapath 10GBASE-R cores for 7 series devices is shown in Figure 3-1. The GT*_CHANNEL primitives require a 312.5 MHz differential reference clock, as well as 322.26 MHz TX and RX user clocks. These user clocks must be created from the `TXOUTCLK` and `RXOUTCLK` outputs respectively.

The 312.5 MHz clock (`coreclk`) must be created from the transceiver reference clock to keep the user logic and transceiver interface synchronous. A management/configuration clock, `dclk`, is used by the core and the transceiver and can be any rate that is supported for the transceiver DRPCLK.

Figure 3-1 shows the clocking architecture with a single GTXE2_CHANNEL or GTHE2_CHANNEL block. This is the hierarchy if shared logic is included in the example design (see Shared Logic).

For 7 series devices, the XGMII interface is synchronous to `coreclk`.

XGMII 32-bit Interface—UltraScale Architecture

The clocking scheme for the 32-bit XGMII interface for UltraScale devices is shown in Figure 3-2. The GT*_CHANNEL primitives require a differential reference clock which can be one of several frequencies, selectable at core generation time, as well as TX and RX user clocks. These user clocks must be created from the TXOUTCLK and RXOUTCLK outputs respectively.

A management/configuration clock, `dclk`, is used by the core and the transceiver and can be any rate that is supported for the transceiver DRPCLK.

The transmitter signals of the XGMII interface are synchronous to `txusrclk2`. The receiver signals of the XGMII are also synchronous to `txusrclk2` unless the E**xclude RX Elastic Buffer** option is selected, in which case they are synchronous to `rxrecclk_out`.

Resets

All register resets within the 10GBASE-R/KR core netlist are synchronized to the relevant clock port.

Shared Logic

In earlier versions of the core, the RTL hierarchy for the core was fixed. This resulted in some difficulty because shareable clocking and reset logic needed to be extracted from the core example design for use with a single instance or multiple instances of the core. Shared Logic is a feature that provides a more flexible architecture that works both as a standalone core and as a part of a larger design with one or more core instances. This minimizes the amount of HDL modifications required, but at the same time retains the flexibility to address more core configurations.

The new level of hierarchy is called `<component_name>_support`. Figure 3-3 and Figure 3-4 show two hierarchies where the shared logic block is contained either in the core or in the example design. In these figures, `<component_name>` is the name of the generated core. The difference between the two hierarchies is the boundary of the core. It is controlled using the Shared Logic option in the Vivado IDE (see Figure 3-3).

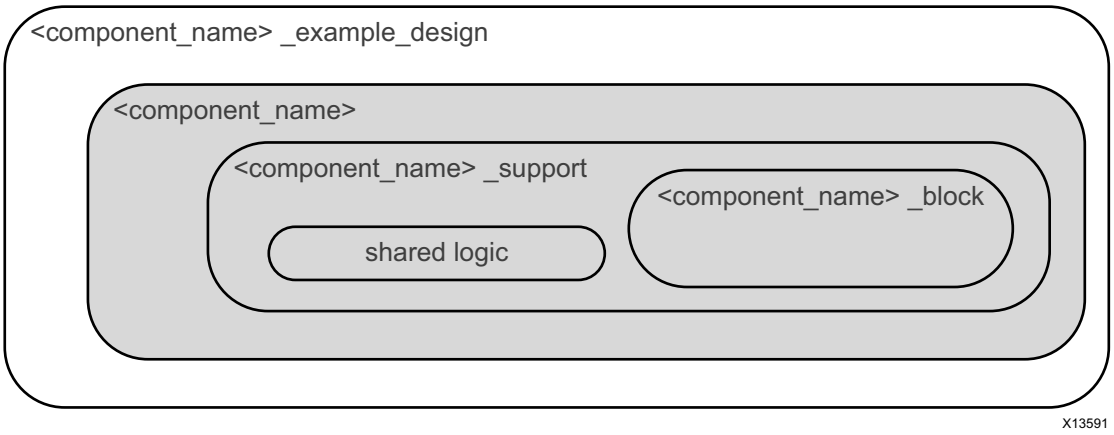

Figure 3-3: **Shared Logic Included in Core**

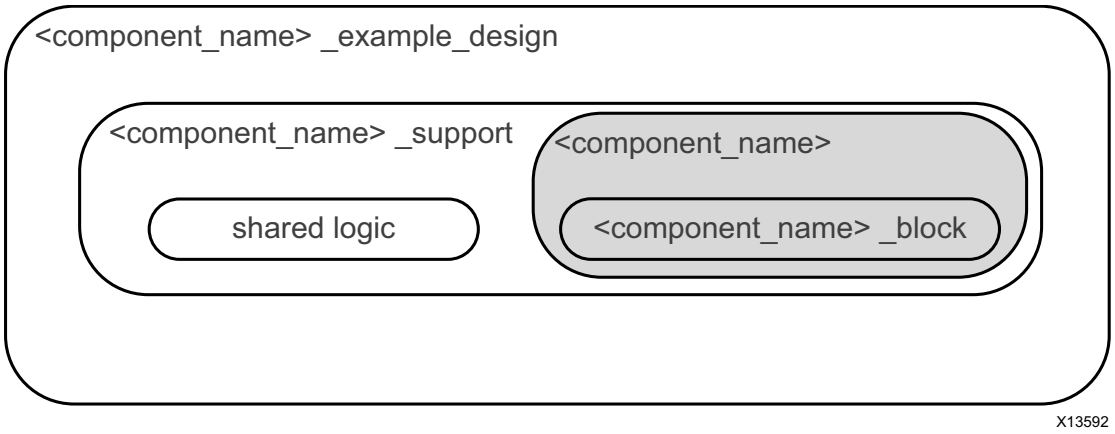

Figure 3-4: **Shared Logic Included in Example Design**

Figure 3-5 shows the core hierarchy when Shared Logic is included in the core. The component is the shaded support layer.

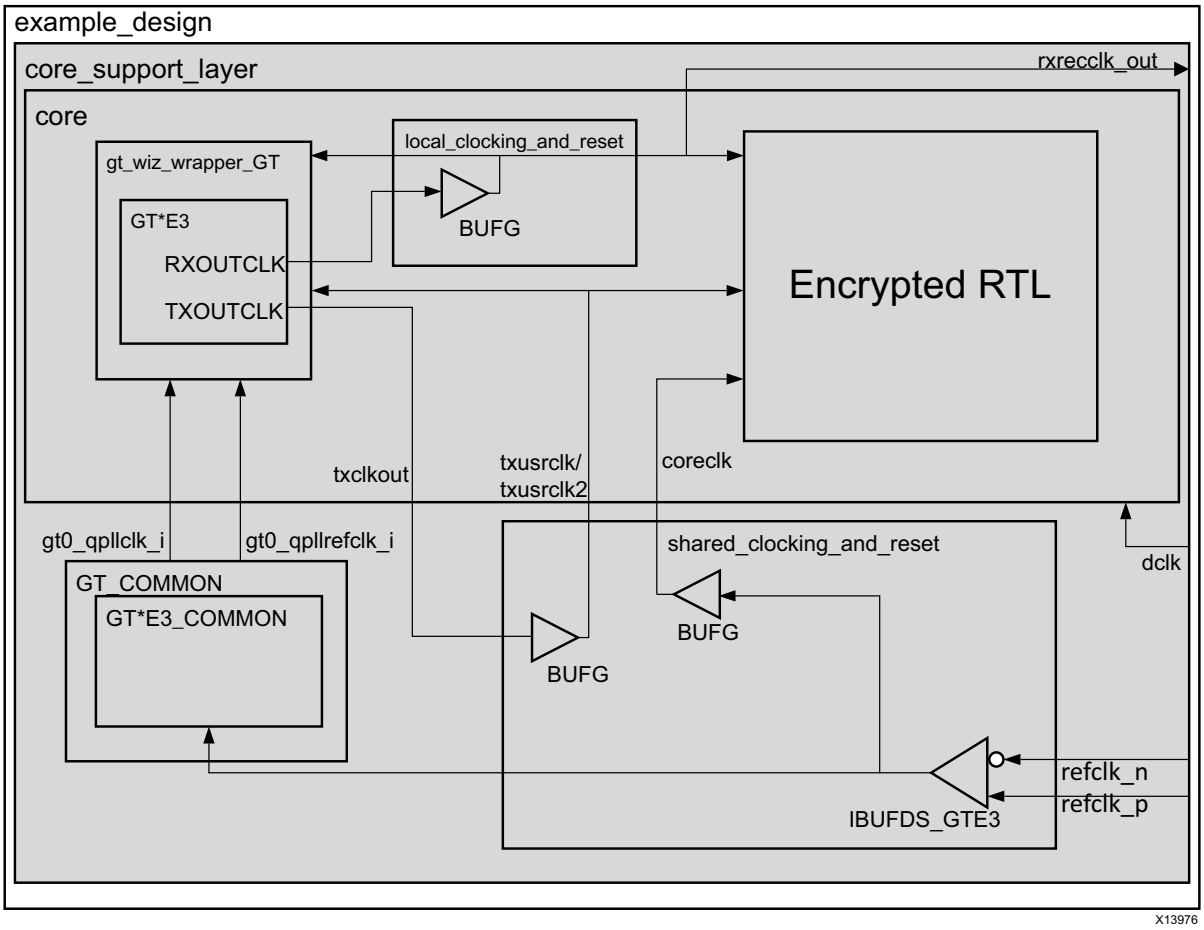

Figure 3-5: **Core Hierarchy in Shared Logic Included in Core**

Figure 3-6 shows the core hierarchy when Shared Logic is included in the example design. The component is the shaded core layer.

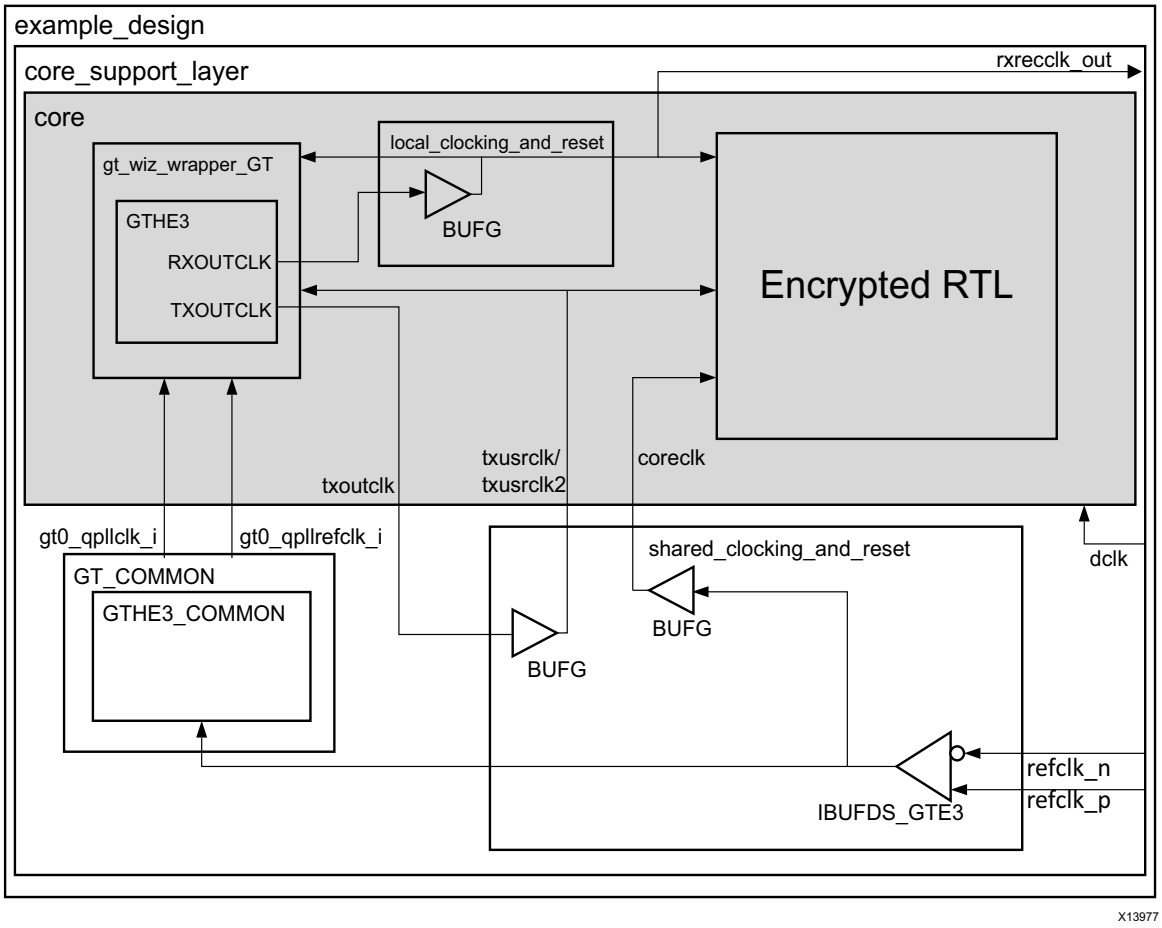

Figure 3-6: **Core Hierarchy with Shared Logic Included in Example Design**

See Special Design Considerations for more information on sharing logic between cores.

Interfacing to the Core

Interfaces to the core include the 64-bit or 32-bit XGMII data interface and the configuration and status interfaces, which use either the MDIO interface or the configuration and status vectors, depending on how the core is configured. For information on the configuration and status vectors, see Configuration and Status Signals.

Control Characters Definitions

Several XGMII control characters, defined in *IEEE Std 802.3* and reproduced in Table 3-2, are referenced in the text. These control characters, signifying Start, Terminate, and Error,

among others, have the control line for that lane set to 1 and have a specific data byte value.

Table 3-2: **Partial List of XGMII Characters**

Data (Hex)	Control	Name, Abbreviation
00 to FF	0	Data (D)
07	1	Idle (I)
FB	1	Start (S)
FD	1	Terminate (T)
FE	1	Error (E)

64-Bit Data Interface

The 64-bit data interface is required for 10GBASE-KR and is optional for 10GBASE-R core permutations. This provides a 64-bit datapath which is synchronous to a 156.25 MHz clock source at 10 Gb/s (the clock source is dependent on device architecture and can be determined from Table 3-1.)

The 64-bit single-data rate (SDR) XGMII interface is based upon the industry-standard 32-bit XGMII interface. The bus is demultiplexed from 32-bits wide to 64-bits wide on a single rising clock edge. This demultiplexing is done by extending the bus upwards so that there are now eight lanes of data numbered 0–7; the lanes are organized such that data appearing on lanes 4–7 is transmitted or received *later* in time than that in lanes 0–3.

The mapping of lanes to data bits is shown in Table 3-3. The lane number is also the index of the control bit for that particular lane; for example, `xgmii_txc[2]` and `xgmii_txd[23:16]` are the control and data bits respectively for lane 2.

