Virtex-5 FPGA Embedded Tri-Mode Ethernet MAC User Guide

UG194 (v1.10) February 14, 2011

XILINX®

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
09/06/06	1.0	Initial Xilinx release on CD.
10/13/06	1.1	Added further descriptions and figures to "SGMII RX Elastic Buffer" in Chapter 6.
03/06/07	1.2	Modified RGMII IDELAY placement and inputs in Figure 6-3, Figure 6-4, Figure 6-6 through Figure 6-10, and Figure 6-12 through Figure 6-17.
08/08/07	1.3	Chapter 1: Minor grammatical edits in "Preamble" and "Model Considerations." Chapter 2: Modified description of EMAC#_LTCHECK_DISABLE in Table 2-17. Chapter 3: • Added "Standard Conditions" and "1000BASE-X/SGMII Specific Conditions" sections. • Added new paragraph to "Enabled." • Modified paragraph in "Receiving a PAUSE Control Frame." Chapter 4: • Modified Figure 4-1.

Date	Version	Revision
08/08/07 (*cont'd*)	1.3 (*cont'd*)	• Edited text in "Clocking Requirements." • Minor grammatical edits in paragraph above Table 4-16, and "Ethernet MAC Configuration and Address Filter Access." Chapter 6: • Added IDELAY elements to GMII_RX_CLK and GMII_RXD[7:0] lines in Figure 6-6 to Figure 6-10. • Added description of fixed-mode IDELAYs to "Gigabit Media Independent Interface (GMII)." • Added IDELAY elements to RGMII_RXC lines between IBUFG and BUFG in Figure 6-12 to Figure 6-17. • Modified description of IDELAY in "Reduced Gigabit Media Independent Interface (RGMII)." • Modified speed data in Table 6-2. • Modified Ethernet MAC connections in Figure 6-36 and Figure 6-37.
03/31/08	1.4	• Updated the following: Table 3-4, Figure 6-3, Figure 6-4, Figure 6-6, Figure 6-7, Figure 6-10, Figure 6-12, Figure 6-13, Figure 6-15, Figure 6-17, Figure 6-20, Figure 6-21, Figure 6-22 (and notes), Figure 6-34, Figure 6-35, Figure 6-36, Figure 6-39, and Figure 6-40. • Updated the following sections: "Ethernet Communications Port for an Embedded Processor," "RocketIO Serial Transceiver Signals," "Statistics Registers/Counters," "GMII Clock Management for 1 Gb/s Only," "GMII Clock Management for Tri-Speed Operation Using Clock Enables," "16-Bit Data Client," "RocketIO Serial Transceiver Logic Using the RX Elastic Buffer in FPGA Logic," "SGMII Clock Management (LXT and SXT Devices)," "1000BASE-X PCS/PMA (16-Bit Data Client) Mode," and "Host Clock." • Added LXT device information. • Added Figure 6-23, Figure 6-24, Figure 6-36, and Figure 6-37.
06/06/08	1.5	Chapter 2: Updated Table 2-16. Chapter 3: • Updated "Client-Supplied FCS Passing," page 53, "Frame Collisions - Half-Duplex 10/100 Mb/s Operation Only," page 56, "VLAN Tagged Frames," page 66, and the enabled discussion in "Length/Type Field Error Checks." Chapter 4: • Updated Table 4-7, Table 4-8, Figure 4-1, Figure 4-4, and Figure 4-5. • Updated "Introduction to the Ethernet MAC Host Interface." Chapter 6: Updated "Overview of Operation." Appendix C: Updated "DCR Bus Modifications." Added Appendix D, "Differences between Soft IP Cores and the Tri-Mode Ethernet MAC."
07/24/08	1.6	Updated "Transmitting a PAUSE Control Frame," Table 5-6, and Table 5-7.
10/17/08	1.7	• Added HOSTEMAC1SEL signal to Figure 4-6 and Figure 4-7. • Added host_stats_lsw_rdy and host_stats_msw_rdy signals to Figure 7-1.

Date	Version	Revision
04/28/09	1.8	All chapters: • Changed references to FXT devices to include both TXT and FXT devices. • Replaced the term platform with the term device. • Replaced the term SmartModel with the term SecureIP model. Chapter 2: • Changed references to GTP transceiver to GTP/GTX transceiver in Figure 2-1, page 27, Figure 2-2, page 29, and Figure 2-3, page 32. Chapter 3: • Revised the text of "Client-Supplied FCS Passing" on page 53. Chapter 5: • Changed references to GTP transceiver to GTP/GTX transceiver in Figure 5-4, page 119, Figure 5-6, page 121, Figure 5-7, page 122, and Figure 5-8, page 130. Chapter 6: • Revised Note 1 to indicate the clock input of IFD can be driven by a BUFIO in Figure 6-3, page 141, Figure 6-8, page 148, Figure 6-14, page 157, and Figure 6-15, page 159. • Added Note 1 indicating a BUFG buffer can be replaced by a BUFR and the clock input of IFD can be driven by a BUFIO in Figure 6-4, page 142, Figure 6-6, page 145, Figure 6-7, page 147, and Figure 6-10, page 152. • Added Note 1 indicating a BUFG buffer can be replaced by a BUFR and the clock input of IDDR can be driven by a BUFIO in Figure 6-12, page 154, Figure 6-13, page 156, Figure 6-16, page 160, and Figure 6-17, page 162. • Where a RocketIO® transceiver is shown, added the term RocketIO in Figure 6-21, page 169, Figure 6-22, page 170, Figure 6-36, page 191, and Figure 6-37, page 192. • Revised schematic by adding a BUFGMUX to the PHYEMAC#MIITXCLK input in Figure 6-8, page 148. • Revised schematic by changing the I0 input source for the BUFGMUX on the PHYEMAC#TXGMIIMIICLKIN input in Figure 6-8, page 148. • Revised schematic by adding a BUFG buffer between the transceiver REFCLKOUT output and the DCM CLKIN input in Figure 6-22, page 170, Figure 6-23, page 172, Figure 6-24, page 173, Figure 6-38, page 194, and Figure 6-39, page 195. • Changed references to GTP transceiver to GTP/GTX transceiver in Figure 6-28, page 179, Figure 6-29, page 182, Figure 6-33, page 186, and Figure 6-41, page 198. • Added link to UG198 in "RX Elastic Buffer Implementations" on page 181 and "The FPGA RX Elastic Buffer Requirement" on page 181.

Date	Version	Revision
10/01/09	1.9	Chapter 1: • Revised first paragraph in "Frame Transmission and Interframe Gap," page 23. Chapter 2: • Revised EMAC#_RXFLOWCTRL_ENABLE and EMAC#_TXFLOWCTRL_ENABLE descriptions in Table 2-17, page 45. Chapter 3: • Revised first paragraph in "Flow Control Block," page 73. Chapter 5: • Revised Link Status description in Table 5-4, page 124.
02/14/11	1.10	Updated description of EMAC#_PAUSEADDR[47:0] in Table 2-17. Added description of standard and alternative clock management to "Transmitter Statistics Vector" and "Receiver Statistics Vector." Updated Table 4-14. Added "Use of Clock Correction Sequences."

Table of Contents

Chapter 4: Host/DCR Bus Interfaces

Chapter 5: MDIO Interface

Chapter 6: Physical Interface

Chapter 7: Interfacing to a Statistics Block

Appendix A: Pinout Guidelines

Appendix B: Ethernet MAC Clocks

Appendix C: Virtex-4 to Virtex-5 FPGA Enhancements

Appendix D: Differences between Soft IP Cores and the Tri-Mode Ethernet MAC

About This Guide

This user guide is a description of the Virtex®-5 FPGA Embedded Tri-Mode Ethernet MAC. Complete and up-to-date documentation of the Virtex-5 family of FPGAs is available on the Xilinx® website at http://www.xilinx.com/virtex5.

Guide Contents

This user guide contains the following chapters:

- Chapter 1, "Introduction," introduces the Virtex-5 FPGA Embedded Tri-Mode Ethernet MAC and describes how to get started using the Ethernet MAC.
- Chapter 2, "Ethernet MAC Overview," describes the architecture of the Ethernet MAC, defines its signal interface, and gives an overview of its possible configurations.
- Chapter 3, "Client Interface," provides details of the client interface protocol and the Ethernet functionality associated with the client interface.
- Chapter 4, "Host/DCR Bus Interfaces," describes how to access registers in the Ethernet MAC using either the generic host bus or the DCR bus. The Ethernet MAC register descriptions are also found in this chapter.
- Chapter 5, "MDIO Interface," describes the MDIO interface protocol and the MDIO implementation in the Ethernet MAC. The SGMII and 1000BASE-X PCS/PMA management register descriptions are also found in this chapter.
- Chapter 6, "Physical Interface," describes all of the possible configurations for the physical interface and includes a description of optimized clocking schemes that can be used with the Ethernet MAC.
- Chapter 7, "Interfacing to a Statistics Block," provides details on how to use the Ethernet MAC with an FPGA logic-based statistics block.
- Appendix A, "Pinout Guidelines," lists recommendations to improve design timing when using the Virtex-5 FPGA Tri-Mode Ethernet MAC.
- Appendix B, "Ethernet MAC Clocks," gives an overview of the internal clock logic of the Ethernet MAC and summarizes the main clocking schemes. Definitions of each clock and its frequency, in all possible Ethernet MAC configurations, are also provided.
- Appendix C, "Virtex-4 to Virtex-5 FPGA Enhancements," documents some key enhancements and a few minor modifications.
- Appendix D, "Differences between Soft IP Cores and the Tri-Mode Ethernet MAC," explains the differences between soft IP cores and the Tri-Mode Ethernet MAC by listing the features exclusive to both.

Additional Documentation

The following documents are also available for download at http://www.xilinx.com/virtex5.

- Virtex-5 Family Overview

  The features and product selection of the Virtex-5 family are outlined in this overview.

- Virtex-5 FPGA Data Sheet: DC and Switching Characteristics

  This data sheet contains the DC and Switching Characteristic specifications for the Virtex-5 family.

- Virtex-5 FPGA User Guide

  This user guide includes chapters on:

  - Clocking Resources
  - Clock Management Technology (CMT)
  - Phase-Locked Loops (PLLs)
  - Block RAM
  - Configurable Logic Blocks (CLBs)
  - SelectIO™ Resources
  - SelectIO Logic Resources
  - Advanced SelectIO Logic Resources

- Virtex-5 FPGA RocketIO GTP Transceiver User Guide

  This guide describes the RocketIO™ GTP transceivers available in the Virtex-5 LXT and SXT devices.

- Virtex-5 FPGA RocketIO GTX Transceiver User Guide

  This guide describes the RocketIO GTX transceivers available in the Virtex-5 TXT and FXT devices.

- Virtex-5 FPGA Embedded Processor Block for PowerPC® 440 Designs

  This reference guide is a description of the embedded processor block available in the Virtex-5 TXT and FXT devices.

- Virtex-5 FPGA Integrated Endpoint Block User Guide for PCI Express Designs

  This guide describes the integrated Endpoint blocks in the Virtex-5 LXT, SXT, TXT, and FXT devices used for PCI Express® designs.

- XtremeDSP Design Considerations

  This guide describes the XtremeDSP™ slice and includes reference designs for using the DSP48E slice.

- Virtex-5 FPGA Configuration Guide

  This all-encompassing configuration guide includes chapters on configuration interfaces (serial and SelectMAP), bitstream encryption, Boundary-Scan and JTAG configuration, reconfiguration techniques, and readback through the SelectMAP and JTAG interfaces.

- Virtex-5 FPGA System Monitor User Guide

  The System Monitor functionality available in all the Virtex-5 devices is outlined in this guide.

- Virtex-5 FPGA Packaging and Pinout Specifications

  This specification includes the tables for device/package combinations and maximum I/Os, pin definitions, pinout tables, pinout diagrams, mechanical drawings, and thermal specifications.

- Virtex-5 FPGA PCB Designer's Guide

  This guide provides information on PCB design for Virtex-5 devices, with a focus on strategies for making design decisions at the PCB and interface level.

Additional Support Resources

To search the database of silicon and software questions and answers, or to create a technical support case in WebCase, see the Xilinx website at: http://www.xilinx.com/support.

User Guide Conventions

This document uses the following conventions.

Acronyms

Table 1-1 lists the acronyms and abbreviations that are used throughout this User Guide.

Table 1-1: **Acronyms and Abbreviations in this User Guide**

Acronym	Definition
Byte PHY	Name given to a particular Ethernet MAC clocking scheme described in this User Guide
CRC	Cyclic Redundancy Check
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
DA	Destination Address (MAC frame)
DCR	Device Control Register
DMA	Direct Memory Access
EDK	Embedded Development Kit (Xilinx software)
FCS	Frame Check Sequence (MAC frame)
FIFO	First In, First Out memory
FPGA	Field Programmable Gate Array
GBIC	Gigabit Interface Converter (optical transceiver)
Gb/s	Gigabits per second
GMII	Gigabit Media Independent Interface
GTP	High-speed serial transceivers offered in the Virtex-5 LXT and SXT platforms
GTX	High-speed serial transceivers offered in the Virtex-5 FXT and TXT platforms
IFG	Interframe Gap (MAC frame)

Table 1-1: **Acronyms and Abbreviations in this User Guide**

Acronym	Definition
IOB	Input/Output Block
ISE	Xilinx Integrated Software Environment
L/T	Length/Type field (MAC frame)
LAN	Local Area Network
MAC	Media Access Controller
Mb/s	Megabits per second
MDIO	Management Data Input/Output
MII	Media Independent Interface
OP	Operation code (read or write) for MDIO frame (sometimes called OPCODE in text)
OSI	Open Systems Interconnection
PCB	Printed Circuit Board
PCS	Physical Coding Sublayer
PHY	Physical Layer The term refers to all physical sublayers (PCS, PMA, PMD). Often applied to a device, e.g., BASE-T PHY: an external chip which can connect to the Ethernet MAC to perform this physical standard
PHYAD	Physical Address (MDIO Frame)
PLB	Processor Local Bus
PMA	Physical Medium Attachment
PMD	Physical Medium Dependent
PRE	Preamble (MDIO Frame)
REGAD	Register Address (MDIO Frame)
RGMII	Reduced Gigabit Media Independent Interface
RX	Receiver
SA	Source Address (MAC frame)
SFD	Start of Frame Delimiter (MAC frame)
SFP	Small Form-factor Pluggable (optical transceiver)
SGMII	Serial Gigabit Media Independent Interface
ST	Start Code (MDIO Frame) or Start of Frame
Stats	Abbreviation of Statistics
TA	Turnaround (MDIO Frame)
TX	Transmitter
VLAN	Virtual Local Area Network

Typographical

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

Convention	Meaning or Use	Example
Italic font	References to other documents	See the *Virtex-5 User Guide* for more information.
	Emphasis in text	The address (F) is asserted *after* clock event 2.
Underlined Text	Indicates a link to a web page.	http://www.xilinx.com/virtex5

Online Document

The following conventions 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 Documentation" for details. Refer to "Address Filtering" in Chapter 3 for details.
Blue, underlined text	Hyperlink to a website (URL)	Go to http://www.xilinx.com for the latest documentation.

XILINX®

Chapter 1

Introduction

This chapter introduces the Virtex®-5 FPGA Embedded Tri-Mode Ethernet MAC and provides an overview of typical Ethernet applications and the Ethernet protocol. It also provides guidance on how to successfully incorporate the Virtex-5 FPGA Ethernet Media Access Controller (MAC) into a larger design using two alternative Xilinx® tools.

The Virtex-5 device family contains paired embedded Ethernet MACs that are independently configurable to meet all common Ethernet system connectivity needs. Table 1 in the *Virtex-5 Family Overview* lists the device type versus the number of available Ethernet MACs.

This chapter contains the following sections:

Key Features

The key features of the Virtex-5 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 interframe 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 transmit and stripping on receive paths
- 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, general, and broadcast addresses
- Media Independent Interface (MII), Gigabit Media Independent Interface (GMII), and Reduced Gigabit Media Independent Interface (RGMII)
- 1000BASE-X Physical Coding Sublayer (PCS) and a Physical Medium Attachment (PMA) sublayer included for use with the Virtex-5 RocketIO™ serial transceivers to provide a complete on-chip 1000BASE-X implementation
- Serial Gigabit Media Independent Interface (SGMII) supported through the RocketIO serial transceivers' interfaces to external copper PHY layer for full-duplex operation

Typical Ethernet Application Overview

Typical applications for the Ethernet MAC include:

- "Ethernet Switch or Router"
- "Ethernet Communications Port for an Embedded Processor"

Ethernet Switch or Router

Figure 1-1 illustrates a typical application for a single Ethernet MAC. The PHY side of the core is connected to an off-the-shelf Ethernet PHY device, which performs the BASE-T standard at 1 Gb/s, 100 Mb/s, and 10 Mb/s speeds. The PHY device can be connected using any of the following supported interfaces: GMII/MII, RGMII, or SGMII.

The client side of the Embedded Ethernet MAC is connected to a FIFO to complete a single Ethernet port. This port is connected to a Switch or Routing matrix, which can contain several ports.

For this application, the recommended place to start is in "Accessing the Ethernet MAC from the CORE Generator Tool," page 25. The CORE Generator™ tool provides an example design for the Virtex-5 FPGA Embedded Tri-Mode Ethernet MAC for any of the supported physical interfaces. A FIFO example is also generated, which can be used as the FIFO in the illustration, for a typical application.

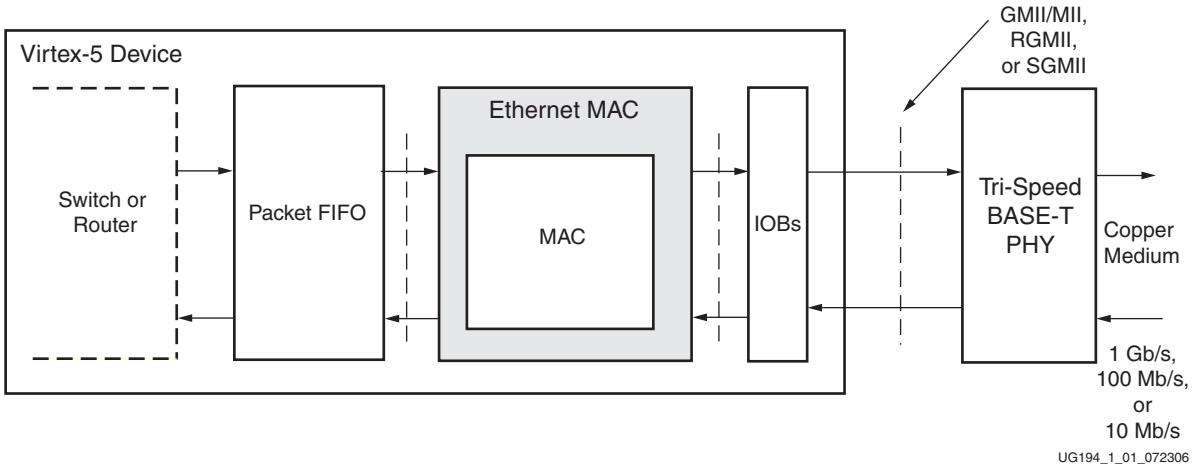


Figure 1-1: **Typical Application: Ethernet Switch or Router**

Ethernet Communications Port for an Embedded Processor

Figure 1-2 illustrates a typical application for a single Ethernet MAC. The PHY side of the core is connected to an off-the-shelf Ethernet PHY device, which performs the BASE-T standard at 1 Gb/s, 100 Mb/s, and 10 Mb/s speeds. The PHY device can be connected using any of the following supported interfaces: GMII/MII, RGMII, or SGMII.

A soft core is provided as part of the Xilinx Platform Studio (XPS), Embedded Development Kit (EDK) IP portfolio to connect the client interface of the Embedded Ethernet MAC to the DMA port of a processor. DS537, *XPSLL TEMAC Data Sheet*, describes the XPS_LL_TEMAC, which can be instantiated for an intended processor application.

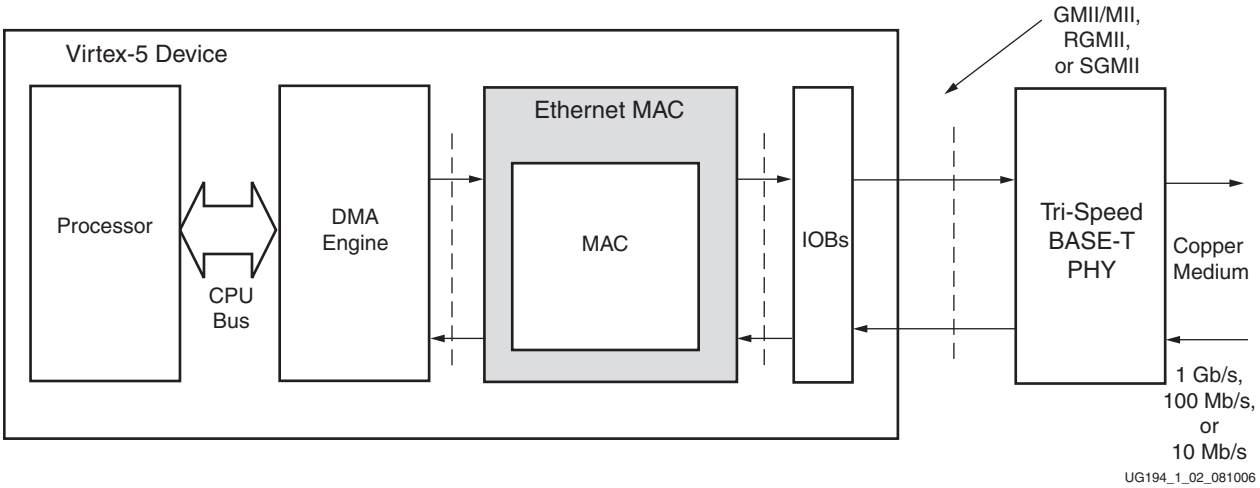


Figure 1-2: **Typical Application: Ethernet Communications Port for Embedded Processor**

Ethernet Protocol Overview

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

Ethernet Sublayer Architecture

Figure 1-3 illustrates the relationship between the OSI reference model and the Ethernet MAC, as defined in the IEEE 802.3 specification. The grayed-in layers show the functionality that the Ethernet MAC handles. Figure 1-3 also shows where the supported physical interfaces fit into the architecture.

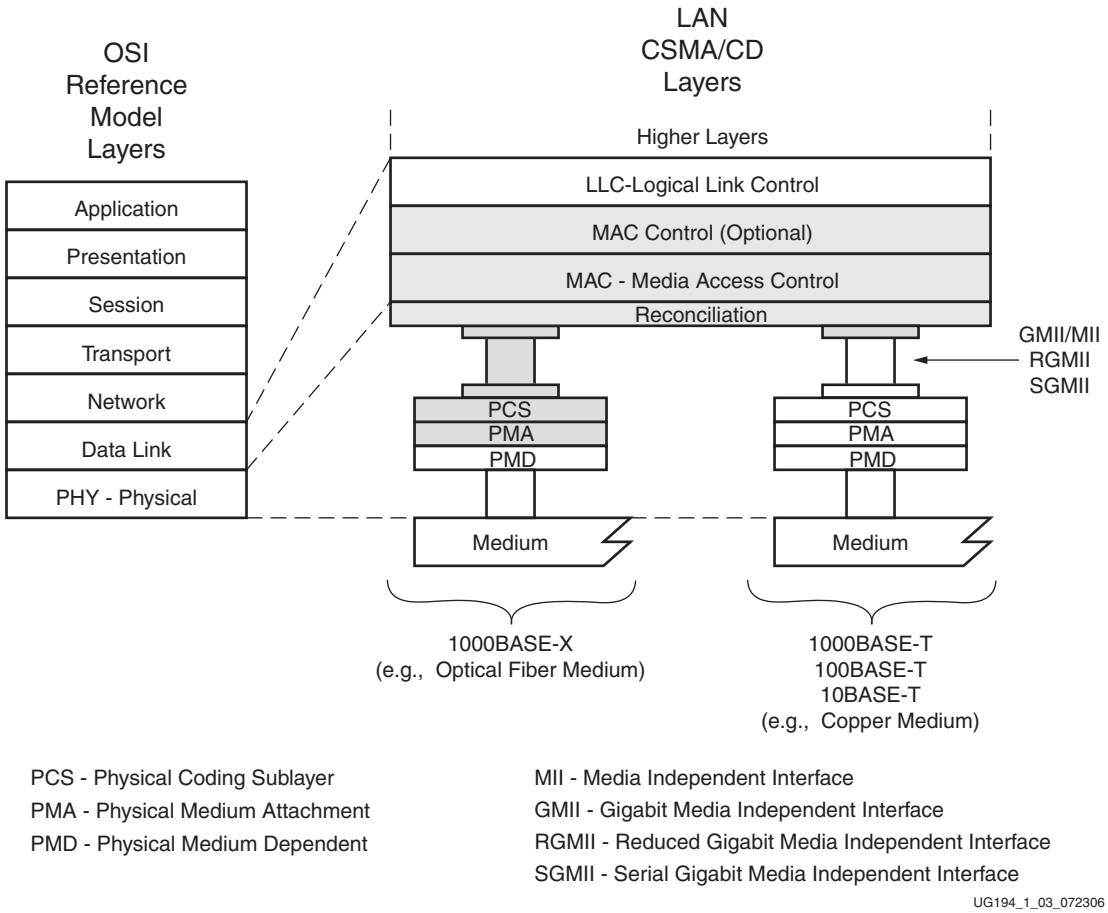


Figure 1-3: **IEEE 802_3.2002 Ethernet Model**

MAC and MAC CONTROL Sublayer

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

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

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

Physical Sublayers PCS, PMA, and PMD

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

- **BASE-T** PHYs provide a link between the MAC and copper mediums. This functionality is not offered within the Virtex-5 FPGA Embedded Tri-Mode Ethernet MAC. However, external BASE-T PHY devices are readily available on the market. These can connect to the Ethernet MAC, using GMII/MII, RGMII, or SGMII interfaces.
- **BASE-X** PHYs provide a link between the MAC and (usually) fibre optic mediums. The Virtex-5 FPGA Embedded Tri-Mode Ethernet MAC is capable of supporting the 1 Gb/s BASE-X standard; 1000BASE-X PCS and PMA sublayers can be offered by connecting the Ethernet MAC to a RocketIO serial transceiver. An optical transceiver can be directly connected to the RocketIO serial transceiver to complete the PMD sublayer functionality.

Ethernet Data Format

Ethernet data is encapsulated in frames, as shown in Figure 1-4, for standard Ethernet frames. The fields in the frame are transmitted from left to right. The bytes within the frame are transmitted from left to right (from least significant bit to most significant bit unless specified otherwise). The Ethernet MAC can handle jumbo Ethernet frames where the data field can be much larger than 1500 bytes.

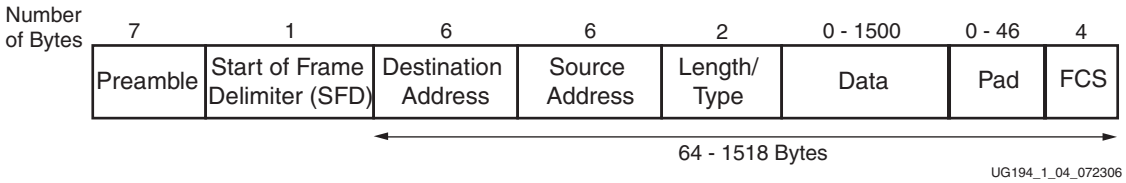


Figure 1-4: **Standard Ethernet Frame Format**

The Ethernet MAC can also accept VLAN frames. The VLAN frame format is shown in Figure 1-5. If the frame is a VLAN type frame, the Ethernet MAC accepts four additional bytes.

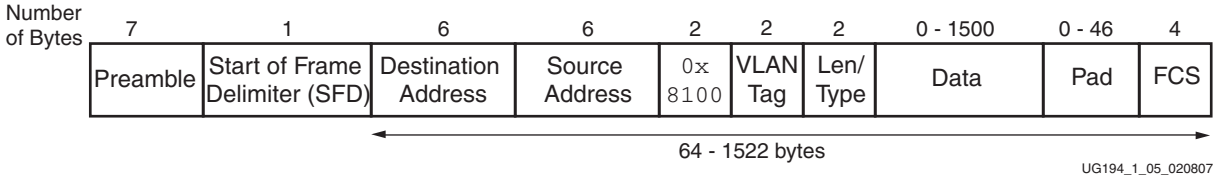


Figure 1-5: **Ethernet VLAN Frame Format**

Ethernet pause/flow control frames can be transmitted and received by the Ethernet MAC. Figure 3-29, page 75 shows how a pause/flow control frame differs from the standard Ethernet frame format.

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

Preamble

For transmission, this field is automatically inserted by the Ethernet MAC. The preamble field was historically used for synchronization and contains seven bytes with the pattern `0x55`, transmitted from left to right. For reception, this field is always stripped from the incoming frame, before the data is passed to the client. The Ethernet MAC can receive Ethernet frames, even if the preamble does not exist, as long as a valid start of frame is available.

Start of Frame Delimiter

The start of frame delimiter field marks the start of the frame and must contain the pattern `0xD5`.

For transmission on the physical interface, this field is automatically inserted by the Ethernet MAC. For reception, this field is always stripped from the incoming frame before the data is passed to the client.

Destination Address

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

This field is the first field of the Ethernet frame that is always provided in the packet data for transmissions and is always retained in the receive packet data.

Source Address

For transmission, the source address of the Ethernet frame should be provided by the client because it is unmodified by the Ethernet MAC. The unicast address for the Ethernet MAC is used as the source address when the Ethernet MAC creates pause control frames. The source address field is always retained in the receive packet data.

Length/Type

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

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

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

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

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

Data

The data field can vary from 0 to 1500 bytes in length for a normal frame. The Ethernet MAC can handle jumbo frames of any length.

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

Pad

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

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

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

FCS

The value of the FCS field is calculated over the destination address, source address, length/type, data, and pad fields using a 32-bit Cyclic Redundancy Check (CRC), as defined in IEEE Std 802.3-2002 para. 3.2.8:

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

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

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

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

Frame Transmission and Interframe Gap

Frames are transmitted over the Ethernet medium with an interframe gap, as specified by IEEE Std 802.3. For full-duplex systems, the minimum IFG is 96 bit times (9.6 µs for 10 Mb/s, 0.96 µs for 100 Mb/s, and 96 ns for 1 Gb/s). For half-duplex systems, the minimum IFG is 208 bit times and 144 bit times for RGMII and MII respectively. The defined IFG is a minimum and can be increased with a resulting decrease in throughput.

The process for frame transmission is different for half-duplex and full-duplex systems.

Half-Duplex Frame Transmission

In a half-duplex system, the CSMA/CD media access method defines how two or more stations share a common medium.

1. Even when it has nothing to transmit, the Ethernet MAC monitors the Ethernet medium for traffic by watching the carrier sense signal (CRS) from the external PHY. Whenever the medium is busy (CRS = 1), the Ethernet MAC defers to the passing frame by delaying any pending transmission of its own.
2. After the last bit of the passing frame (when the carrier sense signal changes from TRUE to FALSE), the Ethernet MAC starts the timing of the interframe gap.
3. The Ethernet MAC resets the interframe gap timer if the carrier sense becomes TRUE during the period defined by "interframe gap part 1 (IFG1)." The IEEE Std 802.3 states that this should be the first 2/3 of the interframe gap timing interval (64 bit times) but can be shorter and as small as zero. The purpose of this option is to support a possible brief failure of the carrier sense signal during a collision condition and is described in paragraph 4.2.3.2.1 of the IEEE standard.
4. The Ethernet MAC does not reset the interframe gap timer if carrier sense becomes TRUE during the period defined by "interframe gap part 2 (IFG2)" to ensure fair access to the bus. The IEEE Std 802.3 states that this should be the last 1/3 of the interframe gap timing interval.

If, after initiating a transmission, the message collides with the message of another station (COL = 1), then each transmitting station intentionally continues to transmit (jam) for an additional predefined period (32 bit times for 10/100 Mb/s) to ensure propagation of the collision throughout the system. The station remains silent for a random amount of time (backoff) before attempting to transmit again.

A station can experience a collision during the beginning of its transmission (the collision window) before its transmission has had time to propagate to all stations on the bus. After the collision window has passed, a transmitting station has acquired the bus. Subsequent collisions (late collisions) are avoided because all other (properly functioning) stations are assumed to have detected the transmission and are deferring to it.

Full-Duplex Frame Transmission

In a full-duplex system, there is a point-to-point dedicated connection between two Ethernet devices, capable of simultaneous transmit and receive with no possibility of collisions. The Ethernet MAC does not use the carrier sense signal from the external PHY because the medium is not shared, and the Ethernet MAC only needs to monitor its own transmissions. After the last bit of an Ethernet MAC frame transmission, the Ethernet MAC starts the interframe gap timer and defers transmissions until it reaches 96 bit times the value represented by CLIENTEMAC#TXIFGDELAY.

Using the Embedded Ethernet MAC

This section describes how the TEMAC User Guide can be easily incorporated into customer designs using Xilinx software:

- "Accessing the Ethernet MAC from the CORE Generator Tool": The CORE Generator tool provides wrapper files to help instantiate and configure the Ethernet MAC.
- "Simulating the Ethernet MAC using SecureIP Models": SecureIP models are provided with the ISE® software to allow the Ethernet MAC to be simulated both functionally and with timing information.

This section provides more information on how to use these Xilinx software tools to access the Ethernet MAC.

Accessing the Ethernet MAC from the CORE Generator Tool

Generating the Virtex-5 FPGA Embedded Tri-Mode Ethernet MAC wrapper files greatly simplifies the use of the Virtex-5 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 only bringing out 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/EMAC1) from the Embedded Ethernet MAC primitive
- Sets the values of the EMAC0/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 serial 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, etc.
- 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

For further details, refer to DS550, *Virtex-5 Embedded Tri-Mode Ethernet MAC Wrapper Data Sheet*.

Simulating the Ethernet MAC using SecureIP Models

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

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

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

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

Model Considerations

If the Ethernet MAC is configured in either SGMII or 1000BASE-X PCS/PMA mode, simulation times are much longer than when using other physical interfaces due to the auto-negotiation sequence. It is recommended that auto negotiation is disabled (by setting the EMAC#_PHYINITAUTONEG_ENABLE attribute to FALSE or by disabling auto-negotiation by writing through the host interface) to allow frame transmission to occur within a reasonable period of simulation time. See "Ethernet MAC Attributes," page 42. If simulating auto-negotiation, then simulation time can be reduced by setting the programmable link timer to a small value. This is achieved by modifying the EMAC#_LINKTIMERVAL[8:0] attribute.

Chapter 2

Ethernet MAC Overview

This chapter describes the architecture of the Virtex®-5 Tri-Mode Embedded Ethernet Media Access Controller (MAC) block. It contains the following sections:

- "Architecture Description"
- "Ethernet MAC Configuration Options"
- "Ethernet MAC Primitive"
- "Ethernet MAC Signal Descriptions"
- "Ethernet MAC Attributes"

Architecture Description

Figure 2-1 shows a block diagram of the Ethernet MAC block. The block contains two Ethernet MACs sharing a single host interface.

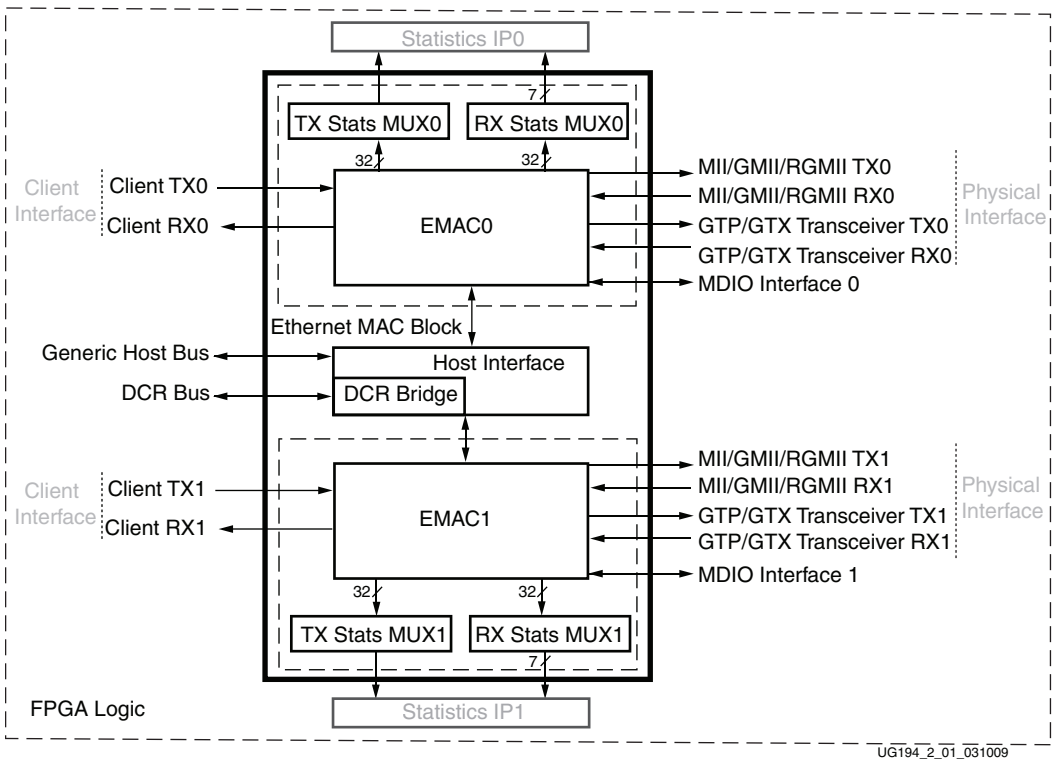


Figure 2-1: **Embedded Ethernet MAC Block**

The Ethernet MACs within the block are the same. When describing the functionality of the Ethernet MACs in this document, EMAC# is used when the description is for EMAC0 or EMAC1.

Each Ethernet MAC has an independent transmit (TX) and receive (RX) datapath, providing full-duplex capability.

The host interface of the Ethernet MAC provides access to the configuration registers of both Ethernet MACs, illustrated in Figure 2-1. The host interface can be accessed using either the generic host bus or the DCR bus through the DCR bridge.

The physical interface of each Ethernet MAC can be configured as either MII, GMII, RGMII, SGMII, or 1000BASE-X. Figure 2-1 shows all possible data ports for the physical interface; however, only one set of these TX and RX interfaces is active per Ethernet MAC, based on the selected physical interface configuration. The individual Ethernet MACs can be configured to have different physical interfaces.

Each Ethernet MAC has an optional Management Data I/O interface (MDIO), allowing access to the management registers of an external PHY or access to the physical interface management registers within the Ethernet MAC (used only when it is configured in either 1000BASE-X or SGMII modes).

Each Ethernet MAC outputs statistics vectors containing information about the Ethernet frames seen on its transmit and receive datapaths. The statistics vectors are multiplexed to reduce the number of pins at the block boundary. An external module (Statistics IP0 and/or Statistics IP1) can be designed and implemented in the FPGA logic to accumulate all the TX and RX datapath statistics of each Ethernet MAC. Additional information on interfacing to the statistics block is provided in Chapter 7, "Interfacing to a Statistics Block."

A detailed functional block diagram of a single Ethernet MAC is shown in Figure 2-2. This is labeled as EMAC#, which can represent either EMAC0 or EMAC1 from Figure 2-1.

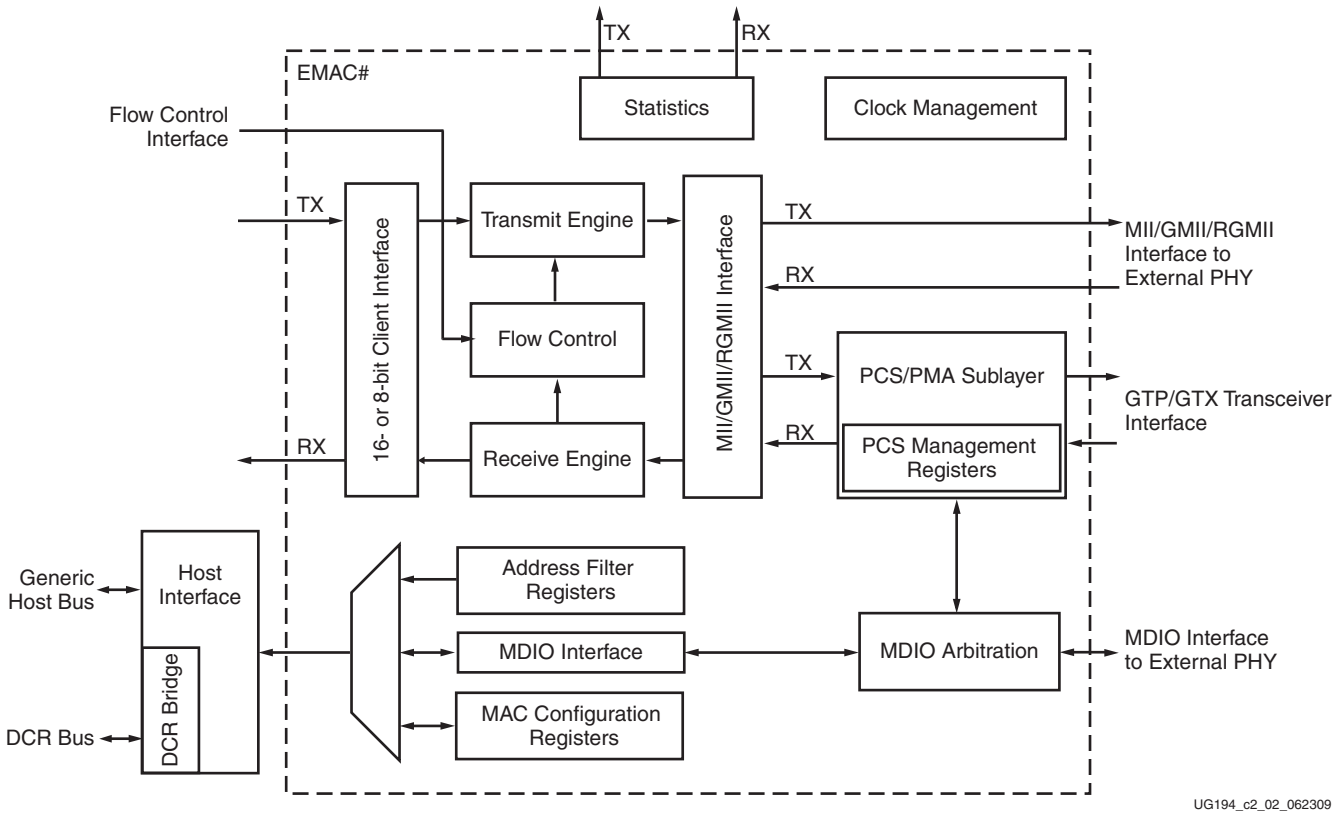


Figure 2-2: **Functional Block Diagram of Single Ethernet MAC (with Host Interface)**

The client interface shows the user transmit and receive interfaces connected to the transmit and receive engines of the Ethernet MAC, respectively. For detailed information on the protocol for data transmission and reception on this interface, refer to "Client Interface," page 51.

The flow control interface allows the client logic to force the physical layer to stop sending frames until the client is capable of accepting more. A description of how to use this interface from the client logic can be found in "Flow Control Block," page 73.

The Ethernet MAC has a programmable address filter to accept or reject incoming frames on the receive datapath. The full functionality of the Ethernet MAC address filter is described in "Address Filtering," page 71.

The shared host interface can access the MAC Configuration registers and the Address Filter registers. See "Host/DCR Bus Interfaces," page 85 to learn how to interface to any host via the generic host bus, or how to interface to a processor through the DCR bus.

The host interface can also initiate transactions on the MDIO interface, when the MDIO interface is enabled. This allows the host interface to access the PCS Management Registers within the Ethernet MAC, when it is configured in 1000BASE-X or SGMII mode. It is also possible for the host interface to initiate transactions via the MDIO interface to an external PHY device. When the MDIO interface is enabled and the host interface is disabled, an external device can still access the PCS Management Registers of the Ethernet MAC when it is configured in 1000BASE-X or SGMII mode. Information on these configurations and on using the MDIO protocol to access PCS Management registers can be found in "MDIO Interface," page 115.

On the physical side of the Ethernet MAC, a common MII/GMII/RGMII interface block is shown. When the physical interface of the Ethernet MAC is configured to MII/GMII or RGMII, these signals can be used via standard I/Os to connect to an external physical interface. When the physical interface is configured in 1000BASE-X or SGMII mode, the PCS/PMA sublayer block in the Ethernet MAC is enabled. The PCS/PMA sublayer block interfaces directly to the RocketIO™ serial transceiver. See "Physical Interface," page 137 for information on all physical interfaces, including clocking schemes and interfacing with serial transceivers.

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).

Ethernet MAC Configuration Options

The Virtex-5 FPGA Embedded Tri-Mode Ethernet MAC addresses all common configuration requirements for embedded Ethernet connectivity. This section provides an overview of the possible options for each Ethernet MAC. A full list of the mode configuration options is in Table 2-16, and further details on the configuration options and how to set these using the CORE Generator™ tool can be found in "Accessing the Ethernet MAC from the CORE Generator Tool," page 25.

The choice of physical interface for the Ethernet MAC affects some of the other configuration options. Table 2-1 provides a breakdown of compatible parameters.

Table 2-1: **Ethernet MAC Block Static Configurations**

Physical Interface	Client Interface		Speed			Advanced Clocking	
	8-Bit	16-Bit	Tri-Speed	1G	10/100M	Clock Enable	Byte PHY
MII only	Y	N	N	N	Y	Y	N
GMII only	Y	N	N	Y	N	N	N
GMII/MII	Y	N	Y	Y	Y	Y (Tri-speed)	Y (Tri-speed)
RGMII	Y	N	Y	Y	Y	Y (Tri-speed)	N
SGMII	Y	N	Y	Y	Y	N	N
1000BASE-X	Y	Y	N	Y	N	N	N

A 16-bit client interface can only be used when the Ethernet MAC physical interface is configured to be 1000BASE-X. 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 possible speed configurations for the Ethernet MAC vary with the physical interface configuration. An initial value for the speed can be chosen for the Ethernet MAC at instantiation, and depending on the mode, this speed can later be modified by writing to the Ethernet MAC mode configuration register.

To minimize resource utilization, advanced (optimized) clocking schemes are available for specific physical interface configurations.

In addition to the configuration options listed in Table 2-1, it is also possible to statically configure other interfaces:

- Out of the pair of Ethernet MACs, either or both can be enabled. To use a single Ethernet MAC, a pair is instantiated and one can be disabled.
- The host interface can be enabled/disabled. This affects both Ethernet MACs.
- Per Ethernet MAC, the MDIO interface can be enabled/disabled.

Within the paired Ethernet MACs, the individual MACs can have different configuration settings.

Ethernet MAC Primitive

The Virtex-5 FPGA Embedded Tri-Mode Ethernet MAC is available to the Xilinx® software tools as a library primitive, named TEMAC. This primitive encompasses the Ethernet MAC block as illustrated in Figure 2-1; it includes a pair of Ethernet MACs (EMAC0 and EMAC1) sharing a common host interface.

The Ethernet MAC primitive can be simplified for specific customer needs by using the CORE Generator tool to create Ethernet MAC wrappers. "Accessing the Ethernet MAC from the CORE Generator Tool," page 25 describes how to use the CORE Generator software to simplify the Ethernet MAC primitive.

Figure 2-3 illustrates the Tri-Mode Ethernet MAC primitive. The # sign denotes that a port of that name exists for each of the Ethernet MACs (EMAC0 and EMAC1) in the TEMAC primitive.

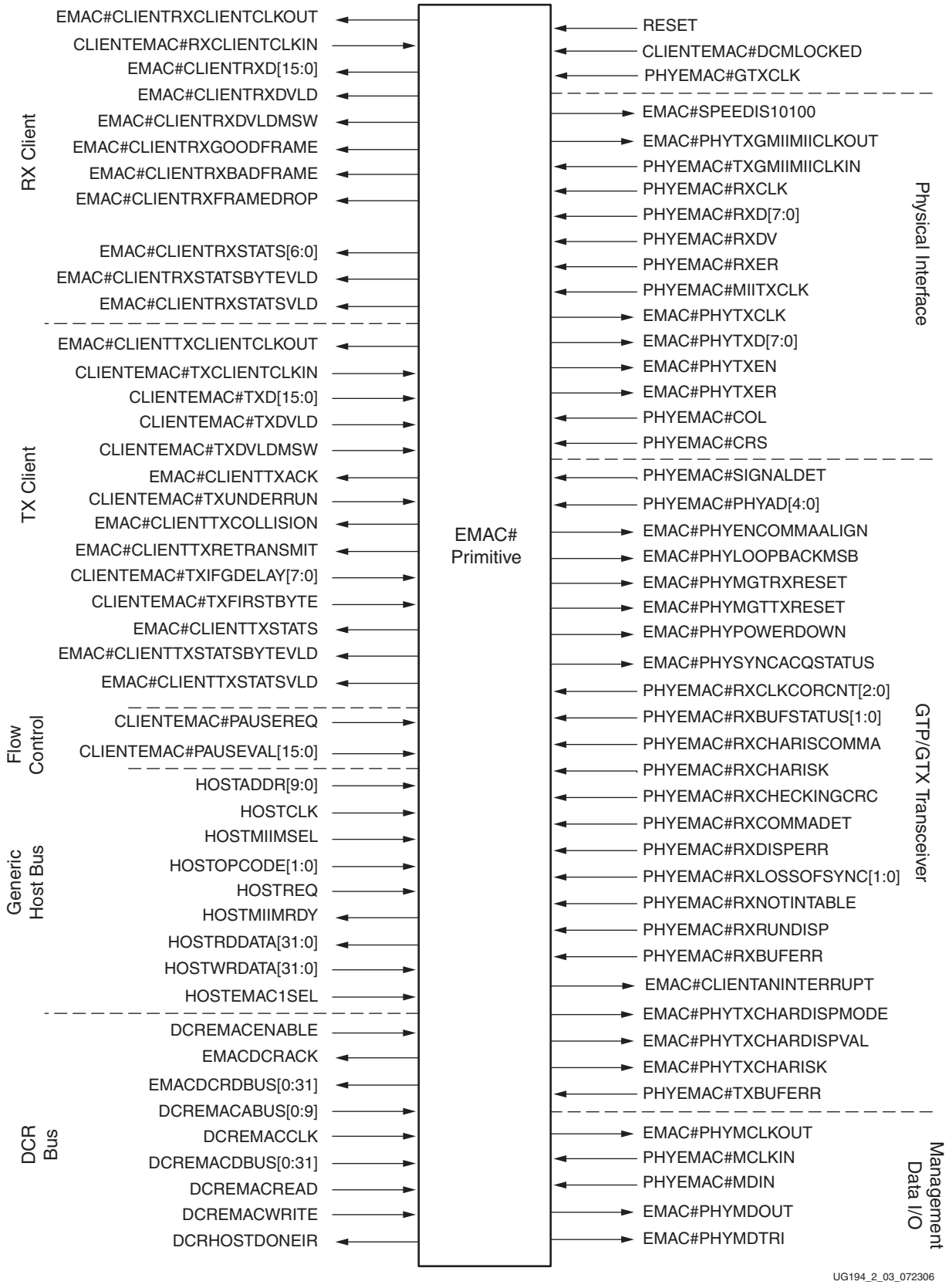


Figure 2-3: **Ethernet MAC Primitive**

Ethernet MAC Signal Descriptions

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

- "Client-Side Signals"
- "Host Interface Signals"
- "Management Data Input/Output (MDIO) Signals"
- "MII/GMII/RGMII Physical Interface Signals"
- "RocketIO Serial Transceiver Signals"
- "Global Clock and Reset Signals"

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

Client-Side Signals

TX Client Signals

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

Signal	Direction	Description
CLIENTEMAC#TXD[15:0]	Input	Frame data for transmit. The datapath 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 Interface," page 60.
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, this signal denotes an odd number of bytes in the transmit datapath. 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, this signal is connected to ground.
CLIENTEMAC#TXFIRSTBYTE	Input	This signal should be tied Low.
CLIENTEMAC#TXIFGDELAY[7:0]	Input	Configurable interframe 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 "Ethernet MAC Clocks," page 205.
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 Across Client Interface," page 53.

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

Signal	Direction	Description
EMAC#CLIENTTXCOLLISION	Output	Asserted by the Ethernet MAC to signal a collision on the medium. Any transmission in progress must 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 resupply 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," page 78.
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," page 78.
EMAC#CLIENTTXCLIENTCLKOUT	Output	See "Ethernet MAC Clocks," page 205.

RX Client Signals

Table 2-3 describes the client-side receive signals. These signals are used by the Ethernet MAC to transfer data to the client.

Table 2-3: **Receive Client Interface Signals**

Signal	Direction	Description
CLIENTEMAC#RXCLIENTCLKIN	Input	See "Ethernet MAC Clocks," page 205.
EMAC#CLIENTRXD[15:0]	Output	Frame data received from the Ethernet MAC. The datapath can be configured to either 8 bits or 16 bits wide. Bits [7:0] are used for 8-bit widths. The 16-bit interface is used in 1000BASE-X PCS/PMA mode. See "Receive (RX) Client: 16-Bit Interface," page 69.
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.
EMAC#CLIENTRXDVLDMSW	Output	This signal denotes an odd number of bytes in the receive datapath when the width of EMAC#CLIENTRXD is set to 16 bits. 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, 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.

Table 2-3: **Receive Client Interface Signals *(Cont'd)***

Signal	Direction	Description
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 7-bits-per-RX clock cycle output for statistics gathering. See "Receiver Statistics Vector," page 81.
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 "Ethernet MAC Clocks," page 205.

Flow Control Signals

Table 2-4 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 specification. Flow control frames received by the Ethernet MAC are automatically handled.

Table 2-4: **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.

Host Interface Signals

Generic Host Bus Signals

Table 2-5 outlines the host bus interface signals.

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

Signal	Direction	Description
HOSTCLK	Input	Clock supplied for running the host.
HOSTOPCODE[1:0]	Input	Defines the operation to be performed over the MDIO interface. Bit 1 is also used in configuration register access. See "Using the Host Bus," page 97.
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 a host processor with the DCR bus, the host bus signals are used to read the optional FPGA logic-based statistics registers. See Chapter 7, "Interfacing to a Statistics Block."

DCR Bus 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.
DCREMACABUS[0:9]	Input	DCR address bus. Bits 0 through 7 must be driven to match DCR base address of one of the Ethernet MACs, as set by the EMAC#_DCRBASEADDR[7:0] attribute. See Table 2-20, page 49 for more information on this attribute.
DCREMACREAD	Input	DCR read request.
DCREMACWRITE	Input	DCR write request.
DCREMACDBUS[0:31]	Input	DCR write data bus.
DCREMACENABLE	Input	When using the DCR bus, this signal is asserted High. When using the host bus interface, the signal is asserted Low.
EMACDCRDBUS[0:31]	Output	DCR read data bus.
EMACDCRACK	Output	DCR acknowledge.
DCRHOSTDONEIR	Output	Interrupt signal to indicate when the Ethernet MAC register access is complete.

Management Data Input/Output (MDIO) Signals

Table 2-7 describes the Management Data Input/Output (MDIO) interface signals. The MDIO format is defined in IEEE Std 802.3, clause 22.

Table 2-7: **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 EMAC block.
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 three-state control to accompany `EMAC#PHYMDOUT`.

MII/GMII/RGMII Physical Interface Signals

The Ethernet MAC has several signals that change definition depending on the selected physical interface configuration. This section describes the physical interface signals when the Ethernet MAC physical interface configuration is set to either MII, GMII, or RGMII.

Data and Control Signals

Table 2-8 shows the data and control signals for the different physical interface modes. The functionality of each of these signals is defined after the static configuration mode is set. This is achieved via attributes, as described in "Ethernet MAC Attributes," page 42.

Table 2-8: **PHY Data and Control Signals**

Signal	Direction	Mode	Description
EMAC#PHYTXEN	Output	GMII	The data enable control signal to the PHY.
		RGMII	
		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 bus to the PHY. EMAC#PHYTXD[7:4] are driven Low.
		GMII	The transmit data signal to the PHY.
		RGMII	EMAC#PHYTXD[3:0] are the RGMII_TXD_RISING signals and EMAC#PHYTXD[7:4] are the RGMII_TXD_FALLING signals to the PHY.
		SGMII	The TX_DATA signal to the RocketIO serial transceiver.
		1000BASE-X	
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 RocketIO serial transceiver.
		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] are the received data signals from the PHY. PHYEMAC#RXD[7:4] are left unconnected.
		GMII	The received data signal to the PHY.
		RGMII	PHYEMAC#RXD[3:0] are the RGMII_RXD_RISING signals and PHYEMAC#RXD[7:4] are the RGMII_RXD_FALLING signals from the PHY.
		SGMII	The RX_DATA signal from the RocketIO serial transceiver.
		1000BASE-X	

Table 2-8: **PHY Data and Control Signals *(Cont'd)***

Signal	Direction	Mode	Description
PHYEMAC#COL	Input	10/100 MII	The collision control signal from the PHY, used in half-duplex mode.
		SGMII	The TXRUNDISP signal from the RocketIO serial transceiver.
		1000BASE-X	
PHYEMAC#CRS	Input	10/100 MII	The carrier sense control signal from the PHY, used in half-duplex mode.

The usage of the Ethernet MAC clock signals associated with the physical interface also changes, based on the selected configuration. This is fully described in Appendix B, "Ethernet MAC Clocks." Table 2-9 lists the physical interface clock signals and references the associated descriptions of each clock.

Table 2-9: **PHY Clock Signals**

Signal	Direction	Description
PHYEMAC#MIITXCLK	Input	See "PHYEMAC#MIITXCLK," page 212.
EMAC#PHYTXCLK	Output	See "EMAC#PHYTXCLK," page 214.
EMAC#PHYTXGMIIMIICLKOUT	Output	See "EMAC#PHYTXGMIIMIICLKOUT, PHYEMAC#TXGMIIMIICLKIN," page 213.
PHYEMAC#TXGMIIMIICLKIN	Input	See "EMAC#PHYTXGMIIMIICLKOUT, PHYEMAC#TXGMIIMIICLKIN," page 213.
PHYEMAC#RXCLK	Input	See "PHYEMAC#RXCLK," page 212.

Table 2-10 shows Ethernet MAC output that indicates the current operating speed of the Ethernet MAC.

Table 2-10: **Physical Interface Speed Signal**

Signal	Direction	Description
EMAC#SPEEDIS10100	Output	When asserted High, the physical interface is operating at 10 Mb/s or 100 Mb/s. When asserted Low, the physical interface is operating at 1 Gb/s.

RocketIO Serial Transceiver Signals

RocketIO serial transceivers are defined by device family in the following way:

- Virtex-5 LXT and SXT devices: RocketIO GTP transceivers
- Virtex-5 TXT and FXT devices: RocketIO GTX transceivers

RocketIO GTP serial transceivers are placed as dual transceiver GTP_DUAL tiles in Virtex-5 LXT and SXT devices. RocketIO GTX serial transceivers are placed as dual transceiver GTX_DUAL tiles in Virtex-5 TXT and FXT devices. Throughout this guide, the term RocketIO transceiver is used to represent any or all of the RocketIO transceivers; the RocketIO transceiver that is specific to the desired target device should be selected.

Table 2-11 shows the signals used to connect the Virtex-5 FPGA Ethernet MAC to the RocketIO serial transceiver (see the UG196, *Virtex-5 FPGA RocketIO GTP Transceiver User Guide* and UG198, *Virtex-5 FPGA RocketIO GTX Transceiver User Guide*).

Table 2-11: **RocketIO Serial Transceiver Connections**

Signal	Direction	Description
EMAC#PHYENCOMMAALIGN	Output	Enable RocketIO serial transceiver PMA layer to realign to commas.
EMAC#PHYLOOPBACKMSB	Output	Perform loopback testing in the RocketIO serial transceiver from TX to RX.
EMAC#PHYMGTRXRESET	Output	Reset to RocketIO serial transceiver RXRESET.
EMAC#PHYMGTTXRESET	Output	Reset to RocketIO serial transceiver TXRESET.
EMAC#PHYPOWERDOWN	Output	Power-down the RocketIO serial transceiver.
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 2-12 shows the PCS/PMA signals.

Table 2-12: **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.

Global Clock and Reset Signals

Global Clock Signal

Table 2-13 shows the global clock signal that is necessary for most configurations of the Virtex-5 FPGA Ethernet MAC. See Appendix B, "Ethernet MAC Clocks" for further details.

Table 2-13: **Global 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.

Reset Signal

Table 2-14 describes the Reset signal.

Table 2-14: **Reset Signal**

Signal	Direction	Description
Reset	Input	Asynchronous reset of the entire Ethernet MAC.

CLIENTEMAC#DCMLOCKED Signal

Table 2-15 describes the CLIENTEMAC#DCMLOCKED signal.

Table 2-15: **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 the RocketIO serial transceiver PLL is used, the PLLLKDET signal should be ANDed with the DCM LOCK signal. If a DCM is not used, both CLIENTEMAC#DCMLOCKED ports from EMAC0 and EMAC 1 must be tied High. If any Ethernet MAC is not used, CLIENTEMAC#DCMLOCKED must be tied to High.

Ethernet MAC Attributes

This section describes the attributes (control parameters) that are used to configure the Virtex-5 FPGA Embedded Tri-Mode Ethernet MAC. Some of these attributes control the static configuration of the Ethernet MAC and others are used to preconfigure the internal control registers. The values of these attributes are loaded into the Ethernet MAC at power-up or when the Ethernet MAC is reset(1).

The attributes are divided into five sections: mode configuration, MAC configuration, physical interface, address filter, and DCR base address attributes.

The mode configuration attributes control the static behavior of the Ethernet MAC, including whether the Ethernet MAC has a host interface, the type of physical interface, the width of the client interface and whether the MDIO interface is enabled. These attributes are not dynamically reconfigurable.

All other attributes described in Table 2-16 (unless stated otherwise) define default values for internal registers. If the host interface is not enabled on the Ethernet MAC, these attributes allow the user to set the internal control register defaults. If the host interface is enabled (by setting the EMAC#_HOST_ENABLE attribute to be TRUE), then the host interface can be used to dynamically change the associated register contents or to read the register contents.

1. An attribute value of TRUE is translated into a bit value of 1; a value of FALSE equals a bit value of 0.

Table 2-16: **Mode Configuration Attributes**

Attribute	Type	Description
Attribute to Configure Default Register Values for Management Configuration Register		See "Management Configuration Register," page 94 for register details. The MDIO interface can be enabled/disabled using the host interface, when the host interface is enabled. (The host interface is enabled by setting the EMAC#_HOST_ENABLE attribute to TRUE). If the host interface is not enabled then the MDIO enable value is fixed by this attribute.
EMAC#_MDIO_ENABLE	Boolean	When EMAC#_HOST_ENABLE is FALSE, setting this attribute to TRUE enables the use of MDIO in the Ethernet MAC. When EMAC#_HOST_ENABLE and EMAC#_MDIO_ENABLE are both TRUE, the MDIO interface is enabled when the MDIO enable bit is set in the Ethernet MAC Management Configuration Register. See "MDIO Interface," page 115.
Attributes to Configure Fixed Mode for the Ethernet MAC		See "Ethernet MAC Mode Configuration Register," page 92 for register details.
EMAC#_HOST_ENABLE	Boolean	Setting this attribute to TRUE enables the use of the Ethernet MAC host interface via either the generic host bus or the DCR bus.
EMAC#_TX16BITCLIENT_ENABLE	Boolean	When this attribute is set to TRUE, the TX client data interface is 16 bits wide. When this attribute is set to FALSE, the TX client data interface is 8 bits wide.
EMAC#_RX16BITCLIENT_ENABLE	Boolean	When this attribute is set to TRUE, the RX client data interface is 16 bits wide. When this attribute is set to FALSE, the RX client data interface is 8 bits wide.
Attributes to Configure Fixed Physical Interface for the Ethernet MAC		The following three attributes define the physical interface of the Ethernet MAC. These attributes are mutually exclusive. If none of these three attributes is set to TRUE, then the physical interface is enabled for 10/100 MII and GMII modes. See "Ethernet MAC Mode Configuration Register," page 92 for register details.
EMAC#_RGMII_ENABLE	Boolean	Setting this attribute to TRUE enables the Ethernet MAC for RGMII mode.
EMAC#_SGMII_ENABLE	Boolean	Setting this attribute to TRUE enables the Ethernet MAC for SGMII mode.
EMAC#_1000BASEX_ENABLE	Boolean	Setting this attribute to TRUE enables the Ethernet MAC for 1000BASE-X mode.

Table 2-16: **Mode Configuration Attributes *(Cont'd)***

Attribute	Type	Description
Attributes to Configure Fixed Speed for the Ethernet MAC		These attributes determine the speed of the Ethernet MAC after reset or power-up. The speed of the Ethernet MAC can be changed in multi-speed configurations using the host interface, when the host interface is enabled. (The host interface is enabled by setting the EMAC#_HOST_ENABLE attribute to TRUE). If the host interface is not enabled, then the speed of the Ethernet MAC is directly set by these two attributes. (EMAC#_SPEED_MSB: EMAC#_SPEED_LSB) `1:0` = 1000 Mb/s `0:1` = 100 Mb/s `0:0` = 10 Mb/s `1:1` = not applicable See "Ethernet MAC Mode Configuration Register," page 92 for register details.
EMAC#_SPEED_MSB	Boolean	EMAC#_SPEED[1]. (TRUE takes the bit value `1` in the table cell above.)
EMAC#_SPEED_LSB	Boolean	EMAC#_SPEED[0]. (TRUE takes the bit value `1` in the table cell above.)
Attributes to Configure Advanced Clocking Scheme for the Ethernet MAC		To reduce the number of BUFGs used for the Ethernet MAC, optimized clocking modes are available. See "Clock Connections to and from FPGA Logic," page 206 for a description of the possible clocking mode options.
EMAC#_BYTEPHY	Boolean	When this attribute is set to TRUE, the GMII/MII interface bus remains the same as GMII interface. Specifically, it is 8 bits wide and runs at the EMAC block's ½ clock rate of 12.5 MHz or 1.25 MHz. See "Advanced Clocking Schemes," page 208 for an introduction to this clocking mode option.
EMAC#_USECLKEN	Boolean	When this attribute is set to TRUE, the client interface clock output is switched to clock enable, thus eliminating BUFGs on the client interface. See "Advanced Clocking Schemes," page 208 for an introduction to this clocking mode option.

Table 2-17: **MAC Configuration Attributes**

Attribute	Type	Description
Attribute to Configure Default Register Value for the Address Filter Mode Register		The address filter can be enabled/disabled using the host interface, when the host interface is enabled. (The host interface is enabled by setting the EMAC#_HOST_ENABLE attribute to TRUE). If the host interface is not enabled, the address filter mode is fixed by the following attribute value. See "Address Filter Mode," page 97 for register details.
EMAC#_ADDRFILTER_ENABLE	Boolean	Address Filter Enable: This attribute sets the default value of the Promiscuous Mode enable bit in the Address Filter Mode register. Setting this attribute to TRUE enables the use of the address filter module in the Ethernet MAC.
Attributes to Configure Default Register Value for the Flow Control Configuration Register		The following two attributes configure the Ethernet MAC flow control module. See "Flow Control Configuration Register," page 91 for register details.
EMAC#_RXFLOWCTRL_ENABLE	Boolean	Receive Flow Control Enable: This attribute sets the default value of the Flow control enable (RX) bit in the Flow Control Configuration Register. When this attribute is set to TRUE and full-duplex mode is enabled, receive flow control is enabled. Any flow control frames received by the Ethernet MAC cause the transmitter operation to be inhibited. When this attribute is set to FALSE, any flow control frames received by the Ethernet MAC are passed on to the client. See "Flow Control Block," page 73 for description of flow control functionality.
EMAC#_TXFLOWCTRL_ENABLE	Boolean	Transmit Flow Control Enable: This attribute sets the default value of the Flow control enable (TX) bit in the Flow Control Configuration Register. When this attribute is set to TRUE and full-duplex mode is enabled, asserting the `CLIENTEMAC#PAUSEREQ` signal causes the Ethernet MAC to send a flow control frame out from the transmitter. When this attribute is set to FALSE, asserting the `CLIENTEMAC#PAUSEREQ` signal has no effect.
Attributes to Configure Default Register Value for the Transmitter Configuration Register		The following attributes configure the transmit engine of the Ethernet MAC. See "Transmitter Configuration Register," page 90 for register details.
EMAC#_TXRESET	Boolean	This attribute sets the default value of the Reset bit in the Transmitter Configuration Register. When this attribute is set to TRUE, the Ethernet MAC transmitter is reset. This reset also sets all of the transmitter configuration registers to their default values.

