Virtex-4 FPGA Embedded Tri-Mode Ethernet MAC

User Guide

UG074 (v2.2) February 22, 2010

XILINX®

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
11/11/04	1.0	Initial Xilinx release.
07/05/05	1.1	In Chapter 2: • Revised Table 2-6, page 23, "Tie-Off Pins," page 24, and Table 2-10, page 25. In Chapter 3: • Revised Table 3-6 and Figure 3-5. Added information to "Normal Frame Transmission". Revised Figure 3-6, "Client Underrun", Figure 3-7, and Figure 3-9. • Added information to "Transmit (TX) Client – 16-bit Wide Interface," page 43. • Revised "SGMII/1000BASE-X PCS/PMA," page 48 and Figure 3-18, page 49, Figure 3-19, page 49, and Figure 3-21, page 50. • Added information to "Flow Control Block," page 57 and the "Flow Control Implementation Example," page 60. • Revised Figure 3-32, page 59. • Revised "Generic Host Bus," page 70. • Revised Table 3-17, page 73, and Table 3-20, page 76. • Revised Table 3-22, page 78 through Table 3-31, page 83. • Added "Connecting the MDIO to an External PHY" and Figure 3-49, page 96. • Revised Ethernet MAC I/Os in Figure 3-55, page 97, Figure 3-60, page 103, Figure 3-65, page 112, Figure 3-71, page 120, and Figure 3-75, page 128. • Revised Figure 3-66, page 113. • Added information to "Auto-Negotiation Interrupt," page 145. • Added information to "SGMII Standard," page 146. In Chapter 4: • Revised sample code in "Interfacing to the Processor DCR," page 150. In Chapter 5: • Replaced Figure 5-2, page 154 and Figure 5-3, page 156. Added Appendix A, "Ethernet MAC Timing Model."
07/20/05	1.2	Revised figures.
09/07/05	1.3	Corrected items from version 1.1 and 1.2. Inserted eight new figures into Chapter 5, "Miscellaneous Functions." Updated pin 76 in Table 2-9. Revised Figure 3-56, Figure 3-57, Figure 3-61, Figure 3-63, Figure 3-67, Figure 3-68, Figure 3-72, Figure 5-1, Figure 5-2, and Figure 5-3. Added "Tri-Mode RGMII v1.3" in Chapter 3. Moved the "Using the DCR Bus as the Host Bus", "Description of Ethernet MAC Register Access through the DCR Bus", and "Address Code" sections to the "Host Interface" section.

Date	Version	Revision
02/07/06	1.4	In Chapter 2, "Ethernet MAC Architecture": • Added EMAC#CLIENTRXCLIENTCLKOUT to Table 2-2, page 19. In Chapter 5, "Miscellaneous Functions": • Corrected "Client RX Data/Control Interface," page 55. • Inserted new text to "Length/Type Field Error Checks," page 52, "Tri-Mode RGMII v2.0," page 115, and "10/100/1000 SGMII Clock Management," page 121. • Edited waveforms in Figure 3-39, page 66. • Added sections to Table 3-36, page 88. • Added "Introduction to MDIO," page 90, "MDIO Implementation in the EMAC," page 92, "Tri-Mode Operation with Byte PHY Enabled (Full-Duplex Only)," page 109, and "1000BASE-X PCS/PMA (8-bit Data Client) Clock Management," page 126. • Added IDELAY component to Figure 3-61, Figure 3-62, Figure 3-63, and Figure 3-64. • Revised the GT11 block in Figure 3-72 and Figure 3-73. Modified text in "10/100/1000 SGMII Clock Management," page 121, "8-Bit Data Client," page 129, and "16-Bit Data Client," page 130. In Chapter 6, "Use Models": • Replaced "Interfacing to an FPGA Fabric-Based Statistics Block," page 154 with new content. • Revised example code in "Writing to the PHY Registers Using MDIO," page 153. In Chapter 7, "Ethernet MAC Wrappers": • Added new text to "VHDL and Verilog CORE Generator Wrappers," page 159. • Added "Advanced Clocking," page 155 and "Client Side Data Width," page 157 options. • Replaced "File Generation" with new content. • Replaced Figure 5-1, page 153.
02/09/06	1.4.1	Cleaned up formatting issues. No content changes.

Date	Version	Revision

11/11/06	1.5	• Book restructuring of Chapter 3 content:
		  • Sections on Host, Client, and MDIO interfaces remain in Chapter 3, now called "Client, Host, and MDIO Interfaces."
		  • Pulled out section on Physical Interface to new Chapter 4, "Physical Interface."
		  • Pulled out sections on Clock Frequency Support, Ethernet MAC Configuration, and Auto-Negotiation Interrupt to new Chapter 5, "Miscellaneous Functions."
		• Globally changed all instances of PHYEMAC#TXD to EMAC#PHYTXD.
		• Section "Features" in Chapter 1: Added full-duplex qualifier for SGMII support.
		• Table 3-30: Added EMAC1 registers IRSTATUS, IRENABLE, MIIMWRDATA, and MIIMCNTL.
		• Updated/corrected Figure 3-48, Figure 4-1, Figure 4-3, Figure 4-22, and Figure 4-24.
		• Section "SGMII RX Elastic Buffer" added to Chapter 4, "Physical Interface."
		• Corrected section "1 Gb/s RGMII Clock Management" in Chapter 4.
		• Section "10/100/1000 SGMII Clock Management" in Chapter 4: The external, high-quality reference clock for the RocketIO transceiver changed from 125 MHz to 250 MHz. The GT11 clock schemes are simplified as shown in Figure 4-24.
		• Section "1000BASE-X PCS/PMA (8-bit Data Client) Clock Management" in Chapter 4: The GT11 clock schemes are simplified as shown in Figure 4-27.
		• Section "1000BASE-X PCS/PMA Clock Management" in Chapter 4: The external, high-quality reference clock for the 16-bit Data Client MGT changed from 125 MHz to 250 MHz. The GT11 clock schemes are simplified as shown in Figure 4-28.
		• Table 4-13: Corrected 4.8:7 Pause, `10` and `11` (reversed)
		• Section "MGT Elastic Buffer (Ring Buffer)" in Chapter 4: Corrected underflow/overflow marks, and as a result corrected maximum frame size (Table 4-4).
		• Section "Transmit Clocking Scheme" in Chapter 5: Corrected description of inputs used to generate client-side clock.
		• Section "Reading the PHY Registers Using MDIO" in Chapter 6: Corrected decode address.
		• Section "Writing to the PHY Registers Using MDIO" in Chapter 6: Corrected register address for EMAC1 Management Configuration; corrected decode address.
		• Chapter 7, "Using the Embedded Ethernet MAC" updated.

03/27/07	1.6	• Table 2-9: Removed Note (1). • Table 2-11: Revised Note (1). • Table 3-5: Removed exception from description of Length/Type Out of Range. • Table 3-20: Corrected default value for bit 31 (Promiscuous Mode Enable). • Chapter 4: Multiple revisions in block schematic diagram figures. • Chapter 4: Former section "1000BASE-X PCS/PMA (8-bit Data Client) Clock Management" in Chapter 4 incorporated into section "8-Bit Data Client." • Table 4-3: Consolidated "RISING" and "FALLING" RGMII signals into single signals with revised descriptions. Added Note (1) to RGMII_TX_CTL_# and RGMII_RX_CTL_#.

Date	Version	Revision
08/20/07	1.7	• Replaced screen shots with updated images from v4.5 software. • Table 2-11: Revised description of TIEEMAC#CONFIGVEC[63]. • Section "Length/Type Field Error Checks" in Chapter 3: Numerous textual revisions throughout this section. • Section "Receiving a PAUSE Control Frame" in Chapter 3: Last paragraph revised. • Table 3-9: Revised definition of Receiver Configuration register bit [25] (LT_DIS). • Figure 4-7 through Figure 4-10 and Figure 4-12 through Figure 4-16 and accompanying text: Added IDELAY element to clock input. • Figure 4-24: Corrected placement of BUFG to TXCLIENTCLKOUT. • Made other typographical edits.
02/06/08	1.8	• Added conditions causing assertion of EMAC#CLIENTRXBADFRAME to "Frame Reception with Errors," page 53. • Rewrote description of ALIGNMENT_ERROR bit in Table 3-5, page 70. • Updated Figure 4-28, page 139. • Added "Core Latency" in Appendix A.
08/13/08	1.9	• Corrected transposition error in the description column for bit [1] in Table 3-13, page 78 (values `0` and `1` were switched in second sentence). • Corrected transposition error in the description column for bits 5.8:7 in Table 4-14, page 144 (values `01` and `10` were switched). • Rewrote item number 1 for 1000BASE-X auto-negotiation summary ("Overview of Operation," page 151). • Updated the link in the first bullet of "Global Buffer Usage," page 165.
05/12/09	2.0	• Chapter 4: In sections "1 Gb/s GMII Only," page 107, "Tri-Mode Operation," page 109, "Tri-Mode Operation with Byte PHY Enabled (Full-Duplex Only)," page 110, "1 Gb/s RGMII Clock Management," page 115, "Tri-Mode RGMII v2.0," page 117, and "Tri-Mode RGMII v1.3," page 120, changed the description of the method used to delay the received clock (GMII_RX_CLK in case of GMII and RGMII_RXC in case of RGMII) with respect to data. Original method described use of IDELAY and BUFG. Revised method describes use of a DCM and BUFG. In Figure 4-7, page 107, Figure 4-8, page 108, Figure 4-9, page 110, Figure 4-10, page 111, Figure 4-12, page 115, Figure 4-13, page 116, Figure 4-14, page 118, Figure 4-15, page 119, Figure 4-16, page 121, and "16-Bit Data Client," page 139, Replaced IDELAY block in clock path with DCM. • Chapter 6: "Simulation Models," page 155: Removed description of SmartModels. Replaced references to SmartModel with references to SecureIP throughout this section. Added "SecureIP Model," page 155. • Chapter 7: Removed content discussing Ethernet MAC wrappers and replaced it with"Using the Embedded Ethernet MAC," page 167.

Date	Version	Revision
10/28/09	2.1	• Chapter 2: Updated the description of TIEEMAC#CONFIGVEC[62] and TIEEMAC#CONFIGVEC[61] Table 2-11, page 30 with the full-duplex mode requirement. • Chapter 3: Updated"IFG Adjustment," page 46 and "Flow Control Block," page 61 with the full-duplex mode requirement.
02/22/10	2.2	• Chapter 4: • Replaced EMAC#CLIENTTXGMIIMIICLKOUT with GTX_CLK in "Tri-Mode Operation," page 109. • Updated BUFGMUX and GTX_CLK in upper portion of Figure 4-9, page 110. • Updated description of Link Status and added table note to Table 4-11, page 141. • Appendix A: Changed T_{xMACWL} to Clock Low and T_{xMACWH} to Clock High in Figure A-1, page 171.

Table of Contents

Chapter 4: Physical Interface

Chapter 5: Miscellaneous Functions

Chapter 6: Use Models

About This Guide

This document is the Virtex®-4 FPGA Embedded Tri-Mode Ethernet MAC User Guide.

Guide Contents

This user guide contains the following chapters:

- Chapter 1, "Introduction," introduces the Virtex-4 FPGA Embedded Tri-Mode Ethernet MAC and summarizes its features.
- Chapter 2, "Ethernet MAC Architecture," describes the architecture of the Ethernet MAC and defines its signals.
- Chapter 3, "Client, Host, and MDIO Interfaces," provides design information for the client, host, and MDIO interfaces.
- Chapter 4, "Physical Interface," describes design considerations for the supported interfaces when using the Ethernet MAC.
- Chapter 5, "Miscellaneous Functions," provides useful information for designing with the Ethernet MAC.
- Chapter 6, "Use Models," describes some available models and how to interface the Ethernet MAC to a processor DCR or an FPGA statistics block.
- Chapter 7, "Using the Embedded Ethernet MAC," describes the Verilog and VHDL CORE Generator™ wrappers.
- Appendix A, "Ethernet MAC Timing Model," provides details on the timing parameters of the Ethernet MAC timing model.

Additional Resources

For additional information, go to http://www.xilinx.com/support.

Conventions

This document uses the following conventions. An example illustrates each convention.

Typographical

The typographical conventions in the following table are used in this document:

Convention	Meaning or Use	Example
Italic font	References to other manuals	See the *Virtex-4 Data Sheet* for more information.
	Emphasis in text	If a wire is drawn so that it overlaps the pin of a symbol, the two nets are *not* connected.
Byte/bit	"B" means "byte,"and "b" means "bit"	8B word 64b word
	EXCEPTIONS:	8B/10B = 8 bit/10 bit 64B/66B = 64 bit/66 bit
Hex and binary notation	Hex and binary numbers are in a monospace typeface. The `0x` prefix is often used to designate a hex string.	`0` or `1` [binary] `11010011` [binary] `0x0123456789ABCDEF`

Vertical ellipsis
.
.
.
	Repetitive material that has been omitted	`IOB #1: Name = QOUT'`
		`IOB #2: Name = CLKIN'`
		.
		.
		.

Horizontal ellipsis ...	Repetitive material that has been omitted	**`allow block`** *`block_name loc1 loc2 ... locn;`*

Online Document

The cross-reference and linking conventions shown in the following table are used in this document:

Convention	Meaning or Use	Example
Blue text	Cross-reference link to a location in the current document	See the section "Additional Resources" for details. Refer to "Title Formats" in Chapter 1 for details.
Red text	Cross-reference link to a location in another document	See the *Virtex-4 User Guide*.
Blue, underlined text	Hyperlink to a website (URL)	Go to http://www.xilinx.com for the latest speed files.

Chapter 1

Introduction

This chapter introduces the Virtex®-4 FPGA Embedded Tri-Mode Ethernet Media Access Controller (MAC). It contains the following sections:

- "Ethernet MAC Overview"
- "Features"

Ethernet MAC Overview

Figure 1-1 shows the Virtex-4 FPGA Tri-Mode Ethernet MAC that is used to provide Ethernet connectivity to the Virtex-4 FX family of devices.

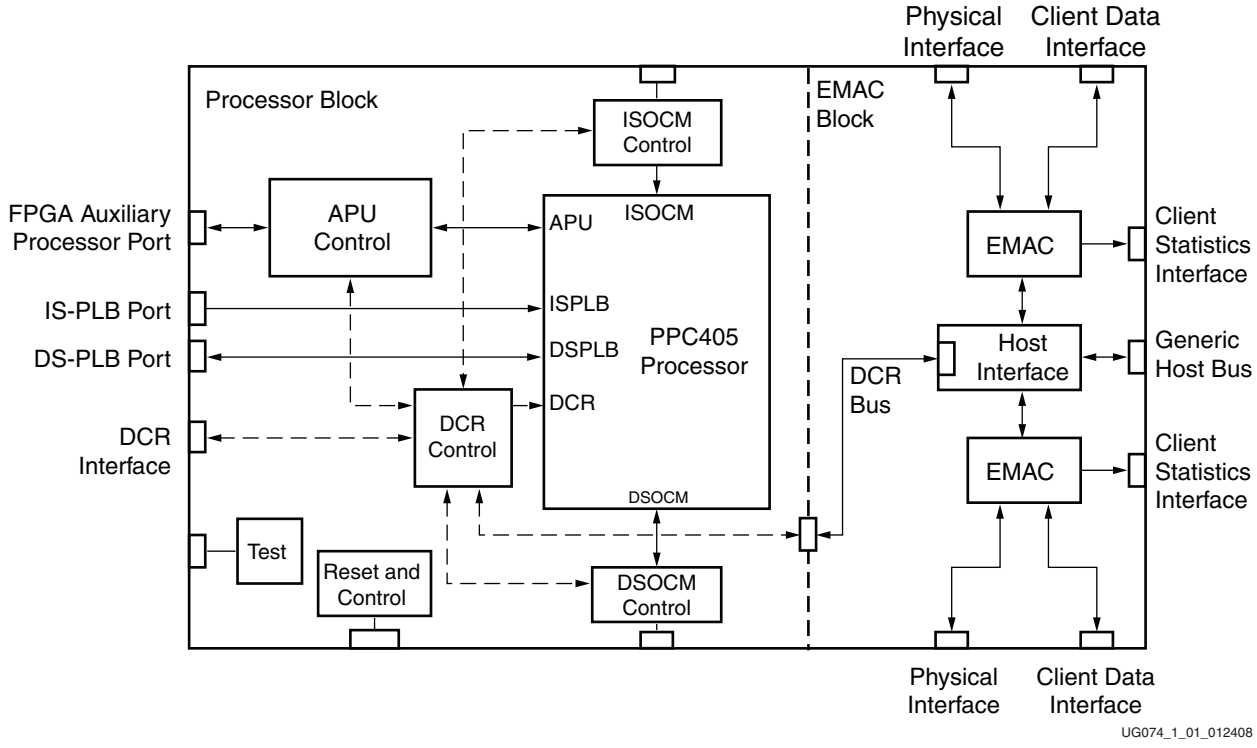


Figure 1-1: **Virtex-4 FPGA Embedded Tri-Mode Ethernet MAC**

The Virtex-4 FPGA Ethernet MAC has two Ethernet MACs sharing a single host interface. Either or both of the Ethernet MACs can be selected to access the Ethernet MAC registers.

Features

The key features of the Virtex-4 FPGA Ethernet MAC are:

- Fully integrated 10/100/1000 Mb/s Ethernet MAC
- Designed to the IEEE Std 802.3-2002 specification
- Configurable full-duplex operation in 10/100/1000 Mb/s
- Configurable half-duplex operation in 10/100 Mb/s
- Management Data Input/Output (MDIO) interface to manage objects in the physical layer
- User-accessible raw statistic vector outputs
- Support for VLAN frames
- Configurable inter-frame gap (IFG) adjustment in full-duplex operation
- Configurable in-band Frame Check Sequence (FCS) field passing on both transmit and receive paths
- Auto padding on transmits and stripping on receives
- Configured and monitored through a host interface
- Hardware-selectable Device Control Register (DCR) bus or generic host bus interface
- Configurable flow control through Ethernet MAC Control PAUSE frames; symmetrically or asymmetrically enabled
- Configurable support for jumbo frames of any length
- Configurable receive address filter for unicast, multicast, and broadcast addresses
- Media Independent Interface (MII), Gigabit Media Independent Interface (GMII), and Reduced Gigabit Media Independent Interface (RGMII)
- Includes a 1000BASE-X Physical Coding Sublayer (PCS) and a Physical Medium Attachment (PMA) sublayer for use with the RocketIO™ Multi-Gigabit Transceiver (MGT) to provide a complete on-chip 1000BASE-X implementation
- Serial Gigabit Media Independent Interface (SGMII) supported through MGT interface to external copper PHY layer for full-duplex operation only

XILINX®

Chapter 2

Ethernet MAC Architecture

This chapter describes the architecture of the Virtex®-4 FPGA Embedded Tri-Mode Ethernet Media Access Controller (MAC). It contains the following sections:

- "Architecture Overview"
- "Ethernet MAC Primitive"
- "Ethernet MAC Signal Descriptions"

Architecture Overview

The Virtex-4 FPGA Embedded Tri-Mode Ethernet MAC supports 10/100/1000 Mb/s data rates and is designed to IEEE Std 802.3-2002 specifications. The Ethernet MAC can operate as a single speed Ethernet MAC at 10, 100, or 1000 Mb/s or as a tri-mode Ethernet MAC. The Ethernet MAC supports the IEEE standard GMII and the RGMII protocols to reduce the width of the bus to the external physical interface. A 1000BASE-X PCS/PMA sublayer, when used in conjunction with the Virtex-4 FPGA RocketIO™ Multi-Gigabit Transceiver (MGT), provides a complete on-chip 1000BASE-X implementation.

Figure 2-1 shows a block diagram of the Ethernet MAC block. The block contains two Ethernet MACs sharing a single host interface. The host interface can use either the generic host bus or the DCR bus through the DCR bridge. Each Ethernet MAC has an address filter to accept or reject incoming frames on the receive datapath. The Ethernet MAC outputs raw statistic vectors to enable statistics gathering. The statistics vectors are multiplexed to reduce the number of pins at the block boundary. An external module (StatsIP0 and/or StatsIP1) can be designed and implemented in the FPGA fabric to accumulate all the statistics of the Ethernet MAC. The "Interfacing to an FPGA Fabric-Based Statistics Block" section provides additional information on the statistics block.

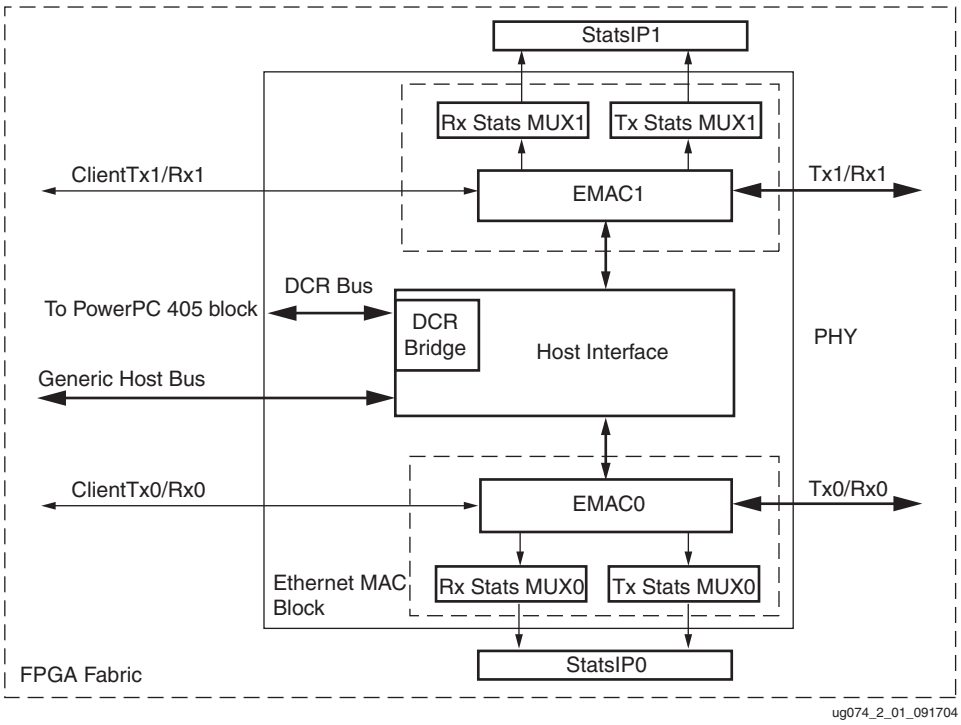


Figure 2-1: **Ethernet MAC Block**

A detailed block diagram of the 10/100/1000 Ethernet MAC is shown in Figure 2-2. On the physical side, it consists of the GMII and RGMII interfaces using standard I/Os to access data and control signals to an external physical interface. In addition, the PCS/PMA sublayer interfaces directly to the MGT.

The client side consists of the user transmit and receive interfaces. The flow control module keeps traffic from being congested in the Ethernet MAC. The Management Data I/O interface, (MDIO), allows access to the control and status registers in the external physical interface or in the PCS sublayer when configured in 1000BASE-X and SGMII modes.

The clock management module automatically configures the output clocks to the correct frequency based on the internal speed of the Ethernet MAC (10 Mb/s, 100 Mb/s, or 1000 Mb/s) and the Ethernet MAC mode settings (GMII, MII, RGMII, SGMII, and 1000BASE-X).

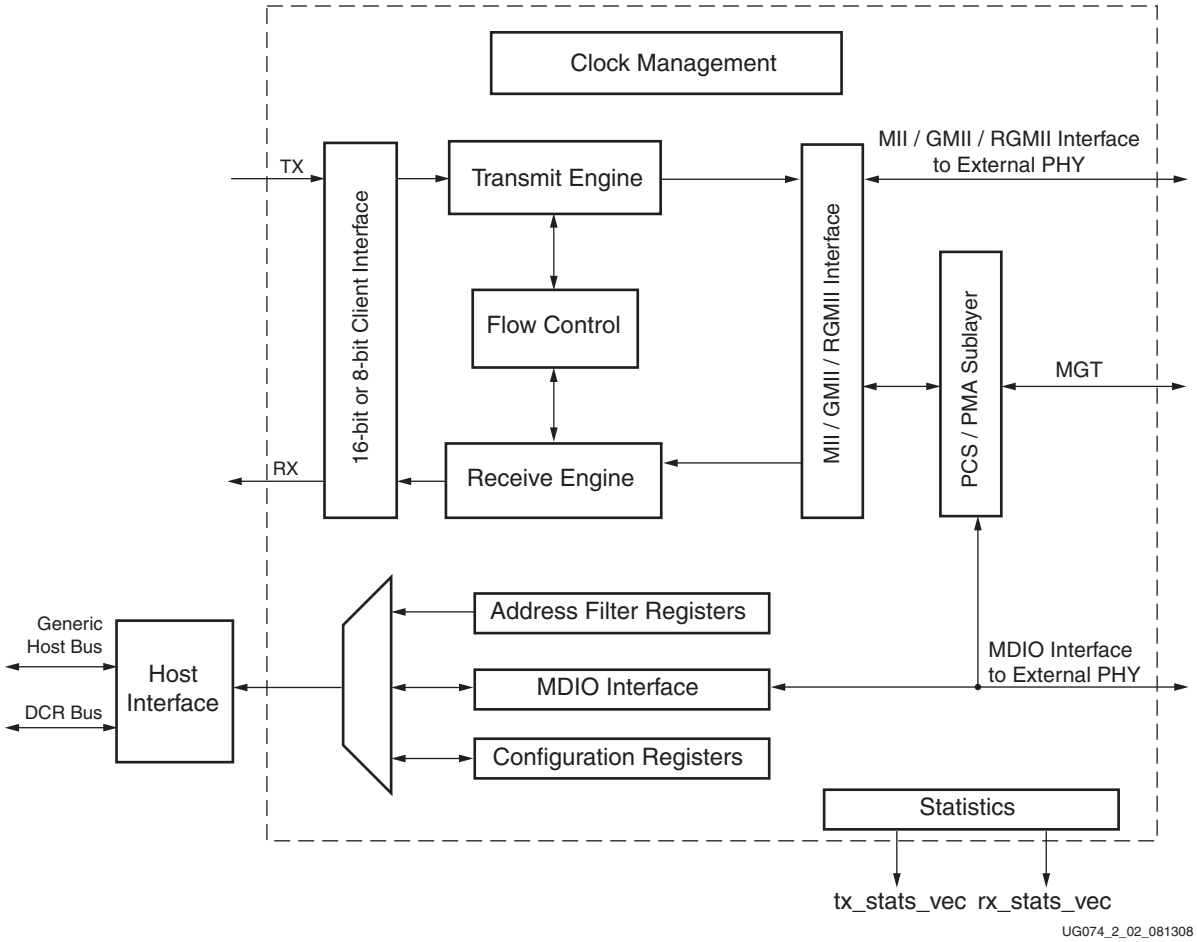


Figure 2-2: **Functional Block Diagram of 10/100/1000 Ethernet MAC**

Ethernet MAC Primitive

The Virtex-4 FPGA Embedded Tri-Mode Ethernet MAC has an EMAC primitive. The primitive contains access to both Ethernet MACs (EMAC0 and EMAC1).

By using the Ethernet MAC primitive, any of these supported interfaces can be created:

- Gigabit Media Independent Interface (GMII)
- Media Independent Interface (MII)
- Reduced Gigabit Media Independent Interface (RGMII)
- Serial Gigabit Media Independent Interface (SGMII)
- 1000BASE-X Physical Coding Sublayer (PCS) and Physical Medium Attachment (PMA)

Detailed connections to support/create these interfaces using the Ethernet MAC primitive are found in Chapter 4, "Physical Interface."

The Ethernet MAC primitive is discussed in different sections:

- **Receive / Transmit Client Interface**
- See "Client Interface" in Chapter 3.
- **Flow Control**

  See "Flow Control Block" in Chapter 3.
- **Generic Host Bus**

  Interface to any host (i.e., PowerPC® or MicroBlaze™ processor) to access the control and status of both Ethernet MACs. See "Host Interface" in Chapter 3.
- **DCR Bus**

  Interface to the PowerPC processor through the DCR bus to access the control and status of both Ethernet MACs. See "Using the DCR Bus as the Host Bus" in Chapter 3.
- **Physical Interface**

  Physical interface depending on the mode of configuration. See Chapter 4, "Physical Interface."
- **Multi-Gigabit Transceiver**

  Interface to the RocketIO transceivers when the Ethernet MAC is configured in either SGMII or 1000BASE-X PCS/PMA mode. See "10/100/1000 Serial Gigabit Media Independent Interface (SGMII)" or "1000BASE-X PCS/PMA" in Chapter 4.
- **Management Data I/O (MDIO)**

  Interface to the Management Data I/O of either an external physical interface or the PCS sublayer when configured in SGMII or 1000BASE-X PCS/PMA mode. See "MDIO Interface" in Chapter 3.

The Ethernet MAC primitive can be simplified for specific customer needs using Ethernet MAC wrappers. Chapter 7, "Using the Embedded Ethernet MAC," describes how Ethernet MAC wrappers use the CORE Generator™ software to simplify the Ethernet MAC primitives.

Figure 2-3 illustrates the Ethernet MAC primitive. The # sign denotes both Ethernet MACs (EMAC0 and EMAC1) in the EMAC primitive.

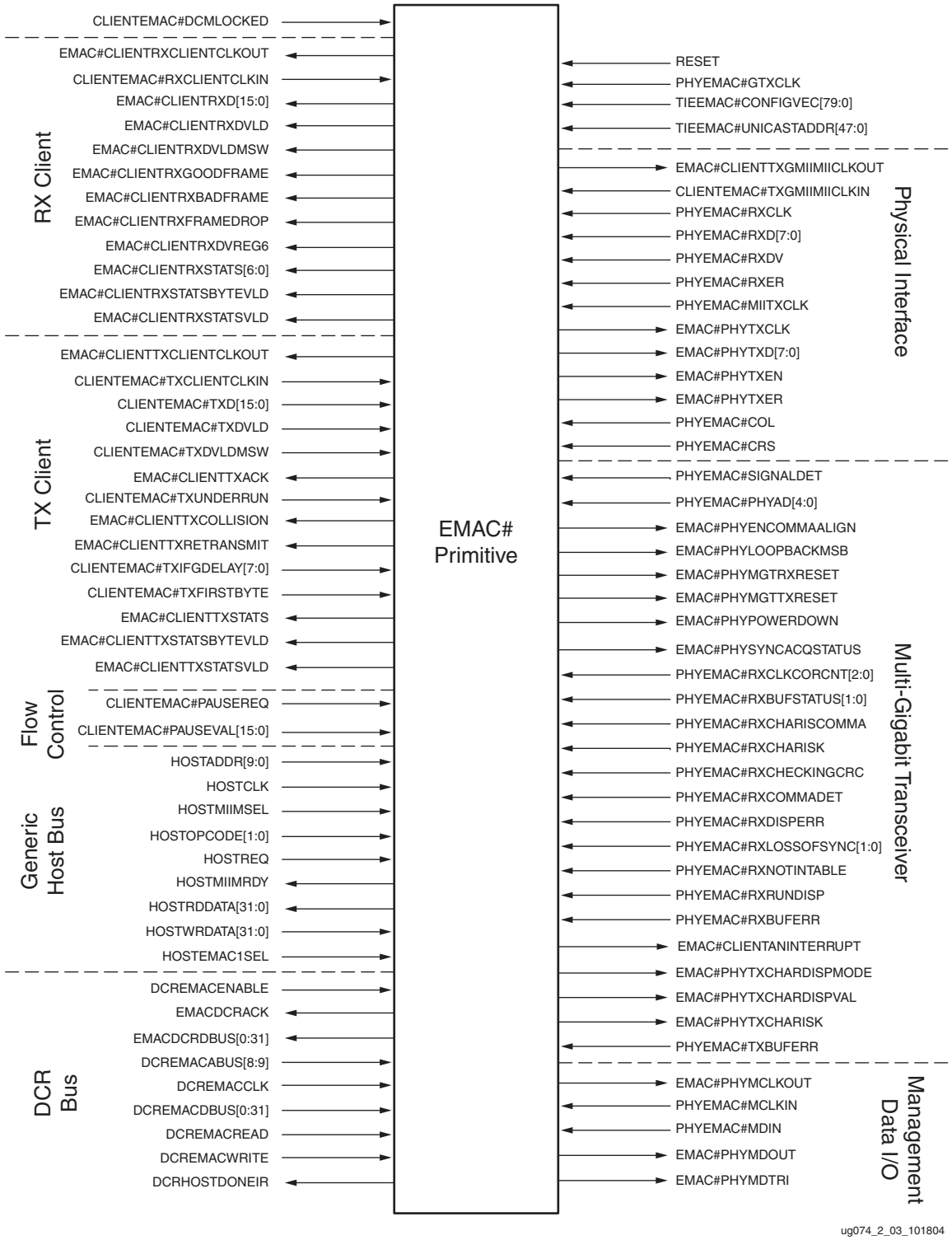


Figure 2-3: **Ethernet MAC Primitive**

Ethernet MAC Signal Descriptions

This section defines all of the Ethernet MAC primitive signals. The signals are divided into the following categories:

All the signals available in the Ethernet MAC primitive are described in this section. The # symbol denotes the EMAC0 or EMAC1 signals.

Client Signals

Client-Side Transmit (TX) Signals

Table 2-1 describes the client-side transmit signals in the Ethernet MAC. These signals are used to transmit data from the client to the Ethernet MAC.

Table 2-1: **Transmit Client Interface Signals**

Signal	Direction	Description
CLIENTEMAC#TXD[15:0]	Input	Frame data for transmit. The data path can be configured to be either 8 or 16 bits wide. Bits [7:0] are used for 8-bit width. The 16-bit interface is available only in 1000BASE-X PCS/PMA mode. See "Transmit (TX) Client – 16-bit Wide Interface" in Chapter 3.
CLIENTEMAC#TXDVLD	Input	Asserted by the client to indicate a valid data input at CLIENTEMAC#TXD[7:0].
CLIENTEMAC#TXDVLDMSW	Input	When the width of CLIENTEMAC#TXD is set to 16 bits wide, this signal denotes an odd number of bytes in the transmit data path. In the frame with an odd number of bytes, the CLIENTEMAC#TXD[7:0] is valid on the last byte. When the width of CLIENTEMAC#TXD is set to 8 bits wide, this signal is connected to ground.
CLIENTEMAC#TXFIRSTBYTE	Input	Is asserted High for one clock cycle to indicate the start of data flow on CLIENTEMAC#TXD. See Figure 3-3, "Normal Frame Transmission Across Client Interface" in Chapter 3. Can be grounded if not used.
CLIENTEMAC#TXIFGDELAY[7:0]	Input	Configurable inter-frame gap (IFG) adjustment for full-duplex mode.
CLIENTEMAC#TXUNDERRUN	Input	Asserted by the client to force the Ethernet MAC to corrupt the current frame.
CLIENTEMAC#TXCLIENTCLKIN	Input	See "Transmit Clocking Scheme" in Chapter 5.

Table 2-1: **Transmit Client Interface Signals *(Cont'd)***

Signal	Direction	Description
EMAC#CLIENTTXACK	Output	Handshake signal – Asserted when the Ethernet MAC accepts the first byte of data. On the next and subsequent rising clock edges, the client must provide the remainder of the frame data. See "Normal Frame Transmission" in Chapter 3.
EMAC#CLIENTTXCOLLISION	Output	Asserted by the Ethernet MAC to signal a collision on the medium. Any transmission in progress should be aborted. This signal is always deasserted in full-duplex mode.
EMAC#CLIENTTXRETRANSMIT	Output	Asserted by the Ethernet MAC at the same time as the EMAC#CLIENTTXCOLLISION signal. The client should re-supply the aborted frame to the Ethernet MAC for retransmission. This signal is always deasserted in full-duplex mode.
EMAC#CLIENTTXSTATS	Output	The statistics data on the last data frame sent. The 32-bit TX raw statistics vector is output by one bit per cycle for statistics gathering. See "Transmitter Statistics Vector" in Chapter 3.
EMAC#CLIENTTXSTATSBYTEVLD	Output	Asserted if an Ethernet MAC frame byte is transmitted (including destination address to FCS). This is valid on every TX clock cycle.
EMAC#CLIENTTXSTATSVLD	Output	Asserted by the Ethernet MAC after a frame transmission to indicate a valid EMAC#CLIENTTXSTATS output. See "Transmitter Statistics Vector" in Chapter 3.
EMAC#CLIENTTXCLIENTCLKOUT	Output	See "Transmit Clocking Scheme" in Chapter 5.

Client-Side Receive (RX) Signals

Table 2-2 describes the client-side receive signals. These signals are used by the Ethernet MAC to transfer data to the client.

Table 2-2: **Receive Client Interface Signals**

Signal	Direction	Description
CLIENTEMAC#RXCLIENTCLKIN	Input	See "Receive Clocking Scheme" in Chapter 5.
EMAC#CLIENTRXD[15:0]	Output	Frame data received from the Ethernet MAC. The data path can be configured to either 8 bits or 16 bits wide. Bits [7:0] are used for 8-bit width. The 16-bit interface is intended to be used in 1000BASE-X PCS/PMA mode. See "Receive (RX) Client – 16-bit Wide Interface" in Chapter 3.
EMAC#CLIENTRXDVLD	Output	The Ethernet MAC indicates to the client the receipt of valid frame data.
EMAC#CLIENTRXFRAMEDROP	Output	This signal is asserted to notify the client that an incoming receive frames destination address does not match any addresses in the address filter. The signal functions even when the address filter is not enabled.

Table 2-2: **Receive Client Interface Signals *(Cont'd)***

Signal	Direction	Description
EMAC#CLIENTRXDVLDMSW	Output	This signal denotes an odd number of bytes in the receive data path when the width of EMAC#CLIENTRXD is set to 16 bits wide. In a frame with an odd number of bytes, the EMAC#CLIENTRXD[7:0] byte is valid on the last byte. When the width of EMAC#CLIENTRXD is set to 8 bits wide, this signal should be left unconnected.
EMAC#CLIENTRXGOODFRAME	Output	This signal is asserted after the last receipt of data to indicate the reception of a compliant frame.
EMAC#CLIENTRXBADFRAME	Output	This signal is asserted after the last receipt of data to indicate the reception of a non-compliant frame.
EMAC#CLIENTRXSTATS[6:0]	Output	The statistics data on the last received data frame. The 27-bit raw RX statistics vector is multiplexed into a seven-bits per RX clock cycle output for statistics gathering. See "Receiver Statistics Vector" in Chapter 3.
EMAC#CLIENTRXSTATSBYTEVLD	Output	Asserted if an Ethernet MAC frame byte (including destination address to FCS) is received. Valid on every RX clock cycle.
EMAC#CLIENTRXSTATSVLD	Output	Asserted by the Ethernet MAC after the end of receiving a frame to indicate a valid EMAC#CLIENTRXSTATS[6:0] output.
EMAC#CLIENTRXCLIENTCLKOUT	Output	See "Receive Clocking Scheme" in Chapter 5.
EMAC#CLIENTRXDVREG6	Output	Reserved - not used.

Flow Control Client-Side Interface Signals

Table 2-3 describes the signals used by the client to request a flow control action from the transmit engine. The flow control block is designed per the specifications in Clause 31 of the IEEE Std 802.3-2002 standard. Flow control frames received by the Ethernet MAC are automatically handled.

Table 2-3: **Flow Control Interface Signals**

Signal	Direction	Description
CLIENTEMAC#PAUSEREQ	Input	Asserted by client to transmit a pause frame.
CLIENTEMAC#PAUSEVAL[15:0]	Input	The amount of pause time for the transmitter as defined in the IEEE Std 802.3-2002 specification.

Clock Signals

Table 2-4 shows the clock signals necessary to drive the Ethernet MAC.

Table 2-4: **Clock Signals**

Signal	Direction	Description
PHYEMAC#GTXCLK	Input	Clock supplied by the user to derive the other transmit clocks. Clock tolerance must be within the IEEE Std 802.3-2002 specification.
EMAC#CLIENTRXCLIENTCLKOUT	Output	Clock for receive client generated by the clock generator of the Ethernet MAC.
EMAC#CLIENTTXCLIENTCLKOUT	Output	Clock for transmit client generated by the clock generator of the Ethernet MAC.
CLIENTEMAC#RXCLIENTCLKIN	Input	Clock from receive client for the running of the receiver engine of the Ethernet MAC.(1)
CLIENTEMAC#TXCLIENTCLKIN	Input	Clock from transmit client for the running of the transmitter engine of the Ethernet MAC.(1)
EMAC#CLIENTTXGMIIMIICLKOUT	Output	Clock for MII, GMII, and RGMII modules. Generated by the clock generator of the Ethernet MAC.
CLIENTEMAC#TXGMIIMIICLKIN	Input	Clock from MII, GMII, and RGMII modules for the running of the MII/GMII/RGMII transmitter layer of the Ethernet MAC.(1)

Notes:

1. The Ethernet MAC uses this clock to generate an internal clock that eliminates clock skew between the Ethernet MAC and the client logic in the FPGA.

Host Interface Signals

Host Bus Signals

Table 2-5 outlines the host bus interface signals.

Table 2-5: **Host Bus Signals**

Signal	Direction	Description
HOSTCLK	Input	Clock supplied for running the host. User must supply this clock at all times even if the host interface is not used.
HOSTOPCODE[1:0]	Input	Defines operation to be performed over MDIO interface. Bit [1] is also used in configuration register access. See "Configuration Registers" in Chapter 3.
HOSTADDR[9:0]	Input	Address of register to be accessed.
HOSTWRDATA[31:0]	Input	Data bus to write to register.
HOSTRDDATA[31:0]	Output	Data bus to read from register.
HOSTMIIMSEL	Input	When asserted, the MDIO interface is accessed. When deasserted, the Ethernet MAC internal configuration registers are accessed.
HOSTREQ	Input	Used to signal a transaction on the MDIO interface.
HOSTEMAC1SEL	Input	This signal is asserted when EMAC1 is being accessed through the host interface and deasserted when EMAC0 is being accessed through the host interface. It is ignored when the host interface is not used.
HOSTMIIMRDY	Output	When High, the MDIO interface has completed any pending transaction and is ready for a new transaction.

Notes:

1. All signals are synchronous to HOSTCLK and are active High.
2. When using the PowerPC 405 processor as a host processor and using the DCR bus for host access, the host bus signals are used to read the optional FPGA fabric-based statistics registers. See "Interfacing to an FPGA Fabric-Based Statistics Block" in Chapter 6.

DCR Bus Interface Signals

Table 2-6 outlines the DCR bus interface signals.

Table 2-6: **DCR Bus Signals**

Signal	Direction	Description
DCREMACCLK	Input	Clock for the DCR interface from the PowerPC processor.
DCREMACABUS[8:9]	Input	Two LSBs of the DCR address bus. Bits[0] through [7] are decoded in conjunction with the PowerPC block.
DCREMACREAD	Input	DCR read request.
DCREMACWRITE	Input	DCR write request.
DCREMACDBUS[0:31]	Input	DCR write data bus.
DCREMACENABLE(1)	Input	When using the DCR bus, this signal is connected directly to the PPC405 output port DCREMACENABLER for DCR bus access. When using the host bus interface, the signal is connected to the logic ground.
EMACDCRDBUS[0:31]	Output	DCR read data bus.
EMACDCRACK	Output	DCR acknowledge.
DCRHOSTDONEIR(1)	Output	Interrupt signal to the PowerPC processor when the Ethernet MAC register access is done.

Notes:

1. All the DCR bus signals are internally connected to the PowerPC processor except for the DCREMACENABLE and DCRHOSTDONEIR signals.

Reset and CLIENTEMAC#DCMLOCKED Signals

Table 2-7 describes the Reset signal.

Table 2-7: **Reset Signal**

Signal	Direction	Description
Reset	Input	Asynchronous reset of both Ethernet MACs.

Table 2-8 describes the CLIENTEMAC#DCMLOCKED signal.

Table 2-8: **CLIENTEMAC#DCMLOCKED Signal**

Signal	Direction	Description
CLIENTEMAC#DCMLOCKED	Input	If a DCM is used to derive any of the clock signals, the LOCKED port of the DCM must be connected to the CLIENTEMAC#DCMLOCKED port. The Ethernet MAC is held in reset until CLIENTEMAC#DCMLOCKED is asserted High. If a DCM is not used, both CLIENTEMAC#DCMLOCKED ports from EMAC0 and EMAC1 must be tied High. If any Ethernet MAC is not used, CLIENTEMAC#DCMLOCKED must be tied to High.

Tie-Off Pins

Configuration Vectors

This section describes the 80 tie-off pins (TIEEMAC#CONFIGVEC[79:0]) used to configure the Virtex-4 FPGA Embedded Tri-Mode Ethernet MAC. The values of these tie-off pins are loaded into the Ethernet MAC at power-up or when the Ethernet MAC is reset.

When TIEEMAC#CONFIGVEC[67] is High, the host interface is selected. Tie-off pins pre-configure the internal control registers of the Ethernet MAC. The host interface is then used to dynamically change the register contents or to read the registers. When the host interface is not selected, the tie-off pins directly control the behavior of the Ethernet MAC. However, dynamically changing the register contents using the tie-off pins is not recommended.

The configuration vectors are divided into three sections: physical interface configuration vectors (Table 2-9), mode configuration vectors (Table 2-10), and MAC configuration vectors (Table 2-11). The MAC and physical interface configuration vectors can be configured through the host interface and are intended to be used dynamically to change register contents or read status registers. The mode configuration vectors preconfigure the internal control registers (16-bit, PCS/PMA, Host, SGMII, RGMII, and MDIO interfaces) but are not dynamically reconfigurable.

Table 2-9: **Physical Interface Configuration Pins**

Signal	Direction	Description
TIEEMAC#CONFIGVEC[79]	Input	Reserved, set to 1.
TIEEMAC#CONFIGVEC[78:74] — Only used in SGMII or 1000BASE-X modes. When MDIO and host are omitted from the Ethernet MAC, this alternative can be used.		
TIEEMAC#CONFIGVEC[78]	Input	PHY_RESET: Asserting this pin resets the PCS/PMA module.
TIEEMAC#CONFIGVEC[77]	Input	PHY_INIT_AN_ENABLE: Asserting this pin enables auto-negotiation of the PCS/PMA module.
TIEEMAC#CONFIGVEC[76]	Input	PHY_ISOLATE: Asserting this pin causes the PCS/PMA sublayer logic to behave as if it is electrically isolated from the attached Ethernet MAC, as defined in IEEE Std 802.3, Clause 22.2.4.1.6. Therefore, frames transmitted by the Ethernet MAC are not forwarded through the PCS/PMA. Frames received by the PCS/PMA are not relayed to the Ethernet MAC.
TIEEMAC#CONFIGVEC[75]	Input	PHY_POWERDOWN: Asserting this pin causes the MGT to be placed in a Low power state. A reset must be applied to clear the Low power state.
TIEEMAC#CONFIGVEC[74]	Input	PHY_LOOPBACK_MSB: Asserting this pin sets serial loopback in the MGT.

Table 2-10: **Mode Configuration Pins**

Signal	Direction	Description
TIEEMAC#CONFIGVEC[73]	Input	MDIO enable. Asserting this pin enables the use of MDIO in the Ethernet MAC. See "MDIO Interface" in Chapter 3.
TIEEMAC#CONFIGVEC[72:71] — These pins determine the speed of the Ethernet MAC after reset or power-up. These bits can be changed in the Ethernet MAC mode configuration register (Table 3-12, page 77) through the host interface when the host interface is selected (by setting TIEEMAC#CONFIG[67] High). When TIEEMAC#CONFIG[67] is Low, the speed of the Ethernet MAC is directly set by these two bits. 10 = 1000 Mb/s 01 = 100 Mb/s 00 = 10 Mb/s 11 = not applicable		
TIEEMAC#CONFIGVEC[72]	Input	SPEED[1]
TIEEMAC#CONFIGVEC[71]	Input	SPEED[0]
TIEEMAC#CONFIGVEC[70:68] — Defines the physical interface of the Ethernet MAC. These pins are mutually exclusive. 10/100 MII and GMII modes are enabled when TIEEMAC#CONFIGVEC[70:68] are deasserted; the RGMII, SGMII, and 1000BASE-X modes are not set.		
TIEEMAC#CONFIGVEC[70]	Input	RGMII mode enable. Asserting this pin sets the Ethernet MAC in RGMII mode.
TIEEMAC#CONFIGVEC[69]	Input	SGMII mode enable. Asserting this pin sets the Ethernet MAC in SGMII mode.
TIEEMAC#CONFIGVEC[68]	Input	1000BASE-X PCS/PMA mode enable. Asserting this pin sets the Ethernet MAC in 1000BASE-X mode.
TIEEMAC#CONFIGVEC[67]	Input	Host Interface enable. Asserting this pin enables the use of the Ethernet MAC host interface.
TIEEMAC#CONFIGVEC[66]	Input	Transmit 16-bit client interface enable. When asserted, the TX client data interface is 16 bits wide. When deasserted, the TX client data interface is 8 bits wide.
TIEEMAC#CONFIGVEC[65]	Input	Receive 16-bit client interface enable. When asserted, the RX client data interface is 16 bits wide. When deasserted, the RX client data interface is 8 bits wide.

Table 2-11: **MAC Configuration Pins**

Signal	Direction	Description
TIEEMAC#CONFIGVEC[64]	Input	Address Filter Enable: Asserting this pin enables the use of the address filter module in the Ethernet MAC.
TIEEMAC#CONFIGVEC[63]	Input	Length/Type Check Disable: When this pin is asserted, it disables the comparison of the L/T field with the size of the data.
TIEEMAC#CONFIGVEC[62:61] — These pins configure the Ethernet MAC flow control module.		
TIEEMAC#CONFIGVEC[62]	Input	Receive Flow Control Enable. When this bit is 1 and full-duplex mode is enabled, the received flow control frames inhibit transmitter operation. When this bit is 0, the received flow frames are passed up to the client.
TIEEMAC#CONFIGVEC[61]	Input	Transmit Flow Control Enable. When this bit is 1 and full duplex mode is enabled, asserting the CLIENTEMAC#PAUSE_REQ signal causes the Ethernet MAC to send a flow control frame out from the transmitter. When 0, asserting the CLIENTEMAC#PAUSE_REQ signal has no effect.
TIEEMAC#CONFIGVEC[60:54] — Configures the transmit engine of the Ethernet MAC.		
TIEEMAC#CONFIGVEC[60]	Input	Transmitter Reset. When this bit is 1, the Ethernet MAC transmitter is held in reset. This signal is an input to the reset circuit for the transmitter block.
TIEEMAC#CONFIGVEC[59]	Input	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 Std 802.3-2002 to be sent. When this bit is 0, the Ethernet MAC transmitter only allows frames up to the legal maximum to be sent.
TIEEMAC#CONFIGVEC[58]	Input	Transmitter In-Band FCS Enable. When this bit is 1, the Ethernet MAC transmitter expects the FCS field to be passed in by the client. When this bit is 0, the Ethernet MAC transmitter appends padding as required, computes the FCS, and appends it to the frame.
TIEEMAC#CONFIGVEC[57]	Input	Transmitter Enable. When this bit is 1, the transmitter is operational. When this bit is 0, the transmitter is disabled.
TIEEMAC#CONFIGVEC[56]	Input	Transmitter VLAN Enable. When this bit is 1, the transmitter allows the transmission of VLAN tagged frames.
TIEEMAC#CONFIGVEC[55]	Input	Transmitter Half Duplex. When this bit is 1, the transmitter operates in half-duplex mode. When this bit is 0, the transmitter operates in full-duplex mode.
TIEEMAC#CONFIGVEC[54]	Input	Transmitter IFG Adjust enable. When this bit is 1, the transmitter reads the value of the CLIENTEMAC#TXIFGDELAY[7:0] port and sets the IFG accordingly. When this bit is 0, the transmitter always inserts at least the legal minimum IFG.

Table 2-11: **MAC Configuration Pins *(Cont'd)***

Signal	Direction	Description
TIEEMAC#CONFIGVEC[53:0] — Configures the receive engine of the Ethernet MAC.		
TIEEMAC#CONFIGVEC[53]	Input	Receiver Reset. When this bit is 1, the receiver is held in reset. This signal is an input to the reset circuit for the receiver block.
TIEEMAC#CONFIGVEC[52]	Input	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 Std 802.3-2002. When this bit is 1, the receiver does not have an upper limit on frame size.
TIEEMAC#CONFIGVEC[51]	Input	Receiver In-band FCS Enable. When this bit is 1, the Ethernet MAC receiver passes the FCS field up to the client. When this bit is 0, the Ethernet MAC receiver does not pass the FCS field. In both cases, the FCS field are verified on the frame.
TIEEMAC#CONFIGVEC[50]	Input	Receiver Enable. When this bit is 1, the receiver block is operational. When this bit is 0, the block ignores activity on the physical interface RX port.
TIEEMAC#CONFIGVEC[49]	Input	Receiver VLAN Enable. When this bit is 1, VLAN tagged frames are accepted by the receiver.
TIEEMAC#CONFIGVEC[48]	Input	Receiver Half Duplex. When this bit is 1, the receiver operates in half-duplex mode. When this bit is 0, the receiver operates in full-duplex mode.
TIEEMAC#CONFIGVEC[47:0]	Input	Pause frame Ethernet MAC Source Address[47:0]. This address is used by the Ethernet MAC to match against the destination address of any incoming flow control frames, and as the source address for any outbound flow control frames. The address is ordered for the least significant byte in the register to have the first byte transmitted or received; for example, an Ethernet MAC address of AA-BB-CC-DD-EE-FF is stored in byte [47:0] as 0xFFEEDDCCBBAA. Tied to the same Ethernet MAC address as TIEEMAC#UNICASTADDR[47:0].

Notes:

1. A reset is needed before changes on TIEEMAC#CONFIGVEC[73] and [70:64] take effect.

Unicast Address

Table 2-12 describes the 48 tie-off pins (TIEEMAC#UNICASTADDR[47:0]) used to set the Ethernet MAC address for the Virtex-4 FPGA Embedded Tri-Mode Ethernet MAC.

Table 2-12: **Unicast Address Pins**

Signal	Direction	Description
TIEEMAC#UNICASTADDR[47:0]	Input	This 48-bit wide tie-off is used to set the Ethernet MAC unicast address used by the address filter block to see if the incoming frame is destined for the Ethernet MAC. The address is ordered for the least significant byte in the register to have the first byte transmitted or received; for example, an Ethernet MAC address of 06-05-04-03-02-01 is stored in byte [47:0] as `0x010203040506`.

Notes:

1. All of the TIEEMAC#UNICASTADDR[47:0] bits are registered on input and can be treated as asynchronous inputs.

Management Data Input/Output (MDIO) Interface Signals

Table 2-13 describes the Management Data Input/Output (MDIO) interface signals. The MDIO format is defined in IEEE Std 802.3, Clause 22.

Table 2-13: **MDIO Interface Signals**

Signal	Direction	Description
EMAC#PHYMCLKOUT	Output	Management clock derived from the host clock or PHYEMAC#MCLKIN.
PHYEMAC#MCLKIN	Input	When the host is not used, access to the PCS must be provided by an external MDIO controller. In this situation, the management clock is an input to the core.
PHYEMAC#MDIN	Input	Signal from the physical interface for communicating the configuration and status. If unused, must be tied High.
EMAC#PHYMDOUT	Output	Signal to output the configuration and command to the physical interface.
EMAC#PHYMDTRI	Output	The 3-state control to accompany EMAC#PHYMDOUT.

Mode-Dependent Signals

The Ethernet MAC has several signals that change definition depending on the selected operating mode. This section describes the basic signals in the various operating modes.

Data and Control Signals

Table 2-14 shows the data and control signals for the different modes. They are set from the tie-off pins. These signals are multiplexed, and their functionality is defined when the mode is set.

Table 2-14: **PHY Data and Control Signals**

Signal	Direction	Mode	Description
PHYEMAC#MIITXCLK	Input	10/100 MII	The TX clock generated from the PHY when operating in 10/100 MII mode.
		16-bit client interface used in 1000BASE-X PCS/PMA	When the transmit client interface is configured to be 16 bits wide, this is the clock input port for the CLIENTEMAC#TXCLIENTCLKIN/2. See "Transmit (TX) Client – 16-bit Wide Interface" in Chapter 3.
EMAC#PHYTXCLK	Output	GMII	The TX clock out to the PHY in GMII 1000 Mb/s mode only.
EMAC#CLIENTTXGMIIMIICLKOUT	Output	GMII	The TX clock out to the PHY for GMII tri-speed mode operation and RGMII.
		RGMII	
EMAC#PHYTXEN	Output	10/100 MII	The data enable control signal to the PHY.
		GMII	
		RGMII	The RGMII_TX_CTL_RISING signal to the PHY.
EMAC#PHYTXER	Output	10/100 MII	The error control signal to the PHY.
		GMII	
		RGMII	The RGMII_TX_CTL_FALLING signal to the PHY.
EMAC#PHYTXD[7:0]	Output	10/100 MII	EMAC#PHYTXD[3:0] is the transmit data signal to the PHY. EMAC#PHYTXD[7:4] are driven Low.
		GMII	The transmit data signal to the PHY.
		RGMII	EMAC#PHYTXD[3:0] is the RGMII_TXD_RISING and EMAC#PHYTXD[7:4] is the RGMII_TXD_FALLING signal to the PHY.
		SGMII	The TX_DATA signal to the MGT.
		1000BASE-X	
PHYEMAC#RXCLK	Input	10/100 MII	The recovered clock from received data stream by the PHY.
		GMII	
		RGMII	
		16-bit client interface used in 1000BASE-X PCS/PMA	When the receive client interface is configured to be 16 bits wide, this signal is the clock input port for the CLIENTEMAC#RXCLIENTCLKIN/2. See "Receive (RX) Client – 16-bit Wide Interface" in Chapter 3.

Table 2-14: **PHY Data and Control Signals *(Cont'd)***

Signal	Direction	Mode	Description
PHYEMAC#RXDV	Input	10/100 MII	The data valid control signal from the PHY.
		GMII	
		RGMII	The RGMII_RX_CTL_RISING signal.
		SGMII	The RXREALIGN signal from the MGT.
		1000BASE-X	
PHYEMAC#RXER	Input	10/100 MII	The error control signal from the PHY.
		GMII	
		RGMII	The RGMII_RX_CTL_FALLING signal from the PHY.
PHYEMAC#RXD[7:0]	Input	10/100 MII	PHYEMAC#RXD [3:0] is the received data signal from the PHY. PHYEMAC#RXD [7:4] is left unconnected.
		GMII	The received data signal to the PHY.
		RGMII	PHYEMAC#RXD [3:0] is the RGMII_RXD_RISING and PHYEMAC#RXD [7:4] is the RGMII_RXD_FALLING signal from the PHY.
		SGMII	The RX_DATA from the MGT.
		1000BASE-X	
PHYEMAC#COL	Input	10/100 MII	The collision control signal from the PHY, used in half-duplex mode.
		SGMII	The TXRUNDISP signal from the MGT.
		1000BASE-X	
PHYEMAC#CRS	Input	10/100 MII	The carrier sense control signal from the PHY, used in half-duplex mode.

RocketIO Multi-Gigabit Transceiver Signals

Table 2-15 shows the signals used to connect the Ethernet MAC to the RocketIO Multi-Gigabit Transceiver (see UG076, *Virtex-4 RocketIO Multi-Gigabit Transceiver User Guide*).

Table 2-15: **Multi-Gigabit Transceiver Connections**

Signal	Direction	Description
EMAC#PHYENCOMMAALIGN	Output	Enable RocketIO PMA layer to realign to commas.
EMAC#PHYLOOPBACKMSB	Output	Loopback tests within the RocketIO Multi-Gigabit Transceivers.
EMAC#PHYMGTRXRESET	Output	Reset to RocketIO RXRESET.
EMAC#PHYMGTTXRESET	Output	Reset to RocketIO TXRESET.
EMAC#PHYPOWERDOWN	Output	Power-down the RocketIO Multi-Gigabit Transceivers.