Table 3-3: **XGMII_TXD, XGMII_RXD Lanes for 64-bit XGMII Interface**

Lane	XGMII_TXD, XGMII_RXD Bits
0	7:0
1	15:8
2	23:16
3	31:24
4	39:32
5	47:40
6	55:48
7	63:56

Interfacing to the Transmit XGMII Interface—64-Bits

The timing of a data frame transmission through the 64-bit interface is shown in Figure 3-7. The beginning of the data frame is shown by the presence of the Start character (the /S/

codegroup in lane 4 of Figure 3-7) followed by data characters in lanes 5, 6, and 7. Alternatively the start of the data frame can be marked by the occurrence of a Start character in lane 0, with the data characters in lanes 1 to 7.

When the frame is complete, it is completed by a Terminate character (the T in lane 1 of Figure 3-7). The Terminate character can occur in any lane; the remaining lanes are padded by XGMII idle characters. The clock source for Figure 3-7 can be determined from Table 3-1.

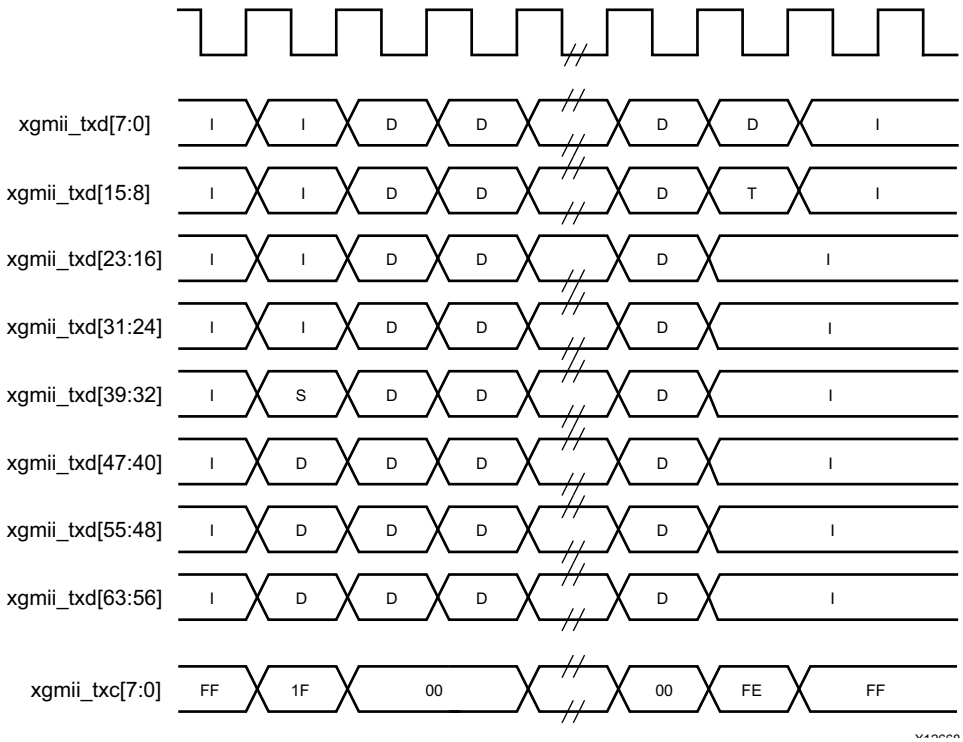

Figure 3-7: **Normal Frame Transmission Across the 64-bit XGMII Interface**

Figure 3-8 depicts a similar frame to that in Figure 3-7, with the exception that this frame is propagating an error. The error code is denoted by the letter E, with the relevant control bits set. The clock source for Figure 3-8 can be determined from Table 3-1.

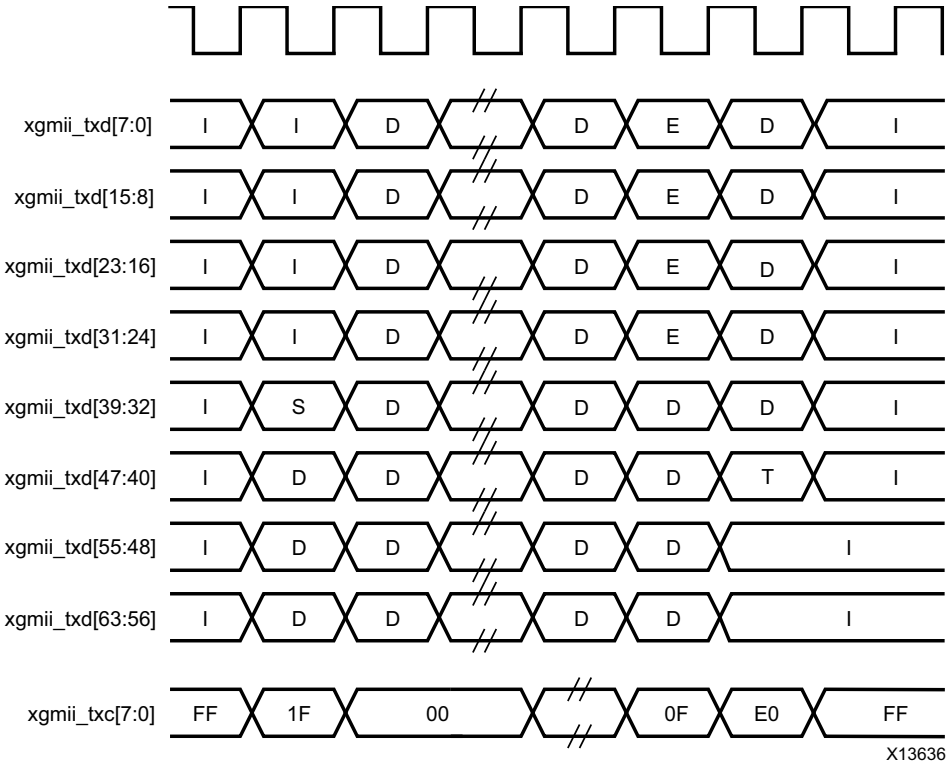

Figure 3-8: **Frame Transmission with Error Across the 64-Bit XGMII Interface**

Interfacing to the Receive XGMII Interface—64-Bits

The timing of a normal inbound frame transfer is shown in Figure 3-9. As in the transmit case, the frame is delimited by a Start character (S) and by a Terminate character (T). The Start character in this implementation can occur in either lane 0 or in lane 4. The Terminate character, T, can occur in any lane. The clock source for Figure 3-9 can be determined from Table 3-1.

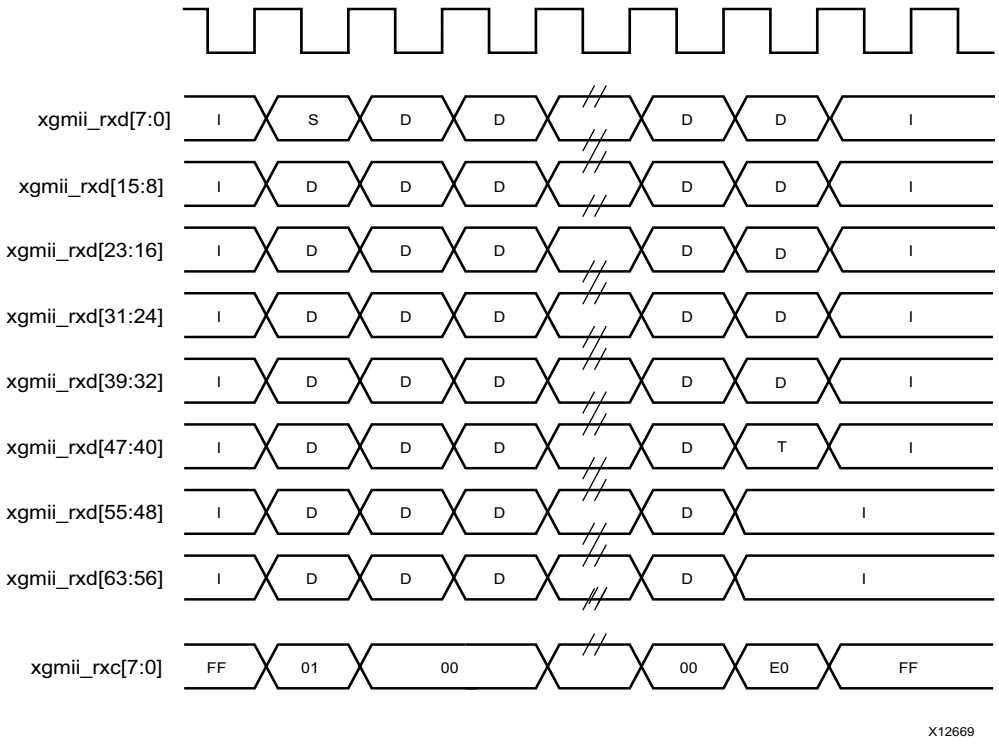

Figure 3-9: **Frame Reception Across the 64-Bit XGMII Interface**

Figure 3-10 shows an inbound frame of data propagating an error. In this instance, the error is propagated in lanes 4 to 7, shown by the letter E. The clock source for Figure 3-10 can be determined from Table 3-1.

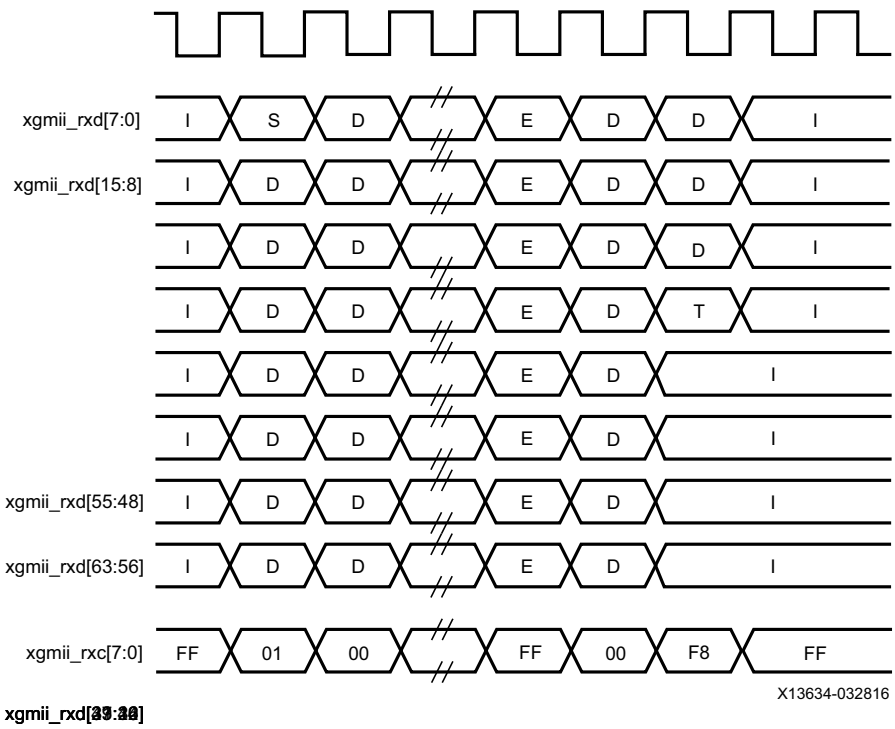

Figure 3-10: **Frame Reception with Error Across the 64-bit XGMII Interface**

32-Bit Data Interface

The 32-bit configuration of the 10GBASE-R core has four lanes of SDR data and control. A 312.5 MHz clock provides or consumes four lanes per clock tick.

The 32-bit XGMII data interface is optionally available for 10GBASE-R core permutations. This provides a 32-bit datapath which is synchronous to a 312.5 MHz clock source as shown in Table 3-1.

The 32-bit single-data rate (SDR) XGMII interface is based upon the industry-standard 32-bit XGMII interface.

The mapping of lanes to data bits is shown in Table 3-4. The lane number is also the index of the control bit for that particular lane; for example, `xgmii_txc[2]` and `xgmii_txd[23:16]` are the control and data bits respectively for lane 2.

Table 3-4: **XGMII_TXD, XGMII_RXD Lanes for 32-bit XGMII Interface**

Lane	XGMII_TXD, XGMII_RXD Bits
0	7:0
1	15:8
2	23:16
3	31:24

Interfacing to the Transmit XGMII Interface—32-Bits

The timing of a data frame transmission through the 32-bit XGMII interface is shown in Figure 3-11. The beginning of the data frame is shown by the presence of the Start character (the /S/ codegroup in lane 0 of Figure 3-11) followed by data characters in lanes 1, 2, and 3.

When the frame is complete, it is completed by a Terminate character (the T in lane 1 of Figure 3-11). The Terminate character can occur in any lane; the remaining lanes are padded by XGMII idle characters. The clock source for Figure 3-11 can be determined from Table 3-1.

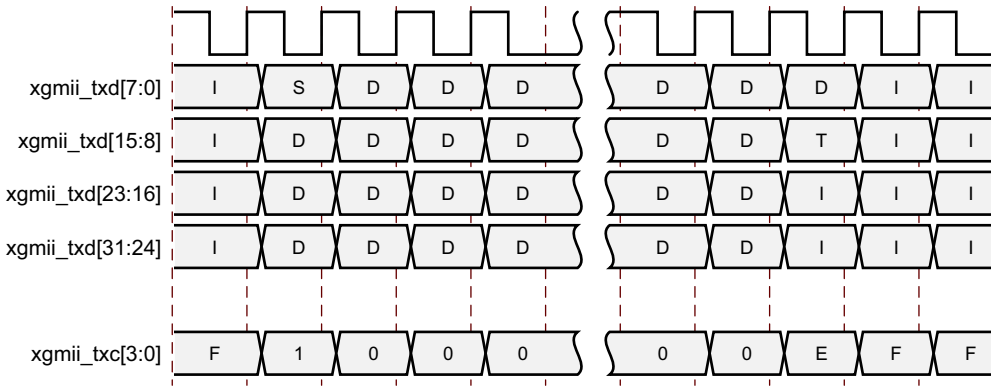

Figure 3-11: **Normal Frame Transmission Across the 32-bit XGMII Interface**

Figure 3-12 depicts a similar frame to that in Figure 3-11, with the exception that this frame is propagating an error. The error code is denoted by the letter E, with the relevant control bits set. The clock source for Figure 3-12 can be determined from Table 3-1.