Table 2-17: **MAC Configuration Attributes *(Cont'd)***

Attribute	Type	Description
EMAC#_TXJUMBOFRAME_ENABLE	Boolean	This attribute sets the default value of the Jumbo Frame Enable bit in the Transmitter Configuration Register. When this attribute is set to TRUE, the Ethernet MAC transmitter allows frames larger than the maximum legal frame length specified in IEEE Std 802.3-2002 to be sent. When FALSE, the Ethernet MAC transmitter only allows frames up to the legal maximum to be sent.
EMAC#_TXINBANDFCS_ENABLE	Boolean	This attribute sets the default value of the In-Band FCS Enable bit in the Transmitter Configuration Register. When this attribute is set to TRUE, the Ethernet MAC transmitter expects the FCS field to be passed in by the client. When FALSE, the Ethernet MAC transmitter appends padding as required, computes the FCS, and appends it to the frame.
EMAC#_TX_ENABLE	Boolean	This attribute sets the default value of the Transmit Enable bit in the Transmitter Configuration Register. When this attribute is set to TRUE, the transmitter is operational. When FALSE, the transmitter is disabled.
EMAC#_TXVLAN_ENABLE	Boolean	This attribute sets the default value of the VLAN Enable bit in the Transmitter Configuration Register. When this attribute is set to TRUE, the transmitter allows the transmission of VLAN tagged frames.
EMAC#_TXHALFDUPLEX	Boolean	This attribute sets the default value of the Half Duplex mode bit in the Transmitter Configuration Register. When this attribute is set to TRUE, the transmitter operates in half-duplex mode. When FALSE, the transmitter operates in full-duplex mode.
EMAC#_TXIFGADJUST_ENABLE	Boolean	This attribute sets the default value of the IFG Adjustment Enable bit in the Transmitter Configuration Register. When this attribute is set to TRUE, the transmitter reads the value of the `CLIENTEMAC#TXIFGDELAY[7:0]` port and sets the IFG accordingly. When FALSE, the transmitter always inserts at least the legal minimum IFG.
Attributes to Configure Default Register Value for the Receiver Configuration Registers		Configures the receive engine of the Ethernet MAC. See "Receiver Configuration Register (Word 0)" and "Receiver Configuration Register (Word 1)," page 89 for register details.
EMAC#_RXRESET	Boolean	This attribute sets the default value of the Reset bit in the Receiver Configuration Register. When this attribute is set to TRUE, the receiver is held in reset. This signal is an input to the reset circuit for the receiver block.
EMAC#_RXJUMBOFRAME_ENABLE	Boolean	This attribute sets the default value of the Jumbo Frame Enable bit in the Receiver Configuration Register. When this attribute is set to FALSE, the receiver does not pass frames longer than the maximum legal frame size specified in IEEE Std 802.3-2002. When TRUE, the receiver does not have an upper limit on frame size.

Table 2-17: **MAC Configuration Attributes *(Cont'd)***

Attribute	Type	Description
EMAC#_RXINBANDFCS_ENABLE	Boolean	This attribute sets the default value of the In-band FCS Enable bit in the Receiver Configuration Register. When this attribute is set to TRUE, the Ethernet MAC receiver passes the FCS field up to the client. When FALSE, the Ethernet MAC receiver does not pass the FCS field. In both cases, the FCS field is verified on the frame.
EMAC#_RX_ENABLE	Boolean	This attribute sets the default value of the Receive Enable bit in the Receiver Configuration Register. When this attribute is set to TRUE, the receiver block is operational. When FALSE, the block ignores activity on the physical interface RX port.
EMAC#_RXVLAN_ENABLE	Boolean	This attribute sets the default value of the VLAN Enable bit in the Receiver Configuration Register. When this attribute is set to TRUE, VLAN tagged frames are accepted by the receiver.
EMAC#_RXHALFDUPLEX	Boolean	This attribute sets the default value of the Half Duplex mode bit in the Receiver Configuration Register. When this attribute is set to TRUE, the receiver operates in half-duplex mode. When FALSE, the receiver operates in full-duplex mode.
EMAC#_LTCHECK_DISABLE	Boolean	This attribute sets the default value of the Length/Type Check Disable bit in the Receiver Configuration register. Setting this attribute to TRUE disables the Length/Type check on received frames where the Length/Type field is less than `0x0600`. See "Length/Type Field Error Checks" for more information.
EMAC#_PAUSEADDR[47:0]	48-bit Binary	This attribute sets the default value of the Pause Frame Ethernet MAC Address [47:0] in the Receiver Configuration (Word 0) and (Word 1) registers. 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`. The Pause frame address should be set to the same Ethernet MAC address as the EMAC#_UNICASTADDR[47:0] attributes.

Table 2-18: **Physical Interface Attributes**

Attribute	Type	Description
EMAC#_CONFIGVEC_79	Boolean	Reserved, set to TRUE.
Attributes to Configure Default Register Values for SGMII or 1000BASE-X Management Registers		The following attributes are only used in SGMII or 1000BASE-X modes. See "1000BASE-X PCS/PMA Management Registers," page 122 and "SGMII Management Registers," page 130 for register details. See "MDIO Implementation in the Ethernet MAC," page 119 for information on how to access these registers.
EMAC#_PHYRESET	Boolean	This attribute sets the default value of the Reset bit in the PCS/PMA Management Register 0 (Control Register). A value of TRUE resets the PCS/PMA module.
EMAC#_PHYINITAUTONEG_ENABLE	Boolean	This attribute sets the default value of the Auto-Negotiation Enable bit in the PCS/PMA Management Register 0 (Control Register). A value of TRUE enables auto-negotiation of the PCS/PMA module.
EMAC#_PHYISOLATE	Boolean	This attribute sets the default value of the Isolate bit in the PCS/PMA Management Register 0 (Control Register). A value of TRUE 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.
EMAC#_PHYPOWERDOWN	Boolean	This attribute sets the default value of the Power Down bit in the PCS/PMA Management Register 0 (Control Register). A value of TRUE causes the RocketIO serial transceiver to be placed in a Low power state. A reset must be applied to clear the Low power state.
EMAC#_PHYLOOPBACKMSB	Boolean	This attribute sets the default value of the Loopback bit in the PCS/PMA Management Register 0 (Control Register). A value of TRUE for this attribute enables loopback. EMAC#_GTLOOPBACK sets if the loopback occurs in the Ethernet MAC or in the RocketIO serial transceiver.
EMAC#_UNIDIRECTION_ENABLE	Boolean	This attribute sets the default value of the unidirection enable bit in the PCS/PMA Management Register 0 (Control Register). Setting this attribute to TRUE allows the Ethernet MAC to transmit regardless of whether a valid link has been established.
EMAC#_GTLOOPBACK	Boolean	This attribute sets the default value of the GT Loopback bit in "Vendor-Specific Register: Loopback Control Register (Register 17)," page 129. This attribute controls where the loopback happens when loopback is selected. When EMAC#_GTLOOPBACK is TRUE, the loopback occurs in the RocketIO serial transceiver. Near-End parallel PCS loopback is used. When EMAC#_GTLOOPBACK is FALSE, the loopback is internal to the EMAC. In this state, idles are transmitted to the RocketIO serial transceiver.

Table 2-18: **Physical Interface Attributes *(Cont'd)***

Attribute	Type	Description
Attribute to Configure Auto-negotiation Parameters		See the following sections for details on attribute value settings for Physical interface settings of SGMII or 1000BASE-X: • SGMII: "Auto-Negotiation Link Timer," page 197. • 1000BASE-X: "Auto-Negotiation Link Timer," page 175.
EMAC#_LINKTIMERVAL[8:0]	9-bit Binary	Programmable auto-negotiation link timer value.

Table 2-19: **Address Filter Attributes**

Attribute	Type	Description
Attribute to Configure Default Register Value for the Unicast Address Register		This attribute sets the default value of the Unicast Address[47:0] in the Unicast Address (Word 0) and (Word 1) registers. See "Unicast Address (Word 0)," page 95 and "Unicast Address (Word 1)," page 95 for register details.
EMAC#_UNICASTADDR[47:0]	48-bit Binary	This 48-bit attribute 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 bits [47:0] as `0x010203040506`. This default value for the unicast address can be overridden by software through the host interface.

Table 2-20: **DCR Base Address Attributes**

Attribute	Type	Description
Attribute to Configure Default Value for DCR Base Address		See "Using the DCR Bus," page 101 for a description of how the DCR Base Addresses are used to access different registers.
EMAC#_DCRBASEADDR[7:0]	8-bit Binary	This attribute allows separate DCR base addresses to be used for each Ethernet MAC.

XILINX®

Chapter 3

Client Interface

This chapter provides useful design information for the Virtex®-5 Ethernet MAC. It contains the following sections:

- "Transmit (TX) Client: 8-Bit Interface (without Clock Enables)"
- "Transmit (TX) Client: 8-Bit Interface (with Clock Enables)"
- "Transmit (TX) Client: 16-Bit Interface"
- "Receive (RX) Client: 8-Bit Interface (without Clock Enables)"
- "Receive (RX) Client: 8-Bit Interface (with Clock Enables)"
- "Receive (RX) Client: 16-Bit Interface"
- "Address Filtering"
- "Flow Control Block"
- "Statistics Vectors"

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

The transmit and receive client interfaces can be configured to handle either 8-bit data transfers or 16-bit data transfers, where the default is 8 bits.

The 8-bit client operation supports all physical interfaces and is offered in two alterative clocking modes: the standard clocking scheme without clock enables or an alternative clock enable scheme. "Ethernet MAC Clocks," page 205 introduces these two clocking models and describes how the clock enable scheme can be used to reduce the number of global clock buffers required for the design. Table 2-1, page 30 illustrates the physical interface configurations that can use the advanced clock enable scheme.

The 16-bit client operation is enabled by setting the EMAC#_RX16BITCLIENT_ENABLE and EMAC#_TX16BITCLIENT_ENABLE attributes to TRUE. This mode of operation is only available with the 1000BASE-X PCS/PMA physical interface. Using this scheme, the Ethernet MAC can be over-clocked to provide operation above 1 Gb/s. "16-Bit Data Client," page 170 provides a description of the clocking scheme for this mode.

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

Transmit (TX) Client: 8-Bit Interface (without Clock Enables)

This section covers the client transmit interface when it is configured to be eight bits wide and when the default clocking scheme is used. "Ethernet MAC Attributes," page 42 describes the EMAC#_USECLKEN attribute, which is set to FALSE for this configuration. In this configuration, `CLIENTEMAC#TXD[15:8]` and `CLIENTEMAC#TXDVLDMSW` must be tied to ground because the upper eight bits of the data bus are not necessary.

See "Ethernet MAC Clocks," page 205 for some general information on this standard clocking scheme. More details are available for specific physical interface configurations in the following sections:

- MII
  - "MII Standard Clock Management"
- GMII
  - "GMII Clock Management for 1 Gb/s Only"
  - "GMII Standard Clock Management for Tri-Speed Operation"
  - "GMII Clock Management for Tri-Speed Operation Using Byte PHY"
- RGMII
  - "RGMII Clock Management for 1 Gb/s Only"
  - "RGMII Standard Clock Management for Tri-Speed Operation"
- SGMII
  - "SGMII Clock Management (LXT and SXT Devices)"
  - "SGMII Clock Management (TXT and FXT Devices)"
- 1000BASE-X
  - "1000BASE-X PCS/PMA Clock Management (LXT and SXT Devices)"
  - "1000BASE-X PCS/PMA Clock Management (TXT and FXT Devices)"

Normal Frame Transmission

The timing of a normal outbound frame transfer is shown in Figure 3-1. 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.

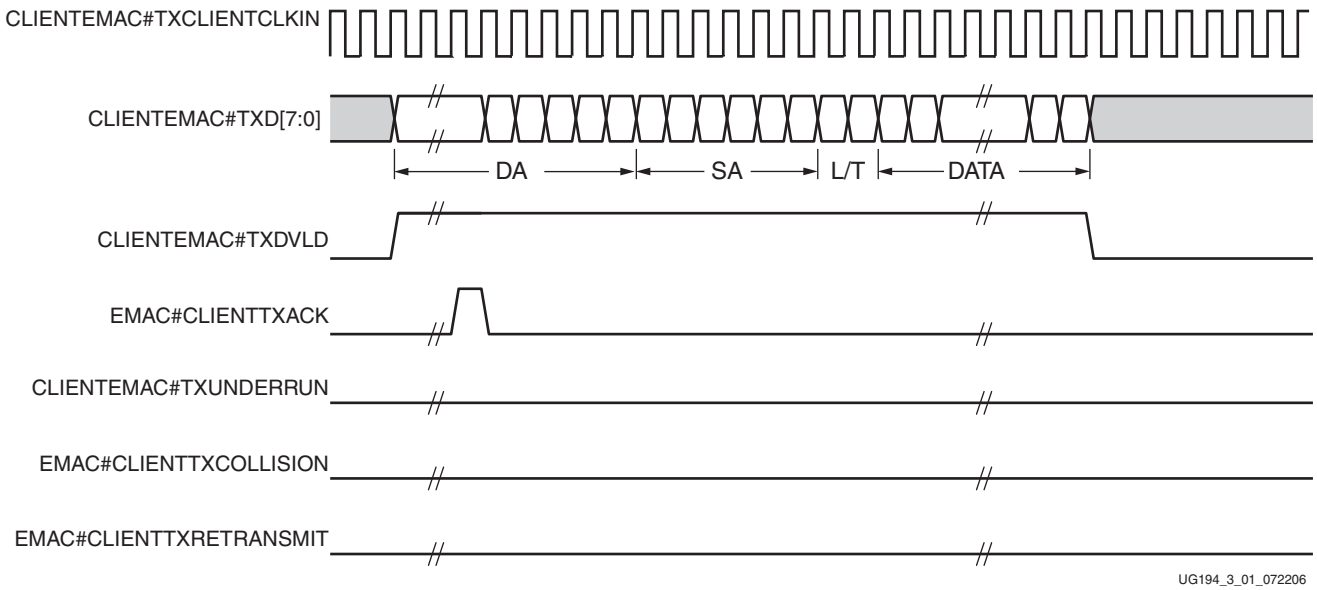


Figure 3-1: **Normal Frame Transmission Across Client Interface**

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. The preamble, and optionally the FCS, are added to the frame by the Ethernet MAC.

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 53).

Client-Supplied FCS Passing

In the transmission timing case shown in Figure 3-2, an Ethernet MAC is configured to have the FCS field passed in by the client (see "Configuration Registers," page 88). The client must ensure that the frame meets the Ethernet minimum frame length requirements. If the frame does not meet these requirements, the Ethernet MAC pads the frame to the minimum frame length. Although this does not cause the transmitter statistics vector to indicate a bad frame, it will result in an errored frame as received by the link partner Ethernet MAC.

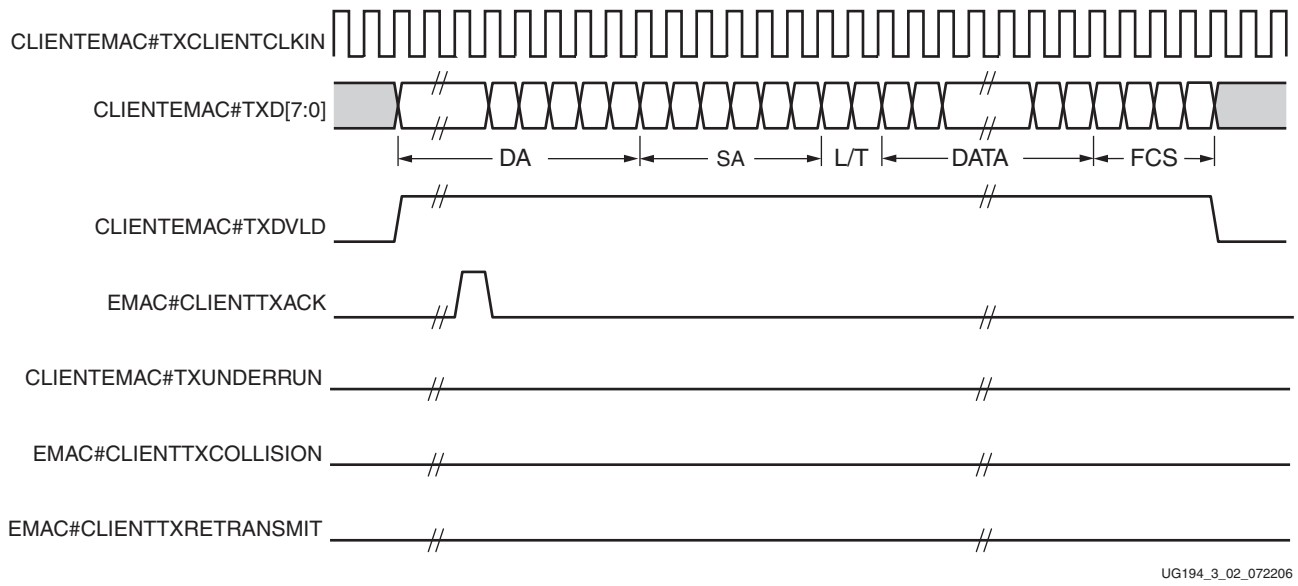


Figure 3-2: **Frame Transmission with Client-Supplied FCS**

Client Underrun

The timing of an aborted transfer is shown in Figure 3-3. An aborted transfer can occur if a FIFO connected to the client interface empties before a frame is completed. The CLIENTEMAC#TXUNDERRUN signal on the client interface is used to signal to the MAC that an underrun condition exists.

When the client asserts CLIENTEMAC#TXUNDERRUN during a frame transmission, the EMAC#PHYTXER output is asserted for one clock cycle to notify the external GMII, RGMII, or MII PHY that the frame is corrupted.

If 1000BASE-X PCS/PMA or SGMII mode is operational, the MAC inserts an error code into the current frame to signal corruption. The Ethernet MAC then falls back to idle transmission. The client must requeue the aborted frame for transmission.

After an underrun occurs, CLIENTEMAC#TXDVLD must be deasserted for at least one clock cycle before transmission can recommence. CLIENTEMAC#TXDVLD can be deasserted on the same cycle as underrun is asserted. There is no requirement to deassert CLIENTEMAC#TXDVLD when underrun is asserted; it can be deasserted later.

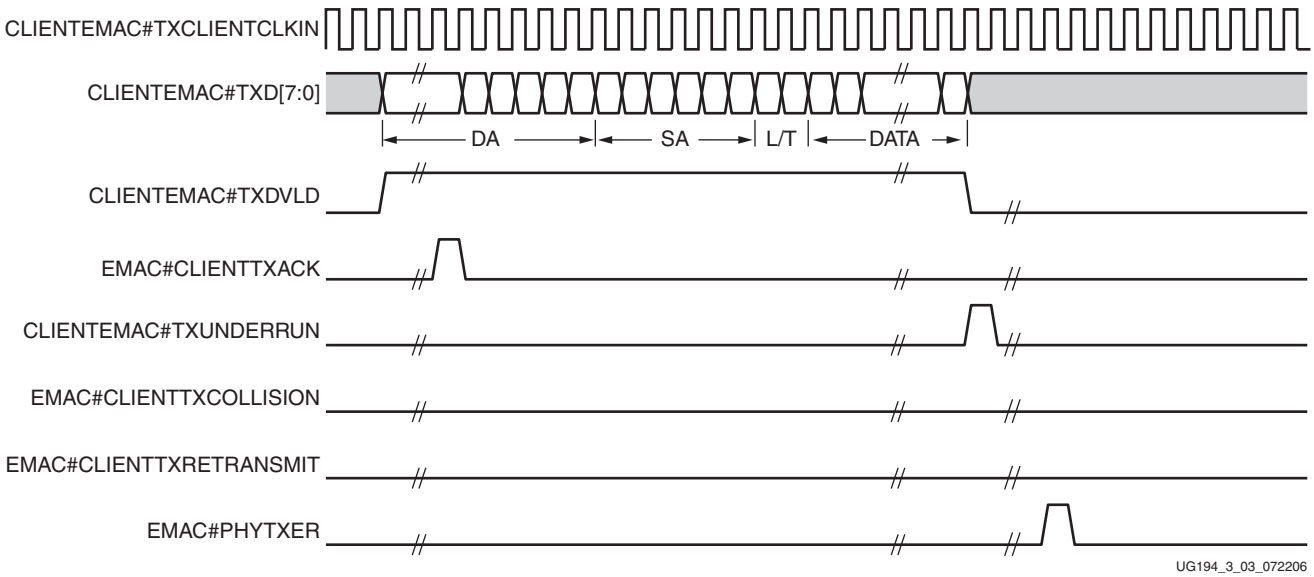


Figure 3-3: **Frame Transmission with Underrun**

Back-to-Back Transfers

Back-to-back transfers can occur when the Ethernet MAC client is ready to transmit a second frame of data immediately following completion of the first frame. In Figure 3-4, 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, indicating that the first byte of the destination address of the second frame is awaiting transmission on `CLIENTEMAC#TXD`.

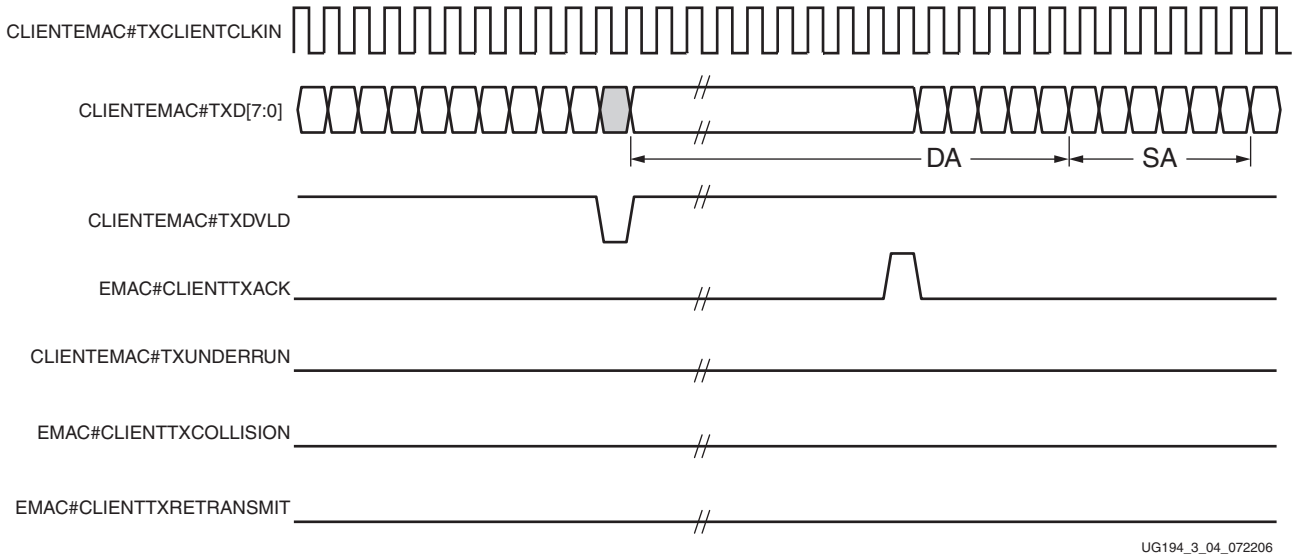


Figure 3-4: **Back-to-Back Frame Transmission**

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 3-5 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 positions in the data stream. More information on the contents of these two bytes can be found in the IEEE Std 802.3-2002 specification.

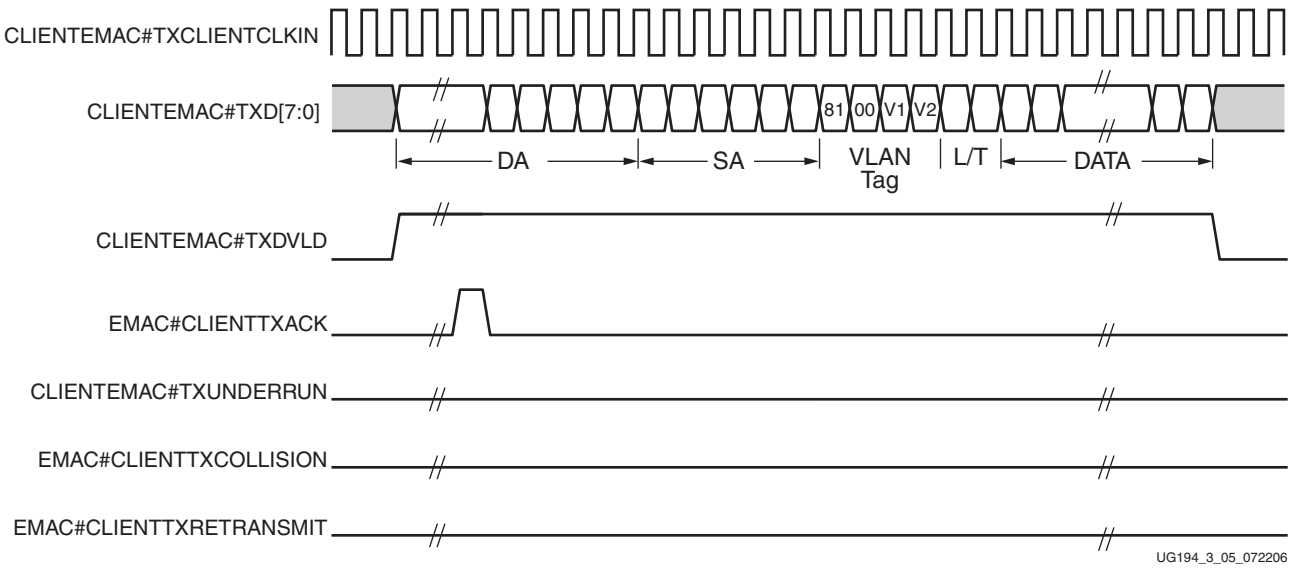


Figure 3-5: **Transmission of a VLAN Tagged Frame**

Maximum Permitted Frame Length/Jumbo Frames

The maximum length of a frame specified in the IEEE Std 802.3-2002 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 88.

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 resupplied as follows:

- If there is a collision, the `EMAC#CLIENTTXCOLLISION` signal is set High by the Ethernet MAC. If a frame is in progress, the client must abort the transfer, and `CLIENTEMAC#TXDVLD` is deasserted Low.
- If the `EMAC#CLIENTTXRETRANSMIT` signal is High in the same clock cycle that the `EMAC#CLIENTTXCOLLISION` signal is High, 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 3-6. If the frame presented to the client interface is shorter than the collision window (slot time) defined in IEEE Std 802.3-2002, a retransmission request can occur after the end of the frame on the client interface. This can also occur on slightly longer frames due to the latency of the transmit section of the Ethernet MAC. In half duplex mode, the client should wait until the collision window has passed before submitting a new frame.

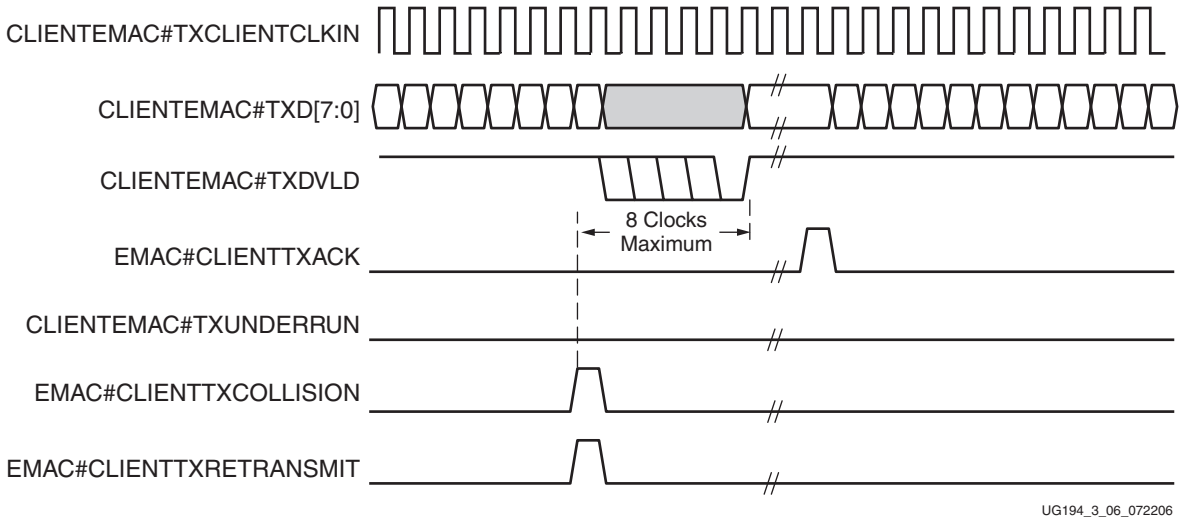


Figure 3-6: **Collision Handling - Frame Retransmission Required**

- If the EMAC#CLIENTTXRETRANSMIT signal is Low in the same clock cycle that the EMAC#CLIENTTXCOLLISION signal is High, the number of retries for this frame has exceeded the Ethernet specification, and the frame should be dropped by the client. The user must drop TXDVLD the cycle after TXCOLLISION is asserted. The client can then make any new frame available to the Ethernet MAC for transmission without timing restriction. This process is shown in Figure 3-7.

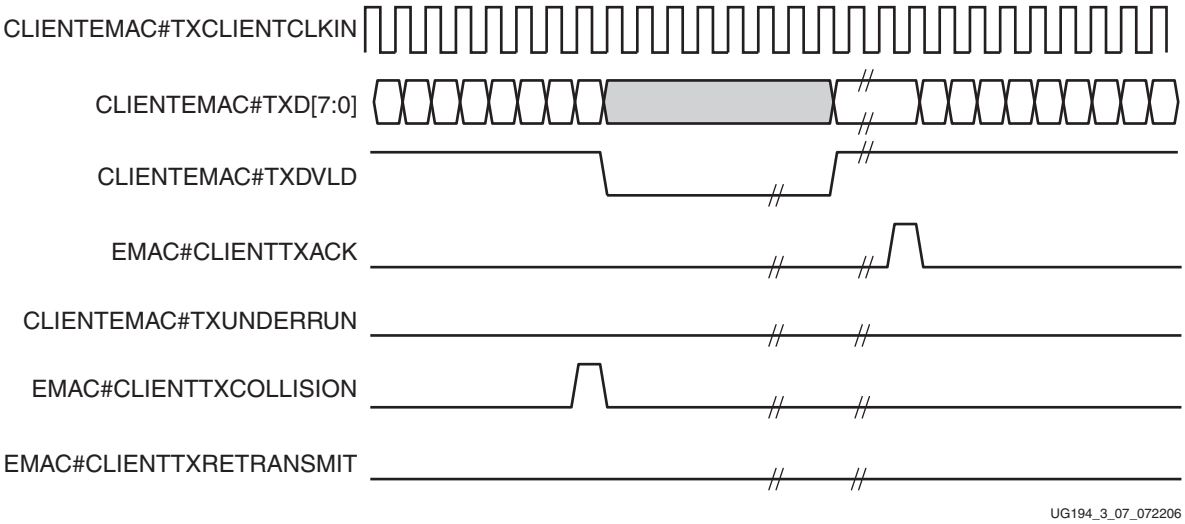


Figure 3-7: **Collision Handling - No Frame Retransmission Required**

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 88), 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 can be increased. The number of cycles is controlled by the value on the `CLIENTEMAC#TXIFGDELAY` port at the start-of-frame transmission if the IFG Adjustment Enable bit in the Transmitter Configuration Register is set. If IFG adjustment is disabled or the value on `CLIENTEMAC#TXIFGDELAY` is less than the minimum IFG specified in IEEE 802.3, the minimum IFG (12 idles) is used. Figure 3-8 shows the Ethernet MAC operating in this mode.

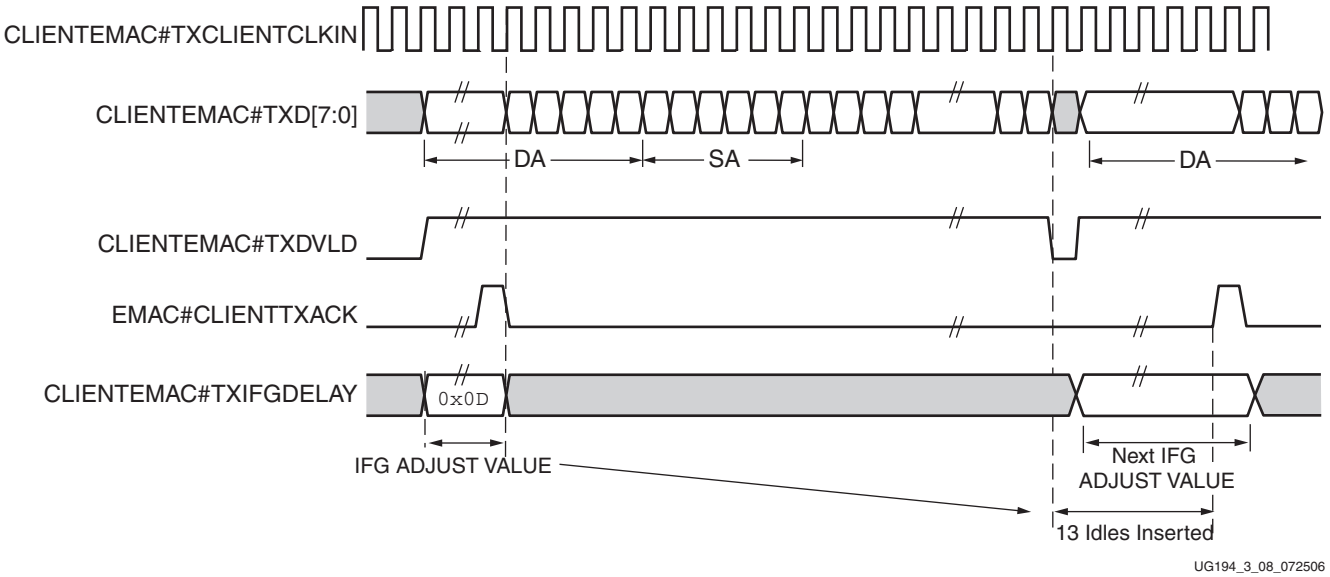


Figure 3-8: **IFG Adjustment**

Transmit (TX) Client: 8-Bit Interface (with Clock Enables)

This section covers the client transmit interface when it is configured to be 8 bits wide and when the optional clock enable clocking scheme is used. This optional configuration can be used when the Ethernet MAC is configured in MII/GMII or RGMII mode. "Ethernet MAC Attributes," page 42 describes the EMAC#_USECLKEN attribute, which is set to TRUE for this configuration.

In this mode, the client interface is synchronous to the physical interface clock. This configuration reduces the number of BUFGs that are used to implement the Ethernet MAC design. See "Ethernet MAC Clocks," page 205 for some general information on this clocking scheme. More details are available for specific physical interface configurations in the following sections:

- "MII Clock Management using Clock Enables"
- "GMII Clock Management for Tri-Speed Operation Using Clock Enables"
- "RGMII Clock Management for Tri-Speed Operation Using Clock Enables"

For the transmit interface, the `PHYEMAC#GMIIMIICLKIN` signal is used as the clock and the `EMAC#CLIENTTXCLIENTCLKOUT` signal is used as the clock enable.

Normal Frame Transmission

Figure 3-9 shows the transmission of a normal frame at 1 Gb/s. At 1 Gb/s, no clock enable is required, and the clock enable signal EMAC#CLIENTTXCLIENTCLKOUT is held High. The data is input on the rising edge of the 125 MHz PHYEMAC#GMIIMIICLKIN clock.

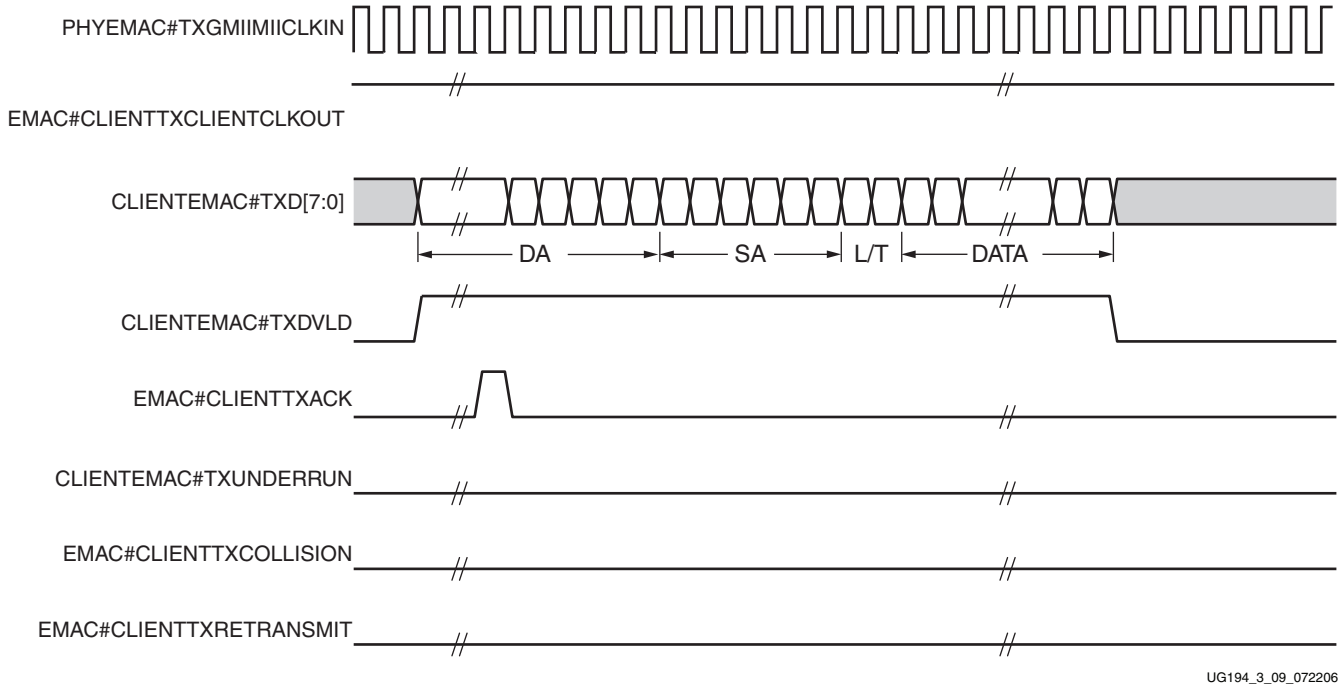


Figure 3-9: **Normal Frame Transmission at 1 Gb/s with EMAC#_USECLKEN Set to TRUE**

At speeds slower than 1 Gb/s, the physical interface clock runs twice as fast as the client interface clock. The data is still synchronous to PHYEMAC#GMIIMIICLKIN, but to maintain the correct data rate, the EMAC#CLIENTTXCLIENTCLKOUT clock signal is used as an enable rather than a clock. The EMAC#CLIENTTXCLIENTCLKOUT signal is asserted every second cycle of PHYEMAC#GMIIMIICLKIN. Data is presented to the transmitter interface only when EMAC#CLIENTTXCLIENTCLKOUT is High. This is illustrated in Figure 3-10.

Note: Due to the timing relationship between the internal and external clocks, the EMAC#CLIENTTXACK signal should be registered at the output of the EMAC at speed lower than 1 Gb/s.

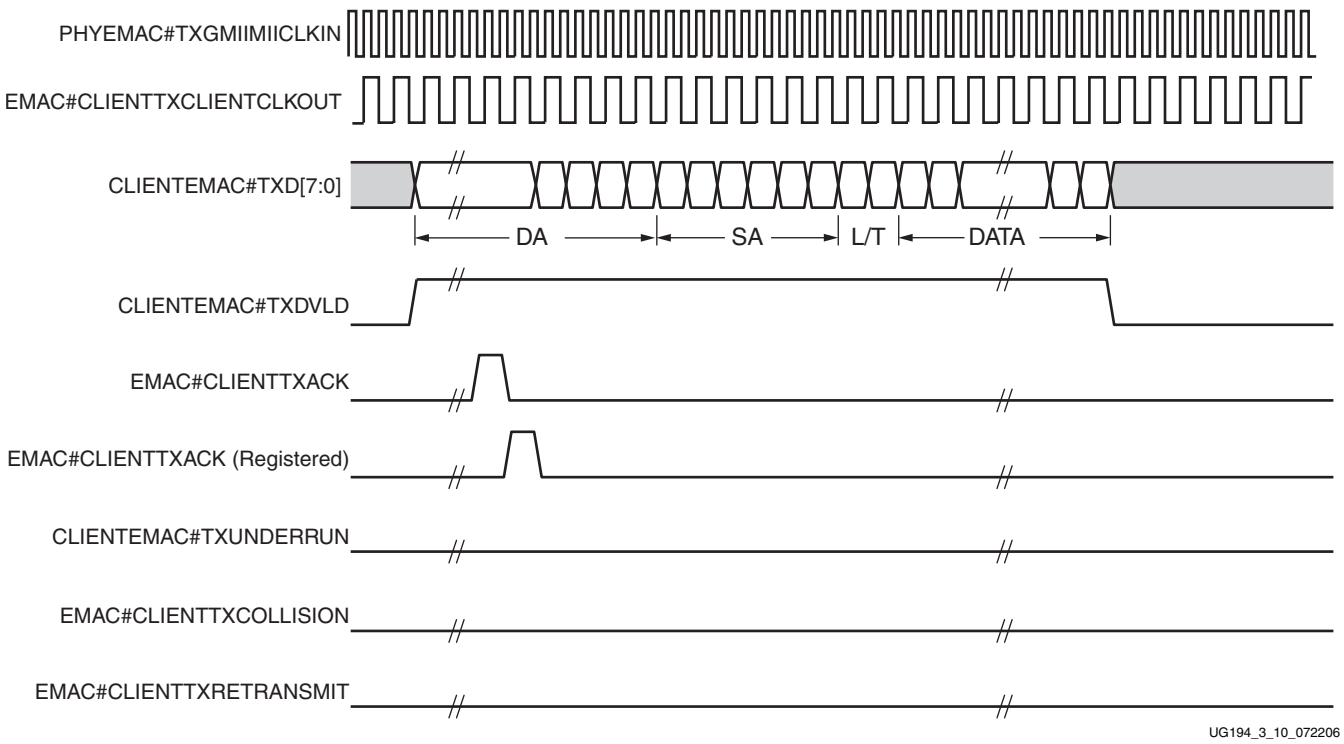


Figure 3-10: **Normal Frame Transmission at 10/100 Mb/s with EMAC#_USECLKEN Set to TRUE**

Transmit (TX) Client: 16-Bit Interface

This optional configuration can only be used when the Ethernet MAC physical interface is configured in 1000BASE-X PCS/PMA mode.

When a 16-bit client interface is selected, the input clock for the Ethernet MAC TX client logic (CLIENTEMAC#TXCLIENTCLKIN) should be twice the frequency of the clock used for the TX client logic in the FPGA logic. Figure 6-22, page 170 illustrates how this clocking scheme is achieved for this mode of operation.

This mode can operate with the Ethernet MAC TX client logic clocked at 125 MHz and the logic in the FPGA logic clocked at 62.5 MHz. However, this mode allows the Ethernet MAC TX client logic to operate at up to 250 MHz while the logic in the FPGA logic is clocked at 125 MHz. By using the faster clock speeds, a line rate of up to 2.5 Gb/s can be achieved. This is 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.

"Ethernet MAC Attributes," page 42 describes the EMAC#_TX16BITCLIENT_ENABLE attribute, which is set to TRUE for this configuration.

Figure 3-11 shows the timing of a normal outbound frame transfer for the case with an even number of bytes in the frame. The CLIENTEMAC#TXCLIENTCLKIN clock is used as an input to the Ethernet MAC and the PHYEMAC#MIITXCLK clock (CLIENTEMAC#TXCLIENTCLKIN/2) is used to clock the TX client logic in the FPGA logic.

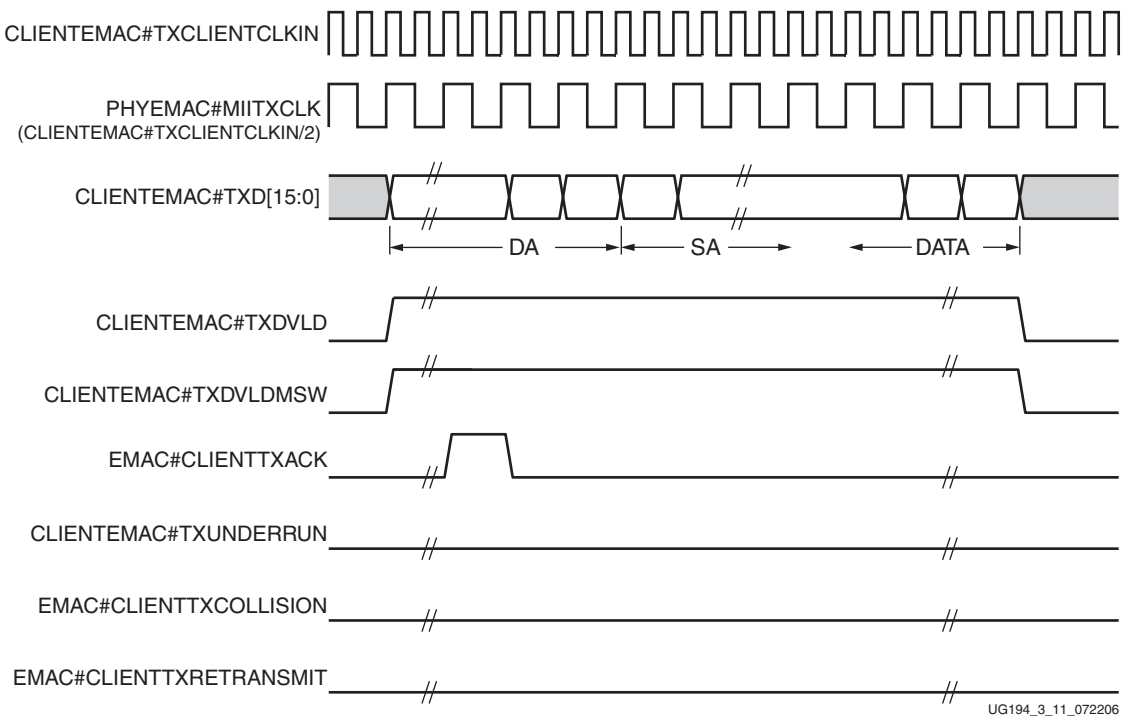


Figure 3-11: **16-Bit Transmit (Even Byte Case)**

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

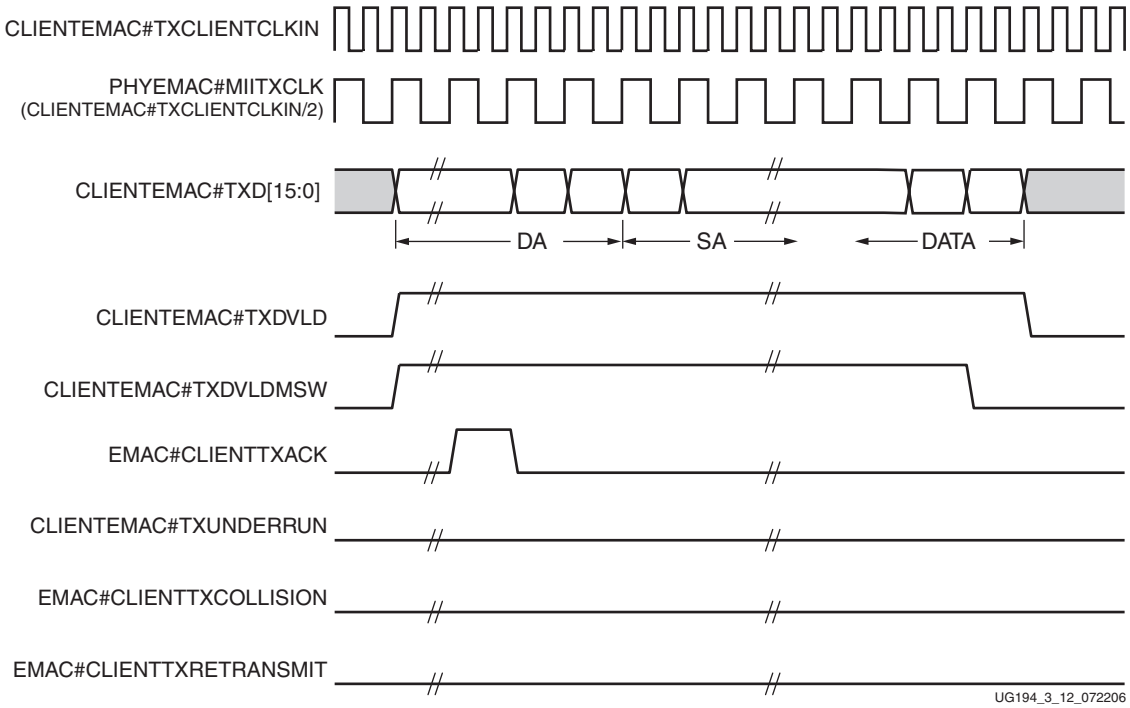


Figure 3-12: **16-Bit Transmit (Odd Byte Case)**

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 CLIENTEMAC#TXDVLD and CLIENTEMAC#TXDVLDMSW must be deasserted for one PHYEMAC#MIITXCLK (half the clock frequency of CLIENTEMAC#TXCLIENTCLKIN) clock cycle after the first frame. In the following PHYEMAC#MIITXCLK clock cycle, both CLIENTEMAC#TXDLVD and CLIENTEMAC#TXDVLDMSW are asserted 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.

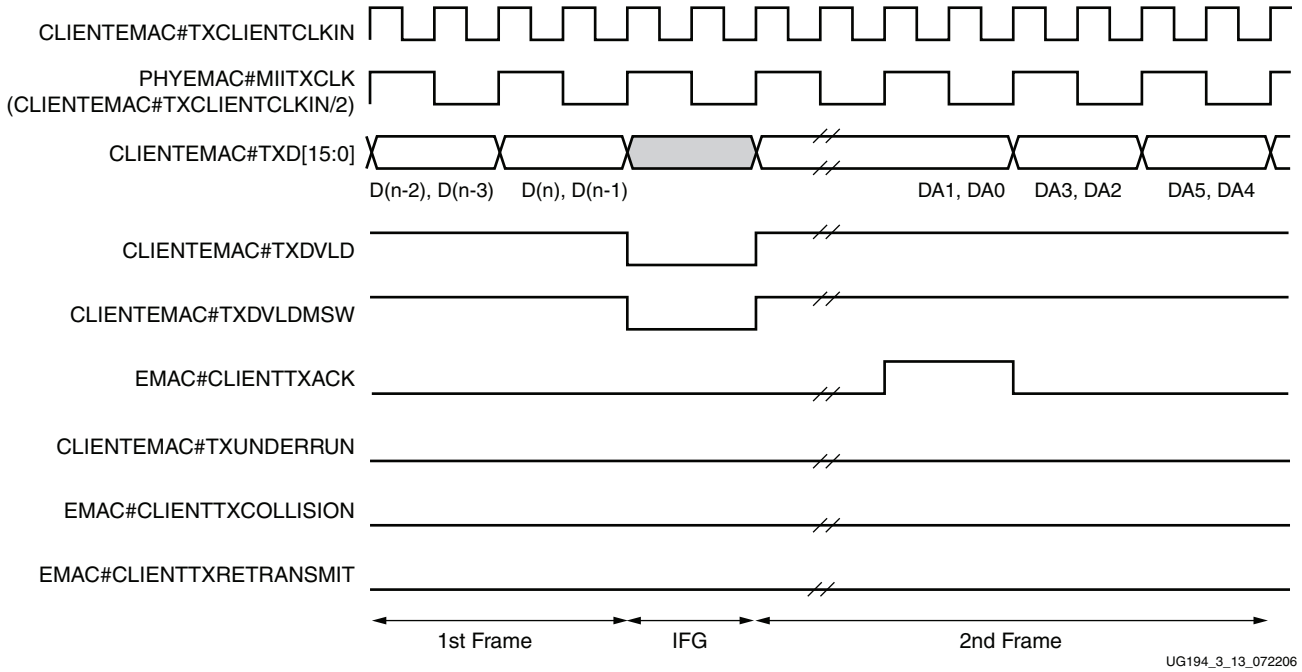


Figure 3-13: **16-Bit Transmit Back-to-Back Transfer (Even Byte Case)**

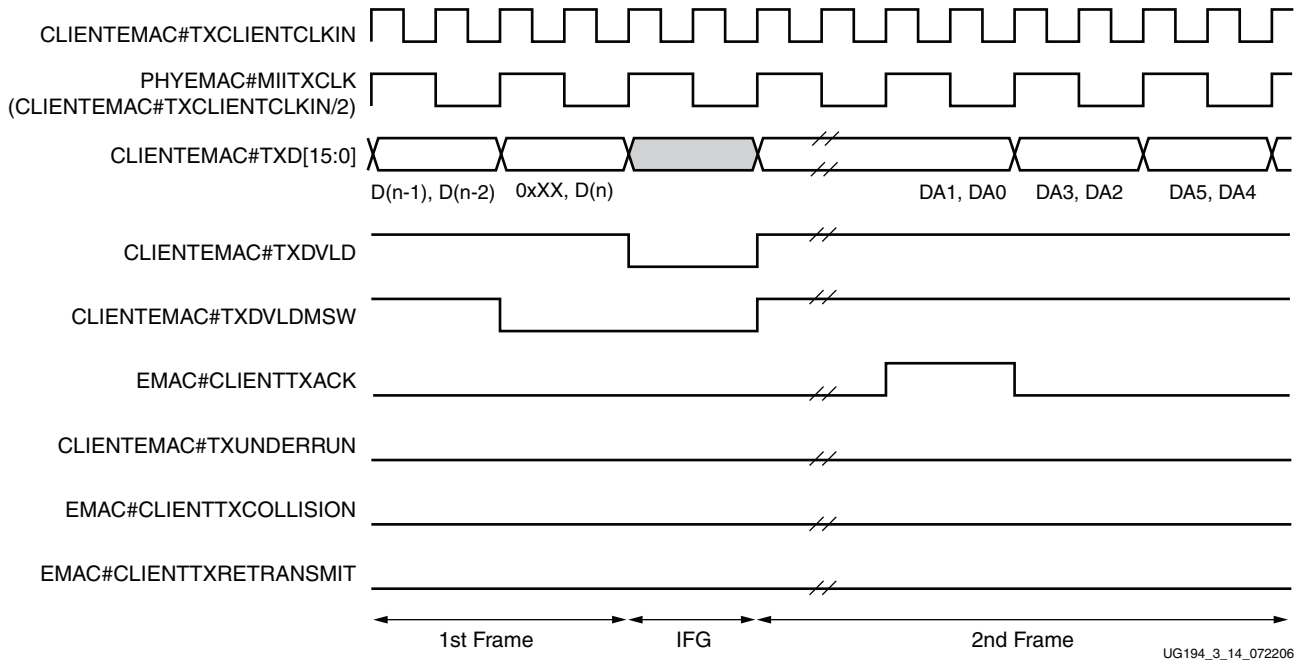


Figure 3-14: **16-Bit Transmit Back-to-Back Transfer (Odd Byte Case)**

Receive (RX) Client: 8-Bit Interface (without Clock Enables)

This section covers the client receive interface when it is configured to be 8 bits wide and when the default clocking scheme is used. "Ethernet MAC Attributes," page 42 describes the EMAC#_USECLKEN attribute, which is set to FALSE for this configuration. In this configuration, `EMAC#CLIENTRXD[15:8]` and `EMAC#CLIENTRXDVLDMSW` are left unconnected because the upper eight bits of the data bus are not required.

See "Ethernet MAC Clocks," page 205 for some general information on this standard clocking scheme. More details are available for specific physical interface configurations in the following sections:

- MII
  - "MII Standard Clock Management"
- GMII
  - "GMII Clock Management for 1 Gb/s Only"
  - "GMII Standard Clock Management for Tri-Speed Operation"
  - "GMII Clock Management for Tri-Speed Operation Using Byte PHY"
- RGMII
  - "RGMII Clock Management for 1 Gb/s Only"
  - "RGMII Standard Clock Management for Tri-Speed Operation"
- SGMII
  - "SGMII Clock Management (LXT and SXT Devices)"
  - "SGMII Clock Management (TXT and FXT Devices)"
- 1000BASE-X
  - "1000BASE-X PCS/PMA Clock Management (LXT and SXT Devices)"
  - "1000BASE-X PCS/PMA Clock Management (TXT and FXT Devices)"

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.

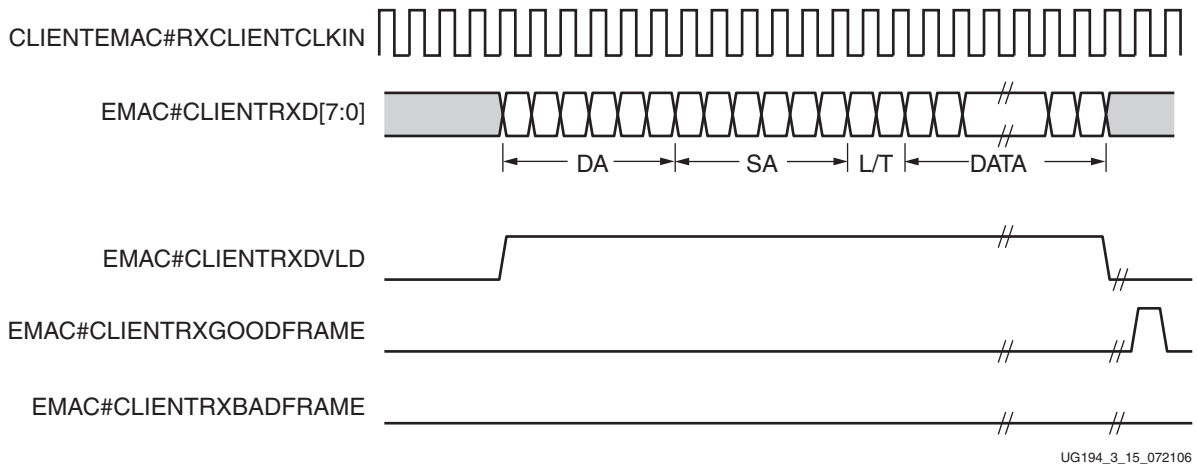


Figure 3-15: **Normal Frame Reception**

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 52.

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 66.

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

Signal Timing of Good/Bad Frame Pulses

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. These frame checks occur only after receipt of any padding (if present), followed by the FCS and any carrier extension (if present). Therefore, either EMAC#CLIENTRXGOODFRAME or EMAC#CLIENTRXBADFRAME is asserted following frame reception at the beginning of the IFG.

Frame Reception with Errors

An unsuccessful frame reception (for example, a fragment frame or a frame with an incorrect FCS) is shown in Figure 3-16. In this case, the EMAC#CLIENTRXBADFRAME signal is asserted to the client after the end of the frame. The client is responsible for dropping the data already transferred for this frame.

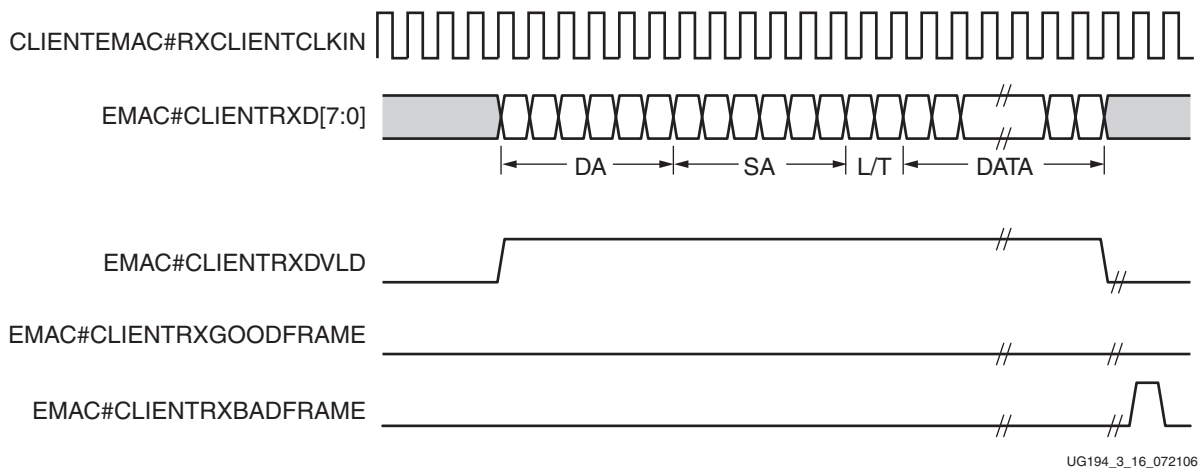


Figure 3-16: **Frame Reception with Error**

The following conditions cause the assertion of BAD_FRAME:

Standard Conditions

- FCS errors occur.
- Packets are shorter than 64 bytes (undersize or fragment frames).
- Jumbo frames are received when jumbo frames are not enabled.
- The length/type field is length, but the real length of the received frame does not match the value in the length/type field (when length/type checking is enabled).
- The length/type field is length, in which the length value is less than 46. In this situation, the frame should be padded to minimum length. If it is not padded to exactly minimum frame length, the frame is marked as bad (when length/type checking is enabled).
- Any control frame that is received is not exactly the minimum frame length.
- PHYEMAC#RXER is asserted at any point during frame reception.
- An error code is received in the 1-Gigabit frame extension field.
- A valid pause frame, addressed to the MAC, is received when flow control is enabled. Please see "Flow Control Block" for more information.

1000BASE-X/SGMII Specific Conditions

When in 1000BASE-X mode or SGMII mode, the following 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 occur during the packet.
- Unexpected K characters or sequences appearing in the wrong order/byte position.

Client-Supplied FCS Passing

Figure 3-17 shows how the Ethernet MAC is configured to pass the FCS field to the client (see "Configuration Registers," page 88). 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.

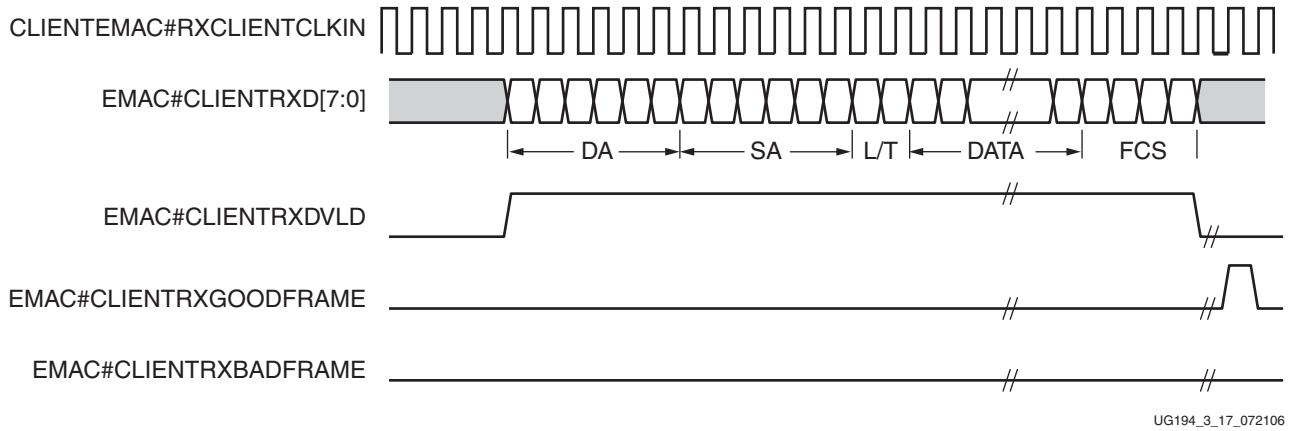


Figure 3-17: **Frame Reception with In-Band FCS Field**

VLAN Tagged Frames

The reception of a VLAN tagged frame (if enabled) is shown in Figure 3-18. 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. The length/type field after the tag control information bytes is not checked for errors. More information on the interpretation of these bytes is described in the IEEE Std 802.3-2002 standard.

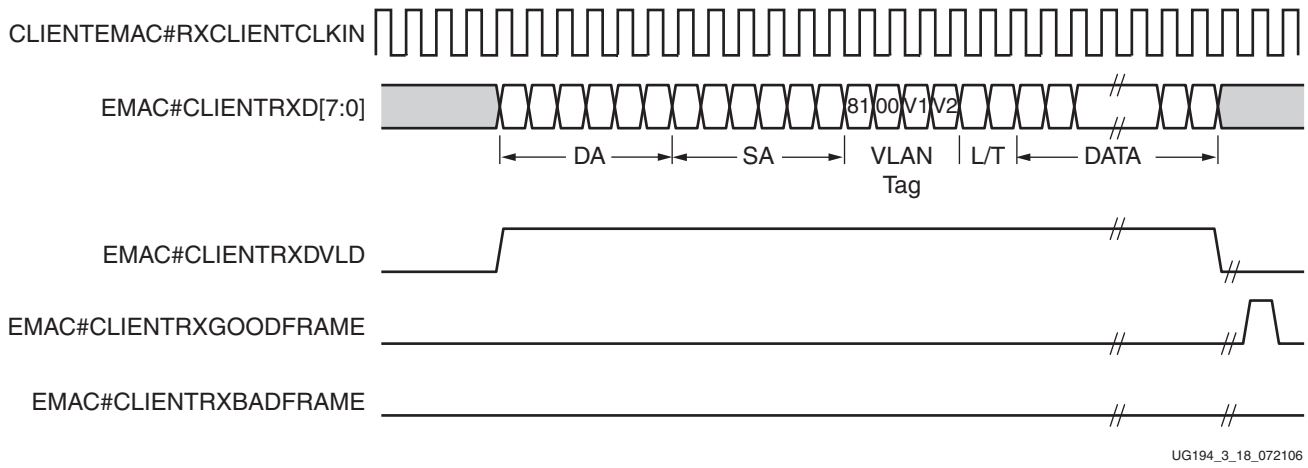


Figure 3-18: **Reception of a VLAN Tagged Frame**

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 88.

Length/Type Field Error Checks

Length/Type Field Error checking is specified in IEEE Std 802.3. 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 89), the following checks are made on all frames received. (If either of these checks fails, EMAC#CLIENTRXBADFRAME is asserted):

- A value greater than or equal to decimal 46 but less than decimal 1500 (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. The resultant frame is now the minimum frame size of 64 bytes total in length.

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.

When a frame is a VLAN frame, only the initial length/type field containing the VLAN type 0x8100 is checked and interpreted as a type. The subsequent length/type field in the VLAN frame is not checked.

Disabled

When the length/type error checking is disabled (see "Receiver Configuration Register (Word 1)," page 89), the length/type error checks previously described are not performed. A frame containing only these errors causes EMAC#CLIENTRXGOODFRAME to be asserted.

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.

It should be noted that an illegal length frame, e.g., a frame of less than 64 bytes in total length, is still marked as bad. In addition, the disabling of Length/Type checks has no effect on control frames. If a control frame is received that is not 64 bytes in total length, it is still marked as a bad frame.

Receive (RX) Client: 8-Bit Interface (with Clock Enables)

This section covers the client receive interface when it is configured to be 8 bits wide and when the optional clock enable clocking scheme is used. This optional configuration can be used when the Ethernet MAC is configured in MII/GMII or RGMII mode. "Ethernet MAC

Attributes," page 42 describes the EMAC#_USECLKEN attribute, which is set to TRUE for this configuration.

In this mode, the client interface is synchronous to the physical interface clock. This configuration reduces the number of BUFGs that are used to implement the Ethernet MAC design. See "Ethernet MAC Clocks," page 205 for some general information on this clocking scheme. More details are available for specific physical interface configurations in the following sections:

- "MII Clock Management using Clock Enables"
- "GMII Clock Management for Tri-Speed Operation Using Clock Enables"
- "RGMII Clock Management for Tri-Speed Operation Using Clock Enables"

For the receive interface, the PHYEMAC#RXCLK signal is used as the clock, and the EMAC#CLIENTRXCLIENTCLKOUT signal is used as the clock enable.

Figure 3-19 shows the timing of a normal inbound frame transfer at 1 Gb/s. Data is transferred to the client on the rising edge of PHYEMAC#RXCLK. The output clock enable, EMAC#CLIENTRXCLIENTCLKOUT, is held High.

