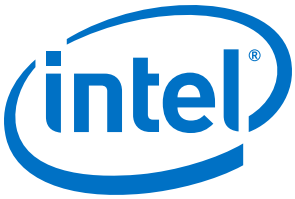


# Triple-Speed Ethernet Intel® FPGA IP User Guide

Updated for Intel® Quartus® Prime Design Suite: **19.4**

IP Version: **19.4.0**

Subscribe
Send Feedback

**UG-01008 | 2020.02.27**
Latest document on the web: **PDF** | **HTML**

# Contents

**1. About This IP Core ..... 6**
1.1. Device Family Support ..... 6
1.2. Features ..... 7
1.3. 10/100/1000 Ethernet MAC Versus Small MAC ..... 8
1.4. High-Level Block Diagrams ..... 9
1.5. Example Applications ..... 11
1.6. IP Core Verification ..... 12
1.6.1. Optical Platform ..... 13
1.6.2. Copper Platform ..... 13
1.7. Performance and Resource Utilization ..... 13
1.8. Release Information ..... 19

**2. Getting Started with Intel FPGA IP Cores ..... 21**
2.1. Design Walkthrough ..... 21
2.1.1. Creating a New Intel Quartus Prime Project ..... 21
2.1.2. Generating a Design Example or Simulation Model ..... 22
2.1.3. Simulate the System ..... 22
2.1.4. Compiling the Triple-Speed Ethernet IP Core Design ..... 23
2.1.5. Programming an FPGA Device ..... 23
2.2. Generated Files ..... 23
2.2.1. Design Constraint File No Longer Generated ..... 24

**3. Parameter Settings ..... 26**
3.1. Core Configuration ..... 26
3.2. Ethernet MAC Options ..... 27
3.3. FIFO Options ..... 29
3.4. Timestamp Options ..... 29
3.5. PCS/Transceiver Options ..... 29

**4. Functional Description ..... 32**
4.1. 10/100/1000 Ethernet MAC ..... 32
4.1.1. MAC Architecture ..... 33
4.1.2. MAC Interfaces ..... 34
4.1.3. MAC Transmit Datapath ..... 36
4.1.4. MAC Receive Datapath ..... 38
4.1.5. MAC Transmit and Receive Latencies ..... 44
4.1.6. FIFO Buffer Thresholds ..... 45
4.1.7. Congestion and Flow Control ..... 49
4.1.8. Magic Packets ..... 50
4.1.9. MAC Local Loopback ..... 51
4.1.10. MAC Error Correction Code (ECC) ..... 52
4.1.11. MAC Reset ..... 52
4.1.12. PHY Management (MDIO) ..... 53
4.1.13. Connecting MAC to External PHYs ..... 55
4.2. 1000BASE-X/SGMII PCS With Optional Embedded PMA ..... 59
4.2.1. 1000BASE-X/SGMII PCS Architecture ..... 59
4.2.2. Transmit Operation ..... 61
4.2.3. Receive Operation ..... 61

4.2.4. Transmit and Receive Latencies....63
4.2.5. GMII Converter....65
4.2.6. SGMII Converter....65
4.2.7. Auto-Negotiation....66
4.2.8. Ten-bit Interface....69
4.2.9. PHY Loopback....70
4.2.10. PHY Power-Down....71
4.2.11. 1000BASE-X/SGMII PCS Reset....72
4.2.12. Intel FPGA IEEE 1588v2 Feature....73

**5. Configuration Register Space....81**
5.1. MAC Configuration Register Space....81
5.1.1. Base Configuration Registers (Dword Offset 0x00 – 0x17)....82
5.1.2. Statistics Counters (Dword Offset 0x18 – 0x38)....88
5.1.3. Transmit and Receive Command Registers (Dword Offset 0x3A – 0x3B)....91
5.1.4. Supplementary Address (Dword Offset 0xC0 – 0xC7)....91
5.1.5. IEEE 1588v2 Feature (Dword Offset 0xD0 – 0xD6)....92
5.1.6. IEEE 1588v2 Feature PMA Delay....93
5.2. PCS Configuration Register Space....93
5.2.1. Control Register (Word Offset 0x00)....95
5.2.2. Status Register (Word Offset 0x01)....96
5.2.3. Dev_Ability and Partner_Ability Registers (Word Offset 0x04 – 0x05)....97
5.2.4. An_Expansion Register (Word Offset 0x06)....99
5.2.5. If_Mode Register (Word Offset 0x14)....100
5.3. Register Initialization....100
5.3.1. Triple-Speed Ethernet System with MII/GMII or RGMII....101
5.3.2. Triple-Speed Ethernet System with SGMII....103
5.3.3. Triple-Speed Ethernet System with 1000BASE-X Interface....104

