# LogiCORE IP Ten Gigabit Ethernet PCS/PMA v2.3

## *User Guide*

**UG692 April 24, 2012**

XILINX®



# Revision History

The table shows the revision history for this document.

| Date | Version | Revision |
|---|---|---|
| 12/02/09 | 1.1 | Initial Xilinx release. |
| 04/19/10 | 1.2 | Updated to core version 1.2; updated to Xilinx tools 12.1. |
| 03/01/11 | 2.1 | Updated to core version 2.1; updated to Xilinx tools 13.1. |
| 10/19/11 | 2.2 | Updated to core version 2.2 and Xilinx tools 13.3. Revised resources in the IP Facts table and slice counts in Table 7 and Table 8. Changed transmit and receive data widths to 32 bits for Virtex®-7 and Kintex™-7 devices. |
| 04/24/12 | 2.3 | Updated to core version 2.3; updated to Xilinx tools 14.1.<br>Initial release of 10GBASE-R/10GBASE-KR version.<br>Added support for Virtex-7 GTHE2 transceivers |

# Table of Contents


**Revision History** . . . . . 2

## Chapter 1: Introduction

**System and Software Requirements** . . . . . 7
**About the Core** . . . . . 7
**Recommended Design Experience** . . . . . 7
**Additional Core Resources** . . . . . 8
Documentation . . . . . 8
10GBASE-R/KR Technology . . . . . 8
Ethernet Specifications . . . . . 8
Other Information . . . . . 8
**Technical Support** . . . . . 8
**Feedback** . . . . . 9
Core . . . . . 9
Document . . . . . 9

## Chapter 2: Licensing the Core

**Before you Begin** . . . . . 11
**License Options** . . . . . 11
Simulation Only . . . . . 11
Full System Hardware Evaluation . . . . . 11
Full . . . . . 12
**Obtaining Your License Key** . . . . . 12
Simulation License . . . . . 12
Full System Hardware Evaluation License . . . . . 12
Obtaining a Full License Key . . . . . 12
**Installing Your License File** . . . . . 12

## Chapter 3: Core Architecture

**System Overview** . . . . . 13
**Functional Description** . . . . . 13
**Applications** . . . . . 16
**Core Interfaces and Modules** . . . . . 17
Client-Side Interface . . . . . 17
Transceiver Data Interface - Virtex-7/Kintex-7 FPGA GTX/GTH Transceiver . . . . . 18
Transceiver Data Interface - Virtex-6 FPGA GTH Transceiver . . . . . 18
Optical Module Interface . . . . . 19
MDIO Interface . . . . . 19
Configuration and Status Signals . . . . . 19
Clocking and Reset Signals - Virtex-7/Kintex-7 FPGAs . . . . . 20
Clocking and Reset Signals - Virtex-6 FPGAs . . . . . 20
Transceiver Management Interface - Virtex-6 FPGAs . . . . . 21
Transceiver DRP Interface - Virtex-7/Kintex-7 FPGAs . . . . . 21

Training Interface - Virtex-7/Kintex-7 FPGAs, BASE-KR Only . . . . . 22
**Miscellaneous Signals - Virtex-7/Kintex-7 FPGAs** . . . . . 23

## Chapter 4: Customizing and Generating the Core

**Graphical User Interface** . . . . . 25
Component Name . . . . . 26
MDIO Management . . . . . 26
BASE-R or BASE-KR . . . . . 26
**Parameter Values in the XCO File** . . . . . 26
**Output Generation** . . . . . 27

## Chapter 5: Designing with the Core

**Use the Example Design as a Starting Point** . . . . . 29
**Know the Degree of Difficulty** . . . . . 29
**Keep It Registered** . . . . . 29
**Recognize Timing Critical Signals** . . . . . 30
**Use Supported Design Flows** . . . . . 30
**Make Only Allowed Modifications** . . . . . 30

## Chapter 6: Interfacing to the Core

**Data Interface: Internal Interfaces** . . . . . 31
Internal 64-bit SDR Client-side Interface . . . . . 31
Definitions of Control Characters . . . . . 32
**Interfacing to the Transmit Client Interface** . . . . . 33
Internal 64-bit Client-Side Interface . . . . . 33
**Interfacing to the Receive Client Interface** . . . . . 35
Internal 64-bit Client-Side Interface . . . . . 35
**Interfacing to the Transceivers** . . . . . 37
Virtex-7/Kintex-7 FPGAs . . . . . 37
Virtex-6 HXT FPGAs . . . . . 38
**Configuration and Status Interfaces** . . . . . 38
**MDIO Interface** . . . . . 38
MDIO Ports . . . . . 39
MDIO Transactions . . . . . 40
10GBASE-R PCS/PMA Register Map . . . . . 42
10GBASE-KR PCS/PMA Register Map . . . . . 43
**Configuration and Status Vectors** . . . . . 91
BASE-R . . . . . 91
BASE-KR . . . . . 92

## Chapter 7: Constraining the Core

**Device, Package, and Speed Grade Selection** . . . 97
Virtex-7/Kintex-7 FPGAs . . . 97
Virtex-6 FPGAs . . . 97
**Clock Frequencies, Clock Management, and Placement** . . . 97
Virtex-7/Kintex-7 FPGAs . . . 97
Virtex-6 FPGAs . . . 98
**Other Constraints** . . . 98
**Transceiver Placement** . . . 98
Virtex-7/Kintex-7 FPGAs . . . 98
Virtex-6 HXT FPGAs . . . 98
**MDIO** . . . 99

## Chapter 8: Design Considerations

**Virtex-7/Kintex-7 FPGAs Clocking** . . . 101
Reference Clock . . . 101
Transceiver Placement . . . 101
Internal Client-Side Interface . . . 101
**Virtex-6 FPGAs Clocking** . . . 102
Reference Clock . . . 102
Transceiver Placement . . . 102
Internal Client-Side Interface . . . 103
**Connecting Multiple Core Instances in Virtex-7 FPGAs** . . . 104
**Connecting Multiple Core Instances in Virtex-6 HXT FPGAs** . . . 105
**Using the DRP in Virtex-6 HXT FPGAs** . . . 106
**Reset Circuits** . . . 106
**Receiver Termination: Virtex-7/Kintex-7 FPGAs** . . . 106
**Receiver Termination: Virtex-6 FPGAs** . . . 106

## Chapter 9: Implementing the Core

**Pre-implementation Simulation** . . . 107
**VHDL** . . . 107
**Verilog** . . . 107
**Synthesis** . . . 108
XST: VHDL . . . 108
XST: Verilog . . . 108
**Implementation** . . . 109
Generating the Xilinx Netlist . . . 109
Mapping the Design . . . 109
Placing and Routing the Design . . . 109
Static Timing Analysis . . . 109
Generating a Bitstream . . . 109
**Post-Implementation Simulation** . . . 110
Generating a Simulation Model . . . 110
Using the Model . . . 110
**Other Implementation Information** . . . 110

## Chapter 10: Detailed Example Design

**Directory and File Contents** . . . 112
<project directory> . . . 112
<project directory>/<component name> . . . 112
<component_name>/doc . . . 113
<component_name>/example_design . . . 113
Virtex-7/Kintex-7 FPGAs . . . 114
Virtex-6 FPGAs . . . 116
**Implementation and Test Scripts** . . . 119
Implementation Script . . . 119
Setting up for Simulation . . . 119
Simulation Scripts . . . 120
**10GBASE-R/KR Core** . . . 121
Example HDL Wrapper - Virtex-7/Kintex-7 FPGAs . . . 121
Example HDL Wrapper (Virtex-6 FPGAs) . . . 122
Demonstration Test Bench . . . 123

## Chapter 11: Quick Start Example Design

**Introduction** . . . 125
**Generating the Core** . . . 126
**Implementing the 10GBASE-R/KR Example Design** . . . 127
**Linux** . . . 127
**Windows** . . . 127
**Simulating the 10GBASE-R/KR Example Design** . . . 127
Setting up for Simulation . . . 127
Pre-Implementation Simulation . . . 128
Post-Implementation Simulation (For Production Silicon Only) . . . 128
**Additional Information** . . . 128

## Appendix A: Additional Resources

**Xilinx Resources** . . . 129
**Solution Centers** . . . 129
**Technical Support** . . . 129

## Appendix B: Verification and Interoperability

## Appendix C: Core Latency

**Virtex-7/Kintex-7 FPGAs** . . . 133
Transmit Path Latency . . . 133
Receive Path Latency . . . 133
Transceiver Latency . . . 133
**Virtex-6 HXT FPGAs** . . . 133
Transmit Path Latency . . . 133
Receive Path Latency . . . 134
GTH Transceiver Latency . . . 134
Total Latency . . . 134

XILINX®

Chapter 1

# Introduction

The 10GBASE-R/KR LogiCORE™ IP core has been verified in IDS 14.1 software with the production Virtex®-6 HXT FPGA and pre-production Virtex-7/Kintex™-7 FPGA speed files. Pre-production means that the speed files are still subject to change. The 10GBASE-R/KR LogiCORE IP core and example design are provided in Verilog and VHDL.

This chapter introduces the 10GBASE-R/KR core and provides related information, including recommended design experience, additional resources, technical support, and submitting feedback to Xilinx.

What distinguishes the 10GBASE-KR core from the 10GBASE-R core is that the 10GBASE-KR core includes a Link Training block as well as optional Auto-negotiation (AN) and Forward Error Correction (FEC) features, all to help support a 10 Gb/s data stream across a backplane. The 10GBASE-R core is not suitable for use with backplanes and is only for use with optical links.

## System and Software Requirements

For system and software requirements, see the ISE Design Suite 14: Release Notes Guide.

## About the Core

The 10GBASE-R/KR core is a Xilinx® CORE Generator™ tool IP core, included in the latest Integrated Software Environment (ISE) Update on the Xilinx IP Center. For detailed information about the core, see the 10GBASE-R product page. For information about licensing options, see *Chapter 2, Licensing the Core*.

## Recommended Design Experience

Although the 10GBASE-R/KR core is a fully-verified solution, the challenge associated with implementing a complete design varies depending on the configuration and functionality of the application. For best results, previous experience building high performance, pipelined FPGA designs using Xilinx implementation software and User Constraints File (UCF) is recommended.

Contact your local Xilinx representative for a closer review and estimation for your specific requirements.

# Additional Core Resources

For detailed information about 10GBASE-R/KR technology and updates to the 10GBASE-R/KR core, see the following:

## Documentation

From the 10GBASE-R product page:

- *10GBASE-R/KR Release Notes*
- *10GBASE-R/KR Data Sheet*

From the document directory after generating the core:

- *10GBASE-R/KR Release Notes*
- *10GBASE-R/KR Data Sheet*

## 10GBASE-R/KR Technology

For information about 10GBASE-R/KR technology basics, including features, FAQs, the 10GBASE-R/KR device interface, typical applications, specifications, and other important information, see www.xilinx.com/products/ipcenter/10GBASE-R.htm.

## Ethernet Specifications

The relevant 10GBASE-R/KR standards are *IEEE Std. 802.3-2008*.

## Other Information

The 10-Gigabit Ethernet Consortium at the University of New Hampshire Interoperability Lab is an excellent source of information on 10-Gigabit Ethernet technology: www.iol.unh.edu/consortiums/10gec/index.html.

# Technical Support

For technical support, visit www.xilinx.com/support. Questions are routed to a team of engineers with expertise using the 10GBASE-R/KR core.

Xilinx provides technical support for use of this product as described in the *LogiCORE IP 10GBASE-R/KR User Guide*. Xilinx cannot guarantee timing, functionality, or support of this product for designs that do not follow these guidelines.

# Feedback

Xilinx welcomes comments and suggestions about the 10GBASE-R/KR core and the documentation supplied with the core.

## Core

For comments or suggestions about the 10GBASE-R/KR core, submit a webcase from www.xilinx.com/support. Be sure to include the following information:

- Product name
- Core version number
- Explanation of your comments

## Document

For comments or suggestions about this document, submit a webcase from www.xilinx.com/support. Be sure to include the following information:

- Document title
- Document number
- Page number(s) to which your comments refer
- Explanation of your comments

XILINX®

Chapter 2

# Licensing the Core

This chapter provides instructions for obtaining a license for the 10GBASE-KR core, which you must do before using the core in your designs. The 10GBASE-KR core is provided under the terms of the Xilinx LogiCORE IP Project Agreement. Purchase of the core entitles you to technical support and access to updates for a period of one year. 10GBASE-R IP is available at no charge under the Xilinx End User License Agreement and can be generated using the Core Generator™ tool v14.1 and higher.

## Before you Begin

This chapter assumes that you have installed all required software specified on the 10GBASE -KR product page or 10GBASE-R product page for this core.

## License Options

The 10GBASE-KR core provides three licensing options. After installing the required Xilinx ® ISE ® Design Suite and IP Service Packs, choose a license option. 10GBASE-R does not require a license key.

### Simulation Only

The Simulation Only Evaluation license key is provided with the Xilinx CORE Generator tool. This key lets you assess core functionality with either the example design provided with the core, or alongside your own design and demonstrates the various interfaces to the core in simulation. (Functional simulation is supported by a dynamically generated HDL structural model.)

### Full System Hardware Evaluation

The Full System Hardware Evaluation license is available at no cost and lets you fully integrate the core into an FPGA design, place and route the design, evaluate timing, and perform back-annotated gate-level simulation of the core using the demonstration test bench provided with the core.

In addition, the license key lets you generate a bitstream from the placed and routed design, which can then be downloaded to a supported device and tested in hardware. The core can be tested in the target device for a limited time before *timing out* (ceasing to function) at which time it can be reactivated by reconfiguring the device.

### Full

The Full license key is available when you purchase the core and provides full access to all core functionality both in simulation and in hardware, including:

- Functional simulation support
- Full implementation support including place and route, and bitstream generation
- Full functionality in the programmed device with no timeouts

## Obtaining Your License Key

This section contains information about obtaining a simulation, full system hardware evaluation, and full license keys.

### Simulation License

No action is required to obtain the Simulation Only Evaluation license key; it is provided by default with the Xilinx CORE Generator tool.

### Full System Hardware Evaluation License

To obtain a Full System Hardware Evaluation license for 10GBASE-KR, contact your Xilinx Sales Representative. 10GBASE-R full access does not require a license.

### Obtaining a Full License Key

To obtain a Full license key for 10GBASE-KR, you must purchase a license for the core. After you purchase a license, a product entitlement is added to your Product Licensing Account on the Xilinx Product Download and Licensing site. The Product Licensing Account Administrator for your site receives an email from Xilinx with instructions on how to access a Full license and a link to access the licensing site. You can obtain a full key through your account administrator, or your administrator can give you access so that you can generate your own keys.

Further details can be found at:

www.xilinx.com/products/intellectual-property/ipaccess_fee.htm

10GBASE-R does not require a license.

## Installing Your License File

The Simulation Only Evaluation license key is provided with the ISE CORE Generator system and does not require installation of an additional license file. For the 10GBASE-KR Full System Hardware Evaluation license and the Full license, an email is sent to you containing instructions for installing your license file. Additional details about IP license key installation can be found in the ISE Design Suite Installation, Licensing and Release Notes document.

XILINX®

Chapter 3

# Core Architecture

This chapter describes the overall architecture of the 10GBASE-R/KR core and also describes the major interfaces to the core.

## System 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).

## Functional Description

Figure 3-1 shows a block diagram of the implementation of the Virtex®-7/Kintex™-7 FPGA 10GBASE-R core. The major functional blocks of the core include the following:

- **Virtex-7/Kintex-7 FPGA GTX or GTH Transceiver**. Provides high-speed transceiver and partial gearbox functionality.
- **PCS Block**. Provides encode/decode, scramble/descramble, block-lock, transmit and receive state machines, test-pattern blocks and BER monitor.
- **Optional MDIO Interface**. A two-wire low-speed serial interface used to manage the core. An alternative vector-based interface might be provided instead.
- **Elastic Buffer**. Identical to that described in the next section. See page 15.

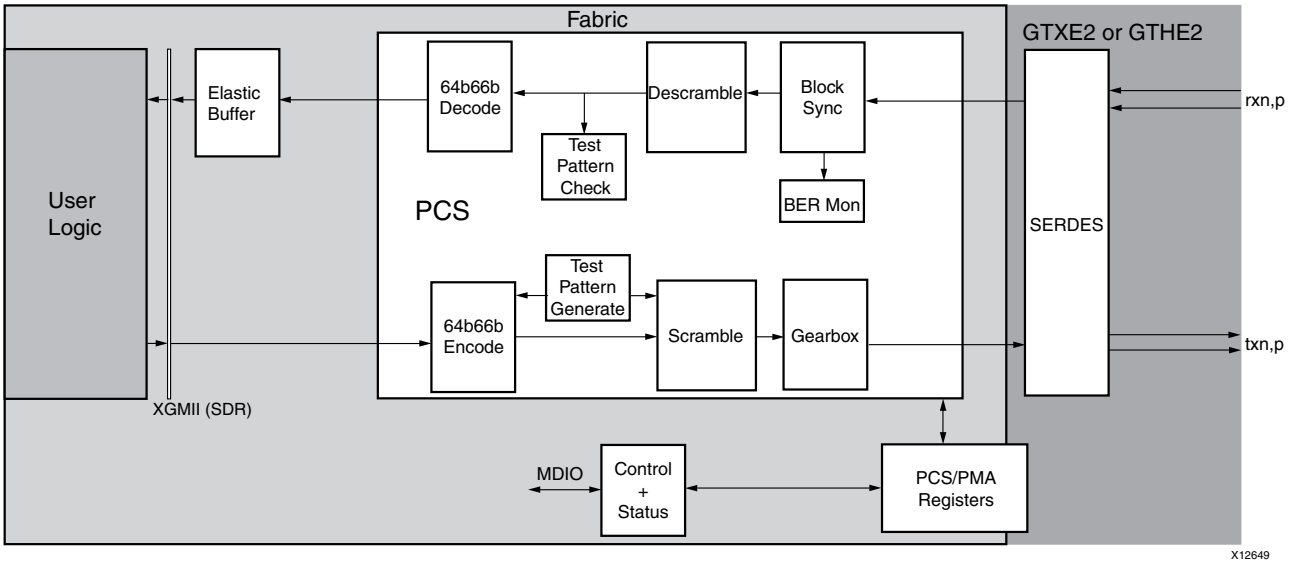


*Figure 3-1:* **Virtex-7/Kintex-7 Implementation of the 10-Gigabit Ethernet PCS/PMA (BASE-R) Core**

Figure 3-2 shows a block diagram of the Virtex7/Kintex-7 10GBASE-KR core. The major functional blocks of the core include the following:

- **Virtex-7/Kintex-7 FPGA GTX/GTH Transceiver**. Provides high-speed transceiver and partial gearbox functionality.
- **Training Block**. Provides backplane training functionality.
- **Optional AN Block**. Provides autonegotiation functionality.
- **Optional FEC Block**. Provides Forward Error Correction (FEC) functionality.
- **PCS Block**. Provides encode/decode, scramble/descramble, block-lock, transmit and receive state machines, test-pattern blocks and BER monitor, in the same configuration as in Figure 3-3.
- **Optional MDIO Interface**. A two-wire low-speed serial interface used to manage the core. An alternative vector-based interface might be provided instead.
- **Elastic Buffer**. Identical to that described in the next section. See page 15.

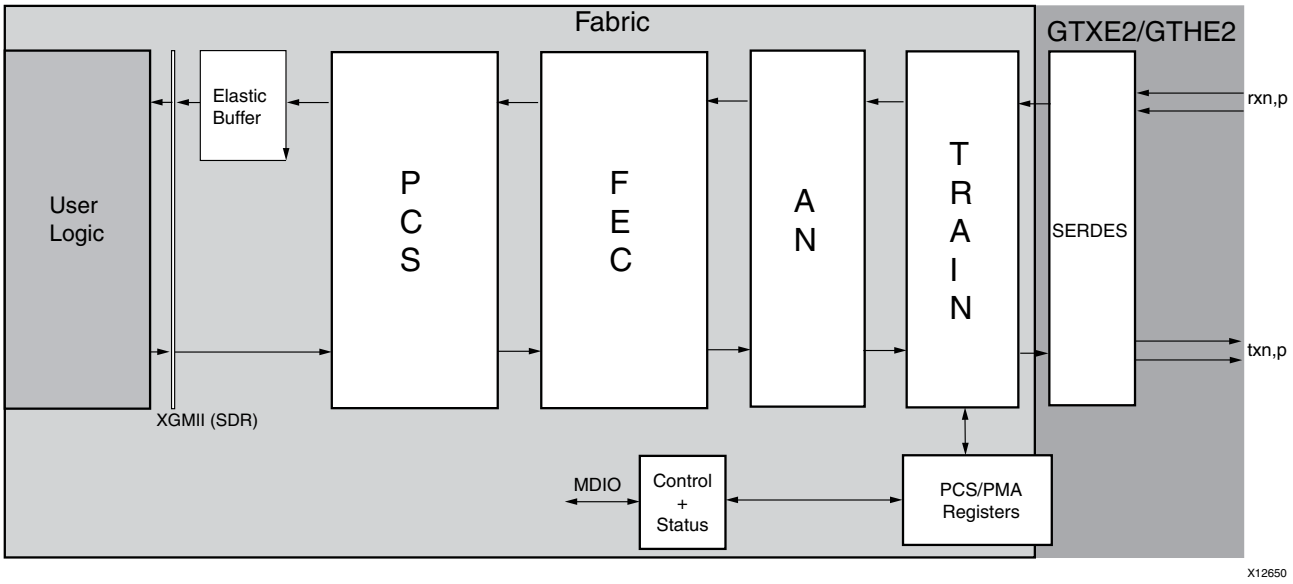


*Figure 3-2:* **Virtex-7/Kintex-7 implementation of the 10-Gigabit Ethernet PCS/PMA (BASE-KR) Core**

Figure 3-3 shows a block diagram of the implementation of the Virtex-6 FPGA 10GBASE-R core. The major functional blocks of the core include the following:

- **Virtex-6 FPGA GTH transceiver.** Provides high-speed transceiver as well as 64B/66B encode and decode, Block Lock, TX and RX state machines and BER monitor.
- **Management Interface.** Provides a simple interface to the management registers in the transceiver.
- **Optional MDIO interface.** A two-wire low-speed serial interface used to manage the core. An alternative vector-based interface might be provided instead.
- **Elastic Buffer** in the receive datapath.
  The Elastic Buffer is 32 words deep (1 word = 32 bits data + 4 control).

  If the buffer empties, Local Fault codes are inserted instead of data.

  This allows you to collect up to 32 clock correction (CC) sequences before the buffer overflows (and words are dropped). The buffer normally fills up to one quarter and only drop CC sequences when over half full, and only insert CC sequences when under one quarter full.

  So from a half-full state, you can (conservatively) accept an extra 14, 32-bit sequences (that is, receiving at +200ppm) without dropping any and from a quarter-full state you can cope with half that number of missing bits without inserting Local faults (for -200ppm).

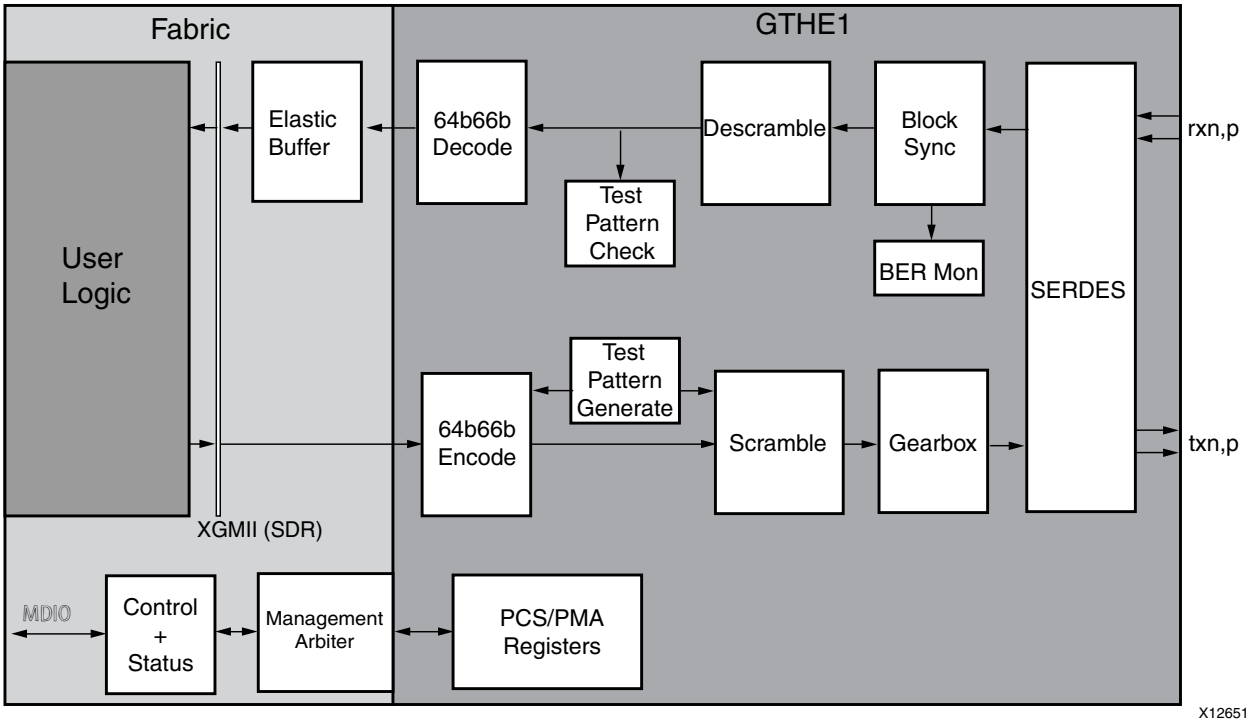


*Figure 3-3:* **Virtex-6 Implementation of the 10-Gigabit Ethernet PCS/PMA (BASE-R) Core**

# Applications

Figure 3-4 shows a typical Ethernet system architecture and the 10-Gigabit Ethernet PCS/PMA core within it. The MAC and all the blocks to the right are defined in Ethernet IEEE specifications.

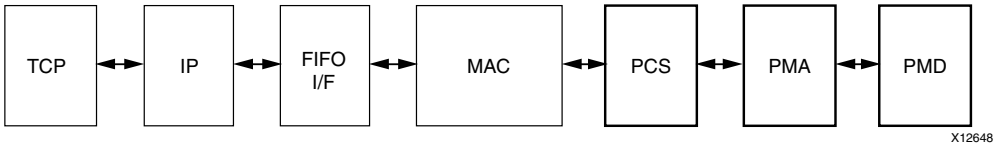


*Figure 3-4:* **Typical Ethernet System Architecture**

Figure 3-5 shows the 10-Gigabit Ethernet PCS/PMA core connected on one side to a 10-Gigabit MAC and on the other to an optical module using a serial interface.

The 10-Gigabit Ethernet PCS/PMA core is designed to be attached to the Xilinx® IP 10-Gigabit Ethernet MAC core over XGMII. More details are provided in Chapter 8, Design Considerations.

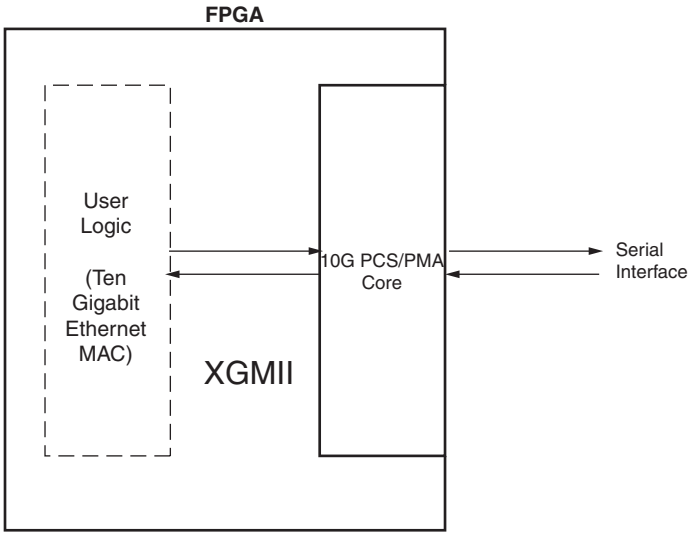


*Figure 3-5:* **10-Gigabit Ethernet PCS/PMA Core Connected to MAC Core Using XGMII Interface**

# Core Interfaces and Modules

## Client-Side Interface

The signals of the client-side interface are shown in Table 3-1. See Chapter 6, Interfacing to the Core for details on connecting to the client-side interface.

