# LogiCORE IP 10-Gigabit Ethernet MAC v13.1

## *Product Guide*

**Vivado Design Suite**

**PG072 April 2, 2014**

XILINX ALL PROGRAMMABLE™

# Table of Contents

## Chapter 5: Example Design

## Chapter 6: Test Bench



## Appendix A: Verification, Compliance, and Interoperability



## Appendix B: Migrating and Upgrading



## Appendix C: Calculating the DCM Fixed Phase-Shift Value



## Appendix D: Debugging



## Appendix E: Additional Resources and Legal Notices

# 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
- 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 802.3 and 802.1Qbb (priority-based) flow control in both directions
- Provides MDIO STA master interface to manage PHY layers
- Extremely customizable; trade resource usage against functionality
- 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.

| LogiCORE IP Facts Table | |
|---|---|
| **Core Specifics** | |
| Supported Device Family[(1)] | UltraScale™ Architecture, Zynq®-7000, 7 Series |
| Supported User Interfaces | AXI4-Lite, AXI4-Stream |
| Resources | See Table 2-1. |
| **Provided with Core** | |
| Design Files | Encrypted RTL |
| Example Design | Verilog and VHDL |
| Test Bench | Verilog and VHDL |
| Constraints File | XDC |
| Simulation Model | Verilog and VHDL |
| Supported S/W Driver | N/A |
| **Tested Design Flows[(2)]** | |
| Design Entry | Vivado® Design Suite<br>IP Integrator |
| Simulation | For supported simulators, see the Xilinx Design Tools: Release Notes Guide. |
| Synthesis | Vivado Synthesis |
| **Support** | |
| Provided by Xilinx @ www.xilinx.com/support | |

**Notes:**

1. For a complete listing of supported devices, see the Vivado IP catalog. Speed grades are –2 for Artix-7 devices.
2. For the supported versions of the tools, see the Xilinx Design Tools: Release Notes Guide.

# 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* [Ref 1] and supports the high-bandwidth demands of network Internet Protocol (IP) traffic on LAN, MAN, and WAN networks. The core works with the 7 series devices.

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

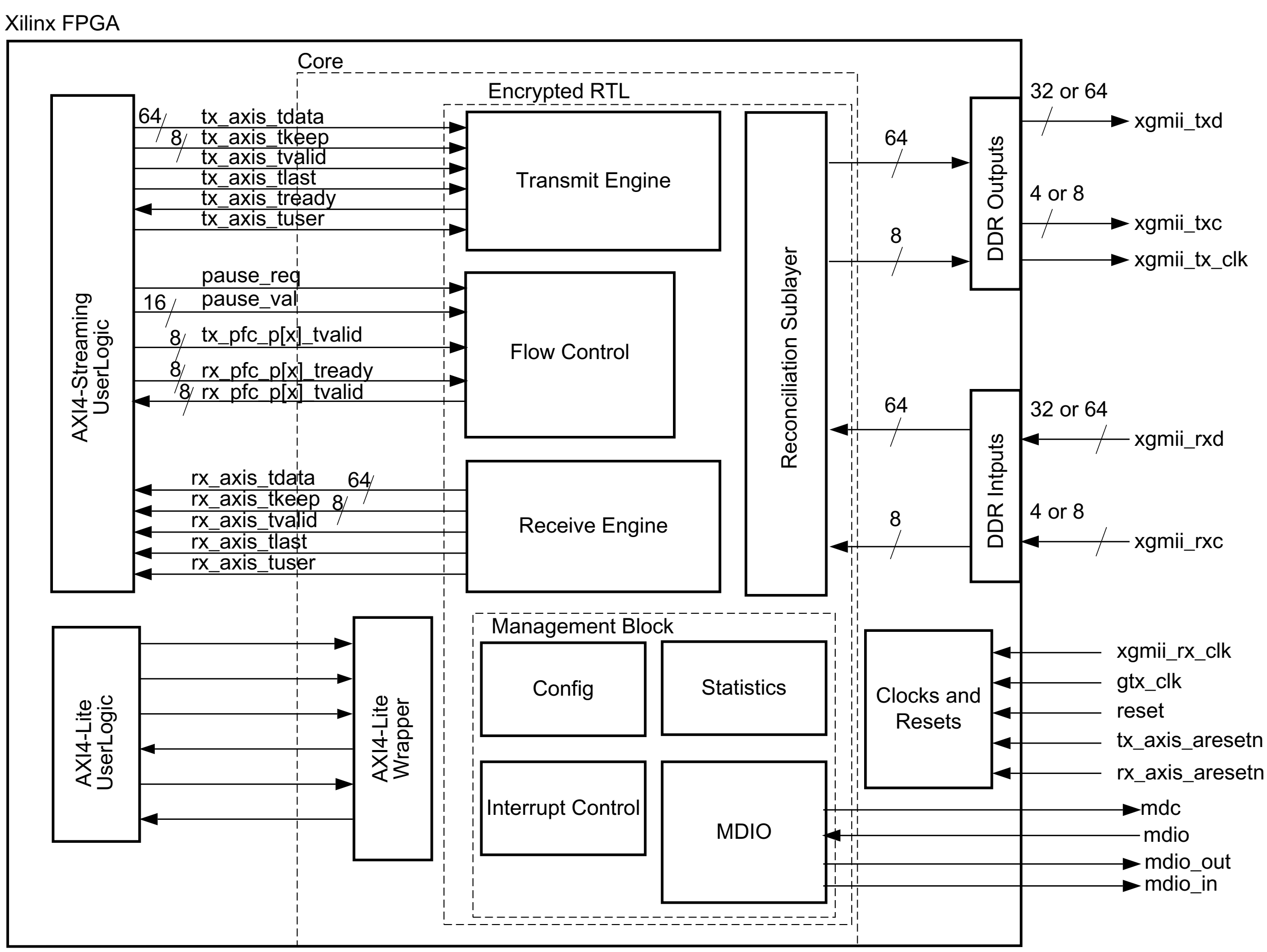


*Figure 1-1:* **Implementation of the 10-Gigabit Ethernet MAC Core**

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 XDC 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. 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 both 802.3 and, optionally, 802.1Qbb (priority-based) flow control in both transmit and receive directions. The Transmit side of the core modifies the interframe gap (IFG), using Deficit Idle Count to maintain the effective data rate of 10 Gb/s as described in *IEEE Standard 802.3-2008* [Ref 1].

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.

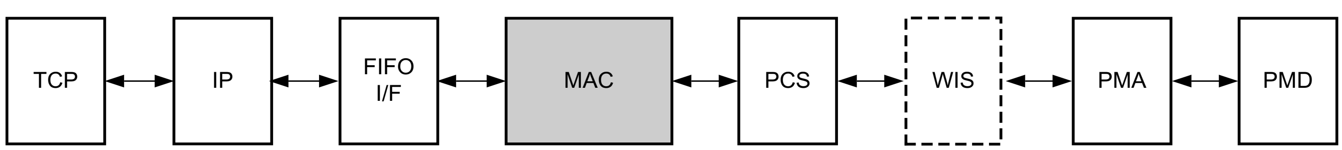


*Figure 1-2:* **Typical Ethernet System Architecture**

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.

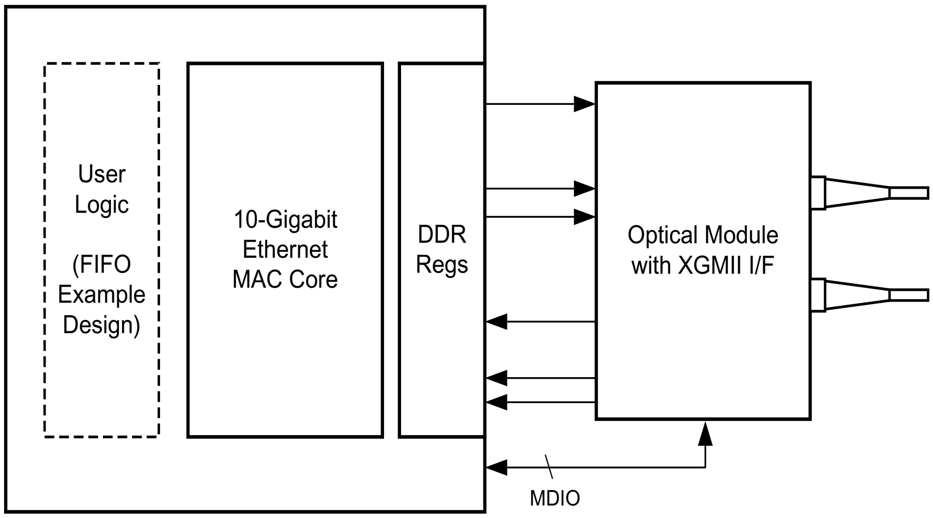


*Figure 1-3:* **10-Gigabit Ethernet MAC Core Connected to PHY with XGMII Interface**

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.

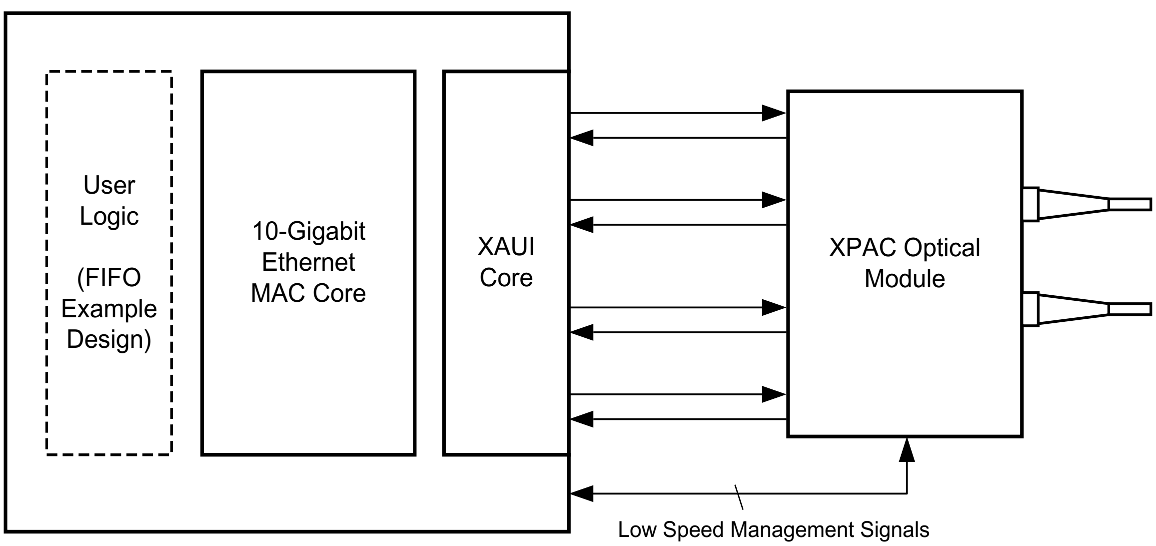


*Figure 1-4:* **10-Gigabit Ethernet MAC Core Used with Xilinx XAUI Core**

See Interfacing to the Xilinx XAUI IP Core, page 93 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 95 and Interfacing to the 10-Gigabit Ethernet PCS/PMA Core, page 97 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 Vivado Design Suite. For full access to all core functionalities in simulation and in hardware, you must purchase a license for the core. Contact your local Xilinx sales representative for information about pricing and availability.

For more information, visit the 10-Gigabit Ethernet 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.

## License Checkers

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

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

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

# 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

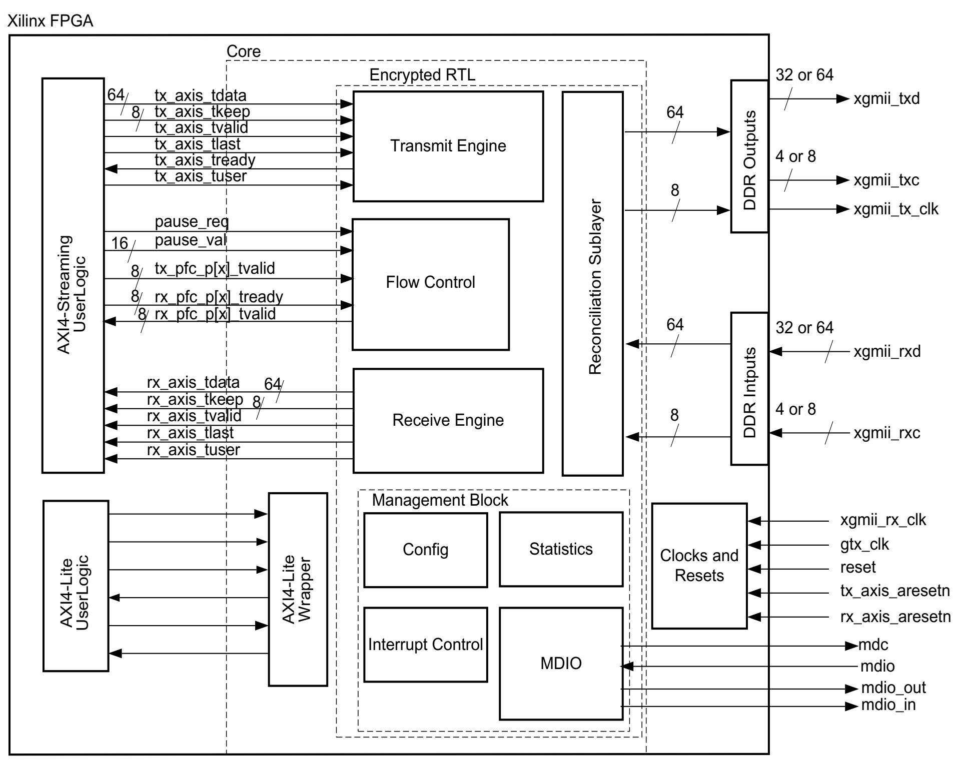


*Figure 2-1:* **Implementation of the 10-Gigabit Ethernet MAC Core**

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.

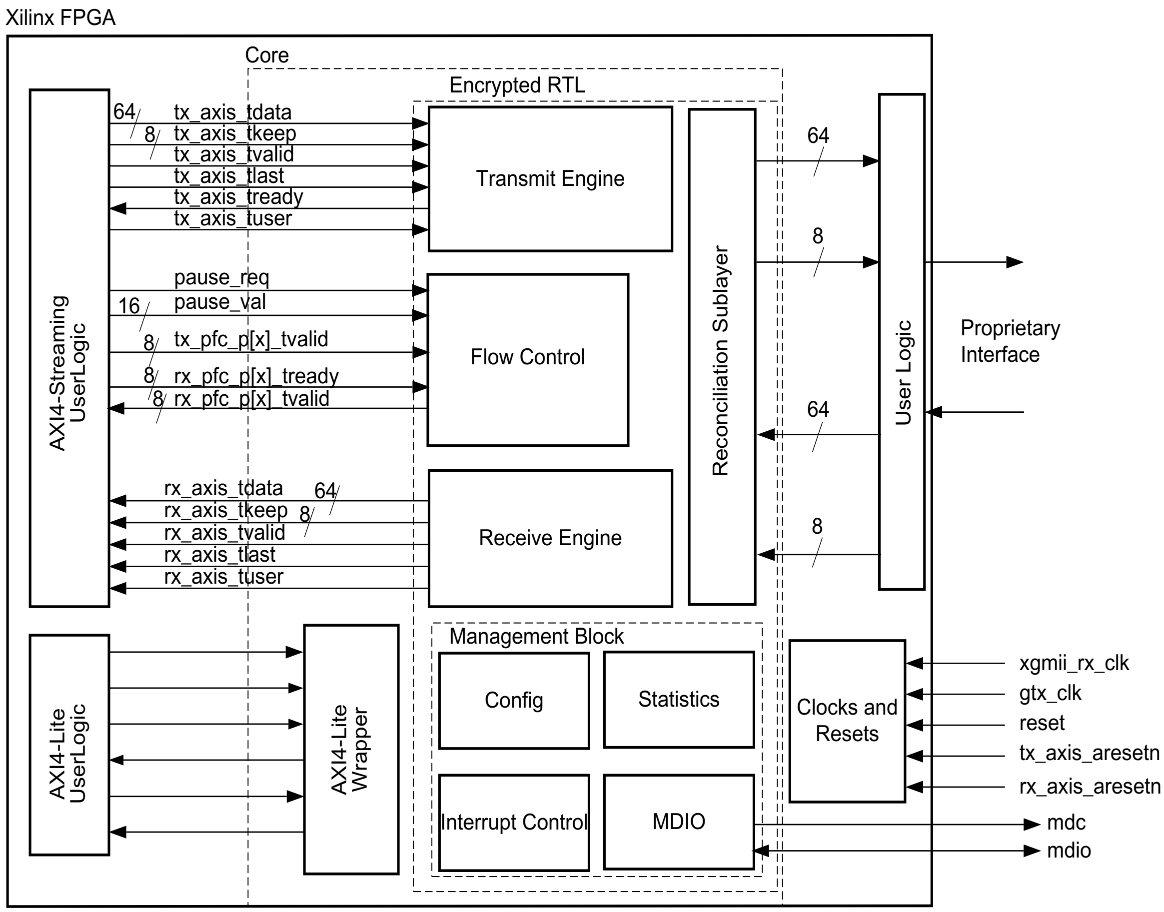


*Figure 2-2:* **Implementation of the Core with User Logic on PHY Interface**

# Standards

The LogiCORE IP 10-Gigabit Ethernet MAC core is designed to the *IEEE Standard 802.3-2008* [Ref 1] 10-Gigabit Ethernet specification.

Supports IEEE 802.1Qbb priority-based flow control defined in *IEEE Standard 802.1Qbb-Priority-based Flow Control* [Ref 2].

# 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 six 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 resource counts for the various core options using Virtex®-7 FPGAs.

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 2-1:* **Device Utilization – 7 Series FPGAs**

| Parameter Values | | | | Resource Usage | | |
|---|---|---|---|---|---|---|
| **Device Family** | **Physical Interface** | **Management Interface** | **Statistic Counters** | **LUTs**[1] | **FFs**[1] | **BUFGs** |
| Virtex-7 | XGMII | TRUE | TRUE | 3940[2] | 4360[2] | 2 |
| | | | FALSE | 3130[2] | 3070[2] | 2 |
| | | FALSE | FALSE | 2860[3] | 2840[3] | 2 |
| | Internal | TRUE | TRUE | 3940[2] | 4220[2] | 2 |
| | | | FALSE | 3130[2] | 3060[2] | 2 |
| | | FALSE | FALSE | 2860[3] | 2700[3] | 2 |

**Notes:**

1. For WAN support, add 100 LUTs and 30 FFs to these numbers.
2. For PFC enabled, add 550 LUTs and 320 FFs to these numbers.
3. For PFC enabled, add 470 LUTs and 300 FFs to these numbers.

The results are post-implementation, using tool default settings. LUT counts include SRL16s or SRL32s.

Resources required for the 10-Gigabit Ethernet MAC core have been estimated for the Virtex-7 devices (Table 2-1). These values were generated using the Vivado® Design Suite.

# Port Descriptions

Port descriptions for the 10-Gigabit Ethernet MAC IP are located in these sections:

- AXI4-Stream Interface – Transmit
- AXI4-Stream Interface – Receive
- Flow Control Interface (IEEE 802.3)
- Priority Flow Control (802.1Qbb)
- 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 2-2. See Connecting the Data Interfaces for details on connecting to the transmit interface.

*Table 2-2:* **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[0:0] | In | AXI4-Stream User signal used to signal explicit underrun. This is a vector of length 1 rather than a single bit to allow for future expansion. |
| tx_ifg_delay[7:0] | In | Configures Interframe Gap adjustment between packets. |
| 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 2-3. See Connecting the Data Interfaces for details on connecting to the receive interface.

*Table 2-3:* **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<br>0 indicates that a bad packet has been received.<br>1 indicates that a good packet has been received. |
| rx_axis_tlast | Out | AXI4-Stream signal from XGMAC indicating the end of a packet |

## Flow Control Interface (IEEE 802.3)

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 2-4.

*Table 2-4:* **Flow Control (IEE802.3) 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. |

## Priority Flow Control (802.1Qbb)

The Priority Flow Control (PFC) interface is used to initiate the transmission of PFC frames from the core. The ports associated with this interface are shown in Table 2-5. This interface is only present when priority-based flow control is enabled at the core customization stage.

*Table 2-5:* **Priority Flow Control Ports**

| Name | Direction | Description |
|---|---|---|
| rx_pfc_p0_tvalid | Output | Pause request to priority 0 RX FIFO |
| rx_pfc_p0_tready | Input | Pause acknowledge from priority 0 RX FIFO. The captured quanta only start to expire when this is asserted. If unused, this should be tied High. |
| rx_pfc_p1_tvalid | Output | Pause request to priority 1 RX FIFO |
| rx_pfc_p1_tready | Input | Pause acknowledge from priority 1 RX FIFO. The captured quanta only start to expire when this is asserted. If unused, this should be tied High. |
| rx_pfc_p2_tvalid | Output | Pause request to priority 2 RX FIFO |
| rx_pfc_p2_tready | Input | Pause acknowledge from priority 2 RX FIFO. The captured quanta only start to expire when this is asserted. If unused, this should be tied High. |
| rx_pfc_p3_tvalid | Output | Pause request to priority 3 RX FIFO |
| rx_pfc_p3_tready | Input | Pause acknowledge from priority 3 RX FIFO. The captured quanta only start to expire when this is asserted. If unused, this should be tied High. |
| rx_pfc_p4_tvalid | Output | Pause request to priority 4 RX FIFO |
| rx_pfc_p4_tready | Input | Pause acknowledge from priority 4 RX FIFO. The captured quanta only start to expire when this is asserted. If unused, this should be tied High. |
| rx_pfc_p5_tvalid | Output | Pause request to priority 5 RX FIFO |
| rx_pfc_p5_tready | Input | Pause acknowledge from priority 5 RX FIFO. The captured quanta only start to expire when this is asserted. If unused, this should be tied High. |
| rx_pfc_p6_tvalid | Output | Pause request to priority 6 RX FIFO |
| rx_pfc_p6_tready | Input | Pause acknowledge from priority 6 RX FIFO. The captured quanta only start to expire when this is asserted. If unused, this should be tied High. |
| rx_pfc_p7_tvalid | Output | Pause request to priority 7 RX FIFO |

*Table 2-5:* **Priority Flow Control Ports *(Cont'd)***

| Name | Direction | Description |
|---|---|---|
| rx_pfc_p7_tready | Input | Pause acknowledge from priority 7 RX FIFO. The captured quanta only start to expire when this is asserted. If unused, this should be tied High. |
| tx_pfc_p0_tvalid | Input | Pause request from priority FIFO. This results in a PFC frame at the next available point |
| tx_pfc_p1_tvalid | Input | Pause request from priority FIFO. This results in a PFC frame at the next available point |
| tx_pfc_p2_tvalid | Input | Pause request from priority FIFO. This results in a PFC frame at the next available point |
| tx_pfc_p3_tvalid | Input | Pause request from priority FIFO. This results in a PFC frame at the next available point |
| tx_pfc_p4_tvalid | Input | Pause request from priority FIFO. This results in a PFC frame at the next available point |
| tx_pfc_p5_tvalid | Input | Pause request from priority FIFO. This results in a PFC frame at the next available point |
| tx_pfc_p6_tvalid | Input | Pause request from priority FIFO. This results in a PFC frame at the next available point |
| tx_pfc_p7_tvalid | Input | Pause request from priority FIFO. This results in a PFC frame at the next available point |

## 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 2-6 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.

*Table 2-6:* **PHY Interface Port Descriptions**

| Name | Direction | Description |
|---|---|---|
| xgmii_txd[63 or 31:0] | Out | Transmit data to PHY |
| xgmii_txc[7 or 3:0] | Out | Transmit control to PHY |
| xgmii_rxd[63 or 31:0] | In | Received data from PHY |
| xgmii_rxc[7 or 3:0] | 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 2-7 defines the ports associated with the Management Interface.

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

| Name | Direction | Description |
|---|---|---|
| s_axi_aclk | In | AXI4-Lite clock. Range between 10 MHz and 300 MHz |
| s_axi_aresetn | In | Asynchronous active-Low reset |
| s_axi_awaddr[10:0] | 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[10:0] | 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 2-8 describes the configuration and Status signals. See Connecting the Management Interface for details on this signal, including a breakdown of the configuration and status vector bits.

*Table 2-8:* **Configuration and Status Signals**

| Name | Direction | Description |
|---|---|---|
| tx_configuration_vector[79:0][(1)] | In | Configuration signals for the Transmitter |
| rx_configuration_vector[79:0][(2)] | In | Configuration signals for the Receiver |
| status_vector[1:0] | Out | Status signals for the core |

1. When PFC is enabled tx_configuration_vector has a bus width of 367:0
2. When PFC is enabled rx_configuration_vector has a bus width of 95:0

## MDIO Interface Signals

The MDIO Interface signals are shown in Table 2-9. See Connecting the Management Interface for details on the use of this interface.

*Table 2-9:* **MDIO Interface Port Descriptions**

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

## Interrupt Signal

The Interrupt output signal is shown in Table 2-10. See Interrupt Output for more details.

*Table 2-10:* **Interrupt Output Port Description**

| Name | Direction | Description |
|---|---|---|
| xgmacint | Out | Interrupt output. |

## Statistic Vector Signals

In addition to the statistic counters described in Statistics Counters and Register Space, there are two statistics vector outputs on the core that are used to signal the core state. The signals are shown in Table 2-11. The contents of the vectors themselves are described in Connecting the Management Interface.

