

Triple Speed Ethernet

MegaCore Function User Guide



101 Innovation Drive
San Jose, CA 95134
www.altera.com

MegaCore Version: 7.1
Document Date: May 2007

Copyright © 2007 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.



UG-01008-1.3



About This User Guide	ix
Introduction	ix
References	ix
Revision History	x
How to Contact Altera	x
Typographic Conventions	xi
 Chapter 1. About This MegaCore Function	
Release Information	1-1
Device Family Support	1-1
Features	1-2
General Description	1-4
MegaCore Verification	1-8
Hardware Verification	1-8
Optical Platform	1-8
Copper Platform	1-9
Performance	1-9
Installation and Licensing	1-10
OpenCore Plus Evaluation	1-11
OpenCore Plus Time-Out Behavior	1-11
 Chapter 2. Getting Started	
Triple Speed Ethernet Design Flow	2-1
Parameterization Flow Selection	2-2
SOPC Builder Flow	2-2
Specify MegaCore Function Parameters	2-3
Complete the SOPC Builder System	2-3
Simulate the System	2-4
MegaWizard Plug-in Manager Flow	2-5
Specify MegaCore Function Parameters	2-5
Generated Files	2-6
Simulate the MegaCore Function with Provided Testbench	2-7
Simulating with the ModelSim Simulator	2-8
Simulating with Other Simulators	2-8
Instantiate the MegaCore Function in your Design	2-9
Timing Constraints	2-9
Design Compilation and Device Programming	2-10
 Chapter 3. Configuring the MegaCore Function	
Introduction	3-1

Core Configuration	3-2
Core Variation	3-2
Interface	3-3
Use Transceiver Block	3-3
MAC Options	3-4
Ethernet MAC Options	3-4
MDIO Module	3-5
FIFO Options	3-6
Width and Memory Type	3-6
Depth	3-7
PCS/SGMII Options	3-7
PCS/SGMII	3-7

Chapter 4. Specifications

10/100/1000	
Ethernet MAC	4-1
Ethernet Frame Format	4-2
MAC Receive Operation	4-5
Preamble Processing	4-5
Collision Detection in Half-Duplex Mode	4-6
MAC Address Checking	4-6
Frame Length/Type Checking	4-8
Payload Pad Removal	4-10
Frame Length Checking	4-10
CRC Checking	4-11
Receive FIFO Thresholds	4-12
MAC Transmit Operation	4-14
MAC Address Insertion	4-15
Frame Payload Padding	4-15
CRC-32 Generation	4-15
Inter Packet Gap	4-16
Collision Detection in Half-Duplex Mode	4-16
Transmit FIFO Thresholds	4-18
Transmit FIFO Underflow	4-19
Transmit FIFO Overflow	4-20
MAC Receive and Transmit FIFO Interfaces	4-21
MAC Full Duplex Flow Control Operation	4-22
Remote Device Congestion	4-22
Local Device / FIFO Congestion	4-23
Pause Frames	4-24
Magic Packet Detection	4-25
Sleep Mode	4-26
Magic Packet Detection	4-26
Wakeup	4-26
Software Reset	4-27
PHY Management (MDIO)	4-28
MDIO Frame Format	4-29

MDIO Registers	4-30
MDIO Clock Generation	4-30
MDIO Buffer Connection	4-30
MAC Interface Register Map	4-31
Complete MAC Interface Register Map	4-32
Command_Config Register	4-39
Reg_Status Register	4-43
Tx_Cmd_Stat Register	4-43
Rx_Cmd_Stat Register	4-44
MAC SNMP MIB Statistics Registers	4-45
MII, GMII and RGMII Interfaces	4-48
GMII Interface	4-48
RGMII Interface	4-50
MII Interface	4-52
Connecting MAC to External PHY	4-54
Gigabit Ethernet	4-55
Programmable 10/100/1000 Operation	4-55
1000BASE-X/SGMII PCS with Optional PMA	4-58
PCS Receive	4-59
Comma Detection	4-59
8b/10b Decoding	4-60
Frame De-encapsulation	4-60
Synchronization	4-60
Carrier Sense	4-61
PCS Transmit	4-61
Frame Encapsulation	4-61
8b/10b Encoding	4-61
SGMII Converter	4-62
Transmit	4-62
Receive	4-62
Clock Distribution with External PMA	4-62
Clock Distribution with Embedded PMA	4-64
Auto-Negotiation	4-64
1000BASE-X Auto-Negotiation	4-65
SGMII Auto-Negotiation	4-67
TBI Interface	4-67
PHY Loopback	4-68
PHY Power-Down	4-68
Power-Down with Embedded PMA	4-69
PCS Control Interface Register Map	4-70
PCS Control Register	4-72
Status Register	4-72
Dev_Ability and Partner_Ability Registers	4-74
An_Expansion Register	4-75
If_Mode Register	4-76
Signals	4-77
10/100/1000 Ethernet MAC Signals	4-77

Control Interface Signals	4-78
MAC System-Side Signals	4-79
MAC Ethernet-Side Signals	4-85
10/100/1000 Ethernet MAC with 1000BASE-X/SGMII PCS Signals	4-89
Control Interface Signals	4-90
MAC System-Side Signals	4-90
PCS Ethernet-Side Signals	4-90
10/100/1000 Ethernet MAC with 1000BASE-X/SGMII PCS and PMA Signals	4-92
Control Interface Signals	4-93
MAC System-Side Signals	4-93
PCS Ethernet-Side Signals	4-93
1000BASE-X/SGMII PCS Signals	4-95
Control Interface Signals	4-96
PCS MAC-Side Signals	4-97
PCS Ethernet-Side Signals	4-99
1000BASE-X/SGMII PCS and PMA Signals	4-100
Control Interface Signals	4-100
PCS MAC-Side Signals	4-101
PCS Ethernet-Side Signals	4-101

Chapter 5. Testbench

Introduction	5-1
Testbench Specifications	5-1
Avalon-ST Ethernet Frame Generator	5-1
Avalon-ST Ethernet Frame Monitor	5-1
MII/RGMII/GMII Ethernet Frame Generator	5-2
MII/RGMII/GMII Ethernet Frame Monitor	5-2
MDIO Slave	5-2
Configuration	5-2
Verification	5-3
Testbench Architecture	5-4
10/100/1000 Ethernet MAC	5-4
Overview	5-4
Default Testbench Configuration	5-5
Test Flow	5-5
1000BASE-X/SGMII PCS	5-7
Overview	5-7
Default Testbench Configuration	5-7
Test Flow	5-8
1000BASE-X/SGMII PCS with embedded PMA	5-9
Overview	5-9
Default Testbench Configuration	5-10
Test Flow	5-10
10/100/1000 Ethernet MAC with 1000BASE-X/SGMII PCS	5-11
Overview	5-11
Default Testbench Configuration	5-12
Test Flow	5-12

10/100/1000 Ethernet MAC with 1000BASE-X/SGMII PCS and Embedded PMA	5-14
Overview	5-14
Default Testbench Configuration	5-15
Test Flow	5-15

Chapter 6. Software Programming Interface

Triple Speed Ethernet Driver Architecture	6-1
Directory Structure	6-3
API Functions	6-4
triple_speed_ethernet_init()	6-4
tse_mac_close()	6-5
tse_mac_raw_send()	6-6
tse_mac_setGMII mode()	6-7
tse_mac_setMII mode()	6-8
tse_mac_SwReset()	6-9
Constants	6-10

Appendix A. Simulation Parameters

Functionality Configuration Parameters	A-1
Test Configuration Parameters	A-4



About This User Guide

Introduction

This user guide comprises the following chapters:

- [Chapter 1, About This MegaCore Function](#) introduces the Triple Speed Ethernet MegaCore® Function and gives a high-level description of the features.
- [Chapter 2, Getting Started](#) describes the design flow for creating a custom variation of the MegaCore function.
- [Chapter 3, Configuring the MegaCore Function](#) shows the parameterization wizard pages and describes the parameters.
- [Chapter 4, Specifications](#) provides detailed reference on the features of the MegaCore function.
- [Chapter 5, Testbench](#) describes the testbench provided with the MegaCore function.
- [Chapter 6, Software Programming Interface](#) describes the software architecture and programming interface provided.
- [Appendix A, Simulation Parameters](#) provides reference on parameters for simulating the MegaCore function.

References

This user guide references the following specifications and articles:

- IEEE 802.3 2002 Edition
- IEEE 802.1Q 1998 Edition
- RFC2665, Definitions of Managed Objects for the Ethernet-like Interface Type, www.ietf.org
- RFC2863, The Interfaces Group MIB, www.ietf.org
- AMD, Magic Packet Technology White Paper, Publication 20213 Rev. A November 1995
- Serial-GMII Specification Revision 1.7, Cisco Specification
- Reduced Gigabit Media Independent Interface (RGMII), 4/1/2002 Version 2.0

Revision History

The table below displays the revision history for the chapters in this user guide.

Chapter	Date	Version	Changes Made
All	May 2007	1.3	<ul style="list-style-type: none"> Added chapters 2, 3, 5 and 6. Contents updated to reflect changes and enhancements in the current version.
1	March 2007	1.2	Updated signal names and description.
2	March 2007	1.2	
3	March 2007	1.2	
Appendix A	March 2007	1.2	<ul style="list-style-type: none"> No change from previous release
1	December 2006	1.1	<ul style="list-style-type: none"> Global terminology changes: 1000BASE-X PCS/SGMII to 1000BASE-X/SGMII PCS, host side or client side to internal system side, HD to half-duplex
2	December 2006	1.1	
3	December 2006	1.1	
Appendix A	December 2006	1.1	<ul style="list-style-type: none"> Initial release of document on Web.
1	December 2006	1.0	<ul style="list-style-type: none"> Initial release of document on DVD.
2	December 2006	1.0	<ul style="list-style-type: none"> Initial release of document on DVD.
3	December 2006	1.0	<ul style="list-style-type: none"> Initial release of document on DVD.
Appendix A	December 2006	1.0	<ul style="list-style-type: none"> Initial release of document on DVD.

How to Contact Altera

For the most up-to-date information about Altera® products, refer to the following table.








Information Type	Contact (1)
Technical support	www.altera.com/mysupport/
Technical training	www.altera.com/training/custrain@altera.com
Product literature	www.altera.com/literature/
Altera literature services	literature@altera.com
FTP site	ftp.altera.com

Note to table:

(1) You can also contact your local Altera sales office or sales representative.

Typographic Conventions

This document uses the typographic conventions shown below.

Visual Cue	Meaning
Bold Type with Initial Capital Letters	Command names, dialog box titles, checkbox options, and dialog box options are shown in bold, initial capital letters. Example: Save As dialog box.
bold type	External timing parameters, directory names, project names, disk drive names, filenames, filename extensions, and software utility names are shown in bold type. Examples: f_{MAX} , lqdesigns directory, d: drive, chiptrip.gdf file.
<i>Italic Type with Initial Capital Letters</i>	Document titles are shown in italic type with initial capital letters. Example: <i>AN 75: High-Speed Board Design</i> .
<i>Italic type</i>	Internal timing parameters and variables are shown in italic type. Examples: <i>t_{PIA}</i> , <i>n + 1</i> . Variable names are enclosed in angle brackets (< >) and shown in italic type. Example: <file name>, <project name>.pdf file.
Initial Capital Letters	Keyboard keys and menu names are shown with initial capital letters. Examples: Delete key, the Options menu.
"Subheading Title"	References to sections within a document and titles of on-line help topics are shown in quotation marks. Example: "Typographic Conventions."
Courier type	Signal and port names are shown in lowercase Courier type. Examples: data1, tdi, input. Active-low signals are denoted by suffix n, e.g., resetn. Anything that must be typed exactly as it appears is shown in Courier type. For example: c:\qdesigns\tutorial\chiptrip.gdf. Also, sections of an actual file, such as a Report File, references to parts of files (e.g., the AHDL keyword SUBDESIGN), as well as logic function names (e.g., TRI) are shown in Courier.
1., 2., 3., and a., b., c., etc.	Numbered steps are used in a list of items when the sequence of the items is important, such as the steps listed in a procedure.
	Bullets are used in a list of items when the sequence of the items is not important.
	The checkmark indicates a procedure that consists of one step only.
	The hand points to information that requires special attention.
	A caution calls attention to a condition or possible situation that can damage or destroy the product or the user's work.
	A warning calls attention to a condition or possible situation that can cause injury to the user.
	The angled arrow indicates you should press the Enter key.
	The feet direct you to more information on a particular topic.



1. About This MegaCore Function

Release Information

Table 1–1 provides information about this release of the Altera® Triple Speed Ethernet MegaCore Function.

Table 1–1. Triple Speed Ethernet MegaCore Function Release Information	
Item	Description
Version	7.1
Release Date	May 2007
Ordering Code	IP-TRIETHERNET
Product ID(s)	00BD
Vendor ID(s)	6AF7

Device Family Support

MegaCore® functions provide either full or preliminary support for target Altera device families:

- *Full support* means the MegaCore function meets all functional and timing requirements for the device family and may be used in production designs
- *Preliminary support* means the MegaCore function meets all functional requirements, but may still be undergoing timing analysis for the device family; it may be used in production designs with caution.

Table 1–2 shows the level of support offered by the Triple Speed Ethernet MegaCore Function to each Altera device family.

Table 1–2. Device Family Support (Part 1 of 2)	
Device Family	Support
Stratix® III	Preliminary
Stratix II	Full
Stratix II GX	Preliminary
Arria™ GX	Preliminary
Cyclone® III	Preliminary
Cyclone II	Full

Table 1–2. Device Family Support (Part 2 of 2)

Device Family	Support
HardCopy® II	Preliminary
Other device families	No support

Features

The Triple Speed Ethernet MegaCore function combines the features of a 10/100/1000 Ethernet media access controller (MAC) and a 1000BASE-X physical coding sub-layer (PCS). The following list provides a high-level overview of the features of the MAC and PCS functions.

- IEEE 802.3 Standard compliant and tested by the University of New Hampshire (UNH) InterOperability Lab
- Easy-to-use MegaWizard® interface
- IP functional simulation models for use in Altera-supported VHDL and Verilog HDL simulators
- Support for OpenCore Plus evaluation
- IEEE 802.3 Compliant 10/100/1000 Mbps Ethernet MAC providing the following features:
 - Preamble and SFD generation
 - Frame padding generation
 - CRC generation on transmit and CRC checking on receive
 - Dynamically configurable support for 10 Mbps, 100 Mbps, or 1Gbps operation
 - AMD magic packet detection for remote power management
 - Full duplex or half duplex operation in 10/100 Mbps mode
 - Flexible, standard interfaces:
 - Seamless interface to commercial gigabit Ethernet PHY devices via a 8-bit Gigabit Media Independent Interface (GMII) operating at 125 MHz, or a 4-bit Reduced Gigabit Media Independent Interface (RGMII) operating at 250 MHz
 - Seamless interface to commercial fast Ethernet PHY device via a 4-bit Media Independent Interface (MII) operating at 25 MHz and interface to 10 Mbps Ethernet PHY device using MII operating at 2.5 MHz
 - Simple 8-bit or 32-bit FIFO interface to application logic based on the Avalon Streaming (Avalon-ST) interface



For Information About	Refer To
Avalon-ST interface protocol	<i>Avalon Streaming Interface Specification</i>

- CRC-32 generation and append on transmit, or forwarding of application-provided frame checksum, selectable on a per-frame basis
- CRC-32 checking at full speed with optional forwarding of the frame checksum field to the application
- Dynamically programmable pause quanta used to form XOFF pause frames when operating in full-duplex mode
- Pause frame generation controllable by the application, enabling flexible traffic flow control
- Optional forwarding of received pause frames to the application when operating in full-duplex mode
- Standard flow-control mechanism in full-duplex mode
- Full collision support, including jamming, backoff, and automatic retransmission, when operating in half-duplex mode
- Support for VLAN and stacked VLAN tagged frames according to IEEE 802.1Q
- Programmable MAC address. The MegaCore function automatically inserts the MAC address on transmit, and discards frames with mismatching destination MAC address on receive (except broadcast frames)
- Programmable group of four supplemental destination MAC addresses which can be used to accept or reject unicast traffic
- Programmable promiscuous mode support to omit destination MAC address checking on receive
- Multicast address detection and filtering on receive based on a 64-entry hash table, reducing higher-layer processing load
- Programmable maximum frame-length, providing support for any standard or proprietary frame length up to 64 KBytes, such as 9 KByte jumbo frames
- Optional internal loopback on GMII and MII interfaces
- Statistics indicators for frame traffic as well as errors (alignment, CRC, length) and pause frames, enabling Simple Network Management Protocol (SNMP) management environments. The MegaCore function provides optional support for IEEE 802.3 basic and mandatory Management Information Database (MIB) package, as well as Ethernet MIB (RFC 2665) and Remote Network Monitoring (RFC 2819)
- Configurable FIFO size (64 Bytes to 256 KBytes) and programmable threshold levels ensuring data rates of 1 Gbps with back-to-back frame transfer support
- Automatic discard of received frames with error
- Separate status word for each received frame providing information such as frame length, frame type, VLAN tag, and error information
- MDIO master interface for PHY device configuration and management, with two programmable MDIO base addresses

- Optional alignment of packet headers to 32-bit boundaries independently programmable on transmit and receive paths
- Optional Reduced Gigabit Medium-Independent Interface (RGMI)
- Management interface (MDIO-MDC signals) for managing external PHY devices
- 1000BASE-X/SGMII PCS (Serial GMII) providing the following features:
 - Support for Clause 36 of IEEE 802.3-2000 specification for 1000BASE-X family of PCS
 - PCS frame encapsulation and de-encapsulation with /S/, /T/ ordered set insertion and termination, and /I/ Idle ordered set generation during inter-packet gap (IPG)
 - Receive link synchronization state machine and 10-bit data alignment from SERDES (SERializer/DESerializer) chip with K28.5 character detection
 - Link coding implemented with 8B/10B code, providing a DC-balanced bitstream for efficient SERDES operation
 - Support for 1000BASE-X Auto-Negotiation (IEEE 802.3 Clause 37) with fully programmable node ability register
 - Optional SGMII bridge logic to support 10 Mbps, 100 Mbps or 1000 Mbps and SGMII auto-negotiation and support for SGMII auto-negotiation
 - Programmable link timer, used during standard or SGMII auto-negotiation
 - Standard management register set as defined in Clause 22 of IEEE 802.3 with extended registers providing improved flexibility
 - PHY isolation support to allow the implementation of a hot-swappable PHY device
 - Encapsulation and auto-negotiation functions that can optionally be by-passed
- Optional integrated Physical Medium Attachment (PMA)—On-Chip serialization and deserialization of GbE and SGMII data streams in Stratix II GX and Arria GX Devices using integrated GX transceivers

General Description

The Triple Speed Ethernet MegaCore function provides an integrated Ethernet MAC and PCS solution for ethernet applications, such as line cards, NIC cards, and switches, operating at 10/100 Mbps (Fast Ethernet) or 1000 Mbps (Gigabit Ethernet). In full-duplex mode, the MAC function supports both switching and NIC or line-card applications, by providing transparent and full Ethernet frame termination and generation. For efficient power management, the MegaCore function also implements magic packet detection.

Figure 1–1 shows a block diagram of the Triple Speed Ethernet MegaCore function. The function comprises the following main blocks: MAC, PCS, and PMA. All blocks are optional and configurable at synthesis time. A memory-mapped register interface controls the MAC and PCS blocks. External I/O signals provide additional control and status for the MegaCore function.

Figure 1–1. Triple Speed Ethernet MegaCore Function Block Diagram

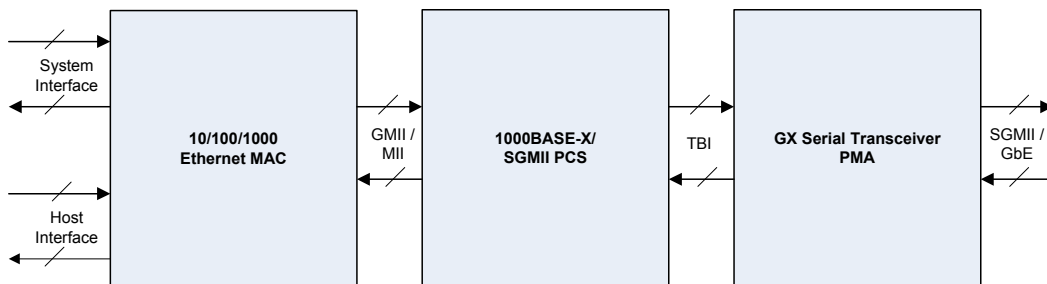


Table 1–3 shows the possible configurations for the Triple Speed Ethernet MegaCore function. Depending on the configuration, the interfaces on the system side and the Ethernet side change.

Table 1–3. Configuration Options for the MAC, PCS, and Embedded GX Transceivers				
MegaCore Configuration			Interfaces	
10/100/1000 Ethernet MAC	1000BASE-X /SGMII PCS	Include GX Transceivers	System Side	Ethernet Side
x			Avalon-ST	GMII/RGMII/MII and optional MDIO
x	x		Avalon-ST	TBI and optional MDIO
x	x	x	Avalon-ST	1.25 Gbps SGMII MDI (GbE/SGMII)
	x	x	GMII and MII	1.25 Gbps MDI (GbE/SGMII)
	x		GMII and MII	TBI

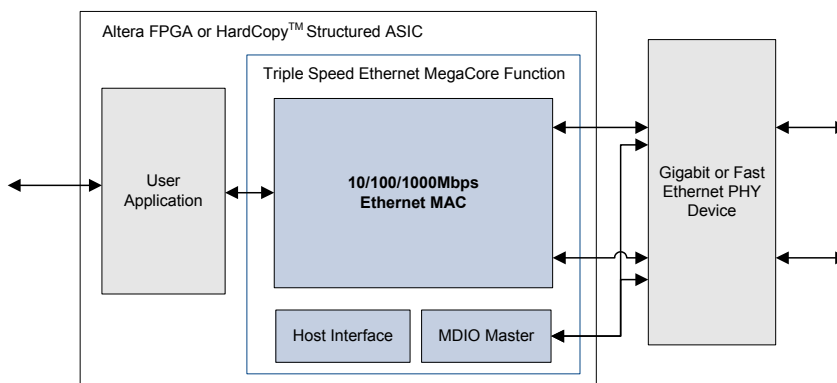
On the system side, the MAC function provides standard Avalon-ST interfaces, providing interoperability with other Avalon Streaming-based IP. A host CPU can manage the MAC function via a separate Avalon-MM interface that provides access to the control register space. On the

Ethernet side, the MAC can seamlessly connect to any industry standard gigabit Ethernet PHY device via GMII or RGMII or to a 10/100 Ethernet PHY device via MII.

The 1000BASE-X PCS function is compliant with Clause 36 of the IEEE802.3 standard and implements 8B/10B coding, link synchronization, and frame encapsulation generation and termination. The PCS function also supports auto-negotiation as defined in Clause 37 of the IEEE802.3 standard, which is used to exchange ability information between the PCS function and the remote link partner. Auto-negotiation allows the PCS function to take the best advantage of the advertised features of the remote node, either automatically or under software control.

Figure 1–2 illustrates an example application using the Triple Speed Ethernet MegaCore function as a stand-alone IP block, serving as a bridge between the user application and standard 10/100 and gigabit Ethernet PHY chips. This example application does not include the PCS function.

Figure 1–2. Example Application



When configured to include the 1000BASE-X/SGMII PCS function, the MegaCore function can seamlessly connect to any industry standard gigabit Ethernet PHY device via a gigabit Ten-bit Interface(TBI). Alternatively, when configured to include both the 1000BASE-X/SGMII PCS and GX transceiver blocks, the MegaCore function can be connected directly to a Gigabit Interface Converter (GBIC), SFP optical module or an SGMII PHY.

Figure 1–3 illustrates an example application using the Triple Speed Ethernet MegaCore function with PCS operating in 1000BASE-X mode and GX transceiver block as a PMA. The transceiver connects to an off-the-shelf GBIC or small form-factor pluggable module (SFP) optical module to communicate directly over the optical link.

Figure 1–3. 1000BASE-X/SGMII PCS Core and GX Transceiver PMA (1000BASE-X Mode)

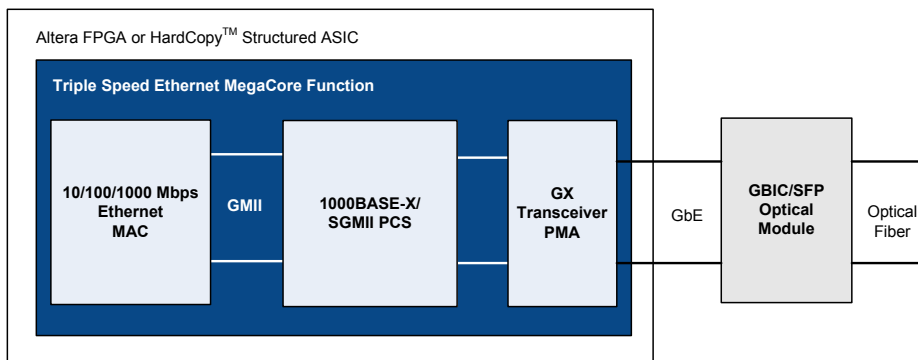
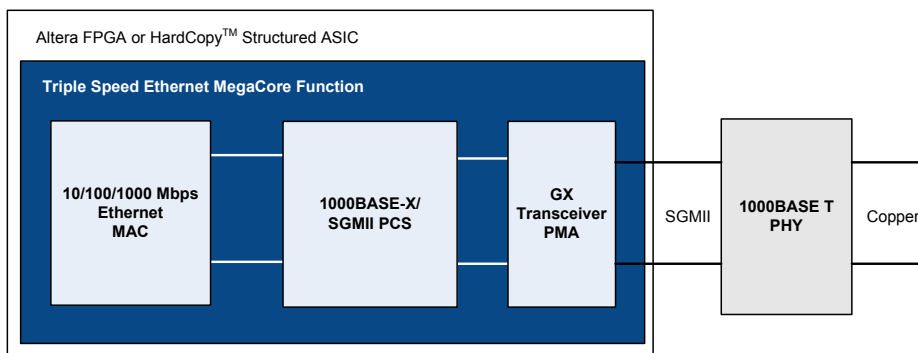


Figure 1–4 shows a similar configuration in which the PCS function is configured to operate in SGMII mode and acts as a GMII-to-SGMII bridge. In this case, the transceiver I/O connects to an off-the-shelf Ethernet PHY that supports SGMII (10BASE-T, 100BASE-T or 1000BASE-T Ethernet PHY.)

Figure 1–4. 1000BASE-X/SGMII PCS MegaCore with GX Transceiver PMA (GMII to SGMII Bridge Mode)





Throughout this document, the terms *GX transceiver* and *SERDES* are used interchangeably to refer to FPGA device features that enable the transmission and reception of high-speed serial data streams.



For Information About	Refer To
Stratix II GX transceivers	<i>Stratix II GX Handbook</i>

MegaCore Verification

Altera verified the Triple Speed Ethernet MegaCore function through extensive in-house simulation and internal hardware verification. The University of New Hampshire (UNH) InterOperability Lab also successfully verified the MegaCore function prior to its release.

Altera used a highly parameterizeable transaction-based test bench to test the following aspects of the MegaCore function:

- Register access
- MDIO access
- Frame transmission and error handling
- Frame reception and error handling
- Ethernet frame MAC address filtering
- Flow control
- Retransmission in half-duplex

Hardware Verification

Altera has validated the Triple Speed Ethernet MegaCore function in both optical and copper platforms using the following development kits:

- Altera Nios II Development Kit, Cyclone II Edition (2C35)
- Altera Nios II Development Kit, Stratix II Edition (2S60)
- Altera PCI Express Development Kit, Stratix II GX Edition
- MorethanIP 10/100 and 10/100/1000 Ethernet PHY Daughtercards

Optical Platform

In the optical platform, both the 10/100/1000 Mbps Ethernet MAC and 1000BASE-X/SGMII PCS functions are instantiated.

The FPGA application implements the Ethernet MAC, the 1000BASE-X PCS and an internal system using Ethernet connectivity. This internal system retrieves all frames received by the MAC function and returns them to the sender manipulating the address fields, thus implementing a loopback behavior. A direct connection to an optical module is provided through an external SFP module.

Copper Platform

In the copper platform, Altera tested the Triple Speed Ethernet MegaCore function with an external 1000BASE-T PHY devices. The MegaCore function is connected to the external PHY device using GMII, RGMII and SGMII, in conjunction with the 1000BASE-X/SGMII PCS function.

A 10/100/1000 Mbps Ethernet MAC and an internal system are implemented in the FPGA. The internal system retrieves all frames received by the MAC function and returns them to the sender by manipulating the address fields, implementing a loopback. A direct connection to an Ethernet link is provided through a combined MII to an external PHY module.

Performance

Table 1–4 provides information on resource utilization and performance of the five configurations of the Triple Speed Ethernet MegaCore function. It shows the typical expected performance and resource utilization for different configuration options, using the Quartus II software targeting a Stratix II GX (EP2SGX60CF780C3) device.

Table 1–4. Triple Speed Ethernet Performance and Utilization (Part 1 of 2)

Configuration / Setting	FIFO Size	Speed Grade Used (1)	Comb. ALUT	Logic Regs	fMax	Memory (M4K / M512)
MAC Half duplex Statistics counters enabled	64x8-bit	C3	2487	2934	>125 MHz	8 M4K 2 M512
	64x32-bit	C3	2479	3039	>125 MHz	8 M4K 2 M512
	2048x8-bit	C3	2723	3244	>125 MHz	1 M512 22 M4K
	2048x32-bit	C3	2720	3273	>125 MHz	1 M512 46 M4K
MAC Full duplex	2048x8-bit	C3	1418	2137	>125 MHz	17 M4K
	2048x32-bit	C3	1400	2145	>125 MHz	41 M4K
MAC Full duplex MII/RGMII	2048x8-bit	C3	1427	2158	>125 MHz	17 M4K
	2048x32-bit	C3	1399	2163	>125 MHz	41 M4K
1000BASE-X/SGMII PCS		C3	906	962	>125 MHz	2 M512
MAC and 1000BASE-X/SGMII PCS Half duplex Statistics counters enabled	2048x32-bit	C3	3506	4133	>125 MHz	3 M512 46 M4K
	2048x8-bit	C3	3514	4085	>125 MHz	3 M512 22 M4K

Table 1–4. Triple Speed Ethernet Performance and Utilization (Part 2 of 2)

Configuration / Setting	FIFO Size	Speed Grade Used (1)	Comb. ALUT	Logic Regs	fMax	Memory (M4K / M512)
MAC and 1000BASE-X/SGMII PCS Full duplex	2048x8-bit	C3	2150	2985	>125 MHz	2 M512 17 M4K
	2048x32-bit	C3	2135	3041	>125 MHz	2 M512 41 M4k
MAC and 1000BASE-X PCS with PMA (2) Half-duplex Statistics counters enabled	2048x8-bit	C3	3513	4090	>125 MHz	3 M512 22 M4K
	2048x32-bit	C3	3504	4113	>125 MHz	3 M512 46 M4K

Note:

- (1) This column indicates the speed grade used to obtain the performance and utilization data. The MegaCore function supports all device speed grades unless otherwise noted.
- (2) Uses 1 GX transceiver.

Installation and Licensing

The Triple Speed Ethernet MegaCore function is included in Altera MegaCore IP Library, which is a part of the Altera Complete Design Suite. To install the MegaCore function, install the MegaCore IP Library.

You can use Altera's free OpenCore Plus evaluation feature to evaluate the MegaCore function in simulation and in hardware before you purchase a license. You need to purchase a license for the MegaCore function only when you are satisfied with its functionality and performance, and you want to take your design to production.

After you purchase a license for the Triple Speed Ethernet MegaCore function, you can request a license file from the Altera website at www.altera.com/licensing and install it on your computer. When you request a license file, Altera emails you a license.dat file. If you do not have internet access, contact your local Altera representative.



For Information About	Refer To
Installation and licensing	<i>Quartus II Installation & Licensing for Windows</i> or <i>Quartus II Installation & Licensing for UNIX and Linux Workstations</i> manual

OpenCore Plus Evaluation

With Altera's free OpenCore Plus evaluation feature, you can perform the following actions:

- Simulate the behavior of a megafunction (Altera MegaCore function or AMPPSM megafunction) within your system
- Verify the functionality of your design, as well as evaluate its size and speed quickly and easily
- Generate time-limited device programming files for designs that include megafunctions
- Program a device and verify your design in hardware.



For Information About	Refer To
OpenCore Plus hardware evaluation	<i>AN 320: OpenCore Plus Evaluation of Megafunctions</i>

OpenCore Plus Time-Out Behavior

OpenCore Plus hardware evaluation supports the following two operation modes:

- *Untethered*—the design runs for a limited time.
- *Tethered*—requires a connection between your board and the host computer. If tethered mode is supported by all megafunctions in a design, the device can operate for a longer time or indefinitely.

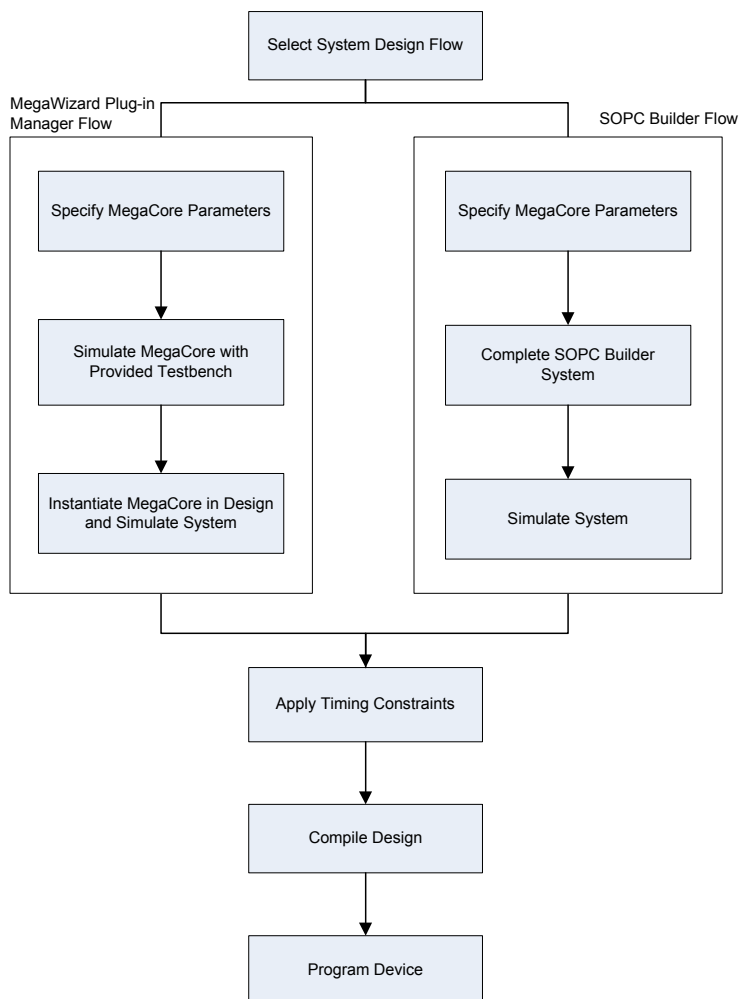
All megafunctions in a device time out simultaneously when the most restrictive evaluation time is reached. If there is more than one megafunction in a design, a specific megafunction's time-out behavior may be masked by the time-out behavior of the other megafunctions.

For MegaCore functions, the untethered timeout is 1 hour; the tethered timeout value is indefinite. Your design stops working after the hardware evaluation time expires.

Triple Speed Ethernet Design Flow

The following figure shows the stages for creating a system with the Triple Speed Ethernet MegaCore function and the Quartus II software. Each of the stages is described in detail in subsequent sections.

Figure 2–1. Triple Speed Ethernet Design Flow



Parameterization Flow Selection

You can parameterize the Triple Speed Ethernet MegaCore function using either one of the following flows:

- SOPC Builder flow
- MegaWizard® Plug-in Manager flow

Table 2–1 summarizes the advantages offered by the different parameterization flows.

Table 2–1. Parameterization Flow Selection Criteria	
SOPC Builder Flow	MegaWizard Plug-in Manager Flow
<ul style="list-style-type: none">• You want to rapidly create a new SOPC Builder system design that includes an Ethernet interface.• You wish to use the Triple Speed Ethernet MegaCore Function in conjunction with other components available in SOPC Builder such as the Nios II processor, External Memory Controllers and the Scatter/Gather DMA Controller.• You wish to make use of the accompanying Nios II/Interniche TCP/IP Protocol Stack software driver support in your system.	<ul style="list-style-type: none">• You would like to parameterize the Triple Speed Ethernet MegaCore function, to create a variant that you can instantiate manually in your design.• You would like to use the 1000BASE-X\SGMII PCS function standalone, without any MAC, in your design.

SOPC Builder Flow

The SOPC Builder flow allows you to add the Triple Speed Ethernet MegaCore function directly to a new or existing SOPC Builder system. You can also easily add other available components to quickly create an SOPC Builder system with an Ethernet interface, such as the Nios II processor, External Memory Controllers and Scatter/Gather DMA Controllers. SOPC Builder automatically creates the system interconnect logic and system simulation environment.



For Information About	Refer To
SOPC Builder	Volume 4 of the <i>Quartus II Handbook</i>
Quartus II software	Quartus II Help

Specify MegaCore Function Parameters

Follow the steps below to specify Triple Speed Ethernet parameters using the SOPC Builder flow.