*Table 3-1:* **Client-Side Interface Ports**

| Signal Name | Direction | Description |
|---|---|---|
| xgmii_txd[63:0] | IN | Transmit data, eight bytes wide |
| xgmii_txc[7:0] | IN | Transmit control bits, one bit per transmit data byte |
| xgmii_rxd[63:0] | OUT | Received data, eight bytes wide |
| xgmii_rxc[7:0] | OUT | Receive control bits, one bit per received data byte |

## Transceiver Data Interface - Virtex-7/Kintex-7 FPGA GTX/GTH Transceiver

The interface to the device-specific transceivers is not a simple one-to-one interface on those pins that need to be connected. The signals are described in Table 3-2. See Chapter 6, Interfacing to the Core for details on connecting the device-specific transceivers to the 10GBASE-R/KR core. The `gt_txc[7:2]` on the core should be connected to `txsequence[5:0]` on the transceiver and `gt_rxc[2]` and `gt_rxc[3]` on the core should be connected to `rxdatavalid` and `rxheadervalid` on the transceiver.

*Table 3-2:* **Transceiver Interface Ports - Virtex-7/Kintex-7 FPGA GTX/GTH Transceiver**

| Signal Name | Direction | Description |
|---|---|---|
| gt_txd[31:0] | Out | 32-bit transmit data word |
| gt_txc[1:0] | Out | 2-bit transmit sync header |
| gt_txc[7:2] | Out | 6-bit TXSEQUENCE count (0..32) |
| gt_rxd[31:0] | In | 32-bit receive data word |
| gt_rxc[1:0] | In | 2-bit receive sync header |
| gt_rxc[2] | In | RXDATAVALID (high for 64 in 66 rxusrclk2 cycles) |
| gt_rxc[3] | In | RXHEADERVALID (high on alternating cycles of rxusrclk2, while RXDATAVALID is also high) |
| gt_rxc[7:4] | In | Not Used |
| gt_slip | Out | RXGEARBOXSLIP |

## Transceiver Data Interface - Virtex-6 FPGA GTH Transceiver

The interface to the device-specific transceivers is a simple pin-to-pin interface on those pins that need to be connected. The signals are described in Table 3-3. See Chapter 6, Interfacing to the Core for details on connecting the device-specific transceivers to the 10GBASE-R/KR core.

*Table 3-3:* **Transceiver Interface Ports - Virtex-6 FPGA GTH Transceiver**

| Signal Name | Direction | Description |
|---|---|---|
| gt_txd[63:0] | OUT | Transceiver transmit data |
| gt_txc[7:0] | OUT | Transceiver transmit control flag |
| gt_rxd[63:0] | IN | Transceiver receive data |
| gt_rxc[7:0] | IN | Transceiver receive control signals |

## Optical Module Interface

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 3-3. See Chapter 6, Interfacing to the Core for details on connecting an optical module to the 10GBASE-R core.

*Table 3-4:* **Optical Module Interface Ports**

| Signal Name | Direction | Description |
|---|---|---|
| signal_detect | IN | Status signal from attached optical module[a] |
| tx_fault | IN | Status signal from attached optical module[ab] |
| tx_disable | OUT | Control signal to attached optical module |

a. These signals are not connected inside this version of the core. It is left to users to handle these inputs and reset their design as they see fit.

b. Connect to `SFP+ tx_fault` signal, or `XFP MOD_NR` signal, depending on which is present.

## MDIO Interface

The Management Data Input/Output (MDIO) Interface signals are shown in Table 3-5. More information on using this interface can be found in Chapter 6, Interfacing to the Core.

*Table 3-5:* **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; '1' disconnects the output driver from the MDIO bus. |
| prtad[4:0] | IN | MDIO port address; this should be set by you to provide a unique ID on the MDIO bus. |

## Configuration and Status Signals

The Configuration and Status Signals are shown in Table 3-6. See Configuration and Status Vectors, page 91 for details on these signals, including a breakdown of the configuration and status vectors.

*Table 3-6:* **Configuration and Status Ports**

| Signal Name | Direction | Description |
|---|---|---|
| configuration_ vector[535:0] | IN | Configuration information for the core. |
| status_vector[447:0] | OUT | Status information from the core. |

## Clocking and Reset Signals - Virtex-7/Kintex-7 FPGAs

Included in the example design top-level sources are circuits for clock and reset management. These can include clock generators, reset synchronizers, or other useful utility circuits that can be useful in your particular application.

Table 3-7 shows the ports on the netlist that are associated with system clocks and resets.

*Table 3-7:* **Clock and Reset Ports- Virtex-7/Kintex-7**

| Signal Name | Direction | Description |
|---|---|---|
| clk156 | IN | System clock for core |
| rxusrclk2 | IN | Receive path clock, derived from recovered clock on the GTX/GTH transceiver |
| txusrclk2 | IN | Transmit path clock, derived from TXCLKOUT on the GTX/GTH transceiver |
| dclk | IN | Management/DRP clock, at half the rate of clk156 |
| reset | IN | Synchronous reset in clk156 domain |
| rxreset322 | IN | Synchronous reset in rxusrclk2 domain |
| txreset322 | IN | Synchronous reset in txusrclk2 domain |
| dclk_reset | IN | Synchronous reset in dclk domain |
| pma_resetout | OUT | Reset signal from core to transceiver |
| pcs_resetout | OUT | Reset signal from core to transceiver |
| resetdone | IN | Signal from transceiver to core - the requested reset is complete |

## Clocking and Reset Signals - Virtex-6 FPGAs

Included in the example design top-level sources are circuits for clock and reset management. These can include clock generators, reset synchronizers, or other useful utility circuits that can be useful in your particular application.

Table 3-8 shows the ports on the netlist that are associated with system clocks and resets.

*Table 3-8:* **Clock and Reset Ports - Virtex-6 FPGAs**

| Signal Name | Direction | Description |
|---|---|---|
| clk156 | IN | System clock for core. |
| rxclk156 | IN | Receiver clock to transceiver side of elastic buffer. |
| dclk | IN | Management clock used to access transceiver registers. |
| reset | IN | Reset port synchronous to clk156. |

## Transceiver Management Interface - Virtex-6 FPGAs

As shown in the example design block-level sources, the core communicates with the transceiver through a fixed management interface, through a management interface arbiter that allows up to four cores to share access to a single GTH_QUAD component.

Table 3-9 shows the ports on the netlist that are associated with the transceiver management interface.

*Table 3-9:* **Transceiver Management Interface Ports - Virtex-6 FPGAs**

| Signal Name | Direction | Description |
|---|---|---|
| mgmt_req | OUT | Request access to management interface arbiter |
| mgmt_gnt | IN | Access granted to management interface arbiter |
| mgmt_rd_out | OUT | Read pulse to management interface |
| mgmt_wr_out | OUT | Write pulse to management interface |
| mgmt_addr_out [20:0] | OUT | Address for management interface, with the 5-bit DEVAD (Device Address) at bits 20..16. |
| mgmt_rdack_in | IN | Read Acknowledge/Data Valid signal from the transceiver management interface |
| mgmt_rddata_in [15:0] | IN | Read data from management interface |
| mgmt_wrdata_out [15:0] | OUT | Write data to management interface |

## Transceiver DRP Interface - Virtex-7/Kintex-7 FPGAs

In the 7 series devices, the core can communicate with the GTX/GTH transceiver using the Dynamic Reconfiguration Port (DRP) interface. Table 3-10 shows the ports on the netlist that are associated with that interface.

*Table 3-10:* **Transceiver DRP Interface Ports - Virtex-7/Kintex-7 FPGAs**

| Signal Name | Direction | Description |
|---|---|---|
| drp_req | out | Request access to the DRP Interface, in case there is an external arbiter |
| drp_gnt | in | Access Granted to DRP Interface by external arbiter |
| drp_den | out | DRP Enable |
| drp_dwe | out | DRP Write Enable |
| drp_daddr [15:0] | out | DRP Address |
| drp_di [15:0] | out | DRP Write Data |
| drp_drdy | in | DRP Data Ready |
| drp_drpdo [15:0] | in | DRP Read Data |

## Training Interface - Virtex-7/Kintex-7 FPGAs, BASE-KR Only

In the 7 series devices, an external Training Algorithm must be connected to the Training Interface, which allows access to both the 802.3 registers in the core and the DRP registers in the GTX/GTH transceiver. Table 3-11 shows the ports on the netlist that are associated with that interface.

*Table 3-11:* **Training Interface Ports - Virtex-7/Kintex-7 FPGAs, BASE-KR Only**

| Signal Name | Direction | Description |
|---|---|---|
| training_enable | in | Signal from external Training Algorithm to enable the training interface |
| 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 GTX/GTH transceiver |
| training_rddata[15:0] | out | Read data from DRP or 802.3 registers |
| training_rdack | out | Read Acknowledge signal to external Training Algorithm |
| training_wrack | out | Write Acknowledge signal to external Training Algorithm |

1. This signal has no meaning or effect when the core is created without an MDIO interface. This should be tied to '0' in that case. The rest of the Training interface is unaffected.

# Miscellaneous Signals - Virtex-7/Kintex-7 FPGAs

*Table 3-12:* **Miscellaneous Signals**

| Signal Name | Direction | Description |
|---|---|---|
| core_status[7:0] | OUT | Bit 0 = PCS Block Lock, Bits [7:4] are reserved<br>BASE-KR cores: FEC Signal OK in bit 1, pmd_signal_detect (Training Done) in bit 2, AN Complete in bit 3. |
| is_eval | OUT | Base-KR only: Constant output which is '1' if this is an Evaluation Licensed core |
| an_enable | IN | Base-KR only: Used to disable Autonegotiation during simulation - normally tie this to '1'. Only for cores with Optional Autonegotiation block |
| tx_prbs31_en | OUT | Used to enable built-in PRBS31 transmission in the transceiver |
| rx_prbs31_en | OUT | Used to enable built-in PRBS31 checking in the transceiver |
| clear_rx_prbs_err_count | OUT | Signal to transceiver to clear the RX PRBS31 error counter. |
| loopback_ctrl [2:0] | OUT | Loopback control from core to transceiver |

# Customizing and Generating the Core

The 10GBASE-R/KR core is generated using the Xilinx® CORE Generator™ system. This chapter describes how to customize the 10GBASE-R/KR core to your requirements and then generate the core netlist.

## Graphical User Interface

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

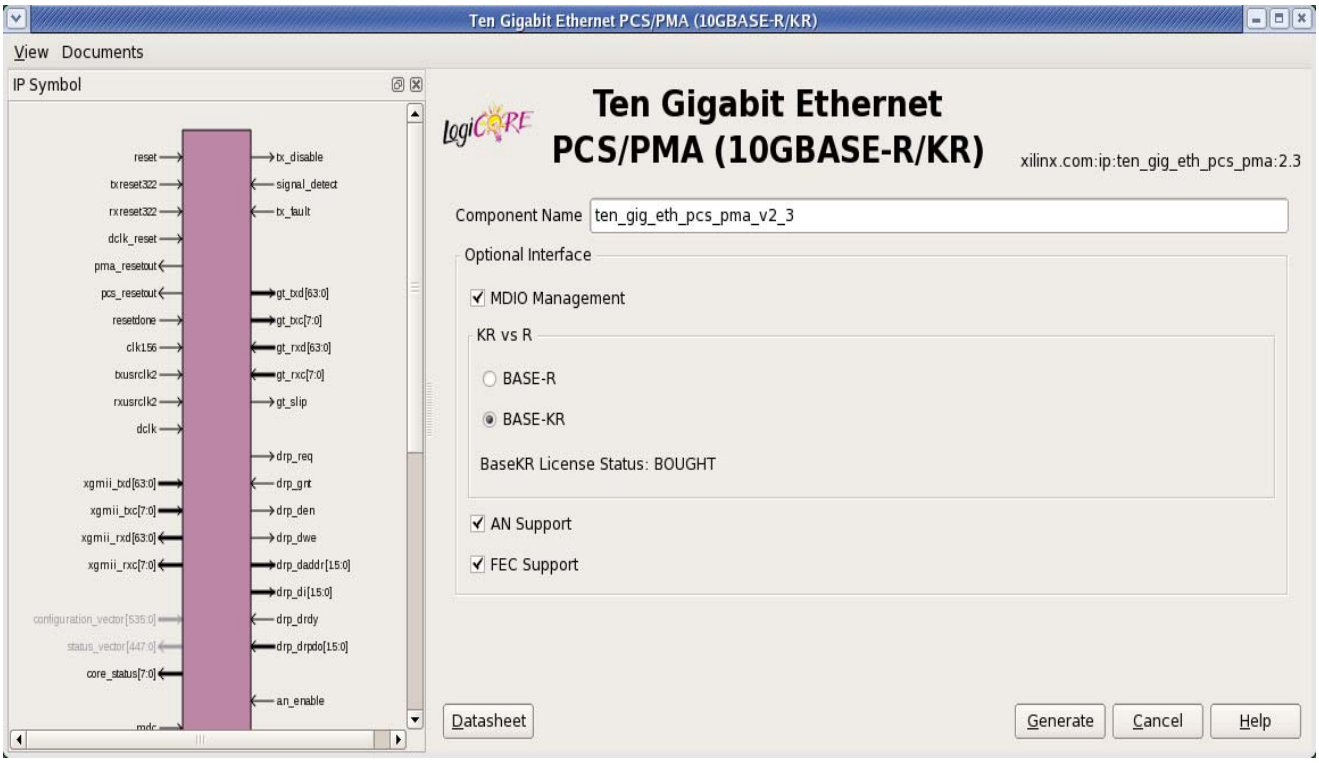


*Figure 4-1:* **10GBASE-R/KR Main Screen**

For general help with starting and using the CORE Generator tool on your development system, see the documentation supplied with the ISE® tools.

### 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).

### 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.

### BASE-R or BASE-KR

Select the Base-KR option to get a Base-KR core and have access to the following two options.

#### AN Support

Select this option to include the Autonegotiation (AN) block in the Base-KR core.

#### FEC Support

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

## Parameter Values in the XCO File

Xilinx CORE Generator core source files (XCO) contain parameterization information for an instance of a core; an XCO file is created when a core is generated and can be used to recreate a core. The text in an XCO file is case-insensitive.

Table 4-1 shows the XCO file parameters and values, and summarizes the Graphical User Interface (GUI) defaults. The following is an example extract from an XCO file:

```
SELECT Ten_Gigabit_Ethernet_PCS/PMA_(10GBASE-R/KR) family Xilinx,_Inc.
2.3
CSET component_name = the_core
CSET mdio_management = true
GENERATE
```

*Table 4-1:* **XCO File Values and Defaults**

| Parameter | XCO File Values | Defaults |
|---|---|---|
| component_name | ASCII text starting with a letter and based upon the following character set: a...z, 0...9 and _ | ten_gig_eth_pcs_pma_v2_3 |
| mdio_management | TRUE, FALSE | TRUE |
| base_kr (1) | BASE-R, BASE-KR | BASE-R |
| fec (2) | TRUE, FALSE | FALSE |
| autonegotiation(2) | TRUE, FALSE | FALSE |

1. 7 series FPGAs only
2. Base-KR only

# Output Generation

The output files generated from the CORE Generator tool are placed in the project directory. The list of output files includes:

- The netlist files for the core
- XCO files
- Release notes and documentation
- An Hardware Description Language (HDL) example design
- Scripts to synthesize, implement and simulate the example design.

See Chapter 10, Detailed Example Design, for a complete description of the CORE Generator tool output files and for details of the HDL example design.

XILINX®

Chapter 5

# Designing with the Core

This chapter provides a general description of how to use the 10GBASE-R/KR core in your designs and should be used in conjunction with Chapter 6, Interfacing to the Core, which describes 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.

## Use the Example Design as a Starting Point

Each instance of the 10GBASE-R/KR core created by the CORE Generator™ 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 the Chapter 10, 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 cannot be 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 UCF provided with the example design for the core identifies the critical signals and the timing constraints that should be applied. See Chapter 7, Constraining the Core for further information.

## Use Supported Design Flows

The core is synthesized in the CORE Generator tool and is delivered to you as an NGC netlist. The example implementation scripts provided currently use Xilinx Synthesis Technology (XST) as the synthesis tool for the HDL example design that is delivered with the core. Other synthesis tools can be used for your application logic; the core is always unknown to the synthesis tool and appears as a black box.

Post synthesis, only Xilinx® ISE® v14.1 tools are supported.

## 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 compliance. Supported user configurations of the 10GBASE-R/KR core can only be made by selecting the options from within the CORE Generator tool when the core is generated. See Chapter 4, Customizing and Generating the Core.

XILINX®

Chapter 6

# Interfacing to the Core

This chapter describes how to connect to the data interfaces of the core and configuration and status interfaces of the 10GBASE-R/KR core.

## Data Interface: Internal Interfaces

### Internal 64-bit SDR Client-side Interface

The 64-bit single-data rate (SDR) client-side interface is based upon the 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 6-1. 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 6-1:* **XGMII_TXD, XGMII_RXD Lanes for Internal 64-bit Client-Side 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 |

## Definitions of Control Characters

Reference is regularly made to certain XGMII control characters signifying Start, Terminate, Error, and so forth. These control characters all have in common that the control line for that lane is '1' for the character and a certain data byte value. The relevant characters are defined in the *IEEE Std. 802.3-2008* and are reproduced in Table 6-2 for reference.

*Table 6-2:* **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) |

# Interfacing to the Transmit Client Interface

## Internal 64-bit Client-Side Interface

The timing of a data frame transmission through the internal 64-bit client-side interface is shown in Figure 6-1. The beginning of the data frame is shown by the presence of the Start character (the /S/ codegroup in lane 4 of Figure 6-1) 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 6-1). The Terminate character can occur in any lane; the remaining lanes are padded by XGMII idle characters.

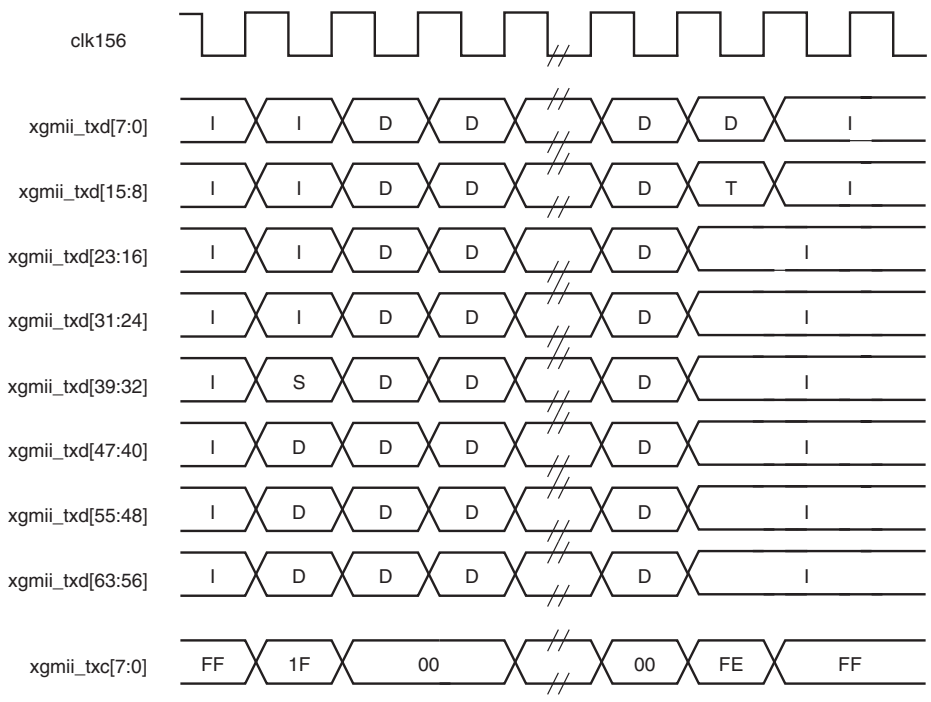


*Figure 6-1:* **Normal Frame Transmission Across the Internal 64-bit Client-Side I/F**

Figure 6-2 depicts a similar frame to that in Figure 6-1, 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.

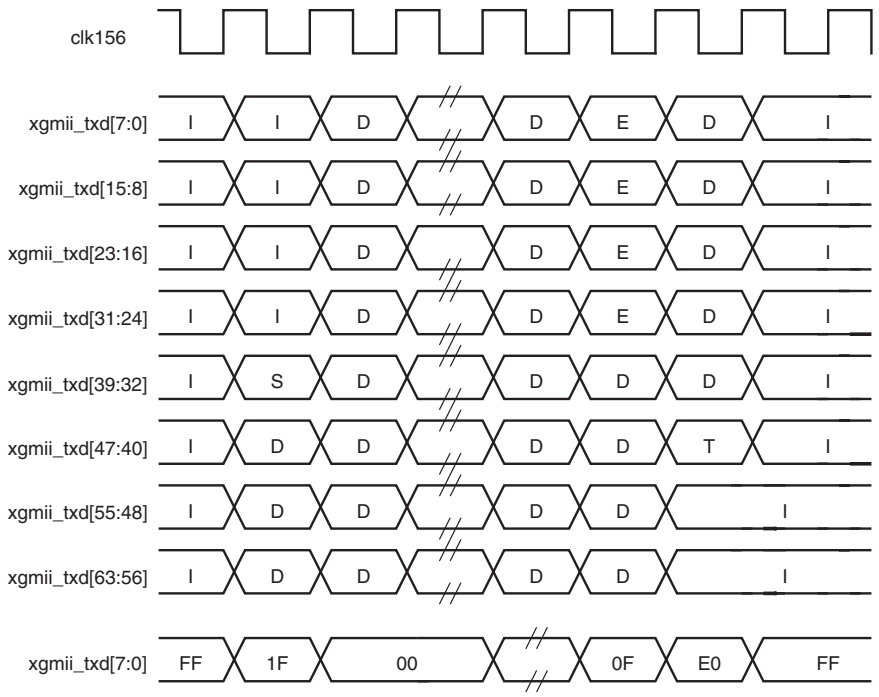


*Figure 6-2:* **Frame Transmission with Error Across Internal 64-bit Client-Side I/F**

# Interfacing to the Receive Client Interface

## Internal 64-bit Client-Side Interface

The timing of a normal inbound frame transfer is shown in Figure 6-3. 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.

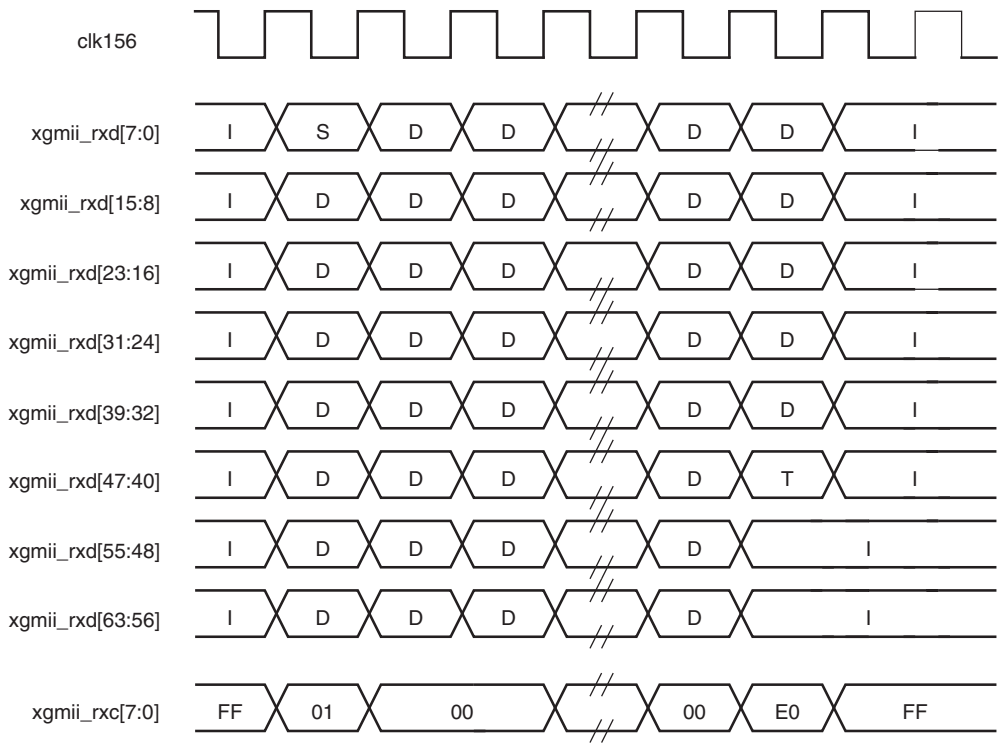


*Figure 6-3:* **Frame Reception Across the Internal 64-bit Client Interface**

Figure 6-4 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.

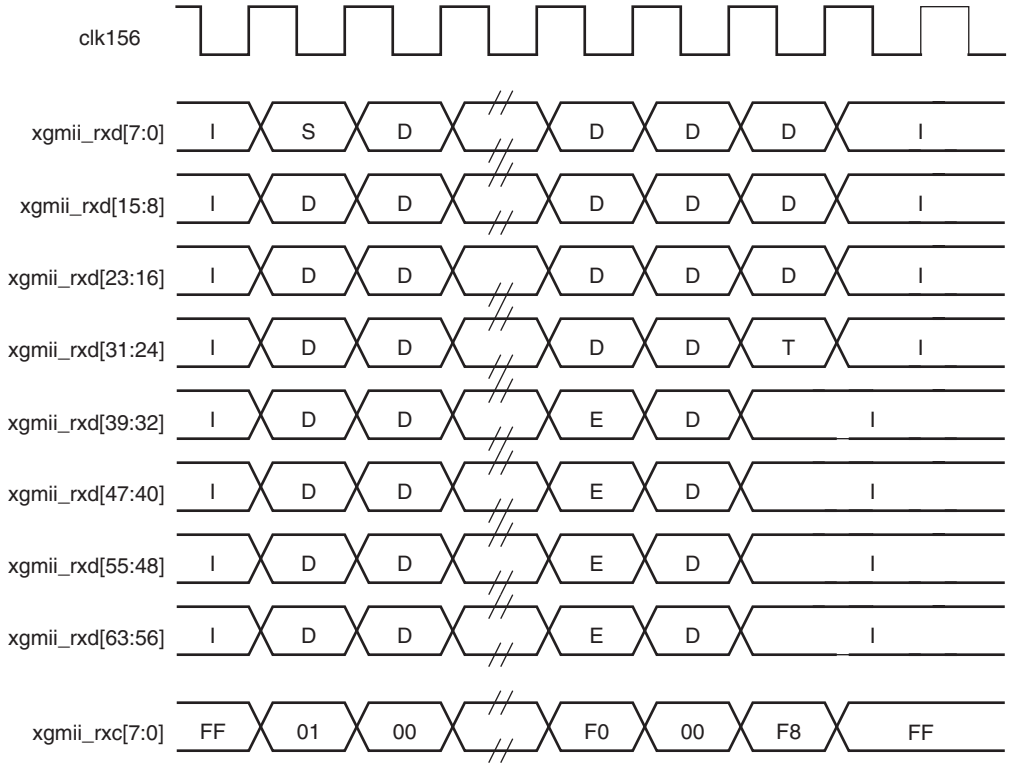


*Figure 6-4:* **Frame Reception with Error Across the Internal 64-bit Client Interface**

# Interfacing to the Transceivers

## Virtex-7/Kintex-7 FPGAs

*Table 6-3:* **Transceiver Interface Ports for Virtex-7/Kintex-7 FPGA GTX/GTH Transceivers**

| Signal Name | Direction | Description |
|---|---|---|
| gt_txd[31:0] | Out | 64-bit transmit data word |
| gt_txc[1:0] | Out | 2-bit transmit sync header |
| gt_txc[7:2] | Out | 6-bit TXSEQUENCE count (0..32) |
| gt_rxd[31:0] | In | 64-bit receive data word |
| gt_rxc[1:0] | In | 2-bit receive sync header |
| gt_rxc[2] | In | RXDATAVALID (high for 64 in 66 rxusrclk2 cycles) |
| gt_rxc[3] | In | RXHEADERVALID (high on alternate cycles of rxusrclk2, when RXDATAVALID is high) |
| gt_rxc[7:4] | In | Not Used |
| gt_slip | In | RXGEARBOXSLIP on transceiver |
| tx_prbs31_en | Out | Transmit PRBS31 pattern enable |
| rx_prbs31_en | Out | Enable Receive PRBS31 pattern checking |
| clear_rx_prbs_ err_count | Out | Signal to transceiver to clear the RX PRBS31 error counter. |
| loopback_ctl[2:0] | Out | Control for transceiver internal loopback |
| resetdone | In | Signal from the transceiver wrapper logic |
| drp_req | Out | Request access to the transceiver DRP port (not connected in example design) |
| drp_gnt | In | Access to transceiver DRP port is granted (not connected in example design) |
| drp_den | Out | Enable DRP |
| drp_dwe | Out | DRP Write Enable |
| drp_daddr[15:0] | Out | DRP address |
| drp_di[15:0] | Out | DRP Write data |
| drp_drdy | In | DRP Data Ready |
| drp_drpdo[15:0] | In | DRP Read data |

The remainder of the device-specific transceiver ports are not connected to the netlist, but are connected in the core source code (block-level and transceiver wrapper files) or are wired to static values.