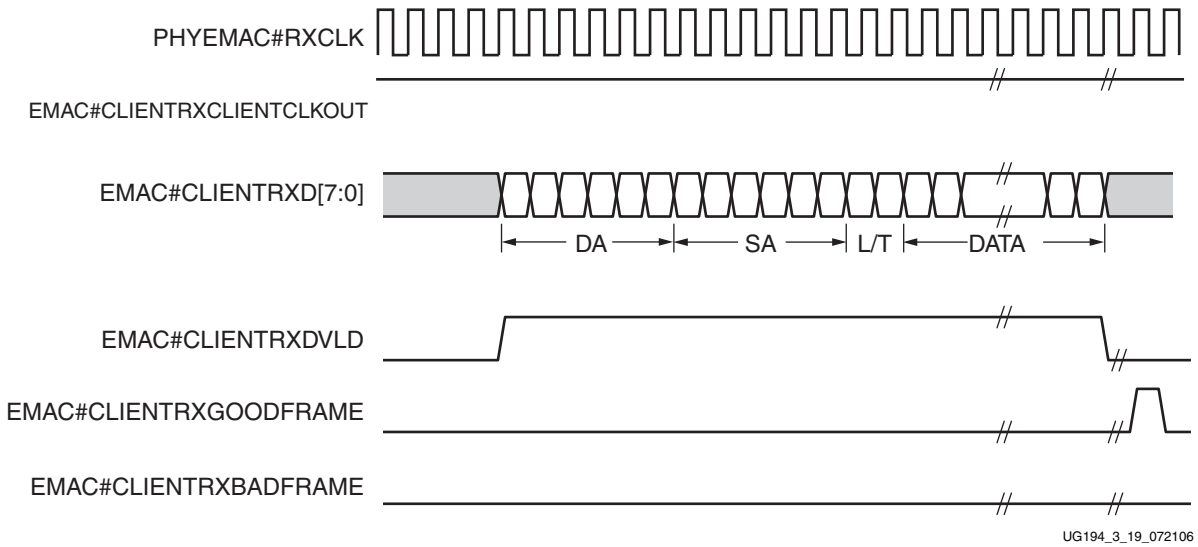


Figure 3-19: **Normal Frame Reception at 1 Gb/s with EMAC#_USECLKEN Set to TRUE**

Figure 3-20 shows the reception of a frame at speeds below 1 Gb/s. The physical interface clock runs at twice the speed of the client interface clock and so the output clock enable, EMAC#CLIENTRXCLIENTCLKOUT, is only asserted on every second cycle of

PHYEMAC#RXCLK. Data is written into the client on the rising edge of PHYEMAC#RXCLK when EMAC#CLIENTRXCLIENTCLKOUT is High.

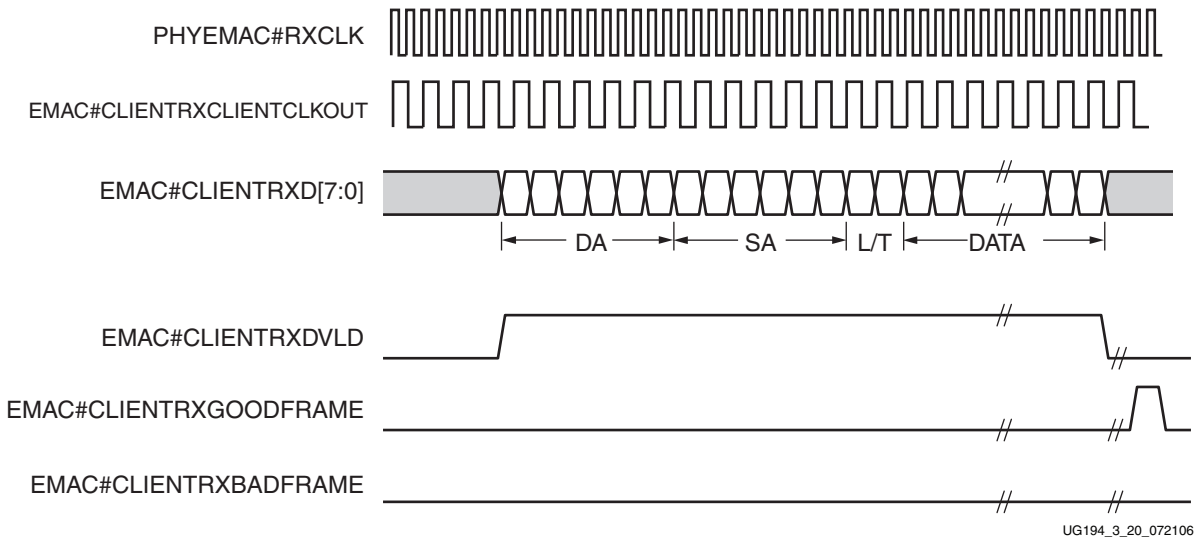


Figure 3-20: **Normal Frame Reception at 10/100 Mb/s with EMAC#_USECLKEN Set to TRUE**

Receive (RX) Client: 16-Bit Interface

This optional configuration can only be used when the Ethernet MAC physical interface is configured in 1000BASE-X PCS/PMA mode.

When a 16-bit client interface is selected, the input clock for the Ethernet MAC RX client logic (CLIENTEMAC#RXCLIENTCLKIN) should be twice the frequency of the clock used for the RX client logic in the FPGA logic. Figure 6-22, page 170 illustrates how this clocking scheme is achieved for this mode of operation.

This mode can operate with the Ethernet MAC RX client logic clocked at 125 MHz and the logic in the FPGA logic clocked at 62.5 MHz. However, this mode allows the Ethernet MAC RX client logic to operate at up to 250 MHz while the logic in the FPGA logic is clocked at 125 MHz. By using the faster clock speeds, a line rate of up to 2.5 Gb/s can be achieved. This is 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.

"Ethernet MAC Attributes," page 42 describes the EMAC#_RX16BITCLIENT_ENABLE attribute, which is set to TRUE for this configuration.

Figure 3-21 shows the timing of a normal inbound frame transfer for the case with an even number of bytes in the frame. The CLIENTEMAC#RXCLIENTCLKIN clock is used as an input to the Ethernet MAC and the PHYEMAC#RXCLK clock (CLIENTEMAC#RXCLIENTCLKIN/2) is used to clock the RX client logic in the FPGA logic.

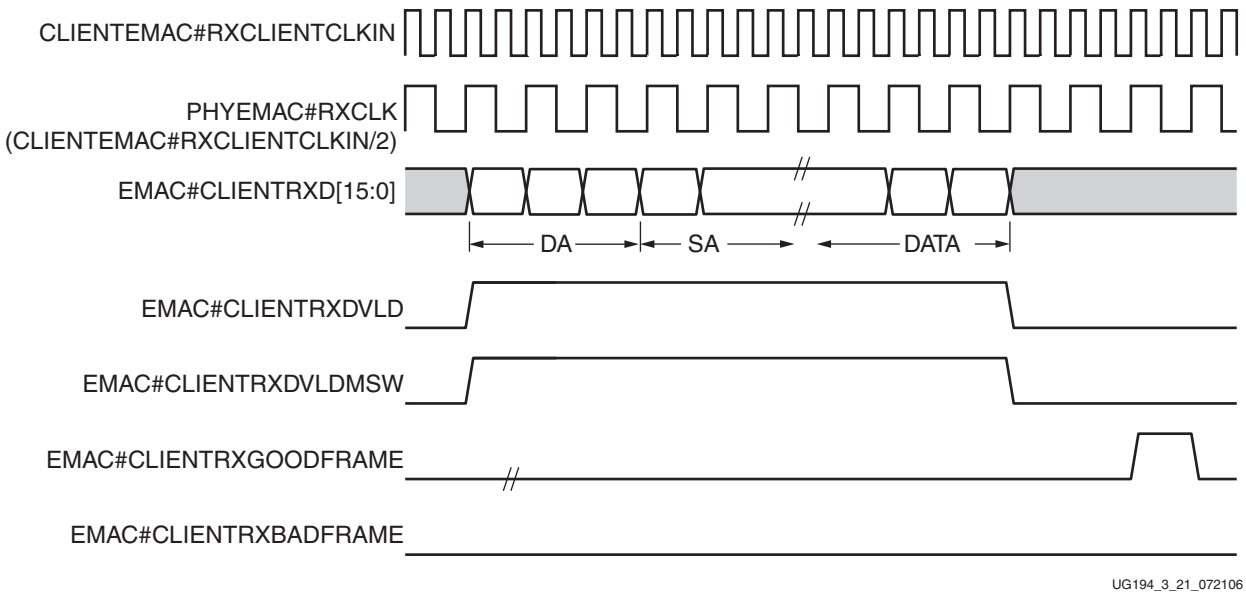


Figure 3-21: **16-Bit Receive (Even Byte Case)**

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

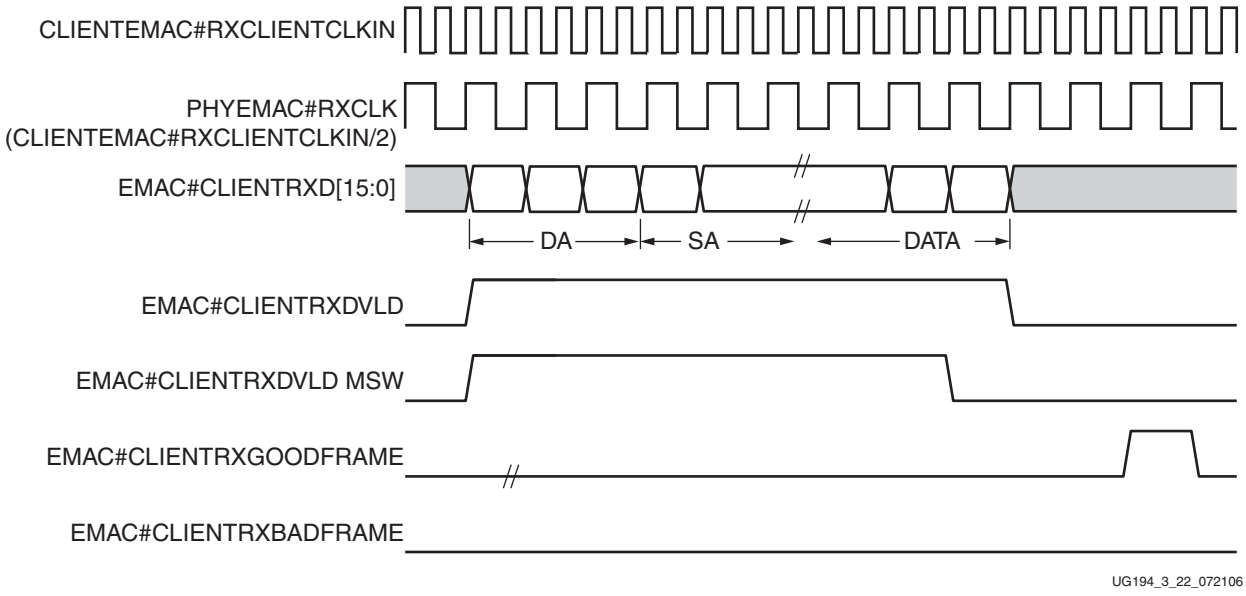


Figure 3-22: **16-Bit Receive (Odd Byte Case)**

As shown in Figure 3-21 and Figure 3-22, `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 3-21). In the odd byte case (Figure 3-22), `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:

- Matching of programmable unicast destination address
- Matching of four additional programmable general addresses
- 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. 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 94 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.

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, EMAC#CLIENTRXDVLD is deasserted. However, using the EMAC#CLIENTRXFRAMEDROP signal, the nature of an incoming RX frame can be distinguished for system monitoring. See Table 3-2.

Table 3-2: **EMAC#CLIENTRXDVLD and EMAC#CLIENTRXFRAMEDROP Values**

Result of an Incoming RX Frame	EMAC#CLIENTRXDVLD	EMAC#CLIENTRXFRAMEDROP
No frame received	0	0
Frame received and AF address match	1	0
Frame received and no AF address match	0	1
Frame received and no AF address match but AF disabled	1	1
Frame received and AF address match but AF disabled	1	0

Address Filter Attributes

The unicast address register, pause frame source address, and address filter promiscuous mode bit are set by attributes when the FPGA is configured. In this way, the address filter performs functions with the unicast address without using the host interface. The four general address register values are not included as attributes.

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 attribute settings of the registers can be overridden by the software through the host interface. Also, the four general 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 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 3-23 shows the timing diagram when a frame matches a valid location in the AF (8-bit mode).

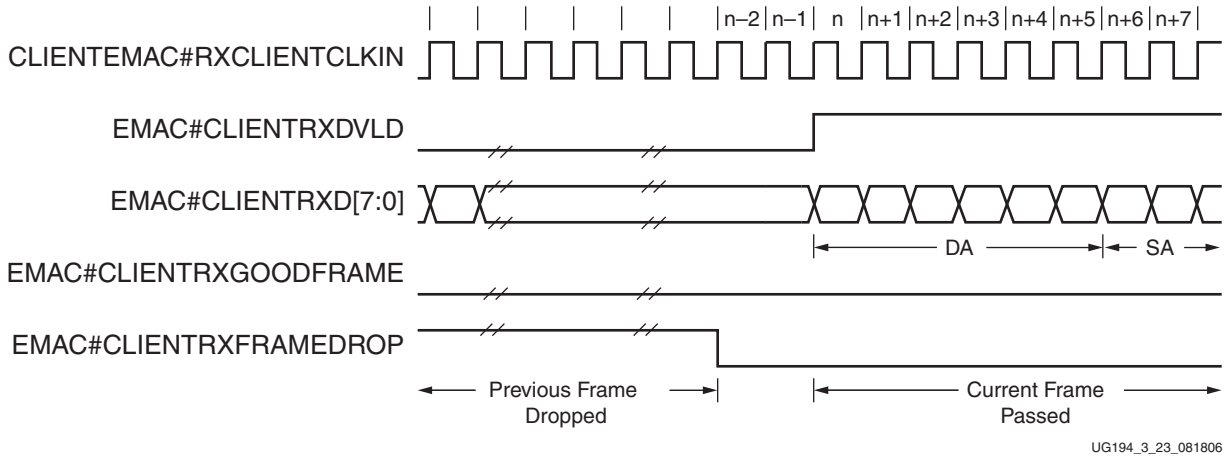


Figure 3-23: **Frame Matching Timing Diagram (8-Bit Mode)**

Figure 3-24 shows the timing diagram when a frame matches a valid location in the AF (16-bit mode).

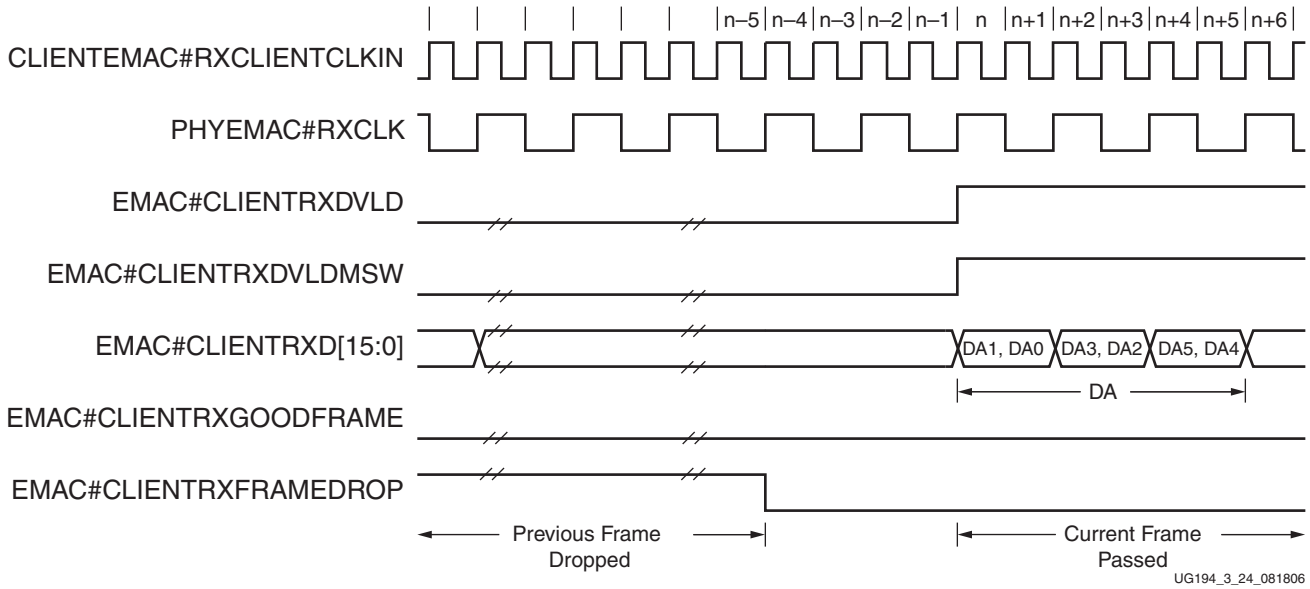


Figure 3-24: **Frame Matching Timing Diagram (16-Bit Mode)**

Figure 3-25 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.

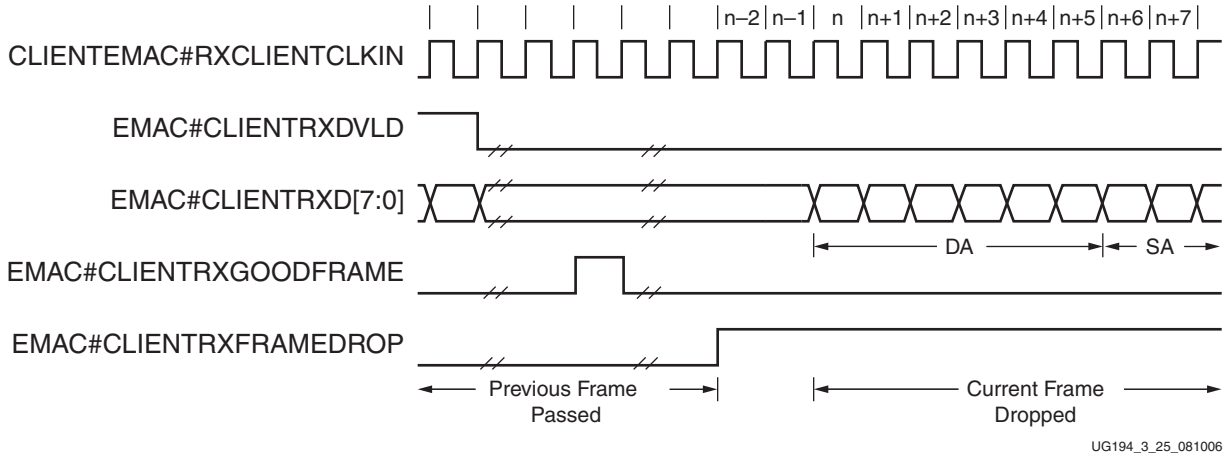


Figure 3-25: **Frame Matching Failed Timing Diagram (8-Bit Mode)**

Figure 3-26 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.

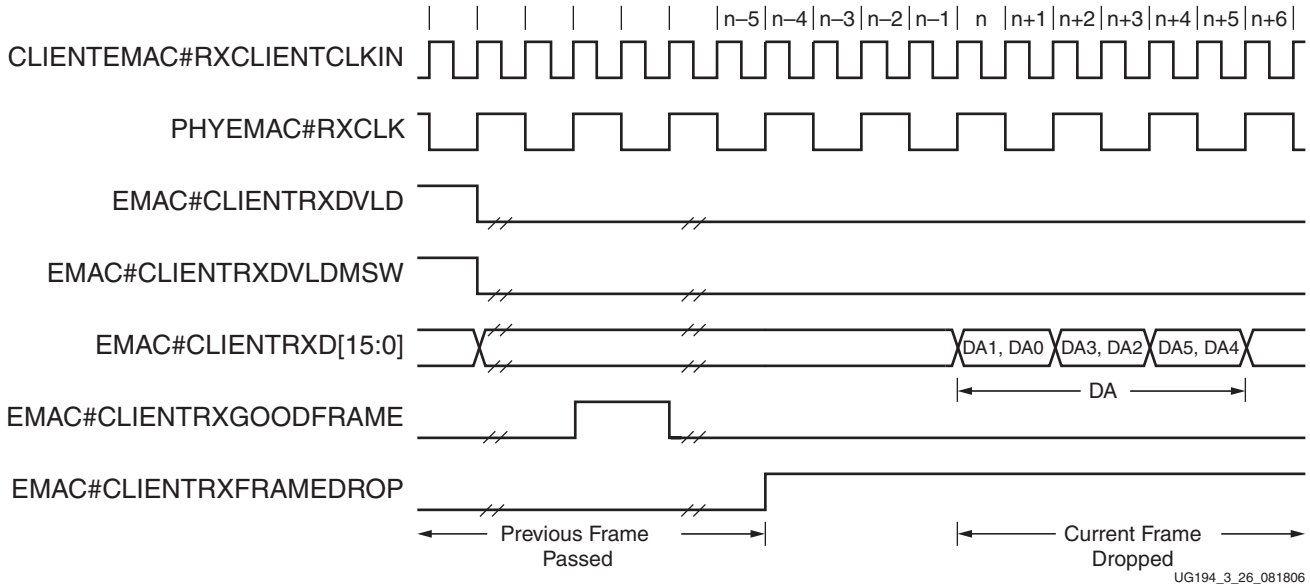


Figure 3-26: **Frame Matching Failed Timing Diagram (16-Bit Mode)**

Flow Control Block

The flow control block is designed to clause 31 of the IEEE Std 802.3-2005 standard. The Ethernet MAC can be configured to send PAUSE frames and to act upon the PAUSE frame's reception during full-duplex operation. These two behaviors can be configured asymmetrically (see "Configuration Registers," page 88).

Requirement for Flow Control

Figure 3-27 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.

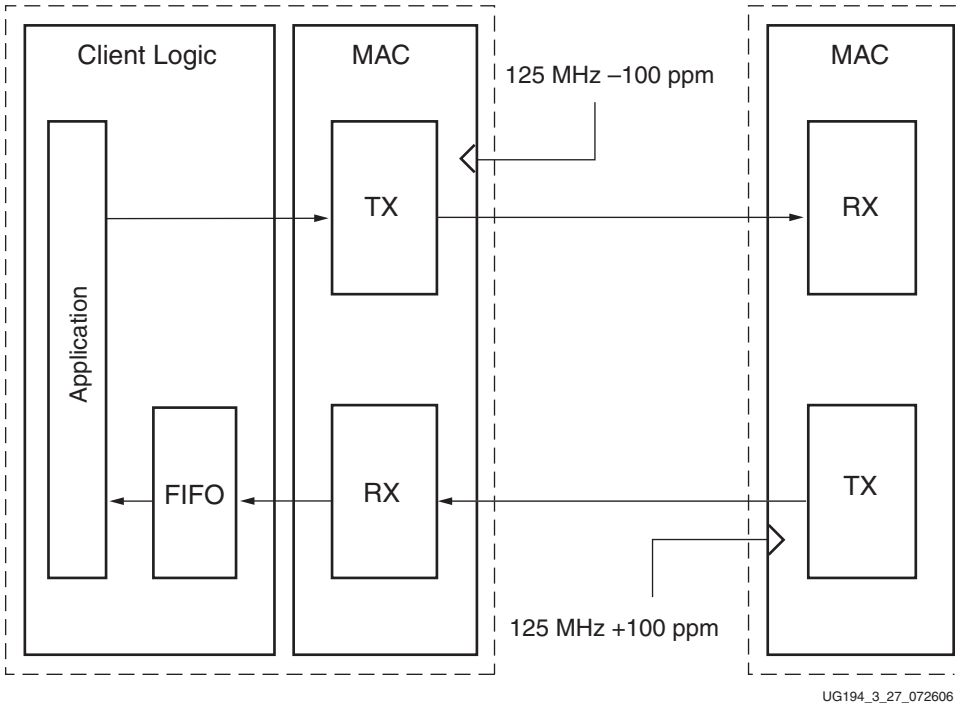


Figure 3-27: **Requirement for Flow Control**

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 3-27 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 3-27 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 overflows. This provides time to empty the FIFO to a safe level before normal operation resumes. It also safeguards 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. When the optional clock enables are not used, these signals are synchronous to CLIENTEMAC#TXCLIENTCLKIN. The timing is shown in Figure 3-28.

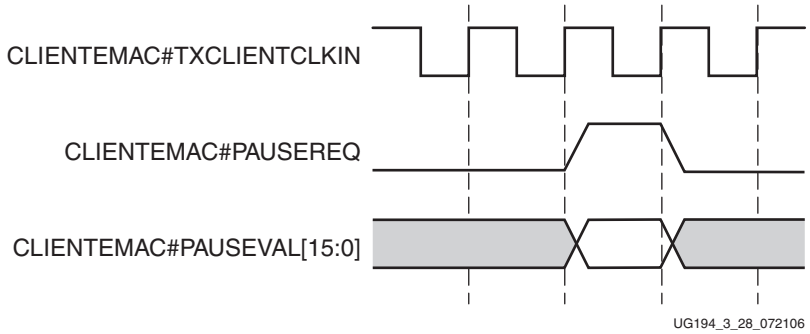


Figure 3-28: **Pause Request Timing**

When the optional clock enable clocking scheme is used, the signals are synchronous to PHYEMAC#TXGMIIMIICLKIN. These signals are written into the Ethernet MAC when the enable signal, EMAC#CLIENTTXCLIENTCLKOUT, is High.

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 3-29.

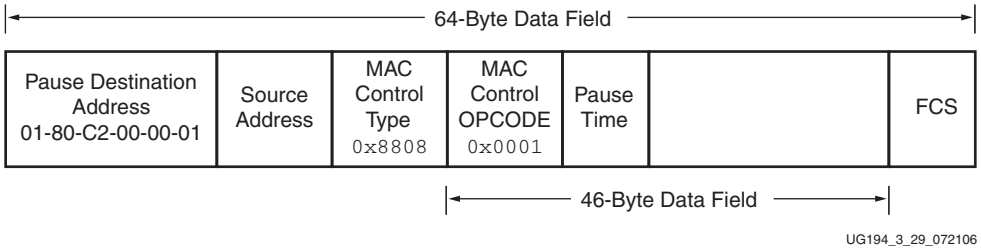


Figure 3-29: **Pause Frame Example**

The pause source address can be configured using the "Configuration Registers," page 88. The pause time transmitted in the PAUSE frame is the current sampled CLIENTEMAC#PAUSEVAL[15:0] value.

Because only a single pause control frame request is stored by the transmitter, if CLIENTEMAC#PAUSEREQ is asserted multiple times prior to the transmission of a control pause frame (i.e., while the previous frame transmission is still in progress), only a single pause frame is transmitted.

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 (01-80-C2-00-00-01) and the configured address for the Ethernet MAC (see "Address Filter Registers," page 94).
- 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 for pause frames.

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 by delaying the assertion of EMAC#CLIENTTXACK at the transmitter client interface until the requested pause duration has expired. Because the received pause frame has been acted upon, it is passed to the client with EMAC#CLIENTRXBADFRAME asserted, indicating to the client that the frame should be dropped.

If the second match is TRUE and the frame is not exactly 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.

If any of the previous matches are FALSE or if the Ethernet MAC flow control logic for the receiver is disabled, the frame is ignored by the flow control logic and is passed on to the client. It is the responsibility of the receiver client logic to drop any frames where the LT field matches the Ethernet MAC control type code. The receiver statistics vector (bit 19) indicates if the previous frame is a control frame. See "Receiver Statistics Vector" for more information.

Flow Control Implementation Example

In the system illustrated in Figure 3-27, 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 1/2 (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 (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 3-30 illustrates the FIFO occupancy over time.

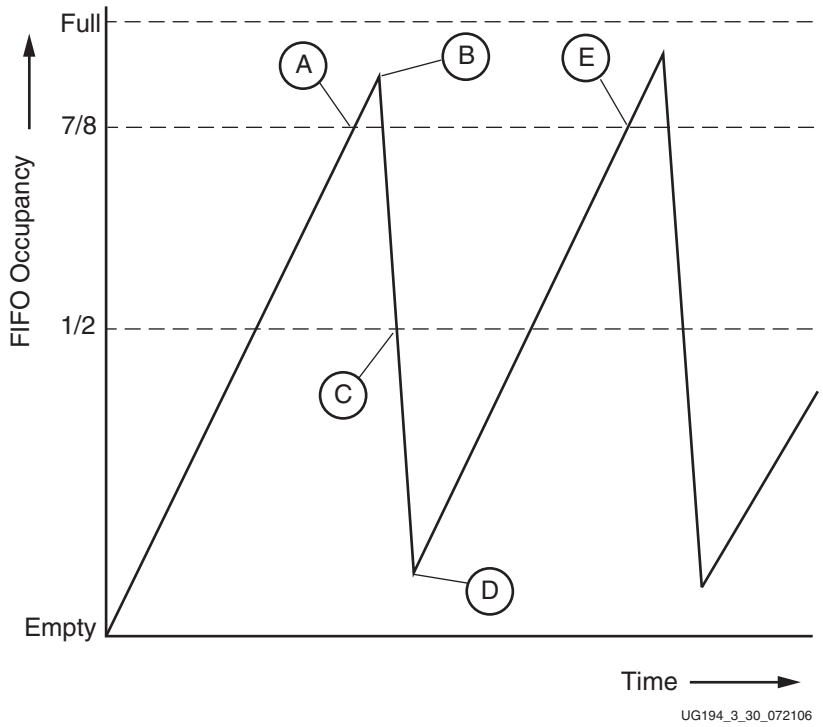


Figure 3-30: **Flow Control Implementation Triggered from FIFO Occupancy**

- The 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 full, triggering the maximum duration pause control frame request.
- Upon receiving the pause control frame, the right Ethernet MAC ceases transmission at point B. The system designer should ensure sufficient space in the FIFO above the threshold (point A) to allow the transmitter to stop.
- After the right Ethernet MAC ceases transmission, the occupancy of the FIFO attached to the left Ethernet MAC empties. The occupancy falls to the second threshold of 1/2 occupancy at point C, triggering the zero duration pause control frame request (the pause cancel command).
- Upon receiving this second pause control frame, the right hand Ethernet MAC resumes transmission. To create an efficient system with the maximum possible line rate, the system designer should ensure sufficient space in the FIFO, below the threshold (point C), to prevent the FIFO emptying before transmission starts at point D.
- Normal operation resumes, and the FIFO occupancy again gradually increases over time. At point E, the flow control cycle repeats.

Statistics Vectors

Transmitter Statistics Vector

TX_STATISTICS_VECTOR contains the statistics for the frame transmitted. The TX_STATISTICS_VECTOR is a 32-bit vector and an internal signal. This vector is muxed out to a one-bit signal, `EMAC#CLIENTTXSTATS`, as shown in Figure 3-31 , following frame transmission. For standard clock management, the vector is driven synchronously by the transmitter clock, `CLIENTEMAC#TXCLIENTCLKIN`. For alternative clock management, including the use of clock enables, the vector is driven synchronously to other TX client logic signals. See Chapter 6, "Physical Interface," for relevant clocking networks.

The bit-field definition for the transmitter statistics vector is defined in Table 3-3, page 80.

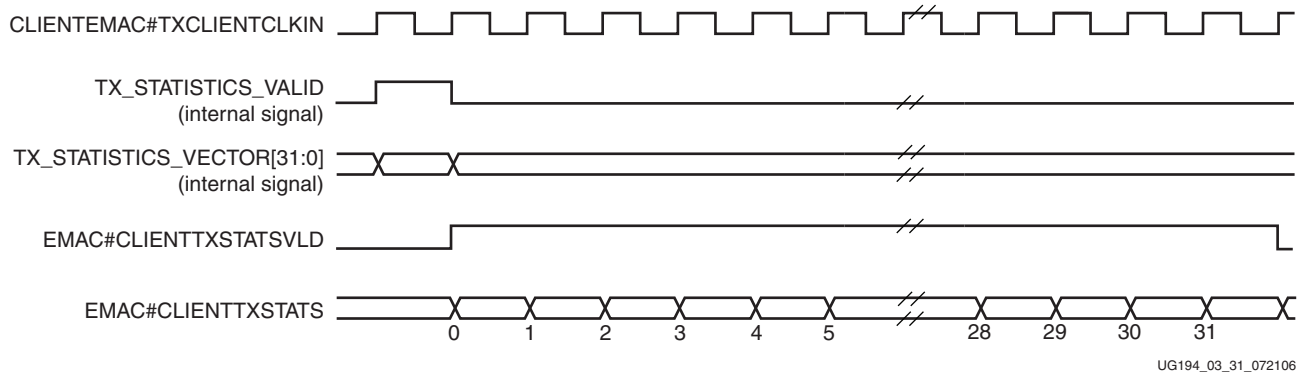


Figure 3-31: **Transmitter Statistics Mux Timing**

The block diagram for the transmitter statistics MUX in the Ethernet MAC is shown in Figure 3-32.

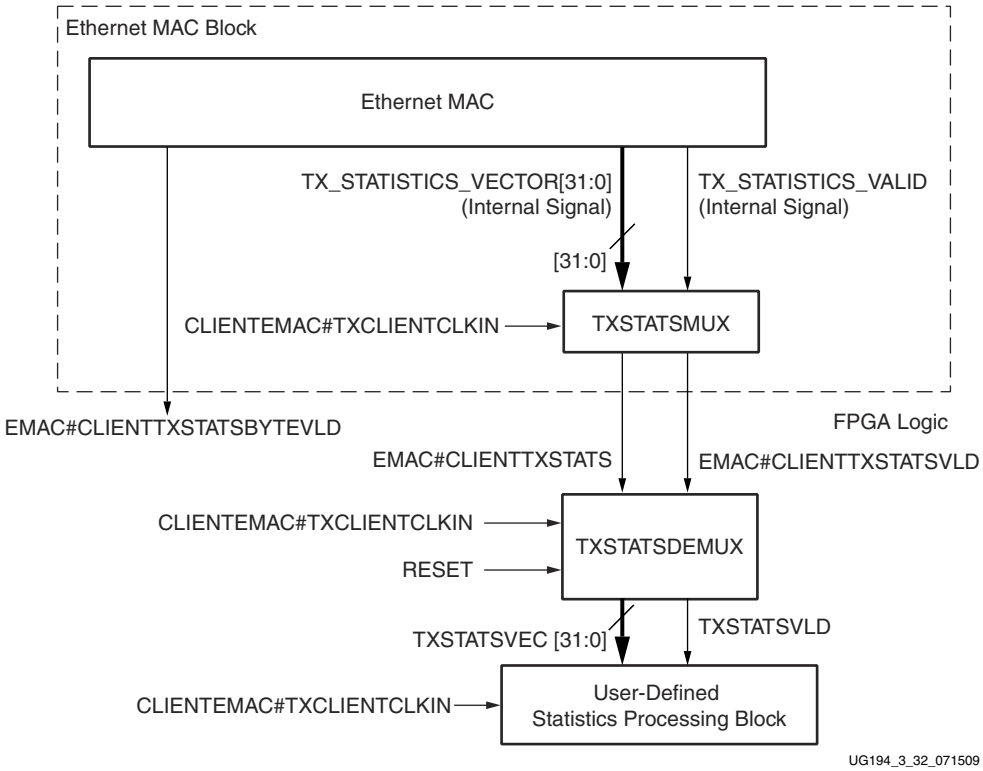


Figure 3-32: **Transmitter Statistics MUX Block Diagram**

All bit fields in EMAC#CLIENTTXSTATS are only valid when EMAC#CLIENTTXSTATSVLD is asserted as illustrated in Figure 3-33. 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 reset to a logic 0.

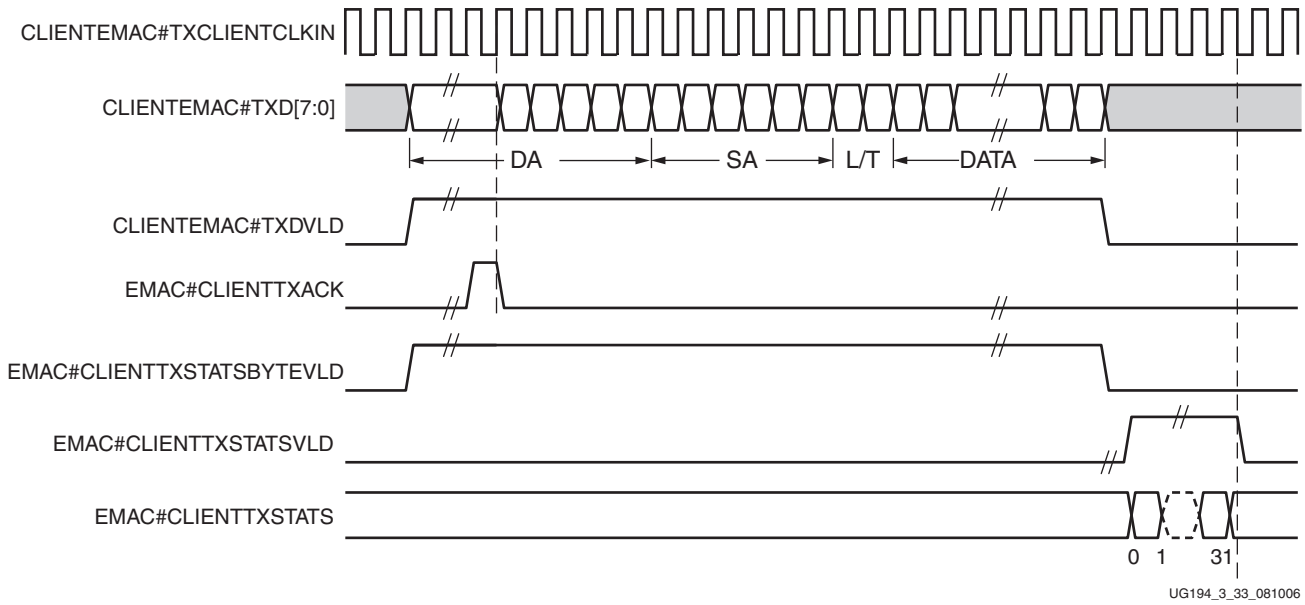


Figure 3-33: **Transmitter Statistics Vector Timing**

Table 3-3: **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] (1)	TX_ATTEMPTS[3:0]	The number of attempts made to transmit the previous frame. A 4-bit number where 0 is one attempt; 1 is two attempts...up to 15 to describe 16 attempts.
24 (1)	Reserved	Returns a logic 0.
23 (1)	EXCESSIVE_COLLISION	Asserted if a collision is detected on each of the last 16 attempts to transmit the previous frame.
22 (1)	LATE_COLLISION	Asserted if a late collision occurred during frame transmission.
21 (1)	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.
20 (1)	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.

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

TX_STATISTICS_VECTOR	Name	Description
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. It is a 27-bit vector and an internal signal. This vector is muxed out to a 7-bit signal `EMAC#CLIENTRXSTATS[6:0]`, as shown in Figure 3-34 , following frame reception. For standard clock management, the vector is driven synchronously by the receiver clock, `CLIENTEMAC#RXCLIENTCLKIN`. For alternative clock management, including the use of clock enables, the vector is driven synchronously to other RX client logic signals. See Chapter 6, "Physical Interface," for relevant clocking networks.

The bit-field definition for the receiver statistics vector is defined in Table 3-4.

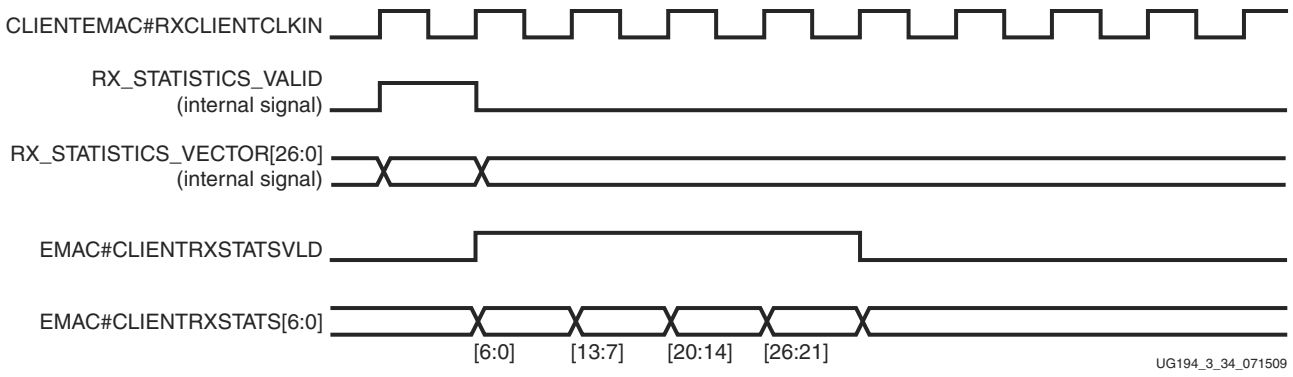


Figure 3-34: **Receiver Statistics MUX Timing**

The block diagram for the receiver statistics MUX in the Ethernet MAC is shown in Figure 3-35.

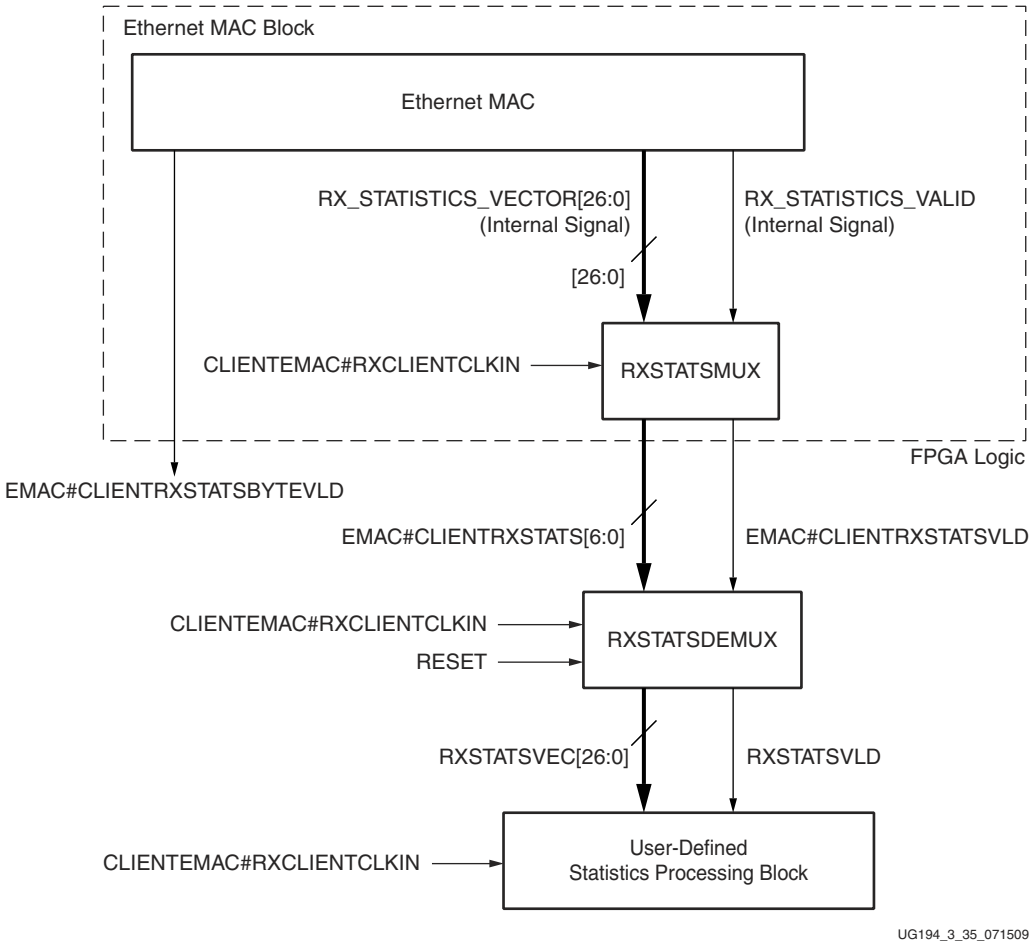


Figure 3-35: **Receiver Statistics MUX Block Diagram**

All bit-fields for EMAC#CLIENTRXSTATS[6:0] are valid only when EMAC#CLIENTRXSTATSVLD is asserted as illustrated in Figure 3-36. EMAC#CLIENTRXSTATSBYTEVLD is asserted if an Ethernet MAC frame byte (DA to FCS inclusive) is received. This is valid on every CLIENTEMAC#RXCLIENTCLKIN cycle.

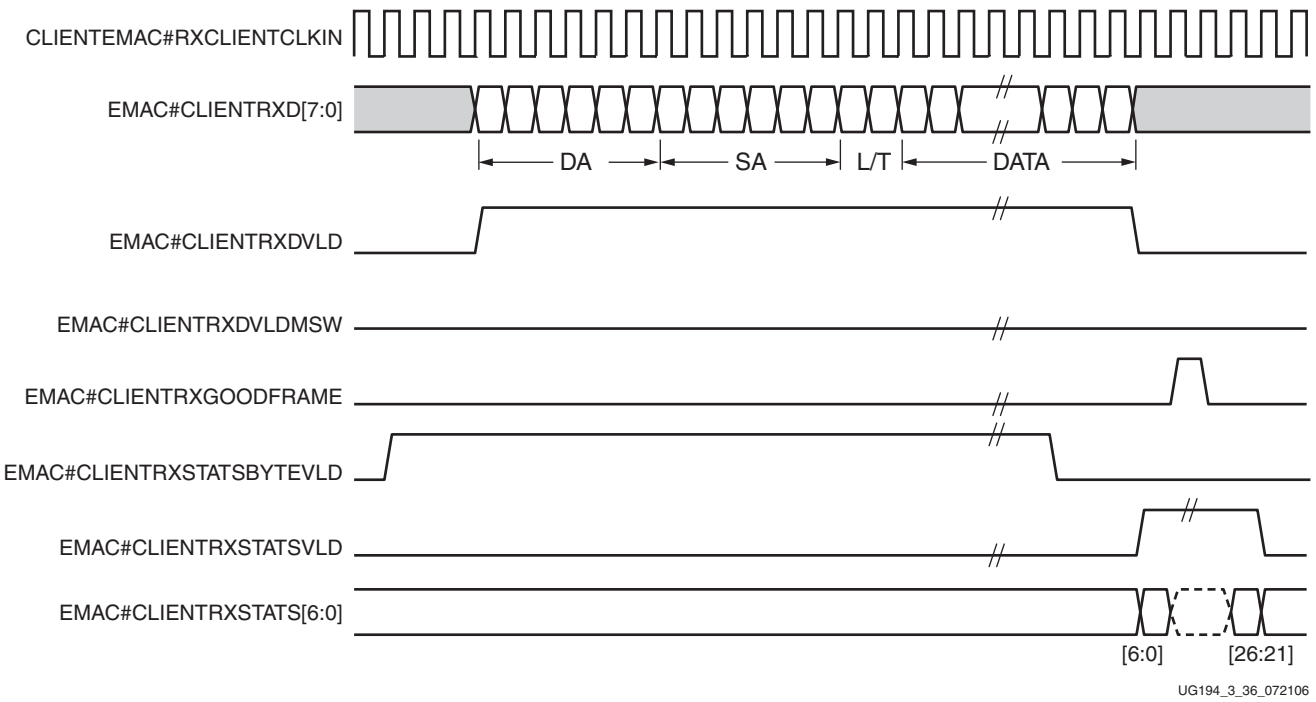


Figure 3-36: **Receiver Statistics Vector Timing**

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

RX_STATISTICS_VECTOR	Name	Description
26	ALIGNMENT_ERROR	Used in 10/100 MII mode. Asserted if the previous frame received has an incorrect FCS value and a misalignment occurs when the 4-bit MII data bus is converted to the 8-bit GMII data bus.
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 64 bytes (minimum frame size). 'This bit is not defined when LT field error-checks are disabled.
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 3-4: **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 66). 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 logic to collect the statistics. A parameterizable Ethernet statistics core is available from the LogiCORE™ libraries. For more information, see:

http://www.xilinx.com/products/ipcenter/ETHERNET_STATS.htm.

When the DCR is used, it can access the statistics counter registers in the FPGA logic through the DCR bridge in the host interface. Chapter 7, "Interfacing to a Statistics Block," describes how to integrate Ethernet Statistics LogiCORE core into a design, as well as how to access the statistics registers using either the host bus or the DCR bus.

XILINX®

Chapter 4

Host/DCR Bus Interfaces

This chapter contains the following sections:

Introduction to the Ethernet MAC Host Interface

Access to the Ethernet MAC in the Virtex®-5 FPGA through the host interface allows the user to:

- Read and write Ethernet MAC configuration registers
- Read and write Ethernet MAC address filter registers
- Read and write the PCS/PMA sublayer registers
- Drive the Management Data I/O (MDIO) interface to control an external device

The Ethernet MAC host interface can be accessed using either the host bus or the DCR bus. The Ethernet MAC should be configured to use the required bus by setting the DCREMACENABLE input bit accordingly, see "DCR Bus Signals," page 37.

Figure 4-1 shows the internal structure of the host interface. The two Ethernet MACs within the same block share a single host interface.

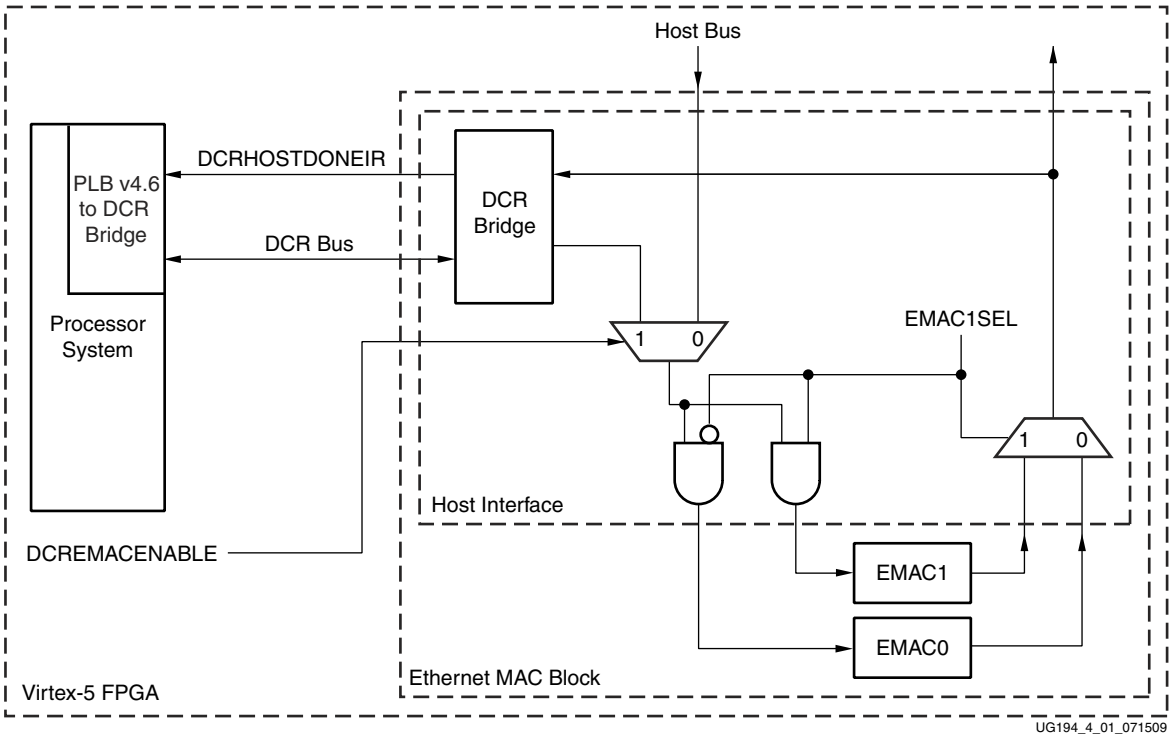


Figure 4-1: **Host Interface**

EMAC1SEL (internal signal) is driven by the HOSTEMAC1SEL input signal when using the host bus or decoded by the DCR bridge when using the DCR bus.

The XPS_LL_TEMAC soft core delivered through the XPS tool controls the Ethernet MAC via the processor local bus (PLB) v4.6 interface. A bridge is provided to convert between the PLB v4.6 and DCR bus standards. For more information on the PLB v4.6 interface, see DS531, *Processor Local Bus (PLB) v4.6 (v1.00a)*.

Figure 4-2 shows the block diagram of the host interface.

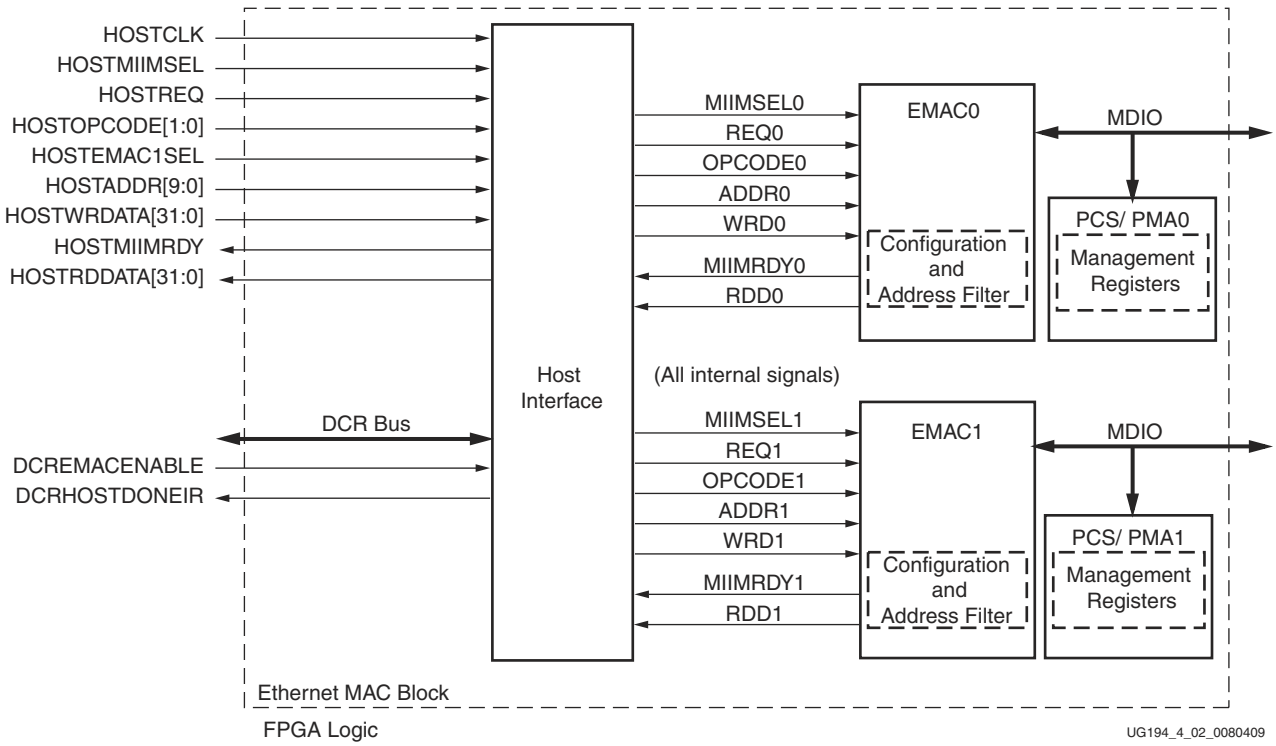


Figure 4-2: **Ethernet MAC Host Interface Block Diagram**

To access management registers in the PCS/PMA sublayer, the MDIO interface is used. The MDIO interface is described in detail in Chapter 5, "MDIO Interface," and access to the MDIO through the host interface is discussed later in this chapter.

Ethernet MAC Register Descriptions

Each Ethernet MAC has twelve configuration and address filter registers. These registers are accessed through the host interface and can be written to at any time. Both the receiver and transmitter logic only responds to configuration changes during Inter Frame gaps. The configurable resets are the only exception that take effect immediately. Writes made to address filter registers take effect immediately. To avoid corruption of incoming frames, writing to these registers should be made only while frame reception is not in progress.

Table 4-1: **Ethernet MAC Register Addresses**

ADDRESS[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

Table 4-1: **Ethernet MAC Register Addresses *(Cont'd)***

ADDRESS[9:0]	Register Description
0x340	Management Configuration
0x380	Unicast Address (Word 0)
0x384	Unicast Address (Word 1)
0x388	Additional Address Table Access (Word 0)
0x38C	Additional Address Table Access (Word 1)
0x390	Address Filter Mode

Configuration Registers

The Ethernet MAC configuration registers and the contents of the registers are shown in Table 4-2 through Table 4-8.

Table 4-2: **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. The address is ordered so the first byte transmitted/received is the lowest positioned byte in the register; for example, a MAC address of AA-BB-CC-DD-EE-FF is stored in address [47:0] as 0xFFEEDDCCBBAA. Tie to the same value as EMAC#_UNICASTADDR[31:0].	EMAC#_PAUSEADDR[31:0]	R/W

Table 4-3: **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]															

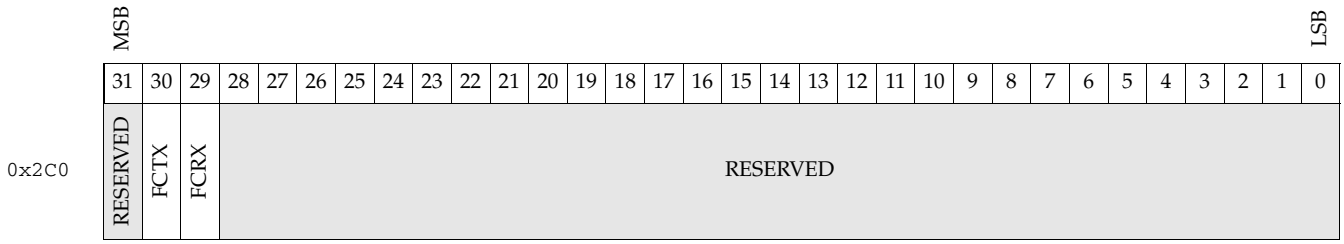
Bit	Description	Default Value	R/W
[15:0]	Pause frame Ethernet MAC Address [47:32]. Tie to the same value as EMAC#_UNICASTADDR[47:32].	EMAC#_PAUSEADDR[47:32]	R/W
[24:16]	Reserved.	–	
[25]	Length/Type Check disable. When this bit is 1, it disables the Length/Type field check on the frame.	EMAC#_LTCHECK_DISABLE	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.	EMAC#_RXHALFDUPLEX	R/W
[27]	VLAN enable. When this bit is 1, the receiver accepts VLAN tagged frames. The maximum payload length increases by four bytes.	EMAC#_RXVLAN_ENABLE	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.	EMAC#_RX_ENABLE	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.	EMAC#_RXINBANDFCS_ENABLE	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.	EMAC#_RXJUMBOFRAME_ENABLE	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.	EMAC#_RXRESET	R/W

Table 4-4: **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.	EMAC#_TXIFGADJUST_ENABLE	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.	EMAC#_TXHALFDUPLEX	R/W
[27]	VLAN enable. When this bit is 1, the transmitter allows transmission of the VLAN tagged frames.	EMAC#_TXVLAN_ENABLE	R/W
[28]	Transmit enable. When this bit is 1, the transmitter is enabled for operation.	EMAC#_TX_ENABLE	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.	EMAC#_TXINBANDFCS_ENABLE	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.	EMAC#_TXJUMBOFRAME_ENABLE	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.	EMAC#_TXRESET	R/W

Table 4-5: **Flow Control Configuration Register**



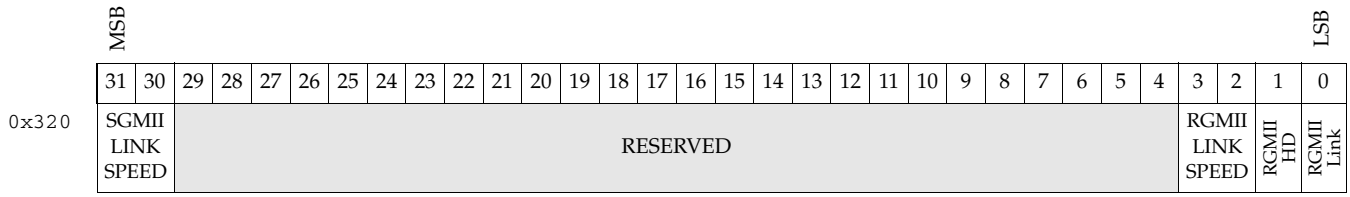
Bit	Description	Default Value	R/W
[28:0]	Reserved.	–	
[29]	Flow control enable (RX). When this bit is 1, the received flow control frames inhibit transmitter operation. When this bit is 0, the flow control frame is passed to the client.	EMAC#_RXFLOWCTRL_ENABLE	R/W
[30]	Flow control enable (TX). When 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.	EMAC#_TXFLOWCTRL_ENABLE	R/W
[31]	Reserved.	–	

Table 4-6: **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.	EMAC#_RX16BITCLIENT_ENABLE	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.	EMAC#_TX16BITCLIENT_ENABLE	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 "Ethernet MAC Attributes" on page 42.	EMAC#_HOST_ENABLE	R
[27]	1000BASE-X mode enable. When this bit is 1, the Ethernet MAC is configured in 1000BASE-X mode.	EMAC#_1000BASEX_ENABLE	R
[28]	SGMII mode enable. When this bit is 1, the Ethernet MAC is configured in SGMII mode.	EMAC#_SGMII_ENABLE	R
[29]	RGMII mode enable. When this bit is 1, the Ethernet MAC is configured in RGMII mode.	EMAC#_RGMII_ENABLE	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	{EMAC#_SPEED_MSB, EMAC#_SPEED_LSB}	R/W

Table 4-7: **RGMII/SGMII Configuration Register**



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 the PHY to the Ethernet MAC, encoded by GMII_RX_DV and GMII_RX_ER during the IFG.	0	R
[1]	RGMII duplex status. Valid in RGMII mode configuration only. This bit is 0 for half-duplex and 1 for full-duplex. This displays the duplex information from the PHY to the 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 the PHY to the Ethernet MAC as encoded by GMII_RX_DV and GMII_RX_ER during the IFG. This two-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 5-19, page 133. This two-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 4-8: **Management Configuration Register**

	MSB 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	LSB 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 Chapter 5, "MDIO Interface."	All 0s	R/W
[6]	MDIO enable. When this bit is 1, the MDIO interface is used to access the PHY. This bit can only be asserted when clock divide is also loaded with a non-zero value. When this bit is 0, the MDIO interface is disabled, and the MDIO signals remain inactive. See Chapter 5, "MDIO Interface."	0	R/W
[31:7]	Reserved.	–	

Address Filter Registers

The Ethernet MAC Address filter has five address filter registers accessible through the host interface. One unicast address can be written into the unicast address register and four additional addresses can be written into the general address table, which is accessed indirectly through the General Address access registers. Figure 4-3 describes the General Address table access.

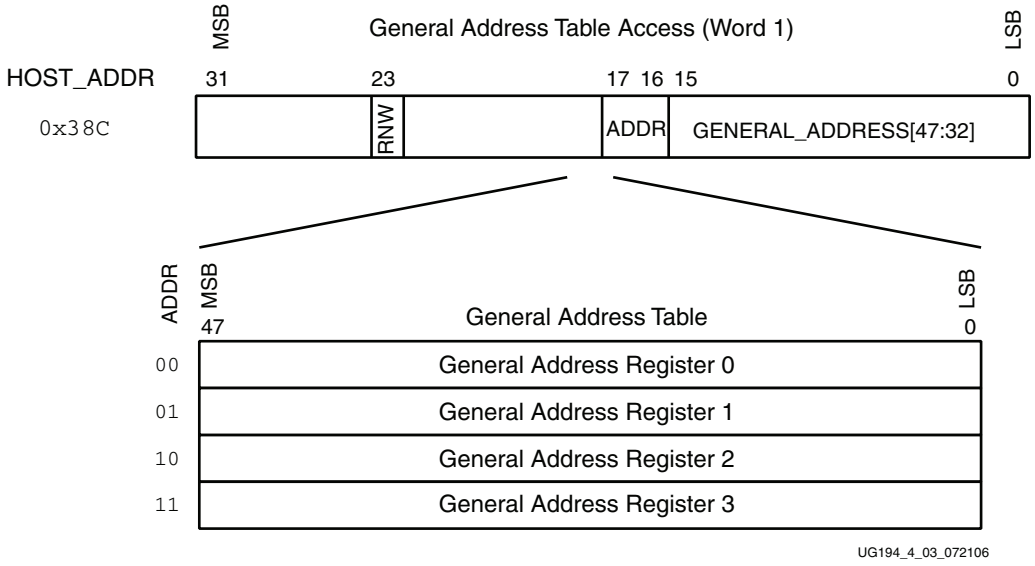
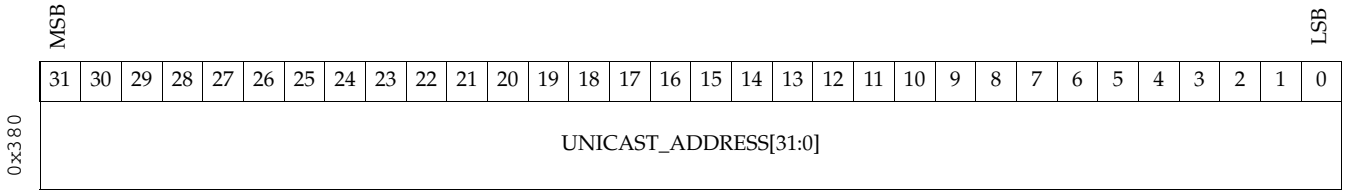


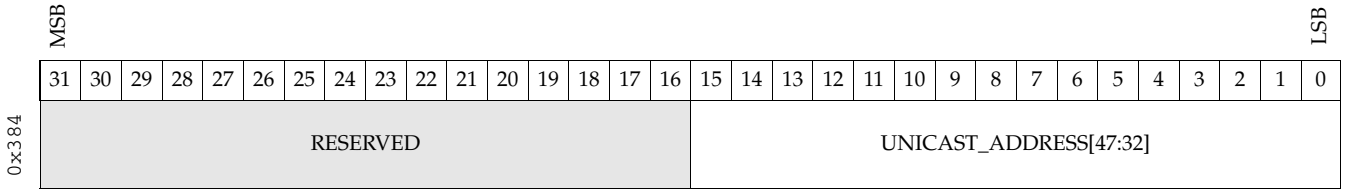
Figure 4-3: **General Address Table Memory Diagram**

Table 4-9: **Unicast Address (Word 0)**



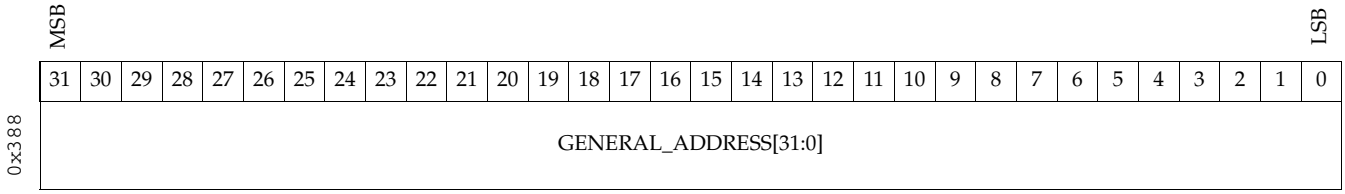
Bit	Description	Default Value	R/W
[31:0]	Unicast Address [31:0]. This address is used to match the Ethernet MAC against the destination address of any incoming frames. The address is ordered so the first byte transmitted/received is the lowest positioned byte in the register; for example, a MAC address of AA-BB-CC-DD-EE-FF is stored in address [47:0] as `0xFFEEDDCCBBAA`.	EMAC#_UNICASTADDR[31:0]	R/W

Table 4-10: **Unicast Address (Word 1)**



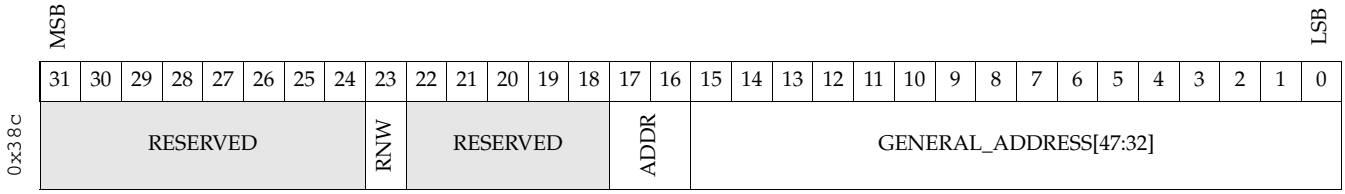
Bit	Description	Default Value	R/W
[15:0]	Unicast Address [47:32]	EMAC#_UNICASTADDR[47:32]	R/W
[31:16]	Reserved.	–	

Table 4-11: **Address Table Access (Word 0)**



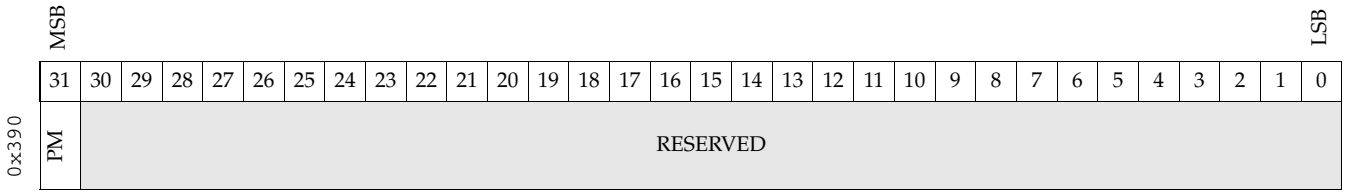
Bit	Description	Default Value	R/W
[31:0]	General Address [31:0]. The general address bits [31:0] are temporarily deposited into this register for writing into a general address register. When a read from an address register has been performed, the address bits [31:0] are accessible from this register. The address is ordered so the first byte transmitted/received is the lowest positioned byte in the register; for example, a MAC address of AA-BB-CC-DD-EE-FF is stored in address [47:0] as `0xFFEEDDCCBBAA`.	All 0s	R/W

Table 4-12: **Address Table Access (Word 1)**



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

Table 4-13: **Address Filter Mode**



Bit	Description	Default Value	R/W
Address Filter Mode			
[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 EMAC#_ADDRFILTERENABLE is FALSE 0: When EMAC#_ADDRFILTERENABLE is TRUE	R/W

Using the Host Bus

To access the host interface using the host bus, the DCREMACENABLE signal should be tied to logic 0.

