Triple Speed Ethernet MegaCore Function User Guide

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

Software Version: 9.0
Document Date: March 2009

UG-01008-1.7

Contents

Chapter 5. Testbench

Chapter 6. Software Programming Interface

Appendix A. Upgrading Triple Speed Ethernet MegaCore Function

Appendix B. Simulation Parameters

Additional Information

List of Figures

List of Tables

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	9.0
Release Date	March 2009
Ordering Code	IP-TRIETHERNET
Product ID(s)	00BD
Vendor ID(s)	6AF7

For more information about this release, refer to the *MegaCore IP Library Release Notes and Errata*.

Altera verifies that the current version of the Quartus® II software compiles the previous version of each MegaCore function. The *MegaCore IP Library Release Notes and Errata* report any exceptions to this verification. Altera does not verify compilation with MegaCore function versions older than one release.

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
Arria™ GX	Full
Arria II GX	Preliminary
Cyclone® II	Full
Cyclone III	Full
HardCopy® II	Preliminary
Stratix® II	Full

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

Device Family	Support
Stratix II GX	Full
Stratix III	Full
Stratix IV	Preliminary
Other device families	No support

Features

The Triple Speed Ethernet MegaCore function combines the features of a 10/100/1000 Mbps Ethernet media access controller (MAC) and a 1000BASE-X physical coding sub-layer (PCS) with an optional physical medium attachment (PMA). The following sections provide a high-level overview of the features.

General Features

- IEEE Standard 802.3 compliant—Tested and succesfully validated by the University of New Hampshire (UNH) InterOperability Lab.
- 10/100/1000 Mbps Ethernet MAC in half-duplex and full-duplex modes; half-duplex is only supported in MACs operating at 10/100 Mbps.
- 10/100 Mbps or 1000 Mbps small MAC.
- Multi-channel MAC—Supports up to 24 ports in MAC only configurations and configurations that implement LVDS I/O; 20 ports in configurations that implement gigabit transceiver blocks.
- FIFO-less MAC—Option to exclude internal FIFOs for low-latency system.
- 1000BASE-X/SGMII PCS with optional auto-negotiation.
- Virtual local area network (VLAN) and stacked VLAN tagged frames support as specified by IEEE 802.IQ.
- Easy-to-use MegaWizard® interface.
- IP functional simulation models for use in Altera-supported VHDL and Verilog HDL simulators.
- Support for OpenCore Plus evaluation.

Flexible Standard Interfaces

- Seamless interface to commercial Ethernet PHY devices via medium independent interface (MII) and gigabit medium independent interface (GMII).
- Support for reduced gigabit medium independent interface (RGMII) in 10/100/1000 Mbps.
- Optional management data input/output (MDIO) master interface for PHY device management. This interface is shared among all ports in a multi-channel MAC.
- Simple 8- or 32-bit interface to user application based on the Avalon® Streaming (Avalon-ST) specification.
- Common interface to the register space in multi-port MACs.

- Optional integrated Physical Medium Attachment (PMA).

High Throughput Rate

- Flow control by programmable pause quanta.
- Pause frame generation can be controlled by user applications; enabling flexible traffic flow control.
- Configurable FIFO size (64 bytes to 256 Kbytes) and programmable threshold levels.
- Programmable source and destination MAC addresses and receive frame filtering based on:
  - Up to 5 unicast destination MAC addresses
  - Up to 64 multicast destination MAC addresses
- Programmable maximum frame length up to 64 Kbytes, including jumbo frames.
- Programmable promiscuous mode support to omit destination MAC address checking on receive.

Management Interface and Loopback

- Optional statistics counters for Simple Network Management Protocol (SNMP) environments, supporting IEEE 802.3 basic and mandatory Management Information Database (MIB) package as well as Ethernet MIC (RFC 2665) and Remote Network Monitoring (RFC 2819).
- Optional internal loopback on the MAC's MII, GMII, and RGMII.
- PHY isolation support to allow the implementation of a hot swappable PHY device.
- Shared MDIO management block in multi-port MACs.

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 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 (Wake-on LAN).

The Triple Speed Ethernet MegaCore function comprises the following main blocks: MAC, PCS, and PMA (Figure 1–1). All blocks are optional and configurable at synthesis time. 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

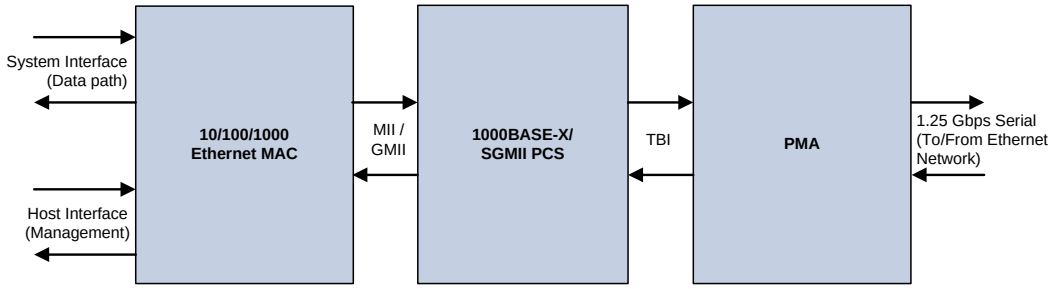


Table 1–3 shows the possible configurations for the Triple Speed Ethernet MegaCore function. Depending on the configuration, the interfaces on the system side and the Ethernet side change.

Table 1–3. Configuration Options for the MAC, PCS, and Embedded PMA

MegaCore Configuration			Interfaces	
10/100/1000 Ethernet MAC	1000BASE-X/ SGMII PCS	PMA	System Side	Ethernet Side
✓	—	—	Avalon-ST	GMII/RGMII/MII and optional MDIO
✓	✓	—	Avalon-ST	TBI and optional MDIO
✓	✓	✓	Avalon-ST	1.25 Gbps Serial
—	✓	✓	MII / GMII	1.25 Gbps Serial
—	✓	—	MII / GMII	TBI

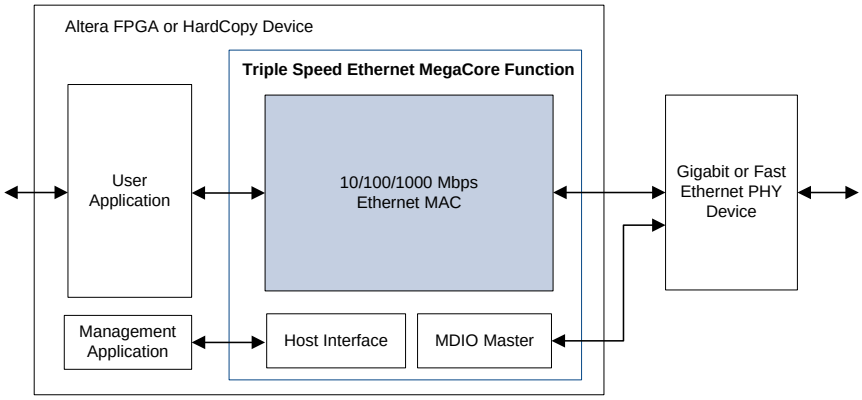
On the system side, the MAC function provides Altera's standard Avalon-ST interface, thus enabling interoperability with other Avalon-ST components. A host CPU can manage the MAC and PCS functions via an Avalon Memory-Mapped (Avalon-MM) interface, which provides access to the control register space.

On the Ethernet side, the MAC function can seamlessly connect to any industry standard gigabit Ethernet PHY device via GMII or RGMII, or to a fast Ethernet PHY device via MII or RGMII.

The 1000BASE-X PCS function is compliant with Clause 36 of the IEEE Standard 802.3 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 IEEE Standard 802.3, which is used to exchange ability information between the PCS function and a remote link partner. Auto-negotiation allows the PCS function to take 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 fast or gigabit Ethernet PHY devices. This example application does not include the PCS function.

Figure 1–2. Stand-Alone 10/100/1000 Mbps Ethernet MAC



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 ten-bit interface (TBI). Alternatively, when configured to include both the 1000BASE-X/SGMII PCS and PMA blocks, the MegaCore function can be connected directly to a gigabit interface converter (GBIC), small form-factor pluggable (SFP) module, or an SGMII PHY.

Figure 1–3 illustrates an example application using the Triple Speed Ethernet MegaCore function with 1000BASE-X and PMA. The PMA function connects to an off-the-shelf GBIC or SFP module to communicate directly over the optical link.

Figure 1–3. 10/100/1000 Mbps Ethernet MAC and 1000BASE-X PCS with Embedded PMA

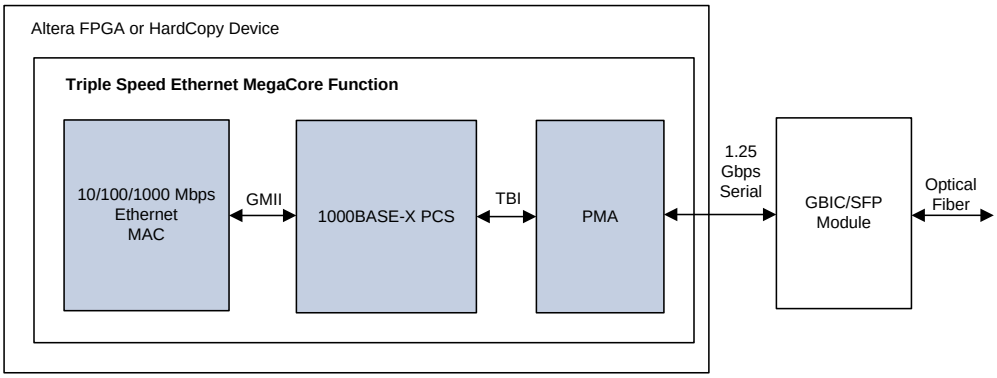
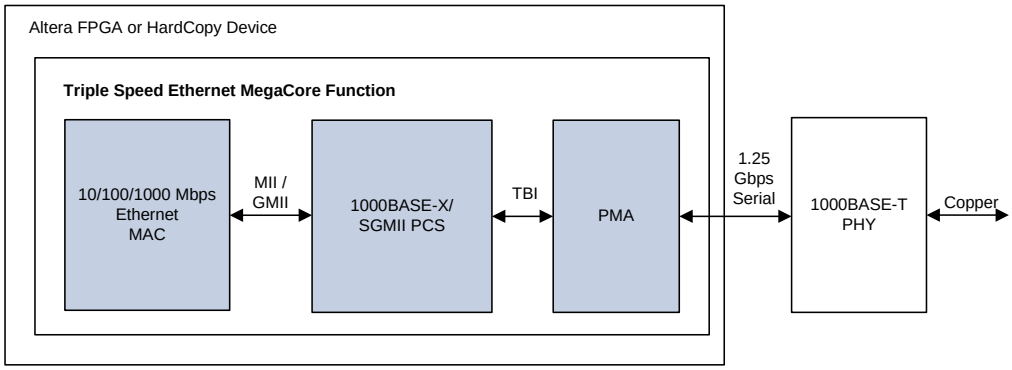


Figure 1–4 shows a similar configuration in which the PCS function is configured to operate in SGMII mode and acts as a GMII-to-SGMII bridge. In this case, the transceiver I/O connects to an off-the-shelf Ethernet PHY that supports SGMII (10BASE-T, 100BASE-T, or 1000BASE-T Ethernet PHY).

Figure 1–4. 10/100/1000 Mbps Ethernet MAC and SGMII PCS with Embedded PMA—GMII/MII to 1.25 Gbps Serial Bridge Mode



Throughout this document, the terms transceiver and serializer/deserializer (SERDES) are used interchangeably to refer to FPGA device features that enable transmission and reception of high-speed serial data streams. The terms packet and frame are also used interchangeably.

MegaCore Verification

For each release, Altera verifies the Triple Speed Ethernet MegaCore function through extensive simulation and internal hardware verification in various Altera device families. 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

Altera has also 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, the 10/100/1000 Mbps Ethernet MAC, 1000BASE-X/SGMII PCS, and PMA 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 by manipulating the MAC address fields, thus implementing a loopback. A direct connection to an optical module is provided through an external SFP optical module. Certified 1.25 GBaud optical SFP transceivers are Finisar 1000BASE-SX FTLF8519P2BNL, Finisar 1000BASE-LX FTRJ-1319-3, and Avago Technologies AFBR-5710Z.

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 MII, GMII, RGMII, and SGMII, in conjunction with the 1000BASE-X/SGMII PCS and PMA functions.

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 MAC address fields, thus implementing a loopback. A direct connection to an Ethernet link is provided through a combined MII to an external PHY module. Certified 1.25 GBaud copper SFP transceivers are Finisar FCMJ-8521-3, Methode DM7041, and Avago Technologies ABCU-5700RZ.

Performance and Resource Utilization

Table 1–4 provides the estimated resource utilization and performance of the Triple Speed Ethernet MegaCore function for the Stratix II GX device family. The estimates are obtained by compiling the Triple Speed Ethernet MegaCore function using the Quartus II software targeting a Stratix II GX (EP2SGX130GF1508C5) device with speed grade -5.

Table 1–4. Stratix II GX Performance and Resource Utilization

MegaCore Function	Settings	Comb. ALUT	Logic Regs	Memory (M512 / M4K/ MRAM)	f_{max} MHz
MAC and PCS/PMA	Full duplex Internal FIFO disabled	1,876	2,306	3/4/0	> 125
4-port MAC and PCS/PMA	Statistics counters enabled SGMII bridge enabled	7,113	8,742	12/16/0	> 125
MAC and PCS/PMA	Full duplex	1,164	1,736	2/1/0	> 125
12-port MAC and PCS/PMA	Internal FIFO disabled	12,909	19,058	37/0/0	> 125
20-port MAC and PCS/PMA	Statistics counters disabled SGMII bridge enabled	21,485	31,652	60/1/0	> 125

Table 1–5 provides the estimated resource utilization and performance of the Triple Speed Ethernet MegaCore function for the Stratix III device family. The estimates are obtained by compiling the Triple Speed Ethernet MegaCore function using the Quartus II software targeting a Stratix III (EP3SL340H1152C4) device with speed grade -4.

Table 1–5. Stratix III Performance and Resource Utilization (Part 1 of 2)

MegaCore Function	Settings	FIFO Size	Comb. ALUT	Logic Regs	Memory (M9K / M144K/ MLAB)	f_{max} MHz
10/100 Mbps Small MAC	Half-duplex	64×32-bit	1,029	1,482	3/0/2720	
		2048×32-bit	1,095	1,547	23/0/128	
	Full-duplex	64×32-bit	706	1,124	4/0/160	>125
		2048×32-bit	851	1,358	22/0/128	
	Full-duplex All MAC options enabled	2048×32-bit	1,036	1,611	22/0/128	
1000Mbps Small MAC	GMII	64×32-bit	687	1,055	4/0/160	
		2048×32-bit	832	1,290	22/0/128	
	RGMII	2048×32-bit	843	1,329	22/0/128	>125
	GMII All MAC options enabled	2048×32-bit	1,023	1,544	22/0/128	
MAC	Half duplex MII/GMII	64×8-bit	1,040	1,394	4/0/136	
		64×32-bit	1,120	1,511	5/0/160	
		2048×8-bit	1,235	1,673	10/0/128	>125
		2048×32-bit	1,305	1,786	23/0/128	
	Half duplex MII/GMII Statistics counters enabled	64×8-bit	1,755	2,241	4/0/1672	
		64×32-bit	1,837	2,362	5/0/1696	
		2048×8-bit	1,952	2,523	10/0/1664	>125
		2048×32-bit	2,018	2,634	23/0/1664	
	Full duplex MII/GMII	2048×8-bit	1,020	1,483	9/0/128	
		2048×32-bit	1,096	1,596	22/0/128	>125
	Full duplex MII/GMII All MAC options enabled	2048×8-bit	2,728	3,222	10/0/1764	
		2048×32-bit	2,896	3,397	23/0/1764	>125
	Full duplex RGMII	2048×8-bit	1,033	1,529	9/0/128	
		2048×32-bit	1,107	1,642	22/0/128	>125
PCS	1000BASE-X	—	613	622	0/0/0	
	SGMII bridge enabled	—	776	892	2/0/0	>125

Table 1–5. Stratix III Performance and Resource Utilization (Part 2 of 2)

MegaCore Function	Settings	FIFO Size	Comb. ALUT	Logic Regs	Memory (M9K / M144K/ MLAB)	f_{max} MHz
MAC and PCS	Half-duplex	64×8-bit	2,541	3,187	6/0/1672	
	Statistics counters enabled	64×32-bit	2,633	3,304	7/0/1696	
	SGMII bridge enabled	2048×8-bit	2,754	3,466	12/0/1664	>125
		2048×32-bit	2,839	3,577	25/0/1664	
	Full-duplex	2048×8-bit	3,548	4,140	12/0/1764	
	All MAC options enabled	2048×32-bit	3,667	4,316	25/0/1764	>125
	SGMII bridge enabled					
MAC and PCS/ PMA	Half-duplex	2048×8-bit	2,761	3,518	12/0/1664	
	Statistics counters enabled	2048×32-bit	2,842	3,630	25/0/1664	>125
	SGMII bridge enabled					
	Full-duplex	2048×8-bit	3,563	4,180	13/0/1700	
	All MAC options enabled	2048×32-bit	3,695	4,354	26/0/1700	>125
	SGMII bridge enabled					
MAC and PCS/PMA	Full duplex	—	2,035	2,791	2/0/1664	>125
4-port MAC and PCS/PMA	Internal FIFO disabled	—	7,844	10,673	8/0/6656	>125
12-port MAC and PCS/PMA	Statistics counters enabled	—	23,277	31,689	24/0/20040	>125
24-port MAC and PCS/PMA	SGMII bridge enabled	—	46,101	63,192	48/0/40080	>125
MAC and PCS/PMA	Full duplex	—	1,298	1,833	2/1/0	>125
12-port MAC and PCS/PMA	Internal FIFO disabled	—	14,308	19,732	38/1/0	>125
20-port MAC and PCS/PMA	Statistics counters disabled SGMII bridge enabled	—	23,795	32,747	61/3/0	>125

Table 1–6 provides the estimated resource utilization and performance of the Triple Speed Ethernet MegaCore function for the Stratix IV device family. The estimates are obtained by compiling the Triple Speed Ethernet MegaCore function using the Quartus II software targeting a Stratix IV GX (EP4SGX530NF45C4) device with speed grade -4.

Table 1–6. Stratix IV Performance and Resource Utilization (Part 1 of 2)

MegaCore Function	Settings	FIFO Size	Comb. ALUT	Logic Regs	Memory (M9K / M144K/ MLAB)	f_{max} MHz
10/100 Mbps Small MAC	Half-duplex	64×32-bit	974	1,369	3/0/2080	>125
		2048×32-bit	1118	1,582	22/0/1408	
	Full-duplex	64×32-bit	705	1,124	4/0/160	
		2048×32-bit	851	1,360	22/0/128	
	Full-duplex All MAC options enabled	2048×32-bit	1,036	1,612	22/0/128	
1000Mbps Small MAC	GMII	64×32-bit	687	1,056	4/0/160	>125
		2048×32-bit	832	1,291	22/0/128	
	RGMII	2048×32-bit	843	1,386	22/0/128	
	GMII All MAC options enabled	2048×32-bit	1,023	1,545	22/0/128	
MAC	Half duplex MII/GMII	2048×32-bit	1,329	1,824	22/0/1408	>125
	Half duplex MII/GMII Statistics counters enabled	64×8-bit	1,769	2,278	3/0/2952	>125
		64×32-bit	1,851	2,395	4/0/2976	
		2048×8-bit	1,976	2,576	8/0/3200	
		2048×32-bit	2,036	2,670	22/0/2944	
	Full duplex MII/GMII	2048×8-bit	1,024	1,504	8/0/384	>125
		2048×32-bit	1,096	1,599	22/0/128	
	Full duplex MII/GMII All MAC options enabled	2048×8-bit	2,883	3,442	9/0/2,020	>125
		2048×32-bit	3,011	3,600	23/0/1,764	
	Full duplex RGMII	2048×8-bit	1,037	1,564	8/0/384	>125
		2048×32-bit	1,107	1,702	22/0/128	
PCS	1000BASE-X	—	614	623	0/0/0	>125
	SGMII bridge enabled	—	774	891	2/0/0	
MAC and PCS	Half-duplex Statistics counters enabled SGMII bridge enabled	64×8-bit	2,560	3,222	5/0/2952	>125
		64×32-bit	2,651	3,339	6/0/2976	
		2048×8-bit	2783	3520	10/0/3200	
		2048×32-bit	2852	3614	24/0/2944	
	Full-duplex All MAC options enabled SGMII bridge enabled	2048×8-bit	3509	4161	11/0/2020	>125
		2048×32-bit	3661	4317	25/0/1764	

Table 1–6. Stratix IV Performance and Resource Utilization (Part 2 of 2)

MegaCore Function	Settings	FIFO Size	Comb. ALUT	Logic Regs	Memory (M9K / M144K/ MLAB)	f_{max} MHz
MAC and PCS/ PMA (GXB)	Half-duplex Statistics counters enabled SGMII bridge enabled	2048×8-bit	2,592	3,385	10/0/3200	>125
		2048×32-bit	2,659	3,479	24/0/2944	
	Full-duplex All MAC options enabled SGMII bridge enabled	2048×8-bit	3,346	4,021	11/0/2020	>125
		2048×32-bit	3,513	4,178	25/0/1764	
MAC and PCS/ PMA (LVDS_IO)	Half-duplex Statistics counters enabled SGMII bridge enabled	2048×8-bit	2,785	3,573	10/0/3200	
		2048×32-bit	2,870	3,672	24/0/2944	
	Full-duplex All MAC options enabled SGMII bridge enabled	2048×8-bit	3,559	4,207	12/0/1956	
		2048×32-bit	3,702	4,362	26/0/1700	
MAC and PCS/PMA (GXB)	Full duplex Internal FIFO disabled Statistics counters enabled SGMII bridge enabled	—	1,876	2,622	2/0/1664	>125
4-port MAC and PCS/PMA (GXB)		—	7,127	9,955	8/0/6656	>125
12-port MAC and PCS/PMA (GXB)		—	21,194	29,505	24/0/20004	>125
24-port MAC and PCS/PMA (GXB)		—	42,121	58,829	48/0/40,008	>125
MAC and PCS/PMA (LVDS_IO)	Full duplex Internal FIFO disabled Statistics counters enabled SGMII bridge enabled	—	2,040	2,794	2/0/1664	>125
4-port MAC and PCS/PMA (LVDS_IO)		—	7,853	10,679	8/0/6656	>125
12-port MAC and PCS/PMA (LVDS_IO)		—	23,273	31,695	24/0/20040	>125
24-port MAC and PCS/PMA (LVDS_IO)		—	46,107	63,196	48/0/40,080	>125

Table 1–7 provides the estimated resource utilization and performance of the Triple Speed Ethernet MegaCore function for the Cyclone III device family. The estimates are obtained by compiling the Triple Speed Ethernet MegaCore function using the Quartus II software targeting a Cyclone III (EP3C120F780I7) device with speed grade -7.

Table 1–7. Cyclone III Performance and Resource Utilization (Part 1 of 2)

MegaCore Function	Settings	FIFO Size (Bit)	Logic Elements	Registers	Memory (M9K)	f_{max} MHz
1000Mbps Small MAC	Full duplex RGMII	2048x32	1,655	1,279	24	>125
MAC only		2048x32	2,167	1,634	23	>125

Table 1–7. Cyclone III Performance and Resource Utilization (Part 2 of 2)

MegaCore Function	Settings	FIFO Size (Bit)	Logic Elements	Registers	Memory (M9K)	f_{max} MHz
MAC only	Full duplex	—	2,272	1,577	5	>125
12-port MAC	Internal FIFO disabled RGMII	—	24,540	17,539	60	>125
MAC and PCS	Full duplex	—	3,629	2,436	7	> 125
4-port MAC and PCS	Internal FIFO disabled	—	13,660	9,276	28	> 125
12-port MAC and PCS	Statistics counters enabled SGMII	—	40,646	27,486	85	> 125

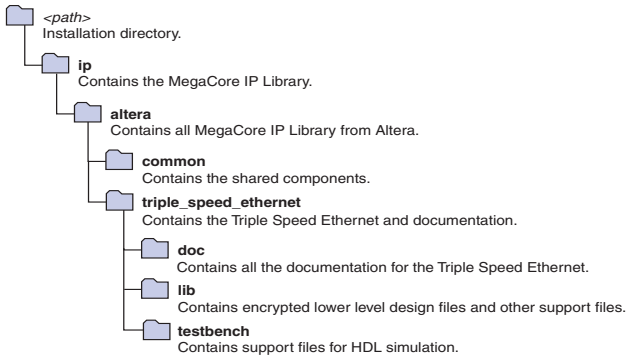
Installation and Licensing

The Triple Speed Ethernet MegaCore function is part of the MegaCore IP Library, which is distributed with the Quartus II software and downloadable from the Altera website at www.altera.com.

For system requirements and installation instructions, refer to *Quartus II Installation & Licensing for Windows and Linux Workstations*.

Figure 1–5 shows the directory structure after you install the Triple Speed Ethernet MegaCore function, where <*path*> is the installation directory. The default installation directory on Windows is **c:\altera**\<*version*>; on Linux it is **/opt/**<*version*>.

Figure 1–5. Directory Structure



The directory <*path*>**/ip/altera/triple_speed_ethernet/lib** contains common files for the Triple Speed Ethernet MegaCore function. If you intend to edit any of these files, such as the top-level definition of the altgx megafunction, make a local copy of the file in your project directory and edit the local copy.

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 AMPP^{SM} 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.

You only need to purchase a license for the megafunction when you are completely satisfied with its functionality and performance, and want to take your design to production.

After you purchase a license for the 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 more information about OpenCore Plus hardware evaluation, refer to *AN 320: OpenCore Plus Evaluation of Megafunctions*.

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.

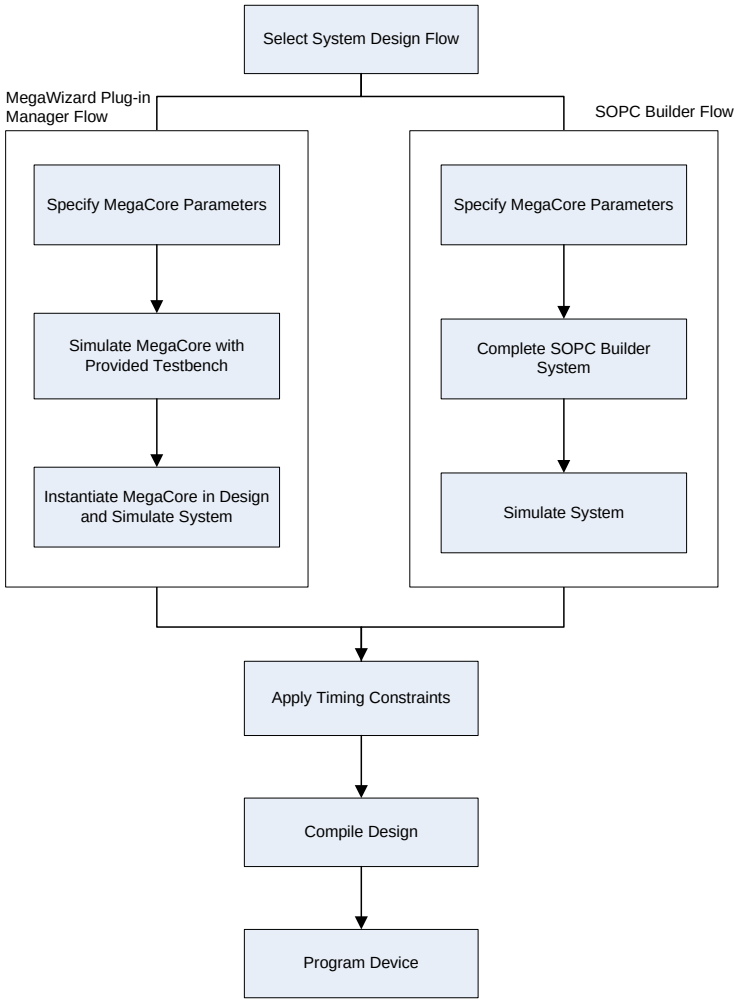
Your design stops working after the hardware evaluation time expires, and some signals are forced low. The following signals are forced low in configurations that contain the 10/100/1000 Ethernet MAC function:
`ff_rx_data`, `ff_rx_mod`, `ff_rx_eop`, `ff_rx_sop`, `ff_rx_dval`, `rx_err`, `gm_tx_d`, `gm_tx_en`, `gm_tx_err`, `m_tx_d`, `m_tx_en`, and `m_tx_err`.

In configurations that contain the 1000BASE-X/SGMII PCS function, the following signals are forced low:
`gmii_rx_d`, `gmii_rx_dv`, `gmii_rx_err`, `mii_rx_d`, `mii_rx_dv`, `mii_rx_err`, `mii_rx_col`, `mii_rx_crs`, and `tbi_tx_d`.

Triple Speed Ethernet Design Flow

Figure 2–1 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



Design 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
■ 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.	■ 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 more information about SOPC Builder, refer to Volume 4 of the *Quartus II Handbook*. For more information about the Quartus II software, refer to the 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 "Parameter Settings" 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, turn on **Simulation** 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 more information about simulating SOPC Builder systems, refer to volume 4 of the *Quartus II Handbook* and *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. 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 more information about MegaWizard Plug-in Manager and the Quartus II software, refer to the Quartus II Help.

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** 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 "Parameter Settings" on page 3–1.
4. On the **EDA** tab, turn on **Generate simulation model** 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.

   Some third-party synthesis tools can use a netlist that contains only the structure of the MegaCore function, but not detailed logic, to optimize performance of the design that contains the MegaCore function. If your synthesis tool supports this feature, turn on **Generate netlist**.

   CAUTION: 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.

5. 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 2–5.
6. Click **Finish** to generate the MegaCore function and supporting files.

The Quartus II IP File (**.qip**) is a file generated by the MegaWizard interface that contains information about a generated IP core. You are prompted to add this **.qip** file to the current Quartus II project at the time of file generation. In most cases, the **.qip** file contains all of the necessary information and assignments required to process the core or system in the Quartus II compiler. Generally, a single **.qip** file is generated for each MegaCore function and for each SOPC Builder system. However, some more complex SOPC Builder components generate a separate **.qip** file, so the system **.qip** file references the component **.qip** file.

Generated Files

Table 2–2 lists the files generated in your project directory. The type of files generated and their names vary depending on the custom variation of the MegaCore function you created.

Table 2–2. *Generated Files* **(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*>**.qip**	Contains Quartus II project information for your MegaCore function variations.
<*variation name*>**_gb.v**	A timing and resource estimation netlist for use in some third-party synthesis tools. This file is generated when the option **Generate netlist** on the **EDA** page is turned on.
<*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*>**.html**	MegaCore function report file.
<*variation name*>**.vho** or <*variation name*>**.vo**	VHDL or Verilog HDL IP functional simulation model.
<*variation_name*>**_contraints.tcl**	A Tcl script that creates necessary constraints for the Quartus II compilation of your MegaCore Function variation.
<*variation_name*>**_contraints.sdc**	Quartus II SDC constraint file for use with TimeQuest timing analyzer.
<*variation_name*>**_nativelink.tcl**	A Tcl script that assigns NativeLink simulation testbench settings to the Quartus II project.
/testbench/<*variation_name*>**/**<*variation_name*>**_tb.vhd** or **/testbench/**<*variation_name*>**/**<*variation_name*>**_tb.v**	VHDL or Verilog HDL testbench that exercises your MegaCore function variation in a third party simulator.
/testbench/<*variation_name*>**/run_**<*variation_name*>**_tb.tcl**	A Tcl script for use with the ModelSim simulation software.

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

File Name	Description
/testbench/<*variation_name*>**/**<*variation_name*>**_wave.do**	A signal tracing macro script used with the ModelSim simulation software to display testbench signals.
/testbench/model	A directory containing VHDL and Verilog HDL models of the Ethernet generators and monitors used by the generated testbench.

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. For more information, see sections "Generated Files" on page 2–5 and "Testbench Architecture" on page 5–1.

You can use any supported simulator for this purpose. The Triple Speed Ethernet MegaWizard interface 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 IP functional simulation models, refer to *Simulating Altera IP in Third-Party Simulation Tools* chapter in volume 3 of the *Quartus II Handbook*.

Simulating with the ModelSim Simulator

Perform the following steps 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

Perform the following steps to use the Quartus II NativeLink feature to run simulation in other Altera-supported third party simulators:

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

   This setting is global 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. On the Processing menu, point to **Start** and click **Analysis and Synthesis** to create the required netlist.
5. Run the simulation.

For more information about Simulating using NativeLink, refer to the *Simulating Altera IP in Third-Party Simulation Tools* chapter in volume 3 of the *Quartus II Handbook*.

