LogiCORE IP 10-Gigabit Ethernet PCS/PMA v2.6

Product Guide

PG068 December 18, 2012

Table of Contents

SECTION I: SUMMARY

Introduction

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

10GBASE-KR and 10GBASE-R are supported on Virtex®-7 and Kintex™-7 devices on GTH and GTX transceivers. Xilinx also supports an integrated 10GBASE-R IP on Virtex-6 HXT devices. 10GBASE-KR for backplane applications has additional features over 10GBASE-R which consist of Link Training and optional Forward Error Correction (FEC) and Auto-negotiation (AN).

Features

- Designed to 10-Gigabit Ethernet specification IEEE 802.3-2008 clause 49, 72, 73, 74
- Optional Management Data Interface (MDIO) interface to manage PCS/PMA registers according to specification IEEE 802.3-2008 clause 45
- Delivered through the Xilinx CORE Generator™ and Vivado™ IP catalog tools
- Supports 10GBASE-SR, -LR and -ER optical links in Virtex-7, Kintex-7 and Virtex-6 devices (LAN mode only)
- Supports 10GBASE-KR backplane links in Kintex-7/Virtex-7, including Auto-Negotiation (AN), Training and Forward Error Correction (FEC).
- SDR 10-Gigabit Ethernet Media Independent Interface (XGMII) connects seamlessly to the Xilinx 10G Ethernet MAC

LogiCORE IP Facts Table	
Core Specifics	
Supported Device Family[1]	Virtex-7, Kintex-7[2], Virtex-6 HXT
Supported User Interfaces	XGMII
Resources	See Table 2-1 through Table 2-3.
Provided with Core	
Design Files	ISE®: Native Generic Circuit (NGC) Netlist Vivado™: Encrypted RTL
Example Design	Verilog and VHDL
Test Bench	Verilog and VHDL
Constraints File	ISE: Xilinx Constraints File Vivado: Xilinx Design Constraint (XDC)
Simulation Model	Verilog or VHDL Structural Model
Supported S/W Driver	N/A
Tested Design Flows[3]	
Design Entry	ISE Design Suite v14.4 Vivado Design Suite v2012.4[4]
Simulation	Mentor Graphics ModelSim Cadence Incisive Enterprise Simulator (IES) Synopsys VCS and VCS MX
Synthesis	XST Vivado Synthesis
Support	
Provided by Xilinx @ www.xilinx.com/support	

Notes:

1. For a complete listing of supported devices, see the release notes for this core.
2. -2, -2L or -3.
3. For the supported versions of the tools, see the Xilinx Design Tools: Release Notes Guide.
4. Supports only 7 series devices.

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).

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.

10GBASE-R

For Virtex-7/Kintex™-7 devices, all of the PCS and Management blocks illustrated are implemented in logic, except for part of the Gearbox and SerDes. Figure 1-1 shows the architecture.

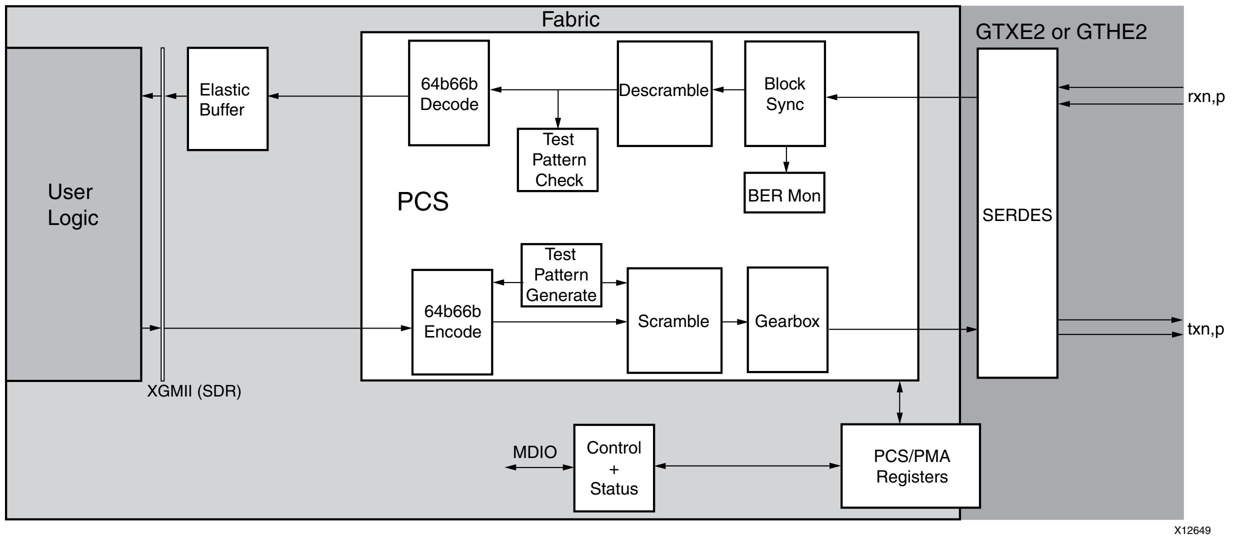


Figure 1-1: **Virtex-7/Kintex-7 Implementation of the 10-Gigabit Ethernet PCS/PMA (BASE-R) Core**

The major functional blocks of the core include the following:

- XGMII interface, designed for simple attachment of 10-Gigabit Ethernet MAC
- Transmit path, including Scrambler, 64B/66b Encoder and Gearbox
- Receive path, including Block Synchronization, Descrambler, Decoder and BER (Bit Error Rate) monitor
- Test Pattern Generation and Checking
- Serial interface to optics
- Management registers (PCS/PMA) with optional MDIO interface

Figure 1-2 illustrates a block diagram of the 10-Gigabit Ethernet PCS/PMA (BASE-R) core implementation on Virtex®-6 devices. As you can see, in Virtex-6 devices, most of the functionality is contained within the GTHE1 transceiver.

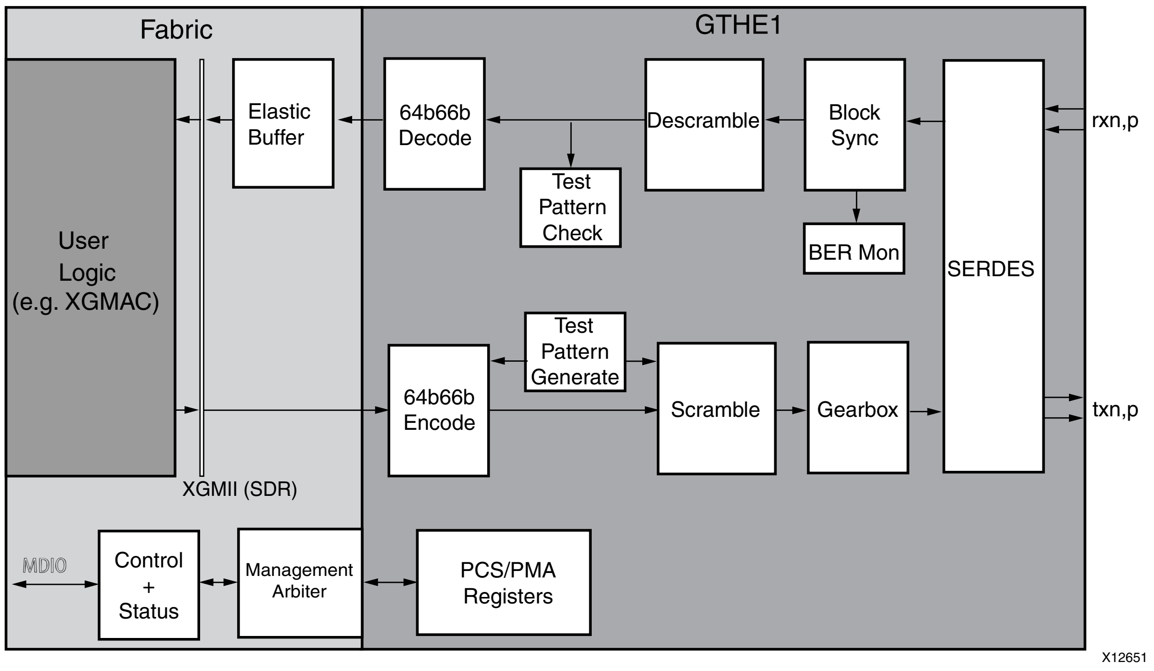


Figure 1-2: **Virtex-6 Implementation of the BASE-R Core**

10GBASE-KR

Figure 1-3 illustrates a block diagram of the 10-Gigabit Ethernet PCS/PMA (BASE-KR) core implementation. The major functional blocks of the core include the following:

- XGMII interface, designed for simple attachment of 10-Gigabit Ethernet MAC
- Transmit path, including Scrambler, 64B/66B Encoder, FEC, AN and Training
- Receive path, including Block Synchronization, Descrambler, Decoder and BER (Bit Error Rate) monitor, FEC, AN and Training
- Test Pattern Generation and Checking
- Serial interface to backplane connector
- Management registers (PCS/PMA) with optional MDIO interface

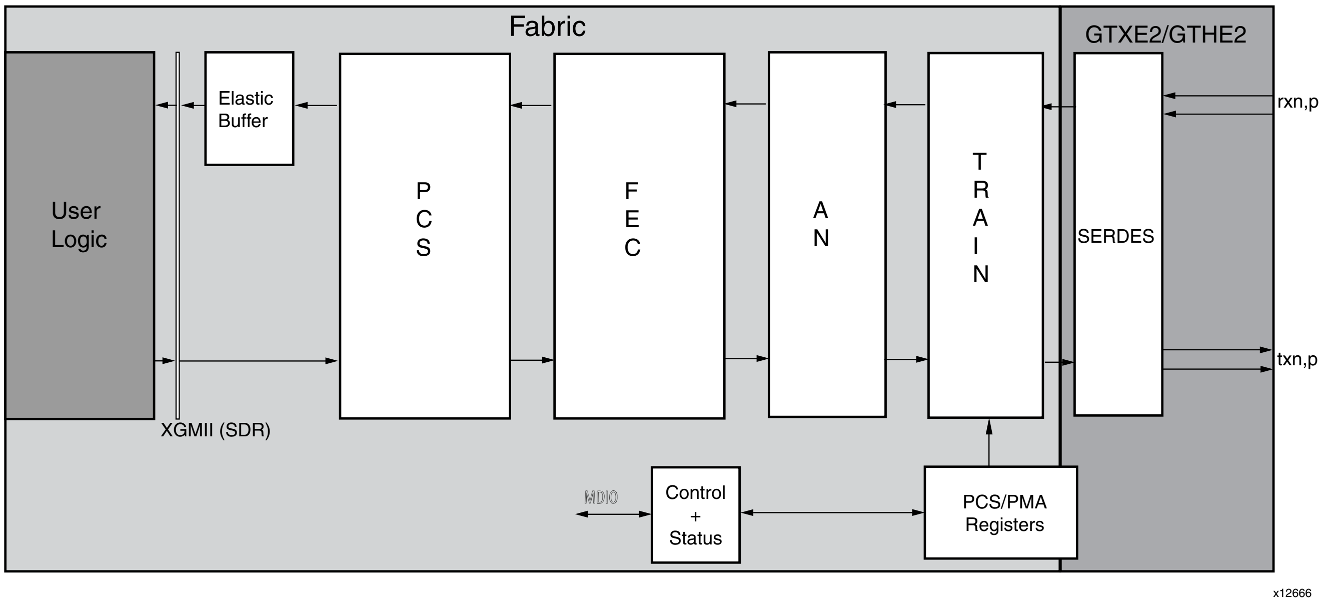


Figure 1-3: **Virtex-7/Kintex-7 Implementation of the BASE-KR Core**

System Requirements

For a list of System Requirements, see the Xilinx Design Tools: Release Notes Guide.

Recommended Design Experience

Although the 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 Field Programmable Gate Array (FPGA) designs using Xilinx implementation software and User Constraints File (UCF) or Xilinx Design Constraints (XDC) is recommended.

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

Applications

Figure 1-4 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 [Ref 1][Ref 2].

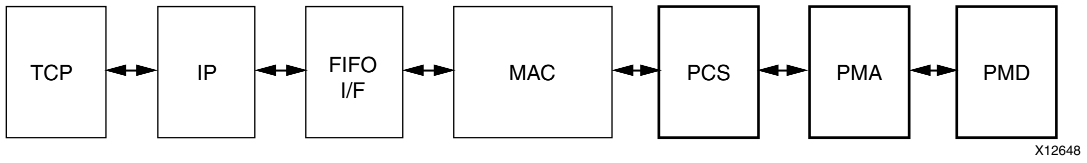


Figure 1-4: **Typical Ethernet System Architecture**

Figure 1-5 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 (BASE-R) or backplane (BASE-KR) using a serial interface. The optional WAN Interface Sublayer (WIS) part of the 10GBASE-R standard is not implemented in this core.

The 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 3, Designing with the Core.

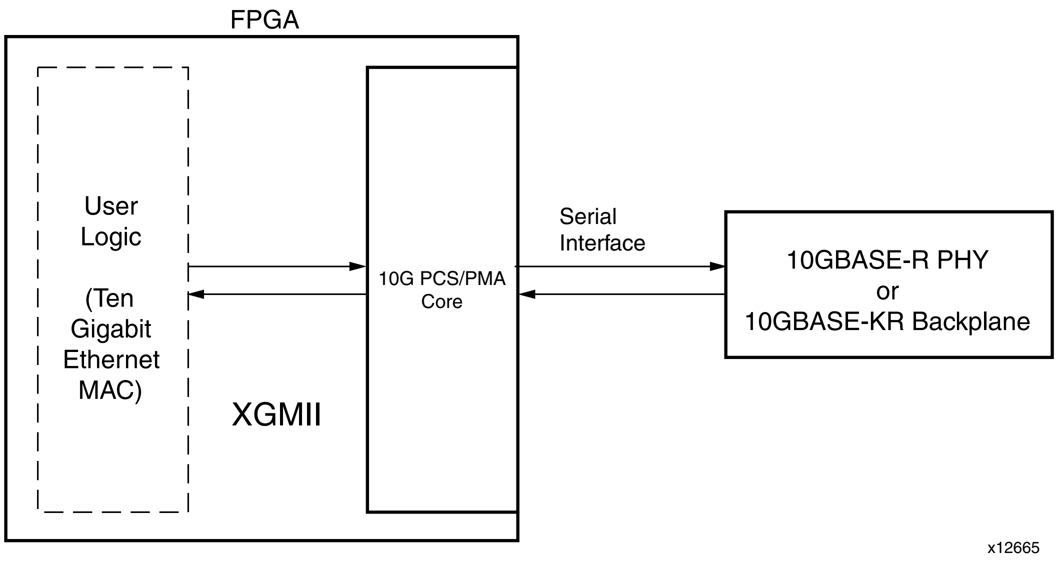


Figure 1-5: **Core Connected to MAC Core Using XGMII Interface**

Unsupported Features

While the Training Protocol is supported natively by the core, no logic is provided that controls the far-end transmitter adaptation based on analysis of the received signal quality.

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

Licensing and Ordering Information

The section contains the following subsections.

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

10GBASE-R PCS/PMA

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

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

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

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

For more information, visit the 10 Gigabit Ethernet PCS/PMA with FEC/Auto-Negotiation (10GBASE-KR) product web page.

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

Product Specification

Standards

The 10GBASE-R/KR core is designed to the standard specified in clauses 45, 49, 72, 73 and 74 of the 10-Gigabit Ethernet specification IEEE Std. 802.3-2008.

Performance

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`. When the optional FEC functionality is included in the core, this increases to 26 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 by 70 cycles of `rxclk322` and if error reporting to the PCS layer is enabled, there will be an extra 66 cycles of `rxclk322` latency.

Transceiver Latency

See *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.

Resource Utilization

Resources required for the core have been estimated for the Virtex®-7, Kintex™-7, and Virtex-6 FPGAs (Table 2-1, Table 2-2, and Table 2-3. These values were generated using Xilinx CORE Generator™ tools, v14.4. They are derived from post-synthesis reports, and might change during MAP and PAR.

Virtex-7/Kintex-7 FPGAs

Table 2-1 provides approximate slice counts for the BASE-R options on Virtex-7 and Kintex-7 FPGAs.

Table 2-1: **Device Utilization - BASE-R on Virtex-7/Kintex-7 FPGAs**

Parameter Values	Device Resources		
MDIO Interface	**Slices**	**LUTs**	**FFs**
No	952	2160	2281
Yes	1171	2736	2708

Table 2-2 provides the approximate slice counts for the BASE-KR options on Virtex-7 and Kintex-7 FPGAs.

Table 2-2: **Device Utilization on Virtex-7/Kintex-7 FPGAs**

Parameter Values			Device Resources		
FEC	**Auto-Negotiation**	**MDIO Interface**	**Slices**	**LUTs**	**FFs**
No	No	No	1549	3318	3494
No	No	Yes	1709	3745	3734
No	Yes	No	1797	3815	4100
No	Yes	Yes	1972	4504	4517
Yes	No	No	2854	6899	5176
Yes	No	Yes	2886	7372	5446
Yes	Yes	No	3343	7465	5806
Yes	Yes	Yes	3443	8191	6200

Virtex-6 HXT FPGAs

Table 2-3 provides approximate slice counts for the two BASE-R core options on Virtex-6 HXT FPGAs.

Table 2-3: **Device Utilization - Virtex-6 HXT FPGAs**

Parameter Values	Device Resources		
MDIO Interface	**Slices**	**LUTs**	**FFs**
No	446	877	1019
Yes	296	687	770

Resource Utilization for Example Design Using Vivado Design Suite

These resource numbers are created for the core example design and are not directly comparable to the ISE tools resource numbers which are for the core-only.

Table 2-4: **10GBASE-R**

Parameter	Slices	LUTs	FFs
MDIO = NO	1121	2724	2427
MDIO = YES	1103	3351	2758

Table 2-5: **10GBASE-KR — Parameter (AN = Auto Negotiation)**

FEC	AN	MDIO	Slices	LUTs	FFs
N	N	N	1752	3887	3654
N	N	Y	1593	4575	3876
N	Y	N	2232	4350	4606
N	Y	Y	1871	5298	4950
Y	N	N	3391	8896	5443
Y	N	Y	2959	8997	5285
Y	Y	N	3589	8651	5936
Y	Y	Y	3245	9654	6287

Port Descriptions

MAC-Side Interface: XGMII

The MAC (or client) side of the core has a 64-bit datapath plus 8 control bits implementing an XGMII interface. Table 2-6 defines the signals, which are all synchronous to the 156.25 MHz core clock. It is designed to be connected to either user logic within the FPGA or, by using SelectIO™ technology Double Data Rate (DDR) registers in your own design top-level, to provide an external 32-bit 312.5 MHz DDR XGMII, defined in clause 46 of *IEEE 802.3-2008*.

Table 2-6: **MAC-Side Interface Ports**

Signal Name	Direction	Description
xgmii_txd[63:0]	In	64-bit transmit data word
xgmii_txc[7:0]	In	8-bit transmit control word
xgmii_rxd[63:0]	Out	64-bit receive data word
xgmii_rxc[7:0]	Out	8-bit receive control word

Figure 2-1 illustrates transmitting a frame through the client-side interface.

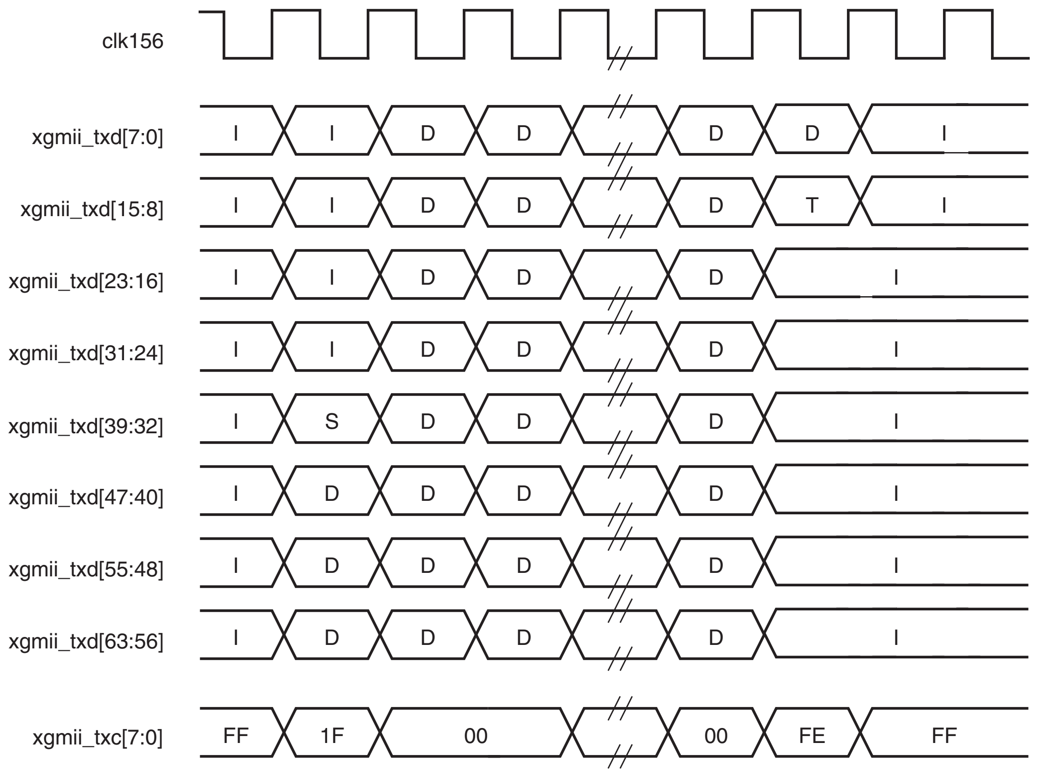


Figure 2-1: **Transmitting a Frame Through the Client-Side Interface**

Figure 2-2 illustrates receiving a frame through the client-side interface.

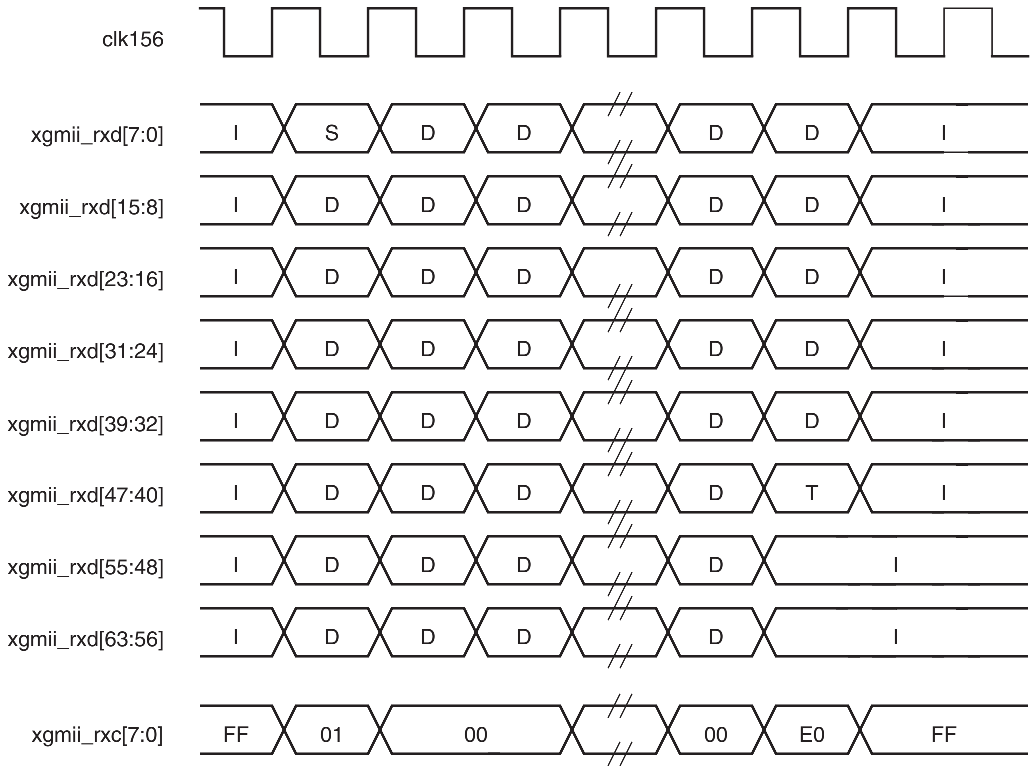


Figure 2-2: **Receiving a Frame Through the Client-Side Interface**

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 2-7. See Chapter 3, Designing with 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 2-7: **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

Table 2-7: **Transceiver Interface Ports - Virtex-7/Kintex-7 FPGA GTX/GTH Transceiver *(Cont'd)***

Signal Name	Direction	Description
gt_slip	Out	RXGEARBOXSLIP[a]
coeff_minus_1[4:0]	Out	Control of TXPRECURSOR for Link Training[bc]
coeff_plus_1[4:0]	Out	Control of TXPOSTCURSOR for Link Training[bc]
coeff_zero[6:0]	Out	Control of TXMAINCURSOR for Link Training[b]

a. Also drives RXSLIDE on transceiver during Training Protocol in 10GBASEKR cores.
b. 10GBASEKR cores only.
c. These two values indicate the scalar value of negative numbers.

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 2-8. See Chapter 3, Designing with the Core for details on connecting the device-specific transceivers to the 10GBASE-R core.

Table 2-8: **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 2-8. See Chapter 3, Designing with the Core for details on connecting an optical module to the 10GBASE-R core.

Table 2-9: **Optical Module Interface Ports**

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

a. This signal is 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.

Management Interface (MDIO)

The optional MDIO interface is a simple low-speed two-wire interface for management of the 10-Gigabit Ethernet PCS/PMA core, consisting of a clock signal and a bidirectional data signal. The interface is defined in clause 45 of the *IEEE 802.3-2008* standard.

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

For the BASE-R core in Virtex-6 FPGAs, the appropriate register in the GTHE1 transceiver is pre-read as soon as the address phase of the MDIO transfer is complete and this data is provided back to the MDIO interface on completion of the READ phase of the MDIO transfer.

Table 2-10: **MDIO Management Interface Ports**

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

GTHE1 Management Interface Ports

There is a management interface on the GTHE1 transceiver which connects to the associated ports on the core, through an arbiter block provided with the core.

