GMII to RGMII v4.0

LogiCORE IP Product Guide

Vivado Design Suite

PG160 April 6, 2016

XILINX
ALL PROGRAMMABLE™

Table of Contents

Chapter 6: Test Bench

Appendix A: Migrating and Upgrading

Appendix B: Debugging

Appendix C: Additional Resources and Legal Notices

Introduction

The Xilinx® LogiCORE™ IP Gigabit Media Independent Interface (GMII) to Reduced Gigabit Media Independent Interface (RGMII) core provides the RGMII between RGMII-compliant Ethernet physical media devices (PHY) and the Gigabit Ethernet controller (GEM) in the Zynq®-7000 All Programmable (AP) SoCs and Zynq® UltraScale+™ MP SoCs. This core can be used in all three modes of operation (10/100/1000 Mb/s). The Management Data Input/Output (MDIO) interface is used to determine the speed of operation. This core can switch dynamically between the three different speed modes.

Features

- Tri-speed (10/100/1000 Mb/s) operation
- Full-duplex operation
- MDIO interface to set operating speed and duplex mode by MAC

LogiCORE IP Facts Table	
Core Specifics	
Supported Device Family(1)	Zynq® UltraScale+™ MP SoC, Zynq®-7000 All Programmable SoC
Supported User Interfaces	GMII
Resources	Performance and Resource Utilization web page
Provided with Core	
Design Files	Encrypted RTL
Example Design	GMII to RGMII with internally generated GMII clock GMII to RGMII with externally generated GMII clock
Test Bench	Demonstration Test Bench
Constraints File	XDC
Simulation Model	Not Provided
Supported S/W Driver	N/A
Tested Design Flows(2)	
Design Entry	Vivado® Design Suite Vivado
Simulation	For supported simulators, see the Xilinx Design Tools: Release Notes Guide.
Synthesis	Vivado Synthesis
Support	
Provided by Xilinx at the Xilinx Support web page	

Notes:

1. For a complete list of supported devices, see the Vivado IP catalog.
2. For the supported versions of the tools, see the Xilinx Design Tools: Release Notes Guide.

Overview

The GMII to RGMII IP core provides the Reduced Gigabit Media Independent Interface (RGMII) between Ethernet physical media devices and the Gigabit Ethernet controller in Zynq®-7000 All Programmable SoCs and Zynq® UltraScale+™ MP SoCs. This core can switch dynamically between the three different speed modes of operation (10/100/1000 Mb/s).

Feature Summary

- Tri-speed operation (10/100/1000 Mb/s)

  The line speed can be changed dynamically (for example, during run time) by programming the speed bits in the control register.

- Full-duplex operation
- MDIO interface to set operating speed by Ethernet MAC

  Speed settings are contained in the control register implemented within the core. MDIO transactions are used to program this control register.

Applications

The GMII to RGMII IP core is designed for use with the Gigabit Ethernet embedded blocks in the Zynq-7000 AP SoC and Zynq® UltraScale+™ MP SoC devices. The Gigabit Ethernet MAC embedded blocks present in the Zynq-7000 AP SoC or Zynq® UltraScale+™ MP SoC device would provide an RGMII interface through the Multiplexed I/O pins (MIO) and a GMII interface through the EMIO interface to route through the Programmable Logic (PL). The GMII to RGMII IP can be used to provide an RGMII interface using the PL. For more information on the device specific Gigabit Ethernet Controller, see the *Xilinx Zynq-7000 All Programmable SoC Technical Reference Manual* [Ref 1] and *Zynq UltraScale+ MPSoC Technical Reference Manual [Ref 11]*.

Unsupported Features

There are no unsupported features for this core.

Licensing and Ordering Information

This Xilinx® LogiCORE™ IP module is provided at no additional cost with the Xilinx Vivado® Design Suite under the terms of the Xilinx End User License.

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

Product Specification

Figure 2-1 illustrates the connection of the Gigabit Ethernet Controller in the Zynq®-7000 All Programmable SoC to the GMII to RGMII core. The same connection is applicable for Zynq® UltraScale+™ MP SoC too.

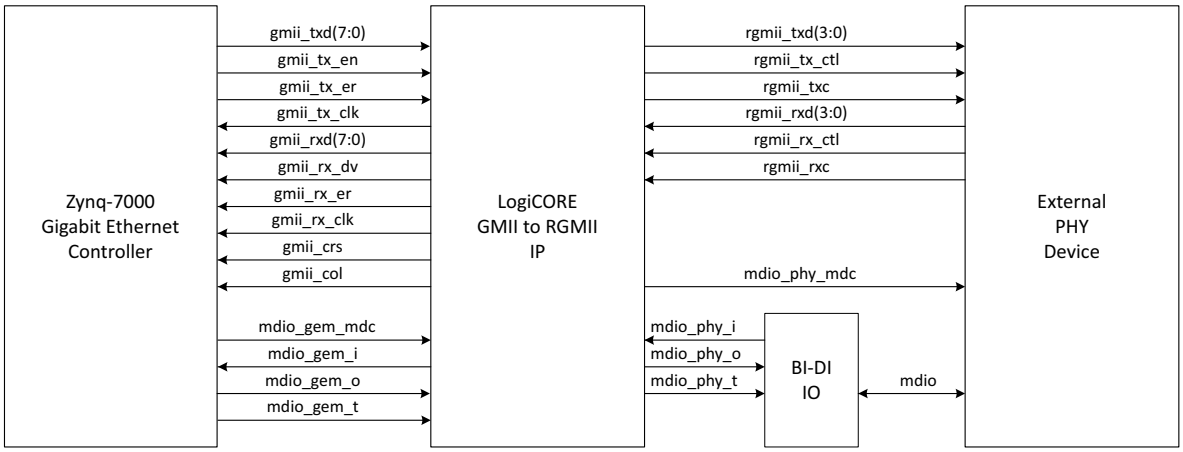

Figure 2-1: **GMII to RGMII Core Ports and Interfaces**

IMPORTANT: *The MDIO interface is necessary for the operation of the core because the auto-negotiated speed of operation from the PHY is communicated to the Ethernet MAC through MDIO.*

The 200 MHz clock input is used as a reference clock for the IDELAYCTRL elements and input for the management modules.

If the GMII clock is sourced internally (C_EXTERNAL_CLOCK = 0), this 200 MHz clock is the input clock to the MMCM from which the TX clocks for all line rates (125/12.5/2.5 MHz for 1000/100/10 Mb/s, respectively) are generated.

Standards

- Ethernet standard IEEE 802.3-2012 Clauses 22 and 35 [Ref 2]
- Reduced Gigabit Media Independent Interface (RGMII) V2.0 [Ref 4]

Performance

This section describes the performance of the GMII to RGMII core.

Maximum Frequencies

The GMII to RGMII core operates at 125 MHz. The Management module operates at 200 MHz.

Latency

The following measurements are for the core only and do not include any IOB registers.

Transmit Path

As measured from a data octet input into `gmii_txd[7:0]` of the transmitter side of GMII interface until that data appears on the `rgmii_txd[3:0]` on the RGMII interface, the latency through the core through the transmit direction is one clock period of `gmii_tx_clk_int`.

Receive Path

Measured from a data octet input into `rgmii_rxd[3:0]` of the receiver side of the RGMII interface until that data appears on the `gmii_rxd[7:0]` on the GMII interface, the latency through the core through the receive direction is one clock period of `rgmii_rx_clk`, plus the additional delay equal to the fixed delay specified on the IDELAY component.

Throughput

The GMII to RGMII core operates at full line rates of 10/100/1000 Mb/s.

Power

No information is currently provided for this core.

Resource Utilization

For details about Resource Utilization, visit Performance and Resource Utilization.

Port Descriptions

This section describes the ports for the GMII to RGMII core.

Internal Encrypted Hierarchy of the Core Level Ports

All ports in the encrypted hierarchy of the core are internal connections in FPGA logic.

Unencrypted HDL in the core and example design (delivered with the core) connects the GMII to RGMII core and adds IBUFs, OBUFs, and IOB flip-flops to the external signals of the core. IOBs are added to the remaining unconnected ports to run the example design using Xilinx implementation software.

All the ports described here indicate the pins in the encrypted hierarchy at the core level. The block level design instantiates the core, clock selection logic, and shared logic if **Include Shared Logic in Core** is selected.

Input/Output Signals

The I/O signals for the GMII to RGMII core are listed in Table 2-1. The interfaces referenced in this table are shown in Figure 2-1.

Table 2-1: **I/O Signals**

Signal Name	Direction	Description
tx_reset	Input	Active-High reset for TX datapath
rx_reset	Input	Active-High reset for RX datapath
ref_clk	Input	This 200 MHz clock is used to clock the Management modules.
speed_mode[1:0]	Output	Indicates line-rate. When the GMII clock is generated internally this signal is used by the clock MUXes. 00: 10 Mb/s 01: 100 Mb/s 10: 1 Gb/s 11: Reserved
gmii_tx_clk	Input	Transmit clock from Zynq-7000 AP SoC device GEM. This clock is 125/25/2.5 MHz for 1000/100/10 Mb/s line rates.
gmii_tx_clk_90	Input	gmii_tx_clk with 90° phase shift

Table 2-1: **I/O Signals *(Cont'd)***

Signal Name	Direction	Description
gmii_tx_en	Input	Data Enable control signal from the Zynq-7000 AP SoC device GEM
gmii_txd[7:0]	Input	Transmit data from the Zynq-7000 AP SoC device GEM
gmii_tx_er	Input	Error control signal from the Zynq-7000 AP SoC device GEM
gmii_crs	Output	Carrier sense signal to the Zynq-7000 AP SoC device GEM (GEM does not use the Carrier Sense signal in Full Duplex mode)
gmii_col	Output	Collision signal to the Zynq-7000 AP SoC device GEM (GEM does not use the Collision signal in Full Duplex mode)
gmii_rx_clk	Output	Receive clock output from the GMII to RGMII IP core to the Zynq-7000 AP SoC device GEM
gmii_rx_dv	Output	Data Valid control signal to the Zynq-7000 AP SoC device GEM
gmii_rxd[7:0]	Output	Received data to the Zynq-7000 AP SoC device GEM
gmii_rx_er	Output	Error control signal to the Zynq-7000 AP SoC device GEM
mdio_gem_mdc	Input	MDIO clock input to the GMII to RGMII core
mdio_gem_i	Output	The mdio_i line driven by Zynq-7000 AP SoC device GEM. It is used by the core during Register write operation.
mdio_gem_o	Input	The mdio_o line to the Zynq-7000 AP SoC device GEM.
mdio_gem_t	Input	The mdio_t line driven by the Zynq-7000 AP SoC device GEM.
link_status	Output	Link status decoded from RGMII in-band signaling 0 = Link Down 1 = Link Up
clock_speed[1:0]	Output	Link Speed decoded from RGMII in-band signaling 00: 10 Mb/s 01: 100 Mb/s 10: 1 Gb/s 11: Reserved
duplex_status	Output	Duplex status decoded from RGMII in-band signaling. 0 = Half Duplex 1 = Full Duplex
rgmii_txc	Output	Transmit Clock to the external PHY device
rgmii_tx_ctl	Output	Transmit Control to the external PHY device
rgmii_txd[3:0]	Output	Transmit Data to the external PHY device
rgmii_rxc	Input	Receive Clock from the external PHY device
rgmii_rx_ctl	Input	Receive Control from the external PHY device
rgmii_rxd[3:0]	Input	Receive Data from the external PHY device
mdio_phy_mdc	Output	MDIO clock to the external PHY device
mdio_phy_i	Input	The mdio_i line from the bidirectional buffer connected to the External PHY device MDIO line.

Table 2-1: **I/O Signals *(Cont'd)***

Signal Name	Direction	Description
mdio_phy_o	Output	The mdio_o line to the bidirectional buffer connected to the external PHY device MDIO line.
mdio_phy_t	Output	The mdio_t line to the bidirectional buffer connected to the External PHY device MDIO line.

Interfaces

Figure 2-2 shows the ports and interfaces for the GMII to RGMII core.

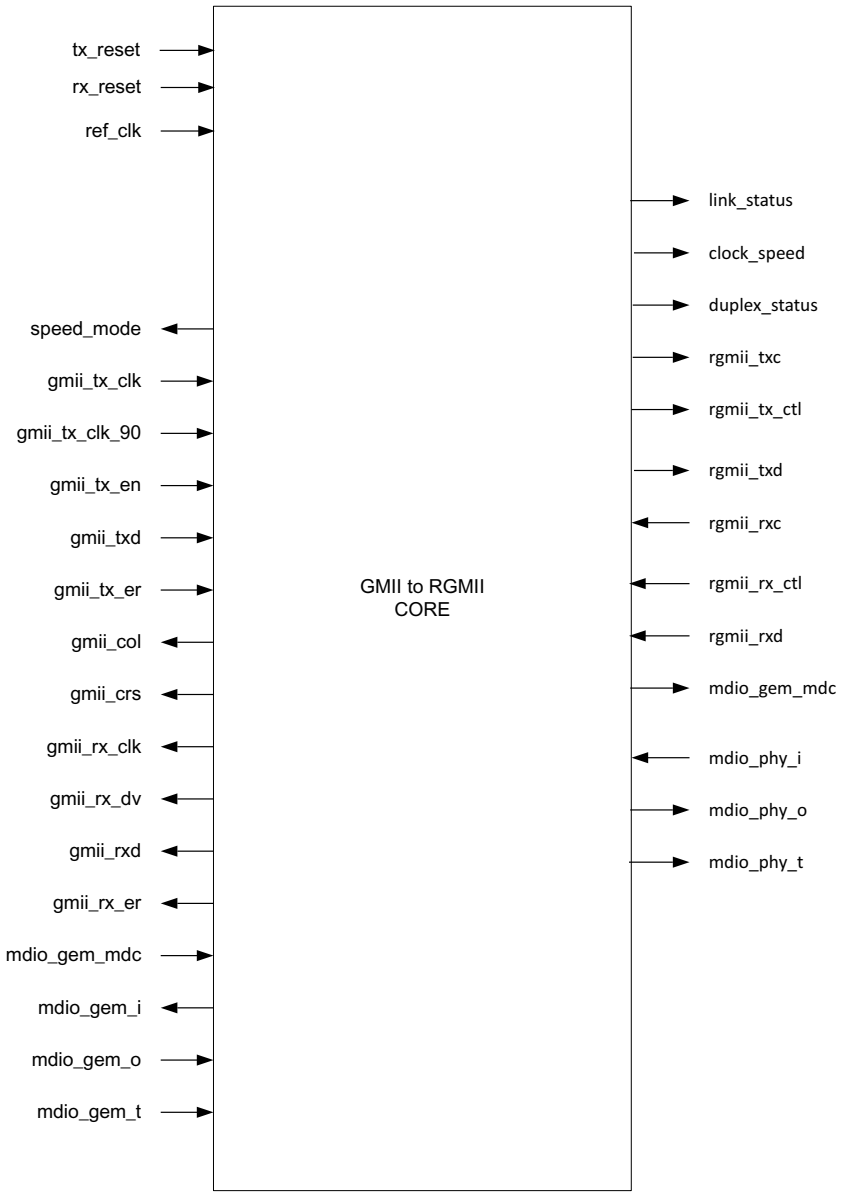

Figure 2-2: **Interface of GMII to RGMII Core**

Block Hierarchy Level Ports

The ports described here indicate the pins at the block level. The block level design instantiates the core, clock selection logic, and shared logic if the **Include Shared Logic in Core** option is selected. In most cases, the block level design is located in the IP catalog and placed in the Vivado IP integrator.

