LogiCORE IP
10-Gigabit Ethernet
MAC v11.6

Product Guide

PG072 March 20, 2013

XILINX®

Table of Contents

Introduction

The LogiCORE™ IP 10-Gigabit Ethernet MAC core is a single-speed, full-duplex 10 Gb/s Ethernet Media Access Controller (MAC) solution enabling the design of high-speed Ethernet systems and subsystems.

Features

- Choice of external XGMII or internal FPGA interface to PHY layer (internal interface only on Spartan®-6 devices)
- AXI4-Stream protocol support on client transmit and receive interfaces
- Cut-through operation with minimum buffering for maximum flexibility in client-side interfacing
- Supports Deficit Idle Count for maximum data throughput; maintains minimum IFG under all conditions and provides line rate performance
- Supports Deficit Idle Count with In-Band FCS and without In-Band FCS for all devices
- Configured and monitored through an AXI4-Lite Management Interface
- Comprehensive statistics gathering with statistic vector outputs
- Supports flow control in both directions
- Provides MDIO STA master interface to manage PHY layers
- Extremely customizable; trade resource usage against functionality

LogiCORE IP Facts Table	
Core Specifics	
Supported Device Family[1]	Zynq™-7000[2], Virtex®-7, Kintex™-7, Artix™-7, Virtex-6, Spartan-6[3]
Supported User Interfaces	AXI4-Lite, AXI4-Stream
Resources[4]	See Table 2-1 (Virtex-7/Kintex-7), Table 2-2 (Virtex-6), Table 2-3 (Spartan-6)
Provided with Core	
Design Files	NGC Netlist
Example Design	Verilog and VHDL
Test Bench	Verilog and VHDL
Constraints File	UCF
Simulation Model	Verilog and VHDL
Supported S/W Driver	N/A
Tested Design Flows[5]	
Design Entry	ISE Design Suite v14.5
Simulation	Mentor Graphics Questa® SIM Cadence Incisive Enterprise Simulator (IES) Synopsys VCS and VCS MX
Synthesis	Xilinx Synthesis Technology (XST)
Support	
Provided by Xilinx @ www.xilinx.com/support	

Notes:

1. For a complete listing of supported devices, see the release notes for this core. Speed grades are -2 for Artix-7 devices.
2. Only Zynq-7030 and Zynq-7045 devices are supported.
3. External XGMII interface and WAN mode not supported on Spartan-6 devices.
4. Numbers are approximate for default configuration in Virtex-6 devices. See Table 2-1 through Table 2-3 for a complete description of device utilization by configuration and device family.
5. For the supported versions of the tools, see the Xilinx Design Tools: Release Notes Guide.

Features (Continued)

- Supports VLAN, jumbo frames, and WAN mode
- Custom Preamble mode
- Maximum Transmission Unit (MTU) frame length can be set independently for transmit and receive operations.

Overview

The Xilinx LogiCORE™ IP 10-Gigabit Ethernet MAC core is a fully verified solution for the 10-Gigabit per second (Gb/s) Ethernet Media Access Controller function that interfaces to physical layer devices in a 10 Gb/s Ethernet system. The core is designed to the *IEEE Standard 802.3-2008 specification* and supports the high-bandwidth demands of network Internet Protocol (IP) traffic on LAN, MAN, and WAN networks. The core works with the latest Zynq™-7000, Virtex®-7, Kintex™-7, Artix™-7, Virtex-6, and Spartan®-6 devices.

Figure 1-1 illustrates a block diagram of a 10-Gigabit Ethernet MAC core implementation.

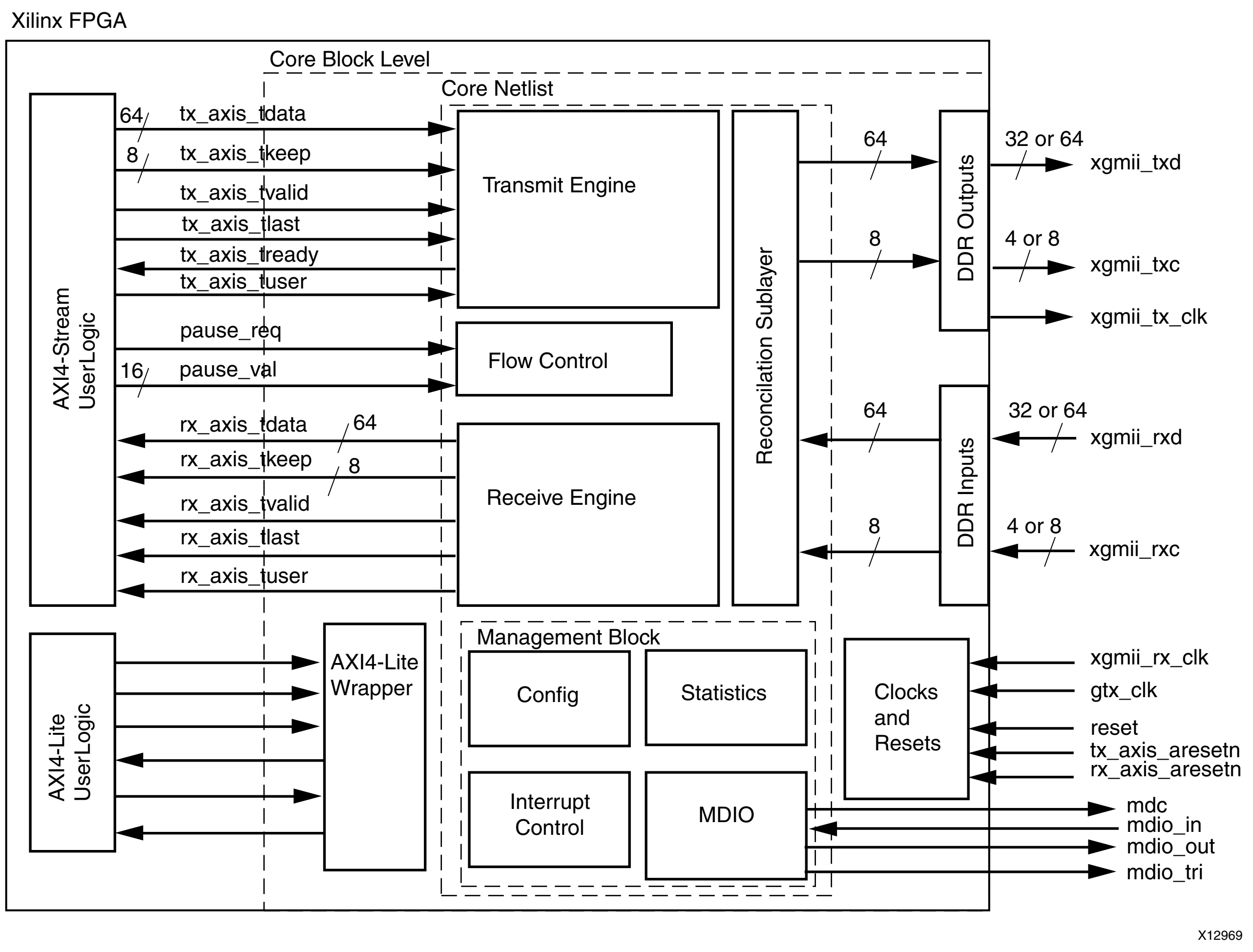


Figure 1-1: **Implementation of the 10-Gigabit Ethernet MAC Core**

Although the 10-Gigabit Ethernet MAC core is a fully verified solution, the challenge associated with implementing a complete design varies depending on the configuration and functionality of the application.

RECOMMENDED: *For best results, previous experience building high performance, pipelined FPGA designs using Xilinx implementation software and UCF files is recommended. Contact your local Xilinx representative for a closer review and estimation for your specific requirements.*

Feature Summary

The 10-Gigabit Ethernet MAC core connects to the PHY layer through an external XGMII or internal FPGA interface (the internal interface is only available on Spartan-6 devices). The PHY layers are managed through an optional MDIO STA master interface. Configuration of the core is done through an AXI4-Lite Management interface. The AXI4-Stream Transmit and Receive interfaces allow for simple connection to user logic.

The Ethernet MAC core performs the Link function of the 10 Gb Ethernet standard. The core supports flow control in both transmit and receive directions. The Transmit side of the core supports interframe gap (IFG), obtained through the Deficit Idle Count, to maintain the effective data rate of 10 Gb/s as described in *IEEE Standard 802.3-2008*.

The optional statistics counters collect statistics on the success and failure of various operations. These are accessed through the AXI4-Lite Management interface.

Applications

Figure 1-2 shows a typical Ethernet system architecture and the 10-Gigabit Ethernet MAC core within it. The Ethernet MAC and all the blocks to the right are defined in Ethernet IEEE specifications.

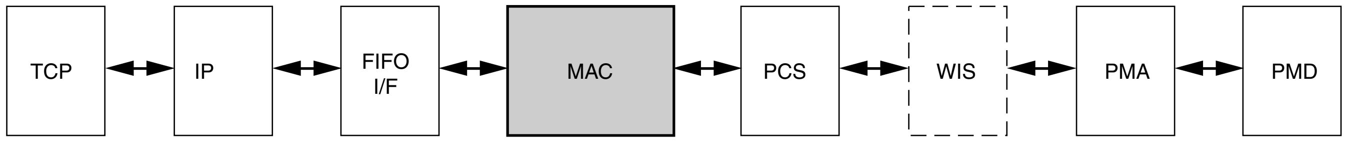


Figure 1-2: **Typical Ethernet System Architecture**

Figure 1-3 shows the 10-Gigabit Ethernet MAC core connected to a physical layer (PHY) device, for example, an optical module using the XGMII interface.

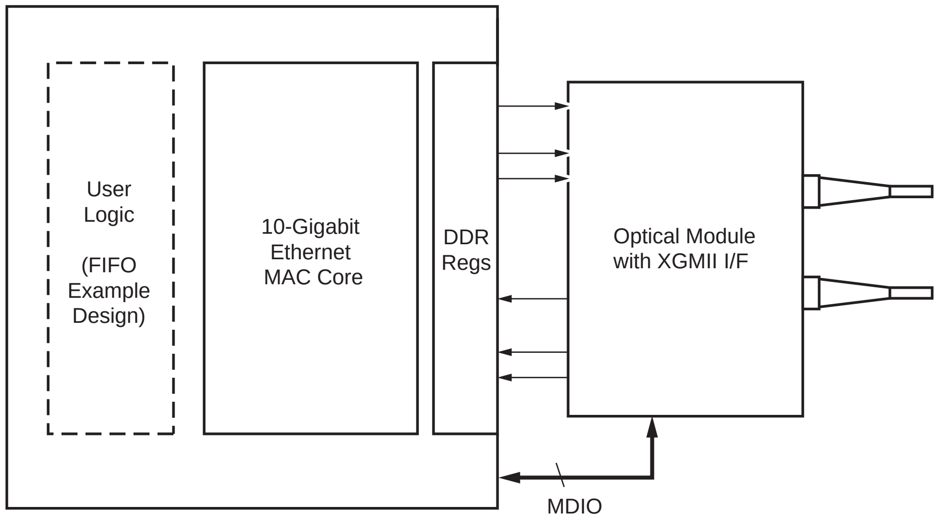


Figure 1-3: **10-Gigabit Ethernet MAC Core Connected to PHY with XGMII Interface**

The 10-Gigabit Ethernet MAC core is designed to be attached to the Xilinx IP XAUI core, the Xilinx IP RXAUI core, and the Xilinx IP 10G Ethernet PCS/PMA. Figure 1-4 illustrates the 10-Gigabit Ethernet MAC and XAUI cores in a system using an XPAK optical module.

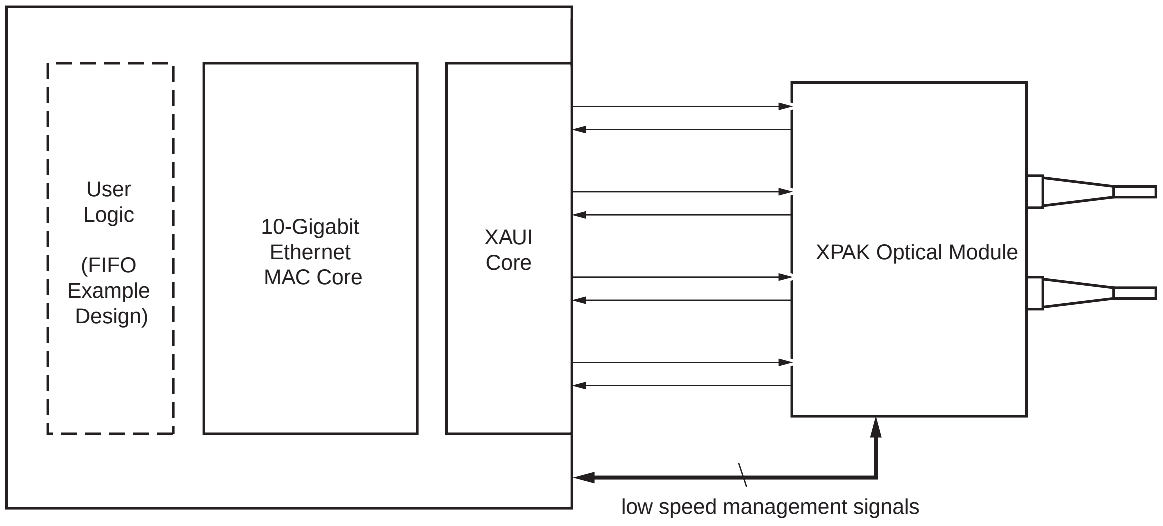


Figure 1-4: **10-Gigabit Ethernet MAC Core Used with Xilinx XAUI Core**

See Interfacing to the Xilinx XAUI Core, page 75 for details on using the two cores together in a system.

The 10-Gigabit Ethernet MAC core can also be attached to the Xilinx RXAUI core and the Xilinx 10-Gigabit Ethernet PCS/PMA core. See Interfacing with the RXAUI Core, page 78 and Interfacing to the 10-Gigabit Ethernet PCS/PMA Core, page 81 for details.

Licensing and Ordering Information

This Xilinx LogiCORE IP module is provided under the terms of the Xilinx Core License Agreement. The module is shipped as part of the ISE Design Suite tools. 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 MAC product 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

Figure 2-1 shows a block diagram of the implementation of the LogiCORE™ IP 10-Gigabit Ethernet MAC core. The major functional blocks of the core are:

- AXI4-Stream Interface – Designed for simple attachment of user logic
- Transmitter
- Receiver
- Flow Control block – Implements both Receive Flow Control and Transmit Flow Control
- Reconciliation Sublayer (RS) – Processes XGMII Local Fault and Remote Fault messages and handles DDR conversion
- AXI4-Lite Management interface and MDIO (optional)
- Statistics counters (optional)
- XGMII interface – Connection to the physical layer device or logic

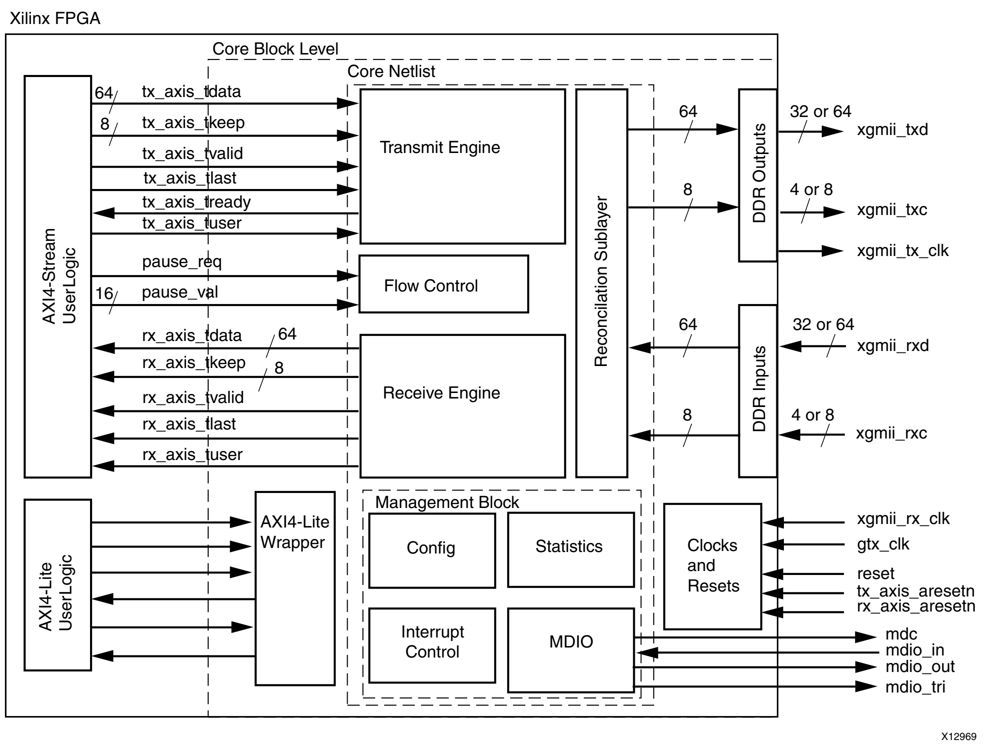


Figure 2-1: **Implementation of the 10-Gigabit Ethernet MAC Core**

Some customer applications do not require an external XGMII interface but instead need a connection to user logic. This application architecture is shown in Figure 2-2.

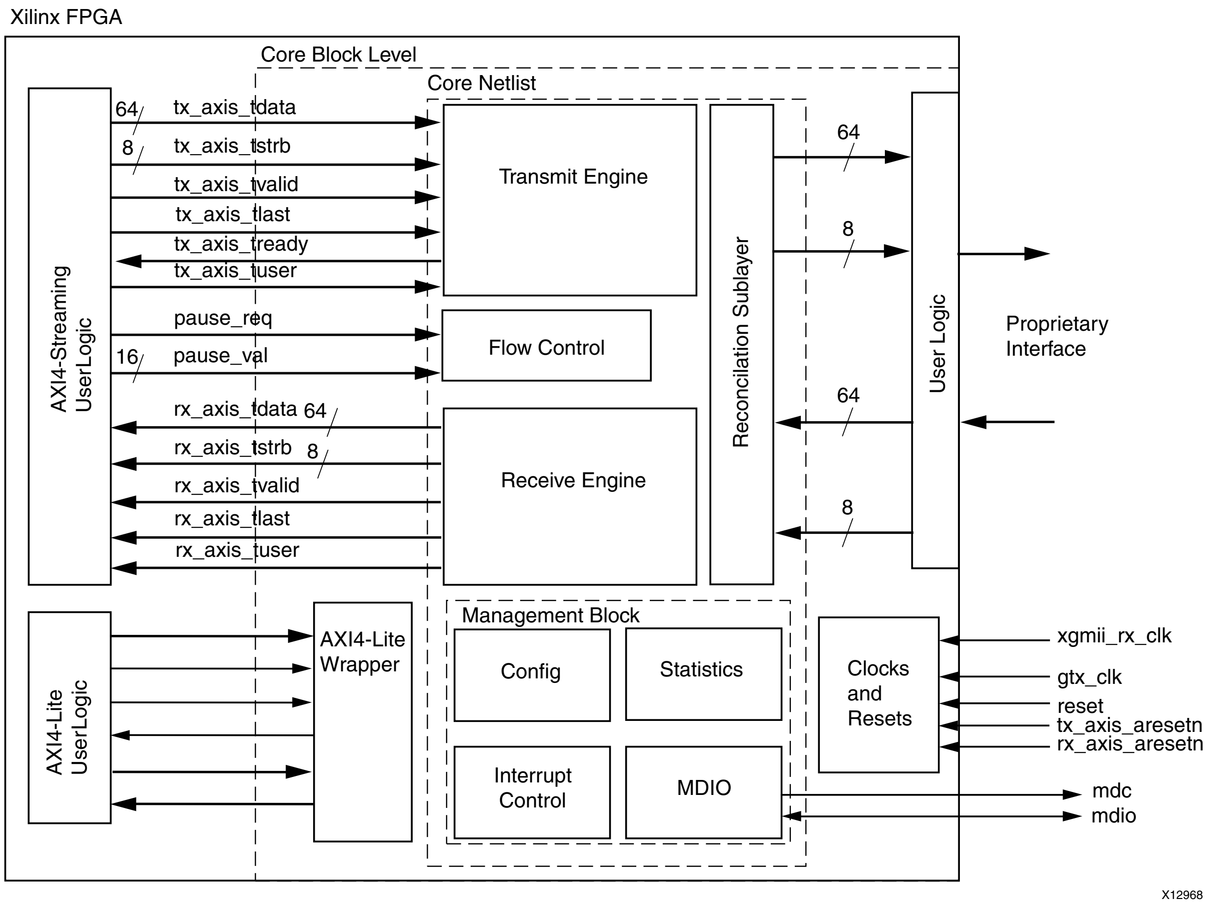


Figure 2-2: **Implementation of the Core with User Logic on PHY Interface**

Standards

The LogiCORE IP 10-Gigabit Ethernet MAC core is designed to the *IEEE Standard 802.3-2008,* 10-Gigabit Ethernet specification.

Performance

This section details the performance information for various core configurations.

Latency

These measurements are for the core only; they do not include the latency through the example design FIFO or IOB registers.

Transmit Path Latency

As measured from the input port `tx_axis_tdata` of the AXI4-Stream Transmit interface (until that data appears on `xgmii_txd` on the PHY-side interface), the latency through the core in the transmit direction is 15 clock periods of the `tx_clk0`.

Receive Path Latency

Measured from the `xgmii_rxd` port on the PHY-side Receive interface (until the data appears on the `rx_axis_tdata` port of the receiver side AXI4-Stream interface), the latency through the core in the receive direction is 14 clock periods of `rx_clk0`. This can increase to 15 clock periods if the core needs to modify the alignment of data at the AXI4-Stream Receive interface.

Resource Utilization

7 Series FPGAs

Table 2-1 provides approximate utilization figures for various core options when a single instance of the core is instantiated in a Virtex-7 device.

Utilization figures are obtained by implementing the block-level wrapper for the core. This wrapper is part of the example design and connects the core to the selected physical interface.

Note: Resources numbers for Zynq™-7000 devices are expected to be similar to 7 series device numbers.

Table 2-1: **Device Utilization for the 10-Gigabit Ethernet MAC Core (7 Series FPGAs)**

Parameter Values				Resource Usage			
Device Family	**Physical Interface**	**Management Interface**	**Statistic Counters**	**Slices**	**LUTs**	**FFs**	**BUFGs**
virtex7	XGMII	TRUE	TRUE	2179	3792	4101	2
			FALSE	1714	2868	3184	2
		FALSE	FALSE	1458	2656	2849	2
	Internal	TRUE	TRUE	2155	3867	4245	2
			FALSE	1707	2915	3327	2
		FALSE	FALSE	1609	2662	2994	2

Virtex-6 FPGAs

Table 2-2 provides approximate utilization figures for various core options when a single instance of the core is instantiated in a Virtex-6 device.

Utilization figures are obtained by implementing the block-level wrapper for the core. This wrapper is part of the example design and connects the core to the selected physical interface.

Table 2-2: **Device Utilization for the 10-Gigabit Ethernet MAC Core (Virtex-6 FPGAs)**

Parameter Values				Resource Usage			
Device Family	**Physical Interface**	**Management Interface**	**Statistic Counters**	**Slices**	**LUTs**	**FFs**	**BUFGs**
virtex6	XGMII	TRUE	TRUE	2127	3887	3957	2
			FALSE	1661	2913	3040	2
		FALSE	FALSE	1357	2748	2705	2
	Internal	TRUE	TRUE	2101	3921	3957	2
			FALSE	1609	2846	2969	2
		FALSE	FALSE	1439	2694	2706	2

Spartan-6 FPGAs

Table 2-3 provides approximate utilization figures for various core options when a single instance of the core is instantiated in a Spartan-6 device.

Utilization figures are obtained by implementing the block-level wrapper for the core. This wrapper is part of the example design and connects the core to the selected physical interface.

Table 2-3: **Device Utilization for the 10-Gigabit Ethernet MAC Core (Spartan-6 FPGAs)**

Parameter Values				Resource Usage			
Device Family	**Physical Interface**	**Management Interface**	**Statistic Counters**	**Slices**	**LUTs**	**FFs**	**BUFGs**
spartan6	Internal	TRUE	TRUE	2003	3810	4031	1
			FALSE	1486	2951	3114	1
		FALSE	FALSE	1352	2702	2780	1

Port Descriptions

The descriptions are located in these sections:

- AXI4-Stream Interface – Transmit
- AXI4-Stream Interface – Receive
- Flow Control Interface
- 32-Bit XGMII PHY Interface or 64-Bit SDR PHY Interface
- Management Interface Ports
- Configuration and Status Signals
- MDIO Interface Signals
- Interrupt Signal
- Statistic Vector Signals
- Clocking and Reset Signals

AXI4-Stream Interface – Transmit

The signals of the transmit AXI4-Stream interface are shown in Table 2-4. See Interfacing to the Data Interfaces, page 37 for details on connecting to the transmit interface.

Table 2-4: **AXI4-Stream Interface Ports – Transmit**

Name	Direction	Description
tx_axis_aresetn	In	AXI4-Stream active-Low reset for Transmit path XGMAC
tx_axis_tdata[63:0]	In	AXI4-Stream Data to XGMAC
tx_axis_tkeep[7:0]	In	AXI4-Stream Data Control to XGMAC
tx_axis_tvalid	In	AXI4-Stream Data Valid input to XGMAC
tx_axis_tuser	In	AXI4-Stream User signal used to signal explicit underrun
tx_ifg_delay[7:0]	In	Configures Interframe Gap adjustment between packets.

Table 2-4: **AXI4-Stream Interface Ports – Transmit *(Cont'd)***

Name	Direction	Description
tx_axis_tlast	In	AXI4-Stream signal to XGMAC indicating End of Ethernet Packet
tx_axis_tready	Out	AXI4-Stream acknowledge signal from XGMAC to indicate the start of a Data transfer.

AXI4-Stream Interface – Receive

The signals of the AXI4-Stream interface are shown in Table 2-5. See Interfacing to the Data Interfaces for details on connecting to the receive interface.

Table 2-5: **AXI4-Stream Interface Ports – Receive**

Name	Direction	Description
rx_axis_aresetn	In	AXI4-Stream active-Low reset for Receive path XGMAC
rx_axis_tdata	Out	AXI4-Stream data from XGMAC to upper layer
rx_axis_tkeep	Out	AXI4-Stream data control from XGMAC to upper layer
rx_axis_tvalid	Out	AXI4-Stream Data Valid from XGMAC
rx_axis_tuser	Out	AXI4-Stream User signal from XGMAC 1 indicates that a good packet has been received. 0 indicates that a bad packet has been received.
rx_axis_tlast	Out	AXI4-Stream signal from XGMAC indicating the end of a packet

Flow Control Interface

The flow control interface is used to initiate the transmission of flow control frames from the core. The ports associated with this interface are shown in Table 2-6.

Table 2-6: **Flow Control Interface Ports**

Name	Direction	Description
pause_req	In	Request that a flow control frame is emitted from the Ethernet MAC core.
pause_val[15:0]	In	Pause value field for flow control frame to be sent when pause_req asserted.

32-Bit XGMII PHY Interface or 64-Bit SDR PHY Interface

This interface is used to connect to the physical layer, whether this is a separate device or implemented in the FPGA beside the Ethernet MAC core. Table 2-7 shows the ports associated with this interface. The PHY interface can be a 32-bit DDR XGMII interface a or 64-bit SDR interface, depending on the customization of the core. However, the netlist ports are always the same width and the translation between 32-bit and 64-bit is performed in the HDL wrapper, if required.

Table 2-7: **PHY Interface Port Descriptions**

Name	Direction	Description
xgmii_txd[63:0]	Out	Transmit data to PHY
xgmii_txc[7:0]	Out	Transmit control to PHY
xgmii_rxd[63:0]	In	Received data from PHY
xgmii_rxc[7:0]	In	Received control from PHY

Management Interface Ports

Configuration of the core, access to the statistics block, access to the MDIO port, and access to the interrupt block can be provided through the Management Interface, a 32-bit AXI4-Lite interface independent of the Ethernet datapath. Table 2-8 defines the ports associated with the Management Interface.

Table 2-8: **Management Interface Port Descriptions**

Name	Direction	Description
s_axi_aclk	In	AXI4-Lite clock. Range between 10 MHz and 156.25 MHz
s_axi_aresetn	In	Asynchronous active-Low reset
s_axi_awaddr[31:0]	In	Write address Bus
s_axi_awvalid	In	Write address valid
s_axi_awready	Out	Write address acknowledge
s_axi_wdata[31:0]	In	Write data bus
s_axi_wvalid	Out	Write data valid
s_axi_wready	Out	Write data acknowledge
s_axi_bresp[1:0]	Out	Write transaction response
s_axi_bvalid	Out	Write response valid
s_axi_bready	In	Write response acknowledge
s_axi_araddr[31:0]	In	Read address bus
s_axi_arvalid	In	Read address valid
s_axi_arready	Out	Read address acknowledge
s_axi_rdata[31:0]	Out	Read data output
s_axi_rresp[1:0]	Out	Read data response
s_axi_rvalid	Out	Read data/response valid
s_axi_rready	In	Read data acknowledge

The Management Interface can be omitted at core customization stage; if omitted, `configuration_vector_tx/rx` is available instead.

Configuration and Status Signals

If the Management Interface is omitted at core customization time, configuration and status vectors are exposed by the core. This allows you to configure the core by statically or dynamically driving the constituent bits of the port. Table 2-9 describes the configuration and Status signals. See Interfacing to the Management Interface, page 54 for details on this signal, including a breakdown of the configuration and status vector bits.

Table 2-9: **Configuration and Status Signals**

Name	Direction	Description
tx_configuration_vector[79:0]	Input	Configuration signals for the Transmitter
rx_configuration_vector[79:0]	Input	Configuration signals for the Receiver
status_vector[1:0]	Output	Status signals for the core

MDIO Interface Signals

The MDIO Interface signals are shown in Table 2-10. See Interfacing to the Management Interface for details on the use of this interface.

Table 2-10: **MDIO Interface Port Descriptions**

Name	Direction	Description
mdc	Output	MDIO clock
mdio_in	Input	MDIO input
mdio_out	Output	MDIO output
mdio_tri	Output	MDIO 3-state. A 1 disconnects the output driver from the MDIO bus.

Interrupt Signal

The Interrupt output signal is shown in Table 2-11. See Interrupt Output, page 61 for more details.

Table 2-11: **Interrupt Output Port Description**

Name	Direction	Description
xgmacint	Output	Interrupt output.

Statistic Vector Signals

In addition to the statistic counters described in Statistics Counters and Register Space, page 18, there are two statistics vector outputs on the core netlist that are used to signal the core state. These vectors are used as the inputs of the counter logic internal to the core; so if you omit the statistic counters at the CORE Generator™ tool customization stage, a relevant subset can be implemented in user logic. The signals are shown in Table 2-12. The contents of the vectors themselves are described in Interfacing to the Management

Interface.

Table 2-12: **Statistic Vector Signals**

Name	Direction	Description
tx_statistics_vector[25:0]	Output	Aggregated statistics flags for transmitted frame.
tx_statistics_valid	Output	Valid strobe for tx_statistics_vector.
rx_statistics_vector[29:0]	Output	Aggregated statistics flags for received frames.
rx_statistics_valid	Output	Valid strobe for rx_statistics_vector.

Clocking and Reset Signals

Included in the example design top-level sources are circuits for clock and reset management. These can include Digital Clock Managers (DCMs) or Mixed-Mode Clock Managers (MMCMs), reset synchronizers, or other useful utility circuits that can be useful in your particular application.

Table 2-13 shows the ports on the netlist associated with system clocks and resets.

Table 2-13: **Clock, Clock Management, and Reset Ports**

Name	Direction	Description
tx_clk0	Input	System clock for transmit side of core; derived from gtx_clk in example design
tx_dcm_lock	Input	Status flag from DCM/MMCM
rx_clk0	Input	System clock for receive side of core; derived from xgmii_rx_clk in example design
rx_dcm_lock	Input	Status flag from DCM/MMCM

Statistics Counters and Register Space

Statistics Counters

During operation, the Ethernet MAC core collects statistics on the success and failure of various operations for processing by network management entities elsewhere in the system. These statistics are accessed through the Management Interface. A list of statistics is shown in Table 2-14.

As per *IEEE Standard 802.3-2008* [Ref 1], sub-clause 5.2.1, these statistic counters are wraparound counters and do not have a reset function. They do not reset upon being read and only return to zero when they naturally wrap around or when the device is reconfigured.

All statistics counters are read only, Write attempts to Statistics Counters are acknowledged with a SLVERR on the AXI4-Lite bus.

Read of MSW of a particular counter is allowed only if the previous transaction was addressed to the LSW of the same counter, otherwise the MSW read operation is acknowledged with a SLVERR on the AXI4-Lite Bus. This restriction is to avoid the rollover of LSW counter into MSW counter between the read transactions.