1. Create a new Quartus II project using the **New Project Wizard** available from the File menu.
2. Launch **SOPC Builder** from the Tools menu.
3. For a new system, specify the system name and language.
4. Add **Triple Speed Ethernet** to your system from the **System Contents** tab.



You can find **Triple Speed Ethernet** by expanding **Interface Protocols > Ethernet**.

5. Specify the required parameters on all pages in the **Parameter Settings** tab.

For detailed explanation of the parameters, refer to the [“Configuring the MegaCore Function” on page 3–1](#).

6. Click **Finish** to complete the Triple Speed Ethernet MegaCore function and add it to the system.

Complete the SOPC Builder System

Follow the steps below to complete the SOPC Builder system.

1. Add and parameterize any additional components to the system.



A typical SOPC builder system that enables Ethernet connectivity utilizes a Scatter/Gather DMA controller on each of the transmit and receive paths, and a Nios II processor for configuration and control.



For a complete example of an SOPC Builder system containing the Triple-Speed Ethernet MegaCore function, refer to the Triple-Speed Ethernet/Scatter-Gather DMA controller example design in the Nios II Embedded Design Suite (EDS).

2. Connect the components using the SOPC Builder patch panel.

3. If you intend to simulate your SOPC builder system, select **Simulate** on the **System Generation** tab to generate a functional simulation model for the system.
4. Click **Generate** to generate the system.

Simulate the System

During system generation, SOPC Builder optionally generates a simulation model and testbench for the entire system which you can use to easily simulate your system in any of Altera's supported simulation tools. SOPC Builder also generates a set of Modelsim Tcl scripts and macros that you can use to compile the testbench, IP Functional simulation models and plain-text RTL design files that describe your system in the Modelsim simulation software.



For Information About	Refer To
Simulating SOPC Builder systems	Volume 4 of the <i>Quartus II Handbook</i>
	AN 351 :Simulating Nios II Systems

When a Triple-Speed Ethernet MegaCore function is present in your system, SOPC Builder also instantiates a loopback module and connects it to your system simulation model. The loopback module connects the Ethernet-side transmit interface to the receive interface via a FIFO. It also provides the Ethernet-side clocks to the simulation model.

MegaWizard Plug-in Manager Flow



The MegaWizard Plug-in Manager flow allows you to customize the Triple Speed Ethernet MegaCore function, and manually integrate the function into your design.

For Information About	Refer To
MegaWizard Plug-in Manager	Quartus II Help
Quartus II software	

Specify MegaCore Function Parameters

Follow the steps below to specify Triple Speed Ethernet parameters using the MegaWizard Plug-in Manager flow.

1. Create a Quartus II project using the **New Project Wizard** available from the File menu.
2. Launch **MegaWizard Plug-in Manager** from the Tools menu, and follow the prompts in the MegaWizard Plug-in Manager interface to create a custom megafunction variation.



You can find **Triple Speed Ethernet v7.1** by expanding **Installed Plug-Ins > Interfaces > Ethernet**.

3. Specify the parameters on all pages in the **Parameter Settings** tab.

For detailed explanation of the parameters, refer to the [“Configuring the MegaCore Function” on page 3–1](#).

4. On the **Simulation Model** tab, select **Enable Simulation Model** and **Language** to generate an IP functional simulation model for the MegaCore function in the selected language.

An IP functional simulation model is a cycle-accurate VHDL or Verilog HDL model produced by the Quartus II software.



Use the simulation models only for simulation and not for synthesis or any other purposes. Using these models for synthesis creates a non-functional design.

- On the **Summary** tab, select the files you want to generate. A grey checkmark indicates a file that is automatically generated. All other files are optional.

For more information about the files generated to your project directory, refer to “[Generated Files](#)” on page 2–6.

- Click **Finish** to generate the MegaCore function and supporting files.

Generated Files

[Table 2–2](#) lists the files generated in your project directory. The names and types of files vary depending on the variation name and HDL type you specify during parameterization.

Table 2–2. Generated Files (1) (Part 1 of 2)	
File Name	Description
<variation_name>.v or <variation_name>.vhd	A MegaCore function variation file, which defines a VHDL or Verilog HDL top-level description of the custom MegaCore function. Instantiate the entity defined by this file inside your design. Include this file when compiling your design in the Quartus II software.
<variation name>_bb.v	Verilog HDL black-box file for the MegaCore function variation. Use this file when using a third-party EDA tool to synthesize your design.
<variation name>.bsf	Quartus II symbol file for the MegaCore function variation. You can use this file in the Quartus II block diagram editor.
<variation name>.cmp	A VHDL component declaration file for the MegaCore function variation. Add the contents of this file to any VHDL architecture that instantiates the MegaCore.
<variation name>_with_pma.vhd or <variation name>_with_pma.v (2)	A MegaCore function variation file, which defines a VHDL or Verilog HDL top-level description of a MegaCore function instantiating a GX serial transceiver as a physical medium attachment (PMA).
<variation name>.html	MegaCore function report file.
<variation name>.vho or <variation name>.vo	VHDL or Verilog HDL IP functional simulation model.
<variation_name>_constraints.tcl	A Tcl script that creates necessary constraints for the Quartus II compilation of your MegaCore Function variation.
<variation_name>_constraints.sdc	Quartus II SDC constraint file for use with TimeQuest timing analyzer

Table 2–2. Generated Files (1) (Part 2 of 2)

File Name	Description
<code><variation_name>_nativelink.tcl</code>	A Tcl script that assigns NativeLink simulation testbench settings to the Quartus II project
<code>/testbench/<variation_name>/<variation_name>_tb.vhd</code> or <code>/testbench/<variation_name>/<variation_name>_tb.v</code>	VHDL or Verilog testbench that exercises your MegaCore function variation in a third party simulator
<code>/testbench/<variation_name>/run_<variation_name>_tb.tcl</code>	A Tcl script for use with the ModelSim simulation software.
<code>/testbench/<variation_name>/<variation_name>_wave.do</code>	A signal tracing macro script used with the ModelSim simulation software to display testbench signals.
<code>/testbench/model</code>	A directory containing VHDL and Verilog HDL models of the Ethernet generators and monitors used by the generated testbench

Notes:

- (1) The files generated are dependent on the custom variation of the MegaCore function you created. Some files might be absent or their names might be different.
- (2) This file is only created when targeting the Stratix II GX device family and the option to use the serial transceiver is selected.

Simulate the MegaCore Function with Provided Testbench

You can simulate the MegaCore function using the IP functional simulation model and testbench generated by the Triple Speed Ethernet MegaWizard. The model and testbench files are generated in the **testbench** sub-directory of the project directory.

You can use any supported simulator for this purpose. The Triple Speed Ethernet MegaWizard generates a script for the Modelsim simulator and a NativeLink script, which can be used by the Quartus II software to generate scripts for all other supported simulators.

For more information on the files generated, refer to [“Generated Files” on page 2–6](#).

For more information on the testbench architecture, refer to the *Testbench* chapter.



For Information About	Refer To
IP functional simulation models	Simulating Altera IP in Third-Party Simulation Tools chapter in volume 3 of the <i>Quartus II Handbook</i>

Simulating with the ModelSim Simulator

Follow the steps below to run a simulation using the ModelSim simulator.

1. Start the ModelSim simulator.
2. Change the working directory to `<project directory>/testbench/<variation name>`.
3. Run the following command to set up the required libraries, compile the generated IP Functional simulation model, and exercise the simulation model with the provided testbench:

```
do run_<variation_name>_tb.tcl
```

The ModelSim transcript pane (in Main window) displays messages from the testbench reflecting the current task being performed.

Simulating with Other Simulators

Follow the steps below to use the Quartus II NativeLink feature to run simulation in other Altera-supported 3rd party simulators.

1. On the **EDA Tool Option** page in the Quartus II software (**Tools > Options > EDA Tools Option**), set the location of your preferred EDA simulation tool executable.



This is a global setting, and needs to be done only once.

2. Type the following command at the Quartus II Tcl console:

```
source <variation name>_nativelink.tcl
```

3. On the **Simulation** page (**Assignments > EDA Tools Settings > Simulation**), make the following selection:
 - From the **Tool name** list, select your preferred simulator.
 - Under **NativeLink settings**, select **Compile test bench**.

- 4. Perform analysis and synthesis to create the required netlist.
- 5. Run the simulation.



For Information About	Refer To
Simulating using NativeLink	Simulating Altera IP in Third-Party Simulation Tools chapter in volume 3 of the <i>Quartus II Handbook</i>

Instantiate the MegaCore Function in your Design

You can now integrate your Triple Speed Ethernet MegaCore function variation into your design, and simulate the system with your custom testbench.

Timing Constraints

Altera provides constraint files to ensure that the Triple Speed Ethernet MegaCore function meets IEEE 802.3 specification and design timing requirements in Altera devices.

Follow the steps below to use the generated constraint files:

- 1. Open your Quartus II project in the Quartus II software.
- 2. Ensure your preferred timing analyzer is selected (Timing Analysis Settings in Assignments menu).
- 3. Source the generated constraint file by typing the following command at the Tcl console (**View > Utility Windows > Tcl Console**) command prompt:

```
source <variation_name>_constraints.tcl
```

This command adds the necessary logic constraints to your Quartus II project. It also creates the timing constraints required for use with the Quartus II Classic timing analyzer.

- 4. If you select TimeQuest timing analyzer as the default timing analysis tool, type the following command at the Tcl console:

```
set_global_assignment -name SDC_FILE  
<variation_name>_constraints.sdc
```



For Information About	Refer To
Timing Analyzers	Timing Analysis section in volume 3 of the <i>Quartus II Handbook</i>
	Quartus II Help

Design Compilation and Device Programming

You can use the Quartus II software to compile your design. After a successful compilation, you can program the targeted Altera device and verify the design in hardware.



For Information About	Refer To
Compiling a design and programming Altera devices	Quartus II Help

Introduction

You customize the Triple Speed Ethernet MegaCore function by specifying parameters using the Triple Speed Ethernet MegaWizard interface, launched from either the MegaWizard Plug-in Manager or SOPC Builder in the Quartus II software.

This chapter describes the parameters and how they affect the behavior of the MegaCore function. Each section corresponds to a page in the **Parameter Settings** tab in the Triple Speed Ethernet MegaWizard interface.



When parameterizing a MegaCore function using the SOPC Builder flow, **Simulation Model** and **Summary** tabs are not visible. In the SOPC Builder flow, simulation model settings are inherited from options specified in the SOPC Builder GUI.

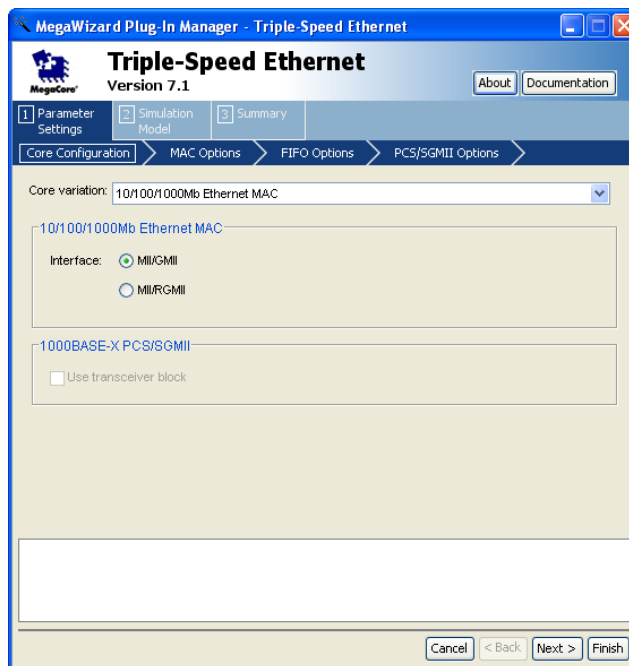


For Information About	Refer To
Setting simulation options	Quartus II Help

Core Configuration

The options on the **Core Configuration** page allow you to specify the primary Ethernet functional block, the interface for MAC block if no PCS support is selected, and to enable the GX transceiver block, if applicable.

Figure 3–1. Core Configuration



Core Variation

This setting determines which of the primary blocks to include in the variation. The following options are available:

- 10/100/1000 Mbps Ethernet MAC only
- 10/100/1000 Mbps Ethernet MAC with 1000BASE-X/SGMII PCS
- 1000BASE-X/SGMII PCS only

When instantiating the Triple Speed Ethernet MegaCore function using the SOPC Builder flow, **1000BASE-X PCS/SGMII only** option is not available.

Interface

This setting determines the Ethernet-side interface for the MAC block. The following synthesis options are available:

- **MII/GMII**—If this is selected, Media Independent Interface (MII) is used for the 10/100 interface, and Gigabit Media Independent Interface (GMII) for the gigabit interface.
- **MII/RGMII**—If this is selected, MII is used for the 10/100 interface, and Reduced Gigabit Media Independent Interface (RGMII) for the gigabit interface.

These options are only available if the **Core Variation** is set to **10/100/1000 Mbps Ethernet MAC** only.

Use Transceiver Block

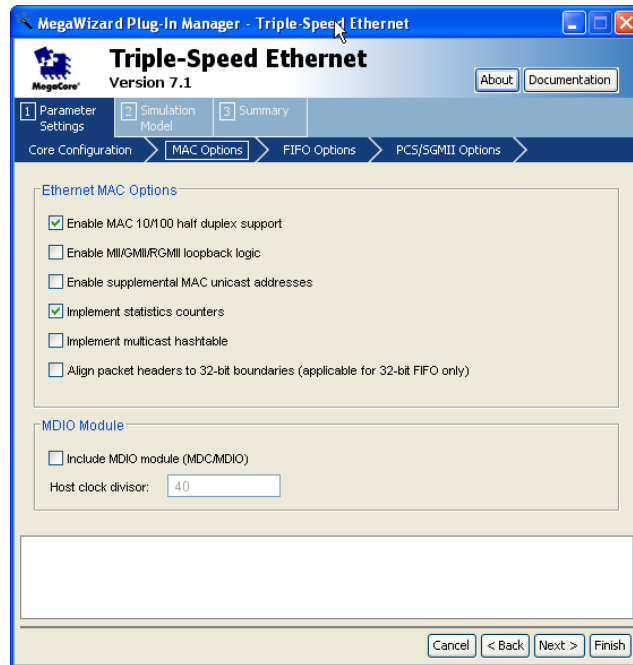
The **Use transceiver block** setting determines the external network interface for the PCS block. These options are only available if the MegaCore function includes the PCS block.

- When turned off, the PCS block implements a Ten-bit Interface (TBI) to an external SERDES chip or optical module.
- When turned on, the MegaCore function includes the GX transceivers to implement a 1.25 Gbps Medium Dependent Interface (MDI). This option is only available when targeting a device with GX transceivers.

MAC Options

The options on the **MAC Options** page allow you to configure the features of the MAC block. These options are only available if the MegaCore function includes the MAC block.

Figure 3–2. MAC Options



Ethernet MAC Options

Certain features of the MAC block are optional. The following options allow you to add or remove feature support based on your end-application needs.

The following options determine the logic included in the MAC block. These options provide flexibility for the designers to use only resources that are needed for their system needs and also simplify their design where needed.

- **Enable MAC 10/100 half duplex support**—Turn on this option to include support for half duplex operation on 10/100 Mbps connections.

- **Enable MII/GMII/RGMII loopback logic**—Turn on this option to include loopback logic on the MII, GMII, or RGMII interface. If you turn on this option, the loopback functionality can be dynamically enabled or disabled via the MAC configuration register space.
- **Enable supplemental MAC unicast addresses**—Turn on this option to include support for supplemental destination MAC unicast addresses for fast hardware-based received packet filtering.
- **Implement statistics counters**—Turn on this option to include support for simple network monitoring protocol (SNMP) management information base (MIB) and remote monitoring (RMON) statistics registers.
- **Implement multicast hashtable**—Turn on this option to include logic for a hash table to implement fast hardware-based multicast destination MAC address detection and filtering.
- **Align packet headers to 32-bit boundaries (applicable to 32-bit FIFO only)**—Turn on this option to include logic to align all packet headers to 32-bit boundaries. This option is only available when the FIFO width is 32 bits. This helps reduce software overhead processing in realignment of data buffers.



Turn on this option if you intend to use the Triple-speed Ethernet MegaCore function with the Interniche TCP/IP protocol stack.

MDIO Module

The following options control the PHY Management Module associated with the MAC block.

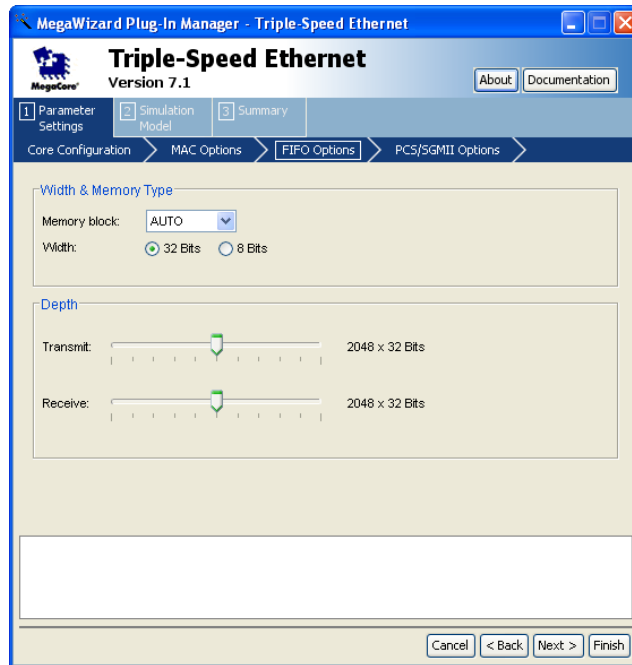
- **Include MDIO module (MDC/MDIO)**—Turn on this option to include the PHY management module. When turned off, the core does not include the logic or signals associated with the MDIO interface.
- **Host Clock Divisor** —This is a clock divisor to divide the MAC control register interface clock to produce the MDC clock output on the MDIO interface. Typically, the MDC clock should be 2.5 MHz.

For example, if the MAC control register interface clock frequency is 100 MHz and the desired MDC clock frequency is 2.5 MHz, a host clock divisor of 40 should be specified.

FIFO Options

The options on the **FIFO Options** page allow you to configure the transmit and receive FIFOs in the MAC block. These settings are only available if the MegaCore function includes the MAC block.

Figure 3–3. FIFO Options



Width and Memory Type

The following options allow you to set the width of the transmit and receive FIFOs and the memory block type used in their implementation.

- **Memory block**—Determines the type of memory block to be used by the Quartus II software to implement the FIFO Memory. Possible options are M4K, M9K, M144K, MRAM and AUTO. The options available depend on your targeted Altera device family.
- **Width**—Determines the data width of the transmit and receive FIFOs. The available widths are 8 and 32 bits. Set the data width to 32 bits if you intend to use the Triple Speed Ethernet MegaCore function with the Interniche TCP/IP protocol stack.

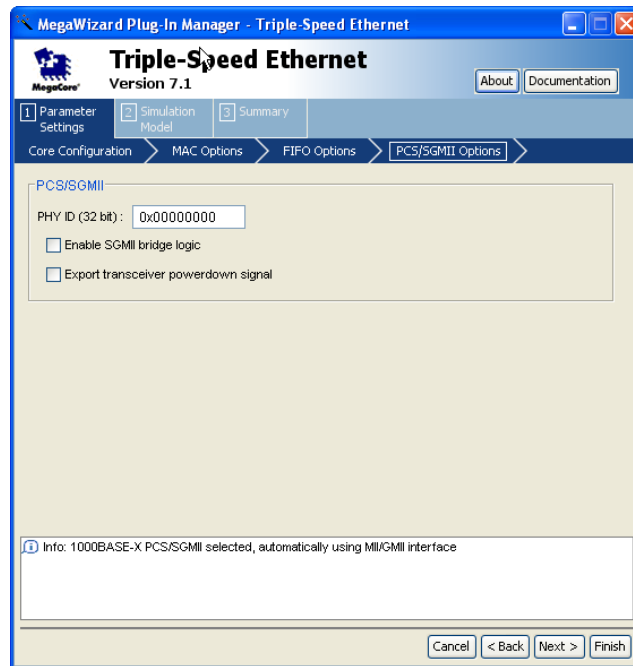
Depth

These options define the depths of the transmit and receive FIFOs. Available depths range between 64 and 64k in powers of two.

PCS/SGMII Options

The options on the **PCS/SGMII Options** page allows you to configure the PCS block. These options are only available if the MegaCore function includes the PCS block.

Figure 3–4. PCS/SGMII Options



PCS/SGMII

- **PHY ID (32 bit)**—Configures the PHY ID of the PCS block. For details, see [“MDIO Registers” on page 4–30](#).
- **Enable SGMII bridge logic**—Turn on this option to add SGMII clock and rate-adaptation logic to the PCS block. If your application requires 1000BASE-X PCS functionality only, turning off this option results in resource usage savings.

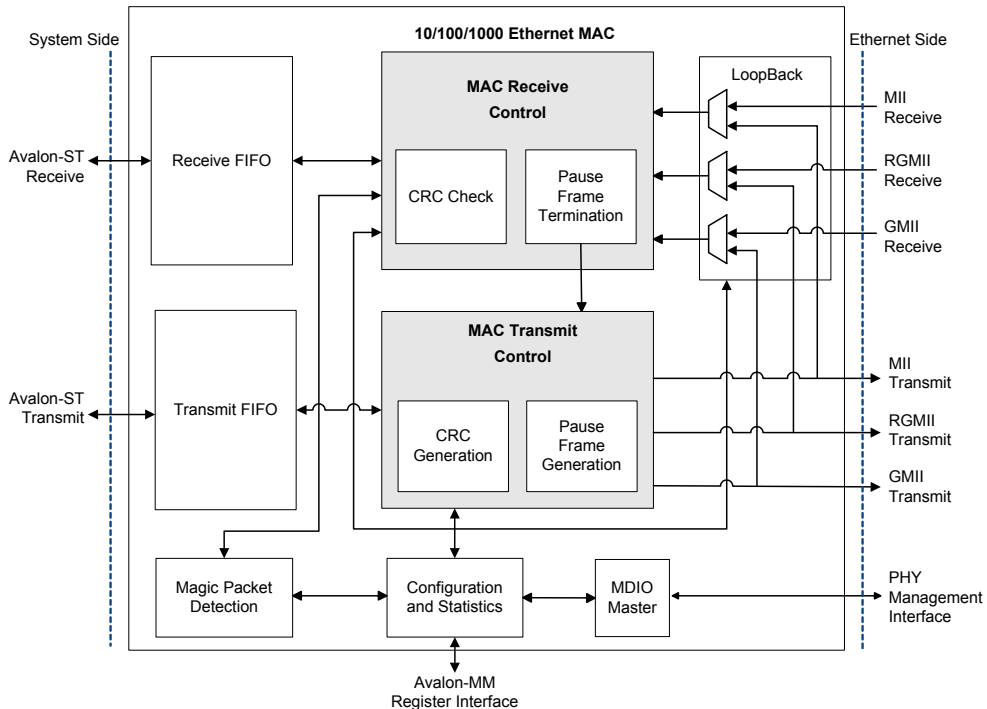
- **Export transceiver powerdown signal**—Turn on this option to export the powerdown signal of the GX transceiver to the top-level of your design. Since powerdown functionality is shared across Quad-port transceiver blocks in GX devices, this option is useful in multiport Ethernet designs to maximize efficient use of transceivers within a given quad-port block. Turn off this option to connect the powerdown signal internally to the PCS control register interface to control transceiver powerdown by the host processor in your system.

10/100/1000 Ethernet MAC

The 10/100/1000 Ethernet MAC function is connected to application logic via an Avalon® Streaming (Avalon-ST) interface. The Ethernet side of the MAC function provides an industry standard interface to an external PHY via one of the three possible media independent interfaces; Media Independent Interface (MII), Gigabit Media Independent Interface (GMII), or Reduced Gigabit Media Independent Interface (RGMII).

Figure 4–1 shows a block diagram of the MAC function.

Figure 4–1. MAC Block Diagram



Ethernet Frame Format

According to the IEEE 802.3 Standard, an Ethernet frame has a minimum length of 64 bytes and a maximum length of 1518 bytes, excluding the preamble and the SFD bytes. An Ethernet frame consists of the following fields:

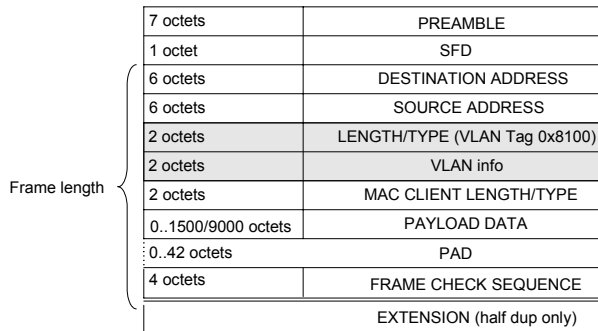
- Seven bytes preamble.
- Start frame delimiter (SFD).
- Two address fields.
- Length or type field.
- Data field.
- Frame check sequence (FCS) which is a CRC value.
- An extension field—defined only for gigabit Ethernet half-duplex implementations and is not supported by the MAC function.

Figure 4-2 shows a MAC frame format.

Figure 4-2. MAC Frame Format

Frame length {	7 octets	PREAMBLE
	1 octet	SFD
	6 octets	DESTINATION ADDRESS
	6 octets	SOURCE ADDRESS
	2 octets	LENGTH/TYPE
	0..1500/9000 octets	PAYLOAD DATA
	0..46 octets	PAD
	4 octets	FRAME CHECK SEQUENCE
	EXTENSION (half dup only)	

Optionally, MAC frames can be Virtual Local Area Network (VLAN) tagged with an additional 4-byte field containing VLAN Tag and VLAN Info. The additional 4-byte field is inserted between the MAC Source Address and Length/Type fields. VLAN tagging is defined by the IEEE P802.1Q Specification. VLAN tagged frames have a maximum length of 1522 bytes, excluding the preamble and the SFD bytes. Figure 4-3 shows a MAC frame format with optional VLAN tag fields.

Figure 4–3. VLAN Tagged MAC Frame Format

In certain applications, MAC frames can be tagged with stacked VLANs, two consecutive VLAN Tags. An additional 8-byte field is inserted between the MAC Source Address and Length/Type fields. [Figure 4–4](#) illustrates this format.

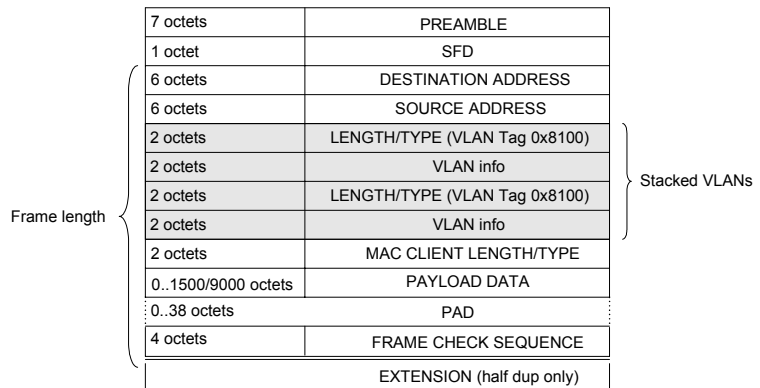
Figure 4–4. Stacked VLAN Tagged MAC Frame Format

Table 4–1 describes the elements of a MAC frame.

Table 4–1. MAC Frame Elements	
Element	Description
Frame Length	The length, in octets, is the length of a complete frame without the preamble and SFD bytes. A frame is of valid length if it contains at least 64 octets and does not exceed the maximum length configured in the <code>frm_length</code> register, which is typically 1518.
Length/Type	<p>The following conditions determines the content of this field:</p> <ul style="list-style-type: none"> • Value less than 1536 (0x600): Length field • Value greater than or equal to 1536 (0x600): Type field <p>The most significant byte of this field is sent and received first.</p>
Destination and Source Addresses	48-bit MAC addresses. The least significant byte is sent and received first. See “ MAC Address Checking ” on page 4–6 for more information on how to identify the different address types.

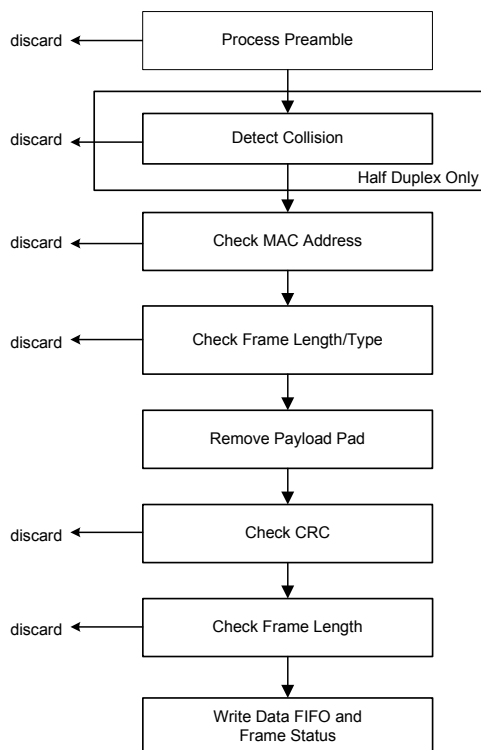


Although the IEEE specification defines a maximum frame length, the MAC function provides users the flexibility to configure the maximum frame length to any value.

MAC Receive Operation

This section describes the MAC receive operation. Figure 4-5 illustrates the flow of the MAC receive operation.

Figure 4-5. MAC Receive Flow



Preamble Processing

The IEEE 802.3 Standard allows a maximum size of 56 bits (7 bytes) for the preamble, while the MAC function allows any arbitrary preamble length. The MAC function checks for the SFD byte, which is 0xD5. If the SFD byte is not found after the last preamble byte (0x55), the frame is discarded.

Although the IEEE specification specifies that frames should be separated by at least 96 bits Inter Packet Gap (IPG), the MAC function is designed to accept frames separated by only 48 and 64 bits in GMII (1000 Mbps operation) and MII (10/100 Mbps operation), respectively.

The MAC function removes all preamble and SFD bytes.

Collision Detection in Half-Duplex Mode

When a MAC function is configured to operate in half-duplex mode, it checks if frames are received with collision. If a collision is detected when receiving the first 64 bytes, the frame is either discarded or transmitted to the user application with an error.

MAC Address Checking

Bit 0 in the destination address specifies the type of MAC address.

- If bit 0 is 0, the MAC address is a unicast (individual) address.
- If bit 0 is 1, the MAC address defines a multicast (group) address.
- If all 48 bits in the destination address are 1, it is a broadcast address.

Unicast Address Checking

When a unicast address is received, the destination MAC address is compared to the following addresses:

- The MAC address configured by the user application in the registers `mac_0` and `mac_1`.
- The supplemental MAC addresses configured in the registers `smac_0_0/smac_0_1`, `smac_1_0/smac_1_1`, `smac_2_0/smac_2_1` and `smac_3_0/smac_3_1`.

If the destination address matches any of the configured MAC addresses, the frame is accepted. Otherwise, the frame is discarded. For more information on these registers, refer to [Table 4–13 on page 4–32](#).

If only one MAC address is required, all supplemental MAC addresses should be configured with the node MAC address.

If promiscuous mode is enabled (the `PROMIS_EN` bit in the `command_config` register is set to 1), no address checking is performed and all received frames are accepted.

Multicast Address Resolution

Multicast addresses are, in typical implementations, resolved using a software task running on the system host processor. If the multicast address resolution generates acceptable processor load for 10 Mbps or 100 Mbps Ethernet connections, with gigabit Ethernet connections, this task can significantly load the host processor. To reduce the load from the host processor, the MAC function implements a hardware multicast address resolution engine.

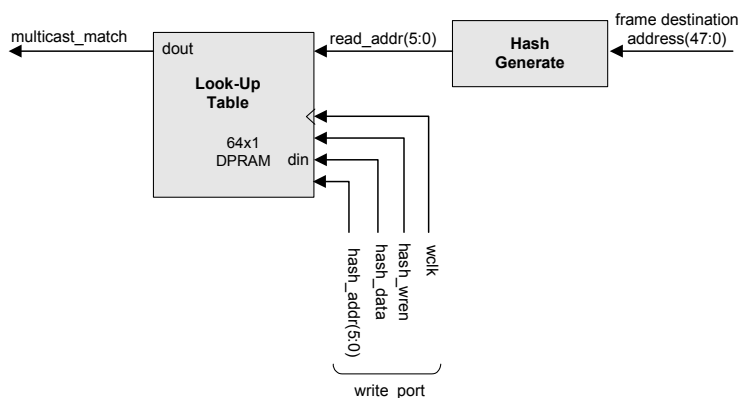
A 64-entry hash table is calculated and written by the host processor in a 64x1 look-up table (dual-ported RAM). When a multicast frame is received, the MAC address decoding function sends the destination MAC address to a hashing code generation function, which reduces the destination MAC address to a 6-bit code. The look-up table, programmed by the host processor is used as a fast matching engine, which, in one clock cycle, indicates if the hash code is or is not programmed in the table. If the hash code matching fails, the frame is rejected.

The 6-bit hash code can optionally be generated from:

- The complete 48-bit destination MAC address when the `MHASH_SEL` bit in the `command_config` register is 0.
- The lower 24 bits of the destination MAC address when the `MHASH_SEL` bit is 1. This option is provided to ignore the manufacturer code that is typically coded on the upper 24 bits of the MAC address and which is therefore typically a static field.

If all entries of the multicast table are programmed to 1, all multicast frames are accepted. If all entries of the multicast table are programmed to 0, all multicast frames are rejected.

Figure 4–6. Multicast Address Resolution Overview



In FPGA devices, each look-up table entry is typically set to 0x00 and all multicast frames are then subsequently rejected.

To build the hash table, the host processor generates a 6-bit code by XORing the MAC address bits as detailed in [Table 4–2](#). The 6-bit code is used to address the look-up table. For each code (look-up address), a one

indicates that all multicast MAC addresses represented by the code should be accepted, and a zero indicates that all multicast MAC addresses represented by the code should be rejected.

Table 4–2. Hash Code Generation from Full 48-Bit MAC Address

Hash Code Bit	Value
0	xor multicast MAC address bits 7:0
1	xor multicast MAC address bits 15:8
2	xor multicast MAC address bits 23:16
3	xor multicast MAC address bits 31:24
4	xor multicast MAC address bits 39:32
5	xor multicast MAC address bits 47:40

Table 4–3 shows the algorithm used to generate the hash code from the lower 24 bits of the MAC address.

Table 4–3. Hash Code Generation from Lower 24-Bit MAC Address

Hash Code Bit	Value
0	xor multicast MAC address bits 3:0
1	xor multicast MAC address bits 7:4
2	xor multicast MAC address bits 11:8
3	xor multicast MAC address bits 15:12
4	xor multicast MAC address bits 19:16
5	xor multicast MAC address bits 23:20

The MAC function always accepts broadcast frames with destination address set to a broadcast address.

Frame Length/Type Checking

If the Length/Type field contains the frame length (value less than 0x600), the MAC function checks the payload length and reports any error in the frame status word, `rx_err(1)`. Otherwise, the MAC function forwards the frame to the user application.

Control and VLAN frames, indicated by values of 0x8808 and 0x8100, respectively, are processed by the MAC function as described in the following sections.

VLAN Frame Processing

The length/type field in VLAN frames is set to 0x8100, followed by a 16-bit VLAN control information field. VLAN tagged frames are received as normal frames and forwarded to user applications, including the VLAN tag.

If the MAC client length/type field of the VLAN tagged frame, found 4 octets later in the frame is less than 42 and the `PAD_EN` bit in the `command_config` register is set to 1, padding is removed. The MAC function sets bit 16 of the signal `rx_err_stat` to indicate that the current frame is VLAN tagged.

Stacked VLAN Frame Processing

If the length/type field of the VLAN tagged frame, found 4 octets later in the frame is 0x8100, stacked VLANs are detected.

If the MAC client length/type field of the double VLAN tagged frame, which is found 4 octets after the second VLAN Tag is less than 38 and the `PAD_EN` bit in the `command_config` register is set to 1, padding is removed. The MAC function sets bit 17 of the signal `rx_err_stat` to indicate that the current frame is stacked VLAN tagged.

Pause Frame Termination

Pause frames are terminated within the receive engine and not transferred to the receive FIFO. The quanta is extracted and sent to the MAC transmit path via a small internal clock rate decoupling asynchronous FIFO. The quanta is written only if a correct CRC and a correct frame length are detected by the control state machine. Otherwise, the quanta is discarded.

When a pause frame is received, the statistic counter `aPAUSEMACCtrlFramesReceive` is incremented.

Command Frame Filtering

If a MAC control frame, frame type field 0x8808, with an opcode other than 0x0001 is received, the control frame can be rejected, when the `CNTL_FRM_ENA` bit in the `command_config` register is 0. If the bit is 1, the control frame is accepted and forwarded to the internal system interface.

Pause frames, which are MAC control frames with the opcode set to 0x0001, are always accepted regardless of the value of the `CNTL_FRM_ENA` bit. Pause frames are controlled by the `PAUSE_FWD` and `PAUSE_IGNORE` bits in the `command_config` register.