Input/Output Signals

Table 2-2: **Block Level I/O Signals**

Signal Name	Direction	Description
tx_reset	Input	Active-High reset for TX Datapath.
rx_reset	Input	Active-High reset for RX Datapath.
clkin	Input	Valid only for `Shared Logic in the Core` configuration. This 200 MHz clock is used as the reference clock for the IDELAYCTRL elements and to clock the Management modules. This clock is input to the MMCM when external clock mode is not enabled for generating 125/25/2.5 MHz clocks.
ref_clk_out	Output	Valid only for `Shared Logic in the Core` configuration. This is a clkin signal passed through a BUFG and used to drive ref_clk_in for multiple instances of the core.
gmii_clk	Input	Valid only for `Shared Logic in the Core` configuration and if the `external clock` option is selected. The clock has a frequency of 125/25/2.5 MHz, depending on the speed selected.
gmii_clk_out	Output	Valid only for `Shared Logic in the Core` configuration and if the `external clock` option is selected. The clock is gmii_clk passed through a BUFG and is used to drive gmii_clk for multiple instances of the core.
gmii_clk_90	Input	gmii_clk phase shifted by 90°. This signal is valid only when 2 ns skew is added through MMCM.
gmii_clk_90_out	Output	gmii_clk_out phase shifted by 90°. This signal is valid only when 2 ns skew is added through MMCM.
gmii_clk_125m_out	Output	Valid only for `Shared Logic in the Core` configuration and if the `external clock` option is not selected. This 125 MHz clock is generated by MMCM from clkin.
gmii_clk_25m_out	Output	Valid only for `Shared Logic in the Core` configuration and if the `external clock` option is not selected. This 25 MHz clock is generated by MMCM from clkin.
gmii_clk_2_5m_out	Output	Valid only for `Shared Logic in the Core` configuration and if the `external clock` option is not selected. This 2.5 MHz clock is generated through clock division.
mmcm_locked_out	Output	Valid only for `Shared Logic in the Core` configuration and if the external clock option is not selected. This indicates that the MMCM has locked.

Table 2-2: **Block Level I/O Signals *(Cont'd)***

Signal Name	Direction	Description
ref_clk_in	Input	Valid only for `Shared Logic in the Example Design` configuration. This signal is connected to the ref_clk_out signal of the core instance generated in the `Shared Logic in the Core` configuration.
gmii_clk_125m_90_out	Output	gmii_clk_125m_out phase shifted by 90°. This signal is valid only when 2 ns skew is added through MMCM.
gmii_clk_25m_90_out	Output	gmii_clk_25m_out phase shifted by 90°. This signal is valid only when 2 ns skew is added through MMCM.
gmii_clk_2_5m_90_out	Output	gmii_clk_2_5m_out phase shifted by 90°. This signal is valid only when 2 ns skew is added through MMCM.
gmii_clk	Input	Valid only for `Shared Logic in the Example Design` configuration and if `external clock` option is selected. This signal is connected to the gmii_clk_out signal of the core instance generated in the `Shared Logic in the Core` configuration.
gmii_clk_125m_in	Input	Valid only for `Shared Logic in the Example Design` configuration and if `external clock` option is not selected. This signal is connected to the gmii_clk_125m_out signal of the core instance generated in the `Shared Logic in the Core` configuration.
gmii_clk_25m_in	Input	Valid only for `Shared Logic in the Example Design` configuration and if `external clock` option is not selected. This signal is connected to the gmii_clk_25m_out signal of the core instance generated in the `Shared Logic in the Core` configuration.
gmii_clk_2_5m_in	Input	Valid only for `Shared Logic in the Example Design` configuration and if `external clock` option is not selected. This signal is connected to the gmii_clk_2_5m_out signal of the core instance generated in the `Shared Logic in the Core` configuration.
gmii_clk_125m_90_in	Input	gmii_clk_125m_in phase shifted by 90°. This signal is valid only when 2 ns skew is added through MMCM.
gmii_clk_25m_90_in	Input	gmii_clk_25m_in phase shifted by 90°. This signal is valid only when 2 ns skew is added through MMCM.
gmii_clk_2_5m_90_in	Input	gmii_clk_2_5m_in phase shifted by 90°. This signal is valid only when 2 ns skew is added through MMCM.
mmcm_locked_in	Input	This signal is valid only when the external clock option is not selected for `Shared Logic in the Example Design` configuration. This is used in tx reset control.

Table 2-2: **Block Level I/O Signals *(Cont'd)***

Signal Name	Direction	Description
speed_mode[1:0]	Output	See Table 2-1 for details about these signals.
gmii_tx_clk	Output	
gmii_tx_en	Input	
gmii_txd]7:0]	Input	
gmii_tx_er	Input	
gmii_crs	Output	
gmii_col	Output	
gmii_rx_clk	Output	
gmii_rx_dv	Output	
gmii_rxd[7:0]	Output	
gmii_rx_er	Output	
mdio_gem_mdc	In	
mdio_gem_i	Output	
mdio_gem_o	Input	
mdio_gem_t	Input	
link_status	Output	
clock_speed[1:0]	Output	
duplex_status	Output	
rgmii_txd[3:0]	Output	
rgmii_tx_ctl	Output	
rgmii_txc	Output	
rgmii_rxd[3:0]	Input	
rgmii_rx_ctl	Input	
rgmii_rxc	Input	
mdio_phy_mdc	Output	
mdio_phy_i	Input	
mdio_phy_o	Input	
mdio_phy_t	Input	

Figure 2-3 shows the ports and interfaces for the GMII to RGMII block.

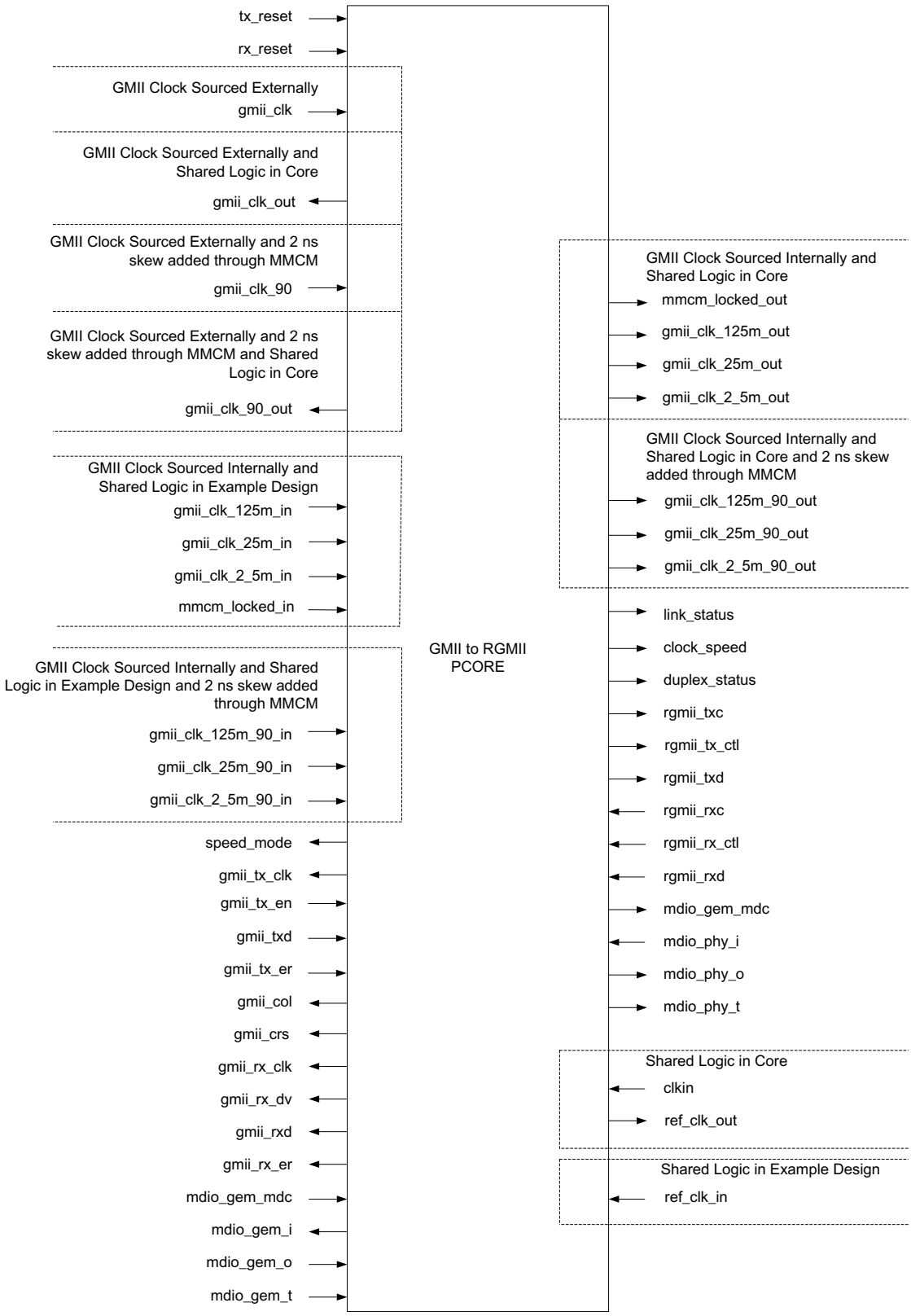

Figure 2-3: **GMII to RGMII Block Level**

MDIO Management System

This section describes the Management Data Input/Output Interface (MDIO) management system. The configuration and status of the GMII to RGMII module instance is achieved by the core registers accessed through the serial MDIO.

MDIO Bus System

The MDIO interface for 1 Gb/s operation (and slower speeds) is defined in IEEE 802.3-2012, clause 22. Figure 2-4 illustrates an example MDIO bus system. 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. 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 of which are MDIO slaves (MDIO Managed Device (MMD) entities).

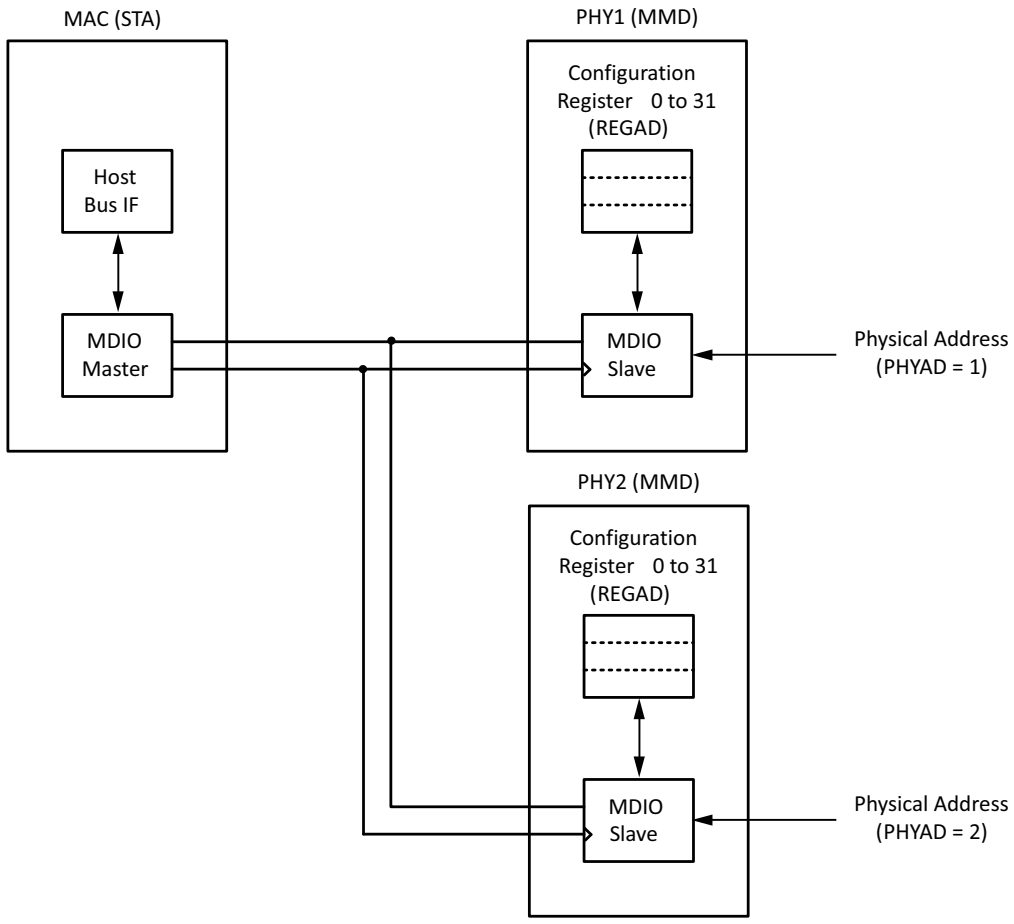

Figure 2-4: **Typical MDIO Managed System**

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 I/F of the Ethernet MAC is able to access the configuration and status registers of two PHY devices through 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 abbreviations that are used in this section are explained in Table 2-3.

Table 2-3: **Abbreviations and Terms**

Abbreviation	Term
PRE	Preamble
ST	Start of Frame
OP	Operation Code
PHYAD	Physical Address
REGAD	Register Address
TA	Turnaround

Write Transaction

Figure 2-5 shows a write transaction across the MDIO, defined as OP = "01." 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.

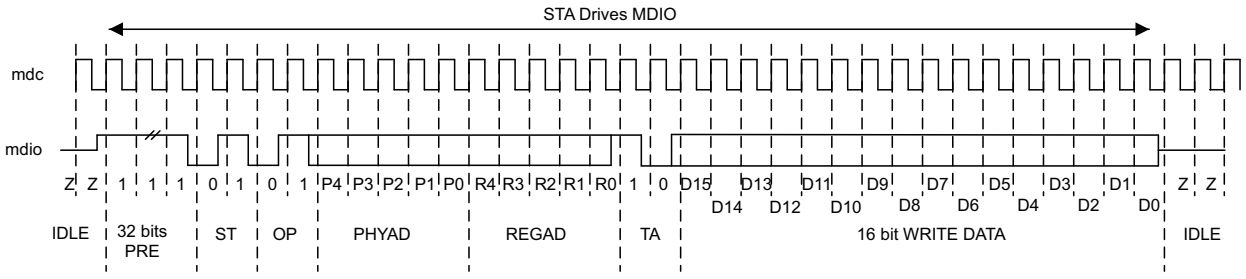

Figure 2-5: **MDIO Write Transaction**

Read Transaction

Figure 2-6 shows a read transaction, defined as OP = "10." The addressed PHY device (with physical address PHYAD) takes control of the MDIO wire during the turn-around cycle and then returns the 16-bit word from the register at REGAD.

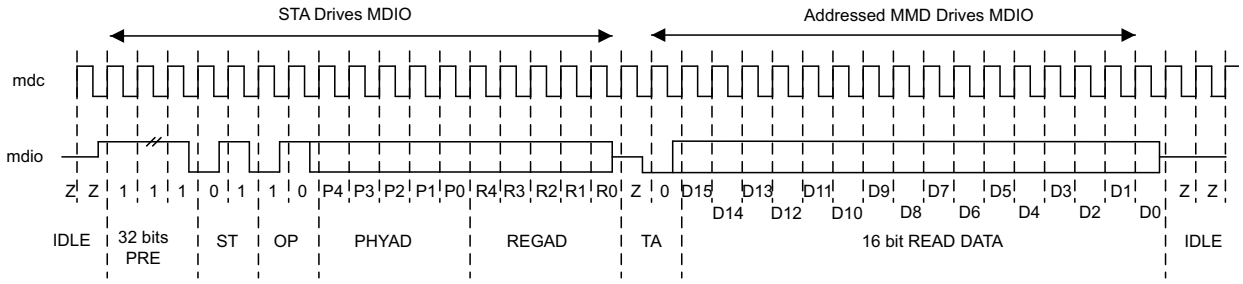

Figure 2-6: **MDIO Read Transaction**

MDIO Addressing

MDIO Addresses consists of two stages: Physical Address (PHYAD) and Register Address (REGAD).

Physical Address

As shown in Figure 2-4, 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 physical address (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. Every MDIO slave must respond to physical address 0. This is a protocol bridging core. In many applications, it is expected that the external slave connected to the RGMII interface would respond to the physical address 0. This core should not respond to block that transaction. This core should respond only to the physical address assigned to it. This requirement dictates that the physical address for any particular physical address (PHY) must not be set to 0 to avoid MDIO contention. Physical Addresses 1 through to 31 can be used to connect up to 31 PHY devices onto a single MDIO bus.

Register Address

Having targeted a particular PHY using PHYAD, the individual configuration or status register within that particular PHY must now be addressed. This is achieved by placing the individual register address into the register address (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 802.3-2012. The

remaining 16 (registers 16 to 31) are reserved for PHY vendors own register definitions. For details of the register map of PHY layer devices and a more extensive description of the operation of the MDIO Interface, see IEEE 802.3-2012.