Table 2-14: **Statistics Counters**

Address (Hex)	Name	Description
0x200	Received bytes - LSW	A count of bytes of frames that are received (destination address to frame check sequence inclusive).
0x204	Received bytes - MSW	
0x208	Transmitted bytes - LSW	A count of bytes of frames that are transmitted (destination address to frame check sequence inclusive).
0x20C	Transmitted bytes - MSW	
0x210	Undersize frames received - LSW	A count of the number of frames that were less than 64 bytes in length but were otherwise well formed.
0x214	Undersize frames received - MSW	
0x218	Fragment frames received – LSW	A count of the number of packets received that were less than 64 bytes in length and had a bad frame check sequence field.
0x21C	Fragment frames received – MSW	
0x220	64 byte frames received OK – LSW	A count of error-free frames received that were 64 bytes in length.
0x224	64 byte frames received OK – MSW	
0x228	65-127 byte frames received OK – LSW	A count of error-free frames received that were between 65 and 127 bytes in length inclusive.
0x22C	65-127 byte frames received OK – MSW	
0x230	128-255 byte frames received OK – LSW	A count of error-free frames received that were between 128 and 255 bytes in length inclusive.
0x234	128-255 byte frames received OK – MSW	
0x238	256-511 byte frames received OK – LSW	A count of error-free frames received that were between 256 and 511 bytes in length inclusive.
0x23C	256-511 byte frames received OK – MSW	
0x240	512-1023 byte frames received OK – LSW	A count of error-free frames received that were between 512 and 1023 bytes in length inclusive.
0x244	512-1023 byte frames received OK – MSW	
0x248	1024-MaxFrameSize byte frames received OK – LSW	A count of error-free frames received that were between 1024 bytes and the maximum legal frame size as specified in *IEEE Standard 802.3-2008* [Ref 1].
0x24C	1024-MaxFrameSize byte frames received OK – MSW	
0x250	Oversize frames received OK – LSW	A count of otherwise error-free frames received that exceeded the maximum legal frame length specified in *IEEE Standard 802.3-2008*.
0x254	Oversize frames received OK – MSW	
0x258	64 byte frames transmitted OK – LSW	A count of error-free frames transmitted that were 64 bytes in length.
0x25C	64 byte frames transmitted OK – MSW	

Table 2-14: **Statistics Counters *(Cont'd)***

Address (Hex)	Name	Description
0x260	65-127 byte frames transmitted OK – LSW	A count of error-free frames transmitted that were between 65 and 127 bytes in length.
0x264	65-127 byte frames transmitted OK – MSW	
0x268	128-255 byte frames transmitted OK – LSW	A count of error-free frames transmitted that were between 128 and 255 bytes in length.
0x26C	128-255 byte frames transmitted OK – MSW	
0x270	256-511 byte frames transmitted OK – LSW	A count of error-free frames transmitted that were between 256 and 511 bytes in length.
0x274	256-511 byte frames transmitted OK – MSW	
0x278	512-1023 byte frames transmitted OK – LSW	A count of error-free frames transmitted that were between 512 and 1023 bytes in length.
0x27C	512-1023 byte frames transmitted OK – MSW	
0x280	1024-MaxFrameSize byte frames transmitted OK – LSW	A count of error-free frames transmitted that were between 1024 bytes and the maximum legal frame length specified in *IEEE Standard 802.3-2008* [Ref 1].
0x284	1024-MaxFrameSize byte frames transmitted OK	
0x288	Oversize frames transmitted OK – LSW	A count of otherwise error-free frames transmitted that exceeded the maximum legal frame length specified in *IEEE Standard 802.3-2008*.
0x28C	Oversize frames transmitted OK – MSW	
0x290	Frames received OK – LSW	A count of error free frames received.
0x294	Frames received OK – MSW	
0x298	Frame Check Sequence errors – LSW	A count of received frames that failed the CRC check and were at least 64 bytes in length.
0x29C	Frame Check Sequence errors – MSW	
0x2A0	Broadcast frames received OK – LSW	A count of frames that were successfully received and were directed to the broadcast group address.
0x2A4	Broadcast frames received OK – MSW	
0x2A8	Multicast frames received OK – LSW	A count of frames that were successfully received and were directed to a non-broadcast group address.
0x2AC	Multicast frames received OK – MSW	
0x2B0	Control frames received OK – LSW	A count of error-free frames received that contained the MAC Control type identifier in the length/type field.
0x2B4	Control frames received OK – MSW	
0x2B8	Length/Type out of range – LSW	A count of error-free frames received that were at least 64 bytes in length where the length/type field contained a length value that did not match the number of MAC client data bytes received. The counter also increments for frames in which the length/type field indicated that the frame contained padding but where the number of MAC client data bytes received was greater than 64 bytes (minimum frame size).
0x2BC	Length/Type out of range – MSW	
0x2C0	VLAN tagged frames received OK – LSW	A count of error-free frames received with VLAN tags. This counter only increments when the receiver has VLAN operation enabled.
0x2C4	VLAN tagged frames received OK – MSW	

Table 2-14: **Statistics Counters *(Cont'd)***

Address (Hex)	Name	Description
0x2C8	PAUSE frames received OK – LSW	A count of error-free frames received that contained the MAC Control type identifier 88-08 in the length/type field, contained a destination address that matched either the MAC Control multicast address or the configured source address of the Ethernet MAC, contained the Pause opcode and were acted on by the Ethernet MAC.
0x2CC	PAUSE frames received OK – MSW	
0x2D0	Control frames received with unsupported opcode – LSW	A count of error-free frames received that contained the MAC Control type identifier 88-08 in the length/type field but were received with an opcode other than the Pause opcode.
0x2D4	Control frames received with unsupported opcode – MSW	
0x2D8	Frames transmitted OK – LSW	A count of error-free frames transmitted.
0x2DC	Frames transmitted OK – MSW	
0x2E0	Broadcast frames transmitted OK – LSW	A count of error-free frames transmitted to the broadcast address.
0x2E4	Broadcast frames transmitted OK – MSW	
0x2E8	Multicast frames transmitted OK – LSW	A count of error-free frames transmitted to group addresses other than the broadcast address.
0x2EC	Multicast frames transmitted OK – MSW	
0x2F0	Underrun errors – LSW	A count of frames that would otherwise be transmitted by the core but could not be completed due to the assertion of underrun during the frame transmission. This does not count frames which are less than 64 bytes in length.
0x2F4	Underrun errors – MSW	
0x2F8	Control frames transmitted OK – LSW	A count of error-free frames transmitted that contained the MAC Control Frame type identifier 88-08 in the length/type field.
0x2FC	Control frames transmitted OK – MSW	
0x300	VLAN tagged frames transmitted OK – LSW	A count of error-free frames transmitted that contained a VLAN tag. This counter only increments when the transmitter has VLAN operation enabled.
0x304	VLAN tagged frames transmitted OK – MSW	
0x308	PAUSE frames transmitted OK – LSW	A count of error-free pause frames generated and transmitted by the core in response to an assertion of pause_req.
0x30C	PAUSE frames transmitted OK – MSW	

Configuration Registers

After the core is powered up and reset, the client can reconfigure some of the core parameters from their defaults, such as flow control support and WAN/LAN connections. Configuration changes can be written at any time. Both the receiver and transmitter configuration register changes only take effect during interframe gaps. The exceptions to this are the configurable soft resets, which take effect immediately. Configuration of the Ethernet MAC core is performed through a register bank accessed through the Management Interface. The configuration registers available in the core are detailed in

Table 2-15.

Table 2-15: **Configuration Registers**

Address (Hex)	Description
0x400	Receiver Configuration Word 0
0x404	Receiver Configuration Word 1
0x408	Transmitter Configuration
0x40C	Flow Control Configuration
0x410	Reconciliation Sublayer Configuration
0x414	Receiver MTU Configuration Word
0x418	Transmitter MTU Configuration Word
0x4F8	Version Register (Read Only)
0x4FC	Capability Register (Read Only)

The contents of each configuration register are shown in Tables 2-16 through Table 2-20.

Table 2-16: **Receiver Configuration Word 0**

Bits	Default Value	Description
31:0	All 0s	**Pause frame MAC address [31:0]** This address is used by the Ethernet MAC to match against the destination address of any incoming flow control frames. It is also used by the flow control block as the source address (SA) for any outbound flow control frames. This address does not have any affect on frames passing through the main transmit and receive datapaths of the Ethernet MAC. The address is ordered so the first byte transmitted or received is the lowest positioned byte in the register; for example, a MAC address of AA-BB-CC-DD-EE-FF would be stored in Address[47:0] as 0xFFEEDDCCBBAA.

Table 2-17: **Receiver Configuration Word 1**

Bits	Default Value	Description
31	0	**Receiver reset**. When this bit is set to 1, the receiver is reset. The bit then automatically reverts to 0. This reset also sets all of the receiver configuration registers to their default values.
30	0	**Jumbo Frame Enable**. When this bit is set to 1, the Ethernet MAC receiver accepts frames that are greater than the maximum legal frame length specified in *IEEE Standard 802.3-2008* [Ref 1]. When this bit is 0, the Ethernet MAC only accepts frames up to the legal maximum.
29	0	**In-band FCS Enable**. When this bit is 1, the Ethernet MAC receiver passes the FCS field up to the client as described in Reception with In-Band FCS Passing, page 49. When it is 0, the client is not passed to the FCS. In both cases, the FCS is verified on the frame.
28	1	**Receiver Enable**. If set to 1, the receiver block is operational. If set to 0, the block ignores activity on the physical interface RX port.
27	0	**VLAN Enable.** When this bit is set to 1, VLAN tagged frames are accepted by the receiver.

Table 2-17: **Receiver Configuration Word 1 *(Cont'd)***

Bits	Default Value	Description
26	0	**Receiver Preserve Preamble Enable**. When this bit is set to 1, the Ethernet MAC receiver preserves the preamble field of the received frame. When it is 0, the preamble field is discarded as specified in *IEEE Standard 802.3-2008* [Ref 1].
25	0	**Length/Type Error Check Disable.** When this bit is set to 1, the core does not perform the length/type field error checks as described in Length/Type Field Error Checks, page 52. When this bit is set to 0, the length/type field checks are performed; this is normal operation.
24	0	**Control Frame Length Check Disable.** When this bit is set to 1, the core does not mark MAC Control frames as "bad" if they are greater than minimum frame length.
23:16	N/A	Reserved
15:0	All 0s	**Pause frame MAC address [47:32]**. See description in Table 2-16.

Table 2-18: **Transmitter Configuration Word**

Bits	Default Value	Description
31	0	**Transmitter Reset**. When this bit is set to 1, the transmitter is reset. The bit then automatically reverts to 0. This reset also sets all of the transmitter configuration registers to their default values.
30	0	**Jumbo Frame Enable**. When this bit is set to 1, the Ethernet MAC transmitter sends frames that are greater than the maximum legal frame length specified in *IEEE Standard 802.3-2008* [Ref 1]. When this bit is 0, the Ethernet MAC only sends frames up to the legal maximum.
29	0	**In-band FCS Enable.** When this bit is 1, the Ethernet MAC transmitter expects the FCS field to be passed in by the client as described in Transmission with In-Band FCS Passing, page 39. When this bit is 0, the Ethernet MAC transmitter appends padding as required, computes the value for the FCS field and appends it to the frame.
28	1	**Transmitter Enable.** When this bit is 1, the transmitter is operational. When it is 0, the transmitter is disabled.
27	0	**VLAN Enable.** When this bit is set to 1, the transmitter allows the transmission of VLAN tagged frames.
26	0	**WAN Mode Enable.** When this bit is set to 1, the transmitter automatically inserts extra idles into the interframe gap (IFG) to reduce the average data rate to that of the OC-192 SONET payload rate (WAN mode). When this bit is set to 0, the transmitter uses normal Ethernet interframe gaps (LAN mode). When the transmitter is in WAN mode, jumbo frames should be limited to 16384 bytes maximum
25	0	**Interframe Gap Adjust Enable**. When this bit is set to 1, the core reads the value on the port tx_ifg_delay at the start of a frame transmission and adjust the interframe gap accordingly. See Interframe Gap Adjustment, page 44. When this bit is set to 0, the transmitter outputs the minimum Inter Frame Gap. This bit has no effect when Bit[26] (LAN/WAN mode) is set to 1.
24	0	**Deficit Idle Count Enable**. When this bit is set to 1, the core reduces the IFG as described in *IEE 803.2ae-2008* 46.3.1.4 Option 2 to support the maximum data transfer rate. When this bit is set to 0, the core always stretches the IFG to maintain start alignment. This bit is cleared and has no effect if Interframe Gap Adjust is enabled.

Table 2-18: **Transmitter Configuration Word *(Cont'd)***

Bits	Default Value	Description
23	0	**Transmitter Preserve Preamble Enable**. When this bit is set to 1, the Ethernet MAC transmitter preserves the custom preamble field presented on the Client Interface. When it is 0, the standard preamble field specified in *IEEE Standard 802.3-2008* [Ref 1] is transmitted.
22:0	N/A	Reserved

Table 2-19: **Flow Control Configuration Word**

Bits	Default Value	Description
31	N/A	Reserved
30	1	**Flow Control Enable (TX).** When this bit is 1, asserting the PAUSE_REQ signal sends a flow control frame out from the transmitter. When this bit is 0, asserting the PAUSE_REQ signal has no effect.
29	1	**Flow Control Enable (RX)**. When this bit is 1, received flow control frames inhibit the transmitter operation as described in Receiving a Pause Frame, page 53. When this bit is 0, received flow control frames are always passed up to the client.
28:0	N/A	Reserved

Table 2-20: **Reconciliation Sublayer Configuration Word**

Bits	Default Value	Description
31	N/A	**Receive DCM Locked**. If this bit is 1, the Digital Clock Management (DCM) block for the receive-side clocks (XGMII_RX_CLK, RX_CLK) is locked. If this bit is 0, the DCM is not locked. Read-only.
30	N/A	**Transmit DCM Locked**. If this bit is 1, the Digital Clock Management (DCM) block for the transmit-side clocks (GTX_CLK, XGMII_TX_CLK, TX_CLK) is locked. If this bit is 0, the DCM is not locked. Read-only.
29	N/A	**Remote Fault Received**. If this bit is 1, the RS layer is receiving remote fault sequence ordered sets. Read-only.
28	N/A	**Local Fault Received**. If this bit is 1, the RS layer is receiving local fault sequence ordered sets. Read-only.
27	0	**Fault Inhibit**. When this bit is set to 0, the Reconciliation Sublayer transmits ordered sets as laid out in *IEEE Standard 802.3-2008* [Ref 1]; that is, when the RS is receiving Local Fault ordered sets, it transmits Remote Fault ordered sets. When it is receiving Remote Fault ordered sets, it transmits idles code words. When this bit is set to 1, the reconciliation sublayer always transmits data presented to it by the Ethernet MAC, regardless of whether fault ordered sets are being received.
26:0	N/A	Reserved

Table 2-21: **Receiver MTU Configuration Word**

Bits	Default Value	Description
31:17	N/A	Reserved
16	0	**RX MTU Enable**. When this bit is set to 1, the value in RX MTU Size is used as the maximum frame size allowed as described in Receiver Maximum Permitted Frame Length. When set to 0 frame handling depends on the other configuration settings.
15	N/A	Reserved
14:0	0x05EE	**RX MTU Size**. This value is used as the maximum frame size allowed as described in Receiver Maximum Permitted Frame Length, page 52 when RX MTU Enable is set to 1. Only values of 1518 or greater are legal for RX MTU size and the core does not enforce this size on write. Ensure that only legal values are written to this register for correct core operation.

Table 2-22: **Transmitter MTU Configuration Word**

Bits	Default Value	Description
31:17	N/A	Reserved
16	0	**TX MTU Enable**. When this bit is set to 1, the value in TX MTU Size is used as the maximum frame size allowed as described in Transmitter Maximum Permitted Frame Length. When set to 0 frame handling depends on the other configuration settings.
15	N/A	Reserved
14:0	0x05EE	**TX MTU Size**. This value is used as the maximum frame size allowed as described in Transmitter Maximum Permitted Frame Length, page 43 when TX MTU Enable is set to 1. Only values of 1518 or greater are legal for TX MTU size and the core does not enforce this size on write. Ensure that only legal values are written to this register for correct core operation.

Table 2-23: **Version Register**

Bits	Default Value	Description
31:24	0x0B	**Major Revision**. This field indicates the major revision of the core.
23:16	0x04	**Minor Revision**. This field indicates the minor revision of the core.
15:8	N/A	Reserved
7:0	All 0s	**Patch Level**. This field indicates the patch status of the core. (When this value is 0x00 it indicates a non-patched version, when 0x01 indicates Rev 1, and so forth.)

Table 2-24: **Capability Register**

Bits	Default Value	Description
31:9	N/A	Reserved
8	1	**Statistics Counter**. This bit indicates that the core has statistics counters.
7:4	N/A	Reserved
3	1	**Line rate 10 Gbit**. This bit indicates that the core has a capability to support the 10 Gb line rate.
2	0	**Line rate 1 Gbit**. This bit indicates that the core has a capability to support the 1 Gb line rate.

Table 2-24: **Capability Register *(Cont'd)***

Bits	Default Value	Description
1	0	**Line rate 100 Mbit**. This bit indicates that the core has a capability to support the 100 Mb line rate.
0	0	**Line rate 10 Mbit**. This bit indicates that the core has a capability to support the 10 Mb line rate.

MDIO Registers

A list of MDIO registers is shown in Table 2-25.

Table 2-25: **MDIO Configuration Registers**

Address (Hex)	Description
0x500	MDIO Configuration Word 0
0x504	MDIO Configuration Word 1
0x508	MDIO TX Data
0x50C	MDIO RX Data (read-only)

The contents of each configuration register are shown in Table 2-26 through Table 2-29.

Table 2-26: **MDIO Configuration Word 0**

Bits	Default Value	Description
31:7	N/A	Reserved
6	0	**MDIO Enable**. When this bit is 1, the MDIO interface can be used to access attached PHY devices. When this bit is 0, the MDIO interface is disabled and the MDIO signal remains inactive.
5:0	All 0s	**Clock Divide**. Used as a divider value to generate the MDC signal at 2.5 MHz. See MDIO Interface, page 56.

Table 2-27: **MDIO Configuration Word 1**

Bits	Default Value	Description
31:29	N/A	Reserved
28:24	All 0s	**PRTAD**. Port address for the MDIO transaction
23:21	N/A	Reserved
20:16	All 0s	**DEVAD**. Device address for the MDIO transaction
15:14	0	**TX OP**. Opcode for the MDIO transaction. For more details, see the MDIO transactions Figure 3-28 through Figure 3-31.
13:12	N/A	Reserved
10:8	N/A	Reserved
11	0	**Initiate**. If a 1 is written to this bit when MDIO Ready is 1, an MDIO transaction is initiated. This bit goes to 0 automatically when the pending transaction completed.

Table 2-27: **MDIO Configuration Word 1 *(Cont'd)***

Bits	Default Value	Description
7	1	**MDIO Ready**. When this bit is 1, the MDIO master is ready for an MDIO transaction. When this bit is 0, MDIO master is busy in a transaction and goes to 1 when the pending transaction is complete. This bit is read-only.
6:0	N/A	Reserved

Table 2-28: **MDIO TX Data**

Bits	Default Value	Description
31:16	N/A	Reserved
15:0	All 0s	**MDIO TX Data**. MDIO Write data. Can be the address of the device based on the opcode.

Table 2-29: **MDIO RX Data**

Bits	Default Value	Description
31:16	N/A	Reserved
15:0	All 0s	**MDIO RX Data**. MDIO Read data.

Designing with the Core

This chapter includes guidelines and additional information to make designing with the 10-Gigabit Ethernet MAC core easier. It contains these sections:

General Design Guidelines

This section describes the steps required to turn a 10-Gigabit Ethernet MAC core into a fully functioning design with user application logic. Not all implementations require all of the design steps listed in this section. Follow the logic design guidelines in this document carefully.

Use the Example Design as a Starting Point

Every instance of the 10-Gigabit Ethernet MAC core created by Xilinx 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.

For information on using and customizing the example designs for the 10-Gigabit Ethernet MAC core, consult Chapter 6, Quick Start Example Design and Chapter 7, Detailed Example Design.

Know the Degree of Difficulty

10-Gigabit Ethernet designs are challenging to implement in any technology. The degree of difficulty is sharply influenced by:

- Maximum system clock frequency
- Targeted device architecture
- Nature of the user application

All 10-Gigabit Ethernet 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 the user application and the core. This means that all inputs and outputs from the user application should come from, or connect to, a flip-flop. While registering signals might not be possible for all paths, it simplifies timing analysis and makes it easier for the Xilinx tools to place and route the design.

Recognize Timing Critical Signals

The UCF constraints file provided with the example design for the core identifies the critical signals and the timing constraints that should be applied. For further information, see Chapter 5, Constraining the Core.

Make Only Allowed Modifications

The 10-Gigabit Ethernet MAC 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 10-Gigabit Ethernet MAC core can only be made by the selecting the options from within the CORE Generator tool when the core is generated. For more information, see Chapter 4, Customizing and Generating the Core.

Clocking

Figure 3-1 shows the clock arrangement for the Internal interface option of the 10-Gigabit Ethernet MAC. Clock logic that can be shared across multiple cores (such as the transmit clock management resources) is in the top level of the example design, and logic that must be replicated per core is in the block level of the example design. See Multiple Core Instances, page 73.

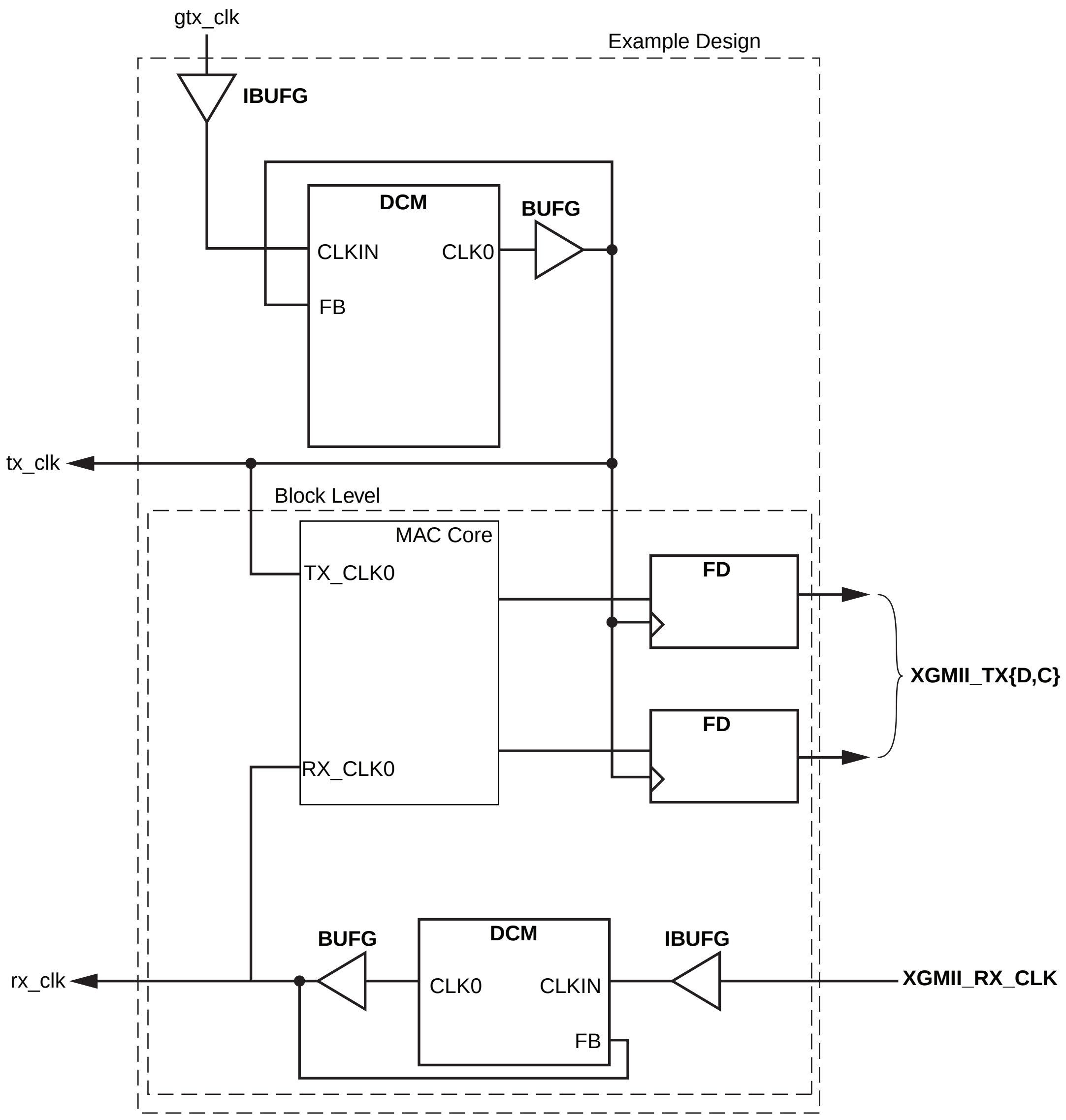


Figure 3-1: **Clocking Logic for the Ethernet MAC Internal Interface Option**

Resets

Internally, the core is divided up into clock/reset domains, which group together elements with the common clock and reset signals. The reset circuitry for one of these domains is illustrated in Figure 3-2.

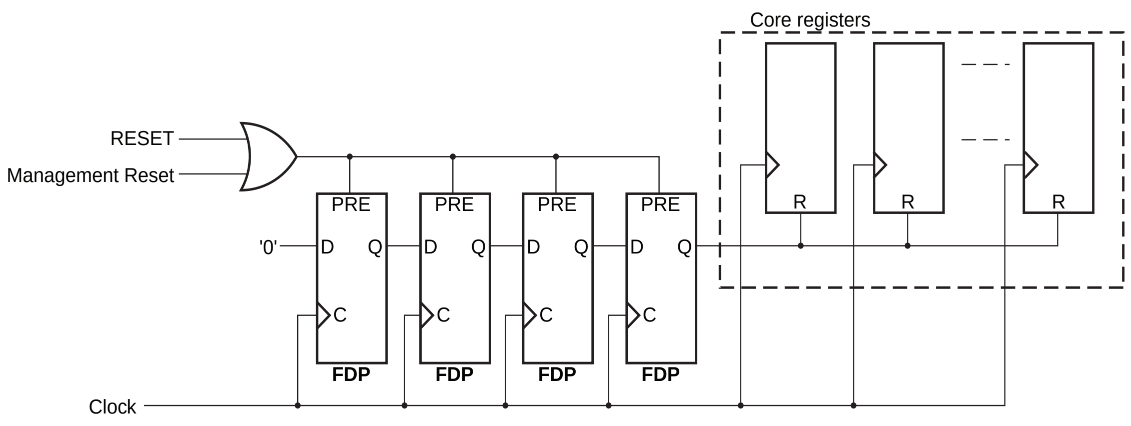


Figure 3-2: **Reset Circuit for a Single Clock/Reset Domain**

Protocol Description

Ethernet Protocol Overview

This section gives an overview of where the Ethernet MAC fits into an Ethernet system and provides a description of some basic Ethernet terminology.

Ethernet Sublayer Architecture

Figure 3-3 illustrates the relationship between the Open Systems Interconnection (OSI) reference model and the Ethernet MAC. The grayed-in layers show the functionality that the Ethernet MAC handles. Figure 3-3 also shows where the supported physical interfaces fit into the architecture.

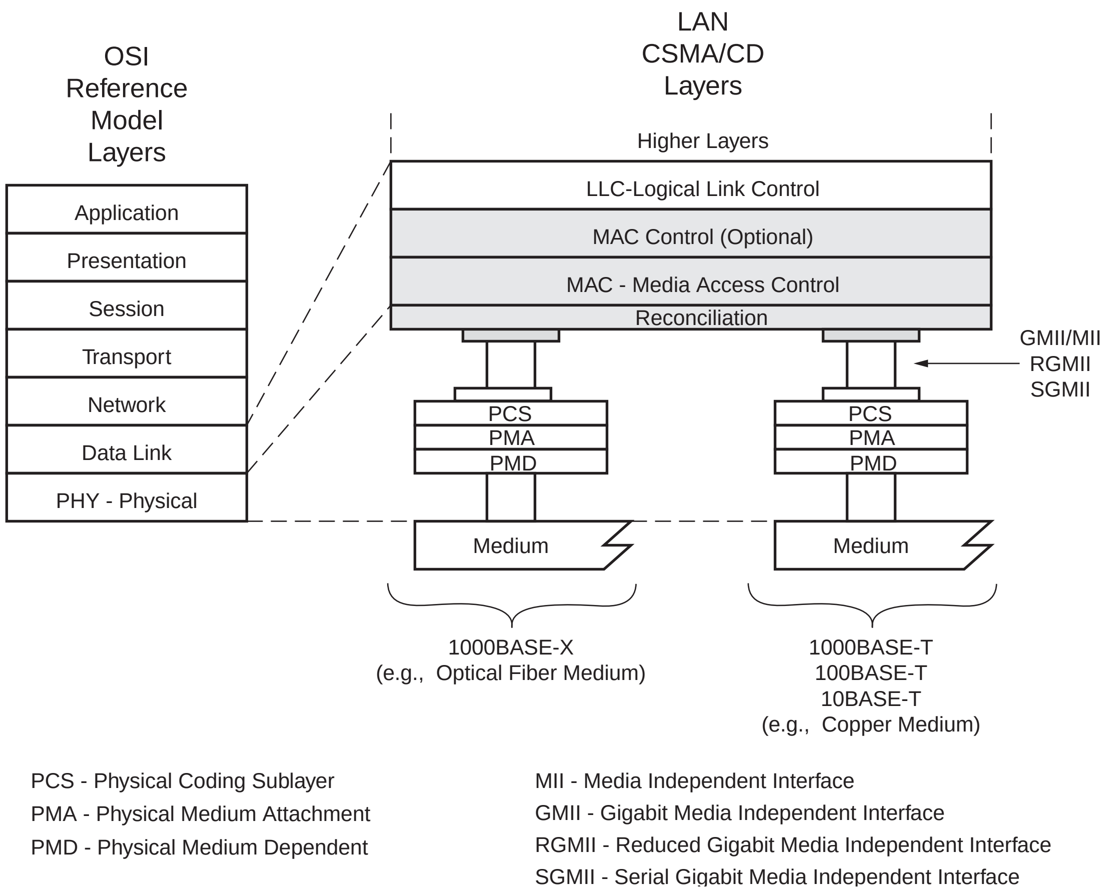

OSI Reference Model Layers
Application
Presentation
Session
Transport
Network
Data Link
PHY - Physical
LAN CSMA/CD Layers
Higher Layers
LLC-Logical Link Control
MAC Control (Optional)
MAC - Media Access Control
Reconciliation
GMII/MII
RGMII
SGMII
PCS
PMA
PMD
Medium
PCS
PMA
PMD
Medium
1000BASE-X (e.g., Optical Fiber Medium)
1000BASE-T
100BASE-T
10BASE-T
(e.g., Copper Medium)
PCS - Physical Coding Sublayer
PMA - Physical Medium Attachment
PMD - Physical Medium Dependent
MII - Media Independent Interface
GMII - Gigabit Media Independent Interface
RGMII - Reduced Gigabit Media Independent Interface
SGMII - Serial Gigabit Media Independent Interface

Figure 3-3: **IEEE Std 802.3-2008 Ethernet Model**

MAC and MAC CONTROL Sublayer

The Ethernet MAC is defined in *IEEE Std 802.3-2008*, clauses 2, 3, and 4. A MAC is responsible for the Ethernet framing protocols described in Ethernet Data Format and error detection of these frames. The MAC is independent of and can connect to any type of physical layer device.

The MAC Control sublayer is defined in *IEEE Std 802.3-2008*, clause 31. This provides real-time flow control manipulation of the MAC sublayer.

Both the MAC CONTROL and MAC sublayers are provided by the Ethernet MAC in all modes of operation.

Physical Sublayers PCS, PMA, and PMD

The combination of the Physical Coding Sublayer (PCS), the Physical Medium Attachment (PMA), and the Physical Medium Dependent (PMD) sublayer constitute the physical layers for the protocol. Several physical standards are specified including:

- **10GBASE-R/KR** – PHYs provide a link between the MAC and single optical and backplane channels at 10.3125 Gb/s. This is provided by the Ethernet 10-Gigabit Ethernet PCS/PMA core.
- **10GBASE-X/XAUI** – PHYS provide a link between the MAC and 4-lane backplane and chip-to-chip channels at 3.125 Gb/s per lane. This is provided by the Ethernet XAUI core.
- **RXAUI** – PHYs provide a link between the MAC and 2-lane backplane and chip-to-chip channels at 6.25 Gb/s per lane. This is provided by the Ethernet RXAUI core.

Ethernet Data Format

Ethernet data is encapsulated in frames, as shown in Figure 3-4, for standard Ethernet frames. The fields in the frame are transmitted from left to right. The bytes within the fields are transmitted from left to right (from least significant bit to most significant bit unless specified otherwise). The Ethernet MAC can handle jumbo Ethernet frames where the data field can be much larger than 1,500 bytes.

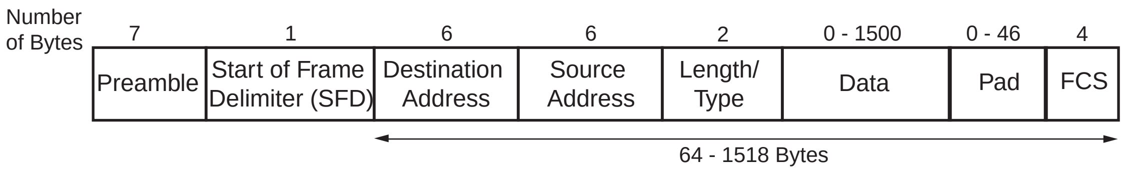


Figure 3-4: **Standard Ethernet Frame Format**

The Ethernet MAC can also accept Virtual LAN (VLAN) frames. The VLAN frame format is shown in Figure 3-5. If the frame is a VLAN type frame, the Ethernet MAC accepts four additional bytes.

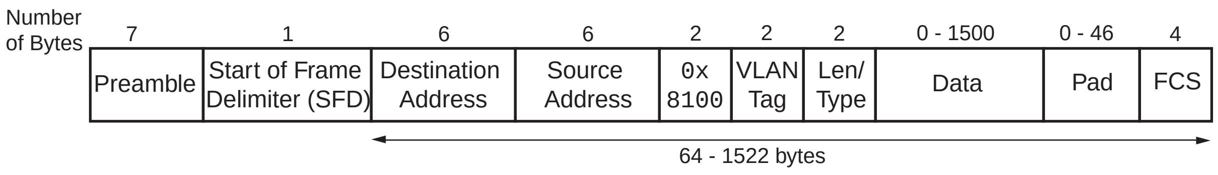


Figure 3-5: **Ethernet VLAN Frame Format**

Ethernet PAUSE/flow control frames can be transmitted and received by the Ethernet MAC. Figure 3-35, page 68 shows how a PAUSE/flow control frame differs from the standard Ethernet frame format.

The following subsections describe the individual fields of an Ethernet frame and some basic functionality of the Ethernet MAC.

Preamble

For transmission, this field is automatically inserted by the Ethernet MAC. The preamble field was historically used for synchronization and contains seven bytes with the pattern

`0x55`, transmitted from left to right. For reception, this field is always stripped from the incoming frame, before the data is passed to the user.

Some applications use the time occupied by the preamble bytes to send network information around without overhead. This is supported by the custom preamble mode in the MAC core.

Start of Frame Delimiter

The start of frame delimiter field marks the start of the frame and must contain the pattern `0xD5`. For transmission on the physical interface, this field is automatically inserted by the Ethernet MAC. For reception, this field is always stripped from the incoming frame before the data is passed to the user.

MAC Address Fields

MAC Address

The least significant bit of the first octet of a MAC address determines if the address is an individual/unicast (`0`) or group/multicast (`1`) address. Multicast addresses are used to group logically related stations. The broadcast address (destination address field is all `1`s) is a multicast address that addresses all stations on the Local Area Network (LAN). The Ethernet MAC supports transmission and reception of unicast, multicast, and broadcast packets.

The address is transmitted in an Ethernet frame least significant bit first: so the bit representing an individual or group address is the first bit to appear in an address field of an Ethernet frame.

Destination Address

This MAC Address field is the first field of the Ethernet frame that is always provided in the packet data for transmissions and is always retained in the receive packet data. It provides the MAC address of the intended recipient on the network.

Source Address

This MAC Address field is the second field of the Ethernet frame that is always provided in the packet data for transmissions and is always retained in the receive packet data. It provides the MAC address of the frame initiator on the network.

For transmission, the source address of the Ethernet frame should always be provided by the user because it is unmodified by the Ethernet MAC.

Length/Type

The value of this field determines if it is interpreted as a length or a type field, as defined by *IEEE Std 802.3-2008*. A value of 1536 decimal or greater is interpreted by the Ethernet MAC as a type field.

When used as a length field, the value in this field represents the number of bytes in the following data field. This value does not include any bytes that can be inserted in the pad field following the data field.

A length/type field value of `0x8100` indicates that the frame is a VLAN frame, and a value of `0x8808` indicates a PAUSE MAC control frame.

For transmission, the Ethernet MAC does not perform any processing of the length/type field.

For reception, if this field is a length field, the Ethernet MAC receive engine interprets this value and removes any padding in the pad field (if necessary). If the field is a length field and length/type checking is enabled, the Ethernet MAC compares the length against the actual data field length and flags an error if a mismatch occurs. If the field is a type field, the Ethernet MAC ignores the value and passes it along with the packet data with no further processing. The length/type field is always retained in the receive packet data.

Data

The data field can vary from 0 to 1,500 bytes in length for a normal frame. The Ethernet MAC can handle jumbo frames of any length.

This field is always provided in the packet data for transmissions and is always retained in the receive packet data.

Pad

The pad field can vary from 0 to 46 bytes in length. This field is used to ensure that the frame length is at least 64 bytes in length (the preamble and SFD fields are not considered part of the frame for this calculation), which is required for successful CSMA/CD operation. The values in this field are used in the frame check sequence calculation but are not included in the length field value, if it is used. The length of this field and the data field combined must be at least 46 bytes. If the data field contains 0 bytes, the pad field is 46 bytes. If the data field is 46 bytes or more, the pad field has 0 bytes.

For transmission, this field can be inserted automatically by the Ethernet MAC or can be supplied by the user. If the pad field is inserted by the Ethernet MAC, the FCS field is calculated and inserted by the Ethernet MAC. If the pad field is supplied by the user, the FCS can be either inserted by the Ethernet MAC or provided by the user, as indicated by a configuration register bit.