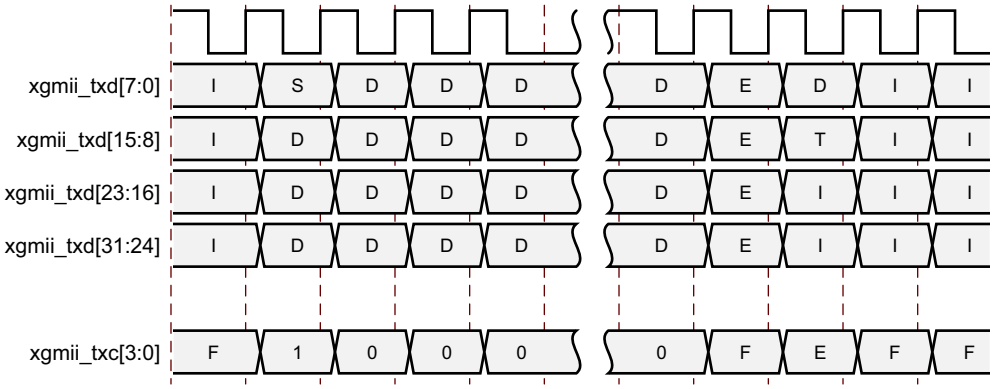

Figure 3-12: **Frame Transmission with Error Across 32-bit XGMII Interface**

Interfacing to the Receive XGMII Interface—32-Bits

The timing of a normal inbound frame transfer is shown in Figure 3-13. As in the transmit case, the frame is delimited by a Start character (S) and by a Terminate character (T). The Start character in this implementation can only occur in lane 0. The Terminate character, T, can occur in any lane. The clock source for Figure 3-13 can be determined from Table 3-1.

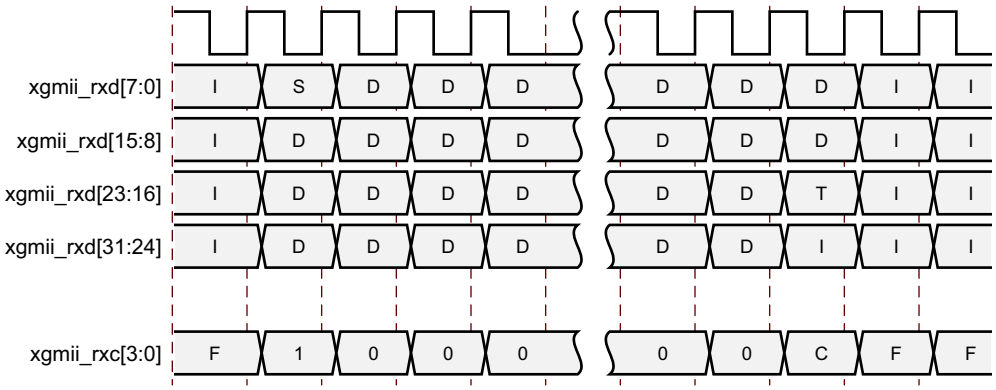

Figure 3-13: **Frame Reception Across the 32-bit XGMII Interface**

Figure 3-14 shows an inbound frame of data propagating an error. In this instance, the error is propagated in lanes 0 to 3, shown by the letter E. The clock source for Figure 3-14 can be determined from Table 3-1.

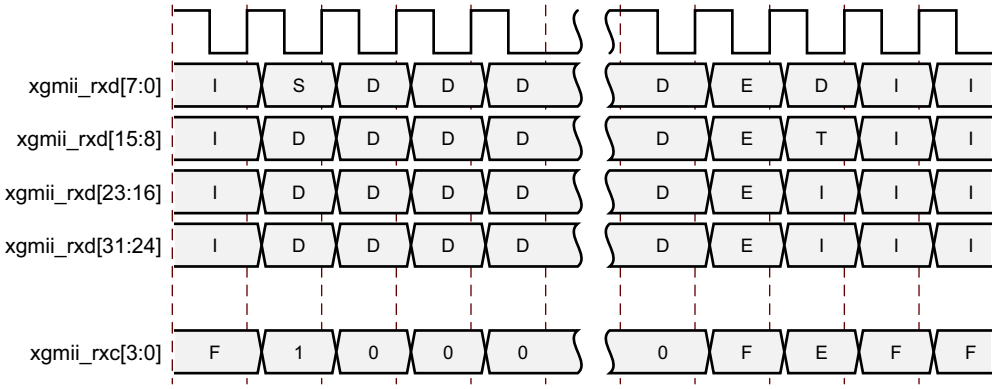

Figure 3-14: **Frame Reception with Error Across the 32-bit XGMII Interface**

MDIO Interface

The Management Data Input/Output (MDIO) interface is a simple, low-speed 2-wire interface for management of the 10GBASE-R/KR core consisting of a clock signal and a bidirectional data signal. It is defined in clause 45 of *IEEE Std 802.3*.

An MDIO bus in a system consists of a single Station Management (STA) master management entity and several MDIO Managed Device (MMD) slave entities. Figure 3-15 illustrates a typical system. All transactions are initiated by the STA entity. The 10GBASE-R/KR core implements an MMD.

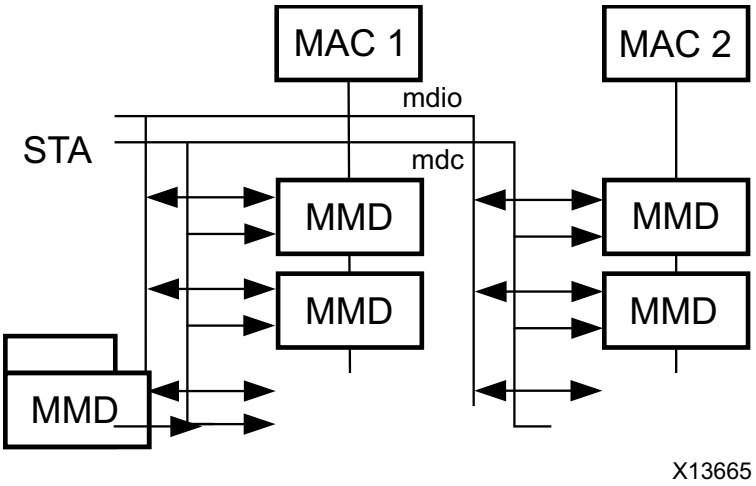

Figure 3-15: **A Typical MDIO-Managed System**

MDIO Ports

The core ports associated with MDIO are shown in Table 2-5, page 15. If implemented, the MDIO interface is implemented as four unidirectional signals. These can be used to drive a 3-state buffer either in the FPGA SelectIO™ interface buffer or in a separate device.

The `prtad[4:0]` port sets the port address of the core instance. Multiple instances of the same core can be supported on the same MDIO bus by setting the `prtad[4:0]` to specify a unique value for each instance; the 10GBASE-R/KR core ignores transactions with the PRTAD field set to a value other than that on its `prtad[4:0]` port.

MDIO Transactions

The MDIO interface should be driven from a STA master according to the protocol defined in *IEEE Std 802.3*. An outline of each transaction type is described in the following sections. In these sections, these abbreviations apply:

- PRE: preamble
- ST: start
- OP: operation code
- PRTAD: port address
- DEVAD: device address
- TA: turnaround

DEVAD

The device address in this case will be either 00001 for the PMA device or 00011 for the PCS device. For BASE-KR cores that include the optional Auto-Negotiation block, a DEVAD of 00111 should be used to access the associated Auto-Negotiation registers.

Set Address Transaction

Figure 3-16 shows a Set Address transaction defined by OP='00.' Set Address is used to set the internal 16-bit address register which is particular to the given DEVAD (called the "current address" in the following sections), for subsequent data transactions. The core contains two or three such address registers, one for PCS and one for PMA and possibly a third for Auto-Negotiation.

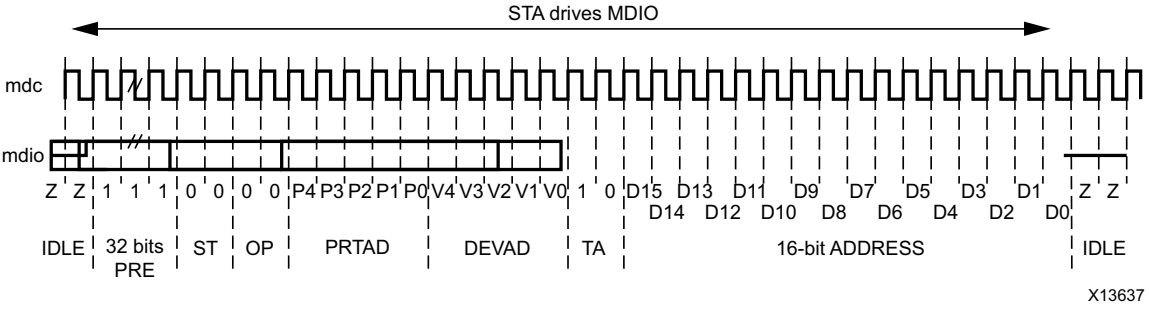

Figure 3-16: **MDIO Set Address Transaction**

Write Transaction

Figure 3-17 shows a Write transaction defined by OP=01. The 10GBASE-R/KR core takes the 16-bit word in the data field and writes it to the register at the current address.

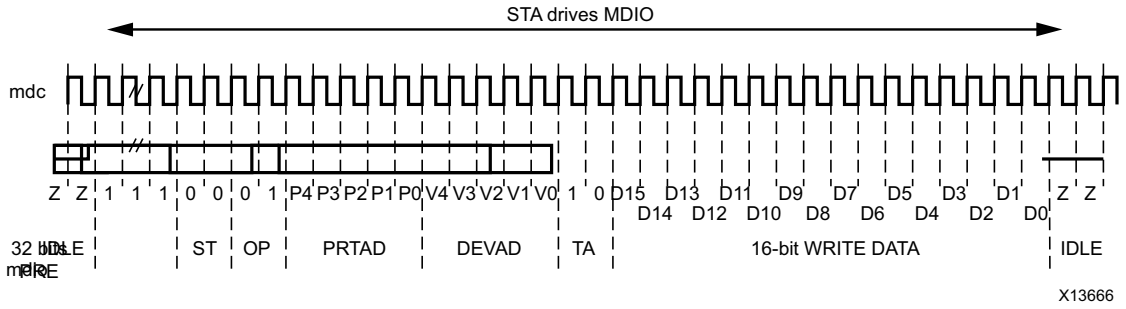

Figure 3-17: **MDIO Write Transaction**

Read Transaction

Figure 3-18 shows a Read transaction defined by OP=11. The 10GBASE-R/KR core returns the 16-bit word from the register at the current address.

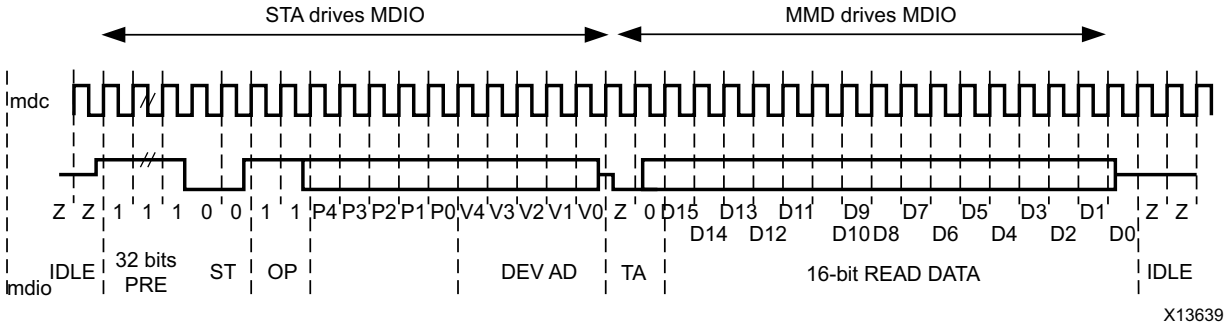

Figure 3-18: **MDIO Read Transaction**

Post-Read-increment-address Transaction

Figure 3-19 shows a Post-read-increment-address transaction, defined by OP=10. The 10GBASE-R/KR core returns the 16-bit word from the register at the current address for the given DEVAD then increments that current address. This allows sequential reading or writing by a STA master of a block of contiguous register addresses.

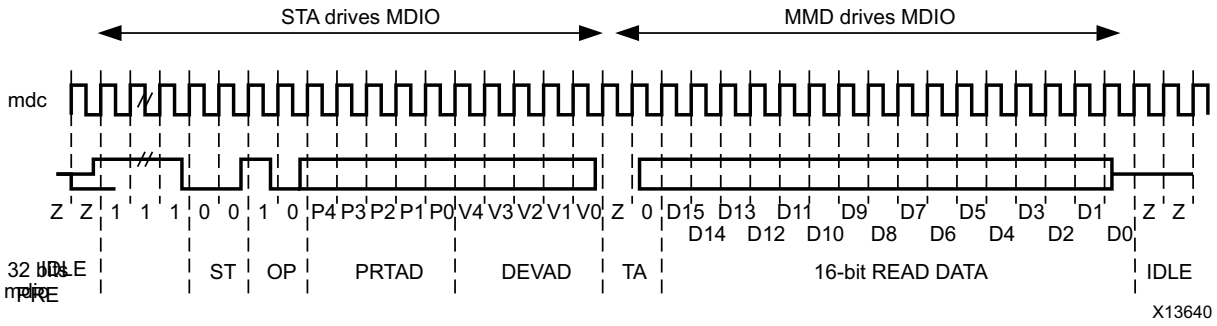

Figure 3-19: **MDIO Read-and-increment Transaction**

Special Considerations

Due to the special implementation of the PCS Test Pattern Error Counter register (3.43) for cores with the MDIO interface, whenever the MDIO PCS Address is set to point to that register ***and*** PRBS31 RX error checking is enabled in register 3.42.5, no other MDIO commands are accepted until a different PCS address is selected with an MDIO ADDRESS command. PRBS31 RX error checking requires special handling for UltraScale devices (see MDIO Register 3.43: 10GBASE-R Test Pattern Error Counter.)

The MDIO bus is not available on UltraScale devices for up to 5 ms after initiating a PMA or PCS reset.

DRP Interface

To access the DRP interface on the transceiver CHANNEL block used in the core, the interface between the core and the CHANNEL DRP ports is brought out to the boundary of the core.

The core-side signals are grouped into the `core_gt_drp_interface` (7 series) or `core_to_gt_drp` (UltraScale) interface and the GT_CHANNEL-side signals are grouped into the `user_gt_drp_interface` (7 series) or `gt_drp` (UltraScale) interface.

If access to the DRP interface is not required, then connecting one interface to the other, port-by-port, and connecting `drp_req` to `drp_gnt` allows the core to access the DRP when required (see Figure 3-20).

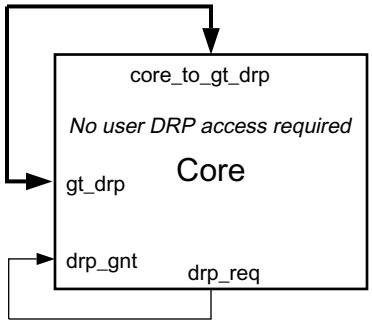

Figure 3-20: **DRP Interface Connections—No Access Required**

If access is required then an external arbiter block must be created. This block should arbitrate between the `drp_req` signals from the core and the user and connect the correct (core or user) DRP interface to the `user_gt_drp_interface` (or `gt_drp`, for UltraScale) interface on the core as arbitration requires (see Figure 3-21).