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. You might need to add timing constraints which are external to the MegaCore function.

Follow the steps below to use the generated constraint files:

1. Edit <*variation_name*>`_constraints.tcl` and <*variation_name*>`_constraints.sdc` according to your customized design.
2. Open your Quartus II project in the Quartus II software.
3. Ensure your preferred timing analyzer is selected (**Timing Analysis Settings** in the Assignments menu).
4. 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 except for configurations that contain multi-port MAC variations or embedded PMA using LVDS Soft-CDR I/O.
5. If you select the 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 more information about timing analyzers, refer to the *Timing Analysis* section in volume 3 of the *Quartus II Handbook* and the 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 more information about compiling a design and programming Altera devices, refer to Quartus II Help.

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.

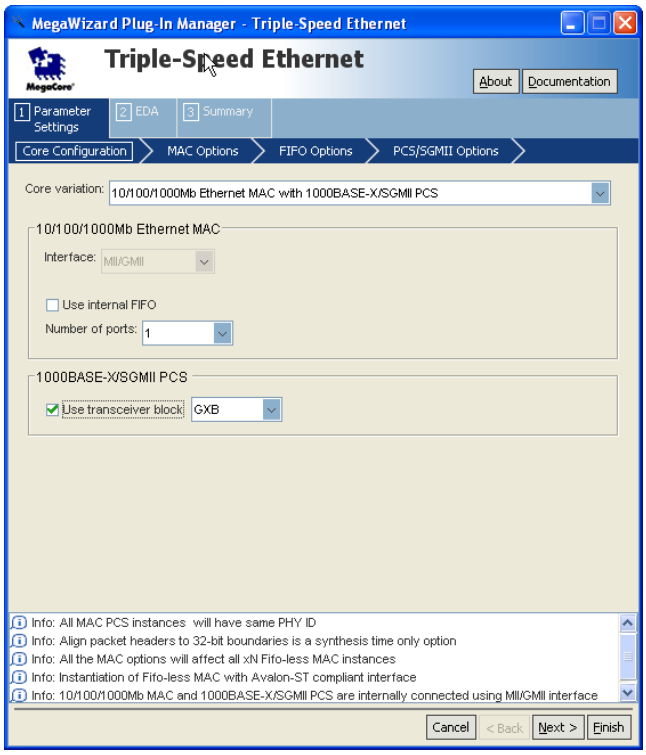
When parameterizing a MegaCore function using the SOPC Builder flow, the **EDA** and **Summary** tabs are not visible. In the SOPC Builder flow, you specify simulation model settings in the SOPC Builder GUI.

For more information on setting simulation options, refer to the Quartus II Help.

Core Configuration

The options on the **Core Configuration** page allow you to specify the primary Ethernet functional block, the interface for the MAC block if no PCS support is selected, the inclusion of internal FIFOs in the MAC block, the number of ethernet ports, and to enable the integrated transceiver block, if applicable. The selected configuration enables specific core features during synthesis and generation.

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**
- **1000 Mbps Small MAC**
- **10/100 Mbps Small MAC**

When instantiating the Triple Speed Ethernet MegaCore function using the SOPC Builder flow, the **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**—The only option available for 10/100 Mbps Small MAC core variations.
- **GMII**—Available only for 1000 Mbps Small MAC core variations.
- **RGMII**—Available for 10/100/1000 Mbps Ethernet MAC and 1000 Mbps Small MAC core variations.
- **MII/GMII**—Available only for 10/100/1000 Mbps Ethernet MAC core variations. 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.

Use Internal FIFO

If this parameter is turned on, internal FIFOs are included in the core. You can only include internal FIFOs in single-port MACs.

Number of Ports

The total number of Ethernet ports supported by the core. Legal values are **1**, **4**, **8**, **12**, **16**, **20**, and **24**. This parameter is enabled if the parameter **Use internal FIFO** is turned off. A multi-port MAC does not support internal FIFOs.

Use Transceiver Block

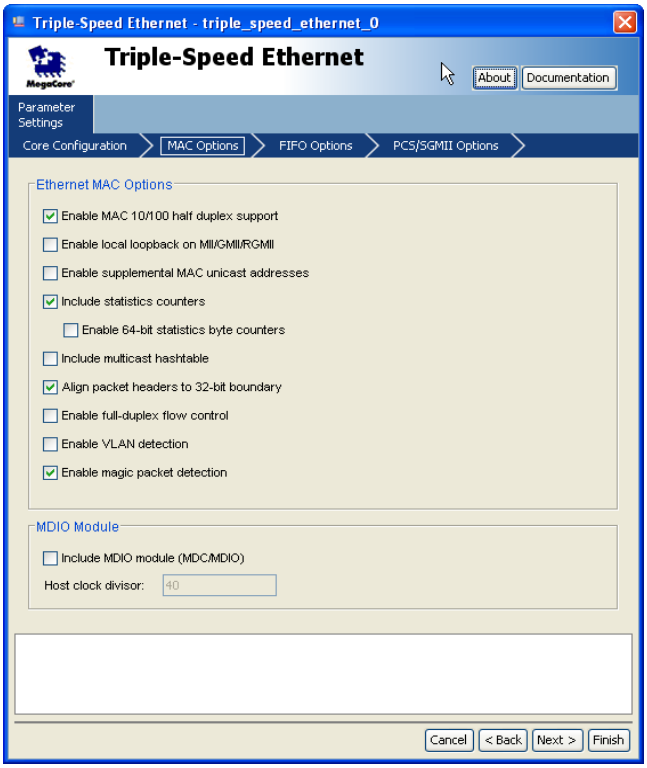
The **Use transceiver block** setting determines the external network interface for the PCS block. This option is only available if the MegaCore function includes the PCS block.

- When turned off, the PCS block implements a Ten-bit Interface (TBI) to an external SERDES chip.
- When turned on, the MegaCore function includes an integrated transceiver module to implement a 1.25 Gbps transceiver. Respective GXB module is included for target devices with GX transceivers. For target devices with LVDS I/O including Soft-CDR such as Stratix III, the ALTLVDS module is included.

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 Ethernet MAC options allow you to add or remove feature support based on your end-application needs.

These options provide flexibility for you to use only resources that are needed for your system needs and also simplify your 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 enable local loopback on the MAC's MII, GMII, or RGMII interface. If you turn on this option, the loopback function can be dynamically enabled or disabled during system operation via the MAC configuration register.
- **Enable supplemental MAC unicast addresses**—Turn on this option to include support for supplemental destination MAC unicast addresses for fast hardware-based received frame filtering.

- **Include statistics counters**—Turn on this option to include support for simple network monitoring protocol (SNMP) management information base (MIB) and remote monitoring (RMON) statistics counter registers for incoming and outgoing Ethernet packets.

  By default, the width of all statistics counters are 32 bits. The option **Enable 64-bit statistics byte counters** allows you to extend the width of selected statistics counters— `aOctetsTransmittedOK`, `aOctetsReceivedOK`, and `etherStatsOctets`—to 64 bits.

- **Include multicast hashtable**—Turn on this option to implement a hash table, a fast hardware-based mechanism to detect and filter multicast destination MAC address in received Ethernet packets.

- **Align packet headers to 32-bit boundaries (applicable to 32-bit FIFO only)**—Turn on this option to include logic that aligns all packet headers to 32-bit boundaries. This helps reduce software overhead processing in realignment of data buffers.

  This option is only available when the FIFO width is 32 bits and does not apply to multi-port MACs.

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

- **Enable full-duplex flow control**—Turn on this option to include logic for full-duplex flow control which includes pause frames generation and termination.

- **Enable VLAN detection**—Turn on this option to include logic for VLAN and stacked VLAN frame detection.

- **Enable magic packet detection**—Turn on this option to include logic for magic packet detection (Wake-on LAN).

  
  For 10/100 Small MAC core variations, the options available are **Enable MAC 10/100 half duplex support** and **Align packet headers to 32-bit boundaries (applicable to 32-bit FIFO only)**.

  For 1000 Small MAC core variations, the only option available is **Align packet headers to 32-bit boundaries (applicable to 32-bit FIFO only)**.

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.

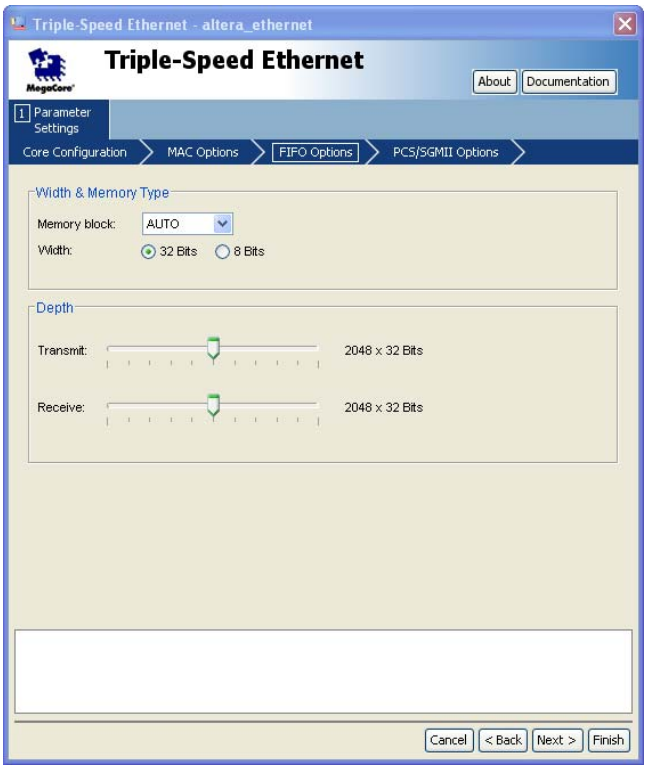
- **Host Clock Divisor** —This is a clock divisor to divide the MAC control register interface clock to produce the MDC clock output on the MDIO interface. Typically, the MDC clock should be 2.5 MHz.

  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 with the option **Use Internal FIFO** turned on.

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.

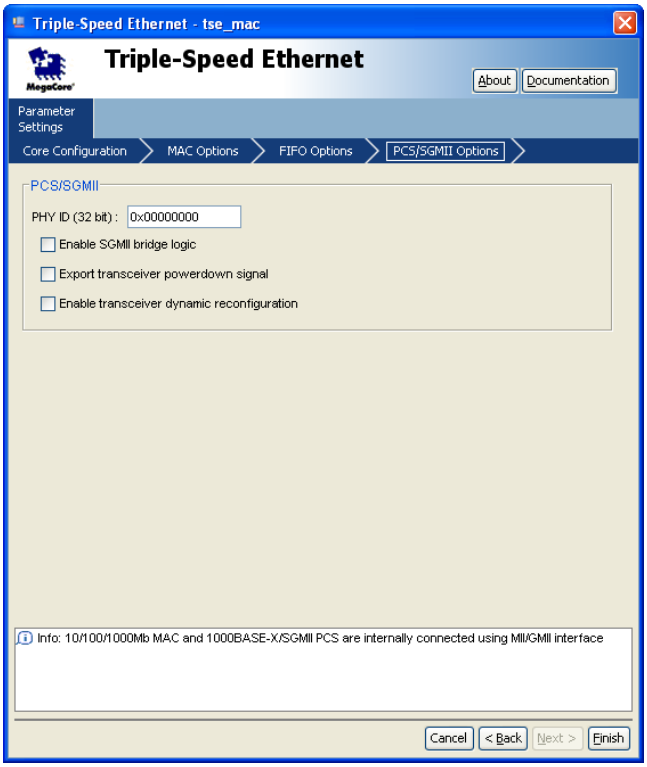
Depth

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 4–45.
- **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 reduces resource usage.

- **Export transceiver powerdown signal—**This option applies only to target devices with GX transceivers. 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. This allows the host processor to control the transceiver powerdown in your system.

  
  For UNH-IOL certification purposes, the embedded GX transceiver megafunction must be set to use 7-bit word alignment pattern length to recognize the comman character found in /K28.1/, /K28.5/, and /K28.7/. Use the MegaWizard Plug-in Manager to edit the megafunction and change the default word alignment setting to 7 bits.

- **Enable transceiver dynamic reconfiguration**—This option applies only to target devices with GX transceivers. Turn on this option if you are including an external dynamic reconfiguration controller in your design.

  If your design targets a Stratix II GX or an Arria GX device, turn on this option to share the PMA quad with other protocols such as PCI Express. This option is automatically turned on if your design targets a Stratix IV GX or an Arria II GX device.

  
  Refer to the respective device handbook for more information on dynamic reconfiguration in Altera devices.

10/100/1000 Ethernet MAC

The 10/100/1000 Ethernet MAC function provides an Avalon-ST interface to user applications and an industry standard interface to external PHY devices. The function supports the following interfaces to external PHYs: media independent interface (MII), gigabit media independent interface (GMII), and reduced gigabit media independent interface (RGMII).

The MAC function supports up to 24 ports in a single instance. You can configure single-port MACs to include internal FIFOs to buffer transmit and receive data. In a multi-port MAC, the ports can operate at different speeds and there is no option to include internal FIFOs. You can use the Avalon-ST Multi-Channel Shared Memory FIFO core in SOPC Builder to buffer the data.

Figure 4–1 shows a block diagram of the 10/100/1000 Ethernet MAC function with internal FIFOs.

Figure 4–1. 10/100/1000 Ethernet MAC With Internal FIFOs

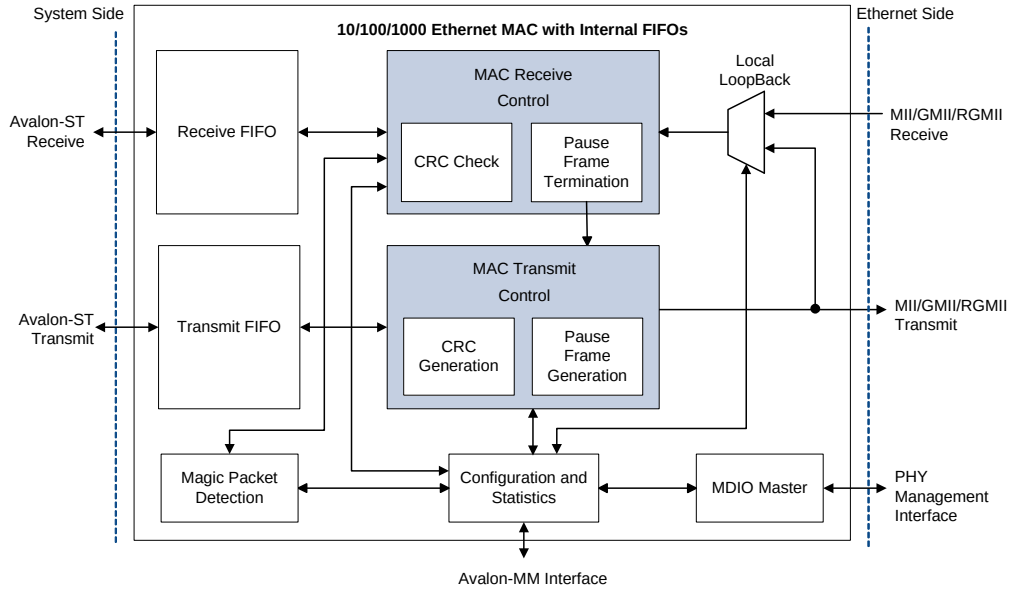
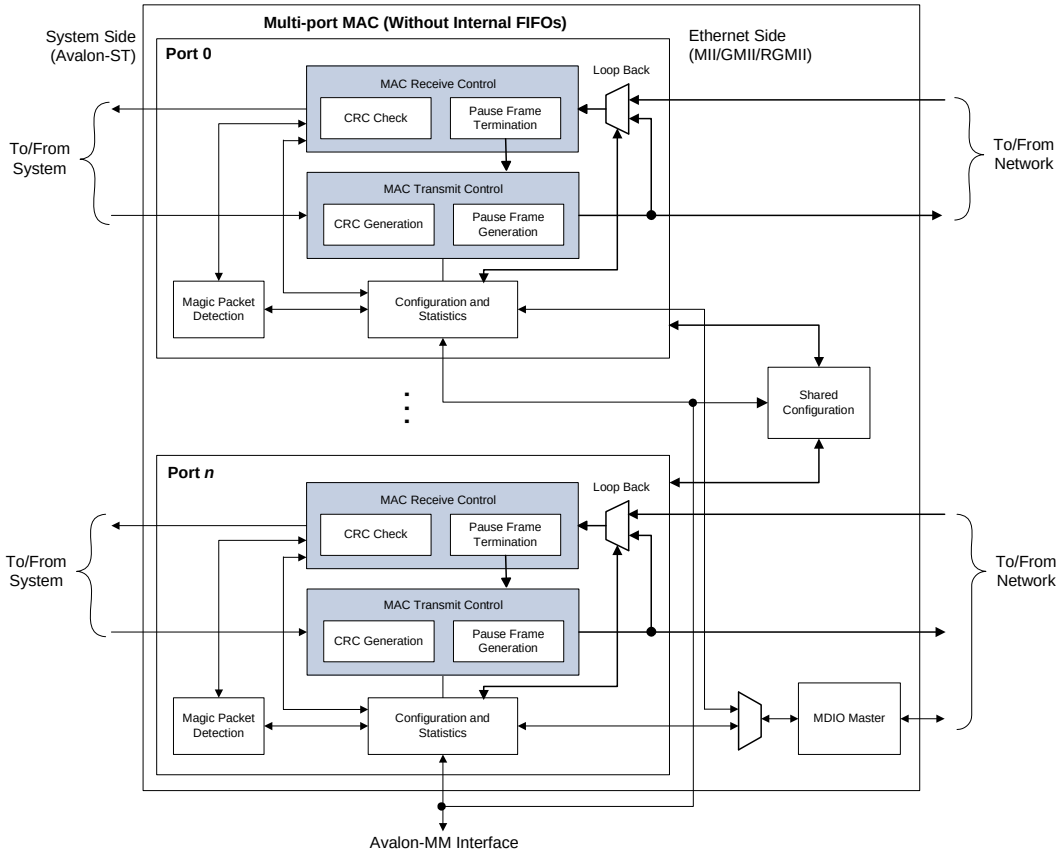


Figure 4–2 shows a block diagram of the 10/100/1000 Ethernet MAC function with multiple ports.

Figure 4–2. Multi-port MAC Without Internal FIFOs



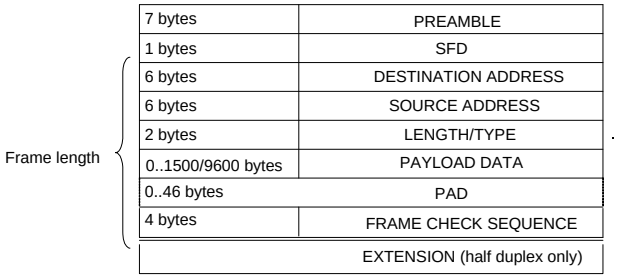
MAC Frame Format

A basic MAC frame comprises the following fields:

- Preamble—A maximum of 7 bytes.
- Start frame delimiter (SFD)—A fixed value of 0xD5 which marks the beginning of a frame.
- Destination and source addresses—Each address is 48 bits. The least significant byte is transmitted first.
- Length or type—A value equal to or greater than 1536 (0x600) indicates a type field. Otherwise, this field contains the length of the payload data. The most significant byte of this field is transmitted first.
- Payload Data and Pad—Variable length data and padding.
- Frame check sequence (FCS)—Cyclic Redundancy Check (CRC) value.
- An extension field—Required only for gigabit Ethernet operating in half-duplex mode. The MAC function does not support this implementation.

Figure 4–3 shows the format of a basic MAC frame.

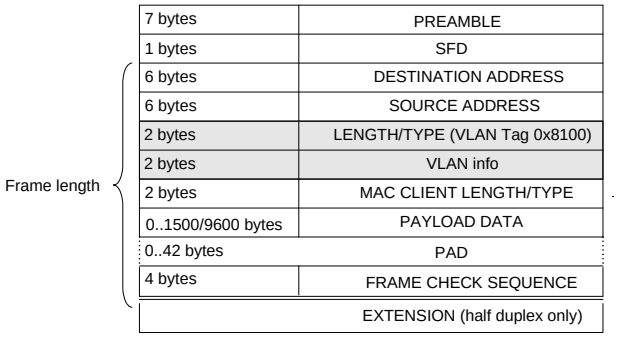
Figure 4–3. MAC Frame Format



The extension of a basic MAC frame is a virtual local area network (VLAN) tagged frame, which contains an additional 4-byte field for the VLAN tag and information between the MAC Source Address and Length/Type fields. VLAN tagging is defined by the IEEE Standard 802.1Q and used for scaling and securing Ethernet networks in enterprise and metro networks.

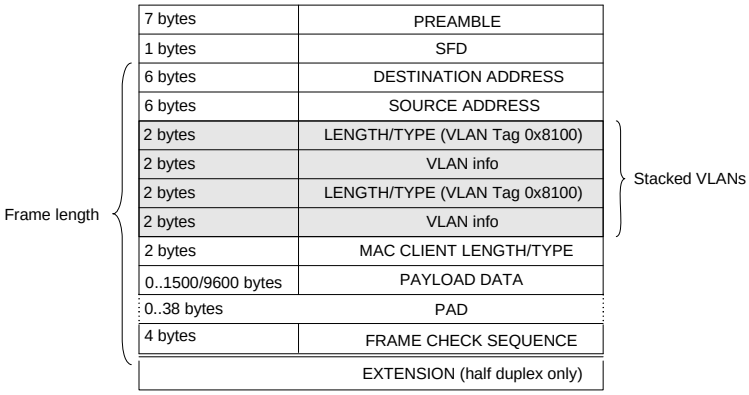
VLAN tagged frames have a maximum length of 1522 bytes, excluding the preamble and the SFD bytes. Figure 4–4 shows the format of a VLAN tagged frame.

Figure 4–4. VLAN Tagged MAC Frame Format



In metro Ethernet applications, which require more scalability and security due to the sharing of an Ethernet link by many service providers, MAC frames can be tagged with two consecutive VLAN tags (stacked VLAN). Stacked VLAN frames contain an additional 8-byte field between the MAC Source Address and MAC Client Length/Type fields, as illustrated in Figure 4–5.

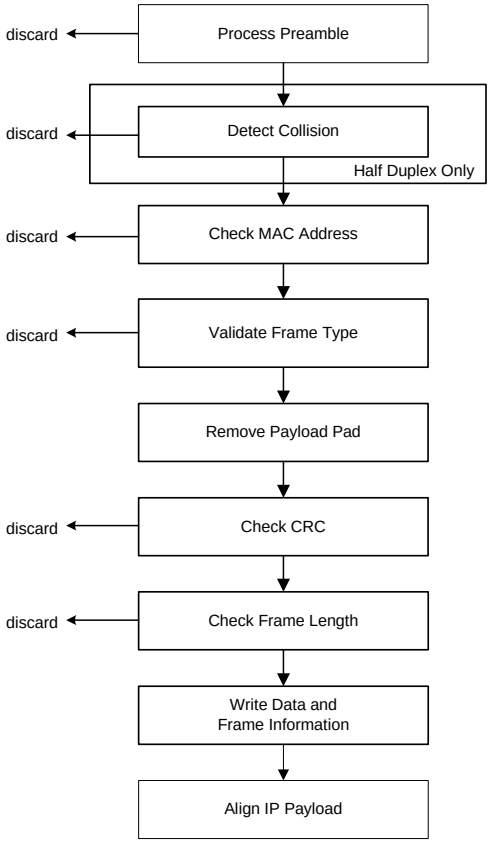
Figure 4–5. Stacked VLAN Tagged MAC Frame Format



MAC Receive Operation

Figure 4–6 illustrates the flow of the MAC receive operation.

Figure 4–6. MAC Receive Flow



The MAC receive interface on the system side complies with the Avalon-ST specification. For MACs with internal FIFOs, the ready latency on this interface is two. In SOPC Builder systems, however, the ready latency is reduced to zero because a timing adapter is automatically inserted. For MACs without internal FIFOs, the ready latency is always zero.

For more information about Avalon-ST interface protocol, refer to the *Avalon Interface Specifications*.

Preamble Processing

As specified by the IEEE Standard 802.3, the preamble can be up to 7 bytes long. The MAC function uses the SFD byte (`0xD5`) to identify the last byte of the preamble. If an SFD byte is not found after the maximum length allowed for the preamble, the MAC function rejects the frame and discards it.

The IEEE Standard specifies that frames must be separated by an interpacket gap (IPG) of at least 96 bit times. The MAC function, however, can accept frames with an IPG of less than 96 bit times; at least 48 and 64 bit times in RGMII/GMII (1000 Mbps operation) and RGMII/MII (10/100 Mbps operation) respectively.

The MAC function removes all preamble and SFD bytes from accepted frames.

Collision Detection in Half-Duplex Mode

In half-duplex mode, the MAC function checks for collisions during frame reception. When a collision is detected during the reception of the first 64 bytes, the MAC function discards the frame if the `RX_ERR_DISC` bit is set to 1. Otherwise, the frame is sent to the user application with an error.

MAC Address Checking

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

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

The MAC function always accepts broadcast frames. If promiscuous mode is enabled (`PROMIS_EN` bit in the `command_config` register = 1), address checking is omitted and all received frames are accepted.

Unicast Address Checking

If promiscuous mode is disabled, a frame is accepted only if its unicast destination address matches any of the following addresses:

- The primary MAC address, configured in the registers `mac_0` and `mac_1`.
- The supplemental MAC addresses, configured in the following 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`.

Otherwise, the MAC function discards the frame. For more information about the MAC address registers, refer to Table 4–9 on page 4–28.

If your system does not require the use of multiple addresses, Altera recommends that you configure all supplemental addresses to the primary MAC address.

Multicast Address Resolution

Multicast addresses can be resolved using either a software program running on the host processor or a hardware multicast address resolution engine. Using a software program to resolve multicast addresses, which is a typical implementation, can affect the overall system performance especially in gigabit mode.

The hardware multicast address resolution engine is implemented using a 64-entry hash table, as illustrated in Figure 4–7. The host processor must build the hash table according to the specified algorithm. A 6-bit code is generated from each multicast address by XORing the address bits as shown in Table 4–1 and Table 4–2. This code represents the address of an entry in the hash table, and a one must be written to the entry to indicate a valid multicast address represented by the 6-bit code. A zero indicates an invalid multicast address.

You can choose to generate the 6-bit code from all 48 bits of the destination address by setting the MHASH_SEL bit in the command_config register to 0, or from the lower 24 bits by setting the same bit to 1. The latter option is provided to omit the manufacturer's code, which typically resides in the upper 24 bits of the destination address, when generating the 6-bit code.

Figure 4–7. Hardware Multicast Address Resolution Engine

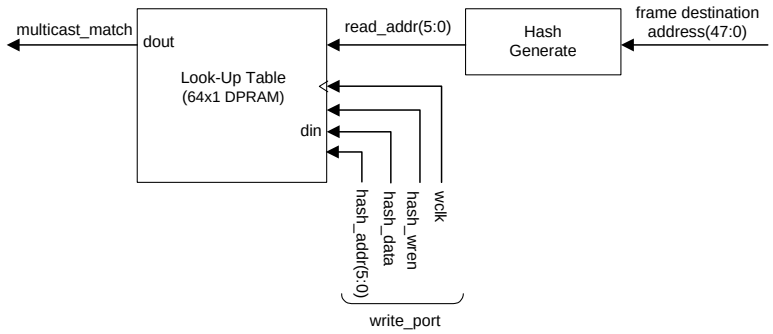


Table 4–1 shows the algorithm for generating the 6-bit code from the entire destination address.

Table 4–1. Hash Code Generation—Full Destination 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 4–2 shows the algorithm for generating the 6-bit code from the lower 24 bits of the destination address.

Table 4–2. Hash Code Generation—Lower 24 Bits of Destination 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 checks each multicast address received against the hash table, which serves as a fast matching engine, and a match is returned within one clock cycle. If there is no match, the MAC function discards the frame.

All multicast frames are accepted if all entries in the hash table are one.

Frame Type Validation

If the Length/Type field represents the frame type, the MAC function validates the type and processes each type accordingly. Invalid frame types are discarded.

VLAN Frame Processing

A value of 0x8100 in the Length/Type field denotes a VLAN tagged frame. A 2-byte VLAN tag follows the Length/Type field. VLAN tagged frames are received in the same manner as basic MAC frames, and the entire frame, including the VLAN tag, is forwarded to the user application. The MAC function asserts the signal rx_err_stat[16] (pkt_class_data[1] in MAC without internal FIFOs) to indicate that the current frame is a VLAN tagged frame.

The MAC function removes the padding from VLAN tagged frames only when the value of the MAC client Length/Type field, which comes after the VLAN control information field, is less than 42 and the PAD_EN bit in the command_config register is set to 1.

Stacked VLAN Frame Processing

A value of 0x8100 in the Length/Type field following the additional 2 bytes in VLAN tagged frames denotes a stacked VLAN tagged frame (See "MAC Frame Format" on page 4–2). The MAC function asserts the signal rx_err_stat[17] (pkt_class_data[0] in MACs without internal FIFOs) to indicate that the current frame is a stacked VLAN tagged frame.

The MAC function removes the padding from stacked VLAN tagged frames only when the value of the MAC client Length/Type field, which comes after the second VLAN tag field, is less than 38 and the PAD_EN bit in the command_config register is set to 1.

Pause Frame Termination

Pause frames are terminated within the MAC receive engine; they are not forwarded to the receive FIFO or the Avalon-ST interface. The MAC function determines if a pause frame is valid by checking its CRC and frame length. The pause quanta in a valid pause frame is extracted and forwarded to the MAC transmit engine. Invalid pause frames are ignored.

The statistics counter `aPAUSEMACCtrlFramesReceive` is incremented each time a valid pause frame is received.

Command Frame Filtering

A value of `0x8808` in the Length/Type field denotes command frames. The MAC function only accepts command frames with an opcode other than `0x0001` if the `CNTL_FRM_ENA` bit in the `command_config` register is set to 1. Accepted command frames are then forwarded to the receive FIFO or the Avalon-ST interface.

Pause frames, which are command frames with an opcode of `0x0001`, are always accepted. The `PAUSE_FWD` and `PAUSE_IGNORE` bits in the `command_config` register determine how pause frames are managed. See "Pause Frames" on page 4–20 for more information on pause frames.

Payload Pad Removal

The padding removal can be optional, depending on the payload length and the value of the `PAD_EN` bit in the `command_config` register.

The MAC function removes the padding, prior to sending the frames to the receive FIFO or Avalon-ST interface, when the `PAD_EN` bit is set to 1 and the payload length is less than the following values for the different frame types:

- 46 bytes for basic MAC frames
- 42 bytes for VLAN tagged frames
- 38 bytes for stacked VLAN tagged frames

If the `PAD_EN` bit is set to 0, complete frames including the padding are forwarded to the receive FIFO or the Avalon-ST interface.

CRC Checking

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

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

The 32-bit CRC value occupies the FCS field with X^{31} in the least significant bit of the first byte. The CRC bits are thus received in the following order: X^{31}, X^{30},..., X^{1}, X^{0}.

If a CRC-32 error is detected, the MAC function marks the frame invalid by asserting the signal `rx_err[2]` (`data_rx_error[1]` in MACs without internal FIFOs). Frames with such errors are discarded if the `RX_ERR_DISC` bit in the `command_config` register is set to 1.

The CRC-32 field is forwarded to the receive FIFO or Avalon-ST interface if the `CRC_FWD` and `PAD_EN` bits in the `command_config` register are 1 and 0, respectively.

Frame Length Checking

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

- The length of all frame types is not less than 64 bytes.
- The length of basic MAC frames is not greater than the maximum length specified in the `frm_length` register.
- The length of VLAN tagged frames is not greater than the maximum length specified in the `frm_length` register plus four.
- The length of stacked VLAN tagged frames is not greater than the maximum length specified in the `frm_length` register plus eight.

If the frame length is greater than the maximum length allowed, the MAC function marks the frame invalid by asserting the signal `rx_err[1]` (`data_rx_error[0]` in MACs without internal FIFOs). The MAC truncates the frame if its length is more than 11 bytes longer than the maximum length allowed to prevent FIFO overflow.

The MAC function keeps track of the actual frame payload length as it receives a frame. The actual frame payload length is checked against the Length/Type or MAC Client Length/Type field, depending on the frame type, when the `NO_LGTH_CHECK` bit in the `command_config` register is set to 0 and a valid frame length is received for the following frame types:

- Basic MAC frames—the Length/Type field is between `0x2E` (46 decimal) and `0x0600` (1536 decimal), excluding `0x600`.
- VLAN tagged frames—the MAC Client Length/Type field is between `0x2A` and `0x0600`, excluding `0x600`.
- Stacked VLAN tagged frames—the MAC Client Length/Type field is between `0x26` and `0x0600`, excluding `0x600`.