Table 2-15: **Multi-Gigabit Transceiver Connections *(Cont'd)***

Signal	Direction	Description
EMAC#PHYSYNCACQSTATUS	Output	The output from the receiver's synchronization state machine of IEEE Std 802.3, Clause 36. When asserted High, the received bitstream is synchronized. The state machine is in one of the SYNC_ACQUIRED states of IEEE Std 802.3, figures 36-39. When deasserted Low, no synchronization has been obtained.
EMAC#PHYTXCHARDISPMODE	Output	Set running disparity for current byte.
EMAC#PHYTXCHARDISPVAL	Output	Set running disparity value.
EMAC#PHYTXCHARISK	Output	K character transmitted in TXDATA.
PHYEMAC#RXBUFSTATUS[1:0]	Input	Receiver Elastic Buffer Status: Bit[1] asserted indicates overflow or underflow.
PHYEMAC#RXCHARISCOMMA	Input	Comma detected in RXDATA.
PHYEMAC#RXCHARISK	Input	K character received or extra data bit in RXDATA. When RXNOTINTABLE is asserted, this signal becomes the tenth bit in RXDATA.
PHYEMAC#RXCHECKINGCRC	Input	Reserved - tied to GND.
PHYEMAC#RXBUFERR	Input	Reserved - tied to GND.
PHYEMAC#RXCOMMADET	Input	Reserved - tied to GND.
PHYEMAC#RXDISPERR	Input	Disparity error in RXDATA.
PHYEMAC#RXLOSSOFSYNC[1:0]	Input	Reserved - tied to GND.
PHYEMAC#RXNOTINTABLE	Input	Indicates non-existent 8B/10 code.
PHYEMAC#RXRUNDISP	Input	Running disparity in the received serial data. When RXNOTINTABLE is asserted in RXDATA, this signal becomes the ninth data bit.
PHYEMAC#RXCLKCORCNT[2:0]	Input	Status showing the occurrence of a clock correction.
PHYEMAC#TXBUFERR	Input	TX buffer error (overflow or underflow).

Table 2-16 shows the PCS/PMA signals.

Table 2-16: **PCS/PMA Signals**

Signal	Direction	Description
PHYEMAC#PHYAD[4:0]	Input	Physical interface address of MDIO register set for the PCS sublayer.
PHYEMAC#SIGNALDET	Input	Signal direct from PMD sublayer indicating the presence of light detected at the optical receiver, as defined in IEEE Std 802.3, Clause 36. If asserted High, the optical receiver has detected light. If deasserted Low, indicates the absence of light. If unused, this signal should be tied High for correct operation.
EMAC#CLIENTANINTERRUPT	Output	Interrupt upon auto-negotiation.

XILINX®

Chapter 3

Client, Host, and MDIO Interfaces

This chapter provides useful design information for user interaction with the Virtex®-4 FPGA Tri-Mode Ethernet MAC. It contains the following sections:

Client Interface

The client interface is designed for maximum flexibility for matching the client switching fabric or network processor interface.

Both the transmit and receive data pathway can be configured to be either 8 bits wide or 16 bits wide, with each pathway synchronous to the CLIENTEMAC#TXCLIENTCLKIN (transmit) or CLIENTEMAC#RXCLIENTCLKIN (receive) for completely independent full-duplex operation.

Figure 3-1 shows a block diagram of the transmit client interface. In 16-bit client mode, PHYEMAC#MIITXCLK functions as CLIENTEMAC#TXCLIENTCLKIN/2. TIEEMAC#CONFIGVEC[66] selects between an 8-bit or 16-bit client interface.

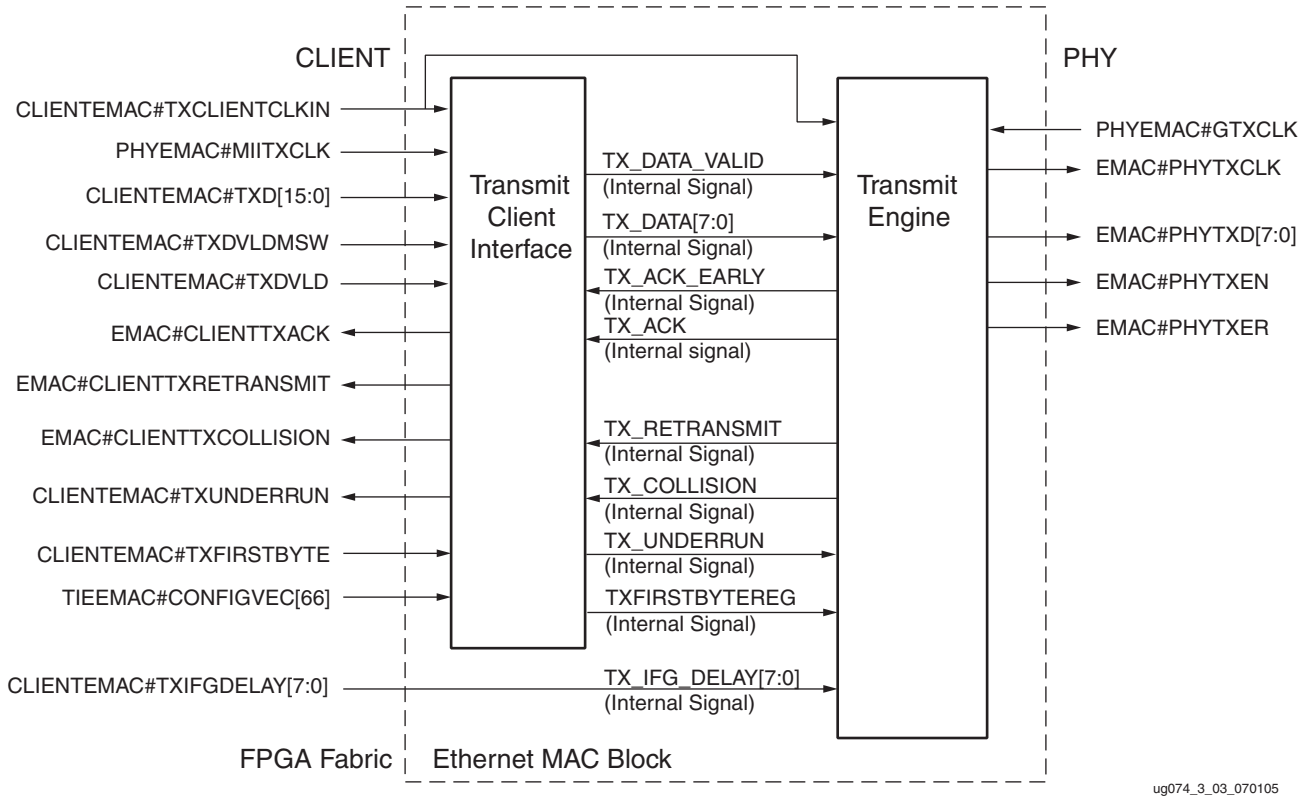


Figure 3-1: **Transmit Client Block Diagram**

Figure 3-2 shows a block diagram of the receive client interface. In 16-bit client mode, PHYEMAC#RXCLK functions as CLIENTEMAC#RXCLIENTCLKIN/2. TIEEMAC#CONFIGVEC[65] selects between an 8-bit or 16-bit client interface.

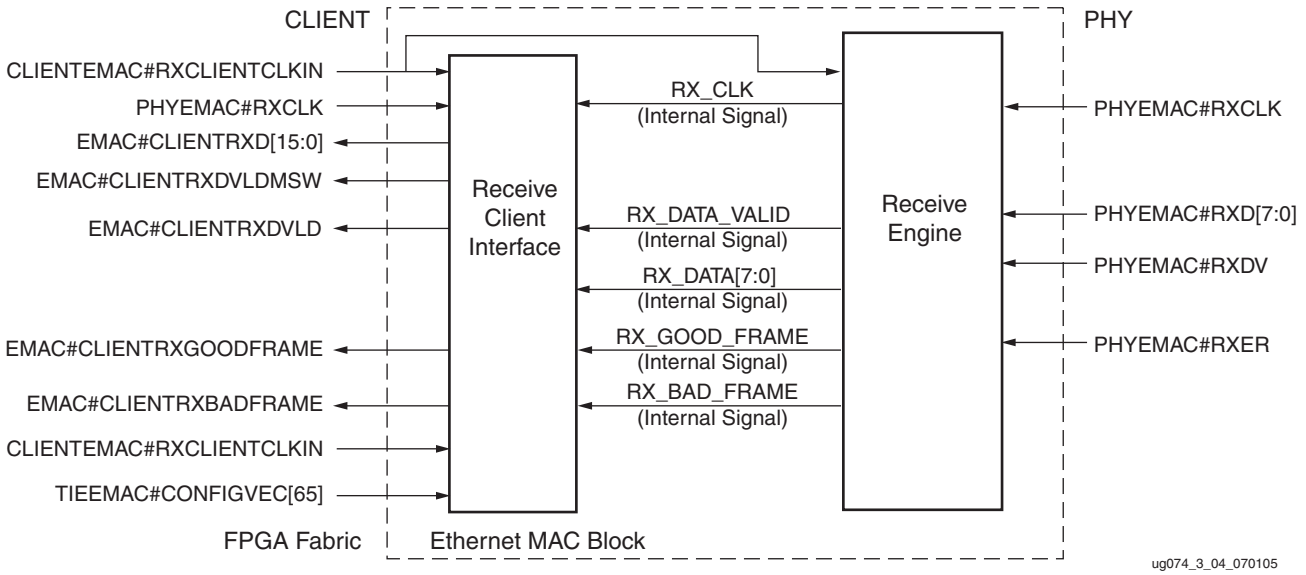


Figure 3-2: **Receive Client Interface Block Diagram**

Table 3-1 defines the abbreviations used throughout this chapter.

Table 3-1: **Abbreviations Used in this Chapter**

Abbreviation	Definition	Length
DA	Destination address	6 bytes
SA	Source address	6 bytes
L/T	Length/Type field	2 bytes
FCS	Frame check sequences	4 bytes
SGMII and 1000BASE-X PCS/PMA Only:		
PRE	Preamble	7 bytes
SFD	Start of frame delimiter	1 byte
/I1/	IDLE_1 (K28.5/D5.6)	2 bytes
/I2/	IDLE_2 (K28.5/D16.2)	2 bytes
/R/	Carrier Extend (K23.7)	1 byte
/S/	Start of Packet (K27.7)	1 byte
/T/	End of Packet (K29.7)	1 byte
/V/	Error Propagation (K30.7)	1 byte

Transmit (TX) Client – 8-bit Wide Interface

In this configuration, CLIENTEMAC#TXD[15:8] and CLIENTEMAC#TXDVLDMSW must be tied to ground.

Normal Frame Transmission

The timing of a normal outbound frame transfer is shown in Figure 3-3. When the client transmits a frame, the first column of data is placed on the CLIENTEMAC#TXD[7:0] port, and CLIENTEMAC#TXDVLD is asserted High. After the Ethernet MAC reads the first byte of data, it asserts the EMAC#CLIENTTXACK signal. On subsequent rising clock edges, the client must provide the rest of the frame data. CLIENTEMAC#TXDVLD is deasserted Low to signal an end-of-frame to the Ethernet MAC. In SGMII or 1000BASE-X PCS/PMA mode, the PCS engine inserts code characters in the data stream from CLIENTEMAC#TXD. Table 3-1 describes these code characters, and IEEE Std 802.3, Clause 36 has further definitions. The encapsulated data stream then appears on EMAC#PHYTXD and goes to the MGT. Along with EMAC#PHYTXCHARISK and EMAC#PHYTXCHARDISPMODE, the MGT encodes the incoming data to the appropriate 8B/10B stream.

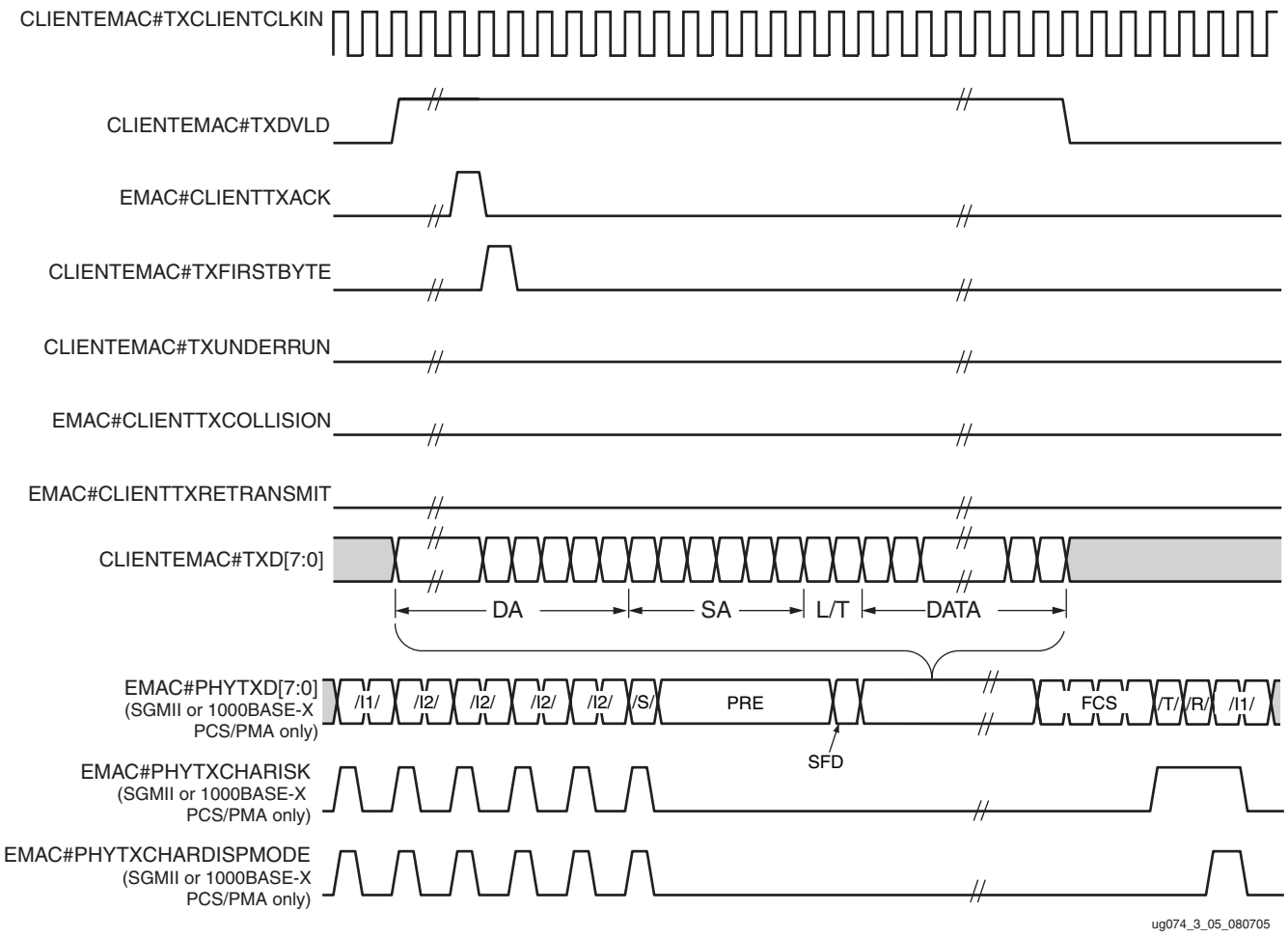


Figure 3-3: **Normal Frame Transmission Across Client Interface**

In-Band Parameter Encoding

The Ethernet MAC frame parameters, destination address, source address, length/type, and the FCS are encoded within the same data stream used to transfer the frame payload instead of separate ports. This provides the maximum flexibility in switching applications.

Padding

When fewer than 46 bytes of data are supplied by the client to the Ethernet MAC, the transmitter module adds padding – up to the minimum frame length. However, if the Ethernet MAC is configured for client-passed FCS, the client must also supply the padding to maintain the minimum frame length (see "Client-Supplied FCS Passing," page 41).

Client-Supplied FCS Passing

In the transmission timing case shown in Figure 3-4, an Ethernet MAC is configured to have the FCS field passed in by the client (see "Configuration Registers," page 74). The client must ensure that the frame meets the Ethernet minimum frame length requirements; the Ethernet MAC does not pad the payload.

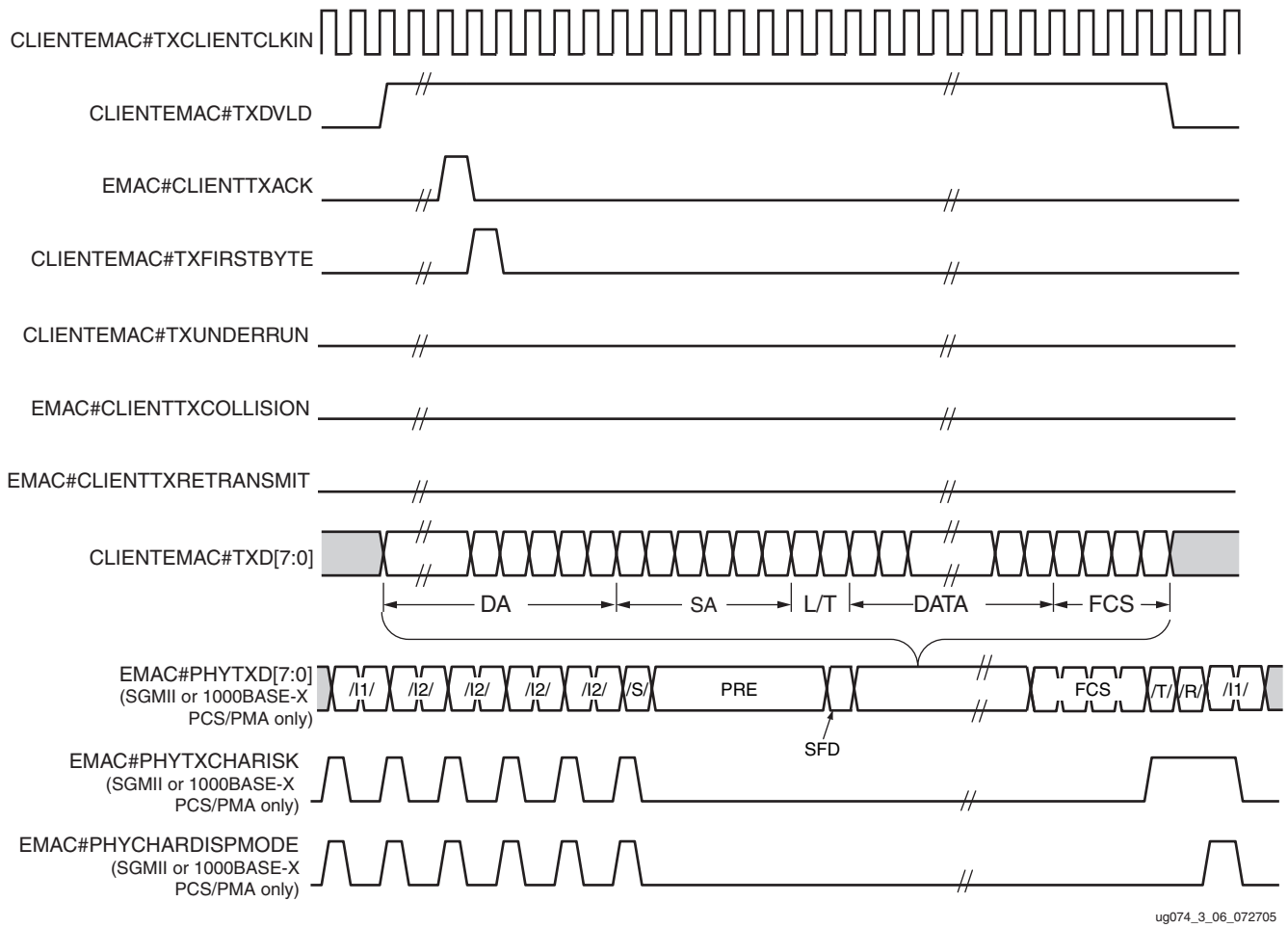


Figure 3-4: **Frame Transmission with Client-Supplied FCS**

Client Underrun

The timing of an aborted transfer is shown in Figure 3-5. An aborted transfer can occur if a FIFO connected to the client interface empties, before a frame is completed. When the client asserts CLIENTEMAC#TXUNDERRUN during a frame transmission, the EMAC#PHYTXER is asserted for one clock cycle to notify the external PHY that the frame is corrupted in MII and GMII modes. In 1000BASE-X PCS/PMA mode, the Ethernet MAC inserts an error code (/V/) into the current frame to signal corruption. It then falls back to idle transmission. EMAC#PHYTXER is asserted some cycles after. The client must requeue the aborted frame for transmission.

When an underrun occurs, to request a new transmission, reassert CLIENTEMAC#TXDVLD on the clock cycle after the CLIENTEMAC#TXUNDERRUN is asserted.

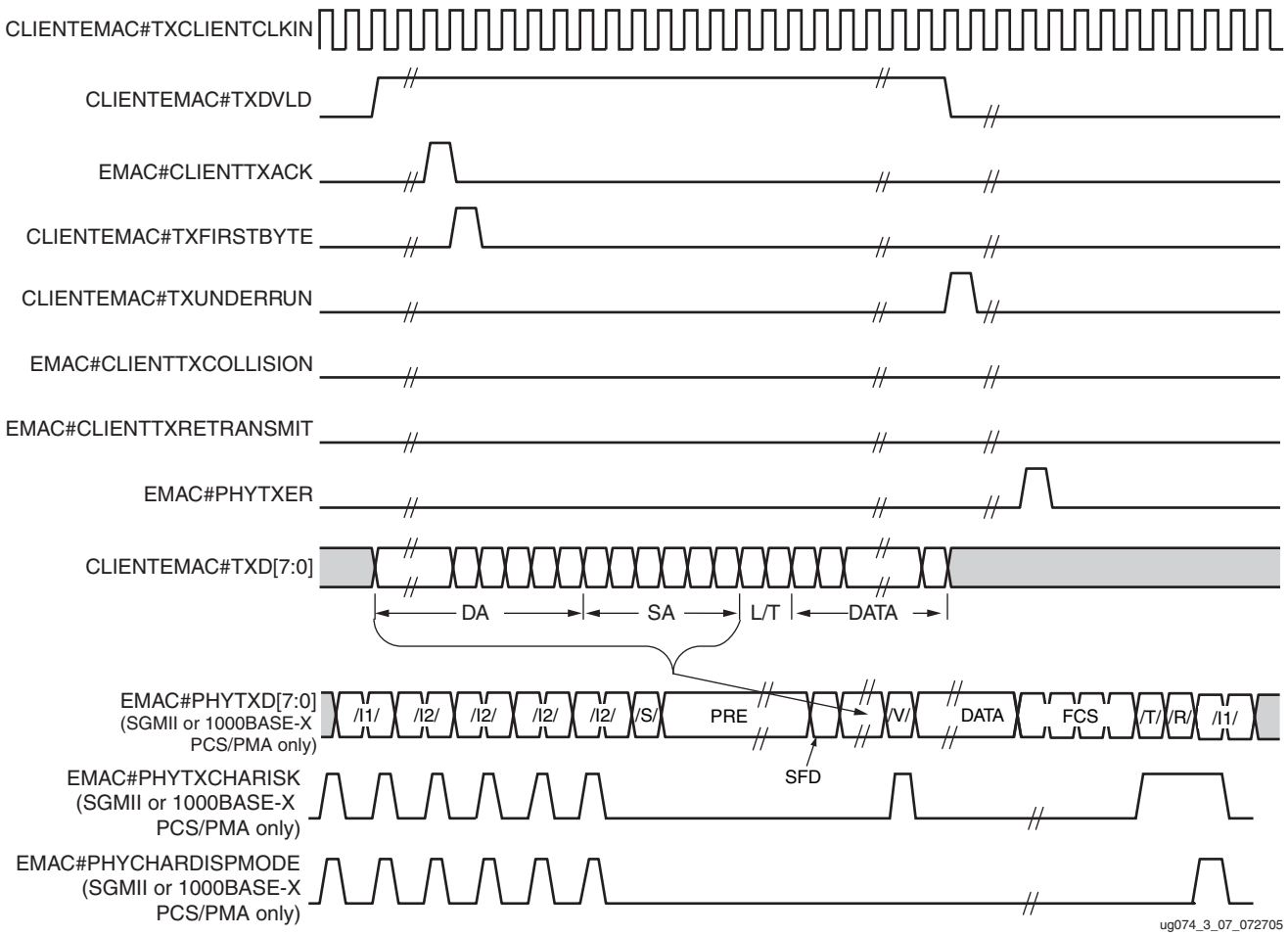


Figure 3-5: **Frame Transmission with Underrun**

Back-to-Back Transfers

Back-to-back transfers can occur when the Ethernet MAC client is immediately ready to transmit a second frame of data following completion of the first frame. In Figure 3-6, the end of the first frame is shown on the left. At the clock cycle immediately following the final byte of the first frame, CLIENTEMAC#TXDVLD is deasserted by the client. One clock cycle later, CLIENTEMAC#TXDVLD is asserted High. This indicates that the first byte of the destination address of the second frame is awaiting transmission on CLIENTEMAC#TXD.

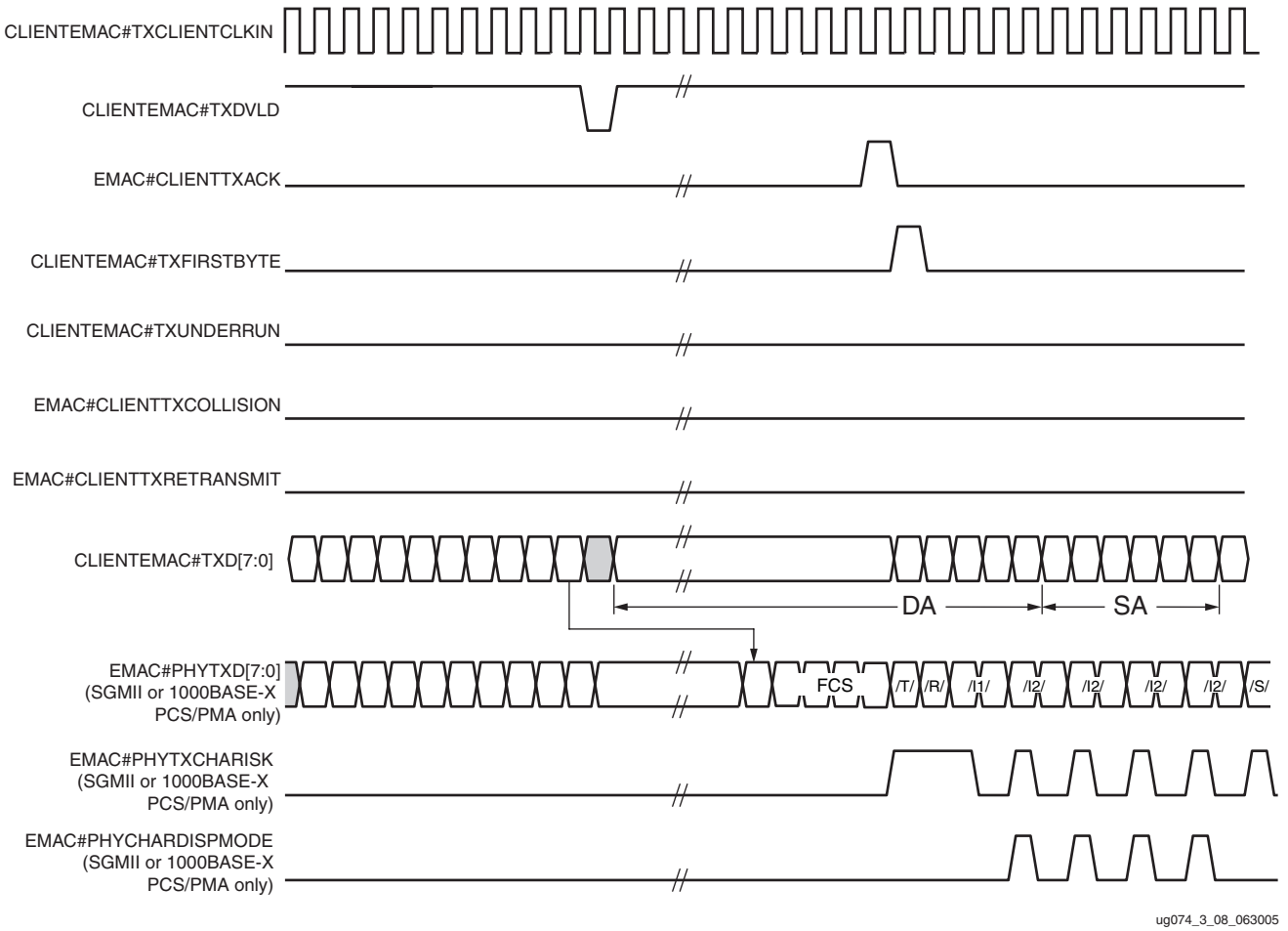


Figure 3-6: **Back-to-Back Frame Transmission**

When the Ethernet MAC is ready to accept data, EMAC#CLIENTTXACK is asserted and the transmission continues in the same manner as the single frame case. The Ethernet MAC defers the assertion of EMAC#CLIENTTXACK to comply with inter-packet gap requirements and flow control requests.

Virtual LAN (VLAN) Tagged Frames

Figure 3-7 shows the transmission of a VLAN tagged frame (if enabled). The handshaking signals across the interface do not change. However, the VLAN type tag `0x8100` must be supplied by the client to show the frame as VLAN tagged. The client also supplies the two bytes of tag control information, V1 and V2, at the appropriate times in the data stream. More information on the contents of these two bytes can be found in the IEEE Std 802.3-2002 specification.

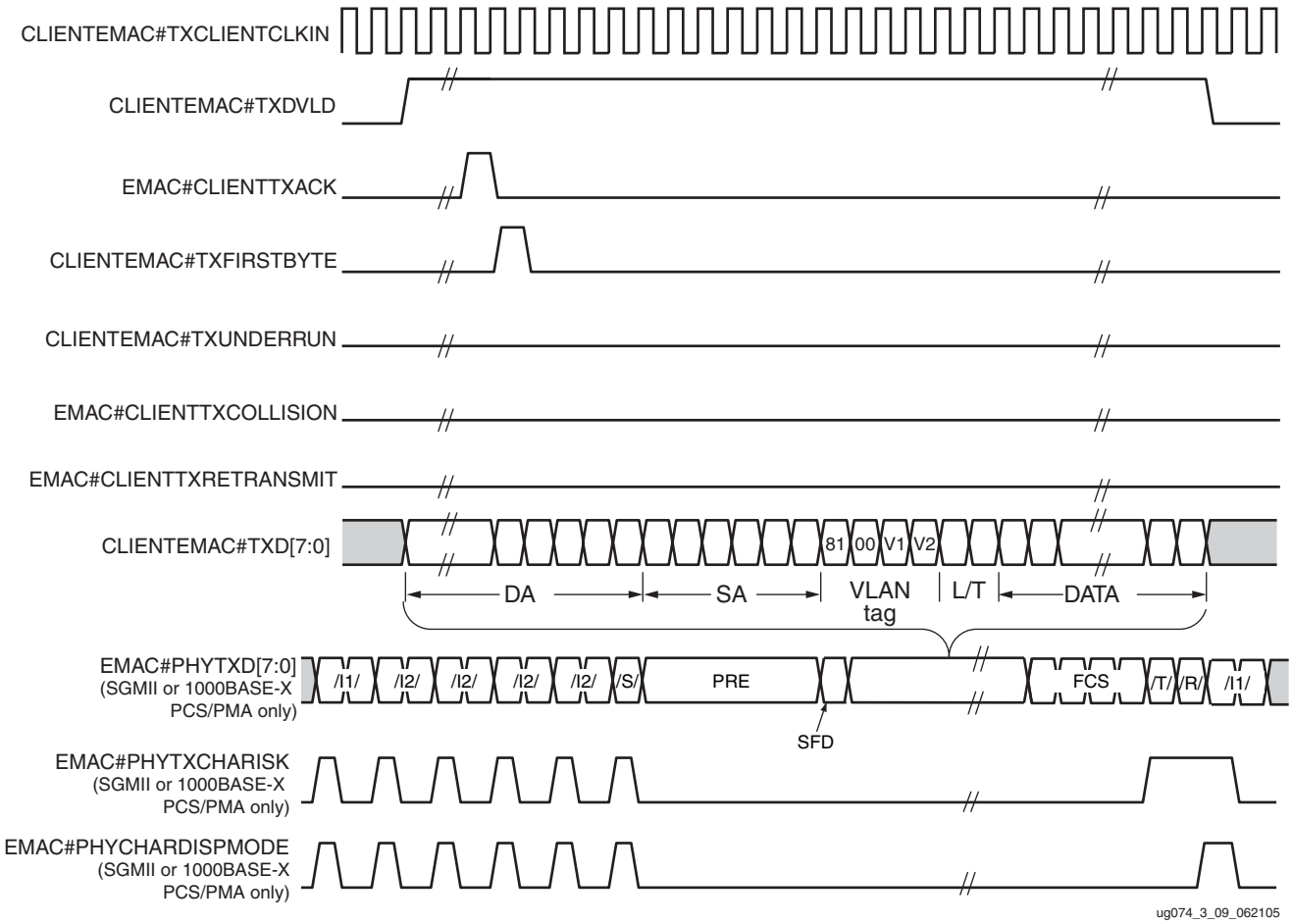


Figure 3-7: **Transmission of a VLAN Tagged Frame**

Maximum Permitted Frame Length and Jumbo Frames

The maximum length of a frame specified in the IEEE Std 802.3-2002 specification is 1518 bytes for non-VLAN tagged frames. VLAN tagged frames can be extended to 1522 bytes. When jumbo frame handling is disabled and the client attempts to transmit a frame that exceeds the maximum legal length, the Ethernet MAC 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 longer than the legal maximum are transmitted error free.

For more information on enabling and disabling jumbo frame handling, see "Configuration Registers," page 74.

Frame Collisions (Half-Duplex 10/100 Mb/s Operation Only)

In half-duplex Ethernet operation, collisions occur on the medium. This is how the arbitration algorithm is fulfilled. When there is a collision, the Ethernet MAC signals to the client a need to have data re-supplied as follows:

- If there is a collision, the EMAC#CLIENTTXCOLLISION signal is set to `1` by the Ethernet MAC. If a frame is in progress, the client must abort the transfer and CLIENTEMAC#TXDVLD is deasserted to `0`.
- If the EMAC#CLIENTTXRETRANSMIT signal is `1` in the same clock cycle as the EMAC#CLIENTTXCOLLISION signal is `1`, the client must resubmit the previous frame to the Ethernet MAC for retransmission; CLIENTEMAC#TXDVLD must be asserted to the Ethernet MAC within eight clock cycles of the EMAC#CLIENTTXCOLLISION signal to meet Ethernet timing requirements. This operation is shown in Figure 3-8.

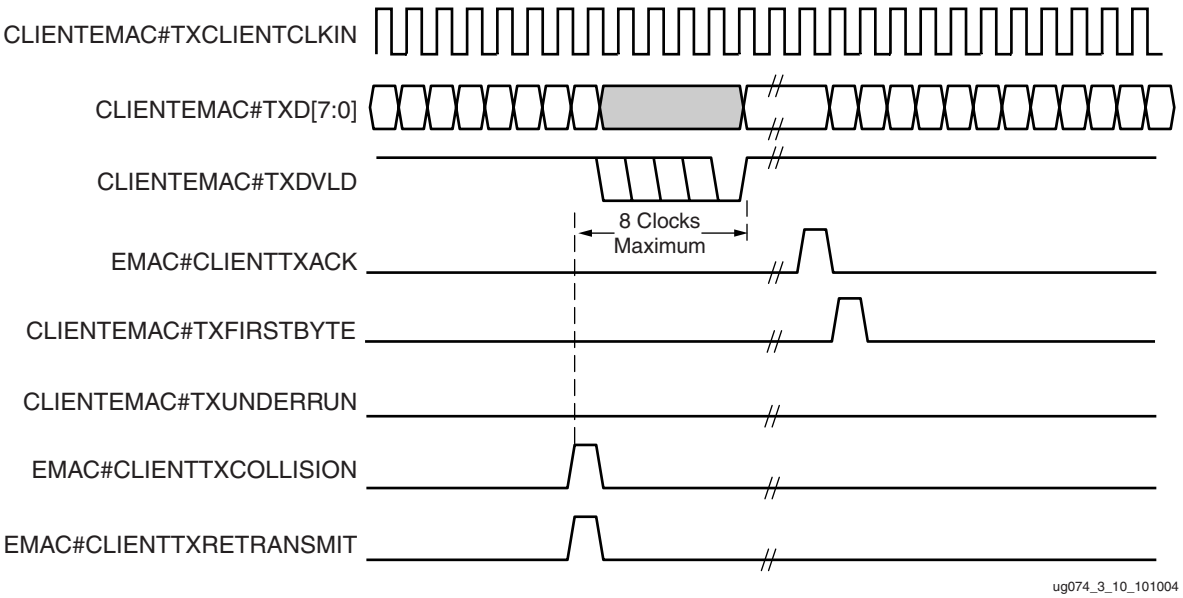


Figure 3-8: **Collision Handling - Frame Retransmission Required**

- If the EMAC#CLIENTTXRETRANSMIT signal is `0` in the same clock cycle when the EMAC#CLIENTTXCOLLISION signal is `1`, the number of retries for this frame has exceeded the Ethernet specification, and the frame should be dropped by the client. The client can then make any new frame available to the Ethernet MAC for transmission without timing restriction. This process is shown in Figure 3-9.

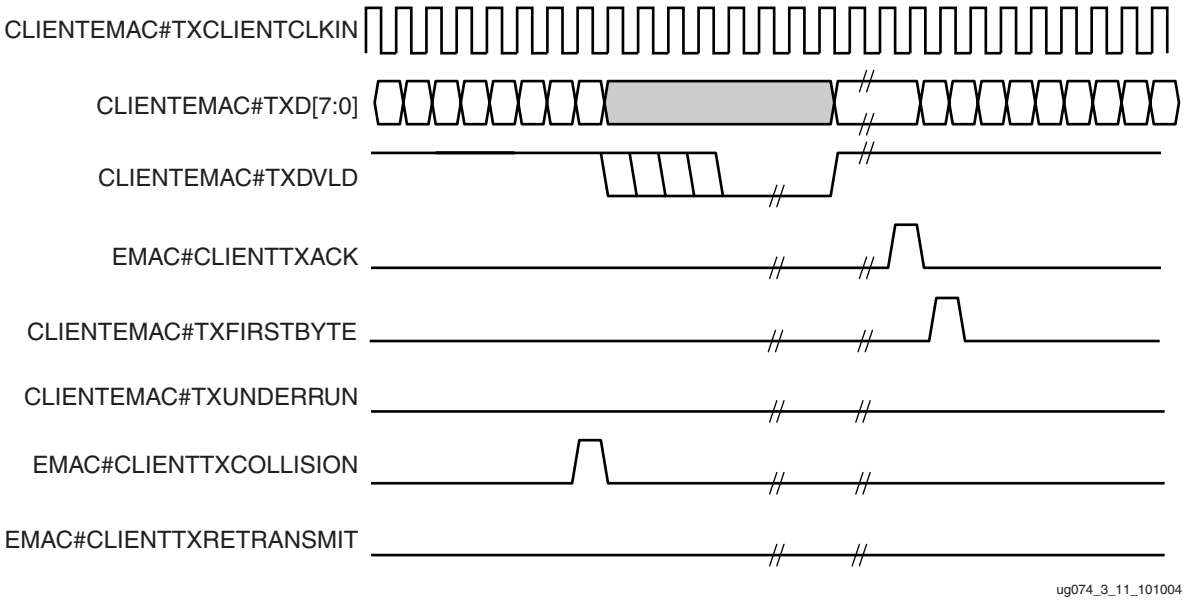


Figure 3-9: **Collision Handling - No Frame Retransmission Required**

IFG Adjustment

The length of the IFG can be varied in full-duplex mode. If this function is selected (using a configuration bit in the transmitter control register, see "Configuration Registers," page 74), then the Ethernet MAC exerts back pressure to delay the transmission of the next frame, until the requested number of idle cycles has elapsed. The number of idle cycles is controlled by the value on the CLIENTEMAC#TXIFGDELAY port at the start-of-frame transmission. Figure 3-10 shows the Ethernet MAC operating in this mode.

In full-duplex configurations, the minimum IFG is 12 bytes (96 bit times). In half-duplex configurations, the minimum supported IFG is 18 bytes (144 bit times) when using the MII physical interface, or 26 bytes (208 bit times) when using the RGMII physical interface.

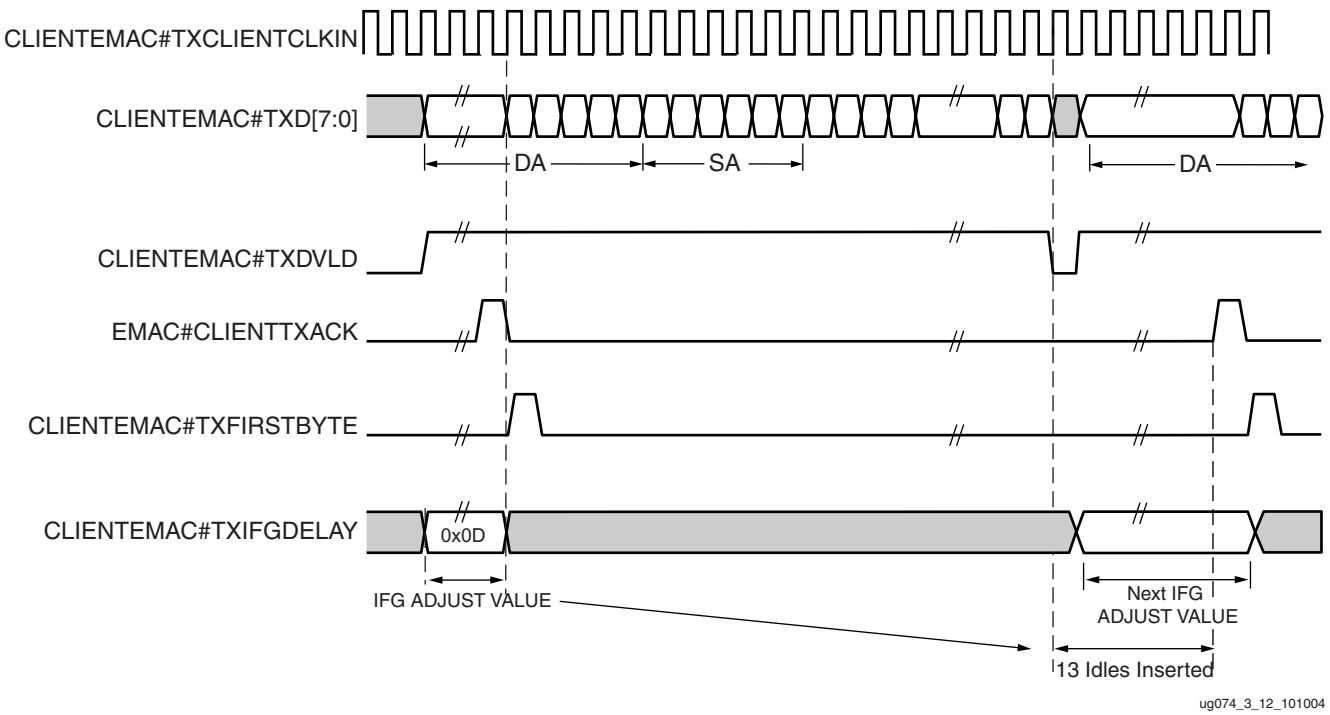


Figure 3-10: **IFG Adjustment**

Transmit (TX) Client – 16-bit Wide Interface

This optional configuration can only be used when the Ethernet MAC is configured in 1000BASE-X PCS/PMA mode. The frequency of the transmit client clock is half the frequency of the internal transmit. The 16-bit client interface allows the Ethernet MAC to run at an internal clock frequency of greater than 125 MHz. The Ethernet MAC can run at a line rate greater than 1 Gb/s as specified in IEEE Std 802.3. Therefore, this interface should *not* be used for Ethernet compliant designs, but it can be useful for backplane applications.

The PHYEMAC#MIITXCLK is used as the input clock port for the CLIENTEMAC#TXCLIENTCLKOUT, divided by two, as shown in Figure 3-1, page 38. Using a DCM with the transmit client clock (EMACCLIENT#TXCLIENTCLKOUT) as an input, the divide-by-two clock signal is generated. See Figure 4-28, page 139 for more information.

As in the 8-bit client interface, the PCS engine inserts code characters in the data stream from CLIENTEMAC#TXD. Table 3-1 describes these code characters, and IEEE Std 802.3, Clause 36 has further definitions. CLIENTEMAC#TXD[7:0] is transmitted to EMAC#PHYTXD first. Along with EMAC#PHYTXCHARISK and EMAC#PHYTXCHARDISPMODE, the MGT encodes the incoming data to the appropriate 8B/10B stream.

Figure 3-11 shows the timing of a normal outbound frame transfer for the case with an even number of bytes in the frame.

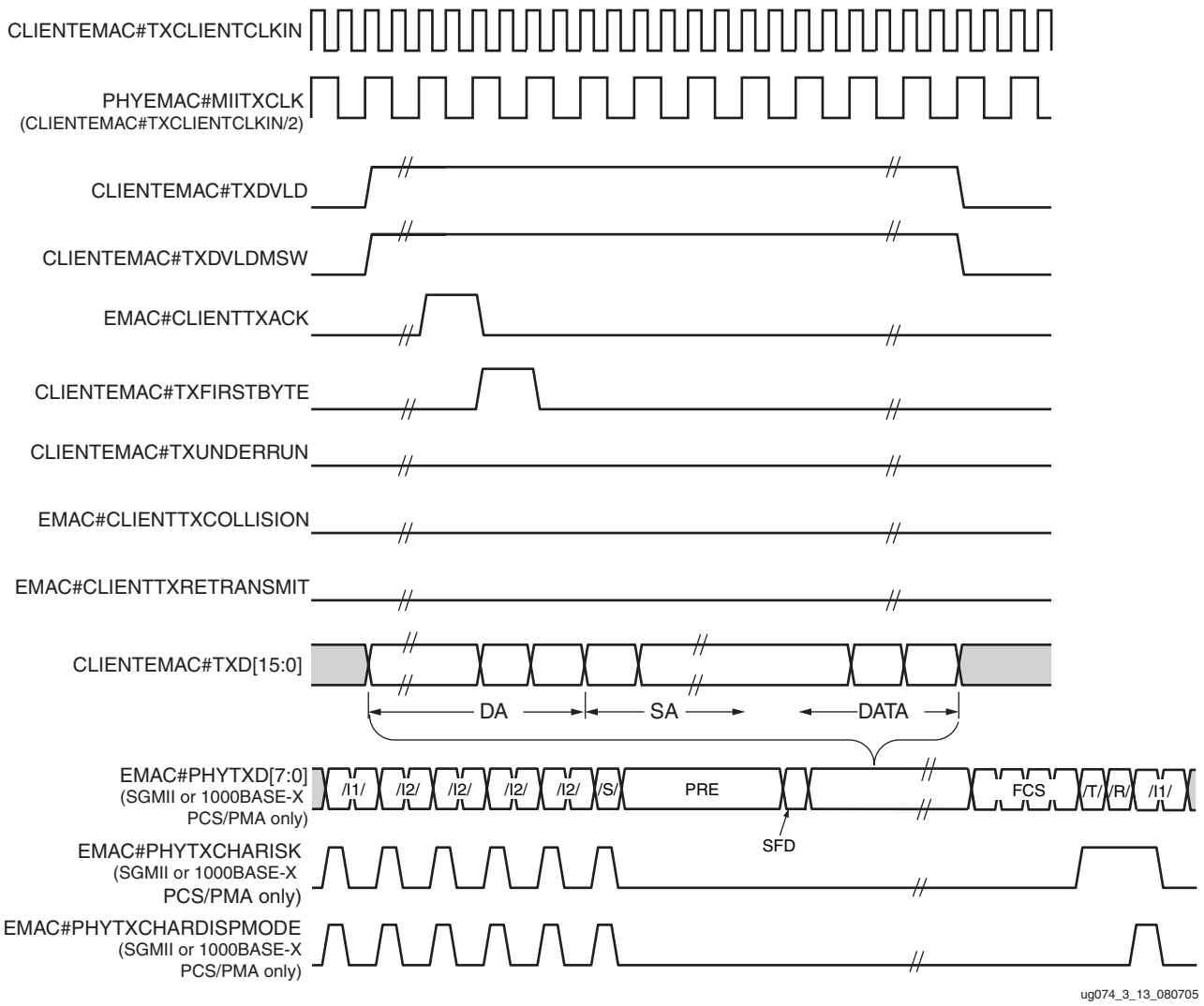


Figure 3-11: **16-Bit Transmit (Even Byte Case)**

Figure 3-12 shows the timing of a normal outbound frame transfer for the case with an odd number of bytes in the frame.

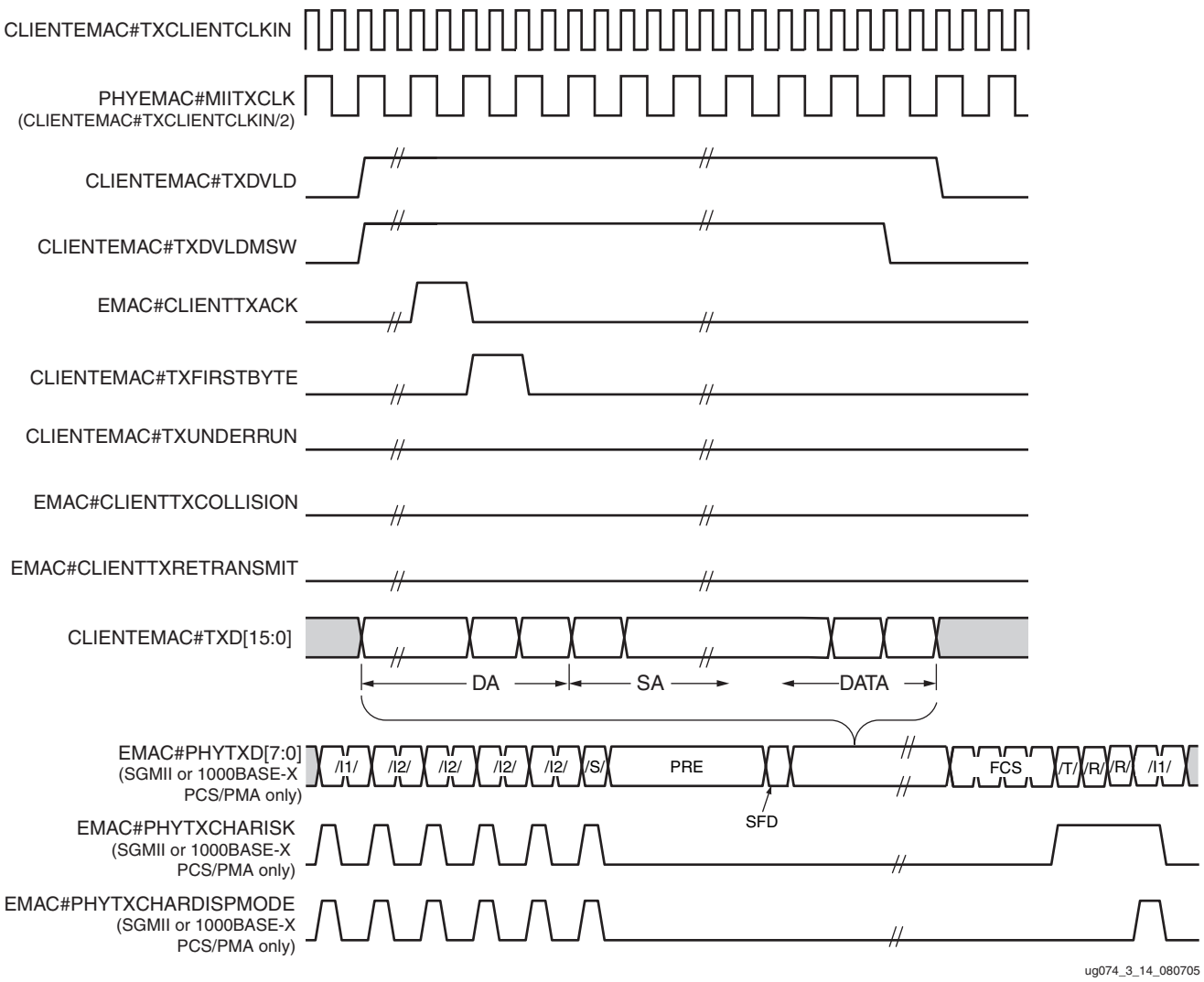


Figure 3-12: **16-Bit Transmit (Odd Byte Case)**

As shown in Figure 3-12, CLIENTEMAC#TXDVLDMSW denotes an odd number of bytes in the frame. In the odd byte case, CLIENTEMAC#TXDVLDMSW is deasserted one clock cycle earlier than the CLIENTEMAC#TXDVLD signal, after the transmission of the frame. Otherwise, these data valid signals are the same as shown in the even byte case (Figure 3-11).

Back-to-Back Transfers

For back-to-back transfers, both the CLIENTEMAC#TXDVLD and CLIENTEMAC#TXDVLDMSW must be deasserted for one PHYEMAC#MIITXCLK clock cycle (half the clock frequency of CLIENTEMAC#TXCLIENTCLKIN) after the first frame. During the following PHYEMAC#MIITXCLK clock cycle, both CLIENTEMAC#TXDLVD and CLIENTEMAC#TXDVLDMSW must be set High to indicate that the first two bytes of the destination address of the second frame is ready for transmission on CLIENTEMAC#TXD[15:0]. In 16-bit mode, this one PHYEMAC#MIITXCLK clock cycle IFG corresponds to a 2-byte gap (versus a 1-byte gap in 8-bit mode) between frames in the back-to-back transfer.

Figure 3-13 shows the timing diagram for 16-bit transmit for an even-byte case, and Figure 3-14 shows the timing diagram for an odd-byte case.

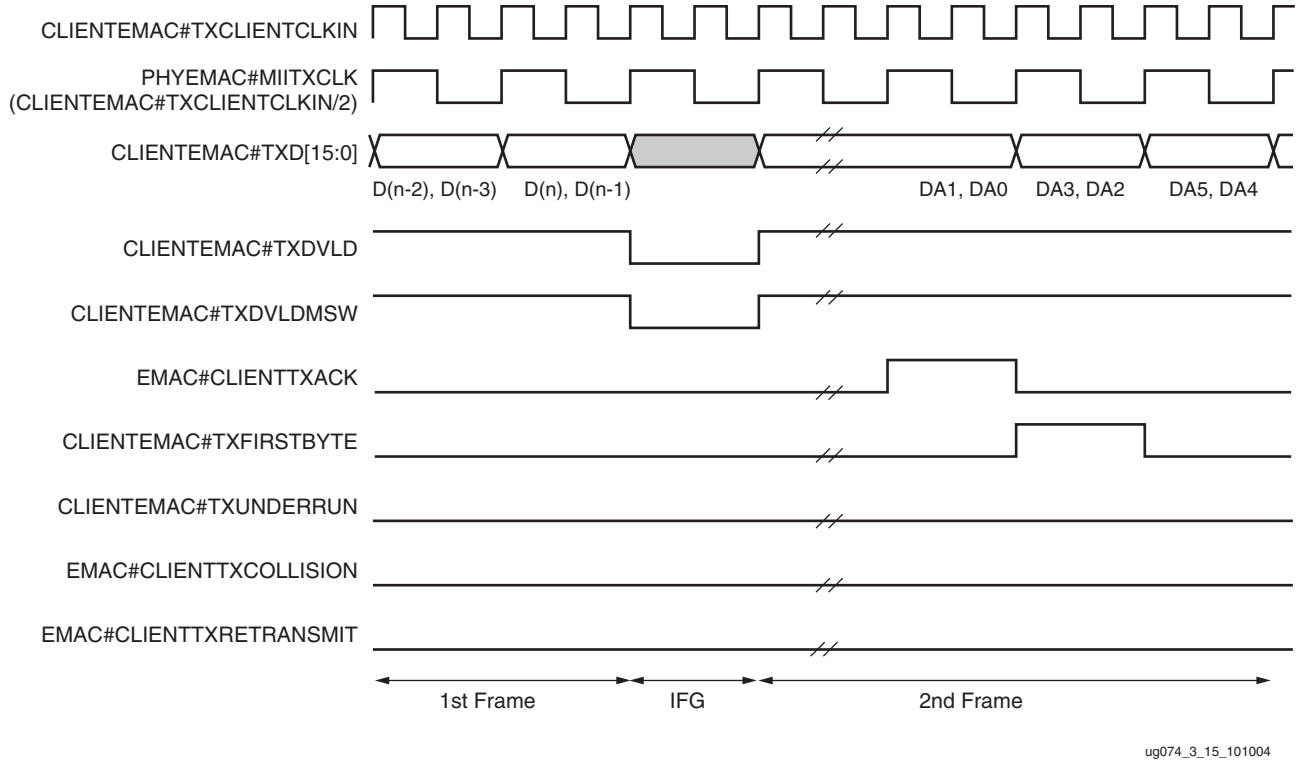


Figure 3-13: **16-Bit Transmit Back-to-Back Transfer (Even Byte Case)**

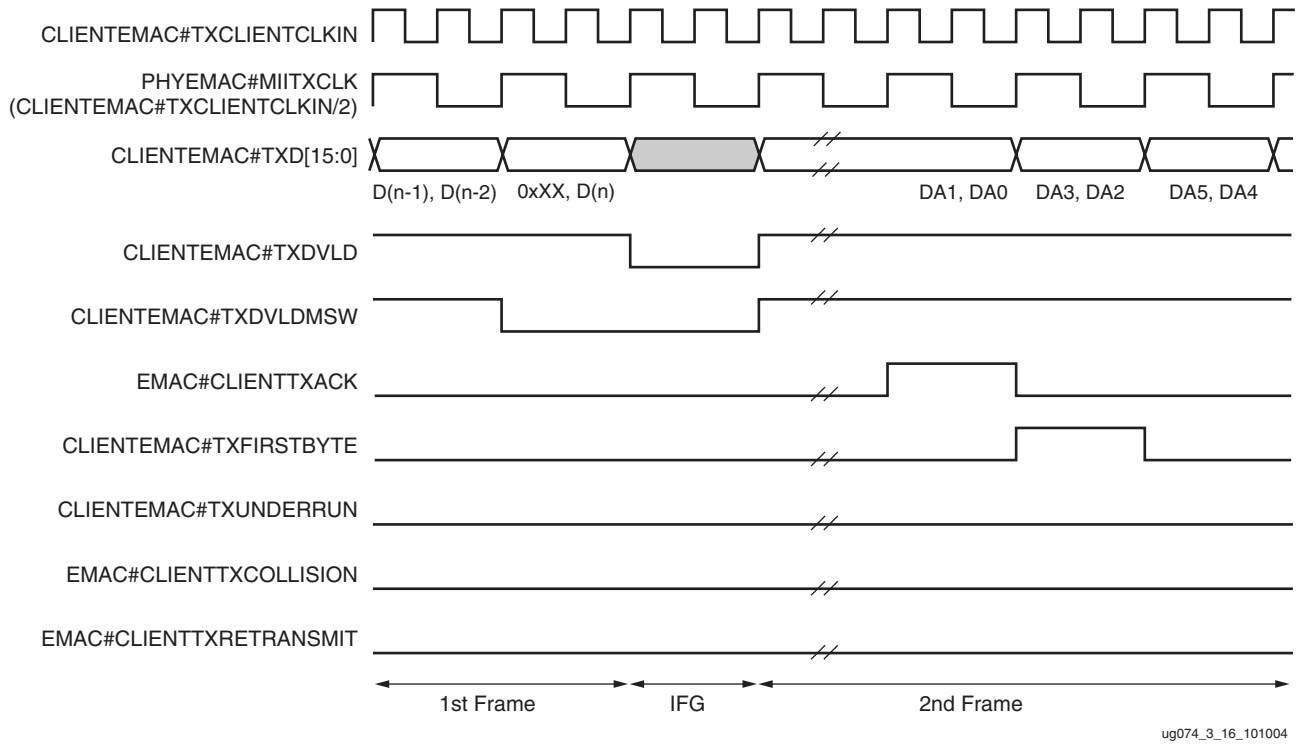


Figure 3-14: **16-Bit Transmit Back-to-Back Transfer (Odd Byte Case)**

Receive (RX) Client – 8-bit Wide Interface

In this configuration, EMAC#CLIENTRXD[15:8] and EMAC#CLIENTRXDVLDMSW signals are left unconnected.

Normal Frame Reception

The timing of a normal inbound frame transfer is shown in Figure 3-15. The client must accept data at any time; there is no buffering within the Ethernet MAC to allow for receive client latency. After frame reception begins, data is transferred on consecutive clock cycles to the receive client until the frame is complete. The Ethernet MAC asserts the EMAC#CLIENTRXGOODFRAME signal to indicate successful receipt of the frame and the ability to analyze the frame by the client.