Table 2-11: **GTHE1 Management Interface Ports**

Signal Name	Direction	Description
mgmt_req	Out	Request access to the MGMT interface on the GTHE1
mgmt_gnt	In	Access granted to the MGMT interface on the GTHE1
mgmt_rd_out	Out	Read enable
mgmt_wr_out	Out	Write enable
mgmt_addr_out[20:0]	Out	Address
mgmt_rdack_in	In	Read Acknowledge
mgmt_rddata_in[15:0]	In	Read data
mgmt_wrdata_out[15:0]	In	Write data

GTXE2/GTHE2 DRP Interface Ports

There is a DRP interface on the GTXE2/GTHE2 transceivers which connect to the associated ports on the core, perhaps through an arbiter block (not provided with the core).

Table 2-12: **GTXE2/GTHE2 DRP Interface Ports**

Signal Name	Direction	Description
drp_req[a]	Out	Request access to the DRP interface on the GTXE2/GTHE2
drp_gnt	In	Access granted to the DRP interface on the GTXE2/GTHE2
drp_den	Out	DRP enable
drp_dwe	Out	Write enable
drp_daddr[15:0]	Out	Address
drp_di[15:0]	In	Write data
drp_drdy	In	Read data ready/Write complete
drp_drpdo[15:0]	In	Read data

a. Can be wired directly to drp_gnt if this is the only block requiring access to the DRP interface.

Configuration and Status Signals

As an alternative to the MDIO interface, vector-based interfaces are provided to allow control and status to flow to and from the core. Table 2-13 describes these two vectors. Neither vector is completely populated so the actual number of pins required is much lower than the maximum widths of the vectors. For the status vector, correct default values are provided for all bits in the associated IEEE registers. Further details of these vectors can be found in Table 3-6 to Table 3-9.

Table 2-13: **Configuration and Status Vectors**

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

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 2-14 shows the ports on the netlist that are associated with system clocks and resets.

Table 2-14: **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, *must* run at exactly 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 2-15 shows the ports on the netlist that are associated with system clocks and resets.

Table 2-15: **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.

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 2-16 shows the ports on the netlist that are associated with that interface.

Table 2-16: **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. This should not be confused with the IEEE register 1.150.1 - Training Enable.
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 because all registers are exposed through the configuration and status vectors. This should be tied to '0' in that case. Access to transceiver DRP registers through the Training interface is unaffected.

Figure 2-3 and Figure 2-4 show the timing diagrams for Using the Training Interface to Access Internal Core Registers and Transceiver Registers through the DRP Port. As shown in Figure 2-3 and Figure 2-4, `training_drp_cs`, `training_ipif_cs`, and `training_enable` should be low between read or write accesses.

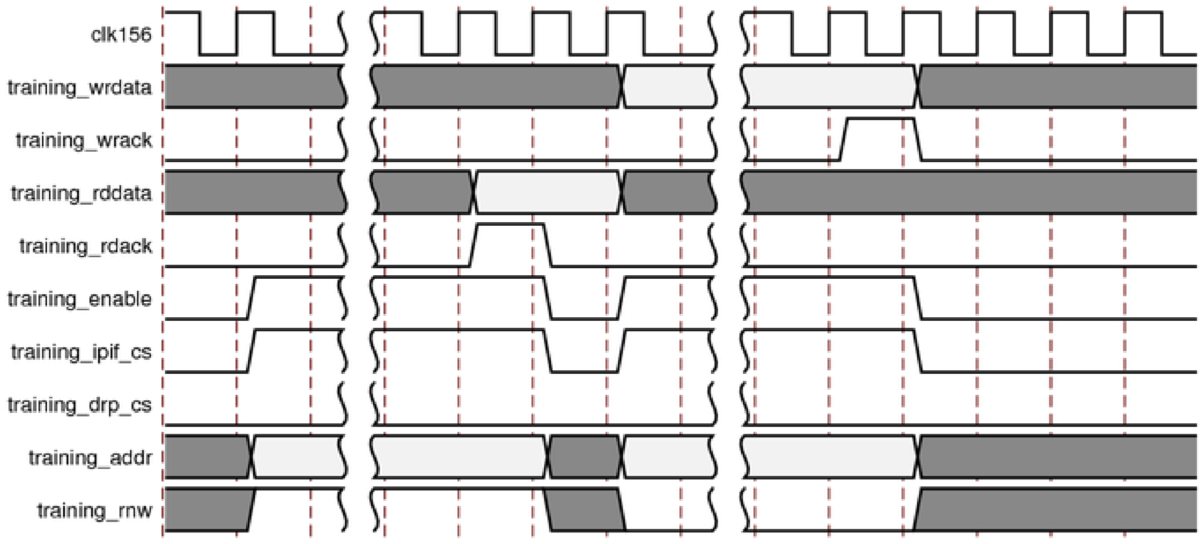


Figure 2-3: **Using the Training Interface to Access Internal Core Registers**

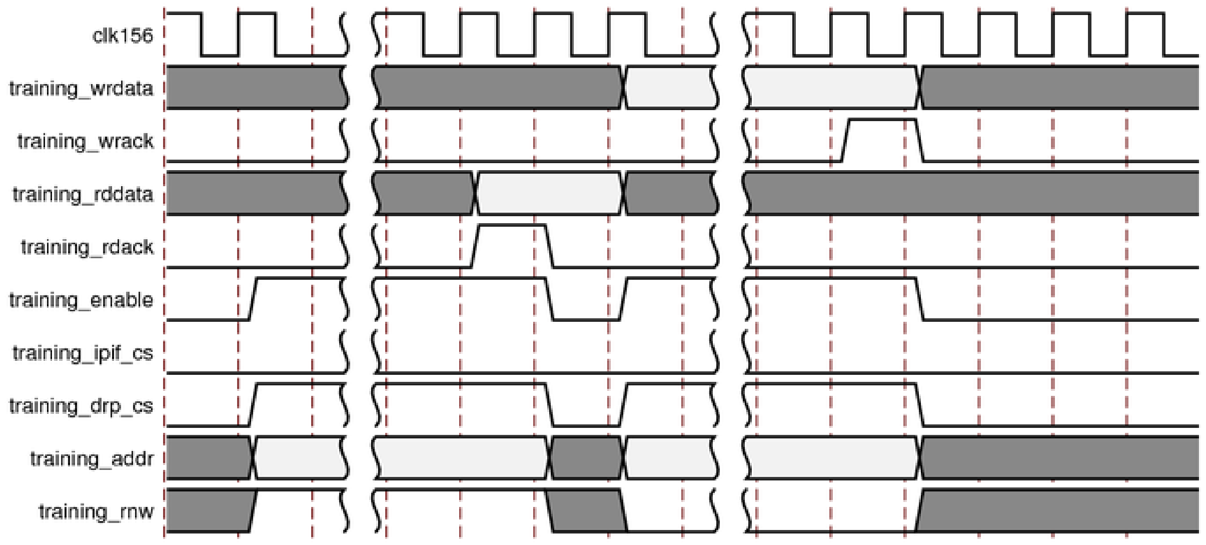


Figure 2-4: **Using the Training Interface to Access Transceiver Registers via the DRP Port**

Miscellaneous Signals - Virtex-7/Kintex-7 FPGAs

Table 2-17: **Miscellaneous Signals**

Signal Name	Direction	Description
core_status[7:0]	OUT	Bit 0 = PCS Block Lock, Bits [7:6] are reserved BASE-KR cores: FEC Signal OK in bit 1, pmd_signal_detect (Training Done) in bit 2, AN Complete in bit 3, AN Enable is bit 4 and an_link_up is bit 5.[a]
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

a. The latter two signals are required in the block level of the core to enable a switching of transceiver RX modes during AutoNegotiation. When the optional AutoNegotiation block is not included with the core, an_link_up (bit 5) will be fixed to a constant 1 and bits 3 and 4 will be constant 0.

Register Space

This core implements registers which are further described in 802.3 Clause 45. If the core is generated without an MDIO interface, these registers are still implemented but generally using only configuration or status pins on the core. For example, register 1.0, bit 15 (PMA Reset) is implemented as bit 15 of the configuration vector and register 1.1, bit 7 (PMA/PMD Fault) is implemented as status vector bit 23. These mappings are described in Configuration and Status Vectors in Chapter 3.

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 2-18.

Table 2-18: **10GBASE-R/KR PCS/PMA MDIO Registers**

Register Address	Register Name
1.0	MDIO Register 1.0: PMA/PMD Control 1
1.1	MDIO Register 1.1: PMA/PMD Status 1
1.4	MDIO Register 1.4: PMA/PMD Speed Ability
1.5, 1.6	MDIO Registers 1.5 and 1.6: PMA/PMD Devices in Package
1.7	MDIO Register 1.7: 10G PMA/PMD Control 2
1.8	MDIO Register 1.8: 10G PMA/PMD Status 2
1.9	MDIO Register 1.9: 10G PMD Transmit Disable
1.10	MDIO Register 1.10: 10G PMD Signal Receive OK
1.11 to 1.32787	Reserved
1.32788	MDIO Register 1.32788: Vendor-Specific PMA Loopback Control (Virtex-6 FPGAs Only)
1.32789 to 1.65534	Reserved
1.65535	MDIO Register 1.65535: Core Version Info - Virtex-7/Kintex-7 FPGAs Only
3.0	MDIO Register 3.0: PCS Control 1
3.1	MDIO Register 3.1: PCS Status 1
3.4	MDIO Register 3.4: PCS Speed Ability
3.5, 3.6	MDIO Registers 3.5 and 3.6: PCS Devices in Package
3.7	MDIO Register 3.7: 10G PCS Control 2
3.8	MDIO Register 3.8: 10G PCS Status 2
3.9 to 3.31	Reserved
3.32	MDIO Register 3.32: 10GBASE-R Status 1
3.33	MDIO Register 3.33: Clear 10GBASE-R Status 2

Table 2-18: **10GBASE-R/KR PCS/PMA MDIO Registers *(Cont'd)***

Register Address	Register Name
3.34-37	MDIO Register 3.34-37: 10GBASE-R Test Pattern Seed A0-3
3.38-41	MDIO Register 3.38-41: 10GBASE-R Test Pattern Seed B0-3
3.42	MDIO Register 3.42: 10GBASE-R Test Pattern Control
3.43	MDIO Register 3.43: Clear 10GBASE-R Test Pattern Error Counter
3.44 to 3.65534	Reserved
3.32769 to 3.65534	Reserved
3.65535	MDIO Register 3.65535: 125 μs Timer Control - Virtex-7/Kintex-7 FPGAs Only

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 2-19.

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

Register Address	Register Name
1.0	MDIO Register 1.0: PMA/PMD Control 1
1.1	MDIO Register 1.1: PMA/PMD Status 1
1.150	MDIO Register 1.150: 10GBASE-KR PMD Control
1.151	MDIO Register 1.151: 10GBASE-KR PMD Status
1.152	MDIO Register 1.152: 10GBASE-KR LP Coefficient Update
1.153	MDIO Register 1.153: 10GBASE-KR LP Status
1.154	MDIO Register 1.154: 10GBASE-KR LD Coefficient Update
1.155	MDIO Register 1.155: 10GBASE-KR LD Status
1.170	MDIO Register 1.170: 10GBASE-R FEC Ability (1)
1.171	MDIO Register 1.171: 10GBASE-R FEC Control(1)
1.172 to 1.173	MDIO Register 1.172: 10GBASE-R FEC Corrected Blocks (Lower) (1) MDIO Register 1.173: 10GBASE-R FEC Corrected Blocks (Upper) (1)
1.174 to 1.175	MDIO Register 1.174: 10GBASE-R FEC Uncorrected Blocks (Lower) (1) MDIO Register 1.175: 10GBASE-R FEC Uncorrected Blocks (Upper) (1)
1.4	MDIO Register 1.4: PMA/PMD Speed Ability
1.5, 1.6	MDIO Registers 1.5 and 1.6: PMA/PMD Devices in Package
1.7	MDIO Register 1.7: 10G PMA/PMD Control 2
1.8	MDIO Register 1.8: 10G PMA/PMD Status 2
1.9	MDIO Register 1.9: 10G PMD Transmit Disable
1.10	MDIO Register 1.10: 10G PMD Signal Receive OK
1.11 to 1.149	Reserved
1.176 to 1.65519	Reserved

Table 2-19: **10GBASE-KR PCS/PMA Registers *(Cont'd)***

Register Address	Register Name
1.65520	MDIO Register: 1.65520: Vendor-Specific LD Training (vendor-specific register where Local Device Coefficient Updates are to be written by Training Algorithm)
1.65521 to 1.65534	Reserved
1.65535	MDIO Register 1.65535: Core Version Info - Virtex-7/Kintex-7 FPGAs Only
3.0	MDIO Register 3.0: PCS Control 1
3.1	MDIO Register 3.1: PCS Status 1
3.4	MDIO Register 3.4: PCS Speed Ability
3.5, 3.6	MDIO Registers 3.5 and 3.6: PCS Devices in Package
3.7	MDIO Register 3.7: 10G PCS Control 2
3.8	MDIO Register 3.8: 10G PCS Status 2
3.9 to 3.31	Reserved
3.32	MDIO Register 3.32: 10GBASE-R Status 1
3.33	MDIO Register 3.33: Clear 10GBASE-R Status 2
3.34-37	MDIO Register 3.34-37: 10GBASE-R Test Pattern Seed A0-3
3.38-41	MDIO Register 3.38-41: 10GBASE-R Test Pattern Seed B0-3
3.42	MDIO Register 3.42: 10GBASE-R Test Pattern Control
3.43	MDIO Register 3.43: Clear 10GBASE-R Test Pattern Error Counter
3.44 to 3.32767	Reserved
3.32768	MDIO Register 3.32768: Vendor-Specific PCS Loopback Control - Virtex-6 FPGAs Only
3.65535	MDIO Register 3.65535: 125 μs Timer Control - Virtex-7/Kintex-7 FPGAs Only
7.0	MDIO Register 7.0: AN Control (2)
7.1	MDIO Register 7.1: AN Status (2)
7.16, 17, 18	MDIO Register 7.16:17:18: AN Advertisement (2)
7.19, 20, 21	MDIO Register 7.19, 20, 21: AN LP Base Page AbilityA (2)
7.22, 23, 24	MDIO Register 7.22, 23, 24: AN XNP TransmitA (2)
7.25, 26, 27	MDIO Register 7.25, 26, 27: AN LP XNP Ability (2)
7.48	MDIO Register 7.48: Backplane Ethernet Status(2)

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

MDIO Register 1.0: PMA/PMD Control 1

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

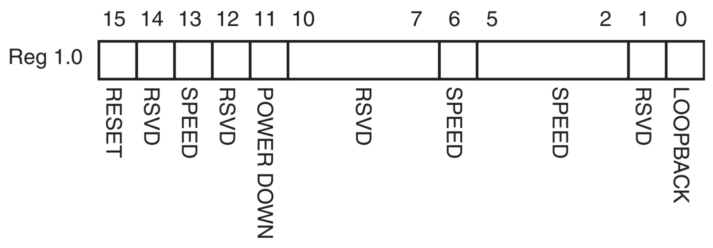


Figure 2-5: **PMA/PMD Control 1 Register**

Table 2-20 shows the PMA Control 1 register bit definitions.

Table 2-20: **PMA/PMD Control 1 Register Bit Definitions**

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

Table 2-20: **PMA/PMD Control 1 Register Bit Definitions *(Cont'd)***

Bit(s)	Name	Description	Attributes	Default Value
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 0 = Disable PMA loopback mode Virtex-6 FPGAs: The 10GBASE-R/KR block will loop the transmit signal inside the GTH transceiver back into the receiver. 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 2-6 shows the MDIO Register 1.1: PMA/PMD Status 1.

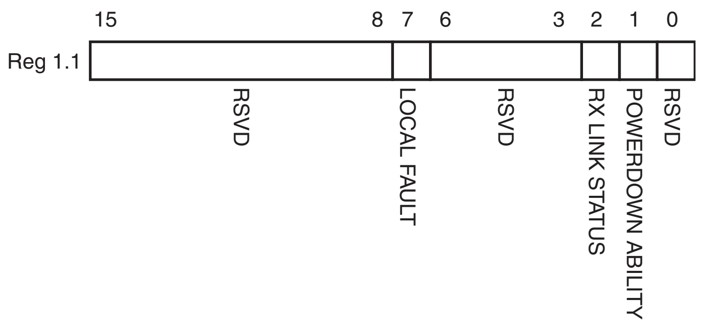


Figure 2-6: **PMA/PMD Status 1 Register**

Table 2-21 shows the PMA/PMD Status 1 register bit definitions.

Table 2-21: **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. 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 Virtex-7/Kintex-7: 1 = Receive Link UP	R/O 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 2-7 shows the MDIO Register 1.4: PMA/PMD Speed Ability.

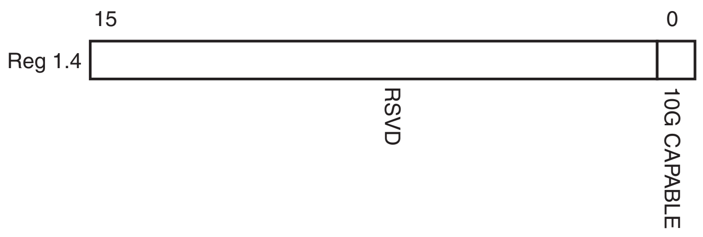


Figure 2-7: **PMA/PMD Speed Ability Register**

Table 2-22 shows the PMA/PMD Speed Ability register bit definitions.

Table 2-22: **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 2-8 shows the MDIO Registers 1.5 and 1.6: PMA/PMD Devices in Package.

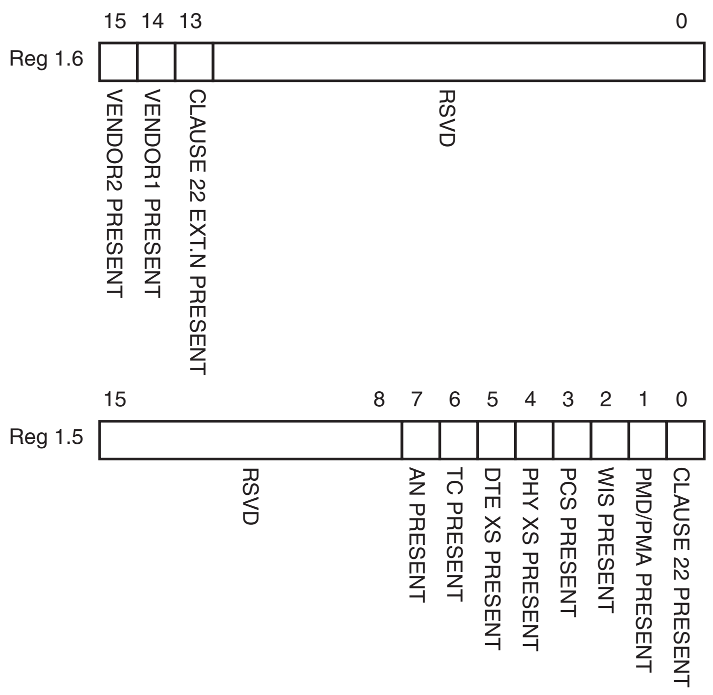


Figure 2-8: **PMA/PMD Devices in Package Registers**

Table 2-23 shows the PMA/PMD Device in Package registers bit definitions.

Table 2-23: **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. 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

Table 2-23: **PMA/PMD Devices in Package Registers Bit Definitions *(Cont'd)***

Bit(s)	Name	Description	Attributes	Default Value
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 2-9 shows the MDIO Register 1.7: 10G PMA/PMD Control 2.

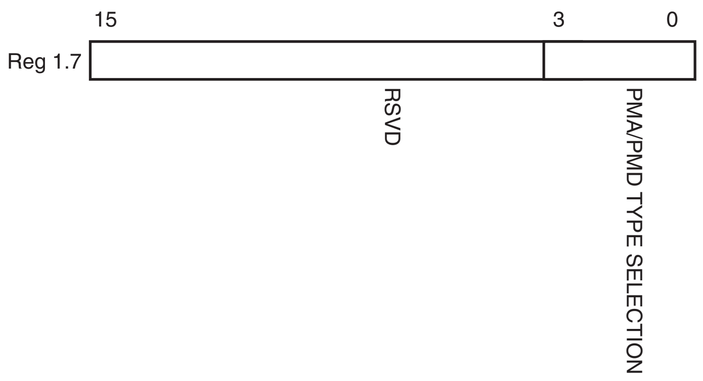


Figure 2-9: **10G PMA/PMD Control 2 Register**

Table 2-24 shows the PMA/PMD Control 2 register bit definitions.

Table 2-24: **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: This returns the value '0xyz', where 'xyz' is set from the top level core port pma_pmd_type vector. 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: Base-R: Set from pma_pmd_type port. BASE-KR: returns 0xB Virtex-6 FPGAs: Set from GTH transceiver attribute.

MDIO Register 1.8: 10G PMA/PMD Status 2

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

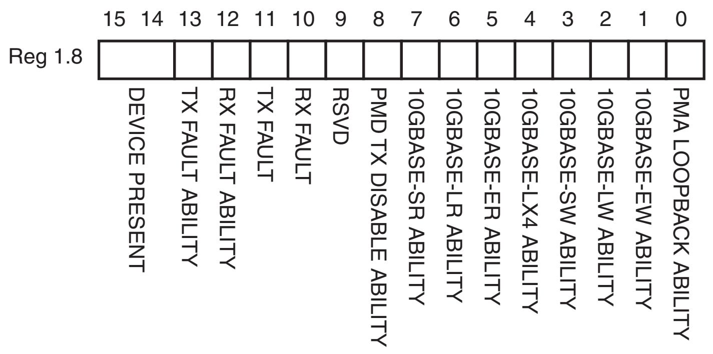


Figure 2-10: **10G PMA/PMD Status 2 Register**

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

Table 2-25: **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. Virtex-7/Kintex-7: The block always returns a '1' for this bit.	R/O	Virtex-6: 0 Virtex-7/Kintex-7: 1
1.8.12	Receive Local Fault Ability	Virtex-6: The block always returns '0' for this bit Virtex-7/Kintex-7: The block always returns a '1' for this bit.	R/O	Virtex-6: 0 Virtex-7/Kintex-7: 1
1.8.11	Transmit Fault	Virtex-6: The block always returns '0' for this bit. Virtex-7/Kintex-7: 1 = Transmit Fault detected	R/O V7/K7: Latches High	0
1.8.10	Receive Fault	Virtex-6: The block always returns '0' for this bit. Virtex-7/Kintex-7: 1 = Receive Fault detected	R/O 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: Base-R only: Returns a '1' if pma_pmd_type port is set to '111' Virtex-6 FPGAs: The block always returns ‘1’ for this bit.	R/O	Virtex-7/Kintex-7: Depends on pma_pmd_type port Virtex-6 FPGAs: 1

Table 2-25: **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' 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' 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' Virtex-6 FPGAs: The block always returns '1' for this bit.	R/O	Virtex-7/Kintex-7: Depends on pma_pmd_type port 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 2-11 shows the MDIO 1.9 Register: 10G PMD Transmit Disable.

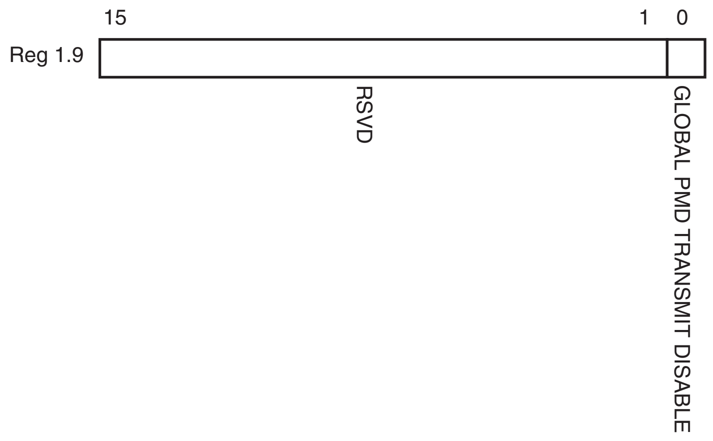


Figure 2-11: **10G PMD Transmit Disable Register**

Table 2-26: **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) 0 = Enable Transmit path	Virtex-6: R/W V7/K7: R/W	Virtex-6: Set from GTH attribute. V7/K7: 0

MDIO Register 1.10: 10G PMD Signal Receive OK

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

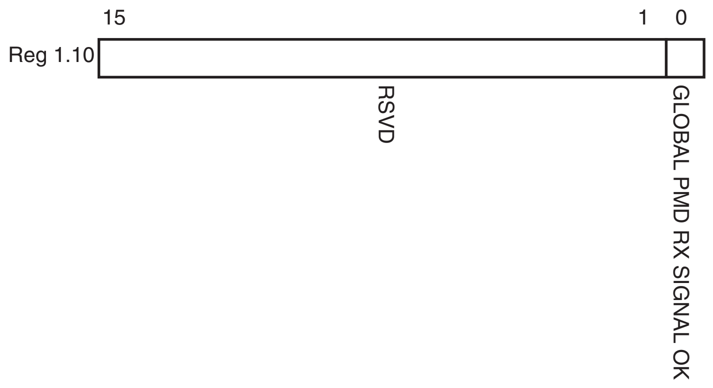


Figure 2-12: **10G PMD Signal Receive OK Register**

Table 2-25 shows the PMD Signal Receive OK register bit definitions.

Table 2-27: **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 0 = Signal not detected on receive	R/O	n/a

MDIO Register 1.150: 10GBASE-KR PMD Control

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

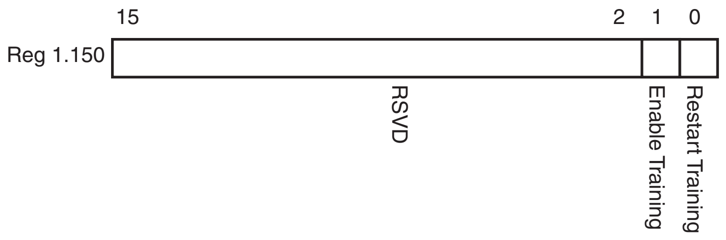


Figure 2-13: **10GBASE-KR PMD Control Register**

Table 2-28 shows the 10GBASE-KR PMD Control register bit definitions.