When using the host bus, the type of transaction initiated is dependant on the host bus signals HOSTMIIMSEL, HOSTADDR[9], and HOSTEMAC1SEL. Table 4-14 shows the access method required for each transaction type.

Table 4-14: **Host Bus Transaction Types**

Transaction	R/W	HOSTEMAC1SEL	HOSTMIIMSEL	HOSTADDR[9]	HOSTOPCODE[1]	HOSTOPCODE[0]
EMAC0 Configuration/Address Filter	Read	0	0	1	1	X
	Write	0	0	1	0	X
EMAC0 MDIO access	Read	0	1	X	1	0
	Write	0	1	X	0	1
EMAC1 Configuration/Address Filter	Read	1	0	1	1	X
	Write	1	0	1	0	X
EMAC1 MDIO access	Read	1	1	X	1	0
	Write	1	1	X	0	1

Clocking Requirements

The HOSTCLK can operate at speeds of up to 125 MHz. This clock can be tied to a user-supplied, 125 MHz input clock to save on clock resources, such as the 125 MHz clock provided into the PHYEMAC#GTXCLK port (see Appendix C, "Virtex-4 to Virtex-5 FPGA Enhancements").

The host clock (HOSTCLK) is used to derive the MDIO clock, MDC. Configuring the Ethernet MAC to derive MDC is detailed in "Accessing MDIO through the Host Interface," page 119.

Reading and Writing MAC Configuration Registers

Figure 4-4 shows the write timing for the configuration registers using the host bus. When accessing the configuration registers (i.e., when HOSTADDR[9] = 1 and HOSTMIIMSEL= 0), HOSTOPCODE[1] functions as an active-Low write enable signal, and HOSTOPCODE[0] is a don't care.

HOSTADDR[9:0] takes the address of the configuration register to be accessed, as described in Table 4-1.

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 of the Ethernet MACs.

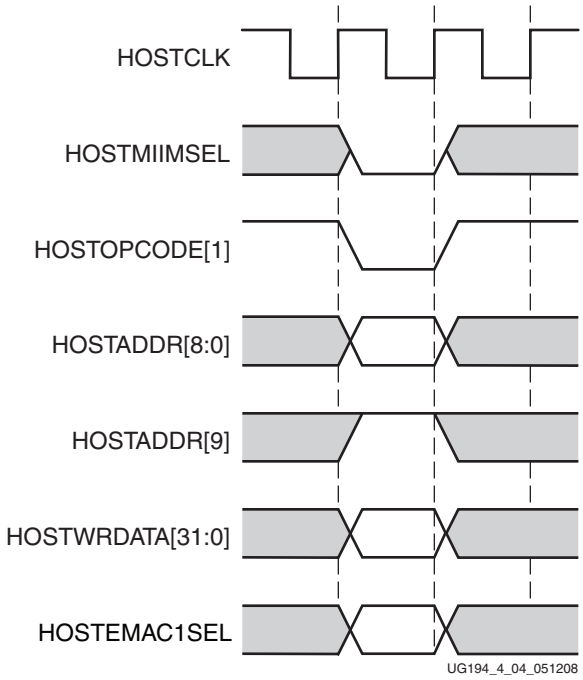


Figure 4-4: **Configuration Register Write Timing**

Figure 4-5 shows the read timing from the configuration registers. A High on HOSTOPCODE[1] indicates a read transaction on the bus. The contents of the register appear on HOSTRDDATA[31:0] on the HOSTCLK edge after the register address is asserted onto HOSTADDR. A Low on HOSTMIIMSEL indicates a configuration access. It must be held Low for an even number of clock cycles during a read operation.

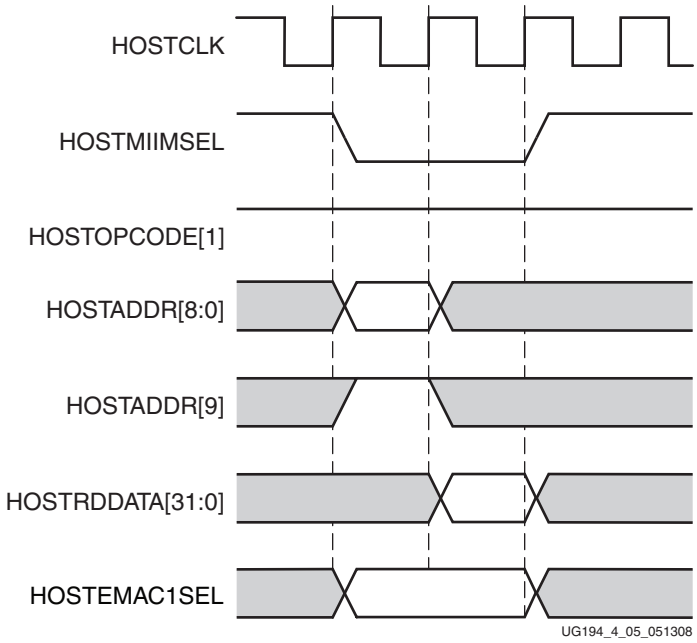


Figure 4-5: **Configuration Register Read Timing**

Reading and Writing Address Filter Registers

Writing to registers in the Address Filter is the same as writing to configuration registers. The timing diagram shown for writing to the Ethernet MAC Configuration Registers (Figure 4-4) holds for Address Filter registers. `HOSTADDR[9:0]` takes the address of the address filter register to be accessed, as described in Table 4-1.

To write a general address register, two write operations are required. First bits [31:0] of the general address must be written into the General Address Table Access Word 0 register. The upper bits [47:32] of the general address must then be written into the General Address Table Access Word 1 register, along with the Address Table address and the RNW bit at logic 0.

Reading the Unicast Address registers and the Address Mode register is also the same as reading from configuration registers. The timing diagram shown for reading the Ethernet MAC configuration registers (Figure 4-5) applies to accesses for these registers.

Figure 4-6 shows the timing diagram for reading the general address from one of the four general address registers in the Address Table.

To read a general address register, a write must first be made to the General Address Table Access Word 1 register. The RNW field is set to 1, and the Address Table address field should be set with the register number to be read. The general address register data is returned in `HOSTRDDATA[31:0]` as shown in Figure 4-6. The LSW is the general address [31:0]. The MSW contains `0x0000` and the general address [47:32].

A Low on `HOSTMIIMSEL` indicates a configuration access. It must be held Low for an even number of clock cycles during a read operation.

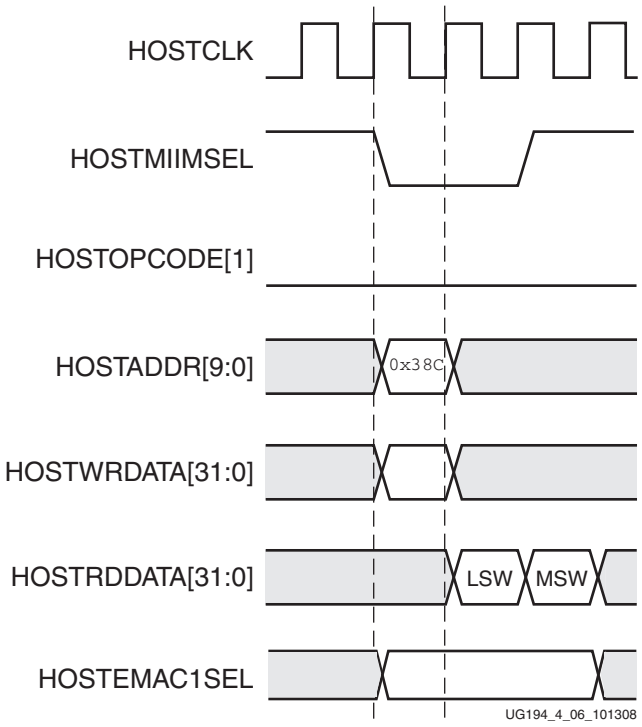


Figure 4-6: **Address Filter General Address Register Read**

PCS/PMA Sublayer or External Device Access via MDIO

Access to the MDIO interface through the management interface is shown in the Figure 4-7 timing diagram. See Chapter 5, "MDIO Interface," for a full description of the MDIO interface and PCS/PMA sublayer management registers. The MDIO interface must be enabled using the Management Configuration Register (see Table 4-8). If the MDIO interface is not enabled, accesses complete in a single cycle (`MIIMRDY` remains High) but are ignored.

In MDIO transactions, the following applies:

- `HOSTMIIMSEL` is held High to indicate an MDIO transaction.
- `HOSTOPCODE` maps to the OPCODE field of the MDIO frame.
- `HOSTADDR` maps to the two address fields of the MDIO frame. PHYAD (the physical address) is `HOSTADDR[9:5]`, and REGAD (the register address) 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.

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 4-7, 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. The bus is locked until the transfer completes.

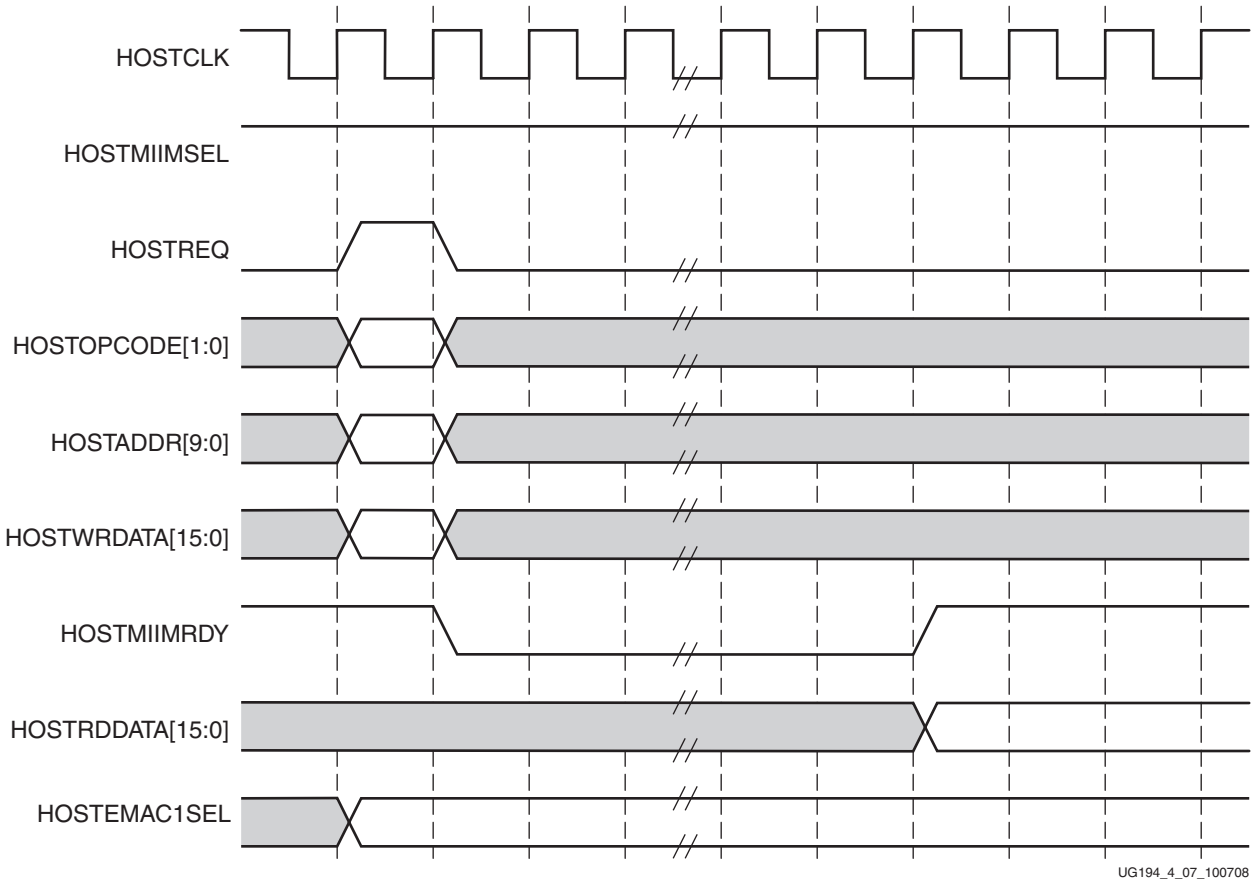


Figure 4-7: **MDIO Access Through the Host Bus**

Using the DCR Bus

The DCREMACENABLE signal selects between the DCR bus or the host bus during the FPGA power-up configuration. To access the host interface through the DCR bus, the DCREMACENABLE signal must be asserted by tying the signal High in the FPGA logic.

The address space available on the DCR bus is limited. The host interface is accessed indirectly using only four registers per EMAC in the DCR address space. Accessing the host interface though the DCR bus requires more software complexity than though the host bus. For further information, see the example software routines in "Interfacing to a Processor," page 107.

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 then configure one of the Ethernet MACs.

Clocking Requirements

The HOSTCLK frequency must be an integer divide of the DCR clock frequency, and the two clocks must be phase-aligned.

Device Control Registers

The DCR bus bridge contains four registers directly addressable through the DCR bus for each Ethernet MAC. Of these registers, three are shared but are addressable from either Ethernet MAC. The upper bits of the DCR address for these registers are user assigned through the EMAC0_DCBASEADDR and EMAC1_DCRBASEADDR attributes.

Figure 4-8 shows how the DCR bridge manages access to the registers within the DCR bus and to the FPGA logic.

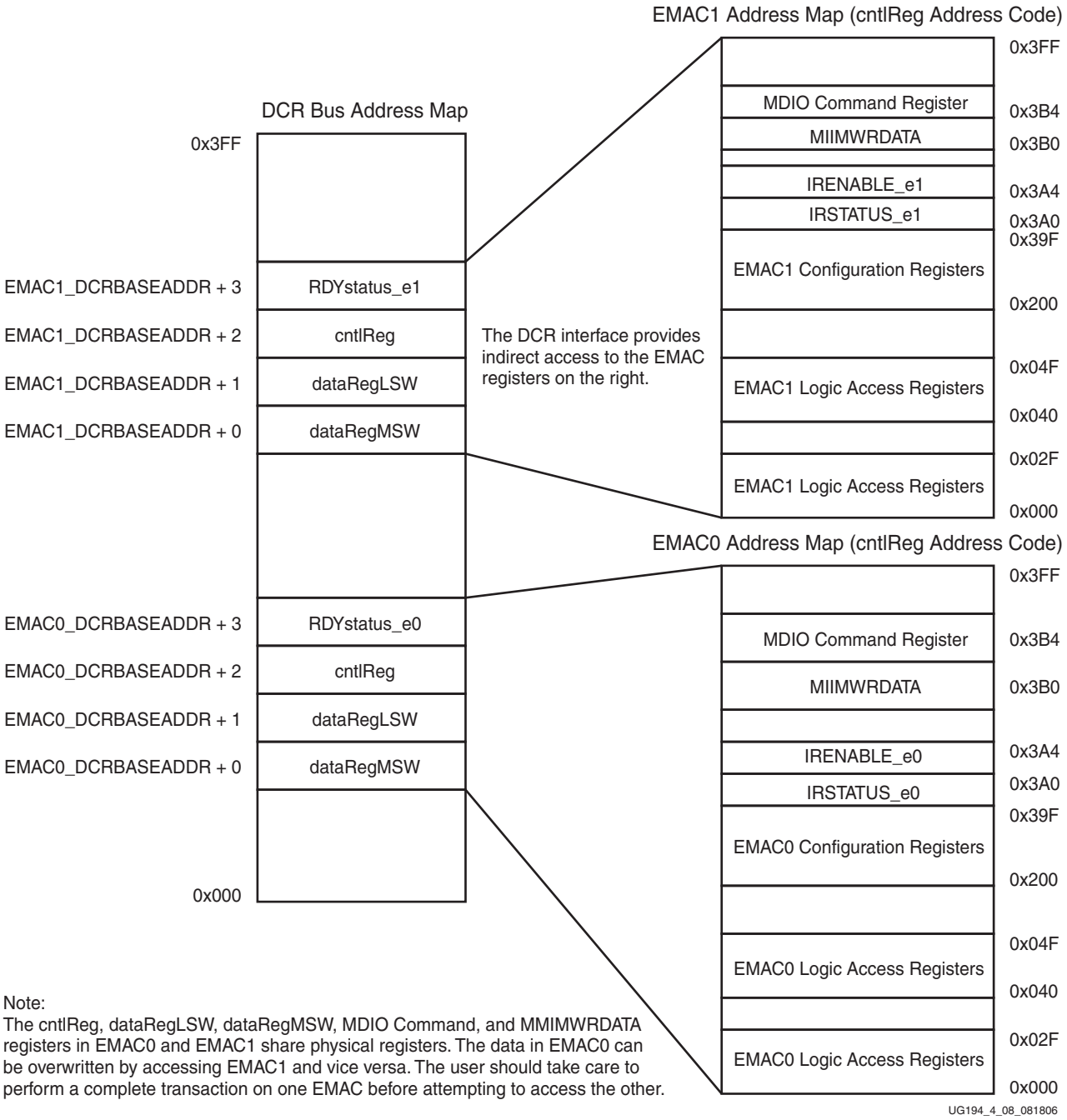


Figure 4-8: **DCR Bridge Registers**

The directly addressable dataRegMSW, dataRegLSW, cntlReg, and RDYstatus_e# registers, can be read or written to using the appropriate DCR address. By writing to the cntlReg register with a particular address code, the DCR bridge can access the indirect address registers within the DCR bus, such as IRSTATUS_e#, the Ethernet MAC configuration registers, FPGA logic-based registers, or perform an MDIO transaction. Data is read and written to indirect registers through the dataRegLSW register. The dataRegMSW register is also used for transactions with an MSW of data. Selection between the EMAC0 and EMAC1 indirect register address space is determined by writing to the cntlReg register at either EMAC0_DCRBASEADDR[0:7]_10 for an EMAC0 access and EMAC1_DCRBASEADDR[0:7]_10 for an EMAC1 access.

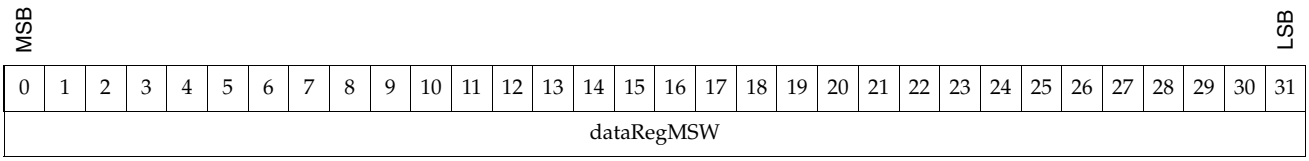
Address codes for each transaction type are summarized in Table 4-15.

Table 4-15: **DCR Address Code Transaction Types**

Address Code[9:0]	Transaction Description	Register Accessible Through dataRegLSW
0x000:0x02F	Read from FPGA logic through unused host pins. See "Accessing FPGA Logic via Unused Host Bus Pins."	User-defined FPGA logic register.
0x040:0x04F	Read from FPGA logic through unused host pins. See "Accessing FPGA Logic via Unused Host Bus Pins."	User-defined FPGA logic register.
0x200:0x390	MAC configuration or address filter register access. See "Ethernet MAC Configuration and Address Filter Access."	MAC configuration or address filter register. Address codes correlate to MAC register addresses in Table 4-1.
0x3A0	Interrupt Status. See "DCR Interrupts."	IRSTATUS_e#
0x3A4	Interrupt Configuration. See "DCR Interrupts."	IRENABLE_e#
0x3B0	MDIO write. See "PCS/PMA Sublayer or External Device Access via MDIO."	MIIMWRDATA_e#
0x3B4	MDIO access. See "PCS/PMA Sublayer or External Device Access via MDIO."	Registers in PCS/PMA sublayer or from an external device via MDIO.

The register definitions for the DCR bridge registers are shown in Table 4-16 to Table 4-19. The DCR registers (dataRegMSW, dataRegLSW, cntlReg, and RDYstatus_e#) and the DCR bridge registers (IRSTATUS_e#, IRENABLE_e#, and MIIMWRDATA) use the big-endian bit numbering convention. 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], bit [1] is mapped directly to little endian bit [30], and so forth).

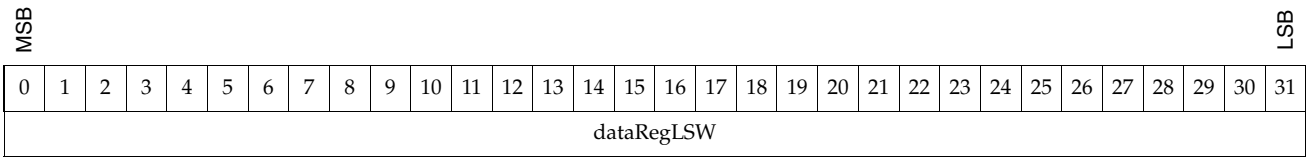
Table 4-16: **DCR Data Register dataRegMSW**



Bit	Description	Default Value
[0:31]	Data. Data input from the DCR bus for the Ethernet MAC registers or other accessible registers is written into this register. The most significant word of data is read out from the Ethernet MAC registers or other registers and deposited into this register. Because dataRegMSW is shared between EMAC0 and EMAC1, data is overwritten by a read/write operation to the other Ethernet MAC.	Undefined

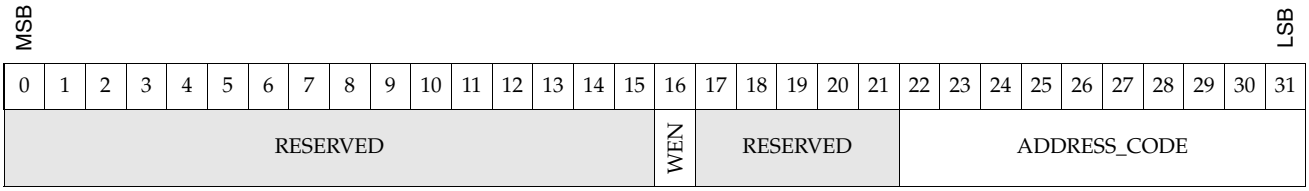
The MSW data register is used when reading from the General Address table and for a read from FPGA logic registers. For all other transactions, only the LSW data register is used.

Table 4-17: **DCR Data Register dataRegLSW**



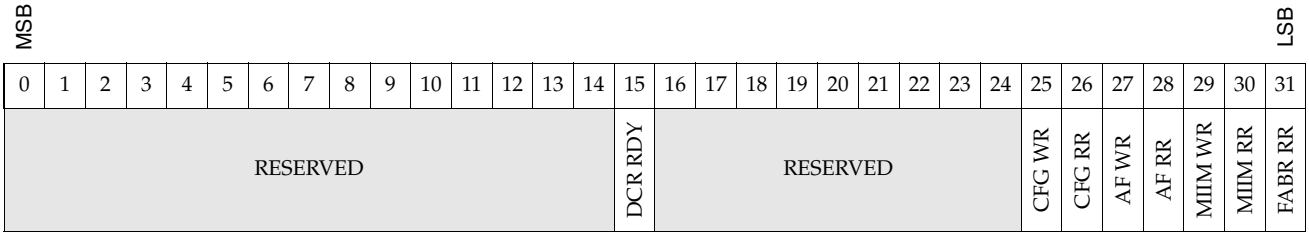
Bit	Description	Default Value
DCR Data Register (dataRegLSW)		
[0:31]	Data. Data input from the DCR bus for the Ethernet MAC registers or other registers is written into this register. The least significant word of data is read out from the Ethernet MAC registers or other registers is deposited into this register. Because dataRegLSW is shared between EMAC0 and EMAC1, data is overwritten by a read/write operation to the other Ethernet MAC.	Undefined

Table 4-18: **DCR Control Register cntlReg**



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 to the Ethernet MAC host interface. When this bit is deasserted, the operation to be performed is read.	0
[17:21]	Reserved.	All 0s
[22:31]	Address Code. The DCR bus bridge translates the address code for this transaction into the appropriate signals on the Ethernet MAC host interface. See Table 4-15, page 103 for a description of address codes.	All 0s

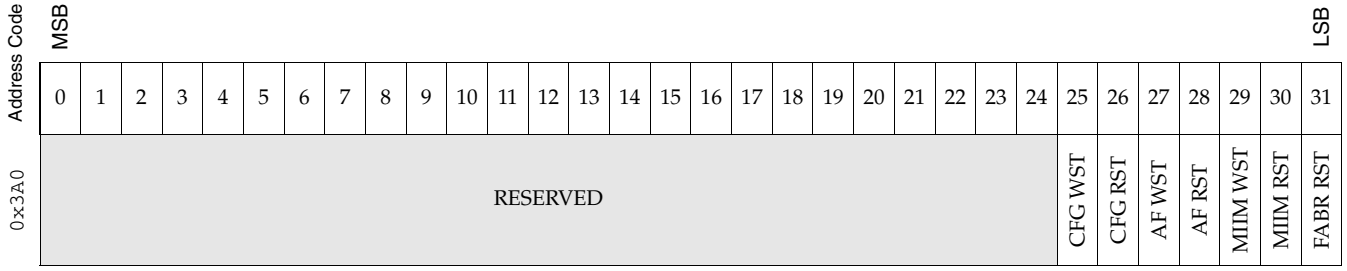
Table 4-19: **DCR Ready Status Register RDYstatus_e# (Read Only)**



Bit	Description	Default Value
[0:14]	Reserved.	All 0s
[15]	DCR Bridge Ready bit. High when all of the bits in RDYstatus_e0 or RDYstatus_e1 are High.	1
[16:24]	Reserved.	All 0s
[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]	FPGA Logic Read-Ready bit.	1

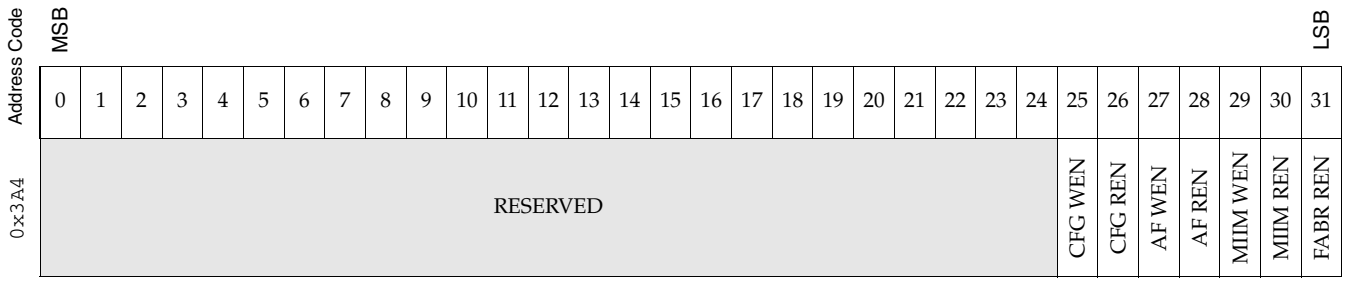
The bits in the Ready Status register are set to 1 when there is no access in progress. When an access is in progress, a bit corresponding to the type of access is automatically cleared. The bit is automatically set again when the access is complete.

Table 4-20: **Interrupt Status Register IRSTATUS_e#**



Bit	Description	Default Value
[0:24]	Reserved.	0
[25]	Configuration Write Interrupt Request bit.	0
[26]	Configuration Read Interrupt Request bit.	0
[27]	Address Filter Write Interrupt Request bit.	0
[28]	Address Filter Read Interrupt Request bit.	0
[29]	MDIO Write Interrupt Request bit.	0
[30]	MDIO Read Interrupt Request bit.	0
[31]	FPGA Logic Read Interrupt Request bit.	0

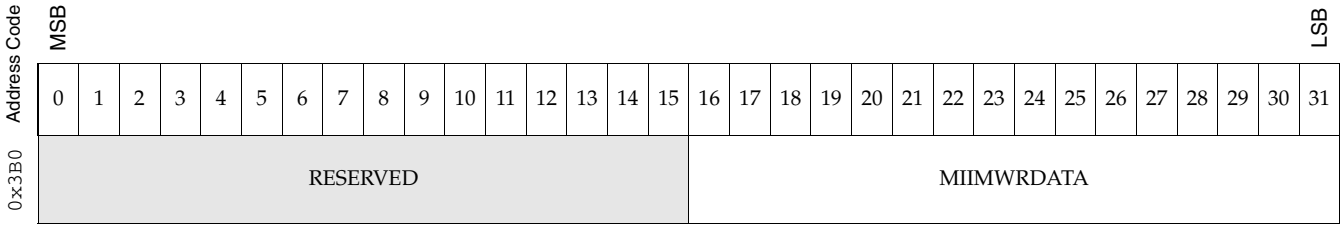
Table 4-21: **Interrupt Enable Register IRENABLE_e#**



Bit	Description	Default Value
[0:24]	Reserved.	0
[25]	Configuration Write IR-enable bit.	0
[26]	Configuration Read IR-enable bit.	0
[27]	Address Filter Write IR-enable bit.	0
[28]	Address Filter Read IR-enable bit.	0
[29]	MDIO Write IR-enable bit.	0
[30]	MDIO Read IR-enable bit.	0
[31]	FPGA Logic Read IR enable bit.	0

The IRENABLE_e# register is programmed to allow updating of the interrupt request in the IRSTATUS_e# register. When an enable bit is cleared, the corresponding bit in the IRSTATUS_e# register is not updated.

Table 4-22: **Host Interface MDIO Write Data Register (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

Interfacing to a Processor

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. While some commands complete within one clock cycle on the DCR bus, others such as MDIO accesses require multiple clock cycles.

There are two ways to use the DCR bus with the host processor. The user can select either polling or interrupts as a method of informing the host processor of access status. The interrupt method is more efficient as it frees the processor to operate while waiting for the DCR transaction to finish.

- Interrupt request informs the host processor when the host interface completes a DCR host access command. The interrupt request signal, `DCRHOSTDONEIR`, is only active when the DCR bus is used, and the host interface register IRENABLE_e# 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 processor needs to service the interrupt request and clear the host interface register (IRSTATUS_e#) to deassert this signal. See "DCR Interrupts" for a description of the DCR Interrupts.
- Polling of the DCR status register RDYstatus_e#. The 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. This bit is automatically reasserted when the access is complete. RDYstatus_e#[15] is asserted only when there are no DCR transactions in progress, allowing the polling process to check this one bit for all transaction types, across both EMACs.

DCR Interrupts

The DCRHOSTDONEIR interrupt can be used to manage processor DCR access. The operation and status of the interrupt can be accessed through the IRSTATUS_e# and IRENABLE_e# registers.

The IRSTATUS_e# register indicates which transaction type has triggered the interrupt. This allows the host to read this register to determine the type of host access completed.

Before exiting the interrupt service routine, the host should write to this register to clear the interrupt request bit.

To read and clear the IRSTATUS_e# register, the following operations must be performed on the DCR bus:

1. To access the IRSTATUS_e# register, write an address code of `0x3A0` to the cntlReg register, addressing cntlReg using EMAC0_DCRBASEADDR for IRSTATUS_e0, and EMAC1_DCRBASEADDR for IRSTATUS_e1. The write enable bit should be cleared to indicate a read.
2. The register information for IRSTATUS_e# is loaded into dataRegLSW. Read this register to access the data.
3. To clear the interrupt, the contents of the IRSTATUS_e# register should be written to zero. Write a value of all zeros into the dataRegLSW.
4. Write the cntlReg with the address `0x3A0` to load the data from dataRegLSW into the register IRSTATUS_e# and clear the interrupt. The write enable bit should be set to 1 to indicate a write.

To read from the EMAC0 IRSTATUS_e0 register and then clear the register:

```
// Write the address of IRStatus_e0 register 0x3A0 to the
// cntlReg_e0 register. Ensure that the write enable bit is not asserted
dcr_write(EMAC0_DCRBASEADDR + 2, 0x000003A0);
// Read the IRSTATUS_e0 data from the dataRegLSW
dcr_read (EMAC0_DCRBASEADDR + 1);
// Write a value of all zeros to the dataRegLSW
dcr_write(EMAC0_DCRBASEADDR + 1, 0x00000000);
// Write to the cntlReg_e0 register to load the data into the
// IRStatus_e0 register to clear the interrupt
dcr_write(EMAC0_DCRBASEADDR + 2, 0x000083A0);
```

The IRENABLE_e# register is programmed to enable the different interrupt types to trigger the DCRHOSTDONEIR interrupt. When an enable bit is set, the corresponding bit in the IRSTATUS_e# register is updated when an interrupt of that type occurs. To read from this register and then write back, the following steps are required.

1. To read the IRENABLE_e# register, write an address code `0x3A4` to the cntlReg, addressing the cntlReg using EMAC0_DCRBASEADDR for IRENABLE_e0, and EMAC1_DCRBASEADDR for IRENABLE_e1, the write enable bit should be cleared to indicate a read operation.
2. The register information for IRENABLE_e# is loaded into the dataRegLSW register. Read this register to access the data.
3. To enable specific interrupts, write the required bitmask value to the dataRegLSW register.
4. Write to the cntlReg with the address code `0x3A4` to load the data from datRegLSW into the IRENABLE_e# register. The write enable bit should be set to 1 to indicate a write operation.

To read from the EMAC1 IRENABLE_e1 register and then enable all interrupts:

```
// Write the address of IRENABLE_e1 register 0x3A4 to the
// cntlReg_e1 register. Ensure that the write enable bit is not asserted
dcr_write(EMAC1_DCRBASEADDR + 2, 0x000003A4);
// Read the IRENABLE_e1 data from the dataRegLSW
dcr_read (EMAC1_DCRBASEADDR + 1);
// Write a value of all ones to the dataRegLSW
dcr_write(EMAC1_DCRBASEADDR + 1, 0x0000007F);
```

```
// Write to the cntlReg_e1 register to load the data into the
// IRENABLE_e1 register and enable all interrupts
dcr_write(EMAC1_DCRBASEADDR + 2, 0x000083A4);
```

Ethernet MAC Configuration and Address Filter Access

Reading from an Ethernet MAC Configuration Register

To read from an Ethernet MAC configuration register, the user must perform the following sequence of operations on the DCR bus:

1. To access EMAC0, use EMAC0_DCRBASEADDR. To access EMAC1, use EMAC1_DCRBASEADDR. Write to the cntlReg register, with the write enable bit cleared to 0, to indicate a read operation and set the address code to the desired address of the Ethernet MAC configuration register to be accessed. The address code maps directly to the Ethernet MAC address codes given in Table 4-1.
2. Poll the RDYstatus_e# register until the configuration Read-Ready bit is set to 1 or wait until the DCRHOSTDONEIR interrupt is asserted.
3. Read from the dataRegLSW register to access data from the Ethernet MAC configuration register.

The following code demonstrates a processor routine to read from an Ethernet MAC configuration register. 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_e0 register
dcr_write(EMAC0_DCRBASEADDR + 2, 0x00000280);
// Poll the RDYstatus_e0 register
while ( !(dcr_read(EMAC0_DCRBASEADDR + 3) & 0x00010000) );
// Read the dataRegLSW with the values returned from the EMAC0
// Transmitter Configuration register
dcr_read (EMAC0_DCRBASEADDR + 1);
```

Writing to an Ethernet MAC Configuration Register

To write to an Ethernet MAC configuration register, the user must perform the following sequence of operations on the DCR bus:

1. Write to the dataRegLSW register with the desired value for the Ethernet MAC configuration register. This value is mapped to the host configuration register.
2. Write to the cntlReg register with the desired address of the Ethernet MAC configuration register and the write enable bit to logic 1. Use EMAC0_DCRBASEADDR to access EMAC0 and EMAC1_DCRBASEADDR to access EMAC1.
3. Poll the RDYstatus_e# register until the configuration Write-Ready bit is set to 1 or wait until the DCRHOSTDONEIR interrupt is asserted.

To write to the EMAC1 flow control register:

```
// EMAC Configuration Register 0x2C0 (EMAC Flow Control)
// Write to enable the flow control on both the transmit and receive
// side of EMAC1, set bits 29 and 30 to 1
dcr_write(EMAC1_DCRBASEADDR + 1, 0x60000000);
// Write the address of Flow Control register to the cntlReg_e1
// register, with the write enable bit asserted
dcr_write(EMAC1_DCRBASEADDR + 2, 0x000082C0);
```

```
// Poll the RDYstatus_e1 register
while ( !(dcr_read(EMAC1_DCRBASEADDR + 3) & 0x00010000) );
```

Reading from the General 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. The only operations that differ are reading and writing from the general address table.

To read the desired general address table register of the address filter, a DCR write operation must be performed.

1. Write to the dataRegLSW register, setting the two-bit address of the general address table of the register to be accessed (four general address table registers are in the address filter block) and the RNW bit to 1.
2. Write to the cntlReg register with the address register of general address (Word 1) to `0x38C`. Set the Write enable bit to write to the general address (Word 1) (see Table 4-12). Use EMAC0_DCRBASEADDR to access EMAC0 and EMAC1_DCRBASEADDR to access EMAC1.
3. Poll the RDYstatus_e# register until the address filter Read-Ready bit is set to 1 or wait until the `DCRHOSTDONEIR` interrupt is asserted.
4. Read from the dataRegMSW to read the general address [47:32] and dataRegLSW to read general address [31:0].

To read from the general 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
dcr_write(EMAC0_DCRBASEADDR + 1, 0x00820000);
// Write the address of EMAC0 General Address register to the cntlReg
// register, with the write enable bit asserted
dcr_write(EMAC0_DCRBASEADDR + 2, 0x0000838C);
// Poll the RDYstatus_e0 register
while ( !(dcr_read(EMAC0_DCRBASEADDR + 3) & 0x00010000) );
// Read the values returned of the General Address Word0 and Word1
// registers (48-bit value)
// from the dataRegMSW and dataRegLSW registers
gen_addr_msw = dcr_read(EMAC0_DCRBASEADDR + 0);
gen_addr_lsw = dcr_read(EMAC0_DCRBASEADDR + 1);
```

Writing to the General 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. The only operations that differ are reading and writing from the general address table.

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

1. Write to the dataRegLSW register the general address [31:0] to be stored on the desired general address table register.
2. Write to the cntlReg# register with the address for general address word 0 (`0x388`), setting the write enable bit. Use EMAC0_DCRBASEADDR to access EMAC0 and EMAC1_DCRBASEADDR to access EMAC1.
3. Poll the RDYstatus_e# register until the address configuration write bit is set to 1.

4. Write to the dataRegLSW register the general address [47:32] with the write mask bit and the value of the general address table register to be accessed.
5. Write to cntlReg register with the address for general address word 1 (0x38C). Set the write enable bit.
6. Poll the RDYstatus_e# register until the address configuration write bit is set or wait until the DCRHOSTDONEIR interrupt is asserted.

To write the general address 0xFACEDEAFCAFE to the general address table register 0x1 of EMAC1:

```
// Write the general address[31:0] to the dataRegLSW register
dcr_write(EMAC1_DCRBASEADDR + 1, 0xDEAFCAFE);
// Write the address of EMAC1 General Address Word 0 register to the
// cntlReg_e1 register
dcr_write(EMAC1_DCRBASEADDR + 2, 0x00008388);
// Poll the RDYstatus_e1 register
while ( !(dcr_read(EMAC1_DCRBASEADDR + 3) & 0x00010000) );
// MULTI_ADDR Register 1 of AF Block
// Write the general address [47:32] with the MULTI_ADDR RNW
// write mask bit deasserted to the dataRegLSW register
dcr_write(EMAC1_DCRBASEADDR + 1, 0x0001FACE);
// Write the address of EMAC1 General Address Word 1 register to the
// cntlReg_e1 register
dcr_write(EMAC1_DCRBASEADDR + 2, 0x0000838C);
// Poll the RDYstatus_e1 register
while ( !(dcr_read(EMAC1_DCRBASEADDR + 3) & 0x00010000) );
```

PCS/PMA Sublayer or External Device Access via MDIO

MDIO register reads take a multiple number of host clock cycles depending on the physical interface device. To write to any of the PCS layer registers ("1000BASE-X PCS/PMA Management Registers," page 122), the data must be written to the MIIMWRDATA register. The physical address (PHYAD) and PCS register number (REGAD) are written to the DCR dataRegLSW register for both read and write operations. Writing the address code 0x3B4 to the control register indicates a request for an MDIO transaction. The format for writing the PHYAD and REGAD into the dataRegLSW register is shown in Figure 4-9. To access the internal PCS/PMA sublayer registers, the PHYAD should match that set on the PHYEMAC#PHYAD[4:0] input port. The MDIO interface is described in detail in Chapter 5.

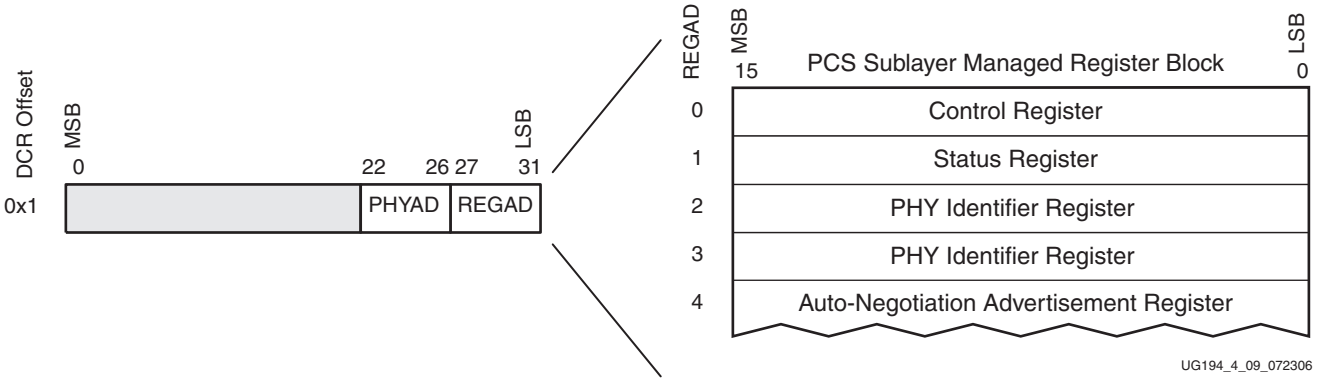


Figure 4-9: **MDIO Address Register to Access PCS Sublayer Register Block**

Reading the PHY Registers using MDIO

To read from a configuration register either in the Ethernet MAC PCS/PMA layer or from an external PHY, the user must perform the following sequence of operations on the DCR bus:

1. Write to the dataRegLSW register with the PHYAD and register to be accessed. This value should be formatted as shown in Figure 4-9.
2. Write to the cntlReg register with the decode address for MDIO transaction (0x3B4) and the write enable bit cleared to indicate a read operation. Use EMAC0_DCRBASEADDR to perform the transaction on the EMAC0 MDIO interface and EMAC1_DCRBASEADDR to perform the transaction on the EMAC1 MDIO interface.
3. Poll the RDYstatus_e# register until the MDIO Write-Ready bit is set or wait until the DCRHOSTDONEIR interrupt is asserted.
4. Read the PHY register contents from the dataRegLSW.

To read PHYAD 0x1 and PHY register 0x4 through the EMAC0 MDIO interface, MDIO must be enabled by writing to the management configuration register with the clock divider for MDC, as described in "Writing to an Ethernet MAC Configuration Register."

```
// Write the PHY address and PHY register to be accessed to the
// dataRegLSW register
dcr_write(EMAC0_DCRBASEADDR + 1, (0x1 << 5) + 0x4);
// Write the decode address for MDIO address output to the cntlReg_e0
// register
dcr_write(EMAC0_DCRBASEADDR + 2, 0x000003B4);
// Poll the RDYstatus_e0 register
while ( !(dcr_read(EMAC0_DCRBASEADDR + 3) & 0x00010000) );
// Read the PHY register contents from dataRegLSW
dcr_read(EMAC0_DCRBASEADDR + 1);
```

Writing to the PHY Registers using MDIO

The MIIMWRDATA register temporarily holds MDIO write data for output to the MDIO write data bus. To write to a configuration register either in the Ethernet MAC PCS/PMA layer or to an external PHY, the user must perform the following sequence of operations on the DCR bus.

1. Write to the dataRegLSW register with the data to be written to the PHY register.
2. Write to the cntlReg_e# register with the address for the MDIO write data register (0x3B0). This operation loads the write data from dataRegLSW into the MIIMWRDATA register.
3. Write to the dataRegLSW register with the PHY address and register to be accessed. This operation should be formatted as shown in Figure 4-9.
4. Write to the cntlReg register the decode address for MDIO transaction (0x3B4) with the write enable bit set to indicate a write operation. Use EMAC0_DCRBASEADDR to perform the transaction on the EMAC0 MDIO interface and EMAC1_DCRBASEADDR to perform the transaction on the EMAC1 MDIO interface.
5. Poll the RDYstatus_e# register until the MDIO Write-Ready bit is set or wait until the DCRHOSTDONEIR interrupt is asserted.

To write 0x1140 to PHYAD 0x2 and PHY register 0x0 of EMAC1, MDIO must be enabled by writing to the management configuration register with the clock divider for MDC, as described in "Writing to an Ethernet MAC Configuration Register."

```
// EMAC Management Register 0x340 (EMAC1 Management Configuration)
// Write the PHY data to the dataRegLSW
dcr_write(EMAC1_DCRBASEADDR + 1, 0x00001140);
// Write the address for MIIMWRDATA to the cntlReg_e1
// register to load the write data to MIIMWRDATA
dcr_write(EMAC1_DCRBASEADDR + 2, 0x000083B0);
// Write the PHY address and PHY register to be accessed to the
// dataRegLSW register
dcr_write(EMAC1_DCRBASEADDR + 1, (0x2 << 5) + 0x0);
// Write the decode address for MDIO address output to the cntlReg_e1
// register to initiate the transaction
dcr_write(EMAC1_DCRBASEADDR + 2, 0x000083B4);
// Poll the RDYstatus_e1 register
while ( !(dcr_read(EMAC1_DCRBASEADDR + 3) & 0x00010000) );
```

Accessing FPGA Logic via Unused Host Bus Pins

The host bus I/O pins of the Ethernet MAC are enabled for access to the FPGA logic when a DCR read operation is made for address codes 0x000 to 0x02F and 0x040 to 0x04F inclusive (Table 4-15 describes the DCR address code space). This use of the host bus I/O signals provides a means of accessing the FPGA logic from the processor with space for 64 addresses for each EMAC.

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 4-23 and Figure 4-10. These signals can be used to access an FPGA logic-based Statistics block or other IP. Data can be read in from the logic from HOSTWRDATA[31:0], as shown in Figure 4-10. The data read from the host bus can then be accessed by reading the DCR data registers.

Table 4-23: **DCR to Host Mapping**

EMAC Host Bus Port		Port Direction on EMAC	DCR Register Decode	Equivalent Stand-alone Host Bus Signal
HOSTRDDATA	[15]	Out		HOSTREQ
	[14:13]	Out		HOSTOPCODE[1:0]
	[10]	Out	logic 1 if access to cntlReg_e1, logic 0 if access to cntlReg_e0	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

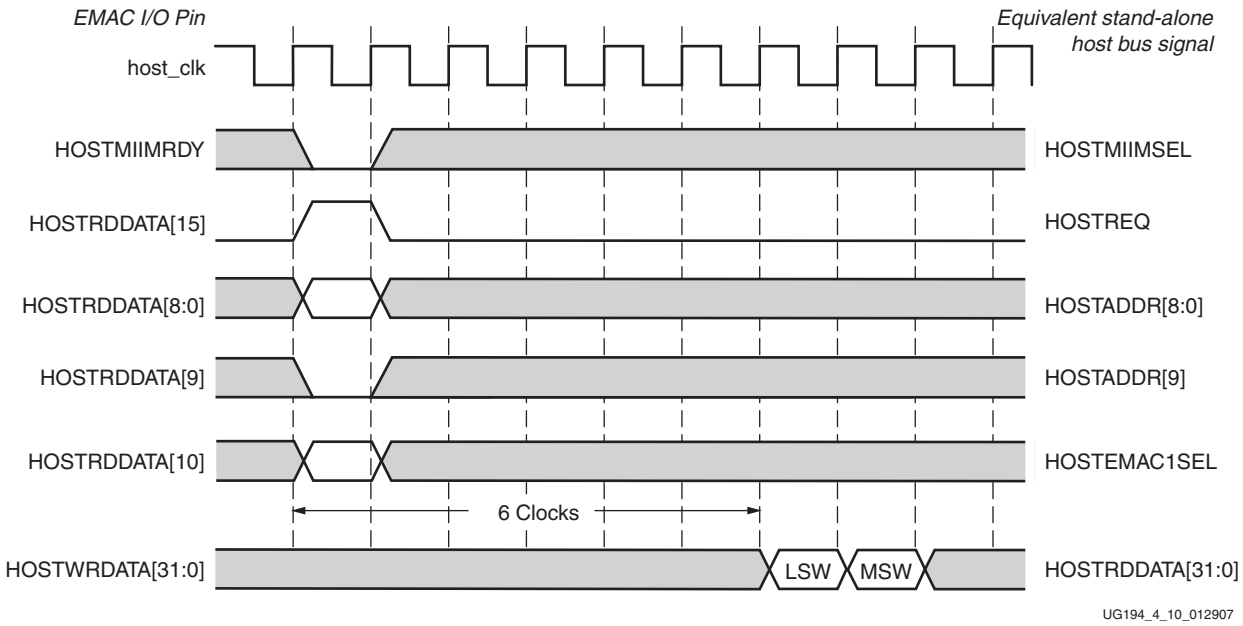


Figure 4-10: **FPGA-Logic Based Register Read Timing**

To read from the FPGA logic, the following operations should take place on the DCR bus. Which Ethernet MAC registers are used only affects the encoding of the I/O pin `HOSTRDDATA[10]`, the EMAC1SEL encoding. Otherwise output signals are identical.

1. Write to the cntlReg register with the address code in the range `0x000` to `0x02F` or `0x040` to `0x04F`. This address is encoded onto the `HOSTRDDATA[9:0]` I/O pins. The write enable bit is cleared to indicate a read operation.
2. Poll the RDYstatus_e# register until the Statistics IP Read-Ready bit is set or wait until the `DCRHOSTDONEIR` interrupt is asserted.
3. Read the dataRegMSW and dataRegLSW registers to get the contents of the FPGA logic-based registers.

To read a logic-based register addressed `0x020`:

```
// FPGA Logic Register at 0x020 (User defined)
// Write the decode address for FPGA logic access to the cntlReg_e0
register
dcr_write(EMAC0_DCRBASEADDR + 2, 0x00000020);
// Poll the RDYstatus_e0 register
while ( !(dcr_read(EMAC0_DCRBASEADDR + 3) & 0x00010000) );
// Read the value from MSW and LSW registers
fabric_msw = dcr_read(EMAC0_DCRBASEADDR + 0);
fabric_lsw = dcr_read(EMAC0_DCRBASEADDR + 1);
```

This method can be used to connect the Ethernet MAC to the statistics counters as described in Chapter 7, "Interfacing to a Statistics Block."

XILINX®

Chapter 5

MDIO Interface

This chapter describes the Management Data Input/Output (MDIO) interface that is used to access physical device registers, both external to the Ethernet MAC (for example an external PHY device) and internal to the Ethernet MAC (the physical device registers present in the PCS/PMA sublayer block).

The internal physical device (i.e., the PCS/PMA sublayer block), is capable of 1000BASE-X PCS/PMA and SGMII operation (see Chapter 6, "Physical Interface"). These two modes of operation each have associated management registers.

This chapter contains the following sections:

Introduction to MDIO

MDIO Bus System

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). The maximum permitted frequency of MDC is set at 2.5 MHz.

Figure 5-1 illustrates an example MDIO bus system.

- An Ethernet MAC is shown as the MDIO bus master (the Station Management (STA) entity).
- Two PHY devices are shown connected to the same bus, both are MDIO slaves (MDIO Managed Device (MMD) entities).

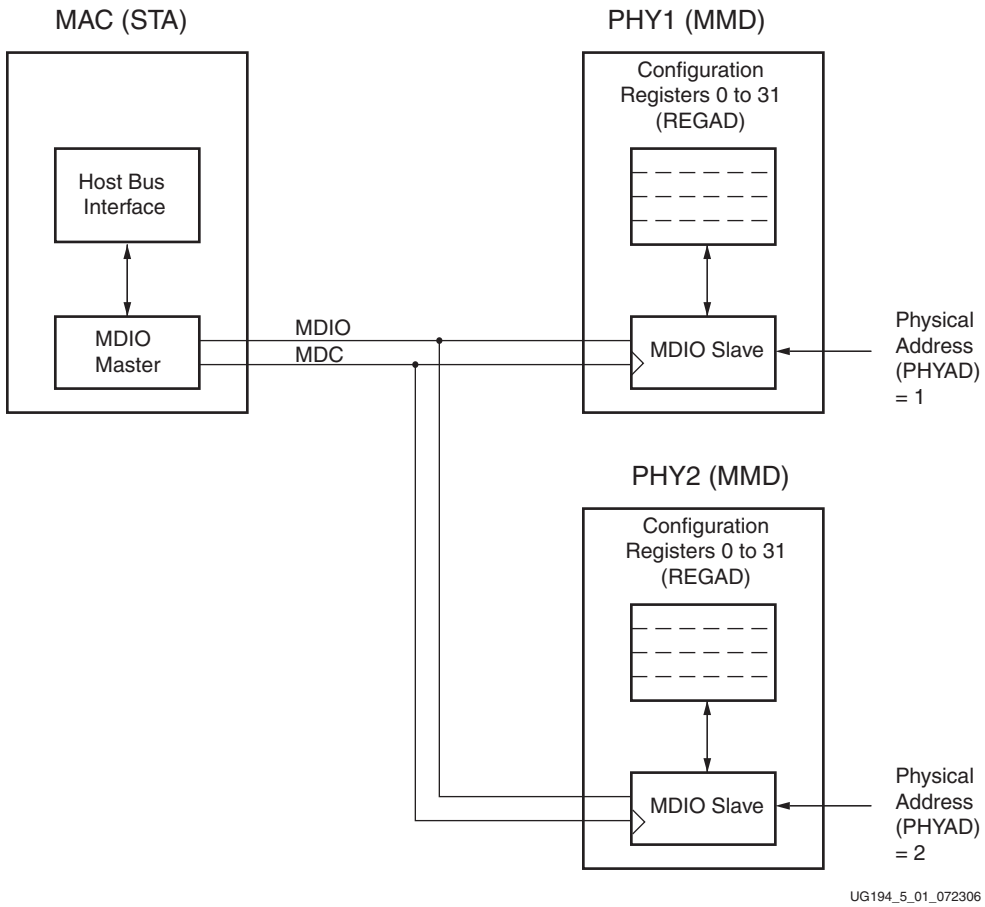


Figure 5-1: **Typical MDIO-Managed System**

Described simply, the MDIO bus system is a standardized interface for accessing the configuration and status registers of Ethernet PHY devices. In the example illustrated, the Management Host Bus Interface of the Ethernet MAC is able to access the configuration and status registers of two PHY devices via the MDIO bus.

MDIO Transactions

All transactions, read or write, are initiated by the MDIO master. All MDIO slave devices, when addressed, must respond. MDIO transactions take the form of an MDIO frame, containing fields for transaction type, address and data. This MDIO frame is transferred across the MDIO wire synchronously to MDC. The following abbreviations are used in this chapter:

Table 5-1: **Abbreviations Used in this Chapter**

Abbreviation	Definition
OP	Operation code (read or write)
PHYAD	Physical address
PRE	Preamble
REGAD	Register address
ST	Start of frame
TA	Turnaround

Write Transaction

Figure 5-2 shows a write transaction frame across the MDIO, as defined by OP = `0b01`. The addressed PHY device (with physical address PHYAD) takes the 16-bit word in the Data field and writes it to the register at REGAD.

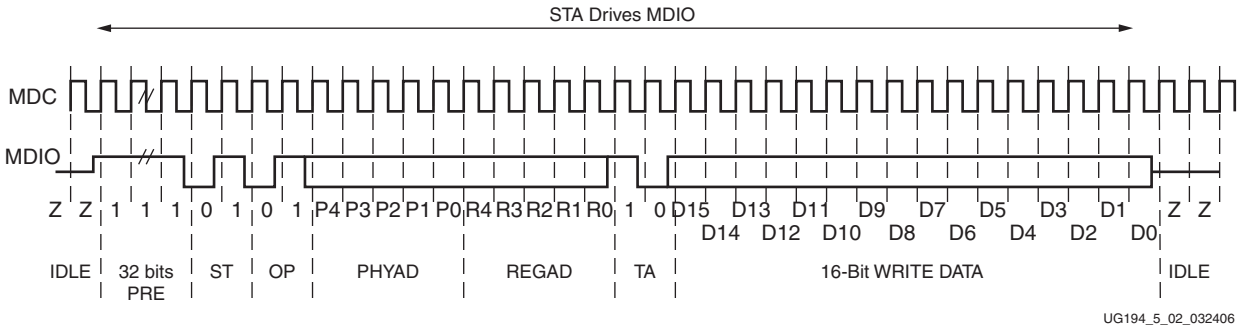


Figure 5-2: **MDIO Write Transaction**

Read Transaction

Figure 5-3 shows a read transaction as defined by OP = 0b10. The addressed PHY device (with physical address PHYAD) takes control of the MDIO wire during the turnaround cycle and then returns the 16-bit word from the register at REGAD.

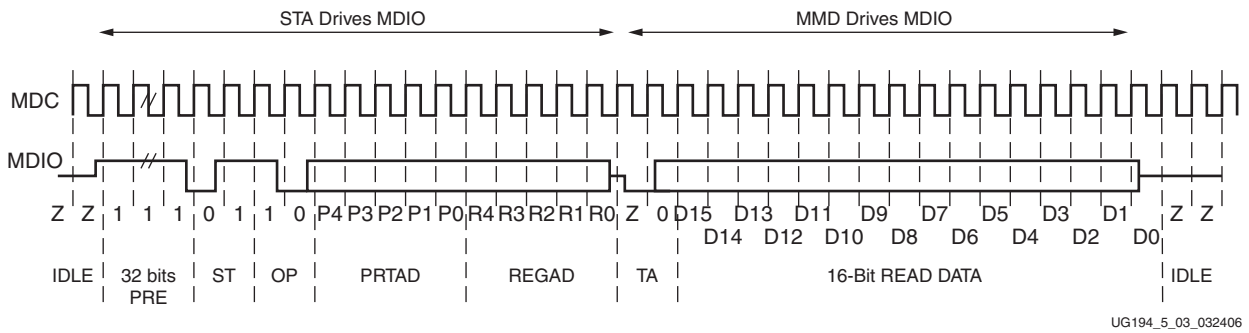


Figure 5-3: **MDIO Read Transaction**

MDIO Addressing

MDIO addresses consists of two stages: physical address (PHYAD) and register address (REGAD).

Physical Address (PHYAD)

As shown in Figure 5-1, two PHY devices are attached to the MDIO bus. Each of these has a different physical address. To address the intended PHY, its physical address should be known by the MDIO master (in this case an Ethernet MAC) and placed into the PHYAD field of the MDIO frame (see "MDIO Transactions").

The PHYAD field for an MDIO frame is a 5-bit binary value capable of addressing 32 unique addresses. However, every MDIO slave must respond to physical address 0. This requirement dictates that the physical address for any particular PHY must not be set to 0 to avoid MDIO contention. Physical Addresses 1 through to 31, therefore, can be used to connect up to 31 PHY devices onto a single MDIO bus.

Physical Address 0 can be used to write a single command that is obeyed by all attached PHYs, such as a reset or power-down command.

Register Address (REGAD)

Having targeted a particular PHY using PHYAD, the individual configuration or status register within that particular PHY must now be addressed by placing the individual register address into the REGAD field of the MDIO frame (see "MDIO Transactions").

The REGAD field for an MDIO frame is a 5-bit binary value capable of addressing 32 unique addresses. The first 16 of these registers (0 to 15) are defined by the IEEE 802.3. The remaining 16 registers (16 to 31) are reserved for PHY vendors' own register definitions.

MDIO Implementation in the Ethernet MAC

Figure 5-4 shows the functional block diagram of the Ethernet MAC with the host interface, DCR bridge, MDIO interface, PCS management, and MDIO intersection blocks highlighted.

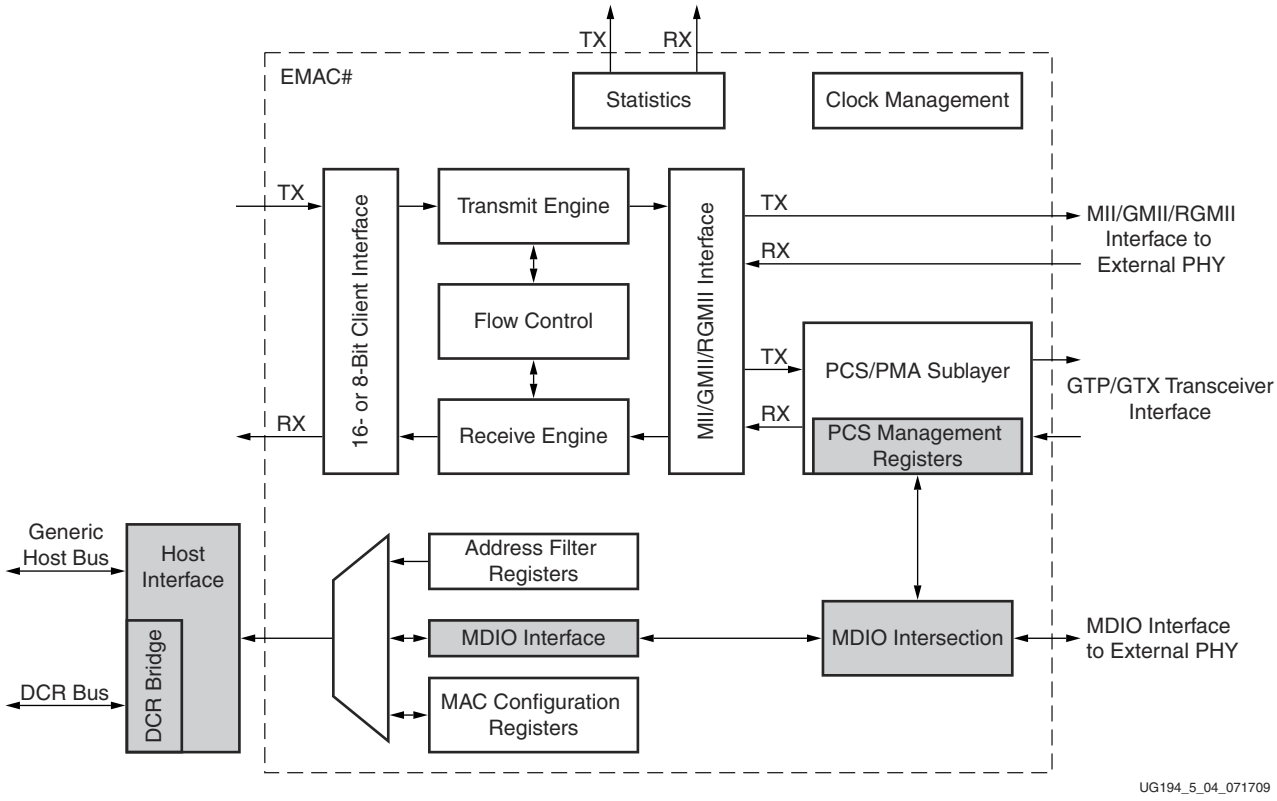


Figure 5-4: **MDIO Implementation of the Ethernet MAC**

Accessing MDIO through the Host Interface

The Ethernet MAC contains an MDIO interface block which is an MDIO master. This can initiate MDIO transactions when accessed through either the generic host bus or the DCR bus. See "PCS/PMA Sublayer or External Device Access via MDIO," page 100, for reference.

The MDIO interface supplies a clock, `EMAC#PHYMCLKOUT`, to the external devices 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 Equation 5-1:

$$f_{MDC} = \frac{f_{HOSTCLK}}{((1 + \text{CLOCK\_DIVIDE}[5:0]) \times 2)} \qquad \textit{Equation 5-1}$$

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. Upon reset, the MDIO port is disabled until a non-zero value has been written into the clock divide bits.

From the MDIO Interface block, the MDIO bus is internally connected within the Ethernet MAC to the MDIO Intersection block. Here the MDIO bus is split. It is internally connected to the 1000BASE-X PCS/PMA sublayer logic and connected to the Ethernet MAC MDIO ports (see "Management Data Input/Output (MDIO) Signals," page 37), which can provide MDIO access to and from an external PHY device. See Table 2-16, page 43. The EMAC#_MDIO_ENABLE attribute must be set (or MDIO Enable must be set in the Management Configuration Register) to allow access to the internal MDIO registers and the external MDIO bus.

Accessing the Internal PCS/PMA Sublayer Management Registers

The PCS/PMA sublayer logic, illustrated as a block in Figure 5-4, contains an MDIO slave, allowing access to its configuration and status registers. All connections are internal and active whenever MDIO operation is enabled.

The PHYAD of the internal PCS/PMA sublayer registers can be set with the `PHYEMAC#PHYAD[4:0]` port. To access the PCS/PMA sublayer registers, simply initiate an MDIO transaction using the matching physical address. The PCS/PMA sublayer also responds to Physical Address 0 (see "MDIO Addressing," page 118).

The PCS/PMA Management registers have different definitions depending on the mode of operation. See as appropriate:

- "1000BASE-X PCS/PMA Management Registers"
- "SGMII Management Registers"

When no external PHY is present, the `PHYEMAC#MDIN` port should be tied High and the `PHYEMAC#MCLKIN` port tied Low.

Accessing the Management Registers of an External PHY using MDIO

Whenever an MDIO operation is enabled (see Table 2-16, page 43), connections can be made to the Ethernet MAC MDIO ports (see "Management Data Input/Output (MDIO) Signals," page 37), which can provide MDIO access to and from an external PHY device.

`PHYEMAC#MDIN`, `EMAC#PHYMDOUT`, and `EMAC#PHYMDTRI` must be connected to a three-state buffer to create the bidirectional wire, MDIO. This three-state buffer can be either external to the FPGA or internally integrated using an IOBUF with an appropriate I/O standard for the external PHY. This is illustrated in Figure 5-5.

In this mode of operation, the MDC clock is generated by the Ethernet MAC and obtained from the output port `EMAC#PHYMCLKOUT`; the input `PHYEMAC#MCLKIN` port is unused and should be connected to logic 0.

To access the external PHY registers, simply initiate an MDIO transaction using the appropriate physical address. The PCS/PMA sublayer, even if it is not the physical interface, is still connected and responds to physical addresses 0 and the address matching the value of the `PHYEMAC#PHYAD[4:0]` port.

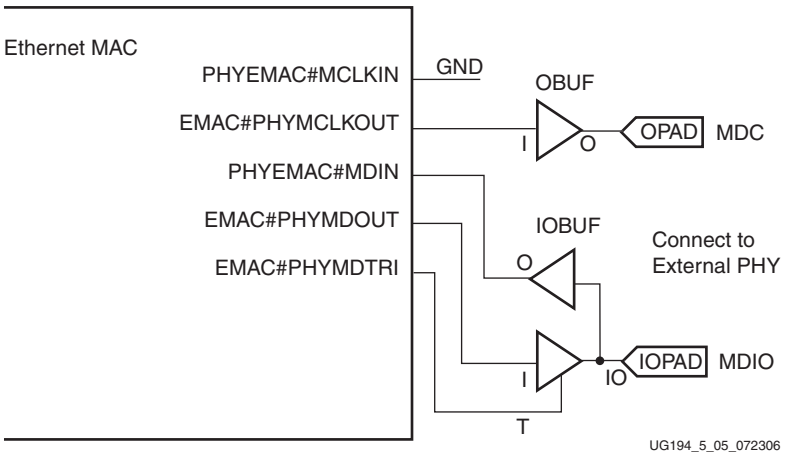


Figure 5-5: **MDIO Access to External PHY**

Accessing PCS/PMA Sublayer Management Registers using MDIO

Figure 5-6 shows the functional block diagram of the Ethernet MAC with the PCS management register, MDIO intersection, and MDIO master blocks highlighted. This figure shows that the PCS/PMA sublayer registers can still be accessed via MDIO when the host interface is not in use.