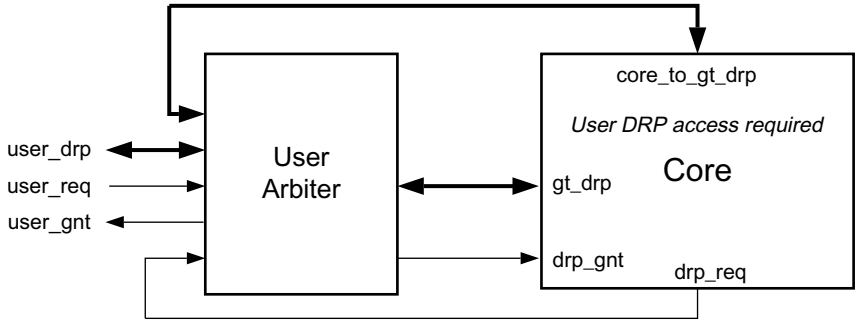

Figure 3-21: **DRP Interface Connections—External Access Required**

When the `drp_req` signal is set by the core, the core waits until it sees `drp_gnt` asserted before starting any DRP accesses.

The arbiter should hold `drp_gnt` High until `drp_req` is brought Low, allowing the core to stay in control of the transceiver DRP interface as long as it needs to. See Figure 3-21 for an example of the User Arbiter.

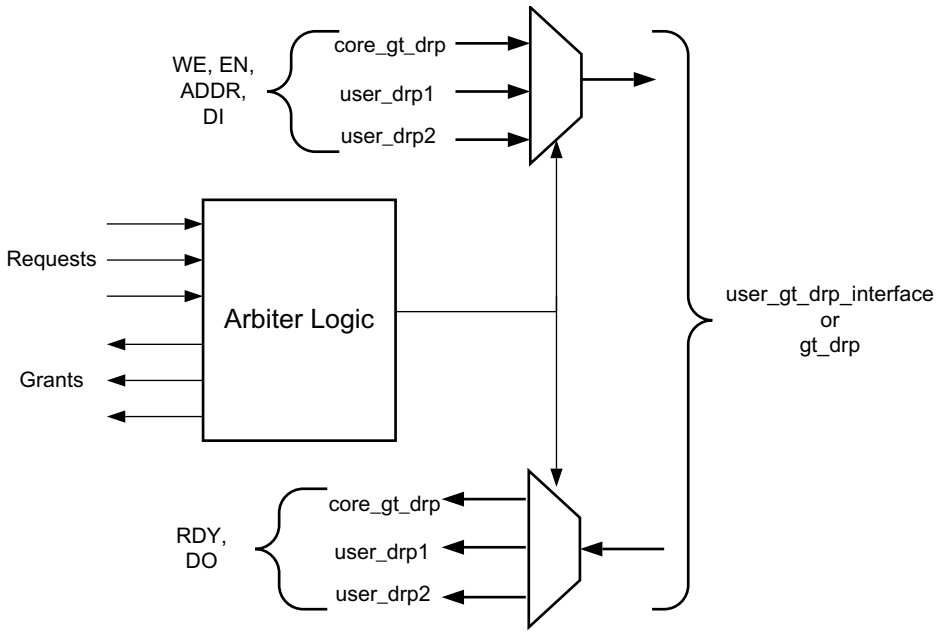

Figure 3-22: **DRP Interface—User Arbiter Example**

Receiver Termination

The receiver termination for Zynq-7000, Virtex-7, and Kintex-7 devices must be set correctly. See the *7 Series Transceivers User Guide* (UG476) [Ref 4].

For UltraScale architecture, see the *UltraScale Architecture GTH Transceivers User Guide* (UG576) [[Ref 5].

Special Design Considerations

This section describes considerations that can apply in particular design cases.

Multiple Core Instances

The example design provided with the core shows the use of a single core but it is possible to use this and the core support layer code to create a design with multiple instances of the core, as illustrated in Figure 3-23.

The GT Common block within the core support layer can be used to supply the reference clock to up to four transceivers, if they are all placed into the same GT quad.

The shared clock and reset block can be similarly shared between multiple cores. Where multiple cores are to share that block, only one `txoutclk` signal needs to be connected

from a single core instance to that shared clock and reset block. The `txoutclk` outputs from the other cores can be left dangling.

When creating a design with multiple core instances, you need to take care to replicate the correct items and not to replicate items which should be shared. There is logic in the core support layer which combines the synchronized TX and RX `resetdone` signals to create a single `resetdone_out` signal. When multiple instances of the core are required, the synchronized TX and RX `resetdone` signals from each core should be included in this combined signal.

You should be aware that the core PMA reset issues `gttxreset` and `gtrxreset` signals to the transceivers. The `gttxreset` to the transceiver results in `txresetdone` going Low and the `txoutclk` output from the transceiver being lost for a short time. This affects all cores that have a shared `txoutclk`.

Where up to four 10GBASE-R/KR cores are required which are all in the same GT_QUAD on the target device, you should generate one core with **Include Shared Logic in core** selected and a second core with **Include Shared Logic in example design** selected. On UltraScale devices, *all* other cores should be generated individually (that is, generate cores B, C and D). This is shown in Figure 3-23. The former core, core A, can be used to provide the clocks and control signals required by up to three instances of the latter, core B (or for UltraScale devices, instances of cores B, C and D), with no further editing of core output products required.

This simplifies the previous methodology where you would need to edit the core output products to produce the same result. The architecture of the multi core design resembles that in Figure 3-23.

For designs which require more than a single QUAD of transceivers, it is still possible that they can share a single IBUFDS between multiple QUADs and multiple GT_COMMON blocks. In this case, every core should be generated with **Include Shared Logic in example design** selected. The shared logic should then be manually edited to create the correct structure of the IBUFDS, GT_COMMON and GT_CHANNEL blocks.

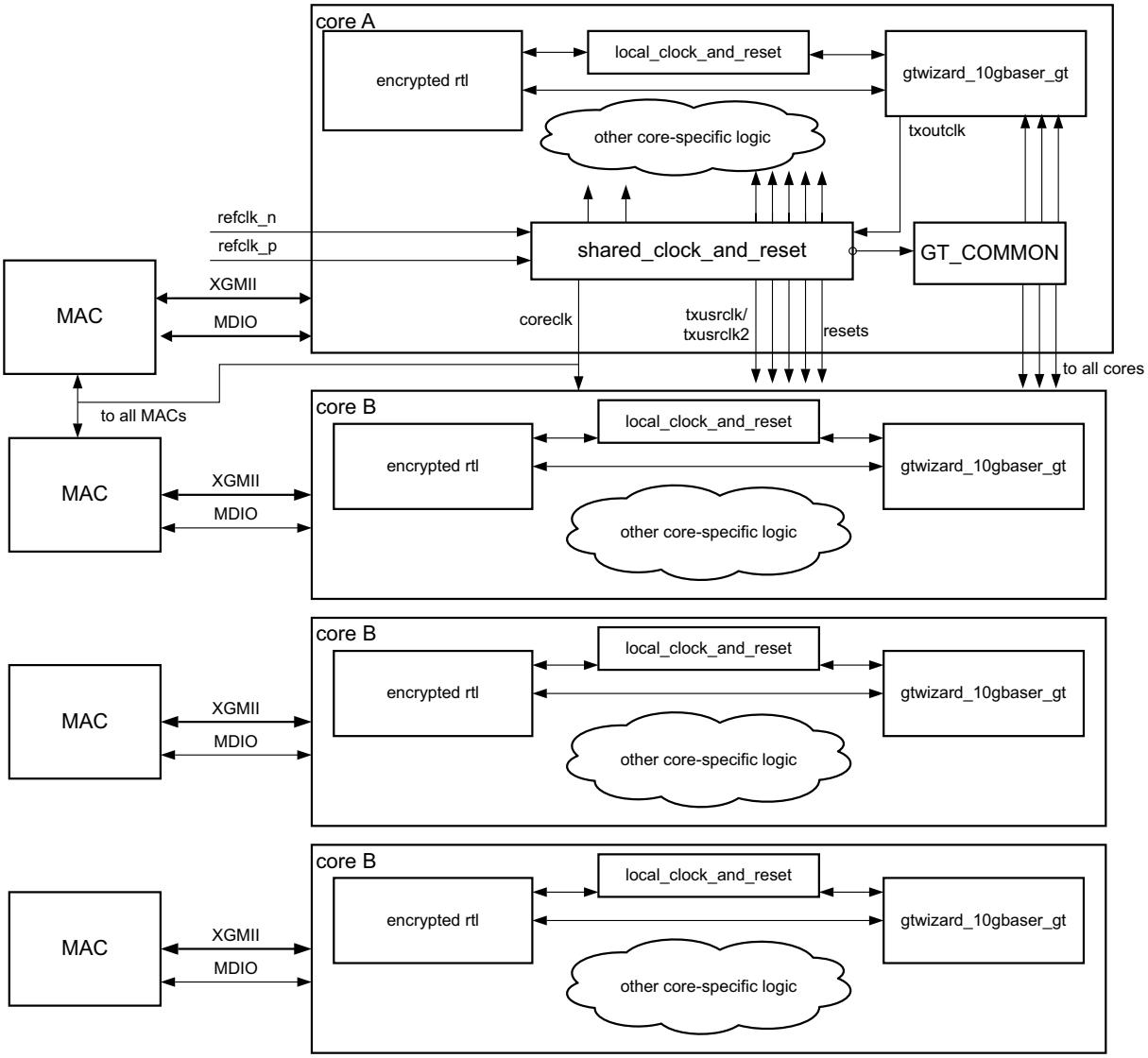

Figure 3-23: **Attaching Multiple Cores to a GT_QUAD Tile Using the Shared Logic Feature**

Using Training and Auto-Negotiation with the MDIO Interface

This section applies to UltraScale, and Virtex-7 devices.

When a BASE-KR core is created with the optional MDIO interface and with the optional Auto-Negotiation block, there are extra steps that you must take when bringing the core up in a link.

First, write to the MDIO registers to disable training (register bit 1.150.1), and then set the training restart bit (register bit 1.150.0, which will self-clear).

Then, monitor either the `core_status[5]` output from the core, or register bit 7.1.2 (which latches Low and clears on read), to wait for the AN Link Up indication which is set in

the AN_GOOD_CHECK state (see IEEE Std 803.2 [Ref 2] Clause 73, Figure 11). Now enable training (1.150.1) and then immediately restart training (1.150.0).

Training must complete within 500 ms in order for Auto-Negotiation to also complete and set AN Complete. The Training block automatically disables itself if it does get to the Training Done state.

RECOMMENDED: *Currently Xilinx recommends setting the Training Done bit – register bit 1.65520.15. This means that the core will not attempt to train the far-end device but can still be trained by the far-end device.*

If training does not complete within the time allowed by Auto-Negotiation, then you must manually disable training (register 1.150.1) and restart training (1.150.0) to allow auto-negotiation to restart the process.

Using Training and Auto-Negotiation with No MDIO Interface

This section applies to UltraScale, and Virtex-7 devices.

When a BASE-KR core is created with no MDIO interface, logic in the block level can be used to control the interaction between auto-negotiation and training.

When auto-negotiation is not included with the core, or when it is present and it reaches AN Link Up, the Training block is automatically enabled (if the Configuration vector bit 33 to enable training is also set) and restarted. If Auto-Negotiation needs to restart, training is automatically disabled until Auto-Negotiation again reaches AN Link Up.

If training is disabled using the Configuration vector bit 33, training is never run. If Auto-Negotiation is either not included with the core, or is disabled by Configuration vector bit 284, training can still be used by programming the Configuration bits to drive the process.

Using FEC in the Core with Auto-Negotiation

IMPORTANT: *When the FEC feature is recognized in the Auto-Negotiation Advertisement (Base Page Ability) data, from this and the other 10GBASE-KR device, and FEC is requested to be enabled by either end of the link, then FEC in the core is automatically enabled during the AN_GOOD_CHECK phase. This is a change in functionality from v5.0.*

The FEC Request bit is in register bit 7.21.15, or on status vector bit 383 if there is no MDIO interface.

Loopback

There are two possible loopback settings for the 10GBASE-R core and one for the 10GBASE-KR core.

Near-end PMA Loopback

This loopback option is available for all cores and uses the transceiver internal Near-end PMA Loopback feature. This mode can be activated by writing to the Loopback bit in the PMA/PMD Control 1 register (bit 1.0.0), or to the Configuration vector, bit 0, if there is no MDIO interface. Alternatively it can be activated by the `gt_loopback[2:0]` port when the transceiver debug interface is enabled. The transceiver requires a GTRXRESET after entering or exiting this loopback mode. This is done by setting the reset bit in the PMA/PMD Control 1 register (bit 1.0.15), or configuration vector bit 15, if there is no MDIO interface.

Near-end PCS Loopback

This loopback option is not available for 10GBASE-R cores on UltraScale devices that have excluded the RX elastic buffer.

While not officially supported for 10GBASE-KR in IEEE Std 802.3, the loopback path has been implemented for 10GBASE-KR cores for convenience. However, only 10GBASE-R cores transmit the expected 0x00FF pattern when in PCS loopback.

The core loops back the XGMII_TX ports directly to the XGMII_RX ports and for 10GBASE-R cores also transmits the 0x00FF pattern on the TX serial ports. This mode can be activated by writing to the loopback bit in the PCS Control 1 register (bit 3.0.14), or writing to the Configuration vector, bit 110, if there is no MDIO interface.

Design Flow Steps

This chapter describes customizing and generating the core, constraining the core, and the simulation, synthesis and implementation steps that are specific to this IP core. More detailed information about the standard Vivado® design flows and the IP integrator can be found in the following Vivado Design Suite user guides:

- *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 10]
- *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 8]
- *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 9]
- *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 7]

Customizing and Generating the Core

This section includes information about using Xilinx tools to customize and generate the core in the Vivado® Design Suite.

If you are customizing and generating the core in the IP integrator, see the *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 10] for detailed information. Vivado IDE might auto-compute certain configuration values when validating or generating the design, as noted in this section. You can view the parameter value after successful completion of the `validate_bd_design` command.

You can customize the IP for use in your design by specifying values for the various parameters associated with the IP core using the following steps:

1. Select the IP from the IP catalog.
2. Double-click the selected IP or select the Customize IP command from the toolbar or right-click menu.

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 8] and the *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 9].

Note: Figures in this chapter are illustrations of the Vivado Integrated Design Environment (IDE). This layout might vary from the current version.