Payload Pad Removal

When a frame is received with a payload length lower than 46 Bytes (42 Bytes for VLAN tagged frames and 38 Bytes for frames with stacked VLANs), the zero padding can be optionally removed before the frame is written to the data FIFO if the `PAD_EN` bit in the `command_config` register is set to 1.

If the `PAD_EN` bit in the `command_config` register is set to 0, complete frames including the zero padding are transmitted to the MAC function FIFO interface.

Frame Length Checking

The MAC function checks the complete frame length to ensure the length conforms to the following conditions:

- Frame length is not less than 64 bytes for all types of frames.
- For untagged frames, frame length is not greater than the maximum length configured in the `frm_length` register which is typically 1518.
- For VLAN tagged frames, the frame length is not greater than the maximum length configured in the `frm_length` register plus 4.
- For stacked VLAN tagged frames, the frame length is not greater than the maximum length configured in the `frm_length` register plus 8.

If the frame length is greater than the maximum length allowed, the frame is truncated and marked invalid by setting the frame status bit `rx_err(1)` to 1.

If the `NO_LGTH_CHECK` bit in the `command_config` register is set to 0, the MAC function also checks if the frame payload length is consistent with the frame length field if the length/type field is set to the any of the following value:

- Between 0x2E and 0x0600, excluding 0x600 for untagged frames (length/type).
- Between 0x2A and 0x0600, excluding 0x600 for VLAN tagged frames (MAC client length/type field).
- Between 0x26 and 0x0600, excluding 0x600 for stacked VLAN tagged frames (MAC client length/type field).

In the case of mismatch, the received frame status bit `rx_err(1)` is set to 1 to indicate length error. If the `RX_ERR_DISC` bit in the `command_config` register is set to 1, received frames with length error are discarded.

The payload length check is disabled when the register bit `NO_LGTH_CHECK` is 1.

CRC Checking

The CRC-32 field is checked and forwarded to the MAC receive FIFO if the `CRC_FWD` and `PAD_EN` bits in the `command_config` register are 1 and 0, respectively. When the `CRC_FWD` bit is 0, the CRC-32 field is checked and terminated, not forwarded to the FIFO interface.

The CRC polynomial, as specified in the 802.3 Standard, is shown in the following equation:

$$FCS(X) = X^{31} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X^1 + 1$$

The 32 bits of the CRC value are placed in the FCS field so that the X^{31} term is the right-most bit of the first octet. The CRC bits are thus received in the following order: $X^{31}, X^{30}, \dots, X^1, X^0$.

If a CRC-32 error is detected, the frame is marked invalid and the frame status, `rx_err(2)` is set to 1. The MAC function discards received frames with CRC-32 error if the `RX_ERR_DISC` bit in the `command_config` register is 1.

Receive FIFO Thresholds

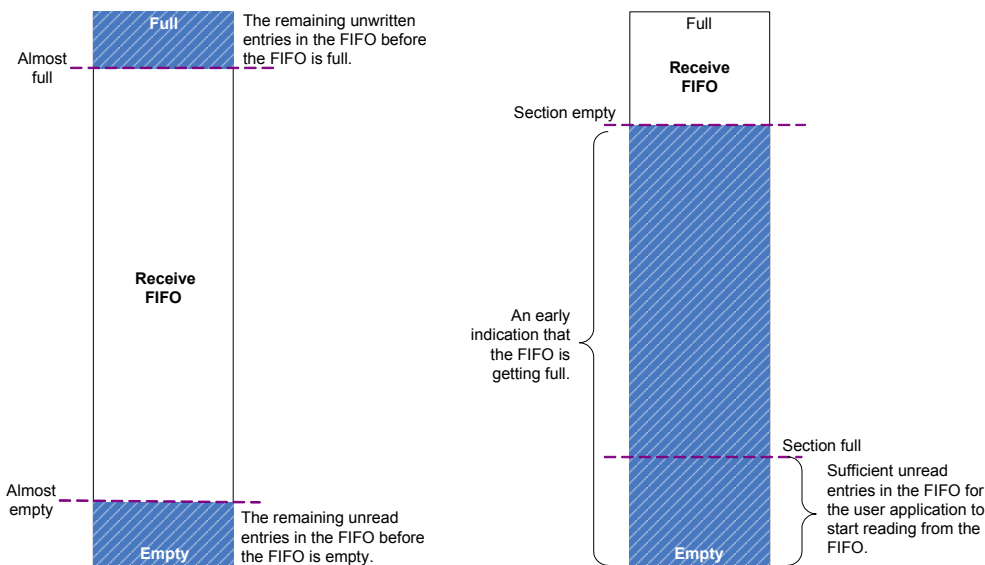
You can configure FIFO thresholds to dynamically change FIFO operations and effectively manage potential FIFO overflow or underflow.

There are four receive FIFO thresholds which you can configure:

- Almost empty
- Almost full
- Section empty
- Section full

These thresholds are defined in bytes for 8-bit wide FIFOs and words for 32-bit wide FIFOs. [Figure 4–7](#) illustrates the receive FIFO thresholds.

Figure 4–7. Receive FIFO Thresholds



The receive FIFO thresholds are configured via the registers. For more information on the threshold registers, refer to [Table 4–13](#) on [page 4–32](#).

Table 4–4 describes how each threshold can be used to change and manage FIFO operations.

Table 4–4. Receive Threshold Registers Description

Threshold	Register Name	Description
Almost empty	<code>rx_almost_empty</code>	The number of unread entries in the FIFO before the FIFO is empty. When the FIFO level reaches this threshold, the MAC receive control stops reading from the FIFO and subsequently stops transferring data to the user application to avoid FIFO underflow.
Almost full	<code>rx_almost_full</code>	<p>The number of unwritten entries in the FIFO before the FIFO is full. When the FIFO level reaches this threshold and the user application is not ready to receive data (<code>ff_rx_rdy</code> is 0), the MAC receive control performs the following operations:</p> <ul style="list-style-type: none"> • Stops writing data to the FIFO • Truncates received frames to avoid FIFO overflow • Asserts the signal <code>rx_err[0]</code> simultaneously with the assertion of the signal <code>ff_rx_eop</code>. • Marks the truncated frame invalid by setting <code>rx_err_stat(3)</code> to 1. <p>If the <code>RX_ERR_DISC</code> bit in the <code>command_config</code> register is set to 1 and the section-full (<code>rx_section_full</code>) threshold is set to 0, erroneous frames received on the FIFO interface are discarded.</p>
Section empty	<code>rx_section_empty</code>	<p>An early indication that the FIFO is getting full. When the FIFO level reaches the section-empty threshold, the MAC transmit generates an XOFF pause frame to indicate FIFO congestion to the remote Ethernet device. When the FIFO level goes below the section-empty threshold, the MAC transmit generates an XON pause frame to indicate its readiness to receive new frames.</p> <p>You can use this threshold to avoid loss of data by providing an early warning to the remote Ethernet device about potential FIFO congestion before the FIFO level hits the almost-full threshold, upon which received frames are truncated.</p>
Section full	<code>rx_section_full</code>	<p>The section-full threshold indicates that there are sufficient entries in the FIFO for the user application to start reading from it. The signal <code>ff_rx_dsav</code> is asserted when the FIFO level reaches the section-full threshold.</p> <p>Set this threshold to 0 to enable store and forward on the receive datapath. When store and forward is enabled, the signal <code>ff_rx_dsav</code> is asserted as soon as a complete frame is written to the FIFO.</p>

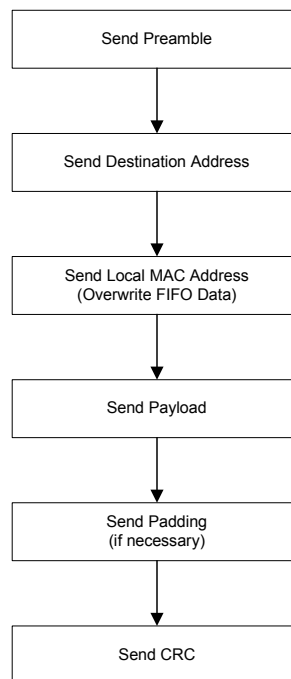
MAC Transmit Operation

Frame transmission starts when the transmit FIFO holds enough data. Once a transfer has started, the MAC transmit function performs the following tasks:

- Generates preamble and SFD field before frame transmission
- Generates XOFF pause frames if the receive FIFO reports a congestion or if the `xoff_gen` signal is asserted
- Generates XON pause frames if the receive FIFO congestion condition is cleared or if the `xon_gen` signal is asserted
- Suspends Ethernet frame transfer (XOFF) if a non-zero pause quanta is received from the MAC receive path
- Adds padding to the frame if required
- Calculates and appends CRC-32 to the transmitted frame, if required
- Sends frame with correct interpacket gap (IPG)

Figure 4-8 illustrates the MAC transmit flow.

Figure 4-8. MAC Transmit Flow



In half-duplex mode, the following additional tasks are performed:

- Collision detection.
- Frame retransmission when the backoff timer expires.

MAC Address Insertion

If the TX_ADDR_INS bit in the `command_config` register is set to 1, the source MAC address in frames received from the MAC transmit FIFO is replaced with the MAC primary address or supplemental address as specified by the TX_ADDR_SEL bits in the `command_config` register. For more information on the selection, refer to [Table 4-15 on page 4-39](#).

The source MAC address is forwarded to the Ethernet-side interface if the TX_ADDR_INS bit in the `command_config` register is set to 0.

Frame Payload Padding

The IEEE specification defines a minimum frame length of 64 bytes. To avoid violating this specification, the MAC function automatically inserts padding bytes (0x00) if it receives frames with payload length less than the following number of bytes from the user application:

- 46 bytes for untagged frames.
- 42 bytes for VLAN tagged frames.
- 38 bytes for stacked VLAN tagged frames.

CRC-32 Generation

The CRC-32 field is generated and appended at the end of a frame if the frame is transmitted to the MAC function with the signal `ff_tx_crc_fwd` set to 0 when `ff_tx_eop` is asserted.

The CRC polynomial, as specified in the IEEE 802.3 Standard, is shown in the following equation:

$$\text{FCS}(X) = X^{31} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X^1 + 1$$

The 32 bits of the CRC value are placed in the FCS field so that the X^{31} term is the right-most bit of the first octet. The CRC bits are thus transmitted in the following order: X^{31} , X^{30} , ..., X^1 , X^0 .

Inter Packet Gap

In full-duplex mode, the IPG programmed in the `tx_ipg_length` register is maintained between transmissions. The minimum IPG can be programmed to any value between 64 and 216 bit times, where 64 bit times is the time it takes to transmit 64 bits of raw data on the medium.

In half-duplex mode, the MAC function constantly monitors the line. Actual data transmission occurs only if the line has been idle for a period of 96 bit times and any backoff time requirements have been satisfied. In accordance with the standard, the MAC function begins to measure the IPG from the deassertion of the signal `m_rx_crs`.

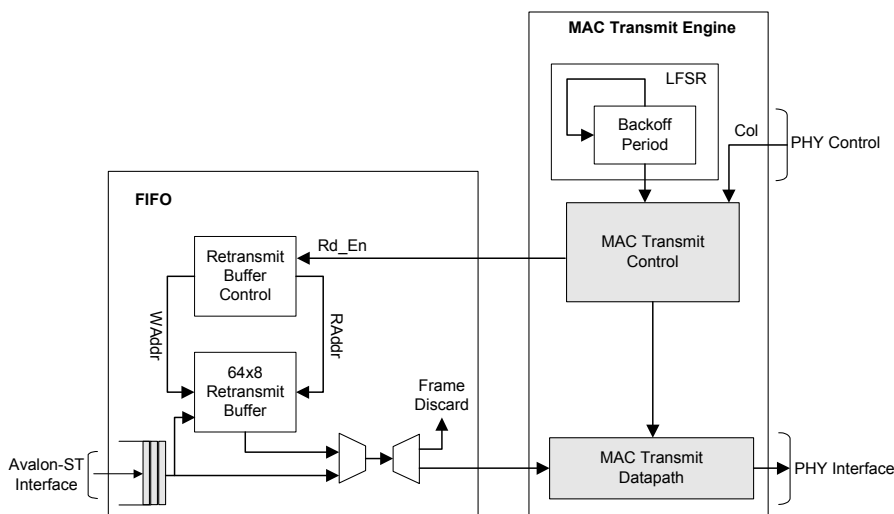
Collision Detection in Half-Duplex Mode

A collision occurs in a half-duplex network when two or more nodes transmit concurrently. During transmission, the MAC function monitors the line condition and detects a collision when the PHY device asserts the `m_rx_col` signal.

If a collision is detected during transmission, the MAC function stops the transmission and sends a 32-bit jam pattern instead.

If a collision is detected while transmitting the preamble or SFD byte, the MAC function sends the jam pattern only after the SFD byte is transmitted. This results in a minimum of 96-bit fragment. A jam pattern is a fixed pattern, 0x648532A6, and not compared to the actual frame CRC. The probability of a jam pattern to be identical to the CRC is very low, 0.532.

If a collision occurs before the transmission of 64 bytes, including the preamble and SFD, the MAC function waits for an interval equal to the backoff period and then retransmits the packet data stored in a 64-byte retransmit buffer. The backoff period is generated from a pseudo random process, truncated binary exponential backoff. [Figure 4–9](#) illustrates packet retransmission.

Figure 4–9. Packet Retransmission

The backoff time is a multiple of slot times. One slot is equal to a 512 bit times period. The number of the delay slot times, before the N^{th} retransmission attempt, is chosen as a uniformly distributed random integer in the following range:

$$0 \leq r < 2^k$$

$k = \min(n, N)$, where n is the number of retransmissions and $N = 10$

For example, after the first collision, the backoff period, in slot time, is 0 or 1. If a collision occurs on the first retransmission, the backoff period, in slot time, is 0, 1, 2 or 3.

The maximum backoff time, in 512 bit times slots, is limited by N set to 10 as specified in the IEEE 802.3 Standard.

If a collision occurs after 16 consecutive retransmissions, the MAC function reports an excessive collision condition by setting the `EXCESS_COL` bit in the `command_config` register to 1, and discards the current packet from the FIFO.

In networks that violate standard requirements, a collision may occur after transmission of the first 64 bytes. When the standard is violated, the MAC function stops the current packet transmission and discards the rest of the packet from the transmit FIFO. The MAC function resumes transmitting the next available packet in the transmit FIFO.

Transmit FIFO Thresholds

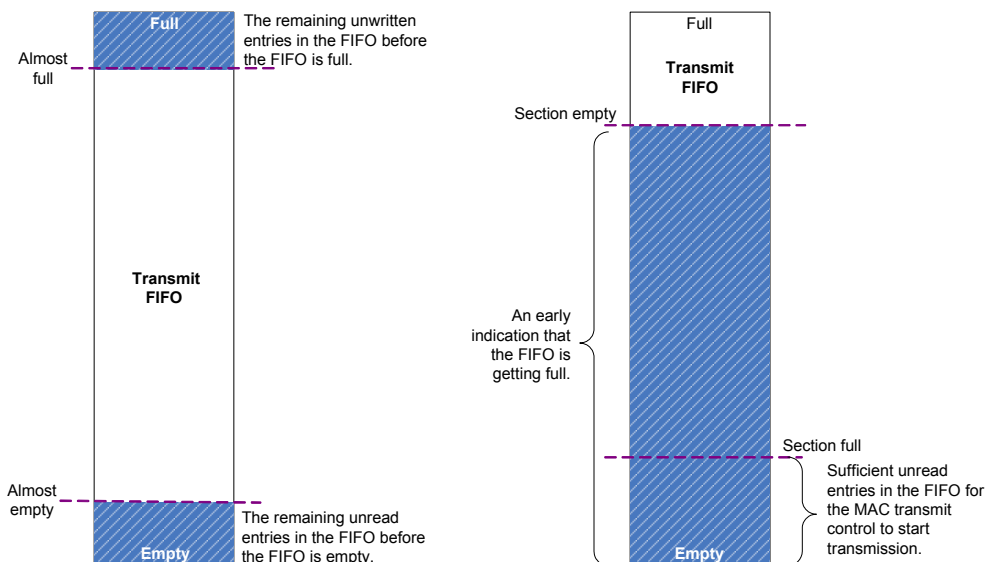
You can configure FIFO thresholds to dynamically change FIFO operation and effectively manage potential FIFO overflow or underflow.

There are four transmit thresholds which you can configure:

- Almost empty
- Almost full
- Section empty
- Section full

These thresholds are defined in bytes for 8-bit wide FIFOs and words for 32-bit wide FIFOs. [Figure 4-7](#) illustrates the transmit thresholds.

Figure 4-10. Transmit FIFO Thresholds



The transmit FIFO thresholds are configured via the registers. For more information on the threshold registers, refer to [Table 4-13 on page 4-32](#).

Table 4-5 describes how each threshold can be used to change and manage FIFO operations.

Table 4-5. Threshold Registers Description		
Threshold	Register Name	Description
Almost empty	tx_almost_empty	The number of unread entries in the FIFO before the FIFO is empty. When the FIFO level reaches this threshold, the MAC transmit control stops reading from the FIFO and sends the Ethernet frame with an GMII / MII/ RGMII error indication to avoid FIFO underflow.
Almost full	tx_almost_full	The number of unwritten entries in the FIFO before the FIFO is full. When the FIFO level reaches this threshold, the MAC transmit control performs the following operations: <ul style="list-style-type: none"> • Deasserts the signal <code>ff_tx_rdy</code>. • Truncates the current frame. • Sets the ERROR status to avoid FIFO overflow.
Section empty	tx_section_empty	An early indication that the FIFO is getting full. When the FIFO level reaches the section-empty threshold, the signal <code>ff_tx_septy</code> is deasserted. You can use this threshold to warn the user application about potential FIFO congestion.
Section full	tx_section_full	The section-full threshold indicates that there are sufficient entries in the FIFO for the MAC transmit control to start frame transmission. Set this threshold to 0 to enable store and forward on the transmit datapath. When store and forward is enabled, the MAC function forwards each frame as soon as it is completely written to the transmit FIFO.

Transmit FIFO Underflow

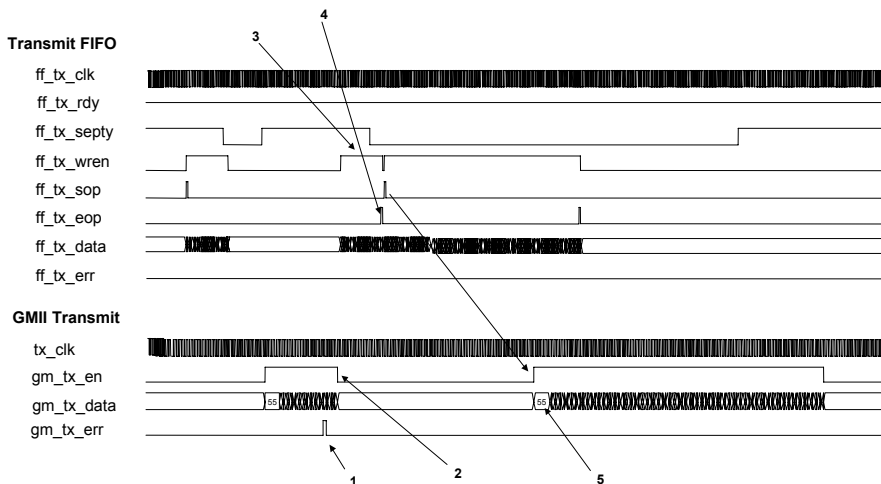
During a frame transfer, if the transmit FIFO reaches the almost-empty threshold with no end of frame indication stored in the FIFO, the MAC transmit control stops reading data from the FIFO and initiates the following actions:

1. The MAC sets the RGMII/GMII/MII error signal (`tx_control/gm_tx_err/m_tx_err`) to 1 to indicate that the fragment transferred is not valid.
2. The MAC drives the RGMII/GMII/MII transmit enable signal (`tx_control/gm_tx_en/m_tx_en`) to terminate the frame transfer.
3. After the underflow, the application completes the frame transfer.
4. The MAC transmit control discards any new data in the FIFO until the end of packet is reached.

5. The MAC starts to transfer data on the RGMII/GMII/MII interface when the application sends a new frame with a start of frame indication.

Figure 4–11 illustrates the FIFO underflow protection algorithm for Gigabit Ethernet system.

Figure 4–11. Transmit FIFO Underflow Protection



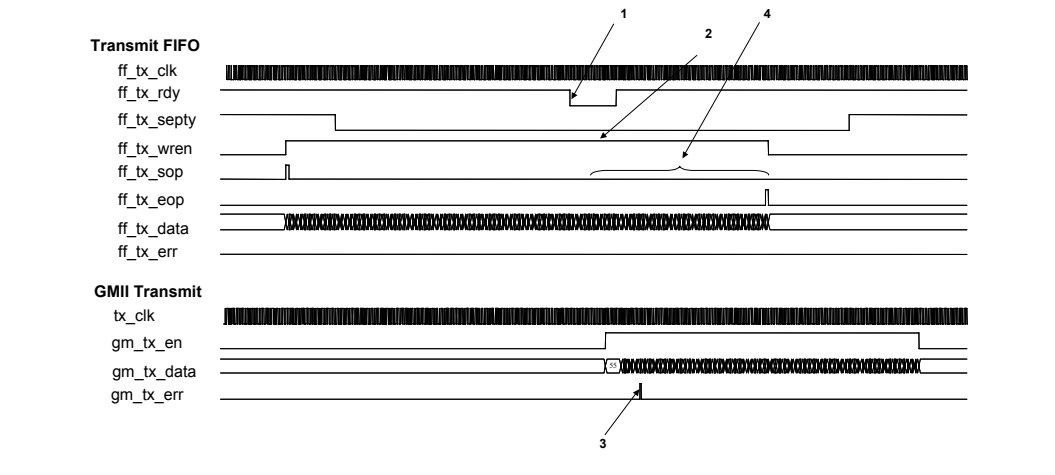
Transmit FIFO Overflow

On the transmit path, when the FIFO reaches the almost-full threshold, the MAC transmit control initiates the following actions:

1. The MAC ready signal **ff_tx_rdy** is de-asserted.
2. The user application continues to assert the signal **ff_tx_wren**.
3. The FIFO protection logic truncates the frame which is sent on the RGMII/GMII/MII interface with an error (**tx_control/gm_tx_err/m_tx_err**).
4. The MAC function discards any new data from the user application after the frame truncation.

Figure 4–12 illustrates the FIFO overflow protection algorithm for Gigabit Ethernet system.

Figure 4–12. Transmit FIFO Overflow Protection



MAC Receive and Transmit FIFO Interfaces

The MAC receive and transmit FIFO interfaces are Avalon-ST compliant ports and adhere to the *Avalon-ST Interface Specification*.



For Information About	Refer To
Avalon-ST interface protocol	Avalon Streaming Interface Specification

Frames on the FIFO interface do not contain any preamble or SFD, which are inserted and discarded by the MAC function on transmit and receive, respectively. The frame length must not exceed the maximum length configured in the `frm_length` register.



The standard requires that the least significant byte of the MAC address is sent and received first. For all the other header fields, including length/type, VLAN tag, VLAN info and pause quanta, the most significant byte is sent and received first.

Decoded information on received frames can be found in `rx_err(5:0)`, `rx_frm_type(3:0)` and `rx_err_stat(17:0)`. The information is valid when the signal `ff_rx_eop` is asserted.

The network stack makes frequent use of the IP addresses stored in Ethernet frames. By padding the beginning of an Ethernet frame by two bytes, those IP addresses are aligned on a four-byte boundary. The padding of Ethernet frames are determined by the registers `tx_cmd_stat` and `rx_cmd_stat` on transmit and receive, respectively.

Table 4-6 illustrates the structure of a non-IP aligned Ethernet frame.

<i>Table 4-6. 32-Bit Interface Data Structure — Non-IP aligned</i>			
31...24	23...16	15...8	7...0
Byte 0	Byte 1	Byte 2	Byte 3
Byte 4	Byte 5	Byte 6	Byte 7

Table 4-7 illustrates the structure of an IP aligned Ethernet frame.

<i>Table 4-7. 32-Bit Interface Data Structure — IP aligned</i>			
31...24	23...16	15...8	7...0
padded with zeros		Byte 0	Byte 1
Byte 2	Byte 3	Byte 4	Byte 5

MAC Full Duplex Flow Control Operation

Three conditions are handled by the MAC flow control engine:

- **Remote Device Congestion:** The remote device connected to the same Ethernet segment as the MAC function reports congestion and requests the MAC function to stop sending data.
- **MAC FIFO Congestion:** When the MAC receive FIFO reaches a user defined section-empty threshold (`rx_section_empty`), the MAC function sends a pause frame to the remote device requesting the remote device to stop sending data.
- **Local Device Congestion:** Any device connected to the MAC function can request the remote device to stop data transmission. This is typically done via the host processor.

Remote Device Congestion

When the MAC transmit control receives a valid pause quanta from the receive path, the MAC function completes the transfer of the current frame and stops sending data for the amount of time specified by the pause quanta in 512 bit times increments.

Frame transfer resumes when the time specified by the quanta expires, and no new quanta value or pause frame with a quanta value set to 0x0000 is received.

Local Device / FIFO Congestion

The MAC transmit control generates pause frames when the receive FIFO level hits the section-empty or almost-full threshold, or at the request of the user application.

To generate an XOFF pause frame, the user application sets the XOFF_GEN bit in the `command_config` register to 1. A pause frame is generated when the current frame transfer is completed and as long as the XOFF_GEN bit remains 1.

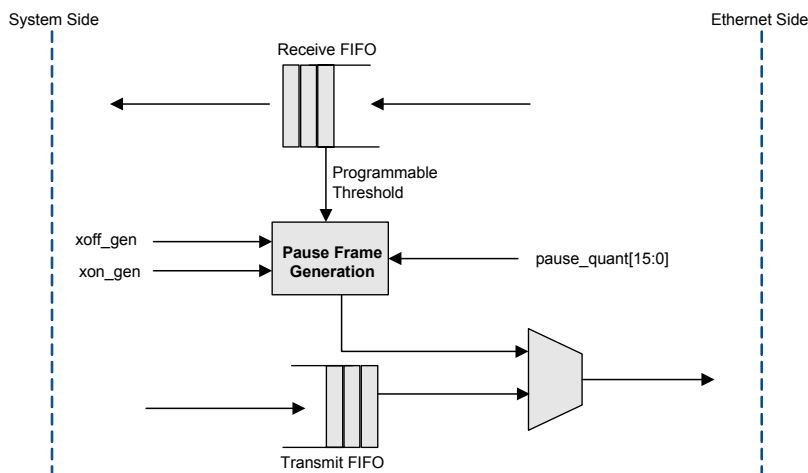
An XOFF pause frame is generated when the receive FIFO asserts its section empty flag (internal). A pause frame is generated automatically, when the current frame transfer is completed.

An XON pause frame is generated when the receive FIFO de-asserts its section empty flag (internal). An XON pause frame is generated automatically, when the current frame transfer is completed.

When an XOFF pause frame is generated, the pause quanta bytes P1 and P2 (see [“Pause Frames” on page 4–24](#)) are filled with the value configured in the `pause_quant` register. The source address is set to the MAC address configured in the `mac_0` and `mac_1` registers and the destination address is set to a fixed multicast address 01-80-C2-00-00-01 (0x010000c28001).

When an XON pause frame is generated, the pause quanta (payload bytes P1 and P2) is filled with 0x0000 (zero quanta). The source address is set to the MAC address configured in the `mac_0` and `mac_1` registers and the destination address is set to a fixed multicast address 01-80-C2-00-00-01 (0x010000c28001).

Pause frames generated are compliant to the IEEE 802.3 annex 31A & B. For more information on pause frames, refer to [“Pause Frames” on page 4–24](#).

Figure 4–13. Pause Frame Generation

Even though the flow control mechanism should prevent any FIFO overflow on the MAC receive path, the MAC receive FIFO is protected. When an overflow is detected on the receive FIFO, the current frame is truncated with an error indication set in the frame status bit, `rx_err(3)`. The user application should subsequently discard the frame by setting the `RX_ERR_DISC` bit in the `command_config` register to 1.

Pause Frames

A pause frame is generated by the receiving device to indicate congestion to the emitting device. The emitting device should stop sending data upon receiving pause frames if it supports flow control.

The Length/Type field of pause frames is always set to 0x8808. The first 2 bytes of pause frames following the Length/Type field, define a 16-bit opcode field. In pause frames, the opcode field is always set to 0x0001. A 16-bit pause quanta is defined in the frame payload bytes 2 (byte P1) and

3 (byte P2) as defined in Table 4–8. The pause quanta byte P1 is the most significant. A pause frame has no payload length field, and is always padded with 42 bytes (0x00).

Table 4–8. Pause Frame Format (Values in Hex)

1	2	3	4	5	6	7	8	9	10	11	12	13	14
55	55	55	55	55	55	55	D5	01	80	C2	00	00	01
Preamble							SFD	Multicast Destination Address					
15	16	17	18	19	20	21	22	23	24	25	26	27 - 68	
00	00	00	00	00	00	88	08	00	01	hi	lo	00	
Source Address							(Length)/Type	Opcode		P1	P2	pad (42)	
69		70		71		72							
26		6B		AE		0A							
CRC-32													

If a pause frame with a pause value greater than zero (XOFF condition) is received, the MAC function stops data transmission as soon as the current frame transfer is completed. Data transmission is suspended for the length of time defined by the value in the pause quanta. One pause quanta fraction refers to 512 bit times.

If a pause frame with a pause value of zero (XON condition) is received, data transmission resumes immediately.

Magic Packet Detection

A magic packet can be a unicast, multicast or broadcast packet which carries a defined sequence in the payload section. Magic packets are received and acted upon only under specific conditions, typically in power-down mode.

The defined sequence used to decode a magic packet is formed with a synchronization stream of six consecutive 0xFF bytes followed by sequence of 16 consecutive unicast MAC addresses. The unicast address is of the node to be awakened.

The sequence can be located anywhere in the magic packet payload and the magic packet is formed with a standard Ethernet header, optional padding and CRC.

Sleep Mode

To put the MAC function to sleep, set the `MAGIC_ENA` and `SLEEP` bits in the `command_config` register to 1.

When the MAC function is in sleep mode, the following operations are disabled:

- MAC transmit.
- MAC FIFO receive and transmit.

The MAC receive is kept in normal mode but it ignores all traffic from the line except magic frames, to allow a remote agent to wake up the node.

Magic Packet Detection

Magic packet detection wakes up a node that is in sleep mode. It is applicable only when the `MAGIC_ENA` and `SLEEP` bits in the `command_config` register are set to 1.

The MAC function detects magic frames with destination address set to any of the following addresses:

- Any multicast address.
- A broadcast address.
- The unicast address configured in the `mac_0` and `mac_1` registers.
- Any unicast addresses configured in the supplemental MAC address registers; `smac_0_0`, `smac_0_1`, `smac_1_0`, `smac_1_1`, `smac_2_0`, `smac_2_1`, `smac_3_0` and `smac_3_1`, if these registers are enabled.

When a magic frame is detected, the `WAKEUP` bit in the `command_config` register is asserted and none of the statistics registers is incremented.

Wakeup

Magic packet detection is disabled when the `SLEEP` bit in the `command_config` register is deasserted. Deasserting the `SLEEP` bit also resets the `WAKEUP` bit to 0 and resumes the MAC FIFO transmit and receive operations.

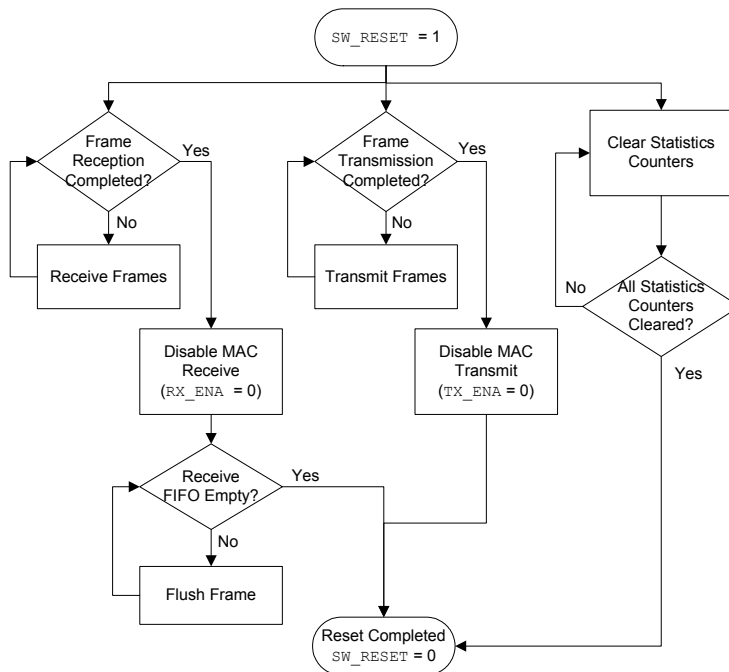
Software Reset

A software application can reset the MAC function by setting the `SW_RESET` bit in the `command_config` register to 1. During a software reset, the MAC function clears all statistics registers, flushes the MAC receive FIFO and disables the MAC transmit and receive by setting the `TX_ENA` and `RX_ENA` bits in the `command_config` register to 0.

The value of configuration registers, such as the MAC address and FIFO thresholds are preserved. The `SW_RESET` bit is cleared automatically when the software reset ends. For more information about the reset signal, refer to “[Command_Config Register](#)” on page 4–39.

Figure 4–14 illustrates the sequence of a software reset.

Figure 4–14. Software Reset Sequence



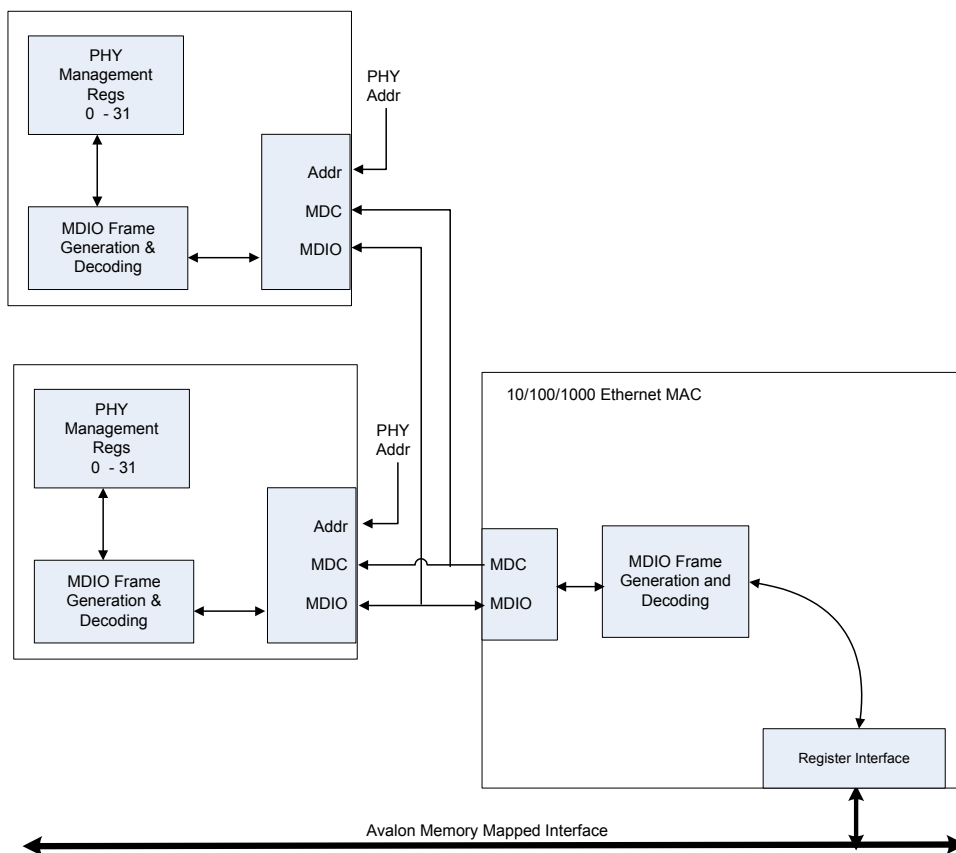
If the `SW_RESET` bit is 1 when the line clocks are not available, (for example, cable is disconnected), the statistics registers may not be cleared. The `READ_TIMEOUT` bit in the `reg_status` register is then set to 1 to indicate that the statistics registers were not cleared.

PHY Management (MDIO)

The Management Data Input/Output (MDIO) interface is a two-wire management interface. The MDIO management interface implements a standardized method to access the PHY device management registers. The MAC function implements a master MDIO interface, which can be connected to up to two PHY devices.

The PHY Device MDIO registers of the two PHY devices are mapped in the MAC register space and can be read and written from the Avalon-MM register interface. The MAC function provides the flexibility to access PHY devices with the MDIO address set to any legal value.

Figure 4–15. MDIO Interface



MDIO Frame Format

The MAC MDIO master controller communicates with the slave PHY device using frames, which are defined in Table 4–9. A complete frame is 64 bits long and consists of 32-bit preamble, 14-bit command, 2-bit bus direction change and 16-bit data. Each bit is transferred on the rising edge of the MDIO clock, the MDC signal. The PHY management interface supports the standard MDIO specification (IEEE803.2 Clause 22).