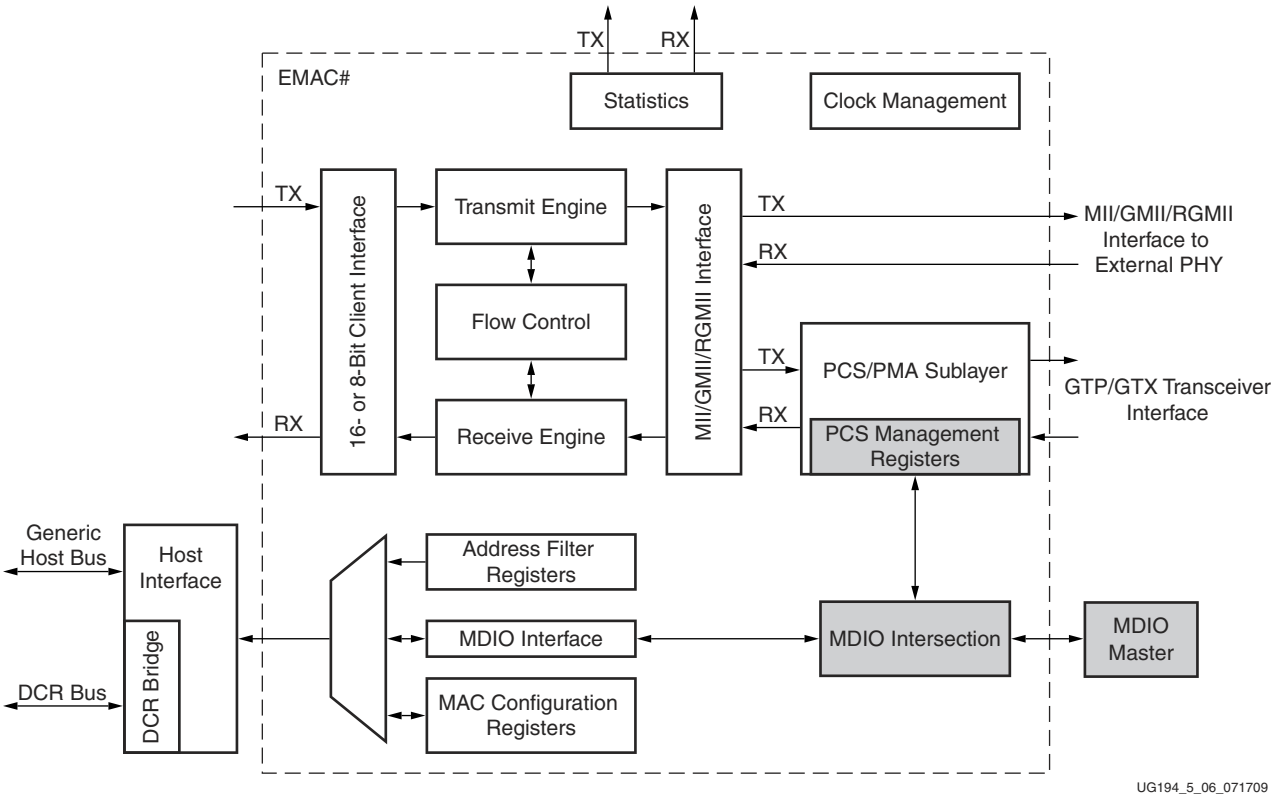


Figure 5-6: **Accessing PCS/PMA Sublayer Management Registers in the Ethernet MAC**

To enable this functionality, the host interface should be disabled, and MDIO operation should be enabled (see Table 2-16, page 43).

In this situation, an MDIO master external to the Ethernet MAC must initiate the "MDIO Transactions." This MDIO master can exist in FPGA logic or as an external device. In this situation, the MDC clock must be provided to the Ethernet MAC through the `PHYEMAC#MCKIN` port. The output signal `EMAC#PHYMCLKOUT` should be left unconnected.

The PCS/PMA Management registers have different definitions depending on the mode of operation. See as appropriate:

- "1000BASE-X PCS/PMA Management Registers"
- "SGMII Management Registers"

1000BASE-X PCS/PMA Management Registers

The PCS/PMA sublayer block, shown in Figure 5-7, is contained within the Ethernet MAC. The PCS/PMA sublayer contains PCS Management registers, as listed in Table 5-2. These registers are described in further detail in Table 5-3 through to Table 5-13.

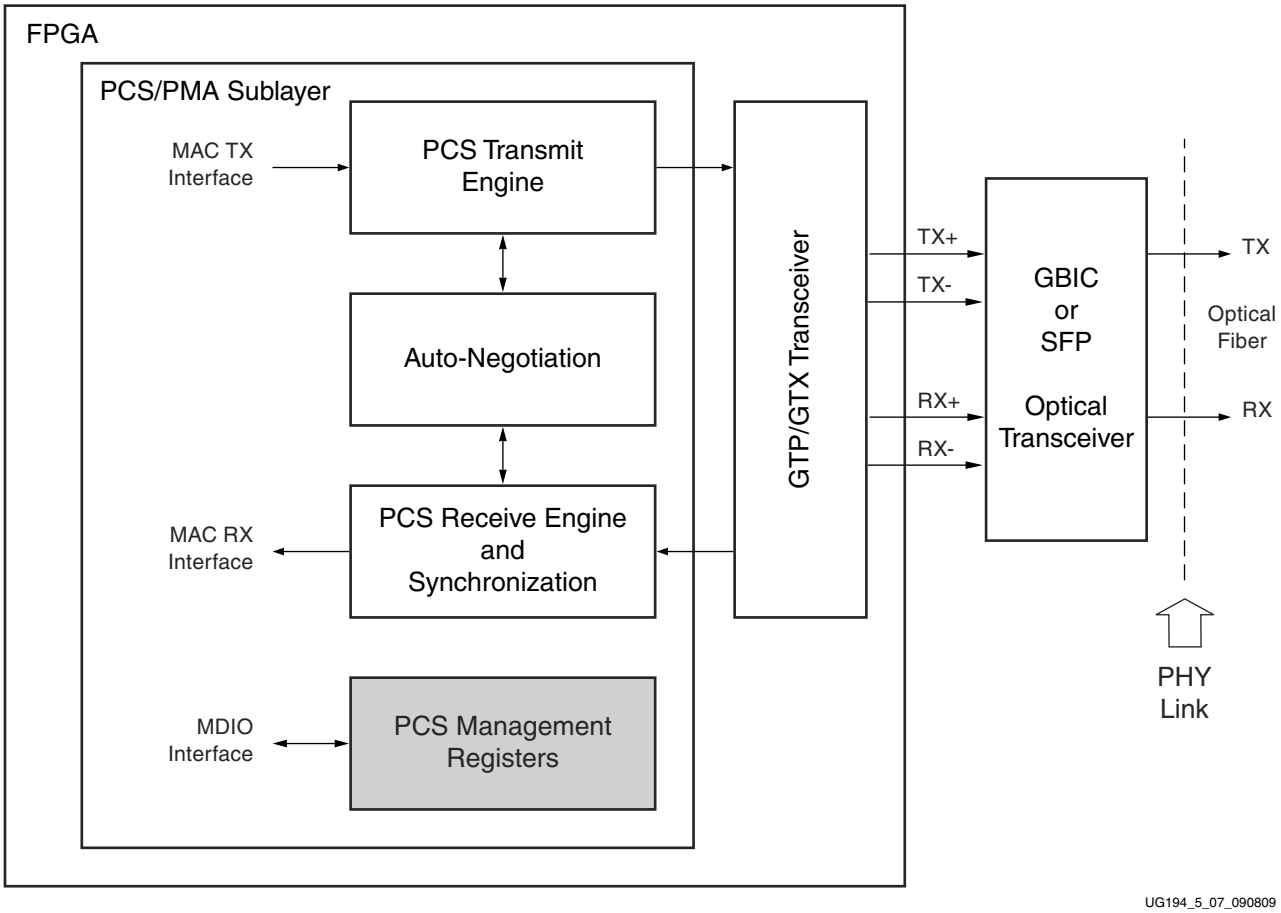


Figure 5-7: **1000BASE-X PCS/PMA Sublayer Logic and Physical Connections**

Registers 0 through to 15, shown in Figure 5-7, are defined in IEEE Std 802.3. These contain information relating to the operation of the 1000BASE-X PCS/PMA sublayer, including the status of the physical Ethernet link (PHY link). Additionally, these registers are directly

involved in the operation of the 1000BASE-X Auto-Negotiation function that occurs between the Ethernet MAC and its link partner, which is the Ethernet device connected at the far end of the PHY link (see "1000BASE-X Auto-Negotiation," page 174).

Registers 16 and 17 are vendor-specific registers, defined by Xilinx.

Table 5-2: **Configuration Registers for 1000BASE-X PCS/PMA**

Register Address (REGAD)	Register Name
0	"Control Register (Register 0)"
1	"Status Register (Register 1)"
2,3	"PHY Identifier (Registers 2 and 3)"
4	"Auto-Negotiation Advertisement Register (Register 4)"
5	"Auto-Negotiation Link Partner Ability Base Register (Register 5)"
6	"Auto-Negotiation Expansion Register (Register 6)"
7	"Auto-Negotiation Next Page Transmit Register (Register 7)"
8	"Auto-Negotiation Next Page Receive Register (Register 8)"
15	"Extended Status Register (Register 15)"
16	"Vendor-Specific Register: Auto-Negotiation Interrupt Control Register (Register 16)"
17	"Vendor-Specific Register: Loopback Control Register (Register 17)"

Table 5-3: **Control Register (Register 0)**

Bit(s)	Name	Description	Type	Default Value
0.15	Reset	1 = PCS/PMA reset. 0 = Normal operation.	Read/Write Self Clearing	EMAC#_PHYRESET "Physical Interface Attributes"
0.14	Loopback	1 = Enable loopback mode. 0 = Disable loopback mode.	Read/Write	EMAC#_PHYLOOPBACKMSB "Physical Interface Attributes"
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	EMAC#_PHYINITAUTONEG_ENABLE "Physical Interface Attributes"
0.11	Power Down	1 = Power down. 0 = Normal operation. When set to 1, the RocketIO™ serial transceiver is placed in a Low power state. This bit requires a reset (see bit 0.15) to clear.	Read/ Write	EMAC#_PHYPOWERDOWN "Physical Interface Attributes"

Table 5-3: **Control Register (Register 0) *(Cont'd)***

Bit(s)	Name	Description	Type	Default Value
0.10	Isolate	1 = Electrically isolate the PHY from GMII. 0 = Normal operation.	Read/Write	EMAC#_PHYISOLATE "Physical Interface Attributes"
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	Unidirectional Enable	Enable transmit regardless of whether a valid link has been established.	Read/Write	EMAC#_UNIDIRECTION_ENABLE "Physical Interface Attributes"
0.4:0	Reserved	Always returns zeros, writes ignored.	Returns 0s	00000

Table 5-4: **Status Register (Register 1)**

Bit(s)	Name	Description	Type	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
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

Table 5-4: **Status Register (Register 1)** ***(Cont'd)***

Bit(s)	Name	Description	Type	Default Value
1.7	Unidirectional Ability	Always returns a 1, writes ignored.	Returns 1	1
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 = Link is up 0 = Link is down (or has been down) Latches 0 if Link Status goes down. Clears to current Link Status on read. See the following Link Status section for further details.	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

Link Status

When Link Status is High, the link is valid and has remained valid since this register was last read. Synchronization of the link has been obtained and Auto Negotiation (if enabled) has completed.

When Low, either:

- A valid link has not been established. Link synchronization has failed or Auto Negotiation (if enabled) has failed to complete.
- Link synchronization was lost at some point because this register was previously read. However, the current link status can be valid. Read this register a second time to get confirmation of the current link status.

Regardless of whether Auto Negotiation is enabled or disabled, there can be some delay to the deassertion of Link Status 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 specification (the an_sync_status variable).

Table 5-5: **PHY Identifier (Registers 2 and 3)**

Bit(s)	Name	Description	Type	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
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 5-6: **Auto-Negotiation Advertisement Register (Register 4)**

Bit(s)	Name	Description	Type	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 5-7: **Auto-Negotiation Link Partner Ability Base Register (Register 5)**

Bit(s)	Name	Description	Type	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

Table 5-7: **Auto-Negotiation Link Partner Ability Base Register (Register 5) *(Cont'd)***

Bit(s)	Name	Description	Type	Default Value
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 5-8: **Auto-Negotiation Expansion Register (Register 6)**

Bit(s)	Name	Description	Type	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 5-9: **Auto-Negotiation Next Page Transmit Register (Register 7)**

Bit(s)	Name	Description	Type	Default Value
7.15	Next Page	1 = Additional next page(s) will 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 5-10: **Auto-Negotiation Next Page Receive Register (Register 8)**

Bit(s)	Name	Description	Type	Default Value
8.15	Next Page	1 = Additional Next Page(s) will 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

Table 5-10: **Auto-Negotiation Next Page Receive Register (Register 8) *(Cont'd)***

Bit(s)	Name	Description	Type	Default Value
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 5-11: **Extended Status Register (Register 15)**

Bit(s)	Name	Description	Type	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
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 5-12: **Vendor-Specific Register: Auto-Negotiation Interrupt Control Register (Register 16)**

Bit(s)	Name	Description	Type	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 set upon the completion of an auto-negotiation cycle; it is cleared only by writing 0 to this bit. If the interrupt is disabled, this bit is reset 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

Table 5-13: **Vendor-Specific Register: Loopback Control Register (Register 17)**

Bit(s)	Name	Description	Type	Default Value
17.15:1	Reserved	Always returns 0s.	Returns 0s	000000000000000
17.0	Loopback Position	0 = Loopback (when enabled) occurs in the Ethernet MAC directly before the interface to the RocketIO serial transceiver. 1 = Loopback (when enabled) occurs in the RocketIO serial transceiver. Note: Loopback is enabled or disabled using 0.14 (see "Control Register (Register 0)").	Read/Write	EMAC#_GTLOOPBACK "Physical Interface Attributes"

SGMII Management Registers

Figure 5-8 shows the PCS/PMA sublayer block in an SGMII implementation. The PCS/PMA sublayer of the Ethernet MAC, when performing the SGMII standard, contains PCS Management registers as listed in Table 5-14. These registers are described in further detail in Table 5-15 through Table 5-25.

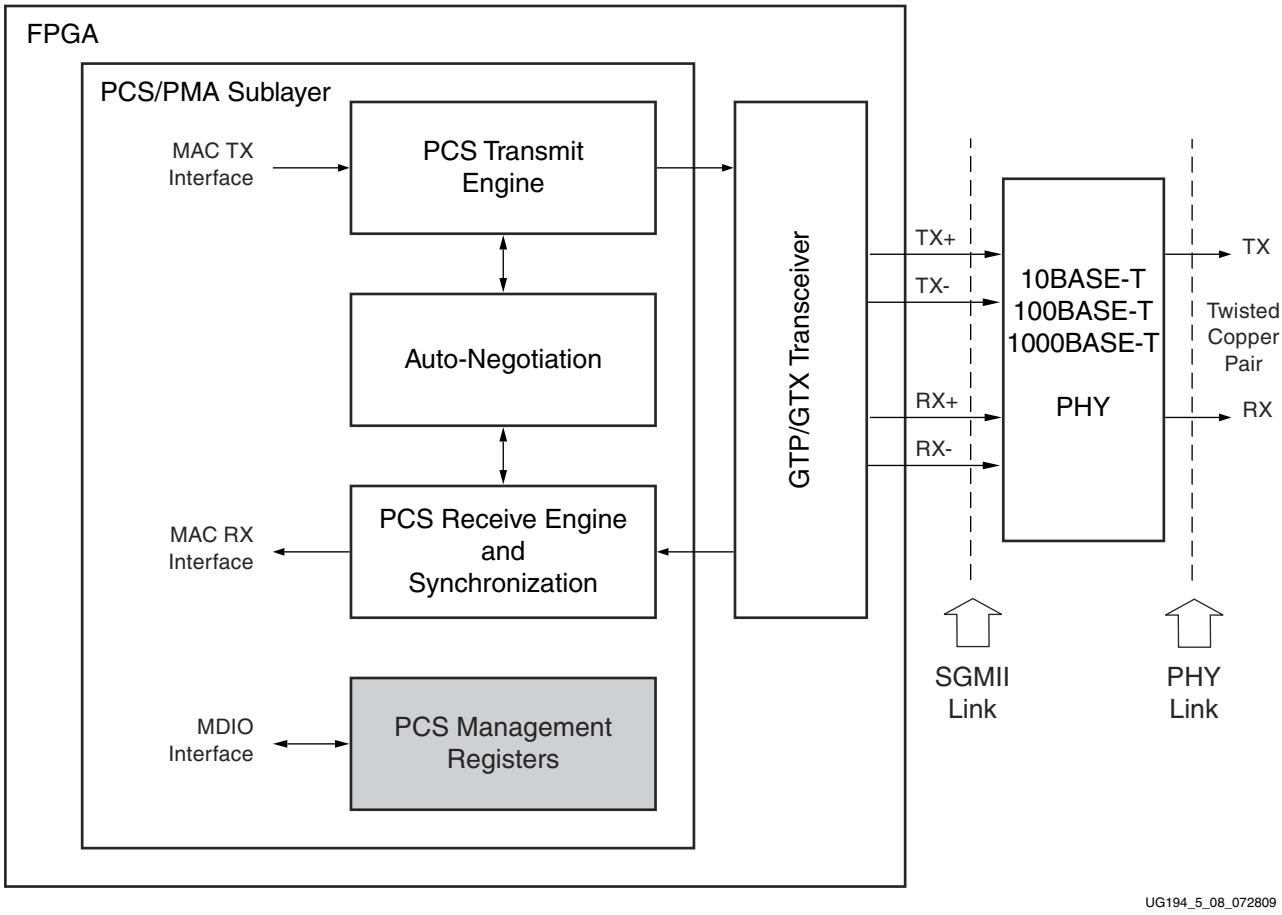


Figure 5-8: **SGMII Logic and Physical Connections**

These registers contain information relating to the operation of the system, including the status of both the SGMII link and the physical Ethernet link (PHY link). Refer to Figure 5-8.

Additionally, these registers are directly involved in the operation of the SGMII Auto-Negotiation function which occurs between the Ethernet MAC and the external PHY device, illustrated as a Tri-speed BASE-T PHY in Figure 5-8. (see "SGMII Auto-Negotiation," page 196).

Table 5-14: **SGMII Configuration Registers for 1000BASE-X PCS/PMA**

Register Address (REGAD)	Register Name
0	"SGMII Control Register (Register 0)"
1	"SGMII Status Register (Register 1)"

Table 5-14: **SGMII Configuration Registers for 1000BASE-X PCS/PMA *(Cont'd)***

Register Address (REGAD)	Register Name
2, 3	"PHY Identifier (Registers 2 and 3)"
4	"SGMII Auto-Negotiation Advertisement Register (Register 4)"
5	"SGMII Auto-Negotiation Link Partner Ability Base Register (Register 5)"
6	"SGMII Auto-Negotiation Expansion Register (Register 6)"
7	"SGMII Auto-Negotiation Next Page Transmit Register (Register 7)"
8	"SGMII Auto-Negotiation Next Page Receive Register (Register 8)"
15	"SGMII Extended Status Register (Register 15)"
16	"SGMII Vendor-Specific Register: Auto-Negotiation Interrupt Control Register (Register 16)"
17	"SGMII Vendor Specific Register: Loopback Control Register (Register 17)"

Table 5-15: **SGMII Control Register (Register 0)**

Bit(s)	Name	Description	Type	Default Value
0.15	Reset	1 = PCS/PMA reset. 0 = Normal operation.	Read/Write Self Clearing	EMAC#_PHYRESET "Physical Interface Attributes"
0.14	Loopback	1 = Enable loopback mode. 0 = Disable loopback mode.	Read/Write	EMAC#_PHYLOOPBACKMSB "Physical Interface Attributes"
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 SGMII Auto-Negotiation process. 0 = Disable SGMII Auto-Negotiation process.	Read/Write	EMAC#_PHYINITAUTONEG_ENABLE "Physical Interface Attributes"
0.11	Power Down	1 = Power down. 0 = Normal operation. When set to 1, the RocketIO serial transceiver is placed in a Low power state. This bit requires a reset (see bit 0.15) to clear.	Read/ Write	EMAC#_PHYPOWERDOWN "Physical Interface Attributes"
0.10	Isolate	1 = Isolate the SGMII logic from GMII. 0 = Normal operation.	Read/Write	EMAC#_PHYISOLATE "Physical Interface Attributes"
0.9	Restart Auto-Negotiation	1 = Restart Auto-Negotiation process across the SGMII link. 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

Table 5-15: **SGMII Control Register (Register 0) *(Cont'd)***

Bit(s)	Name	Description	Type	Default Value
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	Unidirectional Enable	Enable transmit regardless of whether a valid link has been established	Read/Write	EMAC#_UNIDIRECTION_ENABLE "Physical Interface Attributes"
0.4:0	Reserved	Always returns zeros, writes ignored.	Returns 0s	00000

Table 5-16: **SGMII Status Register (Register 1)**

Bit(s)	Name	Description	Type	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
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	Unidirectional Ability	Always returns a 1, writes ignored.	Returns 1	1
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

Table 5-16: **SGMII Status Register (Register 1) *(Cont'd)***

Bit(s)	Name	Description	Type	Default Value
1.2	SGMII Link Status	The link status of the Ethernet MAC with its external PHY across the SGMII link (see Figure 5-8). 1 = SGMII link is up. 0 = SGMII link is down.	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

Table 5-17: **PHY Identifier (Registers 2 and 3)**

Bit(s)	Name	Description	Type	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
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 5-18: **SGMII Auto-Negotiation Advertisement Register (Register 4)**

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

Table 5-19: **SGMII Auto-Negotiation Link Partner Ability Base Register (Register 5)**

Bit(s)	Name	Description	Type	Default Value
5.15	PHY Link Status	This refers to the link status of the external PHY device with its link partner across the PHY link (see Figure 5-8). 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	The resolved duplex mode that the external PHY device has auto-negotiated with its link partner across the PHY link (see Figure 5-8). 1 = Full Duplex. 0 = Half Duplex.	Read only	00

Table 5-19: **SGMII Auto-Negotiation Link Partner Ability Base Register (Register 5) *(Cont'd)***

Bit(s)	Name	Description	Type	Default Value
5.11:10	Speed	The resolved operating speed that the external PHY device has auto-negotiated with its link partner across the PHY link (see Figure 5-8). 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

Table 5-20: **SGMII Auto-Negotiation Expansion Register (Register 6)**

Bit(s)	Name	Description	Type	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 5-21: **SGMII Auto-Negotiation Next Page Transmit Register (Register 7)**

Bit(s)	Name	Description	Type	Default Value
7.15	Next Page	1 = Additional next page(s) will 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 5-22: **SGMII Auto-Negotiation Next Page Receive Register (Register 8)**

Bit(s)	Name	Description	Type	Default Value
8.15	Next Page	1 = Additional Next Page(s) will 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

Table 5-22: **SGMII Auto-Negotiation Next Page Receive Register (Register 8) *(Cont'd)***

Bit(s)	Name	Description	Type	Default Value
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 5-23: **SGMII Extended Status Register (Register 15)**

Bit(s)	Name	Description	Type	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
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 5-24: **SGMII Vendor-Specific Register: Auto-Negotiation Interrupt Control Register (Register 16)**

Bit(s)	Name	Description	Type	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 set upon the completion of an auto-negotiation cycle. Writing 0 to this bit is the only way to clear it. If the interrupt is disabled, this bit is cleared 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

Table 5-25: **SGMII Vendor Specific Register: Loopback Control Register (Register 17)**

Bit(s)	Name	Description	Type	Default Value
17.15:1	Reserved	Always returns 0s.	Returns 0s	000000000000000
17.0	Loopback Position	0 = Loopback (when enabled) occurs in the Ethernet MAC directly before the interface to the RocketIO serial transceiver. 1 = Loopback (when enabled) occurs in the RocketIO serial transceiver. Note: Loopback is enabled or disabled using 0.14 (see "Control Register (Register 0)").	Read/Write	EMAC#_GTLOOPBACK "Physical Interface Attributes"

XILINX®

Chapter 6

Physical Interface

This chapter describes the physical interface options provided by the Ethernet MAC. This chapter contains the following sections:

- "Introduction to the Physical Interfaces"
- "Media Independent Interface (MII)"
- "Gigabit Media Independent Interface (GMII)"
- "Reduced Gigabit Media Independent Interface (RGMII)"
- "1000BASE-X PCS/PMA"
- "Serial Gigabit Media Independent Interface (SGMII)"

For each physical interface, there can be more than one possible clocking scheme. The following sections describe the different optimized clocking schemes for the individual Ethernet MAC configurations.

Ethernet MAC wrappers and example designs are provided for each of the different physical interfaces via the CORE Generator™ tool. The generated designs exactly match the descriptions in this chapter and are available in both VHDL and Verilog. By using the CORE Generator tool, the time required to instantiate the Ethernet MAC into a usable design is greatly reduced. See "Accessing the Ethernet MAC from the CORE Generator Tool," page 25.

Introduction to the Physical Interfaces

Figure 6-1 illustrates the position of the Ethernet MAC sublayers and physical interface options within the OSI model.

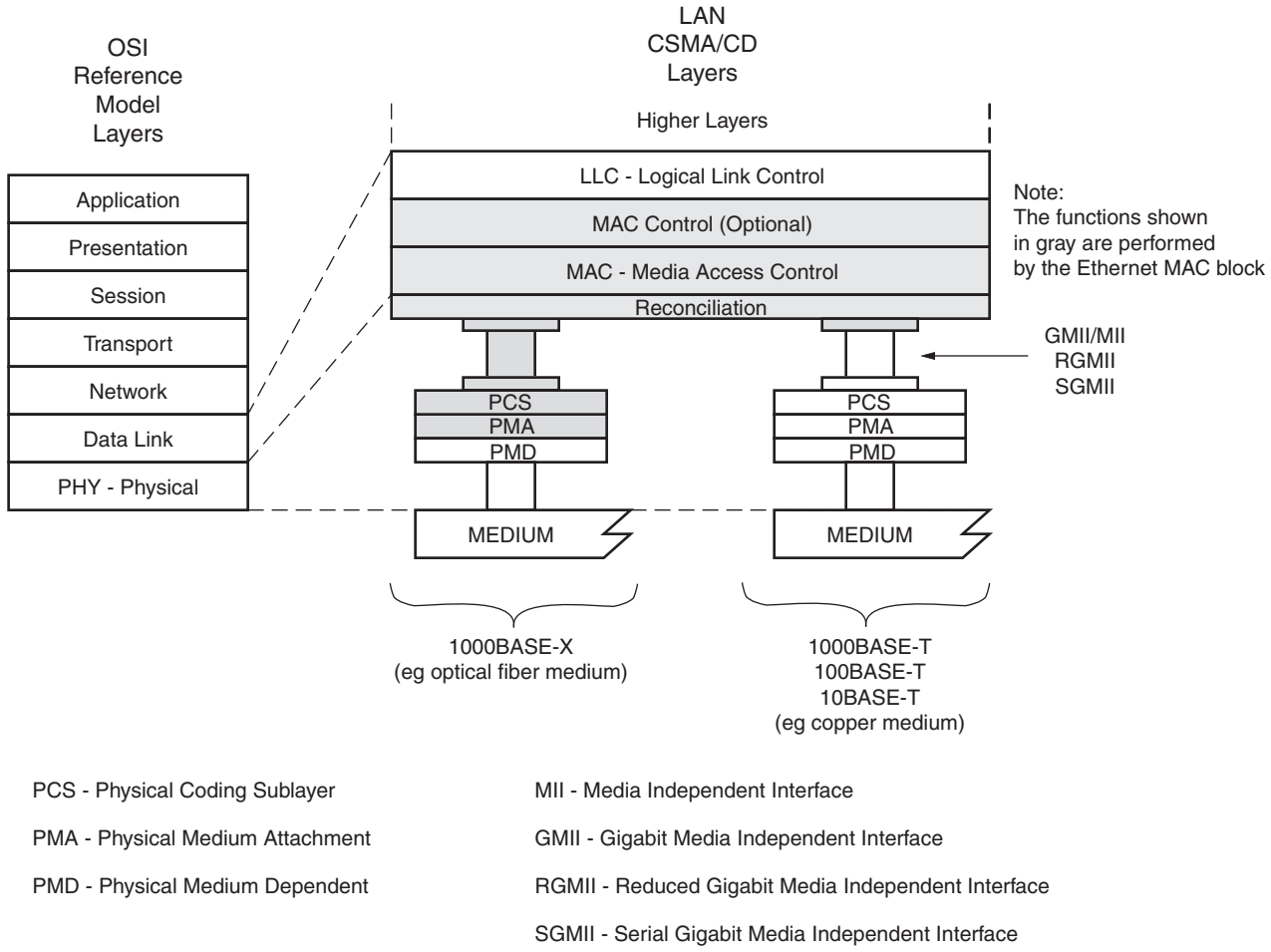


Figure 6-1: **MAC Sublayer Partitioning, Relationship to the ISO/IEC OSI Reference Model**

The Ethernet MAC sublayer itself is independent of, and can connect to, any type of physical layer device. In Figure 6-1, two alternative physical sublayer standards are shown:

- **BASE-T** PHYs provide a link between the MAC and copper mediums. This functionality is not offered within the Ethernet MAC. However, external BASE-T PHY devices are readily available on the market. These can connect to the Ethernet MAC using GMII/MII, RGMII, or SGMII interfaces.
- **BASE-X** PHYs provide a link between the MAC and (usually) fiber optic mediums. The Ethernet MAC itself can support the 1 Gb/s BASE-X standard. 1000BASE-X can be offered by connecting the Ethernet MAC to a RocketIO™ serial transceiver.

The interface type and 1000BASE-X sublayer functionalities are summarized in the following subsections.

GMII / MII

The Media Independent Interface (MII), defined in IEEE 802.3, clause 22 is a parallel interface that connects a 10-Mb/s and/or 100-Mb/s capable MAC to the physical sublayers.

The Gigabit Media Independent Interface (GMII), defined in IEEE 802.3, clause 35 is an extension of the MII used to connect a 1-Gb/s capable MAC to the physical sublayers.

MII can be considered a subset of GMII, and as a result, GMII/MII together can carry Ethernet traffic at 10 Mb/s, 100 Mb/s, and 1 Gb/s.

RGMII

The Reduced Gigabit Media Independent Interface (RGMII) is an alternative to the GMII/MII. RGMII achieves a 50% reduction in the pin count compared with GMII, and therefore, is favored over GMII/MII by PCB designers. This configuration is achieved with the use of double-data-rate (DDR) flip-flops.

RGMII can carry Ethernet traffic at 10 Mb/s, 100 Mb/s, and 1 Gb/s.

SGMII

The Serial-GMII (SGMII) is an alternative interface to the GMII/MII that converts the parallel interface of the GMII into a serial format. It radically reduces the I/O count and is, therefore, often favored by PCB designers. This configuration is achieved by connecting the Ethernet MAC to a RocketIO™ serial transceiver.

SGMII can carry Ethernet traffic at 10 Mb/s, 100 Mb/s, and 1 Gb/s.

1000BASE-X Sublayers

PCS/PMA

The Physical Coding Sublayer (PCS) for 1000BASE-X operation is defined in IEEE 802.3, clauses 36 and 37, and performs the following:

- Encoding (and decoding) of GMII data octets to form a sequence of ordered sets
- 8B/10B encoding (and decoding) of the sequence ordered sets
- 1000BASE-X auto-negotiation for information exchange with the link partner

The Physical Medium Attachment (PMA) for 1000BASE-X operation is defined in IEEE 802.3, clause 36 and performs the following:

- Serialization (and deserialization) of code-groups for transmission (and reception) on the underlying serial PMD
- Recovery of clock from the 8B/10B-coded data supplied by the PMD

1000BASE-X PCS/PMA functionality is provided by connecting the Ethernet MAC to a RocketIO serial transceiver.

PMD

The Physical Medium Dependent (PMD) sublayer is defined in IEEE 802.3, clause 38 for 1000BASE-LX and 1000BASE-SX (long and short wavelength laser). This type of PMD is provided by the external GBIC or SFP optical transceiver, which should be connected directly to the ports of the RocketIO serial transceiver.

Media Independent Interface (MII)

The Media Independent Interface (MII), defined in IEEE 802.3, clause 22, is a parallel interface that connects a 10-Mb/s and/or 100-Mb/s capable MAC to the physical sublayers.

Figure 6-2 shows the Ethernet MAC configured for MII. The physical signals of the Ethernet MAC are connected through IOBs to the external interface. The "logic cloud" indicates that alternative implementations are possible.

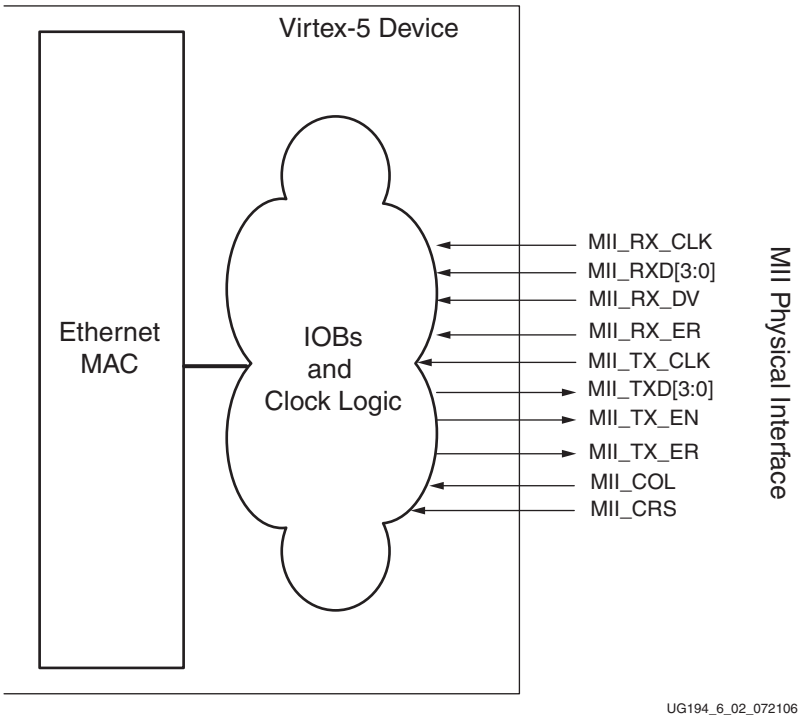


Figure 6-2: **Ethernet MAC Configured in MII Mode**

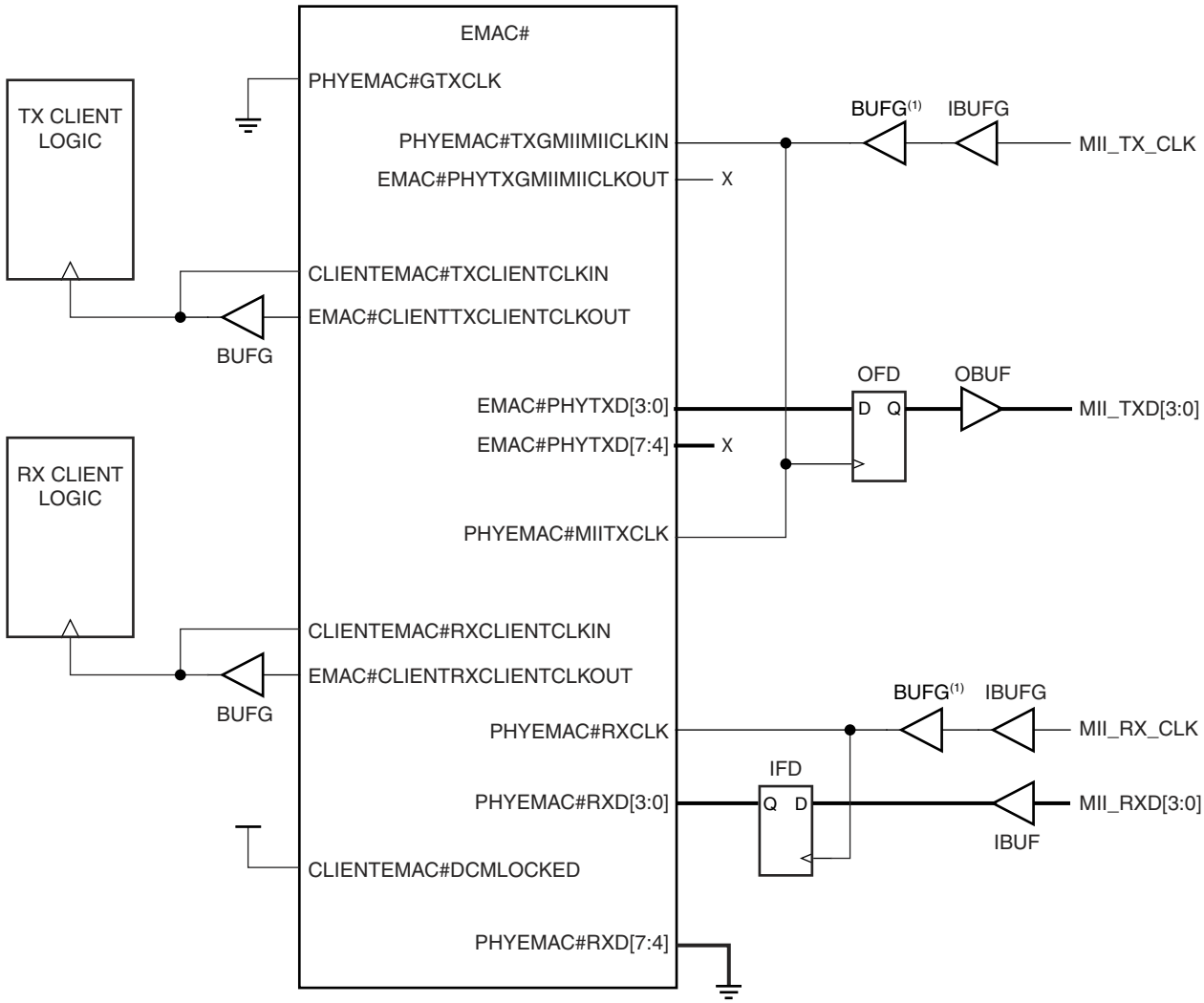
There are two alternative implementations available that use different clocking schemes: the standard clocking scheme without clock enables or an alternative clock enable scheme that saves on global clock resources. "Ethernet MAC Clocks," page 205 introduces these two clocking models, which are described in detail in the following sections:

- "MII Standard Clock Management"
- "MII Clock Management using Clock Enables"

If the CORE Generator tool is used, the wrapper files for the Ethernet MAC that are created will contain the logic described in these sections. By using the CORE Generator tool, the time required to instantiate the Ethernet MAC into a usable design is greatly reduced. See "Accessing the Ethernet MAC from the CORE Generator Tool," page 25.

MII Standard Clock Management

Figure 6-3 shows the standard clock management scheme when the MII interface is selected. 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. `EMAC#CLIENTTXCLIENTCLKOUT` connects to `EMAC#CLIENTTXCLIENTCLKIN` through a BUFG. Its frequency is either 12.5 MHz or 1.25 MHz, depending on the operating speed of the Ethernet MAC. `MII_TX_CLK_#` connects to `PHYEMAC#TXGMIIMIICLKIN` and the MII_TXD registers through a BUFG. Its frequency is either 12.5 MHz or 1.25 MHz, depending on the operating speed of the Ethernet MAC. The RX client clocking is similar.



Notes:
1) A regional buffer (BUFR) can replace this BUFG.
In addition, the clock input of IFD can be driven by a BUFIO.
Refer to UG190, *Virtex-5 FPGA User Guide* for BUFR usage guidelines.

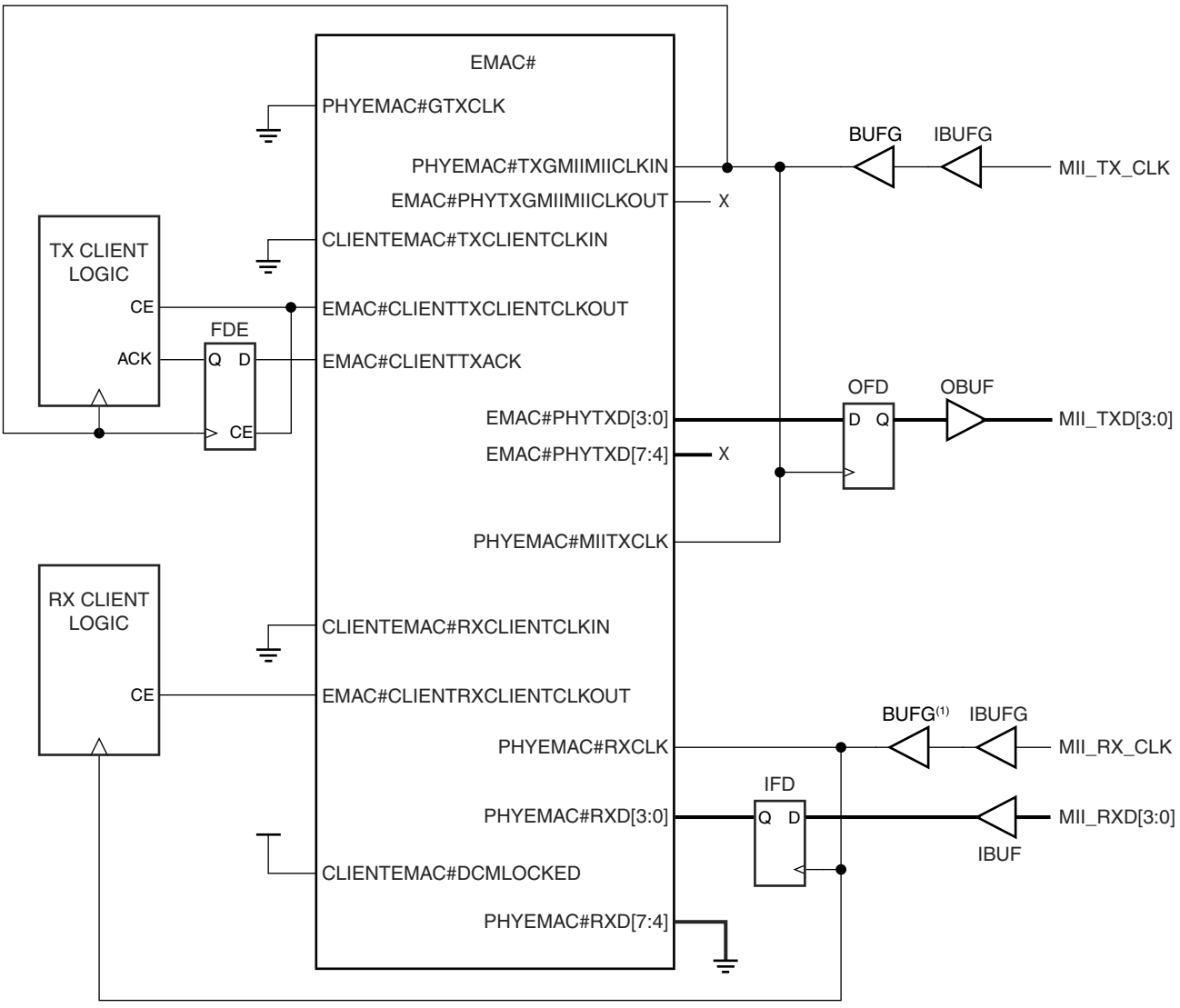
UG194_6_03_081409

Figure 6-3: **MII Clock Management**

The `CLIENTEMAC#DCMLOCKED` port must be tied High.

MII Clock Management using Clock Enables

It is possible to use only two BUFGs by setting the EMAC#_USECLKEN attribute; however, the client and MII logic must be constrained to run at 125 MHz. Figure 6-4 shows the MII clock management scheme when operating in this mode.



Notes:
1) A regional buffer (BUFR) can replace this BUFG.
In addition, the clock input of IFD can be driven by a BUFIO.
Refer to UG190, *Virtex-5 FPGA User Guide* for BUFR usage guidelines.

UG194_6_04_091409

Figure 6-4: **MII Clock Management with Clock Enable**

Using the example in Figure 6-4, all logic is clocked on the MII interface clocks. The frequencies of these clocks are 25 MHz at 100 Mb/s and 2.5 MHz at 10 Mb/s, or twice as fast as the client clock requires. To produce the correct clock frequency at the client, the client logic must be clock enabled to achieve the correct data rate. The clock enables for the client logic are provided by the `EMAC#CLIENTTXCLIENTCLKOUT` and `EMAC#CLIENTRXCLIENTCLKOUT` outputs.

Due to the timing relationship between the MII transmit clock and the internal client clock signals, the TX_ACK signal (CLIENTEMAC#TXACK) is registered at the output of the Ethernet MAC.

Gigabit Media Independent Interface (GMII)

The Media Independent Interface (MII), defined in IEEE 802.3, clause 22, is a parallel interface that connects a 10-Mb/s and/or 100-Mb/s capable MAC to the physical sublayers. The Gigabit Media Independent Interface (GMII), defined in IEEE 802.3, clause 35, is an extension of the MII used to connect a 1-Gb/s capable MAC to the physical sublayers. MII can be considered a subset of GMII, and as a result, GMII/MII can carry Ethernet traffic at 10 Mb/s, 100 Mb/s, and 1 Gb/s.

Figure 6-5 shows the Ethernet MAC configured for GMII. The physical signals of the Ethernet MAC are connected through IOBs to the external interface. The "logic cloud" indicates that alternative implementations are possible.

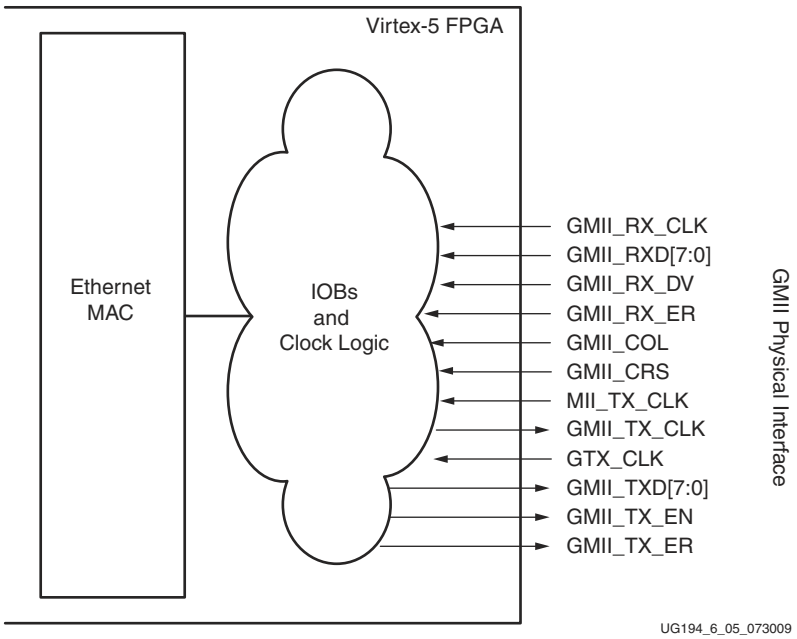


Figure 6-5: **Ethernet MAC Configured in GMII Mode**

There are several implementations available using different clocking schemes. "Ethernet MAC Clocks," page 205 introduces these clocking models. They are described in detail in the following sections:

- "GMII Clock Management for 1 Gb/s Only"

  At 1 Gb/s speeds only, client interface and physical interface clocks always run at the same frequency (125 MHz). This enables efficient clock logic implementations.

- "GMII Standard Clock Management for Tri-Speed Operation"

  The standard clocking scheme for tri-speed operation.

- "GMII Clock Management for Tri-Speed Operation Using Byte PHY"

  An advanced clocking scheme for tri-speed operation, which saves on global clock resources.

- "GMII Clock Management for Tri-Speed Operation Using Clock Enables"

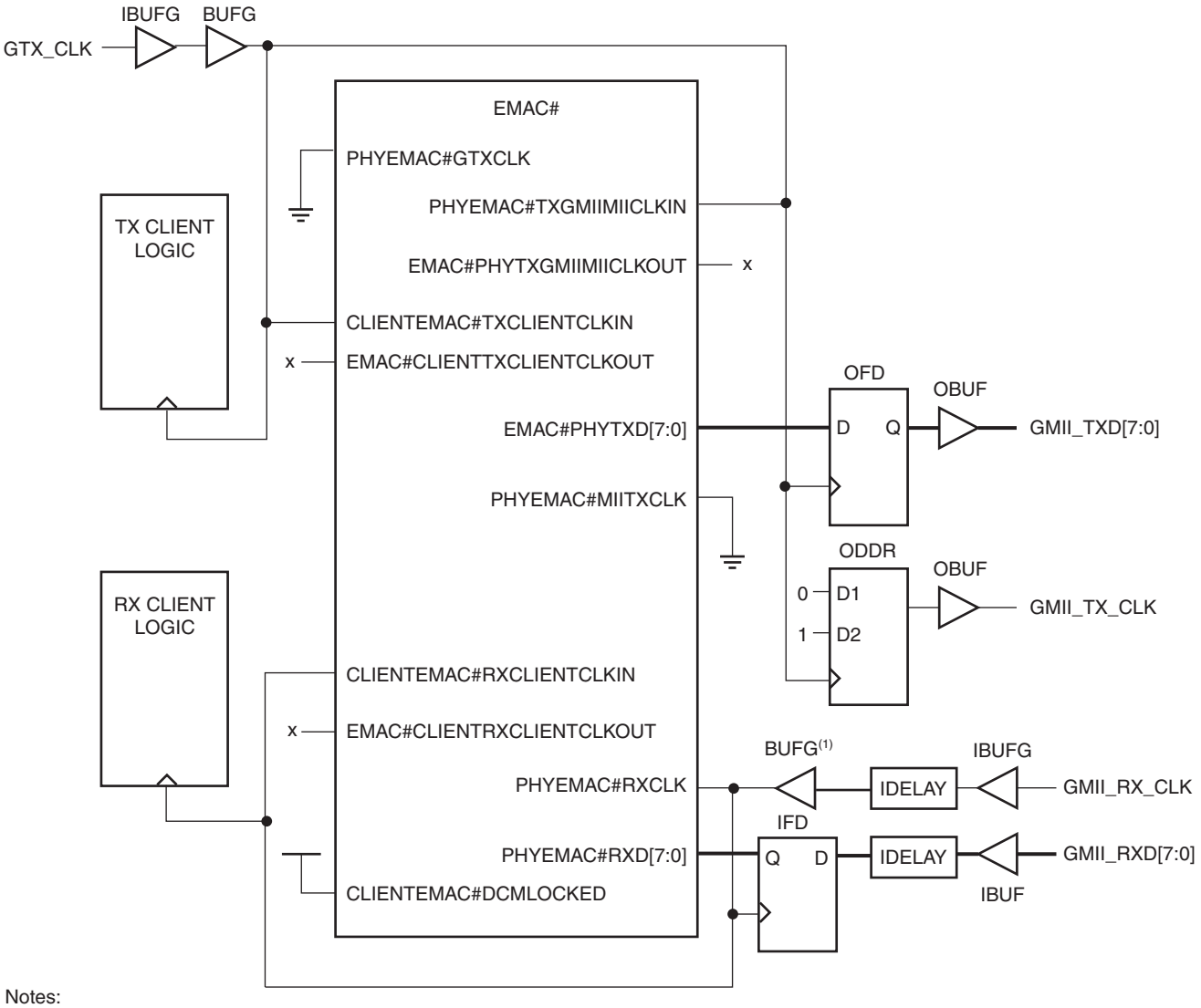
  An alternative advanced clocking scheme for tri-speed operation, which saves on global clock resources.

The advanced clocking schemes, Byte PHY and Clock Enables, both provide the same saving on global clocking resources and can be used interchangeably for GMII, based on user preference.

If the CORE Generator tool is used, the wrapper files for the Ethernet MAC that are created will contain the logic described in these sections. By using the CORE Generator tool, the time required to instantiate the Ethernet MAC into a usable design is greatly reduced. See "Accessing the Ethernet MAC from the CORE Generator Tool," page 25.

GMII Clock Management for 1 Gb/s Only

Figure 6-6 shows GMII clock management when using a single Ethernet MAC.



Notes:
1) A regional buffer (BUFR) can replace this BUFG.
In addition, the clock input of IFD can be driven by a BUFIO.
Refer to UG190, *Virtex-5 FPGA User Guide* for BUFR usage guidelines.

UG194_6_06_091409

Figure 6-6: **1 Gb/s GMII Clock Management**

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

The GTX_CLK input is routed onto the global clock network via an IBUFG and BUFG. The output of the BUFG connects to:

- GMII_TXD registers in the FPGA logic
- `PHYEMAC#TXGMIIMIICLKIN` input port
- `CLIENTEMAC#TXCLIENTCLKIN`
- TX client logic

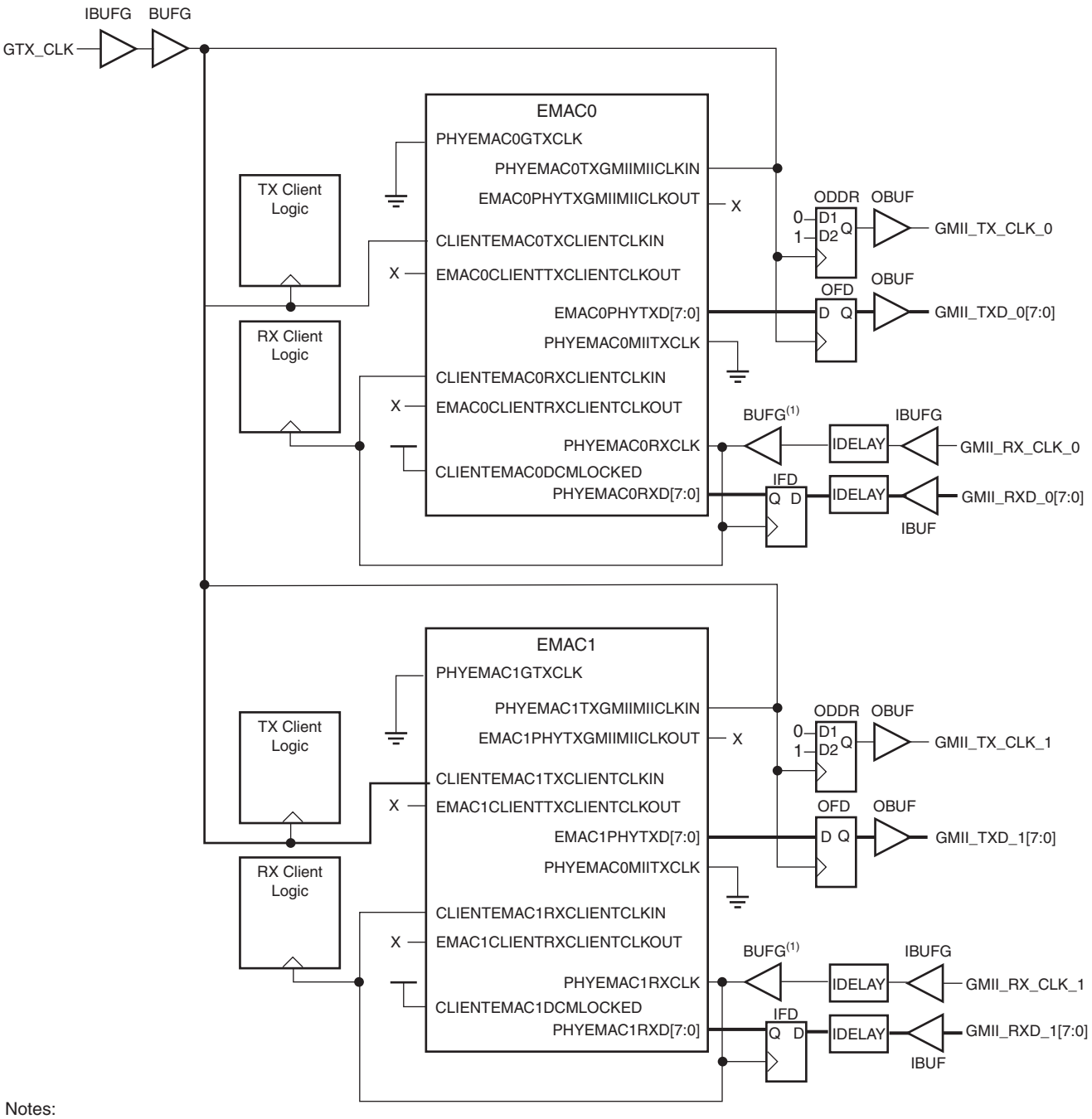
GMII_TX_CLK is forwarded along with the GMII data signals from the FPGA to the PHY. An IOB DDR output register is used; this is a predictable way to produce the clock because the clock-to-pad delay is the same as that for the GMII_TXD signals. This forwarded clock

is inverted with respect to PHYEMAC#TXGMIIMIICLKIN so that its rising edge occurs in the centre of the GMII_TXD data valid window. This produces the best possible setup and hold times for driving the external interface.

PHYEMAC#MIITXCLK is unused and must be tied to ground. The GMII_RX_CLK is generated from the PHY and connected to PHYEMAC#RXCLK through an IBUFG and BUFG. This is also used to clock all FPGA receiver logic, including the receiver client. Fixed-mode IDELAYs are used on the GMII_RX_CLK and GMII_RXD inputs to align the clock and data. These are set to sample a 2 ns setup, 0 ns hold window at the device pads

The CLIENTEMAC#DCMLOCKED port must be tied High.

Figure 6-7 shows the GMII clocking scheme with two Ethernet MACs enabled. It is similar to the single Ethernet MAC clocking scheme; however, the same GTX_CLK input signal is used for all (both Ethernet MACs) transmitter logic.

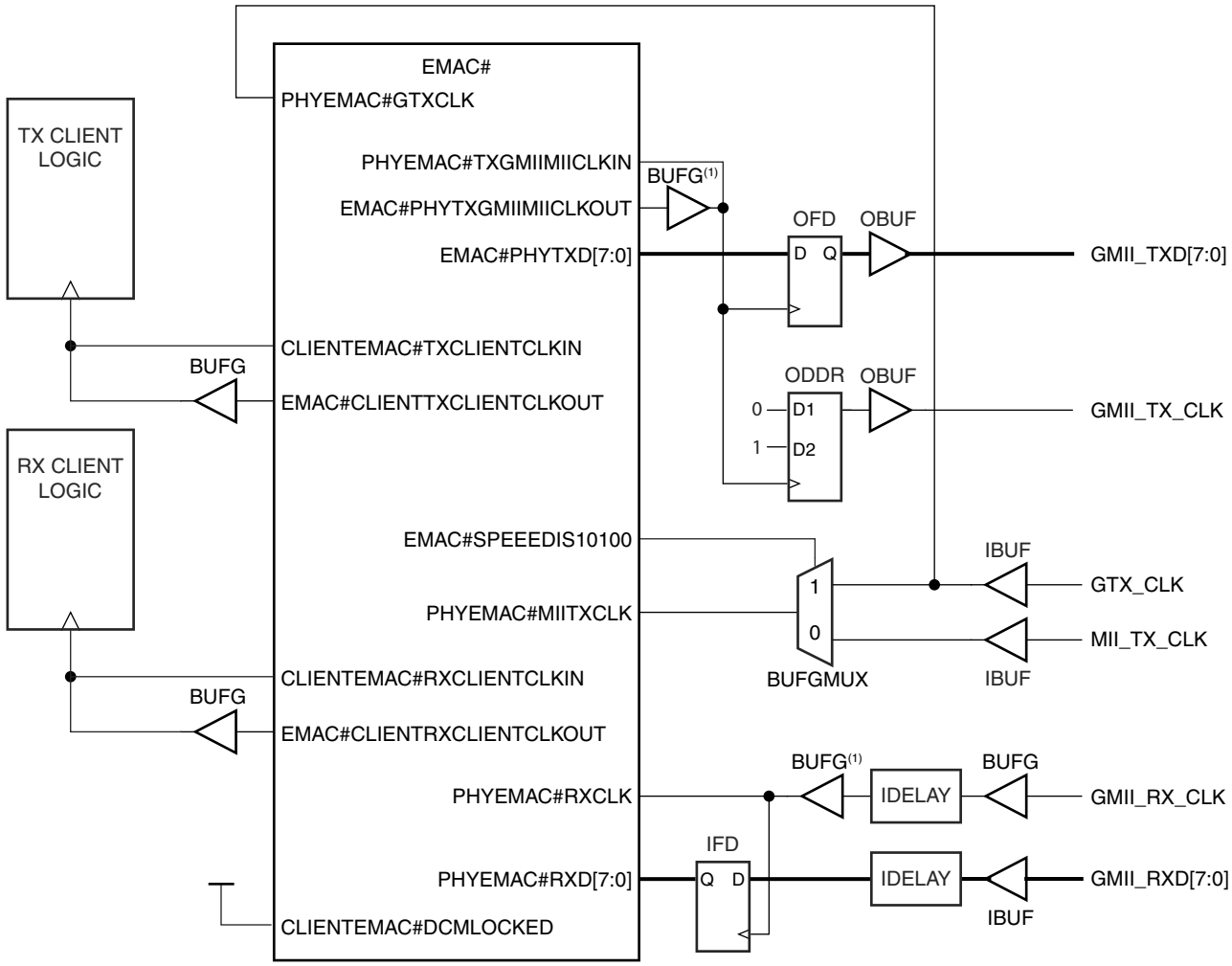


Notes:
1) A regional buffer (BUFR) can replace this BUFG.
In addition, the clock input of IFD can be driven by a BUFIO.
Refer to UG190, *Virtex-5 FPGA User Guide* for BUFR usage guidelines.

Figure 6-7: **1 Gb/s GMII Clock Management with Two Ethernet MACs Enabled**

GMII Standard Clock Management for Tri-Speed Operation

Figure 6-8 shows the clock management used with the tri-speed 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.



Notes:
1) A regional buffer (BUFR) can replace this BUFG.
In addition, the clock input of IFD can be driven by a BUFIO.
Refer to UG190, Virtex-5 FPGA User Guide for BUFR usage guidelines.

UG194_6_08_080309

Figure 6-8: **GMII Tri-Mode Operation Clock Management**

The `EMAC#PHYTXGMIIMIICLKOUT` output port connects to the GMII logic in the FPGA logic and the `PHYEMAC#TXGMIIMIICLKIN` input port through a BUFG.

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

The `MII_TX_CLK` signal generated from the PHY has a frequency of either 2.5 MHz or 25 MHz, depending on the operating speed of the Ethernet MAC. `MII_TX_CLK` connects

to PHYEMAC#MIITXCLK through an IBUF. The frequency of GMII_RX_CLK, generated from the PHY, is 2.5 MHz, 25 MHz, or 125 MHz, depending on the operating speed of the Ethernet MAC. Fixed-mode IDELAYs are used on the GMII_RX_CLK and GMII_RXD inputs to align the clock and data. These are set to sample a 2 ns setup, 0 ns hold window at the device pads.The CLIENTEMAC#DCMLOCKED port must be tied High.

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

GMII Clock Management for Tri-Speed Operation Using Byte PHY

Figure 6-9 shows an alternative clock management scheme for the tri-speed GMII interface. This clock management scheme is used when the EMAC#_BYTEPHY attribute is set to TRUE. In this scheme, the EMAC datapath is 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. DDR 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 6-8. 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.

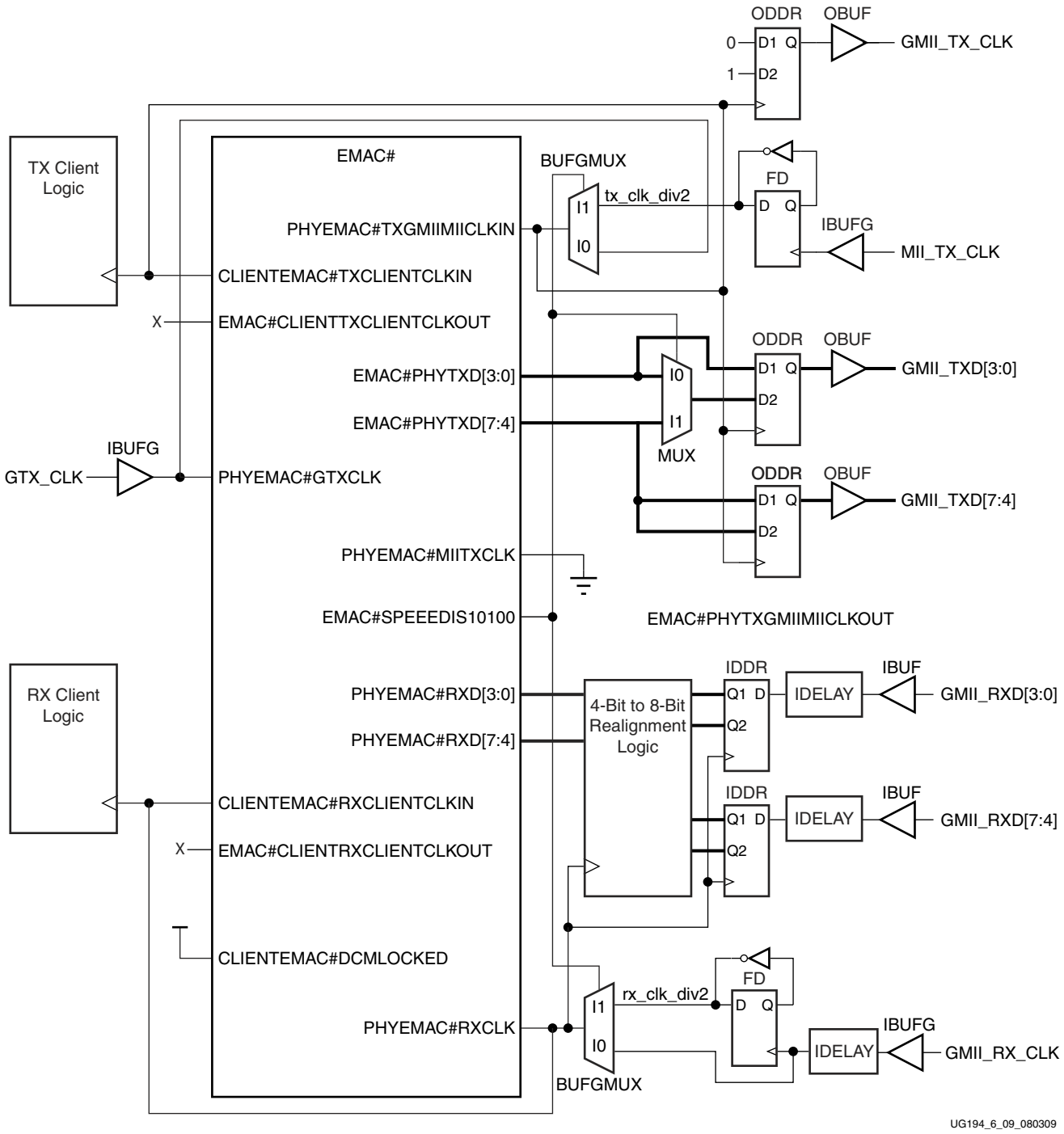


Figure 6-9: **Tri-Mode GMII Clock Management with Byte PHY Enabled**

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 PHYEMAC#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

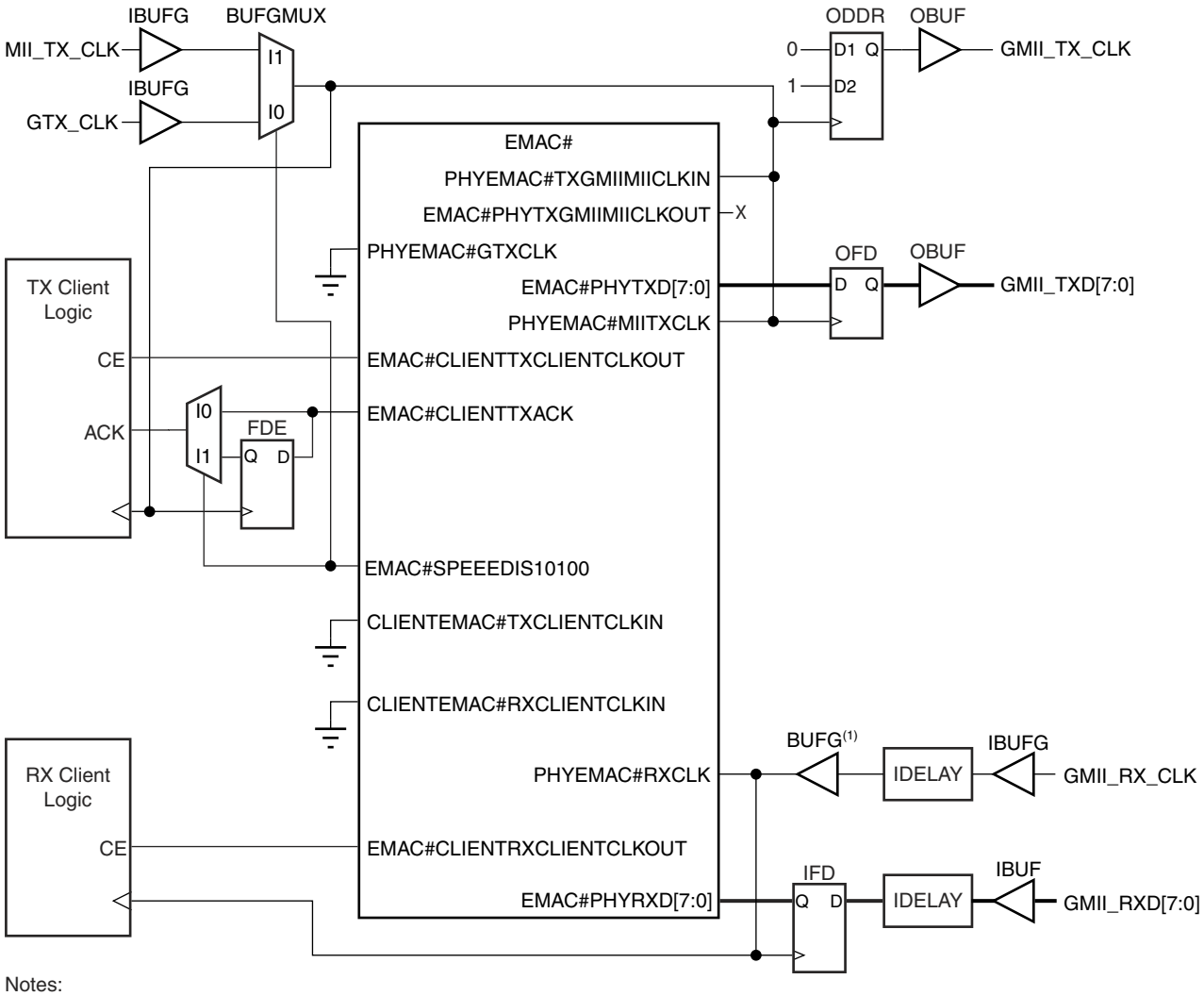
divided by two in frequency. The output of the BUFGMUX also clocks all the transmit client and GMII logic.