Connecting the MDIO to an Internally Integrated STA

The MDIO ports of the GMII to RGMII core is designed to be connected to the MDIO ports of an internally integrated Station Management (STA) entity, such as the MDIO port of the Zynq-7000 AP SoC device GEM.

Connecting the MDIO to an External PHY Device

Figure 2-7 shows the MDIO ports of the GMII to RGMII core connected to the MDIO of an external PHY device. Here, `mdio_phy_i`, `mdio_phy_o`, and `mdio_phy_t` must be connected to a 3-state buffer to create a bidirectional wire, MDIO.

This 3-state buffer can either be external to the FPGA or internally integrated by using an IOB IOBUF component with an appropriate SelectIO™ interface standard suitable for the external PHY.

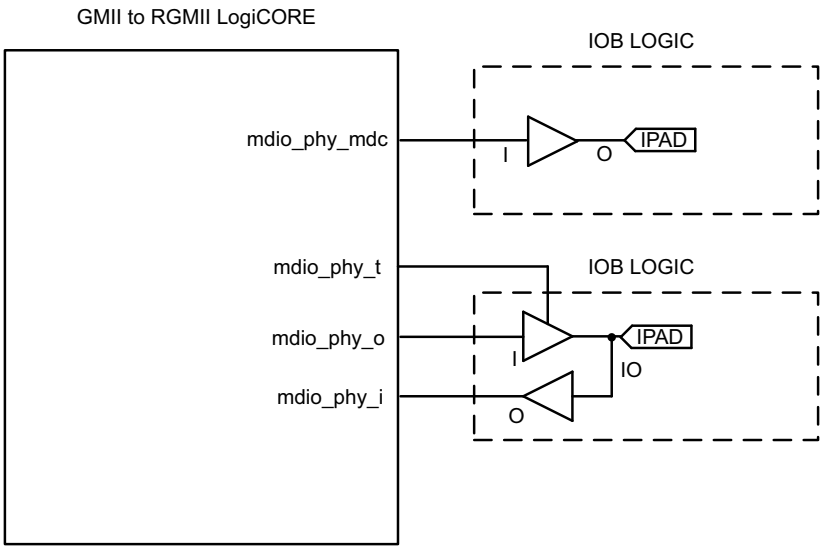

Figure 2-7: **Creating an External MDIO Interface**

Register Space

A control register is implemented in the core that allows the driver software to communicate the line-rate information to the core. This allows the core to dynamically adapt to the line-rate changes. Access to this register is through the MDIO interface. The driver software has to initiate a MDIO read/write cycle to read from and write to this register.

IMPORTANT: *The driver software must use a PHY address which is different from the PHY address used to address the onboard PHY. The PHY address for the core can be set though the VHDL generic C_PHYADDR.*

Table 2-8 illustrates the implementation of the register space in the core. This register is within the management module. The management module monitors the MDIO_GEM_O line for a new MDIO cycle. When a new cycle is initiated and the PHY address matches the PHY address assigned to this core, the module latches the data field of the MDIO frame to the control register for a write cycle or MUXes out the content of the control register to the MDIO_GEM_I line for a read cycle.

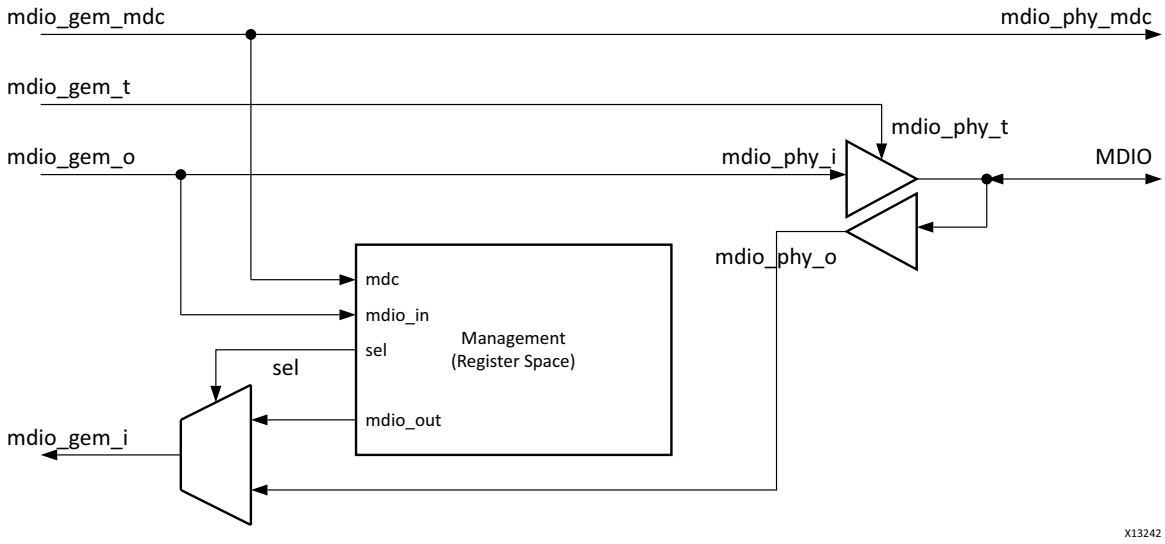

Figure 2-8: **Register Space Implementation**

Control Register

This register is 16-bits wide. Its address is 0x10. The composition of this register is similar to the IEEE standard 802.3 MDIO control register 0x0, which is shown in Table 2-4.

Table 2-4: **Core-Specific Control Register (Address 0x10)**

Bit	Name	Description	R/W
15	Reset	1 = Resets the core and this register 0 = Normal operation	R/W Self-clearing
14	Reserved	Write as 0, ignore on read	R/W
13	Speed Selection (LSB)	6 13 1 1 = Reserved 1 0 = 1,000 Mb/s 0 1 = 100 Mb/s 0 0 = 10 Mb/s	R/W
12:9	Reserved	Write as 0, ignore on read	R/W
8:7	Reserved	Write as 0, ignore on read	R/W
6	Speed Selection (MSB)	6 13 1 1 = Reserved 1 0 = 1,000 Mb/s 0 1 = 100 Mb/s 0 0 = 10 Mb/s	R/W
5:0	Reserved	Write as 0, ignore on read	R/W

Designing with the Core

This chapter includes guidelines and additional information to facilitate designing with the core.

General Design Guidelines

The Vivado® Design Suite is used to synthesize the GMII to RGMII core.

Shared Logic

In version 2.0 of the core and earlier, the RTL hierarchy was fixed. This required shareable clocking and reset logic to be extracted from the core example design for use with a single instance or multiple instances of the core.

The core now includes shared logic, which provides a more flexible architecture that works both as a standalone core and as a part of a larger design with one or more core instances. This minimizes the HDL modifications required but retains flexibility for various uses of the core.

The new hierarchy level is called `<component_name>_support`. In Figure 3-1, the shared logic block is contained in the core. In Figure 3-2, the shared logic block is contained in the example design. The name of the generated core is `<component_name>`.

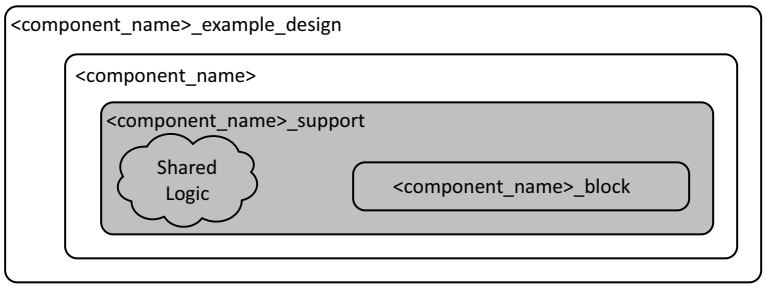

Figure 3-1: **Shared Logic Included in the Core**

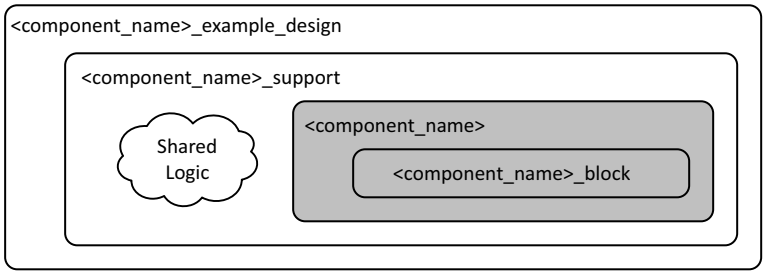

Figure 3-2: **Shared Logic Included in the Example Design**

The difference between the two hierarchies is the boundary of the core. The hierarchy is controlled using the **Shared Logic** option in the IP catalog (see Figure 4-2).

Clocking

This section describes the clocking scheme implemented in the core. There are three main clock inputs to this core, namely the 200 MHz free running clock, the GMII transmit clock, and the RGMII receive clock.

Free Running Clock

The 200 MHz free running clock is used as a reference clock to the IDELAYCTRL primitive (when enabled in the core), input to the GMII TX clock generator module (when GMII clock is sourced internally), and clock the management modules. The **Shared Logic** option determines how this clock is handled in the core.

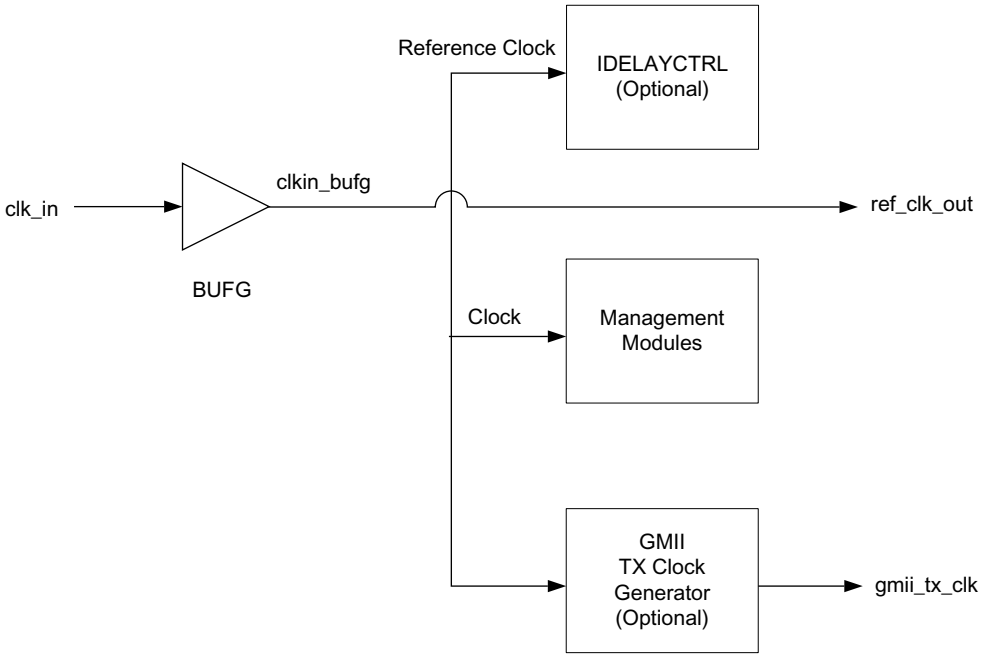

Figure 3-3: **When Shared Logic in Core is Selected**

Figure 3-3 depicts the clocking scheme when the **Shared Logic in Core** option is selected. The 200 MHz free running clock, shown as `clk_in`, is routed through the global clock buffer (BUFG) and then used in the core. It is also an output of this core that can be used by other instances of the GMII to RGMII cores that are configured for the shared logic to be present in the example design. This allows sharing of the clock resources.

GMII Transmit Clock

This clock is used by the transmit circuitry of the core. It is also an output from this core and it is used as the TX clock on the GEM GMII interface. The **Shared Logic** option determines if clock resources are used for this clock. If shared logic is in the core, then clock resources are used. If the shared logic is in the example design, then no clocking resources are used.

This clock can be sourced externally or can be generated internally within the core. The VHDL generic C_EXTERNAL_CLOCK is used to indicate this choice. The following paragraphs describe the clocking scheme in both the cases, that is, when shared logic is in the core and when shared logic is in the example design.

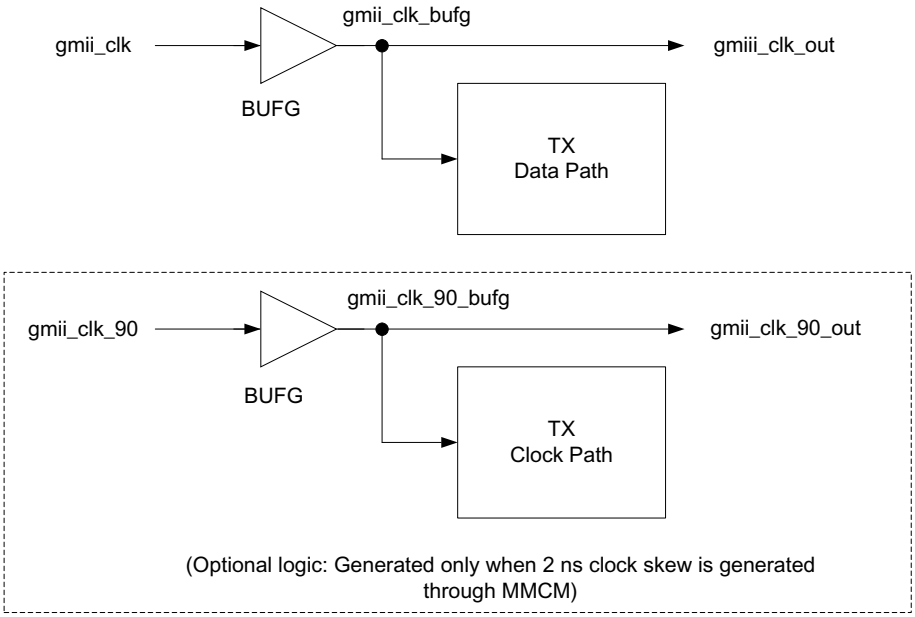

Figure 3-4: **GMII Clock Sourced Externally and Shared Logic in Core**

Figure 3-4 depicts the clocking scheme when the GMII clock is sourced externally and the **Shared Logic in Core** option is selected. This mode is active when the VHDL generic C_EXTERNAL_CLOCK is set to 1. In this case you must ensure that the GMII clock frequency is appropriate for the line rate. That is, it should be 2.5 MHz for 10 Mb/s, 25 MHz for 100 Mb/s and 125 MHz for 1,000 Mb/s. The external GMII clock is routed through the global clock buffer (BUFG) and then used in the core.

The clock, `gmii_clk_out`, is also an output of this IP that can be used by other instances of the GMII to RGMII cores which are configured for the shared logic to be present in the example design. This allows sharing of the clock resources.

`gmii_clk_90` is `gmii_clock` phase shifted by 90°. The logic in the dotted area is generated only if option to add 2 ns skew on `rgmii_txc` by using MMCM is selected.