Table 4–9. MDIO Frame Formats (Read/Write)

Type		Command						
	PRE	ST MSB LSB	OP MSB LSB	Addr1 MSB LSB	Addr2 MSB LSB	TA	Data MSB LSB	Idle
Read	1 ... 1	01	10	xxxxx	xxxxx	Z0	xxxxxxxxxxxxxxxxx	Z
Write	1 ... 1	01	01	xxxxx	xxxxx	10	xxxxxxxxxxxxxxxxx	Z

Table 4–10 describes the fields of the MDIO frame.

Table 4–10. MDIO Frame Field Descriptions

Name	Description
PRE	Preamble. 32 bits of logical 1 sent prior to every transaction.
ST	Start indication. Standard MDIO (Clause 22): 0b01.
OP	The opcode defines whether a read or write operation is performed: <ul style="list-style-type: none"> 0b10: A read operation is performed. 0b01: A write operation is performed.
Addr1	The PHY device address (PHYAD). Up to 32 devices can be addressed. For PHY device 0, the Addr1 field is set to the value configured in the <code>mdio_addr0</code> register. For PHY device 1, the Addr1 field is set to the value configured in the <code>mdio_addr1</code> register.
Addr2	Register Address. Each PHY can have up to 32 registers.
TA	Turnaround time. Two bit times are reserved for read operations to switch the data bus from write to read for read operations. The PHY device presents its register contents in the data phase and drives the bus from the 2 nd bit of the turnaround phase.
Data	16-bit data written to or read from the PHY device.
Idle	Between frames, the MDIO data signal is tri-stated.

MDIO Registers

The host processor can access the MDIO registers of a PHY device connected to the MAC via an Avalon-MM interface. The PHY MDIO registers are mapped in the MAC address space. Each PHY device has 32 registers. [Table 4–11](#) lists and describes the MDIO registers. Registers 6 through 31 are specific registers of the PHY device implementation and not listed in the table.

Table 4–11. MDIO Register Descriptions

Register	Name	Access	Description (Bits)
0	Control	RW	PHY device operation control register.
1	Status	RO	PHY device operation status register.
2	PHY_ID1	RO	Bits 31:16 of PHY identifier.
3	PHY_ID2	RO	Bits 15:0 of PHY identifier.
4	Adv	RW	Auto-negotiation advertisement register.
5	RemAdv	RO	Remote partner base page ability.

MDIO Clock Generation

The Management Data Clock (MDC) is generated from the Avalon-MM interface clock signal, `clk`. The division factor is defined by specifying the value in the **Host clock divisor** parameter. For more information on the parameters, refer to [“MAC Options” on page 3–4](#).

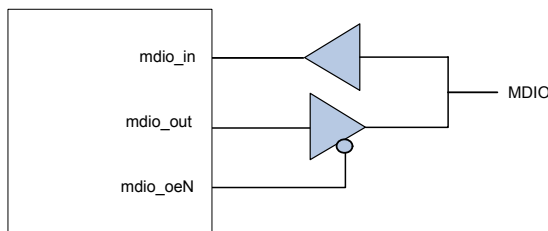


The division factor must be defined such that the MDC frequency does not exceed 2.5 MHz.

MDIO Buffer Connection

[Figure 4–16](#) illustrates the buffers used for the MDIO tri-state bus.

Figure 4–16. MDIO Buffer Connection



MAC Interface Register Map

The control interface to the Triple Speed Ethernet MegaCore function has a register space of 256 registers, providing access to all functional blocks within the MAC function.

Table 4–12 provides an overview of the register space for the MAC function.

Table 4–12. MAC Interface Register Map Overview		
Addr Offset	Section	Description
0x000 – 0x05C	MAC Block Configuration	Base register set to configure the MAC block.
0x060 – 0x0DC	Statistics Counters	Counters collecting traffic statistics.
0x0E8–0x0EB	TX Command Register	Transmit datapath control register. See table Figure 4–18 and Table 4–18 on page 4–44 for register format and bits description.
0x0EC–0x0EF	RX Command Register	Receive datapath control register. See table Figure 4–19 and Table 4–18 on page 4–44 for register format and bits description.
0x100 – 0x1FC	Multicast Hash Table	64-entry hash table.
0x200 – 0x27C	MDIO Space 0 or PCS Block Configuration	MDIO registers for the first PHY device. These registers map directly to the 32 MDIO registers in a connected device. If the configuration includes the PCS function, these registers control the PCS function.
0x280 – 0x2FC	MDIO Space 1	MDIO registers for a second PHY device. These registers map directly to the 32 MDIO registers in a connected device.
0x300 – 0x31C	MAC Addresses	Supplemental unicast MAC addresses.
0x320 – 0x3FC	Reserved	Unused.

Bit 0 of a register is the least significant bit. All bits in reserved registers should be 0 and ignored on read to allow future extensions.

Complete MAC Interface Register Map

This section defines the complete register map for the control interface. Each usable register is listed and described in [Table 4–13](#). The following list describes the columns HW Reset, SW Reset and Access in the table:

- The HW Reset column specifies the value after hardware reset, which is controlled by the `reset` signal.
- The SW Reset column specifies the value or influence after a software reset, which is controlled by the `SW_RESET` bit 9 (bit 13) in the `command_config` register.
 - “_” indicates that `reset` is not relevant to this register and has no influence.
 - “X” indicates that the value is unknown. This is typical for memory-based registers.
- The Access column indicates whether you can only read a register (R0), write it (WO) or read and write it (RW).

Table 4–13. MAC Interface Register Map (Part 1 of 7)

Address Offset	Name	Description	Access	HW Reset	SW Reset
0x000	<code>rev</code>	MegaCore function revision. This register is divided into two 16-bit fields: <ul style="list-style-type: none"> ● Bits 15:0: MegaCore function revision, set to 0x0701 ● Bit 31:16: Customer specific revision, set to 0 during MegaCore function configuration. This field is controlled by the parameter <code>CUST_VERSION</code> defined in the top level generated for the Triple Speed Ethernet MegaCore function instance. 	RO	0x0000 0701	–
0x004	<code>scratch</code>	Scratch Register. Provides a memory location for the host processor to test the device memory operation.	RW	0	–
0x008	<code>command_config</code>	MAC command register. The host processor uses this register to control and configure the MAC block.	RW	0	bit0=0 bit1=0 Others not modified
0x00C	<code>mac_0</code>	MAC address 32-bit word 0, bits 0 to 31 of the MAC address. Bit 0 maps to bit 0 of the MAC address, bit 1 maps to bit 1 of the MAC address, and so on.	RW	0	–

Table 4–13. MAC Interface Register Map (Part 2 of 7)

Address Offset	Name	Description	Access	HW Reset	SW Reset
0x010	mac_1	MAC address 32-bit word 1, bits 32 to 47 of the core MAC address. Bits 16 to 31 are reserved. Bit 0 maps to bit 32 of the MAC address.	RW	0	–
0x014	frm_length	14-bit maximum frame length. The MAC receive logic uses this value to check frames. Typical value is 1518. Bits 14 to 31 are reserved.	RW	1518	–
0x018	pause_quant	16-bit receive pause quanta. The pause quanta is used in each pause frame sent to a remote Ethernet device, in increments of 512 Ethernet bit times. Bits 16 to 31 are reserved.	RW	0	–
0x01C	rx_section_empty	12-bit receive FIFO section-empty threshold. Bits 12 to 31 are unused.	RW	0	–
0x020	rx_section_full	12-bit receive FIFO section-full threshold. Bits 12 to 31 are unused.	RW	0	–
0x024	tx_section_empty	12-bit transmit FIFO section-empty threshold. Bits 12 to 31 are unused.	RW	0	–
0x028	tx_section_full	12-bit transmit FIFO section-full threshold. Bits 12 to 31 are unused.	RW	0	–
0x02c	rx_almost_empty	12-bit receive FIFO almost-empty threshold. When this register is set to 0, the MAC function never asserts the signal <code>ff_rx_a_empty</code> . This register is typically set to a value greater than or equal to 8. Bits 12 to 31 are unused.	RW	0	–
0x030	rx_almost_full	12-bit receive FIFO almost-full threshold. When this register is set to 0, the MAC function never asserts the signal <code>ff_rx_a_full</code> . This register is typically set to a value greater than or equal to 8. Bits 12 to 31 are unused.	RW	0	–
0x034	tx_almost_empty	12-bit transmit FIFO almost-empty threshold. When this register is set to 0, the MAC function never asserts the signal <code>ff_tx_a_empty</code> . This register is typically set to a value greater than or equal to 8. Bits 12 to 31 are unused.	RW	0	–

Table 4–13. MAC Interface Register Map (Part 3 of 7)

Address Offset	Name	Description	Access	HW Reset	SW Reset
0x038	tx_almost_full	12-bit transmit FIFO almost-full threshold. When this register is set to 0, the MAC function never asserts the signal <code>ff_tx_a_full</code> . This register is typically set to a value greater than or equal to 10. Bits 12 to 31 are unused.	RW	0	–
0x03C	mdio_addr0	MDIO address of PHY Device 0. Bits 0 to 4 hold a 5-bit PHY address. Bits 5 to 31 are reserved and set to read only value of 0.	RW	0	–
0x040	mdio_addr1	MDIO address of PHY Device 1. Bits 0 to 4 hold a 5-bit PHY address. Bits 5 to 31 are reserved and set to read only value of 0.	RW	1	–
0x044 to 0x054	Reserved	Reserved for user defined registers.	–	0	–
0x058	reg_status	Register read access status. This register is used to check the correct completion of register read access.	RO	0	–
0x05C	tx_ipg_length	Minimum IPG. Valid values are between 8 and 27 byte-times. If this register is set to an invalid value, it defaults to 12 byte-times which is a typical value of minimum IPG. Bits 5 to 31 are reserved and set to read-only value 0.	RW	0	–
0x060 0x064	aMacID	This register is wired to <code>mac_0</code> and <code>mac_1</code> MAC addresses respectively.	RO	0	–
0x068	aFramesTransmittedOK	See Table 4–19 on page 4–45 .	RO	0	0
0x06C	aFramesReceivedOK	See Table 4–19 on page 4–45 .	RO	0	0
0x070	aFrameCheckSequenceErrors	See Table 4–19 on page 4–45 .	RO	0	0
0x074	aAlignmentErrors	See Table 4–19 on page 4–45 .	RO	0	0
0x078	aOctetsTransmittedOK	See Table 4–19 on page 4–45 .	RO	0	0
0x07C	aOctetsReceivedOK	See Table 4–19 on page 4–45 .	RO	0	0
0x080	aTxPAUSEMACCtrlFrames	Number of transmitted pause frames. See Table 4–19 on page 4–45 .	RO	0	0

Table 4–13. MAC Interface Register Map (Part 4 of 7)

Address Offset	Name	Description	Access	HW Reset	SW Reset
0x084	aRxPAUSEMACCtrlFrames	Number of received pause frames. See Table 4–19 on page 4–45.	RO	0	0
0x088	ifInErrors	See Table 4–20 on page 4–46.	RO	0	0
0x08C	ifOutErrors	See Table 4–20 on page 4–46.	RO	0	0
0x090	ifInUcastPkts	See Table 4–20 on page 4–46.	RO	0	0
0x094	ifInMulticastPkts	See Table 4–20 on page 4–46.	RO	0	0
0x098	ifInBroadcastPkts	See Table 4–20 on page 4–46.	RO	0	0
0x09C	ifOutDiscards	See Table 4–22 on page 4–48.	RO	0	0
0x0A0	ifOutUcastPkts	See Table 4–20 on page 4–46.	RO	0	0
0x0A4	ifOutMulticastPkts	See Table 4–20 on page 4–46.	RO	0	0
0x0A8	ifOutBroadcastPkts	See Table 4–20 on page 4–46.	RO	0	0
0x0AC	etherStatsDropEvents	See Table 4–21 on page 4–47.	RO	0	0
0x0B0	etherStatsOctets	See Table 4–21 on page 4–47.	RO	0	0
0x0B4	etherStatsPkts	See Table 4–21 on page 4–47.	RO	0	0
0x0B8	etherStatsUnderSizePkts	See Table 4–21 on page 4–47.	RO	0	0
0x0BC	etherStatsOversizePkts	See Table 4–21 on page 4–47.	RO	0	0
0x0C0	etherStatsPkts64Octets	See Table 4–21 on page 4–47.	RO	0	0
0x0C4	etherStatsPkts65to127Octets	See Table 4–21 on page 4–47.	RO	0	0
0x0C8	etherStatsPkts128to255Octets	See Table 4–21 on page 4–47.	RO	0	0
0x0CC	etherStatsPkts256to511Octets	See Table 4–21 on page 4–47.	RO	0	0

Table 4–13. MAC Interface Register Map (Part 5 of 7)

Address Offset	Name	Description	Access	HW Reset	SW Reset
0x0D0	etherStatsPkts512to1023Octets	See Table 4–21 on page 4–47.	RO	0	0
0x0D4	etherStatsPkts1024to1518Octets	See Table 4–21 on page 4–47.	RO	0	0
0x0D8	etherStatsPkts1519toXOctets	Any frame length from 1519 to the maximum length configured in the <code>frm_length</code> register, if it is greater than 1518.	RO	0	0
0x0DC	etherStatsJabbers	Too long frames with CRC error.	RO	0	0
0x0E0	etherStatsFragments	Too short frames with CRC error.	RO	0	0
0x0E4	Reserved	Unused	RO	–	–
0x0E8	tx_cmd_stat	Transmit FIFO control register.	RW	0x00040000	-
0x0EC	rx_cmd_stat	Receive FIFO control register.	RW	0x02000000	-
0x100 – 0x1FC	hash_table	<p>Multicast address resolution table, mapped in the controller address space. When programming the table, only bit 0 is significant.</p> <p>If a 1 is written to an address offset in the hash table, all multicast MAC addresses that hash to the value of address (bits 0 to 5) are accepted by the MAC. If a 0 is written, matching multicast addresses are rejected.</p>	WO	0	–

Table 4–13. MAC Interface Register Map (Part 6 of 7)

Address Offset	Name	Description	Access	HW Reset	SW Reset
0x200 – 0x27C	PHY Device 0 Internal Registers	<p>Registers 0 to 31 within PHY device 0 connected to the MDIO PHY management interface. Reading or writing immediately causes a corresponding MDIO transaction to read or write the underlying PHY device register. For configurations that include MAC and PCS blocks, the internal PCS is always device 0. In this case, reading and writing does not require an MDIO module as the application reads/writes directly to the PHY registers through the register interface.</p> <p>The register at address offset 0x200 corresponds to register 0 of PHY device 0. The register at address offset 0x204 corresponds to register 1 of PHY device 0.</p> <p>For all registers, bits 15..0 are significant. Bits 31..16 should be written with 0 and ignored on read.</p>	RW	–	–
0x280 – 0x2FC	PHY Device 1 Internal Registers	<p>Registers 0 to 31 within PHY device 1 connected to the MDIO PHY management interface. Reading or writing immediately causes a corresponding MDIO transaction to read or write the underlying PHY device register.</p> <p>The register at address offset 0x280 corresponds to register 0 of PHY device 1. The register at address offset 0x284 corresponds to register 1 of PHY device 1.</p> <p>For all registers, bits 15..0 are significant. Bits 31..16 should be written with 0 and ignored on read.</p>	RW	–	–
0x300	smac_0_0	Supplemental MAC Address 0, bits 0 to 31. Register bit 0 maps to bit 0 of the MAC address, bit 1 maps to bit 1 of the MAC address, and so on.	RW	0	–
0x304	smac_0_1	Supplemental MAC Address 0, bits 32 to 47. Register bit 0 maps to bit 32 of the MAC address. Register bits 16 to 31 are reserved.	RW	0	–

Table 4–13. MAC Interface Register Map (Part 7 of 7)

Address Offset	Name	Description	Access	HW Reset	SW Reset
0x308	smac_1_0	Supplemental MAC Address 1, bits 0 to 31. Register bit 0 maps to bit 0 of the MAC address, bit 1 maps to bit 1 of the MAC address, and so on.	RW	0	–
0x30C	smac_1_1	Supplemental MAC Address 1, bits 32 to 47. Register bit 0 maps to bit 32 of the MAC address. Register bits 16 to 31 are reserved.	RW	0	–
0x310	smac_2_0	Supplemental MAC Address 2, bits 0 to 31. Register bit 0 maps to bit 0 of the MAC address, bit 1 maps to bit 1 of the MAC address, and so on.	RW	0	–
0x314	smac_2_1	Supplemental MAC Address 2, bits 32 to 47. Register bit 0 maps to bit 32 of the MAC address. Register bits 16 to 31 are reserved.	RW	0	–
0x318	smac_3_0	Supplemental MAC Address 3, bits 0 to 31. Register bit 0 maps to bit 0 of the MAC address, bit 1 maps to bit 1 of the MAC address, and so on.	RW	0	–
0x31C	smac_3_1	Supplemental MAC Address 3, bits 32 to 47. Register bit 0 maps to bit 32 of the MAC address. Register bits 16 to 31 are reserved.	RW	0	–
0x320 – 0x3FC	Reserved	–	–	–	–

Command_Config Register

Table 4–14 shows the bits and fields that comprise the `command_config` register.

Table 4–14. Command_Config Register

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
CNT_RESET	Reserved				RX_ERR_DISC	ENA_10	NO_LGTH_CHECK	CNTL_FRM_ENA	XOFF_GEN	WAKEUP	SLEEP	MAGIC_ENA	TX_ADDR_SEL			LOOP_ENA	MHASH_SEL	SW_RESET	LATE_COL	EXCESS_COL	HD_ENA	TX_ADDR_INS	PAUSE_IGNORE	PAUSE_FWD	CRC_FWD	PAD_EN	PROMIS_EN	ETH_SPEED	XON_GEN	RX_ENA	TX_ENA

Table 4–15 describes the function of each bit and field in the `command_config` register.

Table 4–15. Command_Config Register Bit Descriptions (Part 1 of 4)

Bit(s)	Bit Name	Access ⁽¹⁾	Description
0	TX_ENA	RW	Transmit enable. Setting this bit to 1 enables the transmit datapath. This bit is cleared following a hardware or software reset. See the <code>SW_RESET</code> bit description.
1	RX_ENA	RW	Receive enable. Setting this bit to 1 enables the receive datapath. This bit is cleared following a hardware or software reset. See the <code>SW_RESET</code> bit description.
2	XON_GEN	RW	Pause frames generation. When this bit is set to 1, the MAC function generates a pause frame with a pause quanta of 0, independent of the receive FIFO status.
3	ETH_SPEED	RW	Ethernet speed control. <ul style="list-style-type: none"> Setting this bit to 1 enables gigabit Ethernet operation. The signal <code>set_1000</code> is masked and does not affect the operation. If this bit is set to 0, gigabit Ethernet operation is enabled only if the signal <code>set_1000</code> is asserted. Otherwise, the MAC function operates in 10/100 Mbps Ethernet mode. When the MAC operates in gigabit mode, the output signal <code>eth_mode</code> is asserted.
4	PROMIS_EN	RW	Promiscuous enable. Setting this bit to 1 enables the MAC promiscuous operation. All frames are received without unicast address filtering.

Table 4–15. Command Config Register Bit Descriptions (Part 2 of 4)

Bit(s)	Bit Name	Access ⁽¹⁾	Description
5	PAD_EN	RW	Pad enable. Transmit frames are always padded. Setting this bit to 1 enables pad removal. The MAC function removes receive frames padding before forwarding the frames to the user application.
6	CRC_FWD	RW	CRC forwarding. <ul style="list-style-type: none"> • If this bit is set to 1, the MAC function forwards the CRC field to the user application. • If this bit is set to 0, the MAC function removes the CRC field from the frame before forwarding the frame to the user application. • This bit is ignored if the PAD_EN bit is 1. In this case, the MAC function checks the CRC field and removes it from the frame before forwarding the frame to the user application.
7	PAUSE_FWD	RW	Pause frame forwarding. Terminates or forwards pause frames. <ul style="list-style-type: none"> • If this bit is set to 1, the MAC function forwards pause frames to the user application. • If this bit is set to 0, the MAC function terminates and discards pause frames.
8	PAUSE_IGNORE	RW	Ignore pause frame quanta. <ul style="list-style-type: none"> • Setting this bit to 1 causes the MAC function to ignore received pause frames. • Setting this bit to 0 causes the transmit process to stop for an amount of time specified in the pause quanta within the pause frame.
9	TX_ADDR_INS	RW	Set MAC address on transmit. <ul style="list-style-type: none"> • If this bit is set to 1, the MAC function overwrites the source MAC address with the MAC address configured in the mac_0 and mac_1 registers, or in any of the supplemental MAC address registers. • If this bit is set to 0, the MAC function does not modify the source MAC address.
10	HD_ENA	RW	Enable half-duplex mode. <ul style="list-style-type: none"> • Setting this bit to 1 enables the half-duplex mode. • Setting this bit to 0 enables the full-duplex mode. • This bit is ignored if the MAC function is operating in gigabit Ethernet mode (ETH_SPEED bit is 1).
11	EXCESS_COL	RC	Excessive collision condition. <ul style="list-style-type: none"> • This bit is set to 1 when the MAC function discards a frame after detecting a collision on 16 consecutive packet retransmissions. • This bit is cleared following a hardware or software reset. See the SW_RESET bit description.

Table 4–15. Command_Config Register Bit Descriptions (Part 3 of 4)

Bit(s)	Bit Name	Access ⁽¹⁾	Description
12	LATE_COL	RC	Late collision condition. <ul style="list-style-type: none"> This bit is set to 1 when the MAC function detects a collision after 64 bytes are transmitted, and discards the frame. This bit is cleared following a hardware or software reset. See the SW_RESET bit description.
13	SW_RESET	RW	Software reset command. Setting this bit to 1 causes the MAC function to disable the transmit and receive logic, flush the receive FIFO, and reset the statistics counters. This bit is automatically cleared when the software reset sequence completes.
14	MHASH_SEL	RW	Select multicast address-resolution hash-code mode. <ul style="list-style-type: none"> If this bit is set to 0, the hash code is generated from the full 48-bit destination address. If this bit is set to 1, the hash code is generated from the lower 24-bit of the destination MAC address.
15	LOOP_ENA	RW	GMII and MII interfaces loopback enable. Setting this bit to 1 enables a loopback. Frames sent through the transmit interface are looped back into the receive interface.
16 – 18	TX_ADDR_SEL(2:0)	RW	Source MAC address selection on transmit. If the command_config register bit TX_ADDR_INS is 1, the value of these bits determines which address the MAC function selects to overwrite the source MAC address. <ul style="list-style-type: none"> 000: The node MAC Address configured in the mac_0 and mac_1 registers is selected. 100: The supplemental MAC Address 0 configured in the smac_0_0 and smac_0_1 registers is selected. 101: The supplemental MAC Address 1 configured in the smac_1_0 and smac_1_1 registers is selected. 110: The supplemental MAC Address 2 configured in the smac_2_0 and smac_2_1 registers is selected. 111: The supplemental MAC Address 3 configured in the smac_3_0 and smac_3_1 registers is selected.
19	MAGIC_ENA	RW	Enable magic packet detection or Wake-on-LAN. Setting this bit to 1 enables magic packet detection.
20	SLEEP	RW	Enable sleep mode. When the MAGIC_ENA bit is 1, setting this bit to 1 puts the MAC function into sleep mode and enables magic packet detection.
21	WAKEUP	RC	Node wake up request indication. Read-only status bit. Valid only when the MAGIC_ENA bit is 1. <ul style="list-style-type: none"> This bit is set to 1 when a magic packet is detected. This bit is cleared when the SLEEP bit is set to 0.
22	XOFF_GEN	RW	Pause Frame Generation. If this bit is set to 1, the MAC function generates a pause frame with the pause quanta set to the value configured in the pause_quant register, independent of the receive FIFO status.

Table 4–15. Command Config Register Bit Descriptions (Part 4 of 4)

Bit(s)	Bit Name	Access ⁽¹⁾	Description
23	CNTL_FRM_ENA	RW	MAC control frame enable. <ul style="list-style-type: none"> • If this bit is set to 1, MAC control frames with any opcode other than 0x0001 are accepted and forwarded to the Avalon-ST interface. • If this bit is set to 0, MAC control frames with any opcode other than 0x0001 are discarded.
24	NO_LGTH_CHECK	RW	Payload length check disable. <ul style="list-style-type: none"> • If this bit is set to 0, the MAC function checks the actual payload length of received frames against the length/type field in the received frames. • No checking is done if this bit is set to 1.
25	ENA_10	RW	10 Mbps interface enable. <ul style="list-style-type: none"> • Setting this bit to 1 enables the 10Mbps interface, and the output signal <code>ena_10</code> is asserted. • If this bit is set to 0, the output signal <code>ena_10</code> is asserted only when the input signal <code>set_10</code> is asserted.
26	RX_ERR_DISC	RW	Receive erroneous frame discard enable. <ul style="list-style-type: none"> • If this bit is set to 1, the MAC function discards erroneous frames received. Set this bit to 1 only if store and forward operation is enabled on the receive FIFO by setting the <code>rx_section_full</code> register to 0. • If this bit is set to 0, the MAC function forwards erroneous frames to the user application with <code>rx_err(0)</code> asserted.
27 – 30	Reserved	–	
31	CNT_RESET	WC	Self-clearing counter reset command. Setting this bit to 1 clears the statistics counters. This bit is automatically cleared when the counter reset sequence is completed.

Note:

(1) RW = Read/Write, RC = Read/Clear, WC = Write/Clear

Reg_Status Register

Figure 4–17 shows the bits and fields that comprise the `reg_status` register.

Figure 4–17. Reg_Status Register

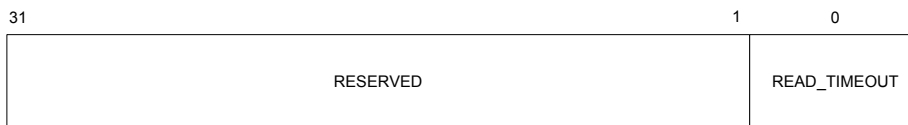


Table 4–16 provides the bit description.

Table 4–16. Reg_Status Register Bit Description

Bit	Bit Name	Access ⁽¹⁾	Description
0	READ_TIMEOUT	RC	<p>Read access timeout indication. A value of 1 indicates that the read access was terminated with a timeout. A value of 0 indicates that the read access was terminated successfully. Under normal operation, It takes about 11 clock cycles from register set to register clear. The READ_TIMEOUT condition might occur if the MAC function loses the <code>rx_clk</code> when reading the statistic counters.</p> <p>Bit resets to 0 when the <code>reg_status</code> register is read or after reset.</p>

Note:

(1) RC = Read/Clear

Tx_Cmd_Stat Register

Figure 4–19 shows the bits and fields that comprise the `tx_cmd_stat` register.

Figure 4–18. Tx_Cmd_Stat Register

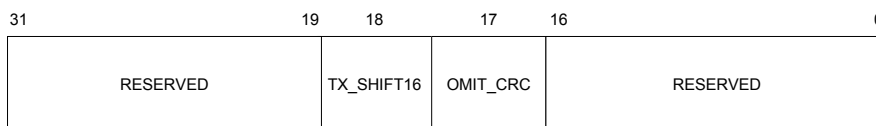


Table 4–18 provides the bit description.

Table 4–17. Tx_Cmd_Stat Register Bit Description			
Bit	Name	Access(1)	Description
17	OMIT_CRC	RW	If this bit is set to 1, the MAC function does not calculate and append the CRC to the frame. In this case, the application is responsible for providing correct frame data and CRC.
18	TX_SHIFT16	RW	If this bit is set to 1, the MAC function expects 32 bits word aligned frames. The MAC function removes the first two bytes from the frame before transmitting it.

Note:

(1) RW = Read/Write

Rx_Cmd_Stat Register

Figure 4–19 shows the bits and fields that comprise the rx_cmd_stat register.

Figure 4–19. Rx_Cmd_Stat Register

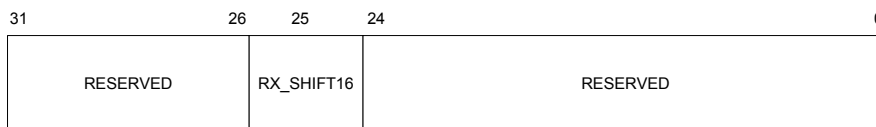


Table 4–18 provides the bit description.

Table 4–18. Rx_Cmd_Stat Register Bit Description			
Bit	Name	Access(1)	Description
25	RX_SHIFT16	RW	If this bit is set to 1, the MAC function shifts the beginning of the packet to the right by 2 bytes and inserts zeros in the empty bytes to word align the packet

Note:

(1) RW = Read/Write

MAC SNMP MIB Statistics Registers

The Simple Network Management Program Management Information Base (SNMP MIB) block accumulates statistics required in IEEE802.3 basic, mandatory and recommended Management Information Packages, IEEE 802.3ah, Clause 30.

In addition, the MAC function provides all signals to generate the applicable objects of the Management Information Base (MIB, MIB-II) according to IETF RFC2665 and Remote Network Monitoring (RMON) according to IETF RFC 2819 for SNMP managed environments.

IEEE 802.3 Management Packages

Table 4–20 lists the resources available to implement the IEEE 802.3 mandatory management packages defined in the Ethernet Standard 802.3 Clause 30 for the managed objects *oMacEntity* and *oPauseEntity*. All objects and attributes not mentioned in the following tables are not applicable to the MAC function and are derived from the user application or higher layers instead.



For Information About	Refer To
Attributes and objects	IEEE 802.3 Standard Clause 30

Table 4–19. IEEE 802.3 MAC *oEntity* and *oPauseEntity* Managed Object Support (Part 1 of 2)

IEEE802.3 Attribute	IEEE Management Packages			Description
	Basic	Mandatory	Recommended (optional)	
oEntity Managed Object Support				
aMACID	X			The MAC addresses.
aFramesTransmittedOK		X		Number of frames transmitted without error including pause frames.
aFramesReceivedOK		X		Number of frames received without error including pause frames.
aFrameCheckSequence Errors		X		Number of frames received with a CRC error.
aAlignmentErrors		X		Frame received with an alignment error.
aOctetsTransmittedOK			X	Sum of payload and padding octets of frames transmitted without error.

Table 4–19. IEEE 802.3 MAC oEntity and oPauseEntity Managed Object Support (Part 2 of 2)

IEEE802.3 Attribute	IEEE Management Packages			Description
	Basic	Mandatory	Recommended (optional)	
aOctetsReceivedOK			X	Sum of payload and padding octets of frames received without error.
oPauseEntity Managed Object Support				
aPAUSEMACCtrlFramesTransmitted			X	Number of transmitted pause frames.
aPAUSEMACCtrlFramesReceived			X	Number of Received pause frames.

IETF Management Information Base – (MIB, MIB-II) Objects Support

The Internet Engineering Task Force (IETF) Request for Comments 2665 (RFC2665) defines the Management Information Base (MIB, MIB-II) objects for the Ethernet-like Interface Types. RFC2665 details the MIB (MIB-II) objects for Ethernet interfaces, which are defined in a more generic manner within RFC 2863.

Table 4–20. IETF MIB (MIB-II) Objects Support

MIB Object Name	Description
ifInUcastPkts	Number of valid received unicast frames.
ifInMulticastPkts	Number of valid received multicast frames (without pause).
ifInBroadcastPkts	Number of valid received broadcast frames.
ifOutUcastPkts	Number of valid transmitted unicast frames.
ifOutMulticastPkts	Number of valid transmitted multicast frames.
ifOutBroadcastPkts	Number of valid transmitted broadcast frames.
ifInErrors	Number of frames received with error: <ul style="list-style-type: none"> • FIFO overflow error • CRC error • Length error • Alignment error
ifOutErrors	Number of frames transmitted with error: <ul style="list-style-type: none"> • FIFO overflow error • FIFO underflow error • User application defined error

IETF Remote Network Monitoring Support

The IETF RFC 2819 defines objects for managing remote network monitoring devices. These objects are usually implemented in a dedicated device (monitor/probe) for traffic monitoring and analysis within a network segment. Such a probe usually samples the values in a periodic manner to give relative usage estimations rather than absolute values. Table 4–21 lists the defined objects. The Remote Monitoring (RMON) MIB counts good and bad packets, defined as:

- Good packets (valid frames): Good packets are error-free packets that have a valid frame length. A valid frame length is defined as between 64 octets long and the value set in the `frm_length` register. The length does not include framing bits (preamble, SFD) but includes the FCS field.
- Bad packets (invalid frames): Bad packets are packets that have proper framing and are therefore recognized as packets, but contain errors within the packet or have an invalid length. On the Ethernet, bad packets have a valid preamble and SFD, but have a bad CRC, or are either shorter than 64 octets or longer than and the value set in the `frm_length` register.

Table 4–21. IETF RMON MIB Object Support

MIB Object Name	MAC Core Support
<code>etherStatsDropEvents</code>	Counts the number of dropped packets due to internal errors of the MAC client. Occurs when FIFO overflow condition persists.
<code>etherStatsOctets</code>	Total number of octets received. Good and bad frames.
<code>etherStatsPkts</code>	Total number of packets received. Counts good and bad packets.
<code>etherStatsUndersizePkts</code>	Number of packets received with less than 64 octets.
<code>etherStatsOversizePkts</code>	Incremented with each well-formed packet that exceeds the valid maximum programmed frame length.
<code>etherStatsPkts64Octets</code>	Incremented when a packet of 64 octets length is received (good and bad frames are counted)
<code>etherStatsPkts65to127Octets</code>	Frames (good and bad) with 65 to 127 octets
<code>etherStatsPkts128to255Octets</code>	Frames (good and bad) with 128 to 255 octets
<code>etherStatsPkts256to511Octets</code>	Frames (good and bad) with 256 to 511 octets
<code>etherStatsPkts512to1023Octets</code>	Frames (good and bad) with 512 to 1023 octets
<code>etherStatsPkts1024to1518Octets</code>	Frames (good and bad) with 1024 to 1518 octets

Software Derived MIB Objects

To extend the management information base, the following counters and objects can be derived by the management or driver software, based on the counters available as indicated in the above tables.

Table 4–22 describes three derived IETF MIB (MIB-II) objects.

Table 4–22. Derived IETF MIB (MIB-II) Objects	
MIB Object	Description
ifInOctets	Sum of octets received except preamble (for example, header, payload, pad and FCS) of all valid received frames. = $18 \times \text{aFramesReceivedOK} + \text{aOctetsReceivedOK}$
ifOutOctets	Sum of octets transmitted except preamble (for example, header, payload, pad and FCS) of all valid transmitted frames. = $18 \times \text{aFramesTransmittedOK} + \text{aOctetsTransmittedOK}$
ifOutDiscards	Not applicable to the MAC, as the MAC does not discard frames which were written into the FIFO by the user application. If a higher layer discards frames to be transmitted it will implement this counter.

Table 4–23 describes three derived IETF RMON MIB objects.

Table 4–23. Derived IETF RMON MIB Objects	
Object	MAC Core Support
etherStatsBroadcastPkts	Any valid frame with Broadcast address: = ifInBroadcastPkts
etherStatsMulticastPkts	Any valid multicast frame, including pause frames: = ifInMulticastPkts + aPAUSEMACCtrlFramesReceived
etherStatsCRCAlignErrors	Incremented when frames of correct length but with CRC error are received: = aFrameCheckSequenceErrors

MII, GMII and RGMII Interfaces

The following three Ethernet interfaces are implemented:

- Fast Ethernet MII (Media Independent Interface)
- Gigabit Ethernet GMII (Gigabit Media Independent Interface)
- Gigabit Ethernet RGMII (Reduced Gigabit Media Independent Interface)

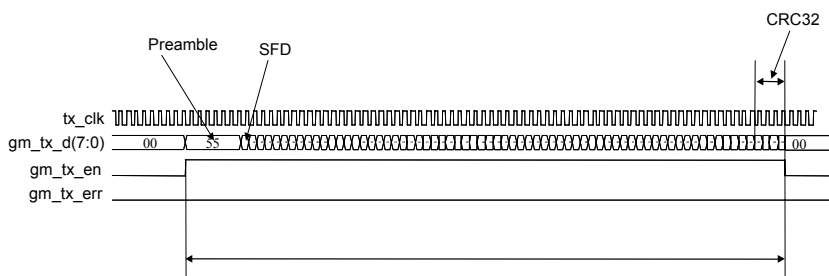
To enable the GMII interface, the host processor sets the `ETH_SPEED` bit in the `command_config` register to 1. Setting the `ETH_SPEED` bit to 0 enables the MII interface.

GMII Interface

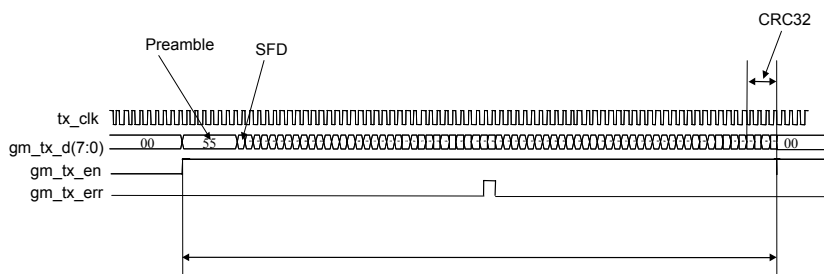
Transmit

On transmit, all data transfers are synchronous to the rising edge of `tx_clk`. The GMII data enable signal `gm_tx_en` is asserted to indicate the start of a new frame and remains asserted until the last byte of the frame is present on `gm_tx_d(7:0)` bus. Between frames, `gm_tx_en` remains de-asserted. Figure 4–20 shows this operation.