Figure 4-1 displays the main screen for customizing the 10GBASE-R/KR core.

Note the following:

- When targeting devices containing only GTXE2 transceivers, the 10GBASE-KR options do not appear.
- When targeting a 7 series device, the transceiver clocking and location selections do not appear. These are still controllable through manually editing XDC LOC constraints.
- When targeting Kintex® UltraScale™ or Virtex® UltraScale devices that contain only a single type of transceiver, the **Transceiver Type** selection does not appear.
- The **Exclude RX Elastic Buffer** option is only activated for 10GBASE-R UltraScale architecture cores.

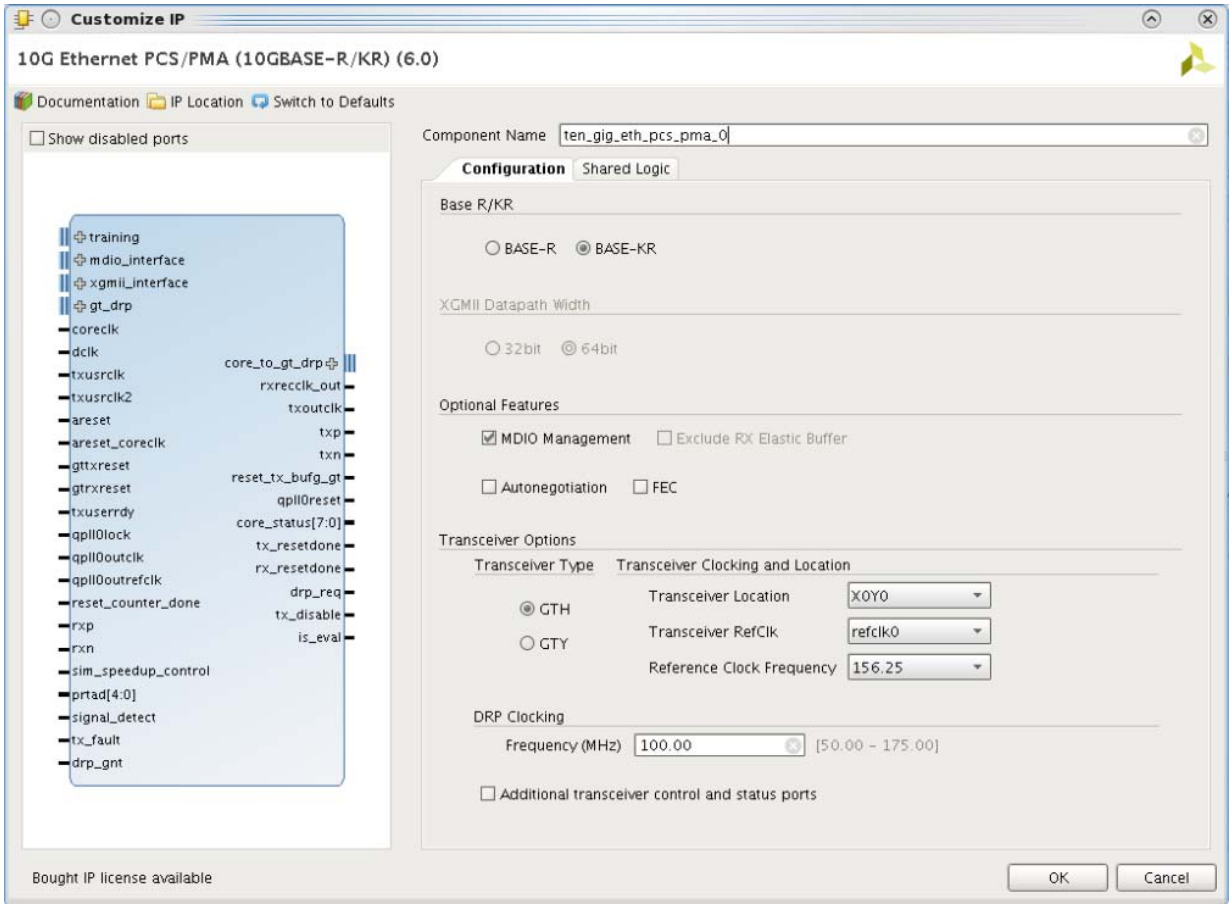

Figure 4-1: **Vivado IP Catalog 10GBASE-R/KR Main Screen**

Component Name

The component name is used as the base name of the output files generated for the core. Names must begin with a letter and must be composed from the following characters: a through z, 0 through 9 and "_" (underscore).

BASE-R or BASE-KR

Select the BASE-KR option to get a 10GBASE-KR core and have access to the Auto-Negotiation and FEC options. This is only available for devices containing GTHE2, GTHE3, or GTYE3, GTHE4, or GTYE4 transceivers.

Licensing Information

When BASE-R is selected, all licensing information displayed can be ignored. Use of the 10GBASE-R is free.

XGMII Datapath Width

Only valid for 10GBASE-R cores; selects between 32-bit and 64-bit datapaths.

Optional Features

MDIO Management

Select this option to implement the MDIO interface for managing the core. Deselect the option to remove the MDIO interface and expose a simple bit vector to manage the core.

The default is to implement the MDIO interface.

Autonegotiation

Select this option to include the Auto-Negotiation (AN) block in the 10GBASE-KR core.

FEC

Select this option to include the FEC block in the 10GBASE-KR core.

Exclude RX Elastic Buffer

Select this option, for UltraScale architecture cores, to exclude the RX elastic buffer. When selected, the receive recovered clock (RXOUTCLK) is output on the `rxrecclk_out` port to provide a clock for a downstream MAC. This allows the elastic buffer feature to be implemented outside the 10GBASE-R core.

Transceiver Options

Transceiver Type

Select the GTH or GTY transceiver to be used (for UltraScale devices that support both transceiver types).

Transceiver Clocking, Location and Frequency (UltraScale)

The order of selecting from these menus is important. Select the RefClk location first.

Transceiver Location

Use the **Transceiver Location** drop-down list to select the transceiver. The selection available in the drop-down list changes depending on the selected refclk. For example, you cannot select any of the lower two quads of transceivers if you have specified that you wish to use a refclk from two QUADs below ('South' of), because there are no refclks from two QUADs below the lowest two QUADs.

Transceiver RefClk

Use the **Transceiver RefClk** drop-down list to first select the relative location of the IBUFDS_GTE3 that is used to clock the transceiver. For example, select **refclk0+2** to use the `refclk0` signal which is provided from the IBUFDS_GTE3 block in the GT_QUAD which lies two QUADs above ('North' of) the QUAD which contains the transceiver itself.

Reference Clock Frequency

Use the **Reference Clock Frequency** drop-down list to select from the available reference clock rates for the current core configuration. The frequencies available depend on the core serial bit rate.

DRP Clocking

Use the DRP Clocking Frequency dialog box to define the DRP clock frequency which is used for the DCLK input to the core. This can be any frequency which is valid for the targeted transceiver.

Transceiver Debug

Select **Additional Transceiver Control and Status Ports** to expose additional transceiver ports on the core interface. These are detailed in Transceiver Debug Ports in Chapter 2, Table 2-12.

Shared Logic Tab

Select **Include Shared Logic in Core** if you want the core to contain the shared logic (the signals generated by the shared logic will be available on the core interface).

Otherwise the Shared Logic will be exposed in the Example Design. See Shared Logic in Chapter 3 and Special Design Considerations for more information.

User Parameters

Table 4-1 shows the relationship between the GUI fields in the Vivado IDE and the user parameters (which can be viewed in the Tcl Console).

Table 4-1: **GUI Parameter to User Parameter Relationship**

GUI Parameter/Value[(1)]	User Parameter/Value[(1)]	Default Value
Transceiver RefClk	RefClk	refclk0
Transceiver Location	Locations	Depends on device and Refclk selection
MDIO Management	MDIO_Management	true
Reference Clock Frequency	refclkrate	Depends on device and serial bitrate selection
FEC	fec	false
Autonegotiation	autonegotiation	false
Additional transceiver control and status ports	TransceiverControl	false
BASE R/KR	base_kr	Depends on device selected
BASE-R	BASE-R	
BASE-KR	BASE-KR	
Transceiver Type	vu_gt_type	GTH
GTH	GTH	
GTY	GTY	
XGMII Datapath Width	baser32	64bit
32bit	32bit	
64bit	64bit	
Shared Logic	SupportLevel	0
Include Shared Logic in core	1	
Include Shared Logic in example design	0	
Exclude RX Elastic Buffer	no_ebuff	false
DRP Clocking Frequency	DClkRate	100 MHz

Notes:

1. Parameter values are listed in the table where the GUI parameter value differs from the user parameter value. Such values are shown in this table as indented below the associated parameter.

Output Generation

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 8].

Constraining the Core

This section contains information about constraining the core in the Vivado Design Suite.

Required Constraints

This section defines the constraint requirements for the core. Constraints are provided in several XDC files which are delivered with the core and the example design to give a starting point for constraints for the user design.

There are four XDC constraint files associated with this core:

- `<corename>_example_design.xdc`
- `<corename>_ooc.xdc`
- `<corename>.xdc`
- `<corename>_clocks.xdc`

The first is used only by the example design; the second file is used for Out Of Context support where this core can be synthesized without any wrappers; the third file is the main XDC file for this core. The last file defines the constraints which depend on clock period definition, either those defined by other XDC files or those generated automatically by the Xilinx® tools, and this XDC file is marked for automatic late processing within the Vivado design tools to ensure that the definitions exist.

Device, Package, and Speed Grade Selections

You are only able to generate the core for supported device, package and speed grade combinations. –1 speed grades are not supported for this core in 7 series devices.

Clock Frequencies—7 Series

The core requires either a 156.25 MHz (for 64-bit datapath) or 312.5 MHz (for 32-bit datapath) reference clock which can be shared among several cores; so it is defined in the example design XDC file:

```
create_clock -name Q1_CLK0_GTREFCLK -period 6.400 [get_ports refclk_p]
```

or

```
create_clock -name Q1_CLK0_GTREFCLK -period 3.200 [get_ports refclk_p]
```

The reference clock is used as the main clock for the core, `coreclk`.

This clock is automatically defined and they can be shared between multiple cores so only the identifiers need be extracted in the XDC files:

```
set clk156name [get_clocks -of_objects [get_ports coreclk]]
```

or

```
set clk312name [get_clocks -of_objects [get_ports coreclk]]
```

The transceiver creates 322.26 MHz clocks that must be constrained in the XDC files. These constraints are required for devices with GTXE2 transceivers.

```
create_clock -period 3.103 [get_pins -of_objects [get_cells * -hierarchical -filter
{REF_NAME=~ GTXE2_CHANNEL}] -filter {NAME =~ *TXOUTCLK}]
set TXOUTCLK_OUT [get_clocks -of  [get_pins -of_objects [get_cells * -hierarchical
-filter {REF_NAME=~ GTXE2_CHANNEL}] -filter {NAME =~ *TXOUTCLK}]]
```

The receive recovered clock is defined in the <corename>.xdc file:

```
create_clock -period 3.103 [get_pins -of_objects [get_cells * -hierarchical -filter
{REF_NAME=~ GTXE2_CHANNEL}] -filter {NAME =~ *RXOUTCLK}]
set RXOUTCLK_OUT [get_clocks -of  [get_pins -of_objects [get_cells * -hierarchical
-filter {REF_NAME=~ GTXE2_CHANNEL}] -filter {NAME =~ *RXOUTCLK}]]
```

Similar constraints are required for devices with GTHE2.

The example design contains a DDR register that can be used to forward the XGMII_RX clock off-chip.

```
create_generated_clock -name ddrclock -divide_by 1 -invert -source [get_pins
*rx_clk_ddr/C] [get_ports xgmii_rx_clk]
set_output_delay -max 0.750 -clock [get_clocks ddrclock] [get_ports * -filter {NAME
=~ *xgmii_rxd*}]
set_output_delay -min -0.750 -clock [get_clocks ddrclock] [get_ports * -filter {NAME
=~ *xgmii_rxd*}]
set_output_delay -max 0.750 -clock [get_clocks ddrclock] [get_ports * -filter {NAME
=~ *xgmii_rxc*}]
set_output_delay -min -0.750 -clock [get_clocks ddrclock] [get_ports * -filter {NAME
=~ *xgmii_rxc*}]
```

Note: These delay values are halved for the 32-bit cores.

Clock Frequencies—UltraScale Architecture

The core requires a reference clock which can be shared among several cores; it is defined in the example design XDC file:

```
create_clock -name Q1_CLK0_GTREFCLK -period <this is variable depending on core
configuration> [get_ports refclk_p]
```

The transceiver creates more clocks but these clock frequencies are automatically derived by the Vivado IDE.

Clock Management

No Clock Management tiles (MMCMs) are required in this design or in the accompanying example design.

Clock Placement

There are no special restrictions.

Banking

All ports should be given Location constraints appropriate to your design within Banking limits

Transceiver Placement

Transceivers should be given location constraints appropriate to your design. An example of these LOC constraints can be found in the example design or core XDC file. For UltraScale devices, the placement is selectable at core configuration time.

I/O Standard and Placement

All ports should be given I/O Standard and Location constraints appropriate to your design.

These constraints are required if the optional MDIO interface is included and if the MDIO interface is on the chip boundary.

Example Design

In the XDC file:

```
set_property IOB TRUE [get_cells * -hierarchical -filter {NAME =~ *mdc_reg1*}]
set_property IOB TRUE [get_cells * -hierarchical -filter {NAME =~ *mdio_in_reg1*}]
set_property IOB TRUE [get_cells * -filter {NAME =~ *mdio_out_reg*}]
set_property IOB TRUE [get_cells * -filter {NAME =~ *mdio_tri_reg*}]
```

Simulation

For comprehensive information about Vivado® simulation components, as well as information about using supported third-party tools, see the *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 7].

IMPORTANT: *For cores targeting 7 series or Zynq®-7000 devices, UNIFAST libraries are not supported. Xilinx IP is tested and qualified with UNISIM libraries only.*

Simulation of 10GBASE-R/KR at the core level is not supported without the addition of an appropriate test bench (not supplied). Simulation of the example design is supported.

Synthesis and Implementation

For details about synthesis and implementation, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 8].

All synthesis sources required by the core are included. For this core there is a mix of both encrypted and unencrypted sources. Only the unencrypted sources are visible and optionally editable by using the **Managed IP** property option.

Detailed Example Design

This chapter contains information about the provided example design in the Vivado® Design Suite.

Example Design and Core Support Layer

This section shows an example HDL wrapper for Zynq®-7000, UltraScale™, Virtex®-7, and Kintex®-7 devices.