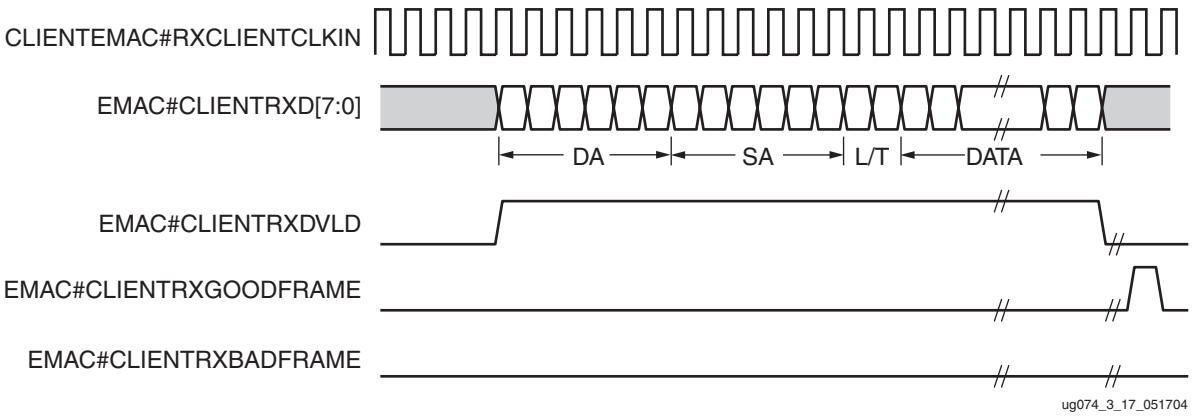


Figure 3-15: **Normal Frame Reception**

Frame parameters (destination address, source address, LT, data, and optionally FCS) are supplied on the data bus as shown in the timing diagram. The abbreviations are the same as those described in Table 3-1, page 39.

If the LT field has a length interpretation, the inbound frame could be padded to meet the Ethernet minimum frame size specification. This padding is not passed to the client in the data payload; an exception is when FCS passing is enabled. See "Client-Supplied FCS Passing," page 54.

Therefore, when client-supplied FCS passing is disabled, EMAC#CLIENTRXDVLD = 0 between frames for the duration of the padding field (if present), the FCS field, carrier extension (if present), the IFG following the frame, and the preamble field of the next frame. When client-supplied FCS passing is enabled, EMAC#CLIENTRXDVLD = 0 between frames for the duration of carrier extension (if present), the IFG, and the preamble field of the following frame.

EMAC#CLIENTRXGOODFRAME, EMAC#CLIENTRXBADFRAME Timing

Although the timing diagram in Figure 3-15 shows the EMAC#CLIENTRXGOODFRAME signal asserted shortly after the last valid data on EMAC#CLIENTRXD[7:0], this is not always the case. The EMAC#CLIENTRXGOODFRAME or EMAC#CLIENTRXBADFRAME signals are asserted only after completing all the frame checks. This is after receipt of the FCS field (and after reception of the carrier extension if present).

Therefore, either EMAC#CLIENTRXGOODFRAME or EMAC#CLIENTRXBADFRAME is asserted following frame reception at the beginning of the IFG.

SGMII/1000BASE-X PCS/PMA

In SGMII or 1000BASE-X PCS/PMA mode, an entire data stream received from PHYEMAC#RXD is passed on to EMAC#CLIENTRXD with some latency. This includes the code group described in Table 3-1. The assertion of EMAC#RXCLIENTDVLD indicates the position of the first destination address byte all the way to the last byte of the payload. Figure 3-16 and Figure 3-17 show the timing of a normal inbound frame transfer in SGMII and 1000BASE-X PCS/PMA mode.

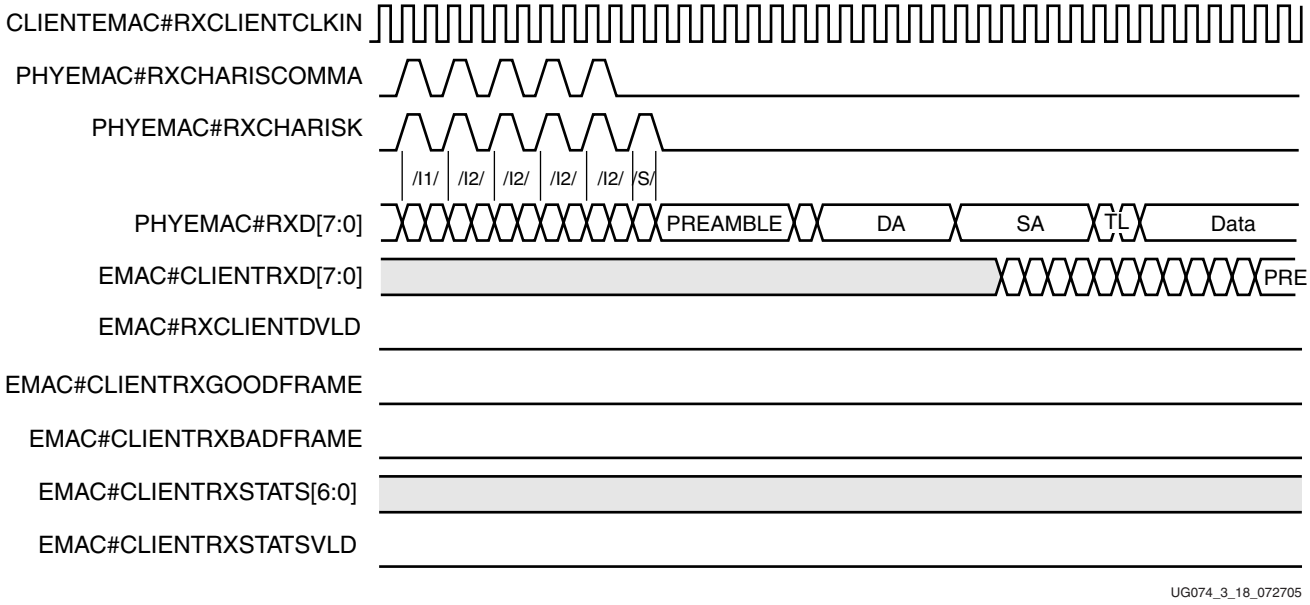


Figure 3-16: **Inbound Frame Transfer (Front)**

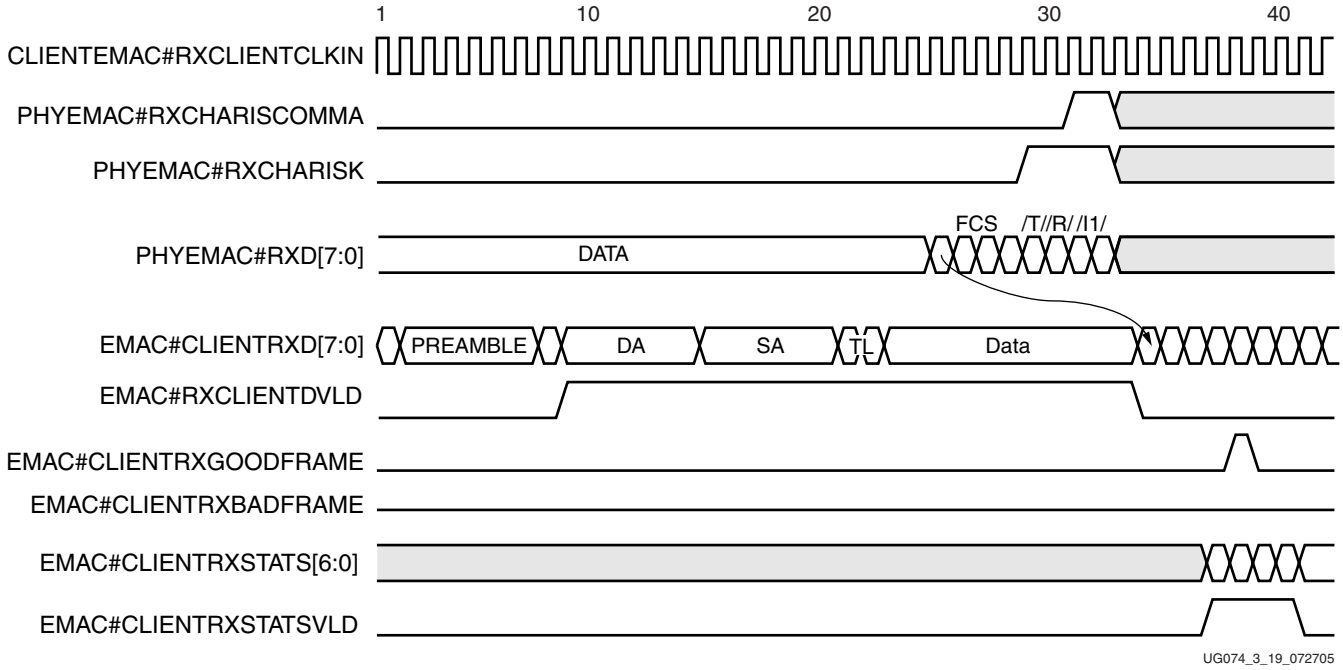


Figure 3-17: **Inbound Frame Transfer (Back)**

Frame Reception with Errors

An unsuccessful frame reception (for example, a fragment frame or a frame with an incorrect FCS) is shown in Figure 3-18. In this case, the EMAC#CLIENTRXBADFRAME signal is asserted to the client at the end of the frame. The client is responsible for dropping the data already transferred for this frame.

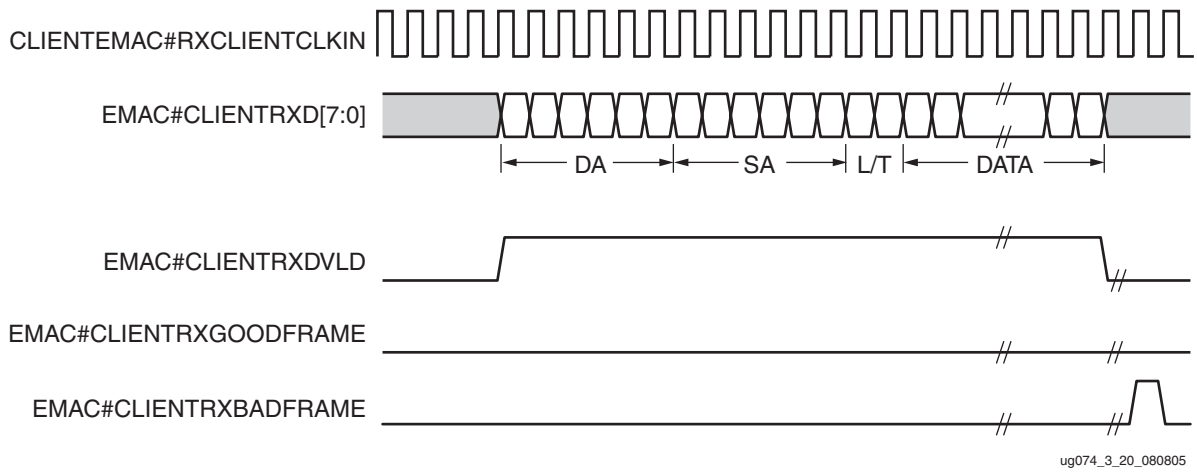


Figure 3-18: **Frame Reception with Error**

Figure 3-19 shows the timing of an unsuccessful frame reception in SGMII and 1000BASE-X PCS/PMA modes.

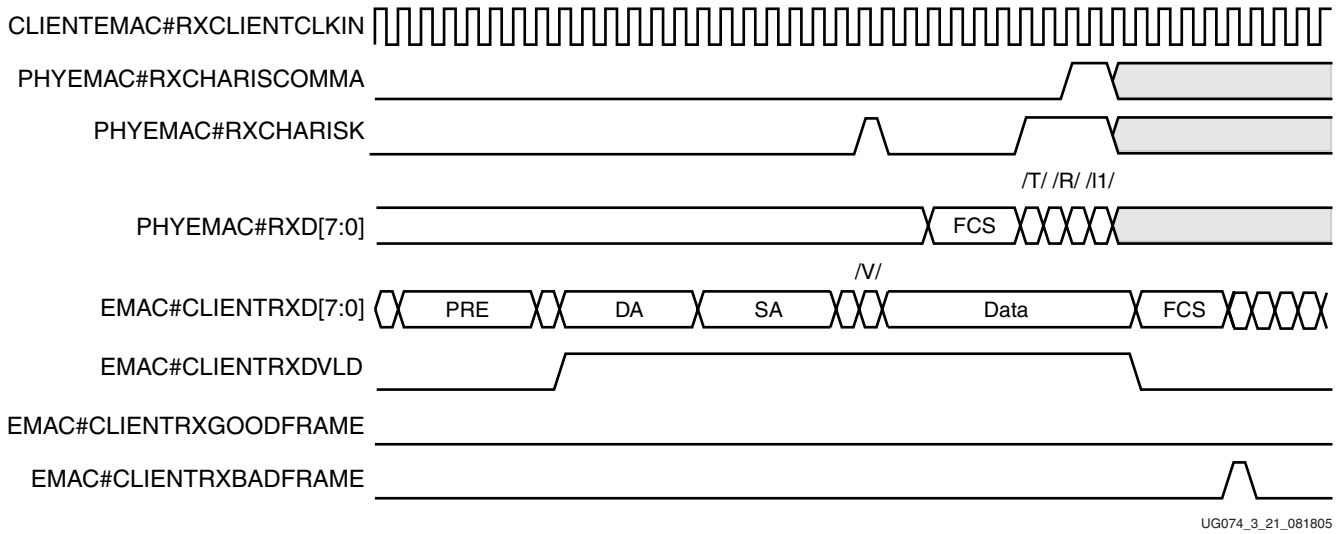


Figure 3-19: **Unsuccessful Frame Reception (SGMII and 1000BASE-X PCS/PMA Modes)**

The following conditions cause the assertion of EMAC#CLIENTRXBADFRAME:

- Standard Conditions:
  - FCS errors occur.
  - Packets are shorter than 64 bytes (undersize or fragment frames).

- VLAN frames of length 1519 to 1522 are received when VLAN frames are not enabled.
- Jumbo frames are received when jumbo frames are not enabled.
- A value of `0x0000` to `0x002D` is in the type/length field. In this situation, the frame should be padded to the minimum length. If not padded to exactly the minimum frame length, the frame is marked as bad (when length/type checking is enabled).
- A value of `0x002E` to `0x0600` is in the type/length field, but the real length of the received frame does not match this value (when length/type checking is enabled).
- Any control frame that is received is not exactly the minimum frame length (64 bytes).
- PHYEMAC#RXER is asserted at any point during frame reception.
- An error code is received in the 1-Gb frame extension field.
- A valid pause frame, addressed to the Ethernet MAC, is received when flow control is enabled.

Refer to "Flow Control Block," page 61 for more information.

- 1000BASE-X/SGMII Specific Conditions:

  When in 1000BASE-X or SGMII mode, these errors can also cause a frame to be marked as bad:

  - Unrecognized 8B/10B code group received during the packet.
  - 8B/10B running disparity errors occurred during the packet.
  - Unexpected K characters or sequences appeared in the wrong order/byte position.

Client-Supplied FCS Passing

Figure 3-20 shows the handling of the case where the Ethernet MAC is configured to pass the FCS field to the client (see "Configuration Registers," page 74). In this case, any padding inserted into the frame to meet Ethernet minimum frame length specifications is left intact and passed to the client.

Even though the FCS is passed up to the client, it is also verified by the Ethernet MAC, and EMAC#CLIENTRXBADFRAME is asserted if the FCS check fails.

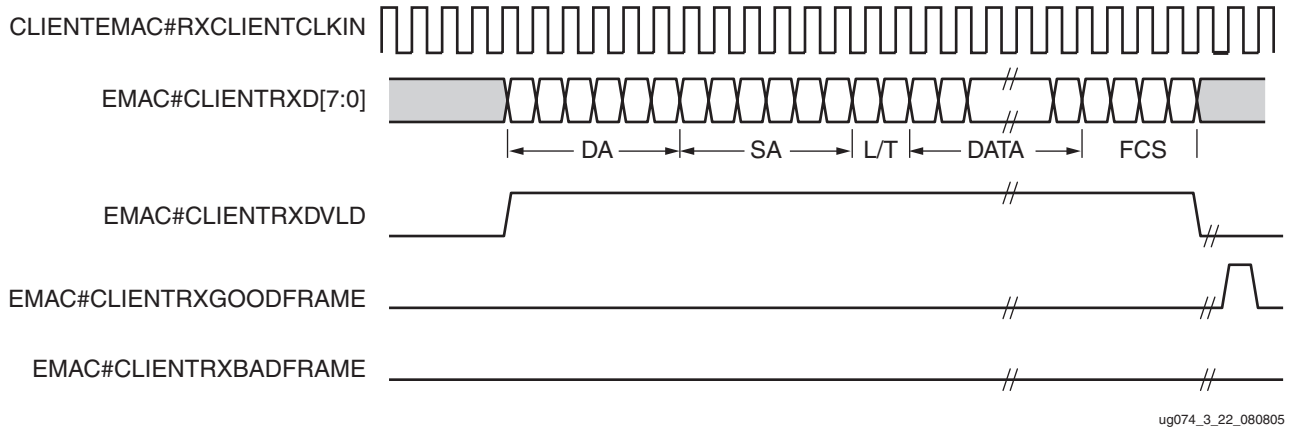


Figure 3-20: **Frame Reception with In-Band FCS Field**

VLAN Tagged Frames

The reception of a VLAN tagged frame (if enabled) is shown in Figure 3-21. The VLAN frame is passed to the client to identify the frame as VLAN tagged. This is followed by the tag control information bytes, V1 and V2. More information on the interpretation of these bytes is described in the IEEE Std 802.3-2002 standard.

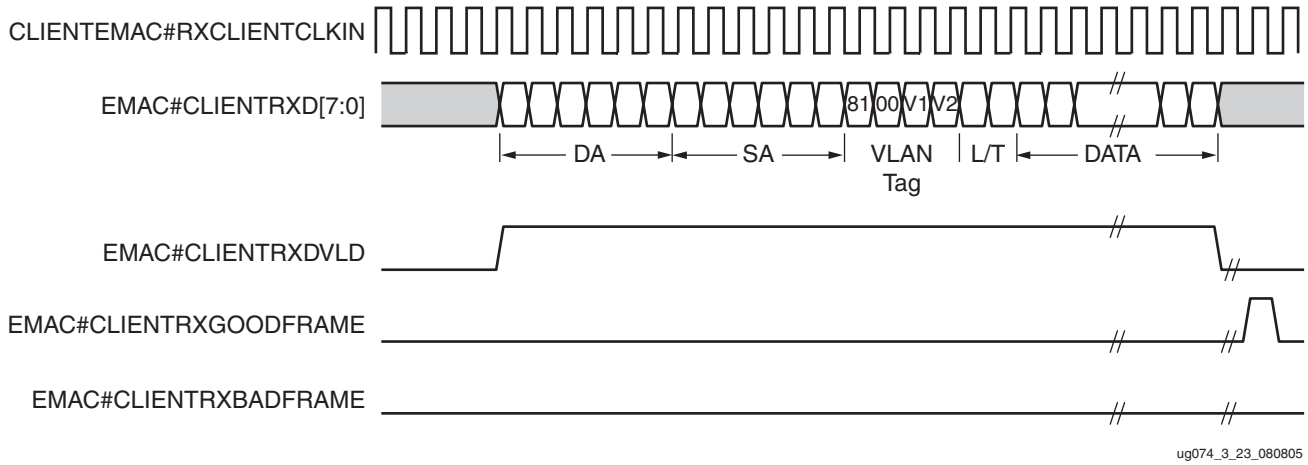


Figure 3-21: **Reception of a VLAN Tagged Frame**

Maximum Permitted Frame Length/ Jumbo Frames

The maximum length of a frame specified in the IEEE Std 802.3-2002 standard is 1518 bytes for non-VLAN tagged frames. VLAN tagged frames can be extended to 1522 bytes. When jumbo frame handling is disabled and the Ethernet MAC receives a frame exceeding the maximum legal length, EMAC#CLIENTRXBADFRAME is asserted. When jumbo frame handling is enabled, frames longer than the legal maximum are received in the same way as shorter frames.

For more information on enabling and disabling jumbo frame handling, see "Configuration Registers," page 74.

Length/Type Field Error Checks

Length/Type Field Error checking is specified in IEEE Std 802.3. The Length/Type Field Error checks consist of comparing the value in the Length/Type field with the actual size of the data field received. This check is only performed when the value in the Length/Type field is less than 1536 (0x600). This check does *not* perform any checks on the total length of the frame. This functionality must be enabled to comply with this specification. Disabling Length/Type checking is intended only for specific applications, such as when using over a proprietary backplane.

Enabled

When Length/Type error checking is enabled (see "Receiver Configuration Register (Word 1)," page 75), the following checks are made on frames with a value less than 1536 (0x600) in the Length/Type field. (If either of these checks fails, EMAC#CLIENTRXBADFRAME is asserted):

- A value greater than or equal to decimal 46 but less than decimal 1536 (a length interpretation) in the Length/Type field is checked against the actual data length received.
- A value less than decimal 46 in the Length/Type field is checked to ensure the data field is padded to exactly 46 bytes.

Furthermore, if padding is indicated (the Length/Type field is less than decimal 46) and client-supplied FCS passing is disabled, then the length value in the Length/Type field is used to deassert EMAC#CLIENTRXDVLD after the indicated number of data bytes, removing the padding bytes from the frame.

Disabled

When the Length/Type error checking is disabled (see "Receiver Configuration Register (Word 1)," page 75), the Length/Type error checks described above are not performed. A frame containing only these errors causes EMAC#CLIENTRXGOODFRAME to be asserted. Disabling this check does *not* disable total frame length checks. Any frame of less than 64 total bytes (minimum frame size) or any frame exceeding the maximum frame size checks (if enabled) will still cause EMAC#CLIENTRXBADFRAME to be asserted. Control frames will still be marked bad if they are not exactly 64 bytes in total length.

Furthermore, 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 EMAC#CLIENTRXDVLD. Instead EMAC#CLIENTRXDVLD is deasserted before the start of the FCS field; in this way, any padding is not removed from the frame.

Receive (RX) Client – 16-bit Wide Interface

This optional configuration can only be used when the Ethernet MAC is configured in 1000BASE-X PCS/PMA mode. The frequency of the receive client clock is half the frequency of the internal receive clock. The 16-bit client interface allows the Ethernet MAC to run at an internal clock frequency of greater than 125 MHz. The interface allows the Ethernet MAC to run at a line rate greater than 1 Gb/s as specified in IEEE Std 802.3. Therefore, the interface should not be used for Ethernet compliant applications, but it can be useful for customer's backplane applications.

The PHYEMAC#RXCLK is used as the input clock port for the CLIENTEMAC#RXCLIENTCLKOUT divided by two, as shown in the receive client block diagram in Figure 3-2, page 39. Using a DCM with the receive client clock,

EMACCLIENT#RXCLIENTCLKOUT as an input, the divide-by-two clock signal is generated. See Figure 4-28, page 139 for more information.

Figure 3-22 shows the timing of a normal inbound frame transfer for the case with an even number of bytes in the frame.

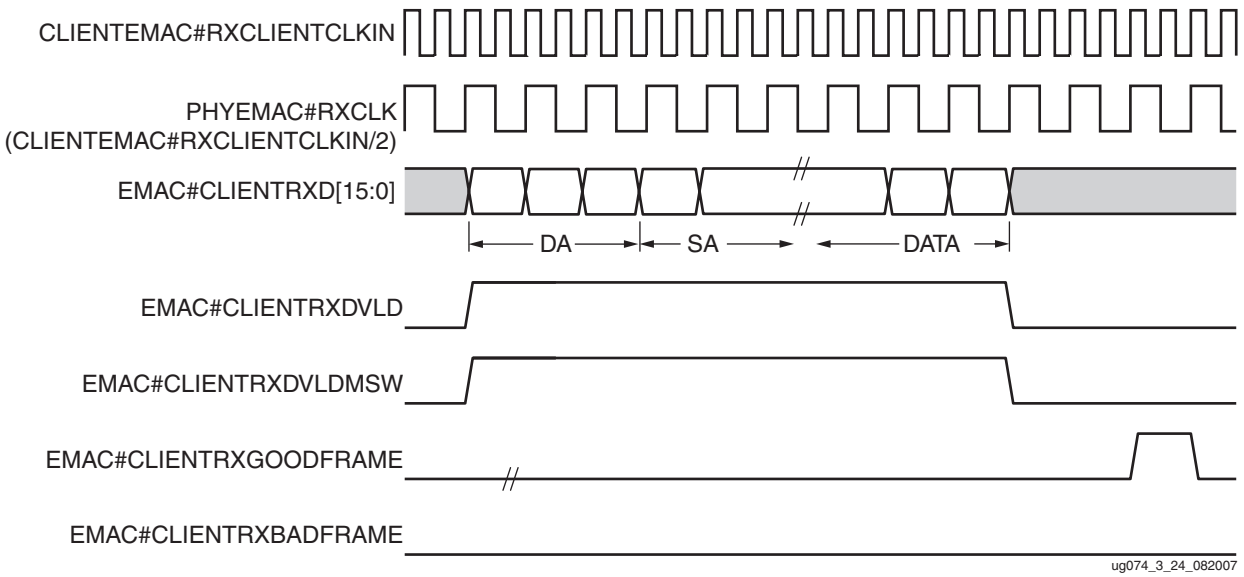


Figure 3-22: **16-Bit Receive (Even Byte Case)**

Figure 3-23 shows the timing of a normal inbound frame transfer for the case with an odd number of bytes in the frame.

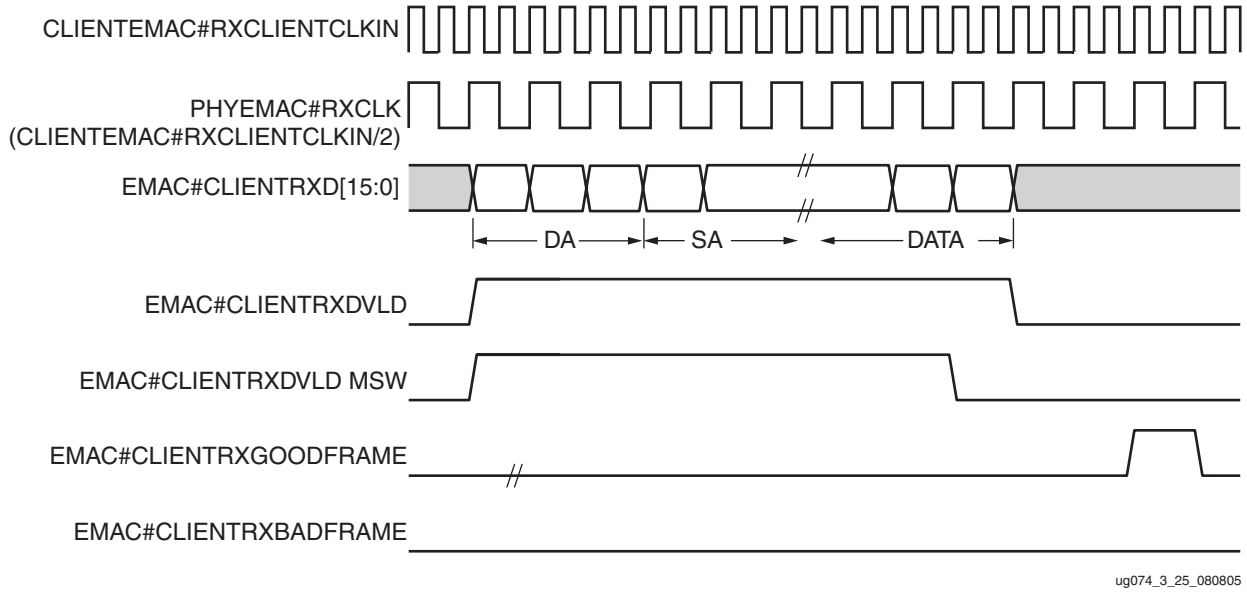


Figure 3-23: **16-Bit Receive (Odd Byte Case)**

As shown in Figure 3-22 and Figure 3-23, EMAC#CLIENTRXDVLDMSW is used to denote an odd number of bytes in the frame. The data valid signals are shown in the even byte case (Figure 3-22). In the odd byte case (Figure 3-23), EMAC#CLIENTRXDVLDMSW is deasserted one clock cycle earlier compared to the EMAC#CLIENTRXDVLD signal, after the reception of the frame. EMAC#CLIENTRXD[7:0] contains the data in this odd byte case.

Address Filtering

The address filtering block accepts or rejects frames by examining the destination address of an incoming frame. This block includes:

- Programmable unicast destination address matching
- Four programmable multicast address matching
- Broadcast address recognition (0xFFFF_FFFF_FFFF)
- Optional pass-through mode with address filter disabled (promiscuous mode)
- Pause control frame address recognition (0x0100_00C2_8001)

The Address Filter (AF) protects the client from extraneous traffic. With this technique, the hardware matches the Destination Address (DA) field of the Ethernet MAC frame. This relieves the task from the bus and software.

The AF is programmed in software through the host interface. Typically this includes unicast and multicast addresses. Pause-frame addresses and broadcast address are hardwired; they do not need to be programmed. The AF can be enabled and disabled under software control, using an enable bit in the control register. See "Address Filter Registers," page 80 for the control register.

When the function is enabled, Ethernet frames are passed to the client interface only if they pass the filter. When the AF function is disabled, all incoming RX frames are passed to the client interface.

For system monitoring, the event of a frame failing the filter is signaled. Equally, when a frame passes the filter, a match is indicated to the client by using the output pins EMAC#CLIENTRXDVLD and EMAC#CLIENTRXFRAMEDROP together. Table 3-2 shows the values of the two signals for possible outcomes of an incoming RX frame when the AF is enabled. Table 3-3 shows the two signal values when the AF is disabled (in promiscuous mode).

When the AF is enabled, it is impossible to determine if there is an incoming RX frame or if the AF has rejected incoming RX frames (using only EMAC#CLIENTRXDVLD) because in both cases, the EMAC#CLIENTRXDVLD is deasserted. See Table 3-2. However, using the EMAC#CLIENTRXFRAMEDROP signal, the nature of an incoming RX frame can be distinguished for system monitoring.

Table 3-2: **EMAC#CLIENTRXDVLD and EMAC#CLIENTRXFRAMEDROP Values When AF is Enabled**

EMAC#CLIENTRXDVLD	EMAC#CLIENTRXFRAMEDROP	Result of an Incoming RX Frame
0	0	No incoming RX frame
0	1	AF rejects RX frame
1	0	AF passes RX frame
1	1	N/A

Table 3-3: **EMAC#CLIENTRXDVLD and EMAC#CLIENTRXFRAMEDROP Values When AF is Disabled**

EMAC#CLIENTRXDVLD	EMAC#CLIENTRXFRAMEDROP	Result of an Incoming RX Frame
0	0	No incoming RX frame
0	1	N/A
1	0	AF passes RX frame
1	1	AF rejects RX frame

Host/Tie Interface

The host/tie interfaces provide the host, or fabric, access to the control registers for the address filter.

The tie-off interface allows the unicast address register, pause frame source address, and address filter promiscuous mode bit to be set directly by the fabric when the FPGA is configured. In this way, the address filter performs functions with the unicast address without using the host interface. The TIEEMAC#UNICASTADDR[47:0] and TIEEMAC#CONFIGVEC[47:0] should both be tied to the unicast address. TIEEMAC#UNICASTADDR[47:0] initializes unicast address word 0 [31:0] and unicast address word 1 [15:0] while TIEEMAC#CONFIGVEC[47:0] initializes the pause frame source address of the receiver configuration register word 0 [31:0] and receiver configuration register word 1[15:0]. TIEEMAC#CONFIGVEC[64] initializes the promiscuous mode bit (bit [31] of the address filter mode register). The tie interface does not initialize the four multicast address register values.

When the host interface is used, all the address filter registers are accessible by software, using either the DCR bus or the generic host bus. The tie interface initialization values to the registers can be overridden by the software through the host interface. Also, the four multicast address registers are programmed through the host interface.

Client RX Data/Control Interface

The AF generates the EMAC#CLIENTRXFRAMEDROP signal to inform the client that the destination MAC address of an incoming receive Ethernet frame does not match with any of the acceptable addresses stored in the AF. This control signal is asserted regardless of whether the AF is enabled or disabled (promiscuous mode).

Figure 3-24 shows the timing diagram when a frame matches a valid location in the AF (8-bit mode). The address filter is disabled in this timing diagram.

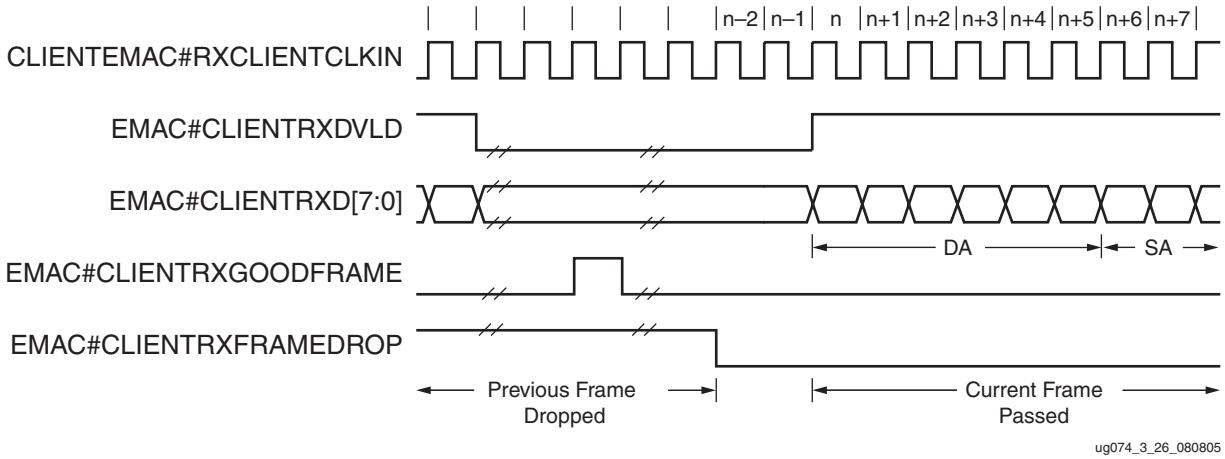


Figure 3-24: **Frame Matching Timing Diagram (8-Bit Mode)**

Figure 3-25 shows the timing diagram when a frame matches a valid location in the AF (16-bit mode). The address filter is disabled in this timing diagram.

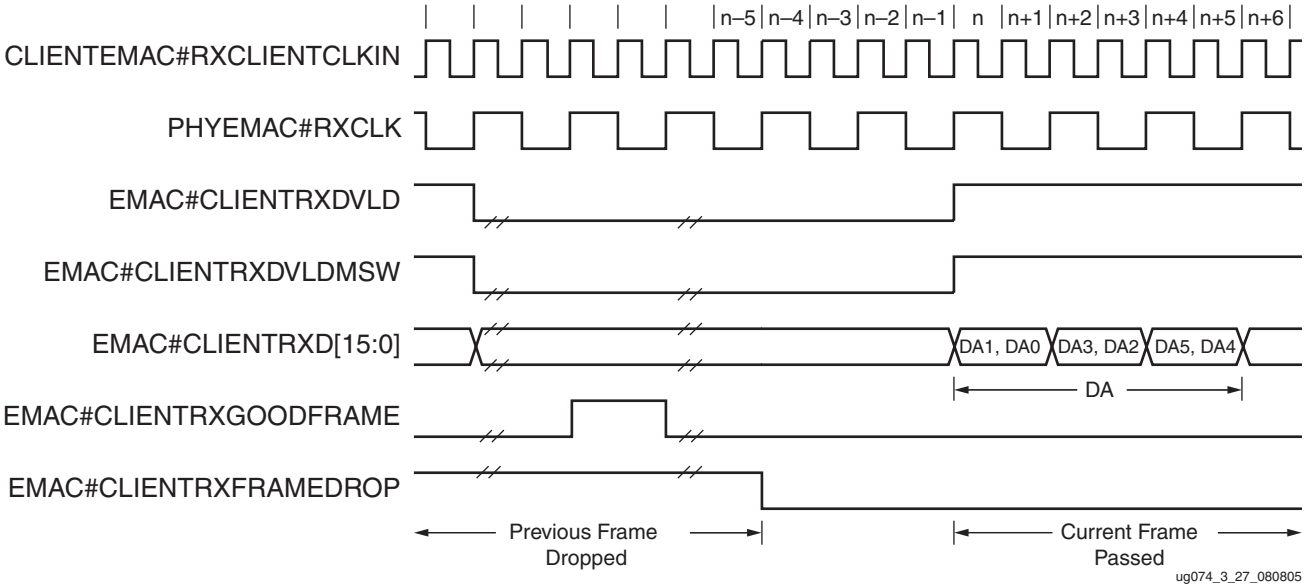


Figure 3-25: **Frame Matching Timing Diagram (16-Bit Mode)**

Figure 3-26 shows the timing diagram when a frame fails to match a valid location in the AF (8-bit mode) and the frame drop signal is generated. The address filter is disabled in this timing diagram.

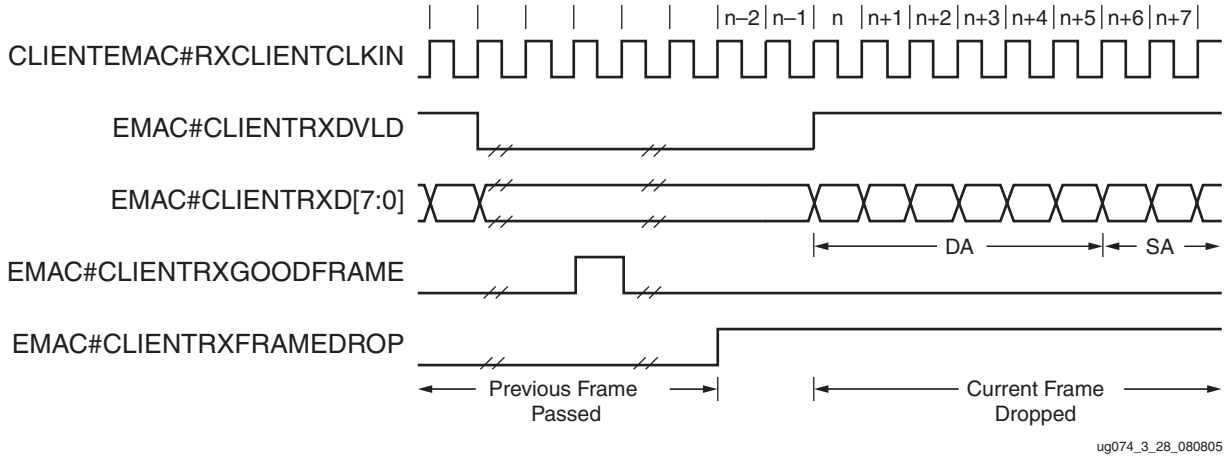


Figure 3-26: **Frame Matching Failed Timing Diagram (8-Bit Mode)**

Figure 3-27 shows the timing diagram when a frame fails to match a valid location in the AF (16-bit mode) and the frame drop signal is generated. The address filter is disabled in this timing diagram.

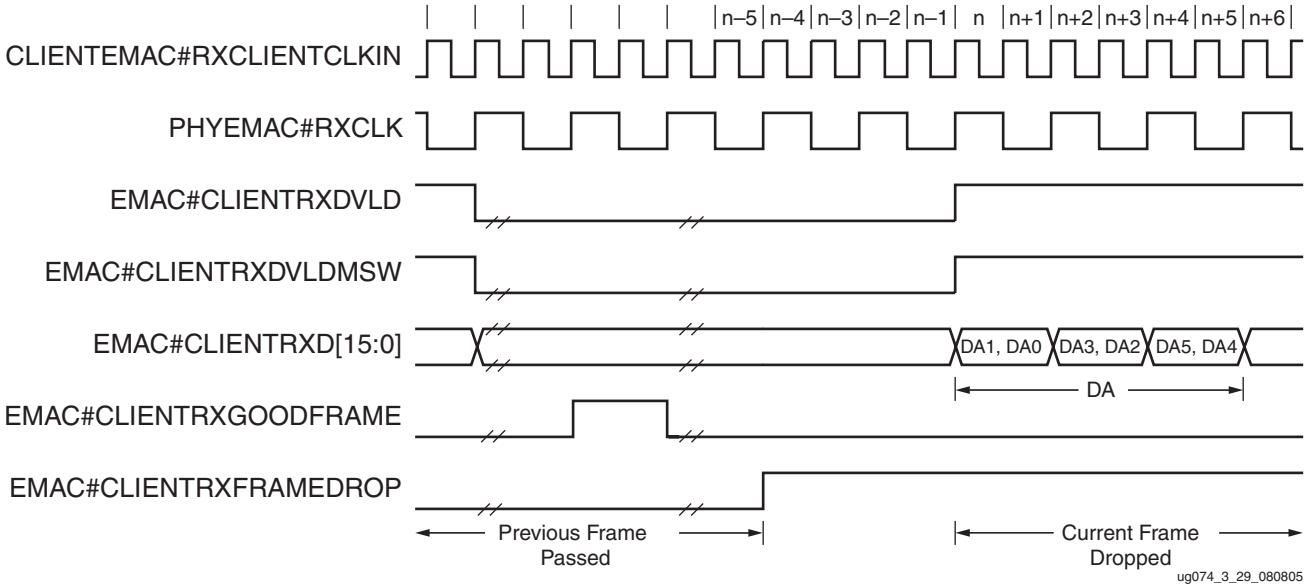


Figure 3-27: **Frame Matching Failed Timing Diagram (16-Bit Mode)**

Flow Control Block

The flow control block is designed to Clause 31 of the IEEE Std 802.3-2002 standard. The Ethernet MAC can be configured to send pause frames to act upon the pause frame's reception during full-duplex operation. These two behaviors can be configured asymmetrically (see "Configuration Registers," page 74).

Requirement for Flow Control

Figure 3-28 illustrates the requirement for flow control. The Ethernet MAC on the right side of the figure has a reference clock slightly faster than the nominal 125 MHz. The Ethernet MAC on the left side of the figure has a reference clock slightly slower than the nominal 125 MHz. This results in the Ethernet MAC on the left side of the figure not being able to match the full line rate of the Ethernet MAC on the right side (due to clock tolerances). The left Ethernet MAC is illustrated as performing a loopback implementation; this 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.

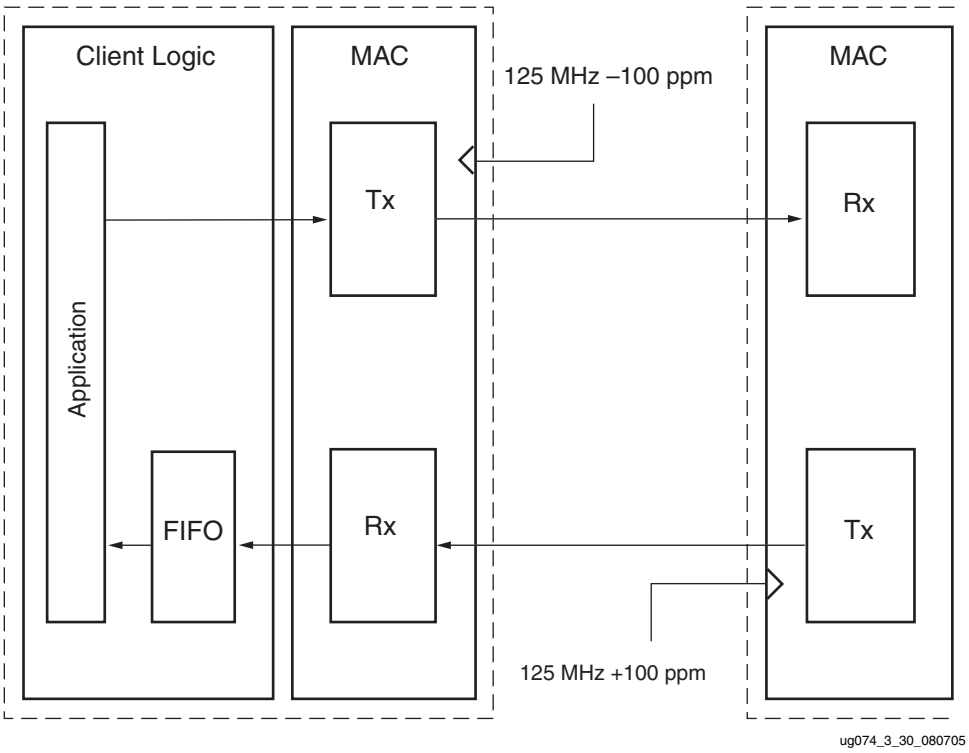


Figure 3-28: **Requirement for Flow Control**

Flow Control Basics

An Ethernet MAC transmits a pause control frame for the link partner to cease transmission for a defined period of time. For example, the left Ethernet MAC of Figure 3-28 initiates a pause request when the client FIFO (illustrated) reaches a nearly full state.

An Ethernet MAC responds 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 right Ethernet MAC of Figure 3-28 ceases transmission after receiving the pause control frame transmitted by the left Ethernet MAC. In a well-designed system, the right Ethernet MAC ceases transmission before the client FIFO of the left Ethernet MAC is overflowed. This provides time to empty the FIFO to a safe level before normal operation resumes. It also safe guards the system against FIFO overflow conditions and frame loss.

Transmitting a PAUSE Control Frame

The client initiates a flow control frame by asserting CLIENTEMAC#PAUSEREQ, when the pause value is on the CLIENTEMAC#PAUSEVAL[15:0] bus. These signals are synchronous to CLIENTEMAC#TXCLIENTCLKIN. The timing is shown in Figure 3-29.

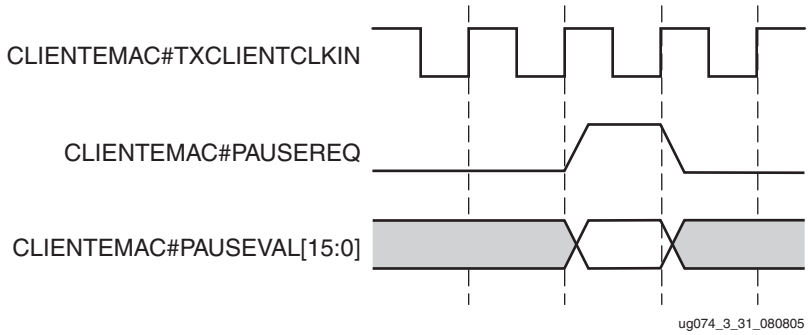


Figure 3-29: **Pause Request Timing**

When the Ethernet MAC is configured to support transmit flow control, a PAUSE control frame is transmitted on the link. When CLIENTEMAC#PAUSEREQ is asserted, the PAUSE parameter is set to the CLIENTEMAC#PAUSEVAL[15:0] value. This does not disrupt any frame transmission in progress, but it takes priority over any pending frame transmission. The PAUSE control frame is transmitted even if the transmitter is in a paused state. An example of a PAUSE frame (not drawn to scale) is shown in Figure 3-30.

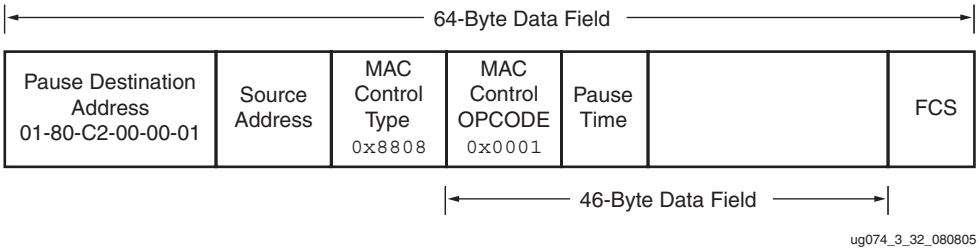


Figure 3-30: **Pause Frame Example**

The pause destination address can be configured using the "Configuration Registers". The pause_time in the PAUSE frame is the value from the CLIENTEMAC#PAUSEVAL[15:0].

Receiving a PAUSE Control Frame

When an error-free frame is received by the Ethernet MAC, it examines the following information:

- The destination address field is matched against the Ethernet MAC control multicast address and the configured source address for the Ethernet MAC (see "Configuration Registers," page 74).
- The LT field is matched against the Ethernet MAC control type code.
- If the second match is true, the OPCODE field contents are matched against the Ethernet MAC control OPCODE.

If any match is false or the Ethernet MAC flow control logic for the receiver is disabled, the frame is ignored by the flow control logic and passed up to the client.

If the frame passes all of the previous checks, is of minimum legal size, and the Ethernet MAC flow control logic for the receiver is enabled, then the pause value parameter in the frame is used to inhibit transmitter operation for a time defined in the IEEE Std 802.3-2002 specification. This inhibit is implemented using the same back pressure scheme shown in Figure 3-6. Because the received pause frame is completed, it is passed to the client with EMAC#CLIENTRXBADFRAME asserted, indicating to the client that the frame should be dropped.

If the second match is true and the frame is not 64 bytes in length, the reception of any frame is considered to be an invalid control frame. This frame is ignored by the flow control logic and passed to the client with EMAC#CLIENTRXBADFRAME asserted. EMAC#CLIENTRXBADFRAME will be asserted even if flow control is not enabled.

Flow Control Implementation Example

In the system illustrated in Figure 3-28, the Ethernet MAC on the left side of the figure cannot match the full line rate of the right Ethernet MAC due to clock tolerances. Over time, the FIFO fills and overflows.

This example implements a flow control method to reduce, over a long time period, the full line rate of the right Ethernet MAC to less than the full line rate capability of the left Ethernet MAC.

Method

1. Choose a FIFO with a nearly full occupancy threshold. A 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 and causes the right Ethernet MAC to cease transmission and the FIFO of the left Ethernet MAC to start emptying.
2. Choose a second FIFO with an occupancy threshold of 3/4 (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 Ethernet MAC immediately resumes transmission (i.e., there is no wait for the original requested pause duration to expire). This PAUSE control frame can, therefore, be considered a “pause cancel” command.

Operation

Figure 3-31 illustrates the FIFO occupancy over time.

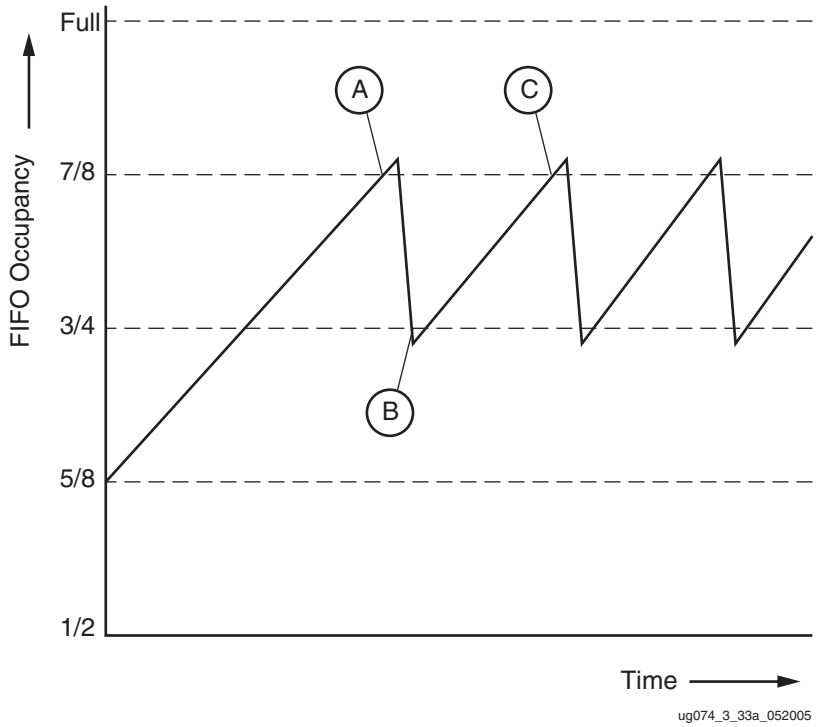


Figure 3-31: **Flow Control Implementation Triggered from FIFO Occupancy**

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

Statistics Vector

Transmitter Statistics Vector

TX_STATISTICS_VECTOR contains the statistics for the frame transmitted. The vector is driven synchronously by the transmitter clock, CLIENTEMAC#TXCLIENTCLKIN, following frame transmission. The bit field definition for the transmitter statistics vector is defined in Table 3-4.

The TX_STATISTICS_VECTOR is a 32-bit wide vector and is internal in the transmit engine. This vector is muxed out to a one-bit signal, EMAC#CLIENTTXSTATS, as shown in Figure 3-32.

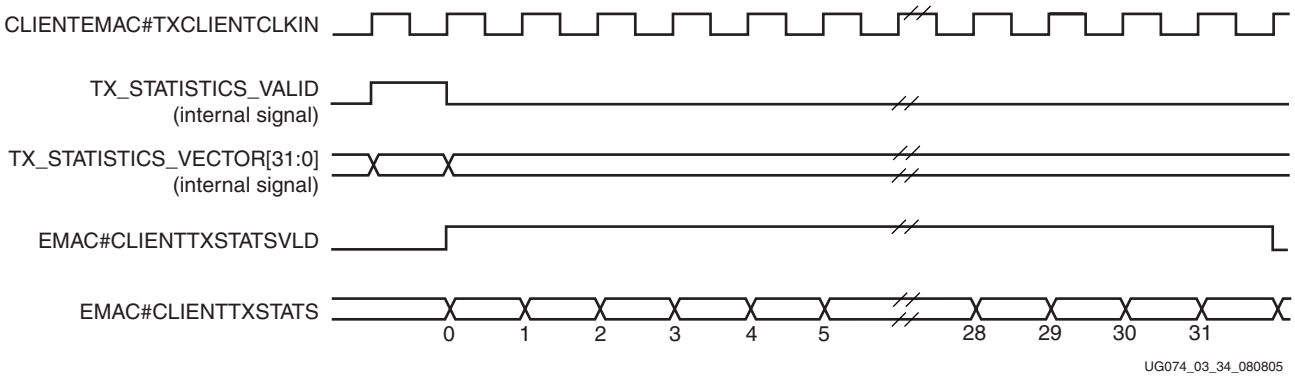


Figure 3-32: **Transmitter Statistics Mux Timing**

The block diagram for the transmitter statistics mux in the Ethernet MAC is shown in Figure 3-33.

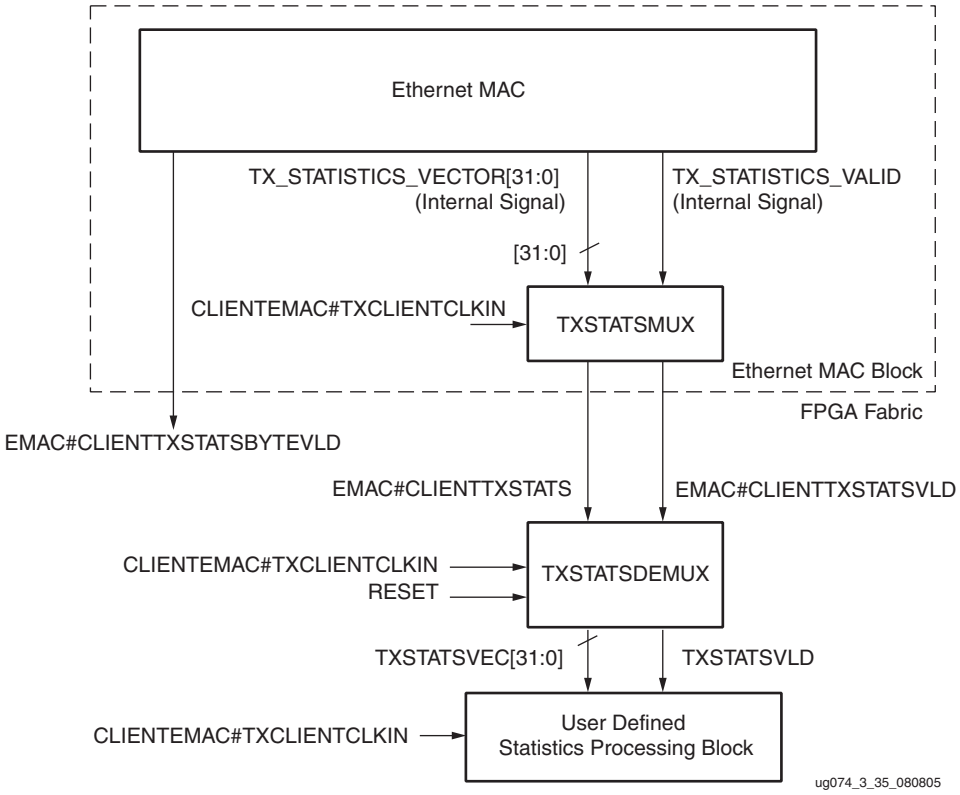


Figure 3-33: **Transmitter Statistics Mux Block Diagram**

All bit fields in EMAC#CLIENTTXSTATS are only valid when the EMAC#CLIENTTXSTATSVLD is asserted as illustrated in Figure 3-34. EMAC#CLIENTTXSTATSBYTEVLD is asserted if an Ethernet MAC frame byte (DA to FCS inclusive) is being transmitted. The signal is valid on every CLIENTEMAC#TXCLIENTCLKIN cycle.

TX_STATISTICS_VECTOR (bits 28 down to 20 inclusive) are only for half-duplex mode. When operating in full-duplex mode these bits are set to a logic 0.

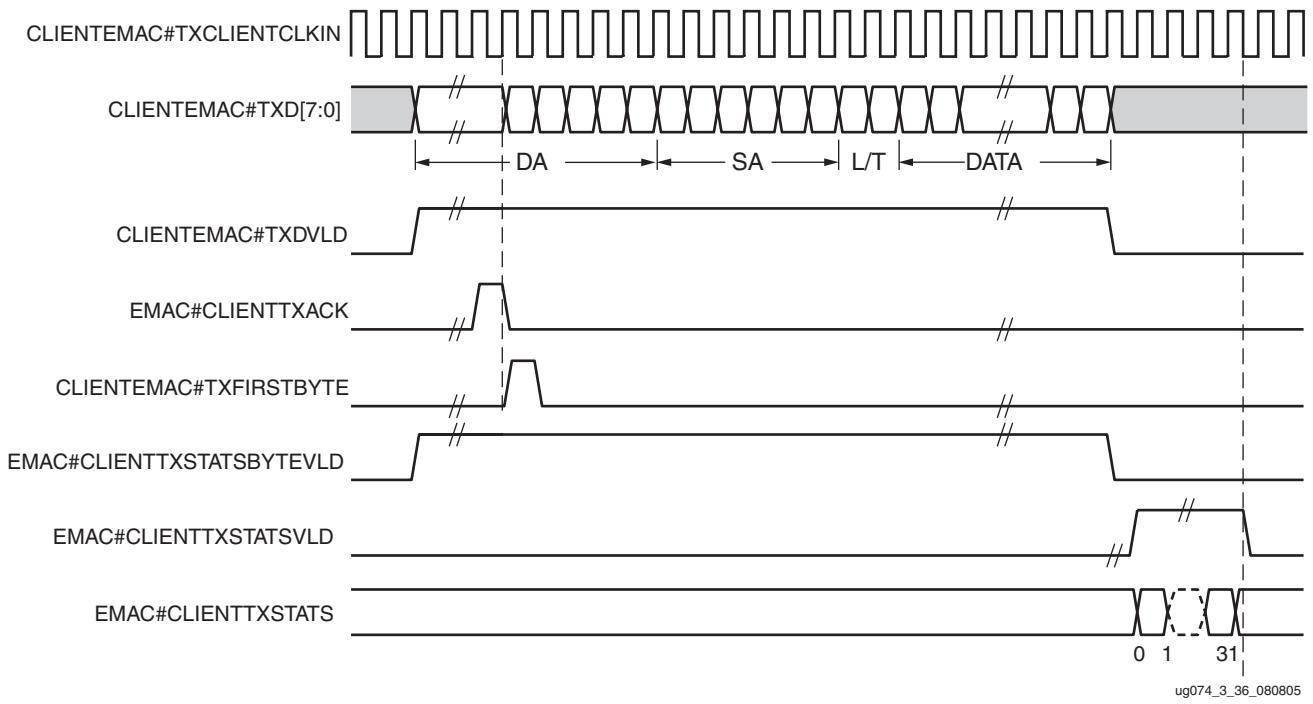


Figure 3-34: **Transmitter Statistics Vector Timing**

Table 3-4: **Bit Definitions for the Transmitter Statistics Vector**

TX_STATISTICS_VECTOR	Name	Description
31	PAUSE_FRAME_TRANSMITTED	Asserted if the previous frame was a pause frame initiated by the Ethernet MAC in response to asserting CLIENTEMAC#PAUSEREQ.
30	Reserved	Undefined.
29	Reserved	Returns a logic 0.
[28:25]	TX_ATTEMPTS[3:0]	The number of attempts made to transmit the previous frame. A 4-bit number where 0x0 = one attempt; 0x1 = two attempts, up to 0xF which describes 16 attempts.
24	Reserved	Returns a logic 0.
23	EXCESSIVE_COLLISION	Asserted if a collision is detected on each of the last 16 attempts to transmit the previous frame.
22	LATE_COLLISION	Asserted if a late collision occurred during frame transmission.
21	EXCESSIVE_DEFERRAL	Asserted if the previous frame was deferred for an excessive amount of time as defined by the maxDeferTime constant in the IEEE Std 802.3-2002 specification.