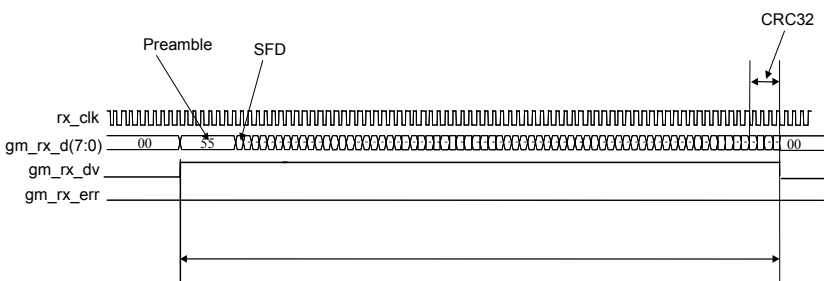
Figure 4–20. GMII Transmit Operation



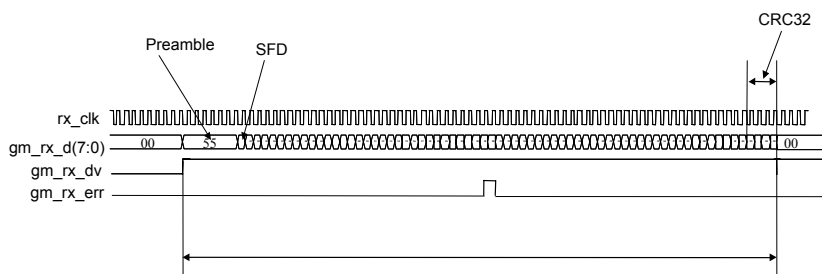
If a frame is received on the FIFO interface with an error (asserted with `ff_tx_eop`), the frame is subsequently transmitted with the GMII `gm_tx_err` error signal at any time during the packet transfer, as shown in Figure 4–21.

Figure 4–21. GMII Transmit Operation - Erroneous Frame**Receive**

On receive all signals are sampled on the rising edge of `rx_clk`. The GMII data enable signal `gm_rx_dv` is asserted by the PHY to indicate the start of a new frame and remains asserted until the last byte of the frame is present on `gm_rx_d(7:0)` bus. Between frames, `gm_rx_dv` remains de-asserted. Figure 4–22 shows this operation

Figure 4–22. GMII Receive Operation

If the PHY detects an error on the frame received from the line, the PHY asserts the GMII error signal, `gm_rx_err`, for at least one clock cycle at any time during the packet transfer, as shown in Figure 4–23.

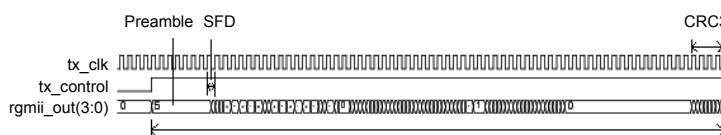
Figure 4–23. GMII Receive Operation - Erroneous Frame

A frame received on the GMII interface with a PHY error indication is subsequently transferred on the FIFO interface with the error signal `rx_err(0)` asserted.

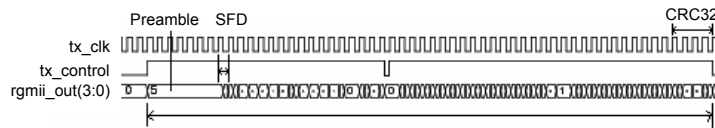
RGMII Interface

Transmit

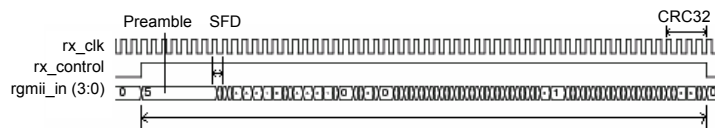
On transmit, all data transfers are synchronous to the both edges of `tx_clk`. The RGMII control signal `tx_control` is asserted to indicate the start of a new frame and remains asserted until the last upper nibble of the frame is present on the `rgmii_out(3:0)` bus. Between frames, `tx_control` remains de-asserted. Figure 4–24 shows this operation.

Figure 4–24. RGMII Transmit Operation

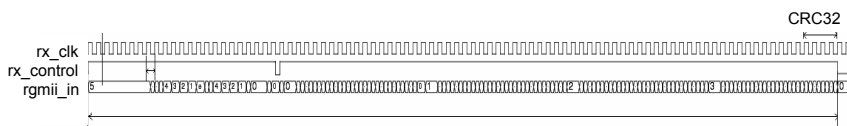
If a frame is received on the FIFO interface with an error (`ff_tx_err` asserted with `ff_tx_eop`), the frame is subsequently transmitted with the RGMII `tx_control` error signal (at the falling edge of `tx_clk`) at any time during the packet transfer, as shown in Figure 4–25.

Figure 4–25. RGMII Transmit Operation - Erroneous Frame**Receive**

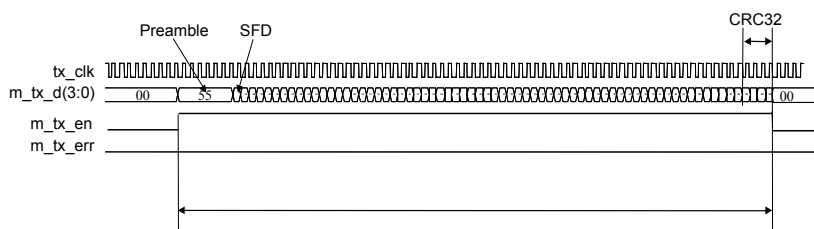
On receive all signals are sampled on the both edges of `rx_clk`. The RGMII control signal `rx_control` is asserted by the PHY to indicate the start of a new frame and remains asserted until the last upper nibble of the frame is present on `rgmii_in(3:0)` bus. Between frames, `rx_control` remains de-asserted. Figure 4–26 shows this operation.

Figure 4–26. RGMII Receive Operation

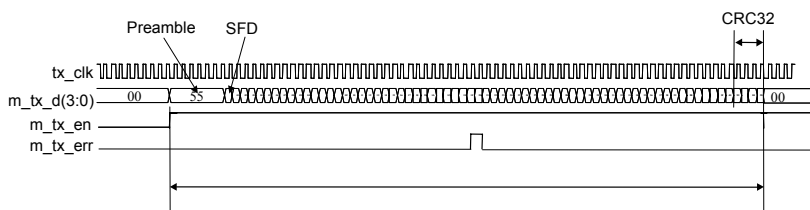
A frame received on the RGMII interface with a PHY error indication is subsequently transferred on the FIFO interface with the error signal `rx_err(0)` asserted, as shown in Figure 4–27.

Figure 4–27. RGMII Receive Operation - Erroneous Frame**MII Interface****Transmit**

On transmit, all data transfers are synchronous to the rising edge of `tx_clk`. The MII data enable signal, `m_tx_en`, is asserted to indicate the start of a new frame and remains asserted until the last byte of the frame is present on `m_tx_d(3:0)` bus. Between frames, `m_tx_en` remains de-asserted. Figure 4–28 shows this operation.

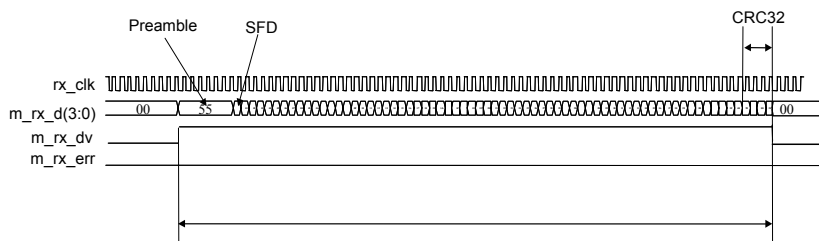
Figure 4–28. MII Transmit Operation

If a frame is received on the FIFO interface with an error (`ff_tx_err` asserted) the frame is subsequently transmitted with the MII `m_tx_err` error signal for one clock cycle at any time during the packet transfer, as shown in Figure 4–29.

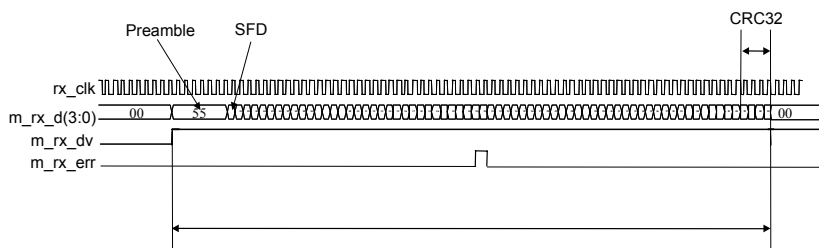
Figure 4–29. MII Transmit Operation - Erroneous Frame

Receive

On receive all signals are sampled on the rising edge of `rx_clk`. The MII data enable signal `m_rx_en` is asserted by the PHY to indicate the start of a new frame and remains asserted until the last byte of the frame is present on `m_rx_d(3:0)` bus. Between frames, `m_rx_en` remains deasserted. Figure 4–30 shows this operation.

Figure 4–30. MII Receive Operation

If the PHY detects an error on the frame received from the line, the PHY asserts the MII error signal, `m_rx_err`, for at least one clock cycle at any time during the packet transfer, as shown in Figure 4–31.

Figure 4–31. MII Receive Operation - Erroneous Frame

A frame received on the MII interface with a PHY error indication is subsequently transferred on the FIFO interface with the error signal `rx_err(0)` asserted.

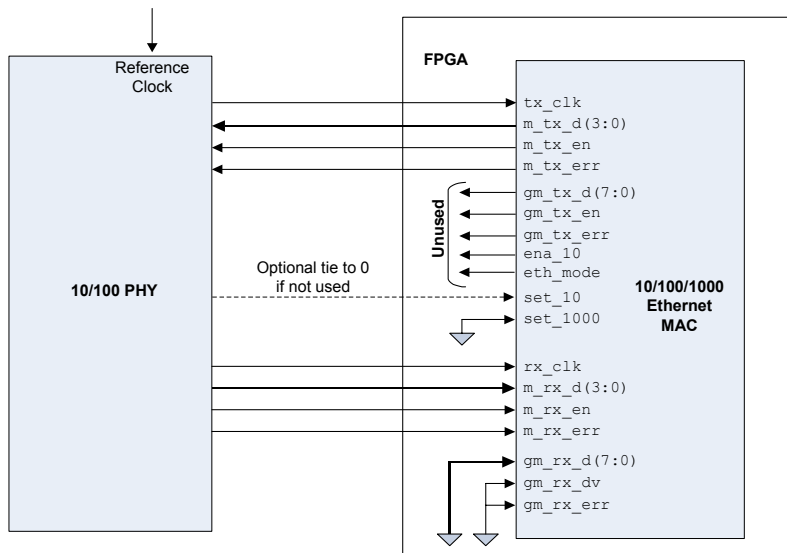
Connecting MAC to External PHY

The MAC function implements a flexible network interface, which supports standard MII for 10/100 and RGMII/GMII for gigabit interfaces. The interface can be used in multiple applications. This section provides implementation guidelines for the following three typical network applications:

- Gigabit Ethernet only operation
- Programmable 10/100 Ethernet operation
- Programmable 10/100/1000 Ethernet operation

10/100 Ethernet PHYs are connected to the MAC function with a MII interface. On the receive path, the PHY device provides a 25 MHz (100 Mbps Ethernet) or a 2.5 MHz (10 Mbps Ethernet) clock which should be connected to the MAC function clock `rx_clk`. On the transmit path, the PHY device provides a 25 MHz (100 Mbps Ethernet) or a 2.5 MHz (10 Mbps Ethernet) clock which should be connected to the MAC function clock `tx_clk`. The MAC MII transmit and receive interfaces are connected to PHY interface, the MAC function GMII interface is not used. This connectivity is illustrated in [Figure 4-32](#).

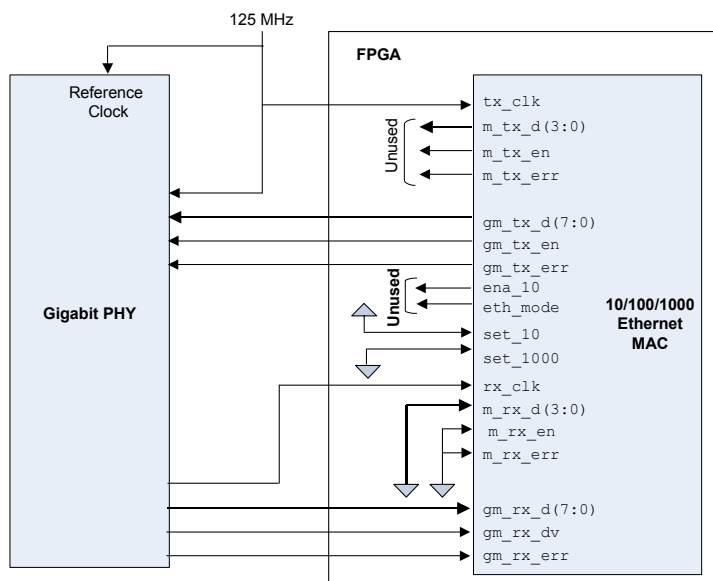
Figure 4-32. 10/100 PHY Interface



Gigabit Ethernet

Gigabit Ethernet PHYs are connected to the MAC function with a GMII interface. On the receive path, the PHY device provides a 125 MHz clock which should be connected to the core clock `rx_clk`. On transmit, a 125 MHz clock should be driven to the PHY GMII interface. The MAC function expects a 125 MHz clock on the transmit clock `tx_clk`.

A clock synchronous scheme can be implemented as shown in [Figure 4-33](#). A technology specific clock driver is required to generate a clock centered with the GMII data from the MAC. The clock driver can be a PLL, a delay line or a DDR flip-flop. The MAC GMII transmit and receive interfaces are connected to the PHY interface, the MAC MII interface is not used.

Figure 4–33. Gigabit PHY Interface

Programmable 10/100/1000 Operation

Typically, triple speed 10/100/1000 Ethernet PHY devices implement a shared interface that can be configured to implement a MII (10/100 Mbps operation) or a GMII interface (gigabit operation).

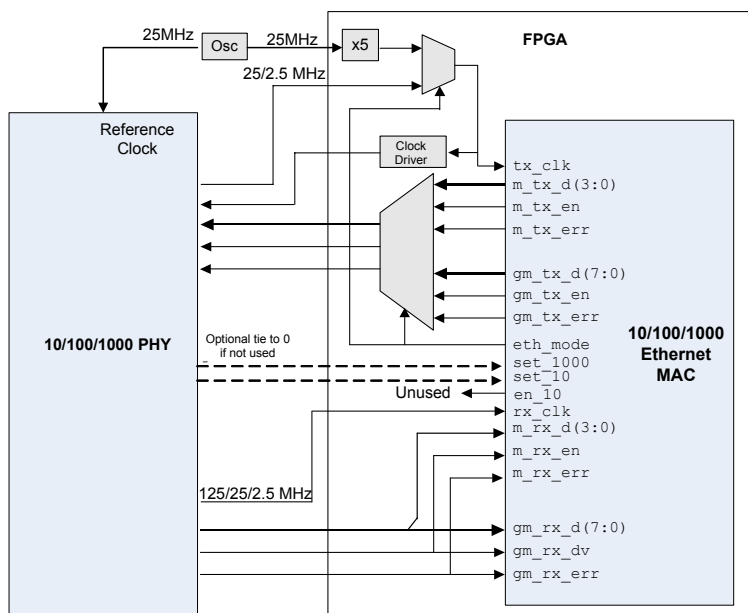
On the receive path, the clock provided by the PHY device (2.5 MHz, 25 MHz or 125 MHz) is connected to the MAC function clock `rx_clk`. The PHY interface is connected to both the MAC MII (active PHY signals) and GMII interfaces.

On the transmit interface, standard programmable PHY devices, when configured to operate in 10/100 mode, generate a 2.5 MHz (10 Mbps) or a 25 MHz (100 Mbps) clock. When configured to operate in gigabit mode, the PHY devices expect a 125 MHz clock from the MAC Layer.

In transmit mode, either the MAC MII and GMII interface is selected by the the MAC control signal `eth_mode`. The signal `eth_mode` is set to 1 when the MAC function is configured to operate in gigabit mode and is set to zero when the MAC function is configured to operate in 10/100 mode. When `eth_mode` is set to one, the MAC GMII interface should be driven to the PHY interface and when `eth_mode` is set to zero, the MAC MII interface should be driven to the PHY interface.

When configured to operate in 10/100 Mbps, the MAC transmit path should be synchronized to the 2.5/25 MHz from the PHY. When configured to operate in gigabit mode, the MAC transmit path should be synchronized with a 125 MHz clock derived from the PHY reference clock. In 10/100 mode, the clock generated by the MAC to the PHY can be tri-stated.

Figure 4–34. 10/100/1000 PHY Interface



1000BASE-X/SGMII PCS with Optional PMA

The 1000BASE-X/SGMII PCS function is accessible via GMII in the case of 1000BASE-X or SGMII, or via MII in SGMII mode only. The PCS function interfaces to an on- or off-chip SERDES component via the industry standard Ten-Bit Interface (TBI).

The PCS function can be configured with Physical Medium Attachment (PMA), the embedded serial transceivers in Altera devices. This configuration complies with the IEEE 1000BASE-X PMA specification. PMA interoperates with an external Physical Medium Dependent (PMD) device, which drives the external copper or optical network. The interconnect between Altera and PMD devices can be TBI or 1.25 Gbps serial.

Figure 4–35 shows a block diagram of the PCS function without PMA.

Figure 4–35. 1000BASE-X/SGMII PCS

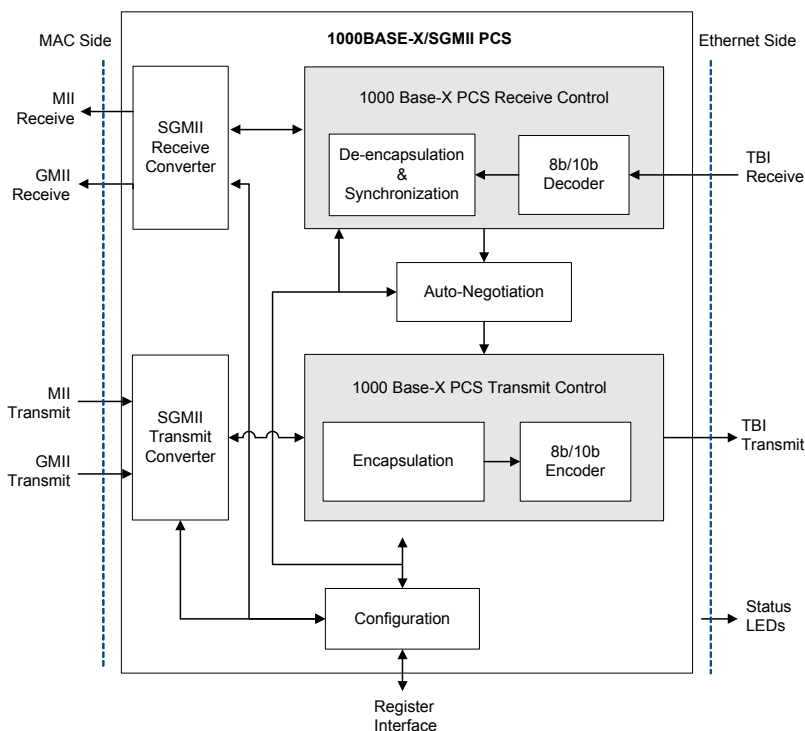
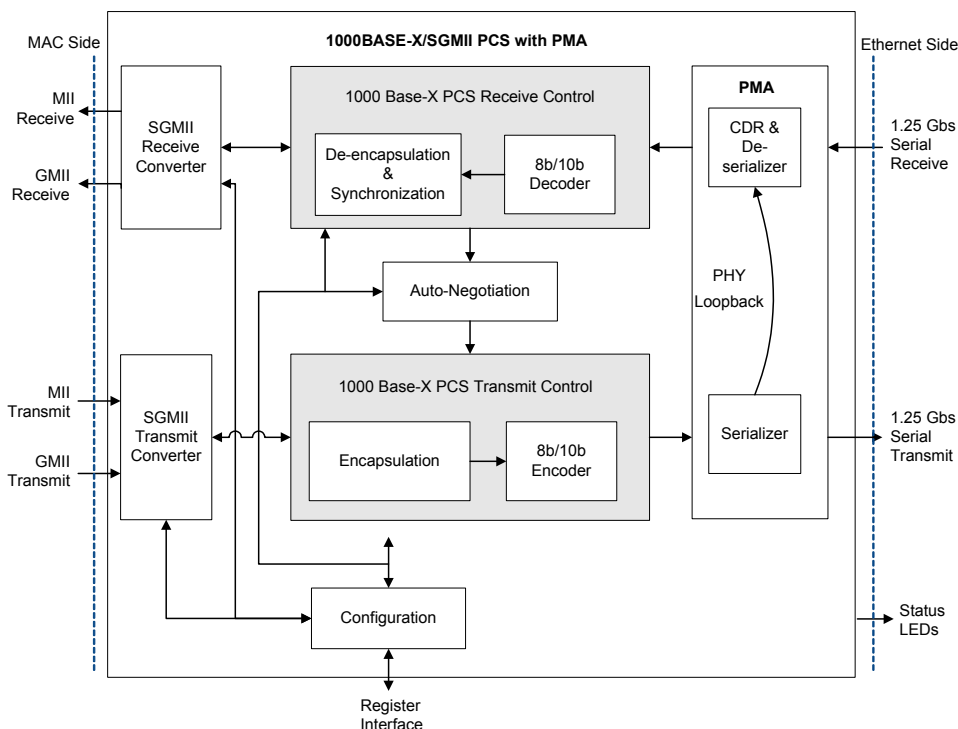


Figure 4–28 shows a block diagram of the PCS function with PMA.

Figure 4–36. 1000BASE-X/SGMII PCS with PMA



PCS Receive

This section describes the PCS receive operation, which includes comma detection, decoding, de-encapsulation, synchronization and carrier sense.

Comma Detection

The 10-bit data received from the PMA device may not be aligned on a valid 10-bit character. The comma detection function searches for the 10-bit encoded comma character K28.5 in consecutive samples received from the PMA. When the K28.5 comma code group is detected, the stream is re-aligned on a valid 10-bit character boundary. The aligned stream can subsequently be decoded with a standard 8b/10b decoder.

The comma detection logic restarts the search for a valid comma character if the receive synchronization state machine loses the link synchronization.

8b/10b Decoding

The 8b/10b decoder checks the DC balancing (disparity check) and produces a decoded 8-bit stream of data for the frame de-encapsulation function.

Frame De-encapsulation

The frame de-encapsulation state machine detects the start of frame when the /I/ /S/ sequence is received. The /S/ is subsequently replaced with a 0x55 preamble byte. The frame bytes are decoded and transmitted to the MAC function. The /T/ /R/ /R/ or the /T/ /R/ sequence is decoded as an end of packet indication for the MAC function.

The reception of a /V/ character is decoded as frame error indication for the MAC. A wrong carrier is decoded when a sequence different from /I/ /I/ (Idle) or /I/ /S/ (Start of Frame) is detected.

During frame reception, the de-encapsulation state machine checks for invalid characters. If invalid characters are detected, the de-encapsulation state machine indicates an error to the MAC function.

Synchronization

The link synchronization constantly monitors the decoded data stream and determines if the underlying receive channel is ready for operation. The link synchronization state machine acquires link synchronization if three /I/s (K28.5/Valid Data sequence) are received consecutively without error.

Once link synchronization is acquired, the link synchronization state machine counts the number of invalid characters received. The state machine increments an internal counter for each invalid character received and incorrectly positioned comma character. The internal counter is decremented when four consecutive valid characters are received. When the counter reaches 0, the link synchronization is lost.

The PCS function drives the signal `led_link` to 1 when link synchronization is acquired. This signal can be used as a common visual activity check using a board LED.

Carrier Sense

The carrier sense state machine detects an activity when the link synchronization is acquired and when the transmit and receive encapsulation or de-encapsulation state machines are not in the idle or error states.

The carrier sense state machine drives the signals `mii_rx_crs` and `led_crs` to 1 when it detects an activity. The signal `led_crs` can be used as a common visual activity check using a board LED.

Collision Detection

A collision is detected when the non-idle frame data is received from the PHY and transmitted to the PHY simultaneously. Collisions can be detected only in SGMII and half-duplex mode.

The collision detection state machine drives the signals `mii_rx_col` and `led_col` to 1 when it detects a collision. The signal `led_col` can be used as a common visual activity check using a board LED.

PCS Transmit

This section describes the PCS transmit operation, which includes frame encapsulation and encoding.

Frame Encapsulation

During transmission, the PCS function encapsulates frames according to the specification in IEEE 802.3 Clause 36. The first byte of the preamble in the MAC frame is replaced with the start of frame `/S/` symbol. Following the insertion on `/S/`, all of the MAC frame is encoded with standard 8B/10B encoded characters. After the last FCS byte, the end of packet `/T/` `/R/` `/R/` or the `/T/` `/R/` sequence is inserted. The selection of the end packet sequence is based on odd/even number of character transmission. Between frames, `/I/` symbols are transmitted.

If a frame is received from the MAC function with an error indication (Signal `gm_tx_err` asserted during frame transfer), the encapsulation function inserts a `/V/` character to encode an error that can be decoded by the remote end PHY device.

8b/10b Encoding

The 8B/10B encoder maps 8-bit words to 10-bit symbols to generate a DC balanced stream with a maximum run length of 5.

SGMII Converter

The SGMII converter is only enabled when the PCS function is configured to operate in SGMII mode by setting the `SGMII_ENA` bit in the `if_mode` register is to 1. In 1000BASE-X mode, the PCS function always operates in gigabit mode and data duplication is disabled.

In SGMII mode, if the `USE_SGMII_AN` bit in the `if_mode` register is set to 1, the SGMII converter is automatically configured with the capabilities advertised by the PHY. Otherwise, it is recommended to configure the SGMII converter with the `SGMII_SPEED` bits in the `if_mode` register.

Transmit

If the PCS function is programmed to operate in gigabit mode, the PCS and the MAC function operate at the same rate. The transmit converter transmits each byte from the MAC function once to the PCS function.

In 100 Mbps mode, the transmit converter replicates each byte received by the PCS function 10 times. In 10 Mbps, the transmit converter replicates each byte transmitted from the MAC function to the PCS function 100 times.

Receive

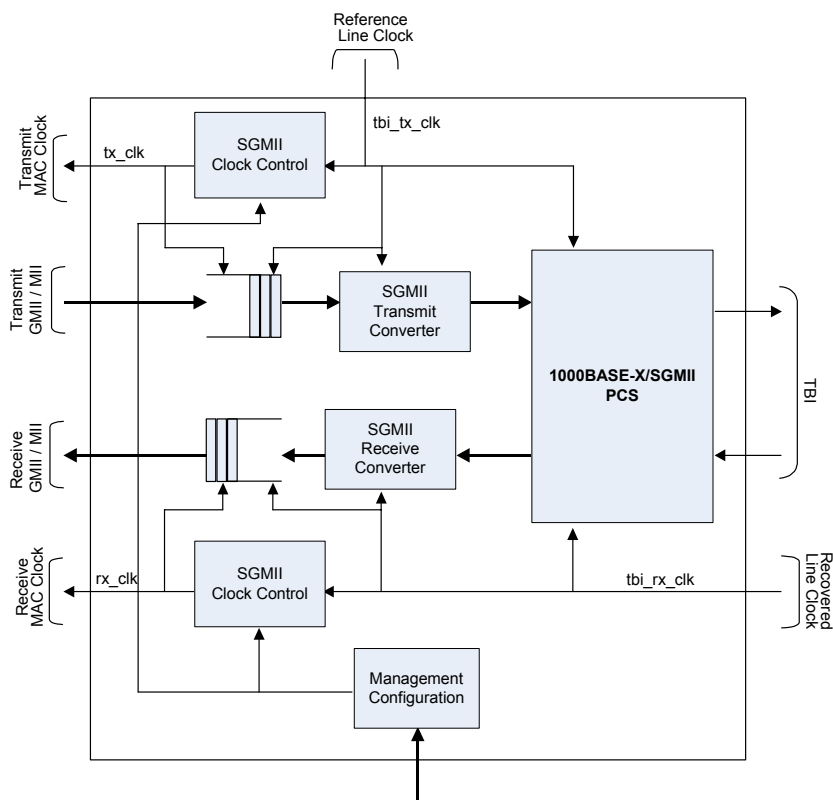
If the PCS function is programmed to operate in gigabit mode, the PCS and the MAC function both operate at the same rate. The transmit converter transmits each byte from the PCS function once to the MAC function.

In 100 Mbps mode, the receive converter transmits one byte out of 10 bytes received from the PCS function to the MAC. In 10 Mbps, the receive converter transmits one byte out of 100 bytes received from the PCS function to the MAC function.

Clock Distribution with External PMA

When connected to an external PHY, the transmit MAC clock (`tx_clk`) is generated from a division of the TBI 125 MHz reference clock `tbi_tx_clk`. The receive MAC clock (`rx_clk`) is generated from a division of the TBI 125 MHz line recovered clock `tbi_rx_clk`.

[Figure 4-37](#) provides a block diagram for the clock distribution with an external PMA.

Figure 4–37. SGMII Clock Distribution with External PMA

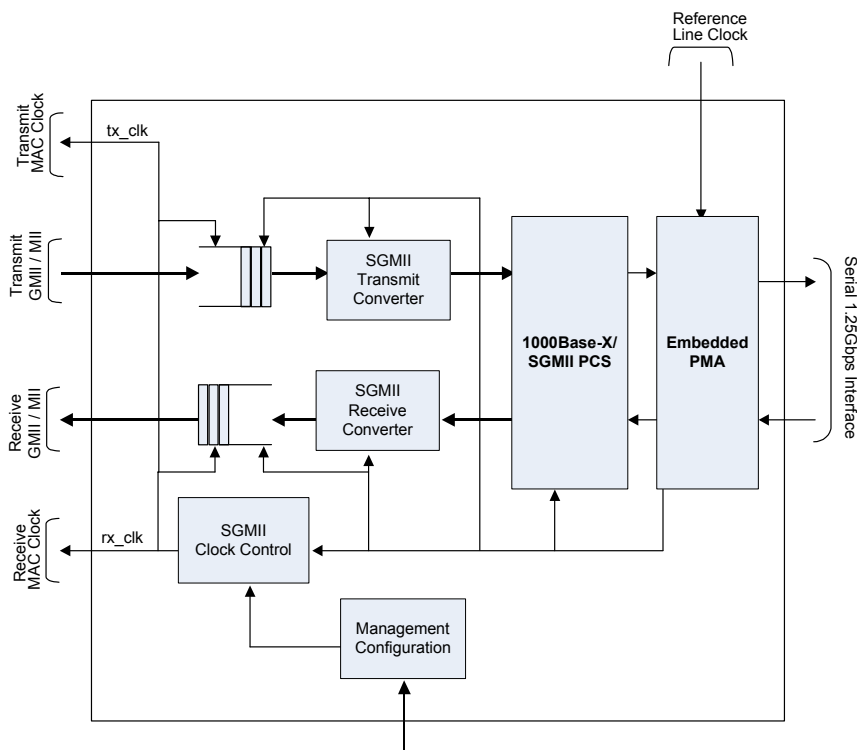
The MAC Clock speed depends on the PCS function operation listed in [Table 4–24](#).

Table 4–24. MAC Clock Frequency	
PCS Function Operation	MAC Clock Frequency
Gigabit	125 Mhz
100 Mbps	25 MHz
10 Mbps	2.5 MHz

Clock Distribution with Embedded PMA

When the PCS function is implemented in Altera devices with GX transceivers, a rate matching function is implemented and a single clock is then used on the PCS transmit and receive domains. [Figure 4-38](#) provides a block diagram for the clock distribution with an embedded PMA.

Figure 4-38. SGMII Clock Distribution with Embedded PMA



The MAC clock speed depends on the PCS function operation, as listed in [Table 4-24 on page 4-63](#).

Auto-Negotiation

Auto-negotiation is an optional function that can be started when link synchronization is acquired during system start up. To start auto-negotiation automatically, set the `AUTO_NEGOTIATION_ENABLE` bit in the `PCS_control` register to 1. During auto-negotiation, the PCS function advertises its device features and exchanges them with a link partner device.

The mode in which the PCS function operates determines how auto-negotiation is carried out. If the `SGMII_ENA` bit in the `if_mode` register is set to 0, the PCS function operates in 1000BASE-X. Otherwise, the operating mode is SGMII. The following sections describe the auto-negotiation process for each operating mode.

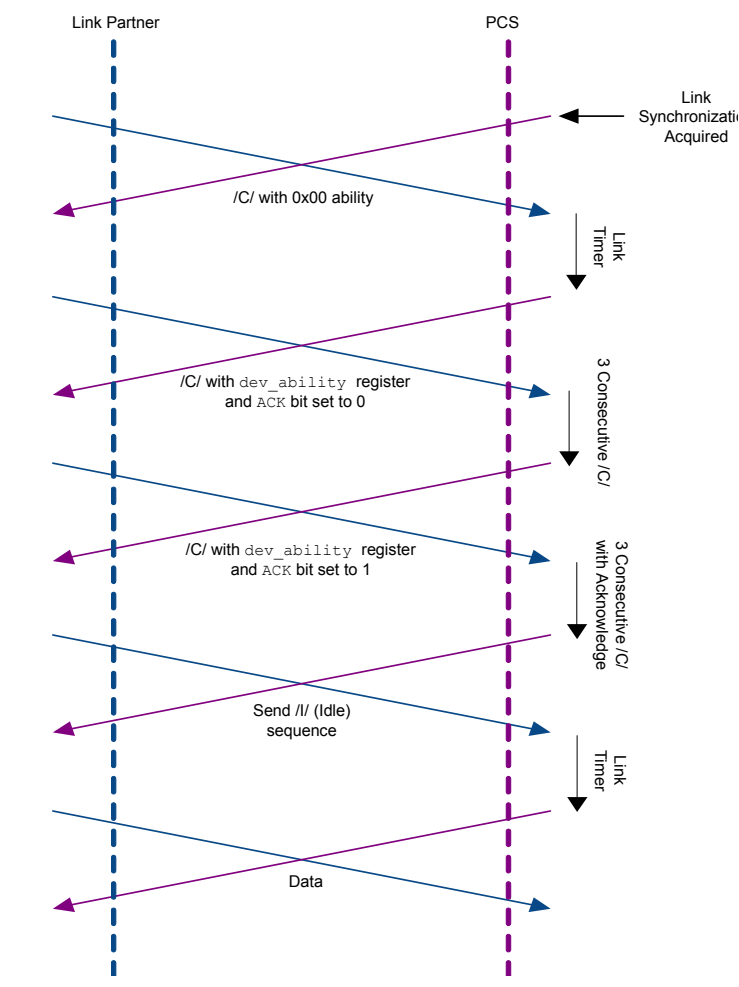
1000BASE-X Auto-Negotiation

When link synchronization is acquired, the PCS function starts sending a `/C/` sequence (configuration sequence) to the link partner device with the advertised register set to 0x00. The sequence is sent for a time specified in the `PCS_link_timer` register mapped in the PCS register space. See “[PCS Control Register](#)” on page 4-72 for a description of the `link_timer` register.

When the `link_timer` time expires, the `PCS_dev_ability` register is advertised, with the `ACK` bit set to 0 for the link partner. The auto-negotiation state machine checks for three consecutive `/C/` sequences received from the link partner.

The auto-negotiation state machine then sets the `ACK` bit to 1 in the advertised `dev_ability` register and checks if three consecutive `/C/` sequences are received from the link partner with the `ACK` bit set to 1.

Auto-negotiation waits for the value configured in the `link_timer` register to ensure no more consecutive `/C/` sequences are received from the link partner. The auto-negotiation is successfully completed when three consecutive idle sequences are received after the link timer expires. This sequence of activities is illustrated by [Figure 4-39](#).

Figure 4–39. Auto-negotiation Simplified

Once auto-negotiation is successfully completed, the ability advertised by the link partner device is available in the partner_ability register (Table 4–28 on page 4–74) and the AUTO_NEGOTIATION_COMPLETE bit in the status register (Table 4–27 on page 4–73) is set to 1.

Auto-negotiation is restarted if link synchronization is lost and re-acquired or if the RESTART_AUTO_NEGOTIATION bit in the PCS control register (Table 4–26 on page 4–72) is set to 1 by the user application.

SGMII Auto-Negotiation

In SGMII, the capabilities of the PHY device are advertised and exchanged with a link partner PHY device.

If the `SGMII_ENA` and `USE_SGMII_AN` bits in the `if_mode` register are 1, the PCS function is automatically configured with the capabilities advertised by the PHY device once the auto-negotiation completes.

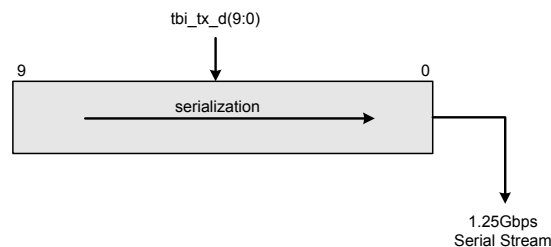
If the `SGMII_ENA` bit is 1 and the `USE_SGMII_AN` bit is 0, the PCS function can be configured with the `SGMII_SPEED` and `SGMII_DUPLEX` bits in the `if_mode` register.