For reception, if the length/type field has a length interpretation, any pad field in the incoming frame is not be passed to the user, unless the Ethernet MAC is configured to pass the FCS field on to the user.

FCS

The value of the FCS field is calculated over the destination address, source address, length/type, data, and pad fields using a 32-bit Cyclic Redundancy Check (CRC), as defined in *IEEE Std 802.3-2008* para. 3.2.8:

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x^{1} + x^{0}$$

The CRC bits are placed in the FCS field with the x^{31} term in the left-most bit of the first byte, and the x^0 term is the right-most bit of the last byte (that is, the bits of the CRC are transmitted in the order x^{31}, x^{30},..., x^1, x^0).

For transmission, this field can be either inserted automatically by the Ethernet MAC or supplied by the user, as indicated by a configuration register bit.

For reception, the incoming FCS value is verified on every frame. If an incorrect FCS value is received, the Ethernet MAC indicates to the user that it has received a bad frame. The FCS field can either be passed on to the user or be dropped by the Ethernet MAC, as indicated by a configuration register bit.

Frame Transmission and Interframe Gap

Frames are transmitted over the Ethernet medium with an interframe gap, as specified by the *IEEE Std 802.3-2008*, to be 96-bit times (9.6 ns for 10 Gb/s). This value is a minimum value and can be increased with a resulting decrease in throughput.

After the last bit of an Ethernet MAC frame transmission, the Ethernet MAC starts the interframe gap timer and defers transmissions until the IFG count completes. The Ethernet MAC then places the Start ordered set code of the next frame on the next available 4-byte boundary in the data stream. This can be further delayed if IFG Adjustment feature of the Ethernet MAC is used.

Deficit Idle Count

In addition to the interframe gap setting described above, the *IEEE 802.3-2008* standard also permits a feature called Deficit Idle Count. This allows periodic shortening of the transmitted interframe gap below 12 to satisfy the Start ordered set alignment rules, as long as a mean value of 12 is maintained over a long period of time. This feature is controlled in the Ethernet MAC by a configuration bit.

Interfacing to the Data Interfaces

This section describes how to connect to the data interfaces of the 10-Gigabit Ethernet MAC core.

Interfacing to the Transmit AXI4-Stream Interface

AXI4-Stream Interface – Transmit

The client-side interface on the transmit side of XGMAC supports an AXI4-Stream interface. It has a 64-bit datapath with eight control bits to delineate bytes within the 64-bit port. Additionally, there are signals to handshake the transfer of data into the core. An example design which includes source code for a FIFO with an AXI4-Stream interface is provided with the core generated by the Xilinx CORE Generator tool. Table 3-1 defines the signals.

Table 3-1: **Transmit Client-Side Interface Port Description**

Name	Direction	Description
tx_axis_aresetn	In	AXI4-Stream active-Low reset for Transmit path XGMAC
tx_axis_tdata[63:0]	In	AXI4-Stream data to XGMAC
tx_axis_tkeep[7:0]	In	AXI4-Stream Data Control to XGMAC
tx_axis_tvalid	In	AXI4-Stream Data Valid input to XGMAC
tx_axis_tuser	In	AXI4-Stream user signal used to indicate explicit underrun
tx_axis_tlast	In	AXI4-Stream signal to XGMAC indicating End of Ethernet Packet
tx_axis_tready	Out	AXI4-Stream acknowledge signal from XGMAC to indicate to start the Data transfer
tx_ifg_delay[7:0]	In	Configures Interframe Gap adjustment between packets.

For transmit data `tx_axis_tdata[63:0]` (Table 3-2), the port is logically divided into lane 0 to lane 7, with the corresponding bit of the `tx_axis_tkeep` word signifying valid data on `tx_axis_tdata`.

Table 3-2: **tx_axis_tdata Lanes**

Lane/tx_axis_tkeep Bit	tx_axis_tdata Bits
0	7:0
1	15:8
2	23:16
3	31:24
4	39:32
5	47:40

Table 3-2: **tx_axis_tdata Lanes *(Cont'd)***

Lane/tx_axis_tkeep Bit	tx_axis_tdata Bits
6	55:48
7	63:56

Normal Frame Transmission

The timing of a normal frame transfer is shown in Figure 3-6. When the client wants to transmit a frame, it asserts the `tx_axis_tvalid` and places the data and control in `tx_axis_tdata` and `tx_axis_tkeep` in the same clock cycle. After the core asserts `tx_axis_tready` to acknowledge the first beat of data, on the next and subsequent clock edges, the client must provide the remainder of the data for the frame to the core. The end of packet is indicated to the core by `tx_axis_tlast` asserted for 1 cycle. The bits of `tx_axis_tkeep` are set appropriately if the packet ends at a non-64-bit boundary. For example, in Figure 3-6, the first packet ends at Lane 3 and any data after that is ignored.

After `tx_axis_tlast` is deasserted, any data and control is deemed invalid until `tx_axis_tvalid` is next asserted.

If custom preamble is enabled, `tx_axis_tready` signal might not be deasserted at the end of the frame to read the custom preamble into the core for the following frame.

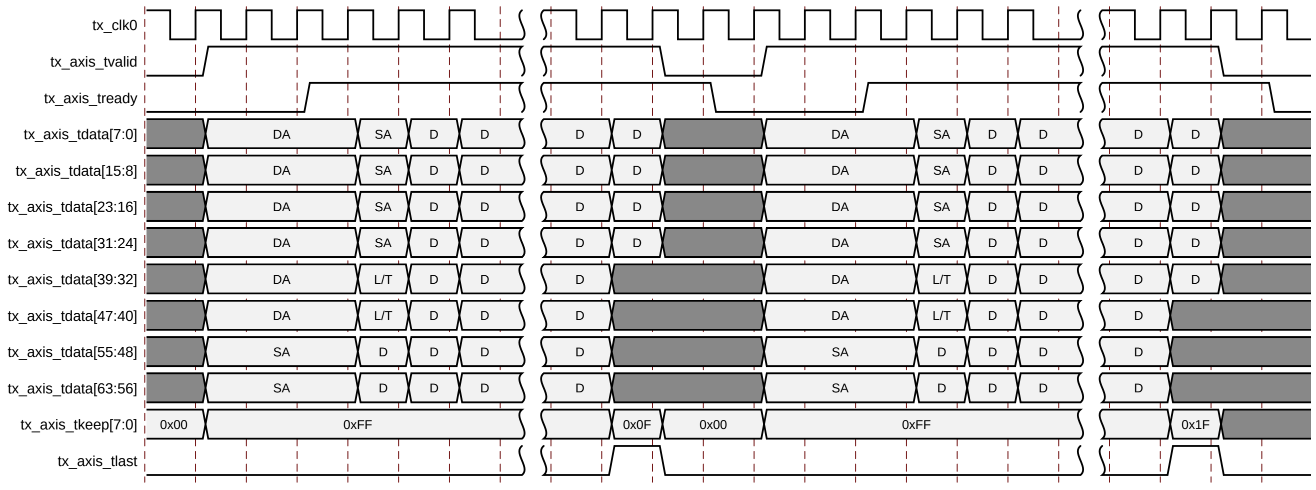


Figure 3-6: **Frame Transmission**

In-Band Ethernet Frame Fields

For maximum flexibility in switching applications, the Ethernet frame parameters (destination address, source address, length/type and optionally FCS) are encoded within the same data stream that the frame payload is transferred on, rather than on separate ports. This is illustrated in the timing diagrams. The destination address must be supplied with the first byte in lane 0 and so on. Similarly, the first byte of the source address must be supplied in lane 6 of the first transfer.The length/type field is similarly encoded, with the first byte placed into lane 4. The definitions of the abbreviations used in the timing diagrams are described in Table 3-3.

Table 3-3: **Abbreviations Used in Timing Diagrams**

Abbreviation	Definition
DA	Destination address
SA	Source address
L/T	Length/type field
FCS	Frame check sequence (CRC)

Padding

When fewer than 46 bytes of data are supplied by the client to the Ethernet MAC core, the transmitter module adds padding up to the minimum frame length, unless the Ethernet MAC core is configured for in-band FCS passing. In the latter case, the client must also supply the padding to maintain the minimum frame length. When in-band FCS is enabled, if the client does not provide a frame with at least 46 bytes of data, the frame is terminated correctly but not padded.

Transmission with In-Band FCS Passing

If the Ethernet MAC core is configured to have the FCS field passed in by the client on the AXI4-Stream Transmit interface, the transmission timing is as shown in Figure 3-7. In this case, it is the responsibility of the client to ensure that the frame meets the Ethernet minimum frame length requirements; the Ethernet MAC core does not perform any padding of the payload. If the client transmits a frame less than the Ethernet minimum length requirement, then the frame is not counted in the statistics.

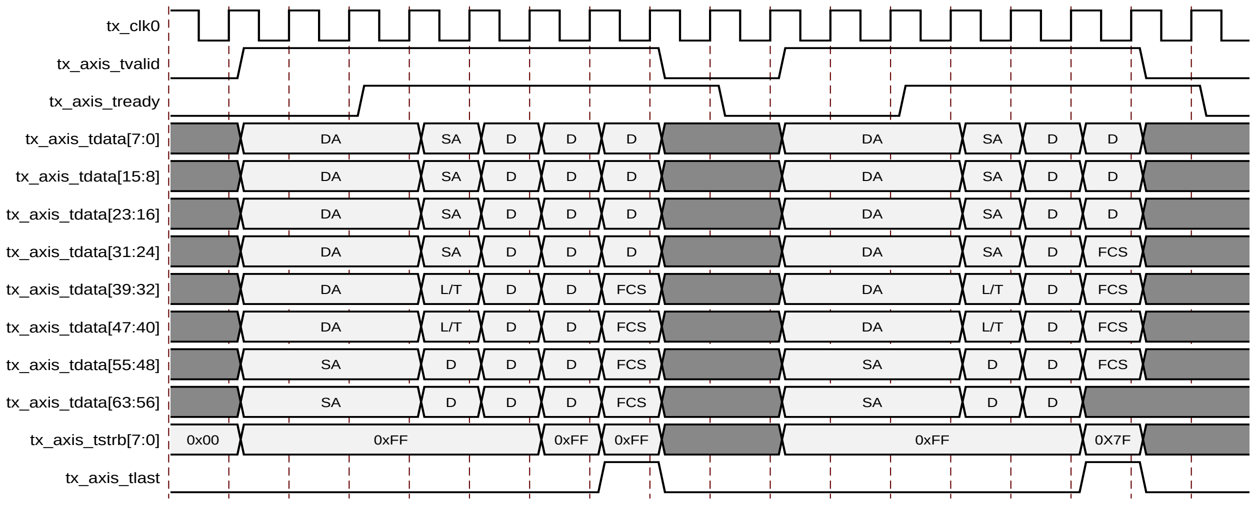


Figure 3-7: **Transmission with In-Band FCS Passing**

Aborting a Transmission

The aborted transfer of a packet on the client interface is called an underrun. This can happen, for instance, if a FIFO in the AXI Transmit client interface empties before a frame is completed. This is indicated to the core in one of two ways:

1. An explicit underrun, in which a Frame Transfer is aborted by asserting `tx_axis_tuser` High while `tx_axis_tvalid` is High and data transfer is continuing. (See Figure 3-8)
   An underrun packet must have the DA, SA, L/T fields in it. This is true even if Custom Preamble is enabled for transmission.
2. An implicit underrun, in which a Frame Transfer is aborted by deasserting `tx_axis_tvalid` without asserting `tx_axis_tlast`. (See Figure 3-9)

Figure 3-8 and Figure 3-9 each show an underrun frame followed by a complete frame. When either of the two scenarios occurs during a frame transmission, the Ethernet MAC core inserts error codes into the XGMII data stream to flag the current frame as an errored frame and then the Ethernet MAC core falls back to idle transmission. The `tx_mac_underun` signal shown on the diagram is an internal signal. It remains the responsibility of the client to re-queue the aborted frame for transmission, if necessary.

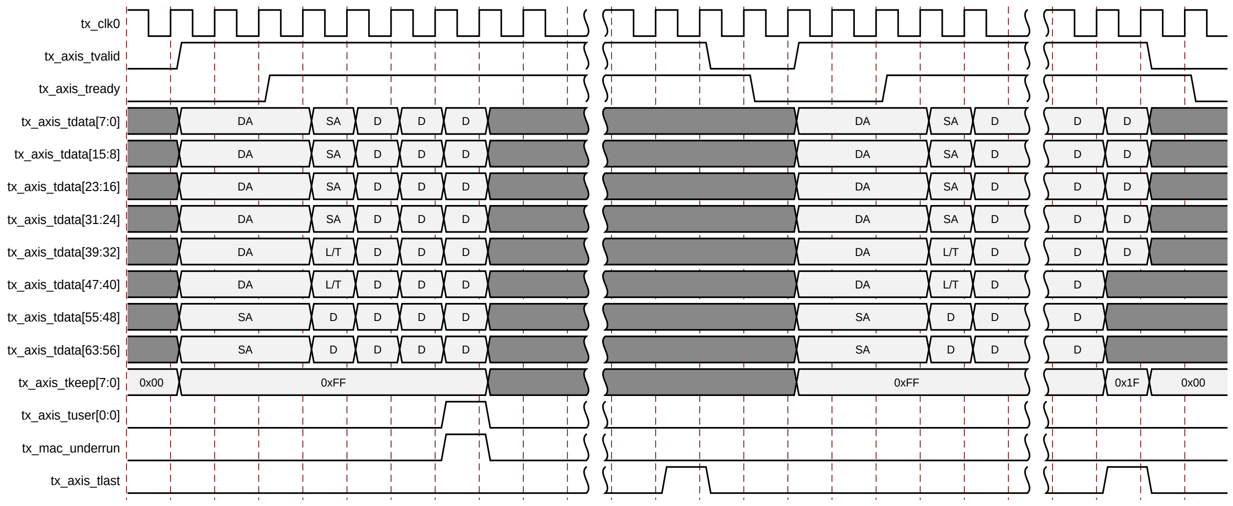


Figure 3-8: **Frame Transfer Abort with tx_axis_tuser asserted**

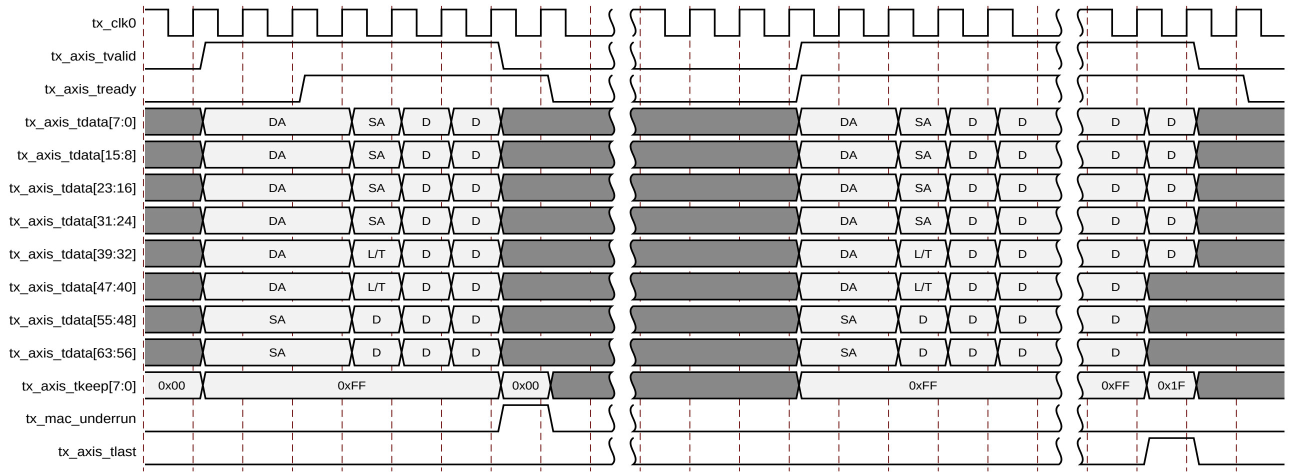


Figure 3-9: **Frame Transfer Abort with tx_axis_tvalid deasserted**

Note: Aborting a frame transfer using the mechanism shown in Figure 3-9 is not fully AXI4-compliant, as no TLAST is asserted to complete the first frame. If AXI4-compliance is important, use the scheme of Figure 3-8.

Back-to-Back Continuous Transfers

Continuous data transfer on Transmit AXI4-Stream interface is possible, as the signal `tx_axis_tvalid` can remain continuously High, with packet boundaries defined solely by `tx_axis_tlast` asserted for the end of the Ethernet packet. However, the Ethernet MAC core can defer the `tx_axis_tready` acknowledgement signal to comply with the inter-packet gap requirements on the XGMII side of the core.

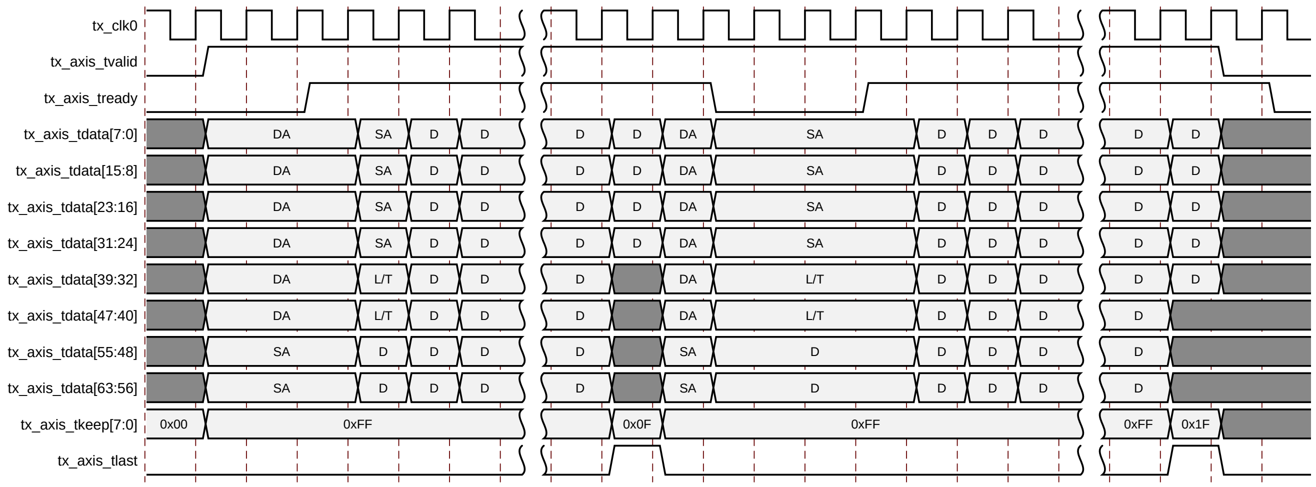


Figure 3-10: **Back-to-Back Continuous Transfer on Transmit Client Interface**

Transmission of Custom Preamble

You can elect to use a custom preamble field. If this function is selected (using a configuration bit, see Configuration Registers, page 21), the standard preamble field can be substituted for custom data. The custom data must be supplied on `tx_axis_tdata[63:8]` in the first column when `tx_axis_tvalid` is first asserted High. Transmission of Custom Preamble can happen in both continuous and non-continuous mode of `tx_axis_tvalid`. Figure 3-11 shows a frame presented at the Transmit Client Interface with a custom preamble where P1 to P7 denote the custom data bytes when `tx_axis_tvalid` is deasserted after `tx_axis_tlast`. Figure 3-12 illustrates the transmission of a custom preamble when `tx_axis_tvalid` remains asserted after `tx_axis_tlast` is asserted.

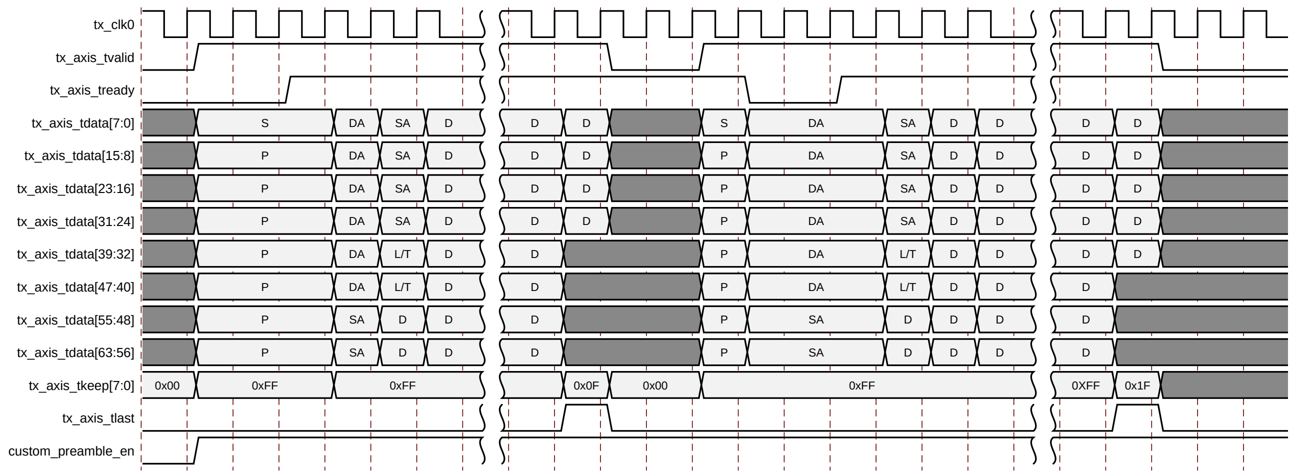


Figure 3-11: **Transmission of Custom Preamble in the Non-Continuous Case**

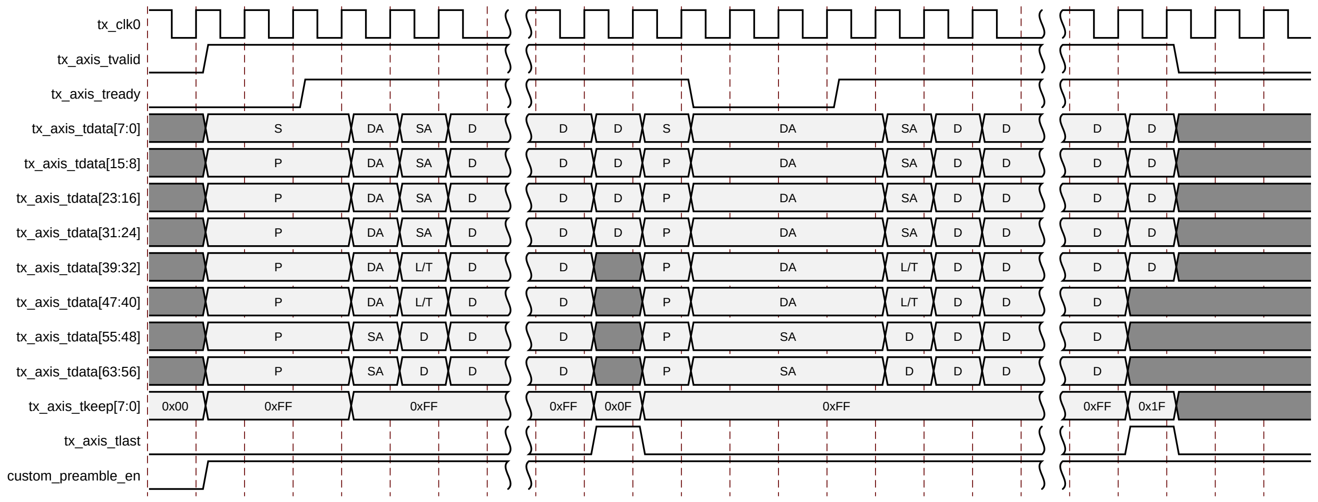


Figure 3-12: **Transmission of Custom Preamble in the Continuous Case**

The Ethernet MAC core substitutes the IEEE standard preamble with that supplied by the client logic. Figure 3-13 shows the transmission of a frame with custom preamble (P1 to P7) at the XGMII interface.

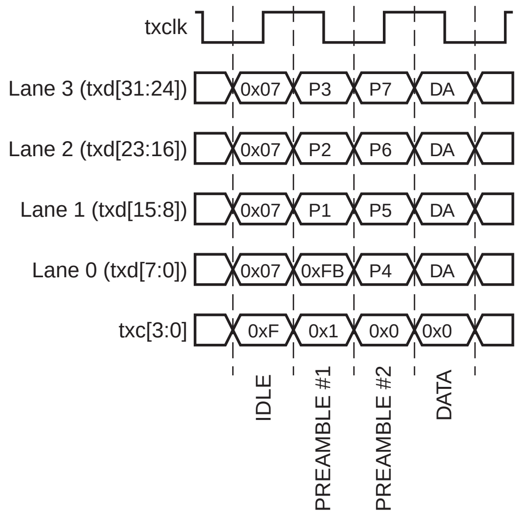


Figure 3-13: **XGMII Frame Transmission of Custom Preamble**

VLAN Tagged Frames

Transmission of a VLAN tagged frame (if enabled) is shown in Figure 3-14. The handshaking signals across the interface do not change; however, the VLAN type tag 81-00 must be supplied by the client to signify that the frame is VLAN tagged. The client also supplies the two bytes of Tag Control Information, V1 and V2, at the appropriate times in the data stream. Additional information about the contents of these two bytes is available in *IEEE Standard 802.3-2008* [Ref 1].

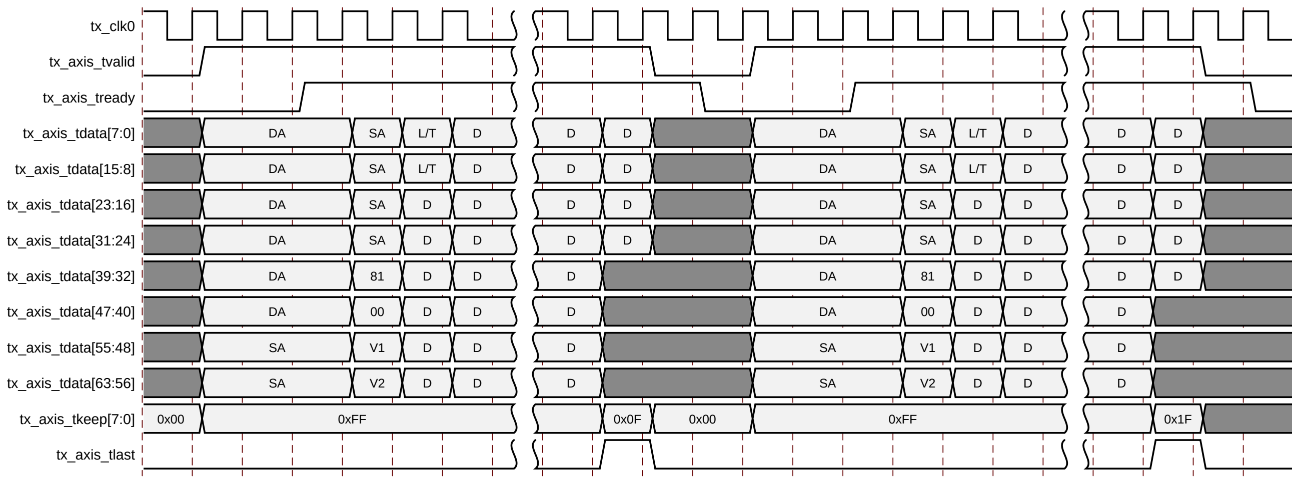


Figure 3-14: **VLAN Tagged Frame Transmission**

Transmitter Maximum Permitted Frame Length

The maximum legal length of a frame specified in *IEEE Standard 802.3-2008* is 1,518 bytes for non-VLAN tagged frames. VLAN tagged frames might be extended to 1,522 bytes. When jumbo frame handling is disabled and the client attempts to transmit a frame which exceeds the maximum legal length, the Ethernet MAC core inserts an error code to corrupt the current frame and the frame is truncated to the maximum legal length. When jumbo frame handling is enabled, frames which are longer than the legal maximum are transmitted error free.

If required, a custom Maximum Transmission Unit (MTU) can be programmed into the core. This allows the transmission of frames up to the programmed MTU size by the core, rather than the 1,518/1,522 byte limit. The programmed MTU must be equal to or greater than 1,518 bytes.

Any Frame transmitted greater than the MTU Frame Size, if Jumbo Frame is disabled, is signaled as a bad frame; error codes are inserted and the frame is truncated.

For details on enabling and disabling jumbo frame handling, see Configuration Registers, page 21.

Note: There are interactions between the configuration bits affecting frame length handling that the user should be aware of. Firstly, if Jumbo Enable and MTU Frame Transfer Enable are enabled at the same time, the Jumbo Enable takes precedence. Secondly, if VLAN Enable and MTU Frame Transfer Enable are both turned on, then MTU frame length rules apply.

Interframe Gap Adjustment

You can elect to vary the length of the interframe gap. If this function is selected (using a configuration bit, see Configuration Registers, page 21), the Ethernet MAC exerts back pressure to delay the transmission of the next frame until the requested number of XGMII columns has elapsed. The number of XGMII columns is controlled by the value on the `tx_ifg_delay` port. The minimum interframe gap of three XGMII columns (12 bytes) is always maintained. Figure 3-15 shows the Ethernet MAC operating in this mode.

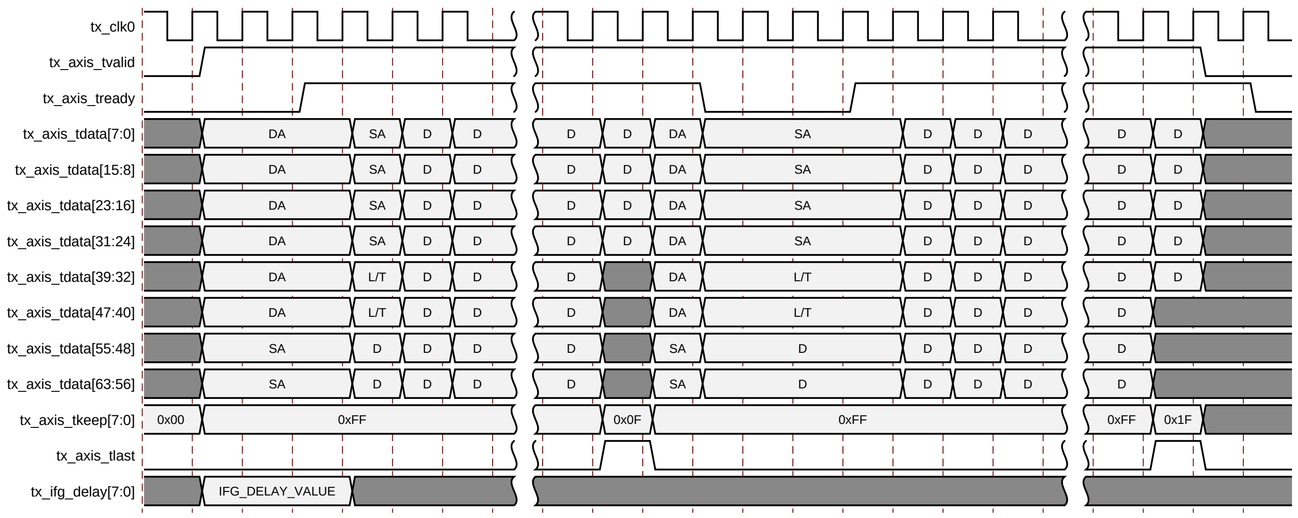


Figure 3-15: **Interframe Gap Adjustment**

Deficit Idle Count (DIC)

The Transmit side XGMAC supports Interframe Gap obtained through Deficit Idle Count to maintain the effective data rate of 10 Gb/s as described in *IEEE Standard 802.3-2008* [Ref 1]. This feature is supported even when the AXI4-Stream sends Ethernet packets with In Band FCS or without FCS. It is also supported when Custom Preamble is enabled for transmission. However, the requirement from the Transmit Streaming Interface is that to maintain the effective data rate of 10 Gb/s through the IPG adjustment with DIC, `axis_tx_tvalid` must be maintained continuously High.

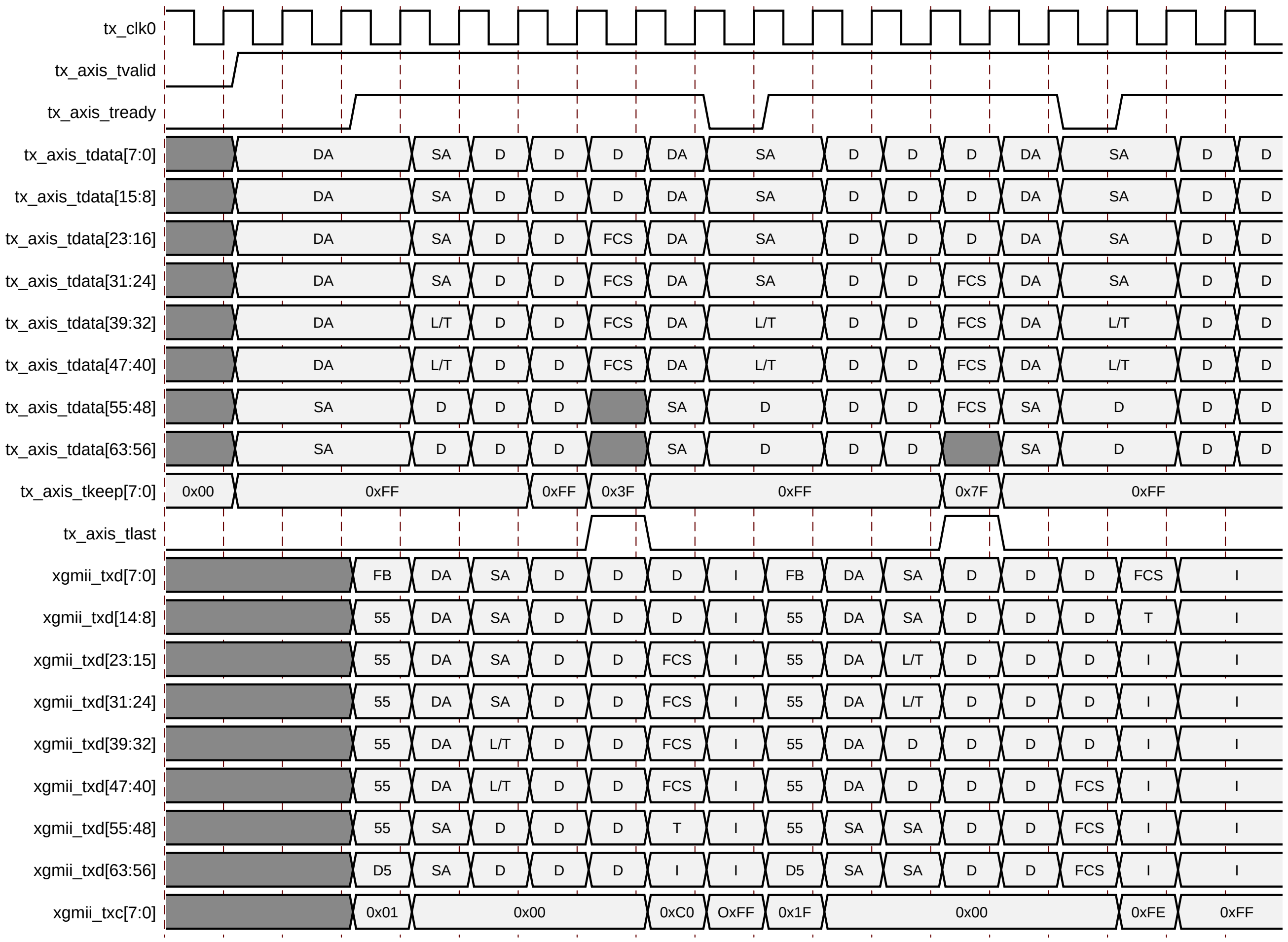


Figure 3-16: **Back-to-Back Continuous Transfer on Transmit XGMII Interface**

Transmission of Frames During Local/Remote Fault Reception

When a local or remote fault has been received, the core might not transmit frames if Fault Inhibit has been disabled (using a configuration bit, see Configuration Registers, page 21). When Fault Inhibit is disabled, the Reconciliation Sublayer transmits ordered sets as presented in *IEEE Standard 802.3-2008* [Ref 1]; that is, when the RS is receiving Local Fault ordered sets, it transmits Remote Fault ordered sets. When receiving Remote Fault ordered sets, it transmits idle code words. If the management interface is included with the core, the status of the local and remote fault register bits can be monitored (bits 28 and 29 of the Reconciliation Sublayer configuration word, address 0x300) and when they are both clear, the core is ready to accept frames for transmission. If the management interface is not included with the core, the status of the local and remote fault register bits can be monitored on bits 0 and 1 of the status vector.