If the actual frame payload length and the length field do not match, the MAC function asserts the signal `rx_err[1]` (`data_rx_error [0]` in MAC without internal FIFOs) to indicate a length error. The frame is truncated if the actual payload is longer than the length in the Length/Type or MAC Client Length/Type field. If the `RX_ERR_DISC` bit in the command_config register is set to 1, frames with such errors are discarded.

Data and Frame Information Writing

If the MAC function is configured with internal FIFOs, the data and frame information are written to the internal FIFOs. Otherwise, they are directly forwarded to the Avalon-ST receive interface.

The MAC function's Avalon-ST receive interface does not support backpressure. If the receiving component is not ready to receive data from the MAC function, the frame in transmission is truncated with an error. Subsequent frames are dropped with an error.

IP Payload Alignment

The network stack makes frequent use of the IP addresses stored in Ethernet frames. If the option **Align packet headers to 32-bit boundaries** is turned on, the MAC aligns IP payload on a four-byte boundary by adding 2 bytes to the beginning of Ethernet frames. The padding of Ethernet frames are determined by the registers `tx_cmd_stat` and `rx_cmd_stat` on transmit and receive, respectively.

Table 4–3 illustrates the structure of a non-IP aligned Ethernet frame.

Table 4–3. 32-Bit Interface Data Structure — Non-IP aligned

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 4–4 illustrates the structure of an IP aligned Ethernet frame.

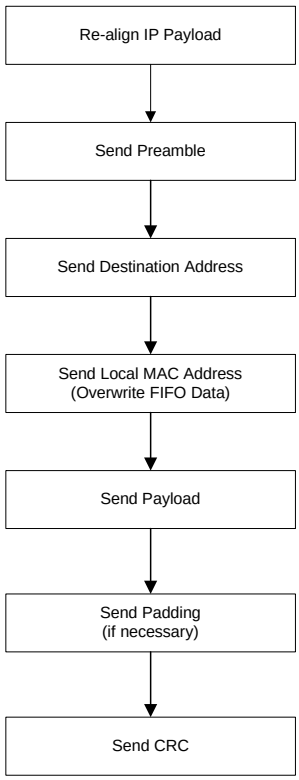
Table 4–4. 32-Bit Interface Data Structure — IP aligned

31…24	23…16	15…8	7…0
padded with zeros		Byte 0	Byte 1
Byte 2	Byte 3	Byte 4	Byte 5

MAC Transmit Operation

Figure 4–8 illustrates the flow of the MAC transmit operation.

Figure 4–8. MAC Transmit Flow



Frame transmission starts when the transmit FIFO holds enough data; a full packet in the store and forward mode or data equal to the transmit almost-full threshold otherwise.

The following tasks are initiated during frame transmission:

- Generates preamble and SFD field before frame transmission
- 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)
- 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 transmission (XOFF) if a non-zero pause quanta is received from the MAC receive path

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

- Collision detection
- Frame retransmission when the backoff timer expires

The MAC transmit interface on the system side complies with the Avalon-ST specification. This interface allows for a variable-length ready latency by configuring the `tx_almost_full` register (See Table 4–7). The `valid` signal must remain asserted throughout the reception of an entire frame on this interface. Otherwise, the frame will be truncated and forwarded to the ethernet-side interface with an error.

For more information about Avalon-ST interface protocol, refer to the *Avalon Interface Specifications*.

IP Payload Re-alignment

If the option **Align packet headers to 32-bit boundaries** is turned on, the MAC function removes the additional two bytes from the beginning of Ethernet frames. See "IP Payload Alignment" on page 4–10 for more information about IP payload alignment.

MAC Address Insertion

If MAC address insertion is enabled (`TX_ADDR_INS` bit in the `command_config` register = 1), the source MAC address in frames received from the MAC transmit FIFO is replaced with either the MAC primary address or any of the supplemental MAC addresses, as specified by the `TX_ADDR_SEL` bits in the `command_config` register. For more information about the MAC address selection, refer to Table 4–10 on page 4–35.

If MAC address insertion is disabled (`TX_ADDR_INS` bit in the `command_config` register = 0), the source MAC address is forwarded to the Ethernet-side interface.

Frame Payload Padding

The IEEE Standard 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

If a frame is sent to the MAC function with the signal ff_tx_crc_fwd set to 0, the MAC function generates the CRC-32 field and appends it to the end of the frame.

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

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

The 32-bit CRC value occupies the FCS field with X^{31} in the least significant bit of the first byte. 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 configured in the tx_ipg_length register is maintained between transmissions. The minimum IPG can be configured 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

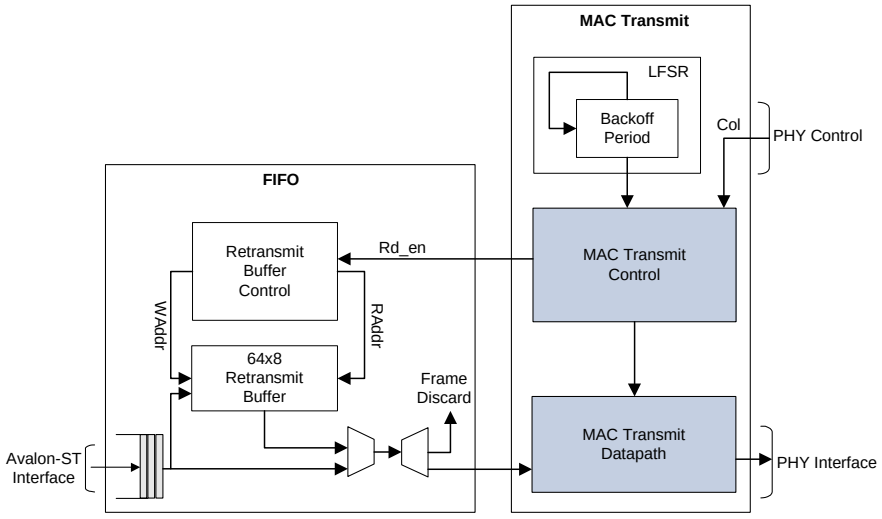
A collision occurs only in a half-duplex network. It occurs when two or more nodes transmit concurrently. During transmission, the MAC function monitors the line condition and detects a collision when a PHY device asserts the m_rx_col signal.

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 frame 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 frame retransmission.

Figure 4–9. Frame 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 *Nth* 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 during 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 Standard 802.3.

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 frame from the FIFO.

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

Transmit and Receive Latency

Altera uses the following definitions for the MAC transmit and receive latencies:

- Transmit latency is the number of MAC clock cycles the MAC function takes to transmit the first bit on the network-side interface (MII/GMII/RGMII) after the bit was first available on the Avalon-ST interface.
- Receive latency is the number of MAC clock cycles the MAC function takes to present the first bit on the Avalon-ST interface after the bit was received on the network-side interface (MII/GMII/RGMII).

Table 4–5 shows the transmit and receive nominal latencies in various modes. The FIFO threshold values are set to the recommended values specified in this user guide when deriving the latencies. See "Complete MAC Interface Register Map" on page 4–28 for the recommended threshold values.

Table 4–5. MAC Transmit and Receive Nominal Latency

MAC Configuration	Latency (Clock Cycles) *(1)*	
	Transmit	Receive
MAC with Internal FIFOs *(2)*		
GMII in cut-through mode	32	110
MII in cut-through mode	41	218
RGMII in gigabit and cut-through mode	33	113
RGMII in 10/100 Mbps and cut-through mode	42	221
MAC without Internal FIFOs *(3)*		
GMII	11	37
MII	22	77
RGMII in gigabit mode	12	40
RGMII in10/100 Mbps	23	80

Note to Table 4–5:

(1) Both the system and reference clocks are running at the same frequency.
(2) The data width is set to 32 bits.
(3) The data width is set to 8 bits.

MAC FIFO Thresholds

For MACs with internal FIFOs, you can change the FIFO operations and manage potential FIFO overflow or underflow by configuring the following thresholds:

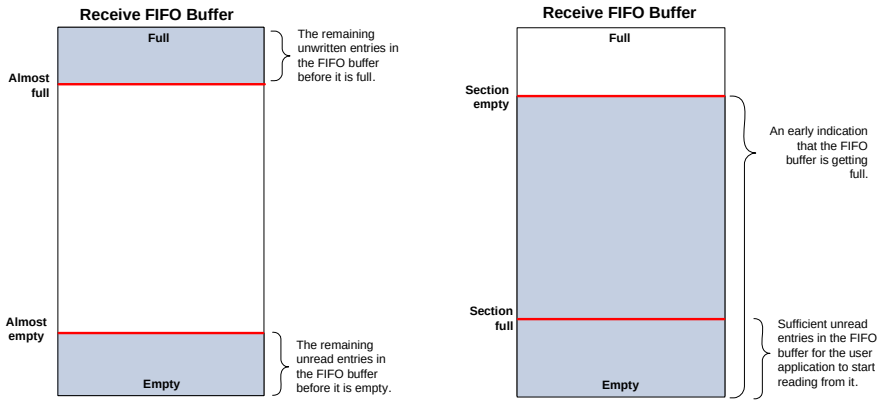
- Almost empty
- Almost full
- Section empty
- Section full

The thresholds are defined in bytes and words for 8-bit and 32-bit wide FIFOs, respectively.

Receive FIFO Thresholds

Figure 4–10 illustrates the receive FIFO thresholds.

Figure 4–10. Receive FIFO Thresholds



The receive FIFO thresholds are configured via the registers. Table 4–6 describes how each threshold can be used to change and manage FIFO operations.

Table 4–6. Receive Threshold Registers Description (Part 1 of 2)

Threshold	Register Name	Description
Almost empty	`rx_almost_empty`	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. When an EOP is detected, the MAC function transfers all data to the user application even if the number of unread entries is below this threshold.
Almost full	`rx_almost_full`	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 (`ff_rx_rdy` is 0), the MAC receive control performs the following operations: ■ Stops writing data to the FIFO ■ Truncates received frames to avoid FIFO overflow ■ Asserts the signal `rx_err[0]` simultaneously with the assertion of the signal `ff_rx_eop`. ■ Marks the truncated frame invalid by setting `rx_err[3]` to 1. If the `RX_ERR_DISC` bit in the `command_config` register is set to 1 and the section-full (`rx_section_full`) threshold is set to 0, erroneous frames received on the FIFO interface are discarded.

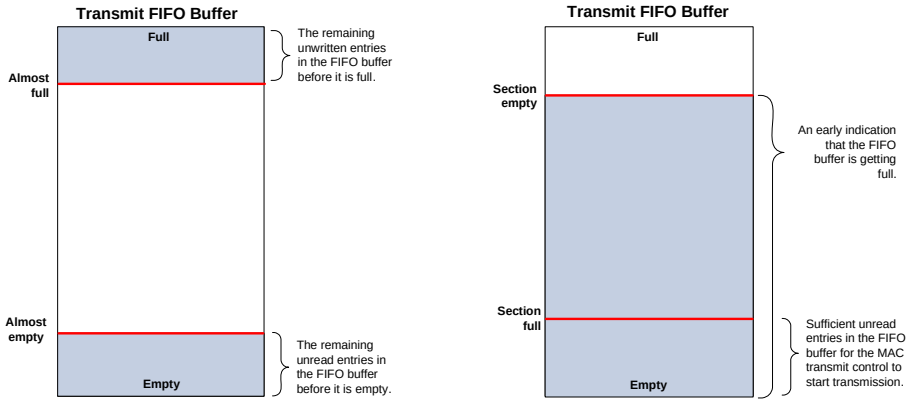
Table 4–6. Receive Threshold Registers Description (Part 2 of 2)

Threshold	Register Name	Description
Section empty	rx_section_empty	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. 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.
Section full	rx_section_full	The section-full threshold indicates that there are sufficient entries in the FIFO for the user application to start reading from it. The signal ff_rx_dsav is asserted when the FIFO level reaches the section-full threshold. Set this threshold to 0 to enable store and forward on the receive datapath. In the store and forward more, the signal ff_rx_dsav remains deasserted. The signal ff_rx_dval is asserted as soon as a complete frame is written to the FIFO.

Transmit FIFO Thresholds

Figure 4–11 illustrates the transmit thresholds.

Figure 4–11. Transmit FIFO Thresholds



The transmit FIFO thresholds are configured via the registers. Table 4–7 describes how each threshold can be used to change and manage FIFO operations.

Table 4–7. Transmit 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 deasserts the `ff_tx_rdy` signal to backpressure the FIFO interface.
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 `ff_tx_septy` 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.

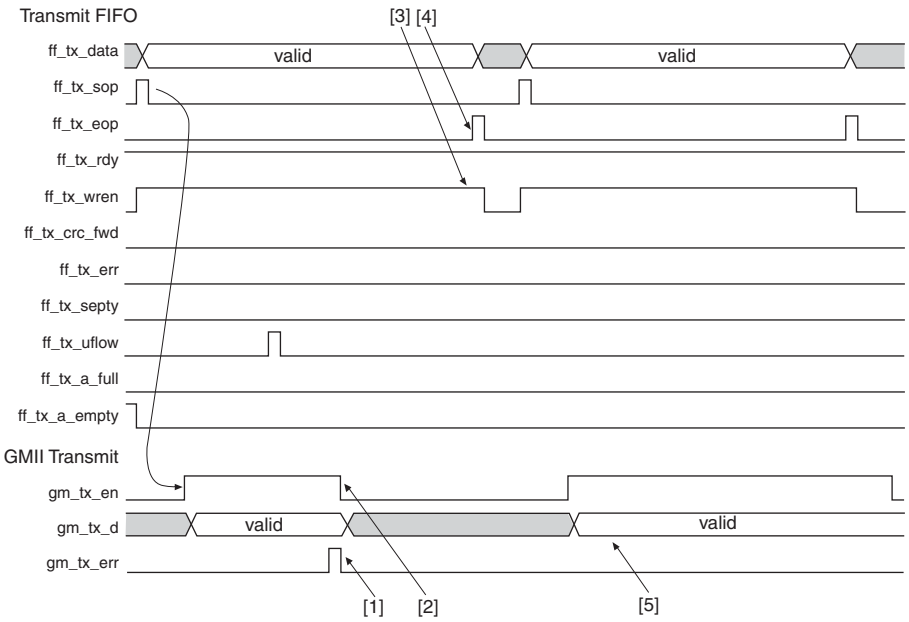
Transmit FIFO Underflow

During a frame transmission, 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 0 to terminate the frame transmission.
3. After the underflow, the application completes the frame transmission.
4. The MAC transmit control discards any new data in the FIFO until the end of frame 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 4–12 illustrates the FIFO underflow protection algorithm for gigabit Ethernet system.

Figure 4–12. Transmit FIFO Underflow



MAC Full Duplex Flow Control Operation

The MAC flow control manages three congestion types:

- 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.
- Receive FIFO Congestion—When the receive FIFO is getting almost full, 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 transmission 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.

Receive FIFO and Local Device Congestion

The MAC transmit control generates pause frames when the receive FIFO level hits a certain level, or at the request of the user application. If an internal receive FIFO is in use, the MAC function triggers pause frame generation (XOFF frames) when the FIFO level reaches the section-empty threshold (`rx_section_empty`) and the current frame transmission completes. The fill level of an external FIFO is obtained via the Avalon-ST receive FIFO status interface.

User applications can force a pause frame generation by setting the `XOFF_GEN` bit in the `command_config` register to 1 or asserting the `xoff_gen` signal.

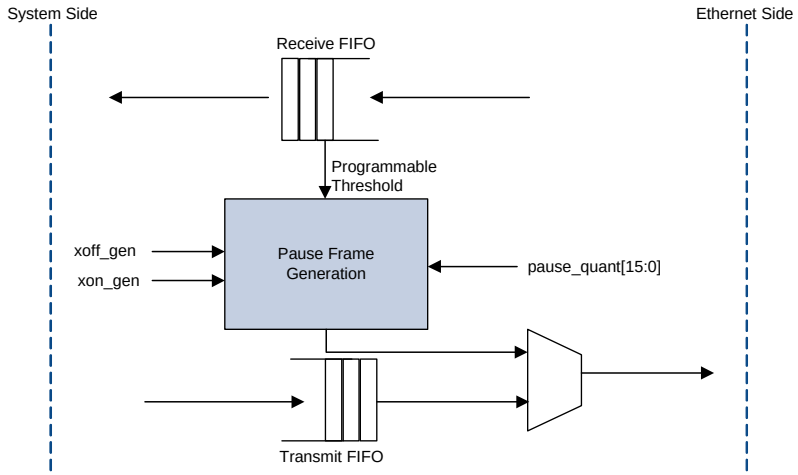
When an XOFF pause frame is generated, the pause quanta bytes P1 and P2 (see "Pause Frames" on page 4–20) are filled with the value configured in the `pause_quant` register. The source address is set to the primary MAC address configured in the `mac_0` and `mac_1` registers, and the destination address is set to a fixed multicast address, 01-80-C2-00-00-01 (0x010000c28001).

An XON pause frame is generated automatically when the FIFO section-empty flag is deasserted and the current frame transmission is completed. User applications can pause an XON pause frame generation by clearing the `XOFF_GEN` bit and signal, and subsequently set the `XON_GEN` bit to 1 or asserting the `XON_GEN` signal.

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 primary 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 Standard 802.3 annex 31A & B. For more information on pause frames, refer to "Pause Frames" on page 4–20.

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.

Figure 4–14 shows the format of pause frames. The Length/Type field has a fixed value of 0x8808, followed by a 2-byte opcode field of 0x0001. A 2-byte pause quanta is defined in the frame payload bytes 2 (P1) and 3 (P2). The pause quanta, P1, is the most significant. A pause frame has no payload length field, and is always padded with 42 bytes of 0x00.

Figure 4–14. Pause Frame Format

7 bytes	PREAMBLE
1 bytes	SFD
6 bytes	DESTINATION ADDRESS
6 bytes	SOURCE ADDRESS
2 bytes	TYPE (0x8808)
2 bytes	OPCODE (0X0001)
2 bytes	PAUSE QUANTA ({0xP1, 0xP2})
42 bytes	PAD
4 bytes	CRC

If a pause frame with a pause quanta greater than zero is received, the MAC function completes the current frame transmission, and subsequently suspends data transmission for a duration specified by the pause quanta. One pause quanta fraction is equivalent to 512 bit times.

Data transmission resumes when a pause quanta of zero is received.

Magic Packets

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

If magic packet detection is enabled (`MAGIC_ENA` bit in the `command_config` register = 1), you can put a node to sleep by setting the `SLEEP` bit in the `command_config` register to 1. The following operations are disabled when a node is put to sleep:

- MAC transmit
- MAC FIFO receive and transmit

The MAC receive remains enabled, but it ignores all traffic from the line except magic frames. This allows a remote agent to wake up the node.

Magic Packet Detection

Magic packet detection wakes up a node which was put to sleep. The MAC function detects magic frames with any of the following addresses in the destination address field:

- Any multicast address
- A broadcast address
- The primary MAC address configured in the `mac_0` and `mac_1` registers
- Any of the supplemental MAC addresses configured in the following registers if they are enabled: `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`

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

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.

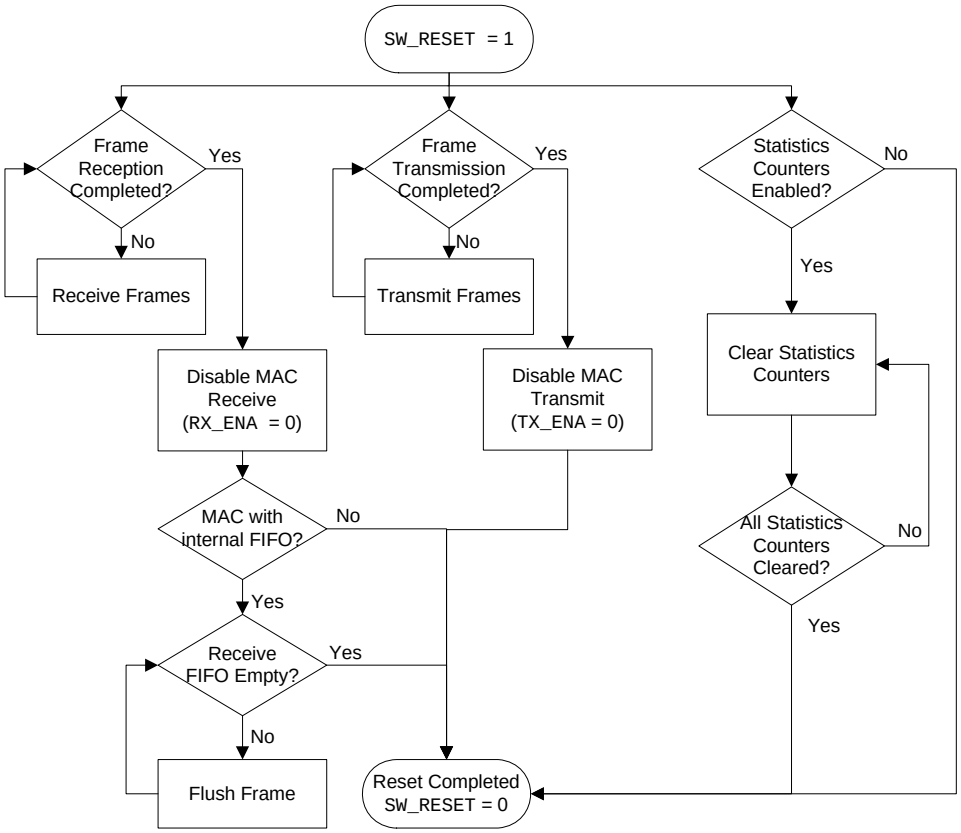
Software Reset

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 4–35.

Figure 4–15 illustrates the sequence of a software reset.

Figure 4–15. 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.

Local Loopback

You can enable a local loopback on the MAC MII/GMII/RGMII to exercise the MAC transmit and receive paths. If you enable local loopback, use the same clock source for both the transmit and receive clocks. If you use different clock sources, ensure that the difference between the transmit and receive clocks is less than ±100 ppm.

To enable a local loopback, perform the following steps:

1. Initiate software reset by setting the SW_RESET bit in command_config register to 1.

   Software reset disables the transmit and receive operations, flushes the internal FIFOs, and clears the statistics counters. The SW_RESET bit is automatically cleared upon completion.

2. When software reset is complete, enable local loopback on the MAC's MII/GMII/RGMII by setting the LOOP_ENA bit in command_config register to 1.

3. Enable transmit and receive operations by setting the TX_ENA and RX_ENA bits in command_config register to 1.

4. Initiate frame transmission.
5. Compare the statistics counters `aFramesTransmittedOK` and `aFramesReceivedOK` to verify that the transmit and receive frame counts are equal.
6. Check the statistics counters `ifInErrors` and `ifOutErrors` to determine the number of packets transmitted and received with errors.
7. To disable loopback, initiate a software reset and set the `LOOP_ENA` bit in `command_config` register to 0.

MAC System-Side Interface

The system-side interface of the MAC function consists of two Avalon-ST ports; an Avalon-ST source port that provides an interface to the MAC receive function and an Avalon-ST sink port that provides an interface to the MAC transmit function.

In multi-port MACs, two additional Avalon-ST ports are implemented; an Avalon-ST source port that streams out receive packet classification information and an Avalon-ST sink port that streams in FIFO fill level.

In addition to the Avalon-ST signals, a few component-specific signals are also implemented on the MAC system-side interface. These component-specific signals are not associated with either Avalon-ST port and not accessible in SOPC Builder systems. See the section "Signals" on page 4–73 for pinout diagrams and signal descriptions.

For more information about the Avalon-ST interface protocol, refer to the *Avalon Interface Specifications*.

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.

Avalon-ST Receive Interface

The Avalon-ST receive interface has the following properties:

- In multi-port MACs, the data width is fixed to 8 bits. In other core variations, the data width can be set to 8 or 32 bits using the **Width** parameter. See "FIFO Options" on page 3–5.
- In multi-port MACs, backpressure is not implemented. In other core variations, backpressure is supported; transmission stops when the FIFO level reaches the respective programmable threshold.
- Supports packet using start- and end-of-packet signals, and partial final packet signals.
- Error reporting.

Figure 4–16 shows the receive operation for MAC core variations with internal FIFOs.

Figure 4–16. Receive Operation—MAC With Internal FIFOs

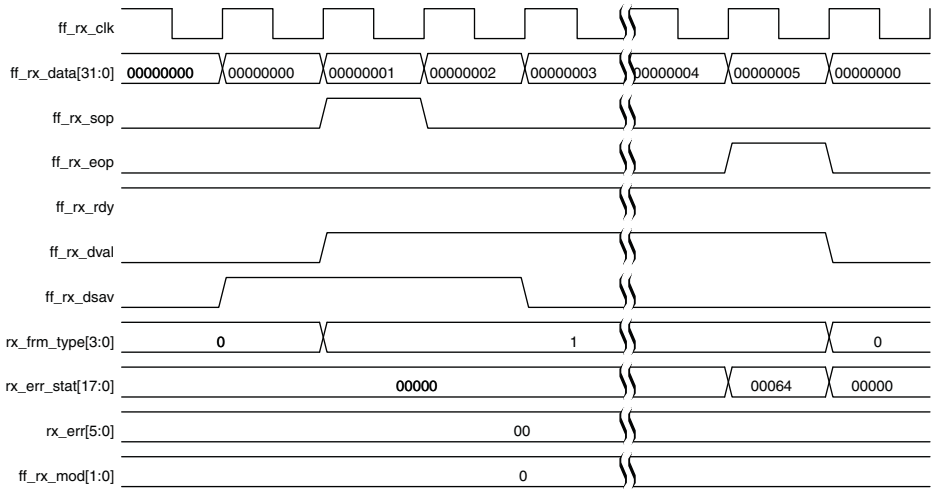


Figure 4–17 shows the receive operation for MAC core variations without internal FIFOs.

Figure 4–17. Receive Operation—MAC Without Internal FIFOs

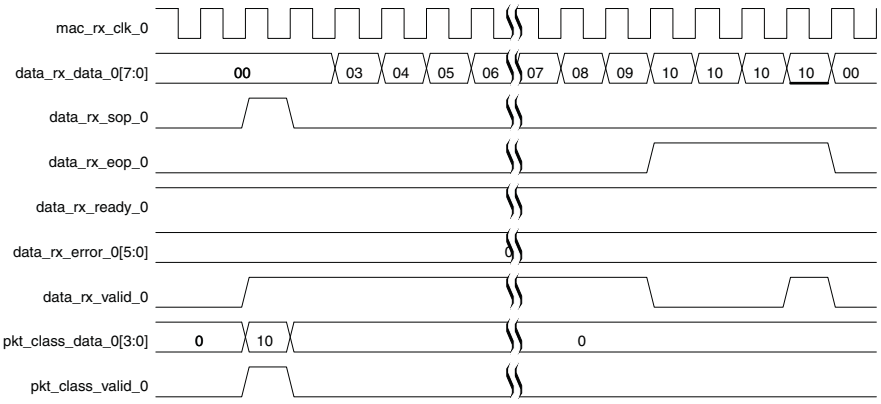


Figure 4–18 depicts an invalid length error during a receive operation for MAC core variations with internal FIFOs.

Figure 4–18. Invalid Length Error—MAC With Internal FIFO

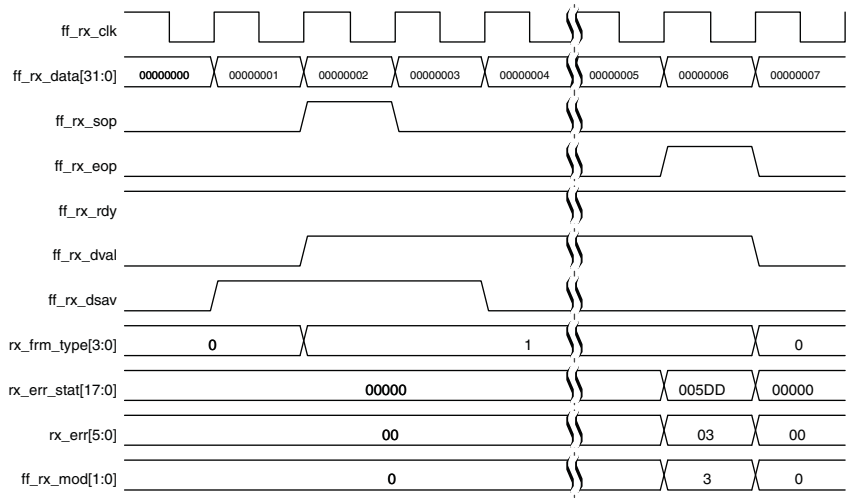
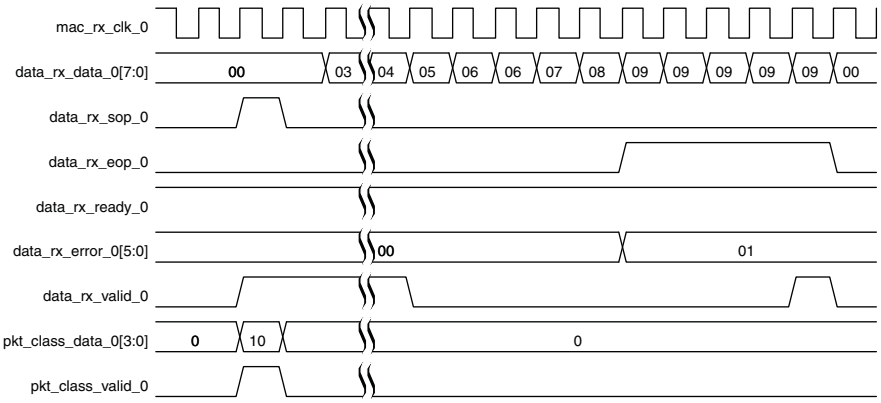


Figure 4–19 depicts an invalid length error during a receive operation for MAC core variations without internal FIFOs.

Figure 4–19. Invalid Length Error—MAC Without Internal FIFOs



Avalon-ST Transmit Interface

The Avalon-ST transmit interface has the following properties:

- Low latency, high throughput data transfer.
- In multi-port MACs, the data width is fixed to 8 bits. In other core variations, the data width can be set to 8 or 32 bits using the **Width** parameter. See "FIFO Options" on page 3–5.
- Supports packet using start- and end-of-packet signals, and partial final packet signals.
- Error reporting.
- Optional CRC calculation.

Figure 4–20 shows the transmit operation for MAC core variations with internal FIFOs.

Figure 4–20. Transmit Operation—MAC With Internal FIFOs

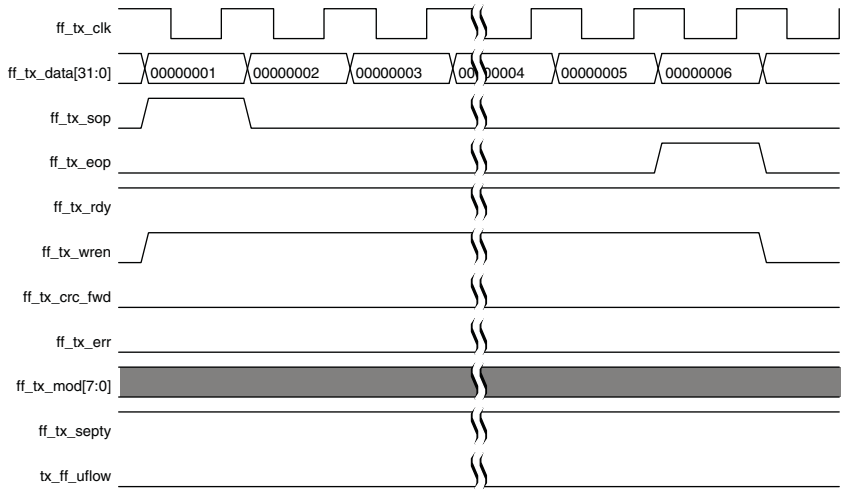
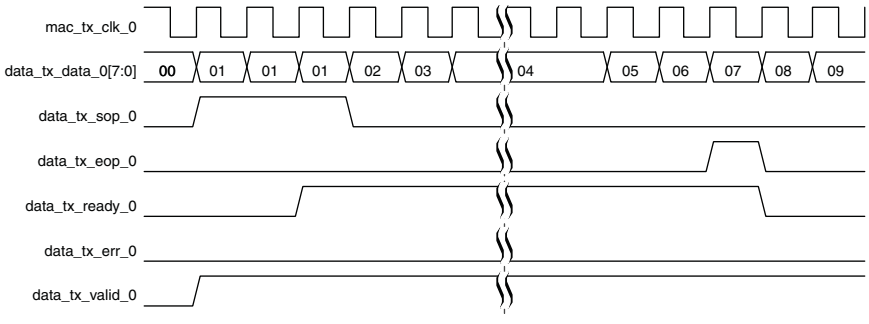


Figure 4–21 shows the transmit operation for MAC core variations without internal FIFOs.