Table 3-4: **Bit Definitions for the Transmitter Statistics Vector *(Cont'd)***

TX_STATISTICS_VECTOR	Name	Description
20	TX_DEFERRED	Asserted if transmission of the frame was deferred.
19	VLAN_FRAME	Asserted if the previous frame contains a VLAN identifier in the LT field when transmitter VLAN operation is enabled.
[18:5]	FRAME_LENGTH_COUNT	The length of the previous frame in number of bytes. The count sticks at 16383 for jumbo frames larger than this value. Stops at 16383.
4	CONTROL_FRAME	Asserted if the previous frame has the special Ethernet MAC control type code 88-08 in the LT field.
3	UNDERRUN_FRAME	Asserted if the previous frame contains an underrun error.
2	MULTICAST_FRAME	Asserted if the previous frame contains a multicast address in the destination address field.
1	BROADCAST_FRAME	Asserted if the previous frame contains a broadcast address in the destination address field.
0	SUCCESSFUL_FRAME	Asserted if the previous frame is transmitted without error.

Notes:

1. Bits [28:20] of TX_STATISTICS_VECTOR are valid for half-duplex mode only.

Receiver Statistics Vector

RX_STATISTICS_VECTOR contains the statistics for the frame received. The vector is driven synchronously by the receiver clock, CLIENTEMAC#RXCLIENTCLKIN, following frame reception. The bit field definition for the receiver statistics vector is defined in Table 3-5.

The RX_STATISTICS_VECTOR is a 27-bit wide vector, and is internal in the receive engine. This vector is muxed out to a seven-bit wide signal EMAC#CLIENTRXSTATS[6:0] as shown in Figure 3-35.

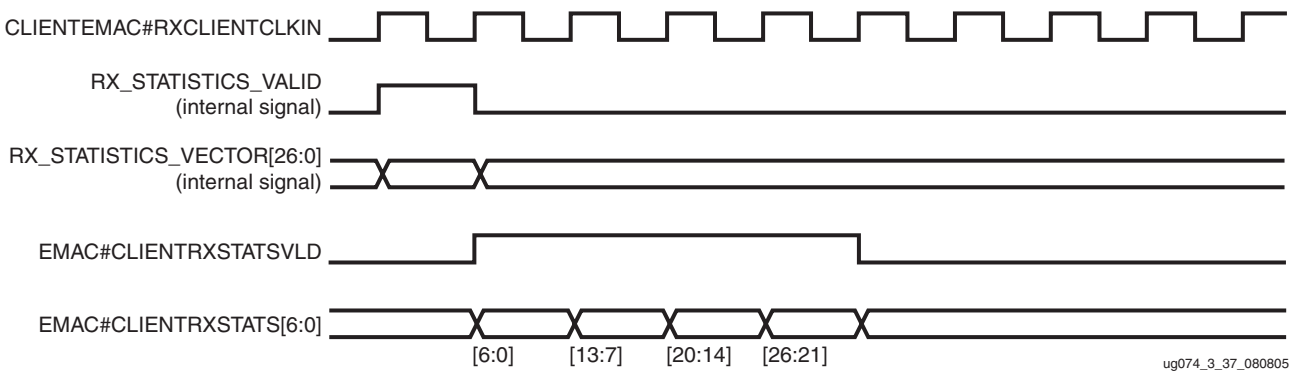


Figure 3-35: **Receiver Statistics MUX Timing**

The block diagram for the receiver statistics MUX in the Ethernet MAC is shown in Figure 3-36.

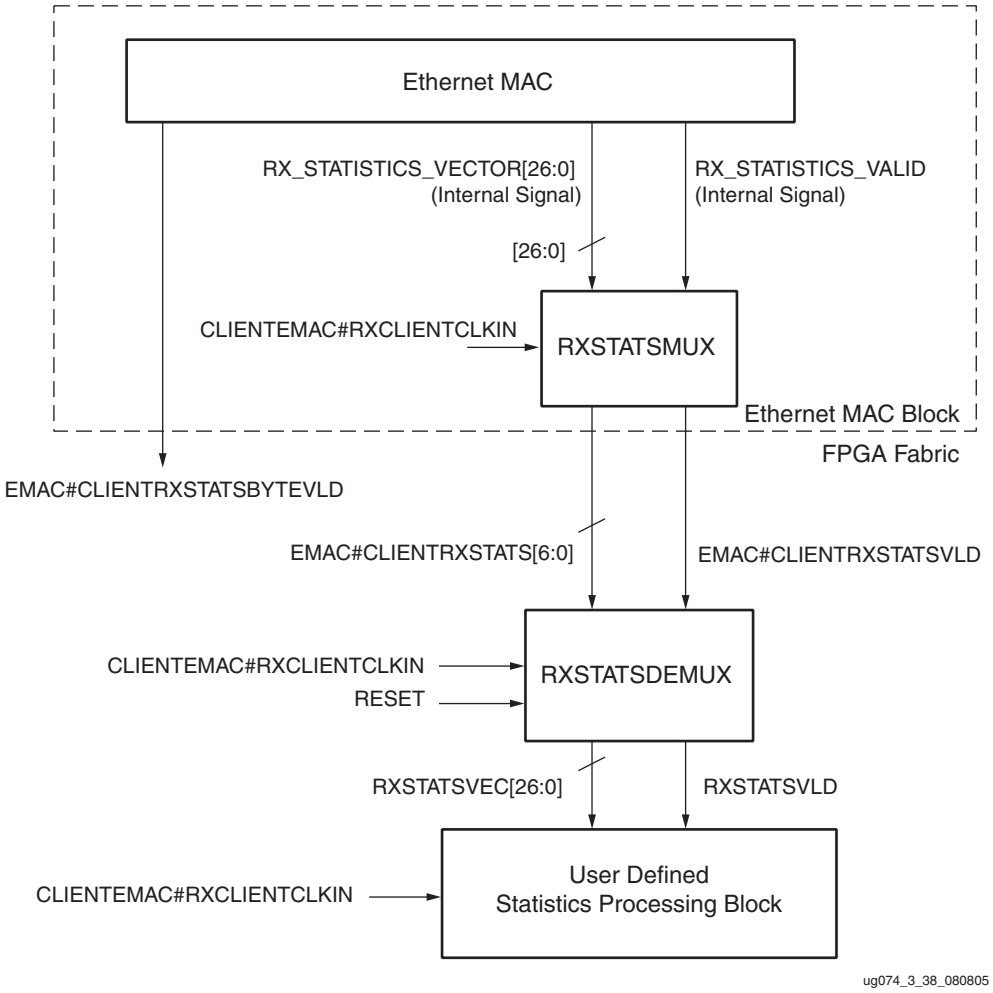


Figure 3-36: **Receiver Statistics MUX Block Diagram**

All bit fields for the EMAC#CLIENTRXSTATS[6:0] are valid only when the EMAC#CLIENTRXSTATSVLD is asserted as illustrated in Figure 3-37. EMAC#CLIENTRXSTATSBYTEVLD is asserted if an Ethernet MAC frame byte (DA to FCS inclusive) is received. This is valid on every CLIENTEMAC#RXCLIENTCLKIN cycle.

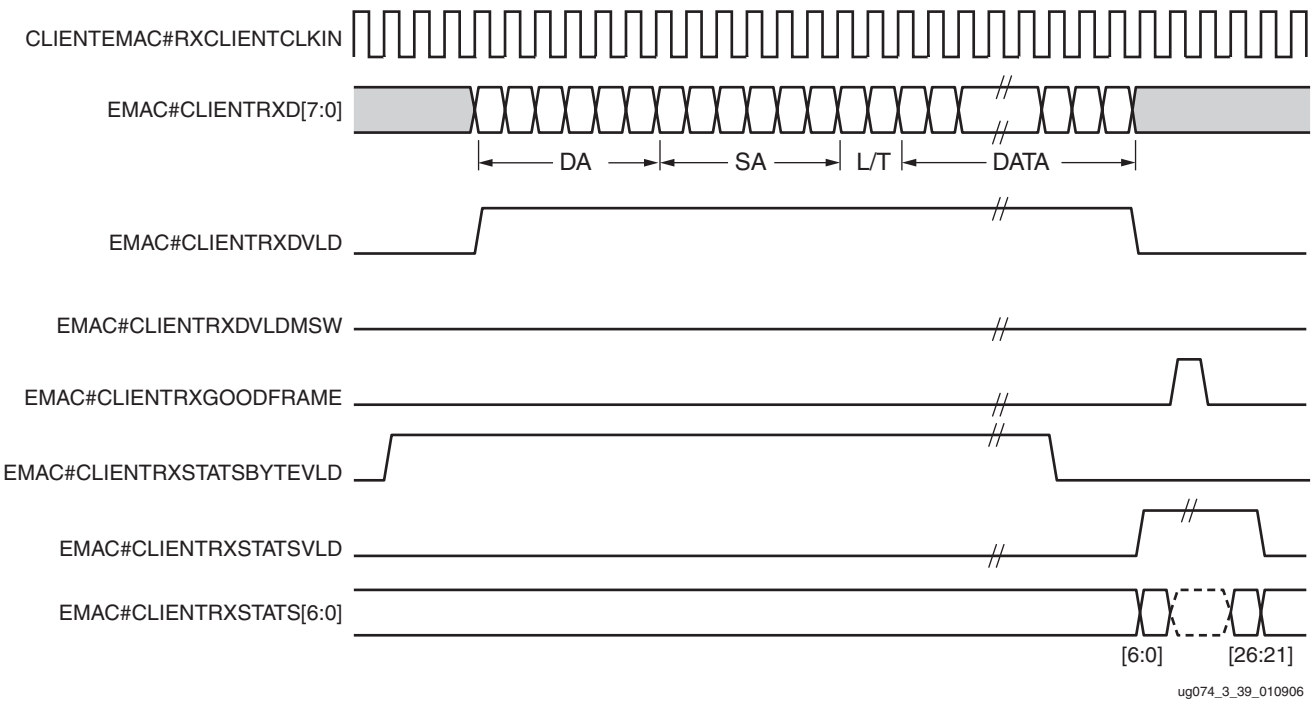


Figure 3-37: **Receiver Statistics Vector Timing**

Table 3-5: **Bit Definitions for the Receiver Statistics Vector**

RX_STATISTICS_VECTOR	Name	Description
26	ALIGNMENT_ERROR	Used in 10/100 MII mode. If an odd number of nibbles is received, the last nibble is ignored. If the frame without this nibble has an incorrect FCS, this bit is asserted. If the frame has a valid FCS, this bit is not asserted.
25	Length/Type Out of Range	Asserted if the LT field contains a length value that does not match the number of Ethernet MAC client data bytes received. Also asserted High if the LT field indicates that the frame contains padding but the number of Ethernet MAC client data bytes received is not equal to 46 bytes (minimum frame size).
24	BAD_OPCODE	Asserted if the previous frame is error free, contains the special control frame identifier in the LT field, but contains an OPCODE unsupported by the Ethernet MAC (any OPCODE other than PAUSE).

Table 3-5: **Bit Definitions for the Receiver Statistics Vector *(Cont'd)***

RX_STATISTICS_VECTOR	Name	Description
23	FLOW_CONTROL_FRAME	Asserted if the previous frame is error-free. Contains the special control frame identifier in the LT field. Contains a destination address matching either the Ethernet MAC control multicast address or the configured source address of the Ethernet MAC. Contains the supported PAUSE OPCODE and is acted upon by the Ethernet MAC.
22	Reserved.	Undefined.
21	VLAN_FRAME	Asserted if the previous frame contains a VLAN identifier in the LT field when receiver VLAN operation is enabled.
20	OUT_OF_BOUNDS	Asserted if the previous frame exceeded the specified IEEE Std 802.3-2002 maximum legal length (see "Maximum Permitted Frame Length/ Jumbo Frames," page 55). This is only valid if jumbo frames are disabled.
19	CONTROL_FRAME	Asserted if the previous frame contains the special control frame identifier in the LT field.
[18:5]	FRAME_LENGTH_COUNT	The length of the previous frame in number of bytes. The count sticks at 16383 for any jumbo frames larger than this value.
4	MULTICAST_FRAME	Asserted if the previous frame contains 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 has an incorrect FCS value or the Ethernet MAC detects error codes during frame reception.
1	BAD_FRAME(1)	Asserted if the previous frame received contains errors.
0	GOOD_FRAME(1)	Asserted if the previous frame received is error-free.

Notes:

1. If the length/type field error checks are disabled, then a frame containing this type of error is marked as a GOOD_FRAME, providing no additional errors were detected.

Statistics Registers/Counters

The Ethernet MAC does not collect statistics on the success and failure of various operations. A custom statistics counter can be implemented in the FPGA fabric to collect the statistics. A parameterizable Ethernet statistics core is available from the LogiCORE™ libraries. For more information, see: http://www.xilinx.com/xlnx/xebiz/designResources/ip_product_details.jsp?sGlobalNavPick=PRODUCTS&sSecondaryNavPick=Intellectual+Property&key=ETHERNET_STATS.

When the PowerPC 405 processor is used as a host processor, it can access the statistics counter registers in the FPGA fabric through the DCR bridge in the host interface. "Interfacing to an FPGA Fabric-Based Statistics Block" in Chapter 6 describes the access method.

Host Interface

To access the Ethernet MAC registers through the host interface, the user must set TIEEMACCONFIGURE[67] = 1. The host interface allows the user to:

- Program the configuration registers in the Ethernet MAC
- Read the accumulated statistics from the statistics unit implemented in the fabric (optional)
- Access the configuration registers and multicast address table register in the address filtering unit
- Access the Management Data I/O (MDIO) registers of the physical components attached to the Ethernet MACs

The two Ethernet MACs share a single host interface. The host interface brings the Ethernet MAC host bus from the Ethernet MAC out to the fabric. The host interface unit also contains a DCR bus bridge. This allows the user to access the Ethernet MAC registers through the DCR bus. Figure 3-38 shows the internal structure of the host interface. The EMAC1 signal is provided by the HOSTEMAC1SEL input signal when using the generic host bus or generated by the DCR bridge when using the DCR bus.

The DCREMACENABLE signal is used to select either the generic host bus or the DCR bus. When this signal is asserted, the host interface uses the DCR bus.

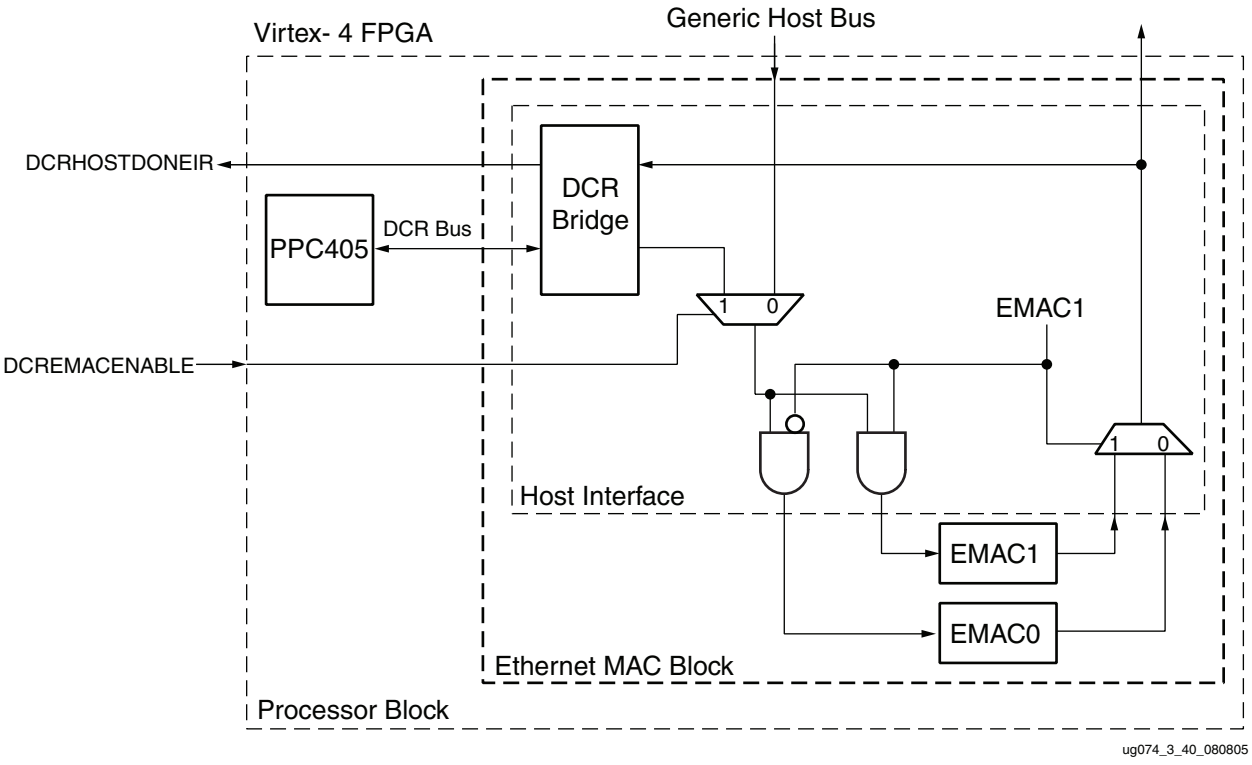


Figure 3-38: **Host Interface**

Figure 3-39 shows the block diagram of the host interface.

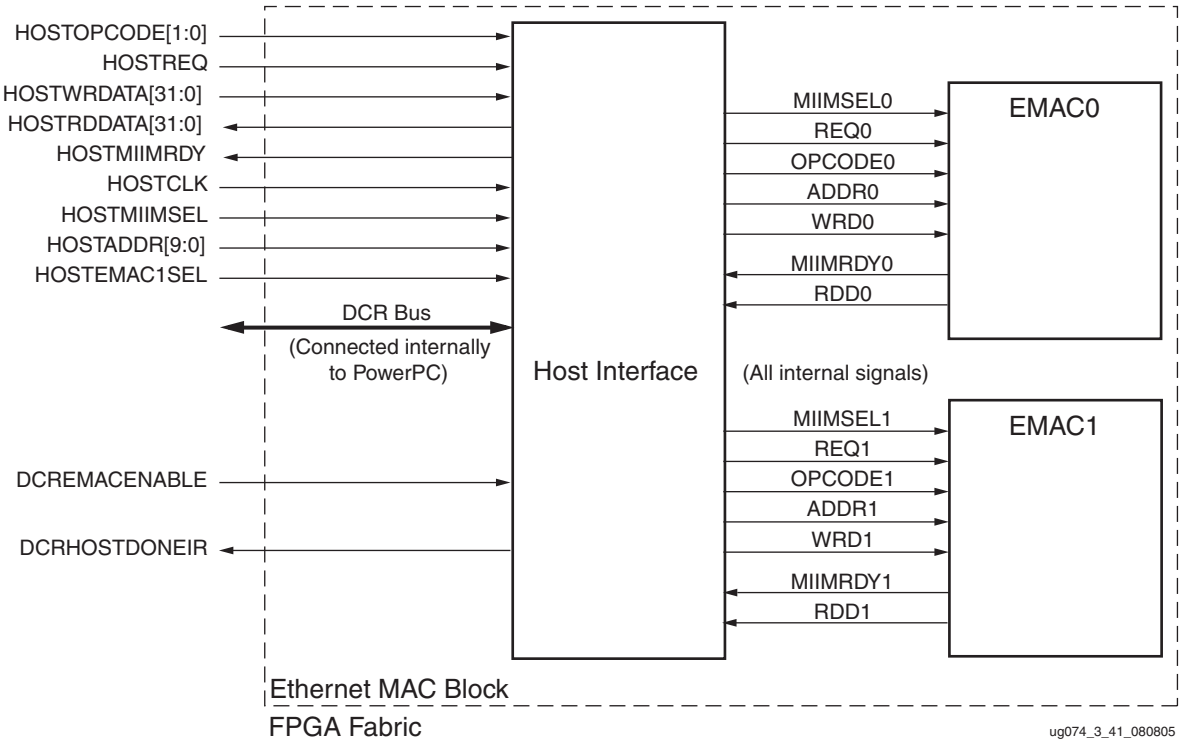


Figure 3-39: **Ethernet MAC Host Interface Block Diagram**

Generic Host Bus

When the generic host bus is used, the HOSTEMAC1SEL signal selects between the host access of EMAC0 or EMAC1. When HOSTEMAC1SEL is asserted, the host accesses EMAC1. If only one Ethernet MAC is used, this signal can be tied-off to use either one of the Ethernet MACs.

To use the DCR bus for the host interface, the DCREMACENABLE signal is asserted. Because the DCREMACENABLE signal is input from the fabric, it can be tied-off to select between the DCR bus or the generic host bus during the FPGA power-up configuration. When using the PowerPC processor and its DCR bus interface to control the Ethernet MAC registers, either connect the EMAC DCREMACENABLE port to the PPC405 DCREMACENABLER port or assert the DCREMACENABLE input to the EMAC.

The PowerPC processor serves as host processor when a DCR bus is used. Interrupt request is one of the methods used by the PowerPC processor to determine when the host interface completes a DCR host access command.

The interrupt request DCRHOSTDONEIR signal is only active when the DCR bus is used, and the host interface register IRENABLE is programmed to enable interrupt. This signal is active High and level sensitive. When a host access through the DCR bus is completed, the DCRHOSTDONEIR signal is asserted. The host needs to service the interrupt request and clear the host interface register (IRSTATUS) to deassert this signal. See "Using the DCR Bus as the Host Bus," page 83 for a description of the DCR.

Access to the management interface depends on the type of transaction. Table 3-6 shows the access method required for each transaction type.

Table 3-6: **Management Interface Transaction Types**

Transaction	HOSTMIIMSEL	HOSTADDR[9]
Configuration/Address Filter	0	1
MDIO access	1	X

Host Clock Frequency

The host clock (HOSTCLK) is used to derive the MDIO clock, MDC, and is subject to the same frequency restrictions. See the *Virtex-4 FPGA Data Sheet* for the HOSTCLK frequency parameters.

Configuration Registers

The Ethernet MAC has seven configuration registers. These registers are accessed through the host interface and can be written to at any time. Both the receiver and transmitter logic only respond to configuration changes during IFGs. The configurable resets are the only exception, because the reset is immediate.

Configuration of the Ethernet MAC is performed through a register bank accessed through the Host interface. Any time an address shown in Table 3-7 is accessed, a 32-bit read or write is performed from the same configuration word, with the exception of the read-only Ethernet MAC mode configuration register and the RGMII/SGMII configuration register. Only the speed selection is both readable and writable in the Ethernet MAC mode configuration register.

Table 3-7: **Configuration Registers**

{HOSTEMAC1SEL, HOST_ADDR[9:0]}	Register Description
0x200	Receiver Configuration (Word 0)
0x240	Receiver Configuration (Word 1)
0x280	Transmitter Configuration
0x2C0	Flow Control Configuration
0x300	Ethernet MAC Mode Configuration
0x320	RGMII/SGMII Configuration
0x340	Management Configuration

Notes:

1. HOSTEMAC1SEL acts as bit 10 of HOSTADDR.

The configuration registers and the contents of the registers are shown in Table 3-8 through Table 3-14.

Table 3-8: **Receiver Configuration Register (Word 0)**

	MSB																													LSB
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x200	PAUSE_FRAME_ADDRESS[31:0]																															

Bit	Description	Default Value	R/W
[31:0]	Pause Frame Ethernet MAC Address [31:0]. This address is used to match the Ethernet MAC against the destination address of any incoming flow control frames. It is also used by the flow control block as the source address for any outbound flow control frames. Tie to the same value as TIEEMAC#UNICASTADDR[31:0].	TIEEMAC#CONFIGVEC[31:0]	R/W

Table 3-9: **Receiver Configuration Register (Word 1)**

	MSB																													LSB
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x240	RST	JUM	FCS	RX	VLAN	HD	LT_DIS	RESERVED									PAUSE_FRAME_ADDRESS[47:32]															

Bit	Description	Default Value	R/W
[15:0]	Pause frame Ethernet MAC Address [47:32]. Tie to the same value as TIEEMAC#UNICASTADDR[47:32].	TIEEMAC#CONFIGVEC[47:32]	R/W
[24:16]	Reserved.	–	
[25]	Length/Type Check disable. When this bit is 1, it disables the comparison of the L/T field with the size of the data.	TIEEMAC#CONFIGVEC[63]	R/W
[26]	Half-duplex mode: When this bit is 1, the receiver operates in half-duplex mode. When the bit is 0, the receiver operates in full-duplex mode.	TIEEMAC#CONFIGVEC[48]	R/W
[27]	VLAN enable: When this bit is 1, the receiver accepts VLAN tagged frames. The maximum payload length increases by four bytes.	TIEEMAC#CONFIGVEC[49]	R/W
[28]	Receive enable: When this bit is 1, the receiver block is enabled to operate. When the bit is 0, the receiver ignores activity on the physical interface receive port.	TIEEMAC#CONFIGVEC[50]	R/W
[29]	In-band FCS enable: When this bit is 1, the receiver passes the FCS field up to the client. When this bit is 0, the FCS field is not passed to the client. In either case, the FCS is verified on the frame.	TIEEMAC#CONFIGVEC[51]	R/W
[30]	Jumbo frame enable: When this bit is 1, the Ethernet MAC receiver accepts frames over the maximum length specified in IEEE Std 802.3-2002 specification. When this bit is 0, the receiver only accepts frames up to the specified maximum.	TIEEMAC#CONFIGVEC[52]	R/W
[31]	Reset: When this bit is 1, the receiver is reset. The bit automatically reverts to 0, This reset also sets all of the receiver configuration registers to their default values.	TIEEMAC#CONFIGVEC[53]	R/W

Table 3-10: **Transmitter Configuration Register**

	MSB																													LSB
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x280	RST	JUM	FCS	TX	VLAN	HD	IFG	RESERVED																								

Bit	Description	Default Value	R/W
[24:0]	Reserved.	–	
[25]	IFG adjustment enable: When this bit is 1, the transmitter reads the value of CLIENTEMAC#TXIFGDELAY at the start of frame transmission and adjusts the IFG.	TIEEMAC#CONFIGVEC[54]	R/W
[26]	Half-duplex mode (applicable in 10/100 Mb/s mode only): When this bit is 1, the transmitter operates in half-duplex mode. When this bit is 0, the transmitter operates in full-duplex mode.	TIEEMAC#CONFIGVEC[55]	R/W
[27]	VLAN enable: When this bit is 1, the transmitter allows transmission of the VLAN tagged frames.	TIEEMAC#CONFIGVEC[56]	R/W
[28]	Transmit enable: When this bit is 1, the transmitter is enabled for operation.	TIEEMAC#CONFIGVEC[57]	R/W
[29]	In-band FCS enable: When this bit is 1, the Ethernet MAC transmitter is ready for the FCS field from the client.	TIEEMAC#CONFIGVEC[58]	R/W
[30]	Jumbo frame enable: When this bit is 1, the transmitter sends frames greater than the maximum length specified in IEEE Std 802.3-2002. When this bit is 0, it only sends frames less than the specified maximum length.	TIEEMAC#CONFIGVEC[59]	R/W
[31]	Reset: When this bit is 1, the transmitter is reset. The bit automatically reverts to 0, This reset also sets all of the transmitter configuration registers to their default values.	TIEEMAC#CONFIGVEC[60]	R/W

Table 3-11: **Flow Control Configuration Register**

	MSB																													LSB
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x2C0	RSVD	FCTX	FCRX	RESERVED																												

Bit	Description	Default Value	R/W
[28:0]	Reserved.	–	
[29]	Flow control enable (RX): When this bit is 1, the received flow control frames inhibit transmitter operation. When this bit is 0, the flow control frame is passed to the client.	TIEEMAC#CONFIGVEC[62]	R/W
[30]	Flow control enable (TX): When this bit is 1, the CLIENTEMAC#PAUSEREQ signal is asserted and a flow control frame is sent from the transmitter. When this bit is 0, the CLIENTEMAC#PAUSEREQ signal has no effect.	TIEEMAC#CONFIGVEC[61]	R/W
[31]	Reserved.	–	

Table 3-12: **Ethernet MAC Mode Configuration Register**

	MSB																												LSB
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x300	LINK SPEED		RGMII	SGMII	GPCS	HOST	TX16	RX16	RESERVED																							

Bit	Description	Default Value	R/W
[23:0]	Reserved.	–	
[24]	Receive 16-bit Client Interface enable: When this bit is `1`, the receive data client interface is 16 bits wide. When this bit is `0`, the receive data client interface is 8 bits wide. This bit is valid only when using 1000BASE-X PCS/PMA mode.	TIEEMAC#CONFIGVEC[65]	R
[25]	Transmit 16-bit Client Interface enable: When this bit is `1`, the transmit data client interface is 16 bits wide. When this bit is `0`, the transmit data client interface is 8 bits wide. This bit is valid only when using 1000BASE-X PCS/PMA mode.	TIEEMAC#CONFIGVEC[66]	R
[26]	Host Interface enable: When this bit is `1`, the host interface is enabled. When this bit is `0`, the host interface is disabled. See "Tie-Off Pins" on page 28.	TIEEMAC#CONFIGVEC[67]	R
[27]	1000BASE-X mode enable: When this bit is `1`, the Ethernet MAC is configured in 1000BASE-X mode.	TIEEMAC#CONFIGVEC[68]	R
[28]	SGMII mode enable: When this bit is `1`, the Ethernet MAC is configured in SGMII mode.	TIEEMAC#CONFIGVEC[69]	R
[29]	RGMII mode enable: When this bit is `1`, the Ethernet MAC is configured in RGMII mode.	TIEEMAC#CONFIGVEC[70]	R
[31:30]	Speed selection: The speed of the Ethernet MAC is defined by the following values: `10` = 1000 Mb/s `01` = 100 Mb/s `00` = 10 Mb/s `11` = N/A	TIEEMAC#CONFIGVEC[72:71]	R/W

Table 3-13: **RGMII/SGMII Configuration Register**

	MSB																												LSB
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x320	SGMII LINK SPEED		RESERVED																										RGMII LINK SPEED		RGMII HD	RGMII Link

Bit	Description	Default Value	R/W
[0]	RGMII link: Valid in RGMII mode configuration only. When this bit is 1, the link is up. When this bit is 0, the link is down. This displays the link information from PHY to Ethernet MAC, encoded by GMII_RX_DV and GMII_RX_ER during the IFG.	0	R
[1]	RGMII half-duplex mode: Valid in RGMII mode configuration only. This bit is 0 for half-duplex mode and 1 for full-duplex mode. This displays the duplex information from PHY to Ethernet MAC, encoded by GMII_RX_DV and GMII_RX_ER during the IFG.	0	R
[3:2]	RGMII speed: Valid in RGMII mode configuration only. Link information from PHY to Ethernet MAC as encoded by GMII_RX_DV and GMII_RX_ER during the IFG. This 2-bit vector is defined with the following values: 10 = 1000 Mb/s 01 = 100 Mb/s 00 = 10 Mb/s 11 = N/A	All 0s	R
[29:4]	Reserved	–	
[31:30]	SGMII speed: Valid in SGMII mode configuration only. This displays the SGMII speed information, as received by TX_CONFIG_REG[11:10] in the PCS/PMA register. See Table 4-8, page 134. This 2-bit vector is defined with the following values: 10 = 1000 Mb/s 01 = 100 Mb/s 00 = 10 Mb/s 11 = N/A	All 0s	R

Table 3-14: **Management Configuration Register**

	MSB																												LSB
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x340	RESERVED																									MDIOEN	CLOCK_DIVIDE[5:0]					

Bit	Description	Default Value	R/W
[5:0]	Clock divide [5:0]: This value is used to derive the EMAC#PHYMCLKOUT for external devices. See "MDIO Interface," page 93.	All 0s	R/W
[6]	MDIO enable: When this bit is 1, the MDIO interface is used to access the PHY. When this bit is 0, the MDIO interface is disabled, and the MDIO signals remain inactive. See "MDIO Interface," page 93.	TIEEMAC#CONFIGVEC[73]	R/W
[31:7]	Reserved.	–	

Figure 3-40 shows the write timing for the configuration registers through the management interface. When accessing the configuration registers (i.e., when HOSTADDR[9] = 1 and HOSTMIIMSEL = 0), the upper bit of HOSTOPCODE functions as an active Low write-enable signal. The lower HOSTOPCODE bit (bit[0]) is a "don't care."

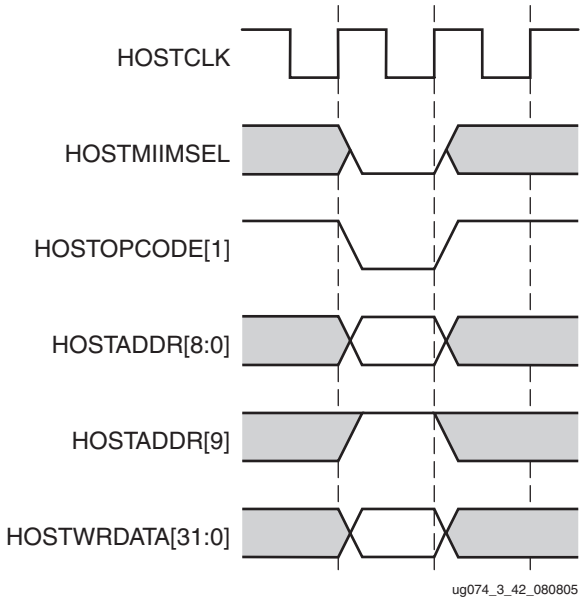


Figure 3-40: **Configuration Register Write Timing**

Figure 3-41 shows the read timing from the configuration registers. The words are similar, but the upper HOSTOPCODE bit = 1. The contents of the register appear on HOSTRDDATA[31:0] the HOSTCLK edge after the register address is asserted onto HOSTADDR. HOSTMIIMSEL acts as a read enable. It must be held Low for an even number of clock cycles during a read operation.

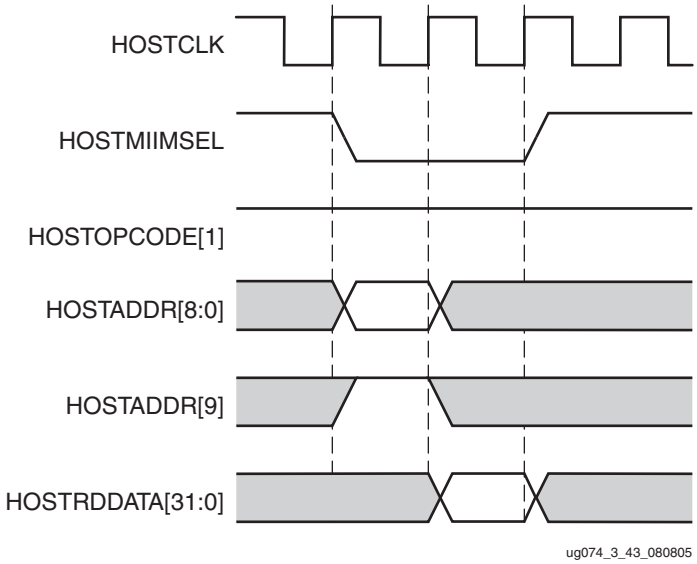


Figure 3-41: **Configuration Register Read Timing**

Address Filter Registers

Address Filter Register access includes the address filter registers and the multicast address table registers. The Ethernet MAC has five address filter registers with access through the host interface (Table 3-15).

Table 3-15: **Address Filter Register**

Address	Register Description
0x380	Unicast Address (Word 0)
0x384	Unicast Address (Word 1)
0x388	Multicast Address Table Access (Word 0)
0x38C	Multicast Address Table Access (Word 1)
0x390	Address Filter Mode

Figure 3-42 shows the multicast address table memory diagram.

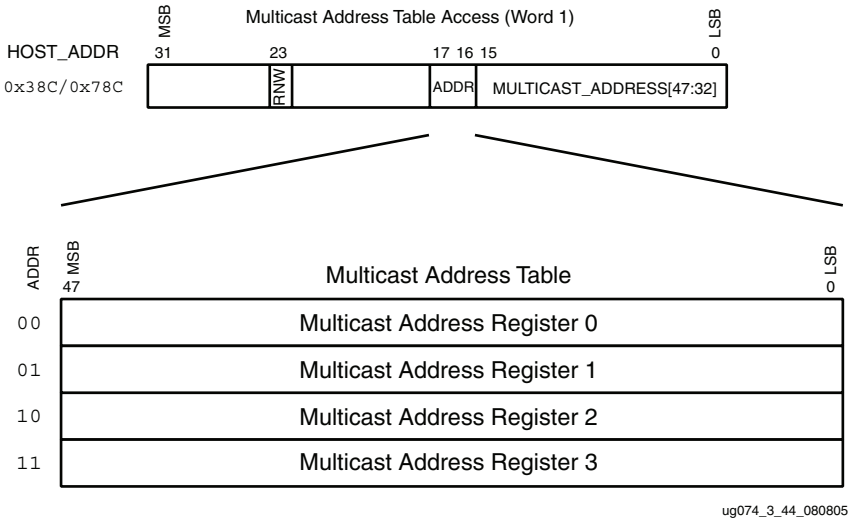


Figure 3-42: **Multicast Address Table Memory Diagram.**

The five address filter registers and the contents of the registers are shown in Table 3-16 through Table 3-20.

Table 3-16: **Unicast Address (Word 0)**

	MSB																												LSB
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x380	UNICAST_ADDRESS[31:0]																															

Bit	Description	Default Value	R/W
[31:0]	Unicast Address [31:0]. This address is used to match the Ethernet MAC against the destination address of any incoming frames.	TIEEMAC#UNICASTADDR[31:0]	R/W

Table 3-17: **Unicast Address (Word 1)**

	MSB																												LSB
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x384	RESERVED																UNICAST_ADDRESS[47:32]															

Bit	Description	Default Value	R/W
[15:0]	Unicast Address [47:32].	TIEEMAC#UNICASTADDR[47:32]	R/W
[31:16]	Reserved.	–	

Table 3-18: **Multicast Address Table Access (Word 0)**

	MSB																												LSB
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x388	MULTICAST_ADDRESS[31:0]																															

Bit	Description	Default Value	R/W
[31:0]	Multicast Address [31:0]. The multicast address bits [31:0] are temporarily deposited into this register for writing into a multicast address register.	All 0s	R/W

Table 3-19: **Multicast Address Table Access (Word 1)**

	MSB																												LSB
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x38c	RESERVED								RNW	RESERVED					ADDR		MULTICAST_ADDRESS[47:32]															

Bit	Description	Default Value	R/W
[15:0]	Multicast Address [47:32]. The multicast address bits [47:32] are temporarily deposited into this register for writing into a multicast address register.	All 0s	R/W
[17:16]	Multicast Address: This 2-bit vector is used to choose the multicast address register to access. 00 = Multicast Address Register 0 01 = Multicast Address Register 1 10 = Multicast Address Register 2 11 = Multicast Address Register 3	All 0s	R/W
[22:18]	Reserved.	–	
[23]	Multicast address read enable (RNW): When this bit is 1, a multicast address register is read. When this bit is 0, a multicast address register is written with the address set in the multicast address table register.	0	R/W
[31:24]	Reserved.	–	

Table 3-20: **Address Filter Mode**

	MSB																												LSB
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x390	PM	RESERVED																														

Bit	Description	Default Value	R/W
[30:0]	Reserved.	–	
[31]	Promiscuous Mode enable: When this bit is 1, the Address Filter block is disabled. When this bit is 0, the Address Filter block is enabled.	1: When TIEEMAC#CONFIGVEC[64] is set to 0 0: When TIEEMAC#CONFIGVEC[64] is set to 1	R/W

A timing diagram for writing to the Address Filter Registers is the same as the one shown for writing to the Ethernet MAC Configuration Registers (Figure 3-40).

Figure 3-43 shows the timing diagram for reading the multicast address from one of the four multicast address registers.

For reading a multicast address register in HOSTWRDATA[31:0], the RNW field is set to `1`, and the multicast address field should be set with the register number to be read. The multicast address register read data is returned in HOSTRDDATA[31:0]. The LSW is the multicast address [31:0]. The MSW contains `0x0000` and the multicast address [47:32]. For examples of accessing a multicast address register, see "Interfacing to the Processor DCR" in Chapter 6.

HOSTMIIMSEL acts as a read enable. It must be held Low for an even number of clock cycles during a read operation.

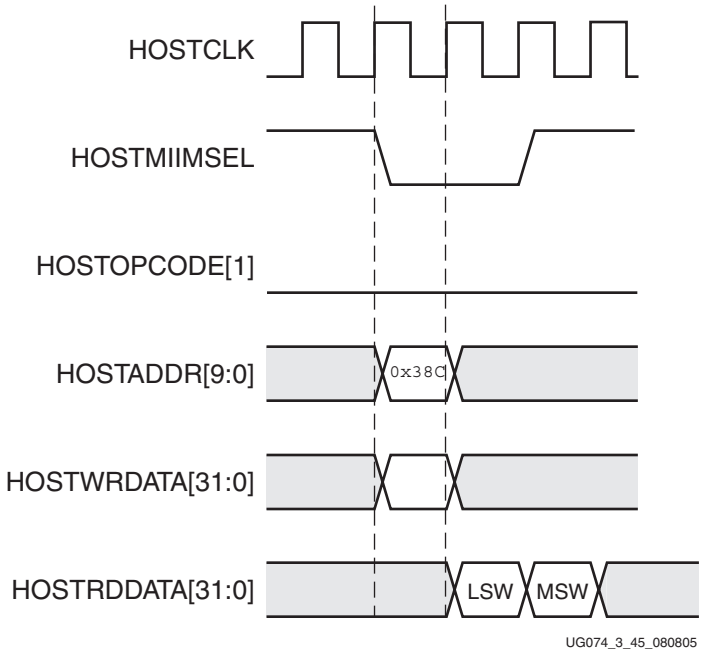


Figure 3-43: **Address Filter Multicast Address Register Read**

Using the DCR Bus as the Host Bus

When the DCR bus is used to access the internal registers of the Ethernet MAC, the DCR bus bridge in the host interface translates commands carried over the DCR bus into Ethernet MAC host bus signals. These signals are then input into one of the Ethernet MACs.

The DCR bus bridge contains four device control registers. The first two are used as data registers, each is 32 bits wide (dataRegMSW and dataRegLSW). The third is used as a control register (cntlReg).

The fourth device control register is used as a ready status register (RDYstatus). The PowerPC 405 processor polls this register to determine access completion status. The bits in this register are asserted when there is no access in progress. When an access is in progress, a bit corresponding to the type of access is automatically deasserted. The bit is automatically reasserted when the access is complete. Alternatively, the host interface can also provide an interrupt request to inform the host of access completion. The user can select either the polling or the interrupt method to inform the host of access status.

The DCR bridge ignores any new DCR command for host access until the current host access is complete. Therefore, it is essential to determine when the host access is complete before issuing a new DCR command.

Table 3-21 shows the DCR addresses for the DCRs. The user assigns the DCR address bits [9:2] in the DCR address space. UG018, *PowerPC 405 Processor Block Reference Guide* describes the DCR operation.

Table 3-21: **Ethernet MAC Host Interface Device Control Register Addresses**

DCR Address	DCR Name	Register Width	R/W
`**_****_1100`	dataRegMSW	32 bits	R/W
`**_****_1101`	dataRegLSW	32 bits	R/W
`**_****_1110`	cntlReg	32 bits	R/W
`**_****_1111`	RDYstatus	32 bits	R(1)

Notes:

1. This register is Read Only.

The four registers and the contents of the registers are shown in Table 3-22 through Table 3-25. DCR registers are big endian.

Table 3-22: **DCR Data Register dataRegMSW**

DCR Offset	MSB																													LSB
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0xC	dataRegMSW																															

Bit	Description	Default Value
[31:0]	Data – Data input from the DCR bus for the Ethernet MAC registers is written into this register, and the most significant word of data is read out from the Ethernet MAC registers and deposited into this register.	Undefined

Table 3-23: **DCR Data Register dataRegLSW**

DCR Offset	MSB																													LSB
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0xD	dataRegLSW																															

Bit	Description	Default Value
[0:31]	Data – Data input from the DCR bus for the Ethernet MAC registers is written into this register, and the least significant word of data is read out from the Ethernet MAC registers and deposited into this register.	Undefined

Table 3-24: **DCR Control Register cntlReg**

DCR Offset	MSB																													LSB
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0xE	RESERVED																WEN	RESERVED				EMAC1	ADDRESS_CODE								

Bit	Description	Default Value
[0:15]	Reserved.	All 0s
[16]	Write Enable – When this bit is asserted, the data in either dataRegLSW or dataRegMSW is written into an Ethernet MAC register. When this bit is deasserted, the operation to be performed is read.	0
[17:20]	Reserved.	All 0s
[21]	EMAC1SEL – When this bit is asserted, the address code is for the EMAC1 registers. Otherwise, the address code is for the EMAC0 registers. This bit is essentially the bit [10] of the address code.	0
[22:31]	Address Code – the DCR bus bridge translates this address code into the Ethernet MAC register address. See Table 3-30, page 91 for address code.	All 0s

Table 3-25: **DCR Ready Status Register RDYStatus (Read Only)**

DCR Offset	MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
0xF	RESERVED																	CFG WR1	CFG RR1	AF WR1	AF RR1	MIIM WR1	MIIM RR1	STAT RR1	RSVD	CFG WR0	CFG RR0	AF WR0	AF RR0	MIIM WR0	MIIM RR0	STAT RR0

Bit	Description	Default Value
[0:16]	Reserved.	All 0s
[24]	Reserved.	0
EMAC1 Only:		
[17]	Configuration Write Ready bit.	1
[18]	Configuration Read Ready bit.	1
[19]	Address Filter Write Ready bit.	1
[20]	Address Filter Read Ready bit.	1
[21]	MDIO Write Ready bit.	1
[22]	MDIO Read Ready bit.	1
[23]	Statistics IP Read Ready bit.(1)	1
EMAC0 Only:		
[25]	Configuration Write Ready bit.	1
[26]	Configuration Read Ready bit.	1
[27]	Address Filter Write Ready bit.	1
[28]	Address Filter Read Ready bit.	1
[29]	MDIO Write Ready bit.	1
[30]	MDIO Read Ready bit.	1
[31]	Statistics IP Read Ready bit.(1)	1

Notes:

1. For more information on Statistics IP, see "Interfacing to an FPGA Fabric-Based Statistics Block" in Chapter 6.

In addition to the DCRs, the DCR bus bridge contains three registers. These registers are accessed indirectly through the DCRs.

When the interrupt method is selected to inform the host of access completion status, the IRSTATUS register contains the interrupt request when an access is completed. When the host services the interrupt, it reads this register to determine the type of host access completed. Before exiting the interrupt service routine, the host writes to this register to clear the interrupt request bit.

The IRENABLE register is programmed to allow updating of the interrupt request in the IRSTATUS register. When an enable bit is cleared, the corresponding bit in the IRSTATUS register is not updated. The MIIMWRDATA register temporarily holds MDIO write data for output to the MDIO write data bus. In the case of an MDIO read, there is no need to

program the MIIMWRDATA register. Writing to the address of the MIIMCNTL register starts the MDIO read or write transaction using the physical and register address in the DCR dataRegLSW register.

The host interface interrupt request registers (IRENABLE and IRSTATUS) and the contents of the registers are shown in Table 3-26 and Table 3-27. The MIIMWRDATA register is shown in Table 3-28.

Table 3-26: **Interrupt Status Register IRSTATUS**

Address Code	MSB																													LSB
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0x3A0	RESERVED																	CFG WST1	CFG RST1	AF WST1	AF RST1	MIIM WST1	MIIM RST1	STAT RST1	RSVD	CFG WST0	CFG RST0	AF WST0	AF RST0	MIIM WST0	MIIM RST0	STAT RST0

Bit	Description	Ethernet MAC	Default Value
[0:16]	Reserved.	–	0
[17]	Configuration Write Interrupt Request bit.	EMAC1	0
[18]	Configuration Read Interrupt Request bit.	EMAC1	0
[19]	Address Filter Write Interrupt Request bit.	EMAC1	0
[20]	Address Filter Read Interrupt Request bit.	EMAC1	0
[21]	MDIO Write Interrupt Request bit.	EMAC1	0
[22]	MDIO Read Interrupt Request bit.	EMAC1	0
[23]	Statistics IP Read Interrupt Request bit.(1)	EMAC1	0
[24]	Reserved.	–	0
[25]	Configuration Write Interrupt Request bit.	EMAC0	0
[26]	Configuration Read Interrupt Request bit.	EMAC0	0
[27]	Address Filter Write Interrupt Request bit.	EMAC0	0
[28]	Address Filter Read Interrupt Request bit.	EMAC0	0
[29]	MDIO Write Interrupt Request bit.	EMAC0	0
[30]	MDIO Read Interrupt Request bit.	EMAC0	0
[31]	Statistics IP Read Interrupt Request bit.(1)	EMAC0	0

Notes:

1. For more information on Statistics IP, see "Interfacing to an FPGA Fabric-Based Statistics Block" in Chapter 6.

Table 3-27: **Interrupt Enable Register IRENABLE**

Address Code	MSB																												LSB
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0x3A4	RESERVED																	CFG WEN1	CFG REN1	AF WEN1	AF REN1	MIIM WEN1	MIIM REN1	STAT REN1	RSVD	CFG WEN0	CFG REN0	AF WEN0	AF REN0	MIIM WEN0	MIIM REN0	STAT REN0

Bit	Description	Ethernet MAC	Default Value
[0:16]	Reserved.	–	0
[17]	Configuration Write IR-enable bit.	EMAC1	0
[18]	Configuration Read IR-enable bit.	EMAC1	0
[19]	Address Filter Write IR-enable bit.	EMAC1	0
[20]	Address Filter Read IR-enable bit.	EMAC1	0
[21]	MDIO Write IR-enable bit.	EMAC1	0
[22]	MDIO Read IR-enable bit.	EMAC1	0
[23]	Statistics IP Read IR-enable bit.(1)	EMAC1	0
[24]	Reserved	–	0
[25]	Configuration Write IR-enable bit.	EMAC0	0
[26]	Configuration Read IR-enable bit.	EMAC0	0
[27]	Address Filter Write IR-enable bit.	EMAC0	0
[28]	Address Filter Read IR-enable bit.	EMAC0	0
[29]	MDIO Write IR-enable bit.	EMAC0	0
[30]	MDIO Read IR-enable bit.	EMAC0	0
[31]	Statistics IP Read IR enable bit.(1)	EMAC0	0

Notes:

1. For more information on Statistics IP, see "Interfacing to an FPGA Fabric-Based Statistics Block" in Chapter 6.

Table 3-28: **Host Interface MDIO Write Data Register (MIIMWRDATA)**

Address Code	MSB																												LSB
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0x3B0	RESERVED																MIIMWRDATA															

Bit	Description	Default Value
[0:15]	Reserved.	All 0s
[16:31]	Data- temporarily holds MDIO write data for output onto the host write data bus.	undefined

Notes:

1. See "Interfacing to an FPGA Fabric-Based Statistics Block" in Chapter 6.

Description of Ethernet MAC Register Access through the DCR Bus

To write data to an Ethernet MAC register through the DCR bus, the host processor must first write the data into the DCR dataRegLSW. The host processor then writes the EMAC0 or EMAC1 select bit, the write control bit, and the Ethernet MAC register address code into the DCRcntlReg. The Ethernet MAC register address code in the cntlReg, ADDRESS_CODE (Table 3-24), is translated into the corresponding Ethernet MAC register address in the Ethernet MAC, and the address is output on the address bus HOSTADDR#[9:0]. See Figure 3-39, page 73. The mapping of the ADDRESS_CODE field to the set of Ethernet MAC registers is also shown in Figure 3-44.

The DCR registers (dataRegMSW, dataRegLSW, cntlReg, and RDYstatus) and the DCR bridge registers (IRSTATUS, IRENABLE, and MIIMWRDATA) use the big endian bit numbering convention. However, the Ethernet MAC host registers, such as Receiver Configuration (Word 0) (host address `0x200`) register, use the little endian bit numbering convention. In the DCR bridge implementation, there is no conversion to or from big endian to little endian. The bit positions are mapped directly in a one-to-one correspondence, (big endian bit [0] is mapped directly to little endian bit [31], big endian bit [1] is mapped directly to little endian bit [30] and onward).

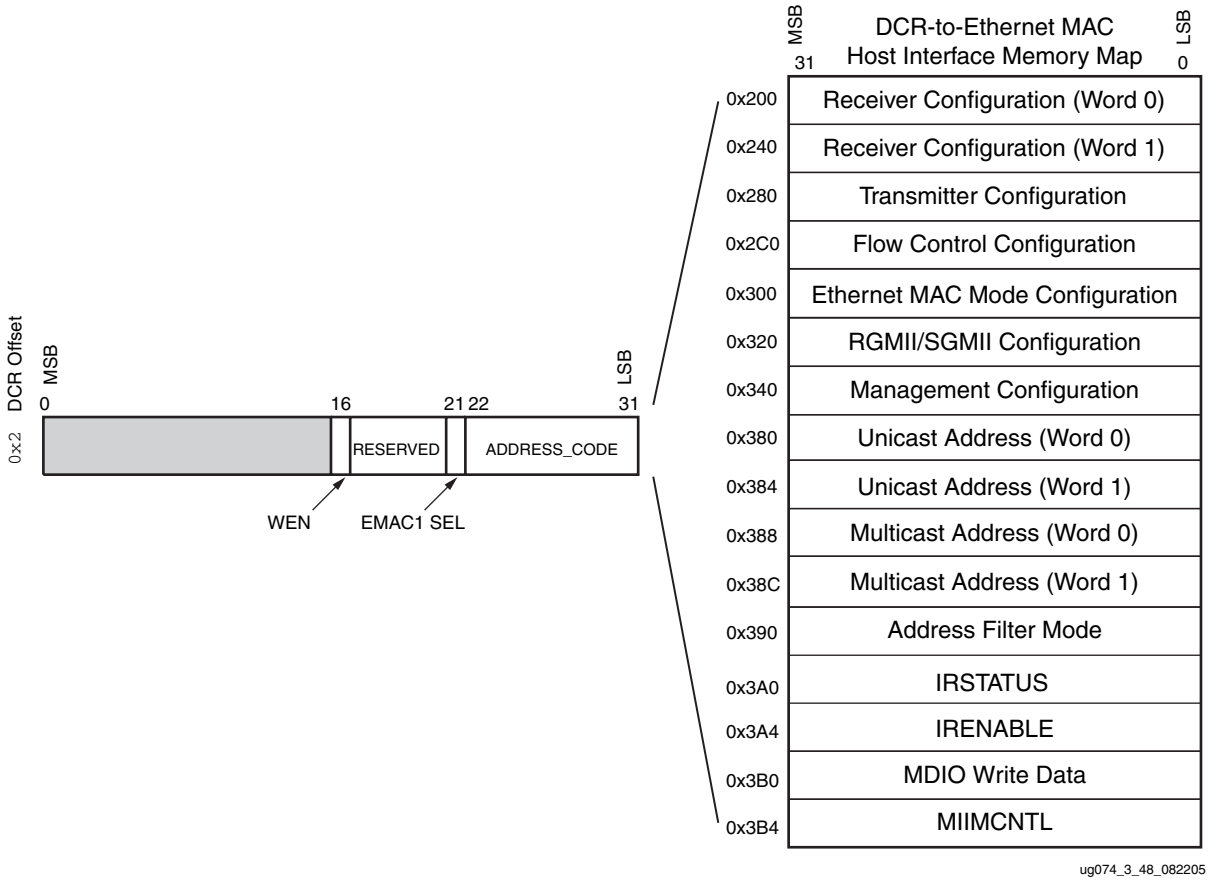


Figure 3-44: **DCR to Ethernet MAC Host Interface Memory Map**

The decode of the address code also generates the control signals MIIMSEL#, REQ#, and OPCODE#[1:0] (see Figure 3-39, page 73). Data in the dataRegLSW is output on the WRD#[31:0]. These signals are output to EMAC0 or EMAC1 when selected by the

EMACISEL bit. All writes to Ethernet MAC registers are accomplished in a single host clock cycle except for the MDIO registers.

To read data from an Ethernet MAC register through the DCR bus, the DCR cntlReg is programmed for read, EMAC0 or EMAC1 select, and the address code. The Ethernet MAC address code is translated and output from the host interface on the address bus ADDR#[9:0].

The decode of the address code generates the control signals MIIMSEL#, REQ#, and OPCODE#[1:0] that are output to the selected Ethernet MAC. The data read out from the Ethernet MAC is deposited in DCR dataRegLSW and dataRegMSW (in the case of an address filter or statistics IP register read) in the host interface.

Reading the configuration registers for the Ethernet MAC and the address filter registers takes a single host clock cycle, while reading from the Ethernet MAC statistics IP registers and MDIO registers takes multiple host clock cycles. An Ethernet MAC statistics IP register read takes six host clock cycles. MDIO registers reads take a multiple number of host clock cycles depending on the physical interface device. To write to any of the PCS layer registers ("Management Registers," page 140), the data must be written to the MDIO Write Data register shown in Figure 3-44. The PHY address and PCS register number are then written to the DCR dataRegLSW register. The mapping is shown in Figure 3-45.

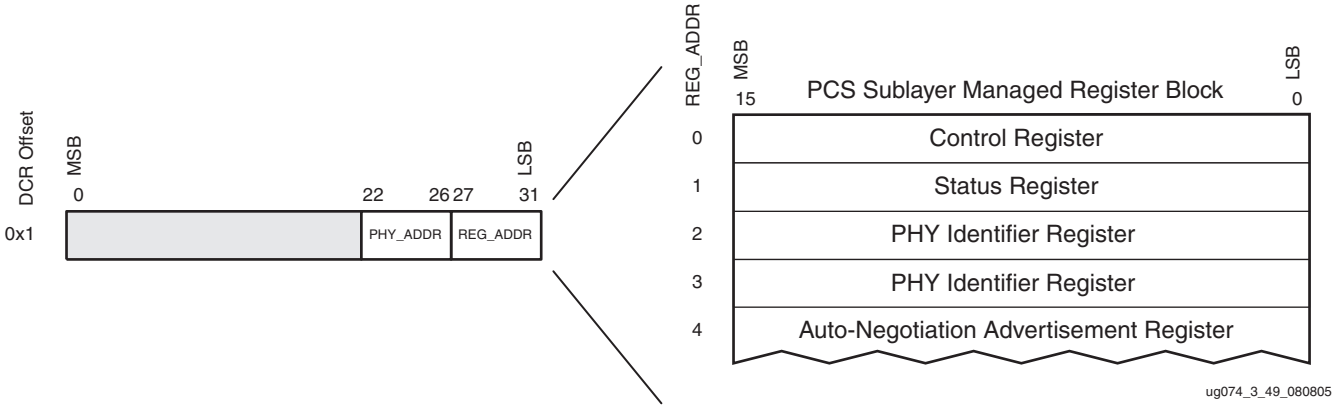


Figure 3-45: **MDIO Address Register to Access PCS Sublayer Register Block**

The DCR bridge runs at the same clock frequency as the PowerPC processor. Because the host bus is not a high performance bus, HOSTCLK runs at a lower frequency. The HOSTCLK frequency must be an integer divide of the DCR clock frequency, and the two clocks must be phase aligned. The DCR bridge ignores any new DCR command in the DCR clock domain until a host access in the HOSTCLK domain is complete. Hence, the PowerPC processor must determine when a host access is complete.

If the interrupt request method is selected, the host interface interrupt request output pin DCRHOSTDONEIR is used to notify the host when an access is completed. In the case of a read, when the host services the interrupt, it must issue DCR reads to dataRegLSW and dataRegMSW to read out the Ethernet MAC register data.

The interrupt request register is located in the IRSTATUS register (Address Code `0x3A0`). After servicing the interrupt request, the host must clear the interrupt request. In addition, the DCR RDYstatus register is provided to indicate when a multiple-cycle access is ready. This register is allows the host to use the polling method for accesses requiring only a few multiple host clock cycles.