The receiver side clocking is similar. At 1 Gb/s, a BUFGMUX supplies the GMII_RX_CLK signal 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. Fixed-mode IDELAYs are used on the GMII_RX_CLK and GMII_RXD inputs to align the clock and data. These are set to sample a 2 ns setup, 0 ns hold window at the device pads.

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

GMII Clock Management for Tri-Speed Operation Using Clock Enables

Figure 6-10 shows the clock management scheme for tri-speed operation when the EMAC#_USECLKEN attribute is set. In this mode of operation, the transmitter signals are synchronous to PHYEMAC#TXGMIIMIICLKIN and the receiver signals are synchronous to PHYEMAC#RXCLK.



Notes:
1) A regional buffer (BUFR) can replace this BUFG.
In addition, the clock input of IFD can be driven by a BUFIO.
Refer to UG190, *Virtex-5 FPGA User Guide* for BUFR usage guidelines.

UG194_6_10_080409

Figure 6-10: **Tri-Mode GMII Clock Management with Clock Enable**

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 25 MHz and 2.5 MHz, respectively. To maintain the correct data rate at the client, interface clock enable signals are output on `EMAC#CLIENTTXCLIENTCLKOUT` for the transmitter logic and on `EMAC#CLIENTRXCLIENTCLKOUT` for the receiver logic. These signals are High during 1 Gb/s operation and toggle on each clock edge at slower speeds. These are used to enable the client interface logic.

Due to the timing relationship between the GMII transmit clock and the internal client clock signals, the TX_ACK signal (`CLIENTEMAC#TXACK`) is registered at the output of the Ethernet MAC when operating at speeds below 1 Gb/s. At 1 G b/s, the register is bypassed.

The GMII_TX_CLK is derived from the GTX_CLK input when the MAC is operating at 1 Gb/s and from the MII_TX_CLK input when the MAC is operating at a lower speed. It is

routed through an OBUF and then connected to the PHY. Its frequency changes between 125 MHz, 25 MHz, or 2.5 MHz depending on the speed setting of the Ethernet MAC.

This clocking scheme requires two fewer BUFGs than the default scheme detailed in Figure 6-8.

Reduced Gigabit Media Independent Interface (RGMII)

Reduced GMII (RGMII), defined by Hewlett-Packard, is an alternative to GMII. It reduces the number of pins required to connect the Ethernet MAC to the PHY from 24 to 12, when compared with GMII. RGMII achieves this 50% pin count reduction in the interface by using DDR flip-flops. For this reason, RGMII is preferred over GMII by PCB designers. RGMII can carry Ethernet traffic at 10 Mb/s, 100 Mb/s, and 1 Gb/s. For more information on RGMII, refer to the *Hewlett-Packard RGMII Specification, version 1.3 and 2.0*.

Figure 6-11 shows the Ethernet MAC configured for RGMII. The physical signals of the Ethernet MAC are connected through IOBs to the external interface. The "logic cloud" indicates that alternative implementations are possible.

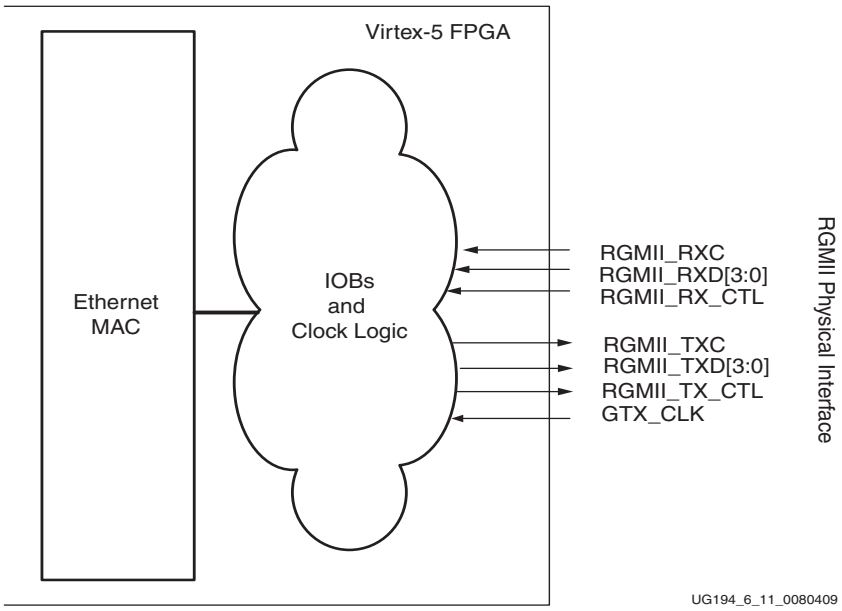


Figure 6-11: **Ethernet MAC Configured in RGMII Mode**

There are several implementations for both RGMII v1.3 and v2.0, which use alternative clocking schemes. "Ethernet MAC Clocks," page 205 introduces these clocking models. They are described in detail in the following sections:

- "RGMII Clock Management for 1 Gb/s Only"

  At 1 Gb/s speeds only, client interface and physical interface clocks always run at the same frequency (125 MHz). This enables efficient clock logic implementations.

- "RGMII Standard Clock Management for Tri-Speed Operation"

  The standard clocking scheme for tri-speed operation.

- "RGMII Clock Management for Tri-Speed Operation Using Clock Enables"

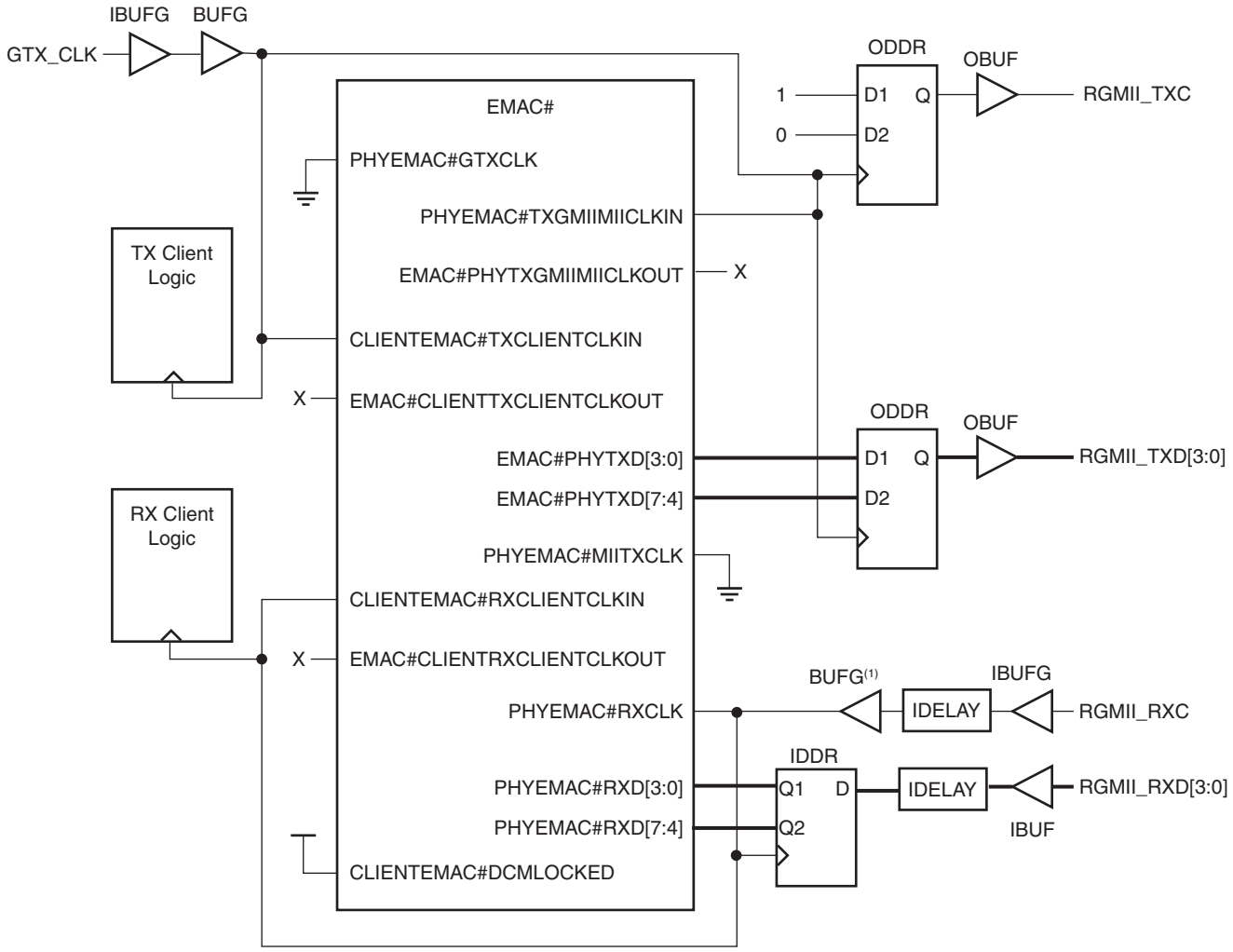
  An advanced clocking scheme for tri-speed operation that saves global clock resources.

If the CORE Generator tool is used, then the wrapper files for the Ethernet MAC that are created contain the logic described in these sections. By using the CORE Generator tool, the time required to instantiate the Ethernet MAC into a usable design is greatly reduced. See "Accessing the Ethernet MAC from the CORE Generator Tool," page 25.

RGMII Clock Management for 1 Gb/s Only

RGMII Version 1.3

Figure 6-12 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#PHYTXGMIIMIICLKOUT` output port drives all transmitter logic through a BUFG.



Notes:
1) A regional buffer (BUFR) can replace this BUFG.
In addition, the clock input of IDDR can be driven by a BUFIO.
Refer to UG190, Virtex-5 FPGA User Guide for BUFR usage guidelines.

UG194_6_12_080409

Figure 6-12: **1 Gb/s RGMII Hewlett Packard v1.3 Clock Management**

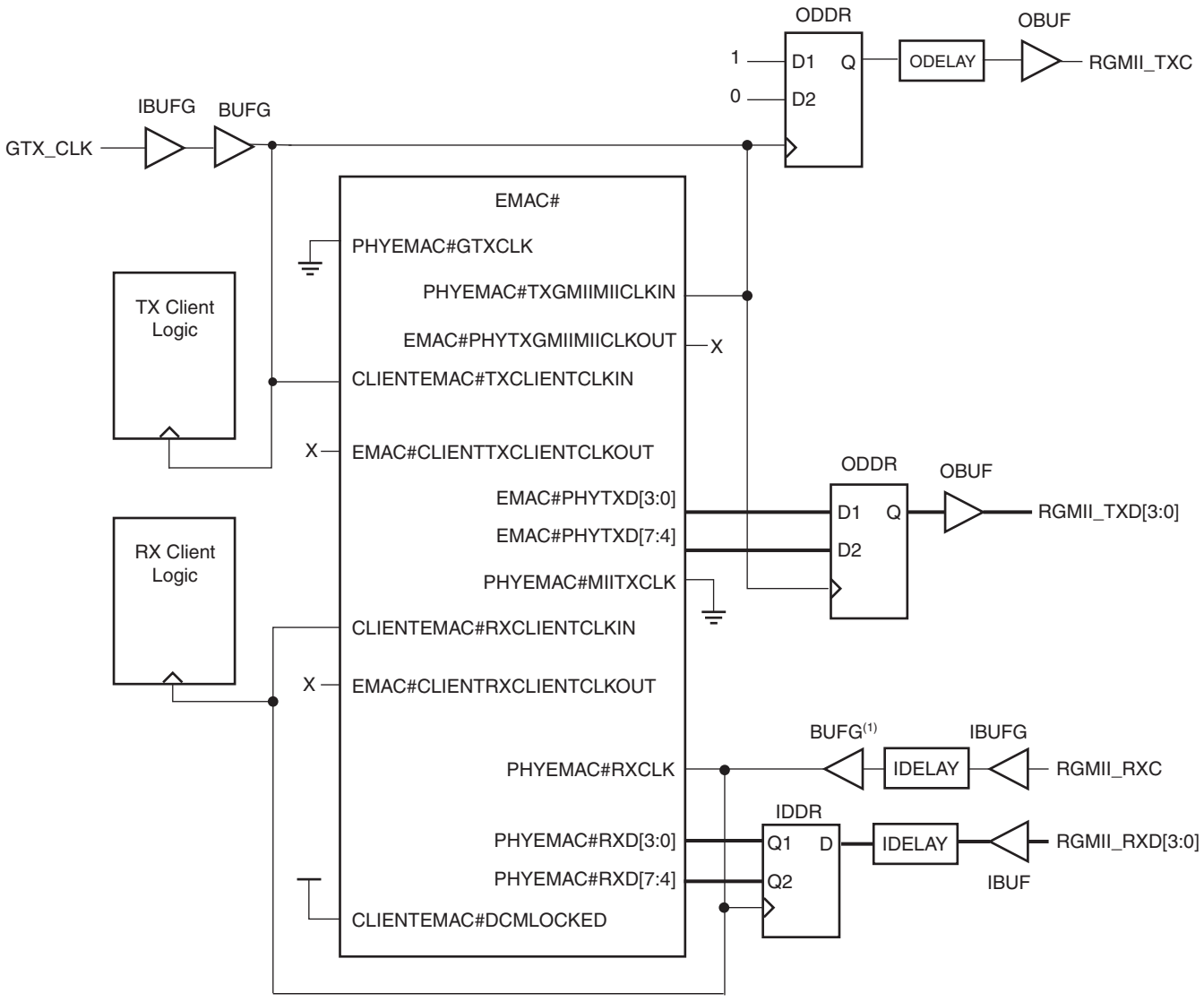
RGMII_TXC is derived from GTX_CLK by routing to an IOB DDR output register followed by an OBUF (which is then connected to the PHY). The use of the DDR register ensures that the forwarded clock is exactly in line with the RGMII transmitter data, as specified in the *Hewlett Packard RGMII Specification,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

RGMII_RXC is generated by the PHY and connected to PHYEMAC#RXCLK via an IBUFG and a BUFG. Fixed-mode IDELAYs are instantiated on the RGMII clock and data lines. These are set to sample a 1 ns setup, 1 ns hold window at the device pads. RGMII_RXC is used to derive the clocks for all receive logic.

The CLIENTEMAC#DCMLOCKED port must be tied High.

RGMII Version 2.0

Figure 6-13 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#PHYTXGMIIMIICLKOUT output port connects to a BUFG, which in turn drives the RGMII transmitter logic in the FPGA logic and the PHYEMAC#TXGMIIMIICLKIN input port.



Notes:
1) A regional buffer (BUFR) can replace this BUFG.
In addition, the clock input of IDDR can be driven by a BUFIO.
Refer to UG190, *Virtex-5 FPGA User Guide* for BUFR usage guidelines.

UG194_6_13_080409

Figure 6-13: **1 Gb/s RGMII Hewlett Packard v2.0 Clock Management**

`RGMII_TXC` is derived from the Ethernet MAC by routing GTX_CLK to an IOB DDR output register, followed by an ODELAY element and an OBUF (which is then connected to the PHY). The ODELAY element is used to generate 2 ns of skew required between `RGMII_TXC` and `RGMII_TXD` at the FPGA device pads. This delay is specified in the *Hewlett Packard RGMII Specification, v2.0* to provide setup and hold times on the external interface.

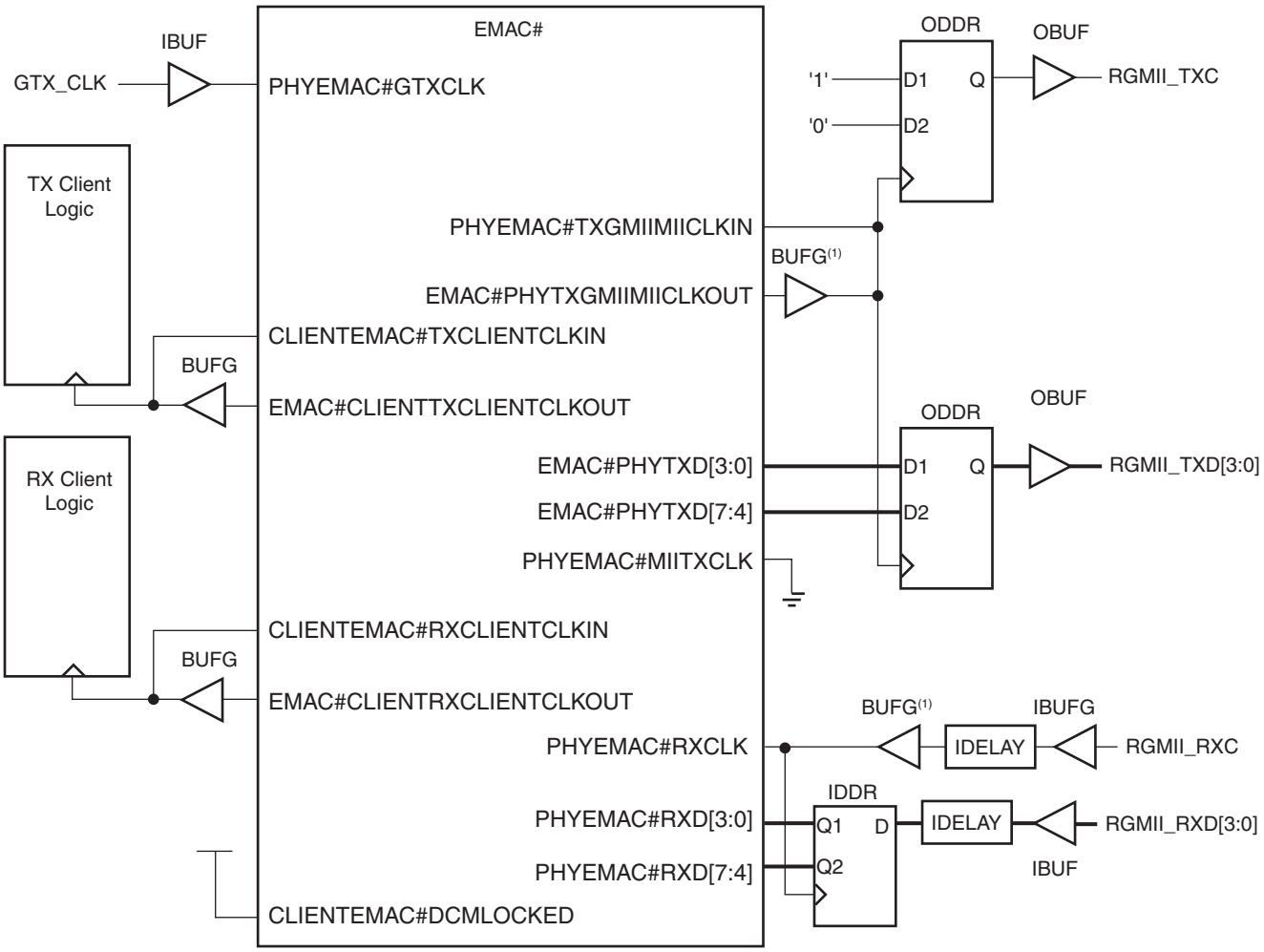
`RGMII_RXC` is generated by the PHY and connected to `PHYEMAC#RXCLK` via an IBUFG and a BUFG. Fixed-mode IDELAYs are instantiated on the RGMII clock and data lines. These are set to sample a 1 ns setup, 1 ns hold window at the device pads. `RGMII_RXC` drives all receive logic.

The `CLIENTEMAC#DCMLOCKED` port must be tied High.

RGMII Standard Clock Management for Tri-Speed Operation

RGMII Version 1.3

Figure 6-14 shows the tri-speed 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#PHYTXGMIIMIICLKOUT port generates the appropriate frequency derived from GTX_CLK and depending on the operating frequency of the link. It clocks directly the RGMII_TXD ODDR registers.



Notes:
1) A regional buffer (BUFR) can replace this BUFG.
In addition, the clock input of IFD can be driven by a BUFIO.
Refer to UG190, *Virtex-5 FPGA User Guide* for BUFR usage guidelines.

UG194_6_14_080409

Figure 6-14: **Tri-Mode RGMII v1.3 Clock Management**

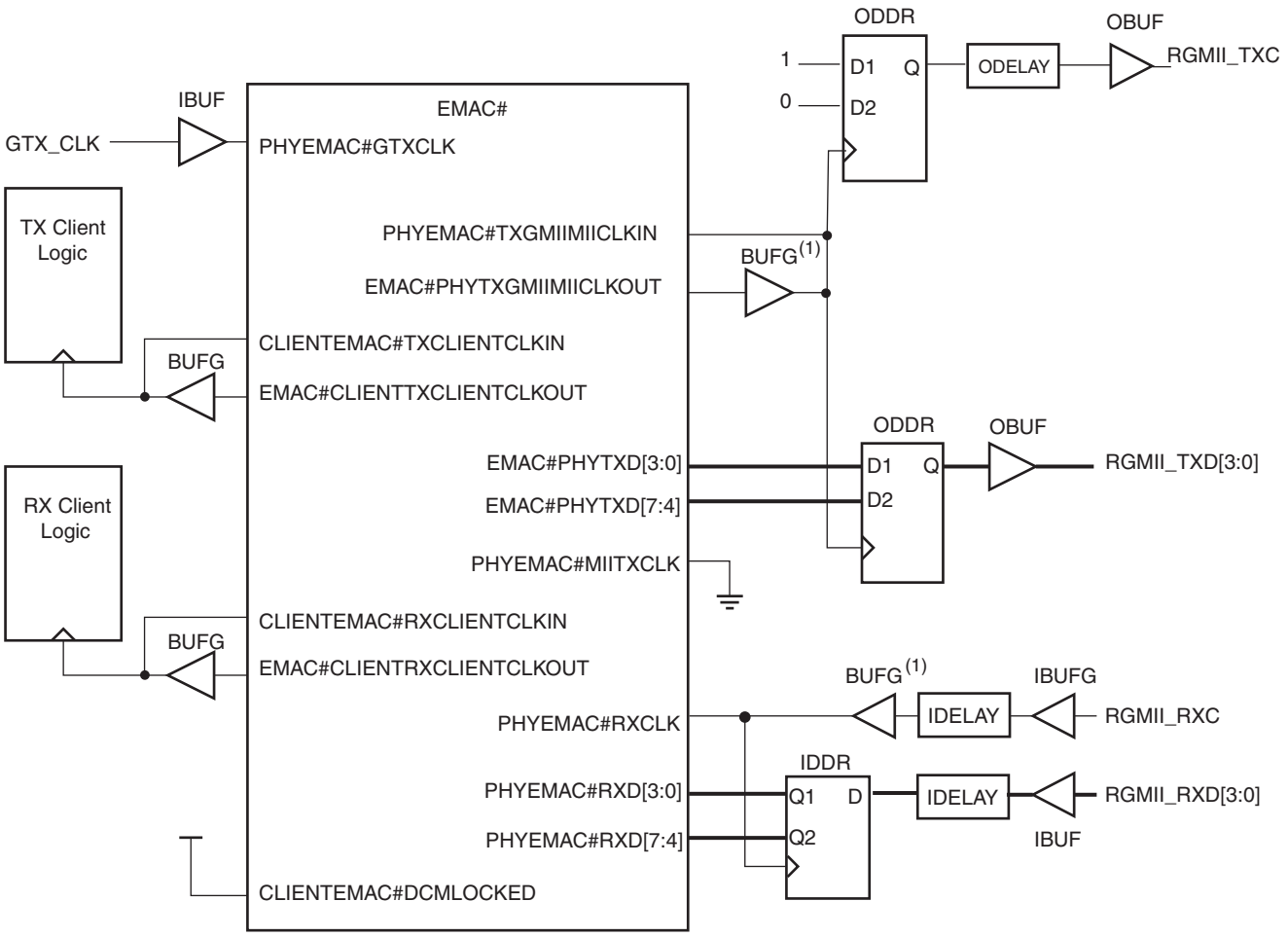
The EMAC#CLIENTTXCLIENTCLKOUT output port connects to the CLIENTEMAC#TXCLIENTCLKIN input port and transmitter client logic in the FPGA logic through a BUFG. The receiver client clocking is similar.

RGMII_TXC is derived from the Ethernet MAC by routing to an IOB 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*.

RGMII_RXC is generated by the PHY and connected to PHYEMAC#RXCLK via an IBUFG and a BUFG. Fixed-mode IDELAYs are instantiated on the RGMII clock and data lines. These are set to sample a 1 ns setup, 1 ns hold window at the device pads. The CLIENTEMAC#DCMLOCKED port must be tied High.

RGMII Version 2.0

Figure 6-15 shows the tri-speed 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#PHYTXGMIIMIICLKOUT 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.



Notes:
1) A regional buffer (BUFR) can replace this BUFG.
In addition, the clock input of IDDR can be driven by a BUFIO.
Refer to UG190, *Virtex-5 FPGA User Guide* for BUFR usage guidelines.

UG194_6_15_080409

Figure 6-15: **Tri-Mode RGMII v2.0 Clock Management**

RGMII_TXC is derived from the Ethernet MAC by routing the EMAC#PHYTXGMIIMIICLKOUT port to an IOB DDR output register, followed by an ODELAY element and an OBUF (which is then connected to the PHY). The ODELAY element is used to generate 2 ns of skew required between RGMII_TXC and RGMII_TXD at the FPGA device pads. This delay is specified in the *Hewlett Packard RGMII Specification, v2.0* to provide setup and hold times on the external interface.

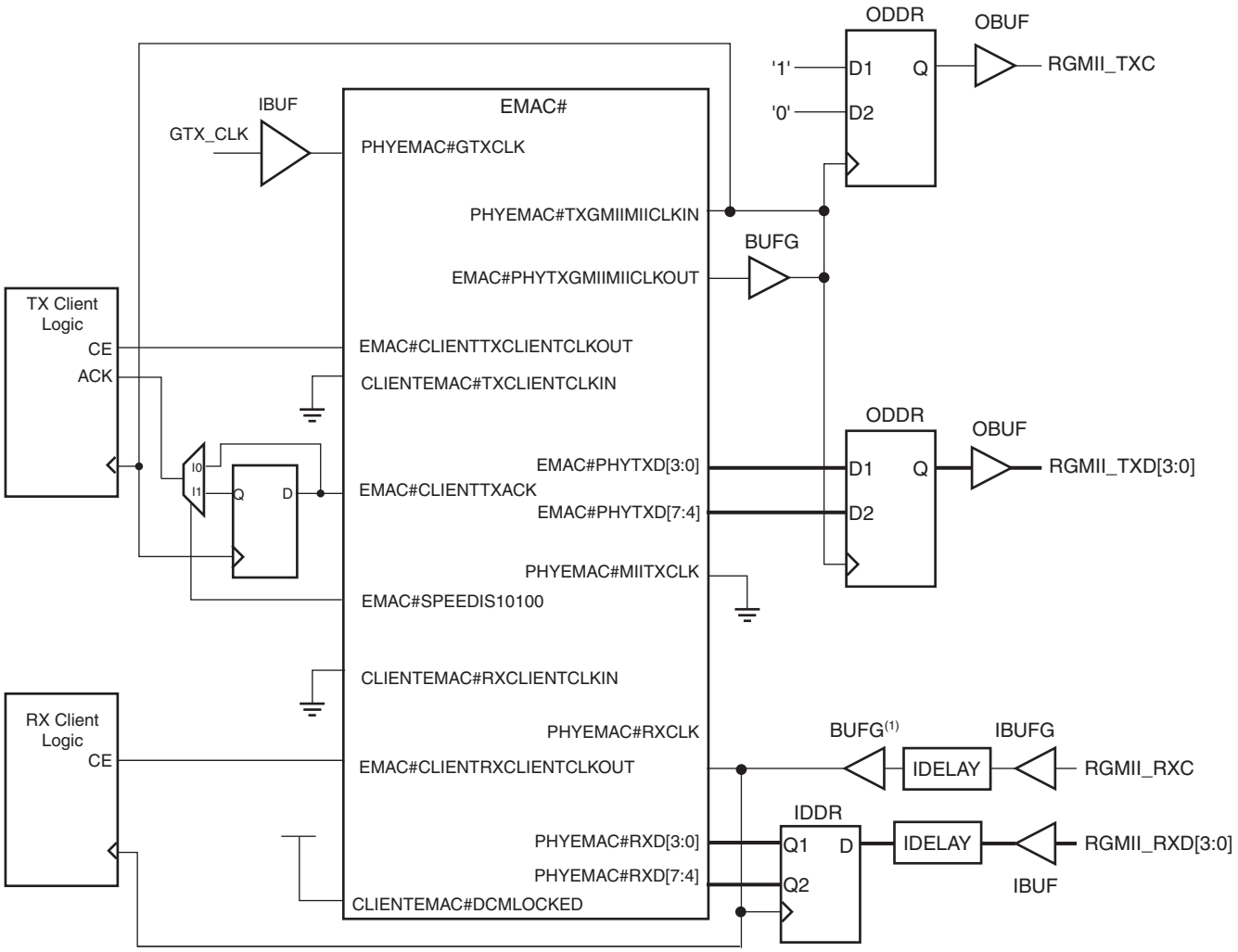
The EMAC#CLIENTTXCLIENTCLKOUT output port connects to the CLIENTEMAC#TXCLIENTCLKIN input port and transmitter client logic in the FPGA logic through a BUFG. The receiver client clocking is similar.

RGMII_RXC is generated by the PHY and connected to PHYEMAC#RXCLK via an IBUFG and a BUFG. Fixed-mode IDELAYs are instantiated on the RGMII clock and data lines. These are set to sample a 1 ns setup, 1 ns hold window at the device pads. The CLIENTEMAC#DCMLOCKED port must be tied High.

RGMII Clock Management for Tri-Speed Operation Using Clock Enables

RGMII Version 1.3

Figure 6-16 shows the clock management scheme for tri-speed RGMII v1.3 operation when the EMAC#_USECLKEN attribute is set.



Notes:
1) A regional buffer (BUFR) can replace this BUFG.
In addition, the clock input of IDDR can be driven by a BUFIO.
Refer to UG190, *Virtex-5 FPGA User Guide* for BUFR usage guidelines.

UG194_6_16_080409

Figure 6-16: **Tri-Mode RGMII with Clock Enables**

In this mode of operation, the transmitter signals are synchronous to PHYEMAC#TXGMIIMIICLKIN, and the receiver signals are synchronous to PHYEMAC#RXCLK.

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 25 MHz and 2.5 MHz, respectively. To maintain the correct data rate at the client interface, clock enable signals are output on EMAC#CLIENTTXCLIENTCLKOUT for the transmitter logic, and EMAC#CLIENTRXCLIENTCLKOUT for the receiver logic. These

signals are High during 1 Gb/s operation and toggle on each clock edge at slower speeds. These are used to enable the client interface logic.

Due to the timing relationship between the RGMII transmit clock and the internal client clock signals, the TX_ACK signal (CLIENTEMAC#TXACK) is registered at the output of the Ethernet MAC when operating at speeds below 1 Gb/s. At 1 Gb/s, the register is bypassed.

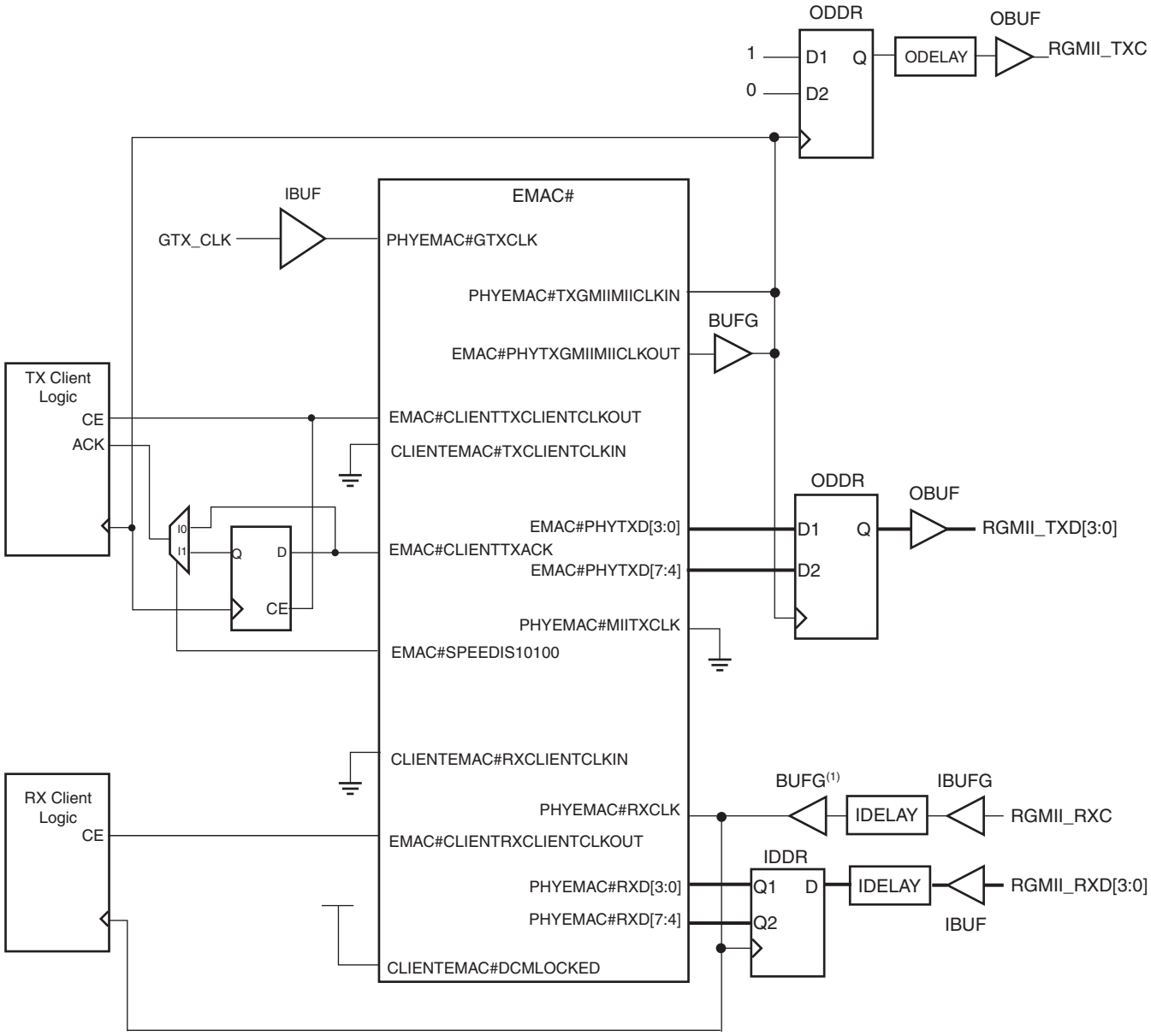
RGMII_TXC is derived from the Ethernet MAC by routing to an IOB DDR output register followed by an OBUF (which is then connected to the PHY). The use of the DDR register ensures that the forwarded clock is exactly in line with the RGMII transmitter data, as specified in the *Hewlett Packard RGMII Specification,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

RGMII_RXC is generated by the PHY and connected and connected to PHYEMAC#RXCLK via an IBUFG and a BUFG. Fixed-mode IDELAYs are instantiated on the RGMII clock and data lines. These are set to sample a 1 ns setup, 1 ns hold window at the device pads. RGMII_RXC drives all receive logic. The CLIENTEMAC#DCMLOCKED port must be tied High.

This clocking scheme requires two fewer BUFGs than the default scheme detailed in Figure 6-14.

RGMII Version 2.0

Figure 6-17 shows the clock management scheme for tri-speed RGMII v2.0 operation when the EMAC#_USECLKEN attribute is set.



Notes:
1) A regional buffer (BUFR) can replace this BUFG.
In addition, the clock input of IDDR can be driven by a BUFIO.
Refer to UG190, *Virtex-5 FPGA User Guide* for BUFR usage guidelines.

UG194_6_17_080409

Figure 6-17: **Tri-Mode RGMII with Clock Enables**

In this mode of operation, the transmitter signals are synchronous to `PHYEMAC#TXGMIIMIICLKIN`, and the receiver signals are synchronous to `PHYEMAC#RXCLK`.

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 25 MHz and 2.5 MHz, respectively. To maintain the correct data rate at the client interface, clock enable signals are output on EMAC#CLIENTTXCLIENTCLKOUT for the transmitter logic, and EMAC#CLIENTRXCLIENTCLKOUT for the receiver logic. These signals are High during 1 Gb/s operation and toggle on each clock edge at slower speeds. These are used to enable the client interface logic.

Due to the timing relationship between the RGMII transmit clock and the internal client clock signals, the TX_ACK signal (CLIENTEMAC#TXACK) is registered at the output of the Ethernet MAC when operating at speeds below 1 Gb/s. At 1 Gb/s, the register is bypassed.

RGMII_TXC is derived from the Ethernet MAC by routing the EMAC#PHYTXGMIIMIICLKOUT port to an IOB DDR output register, followed by an ODELAY element and an OBUF (which is then connected to the PHY). The ODELAY element is used to generate 2 ns of skew required between RGMII_TXC and RGMII_TXD at the FPGA device pads. This delay is specified in the *Hewlett Packard RGMII Specification, v2.0* to provide setup and hold times on the external interface.

RGMII_RXC is generated by the PHY and connected to PHYEMAC#RXCLK via an IBUFG and a BUFG. Fixed-mode IDELAYs are instantiated on the RGMII clock and data lines. These are set to sample a 1 ns setup, 1 ns hold window at the device pads. RGMII_RXC drives all receive logic. The CLIENTEMAC#DCMLOCKED port must be tied High.

This clocking scheme requires two fewer BUFGs than the default scheme detailed in Figure 6-15.

1000BASE-X PCS/PMA

Ethernet MAC PCS/PMA Sublayer

Figure 6-18 shows the functional block diagram of the Ethernet MAC with the PCS/PMA sublayer block highlighted.

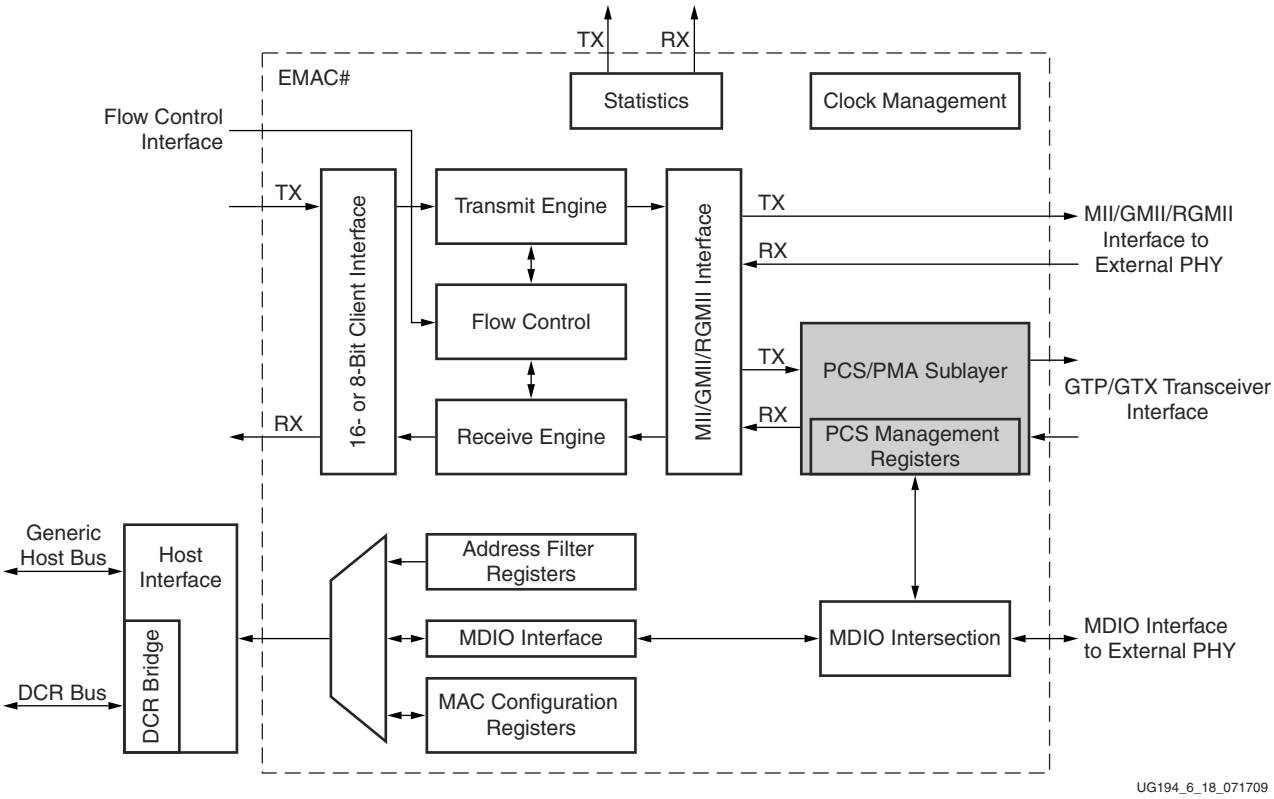


Figure 6-18: **Functional Block Diagram of the Ethernet MAC**

The PCS/PMA sublayer extends the functionality of the Ethernet MAC, replacing the GMII/MII or RGMII parallel interfaces with an interface to a RocketIO serial transceiver. The connected RocketIO serial transceiver then provides the remaining functionality to accommodate the following two standards, both of which require the high-speed serial interface provided by the RocketIO serial transceiver:

- "Serial Gigabit Media Independent Interface (SGMII)"
- "1000BASE-X PCS/PMA"

Introduction to the 1000BASE-X PCS/PMA Implementation

The 1000BASE-X physical standard, described in IEEE 802.3, clauses 36 and 37, defines a physical sublayer that is usually connected to an optical fibre medium. Two common implementations currently exist: 1000BASE-LX and 1000BASE-SX (long and short wavelength laser), which can both be obtained by connecting the RocketIO serial transceiver to a suitable GBIC or SFP optical transceiver.

Figure 6-19 shows an expanded diagram of the PCS/PMA sublayer block. The PCS/PMA sublayer contains its own functional blocks, which are illustrated and summarized in the following subsections. The RocketIO serial transceiver is shown connected to an external optical transceiver to complete the 1000BASE-X optical link.

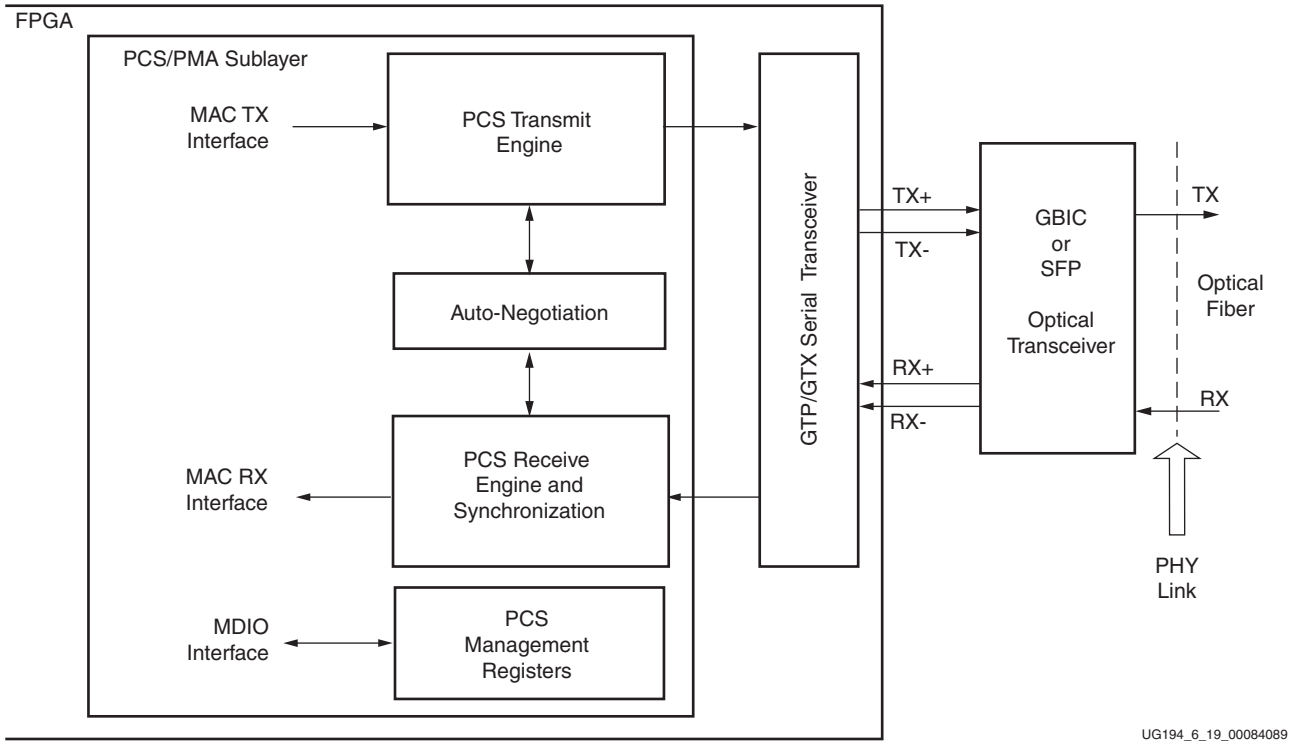


Figure 6-19: **Ethernet PCS/PMA Sublayer Extension**

PCS Transmit Engine

The PCS Transmit Engine encodes the datastream received from the MAC into a sequence of ordered sets as defined by the standard. The data is transmitted to the RocketIO serial transceiver using an 8-bit datapath that then provides 8B/10B encoding of this data and parallel to serial conversion. The output from the RocketIO serial transceiver is a differential pair operating at a rate of 1.25 Gb/s.

For more information on the PCS Transmit Engine, refer to the state diagrams of IEEE 802.3 (Figures 36-5 and 36-6).

PCS Receive Engine and Synchronization

The RocketIO serial transceiver receives a serial differential pair operating at a rate of 1.25 Gb/s from the optical transceiver. The RocketIO serial transceiver, using CDR, performs word alignment on this serial datastream and then converts this to a parallel interface. This data is then 8B/10B decoded by the RocketIO serial transceiver before presenting this to the PCS/PMA sublayer using an 8-bit datapath.

Within the PCS/PMA sublayer, the synchronization process performs analysis of valid/invalid received sequence ordered sets to determine if the link is in good order. The PCS receive engine then decodes these sequence ordered sets into a format that can then be presented to the standard receiver datapath within the Ethernet MAC.

For more information on the synchronization process, refer to IEEE 802.3 (Figure 36-9). For the PCS Receive Engine, refer to IEEE 802.3 (Figures 36-7a and 36-7b).

Auto-Negotiation

The 1000BASE-X Auto-Negotiation function allows a device to advertise the supported modes of operation to a device at the remote end of the optical fibre (the link partner) and to detect corresponding operational modes that the link partner can be advertising. Auto-negotiation is controlled and monitored using a software function through the PCS Management Registers. See "1000BASE-X Auto-Negotiation," page 174.

PCS Management Registers

Configuration and status of the PCS/PMA sublayer, including access to and from the Auto-Negotiation function, is via the PCS Management Registers. These registers are accessed through the serial Management Data Input/Output Interface (MDIO), as shown in Chapter 5, "MDIO Interface."

For the 1000BASE-X standard, the configuration and status registers available are listed in "1000BASE-X PCS/PMA Management Registers," page 122.

Ethernet MAC to RocketIO Serial Transceiver Connections

Figure 6-20 shows the Ethernet MAC configured with 1000BASE-X PCS/PMA as the physical interface. Connections to the Virtex®-5 FPGA RocketIO serial transceiver are illustrated.

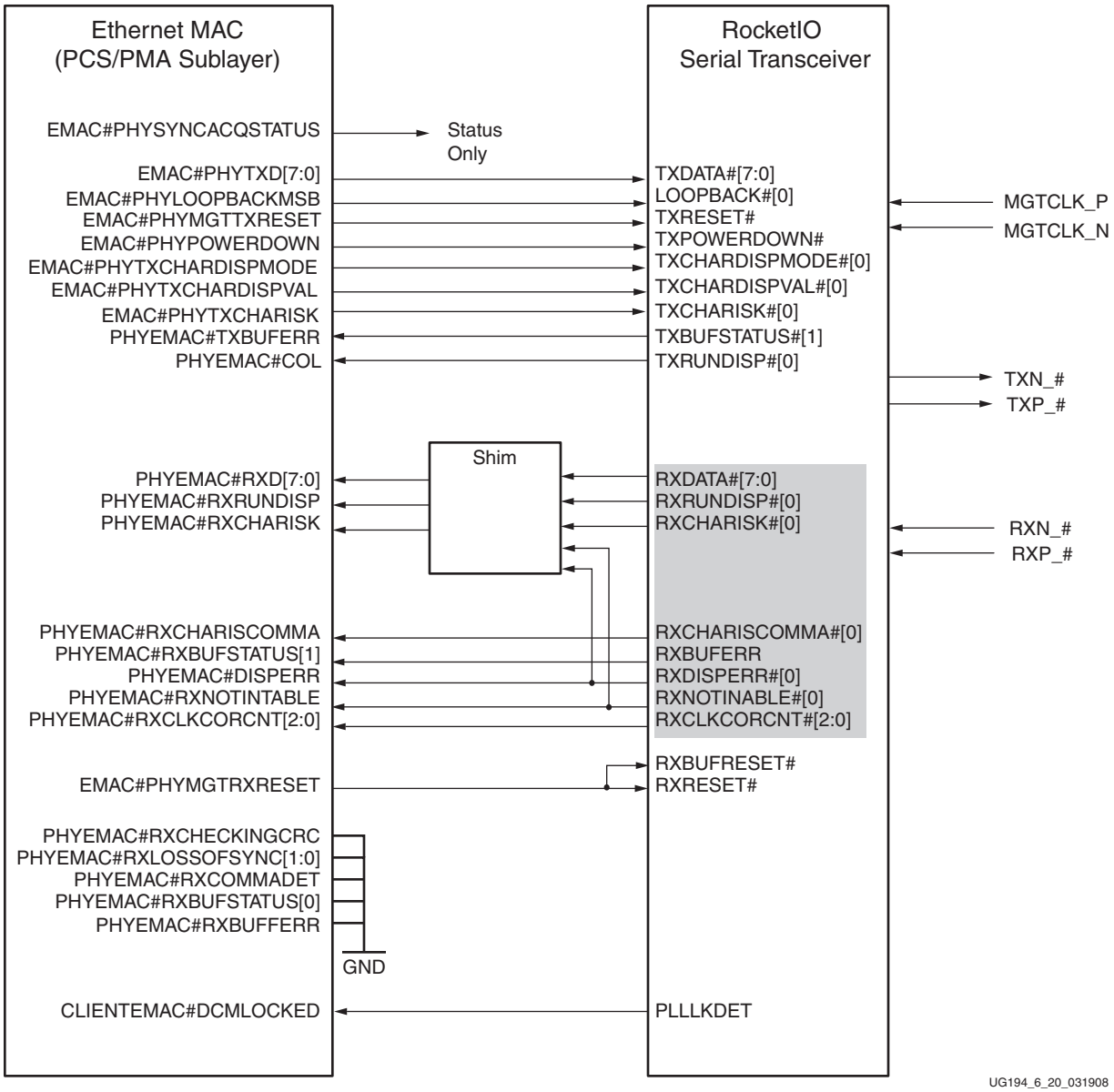


Figure 6-20: **Ethernet MAC Configured in 1000BASE-X PCS/PMA Mode**

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. See "Accessing the Ethernet MAC from the CORE Generator Tool," page 25.

Shim

Because of differences in the way the Virtex-II Pro FPGA MGTs, Virtex-4 FPGA MGTs, and the Virtex-5 FPGA RocketIO serial transceivers output the undecoded 8B/10B code group when `RXNOTINTABLE` is asserted, a shim is needed. The shim modifies the received data from the Virtex-5 FPGA RocketIO serial transceiver to the format that the Ethernet MAC is expecting. Table 6-1 describes the functionality of this shim.

The shim created by the CORE Generator tool is combinatorial. If this logic is pipelined, all of the signals originating from the RocketIO serial transceiver highlighted with gray shading in Figure 6-20 must be pipelined with an equal latency.

Table 6-1: **Shim Functionality**

Ethernet MAC Port Name	Connection to RocketIO Serial Transceiver Port Name	
	When RXNOTINTABLE#[0] = 0	When RXNOTINTABLE#[0] = 1
PHYEMAC#RXD[7]	RXDATA#[7]	RXDATA#[2]
PHYEMAC#RXD[6]	RXDATA#[6]	RXDATA#[3]
PHYEMAC#RXD[5]	RXDATA#[5]	RXDATA#[4]
PHYEMAC#RXD[4]	RXDATA#[4]	RXDATA#[5]
PHYEMAC#RXD[3]	RXDATA#[3]	RXDATA#[6]
PHYEMAC#RXD[2]	RXDATA#[2]	RXDATA#[7]
PHYEMAC#RXD[1]	RXDATA#[1]	RXCHARISK#[0]
PHYEMAC#RXD[0]	RXDATA#[0]	RXDISPERR#[0]
PHYEMAC#RXRUNDISP	RXRUNDISP#[0]	RXDATA#[1]
PHYEMAC#RXCHARISK	RXCHARISK#[0]	RXDATA#[0]

1000BASE-X PCS/PMA Clock Management (LXT and SXT Devices)

8-Bit Data Client

Figure 6-21 shows the clock management used with the 1000BASE-X PCS/PMA interface when the Ethernet MAC client is used with a 8-bit data client in LXT and SXT devices.

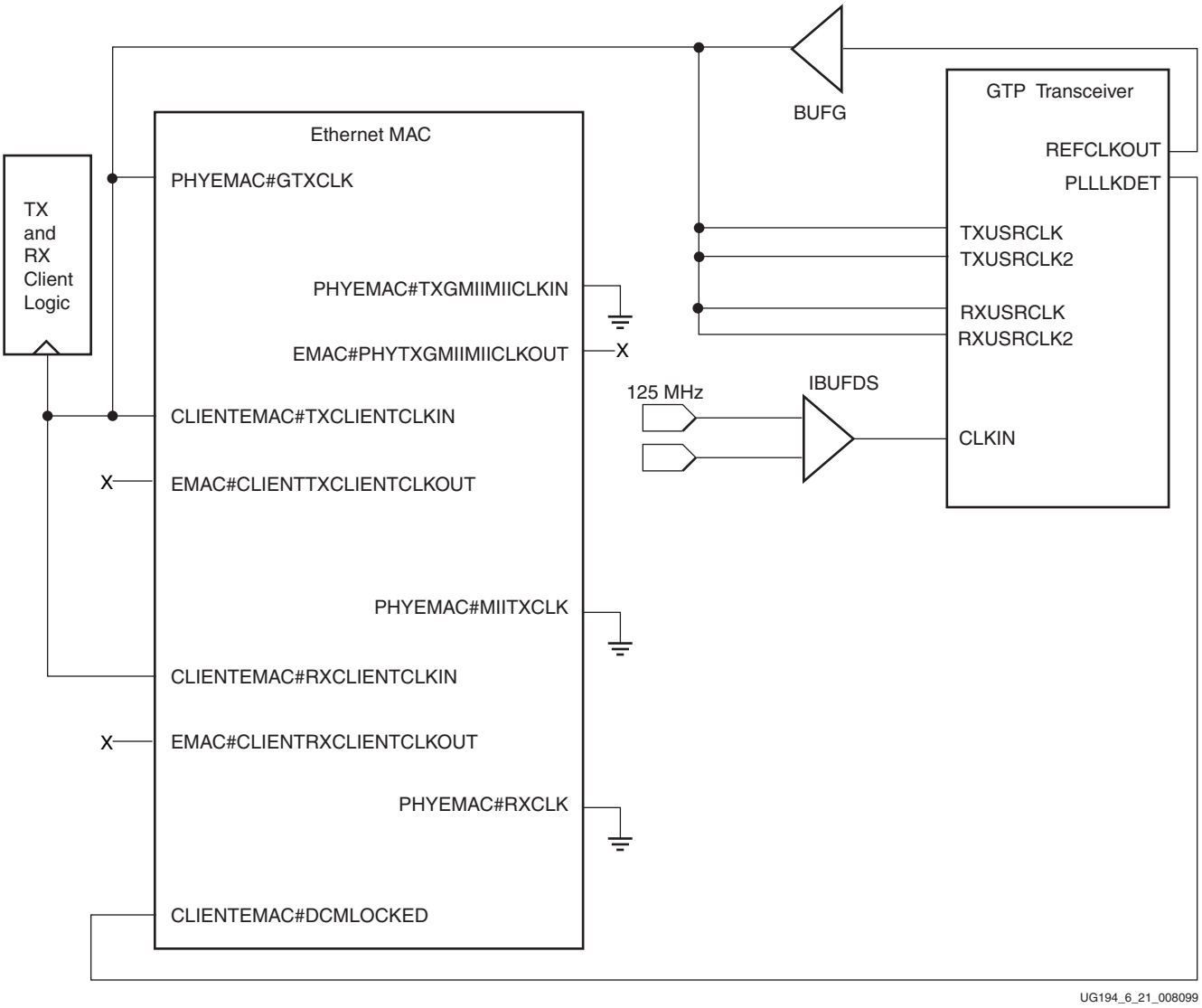


Figure 6-21: **1000BASE-X PCS/PMA (8-Bit Data Client) Clock Management in an LXT or SXT Device**

The CLKIN inputs to the RocketIO serial transceiver must be connected to an external, high-quality differential reference clock of frequency of 125 MHz. A 125 MHz clock source is then provided to the FPGA logic from the `REFCLKOUT` output port of the RocketIO serial transceiver. This is connected to global clock routing using a BUFG as illustrated. This clock should then be routed to the `PHYEMAC#GTXCLK` of the Ethernet MAC.

Additionally, this global clock can be used for both receiver and transmitter client logic as illustrated, and it, therefore, must be routed back to the Ethernet MAC through the `CLIENTEMAC#RXCLIENTCLKIN` and `CLIENTEMAC#TXCLIENTCLKIN` input ports.

The PLLLKDET signal from the RocketIO serial transceiver (indicating that its internal PLLs have locked) is routed to the CLIENTEMAC#DCMLOCKED input port of the Ethernet MAC. This ensures that the state machines of the Ethernet MAC are held in reset until the RocketIO serial transceiver has locked and its clocks are running cleanly.

As described in "Ethernet MAC Clocks," page 205, the following clock signals are unused:

- EMAC#CLIENTRXCLIENTCLKOUT
- EMAC#CLIENTTXCLIENTCLKOUT
- EMAC#PHYTXGMIIMIICLKOUT
- PHYEMAC#TXGMIIMIICLKIN
- PHYEMAC#MIITXCLK
- PHYEMAC#RXCLK

The outputs can be left unconnected, and inputs should be tied to Low.

16-Bit Data Client

Figure 6-22 shows the clock management used with the 1000BASE-X PCS/PMA interface and a 16-bit data client in an LXT or SXT device. This mode supports 2.5 Gb/s line rates.

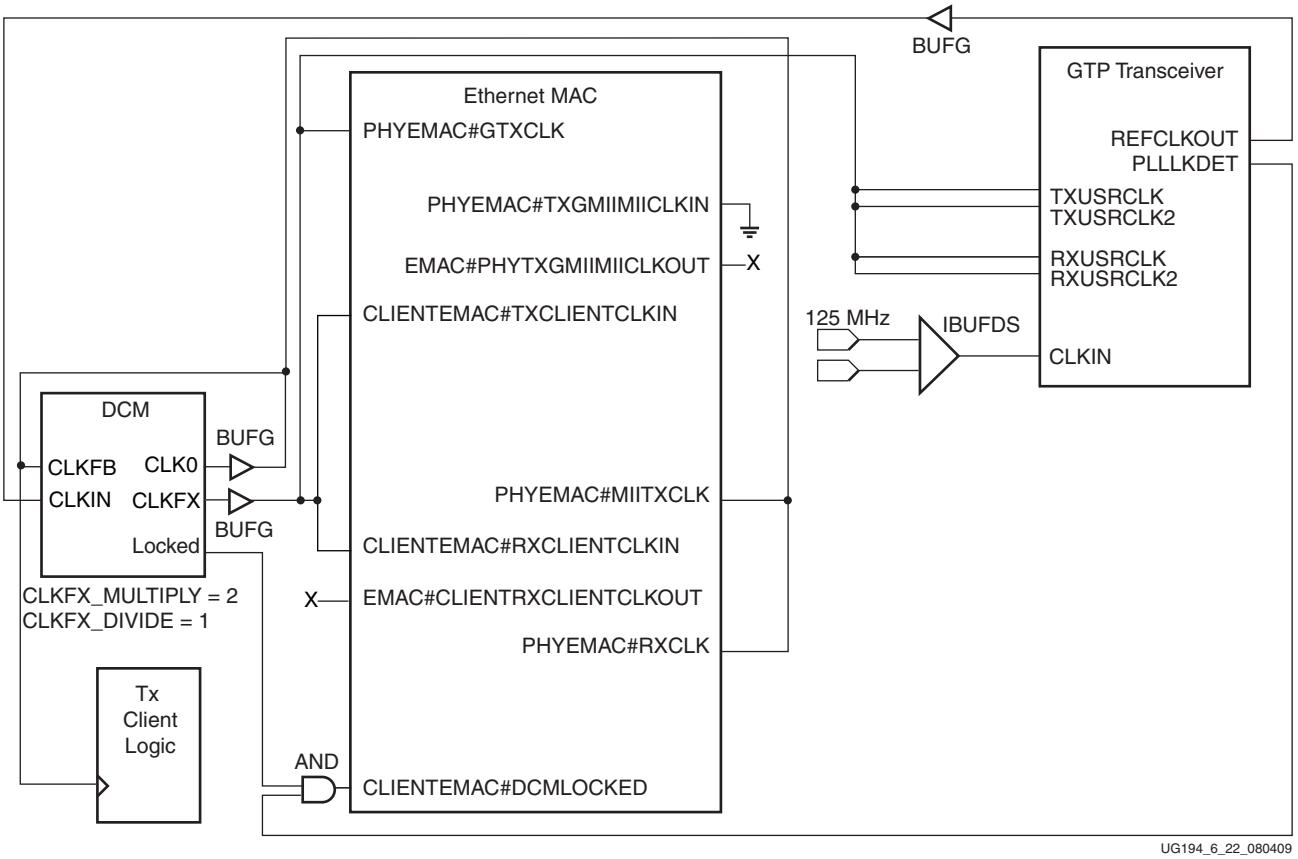


Figure 6-22: **1000BASE-X PCS/PMA (16-Bit Data Client) Clock Management in an LXT or SXT Device**

The CLKIN inputs to the RocketIO serial transceiver must be connected to an external, high-quality differential reference clock with a frequency of 125 MHz. A clock source matching the reference clock frequency is then provided to the FPGA logic from the

REFCLKOUT output port of the RocketIO serial transceiver. This clock is then routed to the CLKIN input of a DCM. CLK0 of the DCM is connected to global clock routing using a BUFG, as illustrated in Figure 6-22. The 125 MHz output of the BUFG is connected to the PHYEMAC#MIITXCLK and PHYEMAC#RXCLK ports of the Ethernet MAC. It is also used to clock both receiver and transmitter logic for the 16-bit Ethernet MAC client interface.

The global clock sourced from DCM CLKFX, running at 250 MHz, must be routed back to the Ethernet MAC through the input ports CLIENTEMAC#RXCLIENTCLKIN and CLIENTEMAC#TXCLIENTCLKIN. This clock is also connected to the PHYEMAC#GTXCLK input of the Ethernet MAC and the USRCLK inputs of the transceiver.

The PLLLKDET signal from the RocketIO serial transceiver (indicating that its internal PLLs have locked) is ANDed with the locked signal from the DCM and routed to the CLIENTEMAC#DCMLOCKED input port of the Ethernet MAC. This ensures that the state machines of the Ethernet MAC are held in reset until the RocketIO serial transceiver has locked and all clocks are running cleanly.

As described in "Ethernet MAC Clock Generation," page 206, the following clock signals are unused and can be left unconnected:

- EMAC#CLIENTRXCLIENTCLKOUT
- EMAC#PHYTXGMIIMIICLKOUT
- EMAC#CLIENTTXCLIENTCLKOUT

1000BASE-X PCS/PMA Clock Management (TXT and FXT Devices)

8-Bit Data Client

Figure 6-23 shows the clock management used with the 1000BASE-X PCS/PMA interface when the Ethernet MAC client is used with a 8-bit data client in a TXT and FXT device.

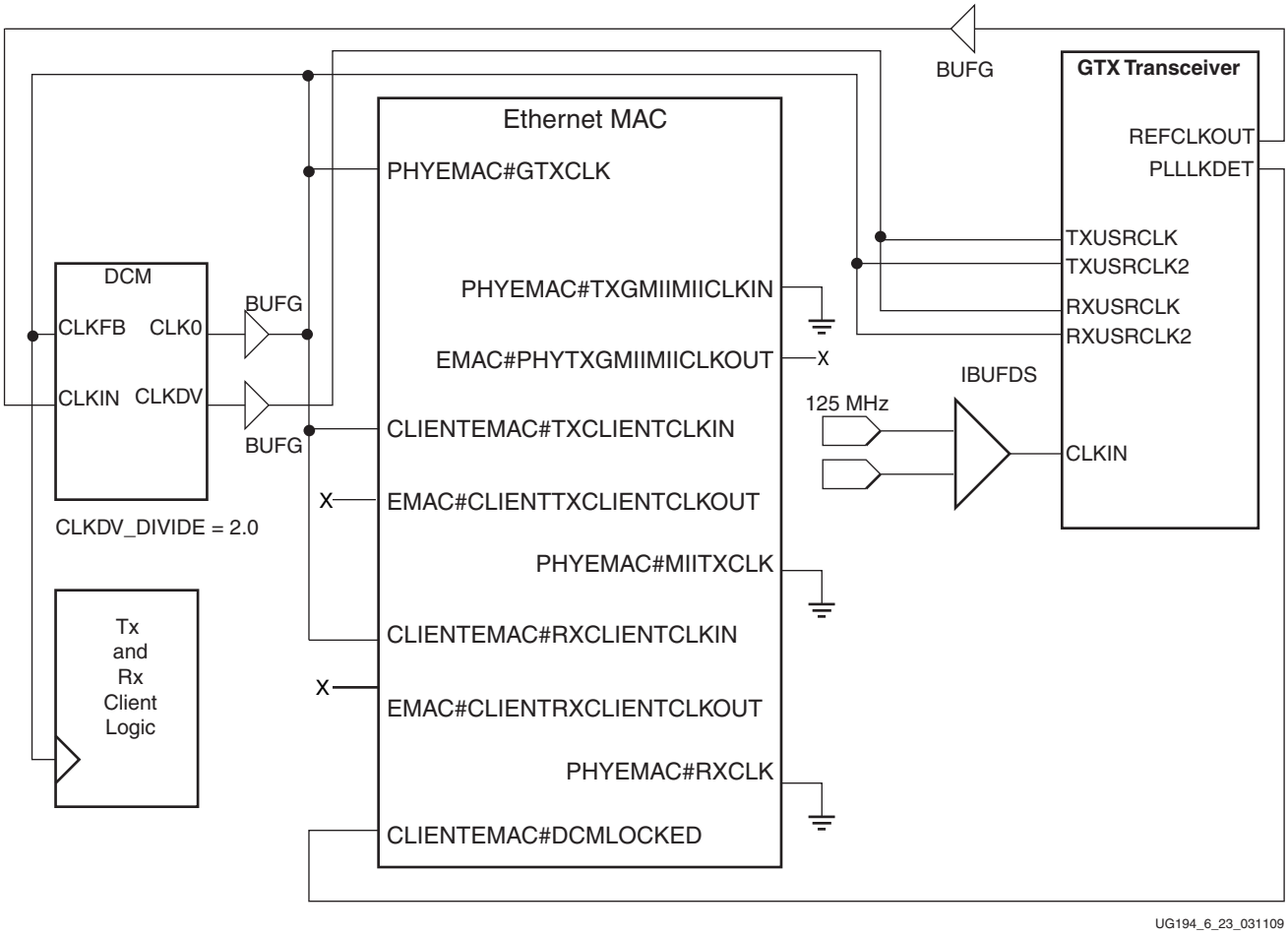


Figure 6-23: **1000BASE-X PCS/PMA (8-Bit Data Client) Clock Management in a TXT and FXT Device**

The CLKIN inputs to the RocketIO serial transceiver must be connected to an external, high-quality differential reference clock of frequency of 125 MHz. A 125 MHz clock source is then provided to the FPGA logic from the REFCLKOUT output port of the RocketIO serial transceiver. This is connected to the CLKIN input of a DCM, as illustrated in Figure 6-23. The CLK0 output of the DCM should then be routed to the PHYEMAC#GTXCLK of the Ethernet MAC via a BUFG. The clock is also routed to the TXUSRCLK2 and RXUSRCLK2 inputs of the GTX transceiver.