Figure 4–21. Transmit Operation—MAC Without Internal FIFOs



MAC Control Interface

The control interface to the Triple Speed Ethernet MegaCore function is an Avalon-MM slave port which provides access to a register space of 256 registers. This interface controls the MAC block as well as the PCS block in the MegaCore function. User applications can change the behavior of the MAC and PCS function by writing to the registers.

In multi-port MACs, a contiguous register space is allocated for all instances and accessed via a common control interface. For example, if the register space base address for the first instance is 0x0000, the base address for the next instance is 0x0400 and so forth. Some of the registers can be shared among the ports. The shared registers occupy the register space of the first port. Updating these registers in the register space of other ports has no effect on the configuration.

The Avalon-MM slave port has the following properties:

- 32-bit readdata and writedata signals
- 8-bit address signal, which provides access to 256 32-bit registers
- Variable wait states

Table 4–8 provides an overview of the MAC function's register space.

Table 4–8. Overview of MAC Registers (Part 1 of 2)

Address Offset	Section	Description
0x000 – 0x05C	MAC Block Configuration	Base register settings to configure the MAC function. In a multi-port MAC, the following registers in this block are shared among all instances: ■ `rev` ■ `scratch` ■ `frm_length` ■ `pause_quant` ■ `mdio_addr0` and `mdio_addr1` ■ `tx_ipg_length`
0x060 – 0x0DC	Statistics Counters	Counters collecting traffic statistics.
0x0E8 – 0x0EB	TX Command Status Register	Transmit datapath control register. See table Figure 4–24 and Table 4–13 on page 4–40 for register format and bits description.
0x0EC – 0x0EF	RX Command Status Register	Receive datapath control register. See table Figure 4–25 and Table 4–13 on page 4–40 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. See "PCS Control Interface" on page 4–61 for more information on PCS registers.
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.

Table 4–8. Overview of MAC Registers (Part 2 of 2)

Address Offset	Section	Description
0x320 – 0x3FC	Reserved *(1)*	Unused.

Note to Table 4–8:

(1) Altera recommends that you set all bits in reserved registers to 0 and ignore them on reads.

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 4–9. 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 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 (RO), write it (WO) or read and write it (RW).

Table 4–9. MAC Register Map (Part 1 of 7)

Address Offset	Name	Description	Access	HW Reset	SW Reset
0x000	`rev`	MegaCore function revision. This register is divided into two 16-bit fields: ■ Bits 15:0: MegaCore function revision, set to the current version of the MegaCore function. ■ Bit 31:16: Customer specific revision, set to 0 during MegaCore function configuration. This field is controlled by the parameter `CUST_VERSION` defined in the top level generated for the Triple Speed Ethernet MegaCore function instance.	RO	0x000 00810	—
0x004	`scratch` *(1)*	Scratch register. Provides a memory location for user applications to test the device memory operation.	RW	0	—
0x008	`command_config`	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

Table 4–9. MAC Register Map (Part 2 of 7)

Address Offset	Name	Description	Access	HW Reset	SW Reset
0x00C	mac_0	MAC address. MAC addresses are 6 bytes long. The first four most signficant bytes of the MAC address occupy mac_0 in reverse order. The last two bytes of the MAC address occupy the two least signficant bytes of mac_1 in reverse order. For example, if the MAC address is 00-1C-23-17-4A-CB, the following assignments are made: mac_0 = 0x17231c00 mac_1 = 0x0000CB4a	RW	0	—
0x010	mac_1		RW	0	—
0x014	frm_length	16-bit maximum frame length in bytes. The MAC receive logic uses this value to check frames. Typical value is 1518. Bits 16 to 31 are reserved. This register is set to 1518 in 10/100 and 1000 Small MAC core variations.	RW	1518	—
0x018	pause_quant	16-bit 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. 10/100 and 1000 Small MAC core variations do not support flow control.	RW	0	—
0x01C	rx_section_empty	Variable-length receive FIFO section-empty threshold. The length is determined by the FIFO depth. Due to internal pipeline latency, set the threshold to a value greater than 3. It is typically set to (FIFO Depth – 16). This threshold is set to a fixed value of (FIFO Depth – 16) in 10/100 and 1000 Small MAC core variations .	RW	0	—
0x020	rx_section_full	Variable-length receive FIFO section-full threshold. The length is determined by the FIFO depth. Due to internal pipeline latency, set the threshold to a value greater than 3. It is typically set to 16. This threshold is set to a fixed value of 16 in 10/100 and 1000 Small MAC core variations.	RW	0	—
0x024	tx_section_empty	Variable-length transmit FIFO section-empty threshold. The length is determined by the FIFO depth. Due to internal pipeline latency, set the threshold to a value greater than 3. It is typically set to (FIFO Depth – 16). This threshold is set to a fixed value of (FIFO Depth – 16) in 10/100 and 1000 Small MAC core variations.	RW	0	—

Table 4–9. MAC Register Map (Part 3 of 7)

Address Offset	Name	Description	Access	HW Reset	SW Reset
0x028	tx_section_full	Variable-length transmit FIFO section-full threshold. The length is determined by the FIFO depth. Due to internal pipeline latency, set the threshold to a value greater than 3. It is typically set to 16. This threshold is set to a fixed value of 16 in 10/100 and 1000 Small MAC core variations.	RW	0	—
0x02C	rx_almost_empty	Variable-length receive FIFO almost-empty threshold. The length is determined by the FIFO depth. When this register is set to 0, the MAC function never asserts the signal ff_rx_a_empty. Due to internal pipeline latency, set the threshold to a value greater than 3. It is typically set to 8. This threshold is set to a fixed value of 8 in 10/100 and 1000 Small MAC core variations.	RW	0	—
0x030	rx_almost_full	Variable-length receive FIFO almost-full threshold. The length is determined by the FIFO depth. When this register is set to 0, the MAC function never asserts the signal ff_rx_a_full. Due to internal pipeline latency, set the threshold to a value greater than 3. It is typically set to 8. This threshold is set to a fixed value of 8 in 10/100 and 1000 Small MAC core variations.	RW	0	—
0x034	tx_almost_empty	Variable-length transmit FIFO almost-empty threshold. The length is determined by the FIFO depth. When this register is set to 0, the MAC function never asserts the signal ff_tx_a_empty. Due to internal pipeline latency, set the threshold to a value greater than 3. It is typically set to 8. This threshold is set to a fixed value of 8 in 10/100 and 1000 Small MAC core variations.	RW	0	—
0x038	tx_almost_full	Variable-length transmit FIFO almost-full threshold. The length is determined by the FIFO depth. When this register is set to 0, the MAC function never asserts the signal ff_tx_a_full. A value of 3 indicates 0 ready latency; a value of 4 indicates 1 ready latency, and so forth. This register is typically set to a value greater than or equal to 3. It is typically set to 3. This threshold i set to a fixed value of 3 in 10/100 and 1000 Small MAC core variations.	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	—

Table 4–9. MAC Register Map (Part 4 of 7)

Address Offset	Name	Description	Access	HW Reset	SW Reset
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. This register is set to a fixed value of 12 in 10/100 and 1000 Small MAC core variations.	RW	0	—
0x060 0x064	aMacID	This register is wired to mac_0 and mac_1 MAC addresses respectively.	RO	0	—
0x068	aFramesTransmittedOK	See Table 4–14 on page 4–40.	RO	0	0
0x06C	aFramesReceivedOK	See Table 4–14 on page 4–40.	RO	0	0
0x070	aFrameCheckSequenceErrors	See Table 4–14 on page 4–40.	RO	0	0
0x074	aAlignmentErrors	See Table 4–14 on page 4–40.	RO	0	0
0x078	aOctetsTransmittedOK	Lower 32 bits of aOctetsTransmittedOK. The upper 32 bits of this statistics counter reside at address offset 0x0F0. See Table 4–14 on page 4–40 for more information on this counter.	RO	0	0
0x07C	aOctetsReceivedOK	Lower 32 bits of aOctetsReceivedOK. The upper 32 bits of this statistics counter reside at address offset 0x0F4. See Table 4–14 on page 4–40 for more information on this counter.	RO	0	0
0x080	aTxPAUSEMACCtrlFrames	Number of transmitted pause frames. See Table 4–14 on page 4–40.	RO	0	0
0x084	aRxPAUSEMACCtrlFrames	Number of received pause frames. See Table 4–14 on page 4–40.	RO	0	0
0x088	ifInErrors	See Table 4–15 on page 4–41.	RO	0	0
0x08C	ifOutErrors	See Table 4–15 on page 4–41.	RO	0	0
0x090	ifInUcastPkts	See Table 4–15 on page 4–41.	RO	0	0

Table 4–9. MAC Register Map (Part 5 of 7)

Address Offset	Name	Description	Access	HW Reset	SW Reset
0x094	ifInMulticast Pkts	See Table 4–15 on page 4–41.	RO	0	0
0x098	ifInBroadcast Pkts	See Table 4–15 on page 4–41.	RO	0	0
0x09C	ifOutDiscards	See Table 4–17 on page 4–43.	RO	0	0
0x0A0	ifOutUcastPkts	SeeTable 4–15 on page 4–41.	RO	0	0
0x0A4	ifOutMulticast Pkts	See Table 4–15 on page 4–41.	RO	0	0
0x0A8	ifOutBroadcast Pkts	See Table 4–15 on page 4–41.	RO	0	0
0x0AC	etherStatsDrop Events	See Table 4–16 on page 4–42.	RO	0	0
0x0B0	etherStatsOctets	Lower 32 bits of etherStatsOctets. The upper 32 bits of this statistics counter reside at address offset 0x0F8. See Table 4–16 on page 4–42.	RO	0	0
0x0B4	etherStatsPkts	See Table 4–16 on page 4–42.	RO	0	0
0x0B8	etherStatsUnder sizePkts	See Table 4–16 on page 4–42.	RO	0	0
0x0BC	etherStatsOver sizePkts	See Table 4–16 on page 4–42.	RO	0	0
0x0C0	etherStatsPkts 64Octets	See Table 4–16 on page 4–42.	RO	0	0
0x0C4	etherStatsPkts65 to127Octets	See Table 4–16 on page 4–42.	RO	0	0
0x0C8	etherStatsPkts128 to255Octets	See Table 4–16 on page 4–42.	RO	0	0
0x0CC	etherStatsPkts 256to511Octets	See Table 4–16 on page 4–42.	RO	0	0
0x0D0	etherStatsPkts 512to1023Octets	See Table 4–16 on page 4–42.	RO	0	0
0x0D4	etherStatsPkts 1024to1518Octets	See Table 4–16 on page 4–42.	RO	0	0
0x0D8	etherStatsPkts 1519toXOctets	Any frame length from 1519 to the maximum length configured in the frm_length register, if it is greater than 1518.	RO	0	0
0x0DC	etherStatsJabbers	Too long frames with CRC error.	RO	0	0

Table 4–9. MAC Register Map (Part 6 of 7)

Address Offset	Name	Description	Access	HW Reset	SW Reset
0x0E0	etherStats Fragments	Too short frames with CRC error.	RO	0	0
0x0E4	Reserved	Unused	RO	—	—
0x0E8	tx_cmd_stat	Transmit FIFO control register. The value in the HW Reset column is valid only when the synthesis option **Align packet headers to 32-bit boundaries** is selected. Otherwise, this register is set to 0x00 after a HW Reset.	RW	0x000 40000	—
0x0EC	rx_cmd_stat	Receive FIFO control register. The value in the HW Reset column is valid only when the synthesis option **Align packet headers to 32-bit boundaries** is selected. Otherwise, this register is set to 0x00 after a HW Reset.	RW	0x020 00000	—
0x0F0	msb_aOctetsTransmittedOK	Upper 32 bits of the respective statistics counters. By default all statistics counters are 32 bits wide. These statistics counters can be extended to 64 bits by turning on the option **Enable 64-bit byte counters**.	RO	0	0
0x0F4	msb_aOctetsReceivedOK		RO	0	0
0x0F8	msb_etherStats Octets		RO	0	0
0x0FC – 0x0FF	Reserved	Unused.	RO	—	—
0x100 – 0x1FC	hash_table	Multicast address resolution table, mapped in the controller address space. When programming the table, only bit 0 is significant. 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 5:0) are accepted by the MAC. If a 0 is written, matching multicast addresses are rejected. Hashing is not supported in 10/100 and 1000 Small MAC core variations.	WO	0	—
0x200 – 0x27C	PHY Device 0 Internal Registers	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. 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. For all registers, bits 15:0 are significant. Bits 31:16 should be written with 0 and ignored on read.	RW	—	—

Table 4–9. MAC Register Map (Part 7 of 7)

Address Offset	Name	Description	Access	HW Reset	SW Reset
0x280 – 0x2FC	PHY Device 1 Internal Registers	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. 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. For all registers, bits 15..0 are significant. Bits 31:16 should be written with 0 and ignored on read.	RW	—	—
0x300	`smac_0_0` *(1)*	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` *(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	—
0x308	`smac_1_0` *(1)*	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` *(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` *(1)*	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` *(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` *(1)*	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` *(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	—	—	—	—

Note to Table 4–12:

(1) Register is not available in 10/100 and 1000 Small MAC core variations.

Command_Config Register

Figure 4–22 shows the bits and fields that comprise the `command_config` register.

Figure 4–22. Command_Config Register

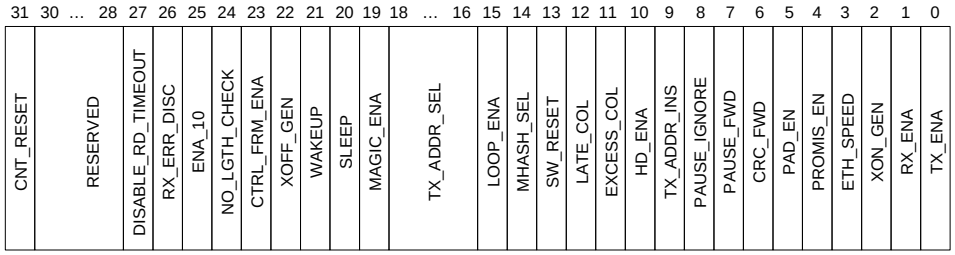


Table 4–10 describes the function of each bit and field in the `command_config` register.

Table 4–10. 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 `SW_RESET` 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 the `SW_RESET` 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. ■ Setting this bit to 1 enables gigabit Ethernet operation. The signal `set_1000` is masked and does not affect the operation. ■ If this bit is set to 0, gigabit Ethernet operation is enabled only if the signal `set_1000` is asserted. Otherwise, the MAC function operates in 10/100 Mbps Ethernet mode. ■ When the MAC operates in gigabit mode, the output signal `eth_mode` 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.
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.

Table 4–10. Command_Config Register Bit Descriptions (Part 2 of 4)

Bit(s)	Bit Name	Access *(1)*	Description
6	CRC_FWD	RW	CRC forwarding. ■ 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 and a padded frame is received. In this case, the MAC function checks the CRC field and removes the checksum and padding from the frame before forwarding the frame to the user application.
7	PAUSE_FWD	RW	Pause frame forwarding. Terminates or forwards pause frames. ■ 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. ■ 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. ■ 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. ■ 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. This is only true when the ETH_SPEED bit is set to 1.
11	EXCESS_COL	RO	Excessive collision condition. ■ This bit is set to 1 when the MAC function discards a frame after detecting a collision on 16 consecutive frame retransmissions. ■ This bit is cleared following a hardware or software reset. See the SW_RESET bit description.
12	LATE_COL	RO	Late collision condition. ■ 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.

Table 4–10. Command_Config Register Bit Descriptions (Part 3 of 4)

Bit(s)	Bit Name	Access *(1)*	Description
14	MHASH_SEL	RW	Select multicast address-resolution hash-code mode. ■ 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	Enables local loopback. Setting this bit to 1 enables a local loopback on the MAC's RGMII/GMII/MII interface. Frames sent through the transmit interface are looped back into the receive interface.
18 – 16	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. ■ 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	RO	Node wake up request indication. Valid only when the MAGIC_ENA bit is 1. ■ 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.
23	CNTL_FRM_ENA	RW	MAC control frame enable. ■ 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. ■ 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.

Table 4–10. Command_Config Register Bit Descriptions (Part 4 of 4)

Bit(s)	Bit Name	Access *(1)*	Description
25	ENA_10	RW	10 Mbps interface enable. ■ Setting this bit to 1 enables the 10Mbps interface, and the output signal ena_10 is asserted. ■ If this bit is set to 0, the output signal ena_10 is asserted only when the input signal set_10 is asserted.
26	RX_ERR_DISC	RW	Receive erroneous frame discard enable. ■ 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 rx_section_full register to 0. ■ If this bit is set to 0, the MAC function forwards erroneous frames to the user application with rx_err[0] asserted.
27	DISABLE_RD_TIMEOUT	RW	Setting this bit to 1 disables read timeout.
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 to Table 4–10:

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

Reg_Status Register

Figure 4–23 shows the bits and fields that comprise the reg_status register.

Figure 4–23. Reg_Status Register

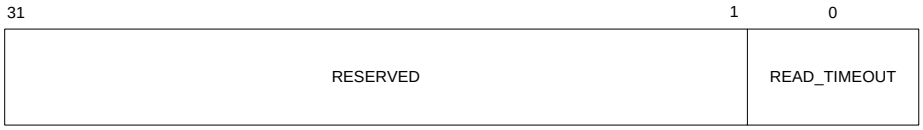


Table 4–11 provides the bit description.

Table 4–11. Reg_Status Register Bit Description

Bit	Bit Name	Access *(1)*	Description
0	READ_TIMEOUT	RC	Read access timeout indication. This bit is only valid when read timeout is not disabled (DISABLE_RD_TIMEOUT = 0). A value of 1 indicates that the read access was terminated with a timeout. A value of 0 indicates that the read access was successfully terminated. 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 rx_clk when reading the statistic counters. Bit resets to 0 when the reg_status register is read or after reset.

Note to Table 4–11:

(1) RC = Read/Clear

Tx_Cmd_Stat Register

Figure 4–24 shows the bits and fields that comprise the `tx_cmd_stat` register.

Figure 4–24. Tx_Cmd_Stat Register

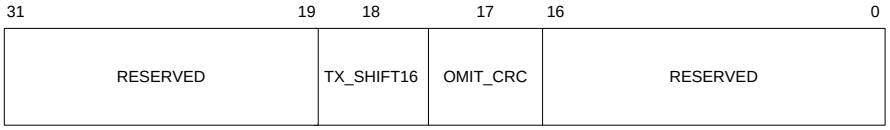


Table 4–12 provides the bit description.

Table 4–12. 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. This bit, when set to 1, always takes precedence over the signal `ff_tx_crc_fwd`.
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. This applies only if the parameter **Align packet headers to 32-bit boundary** is turned on and in MAC variations with 32-bit internal FIFOs. Otherwise, reading this bit always return a 0. In MAC variations without internal FIFOs, this bit is a read-only bit and takes the value of the parameter **Align packet headers to 32-bit boundary**.

Note to Table 4–12:

(1) RW = Read/Write

Rx_Cmd_Stat Register

Figure 4–25 shows the bits and fields that comprise the `rx_cmd_stat` register.

Figure 4–25. Rx_Cmd_Stat Register

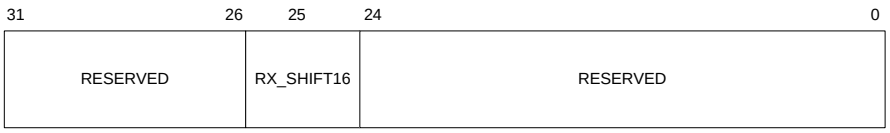


Table 4–13 provides the bit description.

Table 4–13. 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. This applies only if the parameter **Align packet headers to 32-bit boundary** is turned on and in MAC variations with 32-bit internal FIFOs. Otherwise, reading this bit always return a 0. In MAC variations without internal FIFOs, this bit is a read-only bit and takes the value of the parameter **Align packet headers to 32-bit boundary**.

Note to Table 4–13:

(1) RW = Read/Write

MAC SNMP MIB Statistics Registers

The Simple Network Management Program Management Information Base (SNMP MIB) block accumulates statistics required in IEEE Standard 802.3 basic, mandatory and recommended Management Information Packages, IEEE Standard 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 Standard 802.3 Management Packages

Table 4–14 lists the resources available to implement the IEEE Standard 802.3 mandatory management packages defined in Clause 30 for 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.

Table 4–14. IEEE Standard 802.3 MAC oEntity and oPauseEntity Managed Object Support (Part 1 of 2)

IEEE Standard 802.3 Attribute	IEEE Management Packages			Description
	Basic	Mandatory	Recommended (optional)	
oEntity Managed Object Support				
aMACID	√			The MAC addresses.
aFramesTransmittedOK		√		Number of frames transmitted without error including pause frames.
aFramesReceivedOK		√		Number of frames received without error including pause frames.
aFrameCheckSequence Errors		√		Number of frames received with a CRC error.
aAlignmentErrors		√		Frame received with an alignment error.
aOctetsTransmittedOK			√	Sum of payload and padding octets of frames transmitted without error.

Table 4–14. IEEE Standard 802.3 MAC oEntity and oPauseEntity Managed Object Support (Part 2 of 2)

IEEE Standard 802.3 Attribute	IEEE Management Packages			Description
	Basic	Mandatory	Recommended (optional)	
aOctetsReceivedOK			√	Sum of payload and padding octets of frames received without error.
oPauseEntity Managed Object Support				
aPAUSEMACCtrlFrames Transmitted			√	Number of transmitted pause frames.
aPAUSEMACCtrlFrames Received			√	Number of received pause frames.

For more information on Attributes and objects, refer to IEEE Standard 802.3 Clause 30.

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 4–15. 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: ■ FIFO overflow error ■ CRC error ■ Length error ■ Alignment error
ifOutErrors	Number of frames transmitted with error: ■ 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 4–16 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 bytes 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 bytes or longer than and the value set in the `frm_length` register.

Table 4–16. IEFT RMON MIB Object Support

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

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 4–17 describes three derived IETF MIB (MIB-II) objects

Table 4–17. Derived IETF MIB (MIB-II) Objects

MIB Object	Description
`ifInOctets`	Sum of bytes received except preamble (for example, header, payload, pad and FCS) of all valid received frames. = 18 × `aFramesReceivedOK` + `aOctetsReceivedOK`
`ifOutOctets`	Sum of bytes transmitted except preamble (for example, header, payload, pad and FCS) of all valid transmitted frames. = 18 × `aFramesTransmittedOK` + `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 4–18 describes three derived IETF RMON MIB objects.

Table 4–18. Derived IETF RMON MIB Objects

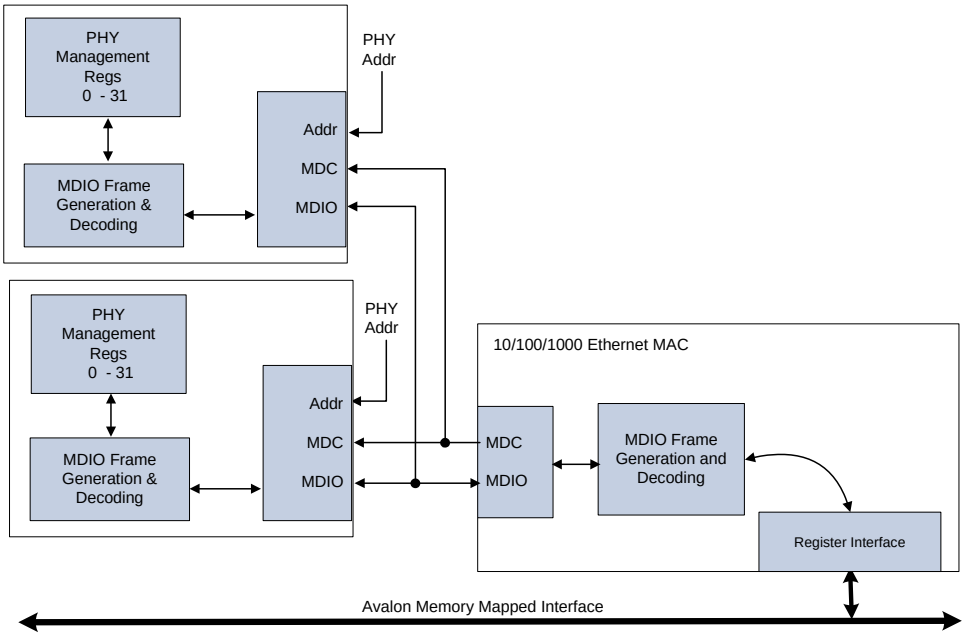
Object	MAC Core Support
`etherStatsBroadcastPkts`	Any valid frame with Broadcast address: = ifInBroadcastPkts
`etherStatsMulticastPkts`	Any valid multicast frame, including pause frames: = ifInMulticastPkts + aPAUSEMACCtrlFramesReceived
`etherStatsCRCAlignErrors`	Incremented when frames of correct length but with CRC error are received: = aFrameCheckSequenceErrors

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 that supports up to 32 PHY devices. To access each PHY device, the PHY address must be written to the register space followed by the transaction data.

For faster access, the MAC function allows up to two PHY devices to be mapped in its register space at any one time. Subsequent transactions to the same PHYs do not require writing the PHY addresses to the register space thus reducing the transaction overhead. The MDIO register space can be accessed via the MAC function's Avalon-MM interface.

Figure 4–26. MDIO Interface



MDIO Frame Format

The MAC MDIO master controller communicates with the slave PHY device using frames, which are defined in Table 4–19. 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 (IEEE Standard 803.2 Clause 22).

Table 4–19. MDIO Frame Formats (Read/Write)

		Command						
Type	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	xxxxxxxxxxxxxxxx	Z
Write	1 ... 1	01	01	xxxxx	xxxxx	10	xxxxxxxxxxxxxxxx	Z

Table 4–20 describes the fields of the MDIO frame.

Table 4–20. 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: ■ 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 `mdio_addr0` register. For PHY device 1, the Addr1 field is set to the value configured in the `mdio_addr1` 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 2nd 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 4–21 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 4–21. 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

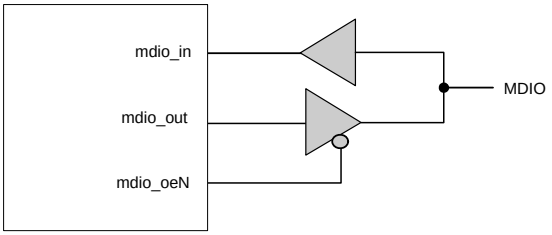
The Management Data Clock (MDC) is generated from the Avalon-MM interface clock signal, `clk`. The division factor is defined by specifying the value in the **Host clock divisor** parameter. For more information about the parameters, refer to "MAC Options" on page 3–3.

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

MDIO Buffer Connection

Figure 4–27 illustrates the buffers used for the MDIO tri-state bus.

Figure 4–27. MDIO Buffer Connection



Media Independent Interfaces

The following media independent interfaces are implemented:

- Fast Ethernet media independent interface (MII)
- Gigabit media independent interface (GMII)
- Reduced gigabit media independent interface (RGMII)

GMII Interface

To use this interface, you must set the **Interface** option on the **Core Configuration** page to **MII/GMII** for 10/100/1000 Mbps Ethernet MAC core variations or **GMII** for Small MAC core variations. User applications can activate GMII by setting the `ETH_SPEED` bit in the `command_config` register to 1. See "Core Configuration" on page 3–1 for more information on the parameter settings.

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 deasserted.

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 frame transfer.

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 the `gm_rx_d[7:0]` bus. Between frames, `gm_rx_dv` remains deasserted.

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 frame transfer.

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

To use this interface, you must set the **Interface** option on the **Core Configuration** page to **RGMII**. User applications can activate RGMII by setting the `ETH_SPEED` bit in the `command_config` register to 1. See "Core Configuration" on page 3–1 for more information on the parameter settings.

Transmit

On transmit, all data transfers are synchronous to 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 deasserted. Figure 4–28 and Figure 4–29 show the timing diagrams of RGMII transmit in 10/100 Mbps and gigabit mode respectively.

Figure 4–28. RGMII Transmit in 10/100 Mbps

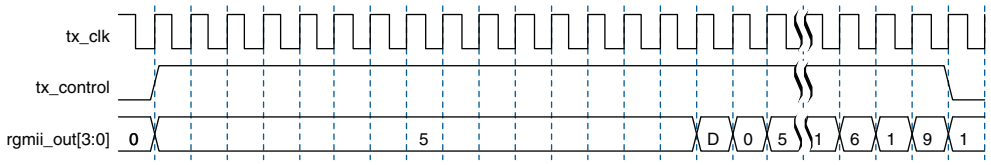
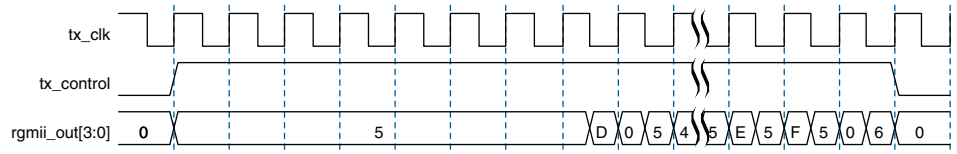
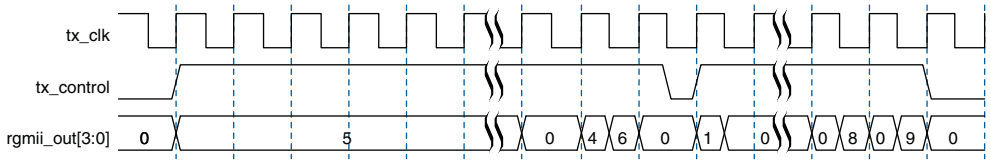


Figure 4–29. RGMII Transmit in Gigabit Mode



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 frame transfer. Figure 4–30 shows the timing diagram of RGMII transmit when an error occurs.

Figure 4–30. RGMII Transmit with Error in 1000 Mbps



Receive

On receive all signals are sampled on 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 deasserted. Figure 4–31 and Figure 4–32 show the timing diagrams of RGMII receive in 10/100 Mbps and 1000 Mbps respectively.

Figure 4–31. RGMII Receive in 10/100 Mbps

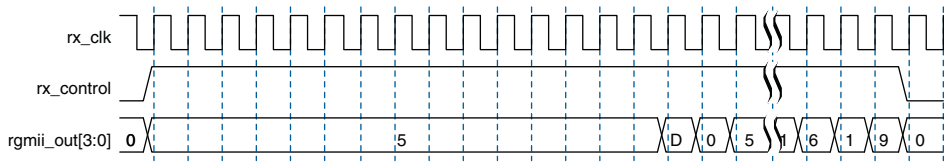
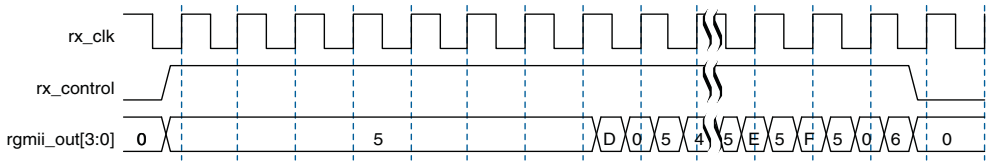
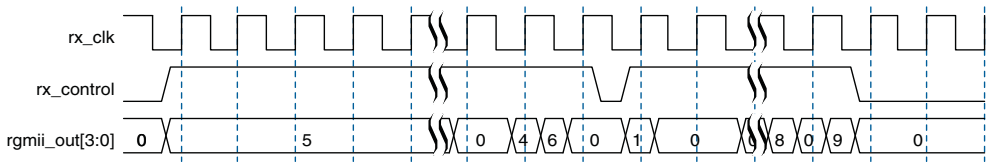


Figure 4–32. RGMII Receive in 1000 Mbps



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. Figure 4–33 shows the timing diagram of RGMII receive when an error occurs.

Figure 4–33. RGMII Receive with Error in Gigabit Mode



The current implementation of the RGMII receive interface expects a positive-delay `rx_clk` relative to the receive data (the clock comes after the data).

MII Interface

To use this interface, you must set the **Interface** option on the **Core Configuration** page to either **MII/GMII** for 10/100/1000 Mbps Ethernet MAC core variations or **MII** for Small MAC core variations. User applications can activate MII by setting the `ETH_SPEED` bit in the `command_config` register to 0. See "Core Configuration" on page 3–1 for more information on the parameter settings.

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 deasserted.

If a frame is received on the FIFO interface with an error (`ff_tx_err` asserted) the frame is subsequently transmitted with the MII error signal `m_tx_err` for one clock cycle at any time during the frame transfer.

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.

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 frame transfer.