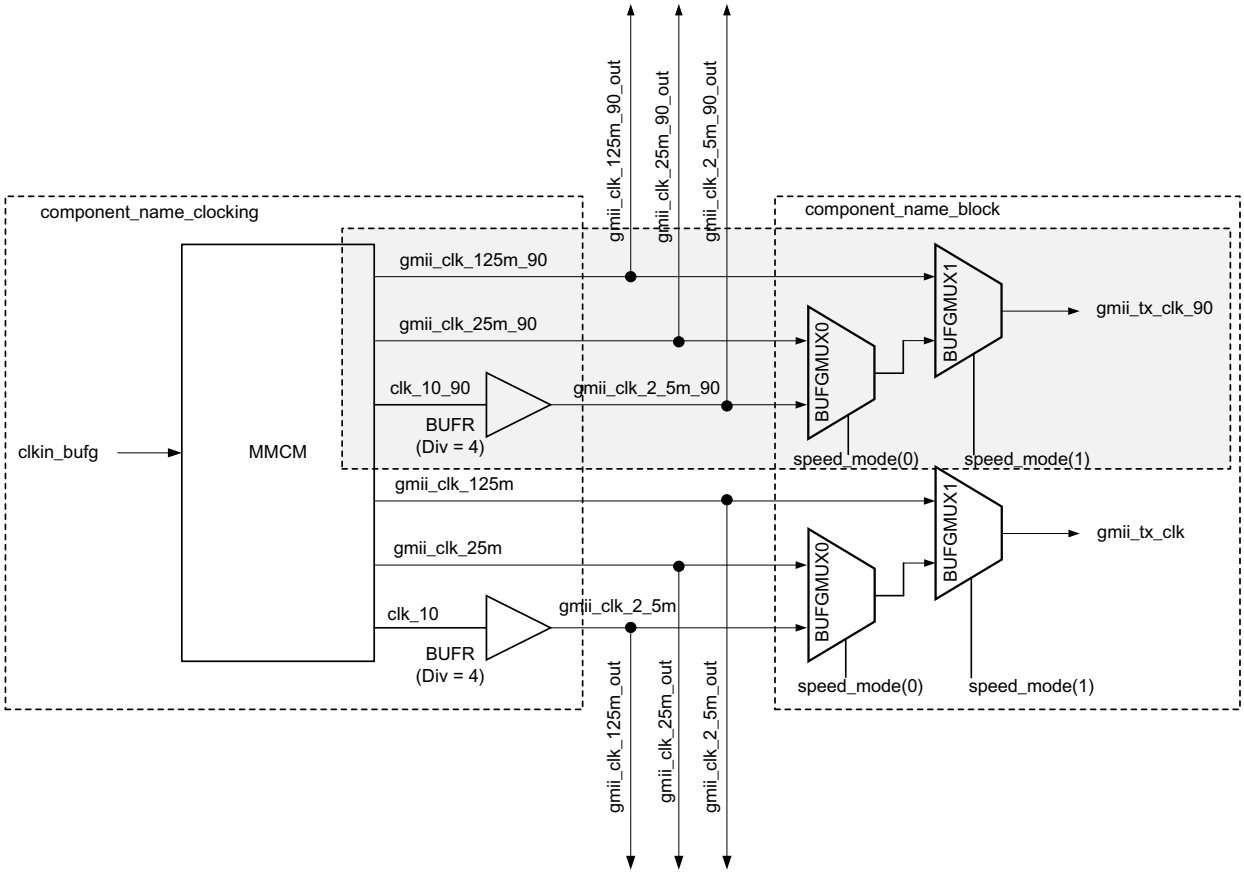

Figure 3-5: **GMII Clock Sourced Internally and Shared Logic in Core**

Figure 3-5 depicts the clocking scheme when the GMII clock is sourced internally and the **Shared Logic in Core** option is selected. This mode is active when the VHDL generic C_EXTERNAL_CLOCK is set to 0. The GMII to RGMII core has a built-in clock generator for providing 2.5 MHz, 25 MHz, and 125 MHz frequency clocks for 10 Mb/s, 100 Mb/s, and 1 Gb/s speeds of operation, respectively. The clock generator uses an MMCM to generate the clocks for all three line-rates. The block wrapper has the clock multiplexers (BUFGMUX) that are used to switch between three different clock frequencies for the three different speed modes of the MAC. The `speed_mode(0)` and `speed_mode(1)` signals are used as selection pins of BUFGMUX0 and BUFGMUX1 respectively.

The clocks shown in the gray area are only generated when option to add 2 ns skew on `rgmii_txc` through MMCM. These clocks are generated to have a phase shift of 90°.

The three clocks, `gmii_clk_2_5m_out`, `gmii_clk_25m_out`, `gmii_clk_125m_out`, are also outputs of this IP core that can be used by other instances of the GMII to RGMII cores which are configured for the shared logic to be present in the example design. This allows sharing of the clock resources.

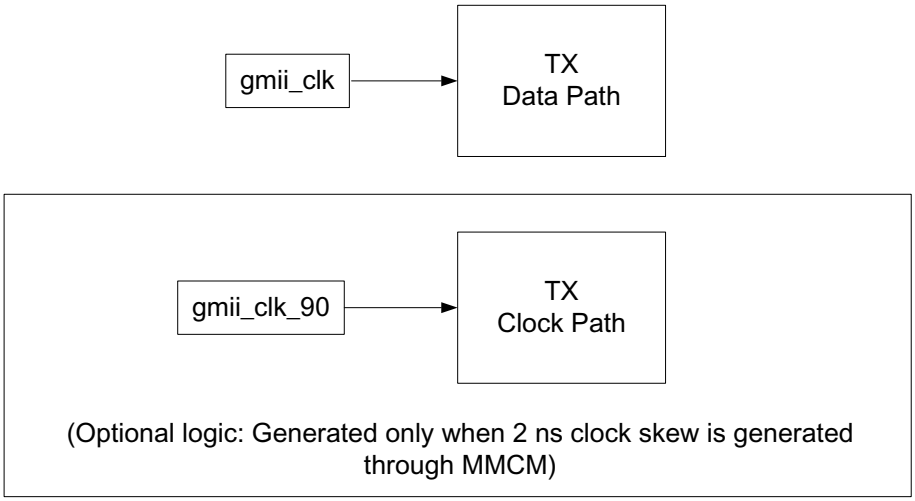

Figure 3-6: **GMII Clock Sourced Externally and Shared Logic in Example Design**

Figure 3-6 depicts the clocking scheme when the GMII clock is sourced externally and the **Shared Logic in Example Design** is selected. In this case no clocking resources are used by the core. The `gmii_clk` signal is the output of another GMII to RGMII core which is configured for the shared logic to be present in the core.

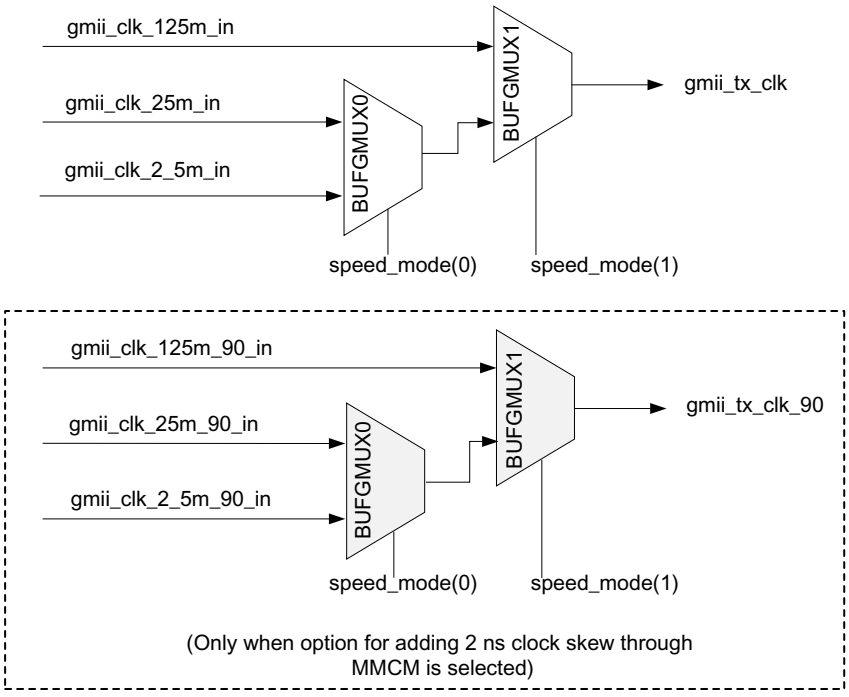

Figure 3-7: **GMII Clock Sourced Internally and Shared Logic in Example Design**

Figure 3-7 depicts the clocking scheme when the GMII clock is sourced internally and the **Shared Logic in Example Design** is selected. In this case the only clock resources used by the core are the two clock multiplexers. The block wrapper has the clock multiplexers (BUFGMUX) that are used to switch between three different clock frequencies for the three different speed modes of the MAC. The `speed_mode(0)` and `speed_mode(1)` signals are used as selection pins of BUFGMUX0 and BUFGMUX1 respectively.

RGMII Transmit Clock to the External PHY Device

Figure 3-8 depicts how the RGMII TX clock output to the External PHY device is generated using an Output DDR buffer (ODDR).The RGMII v2.0 standard specifies that the TX clock to have a setup of 2 ns with respect to the TX data. This core gives you an option to either enable or disable this 2 ns skew between the TX clock and TX data. If the skew is enabled (RGMII_TXC_SKEW = 1), the RGMII TX clock is passed through the ODELAY2 primitive. If the skew is enabled (RGMII_TXC_SKEW = 2), the 90° phase shifted `gmii_tx_clk` is used and the output of ODDR is given out as `rgmii_txc`.

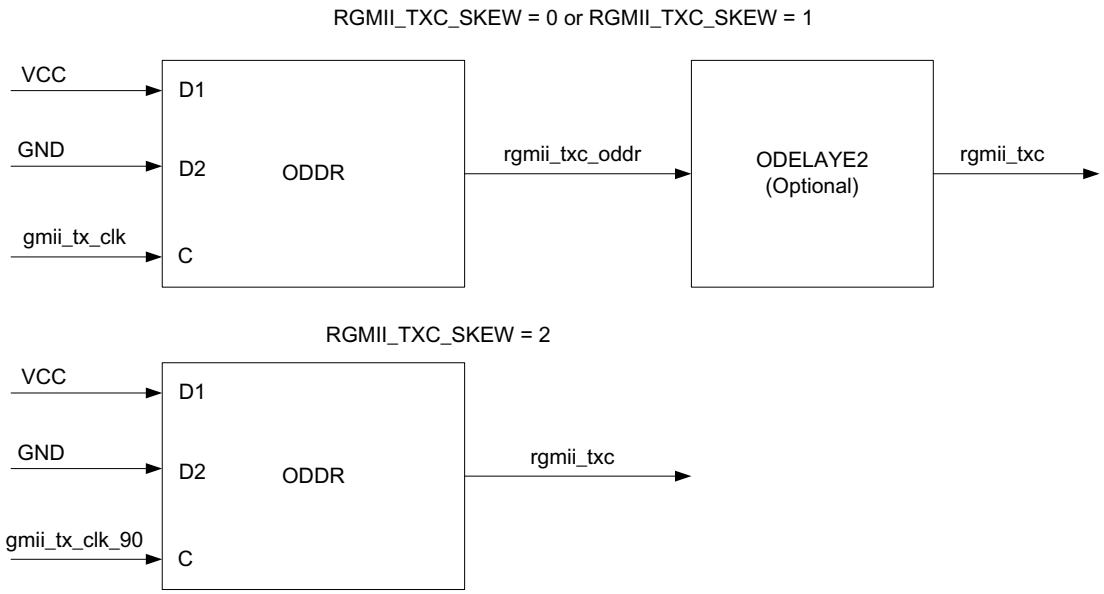

Figure 3-8: **RGMII Transmit Clock to the External PHY Device**

RGMII Receive Clock from the External PHY Device

Figure 3-9 depicts the clocking scheme for RGMII RX clock input. The `rgmii_rxc` clock input is driven through BUFR and then a BUFG which is used to clock the RX datapath in the core. This clock is also an output from this IP and it is used as the RX clock on the GEM GMII interface.

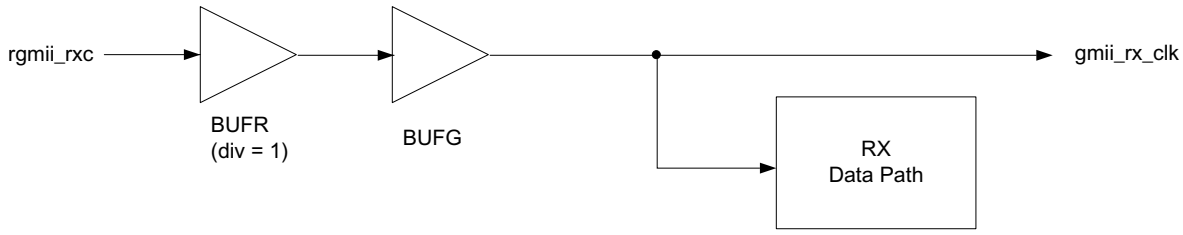

Figure 3-9: **RGMII Receive Clock from the External PHY Device**

Using Multiple Instances of the Core

Figure 3-10 shows how multiple instances of this core should be connected in a design so that clock resources can be saved using the shared logic feature.

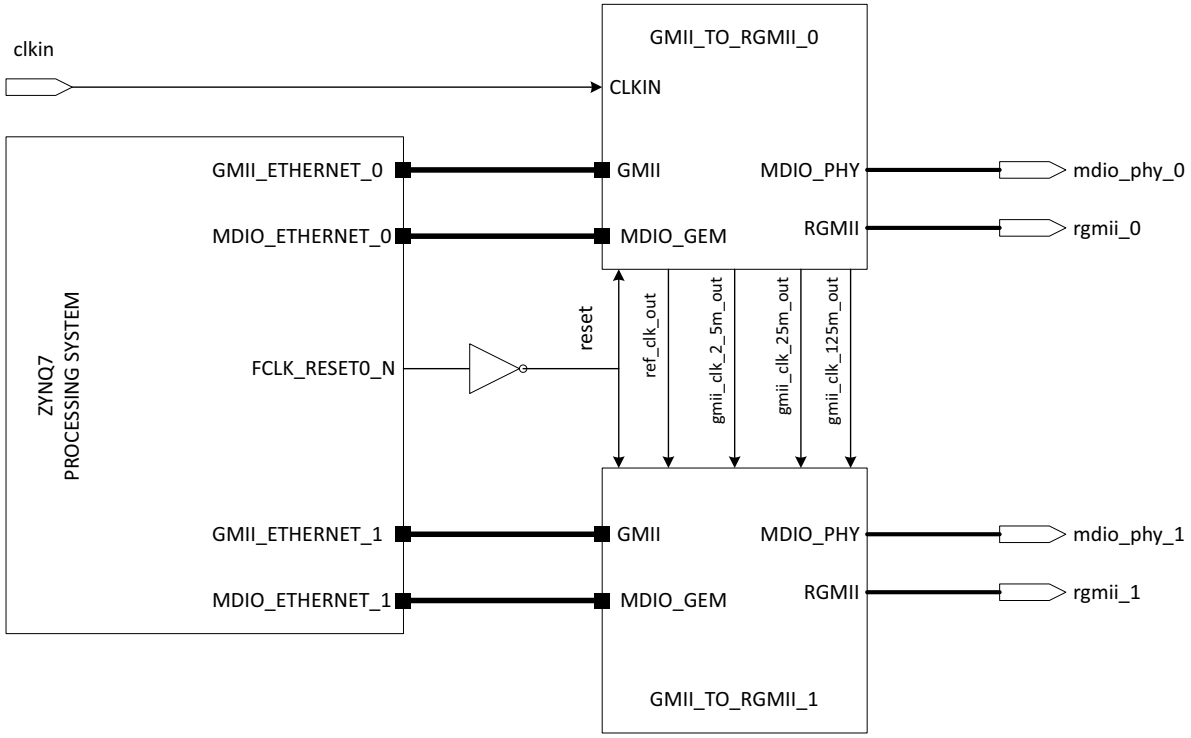

Figure 3-10: **Multiple instances of GMII to RGMII Core in a Design**

Both the core instances are configured for the GMII TX clock to be sourced internally. For example, GMII_TO_RGMII_0, shared logic is in the core and for IP instance GMII_to_RGMII_1, shared logic is in the example design (which means the IP block level does not have any shared clock/reset resources). Instance GMII_TO_RGMII_0 generates all the required clocks, namely `ref_clk` and the GMII TX clock. These clocks are used by instance GMII_TO_RGMII_0 internally and also shared with instance GMII_TO_RGMII_1.

Resets

Figure 3-11 shows the reset structure for the GMII to RGMII core.