No timing diagrams are presented here for the device-specific transceiver signals. You should treat this interface as a black box. If customization of this interface is required, see the *7 Series Transceiver User Guide (UG476)* for detailed descriptions of the transceiver ports.

### Virtex-6 HXT FPGAs

*Table 6-4:* **Transceiver Interface Ports for Virtex-6 FPGA GTH Transceivers**

| Signal Name | Direction | Description |
|---|---|---|
| gt_txd[63:0] | OUT | Transceiver transmit data |
| gt_txc[7:0] | OUT | Transceiver transmit control signals |
| gt_rxd[63:0] | IN | Transceiver receive data |
| gt_rxc[7:0] | IN | Transceiver receive control signals |

The remainder of the device-specific transceiver ports are not connected to the netlist, but are connected in the core source code (block-level and transceiver wrapper files) or are wired to static values.

No timing diagrams are presented here for the device-specific transceiver signals. You should treat this interface as a black box. If customization of this interface is required, see the *Virtex-6 FPGA GTH Transceivers User Guide* for detailed descriptions of the transceiver ports.

The following sections describe the interfaces available for dynamically setting the configuration and obtaining the status of the 10GBASE-R/KR core. There are two interfaces for configuration; depending on the core customization, only one is available in a particular core instance.

## Configuration and Status Interfaces

This section describes the interfaces available for dynamically setting the configuration and obtaining the status of the 10GBASE-R/KR core. There are two interfaces for configuration; depending on the core customization, only one is available in a particular core instance. The interfaces are:

- MDIO Interface, page 38
- Configuration and Status Vectors, page 91

## 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 Standard 802.3-2008*.

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 6-5 illustrates a typical system. All transactions are initiated by the STA entity. The 10GBASE-R/KR core implements an MMD.

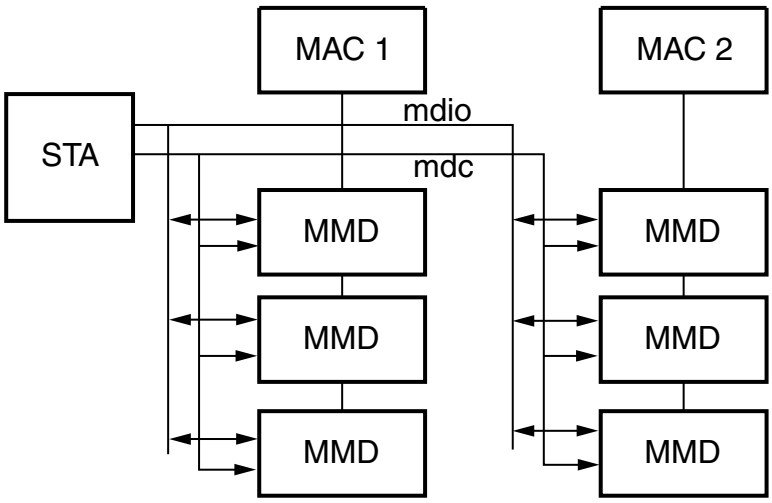


*Figure 6-5:* **A Typical MDIO-Managed System**

## MDIO Ports

The core ports associated with MDIO are shown in Table 6-5.

*Table 6-5:* **MDIO Management Interface Port Description**

| Signal Name | Direction | Description |
|---|---|---|
| mdc | IN | Management clock |
| mdio_in | IN | MDIO input |
| mdio_out | OUT | MDIO output |
| mdio_tri | OUT | MDIO 3-state. '1' disconnects the output driver from the MDIO bus. |
| prtad[4:0] | IN | MDIO port address |

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 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. 802.3-2008*. 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

#### Virtex-6 FPGAs

The device address in this case will be either “00001” for the PMA device or “00011” for the PCS device, both of which are implemented in the GTH Transceiver. MDIO transactions with a DEVAD other than those two values are ignored.

#### Virtex-7/Kintex-7 FPGAs

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 Autonegotiation block, a DEVAD of "00111" should be used to access the associated registers.

### Set Address Transaction

Figure 6-6 shows an 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, for subsequent data transactions (called the “current address” in the following sections). The core contains two such address registers, one for PCS and one for PMA.

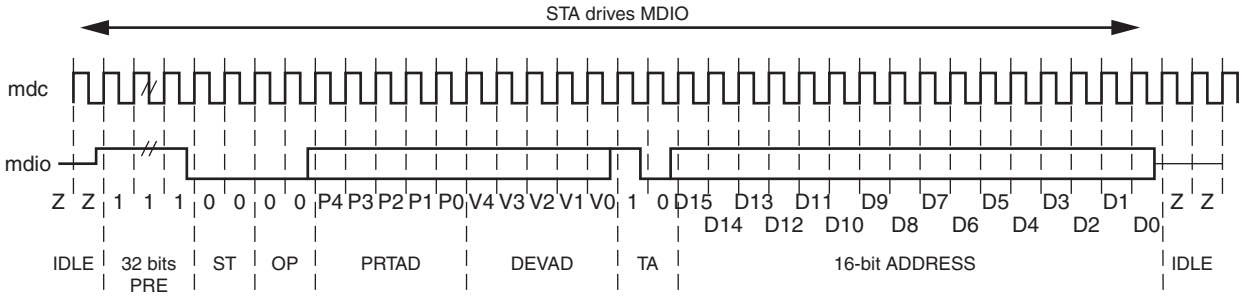


*Figure 6-6:* **MDIO Set Address Transaction**

## Write Transaction

Figure 6-7 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.

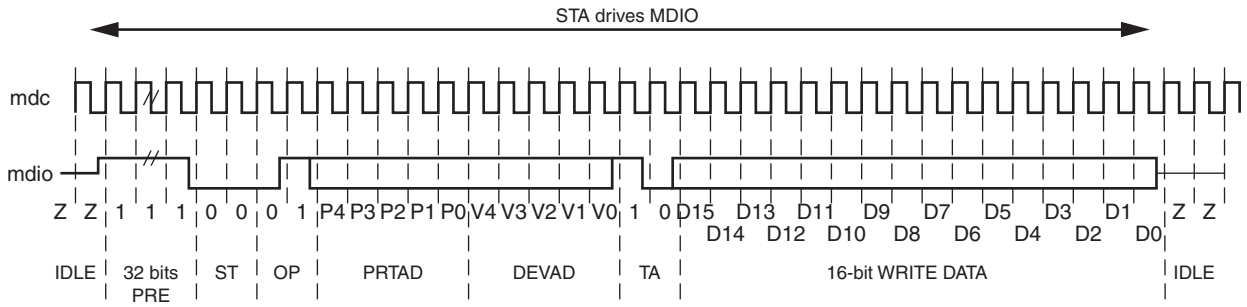


*Figure 6-7:* **MDIO Write Transaction**

## Read Transaction

Figure 6-8 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.

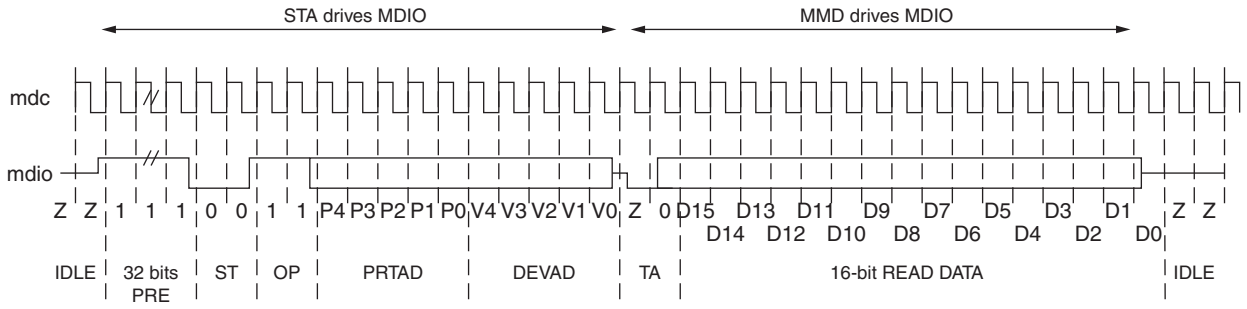


*Figure 6-8:* **MDIO Read Transaction**

## Post-Read-increment-address Transaction

Figure 6-9 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.

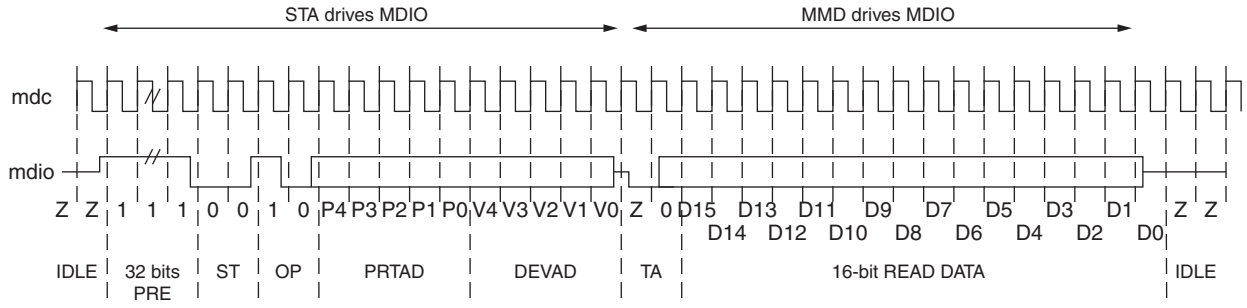


*Figure 6-9:* **MDIO Read-and-increment Transaction**

## 10GBASE-R PCS/PMA Register Map

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 6-6.

*Table 6-6:* **10GBASE-R/KR PCS/PMA MDIO Registers**

| Register Address | Register Name |
|---|---|
| 1.0 | PMA/PMD Control 1 |
| 1.1 | PMA/PMD Status 1 |
| 1.4 | PMA/PMD Speed Ability |
| 1.5, 1.6 | PMA/PMD Devices in Package |
| 1.7 | 10G PMA/PMD Control 2 |
| 1.8 | 10G PMA/PMD Status 2 |
| 1.9 | 10G PMD Transmit Disable |
| 1.10 | 10G PMD Receive Signal OK |
| 1.11 to 1.32787 | Reserved |
| 1.32788 | PMA Vendor Specific Loopback (Virtex®-6 FPGAs only) |
| 1.32789 to 1.65534 | Reserved |
| 1.65535 | Core Version Info |
| 3.0 | PCS Control 1 |
| 3.1 | PCS Status 1 |
| 3.4 | PCS Speed Ability |
| 3.5, 3.6 | PCS Devices in Package |
| 3.7 | 10G PCS Control 2 |
| 3.8 | 10G PCS Status 2 |
| 3.9 to 3.31 | Reserved |
| 3.32 | 10GBASE-R/KR PCS Status 1 |
| 3.33 | 10GBASE-R/KR PCS Status 2 |
| 3.34-37 | 10GBASE-R/KR Test Pattern Seed A |
| 3.38-41 | 10GBASE-R/KR Test Pattern Seed B |
| 3.42 | 10GBASE-R/KR Test Pattern Control |
| 3.43 | 10GBASE-R/KR Test Pattern Error Counter |
| 3.44 to 3.65534 | Reserved |
| 3.32769 to 3.65534 | Reserved |
| 3.65535 | PCS 125 µs timer control |

## 10GBASE-KR PCS/PMA Register Map

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 6-7.

*Table 6-7:* **10GBASE-KR PCS/PMA Registers**

| Register Address | Register Name |
|---|---|
| 1.0 | PMA/PMD Control 1 |
| 1.1 | PMA/PMD Status 1 |
| 1.150 | 10GBASE-KR PMD control |
| 1.151 | 10GBASE-KR PMD status |
| 1.152 | 10GBASE-KR LP coefficient update |
| 1.153 | 10GBASE-KR LP status report |
| 1.154 | 10GBASE-KR LD coefficient update |
| 1.155 | 10GBASE-KR LD status report |
| 1.170 | 10GBASE-R FEC ability (1) |
| 1.171 | 10GBASE-R FEC control (1) |
| 1.172 to 1.173 | 10GBASE-R FEC corrected blocks counter (1) |
| 1.174 to 1.175 | 10GBASE-R FEC uncorrected blocks counter (1) |
| 1.4 | PMA/PMD Speed Ability |
| 1.5, 1.6 | PMA/PMD Devices in Package |
| 1.7 | 10G PMA/PMD Control 2 |
| 1.8 | 10G PMA/PMD Status 2 |
| 1.9 | 10G PMD Transmit Disable |
| 1.10 | 10G PMD Receive Signal OK |
| 1.11 to 1.149 | Reserved |
| 1.176 to 1.65519 | Reserved |
| 1.65520 | 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 | Core Version Info (Virtex-7/Kintex™-7 FPGAs only) |
| 3.0 | PCS Control 1 |
| 3.1 | PCS Status 1 |
| 3.4 | PCS Speed Ability (Virtex-7/Kintex-7 FPGAs only) |
| 3.5, 3.6 | PCS Devices in Package |
| 3.7 | 10G PCS Control 2 |
| 3.8 | 10G PCS Status 2 |

*Table 6-7:* **10GBASE-KR PCS/PMA Registers *(Cont'd)***

| Register Address | Register Name |
|---|---|
| 3.9 to 3.31 | Reserved |
| 3.32 | 10GBASE-R/KR PCS Status 1 |
| 3.33 | 10GBASE-R/KR PCS Status 2 |
| 3.34-37 | 10GBASE-R/KR Test Pattern Seed A |
| 3.38-41 | 10GBASE-R/KR Test Pattern Seed B |
| 3.42 | 10GBASE-R/KR Test Pattern Control |
| 3.43 | 10GBASE-R/KR Test Pattern Error Counter |
| 3.44 to 3.32767 | Reserved |
| 3.32768 | PCS Vendor Specific Loopback (Virtex-6 FPGAs only) |
| 3.65535 | PCS 125 µs timer control (Virtex-7/Kintex-7 FPGAs only) |
| 7.0 | AN Control (2) |
| 7.1 | AN Status (2) |
| 7.16, 17, 18 | AN Advertisement (2) |
| 7.19, 20, 21 | AN LP Base Page Ability (2) |
| 7.22, 23, 24 | AN XNP transmit (2) |
| 7.25, 26, 27 | AN LP XNP ability (2) |
| 7.48 | Backplane Ethernet Status (2) |

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

## MDIO Register 1.0: PMA/PMD Control 1

Figure 6-10 shows the MDIO Register 1.0: Physical Medium Attachment/Physical Medium Dependent (PMA/PMD) Control 1.

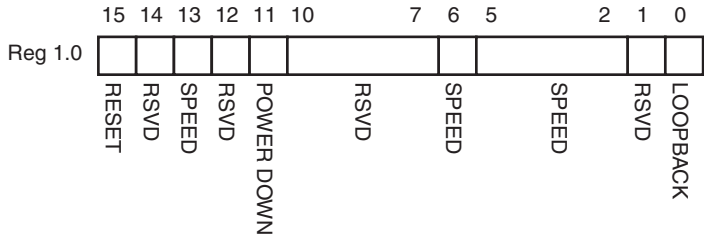


*Figure 6-10:* **PMA/PMD Control 1 Register**

Table 6-8 shows the PMA Control 1 register bit definitions.

*Table 6-8:* **PMA/PMD Control 1 Register Bit Definitions**

| Bit(s) | Name | Description | Attributes | Default Value |
|---|---|---|---|---|
| 1.0.15 | Reset | 1 = Block reset<br>0 = Normal operation<br>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 |
| 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<br>0 = Disable PMA loopback mode<br>Virtex-6 FPGAs: The 10GBASE-R/KR block will loop the transmit signal inside the GTH transceiver back into the receiver.<br>***Note:*** The vendor-specific register bits 1.32788.1:0 take precedence over this bit. | R/W | 0 |

## MDIO Register 1.1: PMA/PMD Status 1

Figure 6-11 shows the MDIO Register 1.1: PMA/PMD Status 1.

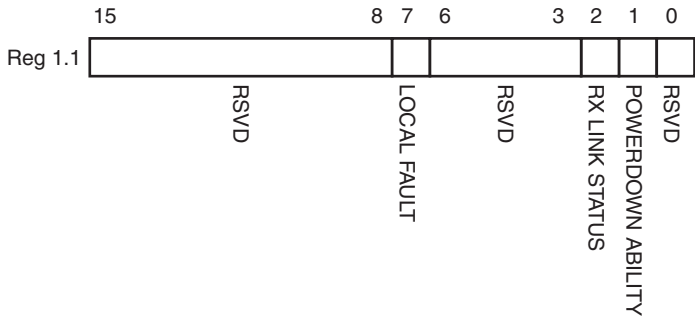


*Figure 6-11:* **PMA/PMD Status 1 Register**

Table 6-9 shows the PMA/PMD Status 1 register bit definitions.

*Table 6-9:* **PMA/PMD Status 1 Register Bit Definitions**

| Bit(s) | 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 | Virtex-6: The block always returns '0' for this bit.<br>Virtex-7/Kintex-7: 1 = Local Fault detected | R/O | 0 |
| 1.1.6:3 | Reserved | The block always returns ‘0’ for this bit. | R/O | 0 |
| 1.1.2 | Receive Link Status | Virtex-6: The block always returns '1' for this bit<br>Virtex-7/Kintex-7: 1 = Receive Link UP | R/O<br>V7/K7: 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 6-12 shows the MDIO Register 1.4: PMA/PMD Speed Ability.

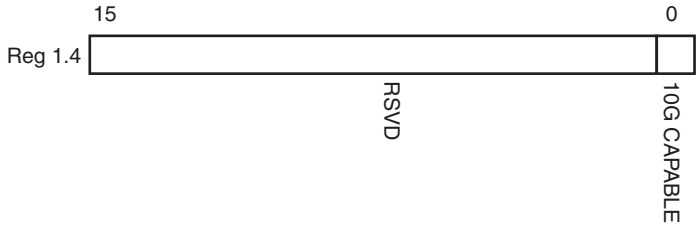


*Figure 6-12:* **PMA/PMD Speed Ability Register**

Table 6-10 shows the PMA/PMD Speed Ability register bit definitions.

*Table 6-10:* **PMA/PMD Speed Ability Register Bit Definitions**

| Bit(s) | 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 6-13 shows the MDIO Registers 1.5 and 1.6: PMA/PMD Devices in Package.

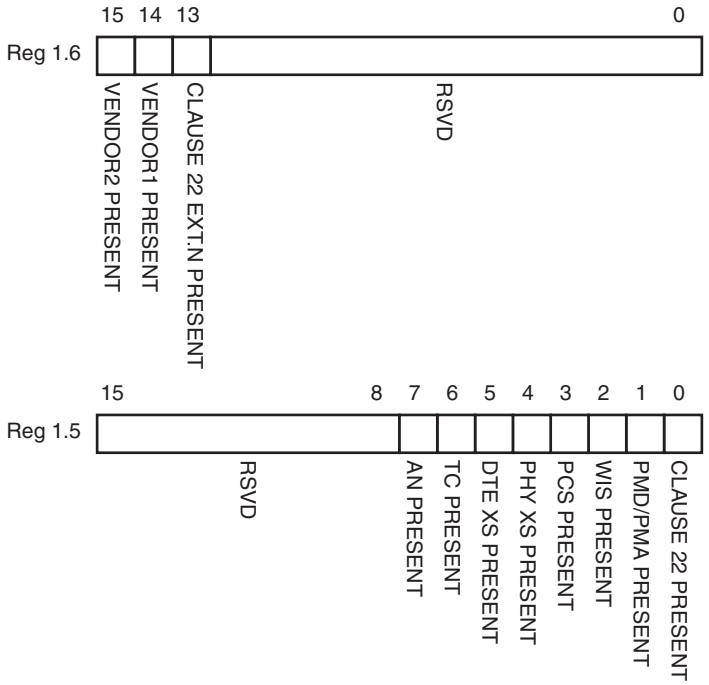


*Figure 6-13:* **PMA/PMD Devices in Package Registers**

Table 6-11 shows the PMA/PMD Device in Package registers bit definitions.

*Table 6-11:* **PMA/PMD Devices in Package Registers Bit Definitions**

| Bit(s) | 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 | Autonegotiation present | Virtex-6: The block always returns '1' for this bit.<br>Virtex-7/Kintex-7: 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 |
| 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 6-14 shows the MDIO Register 1.7: 10G PMA/PMD Control 2.

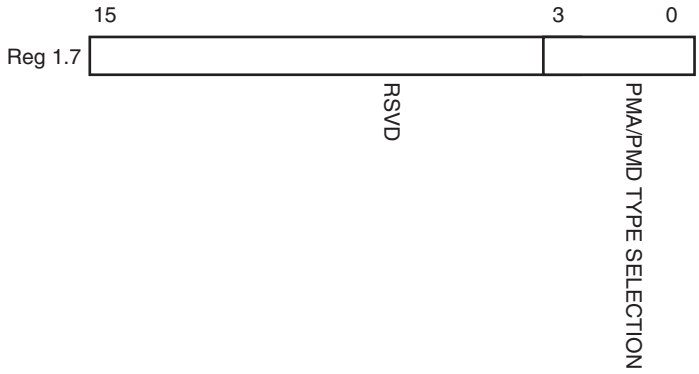


*Figure 6-14:* **10G PMA/PMD Control 2 Register**

Table 6-12 shows the PMA/PMD Control 2 register bit definitions.

*Table 6-12:* **10G PMA/PMD Control 2 Register Bit Definitions**

| Bit(s) | 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 | Virtex-7/Kintex-7 FPGAs:<br>This returns the value '0xyz', where 'xyz' is set from the top level core port pma_pmd_type vector.<br>Virtex-6 FPGAs: The block returns a code for the 10GBASE-*R PMA/PMD and ignores written values which do not correspond to the PCS_ABILITY register settings (1.8.7:1). '0111' denotes 10GBASE-SR, '0110' denotes -LR and '0101' denotes -ER. | R/W | Virtex-7/Kintex-7 FPGAs:<br>Base-R: Set from pma_pmd_type port.<br>BASE-KR: returns 0xB<br>Virtex-6 FPGAs:<br>Set from GTH transceiver attribute. |

## MDIO Register 1.8: 10G PMA/PMD Status 2

Figure 6-15 shows the MDIO Register 1.8: 10G PMA/PMD Status 2.

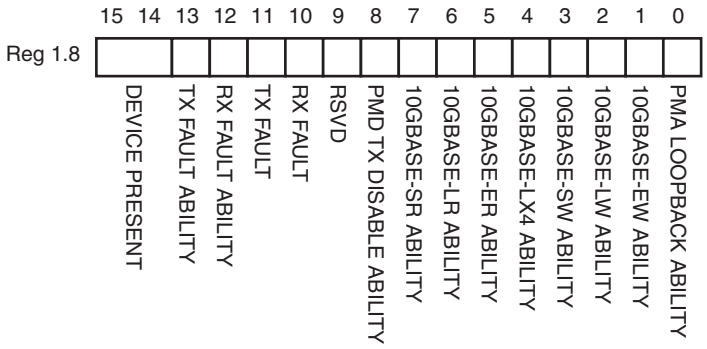


*Figure 6-15:* **10G PMA/PMD Status 2 Register**

Table 6-13 shows the PMA/PMD Status 2 register bit definitions.

*Table 6-13:* **10G PMA/PMD Status 2 Register Bit Definitions**

| Bit(s) | 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 | Virtex-6: The block always returns '0' for this bit.<br>Virtex-7/Kintex-7: The block always returns a '1' for this bit. | R/O | Virtex-6: 0<br>Virtex-7/Kintex-7: 1 |
| 1.8.12 | Receive Local Fault Ability | Virtex-6: The block always returns '0' for this bit<br>Virtex-7/Kintex-7: The block always returns a '1' for this bit. | R/O | Virtex-6: 0<br>Virtex-7/Kintex-7: 1 |
| 1.8.11 | Transmit Fault | Virtex-6: The block always returns '0' for this bit.<br>Virtex-7/Kintex-7: 1 = Transmit Fault detected | R/O<br>V7/K7: Latches High | 0 |
| 1.8.10 | Receive Fault | Virtex-6: The block always returns '0' for this bit.<br>Virtex-7/Kintex-7: 1 = Receive Fault detected | R/O<br>V7/K7: 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 | Virtex-7/Kintex-7 FPGAs:<br>Base-R only: Returns a '1' if pma_pmd_type port is set to '111'<br>Virtex-6 FPGAs:<br>The block always returns ‘1’ for this bit. | R/O | Virtex-7/Kintex-7: Depends on pma_pmd_type port<br>Virtex-6 FPGAs: 1 |

*Table 6-13:* **10G PMA/PMD Status 2 Register Bit Definitions *(Cont'd)***

| Bit(s) | Name | Description | Attributes | Default Value |
|---|---|---|---|---|
| 1.8.6 | 10GBASE-LR Ability | Virtex-7/Kintex-7 FPGAs: Base-R only: Returns a '1' if pma_pld_type port is set to '110'<br>Virtex-6 FPGAs: The block always returns '1' for this bit. | R/O | Virtex-7/Kintex-7: Returns a '1' if pma_pld_type port is set to '110'<br>Virtex-6 FPGAs: 1 |
| 1.8.5 | 10GBASE-ER Ability | Virtex-7/Kintex-7 FPGAs: Base-R only: Returns a '1' if the pma_pmd_type port is set to '101'<br>Virtex-6 FPGAs: The block always returns '1' for this bit. | R/O | Virtex-7/Kintex-7: Depends on pma_pmd_type port<br>Virtex-6 FPGAs: 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 |
| 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 |

## MDIO Register 1.9: 10G PMD Transmit Disable

Figure 6-16 shows the MDIO 1.9 Register: 10G PMD Transmit Disable.