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 PHYs

The MAC function implements a flexible network interface—MII for 10/100 Mb interfaces, RGMII or GMII for gigabit interfaces—that can be used in multiple applications. This section provides implementation guidelines for the following typical network applications:

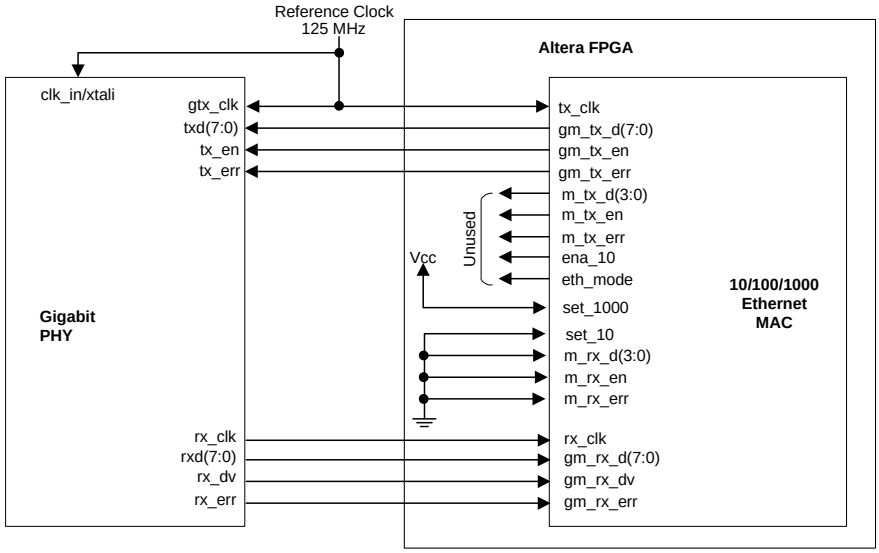
- Gigabit Ethernet operation
- Programmable 10/100 Ethernet operation
- Programmable 10/100/1000 Ethernet operation

Gigabit Ethernet

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

A technology specific clock driver is required to generate a clock centered with the GMII or RGMII data from the MAC. The clock driver can be a PLL, a delay line or a DDR flip-flop. Figure 4–34 shows how gigabit Ethernet PHYs are connected to the MAC via GMII.

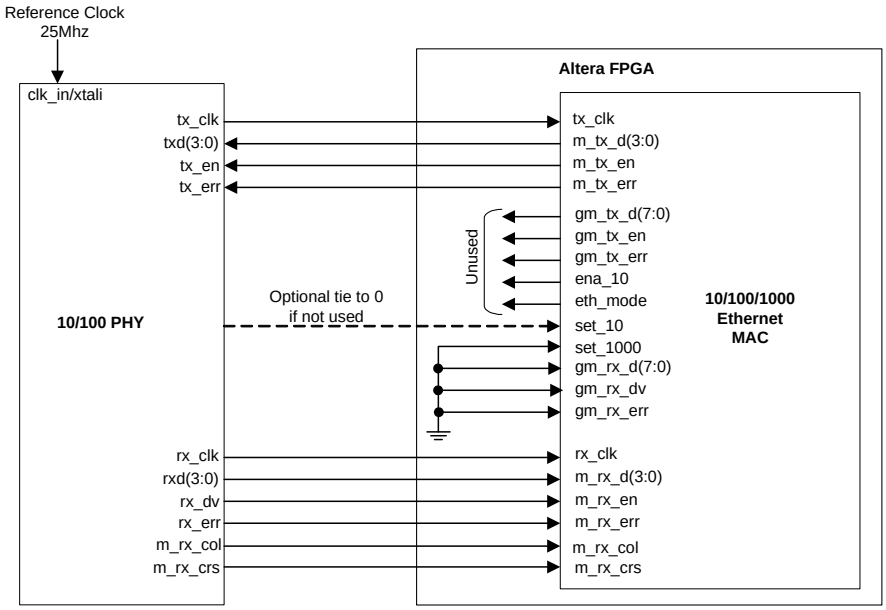
Figure 4–34. Gigabit PHY to MAC via GMII



Programmable 10/100 Ethernet

10/100 Ethernet PHYs are connected to the MAC function via MII. On the receive path, the PHY device provides a 25 MHz (100 Mbps Ethernet) or a 2.5 MHz (10 Mbps Ethernet) clock which is connected to the MAC 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 is connected to the MAC clock, `tx_clk`. Figure 4–35 shows the required connection for programmable 10/100 Ethernet operation.

Figure 4–35. 10/100 PHY Interface



Programmable 10/100/1000 Ethernet Operation

Typically, 10/100/1000 Ethernet PHY devices implement a shared interface that can be configured to connect to the MAC via MII for 10/100 Mbps operation or GMII/RGMII for 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 clock `rx_clk`. The PHY interface is connected to both the MAC MII (active PHY signals) and GMII.

On the transmit path, standard programmable PHY devices operating in 10/100 mode generate a 2.5 MHz (10 Mbps) or a 25 MHz (100 Mbps) clock. When operating in gigabit mode, the PHY devices expect a 125 MHz clock from the MAC. Because the MAC function does not generate a clock output, an external clock module is introduced to drive the 125 MHz clock to the MAC function and PHY devices. In 10/100 mode, the clock generated by the MAC to the PHY can be tri-stated.

During transmission, either MII or GMII is selected by the MAC control signal `eth_mode`. The signal `eth_mode` is set to 1 when the MAC is configured to operate in gigabit mode. This drives the MAC GMII to the PHY interface. The signal `eth_mode` is set to 0 when the MAC function is configured to operate in 10/100 mode. In this mode, the MAC MII is driven to the PHY interface.

Figure 4–36 shows the required connection for programmable 10/100/1000 Ethernet operation via MII/GMII.

Figure 4–36. 10/100/1000 PHY Interface via MII/GMII

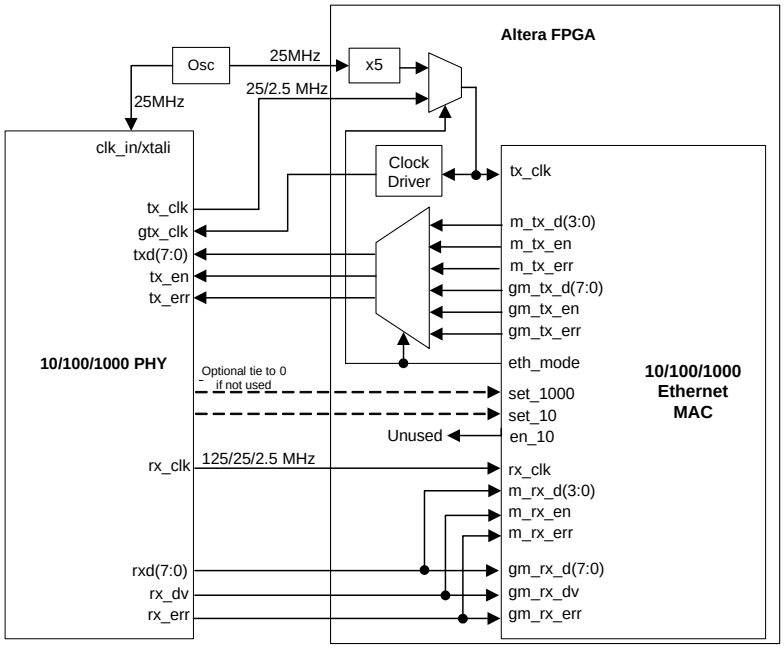
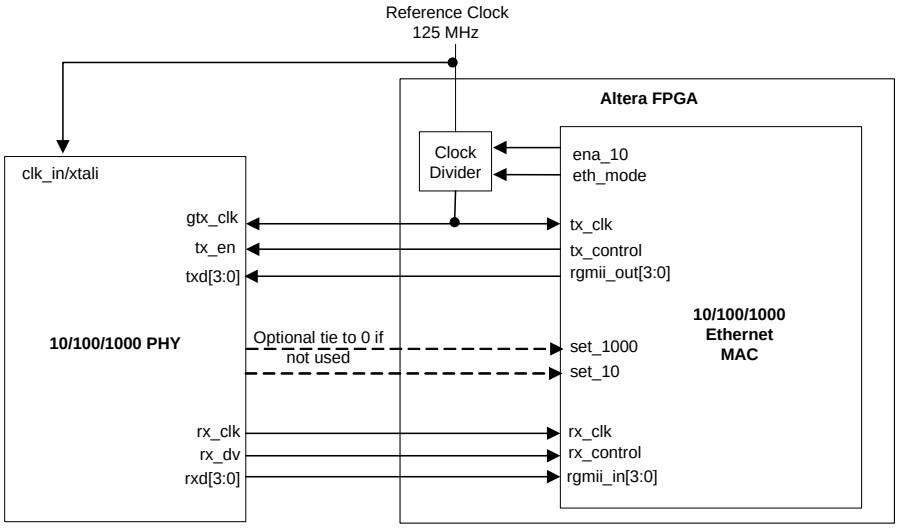


Figure 4–37 shows the required connection for programmable 10/100/1000 Ethernet operation via RGMII.

Figure 4–37. 10/100/1000 PHY Interface via RGMII



1000BASE-X/SGMII PCS with Optional PMA

The 1000BASE-X/SGMII PCS function is accessible via GMII (1000BASE-X/SGMII) or MII (SGMII). 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 an embedded Physical Medium Attachment (PMA). This configuration complies with the IEEE 802.3 Standard 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.

Configurations with PCS and embedded PMA can only support frame lengths longer than 16 Kbytes if the difference between the input reference clock and recovered clock is zero ppm.

Figure 4–38 shows a block diagram of the PCS function.

Figure 4–38. 1000BASE-X/SGMII PCS

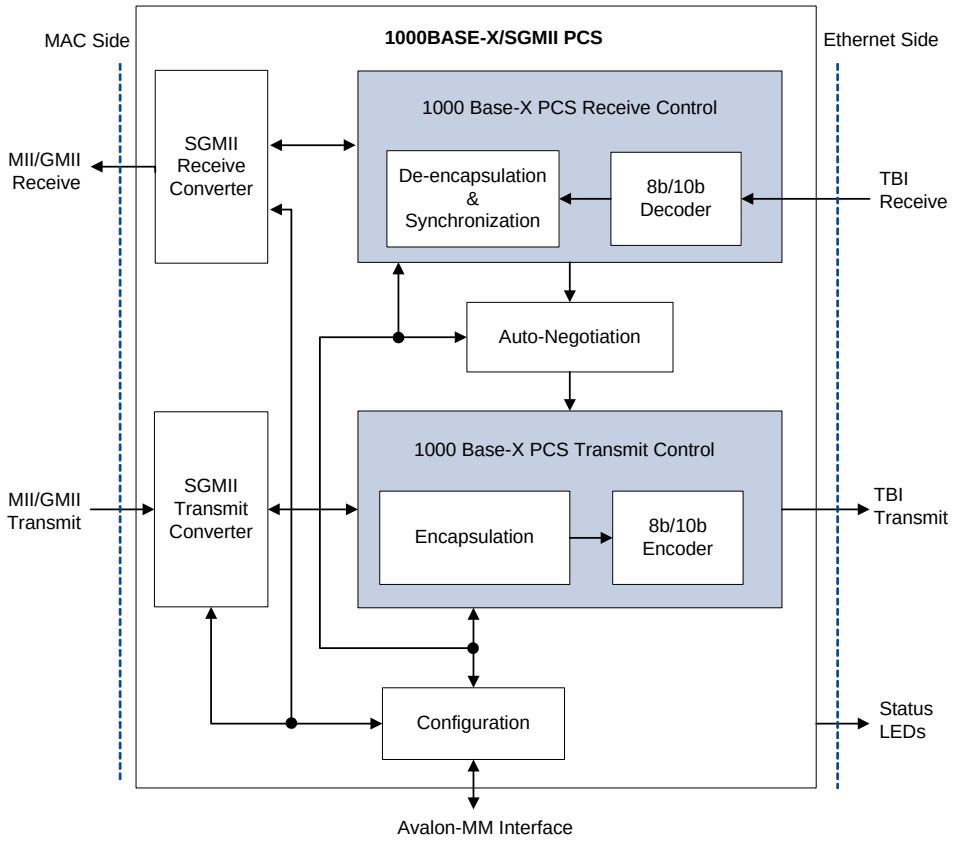
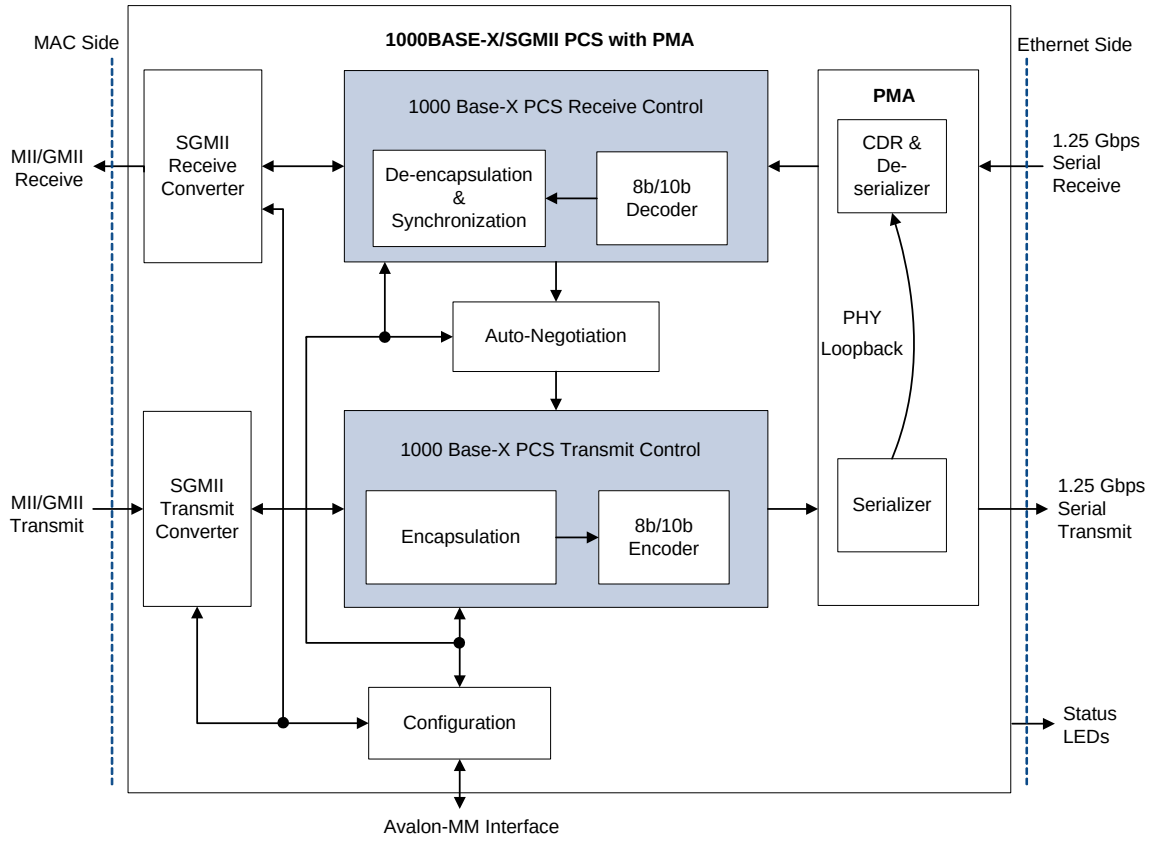


Figure 4–39 shows a block diagram of the PCS function with an embedded PMA.

Figure 4–39. 1000BASE-X/SGMII PCS with PMA



For designs that contain multiple PMA blocks targeting Altera device families with GX transceivers, you can combine the transceiver channels in the same quad. To successfully do so, the transceiver channels must have the same dynamic reconfiguration setting. If some of the channels do not use the dynamic reconfiguration capability, you need to instantiate an additional reconfiguration block and set the signals `reconfig_clk` to `0` and `reconfig_togxb` to `3'b010` for these channels.

PCS Receive

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

☞ The receive latency of the PCS function is 11 clock cyles.

Comma Detection

Ten-bit data received from PMA devices may not align on a valid 10-bit character. The comma detection function searches for the 10-bit encoded comma character, K28.1/K28.5/K28.7, in consecutive samples received from PMA devices. When the K28.1/K28.5/K28.7 comma code group is detected, the stream is realigned on a valid 10-bit character boundary. The aligned stream can subsequently be decoded with a standard 8b/10b decoder.

The comma detection function 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 preamble byte (0x55). 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 frame 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 code groups with comma 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 error counter for each invalid character received and incorrectly positioned comma character. The internal error counter is decremented when four consecutive valid characters are received. When the counter reaches 4, 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.

☞ The transmit latency of the PCS function is 6 clock cycles.

Frame Encapsulation

During transmission, the PCS function encapsulates frames according to the specification in IEEE Standard 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 frame /T/ /R/ /R/ or the /T/ /R/ sequence is inserted. The selection of the end frame 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 transmission), 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 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.

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.

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

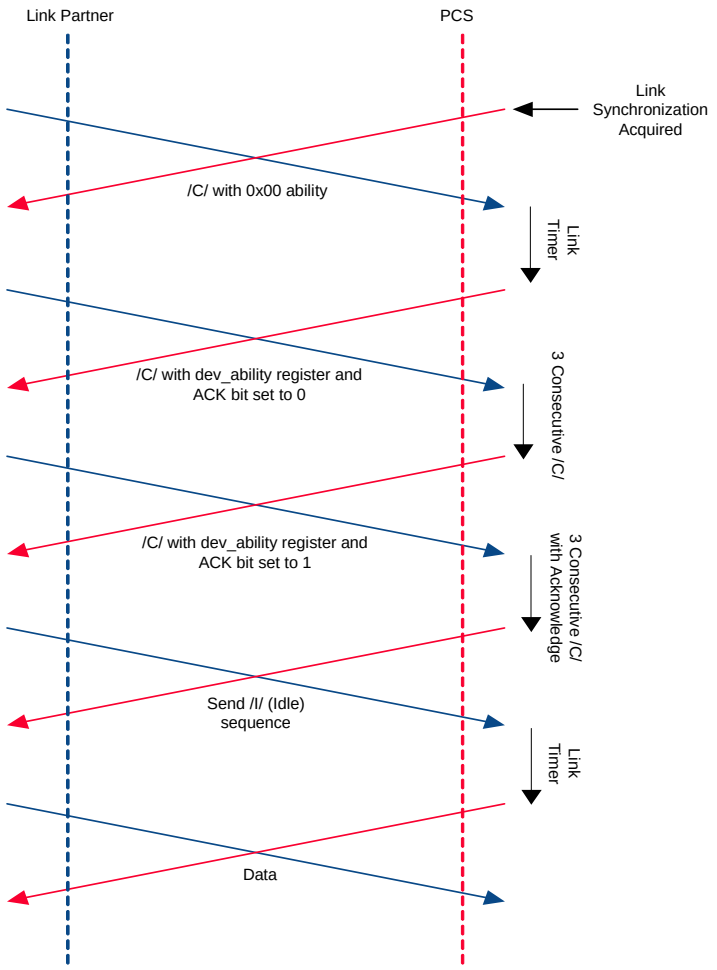
When link synchronization is acquired, the PCS function starts sending a /C/ sequence (configuration sequence) to the link partner device with the advertised register set to 0x00. The sequence is sent for a time specified in the PCS `link_timer` register mapped in the PCS register space. See "PCS Control Register" on page 4–63 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 4–40.

Figure 4–40. 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 4–25 on page 4–64) and the `AUTO_NEGOTIATION_COMPLETE` bit in the `status` register (Table 4–24 on page 4–63) 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 4–23 on page 4–63) 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.

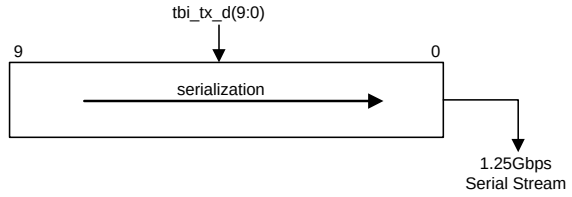
If the `SGMII_ENA` bit is 1 and the `USE_SGMII_AN` bit is 0, the PCS function can be configured with the `SGMII_SPEED` and `SGMII_DUPLEX` bits in the `if_mode` register.

Ten-bit Interface

The PCS function implements a TBI to an external SERDES when the PCS function is implemented without an embedded PMA.

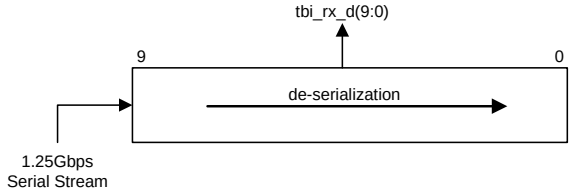
On transmit, the SERDES must serialize `tbi_tx_d[0]`, the least significant bit of the TBI output bus first and `tbi_tx_d[9]`, the most significant bit of the TBI output bus last to ensure the remote node receives the data correctly, as Figure 4–41 illustrates.

Figure 4–41. SERDES Serialization Overview



On receive, the SERDES must serialize the TBI least significant bit first and the TBI most significant bit last, as Figure 4–42 illustrates.

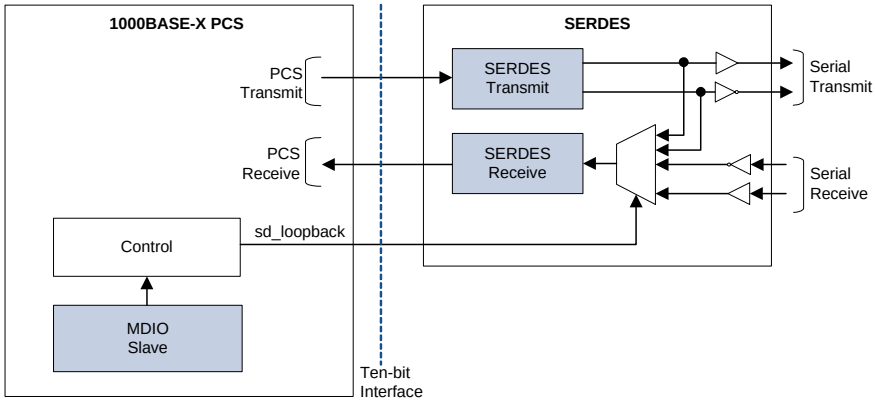
Figure 4–42. SERDES De-Serialization Overview



PHY Loopback

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

Figure 4–43. Serial Loopback

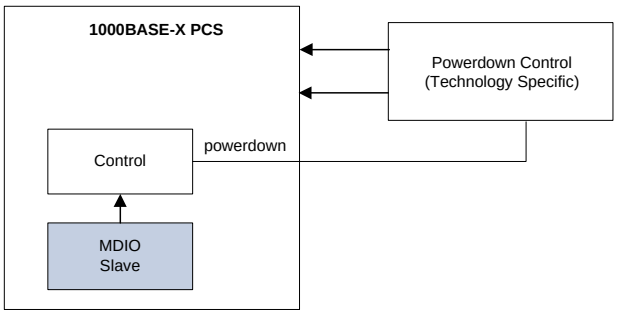


PHY Power-Down

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 4–44. Power-Down



Power-Down with Embedded PMA

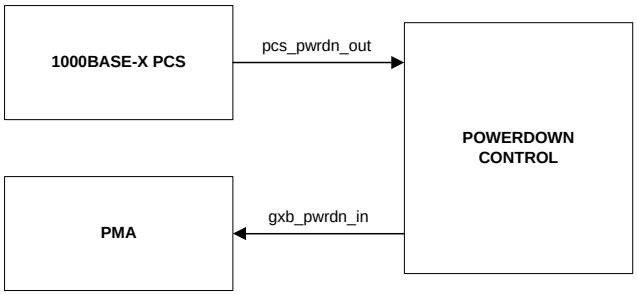
This section applies only to target devices with GX transceivers. When the PCS function is implemented with an embedded PMA, the power-down signal is internally connected to the power-down of the GX transceiver.

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 about the parameter, refer to "PCS/SGMII Options" on page 3–6.

Figure 4–45 illustrates the power-down control with exported power-down signal.

Figure 4–45. Power-Down with Export Transceiver Power-Down Signal



PCS Control Interface

The PCS control interface is an Avalon-MM slave port that provides access to PCS registers. User applications can change the behavior of the PCS function by writing to these registers.

Table 4–22 lists and describes the PCS registers. In configurations that contain both the MAC and PCS functions, see Table 4–8 on page 4–27 for the location of the PCS control interface register space.

The table shows the address offset in PCS and PCS/PMA configurations. For configurations that contain both the MAC and PCS functions, multiply the address offset by 2.

For more information about Avalon Memory-Mapped interface protocol, refer to the *Avalon Interface Specifications*.

Table 4–22. 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 4–23 on page 4–63.	RW
0x02	`status`	Status register. Provides information on the operation of the PCS function.	RO
0x04	`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
0x06			
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 4–25 on page 4–64.	RW

Table 4–22. PCS Interface Register Map (Part 2 of 2)

Address Offset	Register Name	Description	Access *(1)*
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 4–25 on page 4–64 and Table 4–26 on page 4–65, respectively.	RO
0x0C	an_expansion	Auto-negotiation expansion register. Contains the PCS function capability and auto-negotiation status.	RO
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	The PCS function revision. Always set to the current version of the MegaCore function.	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 offest 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 to Table 4–22:

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

PCS Control Register

Table 4–23 describes the function of each bit and field in the PCS `control` register.

Table 4–23. 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	Indicates the operating mode of the PCS function. Bits 6 and 13 are fixed to 1 and 0 respectively. This combination of values represent gigabit mode.	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 to Table 4–23:

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

Status Register

Table 4–24 describes the function of each bit and field in the `status` register. All bits in this register are read-only bits.

Table 4–24. Status Register Bit Descriptions (Part 1 of 2)

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.
3	AUTO_NEGOTIATION_ABILITY	A value of 1 indicates that the PCS function supports auto-negotiation.

Table 4–24. Status Register Bit Descriptions (Part 2 of 2)

Bit Number	Name	Description
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: ■ 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 used 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. The contents of these registers are valid only when the auto-negotiation completes (AUTO_NEGOTIATION_COMPLETE bit in the status register = 1).

1000BASE-X

Table 4–25 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 4–25. Dev_Ability and Partner_Ability Registers Bits Description in 1000BASE-X (Part 1 of 2)

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.

Table 4–25. Dev_Ability and Partner_Ability Registers Bits Description in 1000BASE-X (Part 2 of 2)

Bit(s)	Name	Description
7	PS1	Pause support. ■ 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: ■ 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 dev_ability register, this bit is always set to 0.

Notes to Table 4–25:

(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 4–26 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 4–26. 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: ■ 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: ■ 1: copper interface is capable of operating in full-duplex mode ■ 0: copper interface is capable of operating in half-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: ■ 1: copper interface link is up ■ 0: copper interface link is down

An_Expansion Register

Table 4–27 describes the function of each bit and field in the an_expansion register.

Table 4–27. 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 partner_ability. 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 4–28 describes the function of each bit and field in the if_mode register.

Table 4–28. 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 = 1) and programmed not to be automatically configured (USE_SGMII_AN = 0), set the speed as follows: ■ 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 half-duplex mode. Setting this bit to 1 enables half duplex for 10/100 Mbps speed. This bit is ignored when SGMII_ENA is 0 or USE_SGMII_AN is 1.	RW
5 to 15	Reserved	—	RO

Note to Table 4–28:

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

Clock Distribution

The total number of global and regional clock resources required by your system depends on the following factors:

- Configuration of the Triple Speed Ethernet MegaCore function; the blocks it contains
- PCS operating mode (SGMII or 1000BASE-X)
- PMA technology implemented in the target device
- Number of clocks that can share a single source
- Number of PMAs required in the design
- ALTGX megafunction operating mode

You can use the same clock source to drive clocks that are visible at the top-level design, thus reducing the total number of clock sources required by the entire design. Table 4–29 lists the clock and reset signals that are visible at the top-level design for each possible configuration.

Table 4–29. Clock and Reset Signals Visible at Top-Level Design

Clocks	Configurations *(1)*		
	MAC Only	**MAC and PCS**	**MAC and PCS with PMA**
`ref_clk`	Yes	—	Yes
`clk`	Yes	Yes	Yes
`reset`	Yes	Yes	Yes
`pma_digital_rst2` *(2)*	—	—	No
`ff_tx_clk`	Yes	Yes	Yes
`ff_rx_clk`	Yes	Yes	Yes
`tx_clk`	Yes	No	No
`rx_clk`	Yes	No	No
`tbi_rx_clk`	—	No	No
`tbi_tx_clk`	—	No	No
`gxb_cal_blk_clk` *(3)*	—	—	Yes

Notes to Table 4–29:

(1) Yes indicates that the clock is visible at the top-level design.
No indicates that the clock is not visible at the top-level design.
— indicates that the clock is not applicable for the given configuration.

(2) Applies to LVDS Soft-CDR I/O.

(3) Applies to GX transceiver.

MAC and PCS with PMA in Devices With GX Transceivers

In configurations that contain the MAC, PCS, and PMA blocks in devices with GX transceivers, you have the following options in optimizing clock resources:

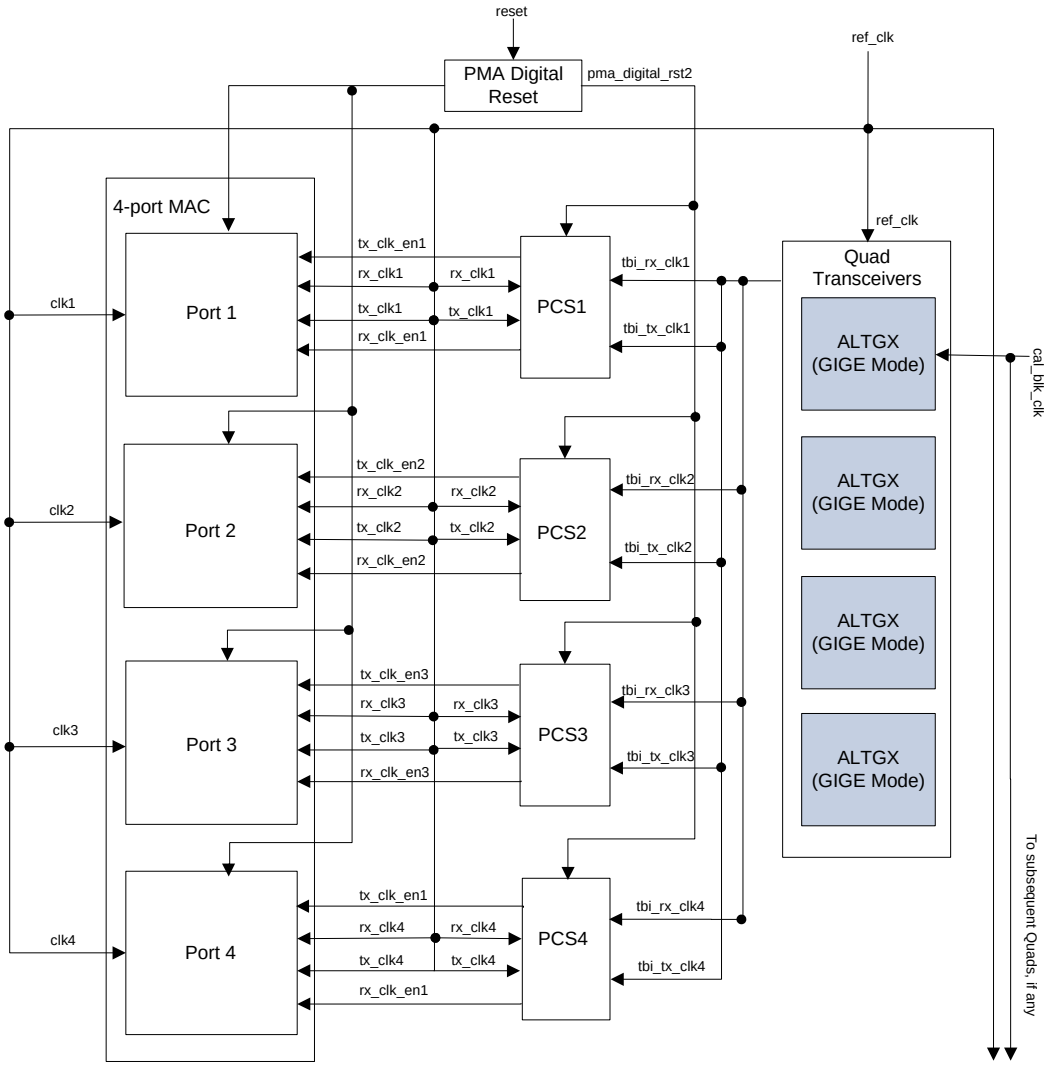
- Utilize the same reset signal for all MAC instances if you do not require a separate reset for each instance.
- Utilize the same reference clock for all PMA quads
- Utilize the same clock source to drive the reference clock, FIFO transmit and receive clocks, and system clocks, if these clocks run at the same frequency.

The Quartus II software automatically optimizes the TBI transmit clocks. Only one clock source drives the TBI transmit clocks from each PMA quad.

The calibration clock (`gxb_cal_blk_clk`) calibrates the termination resistors in all transceiver channels in a device. As there is only one calibration circuit in each device, one clock source suffices.