In Figure 5-1, the example HDL wrapper generated contains the following:

- Core Support Layer
- Pipeline registers on the XGMII interfaces, to aid timing closure when implemented stand alone.
- Double Data Rate (DDR) register on `xgmii_rx_clk`
- 10GBASE-R/KR with no MDIO interface only: Simple Logic to preserve `configuration_vector` and `status_vector` through synthesis to avoid optimizing away logic (because there are not enough I/Os on some devices, to route these signals to and preserve the logic in that way)

The Core Support Layer contains the following:

- The block-level core instance containing the encrypted RTL for the core itself, a local clocking and reset block, and a transceiver wizard wrapper for the GT CHANNEL block (`gtwizard_10gbaser_gt`). This is a per-core block and the logic cannot be shared between multiple cores.
- A wrapper for the GT COMMON block, which can be shared between up to four cores if the cores place their transceivers into the same GT quad.
- A shared clocking and reset block—can be shared between up to 12 cores in 7 series devices and up to 20 cores in UltraScale devices.

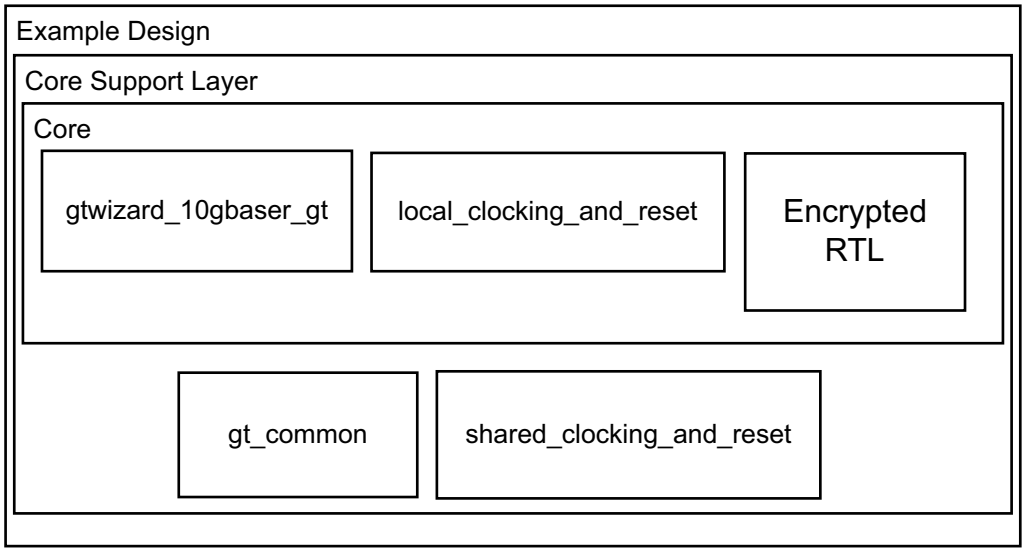

Figure 5-1: **Example HDL Wrapper for 10GBASE-R/KR**

Shared Logic and the Core Support Layer

Depending on the selection made for shared logic in the core customization Vivado IDE, the Core Support Layer can either itself be the core top-level (Include Shared Logic in core) or can simply contain the core top-level (Include Shared Logic in example design).

The difference is subtle but selecting **Include Shared Logic in the core** produces a core that includes all the shared logic and has outputs for clocks and control signals that can be shared between multiple 10GBASE-R/KR IP cores. Selecting **Include Shared Logic in the Example Design** allows you to access the shared logic.

Typically in a multi-core design, you can create one core, core A with Shared Logic included in the core, and one core, core B with the opposite setting. A single instance of core A then provides the clocks for up to three instances of core B. See Special Design Considerations for more information.

Test Bench

This chapter contains information about the test bench provided in the Vivado® Design Suite.

In Figure 6-1, the demonstration test bench is a simple VHDL or Verilog program to exercise the example design and the core itself. This test bench consists of transactor procedures or tasks that connect to the major ports of the example design and a control program that pushes frames of varying length and content through the design and checks the values as they exit the core. The test bench is supplied as part of the Example Simulation output product group.

Note that the demonstration test bench that is used to simulate the example design uses a slightly different clock period to that which the core requires in hardware. This allows the demonstration test bench to run with simplified logic and timing. The correct `refclk` is created with a period based on a bit period of 98 ps instead of the nominal 96.969696 ps, to ease simulation complexity in the demonstration test bench. A similar method is used for 32-bit 10GBASE-R cores.

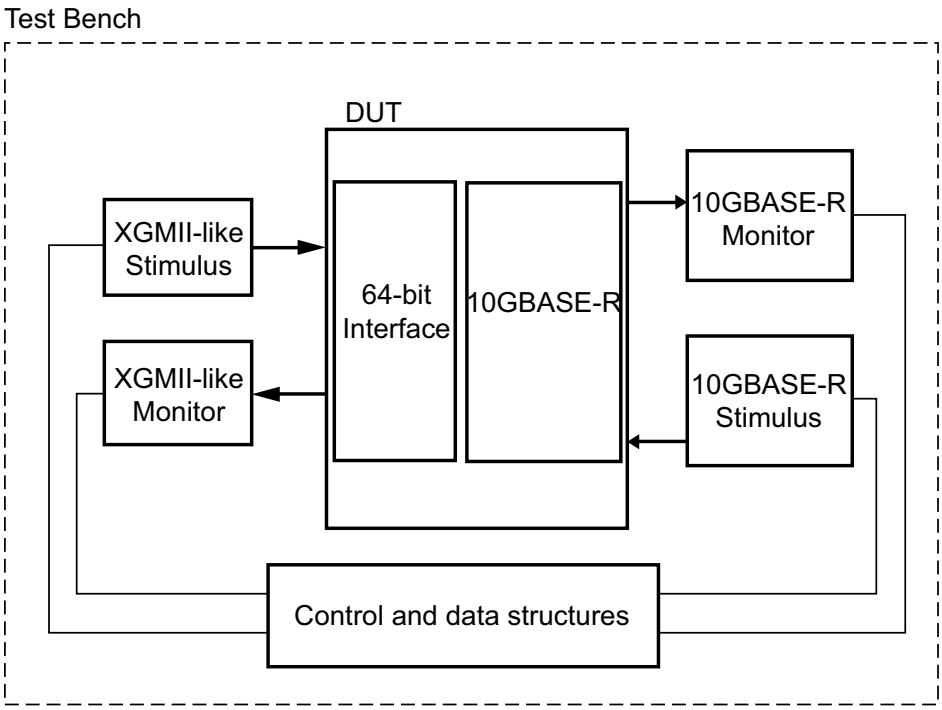

Figure 6-1: **Demonstration Test Bench**

Verification, UNH Testing, and Interoperability

The 10GBASE-R/KR LogiCORE™ IP core has been verified in the Vivado® Design Suite tools with the production Zynq®-7000, UltraScale™, Virtex®-7, and Kintex®-7 device speed files.

Simulation

A highly parameterizable transaction-based simulation test suite has been used to verify the core. Tests included:

- Register access over MDIO or Configuration/Status vectors
- Loss and re-gain of synchronization
- Loss and re-gain of alignment
- Frame transmission
- Frame reception
- Clock compensation
- Recovery from error conditions
- Auto-Negotiation phase
- Training phase
- FEC with correctable and uncorrectable errors

Hardware Verification

The 10GBASE-R and 10GBASE-KR cores have been validated on Kintex-7 and Virtex-7 devices. Hundreds of millions of Ethernet frames have been successfully transmitted and received on each board and other features such as hot-plugging, Auto-Negotiation and FEC error correction have been tested with the setup.

Testing

The 64-bit 10GBASE-R and 10GBASE-KR cores have successfully undergone validation on 7 series devices at the University of New Hampshire Interoperability Lab. Detailed test reports are available from Xilinx. All tests were successful.

Upgrading

This appendix contains information about migrating a design from ISE® to the Vivado® Design Suite, and for upgrading to a more recent version of the IP core. For customers upgrading in the Vivado Design Suite, important details (where applicable) about any port changes and other impact to user logic are included.

Migrating

For information on migrating to the Vivado Design Suite, see the *ISE to Vivado Design Suite Migration Guide* (UG911) [Ref 12].

Device Migration

If you are migrating from a 7 series GTX or GTH device to an UltraScale device, the prefixes of the optional transceiver debug ports for single-lane cores are changed from "gt0", "gt1" to "gt", and the postfix "_in" and "_out" are dropped. For multi-lane cores, the prefixes of the optional transceiver debug ports `gt(n)` are aggregated into a single port. For example: `gt0_gtrxreset` and `gt1_gtrxreset` now become `gt_gtrxreset[1:0]`. This is true for all ports, with the exception of the DRP buses which follow the convention of `gt(n)_drpxyz`.

IMPORTANT: *It is essential to be aware of clocking differences in the UltraScale implementation. See Table 3-1 and the Clocking section.*

For more information about migration to UltraScale devices, see the *UltraScale Architecture Migration Methodology Guide* (UG1026).

Upgrading in the Vivado Design Suite

This section provides information about any changes to the user logic or port designations that take place when you upgrade to a more current version of this IP core in the Vivado Design Suite.

In version 6.0 of the core, there were several changes that make the core pin-incompatible with the previous version(s). These changes were required to enhance the overall customer experience.

Shared Logic

As part of the hierarchical changes to the core, it is now possible to have the core itself include all of the logic that can be shared between multiple cores, which was previously exposed in the example design for the core.

If you are updating a version 3.* to the latest version with shared logic, there is no simple upgrade path; it is recommended to consult the Shared Logic sections of this document for more guidance.

Port Changes from v5.0 to v6.0

Ports Added

Table B-1 shows a new output port `reset_tx_bufg_gt` which appears on the core when Shared Logic is included in the example design, for 64-bit datapath cores in UltraScale devices.

The new output port `areset_coreclk_out` is added when Shared Logic is included in the core, for UltraScale devices only.

The new output port `qpll0reset` is added when Shared Logic is included in the example design, for UltraScale devices only.

The port `rxrecclk_out` is added for all core configurations when previously it was only available when the RX Elastic Buffer was omitted from UltraScale device 32-bit datapath cores. The port `qpll0reset` is added for all UltraScale devices, to connect to the shared clock and reset block, to control QPLLRESET.

Table B-1: **Ports Added in v6.0**

In/Out	Port Name	Description	What to do
Input	gt_pcsrsvdin[15:0]	Bit 2 of this vector can be used to asynchronously reset the DRP bus on UltraScale device GT_CHANNEL blocks. All other bits are tied to 0.	If unused, tie all bits to 0.
Output	reset_tx_bufg_gt	Used to reset the BUFG_GT which supplies the TXUSRCLKs to the transceiver	This signal connects to the port of the same name on the Shared Clock and Reset block

Table B-1: **Ports Added in v6.0 *(Cont'd)***

In/Out	Port Name	Description	What to do
Output	areset_coreclk_out	Master reset synchronized to the free-running reference clock	This signal connects to any user logic which needs the master reset synchronized to the coreclk.
Output	rxrecclk_out	Recovered clock from the transceiver	Use this signal to clock the XGMII RX data from the core, when the Elastic Buffer has been omitted, as well as to clock some of the transceiver debug signals (for information on the transceiver debug ports see Transceiver Debug Ports).
Output	qpll0reset	For UltraScale devices, a reset signal from the core to the QPLL located in the shared logic.	This signal connects to the port of the same name on the Shared Clock and Reset block.

Ports Changed

Table B-2 shows the ports that were changed in name only in v6.0.

Table B-2: **Ports Changed in v6.0**

In/Out	Old Port Name	New Port Name	Description	What to do
Input	clk156/clk312	coreclk	The different names for the clock have been consolidated	For cores with Shared Logic in Example Design, the clock port name has changed. Connect the existing clock to the new port name.
Input	areset_clk156/ areset_clk312	areset_coreclk	The different names for the synchronized master reset have been consolidated	For cores with Shared Logic in Example Design, the reset port name has changed. Connect the existing reset to the new port name.
Output	txclk322	txoutclk	The name for the Transceiver transmit clock has been changed	For cores with Shared Logic in Example Design, the clock port name has changed. Drive the existing clock output signal from the new port name.

Table B-2: **Ports Changed in v6.0 *(Cont'd)***

In/Out	Old Port Name	New Port Name	Description	What to do
Output	core_clk156_out/ core_clk312_out	coreclk_out	The different names for the clock output have been consolidated	For cores with Shared Logic in core, the port name has changed. Drive the existing clock output signal from the new port name. For UltraScale devices only: The new coreclk_out output is free-running at the same rate as the reference clock refclk. However, for 64-bit PCS/PMA cores, the transmit and possibly receive sides of the XGMII interface must be synchronous to the clock txusrclk2 or txusrclk2_out, depending on the core configuration. See UltraScale Clocking Change from v5.0 to v6.0 for a more detailed explanation.
Output	areset_clk156_out/ areset_clk312_out	areset_ datapathclk_ out	The different names for the synchronized master datapath reset output have been consolidated	For cores with Shared Logic in core, the port name has changed. Drive the existing clock output signal from the new port name.
Output	resetdone	resetdone_out	The name of this port has changed	For cores with Shared Logic in core, the port name has changed. Drive the existing output signal from the new port name.

UltraScale Clocking Change from v5.0 to v6.0

The clocking logic for 10GBASE-R designs for UltraScale devices using the 64-bit datapath has changed. The TX datapath (including the TX XGMII interface) is now synchronous to `txusrclk2/txusrclk2_out` (previously `clk156/coreclk156_out`). The RX XGMII interface is now also synchronous to `txusrclk2/txusrclk2_out` unless the **Exclude RX Elastic Buffer** option is selected, in which case it is synchronous to `rxrecclk_out` (previously it was synchronous to `clk156/coreclk156_out` for all 64-bit UltraScale device permutations). This change was made to reduce utilization and latency.

Debugging

This appendix includes details about resources available on the Xilinx® Support website and debugging tools.

TIP: *If the IP generation halts with an error, there might be a license issue. See License Checkers in Chapter 1 for more details.*

Finding Help on Xilinx.com

To help in the design and debug process when using the 10G Ethernet PCS/PMA core, the Xilinx Support web page contains key resources such as product documentation, release notes, answer records, information about known issues, and links for obtaining further product support.

Documentation

This product guide is the main document associated with the 10G Ethernet PCS/PMA core. This guide, along with documentation related to all products that aid in the design process, can be found on the Xilinx Support web page or by using the Xilinx Documentation Navigator.

Download the Xilinx Documentation Navigator from the Downloads page. For more information about this tool and the features available, open the online help after installation.

Solution Centers

See the Xilinx Solution Centers for support on devices, software tools, and intellectual property at all stages of the design cycle. Topics include design assistance, advisories, and troubleshooting tips. The Solution Center specific to the 10G Ethernet PCS/PMA core is the Xilinx Ethernet IP Solution Center.