*Table 2-11:* **Statistic Vector Signals**

| Name | Direction | Description |
|---|---|---|
| tx_statistics_vector[25:0][1] | Out | Aggregated statistics flags for transmitted frame. |
| tx_statistics_valid | Out | Valid strobe for `tx_statistics_vector`. See Transmit Statistics Vector for more information. |
| rx_statistics_vector[29:0][2] | Out | Aggregated statistics flags for received frames. |
| rx_statistics_valid | Out | Valid strobe for `rx_statistics_vector`. See Receive Statistics Vector for more information. |

1. When PFC is enabled tx_statistics_vector has a bus width of 26:0
2. When PFC is enabled rx_statistics_vector has a bus width of 30:0

## Clocking and Reset Signals

Included in the example design top-level sources are circuits for clock and reset management. These can include Mixed-Mode Clock Managers (MMCMs), clock buffers, and reset synchronizers.

Table 2-12 through Table 2-14 show the ports on the core associated with system clocks and resets.

*Table 2-12:* **Clock, Clock Management, and Reset Signals – Internal Physical Interface**

| Name | Direction | Description |
|---|---|---|
| reset | In | Set to 1 to reset core. Treated as an asynchronous input by the core. |
| tx_clk0 | In | System clock for transmit side of core; derived from `gtx_clk` in example design. |
| tx_dcm_locked | In | Status flag from DCM/MMCM. |
| rx_clk0 | In | System clock for receive side of core; derived from `xgmii_rx_clk` in example design. |
| rx_dcm_locked | In | Status flag from DCM/MMCM. |

*Table 2-13:* **Clock, Clock Management, and Reset Signals – External XGMII Interface with Shared Logic in Example Design**

| Name | Direction | Description |
|---|---|---|
| reset | In | Set to 1 to reset core. Treated as an asynchronous input by the core. |
| tx_clk0 | In | System clock for transmit side of core, derived from `gtx_clk` in example design. |
| tx_clk90 | In | 90° phase shift of system clock, derived from `gtx_clk` in example design. |
| tx_dcm_locked | In | Status flag from DCM/MMCM. |
| xgmii_tx_clk | Out | XGMII ODDR output clock sent to external PHY. |
| xgmii_rx_clk | In | Received clock from the connected PHY. |
| rx_clk_out | Out | System clock for the receive logic. |
| rx_dcm_locked_out | Out | Status signal from clock management block in core that performs clock/data alignment on the receive path. |

*Table 2-14:* **Clock, Clock Management, and Reset Signals – External XGMII Interface with Shared Logic in Core**

| Name | Direction | Description |
|---|---|---|
| reset | In | Set to 1 to reset core. Treated as an asynchronous input by the core. |
| gtx_clk | In | 156.25 MHz system clock input. |
| tx_clk0_out | Out | System clock for transmit side of core, derived from `gtx_clk` input. |
| tx_clk90_out | Out | 90° phase shift of transmit system clock, derived from `gtx_clk` input. |
| tx_dcm_locked_out | Out | Status flag from DCM/MMCM. |
| xgmii_tx_clk | Out | XGMII ODDR output clock sent to external PHY. |
| xgmii_rx_clk | In | Received clock from the connected PHY. |
| rx_clk_out | Out | System clock for the receive logic. |
| rx_dcm_locked_out | Out | Status signal from clock management block in core that performs clock/data alignment on the receive path. |

# 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 2-15. 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 2-15:* **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 1,023 bytes in length inclusive. |
| 0x244 | 512–1023 byte frames received OK (MSW) | |

*Table 2-15:* **Statistics Counters *(Cont'd)***

| Address (Hex) | Name | Description |
|---|---|---|
| 0x248 | 1024 – MaxFrameSize byte frames received OK (LSW) | A count of error-free frames received that were between 1,024 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) | |
| 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 1,023 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 1,024 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) | |

*Table 2-15:* **Statistics Counters *(Cont'd)***

| Address (Hex) | Name | Description |
|---|---|---|
| 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.<br>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) | |
| 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) | |

*Table 2-15:* **Statistics Counters *(Cont'd)***

| Address (Hex) | Name | Description |
|---|---|---|
| 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/user logic 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 2-16.

*Table 2-16:* **Configuration Registers**

| Address (Hex) | Description |
|---|---|
| 0x400 | Receiver Configuration Word 0 |
| 0x404 | Receiver Configuration Word 1 |
| 0x408 | Transmitter Configuration Word |
| 0x40C | Flow Control Configuration Register |
| 0x410 | Reconciliation Sublayer Configuration Word |
| 0x414 | Receiver MTU Configuration Word |
| 0x418 | Transmitter MTU Configuration Word |
| 0x480 | Priority 0 Quanta register |
| 0x484 | Priority 1 Quanta register |
| 0x488 | Priority 2 Quanta register |
| 0x48c | Priority 3 Quanta register |
| 0x490 | Priority 4 Quanta register |
| 0x494 | Priority 5 Quanta register |
| 0x498 | Priority 6 Quanta register |
| 0x49c | Priority 7 Quanta register |
| 0x4a0 | Legacy Pause Refresh Register |
| 0x4F8 | Version Register (read-only) |
| 0x4FC | Capability Register (read-only) |

The contents of each configuration register are shown in Tables 2-17 through Table 2-27.

*Table 2-17:* **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.<br>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 2-18:* **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 55. 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. |
| 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 57.<br>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 2-17. |

*Table 2-19:* **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 44. 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 16,384 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 49.<br>When this bit is set to 0, the transmitter outputs the minimum Inter Frame Gap.<br>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.<br>When this bit is set to 0, the core always stretches the IFG to maintain start alignment.<br>This bit is cleared and has no effect if Interframe Gap Adjust is enabled. |
| 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 2-20:* **Flow Control Configuration Register**

| 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. This mode should not be enabled at the same time as PFC (bit 26). |

*Table 2-20:* **Flow Control Configuration Register *(Cont'd)***

| Bits | Default Value | Description |
|---|---|---|
| 29 | 1 | **Flow Control Enable (RX)**. When this bit is 1, received flow control frames inhibit the transmitter operation as described in Receiving an IEEE 802.3 Pause Frame. When this bit is 0, received flow control frames are always passed up to the client. This mode should not be enabled at the same time as PFC (bit 25). |
| 28:27 | N/A | Reserved |
| 26 | 0 | **Priority pause flow control enable (TX)**. Only present when the core has been generated with PFC support. When this bit is 1, asserting an enabled TX PFC tvalid signal results in a PFC frame being sent from the transmitter. When this bit is 0,the TX PFC tvalid inputs are ignored. This mode should not be enabled at the same time as Flow Control (TX) (bit 30). |
| 25 | 0 | **Priority pause flow control enable (RX)**. Only present when the core has been generated with PFC support. When this bit is 1, received PFC frames assert the relevant, enabled RX PFC tvalid outputs as described in Receiving a PFC Frame. When this bit is 0, received PFC frames are ignored and passed to the client. This mode should not be enabled at the same time as Flow Control (RX) (bit 29). |
| 24:21 | N/A | Reserved |
| 20 | 1 | **TX Auto XON**. Only present when the core has been generated with PFC support - this bit defaults to 0 if PFC is not supported. Send a flow control or PFC frame with the relevant quanta set to zero (XON frame) when the relevant, enabled pause request is dropped |
| 19:16 | N/A | Reserved |
| 15 | 1 | **TX Priority 7 pause enable**. Only present when the core has been generated with PFC support- this bit defaults to 0 if PFC is not supported. When this bit is 1, and TX PFC is enabled, assertion or deassertion of the TX PFC tvalid signal results in a PFC frame being transmitted. When this bit is 0 tx_pfc_p7_tvalid is ignored. |
| 14 | 1 | **TX Priority 6 pause enable**. Only present when the core has been generated with PFC support- this bit defaults to 0 if PFC is not supported. Equivalent function to bit 15 but relevant to tx_pfc_p6_tvalid. |
| 13 | 1 | **TX Priority 5 pause enable**. Only present when the core has been generated with PFC support- this bit defaults to 0 if PFC is not supported. Equivalent function to bit 15 but relevant to tx_pfc_p5_tvalid. |
| 12 | 1 | **TX Priority 4 pause enable**. Only present when the core has been generated with PFC support- this bit defaults to 0 if PFC is not supported. Equivalent function to bit 15 but relevant to tx_pfc_p4_tvalid. |
| 11 | 1 | **TX Priority 3 pause enable**. Only present when the core has been generated with PFC support- this bit defaults to 0 if PFC is not supported. Equivalent function to bit 15 but relevant to tx_pfc_p3_tvalid. |
| 10 | 1 | **TX Priority 2 pause enable**. Only present when the core has been generated with PFC support- this bit defaults to 0 if PFC is not supported. Equivalent function to bit 15 but relevant to tx_pfc_p2_tvalid. |
| 9 | 1 | **TX Priority 1 pause enable**. Only present when the core has been generated with PFC support- this bit defaults to 0 if PFC is not supported. Equivalent function to bit 15 but relevant to tx_pfc_p1_tvalid. |

*Table 2-20:* **Flow Control Configuration Register *(Cont'd)***

| Bits | Default Value | Description |
|---|---|---|
| 8 | 1 | **TX Priority 0 pause enable**. Only present when the core has been generated with PFC support- this bit defaults to 0 if PFC is not supported. Equivalent function to bit 15 but relevant to tx_pfc_p0_tvalid. |
| 7 | 1 | **RX Priority 7 pause enable**. Only present when the core has been generated with PFC support- this bit defaults to 0 if PFC is not supported. When this bit is 1, and RX PFC is enabled, reception of a PFC frame with a valid quanta for priority 7 is processed as described in Receiving a PFC Frame When this bit is 0, the `rx_pfc_p7_tvalid` remains at 0. |
| 6 | 1 | **RX Priority 6 pause enable**. Only present when the core has been generated with PFC support- this bit defaults to 0 if PFC is not supported. Equivalent function to bit 7 but relevant to rx_pfc_p6_tvalid. |
| 5 | 1 | **RX Priority 5 pause enable**. Only present when the core has been generated with PFC support- this bit defaults to 0 if PFC is not supported. Equivalent function to bit 7 but relevant to rx_pfc_p5_tvalid. |
| 4 | 1 | **RX Priority 4 pause enable**. Only present when the core has been generated with PFC support- this bit defaults to 0 if PFC is not supported. Equivalent function to bit 7 but relevant to rx_pfc_p4_tvalid. |
| 3 | 1 | **RX Priority 3 pause enable**. Only present when the core has been generated with PFC support- this bit defaults to 0 if PFC is not supported. Equivalent function to bit 7 but relevant to rx_pfc_p3_tvalid. |
| 2 | 1 | **RX Priority 2 pause enable**. Only present when the core has been generated with PFC support- this bit defaults to 0 if PFC is not supported. Equivalent function to bit 7 but relevant to rx_pfc_p2_tvalid. |
| 1 | 1 | **RX Priority 1 pause enable**. Only present when the core has been generated with PFC support- this bit defaults to 0 if PFC is not supported. Equivalent function to bit 7 but relevant to rx_pfc_p1_tvalid. |
| 0 | 1 | **RX Priority 0 pause enable**. Only present when the core has been generated with PFC support- this bit defaults to 0 if PFC is not supported. Equivalent function to bit 7 but relevant to rx_pfc_p0_tvalid. |

*Table 2-21:* **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 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. |

*Table 2-21:* **Reconciliation Sublayer Configuration Word *(Cont'd)***

| Bits | Default Value | Description |
|---|---|---|
| 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.<br>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 2-22:* **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 57 when RX MTU Enable is set to 1. Only values of 1,518 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 2-23:* **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 49 when TX MTU Enable is set to 1. Only values of 1,518 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 2-24:* **Per Priority Quanta/Refresh Register (0x480/0x49C)**

| Bits | Default | Description |
|---|---|---|
| 31:16 | 0xff00 | **Pause Quanta refresh value**. This register is only present when PFC is enabled at the core customization time. When enabled, this register controls how frequently a PFC quanta is refreshed by the transmission of a new PFC frame. When a refresh occurs, all currently active (TX PFC tvalid is High and enabled) priorities are refreshed. |
| 15:0 | 0xFFFF | **Pause Quanta value.** This register is only present when PFC is enabled at core customization time. When enabled, this register sets the quanta value to be inserted in the PFC frame for this priority. |

1. This register is repeated for the eight priorities, priority 0 to priority 7.
2. These registers only exist when the core is generated with PFC support.

*Table 2-25:* **Legacy Pause Refresh Register**

| Bits | Default | Description |
|---|---|---|
| 31:16 | 0xff00 | **Pause Quanta refresh value**. This register is only present when PFC is enabled at the core customization time. When PFC is supported, the 802.3 pause request can also support XON/XOFF Extended Functionality. This controls the frequency of the automatic pause refresh. |
| 15:0 | 0x0 | Reserved |

1. These registers only exist when the core is generated with PFC support.

*Table 2-26:* **Version Register**

| Bits | Default Value | Description |
|---|---|---|
| 31:24 | 0x0D | **Major Revision**. This field indicates the major revision of the core. |
| 23:16 | 0x01 | **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, etc.) |

*Table 2-27:* **Capability Register**

| Bits | Default Value | Description |
|---|---|---|
| 31:9 | N/A | Reserved |
| 16 | 0 | **PFC Support.** This bit indicates that the core has been generated with PFC support. |
| 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. |
| 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 2-28.

*Table 2-28:* **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 2-29 through Table 2-32.

*Table 2-29:* **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 62. |

*Table 2-30:* **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 3-30 through Figure 3-33. |
| 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. |
| 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 2-31:* **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 2-32:* **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 facilitate designing with the 10-Gigabit Ethernet MAC core. It contains these sections:

- General Design Guidelines
- Shared Logic
- Clocking
- Resets
- Protocol Description
- Connecting the Data Interfaces
- Connecting the Management Interface
- Using IEEE 802.3 Flow Control
- Special Design Considerations

## 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 the Vivado® IP catalog 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, see Chapter 5, 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 XDC 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 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 Vivado IP catalog when the core is generated. For more information, see Chapter 4, Design Flow Steps.

# Shared Logic

The core supported up to version 12.0 and the RTL core hierarchy was fixed. This resulted in some difficulty because shareable clocking and reset logic needed to be extracted from the core example design for use with a single instance, or multiple instances of the core.

Shared Logic is a new feature that provides a more flexible architecture that works both as a standalone core and as a part of a larger design with one or more core instances. This minimizes the amount of HDL modifications required, but at the same time retains the flexibility to address more uses of the core.

The new level of hierarchy is called `<component_name>_support`. Figure 3-1 and Figure 3-2 show two hierarchies where the shared logic block is contained either in the core or in the example design. In these figures, `<component_name>` is the name of the generated core. The difference between the two hierarchies is the boundary of the core. It is controlled using the Shared Logic option in the Vivado Integrated Design Environment (IDE) (see Figure 4-1 in Chapter 4, Design Flow Steps).

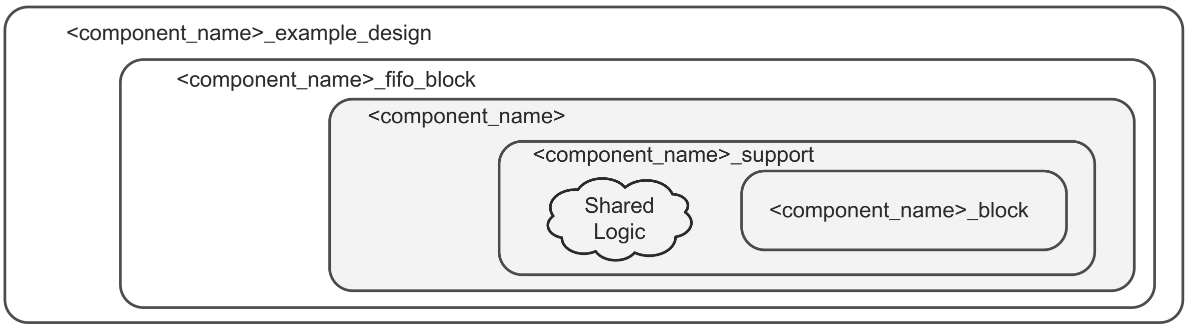


*Figure 3-1:* **Shared Logic in Core**

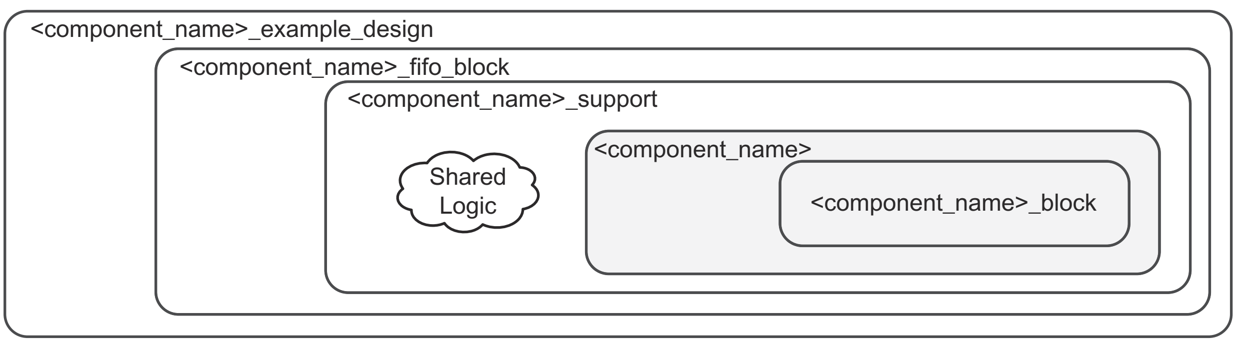


*Figure 3-2:* **Shared Logic in Example Design**

The contents of the shared logic depend upon the physical interface and the target device.

## XGMII

Figure 3-3 shows that the transmit side MMCM and associated clock buffers are instantiated at the `<component_name>_support` level. If the "Shared logic is in core" option is not selected, ensure an MMCM or similar clock management structure is instantiated elsewhere in your design to provide the necessary clocks.

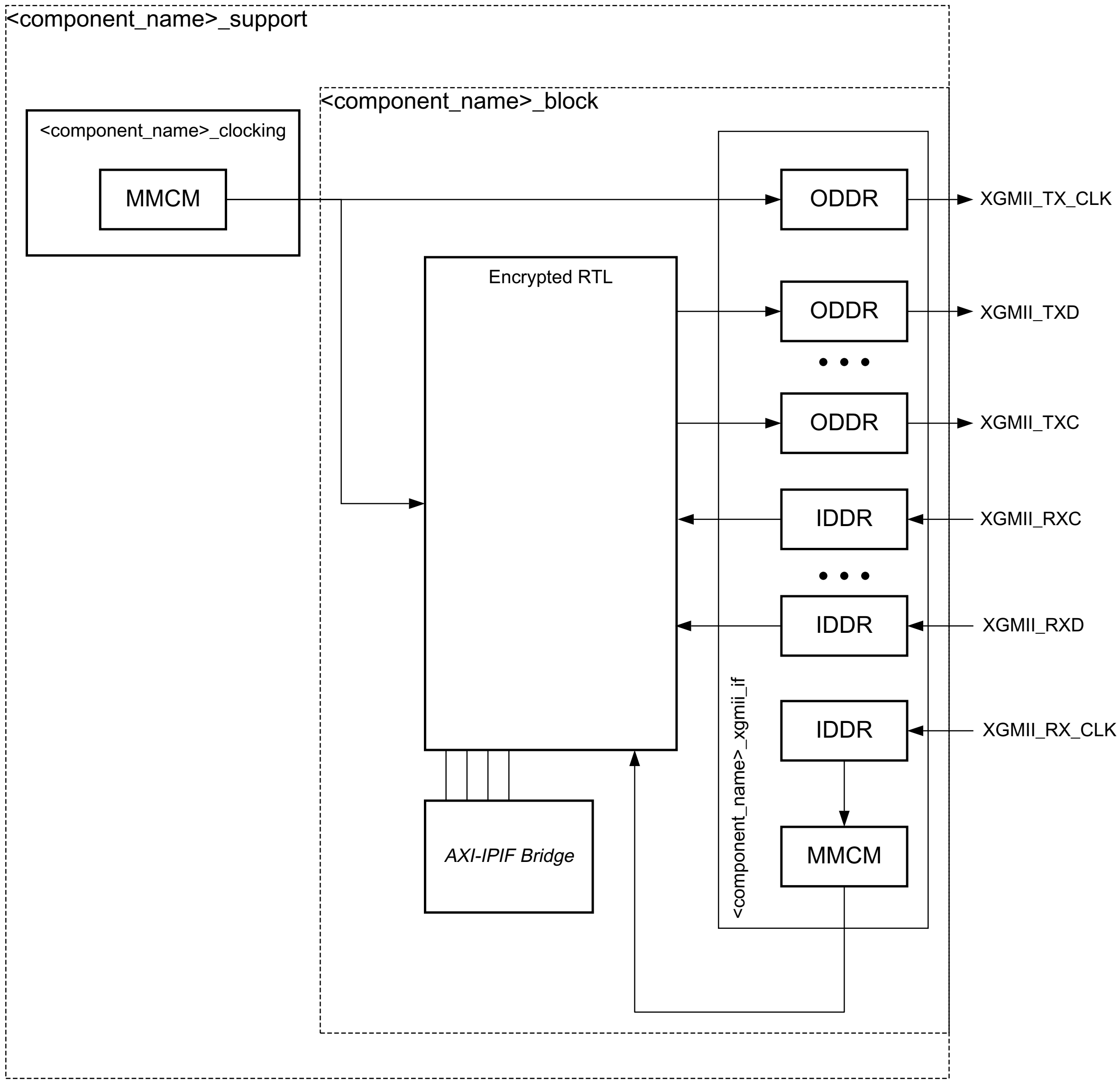


*Figure 3-3:* **XGMII Shared Logic**

## Internal

If the core is generated with the Internal physical interface, the `<component_name>_support` level of hierarchy becomes a logicless wrapper file. In this case, there is no shareable logic between instances.

## Clocking

Figure 3-4 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 91.

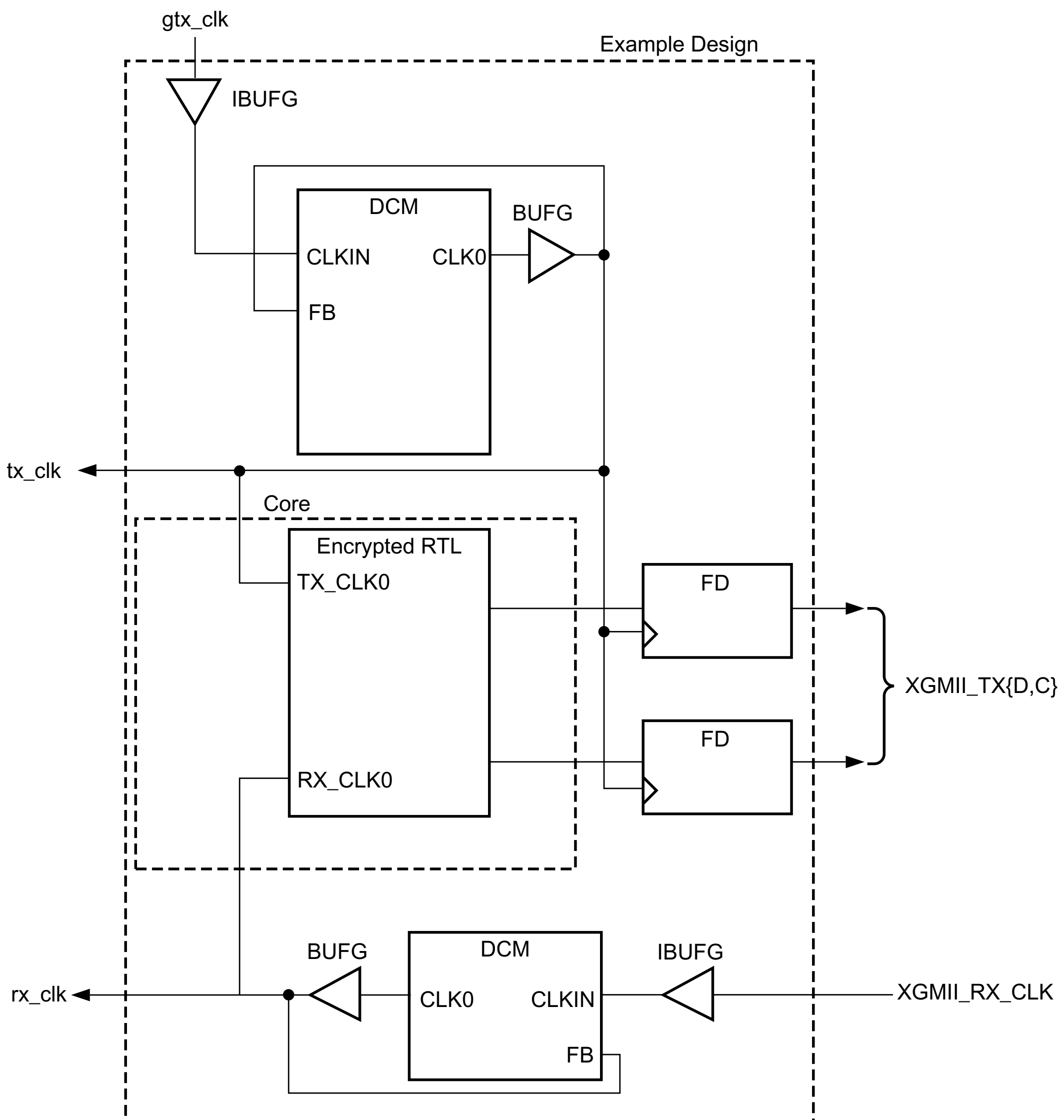


*Figure 3-4:* **Clocking Logic for the Ethernet MAC Internal Interface Option**

# 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 3-5.

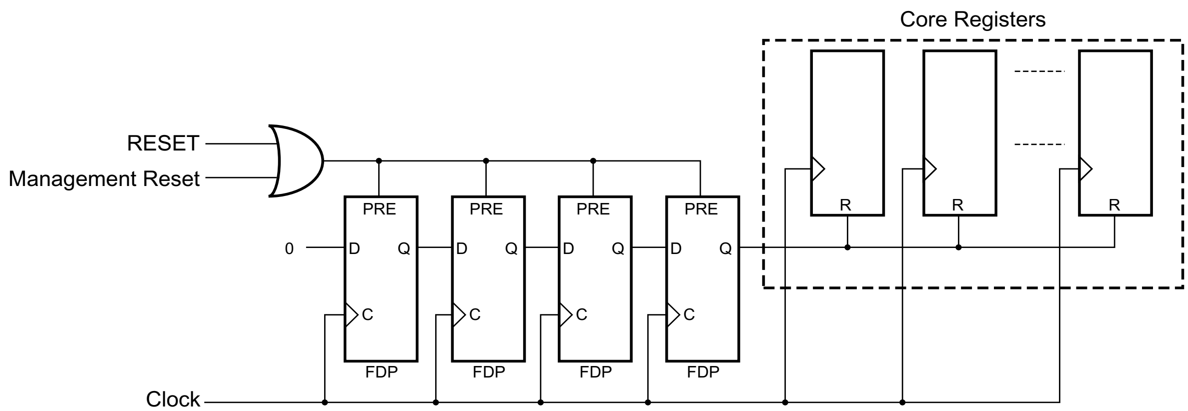


*Figure 3-5:* **Reset Circuit for a Single Clock/Reset Domain**

# 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 3-6 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 3-6 also shows where the supported physical interfaces fit into the architecture.