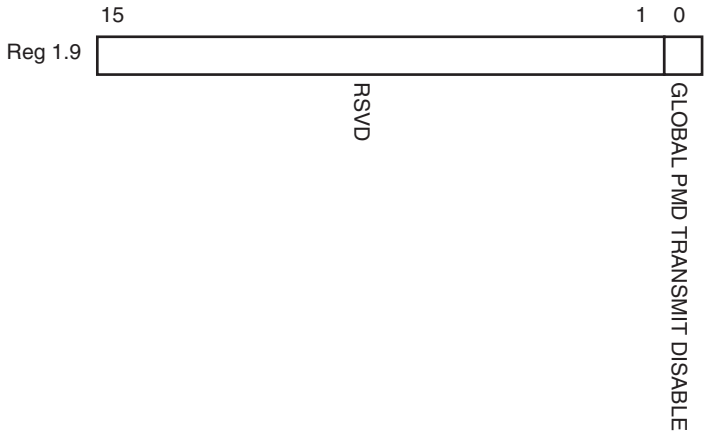


*Figure 6-16:* **10G PMD Transmit Disable Register**

*Table 6-14:* **10G PMD Transmit Disable Register Bit Definitions**

| Bit(s) | 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)<br>0 = Enable Transmit path | Virtex-6: R/W<br>V7/K7: R/W | Virtex-6: Set from GTH attribute.<br>V7/K7: 0 |

## MDIO Register 1.10: 10G PMD Signal Receive OK

Figure 6-17 shows the MDIO 1.10 Register: 10G PMD Signal Receive OK.

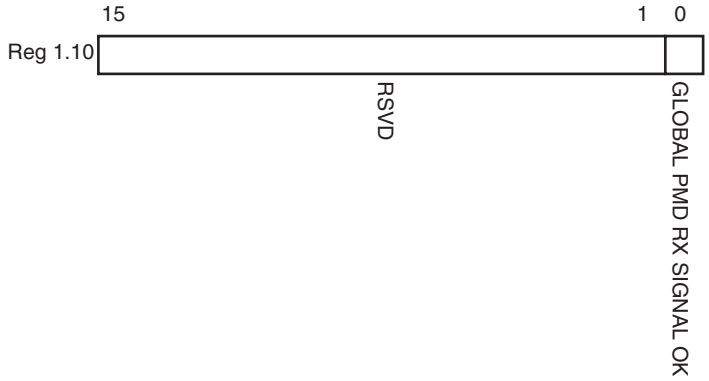


*Figure 6-17:* **10G PMD Signal Receive OK Register**

Table 6-13 shows the PMD Signal Receive OK register bit definitions.

*Table 6-15:* **10G PMD Signal Receive OK Register Bit Definitions**

| Bit(s) | 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<br>0 = Signal not detected on receive | R/O | n/a |

## MDIO Register 1.150: 10GBASE-KR PMD Control

Figure 6-18 shows the MDIO Register 1.150: 10GBASE-KR PMD Control.

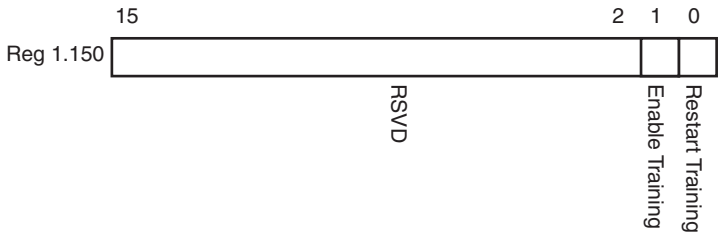


*Figure 6-18:* **10GBASE-KR PMD Control Register**

Table 6-16 shows the 10GBASE-KR PMD Control register bit definitions.

*Table 6-16:* **10GBASE-KR PMD Control Register Bit Definitions**

| Bit(s) | 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<br>0 = Disable | R/W | 0 |
| 1.150.0 | Restart Training | 1 = Reset the 10GBASE-KR start-up protocol<br>0 = Normal operation | R/W Self-clearing | 0 |

## MDIO Register 1.151: 10GBASE-KR PMD Status

Figure 6-19 shows the MDIO Register 1.151: 10GBASE-KR PMD Status.

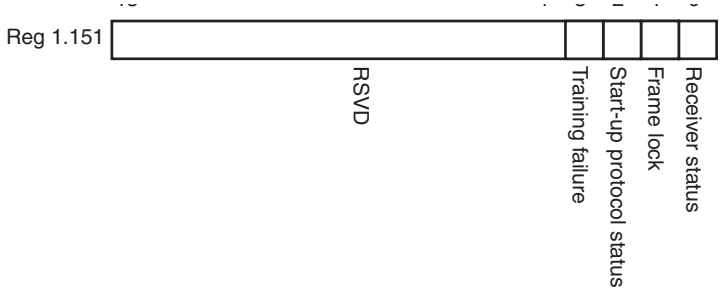


*Figure 6-19:* **10GBASE-KR PMD Status Register**

Table 6-17 shows the 10GBASE-KR PMD Status register bit definitions.

*Table 6-17:* **10GBASE-KR PMD Status Register Bit Definitions**

| Bit(s) | 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<br>0 = Not detected | R/O | 0 |
| 1.151.2 | Start-up Protocol status | 1 = Start-up protocol in progress<br>0 = Protocol complete | R/O | 0 |
| 1.151.1 | Frame Lock | 1 = Training frame delineation detected<br>0 = Not detected | R/O | 0 |
| 1.151.0 | Receiver status | 1 = Receiver trained and ready to receive data<br>0 = Receiver training | R/O | 0 |

## MDIO Register 1.152: 10GBASE-KR LP Coefficient Update

Figure 6-20 shows the MDIO Register 1.152: 10GBASE-KR LP Coefficient Update.

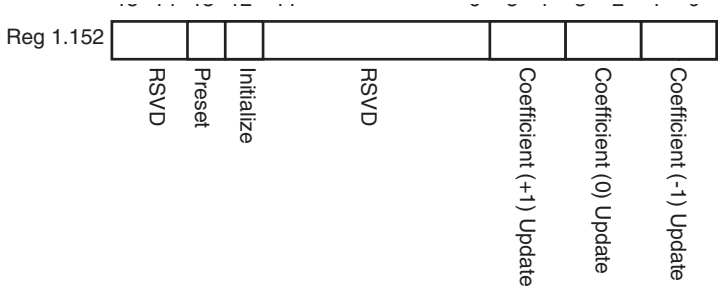


*Figure 6-20:* **10GBASE-KR LP Coefficient Update Register**

Table 6-18 shows the 10GBASE-KR LP coefficient update register bit definitions.

*Table 6-18:* **10GBASE-KR LP Coefficient Update Register Bit Definitions**

| Bit(s) | 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<br>0 = Normal operation | R/W[a] | 0 |
| 1.152.12 | Initialize | 1 = Initialize coefficients<br>0 = Normal operation | R/W[a] | 0 |
| 1.152.11:6 | Reserved | The block always returns '0' for this bit and ignores writes. | R/O | 0's |

*Table 6-18:* **10GBASE-KR LP Coefficient Update Register Bit Definitions *(Cont'd)***

| Bit(s) | Name | Description | Attributes | Default Value |
|---|---|---|---|---|
| 1.152.5:4 | Coefficient (+1) update | 5:4 = 11 = reserved<br>10 = decrement<br>01 = increment<br>00 = hold | R/W[a] | 00 |
| 1.152.3:2 | Coefficient (0) update | 3:2 = 11 = reserved<br>10 = decrement<br>01 = increment<br>00 = hold | R/W[a] | 00 |
| 1.152.1:0 | Coefficient (-1) update | 1:0 = 11 = reserved<br>10 = decrement<br>01 = increment<br>00 = hold | R/W[a] | 00 |

a. Writable only when register 1.150.1 = 0

## MDIO Register 1.153: 10GBASE-KR LP Status

Figure 6-21 shows the MDIO Register 1.153: 10GBASE-KR LP status.

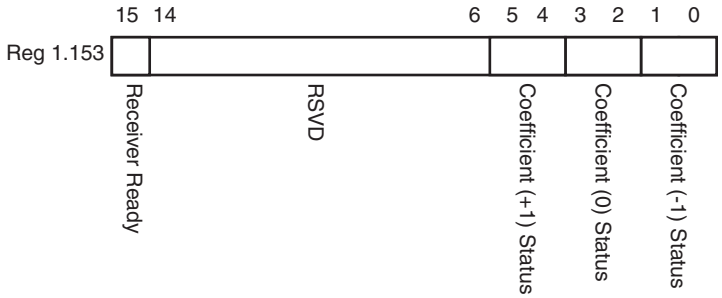


*Figure 6-21:* **10GBASE-KR LP Status Register**

Table 6-19 shows the 10GBASE-KR LP status register bit definitions.

*Table 6-19:* **10GBASE-KR LP Status Register Bit Definitions**

| Bit(s) | Name | Description | Attributes | Default Value |
|---|---|---|---|---|
| 1.153.15:14 | Receiver Ready | 1 = The LP receiver has determined that training is complete and is prepared to receive data<br>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 | 0's |
| 1.153.5:4 | Coefficient (+1) status | 5:4 = 11 = maximum<br>10 = minimum<br>01 = updated<br>00 = not updated | R/O | 00 |
| 1.153.3:2 | Coefficient (0) status | 3:2 = 11 = maximum<br>10 = minimum<br>01 = updated<br>00 = not updated | R/O | 00 |
| 1.153.1:0 | Coefficient (-1) status | 1:0 = 11 = maximum<br>10 = minimum<br>01 = updated<br>00 = not updated | R/O | 00 |

## MDIO Register 1.154: 10GBASE-KR LD Coefficient Update

Figure 6-22 shows the MDIO Register 1.154: 10GBASE-KR LD coefficient update.

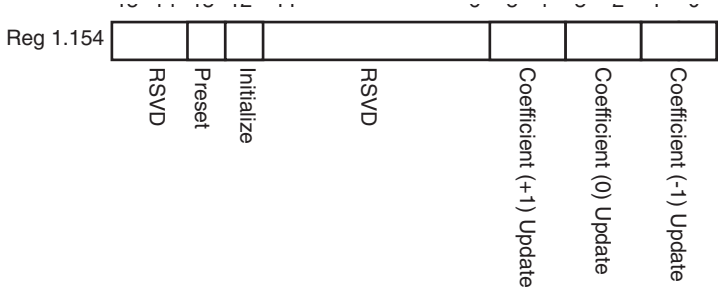


*Figure 6-22:* **10GBASE-KR LD Coefficient Update Register**

Table 6-20 shows the 10GBASE-KR LD coefficient update register bit definitions.

*Table 6-20:* **10GBASE-KR LD Coefficient Update Register Bit Definitions**

| Bit(s) | 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<br>0 = Normal operation | R/O[a] | 0 |
| 1.154.12 | Initialize | 1 = Initialize coefficients<br>0 = Normal operation | R/O[a] | 0 |
| 1.154.11:6 | Reserved | The block always returns '0' for this bit and ignores writes. | R/O | 0's |
| 1.154.5:4 | Coefficient (+1) update | 5:4 = 11 = reserved<br>10 = decrement<br>01 = increment<br>00 = hold | R/O[a] | 00 |
| 1.154.3:2 | Coefficient (0) update | 3:2 = 11 = reserved<br>10 = decrement<br>01 = increment<br>00 = hold | R/O[a] | 00 |
| 1.154.1:0 | Coefficient (-1) update | 1:0 = 11 = reserved<br>10 = decrement<br>01 = increment<br>00 = hold | R/O[a] | 00 |

a. These registers are programmed by writing to register 1.65520

## MDIO Register 1.155: 10GBASE-KR LD Status

Figure 6-23 shows the MDIO Register 1.155: 10GBASE-KR LD status.

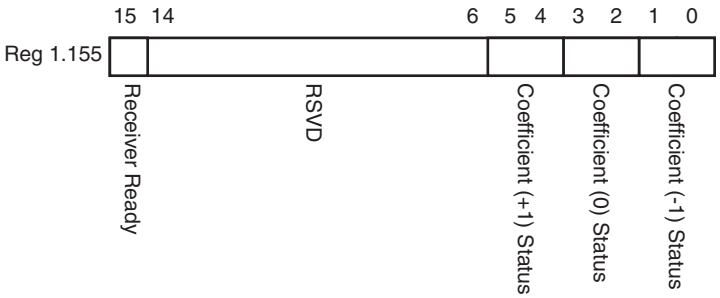


*Figure 6-23:* **10GBASE-KR LD Status Register**

Table 6-21 shows the 10GBASE-KR LD status register bit definitions.

*Table 6-21:* **10GBASE-KR LD Status Register Bit Definitions**

| Bit(s) | 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<br>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 | 0's |
| 1.155.5:4 | Coefficient (+1) status | 5:4 = 11 = maximum<br>10 = minimum<br>01 = updated<br>00 = not updated | R/O | 00 |
| 1.155.3:2 | Coefficient (0) status | 3:2 = 11 = maximum<br>10 = minimum<br>01 = updated<br>00 = not updated | R/O | 00 |
| 1.155.1:0 | Coefficient (-1) status | 1:0 = 11 = maximum<br>10 = minimum<br>01 = updated<br>00 = not updated | R/O | 00 |

## MDIO Register 1.170: 10GBASE-R FEC Ability

Figure 6-24 shows the MDIO Register 1.170: 10GBASE-R FEC Ability.


*Figure 6-24:* **10GBASE-R FEC Ability Register**

Table 6-22 shows the 10GBASE-R FEC Ability register bit definitions.

*Table 6-22:* **10GBASE-R FEC Ability Register Bit Definitions**

| Bit(s) | Name | Description | Attributes | Default Value |
|---|---|---|---|---|
| 1.170.15:2 | Reserved | The block always returns '0' for this bit and ignores writes. | R/O | 0's |
| 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 6-25 shows the MDIO Register 1.170: 10GBASE-R FEC Control.

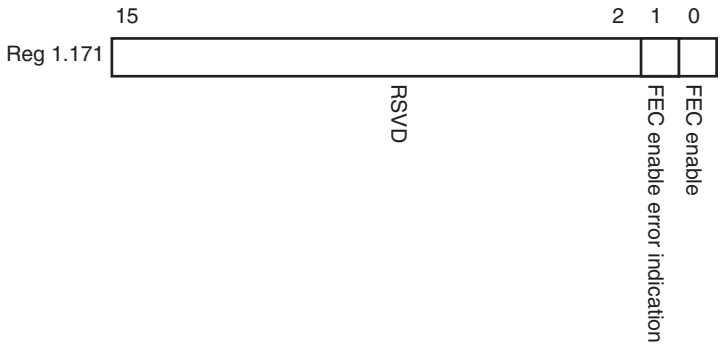


*Figure 6-25:* **10GBASE-R FEC Control Register**

Table 6-23 shows the 10GBASE-R FEC Control register bit definitions.

*Table 6-23:* **10GBASE-R FEC Control Register Bit Definitions**

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

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

Figure 6-26 shows the MDIO Register 1.172: 10GBASE-R FEC Corrected Blocks (lower).

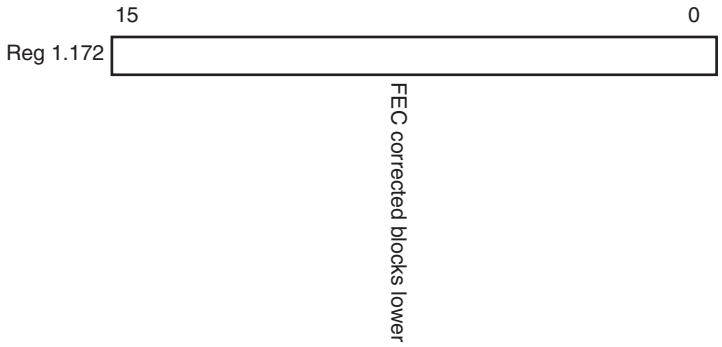


*Figure 6-26:* **10GBASE-R FEC Corrected Blocks (Lower) Register**

Table 6-24 shows the 10GBASE-R FEC Corrected Blocks (lower) register bit definitions.

*Table 6-24:* **10GBASE-R FEC Corrected Blocks (Lower) Register Bit Definitions**

| Bit(s) | Name | Description | Attributes | Default Value |
|---|---|---|---|---|
| 1.172.15:0 | FEC corrected blocks | Bits 15:0 of the Corrected Blocks count | R/O | 0's |

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

Figure 6-27 shows the MDIO Register 1.173: 10GBASE-R FEC Corrected Blocks (upper).

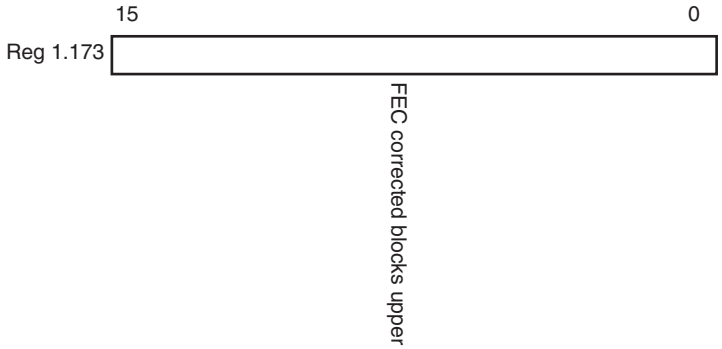


*Figure 6-27:* **10GBASE-R FEC Corrected Blocks (Upper) Register**

Table 6-25 shows the 10GBASE-R FEC Corrected Blocks (upper) register bit definitions.

*Table 6-25:* **10GBASE-R FEC Corrected Blocks (Upper) Register Bit Definitions**

| Bit(s) | Name | Description | Attributes | Default Value |
|---|---|---|---|---|
| 1.172.15:0 | FEC corrected blocks | Bits 31:16 of the Corrected Blocks count | R/O | 0's |

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

Figure 6-28 shows the MDIO Register 1.174: 10GBASE-R FEC Uncorrected Blocks (lower).

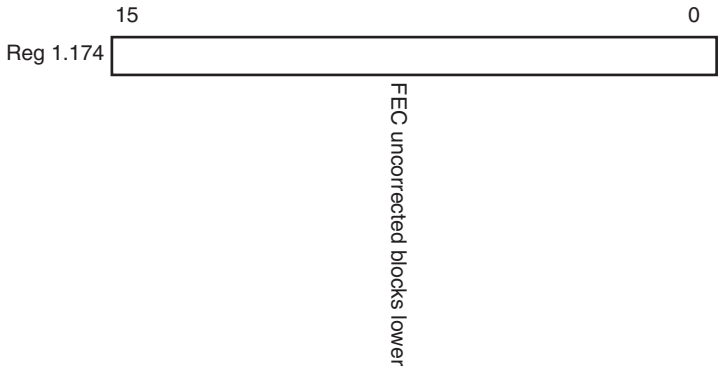


*Figure 6-28:* **10GBASE-R FEC Uncorrected Blocks (Lower) Register**

Table 6-26 shows the 10GBASE-R FEC Uncorrected Blocks (lower) register bit definitions.

*Table 6-26:* **10GBASE-R FEC Uncorrected Blocks (Lower) Register Bit Definitions**

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

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

Figure 6-29 shows the MDIO Register 1.175: 10GBASE-R FEC Uncorrected Blocks (upper).

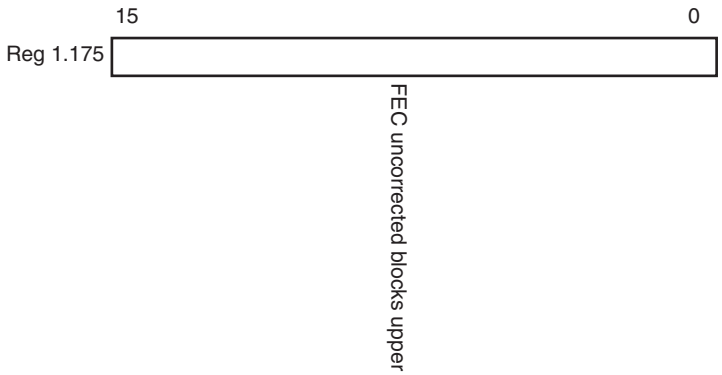


*Figure 6-29:* **10GBASE-R FEC Uncorrected Blocks (Upper) Register**

Table 6-27 shows the 10GBASE-R FEC Uncorrected Blocks (upper) register bit definitions.

*Table 6-27:* **10GBASE-R FEC Uncorrected Blocks (Upper) Register Bit Definitions**

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

## MDIO Register 1.32788: Vendor-Specific PMA Loopback Control (Virtex-6 FPGAs Only)

Figure 6-30 shows the MDIO 1.32788 Register: Vendor-Specific PMA Loopback Control.

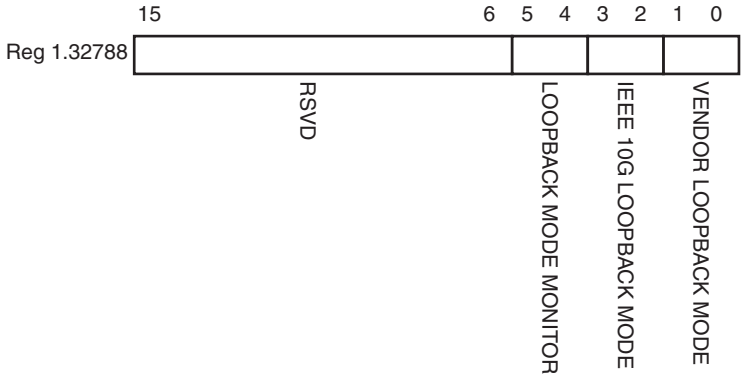


*Figure 6-30:* **Vendor-Specific PMA Loopback Control Register**

*Table 6-28:* **Vendor-Specific PMA Loopback Control**

| Bit(s) | Name | Description | Attributes | Default Value |
|---|---|---|---|---|
| 1.32788.15:6 | Reserved | The block always returns '0' for these bits and writes are ignored | R/O | All '0' |
| 1.32788.5:4 | Loopback Mode Monitor | Bits reflect the current state of loopback control. If 1.32788.1:0 is set to '00' and register 1.0.0 is set to '1', then these bits are equal to 1.32788.3:2, otherwise these bits are equal to 1.32788.1:0.<br>Consult the *Virtex-6 FPGA GTH Transceivers User Guide* for details. | R/O | '00' |
| 1.32788.3:2 | IEEE 10G Loopback Mode | Configure the course of loopback to RX.<br>These values are applicable only when 1.0.0 is asserted and the vendor config lane loopback mode is disabled.<br>Encodings:<br>'00': IEEE PMA loopback disabled<br>'01': TX output<br>'10': TX predriver<br>'11': Reserved | R/W | '01' |
| 1.32788.1:0 | Vendor Specific Loopback Mode | Configure source of on-chip loopback connection to RX.<br>Encodings:<br>'00': Vendor loopback disabled<br>'01': TX output<br>'10': TX predriver<br>'11': Reserved | R/W | '00' |

## MDIO Register: 1.65520: Vendor-Specific LD Training

Figure 6-31 shows the MDIO Register 1.65520: Vendor-specific LD Training.

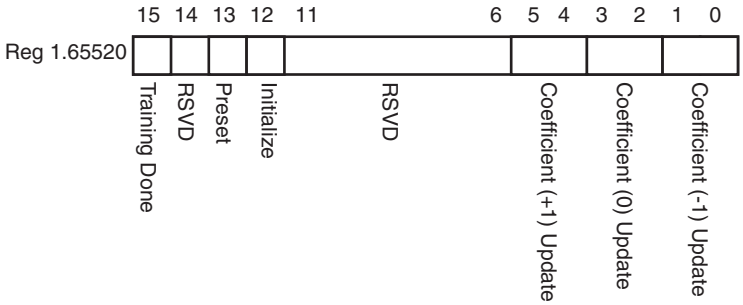


*Figure 6-31:* **Vendor-specific LD Training Register**

Table 6-29 shows the Vendor-specific LD Training register bit definitions.

*Table 6-29:* **Vendor-Specific LD Training Register Bit Definitions**

| Bit(s) | 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[a] | 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<br>0 = Normal operation | R/O[b] | 0 |
| 1.65520.12 | Initialize | 1 = Initialize coefficients<br>0 = Normal operation | R/O[b] | 0 |
| 1.65520.5:4 | Coefficient (+1) update | 5:4 = 11 = reserved<br>10 = decrement<br>01 = increment<br>00 = hold | R/O[b] | 00 |
| 1.65520.3:2 | Coefficient (0) update | 3:2 = 11 = reserved<br>10 = decrement<br>01 = increment<br>00 = hold | R/O[b] | 00 |
| 1.65520.1:0 | Coefficient (-1) update | 1:0 = 11 = reserved<br>10 = decrement<br>01 = increment<br>00 = hold | R/O[b] | 00 |

a. This register will be transferred automatically to register 1.155.15.
b. These registers will be transferred automatically to register 1.154.

## MDIO Register 1.65535: Core Version Info - Virtex-7/Kintex-7 FPGAs Only

Figure 6-32 shows the MDIO 1.65535 Register: Core Version Info

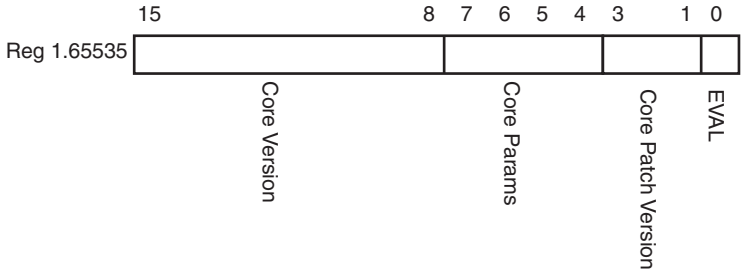


*Figure 6-32:* **Core Version Info Register**

*Table 6-30:* **Core Version Information**

| Bit(s) | 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 | x'22' for version 2.3 of core |
| 1.65535.7:4 | Core parameters | Bit 7 = 1 = KR included<br>Bit 6 - reserved<br>Bit 5 = 1 = AN included<br>Bit 4 = 1 = FEC included | R/O | Depends on core generation parameters |
| 1.65535.3:1 | Core Patch Version | Bits 3..1 give the patch number, if any, for the core. | R/O | '000' |
| 1.65535.0 | V7/K7 only: EVAL | 1 = This core was generated using a Hardware Evaluation license | R/O | '0' |

## MDIO Register 3.0: PCS Control 1

Figure 6-33 shows the MDIO Register 3.0: PCS Control 1.

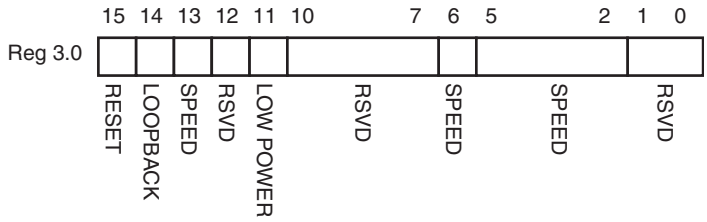


*Figure 6-33:* **PCS Control 1 Register**

Table 6-31 shows the PCS Control 1 register bit definitions.

*Table 6-31:* **PCS Control 1 Register Bit Definitions**

| Bit(s) | Name | Description | Attributes | Default Value |
|---|---|---|---|---|
| 3.0.15 | Reset | 1 = Block reset<br>0 = Normal operation<br>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<br>Self-clearing | 0 |
| 3.0.14 | 10GBASE-R /KR Loopback | 1 = Use PCS Loopback<br>0 = Do not use PCS Loopback | R/W | 0 |
| 3.0.13 | Speed Selection | The block always returns '1' for this bit.<br>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 6-34 shows the MDIO Register 3.1: PCS Status 1.