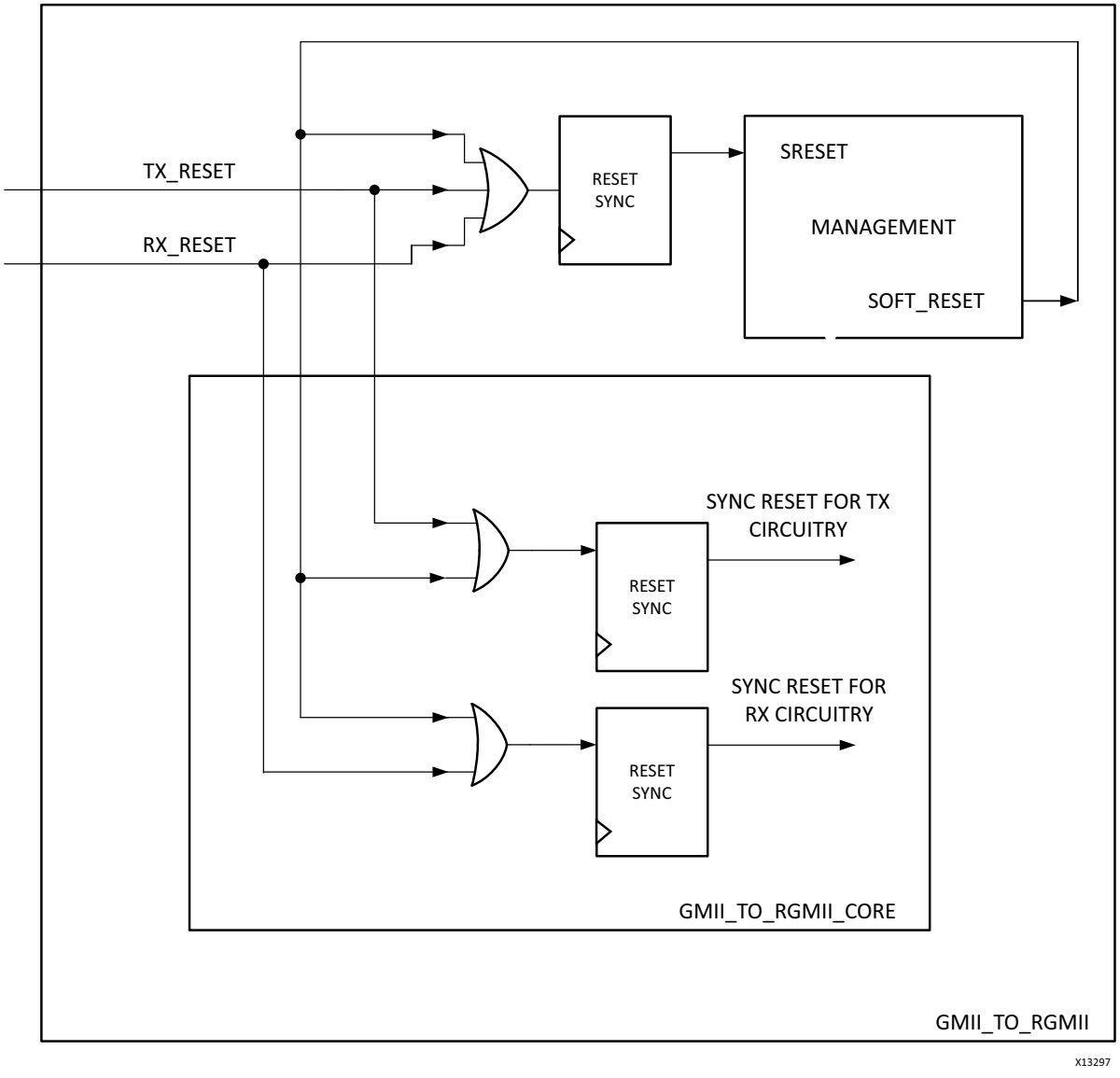

Figure 3-11: **GMII to RGMII Reset Structure**

Protocols

GMII Transmission

This section includes figures that illustrate GMII transmission. See Figure 3-12 through Figure 3-17.

Normal Frame Transmission

Normal outbound frame transfer timing is illustrated in Figure 3-12. This figure shows that an Ethernet frame is preceded by an 8-byte preamble field (inclusive of the Start of Frame Delimiter (SFD), and completed with a 4-byte Frame Check Sequence (FCS) field. This frame is created by the Ethernet MAC connected to the other end of the GMII.

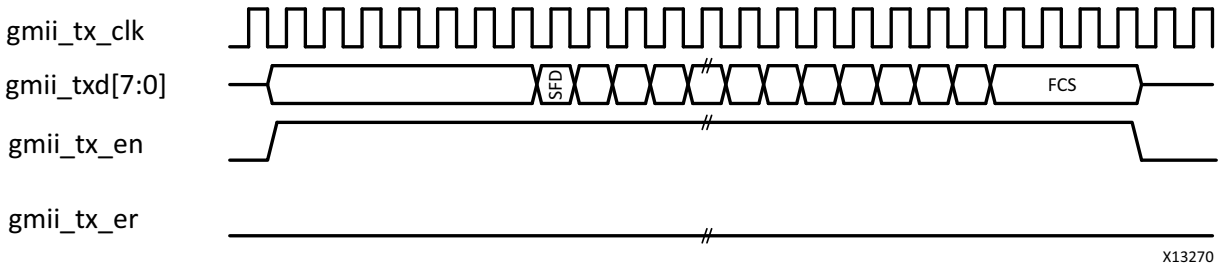

Figure 3-12: **Normal Frame Transmission**

Error Propagation

A corrupted frame transfer is illustrated in Figure 3-13. An error can be injected into the frame by asserting `gmii_tx_er` at any point during the `gmii_tx_en` assertion window.

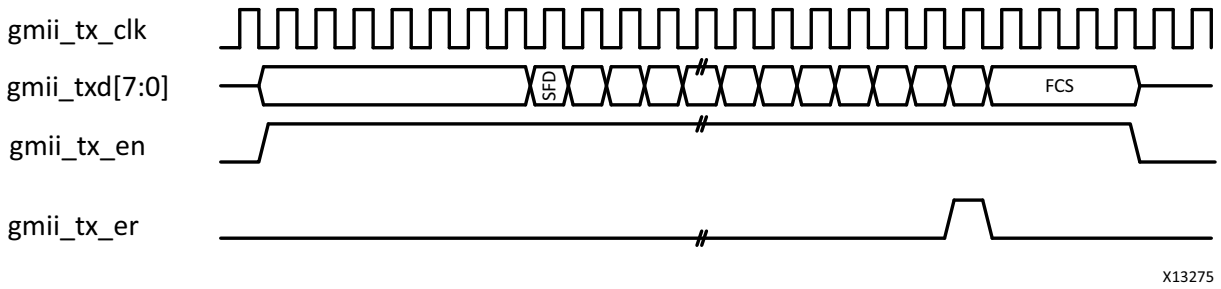

Figure 3-13: **GMII Error Propagation Within a Frame**

GMII Reception

This section includes figures that illustrate GMII reception.

Normal Frame Reception

The timing of normal inbound frame transfer is illustrated in Figure 3-14. This shows that Ethernet frame reception is preceded by a preamble field. The *IEEE 802.3-2012* Specification [Ref 2] (see clause 35) allows for up to all of the seven preamble bytes that precede the Start of Frame Delimiter (SFD) to be lost in the network. The SFD is always present in well-formed frames.

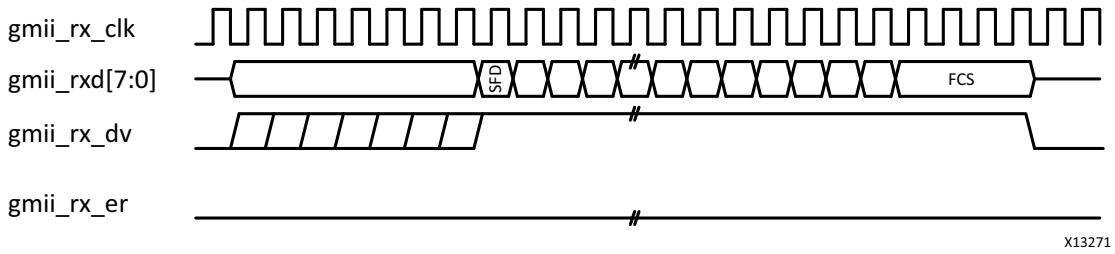

Figure 3-14: **GMII Normal Frame Reception**

Frame Reception with Errors

The signal `gmii_rx_er` when asserted within the assertion window signals that a frame was received with a detected error (Figure 3-15).

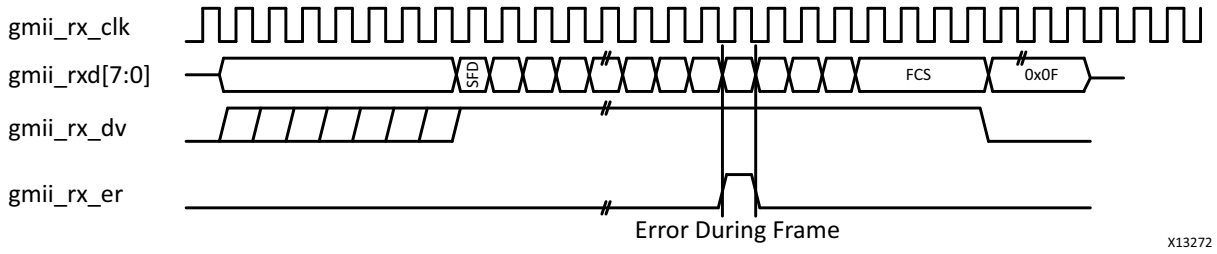

Figure 3-15: **GMII Frame Reception with Errors**

MII Transmission – 10/100 Mb/s Frame

The operation is similar to GMII transmission. In this case only the lower four bits (`gmii_txd[3:0]`) are valid.

MII Reception – 10/100 Mb/s Frame

The operation is similar to GMII reception. In this case only the lower four bits (`gmii_rxd[3:0]`) are valid.

RGMII Interface Protocols

The RGMII is intended as an alternative to the IEEE Std 802.3-2012[Ref 2] Clauses 22 and 35 (MII) and Clauses 34–39, 41–42 (GMII), and the TBI. The principle objective is to reduce the number of pins required to connect the Ethernet MAC and the PHY from a maximum of 28 pins (TBI) to 12 pins in a cost-effective and technology-independent manner. To accomplish this objective, the datapaths and all associated control signals are reduced and control signals multiplexed together, and both edges of the clock are used. For Gigabit operation, the clocks operate at 125 MHz, and for 10/100 operation, the clocks operate at 2.5 MHz and 25 MHz, respectively.

RGMII Transmission and Reception

Normal outbound frame transfer timing is illustrated in Figure 3-16.

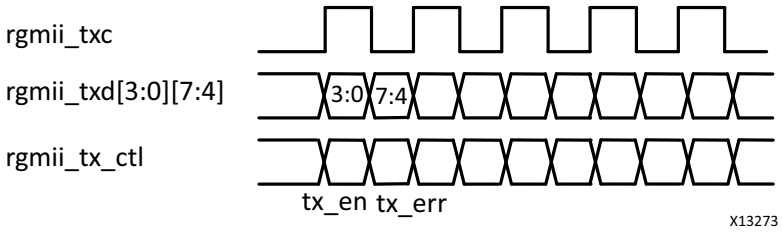

Figure 3-16: **RGMII Normal Frame Transmission**

Normal inbound frame transfer timing is illustrated in Figure 3-17.

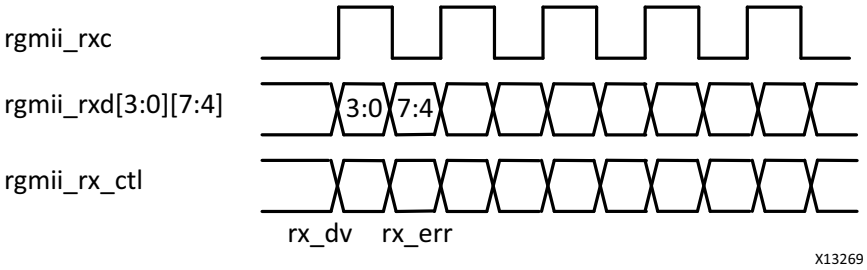

Figure 3-17: **RGMII Normal Frame Reception**

Multiplexing of data and control information is done by using both edges of the reference clocks and sending the lower 4 bits on the rising edge and the upper 4 bits on the falling edge. Control signals can be multiplexed into a single clock cycle using thew same technique.

Design Flow Steps

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

- *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 3]
- *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 5]
- *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 6]
- *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 9]

Customizing and Generating the Core

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

If you are customizing and generating the core in the IP integrator, see the *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 3] for detailed information. IP integrator might auto-compute certain configuration values when validating or generating the design. To determine whether the values change, see the description of the parameter in this chapter. To view the parameter value, run the `validate_bd_design` command in the Tcl console.

The GMII to RGMII core is generated using the IP catalog.

All parameters available for the core in the IP catalog are also available in IP integrator. The Vivado Integrated Design Environment (IDE) is identical, so settings that apply to the core in IP catalog also apply for IP integrator.

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

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

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 5] and the *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 6].

Note: Figures in this chapter are illustrations of the Vivado IDE. This layout might vary from the current version.

Figure 4-1 displays the GMII to RGMII customization screen used to set core parameters and options.

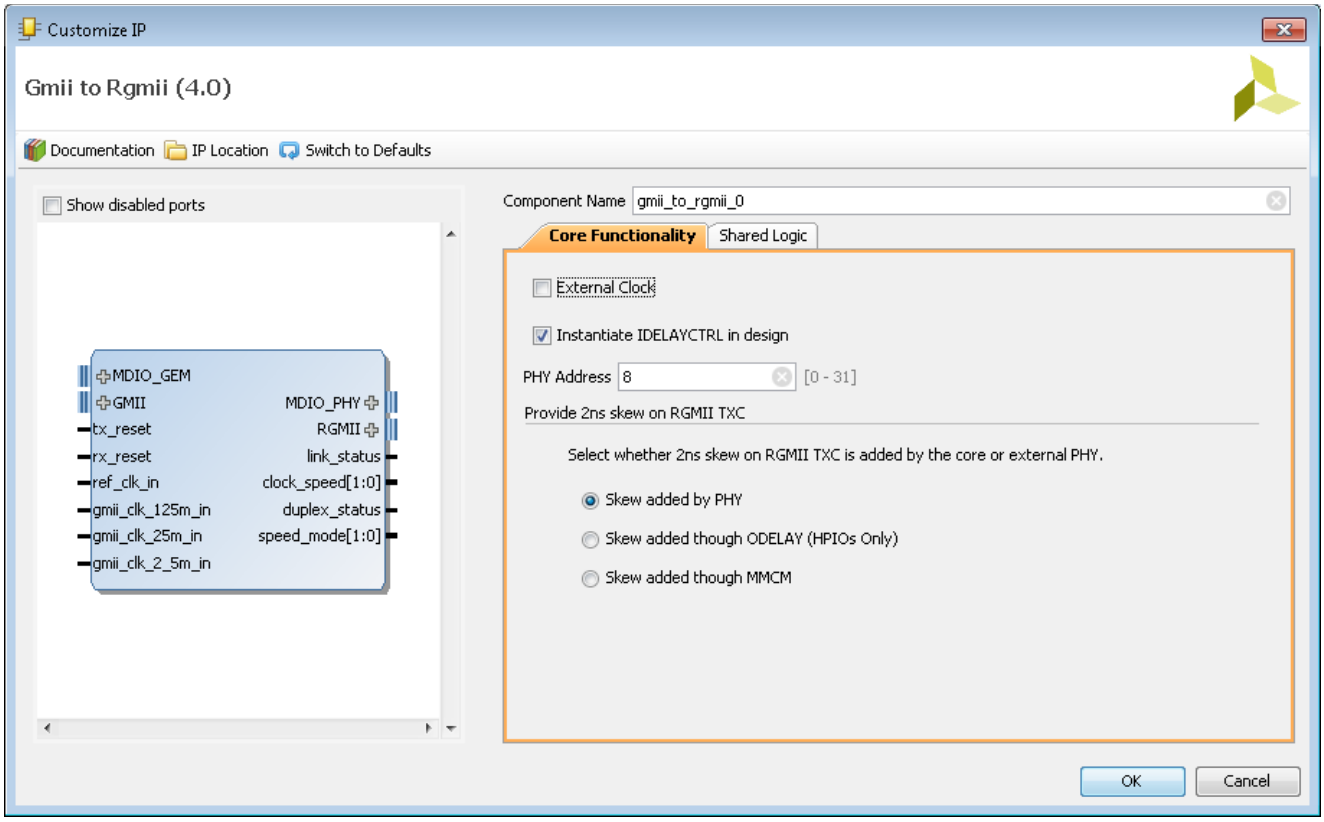

Figure 4-1: **Core Customization Screen**

- **Component Name** – The component name is used as the base name of the output files generated for the core. Names must begin with a letter and can be composed of the following characters: a through z, 0 through 9, and the underscore (_).
- **External Clock** – Select this option to source the GMII clock externally. When selected, ensure that the GMII clock frequency is appropriate for the line rate: 2.5 MHz for 10 Mb/s, 25 MHz for 100 Mb/s, and 125 MHz for 1,000 Mb/s.