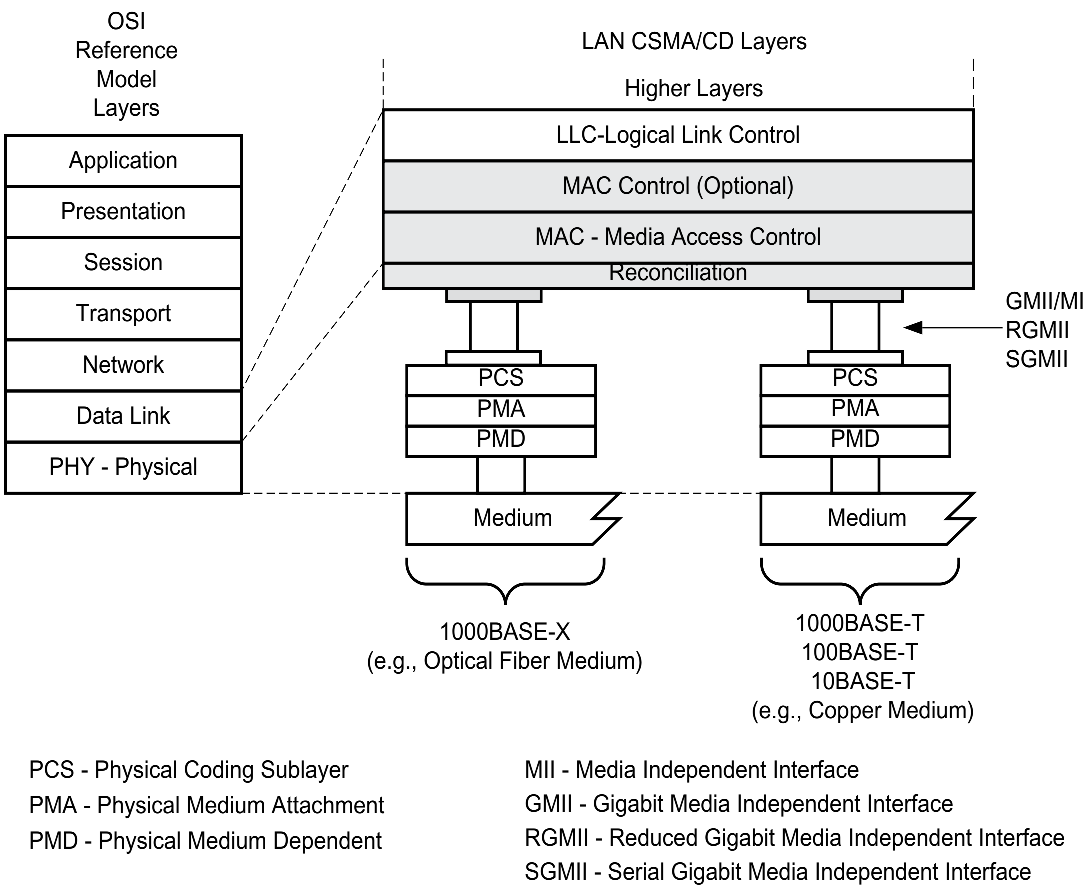


*Figure 3-6:* **IEEE Std 802.3-2008 Ethernet Model**

### MAC and MAC CONTROL Sublayer

The Ethernet MAC is defined in *IEEE Std 802.3-2008* [Ref 1], 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* [Ref 1], 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 3-7, 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.

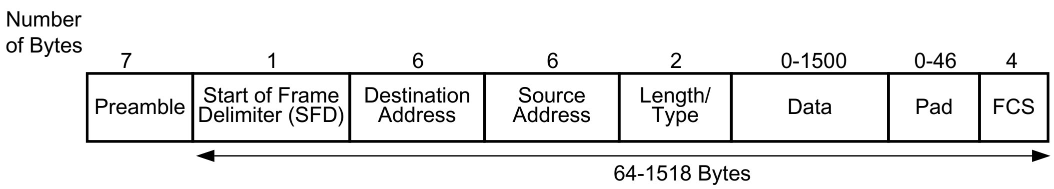


*Figure 3-7:* **Standard Ethernet Frame Format**

The Ethernet MAC can also accept Virtual LAN (VLAN) frames. The VLAN frame format is shown in Figure 3-8. If the frame is a VLAN type frame and the Ethernet MAC configuration registers are set, the core can accept up to four additional bytes above the usual maximum frame length.

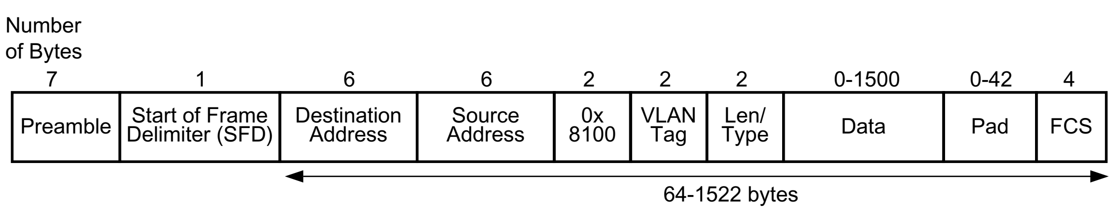


*Figure 3-8:* **Ethernet VLAN Frame Format**

Ethernet PAUSE/flow control frames and optionally 802.1Qbb priority-based flow control frames can be transmitted and received by the Ethernet MAC. Figure 3-37, page 77 shows how an 802.3 PAUSE/flow control frame differs from the standard Ethernet frame format and Figure 3-42 shows how an IEEE 802.1Qbb priority-based 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 you.

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 you.

### 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 you 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* [Ref 1]. A value of 1,536 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 you. 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 you, the FCS can be either inserted by the Ethernet MAC or provided by you, 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 passed to you, unless the Ethernet MAC is configured to pass the FCS field on to you.

#### 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. 3.2.9:

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

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 you, 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 you that it has received a bad frame. The FCS field can either be passed on to you 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.

# Connecting the Data Interfaces

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

## Transmit AXI4-Stream Interface

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 Vivado IP catalog. Table 3-1 defines the signals.

When connecting this interface in IP Integrator, the signals of Table 3-1 (with the exception of `tx_axis_aresetn`) are shown and can be connected as a single bus. This bus is called `s_axis_tx`.

*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[0:0] | 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 3-9. 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 3-9, 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.

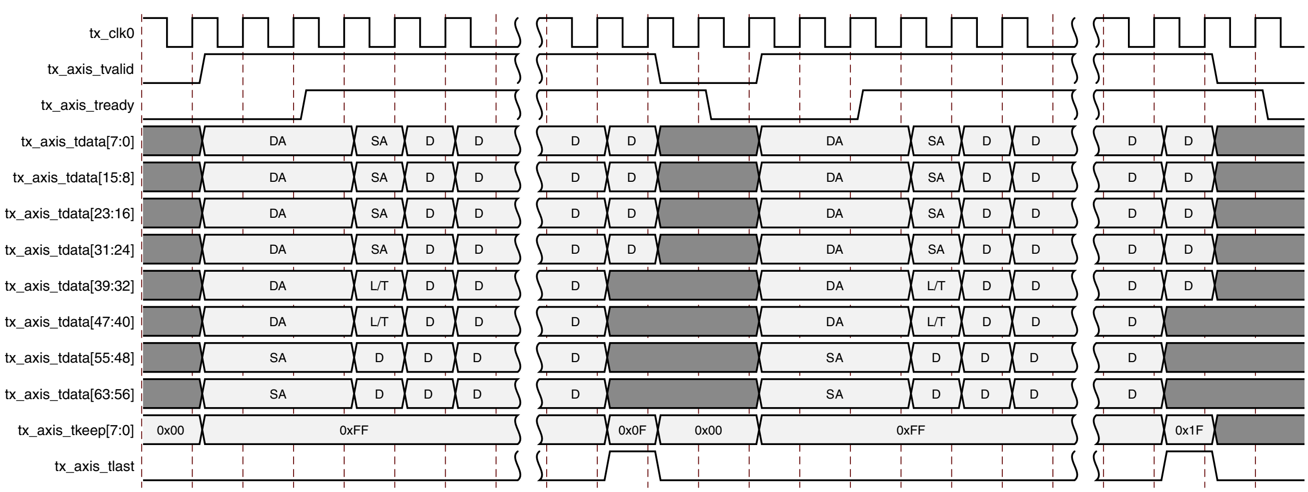


*Figure 3-9:* **Frame Transmission**

### 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 etc.

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 3-10. 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.

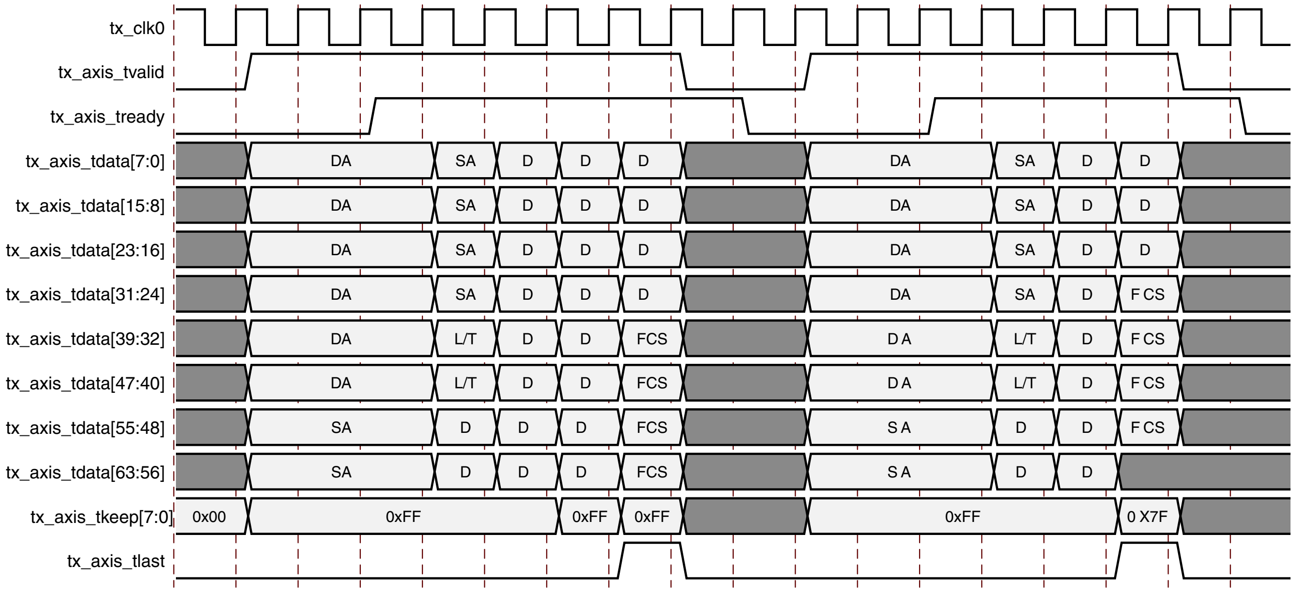


*Figure 3-10:* **Transmission with In-Band FCS Passing**

### *Aborting a Transmission*

The aborted transfer of a packet on the client interface is called an underrun. This can happen, for example, 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 3-11)
   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 3-12)

Figure 3-11 and Figure 3-12 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.

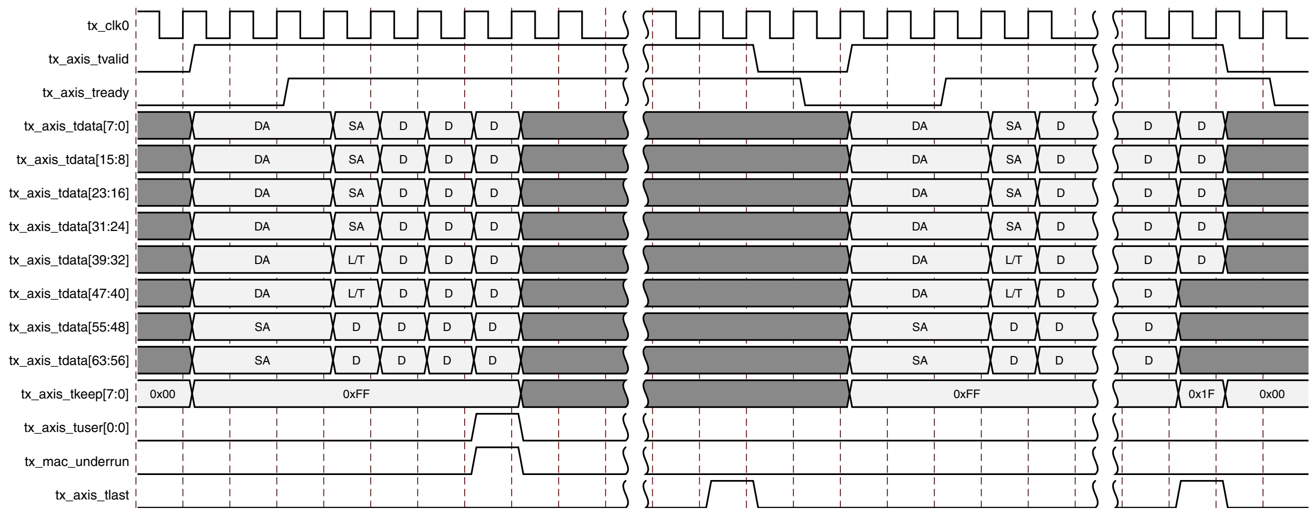


*Figure 3-11:* **Frame Transfer Abort with tx_axis_tuser asserted**

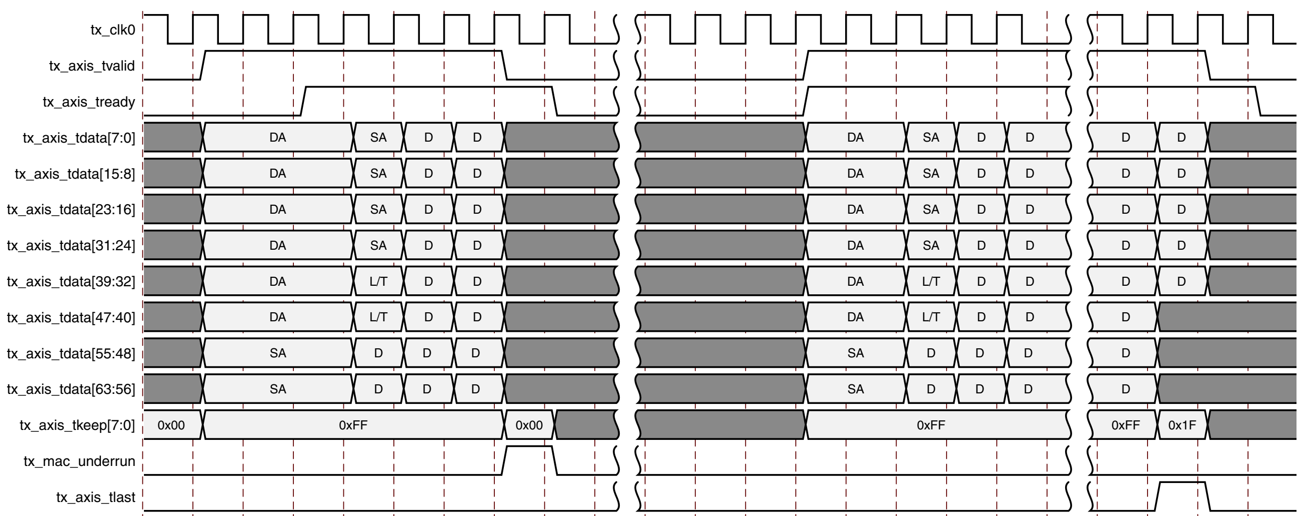


*Figure 3-12:* **Frame Transfer Abort with tx_axis_tvalid deasserted**

***Note:*** Aborting a frame transfer using the mechanism shown in Figure 3-12 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 3-11.

### *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.

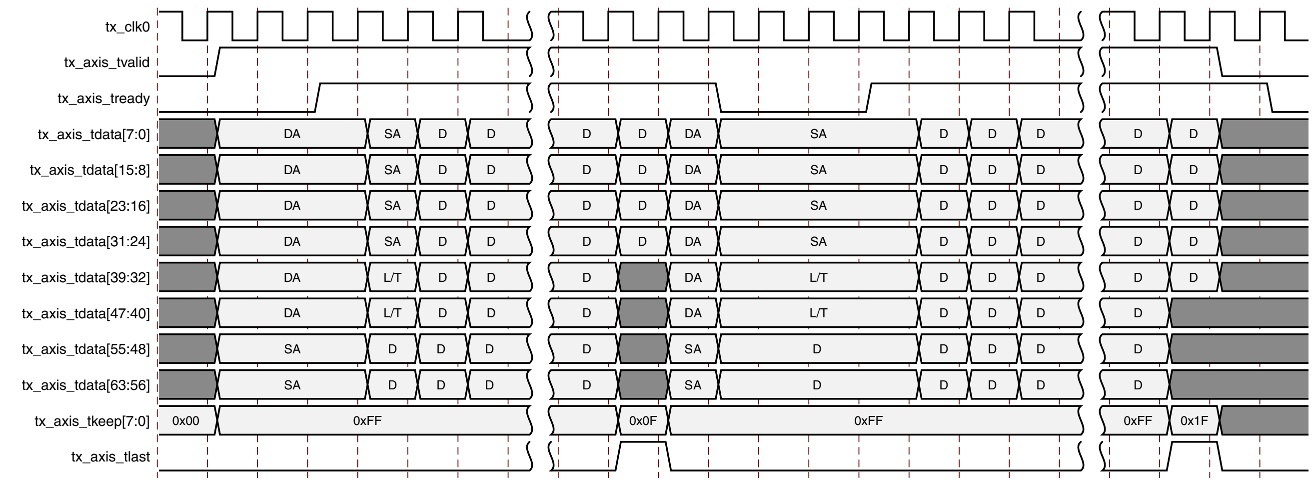


*Figure 3-13:* **Back-to-Back Continuous Transfer on Transmit Client Interface**

## *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), 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 3-14 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 3-15 illustrates the transmission of a custom preamble when `tx_axis_tvalid` remains asserted after `tx_axis_tlast` is asserted.

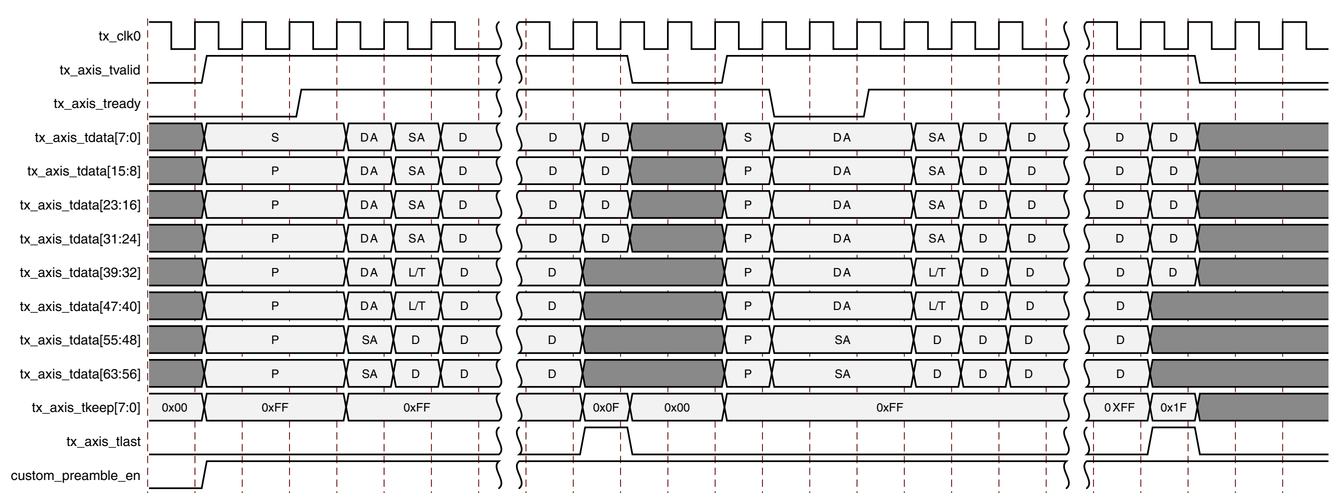


*Figure 3-14:* **Transmission of Custom Preamble in the Non-Continuous Case**

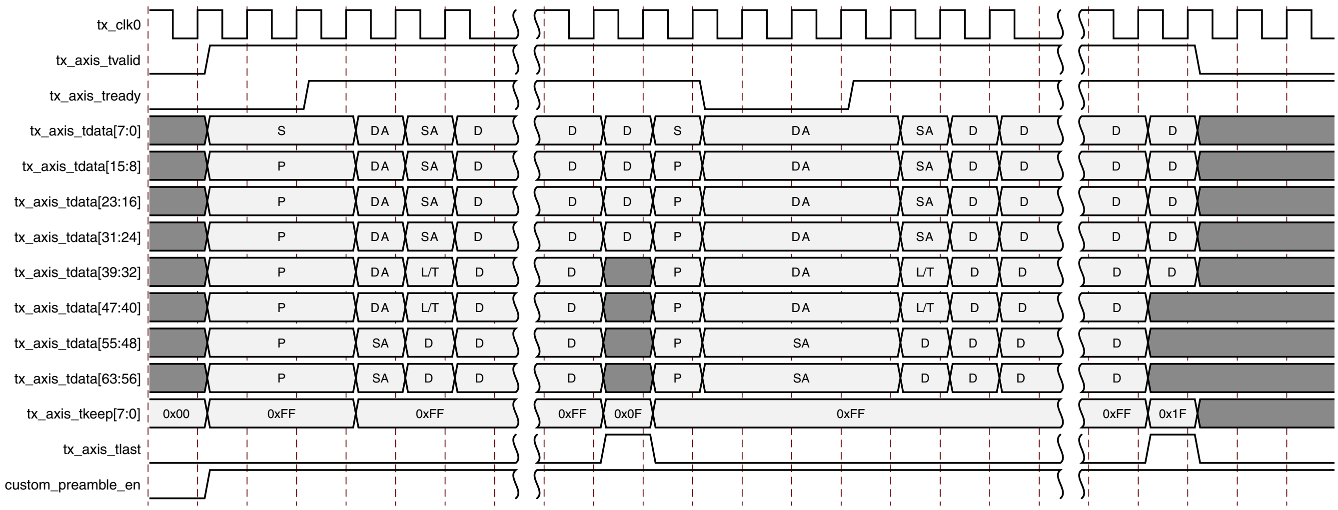


*Figure 3-15:* **Transmission of Custom Preamble in the Continuous Case**

The Ethernet MAC core substitutes the IEEE standard preamble with that supplied by the client logic. Figure 3-16 shows the transmission of a frame with custom preamble (P1 to P7) at the XGMII interface.

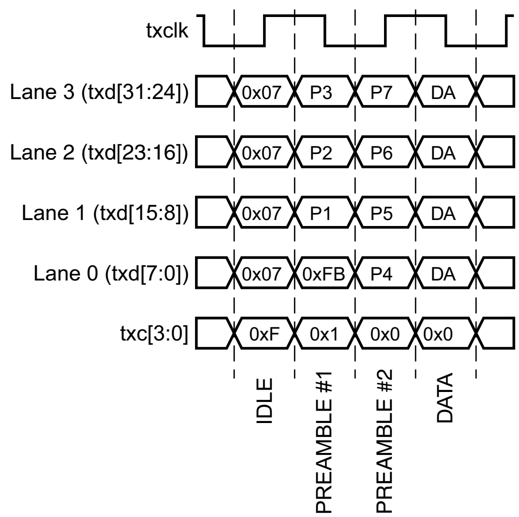


*Figure 3-16:* **XGMII Frame Transmission of Custom Preamble**

## *VLAN Tagged Frames*

Transmission of a VLAN tagged frame (if enabled) is shown in Figure 3-17. 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].