The IRENABLE register (Address Code 0x3A4) in the host interface is used to enable interrupt bits in the IRSTATUS register. To enable an interrupt, the corresponding bit is set. When the enable bit is cleared, the interrupt status is not updated.

For examples of DCR read and write accesses, see "Interfacing to the Processor DCR" in Chapter 6.

Address Code

The address codes for the Ethernet MAC registers are divided into three groups as shown in Table 3-29. The unused address codes are reserved. The detailed address codes for each register are described in Table 3-30. The address codes for the Ethernet MAC registers and registers in the host interface are encoded in hardware. Address codes for statistics IP registers and Ethernet MAC Configuration registers match the 1G Ethernet MAC Host Bus address as specified in the Xilinx® 1G Ethernet MAC core at: http://www.xilinx.com/support/documentation/ip_documentation/gig_eth_mac_ds200.pdf.

Table 3-29: **Address Code Groups for DCR Host Bus Access**

Group	Address Code	Description
EMAC0	0x200 – 0x39F	EMAC0 registers.
Host Interface	0x3A0 – 0x3FF	Host interface registers.
EMAC1	0x600 – 0x79F	EMAC1 registers.

Notes:

1. Any access to the host interface registers does not generate interrupts and does not change the RDYSTATUS register bits.

Table 3-30: **Detailed Address Codes for DCR Host Bus Access**

Address Codes	Register Names	Description	Ethernet MAC Register Address	R/W
0x0:0x1FF		Reserved.		
EMAC0 Registers:				
0x200	E0_RXCONFIGW0	Receiver configuration word 0.	0x200	R/W
0x240	E0_RXCONFIGW1	Receiver configuration word 1.	0x240	R/W
0x280	E0_TXCONFIG	Transmitter configuration.	0x280	R/W
0x2C0	E0_FLOWCONTROL	Flow control configuration.	0x2C0	R/W
0x300	E0_EMACCONFIG	Ethernet MAC configuration.	0x300	R/W
0x320	E0_RGMII_SGMII	RGMII/SGMII configuration.	0x320	R
0x340	E0_MGMTCONFIG	Management configuration.	0x340	R/W
0x380	E0_UNICASTADDRW0	Unicast address [31:0].	0x380	R/W
0x384	E0_UNICASTADDRW1	0x0000, unicast address [47:32].	0x384	R/W
0x388	E0_ADDRTABLECONFIGW0	Multicast address data [31:0]	0x388	R/W
0x38C	E0_ADDRTABLECONFIGW1	0x00, RNW, 00000, ADDR[1:0], Multicast address data [47:32]	0x38C	R/W

Table 3-30: **Detailed Address Codes for DCR Host Bus Access *(Cont'd)***

Address Codes	Register Names	Description	Ethernet MAC Register Address	R/W
0x390	E0_GENERALCONFIG	Promiscuous mode, 0x00000000 bits [31:0].	0x390	R/W
0x394:0x39F		Reserved.	–	-
0x3A0	IRSTATUS	Access done, interrupt request status.	0x3A0	R/W
0x3A4	IRENABLE	Interrupt request enable.	0x3A4	R/W
0x3A8:0x3AF		Reserved.		
0x3B0	MIIMWRDATA	MDIO write data.	0x3B0	R/W
0x3B4	MIIMCNTL	Decode address for MDIO address output.	0x3B4	W
0x3B8:0x5FF		Reserved.	–	-
EMAC1 Registers				
0x600	E1_RXCONFIGW0	Receiver configuration word 0.	0x600	R/W
0x640	E1_RXCONFIGW1	Receiver configuration word 1.	0x640	R/W
0x680	E1_TXCONFIG	Transmitter configuration.	0x680	R/W
0x6C0	E1_FLOWCONTROL	Flow control configuration.	0x6C0	R/W
0x700	E1_EMACCONFIG	Ethernet MAC configuration.	0x700	R/W
0x720	E1_RGMII_SGMII	RGMII/SGMII configuration.	0x720	R
0x740	E1_MGMTCONFIG	Management configuration.	0x740	R/W
0x780	E1_UNICASTADDRW0	Unicast address [31:0].	0x780	R/W
0x784	E1_UNICASTADDRW1	0x0000, Unicast Address [47:32].	0x784	R/W
0x788	E1_ADDRTABLECONFIGW0	Multicast address data[31:0]	0x788	R/W
0x78C	E1_ADDRTABLECONFIGW1	0x00, RNW, 00000, ADDR[1:0], Multicast address data[47:32].	0x78C	R/W
0x790	E1_GENERALCONFIG	Promiscuous mode, 0x00000000 bits [31:0].	0x790	R/W
0x7A0	IRSTATUS	Access done, interrupt request status.	0x7A0	R/W
0x7A4	IRENABLE	Interrupt request enable.	0x7A4	R/W
0x7A8:0x7AF		Reserved.		
0x7B0	MIIMWRDATA	MDIO write data.	0x7B0	R/W
0x7B4	MIIMCNTL	Decode address for MDIO address output.	0x7B4	W
0x7B8:0x7FF		Reserved.	–	-

MDIO Interface

Introduction to MDIO

The MDIO interface for 1 Gb/s operation (and slower speeds) is defined in IEEE Std 802.3, Clause 22. This two-wire interface consists of a clock (MDC) and a shared serial data line (MDIO).

An MDIO bus in a system consists of a single Station Management (STA) master management entity and a number of MDIO Managed Device (MMD) slave entities. Figure 3-46 illustrates a typical system. All transactions, read or write, are initiated by the STA entity. All MMD devices, if addressed, must respond to the transactions from the STA.

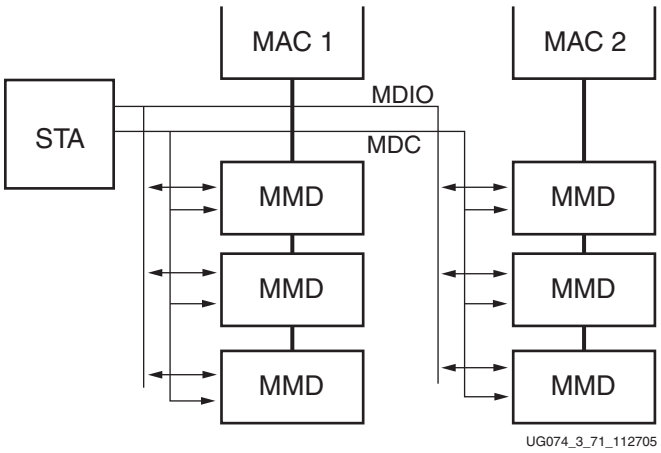


Figure 3-46: **Typical MDIO-Managed System**

The two different MDIO transaction types for writes and reads are described in "Write Transaction," and "Read Transaction."

These abbreviations are used in this chapter:

OP	Operation code
PHYAD	PHY address
PRE	Preamble
REGAD	Register address
ST	Start of frame
TA	Turnaround

Write Transaction

Figure 3-47 shows a Write transaction across the MDIO, as defined by OP = 0b01. The addressed MMD (PHYAD) device takes the 16-bit word in the Data field and writes it to the register at REGAD.

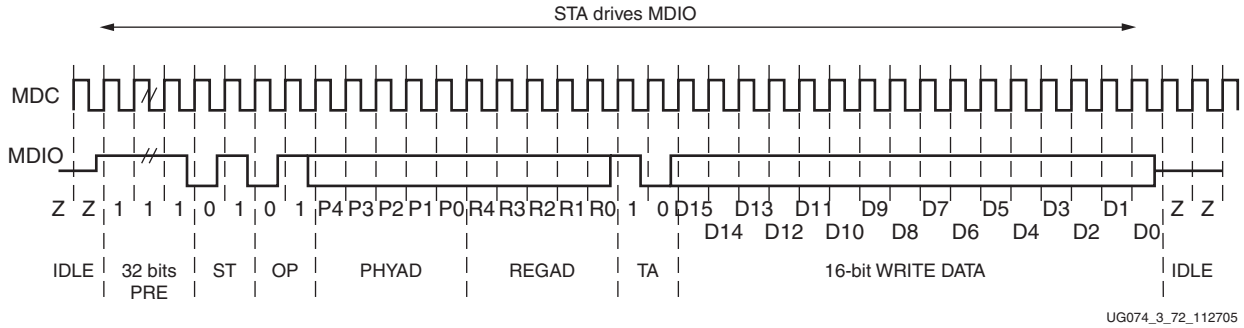


Figure 3-47: **MDIO Write Transaction**

Read Transaction

Figure 3-48 shows a Read transaction as defined by OP = 0b10. The addressed MMD (PHYAD) device returns the 16-bit word from the register at REGAD.

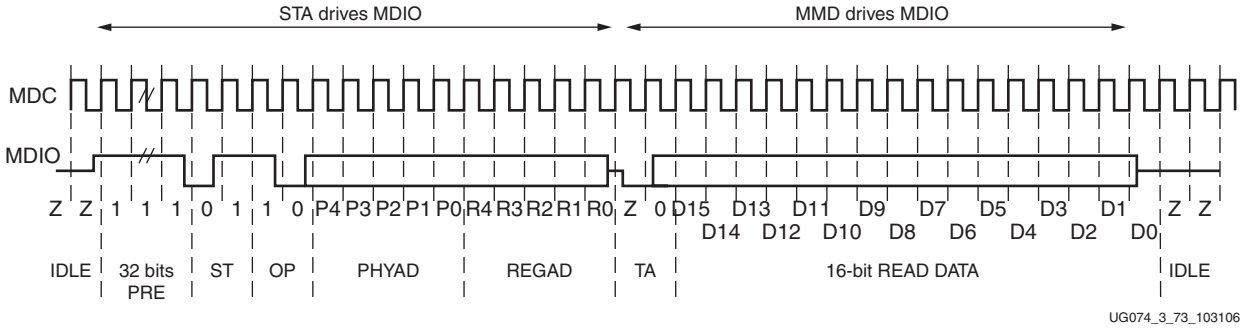


Figure 3-48: **MDIO Read Transaction**

The IEEE specification 802.3-2002 provides details of the register map of MMD (PHY layer devices) and a fuller description of the operation of the MDIO interface.

Special Note on the Physical Addresses

The PHYAD field for the MDIO frame is defined in IEEE Std 802.3, Clause 22.2.4.5.5. This address field is a 5-bit binary value capable of addressing 32 unique addresses. However, every MMD must respond to address 0. Therefore, this address location can be used to write a single command that is obeyed by all attached MMDs such as a reset or power-down command.

This requirement dictates that the PHYAD for any particular MMD must not be set to 0 to avoid possible MDIO contention.

MDIO Implementation in the EMAC

The EMAC implements an STA (MDIO master), controlled through the host interface, which can be connected to one or more MMDs (PHY devices) to access their management registers.

The PCS/PMA sublayer of the EMAC, used for 1000BASE-X or SGMII, also contains an MMD (MDIO slave). The physical address of this MMD is set via the PHYEMAC#PHYAD[4:0] port of the EMAC. However, the PCS/PMA sublayer also responds to a PHYAD of zero.

Example Use Models

Figure 3-49 illustrates a user case example, where the Host Interface is used as the MDIO master to access the configuration registers of an external PHY.

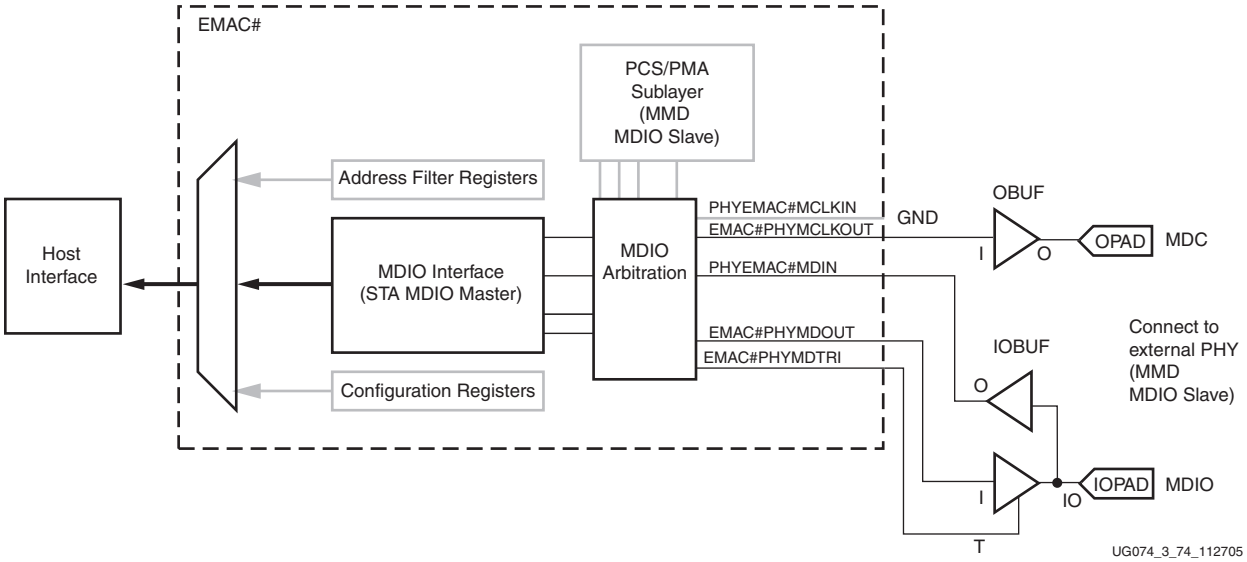


Figure 3-49: **User Case 1: MDIO Access to External PHY**

PHYEMAC#MDIN, EMAC#PHYMDOUT, and EMAC#PHYMDTRI must be connected to a 3-state buffer to create the bidirectional wire, MDIO. This 3-state buffer can be either external to the FPGA or internally integrated by using an IOBUF with an appropriate I/O standard for the external PHY as illustrated in Figure 3-49.

To obtain this functionality whenever the host interface is used, the EMAC's Management Data Input/Output (MDIO) Interface signals are wired as shown in Figure 3-49 with TIEEMAC#CONFIGVEC[73] (MDIO enable) tied High.

This example intentionally does not use the PCS/PMA sublayer (a GMII or RGMII physical interface can be selected, or the PCS/PMA sublayer can be configured only through its tie-off vectors, TIEEMAC#CONFIGVEC[78:74]). However, it is still internally connected to the MDIO and replies to a read or write transaction, if addressed. The PHYAD of the PCS/PMA sublayer must not be addressed.

Figure 3-50 illustrates a second implementation example. The host interface is used as the MDIO master to access the configuration registers of the PCS/PMA sublayer logic, which contains an MDIO slave. All connections are internal and are enabled by pulling TIEEMAC#CONFIGVEC[73] (MDIO enable) High.

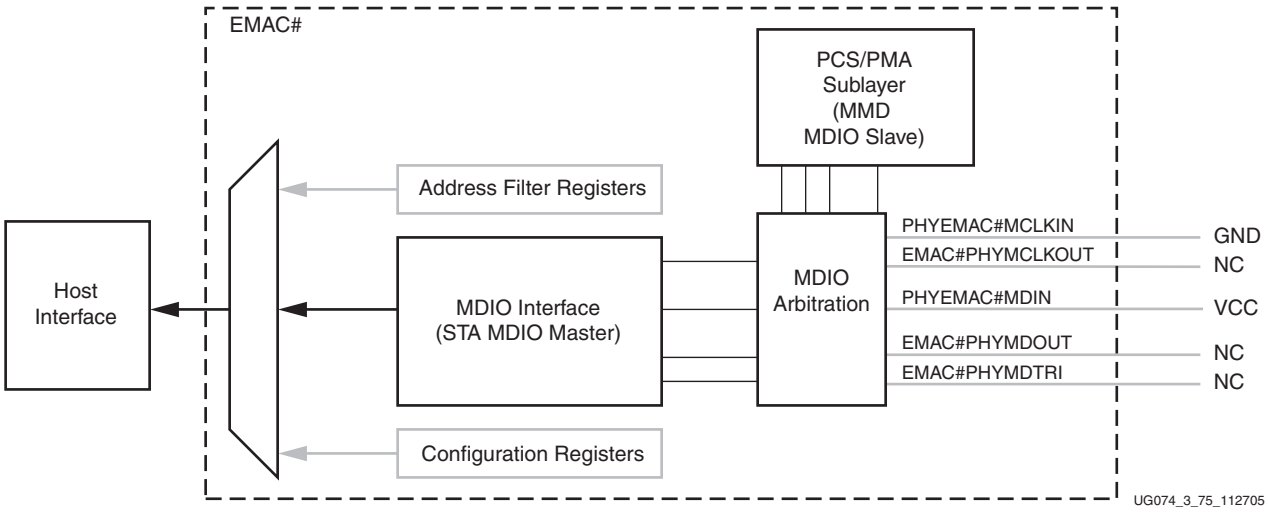


Figure 3-50: **User Case 2: Internal MDIO Access to PCS/PMA Sublayer**

In this example, the EMAC's Managements Data Input/Output (MDIO) Interface signals are not used. The output signals are left unconnected, and the input signals are tied to a logic level. PHYEMAC#MDIN must be tied High when not connected to an external PHY.

Alternatively, the EMAC's Managements Data Input/Output (MDIO) can be connected to a second MMD (for example, an external PHY device) by providing the connections illustrated in Figure 3-49. Externally connected MMDs (MDIO slaves) must have different non-zero physical addresses (PHYAD) from the non-zero address of the PCS/PMA sublayer.

Figure 3-51 illustrates a third user case example that does not use the host interface, but instead uses an external STA (MDIO master). Figure 3-51 shows this as an external device to the FPGA, but the EMAC's Managements Data Input/Output (MDIO) Interface signals can alternatively be connected directly to a STA implemented in the FPGA fabric.

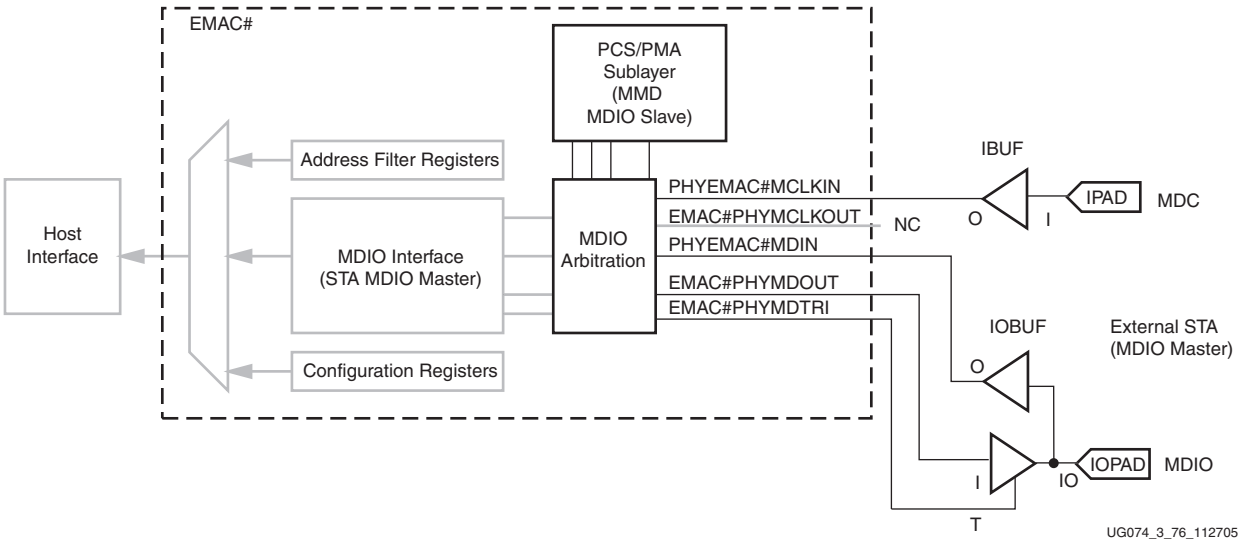


Figure 3-51: **User Case 3: External MDIO Access to the PCS/PMA Sublayer**

This functionality is obtained by asserting High TIEEMAC#CONFIGVEC[73] (MDIO enable) and pulling Low TIEEMAC#CONFIGVEC[67] (Host Interface enable) when not using the host interface. In this case, the MDC clock must be provided through the input port PHYEMAC#MCKIN.

Accessing MDIO via the EMAC Host Interface

The host interface can be used to provide STA (MDIO master) functionality. The remainder of this chapter details how to access this functionality via the host interface.

The MDIO interface supplies a clock to the external devices, EMAC#PHYMCLKOUT when the host interface is enabled. This clock is derived from the HOSTCLK signal using the value in the Clock Divide[5:0] configuration register. The frequency of the MDIO clock is given by the following equation:

$$f_{MDC} = \frac{f_{HOSTCLK}}{(1 + \text{Clock Divide}[5:0]) \times 2}$$

To comply with the IEEE Std 802.3-2002 specification for this interface, the frequency of EMAC#PHYMCLKOUT should not exceed 2.5 MHz. To prevent EMAC#PHYMCLKOUT from being out of specification, the Clock Divide[5:0] value powers up at `000000`. While this value is in the register, it is impossible to enable the MDIO interface. Given this, even if the user has enabled the host interface and the MDIO interface by tieing both TIEEMAC#CONFIGVEC[67] and TIEEMAC#CONFIGVEC[73] High. Upon reset, the MDIO port is still disabled until a non-zero value has been written into the clock divide bits. When the host interface is disabled, the user can still access the management registers in the internal PCS/PMA layer by providing PHYEMAC#MCLKIN and tying TIEEMAC#CONFIGVEC[73] High.

Access to the MDIO interface through the management interface is shown in the Figure 3-52 timing diagram.

In MDIO transactions, the following applies:

- HOSTOPCODE maps to the OPCODE field of the MDIO frame.
- HOSTADDR maps to the two address fields of the MDIO frame; PHY_ADDR is HOSTADDR[9:5], and REG_ADDR is HOSTADDR[4:0].
- HOSTWRDATA[15:0] maps into the data field of the MDIO frame during a write operation.
- The data field of the MDIO frame maps into HOSTRDDATA[15:0] during a read operation.

The Ethernet MAC signals to the host that it is ready for an MDIO transaction by asserting HOSTMIIMRDY. A read or write transaction on the MDIO is initiated by a pulse on the HOSTREQ signal. This pulse is ignored if the MDIO interface already has a transaction in progress. The Ethernet MAC deasserts the HOSTMIIMRDY signal while the transaction across the MDIO is in progress. When the transaction across the MDIO interface is completed, the HOSTMIIMRDY signal is asserted by the Ethernet MAC. If the transaction is a read, the data is also available on the HOSTRDATA[15:0] bus.

As noted in Figure 3-52, if a read transaction is initiated, the HOSTRDDATA bus is valid at the point indicated. If a write transaction is initiated, the HOSTWRDATA bus must be valid at the point indicated. Simultaneous read and write is not permitted.

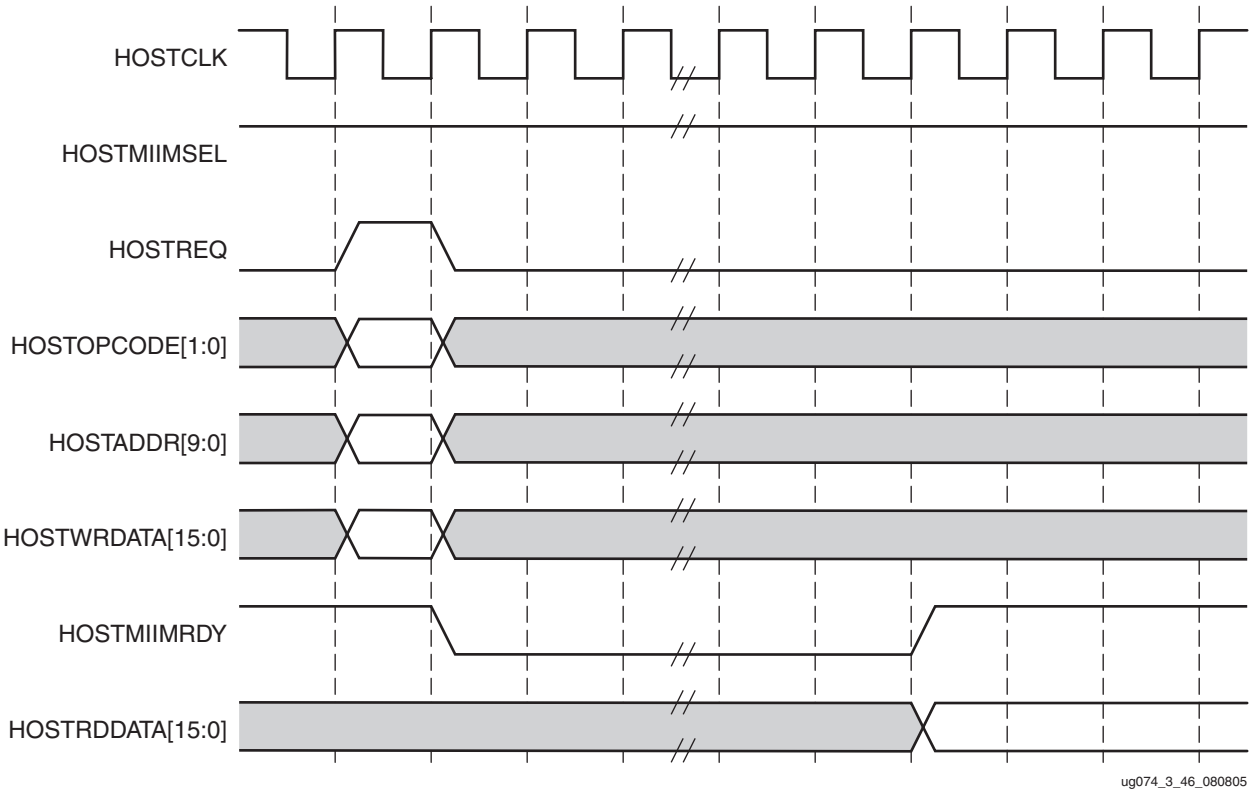


Figure 3-52: **MDIO Access Through the Management Interface**

For register map details of the physical layer devices and a complete description of the operation of the MDIO interface itself, see IEEE specification 802.3-2002.

XILINX®

Chapter 4

Physical Interface

The following sections describe the design considerations when using the Ethernet MAC for the supported interfaces. The wrapper for the different physical interfaces are provided in the Xilinx® CORE Generator™ tool. The interfaces are available in both VHDL and Verilog. The wrapper files created by the CORE Generator tool will contain the clocking logic. By using the CORE Generator tool, the time required to instantiate the Ethernet MAC into a usable design is greatly reduced. See "Using the Embedded Ethernet MAC," page 167. The # in the following sections denotes either EMAC0 or EMAC1.

Media Independent Interface (MII)

MII is designed to IEEE Std 802.3-2002, Clause 22. It is used for 10 Mb/s and 100 Mb/s.

MII Interface

Figure 4-1 shows the Ethernet MAC configured with MII as the physical interface. In this interface, not all the ports of the Ethernet MAC are used.

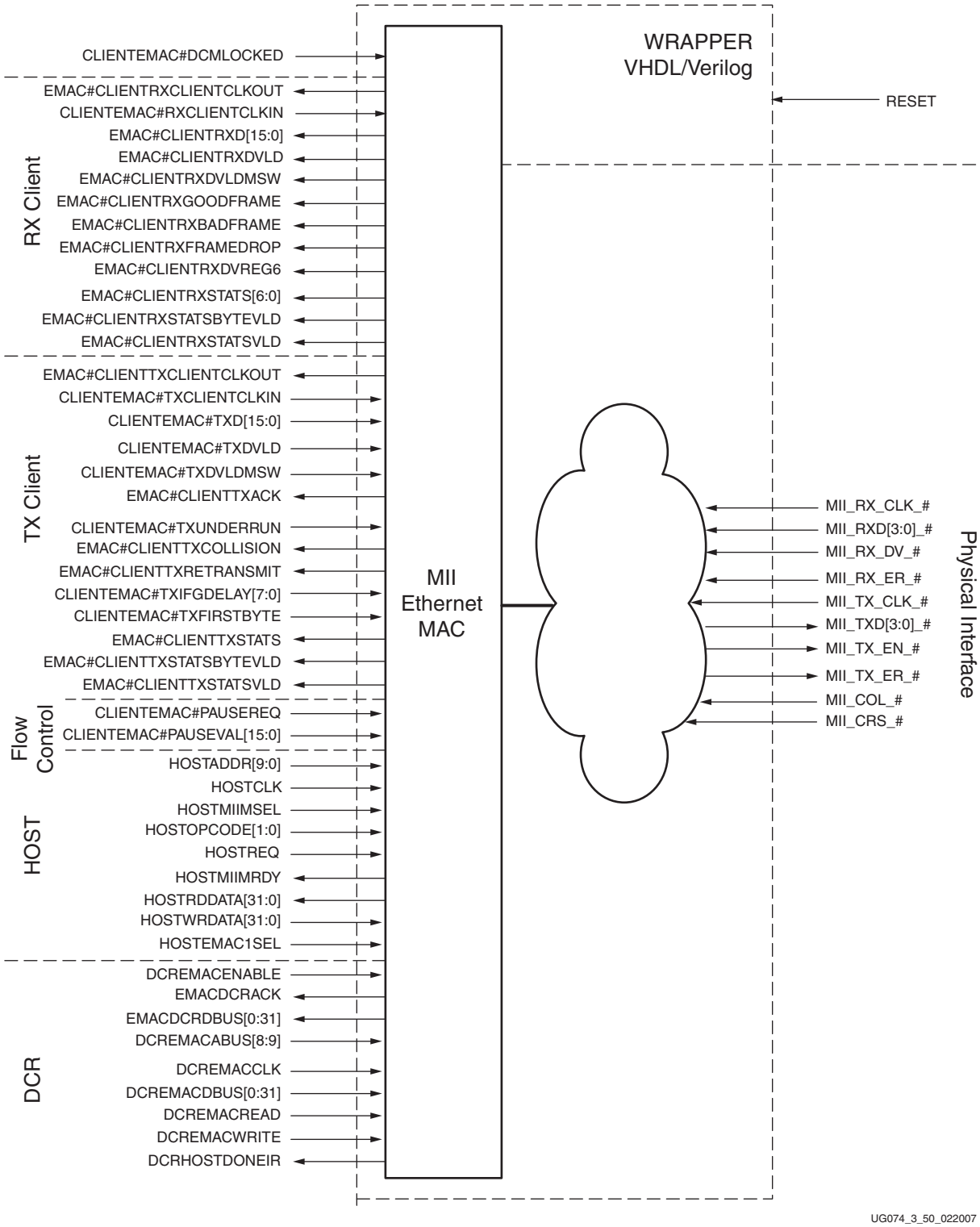


Figure 4-1: **Ethernet MAC Configured in MII Mode**

MII Clock Management

Figure 4-2 shows the clock management used with the MII interface. Both the MII_TX_CLK_# and MII_RX_CLK_#, generated from the PHY, have a frequency of either 2.5 MHz or 25 MHz, depending on the operating speed of the Ethernet MACs. The MII_TX_CLK drives the MII_TXD registers, the CLIENTEMAC#TXGMIIMIICLKIN and the PHYEMAC#MIITXCLK through a BUFG. It has a frequency of 12.5 MHz or 1.25 MHz depending on the operating speed of the Ethernet MAC. The RX clocking is similar.

The CLIENTEMAC#DCMLOCKED port must be tied High.

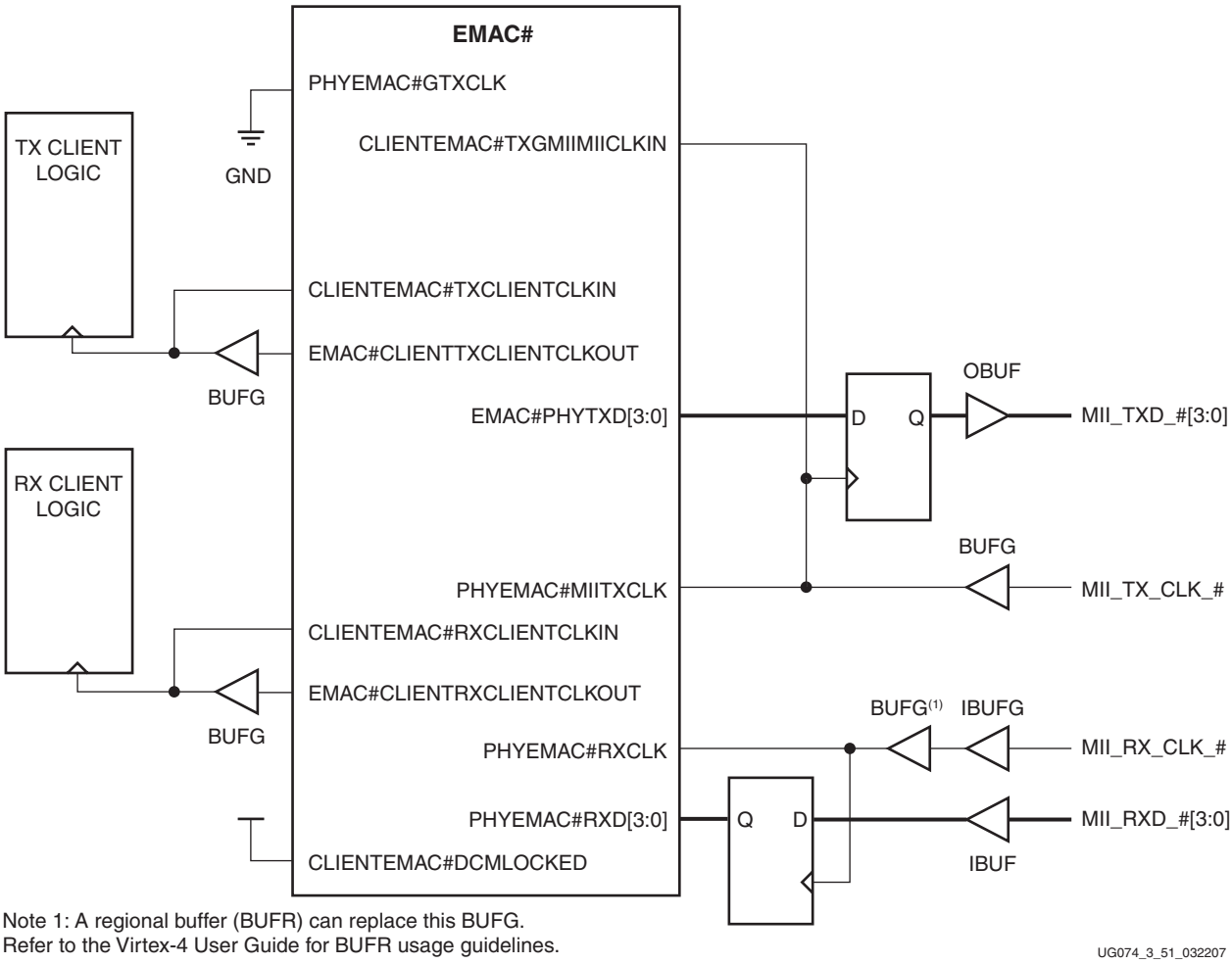


Figure 4-2: **MII Clock Management**

MII Clock Management with Clock Enable

It is possible to only use two BUFGs. To accomplish this BUFG reduction, the client and MII logic must be constrained to run at 125 MHz. Also clock enable signals must be added to the client logic. Figure 4-3 shows the MII clock management with a clock enable scheme.

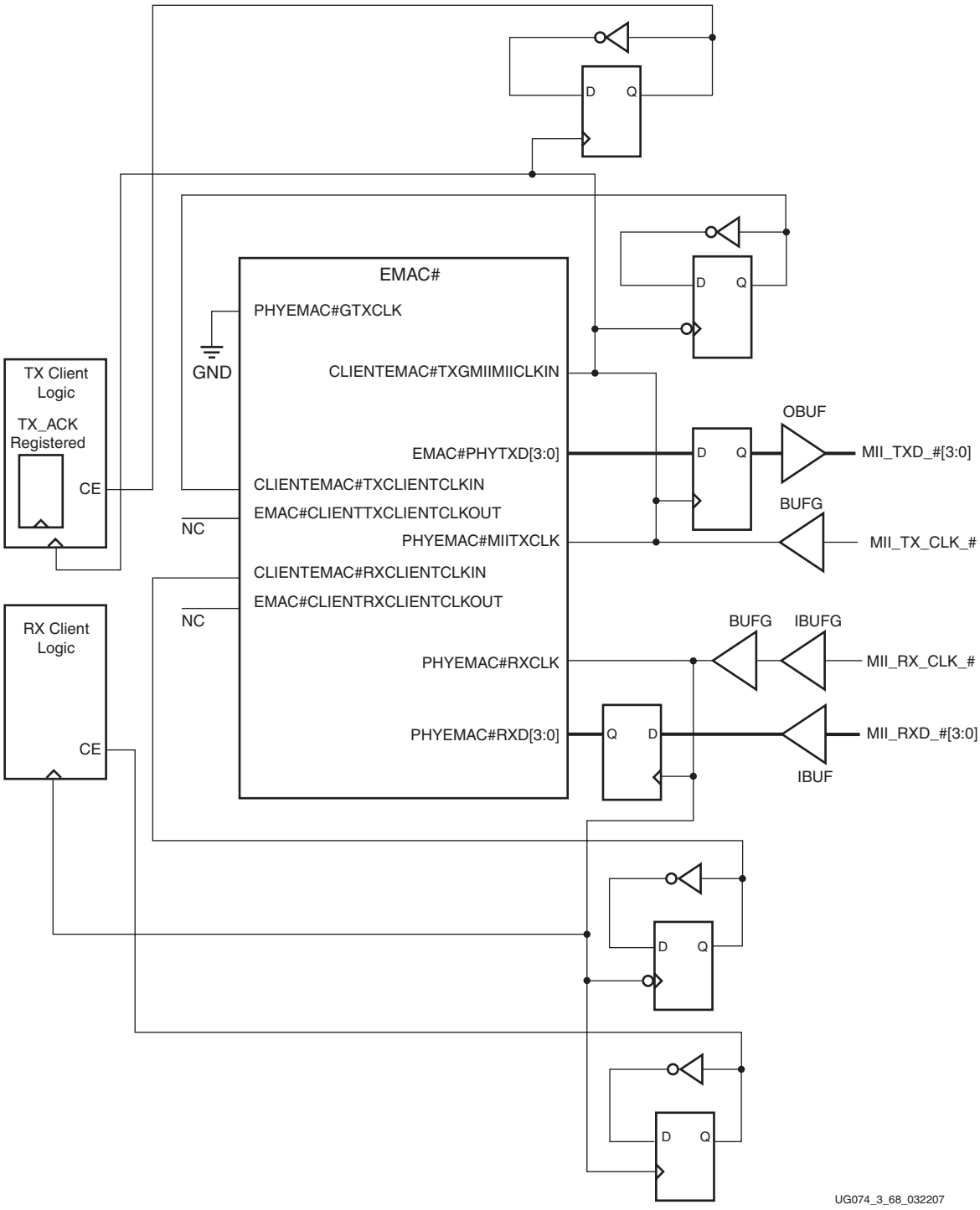


Figure 4-3: **MII Clock Management with Clock Enable**

Using the example in Figure 4-3, all logic is now clocked on the MII interface clock outputs. These outputs run at 25 MHz at 100 Mb/s and 2.5 MHz at 10 Mb/s, or twice as fast as the client clock inputs. To produce the correct clock frequency on these inputs, the MII clocks are put through a toggle flip-flop (clocked on the falling edge of the MII clock) and routed to the client clock inputs. The client logic must also be clock enabled to achieve the correct

data rate at the Ethernet MAC input. The clock enables for the client logic are provided by the output of another toggle flip-flop clocked on the rising edge of the MII clock.

Figure 4-4 shows the timing of a data transfer over the client interface.

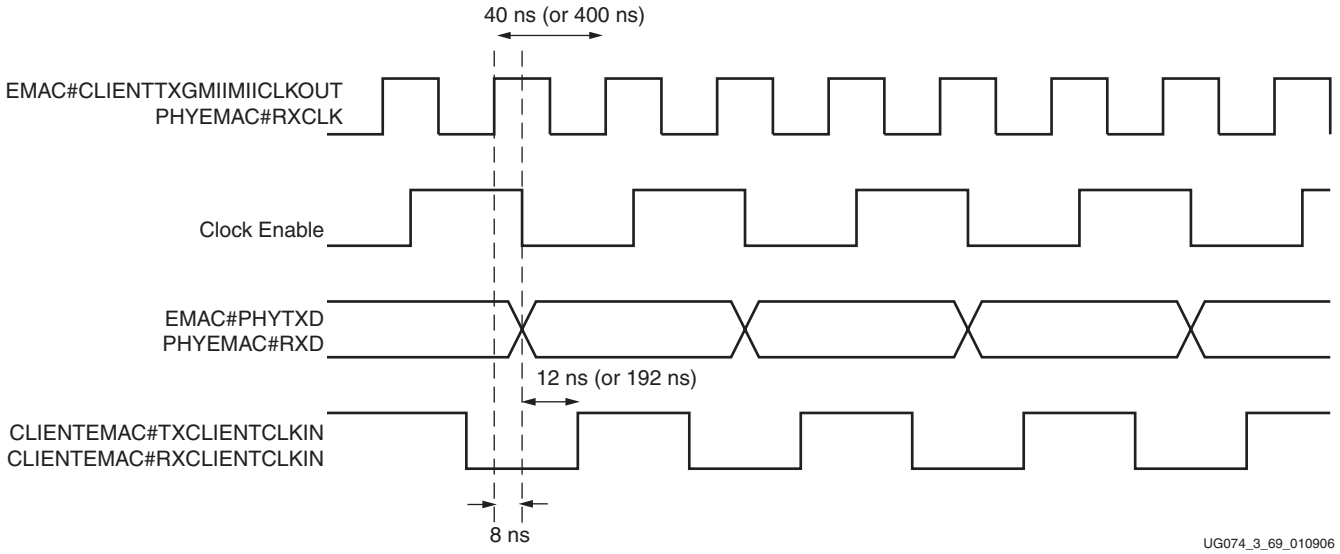


Figure 4-4: **Client Interface Timing**

The MII clock has a period of 40 ns at 100 Mb/s and 400 ns at 10 Mb/s. Since the client and MII logic are constrained to run at 1 Gb/s (125 MHz), the data will be stable 8 ns after the rising clock edge. This gives a window of 32 ns at 100 Mb/s for the data to be sampled into the Ethernet MAC. As the client clock is derived on the falling edge of the MII clock the data is clocked into the MAC at least 12 ns after the data is stable.

Using this technique the EMAC#CLIENTTXACK signal must be registered at the output of the Ethernet MAC. EMAC#CLIENTTXACK is generated on the rising edge of the client clock input CLIENTEMAC#TXCLIENTCLKIN and sampled by the client logic on the rising edge of the MII clock. This technique can lead to the first byte of data being removed from the client transmitter bus too soon. However, if the EMAC#CLIENTTXACK signal is registered on the MII clock (and clock enabled like the rest of the client logic), the data removal problem will not happen. Figure 4-5 illustrates the timing diagram.

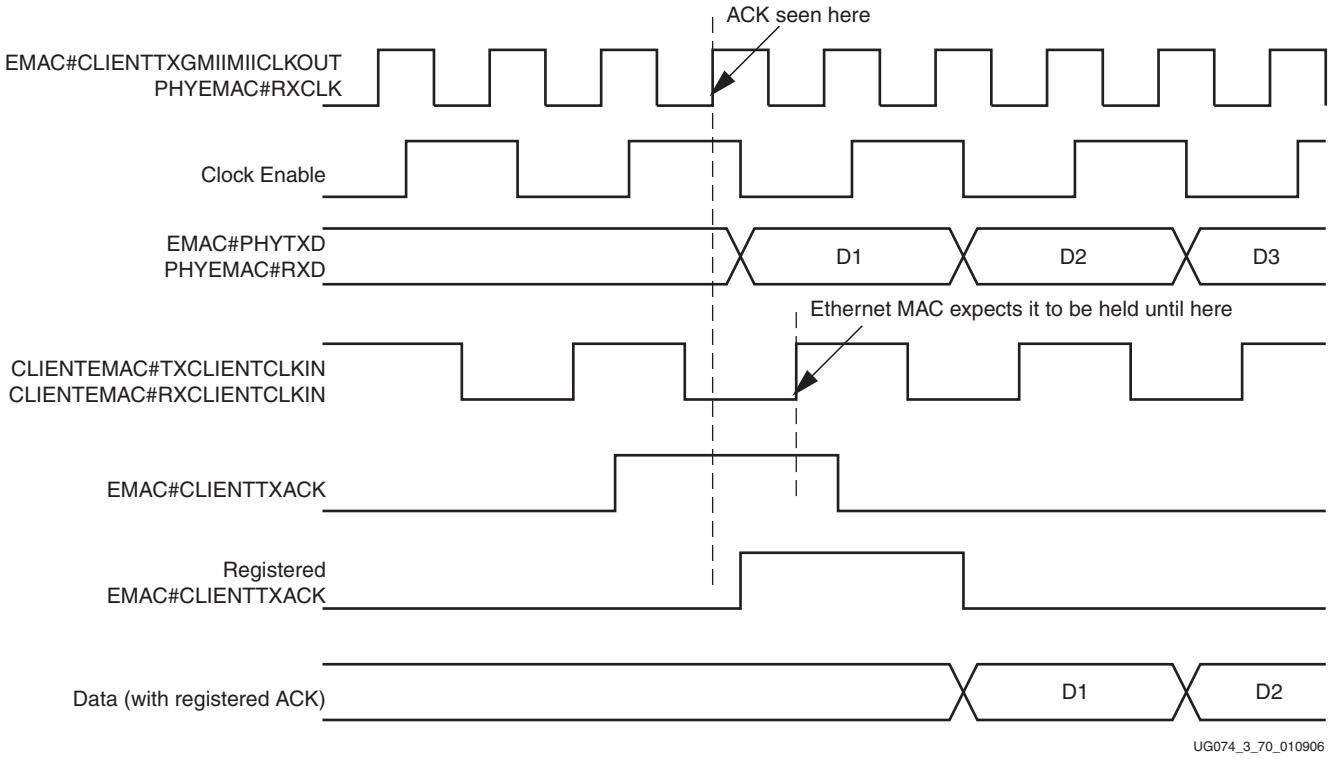


Figure 4-5: **TX Acknowledge Register**

MII Signals

An Ethernet MAC wrapper has all necessary pin connections to configure the primitive into the media independent interface. Table 4-1 describes the MII interface signals.

Table 4-1: **MII Interface Signals**

Signal	Direction	Description
MII_TXD[3:0]_#	Output	Transmits data to PHY
MII_TX_EN_#	Output	Transmits data enable to PHY
MII_TX_ER_#	Output	Transmits error signal to PHY
MII_TX_CLK_#	Input	Recovered transmit clock by PHY
MII_CRS_#	Input	Carrier sense control signal from PHY
MII_COL_#	Input	Collision detect control signal from PHY
MII_RX_CLK_#	Input	Recovered clock from data stream by PHY
MII_RXD[3:0]_#	Input	Receive data from PHY
MII_RX_DV_#	Input	Receive data valid control signal from PHY
MII_RX_ER_#	Input	Receive data error signal from PHY

Gigabit Media Independent Interface (GMII) Signals

Gigabit Media Independent Interface (GMII) is designed to IEEE Std 802.3-2002 Clause 35. It is used for 1000 Mb/s. The physical interface is used for tri-speed operation of the Ethernet MAC.

GMII Interface

Figure 4-6 shows the Ethernet MAC configured with GMII as the physical interface. In this interface, not all the ports of the Ethernet MAC are used.

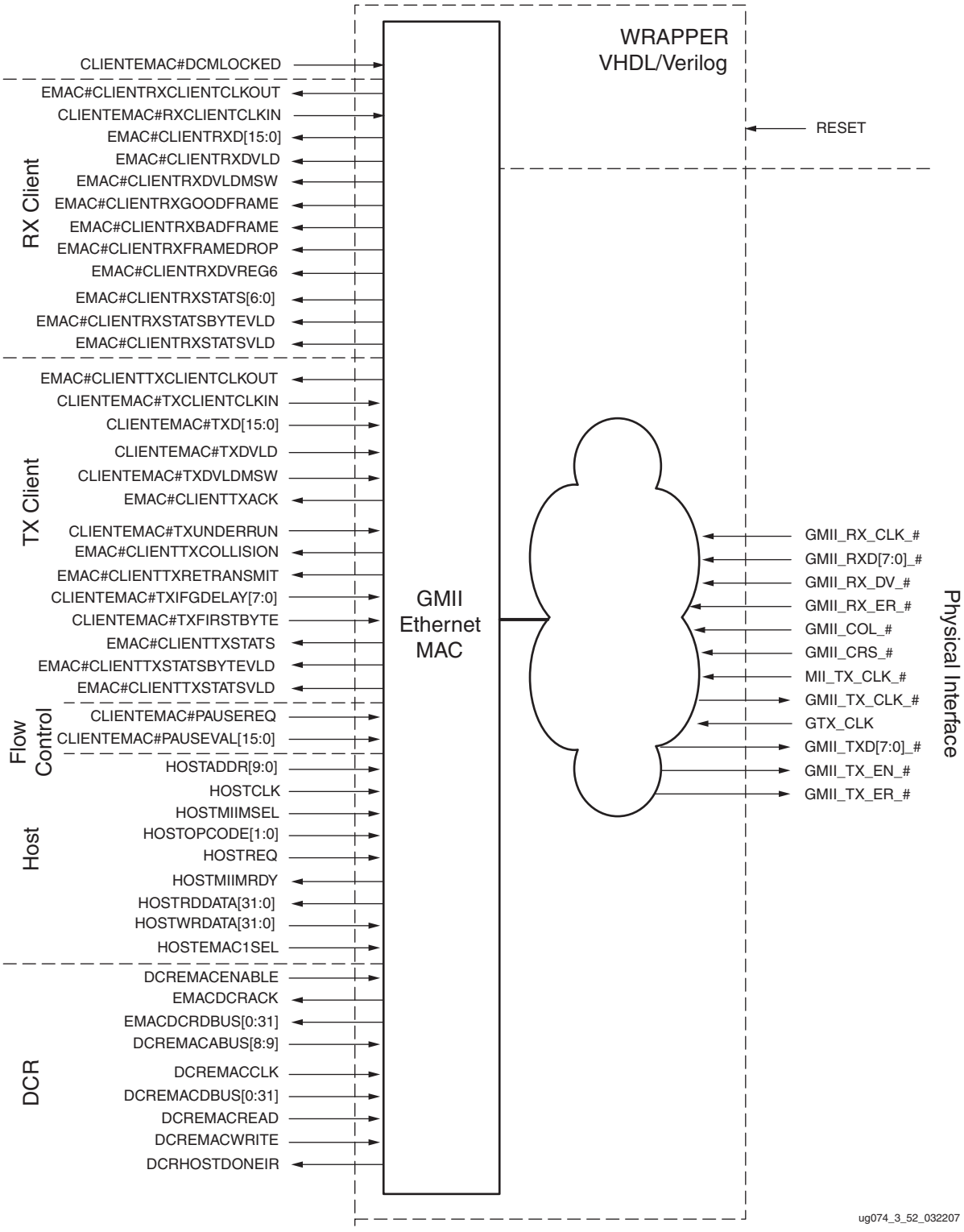


Figure 4-6: **Ethernet MAC Configured in GMII Mode**

GMII Clock Management

1 Gb/s GMII Only

Figure 4-7 shows GMII clock management when using one Ethernet MAC. The GTX_CLK has a frequency of 125 MHz. GTX_CLK must be provided to the Ethernet MAC. This is a high quality 125 MHz clock that satisfies the IEEE Std 802.3-2002 requirements.

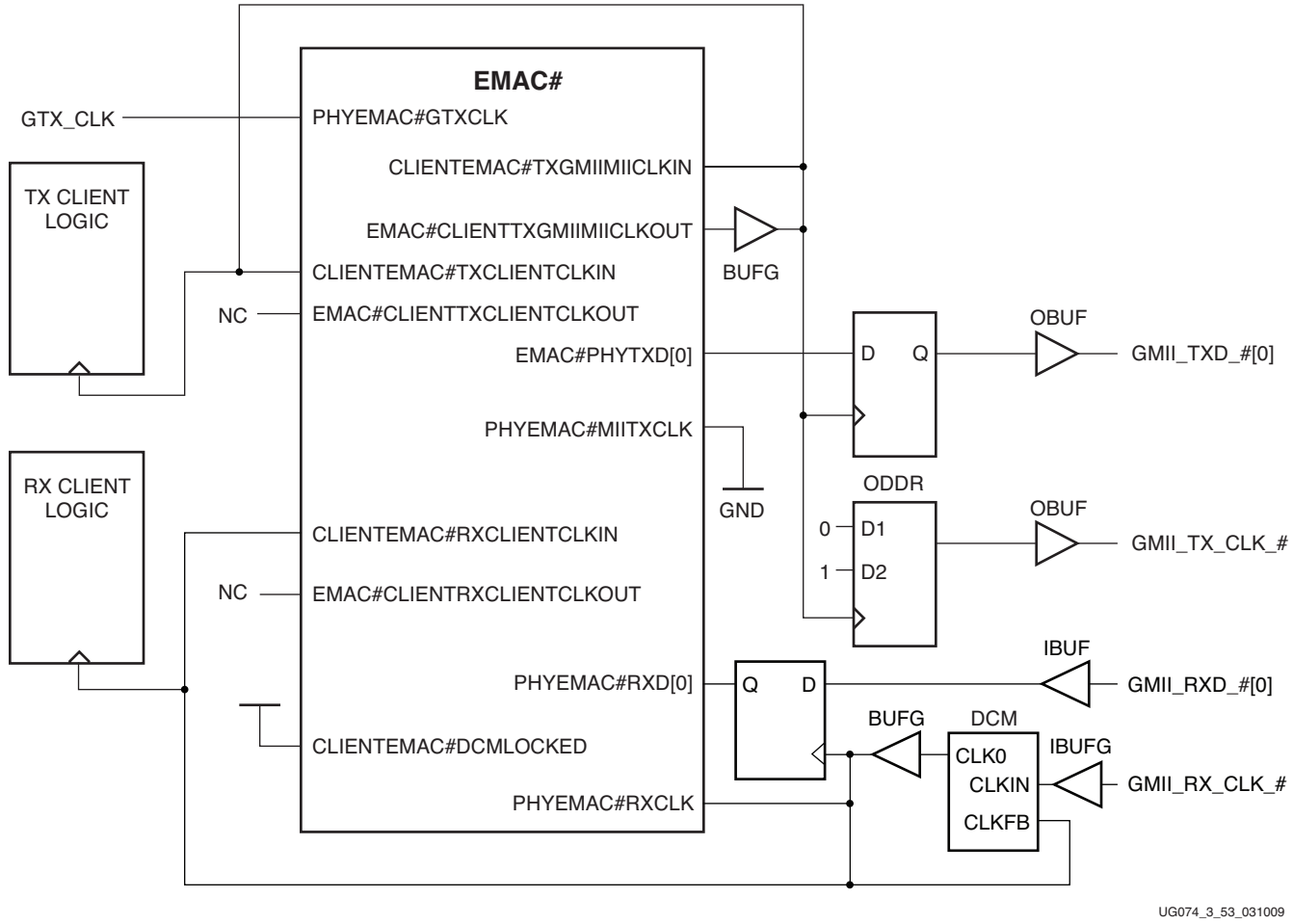


Figure 4-7: **1 Gb/s GMII Clock Management**

The EMAC#CLIENTTXGMIIMIICLKOUT output port drives all of the transmit logic through a BUFG. The output of the BUFG connects to:

- GMII_TXD registers in the FPGA fabric
- Input port CLIENTEMAC#TXGMIIMIICLKIN
- CLIENTEMAC#TXCLIENTCLKIN
- TX client logic

PHYEMAC#MIITXCLK must be tied to ground. The GMII_RX_CLK_# is generated from the PHY and is connected to PHYEMAC#RXCLK through a DCM and a BUFG. The CLIENTEMAC#DCMLOCKED port must be tied High. The DCM is used to shift the received GMII clock with respect to the data, in order to sample a 2 ns setup, 0 ns hold window at the device pads. Phase shifting is applied to the DCM to fine tune the setup and

hold times of the input GMII receiver signals which are sampled at the GMII IOB input flip-flops.

Figure 4-8 shows the GMII clocking scheme with two Ethernet MACs enabled. It is similar to the single Ethernet MAC clocking scheme, however, one of the EMAC#CLIENTTXGMIIMIICLKOUT signals is used for all (both Ethernet MACs) transmitter logic.

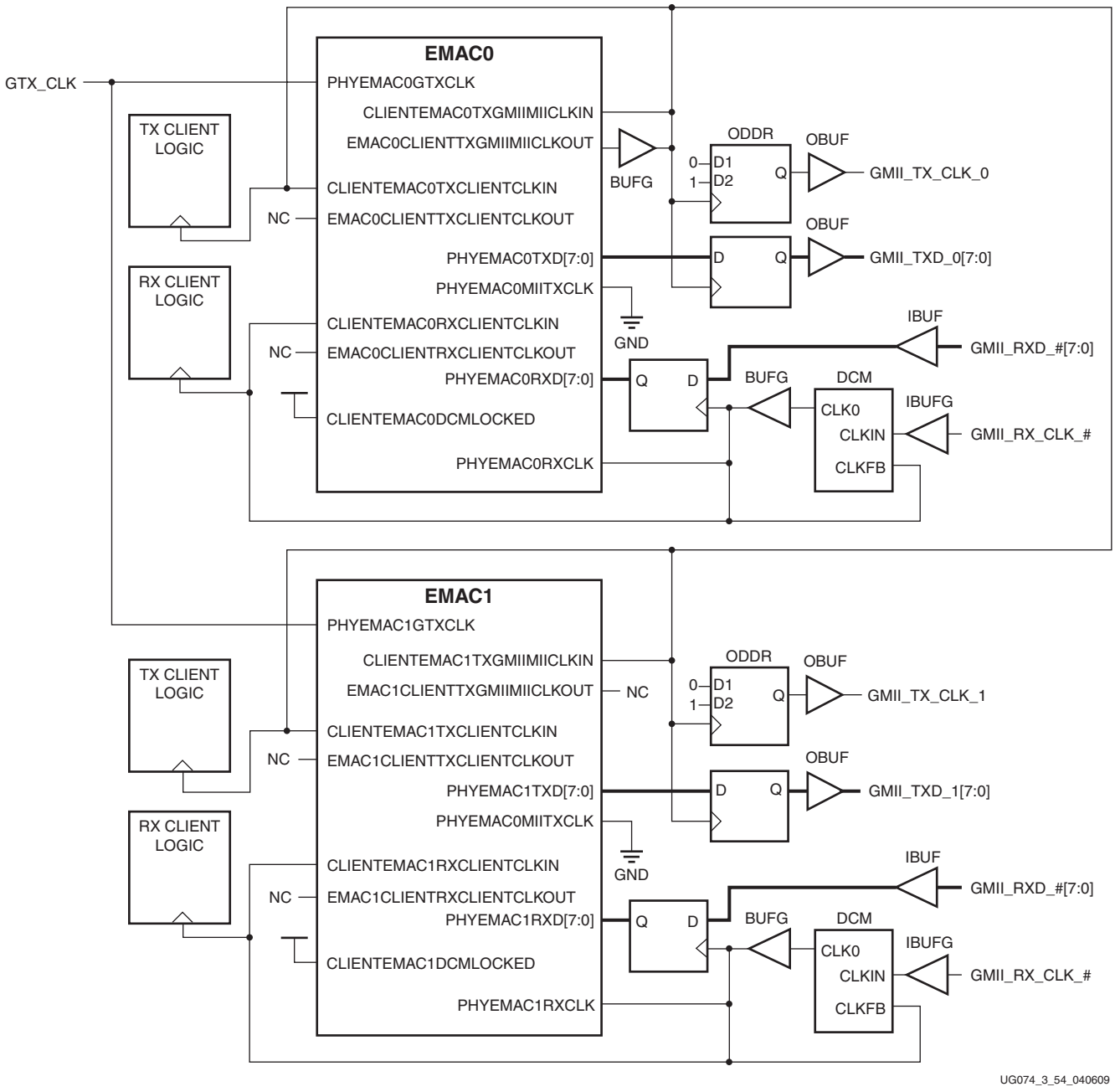


Figure 4-8: **1 Gb/s GMII Clock Management with Two Ethernet MACs Enabled**

Tri-Mode Operation

Figure 4-9 shows the clock management used with the tri-mode GMII interface. GTX_CLK must be provided to the Ethernet MAC with a high quality 125 MHz clock that satisfies the IEEE Std 802.3-2002 requirements.