Figure 4–46 on page 4–69 shows the optimal clock distribution scheme you can achieve in configurations that contain the 10/100/1000 Ethernet MAC, SGMII PCS, and PMA blocks in devices with GX transceivers.

Figure 4–46. Clock Distribution in MAC and SGMII PCS with GXB Configuration—Optimal Case



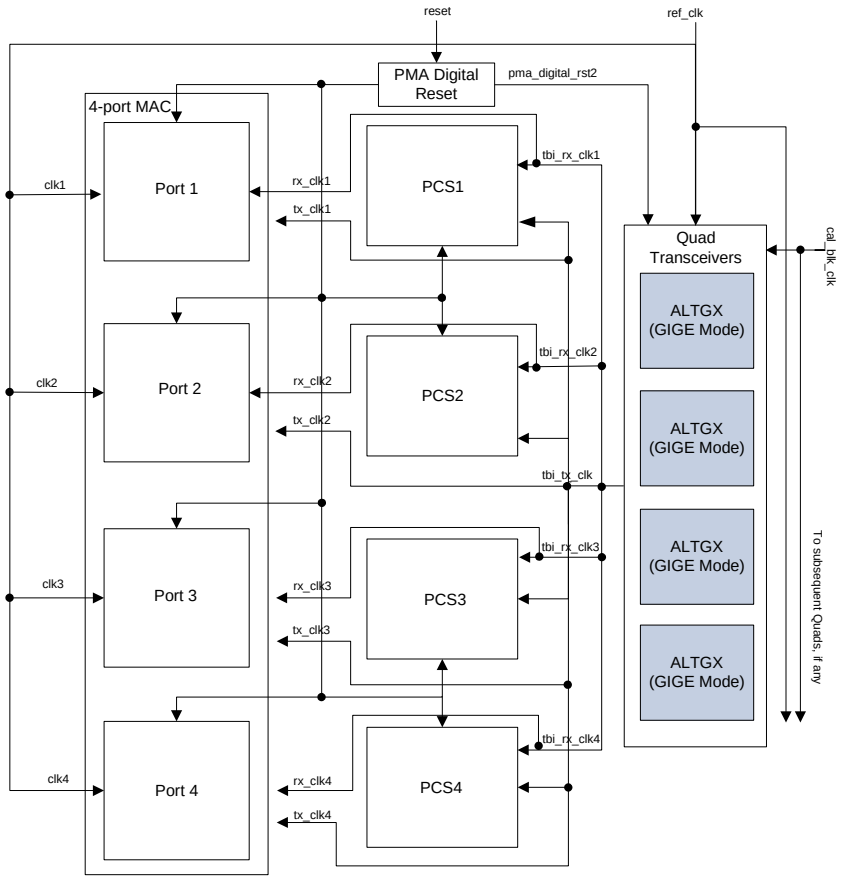
Notes to Figure 4–46:

(1) The PMA layer in devices with GX transceivers uses ALTGX megafunctions.

Figure 4–47 on page 4–70 shows the optimal clock distribution scheme you can achieve in configurations that contain the 10/100/1000 Ethernet MAC, 1000Base-X PCS, and PMA blocks in devices with GX transceivers.

In addition to the aforementioned optimization options, the TBI transmit and receive clocks can be used to drive the MAC transmit and receive clocks, respectively.

Figure 4–47. Clock Distribution in MAC and 1000BASE-X PCS with GXB Configuration—Optimal Case



Note to Figure 4–47:

(1) The PMA layer in devices with GX transceivers uses ALTGX megafunctions.

MAC and PCS with PMA in Devices with LVDS Soft-CDR I/O

In configurations that contain the MAC, PCS, and PMA blocks in devices with LVDS Soft-CDR I/O, you have the following options in optimizing clock resources:

- Utilize the same reset signal for all MAC instances if you don't require a separate reset for each instance.
- Utilize the same clock source to drive the reference clock, FIFO transmit and receive clocks, and system clocks, if these clocks run at the same frequency.

Figure 4–48 on page 4–71 shows the optimal clock distribution scheme you can achieve in configurations that contain the MAC, SGMII PCS and PMA blocks in devices with LVDS Soft-CDR I/O.

Figure 4–48. Clock Distribution in MAC and SGMII PCS with LVDS Configuration—Optimal Case

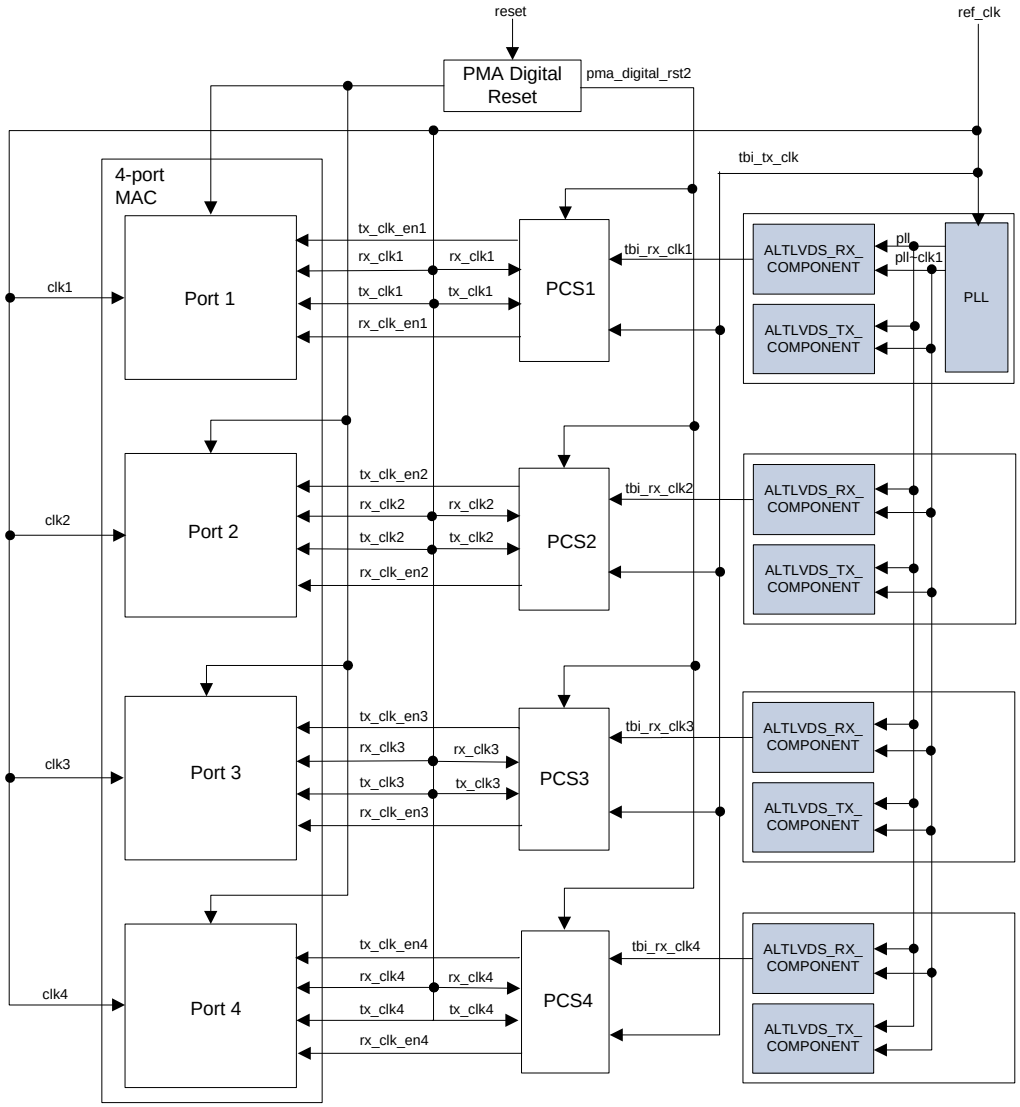
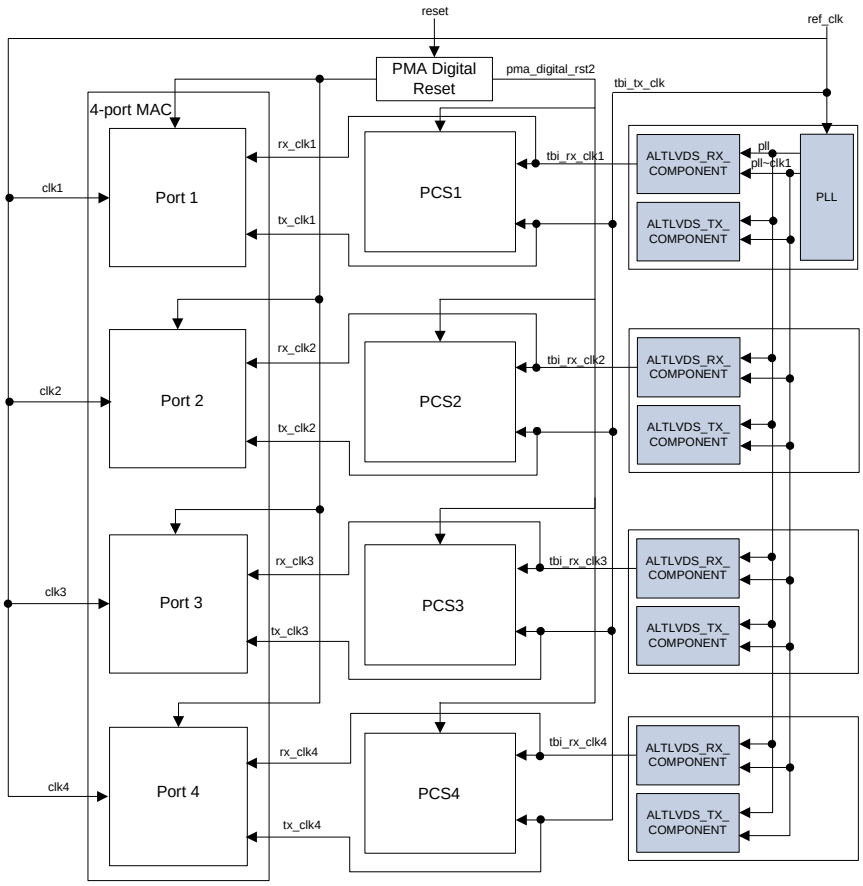


Figure 4–49 shows the optimal clock distribution scheme you can achieve in configurations that contain the MAC, 1000BASE-X PCS and PMA blocks in devices with LVDS.

Figure 4–49. Clock Distribution in MAC and 1000BASE-X PCS with LVDS Configuration—Optimal Case



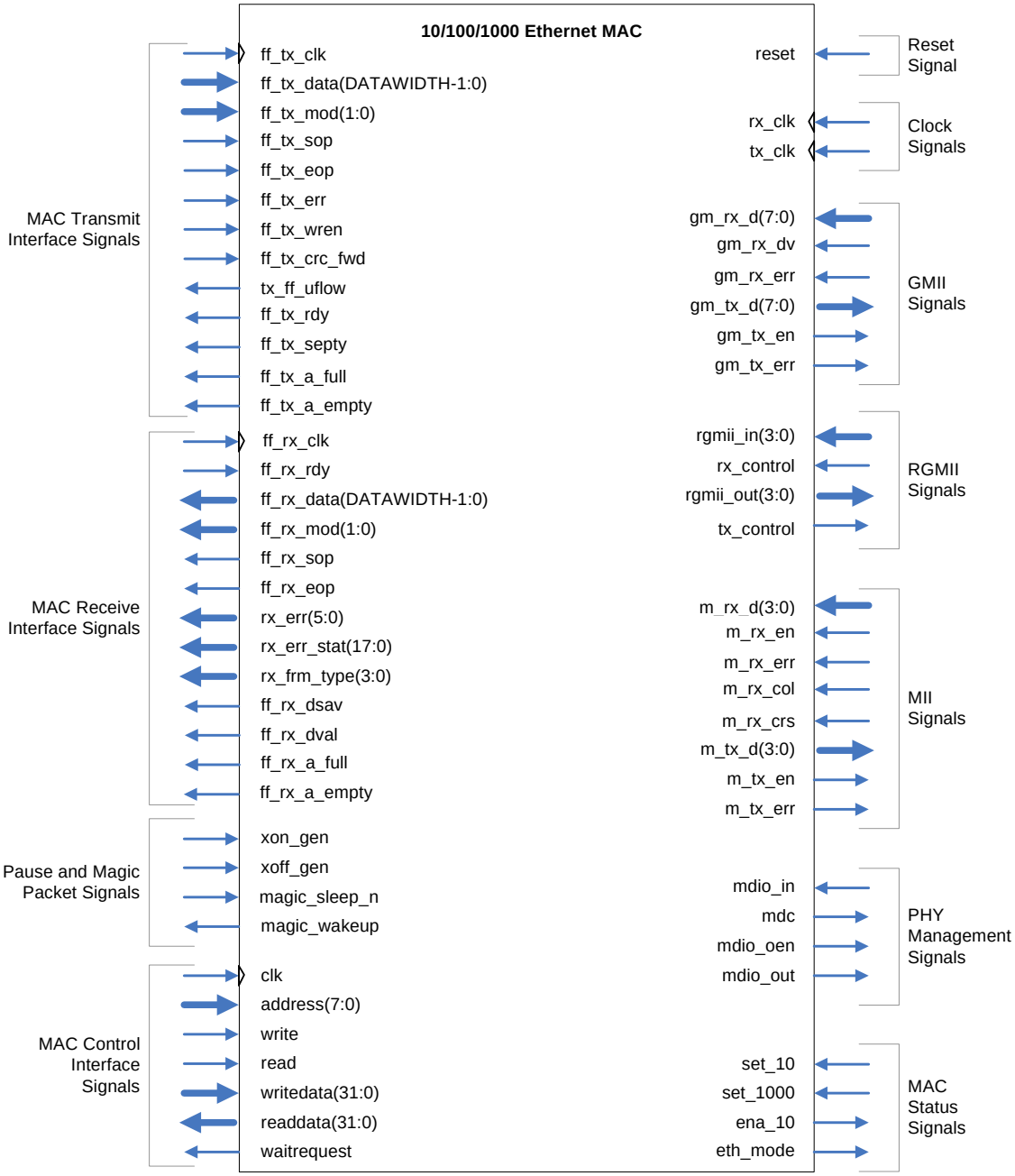
Signals

This section describes all interface signals of each possible configuration.

10/100/1000 Ethernet MAC Signals

Figure 4–50 shows all I/O signals of the 10/100/1000 Ethernet MAC function with internal FIFOs.

Figure 4–50. 10/100/1000 Ethernet MAC Signals



Clock and Reset Signal

Data transfers on the MAC Ethernet interface are synchronous to the receive and transmit clocks. Table 4–30 describes the use of these clock signals.

Table 4–30. GMII/RGMII/MII Clock Signals

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

Table 4–31 describes the reset signal.

Table 4–31. Reset Signal

Signal Name	Direction	Description
reset	In	Single reset for all logic in the MAC and PCS Register Interface.

MAC Control Interface Signals

Table 4–32 describes the signals that comprise the MAC control interface.

Table 4–32. Avalon-MM 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(7:0)	address	In	32-bit word-aligned 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 during register read or register write access. Set to 0 when the current register access completes.

MAC Status Signals

The MAC status signals allow you to set the transfer mode of the interface. Table 4–33 lists all MAC status signals.

Table 4–33. 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 4–33. 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 ETH_SPEED 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 ETH_SPEED is set to 1 or when the register bit ENA_10 is set to 1. The ENA_10 software configuration bit has a higher priority than this signal.

MAC Receive Interface Signals

The MAC receive interface is an Avalon-ST source port that provides an interface to the MAC receive function. Table 4–34 describes all interface signals associated with the MAC receive interface.

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

Signal Name	Avalon-ST Signal Type	Direction	Description
Avalon-ST Signals			
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. The MAC function asserts this signal 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[17:0] 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] and so forth.
ff_rx_mod(1:0)	empty	Out	Receive data modulo. Indicates which part of the final Frame word is not valid: ■ 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	startofpacket	Out	Receive start of packet. Set to 1 when the first byte 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 byte or word of frame data is driven on ff_rx_data[(DATAWIDTH-1):0].

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

Signal Name	Avalon-ST Signal Type	Direction	Description
ff_rx_rdy	ready	In	Receive application ready. Asserted by the receive application to indicate it is ready to receive data from the MAC. The signal ff_rx_rdy must be generated on the rising edge of ff_rx_clk.
rx_err(5:0)	error	Out	Receive error. Asserted with the final byte in the frame to indicate that an error was detected when receiving the frame. See Table 4–36 on page 4–77 for the bit description.
Component-Specific Signals			
ff_rx_dsav	—	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. This signal remains deasserted in the store and forward mode.
rx_frm_type(3:0)	—	Out	Frame type. See Table 4–35 on page 4–76 for the bit description.
ff_rx_a_full	—	Out	Receive FIFO almost-full threshold. This signal is asserted when the FIFO reaches the almost full threshold.
ff_rx_a_empty	—	Out	Receive FIFO almost-empty threshold. This signal is asserted when the FIFO goes below the almost empty threshold.
rx_err_stat(17:0)	—	Out	rx_err_stat[17]: One indicates that the current frame implements stacked VLAN rx_err_stat[16]: One indicates that the current frame implements either VLAN or stacked VLAN rx_err_stat[15:0]: Received frame payload length/type in bytes as found in length/type field

Table 4–35 describes each bit of the signal rx_frm_type.

Table 4–35. Rx_frm_type Bit Description

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

Table 4–36 describes each bit of the signal rx_err.

Table 4–36. Rx_err Bit Description

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 Standard 802.3.
0	Receive frame error. This signal indicates that an error has occurred. It is the logical OR of receive errors 1 through 5.

MAC Transmit Interface Signals

The MAC transmit interface is an Avalon-ST sink port that provides an interface to the MAC transmit function. Table 4–37 describes all signals associated with the MAC transmit interface.

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

Signal Name	Avalon-ST Signal Type	Direction	Description
Avalon-ST Signals			
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: ■ 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 first byte in the frame (the first byte of the destination address) is driven on ff_tx_data.

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

Signal Name	Avalon-ST Signal Type	Direction	Description
ff_tx_eop	endofpacket	In	Transmit end of packet. Set to 1 when the last byte in the frame (the last byte of the FCS field) is driven on ff_tx_data.
ff_tx_err	error	In	Transmit frame error. Asserted with the final byte in the frame to indicate that the transmitted frame is invalid. When ff_tx_err is asserted, the frame is transmitted to the GMII interface with an error.
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.
Component-Specific Signals			
ff_tx_crc_fwd	—	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.
tx_ff_uflow	—	Out	Asserted when a FIFO underflow occurs on the transmit FIFO.
ff_tx_septy	—	Out	FIFO data section empty. Deasserted, when the FIFO is filled to or above the threshold defined in tx_section_empty register. User applications can use this signal to stop FIFO write and initiate backpressure measures.
ff_tx_a_full	—	Out	Asserted when the transmit FIFO reaches the almost full threshold.
ff_tx_a_empty	—	Out	Asserted when the transmit FIFO goes below the almost empty threshold.

Pause and Magic Packet Signals

The pause and magic packet signals are component-specific signals. Table 4–38 describes these signals.

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

Signal Name	Direction	Description
xon_gen	In	XON Generate. When this signal is set to 1 for at least 1 tx_clk 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 xon_gen bit in the command_config register is set to 1. This signal is present only if the **Enable full duplex flow control** option is turned on.
xoff_gen	In	XOFF Pause Frame Generate. When this signal is set to 1 by the user application for at least one tx_clk clock cycle, the MAC function generates, independently of the receive FIFO status, a pause frame with the pause quanta programmed in the pause_quant register. This signal is ignored if the xoff_gen bit in the command_config register is set to 1. This signal is present only if the **Enable full duplex flow control** option is turned on.

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

Signal Name	Direction	Description
magic_sleep_n	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 MAGIC_ENA bit in the command_config 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. This signal is present only if the **Enable magic packet detection** option is turned on.
magic_wakeup	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. This signal is present only if the **Enable magic packet detection** option is turned on.

MII/GMII/RGMII Signals

The Ethernet-side interface for the MAC block uses the standard MII, GMII and RGMII interfaces to connect to an external PHY device. Table 4–39 lists the MII/GMII/RGMII signals.

Table 4–39. GMII/RGMII/MII Signals (Part 1 of 2)

Signal Name	Direction	Description
GMII Transmit		
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		
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		
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		
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		
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.

Table 4–39. GMII/RGMII/MII Signals (Part 2 of 2)

Signal Name	Direction	Description
MII Receive		
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		
m_rx_col	In	Collision detection. Asserted by the PHY device to indicate a collision during a frame transmission. This signal is not used in full- 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. This signal is not used in full-duplex mode or when the MAC function operates in gigabit.

PHY Management Signals

Table 4–40 lists the PHY management signals.

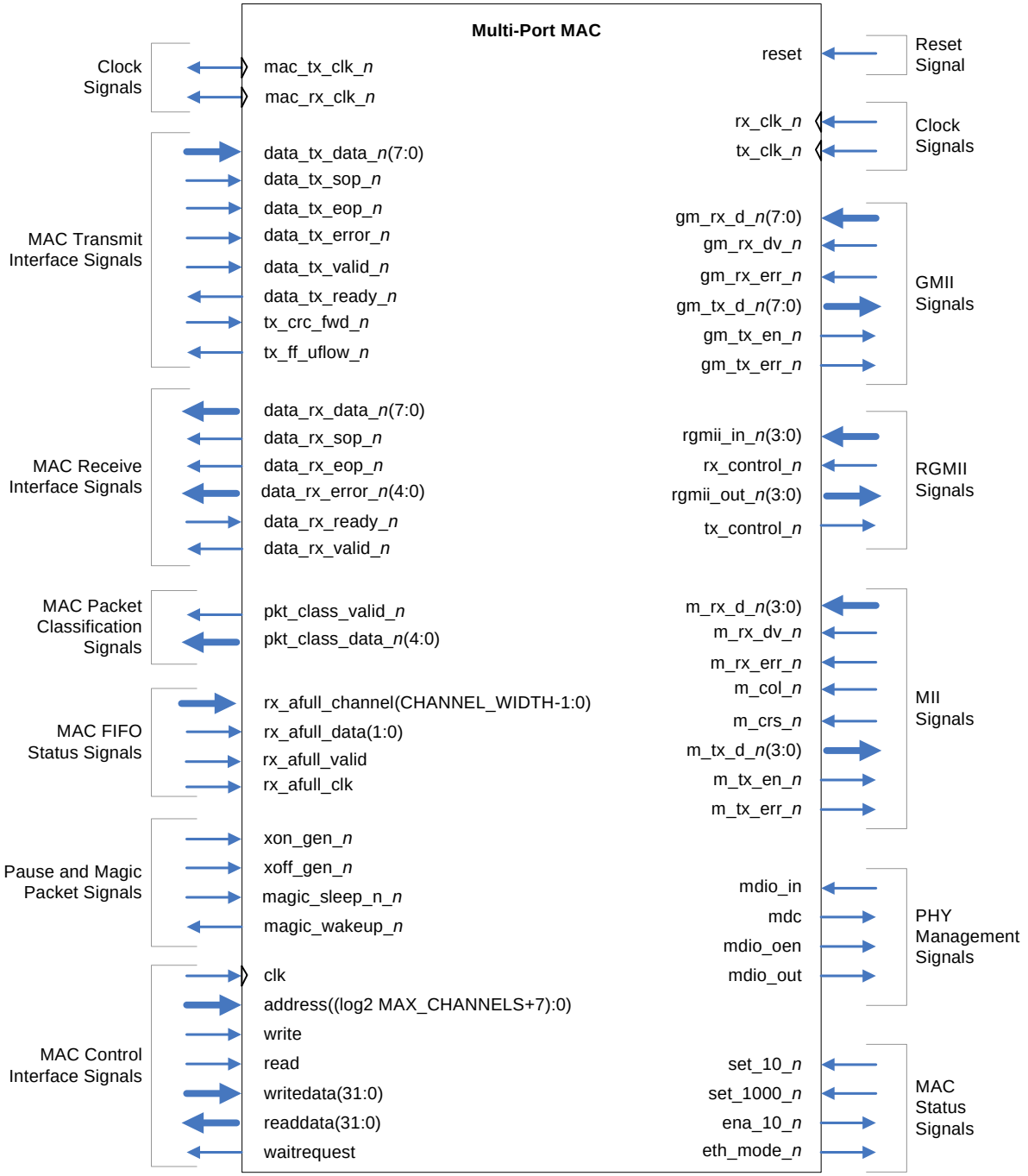
Table 4–40. PHY Management Interface Signals

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.

10/100/1000 Multi-Port Ethernet MAC Signals

Figure 4–51 shows all I/O signals of the 10/100/1000 multi-port Ethernet MAC function, a variation of MAC without internal FIFOs.

Figure 4–51. Multi-Port Ethernet MAC Signals



Clock and Reset Signals

Table 4–41 describes the clock signals.

Table 4–41. Clock Signals

Signal Name	Avalon-ST Signal Type	Direction	Description
mac_rx_clk	clk	Out	Receive MAC clock (2.5/25/125 MHz) for the Avalon-ST receive data and receive packet classification interfaces.
mac_tx_clk	clk	Out	Transmit MAC clock (2.5/25/125 MHz) for the Avalon-ST transmit data interface.

For more information about the rest of the clock and reset signals, refer to "Clock and Reset Signal" on page 4–74.

MAC Receive Interface

The MAC receive interface is an Avalon-ST source port that provides an interface to the MAC receive function. Table 4–42 describes all signals associated with the MAC receive interface.

Table 4–42. MAC Receive Interface Signals

Signal Name	Avalon-ST Signal Type	Direction	Description
data_rx_valid_*n*	valid	Out	Receive data valid. Asserted by the MAC to indicate that data on data_rx_data_*n*, data_rx_sop_*n*, data_rx_eop_*n*, and data_rx_error_*n* are valid.
data_rx_data_*n*(7:0)	data	Out	Receive data.
data_rx_sop_*n*	startofpacket	Out	Receive start of packet. Set to 1 when the first byte or word of a frame is driven on data_rx_data_*n*.
data_rx_eop_*n*	endofpacket	Out	Receive end of packet. Set to 1 when the last byte or word of frame data is driven on data_rx_data_*n*.
data_rx_ready_*n*	ready	In	Receive application ready. Asserted by the receiving application to indicate it is ready to receive data from the MAC. The signal data_rx_ready_*n* must be generated on the rising edge of data_rx_clk_*n*. If the receiving application is not ready to receive data, the packet will be dropped or truncated with an error.
data_rx_error_*n*(4:0)	error	Out	Receive error. Asserted with the final byte in the frame to indicate that an error was detected when receiving the frame. For the description of each bit, refer to the description of bits 5 to 1 in Table 4–36 on page 4–77. Bit 4 of this signal maps to bit 5 in the table and so forth.

MAC Transmit Interface

The MAC transmit interface is an Avalon-ST sink port that provides an interface to the MAC transmit function. Table 4–43 describes all signals associated with the MAC transmit interface.

Table 4–43. MAC Transmit Interface Signals

Signal Name	Avalon-ST Signal Type	Direction	Description
Avalon-ST Signals			
data_tx_valid_*n*	valid	In	Transmit data valid. Asserted by user application to indicate that data on data_tx_data_*n*, data_tx_sop_*n*, data_tx_eop_*n*, and data_tx_error_*n* are valid.
data_tx_data_*n*(7:0)	data	In	Transmit data.
data_tx_sop_*n*	startofpacket	In	Transmit start of packet. Set to 1 when the first byte in the frame is driven on data_tx_data_*n*.
data_tx_eop_*n*	endofpacket	In	Transmit end of packet. Set to 1 when the last byte in the frame (the last byte of the FCS field) is driven on data_tx_data_*n*.
data_tx_error_*n*	error	In	Transmit frame error. Asserted with the final byte in the frame to indicate that the transmitted frame is invalid. When data_tx_error_*n* is asserted, the frame is transmitted to the GMII interface with an error.
data_tx_ready_*n*	ready	Out	MAC ready. Asserted by the MAC function to indicate that it is ready to accept data from the user application.
Componet-Specific Signals			
tx_crc_fwd_*n*	–	In	Transmit CRC insertion. If set to 0 together with data_tx_eop_*n*, 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.
tx_ff_uflow_*n*	–	Out	Asserted when a FIFO underflow occurs on the transmit FIFO.

MAC Packet Classification Signals

The MAC packet classification interface is an Avalon-ST source port which streams out receive packet classifications. Table 4–44 describes the packet classification signals.

Table 4–44. MAC Packet Classification Signals

Signal Name	Avalon-ST Signal Type	Direction	Description
pkt_class_valid_*n*	valid	Out	Asserted by the MAC to indicate that classification data is valid.
pkt_class_data_*n*(4:0)	data	Out	Classification data, presented at the beginning of each packet and contains the following information: ■ pkt_class_data_*n*[4]—Set to 1 for unicast frames. ■ pkt_class_data_*n*[3]—Set to 1 for multicast frames. ■ pkt_class_data_*n*[2]—Set to 1 for broadcast frames. ■ pkt_class_data_*n*[1]—Set to 1 for VLAN frames. ■ pkt_class_data_*n*[0]—Set to 1 for stacked VLAN frames.

MAC FIFO Status Signals

The MAC FIFO status interface is an Avalon-ST sink port which streams in FIFO fill level to the MAC function. Table 4–45 describes the FIFO status signals.

Table 4–45. MAC FIFO Status Signals

Signal Name	Avalon-ST Signal Type	Direction	Description
rx_afull_valid_*n*	valid	In	The status data is valid when this signal is asserted.
rx_afull_data_*n*(1:0)	data	In	Status data which contains the following information: ■ rx_afull_data_*n*[1]—Set to 1 if the external receive FIFO reaches the critical level before it overflows. The FIFO can be considered overflow if this bit is set to 1 in the middle of a packet transfer. ■ rx_afull_data_*n*[0]—Set to 1 if the external receive FIFO reaches the initial warning level indicating that it is almost full. Upon detecting this, the MAC function generates generates pause frames.
rx_afull_channel (CHANNEL_WIDTH-1:0)	channel	In	The port number the status applies to.

MAC Status Signals

For more information about the MAC status signals, refer to "MAC Status Signals" on page 4–74.

Pause and Magic Packets Signals

For more information about pause and magic packet signals, refer to "Pause and Magic Packet Signals" on page 4–78.

MII/GMII/RGMII Signals

For more information about the MII/GMII/RGMII signals, refer to "MII/GMII/RGMII Signals" on page 4–79.

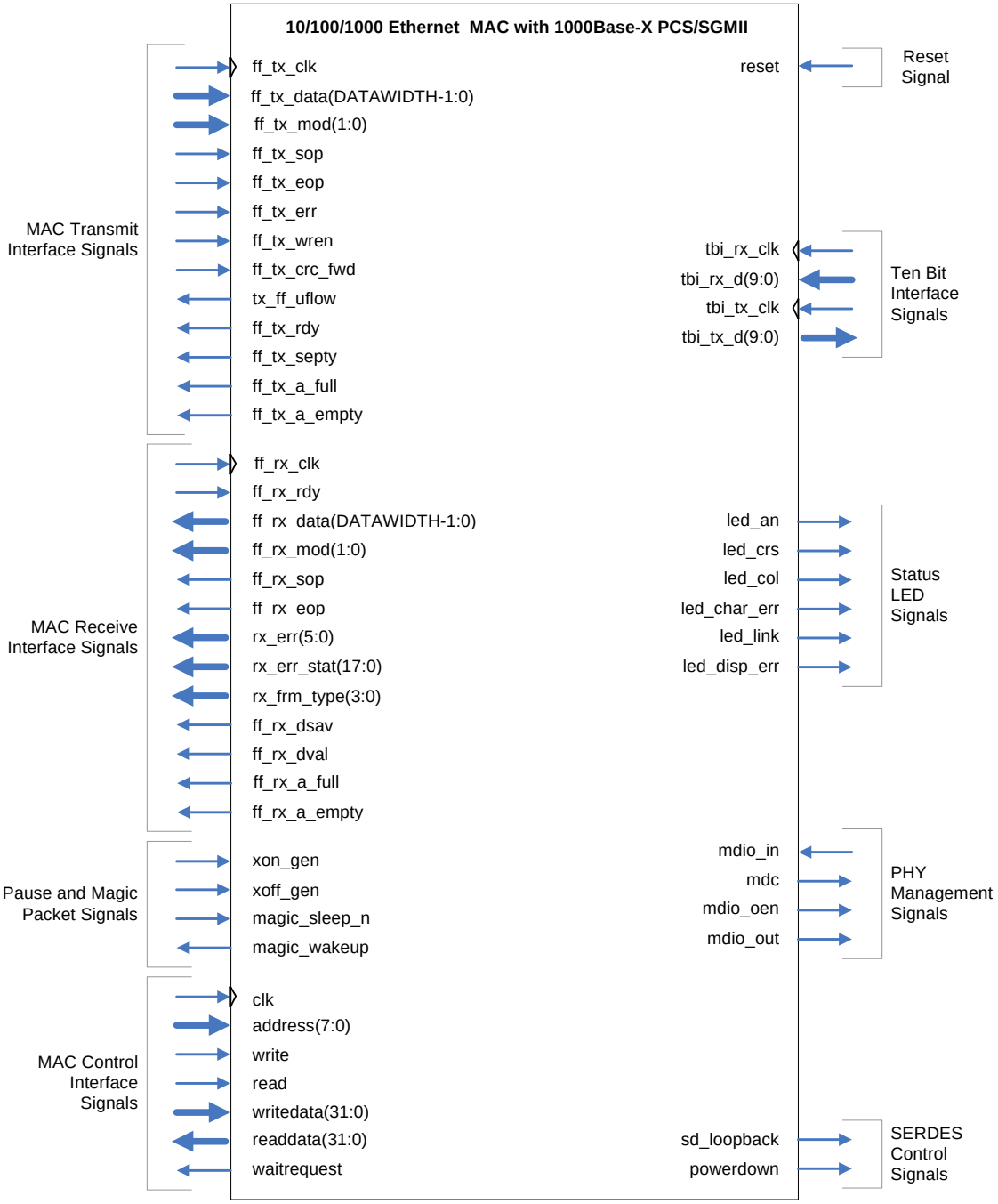
PHY Management Signals

For more information about the PHY management signals, refer to "PHY Management Signals" on page 4–80.