**6. Interface Signals....106**
6.1. Interface Signals....106
6.1.1. 10/100/1000 Ethernet MAC Signals....107
6.1.2. 10/100/1000 Multiport Ethernet MAC Signals....115
6.1.3. 10/100/1000 Ethernet MAC with 1000BASE-X/SGMII PCS Signals....118
6.1.4. 10/100/1000 Ethernet MAC with 1000BASE-X/SGMII 2XTBI PCS Signals....122
6.1.5. 10/100/1000 Multiport Ethernet MAC with 1000BASE-X/SGMII PCS Signals....124
6.1.6. 10/100/1000 Ethernet MAC with 1000BASE-X/SGMII PCS and Embedded PMA Signals....126
6.1.7. 10/100/1000 Multiport Ethernet MAC with 1000BASE-X/SGMII PCS and Embedded PMA....129
6.1.8. 1000BASE-X/SGMII PCS Signals....134
6.1.9. 1000BASE-X/SGMII 2XTBI PCS Signals....138
6.1.10. 1000BASE-X/SGMII PCS and PMA Signals....140
6.2. Timing....141
6.2.1. Avalon Streaming Receive Interface....141
6.2.2. Avalon Streaming Transmit Interface....143
6.2.3. GMII Transmit....144
6.2.4. GMII Receive....144
6.2.5. RGMII Transmit....144
6.2.6. RGMII Receive....145
6.2.7. MII Transmit....146
6.2.8. MII Receive....146

6.2.9. IEEE 1588v2 Timestamp ..... 146

**7. Design Considerations ..... 151**

7.1. Optimizing Clock Resources in Multiport MAC with PCS and Embedded PMA ..... 151
7.1.1. MAC and PCS With GX Transceivers ..... 152
7.1.2. MAC and PCS With LVDS Soft-CDR I/O ..... 154
7.2. Sharing PLLs in Devices with LVDS Soft-CDR I/O ..... 157
7.3. Sharing PLLs in Devices with GIGE PHY ..... 157
7.4. Sharing Transceiver Quads ..... 157
7.5. Migrating From Old to New User Interface For Existing Designs ..... 158
7.5.1. Exposed Ports in the New User Interface ..... 158
7.6. Clocking Scheme of MAC with 2XTBI PCS and Embedded PMA ..... 159

**8. Timing Constraints ..... 162**

8.1. Creating Clock Constraints ..... 162
8.2. Recommended Clock Frequency ..... 163

**9. Testbench ..... 166**

9.1. Triple-Speed Ethernet Testbench Architecture ..... 166
9.2. Testbench Components ..... 166
9.3. Testbench Verification ..... 167
9.4. Testbench Configuration ..... 168
9.5. Test Flow ..... 169
9.6. Simulation Model ..... 169
9.6.1. Generate the Simulation Model ..... 169
9.6.2. Simulate the IP Core ..... 170
9.6.3. Simulation Model Files ..... 171