CLIENTEMAC#TXGMIIMIICLKIN, PHYEMAC#MIITXCLK, and the GMII transmit registers are all clocked by a BUFGMUX. This mux selects GTX_CLK when the speed is 1 Gb/s, and MII_TX_CLK_# when the speed of operation is 10 or 100 Mb/s.The EMAC#CLIENTTXCLIENTCLKOUT output port connects to the CLIENTEMAC#TXCLIENTCLKIN input port and transmit client logic in the FPGA fabric through a BUFG. The receive client clocking is similar.

The GMII_RX_CLK_# generated from the PHY has a frequency of either 2.5 MHz, 25 MHz or 125 MHz, depending on the operating speed of the Ethernet MAC. The CLIENTEMAC#DCMLOCKED port must be tied High.

A DCM must be used on the GMII_RX_CLK_# clock path as illustrated in Figure 4-9 to meet the GMII input setup and hold requirements when operating at 1 Gb/s. Phase shifting may then be applied to the DCM to fine tune the setup and hold times of the input GMII receiver signals which are sampled at the GMII IOB input flip-flops.

When operating at 10 Mb/s and 100 Mb/s, the DCM is bypassed and held in reset. This is achieved using the BUFGMUX global clock multiplexer shown in Figure 4-9. It is a requirement to bypass the DCM because the clock frequency of GMII_RX_CLK_# is 2.5 MHz when operating at 10 Mb/s and 2.5 MHz is below the DCM low frequency threshold for Virtex®-4 FPGAs. However, at the 10 Mb/s and 100 Mb/s operating speeds, input setup and hold margins increase appropriately and the input MII data can be sampled correctly without use of the DCM.

The GMII_TX_CLK_# is derived from the Ethernet MAC, routed through an OBUF, and then connected to the PHY. Since GMII_TX_CLK_# is derived from EMAC#CLIENTTXGMIIMIICLKOUT or MII_TX_CLK_#, its frequency automatically changes between 125 MHz, 25 MHz, or 2.5 MHz depending on the speed setting of the Ethernet MAC.

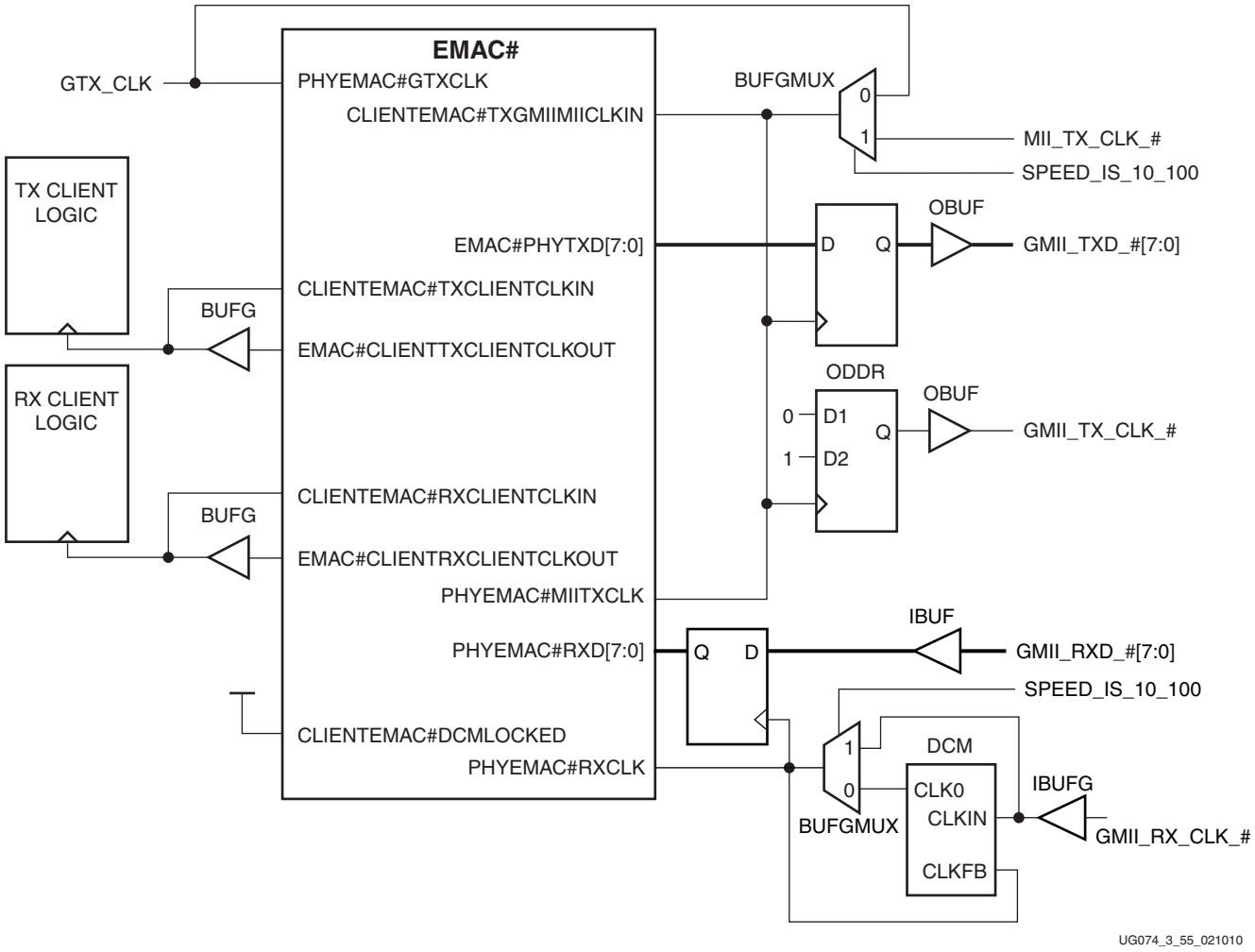


Figure 4-9: **GMII Tri-Mode Operation Clock Management**

Tri-Mode Operation with Byte PHY Enabled (Full-Duplex Only)

Figure 4-10 shows an alternative clock management scheme for the tri-mode GMII interface operating in full-duplex mode. In this scheme, the EMAC is configured to run in 1 Gb/s mode at all times, resulting in the datapath being 8 bits wide at both the client and the physical interfaces at all speeds. At 1 Gb/s all external logic is clocked at 125 MHz. At 100 Mb/s and 10 Mb/s, the logic is clocked at 12.5 MHz and 1.25 MHz, respectively.

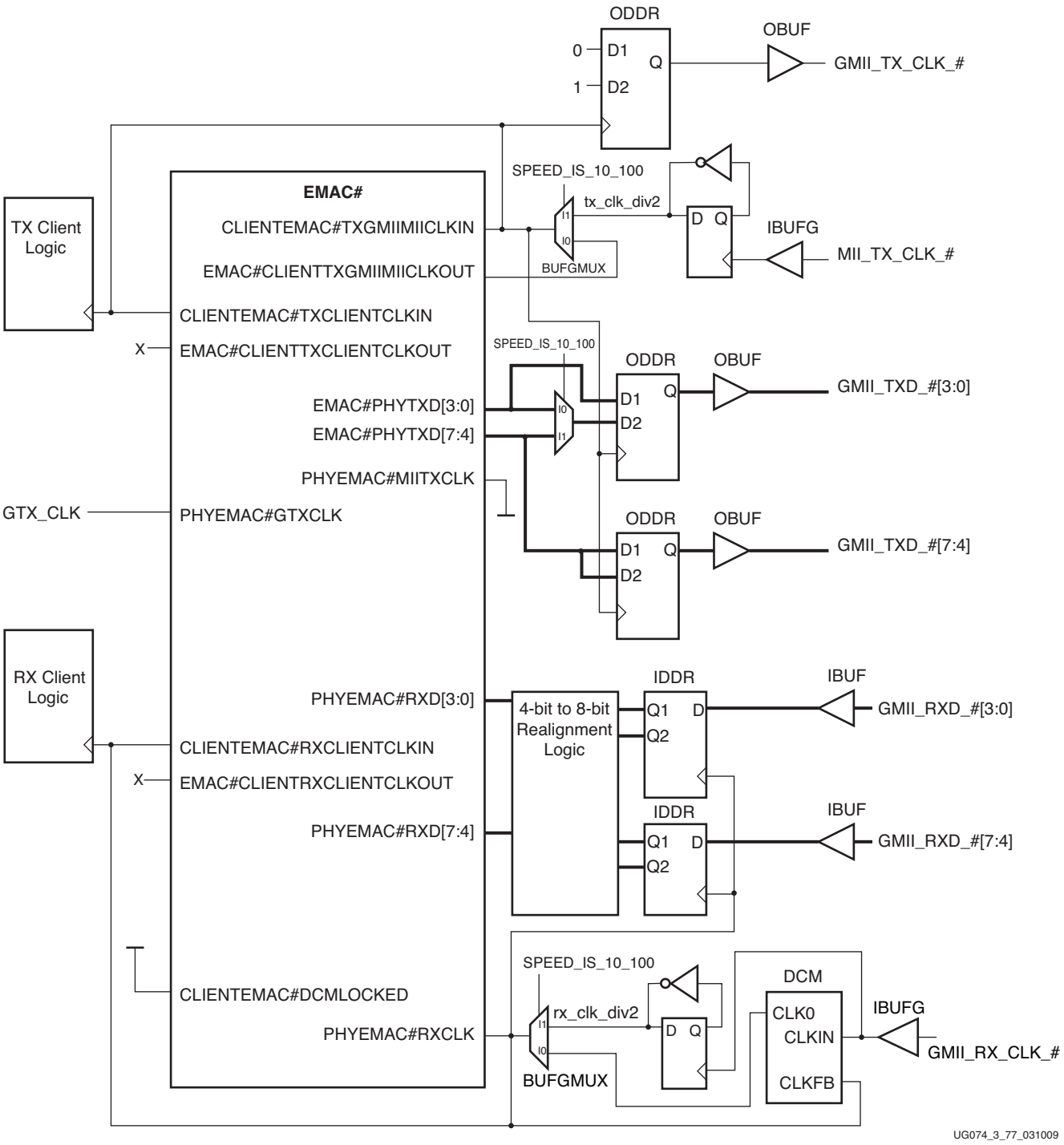


Figure 4-10: **Tri-Mode GMII Clock Management with Byte PHY Enabled**

Double data rate input and output registers are used to achieve the 25 MHz and 2.5 MHz 4-bit data rate at speeds below 1 Gb/s, resulting in a scheme that utilizes two clock buffers less than Figure 4-9. However, because the EMAC is kept in 1 Gb/s mode at all speeds, the scheme does not operate in half-duplex mode. In addition, alignment logic must be provided on the receiver side to align the start of frame delimiter in the 8-bit data input to the EMAC.

GTX_CLK must be provided to the Ethernet MAC with a high-quality 125 MHz clock that satisfies the IEEE Std 802.3-2002 requirements.

The CLIENTEMAC#TXGMIIMIICLKIN and CLIENTEMAC#TXCLIENTCLKIN ports are supplied by the output of a BUFGMUX. At 1 Gb/s, the BUFGMUX routes through the EMAC#CLIENTTXCLIENTCLKOUT signal. At 100 Mb/s and 10 Mb/s, the BUFGMUX is switched to supply tx_clk_div2 to the EMAC. This signal is the MII_TX_CLK_# input divided by two in frequency. The output of the BUFGMUX also clocks all the transmit client and GMII logic.

A DCM must be used on the GMII_RX_CLK_# clock path as illustrated in Figure 4-10 to meet the GMII input setup and hold requirements when operating at 1 Gb/s. Phase shifting may then be applied to the DCM to fine tune the setup and hold times of the input GMII receiver signals which are sampled at the GMII IOB input flip-flops.

When operating at 10 Mb/s and 100 Mb/s, the DCM is bypassed and held in reset. This is achieved using the BUFGMUX global clock multiplexer shown in Figure 4-10. It is a requirement to bypass the DCM because the clock frequency of GMII_RX_CLK_# is 2.5 MHz when operating at 10 Mb/s and 2.5 MHz is below the DCM low frequency threshold for Virtex-4 FPGAs. However, at the 10 Mb/s and 100 Mb/s operating speeds, input setup and hold margins increase appropriately and the input MII data can be sampled correctly without use of the DCM.

At 1 Gb/s, the second BUFGMUX supplies the DCM phase-shifted GMII_RX_CLK_# to the PHYEMAC#RXCLK and CLIENTEMAC#RXCLIENTCLKIN ports. At 100 Mb/s and 10 Mb/s, the BUFGMUX is switched to provide rx_clk_div2 to the EMAC. This clock is the GMII_RX_CLK_# input divided by two in frequency. The output of the BUFGMUX also clocks all the receiver client and GMII logic. The CLIENTEMAC#DCMLOCKED port must be tied High.

GMII Signals

An Ethernet MAC wrapper has all necessary pin connections to configure the primitive into GMII. Table 4-2 describes the GMII interface signals.

Table 4-2: **GMII Interface Signals**

Signal	Direction	Description
GTX_CLK	Input	The transmit clock at 125 MHz. The clock timing and other characteristics meet the IEEE Std 802.3-2002 specification. Other transmit clocks are derived from this clock.
GMII_TXD[7:0]_#	Output	Transmits data to PHY.
GMII_TX_EN_#	Output	Transmits data enable to PHY.
GMII_TX_ER_#	Output	Transmits error signal to PHY.
GMII_TX_CLK_#	Output	Transmits clock out to PHY.
GMII_CRS_#	Input	Carrier sense control signal from PHY, only if tri-mode is selected.

Table 4-2: **GMII Interface Signals *(Cont'd)***

Signal	Direction	Description
GMII_COL_#	Input	Collision detect control signal from PHY, only if tri-mode is selected.
GMII_RX_CLK_#	Input	Recovered clock from data stream by PHY.
GMII_RXD[7:0]_#	Input	Receive data from PHY.
GMII_RX_DV_#	Input	Receive data valid control signal from PHY.
GMII_RX_ER_#	Input	Receive data error signal from PHY.

10/100/1000 RGMII

RGMII, an alternative to GMII, was defined by Hewlett-Packard. It reduces the number of pins required to connect the Ethernet MAC to the PHY from 24 to 12. RGMII achieves this 50% pin count reduction in the interface by using double data rate (DDR) flip-flops.

For more information on RGMII, refer to the *Hewlett-Packard RGMII Specification, version 1.3 and 2.0*.

1 Gb/s RGMII Interface

Figure 4-11 shows the Ethernet MAC configured with RGMII as the physical interface. In this interface, not all the ports of the Ethernet MAC are used.

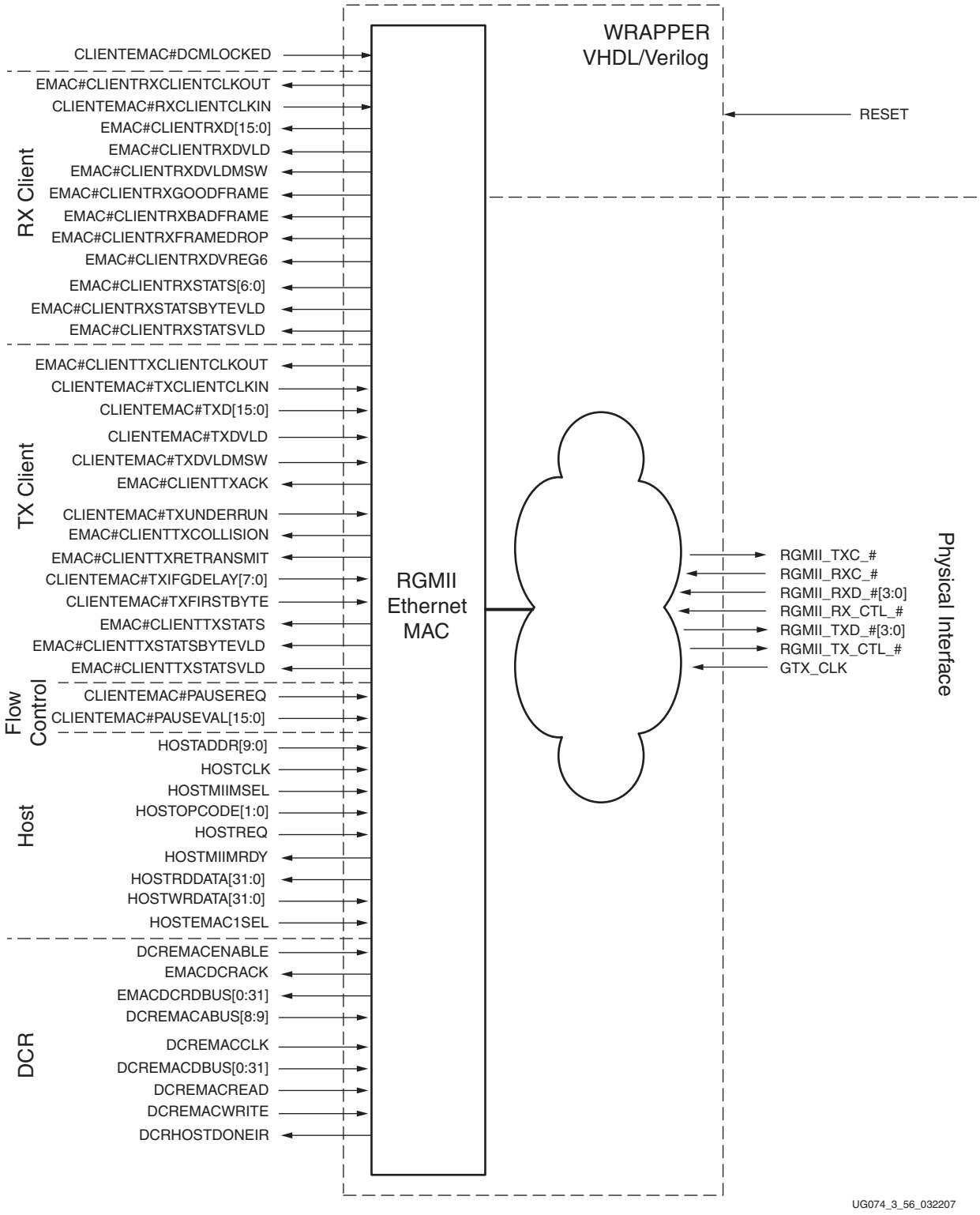


Figure 4-11: **Ethernet MAC Configured in RGMII Mode**

1 Gb/s RGMII Clock Management

Figure 4-12 shows the clock management used with the RGMII interface when using the *Hewlett Packard RGMII Specification v1.3*. GTX_CLK must be provided to the Ethernet MAC with a high quality 125 MHz clock that satisfies the IEEE Std 802.3-2002 requirements. The EMAC#CLIENTTXGMIIMIICLKOUT output port drives all transmitter logic through a BUFG.

The RGMII_TXC_# is derived from the Ethernet MAC by routing to an IOB double data rate (DDR) output register followed by an OBUF (which is then connected to the PHY). The use of the DDR register ensures that the forwarded clock is exactly in line with the RGMII transmitter data as specified in the Hewlett Packard RGMII Specification, v1.3.

The RGMII_RXC_# is generated from the PHY and is connected to the PHYEMAC#RXCLK pin and receive logic through a DCM and a BUFG. A DCM must be used on the RGMII_RXC_# clock path as illustrated in Figure 4-12 to meet the RGMII 1 ns setup and 1 ns hold requirements. Phase shifting may then be applied to the DCM to fine-tune the setup and hold times of the input RGMII receiver signals which are sampled at the RGMII IOB input flip-flops. The CLIENTEMAC#DCMLOCKED port must be tied High.

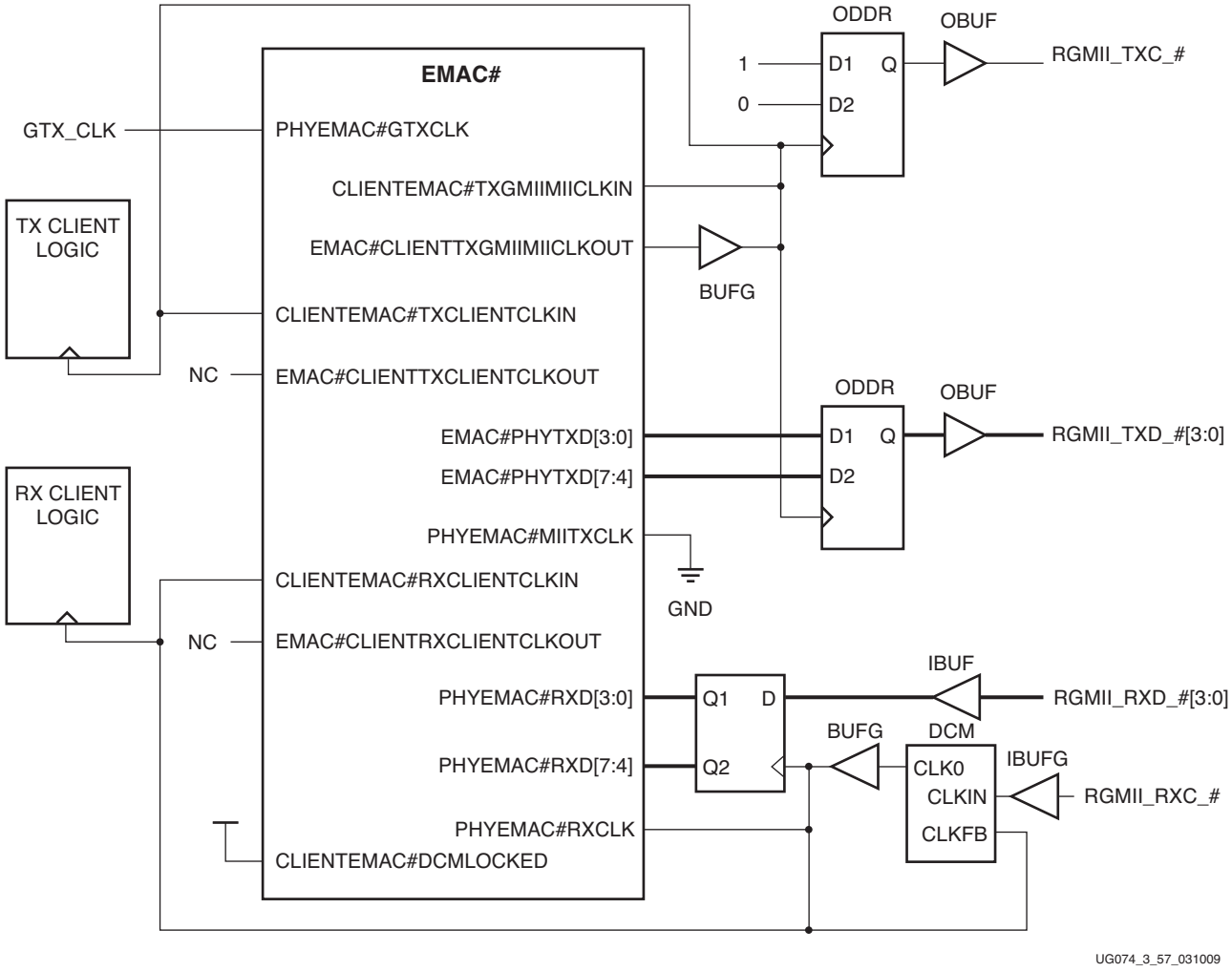
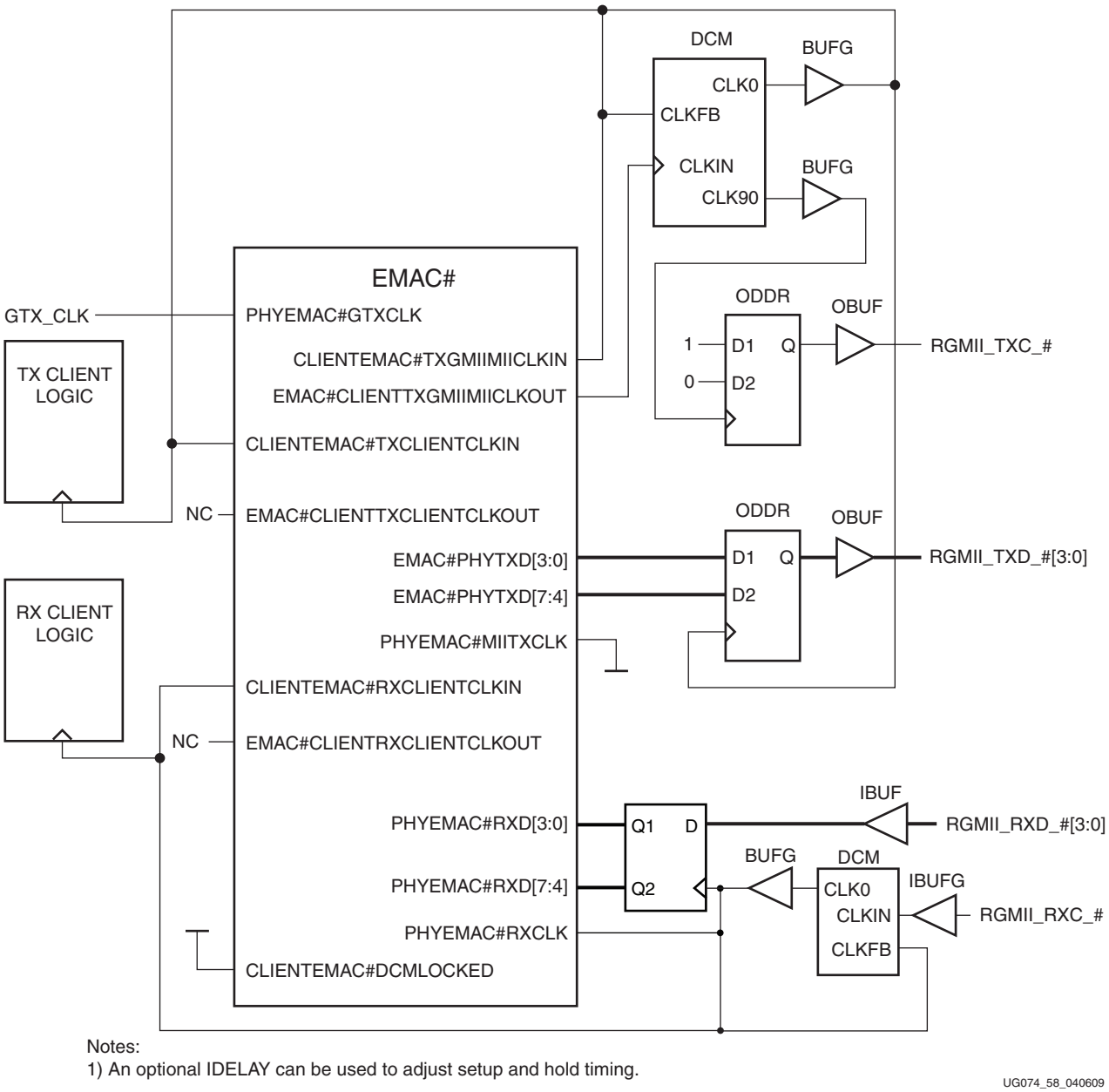


Figure 4-12: **1 Gb/s RGMII Hewlett Packard v1.3 Clock Management**

Figure 4-13 shows the clock management used with the RGMII interface when following the *Hewlett Packard RGMII specification v2.0*. GTX_CLK must be provided to the Ethernet MAC with a high quality 125 MHz clock that satisfies the IEEE Std 802.3-2002 requirements. The EMAC#CLIENTTXGMIIMIICLKOUT output port connects to a DCM which in turn drives the RGMII transmitter logic in the FPGA fabric and the CLIENTEMAC#TXGMIIMIICLKIN input port. The RGMII_TXC_# is derived from the CLK90 output of the DCM. The DCM is used to generate 2 ns of skew required between RGMII_TXC_# and the RGMII_TXD_# at the FPGA device pads. This delay is specified in the Hewlett Packard RGMII Specification, v2.0 to provide setup and hold time on the external interface.



Notes:
1) An optional IDELAY can be used to adjust setup and hold timing.

UG074_58_040609

Figure 4-13: **1 Gb/s RGMII Hewlett Packard v2.0 Clock Management**

The RGMII_RXC_# is generated from the PHY and connected to the PHYEMAC#RXCLK pin and receive logic through a DCM and a BUFG. A DCM must be used on the RGMII_RXC_# clock path as illustrated in Figure 4-13 to meet the RGMII 1 ns setup and 1 ns hold requirements at 1 Gb/s. Phase shifting may then be applied to the DCM to fine tune the setup and hold times of the input RGMII receiver signals which are sampled at the RGMII IOB input flip-flops.

When operating at 10 Mb/s and 100 Mb/s, the DCM is bypassed and held in reset. This is achieved using the BUFGMUX global clock multiplexer shown in Figure 4-13. It is a requirement to bypass the DCM because the clock frequency of RGMII_RXC_# is 2.5 MHz when operating at 10 Mb/s and 2.5 MHz is below the DCM low frequency threshold for Virtex-4 FPGAs. However, at the 10 Mb/s and 100 Mb/s operating speeds, input setup and hold margins increase appropriately and the input RGMII data can be sampled correctly without use of the DCM. The CLIENTEMAC#DCMLOCKED port must be tied High.

Tri-Mode RGMII v2.0

Figure 4-14 shows the tri-mode clock management following the *Hewlett Packard RGMII specification v2.0*. GTX_CLK must be provided to the Ethernet MAC with a high-quality 125 MHz clock that satisfies the IEEE Std 802.3-2002 requirements. The EMAC#CLIENTTXGMIIMIICLKOUT port generates the appropriate frequency deriving from GTX_CLK and the operating frequency of the link. It clocks directly to the RGMII_TXD_# ODDR registers.

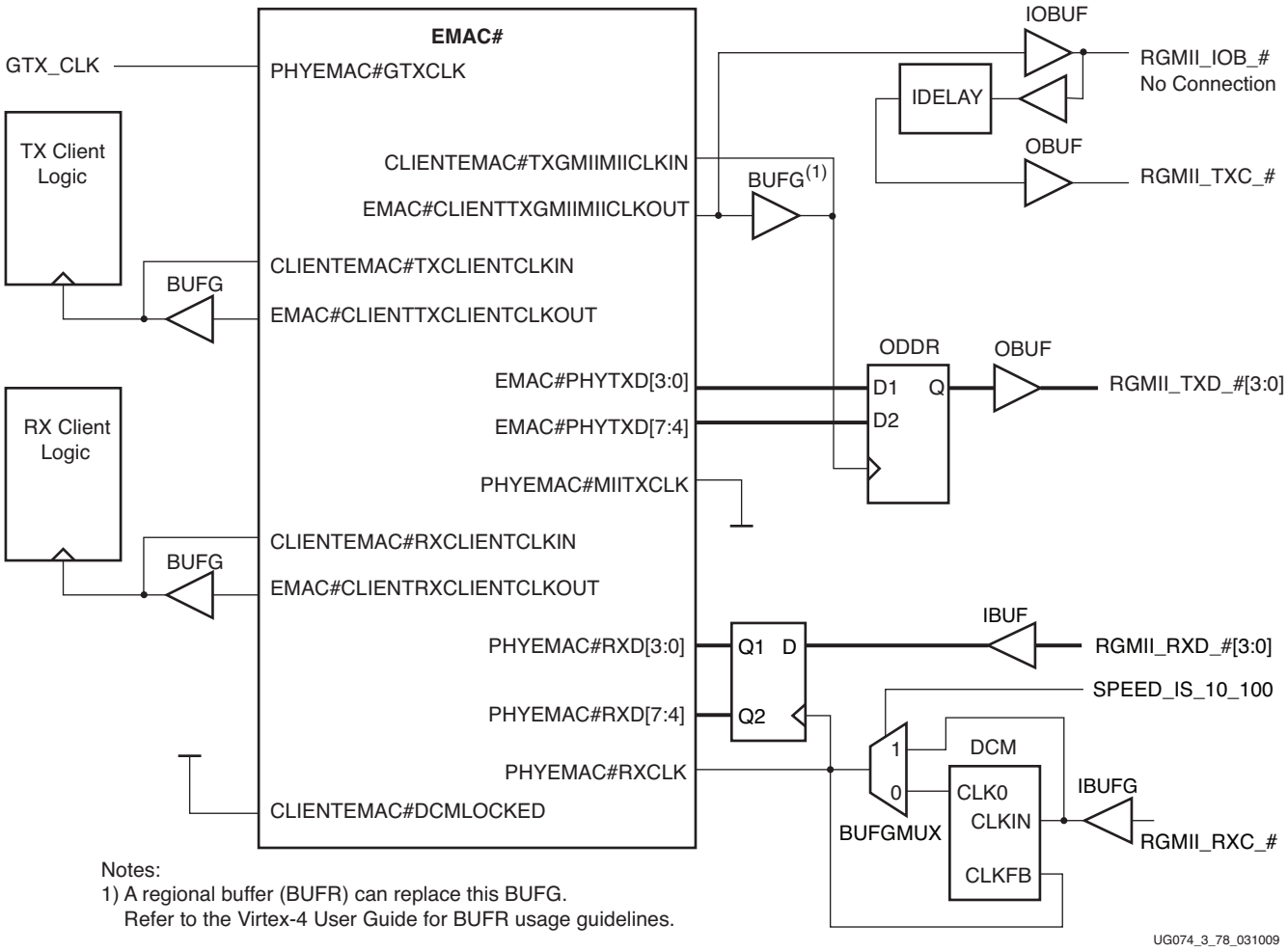


Figure 4-14: **Tri-Mode RGMII v2.0 Clock Management**

An IDELAY is used to generate 2 ns of skew required between RGMII_TXC_# and RGMII_TXD_# at the pin level. To access the IDELAY component, the EMAC#CLIENTTXGMIIMIICLKOUT signal is routed to an unused IOB configured as an IOBUF. The T control pin of the IOBUF is tied to ground to loop the clock back to the IDELAY. The IDELAY setting can be used to shift the clock through the data. The output of the IOB must not be connected to an external signal.

The EMAC#CLIENTTXCLIENTCLKOUT output port connects to the CLIENTEMAC#TXCLIENTCLKIN input port and transmitter client logic in the FPGA fabric through a BUFG. The receiver client clocking is similar.

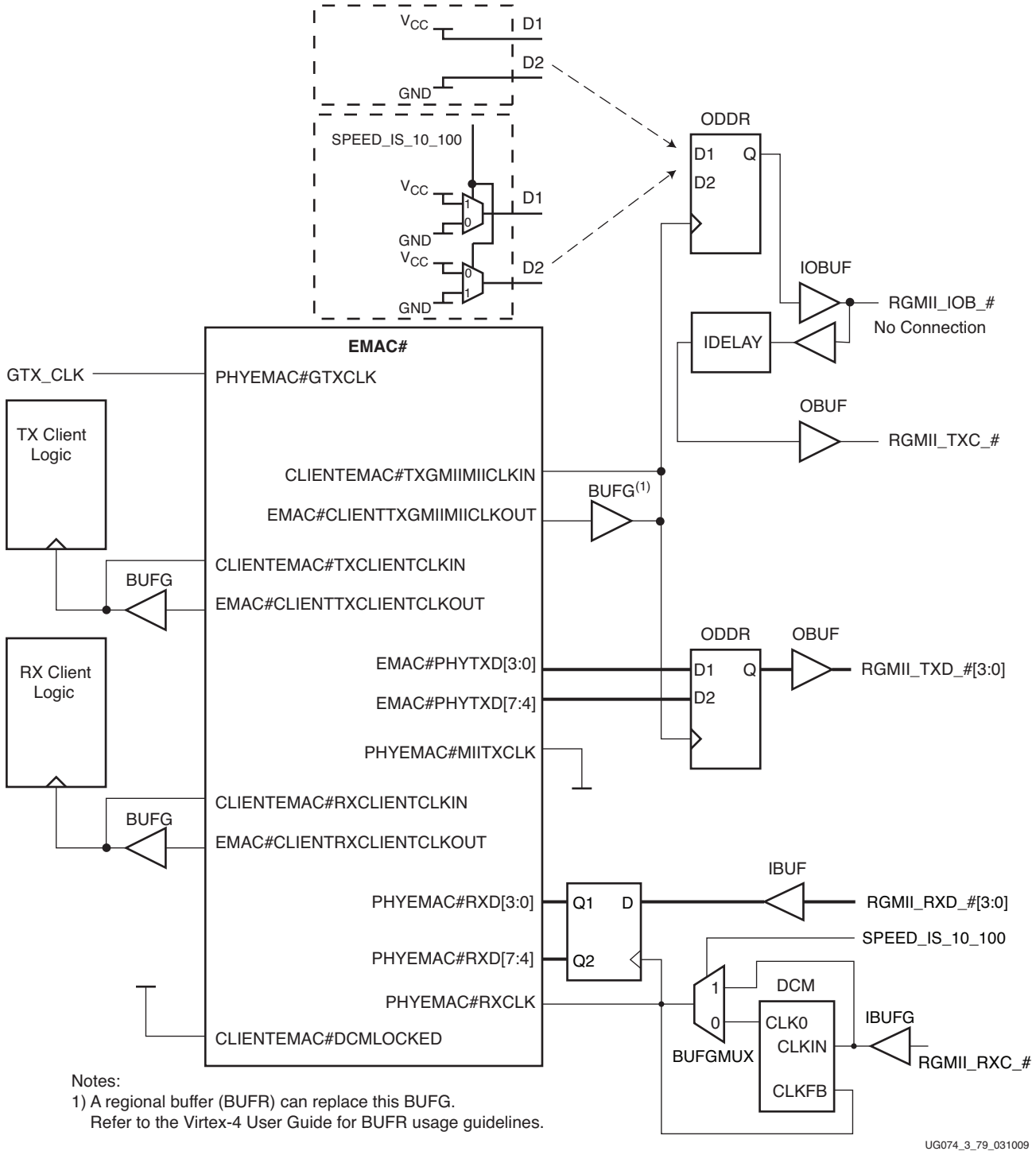


Figure 4-15: **Alternative Tri-Mode RGMII v2.0 Clock Management**

The CLIENTEMAC#DCMLOCKED port must be tied HIGH. The RGMII_RXC_# is generated from the PHY and connected to the PHYEMAC#RXCLK pin and receive logic through a DCM and a BUFG. A DCM must be used on the RGMII_RXC_# clock path as illustrated in Figure 4-14 to meet the RGMII 1 ns setup and 1 ns hold requirements at 1 Gb/s. Phase shifting may then be applied to the DCM to fine-tune the setup and hold

times of the input RGMII receiver signals which are sampled at the RGMII IOB input flip-flops.

When operating at 10 Mb/s and 100 Mb/s, the DCM is bypassed and held in reset. This is achieved using the BUFGMUX global clock multiplexer shown in Figure 4-14. It is a requirement to bypass the DCM because the clock frequency of RGMII_RXC_# is 2.5 MHz when operating at 10 Mb/s and 2.5 MHz is below the DCM low frequency threshold for Virtex-4 FPGAs. However, at the 10 Mb/s and 100 Mb/s operating speeds, input setup and hold margins increase appropriately and the input RGMII data can be sampled correctly without use of the DCM.

In the transmit path, the IDELAY provides a maximum shift of 4.7 ns. If this delay is not sufficient to give the correct skew, the method shown in Figure 4-15 can be used. Here an ODDR is used to control the polarity of the RGMII_IOB_# signal that is fed to the IDELAY.

At 1 Gb/s, if the skew given by connecting the D1 and D2 inputs of ODDR to V_{CC} and GND, respectively, is not sufficient, the D1 and D2 inputs can be inverted to yield a 4 ns shift on the clock with respect to the data. As in Figure 4-14, the IDELAY value is used to provide the correct skew.

The inversion must be removed from the EMAC#CLIENTTXGMIIMIICLKOUT signal for correct operation at speeds below 1 Gb/s. The CLIENTEMAC#DCMLOCKED port must be tied High.

Tri-Mode RGMII v1.3

Figure 4-16 shows the tri-mode clock management following the *Hewlett Packard RGMII specification v1.3*. GTX_CLK must be provided to the Ethernet MAC with a high-quality 125 MHz clock that satisfies the IEEE Std 802.3-2002 requirements. The EMAC#CLIENTTXGMIIMIICLKOUT port generates the appropriate frequency deriving from GTX_CLK and depending on the operating frequency of the link. It clocks directly to the RGMII_TXD_# ODDR registers.

The EMAC#CLIENTTXCLIENTCLKOUT output port connects to the CLIENTEMAC#TXCLIENTCLKIN input port and transmitter client logic in the FPGA fabric through a BUFG. The receiver client clocking is similar.

The RGMII_RXC_# is generated from the PHY and is connected to the PHYEMAC#RXCLK pin and receive logic through a DCM and a BUFG. A DCM must be used on the RGMII_RXC_# clock path as illustrated in Figure 4-16 to meet the RGMII 1 ns setup and 1 ns hold requirements at 1 Gb/s. Phase shifting may then be applied to the DCM to fine tune the setup and hold times of the input RGMII receiver signals which are sampled at the RGMII IOB input flip-flops.

When operating at 10 Mb/s and 100 Mb/s, the DCM is bypassed and held in reset. This is achieved using the BUFGMUX global clock multiplexer shown in Figure 4-16. It is a requirement to bypass the DCM because the clock frequency of RGMII_RXC_# is 2.5 MHz when operating at 10 Mb/s and 2.5 MHz is below the DCM low frequency threshold for Virtex-4 FPGAs. However, at the 10 Mb/s and 100 Mb/s operating speeds, input setup and hold margins increase appropriately and the input RGMII data can be sampled correctly without use of the DCM. The CLIENTEMAC#DCMLOCKED port must be tied High.

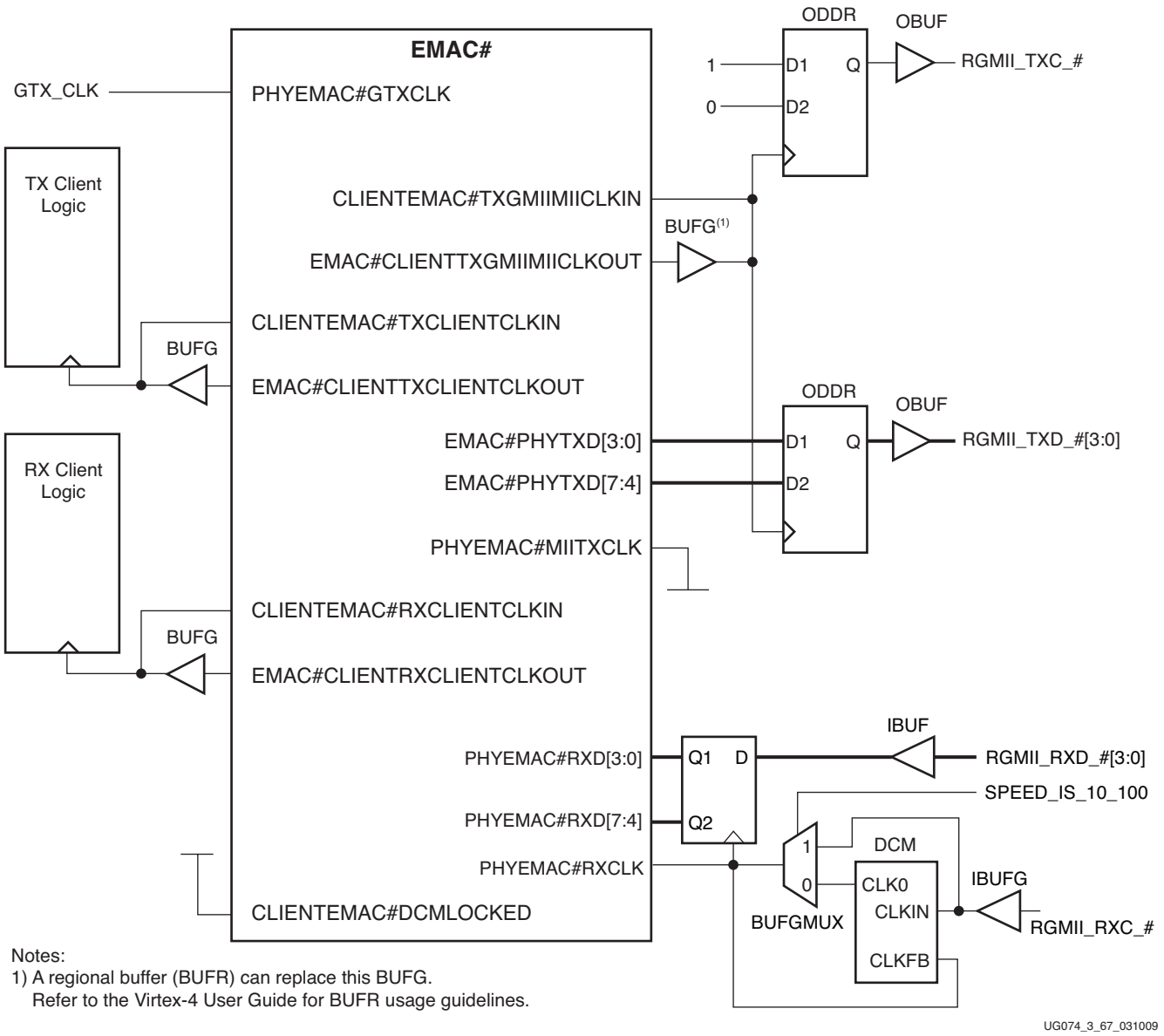


Figure 4-16: **Tri-Mode RGMII v1.3 Clock Management**

RGMII Signals

An Ethernet MAC wrapper has all necessary pin connections to configure the primitive into RGMII. Table 4-3 describes the 10/100/1000 RGMII interface signals.

Table 4-3: **10/100/1000 RGMII Interface Signals**

Signal	Direction	Description
GTX_CLK	Input	The transmit clock at 125 MHz. The clock timing and characteristics meet the IEEE Std 802.3-2002 specification. Other transmit clocks are derived from this clock.
RGMII_TXC_#	Output	Transmits clock out to the PHY.
RGMII_TXD_#	Output	Transmits the first 4 bits of packet data ([3:0]) on the rising edge of RGMII_TXC_#, and the top 4 bits ([7:4]) on the falling edge. TXD is 4 bits wide connecting to the PHY.
RGMII_TX_CTL_#(1)	Output	This signal is TXEN on the rising edge of RGMII_TXC_#, and an encoded TXERR on the falling edge.
RGMII_RXC_#	Input	Recovered clock from data stream by the PHY: 125 MHz, 25 MHz or 2.5 MHz.
RGMII_RXD_#	Input	Receives the lower 4 bits of packet data ([3:0]) on the rising edge, and the upper 4 bits of packet data ([7:4]) on the falling edge of RGMII_RXC_#.
RGMII_RX_CTL_#(1)	Input	This signal is RXDV on the rising edge of RGMII_RXC_#, and an encoded RXERR on the falling edge.

Notes:

1. See the Hewlett Packard RGMII specification v1.3 or v2.0 (section 3.4) for more information.

10/100/1000 Serial Gigabit Media Independent Interface (SGMII)

The SGMII physical interface was defined by Cisco Systems. The data signals operate at a rate of 1.25 Gb/s, and the sideband clock signals operate at 625 MHz. Due to the speed of these signals, differential pairs are used to provide signal integrity and minimize noise.

The sideband clock signals are not implemented in the Ethernet MAC. Instead, the MGT is used to transmit and receive data with clock data recovery (CDR).

When using SGMII mode, the MGT must be instantiated in the design to connect with the Ethernet MAC. For more information on SGMII, refer to the Serial GMII specification v1.7.

This interface only supports full-duplex operation.

SGMII RX Elastic Buffer

The RX elastic buffer can be implemented in one of two ways:

- Using the buffer present in the MGTs
- Using a larger buffer that is implemented in the FPGA logic

This section describes the selection and implementation of two options in the following subsections:

- "Using the MGT RX Elastic Buffer"
- "Using the FPGA Logic Elastic Buffer"

RX Elastic Buffer Implementations

Selecting the Buffer Implementation from the GUI

The GUI for the Ethernet MAC provides two alternative options under the heading "SGMII Capabilities." These options are:

1. **10/100/1000 Mb/s (clock tolerance compliant with Ethernet specification)**.

   Default setting; provides the implementation using the RX elastic buffer in FPGA fabric. This alternative RX elastic buffer utilizes a single block RAM to create a buffer twice as large as the one present in the RocketIO™ transceivers, subsequently consuming extra logic resources. However, this default mode provides reliable SGMII operation under all conditions.

2. **10/100/1000 Mb/s (restricted tolerance for clocks) OR 100/1000 Mb/s**.

   Uses the RX elastic buffer present in the RocketIO transceivers. This is half the size and can potentially under- or overflow during SGMII frame reception at 10 Mb/s operation. However, there are logical implementations where this can be proven reliable; if so, it is favored because of its lower logic utilization.

Option 1, the default mode, provides the implementation using the RX elastic buffer in the FPGA logic. This alternative RX elastic buffer utilizes a single block RAM to create a buffer that is twice as large as the one present in the MGT, therefore consuming extra logic resources. However, this default mode provides reliable SGMII operation.

Option 2 uses the RX elastic buffer in the MGTs. This buffer, which is half the size of the buffer in Option 1, can potentially underflow or overflow during SGMII frame reception at 10 Mb/s operation (see "The FPGA RX Elastic Buffer Requirement"). However, in logical implementations where this case is proven reliable, this option is preferred because of its lower logic utilization.

The FPGA RX Elastic Buffer Requirement

Figure 4-17 illustrates a simplified diagram of a common situation where the Ethernet MAC in SGMII mode is interfaced to an external PHY device. Separate oscillator sources are used for the FPGA and the external PHY. The Ethernet specification uses clock sources with a 100 ppm tolerance. In Figure 4-17, the clock source for the PHY is slightly faster than the clock source to the FPGA. Therefore, during frame reception, the RX elastic buffer (implemented in the MGT in this example) starts to fill up.

Following frame reception in the interframe gap period, idles are removed from the received datastream to return the RX elastic buffer to half-full occupancy. This task is performed by the clock correction circuitry (see UG076, *Virtex-4 RocketIO Multi-Gigabit Transceiver User Guide*).

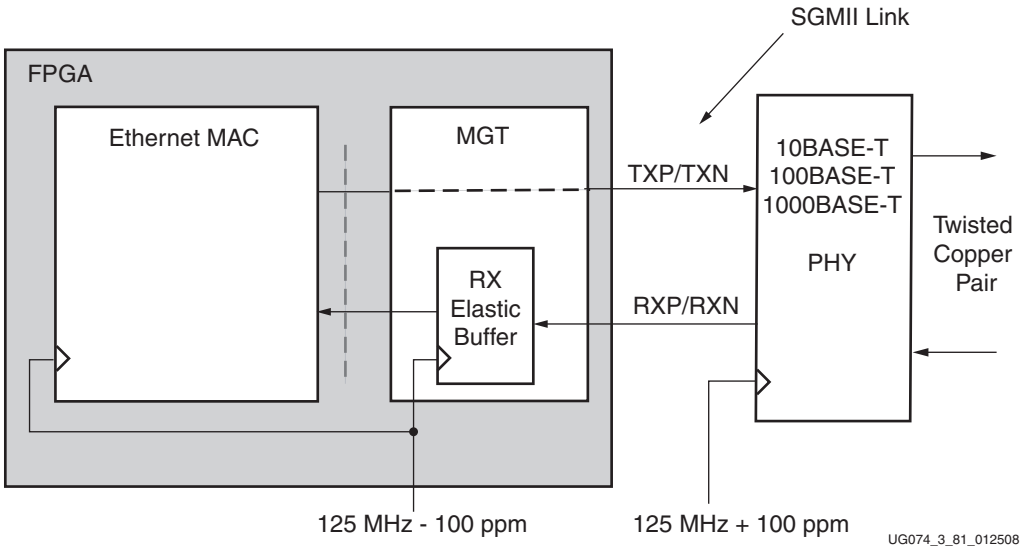


Figure 4-17: **SGMII Implementation Using Separate Clock Sources**

Assuming separate clock sources, each with 100 ppm tolerance, the maximum frequency difference between the two devices can be 200 ppm, which translates into a full clock period difference every 5000 clock periods.

Relating this information to an Ethernet frame, there is a single byte of difference every 5000 bytes of received frame data, causing the RX elastic buffer to either fill or empty by an occupancy of one.

The maximum Ethernet frame (non-jumbo) has a 1522-byte size for a VLAN frame:

- At 1 Gb/s operation, this size translates into 1522 clock cycles
- At 100 Mb/s operation, this size translates into 15220 clock cycles (because each byte is repeated 10 times
- At 10 Mb/s operation, this size translates into 152200 clock cycles (because each byte is repeated 100 times)

Considering the 10 Mb/s case, 152200/5000 = 31 FIFO entries are needed in the elastic buffer above and below the halfway point to insure that the buffer does not underflow or overflow during frame reception. This assumes that frame reception begins when the buffer is exactly half full.

MGT Elastic Buffer (Ring Buffer)

The RX ring buffer in the MGTs has 64 entries, and acts as an elastic buffer. Upon initialization or reset, the buffer goes to half full. However, the buffer cannot be assumed to be exactly half full at the start of frame reception. Additionally, the underflow and overflow thresholds are not exact.

Figure 4-18 illustrates the elastic buffer depths and thresholds of MGTs in Virtex-4 devices. Each FIFO word corresponds to a single character of data (equivalent to a single byte of data following 8B/10B decoding).

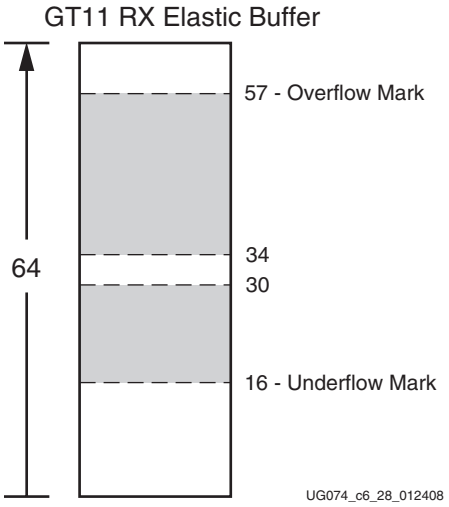


Figure 4-18: **Elastic Buffer Size for MGTs**

The shaded area represents the usable buffer for the duration of frame reception.

- If the buffer is filling during frame reception, then there are 57 – 34 = 23 FIFO locations available before the buffer hits the overflow mark.
- If the buffer is emptying during reception, then there are 30 – 16 = 14 FIFO locations available before the buffer hits the underflow mark.

This analysis assumes that the buffer is approximately at the half-full level at the start of the frame reception. There are, as illustrated, two locations of uncertainty above and below the exact half-full mark of 32. The uncertainty is a result of the clock correction decision, which is performed in a different clock domain.

Since there is a worst case scenario of one clock edge difference every 5000 clock periods, the maximum number of clock cycles (bytes) that can exist in a single frame passing through the buffer before an error occurs is 5,000 x 14 = 70,000 bytes.

Table 4-4 translates this into maximum frame lengths at different Ethernet MAC speeds. The situation is worse at SGMII speeds lower than 1 Gb/s because bytes are repeated multiple times.

Table 4-4: **Maximum Frame Sizes for MGT RX Elastic Buffers (100 ppm Clock Tol.)**

Standard	Speed	Maximum Frame Size
1000BASE-X	1 Gb/s only	70,000
SGMII	1 Gb/s	70,000

Table 4-4: **Maximum Frame Sizes for MGT RX Elastic Buffers (100 ppm Clock Tol.)**

Standard	Speed	Maximum Frame Size
SGMII	100 Gb/s	7,000
SGMII	10 Gb/s	700

FPGA Logic Elastic Buffer

For reliable SGMII operation at 10 Mb/s (non-jumbo frames), the MGT RX elastic buffer must be bypassed and a larger buffer implemented in the FPGA logic. The RX elastic buffer in FPGA logic, provided by the example design, is twice the size and nominally provides 64 entries above and below the half-full threshold. This configuration can manage standard (non-jumbo) Ethernet frames at all three SGMII speeds.

Figure 4-19 illustrates alternative FPGA logic RX elastic buffer depth and thresholds. Each FIFO word corresponds to a single character of data (equivalent to a single byte of data following 8B/10B decoding). This buffer can optionally be used to replace the RX elastic buffers of the MGT. See "Using the FPGA Logic Elastic Buffer," page 128.

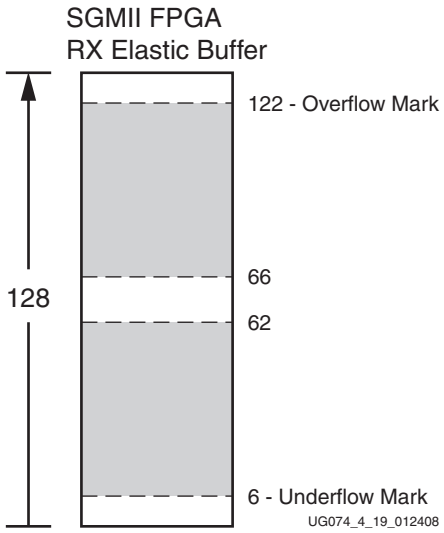


Figure 4-19: **Elastic Buffer Size for FPGA Logic Buffer**

The shaded area in Figure 4-19 represents the usable buffer availability for the duration of frame reception.

- If the buffer is filling during frame reception, then there are 122 – 66 = 56 FIFO locations available before the buffer hits the overflow mark.
- If the buffer is emptying during reception, then there are 62 – 6 = 56 FIFO locations available before the buffer hits the underflow mark.

This analysis assumes that the buffer is approximately at the half-full level at the start of the frame reception. As illustrated, there are two locations of uncertainty above and below the exact half-full mark of 64 as a result of the clock correction decision, which is based across an asynchronous boundary.

Since there is a worst-case scenario of one clock edge difference every 5000 clock periods, the maximum number of clock cycles (bytes) that can exist in a single frame passing through the buffer before an error occurs is 5000 x 56 = 280000 bytes.

Table 4-5 translates this into maximum frame lengths at different Ethernet MAC speeds. The situation is worse at SGMII speeds lower than 1 Gb/s because bytes are repeated multiple times.

Table 4-5: **Maximum Frame Sizes for FPGA Logic RX Elastic Buffers (100 ppm Clock Tol.)**

Standard	Speed	Maximum Frame Size
1000BASE-X	1 Gb/s only	280,000
SGMII	1 Gb/s	280,000
SGMII	100 Mb/s	28,000
SGMII	10 Mb/s	2,800

Using the MGT RX Elastic Buffer

The RX elastic buffer implemented in the MGT can be used reliably when:

- 10 Mb/s operation is not required. Both 1 Gb/s and 100 Mb/s operate on standard Ethernet MAC frame sizes only.
- When the clocks are closely related (see "Closely Related Clock Sources").

Designers are recommended to select the FPGA Logic RX elastic buffer implementation if they have any doubts to reliable operation.

Closely Related Clock Sources

Two cases are described with closely related clocks in SGMII mode.

- **Case 1**

  Figure 4-20 illustrates a simplified diagram of a common situation where the Ethernet MAC in SGMII mode is interfaced to an external PHY device. A common oscillator source is used for both the FPGA and the external PHY.

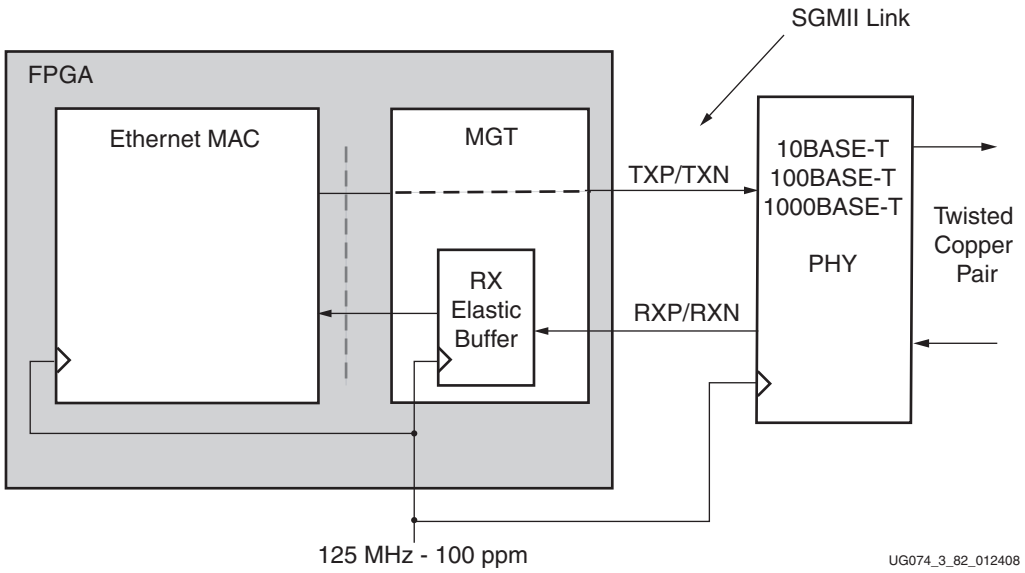


Figure 4-20: **SGMII Implementation Using Shared Clock Sources**

If the PHY device sources the receiver SGMII stream synchronously from the shared oscillator (refer to the PHY data sheet), the MGT receives data at exactly the same rate as that used by the core. That is, the RX elastic buffer neither empties nor fills because the same frequency clock is on either side.