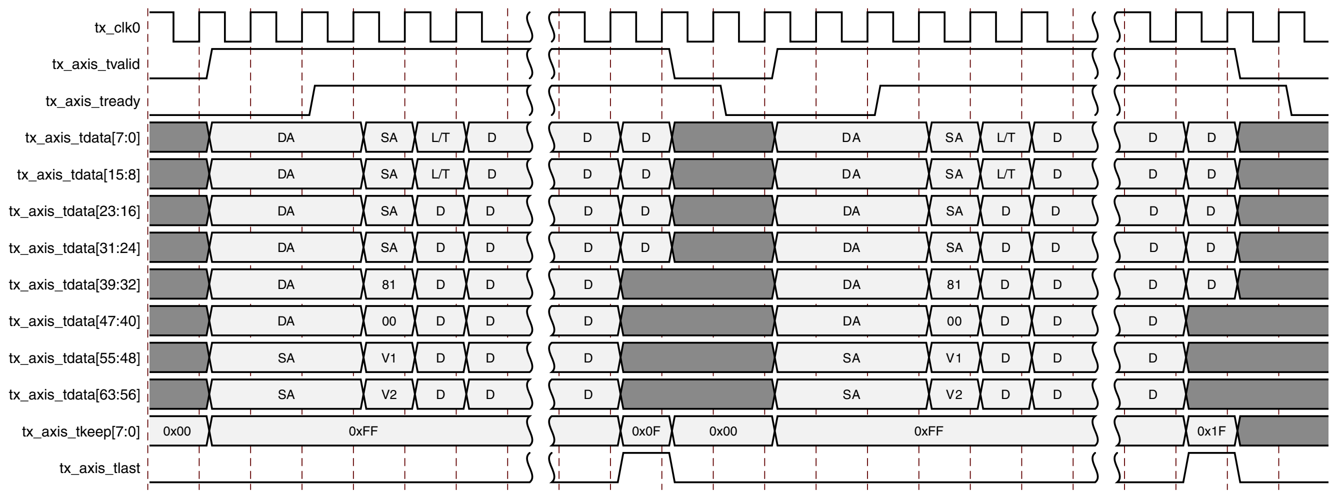


*Figure 3-17:* **VLAN Tagged Frame Transmission**

### *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 ≥ 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.

***Note:*** There are interactions between the configuration bits affecting frame length handling that you 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), 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 3-18 shows the Ethernet MAC operating in this mode.

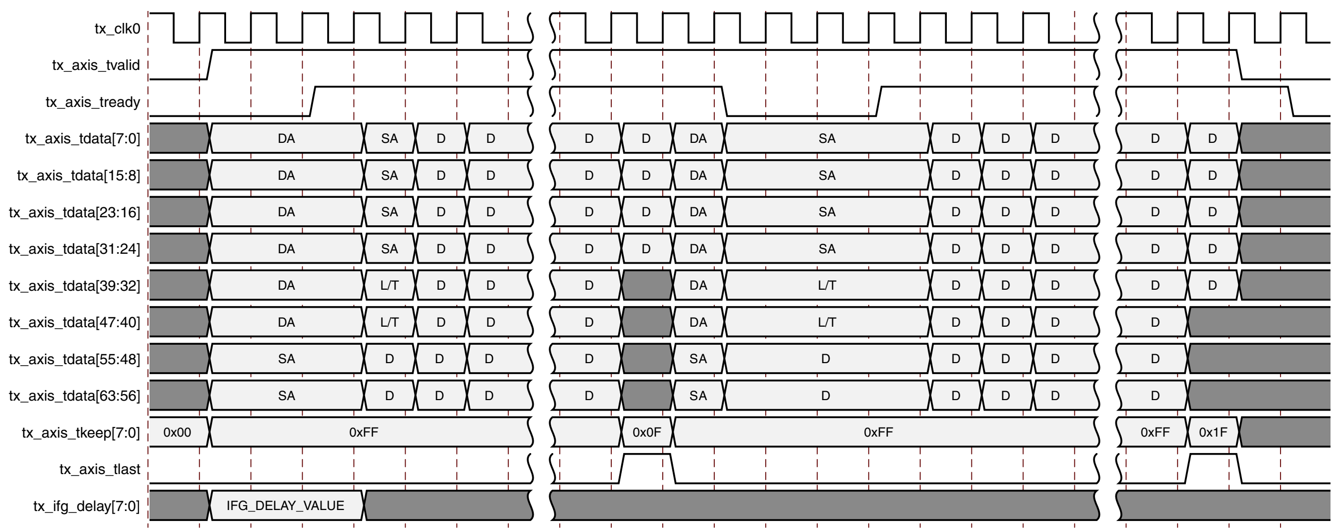


*Figure 3-18:* **Interframe Gap Adjustment**

### *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 IFG adjustment with DIC, `axis_tx_tvalid` must be maintained continuously High.

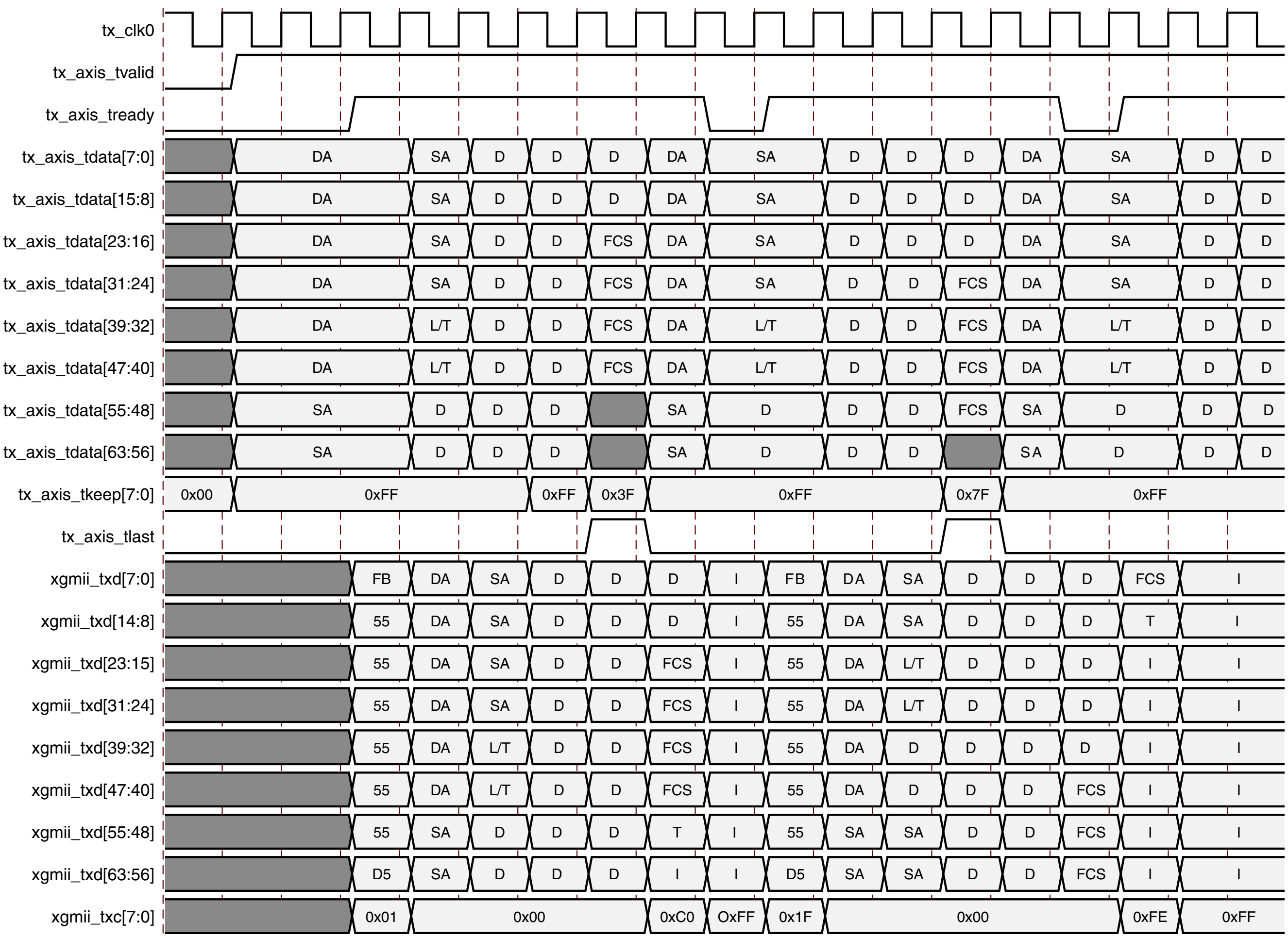


*Figure 3-19:* **Back-to-Back Continuous Transfer on Transmit XGMII Interface**

### *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). 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.

## 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.

When connecting this interface in IP Integrator, the signals of Table 3-4 (with the exception of `rx_axis_aresetn`) are shown and can be connected as a single bus. This is called `m_axis_rx`.

*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<br>0 indicates a bad packet has been received.<br>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 |
| 5 | 47:40 |
| 6 | 55:58 |
| 7 | 63:56 |

The timing of a normal inbound frame transfer is represented in Figure 3-20. 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_tlast` and `rx_axis_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 `rx_axis_tlast` asserted for one cycle.

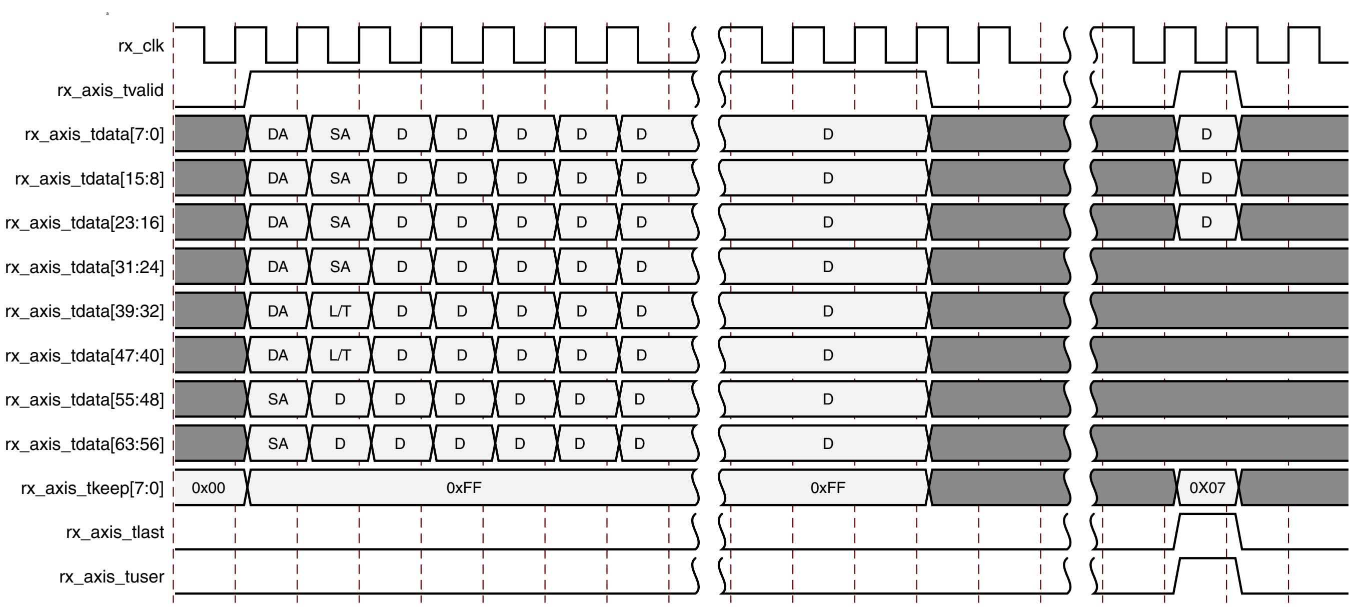


*Figure 3-20:* **Reception of a Good Frame**

### *Timing for a Good or a Bad Frame*

As can be seen in the Figure 3-20, 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`.

### 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 3-21. 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. This frame is only marked as an error because it has been used by the MAC flow control logic and has now served its purpose.
- A valid Priority Flow Control (PFC) frame, addressed to the Ethernet MAC, is received when PFC is enabled. This frame is only marked as an error because it has been used by the MAC PFC logic and has now served its purpose.

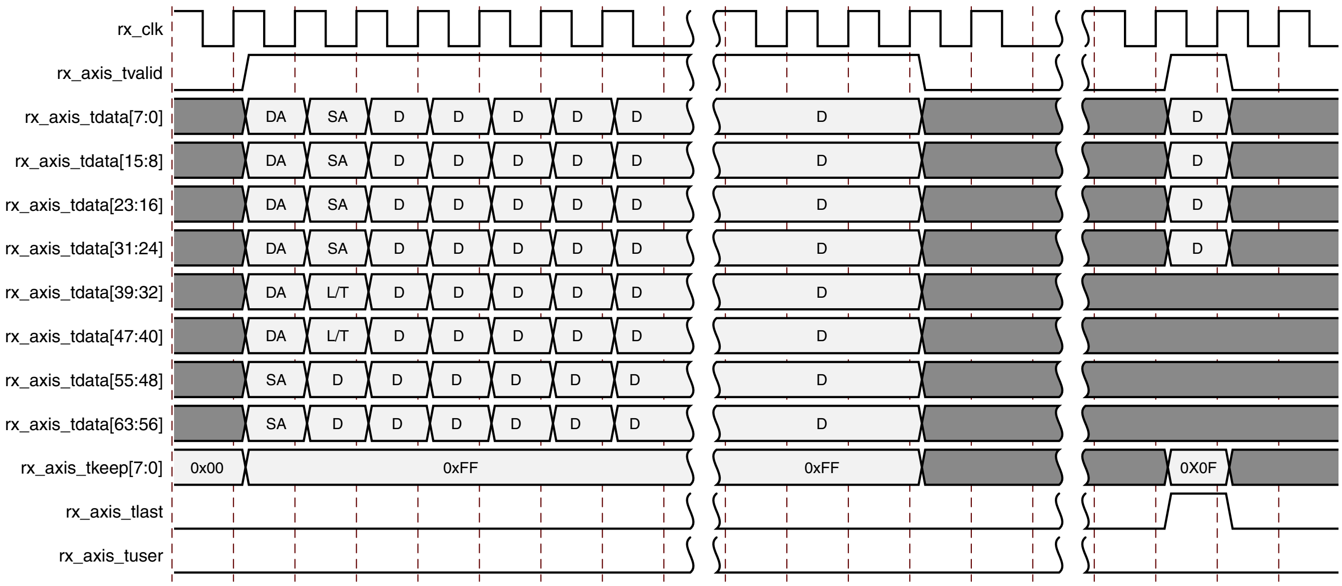


*Figure 3-21:* **Frame Reception with Error**

### *Reception with In-Band FCS Passing*

Figure 3-22 illustrates the Ethernet MAC core configured to pass the FCS field to the client (see Configuration Registers, page 22). 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.

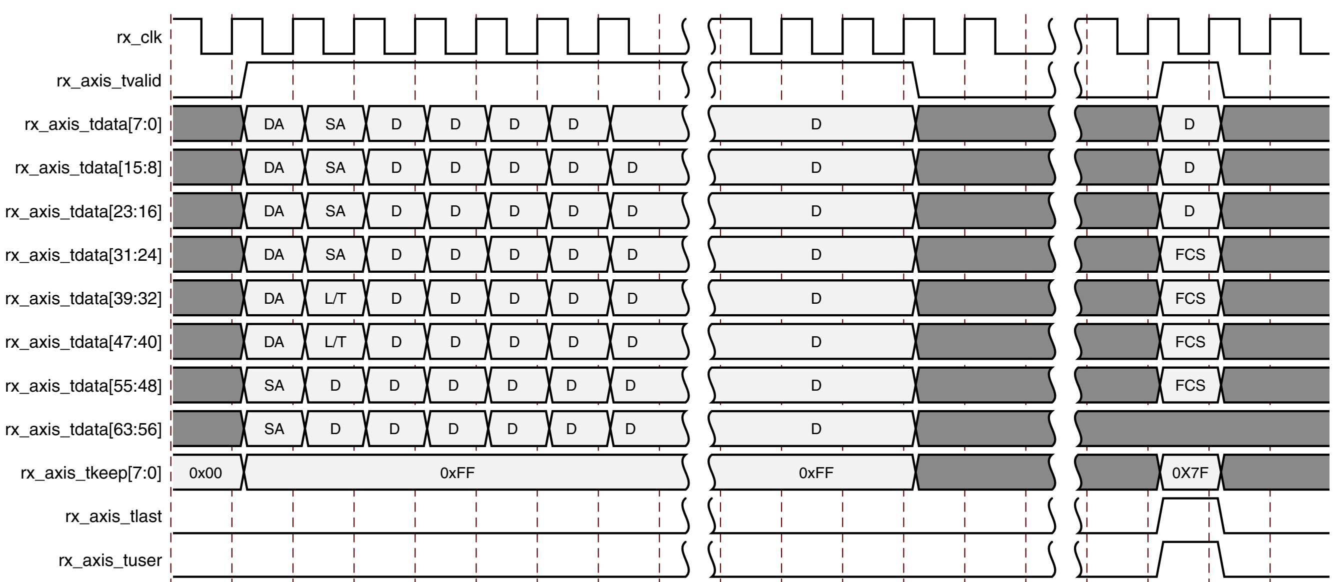


*Figure 3-22:* **Frame Reception with In-Band FCS Passing**

### *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 22), 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 3-23 shows the reception of a frame with custom preamble.

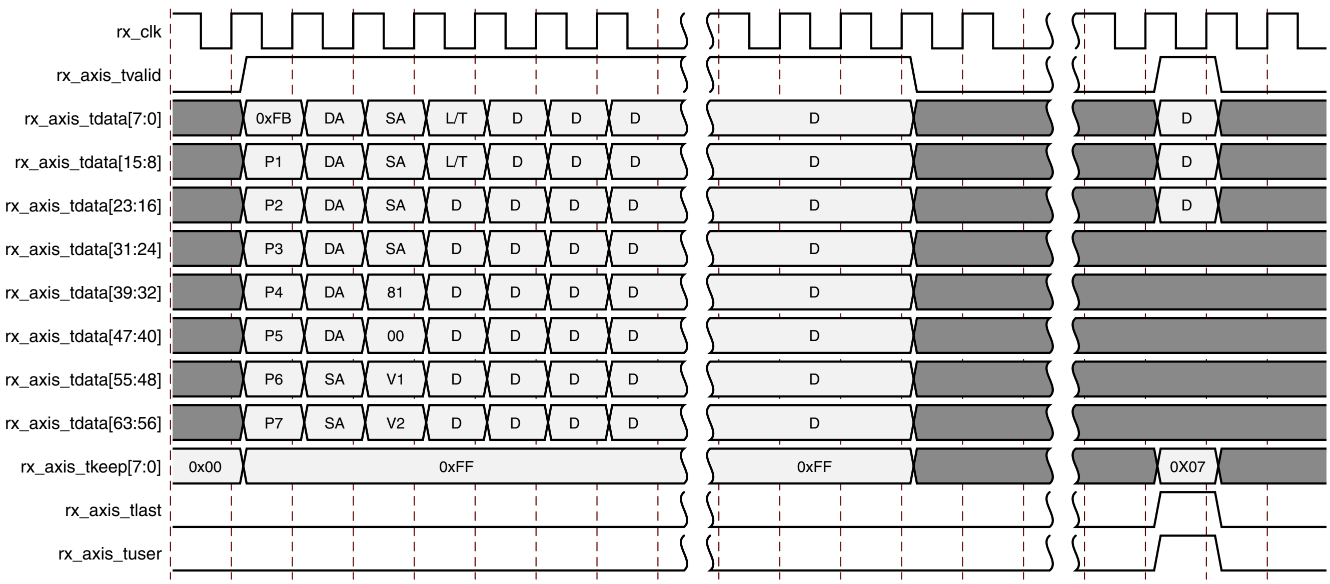


*Figure 3-23:* **Frame Reception with Custom Preamble**

### *VLAN Tagged Frames*

The reception of a VLAN tagged frame (if enabled) is represented in Figure 3-24. 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. If a frame is VLAN tagged, the subsequent length field is not checked and the length/type error is not asserted.

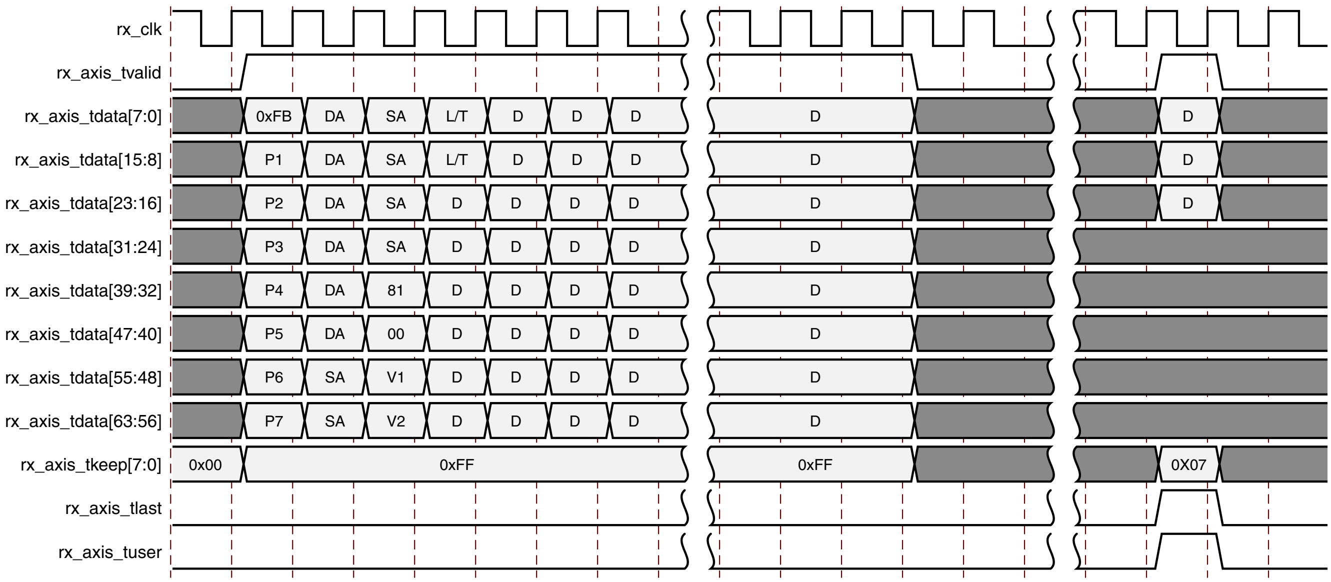


*Figure 3-24:* **Frame Reception with VLAN Tagged Frames**

### *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 22.

### *Length/Type Field Error Checks*

#### Enabled

Default operation is with the length/type error checking enabled (see Configuration Registers, page 22). In this mode the following checks are made on all received, non VLAN-tagged, frames. If either of these checks fail, the frame is marked as bad. If a frame is a control frame or has a VLAN tag, this check is not performed.

- A value in the length/type field which is ≥ 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. See Reception 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 IEEE 802.3 Flow Control Frames

The flow control block is designed to clause 31 of *IEEE Standard 802.3-2008* [Ref 1]. See Flow Control Requirement 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.

### *Transmitting an IEEE802.3 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 3-25.

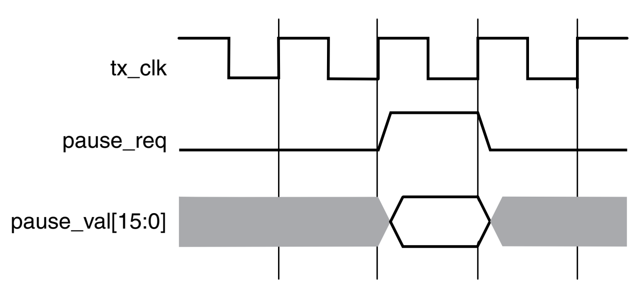


*Figure 3-25:* **Transmitting a Pause Frame**

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.

If the PFC feature is included, the flow control includes additional logic to enable it to be used as an XON/XOFF (transmission ON/ transmission OFF) interface. This is described in more detail in XON/XOFF Extended Functionality.

### *Receiving an IEEE 802.3 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).
- The length/type field is matched against the MAC Control Type code, 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 3-13. Because the received pause frame has been acted on, it is passed to the client with `rx_axis_tuser` deasserted when `rx_axis_tlast` is asserted to indicate that it should be dropped.

Reception of any frame for which the length/type field is the MAC Control Type code (88-08) but is not the legal minimum length and the Control Frame Length Check Disable bit is not set, is considered an invalid control frame. It is ignored by the flow control logic and passed to the client with `rx_axis_tuser` deasserted.

## Sending and Receiving Priority Flow Control Frames

The flow control block is designed to optionally support priority-based flow control frames as per IEEE standard 802.1Qbb. This functionality is mutually exclusive to the IEEE 802.3 flow control as described in Sending and Receiving IEEE 802.3 Flow Control Frames and unexpected behavior might arise if both modes are enabled at the same time. See Using Priority Flow Control for a description of priority-based flow control. The Ethernet MAC can be configured to send PFC frames and to act on their reception. These two behaviors can be configured asymmetrically; see Configuration Registers.