Table 2-28: **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 0 = Disable	R/W	0
1.150.0	Restart Training	1 = Reset the 10GBASE-KR start-up protocol 0 = Normal operation	R/W Self-clearing	0

MDIO Register 1.151: 10GBASE-KR PMD Status

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

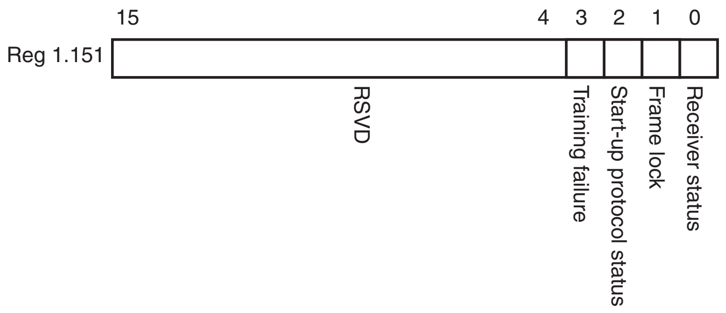


Figure 2-14: **10GBASE-KR PMD Status Register**

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

Table 2-29: **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 0 = Not detected	R/O	0
1.151.2	Start-up Protocol status	1 = Start-up protocol in progress 0 = Protocol complete	R/O	0
1.151.1	Frame Lock	1 = Training frame delineation detected 0 = Not detected	R/O	0
1.151.0	Receiver status	1 = Receiver trained and ready to receive data 0 = Receiver training	R/O	0

MDIO Register 1.152: 10GBASE-KR LP Coefficient Update

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

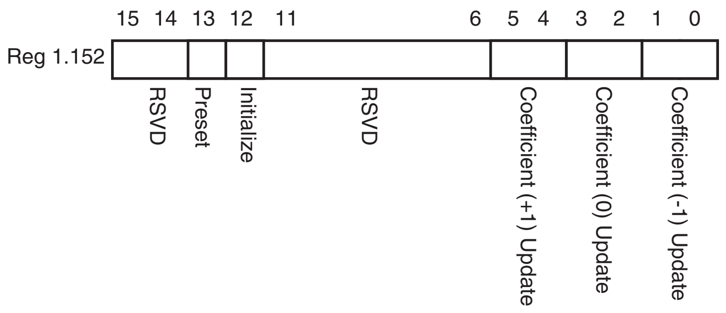


Figure 2-15: **10GBASE-KR LP Coefficient Update Register**

Table 2-30 shows the 10GBASE-KR LP coefficient update register bit definitions.

Table 2-30: **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 0 = Normal operation	R/W[a]	0
1.152.12	Initialize	1 = Initialize coefficients 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	0s
1.152.5:4	Coefficient (+1) update	5:4 = 11 = reserved 10 = decrement 01 = increment 00 = hold	R/W[a]	00
1.152.3:2	Coefficient (0) update	3:2 = 11 = reserved 10 = decrement 01 = increment 00 = hold	R/W[a]	00
1.152.1:0	Coefficient (-1) update	1:0 = 11 = reserved 10 = decrement 01 = increment 00 = hold	R/W[a]	00

a. Writable only when register 1.150.1 = 0

MDIO Register 1.153: 10GBASE-KR LP Status

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

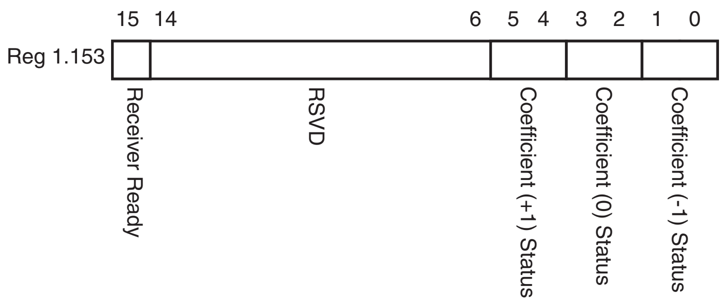


Figure 2-16: **10GBASE-KR LP Status Register**

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

Table 2-31: **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 0 = The LP receiver is requesting that training continue	R/O	0
1.153.14:6	Reserved	The block always returns '0' for this bit and ignores writes.	R/O	0s
1.153.5:4	Coefficient (+1) status	5:4 = 11 = maximum 10 = minimum 01 = updated 00 = not updated	R/O	00
1.153.3:2	Coefficient (0) status	3:2 = 11 = maximum 10 = minimum 01 = updated 00 = not updated	R/O	00
1.153.1:0	Coefficient (-1) status	1:0 = 11 = maximum 10 = minimum 01 = updated 00 = not updated	R/O	00

MDIO Register 1.154: 10GBASE-KR LD Coefficient Update

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

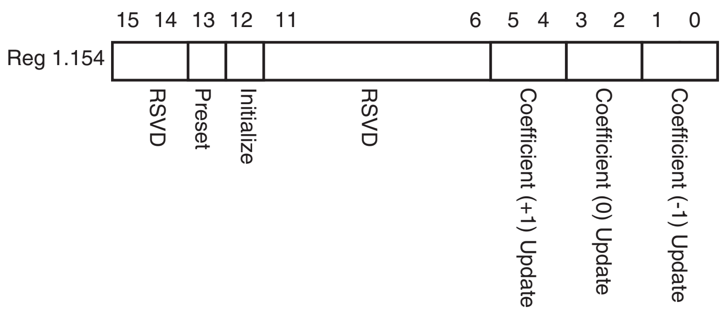


Figure 2-17: **10GBASE-KR LD Coefficient Update Register**

Table 2-32 shows the 10GBASE-KR LD coefficient update register bit definitions.

Table 2-32: **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 0 = Normal operation	R/O[a]	0
1.154.12	Initialize	1 = Initialize coefficients 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	0s
1.154.5:4	Coefficient (+1) update	5:4 = 11 = reserved 10 = decrement 01 = increment 00 = hold	R/O[a]	00
1.154.3:2	Coefficient (0) update	3:2 = 11 = reserved 10 = decrement 01 = increment 00 = hold	R/O[a]	00
1.154.1:0	Coefficient (-1) update	1:0 = 11 = reserved 10 = decrement 01 = increment 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 2-18 shows the MDIO Register 1.155: 10GBASE-KR LD status.

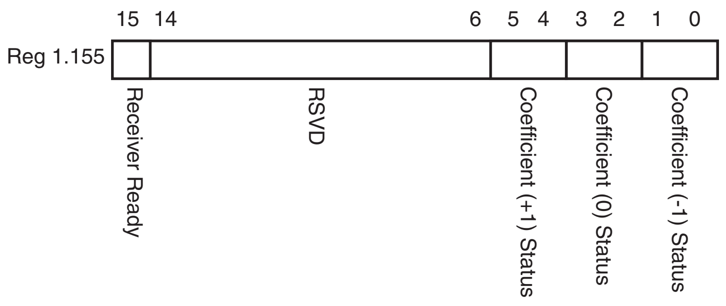


Figure 2-18: **10GBASE-KR LD Status Register**

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

Table 2-33: **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 0 = The LD receiver is requesting that training continue	R/O	0
1.155.14:6	Reserved	The block always returns ‘0’ for this bit and ignores writes.	R/O	0s
1.155.5:4	Coefficient (+1) status	5:4 = 11 = maximum 10 = minimum 01 = updated 00 = not updated	R/O	00
1.155.3:2	Coefficient (0) status	3:2 = 11 = maximum 10 = minimum 01 = updated 00 = not updated	R/O	00
1.155.1:0	Coefficient (-1) status	1:0 = 11 = maximum 10 = minimum 01 = updated 00 = not updated	R/O	00

MDIO Register 1.170: 10GBASE-R FEC Ability

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

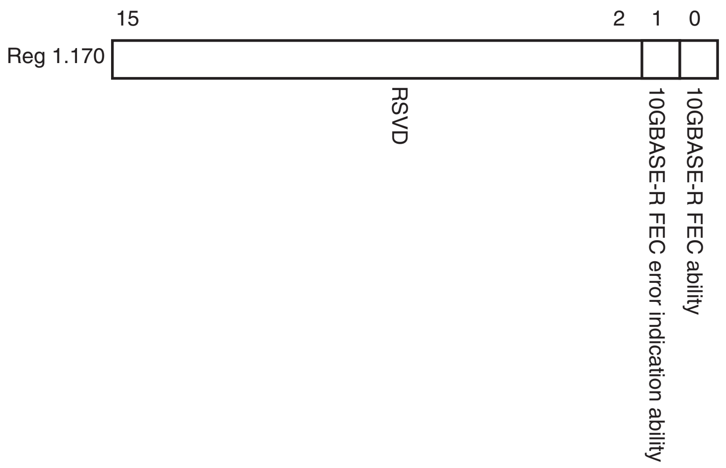


Figure 2-19: **10GBASE-R FEC Ability Register**

Table 2-34 shows the 10GBASE-R FEC Ability register bit definitions.

Table 2-34: **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	0s
1.170.1	10GBASE-R FEC error indication ability	1 = the PHY is able to report FEC decoding errors to the PCS layer	R/O	1
1.170.0	10GBASE-R FEC ability	1 = the PHY supports FEC	R/O	1

MDIO Register 1.171: 10GBASE-R FEC Control

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

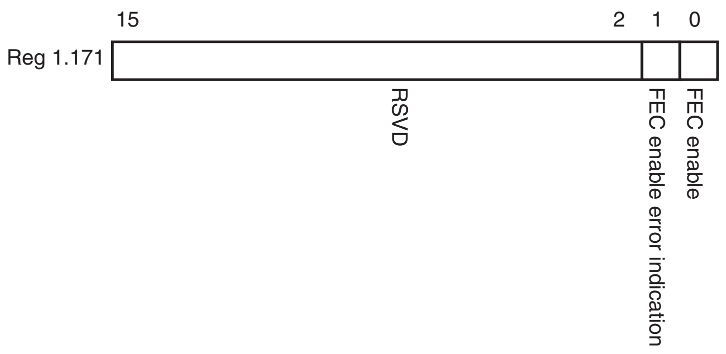


Figure 2-20: **10GBASE-R FEC Control Register**

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

Table 2-35: **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	0s
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 0 = disable FEC	R/W	0

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

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

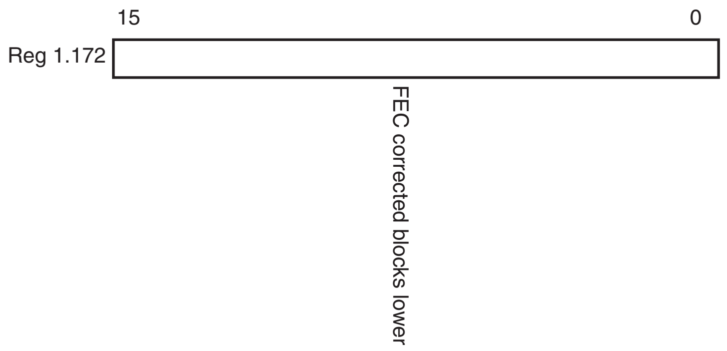


Figure 2-21: **10GBASE-R FEC Corrected Blocks (Lower) Register**

Table 2-36 shows the 10GBASE-R FEC Corrected Blocks (lower) register bit definitions.

Table 2-36: **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[a]	0s

a. Cleared when read.

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

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

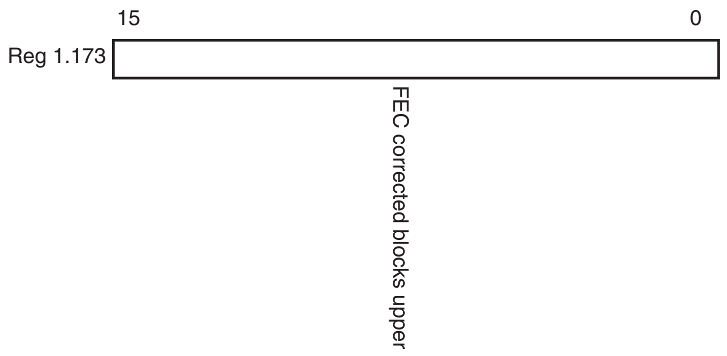


Figure 2-22: **10GBASE-R FEC Corrected Blocks (Upper) Register**

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

Table 2-37: **10GBASE-R FEC Corrected Blocks (Upper) Register Bit Definitions**

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

a. Latched when 1.172 is read. Cleared when read.

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

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

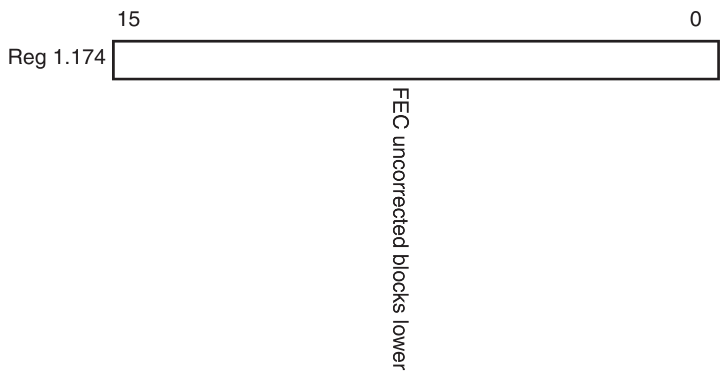


Figure 2-23: **10GBASE-R FEC Uncorrected Blocks (Lower) Register**

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

Table 2-38: **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[a]	0s

a. Cleared when read.

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

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

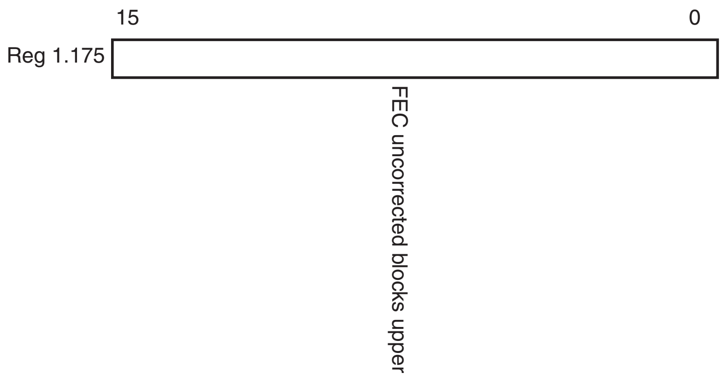


Figure 2-24: **10GBASE-R FEC Uncorrected Blocks (Upper) Register**

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

Table 2-39: **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[a]	0s

a. Latched when 1.174 is read. Cleared when read.

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

Figure 2-25 shows the MDIO 1.32788 Register: Vendor-Specific PMA Loopback Control.

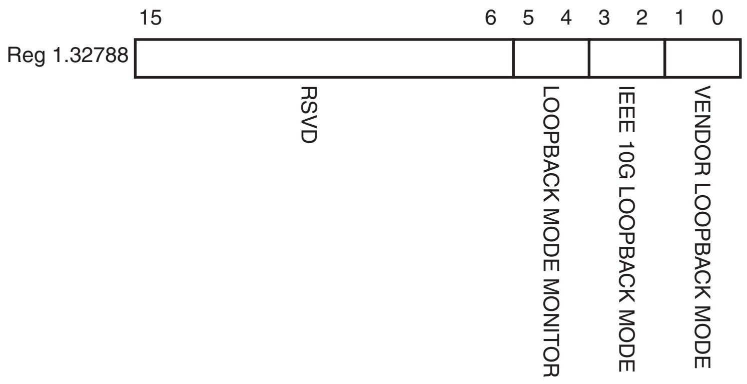


Figure 2-25: **Vendor-Specific PMA Loopback Control Register**

Table 2-40: **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. 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. These values are applicable only when 1.0.0 is asserted and the vendor config lane loopback mode is disabled. Encodings: '00': IEEE PMA loopback disabled '01': TX output '10': TX predriver '11': Reserved	R/W	'01'
1.32788.1:0	Vendor Specific Loopback Mode	Configure source of on-chip loopback connection to RX. Encodings: '00': Vendor loopback disabled '01': TX output '10': TX predriver '11': Reserved	R/W	'00'

MDIO Register: 1.65520: Vendor-Specific LD Training

Figure 2-26 shows the MDIO Register 1.65520: Vendor-specific LD Training.

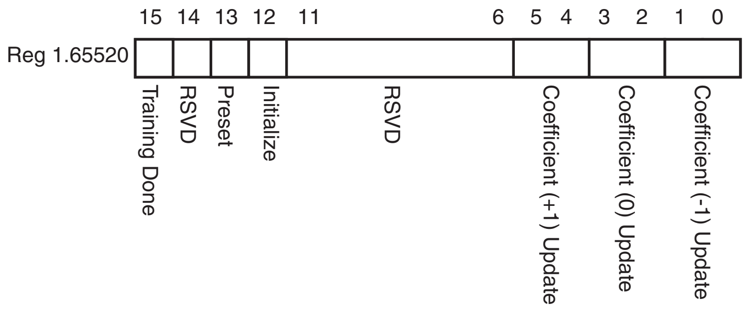


Figure 2-26: **Vendor-specific LD Training Register**

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

Table 2-41: **Vendor-Specific LD Training Register 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 0 = Normal operation	R/O[b]	0
1.65520.12	Initialize	1 = Initialize coefficients 0 = Normal operation	R/O[b]	0
1.65520.5:4	Coefficient (+1) update	5:4 = 11 = reserved 10 = decrement 01 = increment 00 = hold	R/O[b]	00
1.65520.3:2	Coefficient (0) update	3:2 = 11 = reserved 10 = decrement 01 = increment 00 = hold	R/O[b]	00
1.65520.1:0	Coefficient (-1) update	1:0 = 11 = reserved 10 = decrement 01 = increment 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 2-27 shows the MDIO 1.65535 Register: Core Version Info

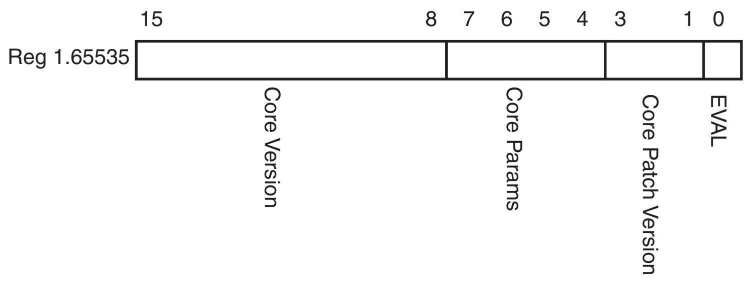


Figure 2-27: **Core Version Info Register**

Table 2-42: **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'26' for version 2.6 of core
1.65535.7:4	Core parameters	Bit 7 = 1 = KR included Bit 6 - reserved Bit 5 = 1 = AN included Bit 4 = 1 = FEC included	R/O	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 2-28 shows the MDIO Register 3.0: PCS Control 1.

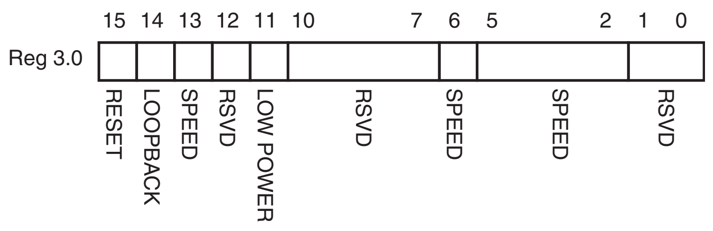


Figure 2-28: **PCS Control 1 Register**

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

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

Bit(s)	Name	Description	Attributes	Default Value
3.0.15	Reset	1 = Block reset 0 = Normal operation The 10GBASE-R/KR block is reset when this bit is set to '1.' It returns to '0' when the reset is complete.	R/W Self-clearing	0
3.0.14	10GBASE-R/KR Loopback	1 = Use PCS Loopback 0 = Do not use PCS Loopback	R/W	0
3.0.13	Speed Selection	The block always returns '1' for this bit. 1 (and bit 6 = 1) = bits 5:2 select the speed	R/O	1
3.0.12	Reserved	The block always returns '0' for this bit and ignores writes.	R/O	0
3.0.11	Power down	This bit has no effect.	R/W	0

Table 2-43: **PCS Control 1 Register Bit Definitions *(Cont'd)***

Bit(s)	Name	Description	Attributes	Default Value
3.0.10:7	Reserved	The block always returns '0' for these bits and ignores writes.	R/O	All 0s
3.0.6	Speed Selection	The block always returns '1' for this bit.	R/O	1
3.0.5:2	Speed Selection	The block always returns "0000" = 10Gb/s	R/O	All 0s
3.0.1:0	Reserved	The block always returns '0' for this bit and ignores writes.	R/O	All 0s

MDIO Register 3.1: PCS Status 1

Figure 2-29 shows the MDIO Register 3.1: PCS Status 1.

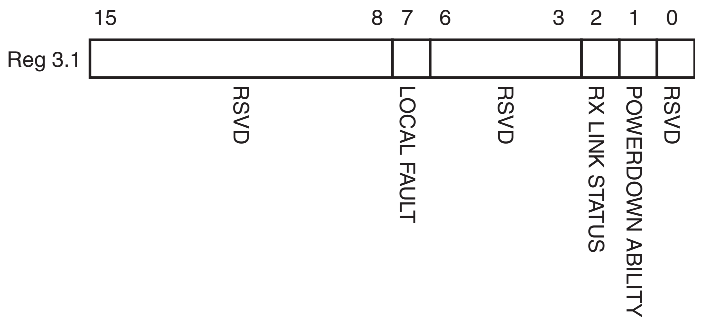


Figure 2-29: **PCS Status 1 Register**

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

Table 2-44: **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. 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 0 = The PCS receive link is down This is a latching Low version of bit 3.32.12.	R/O Self-setting	-

Table 2-44: **PCS Status 1 Register Bit Definition *(Cont'd)***

Bit(s)	Name	Description	Attributes	Default Value
3.1.1	Power Down Ability	The block always returns '1' for this bit.	R/O	1
3.1.0	Reserved	The block always returns'0' for this bit and ignores writes.	R/O	0

MDIO Register 3.4: PCS Speed Ability

Figure 2-30 shows the MDIO Register 3.4: PCS Speed Ability.

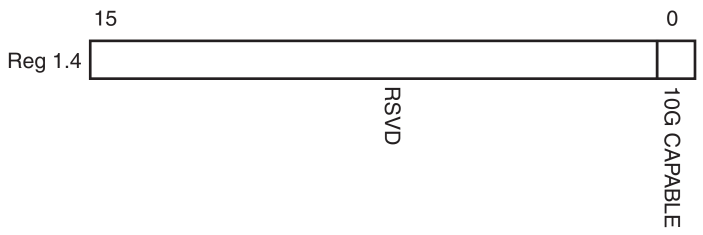


Figure 2-30: **PCS Speed Ability Register**

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

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

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

MDIO Registers 3.5 and 3.6: PCS Devices in Package

Figure 2-31 shows the MDIO Registers 3.5 and 3.6: PCS Devices in Package.

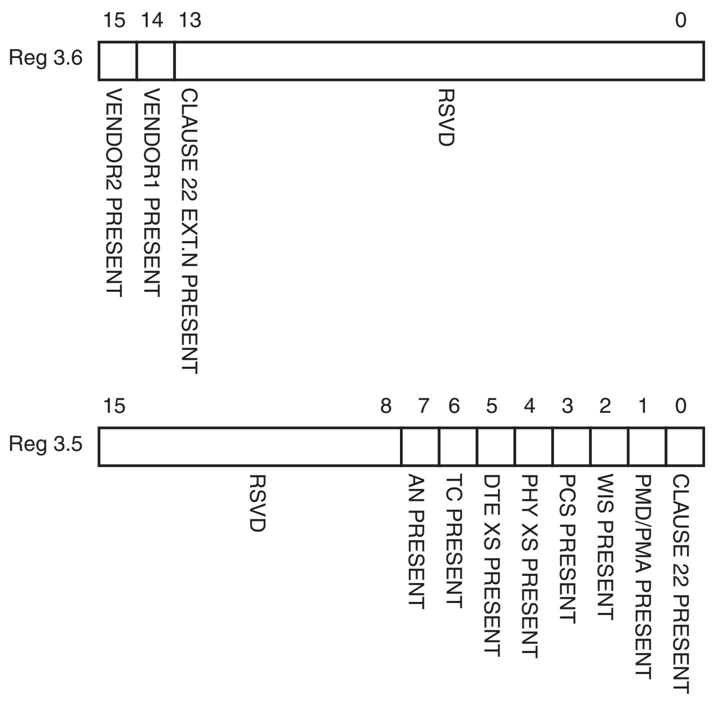


Figure 2-31: **PCS Devices in Package Registers**

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

Table 2-46: **PCS Devices in Package Registers 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. V7/K7: The block always returns '0' for this bit.	1/0	1
3.6.12:0	Reserved	The block always returns '0' for these bits.	R/O	All 0s
3.5.15:8	Reserved	The block always returns '0' for these bits.	R/O	All 0s
3.5.7	Auto Negotiation Present	Virtex-6: The block always returns'1' for this bit. V7/K7: 1 = AN Block included	1/0	1

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

Bit(s)	Name	Description	Attributes	Default Value
3.5.6	TC present	The block always returns'0' for this bit.	R/O	0
3.5.5	PHY XS Present	The block always returns '0' for this bit.	R/O	0
3.5.4	PHY XS Present	The block always returns '0' for this bit.	R/O	0
3.5.3	PCS Present	The block always returns '1' for this bit.	R/O	1
3.5.2	WIS Present	The block always returns '0' for this bit.	R/O	0
3.5.1	PMA/PMD Present	The block always returns '1' for this bit.	R/O	1
3.5.0	Clause 22 device present	The block always returns '0' for this bit.	R/O	0

MDIO Register 3.7: 10G PCS Control 2

Figure 2-32 shows the MDIO Register 3.7: 10G PCS Control 2.

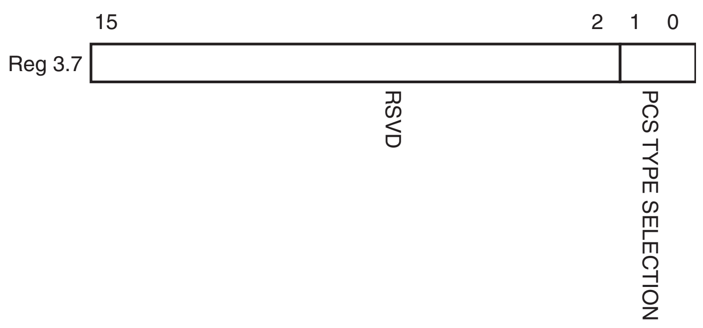


Figure 2-32: **10G PCS Control 2 Register**

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

Table 2-47: **10G PCS Control 2 Register 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 2-33 shows the MDIO Register 3.8: 10G PCS Status 2.