  By default, the GMII clock is generated internally. The GMII to RGMII IP has a built-in clock generator for providing 2.5 MHz, 25 MHz, and 125 MHz frequency clocks for 10 Mb/s, 100 Mb/s, and 1,000 Mb/s speeds of operation, respectively.

- **Instantiate IDELAYCTRL in the Design** – Select this option to instantiate an IDELAYCTRL primitive in the core. The IDELAYCTRL primitive should be instantiated when Input/Output Delay primitives are used in the design. This core uses them and thus this option is selected by default.

  If your design has an IDELAYCTRL primitive instantiated for the I/O bank to which the RGMII I/Os are also mapped, then you should not select this option.

  The IDELAYCTRL primitive is instantiated as part of the Shared Logic and is part of the core in the `Include Shared Logic in the Core` configuration. If only one instance of the core exists in the `Include Shared Logic in the Example Design` configuration, the IDELAYCTRL primitive needs to be instantiated in the design.

- **PHY Address** – The PHY Address is the 5-bit address used to identify the core in a MDIO transaction. Valid ranges are 0 to 31. The PHY address here must be different from the address assigned to the onboard PHY.
- **Provide 2 ns Skew on RGMII TXC** – Select this option to choose the location where 2 ns skew on RGMII TXC is added with respect to RGMII TXD. If the skew is added in the FPGA, it can be by ODELAYs (present in devices HPIOs only) or through MMCM.

IMPORTANT: *This option of using ODELAY can be used only when the RGMII TXC pin is mapped to a high performance I/O Bank on the device.*

Shared Logic Options

Figure 4-2 displays the shared logic options in the Vivado IDE.

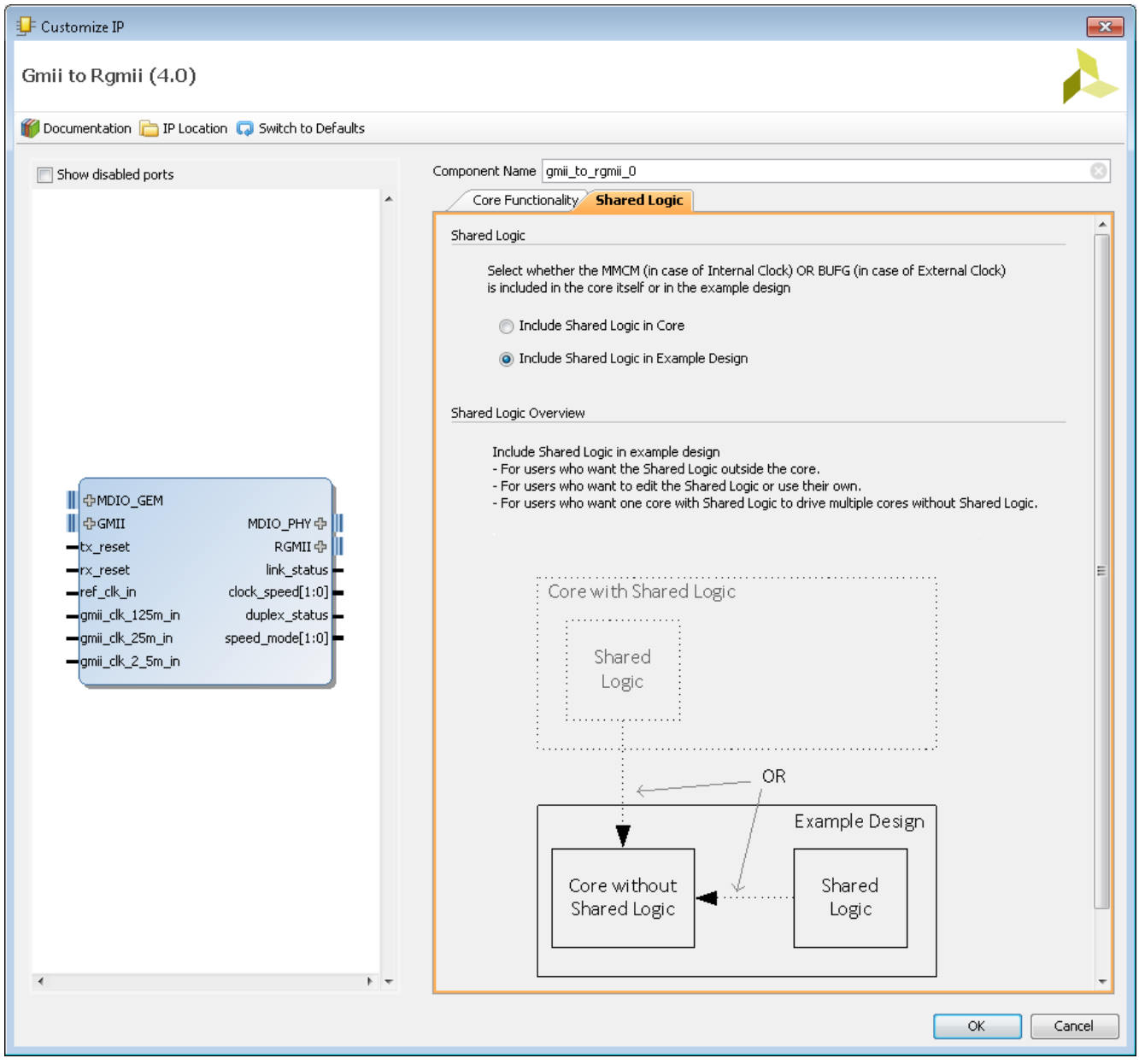

Figure 4-2: **Shared Logic Options**

- **Include Shared Logic in the Core** – Select this option to include the shared clocking resource as part of the core itself. Use this option when only one instance of the core is used in the design, or when this core instance is the first of multiple cores to be generated.

- **Include Shared Logic in the Example Design** – Select this option when the shared clocking resources are not required in the core. Use this option only when multiple instances of the core are used in the design, and the first core instance is already generated. This preserves clocking resources.

User Parameters

Table 4-1 shows the relationship between the GUI fields in the Vivado IDE and the User Parameters (which can be viewed in the Tcl Console).

Table 4-1: **Vivado IDE Parameter to User Parameter Relationship**

Vivado IDE Parameter/Value[(1)]	User Parameter/Value[(1)]	Default Value[(1)]
External Clock	C_EXTERNAL_CLOCK	FALSE
Instantiate IDELAYCTRL in design	C_USE_IDELAY_CTRL	TRUE
PHY Address	C_PHYADDR	8
Provide 2 ns skew on RGMII TXC	RGMII_TXC_SKEW	0
Shared Logic	SupportLevel	Include_Shared_Logic_in_Example_Design

Notes:

1. Parameter values are listed in the table where the Vivado IDE parameter value differs from the user parameter value. Such values are shown in this table as indented below the associated parameter.

Output Generation

The GMII to RGMII solution delivers files to several file groups. By default, the file groups required for use of the GMII to RGMII or for opening the IP example design are generated when the core is generated.

The file groups generated are found in the IP Sources tab of the Sources window, where they are listed for each IP in the project. For details, see “Generating IP Output Products” in the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 5].

Constraining the Core

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

Required Constraints

The GMII to RGMII core requires design constraints to guarantee maximum performance.

Vivado Design Suite

These constraints should be placed in an XDC file at the top level of the design. The example of the constraint text shown in the following paragraphs is based on the port names of the GMII to RGMII core. If these ports are mapped to FPGA pin names that are different, the FPGA pin names should be submitted for the port names in the following example.

```
# Clock Period Constraints
create_clock -period 5.000 -name clkin -add [get_nets clkin]
create_clock -period 8.000 -name rgmii_rxc -add [get_ports rxc]

# Clock constraint if parameter C_EXTERNAL_CLOCK = 1
create_clock -add -name gmii_clk  -period 8.000 [get_ports gmii_clk]

# Clock constraint if parameter C_EXTERNAL_CLOCK = 1 and clock skew on TXC is through
MMCM
create_clock -add -name gmii_clk_90  -period 8.000 -waveform {2 6} [get_ports
gmii_clk_90]

#False path constraints to async inputs coming directly to synchronizer
set_false_path -to [get_pins -hier -filter {name =~ *idelayctrl_reset_gen/
*reset_sync*/PRE }]
set_false_path -to [get_pins -of [get_cells -hier -filter { name =~ *i_MANAGEMENT/
SYNC_*/data_sync* }] -filter { name =~ *D }]
set_false_path -to [get_pins -hier -filter {name =~ *reset_sync*/PRE }]

#False path constraints from Control Register outputs
set_false_path -from [get_pins -hier -filter {name =~ *i_MANAGEMENT/
DUPLEX_MODE_REG*/C }]
set_false_path -from [get_pins -hier -filter {name =~ *i_MANAGEMENT/
SPEED_SELECTION_REG*/C }]

# constraint valid if parameter C_EXTERNAL_CLOCK = 0
set_case_analysis 0 [get_pins -hier -filter {name =~ *i_bufgmux_gmii_clk/CE0}]
set_case_analysis 0 [get_pins -hier -filter {name =~ *i_bufgmux_gmii_clk/S0}]
set_case_analysis 1 [get_pins -hier -filter {name =~ *i_bufgmux_gmii_clk/CE1}]
set_case_analysis 1 [get_pins -hier -filter {name =~ *i_bufgmux_gmii_clk/S1}]

# constraint valid if parameter C_EXTERNAL_CLOCK = 0 and clock skew on TXC is through
MMCM
set_case_analysis 0 [get_pins -hier -filter {name =~ *i_bufgmux_gmii_90_clk/CE0}]
set_case_analysis 0 [get_pins -hier -filter {name =~ *i_bufgmux_gmii_90_clk/S0}]
set_case_analysis 1 [get_pins -hier -filter {name =~ *i_bufgmux_gmii_90_clk/CE1}]
set_case_analysis 1 [get_pins -hier -filter {name =~ *i_bufgmux_gmii_90_clk/S1}]

#IO Placement
set_property PACKAGE_PIN D18 [get_ports clkin_p]
set_property PACKAGE_PIN C19 [get_ports clkin_n]

set_property IOSTANDARD LVDS_25 [get_ports clkin_p]
set_property IOSTANDARD LVDS_25 [get_ports clkin_n]

set_property PACKAGE_PIN A18 [get_ports phy_reset_n]
set_property IOSTANDARD LVCMOS18 [get_ports phy_reset_n]

set_property PACKAGE_PIN N20 [get_ports rgmii_tx_ctl]
```

```
set_property PACKAGE_PIN K19 [get_ports rgmii_txc]
set_property PACKAGE_PIN D20 [get_ports {rgmii_txd[0]}]
set_property PACKAGE_PIN N19 [get_ports {rgmii_txd[1]}]
set_property PACKAGE_PIN K20 [get_ports {rgmii_txd[2]}]
set_property PACKAGE_PIN L21 [get_ports {rgmii_txd[3]}]
set_property PACKAGE_PIN M22 [get_ports rgmii_rx_ctl]
set_property PACKAGE_PIN J20 [get_ports rgmii_rxc]
set_property PACKAGE_PIN L22 [get_ports {rgmii_rxd[0]}]
set_property PACKAGE_PIN M21 [get_ports {rgmii_rxd[1]}]
set_property PACKAGE_PIN K21 [get_ports {rgmii_rxd[2]}]
set_property PACKAGE_PIN N17 [get_ports {rgmii_rxd[3]}]

set_property IOSTANDARD LVCMOS18 [get_ports rgmii_tx_ctl]
set_property IOSTANDARD LVCMOS18 [get_ports rgmii_txc]
set_property IOSTANDARD LVCMOS18 [get_ports {rgmii_txd[0]}]
set_property IOSTANDARD LVCMOS18 [get_ports {rgmii_txd[1]}]
set_property IOSTANDARD LVCMOS18 [get_ports {rgmii_txd[2]}]
set_property IOSTANDARD LVCMOS18 [get_ports {rgmii_txd[3]}]
set_property IOSTANDARD LVCMOS18 [get_ports rgmii_rx_ctl]
set_property IOSTANDARD LVCMOS18 [get_ports rgmii_rxc]
set_property IOSTANDARD LVCMOS18 [get_ports {rgmii_rxd[0]}]
set_property IOSTANDARD LVCMOS18 [get_ports {rgmii_rxd[1]}]
set_property IOSTANDARD LVCMOS18 [get_ports {rgmii_rxd[2]}]
set_property IOSTANDARD LVCMOS18 [get_ports {rgmii_rxd[3]}]

set_property PACKAGE_PIN B19 [get_ports mdio_phy_mdc]
set_property PACKAGE_PIN C20 [get_ports mdio_phy_mdio]

set_property IOSTANDARD LVCMOS18 [get_ports mdio_phy_mdc]
set_property IOSTANDARD LVCMOS18 [get_ports mdio_phy_mdio]

#To Adjust GMII Tx Input Setup/Hold Timing
set_property IDELAY_VALUE 16 [get_cells *delay_rgmii_rx_ctl]
set_property IDELAY_VALUE 16 [get_cells -hier -filter {name =~ *delay_rgmii_rxd*}]
set_property IODELAY_GROUP gpr1 [get_cells *delay_rgmii_rx_ctl]
set_property IODELAY_GROUP gpr1 [get_cells -hier -filter {name =~
*delay_rgmii_rxd*}]
set_property IODELAY_GROUP gpr1 [get_cells *idelayctrl]
```

If the option is selected to provide 2 ns skew to RGMII TXC in devices supporting HPIO banks, these constraints can be added to fine tune the delay:

```
#To adjust RGMII TXC delay when 2 ns skew option is selected
set_property ODELAY_VALUE  "26"   [get_cells -hier -filter {name =~ *_core/
*oddr_rgmii_txc}]
set_property ODELAY_VALUE  "0"    [get_cells -hier -filter {name =~ *_core/
*oddr_rgmii_tx_ctl*}]
set_property ODELAY_VALUE  "0"    [get_cells -hier -filter {name =~ *_core/
*oddr_rgmii_txd*}]
set_property IODELAY_GROUP "gpr1" [get_cells -hier -filter {name =~ *_core/
*oddr_rgmii_txc}]
set_property IODELAY_GROUP "gpr1" [get_cells -hier -filter {name =~ *_core/
*oddr_rgmii_tx_ctl*}]
set_property IODELAY_GROUP "gpr1" [get_cells -hier -filter {name =~ *_core/
*oddr_rgmii_txd*}]
```

```
# Use these constraints to modify input delay on RGMII signals
set_input_delay -clock [get_clocks <core_instance>_rgmii_rx_clk] -max -1.5
[get_ports {rgmii_rxd[*] rgmii_rx_ctl}]
set_input_delay -clock [get_clocks <core_instance>_rgmii_rx_clk] -min -2.8
[get_ports {rgmii_rxd[*] rgmii_rx_ctl}]
set_input_delay -clock [get_clocks <core_instance>_rgmii_rx_clk] -clock_fall -max
-1.5 -add_delay [get_ports {rgmii_rxd[*] rgmii_rx_ctl}]
set_input_delay -clock [get_clocks <core_instance>_rgmii_rx_clk] -clock_fall -min
-2.8 -add_delay [get_ports {rgmii_rxd[*] rgmii_rx_ctl}]

# Use these constraints to modify output delay on RGMII signals if 2ns delay is added
by the core
set_output_delay 0.75 -max -clock [get_clocks rgmii_tx_clk] [get_ports {rgmii_txd[*]
rgmii_tx_ctl}]
set_output_delay -0.7 -min -clock [get_clocks rgmii_tx_clk] [get_ports {rgmii_txd[*]
rgmii_tx_ctl}]
set_output_delay 0.75 -max -clock [get_clocks rgmii_tx_clk] [get_ports {rgmii_txd[*]
rgmii_tx_ctl}] -clock_fall -add_delay
set_output_delay -0.7 -min -clock [get_clocks rgmii_tx_clk] [get_ports {rgmii_txd[*]
rgmii_tx_ctl}] -clock_fall -add_delay

# Use these constraints to modify output delay on RGMII signals if 2ns delay is added
by external PHY
set_output_delay -clock [get_clocks rgmii_tx_clk] -max -1.0 [get_ports {rgmii_txd[*]
rgmii_tx_ctl}]
set_output_delay -clock [get_clocks rgmii_tx_clk] -min -2.6 [get_ports {rgmii_txd[*]
rgmii_tx_ctl}] -add_delay
set_output_delay -clock [get_clocks rgmii_tx_clk] -clock_fall -max -1.0 [get_ports
{rgmii_txd[*] rgmii_tx_ctl}]
set_output_delay -clock [get_clocks rgmii_tx_clk] -clock_fall -min -2.6 [get_ports
{rgmii_txd[*] rgmii_tx_ctl}]
```

The above constraints can be modified to allow the tool to meet timing. Use the following constraint to modify the slew in the IOB:

```
set_property slew FAST [get_ports [list {rgmii_txd[3]} {rgmii_txd[2]} {rgmii_txd[1]}
{rgmii_txd[0]} rgmii_txc rgmii_tx_ctl]]
```

Device, Package, and Speed Grade Selections

The core can be implemented in a Zynq®-7000 All Programmable SoC device with these attributes:

- Is large enough to accommodate the core
- Contains a sufficient number of IOBs
- Has a supported speed grade (–1 or faster)

Clock Frequencies

Following are clock frequency requirements:

- `clkin` – 200 MHz

- `gmii_clk` – (Present when the GMII clock is sourced externally) 2.5/25/125 MHz for 10/100/1000 Mb/s, respectively
- `gmii_clk_90` – `gmii_clk` phase shifted by 90°

Clock Management

No information is currently provided for this core.