### *Transmitting a Priority Flow Control Frame*

In order to transmit a PFC frame that contains pause information for a given priority, both the global "Priority pause flow control enable (TX)" bit and the respective "Tx Priority # pause enable" bit must be enabled in the Flow Control Configuration Register (see Table 2-20). If enabled in this way, then an assertion of the respective `tx_pfc_p[0-7]_tvalid` signal results in a PFC frame being transmitted. Further PFC frames can be auto-generated by the Ethernet MAC and this is described in more detail in Transmitting a PFC Frame.

### *Receiving a Priority Flow Control Frame*

When an error-free control frame as described in Receiving a PFC Frame is received by the Ethernet MAC core *and* the frame contains the priority flow control opcode *and* the Receiver PFC is enabled in the Flow Control Configuration Register (Table 2-20) then this results in the priority enable field and per-priority quanta values being extracted from the received frame. If a given priority is enabled in the frame and the respective RX priority in the Flow Control Configuration Register is enabled, then the quanta for that priority is used to control the respective `rx_pfc_p[0-7]_tvalid`; a non-zero quanta results in the `tvalid` being asserted or remaining asserted, and a zero value results in it being deasserted or remaining deasserted. This is described in more detail in Receiving a PFC Frame. Because the received PFC frame has been acted on, it is passed to the client with `rx_axis_tuser` deasserted to indicate that it should be dropped.

## 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 internals are the same; only the interface 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.

# Connecting 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. When connecting this interface in IP Integrator, the signals of Table 3-6 (with the exception of `s_axi_aclk` and `s_axi_aresetn`) are shown, and can be connected, as a single bus. This bus is called `s_axi`.

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

| Name | Direction | Description |
|---|---|---|
| s_axi_aclk | In | AXI4-Lite clock. Range between 10 MHz and 300 MHz |
| s_axi_aresetn | In | Asynchronous active-Low reset |
| s_axi_awaddr[10:0] | 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[10:0] | 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 |

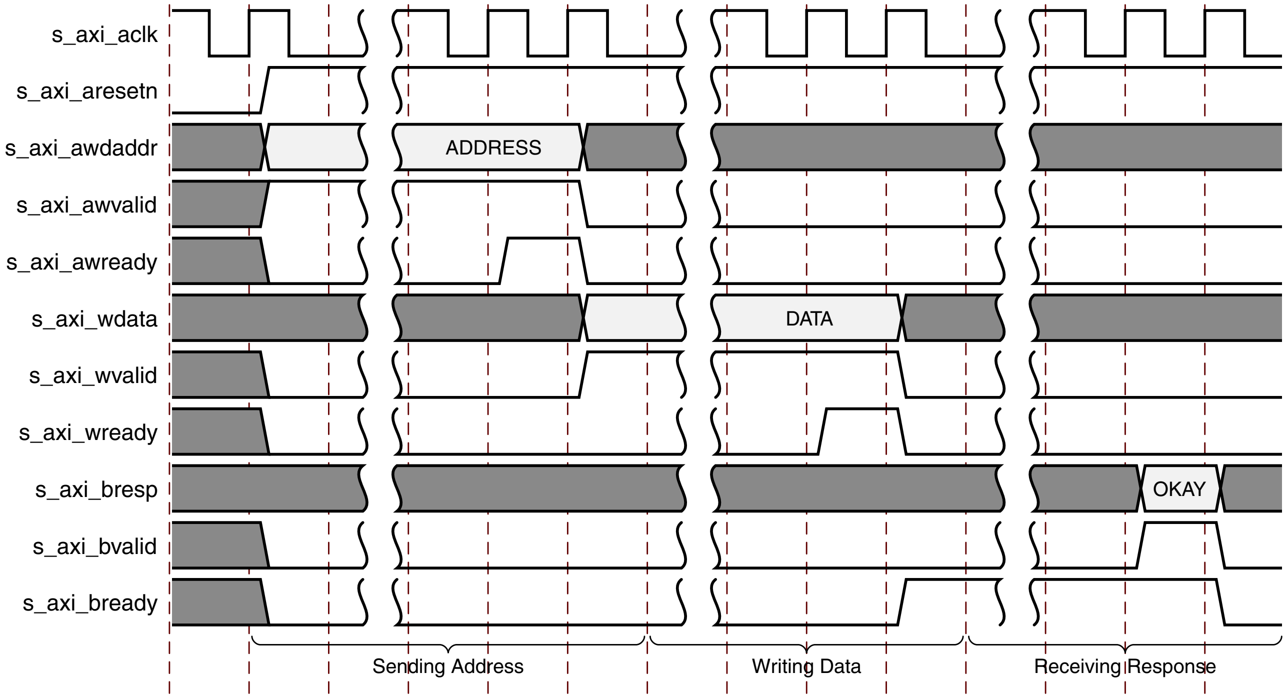


*Figure 3-26:* **Management Register Write Timing**

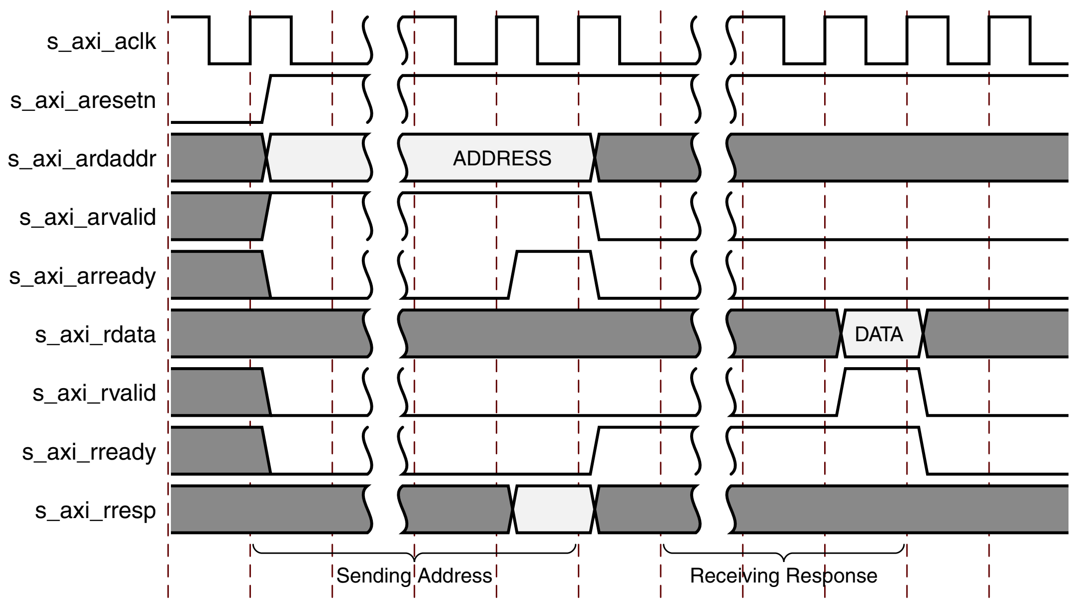


*Figure 3-27:* **Management Register Read Timing**

## 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 2-9, 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:

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

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 3-28.

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).

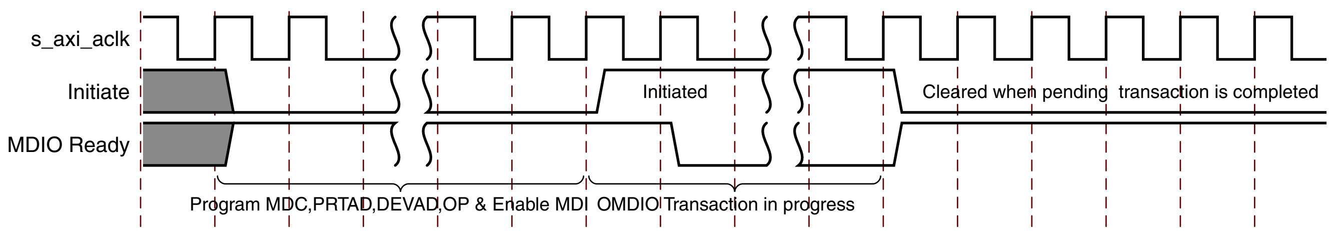


*Figure 3-28:* **MDIO Initiate**

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 3-29 illustrates the used of a SelectIO interface 3-state buffer as the bus interface.

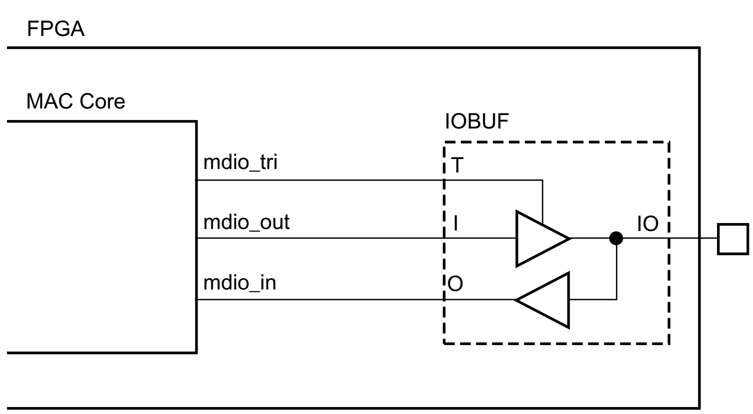


*Figure 3-29:* **Using a SelectIO Interface 3-State Buffer to Drive MDIO**

### 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 3-30 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.

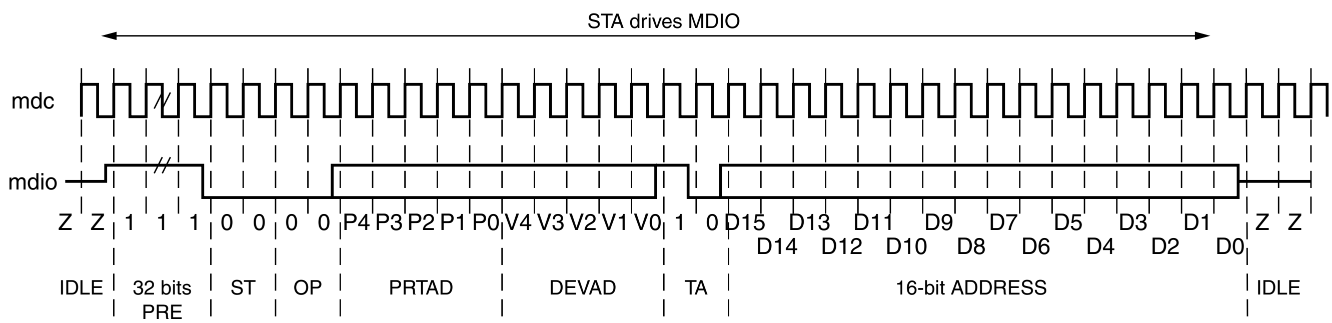


*Figure 3-30:* **MDIO Set Address Transaction**

#### Write Transaction

Figure 3-31 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.

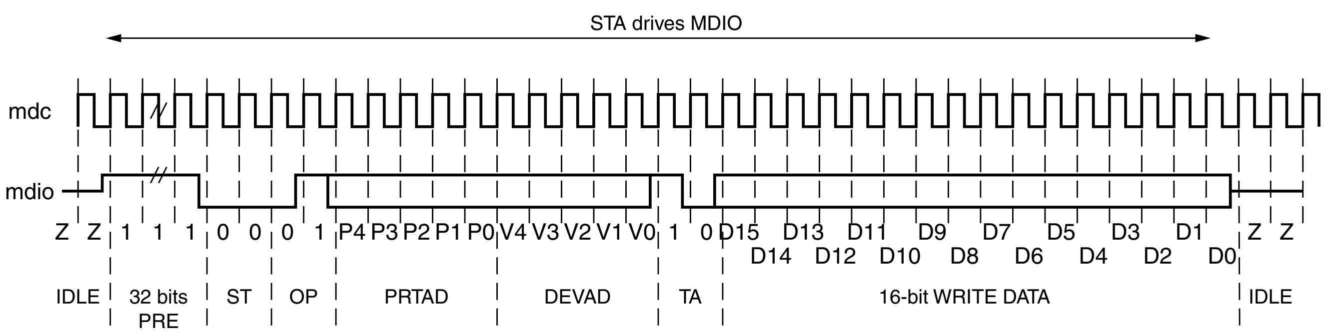


*Figure 3-31:* **MDIO Write Transaction**

### Read Transaction

Figure 3-32 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.

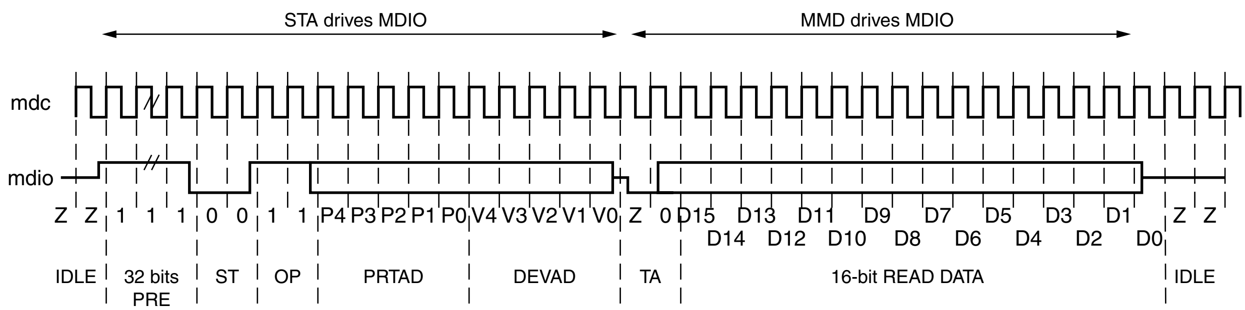


*Figure 3-32:* **MDIO Read Transaction**

### Post-Read-Increment-Address Transaction

Figure 3-33 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.

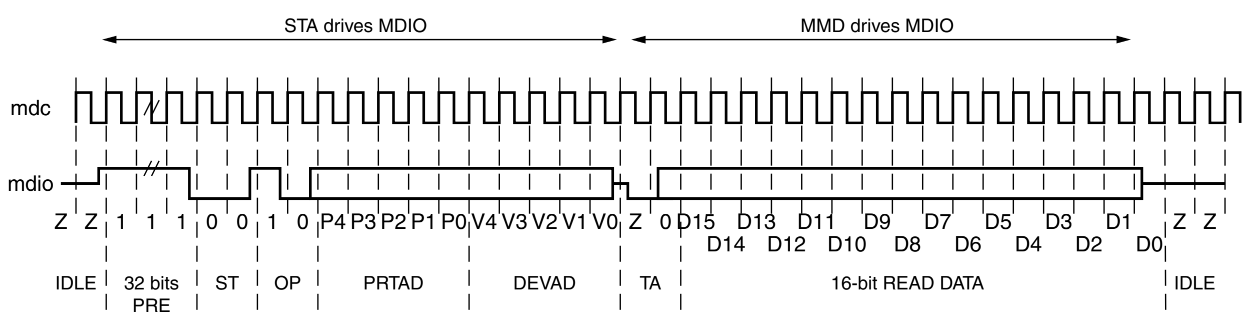


*Figure 3-33:* **MDIO Read-and-Increment Transaction**

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. |
| 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 and 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.

Bits 367:80 of Table 3-10 are only present if PFC has been enabled.
Bits 95:80 of Table 3-11 are only present if PFC has been enabled.

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

| Bits | Description(1) |
|---|---|
| 367:352 | **Legacy Pause refresh value**. When the PFC feature is included, the 802.3 flow control logic also has the capability of being used as a XON/XOFF interface. If the `pause_request` input is asserted and held High a pause frame is transmitted as normal and then refreshed when the internal quanta count reaches this value. When the pause request is deasserted, an XON frame can be automatically sent if the TX Auto XON feature is enabled. |
| 351:336 | **Tx Priority 7 Pause Quanta Refresh value**. This provides the quanta count value at which a new PFC frame is automatically generated if this priority is active and held High. |
| 335:320 | **Tx Priority 7 Pause Quanta**. This provides the quanta value which is included in a transmitted PFC frame if this priority is enabled and asserted. |
| 319:304 | **Tx Priority 6 Pause Quanta Refresh value**. This provides the quanta count value at which a new PFC frame is automatically generated if this priority is active and held High. |
| 303:288 | **Tx Priority 6 Pause Quanta.** This provides the quanta value which is included in a transmitted PFC frame if this priority is enabled and asserted. |
| 287:272 | **Tx Priority 5 Pause Quanta Refresh value**. This provides the quanta count value at which a new PFC frame is automatically generated if this priority is active and held High. |
| 271:256 | **Tx Priority 5 Pause Quanta.** This provides the quanta value which is included in a transmitted PFC frame if this priority is enabled and asserted. |
| 255:240 | **Tx Priority 4 Pause Quanta Refresh value**. This provides the quanta count value at which a new PFC frame is automatically generated if this priority is active and held High. |
| 239:224 | **Tx Priority 4 Pause Quanta.** This provides the quanta value which is included in a transmitted PFC frame if this priority is enabled and asserted. |
| 223:208 | **Tx Priority 5 Pause Quanta Refresh value**. This provides the quanta count value at which a new PFC frame is automatically generated if this priority is active and held High. |
| 207:192 | **Tx Priority 5 Pause Quanta.** This provides the quanta value which is included in a transmitted PFC frame if this priority is enabled and asserted. |
| 191:176 | **Tx Priority 2 Pause Quanta Refresh value**. This provides the quanta count value at which a new PFC frame is automatically generated if this priority is active and held High. |
| 175:160 | **Tx Priority 2 Pause Quanta.** This provides the quanta value which is included in a transmitted PFC frame if this priority is enabled and asserted. |
| 159:144 | **Tx Priority 1 Pause Quanta Refresh value**. This provides the quanta count value at which a new PFC frame is automatically generated if this priority is active and held High. |

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

| Bits | Description[(1)] |
|---|---|
| 143:128 | **Tx Priority 1 Pause Quanta.** This provides the quanta value which is included in a transmitted PFC frame if this priority is enabled and asserted. |
| 127:112 | **Tx Priority 0 Pause Quanta Refresh value**. This provides the quanta count value at which a new PFC frame is automatically generated if this priority is active and held High. |
| 111:96 | **Tx Priority 0 Pause Quanta.** This provides the quanta value which is included in a transmitted PFC frame if this priority is enabled and asserted. |
| 95 | **Tx Priority 7 Flow control enable**. If set this enables the use of tx_pfc_p7_tvalid to generate PFC frames. |
| 94 | **Tx Priority 6 Flow control enable**. If set this enables the use of tx_pfc_p6_tvalid to generate PFC frames. |
| 93 | **Tx Priority 5 Flow control enable**. If set this enables the use of tx_pfc_p5_tvalid to generate PFC frames. |
| 92 | **Tx Priority 4 Flow control enable**. If set this enables the use of tx_pfc_p4_tvalid to generate PFC frames. |
| 91 | **Tx Priority 3 Flow control enable**. If set this enables the use of tx_pfc_p3_tvalid to generate PFC frames. |
| 90 | **Tx Priority 2 Flow control enable**. If set this enables the use of tx_pfc_p2_tvalid to generate PFC frames. |
| 89 | **Tx Priority 1 Flow control enable**. If set this enables the use of tx_pfc_p1_tvalid to generate PFC frames. |
| 88 | **Tx Priority 0 Flow control enable**. If set this enables the use of tx_pfc_p0_tvalid to generate PFC frames. |
| 87:82 | Reserved |
| 81 | **Auto XON enable**. If set the Ethernet MAC automatically generates a flow control frame with the relevant quanta set to zero when the associated tvalid or pause request is deasserted (provided it has been asserted for more than one cycle. |
| 80 | **Priority Flow Control Enable**. If set this enables the TX PFC feature. This should not be set at the same time as the Transmit Flow control Enable defined in bit 5. |
| 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.<br>This address does not have any effect on frames passing through the main transmit datapath of the Ethernet MAC.<br>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 57 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 49. When set to 0 frame handling depends on the other configuration settings. |

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

| Bits | Description[(1)] |
|---|---|
| 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.<br>When this bit is set to 0, the core always stretches the IFG to maintain start alignment.<br>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.<br>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 an IEEE802.3 Pause Frame. 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. |
| 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. 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.<br>This signal is an input to the reset circuit for the transmitter block. See Resets special for details. |

1. All signals are synchronous to tx_clk0.

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

| Bits | Description[1] |
|---|---|
| 95 | **Rx Priority 7 Flow control enable**. If set this allows received, error free PFC frames with priority 7 enabled to assert the rx_pfc_p7_tvalid output for the requested duration. A new RX PFC frame can always then refresh this or cancel. |
| 94 | **Rx Priority 6 Flow control enable**. If set this allows received, error free PFC frames with priority 6 enabled to assert the rx_pfc_p6_tvalid output for the requested duration. A new RX PFC frame can always then refresh this or cancel. |
| 93 | **Rx Priority 5 Flow control enable**. If set this allows received, error free PFC frames with priority 5 enabled to assert the rx_pfc_p5_tvalid output for the requested duration. A new RX PFC frame can always then refresh this or cancel. |
| 92 | **Rx Priority 4 Flow control enable**. If set this allows received, error free PFC frames with priority 4 enabled to assert the rx_pfc_p4_tvalid output for the requested duration. A new RX PFC frame can always then refresh this or cancel. |
| 91 | **Rx Priority 3 Flow control enable**. If set this allows received, error free PFC frames with priority 3 enabled to assert the rx_pfc_p3_tvalid output for the requested duration. A new RX PFC frame can always then refresh this or cancel. |
| 90 | **Rx Priority 2 Flow control enable**. If set this allows received, error free PFC frames with priority 2 enabled to assert the rx_pfc_p2_tvalid output for the requested duration. A new RX PFC frame can always then refresh this or cancel. |
| 89 | **Rx Priority 1 Flow control enable**. If set this allows received, error free PFC frames with priority 1 enabled to assert the rx_pfc_p1_tvalid output for the requested duration. A new RX PFC frame can always then refresh this or cancel. |
| 88 | **Rx Priority 0 Flow control enable**. If set this allows received, error free PFC frames with priority 0 enabled to assert the rx_pfc_p0_tvalid output for the requested duration. A new RX PFC frame can always then refresh this or cancel. |
| 87:81 | Reserved |
| 80 | **Priority Flow Control Enable**. If set this enables the RX PFC feature and any received PFC frames is marked as bad at the client interface. If set to 0 then PFC frames are ignored and marked as good at the client interface. This should not be set at the same time as the Receive Flow control Enable defined in bit 5. |
| 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.<br>This address does not have any effect on frames passing through the main receive datapath of the Ethernet MAC.<br>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 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. When set to 0 frame handling depends on the other configuration settings. |
| 13:11 | Reserved |

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

| Bits | Description[(1)] |
|---|---|
| 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.<br>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. 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 |
| 5 | **Receive Flow Control Enable**. When this bit is 1, received flow control frames inhibit the transmitter operation as described in Receiving an IEEE 802.3 Pause Frame. 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 55. 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.<br>This signal is an input to the reset circuit for the receiver block. See Resets special for details. |

1. All signals are synchronous to rx_clk0.

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

| Bits | Description[(1)] |
|---|---|
| 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. |

1. All signals are synchronous to tx_clk0.

## Transmit Statistics Vector

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 3-34. `byte_valid` is significant on every `tx_clk0` cycle.

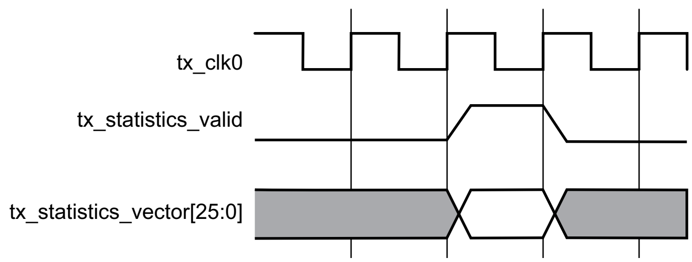


*Figure 3-34:* **Transmitter Statistics Output Timing**

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

| Bits | Name | Description |
|---|---|---|
| 26 | pfc_frame_transmitted | Extra vector bit included when the PFC functionality is included. This indicates the Ethernet MAC has generated and transmitted a PFC frame. |
| 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.<br>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 32,767 for any jumbo frames larger than this value. If maximum MTU is set, the count limits at that MTU size. If jumbo frames and maximum MTU are disabled, the count limits at 1,518 for non-VLAN frames and 1,522 for VLAN frames. The bytes transmitted is the number of bytes of ethernet data transmitted over the AXI4-Stream TX Interface + 4 bytes of CRC (if Inband FCS is off). This number includes any padding bytes in the frame. |
| 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. |

*Table 3-13:* **Transmit Statistics Vector Bit Description *(Cont'd)***

| Bits | Name | Description |
|---|---|---|
| 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 Statistics Vector

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 3-35. `bytes_valid` is significant on every `rx_clk0` cycle.

For any given received frame, `rx_statistics_valid` is High with or before the corresponding `rx_axis_tlast` assertion.

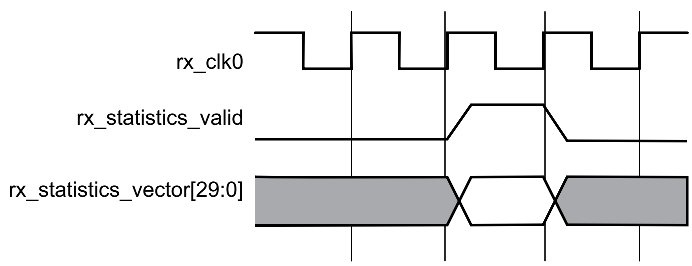


*Figure 3-35:* **Receiver Statistics Output Timing**

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

| Bits | Name | Description |
|---|---|---|
| 30 | pfc_frame | Extra vector bit included when the PFC functionality is included. This indicates the Ethernet MAC has received a valid PFC frame. |
| 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). This check is not performed on VLAN tagged frames. |
| 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. |

*Table 3-14:* **Receive Statistics Vector Description *(Cont'd)***

| Bits | Name | Description |
|---|---|---|
| 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`.<br>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 frame size as defined in Receiver Maximum Permitted Frame Length. 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 32,767 for any Jumbo frames larger than this value. |
| 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 IEEE 802.3 Flow Control

This section 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. If PFC functionality is included, the Ethernet MAC should be configured for either IEEE 802.3 pause frames or PFC frames but not both as they are considered mutually exclusive modes of operation.