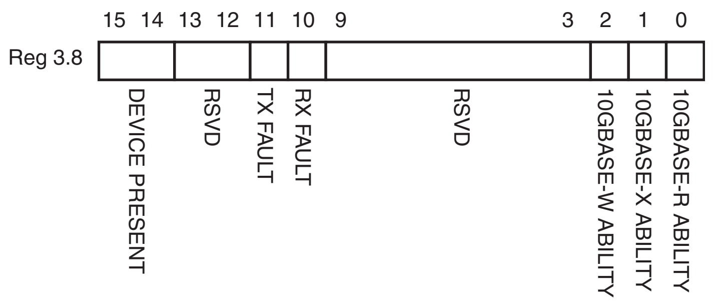


Figure 2-33: **10G PCS Status 2 Register**

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

Table 2-48: **10G PCS Status 2 Register 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. 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. 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 2-34 shows the MDIO Register 3.32: 10GBASE-R Status 1.

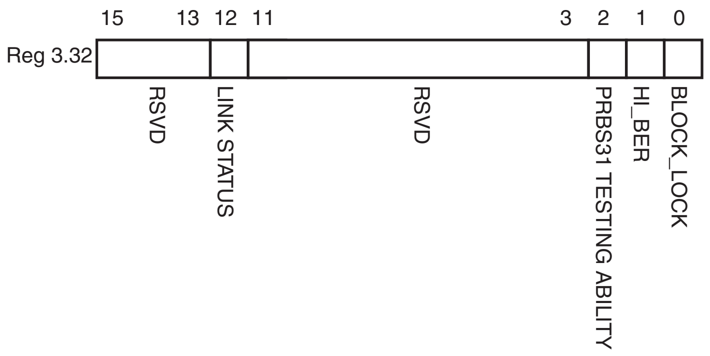


Figure 2-34: **10GBASE-R Status Register 1**

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

Table 2-49: **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; 0 = 10GBASE-R receive is not aligned.	RO	0
3.32.11:3	Reserved	The block always returns '0' for these bits.	R/O	0s
3.32.2	PRBS31 Pattern Testing Ability	The block always returns '1' for this bit.	R/O	1
3.32.1	Hi BER	1 = RX showing hi-ber 0 = RX not showing hi ber	R/O	0
3.32.0	Block Lock	1 = RX is synchronized; 0 = RX is not synchronized.	R/O	0

MDIO Register 3.33: Clear 10GBASE-R Status 2

Figure 2-35 shows the MDIO Register 3.33: 10GBASE-R Status 2.

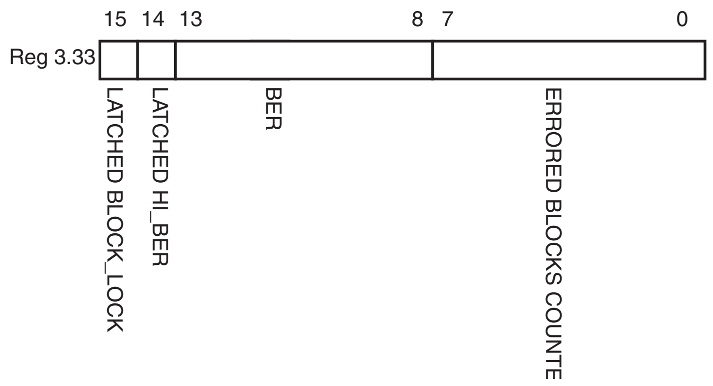


Figure 2-35: **10GBASE-R Status Register 2**

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

Table 2-50: **10GBASE-R Status Register 2 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 2-36 shows the MDIO Register 3.34-37 10GBASE-R Test Pattern Seed A.

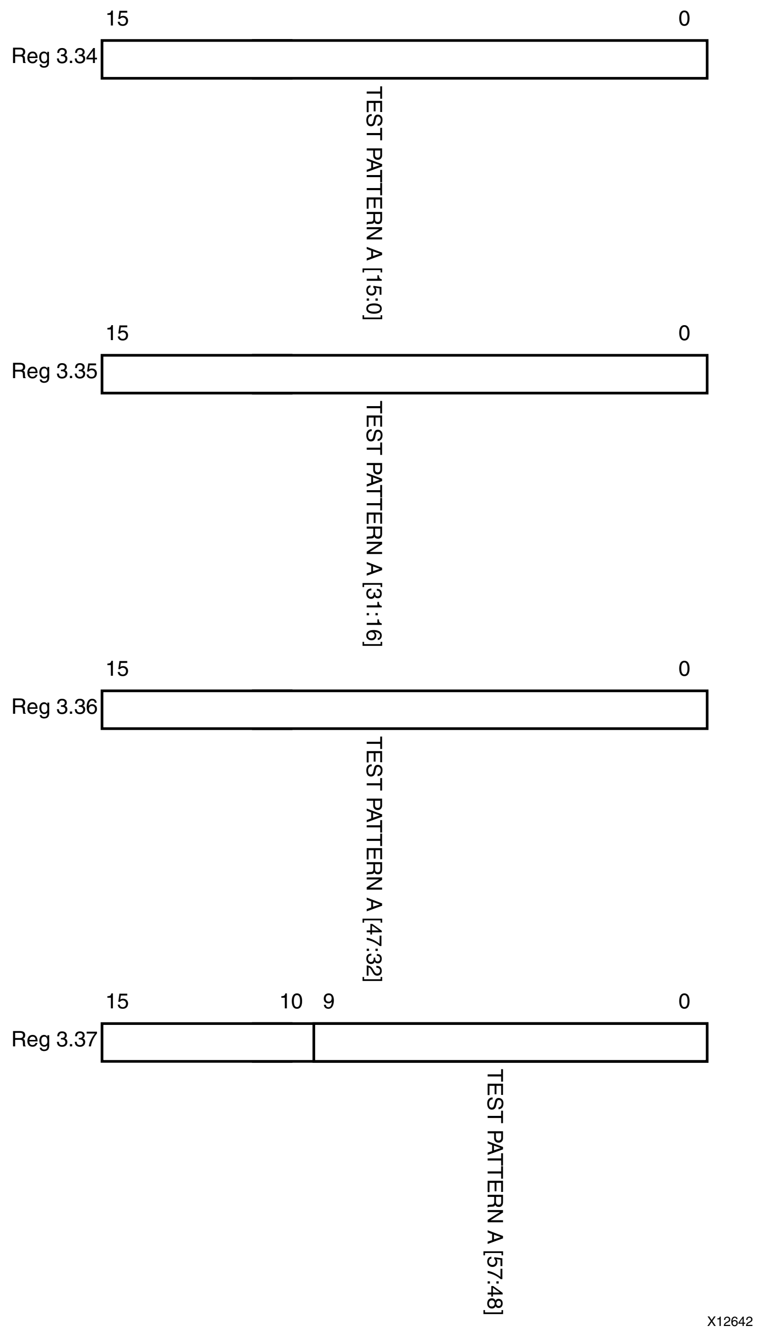


Figure 2-36: **10GBASE-R Test Pattern Seed A0-3 Registers**

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

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

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

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

Figure 2-37 shows the MDIO Register 3.38-41: 10GBASE-R Test Pattern Seed B.

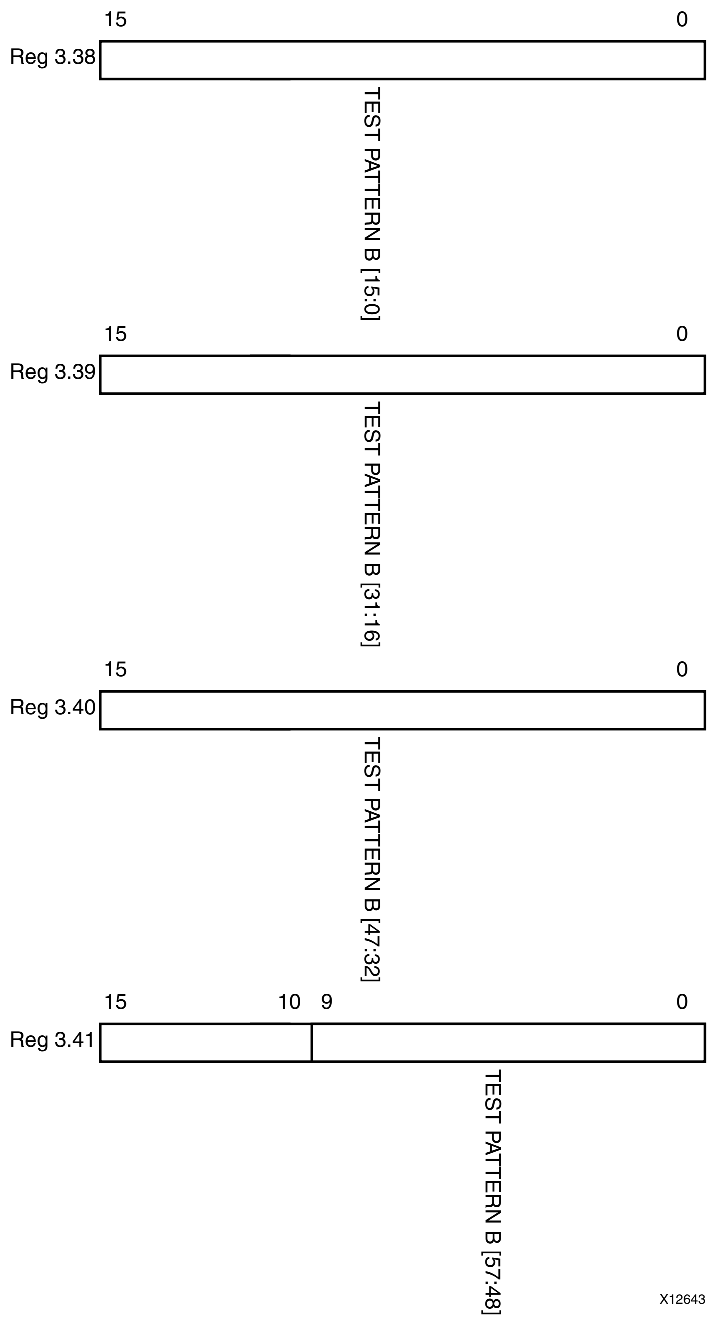


Figure 2-37: **10GBASE-R Test Pattern Seed B0-3 Registers**

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

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

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

MDIO Register 3.42: 10GBASE-R Test Pattern Control

Figure 2-38 shows the MDIO Register 3.42: 10GBASE-R Test Pattern Control.

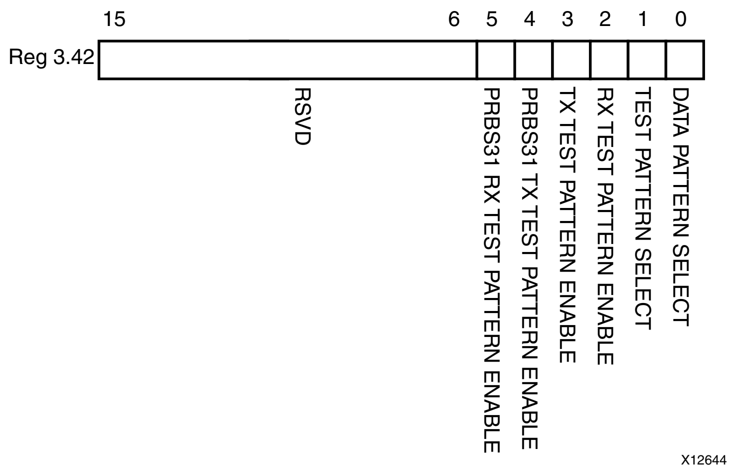


Figure 2-38: **10GBASE-R Test Pattern Control Register**

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

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

Bit(s)	Name	Description	Attributes	Default Value
3.42.15:6	Reserved	The block always returns 0s for these bits.	R/O	All 0s
3.42.5	PRBS31 RX test pattern enable	1 = Enable PRBS RX tests 0 = Disable PRBS RX tests	R/W	0
3.42.4	PRBS31 TX test pattern enable	1 = Enable PRBS TX tests 0 = Disable PRBS TX tests	R/W	0
3.42.3	TX test pattern enable	Enables the TX Test Pattern which has been selected with bits [1:0].	R/W	0
3.42.2	RX test pattern enable	Enables the RX Test Pattern Checking which has been selected with bits [1:0]	R/W	0

Table 2-53: **10GBASE-R Test Pattern Control Register Bit Definitions *(Cont'd)***

Bit(s)	Name	Description	Attributes	Default Value
3.42.1	Test pattern select	1 = Square wave 0 = Pseudo-Random	R/W	0
3.42.0	Data pattern select	1 = Zeros pattern 0 = LF Data pattern	R/W	0

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

Figure 2-39 shows the MDIO Register 3.43: 10GBASE-R Test Pattern Error Counter.

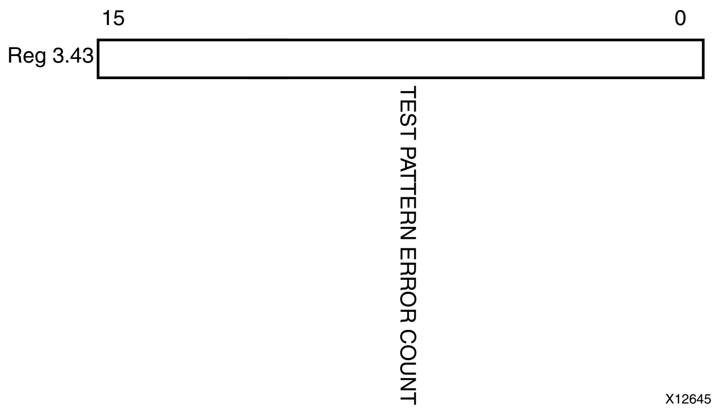


Figure 2-39: **10GBASE-R Test Pattern Error Counter Register**

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

Table 2-54: **10GBASE-R Test Pattern Error Counter Register 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 2-40 shows the MDIO 3.32768 Register: Vendor-Specific PCS Loopback Control. Consult *Virtex-6 FPGA GTH Transceivers User Guide* (UG371) for details.

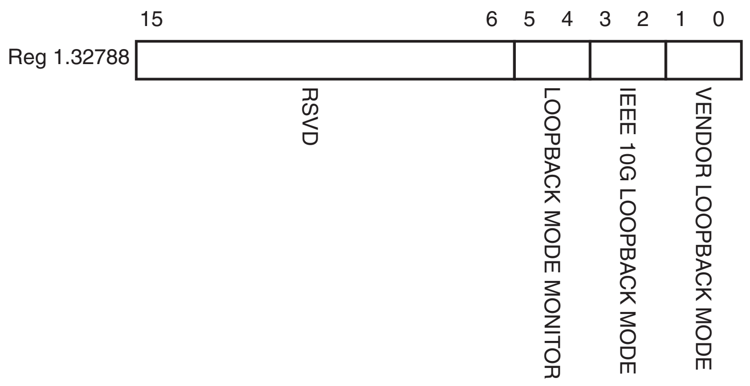


Figure 2-40: **Vendor-Specific PCS Loopback Control Register**

Table 2-55: **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 001: PRBS7 010: PRBS9 011: PRBS11 100: PRBS23 101: PRBS31 110: PPAT 111: Reserved	R/W	'000' Values other than '000' and '101' have no effect.
1.32768.10:8	PRBS Check Mode	000: None 001: PRBS7 010: PRBS9 011: PRBS11 100: PRBS23 101: PRBS31 110: PPAT 111: Reserved	R/W	'000' Values other than '000' and '101' have no effect.

Table 2-55: **Vendor-Specific PCS Loopback Control *(Cont'd)***

Bit(s)	Name	Description	Attributes	Default Value
3.32768.7:4	PCS RX Mode	0000: Zero 0001: 64/66b 0010: XAUI 0011: Reserved 0100: PCIE® 0101: PCIE-500 (fixed PCLK 500MHz) 0110: 8B/10B 8-bit 0111: 8B/10B 16-bit 1000: 8-bit raw data 1001: 10-bit raw data 1010: 16-bit raw data 1011: 20-bit raw data 1100: PRBS 1101: Reserved 1110: 1000 Base-KX 1111: 802.3ap	R/W	Set from GTH transceiver PCS_MODE Attribute = '0001' Do not write any other values to these bits.
3.32768.3:0	PCS TX Mode	0000: Zero 0001: 64/66b 0010: XAUI 0011: Reserved 0100: PCIE 0101: PCIE-500 (fixed PCLK 500MHz) 0110: 8B/10B8-bit 0111: 8B/10B8 16-bit 1000: 8-bit raw data 1001: 10-bit raw data 1010: 16-bit raw data 1011: 20-bit raw data 1100: PRBS 1101: Reserved 1110: 1000 Base-KX 1111: 802.3ap	R/W	Set from GTH PCS_MODE Attribute = '0001' Do not write any other values to these bits.

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

Figure 2-41 shows the MDIO 3.65535 Register: 125 μs timer control.

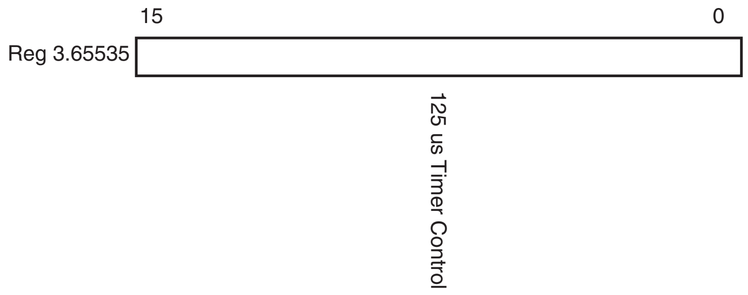


Figure 2-41: **125 μs Timer Control Register**

Table 2-56: **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 2-42 shows the MDIO Register 7.0: AN Control.

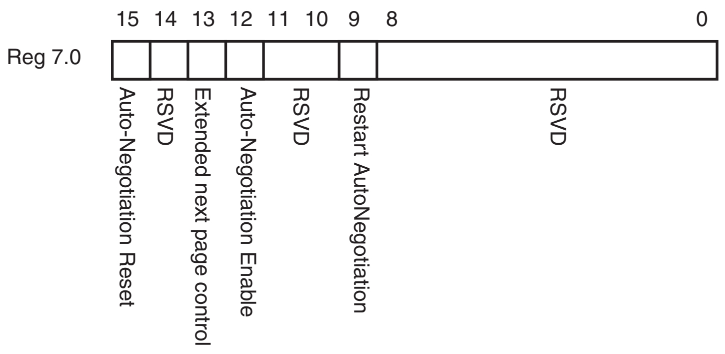


Figure 2-42: **AN Control Register**

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

Table 2-57: **AN Control Register Bit Definitions**

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

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

MDIO Register 7.1: AN Status

Figure 2-43 shows the MDIO Register 7.1: AN Status.

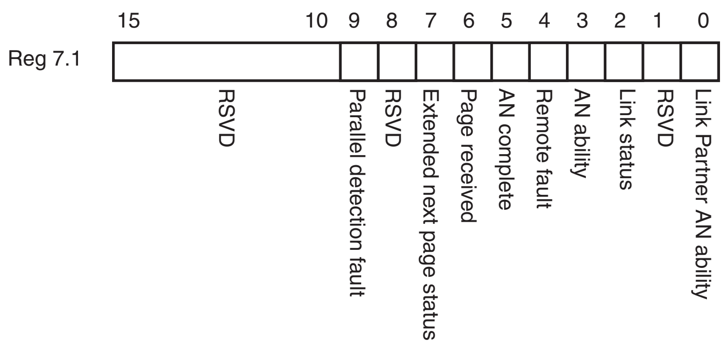


Figure 2-43: **AN Status Register**

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

Table 2-58: **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	0s
7.1.9	Parallel Detection Fault	1 = A fault has been detected through the parallel detection function 0 = no fault detected	R/O Latches High	0
7.1.8	Reserved	The block always returns '0' for this bit and ignores writes.	R/O	0
7.1.7	Extended Next Page status	1 = XNP format is used 0 = XNP format is not allowed	R/O	0
7.1.6	Page Received	1 = A page has been received 0 = No page received	R/O Latches High	0
7.1.5	AN Complete	1 = AN process has completed 0 = not completed	R/O	0
7.1.4	Remote fault	1 = remote fault condition detected 0 = not detected	R/O Latches High	0
7.1.3	AN ability	1 = PHY supports auto-negotiation 0 = PHY does not support auto-negotiation	R/O	1
7.1.2	Link status	1 = Link is up 0 = Link is down	R/O Latches Low	0
7.1.1	Reserved	The block always returns '0' for this bit and ignores writes.	R/O	0
7.1.0	LP AN ability	1 = LP is able to perform AN 0 = not able	R/O	0

MDIO Register 7.16:17:18: AN Advertisement

Figure 2-44 shows the MDIO Register 7.16: AN Advertisement.

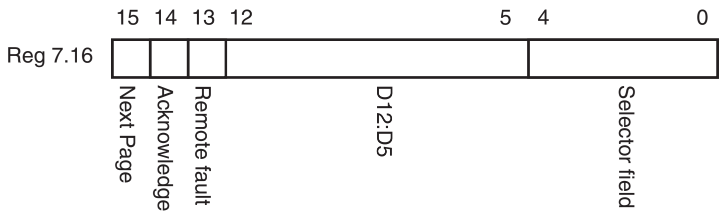


Figure 2-44: **AN Advertisement Register 0**

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

Table 2-59: **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	0s
7.16.4:0	Selector Field	Consult IEEE802.3	R/W	00001s

Figure 2-45 shows the MDIO Register 7.17: AN Advertisement.

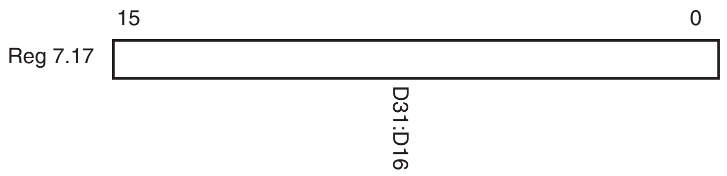


Figure 2-45: **AN Advertisement Register 1**

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

Table 2-60: **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 2-46 shows the MDIO Register 7.18: AN Advertisement.

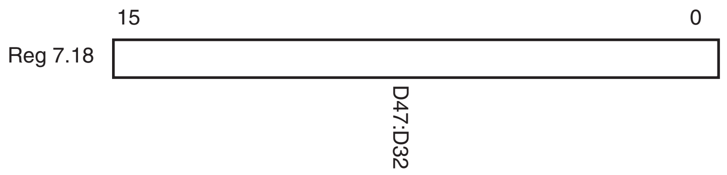


Figure 2-46: **AN Advertisement Register 2**

Table 2-61 shows the AN Advertisement register bit definitions.

Table 2-61: **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 2-47 shows the MDIO Register 7.19: AN LP Base Page Ability.

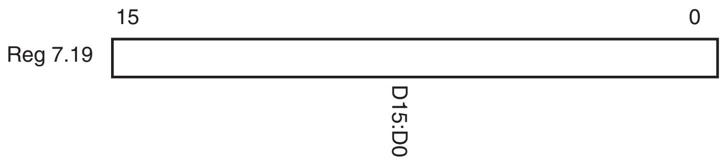


Figure 2-47: **AN LP Base Page Ability Register 0**

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

Table 2-62: **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 2-48 shows the MDIO Register 7.20: AN LP Base Page Ability.

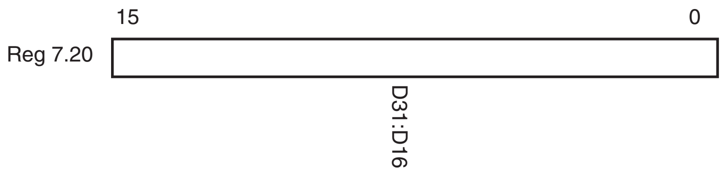


Figure 2-48: **AN LP Base Page Ability Register 1**

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

Table 2-63: **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 2-49 shows the MDIO Register 7.21: AN LP Base Page Ability.

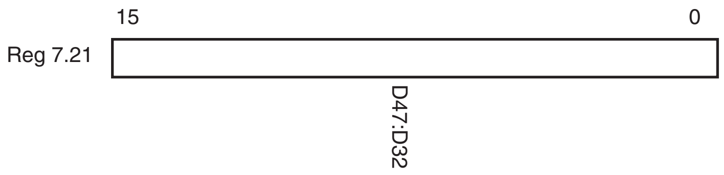


Figure 2-49: **AN LP Base Page Ability Register 2**

Table 2-64 shows the AN LP Base Page Ability register bit definitions.

Table 2-64: **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 2-50 shows the MDIO Register 7.22: AN XNP Transmit.

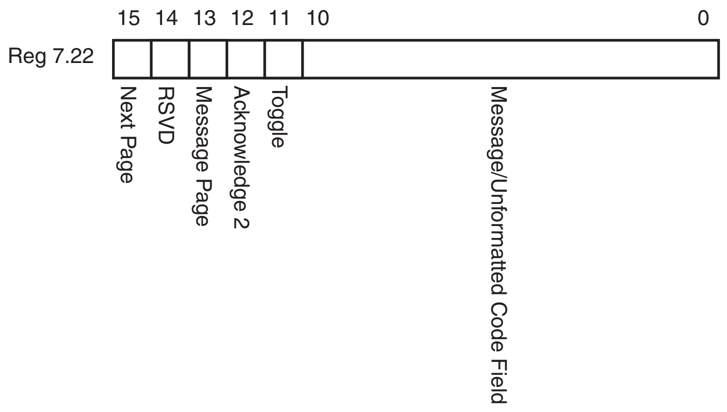


Figure 2-50: **AN XNP Transmit Register 0**

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

Table 2-65: **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	0s

Figure 2-51 shows the MDIO Register 7.23: AN XNP Transmit.

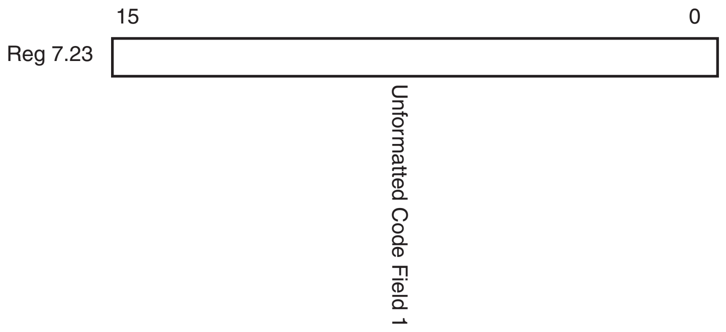


Figure 2-51: **AN XNP Transmit Register 1**

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

Table 2-66: **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	0s

Figure 2-52 shows the MDIO Register 7.24: AN XNP Transmit.