**10. Software Programming Interface ..... 172**

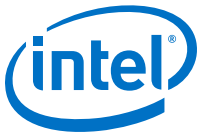
10.1. Driver Architecture ..... 172
10.2. Directory Structure ..... 172
10.3. PHY Definition ..... 173
10.4. Using Multiple SG-DMA Descriptors ..... 175
10.5. Using Jumbo Frames ..... 175
10.6. API Functions ..... 176
10.6.1. alt_tse_mac_get_common_speed() ..... 176
10.6.2. alt_tse_mac_set_common_speed() ..... 176
10.6.3. alt_tse_phy_add_profile() ..... 177
10.6.4. alt_tse_system_add_sys() ..... 177
10.6.5. triple_speed_ethernet_init() ..... 178
10.6.6. tse_mac_close() ..... 178
10.6.7. tse_mac_raw_send() ..... 179
10.6.8. tse_mac_setGMII mode() ..... 179
10.6.9. tse_mac_setMIImode() ..... 180
10.6.10. tse_mac_SwReset() ..... 180
10.7. Constants ..... 180

**11. Triple-Speed Ethernet Intel FPGA IP User Guide Archives ..... 185**

**12. Document Revision History for the Triple-Speed Ethernet Intel FPGA IP User Guide ..... 186**

**A. Ethernet Frame Format ..... 195**

A.1. Basic Frame Format ..... 195



A.2. VLAN and Stacked VLAN Frame Format..... 195
A.3. Pause Frame Format.....197
A.3.1. Pause Frame Generation..... 197

**B. Simulation Parameters..... 199**

B.1. Functionality Configuration Parameters..... 199
B.2. Test Configuration Parameters..... 200

# 1. About This IP Core

The Triple-Speed Ethernet Intel® FPGA IP core is a configurable intellectual property (IP) core that complies with the IEEE 802.3 standard.

It incorporates a 10/100/1000-Mbps Ethernet media access controller (MAC) and an optional 1000BASE-X/SGMII physical coding sublayer (PCS) with an embedded PMA built with either on-chip transceiver I/Os or LVDS I/Os. When offered in MAC-only mode, the IP core connects with an external PHY chip using MII, GMII, or RGMII interface. The IP core was tested and successfully validated by the University of New Hampshire (UNH) interoperability lab.

### Related Information

Triple-Speed Ethernet Intel FPGA IP User Guide Archives on page 185
Provides a list of user guides for previous versions of the Triple-Speed Ethernet Intel FPGA IP core.

## 1.1. Device Family Support

The IP core provides the following support for Intel FPGA device families.

**Table 1. Intel FPGA IP Core Device Support Levels**

| Device Support Level | Definition |
|---|---|
| Advance | The IP core is available for simulation and compilation for this device family. Timing models include initial engineering estimates of delays based on early post-layout information. The timing models are subject to change as silicon testing improves the correlation between the actual silicon and the timing models. You can use this IP core for system architecture and resource utilization studies, simulation, pinout, system latency assessments, basic timing assessments (pipeline budgeting), and I/O transfer strategy (datapath width, burst depth, I/O standards tradeoffs). |
| Preliminary | The IP core is verified with preliminary timing models for this device family. The IP core meets all functional requirements, but might still be undergoing timing analysis for the device family. It can be used in production designs with caution. |
| Final | The IP core is verified with final timing models for this device family. The IP core meets all functional and timing requirements for the device family and can be used in production designs. |

**Table 2. Device Family Support for Triple-Speed Ethernet MAC**

| Device Family | Support | Minimum Speed Grade with 1588 Feature |
|---|---|---|
| Intel Agilex™ | Preliminary | Not supported |
| Intel Stratix® 10 (E-tile) | Advance | -I3 (1) |
| Intel Stratix 10 (L-tile and H-tile) | Final | -I3 |

*continued...*

| Device Family | Support | Minimum Speed Grade with 1588 Feature |
|---|---|---|
| Stratix V | Final | -I3 |
| Stratix IV | Final | Not supported |
| Intel Arria® 10 | Final | -I3 |
| Arria V | Final | -I5 |
| Arria II | Final | Not supported |
| Intel Cyclone® 10 GX | Final | -I3 |
| Intel Cyclone 10 LP | Final | Not supported |
| Cyclone V | Final | -I7 |
| Cyclone IV | Final | Not supported |
| Intel MAX® 10 | Final | -I7 |