## Flow Control Requirement

Figure 3-36 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.

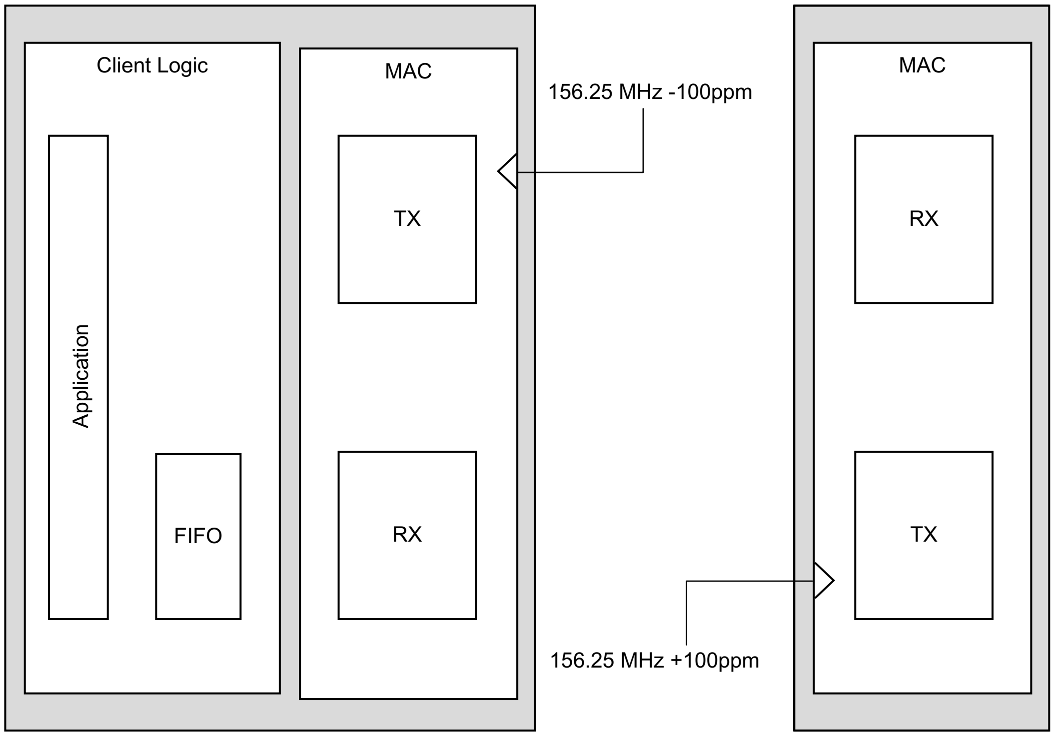


*Figure 3-36:* **Requirement for Flow Control**

## Flow Control Basics

A MAC can 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 3-36 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 3-36 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.

## IEEE 802.3 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 3-37.

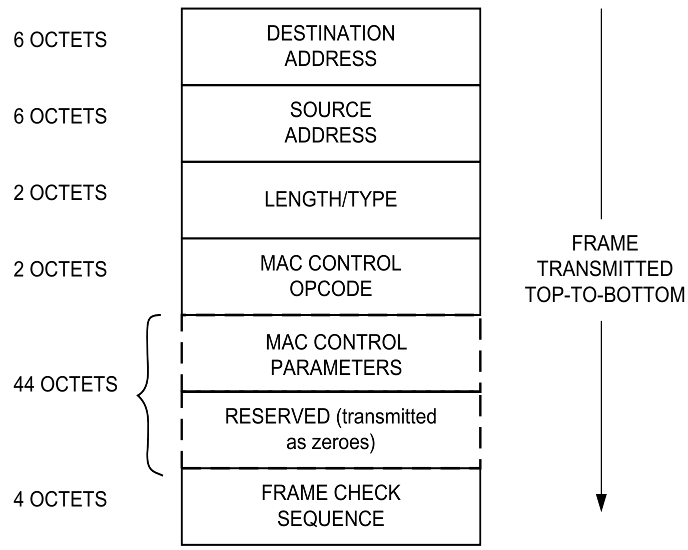


*Figure 3-37:* **MAC Control Frame Format**

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.

## 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 3-38 displays this timing.

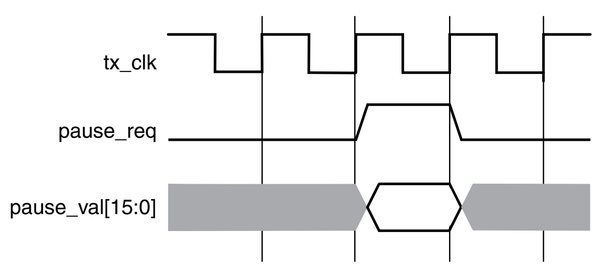


*Figure 3-38:* **Pause Request Timing**

This action causes the core to construct and transmit a pause control frame on the link with the MAC Control frame parameters (see Figure 3-37):

- 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).
- 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.

### XON/XOFF Extended Functionality

If the Ethernet MAC has been generated with PFC functionality included but configured for IEEE 802.3 functionality then the Ethernet MAC supports XON/XOFF functionality. If, as shown in Figure 3-39, the `pause_req` signal is asserted, the Ethernet MAC generates a new pause frame. If it is subsequently held High for more than one clock cycle then the Ethernet MAC automatically generates a new pause frame each time the internal quanta count of the Ethernet MAC reaches the number of quanta, specified by the legacy refresh value (in either the register or the configuration vector). This is shown as an XOFF refresh frame in Figure 3-39. When the `pause_req` is deasserted, an XON frame (standard pause frame with the pause quanta forced to zero) is automatically generated if the auto XON feature is enabled; this functionality is shown in Figure 3-39. If auto XON is not enabled then the remaining quanta are left to expire naturally at the link partner.

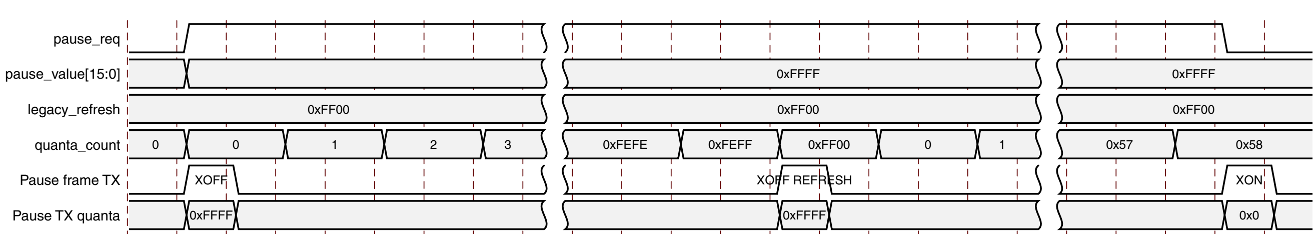


*Figure 3-39:* **XON/XOFF Frame Transmission**

### *Client-Initiated Pause Request*

For maximum flexibility, flow control logic can be disabled in the core (see Configuration Registers) 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).

## 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 3-37. 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) or the Control Frame length check disable bit must be set. See Configuration Registers.

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_axis_tuser` deasserted when `rx_axis_tlast` is asserted to indicate a bad frame.

- **Pause Frame Reception Disabled**

  When pause control reception is disabled (see Configuration Registers), an error free control frame is received through the client interface with the `rx_axis_tuser` asserted when `rx_axis_tlast` is asserted to indicate a good frame. In this way, the frame is passed to the client logic for interpretation (see Client-Initiated Response to a Pause Request).

- **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_axis_tuser` 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_axis_tuser` deasserted to indicate to the client that can now be dropped. The frame is still marked good in the statistics vector and counters.

***Note:*** Any frame in which the length/type field contains the MAC Control Type code should be dropped by the receiver client logic. All control frames are indicated by rx_statistic_vector bit 20 (see Receive Statistics Vector, page 74).

### *Client-Initiated Response to a Pause Request*

For maximum flexibility, flow control logic can be disabled in the core (see Configuration Registers, page 22) 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_axis_tuser` 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 3-36. 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 3-40 illustrates the FIFO occupancy over time.

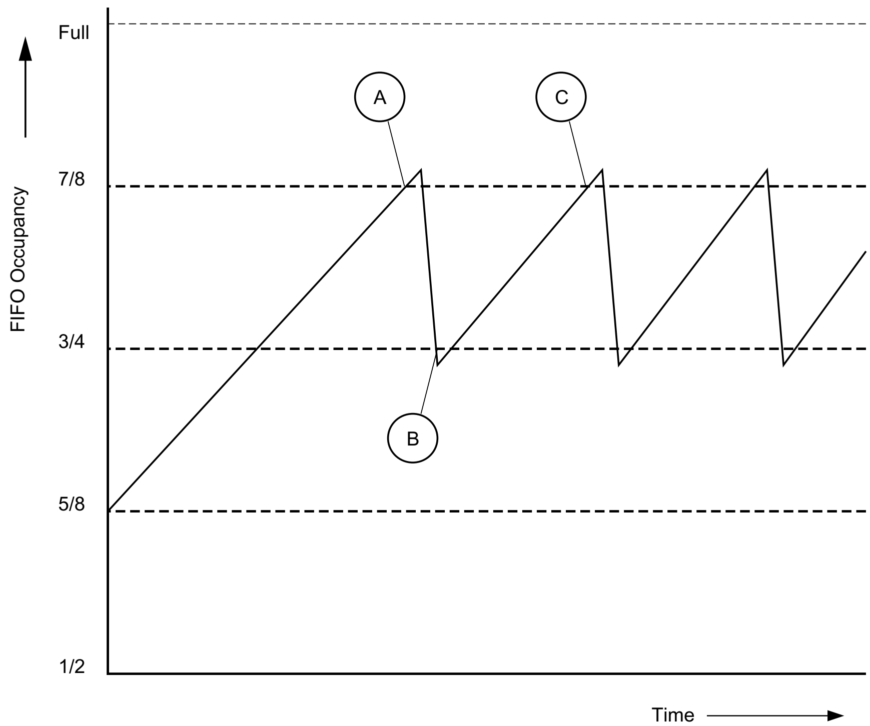


*Figure 3-40:* **Flow Control Implementation Triggered from FIFO Occupancy**

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.

# Using Priority Flow Control

This section describes the operation of the priority flow control logic of the core. Priority-based flow control is defined in the *IEEE Standard 802.1Qbb* [Ref 2]. The Ethernet MAC can be configured to transmit and receive priority-based flow control requests; these modes of operation can be independently enabled or disabled. See Configuration Registers. Operation of the MAC with both PFC and 802.3 flow control enabled is not supported as these modes of operation are mutually exclusive.

## Priority Flow Control Requirement

As described in Using IEEE 802.3 Flow Control the basic requirement for flow control is to avoid dropping frames if a receiving port cannot process frames at the rate at which they are being provided. IEEE 802.3 pause frames operate on the entire link which is not ideal when there are multiple priority queues awaiting transmission with each queue originating from a separate FIFO. In this case the transmission of one or more specific queues (up to eight) can be inhibited using priority-based flow control with the eight frame priorities as defined by IEEE 802.1p. Figure 3-41 illustrates an implementation of two different priority queues.

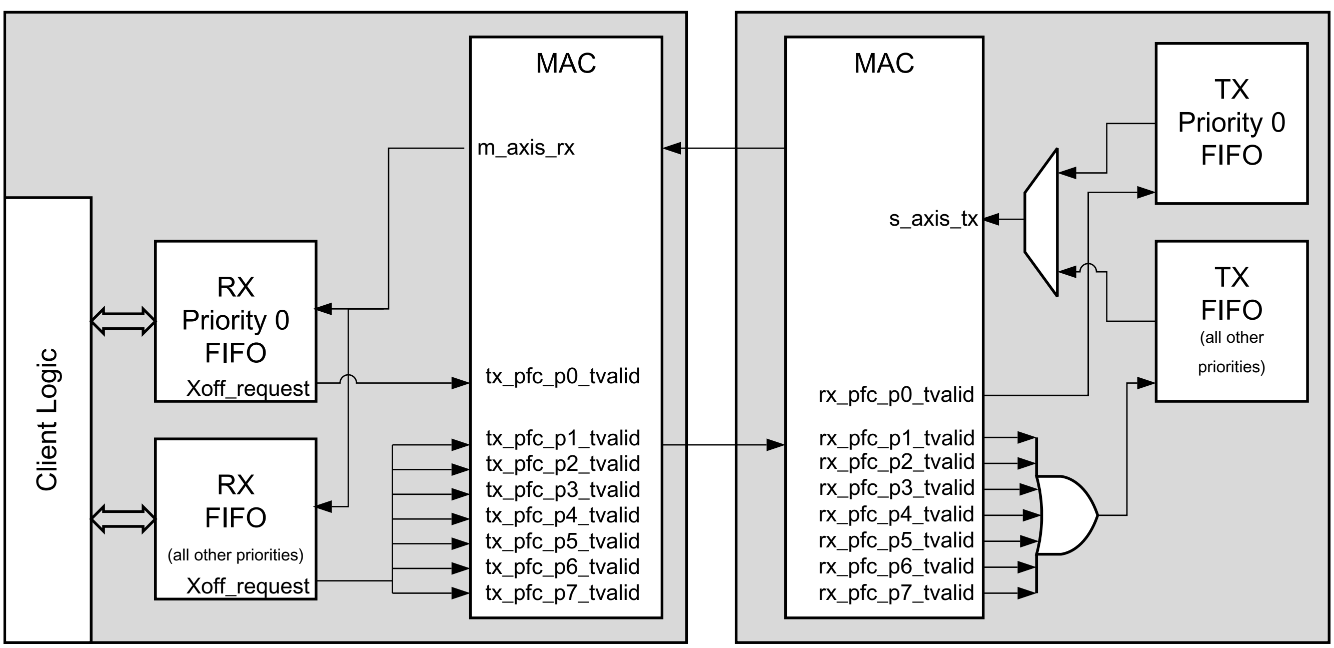


*Figure 3-41:* **Priority Flow Control Requirement**

A MAC can transmit a PFC frame to request that its link partner cease transmission of one or more of the eight frame priority queues for a defined period of time. For example, the Ethernet MAC at the left side of Figure 3-41 can initiate a PFC request if its RX priority 0 FIFO reaches a nearly full state; this is considered to be an XOFF request. To make this request, the RX Priority 0 FIFO drives the `tx_pfc_p0_tvalid` signal High, and holds it

High until it is ready to accept data again (this is one of the modes of operation supported by the MAC). If this FIFO remains nearly full for an extended period of time, the Ethernet MAC automatically generates a new PFC request to prevent the link partner from restarting transmission of this frame class when the initially requested duration (the original requested number of pause quanta) has expired. When the FIFO level drops to a specified level and the FIFO is able to accept data again, the requirement to inhibit transmission of that frame class can be removed by driving the `tx_pfc_p0_tvalid` signal Low. At this point, the Ethernet MAC core can be optionally configured to generate a PFC frame to cancel any remaining pause quanta (duration) by placing a zero pause quanta value into the PFC frame for priority 0; this is considered to be an XON request.

When the Ethernet MAC core receives priority-based flow control frames it cannot directly cease transmission of the specified priority (or priorities) without ceasing transmission of all frames. To enable specific priorities to be paused the MAC has a per priority pause request output which can be asserted to inhibit transmission of specific priorities. For example, the Ethernet MAC at the right side of Figure 3-41 might cease transmission from its TX priority 0 FIFO queue after receiving the PFC frame transmitted by the left-hand MAC. In a well designed system, the right side MAC would cease transmission before the RX priority 0 FIFO of the left side 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.

## Priority-Based Flow Control Frames

Priority-based flow control frames are a special type of Ethernet frame defined in IEEE 802.1Qbb. Control frames are identified from other frame types by a defined value placed into the length/type field (the MAC Control Type code). The priority-based flow control frame format is shown in Figure 3-42.

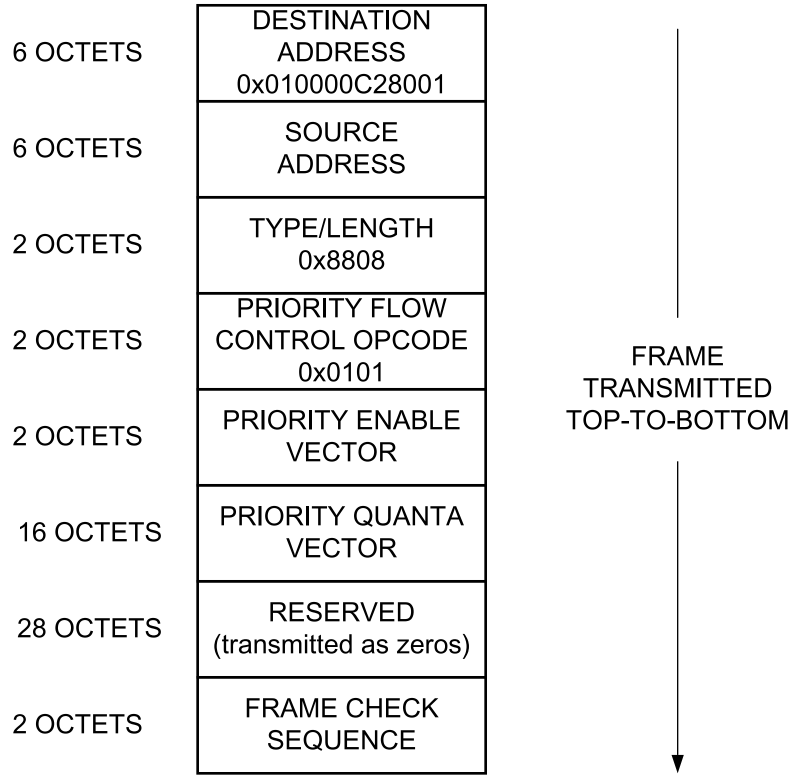


*Figure 3-42:* **MAC Priority Control Flow Frame Format**

A PFC frame is a special type of control frame, identified by a defined value placed into the OPCODE field, this is shown in Figure 3-42.

The 16-bit priority enable vector contains eight priority enable bits with all other bits set to zero. The priority vector field contains eight 16-bit quanta values, one for each priority, with priority 0 being the first. This defines the number of pause_quantum (512-bit times of the particular implementation) for the priority pause duration request. For 10-Gigabit Ethernet, a single `pause_quantum` corresponds to 51.2 ns.

## Transmitting a PFC Frame

### *Core-Initiated Request*

There are three methods of generating a PFC frame, all of which assume that TX PFC is enabled and any actively used priority has the relevant TX priority enable set to 1.

1. The client asserts one or more of the eight `tx_pfc_p[0-7]_tvalid` signals. If the `tx_pfc_p[0-7]_tvalid` signal is asserted for more than a single cycle then this is considered to be an XOFF request and the MAC refreshes the relevant quanta at the link partner whenever a new PFC frame is transmitted until the `tx_pfc_p[0-7]_tvalid` is deasserted. If `tx_pfc_p[0-7]_tvalid` is asserted for a single cycle then it is assumed

that any required refresh is directly controlled by a subsequent reassertion of the respective tvalid as required.

2. The client keeps a priorities tx_pfc_p[0-7]_tvalid signal High and the internal quanta count of the Ethernet MAC reaches the pre-programmed refresh value for that priority. On reaching this refresh value, the MAC "refreshes" the priority pause request by resending a new PFC frame with the pre-programmed pause value for the given priority. For each of the eight priorities, there is a configuration register that contains both the pause duration and the refresh value (see Table 2-24 in Configuration Registers.)
3. The client has held a tx_pfc_p[0-7]_tvalid High for more than one cycle; this is then deasserted and the TX auto XON feature is enabled. This results in a new PFC frame with the relevant priorities quanta being both enabled and forced to zero. This is considered to be an XON request for that priority.

When any new PFC frame is transmitted it also resends the quanta for any currently active, enabled priority, effectively refreshing each priority quanta at the link partner. This also restarts the internal quanta count.

The eight tx_pfc_p[0-7]_tvalid signals are synchronous with respect to tx_clk0. The various transmit methods can be seen in Figure 3-43. In Figure 3-43 the tx_pfc_p0_tvalid is asserted and held High. This results in a PFC frame with only the P0 quanta enabled (set to 0xFFFF). A number of cycles later tx_pfc_p2_tvalid is asserted for a single cycle and this results in a new PFC frame with both the P0 and P2 quantas enabled (and restarts the internal quanta count). The internal quanta count subsequently reaches the programmed refresh value for Priority 0 and a refresh PFC frame is sent with only the P0 quanta enabled (as all other tvalid requests are Low). tx_pfc_p6_tvalid is asserted and held High and this also results in a new PFC frame (P6 XOFF), with both P0 and P6 quantas enabled. Finally in Figure 3-43, tx_pfc_p0_tvalid is deasserted resulting in another PFC frame (P0 XON); this has both the P0 and P6 quantas enabled with the P0 quanta set to 0x0.

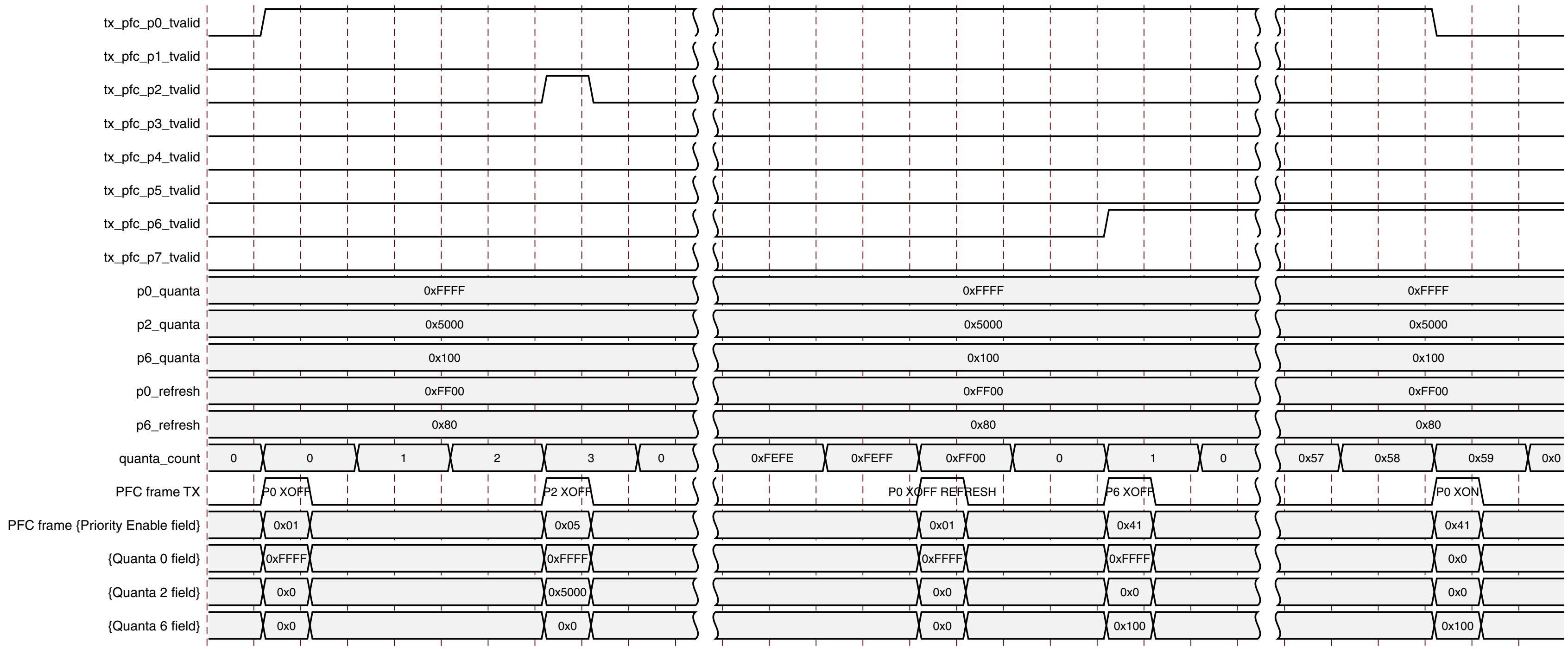


*Figure 3-43:* **TX PFC Frame Transmission**

### *Client-Initiated Request*

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

Send Feedback

## Receiving a PFC Frame

### *Core-Initiated Response to a PFC request*

An error-free control frame is a received frame matching the format of Figure 3-42. 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 the minimum legal Ethernet MAC frame size and the defined size for control frames) or the Control Frame Length Check Disable must be set.

Any control frame received that does not conform to these checks contains an error, and is passed to the receiver client with the `rx_axis_tuser` signal deasserted on the cycle that `rx_axis_tlast` is asserted.

### *PFC Frame Reception Disabled*

When PFC reception is disabled (see Configuration Registers), an error free control frame is received through the client interface with the `rx_axis_tuser` signal asserted. In this way, the frame is passed to the client logic for interpretation (see Client-Initiated Response to a Pause Request).