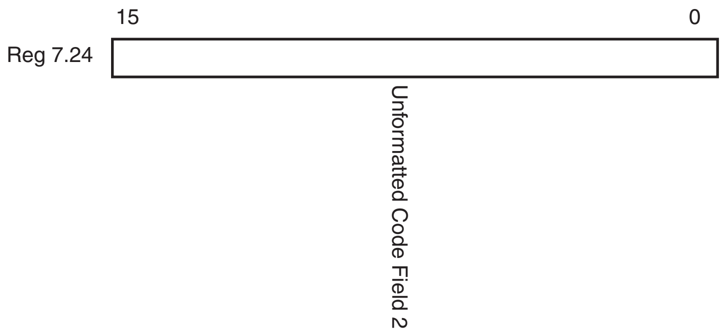


Figure 2-52: **AN XNP Transmit Register 2**

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

Table 2-67: **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	0s

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

Figure 2-53 shows the MDIO Register 7.25: AN LP XNP Ability.

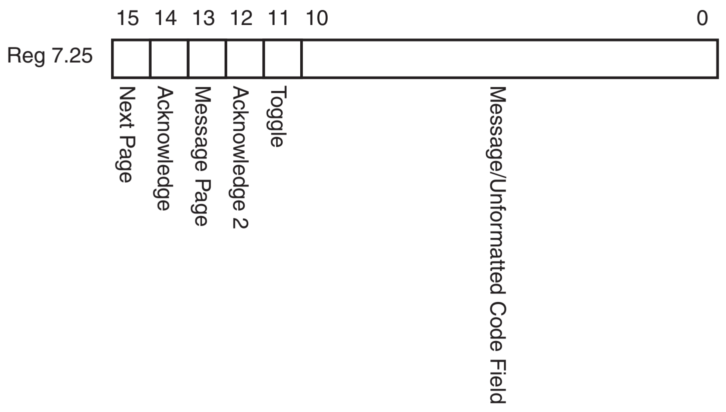


Figure 2-53: **AN LP XNP Ability Register 0**

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

Table 2-68: **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	0s

Figure 2-54 shows the MDIO Register 7.26: AN LP XNP Ability.

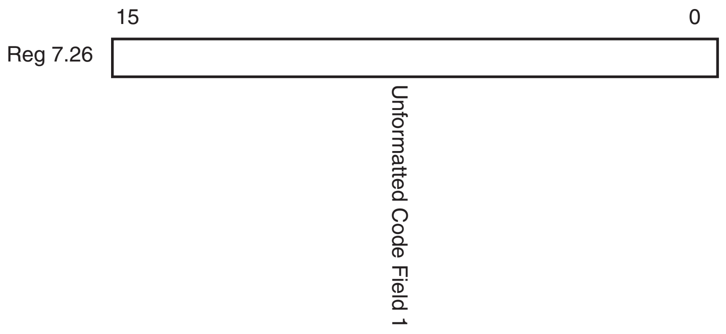


Figure 2-54: **AN LP XNP Ability Register 1**

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

Table 2-69: **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	0s

Figure 2-55 shows the MDIO Register 7.27: AN LP XNP Ability.

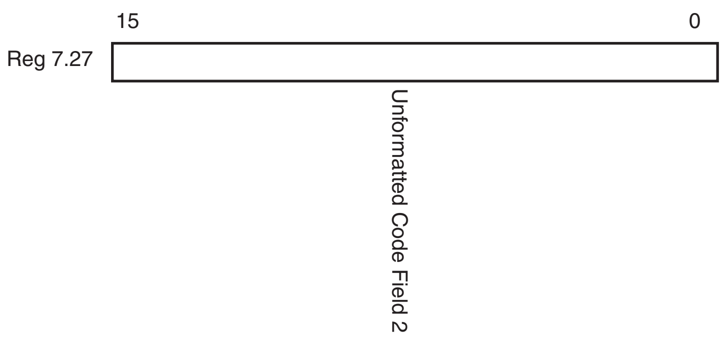


Figure 2-55: **AN LP XNP Ability Register 2**

Table 2-70 shows the AN LP XNP Ability register bit definitions.

Table 2-70: **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	0s

MDIO Register 7.48: Backplane Ethernet Status

Figure 2-56 shows the MDIO Register 7.48: Backplane Ethernet Status.

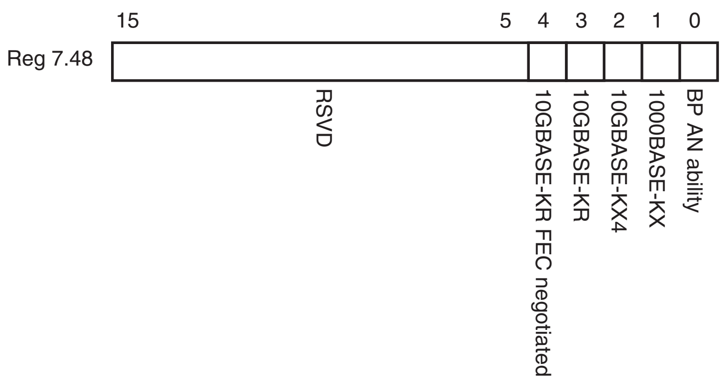


Figure 2-56: **Backplane Ethernet Status Register**

Table 2-71 shows the Backplane Ethernet Status register bit definitions.

Table 2-71: **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 perform 10GBASE-KR FEC 0 = not negotiated	R/O	0
7.48.3	10GBASE-KR	1 = PMA/PMD is negotiated to perform 10GBASE-KR 0 = not negotiated	R/O	0
7.48.2	10GBASE-KX4	1 = PMA/PMD is negotiated to perform 10GBASE-KX4 0 = not negotiated	R/O	0
7.48.1	1000GBASE-KX	1 = PMA/PMD is negotiated to perform 1000GBASE-KX 0 = not negotiated	R/O	0
7.48.0	BP AN ability	1 = PMA/PMD is able to perform one of the preceding protocols 0 = not able	R/O	1

Designing with the Core

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

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

General Design Guidelines

Use the Example Design as a Starting Point

Each instance of the 10GBASE-R/KR core created by the 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 Chapter 9, 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 timing constraint file that is provided with the example design for the core identifies the critical signals and the timing constraints that should be applied. See Chapter 5, Constraining the Core (Vivado™ design tools) and Chapter 8, Constraining the Core (ISE® tools) for further information.

Use Supported Design Flows

See Chapter 4, Customizing and Generating the Core (Vivado design tools) and Chapter 7, Customizing and Generating the Core (ISE design tools) for more information.

Make Only Allowed Modifications

The 10GBASE-R/KR core is not user-modifiable. Do not make modifications as they can have adverse effects on system timing and protocol compliance. Supported user configurations of the 10GBASE-R/KR core can only be made by selecting the options from within the CORE Generator tool or IP catalog when the core is generated. See Chapter 4, Customizing and Generating the Core (Vivado design tools) and Chapter 7, Customizing and Generating the Core (CORE Generator tool).

Interfacing to the 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 3-1. 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 3-1: **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 3-2 for reference.

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

Data (Hex)	Control	Name, Abbreviation
00 to FF	‘0’	Data (D)
07	‘1’	Idle (I)
FB	‘1’	Start (S)
FD	‘1’	Terminate (T)
FE	‘1’	Error (E)

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 3-1. The beginning of the data frame is shown by the presence of the Start character (the /S/ codegroup in lane 4 of Figure 3-1) 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 3-1). The Terminate character can occur in any lane; the remaining lanes are padded by XGMII idle characters.

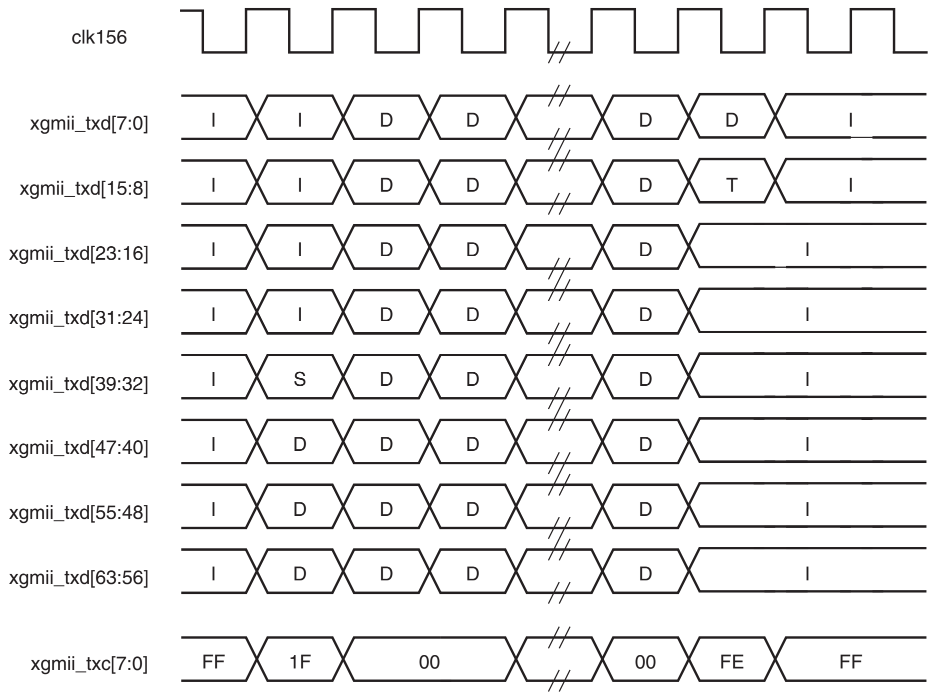


Figure 3-1: **Normal Frame Transmission Across the Internal 64-bit Client-Side I/F**

Figure 3-2 depicts a similar frame to that in Figure 3-1, 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.

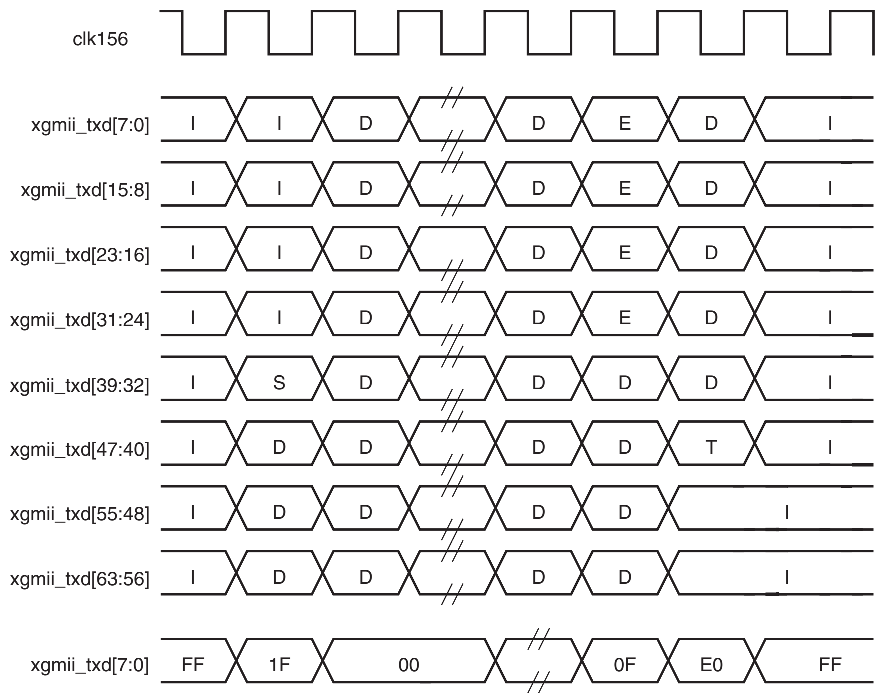


Figure 3-2: **Frame Transmission with Error Across Internal 64-bit Client-Side I/F**

Interfacing to the Receive Client Interface

Internal 64-bit Client-Side Interface

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

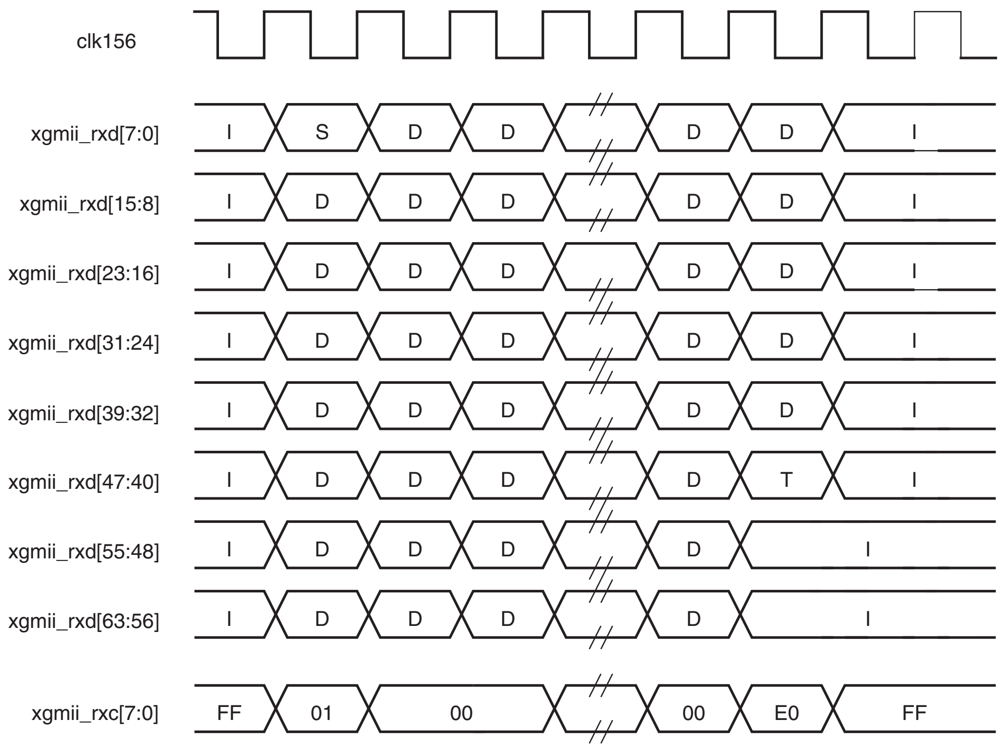


Figure 3-3: **Frame Reception Across the Internal 64-bit Client Interface**

Figure 3-4 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.

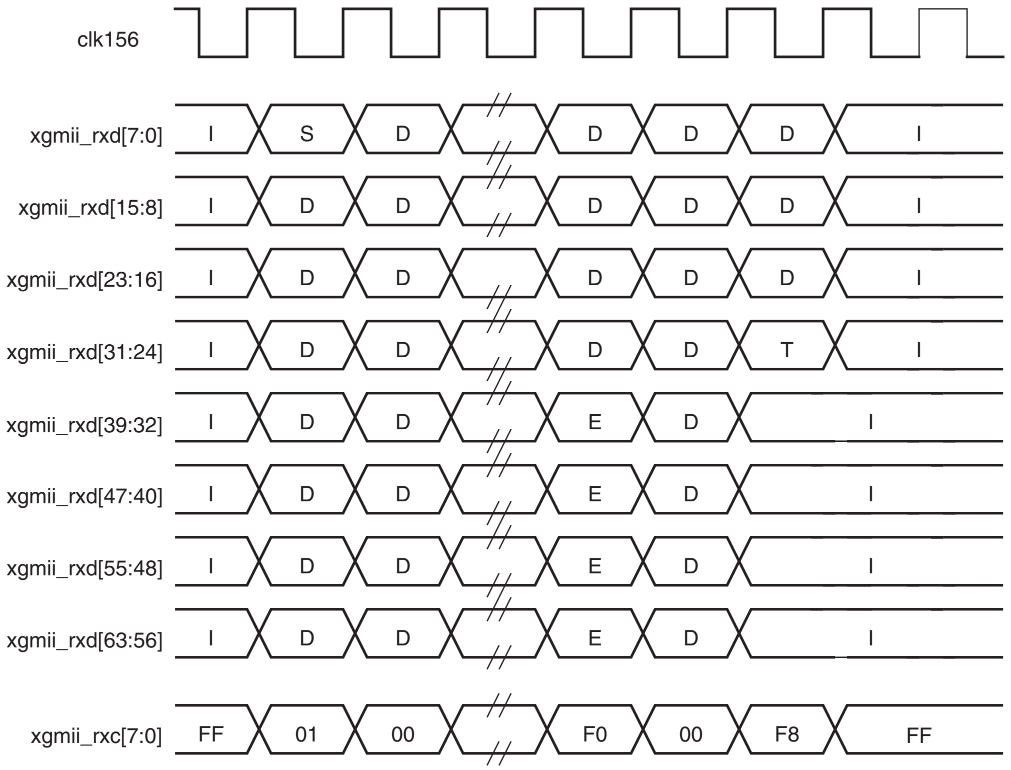


Figure 3-4: **Frame Reception with Error Across the Internal 64-bit Client Interface**

Interfacing to the Transceivers

Virtex-7/Kintex-7 FPGAs

Table 3-3: **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

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

Signal Name	Direction	Description
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 *7 Series Transceiver User Guide* (UG476) for detailed descriptions of the transceiver ports.

Virtex-6 HXT FPGAs

Table 3-4: **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
- Configuration and Status Vectors

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 3-5 illustrates a typical system. All transactions are initiated by the STA entity. The 10GBASE-R/KR core implements an MMD.

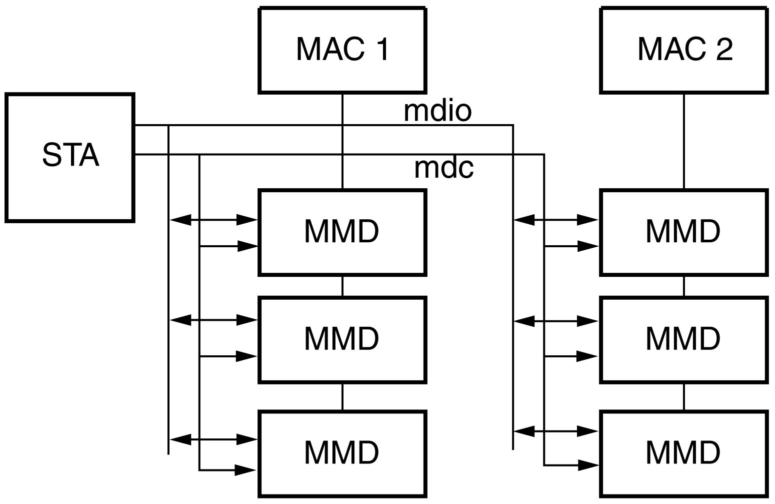


Figure 3-5: **A Typical MDIO-Managed System**

MDIO Ports

The core ports associated with MDIO are shown in Table 3-5.

Table 3-5: **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 3-6 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.

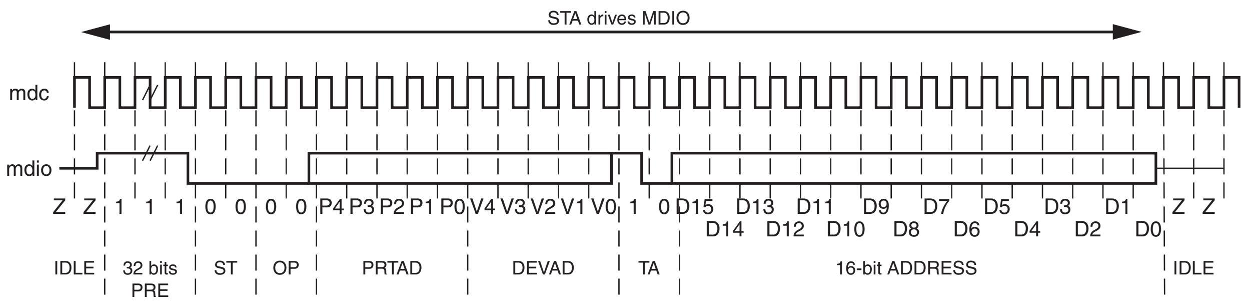

STA drives MDIO
mdc
mdio
Z Z 1 1 1 0 0 0 0 P4 P3 P2 P1 P0 V4 V3 V2 V1 V0 1 0 D15 D13 D11 D9 D7 D5 D3 D1 Z Z
D14 D12 D10 D8 D6 D4 D2 D0
IDLE
32 bits PRE
ST
OP
PRTAD
DEVAD
TA
16-bit ADDRESS
IDLE

Figure 3-6: **MDIO Set Address Transaction**

Write Transaction

Figure 3-7 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.

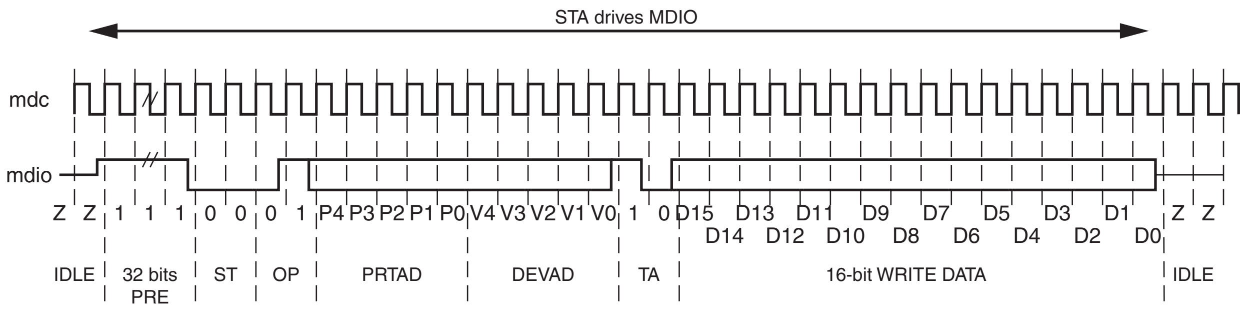

STA drives MDIO
mdc
mdio
Z Z 1 1 1 0 0 0 1 P4 P3 P2 P1 P0 V4 V3 V2 V1 V0 1 0 D15 D13 D11 D9 D7 D5 D3 D1 Z Z
D14 D12 D10 D8 D6 D4 D2 D0
IDLE
32 bits PRE
ST
OP
PRTAD
DEVAD
TA
16-bit WRITE DATA
IDLE

Figure 3-7: **MDIO Write Transaction**

Read Transaction

Figure 3-8 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.

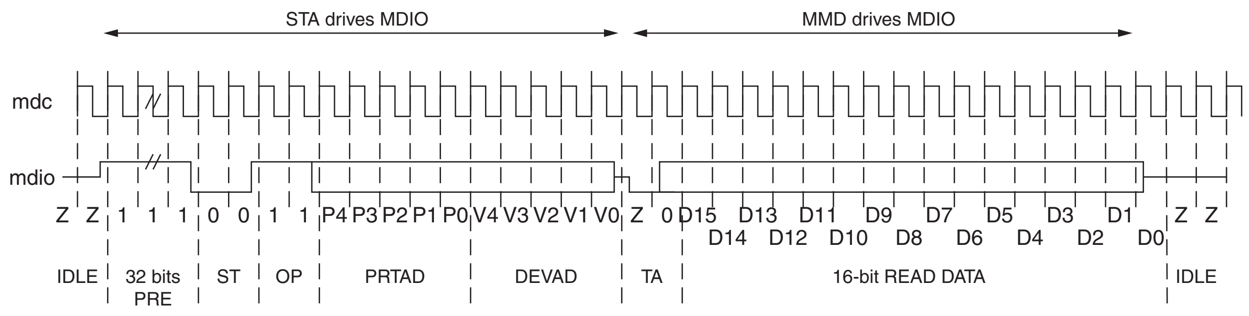

STA drives MDIO
MMD drives MDIO
mdc
mdio
Z Z 1 1 1 0 0 1 1 P4 P3 P2 P1 P0 V4 V3 V2 V1 V0 Z 0 D15 D14 D13 D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0 Z Z
IDLE
32 bits PRE
ST
OP
PRTAD
DEVAD
TA
16-bit READ DATA
IDLE

Figure 3-8: **MDIO Read Transaction**

Post-Read-increment-address Transaction

Figure 3-9 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.

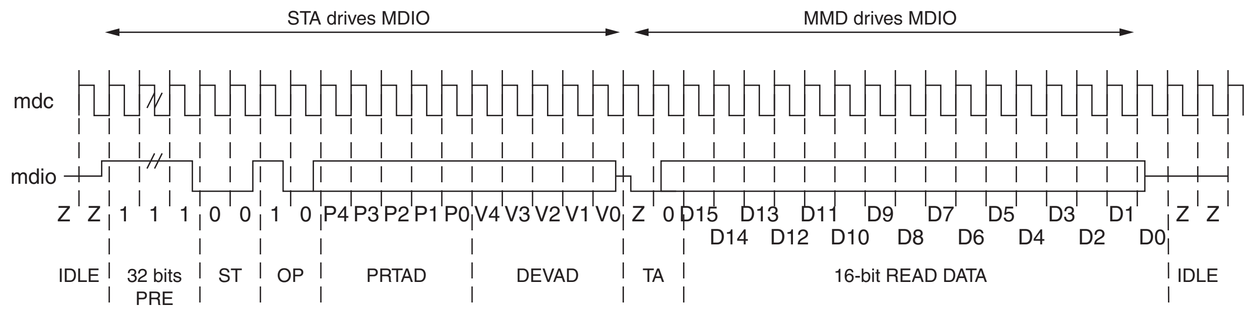

STA drives MDIO
MMD drives MDIO
mdc
mdio
Z Z 1 1 1 0 0 1 0 P4 P3 P2 P1 P0 V4 V3 V2 V1 V0 Z 0 D15 D14 D13 D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0 Z Z
IDLE
32 bits PRE
ST
OP
PRTAD
DEVAD
TA
16-bit READ DATA
IDLE

Figure 3-9: **MDIO Read-and-increment Transaction**

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.

See Register Space in Chapter 2 which describes each of the registers that is emulated with these configuration and status vectors. Clicking the IEEE Register column entries will link to the relevant register description.

Read the notes at the end of this section relating to clearing latching information bits and counters. Some IEEE registers are defined as set/clear-on-read, and because there is no 'read' when using the configuration and status vectors, special controls have been provided to imitate that behavior.

BASE-R

Table 3-6 shows the breakdown of the 10GBASE-R-specific configuration vector and Table 3-7 shows the breakdown of the status vector. Any bits not mentioned are assumed to be 0s.