Additionally, this global clock can be used for both receiver and transmitter client logic, and it, therefore, must be routed back to the Ethernet MAC through the CLIENTEMAC#RXCLIENTCLKIN and CLIENTEMAC#TXCLIENTCLKIN input ports. A second 62.5 MHz clock is output from the DCM on the CLKDV port. This should be used to clock the TXUSRCLK and RXUSRCLK inputs of the GTX transceiver.

The PLLLKDET signal from the RocketIO serial transceiver (indicating that its internal PLLs have locked) is routed to the CLIENTEMAC#DCMLOCKED input port of the Ethernet MAC. This ensures that the state machines of the Ethernet MAC are held in reset until the RocketIO serial transceiver has locked, and its clocks are running cleanly.

As described in Appendix B, "Ethernet MAC Clocks," the following clock signals are unused:

- EMAC#CLIENTRXCLIENTCLKOUT
- EMAC#CLIENTTXCLIENTCLKOUT
- EMAC#PHYTXGMIIMIICLKOUT
- PHYEMAC#TXGMIIMIICLKIN
- PHYEMAC#MIITXCLK
- PHYEMAC#RXCLK

The outputs can be left unconnected, and inputs should be tied to Low.

16-Bit Data Client

Figure 6-24 shows the clock management used with the 1000BASE-X PCS/PMA interface and a 16-bit data client in a TXT and FXT device. This mode supports 2.5 Gb/s line rates.

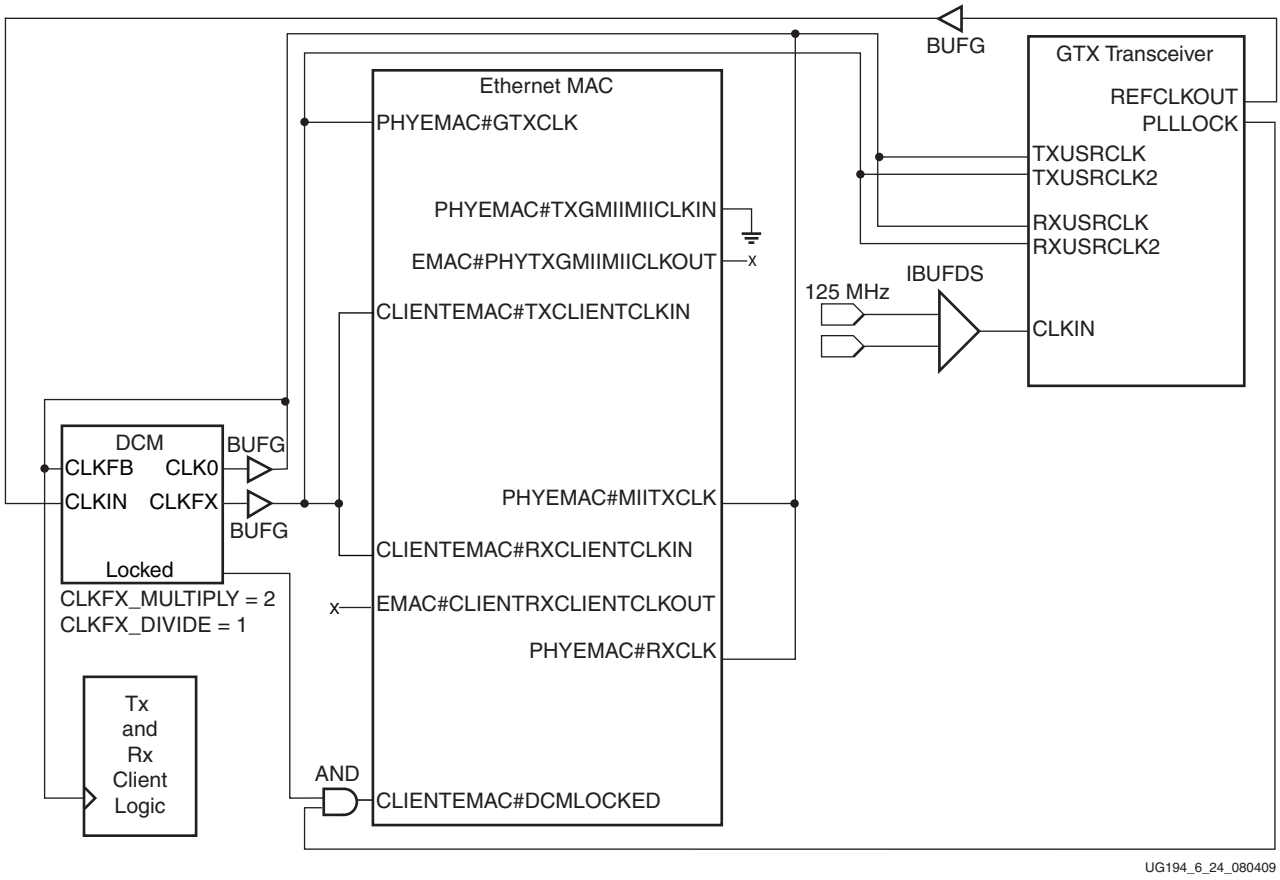


Figure 6-24: **1000BASE-X PCS/PMA (16-Bit Data Client) Clock Management in a TXT and FXT Device**

The CLKIN inputs to the RocketIO serial transceiver must be connected to an external, high-quality, differential reference clock with a frequency of 125 MHz. A clock source matching the reference clock frequency is then provided to the FPGA logic from the REFCLKOUT output port of the RocketIO serial transceiver. This clock is then routed to the CLKIN input of a DCM. CLK0 of the DCM is connected to global clock routing using a BUFG, as illustrated in Figure 6-24. The 125 MHz output of the BUFG is connected to the

PHYEMAC#MIITXCLK and PHYEMAC#RXCLK ports of the Ethernet MAC. It is also used to clock both receiver and transmitter logic for the 16-bit Ethernet MAC client interface and the RXUSRCLK and TXUSRCLK inputs of the GTX transceiver.

The global clock sourced from DCM CLKFX, running at 250 MHz, must be routed back to the Ethernet MAC through the input ports CLIENTEMAC#RXCLIENTCLKIN and CLIENTEMAC#TXCLIENTCLKIN. This clock is also connected to the PHYEMAC#GTXCLK input of the Ethernet MAC and the RXUSRCLK2 and TXUSRCLK2 inputs of the transceiver.

The PLLLKDET signal from the RocketIO serial transceiver (indicating that its internal PLLs have locked) is ANDed with the locked signal from the DCM and routed to the CLIENTEMAC#DCMLOCKED input port of the Ethernet MAC. This ensures that the state machines of the Ethernet MAC are held in reset until the RocketIO serial transceiver has locked and all clocks are running cleanly.

As described in "Ethernet MAC Clock Generation," page 206, the following clock signals are unused and can be left unconnected:

- EMAC#CLIENTRXCLIENTCLKOUT
- EMAC#PHYTXGMIIMIICLKOUT
- EMAC#CLIENTTXCLIENTCLKOUT

1000BASE-X Auto-Negotiation

Overview of Operation

Figure 6-25 illustrates a simplified diagram of the Ethernet MAC instantiated within a Virtex-5 device. The only components shown are two of the PCS Management Registers that are directly involved in the Auto-Negotiation process (see "1000BASE-X PCS/PMA Management Registers," page 122). The corresponding registers of the connected device is also shown.

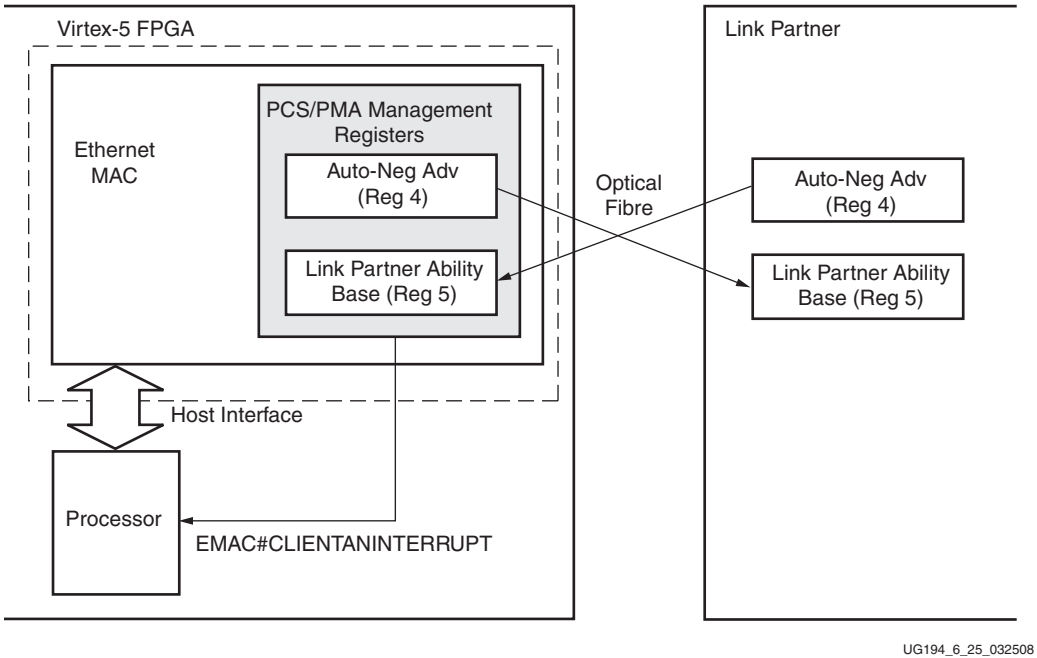


Figure 6-25: **1000BASE-X Auto-Negotiation Overview**

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 below:

- To enable auto-negotiation, both auto-negotiation (see "Control Register (Register 0)," page 123) and MDIO (see "Management Configuration Register," page 94) must be enabled. If enabled, auto-negotiation starts automatically:
  - After power-up/reset
  - Upon loss of synchronization
  - Whenever the link partner initiates auto-negotiation
  - Whenever an auto-negotiation restart is requested (see "Control Register (Register 0)")
  - When the PHY Reset bit is set in the PCS control register (see "Control Register (Register 0)")
- During auto-negotiation, the contents of the "Auto-Negotiation Advertisement Register (Register 4)" are transferred to the link partner. This register can be written to through the management interface, and enables software control of the systems advertised abilities. Information provided in this register includes:
  - Fault condition signalling
  - Duplex mode
  - Flow control capabilities for the Ethernet MAC
- At the same time, the advertised abilities of the link partner are transferred into the "Auto-Negotiation Link Partner Ability Base Register (Register 5)." This includes the same information fields as in the "Auto-Negotiation Advertisement Register (Register 4)."
- Under normal conditions, this completes the Auto-Negotiation information exchange. The results can be read from the "Auto-Negotiation Link Partner Ability Base Register (Register 5)." The mode of the Ethernet MAC should then be configured by a software routine to match; this does not happen automatically within the Ethernet MAC. The two methods by which a host processor can learn of the completion of an Auto-Negotiation cycle are:
  - By polling the Auto-Negotiation completion bit 1.5 in "Status Register (Register 1)."
  - By using the Auto-Negotiation interrupt port (see "Auto-Negotiation Interrupt").

Auto-Negotiation Link Timer

The Auto-Negotiation Link Timer is used to time three phases of the Auto-Negotiation procedure. For the 1000BASE-X standard, this link timer is defined as having a duration of between 10 and 20 ms. Both link partners wait until their own link timer and the partner's link timer expire before moving on to the next phase of the auto negotiation process. The complete process takes a minimum of three times the values of the biggest link timer value.

The duration of the link timer used by the Ethernet MAC can be configured with the EMAC#_LINKTIMERVAL[8:0] attribute (see "Physical Interface Attributes," page 48). The duration of the timer is approximately equal to the binary value placed onto this attribute multiplied by 32.768 µs (4096 clock periods of the 125 MHz clock provided to the Ethernet MAC on PHYEMAC#GTXCLK). The accuracy of this link timer is within the range:

+0 to -32.768 µs

Therefore, for the 1000BASE-X standard, the attribute EMAC#_LINKTIMERVAL[8:0] is set to:

`100111101` = 317 decimal

This setting corresponds to a timer duration of between 10.354 and 10.387 ms. This value can be reduced for simulation.

Auto-Negotiation Interrupt

The Auto-Negotiation function has an `EMAC#CLIENTANINTERRUPT` port. This port is designed to be used with common microprocessor bus architectures.

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

Use of Clock Correction Sequences

The RocketIO serial transceiver is configured by the appropriate Transceiver Wizard to perform clock correction. The output of the Transceiver Wizard is provided as part of the Ethernet MAC wrapper generated using the CORE Generator tool. Two different clock correction sequences can be employed:

- The mandatory clock correction sequence is the /I2/ ordered set; this is a two-byte code-group sequence formed from /K28.5/ and /D16.2/ characters. The /I2/ ordered set is present in the interframe gap. These sequences can therefore be removed or inserted by the transceiver's receiver elastic buffer without causing frame corruption.
- The default Transceiver Wizard configuration enables the CLK_COR_SEQ_2_USE attribute. In this case, the transceiver is also configured to perform clock correction on the /K28.5/D21.5/ sequence; these are the first two code-groups from the /C1/ ordered set (the /C1/ ordered set is four code-groups in length). Because there are no /I2/ ordered sets present during much of the auto-negotiation cycle, this provides a method of allowing clock correction to be performed during auto-negotiation. Because this form of clock correction inserts or removes two code groups into or from a four code-group sequence, this causes ordered-set fragments to be seen by the Ethernet MAC's auto-negotiation state machine. It is therefore important that the transceiver's RXCLKCORCNT[2:0] port be correctly connected to the Ethernet MAC; this indicates a clock correction event (and type) to the Ethernet MAC. Using this signal, the Ethernet MAC's state machine can interpret the clock-correction fragments, and the auto-negotiation function can complete cleanly.

  When the transceiver's CLK_COR_SEQ_2_USE attribute is not enabled, no clock correction can be performed during much of the auto-negotiation cycle. When this is the case, it is possible that the transceiver's receiver elastic buffer could underflow or overflow as asynchronous clock tolerances accumulate. This results in an elastic buffer error. It is therefore important that the transceiver's RXBUFSTATUS[2:0]port is correctly connected to the Ethernet MAC; this indicates a buffer error event to the

Ethernet MAC. Using this signal, the Ethernet MAC's state machine can interpret the buffer error and the auto-negotiation function can complete cleanly.

The RocketIO serial transceiver is by default configured to perform clock correction during the auto-negotiation cycle. If correctly connected as per the example design, the Ethernet MAC's state machine can correctly determine the transceiver's elastic buffer behavior, and auto-negotiation can complete cleanly.

Loopback When Using the PCS/PMA Sublayer

Figure 6-26 illustrates the loopback options when using the PCS/PMA sublayer. Two possible loopback positions are shown:

- **Loopback in the Ethernet MAC.** When placed into loopback, data is routed from the transmitter to the receiver path at the last possible point in the PCS/PMA sublayer. This is immediately before the RocketIO serial transceiver interface. When placed into loopback, a constant stream of Idle code groups are transmitted through the RocketIO serial transceiver.

  Loopback in this position allows test frames to be looped back within the Ethernet MAC without allowing them to be received by the link partner (the device connected at the other end of the optical link). The transmitting of idles allows the link partner to remain in synchronization so that no fault is reported.

- **Loopback in the RocketIO serial transceiver.** The RocketIO serial transceiver can alternatively be switched into loopback by connecting the `EMAC#PHYLOOPBACKMSB` port of the Ethernet MAC to the loopback port of the RocketIO serial transceiver as illustrated. The RocketIO serial transceiver loopback routes data from the transmitter path to the receiver path within the RocketIO serial transceiver. However, this data is also transmitted out of the RocketIO serial transceiver and so any test frames used for a loopback test are received by the link partner.

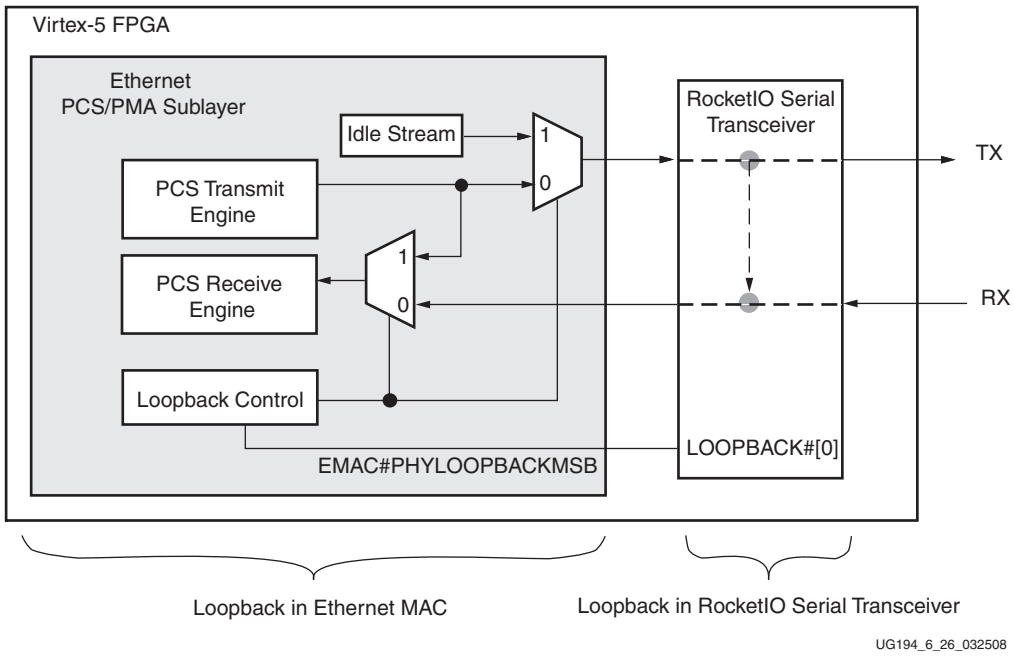


Figure 6-26: **Loopback When Using the PCS/PMA Sublayer**

Loopback itself can be enabled or disabled by writing to the PCS/PMA Sublayer Control Register (Register 0). The loopback position can be controlled through the Loopback Control Register (Register 17). See "1000BASE-X PCS/PMA Management Registers," page 122 for details. Alternatively, loopback can be controlled by setting the EMAC#_PHYLOOPBACKMSB and EMAC#_GTLOOPBACK attributes. See Chapter 2 for details.

Serial Gigabit Media Independent Interface (SGMII)

Ethernet MAC PCS/PMA Sublayer

Figure 6-27 shows the functional block diagram of the Ethernet MAC with the PCS/PMA sublayer block highlighted.

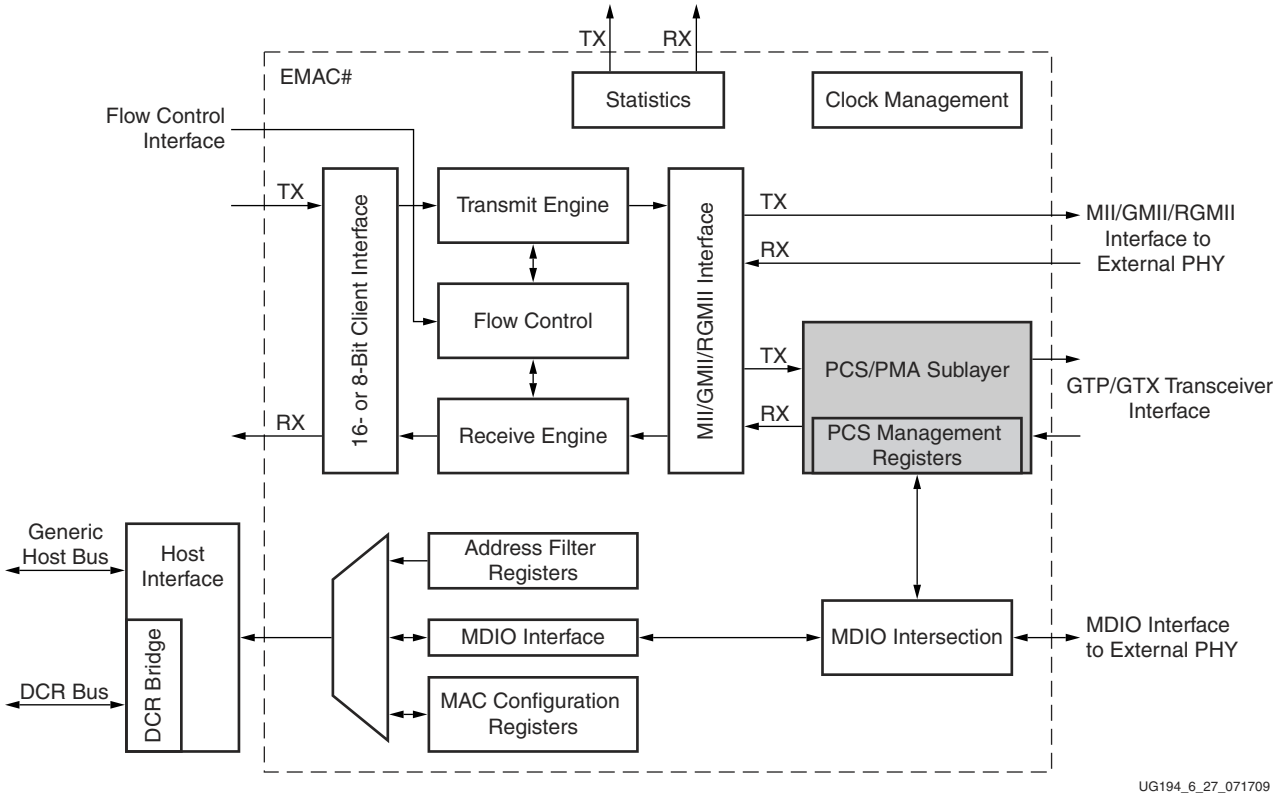


Figure 6-27: **PCS/PMA Sublayer Block Diagram in the Ethernet MAC**

The PCS/PMA sublayer extends the functionality of the Ethernet MAC, replacing the GMII/MII or RGMII parallel interfaces with an interface to a RocketIO serial transceiver. The connected RocketIO serial transceiver then provides the remaining functionality to accommodate the following two standards, both of which require the high-speed serial interface provided by the RocketIO serial transceiver:

- "Serial Gigabit Media Independent Interface (SGMII)"
- "1000BASE-X PCS/PMA"

Introduction to the SGMII Implementation

The Serial GMII (SGMII) is an alternative interface to the GMII/MII that converts the parallel interface of the GMII/MII into a serial format that is capable of carrying traffic at speeds of 10 Mb/s, 100 Mb/s, and 1 Gb/s. It radically reduces the I/O count and is, therefore, often favored by PCB designers.

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

The sideband clock signals defined in the specification are not implemented in the Ethernet MAC. Instead, the RocketIO serial transceiver is used to transmit and receive the differential data at the required rate using clock data recovery (CDR). For more information on SGMII, refer to the Serial GMII specification v1.7.

Figure 6-28 shows an expanded diagram of the PCS/PMA sublayer block. The Ethernet MAC is connected to a RocketIO serial transceiver, which in turn connects to a Tri-Speed Ethernet BASE-T PHY device via SGMII.

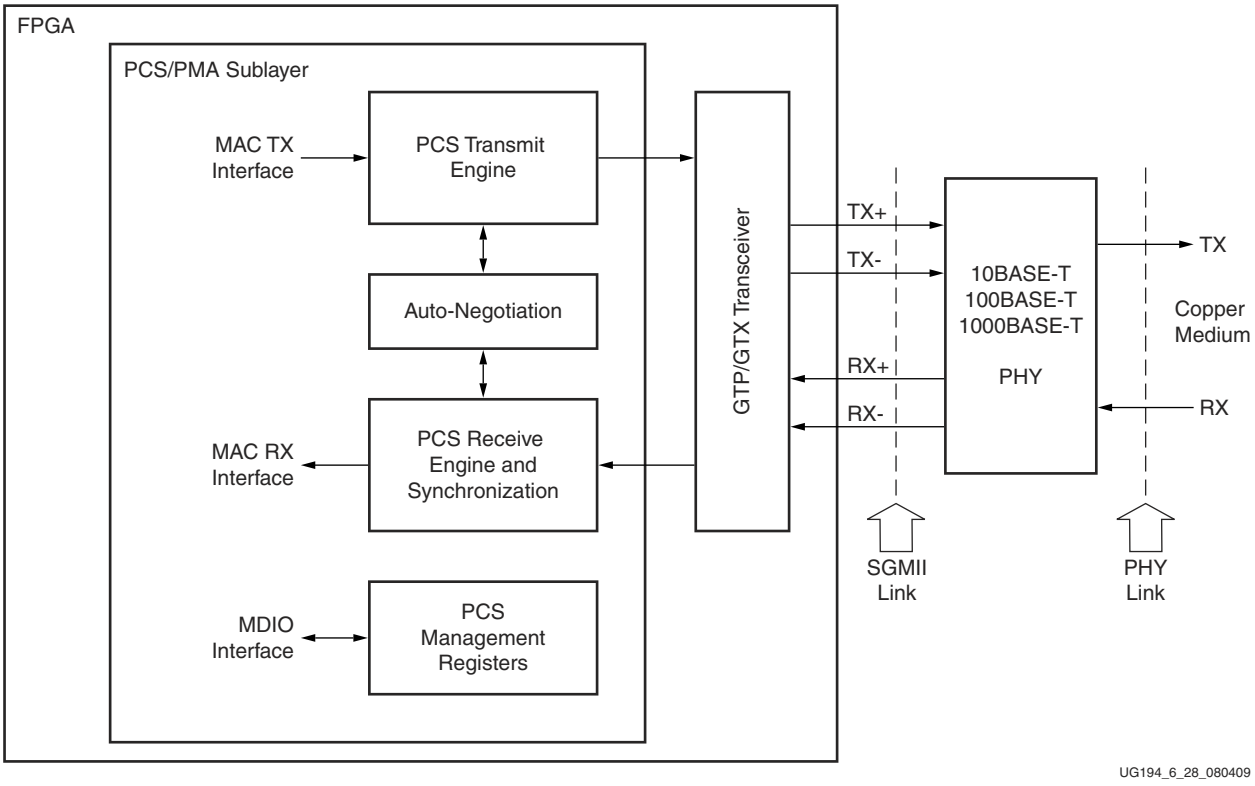


Figure 6-28: **Ethernet PCS/PMA Sublayer Extension**

The PCS/PMA sublayer contains its own functional blocks, which are illustrated and summarized in the following subsections.

PCS Transmit Engine

The PCS Transmit Engine encodes the datastream received from the Ethernet MAC into a sequence of ordered sets. The data is then transmitted to the RocketIO serial transceiver, using an 8-bit datapath, which then provides 8B/10B encoding of this data and parallel to serial conversion. The output from the RocketIO serial transceiver is a differential pair operating at a rate of 1.25 Gb/s.

PCS Receive Engine and Synchronization

The RocketIO serial transceiver receives a serial differential pair operating at a rate of 1.25 Gb/s from the SGMII PHY. The RocketIO serial transceiver, using CDR, performs word alignment on this serial datastream and then converts this to a parallel interface. This data is then 8B/10B decoded by the RocketIO serial transceiver before presenting this to the PCS/PMA sublayer using an 8-bit datapath.

Within the PCS/PMA sublayer, the synchronization process performs analysis of valid/invalid received sequence ordered sets to determine if the link is in good order. The PCS receive engine then decodes these sequence ordered sets into a format that can then be presented to the standard receiver datapath within the Ethernet MAC.

Auto-Negotiation

The SGMII Auto-Negotiation function allows the Ethernet MAC to communicate with the attached PHY device. Auto-negotiation is controlled and monitored using a software function through the PCS Management Registers. See "SGMII Auto-Negotiation," page 196.

PCS Management Registers

Configuration and status of the PCS/PMA sublayer, including access to and from the Auto-Negotiation function, is via the PCS Management Registers. These registers are accessed through the serial MDIO, see Chapter 5, "MDIO Interface."

For the SGMII standard, the configuration and status registers available are listed in "SGMII Management Registers," page 130.

SGMII RX Elastic Buffer

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

- Using the buffer present in the RocketIO serial transceivers
- 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:

- "RocketIO Serial Transceiver Logic Using the RX Elastic Buffer"
- "RocketIO Serial Transceiver Logic Using the RX Elastic Buffer in FPGA Logic"

RX Elastic Buffer Implementations

Selecting the Buffer Implementation from the GUI

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

- Option 1: **10/100/1000 Mb/s clock tolerance compliant with Ethernet specification**
  - For this option, if the FPGA and PHY clocks are independent and each has 100 ppm tolerance, an additional elastic buffer is required (100 slices and 1 block RAM per Ethernet MAC).
- Option 2: **10/100/1000 Mb/s (restricted tolerance for clocks) OR 100/1000 Mb/s**
  - For this option, if 10 Mb/s is not required OR it is required but the FPGA and PHY clocks are closely related (see UG196, *Virtex-5 RocketIO GTP Transceiver User Guide* or UG198, *Virtex-5 RocketIO GTX Transceiver User Guide*), the RX buffer can be used (saves 100 slices and 1 block RAM per Ethernet MAC).

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 RocketIO serial transceiver, therefore consuming extra logic resources. However, this default mode provides reliable SGMII operation.

Option 2 uses the RX elastic buffer in the RocketIO serial transceivers. 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 6-29 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 6-29, 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 RocketIO serial transceiver 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 UG196, *Virtex-5 RocketIO GTP Transceiver User Guide* or UG198, *Virtex-5 RocketIO GTX Transceiver User Guide*).

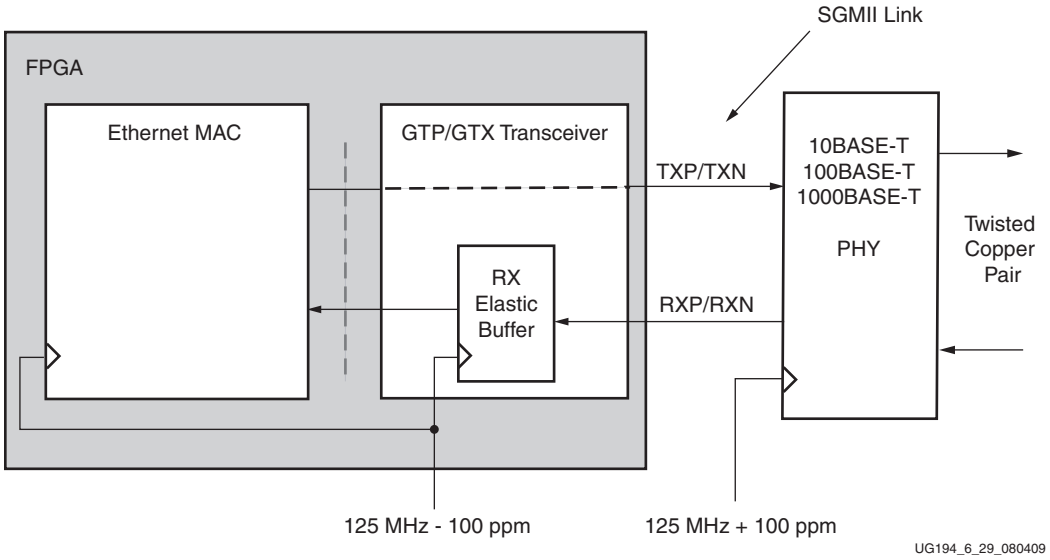


Figure 6-29: **SGMII Implementation Using Separate Clock Sources**

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 ensure that the buffer does not underflow or overflow during frame reception. This assumes that frame reception begins when the buffer is exactly half full.

RocketIO Serial Transceiver Elastic Buffer

The RX elastic buffer in the RocketIO serial transceivers has 64 entries. 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 6-30 illustrates the elastic buffer depths and thresholds of RocketIO serial transceivers in Virtex-5 devices. Each FIFO word corresponds to a single character of data (equivalent to a single byte of data following 8B/10B decoding).

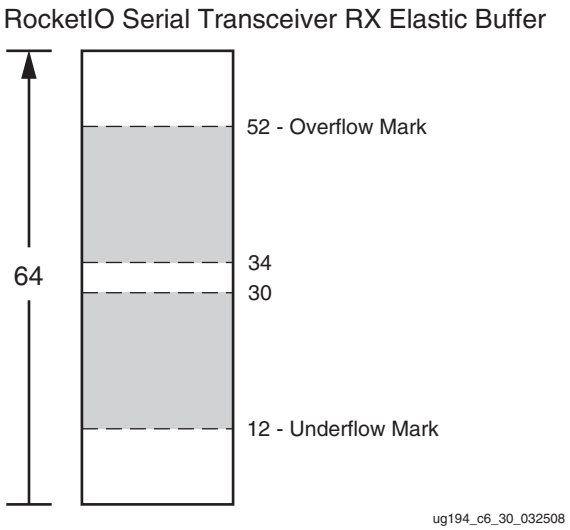


Figure 6-30: **Elastic Buffer Size for RocketIO Serial Transceivers**

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

- If the buffer is filling during frame reception, then 52 – 34 = 18 FIFO locations are available before the buffer hits the overflow mark.
- If the buffer is emptying during reception, then 30 – 12 = 18 FIFO locations are 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 5000 x 18 = 90000 bytes.

Table 6-2 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 6-2: **Maximum Frame Sizes for RocketIO Serial Transceiver RX Elastic Buffers (100 ppm Clock Tolerance)**

Standard	Speed	Maximum Frame Size
SGMII	1 Gb/s	90000
SGMII	100 Mb/s	9000
SGMII	10 Mb/s	900

FPGA Logic Elastic Buffer

For reliable SGMII operation at 10 Mb/s (non-jumbo frames), the RocketIO serial transceiver 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 6-31 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 RocketIO serial transceiver. See "RocketIO Serial Transceiver Logic Using the RX Elastic Buffer in FPGA Logic," page 188.

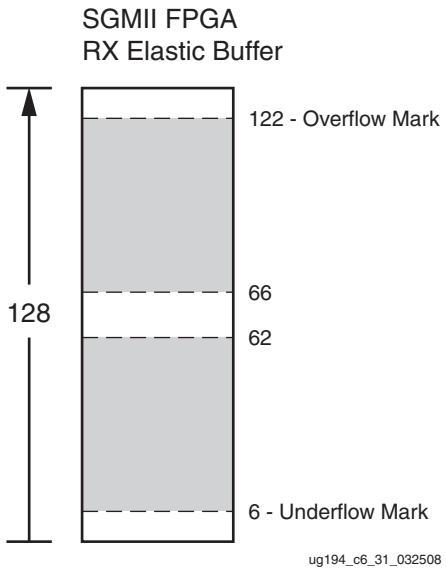


Figure 6-31: **Elastic Buffer Size for FPGA Logic Buffer**

The shaded area in Figure 6-31 represents the usable buffer availability for the duration of frame reception.

- If the buffer is filling during frame reception, then 122 – 66 = 56 FIFO locations are available before the buffer hits the overflow mark.
- If the buffer is emptying during reception, then 62 – 6 = 56 FIFO locations are 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 in Figure 6-31, 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 6-3 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 6-3: **Maximum Frame Sizes for RocketIO Serial Transceiver RX Elastic Buffers (100 ppm Clock Tolerance)**

Standard	Speed	Maximum Frame Size
SGMII	1 Gb/s	280000
SGMII	100 Mb/s	28000
SGMII	10 Mb/s	2800

Using the RocketIO Serial Transceiver RX Elastic Buffer

The RX elastic buffer implemented in the RocketIO serial transceiver 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 6-32 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.

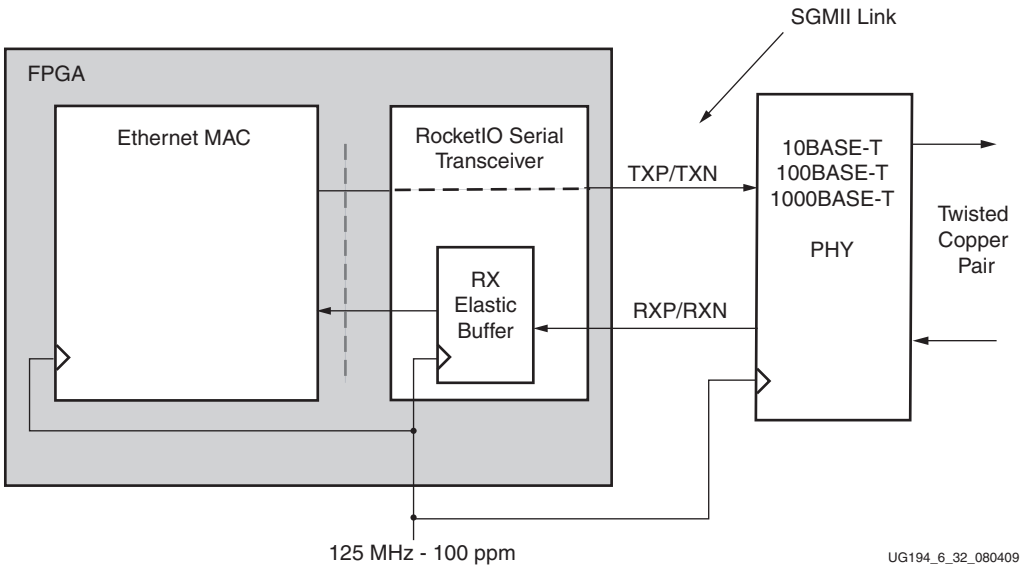


Figure 6-32: **SGMII Implementation Using Shared Clock Sources**

  If the PHY device sources the receiver SGMII stream synchronously from the shared oscillator (refer to the PHY data sheet), the RocketIO serial transceiver 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 RocketIO serial transceiver is recommended to save logic resources.

- Case 2

  Using the case illustrated by Figure 6-29, 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 RocketIO serial transceiver 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 6-33 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 RocketIO serial transceiver's elastic buffer has been bypassed and the FPGA elastic buffer is used.

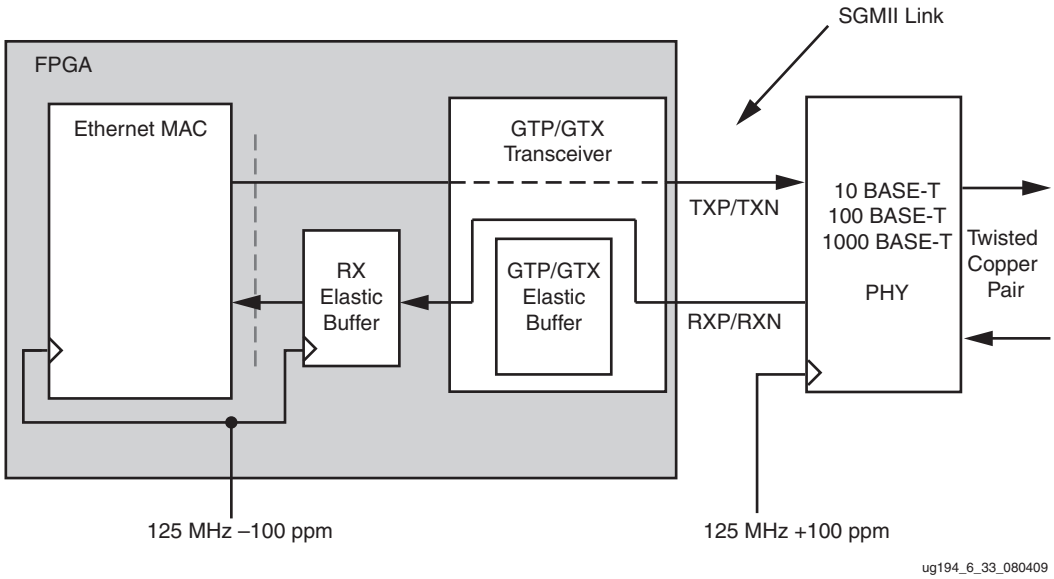


Figure 6-33: **SGMII Implementation Using a Logic Buffer**

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

RocketIO Serial Transceiver Logic Using the RX Elastic Buffer

Figure 6-34 shows the Ethernet MAC configured with SGMII as the physical interface. Connections to the Virtex-5 FPGA RocketIO serial transceiver are illustrated.

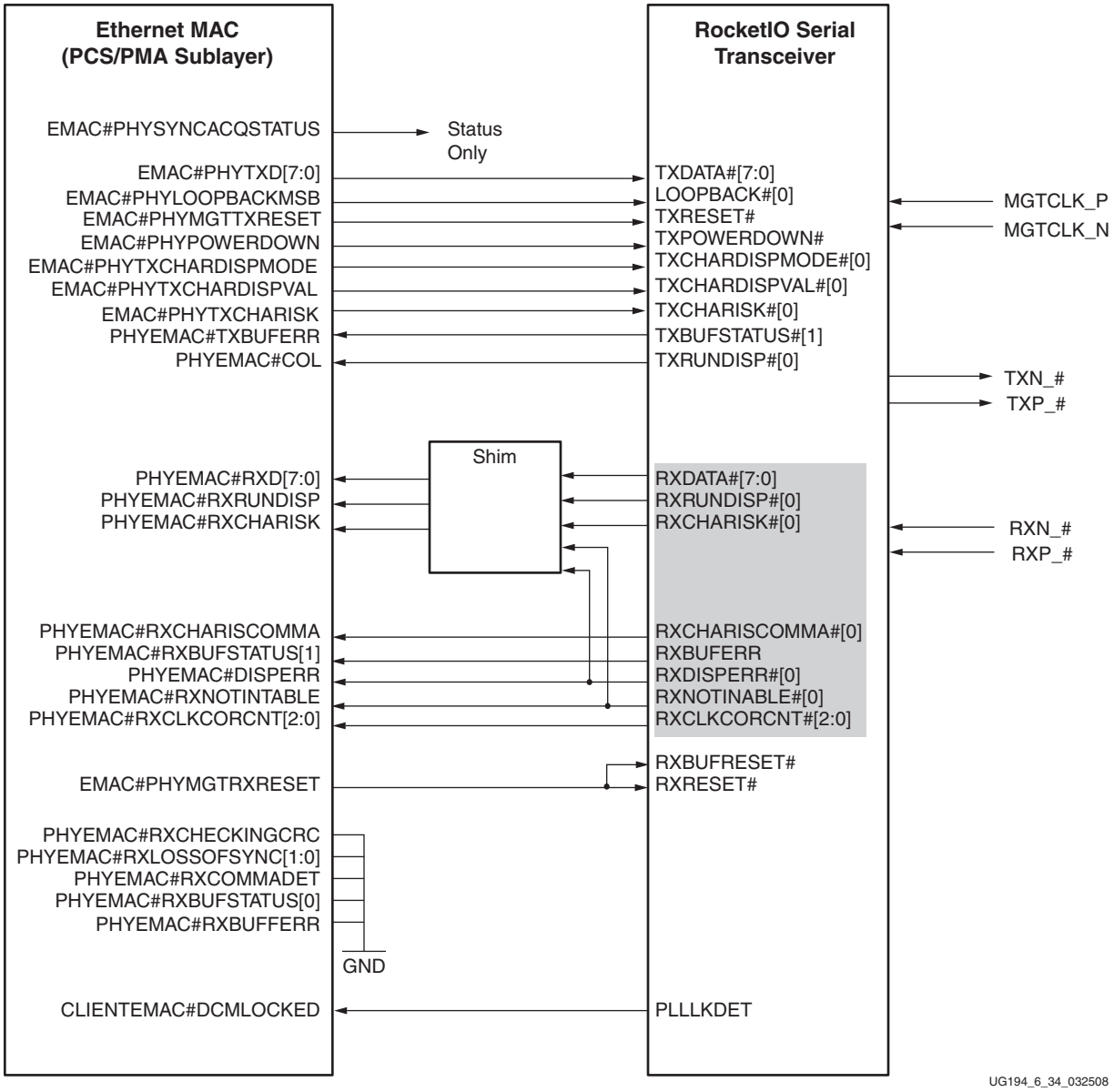


Figure 6-34: **Ethernet MAC Configured in SGMII Mode**

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. See "Accessing the Ethernet MAC from the CORE Generator Tool," page 25.

RocketIO Serial Transceiver Logic Using the RX Elastic Buffer in FPGA Logic

The example design, delivered with the core in the CORE Generator tool, is split between two different hierarchical layers. The block level is designed so that it can be instantiated directly into customer designs. It provides the following functionality:

- Instantiates the core from HDL
- Connects the physical-side interface of the core to a Virtex-5 FPGA RocketIO serial transceiver via the RX elastic buffer in FPGA logic

The Ethernet MAC is designed to integrate with the Virtex-5 FPGA RocketIO serial transceiver. The connections and logic required between the Ethernet MAC and RocketIO serial transceiver are illustrated in Figure 6-35. The signal names and logic shown in Figure 6-35 exactly match those delivered with the example design when the RocketIO serial transceiver is used.

A RocketIO tile consists of a pair of transceivers. For this reason, the RocketIO serial transceiver wrapper delivered with the core always contains two transceiver instantiations, even if only a single transceiver is in use. Figure 6-35 illustrates a single transceiver for clarity.

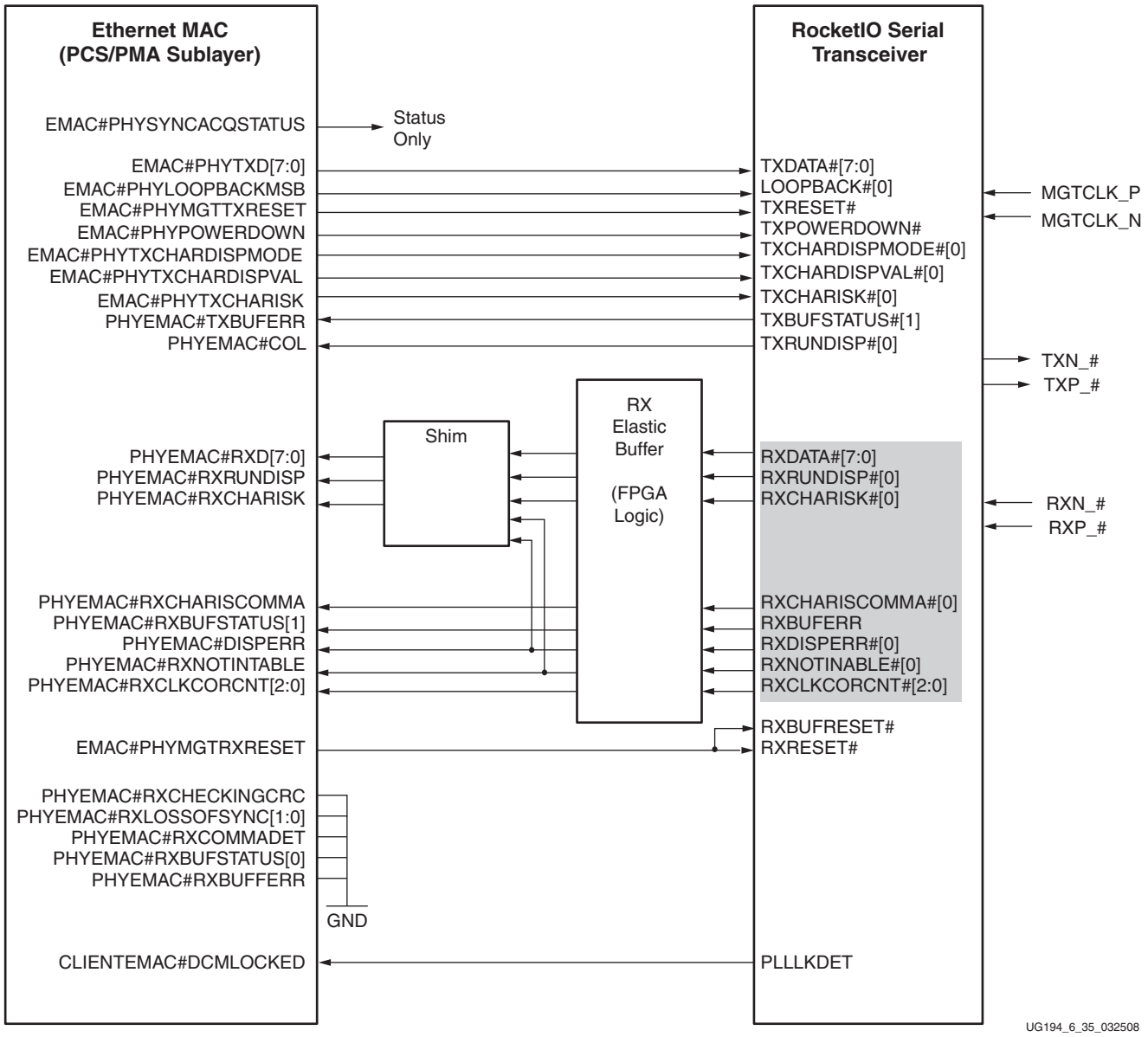


Figure 6-35: **SGMII Connection to a Virtex-5 FPGA RocketIO Serial Transceiver**

Figure 6-35 shows that the RX elastic buffer is implemented in the FPGA logic between the RocketIO serial transceiver and the Ethernet MAC instead of in the RocketIO serial transceiver. 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 RocketIO serial transceiver. This configuration, therefore, can handle larger frame sizes before clock tolerances accumulate and result in emptying or filling of the buffer. This is necessary for SGMII operation at 10 Mb/s, where each frame size is effectively 100 times larger than the same frame is at 1 Gb/s due to each byte being repeated 100 times.

In bypassing the RocketIO serial transceiver RX elastic buffer, data is clocked out of the RocketIO serial transceiver synchronously to RXRECCLK. This clock, placed on a BUFR component, is used to synchronize the transfer of data between the RocketIO serial transceiver and the RX elastic buffer as illustrated in Figure 6-37.

Shim

Because of differences in the way the Virtex-II Pro and Virtex-4 FPGA RocketIO transceivers and the Virtex-5 FPGA RocketIO serial transceivers output the undecoded 8B/10B code group when RXNOTINTABLE is asserted, a shim is needed. This shim modifies the received data from the Virtex-5 FPGA RocketIO serial transceiver to the format that the Ethernet MAC is expecting. Table 6-4 describes the functionality of this shim.

The shim created by the CORE Generator tool is combinatorial. If this logic is pipelined, all of the signals originating from the RocketIO serial transceiver highlighted with gray shading in Figure 6-34 must be pipelined with an equal latency.

Table 6-4: **Shim Functionality**

Ethernet MAC Port Name	Connection to RocketIO Serial Transceiver Port Name	
	When RXNOTINTABLE#[0] = 0	When RXNOTINTABLE#[0] = 1
PHYEMAC#RXD[7]	RXDATA#[7]	RXDATA#[2]
PHYEMAC#RXD[6]	RXDATA#[6]	RXDATA#[3]
PHYEMAC#RXD[5]	RXDATA#[5]	RXDATA#[4]
PHYEMAC#RXD[4]	RXDATA#[4]	RXDATA#[5]
PHYEMAC#RXD[3]	RXDATA#[3]	RXDATA#[6]
PHYEMAC#RXD[2]	RXDATA#[2]	RXDATA#[7]
PHYEMAC#RXD[1]	RXDATA#[1]	RXCHARISK#[0]
PHYEMAC#RXD[0]	RXDATA#[0]	RXDISPERR#[0]
PHYEMAC#RXRUNDISP	RXRUNDISP#[0]	RXDATA#[1]
PHYEMAC#RXCHARISK	RXCHARISK#[0]	RXDATA#[0]

SGMII Clock Management (LXT and SXT Devices)

Tri-Speed Operation

Figure 6-36 shows the clock management used with the SGMII interface without the FPGA logic elastic buffer. The CLKIN inputs to the RocketIO serial transceiver must be connected to an external, high-quality differential reference clock of frequency of 125 MHz. A 125 MHz clock source is then provided to the FPGA logic from the REFCLKOUT output port of the RocketIO serial transceiver. This is connected to global clock routing using a BUFG as illustrated (Figure 6-36). This clock should then be routed to the PHYEMAC#GTXCLK of the Ethernet MAC.

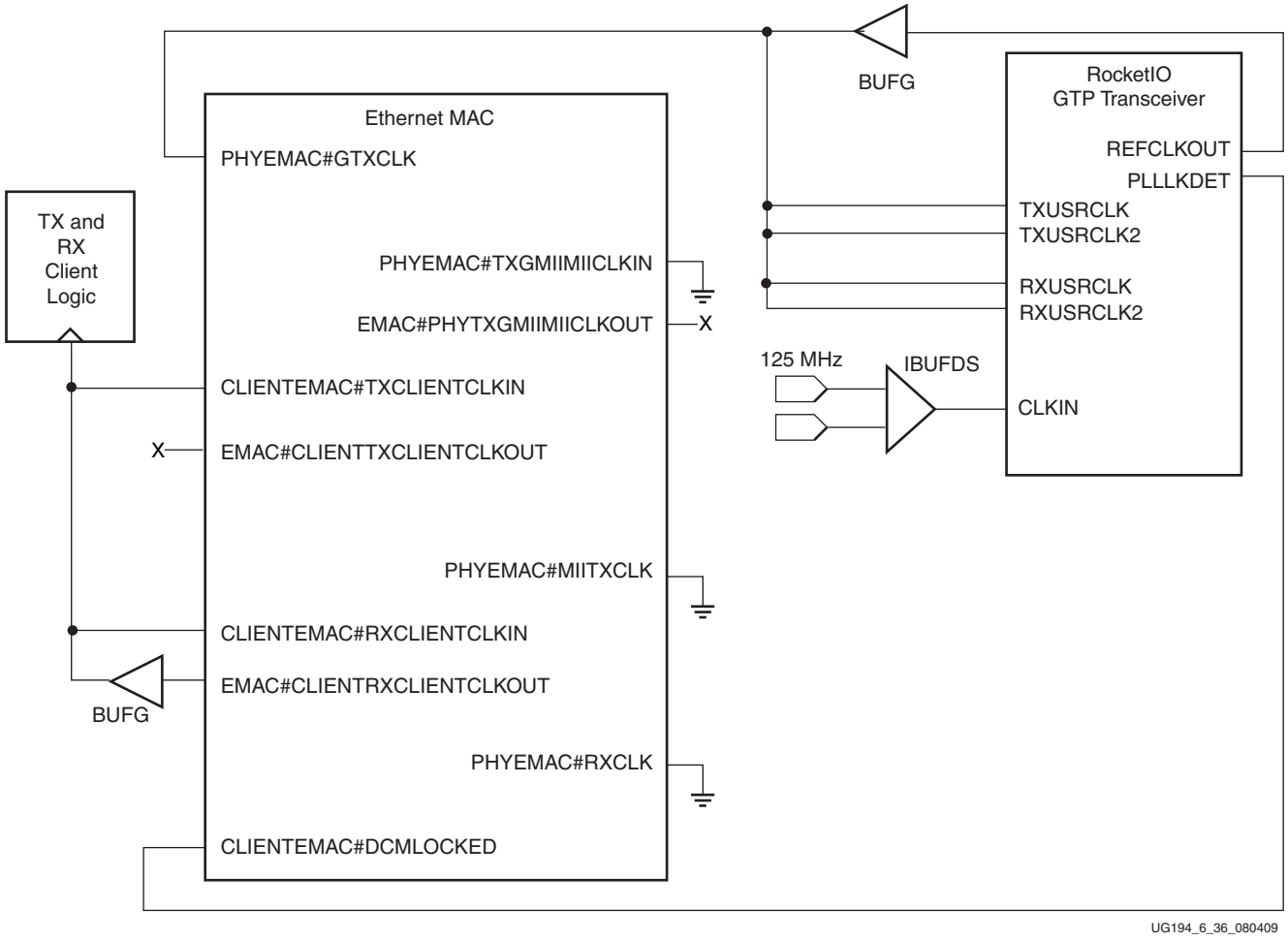


Figure 6-36: **SGMII Clock Management - RocketIO Serial Transceiver RX Elastic Buffer**

In bypassing the RocketIO serial transceiver RX elastic buffer, data is clocked out of the RocketIO serial transceiver synchronously to RXRECCLK. This clock, placed on a BUFR component, drives RXUSRCLK and RXUSRCLK2 and some of the logic in the RX elastic buffer as illustrated in Figure 6-37.

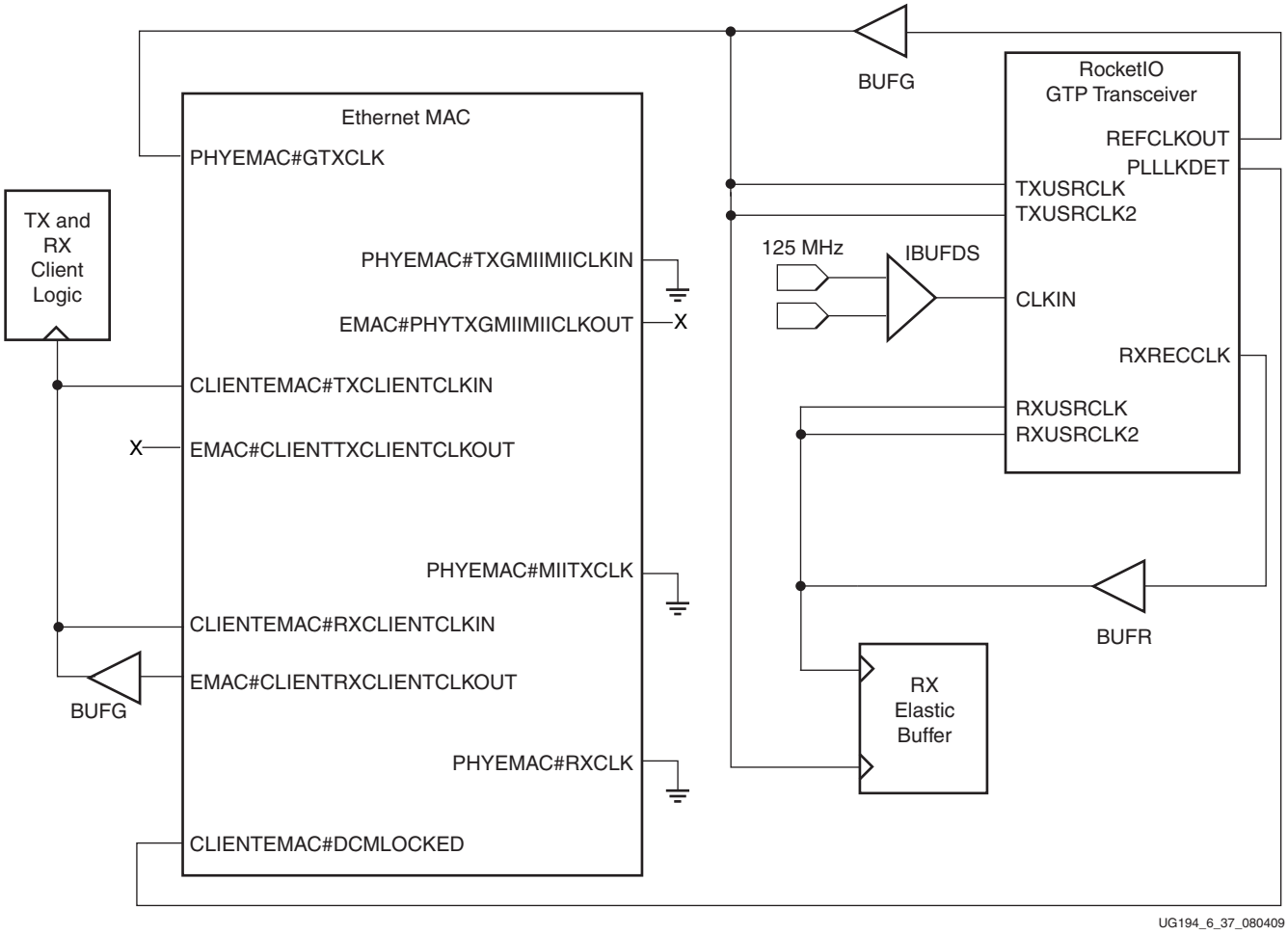


Figure 6-37: **SGMII Clock Management - FPGA Logic RX Elastic Buffer**

In both configurations, the PLLLKDET signal from the RocketIO serial transceiver (indicating that its internal PLLs have locked) is routed to the CLIENTEMAC#DCMLOCKED input port of the Ethernet MAC, which ensures that the state machines of the Ethernet MAC are held in reset until the RocketIO serial transceiver has locked and its clocks are running cleanly.

Either of the EMAC#CLIENTRXCLIENTCLKOUT or EMAC#CLIENTTXCLIENTCLKOUT output ports can be used to obtain the clock used for the Ethernet MAC client logic (both receiver and transmitter logic can share the same clock) with the unused clock being left unconnected. EMAC#CLIENTRXCLIENTCLKOUT is used in the example and is connected to a BUFG, which then provides the client clock to FPGA logic. It must also be routed back to the Ethernet MAC through the CLIENTEMAC#RXCLIENTCLKIN and CLIENTEMAC#TXCLIENTCLKIN input ports.

As described in "Ethernet MAC Clocks," page 205, the following clock signals are unused:

- EMAC#PHYTXGMIIMIICLKOUT
- EMAC#CLIENTTXCLIENTCLKOUT
- PHYEMAC#TXGMIIMIICLKIN
- PHYEMAC#MIITXCLK

- PHYEMAC#RXCLK

The outputs can be left unconnected, and the inputs should be tied Low.

1 Gb/s Only Operation

When the SGMII is used only for 1 Gb/s speeds, further clocking optimization can be performed. The clock logic is then identical to that described in "1000BASE-X PCS/PMA Clock Management (LXT and SXT Devices)," page 169.

SGMII Clock Management (TXT and FXT Devices)

Tri-Speed Operation

Figure 6-38 shows the clock management used with the SGMII interface without the FPGA logic elastic buffer in a TXT and FXT device. The CLKIN inputs to the RocketIO serial transceiver must be connected to an external, high-quality, differential reference clock with a frequency of 125 MHz. A 125 MHz clock source is then provided to the FPGA logic from the REFCLKOUT output port of the RocketIO serial transceiver. This is connected to the CLK0 input of a DCM. The CLK0 output of the DCM is connected to global clock routing using a BUFG, as illustrated in Figure 6-38. This clock should then be routed to the PHYEMAC#GTXCLK of the Ethernet MAC and the USRCLK2 inputs of the GTX transceiver. A second, 62.5 MHz, clock is output from the DCM on the CLKDV port. This should be used to clock the TXUSRCLK and RXUSRCLK inputs of the GTX transceiver.

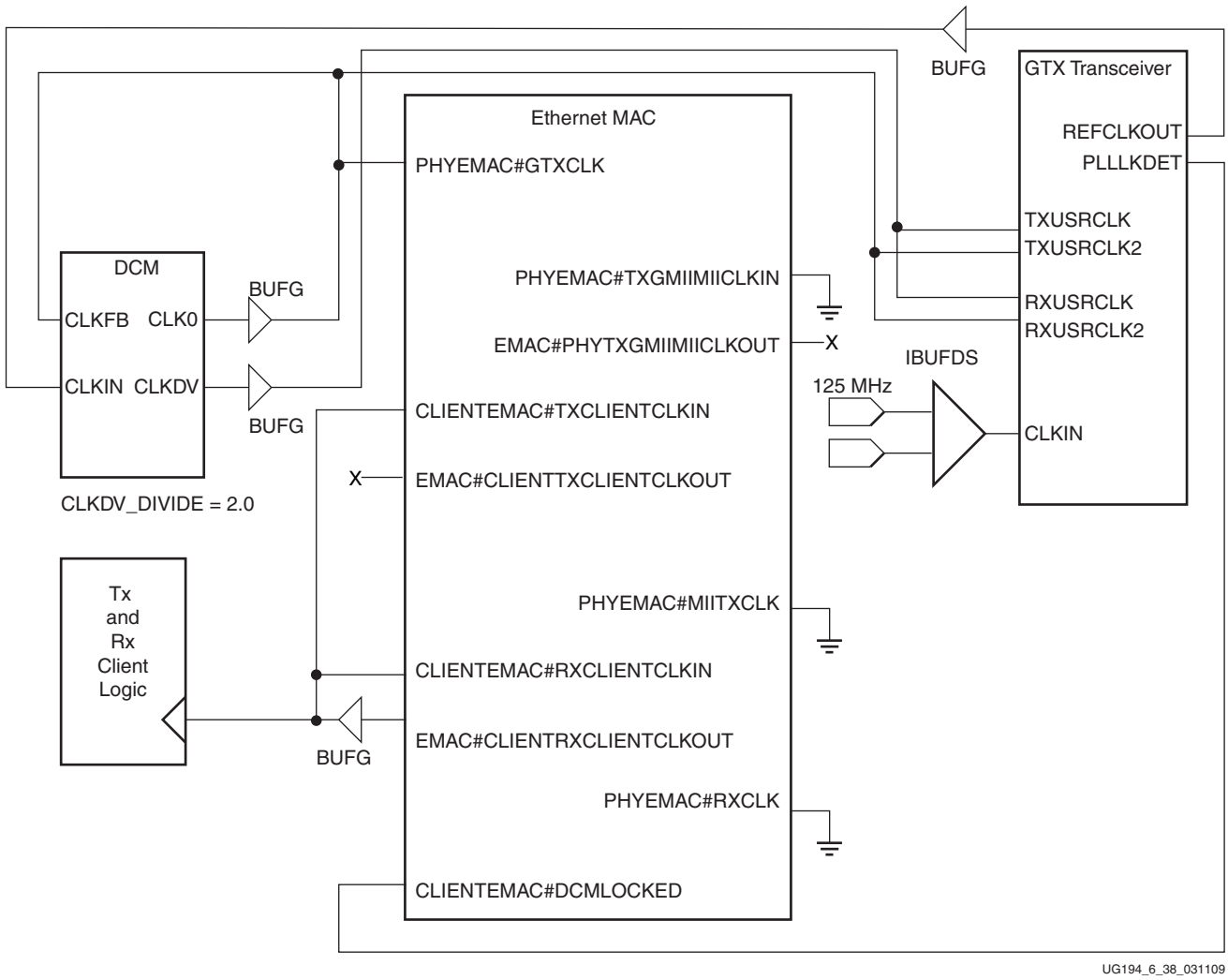


Figure 6-38: **SGMII Clock Management - RocketIO Serial Transceiver RX Elastic Buffer in a TXT and FXT Device**

In bypassing the RocketIO serial transceiver RX elastic buffer, data is clocked out of the RocketIO serial transceiver synchronously to RXRECCLK. This clock, placed on a BUFR component, drives RXUSRCLK, RXUSRCLK2, and some of the logic in the RX elastic buffer, as illustrated in Figure 6-39.