Note: Any frames presented at the client interface prior to both register bits being clear are dropped silently by the core.

When Fault Inhibit mode is enabled, the core transmits data normally regardless of received Local Fault or Remote Fault ordered sets.

Interfacing to the Receive AXI4-Stream Interface

Normal Frame Reception

The client-side interface on receive side of XGMAC supports the AXI4-Stream interface. It has a 64-bit datapath with eight control bits to delineate bytes within the 64-bit port. Additionally, there are signals to indicate to the user logic the validity of the previous frame received. Table 3-4 defines the signals.

Table 3-4: **Receive Client-Side Interface Port Description**

Name	Direction	Description
rx_axis_aresetn	In	AXI4-Stream active-Low reset for Receive path XGMAC
rx_axis_tdata	Out	AXI4-Stream Data from XGMAC to upper layer
rx_axis_tkeep	Out	AXI4-Stream Data Control from XGMAC to upper layer
rx_axis_tvalid	Out	AXI4-Stream Data Valid from XGMAC
rx_axis_tuser	Out	AXI4-Stream User Sideband Interface from XGMAC 0 indicates a bad packet has been received. 1 indicates a good packet has been received.
rx_axis_tlast	Out	AXI4-Stream signal from XGMAC indicating an end of packet

For the receive data port `rx_axis_tdata[63:0]` (Table 3-5), the port is logically divided into lane 0 to lane 7, with the corresponding bit of the `rx_axis_tkeep` word signifying valid data on the `rx_axis_tdata`.

Table 3-5: **rx_axis_tdata Lanes**

Lane/rx_axis_tkeep Bit	rx_axis_tdata Bits
0	7:0
1	15:8
2	23:16
3	31:24
4	39:32

Table 3-5: **rx_axis_tdata Lanes *(Cont'd)***

Lane/rx_axis_tkeep Bit	rx_axis_tdata Bits
5	47:40
6	55:58
7	63:56

The timing of a normal inbound frame transfer is represented in Figure 3-17. The client must be prepared to accept data at any time; there is no buffering within the Ethernet MAC to allow for latency in the receive client. When frame reception begins, data is transferred on consecutive clock cycles to the receive client. The `rx_axis_mac_tlast` and `rx_axis_mac_tuser` signals are asserted, along with the final bytes of the transfer, only after all frame checks are completed. This is after the FCS field has been received.'The Ethernet MAC asserts the `rx_axis_tuser` signal to indicate that the frame was successfully received and that the frame should be analyzed by the client. This is also the end of packet signaled by `tx_axis_tlast` asserted for 1 cycle.

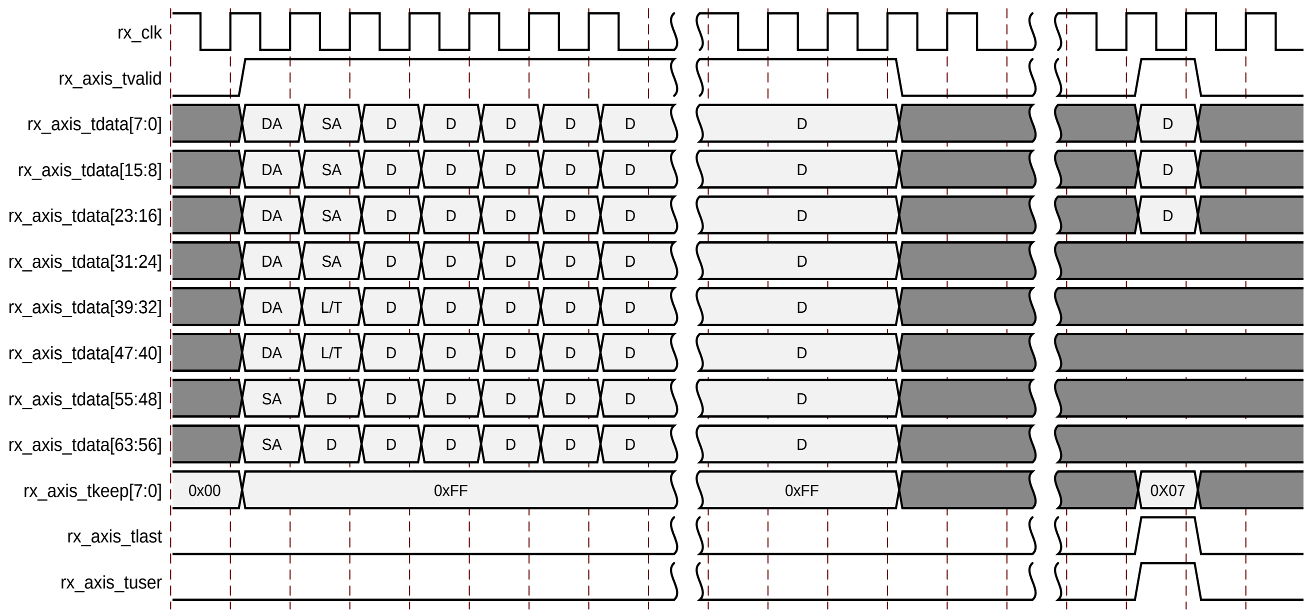


Figure 3-17: **Reception of a Good Frame**

Timing for a Good or a Bad Frame

As can be seen in the Figure 3-17, there is always a gap of up to seven clocks, by which the indication of a good frame or a bad frame is signaled through `rx_axis_tuser` set to 1 or a 0 and `rx_axis_tlast` asserted. This status is only indicated when all frame checks are completed. This can be up to seven clock cycles after the last valid data is presented when the Length/Type field in the frame is valid; for example, this can result from padding at the end of the Ethernet frame. If the Length/Type field in the frame is incorrect and the frame is longer than indicated, then it is a bad frame and hence `rx_axis_tlast` might be deasserted significantly earlier than seven clock cycles after the end of valid data. Although good frame reception is illustrated, the same timing applies to a bad frame. Either the good frame or bad frame signaled through `rx_axis_tuser` and `rx_axis_tlast` is, however, always asserted before the next frame data begins to appear on `rx_axis_tdata`.

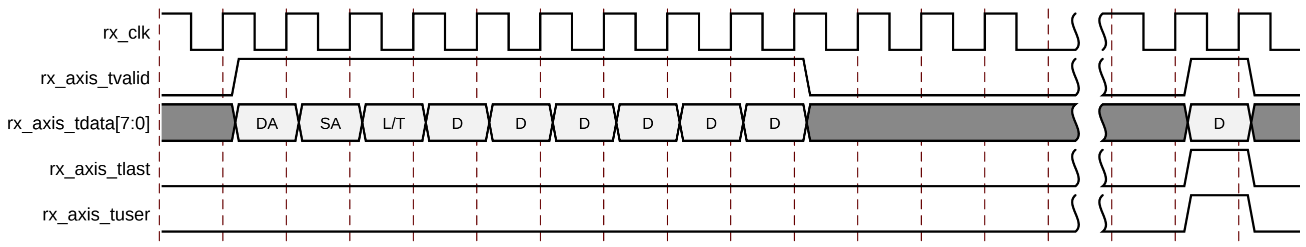


Figure 3-18: **Timing of a Good Frame**

Frame Reception with Errors

The case of an unsuccessful frame reception (for example, a runt frame or a frame with an incorrect FCS) can be seen in Figure 3-19. In this case, the bad frame is received and the signal `rx_axis_tuser` is deasserted to the client at the end of the frame. It is then the responsibility of the client to drop the data already transferred for this frame.

The following conditions cause the assertion of `rx_axis_tlast` along with `rx_axis_tuser` = 0 signifying a bad_frame:

- FCS errors occur.
- Packets are shorter than 64 bytes (undersize or fragment frames).
- Jumbo frames are received when jumbo frames are not enabled.
- Frames of length greater than the MTU Size programmed are received, MTU Size Enable Frames are enabled, and jumbo frames are not enabled.
- The length/type field is length, but the real length of the received frame does not match the value in the length/type field (when length/type checking is enabled).
- The length/type field is length, in which the length value is less than 46. In this situation, the frame should be padded to minimum length. If it is not padded to exactly minimum frame length, the frame is marked as bad (when length/type checking is enabled).
- Any control frame that is received is not exactly the minimum frame length unless Control Frame Length Check Disable is set.
- The XGMII data stream contains error codes.
- A valid pause frame, addressed to the Ethernet MAC, is received when flow control is enabled.

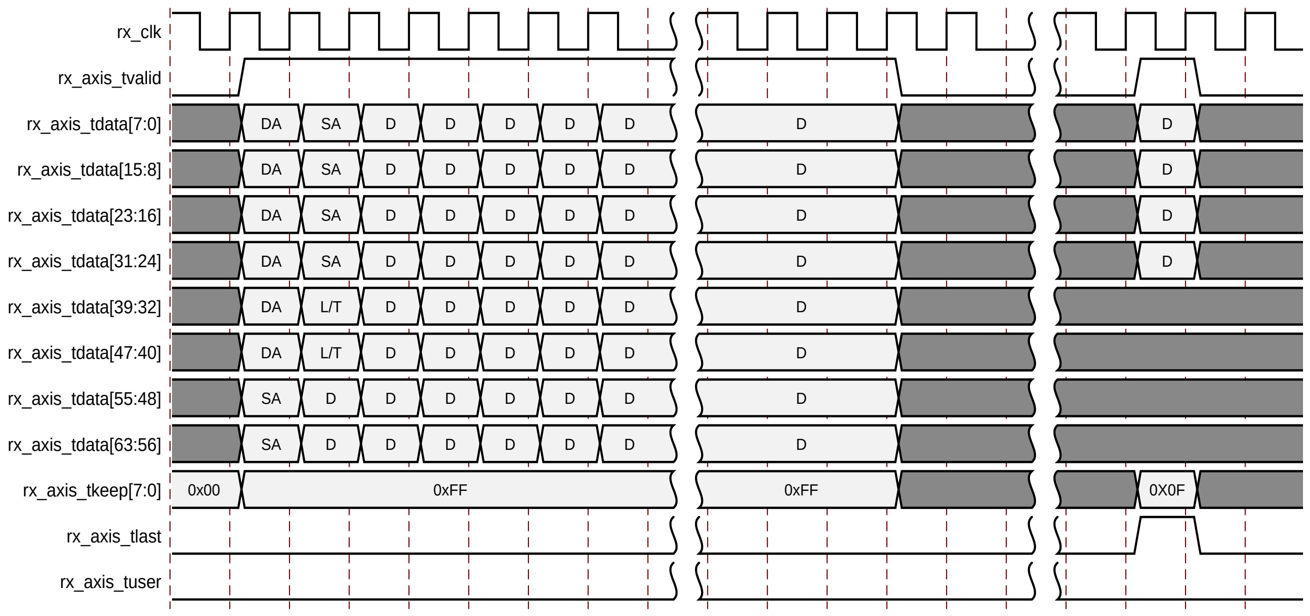


Figure 3-19: **Frame Reception with Error**

Reception with In-Band FCS Passing

Figure 3-20 illustrates the Ethernet MAC core configured to pass the FCS field to the client (see Configuration Registers, page 21). In this case, any padding inserted into the frame to meet the Ethernet minimum frame length specifications is left intact and passed to the client. Although the FCS is passed up to the client, it is also verified by the Ethernet MAC core, and `rx_axis_tuser` is 0 when `rx_axis_tlast` is asserted if the FCS check fails.

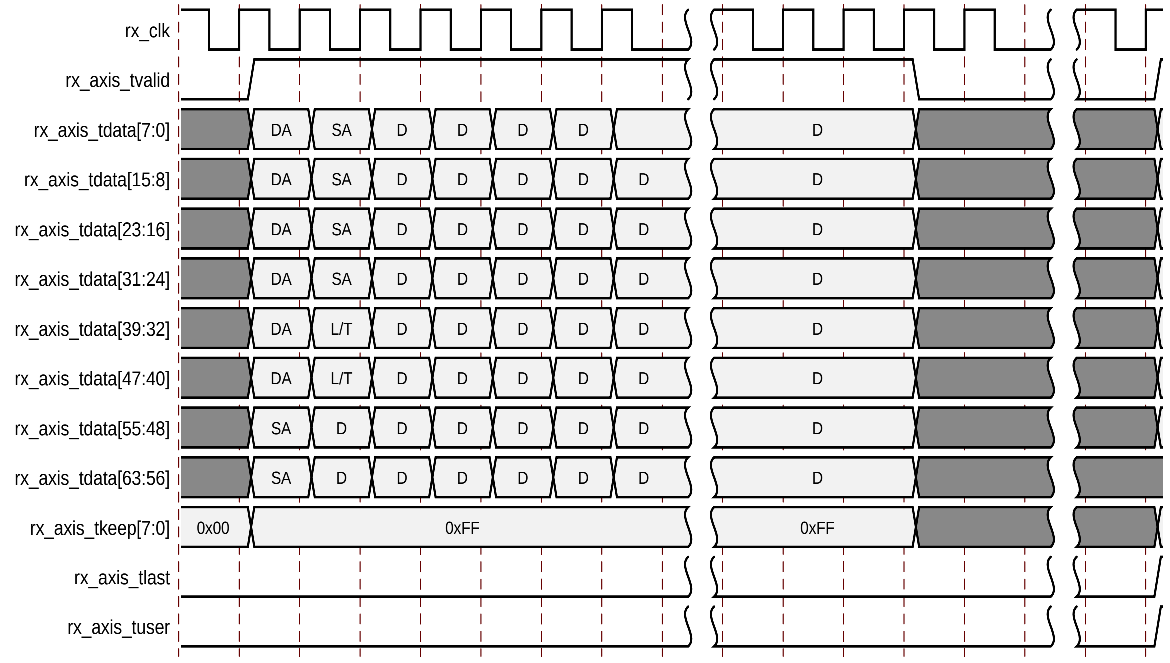


Figure 3-20: **Frame Reception with In-Band FCS Passing**

Reception of Custom Preamble

You can elect to use a custom preamble field. If this function is selected (using a configuration bit, see Configuration Registers, page 21), the preamble field can be recovered from the received data and presented on the Client AXI4-Stream receive Interface. If this mode is enabled, the custom preamble data is present on `rx_axis_tdata[63:8]`. The `rx_axis_tkeep` output is asserted to frame the custom preamble. Figure 3-21 shows the reception of a frame with custom preamble.

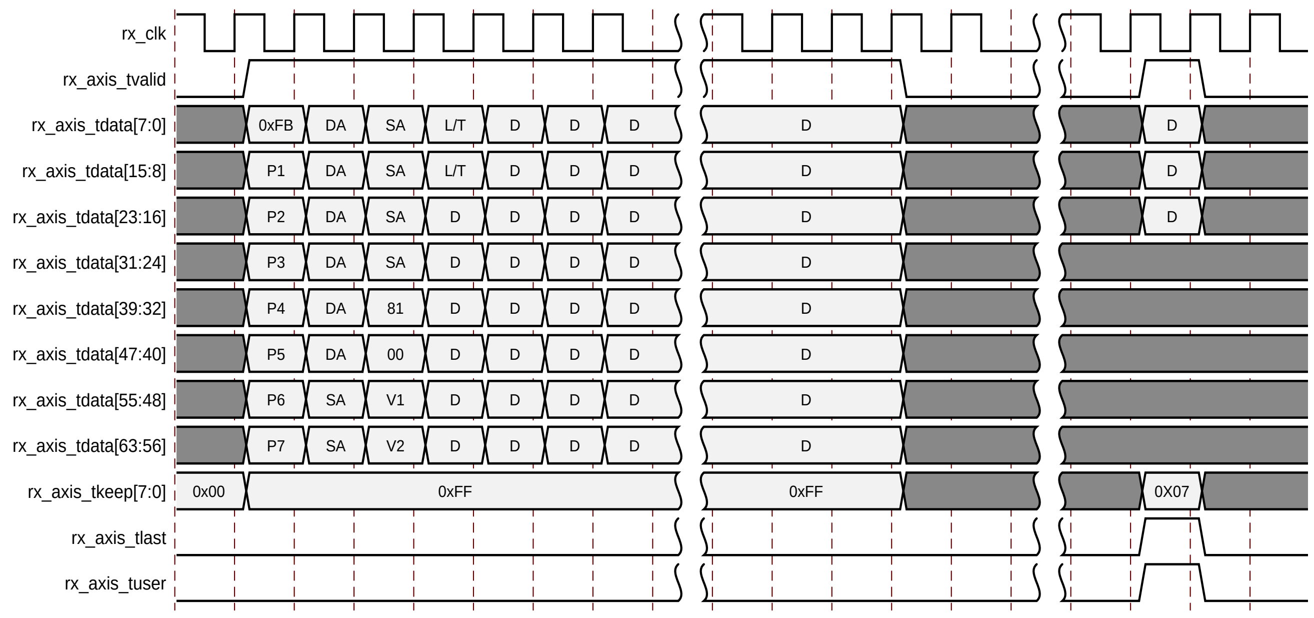


Figure 3-21: **Frame Reception with Custom Preamble**

VLAN Tagged Frames

The reception of a VLAN tagged frame (if enabled) is represented in Figure 3-22. The VLAN frame is passed to the client so that the frame can be identified as VLAN tagged; this is followed by the Tag Control Information bytes, V1 and V2. More information on the interpretation of these bytes can be found in *IEEE Standard 802.3-2008* [Ref 1]. All VLAN tagged frames are treated as Type frames, that is, any padding is treated as valid and passed to the client.

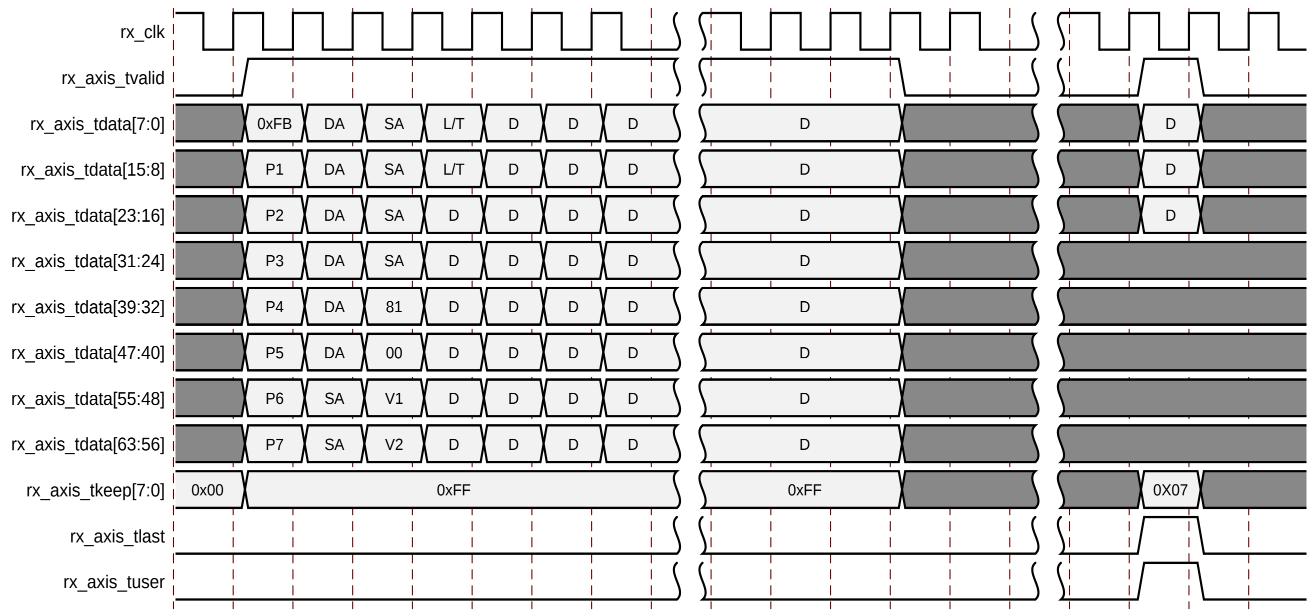


Figure 3-22: **Frame Reception with VLAN Tagged Frames**

Receiver Maximum Permitted Frame Length

The maximum legal length of a frame specified in *IEEE Standard 802.3-2008* [Ref 1] is 1,518 bytes for non- VLAN tagged frames. VLAN tagged frames might be extended to 1,522 bytes. When jumbo frame handling is disabled and the core receives a frame which exceeds the maximum legal length, a bad frame is indicated by `rx_axis_tuser` being 0 when `rx_axis_tlast` is asserted. When jumbo frame handling is enabled, frames which are longer than the legal maximum are received in the same way as shorter frames.

If required, a custom Maximum Transmission Unit (MTU) can be programmed into the core. This allows the reception of frames up to the programmed MTU size by the core, rather than the 1,518/1,522 byte limit. The programmed MTU must be equal to or greater than 1,518 bytes.

Any Frame received greater than the MTU Frame Size, if Jumbo Frame is disabled is signaled as a bad frame.

For details on enabling and disabling jumbo Frame handling and MTU Frame handling, see Configuration Registers, page 21.

Length/Type Field Error Checks

Enabled

Default operation is with the length/type error checking enabled (see Configuration Registers, page 21). In this mode the following checks are made on all received frames. If either of these checks fail, the frame is marked as bad.

- A value in the length/type field which is greater than or equal to decimal 46, but less than 1,536, is checked against the actual data length received.
- A value in the length/type field that is less than decimal 46, (a length interpretation), the frame data length is checked to see if it has been padded to exactly 46 bytes (so that the resultant total frame length is 64 bytes).

Furthermore, if padding is indicated (the length/type field is less than decimal 46) and client-supplied FCS passing is disabled, the length value in the length/type field is used to deassert `rx_axis_tkeep[]` after the indicated number of data bytes so that the padding bytes are removed from the frame. SeeReception with In-Band FCS Passing.

Disabled

When the length/type error checking is disabled and the length/type field has a length interpretation, the Ethernet MAC does not check the length value against the actual data length received as detailed previously. A frame containing only this error is marked as good. However, if the length/type field is less than decimal 46, the Ethernet MAC marks a frame as bad if it is not the minimum frame size of 64 bytes.

If padding is indicated and client-supplied FCS passing is disabled, then a length value in the length/type field is not used to deassert `rx_axis_tkeep[]`. Instead, `rx_axis_tkeep[]` is deasserted before the start of the FCS field, and any padding is not removed from the frame.

Sending and Receiving Flow Control Frames

The flow control block is designed to clause 31 of *IEEE Standard 802.3-2008* [Ref 1]. See Overview of Flow Control, page 67 for a description of Flow Control. The Ethernet MAC can be configured to send pause frames and to act on their reception. These two behaviors can be configured asymmetrically; see Configuration Registers, page 21.

Transmitting a Pause Frame

The client sends a flow control frame by asserting `pause_req` while the pause value is on the `pause_val` bus. These signals are synchronous with respect to `tx_clk0`. The timing of this can be seen in Figure 3-23.

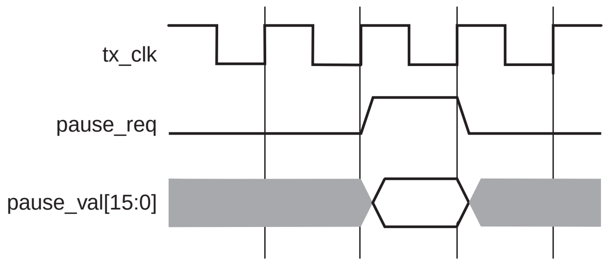


Figure 3-23: **Transmitting a Pause Frame**

If the Ethernet MAC core is configured to support transmit flow control, this action causes the Ethernet MAC core to transmit a pause control frame on the link, with the pause parameter set to the value on `pause_val` in the cycle when `pause_req` was asserted. This does not disrupt any frame transmission in progress but does take priority over any pending frame transmission. This frame is transmitted even if the transmitter is in the paused state itself.

Receiving a Pause Frame

When an error-free frame is received by the Ethernet MAC core, these checks are made:

- The destination address field is matched against the MAC control multicast address or the configured source address for the Ethernet MAC (see Configuration Registers, page 21).
- The length/type field is matched against the MAC Control type indication, 88-08
- The opcode field contents are matched against the Pause opcode

If any of these checks are false or MAC receiver flow control is disabled, the frame is ignored by the flow control logic and passed up to the client.

If the frame passes all of these checks, is of minimum legal size, and MAC receiver flow control is enabled, the pause value parameter in the frame is used to inhibit transmitter operation for the time defined in the Ethernet specification. This inhibit is implemented using the same back pressure scheme shown in Figure 3-10. Because the received pause frame has been acted on, it is passed to the client with `rx_bad_frame` asserted to indicate that it should be dropped. If Simplex Split with Receive Only is implemented then the pause frame is dropped without being acted on.

Reception of any frame for which the length/type field is the MAC Control type indication 88-08 but is not the legal minimum length is considered an invalid Control frame. It is ignored by the flow control logic and passed to the client with `rx_bad_frame` asserted.

PHY-Side Interface

External XGMII versus Internal 64-bit Interfaces

At customization time, you have the choice of selecting a 32-bit DDR XGMII PHY Interface or No Interface, which is a 64-bit SDR interface intended for internal connection. In either case, the core netlist is the same; only the example design changes, with the I/O registers and associated constraints targeting DDR or SDR operation, respectively.

Remember that although a frame to be transmitted should always be presented to the Ethernet MAC core with the start in lane 0; due to internal realignment, the start of the frame /S/ codeword might appear on the PHY side interface in lane 0 or lane 4. Likewise, the /S/ codeword might legally arrive at the RX PHY interface in either lane 0 or lane 4, but the Ethernet MAC always presents it to the client logic with start of frame in lane 0.

Interfacing to the Management Interface

This section describes the interfaces available for dynamically setting and querying the configuration and status of the 10-Gigabit Ethernet MAC core. There are two interfaces available for configuration. Depending on the core customization, only one is available in a particular core instance.

In addition, the statistics counters and vectors are described in this section as well as the use of the MDIO interface.

Management Interface

The Management Interface is an AXI4-Lite Interface. This interface is used for:

- Configuring the Ethernet MAC core
- Configuring the interrupts
- Accessing statistics information for use by high layers, for example, SNMP
- Providing access through the MDIO interface to the management registers located in the PHY attached to the Ethernet MAC core

The ports of the Management Interface are shown in Table 3-6.

Table 3-6: **Management Interface Port Description**

Name	Direction	Description
s_axi_aclk	In	AXI4-Lite clock. Range between 10 MHz and 156.25 MHz
s_axi_aresetn	In	Asynchronous active-Low reset
s_axi_awaddr[31:0]	In	Write address Bus
s_axi_awvalid	In	Write address valid
s_axi_awready	Out	Write address acknowledge
s_axi_wdata[31:0]	In	Write data bus
s_axi_wvalid	In	Write data valid
s_axi_wready	Out	Write data acknowledge
s_axi_bresp[1:0]	Out	Write transaction response. A value of b00 indicates OKAY and a value of b10 indicates SLVERR.
s_axi_bvalid	Out	Write response valid
s_axi_bready	In	Write response acknowledge
s_axi_araddr[31:0]	In	Read address Bus
s_axi_arvalid	In	Read address valid
s_axi_arready	Out	Read address acknowledge
s_axi_rdata[31:0]	Out	Read data output
s_axi_rresp[1:0]	Out	Read data response. A value of b00 indicates OKAY and a value of b10 indicates SLVERR.
s_axi_rvalid	Out	Read data/response valid
s_axi_rready	In	Read data acknowledge

Note: Only 11[10:0] out of the 32 address bits are used for decoding the read/write address.

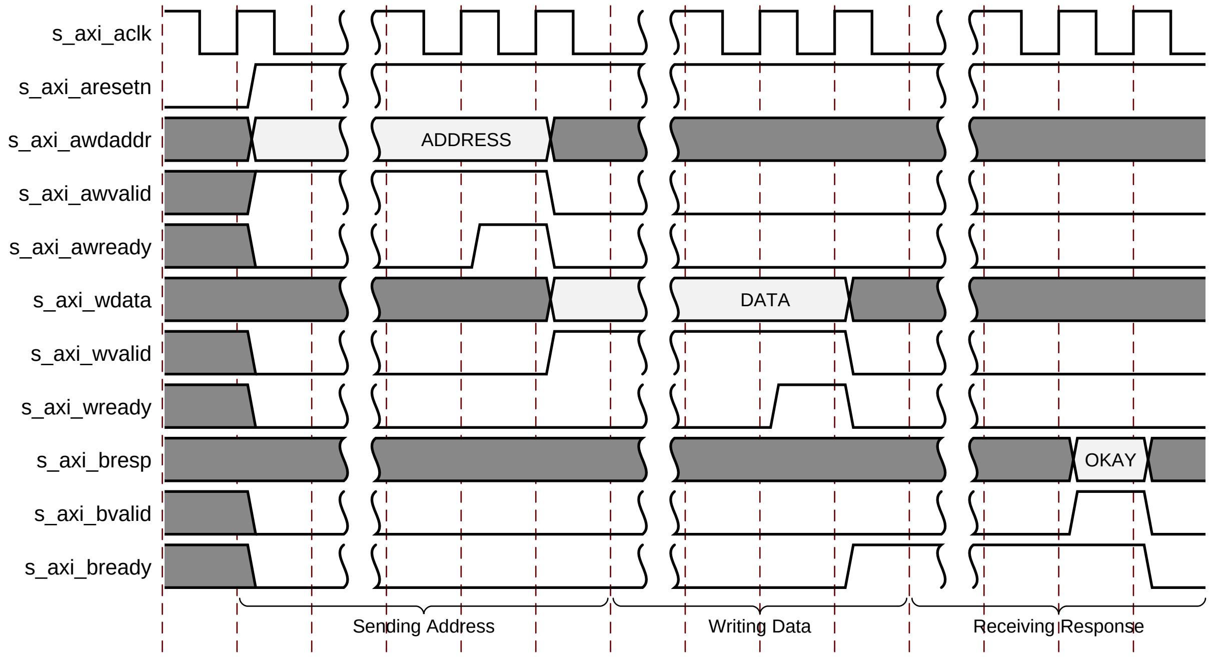


Figure 3-24: **Management Register Write Timing**

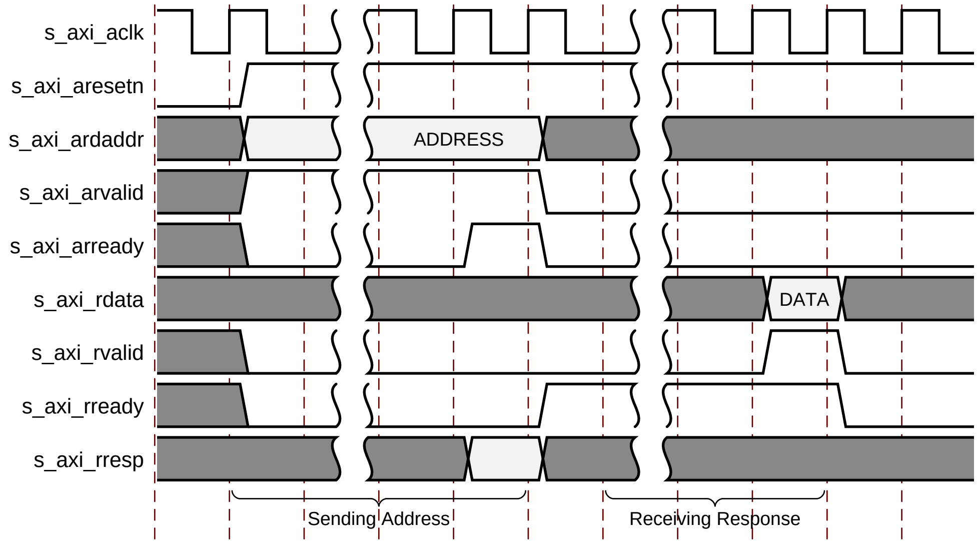


Figure 3-25: **Management Register Read Timing**

MDIO Interface

The Management Interface is used to access the MDIO Interface of the Ethernet MAC core; this interface is used to access the Managed Information Block (MIB) of the PHY components attached to the Ethernet MAC core. The ports of the MDIO interface are described in Table 2-10, page 17.

The MDIO Interface supplies a clock to the external devices, MDC. This clock is derived from the `s_axi_aclk` signal, using the value in the Clock Divide[5:0] configuration register.

The frequency of MDC is given by this equation:

$$f_{MDC} = \frac{f_{HOST\_CLK}}{(1 + \text{Clock Divide}[5:0]) \times 2} \qquad \textit{Equation 3-1}$$

The frequency of MDC given by this equation should not exceed 2.5 MHz to comply with the specification for this interface, *IEEE Standard 802.3-2008* [Ref 1]. To prevent MDC from being out of specification, the Clock Divide[5:0] value powers up at 000000, and while this value is in the register, it is impossible to enable the MDIO Interface.

MDIO Transaction initiation and completion are shown in Figure 3-26.

When MDC, PRTAD, DEVAD, and OP are programmed and MDIO is enabled, if MDIO Ready bit in the MDIO configuration register is 1, the MDIO transaction can be initiated by writing a 1 to the initiate bit (Bit[11] of MDIO Configuration word 1).

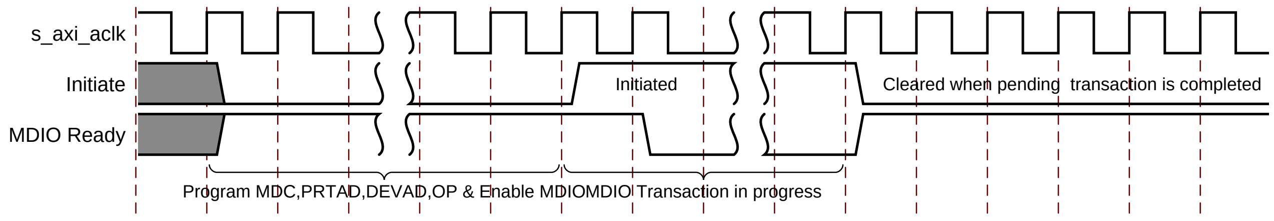

s_axi_aclk
Initiate
MDIO Ready
Initiated
Cleared when pending transaction is completed
Program MDC,PRTAD,DEVAD,OP & Enable MDIO
MDIO Transaction in progress

Figure 3-26: **MDIO Initiate**

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 SelectIO™ interface buffer on in a separate device. Figure 3-27 illustrates the used of a SelectIO interface 3-state buffer as the bus interface.

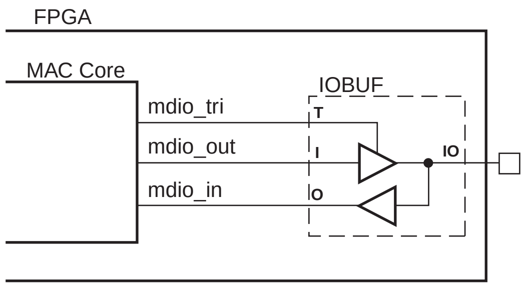


Figure 3-27: **Using a SelectIO Interface 3-State Buffer to Drive MDIO**

MDIO Transaction Types

There are four different transaction types for MDIO, and they are described in the next four sections. In these sections, these abbreviations apply:

- **PRE** – Preamble
- **ST** – Start
- **OP** – Operation code
- **PRTAD** – Port address
- **DEVAD** – Device address
- **TA** – Turnaround

Set Address Transaction

Figure 3-28 shows an Address transaction; this is defined by OP = 00. This is used to set the internal 16-bit address register of the PHY device for subsequent data transactions. This is called the “current address” in the following sections.