Answer Records

Answer Records include information about commonly encountered problems, helpful information on how to resolve these problems, and any known issues with a Xilinx product. Answer Records are created and maintained daily ensuring that users have access to the most accurate information available.

Answer Records can also be located by using the Search Support box on the main Xilinx support web page. To maximize your search results, use proper keywords such as

- Product name
- Tool message(s)
- Summary of the issue encountered

A filter search is available after results are returned to further target the results.

Master Answer Record for the 10G Ethernet PCS/PMA Core

AR: 54669

Technical Support

Xilinx provides technical support in the Xilinx Support web page for this LogiCORE™ IP product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support if you do any of the following:

- Implement the solution in devices that are not defined in the documentation.
- Customize the solution beyond that allowed in the product documentation.
- Change any section of the design labeled DO NOT MODIFY.

To contact Xilinx Technical Support, navigate to the Xilinx Support web page.

Debug Tools

There are many tools available to address 10G Ethernet PCS/PMA core design issues. It is important to know which tools are useful for debugging various situations.

Vivado Design Suite Debug Feature

The Vivado® Design Suite debug feature inserts logic analyzer and virtual I/O cores directly into your design. The debug feature also allows you to set trigger conditions to capture application and integrated block port signals in hardware. Captured signals can then be

analyzed. This feature in the Vivado IDE is used for logic debugging and validation of a design running in Xilinx.

The Vivado logic analyzer is used with the logic debug IP cores, including:

- ILA 4.0 (and later versions)
- VIO 3.0 (and later versions)

See the *Vivado Design Suite User Guide: Programming and Debugging* (UG908) [Ref 11].

Several transceiver ports have been marked for easy access by the debug feature, including those input ports that are already driven by the core logic.

Reference Boards

Contact your Xilinx representative for information on development platforms for this IP core.

Simulation Debug

The simulation debug flow for Mentor Graphics Questa Advanced Simulator is illustrated in Figure C-1. A similar approach can be used with other simulators.

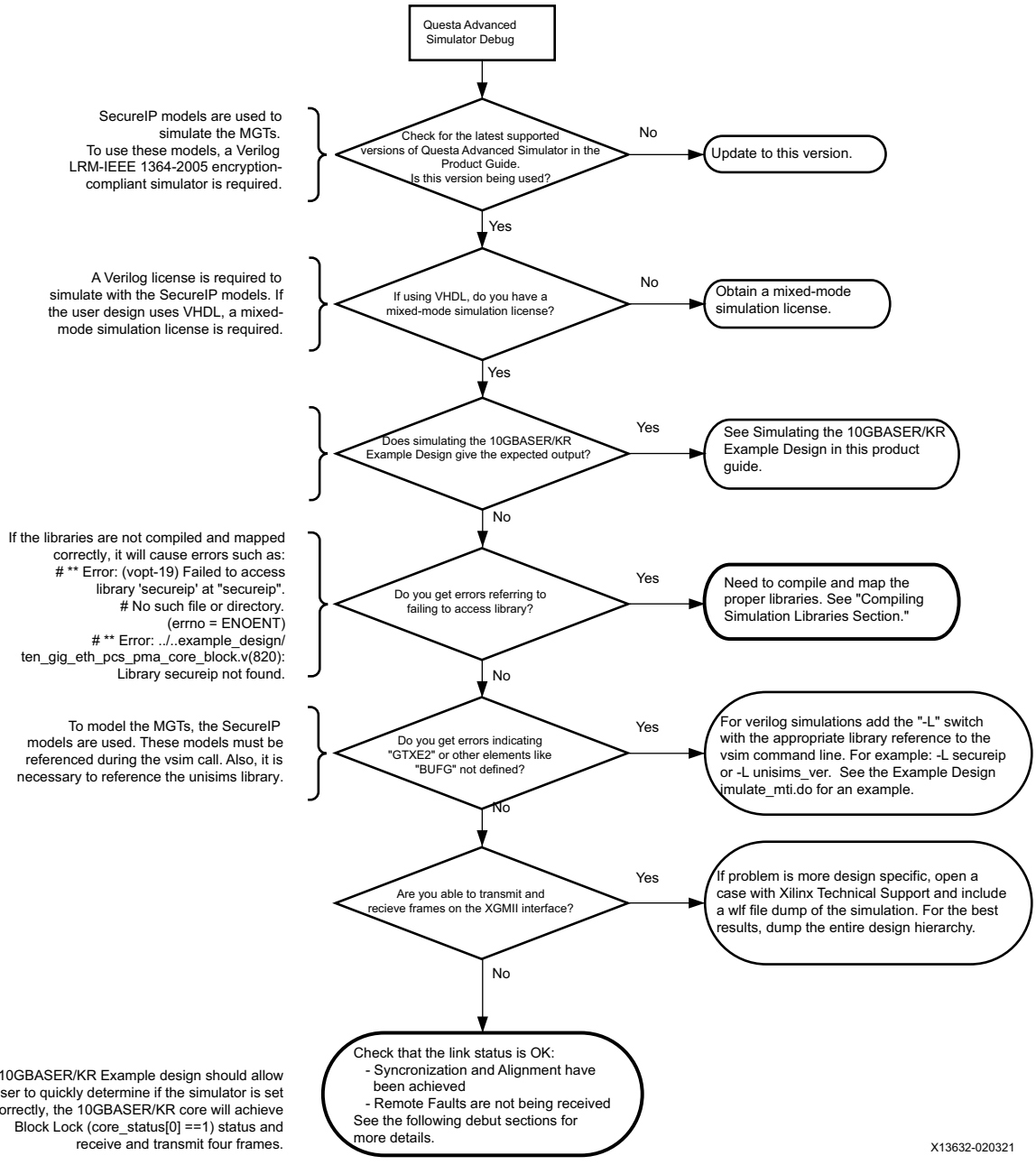

Figure C-1: **Simulation Debug Flow**

Hardware Debug

Hardware issues can range from link bring-up to problems seen after hours of testing. This section provides debug steps for common issues. The Vivado Design Suite debug feature is a valuable resource to use in hardware debug. The signal names mentioned in the following individual sections can be probed using the debug feature for debugging the specific problems. Many of these common issues can also be applied to debugging design simulations.

General Checks

Ensure that all the timing constraints for the core were properly incorporated and that all constraints were met during implementation.

- Does it work in post-place and route timing simulation? If problems are seen in hardware but not in timing simulation, this could indicate a PCB issue. Ensure that all clock sources are active and clean.
- If using MMCMs in the design, ensure that all MMCMs have obtained lock by monitoring the `LOCKED` port.
- If your outputs go to 0 after operating normally for several hours, check your licensing.

Local or Remote Fault

Local Fault and Remote Fault codes both start with the sequence TXD/RXD=0x9C, TXC/RXC=1 in XGMII lane 0. Fault conditions can also be detected by looking at the status vector or MDIO registers. The Local Fault and Link Status are defined as both immediate and latching error indicators by the IEEE specification.

IMPORTANT: *The latching Local Fault and Link Status bits in the status vector or MDIO registers must be cleared with the associated reset bits in the configuration vector or by reading the MDIO registers, or by issuing a PMA or PCS reset.*

Local Fault

The receiver outputs a local fault when the receiver is not up and operational. This RX local fault is also indicated in the status and MDIO registers. The most likely causes for an RX local fault are:

- The transceiver has not locked or the receiver is being reset.
- The block lock state machine has not completed.
- The BER monitor state machine indicates a high BER.
- The elastic buffer has over/underflowed.

Remote Fault

Remote faults are only generated in the MAC reconciliation layer in response to a Local Fault message. When the receiver receives a remote fault, this means that the link partner is in a local fault condition.

When the MAC reconciliation layer receives a remote fault, it silently drops any data being transmitted and instead transmits IDLEs to help the link partner resolve its local fault condition. When the MAC reconciliation layer receives a local fault, it silently drops any data being transmitted and instead transmits a remote fault to inform the link partner that it is in a fault condition. Be aware that the Xilinx 10GEMAC core has an option to disable remote fault transmission.

Link Bring Up—Basic

High Level Link Up (10GBASE-R or 10GBASE-KR with Auto-Negotiation + Training Disabled)

The following link initialization stages describe a possible scenario of the link coming up between device A and device B.

Stage 1: Device A Powered Up, but Device B Powered Down

1. Device A is powered up and reset.
2. Device B powered down.
3. Device A detects a fault because there is no signal received. The Device A 10Gb PCS/PMA core indicates an RX local fault.
4. The Device A MAC reconciliation layer receives the local fault. This triggers the MAC reconciliation layer to silently drop any data being transmitted and instead transmit a remote fault.
5. RX Link Status = 0 (link down) in Device A.

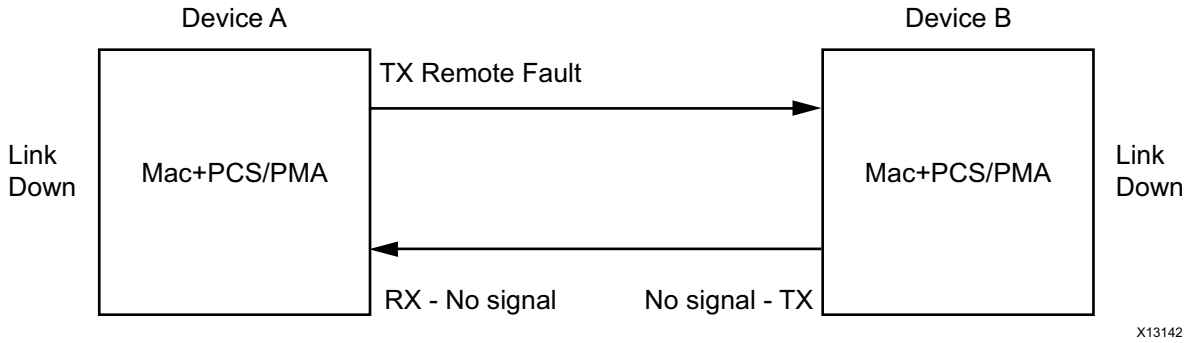

Figure C-2: **Device A Powered Up, but Device B Powered Down**

Stage 2: Device B Powers Up and Resets

1. Device B powers up and resets.
2. Device B 10Gb PCS/PMA completes block lock and high BER state machines.
3. Device A does not have block lock. It continues to send remote faults.
4. Device B 10Gb PCS/PMA passes received remote fault to MAC.
5. Device B MAC reconciliation layer receives the remote fault. It silently drops any data being transmitted and instead transmits IDLEs.
6. Link Status = 0 (link down) in both A and B.

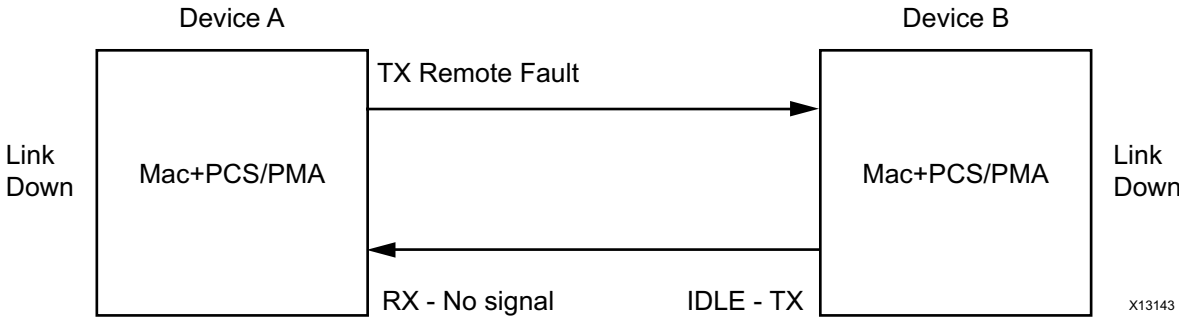

Figure C-3: **Device B Powers Up and Resets**

Stage 3: Device A Receives Idle Sequence

1. Device A PCS/PMA RX detects idles, synchronizes and aligns.
2. Device A reconciliation layer stops dropping frames at the output of the MAC transmitter and stops sending remote faults to Device B.
3. Device A Link Status=1 (Link Up)
4. When Device B stops receiving the remote faults, normal operation starts.

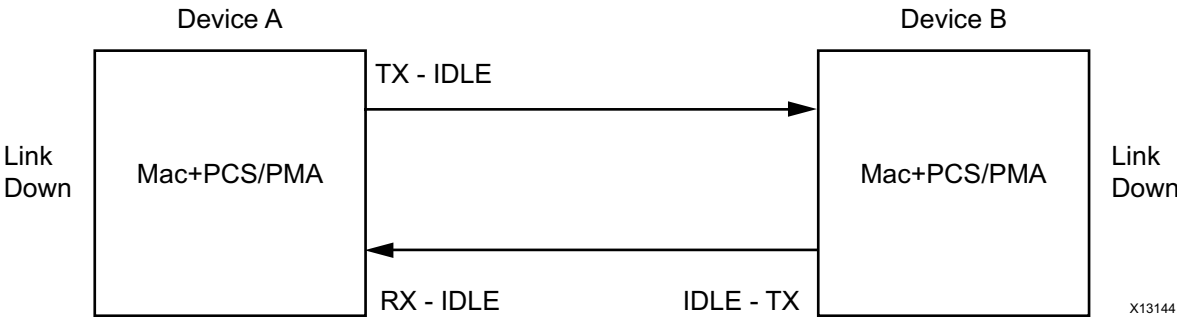

Figure C-4: **Device A Receives Idle Sequence**

Stage 4: Normal Operation

In Stage 4 shown in Figure C-5, Device A and Device B have both powered up and been reset. The link status is 1 (link up) in both A and B and in both the MAC can transmit frames successfully.

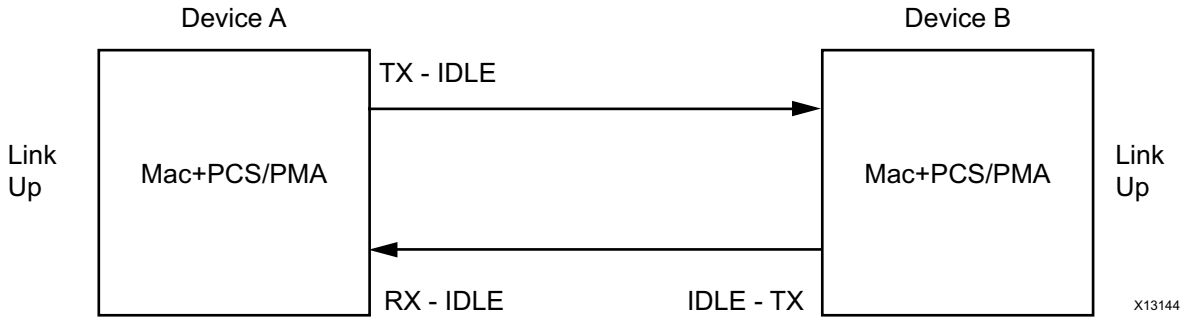

Figure C-5: **Normal Operation**

Link Bring Up—10GBASE-KR

For a 10GBASE-KR core with optional Auto-Negotiation, the bring-up of the link is more complex than for a 10GBASE-R core. First of all, both ends of the link disable transmission to bring down any existing link.