Table 3-6: **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	1.32788.3:2 1.32788.1:0	PMA Vendor Specific Loopback Mode (HXT only)
110	3.0.14	PCS Loopback Enable
111	3.0.15	PCS Reset(1)
169:112	3.37-3.34	MDIO Register 3.34-37: 10GBASE-R Test Pattern Seed A0-3
233:176	3.41-3.38	MDIO Register 3.38-41: 10GBASE-R Test Pattern Seed B0-3
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 3.32768.14	PCS Loopback Type (Virtex-6 HXT FPGA only)
399:384	3.65535.15:0	125 μs timer control (Virtex-7/Kintex-7 FPGAs only)
512	(1.1.2) (2)	Set PMA Link Status
513	(1.8.11) (2) (1.8.10) (2)	Clear PMA/PMD Link Faults (Virtex-7/Kintex-7 FPGAs only)
516	(3.1.2) (2)	Set PCS Link Status
517	(3.8.11) (2) (3.8.10) (2)	Clear PCS Link Faults (K7/V7 only)
518	(3.33) (2)	MDIO Register 3.33: Clear 10GBASE-R Status 2

Table 3-6: **Configuration Vector - BASE-R *(Cont'd)***

Bit	IEEE Register	Description
519	(3.43) [2]	MDIO Register 3.43: Clear 10GBASE-R Test Pattern Error Counter

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

Table 3-7: **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] [4]
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][4]
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
303:288	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 transceiver.
4. This bit is a logical OR of two latching bits and so will exhibit latching behavior without actually being latching itself.

BASE-KR

Table 3-8 shows the additional signals in the configuration vector which are specific to BASE-KR functionality.

Table 3-8: **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	MDIO Register 1.152: 10GBASE-KR LP Coefficient Update
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)]
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	MDIO Register: 1.65520: Vendor-Specific LD Training
412	1.65520.12	LD Training Initialize
413	1.65520.13	LD Training Preset
415	1.65520.15	Training Done
514	(1.173:172)[(3)]	MDIO Register 1.173: 10GBASE-R FEC Corrected Blocks (Upper) MDIO Register 1.172: 10GBASE-R FEC Corrected Blocks (Lower)
515	(1.175:174)[(3)]	MDIO Register 1.175: 10GBASE-R FEC Uncorrected Blocks (Upper) MDIO Register 1.174: 10GBASE-R FEC Uncorrected Blocks (Lower)
520	(7.1.2)[(3)]	Set AN Link Up/Down
521	(7.1.4)[(3)]	Clear AN Remote Fault
522	(7.1.6)[(3)]	Clear AN Page Received
523	(7.18:16)[(4)]	MDIO Register 7.16:17:18: AN Advertisement

Table 3-8: **Configuration Vector - BASE-KR Specific *(Cont'd)***

Bit	IEEE Register	Description
524	(7.24:22)[4]	MDIO Register 7.22, 23, 24: AN XNP Transmit

1. Only valid when the optional FEC block is included
2. Only valid when the optional AN block is included
3. Reset controls for the given registers
4. Toggle to load the AN Page data from the associated configuration vector bits

Table 3-9 shows the additional signals in the status vector which are specific to BASE-KR functionality.

Table 3-9: **Status Vector - BASE-KR Specific**

Bit	IEEE Register	Description
67:64	1.151.3:0	MDIO Register 1.151: 10GBASE-KR PMD Status
85:80	1.153.5:0	MDIO Register 1.153: 10GBASE-KR LP Status
95	1.153.15:14	LP Status Report Training Complete
101:96	1.152.5:0	MDIO Register 1.152: 10GBASE-KR LP Coefficient Update
108	1.152.12	LP Coefficient Initialize
109	1.152.13	LP Coefficient Preset
117:112	1.155.5:0	MDIO Register 1.155: 10GBASE-KR LD Status
127	1.155.15	LD Status Report Training Complete
159:128	1.173:172	[1] MDIO Register 1.173: 10GBASE-R FEC Corrected Blocks (Upper)[1] MDIO Register 1.172: 10GBASE-R FEC Corrected Blocks (Lower)[1]
191:160	1.175:174	MDIO Register 1.175: 10GBASE-R FEC Uncorrected Blocks (Upper)[1] MDIO Register 1.174: 10GBASE-R FEC Uncorrected Blocks (Lower)[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 (latching high)[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	MDIO Register 7.19, 20, 21: AN LP Base Page Ability
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]

Table 3-9: **Status Vector - BASE-KR Specific *(Cont'd)***

Bit	IEEE Register	Description
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 3-10 shows how the status bit is cleared.

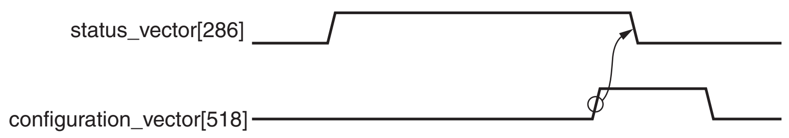


Figure 3-10: **Clearing the Latching-High Bits**

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 3-11 shows how the status bits are set.

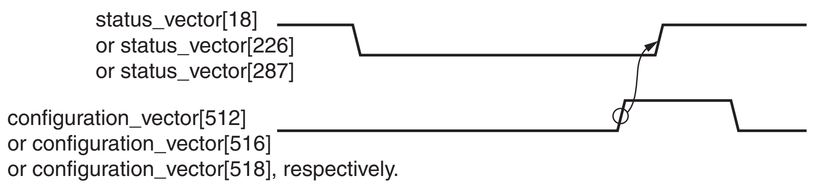

status_vector[18]
or status_vector[226]
or status_vector[287]
configuration_vector[512]
or configuration_vector[516]
or configuration_vector[518], respectively.

Figure 3-11: **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.

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

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

- status vector bits 159:128 - cleared with configuration vector bit 514;
- status vector bits 191:160 - cleared with configuration vector bit 515;
- status vector bit 322 - set with configuration vector bit 520;

- status vector bit 324 - cleared with configuration vector bit 521;
- status vector bit 326 - cleared with configuration vector bit 522.

Finally, configuration vector bits 335:293 and 383:336 each implement three 16-bit registers which are normally latched into the core when the lower register is written, keeping the data coherent. Because there is no need for this behavior when the entire vector is exposed, these bits are latched into the core whenever configuration register bits 523 and 524 respectively are toggled high.

Clocking

Virtex-7/Kintex-7 FPGAs

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 3-12.

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 transceiver interface 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 3-12 shows a possible clocking architecture with a single GTXE2_CHANNEL or GTHE2_CHANNEL block.

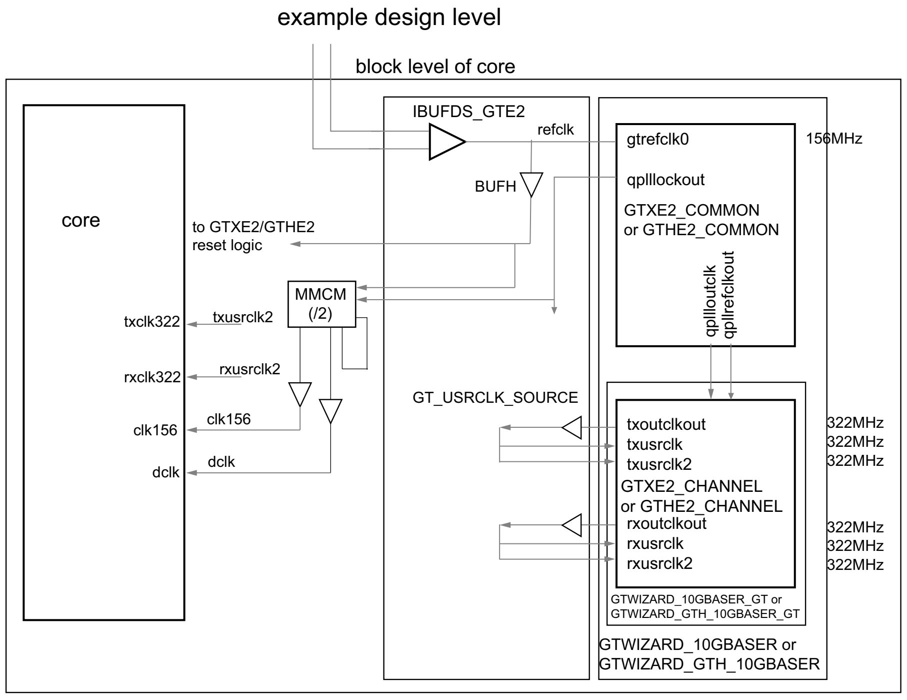


Figure 3-12: **Clocking Scheme for Internal Client-Side Interface: 7 Series FPGAs**

Virtex-6 FPGAs

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 *Virtex-6 FPGA GTH Transceivers User Guide*. (UG371).

Internal Client-Side Interface

The clocking scheme for the internal client interface is shown in Figure 3-13.

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 *Virtex-6 FPGA GTH Transceivers User Guide* (UG371) for details about this clock.

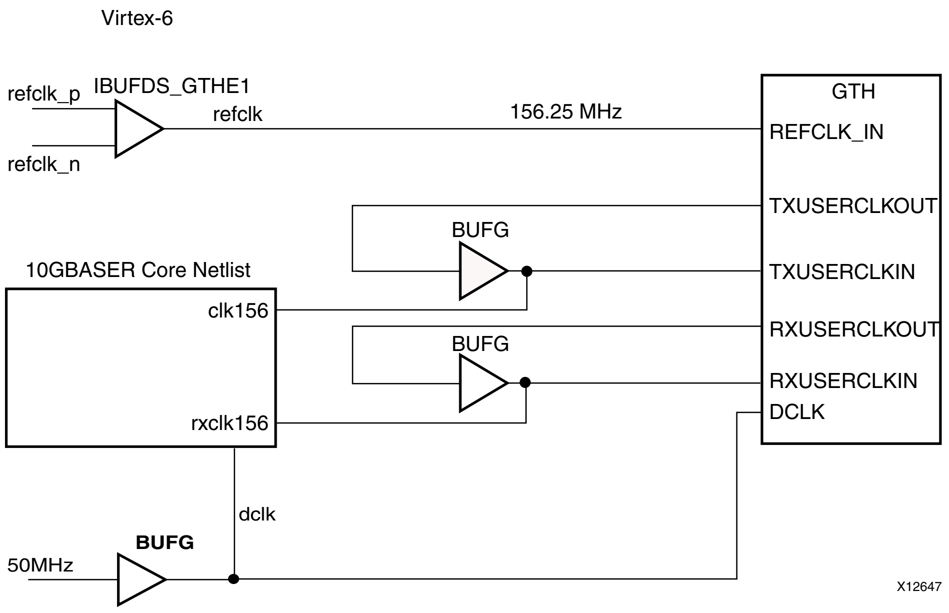


Figure 3-13: **Clock Scheme for Internal Client-Side Interface: Virtex-6 HXT FPGAs**

Resets

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* (UG476).

Virtex-6 FPGAs

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

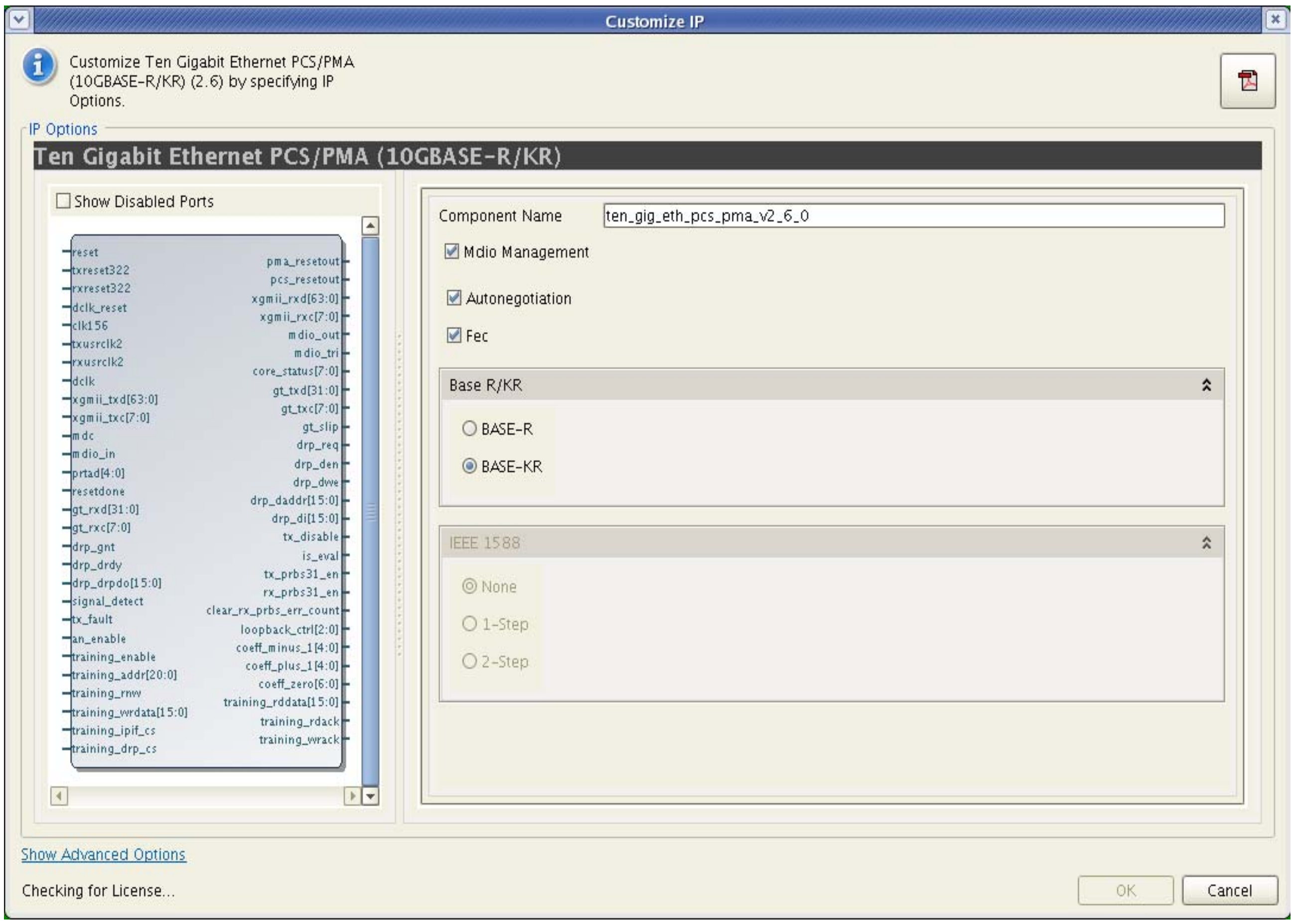
SECTION II: VIVADO DESIGN SUITE

Customizing and Generating the Core

This chapter includes information on using Xilinx tools to customize and generate the core using the Vivado™ design tools.

GUI

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



Customize Ten Gigabit Ethernet PCS/PMA (10GBASE-R/KR) (2.6) by specifying IP Options.

Ten Gigabit Ethernet PCS/PMA (10GBASE-R/KR)

☐ Show Disabled Ports

Component Name: ten_gig_eth_pcs_pma_v2_6_0

☑ Mdio Management

☑ Autonegotiation

☑ Fec

Base R/KR: BASE-R [✓ BASE-KR]

IEEE 1588: [✓ None] 1-Step 2-Step

Checking for License...

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

Component Name

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

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.

Output Generation

The core has various selectable output products. These can be generated by right-clicking on the customized piece of IP in the **Sources** window.

- **Examples** - Source HDL and constraints for the example project
- **Simulation** - Simulation source files
- **Synthesis** - Synthesis source files
- **Examples Simulation** - Test bench for the example design
- **Instantiation Template** - Example instantiation template for the core level module.
- **Miscellaneous** - Simulation scripts and support files required for running netlist based functional simulation. The files delivered as part of this filegroup are not used or understood by Vivado design tools. These files are delivered into the project source directory.

Constraining the Core

This chapter contains information about constraining the core using the Vivado™ Design Suite.

Required Constraints

This section defines the constraint requirements for the core. Constraints are provided in an XDC file which is provided with the HDL example design to give a starting point for constraints for the user design. The required constraints are shown in the following sections.

Device, Package, and Speed Grade Selections

You are only able to generate the core for supported device, package and speed grade combinations; -1 speed grades are not supported for this core.

Clock Frequencies

The core requires a 156.25 MHz reference clock:

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

set clk156name [get_clocks -of_objects [get_pins ten_gig_eth_pcs_pma_block/clkgen_i/
CLKOUT0]]
set dclkname [get_clocks -of_objects [get_pins ten_gig_eth_pcs_pma_block/clkgen_i/
CLKOUT1]]
```

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

```
create_clock -name RXOUTCLK_OUT -period 3.103 [get_pins -of_objects [get_cells *
-hierarchical -filter {REF_NAME=~ GTXE2_CHANNEL}] -filter {NAME =~ *RXOUTCLK}]
create_clock -name TXOUTCLK_OUT -period 3.103 [get_pins -of_objects [get_cells *
-hierarchical -filter {REF_NAME=~ GTXE2_CHANNEL}] -filter {NAME =~ *TXOUTCLK}]
```

Alternatively, these constraints are required for devices with GTHE2 transceivers:

```
create_clock -name RXOUTCLK_OUT -period 3.103 [get_pins -of_objects [get_cells *
-hierarchical -filter {REF_NAME=~ GTHE2_CHANNEL}] -filter {NAME =~ *RXOUTCLK}]
create_clock -name TXOUTCLK_OUT -period 3.103 [get_pins -of_objects [get_cells *
-hierarchical -filter {REF_NAME=~ GTHE2_CHANNEL}] -filter {NAME =~ *TXOUTCLK}]
```

The following multicycle paths must be declared only for BaseKR cores:

```
set_multicycle_path 2 -from [get_cells -of [filter [all_fanout -flat -endpoints_only
-from [get_nets * -hierarchical -filter {NAME =~ *training_inst/txusrclk2_en156}]]
{NAME =~ *CE}]] -to [get_cells -of [filter [all_fanout -flat -endpoints_only -from
[get_nets * -hierarchical -filter {NAME =~ *training_inst/txusrclk2_en156}]] {NAME
=~ *CE}]]
set_multicycle_path -hold 1 -from [get_cells -of [filter [all_fanout -flat
-endpoints_only -from [get_nets * -hierarchical -filter {NAME =~ *training_inst/
txusrclk2_en156}]] {NAME =~ *CE}]] -to [get_cells -of [filter [all_fanout -flat
-endpoints_only -from [get_nets * -hierarchical -filter {NAME =~ *training_inst/
txusrclk2_en156}]] {NAME =~ *CE}]]
```

The following multicycle paths must be declared for all cores:

```
set_multicycle_path 2 -from [get_cells -of [filter [all_fanout -flat -endpoints_only
-from [get_nets * -hierarchical -filter {NAME =~ *rxusrclk2_en156*}]] {NAME =~ *CE}]]
-to [get_cells -of [filter [all_fanout -flat -endpoints_only -from [get_nets *
-hierarchical -filter {NAME =~ *rxusrclk2_en156*}]] {NAME =~ *CE || NAME =~ *WE}]]

set_multicycle_path -hold 1 -from [get_cells -of [filter [all_fanout -flat
-endpoints_only -from [get_nets * -hierarchical -filter {NAME =~
*rxusrclk2_en156*}]] {NAME =~ *CE}]] -to [get_cells -of [filter [all_fanout -flat
-endpoints_only -from [get_nets * -hierarchical -filter {NAME =~
*rxusrclk2_en156*}]] {NAME =~ *CE || NAME =~ *WE}]]
```

The following false paths must be declared because Vivado™ design tools consider all clocks related by default.

```
set_false_path -through [get_nets * -hierarchical -filter {NAME =~
*ten_gig_eth_pcs_pma_block* && NAME =~ *elastic_buffer_i?can_insert_wra}]

set_false_path -from [get_cells -hierarchical -filter {NAME =~
*ten_gig_eth_pcs_pma_block* && PRIMITIVE_SUBGROUP =~ flop}] -to [get_pins
-of_objects [get_cells -hierarchical -filter {NAME =~ *_gt0_rxusrclk2_i_*}] -filter
{NAME =~ *PRE}]

set_false_path -from [get_cells -hierarchical -filter {NAME =~
*ten_gig_eth_pcs_pma_block* && PRIMITIVE_SUBGROUP =~ flop}] -to [get_pins
-of_objects [get_cells -hierarchical -filter {NAME =~ *_gt0_txusrclk2_i_*}] -filter
{NAME =~ *PRE}]

set_false_path -from [get_cells -hierarchical -filter {NAME =~
*ten_gig_eth_pcs_pma_block* && PRIMITIVE_SUBGROUP =~ flop}] -to [get_pins
-of_objects [get_cells -hierarchical -filter {NAME =~ *_gt0_rxusrclk2_i_*}] -filter
{NAME =~ *CLR}]
```

```
set_false_path -from [get_cells -hierarchical -filter {NAME =~
*ten_gig_eth_pcs_pma_block* && PRIMITIVE_SUBGROUP =~ flop}] -to [get_pins
-of_objects [get_cells -hierarchical -filter {NAME =~ *_gt0_txusrclk2_i_*}] -filter
{NAME =~ *CLR}]

set_false_path -from [get_cells -hierarchical -filter {NAME =~ *core_reset* &&
PRIMITIVE_SUBGROUP =~ flop}] -to [get_pins -of_objects [get_cells -hierarchical
-filter {NAME =~ *rxreset322_tmp_reg}] -filter {NAME =~ *D}]

set_false_path -from [get_cells -hierarchical -filter {NAME =~ *core_reset* &&
PRIMITIVE_SUBGROUP =~ flop}] -to [get_pins -of_objects [get_cells -hierarchical
-filter {NAME =~ *txreset322_tmp_reg}] -filter {NAME =~ *D}]

set_false_path -from [get_cells -hierarchical -filter {NAME =~ *core_reset* &&
PRIMITIVE_SUBGROUP =~ flop}] -to [get_pins -of_objects [get_cells -hierarchical
-filter {NAME =~ *dclk_reset_tmp_reg}] -filter {NAME =~ *D}]

set_false_path -from [get_cells -hierarchical -filter {NAME =~
*ten_gig_eth_pcs_pma_block*cable_unpull_enable* && PRIMITIVE_SUBGROUP =~ flop}] -to
[get_pins -of_objects [get_cells -hierarchical -filter {NAME =~
*ten_gig_eth_pcs_pma_block* && PRIMITIVE_SUBGROUP =~ flop}] -filter {NAME =~ *D}]

set_false_path -from [get_cells -hierarchical -filter {NAME =~
*ten_gig_eth_pcs_pma_block* && PRIMITIVE_SUBGROUP =~ flop}] -to [get_pins
-of_objects [get_cells -hierarchical -filter {NAME =~ *psynch_*newedge_reg_reg}]
-filter {NAME =~ *D}]

set_false_path -from [get_cells -hierarchical -filter {NAME =~
*ten_gig_eth_pcs_pma_block* && PRIMITIVE_SUBGROUP =~ flop}] -to [get_pins
-of_objects [get_cells -hierarchical -filter {NAME =~ *psynch_*q_reg}] -filter {NAME
=~ *D}]

set_false_path -from [get_cells -hierarchical -filter {NAME =~
*ten_gig_eth_pcs_pma_block* && PRIMITIVE_SUBGROUP =~ flop}] -to [get_pins
-of_objects [get_cells -hierarchical -filter {NAME =~ *cable_pull_reset_reg_reg}]
-filter {NAME =~ *D}]

set_false_path -from [get_cells -hierarchical -filter {NAME =~
*ten_gig_eth_pcs_pma_block* && PRIMITIVE_SUBGROUP =~ flop}] -to [get_pins
-of_objects [get_cells -hierarchical -filter {NAME =~ *cable_unpull_reset_reg_reg}]
-filter {NAME =~ *D}]
set_false_path -from [get_cells -hierarchical -filter {NAME =~
*ten_gig_eth_pcs_pma_block* && PRIMITIVE_SUBGROUP =~ gt}] -to [get_pins -of_objects
[get_cells -hierarchical -filter {NAME =~ *txresetdone_i_rega_reg}] -filter {NAME =~
*D}]

set_false_path -from [get_cells -hierarchical -filter {NAME =~
*ten_gig_eth_pcs_pma_block* && PRIMITIVE_SUBGROUP =~ gt}] -to [get_pins -of_objects
[get_cells -hierarchical -filter {NAME =~ *rxresetdone_i_rega_reg}] -filter {NAME =~
*D}]
```

The following false paths are required only for BaseKR cores:

```
set_false_path -from [get_clocks RXOUTCLK_OUT] -to [get_cells -hierarchical -filter
{NAME =~ *prbs11_reg* && PRIMITIVE_SUBGROUP =~ flop}]
```

Only for BaseKR cores which include the optional FEC block:

```
set_false_path -from [get_cells -hierarchical -filter {NAME =~
*ten_gig_eth_pcs_pma_block* && PRIMITIVE_SUBGROUP =~ flop}] -to [get_pins
-of_objects [get_cells -hierarchical -filter {NAME =~ *fec_ten_sync*d1_reg}] -filter
{NAME =~ *D}]
```

Only for BaseKR cores which include the optional AN block:

```
set_false_path -from [get_cells -hierarchical -filter {NAME =~
*ten_gig_eth_pcs_pma_block* && PRIMITIVE_SUBGROUP =~ flop}] -to [get_pins
-of_objects [get_cells -hierarchical -filter {NAME =~
*gt0_rxlpmen_i_syncrx_tmp_reg}] -filter {NAME =~ *D}]
```

These constraints are only required if the optional MDIO interface is not included.