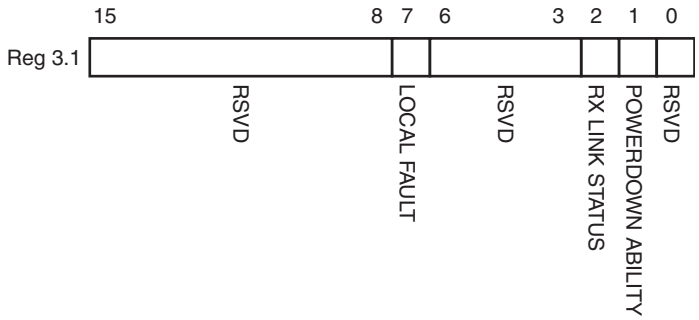


*Figure 6-34:* **PCS Status 1 Register**

Table 6-32 show the PCS 1 register bit definitions.

*Table 6-32:* **PCS Status 1 Register Bit Definition**

| Bit(s) | 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 | Virtex-6: The block always returns '0' for this bit.<br>V7/K7: 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<br>0 = The PCS receive link is down<br>This is a latching Low version of bit 3.32.12. | R/O<br>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 Registers 3.5 and 3.6: PCS Devices in Package

Figure 6-35 shows the MDIO Registers 3.5 and 3.6: PCS Devices in Package.

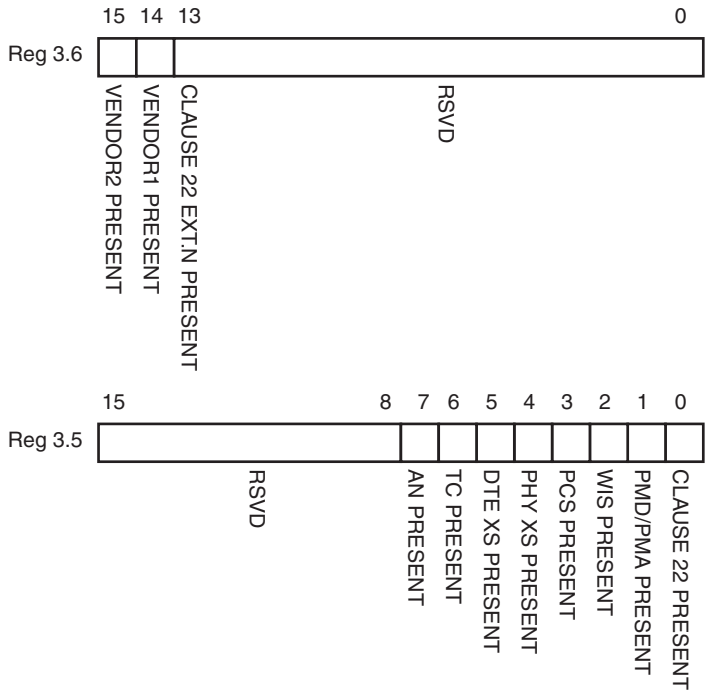


*Figure 6-35:* **PCS Devices in Package Registers**

Table 6-33 shows the PCS Devices in Package registers bit definitions.

*Table 6-33:* **PCS Devices in Package Registers Bit Definitions**

| Bit(s) | 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 | Virtex-6:The block always returns '1' for this bit.<br>V7/K7: The block always returns '0' for this bit. | R/O | 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 | Virtex-6: The block always returns’1’ for this bit.<br>V7/K7: 1 = AN Block included | R/O | 1 |
| 3.5.6 | TC present | The block always returns’0’ for this bit. | R/O | 0 |

*Table 6-33:* **PCS Devices in Package Registers Bit Definitions *(Cont'd)***

| Bit(s) | Name | Description | Attributes | Default Value |
|---|---|---|---|---|
| 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 |
| 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 6-36 shows the MDIO Register 3.7: 10G PCS Control 2.

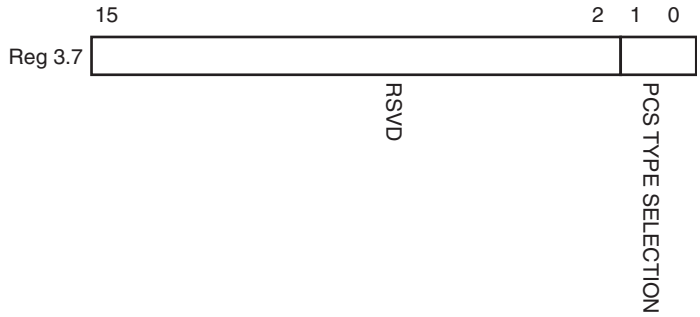


*Figure 6-36:* **10G PCS Control 2 Register**

Table 6-34 shows the 10 G PCS Control 2 register bit definitions.

*Table 6-34:* **10G PCS Control 2 Register Bit Definitions**

| Bit(s) | 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 are ignored. | R/W | 00 |

## MDIO Register 3.8: 10G PCS Status 2

Figure 6-37 shows the MDIO Register 3.8: 10G PCS Status 2.

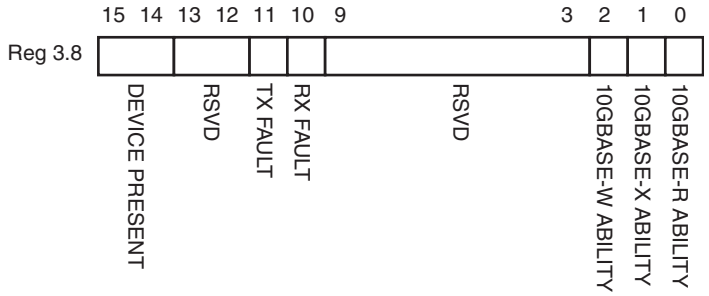


*Figure 6-37:* **10G PCS Status 2 Register**

Table 6-35 shows the 10G PCS Status 2 register bit definitions.

*Table 6-35:* **10G PCS Status 2 Register Bit Definitions**

| Bit(s) | 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 | Virtex-6: The block always returns '0' for this bit.<br>V7/K7: 1 = Transmit Fault detected | R/O | 0 |
| 3.8.10 | Receive local fault | Virtex-6: The block always returns '0' for this bit.<br>V7/K7: 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 6-38 shows the MDIO Register 3.32: 10GBASE-R Status 1.

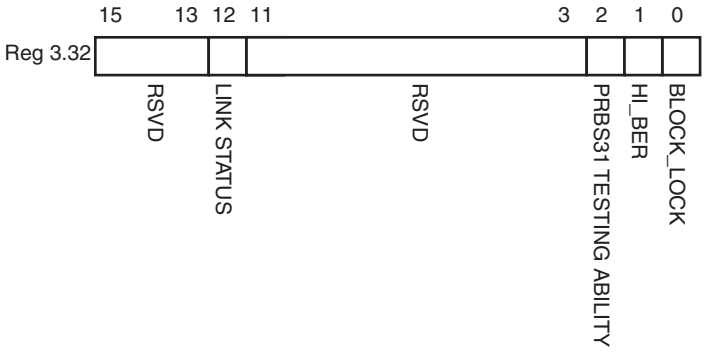


*Figure 6-38:* **10GBASE-R Status Register 1**

Table 6-36 shows the 10GBASE-R Status register bit definitions.

*Table 6-36:* **10GBASE-R Status Register 1 Bit Definitions**

| Bit(s) | 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;<br>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<br>0 = RX not showing hi ber | R/O | 0 |
| 3.32.0 | Block Lock | 1 = RX is synchronized;<br>0 = RX is not synchronized. | R/O | 0 |

## MDIO Register 3.33: 10GBASE-R Status 2

Figure 6-39 shows the MDIO Register 3.33: 10GBASE-R Status 2.

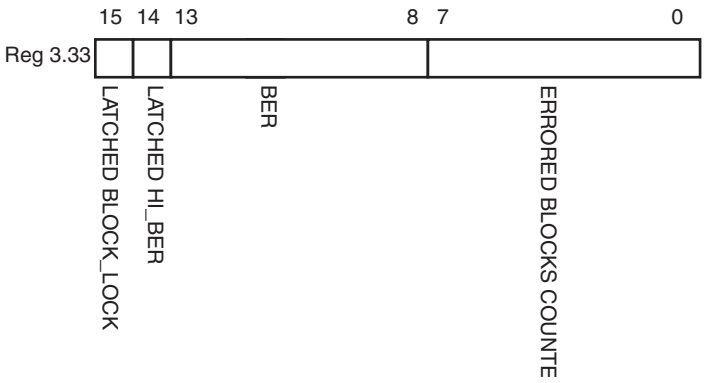


*Figure 6-39:* **10GBASE-R Status Register 2**

Table 6-37 shows the 10GBASE-R Status register bit definition. All bits are cleared when read.

*Table 6-37:* **10GBASE-R Status Register 2 Bit Definitions**

| Bit(s) | 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 6-40 shows the MDIO Register 3.34-37 10GBASE-R Test Pattern Seed A.

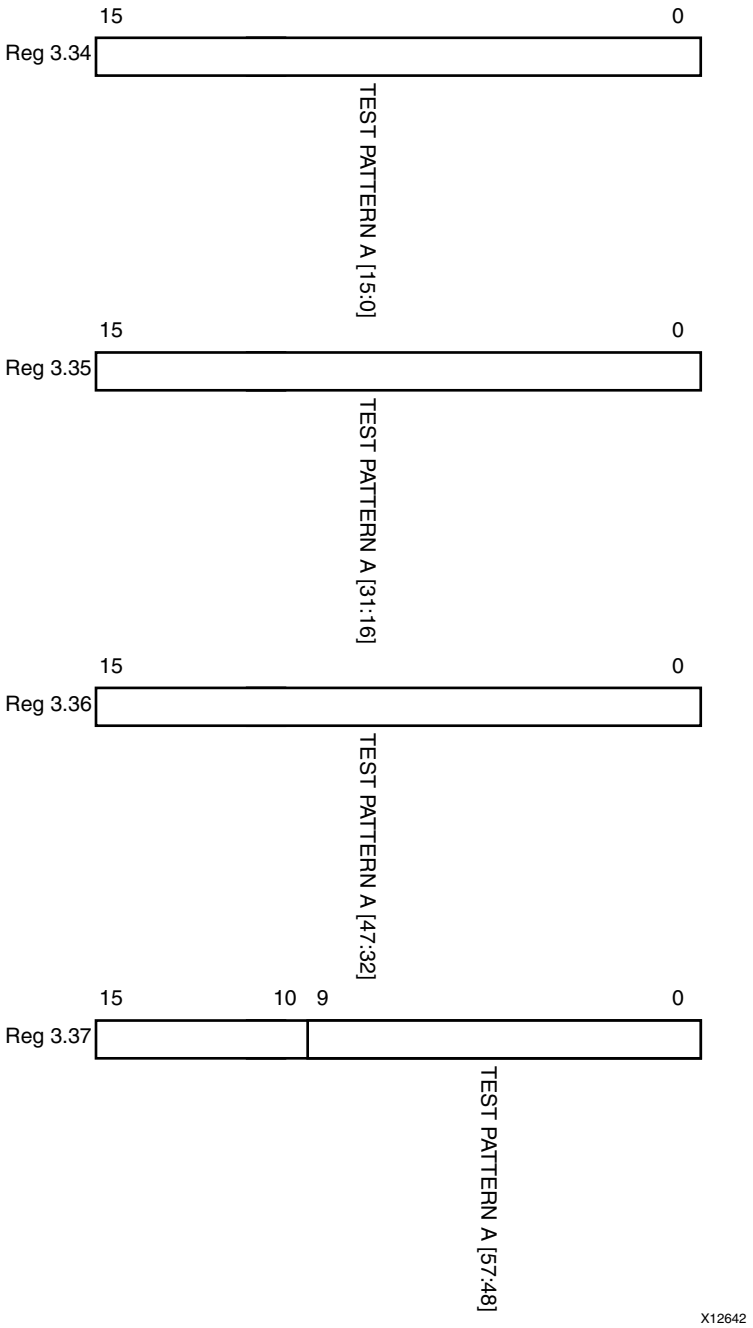


*Figure 6-40:* **10GBASE-R Test Pattern Seed A0-3 Registers**

Table 6-38 shows the 10GBASE-R Test Pattern Seed A0-2 register bit definitions.

*Table 6-38:* **10GBASE-R Test Pattern Seed A0-2 Register Bit Definitions**

| Bit(s) | Name | Description | Attributes | Default Value |
|---|---|---|---|---|
| 3.34-36.15:0<br>3.37.9:0 | Seed A bits 15:0, 31:16, 47:32, 57:48 resp | Seed for PRBS testing | R/W | Virtex-6: 3.34.0 = 1, all other bits = 0<br>V7/K7: all '0's |

## MMDIO Register 3.38-41: 10GBASE-R Test Pattern Seed B0-3

Figure 6-41 shows the MDIO Register 3.38-41: 10GBASE-R Test Pattern Seed B.

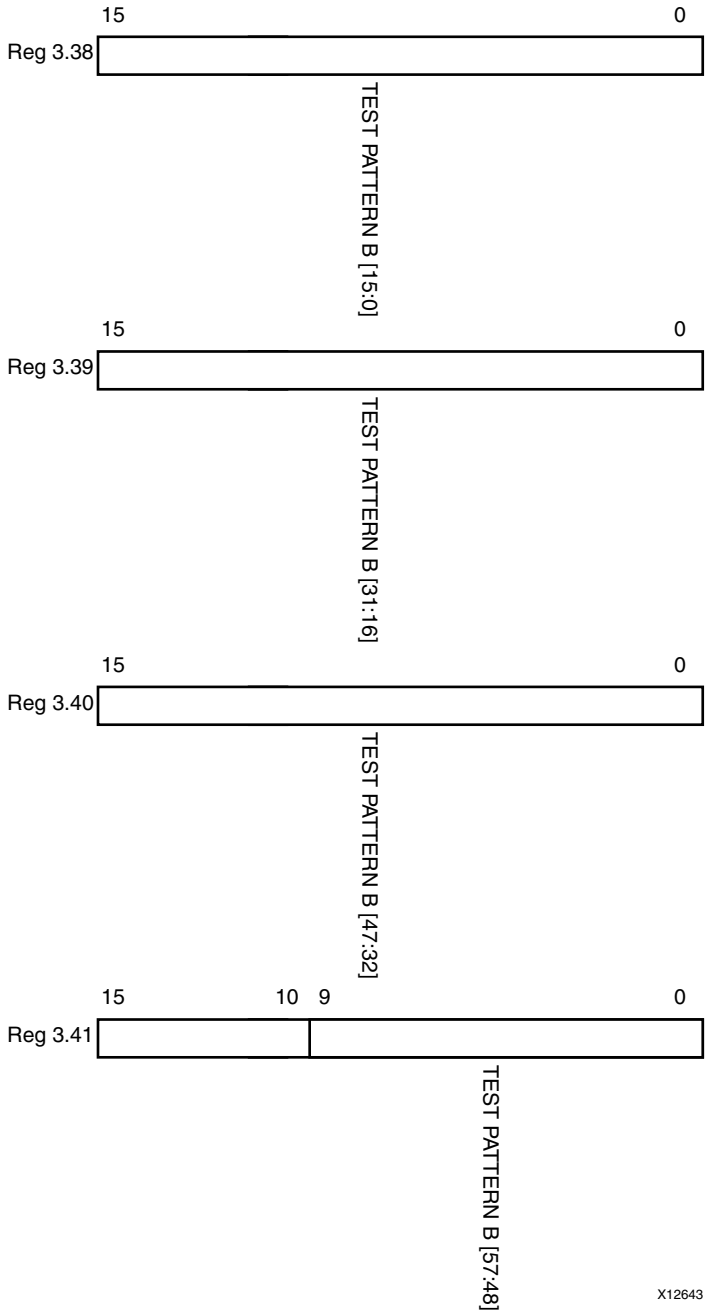


*Figure 6-41:* **10GBASE-R Test Pattern Seed B0-3 Registers**

Table 6-39 shows the 10GBASE-R Test Pattern Seed B0-3 register bit definitions.

*Table 6-39:* **10GBASE-R Test Pattern Seed B0-3 Register Bit Definitions**

| Bit(s) | Name | Description | Attributes | Default Value |
|---|---|---|---|---|
| 3.38-40.15:0<br>3.41.9:0 | Seed B bits 15:0, 31:16, 47:32, 57:48 resp | Seed for PRBS testing | R/W | Virtex-6: 3.38.0 = 1, all other bits = 0<br><br>V7/K7: all '0's |

## MDIO Register 3.42: 10GBASE-R Test Pattern Control

Figure 6-42 shows the MDIO Register 3.42: 10GBASE-R Test Pattern Control

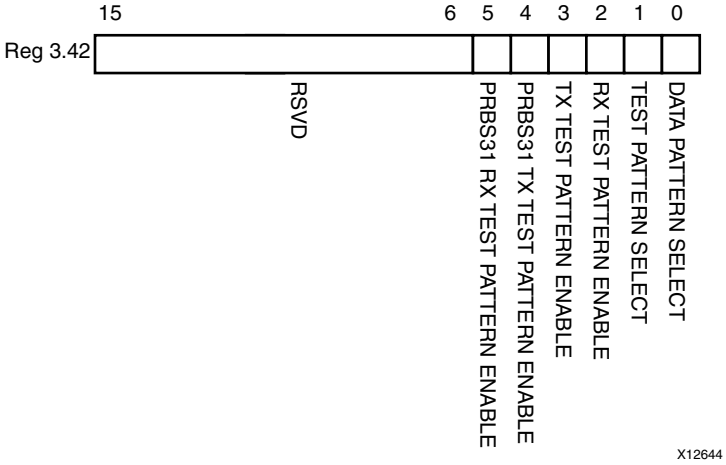


*Figure 6-42:* **10GBASE-R Test Pattern Control Register**

Table 6-40 shows the 10GBASE-R Test Pattern Control register bit definitions.

*Table 6-40:* **10GBASE-R Test Pattern Control Register Bit Definitions**

| Bit(s) | Name | Description | Attributes | Default Value |
|---|---|---|---|---|
| 3.42.15:6 | Reserved | The block always returns '0's for these bits. | R/O | All 0s |
| 3.42.5 | PRBS31 RX test pattern enable | 1 = Enable PRBS RX tests<br>0 = Disable PRBS RX tests | R/W | 0 |
| 3.42.4 | PRBS31 TX test pattern enable | 1 = Enable PRBS TX tests<br>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 |

*Table 6-40:* **10GBASE-R Test Pattern Control Register Bit Definitions *(Cont'd)***

| Bit(s) | Name | Description | Attributes | Default Value |
|---|---|---|---|---|
| 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<br>0 = Pseudo-Random | R/W | 0 |
| 3.42.0 | Data pattern select | 1 = Zeros pattern<br>0 = LF Data pattern | R/W | 0 |

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

Figure 6-43 shows the MDIO Register 3.43: 10GBASE-R Test Pattern Error Counter.

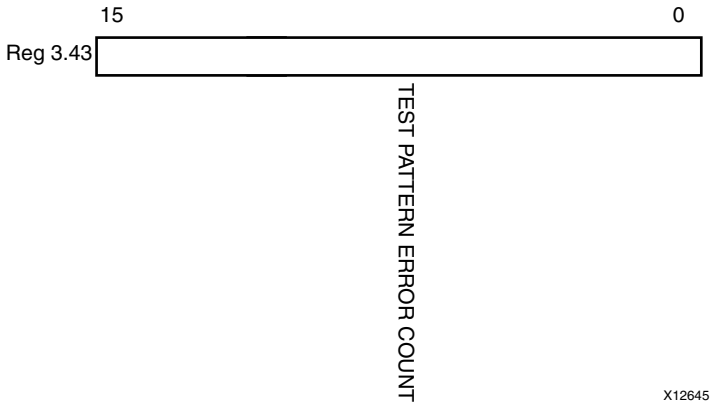


*Figure 6-43:* **10GBASE-R Test Pattern Error Counter Register**

Table 6-41 shows the 10GBASE-R Test Pattern Error Counter register bit definitions. This register is cleared when read.

*Table 6-41:* **10GBASE-R Test Pattern Error Counter Register Bit Definitions**

| Bit(s) | Name | Description | Attributes | Default Value |
|---|---|---|---|---|
| 3.43.15:0 | Test pattern error counter | Count of errors | R/O | All 0s |

## MDIO Register 3.32768: Vendor-Specific PCS Loopback Control - Virtex-6 FPGAs Only

Figure 6-44 shows the MDIO 3.32768 Register: Vendor-Specific PCS Loopback Control. Consult the *Virtex-6 FPGA GTH Transceivers User Guide (UG371)* for details.

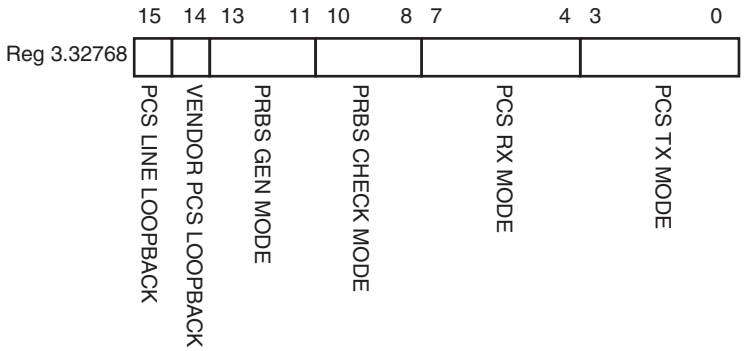


*Figure 6-44:* **Vendor-Specific PCS Loopback Control Register**

*Table 6-42:* **Vendor-Specific PCS Loopback Control**

| Bit(s) | Name | Description | Attributes | Default Value |
|---|---|---|---|---|
| 3.32768.15 | PCS Line Loopback | 1 = Loopback SerDes RX to SerDes TX | R/W | '0' |
| 3.32768.14 | Vendor PCS Loopback | 1 = Loopback PCS TX to PCS RX | R/W | '0' |
| 3.32768.13:11 | PRBS Gen Mode | 000: None<br>001: PRBS7<br>010: PRBS9<br>011: PRBS11<br>100: PRBS23<br>101: PRBS31<br>110: PPAT<br>111: Reserved | R/W | '000'<br>Values other than '000' and '101' have no effect. |

*Table 6-42:* **Vendor-Specific PCS Loopback Control *(Cont'd)***

| Bit(s) | Name | Description | Attributes | Default Value |
|---|---|---|---|---|
| 1.32768.10:8 | PRBS Check Mode | 000: None<br>001: PRBS7<br>010: PRBS9<br>011: PRBS11<br>100: PRBS23<br>101: PRBS31<br>110: PPAT<br>111: Reserved | R/W | '000<br><br>Values other than '000' and '101' have no effect. |
| 3.32768.7:4 | PCS RX Mode | 0000: Zero<br>0001: 64/66b<br>0010: XAUI<br>0011: Reserved<br>0100: PCIE®<br>0101: PCIE-500 (fixed PCLK 500MHz)<br>0110: 8B/10B 8-bit<br>0111: 8B/10B 16-bit<br>1000: 8-bit raw data<br>1001: 10-bit raw data<br>1010: 16-bit raw data<br>1011: 20-bit raw data<br>1100: PRBS<br>1101: Reserved<br>1110: 1000 Base-KX<br>1111: 802.3ap | R/W | Set from GTH transceiver PCS_MODE Attribute = '0001'<br>Do not write any other values to these bits. |

*Table 6-42:* **Vendor-Specific PCS Loopback Control *(Cont'd)***

| Bit(s) | Name | Description | Attributes | Default Value |
|---|---|---|---|---|
| 3.32768.3:0 | PCS TX Mode | 0000: Zero<br>0001: 64/66b<br>0010: XAUI<br>0011: Reserved<br>0100: PCIE<br>0101: PCIE-500 (fixed PCLK 500MHz)<br>0110: 8B/10B8-bit<br>0111: 8B/10B8 16-bit<br>1000: 8-bit raw data<br>1001: 10-bit raw data<br>1010: 16-bit raw data<br>1011: 20-bit raw data<br>1100: PRBS<br>1101: Reserved<br>1110: 1000 Base-KX<br>1111: 802.3ap | R/W | Set from GTH PCS_MODE Attribute = '0001'<br>Do not write any other values to these bits. |

## MDIO Register 3.65535: 125 µs Timer Control - Virtex-7/Kintex-7 FPGAs Only

Figure 6-45 shows the MDIO 3.65535 Register: 125 µs timer control.

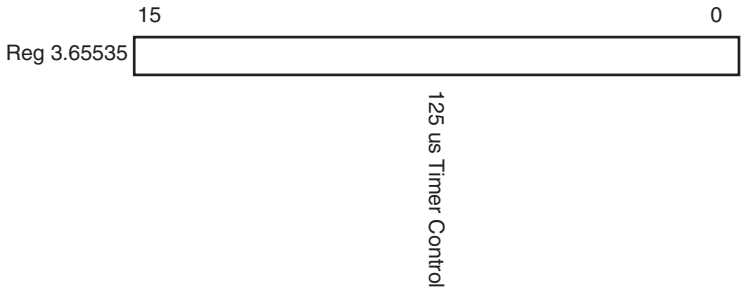


*Figure 6-45:* **25 µs Timer Control Register**

*Table 6-43:* **125 µs Timer Control**

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

## MDIO Register 7.0: AN Control

Figure 6-46 shows the MDIO Register 7.0: AN Control.

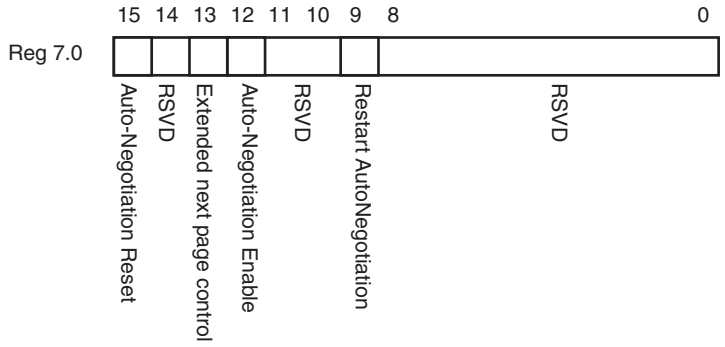


*Figure 6-46:* **AN Control Register**

Table 6-44 shows the AN Control register bit definitions.

*Table 6-44:* **AN Control Register Bit Definitions**

| Bit(s) | Name | Description | Attributes | Default Value |
|---|---|---|---|---|
| 7.0.15 | AN Reset | 1 = AN Reset<br>0 = AN normal operation | R/W<br>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<br>0 = Not supported | R/W | 0 |
| 7.0.12 | AN Enable | 1 = Enable AN Process<br>0 = Disable | R/W[a] | 1 |

*Table 6-44:* **AN Control Register Bit Definitions *(Cont'd)***

| Bit(s) | Name | Description | Attributes | Default Value |
|---|---|---|---|---|
| 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<br>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 | 0's |

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

## MDIO Register 7.1: AN Status

Figure 6-47 shows the MDIO Register 7.1: AN Status.

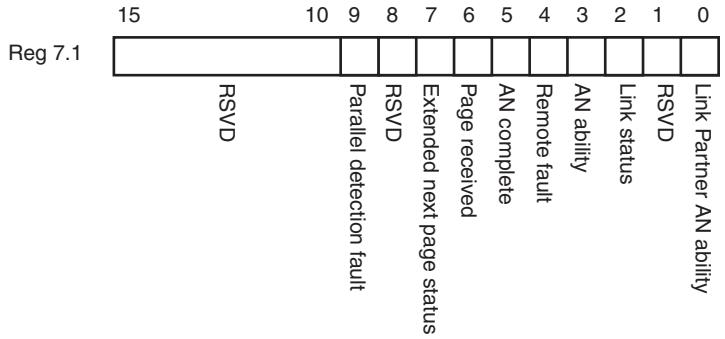


*Figure 6-47:* **AN Status Register**

Table 6-45 shows the AN Status register bit definitions.