### *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:

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

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

If the frame passes all of these checks, is of minimum legal size, or the Control Frame Length Check Disable is set, and the MAC receiver PFC is enabled, then the priority enable field and per priority quanta values are extracted from the frame. If a particular priority is enabled in the frame with a non-zero pause quanta value, then the respective `rx_pfc_p[0-7]_tvalid` output is asserted and the requested quanta value loaded into the control logic for that priority. This control logic consists of a counter which is loaded with the requested pause quanta value, and decrements down to zero. The `rx_pfc_p[0-7]_tvalid` remains asserted while the local quanta counter is non zero. The local quanta counter does not start to decrement until `rx_pfc_p[0-7]_tvalid` is High. Subsequent deassertion of `rx_pfc_p[0-7]_tready` does not stop the quanta expiration. If the quanta counts down to zero and is not refreshed by reception of a new PFC frame with the respective priorities quanta enabled and non-zero then `rx_pfc_p[0-7]_tvalid` is deasserted. An example of this is shown in Figure 3-44 which shows the case where only one priority is active in a frame.

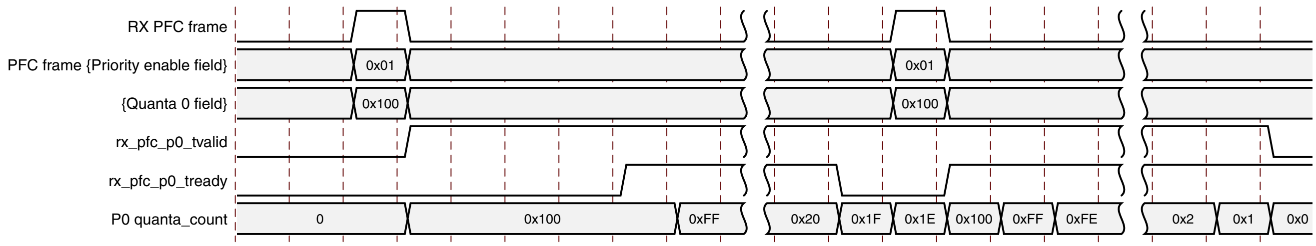


*Figure 3-44:* **RX PFC Frame Reception**

If at any time a new PFC frame is received with a priorities quanta enabled and set to zero then this is loaded into the local quanta count which results in the respective `rx_pfc_p[0-7]_tvalid` being deasserted immediately to re-enable transmission of this particular priority queue.

Because the received PFC frame has been acted on, it is passed to the client with `rx_axis_tuser` deasserted to indicate that it should be dropped. The frame is still marked good in the statistics vector and counters.

***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 20 (see Receive Statistics Vector).

## Client-Initiated Response to a Pause Request

For maximum flexibility, flow control logic can be disabled in the core (see Configuration Registers) 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_axis_tuser` 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 of the appropriate priorities through the core, if applicable.

## PFC Implementation Example

This section describes a simple example of a PFC implementation.

Consider again the system illustrated in Figure 3-41. The client logic connected to the Ethernet MAC on the left side is unable to service the data in the RX priority 0 FIFO for an extended period of time. Over time, the RX priority 0 FIFO illustrated fills and overflows. The aim is to implement a PFC method which applies back pressure on only the priority 0 traffic to ensure no frames are dropped whilst allowing the other priority queues to continue.

### Method

1. Choose an RX priority 0 FIFO nearly-full occupancy threshold (7/8 occupancy is used in this description but the choice of threshold is implementation-specific). When the FIFO exceeds this occupancy, assert the XOFF request signal to initiate a PFC frame with priority 0 enabled and 0xFFFF used as the priority 0 `pause_quantum` duration (0xFFFF is the default value of the priority 0 quanta register). This is the maximum pause duration. This causes the left-hand MAC to transmit a PFC frame, which in turn causes the right-hand MAC to assert its `rx_pfc_p0_tvalid` to request that the TX priority 0 FIFO on the right-hand side stops transmission. If the client logic of the left-hand MAC continues to be unable to service the RX priority 0 FIFO then the left-hand Ethernet MAC automatically re-sends the PFC frame each time 0xFF00 quanta has expired (this is the default value of the priority 0 refresh), that is, before the previously sent quanta has expired.
2. Choose a second RX priority 0 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, send an XON request by deasserting the `tx_pfc_p0_tvalid` signal. If the TX auto XON feature is enabled this initiates a PFC frame with priority 0 enabled and the priority 0 quanta set to 0x0000. This indicates a zero pause duration (XON), and upon receiving this PFC frame, the right-hand MAC deasserts the `rx_pfc_p0_tvalid` allowing the TX priority 0 FIFO on the right to resume transmission (it does not wait for the original requested quantum duration to expire). If the TX auto XON feature is not enabled then no PFC frame is sent and transmission does not restart until the requested quantum duration has expired.

### Operation

Figure 3-45 shows the Priority 0 FIFO occupancy over time.

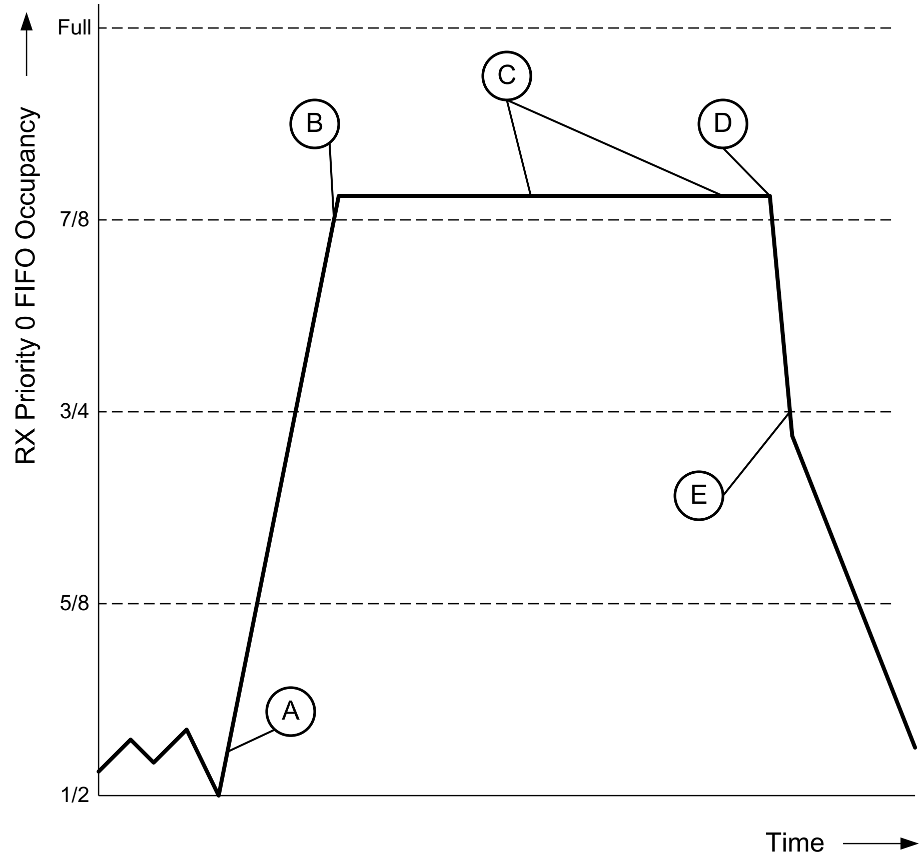


*Figure 3-45:* **Priority Flow Control Implementation Triggered from FIFO Occupancy**

1. The FIFO occupancy is maintained at a low level as the client logic is able to service the frames. At point A, the client logic is unable to service the FIFO and the occupancy increases. At point B the FIFO has reached the threshold of 7/8 occupancy. This triggers the XOFF request assertion and a PFC frame is generated and requested priority 0 traffic is stopped.
2. Upon receiving the PFC frame, the right-hand priority 0 FIFO ceases transmission.
3. The client logic remains unable to service the priority 0 FIFO for an extended duration and subsequent PFC frames are automatically generated at C to ensure the Priority 0 FIFO on the right does not restart.
4. The client begins to service the priority 0 FIFO at point D and the FIFO empties. The occupancy falls to the second threshold of ¾ occupancy at point E. This triggers the XON PFC frame request.
5. Upon receiving this PFC frame, the right-hand MAC deasserts the `rx_pfc_p0_tvalid` and the priority 0 FIFO resumes transmission.
6. Normal operation resumes.

# 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 IP 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 3-46.

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.

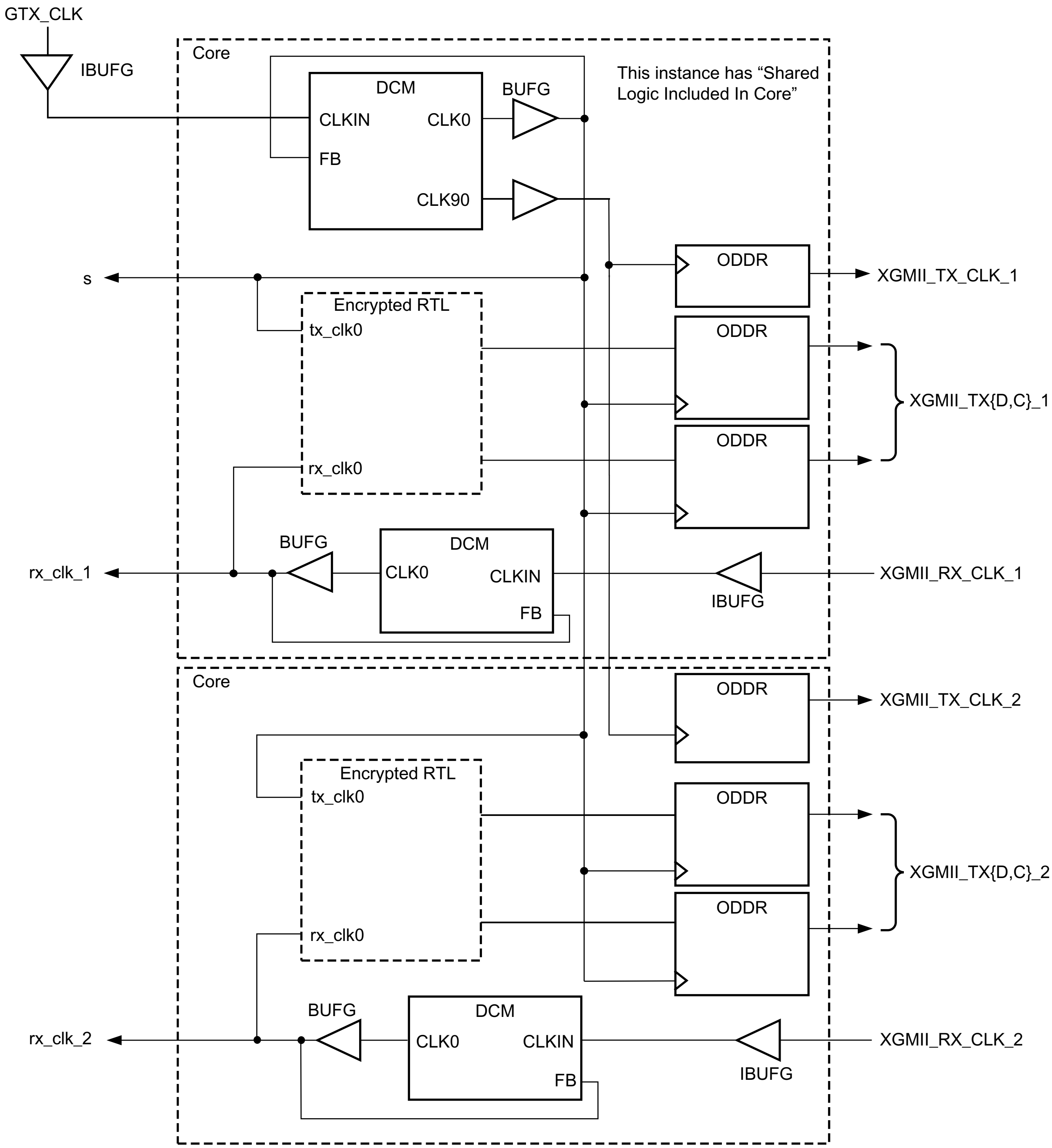


*Figure 3-46:* **Clock Management, Multiple Instances of the Core with XGMII**

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 you. 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 IP 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 web page.

Other documentation for the XAUI core can also be found at the product web page.

Figure 3-47 illustrates the connections and clock management logic required to interface the 10-Gigabit Ethernet MAC core to the XAUI core in 7 series devices. This shows that:

- Use the top-level of both cores as delivered (<component_name>.v/vhd).
- Use the shared clocking and reset logic provided in the XAUI core support level of hierarchy. If the "Include Shared Logic in Core" option is selected when generating the XAUI core, the clocking and reset logic is already included in the core top-level instead of the example design.
- 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.
- If an MMCM is used, the MMCM LOCKED output should be connected to the 10-Gigabit Ethernet MAC core `rx_dcm_locked` and `tx_dcm_locked` inputs.
- 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.

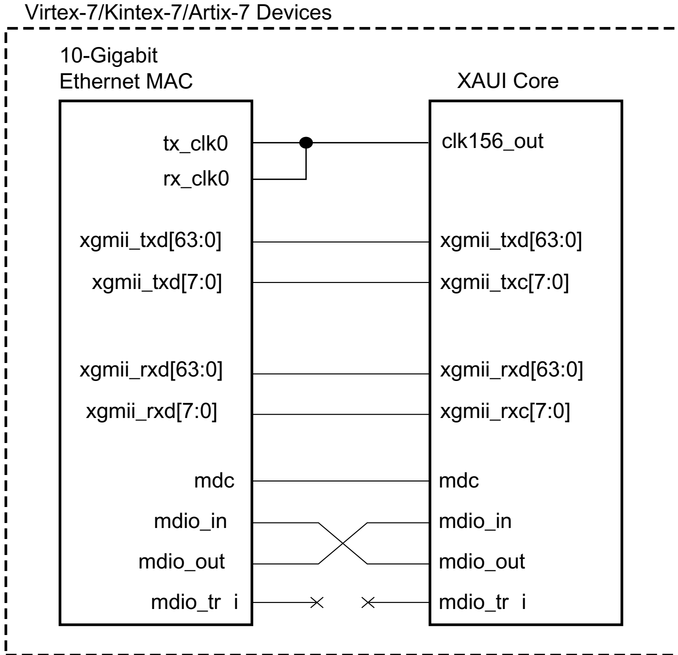


*Figure 3-47:* **10-Gigabit Ethernet MAC Core Integrated with XAUI Core – 7 Series Devices**

Figure 3-47 illustrates a 50 MHz clock being connected into the DCLK of the XAUI core. A wide range of frequencies for this clock is possible (refer to the relevant transceiver User Guide), so the 50 MHz here is shown only as an example. For details on clocks and transceiver placement using the XAUI core, see *LogiCORE IP XAUI Product Guide (*PG053) [Ref 3].

## 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 web page.

Other documentation for the RXAUI core can also be found at the product web page.

Figure 3-48 illustrates the connections and clock management logic required to interface the 10-Gigabit Ethernet MAC core to the RXAUI core in Virtex-7 and Kintex-7 FPGAs. This shows that:

- Use the top-level of both cores as delivered (<component_name>.v/vhd).
- Use the shared clocking and reset logic provided in the RXAUI core support level of hierarchy. If the “Include Shared Logic in Core” option is selected when generating the RXAUI core, the clocking and reset logic is already included in the core top-level instead of the example design.
- 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.
- If an MMCM is used, the MMCM LOCKED output should be connected to the 10-Gigabit Ethernet MAC core `rx_dcm_locked` and `tx_dcm_locked` inputs.
- 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.

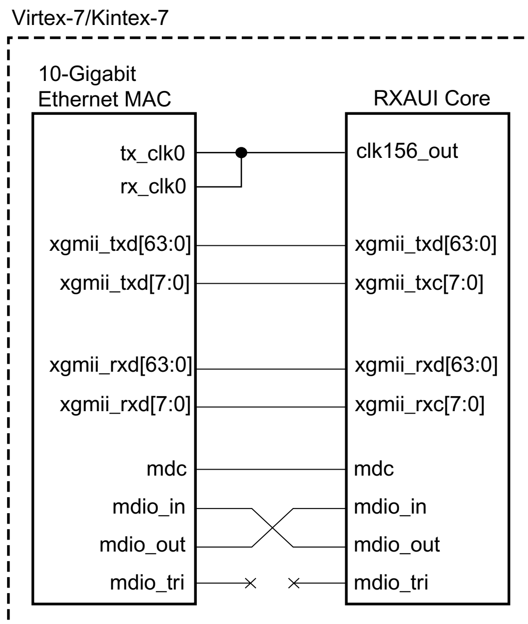


*Figure 3-48:* **10-Gigabit Ethernet MAC Core Integrated with RXAUI Core – Virtex-7 and Kintex-7 FPGAs**

Figure 3-48 illustrates a 50 MHz clock being connected into the DCLK of the XAUI core. A wide range of frequencies for this clock is possible (refer to the relevant transceiver User Guide), so the 50 MHz here is shown only as an example. For details on clocks and transceiver placement using the RXAUI core, see *LogiCORE IP RXAUI User Guide* (PG083) [Ref 4].

## 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 web page.

Other documentation for the PCS/PMA core can also be found at the product web page.

Figure 3-47 illustrates the connections and clock management logic required to interface the 10-Gigabit Ethernet MAC core to the PCS/PMA core in Virtex-7 and Kintex-7 FPGAs. This shows that:

- Use the top-level of both cores as delivered (<component_name>.v/vhd)
- Use the shared clocking and reset logic provided in the PCS/PMA core support level of hierarchy. If the "Include Shared Logic in Core" option is selected when generating the PCS/PMA core, the clocking and reset logic is already included in the core top-level instead of the example design.
- 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.
- If an MMCM is used, the MMCM LOCKED output should be connected to the 10-Gigabit Ethernet MAC core `rx_dcm_locked` and `tx_dcm_locked` inputs.
- 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.

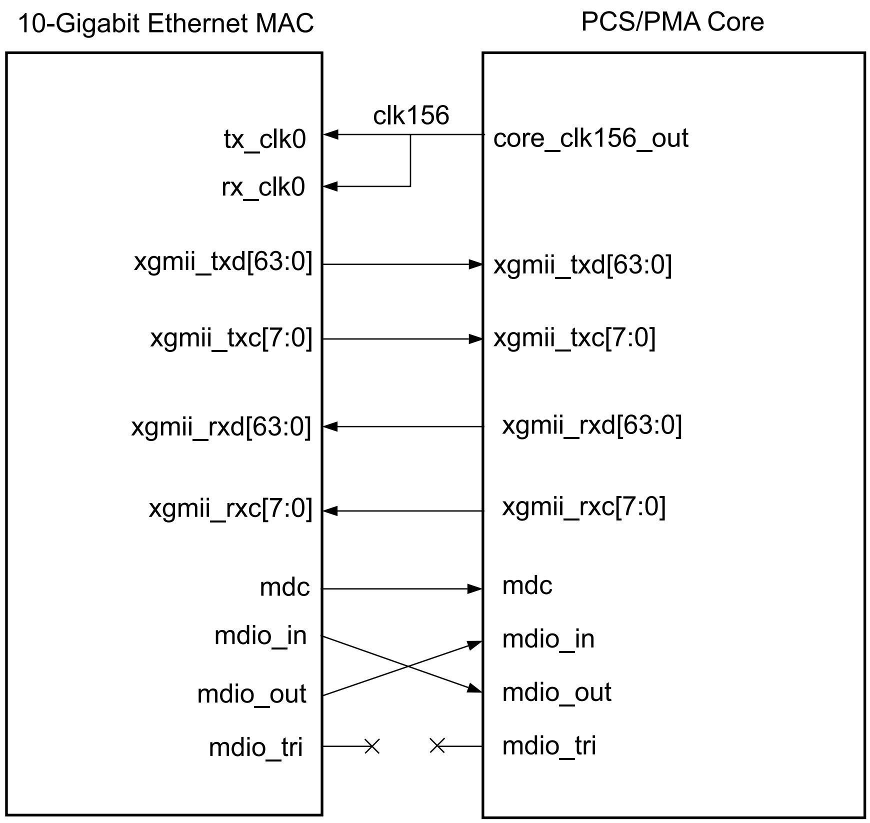


*Figure 3-49:* **10-Gigabit Ethernet MAC Core Integrated with PCS/PMA Core – Virtex-7 and Kintex-7 FPGAs**

For details on clocks and transceiver placement using the PCS/PMA core, see *LogiCORE IP 10-Gigabit Ethernet PCS/PMA Product Guide* (PG068) [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.

# Design Flow Steps

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

- *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 6]
- *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 7]
- *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 8]
- *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 9]]

## Customizing and Generating the Core

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

If you are customizing and generating the core in the Vivado IP Integrator, see the *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 6] for detailed information. IP Integrator might auto-compute certain configuration values when validating or generating the design. To check whether the values do change, see the description of the parameter in this chapter. To view the parameter value you can run the `validate_bd_design` command in the Tcl console.

### Vivado Integrated Design Environment

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

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

For details, see the *Vivado Design Suite User Guide, Designing with IP* (UG896) [Ref 7] and the *Vivado Design Suite User Guide, Getting Started* (UG910) [Ref 8].

***Note:*** Figures in this chapter are illustrations of the Vivado IDE. The layout depicted here might vary from the current version.

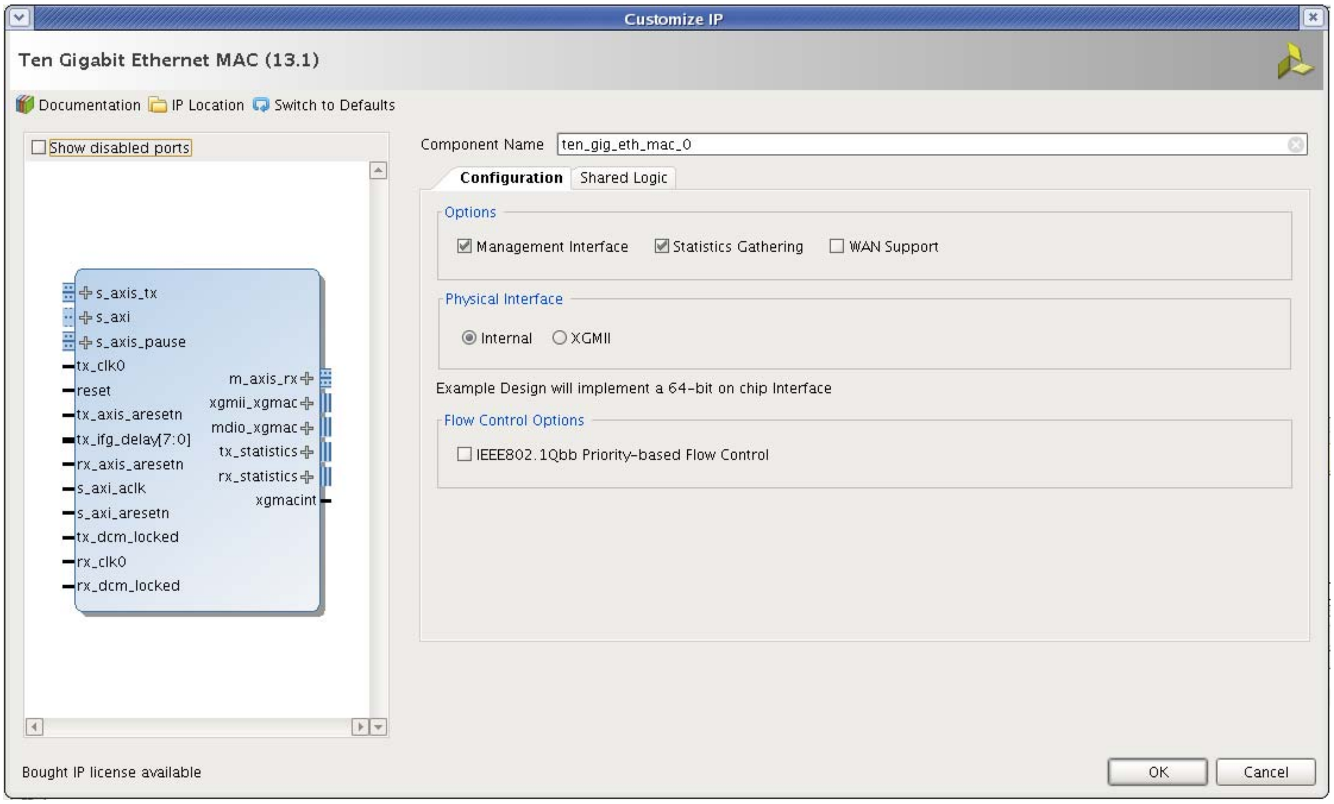


*Figure 4-1:* **10-Gigabit Ethernet MAC Customization Screen**

### *Component Name*

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

### *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.

#### WAN Support

Select this option to include WAN mode support in the core. When included, circuitry to perform Interframe Gap Adjustment, page 49 is included in the core.

#### 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.

#### Enable Priority Flow Control

Select this option to include Priority Flow control support in the core. When included, circuitry to generate PFC frames on transmit and interpret PFC frame on receive is included as well as enhanced XON/XOFF support for IEEE 802.3 pause requests. The default is for this to be disabled.

#### Shared Logic

Select whether some shared logic (including transmit MMCMs, BUFGs) is included in the core itself or in the example design (see Shared Logic in Chapter 3).

### Output Generation

For details, see the *Vivado Design Suite User Guide, Designing with IP* (UG896) [Ref 7].

## Constraining the Core

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

### Required Constraints

This section defines the constraint requirements of the 10-Gigabit Ethernet MAC core. An example XDC is provided with the HDL example design to provide the board-level constraints. This is specific to the example design and is only expected to be used as a template for the user design. See Chapter 5, Example Design. This XDC file, named `<component name>_example_design.xdc`, is found in the IP Sources tab of the Sources window in the Examples file group.