```
set_false_path -from [get_clocks $clk156name] -to [get_ports pcs_loopback]
set_false_path -from [get_ports pcs_loopback] -to [get_clocks $clk156name]
```

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

```
create_generated_clock -name ddrclock -divide_by 1 -invert -source [get_pins
*rx_clk_ddr/C] [get_ports xgmii_rx_clk]
set_output_delay -max 1.500 -clock ddrclock [get_ports * -filter {NAME =~
*xgmii_rxd*}]
set_output_delay -min -1.500 -clock ddrclock [get_ports * -filter {NAME =~
*xgmii_rxd*}]
set_output_delay -max 1.500 -clock ddrclock [get_ports * -filter {NAME =~
*xgmii_rxc*}]
set_output_delay -min -1.500 -clock ddrclock [get_ports * -filter {NAME =~
*xgmii_rxc*}]
```

Some maximum delay paths must be defined:

```
set_max_delay -from [get_cells * -hierarchical -filter {NAME =~
*ten_gig_eth_pcs_pma_block* && NAME =~ *rd_truegray_reg*}] -to [get_cells
-hierarchical -filter {NAME =~ *ten_gig_eth_pcs_pma_block* && NAME =~
*rag_writesync0_reg*}] -datapath_only 5.800

set_max_delay -from [get_cells -hierarchical -filter {NAME =~
*ten_gig_eth_pcs_pma_block* && NAME =~ *wr_gray_reg*}] -to [get_cells -hierarchical
-filter {NAME =~ *ten_gig_eth_pcs_pma_block* && NAME =~ *wr_gray_rdclk0_reg*}]
-datapath_only 5.800

set_max_delay -from [get_cells * -hierarchical -filter {NAME =~
*ten_gig_eth_pcs_pma_block* && NAME =~ *rd_lastgray_reg*}] -to [get_cells *
-hierarchical -filter {NAME =~ *ten_gig_eth_pcs_pma_block* && NAME =~
*rd_lastgray_wrclk0_reg*}] -datapath_only 5.800

set_max_delay -from [get_cells -hierarchical -filter {NAME =~
*ten_gig_eth_pcs_pma_block* && REF_NAME =~ RAMD32}] -to [get_pins -of_objects
[get_cells -hierarchical -filter {NAME =~ *dp_ram_i*fd_i*}] -filter {NAME =~ *D}]
-datapath_only 2.400
```

```
set_max_delay -from [get_cells -hierarchical -filter {NAME =~
*ten_gig_eth_pcs_pma_block* && PRIMITIVE_SUBGROUP =~ flop}] -to [get_pins
-of_objects [get_cells -hierarchical -filter {NAME =~
*elastic_buffer*can_insert_fdr3}] -filter {NAME =~ *D}] -datapath_only 2.400
set_max_delay -from [get_cells -hierarchical -filter {NAME =~
*ten_gig_eth_pcs_pma_block* && PRIMITIVE_SUBGROUP =~ flop}] -to [get_pins
-of_objects [get_cells -hierarchical -filter {NAME =~ *synch_*d1_reg}] -filter {NAME
=~ *D}] -datapath_only 2.400

set_max_delay -from [get_cells -hierarchical -filter {NAME =~
*ten_gig_eth_pcs_pma_block* && PRIMITIVE_SUBGROUP =~ flop}] -to [get_pins
-of_objects [get_cells -hierarchical -filter {NAME =~ *wr_addr1_reg*}] -filter {NAME
=~ *D}] -datapath_only 2.400
```

These set_max_delay constraints must be used when the optional MDIO interface is included:

```
set_max_delay -from [get_cells -hierarchical -filter {NAME =~
*ten_gig_eth_pcs_pma_block* && NAME =~ *reg_3_65535* && PRIMITIVE_SUBGROUP =~ flop}]
-to [get_pins -of_objects [get_cells -hierarchical -filter {NAME =~
*timer_125us_reg*}] -filter {NAME =~ *D}] -datapath_only 6.000

set_max_delay -from [get_cells -hierarchical -filter {NAME =~
*ten_gig_eth_pcs_pma_block* && PRIMITIVE_SUBGROUP =~ flop}] -to [get_pins
-of_objects [get_cells -hierarchical -filter {NAME =~ *mdc_reg1_reg}] -filter {NAME
=~ *D}] -datapath_only 2.400
```

This set_max_delay constraint is required is the optional MDIO interface is not included, for BaseKR cores only:

```
set_max_delay -from [get_cells -hierarchical -filter {NAME =~ *config_vec_serial* &&
PRIMITIVE_SUBGROUP =~ flop}] -to [get_pins -of_objects [get_cells -hierarchical
-filter {NAME =~ *synch_*d1_reg}] -filter {NAME =~ *D}] -datapath_only 2.400
```

Clock Management

Any clock management tiles (MMCMs) are exposed in the source code in the example design files for the core, enabling sharing these resources with other cores and designs.

Clock Placement

Location constraints for MMCMs might be required depending on the rest of the system.

Banking

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

Transceiver Placement

Transceivers should be given location constraints appropriate to your design. An example of these LOC constraints can be found in the XDC file.

I/O Standard and Placement

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

These constraints are required if the optional MDIO interface is included:

```
set_property IOB TRUE [get_cells * -hierarchical -filter {NAME =~ *mdc_reg1*}]
set_property IOB TRUE [get_cells * -hierarchical -filter {NAME =~ *mdio_in_reg1*}]
set_property IOB TRUE [get_cells * -filter {NAME =~ *mdio_out*reg*}]
set_property IOB TRUE [get_cells * -filter {NAME =~ *mdio_tri*reg*}]
```

Detailed Example Design

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

Example Design

This section shows an example HDL wrapper for Virtex®-7/Kintex™-7 FPGAs.

In Figure 6-1, 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)

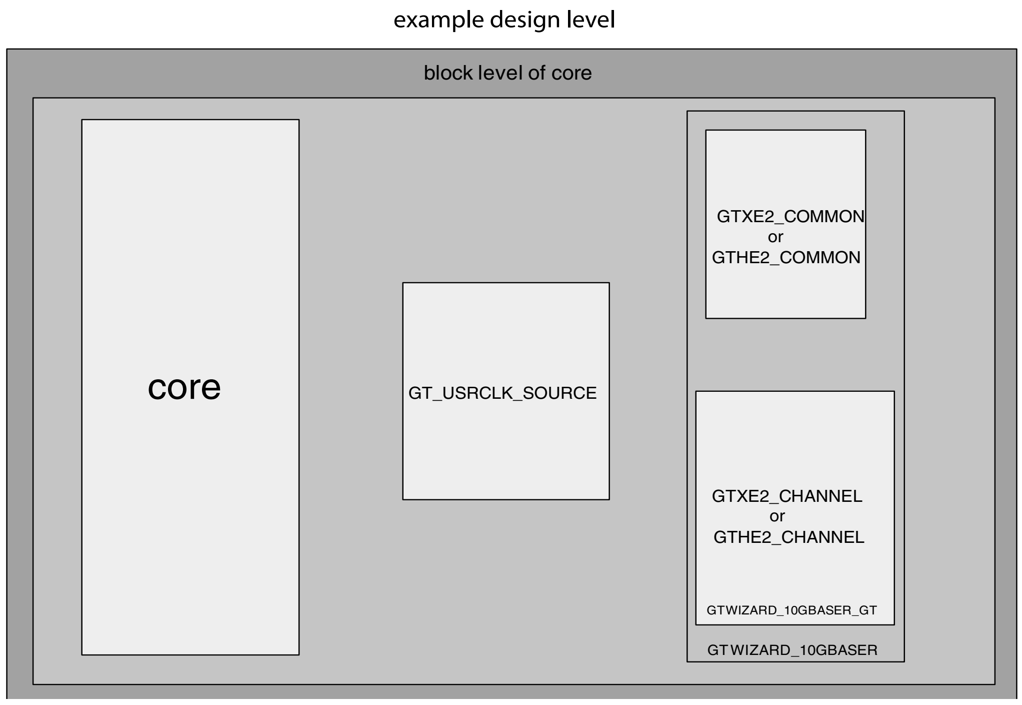


Figure 6-1: **Example HDL Wrapper for 10GBASE-R (Virtex-7/Kintex-7 FPGAs)**

Demonstration Test Bench

In Figure 6-2, the demonstration test bench is a simple VHDL or Verilog program to exercise the example design and the core itself. This test bench consists of transactor procedures or tasks that connect to the major ports of the example design, and a control program that pushes frames of varying length and content through the design and checks the values as they exit the core. The test bench is supplied as part of the Example Simulation output product group.

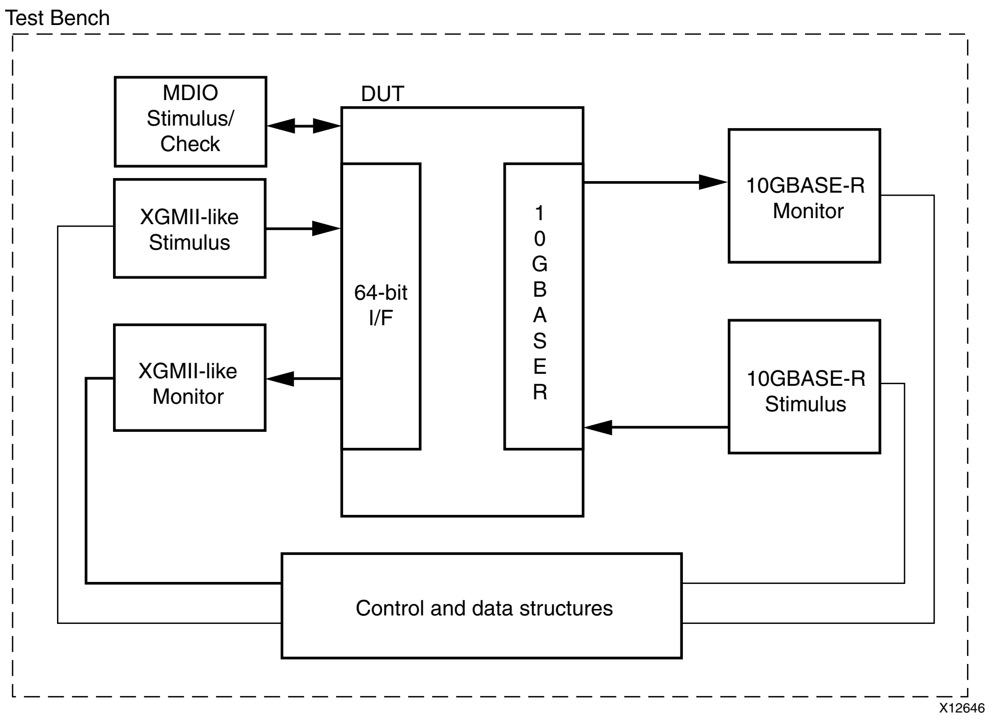


Figure 6-2: **Demonstration Test Bench for 10GBASE-R**

Implementation

See the Vivado Design Suite - 2012.4 User Guides web page.

Simulation

See the Vivado Design Suite - 2012.4 User Guides web page.

SECTION III: ISE DESIGN SUITE

Chapter 7

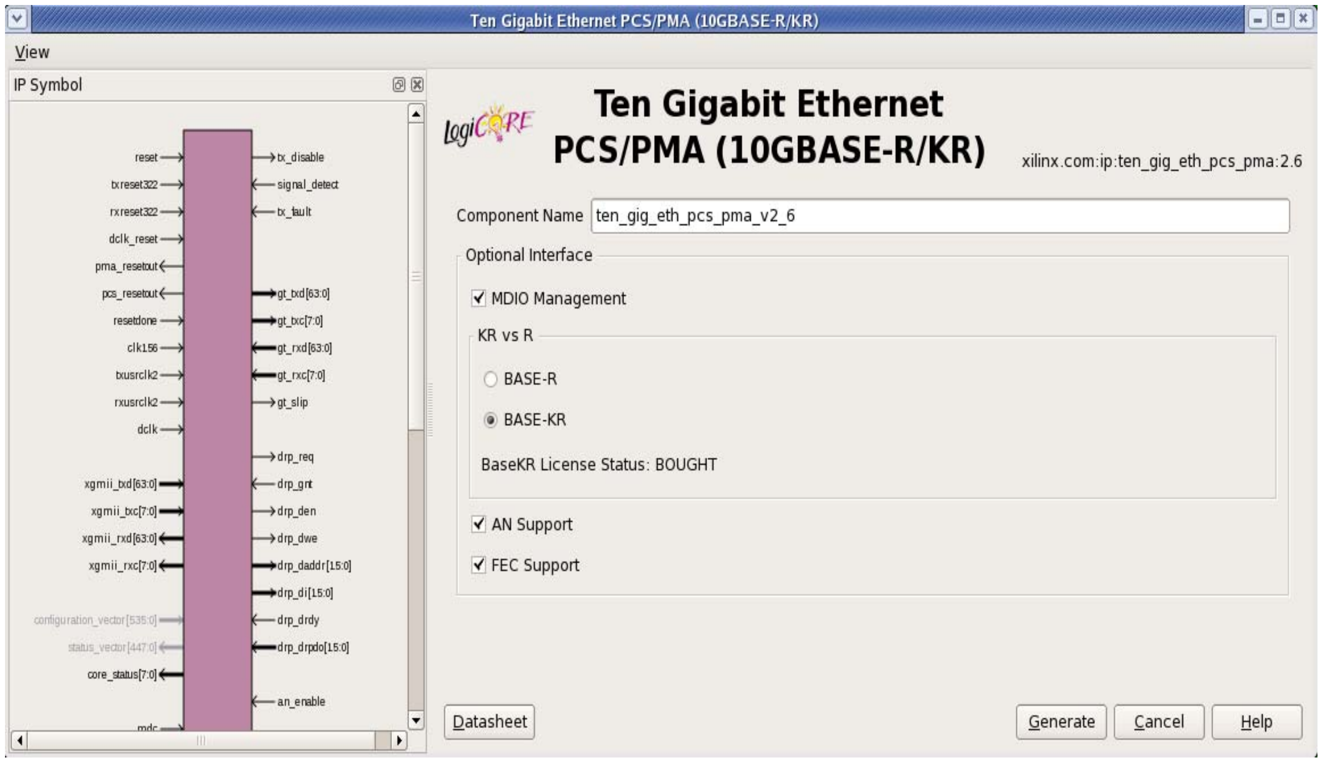
Customizing and Generating the Core

This chapter includes information about using Xilinx tools to customize and generate the core in the ISE® Design Suite environment.

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

GUI

Figure 7-1 displays the main screen for customizing the 10GBASE-R/KR core.



Ten Gigabit Ethernet PCS/PMA (10GBASE-R/KR)

xilinx.com:ip:ten_gig_eth_pcs_pma:2.6

Component Name: ten_gig_eth_pcs_pma_v2_6

Optional Interface

✓ MDIO Management

KR vs R

BASE-R [✓ BASE-KR]

BaseKR License Status: BOUGHT

✓ AN Support

✓ FEC Support

Figure 7-1: **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 7-1 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:

```
# BEGIN Select
SELECT Ten_Gigabit_Ethernet_PCS/PMA_(10GBASE-R/KR)
xilinx.com:ip:ten_gig_eth_pcs_pma:2.6
# END Select
# BEGIN Parameters
CSET autonegotiation=true
CSET base_kr=BASE-KR
CSET component_name=ten_gig_eth_pcs_pma_v2_6
CSET fec=true
CSET mdio_management=true
# END Parameters
```

Table 7-1: **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_6
mdio_management	TRUE, FALSE	TRUE
base_kr (1)	BASE-R, BASE-KR	BASE-KR
fec (2)	TRUE, FALSE	TRUE
autonegotiation(2)	TRUE, FALSE	TRUE

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
- Hardware Description Language (HDL) example design
- Scripts to synthesize, implement and simulate the example design.

- **<project directory>**
  Top-level project directory; name is user-defined
  - <project directory>/<component name>
    Core release notes readme file
    - <component_name>/doc
      Product documentation
    - <component_name>/example_design
      Verilog and VHDL design files
    - <component name>/implement
      Implementation script files
      - implement/results
        Contains implement script results
    - <component_name>/simulation
      Simulation script files
      - simulation/functional
        Functional simulation files

<project directory>

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

Table 7-2: **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 in the readme file provided with the core, which can include tool requirements, updates, and issue resolution.

Table 7-3: **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 7-4: **Doc Directory**

Name	Description
<project_dir>/<component_name>/doc	
pg068-ten-gig-eth-pcs-pma.pdf	10GBASER/KR Product Guide

Back to Top

<component_name>/example_design

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

Table 7-5: **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 7-6: **GTX Directory**

Name	Description
<project_dir>/<component_name>/example_design/gtx	
<component_name>_gt_usrclk_source.v[hd] <component_name>_gtwizard_10gbaser.v[hd] <component_name>_gtwizard_10gbaser_gt.v[hd] <component_name>_gtwizard_10gbaser.xco	Wrappers and support files for the transceivers. 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' You can use the xco file provided to regenerate the latest version of the transceiver wrappers. When regenerating the GT_Wizard output, pay close attention to the gt_usrclk_source block generated. This needs to be altered to remove the BUFG for dclk generation because this core requires 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 7-7: **GTH Directory**

Name	Description
<project_dir>/<component_name>/example_design/gth	
<component_name>_gtwizard_gth_10gbaser_gt_usrclk_source.v[hd] <component_name>_gtwizard_gth_10gbaser.v[hd] <component_name>_gtwizard_gth_10gbaser_gt.v[hd] <component_name>_gtwizard_gth_10gbaser.xco	Wrappers and support files for the transceivers. 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_gth_10gbaser' You can use the xco file provided to regenerate the latest version of the transceiver wrappers. When regenerating the GT_Wizard output, pay close attention to the gt_usrclk_source block generated. This needs to be altered to remove the BUFG for dclk generation because this core requires 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 7-8: **GTH Directory**

Name	Description
<project_dir>/<component_name>/example_design/gth	
<component_name>_v6gth_wrapper.v[hd] <component_name>_v6gth_wrapper_quad.v[hd] <component_name>_v6gth_wrapper_gth_init.v[hd] <component_name>_v6gth_wrapper_gth_reset.v[hd] <component_name>_v6gth_wrapper_gth_rx_pcs_cdr_reset.v[hd] <component_name>_v6gth_wrapper_gth_tx_pcs_reset.v[hd] <component_name>_v6gth_wrapper_pulse_synchronizer.v[hd] <component_name>_v6gth_wrapper.xco	Wrappers and support files for the transceivers. 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 7-9: **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 7-10: **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 7-11: **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 7-12: **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.

Table 7-12: **Functional Directory *(Cont'd)***

Name	Description
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

Constraining the Core

This chapter contains information about constraining the core in the ISE® Design Suite environment. It 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.

Required Constraints

See Chapter 9, 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 Selections

The following subsections describe device, package, and speed grades for Virtex®-7, Kintex™-7, and Virtex-6 devices.

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, -2L, 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 GTH transceivers)

Clock Frequencies

Virtex-7/Kintex-7 FPGAs

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

Main Clock Frequencies

This section specifies the main clock frequencies for the design.

```
###################################################################
# Clock frequencies and clock management #
###################################################################

NET "*/txuserclkout" TNM_NET="clk156";
TIMESPEC "TS_clk156" = PERIOD "clk156" 6400 ps;

NET "*/rxusrclkout" 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.

Clock Management

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.

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.