*Table 6-45:* **AN Status Register Bit Definitions**

| Bit(s) | Name | Description | Attributes | Default Value |
|---|---|---|---|---|
| 7.1.15:10 | Reserved | The block always returns '0' for this bit and ignores writes. | R/O | 0's |
| 7.1.9 | Parallel Detection Fault | 1 = A fault has been detected through the parallel detection function<br>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<br>0 = XNP format is not allowed | R/O | 0 |
| 7.1.6 | Page Received | 1 = A page has been received<br>0 = No page received | R/O Latches High | 0 |

*Table 6-45:* **AN Status Register Bit Definitions *(Cont'd)***

| Bit(s) | Name | Description | Attributes | Default Value |
|---|---|---|---|---|
| 7.1.5 | AN Complete | 1 = AN process has completed<br>0 = not completed | R/O | 0 |
| 7.1.4 | Remote fault | 1 = remote fault condition detected<br>0 = not detected | R/O<br>Latches High | 0 |
| 7.1.3 | AN ability | 1 = PHY supports auto-negotiation<br>0 = PHY does not support auto-negotiation | R/O | 1 |
| 7.1.2 | Link status | 1 = Link is up<br>0 = Link is down | R/O<br>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<br>0 = not able | R/O | 0 |

## MDIO Register 7.16:17:18: AN Advertisement

Figure 6-48 shows the MDIO Register 7.16: AN Advertisement.

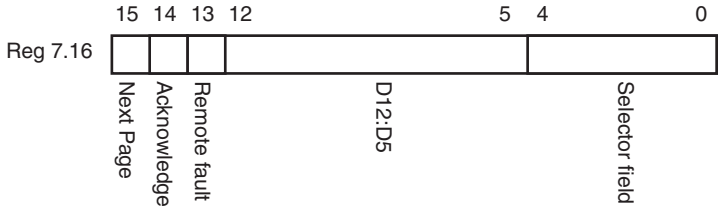


*Figure 6-48:* **AN Advertisement Register 0**

Table 6-46 shows the AN Advertisement register bit definitions.

*Table 6-46:* **AN Advertisement Register 0 Bit Definitions**

| Bit(s) | 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 | 0's |
| 7.16.4:0 | Selector Field | Consult IEEE802.3 | R/W | 0's |

Figure 6-49 shows the MDIO Register 7.17: AN Advertisement.

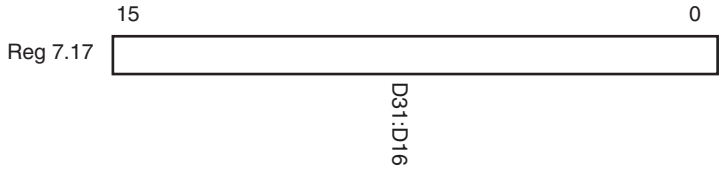


*Figure 6-49:* **AN Advertisement Register 1**

Table 6-47 shows the AN Advertisement register bit definitions.

*Table 6-47:* **AN Advertisement Register 1 Bit Definitions**

| Bit(s) | Name | Description | Attributes | Default Value |
|---|---|---|---|---|
| 7.17.15:0 | D31:D16 | Consult IEEE802.3 | R/W | 0 |

Figure 6-50 shows the MDIO Register 7.18: AN Advertisement.

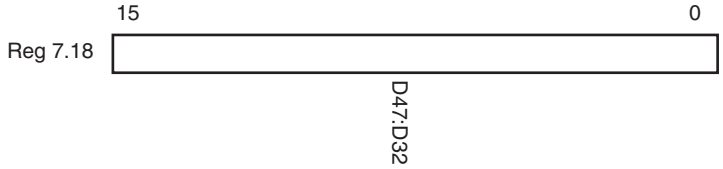


*Figure 6-50:* **AN Advertisement Register 2**

Table 6-48 shows the AN Advertisement register bit definitions.

*Table 6-48:* **AN Advertisement Register 2 Bit Definitions**

| Bit(s) | 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 6-51 shows the MDIO Register 7.19: AN LP Base Page Ability.

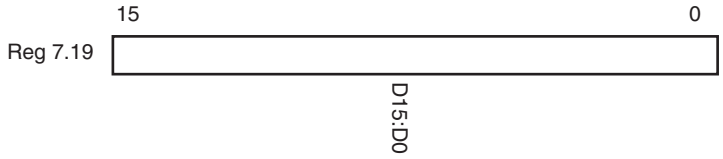


*Figure 6-51:* **AN LP Base Page Ability Register 0**

Table 6-49 shows the AN LP Base Page Ability register bit definitions.

*Table 6-49:* **AN LP Base Page Ability Register 0 Bit Definitions**

| Bit(s) | Name | Description | Attributes | Default Value |
|---|---|---|---|---|
| 7.19.15:0 | D15:D0 | Consult IEEE802.3 | R/O | 0 |

Figure 6-52 shows the MDIO Register 7.20: AN LP Base Page Ability.

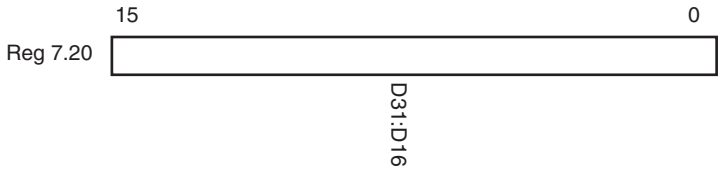


*Figure 6-52:* **AN LP Base Page Ability Register 1**

Table 6-50 shows the AN LP Base Page Ability register bit definitions.

*Table 6-50:* **AN LP Base Page Ability Register 1 Bit Definitions**

| Bit(s) | Name | Description | Attributes | Default Value |
|---|---|---|---|---|
| 7.20.15:0 | D31:D16 | Consult IEEE802.3 | R/W | 0 |

Figure 6-53 shows the MDIO Register 7.21: AN LP Base Page Ability.

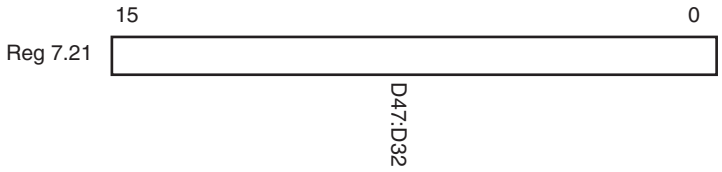


*Figure 6-53:* **AN LP Base Page Ability Register 2**

Table 6-51 shows the AN LP Base Page Ability register bit definitions.

*Table 6-51:* **AN LP Base Page Ability Register 2 Bit Definitions**

| Bit(s) | 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 6-54 shows the MDIO Register 7.22: AN XNP Transmit.

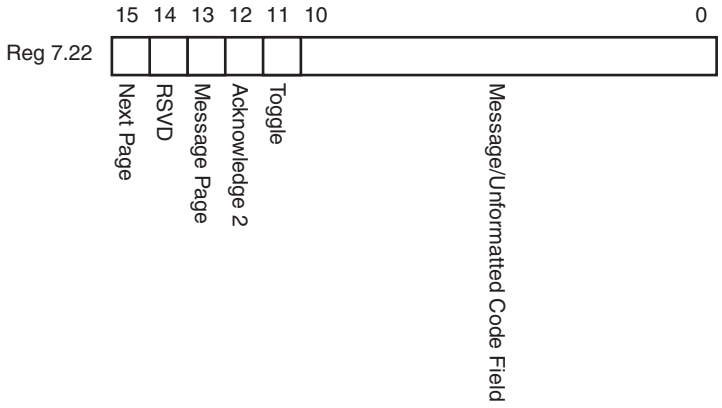


*Figure 6-54:* **AN XNP Transmit Register 0**

Table 6-52 shows the AN XNP Transmit register bit definitions.

*Table 6-52:* **AN XNP Transmit Register 0 Bit Definitions**

| Bit(s) | 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 | 0's |

Figure 6-55 shows the MDIO Register 7.23: AN XNP Transmit.

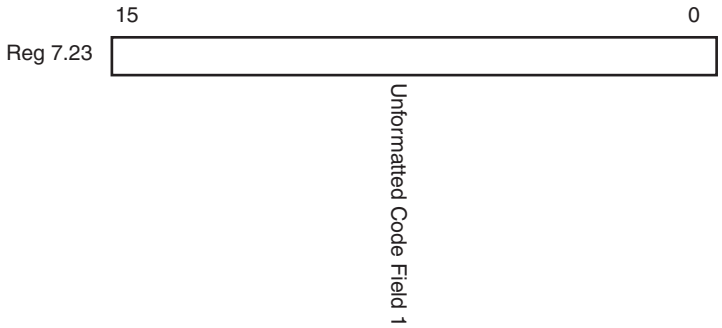


*Figure 6-55:* **AN XNP Transmit Register 1**

Table 6-53 shows the AN XNP Transmit register bit definitions.

*Table 6-53:* **AN XNP Transmit Register 1 Bit Definitions**

| Bit(s) | Name | Description | Attributes | Default Value |
|---|---|---|---|---|
| 7.23.15:0 | Unformatted Code Field 1 | Consult IEEE802.3 | R/W | 0's |

Figure 6-56 shows the MDIO Register 7.24: AN XNP Transmit.

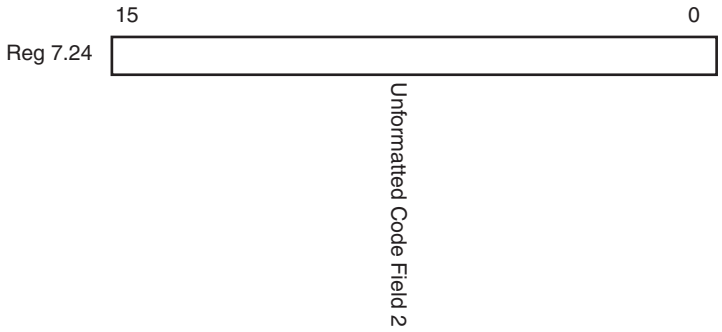


*Figure 6-56:* **AN XNP Transmit Register 2**

Table 6-54 shows the AN XNP Transmit register bit definitions.

*Table 6-54:* **AN XNP Transmit Register 2 Bit Definitions**

| Bit(s) | Name | Description | Attributes | Default Value |
|---|---|---|---|---|
| 7.24.15:0 | Unformatted Code Field 2 | Consult IEEE802.3 | R/W | 0's |

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

Figure 6-57 shows the MDIO Register 7.25: AN LP XNP Ability.

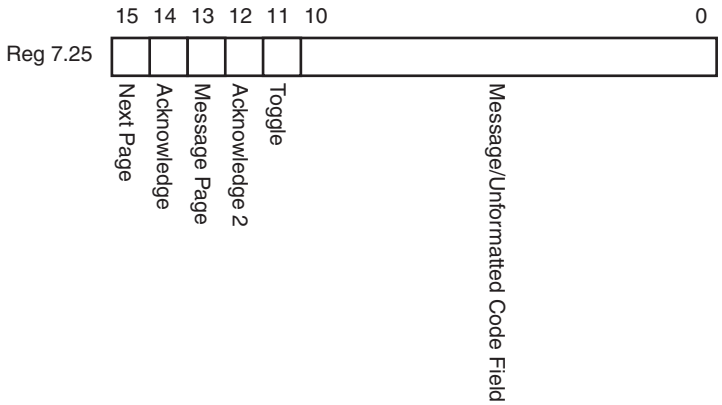


*Figure 6-57:* **AN LP XNP Ability Register 0**

Table 6-55 shows the AN LP XNP Ability register bit definitions.

*Table 6-55:* **AN LP XNP Ability Register 0 Bit Definitions**

| Bit(s) | 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 | 0's |

Figure 6-58 shows the MDIO Register 7.26: AN LP XNP Ability.

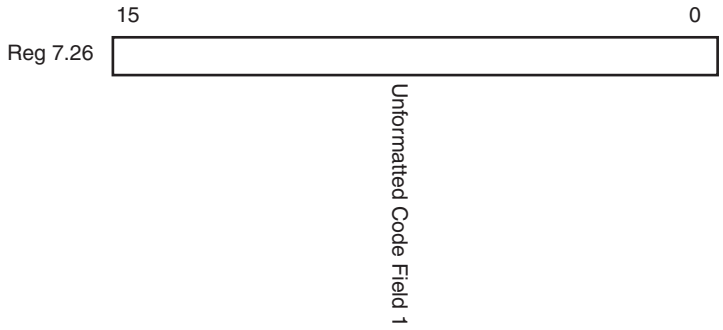


*Figure 6-58:* **AN LP XNP Ability Register 1**

Table 6-56 shows the AN LP XNP Ability register bit definitions.

*Table 6-56:* **AN LP XNP Ability Register 1 Bit Definitions**

| Bit(s) | Name | Description | Attributes | Default Value |
|---|---|---|---|---|
| 7.26.15:0 | Unformatted Code Field 1 | Consult IEEE802.3 | R/O | 0's |

Figure 6-59 shows the MDIO Register 7.27: AN LP XNP Ability.

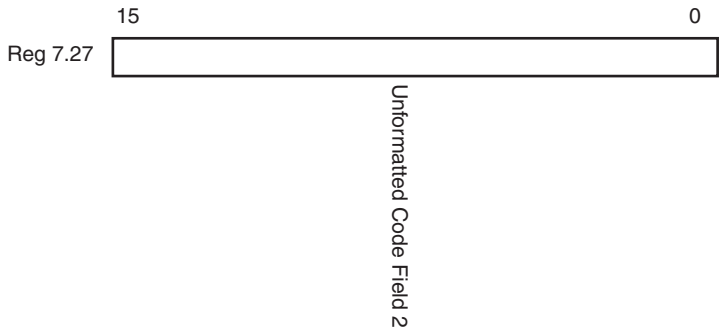


*Figure 6-59:* **AN LP XNP Ability Register 2**

Table 6-57 shows the AN LP XNP Ability register bit definitions.

*Table 6-57:* **AN LP XNP Ability Register 2 Bit Definitions**

| Bit(s) | Name | Description | Attributes | Default Value |
|---|---|---|---|---|
| 7.27.15:0 | Unformatted Code Field 2 | Consult IEEE802.3 | R/O | 0's |

## MDIO Register 7.48: Backplane Ethernet Status

Figure 6-60 shows the MDIO Register 7.48: Backplane Ethernet Status.

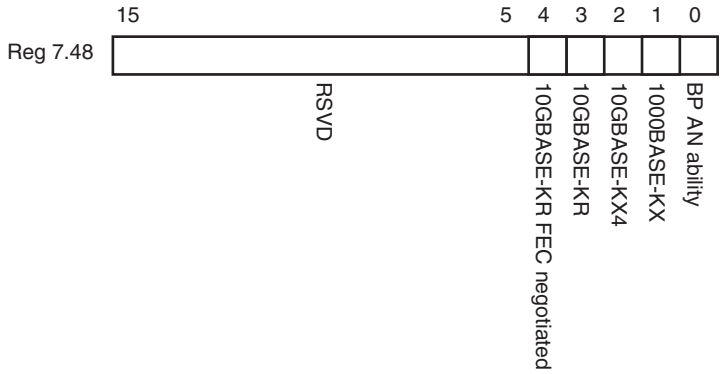


*Figure 6-60:* **Backplane Ethernet Status Register**

Table 6-58 shows the Backplane Ethernet Status register bit definitions.

*Table 6-58:* **Backplane Ethernet Status Register Bit Definitions**

| Bit(s) | 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 performs 10GBASE-KR FEC<br>0 = not negotiated | R/O | 0 |
| 7.48.3 | 10GBASE-KR | 1 = PMA/PMD is negotiated to performs 10GBASE-KR<br>0 = not negotiated | R/O | 0 |
| 7.48.2 | 10GBASE-KX4 | 1 = PMA/PMD is negotiated to performs 10GBASE-KX4<br>0 = not negotiated | R/O | 0 |
| 7.48.1 | 1000GBASE-KX | 1 = PMA/PMD is negotiated to performs 1000GBASE-KX<br>0 = not negotiated | R/O | 0 |
| 7.48.0 | BP AN ability | 1 = PMA/PMD is able to perform one of the preceding protocols<br>0 = not able | R/O | 1 |

# Configuration and Status Vectors

If the 10GBASE-R/KR core is generated without an MDIO interface, the key configuration and status information is carried on simple bit vectors, which are:

- configuration_vector[535:0]
- status_vector[447:0]

These vectors have changed after v1.2 of the core. Legacy-core shims are provided to ensure backward-compatibility.

## BASE-R

Table 6-59 shows the breakdown of the 10GBASE-R-specific configuration vector and Table 6-60 shows the breakdown of the status vector. Any bits not mentioned are assumed to be '0's.

*Table 6-59:* **Configuration Vector - BASE-R**

| Bit | IEEE Register | Description |
|---|---|---|
| 0 | 1.0.0 | PMA Loopback Enable |
| 15 | 1.0.15 | PMA Reset (1) |
| 16 | 1.9.0 | Global PMD Tx Disable |
| 83:80 | 1.32788.3:0 | PMA Vendor Specific Loopback Mode (HXT only) |
| 110 | 3.0.14 | PCS Loopback Enable |
| 111 | 3.0.15 | PCS/PMA Reset () |
| 169:112 | 3.37-3.34 | 10G PCS Test Pattern Seed A |
| 233:176 | 3.41-3.38 | 10G PCS Test Pattern Seed B |
| 240 | 3.42.0 | Data Pattern Select |
| 241 | 3.42.1 | Test Pattern Select |
| 242 | 3.42.2 | Rx Test Pattern Checking Enable |
| 243 | 3.42.3 | TX Test Pattern Enable |
| 244 | 3.42.4 | PRBS31 Tx Test Pattern Enable |
| 245 | 3.42.5 | PRBS31 RX Test Pattern Checking Enable |
| 271:270 | 3.32768.15:14 | PCS Loopback Type (Virtex-6 HXT FPGA only) |
| 399:384 | 3.65535 | 125 µs timer control (Virtex-7/Kintex-7 FPGAs only) |
| 512 | (1.1.2) () | Set PMA Link Status |
| 513 | (1.8.11:10) | Clear PMA/PMD Link Faults (Virtex-7/Kintex-7 FPGAs only) |
| 516 | (3.1.2) | Set PCS Link Status |
| 517 | (3.8.11:10) | Clear PCS Link Faults (K7/V7 only) |
| 518 | (3.33) | Clear PCS Status 2 |
| 519 | (3.43) | Clear Test Pattern Error Count |

1. These reset signals should be asserted for a single clock tick only.
2. Reset controls for the given registers

*Table 6-60:* **Status Vector - BASE-R**

| Bit | IEEE Register | Description |
|---|---|---|
| 15 | 1.0.15 | PMA Reset (2) |
| 18 | 1.1.2 | PMA/PMD Rx Link Status (Latching Low)(1) |
| 23 | 1.1.7 | PMA/PMD Fault (2) |
| 42 | 1.8.10 | PMA/PMD Rx Fault (Latching High)(2) |
| 43 | 1.8.11 | PMA/PMD Tx Fault (Latching High)(2) |
| 48 | 1.10.0 | Global PMD RX Signal Detect |
| 207:192 | 1.65535 | Core Info (Virtex-7/Kintex-7 FPGAs only) |
| 223 | 3.0.15 | PCS Reset (Virtex-7/Kintex-7 FPGAs only) |
| 226 | 3.1.2 | PCS Rx Link Status (Latching Low) |
| 231 | 3.1.7 | PCS Fault(1) |
| 250 | 3.8.10 | PCS Rx Fault (Latching High)(2) |
| 251 | 3.8.11 | PCS Tx Fault (Latching High)(2) |
| 256 | 3.32.0 | 10GBASE-R PCS Rx Locked (3) |
| 257 | 3.32.1 | 10GBASE-R PCS High BER |
| 268 | 3.32.12 | 10GBASE-R PCS Rx Link Status (3) |
| 279:272 | 3.33.7:0 | 10GBASE-R PCS Errored Blocks Counter |
| 285:280 | 3.33.13:8 | 10GBASE-R PCS BER Counter |
| 286 | 3.33.14 | Latched High Rx High BER |
| 287 | 3.33.15 | Latched Low Rx Block Lock |
| 303:228 | 3.43.15:0 | 10GBASE-R Test Pattern Error Counter |

1. This is tied to '1' for Virtex-6 FPGA BASE-R core.
2. This is tied to '0' for Virtex-6 FPGA BASE-R core.
3. For Virtex-6 devices this signal does not provide an instantaneous status value, but rather the value most recently captured from the equivalent latching register in the GTH.

## BASE-KR

Table 6-61 shows the additional signals in the configuration vector which are specific to BASE-KR functionality.

*Table 6-61:* **Configuration Vector - BASE-KR Specific**

| Bit | IEEE Register | Description |
|---|---|---|
| 32 | 1.150.0 | Restart Training |
| 33 | 1.150.1 | Enable Training |
| 53:48 | 1.152.5:0 | LP Training Coefficient Update (valid when 1.150.1 = '0') |
| 60 | 1.152.12 | LP Coefficient Initialize (valid when 1.150.1 = '0') |
| 61 | 1.152.13 | LP Coefficient Preset (valid when 1.150.1 = '0') |
| 64 | 1.171.0 | Enable FEC (1) |

*Table 6-61:* **Configuration Vector - BASE-KR Specific *(Cont'd)***

| Bit | IEEE Register | Description |
|---|---|---|
| 65 | 1.171.1 | FEC signal errors to PCS (1) |
| 281 | 7.0.9 | Restart Autonegotiation (2) |
| 284 | 7.0.12 | Enable Autonegotiation (2) |
| 285 | 7.0.13 | Extended Next Page Support (2) |
| 287 | 7.0.15 | Reset Autonegotiation (2) |
| 300:293 | 7.16.12:5 | AN Advertisement Data D12..D5 |
| 301 | 7.16.13 | AN Advertisement Data - Remote fault |
| 303 | 7.16.15 | AN Advertisement Data - Next Page |
| 319:304 | 7.17.15:0 | AN Advertisement Data - D31..D16 |
| 335:320 | 7.18.15:0 | AN Advertisement Data - D47..D32 |
| 346:336 | 7.22.10:0 | AN XNP - Message Unformatted Code Field |
| 348 | 7.22.12 | AN XNP Acknowledge 2 |
| 349 | 7.22.13 | AN XNP Message Page |
| 351 | 7.22.15 | AN XNP Next Page |
| 367:352 | 7.23.15:0 | AN XNP Unformatted Code Field 1 |
| 383:368 | 7.24.15:0 | AN XNP Unformatted Code Field 2 |
| 405:400 | 1.65520.5:0 | LD Training Coefficient Update |
| 412 | 1.65520.12 | LD Training Initialize |
| 413 | 1.65520.13 | LD Training Preset |
| 415 | 1.65520.15 | Training Done |
| 514 | (1.173:172) | Clear FEC Corrected Blocks Counter |
| 515 | (1.175:174) | Clear FEC Uncorrected Blocks Counter |
| 520 | (7.1.2) | Set AN Link Up/Down |
| 521 | (7.1.4) | Clear AN Remote Fault |
| 522 | (7.1.6) | Clear AN Page Received |
| 523 | (7.18:16) | Clear AN Adv Data Valid |
| 524 | (7.24:22) | Clear AN XNP Data Valid |

1. Only valid when the optional FEC block is included
2. Only valid when the optional AN block is included

Table 6-62 shows the additional signals in the status vector which are specific to BASE-KR functionality.

*Table 6-62:* **Status Vector - BASE-KR Specific**

| Bit | IEEE Register | Description |
|---|---|---|
| 67:64 | 1.151.3:0 | PMD Status |
| 85:80 | 1.153.5:0 | LP Status Report Coeff Status |
| 95 | 1.153.15 | LP Status Report Training Complete |
| 101:96 | 1.152.5:0 | LP Coefficient Update |
| 108 | 1.152.12 | LP Coefficient Initialize |
| 109 | 1.152.13 | LP Coefficient Preset |
| 117:112 | 1.155.5:0 | LD Status Report Coefficient Status |
| 127 | 1.155.15 | LD Status Report Training Complete |
| 159:128 | 1.173:172 | FEC Corrected Blocks Count(1) |
| 191:160 | 1.175:174 | FEC Uncorrected Blocks Count(1) |
| 319 | 7.0.15 | AN Reset(2) |
| 320 | 7.1.0 | LP AN Capable(2) |
| 322 | 7.1.2 | AN Link Up/Down (latching low)(2) |
| 323 | 7.1.3 | AN Ability(2) |
| 324 | 7.1.4 | AN Remote Fault(2) |
| 325 | 7.1.5 | AN Complete(2) |
| 326 | 7.1.6 | AN Page Received (latching high)(2) |
| 327 | 7.1.7 | AN Extended Next Page Used(2) |
| 383:336 | 7.21:19 | AN LP Base Page Ability(2) |
| 394:384 | 7.25.10:0 | AN LP XNP - Message Unformatted Code Field(2) |
| 395 | 7.25.11 | AN LP XNP - Toggle(2) |
| 396 | 7.25.12 | AN LP XNP - Acknowledge2(2) |
| 397 | 7.25.13 | AN LP XNP - Message Page(2) |
| 399 | 7.25.15 | AN LP XNP - Next Page(2) |
| 415:400 | 7.26.15:0 | AN LP XNP - Unformatted Code Field 1(2) |
| 431:416 | 7.27.15:0 | AN LP XNP - Unformatted Code Field 2(2) |
| 432 | 7.48.0 | Backplane AN Ability |
| 435 | 7.48.3 | Backplane Ethernet Status - KR negotiated |
| 436 | 7.48.4 | Backplane Ethernet Status - FEC negotiated |

1. Only valid when the optional FEC block is included
2. 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 6-61 shows how the status bit is cleared.

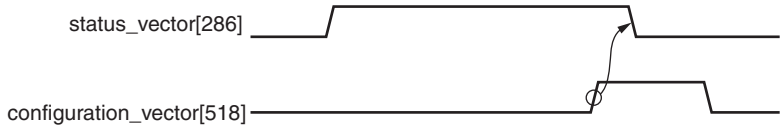


*Figure 6-61:* **Clearing the Latching-High Bits**

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 6-62 shows how the status bits are set.

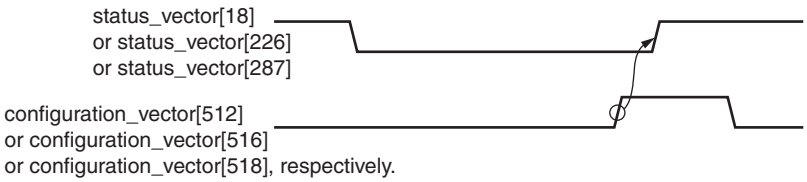


*Figure 6-62:* **Setting the Latching-Low Bits**

Similarly for Virtex®-7/Kintex™-7 FPGA designs, 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.

XILINX®

Chapter 7

# Constraining the Core

This chapter describes how to constrain a design containing the 10GBASE-R/KR core. This is illustrated by the UCF delivered with the core at generation time. See Chapter 10, Detailed Example Design, for a complete description of the CORE Generator™ tool output files and for details of the HDL example design.

***Caution!*** Not all constraints are relevant to specific implementations of the core; consult the UCF created with the core instance to see exactly which constraints are relevant.

## Device, Package, and Speed Grade Selection

This line selects the part to be used in the implementation run. Change this line so that it matches the part intended for the final application.

```
# Select the part to be used in the implementation run
CONFIG PART = xc6vhx255t-ff1155-2;
```

### Virtex-7/Kintex-7 FPGAs

The 10GBASE-R/KR core can be implemented in many Virtex®-7/Kintex™-7 FPGA packages, as long as the package itself is a flip-chip package, and the GTX/GTH transceiver supports the 10.3125 Gb/s line-rate, which requires speed grades -2 or -3.

### Virtex-6 FPGAs

The 10GBASE-R/KR core can be implemented in most Virtex-6 HXT devices with any speed grade. Only the FF1154 packages are **not** supported (no bonded-out GTHs)

## Clock Frequencies, Clock Management, and Placement

### Virtex-7/Kintex-7 FPGAs

The 10GBASE-R/KR core has one user clock domain:

- clk156 - 156.25 MHz derived from the Virtex-7/Kintex-7 FPGA transceiver.