TBI Interface

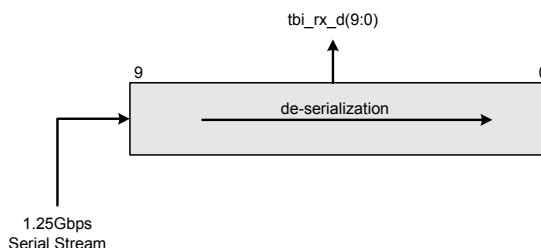
The PCS function implements a TBI connected to an external SERDES when the PCS function is implemented in devices without internal GX transceivers.

On transmit, the SERDES must serialize `tbi_tx_d(0)`, the least significant bit of the TBI output bus first and `tbi_tx_d(9)`, the most significant bit of the TBI output bus last to ensure the remote node receives the data correctly, as [Figure 4-40](#) illustrates.

Figure 4-40. SERDES Serialization Overview

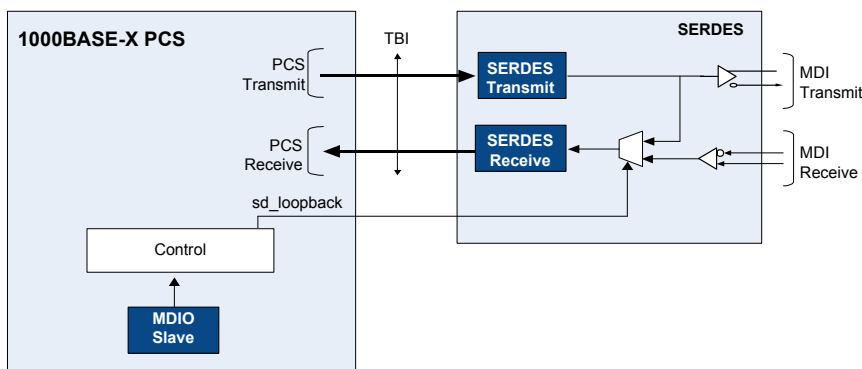


On receive, the SERDES must serialize the TBI least significant bit first and the TBI most significant bit last, as [Figure 4-41](#) illustrates.

Figure 4–41. SERDES De-Serialization Overview

PHY Loopback

A loopback is typically implemented on the serial MDI interface to allow testing of the PCS and PMA function in isolation of the PMD. To enable loopback, set the `sd_loopback` bit in the PCS `control` register to 1.

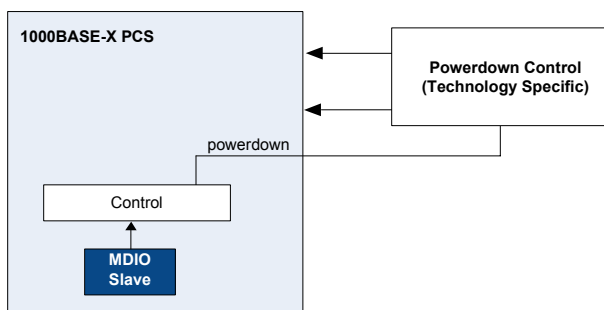
Figure 4–42. Serial Loopback

PHY Power-Down

Power-down is controlled by the `POWERDOWN` bit in the PCS `control` register. When the system management agent enables the power-down, the PCS function drives the `powerdown` signal, which can be used to control a technology specific circuit to switch off the PCS function clocks to reduce the application activity.

When the PHY is in power down state, the PCS function is in reset and any activities on the GMII transmit and the TBI receive interfaces are ignored. The management interface remains active and responds to management transactions from the MAC layer device.

Figure 4–43. Power-Down



Power-Down with Embedded PMA

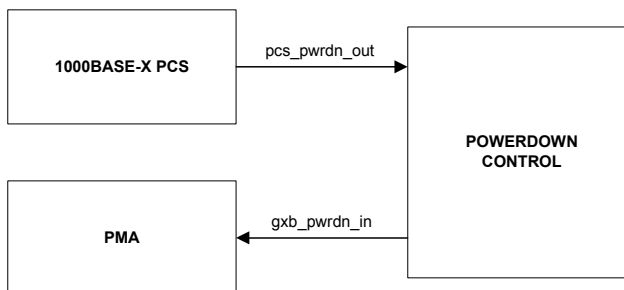
When the PCS function is implemented with an embedded PMA, the power-down signal is internally connected to the power-down of the GX transceiver.

In Altera devices with internal GX transceivers, the power-down functionality is shared across quad-port transceiver blocks. Multi-port Ethernet designs must share a common `gxb_powerdown_in` signal to use the same quad-port transceiver block.

You can export the power-down signals to implement your own power-down logic to efficiently use the transceivers within a particular quad-port block. The power-down signal can be exported by turning on the **Export transceiver powerdown signal** parameter. For more information on the parameter, refer to [“PCS/SGMII Options” on page 3–7](#).

Figure 4–44 illustrates the power-down control with exported power-down signal.

Figure 4–44. Power-Down with Export Transceiver Power-Down Signal



PCS Control Interface Register Map

Table 4–25 lists and describes the registers that provide access to the PCS clock.

Table 4–25. PCS Interface Register Map (Part 1 of 2)

Address Offset	Register Name	Description	Access ⁽¹⁾
0x00	control	PCS Control register. Use the bits in this register to control and configure the PCS function. For the register bits description, see Table 4–26 on page 4–72.	RW
0x02	status	Status register. Provides information on the operation of the PCS function.	RO
0x04 0x06	phy_identifier	32-bit PHY identification register. You can set this register to a customer unique value when parameterizing the PCS function. Bits 31:16 are written to offset 0x04. Bits 15:0 are written to offset 0x06.	RO
0x08	dev_ability	Use this register to advertise the device abilities to a link partner during auto-negotiation. In SGMII mode, the PHY does not use this register during auto-negotiation. For the register bits description, see Table 4–28 on page 4–74.	RW
0x0A	partner_ability	Contains the device abilities advertised by the link partner during auto-negotiation. For the register bits description in 1000BASE-X and SGMII mode, refer to Table 4–28 on page 4–74 and Table 4–29 on page 4–75, respectively.	RO
0x0C	an_expansion	Auto-negotiation expansion register. Contains the PCS function capability and auto-negotiation status.	RO

Table 4–25. PCS Interface Register Map (Part 2 of 2)

Address Offset	Register Name	Description	Access ⁽¹⁾
0x0E	device_next_page	The PCS function does not support next page. These registers are always set to 0x0000 and any write access to the registers is ignored.	RO
0x10	partner_next_page		
0x12	master_slave_cntl	The PCS function does not support 100Base-T or 1000Base-T operation. The registers are always set 0x0000.	RO
0x14	master_slave_stat		
0x16 to 0x1C	Reserved	-	
0x1E	extended_status	The PCS function does not implement extended status registers.	RO
Specific Extended Registers			
0x20	scratch	Scratch register. Provides a memory location to test register read and write operations.	RW
0x22	rev	(RO)The PCS function revision. Always set to 0x0701.	RO
0x24	link_timer	21-bit auto-negotiation link timer. Set the link timer value from 0 to 16 ms in 8 ns steps (125 MHz clock periods). The reset value sets the link timer to 10 ms. Bits 15:0 are written to offset 0x24. Bit 0 of offset 0x24 is always set to 0, thus any value written to it is ignored. Bits 20:16 are written to offset 0x26. The remaining bits are reserved and always set to 0.	RW
0x26			
0x28	if_mode	Interface mode. Use this register to specify the operating mode of the PCS function; 1000BASE-X or SGMII.	RW
0x2A to 0x3E	Reserved	-	

Note:

(1) RW = Read/Write, RO = Read only

PCS Control Register

Table 4–26 describes the function of each bit and field in the PCS control register.

Table 4–26. PCS Control Register Bit Descriptions

Bit(s)	Name	Description	Access ⁽¹⁾
0 to 5	Reserved	Always set to 0.	RO
6 and 13	SPEED_SELECTION	Defines a PCS function operating in gigabit mode. To program gigabit mode, set <ul style="list-style-type: none"> • Bit 13 set to 0. • Bit 6 set to 1. 	RO
7	COLLISION_TEST	The PCS function does not support half-duplex mode. Always set to 0.	RO
8	DUPLEX_MODE	The PCS function only supports full-duplex mode. Always set to 1.	RO
9	RESTART_AUTO_NEGOTIATION	Setting this bit to 1 restarts the auto-negotiation sequence. For normal operation, set this bit to 0 (reset value).	RW
10	ISOLATE	Setting this bit to 1 isolates the PCS function from the MAC Layer device. For normal operation, set this bit to 0 (reset value).	RW
11	POWERDOWN	Setting this bit to 1 cause the PCS function to drive its power down output signal.	RW
12	AUTO_NEGOTIATION_ENABLE	Setting this bit to 1 (reset value) enables auto-negotiation.	RW
14	LOOPBACK	PHY loopback. Setting this bit to 1 implements a loopback in the PMA. For normal operation, set this bit to 0 (reset value). This bit is ignored if reduced ten-bit interface (RTBI) is implemented.	RW
15	RESET	Setting this bit to 1 generates a synchronous reset pulse which resets all the PCS function state machines, comma detection function and 8b/10b encoder and decoder. For normal operation, set this bit to 0 (asynchronous reset value).	RW/SC

Note:

(1) RW = Read/Write, RO = Read only, SC = Self-clearing

Status Register

Table 4–27 describes the function of each bit and field in the status register. All bits in this register are read-only bits.

Table 4–27. Status Register Bit Descriptions

Bit Number	Name	Description
0	EXTENDED_CAPABILITY	A value of 0 indicates that the PCS function supports extended registers.
1	JABBER_DETECT	The PCS function does not support the optional Jabber detection function. Always set to 0.
2	LINK_STATUS	A value of 1 indicates that a valid link is established. A value of 0 indicates an invalid link. If the link synchronization is lost, a 0 is latched which is cleared only after a register read access.
3	AUTO_NEGOTIATION_ABILITY	A value of 1 indicates that the PCS function supports auto-negotiation.
4	REMOTE_FAULT	The PCS function does not implement a PHY specific remote fault detection optional function. Always set to 0.
5	AUTO_NEGOTIATION_COMPLETE	A value of 1 indicates the following status: <ul style="list-style-type: none"> • The auto-negotiation process is completed. • The auto-negotiation control registers are valid.
6	MF_PREAMBLE_SUPPRESSION	The PHY does not accept management frames with preamble suppressed. Always set to 0.
7	UNIDIRECTIONAL_ABILITY	Always set to 0 to indicate that the PHY is able to transmit from media independent interface only when a valid link is established.
8	EXTENDED_STATUS	The PCS function does not implement an extended status register. Always set to 0.
9	100BASET2_HALF_DUPLEX	The PCS function does not support 100Base-T2 operation. Always set to 0.
10	100BASET2_FULL_DUPLEX	
11	10MBPS_HALF_DUPLEX	The PCS function does not support 10 Mbps operation. Always set to 0.
12	10MBPS_FULL_DUPLEX	
13	100BASE-X_HALF_DUPLEX	The PCS function does not support 100BASE-X operation. Always set to 0.
14	100BASE-X_FULL_DUPLEX	
15	100BASE-T4	The PCS function does not support 100Base-T4 operation. Always set to 0.

Dev_Ability and Partner_Ability Registers

The bits and fields in the `partner_ability` registers are defined differently depending on the mode in which the PCS function operates.

The `dev_ability` register is valid only in 1000BASE-X mode. In this mode, the bits and fields in the `dev_ability` register are the same as the bits and fields in the `partner_ability` register.

1000BASE-X

Table 4–28 describes the function of each bit and field in the `dev_ability` and `partner_ability` register when the PCS function operates in 1000BASE-X mode.

Table 4–28. Dev_Ability and Partner_Ability Registers Bits Description in 1000Base-X

Bit(s)	Name	Description
0 to 4	Reserved	Always set these bits to 0.
5	FD	Full duplex enable. A value of 1 indicates support for full duplex.
6	HD	Half duplex enable. A value of 1 indicates support for half duplex.
7	PS1	Pause support. <ul style="list-style-type: none"> PS1=0 / PS2=0: Pause is not supported. PS1=0 / PS2=1: Asymmetric pause toward link partner. PS1=1 / PS2=0: Symmetric pause. PS1=1 / PS2=1: Pause is supported on transmit and receive.
8	PS2	
9 to 11	Reserved	Always set these bits to 0.
12	RF1	Remote fault condition: <ul style="list-style-type: none"> RF1=0 / RF2=0: No error, link is valid (reset condition). RF1=0 / RF2=1: Offline. RF1=1 / RF2=0: Failure condition. RF1=1 / RF2=1: Auto-negotiation error.
13	RF2	
14	ACK	Acknowledge. A value of 1 indicates that the device has received 3 consecutive matching ability values from its link partner.
15	NP	Next page. In <code>dev_ability</code> register, this bit is always set to 0.

Note:

- (1) All bits in the `dev_ability` register have read/write access.
- (2) All bits in the `partner_ability` register are read-only.

SGMII

Table 4–29 describes the function of each bit and field in the `partner_ability` register when the PCS function operates in SGMII mode. All bits in this register are read-only.

Table 4–29. Partner_Ability Register Bits Description in SGMII

Bit(s)	Name	Description
0 to 9	Reserved	-
10 to 11	COPPER_SPEED(1:0)	Link partner interface speed: <ul style="list-style-type: none"> 00: copper interface speed is 10 Mbps 01: copper interface speed is 100 Mbps 10: copper interface speed is gigabit 11: reserved
12	COPPER_DUPLEX_STATUS	Link partner duplex capability: <ul style="list-style-type: none"> 1: copper Interface is capable of operating in half-duplex mode 0: copper Interface is capable of operating in full-duplex mode
13	Reserved	-
14	ACK	Acknowledge. A value of 1 indicates that the link partner has received 3 consecutive matching ability values from the device.
15	COPPER_LINK_STATUS	Copper link partner status: <ul style="list-style-type: none"> 1: copper interface link is up 0: copper interface link is down

An_Expansion Register

Table 4–30 describes the function of each bit and field in the `an_expansion` register. All bits in this register are read-only.

Table 4–30. An_Expansion Register Description

Bit(s)	Name	Description
0	LINK_PARTNER_AUTO_NEGOTIATION_ABLE	A value of 1 indicated that the link partner supports auto-negotiation. The reset value is 0.
1	PAGE_RECEIVE	A value of 1 indicates that a new page is received with new partner ability available in the register <code>partner_ability</code> . The bit is set to 0 (reset value) when the system management agent performs a read access.
2	NEXT_PAGE_ABLE	The PCS function does not support next page. This bit is always 0.
3 to 15	Reserved	-

If_Mode Register

Table 4–31 describes the function of each bit and field in the `if_mode` register.

Table 4–31. IF_Mode Register Description			
Bit(s)	Name	Description	Access⁽¹⁾
0	SGMII_ENA	Determines the PCS function operating mode. Setting this bit to 1 enables SGMII mode. Setting this bit to 0 enables 1000BASE-X gigabit mode.	RW
1	USE_SGMII_AN	This bit applies only to SGMII mode. Setting this bit to 1 causes the PCS function to be configured with the link partner abilities advertised during auto-negotiation. If this bit is set to 0, it is recommended for the PCS function to be configured with the SGMII_SPEED and SGMII_DUPLEX bits.	RW
2 to 3	SGMII_SPEED(1:0)	SGMII speed. When the PCS function operates in SGMII mode (SGMII_ENA set to 1) and programmed not to be automatically configured (USE_SGMII_AN is 0), set the speed as follows: <ul style="list-style-type: none"> • 00: 10 Mbps • 01: 100 Mbps • 10: Gigabit • 11: Reserved These bits are ignored when SGMII_ENA is 0 or USE_SGMII_AN is 1	RW
4	SGMII_DUPLEX	SGMII duplex mode. Setting this bit to 1 enables full duplex.	RW
5 to 15	Reserved	-	RO

Note:

(1) RW = Read/Write, RO = Read only

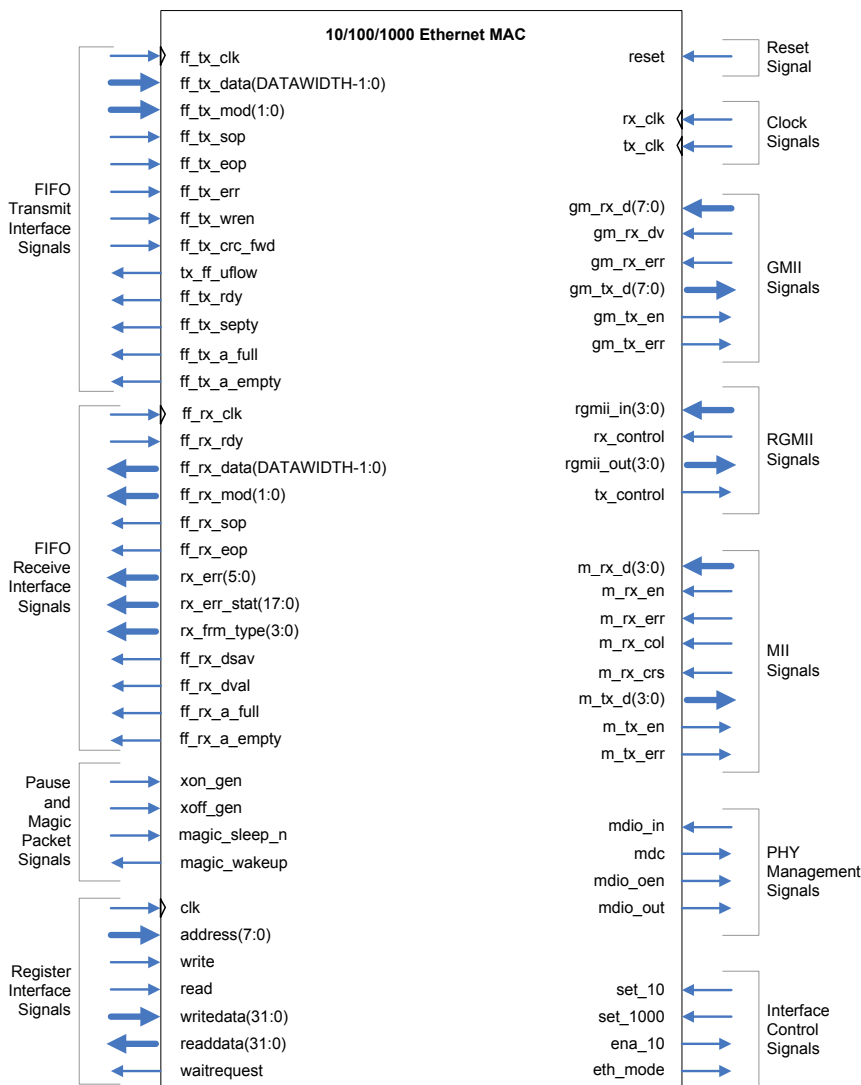
Signals

This section describes all interface signals of each possible configuration.

10/100/1000 Ethernet MAC Signals

Figure 4–45 shows all I/O signals of the 10/100/1000 Ethernet MAC function.

Figure 4–45. 10/100/1000 MAC Function Signals



Control Interface Signals

The control register interface is an Avalon Memory Mapped (Avalon-MM) slave port. This interface controls both the MAC and PCS blocks in the MegaCore function. The slave port has the following properties, and behaves in accordance with the *Avalon Memory-Mapped Interface Specification*:

- 32-bit readdata and writedata signals
- 8-bit address signal
- No wait states

The 8-bit address provides access to a register space of 256, 32-bit registers. The complete register map is described in [Table 4-12 on page 4-31](#).

Register Interface Signals

[Table 4-32](#) describes the signals that comprise the register interface for the MAC block.

Table 4-32. MAC Register Interface Signals			
Signal Name	Avalon-MM Signal Type	Direction	Description
clk	clk	In	Register access reference clock.
write	write	In	Register write enable.
read	read	In	Register read enable.
address(9:2)	address	In	Register address.
writedata(31:0)	writedata	In	Register write data. Bit 0 is the least significant bit.
readdata(31:0)	readdata	Out	Register read data. Bit 0 is the least significant bit.
waitrequest	waitrequest	Out	Register interface busy. Asserted (1) during register read or register write access. Set to 0 to indicate the completion of the current register access.

Reset Signal

[Table 4-33](#) describes the reset signal.

Table 4-33. Reset Signal		
Signal Name	Direction	Description
reset	In	Single reset for all logic in the MAC and PCS Register Interface.

MAC System-Side Signals

The system-side interface to the MAC block is implemented as two Avalon Streaming (Avalon-ST) ports. An Avalon-ST source port provides an interface to the MAC receive FIFO; an Avalon-ST sink port provides an interface to the MAC transmit FIFO. There are a few additional signals that are not associated with either Avalon-ST port.



For Information About	Refer To
Avalon-ST interface protocol	Avalon Streaming Interface Specification

When instantiating the MegaCore function in an SOPC Builder system, SOPC Builder automatically connects the Avalon-ST ports to the rest of the system. When instantiating the MegaCore function stand-alone, the Avalon-ST signals appear at the top-level of the variant HDL file, and you must manually connect them.

Receive FIFO Signals

The Avalon-ST source port has the following properties and behaves in accordance with the *Avalon Streaming Interface Specification*:

- Data width of 8 or 32 bits.
- Use of backpressure to stop transmission when the FIFO reaches a programmable threshold.
- Packet support using start and end of packet signals, and partial final packet signals.
- Error reporting.

[Table 4–34](#) lists all interface signals associated with the receive FIFO.



DATAWIDTH can be set to either 8 or 32 bit by configuring the parameter **Width**. For more information on the parameter, refer to “FIFO Options” on page 3–6.

Table 4–34. Receive FIFO Signals (Part 1 of 2)

Signal Name	Avalon-ST Type	Direction	Description
ff_rx_clk	clk	In	Receive FIFO clock. Can be set to any value required to get the required bandwidth with the FIFO interface. Can be completely independent from the GMII / MII clocks (rx_clk). All the receive FIFO signals are synchronized on the rising edge of ff_rx_clk.
ff_rx_dval	valid	Out	Receive data valid. Asserted (set to 1) by the MAC to indicate that data on ff_rx_data ([DATAWIDTH-1]:0), ff_rx_sop, ff_rx_eop, rx_err (5:0), rx_frm_type (3:0) and rx_err_stat (23:16) are valid.
ff_rx_data ([DATAWIDTH-1]:0)	data	Out	Receive data. When DATAWIDTH is 32, the first byte received is ff_rx_data (31:24) followed by ff_rx_data (23:16) etc.
ff_rx_mod (1:0)	empty	Out	Receive data modulo. Indicates which part of the final Frame word is not valid: <ul style="list-style-type: none"> 11: ff_rx_data (23:0) is not valid 10: ff_rx_data (15:0) is not valid 01: ff_rx_data (7:0) is not valid 00: ff_rx_data (31:0) is valid This signal applies only to 32-bit DATAWIDTH.
ff_rx_sop	startof-packet	Out	Receive start of packet. Set to 1 when the first octet or word of a frame is driven on ff_rx_data ([DATAWIDTH-1]:0).
ff_rx_eop	endofpacket	Out	Receive end of packet. Set to 1 when the last octet or word of frame data is driven on ff_rx_data ([DATAWIDTH-1]:0).
ff_rx_dsav	component specific signal	Out	Receive frame available. Indicates that the Receive FIFO contains data to be read (not necessarily a complete frame). The application might want to start reading from the FIFO.
rx_frm_type (3:0)	component specific signal	Out	For the description of each bit, refer to Table 4–35 on page 4–81 .
ff_rx_a_full	component specific signal	Out	Receive FIFO almost-full threshold. This signal is asserted when the FIFO reaches the almost full threshold.

Table 4–34. Receive FIFO Signals (Part 2 of 2)

Signal Name	Avalon-ST Type	Direction	Description
<code>ff_rx_a_empty</code>	component specific signal	Out	Receive FIFO almost-empty threshold. This signal is asserted when the FIFO goes below the almost empty threshold.
<code>ff_rx_rdy</code>	ready	In	Receive application ready. Asserted (1) by the receive application to indicate it is ready to receive data from the MAC. The signal <code>ff_rx_rdy</code> must be generated on the rising edge of <code>ff_rx_clk</code> .
<code>rx_err(5:0)</code>	error	Out	Receive error. Asserted with the final frame octet to indicate that an error was detected when receiving the frame. For the description of each bit, refer to Table 4–36 on page 4–82 .
<code>rx_err_stat(17:0)</code>	component specific signals	Out	<code>rx_err_stat(17)</code> : One indicates that the current frame implements stacked VLAN <code>rx_err_stat(16)</code> : One indicates that the current frame implements either VLAN or stacked VLAN <code>rx_err_stat(15:0)</code> : Received frame payload length/type in octets as found in length/type field

[Table 4–35](#) describes the each bit of the VLAN frame indication signals.

Table 4–35. VLAN Frame Indication Signals (`rx_frm_type(3:0)`)

Bit Number	Description
3	VLAN frame indication. Asserted together with <code>ff_rx_sop</code> and remains asserted until the end of the frame. (<code>ff_rx_eop</code> asserted)
2	Broadcast frame indication. Asserted together with <code>ff_rx_sop</code> and remains asserted until the end of the frame. (<code>ff_rx_eop</code> asserted)
1	Multicast frame indication. Asserted together with <code>ff_rx_sop</code> and remains asserted until the end of the frame. (<code>ff_rx_eop</code> asserted)
0	Unicast frame indication. Asserted together with <code>ff_rx_sop</code> and remains asserted until the end of the frame. (<code>ff_rx_eop</code> asserted)

Table 4–36 describes the each bit of the Avalon-ST Receive Channel Error Types.

Table 4–36. Avalon-ST Receive Channel Error Types (rx_err(5:0))	
Bit Number	Description
5	Collision error. Asserted when the frame was received with a collision
4	Received frame corrupted due to PHY error. (The PHY has asserted an error on the receive GMII interface.)
3	Receive frame truncated. Asserted when the received frame has been truncated due to receive FIFO overflow
2	CRC error. Asserted when the frame has been received with a CRC-32 error.
1	Invalid length error. Asserted when the received frame has an invalid length as defined by the IEEE 802.3 standard.
0	Receive frame error. This signal indicates that an error has occurred. It is the logical OR of receive errors 1 through 5.

Transmit FIFO Signals

The Avalon-ST sink port has the following properties, and behaves in accordance with the *Avalon Streaming Interface Specification*:

- Low latency, high throughput data transfer
- Data width of 8 or 32 bits
- Support for packets using start, end of packet signals and partial final packet signals
- Error reporting
- Optional calculation of CRC

Table 4–37 lists all interface signals associated with the transmit FIFO.



DATAWIDTH can be set to either 8 or 32 bit by configuring the parameter **Width**. For more information on the parameter, refer to “FIFO Options” on page 3–6.

Table 4–37. Transmit FIFO Signals (Part 1 of 2)

Signal Name	Avalon-ST Type	Direction	Description
ff_tx_clk	clk	In	Transmit FIFO clock. Can be set to any value required to get the bandwidth on the FIFO interface. Can be completely independent from the GMII / MII clock tx_clk. All transmit FIFO signals are synchronized on the rising edge of ff_tx_clk.
ff_tx_wren	valid	In	Transmit data write enable. Asserted by the transmit application to write data to the FIFO. Indicates that ff_tx_data([DATAWIDTH-1]:0), ff_tx_sop, ff_tx_eop are valid.
ff_tx_data([DATAWIDTH-1]:0)	data	In	Transmit data. DATAWIDTH is either 8 or 32 depending on the FIFO data width configuring during MegaCore parameterization. When DATAWIDTH is 32, the first byte transmitted is ff_tx_data(31:24) followed by ff_tx_data(23:16) and so forth.
ff_tx_mod(1:0)	empty	In	Transmit data modulo. Indicates which part of the final frame word is valid: <ul style="list-style-type: none"> 11: ff_tx_data(23:0) is not valid 10: ff_tx_data(15:0) is not valid 01: ff_tx_data(7:0) is not valid 00: ff_tx_data(31:0) is valid This signal applies only to 32-bit DATAWIDTH.
ff_tx_sop	startofpacket	In	Transmit start of packet. Set to 1 when the frame first octet, the first octet of the destination address, is driven on ff_tx_data.
ff_tx_eop	endofpacket	In	Transmit end of packet. Set to 1 when the frame final octet, the last FCS field, is driven on ff_tx_data.
ff_tx_err	error	In	Transmit frame error. Asserted with the frame final octet to indicate that the transmitted frame is invalid. When ff_tx_err is asserted, the frame is either transmitted to the GMII interface with an error or discarded by the MAC function.
ff_tx_crc_fwd	component specific signal	In	Transmit CRC insertion. If set to 0 together with tx_ff_eop, a CRC is calculated and inserted into the frame. Otherwise, the MAC does not insert a CRC into. In this case, the user application is expected to provide the CRC.

Table 4–37. Transmit FIFO Signals (Part 2 of 2)

Signal Name	Avalon-ST Type	Direction	Description
tx_ff_uflow	component specific signal	Out	Asserted when a FIFO underflow occurs on the transmit FIFO.
ff_tx_rdy	ready	Out	MAC ready. Asserted by the MAC function to indicate that it is ready to accept data from the user application.
ff_tx_septy	component specific signal	Out	FIFO data section empty. Deasserted, when the FIFO is filled to or above the threshold defined in <code>tx_section_empty</code> register. User applications can use this signal to stop FIFO write and initiate backpressure measures.
ff_tx_a_full	component specific signal	Out	Asserted when the transmit FIFO reaches the almost full threshold.
ff_tx_a_empty	component specific signal	Out	Asserted when the transmit FIFO goes below the almost empty threshold.

Table 4–38 lists all pause and magic packet interface signals.

Table 4–38. Pause and Magic Packet Signals (Part 1 of 2)

Signal Name	Avalon-ST Type	Direction	Description
xon_gen	component specific signal	In	XON Generate. When this signal is set to 1 by the user application for at least one <code>tx_clk</code> clock cycle, the MAC function generates, independently of the receive FIFO status, a pause frame with a 0 pause quanta. This signal is ignored if the <code>xon_gen</code> bit in the <code>command_config</code> register is set to 1.
xoff_gen	component specific signal	In	XOFF Pause Frame Generate. When this signal is set to 1 by the user application for at least one <code>tx_clk</code> clock cycle, the MAC function generates, independently of the receive FIFO status, a pause frame with the pause quanta programmed in the <code>pause_quant</code> register. This signal is ignored if the <code>xoff_gen</code> bit in the <code>command_config</code> register is set to 1.

Table 4–38. Pause and Magic Packet Signals (Part 2 of 2)

Signal Name	Avalon-ST Type	Direction	Description
<code>magic_sleep_n</code>	component specific signal	In	Enable Magic Packet Mode. Setting this signal to 0 puts the node into a power-down state. If Magic Packet support is enabled, indicated by the <code>MAGIC_ENA</code> bit in the <code>command_config</code> register, the MAC receive logic stops writing data to the MAC receive FIFO and the Magic Packet detection logic is enabled. Setting this signal to 1 puts the MAC in normal frame reception mode.
<code>magic_wakeup</code>	component specific signal	Out	Node Wake Up Status. If the MAC function is in the Magic Packet Mode, a signal value of 1 indicates that a Magic Packet has been detected and the node is requested to leave the power-down state.

MAC Ethernet-Side Signals

The following sections list the MAC PHY interface signals.

Clock Signals

Data transfers on the MAC Ethernet interface are synchronous to the receive and transmit clocks. [Table 4–39](#) describes the use of these clock signals.

Table 4–39. GMII/RGMII/MII Clock Signals

Signal Name	Direction	Description
<code>tx_clk</code>	In	GMII / RGMII/ MII transmit clock. Provides the timing reference for all GMII / MII transmit signals. The values of <code>gm_tx_d(7:0)</code> , <code>gm_tx_en</code> , <code>gm_tx_err</code> , and of <code>m_tx_d(3:0)</code> , <code>m_tx_en</code> , <code>m_tx_err</code> are valid on the rising edge of <code>tx_clk</code> .
<code>rx_clk</code>	In	GMII / RGMII/ MII receive clock. Provides the timing reference for all rx related signals. The values of <code>gm_rx_d(7:0)</code> , <code>gm_rx_dv</code> , <code>gm_rx_err</code> , and of <code>m_rx_d(3:0)</code> , <code>m_rx_en</code> , <code>m_rx_err</code> are valid on the rising edge of <code>rx_clk</code> .

MII/GMII/RGMII Interface Signals

The Ethernet-side interface for the MAC block uses the standard MII, GMII and RGMII interfaces to connect to an external PHY device. The MAC block provides an MDIO PHY management interface and other interface control signals.

Table 4–40 lists the MAC signals on the Ethernet-side interface.

Table 4–40. GMII/RGMII/MII Signals (Part 1 of 2)		
Signal Name	Direction	Description
GMII Transmit Interface		
gm_tx_d(7:0)	In	GMII transmit data bus.
gm_tx_en	Out	Asserted to indicate data on the data bus, gm_tx_d(7:0) is valid.
gm_tx_err	Out	Asserted to indicate to the PHY device that the current frame sent is invalid.
GMII Receive Interface		
gm_rx_d(7:0)	In	GMII receive data bus.
gm_rx_dv	In	Asserted to indicate data on the receive data bus is valid. Stays asserted during frame reception, from the first preamble byte until the last byte of the CRC field is received.
gm_rx_err	In	Asserted by the PHY to indicate that the current frame contains erroneous data.
RGMII Transmit Interface		
rgmii_out(3:0)	Out	RGMII transmit data bus. Drives gm_tx_d(3:0) on the positive edge of tx_clk and gm_tx_d(7:4) on the negative edge of tx_clk
tx_control	Out	Control output signal. Drives gm_tx_en on positive edge of tx_clk and a logical derivative of gm_tx_en XOR gm_tx_err on the negative edge of tx_clk.
RGMII Receive Interface		
rgmii_in(3:0)	In	RGMII receive data bus. Expects gm_rx_d(3:0) on the positive edge of rx_clk and gm_rx_d(7:4) on the negative edge of rx_clk
rx_control	In	RGMII control input signal. Expects gm_rx_dv on positive edge of rx_clk and a logical derivative of gm_rx_dv XOR gm_rx_err on the negative edge of rx_clk
MII Transmit Interface		
m_tx_d(3:0)	Out	MII transmit data bus.
m_tx_en	Out	Asserted to indicate data on the data bus, m_tx_d(3:0) is valid.
m_tx_err	Out	Asserted to indicate to the PHY device that the current frame sent is invalid.
MII Receive Interface		
m_rx_d(3:0)	In	MII receive data bus.
m_rx_en	In	Asserted to indicate data on the receive data bus is valid. Remains asserted during frame reception, from the first preamble byte until the last byte of the CRC field is received.
m_rx_err	In	Asserted by the PHY to indicate that the current frame contains erroneous data.
MII PHY Status		

Table 4–40. GMII/RGMII/MII Signals (Part 2 of 2)

Signal Name	Direction	Description
m_rx_col	In	Collision detection. Asserted by the PHY device to indicate a collision during a frame transfer. This signal is not used in half- duplex mode or when the MAC function operates in gigabit.
m_rx_crs	In	Carrier sense detection. Asserted by the PHY device to indicate that transmit or receive activity is detected on the Ethernet line.

PHY Management Signals

Table 4–41 lists the signals that manage the PHY interface.

Table 4–41. PHY Management Interface Signals

Signal Name	Direction	Description
mdio_in	In	Management data input.
mdio_out	Out	Management data output.
mdio_oeN	Out	Management data active low output enable.
mdc	Out	Management data clock. A data bit is shifted in/out on each rising edge of MDC. All fields are shifted in/out most significant bit first.

Interface Control Signals

The control interface signals allow you to set the transfer mode of the interface. Table 4–42 lists all signals associated with the control interface.

Table 4–42. Interface Control Signals (Part 1 of 2)

Signal Name	Direction	Description
eth_mode	Out	Ethernet mode. Set to 1 when the MAC function is configured to operate in gigabit mode. Set to 0 when the MAC function is configured to operate in 10/100 mode
ena_10	Out	10 Mbps enable. Set to 1 to indicate that the PHY interface should operate in 10 Mbps mode. This signal is only valid when the signal eth_mode is set to 0.