Clock Placement

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;
```

Banking

Banking limitations might apply to the device selected and so you should take care not to break those limitations.

Transceiver Placement

Virtex-7/Kintex-7 FPGAs

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

Virtex-6 HXT FPGAs

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

I/O Standard and Placement

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 (the upper limit is 12.5 MHz).

Detailed Example Design

This chapter contains information about the provided example design in the ISE® Design Suite environment. It 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.

Directory and File Contents

See Output Generation in Chapter 7 for a detailed description of the directories and files.

Example Design

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

In Figure 9-1, 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)

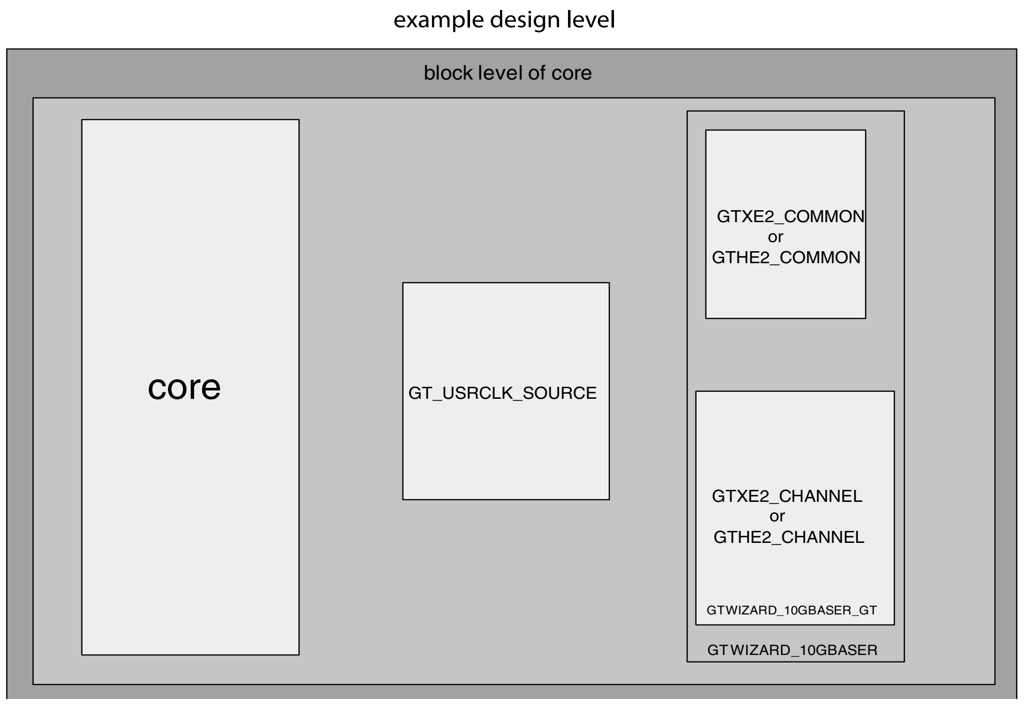


Figure 9-1: **Example HDL Wrapper for 10GBASE-R (Virtex-7/Kintex-7 FPGAs)**

Example HDL Wrapper (Virtex-6 FPGAs)

In Figure 9-2, 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`

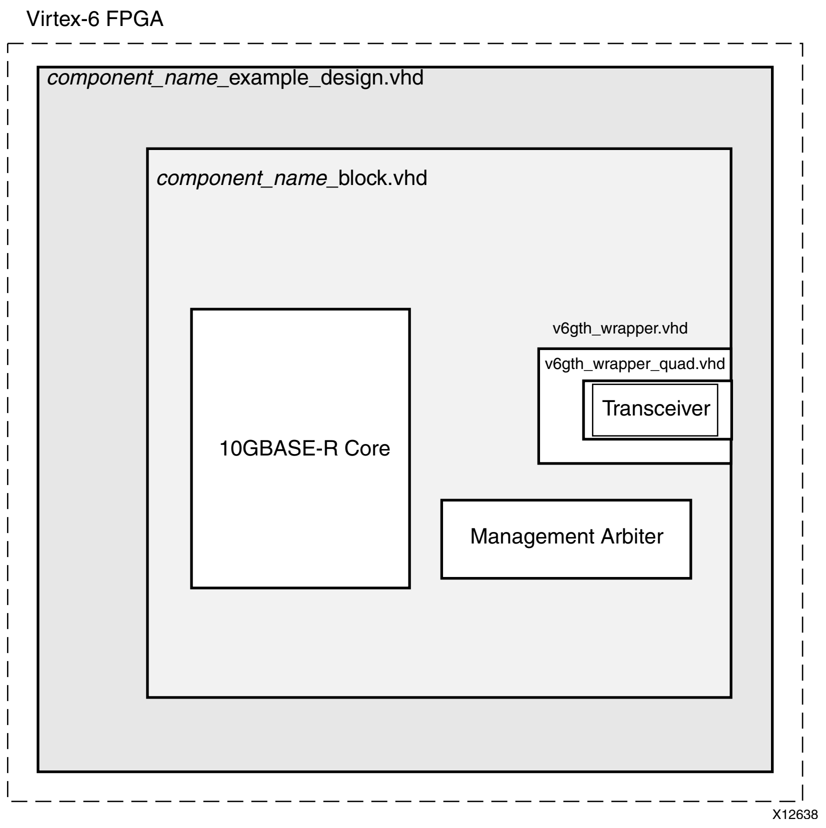


Figure 9-2: **Example HDL Wrapper for 10GBASE-R/KR (Virtex-6 FPGAs)**

Demonstration Test Bench

In Figure 9-3, 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 from the original release. The rtl files `legacy_config_shim` and `legacy_status_shim` can be used to replicate the original bit numbering.

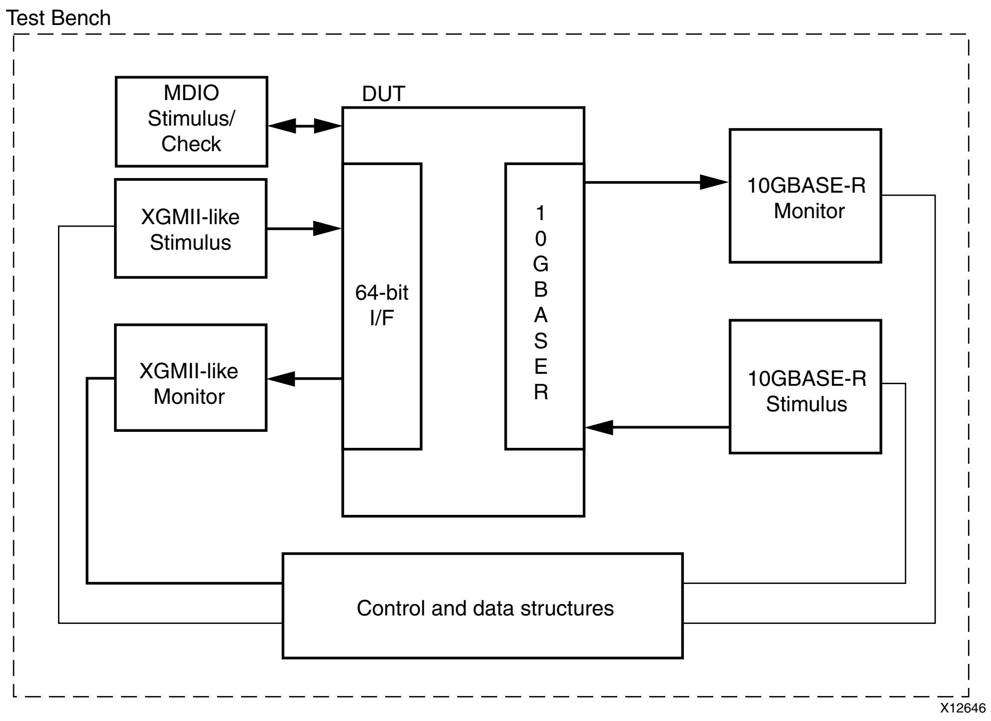


Figure 9-3: **Demonstration Test Bench for 10GBASE-R**

Implementation

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.

Simulation

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).

SECTION IV: APPENDICES

Verification, Compliance, and Interoperability

The 10GBASE-R/KR LogiCORE™ IP core has been verified in ISE® 14.4 and Vivado™ 2012.4 design tools 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.

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

Hardware Testing

The 10GBASE-R core has been validated on Virtex-6 devices using a development/demonstration platform on a single development board that includes a 10 Gigabit Ethernet MAC, a processor and data generation and checking hardware, using near-end loopback in PMA, PCS and in optics. Hundreds of millions of Ethernet frames of varying lengths have been successfully transmitted and received on this platform.

The 10GBASE-R and 10GBASE-KR cores have been validated on Kintex™-7 devices using the same development/demonstration platform, but on two separate development boards. Again, hundreds of millions of ethernet frames have been successfully transmitted and received on each board and other features such as hot-plugging, autonegotiation and FEC error correction have been tested with the setup.

For 10GBASE-KR, successful, uncorrectable-error-free reception across a backplane emulator of 36inches of '4000 13si' trace on a Nelco Quad Serial Loop (NQSL) board has been demonstrated.

Compliance Testing

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

Debugging

This appendix includes details about resources available on the Xilinx Support website and debugging tools. In addition, this appendix provides a step-by-step debugging process and a flow diagram to guide you through debugging the Ten Gigabit Ethernet PCS/PMA core.

The following topics are included in this appendix:

- Finding Help on Xilinx.com
- Debug Tools
- Simulation Debug
- Hardware Debug

Finding Help on Xilinx.com

To help in the design and debug process when using the Ten Gigabit Ethernet PCS/PMA core, the Xilinx Support web page (www.xilinx.com/support) contains key resources such as product documentation, release notes, answer records, information about known issues, and links for opening a Technical Support WebCase.

Documentation

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

Download the Xilinx Documentation Navigator from the Design Tools tab on the Downloads page (www.xilinx.com/download.) For more information about this tool and the features available, open the online help after installation.

Release Notes

Known issues for all cores, including the Ten Gigabit Ethernet PCS/PMA core are described in the IP Release Notes Guide (XTP025).

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 PCS/PMA core is the Xilinx Ethernet IP Solution Center.

Known Issues

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

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

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

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

Answer Records for the Ten Gigabit Ethernet PCS/PMA Core

47698 (www.xilinx.com/support/answers/47698.htm)

Contacting Technical Support

Xilinx provides premier technical support for customers encountering issues that require additional assistance.

To contact Xilinx Technical Support:

1. Navigate to www.xilinx.com/support.
2. Open a WebCase by selecting the WebCase link located under Support Quick Links.

When opening a WebCase, include:

- Target FPGA including package and speed grade.
- All applicable Xilinx Design Tools and simulator software versions.
- Additional files based on the specific issue might also be required. See the relevant sections in this debug guide for guidelines about which file(s) to include with the WebCase.

Debug Tools

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

Example Design

The Ten Gigabit Ethernet PCS/PMA core is delivered with an example design that can be synthesized, complete with functional test benches. Information about the example design can be found in *Chapter 6, Detailed Example Design*.

ChipScope Pro Debugging Tool

The ChipScope™ Pro debugging tool inserts logic analyzer, bus analyzer, and virtual I/O cores directly into your design. The ChipScope Pro debugging tool allows you to set trigger conditions to capture application and integrated block port signals in hardware. Captured signals can then be analyzed through the ChipScope Pro logic analyzer tool. For detailed information for using the ChipScope Pro debugging tool, see www.xilinx.com/tools/cspro.htm.

Vivado Lab Tools

Vivado Lab Tools inserts logic analyzer, bus analyzer, and virtual I/O cores directly into your design. Vivado Lab Tools allows you to set trigger conditions to capture application and integrated block port signals in hardware. Captured signals can then be analyzed.

Reference Boards

Please contact your Xilinx representative for information on development platforms for this IP.

License Checkers

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

- ISE flow: XST, NgdBuild, Bitgen
- Vivado flow: Vivado Synthesis, Vivado Implementation, write_bitstream (Tcl command)

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

Simulation Debug

The simulation debug flow for ModelSim is illustrated in Figure B-1. A similar approach can be used with other simulators.

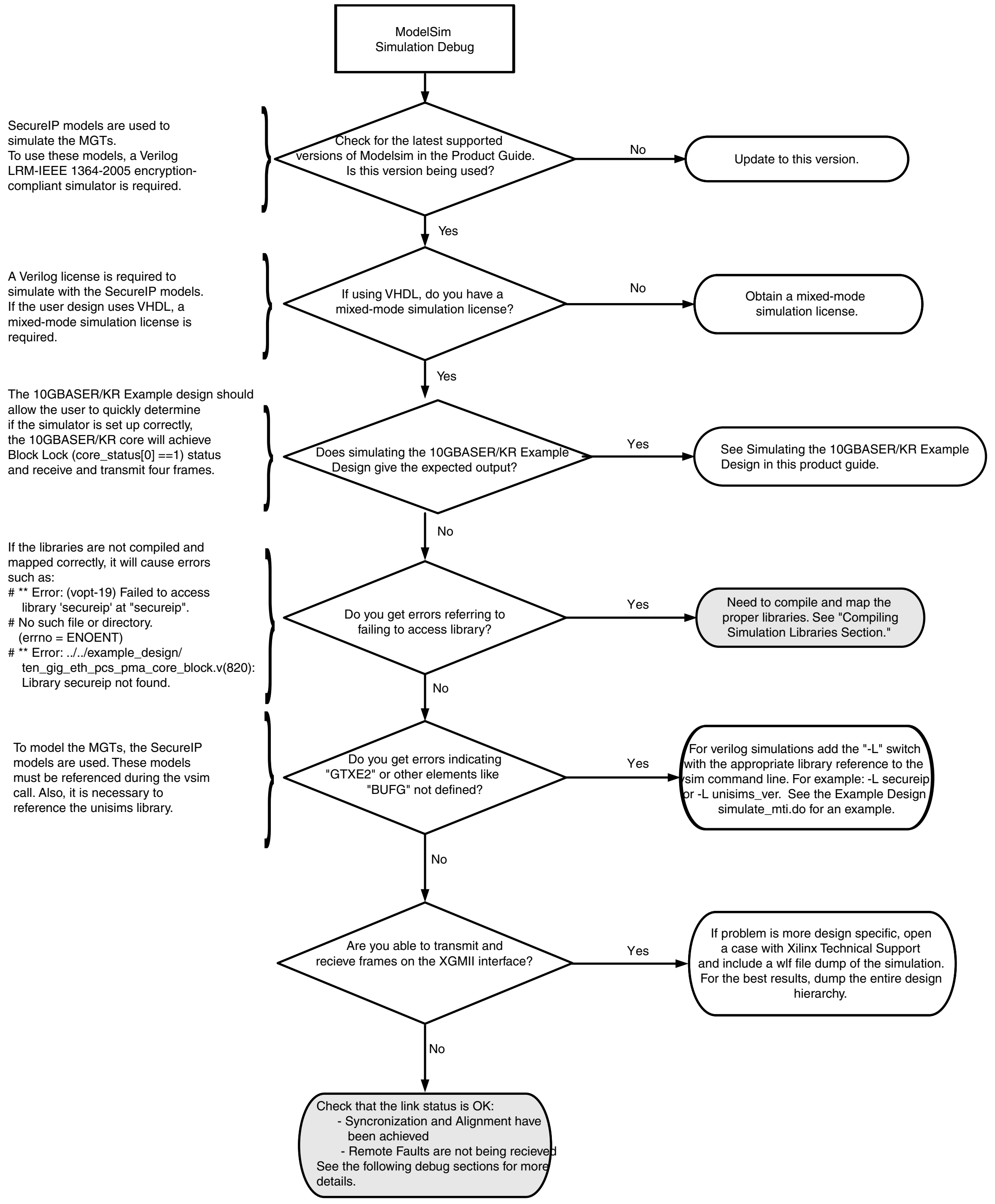

ModelSim Simulation Debug
SecureIP models are used to simulate the MGTs. To use these models, a Verilog LRM-IEEE 1364-2005 encryption-compliant simulator is required.
Check for the latest supported versions of Modelsim in the Product Guide. Is this version being used?
No
Update to this version.
Yes
A Verilog license is required to simulate with the SecureIP models. If the user design uses VHDL, a mixed-mode simulation license is required.
If using VHDL, do you have a mixed-mode simulation license?
No
Obtain a mixed-mode simulation license.
Yes
The 10GBASER/KR Example design should allow the user to quickly determine if the simulator is set up correctly, the 10GBASER/KR core will achieve Block Lock (core_status[0] ==1) status and receive and transmit four frames.
Does simulating the 10GBASER/KR Example Design give the expected output?
Yes
See Simulating the 10GBASER/KR Example Design in this product guide.
No
If the libraries are not compiled and mapped correctly, it will cause errors such as:
# ** Error: (vopt-19) Failed to access library 'secureip' at "secureip".
# No such file or directory. (errno = ENOENT)
# ** Error: ../../example_design/ ten_gig_eth_pcs_pma_core_block.v(820): Library secureip not found.
Do you get errors referring to failing to access library?
Yes
Need to compile and map the proper libraries. See "Compiling Simulation Libraries Section."
No
To model the MGTs, the SecureIP models are used. These models must be referenced during the vsim call. Also, it is necessary to reference the unisims library.
Do you get errors indicating "GTXE2" or other elements like "BUFG" not defined?
Yes
For verilog simulations add the "-L" switch with the appropriate library reference to the vsim command line. For example: -L secureip or -L unisims_ver. See the Example Design simulate_mti.do for an example.
No
Are you able to transmit and recieve frames on the XGMII interface?
Yes
If problem is more design specific, open a case with Xilinx Technical Support and include a wlf file dump of the simulation. For the best results, dump the entire design hierarchy.
No
Check that the link status is OK:
- Syncronization and Alignment have been achieved
- Remote Faults are not being recieved
See the following debug sections for more details.

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

Hardware Debug

Hardware issues can range from link bring-up to problems seen after hours of testing. This section provides debug steps for common issues. The ChipScope debugging tool is a valuable resource to use in hardware debug. The signal names mentioned in the following individual sections can be probed using the ChipScope debugging tool for debugging the specific problems.

Many of these common issues can also be applied to debugging design simulations. Details are provided on:

- General Checks
- What Can Cause a Local or Remote Fault?
- Link Bring Up - Basic
- Link Bring Up - Base-KR
- What Can Cause Block Lock to Fail?
- What Can Cause the 10Gb PCS/PMA Core to Insert Errors?
- Transceiver Specific Checks
- Problems with the MDIO
- Link Training

General Checks

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

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

What Can Cause a Local or Remote Fault?

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

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

Local Fault

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

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

Remote Fault

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

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

Link Bring Up - Basic

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

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

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

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

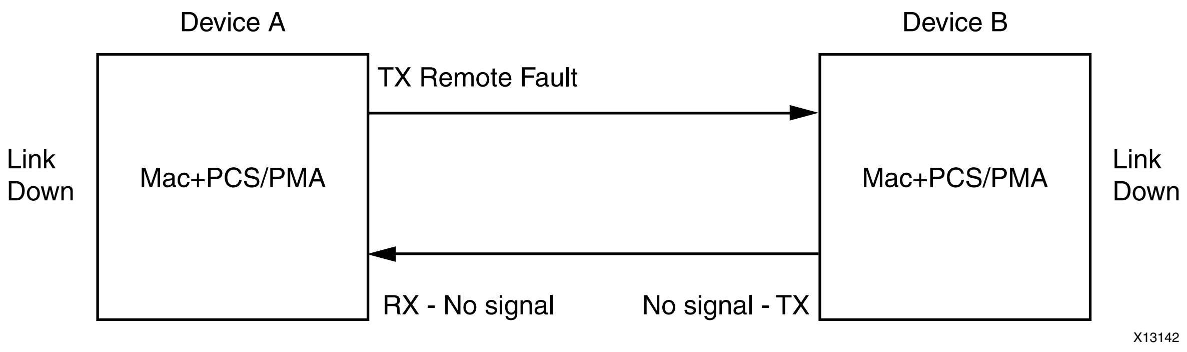


Figure B-2: **Device A Powered Up, but Device B Powered Down**

Stage 2: Device B Powers Up and Resets

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

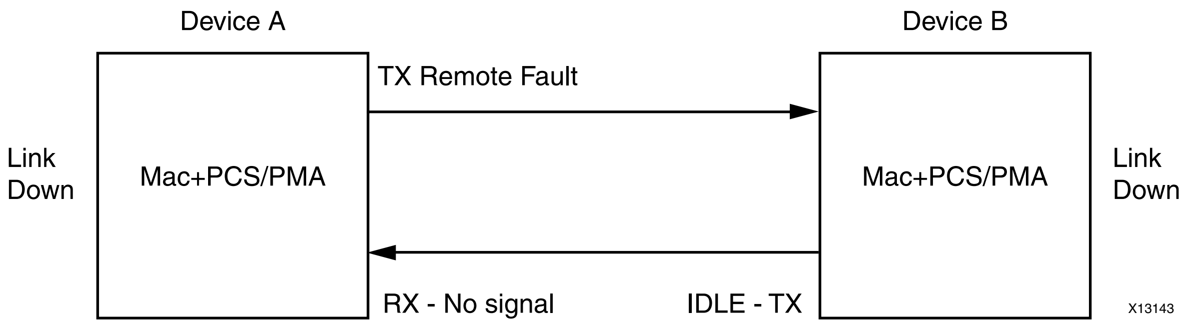


Figure B-3: **Device B Powers Up and Resets**

Stage 3: Device A Receives Idle Sequence

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

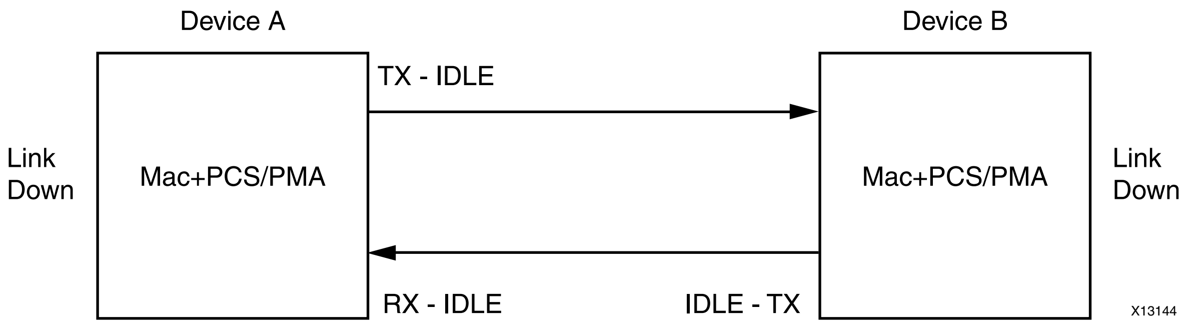


Figure B-4: **Device A Receives Idle Sequence**

Stage 4: Normal Operation

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

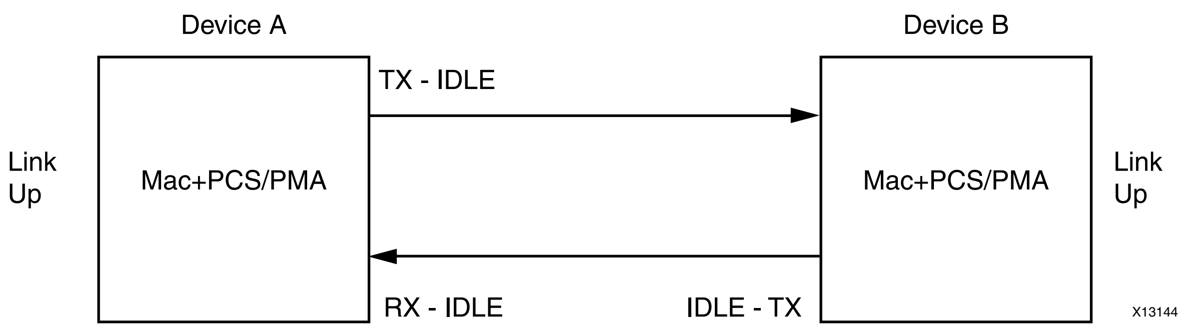


Figure B-5: **Normal Operation**

Link Bring Up - Base-KR

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

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

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

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

Using the Configuration Vector for Link Bring-Up

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

Using the MDIO interface for Link Bring-Up

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

What Can Cause Block Lock to Fail?

Following are suggestions for debugging loss of Block Lock:

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

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

Transceiver-Specific:

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

What Can Cause the 10Gb PCS/PMA Core to Insert Errors?

On the receive path the 10Gb PCS/PMA core receive state machine can insert block errors RXD=FE, RXC=1. The RX block error happens any time the RX_E state is entered into. It could happen if C, S, D, T are seen out of order. Or it could also happen if an /E/ block type is received, (which is defined in IEEE802.3, Section 49.2.13.2.3 as a 66-bit code with a bad sync header or a control word (C S or T) with no matching translation for the block type field.)

Transceiver Specific Checks

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

Problems with the MDIO

See MDIO Interface for detailed information about performing MDIO transactions.

Things to check for:

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

Link Training

There is currently no Link training algorithm included with the core so it is left up to you to implement what you need.

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

When you decide to implement your own Training Algorithm, that should not only include hardware to monitor the received data signal integrity and provide the inc/dec/preset/ initialize commands to send to the far end device, but also must follow the protocol of sending a command until 'updated' is seen on return, and then sending 'hold' until 'not updated' is seen on return. Also, the priority of commands defined in IEEE 802.3 must be adhered to, such as never transmitting 'Preset' with 'Initialize'.

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

For special considerations when using Link Training with Auto Negotiations see:

- Using Training and AutoNegotiation with the MDIO interface in Virtex-7/Kintex-7
- Using Training and AutoNegotiation with No MDIO interface in Virtex-7/Kintex-7

Migrating

See *Vivado Design Suite Migration Methodology Guide* (UG911).

For a complete list of Vivado™ Design Tools User and Methodology Guides, click here.

Parameter Changes in the XCO File

None

Port Changes

None

Functionality Changes

None

Special Design Considerations

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

Connecting Multiple Core Instances

The example design provided with the core shows the use of a single core but it is possible to use this and the block level code to create a design with multiple instances of the core.

It is not possible to simply replicate the block level multiple times because much of the clocking and reset logic will be shared between the multiple instances.

Also, multiple transceiver instances can often share the 'COMMON' blocks between them.

When creating a design with multiple core instances, you need to take care to replicate the correct items and not to replicate items which should be shared.

Virtex-7/Kintex-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 D-1 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.

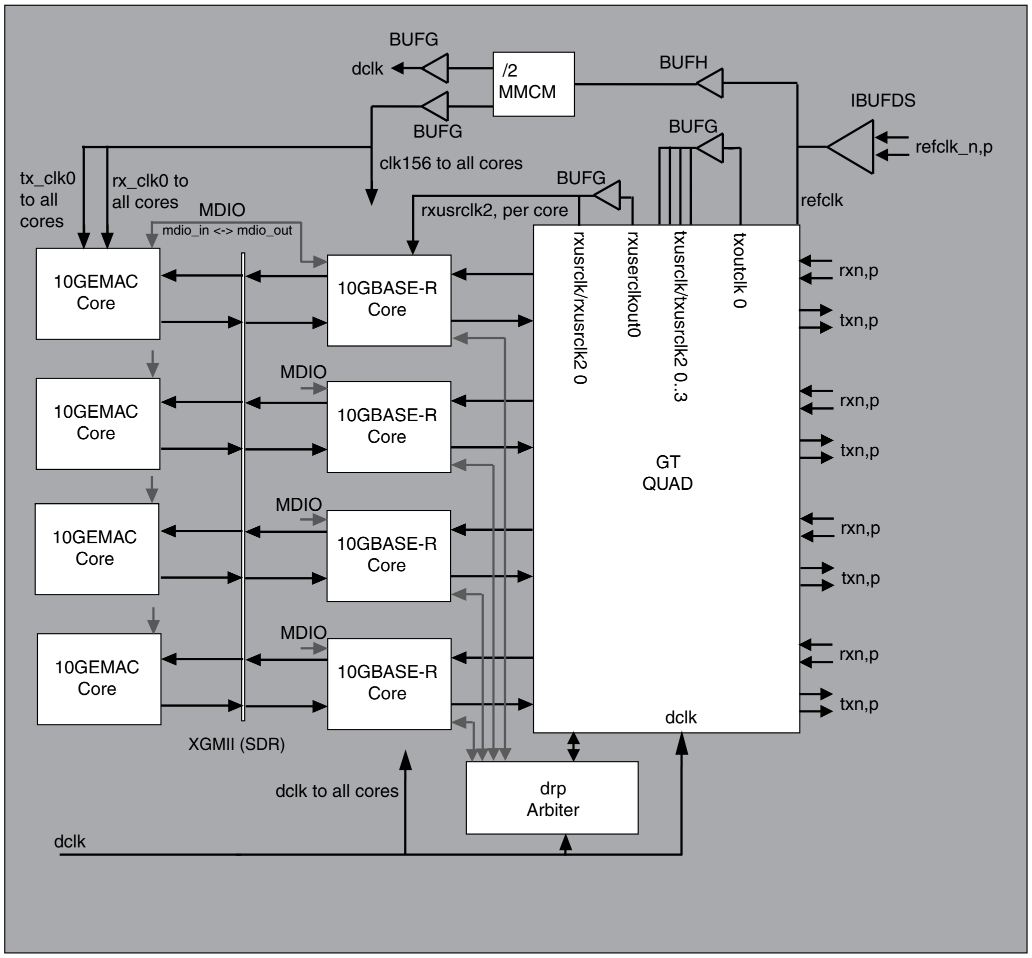


Figure D-1: **Attaching Multiple Cores to a GT_QUAD Tile**

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.

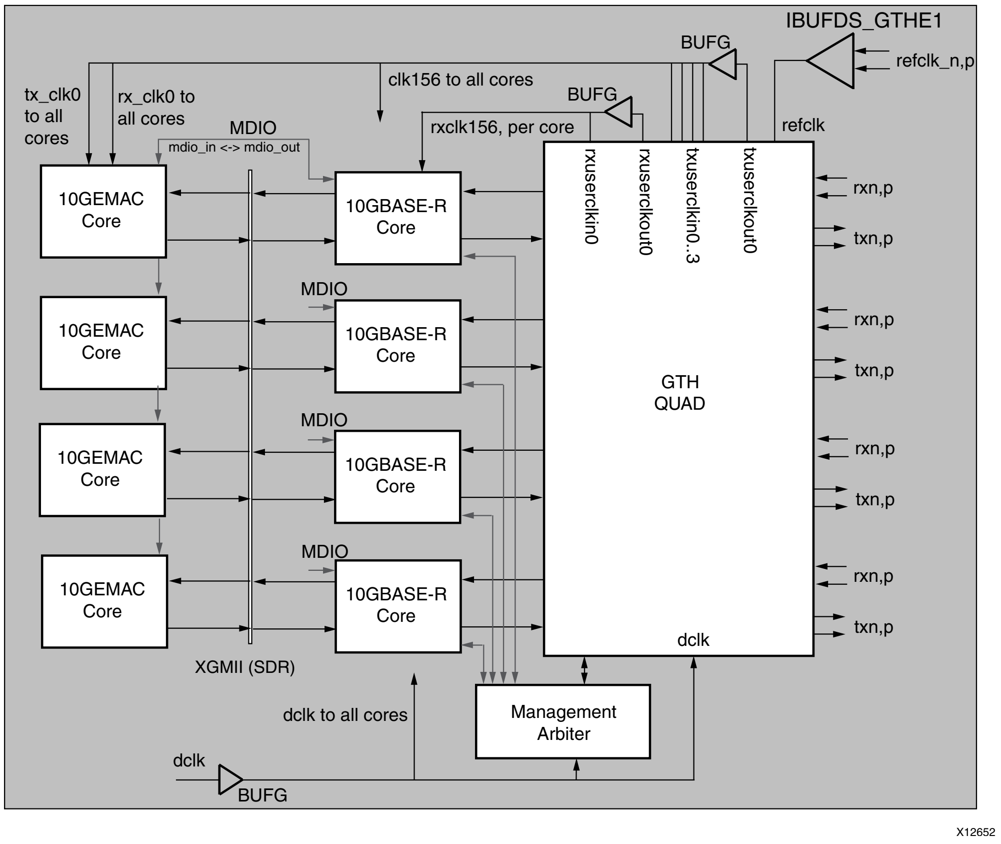


Figure D-2: **Attaching Multiple Cores to a GTH_QUAD Tile**

See *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.

Using Training and AutoNegotiation with the MDIO interface in Virtex-7/Kintex-7

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

Firstly you need to write to the MDIO registers to disable Training (register bit 1.150.1), and then set the Training Restart bit (register bit 1.150.0, which will self-clear).

Then you need to monitor either the core_status[5] output from the core, or register bit 7.1.2 (which latches low and clears on read), to wait for the AN Link Up indication which is set in the AN_GOOD_CHECK state. Now you need to Enable Training (1.150.1) and then immediately Restart Training (1.150.0).

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

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

If Training does not complete within the time allowed by AutoNegotiation, then you must manually disable Training (register 1.150.1) and restart Training (1.150.0) to allow AutoNegotiation to restart the process.

Using Training and AutoNegotiation with No MDIO interface in Virtex-7/Kintex-7

When a Base KR core is created with no MDIO interface, logic in the block level can be used to control the interaction between AutoNegotiation and Training.

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

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

Using FEC in the core with AutoNegotiation

IMPORTANT: *When the FEC feature is recognized in the AutoNegotiation Advertisement (Base Page Ability) data from another 10GBaseKR device, and FEC is requested by the far end, FEC in the core will not be automatically enabled. It is up to you to enable FEC as required.*

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

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/company/terms.htm.

References

These documents provide supplemental material useful with this product guide. Unless otherwise noted, IP references are for the product documentation page.

1. *IEEE Std. 802.3-2008*, Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications
2. *IEEE Std. 802.3-2008*, Media Access Control (MAC) Parameters, Physical Layers, and Management Parameters for 10-Gb/s Operation
3. Vivado™ Design Suite user documentation (www.xilinx.com/cgi-bin/docs/rdoc?v=2012.4;t=vivado+docs)
4. *7 Series Transceivers User Guide* (UG476)
5. *Virtex-6 FPGA GTH Transceivers User Guide* (UG371)
6. *Vivado Design Suite Migration Methodology Guide* (UG911)

To search for Xilinx documentation, go to www.xilinx.com/support.

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

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
07/25/12	1.0	Initial Xilinx product guide release. This document is based on ds739 and ug692.
10/16/12	2.0	• Updated for 14.3 and 2012.3 • Added description of MDIO Register 3.4: PCS Speed Ability • Updated for ease of use of document; Updated core interfaces; Added more descriptions of core usage.
12/18/12	3.0	• Updated for 14.4, 2012.4, and core version 2.6 • Updated Debugging appendix. • Updated false path command and maximum delay paths. • Updated resource numbers. Updated screen captures.

Notice of Disclaimer