This section specifies the main clock frequencies for the design.

- txusrclk2 - 322.27 MHz derived from the txoutclk output of the Virtex-7/Kintex-7 FPGA transceiver.
- rxusrclk2 - 322.27 MHz derived from the rxoutclk output of the Virtex-7/Kintex-7 FPGA transceiver.
- dclk - 78.125 MHz derived from the clk156 clock. (General Clocking)

### Virtex-6 FPGAs

The 10GBASE-R/KR core has one user clock domain:

- The `clk156` domain derived from the `refclk` input of the Virtex-6 FPGA GTH transceiver.

#### Main Clock Frequencies

This section specifies the main clock frequencies for the design.

```
##########################################################################
# Clock frequencies and clock management #
##########################################################################
NET "*txuserclkin" TNM_NET="clk156";
TIMESPEC "TS_clk156" = PERIOD "clk156" 6400 ps;

NET "*rxuserclkin" TNM_NET="rxclk156";
TIMESPEC "TS_rxclk156" = PERIOD "rxclk156" 6400 ps;
```

The clock frequency by default is set for 10-Gigabit Ethernet.

#### General Clocking

```
NET "dclk" TNM_NET="dclk";
TIMESPEC TS_dclk = PERIOD "dclk" 50 MHz;
```

This constraint defines the frequency of DCLK that is supplied to the Virtex-6 FPGA transceivers. The example design uses a nominal 50 MHz clock.

## Other Constraints

Among other constraints you might find in the UCF, the following are required to control the routing between FFs on different clock domains.

```
NET "*elastic_buffer_i*rd_truegray<?>" MAXDELAY = 6.0 ns;
NET "*elastic_buffer_i?can_insert_wra" TIG;
NET "*wr_gray*<?>" MAXDELAY = 6.0 ns;
NET "*rd_lastgray*<?>" MAXDELAY = 6.0 ns;
```

## Transceiver Placement

### Virtex-7/Kintex-7 FPGAs

No placement constraints have been introduced for the example design. Consult the *7 Series Transceivers User Guide* (UG476) for details.

### Virtex-6 HXT FPGAs

No placement constraints have been introduced for the example design. Consult the *Virtex-6 FPGA GTH Transceivers User Guide (UG371)* for details.

# MDIO

```
##################################################################
# MDIO-related constraints #
##################################################################

INST
"ten_gig_eth_pcs_pma_block/ten_gig_eth_pcs_pma_core/BU2/U0/*management
_inst/mdc_reg1" IOB=TRUE;
INST
"ten_gig_eth_pcs_pma_block/ten_gig_eth_pcs_pma_core/BU2/U0/*management
_inst/mdio_in_reg1" IOB=TRUE;
INST
"ten_gig_eth_pcs_pma_block/ten_gig_eth_pcs_pma_core/BU2/U0/*management
_inst/mdio_interface_1/mdio_out" IOB=TRUE;
INST
"ten_gig_eth_pcs_pma_block/ten_gig_eth_pcs_pma_core/BU2/U0/*management
_inst/mdio_interface_1/mdio_tri" IOB=TRUE;
```

These constraints set the correct attributes for the registers at the edge of the MDIO block. The TIMESPEC constrains the MDIO interface to 2.5 MHz. If you wish to overclock the MDIO interface, you must alter this constraint.

XILINX®

Chapter 8

# Design Considerations

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

## Virtex-7/Kintex-7 FPGAs Clocking

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

### Reference Clock

For Virtex®-7/Kintex™-7 FPGA transceivers, the reference clock must be running at 156.25 MHz.

### Transceiver Placement

A single IBUFDS_GTE2 block is used to feed the reference clocks for all GTXE2_CHANNEL and GTHE2_CHANNEL transceivers, through a GTXE2_COMMON or GTHE2_COMMON block.

For details about Virtex-7/Kintex-7 FPGA transceiver clock distribution, see the section on Clocking in the *7 Series Transceivers User Guide* (UG476).

### Internal Client-Side Interface

The clocking scheme for the internal client interface is shown in Figure 8-1.

The GTXE2_CHANNEL and GTHE2_CHANNEL primitives require a 156.25 MHz reference clock, as well as 322.26 MHz TX and RX user clocks. The latter must be created from the 322.26 MHz TXOUTCLK and RXOUTCLK outputs from that block.

The 156.25 MHz user-logic clock clk156 must be created from the transceiver reference clock to keep the user logic and GTXE2 synchronous.

A dedicated management/configuration clock, dclk, is used by the user logic and the transceiver and must be created from the clk156, at half the rate; that is: 78.125 MHz.

Figure 8-1 shows a possible clocking architecture with a single GTXE2_CHANNEL or GTHE2_CHANNEL block.

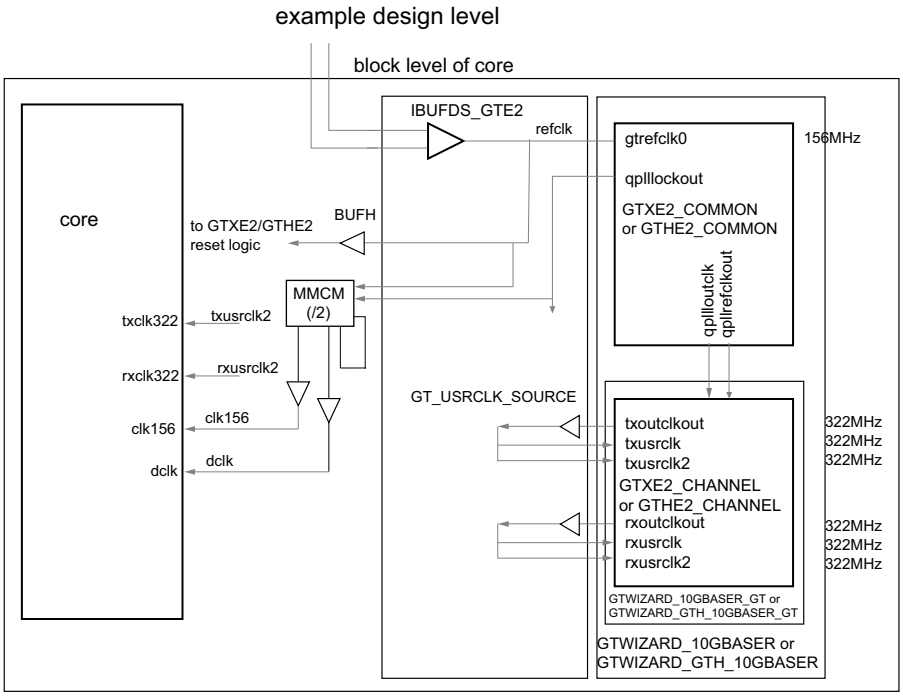


*Figure 8-1:* **Clocking Scheme for Internal Client-Side Interface: 7 Series FPGAs**

# Virtex-6 FPGAs Clocking

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

## Reference Clock

The Virtex®-6 FPGA GTH transceivers typically use a reference clock of 156.25 MHz to operate at a line rate of 10.3125 Gb/s.

## Transceiver Placement

Common to all schemes shown is that a single IBUFDS_GTHE1 block is used to feed the reference clocks for all GTH transceivers.

For details about Virtex-6 FPGA transceiver clock distribution, see the section on Clocking in the *Virtex-6 FPGA GTH Transceivers User Guide (UG371)*.

## Internal Client-Side Interface

The clocking scheme for the internal client interface is shown in Figure 8-2.

The GTH transceiver primitives require a 156.25 MHz clock. The 156.25 MHz clock sourced by the GTH transceiver is used as the clock for the netlist part of the 10GBASE-R/KR core and is typically also used for your logic.

A dedicated management/configuration clock is used by the GTH transceiver tiles. The example design uses a 50 MHz clock. Choosing a different frequency is possible. See the *Virtex-6 FPGA GTH Transceivers User Guide (UG371)* for details about this clock.

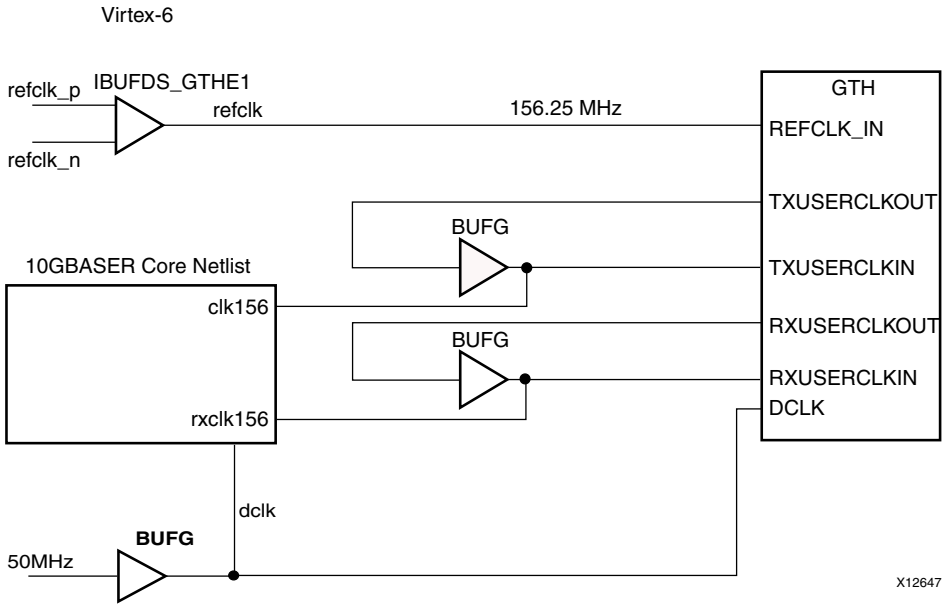


*Figure 8-2:* **Clock Scheme for Internal Client-Side Interface: Virtex-6 HXT FPGAs**

# Connecting Multiple Core Instances in Virtex-7 FPGAs

To instance multiple cores, instantiate those in the 10GBASER/KR block-level component. The `clk156`, `dclk` and `txusrclk2` clocking resources in the block level and gt_usrclk_source blocks can be shared between multiple cores, while the rxusrclk2 resources in the gt_usrclk_source block must be replicated for each core.

Replace the gtwizard_10gbaser (or gtwizard_gth_10gbaser for devices containing GTHE2 transceivers) instance with one with the same name but with up to four transceiver ports exposed (created in the CORE Generator™ tool) and then connect these to the multiple 10GBASER/KR cores and optionally a single MDIO bus.

Also shown in Figure 8-3 are several Ten Gigabit Ethernet MAC cores, to illustrate the simplicity of connecting those to the 10GBASE-R/KR cores, to the clock resources and to the MDIO bus. The MDIO 3-state controls are not required for on-chip MDIO connections – connect `mdio_in` on the 10GEMACs to `mdio_out` on the associated 10GBASE-R/KR cores, and vice versa. The 10GEMAC cores supply the MDC clock to the associated 10GBASE-R/KR cores.

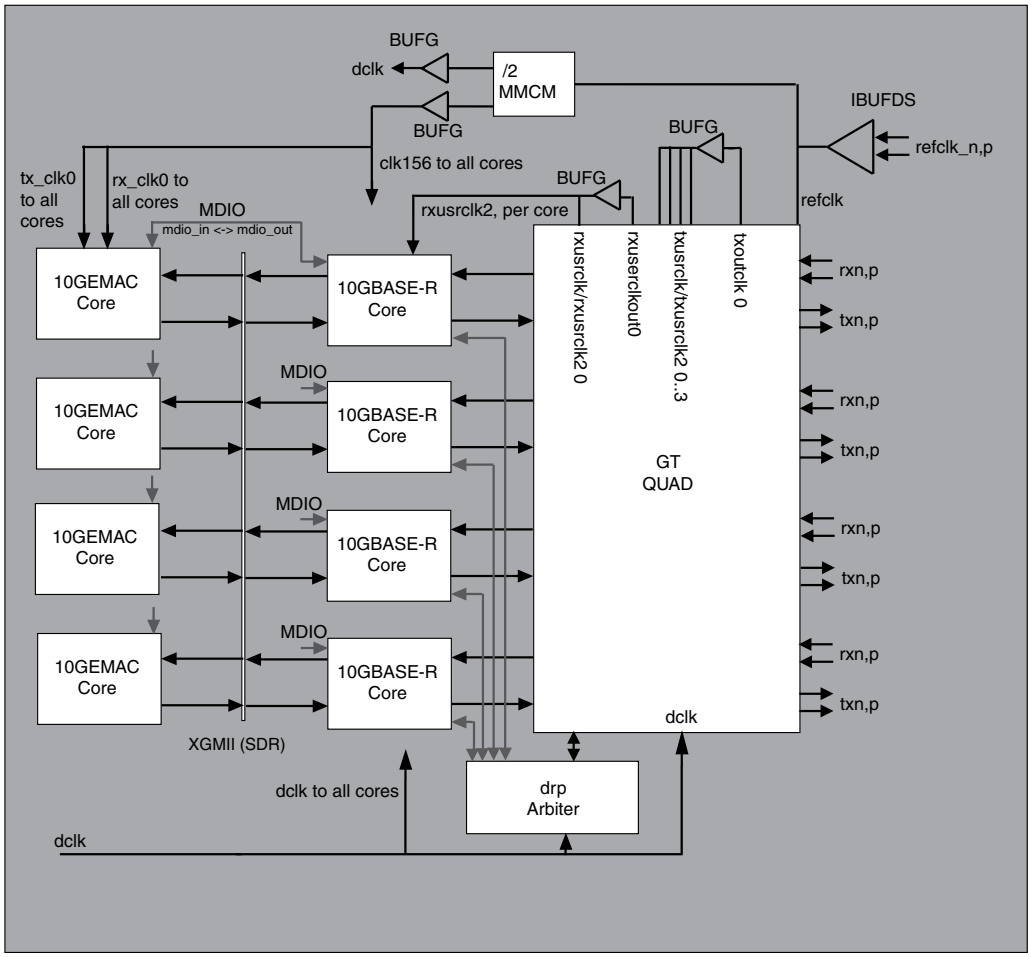


*Figure 8-3:* **Attaching Multiple Cores to a GT_QUAD Tile**

# Connecting Multiple Core Instances in Virtex-6 HXT FPGAs

To instantiate multiple cores, instantiate those in the 10GBASE-R/KR block-level component, multiple times. The clk156 clocking resources in the block level can be shared between multiple cores, while the rxclk156 resources must appear once for each core.

Replace the v6gth_wrapper by one with the same name but with up to 4 transceiver ports exposed and then connect these to the 10GBASE-R/KR cores and a single management arbiter block.

Also shown in the diagram are several Ten Gigabit Ethernet MAC cores, to illustrate the simplicity of connecting those to the 10GBASE-R/KR cores, to the clock resources and to the MDIO bus. The MDIO 3-state controls are not required for on-chip MDIO connections – connect `mdio_in` on the 10GEMACs to `mdio_out` on the associated 10GBASE-R/KR cores, and vice versa. The 10GEMAC cores supply the MDC clock to the associated 10GBASE-R/KR cores.

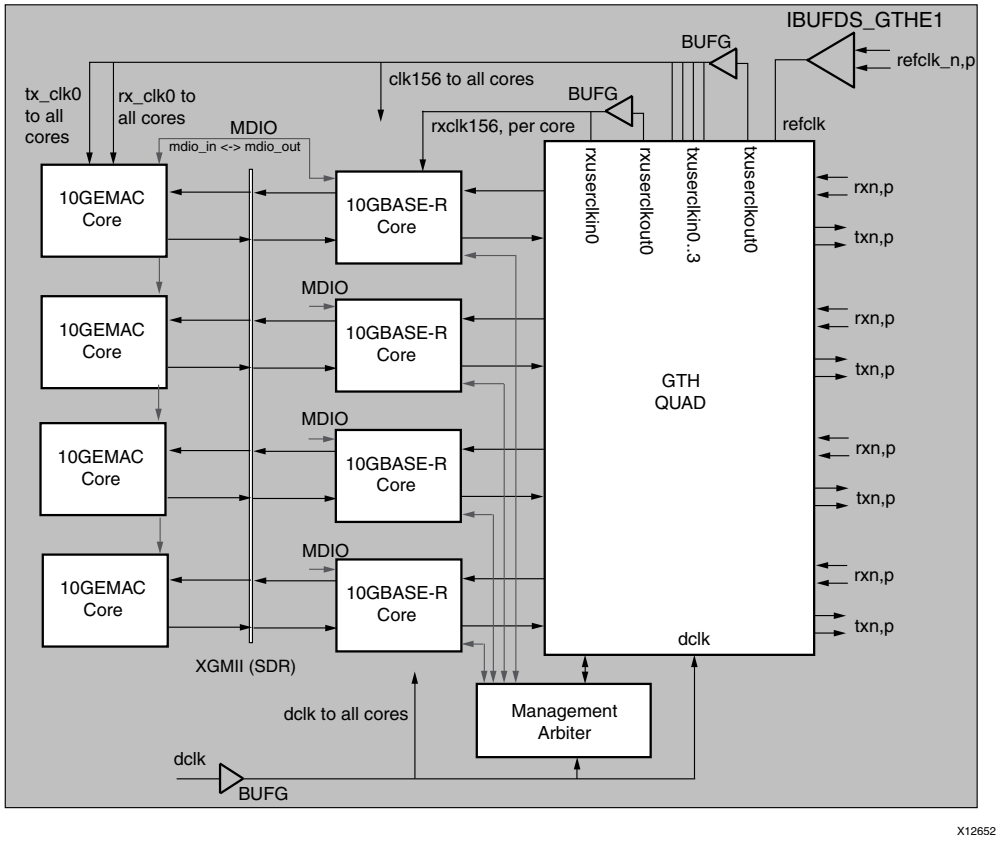


*Figure 8-4:* **Attaching Multiple Cores to a GTH_QUAD Tile**

See the *Virtex-6 FPGA GTH Transceivers User Guide (UG371)* for details on sharing the reference clock and any limitations.

## Using the DRP in Virtex-6 HXT FPGAs

The core, combined with the management arbiter block, accesses the GTH transceiver internal registers through the MGMT interface to the GTH transceiver. To do this, the DRP interface to the GTH transceiver must be disabled. This happens automatically when the core wishes to access those registers.

Access to the DRP is still available to the user, by setting the `drp_req` pin on the management arbiter, and waiting for the `drp_gnt` signal to be asserted before starting any DRP access. This allows any MGMT interface accesses to complete before switching over to the DRP interface and disabling the MGMT interface. The DRP interface is used exclusively until `drp_req` is set low again.

## Reset Circuits

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

## Receiver Termination: Virtex-7/Kintex-7 FPGAs

The receiver termination must be set correctly. See the *7 Series FPGA Transceivers User Guide*.

## Receiver Termination: Virtex-6 FPGAs

The receiver termination must be set correctly. See the *Virtex-6 FPGA GTH Transceivers User Guide (UG371)*.

XILINX®

Chapter 9

# Implementing the Core

This chapter describes how to simulate and implement your design containing the 10GBASE-R/KR core.

## Pre-implementation Simulation

A unit delay gate-level model of the 10GBASE-R/KR core netlist is provided as a CORE Generator™ tool output file. This can be used for simulation of the block in the design phase of a project.

For information about setting up your simulator to use the pre-implemented model, consult the Xilinx® *Synthesis and Verification Design Guide*, included in your Xilinx software installation.

The unit delay gate-level model of the 10GBASE-R/KR core can be found in the CORE Generator tool project directory. Details of the CORE Generator tool outputs can be found in Chapter 10, Detailed Example Design.

### VHDL

*component_name*.vhd

### Verilog

*component_name*.v

# Synthesis

## XST: VHDL

In the CORE Generator tool project directory, there is a *component_name*.vho file that is a component and instantiation template for the core. Use this to help instantiate the 10GBASE-R/KR core into your VHDL source.

After your entire design is complete, create:

- An XST project file *top_level_module_name*.prj listing all your source code files
- An XST script file *top_level_module_name*.scr containing your required synthesis options

To synthesize the design, run:

```
$ xst -ifn top_level_module_name.scr
```

See the *XST User Guide* for details on creating project and synthesis script files and running the xst program.

## XST: Verilog

In the CORE Generator tool project directory, there is a module declaration for the 10GBASE-R/KR core at:

```
<project_directory>/<component_name>/implement/component_name_mod.v
```

Use this module to help instance the 10GBASE-R/KR core into your Verilog source.

After your entire design is complete, create:

- An XST project file *top_level_module_name*.prj listing all your source code files. Make sure you include:

  %XILINX%/verilog/src/iSE/unisim_comp.v

  and

  <*project_directory*>/<*component_name*>/implement/*component_name*_mod.v

  as the first two files in the project list.
- An XST script file *top_level_module_name*.scr containing your required synthesis options.

To synthesize the design, run:

```
$ xst -ifn top_level_module_name.scr
```

See the *XST User Guide* for details about creating project and synthesis script files, and running the xst program.

# Implementation

## Generating the Xilinx Netlist

To generate the Xilinx netlist, the ngdbuild tool is used to translate and merge the individual design netlists into a single design database, the Native Generic Database (NGD) file. Also merged at this stage is the User Constraints File (UCF) for the design. An example of the `ngdbuild` command is:

```
$ ngdbuild top_level_module_name_example_design
```

## Mapping the Design

To map the logic gates of your design netlist into the CLBs and IOBs of the FPGA, run the `map` command. The `map` command writes out a physical design to an Native Circuit Description (NCD) file. An example of the `map` command is:

```
$ map -o mapped.ncd top_level_module_name_example_design
```

## Placing and Routing the Design

To place and route your design logic components (mapped physical logic cells) contained within an NCD file in accordance with the layout and timing requirements specified in the PCF file, the `par` command must be executed. The `par` command outputs the placed and routed physical design to an NCD file. An example of the `par` command is:

```
$ par mapped.ncd routed mapped.pcf
```

## Static Timing Analysis

To evaluate timing closure on a design and create a Timing Report file (TWR) derived from static timing analysis of the Physical Design file (NCD), the `trce` command must be executed. The analysis is typically based on constraints included in the optional PCF file. An example of the `trce` command is:

```
$ trce -e 10 routed -o routed mapped.pcf
```

## Generating a Bitstream

To create the configuration bitstream (BIT) file based on the contents of a physical implementation file (NCD), the `bitgen` command must be executed. The BIT file defines the behavior of the programmed FPGA. An example of the `bitgen` command is:

```
$ bitgen routed.ncd routed.bit mapped.pcf
```

# Post-Implementation Simulation

The purpose of post-implementation simulation is to verify that the design as implemented in the FPGA works as expected. This feature is available only for a core when the target device has reached 'Production' status.

## Generating a Simulation Model

To generate a chip-level simulation netlist for your design, the netgen command must be run.

### VHDL

```
$ netgen -sim -ofmt vhdl \
    -pcf mapped.pcf \
    -sim -dir .
    -tm top_level_module_name_example_design \
    routed.ncd routed.vhd
```

### Verilog

```
$ netgen -sim -ofmt verilog \
    -pcf mapped.pcf \
    -sim -dir . \
    -tm top_level_module_name_example_design \
    -sdf_anno false routed.ncd routed.v
```

## Using the Model

For information on setting up your simulator to use the pre-implemented model, consult the Xilinx *Synthesis and Verification Design Guide*, included in your Xilinx software installation.

# Other Implementation Information

For details about using the Xilinx implementation tool flow including command line switches and options, consult the software manuals that came with your Xilinx ISE® tools.

XILINX®

Chapter 10

# Detailed Example Design

This chapter provides detailed information about the example design, including a description of the files and the directory structure generated by the Xilinx® CORE Generator™ tool, the purpose and contents of the provided scripts, the contents of the example HDL wrappers, and the operation of the demonstration test bench.

- <project directory>
  Top-level project directory; name is user-defined.
  - <project directory>/<component name>
    Core release notes file
    - <component_name>/doc
      Product documentation
    - <component_name>/example_design
      Verilog and VHDL design files
    - <component_name>/implement
      Implementation script files
      - implement/results
        Results directory, created after implementation scripts are run, and contains implement script results
    - <component_name>/simulation
      Simulation scripts
      - simulation/functional
        Functional simulation files

# Directory and File Contents

The core directories and their associated files are defined in the following sections.

## <project directory>

The project directory contains all the CORE Generator tool project files.

*Table 10-1:* **Project Directory**

| Name | Description |
|---|---|
| <project_dir> | |
| <component_name>.ngc | A binary Xilinx implementation netlist. Describes how the core is to be implemented. Used as an input to the Xilinx implementation tools. |
| <component_name>.v[hd] | VHDL or Verilog structural simulation model. File used to support functional simulation of a core. |
| <component_name>.xco | As an output file, the XCO file is a log file which records the settings used to generate a particular core. An XCO file is generated by the CORE Generator tool for each core that it creates in the current project directory. An XCO file can also be used as an input to the CORE Generator tool. |
| <component_name>_flist.txt | List of files delivered with the core |
| <component_name>.{veo\|vho} | A VHDL or Verilog template for the core. This can be copied into your design. |

Back to Top

## <project directory>/<component name>

The <component name> directory contains the release notes file provided with the core, which can include last-minute changes and updates.

*Table 10-2:* **Component Name Directory**

| Name | Description |
|---|---|
| <project_dir>/<component_name> | |
| ten_gig_eth_pcs_pma_readme.txt | Core release notes file |

Back to Top

## <component_name>/doc

The doc directory contains the PDF documentation provided with the core.

*Table 10-3:* **Doc Directory**

| Name | Description |
|---|---|
| <project_dir>/<component_name>/doc | |
| ten_gig_eth_pcs_pma_ds739.pdf | 10GBASE-R/KR Data Sheet |
| ten_gig_eth_pcs_pma_ug692.pdf | 10GBASE-R/KR User Guide |

Back to Top

## <component_name>/example_design

The example design directory contains the example design files provided with the core.

*Table 10-4:* **Example Design Directory**

| Name | Description |
|---|---|
| <project_dir>/<component_name>/example_design | |
| <component_name>_block.v[hd] | Block entity containing the 10GBASE-R/KR core and transceiver wrappers |
| <component_name>_example_design.v[hd] | Top-level entity for the example design containing the block level design and clocking circuitry |
| <component_name>_example_design.ucf [a] | User constraints file for the core and example design |
| <component_name>_mod.v | Wrapper file for the 10GBASE-R/KR core |
| <component_name>_management_arbiter.v[hd] (Virtex-6 FPGAs only) | Arbiter for multiple cores accessing GTH_QUAD |
| <component_name>_legacy_config_shim.v[hd] (Virtex-6 FPGAs only) | Used to reproduce the original core configuration vector bit numbering |
| <component_name>_legacy_status_shim.v[hd] (Virtex-6 FPGAs only) | Used to reproduce the original core status vector bit numbering |

Back to Top

a. Only generated for BASE-KR cores when an Evaluation or Full license is available

## Virtex-7/Kintex-7 FPGAs

Only one of the following directories appears, depending on whether the selected device contains GTXE2 or GTHE2 transceivers.

### <component_name>/example_design/gtx

The gtx directory contains the example GTX transceiver wrappers which are provided with the core.

*Table 10-5:* **GTX Directory**

| Name | Description |
|---|---|
| <project_dir>/<component_name>/example_design/gtx | |
| <component_name>_gt_usrclk_source.v[hd]<br><component_name>_gtwizard_10gbaser.v[hd]<br><component_name>_gtwizard_10gbaser_gt.v[hd]<br><component_name>_gtwizard_10gbaser.xco | Wrappers and support files for the transceivers.<br>These can be regenerated from the latest GT_Wizard in the CORE Generator tool, using the 10GBASE-R/KR Protocol and the component name '<component_name>_gtwizard_10gbaser'<br>You can use the xco file provided to regenerate the latest version of the transceiver wrappers.<br>When regenerating the GT_Wizard output, pay close attention to the gt_usrclk_source block generated. This needs to be altered to include dclk generation from the reference clock, as illustrated in the example file provided with the core. |

Back to Top

### <component_name>/example_design/gth

The gth directory contains the example GTH transceiver wrappers which are provided with the core.