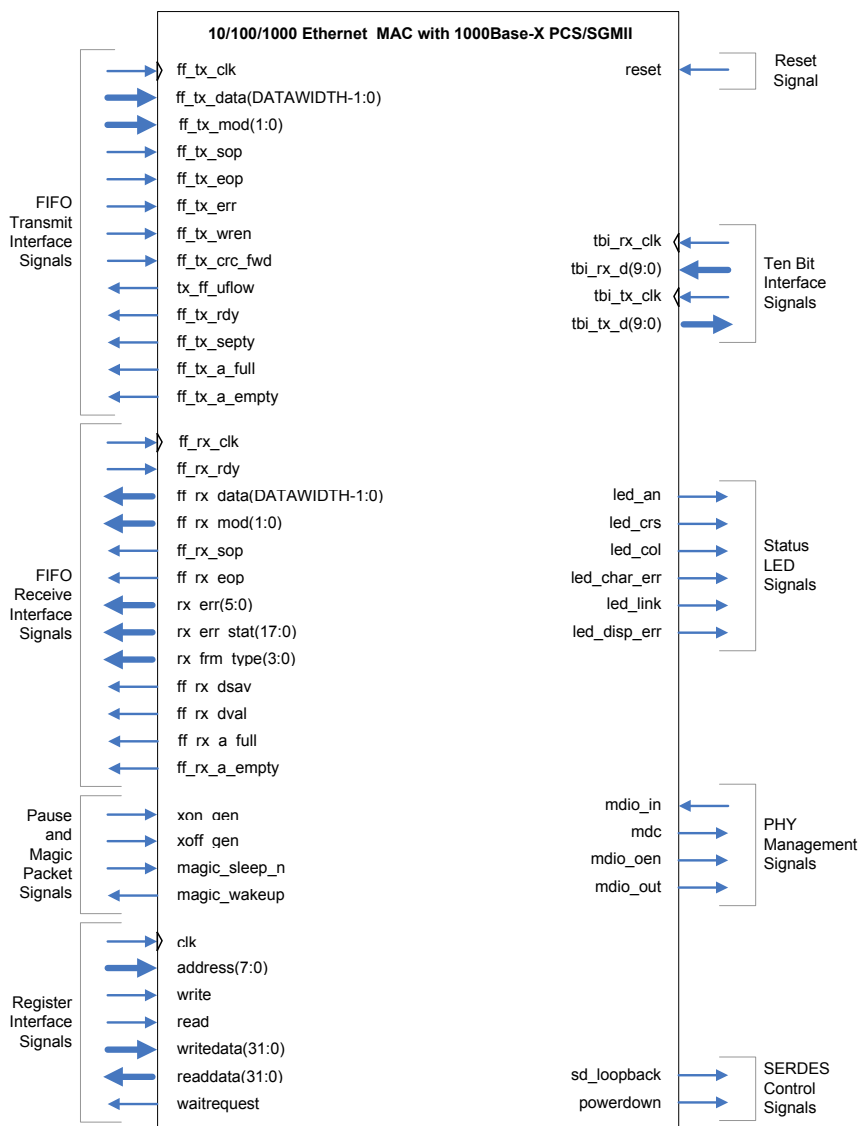
Table 4–42. Interface Control Signals (Part 2 of 2)

Signal Name	Direction	Description
set_1000	In	Gigabit mode selection. Can be driven to 1 by an external device, for example a PHY Device to set the MAC to operate in gigabit mode. When set to 0, the MAC is set to operate in 10/100 mode. The signal is ignored when the register bit <code>ETH_SPEED</code> is set to 1.
set_10	In	10 Mbps mode selection. Can be driven to 1 by an external device, for example a PHY Device to indicate that the MAC is connected to a 10 Mbps PHY device. When set to 0, the MAC is set to operate in 100 Mbps or gigabit mode. This signal is ignored when the register bit <code>ETH_SPEED</code> is set to 1 or when the register bit <code>ENA_10</code> is set to 1. The <code>ENA_10</code> software configuration bit has a higher priority than this signal.

10/100/1000 Ethernet MAC with 1000BASE-X/SGMII PCS Signals

Figure 4–46 shows all I/O signals of the 10/100/1000 Ethernet MAC with 1000BASE-X/SGMII PCS function.

Figure 4–46. 10/100/1000 Ethernet MAC with 1000BASE-X/SGMII PCS Function Signals



Control Interface Signals

For more information about control interface signals, refer to “[Control Interface Signals](#)” on page 4-78.

MAC System-Side Signals

For more information about MAC system-side signals, refer to “[MAC System-Side Signals](#)” on page 4-79.

PCS Ethernet-Side Signals

The Ethernet-side signals of the PCS function depend on whether the core includes GX transceivers. The PCS function also provides an interface for controlling status LEDs and programming the SGMII clock.

TBI Interface Signals for External SERDES Chip

If the variant does not include GX transceivers, the PCS block provides a 125 MHz ten-bit interface (TBI) to an external SERDES chip. [Table 4-43](#) lists the PCS signals to an external SERDES chip.

Table 4-43. TBI Interface Signals for External SERDES Chip

Signal Name	Direction	Description
tbi_tx_d(9:0)	Out	TBI transmit data, data transmitted by the PCS function synchronously with the signal tbi_tx_clk.
tbi_tx_clk	In	125 MHz TBI transmit clock from external SERDES. Typically local reference clock oscillator.
tbi_rx_clk	In	125 MHz TBI receive clock from external SERDES. Typically line clock recovered from encoded line stream.
tbi_rx_d(9:0)	In	TBI receive data, expected to receive from the external SERDES synchronously with the signal tbi_rx_clk. Data from the external SERDES can be arbitrary aligned.

Phy Management Signals

For more information about PHY Management signals, refer to “[PHY Management Signals](#)” on page 4-87.

Status LED Control Signals

Table 4–44 lists the status LED control signals.

Table 4–44. Status LED Interface Signals		
Signal Name	Direction	Description
led_link	Out	Set to 1 to indicate a successful link synchronization.
led_crs	Out	Set to 1 to indicate activity on the transmit and receive path (carrier sense). When set to 0, no traffic is detected on neither transmit nor receive path.
led_col	Out	Set to 1 to indicate that a collision was detected during a frame transfer. Always set to 0 when the PCS function operates in Standard 1000BASE-X mode or in full-duplex mode when SGMII is enabled.
led_an	Out	Auto-negotiation status. Set to 0 when auto-negotiation is completed.
led_char_err	Out	10-bit character error. Asserted for 1 tbi_rx_clk cycle when a wrong 10-bit character is detected.
led_disp_err	Out	10-bit running disparity error. Asserted for 1 tbi_rx_clk cycle when a disparity error is detected. A running disparity error indicates that more than the previous and perhaps the current received group had an error.

SERDES Control Signals

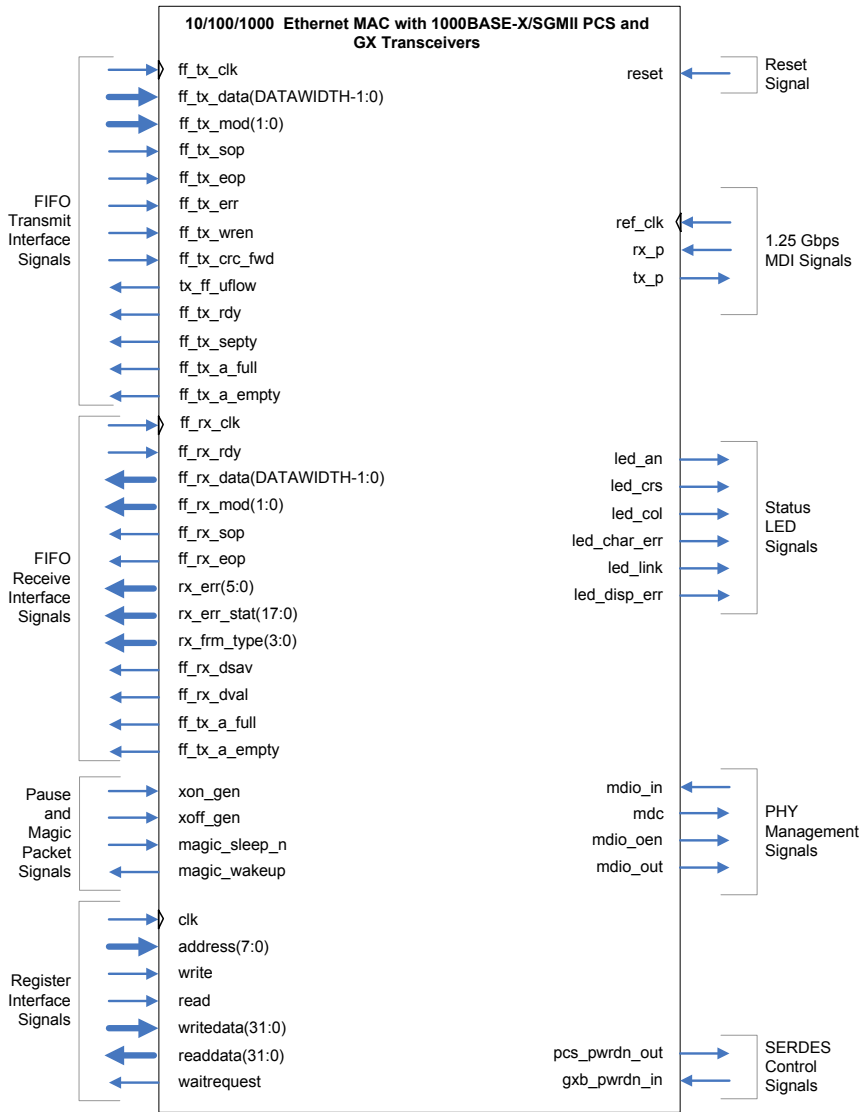
Table 4–45 describes the functionality of the SERDES control signals.

Table 4–45. SERDES Control Signal		
Signal Name	Direction	Description
powerdown	Out	Power-down enable. Set to 1 when the PCS function is configured by the system management agent to operate in power-down mode. Set to 0 when the PCS function operates in normal mode. Only implemented when an external SERDES is used.
sd_loopback	Out	SERDES Loopback Control. Set to 1 when the PCS function is configured by the system management agent to operate in loopback mode. This signal can be used to configure an external SERDES device to operate in loopback mode.

10/100/1000 Ethernet MAC with 1000BASE-X/SGMII PCS and PMA Signals

Figure 4–47 shows all I/O signals of the 10/100/1000 Ethernet MAC with 1000BASE-X/SGMII PCS Function and GX Serial Transceiver PMA.

Figure 4–47. 10/100/1000 Ethernet MAC with 1000BASE-X/SGMII PCS Function and GX Transceiver Signals



Control Interface Signals

For more information about control interface signals, refer to “[Control Interface Signals](#)” on page 4-78.

MAC System-Side Signals

For more information about MAC system-side signals, refer to “[MAC System-Side Signals](#)” on page 4-79.

PCS Ethernet-Side Signals

1.25 Gbps MDI Interface

If the variant includes GX transceivers, the transceivers provide a 1.25 GHz medium-dependent interface (MDI). [Table 4-46](#) lists the MDI signals

Table 4-46. 1.25 Gbps MDI Interface Signals

Signal Name	Dir	Description
ref_clk	In	125 MHz local reference clock oscillator.
rx_p	In	Serial Differential Receive Interface.
tx_p	Out	Serial Differential Transmit Interface.

PHY Management Signals

For more information about PHY Management Signals, refer to “[PHY Management Signals](#)” on page 4-87.

Status LED Control Signals

For more information about Status LED Control Signals, refer to “[Status LED Control Signals](#)” on page 4-91.

SERDES Control Signals

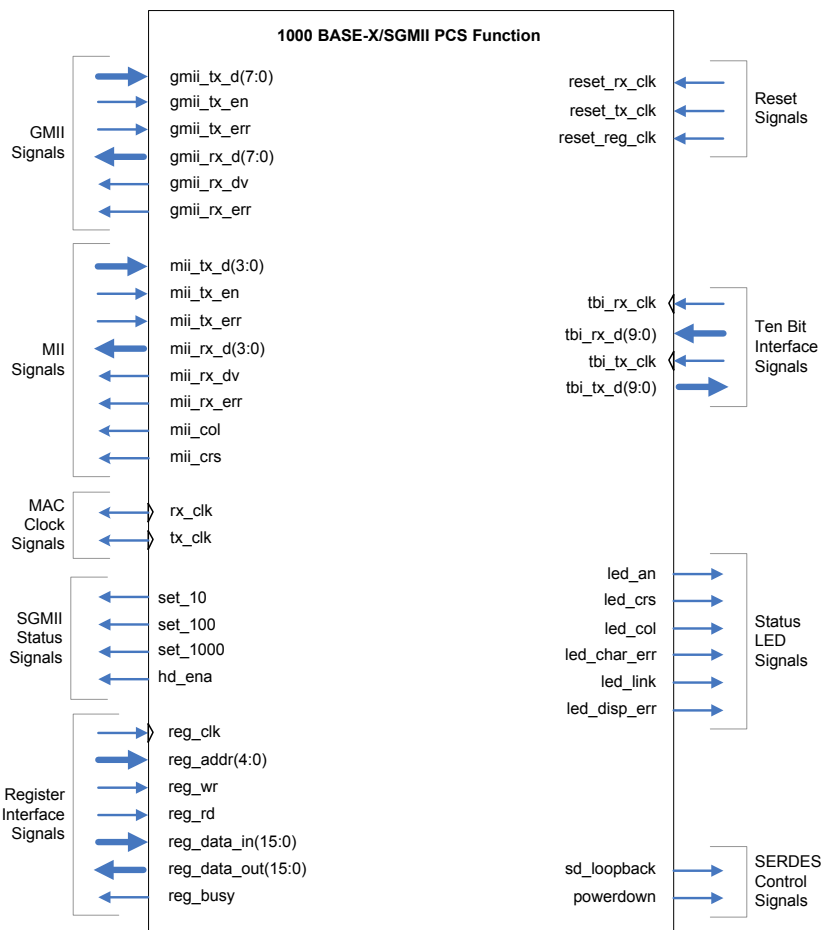
Table 4–47 describes the functionality of the SERDES control signals.

Table 4–47. SERDES Control Signal		
Signal Name	Direction	Description
pcs_pwrdn_out	Out	Power-down enable output. Set to 1 when the PCS function is configured by the system management agent to operate in power-down mode. Set to 0 when the PCS function operates in normal mode. Only implemented when an internal SERDES is used with the option to export the power-down signal.
gxb_pwrdn_in	In	Power-down enable input. Powers down the transceiver quad block when set to 1. Only implemented when an internal SERDES is used with the option to export the power-down signal.

1000BASE-X/SGMII PCS Signals

Figure 4–48 shows all I/O signals of the 1000BASE-X/SGMII PCS function.

Figure 4–48. 1000BASE-X/SGMII PCS Function Signals



Control Interface Signals

Register Interface Signals

Table 4–48 describes the signals that comprise the register interface for the PCS function.

Table 4–48. Register Interface Signals			
Signal Name	Avalon-MM Signal Type	Direction	Description
reg_clk	clk	In	Register access reference clock.
reset_reg_clk	reset	In	Active high reset signal for reg_clk clock domain.
reg_wr	write	In	Register write enable.
reg_rd	read	In	Register read enable.
reg_addr(4:0)	address	In	Register address.
reg_data_in(15:0)	writedata	In	Register write data. Bit 0 is the least significant bit.
reg_data_out(15:0)	readdata	Out	Register read data. Bit 0 is the least significant bit.
reg_busy	waitrequest	Out	Register interface busy. Asserted during register read or register write. Set to 0 to indicate the read or write completion.



For Information About	Refer To
Avalon Memory-Mapped interface protocol	<i>Avalon Memory-Mapped Interface Specification</i>

Reset Signals

Table 4–33 lists all reset signals which you can use to reset the PCS function.

Table 4–49. Reset Signals		
Signal Name	Dir	Description
reset_rx_clk	In	Active high reset signal for PCS rx_clk clock domain. Resets the logic synchronized by the clock rx_clk.
reset_tx_clk	In	Active high reset signal for PCS tx_clk clock domain. Resets the logic synchronized by the clock tx_clk.

PCS MAC-Side Signals

The MAC-side interface for the PCS function uses the standard MII and GMII interfaces to connect to an external MAC device or to the internal MAC block.

MAC Clocks

Data transfers on the PCS MAC interface are synchronous to the receive and transmit clocks. [Table 4–50](#) describes these clock signals.

Table 4–50. MAC Clock Signals

Signal Name	Dir	Description
rx_clk	Out	Receive MAC Clock. Derived from the TBI Clock <code>tbi_rx_clk</code> and set to the following frequency: <ul style="list-style-type: none"> 125 MHz when the PCS operates in gigabit or SGMII mode 25 MHz when the PCS operate in 100 Mbps mode 2.5 MHz when the when the PCS operate in 10 Mbps mode
tx_clk	Out	Transmit MAC Clock. Derived from the TBI Clock <code>tbi_tx_clk</code> and set to the following frequency: <ul style="list-style-type: none"> 125 MHz when the PCS operates in gigabit or SGMII mode 25 MHz when the PCS operate in 100 Mbps mode 2.5 MHz when the when the PCS operate in 10 Mbps mode

GMII Interface

[Table 4–51](#) describes the GMII transmit and receive signals.

Table 4–51. GMII Interface Signals

Signal Name	Dir	Description
GMII Transmit Interface		
gmii_tx_d(7:0)	In	GMII transmit data bus.
gmii_tx_en	In	Asserted to indicate data on the data bus, <code>gmii_tx_d(7:0)</code> is valid.
gmii_tx_err	In	Asserted to indicate to the PHY device that the current frame sent is invalid.
GMII Receive Interface		
gmii_rx_d(7:0)	Out	GMII receive data bus.
gmii_rx_dv	Out	Asserted to indicate data on the receive data bus is valid. Stays asserted during frame reception, from the first preamble byte until the last byte in the CRC field is received.
gmii_rx_err	Out	Asserted by the PHY to indicate that the current frame contains erroneous data.

MII Interface

Table 4–52 describes the MII transmit and receive signals.

Table 4–52. MII Interface Signals		
Signal Name	Dir	Description
MII Transmit Interface		
<code>mii_tx_d(3:0)</code>	In	MII transmit data bus.
<code>mii_tx_en</code>	In	Asserted to indicate data on the data bus, <code>mii_tx_d(3:0)</code> is valid.
<code>mii_tx_err</code>	In	Asserted to indicate to the PHY device that the current frame sent is invalid.
MII Receive Interface		
<code>mii_rx_d(3:0)</code>	Out	MII receive data bus.
<code>mii_rx_dv</code>	Out	Asserted to indicate data on the receive data bus is valid. Stays asserted during frame reception, from the first preamble byte until the last byte of the CRC field is received.
<code>mii_rx_err</code>	Out	Asserted by the PHY to indicate that the current frame contains erroneous data.
<code>mii_col</code>	Out	Collision detection. Asserted by the PCS function to indicate that a collision was detected during a frame transfer.
<code>mii_crs</code>	Out	Carrier sense detection. Asserted by the PCS function to indicate that transmit or receive activity is detected on the Ethernet line.

SGMII Status Signals

The SGMII status signals provide status information to the PCS block. When the PCS is instantiated standalone, these signals are inputs to the MAC and serve as interface control signals for that block. Table 4–53 lists the SGMII status signals.

Table 4–53. SGMII Status Signals		
Signal Name	Dir	Description
<code>set_1000</code>	Out	Gigabit mode enabled. In 1000BASE-X, this signal is always set to 1. In SGMII, this signal is set to 1 if one of the following conditions is met: <ul style="list-style-type: none"> if the <code>USE_SGMII_AN</code> bit is set to 1 and a gigabit link is established with the link partner, as decoded from the <code>partner_ability</code> register if the <code>USE_SGMII_AN</code> bit is set to 0 and the <code>SGMII_SPEED</code> bit is set to 10
<code>set_100</code>	Out	100 Mbps mode enabled. In 1000BASE-X, this signal is always set to 0. In SGMII, this signal is set to 1 if one of the following conditions is met: <ul style="list-style-type: none"> if the <code>USE_SGMII_AN</code> bit is set to 1 and a 100Mbps link is established with the link partner, as decoded from the <code>partner_ability</code> register if the <code>USE_SGMII_AN</code> bit is set to 0 and the <code>SGMII_SPEED</code> bit is set to 01

Table 4–53. SGMII Status Signals

Signal Name	Dir	Description
set_10	Out	10 Mbps mode enabled. In 1000BASE-X, this signal is always set to 0. In SGMII, this signal is set to 1 if one of the following conditions is met: <ul style="list-style-type: none"> • if the USE_SGMII_AN bit is set to 1 and a 10Mbps link is established with the link partner, as decoded from the <code>partner_ability</code> register • if the USE_SGMII_AN bit is set to 0 and the SGMII_SPEED bit is set to 00
hd_ena	Out	Half duplex mode enabled. In 1000BASE-X, this signal is always set to 0. In SGMII, this signal is set to 1 if one of the following conditions is met: <ul style="list-style-type: none"> • if the USE_SGMII_AN bit is set to 1 and a half-duplex link is established with the link partner, as decoded from the <code>partner_ability</code> register • if the USE_SGMII_AN bit is set to 0 and the SGMII_DUPLEX bit is set to 01

PCS Ethernet-Side Signals

TBI Interface Signals for External SERDES Chip

For more information about TBI Interface signals, refer to “[TBI Interface Signals for External SERDES Chip](#)” on page 4–90.

Status LED Control Signals

For more information about Status LED signals, refer to “[Status LED Control Signals](#)” on page 4–91.

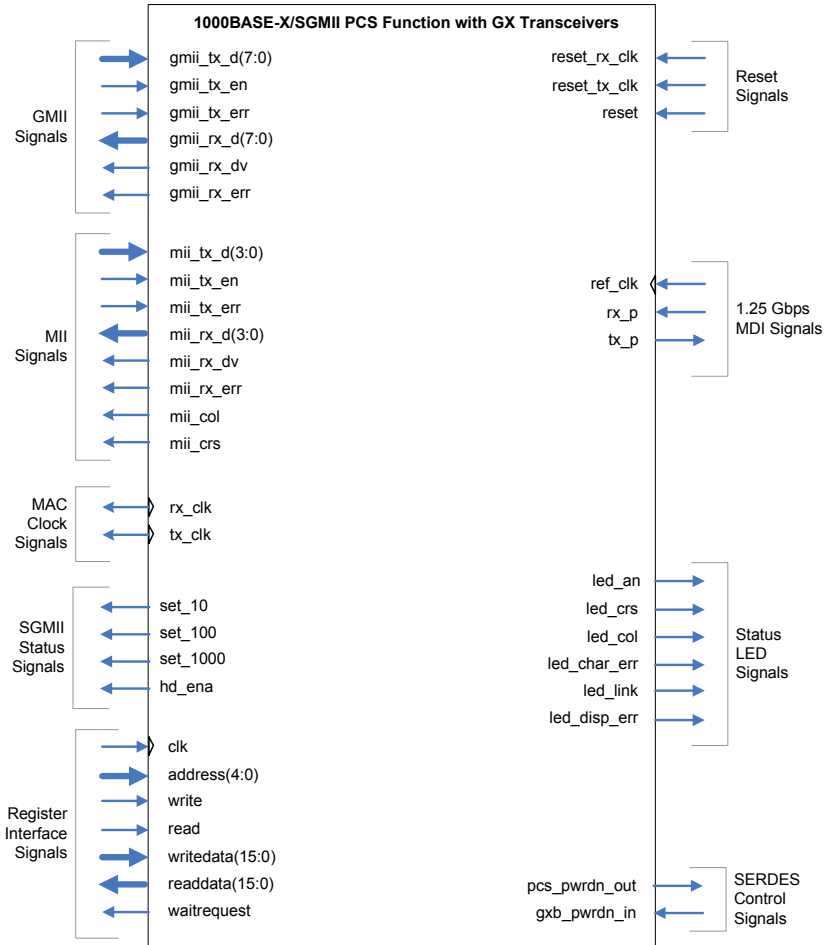
SERDES Control Signals

For more information about SERDES Control signals, refer to “[SERDES Control Signals](#)” on page 4–91.

1000BASE-X/SGMII PCS and PMA Signals

Figure 4-49 shows all I/O signals of the 1000BASE-X/SGMII PCS Function with GX Transceiver PMA.

Figure 4-49. 1000BASE-X/SGMII PCS Function and PMA Signals



Control Interface Signals

For more information about control interface signals, refer to “Control Interface Signals” on page 4-78.

PCS MAC-Side Signals

For more information about PCS MAC-Side Signals, refer to “[PCS MAC-Side Signals](#)” on page 4-97.

PCS Ethernet-Side Signals

1.25 Gbps MDI Signals

For more information about 1.25 Gbps MDI signals, refer to “[1.25 Gbps MDI Interface](#)” on page 4-93.

Status LED Control Signals

For more information about Status LED Control signals, refer to “[Status LED Control Signals](#)” on page 4-91.

SERDES Control Signals

For more information about SERDES Control signals, refer to “[SERDES Control Signals](#)” on page 4-94.

Introduction

You can use the testbench provided with the Triple Speed Ethernet MegaCore function to exercise your custom MegaCore function variation. The testbench includes the following features:

- Easy-to-use simulation environment for any standard HDL simulator
- Simulation of all basic Ethernet packet transactions
- Open source Verilog HDL and VHDL testbench files

The provided testbench applies only to custom MegaCore function variations created using the MegaWizard Plug-in Manager flow.

For more information on the testbench files generated and step-by-step instructions for simulating your design using the ModelSim simulator or other simulators, refer to the Getting Started chapter.

Testbench Specifications

The Triple Speed Ethernet testbench comprises the following modules:

- Device under test (DUT)—your custom MegaCore function variation
- Avalon Streaming (Avalon-ST) Ethernet frame generator
- Avalon-ST Ethernet frame monitor
- MII/RGMII/GMII Ethernet frame generator
- MII/RGMII/GMII Ethernet frame monitor
- MDIO slaves
- Clock and reset generator

Avalon-ST Ethernet Frame Generator

The Avalon-ST Ethernet frame generator simulates a user application connected to the MAC transmit FIFO. It generates frames on the Avalon-ST transmit interface.

Avalon-ST Ethernet Frame Monitor

The Avalon-ST Ethernet frame monitor simulates a user application receiving frames from the MAC receive FIFO. It monitors the Avalon-ST receive interface and decodes all data received from the MAC receive FIFO.

MII/RGMII/GMII Ethernet Frame Generator

If the device under test is a MAC function, the MII/RGMII/GMII Ethernet frame generator simulates a PHY device that sends frames to the MAC function.

If the device under test is a PCS function, the MII/RGMII/GMII Ethernet frame generator simulates a MAC function that sends frames to the PCS function.

MII/RGMII/GMII Ethernet Frame Monitor

If the device under test is a MAC function, the MII/RGMII/GMII Ethernet frame monitor simulates a PHY device that receives frames from the MAC function and decodes them.

If the device under test is a PCS function, the MII/RGMII/GMII Ethernet frame monitor simulates a MAC function that receives frames from the PCS function and decodes them.

MDIO Slave

The MDIO slave module simulates a PHY management interface. It responds to a MDIO master transactor.

Configuration

You can configure separate datapaths in the testbench, or implement a loopback connection between transmit and receive on the MII/GMII/RGMII interface.

Separate datapaths are enabled when the MII/RGMII/GMII Ethernet frame generator is enabled. In this configuration, the testbench control block simulates independent yet complete receive and transmit datapaths.

Loopback is automatically implemented when the MII/RGMII/GMII Ethernet frame generator is disabled (`tb_rxframes` is set to 0). In this configuration, the frames sent through the transmit interface are looped back into the receive interface.

You can also customize other aspects of the testbench using the testbench simulation parameters. For more information on the testbench simulation parameters, refer to [Appendix A, Simulation Parameters](#).

Verification

The testbench is self-checking and determines if a simulation is successful by verifying the frames received. It also checks for any errors detected by the frame monitors.

The testbench does not verify the IEEE statistics generated by the MAC layer. Simulation fails only if the testbench is not able to detect deliberately inserted errors.

At the end of a simulation, the testbench displays messages in the simulator console indicating its results.

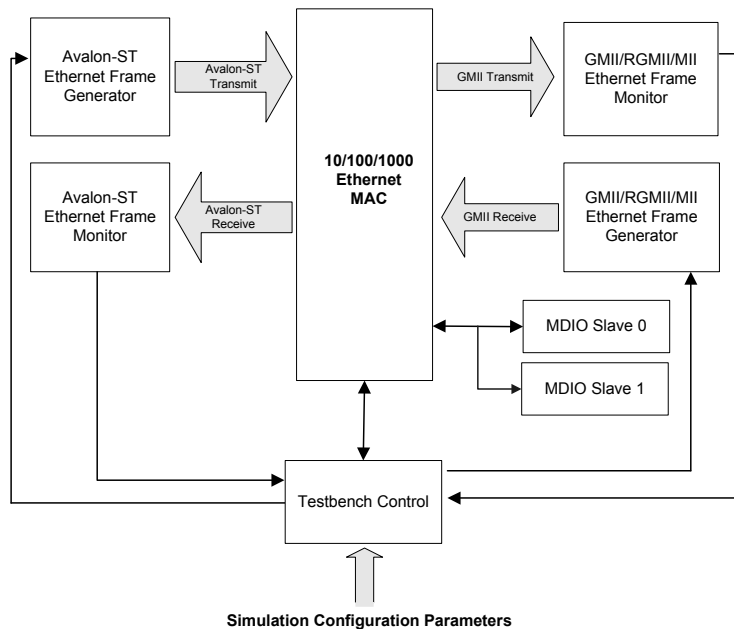
Testbench Architecture

This section describes the testbench architecture for each possible Triple Speed Ethernet MegaCore configuration.

10/100/1000 Ethernet MAC

Table 5–1 illustrates the 10/100/1000 Ethernet MAC testbench architecture.

Figure 5–1. 10/100/1000 Ethernet MAC Testbench Architecture



Overview

The testbench demonstrates and verifies the following MAC functionality:

- Transmit and receive datapaths are functionally correct.
- Ethernet frames of valid length (greater than 60 bytes) received on the Avalon-ST transmit interface are sent out through the GMII transmit interface with correct CRC-32 inserted by the MAC function. Short frames are always padded by the MAC function up to at least 64 bytes in length.

- Untagged received frames of size greater than the maximum frame length are truncated to the maximum frame length with additional bytes up to 12.
- The CRC-32 is optionally discarded before forwarding the frames onto the Avalon-ST receive interface.


Default Testbench Configuration

The following settings apply to the default MAC testbench configuration:

- The simulated link speed is assumed to be Gigabit.
- The `command_config` register is set to 0x0008022B.
- The maximum frame length, register `frm_length`, is configured to 1518.
- The transmit FIFO is set to start data transmission as soon as the FIFO level reaches `tx_section_full`.
- The receive FIFO begins forwarding the Ethernet frame data received from the GMII receive to the Avalon-ST receive when the FIFO level reaches `rx_section_full`.
- The receive and transmit datapaths are simulated independently.
- Five Ethernet frames of payload length 100, 101, 102, 103 and 104 bytes are sent to the Avalon-ST transmit interface as well as the GMII receive interface without errors.
- The MAC function is configured to forward all received frames that are not filtered based on the destination address of the received frames.

Test Flow

The MAC testbench performs the following operations upon a simulated power-on reset:

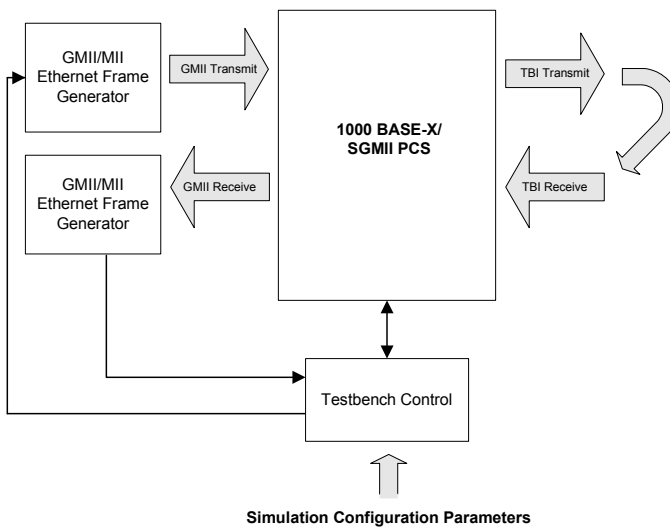
- Initialize the MAC, which consists the following operations:
 - Set the MAC operation mode via the `command_config` register.
 - Set the MAC address via the `mac_0` and `mac_1` registers.
 - Set the interpacket gap for transmit frames via the `tx_ipg_length` register.
 - Set the maximum frame length that can be received by the MAC via the `frm_length` register.
 - Set the pause frames quanta via the `pause_quant` register.
 - Set the MAC Avalon-ST FIFO threshold registers.
- Write to MDIO slave address and command registers to exercise the MDIO slave.
 -  This is done only if MDIO ports are in use by setting the parameter `TB_MDIO_SIMULATION` to 1.

- Set the MAC multicast hash table for multicast address filtering.
- Set the MAC supplemental unicast addresses.
- Start transmission and clear receive and transmit FIFO.
- End transmission and read the MAC statistics registers, if they are implemented.
- Compare the statistics generated by the frame generator and monitor to determine if the simulation is successful.

1000BASE-X/SGMII PCS

Figure 5–2 illustrates the 1000BASE-X/SGMII PCS testbench architecture.

Figure 5–2. 1000BASE-X/SGMII PCS



Overview

The testbench demonstrates and verifies the following PCS functionality:

- Transmit and receive datapaths are functionally correct at 1.25Gbps via a loopback implementation at the Ten Bit Interface (TBI)
- Ethernet frames received on the GMII transmit interface are sent out through the TBI transmit interface and correctly encapsulated.
- On the receiving end, the PCS function is able to receive frames from the TBI receive interface and de-encapsulates the frames before forwarding them in GMII format onto the GMII receive interface.

Default Testbench Configuration

The following settings apply to the default PCS testbench configuration:

- The simulated link speed is assumed to be Gigabit.
- The `if_mode` register is programmed to 0x0000.
- Auto-negotiation between the local PHY and remote link PHY is bypassed.

- The testbench is configured to simulate the receive and transmit datapaths in a loopback mode by connecting the TBI transmit to the TBI receive.
- Five Ethernet frames of payload length 100, 101, 102, 103 and 104 bytes are sent to the transmit GMII without errors.

Test Flow

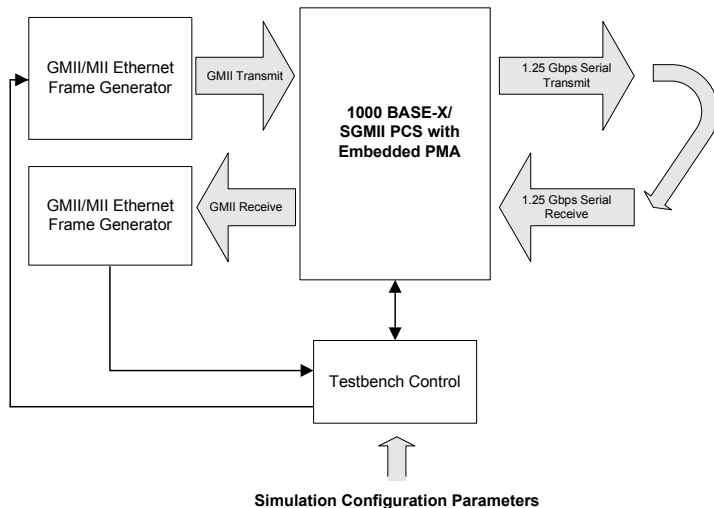
The PCS testbench performs the following operations upon a simulated power-on reset:

- Initialize the PCS function, which consists the following operations:
 - Set the PCS operation mode via the `if_mode` register.
 - Set the PCS auto-negotiation to BYPASS via the `control` register.
- Start transmission.
- End transmission.
- Check the following elements to determine if the simulation is successful:
 - The Ethernet frames sent to the GMII transmit are the same as the Ethernet frames received from the GMII receive.
 - No Ethernet protocol errors detected.

1000BASE-X/SGMII PCS with embedded PMA

Figure 5–3 illustrates the 1000BASE-X/SGMII PCS with embedded PMA testbench architecture.

Figure 5–3. 1000BASE-X/SGMII PCS with Embedded PMA Testbench Architecture



Overview

The testbench demonstrates and verifies the following PCS function with an embedded PMA functionality:

- Transmit and receive datapaths are functionally correct via a loopback implementation at the 1.25 Gbps serial interface.
- Ethernet frames received on the GMII transmit interface are sent out through the 1.25 Gbps transmit serial interface and correctly encapsulated.
- On the receiving end, the PCS function is able to receive frames from the 1.25 Gbps receive serial interface and de-encapsulate the frames before forwarding them in GMII format onto the GMII receive interface.

Default Testbench Configuration

The following settings apply to the default PCS with an embedded PMA testbench configuration:

- The simulated physical link speed is assumed to be Gigabit.
- The `if_mode` register is programmed to 0x0000.
- Auto-negotiation between the local PHY and remote link PHY is bypassed.
- The testbench is configured to simulate the receive and transmit datapaths in a loopback mode by connecting the 1.25 Gbps transmit serial interface to the 1.25 Gbps receive serial interface.
- Five Ethernet frames of payload length 100, 101, 102, 103 and 104 bytes are sent to the transmit GMII without errors.

Test Flow

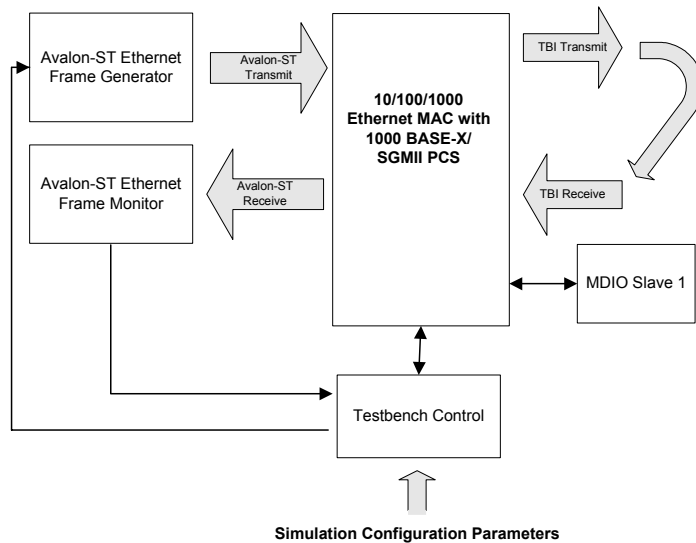
The PCS with an embedded PMA testbench performs the following operations upon a simulated power-on reset:

- Initialize the PCS function, which consists the following operations:
 - Set the PCS operation mode via the `if_mode` register.
 - Set the PCS auto-negotiation to BYPASS via the `control` register.
- Start transmission.
- End transmission.
- Check the following elements to determine if the simulation is successful:
 - The Ethernet frames sent to the GMII transmit are the same as the Ethernet frames received from the GMII receive.
 - No Ethernet protocol errors detected.