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

Figure 4–52 shows all I/O signals of the 10/100/1000 Ethernet MAC with 1000BASE-X/SGMII PCS. This applies to the MAC function with internal FIFOs.

Figure 4–52. 10/100/1000 Ethernet MAC with 1000BASE-X/SGMII PCS Signals



Reset Signals

For more information about the reset signal, refer to "Clock and Reset Signal" on page 4–74.

MAC Control Interface Signals

For more information about MAC system-side signals, refer to "MAC Control Interface Signals" on page 4–74.

MAC Receive Interface Signals

For more information about MAC system-side signals, refer to "MAC Receive Interface Signals" on page 4–75.

MAC Transmit Interface Signals

For more information about MAC system-side signals, refer to "MAC Transmit Interface Signals" on page 4–77.

TBI Interface Signals

If the core variation does not include an embedded PMA, the PCS block provides a 125-MHz ten-bit interface (TBI) to an external SERDES chip. Table 4–46 lists the PCS signals to an external SERDES chip.

Table 4–46. 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.

Status LED Control Signals

Table 4–47 lists the status LED control signals.

Table 4–47. Status LED Interface Signals (Part 1 of 2)

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 transmission. 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 1 when auto-negotiation completes.

Table 4–47. Status LED Interface Signals (Part 2 of 2)

Signal Name	Direction	Description
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 4–48 describes the functionality of the SERDES control signals.

Table 4–48. 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.

Pause and Magic Packet Signals

For more information about pause and magic packet signals, refer to "Pause and Magic Packet Signals" on page 4–78.

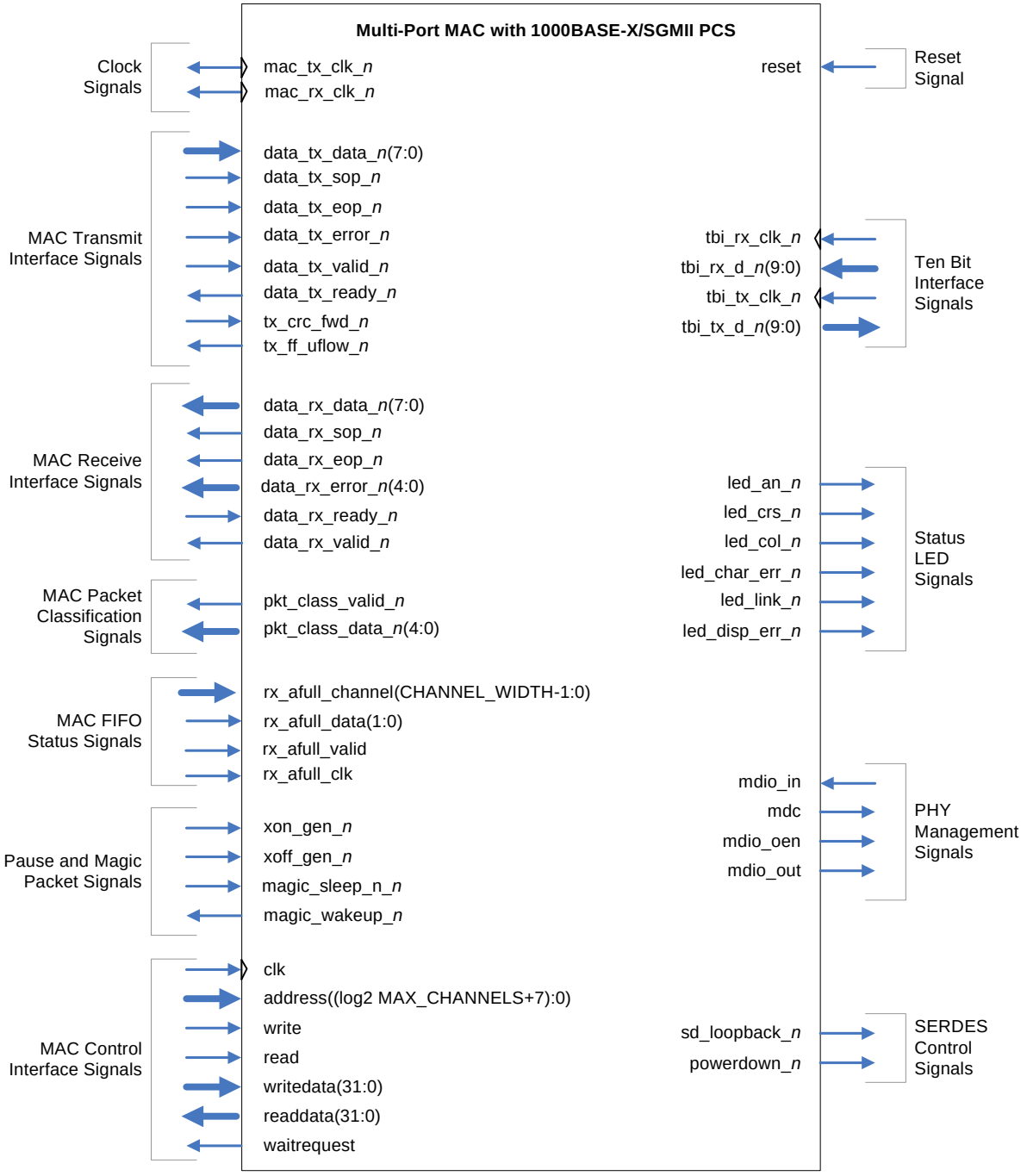
Phy Management Signals

For more information about PHY Management signals, refer to "Table 4–40 lists the PHY management signals." on page 4–80.

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

Figure 4–53 shows all I/O signals of the 10/100/1000 multi-port Ethernet MAC, a variation of MAC without internal FIFOs, with 1000BASE-X/SGMII PCS.

Figure 4–53. Multi-Port Ethernet MAC with 1000BASE-X/SGMII PCS Signals



For more information on the signals, refer to the respective sections shown in Table 4–49.

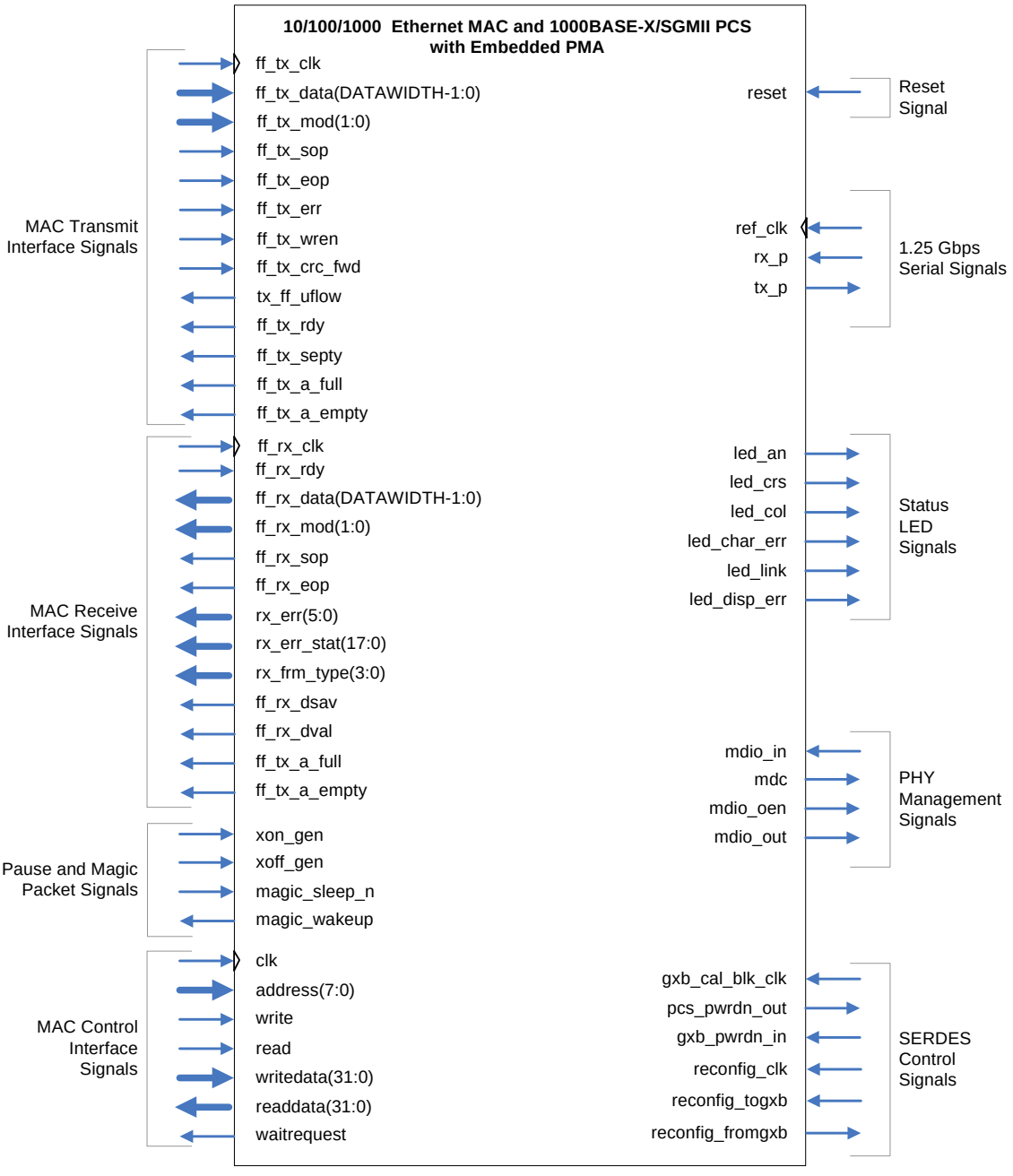
Table 4–49. References

Interface Signal	Section
Clock and reset signals	"Clock and Reset Signals" on page 4–82
MAC control interface	"MAC Control Interface Signals" on page 4–74
MAC transmit interface	"MAC Receive Interface" on page 4–82
MAC receive interface	"MAC Transmit Interface" on page 4–83
MAC packet classification signals	"MAC Packet Classification Signals" on page 4–83
MAC FIFO status signals	"MAC FIFO Status Signals" on page 4–84
Pause and magic packet signals	"Pause and Magic Packet Signals" on page 4–78
PHY management signals	"PHY Management Signals" on page 4–80
Ten-bit interface	"TBI Interface Signals" on page 4–87
Status LED signals	"Status LED Control Signals" on page 4–87
SERDES control signals	"SERDES Control Signals" on page 4–88

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

Figure 4–54 shows all I/O signals of the 10/100/1000 Ethernet MAC and 1000BASE-X/SGMII PCS with an embedded PMA. This applies to the MAC without internal FIFOs.

Figure 4–54. 10/100/1000 Ethernet MAC and 1000BASE-X/SGMII PCS With Embedded PMA Signals



Reset Signals

For more information about the reset signal, refer to "Clock and Reset Signal" on page 4–74.

MAC Control Interface Signals

For more information about MAC system-side signals, refer to "MAC Control Interface Signals" on page 4–74.

MAC Receive Interface Signals

For more information about MAC system-side signals, refer to "MAC Receive Interface Signals" on page 4–75.

MAC Transmit Interface Signals

For more information about MAC system-side signals, refer to "MAC Transmit Interface Signals" on page 4–77.

1.25 Gbps Serial Interface

If the variant includes an embedded PMA, the PMA provides a 1.25 GHz serial interface. Table 4–50 lists the MDI signals.

Table 4–50. 1.25 Gbps MDI Interface Signals

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

Pause and Magic Packet Signals

For more information about pause and magic packet signals, refer to "Pause and Magic Packet Signals" on page 4–78.

PHY Management Signals

For more information about PHY Management Signals, refer to "Table 4–40 lists the PHY management signals." on page 4–80.

Status LED Control Signals

For more information about Status LED Control Signals, refer to "Status LED Control Signals" on page 4–87.

SERDES Control Signals

Table 4–51 describes the functionality of the SERDES control signals. These signals apply only to PMA blocks implemented in devices with GX transceivers.

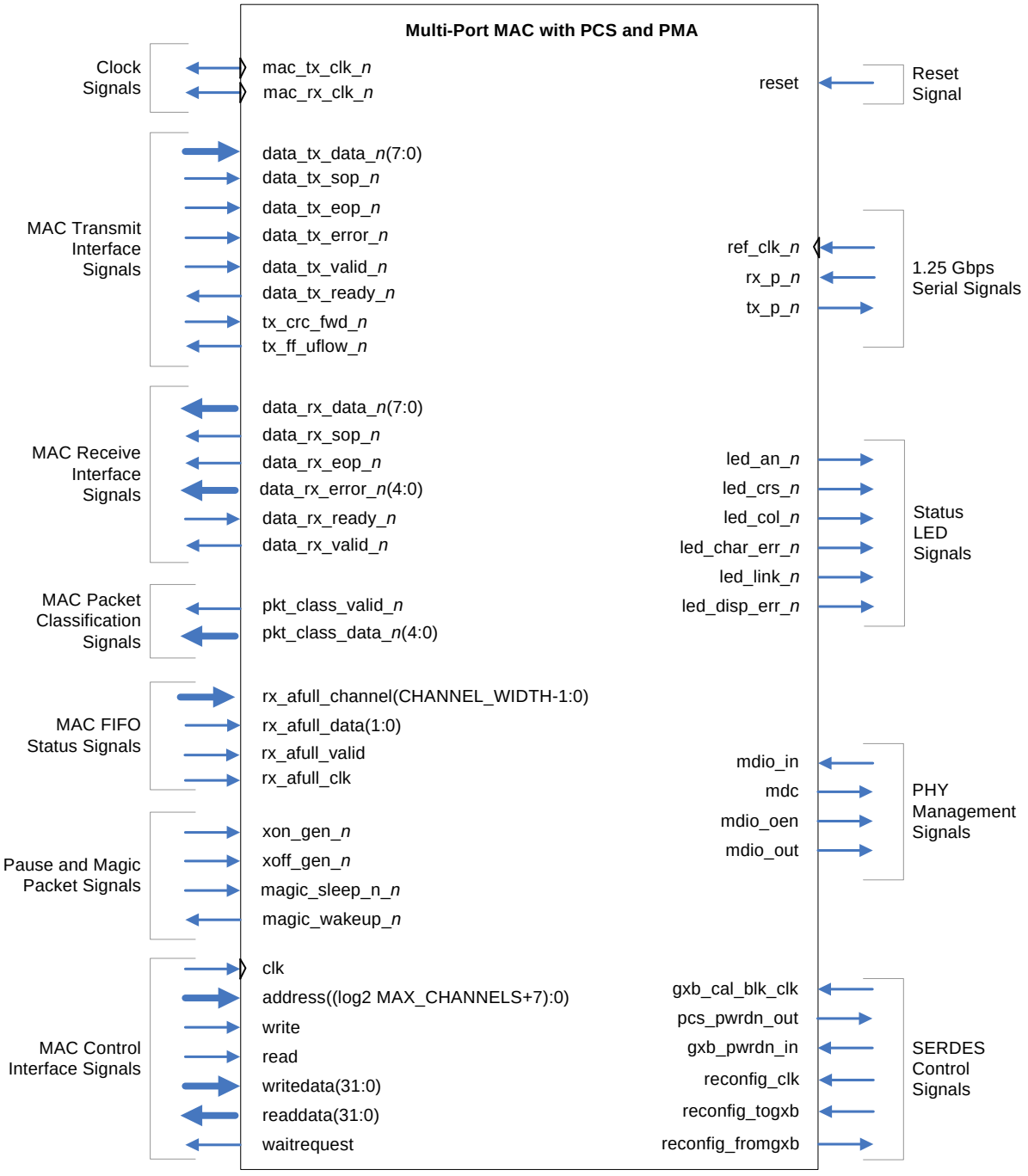
Table 4–51. SERDES Control Signal

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 tranceiver quad block when set to 1. Only implemented when an internal SERDES is used with the option to export the power-down signal.
gxb_cal_blk_clk	In	Calibration block clock for the ALT2GXB module (SERDES). This clock is typically tied to the 125 MHz ref_clk. Only implemented when an internal SERDES is used.
reconfig_clk	In	Reference clock for the dynamic reconfiguration controller. The implementation of the MegaCore function requires this clock to operate between 75–125 MHz. If you use a dynamic reconfiguration controller in your design to dynamically control the transceiver, both the reconfiguration controller and the MegaCore function require this clock. Tie this clock low if an external reconfiguration controller is not used.
reconfig_togxb[n:0]	In	Driven from an external dynamic reconfiguration controller. Supports the selection of multiple transceiver channels for dynamic reconfiguration. Tie this bus to 3'b010 if an external reconfiguration controller is not used.
reconfig_fromgxb[n:0]	Out	Driven to an external dynamic reconfiguration controller. The bus identifies the transceiver channel whose settings are being transmitted to the reconfiguration controller. Leave this bus disconnected if an external reconfiguration controller is not used.

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

Figure 4–55 shows all I/O signals of the 10/100/1000 multi-port Ethernet MAC, a variation of MAC without internal FIFOs, with 1000BASE-X/SGMII PCS and embedded PMA function.

Figure 4–55. Multi-Port Ethernet MAC with 1000BASE-X/SGMII PCS and Embedded PMA Signals



For more information on the signals, refer to the respective sections shown in Table 4–52.

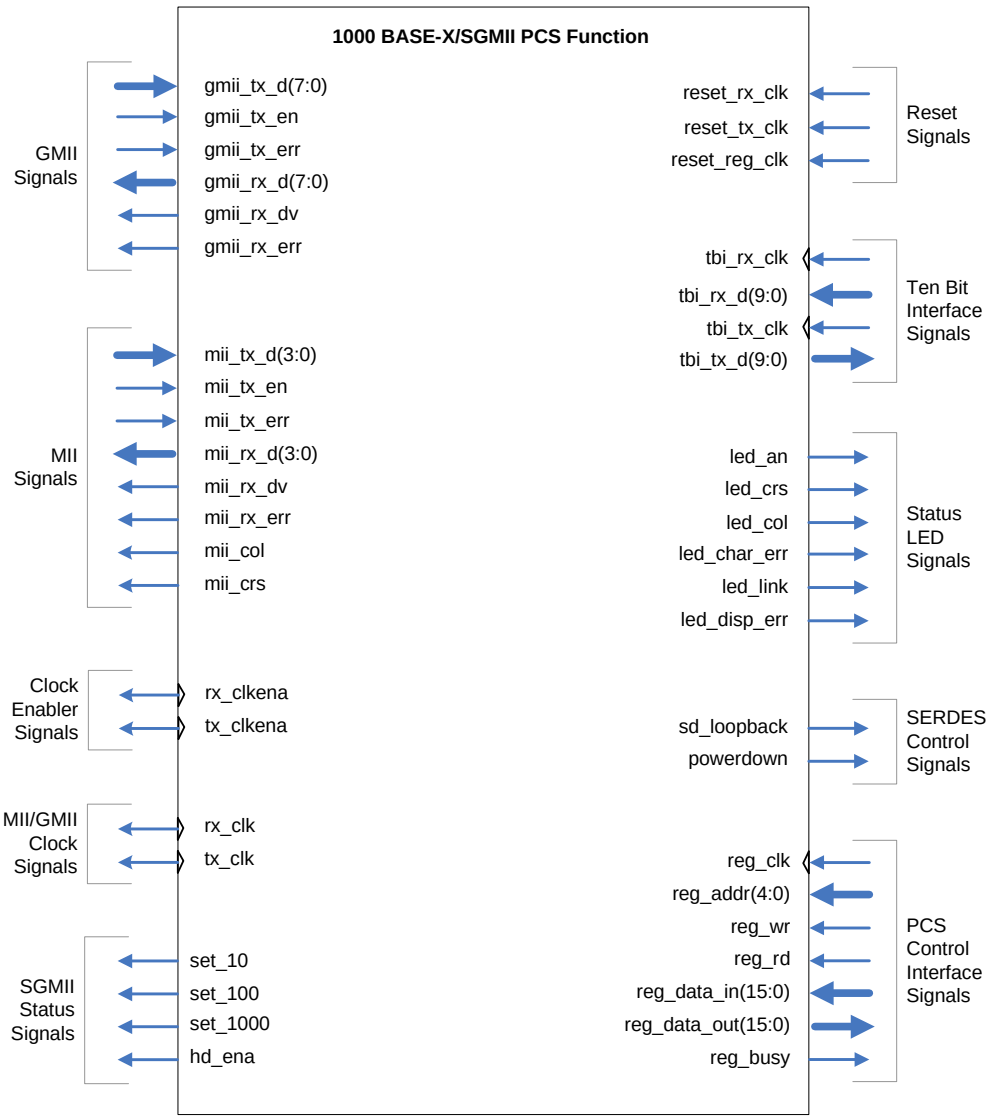
Table 4–52. References

Interface Signal	Section
Clock and reset signals	"Clock and Reset Signals" on page 4–82
MAC control interface	"MAC Control Interface Signals" on page 4–74
MAC transmit interface	"MAC Receive Interface" on page 4–82
MAC receive interface	"MAC Transmit Interface" on page 4–83
MAC packet classification signals	"MAC Packet Classification Signals" on page 4–83
MAC FIFO status signals	"MAC FIFO Status Signals" on page 4–84
Pause and magic packet signals	"Pause and Magic Packet Signals" on page 4–78
PHY management signals	"PHY Management Signals" on page 4–80
1.25 Gbps Serial Signals	"1.25 Gbps Serial Interface" on page 4–92
Status LED signals	"Status LED Control Signals" on page 4–87
SERDES control signals	"SERDES Control Signals" on page 4–93

1000BASE-X/SGMII PCS Signals

Figure 4–56 shows all I/O signals of the 1000BASE-X/SGMII PCS function.

Figure 4–56. 1000BASE-X/SGMII PCS Function Signals



Note to Figure 4–56:

(1) The clock enabler signals are present only in SGMII mode.

PCS Control Interface Signals

Table 4–53 describes the signals that comprise the control interface for the PCS function.

Table 4–53. 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	16-bit word-aligned 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.

Reset Signals

Table 4–31 lists the reset signals which you can use to reset the PCS function.

Table 4–54. Reset Signals

Signal Name	Direction	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.

MII/GMII Clocks and Clock Enablers

Data transfers on the MII/GMII interface are synchronous to the receive and transmit clocks. Table 4–55 describes these clock signals.

Table 4–55. MAC Clock Signals

Signal Name	Direction	Description
rx_clk	Out	Receive clock. This clock is derived from the TBI clock tbi_rx_clk and set to 125 MHz.
tx_clk	Out	Transmit clock. This clock is derived from the TBI clock tbi_tx_clk and set to 125 MHz.
rx_clkena	Out	Receive clock enabler. In SGMII mode, this signal enables rx_clk.
tx_clkena	Out	Transmit clock enabler. In SGMII mode, this signal enables tx_clk.

GMII Interface

Table 4–56 describes the GMII transmit and receive signals.

Table 4–56. GMII Interface Signals

Signal Name	Direction	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, gmii_tx_d[7:0] 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 4–57 describes the MII transmit and receive signals.

Table 4–57. MII Interface Signals

Signal Name	Direction	Description
MII Transmit Interface		
mii_tx_d(3:0)	In	MII transmit data bus.
mii_tx_en	In	Asserted to indicate data on the data bus, mii_tx_d[3:0] is valid.
mii_tx_err	In	Asserted to indicate to the PHY device that the current frame sent is invalid.
MII Receive Interface		
mii_rx_d(3:0)	Out	MII receive data bus.
mii_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 of the CRC field is received.
mii_rx_err	Out	Asserted by the PHY to indicate that the current frame contains erroneous data.
mii_col	Out	Collision detection. Asserted by the PCS function to indicate that a collision was detected during a frame transmission.
mii_crs	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 4–58 lists the SGMII status signals.

Table 4–58. SGMII Status Signals

Signal Name	Direction	Description
set_1000	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: ■ the USE_SGMII_AN bit is set to 1 and a gigabit link is established with the link partner, as decoded from the partner_ability register ■ the USE_SGMII_AN bit is set to 0 and the SGMII_SPEED bit is set to 10
set_100	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: ■ the USE_SGMII_AN bit is set to 1 and a 100Mbps link is established with the link partner, as decoded from the partner_ability register ■ the USE_SGMII_AN bit is set to 0 and the SGMII_SPEED bit is set to 01
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: ■ the USE_SGMII_AN bit is set to 1 and a 10Mbps link is established with the link partner, as decoded from the partner_ability register ■ 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: ■ the USE_SGMII_AN bit is set to 1 and a half-duplex link is established with the link partner, as decoded from the partner_ability register ■ the USE_SGMII_AN bit is set to 0 and the SGMII_DUPLEX bit is set to 1

TBI Interface Signals for External SERDES Chip

For more information about TBI Interface signals, refer to "TBI Interface Signals" on page 4–87.

Status LED Control Signals

For more information about Status LED signals, refer to "Status LED Control Signals" on page 4–87.

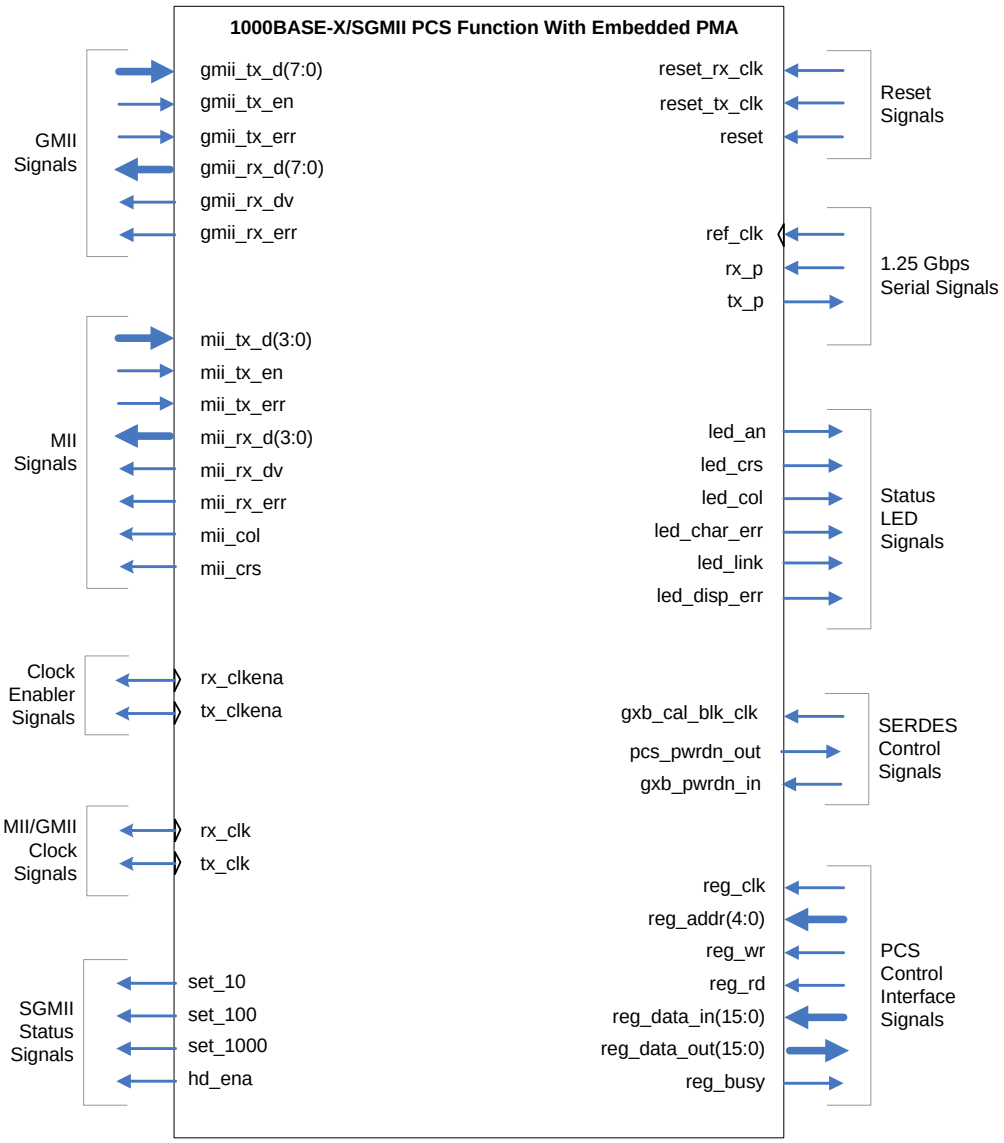
SERDES Control Signals

For more information about SERDES Control signals, refer to"SERDES Control Signals" on page 4–88.

1000BASE-X/SGMII PCS and PMA Signals

Figure 4–57 shows all I/O signals of the 1000BASE-X/SGMII PCS function with an embedded PMA.

Figure 4–57. 1000BASE-X/SGMII PCS Function and PMA Signals



Note to Figure 4–57:

(1) The clock enabler signals are present only in SGMII mode.

For more information on the signals, refer to the respective sections shown in Table 4–52.

Table 4–59. References

Interface Signal	Section
Reset signals	"Reset Signals" on page 4–97
MII/GMII clocks and clock enablers	"MII/GMII Clocks and Clock Enablers" on page 4–97
PCS control interface	"PCS Control Interface Signals" on page 4–97
GMII signals	"GMII Interface" on page 4–98
MII signals	"MII Interface" on page 4–98
SGMII status signals	"SGMII Status Signals" on page 4–99
1.25 Gbps Serial Signals	"1.25 Gbps Serial Interface" on page 4–92
Status LED signals	"Status LED Control Signals" on page 4–87
SERDES control signals	"SERDES Control Signals" on page 4–93

5. Testbench

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.

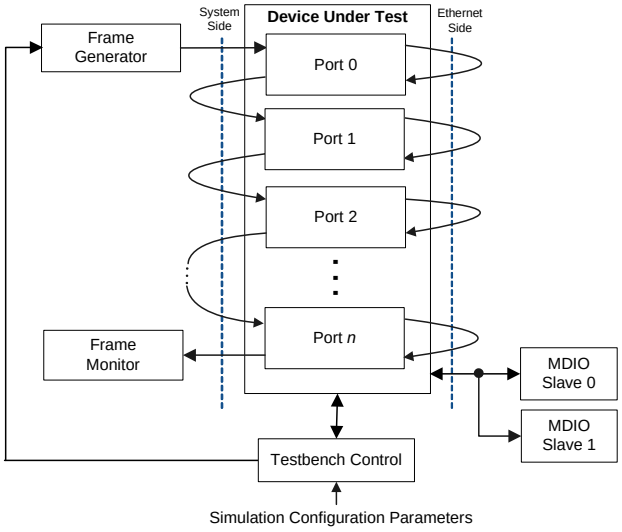
The provided testbench applies only to custom MegaCore function variations created using the MegaWizard Plug-in Manager flow.

For more information about the testbench files generated and step-by-step instructions for simulating your design using the ModelSim simulator or other simulators, refer to Chapter 2, Getting Started.

Testbench Architecture

Figure 5–1 illustrates the testbench architecture for the Triple Speed Ethernet MegaCore function.

Figure 5–1. Triple Speed Ethernet Testbench Architecture



Testbench Components

The testbench comprises the following modules:

- Device under test (DUT)—Your custom MegaCore function variation
- Avalon-ST Ethernet frame generator—Simulates a user application connected to the MAC system-side interface. It generates frames on the Avalon-ST transmit interface.