*Table 10-6:* **GTH Directory**

| Name | Description |
|---|---|
| <project_dir>/<component_name>/example_design/gth | |
| <component_name>gtwizard_gth_10gbaser__gt_usrclk_source.v[hd]<br><component_name>_gtwizard_gth_10gbaser.v[hd]<br><component_name>_gtwizard_gth_10gbaser_gt.v[hd]<br><component_name>_gtwizard_gth_10gbaser.xco | Wrappers and support files for the transceivers.<br>These can be regenerated from the latest GT_Wizard in the CORE Generator tool, using the 10GBASE-R/KR Protocol and the component name '<component_name>_gtwizard_10gbaser'<br>You can use the xco file provided to regenerate the latest version of the transceiver wrappers.<br>When regenerating the GT_Wizard output, pay close attention to the gt_usrclk_source block generated. This needs to be altered to include dclk generation from the reference clock, as illustrated in the example file provided with the core. |

Back to Top

## Virtex-6 FPGAs

### <component_name>/example_design/gth

The gth directory contains the example GTH transceiver wrappers which are provided with the core.

*Table 10-7:* **GTH Directory**

| Name | Description |
|---|---|
| <project_dir>/<component_name>/example_design/gth | |
| <component_name>_v6gth_wrapper.v[hd]<br><component_name>_v6gth_wrapper_quad.v[hd]<br><component_name>_v6gth_wrapper_gth_init.v[hd]<br><component_name>_v6gth_wrapper_gth_reset.v[hd]<br><component_name>_v6gth_wrapper_gth_rx_pcs_cdr_reset.v[hd]<br><component_name>_v6gth_wrapper_gth_tx_pcs_reset.v[hd]<br><component_name>_v6gth_wrapper_pulse_synchronizer.v[hd]<br><component_name>_v6gth_wrapper.xco | Wrappers and support files for the transceivers<br>These can be (re-)generated at any time from the latest Virtex-6 FPGA GTH Wizard, using '<component_name>_v6gth_wrapper' as the component name. You can use the xco file provided to regenerate the latest version of the GTH transceiver wrappers. |

Back to Top

### <component_name>/implement

This directory contains the support files necessary for implementation of the example design with the Xilinx tools. Execution of an implement script creates a results directory and an `xst` project directory.

*Table 10-8:* **Implement Directory**

| Name | Description |
|---|---|
| <project_dir>/<component_name>/implement | |
| implement.bat | Windows batch file that process the example design through the Xilinx tool flow |
| implement.sh | Linux shell script that processes the example design through the Xilinx tool flow |
| xst.scr | XST script file for the example design |
| xst.prj | XST project file for the example design |

Back to Top

### implement/results

This directory is created by the implement scripts and is used to run the example design files and the `<component_name>.ngc` file through the Xilinx implementation tools. On completion of an implement script, this directory contains the following files which can be used for timing simulation. Output files from the Xilinx implementation tools can also be found in this directory.

*Table 10-9:* **Results Directory**

| Name | Description |
|---|---|
| <project_dir>/<component_name>/implement/results | |
| routed.v[hd] | The back-annotated SIMPRIM-based VHDL or Verilog design. Can be used for timing simulation. |
| routed.sdf | Timing information for simulation |

Back to Top

### <component_name>/simulation

The simulation directory and the subdirectories below it contain the files necessary to test a VHDL or Verilog implementation of the example design.

*Table 10-10:* **Simulation Directory**

| Name | Description |
|---|---|
| <project_dir>/<component_name>/simulation | |
| demo_tb.v[hd] | The VHDL or Verilog demonstration test bench for the 10GBASE-R/KR core |

Back to Top

## simulation/functional

The functional directory contains functional simulation scripts provided with the core.

*Table 10-11:* **Functional Directory**

| Name | Description |
|---|---|
| <project_dir>/<component_name>/simulation/functional | |
| simulate_mti.do | ModelSim macro file that compiles the example design sources, the structural simulation model and the demonstration test bench then runs the functional simulation to completion. |
| simulate_ncsim.sh | Linux shell script that compiles the example design sources and the structural simulation model then runs the functional simulation to completion using the Cadence Incisive Enterprise Simulator (IES) simulator. |
| simulate_vcs.sh (verilog only) | Linux shell script that compiles the example design sources and the structural simulation model then runs the functional simulation to completion using Verilog Compiled Simulator (VCS). |
| ucli_commands.key (verilog only) | VCS command file. This file is called by the simulate_vcs.sh script. |
| vcs_session.tcl (verilog only) | VCS DVE tcl script that opens wave windows and adds interesting signals to it. This macro is used by the simulate_vcs.sh script. |
| wave_mti.do | ModelSim macro file that opens a wave window and adds interesting signals to it. This macro is called by the simulate_mti.do macro file. |
| wave_ncsim.sv | The Cadence IES simulator macro file that opens a wave windows and adds interesting signals to it. This macro is called by the simulate_ncsim.sh script. |

Back to Top

# Implementation and Test Scripts

## Implementation Script

The implementation script is either a shell script or batch file that processes the example design through the Xilinx tool flow. The script is located at:

### Linux

```
<project_dir>/<component_name>/implement/implement.sh
```

### Windows

```
<project_dir>/<component_name>/implement/implement.bat
```

The implement script performs these steps:

- The example HDL wrapper is synthesized using XST.
- ngdbuild is run to consolidate the core netlist and the wrapper netlist into the NGD file containing the entire design.
- The design is mapped to the target technology.
- The design is place-and-routed on the target device.
- Static timing analysis is performed on the routed design using trce.
- A bitstream is generated.
- netgen runs on the routed design to generate VHDL and Verilog netlists and timing information in the form of SDF files.

Virtex®-7/Kintex™-7 FPGAs: The implement script is only generated for a BASE-KR core when an Evaluation or Full license is available for the core.

## Setting up for Simulation

The Xilinx UNISIM library must be mapped into the simulator. If the library is not set up for your environment, go to Answer Record 15338 for assistance compiling Xilinx simulation models and for setting up the simulator environment.

All Virtex® FPGA designs require a Verilog LRM-IEEE 1364-2005 encryption-compliant simulator. For a Verilog LRM-IEEE 1364-2005 encryption-compliant simulator, the following simulators are supported.

- Mentor Graphics ModelSim version 10.1a
- Cadence Incisive Enterprise Simulator v11.1
- Synopsys VCS and VCS MX 2011.12

## Simulation Scripts

Simulation macro files are provided for ModelSim and shell scripts are provided for the Cadence IES simulator and Synopsys VCS simulator. The scripts automate the simulation of the test bench and can be found in the following location:

### Functional

`<project_dir>/<component_name>/simulation/functional/simulate_mti.do`

`<project_dir>/<component_name>/simulation/functional/simulate_ncsim.sh`

`<project_dir>/<component_name>/simulation/functional/simulate_vcs.sh`

The scripts perform these tasks:

- Compiles the gate level netlist
- Compiles the demonstration test bench
- Starts a simulation of the test bench
- Opens a Wave window and adds some interesting signals (`wave_mti.do/wave_ncsim.sv/vcs_session.tcl`)
- Runs the simulation to completion

### Libraries

The user must ensure that any required simulation support files are created in the simulation/functional directories. These can include library-directory mappings such as those contained in the modelsim.ini file for ModelSim and cds.lib and hdl.var files for IES (NCSIM).

# 10GBASE-R/KR Core

## Example HDL Wrapper - Virtex-7/Kintex-7 FPGAs

In Figure 10-1, the example HDL wrapper generated contains the following:

- The block-level instance containing the core, GT_USRCLK_SOURCE and GTWIZARD_10GBASER/GTWIZARD_GTH_10GBASER blocks
- Reset synchronizer registers
- User clock generation
- Double Data Rate (DDR) register on xgmii_rx_clk
- 10GBASEKR 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)

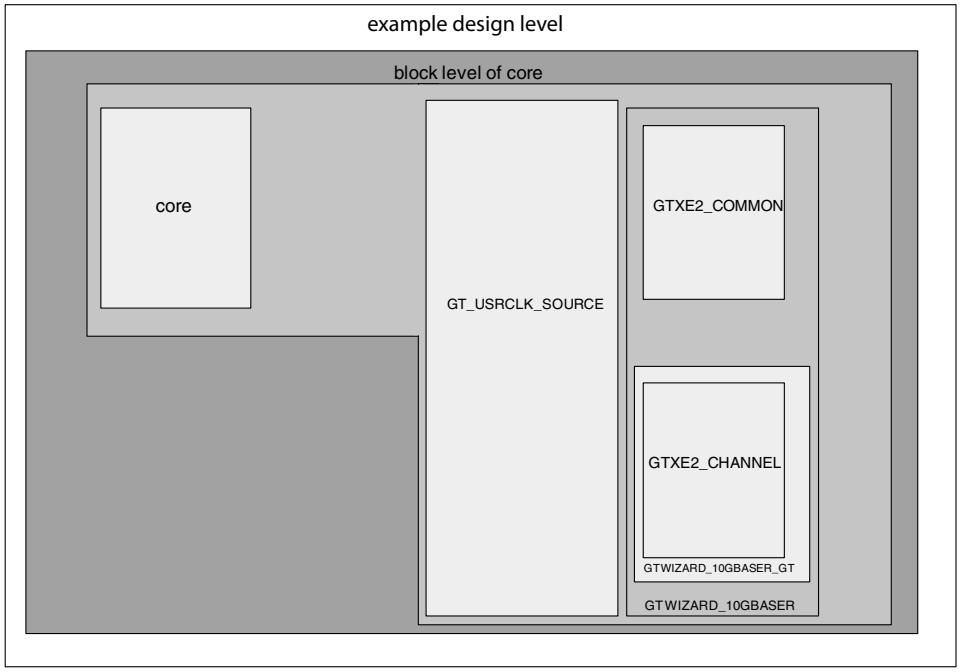


*Figure 10-1:* **Example HDL Wrapper for 10GBASE-R (Virtex-7/Kintex-7 FPGAs)**

## Example HDL Wrapper (Virtex-6 FPGAs)

In Figure 10-2, the example HDL wrapper generated contains the following:

- The block-level instance containing the core, clock buffers and the transceiver
- Clock Buffer instance for DCLK
- Clock buffer for GTH transceiver reference clock
- DDR register on xgmii_rx_clk

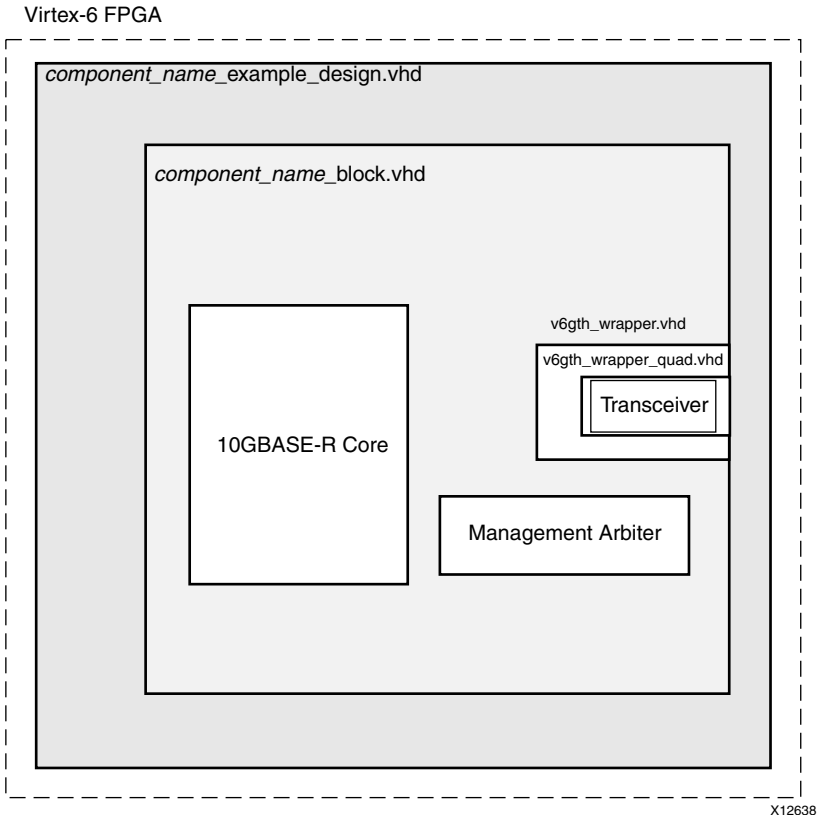


*Figure 10-2:* **Example HDL Wrapper for 10GBASE-R/KR (Virtex-6 FPGAs)**

## Demonstration Test Bench

In Figure 10-3, 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.

When the DUT is generated without the MDIO interface, a change from the previous version of the core is to only connect the critical parts of the configuration and status vectors to suitably-named I/O signals, in the example design level. Also, for Virtex-6 FPGA designs, the numbering of bits on the configuration and status vectors has changed. The rtl files legacy_config_shim and legacy_status_shim can be used to replicate the original bit numbering.

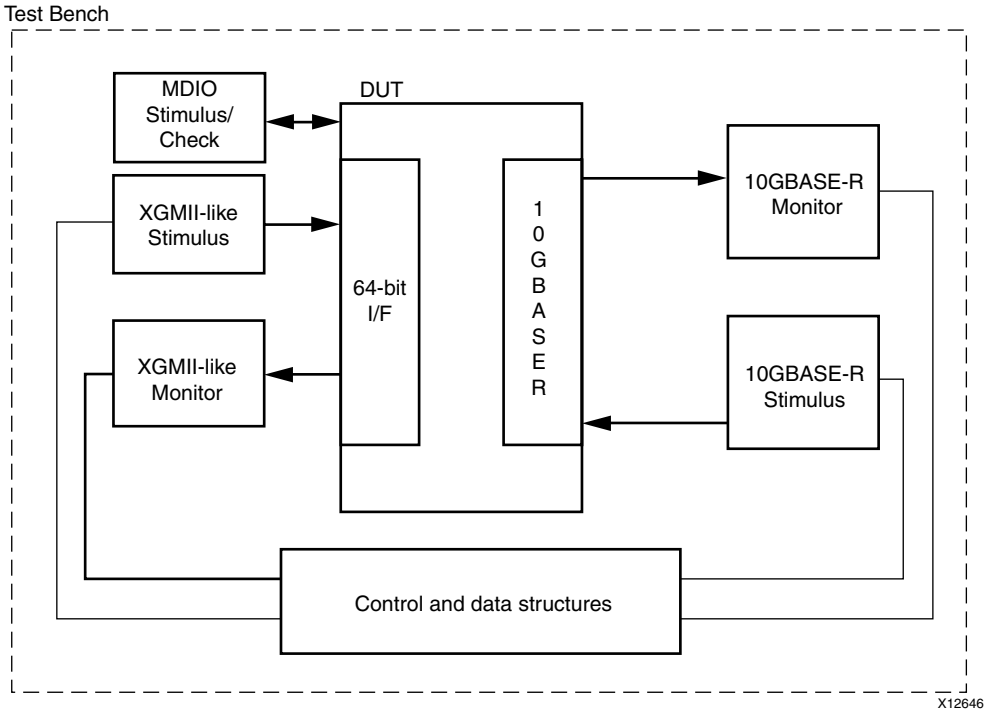


*Figure 10-3:* **Demonstration Test Bench for 10GBASE-R**

Chapter 11

# Quick Start Example Design

This chapter provides instructions for generating a core using the default configuration, implementing the example design, and simulating your design using Mentor Graphics ModelSim version 10.1a, Cadence Incisive Enterprise Simulator (IES) v11.1, and Synopsys VCS and VCS MX 2011.12.

## Introduction

Figure 11-1 illustrates the default configuration of the example design for Virtex-7/Kintex-7 FPGAs.

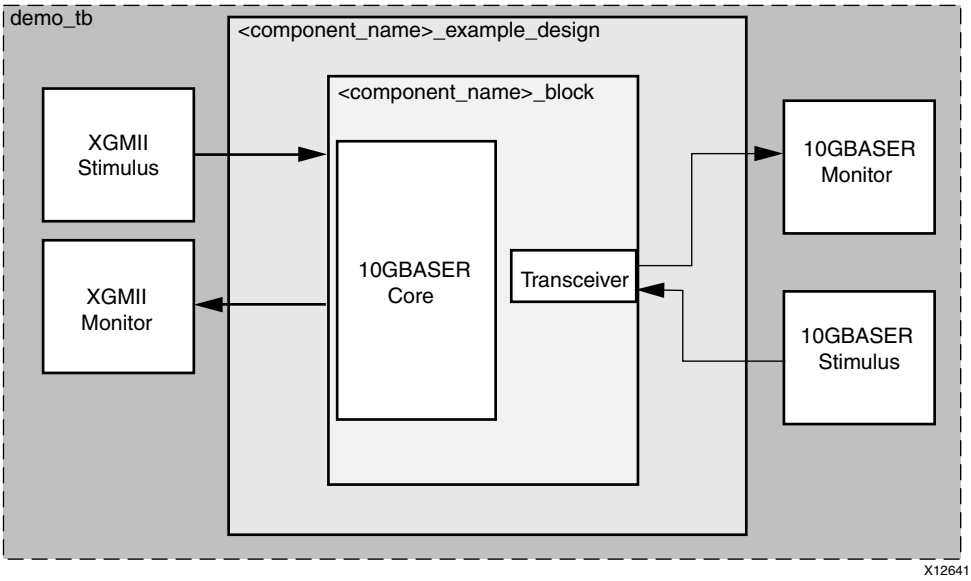


*Figure 11-1:* **Virtex-7/Kintex-7 FPGA 10GBASE-R/KR Example Design and Test Bench**

The 10GBASE-R/KR example design consists of the following:

- A 10GBASE-R/KR core netlist
- Transceiver wrappers
- An example HDL wrapper
- A demonstration test bench to exercise the example design

The 10GBASE-R/KR Design Example has been tested with Xilinx® ISE® tools v14.1, ModelSim version 10.1, Cadence Incisive Enterprise Simulator (IES) v11.1, and Synopsys VCS and VCS MX 2011.12.

Figure 11-2 illustrates the default configuration of the example design for Virtex-6 HXT FPGAs.

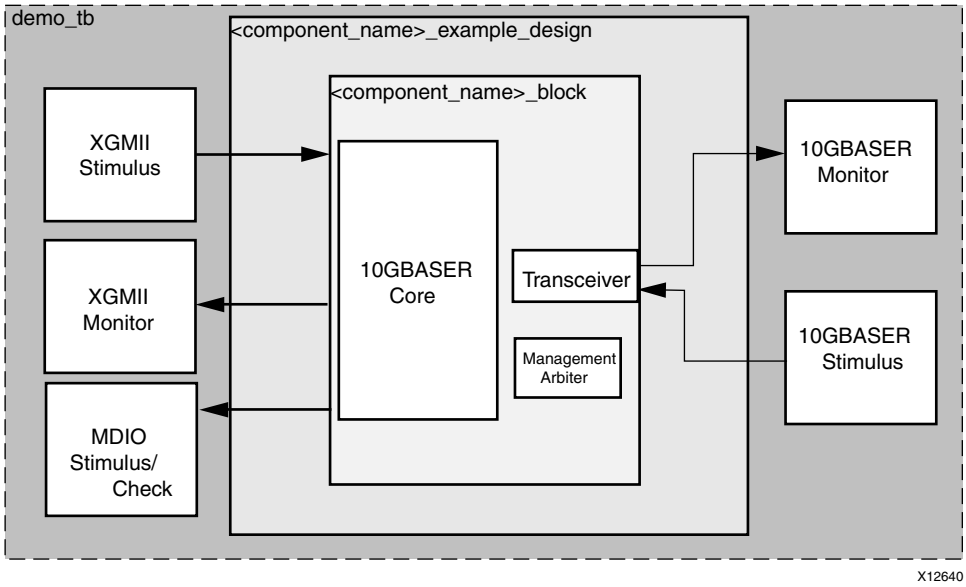


*Figure 11-2:* **Virtex-6 FPGA 10GBASE-R/KR Example Design and Test Bench with MDIO**

# Generating the Core

To generate a 10GBASE-R/KR core with default values using the CORE Generator™ tool do the following:

1. Start the CORE Generator tool.

   For help starting and using the CORE Generator tool, see the documentation supplied with the ISE® tools.

2. Choose File > New Project.
3. Type a directory name.
4. Perform these steps to set project options:
   a. From the Part tab, select a silicon family, part, speed grade, and package that supports the 10GBASE-R/KR core, for example, Virtex®-6 FPGAs.

      ***Note:*** If an unsupported silicon family is selected, the 10GBASE-R/KR core does not appear in the taxonomy tree. For a list of supported architectures, see the *10GBASE-R/KR Data Sheet*.

   b. From the Generation tab, select VHDL or Verilog; for Vendor, select Other.
   c. On the Advanced tab, accept the default values.
5. After creating the project, locate the core in the taxonomy tree at the left side of the CORE Generator tool window. The 10GBASE-R/KR core appears under these categories:
   - Communications & Networking/Ethernet
   - Communications & Networking/Networking
   - Communications & Networking/Telecommunications

6. Double-click the core to open it. A message can appear warning you about the limitations of the Simulation Only license, and then the 10GBASE-R/KR customization screen appears.
7. In the Component Name field, enter a name for the core instance.
8. Accept the remaining default options and click Generate to generate the core.

   The core and its supporting files, including the example design, are generated in the project directory. For a detailed description of the directory structure and files, see Detailed Example Design in Chapter 10.

# Implementing the 10GBASE-R/KR Example Design

After the core is successfully generated, the netlist and example design HDL wrapper can be processed through the Xilinx implementation tools. The generated outputs include several scripts to assist in processing.

Open a command prompt or shell in your project directory and enter the following commands:

## Linux

```
% cd <component_name>/implement
% source ./implement.sh
```

## Windows

```
> cd <component_name>\implement
> implement.bat
```

The implement command accomplishes the following:

- Starts a script to synthesize the example design HDL wrapper
- Builds, maps, and place-and-routes the example design (Full license only)
- Creates gate-level netlist HDL files with associated timing information (SDF files)

The created files are placed in the results directory which is created by the implement script at run time.

# Simulating the 10GBASE-R/KR Example Design

The example design provided with the 10GBASE-R/KR core provides a complete environment which allows you to simulate the core and view the outputs. Scripts are provided for pre- and post-implementation simulation. The simulation model is either in VHDL or Verilog depending on the CORE Generator tool Design Entry project option.

## Setting up for Simulation

To run the gate-level simulation you must have the Xilinx Simulation Libraries compiled for your system. See the Compiling Xilinx Simulation Libraries (COMPXLIB) in the *Xilinx ISE Synthesis and Verification Design Guide*, and the *Xilinx ISE Software Manuals and Help*. You can download these documents from: www.xilinx.com/support/software_manuals.htm.

The Xilinx simulation libraries must be mapped into the simulator. If the libraries are not set for your environment, go to Answer Record 15338 on www.xilinx.com/support for assistance compiling Xilinx simulation models and setting up the simulator environment.

All Virtex device designs require a Verilog LRM-IEEE 1364-2005 encryption-compliant simulator. For a Verilog LRM-IEEE 1364-2005 encryption-compliant simulator, these simulators are supported.

- Mentor Graphics ModelSim version 10.1a
- Cadence Incisive Enterprise Simulator v11.1
- Synopsys VCS and VCS MX 2011.12

### Pre-Implementation Simulation

To run a functional simulation of the example design:

1. Open a command prompt or shell in your project directory and set the current directory to

   ```
   <component_name>/simulation/functional
   ```

2. Launch the simulation script:

   ```
   ModelSim: vsim -do simulate_mti.do
   Cadence sim: ./simulate_ncsim.sh
   vcs: ./simulate_vcs.sh
   ```

The simulation script compiles the functional model and the demonstration test bench, adds some relevant signals to a wave window, and then runs the simulation to completion. You can then inspect the simulation transcript and waveform to observe the operation of the core.

### Post-Implementation Simulation (For Production Silicon Only)

To run a timing simulation of the example design:

1. Open a command prompt or shell in your project directory, then set the current directory to:

   ```
   <component_name>/simulation/timing
   ```

2. Launch the simulation script:

   ```
   ModelSim: vsim -do simulate_mti.do
   Cadence sim: ./simulate_ncsim.sh
   vcs: ./simulate_vcs.sh
   ```

The simulation script compiles the gate-level model and the demonstration test bench, adds some relevant signals to a wave window, and then runs the simulation to completion. You can then inspect the simulation transcript and waveform to observe the operation of the core.

## Additional Information

For details about the example design, including guidelines for modifying the design and extending the test bench, see Chapter 10, Detailed Example Design.

Appendix A

# Additional Resources

## Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see the Xilinx Support website at:

www.xilinx.com/support.

For a glossary of technical terms used in Xilinx documentation, see:

www.xilinx.com/support/company/terms.htm

## 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 Ten Gigabit Ethernet PCA/PMA core is located at the Xilinx Ethernet IP Solution Center.

## Technical Support

Xilinx provides technical support at www.xilinx.com/support for this LogiCORE™ IP product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support of product if implemented in devices that are not defined in the documentation, if customized beyond that allowed in the product documentation, or if changes are made to any section of the design labeled DO NOT MODIFY.

See the IP Release Notes Guide (XTP025) for more information on this core. For each core, there is a master Answer Record that contains the Release Notes and Known Issues list for the core being used. The following information is listed for each version of the core:

- New Features
- Resolved Issues
- Known Issues

XILINX®

*Appendix B*

# *Verification and Interoperability*

The 10GBASE-R/KR core has been verified using 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
- Autonegotiation phase
- Training phase
- FEC with correctable and uncorrectable errors

The 10GBASE-R core has been validated on Virtex-6 devices at the University of New Hampshire Interoperability Lab.

XILINX®

*Appendix C*

# *Core 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.

## Virtex-7/Kintex-7 FPGAs

### 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 core for the internal XGMII interface configuration in the transmit direction is 20 periods of txclk322.

### Receive Path Latency

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 core in the receive direction is nominally equal to 28 cycles of rxclk322, including +/- 4 cycles in the elastic buffer. The latency depends on sync bit alignment position and data positioning within the transceiver 4-byte interface. When the optional FEC functionality is included in the core, this increases to a minimum of 36 cycles but may be a lot more if FEC errors are detected and corrected.

### Transceiver Latency

See the *7 Series Transceivers User Guide* (UG476) for information on the transceiver latency.

## Virtex-6 HXT FPGAs

### 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[63:0]` on the transceiver interface), the latency through the core for the internal XGMII interface configuration in the transmit direction is 2 periods of the core input `clk156`.

## Receive Path Latency

Measured from the input into the core on `gt_rxd[63:0]` until the data appears on `xgmii_rxd[63:0]` of the receiver side XGMII interface, the latency through the core in the receive direction is nominally equal to 10 clock cycles of `clk156`, +/- 4 cycles in the elastic buffer. The latency depends on sync bit alignment position and data positioning within the transceiver 4-byte interface.

## GTH Transceiver Latency

Latency through the GTH transceiver in the transmit direction is nominally 5 cycles of `clk156`.

Latency through the GTH transceiver in the receive direction is nominally 3 cycles of `rxclk156` plus a number of bit-times between zero and 65, in the RX Alignment Buffer.

## Total Latency

The total latency from `xgmii_tx` to `xgmii_rx` if the core is looped back at the serial ports is (1320 + 0..65) Bit Times (BT). This meets the IEEE specification in clause 49.3.6.4 of a maximum of 3586 BT.

With the Rx Elastic Buffer at its maximum fill level, the overall latency is (1584+0..65) BT.