Clock Placement

No information is currently provided for this core.

Banking

All ports should be given location constraints appropriate to your design, within banking limits.

Transceiver Placement

Not applicable. This core does not use transceivers.

I/O Standard and Placement

According to the RGMII v2.0 specification, the I/O pins must use 1.5V HSTL interface voltages. Like the majority of PHYs, those selected for Xilinx development boards are multi-standard and operate at either 1.8V or 2.5V.

A Tcl script that checks the I/O standard of RGMII is included with the IP and can be sourced post-implementation. The Tcl script is located at `<project_dir>/<project_name>/<project_name>.srcs/sources1/ip/<component_name>/drc_check/io`.

Table 4-2 provides information on the I/O standards supported for different Zynq-7000 AP SoC devices that correspond to the Xilinx 7 series device programmable logic equivalent.

Table 4-2: **I/O Standards Supported**

Xilinx 7 Series Device Programmable Logic Equivalent	RGMII Signals Voltage Level Supported			Miscellaneous PHY Signals (MDIO, MDC, RESET) Voltage Level Supported		
	3.3V	**2.5V**	**1.8V**	**3.3V**	**2.5V**	**1.8V**
Artix®-7	No(1)	Yes(2)	Yes(3)	Yes(2)	Yes(2)	Yes(3)

Table 4-2: **I/O Standards Supported *(Cont'd)***

Xilinx 7 Series Device Programmable Logic Equivalent	RGMII Signals Voltage Level Supported			Miscellaneous PHY Signals (MDIO, MDC, RESET) Voltage Level Supported		
Kintex®-7	No[1]	Yes[2]	Yes[3]	Yes[2]	Yes[2]	Yes[3]

Notes:

1. High Range (HR) I/O duty cycle distortion exceeds RGMII specification.
2. Requires the use of HR I/O.
3. Limited 1.8V RGMII-only PHY devices are available. If one of these devices is not used, external voltage level shifting logic is required.

Simulation

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

IMPORTANT: *For cores targeting 7 series or Zynq-7000 AP SoC devices, UNIFAST libraries are not supported. Xilinx IP is tested and qualified with UNISIM libraries only.*

Synthesis and Implementation

This section contains information about synthesis and implementation in the Vivado Design Suite. For details about synthesis and implementation, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 5].

Example Design

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

The GMII to RGMII core is delivered with an example design that can be synthesized, complete with functional test benches.

External Clocking

Figure 5-1 illustrates the example design for top-level HDL when the GMII clock is sourced externally.

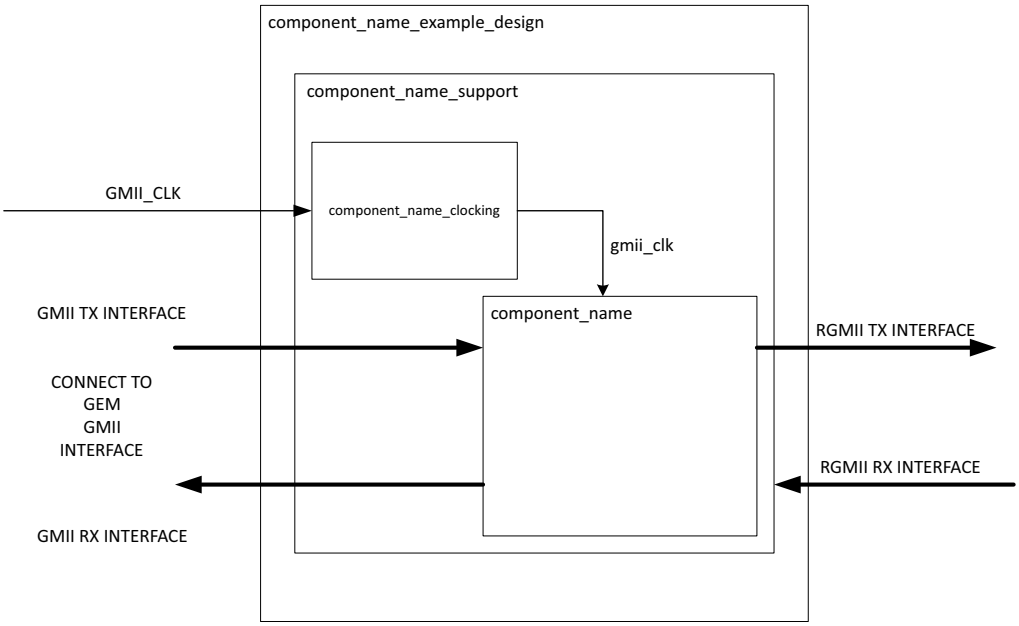

Figure 5-1: **Example Design HDL with External GMII Clock**

In this case, the `gmii_clk` is sourced externally. No additional clocking resources are used.

Internal Clocking

Figure 5-2 illustrates the example design for top-level HDL when the GMII clock is sourced internally.

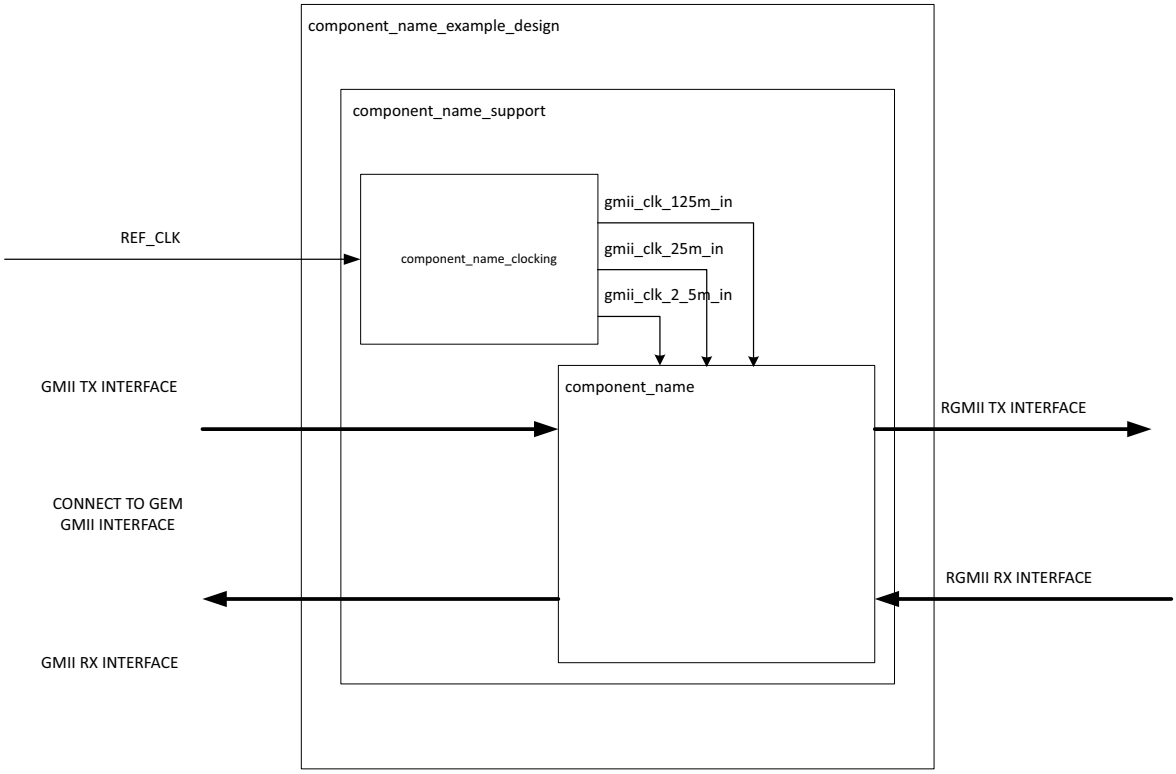

Figure 5-2: **Example Design HDL with Internal GMII Clock**

In both examples, the design is split into two hierarchical layers: block level and top-level. If shared logic is selected in the core, the top-level instantiates the block level from HDL. Otherwise, it instantiates the support level. The block level is designed so that it can be instantiated directly into customer designs and performs the following functions:

- Instantiates the core from HDL
- Connects the physical-side interface of the core to device IOBs, creating an external RGMII.

The top-level creates a specific example that can be simulated, synthesized, implemented, and if required, placed on a suitable board and demonstrated in hardware.

Top-Level Example Design HDL

VHDL Vivado® Design Suite:

```
<project_dir>/<project_name>/<project_name>.srcs/sources1/ip/
<component_name>/<component_name>/example_design/
<component_name>_example_design.vhd
```

Test Bench

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

Demonstration Test Bench

Figure 6-1 illustrates the demonstration test bench for the GMII to RGMII. The demonstration test bench is a simple VHDL program to test the core.

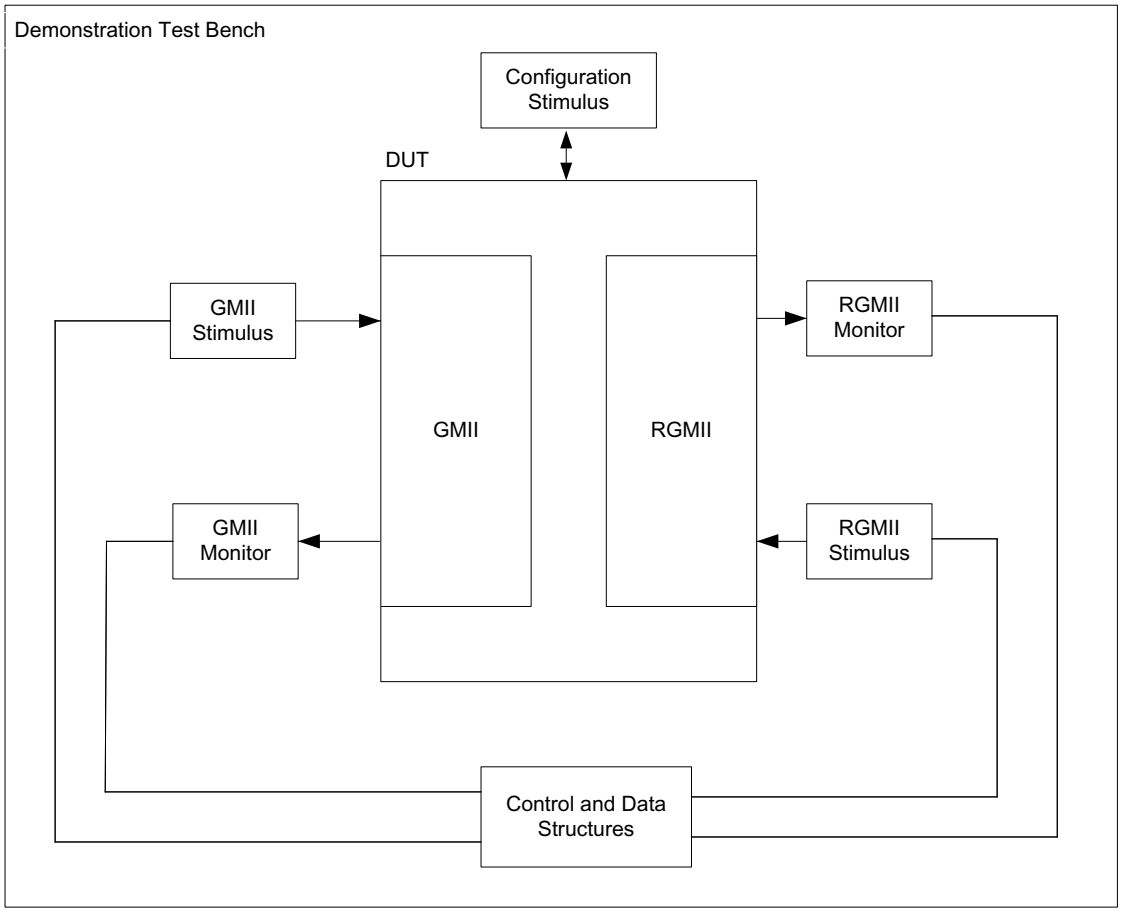

Figure 6-1: **GMII to RGMII Demonstration Test Bench**

The test bench entity instantiates the example design for the core, which is the Device Under Test (DUT). The example design by itself is a wrapper around the core. The test bench entity contains stimulus, clocks, resets, and test bench semaphores. The test bench is located at `<project_dir>/<project_name>/<project_name>.srcs/sources1/ip/<component_name>/simulation/demo_tb.vhd`.

The demonstration test bench performs the following tasks:

- Generates input clock signals.
- Applies a reset to the example design.
- Configures the GMII to RGMII core to operate at 1 Gb/s.
- Injects four frames into the GMII transmitter from the GMII stimulus block.
- Sets the first frame to the minimum length.
- Sets he second frame as a type frame.
- Sets the third frame as an errored frame.
- Sets the fourth frame as a padded frame.

These steps are repeated for 100 Mb/s and 10 Mb/s speeds.

Customizing the Test Bench

Changing Frame Data

You can change the contents of the four frames used by the demonstration test bench by changing the data and valid fields for each frame defined in the stimulus block. You can add new frames by defining a new frame of data. Modified frames are automatically updated in both stimulus and monitor functions.

Changing Frame Error Status

You can insert errors into any of the predefined frames in any position by setting the error field to 1 in any column of that frame. Injected errors are automatically updated in both stimulus and monitor functions.

Migrating and Upgrading

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

Migrating to the Vivado Design Suite

For information on migrating to the Vivado Design Suite, see the *ISE to Vivado Design Suite Migration Guide* (UG911) [Ref 7].

Upgrading in the Vivado Design Suite

In version 3.0 of the core, several changes make the core pin-incompatible with previous versions. These changes were required to enhance functionality.

Shared Logic

As part of the hierarchical changes to the core, it is now possible to have the core itself include all of the logic that can be shared between multiple cores, which was previously exposed in the example design for the core.

If you are updating a previous version to a new one with shared logic, there is no simple upgrade path and Xilinx® recommends to consult the Shared Logic sections of this document for more guidance.

Parameter Changes from v3.0 to v4.0

Table A-1: **Parameters Removed**

Parameter Name	Changes	What to do
C_RGMII_TXC_SKEW_EN	Parameter replaced with RGMII_TXC_SKEW	The previous mapping holds true for the new parameter.