## 1.2. Features

- Complete triple-speed Ethernet IP: 10/100/1000-Mbps Ethernet MAC, 1000BASE-X/SGMII PCS, and embedded PMA.
- Successful validation from the University of New Hampshire (UNH) InterOperability Lab.
- 10/100/1000-Mbps Ethernet MAC features:
  - Multiple variations: 10/100/1000-Mbps Ethernet MAC in full duplex, 10/100-Mbps Ethernet MAC in half duplex, 10/100-Mbps or 1000-Mbps small MAC (resource-efficient variant), and multiport MAC that supports up to 24 ports.
  - Support for basic, VLAN, stacked VLAN, and jumbo Ethernet frames. Also supports control frames including pause frames.
  - Optional internal FIFO buffers, depth from 64 bytes to 256 Kbytes.
  - Optional statistics counters.
- MAC interfaces:
  - Client side—8-bit or 32-bit Avalon® Streaming
  - Network side—medium independent interface (MII), gigabit medium independent interface (GMII), or reduced gigabit medium independent interface (RGMII) on the network side. Optional loopback on these interfaces.
  - Optional management data I/O (MDIO) master interface for PHY device management.

(1) This feature is not supported if the E-tile transceiver variants are selected (for example, 10/100/1000-Mbps Ethernet MAC and 1000BASE-X/SGMII 2XTBI PCS with PMA variant, as well as 1000BASE-X/SGMII 2XTBI PCS variant).

- 1000BASE-X/SGMII PCS features:
  - Compliance with Clause 36 of the IEEE standard 802.3.
  - Optional embedded PMA implemented with serial transceiver or LVDS I/O and soft CDR in Intel FPGA devices that support this interface at 1.25-Gbps data rate.
  - Support for auto-negotiation as defined in Clause 37.
  - Support for connection to 1000BASE-X PHYs. Support for 10BASE-T, 100BASE-T, and 1000BASE-T PHYs if the PHYs support SGMII.
- PCS interfaces:
  - Client side—MII or GMII
  - Network side—ten-bit interface (TBI) for PCS without PMA; 1.25-Gbps serial interface for PCS with PMA implemented with serial transceiver or LVDS I/O and soft CDR in Intel FPGA devices that support this interface at 1.25-Gbps data rate.
- Programmable features via 32-bit configuration registers:
  - FIFO buffer thresholds.
  - Pause quanta for flow control.
  - Source and destination MAC addresses.
  - Address filtering on receive, up to 5 unicast and 64 multicast MAC addresses.
  - Promiscuous mode—receive frame filtering is disabled in this mode.
  - Frame length—in MAC only variation, up to 64 Kbytes including jumbo frames. In all variants containing 1000BASE-X/SGMII PCS (with or without MAC), the frame length is up to 10 Kbytes.
  - Optional auto-negotiation for the 1000BASE-X/SGMII PCS.
- Error correction code protection feature for internal memory blocks.
- Optional IEEE 1588v2 feature for 10/100/1000-Mbps Ethernet MAC with SGMII PCS and embedded serial PMA variation operating without internal FIFO buffer in full-duplex mode, 10/100/1000-Mbps MAC with SGMII PCS and embedded LVDS I/O, or MAC only variation operating without internal FIFO buffer in full-duplex mode. This feature is supported in Intel Stratix 10, Arria V, Intel Arria 10, Intel Cyclone 10 GX, Cyclone V, Intel MAX 10, and Stratix V device families. This feature is not supported in Intel Stratix 10 E-Tile transceiver variants (10/100/1000-Mbps Ethernet MAC and 1000BASE-X/SGMII 2XTBI PCS with PMA variant, as well as 1000BASE-X/SGMII 2XTBI PCS variant).

## 1.3. 10/100/1000 Ethernet MAC Versus Small MAC

**Table 3. Feature Comparison between 10/100/1000 Ethernet MAC and Small MAC**

| Feature | 10/100/1000 Ethernet MAC | Small MAC |
|---|---|---|
| Speed | Triple speed (10/100/1000 Mbps) | 10/100 Mbps or 1000 Mbps |
| External interfaces | MII/GMII or RGMII | MII only for 10/100 Mbps small MAC, GMII or RGMII for 1000 Mbps small MAC |
| Control interface registers | Fully programmable | Limited programmable options. The following options are fixed: |