10/100/1000 Ethernet MAC with 1000BASE-X/SGMII PCS

Figure 5–4 illustrates the 10/100/1000 Ethernet MAC with 1000BASE-X/SGMII PCS testbench architecture.

Figure 5–4. 10/100/1000 Ethernet MAC with 1000BASE-X SGMII PCS Testbench Architecture



Overview

The testbench demonstrates and verifies the following MAC and PCS functionality:

- Transmit and receive datapaths are functionally correct via a loopback implementation at the TBI interface.
- Ethernet frames of valid length (greater than 60 bytes) received on the Avalon-ST Transmit interface are sent out through the TBI transmit and correctly encapsulated. Short Ethernet frames are always padded by the MAC function up to at least 64 bytes in length.
- On the receiving end, the PCS function is able to receive frames on the TBI receive, de-encapsulate the frames and optionally discard the CRC-32 before forwarding the frames onto the Avalon-ST receive interface.


Default Testbench Configuration

The following settings apply to the default MAC and PCS testbench configuration:

- The simulated physical link speed is assumed to be Gigabit.
- The MAC `command_config` register is programmed to 0x0008022B.
- The PCS `if_mode` register is programmed to 0x0000.
- The maximum frame length, register `frm_length`, is programmed to 1518.
- The MAC transmit FIFO is set to start frame transmission as soon as the FIFO level reaches `tx_section_full`.
- The MAC receive FIFO begins passing the Ethernet packet data received to the system side when the FIFO level reaches `rx_section_full`.
- The testbench is configured to simulate the receive and transmit data paths in a loopback mode by connecting the PCS TBI transmit to the PCS TBI receive.
- Five Ethernet frames of payload length 100, 101, 102, 103 and 104 bytes are sent to the Avalon-ST transmit interface.
- The MAC function is programmed to forward all received frames that are not filtered based on the destination address of the received frames.

Test Flow

The MAC and PCS testbench performs the following operations upon a simulated power-on reset:

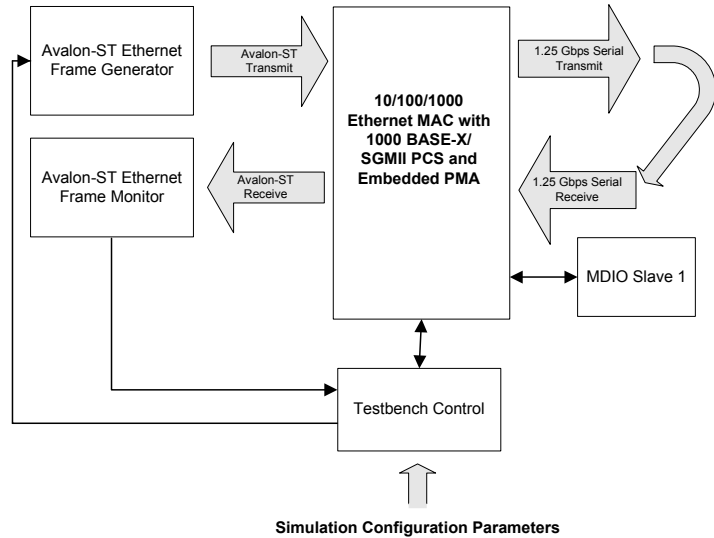
- Initialize the MAC and PCS functions, which consists the following operations:
 - Set the MAC operation mode via the `command_config` register.
 - Set the PCS operation mode via the `if_mode` register.
 - Set the PCS auto-negotiation to BYPASS via the `control` register.
 - Set the MAC address via the `mac_0` and `mac_1` registers.
 - Set the interpacket gap for transmit frames via the `tx_ipg_length` register.
 - Set the maximum frame length that can be received by the MAC via the `frm_length` register.
 - Set the pause frames quanta via the `pause_quant` register.
 - Set the Avalon-ST FIFO threshold registers.
- Write to MDIO slave address and command registers to exercise the MDIO slave.
 -  This is done only if MDIO ports are in use by setting the parameter `TB_MDIO_SIMULATION` to 1.

- Set the MAC multicast hash table for multicast address filtering.
- Set the MAC supplemental unicast addresses.
- Start transmission and clear receive and transmit FIFO.
- End transmission and read the MAC statistics registers, if they are implemented.
- Compare the statistics generated by the frame generator and monitor to determine if the simulation is successful.

10/100/1000 Ethernet MAC with 1000BASE-X/SGMII PCS and Embedded PMA

Figure 5–5 illustrates the 10/100/1000 Ethernet MAC with 1000BASE-X/SGMII PCS and an embedded PMA testbench architecture.

Figure 5–5. 10/100/1000 Ethernet MAC with 1000BASE-X SGMII PCS and Embedded PMA Testbench Architecture



Overview

The testbench demonstrates and verifies the following MAC and PCS function with an embedded PMA functionality:

- Transmit and receive datapaths are functionally correct via a loopback implementation at the 1.25 Gbps serial interface.
- Ethernet frames of valid length (greater than 60 bytes) received on the Avalon-ST transmit interface are sent out through the 1.25 serial transmit interface and correctly encapsulated. Short Ethernet frames are always padded by the MAC function up to at least 64 bytes in length.
- On the receiving end, the PCS function is able to receive frames on the 1.25 Gbps serial receive interface, de-encapsulate the frames and discard the CRC-32 before forwarding the frames onto the Avalon-ST receive interface.

Default Testbench Configuration

The following settings apply to the default MAC and PCS with an embedded PMA testbench configuration:

- The simulated link speed is assumed to be Gigabit.
- The MAC `command_config` register is programmed to 0x0008022B.
- The PCS `if_mode` register is programmed to 0x0000.
- The maximum frame length, register `frm_length`, is programmed to 1518.
- The MAC transmit FIFO is set to start frame transmission as soon as the FIFO level reaches `tx_section_full`.
- The MAC receive FIFO begins passing the Ethernet packet data received to the system side when the FIFO level reaches `rx_section_full`.
- The testbench is configured to simulate the receive and transmit datapaths in a loopback mode by connecting the 1.25 Gbps transmit interface to the 1.25 Gbps receive interface.
- Five Ethernet frames of payload length 100, 101, 102, 103 and 104 bytes are sent to the Avalon-ST transmit interface.
- The MAC function is programmed to forward all received frames that are not filtered based on the destination address of the received frames.

Test Flow

The MAC and PCS with an embedded PMA testbench is designed to perform the following operations upon a simulated power-on reset:

- Initialize the MAC and PCS, which consists the following operations:
 - Set the MAC operation mode via the `command_config` register.
 - Set the PCS operation mode via the `if_mode` register.
 - Set the PCS auto-negotiation to BYPASS via the `control` register.
 - Set the MAC address via the `mac_0` and `mac_1` registers.
 - Set the interpacket gap for transmit frames via the `tx_ipg_length` register.
 - Set the maximum frame length that can be received by the MAC via the `frm_length` register.
 - Set the pause frames quanta via the `pause_quant` register.
 - Set the Avalon-ST FIFO threshold registers.
- Write to MDIO slave address and command registers to exercise the MDIO slave.



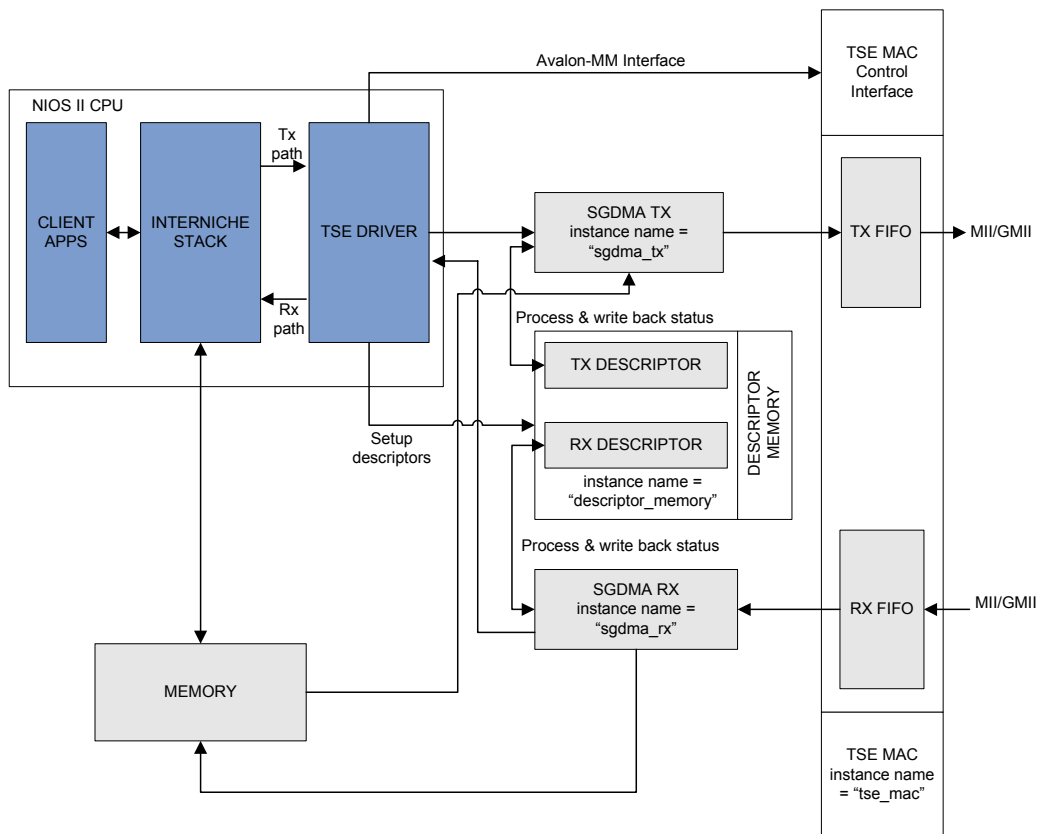
This is done only if MDIO ports are in use by setting the parameter `TB_MDIO_SIMULATION` to 1.

- Set the MAC multicast hash table for multicast address filtering.
- Set the MAC supplemental unicast addresses.
- Start transmission and clear receive and transmit FIFO.
- End transmission and read the MAC statistics registers, if they are implemented.
- Compare the statistics generated by the frame generator and monitor to determine if the simulation is successful.

Triple Speed Ethernet Driver Architecture

Figure 6–1 illustrates the architecture of the Triple Speed Ethernet software driver.

Figure 6–1. Triple Speed Ethernet Software Driver Architecture





The first 128 bytes of the on-chip RAM are reserved for SGDMA RX and TX descriptors. Applications must not use this memory region.

You must specify the Triple Speed Ethernet device instance as `tse_mac`. The device instance names for SGDMA RX and SGDMA TX are `sgdma_rx` and `sgdma_tx`, respectively.

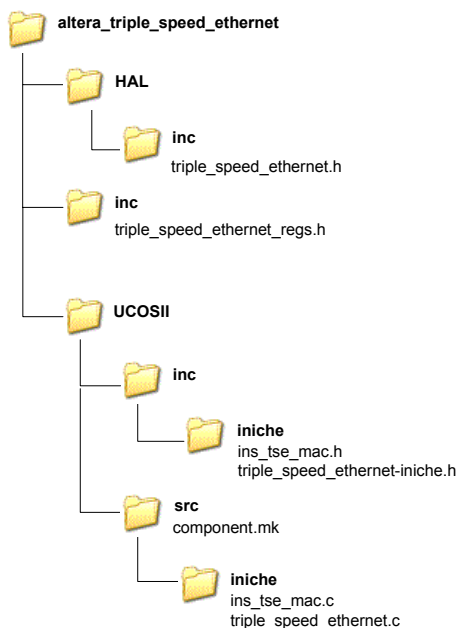
At present, the Triple Speed Ethernet software driver does not support the following features:

- Multiple instances of Triple Speed Ethernet MAC.
- Jumbo ethernet frames.
- 8-bit FIFO interface. You must set the FIFO data width to 32 bits if you intend to use the provided software driver. For more information on configuring the FIFO data width, refer to “[FIFO Options](#)” on page 3–6.
- PHYs other than National DP83865 (10/100/1000) and Marvell 88E1111 (10/100/1000).

Directory Structure

Figure 6–2 shows the directory structure of the `altera_triple_speed_ethernet` directory, and the relevant header and source files it contains.

Figure 6–2. Directory Structure



API Functions

This section describes each provided API function in alphabetical order.

triple_speed_ethernet_init()

Prototype:

```
error_t triple_speed_ethernet_init(alt_niche_dev
*p_dev)
```

Thread-safe:

Yes

Available from ISR:

No

Include:

<triple_speed_ethernet_iniche.h>

Description:

The `triple_speed_ethernet_init()` function opens and initializes the Triple Speed Ethernet driver. Initialization involves the following operations:

- Set up the NET structure of the MAC device instance.
- Configure the MAC PHY Address.
- Register and open the SGDMA RX and TX Module of the MAC device instance.
- Enable the SGDMA RX interrupt and register it to the Operating System.
- Register the SGDMA RX callback function.
- Obtains the PHY Speed of the MAC.
- Set up the Ethernet MAC Register settings for the Triple Speed Ethernet driver operation.
- Set up the initial descriptor chain to start the SGDMA RX operation.

Argument

`p_dev` A pointer to the Triple Speed Ethernet device instance.

Return:

SUCCESS if the Triple Speed Ethernet driver is successfully initialized.

See Also:

`tse_mac_close()`

tse_mac_close()

Protytpe:

```
int tse_mac_close(int iface)
```

Thread-safe:

Yes

Available from ISR:

No

Include:

```
<triple_speed_ethernet_iniche.h>
```

Description:

The `tse_mac_close()` closes the Triple Speed Ethernet driver by performing the following operations:

- Configure the admin and operation status of the NET structure of the Triple Speed Ethernet driver instance to `ALTERA_TSE_ADMIN_STATUS_DOWN`.
- De-register the SGDMA RX interrupt from the operating system.
- Clear the `RX_ENA` bit in the `command_config` register to disable the RX datapath.

Argument

None

Return:

`SUCCESS` if the close operations are successful.

An error code if de-registration of SGDMA RX from the operating system failed.

See Also:

```
triple_speed_ethernet_init()
```

tse_mac_raw_send()

Prototype:

```
int tse_mac_raw_send(NET net, char *data, unsigned
data_bytes)
```

Thread-safe:

Yes

Available from ISR:

Yes

Include:

<triple_speed_ethernet_iniche.h>

Description:

The `tse_mac_raw_send()` function sends Ethernet frames data to the MAC function. It validates the arguments to ensure the data length is greater than the ethernet header size specified by `ALTERA_TSE_MIN_MTU_SIZE`. The function also ensures the SGDMA TX engine is not busy prior to constructing the descriptor for the current transmit operation.

Upon successful validations, this function calls the internal API, `tse_mac_sTxWrite`, to initiate the synchronous SGDMA transmit operation on the current data buffer.

Argument

<code>net</code>	The NET structure of the Triple Speed Ethernet MAC instance.
<code>data</code>	A data pointer to the base of the Ethernet frame data, including the header, to be transmitted to the MAC. The data pointer is assumed to be word-aligned
<code>data_bytes</code>	The total number of bytes in the Ethernet frame including the additional padding bytes as specified by <code>ETHHDR_BIAS</code> .

Return:

`SUCCESS` if the current data buffer is successfully transmitted.

`SEND_DROPPED` if the number of data bytes is less than the Ethernet header size.

`ENP_RESOURCE` if the SGDMA TX engine is busy.

tse_mac_setGMII mode()

Prototype:

```
int tse_mac_setGMII mode(np_tse_mac *pmac)
```

Thread-safe:

Yes

Available from ISR:

Yes

Include:

```
<triple_speed_ethernet_iniche.h>
```

Description:

The `tse_mac_setGMII mode()` function sets the MAC function operation mode to Gigabit (GMII). The settings of the `command_config` register are restored at the end of the function.

Argument:

`pmac` A pointer to the MAC control interface base address.

Return:

SUCCESS

See Also:

```
tse_mac_setMIImode()
```

tse_mac_setMIImode()

Prototype:

```
int tse_mac_setGMIImode(np_tse_mac *pmac)
```

Thread-safe:

Yes

Available from ISR:

Yes

Include:

```
<triple_speed_ethernet_iniche.h>
```

Description:

The `tse_mac_setMIImode()` function sets the MAC function operation mode to MII (10/100). The settings of the `command_config` register are restored at the end of the function.

Argument:

`pmac` A pointer to the MAC control interface base address.

Return:

SUCCESS

See Also:

```
tse_mac_setGMIImode()
```


tse_mac_SwReset()

Prototype:

```
int tse_mac_SwReset(np_tse_mac *pmac)
```

Thread-safe:

Yes

Available from ISR:

Yes

Include:

<triple_speed_ethernet_iniche.h>

Description:

The `tse_mac_SwReset()` performs a software reset on the MAC function. A software reset occurs with some latency as specified by `ALTERA_TSE_SW_RESET_TIME_OUT_CNT`. The settings of the `command_config` register are restored at the end of the function.

Argument:

`pmac` A pointer to the MAC control interface base address.

Return:

SUCCESS

See Also:

`tse_mac_setGMIImode()`

Constants

Table 6–1 lists all constants defined for the MAC registers manipulation and provides links to detailed descriptions of the registers.

Table 6–1. Constants Mapping (Part 1 of 2)

Constant	Value	Description
Command_Config Register (Table 4–15 on page 4–39)		
ALTERA_TSEMAC_CMD_TX_ENA_OFST	0	Configures the TX_ENA bit.
ALTERA_TSEMAC_CMD_TX_ENA_MSK	0x1	
ALTERA_TSEMAC_CMD_RX_ENA_OFST	1	Configures the RX_ENA bit.
ALTERA_TSEMAC_CMD_RX_ENA_MSK	0x2	
ALTERA_TSEMAC_CMD_XON_GEN_OFST	2	Configures the XON_GEN bit.
ALTERA_TSEMAC_CMD_XON_GEN_MSK	0x4	
ALTERA_TSEMAC_CMD_ETH_SPEED_OFST	3	Configures the ETH_SPEED bit.
ALTERA_TSEMAC_CMD_ETH_SPEED_MSK	0x8	
ALTERA_TSEMAC_CMD_PROMIS_EN_OFST	4	Configures the PROMIS_EN bit.
ALTERA_TSEMAC_CMD_PROMIS_EN_MSK	0x10	
ALTERA_TSEMAC_CMD_PAD_EN_OFST	5	Configures the PAD_EN bit.
ALTERA_TSEMAC_CMD_PAD_EN_MSK	0x20	
ALTERA_TSEMAC_CMD_CRC_FWD_OFST	6	Configures the CRC_FWD bit.
ALTERA_TSEMAC_CMD_CRC_FWD_MSK	0x40	
ALTERA_TSEMAC_CMD_PAUSE_FWD_OFST	7	Configures the PAUSE_FWD bit.
ALTERA_TSEMAC_CMD_PAUSE_FWD_MSK	0x80	
ALTERA_TSEMAC_CMD_PAUSE_IGNORE_OFST	8	Configures the PAUSE_IGNORE bit.
ALTERA_TSEMAC_CMD_PAUSE_IGNORE_MSK	0x100	
ALTERA_TSEMAC_CMD_TX_ADDR_INS_OFST	9	Configures the TX_ADDR_INS bit.
ALTERA_TSEMAC_CMD_TX_ADDR_INS_MSK	0x200	
ALTERA_TSEMAC_CMD_HD_ENA_OFST	10	Configures the HD_ENA bit.
ALTERA_TSEMAC_CMD_HD_ENA_MSK	0x400	
ALTERA_TSEMAC_CMD_EXCESS_COL_OFST	11	Configures the EXCESS_COL bit.
ALTERA_TSEMAC_CMD_EXCESS_COL_MSK	0x800	
ALTERA_TSEMAC_CMD_LATE_COL_OFST	12	Configures the LATE_COL bit.
ALTERA_TSEMAC_CMD_LATE_COL_MSK	0x1000	
ALTERA_TSEMAC_CMD_SW_RESET_OFST	13	Configures the SW_RESET bit.
ALTERA_TSEMAC_CMD_SW_RESET_MSK	0x2000	
ALTERA_TSEMAC_CMD_MHASH_SEL_OFST	14	Configures the MHASH_SEL bit.
ALTERA_TSEMAC_CMD_MHASH_SEL_MSK	0x4000	

Table 6–1. Constants Mapping (Part 2 of 2)

Constant	Value	Description
ALTERA_TSEMAC_CMD_LOOPBACK_OFST	15	Configures the LOOP_ENA bit.
ALTERA_TSEMAC_CMD_LOOPBACK_MSK	0x8000	
ALTERA_TSEMAC_CMD_TX_ADDR_SEL_OFST	16	Configures the TX_ADDR_SEL bits (bits 16 - 18).
ALTERA_TSEMAC_CMD_TX_ADDR_SEL_MSK	0x70000	
ALTERA_TSEMAC_CMD_MAGIC_ENA_OFST	19	Configures the MAGIX_ENA bit.
ALTERA_TSEMAC_CMD_MAGIC_ENA_MSK	0x80000	
ALTERA_TSEMAC_CMD_SLEEP_OFST	20	Configures the SLEEP bit.
ALTERA_TSEMAC_CMD_SLEEP_MSK	0x100000	
ALTERA_TSEMAC_CMD_WAKEUP_OFST	21	Configures the WAKEUP bit.
ALTERA_TSEMAC_CMD_WAKEUP_MSK	0x200000	
ALTERA_TSEMAC_CMD_XOFF_GEN_OFST	22	Configures the XOFF_GEN bit.
ALTERA_TSEMAC_CMD_XOFF_GEN_MSK	0x400000	
ALTERA_TSEMAC_CMD_CNTL_FRM_ENA_OFST	23	Configures the CNTL_FRM_ENA bit.
ALTERA_TSEMAC_CMD_CNTL_FRM_ENA_MSK	0x800000	
ALTERA_TSEMAC_CMD_NO_LENGTH_CHECK_OFST	24	Configures the NO_LENGTH_CHECK bit.
ALTERA_TSEMAC_CMD_NO_LENGTH_CHECK_MSK	0x1000000	
ALTERA_TSEMAC_CMD_ENA_10_OFST	25	Configures the ENA_10 bit.
ALTERA_TSEMAC_CMD_ENA_10_MSK	0x2000000	
ALTERA_TSEMAC_CMD_RX_ERR_DISC_OFST	26	Configures the RX_ERR_DISC bit.
ALTERA_TSEMAC_CMD_RX_ERR_DISC_MSK	0x4000000	
ALTERA_TSEMAC_CMD_CNT_RESET_OFST	0x80000000	Configures the CNT_RESET bit.
Tx_Cmd_Stat Register (Table 4–18 on page 4–44)		
ALTERA_TSEMAC_TX_CMD_STAT_OMITCRC_OFST	17	Configures the OMIT_CRC bit.
ALTERA_TSEMAC_TX_CMD_STAT_OMITCRC_MSK	0x20000	
ALTERA_TSEMAC_TX_CMD_STAT_TXSHIFT16_OFST	18	Configures the TX_SHIFT16 bit.
ALTERA_TSEMAC_TX_CMD_STAT_TXSHIFT16_MSK	0x40000	
Rx_Cmd_Stat Register (Table 4–18 on page 4–44)		
ALTERA_TSEMAC_RX_CMD_STAT_RXSHIFT16_OFST	25	Configures the RX_SHIFT16 bit
ALTERA_TSEMAC_RX_CMD_STAT_RXSHIFT16_MSK	0x2000000	

Functionality Configuration Parameters

You can use the parameters in [Table A-1](#) to enable or disable specific functionality within the MegaCore function variation.



In VHDL testbenches, the parameter names are in UPPER case; in Verilog, the names are in lower case.

Table A-1. MegaCore Functionality Configuration Parameters (Part 1 of 4)

Parameter	Description	Default	Supported In
ETH_MODE	Defines the MAC operation. When the speed is set to 10 or 100Mbps, the Core MII interface is enabled, when the speed is set to 1000 Mbps, the GMII interface is enabled.	1000	Configurations that contains the 10/100/1000 Ethernet MAC
HD_ENA	Can only be used when the MAC speed in set 10 or 100 Mbps. Enables collision detection and retransmission buffer.	0	Configurations that contains the 10/100/1000 Ethernet MAC
TB_MACLENMAX	Defines the maximum frame length.	1518	Configurations that contains the 10/100/1000 Ethernet MAC
TB_MACPAUSEQ	Quanta value inserted on each Pause Frame generated by the MAC transmit logic.	15	Configurations that contains the 10/100/1000 Mbps Ethernet MAC
TB_MACIGNORE_PAUSE	When enabled, the MAC ignores the pause frame received from the Ethernet line.	0	Configurations that contains the 10/100/1000 Mbps Ethernet MAC
TB_MACFWD_PAUSE	When enabled, pause frames received by the MAC are forwarded to user application. When disabled, pause frames are terminated in the MAC.	0	Configurations that contains the 10/100/1000 Ethernet MAC
TB_MACFWD_CRC	When enabled, frames are forwarded to user application with the FCS field, when disabled, the MAC terminates the FCS field.	0	Configurations that contains the 10/100/1000 Ethernet MAC

Table A–1. MegaCore Functionality Configuration Parameters (Part 2 of 4)

Parameter	Description	Default	Supported In
TB_MACINSERT_ADDR	When enabled, the MAC forwards the source address received from the application, when disabled, the MAC overwrites the source MAC address from the application by the value programmed on the Core mac_addr.	1 or 0	Configurations that contains the 10/100/1000 Ethernet MAC
TB_PROMIS_ENA	When enabled, the MAC, regardless of their Unicast destination MAC address, accepts All configurations that Contain the 10/100/1000 Mbps Ethernet MAC frames. When disabled, the MAC filters Unicast frames.	0	Configurations that contains the 10/100/1000 Ethernet MAC
TB_MACPADEN	When enabled, the MAC terminates padding before transmitting frame to the application, when disabled, the MAC forward the frames to the application with the padding.	1 or 0	Configurations that contains the 10/100/1000 Ethernet MAC
TB_IPG_LENGTH	Transmit IPG Value. Can be set to any value between 8 and 27.	12	Configurations that contains the 10/100/1000 Ethernet MAC
TB_MDIO_ADDR0	Defines the MDIO address, between 0 and 31, on the PHY device addressable via the Core Slave 0 space.	0	Configurations that contains the 10/100/1000 Ethernet MAC
TB_MDIO_ADDR1	Defines the MDIO address, between 0 and 31, on the PHY device addressable via the Core Slave 1 space.	1	Configurations that contains the 10/100/1000 Ethernet MAC
TX_FIFO_AE	Set the Transmit FIFO almost empty threshold to any value between 0 and the Maximum memory depth.	8	Configurations that contains the 10/100/1000 Ethernet MAC
TX_FIFO_AF	Set the Transmit FIFO almost full threshold to any value between 0 and the Maximum memory depth.	8	Configurations that contains the 10/100/1000 Ethernet MAC
RX_FIFO_AE	Set the Receive FIFO almost empty threshold to any value between 0 and the Maximum memory depth.	8	Configurations that contains the 10/100/1000 Ethernet MAC

Table A–1. MegaCore Functionality Configuration Parameters (Part 3 of 4)

Parameter	Description	Default	Supported In
RX_FIFO_AF	Set the Receive FIFO almost full threshold to any value between 0 and the Maximum memory depth,	10	Configurations that contains the 10/100/1000 Ethernet MAC
TX_FIFO_SECTION_EMPTY	Set the Transmit FIFO Section empty threshold to any value between 0 and the Maximum memory depth.	16	Configurations that contains the 10/100/1000 Ethernet MAC
TX_FIFO_SECTION_FULL	Set the Transmit FIFO Section full threshold to any value between 0 and the Maximum memory depth.	16	Configurations that contains the 10/100/1000 Ethernet MAC
RX_FIFO_SECTION_EMPTY	Set the Receive FIFO Section empty threshold to any value between 0 and the Maximum memory depth.	0	Configurations that contains the 10/100/1000 Ethernet MAC
RX_FIFO_SECTION_FULL	Set the Receive FIFO Section full threshold to any value between 0 and the Maximum memory depth.	16	Configurations that contains the 10/100/1000 Ethernet MAC
MCAST_TABLEN	Number of MAC addresses in the MAC address table	9	Configurations that contains the 10/100/1000 Mbps Ethernet MAC
MCAST_ADDRESSLIST	Addresses in the MAC address table	X"887654332211", X"886644352611", X"ABCDEF012313", X"92456545ab15", X"432680010217", X"adb589215439", X"ffeacfe3434B", X"ffccddaa3123", X"adb358415439"	Configurations that contains the 10/100/1000 Ethernet MAC

Table A–1. MegaCore Functionality Configuration Parameters (Part 4 of 4)

Parameter	Description	Default	Supported In
TB_SGMII_ENA	When selected, set, via a Register management access that the PCS operates in SGMII mode. When deselected the PCS operates in standard 1000Base-X mode.	0	Configurations that contains the 1000BaseX PCS or SGMII
TB_SGMII_AUTO_CONF	When selected, set, via a Register management access that the PCS is configured with the parameters advertised by the PCS mode. When deselected the PCS is configured with static register settings.	0	Configurations that contains the 1000BaseX PCS or SGMII

Test Configuration Parameters

You can use the parameters in [Table A–2](#) to create custom test scenarios.

Table A–2. Test Configuration Parameters

Parameter	Description	Default	Supported In
TB_RXFRAMES	Sets the number of frames that are generated by the Ethernet frame Generator connected to the GMII / MII RX (<i>non-loopback test</i>). If set to 0, a loopback test is performed (GMII / MII TX is directly connected to GMII / MII RX). The Ethernet Generator is then disabled.	5	Configurations that contains the 10/100/1000 Ethernet MAC
TB_TXFRAMES	Sets the number of frames that are generated by the frame Generator connected to the Core transmit FIFO interface.	5 or 12	Configurations that contains the 10/100/1000 Ethernet MAC
TB_TXIPG	Sets the inter-packet gap (IGP) used by the Ethernet Frame generator when generating frames on GMII/MII TX.	12	Configurations that contains the 10/100/1000 Ethernet MAC
TB_RXIPG	Sets the inter-packet gap (IGP) used by the Ethernet Frame generator when generating frames on GMII / MII RX.	12	Configurations that contains the 10/100/1000 Ethernet MAC

Table A–2. Test Configuration Parameters

Parameter	Description	Default	Supported In
TB_ENA_VAR_IPG	When disabled, a stream of frames separated by a constant IPG (Defined by the Testbuilder option "IPG in RX path") is generated to the MAC Core Rx MII / GMII interface. When enabled, the streams of frame separated by a variable IPG is generated to the MAC Core MII / GMII interface. The IPG varies pseudo-randomly from a minimum value set by the Testbuilder option "IPG in RX path" and 48 Byte Time.	0	Configurations that contains the 10/100/1000 Ethernet MAC
TB_LENSTART	Defines the payload length of the first frame generated by the Ethernet and FIFO models. With each new frame this value is incremented by the value specified in the following setting.	1000 or 20	Configurations that contains the 10/100/1000 Ethernet MAC
TB_LENSTEP	Frame payload length increment. During simulation frames are generated starting from "length of first frame" incrementing with each frame generated.	1	Configurations that contains the 10/100/1000 Ethernet MAC
TB_LENMAX	Defines the payload maximum length used by the Ethernet Generator models. This value specifies the wrap around for the frame length of generated frames. I.e. if the frame length increment would exceed this value it wraps around to zero. Can be used to test frame length error detection, when set to any value larger than the MAC length configuration (see MAC configuration parameter TB_MACLENMAX).	1500	Configurations that contains the 10/100/1000 Ethernet MAC
TB_ENA_PADDING	If enabled, GMII / MII PHY model generated frames are padded to 64 octets in length (normal mode). If disabled, no padding occurs and erroneous frames will be sent to the MAC RX.	1 or 0	Configurations that contains the 10/100/1000 Ethernet MAC
TB_ENA_VLAN	If enabled, All configurations that Contain the 10/100/1000 Mbps Ethernet MAC frames sent/received will be VLAN type of frames	0	Configurations that contains the 10/100/1000 Ethernet MAC

Table A–2. Test Configuration Parameters

Parameter	Description	Default	Supported In
TB_STOPREAD	Inhibits the Testbench RX FIFO monitor reading the RX FIFO, after this amount of frames has been sent to the GMII / MII RX. Can be used to test Flow-Control behavior. If more frames are received, the FIFO will get filled. When the threshold level is reached, a Pause-Frame will be generated by the MAC TX. If set to 0, the RX FIFO read is never paused.	0	Configurations that contains the 10/100/1000 Ethernet MAC
TB_HOLDREAD	Number of clock cycles, the RX FIFO should not be read after it has been stopped. Only relevant if the previous configuration (read stop) was set to a non-null value. After this number of RX FIFO clock cycles, the RX FIFO will be emptied again.	1000	Configurations that contains the 10/100/1000 Ethernet MAC
TB_TX_FF_ERR	If enabled, the testbench will set the error pin on the Transmit FIFO interface causing the MAC to send frames marked as erroneous	0	Configurations that contains the 10/100/1000 Ethernet MAC
TB_TRIGGERXOFF	Time at which the Core xoff_gen command pin will be pulsed for one clock cycle to request transmission of a Xoff pause frame with the current pause quanta configuration. No effect (disabled), if set to 0.	0	Configurations that contains the 10/100/1000 Ethernet MAC
TB_TRIGGERXON	Time at which the Core xon_gen command pin will be pulsed for one clock cycle to request transmission of a Xon pause frame with the current pause quanta configuration. No effect (disabled), if set to 0.	0	Configurations that contains the 10/100/1000 Ethernet MAC
TB_MODPAUSEQ	Quanta value inserted on each Pause Frame generated by the GMII / MII Frame generator.	16	Configurations that contains the 10/100/1000 Ethernet MAC
RX_COL_FRM	Specified frame is received with a collision indication. Disabled when 0. Option only available if MAC operates in Half Duplex.	0	Configurations that contains the 10/100/1000 Ethernet MAC
RX_COL_GEN	Force a collision on the specified frame 4-bit nibble number.	0	Configurations that contains the 10/100/1000 Ethernet MAC
TX_COL_FRM	Specified which frame is received with a collision indication. When set to 0, disable collision generation, option only available if the MAC operates in Half Duplex mode.	0	Configurations that contains the 10/100/1000 Ethernet MAC

Table A–2. Test Configuration Parameters

Parameter	Description	Default	Supported In
TX_COL_GEN	Force a collision on the specified frame 4-bit nibble number.	0	Configurations that contains the 10/100/1000 Ethernet MAC
TX_COL_NUM	Defines the number of consecutive collisions during frame re-transmission.	0	Configurations that contains the 10/100/1000 Ethernet MAC
TX_COL_DELAY	Use to, when set to any number greater to 0, to shift the collision by the specified number of Nibbles during each re-transmission.	0	Configurations that contains the 10/100/1000 Ethernet MAC
TB_PAUSECONTROL	If enabled (true) the Testbench will stop the GMII / MII RX Frame generator, if a Pause Frame is sent by the MAC. This simulates a usual flow-control chain correctly. If disabled (false) Pause Frames sent by the MAC are ignored by the Testbench and the GMII / MII RX Frame generator will never be paused. This can be used to test the MACs FIFO overflow behavior. This option has no effect if <i>loopback</i> mode is enabled (i.e. Frame generation in RX path is set 0).	0	Configurations that contains the 10/100/1000 Ethernet MAC
TB_MIDO_SIMULATION	Enable / Disable MDIO simulation.	0	Configurations that contains the 10/100/1000 Ethernet MAC
PERIOD_HASHCLK	66MHz hash table programming	15 ns	Configurations that contains the 10/100/1000 Ethernet MAC
TB_MACLENMAX	This value specifies the wrap around for the frame length of generated frames. I.e. if the frame length increment would exceed this value it wraps around to TB_LENSTART.	1518	Configurations that contains the 10/100/1000 Ethernet MAC
TB_SGMII_HD	When selected, the PCS Core is configured for half-duplex operation.	0	Configurations that contains the 1000BaseX PCS or SGMII
TB_SGMII_1000	When selected, the PCS Core is configured for Gigabit operation.	1	Configurations that contains the 1000BaseX PCS or SGMII

Table A–2. Test Configuration Parameters

Parameter	Description	Default	Supported In
TB_SGMII_100	When selected, the PCS Core is configured for 100 Mbps operation.	0	Configurations that contains the 1000BaseX PCS or SGMII
TB_SGMII_10	When selected, the PCS Core is configured for 10 Mbps operation.	0	Configurations that contains the 1000BaseX PCS or SGMII
TB_TX_ERR	Enable GMII Error	0	Configurations that contains the 10/100/1000 Mbps Ethernet MAC