- Avalon-ST Ethernet frame monitor—Simulates a user application receiving frames from the MAC system-side interface. It monitors the Avalon-ST receive interface and decodes all data received.
- MII/RGMII/GMII Ethernet frame generator—Simulates a MAC function that sends frames to the PCS function.
- MII/RGMII/GMII Ethernet frame monitor—Simulates a MAC function that receives frames from the PCS function and decodes them.
- MDIO slaves—Simulates a PHY management interface. It responds to an MDIO master transactor.
- Clock and reset generator.

Table 5–1 lists the interfaces, frame generator and frame monitor for each configuration.

Table 5–1. Testbench Components

Configuration	System-Side Interface	Ethernet-Side Interface	Frame Generator	Frame Monitor
MAC only	Avalon-ST	GMII/MII/RGMII	Avalon-ST Frame Generator	Avalon-ST Frame Monitor
MAC with PCS	Avalon-ST	TBI	Avalon-ST Frame Generator	Avalon-ST Frame Monitor
MAC with PCS and embedded PMA	Avalon-ST	1.25 Gbps	Avalon-ST Frame Generator	Avalon-ST Frame Monitor
PCS only	GMII/MII	TBI	GMII/MII Frame Generator	GMII/MII Frame Monitor
PCS with embedded PMA	GMII/MII	1.25 Gbps	GMII/MII Frame Generator	GMII/MII Frame Monitor

Verification

The testbench is self-checking and determines the success of a simulation 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.

The testbench verifies the following functionality:

- Transmit and receive datapaths are functionally correct.
- Ethernet frames generated by the frame generator are received by the frame monitor.
- Additional checks for configurations that contain the MAC function:
  - Correct CRC-32 is inserted.
  - Short frames are padded 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.
  - CRC-32 is optionally discarded before the frames are received by the traffic monitor.

- Additional checks for configurations that contain the PCS function with optional embedded PMA:
  - Transmit frames generated by the frame generator are correctly encapsulated.
  - Received frames are de-encapsulated before they are forwarded to the frame monitor.

Configuration

The testbench is configured, by default, to operate in loopback mode. Frames sent through the transmit path are looped back into the receive path.

Separate data paths can be configured for single-channel MAC with internal FIFOs. In this configuration, the MII/GMII Ethernet frame generator is enabled and the testbench control block simulates independent yet complete receive and transmit datapaths.

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 B, Simulation Parameters.

The device under test is configured with the following default settings:

- Link speed is set to Gigabit except for configurations that contain Small MAC. For Small MACs, the default speed is 100 Mbps.
- Five Ethernet frames of payload length 100, 101, 102, 103 and 104 bytes are transmitted to the system-side interface and looped back on the ethernet-side interface.
- Default settings for the MAC function:
  - The `command_config` register is set to 0x0408003B.
  - Promiscuous mode is enabled.
  - The maximum frame length, register `frm_length`, is configured to 1518.
  - For single-channel MAC with internal FIFOs, the transmit FIFO is set to start data transmission as soon as the FIFO level reaches `tx_section_full`. The receive FIFO is set to begin forwarding Ethernet frames to the Avalon-ST receive interface when the FIFO level reaches `rx_section_full`.
- Default setting for the PCS function:
  - The `if_mode` register is set to 0x0000.
  - Auto-negotiation between the local PHY and remote link PHY is bypassed.

Test Flow

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

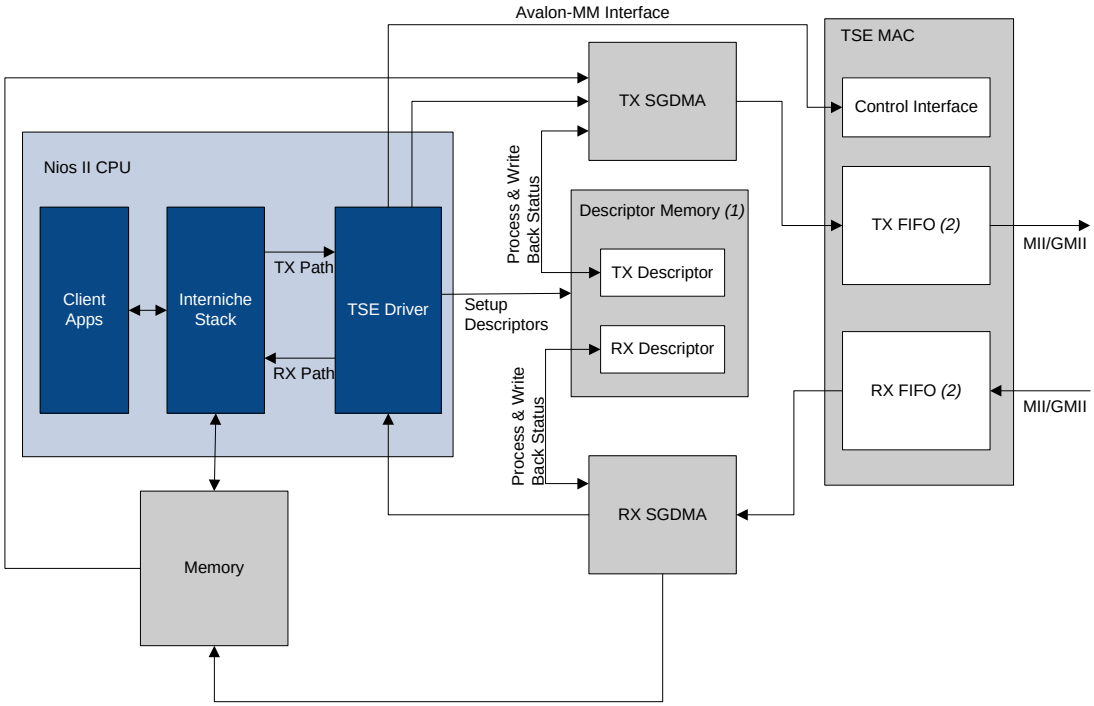
- Initializes the DUT registers.
- Starts transmission. For single-channel MAC with internal FIFOs configurations, clears the FIFOs.

- Ends transmission and checks the following elements to determine that the simulation is successful:
  - No Ethernet protocol errors detected.
  - Ethernet frames generated and transmitted are received by the frame monitor.

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



Notes to Figure 6–1:

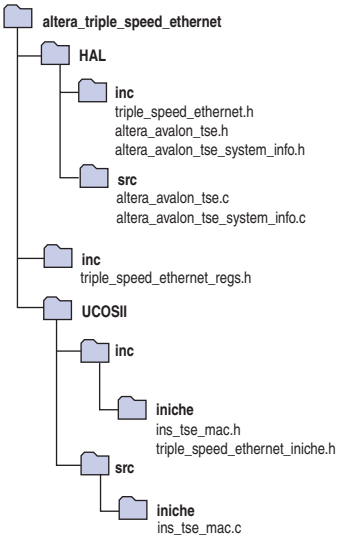
(1) The first *n* bytes are reserved for SGDMA descriptors, where *n* = (Total number of descriptors + 3) × 32. Applications must not use this memory region.

(2) For MAC variations without internal FIFOs, the transmit and receive FIFOs are external FIFOs.

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



PHY Definition

The software driver only supports the following PHYs by default:

- National DP83848C (10/100)
- National DP83865 (10/100/1000)
- Marvell 88E1111 (10/100/1000)
- Marvell 88E1145 (Quad PHY, 10/100/1000).

You can extend the software driver to support other PHYs by defining the PHY profile using the structure `alt_tse_phy_profile` and adding it to the system using the function `alt_tse_phy_add_profile()`. For each PHY instance, use the structure `alt_tse_system_phy_struct` to define it and the function `alt_tse_system_add_sys()` to add the instance to the system.

The software driver automatically detects the PHY's operating mode and speed if the PHY conforms to the following specifications:

- One bit to specify duplex and two consecutive bits (the higher bit being the most significant bit) to specify the speed in the same extended PHY specific register.
- The speed bits are set according to the convention shown in Table 6–1.

Table 6–1. PHY Speed Bit Values

Speed (Mbps)	PHY Speed Bits	
	MSB	LSB
1000	1	0
100	0	1
10	0	0

For PHYs that don't conform to the aforementioned specifications, you can write a function to retrieve the PHY's operating mode and speed, and set the field `*link_status_read` in the PHY data structure to your function's address.

You can also execute a function to initialize a PHY profile or a PHY instance by setting the function pointer (`*phy_cfg` and `*tse_phy_cfg`) in the respective structures to the function's address.

Example 6–1 and Example 6–2 shows PHY profile and instance data structures.

Example 6–1. PHY Profile Structure

```
typedef struct alt_tse_phy_profile_struct{ /* PHY profile */

   /*The name of the PHY*/
   char name[80];

   /*Organizationally Unique Identifier*/
   alt_u32 oui;

   /*PHY model number*/
   alt_u8 model_number;

   /*PHY revision number*/
   alt_u8 revision_number;

   /*The location of the PHY Specific Status Register*/
   alt_u8 status_reg_location;

   /*The location of the Speed Status bit in the PHY Specific Status
   Register*/
   alt_u8 speed_lsb_location;

   /*The location of the Duplex Status bit in the PHY Status Specific
   Register*/
   alt_u8 duplex_bit_location;

   /*The location of the Link Status bit in PHY Status Specific
   Register*/
   alt_u8 link_bit_location;

   /*PHY initialization function pointer—profile specific*/
   alt_32 (*phy_cfg)(np_tse_mac *pmac);

   /*Pointer to the function that reads and returns 32-bit link status.Possible status:
   full duplex (bit 0 = 1), half duplex (bit 0 = 0),gigabit (bit 1 = 1),
   100Mbps (bit 2 = 1), 10Mbps (bit 3 = 1),invalid speed (bit 16 = 1).*/
   alt_u32 (*link_status_read)(np_tse_mac *pmac);

} alt_tse_phy_profile;
```

Example 6–2. PHY Instance Structure

```
typedef struct alt_tse_system_phy_struct { /* PHY instance */

    /* PHY's MDIO address */
   alt_32tse_phy_mdio_address;
   /* PHY initialization function pointer—instance specific */
    alt_32 (*tse_phy_cfg)(np_tse_mac *pmac);

} alt_tse_system_phy;
```

Using Multiple SG-DMA Descriptors

To successfully use multiple SG-DMA descriptors in your application, make the following modifications:

- Set the value of the constant `ALTERA_TSE_SGDMA_RX_DESC_CHAIN_SIZE` in **altera_avalon_tse.h** to the number of descriptors optimal for your application. The default value is 1 and the maximum value is determined by the constant `NUMBIGBUFFS`. For TCP applications, Altera recommends that you use the default value.
- Increase the amount of memory allocated for the Interniche stack.

  The memory space for the Interniche stack is allocated using the Interniche function pk_alloc(). Although user applications and other network interfaces such as LAN91C111 can share the memory space, Altera recommends that you use this memory space for only one purpose, that is storing unprocessed packets for the Triple Speed Ethernet MegaCore function. Each SG-DMA descriptor used by the device driver consumes a buffer size of 1536 bytes (defined by the constant `BIGBUFSIZE`) in the memory space. To achieve reasonable performance and to avoid memory exhaustion, add a new constant named `NUMBIGBUFS` to your application and set its value using the following guideline:

  `NUMBIGBUFS` = <*current value*> + <*number of SG-DMA descriptors*>

  By default, the constant `NUMBIGBUFS` is set to `30` in **ipport.h**. If you changed the default value in the previous release of the MegaCore function to optimize performance and resource usage, use the modified value to compute the new value of `NUMBIGBUFS`.

API Functions

This section describes each provided API function in alphabetical order.

alt_tse_mac_get_common_speed()

Prototype:	`alt_tse_mac_get_common_speed(np_tse_mac *pmac)`
Thread-safe:	No.
Available from ISR:	No.
Include:	<**altera_avalon_tse.h**>
Description:	The `alt_tse_mac_get_common_speed()` obtains the common speed supported by the PHYs connected to a multi-port MAC and remote link partners.
Parameter:	`pmac`—A pointer to the base of the MAC control interface.
Return:	`TSE_PHY_SPEED_1000` if the PHYs common speed is 1000 Mbps. `TSE_PHY_SPEED_100` if the PHYs common speed is 100 Mbps. `TSE_PHY_SPEED_10` if the PHYs common speed is 10 Mbps. `TSE_PHY_SPEED_NO_COMMON` if there isn't a common speed among the PHYs.
See also:	`alt_32 alt_tse_mac_set_common_speed()`

alt_tse_mac_set_common_speed()

Prototype:	`alt_tse_mac_set_common_speed(np_tse_mac *pmac, alt_32 common_speed)`
Thread-safe:	No.
Available from ISR:	No.
Include:	<**altera_avalon_tse.h**>
Description:	The `alt_tse_mac_set_common_speed()` sets the speed of a multi-port MAC and the PHYs connected to it.
Parameter:	`pmac`—A pointer to the base of the MAC control interface. `common_speed`—The speed to set.
Return:	`TSE_PHY_SPEED_1000` if the PHYs common speed is 1000 Mbps. `TSE_PHY_SPEED_100` if the PHYs common speed is 100 Mbps. `TSE_PHY_SPEED_10` if the PHYs common speed is 10 Mbps. `TSE_PHY_SPEED_NO_COMMON` if there isn't a common speed among the PHYs. The current speed of the MAC and PHYs is not changed.
See also:	`alt_32 alt_tse_mac_get_common_speed()`

alt_tse_phy_add_profile()

Prototype:	`alt_tse_phy_add_profile(alt_tse_phy_profile *phy)`
Thread-safe:	No.
Available from ISR:	No.
Include:	<**altera_avalon_tse.h**>
Description:	The `alt_tse_phy_add_profile()` function adds a new PHY to the PHY profile. Use this function if you want to use PHYs other than Marvell 88E1111, Marvell Quad PHY 88E1145, National DP83865, and National DP83848C.
Parameter:	`phy`—A pointer to the PHY structure.
Return:	`ALTERA_TSE_MALLOC_FAILED` if the operation is not successful. Otherwise, the index of the newly added PHY is returned.

alt_tse_system_add_sys()

Prototype:	`alt_tse_system_add_sys(alt_tse_system_mac *psys_mac, alt_tse_system_sgdma *psys_sgdma, alt_tse_system_desc_mem *psys_mem, alt_tse_system_shared_fifo *psys_shared_fifo, alt_tse_system_phy *psys_phy)`
Thread-safe:	No.
Available from ISR:	No.
Include:	<**system.h**> <**altera_avalon_tse.h**> <**altera_avalon_tse_system_info.h**>
Description:	The `alt_tse_system_add_sys()` function defines the TSE system's components: MAC, scatter-gather DMA, memory, FIFO and PHY. This needs to be done for each port in the system.
Parameter:	`psys_mac`—A pointer to the MAC structure. `psys_sgdma`—A pointer to the scatter-gather DMA structure. `psys_mem`—A pointer to the memory structure. `psys_shared_fifo`—A pointer to the FIFO structure. `psys_phy`—A pointer to the PHY structure.
Return:	`SUCCESS` if the operation is successful. `ALTERA_TSE_MALLOC_FAILED` if the operation fails. `ALTERA_TSE_SYSTEM_DEF_ERROR` if one or more of the definitions are incorrect, or empty.

triple_speed_ethernet_init()

Prototype: `error_t triple_speed_ethernet_init(alt_niche_dev *p_dev)`

Thread-safe: No.

Available from ISR: No.

Include: <**triple_speed_ethernet_iniche.h**>

Description: The `triple_speed ethernet_init()` function opens and intializes 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.

Parameter: `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()

Prototype: `int tse_mac_close(int iface)`

Thread-safe: No.

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

Parameter: `iface`—The index of the MAC interface. This argument is reserved for configurations that contain multiple MAC instances.

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:	No.
Available from ISR:	No.
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.
Parameter:	`net`—The NET structure of the Triple Speed Ethernet MAC instance. `data`—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. `data_bytes`—The total number of bytes in the Ethernet frame including the additional padding bytes as specified by `ETHHDR_BIAS`.
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_setGMIImode(np_tse_mac *pmac)`
Thread-safe:	No.
Available from ISR:	No.
Include:	<**triple_speed_ethernet_iniche.h**>
Description:	The `tse_mac_setGMIImode()` 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.
Parameter:	`pmac`—A pointer to the MAC control interface base address.
Return:	`SUCCESS`
See also:	`tse_mac_setMIImode()`

tse_mac_setMIImode()

Prototype:	`int tse_mac_setMIImode(np_tse_mac *pmac)`
Thread-safe:	No.
Available from ISR:	No.
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.
Parameter:	`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:	No.
Available from ISR:	No.
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.
Parameter:	`pmac`—A pointer to the MAC control interface base address.
Return:	`SUCCESS`

Constants

Table 6–2 lists all constants defined for the MAC registers manipulation and provides links to detailed descriptions of the registers. It also list the constants that define the MAC operating mode and timeout values.

Table 6–2. Constants Mapping (Part 1 of 3)

Constant	Value	Description
`ALTERA_TSE_DUPLEX_MODE_DEFAULT`	1	0: Half-duplex 1: Full-duplex
`ALTERA_TSE_MAC_SPEED_DEFAULT`	0	0: 10 Mbps 1: 100 Mbps 2: 1000 Mbps
`ALTERA_TSE_SGDMA_RX_DESC_CHAIN_SIZE`	1	The number of SG-DMA descriptors required for the current operating mode.
`ALTERA_CHECKLINK_TIMEOUT_THRESHOLD`	1000000	The timeout value when the MAC tries to establish a link with a PHY.
`ALTERA_AUTONEG_TIMEOUT_THRESHOLD`	250000	The auto-negotiation timeout value.
Command_Config Register (Table 4–10 on page 4–35)		
`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	

Table 6–2. Constants Mapping (Part 2 of 3)

Constant	Value	Description
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 MHAS_SEL bit.
ALTERA_TSEMAC_CMD_MHASH_SEL_MSK	0x4000	
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	

Table 6–2. Constants Mapping (Part 2 of 3)

Constant	Value	Description
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 MHAS_SEL bit.
ALTERA_TSEMAC_CMD_MHASH_SEL_MSK	0x4000	
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	

Table 6–2. Constants Mapping (Part 3 of 3)

Constant	Value	Description
ALTERA_TSEMAC_CMD_CNT_RESET_OFST	31	Configures the CNT_RESET bit.
ALTERA_TSEMAC_CMD_CNT_RESET_MSK	0x80000000	
Tx_Cmd_Stat Register (Table 4–12 on page 4–39)		
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 4–13 on page 4–40)		
ALTERA_TSEMAC_RX_CMD_STAT_RXSHIFT16_OFST	25	Configures the RX_SHIFT16 bit
ALTERA_TSEMAC_RX_CMD_STAT_RXSHIFT16_MSK	0x2000000	

A. Upgrading Triple Speed Ethernet MegaCore Function

Upgrading Triple Speed Ethernet to Version 7.1 and Above

Symbol ordering and data modulo signals in the Triple Speed Ethernet MegaCore function version 7.1 have been redefined to comply with the Avalon Streaming Specifications. The redefinition affects existing designs that contain the following cores:

- Altera Triple Speed Ethernet version 6.1 in 32-bit standalone mode.
- MoreThanIP 32-bit MAC core.

Required Conversion

To upgrade the affected cores in your design to the Triple Speed Ethernet MegaCore function version 7.1 or later while maintaining compatibility with existing custom logic, add a conversion module as illustrated in Figure A–1.

Figure A–1. Triple Speed Ethernet MegaCore Function Wrapper

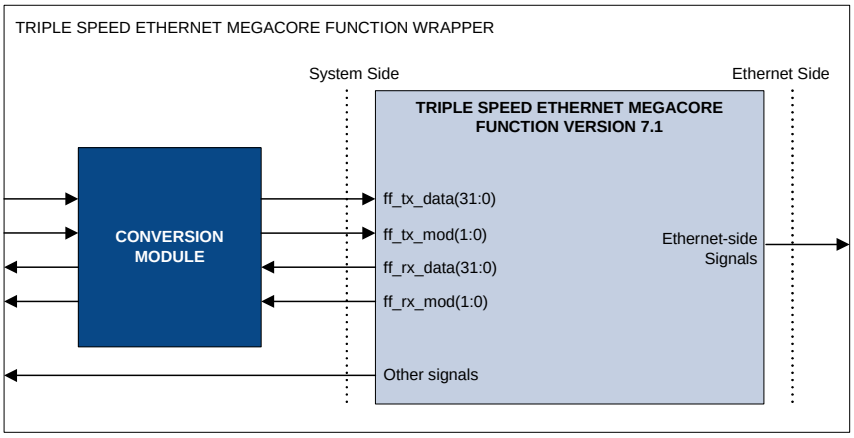


Table A–1 lists the required transmit and receive data conversions.

Table A–1. Transmit and Receive Data Conversion

Triple Speed Ethernet MegaCore Function Version 7.1 and Above	Triple Speed Ethernet Endian Conversion Wrapper
ff_tx_data(7:0) ff_rx_data(7:0)	ff_tx_data(31:24) ff_rx_data(31:24)
ff_tx_data(15:8) ff_rx_data(15:8)	ff_tx_data(23:16) ff_rx_data(23:16)
ff_tx_data(23:16) ff_rx_data(23:16)	ff_tx_data(15:8) ff_rx_data(15:8)
ff_tx_data(31:24) ff_rx_data(31:24)	ff_tx_data(7:0) ff_rx_data(7:0)

Table A–2 lists the required data modulo signals conversions.

Table A–2. Transmit and Receive Data Modulo Signals Conversion

Number of valid bytes	Modulo Signals in Triple Speed Ethernet Version 7.1 and Above	Modulo Signals in Triple Speed Ethernet Version 6.1 and MoreThanIPMAC
1	2'b11	2'b01
2	2'b10	2'b10
3	2'b01	2'b11
4	2'b00	2'b00

Upgrading Triple Speed Ethernet Version 7.1 to 7.2 or Higher

The top-level file naming convention for custom variations that contain embedded PMAs and created using the MegaWizard Plug-in Manager flow has changed in the Triple Speed Ethernet MegaCore function version 7.2 and higher. It is now named as <*variation_name*>**.v/hd** instead of <*variation_name*>**_with_pma.v/hd**, as it was in version 7.1.

Thus, when you upgrade your Triple Speed Ethernet custom variation from version 7.1 to 7.2 or higher, ensure that the current top-level file is instantiated in your system.

Functionality Configuration Parameters

You can use the parameters in Table B–1 to enable or disable specific functionality in the MAC and PCS.

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

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

Parameter	Description	Default
Supported in configurations that contain the 10/100/1000 Ethernet MAC		
ETH_MODE	10: Enables MII. 100: Enables MII. 1000: Enables GMII.	1000
HD_ENA	Sets the HD_ENA bit in the command_config register. See Table 4–10 on page 4–35.	0
TB_MACPAUSEQ	Sets the pause_quant register. See Table 4–9 on page 4–28.	15
TB_MACIGNORE_PAUSE	Sets the PAUSE_IGNORE bit in the command_config register. See Table 4–10 on page 4–35.	0
TB_MACFWD_PAUSE	Sets the PAUSE_FWD bit in the command_config register. See Table 4–10 on page 4–35.	0
TB_MACFWD_CRC	Sets the CRC_FWD bit in the command_config register. See Table 4–10 on page 4–35.	0
TB_MACINSERT_ADDR	Sets the ADDR_INS bit in the command_config register. See Table 4–10 on page 4–35.	0
TB_PROMIS_ENA	Sets the PROMIS_EN bit in the command_config register. See Table 4–10 on page 4–35.	1
TB_MACPADEN	Sets the PAD_EN bit in the command_config register. See Table 4–10 on page 4–35.	1
TB_MACLENMAX	Maximum frame length.	1518
TB_IPG_LENGTH	Sets the tx_ipg_length register. See Table 4–9 on page 4–28.	12
TB_MDIO_ADDR0	Sets the mdio_addr0 register. See Table 4–9 on page 4–28.	0
TB_MDIO_ADDR1	Sets the mdio_addr1 register. See Table 4–9 on page 4–28.	1
TX_FIFO_AE	Sets the tx_almost_empty register. See Table 4–9 on page 4–28.	8
TX_FIFO_AF	Sets the tx_almost_full register. See Table 4–9 on page 4–28.	10

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

Parameter	Description	Default
RX_FIFO_AE	Sets the rx_almost_empty register. See Table 4–9 on page 4–28.	8
RX_FIFO_AF	Sets the rx_almost_full register. See Table 4–9 on page 4–28.	8
TX_FIFO_SECTION_EMPTY	Sets the tx_section_empty register. See Table 4–9 on page 4–28.	16
TX_FIFO_SECTION_FULL	Sets the tx_section_full register. See Table 4–9 on page 4–28.	16
RX_FIFO_SECTION_EMPTY	Sets the rx_section_empty register. See Table 4–9 on page 4–28.	0
RX_FIFO_SECTION_FULL	Sets the rx_section_full register. See Table 4–9 on page 4–28.	16
MCAST_TABLEN	Specifies the first *n* addresses from MCAST_ADDRESSLIST from which multicast address is selected.	9
MCAST_ADDRESSLIST	A list of multicast addresses.	0x887654332211 0x886644352611 0xABCDEF012313 0x92456545AB15 0x432680010217 0xADB589215439 0xFFEACFE3434B 0xFFCCDDAA3123 0xADB358415439
Supported in configurations that contain the 1000BASE-X/SGMII PCS		
TB_SGMII_ENA	Sets the SGMII_ENA bit in the if_mode register. See Table 4–28 on page 4–66.	0
TB_SGMII_AUTO_CONF	Sets the USE_GMII_AN bit in the if_mode register. See Table 4–28 on page 4–66.	0

Test Configuration Parameters

You can use the parameters in Table B–2 to create custom test scenarios.

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

Table B–2. Test Configuration Parameters (Part 1 of 3)

Parameter	Description	Default
Supported in configurations that contain the 10/100/1000 Ethernet MAC		
TB_RXFRAMES	Enables local loopback on the Ethernet side (GMII/MII/RGMII). The value must always be set to 0.	0
TB_TXFRAMES	Specifies the number of frames to be generated by the Avalon-ST Ethernet frame generator.	5
TB_RXIPG	IPG on the receive path.	12

Table B–2. Test Configuration Parameters (Part 2 of 3)

Parameter	Description	Default
TB_ENA_VAR_IPG	0: A constant IPG, TB_RXIPG, is used by the GMII/RGMII/MII Ethernet frame generator. 1: Enables variable IPG on the receive path.	0
TB_LENSTART	Specifies the payload length of the first frame generated by the frame generators. The payload length of each subsequent frame is incremented by the value of TB_LENSTEP.	100
TB_LENSTEP	Specifies the payload length increment.	1
TB_LENMAX	Specifies the maximum payload length generated by the frame generators. If the payload length exceeds this value, it wraps around to TB_LENSTART. This parameter can be used to test frame length error by setting it to a value larger than the value of TB_MACLENMAX.	1500
TB_ENA_PADDING	0: Disables padding. 1: If the length of frames generated by the GMII/RGMII/MII Ethernet frame generator is less than the minimum frame length (64 bytes), the generator inserts padding bytes to the frames to make up the minimum length.	1
TB_ENA_VLAN	0: Only basic frames are generated. 1: Enables VLAN frames generation. This value specifies the number of basic frames generated before a VLAN frame is generated followed by a stacked VLAN frame.	0
TB_STOPREAD	Specifies the number of packets to be read from the receive FIFO before reading is suspended. You can use this parameter to test FIFO overflow and flow control.	0
TB_HOLDREAD	Specifies the number of clock cycles before the Avalon-ST monitor stops reading from the receive FIFO.	1000
TB_TX_FF_ERR	0: Normal behavior. 1: Drives the Avalon-ST error signal high to simulate erroneous frames transmission.	0
TB_TRIGGERXOFF	Specifies the number of clock cycles from the start of simulation before the xoff_gen signal is driven.	0
TB_TRIGGERXON	Specifies the number of clock cyles from the start of simulation before the xon_gen signal is driven high.	0
RX_COL_FRM	Specifies which frame is received with collision. Valid in fast Ethernet and half-duplex mode only.	0
RX_COL_GEN	Specifies which nibble within the frame collision occurs.	0
TX_COL_FRM	Specifies which frame is transmitted with a collision. Valid in fast Ethernet and half-duplex mode only.	0
TX_COL_GEN	Specifies which nibble within the frame collision occurs on the transmit path.	0
TX_COL_NUM	Specifies the number of consecutive collisions during retransmission.	0
TX_COL_DELAY	Specifies the delay, in nibbles, between collision and retransmission.	0
TB_PAUSECONTROL	0: GMII frame generator does not respond to pause frames. 1: Enables flow control in the GMII frame generator. it will respond to pause frames	1
TB_MDIO_SIMULATION	Enable / Disable MDIO simulation.	0

Table B–2. Test Configuration Parameters (Part 3 of 3)

Parameter	Description	Default
Supported in configurations that contain the 1000BASE-X/SGMII PCS		
TB_SGMII_HD	0: Disables half-duplex mode. 1: Enables half-duplex mode.	0
TB_SGMII_1000	0: Disables gigabit operation. 1: Enables gigabit operation.	1
TB_SGMII_100	0: Disables 100 Mbps operation. 1: Enables 100 Mbps operation.	0
TB_SGMII_10	0: Disables 10 Mbps operation. 1: Enables 10 Mbps operation.	0
TB_TX_ERR	0: Disables error generation. 1: Enables error generation.	0

Revision History

The following table shows the revision history for this user guide.

Date	Version	Changes Made
March 2009	9.0	■ Added support for Arria II GX. ■ Updated chapter 3 to include a new parameter that enables wider statistics counters. ■ Updated chapter 4 to reflect support for different speed in multi-port MACs and gated clocks elimination. ■ Updated chapter 6 to reflect enhancements made on the device drivers.
November 2008	8.1	■ Updated Chapters 3 and 4 to add description on dynamic reconfiguration. ■ Updated Chapter 6 to include a procedure to add unsupported PHYs.
May 2008	8.0	■ Revised the performance tables and device support. ■ Updated Chapters 3 and 4 to include information on MAC with multi ports and without internal FIFOs. ■ Revised the clock distribution section in Chapter 4. ■ Reorganized Chapter 5 to remove redundant information and to include the new testbench architecture. ■ Updated Chapter 6 to include new public APIs.
October 2007	7.2	■ Updated Chapter 1 to reflect new device support. ■ Updated Chapters 3 and 4 to include information on Small MAC.
May 2007	7.1	■ Added Chapters 2, 3, 5 and 6. ■ Updated contents to reflect changes and enhancements in the current version.
March 2007	7.0	Updated signal names and description.
December 2006	6.1	■ 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. ■ Initial release of document on Web.
December 2006	6.1	Initial release of document on DVD.

How to Contact Altera

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

Contact *(Note 1)*	Contact Method	Address
Technical support	Website	www.altera.com/support
Technical training	Website	www.altera.com/training
	Email	custrain@altera.com
Altera literature services	Email	literature@altera.com
Non-technical support (General)	Email	nacomp@altera.com
(Software Licensing)	Email	authorization@altera.com

Note:

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

Typographic Conventions

The following table shows the typographic conventions that this document uses.

Visual Cue	Meaning
Bold Type with Initial Capital Letters	Indicates command names, dialog box titles, dialog box options, and other GUI labels. For example, **Save As** dialog box.
bold type	Indicates directory names, project names, disk drive names, file names, file name extensions, and software utility names. For example, **\qdesigns** directory, **d:** drive, and **chiptrip.gdf** file.
Italic Type with Initial Capital Letters	Indicates document titles. For example, *AN 519: Stratix IV Design Guidelines.*
Italic type	Indicates variables. For example, *n* + 1. Variable names are enclosed in angle brackets (< >). For example, *<file name>* and *<project name>***.pof** file.
Initial Capital Letters	Indicates keyboard keys and menu names. For example, Delete key and the Options menu.
"Subheading Title"	Quotation marks indicate references to sections within a document and titles of Quartus II Help topics. For example, "Typographic Conventions."
`Courier type`	Indicates signal, port, register, bit, block, and primitive names. For example, `data1`, `tdi`, and `input`. Active-low signals are denoted by suffix `n`. For example, `resetn`. Indicates command line commands and anything that must be typed exactly as it appears. For example, `c:\qdesigns\tutorial\chiptrip.gdf`. Also indicates sections of an actual file, such as a Report File, references to parts of files (for example, the AHDL keyword `SUBDESIGN`), and logic function names (for example, `TRI`).
1., 2., 3., and a., b., c., and so on.	Numbered steps indicate a list of items when the sequence of the items is important, such as the steps listed in a procedure.
■ ■	Bullets indicate a list of items when the sequence of the items is not important.
	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 your work.

Visual Cue	Meaning
	A warning calls attention to a condition or possible situation that can cause you injury.
	The angled arrow instructs you to press Enter.
	The feet direct you to more information about a particular topic.