*continued...*

| Feature | 10/100/1000 Ethernet MAC | Small MAC |
|---|---|---|
| | | • Maximum frame length is fixed to 1518. Jumbo frames are not supported.<br>• FIFO buffer thresholds are set to fixed values.<br>• Store and forward option is not available.<br>• Interpacket gap is set to 12.<br>• Flow control is not supported; pause quanta is not in use.<br>• Checking of payload length is disabled.<br>• Supplementary MAC addresses are disabled.<br>• Padding removal is disabled.<br>• Sleep mode and magic packet detection is not supported. |
| Synthesis options | Fully configurable | Limited configurable options. The following options are NOT available:<br>• Flow control<br>• VLAN<br>• Statistics counters<br>• Multicast hash table<br>• Loopback<br>• TBI and 1.25 Gbps serial interface<br>• 8-bit wide FIFO buffers |

## 1.4. High-Level Block Diagrams

High-level block diagrams of different variations of the Triple-Speed Ethernet Intel FPGA IP core.

**Figure 1. 10/100/1000-Mbps Ethernet MAC**

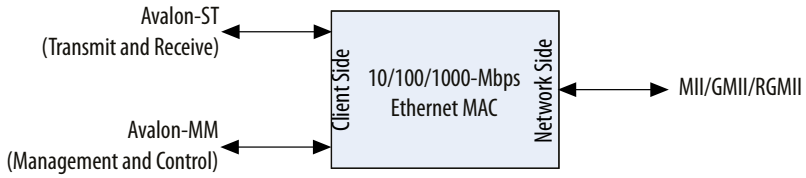



**Figure 2. Multi-port MAC**

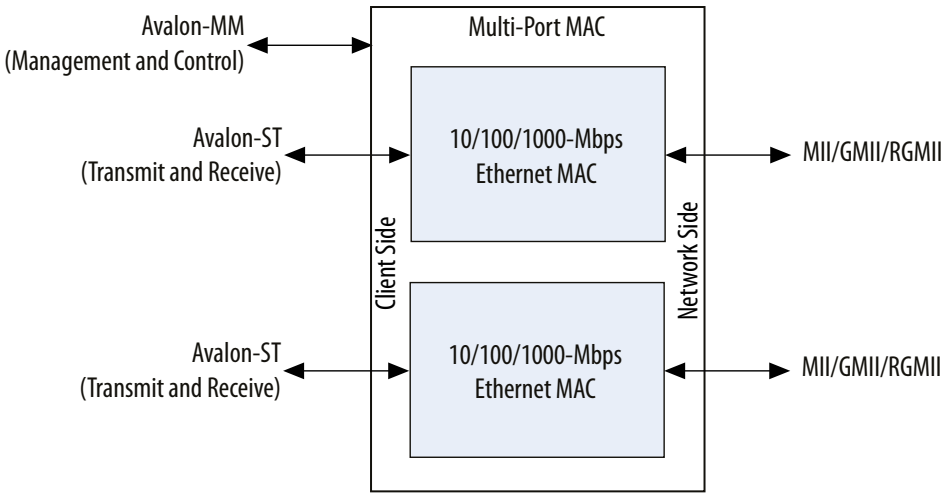


**Figure 3. 10/100/1000-Mbps Ethernet MAC and 1000BASE-X/SGMII PCS with Optional PMA**

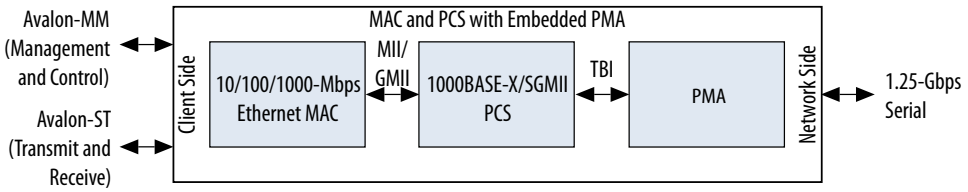



**Figure 4. 10/100/1000-Mbps Ethernet MAC and 1000BASE-X/SGMII 2XTBI PCS with PMA**

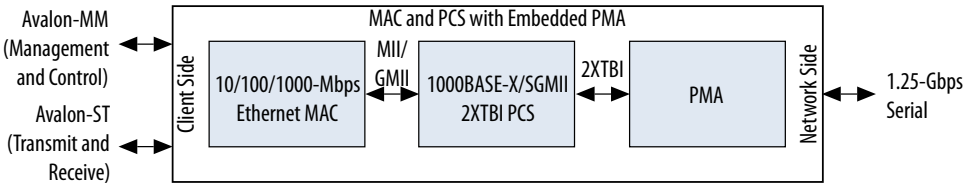



**Figure 5. 1000BASE-X/SGMII PCS with Optional PMA**

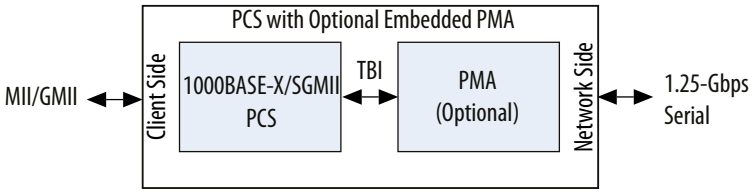



**Figure 6. 1000BASE-X/SGMII 2XTBI PCS**

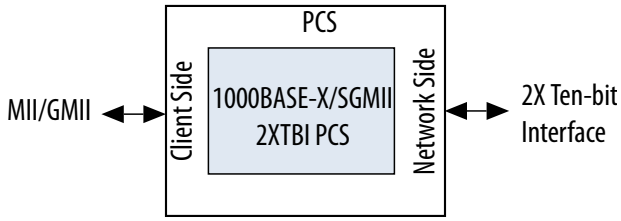
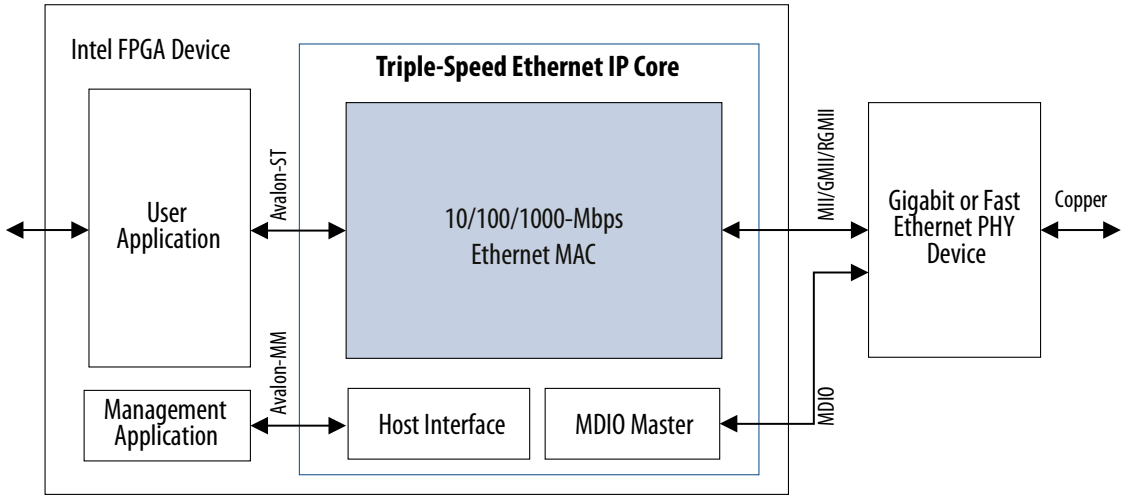