Also, a core-level constraints file (`<component_name>.xdc`) is found in the **IP Sources** tab. This is applied to the core on a per-instance basis and covers all internal constraints.

## Device, Package, and Speed Grade Selections

The core can be implemented in 7 series devices with these attributes:

- Large enough to accommodate the core
- Contains a sufficient number of IOBs
- Device has a supported speed grade (see Table 4-1).

*Table 4-1:* **Supported Speed Grades**

| Device Family | Speed Grade |
|---|---|
| Virtex-7 FPGA | –1 or faster |
| Kintex-7 FPGA | –1 or faster |
| Artix-7 FPGA | –2 or faster |
| Zynq-7000 SoC: 030 or larger | –1 or faster |

## Clock Frequencies

The 10-Gigabit Ethernet MAC solution has a variable number of clocks with the precise number required depending on the specific parameterization.

Because the core targets a specific interface standard (XGMII), there are associated clock frequency requirements (see Table 4-2).

*Table 4-2:* **Ethernet MAC Solution Frequency Requirements**

| Clock Name | Parameterization | Frequency Requirement |
|---|---|---|
| gtx_clk | Always present | 156.25 MHz |
| xgmii_rx_clk | Present when XGMII is present | 156.25 MHz |
| s_axi_aclk | Management Type set to AXI4-Lite | 10–300 MHz |

## Clock Management

This section is not applicable for this IP core.

## Clock Placement

This section is not applicable for this IP core.

## Banking

This section is not applicable for this IP core.

### Transceiver Placement

This section is not applicable for this IP core.

### I/O Standard and Placement

This section is not applicable for this IP core.

## Simulation

For comprehensive information about Vivado simulation components, as well as information about using supported third party tools, see the *Vivado Design Suite User Guide, Logic Simulation* (UG900) [Ref 9].

## Synthesis and Implementation

For details about synthesis and implementation, see the *Vivado Design Suite User Guide, Designing with IP* (UG896) [Ref 7].

# Example Design

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

## 10-Gigabit Ethernet MAC with External XGMII Interface

Figure 5-1 illustrates the example design and HDL wrapper for the core using the XGMII interface.

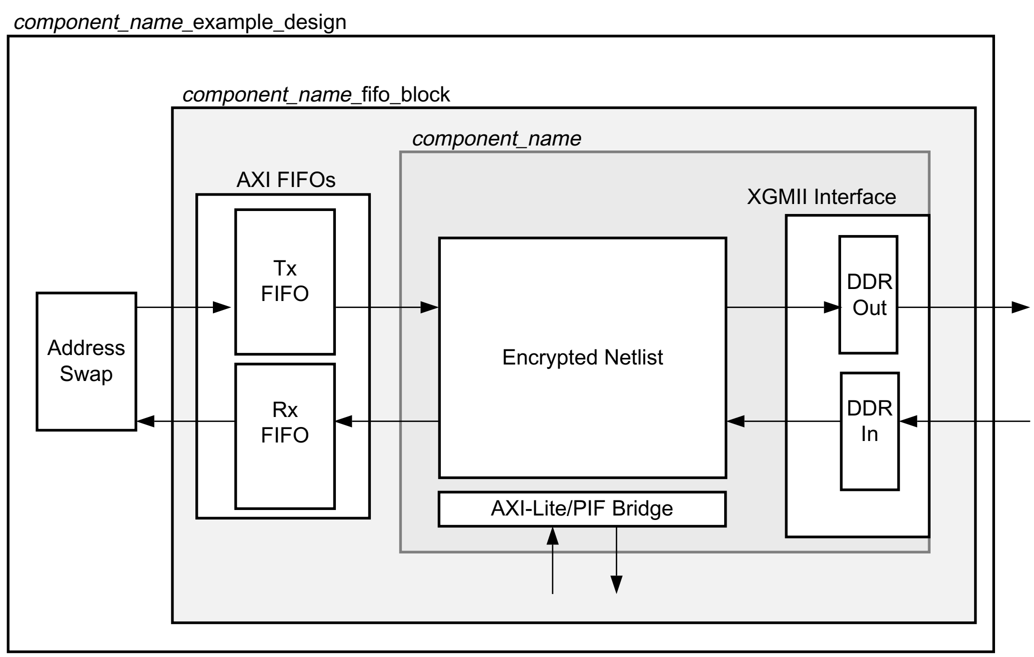


*Figure 5-1:* **Example Design and HDL Wrapper for 10-Gigabit Ethernet MAC with XGMII Interface**

The example design and HDL wrapper contain the following:

- Transmit-side 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

## 10-Gigabit Ethernet MAC with 64-Bit SDR Interface

Figure 5-2 illustrates the example design and the HDL wrapper for the core using a 64-bit interface.

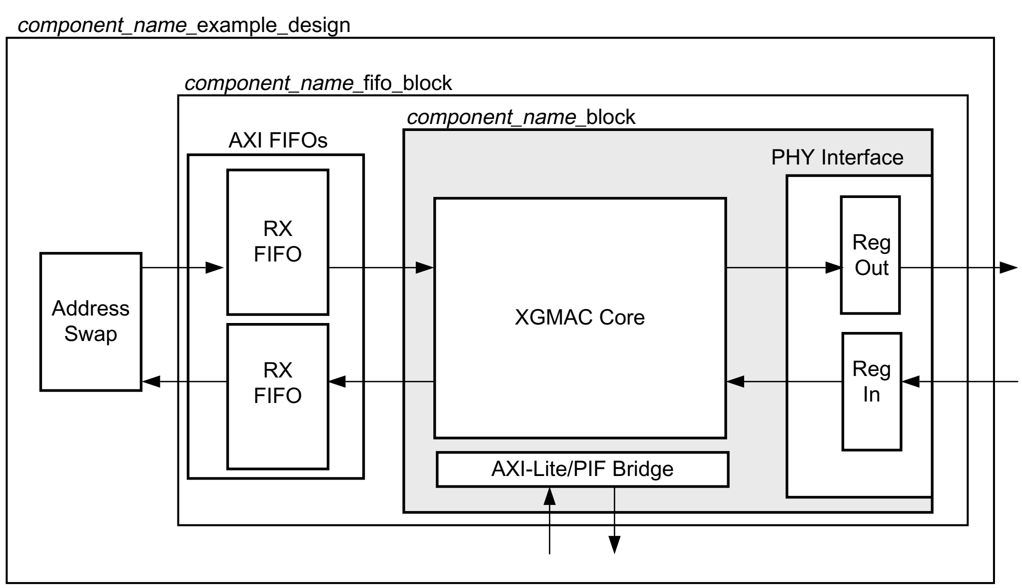


*Figure 5-2:* **Example Design and HDL Wrapper for 10-Gigabit Ethernet MAC with 64-bit Interface**

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:

- Pipeline registers for the 64-bit SDR interface
- Receiver DCM/MMCM and clock buffer

Chapter 6

# Test Bench

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

## 10-Gigabit Ethernet MAC Test Bench Functionality

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 6-1).

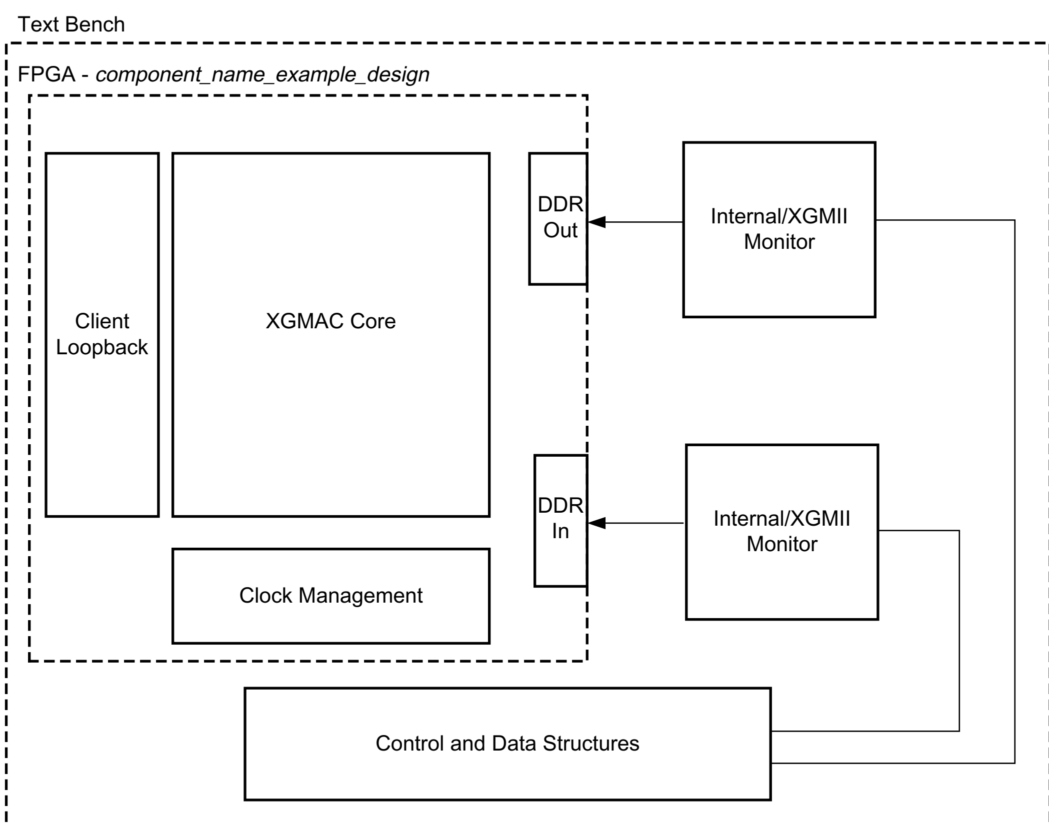


*Figure 6-1:* **Demonstration Test Bench for 10-Gigabit Ethernet MAC**

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.

The demonstration test bench filename is called <component_name>_demo_tb.v[hd]

The test bench consists of the following:

- Clock generators
- A stimulus block that connects to the XGMII or Internal receiver interface of the example design.
- A monitor block to check data returned through the XGMII or Internal transmitter interface of the example design.
- A management block to control core configuration (by writing to the configuration registers) if management is enabled for the core.

The test bench also contains a CRC calculation function (not illustrated) that is used both by the stimulus and monitor blocks.

## Test Bench Tasks

The demonstration test bench performs the following tasks:

- Input clock signals are generated.
- A reset is applied to the example design.
- Four predefined stimulus frames are pushed into the XGMII or Internal receiver interface
    - The first frame is a minimum length frame.
    - The second frame is a type frame.
    - The third frame is an errored frame.
    - The fourth frame is a padded frame.
- As the frame is pushed into the receiver interface by the stimulus block, the CRC is calculated over the data and inserted into the FCS field at the end of the Ethernet frames.
- The frames received at the XGMII or internal transmitter interface are checked against the original stimulus frames to ensure that the data is not corrupted. The monitor process takes into account the Source Address (SA)/ Destination Address (DA) fields, which might be swapped. The monitor also ensures the validity of the inserted FCS field of the MAC by calculating its own independent CRC over the entire observed frame (including over the last four bytes of the data field). This results in a predetermined constant value if there are no errors.

## Core with Statistics

The test bench performs the additional task of checking the number of good and errored frames on both receive and transmit.

## Changing the Test Bench

### *Changing Frame Data*

The contents of the frame data passed into the MAC receiver can be changed by editing the DATA fields for each frame defined in the test bench. The test bench automatically calculates the new FCS field to pass into the MAC. Further frames can be added by defining new frames of data.

### *Changing Frame Length*

There are two ways to change the frame lengths of the stimulus frames. The predefined frames within the test bench can be directly edited to change the frame contents and size. Alternatively, the test bench contains a quicker method that allows blocks of data within the predefined frames to be repeated a given number of times. The block of data which can be repeated is inclusive from the beginning of the TYPE/LENGTH field to the end of the final full XGMII column of data payload. The FRAME_GEN_MULTIPLIER constant provides the information for the number of times to repeat this block of data.

For example, if the index of the last complete control (ctrl) column (for example, ctrl column[14] = 4'b1111) then the block size is (14 + 1) - 3 = 12. That means that 12 x 4 = 48 bytes are contained in one block. If FRAME_GEN_MULTIPLIER is set to 2 then $2 \times 12 \times 4 = 96$ bytes are sent after the SA/DA and the same 48 byte block repeating pattern is sent twice.

If using a LENGTH field rather than a TYPE field for the TYPE/LENGTH field of the defined frame, then the LENGTH value must be manually edited. The general formula for LENGTH/TYPE field is as follows:

[[(index of last complete ctrl column + 1) - 3] x 4 x FRAME_GEN_MULTIPLIER] - 2 + (1,2 or 3 depending from the value of the ctrl column after the last complete ctrl column).

The multiplier constant is applied to every frame inserted into RX; therefore the L/T field has to be set appropriately for every frame unless the frame is a type or a control frame.

### *Enabling In-Band FCS*

The test bench by default operates with In-Band FCS passing disabled. It can be enabled by changing the value of global parameter IN_BAND_FCS_PASSING. When enabled, the frame address fields are not swapped and the address swap check in the monitor block is disabled. In this way the CRC observed on transmit is correct and matches the CRC of the receiver. If the core is configured with the management interface included, then the test bench automatically configures the core for In-Band FCS passing on both transmit and receive. If using the alternative configuration vector, then this mode can still be run but it is left for the user to tie the In-Band FCS Enable bits High (bit[3] of both the transmitter and receiver configuration vectors (see Figure 3-10 and Figure 3-11)).

### *Changing Frame Error Status*

Errors can be inserted into any of the pre-defined frames by changing the error field to 1 in any column of that frame. When an error is introduced into a frame, the ERROR field for that frame must be set to disable the monitor checking for that frame. The error currently written into the third frame can be removed by setting the error field for the frame to 0.

# Verification, Compliance, and Interoperability

This appendix includes information about how the IP was tested for compliance with the protocol to which it was designed.

## 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 7 series silicon with the Xilinx LogiCORE™ IP XAUI, RXAUI, and 10-Gb Ethernet PCS/PMA cores. The design comprises the Ethernet MAC, PHY core, 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.

# Migrating and Upgrading

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

## Migrating to the Vivado Design Suite

For information on migrating to the Vivado Design Suite, see *ISE to Vivado Design Suite Migration Guide* (UG911) [Ref 10].

## Upgrading in the Vivado Design Suite

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

### Parameter Changes

In v12.0, the AXI-Lite address buses were 32 bits wide and the C_BASE_ADDRESS parameter to the core set the base decoded address for the AXI4-Lite/IPIF bridge logic.

In v13.0, the AXI-Lite address bus widths have been reduced to 11 bits, which is the address width required for all of the memory mapped registers within the core. It is now required that address decoding is handled externally to the core (for example, by the AXI4 Interconnect IP core in Vivado IP integrator) and that only the lower-order address bits are passed through to the AXI-Lite/IPIF bridge.

# 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 C-1 defines the system margin.

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

## 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.

**RECOMMENDED:** *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° 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.

**RECOMMENDED:** *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.*

# Debugging

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

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

## Finding Help on Xilinx.com

To help in the design and debug process when using the 10-Gigabit Ethernet MAC, the Xilinx Support web page (www.xilinx.com/support) contains key resources such as product documentation, release notes, answer records, information about known issues, and links for obtaining further product support.

### 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.

### 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.

## Answer Records

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

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

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

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

#### Master Answer Record for the 10-Gigabit Ethernet MAC

AR: 54252

## 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 Additional Resources.

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.

***Note:*** Access to WebCase is not available in all cases. Log in to the WebCase tool to see your specific support options.

# 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.

## Vivado Lab Tools

Vivado lab tools insert logic analyzer and virtual I/O cores directly into your design. Vivado lab tools also allow you to set trigger conditions to capture application and integrated block port signals in hardware. Captured signals can then be analyzed. This feature in the Vivado IDE is used for logic debugging and validation of a design running in Xilinx devices.

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

- ILA 2.0 (and later versions)
- VIO 2.0 (and later versions)

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

# Simulation Debug

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

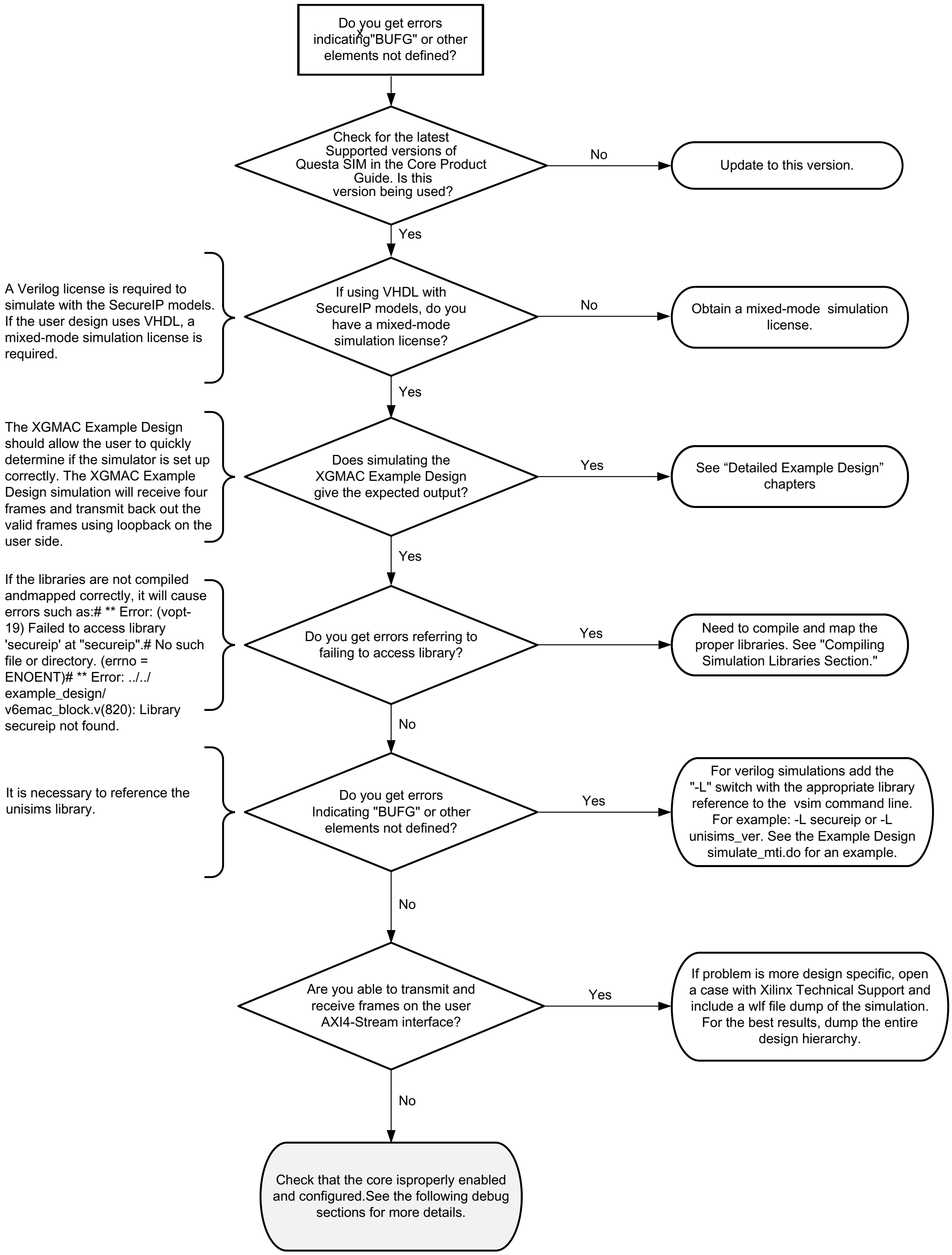


*Figure D-1:* **Questa SIM Debug Flow Diagram**

## 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 Vivado Integrated Design Environment 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 virtex7 -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 Vivado lab tools are a valuable resource to use in hardware debug and the signal names mentioned in the following individual sections can be probed using the Vivado lab tools for debugging the specific problems.

## 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 (*PG053) [Ref 3] or *LogiCORE IP RXAUI Users Guide* (PG083) [Ref 4] 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 Vivado lab tools can be inserted to get more details on the bad frames.
- Add the Vivado lab tools to the design to look at the RX and TX AXI4-Stream and physical interface data signals, control signals and statistics vectors.
- Add the Vivado lab tools to the design to look at the XGMII interface signals:
  - xgmii_txd
  - xgmii_txc
  - xgmii_rxd
  - xgmii_rxc

## 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 Vivado lab tools 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 *IEEE 802.3-2008* 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 51.

### 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 46.
- Check if Deficit Idle Count has been enabled to reduce average IFG transmitted. For more information, see Deficit Idle Count (DIC), page 50.

## 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 3-27 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 Vivado lab tools 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 3-9).
- Check core configuration.

Appendix E

# Additional Resources and Legal Notices

## Xilinx Resources

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

For a glossary of technical terms used in Xilinx documentation, see the Xilinx Glossary.

## 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"
2. *IEEE Standard 802.1Qbb,* "IEEE Standard for Local and Metropolitan Area Networks—Virtual Bridged Local Area Networks—Amendment: Priority-based Flow Control"


These documents provide supplemental material useful with this product guide:


3. *LogiCORE IP XAUI Product Guide (*PG053)
4. *LogiCORE IP RXAUI Product Guide* (PG083)
5. *LogiCORE IP Ten Gigabit Ethernet PCS/PMA Product Guide* (PG068)
6. *Vivado Design Suite User Guide, Designing IP Subsystems Using IP Integrator* (UG994)
7. *Vivado Design Suite User Guide, Designing with IP* (UG896)
8. *Vivado Design Suite User Guide, Getting Started* (UG910)
9. *Vivado Design Suite User Guide, Logic Simulation* (UG900)
10. *ISE to Vivado Design Suite Migration Guide* (UG911)
11. *Vivado Design Suite User Guide, Programming and Debugging* (UG908)
12. *LogiCORE IP Ethernet 1000BASE-X PCS/PMA or SGMII Product Guide (*PG047*)*

## Revision History

The following table shows the revision history for this document.

| Date | Version | Revision |
|---|---|---|
| 04/02/2014 | 13.1 | • Added Priority Flow Control description<br>• Added new functionality description to 802.3 Pause control description<br>• Updated Demonstration test bench<br>• Updated graphics for interfacing to other cores<br>• Updated length check description |
| 10/02/2013 | 13.0 | Revision number advanced to 13.0 to align with core version number.<br>• Added IP Integrator.<br>• Added License Checkers section.<br>• Updated Resource Utilization section.<br>• Updated Table 2-6: Management Interface Port Descriptions.<br>• Updated to six clocks in Transmit Path Latency section.<br>• Updated tables in Clocking and Reset Signals section.<br>• Added Shared Logic section.<br>• Updated description in Receive section.<br>• Updated Fig. 3-41 Clock Management, Multiple Instances of the Core with XGMII.<br>• Updated shared clocking in Interfacing to the XAUI IP Core, Interfacing with the RXAUI Core, and 10-Gigabit Ethernet PCS/PMA Core sections.<br>• Updated Fig. 3-44 10-Gigabit Ethernet MAC Core Integrated with PCS/PMA Core – Virtex-7 and Kintex-7 FPGAs.<br>• Updated Fig. 4-1 and added Shared Logic in Customizing and Generating the Core.<br>• Added Simulation and Synthesis chapters.<br>• Updated supported devices in Hardware Verification section in Verify Appendix.<br>• Updated Migration Appendix.<br>• Updated Debug Appendix. |

| Date | Version | Revision |
|---|---|---|
| 03/20/2013 | 2.1 | • Updated core to v12.0 and this release is for Vivado Design Suite only.<br>• Updated Fig. 1-1 Implementation of the 10-Gigabit Ethernet MAC Core.<br>• Updated Fig. 2-2 Implementation of the Core with User Logic on PHY Interface.<br>• Updated tx_axis_tuser[0:0] in Table 2-2 AXI4-Stream Interface Ports – Transmit.<br>• Updated Table 2-5 PHY Interface Port Descriptions.<br>• Updated to 300 MHz in Table 2-6 Management Interface Port Descriptions.<br>• Updated Table 2-11 Clock, Clock Management, and Reset Ports.<br>• Updated Fig. 3-1 Clocking Logic for the Ethernet MAC Internal Interface Option.<br>• Updated to 300 MHz in Table 3-6 Management Interface Port Description.<br>• Added Bit[31] in Table 3-10 tx_configuration_vector Bit Definitions.<br>• Added Bit[31] in Table 3-11 rx_configuration_vector Bit Definitions.<br>• Updated Bits[19:5] in Table 3-13 Transmit Statistics Vector Bit Description.<br>• Updated Fig. 3-38 Clock Management, Multiple Instances of the Core with XGMII.<br>• Updated GUI in Fig. 4-1.<br>• Added description in Required Constraints.<br>• Updated Fig. 6-1 Example Design and HDL Wrapper for 10-Gigabit Ethernet MAC with XGMII Interface.<br>• Updated to Questa SIM. |
| 12/18/2012 | 2.0 | • Updated core to v11.5 and this release is for Vivado Design Suite v2012.4 only.<br>• Updated License and Ordering Information.<br>• Updated to support 7 series FPGAs only.<br>• Updated tx_axis_tready direction to out in Table 3-1.<br>• Updated Fig. 4-1 GUI and WAN support description.<br>• Updated Migrating Appendix.<br>• Added new Debug section and minor document updates. |
| 07/25/2012 | 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*. |

# Please Read: Important Legal Notices