Then some low-data rate Auto-Negotiation (AN) frames are exchanged and checked at either end. The transceiver is placed into a different receive mode for this low-rate protocol. When this initial exchange of AN frames is complete, the AN_GOOD_CHECK state is entered and then transmission switches to a higher data rate training protocol frame exchange, which must complete within 500 ms of AN starting/restarting. The transceiver is placed into yet another different mode for this part of the bring-up.

Training involves detection and measurement of the received signal and transmission of commands that alter the far-end transmitter characteristics to improve that received signal. This happens in both directions until both ends of the link are receiving the best possible signal.

At that point, training is flagged as COMPLETE and the AN protocol also completes and sets the AN Link Good flag, which then enables normal Ethernet transmission and reception, with the transceiver being placed into normal operating mode. Any time that AN is restarted or reset, this entire process is repeated.

Note that if two identical 10GBASE-KR cores are powered up and reset at identical times, the pseudorandom 'nonce' generation which forms part of Auto-Negotiation will produce identical sequences of 'nonce' values which might stop the two cores from completing Auto-Negotiation. By delaying the reset to one or other core by at least two clock cycles, this can be avoided.

Using the Configuration Vector for Link Bring-Up

When the optional MDIO interface is omitted from the core, the 10GBASE-KR core block level design which is provided as part of the core deliverables contains some simple logic which automatically restarts training at the correct stage of the AN protocol. When the AN block is not included with the core, you must manually drive the training control bits of the Configuration Vector in an appropriate manner.

Using the MDIO interface for Link Bring-Up

When the optional MDIO interface is included with the core, there is no specific logic included with the 10GBASE-KR block level design to control training. You are expected to have a microprocessor controlling the system through the MDIO interface, using the Management interface on the associated MAC. They need to monitor the AN registers which show the current state of the AN protocol and drive the training protocol control registers according to the IEEE Std 802.3.

Block Lock Failure

Following are suggestions for debugging loss of Block Lock:

- Monitor the state of the `signal_detect` input to the core. This should either be:
  - connected to an optical module to detect the presence of light. Logic 1 indicates that the optical module is correctly detecting light; logic '0' indicates a fault. Therefore, ensure that this is driven with the correct polarity.
  - tied to logic 1 (if not connected to an optical module).

  Note: When `signal_detect` is set to logic 0, this forces the receiver synchronization state machine of the core to remain in the loss of sync state.
- Too many Invalid Sync headers are received. This might or might not be reflected in the HIBER output status bit or MDIO register, depending on how many invalid sync headers are received.

Transceiver-Specific:

- Ensure that the polarities of the txn/txp and rxn/rxp lines are not reversed. If they are, these can be fixed by using the TXPOLARITY and RXPOLARITY ports of the transceiver.
- Check that the transceiver is not being held in reset or still being initialized. The RESETDONE outputs from the transceiver indicate when the transceiver is ready.

10 Gb PCS/PMA Core Error Insertion

On the receive path the 10Gb PCS/PMA core receive state machine can insert block errors RXD=FE, RXC=1. The RX block error happens any time the RX_E state is entered into. It could happen if C, S, D, T are seen out of order. Or it could also happen if an /E/ block type

is received, (which is defined in IEEE Std 802.3, Section 49.2.13.2.3 as a 66-bit code with a bad sync header or a control word (C S or T) with no matching translation for the block type field.)

If the RX Elastic Buffer underflows, the core will insert an /E/ block, followed by /L/ blocks.

When FEC is enabled in register bit 1.170.0 and FEC Error Passing is enabled in register bit 1.170.1, any uncorrectable FEC errors cause the FEC block to set the two sync header bits to the same value, creating a bad sync header, which the RX PCS Decoder decodes as an /E/ block.

Transceiver Specific Checks

- Place the transceiver into parallel or serial near-end loopback.
- If correct operation is seen in the transceiver serial loopback, but not when loopback is performed through an optical cable, it might indicate a faulty optical module.
- If the core exhibits correct operation in the transceiver parallel loopback but not in serial loopback, this might indicate a transceiver issue.
- A mild form of bit error rate might be solved by adjusting the transmitter Pre-Emphasis and Differential Swing Control attributes of the transceiver.

Problems with the MDIO

See MDIO Interface for detailed information about performing MDIO transactions.

Things to check for:

- Ensure that the MDIO is driven properly. Check that the `mdc` clock is running and that the frequency is 2.5 MHz or less. Overclocking of the MDIO interface is possible with this core, up to 12.5 MHz.
- Ensure that the 10 Gb Ethernet PCS/PMA core is not held in reset.
- Read from a configuration register that does not have all 0s as a default. If all 0s are read back, the read was unsuccessful. Check that the PRTAD field placed into the MDIO frame matches the value placed on the `prtad[4:0]` port of the core.
- Verify in simulation and/or a Vivado Design Suite debug feature core capture that the waveform is correct for accessing the host interface for a MDIO read/write.

Link Training

There is currently no link training algorithm included with the core so you should implement what is required.

It has been noted in hardware testing that the RX DFE logic in the Xilinx transceivers is usually capable of adapting to almost any link so far-end training might not be required at all and the Training Done register/configuration bit can be set as default.

When you decide to implement your own training algorithm, do not only include hardware to monitor the received data signal integrity and provide the inc/dec/preset/initialize commands to send to the far end device, but also follow the protocol of sending a command until 'updated' is seen on return, and then sending 'hold' until 'not updated' is seen on return.

IMPORTANT: *Also, the priority of commands defined in IEEE Std 802.3 must be adhered to, such as never transmitting 'Preset' with 'Initialize'.*

The 10GBASE-KR core does include logic that allows it to be trained by a far-end device without user-interaction.

For special considerations when using link training with auto-negotiations see:

- Using Training and Auto-Negotiation with the MDIO Interface
- Using Training and Auto-Negotiation with No MDIO Interface

PRBS Testing

This section describes the GT based PRBS testing methodology for Asynchronous gearbox in Ultrascale™ devices. In the 10G PCS/PMA core, asynchronous gearbox in the GT is enabled for the 64B66B encoding requirement when configured for 10GBASE-R in UltraScale.

Note: BASEKR does not use the asynchronous gearbox.

1. The Gearbox must be bypassed to attempt PRBS testing, and the steps to do that are documented below and in Xilinx Answer 63704. Some of the changes are to attributes, so the DRP is required.
   a. Assert TXPCSRESET to put PCS into reset. Set the attribute TXGEARBOX_EN = 1'b0 through the DRP. Originally, the attribute is 1'b1.
   b. Set the port TXOUTCLKSEL to 3'b010 (TXOUTCLKPMA). Originally, the port is 3'b101 (TXPROGDIVCLK).
   c. Set the attribute TXBUF_EN = 1'b1 (Enable TX Buffer) through the DRP. Originally, the attribute is 1'b0.
   d. Set TXPRBSEL and RXPRBSEL to the desired PRBS pattern. Originally, the pattern is 4'b0000 (Pat Gen/Chk off).
   e. Release TXPCSRESET.
2. These are the additional details to help with doing PRBS testing with 10-Gigabit Ethernet PCS/PMA IP.
   a. Set Configuration vector, bit[245:244]=11, Table 2-15, page 27 PRBS TX test pattern enable and RX checking.
   b. Enable the "Additional Transceiver Control and Status Ports" option with the IP. Some of the ports required - TXPCSRESET, TXOUTCLKSEL for PRBS testing are enabled with this option. To know the recommended connections to the other ports enabled with this option, please generate the IP example design. The example design top file shows the recommended connections to the additional ports.
   c. In addition, please note that the external arbiter block is required to drive GT DRP interface. Please refer to page 20 of (PG068) for more details.
   d. Please monitor gt_rxprbserr, gt_rxprbslocked signals to know the status of PRBS checker. They are available when the **Additional Transceiver Control** and **Status Ports** option is enabled.

e. The 10GBASE-R PCS BER counter in the status vector does not show the error count from the PRBS checker in GT. The RX_PRBS_ERR_CNT attribute in the GT can be read using the GT DRP port to know the error count captured with PRBS checker. In UltraScale FPGAs, read out the lower 16 bits at address 0x15E first, followed by the upper 16 bits at address 0x15F.

f. The PRBS Error counter in the GT will start counting after RXPRBSLOCKED is asserted High.

Note: To return to Async Gearbox mode, undo all of the changes and reset the PCS.

Appendix E

Additional Resources and Legal Notices

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see Xilinx Support.

Documentation Navigator and Design Hubs

Xilinx® Documentation Navigator (DocNav) provides access to Xilinx documents, videos, and support resources, which you can filter and search to find information. To open DocNav:

- From the VivadoR IDE, select Help ? Documentation and Tutorials.
- On Windows, select Start ? All Programs ? Xilinx Design Tools ? DocNav.
- At the Linux command prompt, enter docnav.

Xilinx Design Hubs provide links to documentation organized by design tasks and other topics, which you can use to learn key concepts and address frequently asked questions. To access the Design Hubs:

- In DocNav, click the Design Hubs View tab.
- On the Xilinx website, see the Design Hubs page.

Note: For more information on DocNav, see the Documentation Navigator page on the Xilinx website.

References

These documents provide supplemental material useful with this product guide.

1. *IEEE Standard 802.3-2012*, Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications (standards.ieee.org/findstds/standard/802.3-2012.html)
2. *IEEE Standard 802.3-2012*, Media Access Control (MAC) Parameters, Physical Layers, and Management Parameters for 10 Gb/s Operation (standards.ieee.org/findstds/standard/802.3-2012.html)
3. 10 Gigabit Ethernet Subsystem Product Guide (PG157)
4. *7 Series Transceivers User Guide* (UG476)
5. *UltraScale Architecture GTH Transceivers User Guide* (UG576)
6. *UltraScale Architecture GTY Transceivers User Guide* (UG578)
7. *Vivado Design Suite User Guide - Logic Simulation* (UG900)
8. *Vivado Design Suite User Guide: Designing with IP* (UG896)
9. *Vivado Design Suite User Guide: Getting Started* (UG910)
10. *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
11. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)
12. *ISE to Vivado Design Suite Migration Guide* (UG911)
13. *Vivado Design Suite User Guide - Implementation* (UG904)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
02/04/2021	6.0	• General updates.
10/05/2016	6.0	• Updated 10GBASE-KR to use Synchronous gearbox instead of Asynchronous gearbox • Power-down signals added to transceiver debug interface
04/06/2016	6.0	• Remove support for UltraScale+ families. • Added Exception for txoutclk sharing in UltraScale 10GBASE-KR cores.
11/18/2015	6.0	Added support for UltraScale+ families.
09/30/2015	6.0	• Transceiver debug ports added • IDE updated

Date	Version	Revision
04/01/2015	6.0	• Added ability to select transceiver reference clock frequency for UltraScale device core configurations • Added ability to remove the RX Elastic Buffer for all UltraScale device core configurations • Now using the 64bit transceiver interface in UltraScale • Renamed and added some core ports.
10/01/2014	5.0	• Added new core generation options, new ports and interfaces.
06/06/2014	4.1	• Added information about migrating transceiver ports to UltraScale devices. • Updated the register addresses for MDIO registers.
06/04/2014	4.1	• Updated the loopback bit of the PMA/PMD Control 1 register.
04/02/2014	4.1	• Several minor clarifications of information. • Added new information on using MDIO to access the PCS Test Pattern Error Counter register.
12/18/2013	4.1	• Added UltraScale™ architecture support. • Added and changed some transceiver debug ports. • Updated Core Latency description. • Added two more illustrations of Shared Logic in Chapter 3.
10/02/2013	4.0	• Revision number changed to 4.0 to align with core version number. • Updated core interfaces. • Updated validation status. • Added more material on elastic buffer. • Added more information on core_status vector. • Added Upgrading from Previous Version(s) section to Migration chapter • Updated Figure 3-14. • Updated screen capture in Chapter 4.
03/20/2013	4.0	• Updated for Vivado Design Tools and core version 3.0. • Removed all ISE® design tools and Virtex®-6 FPGA material. • Updated core hierarchy, constraints, diagrams, resource numbers, and screen captures.
12/18/2012	3.0	• Updated for 14.4, 2012.4, and core version 2.6 • Updated Debugging appendix. • Updated false path command and maximum delay paths. • Updated resource numbers. Updated screen captures.
10/16/2012	2.0	• Updated for 14.3 and 2012.3 • Added description of MDIO Register 3.4: PCS Speed Ability • Updated for ease of use of document; Updated core interfaces; Added more descriptions of core usage.
07/25/2012	1.0	Initial Xilinx product guide release. This document is based on ds739 and ug692.

Please Read: Important Legal Notices

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 Xilinx's limited warranty, please refer to Xilinx's Terms of Sale which can be viewed at https://www.xilinx.com/legal.htm#tos; 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 such critical applications, please refer to Xilinx's Terms of Sale which can be viewed at https://www.xilinx.com/legal.htm#tos.

AUTOMOTIVE APPLICATIONS DISCLAIMER

AUTOMOTIVE PRODUCTS (IDENTIFIED AS "XA" IN THE PART NUMBER) ARE NOT WARRANTED FOR USE IN THE DEPLOYMENT OF AIRBAGS OR FOR USE IN APPLICATIONS THAT AFFECT CONTROL OF A VEHICLE ("SAFETY APPLICATION") UNLESS THERE IS A SAFETY CONCEPT OR REDUNDANCY FEATURE CONSISTENT WITH THE ISO 26262 AUTOMOTIVE SAFETY STANDARD ("SAFETY DESIGN"). CUSTOMER SHALL, PRIOR TO USING OR DISTRIBUTING ANY SYSTEMS THAT INCORPORATE PRODUCTS, THOROUGHLY TEST SUCH SYSTEMS FOR SAFETY PURPOSES. USE OF PRODUCTS IN A SAFETY APPLICATION WITHOUT A SAFETY DESIGN IS FULLY AT THE RISK OF CUSTOMER, SUBJECT ONLY TO APPLICABLE LAWS AND REGULATIONS GOVERNING LIMITATIONS ON PRODUCT LIABILITY.

© Copyright 2012–2021 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, ISE, Kintex, Spartan, Virtex, Vivado, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. PCI, PCIe and PCI Express are trademarks of PCI-SIG and used under license. All other trademarks are the property of their respective owners.

Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

Xilinx:

EF-DI-25GBASE-KR-PROJ EF-DI-25GBASE-KR-SITE EF-DI-25GBASE-KR-WW