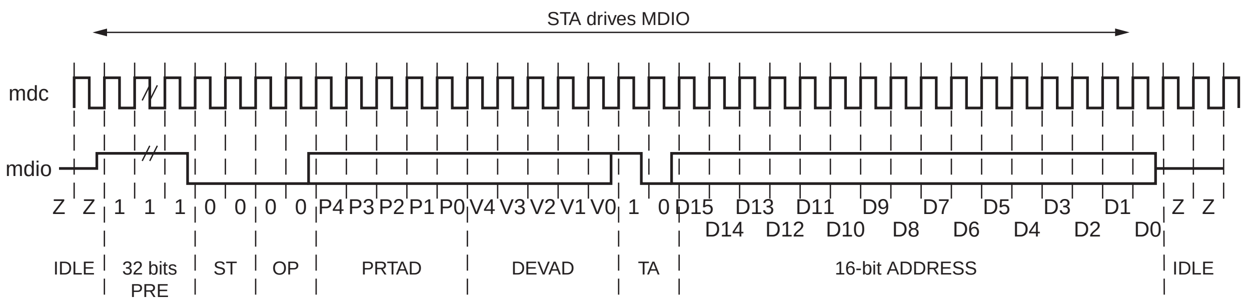

STA drives MDIO
mdc
mdio
Z Z 1 1 1 0 0 0 0 P4 P3 P2 P1 P0 V4 V3 V2 V1 V0 1 0 D15 D13 D11 D9 D7 D5 D3 D1 Z Z
D14 D12 D10 D8 D6 D4 D2 D0
IDLE
32 bits PRE
ST
OP
PRTAD
DEVAD
TA
16-bit ADDRESS
IDLE

Figure 3-28: **MDIO Set Address Transaction**

Write Transaction

Figure 3-29 shows a Write transaction; this is defined by OP = 01. The PHY device takes the 16-bit word in the data field and writes it to the register at the current address.

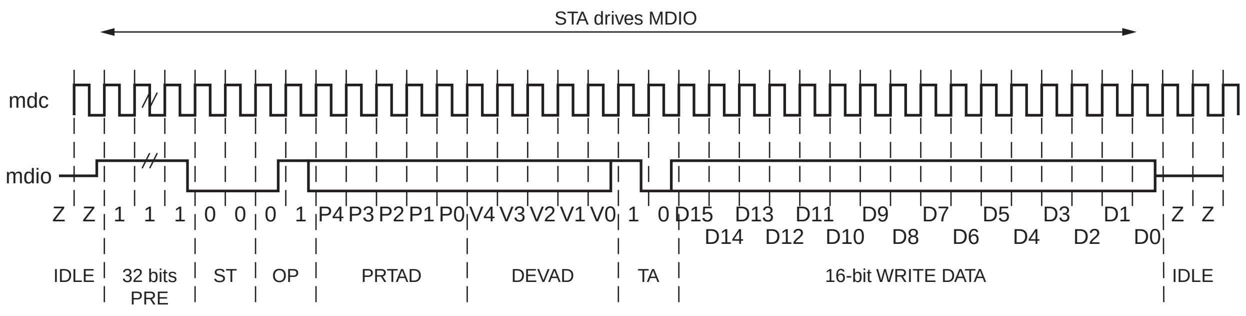

STA drives MDIO
mdc
mdio
Z Z 1 1 1 0 0 0 1 P4 P3 P2 P1 P0 V4 V3 V2 V1 V0 1 0 D15 D13 D11 D9 D7 D5 D3 D1 Z Z
D14 D12 D10 D8 D6 D4 D2 D0
IDLE
32 bits PRE
ST
OP
PRTAD
DEVAD
TA
16-bit WRITE DATA
IDLE

Figure 3-29: **MDIO Write Transaction**

Read Transaction

Figure 3-30 shows a Read transaction; this is defined by OP = 11. The PHY device returns the 16-bit word from the register at the current address.

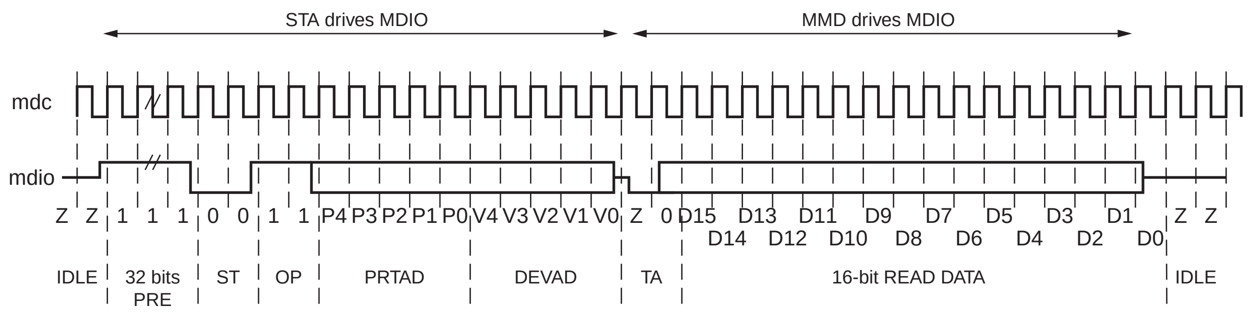

STA drives MDIO
MMD drives MDIO
mdc
mdio
Z Z 1 1 1 0 0 1 1 P4 P3 P2 P1 P0 V4 V3 V2 V1 V0 Z 0 D15 D13 D11 D9 D7 D5 D3 D1 Z Z
D14 D12 D10 D8 D6 D4 D2 D0
IDLE
32 bits PRE
ST
OP
PRTAD
DEVAD
TA
16-bit READ DATA
IDLE

Figure 3-30: **MDIO Read Transaction**

Post-Read-Increment-Address Transaction

Figure 3-31 shows a Post-read-increment-address transaction; this is defined by OP = 10. The PHY device returns the 16-bit word from the register at the current address then increments the current address. This allows sequential reading or writing by a STA master of a block of register addresses.

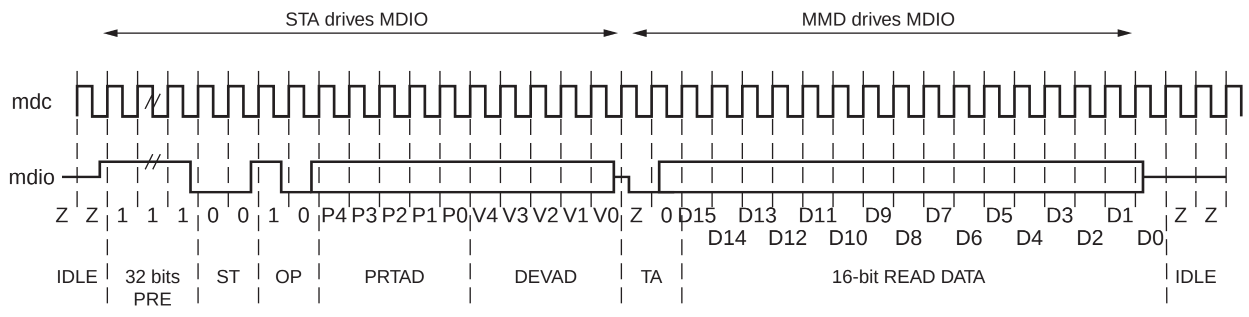

STA drives MDIO
MMD drives MDIO
mdc
mdio
Z Z 1 1 1 0 0 1 0 P4 P3 P2 P1 P0 V4 V3 V2 V1 V0 Z 0 D15 D13 D11 D9 D7 D5 D3 D1 Z Z
D14 D12 D10 D8 D6 D4 D2 D0
IDLE
32 bits PRE
ST
OP
PRTAD
DEVAD
TA
16-bit READ DATA
IDLE

Figure 3-31: **MDIO Read-and-Increment Transaction**

For details of the register map of PHY layer devices and a fuller description of the operation of the MDIO Interface itself, see *IEEE Standard 802.3-2008* [Ref 1].

Using the AXI4-Lite Interface to Access PHY Registers over MDIO

The AXI4-Lite interface is used to access the MDIO ports on the core and access PHY registers either in devices external to the FPGA, or, in the case of XAUI, RXAUI and 10-Gigabit Ethernet PCS/PMA, PHY registers in an associated soft core on the same FPGA. Because the MDIO interface is a relatively slow two-wire interface, MDIO accesses can take many AXI4-Lite cycles to complete.

Prior to any MDIO accesses taking place, the MDIO Configuration Word 0 register must be written to with a valid Clock Divide value and the MDIO Enable bit set.

The target for PHY register accesses is set by the value of the PRTAD and DEVAD fields in the MDIO Configuration Word 1 register. Each port should have a unique 5-bit port address set on each PHY on that port (internal or external).

To write to a PHY register, first the register address must be set, then a second transaction performed to write the value from that address. This is done by setting the target port and device addresses in MDIO Configuration Word 1, setting the target register address in the MDIO TX Data register, setting the TX OP field of MDIO Configuration Word 1 to ADDRESS and starting the transaction; then setting the MDIO TX Data register to the data to be written, the TX OP field to WRITE and starting a follow-up transaction.

To read from a PHY register, first the register address must be set, then a second transaction performed to read the value from that address. This is done by setting the target port and device addresses in MDIO Configuration Word 1, setting the target register address in the MDIO TX Data register, setting the TX OP field of MDIO Configuration Word 1 to ADDRESS and starting the transaction; then setting the TX OP field to READ and starting a follow-up transaction, and reading the result from the MDIO RX Data register.

If successive registers in the same PHY address space are to be read, a special read mode of the protocol can be used. First, the read address should be set as above, but for the first read operation, the Post-Read-Increment-Address opcode should be written into the relevant field of the MDIO Configuration Word1. This returns the read value as above, and also has the side effect of moving the read address to the next register value in the PHY. Thus, repeating the same opcode sequentially returns data from consecutive register addresses in the PHY. Table 3-7 provides an example of a PHY register write using MDIO, to a XAUI configured as a DTE XS on port 0.

Table 3-7: **Example of a PHY Register Write Using MDIO**

Register	Access	Value	Activity
MDIO TX Data	Write	0x00000019	Address of XAUI test control register.
MDIO Configuration Word 1	Write	0x00050800	Initiate the Address transaction by setting the DEVAD (5), PRTAD (0), OP(00) and Initiate bit.
MDIO Configuration Word 1	Read	0x00050080	Poll bit 7 (MDIO Ready) until it becomes 1. The Initiate bit returns to 0.
MDIO TX Data	Write	0x00000006	Turn on transmit test pattern, mixed frequency.
MDIO Configuration Word 1	Write	0x00054800	Initiate the Write transaction by setting the DEVAD (5), PRTAD (0), OP(01) and Initiate bit.
MDIO Configuration Word 1	Read	0x00054080	Initiate the Write transaction by setting the DEVAD (5), PRTAD (0), OP(01) and Initiate bit.

Table 3-8 provides an example of a PHY register read using MDIO, to a PCS on port 7.

Table 3-8: **Example of a PHY Register Read Using MDIO**

Register	Access	Value	Activity
MDIO TX Data	Write	0x00000001	Address of PCS status 1 register.
MDIO Configuration Word 1	Write	0x07030800	Initiate the Address transaction by setting the DEVAD (3), PRTAD (7), OP(00) and Initiate bit.

Table 3-8: **Example of a PHY Register Read Using MDIO *(Cont'd)***

Register	Access	Value	Activity
MDIO Configuration Word 1	Read	0x07030080	Poll bit 7 (MDIO Ready) until it becomes 1. The Initiate bit returns to 0.
MDIO Configuration Word 1	Write	0x0703C800	Initiate the Read transaction by setting the DEVAD (5), PRTAD (0), OP(11) and Initiate bit.
MDIO Configuration Word 1	Read	0x0703C080	Poll bit 7 (MDIO Ready) until it becomes 1. The Initiate bit returns to 0.
MDIO RX Data	Read	0x00000006	Read the status value back from the RX data register.

Interrupt Output

The 10-Gigabit Ethernet MAC core can assert an interrupt when a pending MDIO transaction is completed. If enabled through the Interrupt Enable Register, on the rising edge of `xgmacint`, the MDIO transaction is complete. Furthermore, if the transaction was an MDIO read, the MDIO RX Data results are in the management register `0x50C`. An Interrupt Acknowledge must be issued to clear the interrupt before a new MDIO transaction can be started because the interrupt does not self clear. Table 3-9 lists the Interrupt registers.

Table 3-9: **Interrupt Registers**

Address (Hex)	Default Value	Description
0x600	0x00	**Interrupt Status Register.** Indicates the status of an interrupt. Any asserted interrupt can be cleared by directly writing a 0 to the concerned bit location.
0x610	0x00	**Interrupt Pending Register.** Indicates the pending status of an interrupt. Writing a 1 to any bit of this register clears that particular interrupt. Bits in this register are set only when the corresponding bits in IER & ISR are set.
0x620	0x00	**Interrupt Enable Register.** Indicates the enable state of an interrupt. Writing a 1 to any bit enables that particular interrupt.
0x630	0x00	**Interrupt Acknowledge Register. (Write only)** Writing a 1 to any bit of this register clears that particular interrupt.

Bit[0] of all the interrupt registers is used to indicate that the MDIO transaction has completed. Bits[31:1] are reserved.

Configuration and Status Vector

If the optional Management interface is omitted from the core, all of relevant configuration and status signals are brought out of the core. These signals are bundled into the `configuration_vector` and `status_vector` signals. The bit mapping of the signals are defined in Table 3-10 and Table 3-11. See the corresponding entry in the configuration register tables for the full description of each signal.

You can change the configuration vector signals at any time; however, with the exception of the reset signals and the flow control configuration signals, they do not take effect until the current frame has completed transmission or reception. It is recommended that the configuration vector input signals are driven synchronous from the appropriate clock domain, as detailed in Table 3-10 and Table 3-11.

Table 3-10: **tx_configuration_vector Bit Definitions**

Bits	Description
79:32	**Transmitter Pause Frame Source Address[47:0].** This address is used by the Ethernet MAC core as the source address for any outbound flow control frames. This address does not have any effect on frames passing through the main transmit datapath of the Ethernet MAC. The address is ordered such that the first byte transmitted or received is the least significant byte in the register; for example, a MAC address of AA-BB-CC-DD-EE-FF is stored in byte [79:32] as 0xFFEEDDCCBBAA.
31	Reserved
30:16	**TX MTU Size**. This value is used as the maximum frame size allowed as described in Receiver Maximum Permitted Frame Length, page 52 when RX MTU Enable is set to 1.
15	Reserved
14	**TX MTU Enable**. When this bit is set to 1, the value in TX MTU Size is used as the maximum frame size allowed as described in Transmitter Maximum Permitted Frame Length, page 43. When set to 0 frame handling depends on the other configuration settings.
13:11	Reserved
10	**Deficit Idle Count Enable**. When this bit is set to 1, the core reduces the IFG as described in *IEEE Standard 802.3-2008* [Ref 1], 46.3.1.4 Option 2 to support the maximum data transfer rate. When this bit is set to 0, the core always stretches the IFG to maintain start alignment. This bit is cleared and has no effect if LAN Mode and In-band FCS are both enabled or if Interframe Gap Adjust is enabled.
9	**Transmitter LAN/WAN Mode.** When this bit is 1, the transmitter automatically inserts idles into the Inter Frame Gap to reduce the average data rate to that of the OC-192 SONET payload rate (WAN mode). When this bit is 0, the transmitter uses standard Ethernet interframe gaps (LAN mode).
8	**Transmitter Interframe Gap Adjust Enable**. When this bit is 1, the transmitter reads the value of the tx_ifg_delay port and set the interframe gap accordingly. If it is set to 0, the transmitter inserts a minimum interframe gap. This bit is ignored if Bit[53] (Transmitter LAN/WAN Mode) is set to 1.
7	**Transmitter Preserve Preamble Enable**. When this bit is set to 1, the Ethernet MAC transmitter preserves the custom preamble field presented on the Client Interface. When it is 0, the standard preamble field specified in *IEEE Standard 802.3-2008* is transmitted.
6	Reserved
5	**Transmit Flow Control Enable**. When this bit is 1, asserting the pause_req signal causes the Ethernet MAC core to send a flow control frame out from the transmitter as described in Transmitting a Pause Frame, page 53. When this bit is 0, asserting the pause_req signal has no effect.
4	**Transmitter Jumbo Frame Enable**. When this bit is 1, the Ethernet MAC transmitter allows frames larger than the maximum legal frame length specified in *IEEE Standard 802.3-2008* [Ref 1] to be sent. When set to 0, the Ethernet MAC transmitter only allows frames up to the legal maximum to be sent.

Table 3-10: **tx_configuration_vector Bit Definitions *(Cont'd)***

Bits	Description
3	**Transmitter In-Band FCS Enable**. When this bit is 1, the Ethernet MAC transmitter expects the FCS field to be pass in by the client as described in Transmission with In-Band FCS Passing, page 39. When it is 0, the Ethernet MAC transmitter appends padding as required, compute the FCS and append it to the frame.
2	**Transmitter VLAN Enable**. When this bit is set to 1, the transmitter allows the transmission of VLAN tagged frames.
1	**Transmitter Enable**. When this bit is set to 1, the transmitter is operational. When set to 0, the transmitter is disabled.
0	**Transmitter Reset**. When this bit is 1, the Ethernet MAC transmitter is held in reset. This signal is an input to the reset circuit for the transmitter block. See Resets, page 31 special for details.

Table 3-11: **rx_configuration_vector Bit Definitions**

Bits	Description
79:32	**Receiver Pause Frame Source Address[47:0].** This address is used by the Ethernet MAC core to match against the Destination address of any incoming flow control frames. This address does not have any effect on frames passing through the main receive datapath of the Ethernet MAC. The address is ordered such that the first byte transmitted or received is the least significant byte in the register; for example, a MAC address of AA-BB-CC-DD-EE-FF is stored in byte [47:0] as 0xFFEEDDCCBBAA.
31	Reserved
30:16	**RX MTU Size**. This value is used as the maximum frame size allowed as described in Receiver Maximum Permitted Frame Length, page 52 when RX MTU Enable is set to 1.
15	Reserved
14	**RX MTU Enable**. When this bit is set to 1, the value in RX MTU Size is used as the maximum frame size allowed as described in Receiver Maximum Permitted Frame Length, page 52. When set to 0 frame handling depends on the other configuration settings.
13:11	Reserved
10	**Reconciliation Sublayer Fault Inhibit**. When this bit is 0, the reconciliation sublayer transmits ordered sets as laid out in *IEEE Standard 802.3-2008* [Ref 1]; that is, when the RS is receiving local fault ordered sets, it transmits Remote Fault ordered sets. When it is receiving Remote Fault ordered sets, it transmits idle code words. When this bit is 1, the Reconciliation Sublayer always transmits the data presented to it by the Ethernet MAC, regardless of whether fault ordered sets are being received.
9	**Control Frame Length Check Disable**. When this bit is set to 1, the core does not mark control frames as 'bad' if they are greater than the minimum frame length.
8	**Receiver Length/Type Error Disable**. When this bit is set to 1, the core does not perform the length/type field error check as described in Length/Type Field Error Checks, page 52. When this bit is 0, the length/type field checks are performed; this is normal operation.
7	**Receiver Preserve Preamble Enable**. When this bit is set to 1, the Ethernet MAC receiver preserves the preamble field on the received frame. When it is 0, the preamble field is discarded as specified in *IEEE Standard 802.3-2008*.
6	Reserved

Table 3-11: **rx_configuration_vector Bit Definitions *(Cont'd)***

Bits	Description
5	**Receive Flow Control Enable**. When this bit is 1, received flow control frames inhibit the transmitter operation as described in Receiving a Pause Frame, page 53. When it is 0, received flow frames are passed up to the client.
4	**Receiver Jumbo Frame Enable**. When this bit is 0, the receiver does not pass frames longer than the maximum legal frame size specified in *IEEE Standard 802.3-2008* [Ref 1]. When it is 1, the receiver does not have an upper limit on frame size.
3	**Receiver In-Band FCS Enable**. When this bit is 1, the Ethernet MAC receiver passes the FCS field up to the client as described in Reception with In-Band FCS Passing, page 49. When it is 0, the Ethernet MAC receiver does not pass the FCS field. In both cases, the FCS field is verified on the frame.
2	**Receiver VLAN Enable**. When this bit is set to 1, the receiver allows the reception of VLAN tagged frames.
1	**Receiver Enable**. When this bit is set to 1, the receiver is operational. When set to 0, the receiver is disabled.
0	**Receiver Reset**. When this bit is 1, the Ethernet MAC receiver is held in reset. This signal is an input to the reset circuit for the receiver block. See Resets, page 31 special for details.

Table 3-12: **status_vector Bit Definitions**

Bits	Description
1	**Remote Fault Received**. If this bit is 1, the RS layer is receiving remote fault sequence ordered sets. Read-only.
0	**Local Fault Received**. If this bit is 1, the RS layer is receiving local fault sequence ordered sets. Read-only.

Statistics Vectors

Transmit

The statistics for the frame transmitted are contained within the `tx_statistics_vector`. The vector is synchronous to the transmitter clock, `tx_clk0` and is driven following frame transmission. The bit field definition for the vector is defined in Table 3-13. All bit fields, with the exception of `byte_valid`, are valid only when the `tx_statistics_valid` is asserted. This is illustrated in Figure 3-32. `byte_valid` is significant on every `tx_clk0` cycle.

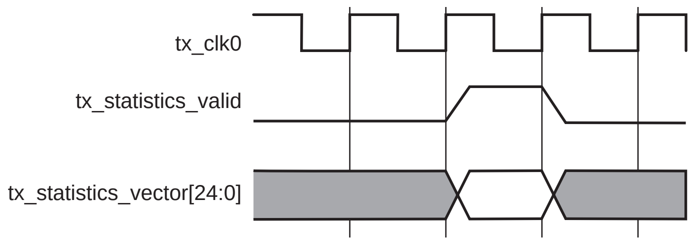


Figure 3-32: **Transmitter Statistics Output Timing**

Table 3-13: **Transmit Statistics Vector Bit Description**

Bits	Name	Description
25	pause_frame_transmitted	Asserted if the previous frame was a pause frame that was initiated by the Ethernet MAC in response to a pause_req assertion.
24:21	bytes_valid	The number of MAC frame bytes transmitted on the last clock cycle (DA to FCS inclusive). This can be between 0 and 8. This is valid on every clock cycle, it is not validated by tx_statistics_valid. The information for the bytes_valid field is sampled at a different point in the transmitter pipeline than the rest of the tx_statistics_vector bits.
20	vlan_frame	Asserted if the previous frame contained a VLAN identifier in the length/ type field and transmitter VLAN operation is enabled.
19:5	frame_length_count	The length of the previously transmitted frame in bytes. The count stays at 32767 for any jumbo frames larger than this value.
4	control_frame	Asserted if the previous frame had the special MAC Control Type code 88-08 in the length/type field.
3	underrun_frame	Asserted if the previous frame transmission was terminated due to an underrun error.
2	multicast_frame	Asserted if the previous frame contained a multicast address in the destination address field.
1	broadcast_frame	Asserted if the previous frame contained the broadcast address in the destination address field.
0	successful_frame	Asserted if the previous frame was transmitted without error.

Receive

The statistics for the frame received are contained within the `rx_statistics_vector`. The vector is driven synchronously by the receiver clock, `rx_clk0`, following frame reception. The bit field definition for the vector is defined in Table 3-14.

All bit fields, with the exception of `bytes_valid`, are valid only when `rx_statistics_valid` is asserted. This is illustrated in Figure 3-33. `bytes_valid` is significant on every `rx_clk0` cycle.

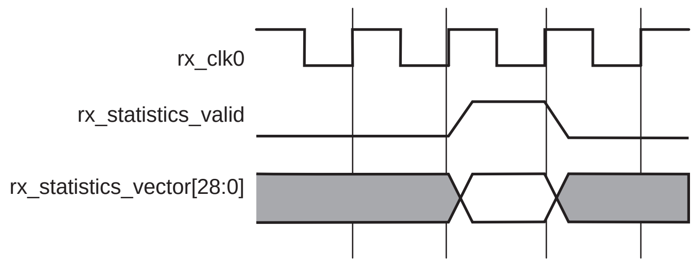


Figure 3-33: **Receiver Statistics Output Timing**

Table 3-14: **Receive Statistics Vector Description**

Bits	Name	Description
29	Length/Type Out of Range	Asserted if the length/type field contained a length value that did not match the number of MAC client data bytes received. Also High if the length/type field indicated that the frame contained padding but the number of client data bytes received was not equal to 64 bytes (minimum frame size).
28	bad_opcode	Asserted if the previous frame was error free, contained the special Control Frame identifier in the length/type field but contained an opcode that is unsupported by the Ethernet MAC (any opcode other than Pause).
27	flow_control_frame	Asserted if the previous frame was error free, contained the Control Frame type identifier 88-08 in the length/type field, contained a destination address that matched either the MAC Control multicast address or the configured source address of the Ethernet MAC, contained the Pause opcode and was acted on by the Ethernet MAC.
26:23	bytes_valid	The number of MAC frame bytes received on the last clock cycle (DA to FCS inclusive). This can be between 0 and 8. This is valid on every clock cycle, it is not validated by `rx_statistics_valid`. The information for the bytes_valid field is sampled at a different point in the transmitter pipeline than the rest of the `rx_statistics_vector` bits.
22	vlan_frame	Asserted if the previous frame contained a VLAN tag in the length/type field and VLAN operation was enabled in the receiver.
21	out_of_bounds	Asserted if the previous frame exceeded the maximum legal frame length specified in *IEEE Standard 802.3-2008* [Ref 1]. This is only asserted if jumbo frames are disabled.
20	control_frame	Asserted if the previous frame contained the MAC Control Frame identifier 88-08 in the length/type field.
19:5	frame_length_count	The length in bytes of the previous received frame. The count stays at 32767 for any Jumbo frames larger than this value. If jumbo frames are disabled, the count stays at 1518 for non-VLAN frames and 1522 for VLAN frames.
4	multicast_frame	Asserted if the previous frame contained a multicast address in the destination address field.
3	broadcast_frame	Asserted if the previous frame contained the broadcast address in the destination address field.
2	fcs_error	Asserted if the previous frame received had an incorrect FCS value or the Ethernet MAC detected error codes during frame reception.
1	bad_frame	Asserted if the previous frame received contained errors.
0	good_frame	Asserted if the previous frame received was error free.

Using Flow Control

This chapter describes the operation of the flow control logic of the core. The flow control block is designed to clause 31 of the *IEEE 802.3-2008* standard. The Ethernet MAC can be configured to transmit pause requests and to act on their reception; these modes of operation can be independently enabled or disabled. See Configuration Registers, page 21.

Overview of Flow Control

Flow Control Requirement

Figure 3-34 illustrates the requirement for Flow Control. The Ethernet MAC at the right side of the figure has a reference clock slightly faster than the nominal 156.25 MHz, and the Ethernet MAC at the left side of the figure has a reference clock slightly slower than the nominal 156.25 MHz. This results in the Ethernet MAC on the left not being able to match the full line rate of the Ethernet MAC on the right (due to clock tolerances). The left MAC is illustrated as performing a loopback implementation, which results in the FIFO filling up over time. Without Flow Control, this FIFO eventually fills and overflows, resulting in the corruption or loss of Ethernet frames. Flow Control is one solution to this issue.

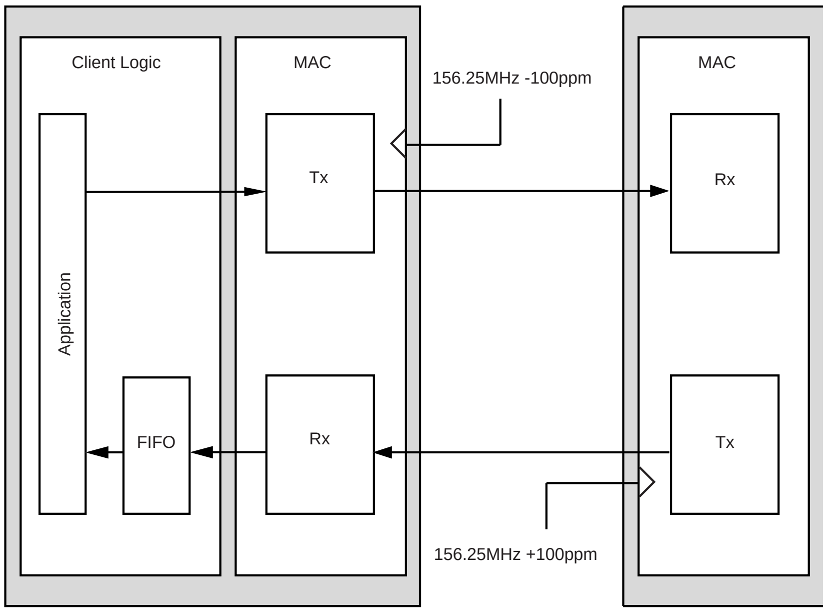


Figure 3-34: **The Requirement for Flow Control**

Flow Control Basics

A MAC might transmit a pause control frame to request that its link partner cease transmission for a defined period of time. For example, the Ethernet MAC at the left side of Figure 3-34 can initiate a pause request when its client FIFO (illustrated) reaches a nearly full state.

A MAC should respond to received pause control frames by ceasing transmission of frames for the period of time defined in the received pause control frame. For example, the Ethernet MAC at the right side of Figure 3-34 might cease transmission after receiving the pause control frame transmitted by the left-hand MAC. In a well designed system, the right MAC would cease transmission before the client FIFO of the left MAC overflowed. This provides time for the FIFO to be emptied to a safe level before normal operation resumes and safeguards the system against FIFO overflow conditions and frame loss.

Pause Control Frames

Control frames are a special type of Ethernet frame defined in clause 31 of the *IEEE 802.3-2008* standard. Control frames are identified from other frame types by a defined value placed into the length/type field (the MAC Control Type code). Control frame format is illustrated in Figure 3-35.

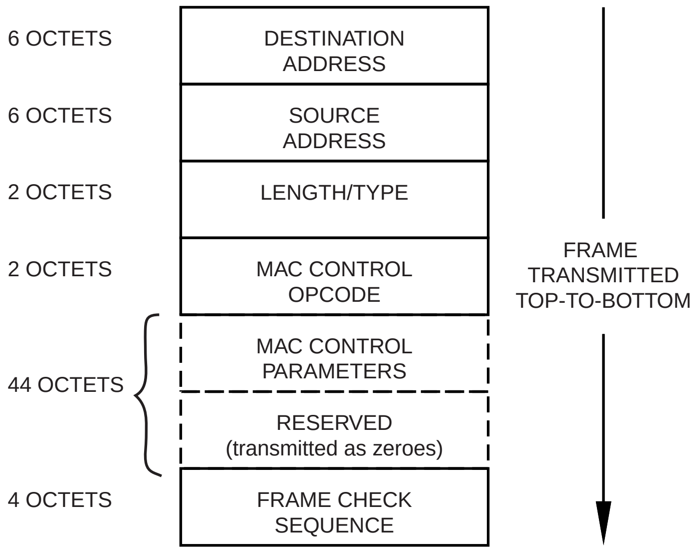


Figure 3-35: **MAC Control Frame Format**

A pause control frame is a special type of control frame, identified by a defined value placed into the MAC Control OPCODE field.

Note: MAC Control OPCODES other than for Pause (Flow Control) frames have recently been defined for Ethernet Passive Optical Networks.

The MAC Control Parameter field of the pause control frame contains a 16-bit field which contains a binary value directly relating to the duration of the pause. This defines the number of *pause_quantum* (512-bit times of the particular implementation). For 10-Gigabit Ethernet, a single *pause_quantum* corresponds to 51.2 ns.

Flow Control Operation of the 10-Gigabit Ethernet MAC Core

Transmitting a Pause Control Frame

Core-Initiated Pause Request

If the Ethernet MAC core is configured to support transmit flow control, the client can initiate a pause control frame by asserting the `pause_req` signal. Figure 3-36 displays this timing.

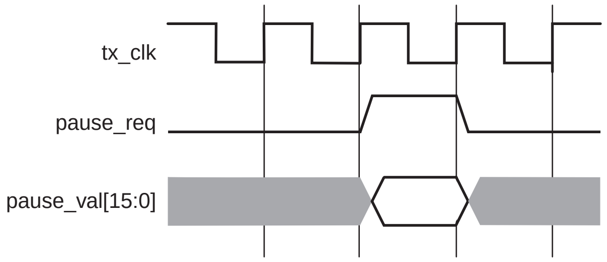


Figure 3-36: **Pause Request Timing**

This action causes the core to construct and transmit a pause control frame on the link with the MAC Control frame parameters (see Figure 3-35):

- The destination address used is an *IEEE 802.3-2008* globally assigned multicast address (which any Flow Control capable MAC responds to).
- The source address used is the configurable Pause Frame MAC Address (see Configuration Registers, page 21).
- The value sampled from the `pause_val[15:0]` port at the time of the `pause_req` assertion is encoded into the MAC control parameter field to select the duration of the pause (in units of *pause_quantum*).

If the transmitter is currently inactive at the time of the pause request, then this pause control frame is transmitted immediately. If the transmitter is currently busy, the current frame being transmitted is allowed to complete; the pause control frame then follows in preference to any pending client supplied frame.

A pause control frame initiated by this method is transmitted even if the transmitter itself has ceased transmission in response to receiving an inbound pause request.

Note: Only a single pause control frame request is stored by the transmitter. If the `pause_req` signal is asserted numerous times in a short time period (before the control pause frame transmission has had a chance to begin), then only a single pause control frame is transmitted. The `pause_val[15:0]` value used is the most recent value sampled.

Client-Initiated Pause Request

For maximum flexibility, flow control logic can be disabled in the core (see Configuration Registers, page 21) and alternatively implemented in the client logic connected to the core. Any type of control frame can be transmitted through the core through the client interface using the same transmission procedure as a standard Ethernet frame (see Normal Frame Transmission, page 38).

Receiving a Pause Control Frame

Core-Initiated Response to a Pause Request

An error-free control frame is a received frame matching the format of Figure 3-35. It must pass all standard receiver frame checks (for example, FCS field checking); in addition, the control frame received must be exactly 64 bytes in length (from destination address through to the FCS field inclusive: this is minimum legal Ethernet MAC frame size and the defined size for control frames).

Any control frame received that does not conform to these checks contains an error, and it is passed to the receiver client with the `rx_bad_frame` signal asserted.

- **Pause Frame Reception Disabled**

  When pause control reception is disabled (see Configuration Registers, page 21), an error free control frame is received through the client interface with the `rx_good_frame` signal asserted. In this way, the frame is passed to the client logic for interpretation (see Transmitting a Pause Frame, page 53).

- **Pause Frame Reception Enabled**

  When pause control reception is enabled (see Configuration Registers) and an error-free frame is received by the Ethernet MAC core, the frame decoding functions are performed:

  a. The destination address field is matched against the *IEEE 802.3-2008* globally assigned multicast address or the configurable Pause Frame MAC Address (see Configuration Registers).

  b. The length/type field is matched against the MAC Control Type code.

  c. The opcode field contents are matched against the Pause opcode.

  If any of these checks are false, the frame is ignored by the flow control logic and passed up to the client logic for interpretation by marking it with `rx_good_frame` asserted. It is then the responsibility of the MAC client logic to decode, act on (if required) and drop this control frame.