Table A-2: **Parameters Added**

Parameter Name	Description	Default Value	Allowable Values
RGMII_TXC_SKEW	Provide 2 ns skew on RGMII TXC with respect to RGMII TXD	0	0 = rgmii_txc is edge aligned with respect to rgmii_txd (that is, 2 ns delay added by external PHY) 1 = rgmii_txc is delayed by 2 ns with respect to rgmii_txd by ODELAY (available only in devices having HPIOs) 2 = rgmii_txc is delayed by 2 ns with respect to rgmii_txd by MMCM

Port Changes from v3.0 to v4.0

Table A-3: **Port Changes**

Port Name and Width	I/O	Description	What to do
gmii_clk_90	Input	Present only when GMII clock is sourced externally (C_EXTERNAL_CLOCK = 1) and clock skew is added though MMCM (RGMII_TXC_SKEW = 2). This is gmii_clk phase shifted by 90°.	This should be driven either by the shared logic provided with the core, or by another cores shared logic block.
gmii_clk_125m_90_in	Input	Present only when GMII clock is sourced internally (C_EXTERNAL_CLOCK = 0) and clock skew is added though MMCM (RGMII_TXC_SKEW = 2) 125 MHz GMII TX clock phase shifted by 90° from the shared logic block to the core	This should be driven either by the shared logic provided with the core, or by another cores shared logic block.
gmii_clk_25m_90_in	Input	Present only when GMII clock is sourced internally (C_EXTERNAL_CLOCK = 0) and clock skew is added though MMCM (RGMII_TXC_SKEW = 2) 25 MHz GMII TX clock phase shifted by 90° from the shared logic block to the core.	This should be driven either by the shared logic provided with the core, or by another cores shared logic block.

Table A-3: **Port Changes *(Cont'd)***

Port Name and Width	I/O	Description	What to do
gmii_clk_2_5m_90_in	Input	Present only when GMII clock is sourced internally (C_EXTERNAL_CLOCK = 0) and clock skew is added though MMCM (RGMII_TXC_SKEW = 2) 2.5 MHz GMII TX clock phase shifted by 90° from the shared logic block to the core	This should be driven either by the shared logic provided with the core, or by another cores shared logic block.
mmcm_locked	Output	Valid only for Shared Logic in Core configuration and if the external clock option is not selected. This indicates that the MMCM has locked.	Can be left open if not used.

Parameter Changes from v2.0 to v3.0

Table A-4: **Parameters Modified**

Parameter Name	Changes	What to do
C_PHYADDR	Data type changed from std_logic_vector to integer	When mapping value to this parameter use integer data type for the corresponding PHY Address

Table A-5: **Parameters Added**

Parameter Name	Description	Default Value	Allowable Values
C_RGMII_TXC_SKEW_EN	Provide 2 ns skew on RGMII TXC with respect to RGMII TXD	0	0 = rgmii_txc is edge aligned with respect to rgmii_txd 1 = rgmii_txc is delayed by 2 ns with respect to rgmii_txd
C_RGMII_TXC_ODELAY_VAL	ODELAY value passed on to ODELAY2 primitive used to provide the skew on RGMII TXC	26	This is valid only when C_RGMII_TXC_SKEW_EN = 1. Specifies the ODELAY tap value to be used to provide skew on RGMII TXC

Port Changes from v2.0 to v3.0

Table A-6: **Ports Deleted**

Port Name and Width	In/Out	Description	What to do
clkin	Input	200 MHz Reference clock	This port is replaced by ref_clk_in. Map the 200 MHz clock from the shared logic block/external source to this port
mdio_mdc	Input	MDIO clock from GEM	This port is replaced by mdio_gem_mdc. Connect the GEM mdc port to this port.
mdio_i	Output	MDIO Input to GEM	This port is replaced by mdio_gem_i. Connect the GEM mdio_i port to this port.
mdio_o	Input	MDIO Output from GEM	This port is replaced by mdio_gem_o. Connect the GEM mdio_o port to this port.
mdio_t	Input	MDIO 3-state Input from GEM	This port is replaced by mdio_gem_t. Connect the GEM mdio_t port to this port.
mdc	Output	MDC clock to External PHY device	This port is replaced by mdio_phy_mdc. Connect the PHY MDC port to this port.
mdio	Bidirectional	MDIO data line to External PHY device	The core no longer instantiates the Bi-Di I/O primitive. Instead it gives out the _i, _o and _t signals which should be connected to a Bi-Di I/O primitive. This primitive has to be user-instantiated.

The following ports were added to the core (non-shared logic).

Table A-7: **Ports Added**

Port Name and Width	In/Out	Description	What to do
ref_clk_in	Input	200 MHz clock from the shared logic block to the core	This should be driven by one of following: • the shared logic provided with the core • by another cores shared logic block • from another cores shared logic block • from an external clock source
gmii_clk_125m_in	Input	Present only when GMII clock is sourced internally (C_EXTERNAL_CLOCK = 0) 125 MHz GMII TX clock from the shared logic block to the core	This should be driven either by the shared logic provided with the core, or by another cores shared logic block.
gmii_clk_25m_in	Input	Present only when GMII clock is sourced internally (C_EXTERNAL_CLOCK = 0) 25 MHz GMII TX clock from the shared logic block to the core	This should be driven either by the shared logic provided with the core, or by another cores shared logic block.
gmii_clk_2_5m_in	Input	Present only when GMII clock is sourced internally (C_EXTERNAL_CLOCK = 0) 2.5 MHz GMII TX clock from the shared logic block to the core	This should be driven either by the shared logic provided with the core, or by another cores shared logic block.
mdio_phy_mdc	Output	MDC clock to External PHY device	Connect this to External PHY device MDC input.
mdio_phy_i	Input	The _o output from the Bi-Di primitive on the MDIO Data line from the External PHY Device	Connect this to the _o port of the Bi-Di I/O primitive on the MDIO to the External PHY device.
mdio_phy_o	Output	The _i input to the Bi-Di primitive on the MDIO Data line from the External PHY Device	Connect this to the _i port of the Bi-Di I/O primitive on the MDIO to the External PHY device.
mdio_phy_t	Output	The _t input to the Bi-Di primitive on the MDIO Data line form the External PHY Device	Connect this to the _t port of the Bi-Di I/O primitive on the MDIO to the External PHY device.

Debugging

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

On the RGMII interface, the RX data should provide sufficient setup time with regard to the RX clock so that it can be sampled correctly. The IDELAY components are provided on the RX datapath. The correct delay value must be specified so that the RX data can be correctly sampled.

Finding Help on Xilinx.com

To help in the design and debug process when using the GMII to RGMII core, the Xilinx Support web page (Xilinx Support web page) contains key resources such as product documentation, release notes, answer records, information about known issues, and links for obtaining further product support.

Documentation

This product guide is the main document associated with the GMII to RGMII core. This guide, along with documentation related to all products that aid in the design process, can be found on the Xilinx Support web page or by using the Xilinx Documentation Navigator.

Download the Xilinx Documentation Navigator from the Downloads page. For more information about this tool and the features available, open the online help after installation.

Answer Records

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

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

- Product name
- Tool messages
- Summary of the issue encountered

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

Master Answer Record for the GMII to RGMII core

AR: 54689

Technical Support

Xilinx provides technical support in the Xilinx Support web page for this LogiCORE™ IP product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support if you do any of the following:

- Implement the solution in devices that are not defined in the documentation.
- Customize the solution beyond that allowed in the product documentation.
- Change any section of the design labeled DO NOT MODIFY.

To contact Xilinx Technical Support, navigate to the Xilinx Support web page.

Debug Tools

Many tools are available to address GMII to RGMII design issues. It is important to know which tools are useful for debugging various situations.

Vivado Design Suite Debug Feature

The Vivado® Design Suite debug feature inserts logic analyzer and virtual I/O cores directly into your design. The debug feature allows you to set trigger conditions to capture application and integrated block port signals in hardware. Captured signals can then be analyzed. This feature in the Vivado IDE is used for logic debugging and validation of a design running in Xilinx.

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

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

See the *Vivado Design Suite User Guide: Programming and Debugging* (UG908) [Ref 8].

Simulation Debug

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

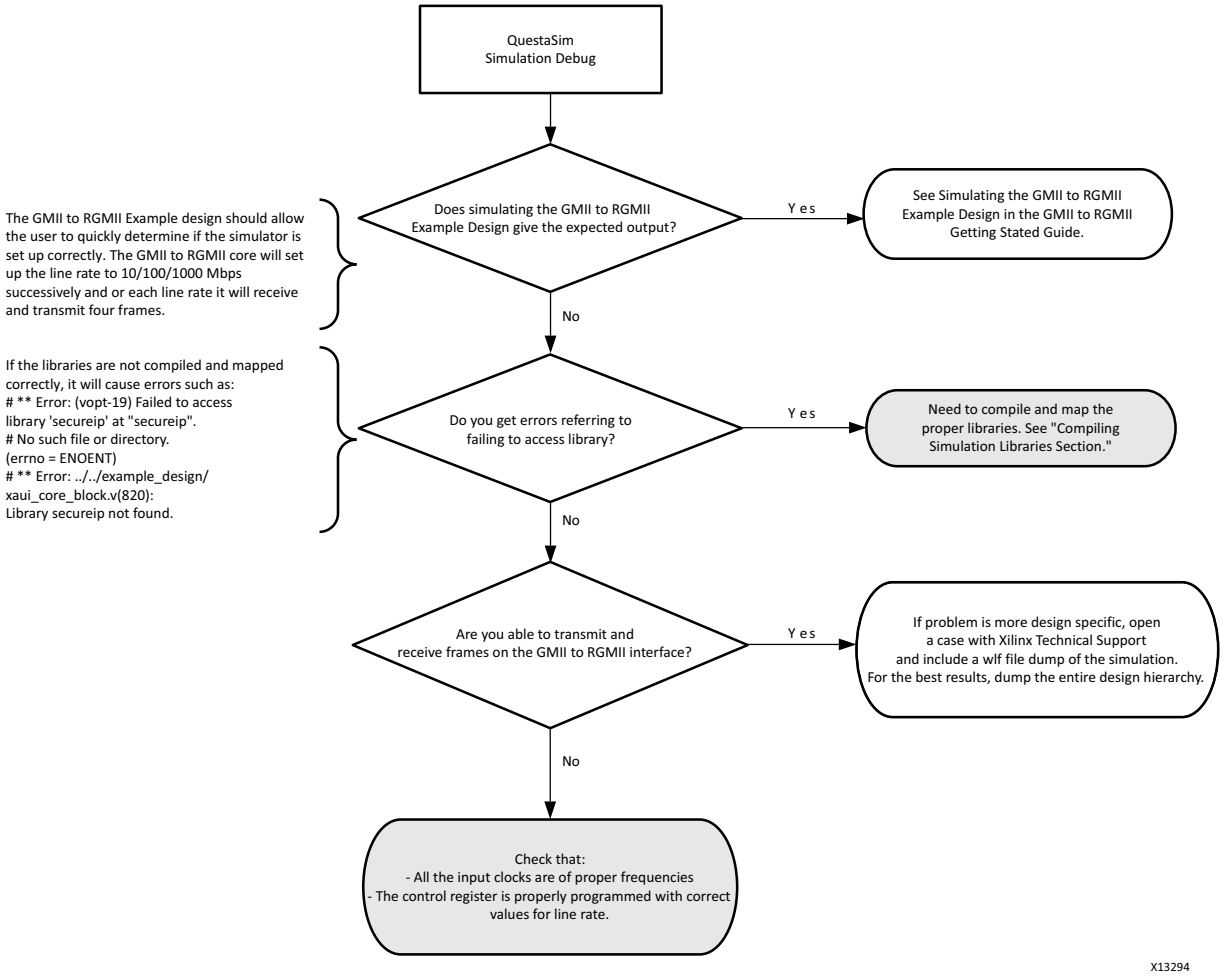

Figure B-1: **Simulation Debug Flow**

Hardware Debug

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

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

- General checks
- Problems with the MDIO
- Problems with data reception or transmission
- Problems with High bit error rates

General Checks

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

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

Problems with the MDIO

- Ensure that the MDIO is driven properly. See MDIO Management System in Chapter 2 for detailed information about performing MDIO transactions.
- Check that the mdc clock is running and that the frequency is 2.5 MHz or less.
- Check that the PHYAD field placed into the MDIO frame matches the attribute value C_PHYADDR of the core.

Problems with Data Reception or Transmission

When no data is being received or transmitted, ensure that a valid link has been established between the external PHY and its link partner, either by auto-negotiation or manual configuration. After the link has been successfully established, ensure that the driver software configures the core register 0x10 with correct values for speed.

Note: By default, the speed is set to 10 Mb/s and the duplex mode to half-duplex.

Problems with High Bit Error Rates

On the RGMII interface, the RX data should provide sufficient set-up time with regard to the RX clock so that it can be sampled correctly. The IDELAY components are provided on the RX datapath. The correct delay value must be specified so that the RX data can be correctly sampled.

Appendix C

Additional Resources and Legal Notices

Xilinx Resources

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

References

These documents provide supplemental material useful with this product guide:

1. *Zynq-7000 All Programmable SoC Technical Reference Manual* (UG585)
2. *IEEE std 802.3-2012* (http://standards.ieee.org/findstds/standard/802.3-2012.html)
3. *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
4. *Reduced Gigabit Media Independent Interface (RGMII) Version* documentation (www.hp.com/rnd/pdfs/RGMIIv2_0_final_hp.pdf)
5. *Vivado Design Suite User Guide: Designing with IP* (UG896)
6. *Vivado Design Suite User Guide: Getting Started* (UG910)
7. *ISE to Vivado Design Suite Migration Guide* (UG911)
8. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)
9. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
10. *Vivado Design Suite User Guide: Implementation* (UG904)
11. *Zynq UltraScale+ MPSoC Technical Reference Manual*(UG1085)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
04/06/2016	4.0	• Added description for mmcm_locked_in and mmcm_locked_out signals in Table 2-2. • Added a note to indicate the exclusion of support for PHY Address 0.
11/18/2015	4.0	• Added support for Zynq® UltraScale+ MP SoCs.
06/24/2015	4.0	• Updated the Required Constraints section in Design Flow Steps Chapter.
04/01/2015	4.0	• Removed XST. • Updated Table 2-1: Zynq-7000 Device Utilization. • Updated and added gmii_tx_clk_90 in Table 2-2: I/O Signals. • Updated Fig. 2-2: Interface of GMII to RGMII Core and Fig. 2-3: GMII to RGMII Block Level. • Updated and added gmii_clk_90, gmii_clk_90_out, gmii_clk_125m_90_out, gmii_clk_25m_90_out, gmii_clk_2_5m_90_out, gmii_clk_125m_90_in, gmii_clk_25m_90_in, mmcm_locked, and gmii_clk_2_5m_90_in in Table 2-3: Block Level I/O Signals. • Updated description in General Design Guidelines section. • Updated Fig. 3-3: When Shared Logic in Core is Selected to Fig. 3-8: RGMII Transmit Clock to the External PHY Device. • Updated Fig. 4-1: Core Customization Screen and Fig. 4-2: Shared Logic Options. • Added User Parameters. • Added gmii_clk_90 in Clock Frequencies. • Added Parameter Changes from v3.0 to v4.0 and Port Changes from v3.0 to v4.0. • Added UNISIM important note in Simulation section. • Updated RGMII_TXC_SKEW and related changes.
10/01/2014	3.0	• Added interfaces for block level of IP and updated interfaces for core level • Added Chapter 6, Test Bench • Added RGMII TXC skew adjustment constraints to Required Constraints
10/02/2013	3.0	• Revision number advanced to 3.0 to align with the core version number. • Updated numbers and text in Resource Utilization section • Replaced Table 2-2 with new table. • Replaced Table 3-1 with new table. • Added Shared Logic section to Chapter 3. • Update Figures 2-1, 2-2, 2-3, 3-3, 4-1, 4-2, 5-1, B-1, and B-2 • Removed PHY Address section from Chapter 4. • Removed Chapter 6, Simulation, Chapter 7, Synthesis and Implementation, Chapter 8, Example Design, and Chapter 9, Test Bench. • Added the section MDIO Management System to Chapter 2. • Added Upgrading in the Vivado Design Suite section to Appendix A.
03/20/2013	1.0	Initial Xilinx release. Based on PB014.

Please Read: Important Legal Notices

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of Xilinx's limited warranty, please refer to Xilinx's Terms of Sale which can be viewed at http://www.xilinx.com/legal.htm#tos; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in such critical applications, please refer to Xilinx's Terms of Sale which can be viewed at http://www.xilinx.com/legal.htm#tos.