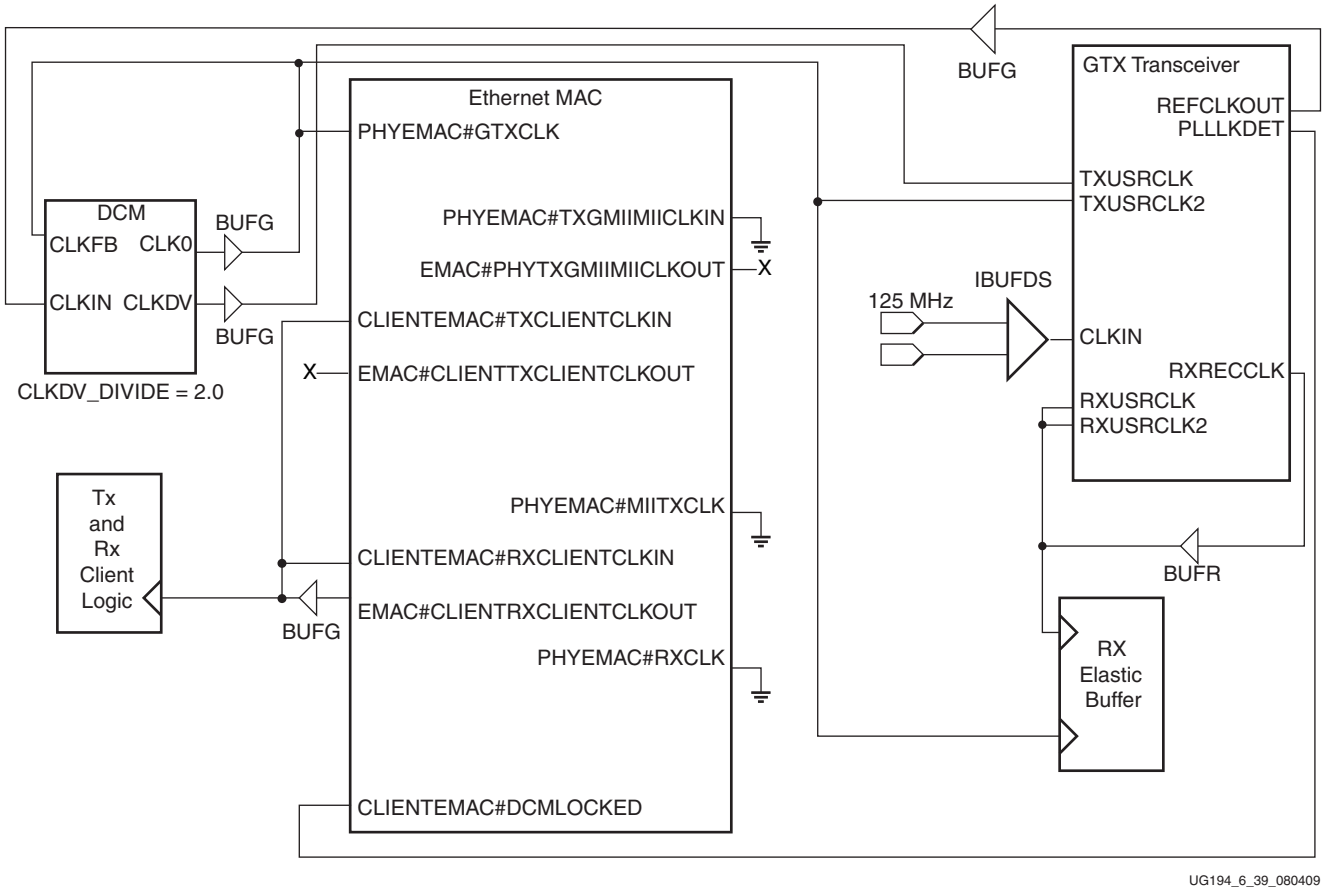


Figure 6-39: Bypassing the RocketIO Serial Transceiver RX Elastic Buffer

In both configurations, the PLLLKDET signal from the RocketIO serial transceiver (indicating that its internal PLLs have locked) is routed to the CLIENTEMAC#DCMLOCKED input port of the Ethernet MAC, which ensures that the state machines of the Ethernet MAC are held in reset until the RocketIO serial transceiver has locked and its clocks are running cleanly. Either of the EMAC#CLIENTRXCLIENTCLKOUT or EMAC#CLIENTTXCLIENTCLKOUT output ports can be used to obtain the clock used for the Ethernet MAC client logic (both receiver and transmitter logic can share the same clock) with the unused clock being left unconnected. EMAC#CLIENTRXCLIENTCLKOUT is used in our example and is connected to a BUFG, which then provides the client clock to FPGA logic. It must also be routed back to the Ethernet MAC through the CLIENTEMAC#RXCLIENTCLKIN and CLIENTEMAC#TXCLIENTCLKIN input ports.

As described in Appendix B, "Ethernet MAC Clocks," the following clock signals are unused:

- EMAC#PHYTXGMIIMIICLKOUT
- EMAC#CLIENTTXCLIENTCLKOUT
- PHYEMAC#TXGMIIMIICLKIN
- PHYEMAC#MIITXCLK
- PHYEMAC#RXCLK

SGMII Auto-Negotiation

Overview of Operation

Figure 6-40 illustrates a simplified diagram of the Ethernet MAC instantiated within a Virtex-5 device. The only components shown are two of the PCS Management Registers, which are directly involved in the Auto-Negotiation process (see "SGMII Management Registers," page 130). The corresponding registers of the connected devices are also shown.

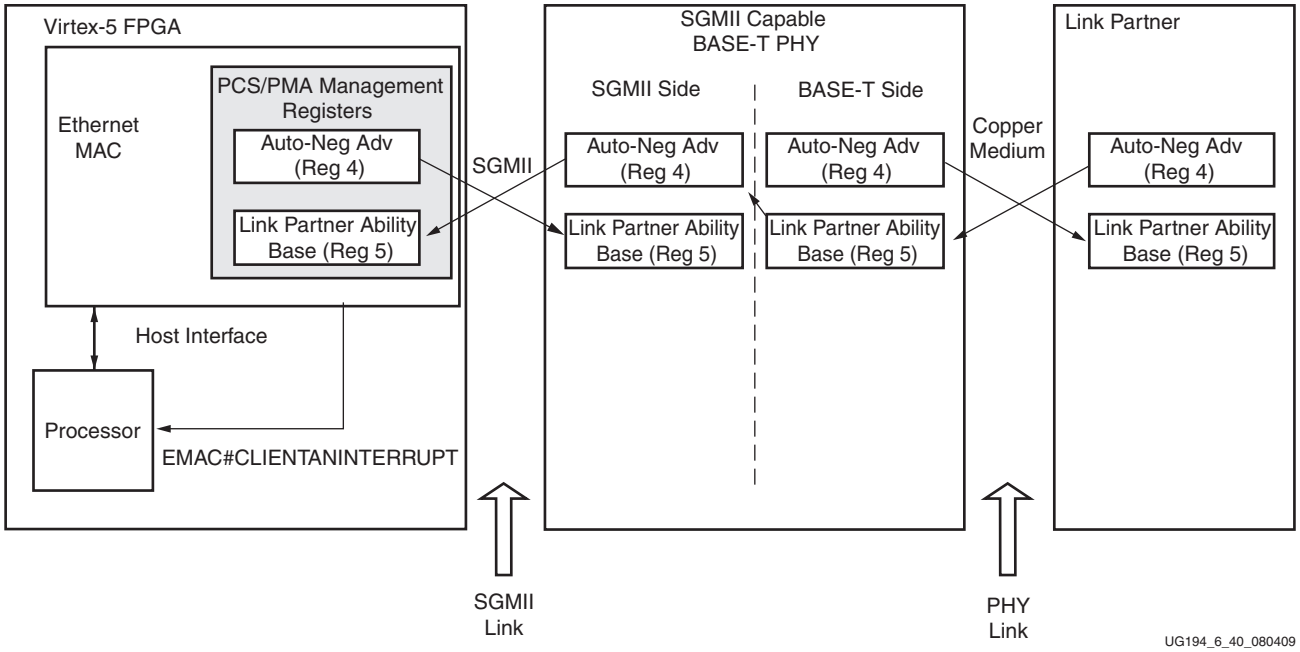


Figure 6-40: **SGMII Auto-Negotiation Overview**

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

- The PHY initially performs auto-negotiation with its link partner across the PHY link using the relevant Auto-Negotiation standard for the chosen medium (BASE-T auto-negotiation, illustrated in Figure 6-40, uses a copper medium). This resolves the operational speed and duplex mode with the link partner.
- The PHY then initiates a secondary Auto-Negotiation process with the Ethernet MAC across the SGMII link. This leverages the 1000BASE-X Auto-Negotiation specification described in "1000BASE-X Auto-Negotiation" with only minor differences. This transfers the results of the initial PHY with link partner auto-negotiation across the SGMII, and this is the only Auto-Negotiation process observed by the Ethernet MAC.

The SGMII Auto-Negotiation function leverages the "1000BASE-X 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").
- The information exchanged now contains speed resolution in addition to duplex mode.

Under normal conditions, this completes the Auto-Negotiation information exchange. The results can be read from the "SGMII Auto-Negotiation Link Partner Ability Base Register (Register 5)." The duplex mode and speed of the Ethernet MAC should then be configured by a software routine to match. This does not happen automatically within the Ethernet MAC. There are two methods by which a host processor can learn of the completion of an Auto-Negotiation cycle:

- By polling the Auto-Negotiation completion bit 1.5 in "SGMII Status Register (Register 1)," page 132.
- By using the Auto-Negotiation interrupt port (see "Auto-Negotiation Interrupt," page 197).

Auto-Negotiation Link Timer

The Auto-Negotiation Link Timer is used to time three phases of the Auto-Negotiation procedure. For the SGMII standard, this link timer is defined as having a duration of 1.6 ms.

The duration of the link timer used by the Ethernet MAC can be configured with the EMAC#_LINKTIMERVAL[8:0] attribute (see "Physical Interface Attributes," page 48). The duration of the timer is approximately equal to the binary value placed onto this attribute multiplied by 32.768 µs (4096 clock periods of the 125 MHz clock provided to the Ethernet MAC on `PHYEMAC#GTXCLK`). The accuracy of this link timer is within the range:

+0 to -32.768 µs

Therefore, for the SGMII standard, set the EMAC#_LINKTIMERVAL[8:0] attribute to:

`000110010` = 50 decimal

This corresponds to a timer duration of between 1.606 and 1.638 ms. This value can be reduced for simulation.

Auto-Negotiation Interrupt

The Auto-Negotiation function has an `EMAC#CLIENTANINTERRUPT` port. This port is designed to be used with common microprocessor bus architectures.

The operation of this port is enabled or disabled and cleared via the "SGMII Vendor-Specific Register: Auto-Negotiation Interrupt Control Register (Register 16)," page 135.

- 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 the "SGMII Vendor-Specific Register: Auto-Negotiation Interrupt Control Register (Register 16)."

Loopback When Using the PCS/PMA Sublayer

Figure 6-41 illustrates the loopback options when using the PCS/PMA sublayer. Two possible loopback positions are illustrated:

- **Loopback in the Ethernet MAC.** When placed into loopback, data is routed from the transmitter to the receiver path at the last possible point in the PCS/PMA sublayer, immediately before the RocketIO serial transceiver interface. When placed into loopback, a constant stream of Idle code groups are transmitted through the RocketIO serial transceiver.

Loopback in this position allows test frames to be looped back within the Ethernet MAC without allowing them to be received by the link partner. The transmission of idles allows the link partner to remain in synchronization so that no fault is reported.

- **Loopback in the RocketIO serial transceiver.** The RocketIO serial transceiver can alternatively be switched into loopback by connecting the `EMAC#PHYLOOPBACKMSB` port of the Ethernet MAC to the loopback port of the RocketIO serial transceiver as illustrated. The RocketIO serial loopback routes data from the transmitter path to the receiver path within the RocketIO serial transceiver. However, this data is also transmitted out of the RocketIO serial transceiver, so any test frames used for a loopback test are received by the link partner.

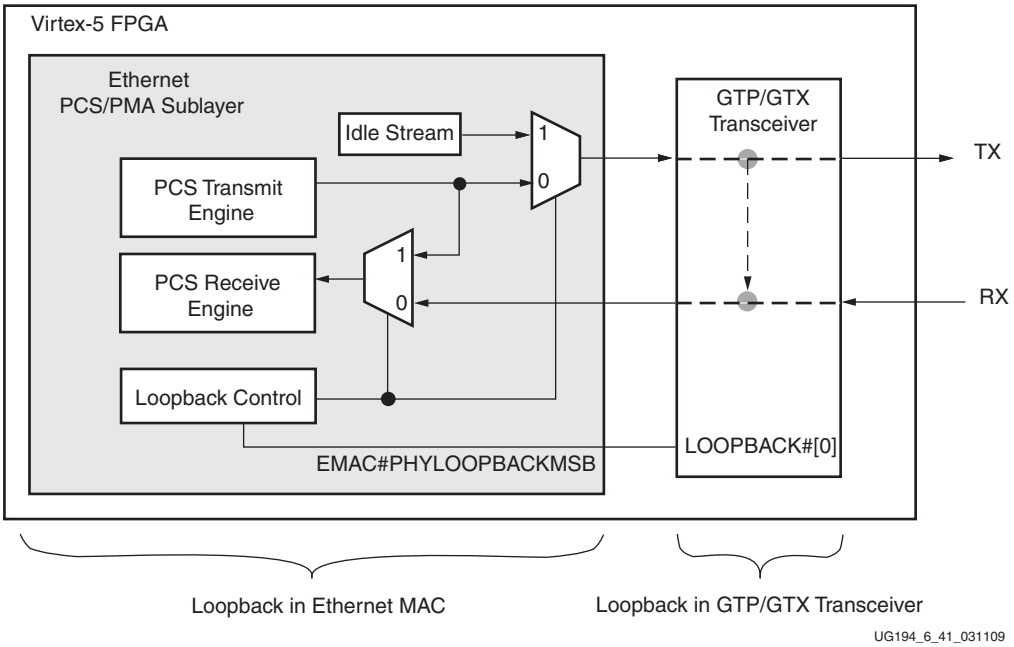


Figure 6-41: **Loopback When Using the PCS/PMA Sublayer**

Loopback itself can be enabled or disabled by writing to the PCS/PMA Sublayer Control Register (Register 0). The loopback position can be controlled through the Loopback Control Register (Register 17). See "SGMII Management Registers," page 130 for details. Alternatively, loopback can be controlled by setting the EMAC#_PHYLOOPBACKMSB and EMAC#_GTLOOPBACK attributes. See Chapter 2 for details.

XILINX®

Chapter 7

Interfacing to a Statistics Block

To collect statistics information from an individual Ethernet MAC, a custom statistics counter can be implemented in the FPGA logic. A parameterizable Ethernet Statistics LogiCORE™ solution that fulfills all the requirements for statistics information collection is available through the CORE Generator ™tool (DS323, *LogiCORE Ethernet Statistics Data Sheet*, which provides a full description of the Ethernet Statistics LogiCORE block). Each Ethernet MAC that is used requires its own instance of this Statistics IP, as shown in Figure 2-1, page 27.

This chapter describes how to connect the Ethernet MAC block to a Statistics block to allow the statistics registers of each Ethernet MAC to be accessed by either the host or DCR bus interface. A description of the address space that can be used for the statistics registers is provided to avoid register access contentions.

"Statistics Vectors," page 78 provides details on the statistics interface of the Ethernet MAC and how the output signals encode the statistics vector information.

Using the Host Bus to Access Statistics Registers

When the Ethernet MAC is used with the host bus, interfacing to an FPGA logic-based statistics block is straight-forward. The statistics values can be read via a host interface that is 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 accesses are avoided by only accessing statistics counters when the `HOSTMIIMSEL` signal is at logic 0. Implementation of the addressing scheme shown in Table 7-1 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 7-1: **Addressing Scheme**

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

Figure 7-1 shows how to integrate the Ethernet MAC with the LogiCORE Ethernet Statistics block, where the Ethernet statistics counters are accessed via the host bus.

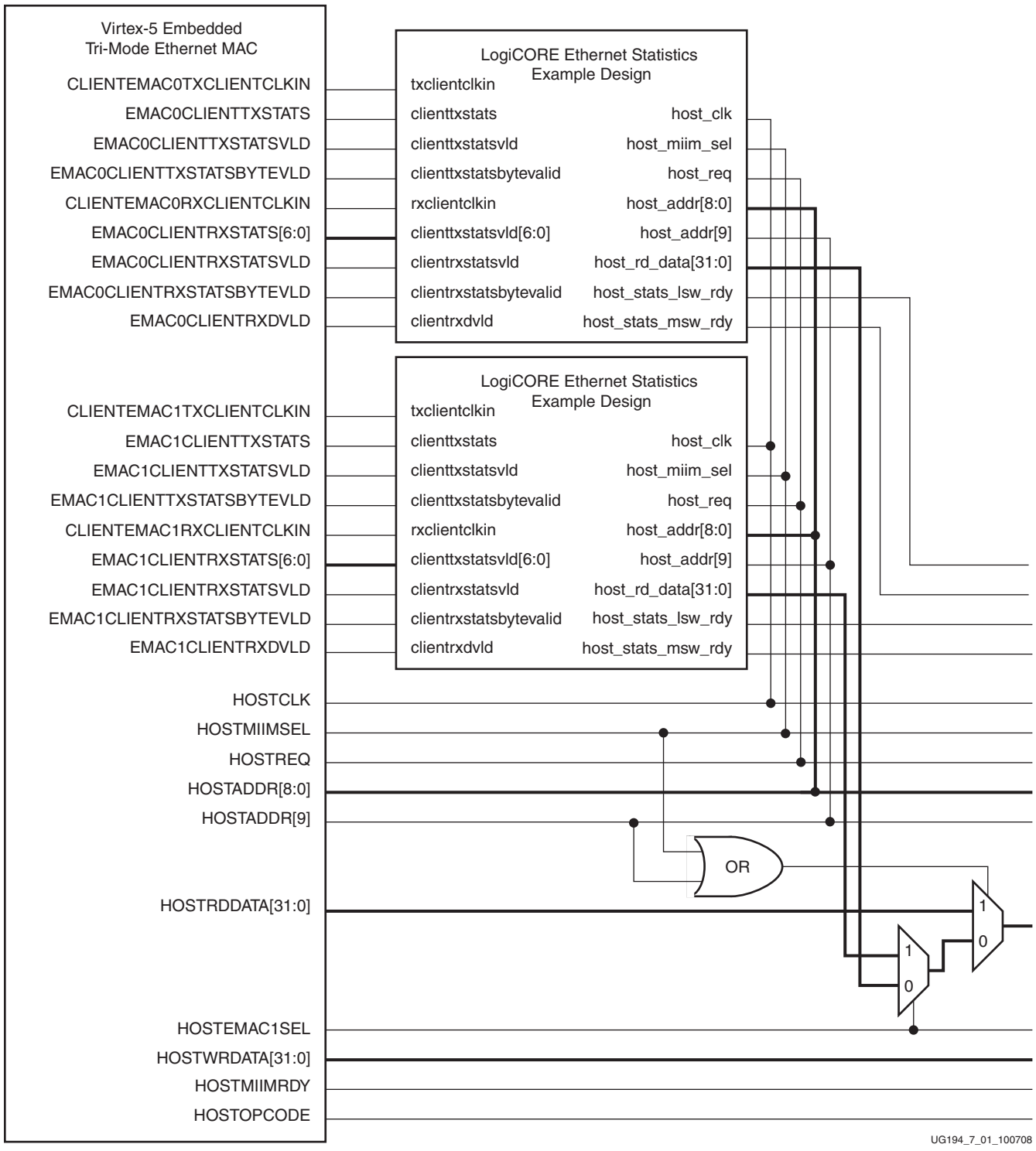


Figure 7-1: **Host Bus to Ethernet Statistics Connection**

The LogiCORE Ethernet Statistics block is used with the addressing scheme shown in Table 7-1. Figure 7-1 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.

Using the DCR Bus to Access Statistics Registers

When the DCR bus interface of the Ethernet MAC is controlled by an embedded processor, the host interface and DCR bridge of the Ethernet MAC allow DCR bus accesses to control the unused host bus pins. The Ethernet MAC host bus interface can then access the statistics counters in the FPGA logic, controlled by register accesses on the DCR bus.

The host bus I/O signals of the Ethernet MAC are enabled for statistics counter access when a DCR read operation is made to address codes `0x000` to `0x02F` and `0x040` to `0x04F` inclusive (Table 4-15, page 103 describes the DCR address code space). This use of the host bus I/O signals provides a means of accessing the FPGA logic 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 4-23 and Figure 4-10. These signals can access statistics counters in the same way as if a stand-alone 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 4-10. The data read from the host bus can then be accessed by reading the DCR data registers.

Figure 7-2 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 full description of the Ethernet Statistics LogiCORE block. Figure 7-2 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.

An example of how to read from the statistics counters through the DCR bus is provided in the example in "Accessing FPGA Logic via Unused Host Bus Pins," page 113.

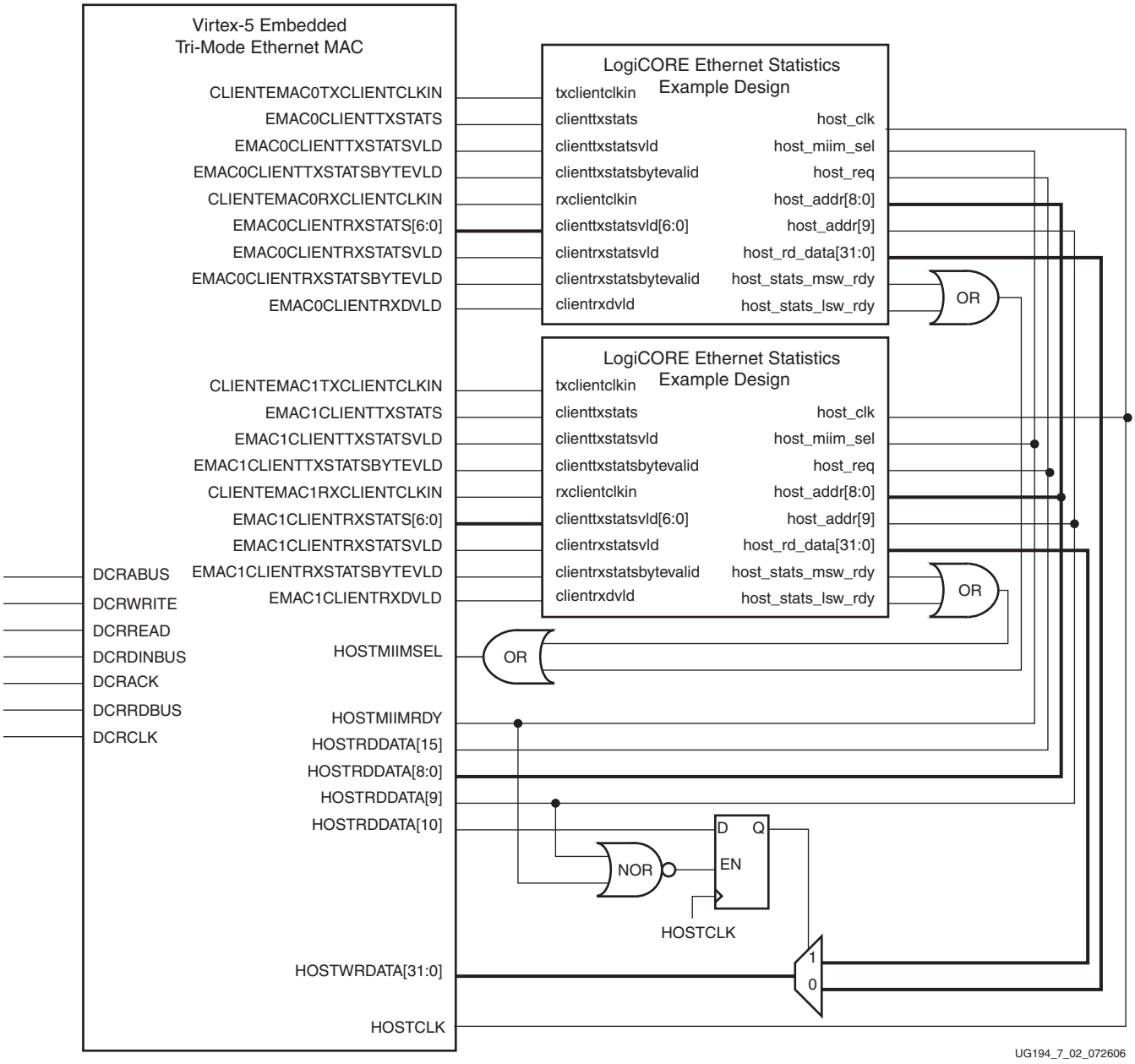


Figure 7-2: **DCR Bus to Ethernet Statistics Connection**

XILINX®

Appendix A

Pinout Guidelines

Xilinx recommends the following guidelines to improve design timing using the Virtex®-5 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 Ethernet MAC block.
- Use the RocketIO™ serial transceivers located closest to the Ethernet MAC block.

Appendix B

Ethernet MAC Clocks

This appendix provides an overview of the Ethernet MAC clocking schemes. The clocking schemes that are internal to the Ethernet MAC block are introduced, followed by the clock connections to and from the FPGA logic. Finally, all clock input and output frequencies and definitions are listed for reference, dependent on the mode of operation.

Ethernet MAC Internal Clock Logic Overview

The large number of clock signals can give an overwhelming first impression. Figure B-1 is a simplified diagram to illustrate all Ethernet MAC input and output clock signals.

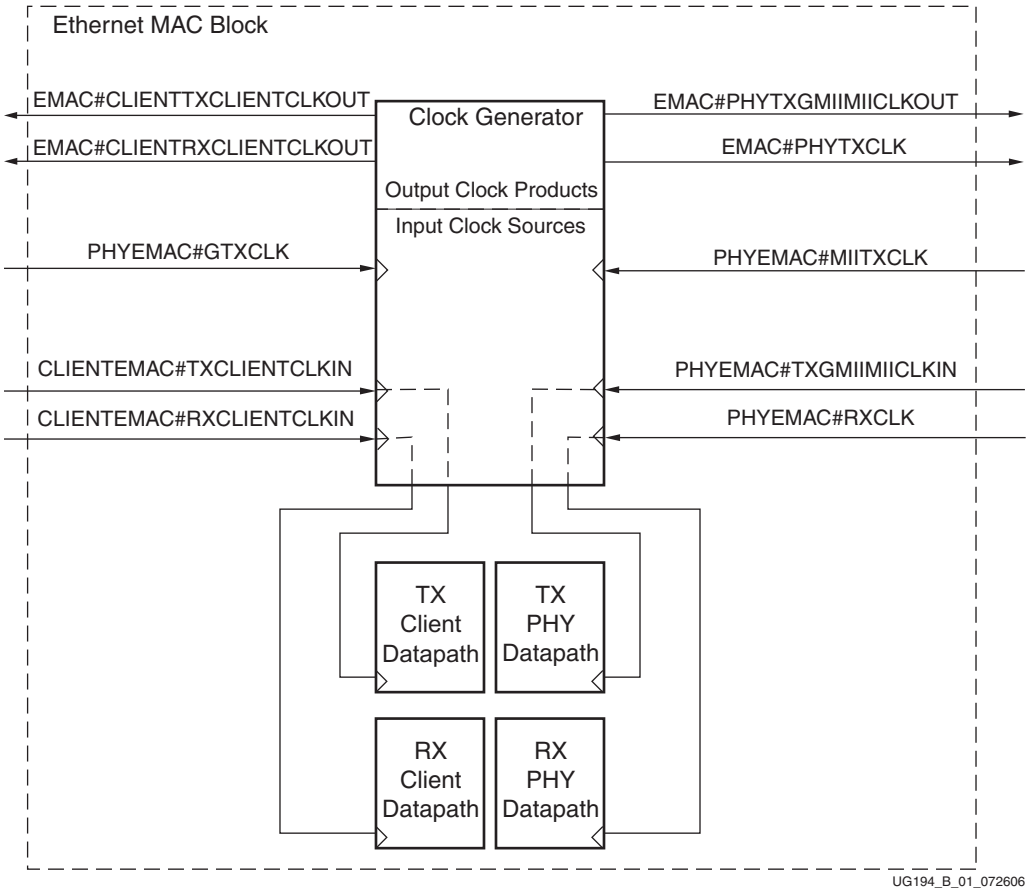


Figure B-1: **Overview of Ethernet MAC Clock Circuitry**

Ethernet MAC Clock Generation

As shown in Figure B-1, included in the Ethernet MAC is a Clock Generator module. This module is provided with input clock sources, from which the following output clock products are generated:

- EMAC#PHYTXGMIIMIICLKOUT
- EMAC#PHYTXCLK
- EMAC#CLIENTTXCLIENTCLKOUT
- EMAC#CLIENTRXCLIENTCLKOUT

For these generated output clocks (refer to "Clock Definitions and Frequencies"):

- the output clocks are always at the correct frequency for their associated interfaces for any mode of operation and switch frequencies cleanly during speed changes
- the output clocks are NOT used internally by any Ethernet MAC logic
- the output clocks are provided only for the convenience of the user; they do not have to be used by the FPGA logic

Ethernet MAC Input Clocks

As shown in Figure B-1, the following clock signals are input into the Ethernet MAC clock generator:

- PHYEMAC#GTXCLK
- PHYEMAC#MIITXCLK
- PHYEMAC#RXCLK
- PHYEMAC#TXGMIIMIICLKIN
- CLIENTEMAC#RXCLIENTCLKIN
- CLIENTEMAC#TXCLIENTCLKIN

Not all clock input sources are required in all modes. Required clock input signals must always be driven by the FPGA logic at the correct frequency for the required mode of operation (see "Clock Definitions and Frequencies").

Only the input clocks are collectively responsible for the correct operation of the Ethernet MAC. It is possible to ignore all clock generator output clocks for correct Ethernet MAC operation, providing the input clocks are alternatively derived. However, the clock generator output clocks are provided precisely for this purpose.

Clock Connections to and from FPGA Logic

The clock logic described here is not optimized for specific user modes. There are cases where two or more clock generator input clocks can be shared from a common source. This can be a common client clock, shared between transmitter and receiver; in other cases, this can be a common transmitter clock shared between both client and physical domains. It is also possible to share common clocks between two of more Ethernet MACs. Chapter 6, "Physical Interface," provides clock management information for optimal clock logic efficiency based on the chosen physical interfaces.

This section provides general-purpose clock logic descriptions to aid in the understanding of the Ethernet MAC, and clock sharing is not considered. Three main clocking modes are to be discussed:

- "Standard Clocking Scheme"
- Advanced Clocking Scheme: "Clock Enables"
- Advanced Clocking Scheme: "Byte PHY"

Standard Clocking Scheme

The Virtex®-5 FPGA Ethernet MAC standard clocking scheme, as shown in Figure B-2, is identical to the standard clocking scheme used by the Virtex-4 FPGA Embedded Tri-Mode Ethernet MAC.

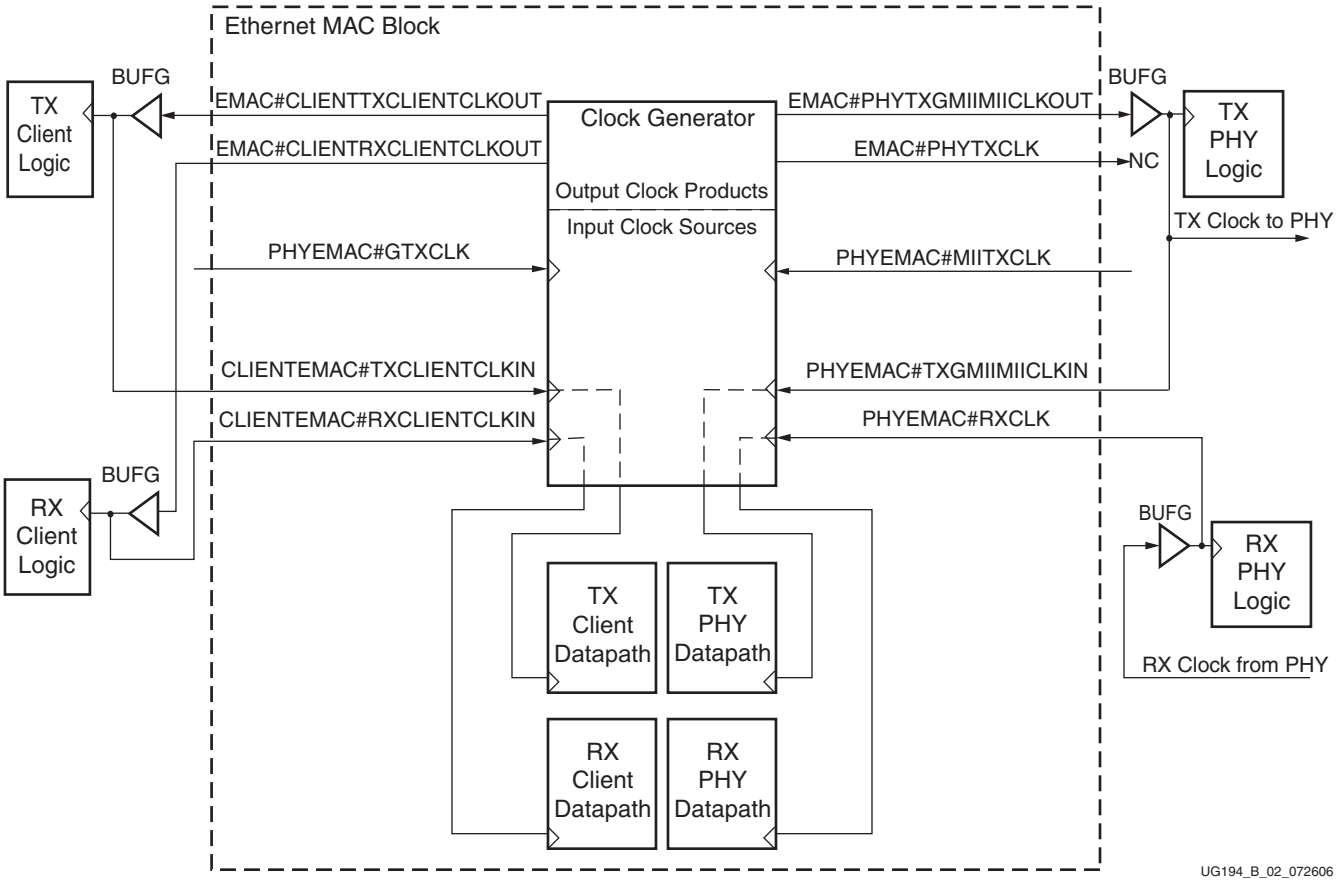


Figure B-2: **Overview of Ethernet MAC Clock Connections to and from the FPGA Logic**

Figure B-2 shows clock logic instantiated in the FPGA logic using the standard method and shows the general case (non-optimized) clocking scheme.

Figure B-2 and the text that follows describes how the clock signals required by the Ethernet MAC can be derived from the output clocks from the clock generator.

Transmitter Client Clock

`EMAC#CLIENTTXCLIENTCLKOUT` can be placed onto global clock routing and used to clock all of the transmitter client logic. This resultant clock is then fed back into the Ethernet MAC on the `CLIENTEMAC#TXCLIENTCLKIN`, where it is used to derive the internal clock used for the transmitter client datapath. This configuration has the effect of eliminating clock skew between the Ethernet MAC and the FPGA logic (caused by the global clock routing), enabling data to be reliably transferred between the two.

Receiver Client Clock

EMAC#CLIENTRXCLIENTCLKOUT can be placed onto global clock routing and used to clock all of the receiver client logic. This resultant clock is then fed back into the Ethernet MAC on the CLIENTEMAC#RXCLIENTCLKIN, where it is used to derive the internal clock used for the receiver client datapath. This configuration has the effect of eliminating clock skew between the Ethernet MAC and the FPGA logic (caused by the global clock routing), enabling data to be reliably transferred between the two.

Transmitter Physical Clock

EMAC#PHYTXGMIIMIICLKOUT can be placed onto global clock routing and used to clock all of the transmitter physical logic. This resultant clock is then fed back into the Ethernet MAC on the PHYEMAC#TXGMIIMIICLKIN, where it is used to derive the internal clock used for the transmitter physical datapath. This configuration has the effect of eliminating clock skew between the Ethernet MAC and the FPGA logic (caused by the global clock routing), enabling data to be reliably transferred between the two.

Receiver Physical Clock

The physical clock for the receiver interface is sourced by the connected PHY. When placed onto global clock routing, this resultant clock is also fed into the Ethernet MAC on the PHYEMAC#RXCLK port and is used to derive the internal clock used for the receiver physical datapath. This enables data to be reliably transferred from the FPGA logic into the Ethernet MAC.

Advanced Clocking Schemes

Two advanced clocking schemes are developed for the Virtex-5 FPGA Ethernet MAC. Both of these clocking schemes reduce the global clock usage in the FPGA logic.

- "Clock Enables" evolved from an FPGA logic enhancement for the Virtex-4 FPGA Ethernet MAC. The Virtex-4 FPGA clock enable signals were created in the FPGA logic, resulting in a clocking scheme that halved the global clock usage when using the MII physical interface for 10 Mb/s or 100 Mb/s.

  The Virtex-5 FPGA clock enables are provided by the Ethernet MAC to the FPGA logic, and their use extends to cover MII, GMII, and RGMII interfaces at all three Ethernet speeds.

- "Byte PHY" also evolved from an FPGA logic enhancement for the Virtex-4 FPGA Ethernet MAC and provided a solution that halved the global clock usage when using the GMII/MII physical interface at all three speeds. However, it supported only full-duplex mode.

  In Virtex-5 devices, the Byte PHY functionality extends to cover the GMII/MII, at all three speeds, supporting both full- and half-duplex modes.

Clock Enables and Byte PHY advanced clocking modes are offered by the CORE Generator™ tool for both the Virtex-4 and Virtex-5 FPGA Ethernet MACs. However, as previously described, these options are available for the Virtex-5 FPGA Ethernet MAC over a wider range of configurations.

Clock Enables

For GMII transmitter operation at 1 Gb/s, both client and physical interfaces have an 8-bit datapath operating at 125 MHz. If 1 Gb/s is the only speed of interest, a single clock domain can be used for both client and physical interfaces.

For MII transmitter operation at 100 Mb/s, the client interface still has an 8-bit datapath that operates at 12.5 MHz. However, the physical interface width dropped to 4 bits and is, therefore, clocked at twice the frequency of the client (25 MHz).

The Clock Enable approach always clocks the client interface with the physical interface clock, while also providing the client with a clock enable. At 1 Gb/s, the physical interface clock is 125 MHz, which is also required by the client; the clock enable is constantly held High. At 100 Mb/s, the physical interface clock is 25 MHz, which is twice that required by the client. The clock enable toggles on alternative 25 MHz clock cycles to enable the client logic at the resultant rate of 12.5 MHz.

This methodology is applied similarly to the receiver path. GMII/MII and RGMII physical interfaces can all use this clocking scheme at all three Ethernet speeds.

Figure B-3 shows clock logic instantiated in the FPGA logic when using the clock enables. Refer to "Clock Definitions and Frequencies," page 211 for relationships between clocks and clock enables). Figure B-3 and the text that follows describe how the clock signals required to use the Ethernet MAC can be derived from the output clocks (and clock enables) from the Ethernet MACs clock generator.

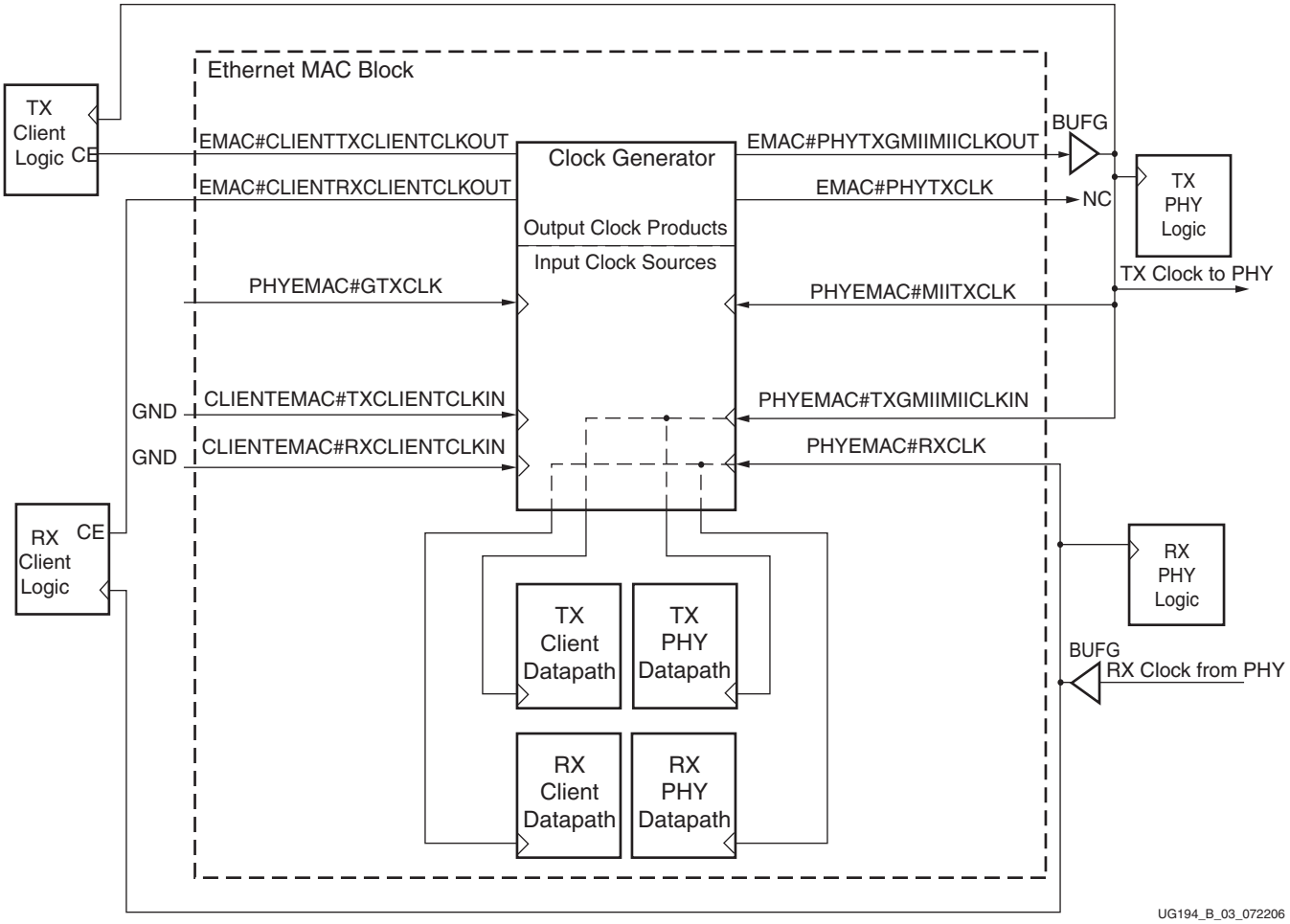


Figure B-3: **Overview of Ethernet MAC Clock Connections to and from the FPGA Logic using Clock Enables**

Note: There are some complications in the use of the `EMAC#CLIENTTXACK` signal when using this Clock Enable scheme. Refer to the specific physical interface section in Chapter 6 for details.

Transmitter Clock

`EMAC#PHYTXGMIIMIICLKOUT` can be placed onto global clock routing and used to clock all of the transmitter physical logic. This resultant clock is then fed back into the Ethernet MAC on the `PHYEMAC#TXGMIIMIICLKIN`, where it is used to derive the internal clock used for the entire transmitter datapath (both physical and client). This has the effect of eliminating clock skew between the Ethernet MAC and the FPGA logic (caused by the global clock routing), enabling data to be reliably transferred between the two.

`EMAC#CLIENTTXCLIENTCLKOUT` is no longer used as a clock but instead as a clock enable. Every flip-flop of the transmitter client logic can be clocked with the clock connected to `PHYEMAC#TXGMIIMIICLKIN`, providing it has its clock enable connected as illustrated. This configuration saves global clock resources when compared with the non clock enable approach.

Receiver Clock

The Physical clock for the receiver interface is sourced by the connected PHY. When placed onto global clock routing, this resultant clock is also fed into the Ethernet MAC on the PHYEMAC#RXCLK port. This is used to derive the internal clock used for the receiver datapath (both physical and client). This enables data to be reliably transferred from the FPGA logic into the Ethernet MAC.

EMAC#CLIENTRXCLIENTCLKOUT is no longer used as a clock but instead as a clock enable. Every flip-flop of the receiver client logic can be clocked with the clock connected to PHYEMAC#RXCLK, providing it has its clock enable connected as illustrated. This configuration saves global clock resources when compared with the non-clock enable approach.

Byte PHY

For GMII transmitter operation at 1 Gb/s, both client and physical interfaces have an 8-bit datapath operating at 125 MHz. If 1 Gb/s is the only speed of interest, a single clock domain can be used for both client and physical interfaces.

For MII transmitter operation at 100 Mb/s, the client interface still has an 8-bit datapath, that now operates at 12.5 MHz. However, in the standard operating mode, the physical interface width drops to 4 bits, requiring a clock at twice the frequency (25 MHz).

In Byte PHY mode, the Ethernet MAC always outputs the physical interface as an 8-bit datapath, regardless of operating speed. Therefore, at 100 Mb/s, the data is output as an 8-bit datapath at a frequency of 12.5 MHz. This enables the client and physical interface to share the same clock domain. The same situation occurs also for 10 Mb/s with a shared clock of frequency 1.25 MHz.

When performing GMII at 1 Gb/s, the physical interface datapath is the correct width at 8 bits. When performing MII at 10/100 Mb/s, the physical interface datapath is of an incorrect width. The datapath has to be converted from eight bits to the correct width of four bits using the DDR output registers in the IOBs.

This methodology is applied similarly to the receiver path to provide a GMII/MII compatible physical interface.

Please refer to "GMII Clock Management for Tri-Speed Operation Using Byte PHY," page 149 for further information and for the specific clocking diagram.

Clock Definitions and Frequencies

The following sections provide definitions for all of the Ethernet MAC input and output clocks, which are often mode dependent. Specific clocking schemes, which are optimized for the chosen physical interface, are provided in Chapter 6, "Physical Interface." Chapter 6 provides useful reference material when studying the following sections.

PHYEMAC#GTXCLK

MII Only Mode (10/100 Mb/s Operation)

This clock signal is unused and can be connected to logic 0.

All Other Modes

All transmitter clocks, both client and physical, are derived from this clock source with one exception: it is not used for 10/100 Mb/s operation when using the MII. For all other modes, PHYEMAC#GTXCLK is required and should be connected to a 125 MHz reference clock source, which is within the IEEE Std 802.3 specification (100 ppm).

Table B-1: **PHYEMAC#GTXCLK Clock Frequencies**

Clock Signal	Direction	Operating Speed		
		1000 Mb/s	100 Mb/s	10 Mb/s
PHYEMAC#GTXCLK	Input	125 MHz	125 MHz	125 MHz

PHYEMAC#MIITXCLK

MII or Tri-Speed GMII Modes

PHYEMAC#MIITXCLK is used as the transmitter reference clock source when using the MII, where this clock is provided by the attached external PHY device.

Table B-2: **PHYEMAC#MIITXCLK Clock Frequencies**

Clock Signal	Direction	Operating Speed		
		1000 Mb/s	100 Mb/s	10 Mb/s
PHYEMAC#MIITXCLK	Input	n/a	25 MHz	2.5 MHz

1000BASE-X PCS/PMA (16-Bit Data Client) Mode

When operating in 1000BASE-X PCS/PMA (see "16-Bit Data Client"), this clock input should be connected to a clock source that is half the frequency of the clock input into the CLIENTEMAC#TXCLIENTCLKIN port.

All Other Modes

This clock signal is unused and can be connected to logic 0.

PHYEMAC#RXCLK

GMII (non-Byte PHY), MII, and RGMII Modes

The clock signal input into PHYEMAC#RXCLK is sourced by the external PHY device and should be used by the FPGA logic to clock the receiver physical interface logic. Internally in the Ethernet MAC, this clock is used to derive all physical and client receiver clocks.

Table B-3: **PHYEMAC#RXCLK Clock Frequencies**

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

GMII (Byte PHY) Mode

The clock signal input into PHYEMAC#RXCLK is sourced by the external PHY device at 1 Gb/s. At 10/100 Mb/s speeds, this clock should also originate from the external PHY device but its frequency is divided by two in the FPGA logic (see "GMII Clock Management for Tri-Speed Operation Using Byte PHY"). The resultant clock should be used by the FPGA logic to clock the receiver client and physical interface logic. Internally in the Ethernet MAC, this clock is used to derive all physical and client receiver clocks.

Table B-4: **PHYEMAC#RXCLK Clock Frequencies**

Clock Signal	Direction	Operating Speed		
		1000 Mb/s	100 Mb/s	10 Mb/s
PHYEMAC#RXCLK	Input	125 MHz	12.5 MHz	1.25 MHz

SGMII and 1000BASE-X PCS/PMA (8-Bit Data Client) Modes

This clock signal is unused and can be connected to logic 0.

1000BASE-X PCS/PMA (16-Bit Data Client) Mode

When operating in 1000BASE-X PCS/PMA (see "16-Bit Data Client"), this clock input should be connected to a clock source that is half the frequency of the clock input into the PHYEMAC#GTXCLK port.

EMAC#PHYTXGMIIMIICLKOUT, PHYEMAC#TXGMIIMIICLKIN

MII, GMII (non-Byte PHY), and RGMII Modes

EMAC#PHYTXGMIIMIICLKOUT is created by the clock generator to provide the frequencies defined in Table B-5.

The clock signal input into the PHYEMAC#TXGMIIMIICLKIN port should be used to clock the transmitter physical interface logic. This can be derived from EMAC#PHYTXGMIIMIICLKOUT as illustrated in Figure B-2 and Figure B-3.

Table B-5: **EMAC#PHYTXGMIIMIICLKOUT, PHYEMAC#TXGMIIMIICLKIN Clock Frequencies**

Clock Signal	Direction	Operating Speed		
		1000 Mb/s	100 Mb/s	10 Mb/s
EMAC#PHYTXGMIIMIICLKOUT	Output	125 MHz	25 MHz	2.5 MHz
PHYEMAC#TXGMIIMIICLKIN	Input	125 MHz	25 MHz	2.5 MHz

GMII (Byte PHY) Mode

EMAC#PHYTXGMIIMIICLKOUT is created by the clock generator to provide the frequencies defined in Table B-6. In Byte PHY mode, this should only be used at 1 Gb/s.

The clock signal input into PHYEMAC#TXGMIIMIICLKIN should be derived from EMAC#PHYTXGMIIMIICLKOUT at 1 Gb/s. At 10/100 Mb/s speeds, this clock should originate from the external PHY device but its frequency is divided by two in the FPGA logic (see "GMII Clock Management for Tri-Speed Operation Using Byte PHY"). The resultant clock should be used by the FPGA logic to clock the transmitter client and

physical interface logic. Internally in the Ethernet MAC, this clock is used to derive all physical and client transmitter clocks.

Table B-6: **EMAC#PHYTXGMIIMIICLKOUT, PHYEMAC#TXGMIIMIICLKIN Clock Frequencies**

Clock Signal	Direction	Operating Speed		
		1000 Mb/s	100 Mb/s	10 Mb/s
EMAC#PHYTXGMIIMIICLKOUT	Output	125 MHz	n/a	n/a
PHYEMAC#TXGMIIMIICLKIN	Input	125 MHz	12.5 MHz	1.25 MHz

SGMII and 1000BASE-X PCS/PMA Modes

These clock signals are unused. EMAC#PHYTXGMIIMIICLKOUT should be left unconnected, and PHYEMAC#TXGMIIMIICLKIN should be connected to logic 0.

EMAC#PHYTXCLK

This clock signal is not used in normal operation and should be left unconnected.

EMAC#CLIENTTXCLIENTCLKOUT, CLIENTEMAC#TXCLIENTCLKIN

Clock Enables Not in Use

EMAC#CLIENTTXCLIENTCLKOUT is created by the clock generator to provide the frequencies defined in Table B-7.

The clock signal input into the CLIENTEMAC#TXCLIENTCLKIN port should be used to clock the transmitter client interface logic. This can be derived from EMAC#CLIENTTXCLIENTCLKOUT as illustrated in Figure B-2.

Table B-7: **EMAC#CLIENTTXCLIENTCLKOUT, CLIENTEMAC#TXCLIENTCLKIN Clock Frequencies**

Clock Signal	Direction	Operating Speed		
		1000 Mb/s	100 Mb/s	10 Mb/s
EMAC#CLIENTTXCLIENTCLKOUT	Output	125 MHz	12.5 MHz	1.25 MHz
CLIENTEMAC#TXCLIENTCLKIN	Input	125 MHz	12.5 MHz	1.25 MHz

Clock Enables in Use

CLIENTEMAC#TXCLIENTCLKIN is unused and can be connected to logic 0.

EMAC#CLIENTTXCLIENTCLKOUT is a clock enable rather than a clock. This clock enable is synchronous to the clock input into the PHYEMAC#TXGMIIMIICLKIN port as illustrated in Figure B-3. All transmitter client logic should be clocked from PHYEMAC#TXGMIIMIICLKIN and should use EMAC#CLIENTTXCLIENTCLKOUT as a clock enable as illustrated in Figure B-3.

Table B-8 shows the relationship between the PHYEMAC#TXGMIIMIICLKIN clock and the EMAC#CLIENTTXCLIENTCLKOUT clock enable.

Table B-8: **EMAC#CLIENTTXCLIENTCLKOUT, CLIENTEMAC#TXCLIENTCLKIN Clock Frequencies**

Clock Signal	Direction	Operating Speed		
		1000 Mb/s	100 Mb/s	10 Mb/s
PHYEMAC#TXGMIIMIICLKIN	Input	125 MHz	25 MHz	2.5 MHz
EMAC#CLIENTTXCLIENTCLKOUT	Output	Held at logic 1	Alternates between logic 0 and 1 to produce half clock rate	

EMAC#CLIENTRXCLIENTCLKOUT, CLIENTEMAC#RXCLIENTCLKIN

Clock Enables Not in Use

EMAC#CLIENTRXCLIENTCLKOUT is created by the clock generator to provide the frequencies defined in Table B-9.

The clock signal input into the CLIENTEMAC#RXCLIENTCLKIN port should be used to clock the receiver client interface logic. This can be derived from EMAC#CLIENTRXCLIENTCLKOUT as illustrated in Figure B-2.

Table B-9: **EMAC#CLIENTRXCLIENTCLKOUT, CLIENTEMAC#RXCLIENTCLKIN Clock Frequencies**

Clock Signal	Direction	Operating Speed		
		1000 Mb/s	100 Mb/s	10 Mb/s
EMAC#CLIENTRXCLIENTCLKOUT	Output	125 MHz	12.5 MHz	1.25 MHz
CLIENTEMAC#RXCLIENTCLKIN	Input	125 MHz	12.5 MHz	1.25 MHz

Clock Enables in Use

CLIENTEMAC#RXCLIENTCLKIN is unused and can be connected to logic 0.

EMAC#CLIENTRXCLIENTCLKOUT is a clock enable rather than a clock. This clock enable is synchronous to the clock input into the port PHYEMAC#RXCLK as illustrated in Figure B-3. All receiver client logic should be clocked from the clock input PHYEMAC#RXCLK and should use EMAC#CLIENTRXCLIENTCLKOUT as a clock enable as illustrated in Figure B-3.

Table B-10 shows the relationship between the PHYEMAC#RXCLK clock and the EMAC#CLIENTRXCLIENTCLKOUT clock enable.

Table B-10: **EMAC#CLIENTRXCLIENTCLKOUT, PHYEMAC#MIITXCLK Clock Frequencies**

Clock Signal	Direction	Operating Speed		
		1000 Mb/s	100 Mb/s	10 Mb/s
PHYEMAC#RXCLK	Input	125 MHz	25 MHz	2.5 MHz
EMAC#CLIENTRXCLIENTCLKOUT	Output	Held at logic 1	Alternates between logic 0 and 1 to produce half clock rate	

XILINX®

Appendix C

Virtex-4 to Virtex-5 FPGA Enhancements

The Virtex®-5 Embedded Tri-Mode Ethernet MAC is derived from the Virtex-4 FPGA Embedded Tri-mode Ethernet MAC with some key enhancements and a few minor modifications, as documented in this appendix.

The Virtex-5 FPGA Embedded Tri-Mode Ethernet MAC is a stand-alone block located in the block RAM columns. It is no longer resides inside in the processor block, as is the case for Virtex-4 FPGA Ethernet MACs.

New Features

Unidirectional Enable

Unidirectional enable is a new mode that allows Ethernet MAC to transmit even while there is a loss of synchronization at the receiver (1000BASE-X PCS/PMA or SGMII modes). This mode can be turned on/off via an MDIO write and its value can be read from bit 0.5 in the PCS Configuration Register 0. This bit is also new in the IEEE 802.3 specification and was previously reserved. The default value of this register bit is set by the EMAC#_UNIDIRECTION_ENABLE attribute, as described in "Additional Attributes," page 221.

Programmable Auto-Negotiation Link Timer

A programmable link timer interrupt value for auto-negotiation is available in the Virtex-5 FPGA Ethernet MAC for 1000BASE-X PCS/PMA or SGMII modes. This value is set using the EMAC#_LINKTIMERVAL[8:0] attribute, as described in "Additional Attributes," page 221.

GT Loopback

This mode allows the position of the loopback to be selected between loopback in the RocketIO™ serial transceiver or a loopback internal to the Ethernet MAC. This is applicable for 1000BASE-X PCS/PMA or SGMII modes only. When the loopback is internal to the Ethernet MAC, a constant stream of Idle code groups are transmitted through the RocketIO serial transceiver. The EMAC#_GTLOOPBACK attribute, as described in "Additional Attributes," page 221, can be used to select the default position of the loopback.

DCR Bus Modifications

In Virtex-5 devices, the DCR bus signals can be directly accessed in the FPGA logic; whereas in Virtex-4 devices, these signals are inaccessible because they are directly connected to the PowerPC™ processor.

- The PowerPC 405 processor is the current standard in Virtex-4 FPGAs. Changing processors requires some changes to the DCR acknowledge operation. These changes are backward-compatible. In the XPS_LL_TEMAC soft core that is delivered through the XPS tool, the DCR bus is not exposed. The core contains a bridge to enable the control of the Ethernet MAC via the PLB v4.6 interface. For more information on the PLB v4.6 interface, see DS531, *Processor Local Bus (PLB) v4.6 (v1.00a)*.
- The DCR register definitions have been modified. In Virtex-4 FPGA designs, the DCR registers holds information about EMAC0 and EMAC1 in the same register. In Virtex-5 FPGA designs, information on each EMAC is be accessed specifically through the EMAC0_DCRBASEADDR and EMAC1_DCRBASEADDR registers.
- There have also been changes to the DCR register addressing. This change requires Virtex-4 FPGA designs to update the address schemes. In Virtex-4 FPGA designs, the DCR bus is responsible for decoding the user-defined DCR_BASEADDR to enable the EMAC host interface access. In Virtex-5 FPGA designs, the interface only responds to accesses made to addresses matching the EMAC#_DCRBASEADDR attributes. Table C-1 shows the modifications to the DCR register addressing.

Table C-1: **DCR Register Address Modifications**

DCR Register	Virtex-4 FPGA Address	Virtex-5 FPGA Address
dataRegMSW	DCR_BASEADDR_00	EMAC#_DCRBASEADDR_00
dataRegLSW	DCR_BASEADDR_01	EMAC#_DCRBASEADDR_01
cntlReg	DCR_BASEADDR_10	EMAC#_DCRBASEADDR_10
RdyStatus	DCR_BASEADDR_11	EMAC#_DCRBASEADDR_11

Clocking Scheme Enhancements

Host Clock

The Virtex-4 HOSTCLK input clock has to be supplied at all times even if the host interface is not used. In Virtex-5 designs, when neither the generic host bus nor the DCR bus are enabled, this clock input can be connected to logic 0.

Advanced Clocking Schemes

In the Virtex-5 FPGA Embedded Tri-Mode Ethernet MAC, two advanced clocking schemes are available, named “Clock Enables” and “Byte PHY.” In Virtex-4 FPGA designs, these clocking schemes can be implemented in the FPGA logic but for a smaller subset of physical interface configurations. “Advanced Clocking Schemes,” page 208 provides further details on theses clocking schemes.

Clock Enables

Virtex-4 FPGA clock enable signals are created in the FPGA logic, resulting in a clocking scheme that halves the global clock usage when using the MII physical interface for 10 Mb/s or 100 Mb/s.

The Virtex-5 FPGA clock enables are provided by the Ethernet MAC to the FPGA logic, and their use has extended to cover MII, GMII, and RGMII interfaces at all three Ethernet speeds.

Byte PHY

For Virtex-4 FPGA designs, a solution is possible in the FPGA logic that halves the global clock usage when using the GMII/MII physical interface at all three speeds. But, it only supports full-duplex mode.

In Virtex-5 FPGA designs, the Byte PHY functionality is extended to cover the GMII/MII, again at all three speeds but supports both full- and half-duplex modes.

Modifications Related to the Physical Interface

Collision Handling

In Virtex-4 FPGA designs, the `GMII_COL/MII_COL` signal (half-duplex mode collision indicator from GMII/MII interface) had to be lengthened in the FPGA logic. In Virtex-5 FPGA designs, the signal can be input directly to the `PHYEMAC#COL` port of the Ethernet MAC.

RGMII Version 2.0 Clock Management

In Virtex-5 FPGA designs, an ODELAY design element can be used to simplify the generation of 2 ns skew between the transmit clock and data at the FPGA device pads. This is part of the IOB and clock logic that can be used with the Ethernet MAC to meet the RGMII v2.0 physical interface specification.

Port Map Changes

The following name changes occurred to clarify functionality:

- `EMAC#CLIENTTXGMIIMIICLKOUT` changed to `EMAC#PHYTXGMIIMIICLKOUT`
- `CLIENTEMAC#TXGMIIMIICLKIN` changed to `PHYEMAC#TXGMIIMIICLKIN`

The following ports were removed:

- `EMAC#CLIENTRXDVREG6`: In Virtex-4 devices, this signal is reserved and not used.
- `TIEEMAC#CONFIGVEC[79:0]`: This port has been replaced by attributes
- `TIEEMAC#UNICASTADDR[47:0]`: This port has been replaced by an attribute.

The following port was added to enable advanced clocking schemes to be used:

- `EMAC#SPEEDIS10100`

Tie-Off Pins Changed to Attributes

The Virtex-4 FPGA Embedded Tri-Mode Ethernet MAC has 80 tie-off pins (`TIEEMAC#CONFIGVEC[79:0]`) that can be used to configure the Ethernet MAC at power-up or when the Ethernet MAC is reset.

In the Virtex-5 FPGA Embedded Tri-Mode Ethernet MAC, these 80 tie-off pins have been replaced with attributes. Table C-2 shows how the Virtex-5 FPGA attributes relate to the Virtex-4 FPGA tie-off pins.

Table C-2: **Mapping Tie-Off Pin Names to Attribute Names**

Virtex-4 FPGA Tie-off Pin Name	Virtex-5 FPGA Attribute Name
TIEEMAC#CONFIGVEC[79]	EMAC#_CONFIGVEC_79
TIEEMAC#CONFIGVEC[78]	EMAC#_PHYRESET
TIEEMAC#CONFIGVEC[77]	EMAC#_PHYINITAUTONEG_ENABLE
TIEEMAC#CONFIGVEC[76]	EMAC#_PHYISOLATE
TIEEMAC#CONFIGVEC[75]	EMAC#_PHYPOWERDOWN
TIEEMAC#CONFIGVEC[74]	EMAC#_PHYLOOPBACKMSB
TIEEMAC#CONFIGVEC[73]	EMAC#_MDIO_ENABLE
TIEEMAC#CONFIGVEC[72]	EMAC#_SPEED_MSB
TIEEMAC#CONFIGVEC[71]	EMAC#_SPEED_LSB
TIEEMAC#CONFIGVEC[70]	EMAC#_RGMII_ENABLE
TIEEMAC#CONFIGVEC[69]	EMAC#_SGMII_ENABLE
TIEEMAC#CONFIGVEC[68]	EMAC#_1000BASEX_ENABLE
TIEEMAC#CONFIGVEC[67]	EMAC#_HOST_ENABLE
TIEEMAC#CONFIGVEC[66]	EMAC#_TX16BITCLIENT_ENABLE
TIEEMAC#CONFIGVEC[65]	EMAC#_RX16BITCLIENT_ENABLE
TIEEMAC#CONFIGVEC[64]	EMAC#_ADDRFILTER_ENABLE
TIEEMAC#CONFIGVEC[63]	EMAC#_LTCHECK_DISABLE
TIEEMAC#CONFIGVEC[62]	EMAC#_RXFLOWCTRL_ENABLE
TIEEMAC#CONFIGVEC[61]	EMAC#_TXFLOWCTRL_ENABLE
TIEEMAC#CONFIGVEC[60]	EMAC#_TXRESET
TIEEMAC#CONFIGVEC[59]	EMAC#_TXJUMBOFRAME_ENABLE
TIEEMAC#CONFIGVEC[58]	EMAC#_TXINBANDFCS_ENABLE
TIEEMAC#CONFIGVEC[57]	EMAC#_TX_ENABLE
TIEEMAC#CONFIGVEC[56]	EMAC#_TXVLAN_ENABLE
TIEEMAC#CONFIGVEC[55]	EMAC#_TXHALFDUPLEX
TIEEMAC#CONFIGVEC[54]	EMAC#_TXIFGADJUST_ENABLE
TIEEMAC#CONFIGVEC[53]	EMAC#_RXRESET

Table C-2: **Mapping Tie-Off Pin Names to Attribute Names *(Cont'd)***

Virtex-4 FPGA Tie-off Pin Name	Virtex-5 FPGA Attribute Name
TIEEMAC#CONFIGVEC[52]	EMAC#_RXJUMBOFRAME_ENABLE
TIEEMAC#CONFIGVEC[51]	EMAC#_RXINBANDFCS_ENABLE
TIEEMAC#CONFIGVEC[50]	EMAC#_RX_ENABLE
TIEEMAC#CONFIGVEC[49]	EMAC#_RXVLAN_ENABLE
TIEEMAC#CONFIGVEC[48]	EMAC#_RXHALFDUPLEX
TIEEMAC#CONFIGVEC[47:0]	EMAC#_PAUSEADDR[47:0]

Additional Attributes

In addition to the tie-off pins being converted to attributes, six new parameters have been added to the Virtex-5 FPGA Embedded Tri-Mode Ethernet MAC that are also controlled by attributes. The following list provides a brief summary:

- EMAC#_DCRBASEADDR[0:7]: Separate DCR base addresses are specified for each Ethernet MAC.
- EMAC#_UNIDIRECTION_ENABLE: Allows the Ethernet MAC to transmit even while loss of synchronization at the receiver (1000BASE-X PCS/PMA or SGMII mode).
- EMAC#_LINKTIMERVAL[8:0]: Programmable link timer interrupt value for auto-negotiation (1000BASE-X PCS/PMA or SGMII mode).
- EMAC#_BYTEPHY: Optional advanced clocking scheme to reduce use of global clock resources.
- EMAC#_USECLKEN: Optional advanced clocking scheme to reduce use of global clock resources. This mode switches the client interface clock output to become a clock enable.
- EMAC#_GTLOOPBACK: Enables loopback in the RocketIO serial transceiver, otherwise an internal loopback is used (1000BASE-X PCS/PMA or SGMII mode).

More detailed information can be found in Chapter 2.

XILINX®

Appendix D

Differences between Soft IP Cores and the Tri-Mode Ethernet MAC

This appendix describes the differences between the Embedded Tri-Mode Ethernet MAC block and the soft IP core solutions provided by the CORE Generator™ software. The functionality provided by the Embedded Tri-Mode Ethernet MAC can also be provided by linking together the Tri-Mode Ethernet MAC soft IP core and the Ethernet 1000BASE-X PCS/PMA or SGMII core. More details are available at:

http://www.xilinx.com/products/design_resources/conn_central/protocols/gigabit_ethernet.htm

There are, however, some differences in the operation of the soft IP cores and the Tri-Mode Ethernet MAC. These differences are detailed below.

Features Exclusive to the Embedded Tri-Mode Ethernet MAC

The Tri-Mode Ethernet MAC tile features two Tri-Mode Ethernet MACs, each with optional PCS/PMA functionality. In addition, each Embedded Tri-Mode Ethernet MAC:

- Supports 2 Gb/s operation with a 16-bit client interface.
- Includes an RGMII/SGMII status register. See "RGMII/SGMII Configuration Register," page 93.
- Can be configured using the DCR bus.

Features Exclusive to Soft IP Cores

These features are exclusive to soft IP cores:

- The soft Tri-Mode Ethernet MAC core:
  - Supports 1 GB half duplex mode for parallel physical interfaces.
  - Supports control frames larger than 64 bytes.
  - Includes a statistics bit to indicate an address match.
  - Has parallel statistics outputs.

- The soft PCS/PMA core:
  - Supports the ten bit interface (TBI).
  - Supports dynamic switching between 1000BASE-X PCS/PMA and SGMII.
  - Outputs a status vector with bits that indicate:
    - The status of the link.
    - The status of the link synchronization state machine.
    - When the core is receiving /C/ ordered sets.
    - When the core is receiving /I/ ordered sets.
    - When the core is receiving invalid data.
- Soft PCS/PMA and Tri-Mode Ethernet MAC cores:
  - Can be connected together to provide a single Ethernet interface.
  - Can be configured by a vector when the management interface is not required. The vector signals equate to attributes in the hard Tri-Mode Ethernet MAC.
  - Can be targeted to a wide variety of Xilinx® devices.