  If all these checks are true, the 16-bit binary value in the MAC Control Parameters field of the control frame is then used to inhibit transmitter operation for the required number of *pause_quantum*. This inhibit is implemented by delaying the assertion of `tx_ack` at the transmitter client interface until the requested pause duration has expired. Because the received pause frame has been acted upon, it is passed to the client with `rx_bad_frame` asserted to indicate to the client that can now be dropped.

  Note: Any frame in which the length/type field contains the MAC Control Type should be dropped by the receiver client logic. All control frames are indicated by `rx_statistic_vector` bit 19 (see Receive, page 65).

Client-Initiated Response to a Pause Request

For maximum flexibility, flow control logic can be disabled in the core (see Configuration Registers, page 21) and alternatively implemented in the client logic connected to the core. Any type of error free control frame is then passed through the core with the `rx_good_frame` signal asserted. In this way, the frame is passed to the client for interpretation. It is then the responsibility of the client to drop this control frame and to act on it by ceasing transmission through the core, if applicable.

Flow Control Implementation Example

This explanation is intended to describe a simple (but crude) example of a Flow Control implementation to introduce the concept.

Consider the system illustrated in Figure 3-34. The Ethernet MAC on the left-hand side of the figure cannot match the full line rate of the right-hand Ethernet MAC due to clock tolerances. Over time, the FIFO illustrated fills and overflows. The aim is to implement a Flow Control method which, over a long time period, reduces the full line rate of the right-hand MAC to average that of the lesser full line rate capability of the left-hand MAC.

Method

1. Choose a FIFO nearly full occupancy threshold (7/8 occupancy is used in this description but the choice of threshold is implementation specific). When the occupancy of the FIFO exceeds this occupancy, initiate a single pause control frame with 0xFFFF used as the *pause_quantum* duration (0xFFFF is placed on `pause_val[15:0]`). This is the maximum pause duration. This causes the right-hand MAC to cease transmission and the FIFO of the left-hand MAC starts to empty.
2. Choose a second FIFO occupancy threshold (3/4 is used in this description but the choice of threshold is implementation specific). When the occupancy of the FIFO falls below this occupancy, initiate a second pause control frame with 0x0000 used as the *pause_quantum* duration (0x0000 is placed on `pause_val[15:0]`). This indicates a zero pause duration, and upon receiving this pause control frame, the right-hand MAC immediately resumes transmission (it does not wait for the original requested pause duration to expire). This pause control frame can therefore be considered a “pause cancel” command.

Operation

Figure 3-37 illustrates the FIFO occupancy over time.

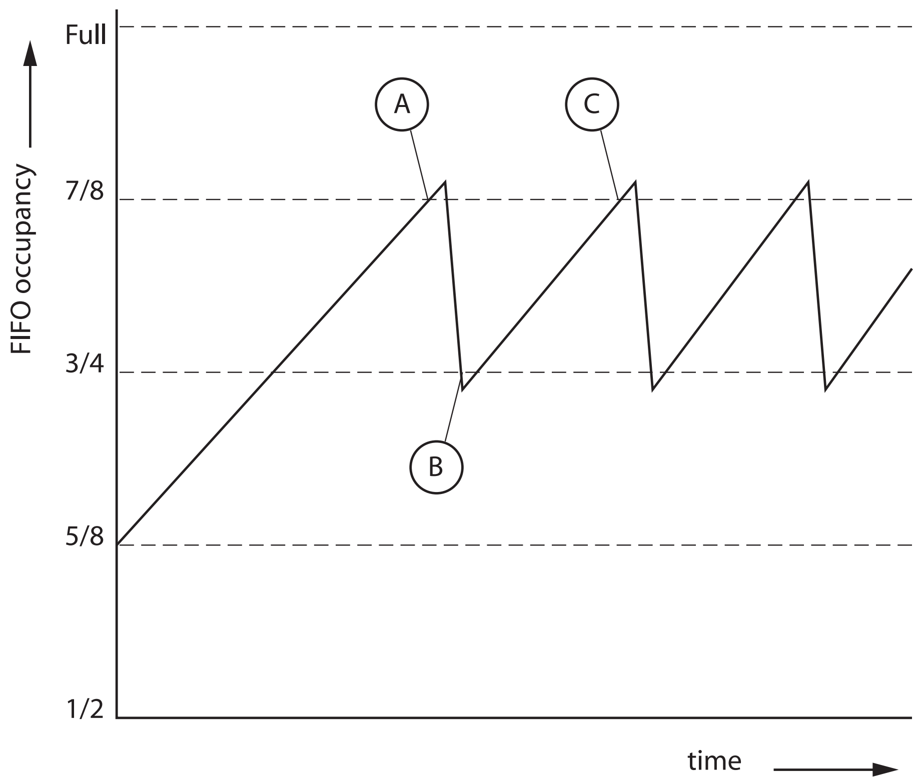


Figure 3-37: **Flow Control Implementation Triggered from FIFO Occupancy**

1. The average FIFO occupancy of the left-hand MAC gradually increases over time due to the clock tolerances. At point A, the occupancy has reached the threshold of 7/8 occupancy. This triggers the maximum duration pause control frame request.
2. Upon receiving the pause control frame, the right-hand MAC ceases transmission.
3. After the right-hand MAC ceases transmission, the occupancy of the FIFO attached to the left-hand MAC rapidly empties. The occupancy falls to the second threshold of 3/4 occupancy at point B. This triggers the zero duration pause control frame request (the pause cancel command).
4. Upon receiving this second pause control frame, the right-hand MAC resumes transmission.
5. Normal operation resumes and the FIFO occupancy again gradually increases over time. At point C, this cycle of Flow Control repeats.

Special Design Considerations

This section describes considerations that can apply in particular design cases. It contains these subsections:

- Multiple Core Instances
- Pin Location Considerations for XGMII Interface
- Interfacing to the Xilinx XAUI Core
- Interfacing with the RXAUI Core
- Interfacing to the 10-Gigabit Ethernet PCS/PMA Core
- Behavior of the Evaluation Core in Hardware

Multiple Core Instances

In a large design, it might be necessary or desirable to have more than one instance of the 10-Gigabit Ethernet MAC core on a single FPGA. One possible clock scheme for two instance with XGMII interfaces is shown in Figure 3-38.

The transmit clock `tx_clk0` can be shared among multiple core instances as illustrated, resulting in a common transmitter clock domain across the device.

A common receiver clock domain is not possible; each core derives an independent receiver clock from its XGMII interface as shown.

Although not illustrated, if the optional Management Interface is used, `host_clk` can also be shared between cores. The `host_clk` signal consumes another BUFG global clock buffer resource.

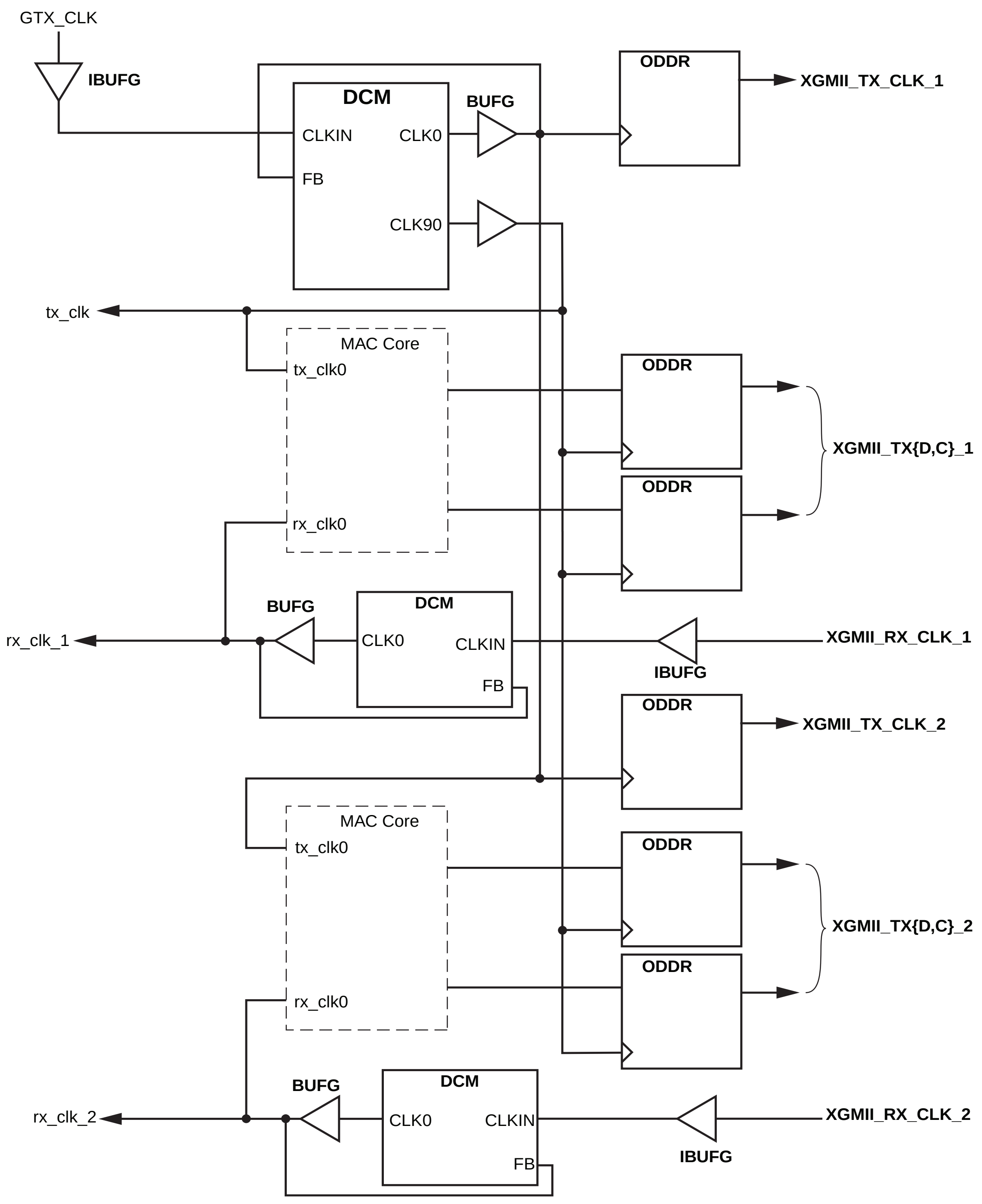


Figure 3-38: **Clock Management, Multiple Instances of the Core with XGMII**

Clock management for multiple cores with the 64-bit SDR interface is similar to that for the XGMII interface.

Pin Location Considerations for XGMII Interface

The Ethernet MAC core allows for a flexible pinout of the XGMII and the exact pin locations are left to the designer. In doing so, codes of practice and device restrictions must be followed.

I/Os should be grouped in their own separate clock domains. XGMII contains two of these:

- `xgmii_rxd[31:0]` and `xgmii_rxc[3:0]`, which are centered with respect to `xgmii_rx_clk`
- `xgmii_txd[31:0]` and `xgmii_txc[3:0]`, which are centered with respect to `xgmii_tx_clk`

Interfacing to the Xilinx XAUI Core

The 10-Gigabit Ethernet MAC core can be integrated with the Xilinx XAUI core in a single device to provide the PHY interface for the Ethernet MAC.

A description of the latest available IP Update containing the XAUI core and instructions on obtaining and installing the IP Update can be found on the Xilinx XAUI core product page at:

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

Other documentation for the XAUI core can also be found at this URL.

Figure 3-41 illustrates the connections and clock management logic required to interface the 10-Gigabit Ethernet MAC core to the XAUI core in Virtex®-6 FPGAs. This shows that:

- Direct connections are made between the PHY-side interface of the 10-Gigabit Ethernet MAC and the client-side interface of the XAUI core.
- If the 10-Gigabit Ethernet MAC core instance has been customized with the Management Interface, then the MDIO port can be connected directly to the XAUI core MDIO port to access the embedded configuration and status registers.
- Both the transmit and receive sides of the XAUI core operate on a single clock domain. This single clock is used as the 156.25 MHz system clock for both cores and the transmitter and receiver logic in the 10-Gigabit Ethernet MAC core now operate in a single unified clock domain.

Note: This final point indicates that some simplification to the UCF for the 10-Gigabit Ethernet MAC core is possible. The constraints that refer to clock-domain crossings from the transmit clock domain to the receive clock domain and vice-versa can be safely removed (although these do not cause harm if left in place).

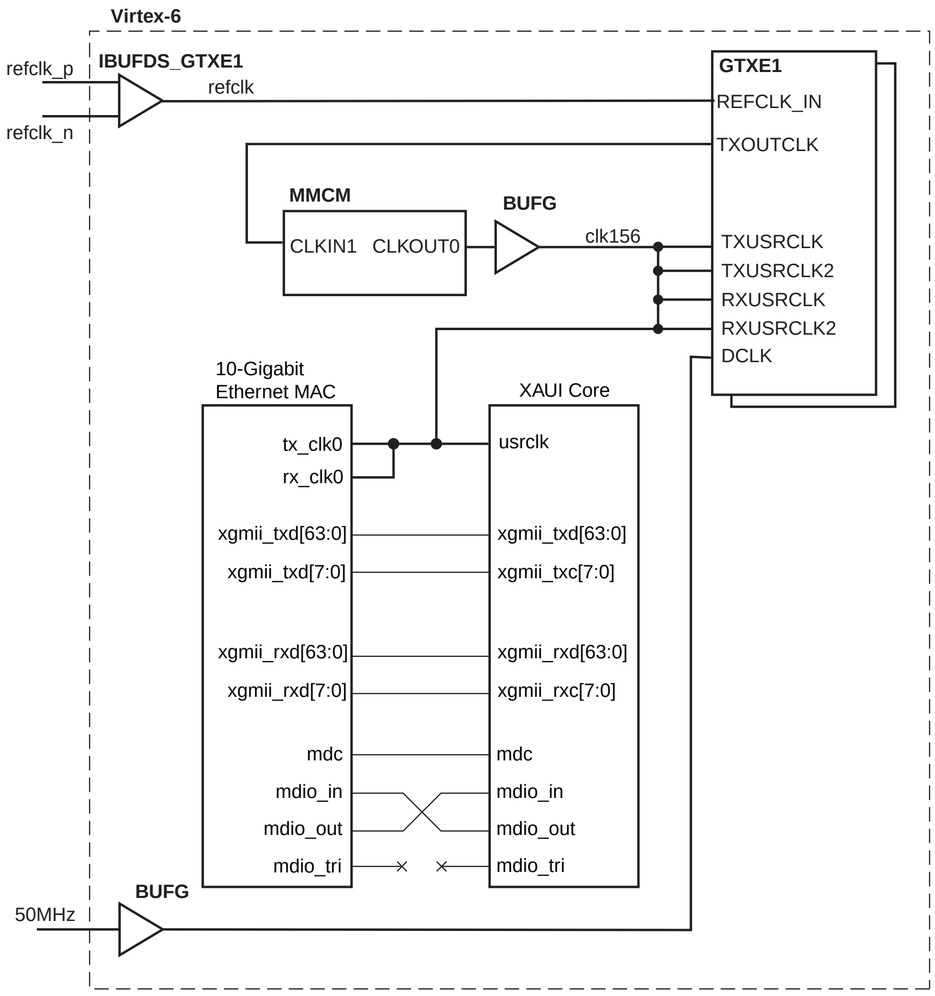


Figure 3-39: **10-Gigabit Ethernet MAC Core Integrated with XAUI Core – Virtex-6 FPGAs**

Figure 3-40 shows the Ethernet MAC core integrated with the XAUI core on Spartan®-6 devices. Figure 3-41 shows the Ethernet MAC core integrated with the XAUI core on Virtex-7, Kintex™-7, Artix™-7, and Zynq™-7000 devices. All of the points made with respect to the Virtex-6 FPGA implementation apply here also.

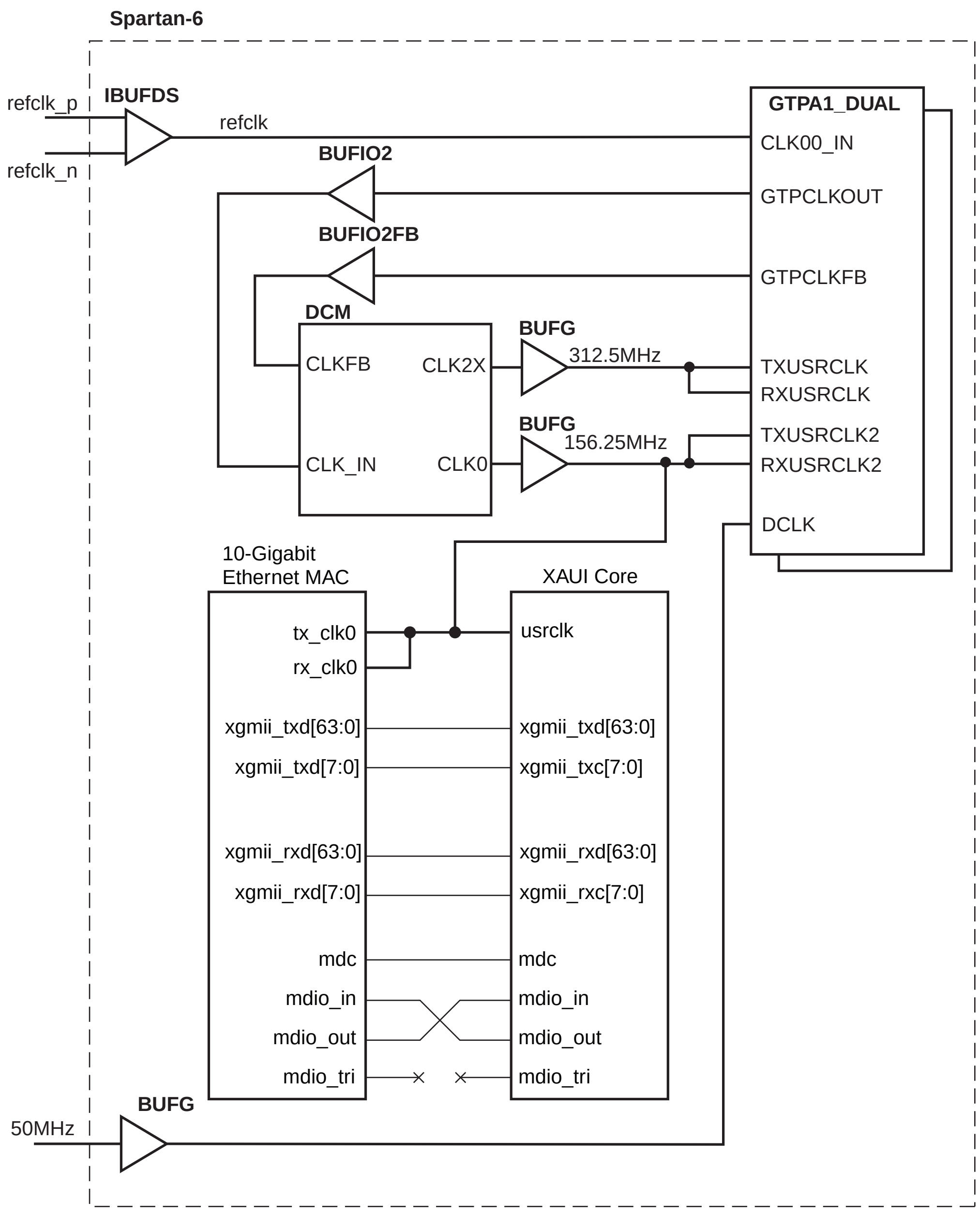


Figure 3-40: **10-Gigabit Ethernet MAC Core Integrated with XAUI Core – Spartan 6 FPGA**

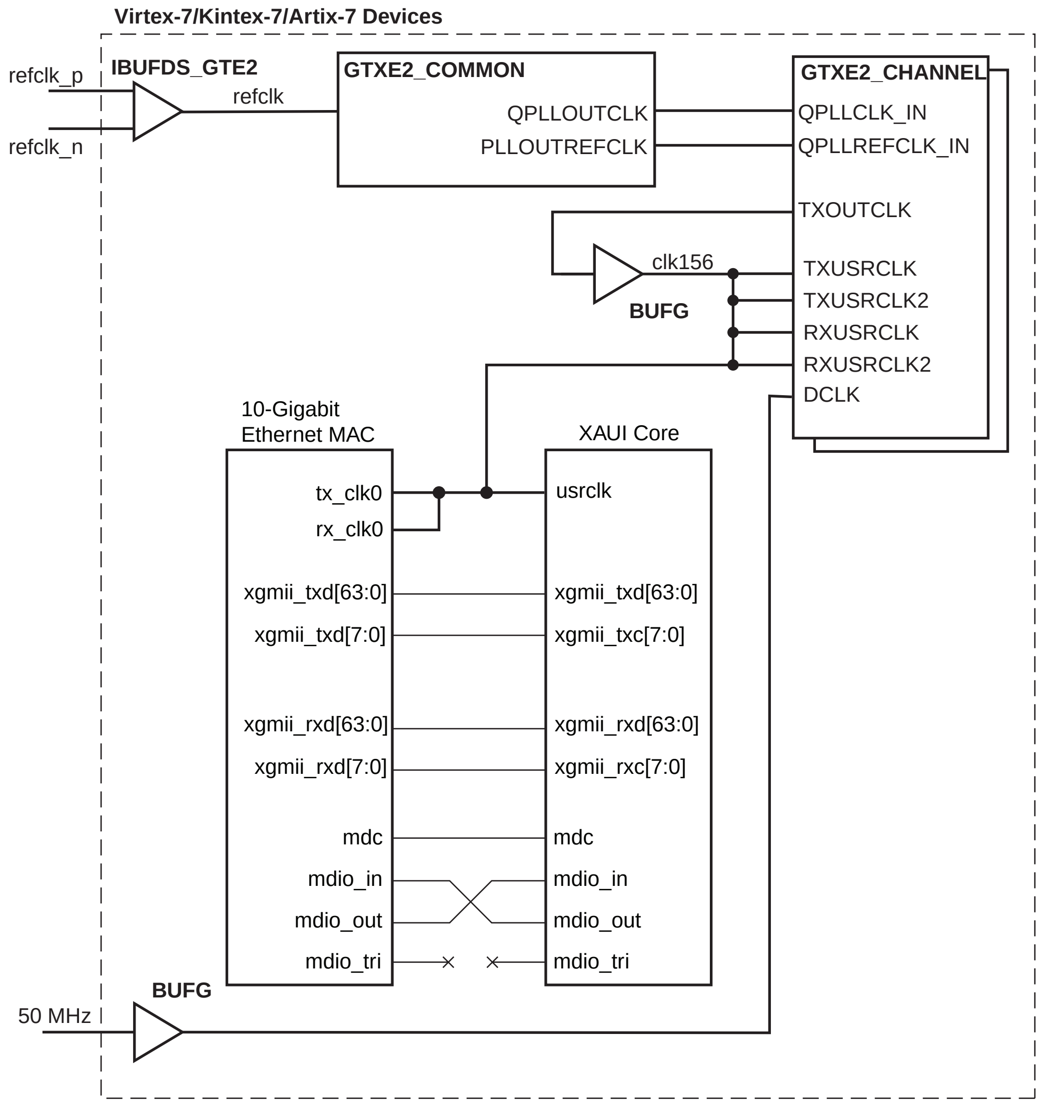


Figure 3-41: **10-Gigabit Ethernet MAC Core Integrated with XAUI Core – Virtex-7, Kintex-7, Artix-7, and Zynq-7000 Devices**

For details on clocks and transceiver placement using the XAUI core, see *LogiCORE IP XAUI Product Guide* [Ref 6].

Interfacing with the RXAUI Core

The 10-Gigabit Ethernet MAC core can be integrated with the Xilinx RXAUI core in a single device to provide the PHY interface for the Ethernet MAC.

A description of the latest available IP Update containing the RXAUI core and instructions on obtaining and installing the IP Update can be found on the Xilinx RXAUI core product page at:

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

Other documentation for the RXAUI core can also be found at this URL.

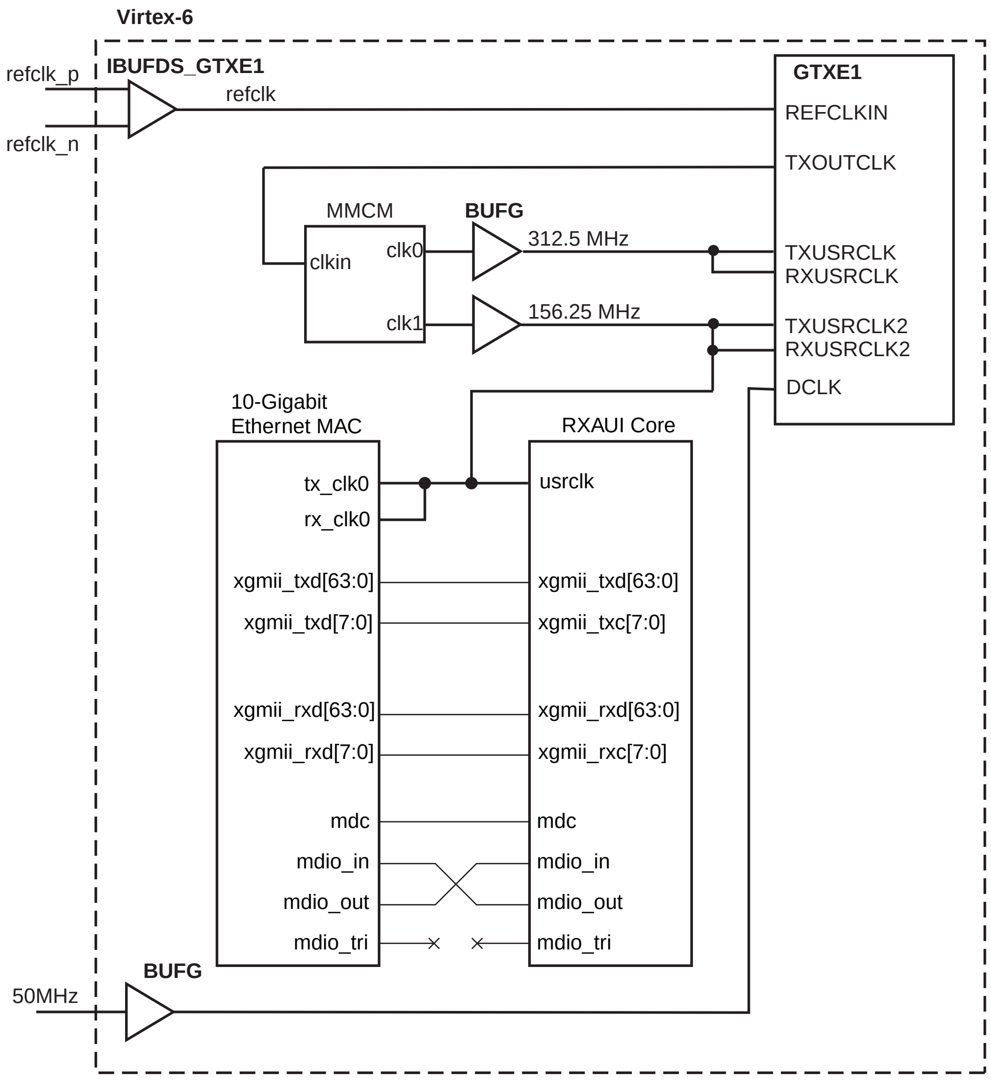


Figure 3-42: **10-Gigabit Ethernet MAC Core Integrated with RXAUI Core – Virtex 6 FPGA**

Figure 3-42 illustrates the connections and clock management logic required to interface the 10-Gigabit Ethernet MAC core to the RXAUI core in Virtex-6 FPGAs. This shows that:

- Direct connections are made between the PHY-side interface of the 10-Gigabit Ethernet MAC and the client-side interface of the RXAUI core.
- If the 10-Gigabit Ethernet MAC core instance has been customized with the Management Interface, then the MDIO port can be connected directly to the RXAUI core MDIO port to access the embedded configuration and status registers.
- Both the transmit and receive client interfaces of the RXAUI core operate on a single clock domain. This single clock is used as the 156.25 MHz system clock for both cores and the transmitter and receiver logic in the 10-Gigabit Ethernet MAC core now operate in a single unified clock domain.

Note: This final point indicates that some simplification to the UCF for the 10-Gigabit Ethernet MAC core is possible. The constraints that refer to clock-domain crossings from the transmit clock domain to the receive clock domain and vice-versa can be safely removed (although these do not cause harm if left).

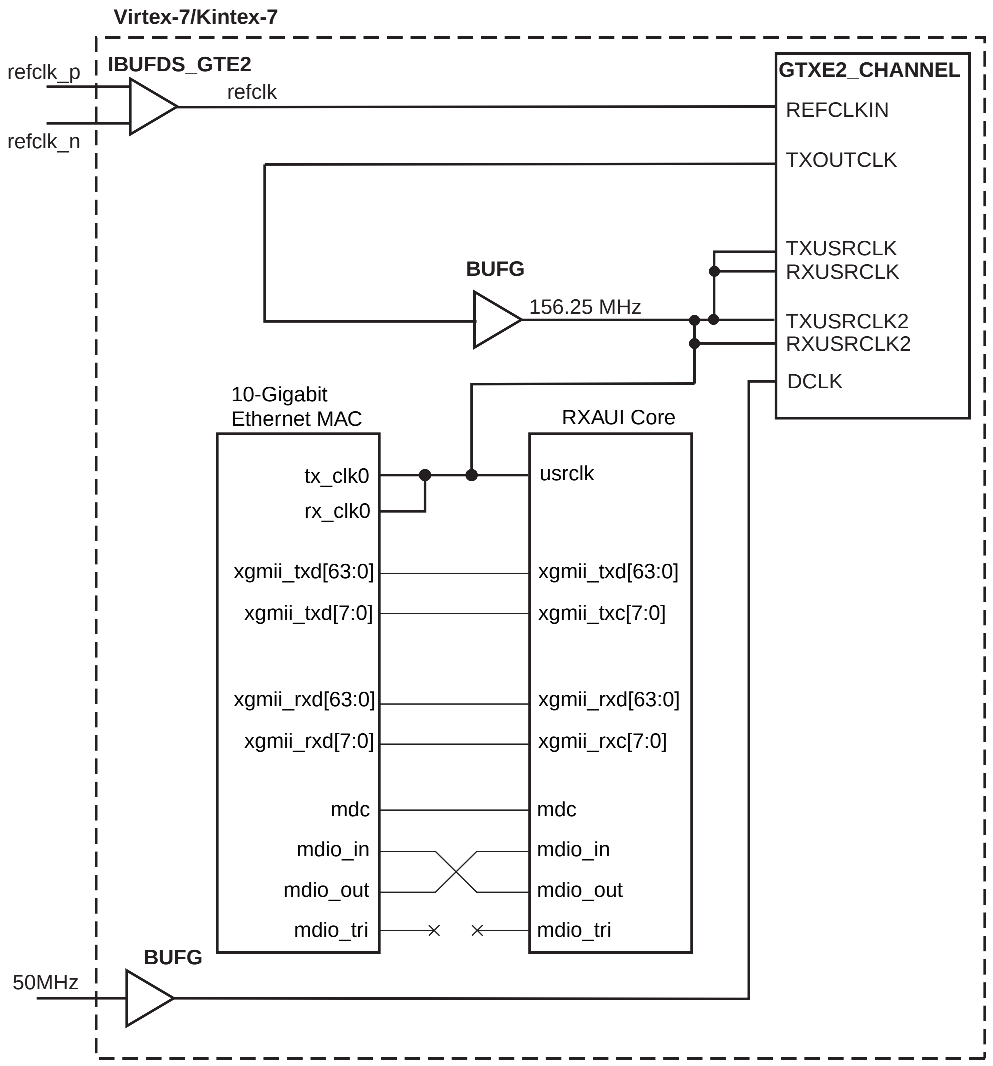


Figure 3-43: **10-Gigabit Ethernet MAC Core Integrated with RXAUI Core – Virtex-7 and Kintex-7 FPGAs**

Figure 3-43 shows the Ethernet MAC core integrated with the RXAUI core on Virtex-7 and Kintex-7 FPGAs. All of the points made with respect to the Virtex-6 FPGA implementation apply here also.

For details on clocks and transceiver placement using the RXAUI core, see *LogiCORE IP RXAUI User Guide* [Ref 7].

Interfacing to the 10-Gigabit Ethernet PCS/PMA Core

The 10-Gigabit Ethernet MAC core can be integrated with the Xilinx 10-Gigabit Ethernet PCS/PMA core in a single device to provide the PHY interface for the Ethernet MAC.

A description of the latest available IP Update containing the 10-Gigabit Ethernet PCS/PMA core and instructions on obtaining and installing the IP Update can be found on the Xilinx 10-Gigabit Ethernet PCS/PMA Product Page at:

www.xilinx.com/products/intellectual-property/10GBASE-R.htm

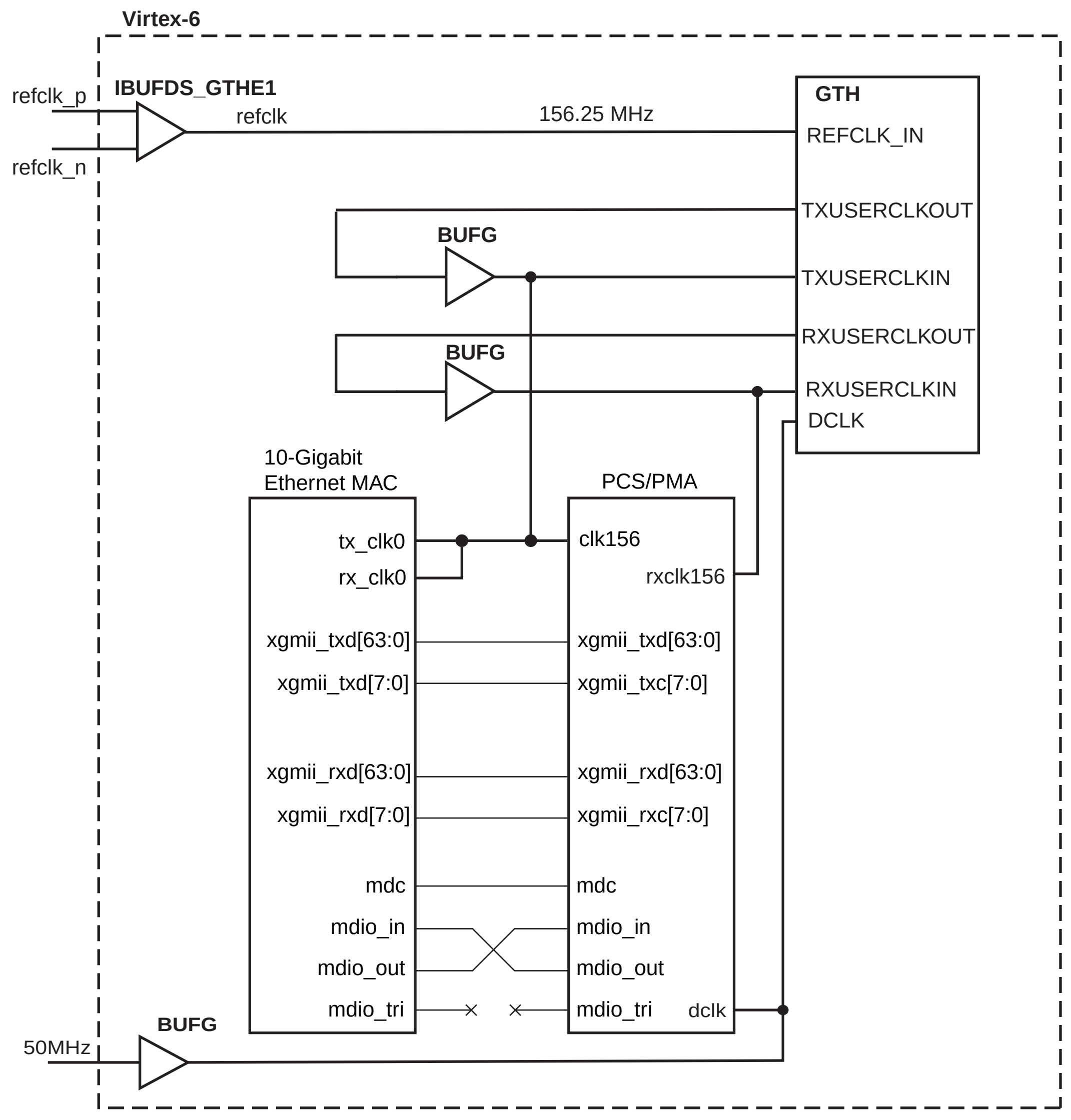


Figure 3-44: **10-Gigabit Ethernet MAC Core Integrated with PCS/PMA Core – Virtex-6 FPGA**

Figure 3-44 illustrates the connections and clock management logic required to interface the 10-Gigabit Ethernet MAC core to the PCS/PMA core in Virtex-6 FPGAs. This shows that:

- Direct connections are made between the PHY-side interface of the 10-Gigabit Ethernet MAC and the client-side interface of the PCS/PMA core.
- If the 10-Gigabit Ethernet MAC core instance has been customized with the Management Interface, then the MDIO port can be connected directly to the PCS/PMA core MDIO port to access the embedded configuration and status registers.
- Both the transmit and receive client interfaces of the PCS/PMA core operate on a single clock domain. This single clock is used as the 156.25 MHz system clock for both cores and the transmitter and receiver logic in the 10-Gigabit Ethernet MAC core now operate in a single unified clock domain.