In this situation, the RX elastic buffer does not underflow or overflow, and the RX elastic buffer implementation in the MGT is recommended to save logic resources.

- **Case 2**

  Using the case illustrated by Figure 4-17, assume that both clock sources used are 50 ppm. The maximum frequency difference between the two devices is 100 ppm, translating into a full clock period difference every 10000 clock periods and resulting in a requirement for 16 FIFO entries above and below the half-full point. This case provides reliable operation with the MGT RX elastic buffers. However, the designer must check the PHY data sheet to ensure that the PHY device sources the receiver SGMII stream synchronously to its reference oscillator.

Using the FPGA Logic Elastic Buffer

Figure 4-21 illustrates a simplified diagram of a situation where the Ethernet MAC in SGMII mode is interfaced to an external PHY device with an independent clock. The MGT's elastic buffer has been bypassed and the FPGA elastic buffer is used.

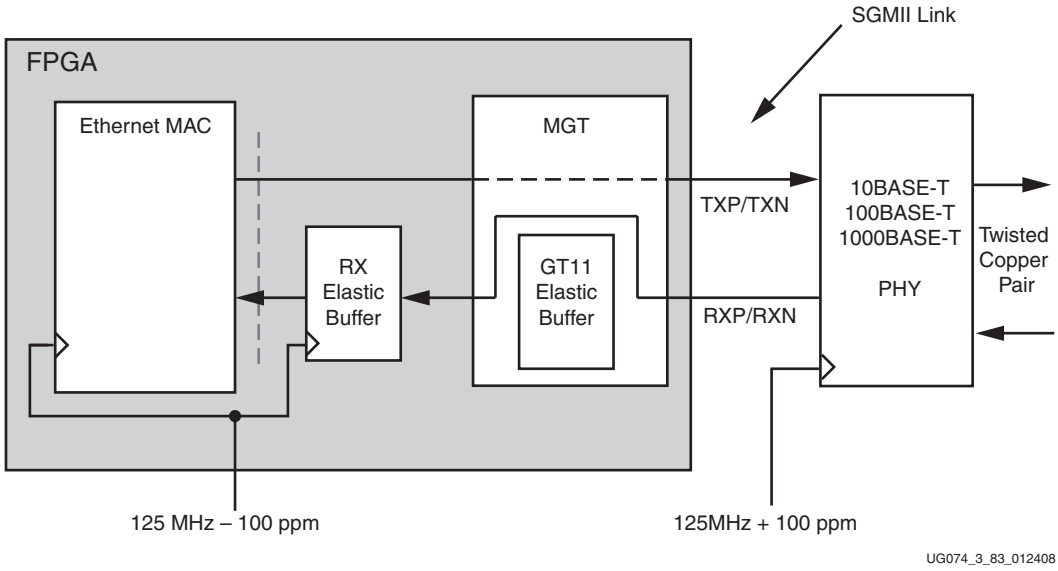


Figure 4-21: **SGMII Implementation Using a Logic Buffer**

Using the SGMII in this configuration eliminates the possibility of buffer error if the clocks are not tightly controlled enough to use the MGT elastic buffer.

MGT Logic Using the MGT RX Elastic Buffer

Figure 4-23 shows the Ethernet MAC configured with SGMII as the physical interface. Connections to the RocketIO transceiver are illustrated.

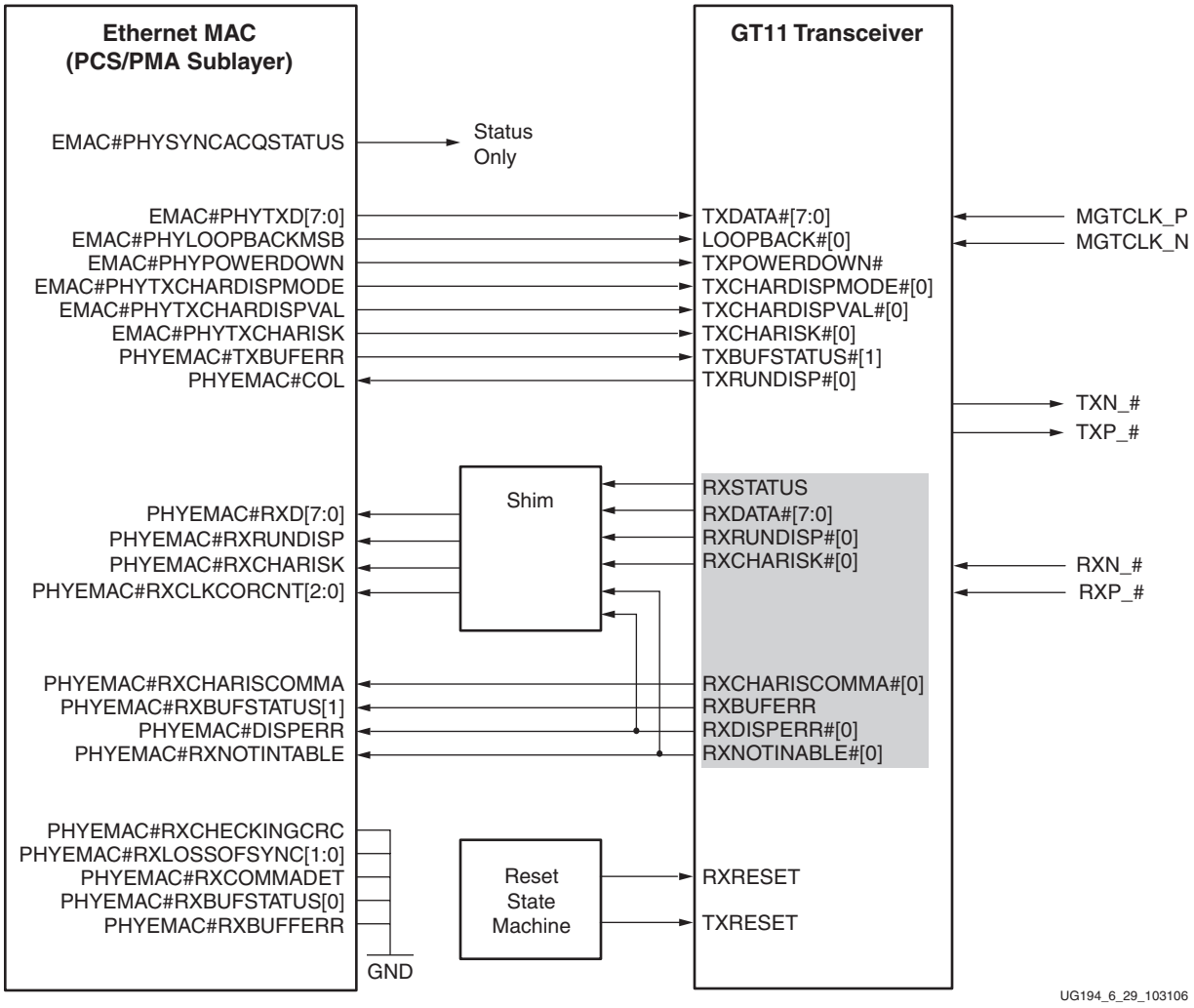


Figure 4-22: **Ethernet MAC Configured in SGMII Mode**

These connections and the shim are created by the CORE Generator tool when the physical interface is selected to be either SGMII or 1000BASE-X PCS/PMA. By using the CORE Generator tool, the time required to instantiate the Ethernet MAC into a usable design is greatly reduced.

10/100/1000 SGMII Interface

Figure 4-23 shows the Ethernet MAC configured with SGMII as the physical interface. In this interface, not all the ports of the Ethernet MAC are used.

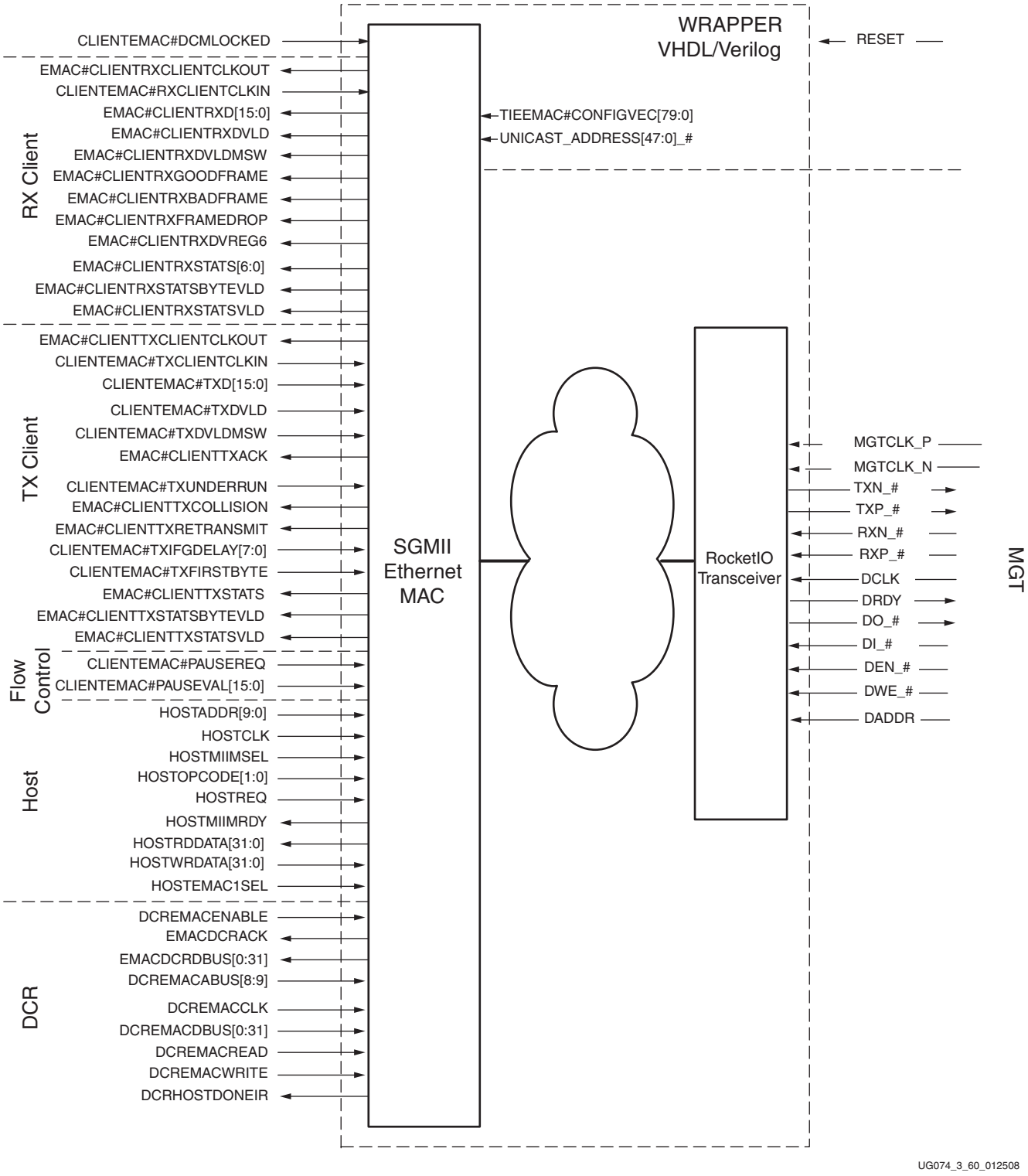


Figure 4-23: **Ethernet MAC Configured in SGMII Mode**

10/100/1000 SGMII Clock Management

Figure 4-24 shows the clock management used with the SGMII interface. At a line rate of 1.25 Gb/s or below, oversampling is used by the built-in MGT digital receiver to recover clock and data. Chapter 3 of UG076, *Virtex-4 RocketIO Multi-Gigabit Transceiver User Guide* provides more details about the digital receiver oversampling operation. The inputs of the GT11CLK_MGT primitive connect to an external, high-quality reference clock with a frequency of 250 MHz specifically for the MGT. The output SYNCLK1OUT connects to the PLL reference clock input REFCLK1. TXOUTCLK1 is derived from the transmitter PLL. TXOUTCLK1 feeds TXUSRCLK2 and the PHYEMAC#GTXCLK. RXRECCLK1 feeds a BUFR that is used to clock the internal elastic buffer. This buffer is only necessary for SGMII. The output of the BUFR also drives RXUSRCLK2. RXUSRCLK and TXUSRCLK are both tied to ground.

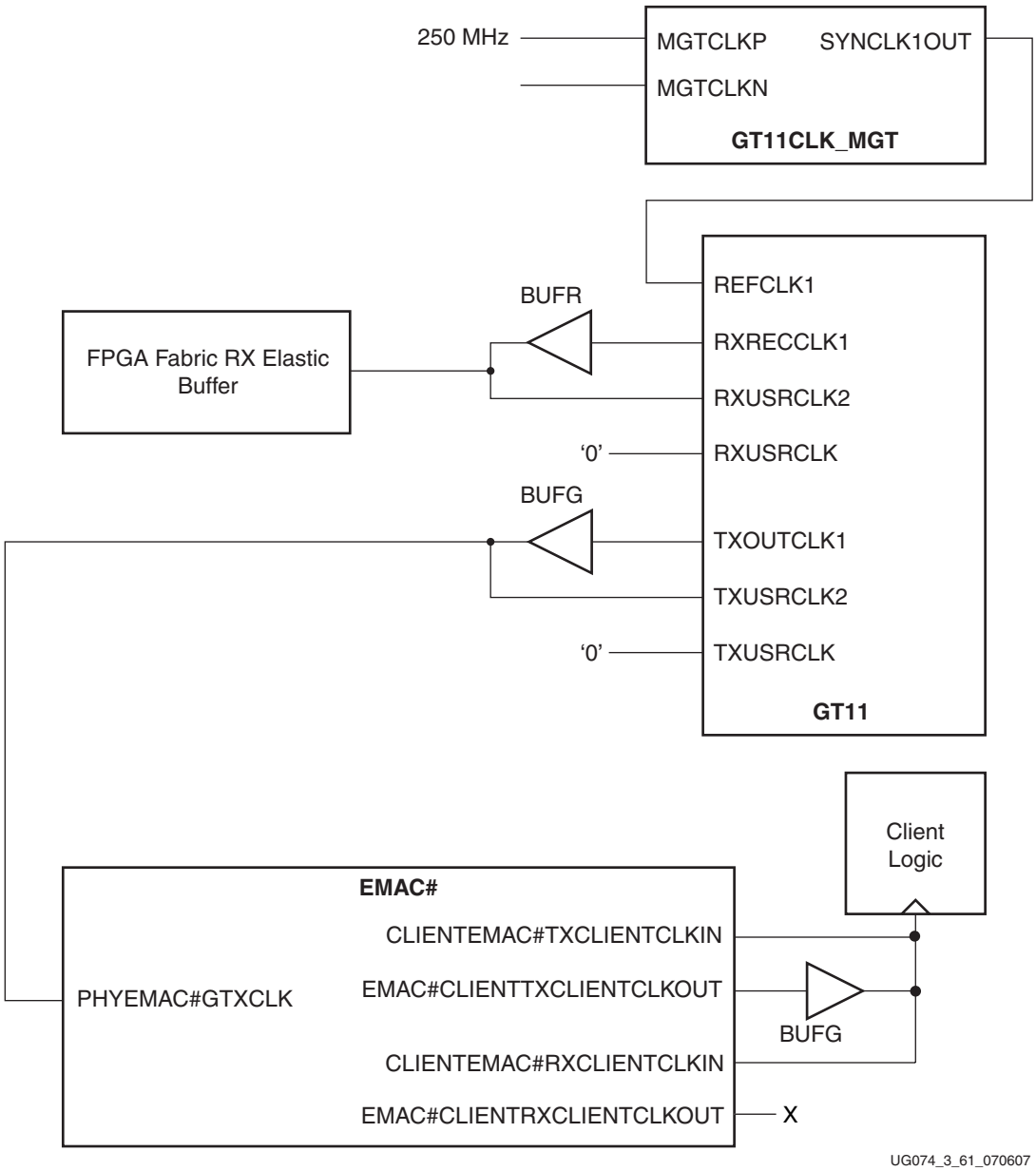


Figure 4-24: **SGMII Clock Management**

To ensure that the Ethernet MAC does not operate until the MGT has achieved all necessary locks, the CLIENTEMAC#DCMLOCKED input signal to the EMAC# block is generated using the TXLOCK and RXLOCK signals from the MGT, and the DCM LOCKED output. Refer to the CORE Generator Ethernet MAC wrapper for the actual implementation of this combined lock signal.

The EMAC#CLIENTRXCLIENTCLKOUT output port must be connected to a BUFG to drive the receive client logic in the FPGA fabric, and then is routed back into the input ports CLIENTEMAC#RXCLIENTCLKIN and CLIENTEMAC#TXCLIENTCLKIN. This clock is also used for the transmit client logic.

SGMII Signals

An Ethernet MAC wrapper has all necessary pin connections to configure the primitive into SGMII. Table 4-6 describes the 10/100/1000 SGMII interface signals.

Table 4-6: **10/100/1000 SGMII and 1000BASE-X PCS/PMA Interface Signals**

Signal	Direction	Description
ENCOMMAALIGN_#	Output	Enable PMA layer (MGT) to realign to commas.
LOOPBACKMSB_#	Output	Loopback tests within the MGTs.
MGTRXRESET_#	Output	Reset to receive PCS of MGT.
MGTTXRESET_#	Output	Reset to transmit PCS of MGT.
POWERDOWN_#	Output	Power down the MGTs.
SYNC_ACQ_STATUS_#	Output	The output from the receiver's synchronization state machine of IEEE Std 802.3, Clause 36. When asserted High, synchronization on the received bitstream is obtained. The state machine is in one of the SYNC_AQUIRED states of IEEE Std 802.3 Figure 36-9. When deasserted Low, synchronization is not yet obtained.
TXCHARDISPMODE_#	Output	Set running disparity for current byte.
TXCHARDISPVAL_#	Output	Set running disparity value.
TXCHARISK_#	Output	K character transmitted in TXDATA.
PHYAD[4:0]_#	Input	PHY address of MDIO register set for the PCS sublayer.
RXBUFSTATUS[1:0]_#	Input	Receiver elastic buffer status: Bit [1] asserted indicates over flow or underflow.
RXCHARISCOMMA_#	Input	Comma detected in RXDATA.
RXCHARISK_#	Input	K character received or extra data bit in RXDATA. Becomes the 10th bit in RXDATA when RXNOTINTABLE is asserted.
RXCHECKINGCRC_#	Input	Reserved, tie to GND.
RXBUFERR_#	Input	Reserved, tie to GND.
RXCOMMADET_#	Input	Reserved, tie to GND.
RXDISPERR_#	Input	Disparity error in RXDATA.
RXLOSSOFSYNC[1:0]_#	Input	Reserved, tie to GND.

Table 4-6: **10/100/1000 SGMII and 1000BASE-X PCS/PMA Interface Signals *(Cont'd)***

Signal	Direction	Description
RXNOTINTABLE_#	Input	Indicates non-existent 8B/10 code.
RXRUNDISP _#	Input	Running disparity in the received serial data. When RXNOTINTABLE is asserted in RXDATA, this signal becomes the ninth data bit.
RXCLKCORCNT[2:0]_#	Input	Status denoting the occurrence of clock correction.
SIGNAL_DETECT _#	Input	Signal direct from PMD sublayer indicating the presence of light detected at the optical receiver, as defined in IEEE Std 802.3, Clause 36. If asserted High, the optical receiver has detected light. When deasserted Low this indicates the absence of light. If unused, this signal should be tied High to enable correct operation the Ethernet MAC.
TXBUFERR_#	Input	TX buffer error (overflow or underflow).
CLIENTEMAC#DCMLOCKED	Input	If a DCM is used to derive any of the clock signals going to the Ethernet MAC, the locked port of the DCM must be connected to the CLIENTEMAC#DCMLOCKED port of the Ethernet MAC. The Ethernet MAC is held in reset until CLIENTEMAC#DCMLOCKED is driven to logic 1. If DCM is not used, tie this port to a logic 1.
DADDR	Input	Dynamic configuration address bus.
DO_#	Output	Configuration output data bus.
DRDY	Output	Strobe that indicates read/write cycle is complete.
DCLK	Input	Dynamic configuration bus clock.
DI_#	Input	Dynamic configuration input data bus.
DEN_#	Input	Dynamic configuration bus enable when set to a logic 1.
DWE_#	Input	Dynamic configuration write enable when set to a logic 1.

Management Registers

SGMII has a similar PCS sublayer managed register block defined in IEEE Std 802.3-2002 Clause 37. This set of 10 management registers, accessed through MDIO, is described in the 1000BASE-X PCS/PMA section ("Management Registers," page 140). However, the auto-negotiation Advertisement Register (register 4) and the auto-negotiation Link Partner Ability BASE Register (register 5) are different in the SGMII specification. Table 4-7 and Table 4-8 describe these two registers with regard to the SGMII specification.

Table 4-7: **SGMII Auto-Negotiation Advertisement Register (Register 4)**

Bit(s)	Name	Description	Attributes	Default Value
4.15:0	All bits	SGMII defined value sent from the MAC to the PHY.	Read Only	0000000000000001

Table 4-8: **SGMII Auto-Negotiation Link Partner Ability Base Register (Register 5)**

Bit(s)	Name	Description	Attributes	Default Value
5.15	Link up/down	1 = Link up. 0 = Link down.	Read only	1
5.14	Acknowledge	Used by the auto-negotiation function to indicate reception of a link partner's base or next page.	Read only	0
5.13	Reserved	Always return 0.	Returns 0	0
5.12	Duplex mode	1 = Full Duplex. 0 = Half Duplex.	Read only	00
5.11:10	Speed	00 = 10 Mb/s. 01 = 100 Mb/s. 10 = 1000 Mb/s. 11 = Reserved.	Read only	00
5.9:1	Reserved	Always returns 0s.	Returns 0s	000000000
5.0	Reserved	Always returns 1.	Returns 1	1

1000BASE-X PCS/PMA

The full-duplex PCS with PMA interface for 1000BASE-X is defined in IEEE Std 802.3-2002, Clauses 36 and 37.

Figure 4-25 shows a block diagram of the PCS and PMA sublayers. The functional blocks of the PCS and PMA sublayers can replace the GMII interface. The functional blocks of the PCS and PMA sublayers are:

- Transmit engine
- Auto-negotiation block
- Receive engine and synchronization block
- PCS management register block
- RocketIO transceiver

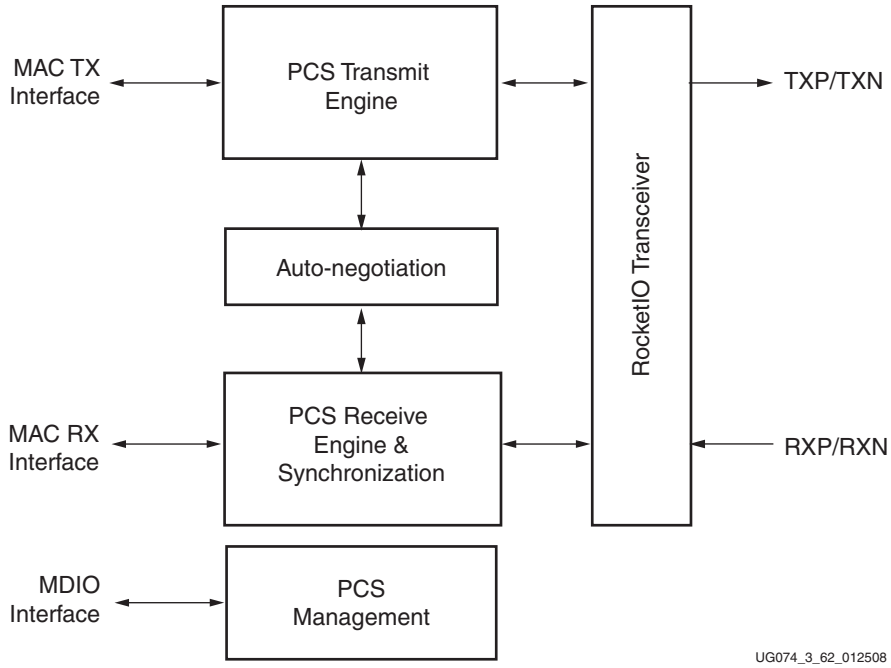


Figure 4-25: **Ethernet PCS/PMA Sublayer Extension**

The management register block in the PCS sublayer is accessed through the MDIO interface as though there is an externally connected PHY. This block is used to configure the operation of the PCS sublayer, the PMA sublayer, and auto-negotiation. The host interface can control the MDIO interface as defined in "MDIO Interface" in Chapter 3. When the management interface is not present, the PCS sublayer management register block must be accessed using a separate MDIO controller outside the Ethernet MAC.

The MGT provides some of the PCS layer functionality including 8B/10B encoding and decoding, and serialization and deserialization in the PMA layer.

1000BASE-X PCS/PMA Interface

Figure 4-26 shows the Ethernet MAC configured with 1000BASE-X PCS/PMA as the physical interface. The client interfaces can be either 8 bits or 16 bits wide. In 16-bit client mode, the Ethernet MAC can operate at 250 MHz, enabling a 2.5 Gb/s line rate. In this interface, not all the ports of the Ethernet MAC are used.

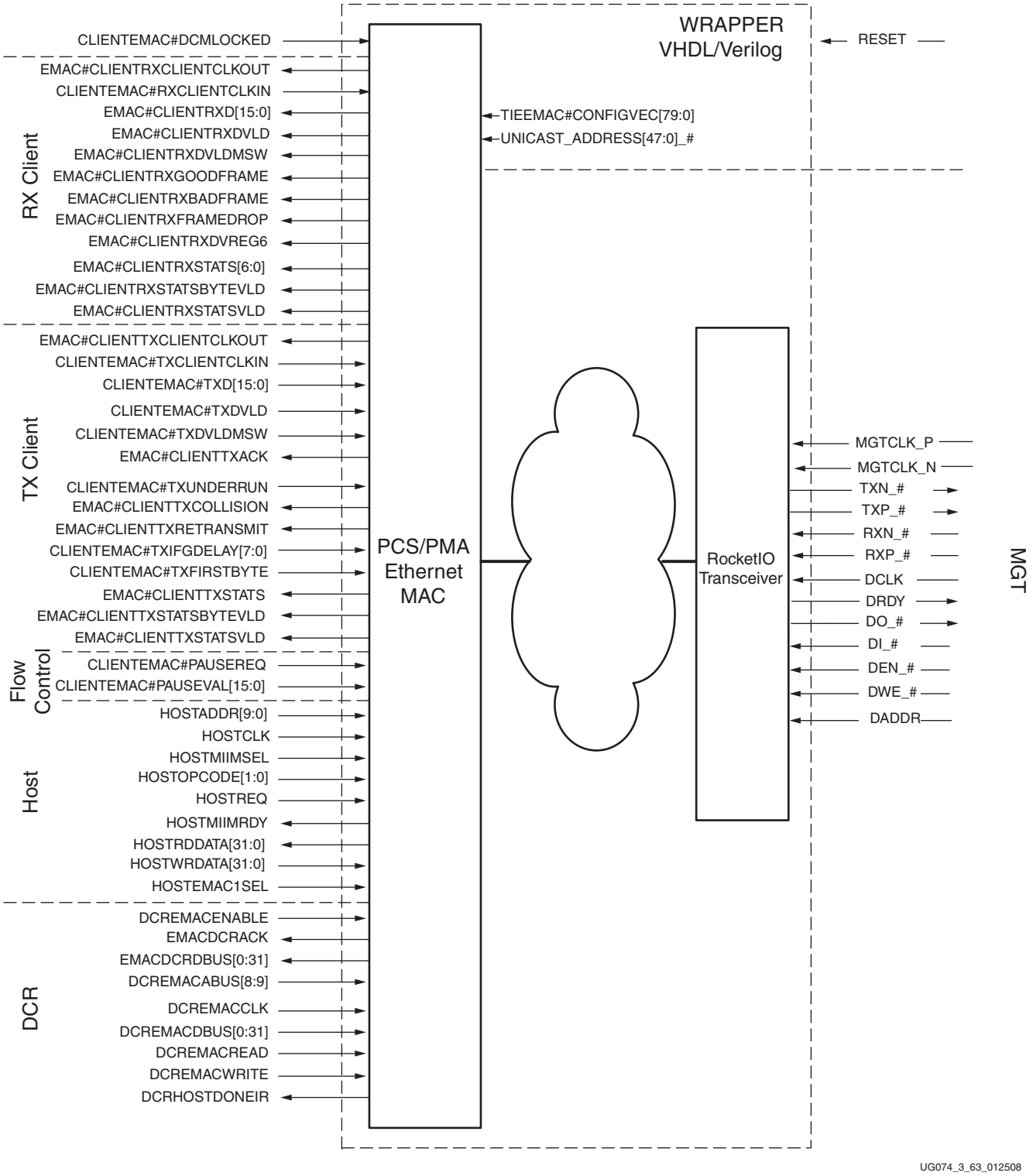


Figure 4-26: **Ethernet MAC Configured in 1000BASE-X PCS/PMA Mode**

Shim

Because of small differences in the way the Virtex-II Pro and Virtex-4 FPGA RocketIO transceivers output the clock correction status and data when RXNOTINTABLE is asserted, a shim is needed to modify the received data from the Virtex-4 FPGA RocketIO transceiver (GT11) to the format that the EMAC is expecting. Table 4-9 shows the ports of this shim created by the CORE Generator tool for the SGMII and 1000BASE-X designs.

Table 4-9: **Shim Port Names**

Port Name	I/O	Width	Description
rxusrclk2	I	1	Logic Clock.
Rxstatus	I	6	Status from GT11.
Rxnotintable	I	1	Status from GT11.
rxd_in	I	8	Data from GT11.
rxcharisk_in	I	1	Control from GT11.
rxrundisp_in	I	1	Status from GT11.
Rxclkcorcnt	O	3	Status in GT format.
rxd_out	O	8	Data in GT format when rxnotintable asserted.
rxcharisk_out	O	1	Control in GT/GT11 format.
rxrundisp_out	O	1	Control in GT/GT11 format.

1000BASE-X PCS/PMA Clock Management

8-Bit Data Client

Figure 4-27 shows the clock management used with the 1000BASE-X PCS/PMA interface and an 8-bit data client. At a line rate of 1.25 Gb/s or below, oversampling is used by the built-in MGT digital receiver to recover clock and data. See Chapter 3 of UG076, *Virtex-4 RocketIO Multi-Gigabit Transceiver User Guide* for more details about the digital receiver oversampling operation. The inputs of the GT11CLK_MGT primitive are connected to an external, high-quality reference clock with a frequency of 250 MHz specifically for the MGT. The output SYNCLKOUT connects to the PLL reference clock input REFCLK. TXOUTCLK1, derived from the transmitter PLL, reflects the reference clock and drives the other clock inputs. It connects through a global buffer to PHYEMAC#GTXCLK and into TXUSRCLK2 and RXUSRCLK2. TXUSRCLK and RXUSRCLK are not used and are tied to ground.

To ensure that the Ethernet MAC does not operate until the MGT has achieved all necessary locks, the CLIENTEMAC#DCMLOCKED input signal to the EMAC# block is generated using the TXLOCK and RXLOCK signals from the MGT, and the DCM LOCKED output. Refer to the CORE Generator Ethernet MAC wrapper for the actual implementation of this combined lock signal.

The EMAC#CLIENTTXCLIENTCLKOUT output port must be connected to a BUFG to drive the transmit client logic in the FPGA fabric, and then is routed back into the input port CLIENTEMAC#TXCLIENTCLKIN. This method is also used for the receive client logic.

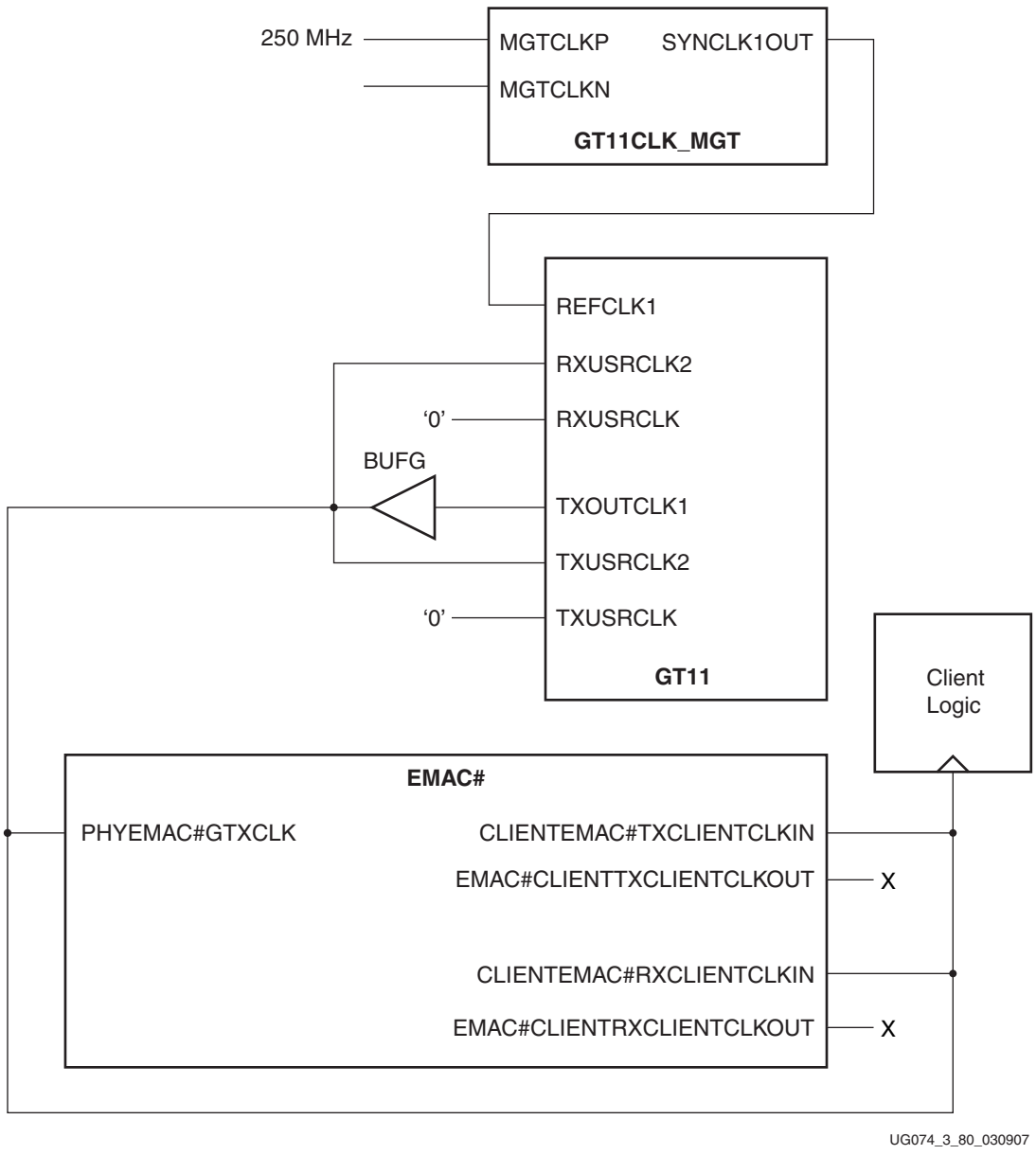


Figure 4-27: **1000BASE-X PCS/PMA (8-bit Data Client) Clock Management**

16-Bit Data Client

Figure 4-28 shows the clock management used with the 1000BASE-X PCS/PMA interface and a 16-bit data client. This mode supports 1.25 Gb/s and 2.5 Gb/s line rates. For a 2.5 Gb/s line rate, clock and data are recovered from the incoming data stream.

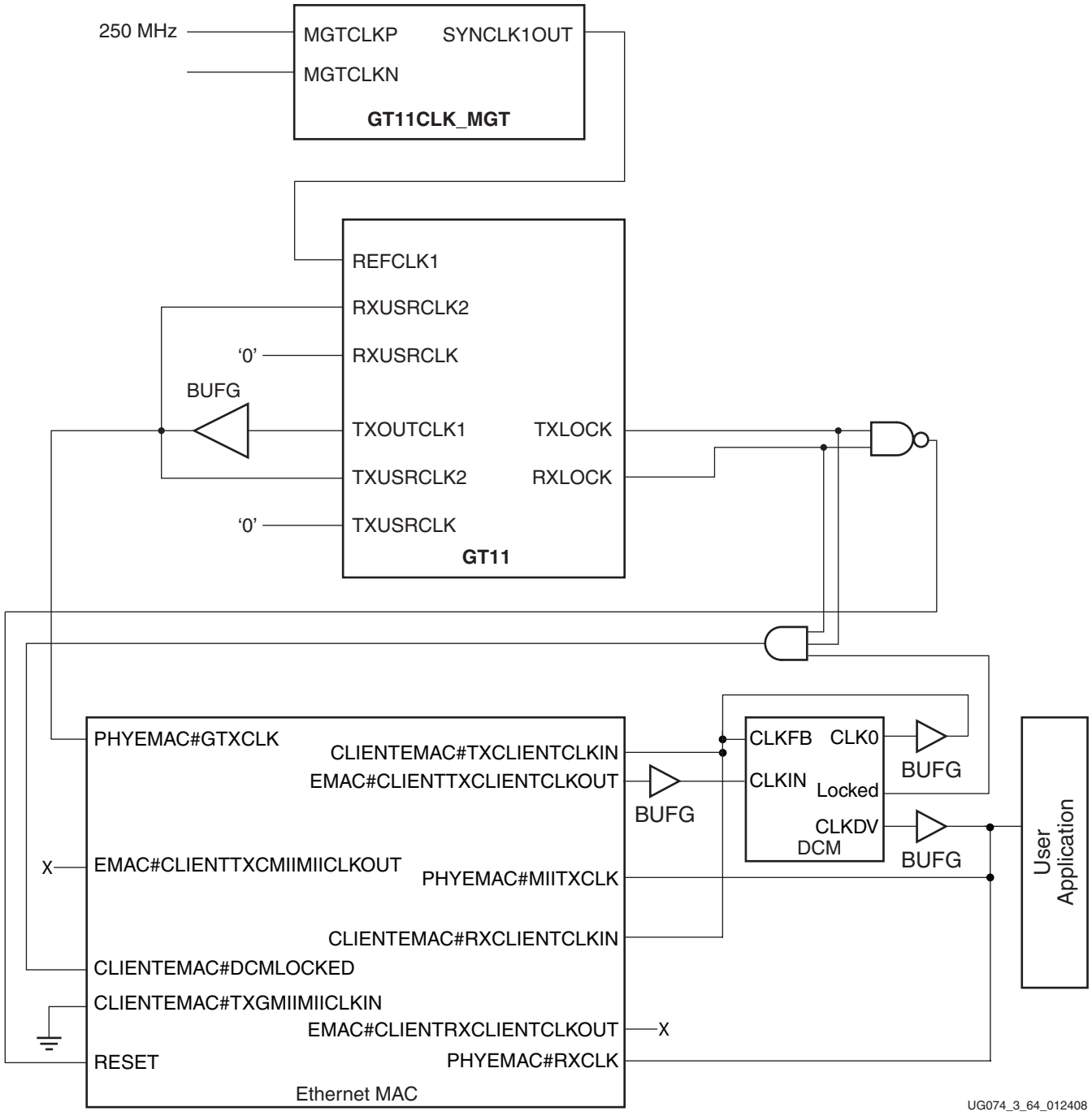


Figure 4-28: **1000BASE-X PCS/PMA (16-Bit Data Client) Clock Management**

The inputs of the GT11CLK_MGT primitive connect to an external, high-quality reference clock with a frequency of 250 MHz for 1.25 Gb/s and 2.5 Gb/s line rates. The output SYNCLK1OUT connects to the PLL reference clock input REFCLK1. TXOUTCLK1,

derived from the transmitter PLL, reflects the reference clock and drives the other clock inputs. It connects through a global buffer to PHYEMAC#GTXCLK and into TXUSRCLK2 and RXUSRCLK2. TXUSRCLK and RXUSRCLK are not used and are tied to ground.

Because no GMII logic is in the 1000BASE-X PCS/PMA mode of configuration, the EMAC#CLIENTTXGMIIMIICLKOUT output port is not connected, and the input port CLIENTEMAC#TXGMIIMIICLKIN is connected to ground.

The EMAC#CLIENTTXCLIENTCLKOUT output port must be connected to the DCM CLKIN input. The CLKDV output of the DCM, with the CLKDV_DIVIDE attribute set to two, is routed to a BUFG and is connected to the transmit client logic and to PHYEMAC#MIITXCLK. The CLK0 output of the DCM is routed to a BUFG and is connected to the input port CLIENTEMAC#TXCLIENTCLKIN. This arrangement reduces the clock rate on the 16-bit client interface to half of the PHYEMAC#GTXCLK. The clocks generated by this DCM can also be shared with the receive clocks CLIENTEMAC#RXCLIENTCLKIN and PHYEMAC#RXCLK as shown in Figure 4-28.

TXLOCK and RXLOCK from the MGT and the locked signals from the client interface DCM are ANDed together to generate a combined lock signal for CLIENTEMAC#DCMLOCKED. The lock signal ensures the Ethernet MAC does not operate until the MGT has achieved all the necessary locks.

The phase-matched clock divider (PMCD) feature of certain Virtex-4 devices can be used to replace the client interface DCM.

PCS/PMA Signals

An Ethernet MAC wrapper has all the necessary pin connections to configure the primitive into 1000BASE-X PCS/PMA. Table 4-6 also describes the 1000BASE-X PCS/PMA interface signals.

Management Registers

The PCS in the 1000BASE-X PCS/PMA sublayer contains the full managed register block defined in IEEE Std 802.3-2002, Clause 37. This utilizes 10 dedicated management registers, accessed from the MDIO interface. Table 4-10 to Table 4-19 define these 10 registers.

Table 4-10: **Control Register (Register 0)**

Bit(s)	Name	Description	Attributes	Default Value
0.15	Reset	1 = PCS/PMA reset. 0 = Normal operation.	Read/Write Self Clearing	TIEEMAC#CONFIGVEC[78]
0.14	Loopback	1 = Enable loopback mode. 0 = Disable loopback mode.	Read/Write	TIEEMAC#CONFIGVEC[74]
0.13	Speed Selection (LSB)	The Ethernet MAC always returns a 0 for this bit. Along with bit 0.6, speed selection of 1000 Mb/s is identified.	Returns 0	0
0.12	Auto-Negotiation Enable	1 = Enable auto-negotiation process. 0 = Disable auto-negotiation process.	Read/Write	TIEEMAC#CONFIGVEC[77]

Table 4-10: **Control Register (Register 0) *(Cont'd)***

Bit(s)	Name	Description	Attributes	Default Value
0.11	Power Down	1 = Power down. 0 = Normal operation. When set to 1, the MGT is placed in a Low power state. This bit requires a reset (see bit 0.15) to clear.	Read/ Write	TIEEMAC#CONFIGVEC[75]
0.10	Isolate	1 = Electrically isolate the PHY from GMII. 0 = Normal operation.	Read/Write	TIEEMAC#CONFIGVEC[76]
0.9	Restart Auto-Negotiation	1 = Restart auto-negotiation process. 0 = Normal operation.	Read/Write Self Clearing	0
0.8	Duplex Mode	The Ethernet MAC always returns a 1 for this bit to signal full-duplex mode.	Returns 1	1
0.7	Collision Test	The Ethernet MAC always returns a 0 for this bit to disable COL test.	Returns 0	0
0.6	Speed Selection (MSB)	The Ethernet MAC always returns a 1 for this bit. Together with bit 0.13, speed selection of 1000 Mb/s is identified.	Returns 1	1
0.5:0	Reserved	Always returns zeros, writes ignored.	Returns 0s	000000

Table 4-11: **Status Register (Register 1)**

Bit(s)	Name	Description	Attributes	Default Value
1.15	100BASE-T4	The Ethernet MAC always returns a 0 for this bit because 100BASE-T4 is not supported.	Returns 0	0
1.14	100BASE-X Full Duplex	The Ethernet MAC always returns a 0 for this bit because 100BASE-X full duplex is not supported.	Returns 0	0
1.13	100BASE-X Half Duplex	The Ethernet MAC always returns a 0 for this bit because 100BASE-X half duplex is not supported.	Returns 0	0
1.12	10 Mb/s Full Duplex	The Ethernet MAC always returns a 0 for this bit because 10 Mb/s full duplex is not supported.	Returns 0	0
1.11	10 Mb/s Half Duplex	The Ethernet MAC always returns a 0 for this bit because 10 Mb/s half duplex is not supported.	Returns 0	0
1.10	100BASE-T2 Full Duplex	The Ethernet MAC always returns a 0 for this bit because 100BASE-T2 full duplex is not supported.	Returns 0	0
1.9	100BASE-T2 Half Duplex	The Ethernet MAC always returns a 0 for this bit because 100BASE-T2 half duplex is not supported.	Returns 0	0

Table 4-11: **Status Register (Register 1) *(Cont'd)***

Bit(s)	Name	Description	Attributes	Default Value
1.8	Extended Status	The Ethernet MAC always returns a 1 for this bit, indicating the presence of the extended register (Register 15).	Returns 1	1
1.7	Reserved	Always returns 0, writes ignored.	Returns 0	0
1.6	MF Preamble Suppression	The Ethernet MAC always returns a 1 for this bit to indicate the support of management frame preamble suppression.	Returns 1	1
1.5	Auto-Negotiation Complete	1 = Auto-negotiation process completed. 0 = Auto-negotiation process not completed.	Read Only	0
1.4	Remote Fault	1 = Remote fault condition detected. 0 = No remote fault condition detected.	Read Only Self Clearing on read	0
1.3	Auto-Negotiation Ability	The Ethernet MAC always returns a 1 for this bit, indicating that the PHY is capable of auto-negotiation.	Returns 1	1
1.2	Link Status(1)	1 = Link is up. 0 = Link is down. This bit latches to 0 if the link status goes down, and clears to the current link status on read.	Read Only Self Clearing on read	0
1.1	Jabber Detect	The Ethernet MAC always returns a 0 for this bit because jabber detect is not supported.	Returns 0	0
1.0	Extended Capability	The Ethernet MAC always returns a 0 for this bit because no extended register set is supported.	Returns 0	0

Notes:

1. When High, the link is valid: synchronization of the link has been obtained and Auto-Negotiation (if present and enabled) has completed. When Low, a valid link has not been established. Either link synchronization has failed or Auto-Negotiation (if present and enabled) has failed to complete. Regardless of whether Auto-Negotiation is enabled or disabled, there can be some delay to the deassertion of this signal following the loss of synchronization of a previously successful link. This is due to the Auto-Negotiation state machine which requires that synchronization is lost for an entire link timer duration before changing state. For more information, see the IEEE Std 802.3-2002 specification.

Table 4-12: **PHY Identifier (Registers 2 and 3)**

Bit(s)	Name	Description	Attributes	Default Value
2.15:0	Organizationally Unique Identifier	Organizationally Unique Identifier (OUI) from IEEE is 0x000A35.	Returns OUI(3-18)	0000000000101000
3.15:10	Organizationally Unique Identifier	Organizationally Unique Identifier (OUI) from IEEE is 0x000A35.	Returns OUI(19-24)	110101

Table 4-12: **PHY Identifier (Registers 2 and 3) *(Cont'd)***

Bit(s)	Name	Description	Attributes	Default Value
3.9:4	Manufacturer's Model Number	Always returns 0s.	Returns 0s	000000
3.3:0	Revision Number	Always returns 0s.	Returns 0s	0000

Table 4-13: **Auto-Negotiation Advertisement Register (Register 4)**

Bit(s)	Name	Description	Attributes	Default Value
4.15	Next Page	1 = Next page functionality is advertised 0 = Next page functionality is not advertised	Read/Write	0
4.14	Reserved	Always returns 0, writes ignored.	Returns 0	0
4.13:12	Remote Fault	00 = No error. 01 = Offline. 10 = Link failure. 11 = Auto-negotiation error.	Read/Write Self clearing to 00 after auto-negotiation	00
4.11:9	Reserved	Always return 0, writes ignored.	Returns 0	0
4.8:7	Pause	00 = No PAUSE. 01 = Symmetric PAUSE. 10 = Asymmetric PAUSE towards link partner. 11 = Both symmetric PAUSE and asymmetric PAUSE towards link partner.	Read/Write	11
4.6	Half Duplex	The Ethernet MAC always returns a 0 for this bit because half-duplex mode is not supported.	Returns 0	0
4.5	Full Duplex	1 = Full-duplex mode is advertised. 0 = Full-duplex mode is not advertised.	Read/Write	1
4.4:0	Reserved	Always returns 0s, writes ignored.	Returns 0s	00000

Table 4-14: **Auto-Negotiation Link Partner Ability Base Register (Register 5)**

Bit(s)	Name	Description	Attributes	Default Value
5.15	Next Page	1 = Next page functionality is supported. 0 = Next page functionality is not supported.	Read Only	0
5.14	Acknowledge	Used by the auto-negotiation function to indicate reception of a link partner's base or next page.	Read Only	0
5.13:12	Remote Fault	00 = No Error. 01 = Offline. 10 = Link Failure. 11 = Auto-Negotiation Error.	Read Only	00
5.11:9	Reserved	Always returns 0s.	Returns 0s	000
5.8:7	Pause	00 = No PAUSE. 01 = Symmetric PAUSE supported. 10 = Asymmetric PAUSE supported. 11 = Both symmetric PAUSE and asymmetric PAUSE supported.	Read Only	00
5.6	Half Duplex	1 = Half-duplex mode is supported. 0 = Half-duplex mode is not supported.	Read Only	0
5.5	Full Duplex	1 = Full-duplex mode is supported. 0 = Full-duplex mode is not supported.	Read Only	0
5.4:0	Reserved	Always returns 0s.	Returns 0s	00000

Table 4-15: **Auto-Negotiation Expansion Register (Register 6)**

Bit(s)	Name	Description	Attributes	Default Value
6.15:3	Reserved	Always returns 0s.	Returns 0s	0000000000000
6.2	Next Page Able	The Ethernet MAC always returns a 1 for this bit because the device is Next Page Able.	Returns 1	1
6.1	Page Received	1 = A new page is received. 0 = A new page is not received.	Read only. Self clearing on read.	0
6.0	Reserved	Always returns 0s.	Returns 0s	0000000

Table 4-16: **Auto-Negotiation Next Page Transmit Register (Register 7)**

Bit(s)	Name	Description	Attributes	Default Value
7.15	Next Page	1 = Additional next page(s) follow 0 = Last page.	Read/Write	0
7.14	Reserved	Always returns 0.	Returns 0	0
7.13	Message Page	1 = Message Page. 0 = Unformatted Page.	Read/Write	1
7.12	Acknowledge 2	1 = Complies with message. 0 = Cannot comply with message.	Read/Write	0
7.11	Toggle	Value toggles between subsequent pages.	Read Only	0
7.10:0	Message or Unformatted Code Field	Message code field or unformatted page encoding as dictated by 7.13.	Read/Write	00000000001 (Null Message Code)

Table 4-17: **Auto-Negotiation Next Page Receive Register (Register 8)**

Bit(s)	Name	Description	Attributes	Default Value
8.15	Next Page	1 = Additional Next Page(s) follow 0 = Last page.	Read Only	0
8.14	Acknowledge	Used by auto-negotiation function to indicate reception of a link partner's base or next page.	Read Only	0
8.13	Message Page	1 = Message Page. 0 = Unformatted Page.	Read Only	0
8.12	Acknowledge 2	1 = Complies with message. 0 = Cannot comply with message.	Read Only	0
8.11	Toggle	Value toggles between subsequent next pages.	Read Only	0
8.10:0	Message / Unformatted Code Field	Message code field or unformatted page encoding as dictated by 8.13.	Read Only	00000000000

Table 4-18: **Extended Status Register (Register 15)**

Bit(s)	Name	Description	Attributes	Default Value
15.15	1000BASE-X Full Duplex	The Ethernet MAC always returns a 1 for this bit because 1000BASE-X full duplex is supported.	Returns 1	1
15.14	1000BASE-X Half Duplex	The Ethernet MAC always returns a 0 for this bit because 1000BASE-X half duplex is not supported.	Returns 0	0
15.13	1000BASE-T Full Duplex	The Ethernet MAC always returns a 0 for this bit because 1000BASE-T full duplex is not supported.	Returns 0	0

Table 4-18: **Extended Status Register (Register 15) *(Cont'd)***

Bit(s)	Name	Description	Attributes	Default Value
15.12	1000BASE-T Half Duplex	The Ethernet MAC always returns a 0 for this bit because 1000BASE-T half duplex is not supported.	Returns 0	0
15:11:0	Reserved	Always returns 0s.	Returns 0s	000000000000

Table 4-19: **Vendor-Specific Register: Auto-Negotiation Interrupt Control Register (Register 16)**

Bit(s)	Name	Description	Attributes	Default Value
16.15:2	Reserved	Always returns 0s.	Returns 0s	00000000000000
16.1	Interrupt Status	1 = Interrupt is asserted. 0 = Interrupt is not asserted. If the interrupt is enabled, this bit is asserted upon the completion of an auto-negotiation cycle; it can only be cleared by writing 0 to this bit. If the interrupt is disabled, this bit is set to 0. The EMAC#CLIENTANINTERRUPT port is wired to this bit.	Read/Write	0
16.0	Interrupt Enable	1 = Interrupt is enabled. 0 = Interrupt is disabled.	Read/Write	1

XILINX®

Chapter 5

Miscellaneous Functions

This chapter provides useful design information for the Virtex®-4 FPGA Embedded Tri-Mode Ethernet MAC. It contains the following sections:

Clock Frequency Support

Table 5-1 through Table 5-5 summarize the supported clock frequencies of the Ethernet MAC. All clock signal output frequencies for both the transmit and receive modules are generated in the clock-generation module of the Ethernet MAC.

Table 5-1: **Transmit Clock Speeds (PHYEMAC#GTXCLK)**

Clock Signals	Direction	1000 Mb/s	100 Mb/s	10 Mb/s
PHYEMAC#GTXCLK	Input	125 MHz	125 MHz	125 MHz

Table 5-2: **Receive Clock Speeds (PHYEMAC#RXCLK)**

Clock Signals	Direction	1000 Mb/s	100 Mb/s	10 Mb/s
PHYEMAC#RXCLK	Input	125 MHz	25 MHz	2.5 MHz

Table 5-3: **Client Clock Frequency**

Data Rate	Direction	1000 Mb/s	100 Mb/s	10 Mb/s
EMAC#CLIENTRXCLIENTCLKOUT/ CLIENTEMAC#RXCLIENTCLKIN	Output/ Input	125 MHz	12.5 MHz	1.25 MHz
EMAC#CLIENTTXCLIENTCLKOUT/ CLIENTEMAC#TXCLIENTCLKIN	Output/ Input	125 MHz	12.5 MHz	1.25 MHz

Table 5-4: **MII/GMII/RGMII Clock Frequency**

Clock Signals	Direction	1000 Mb/s	100 Mb/s	10 Mb/s
EMAC#CLIENTTXGMIIMIICLKOUT/ CLIENTEMAC#TXGMIIMIICLKIN	Output/ Input	125 MHz	25 MHz	2.5 MHz

Table 5-5: **Host Interface and MDIO Clock Frequencies**

Clock Signals	Direction	Frequency
HOSTCLK	Input	Up to 125 MHz.
DCREMACCLK	Input	The same clock frequency as the PowerPC clock, CPMC405CLOCK. Must be phase-aligned with the PowerPC clock.
EMAC#PHYMCLKOUT	Output	Not to exceed 2.5 MHz for IEEE Std 802.3 compliance.

Transmit Clocking Scheme

Figure 5-1 shows the clocks used in the transmit module of the Ethernet MAC. In this figure, TX_CORE_CLK and TX_GMII_MII_CLK are internal clock signals.

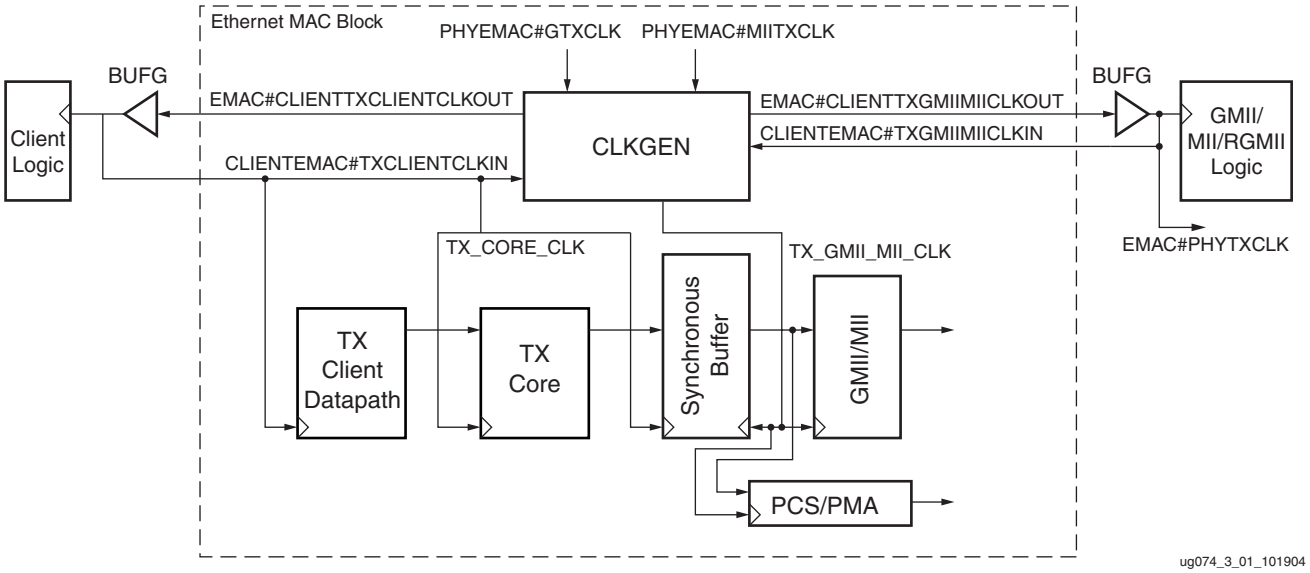


Figure 5-1: **Transmit Clock**

The clock generation module uses the PHYEMAC#GTXCLK (1 Gb/s) and PHYEMAC#MIITXCLK (10/100 Mb/s) inputs to generate EMAC#CLIENTTXCLIENTCLKOUT to run the circuitry in the FPGA fabric connecting to the client side. The CLIENTEMAC#TXCLIENTCLKIN signal runs the client logic and transmit engine inside the Ethernet MAC. This clock signal must be from the FPGA clock drivers (BUFG) of EMAC#CLIENTTXCLIENTCLKOUT.

On the physical interface side, when the Ethernet MAC is configured in GMII, MII, or RGMII mode, EMAC#CLIENTTXGMIIMIICLKOUT drives the clock in the FPGA fabric connecting to the GMII/MII/RGMII sublayer. EMAC#CLIENTTXGMIIMIICLKOUT is also fed back into CLIENTEMAC#TXGMIIMIICLKIN.

The CLIENTEMAC#TXGMIIMIICLKIN signal runs the MII/GMII/RGMII logic inside the Ethernet MAC. This clock signal must be from the FPGA clock drivers (BUFG) of EMAC#CLIENTTXGMIIMIICLKOUT. When Ethernet MAC is configured in SGMII or 1000BASE-X mode, TX_GMII_MII_CLK is driven by PHYEMAC#GTXCLK, and the CLIENTEMAC#TXGMIIMIICLKIN clock is not used.

When configured in MII mode, EMAC#CLIENTTXGMIIMIICLKOUT is derived from PHYEMAC#MIITXCLK. When configured in either GMII, RGMII, SGMII, or 1000BASE-X

PCS/PMA mode, EMAC#CLIENTTXGMIIMIICLKOUT is derived from the PHYEMAC#GTXCLK. See Chapter 4, "Physical Interface" for clock usage.