Note: This final point indicates that some simplification to the UCF/XDC file for the 10-Gigabit Ethernet MAC core is possible. The constraints that refer to clock-domain crossings from the transmit clock domain to the receive clock domain and vice-versa can be safely removed (although these do not cause harm if left).

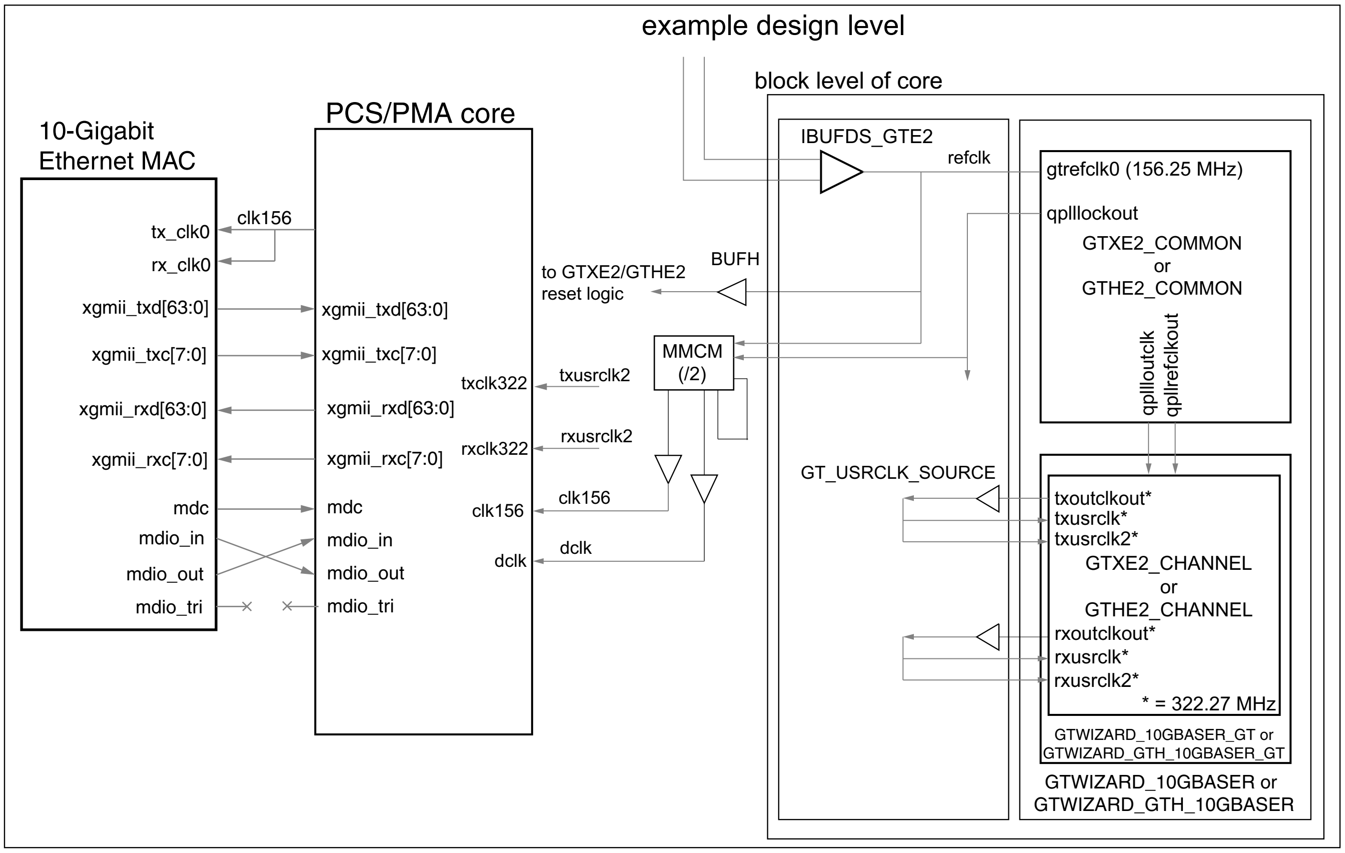


Figure 3-45: **10-Gigabit Ethernet MAC Core Integrated with PCS/PMA Core – Virtex-7 and Kintex-7 FPGAs**

Figure 3-45 shows the Ethernet MAC core integrated with the PCS/PMA core on Virtex-7 and Kintex-7 FPGAs. All of the points made with respect to the Virtex-6 FPGA implementation apply here also.

For details on clocks and transceiver placement using the PCS/PMA core, see *LogiCORE IP 10-Gigabit Ethernet PCS/PMA Product Guide* [Ref 5].

Behavior of the Evaluation Core in Hardware

When the core is generated with a Full System Hardware Evaluation, the core can be tested in the target device for several hours before ceasing to function.

Symptoms of the hardware evaluation timeout include:

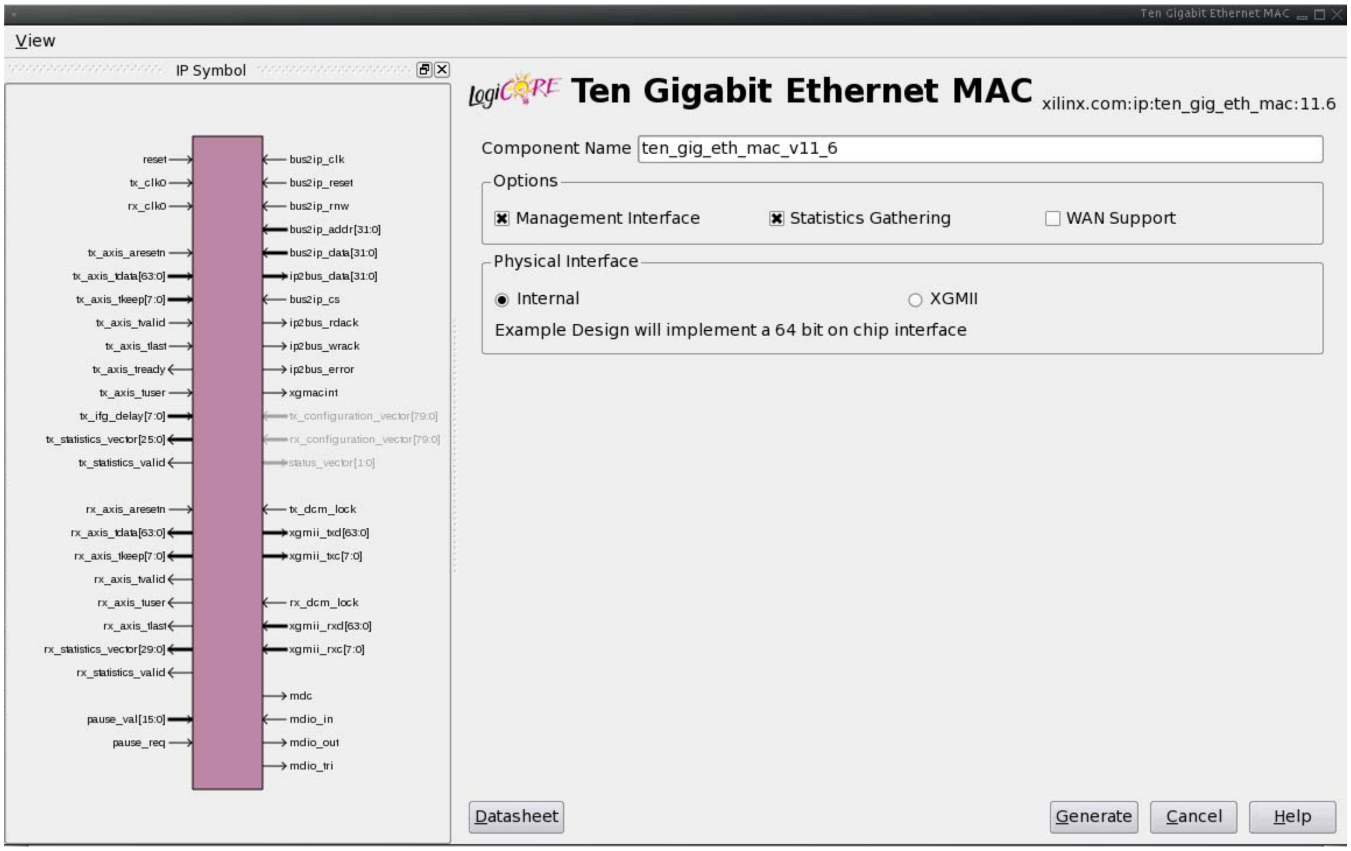
- The transmitter failing to assert TX_AXIS_TREADY in response to TX_AXIS_TVALID.
- The receiver failing to recognize frames in the inbound data stream.
- After the timeout occurs, the core can be reactivated by reconfiguring the FPGA.

Customizing and Generating the Core

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

GUI

Figure 4-1 displays the CORE Generator tool customization screen for the 10-Gigabit Ethernet MAC core.



Ten Gigabit Ethernet MAC xilinx.com:ip:ten_gig_eth_mac:11.6

Component Name: ten_gig_eth_mac_v11_6

Options

[✓ Management Interface] [✓ Statistics Gathering] WAN Support

Physical Interface

[✓ Internal] XGMII

Example Design will implement a 64 bit on chip interface

Figure 4-1: **10-Gigabit Ethernet MAC Customization Screen**

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

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).

Statistics Gathering

This checkbox selects whether the statistics counters are included in the generated core.

This option is only available if the Management Interface option is selected. The default is to have statistics counters included.

Management Interface

Select this option to include the Management Interface in the generated core. Deselect this option to remove the Management Interface and expose a simple bit vector to manage the core.

The default is to have the Management Interface included.

Physical Interface

The physical interface section has a choice of two selections; *XGMII*, which implements the 32-bit DDR interface to the physical layer, and *Internal*, which selects the internal 64-bit SDR interface to the physical layer.

Note: On Spartan®-6 devices, only the internal 64-bit interface is supported.

Parameter Values in the XCO File

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

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

```
SELECT Ten_Gigabit_Ethernet_MAC Virtex6 Xilinx,_Inc. 11.1
CSET physical_interface = XGMII
CSET statistics_gathering = true
CSET component_name = the_core
CSET management_interface = true
GENERATE
```

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

Parameter	XCO File Values	Defaults
component_name	ASCII text starting with a letter and based on the following character set: a..z, 0..9 and _	Blank
physical_interface	XGMII, Internal	Internal
statistics_gathering	True, false	True
management_interface	True, false	True

Output Generation

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

- The netlist files for the core
- XCO files
- Release notes and documentation
- An HDL example design
- Scripts to synthesize, implement and simulate the example design

See Chapter 7, Detailed Example Design for a complete description of the CORE Generator tool output files, for details of the HDL example design, and for the directory and file contents.

Implementing your Design

Synthesis

XST – VHDL

In the CORE Generator™ tool project directory, there is a `*xgmac_component_name*.vho` file that is a component and instantiation template for the core. Use this template to help instance the 10-Gigabit Ethernet MAC core into your VHDL source.

After your entire design is complete, create:

- An XST project file `*top_level_module_name*.prj` listing all the user source code files

- An XST script file *top_level_module_name*.scr containing your required synthesis options.

To synthesize the design, run:

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

See *XST User Guide* [Ref 2] for details on creating project and synthesis script files and running the xst program.

XST – Verilog

In the Xilinx CORE Generator™ project directory, there is a module declaration for the 10-Gigabit Ethernet MAC core at:

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

Use this module to help instance the 10-Gigabit Ethernet MAC core into your Verilog source.

After your entire design is complete, create:

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

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

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

To synthesize the design, run:

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

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

Implementation

For details on the use of the Xilinx implementation tool flow including command line switches and options, consult the software manuals that came with the ISE Design Suite.

Generating the Xilinx Netlist

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

```
$ ngdbuild -sd path_to_xgmac_netlist -sd path_to_user_synth_results \
    -uc top_level_module_name.ucf top_level_module_name
```

Mapping the Design

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

```
$ map -o top_level_module_name_map.ncd top_level_module_name.ngd \
    top_level_module_name.pcf
```

Placing and Routing the Design

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

```
$ par top_level_module_name_map.ncd top_level_module_name.ncd \
    top_level_module_name.pcf
```

Static Timing Analysis

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

```
$ trce -o top_level_module_name.twr top_level_module_name.ncd \
    top_level_module_name.pcf
```

Generating a Bitstream

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

```
$ bitgen -w top_level_module_name.ncd
```

Post-Implementation Simulation

The purpose of post-implementation simulation is to verify that the design as implemented in the FPGA works as expected.

Generating a Simulation Model

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

- VHDL

```
$ netgen -sim -ofmt vhdl -pcf top_level_module_name.pcf \
   -tm netlist top_level_module_name.ncd \
   top_level_module_name_postimp.vhd
```

- Verilog

```
$ netgen -sim -ofmt verilog -pcf top_level_module_name.pcf \
   -tm netlist top_level_module_name.ncd \
   top_level_module_name_postimp.v
```

Using the Model

For information on setting up your simulator to use the pre-implemented model, consult *Xilinx Synthesis and Simulation Design Guide* [Ref 3], also included in your Xilinx software installation.

Constraining the Core

This chapter describes how to constrain a design containing the 10-Gigabit Ethernet MAC core. This is illustrated by the User Constraint File (UCF) delivered with the core at generation time. See Chapter 7, Detailed Example Design for a complete description of the Xilinx® CORE Generator™ tool output files.

Not all constraints are relevant for a particular implementation of the core; consult the UCF created with the core instance to see exactly what constraints are relevant.

Constraints

MDIO Interface

This section constrains the low-speed MDIO logic.

```
# The constraint on the clock period for the MDIO block is half the MDC
# minimum period of 400 ns; the turnaround phase of a read operation leads
# to a half-cycle operation in the middle of the transaction.
# Value below is correct for a 133 MHz bus2ip_clk - 200 ns / 7518 ps = 26.60
NET "*xgmac_core/BU2/U0/G_MANAGEMENT.managen/mdio_master_i/mdc_ce" TNM = "mdc_grp";
TIMESPEC "TSmdc" = FROM "mdc_grp" TO "mdc_grp" TS_s_axi_aclk_clk * 26;
```

I/O Constraints

These constraints set the I/O standards used for the SelectIO™ interface pads and ensure that the DDR registers are placed in the I/O buffer. This section is only used in an XGMII implementation.

```
#######################################################
# I/O constraints                                     #
#######################################################

#  Ensure that XGMII DDR registers are placed in IOBs
#  and utilize the HSTL_I voltage standard.
INST "*txd_ddr*"                  IOB             = "TRUE";
NET  "xgmii_txd<*>"              IOSTANDARD      = "HSTL_I";
INST "*txc_ddr*"                  IOB             = "TRUE";
NET  "xgmii_txc<*>"              IOSTANDARD      = "HSTL_I";
INST "*tx_clk_ddr"                IOB        = "TRUE";
NET  "xgmii_tx_clk"              IOSTANDARD = "HSTL_I";
```

```
#  Ensure that XGMII DDR registers are placed in IOBs
#  and utilize the HSTL_I voltage standard.
NET  "xgmii_rx_clk"              IOSTANDARD  = "HSTL_I";
NET  "xgmii_rx_clk"              IOBDELAY    = "NONE";
NET  "xgmii_rxd<*>"              IOSTANDARD   = "HSTL_I";
NET  "xgmii_rxd<*>"              IOBDELAY     = "NONE";
INST "*xgmii_if/rxd_loop[*].rxd_ddr" IOB            = "TRUE";

#  XGMII RXC Constraints.
NET "xgmii_rxc<*>" IOSTANDARD   = "HSTL_I";
NET "xgmii_rxc<*>" IOBDELAY     = "NONE\";

INST "*xgmii_if/rxc_loop[*].rxc_ddr"  IOB = "TRUE";

#  DDR Setup and Hold
INST "xgmii_rxd<*>" TNM = IN_XGMII;
INST "xgmii_rxc<?>" TNM = IN_XGMII;

# Define setup respect to the clock
TIMEGRP "IN_XGMII" OFFSET = IN  480 ps BEFORE "xgmii_rx_clk" RISING;
TIMEGRP "IN_XGMII" OFFSET = IN  480 ps BEFORE "xgmii_rx_clk" FALLING;
```

Device, Package, and Speed Grade Selections

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

```
# set the part and package
CONFIG PART = xc6vlx240tff1156-1;
```

The 10-Gigabit Ethernet MAC core can be implemented in these devices with the following speed grades:

- Virtex®-6 family devices, speed grade of -1 or faster
- Spartan®-6 family devices, speed grade of -3 or faster (no XGMII interface or WAN mode)

Clock Frequencies, Management, and Placement

The core can have up to three clock domains; the transmit clock domain, derived from the `gtx_clk` signal, the receive clock domain, derived from the `xgmii_rx_clk` signal, and the `host_clk` domain.

```
#######################################################
# Clock/period constraints                            #
#######################################################
# Main transmit clock/period constraints
```

```
NET "gtx_clk" TNM_NET = "xgmactxclk";
TIMESPEC "TS_xgmactx" = PERIOD "xgmactxclk" 6400 ps HIGH 50 %;

# Group the driven storage elements together for use in FROM-TO constraints.
NET "tx_clk0" TNM_NET = "xgmactxgrp";
TIMESPEC "TS_clk0" = PERIOD "xgmactxgrp" 6400 ps HIGH 50 %;
```

This section sets the period of the transmit clock.

```
# Main receive clock/period constraints
NET "xgmii_rx_clk" TNM_NET = "xgmacrxclk";
TIMESPEC "TS_xgmacrx"    = PERIOD "xgmacrxclk" 6400 ps HIGH 50 %;

# Group the driven storage elements together for use in FROM-TO constraints.
NET "fifo_block/xgmac_block/*_if/rx_clk0" TNM_NET = "xgmacrxgrp";
TIMESPEC "TS_rxclk0" = PERIOD "xgmacrxgrp" 6400 ps HIGH 50 %;
```

This section sets the period of the receive clock.

```
# The management clock can run at same speed as RX & TX.
NET "*s_axi_aclk_int" TNM_NET = "xgmacmangementgrp";
TIMESPEC "TS_s_axi_aclk_clk" = PERIOD "xgmacmangementgrp" 6400 ps HIGH 50 %;
```

This sets the period of `s_axi_clk`.

Banking

If an XGMII interface is selected for the physical interface of the core instance, care should be taken to observe the I/O banking requirements of the particular FPGA part targeted in the user design.

Quick Start Example Design

The 10-Gigabit Ethernet MAC example design is shown in Figure 6-1.

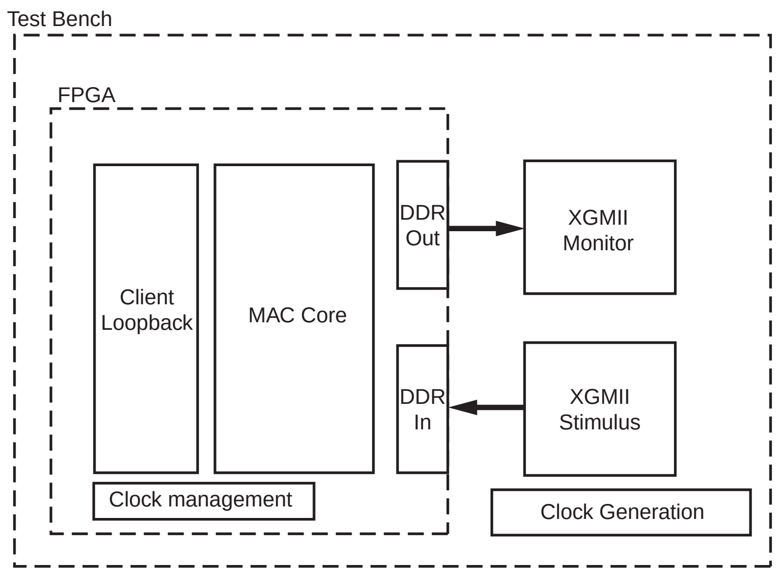


Figure 6-1: **Example Design and Test Bench**

The example design consists of the following:

- The 10-Gigabit Ethernet MAC core netlist
- An example HDL wrapper/top level
- A *ping* client-side loopback circuit including asynchronous FIFO that returns received frames on the transmit port
- A demonstration test bench to exercise the example design
- An AXI4-Lite-to-IPIF bridge for the management interface of the core.

The 10-Gigabit Ethernet MAC example design has been tested with Xilinx ISE® Design Suite v14.5, Mentor Graphics Questa® SIM, Cadence Incisive Enterprise Simulator (IES), and Synopsys VCS and VCS MX. For the supported version of the tools see the Xilinx Design Tools: Release Notes Guide.

Generating the Core

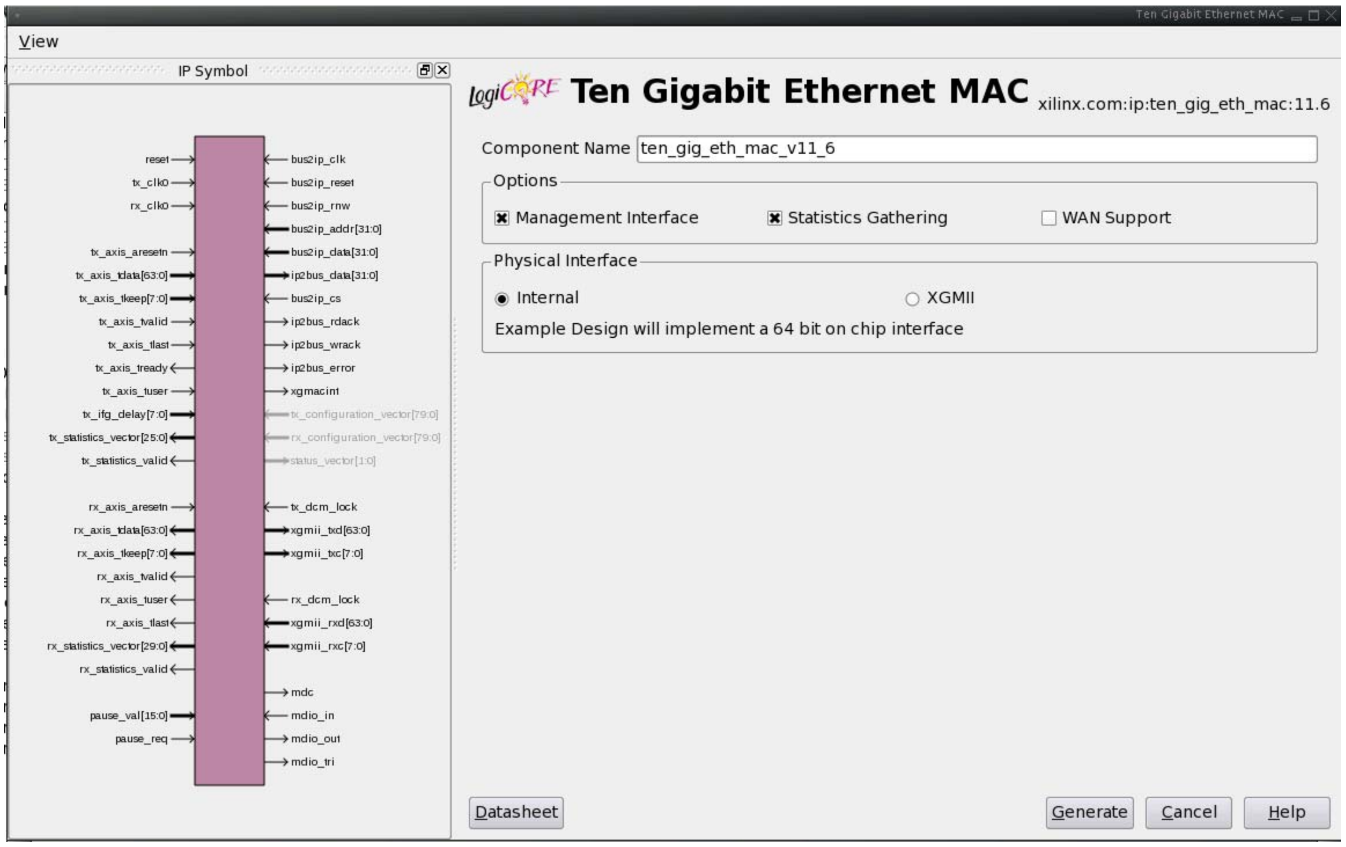
To begin working with the example design, start by generating the core with the default settings.

To generate the core:

1. Start the Xilinx CORE Generator™ tool.

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

2. Create a new project.
3. In the Project Options dialog box, select a silicon family that supports the 10-Gigabit Ethernet MAC core for example, Virtex®-6 FPGAs.
4. In the Generate section of the Project Options, select either VHDL or Verilog, and then select Other for the Vendor.
5. Locate the 10-Gigabit Ethernet MAC core in the taxonomy tree, displayed under Communications & Networking/Ethernet; then double-click the core to open the main GUI screen.
6. A dialog box warning of the limitations of the Simulation Only Evaluation license might appear; click OK to continue. The 10-Gigabit Ethernet MAC customization screen appears.



Ten Gigabit Ethernet MAC xilinx.com:ip:ten_gig_eth_mac:11.6

Component Name: ten_gig_eth_mac_v11_6

Options

[✓ Management Interface] [✓ Statistics Gathering] WAN Support

Physical Interface

[✓ Internal] XGMII

Example Design will implement a 64 bit on chip interface

Figure 6-2: **10-Gigabit Ethernet MAC Customization Screen**

7. In the Component Name field, enter a name for the core instance. For this example, the name *quickstart* is used.
8. Accept the default settings; then click Finish to generate the core.

The core and its supporting files, including the example design, are generated in the project directory. For a detailed description of the design example files and directories, see Chapter 7, Detailed Example Design.

Implementation

After the core is successfully generated, the netlist and example design HDL wrapper can be processed through the Xilinx implementation toolset. Included in the generated outputs are several scripts to assist in this processing.

To implement the example design:

Open a command prompt or shell in your project directory, then enter these commands for either Linux or Windows platforms:

Linux

```
% cd quickstart/implement
% ./implement.sh
```

Windows

```
> cd quickstart\implement
> implement.bat
```

This starts a script that synthesizes the example design HDL wrapper, builds, maps, and places-and-routes the example design, and then creates gate-level netlist HDL files in both VHDL and Verilog with associated timing information (SDF) files. Finally these files are copied into the test area directories. This process occurs only when using the Full System Hardware Evaluation or Full licenses.

Simulation

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

Setting up for Simulation

The Xilinx UNISIM and SIMPRIM libraries must be mapped into the simulator. If the UNISIM and SIMPRIM libraries are not set up for your environment, see *Xilinx Synthesis and Simulation Design Guide* [Ref 3] for help on compiling Xilinx simulation models and setting up the simulator environment.

Pre-implementation Simulation

To run a functional simulation of the example design:

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

   ```
   quickstart/simulation/functional
   ```

2. Launch the simulation script:

   ```
   Questa SIM: vsim -do simulate_mti.do
   nc-sim: ./simulate_ncsim.sh
   vcs: ./simulate_vcs.sh
   ```

The simulation script compiles the functional model and the demonstration test bench, adds some relevant signals to a wave window, and then runs the simulation to completion.

You can then inspect the simulation transcript and waveform to observe the operation of the core.

Post-implementation Simulation

To run a timing simulation of the example design:

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

   ```
   quickstart/simulation/timing
   ```

2. Launch the simulation script:

   ```
   Questa SIM: vsim -do simulate_mti.do
   nc-sim: ./simulate_ncsim.sh
   vcs: ./simulate_vcs.sh
   ```

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

Demonstration Test Bench

For more information on the demonstration test bench, see Demonstration Test Bench, page 94 for the 10-Gigabit Ethernet MAC with XGMII interface and Demonstration Test Bench, page 96 for the 10-Gigabit Ethernet MAC with 64-bit interface.

Detailed Example Design

This chapter provides detailed information about the example design, including a description of 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

- **<project directory>**
  Top-level project directory; name is user-defined.
  - <project directory>/<component name>
    Core release notes file
    - <component name>/doc
      Product documentation
    - <component name>/example design
      - example_design/axi_ipif
        Files for the AXI-Lite to IPIF Bridge instanced in the <component_name>_block
      - example_design/fifo
        Files for the FIFO instanced in the client_loopback example design
    - <component name>/implement
      Implementation script files
      - implement/results
        Results directory, created after implementation scripts are run, and contains implement script results
    - <component name>/simulation
      Simulation scripts
      - simulation/functional
        Functional simulation files
      - simulation/timing
        Timing simulation files

The 10-Gigabit Ethernet MAC core directories and their associated files are defined in the following sections.

Note: The implement and timing simulation directories are only present when the core is generated with a Full or Hardware Evaluation license.

<project directory>

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

Table 7-1: **Project Directory**

Name	Description
<project_dir>	
<component_name>.ngc	Binary Xilinx implementation netlist. Describes how the core is to be implemented. Used as input to the Xilinx Implementation Tools.
<component_name>.v[hd]	VHDL structural simulation model. File used to support VHDL functional simulation of a core. The VHDL model passes customized parameters to the generic core simulation model.
<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>.xcp	As an output file, similar to the XCO file, except that it does not specify project-specific settings such as target architecture and output products.
<component_name>_flist.txt	Text file listing all of the output files produced when customized core was generated in the CORE Generator tool.
<component_name>.{veo|vho}	VHDL or Verilog instantiation template. This can be copied into the user design.

Back to Top

<project directory>/<component name>

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

Table 7-2: **Component Name Directory**

Name	Description
<project_dir>/<component_name>	
ten_gig_eth_mac_readme.txt	Core release notes file

Back to Top

<component name>/doc

The doc directory contains a hyperlink to the PDF documentation provided with the core.

Table 7-3: **Doc Directory**

Name	Description
<project_dir>/<component_name>/doc	
pg072-ten-gig-eth-mac.pdf	Hyperlink to *LogiCORE IP 10-Gigabit Ethernet MAC Product Guide* on the Xilinx website.

Back to Top

<component name>/example design

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

Table 7-4: **Example Design Directory**

Name	Description
<project_dir>/<component_name>/example_design	
<component_name>_example_design.v[hd]	Example design level VHDL or Verilog file for the example design. The fifo_block module is instanced along with the address swap block, if required.
<component_name>_example_design.ucf	UCF for the core and the example design.
<component_name>_fifo_block.v[hd]	FIFO and Block level VHDL/Verilog file. For cores without a simplex split, this instances the AXI4-Stream FIFO in addition to the block level.
<component_name>_block.v[hd]	Block level VHDL/Verilog file. This instances the appropriate interface block and the Ethernet MAC core.
address_swap.v[hd]	Address swap VHDL or Verilog file. This connects to the AXI4-Stream interface and swaps the destination and source addresses of frame.
xgmii_if.v[hd]	XGMII interface VHDL or Verilog file. This contains the DDR registers to implement the external XGMII interface. See 10-Gigabit Ethernet MAC with External XGMII Interface, page 104.
physical_if.v[hd]	PHY interface VHDL or Verilog file. This contains the SDR registers to implement the 64-bit interface. See 10-Gigabit Ethernet MAC with 64-bit SDR Interface, page 107.
client_loopback.v[hd]	Client loopback design example instanced in the FIFO and block level.

Back to Top

example_design/axi_ipif

This directory contains files for the AXI4-Lite to IPIF Bridge instanced in the <component_name>_block.

Table 7-5: **axi_ipif Directory**

Name	Description
<project_dir>/<component_name>/example_design/axi_ipif	
address_decoder.v[hd]	Address decoder VHDL or Verilog file.
axi_lite_ipif.v[hd]	AXI4-Lite-to-IPIF bridge VHDL or Verilog file. Top level for the function.
axi_lite_ipif_wrapper.v[hd]	VHDL or Verilog wrapper to map permitted generics/parameters.
counter_f.v[hd]	Parameterizable N bit counter VHDL or Verilog file.
ipif_pkg.vhd	VHDL package containing component declarations for the AXI4-Lite to IPIF bridge.
pselect_f.v[hd]	Peripheral select VHDL or Verilog file.
slave_attachment.v[hd]	AXI4-Lite Slave function VHDL or Verilog file.

Back to Top

example_design/fifo

This directory contains the files for the FIFO instanced in the client_loopback example design.

Table 7-6: **FIFO Directory**

Name	Description
<project_dir>/<component_name>/example_design/fifo	
xgmac_fifo.v[hd]	Top-level of the FIFO.
xgmac_fifo_pack.vhd	Component declarations for the FIFO.
axi_fifo.v[hd]	Generic AXI4-Stream Fifo. This implements the logic to store and forward a good frame and drop a bad frame on AXI4-Stream interface.
fifo_ram.v[hd]	Block RAM wrapper used by both FIFOs.

Back to Top

<component name>/implement

The implement directory contains the core implementation script files.

Table 7-7: **Implement Directory**

Name	Description
<project_dir>/<component_name>/implement	
implement.sh	Linux shell script that processes the example design through the Xilinx tool flow.
implement.bat	Windows batch file that processes the example design through the Xilinx tool flow.
xst.prj	XST project file for the example design; it enumerates all the HDL files that need to be synthesized.
xst.scr	XST script file for the example design.

Back to Top

implement/results

This directory is produced by the implement scripts and is used to run the example design files and the `<component_name>.ngc` file through the Xilinx implementation tools. After these are run, this directory contains the following files for timing simulation.

Table 7-8: **Results Directory**

Name	Description
<project_dir>/<component_name>/implement/results	
routed.v[hd]	Back-annotated SimPrim-based VHDL or Verilog design. Used for timing simulation.
routed.sdf	Timing information for simulation.

Back to Top

<component name>/simulation

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

Table 7-9: **Simulation Directory**

Name	Description
<project_dir>/<component_name>/simulation	
demo_tb.v[hd]	VHDL or Verilog demonstration test bench for the 10-Gigabit Ethernet MAC core. More information can either be found in 10-Gigabit Ethernet MAC with External XGMII Interface in Chapter 7 or 10-Gigabit Ethernet MAC with 64-bit SDR Interface in Chapter 7.

Back to Top

simulation/functional

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

Table 7-10: **Functional Directory**

Name	Description
<project_dir>/<component_name>/simulation/functional	
simulate_mti.do	Questa SIM macro file that compiles the example design sources and the structural simulation model then runs the functional simulation to completion.
wave_mti.do	Questa SIM macro file that opens a wave window and adds interesting signals to it. It is called by the simulate_mti.do macro file.
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 Cadence IES.
wave_ncsim.sv	Cadence IES macro file that opens a wave window and adds interesting signals to it. It is called by the simulate_ncsim.sh script.
simulate_vcs.sh (Verilog only)	Shell script that compiles the example design sources and the structural simulation model then runs the functional simulation to completion using VCS.
vcs_session.tcl (Verilog only)	VCS DVE Tcl script that opens a wave window and adds interesting signals to it. This macro is used by the simulate_vcs.sh script.
vcs_commands.key (Verilog only)	VCS commands file. This file is called by the simulate_vcs.sh script.

Back to Top

simulation/timing

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

Table 7-11: **Timing Directory**

Name	Description
<project_dir>/<component_name>/simulation/timing	
simulate_mti.do	Questa SIM macro file that compiles the VHDL or Verilog timing model and demonstration test bench then runs the timing simulation to completion.
wave_mti.do	Questa SIM macro file that opens a wave window and adds interesting signals to it. It is called by the simulate_mti.do macro file.

Table 7-11: **Timing Directory *(Cont'd)***

Name	Description
simulate_ncsim.sh	Linux shell script that compiles the example design sources and the timing model then runs the functional simulation to completion using Cadence IES.
wave_ncsim.sv	Cadence IES macro file that opens a wave window and adds interesting signals to it. It is called by the simulate_ncsim.sh script.
simulate_vcs.sh (Verilog only)	Shell script that compiles the example design sources and the structural simulation model then runs the timing simulation to completion using VCS.
vcs_session.tcl (Verilog only)	VCS DVE Tcl script that opens a wave window and adds interesting signals to it. This macro is used by the simulate_vcs.sh script.
vcs_commands.key (Verilog only)	VCS commands file. This file is called by the simulate_vcs.sh script.

Back to Top

Example Designs and Demonstration Test Benches

This section discusses the designs and test benches for these examples:

- 10-Gigabit Ethernet MAC with External XGMII Interface
- 10-Gigabit Ethernet MAC with 64-bit SDR Interface
- AXI4-Lite to IPIF Converter Block

10-Gigabit Ethernet MAC with External XGMII Interface

Note: External XGMII interface is not supported on Spartan®-6 FPGA designs.

Example Design and HDL Wrapper

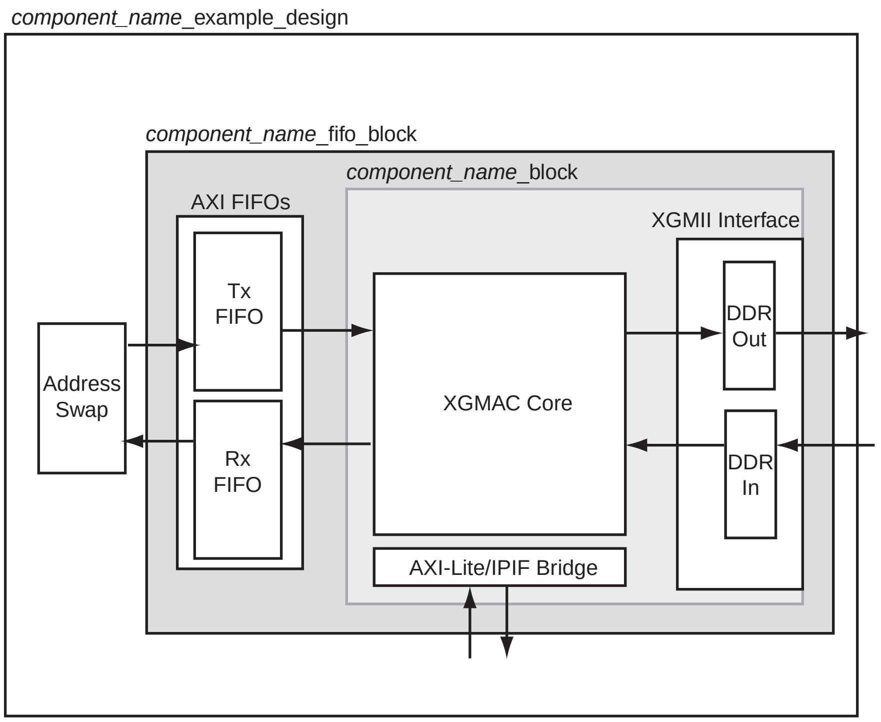


Figure 7-1: **Example Design and HDL Wrapper for 10-Gigabit Ethernet MAC with XGMII Interface**

The example design and HDL wrapper contain the following:

- Global clock buffers and Digital Clock Managers (DCMs) or Mixed-Mode Clock Managers (MMCMs)
- HDL sources for client loopback design

The client loopback design performs these functions:

- Drops frame marked as bad by the core
- Crosses clock domain from received clock to transmit clock safely using an asynchronous FIFO

The address swap module performs this function:

- Swaps the destination and source address field in the received Ethernet frame

The XGMII block performs these functions:

- Receiver DCM/MMCM and clock buffer
- DDR logic for the XGMII Interface
- AXI4-Lite to IPIF bridge logic

Demonstration Test Bench

The demonstration test bench is a simple VHDL or Verilog program to exercise the example design and the core. It consists of transactor procedures or tasks that connect to the PHY-side 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.

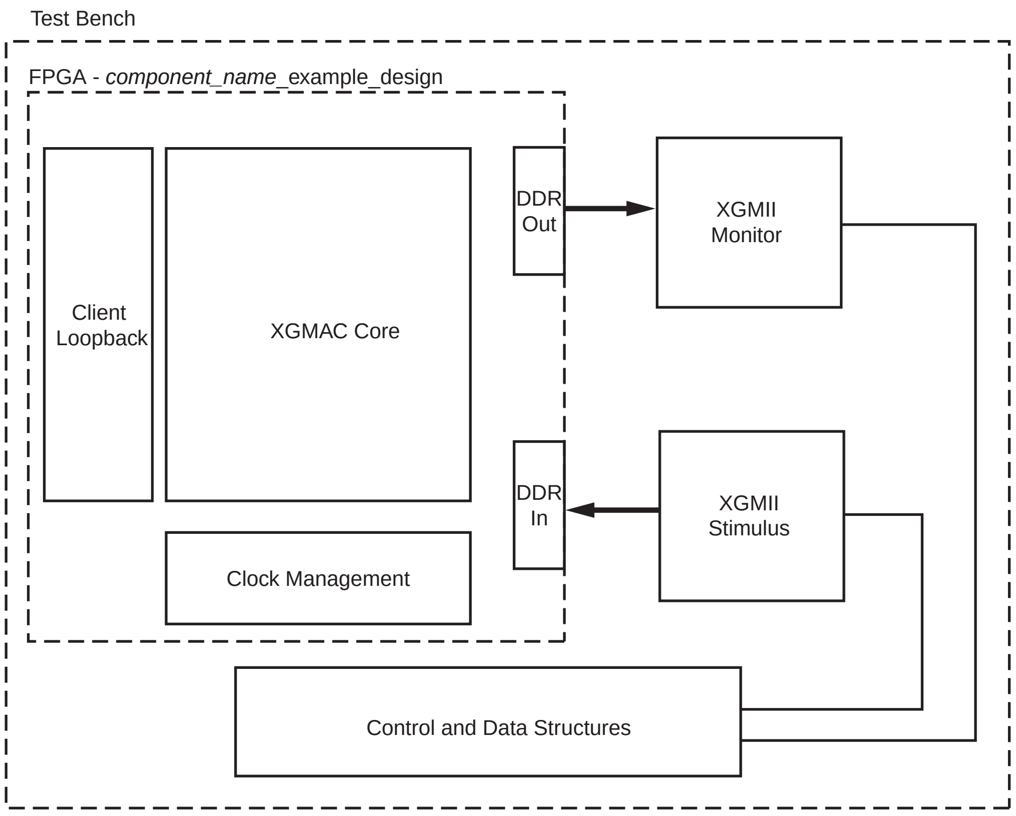


Figure 7-2: **Demonstration Test Bench for 10-Gigabit Ethernet MAC with XGMII Interface**

Because the address swap design swaps the destination and source field, the CRC is different on the outbound frame compared to that injected into the receiver. This is taken into account by the test bench when checking the transmitted data.

10-Gigabit Ethernet MAC with 64-bit SDR Interface

Example Design and HDL Wrapper

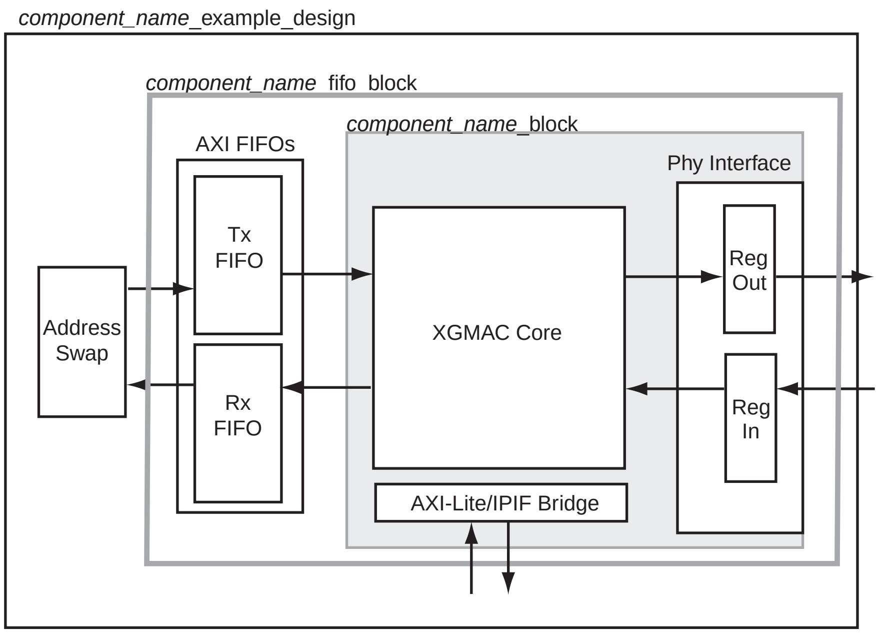


Figure 7-3: **Example Design and HDL Wrapper for 10-Gigabit Ethernet MAC with 64-bit Interface**

The example design and HDL wrappers contain the following:

- Global clock buffers and Digital Clock Managers (DCMs) or Mixed-Mode Clock Managers (MMCMs)
- HDL sources for client loopback design

The client loopback design performs these functions:

- Drops frame marked as bad by the 10-Gigabit Ethernet MAC core
- Crosses clock domain from received clock to transmit clock safely using an asynchronous FIFO

The address swap module performs this function:

- Swaps the destination and source address field in the received Ethernet frame

The physical interface block performs these functions:

- Registers for the 64-bit SDR interface
- Receiver DCM/MMCM and clock buffer
- AXI4-Lite to IPIF bridge logic

Demonstration Test Bench

The demonstration test bench is a simple VHDL or Verilog program to exercise the example design and the core itself. It consists of transactor procedures or tasks which connect to the PHY-side 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.

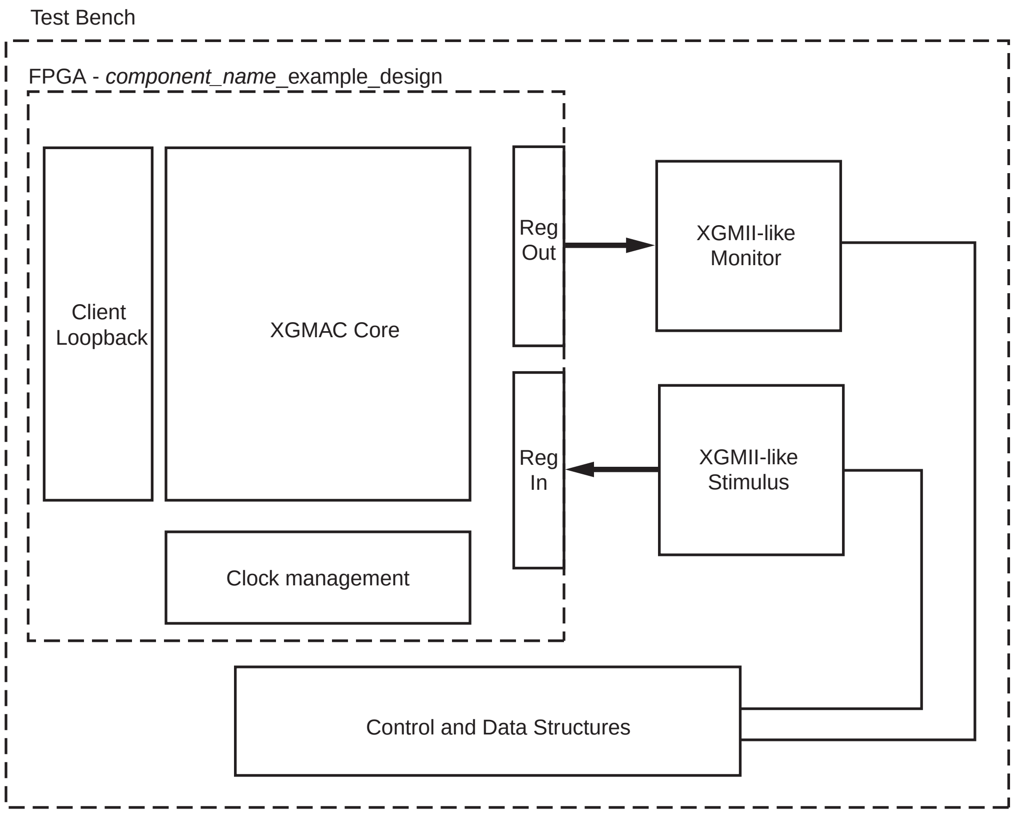


Figure 7-4: **Demonstration Test Bench for 10-Gigabit Ethernet MAC with 64-bit Interface**

Because the address swap design swaps the destination and source field, the CRC is different on the outbound frame compared to that injected into the receiver. This is taken into account by the test bench when checking the transmitted data.

AXI4-Lite to IPIF Converter Block

AXI4-Lite to IPIF converter block is used in the example design at block level to convert the AXI4-Lite transactions to IPIF transactions which is the actual management interface of the core. Table 7-12 describes the ports associated with the IPIF interface.

Table 7-12: **IPIF Interface Port Description**

Name	Direction	Description
bus2ip_clk	In	IPIF Clock.
bus2ip_reset	In	IPIF Reset.
bus2ip_cs	In	IPIF Chip select, qualifies the bus2ip_rnw signal.
bus2ip_rnw	In	IPIF read not write signal. When this signal is 0 along with bus2ip_cs being 1 during clock positive edge Write operation is requested. When this signal is 1 along with bus2ip_cs being 1 during clock positive edge Read operation is requested.
bus2ip_data[31:0]	In	IPIF Write data to the core.
ip2bus_data[31:0]	Out	IPIF Read data from the core.
bus2ip_addr[31:0]	In	Address of the register accessed.
ip2bus_rdack	Out	Read Transaction acknowledge signal
ip2bus_wrack	Out	Write Transaction acknowledge signal
ip2bus_error	Out	Error on invalid transaction, asserted along with the read/write acknowledge signal.

Figure 7-5 explains the timing of IPIF read and write transactions.

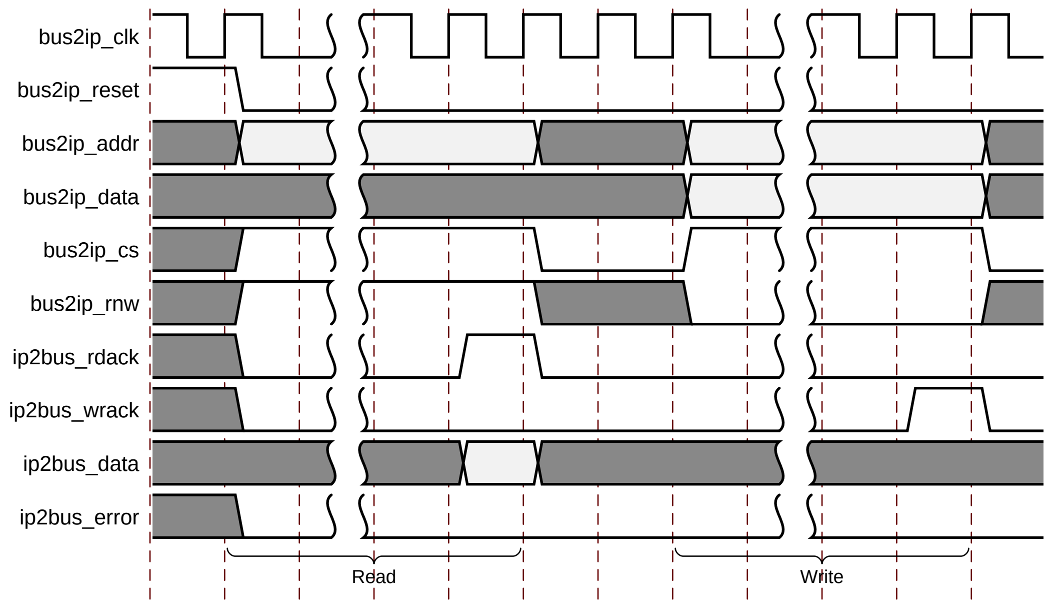


Figure 7-5: **Timing of IPIF Read and Write Transactions**

Implementation and Test Scripts

Implementation Script

The implementation script is either a shell script or batch file that processes the example design through the Xilinx tool flow. It is located in one of the following directories, depending on the platform:

Linux

```
project_dir/component_name/implement/implement.sh
```

Windows

```
project_dir/component_name/implement/implement.bat
```

Full System Hardware Evaluation or Full License Implement Script

If the core is generated with the Full System Hardware Evaluation license or Full license, the implement script performs these steps:

- The example HDL wrapper and client loopback logic 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
- These files are copied into the `<component name>/implement/results` directory

Simulation Only Evaluation License Implement Script

If the core is generated with the Simulation Only Evaluation license, no implement directory or script is generated.

Simulation Scripts

Simulation macro files are provided for Questa SIM and shell scripts are provided for Cadence IES and VCS. The scripts automate the simulation of the test bench and can be found in these locations:

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
```

Timing

```
<project_dir>/<component_name>/simulation/timing/simulate_mti.do
<project_dir>/<component_name>/simulation/timing/simulate_ncsim.sh
<project_dir>/<component_name>/simulation/timing/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 (with timing information if a Full System Evaluation license or Full license is in use)
- Opens a Wave window and adds some interesting signals (wave.do)
- Runs the simulation to completion

Verification, Compliance, and Interoperability

Simulation

The 10-Gigabit Ethernet MAC core has been verified in simulation. A highly parameterizable constrained random simulation test suite has been used to verify the core. Tests included:

- Configuration register access through Management Interface
- Local Fault and Remote Fault handling
- Frame transmission
- Frame reception
- CRC validity
- Handling of CRC errors
- Statistic counter access through Management Interface and validity of counts
- Statistic vector validity
- Initiating MDIO transactions through Management Interface
- Use of custom preamble field
- Variable Frame Length and MTU

Hardware Verification

The core has been hardware validated on Virtex®-6 and Spartan®-6 silicon with the Xilinx LogiCORE™ IP XAUI core. The design comprises the Ethernet MAC, XAUI, a ping loopback FIFO, and test pattern generator all under embedded MicroBlaze™ processor control.

This design has also been used for conformance and interoperability testing at the University of New Hampshire Interoperability Lab.

Calculating the DCM Fixed Phase-Shift Value

This appendix describes how to calculate the fixed phase-shift value of the Digital Clock Manager (DCM).

Requirement for DCM Phase Shifting

A DCM is used in the receiver clock path to meet the input setup and hold requirements when using the core with an XGMII. In these cases, a fixed phase-shift offset is applied to the receiver clock DCM to skew the clock; this performs static alignment by using the receiver clock DCM to shift the internal version of the receiver clock such that its edges are centered on the data eye at the IOB DDR flip-flops. The ability to shift the internal clock in small increments is critical for sampling high-speed source synchronous signals such as XGMII. For statically aligned systems, the DCM output clock phase offset (as set by the phase-shift value) is a critical part of the system, as is the requirement that the PCB is designed with precise delay and impedance-matching for all the XGMII receiver data bus and control signals.

You must determine the best DCM setting (phase shift) to ensure that the target system has the maximum system margin to perform across voltage, temperature, and process (multiple chips) variations. Testing the system to determine the best DCM phase-shift setting has the added advantage of providing a benchmark of the system margin based on the UI (unit interval or bit time). Equation B-1 defines the system margin.

$$\text{System Margin (ps)} = \text{UI(ps)} \times (\text{working phase-shift range}/128) \qquad \textit{Equation B-1}$$

Finding the Ideal Phase-Shift Value for Your System

Xilinx cannot recommend a singular phase-shift value that is effective across all hardware families. Xilinx does not recommend attempting to determine the phase-shift setting empirically. In addition to the clock-to-data phase relationship, other factors such as package flight time (package skew) and clock routing delays (internal to the device) affect the clock to data relationship at the sample point (in the IOB) and are difficult to characterize.

Xilinx recommends extensive investigation of the phase-shift setting during hardware integration and debugging. The phase-shift settings provided in the example design constraint file is a placeholder, and works successfully in back-annotated simulation of the example design.

Perform a complete sweep of phase-shift settings during your initial system test. Use only positive (0 to 255) phase-shift settings, and use a test range that covers a range of no less than 128, corresponding to a total 180 degrees of clock offset. This does not imply that 128 phase-shift values must be tested; increments of 4 (52, 56, 60) correspond to roughly one DCM tap, and consequently provide an appropriate step size. Additionally, it is not necessary to characterize areas outside the working phase-shift range.

At the edge of the operating phase-shift range, system behavior changes dramatically. In eight phase-shift settings or less, the system can transition from no errors to exhibiting errors. Checking the operational edge at a step size of two (on more than one board) refines the typical operational phase-shift range. After the range is determined, choose the average of the high and low working phase-shift values as the default. During the production test, Xilinx recommends that you re-examine the working range at corner case operating conditions to determine whether any final adjustments to the final phase-shift setting are needed.

Use the FPGA Editor to generate the required test file set instead of resorting to multiple PAR runs. Performing the test on design files that differ only in phase-shift setting prevents other variables from affecting the test results. FPGA Editor operations can even be scripted further, reducing the effort needed to perform this characterization.

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 process for debugging process and a flow diagram to guide you through debugging the 10-Gigabit Ethernet MAC core.

The following topics are included in this appendix:

Finding Help on Xilinx.com

To help in the design and debug process when using the 10-Gigabit Ethernet MAC, the Xilinx Support web page 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 10-Gigabit Ethernet MAC. 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 10-Gigabit Ethernet MAC 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 relevant to Ethernet IP is located at 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 for this core are listed below, and 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.

Contacting 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.

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 debug 10-Gigabit Ethernet MAC design issues. It is important to know which tools are useful for debugging various situations.

Example Design

The 10-Gigabit Ethernet MAC comes with a synthesizable example design complete with a functional test benches. Information on the example design can be found in Chapter 7, Detailed Example Design for the ISE Design Suite.

ChipScope Pro Tool

The ChipScope™ Pro tool inserts logic analyzer, bus analyzer, and virtual I/O cores directly into your design. The ChipScope Pro 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 on the ChipScope Pro tool, see www.xilinx.com/tools/cspro.htm.

Reference Boards

Various Xilinx development boards support 10 Gb Ethernet. These boards can be used to prototype designs and establish that the core can communicate with the system.

- 7 series evaluation boards:
  - KC705
  - KC724

Simulation Debug

The simulation debug flow for Questa® SIM is shown in Figure C-1. A similar approach can be used with other simulators.

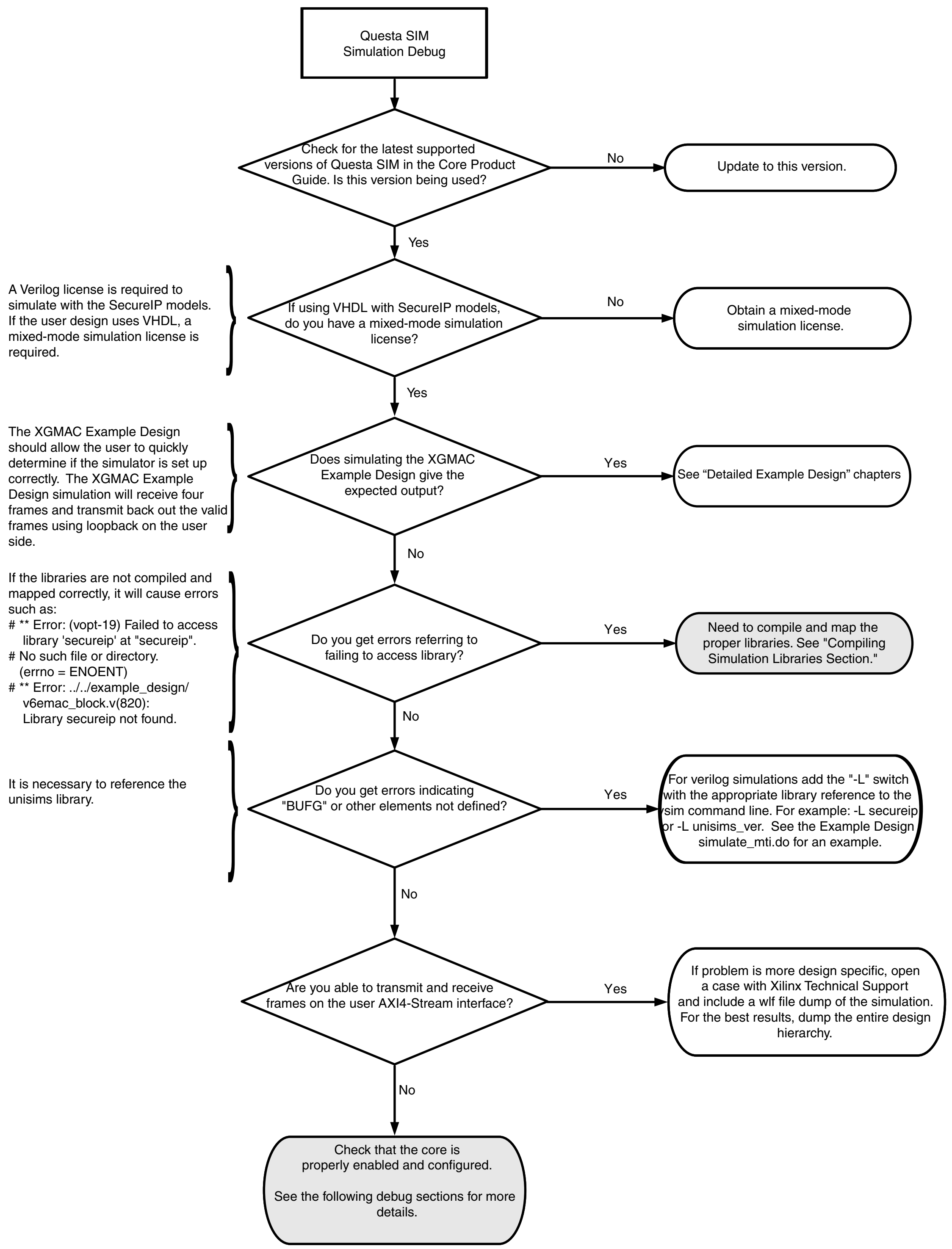

Questa SIM Simulation Debug
Check for the latest supported versions of Questa SIM in the Core Product Guide. Is this version being used?
No
Update to this version.
Yes
A Verilog license is required to simulate with the SecureIP models. If the user design uses VHDL, a mixed-mode simulation license is required.
If using VHDL with SecureIP models, do you have a mixed-mode simulation license?
No
Obtain a mixed-mode simulation license.
Yes
The XGMAC Example Design should allow the user to quickly determine if the simulator is set up correctly. The XGMAC Example Design simulation will receive four frames and transmit back out the valid frames using loopback on the user side.
Does simulating the XGMAC Example Design give the expected output?
Yes
See "Detailed Example Design" chapters
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/v6emac_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
It is necessary to reference the unisims library.
Do you get errors indicating "BUFG" or other elements 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 receive frames on the user AXI4-Stream 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 core is properly enabled and configured.
See the following debug sections for more details.

Figure C-1: **Questa SIM Debug Flow Diagram**

Compiling Simulation Libraries

Compile the Xilinx simulation libraries, either by using the Xilinx Simulation Library Compilation Wizard, or by using the compxlib command line tool.

Xilinx Simulation Library Compilation Wizard

A GUI wizard provided as part of the Xilinx software can be launched to assist in compiling the simulation libraries by typing `compxlib` in the command prompt.

For more information see the Software Manuals and specifically the *Command Line Tools Reference Guide* under the section titled compxlib.

Assuming the Xilinx and Questa SIM environments are set up correctly, this is an example of compiling the SecureIP and UNISIM libraries for Verilog into the current directory.

```
compxlib -s mti_se -arch virtex6 -l verilog -lib secureip -lib unisims
    -dir ./
```

There are many other options available for compxlib described in the *Command Line Tools Reference Guide*.

Compxlib produces a questasim.ini file containing the library mappings. In Questa SIM, to see the current library mappings, type `vmap` at the prompt. The mappings can be updated in the .ini file or to map a library at the Questa SIM prompt type:

```
vmap [<logical_name>] [<path>]
```

For example:

```
vmap unisims_ver C:\my_unisim_lib
```

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 tool is a valuable resource to use in hardware debug and the signal names mentioned in the following individual sections can be probed using the ChipScope 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
- Problems with Transmitting and Receiving Frames
- Problems with the MDIO
- Link Fault

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.

Problems with Transmitting and Receiving Frames

Problems with data reception or transmission can be caused by a wide range of factors. The following list contains common causes to check for:

- Verify that the whole 10-Gigabit Ethernet MAC block is not being held in reset. The whole block is held in reset if the main reset input or if a locked signal from an MMCM is low.
- Verify that both the receiver and transmitter are enabled and not being held in reset.
- Verify that the 10-Gigabit Ethernet MAC is configured correctly and that the latest core version is being used. Try running a simulation to check if the failure is hardware-specific.
- If using an external XGMII interface, check if setup and hold requirements are met.
- Verify that the link is up between the PHY and its link partner. Frames can be dropped if in a link fault condition (see Link Fault for more details on the behavior of the 10-Gigabit Ethernet MAC). If using the XAUI or RXAUI cores, see the Debugging Guide section of the *LogiCORE IP XAUI Product Guide* [Ref 6] or *LogiCORE IP RXAUI Users Guide* [Ref 7] for more details on troubleshooting the cause of the link fault.
- If using an external PHY, is data received correctly if the PHY is put in loopback? If so, the issue might be on the link between the PHY and its link partner.
- Are received frames being dropped by user logic because `rx_axis_mac_tuser` is asserted? See Frame Reception with Errors in Chapter 3 for details on why frames are marked bad by the Ethernet MAC. The ChipScope tool can be inserted to get more details on the bad frames.
- Add the ChipScope tool to the design to look at the RX and TX AXI4-Stream and physical interface data signals, control signals and statistics vectors.

Problems with the MDIO

See MDIO Interface in Chapter 3 for detailed information about performing MDIO transactions.

Things to check for:

- Check that the MDC clock is running and that the frequency is 2.5 MHz or less. If using the MDIO control registers to perform MDIO accesses, the MDIO interface does not work until the clock frequency is set with CLOCK_DIVIDE. The MDIO clock with a maximum frequency of 2.5 MHz is derived from the `s_axi_aclk` clock.

- Ensure that the MAC and PHY are not held in reset. Be sure to check the polarity of the reset to your external PHY. Many PHYs have an active-low 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.
- If using the management interface to access the MDIO, check if the issue is just with the MDIO control registers or if there are also issues reading and writing MAC registers with the management interface.
- If accessing MDIO registers of the Ethernet 10-Gigabit Ethernet PCS/PMA, XAUI or RXAUI core, check that the PHYAD field placed into the MDIO frame matches the value placed on the `phyad`[4:0] port of the PHY core.
- Has a simulation been run? Verify in simulation and/or a ChipScope tool capture that the waveform is correct for accessing the management interface for a MDIO read/write. The demonstration test bench delivered with the core provides an example of MDIO accesses.

Link Fault

The 10-Gigabit Ethernet MAC contains a Link Fault State machine as described in the *IEEE802.3-2008* standard. This mandates that:

- When a MAC receives Local Fault (LF) ordered sets, it continuously transmits Remote Fault (RF) ordered sets;
- When a MAC receives a Remote Fault, it continuously transmits Idle ordered sets.

This latter fault mode can be interpreted as inactivity on the part of the MAC; if no traffic is appearing on the XGMII interface in the transmit direction, check the fault state in the management registers.

For more information, see Transmission of Frames During Local/Remote Fault Reception, page 45.

Data Throughput

The 10-Gigabit Ethernet MAC is capable of running at the maximum throughput designed by the IEEE specification. If maximum throughput is not being seen on transmission:

- Check that back-to-back frames are being presented on the AXI interface. For more information, see Back-to-Back Continuous Transfers, page 41.
- Check if Deficit Idle Count has been enabled to reduce average IFG transmitted. For more information, see Deficit Idle Count (DIC), page 45.

Interface Debug

AXI4-Lite Interfaces

Read from a register that does not have all 0s as a default to verify that the interface is functional. See Figure 3-25 for a read timing diagram. Output `s_axi_arready` asserts when the read address is valid, and output `s_axi_rvalid` asserts when the read data/ response is valid. If the interface is unresponsive, ensure that the following conditions are met:

- The `S_AXI_ACLK and ACLK` inputs are connected and toggling.
- The interface is not being held in reset, and `S_AXI_ARESET` is an active-Low reset.
- The interface is enabled, and `s_axi_aclken` is active-High (if used).
- The main core clocks are toggling and that the enables are also asserted.
- If the simulation has been run, verify in simulation and/or a ChipScope tool capture that the waveform is correct for accessing the AXI4-Lite interface.

AXI4-Stream Interfaces

If data is not being transmitted or received, check the following conditions.

- If transmit `<interface_name>_tready` is stuck low following the `<interface_name>_tvalid` input being asserted, the core cannot send data.
- If the receive `<interface_name>_tvalid` is stuck low, the core is not receiving data.
- Check that the ACLK inputs are connected and toggling.
- Check that the AXI4-Stream waveforms are being followed (Figure 3-6).
- Check core configuration.
- Add appropriate core specific checks.

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

See the IEEE website for more information on this standard:

1. *IEEE Standard 802.3-2008,* “Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications”

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

2. *XST User Guide* (ISE documentation)
3. *Xilinx Synthesis and Simulation Design Guide*
4. *LogiCORE IP Ethernet 1000BASE-X PCS/PMA or SGMII Product Guide* (PG047)
5. *LogiCORE IP Ten Gigabit Ethernet PCS/PMA Product Guide* (PG068)
6. *LogiCORE IP XAUI Product Guide* (PG053)
7. *LogiCORE IP RXAUI Product Guide* (PG083)

Technical Support

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 release. This release is for ISE Design Suite v14.2 and Vivado Design Suite v2012.2. The core version is v11.4. This document replaces UG773, *LogiCORE IP 10-Gigabit Ethernet MAC User Guide*, and DS813, *LogiCORE IP 10-Gigabit Ethernet MAC Data Sheet*.
12/18/12	2.0	• Updated core to v11.5 and this release is for Vivado Design Suite v2012.4 only. • Updated License and Ordering Information. • Updated to support 7 series FPGAs only. • Updated tx_axis_tready direction to out in Table 3-1. • Updated Fig. 4-1 GUI and WAN support description. • Added new Debug section and minor document updates.
03/20/13	2.1	• Updated core to v11.6 for ISE Design Suite v14.5 without Vivado content. • Added Bit[31] Reserved for Table 3-11 • Updated Fig. 4-1 GUI • Updated to Questa SIM

Notice of Disclaimer