Receive Clocking Scheme

Figure 5-2 shows the clocks used in the receive module of the Ethernet MAC. In this figure, RX_CORE_CLK and RX_GMII_MII_CLK are internal clock signals.

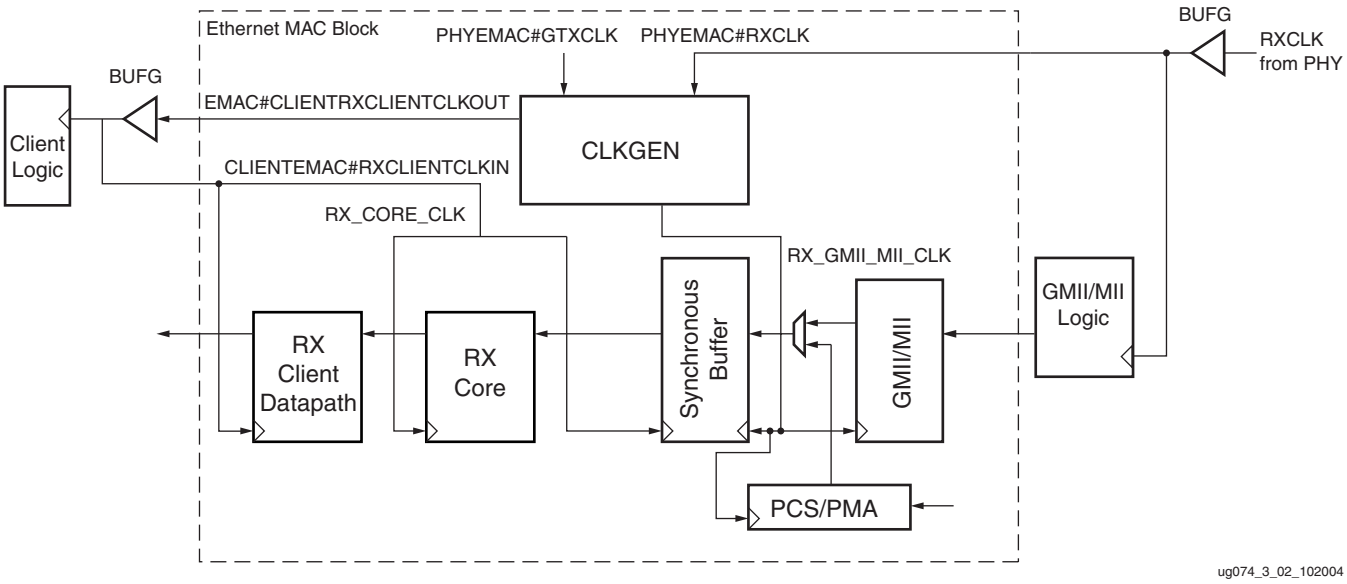


Figure 5-2: **Receive Clocks**

The clock generation module takes the PHYEMAC#RXCLK from the physical interface and generates the EMAC#CLIENTRXCLIENTCLKOUT to run the circuitry in the FPGA fabric connecting to the client side. CLIENTEMAC#RXCLIENTCLKIN runs the client logic and receive engine inside the Ethernet MAC. This clock signal must be from the FPGA clock drivers (BUFG) of EMAC#CLIENTRXCLIENTCLKOUT.

When configured in MII/GMII/RGMII mode, the internal RX_GMII_MII_CLK is derived from the PHYEMAC#RXCLK and used to run the MII/GMII/RGMII sublayer. When the Ethernet MAC is configured in either SGMII or 1000BASE-X PCS/PMA mode, the clock to run the PCS/PMA sublayer is generated from the PHYEMAC#GTXCLK. See Chapter 4, "Physical Interface" for clock usage.

Ethernet MAC Configuration

The Ethernet MAC can be configured using hardware or by accessing the registers through the host interface in software. The three methods for configuration are described in the following sections:

1. Tie-off pins in hardware (see "Tie-Off Pins" in Chapter 2)
2. Generic host bus using the host interface (see "Generic Host Bus" in Chapter 3)
3. DCR using the host interface (see "Using the DCR Bus as the Host Bus" in Chapter 3)

Table 5-6 shows the register addresses for each of the two Ethernet MACs.

Table 5-6: **Ethernet MAC Register Addresses**

	{HOSTEMAC1SEL, HOSTADDR[9:0]}	Register Description
EMAC0	0x200	Receiver Configuration (Word 0)
	0x240	Receiver Configuration (Word 1)
	0x280	Transmitter Configuration
	0x2C0	Flow Control Configuration
	0x300	Ethernet MAC Mode Configuration
	0x320	RGMII/SGMII Configuration
	0x340	Management Configuration
	0x380	Unicast Address (Word 0)
	0x384	Unicast Address (Word 1)
	0x388	Multicast Address Table Access (Word 0)
	0x38C	Multicast Address Table Access (Word 1)
	0x390	Address Filter Mode
EMAC1	0x600	Receiver Configuration (Word 0)
	0x640	Receiver Configuration (Word 1)
	0x680	Transmitter Configuration
	0x6C0	Flow Control Configuration
	0x700	Ethernet MAC Mode Configuration
	0x720	RGMII/SGMII Configuration
	0x740	Management Configuration
	0x780	Unicast Address (Word 0)
	0x784	Unicast Address (Word 1)
	0x788	Multicast Address Table Access (Word 0)
	0x78C	Multicast Address Table Access (Word 1)
	0x790	Address Filter Mode

Notes:

1. HOSTEMAC1SEL acts as address bit 10 to select between EMAC0 and EMAC1.

Auto-Negotiation Interrupt

Auto-Negotiation is controlled and monitored through the PCS Management Registers (see Table 4-10, "Control Register (Register 0)") and is consequently only available when the optional MDIO Management Interface is present.

Overview of Operation

1000BASE-X Standard

Figure 5-3 illustrates the operation of 1000BASE-X auto-negotiation.

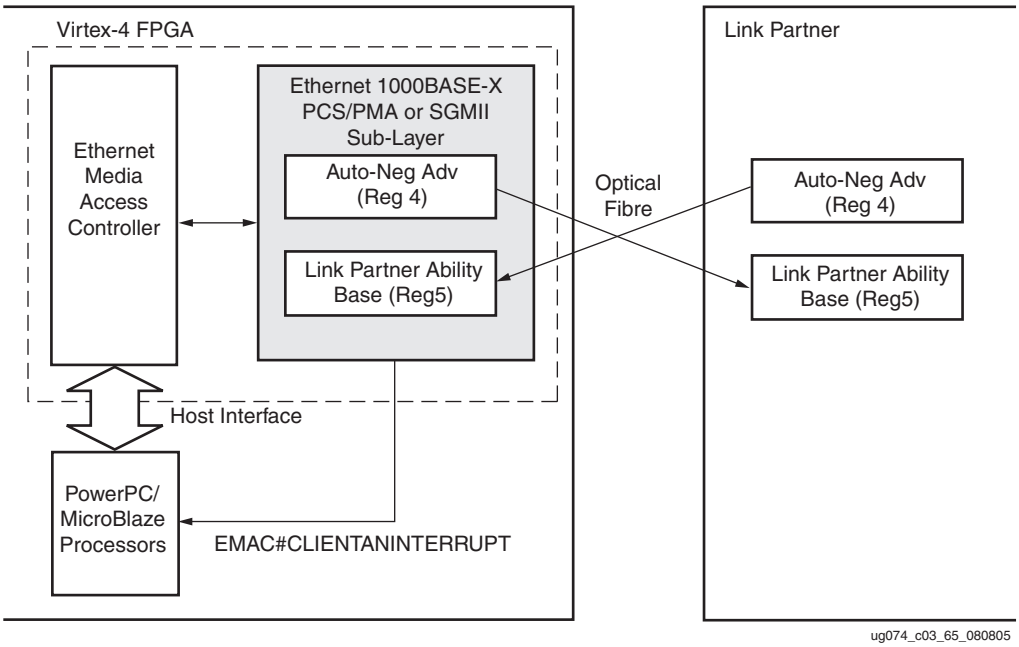


Figure 5-3: **1000BASE-X Auto-Negotiation Overview**

IEEE Std 802.3, Clause 37 describes the 1000BASE-X auto-negotiation function. This function allows a device to advertise the supported modes of operation to a device at the remote end of a link segment (the link partner), and to detect corresponding operational modes advertised by the Link Partner. The operation of the 1000BASE-X auto-negotiation is summarized:

1. To enable auto-negotiation, both auto-negotiation (see Table 4-10, "Control Register (Register 0)"), and MDIO (see Table 3-14, "Management Configuration Register") must be enabled. Then, auto-negotiation automatically starts:
   - After power-up, reset, or enabling of the MDIO interface
   - Upon loss of synchronization
   - Whenever the link partner initiates auto-negotiation
   - Whenever an Auto-Negotiation Restart is requested (see Table 4-10, "Control Register (Register 0)")
   - When the PCS/PMA Reset bit is set (see Table 4-10, "Control Register (Register 0)").

2. During auto-negotiation, the contents of Register 4 (see Table 4-13, "Auto-Negotiation Advertisement Register (Register 4)") are transferred to the link partner. This register can be written through the management interface, and enables software control of the system's advertised abilities. Information provided in this register includes:
   - Fault condition signalling
   - Duplex mode
   - Flow control capabilities for the attached Ethernet MAC.
3. At the same time, the advertised abilities of the Link Partner are transferred into Register 5 (see Table 4-14, "Auto-Negotiation Link Partner Ability Base Register (Register 5)"). This includes the same information as in Register 4.
4. Under normal conditions, this completes the auto-negotiation information exchange. It is now the responsibility of system management (for example, software running on an embedded PowerPC or MicroBlaze device) to complete the cycle. The results of the auto-negotiation should be read from Register 5, and other networking components, such as an attached Ethernet MAC, should be configured accordingly. There are two methods by which a host processor may learn of the competition of an auto-negotiation cycle:
   - By polling the auto-negotiation completion bit 1.5 in Register 1 (see Table 4-11, "Status Register (Register 1)").
   - By using the auto-negotiation interrupt port (see "Using the Auto-Negotiation Interrupt," page 153).

SGMII Standard

Figure 5-4 illustrates the operation of SGMII auto-negotiation.

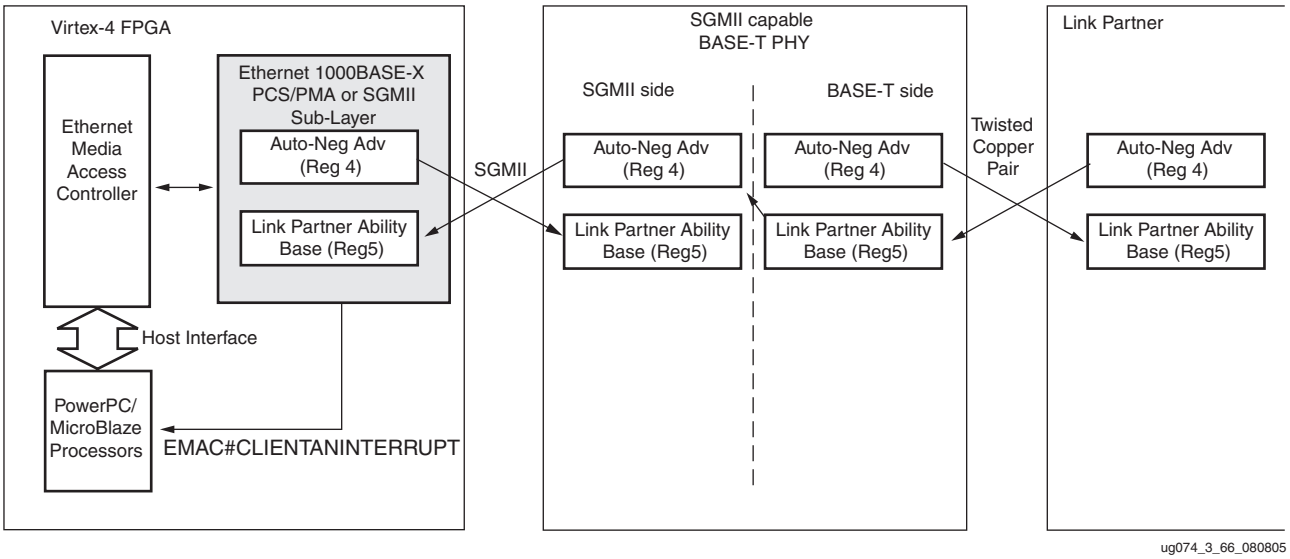


Figure 5-4: **SGMII Auto-Negotiation Overview**

The SGMII capable PHY has two distinctive sides to auto-negotiation:

- The PHY performs auto-negotiation with its link partner using the relevant auto-negotiation standard for the chosen medium (BASE-T auto-negotiation, illustrated in Figure 5-4, uses a twisted copper pair medium). This resolves the operational speed and duplex mode with the link partner.

- The PHY then passes the results of the auto-negotiation process with the link partner to the Ethernet 1000BASE-X PCS/PMA or SGMII (in SGMII mode), by leveraging the 1000BASE-X auto-negotiation specification shown in Figure 5-3. This transfers the results of the Link Partner auto-negotiation across the SGMII and this is the only auto-negotiation observed by the sub-layer.

The SGMII auto-negotiation function leverages the 1000BASE-X PCS/PMA auto-negotiation function with the exception of:

- The duration of the Link Timer of the SGMII auto-negotiation decreases from 10 ms to 1.6 ms making the entire auto-negotiation cycle faster (see "Auto-Negotiation Link Timer," page 153).
- The information exchanged now contains speed resolution in addition to duplex mode.

The results of SGMII auto-negotiation can be used as described in "1000BASE-X Auto-Negotiation Overview," page 151.

Auto-Negotiation Link Timer

The built-in auto-negotiation Link Timer has different durations for different standards.

1000BASE-X Standard

The 1000BASE-X standard Link Timer is defined as having a duration somewhere between 10.354 ms and 10.387 ms.

SGMII Standard

The SGMII standard Link Timer is defined as having a duration of 1.606 ms to 1.638 ms.

Using the Auto-Negotiation Interrupt

The auto-negotiation function has an EMAC#CLIENTANINTERRUPT port. This port is designed to be used with common microprocessor bus architectures (for example, the CoreConnect™ bus interfacing to a MicroBlaze design or the PPC405 processor implemented in the Virtex-4 device).

The operation of this port is enabled or disabled and cleared via Register 16 (see Table 4-19, "Vendor-Specific Register: Auto-Negotiation Interrupt Control Register (Register 16)").

- When disabled, this port is permanently driven Low.
- When enabled, this port is set to logic `1` following the completion of an auto-negotiation cycle. It remains high until cleared after a zero is written to bit 16.1 (Interrupt Status bit) of Register 16.

XILINX®

Chapter 6

Use Models

This chapter contains the following sections:

- "Simulation Models"
- "Pinout Guidelines"
- "Interfacing to the Processor DCR"
- "Interfacing to an FPGA Fabric-Based Statistics Block"

Simulation Models

SecureIP Model

SecureIP models are encrypted versions of the actual HDL code. These models allow the user to simulate the functionality of the design without the overhead of simulating RTL. A Verilog LRM-IEEE 1364-2005 encryption-compliant simulator is required to use SecureIP.

The SecureIP model of the Ethernet MAC is installed with Xilinx® tools and can be precompiled into UniSim and SimPrim libraries. These libraries are used for functional and timing simulations, respectively. VHDL and Verilog wrappers are generated by the CORE Generator™ tool in the ISE software, as well as the scripts to simulate the Secure IP model.

For further help using the Ethernet MAC SecureIP model, see the documentation supplied with ISE software, especially the Synthesis and Simulation Design Guide at:

http://www.xilinx.com/support/software_manuals.htm.

Model Considerations

The DCR bus, except for DCREMACENABLE, is internally connected to the PPC405 processor in the Virtex-4 FPGA processor block. However, for the SecureIP model of the Ethernet MAC, the user has to connect the DCR portion with the SecureIP model of the PowerPC 405 processor including DCREMACENABLE.

When simulating with the PCS/PMA layer (i.e., the Ethernet MAC is configured in either SGMII or 1000BASE-X PCS/PMA mode) and auto-negotiation is initialized to OFF (TIEEMAC#CONFIGVEC[77] = 0), EMAC#PHYSYNCACQSTATUS waits to be asserted High before the start of transmission. This allows the PCS layer to obtain synchronization.

Pinout Guidelines

Xilinx recommends the following guidelines to improve design timing using the Virtex-4 FPGA Embedded Tri-Mode Ethernet MAC:

- If available, use dedicated global clock pins for the Ethernet MAC input clocks.
- Use the column of IOBs located closest to the PowerPC processor and Ethernet MAC block.
- Use the MGTs located closest to the PowerPC processor and Ethernet MAC block.

Interfacing to the Processor DCR

As described in the Host Interface section, the host interface allows the user to:

- Access the Ethernet MAC configuration registers.
- Access the optional accumulated statistics IP registers if implemented in the fabric.
- Access the configuration and multicast address table registers for the Address Filter block.
- Access the MDIO registers of the PHY.
- Access the Virtex®-4 FX FPGA processor blocks. For additional information on these processor blocks, see UG018, *PowerPC405 Processor Block Reference Guide*.

The following subsections describe sample codes of various transactions for interfacing to the processor DCR.

Reading from the Ethernet MAC Configuration Register

1. Write to cntlReg register with the desired address of the Ethernet MAC configuration register.
2. Poll the RDYstatus register until the configuration Read-Ready bit is asserted.
3. Read from the dataRegLSW register to show the value of the Ethernet MAC configuration register.

Assuming the DCR base address is `0x0`, to read from the EMAC0 transmitter configuration register:

```
// EMAC Configuration Register 0x280 (EMAC0 Transmitter Configuration)
// Write the address of EMAC0 Transmitter Configuration register to the
// cntlReg register
mtdcr(0x0 + 14, 0x280);
// Poll the RDYstatus register
while ( !(mfdcr(0x0 + 15) & 0x00000020) );
// Read the dataRegLSW with the values returned from the EMAC0
// Transmitter Configuration register
mfdcr (0x0 + 13);
```

Writing to the Ethernet MAC Configuration Register

1. Write to dataRegLSW register with the desired value for the Ethernet MAC configuration register.
2. Write to cntlReg register with the desired address of the Ethernet MAC configuration register.
3. Poll the RDYstatus register until the configuration write ready bit is asserted.

Assuming the DCR base address is `0x0`, to write to the EMAC1 flow control register:

```
// EMAC Configuration Register 0x6C0 (EMAC1 Flow Control)
// Write to enable the flow control on both the transmit and receive
// side of EMAC1, set bits 29 and 30 to "1"
mtdcr(0x0 + 13, 0x60000000)
// Write the address of EMAC1 Flow Control register to the cntlReg
// register
mtdcr(0x0 + 14, 0x86C0);
// Poll the RDYstatus register
while ( !(mfdcr(0x0 + 15) & 0x00004000) );
```

Reading From the Statistics IP Register (When Implemented in the Fabric)

1. Write to cntlReg register with the desired address of the statistics IP register.
2. Poll the RDYstatus register until the statistics Read-Ready bit is asserted.
3. Read from the dataRegMSW and dataRegLSW registers to show the value from the statistics IP register.

Assuming the DCR base address is `0x0`, to read from EMAC0 statistics IP register `0x0`:

```
// Statistics IP Register 0x0 (Check how many frames were received OK)
// Write the address of the Statistics IP register to the cntlReg register
mtdcr(0x0 + 14, 0x0);

// Poll the RDYstatus register
while ( !(mfdcr(0x0 + 15) & 0x00000001) );

// Read the values returned of the Statistics IP Register 0x0 (64-bit
// value)from the dataRegMSW and dataRegLSW registers
stats_msw = mfdcr(0x0 + 12);
stats_lsw = mfdcr(0x0 + 13);
```

Reading from the Multicast Address Table Register of the Address Filter Block

The same methods used in reading and writing to the Ethernet MAC configuration registers through the DCR apply to the address filter configuration registers.

1. Write to dataRegLSW register the read mask bit to read the multicast address table register with the respective register being accessed (there are four multicast address table registers in the address filter block).
2. Write to cntlReg register with the address register of multicast address - `0x38C` for EMAC0 and `0x78C` for EMAC1. Set the Write enable bit to write to the multicast address (Word 1) (see Table 3-19, "Multicast Address Table Access (Word 1)").
3. Poll the RDYstatus register until the address filter read-ready bit is asserted.
4. Read from the dataRegMSW and dataRegLSW registers to show the address stored in the multicast address table register selected for reading.

Assuming the DCR base address is `0x0`, to read from the multicast address table register (2) of EMAC0:

```
// MULTI_ADDR Register 2 of AF Block
// Set the enable bit of the MULTI_ADDR RNW and MULTI_ADDR Register 2
mtdcr(0x0 + 13, 0x00820000);

// Write the address of EMAC0 Multicast Address register to the cntlReg
// register
mtdcr(0x0 + 14, 0x838C);

// Poll the RDYstatus register
while ( !(mfdcr(0x0 + 15) & 0x00000008) );

// Read the values returned of the Multicast Address Word0 and Word1
// registers (48-bit value)
// from the dataRegMSW and dataRegLSW registers
mult_addr_msw = mfdcr (0x0 + 12);
mult_addr_lsw = mfdcr (0x0 + 13);
```

Writing to the Multicast Address Table Register of the Address Filter Block

For writing to the desired multicast address table register of the AF block, two write operations must be performed.

1. Write to dataRegLSW register the multicast address[31:0] to be stored on the desired multicast address table register.
2. Write to cntlReg register with the address for multicast address word 0 - `0x388` for EMAC0 and `0x788` for EMAC1. Set the Write enable bit.
3. Poll the RDYstatus register until the address configuration write bit is asserted.
4. Write to dataRegLSW register the multicast address[47:32] with the write mask bit and the value of the multicast address table register to be accessed.
5. Write to cntlReg register with the address for multicast address word 1 - `0x38C` for EMAC0 and `0x78C` for EMAC1. Set the Write enable bit.
6. Poll the RDYstatus register until the address configuration write bit is asserted.

Assuming the DCR base address is `0x0`, to write the multicast address `0xFACEDEAFCAFE` to the multicast address table register `0x1` of EMAC1:

```
// Write the multicast address[31:0] to the dataRegLSW register
mtdcr(0x0 + 13, 0xDEAFCAFE);
// Write the address of EMAC1 Multicast Address Word 0 register to the
// cntlReg register
mtdcr(0x0 + 14, 0x8788);
// Poll the RDYstatus register
while ( !(mfdcr(0x0 + 15) & 0x00001000) );
// MULTI_ADDR Register 1 of AF Block
// Write the multicast address [47:32] with the MULTI_ADDR write mask
// bit to the dataRegLSW register
mtdcr(0x0 + 13, 0x0081FACE);
// Write the address of EMAC1 Multicast Address Word 1 register to the
// cntlReg register
mtdcr(0x0 + 14, 0x878C);
// Poll the RDYstatus register
while ( !(mfdcr(0x0 + 3) & 0x00001000) );
```

Reading the PHY Registers Using MDIO

1. Write to dataRegLSW register with the MDIO enable bit and the clock divider frequency for MDC.
2. Write to cntlReg register to write to the Ethernet MAC management configuration register.
3. Poll the RDYstatus register until the configuration Write-Ready bit is asserted.
4. Write to the dataRegLSW register with the PHY address and register to be accessed.
5. Write to the cntlReg register the decode address for a MDIO address output with the Read-Enable mask asserted.
6. Poll the RDYstatus register until the MDIO Read-Ready bit is asserted.
7. Read from the dataRegLSW, to shows the value of the PHY register being accessed.

Assume the DCR base address is `0x0` to read from the PHY address 1 and PHY register `0x0` of EMAC0. MDIO must be enabled by writing to the management configuration register with the clock divider for MDC. Assuming the host frequency is 50 MHz and the divider is `0xA`, results in an MDC frequency of 2.27 MHz.

```
// EMAC Management Register 0x340 (EMAC0 Management Configuration)
// Write the data to the EMAC0 Management Configuration register to
// enable MDIO with the clock divider 0xA
mtdcr(0x0 + 13, 0x0000004A);
// Write the address of EMAC0 Management Configuration register to the
// cntlReg register
mtdcr(0x0 + 14, 0x8340);
// Poll the RDYstatus register for writing completion
while ( !(mfdcr(0x0 + 15) & 0x00000040) );
// Write the PHY address and PHY register to be accessed to the
// dataRegLSW register
mtdcr(0x0 + 13, 0x00000020);
// Write the decode address for MDIO address output to the cntlReg
// register
mtdcr(0x0 + 14, 0x03B4);
// Poll the RDYstatus register
while ( !(mfdcr(0x0 + 15) & 0x00000002) );
// Read the dataRegLSW with the values returned from the
// PHY Register 0x0
mfdcr (0x0 + 13);
```

Writing to the PHY Registers Using MDIO

1. Write to the dataRegLSW register with the MDIO enable bit and the clock divider frequency for MDC.
2. Write to the cntlReg register to write to the Ethernet MAC management configuration register.
3. Poll the RDYstatus register until the configuration Write-Ready bit is asserted.
4. Write to the dataRegLSW register with the data to be written to the PHY register.
5. Write to the cntlReg register the decode address for MDIO write data.
6. Write to the dataRegLSW register with the PHY address and register to be accessed.
7. Write to the cntlReg register the decode address for MDIO address output with the Write-Enable mask asserted.
8. Poll the RDYstatus register until the MDIO write ready bit is asserted.

Assume the DCR base address is `0x0`, to write `0x1140` to PHY address 1 and PHY register `0x0` of EMAC1. The isolate bit of the PCS/PMA sublayer is not set, and auto-negotiation is enabled (see "1000BASE-X PCS/PMA" in Chapter 4 for more information). MDIO must be enabled by writing to the management configuration register with the clock divider for MDC. Assuming the host frequency is 50 MHz and the divider is `0xA`, the result is an MDC frequency of 2.27 MHz.

```
// EMAC Management Register 0x740 (EMAC1 Management Configuration)
// Write the data to the EMAC0 Management Configuration register to
// enable MDIO with the clock divider 0xA
mtdcr(0x0 + 13, 0x0000004A);
// Write the address of EMAC1 Management Configuration register to the
// cntlReg register
mtdcr(0x0 + 14, 0x8740);
// Poll the RDYstatus register for writing completion
while ( !(mfdcr(0x0 + 15) & 0x00000400) );
// Write the data to the PHY 0x0 register
mtdcr(0x0 + 13, 0x00001140);
// Write the decode address for MDIO Write Data to the cntlReg register
mtdcr(0x0 + 14, 0x87B0);
```

```
// Write the PHY address and PHY register to be accessed to the
// dataRegLSW register
mtdcr(0x0 + 13, 0x00000020);
// Write the decode address for MDIO address output to the cntlReg
// register
mtdcr(0x0 + 14, 0x87B4);
// Poll the RDYstatus register
while ( !(mfdcr(0x0 + 15) & 0x00000400) );
```

Interfacing to an FPGA Fabric-Based Statistics Block

When the Ethernet MAC Is Implemented with the Host Bus

When the Ethernet MAC is used with the host bus, interfacing to a fabric-based statistics block is straight forward. Statistics information is passed from the Ethernet MAC via the statistics vectors EMAC#CLIENTTXSTATS and EMAC#CLIENTRXSTATS. The statistics values can then be read via a host interface, shared between the statistics counters and the Ethernet MAC block.

To share the host bus without contention, statistics counters need to use a different address space than the Ethernet MAC configuration registers. Conflicts with MDIO register access are avoided by only accessing statistics counters when the signal HOSTMIIMSEL is at logic `0`. Implementation of the addressing scheme shown in Table 6-1 ensures that the host bus can be shared without contention. This scheme provides space to address 512 statistics counters per Ethernet MAC, using addresses `0x000` to `0x1FF`.

Table 6-1: **Addressing Scheme**

Transaction	Host_miim_sel	Host_addr[9]
Configuration	0	1
MIIM Access	1	X
Statistics Access	0	0

Figure 6-1 shows how to integrate the Ethernet MAC with the LogiCORE Ethernet Statistics block, where the Ethernet statistics counters are accessed via the host bus. The LogiCORE Ethernet Statistics block is used with the addressing scheme shown in Table 6-1. DS323, *LogiCORE Ethernet Statistics Data Sheet*, provides a a full description of the Ethernet Statistics LogiCORE block. Figure 6-1 illustrates how to connect Ethernet Statistics blocks to both Ethernet MACs within the Ethernet MAC block. If statistics are required for only one Ethernet MAC, then the multiplexing between the statistics cores is simply replaced with a straight-through connection.

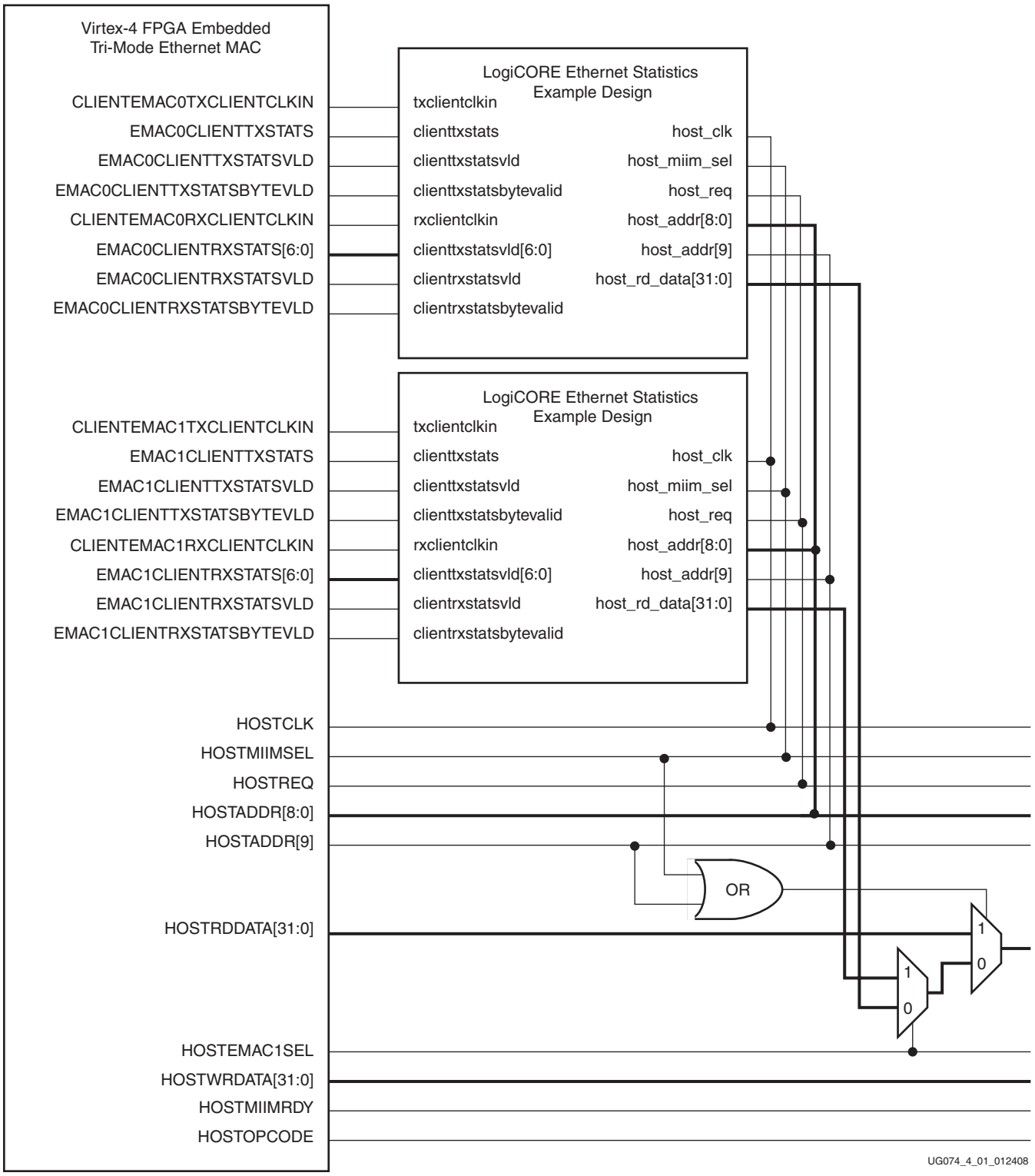


Figure 6-1: **Host Bus to Ethernet Statistics Connection**

When the Ethernet MAC Is Implemented with the DCR Bus

When the Ethernet MAC is implemented with the DCR bus connected directly to the PPC405, the host bus signals of the Ethernet MAC can access the statistics counters.

The host bus I/O signals of the Ethernet MAC are enabled for statistics counter access when a DCR read operation is made for address codes `0x000` to `0x02F` and `0x040` to `0x04F` inclusive (Table 3-30, page 91 describes the DCR address code space). This use of the host bus I/O signals provides a means of accessing the FPGA fabric from the processor with space for 64 addresses.

When the DCR bus is instructed to access registers in this address code region, the DCR bridge translates the DCR commands into generic host read signals on the host bus I/O signals. The DCR transaction is encoded on the host bus signals HOSTRDDATA[31:0] and HOSTMIIMRDY as described in Table 6-2 and Figure 6-2. These signals can access statistics counters in the same way as if a standalone host bus is used. The statistics values read from statistics counters are captured from the host bus signals HOSTWRDATA[31:0] as shown in Figure 6-2. The data read from the host bus can then be accessed by reading the DCR data registers.

Table 6-2: **DCR to Host Mapping**

EMAC Host Bus Port		Port Direction on EMAC	DCR Register	Equivalent Standalone Host Bus Signal
HOSTRDDATA	[15]	Out		HOSTREQ
	[14:13]	Out		HOSTOPCODE[1:0]
	[10]	Out	cntlReg[21], EMAC1Sel	HOSTEMAC1SEL
	[9:0]	Out	cntlReg[22:31], Address Code	HOSTADDR[9:0]
HOSTMIIMRDY		Out		HOSTMIIMSEL
HOSTWRDATA[31:0]		In	dataRegLSW / dataRegMSW	HOSRTRDDATA[31:0]
HOSTMIIMSEL		In		HOSTMIIMRDY

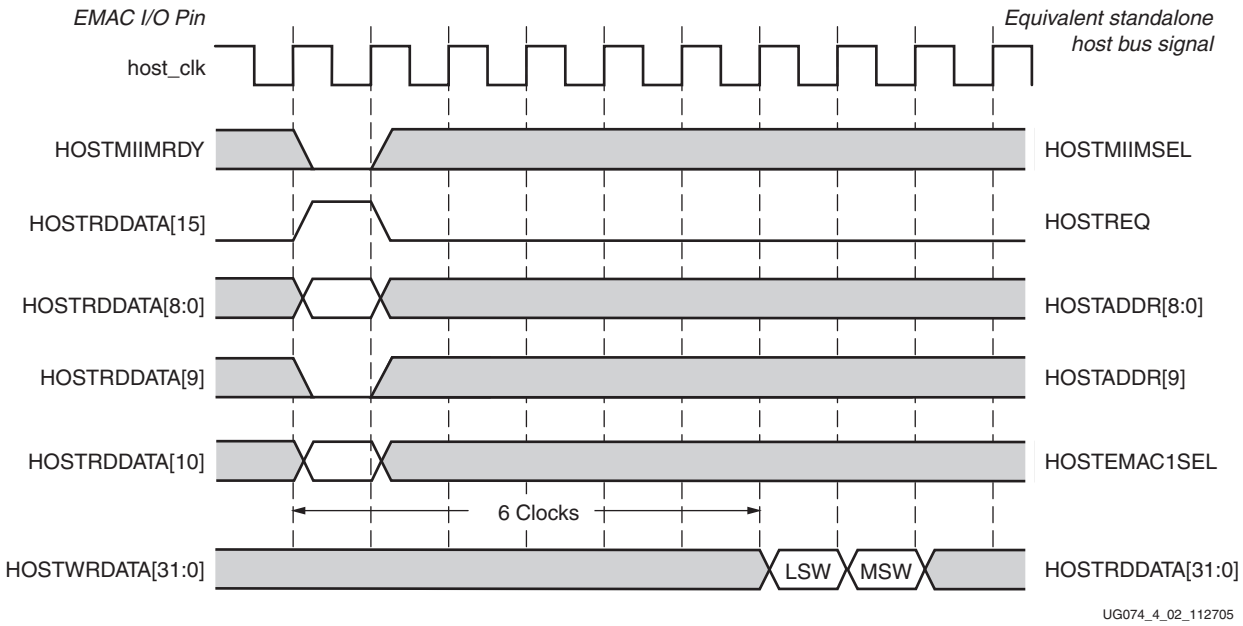


Figure 6-2: **Statistics Register Read Timing**

Figure 6-3 shows how to integrate the Ethernet MAC with the LogiCORE Ethernet Statistics block, where the LogiCORE Ethernet statistics counters are accessed via the DCR bus. DS323, *LogiCORE Ethernet Statistics Data Sheet*, provides a a full description of the Ethernet Statistics LogiCORE block. Figure 6-3 illustrates how to connect LogiCORE Ethernet Statistics blocks to both Ethernet MACs within the Ethernet MAC block. If statistics are required for only one Ethernet MAC, then the multiplexing between the statistics cores is simply replaced with a straight-through connection.

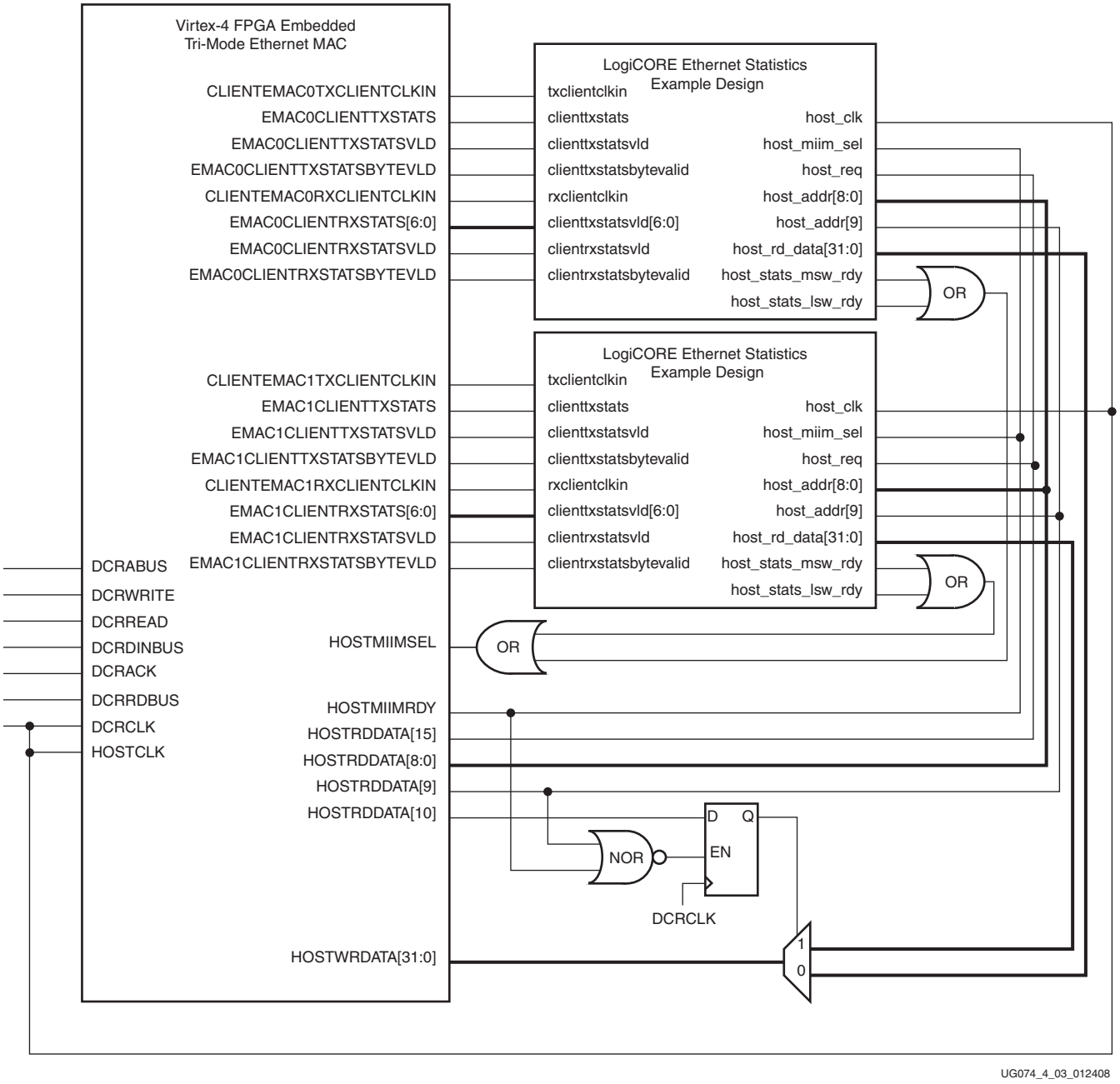


Figure 6-3: **DCR Bus to Ethernet Statistics Connection**

XILINX®

Chapter 7

Using the Embedded Ethernet MAC

This chapter describes how the embedded Ethernet MAC can be incorporated into a design using the CORE Generator™ tool.

Accessing the Ethernet MAC from the CORE Generator tool

Generating the Virtex®-4 FPGA embedded Ethernet MAC wrapper files using the CORE Generator tool greatly simplifies the use of the Virtex-4 FPGA Ethernet MAC. The Ethernet MAC is highly configurable, and not all pins/interfaces are required for every configuration. The CORE Generator tool allows the configuration of the Ethernet MAC to be selected using a GUI and generates HDL wrapper files for the configuration. These wrapper files hide much of the complexity of the Ethernet MAC by bringing out only the interface signals for the selected configuration. Accessing the Ethernet MAC from the CORE Generator tool provides the following features:

- Allows selection of one or both of the two Ethernet MACs (EMAC0 and EMAC1) from the embedded Ethernet MAC primitive
- Sets the values of the EMAC0 and EMAC1 attributes based on user options
- Provides user-configurable Ethernet MAC physical interfaces
  - Supports MII, GMII, RGMII v1.3, RGMII v2.0, SGMII, and 1000BASE-X PCS/PMA interfaces
  - Provides off-chip connections for physical interfaces by instantiating RocketIO™ transceivers, and logic as required, for the selected physical interfaces
- Provides an optimized clocking scheme for the selected physical interface and instantiates the required clock buffers, DCMs, and other required components.
- Provides a simple FIFO-loopback example design, which is connected to the MAC client interfaces
- Provides a simple demonstration test bench based on the selected configuration
- Generates VHDL or Verilog wrapper files

Simulating the Ethernet MAC using the Ethernet MAC wrappers

The Ethernet MAC wrappers generated by the CORE Generator tool also provide a loopback design of the embedded Ethernet MAC, a demo test bench to exercise the wrappers and the example design, and scripts for simulation and implementation.

The example design and the demo testbench are set up to provide fast simulation cycles (for configurations using DCMs). To speed up simulations that require DCMs, the DCM reset signals by default are not connected to the 200 ms reset pulse that is required by Virtex-4 devices (see UG070, *Virtex-4 FPGA User Guide*). To use the design in hardware, a

DCM reset module is provided to generate the required reset pulse and users need to connect the output of this module to the DCM reset pins. This can be achieved by connecting the reset_200ms_# signal to the reset_200ms_in_# signal at any level of example design HDL hierarchy. See the block level wrapper file for more information.

For further details on the Ethernet MAC wrappers, refer to DS307, *Virtex-4 Embedded Tri-Mode Ethernet MAC Wrapper Data Shee*t and GSG240, *Virtex-4 Embedded Tri-Mode Ethernet MAC Wrapper Getting Started Guide*.

XILINX®

Appendix A

Ethernet MAC Timing Model

This appendix explains the timing parameters associated with the Ethernet MAC block. It is intended to be used in conjunction with the Timing Analyzer (TRCE) report from Xilinx® software.

Many signals enter and exit the Ethernet MAC block (as shown in Figure 2-3, page 21). The model presented in this appendix treats the Ethernet MAC block as a "black box." Propagation delays internal to the Ethernet MAC block logic are ignored. Signals are characterized with setup and hold times for inputs, and with clock to valid output times for outputs.

There are seven clocks associated with the Ethernet MAC block. Table 2-4, page 25 briefly describes the clock signals necessary to drive the Virtex®-4 FPGA Embedded Tri-Mode Ethernet MAC.

Timing Parameters

Parameter designations are constructed to reflect the functions they perform as well as the I/O signals to which they are synchronous. The following subsections explain the meaning of each of the basic timing parameter designations used in Table A-1 through Table A-6.

Input Setup/Hold Times Relative to Clock

Basic Format:

ParameterName_SIGNAL

where

ParameterName = T with subscript string defining the timing relationship

SIGNAL = name of Ethernet MAC signal synchronous to the clock

ParameterName Format:

$T_{MAC\mathit{x}CK}$ = Setup time before clock edge

$T_{MACCK\mathit{x}}$ = Hold time after clock edge

where:

x = {C (Control inputs)} {D (Data inputs)}

Setup/Hold Time (Examples):

T_{MACDCK_TXD}/T_{MACCKD_TXD} setup/hold times of TX data input relative to the rising edge of CLIENTEMAC#RXCLIENTCLKIN

Clock to Output Delays

Basic Format:

ParameterName_SIGNAL

where

ParameterName = T with subscript string defining the timing relationship

SIGNAL = name of Ethernet MAC signal synchronous to the clock

ParameterName Format:

T_{MACCKo} = Delay time from clock edge to output

Output Delay Time (Examples):

T_{MACCKO_VALID} rising edge of CLIENTEMAC#RXCLIENTCLKIN to RX data valid signals

T_{MACCKO_RXD} rising edge of CLIENTEMAC#RXCLIENTCLKIN to RX data signals

Core Latency

The latency values given in the following subsections can vary by three clock ticks in either direction, due to the crossing of clock domains within the core. All clock cycles refer to cycles of the appropriate (TX or RX) client interface clock.

Transmit Path Latency

The transmit path latency is measured by counting the number of clock cycles between a data byte being placed on the client interface and its appearance at the PHY interface of the Ethernet MAC. For RGMII/GMII/MII at all speeds, the latency is 13 clock cycles. SGMII has a latency of 13 clock cycles for 1 Gb/s and 11 clock cycles for 10/100 Mb/s. 1000BASE-X has a latency of 14 clock cycles.

Receive Path Latency

The receive path latency is measured as the number of clock cycles between a byte being driven onto the PHY receive interface of the EMAC and its appearance at the client. For GMII/MII, the latency is 17 clock cycles at all speeds. RGMII has a latency of 20 clock cycles. SGMII has a latency of 20 clock cycles for 1 Gb/s and 15 clock cycles for 10/100 Mb/s. 1000BASE-X has a latency of 22 clock cycles.

Timing Diagram and Timing Parameter Tables

The timing relationships described in this section are shown in Figure A-1.

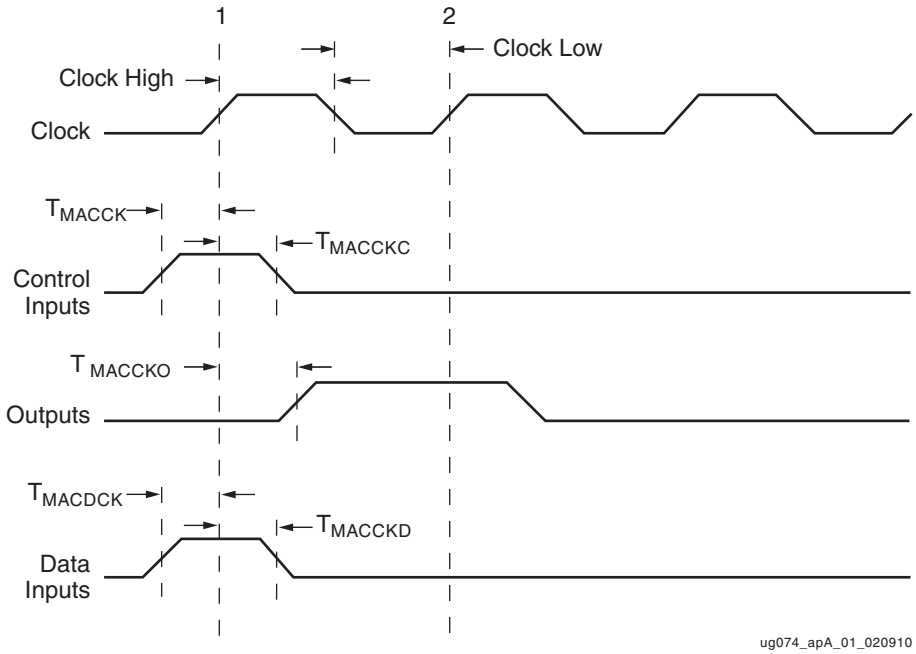


Figure A-1: **Ethernet MAC Timing Relative to Clock Edge**

Table A-1 through Table A-6 list the timing parameters as reported by the implementation tools relative to the clocks given in Table A-1, page 149, along with the Ethernet MAC signals that are synchronous to each clock.

- Table A-1, "CLIENTEMAC#RXCLIENTCLKIN Switching Characteristics," on page 171
- Table A-2, "CLIENTEMAC#TXCLIENTCLKIN Switching Characteristics," on page 172
- Table A-3, "HOSTCLK Switching Characteristics," on page 172
- Table A-4, "PHYEMAC#GTXCLK Switching Characteristics," on page 173
- Table A-5, "PHYEMAC#MIITXCLK Switching Characteristics," on page 174
- Table A-6, "PHYEMAC#RXCLK Switching Characteristics," on page 174

Table A-1: **CLIENTEMAC#RXCLIENTCLKIN Switching Characteristics**

Parameter	Function	Signal
Clock To Out:		
Tmaccko_ERROR	Data Output	EMAC#CLIENTRXBADFRAME
Tmaccko_CLKOUT	Data Output	EMAC#CLIENTRXCLIENTCLKOUT
Tmaccko_RXD	Data Output	EMAC#CLIENTRXD
Tmaccko_VALID	Data Output	EMAC#CLIENTRXDVLD
Tmaccko_VALID	Data Output	EMAC#CLIENTRXDVLDMSW
Tmaccko_FRAME	Data Output	EMAC#CLIENTRXFRAMEDROP
Tmaccko_FRAME	Data Output	EMAC#CLIENTRXGOODFRAME

Table A-2: **CLIENTEMAC#TXCLIENTCLKIN Switching Characteristics**

Parameter	Function	Signal
Setup and Hold Relative to Clock:		
Tmacckd_DCMLOCK	Data Hold	CLIENTEMAC#DCMLOCKED
Tmacdck_DCMLOCK	Data Setup	CLIENTEMAC#DCMLOCKED
Tmacckd_PAUSE	Data Hold	CLIENTEMAC#PAUSEREQ
Tmacdck_PAUSE	Data Setup	CLIENTEMAC#PAUSEREQ
Tmacckd_PAUSE	Data Hold	CLIENTEMAC#PAUSEVAL
Tmacdck_PAUSE	Data Setup	CLIENTEMAC#PAUSEVAL
Tmacckd_TXD	Data Hold	CLIENTEMAC#TXD
Tmacdck_TXD	Data Setup	CLIENTEMAC#TXD
Tmacckd_VALID	Data Hold	CLIENTEMAC#TXDVLD
Tmacdck_VALID	Data Setup	CLIENTEMAC#TXDVLD
Tmacckd_VALID	Data Hold	CLIENTEMAC#TXDVLDMSW
Tmacdck_VALID	Data Setup	CLIENTEMAC#TXDVLDMSW
Tmacckd_TXD	Data Hold	CLIENTEMAC#TXFIRSTBYTE
Tmacdck_TXD	Data Setup	CLIENTEMAC#TXFIRSTBYTE
Tmacckd_DELAY	Data Hold	CLIENTEMAC#TXIFGDELAY
Tmacdck_DELAY	Data Setup	CLIENTEMAC#TXIFGDELAY
Tmacckd_ERROR	Data Hold	CLIENTEMAC#TXUNDERRUN
Tmacdck_ERROR	Data Setup	CLIENTEMAC#TXUNDERRUN
Clock To Out:		
Tmaccko_IRQ	Data Output	EMAC#CLIENTANINTERRUPT
Tmaccko_ACK	Data Output	EMAC#CLIENTTXACK
Tmaccko_CLKOUT	Data Output	EMAC#CLIENTTXCLIENTCLKOUT
Tmaccko_ERROR	Data Output	EMAC#CLIENTTXCOLLISION
Tmaccko_CLKOUT	Data Output	EMAC#CLIENTTXGMIIMIICLKOUT
Tmaccko_RETRANS	Data Output	EMAC#CLIENTTXRETRANSMIT
Tmaccko_STATS	Data Output	EMAC#CLIENTTXSTATS
Tmaccko_VALID	Data Output	EMAC#CLIENTTXSTATSBYTEVLD
Tmaccko_VALID	Data Output	EMAC#CLIENTTXSTATSVLD

Table A-3: **HOSTCLK Switching Characteristics**

Parameter	Function	Signal
Setup and Hold Relative to Clock:		
Tmacckd_ENABLE	Data Hold	DCREMACENABLE
Tmacdck_ENABLE	Data Setup	DCREMACENABLE
Tmacckd_HOST	Data Hold	HOSTADDR
Tmacdck_HOST	Data Setup	HOSTADDR
Tmacckd_HOST	Data Hold	HOSTEMAC1SEL
Tmacdck_HOST	Data Setup	HOSTEMAC1SEL
Tmacckd_HOST	Data Hold	HOSTMIIMSEL

Table A-3: **HOSTCLK Switching Characteristics**

Parameter	Function	Signal
Tmacdck_HOST	Data Setup	HOSTMIIMSEL
Tmacckd_HOST	Data Hold	HOSTOPCODE
Tmacdck_HOST	Data Setup	HOSTOPCODE
Tmacckd_HOST	Data Hold	HOSTREQ
Tmacdck_HOST	Data Setup	HOSTREQ
Tmacckd_HOST	Data Hold	HOSTWRDATA
Tmacdck_HOST	Data Setup	HOSTWRDATA
Tmacckd_PHYAD	Data Hold	PHYEMAC#PHYAD
Tmacdck_PHYAD	Data Setup	PHYEMAC#PHYAD
Tmacckd_RESET	Data Hold	RESET
Tmacdck_RESET	Data Setup	RESET
Clock To Out:		
Tmaccko_DCRDONE	Data Output	DCRHOSTDONEIR
Tmaccko_CLKOUT	Data Output	EMAC#PHYMCLKOUT
Tmaccko_DOUT	Data Output	EMAC#PHYMDOUT
Tmaccko_MDTRI	Data Output	EMAC#PHYMDTRI
Tmaccko_HOST	Data Output	HOSTMIIMRDY
Tmaccko_HOST	Data Output	HOSTRDDATA

Table A-4: **PHYEMAC#GTXCLK Switching Characteristics**

Parameter	Function	Signal
Setup and Hold Relative to Clock:		
Tmacckd_MDIN	Data Hold	PHYEMAC#MDIN
Tmacdck_MDIN	Data Setup	PHYEMAC#MDIN
Tmacckd_ERROR	Data Hold	PHYEMAC#RXBUFERR
Tmacdck_ERROR	Data Setup	PHYEMAC#RXBUFERR
Tmacckd_STATUS	Data Hold	PHYEMAC#RXBUFSTATUS
Tmacdck_STATUS	Data Setup	PHYEMAC#RXBUFSTATUS
Tmacckd_COMMA	Data Hold	PHYEMAC#RXCHARISCOMMA
Tmacdck_COMMA	Data Setup	PHYEMAC#RXCHARISCOMMA
Tmacckd_CHAR	Data Hold	PHYEMAC#RXCHARISK
Tmacdck_CHAR	Data Setup	PHYEMAC#RXCHARISK
Tmacckd_CRC	Data Hold	PHYEMAC#RXCHECKINGCRC
Tmacdck_CRC	Data Setup	PHYEMAC#RXCHECKINGCRC
Tmacckd_CORCNT	Data Hold	PHYEMAC#RXCLKCORCNT
Tmacdck_CORCNT	Data Setup	PHYEMAC#RXCLKCORCNT
Tmacckd_COMMA	Data Hold	PHYEMAC#RXCOMMADET
Tmacdck_COMMA	Data Setup	PHYEMAC#RXCOMMADET
Tmacckd_ERROR	Data Hold	PHYEMAC#RXDISPERR
Tmacdck_ERROR	Data Setup	PHYEMAC#RXDISPERR

Table A-4: **PHYEMAC#GTXCLK Switching Characteristics *(Cont'd)***

Parameter	Function	Signal
Tmacckd_SYNC	Data Hold	PHYEMAC#RXLOSSOFSYNC
Tmacdck_SYNC	Data Setup	PHYEMAC#RXLOSSOFSYNC
Tmacckd_NOTINT	Data Hold	PHYEMAC#RXNOTINTABLE
Tmacdck_NOTINT	Data Setup	PHYEMAC#RXNOTINTABLE
Tmacckd_DISP	Data Hold	PHYEMAC#RXRUNDISP
Tmacdck_DISP	Data Setup	PHYEMAC#RXRUNDISP
Tmacckd_DETECT	Data Hold	PHYEMAC#SIGNALDET
Tmacdck_DETECT	Data Setup	PHYEMAC#SIGNALDET
Tmacckd_ERROR	Data Hold	PHYEMAC#TXBUFERR
Tmacdck_ERROR	Data Setup	PHYEMAC#TXBUFERR
Clock To Out:		
Tmaccko_COMMA	Data Output	EMAC#PHYENCOMMAALIGN
Tmaccko_LOOPBACK	Data Output	EMAC#PHYLOOPBACKMSB
Tmaccko_RESET	Data Output	EMAC#PHYMGTTXRESET
Tmaccko_POWER	Data Output	EMAC#PHYPOWERDOWN
Tmaccko_SYNC	Data Output	EMAC#PHYSYNCACQSTATUS
Tmaccko_DISP	Data Output	EMAC#PHYTXCHARDISPMODE
Tmaccko_DISP	Data Output	EMAC#PHYTXCHARDISPVAL
Tmaccko_CHAR	Data Output	EMAC#PHYTXCHARISK

Table A-5: **PHYEMAC#MIITXCLK Switching Characteristics**

Parameter	Function	Signal
Clock To Out:		
Tmaccko_TXCLK	Data Output	EMAC#PHYTXCLK
Tmaccko_TXD	Data Output	EMAC#PHYTXD
Tmaccko_EN	Data Output	EMAC#PHYTXEN
Tmaccko_ERROR	Data Output	EMAC#PHYTXER

Table A-6: **PHYEMAC#RXCLK Switching Characteristics**

Parameter	Function	Signal
Setup and Hold Relative to Clock:		
Tmacckd_COL	Data Hold	PHYEMAC#COL
Tmacdck_COL	Data Setup	PHYEMAC#COL
Tmacckd_CRS	Data Hold	PHYEMAC#CRS
Tmacdck_CRS	Data Setup	PHYEMAC#CRS
Tmacckd_RXD	Data Hold	PHYEMAC#RXD
Tmacdck_RXD	Data Setup	PHYEMAC#RXD
Tmacckd_VALID	Data Hold	PHYEMAC#RXDV
Tmacdck_VALID	Data Setup	PHYEMAC#RXDV
Tmacckd_ERROR	Data Hold	PHYEMAC#RXER

Table A-6: **PHYEMAC#RXCLK Switching Characteristics**

Parameter	Function	Signal
Tmacdck_ERROR	Data Setup	PHYEMAC#RXER
Tmacckd_CONFIG	Data Hold	TIEEMAC#CONFIGVEC
Tmacdck_CONFIG	Data Setup	TIEEMAC#CONFIGVEC
Clock To Out:		
Tmaccko_RXD	Data Output	EMAC#CLIENTRXDVREG
Tmaccko_STATS	Data Output	EMAC#CLIENTRXSTATS
Tmaccko_VALID	Data Output	EMAC#CLIENTRXSTATSBYTEVLD
Tmaccko_VALID	Data Output	EMAC#CLIENTRXSTATSVLD
Tmaccko_RESET	Data Output	EMAC#PHYMGTRXRESET
Tmaccko_VALID	Data Output	EMAC1CLIENTRXDVLD
Tmaccko_VALID	Data Output	EMAC1CLIENTRXDVLDMSW
Tmaccko_VALID	Data Output	EMAC1CLIENTRXDVREG