**Figure 7. Stand-Alone 10/100/1000-Mbps Ethernet MAC**



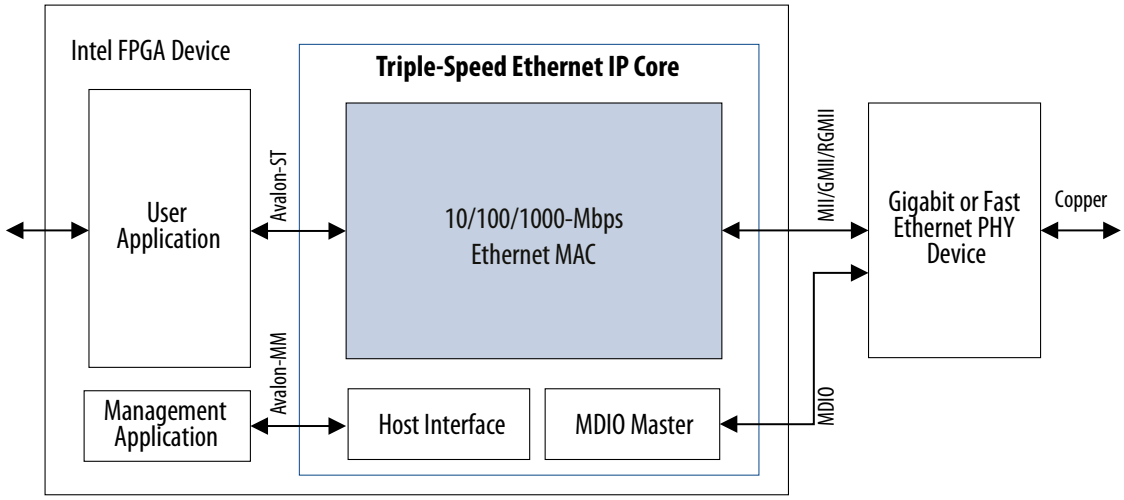

## 1.5. Example Applications

This section shows example applications of different variations of the Triple-Speed Ethernet Intel FPGA IP core.

The 10/100/1000-Gbps Ethernet MAC only variation can serve as a bridge between the user application and standard fast or gigabit Ethernet PHY devices.

**Figure 8. Stand-Alone 10/100/1000 Mbps Ethernet MAC**

Example application using this variation for a copper network.




When configured to include the 1000BASE-X/SGMII PCS function, the IP core can seamlessly connect to any industry standard gigabit Ethernet PHY device via a TBI. Alternatively, when the 1000BASE-X/SGMII PCS function is configured to include an embedded PMA, the IP core can connect directly to a gigabit interface converter (GBIC), small form-factor pluggable (SFP) module, or an SGMII PHY.

**Figure 9. 10/100/1000 Mbps Ethernet MAC and 1000BASE-X PCS with Embedded PMA**

Example application using the Triple-Speed Ethernet Intel FPGA IP core with 1000BASE-X and PMA. The PMA block connects to an off-the-shelf GBIC or SFP module to communicate directly over the optical link.

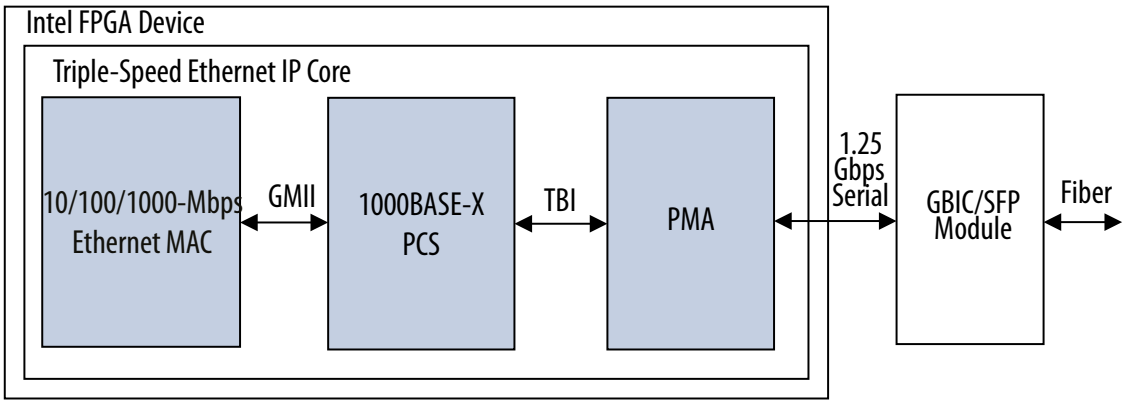
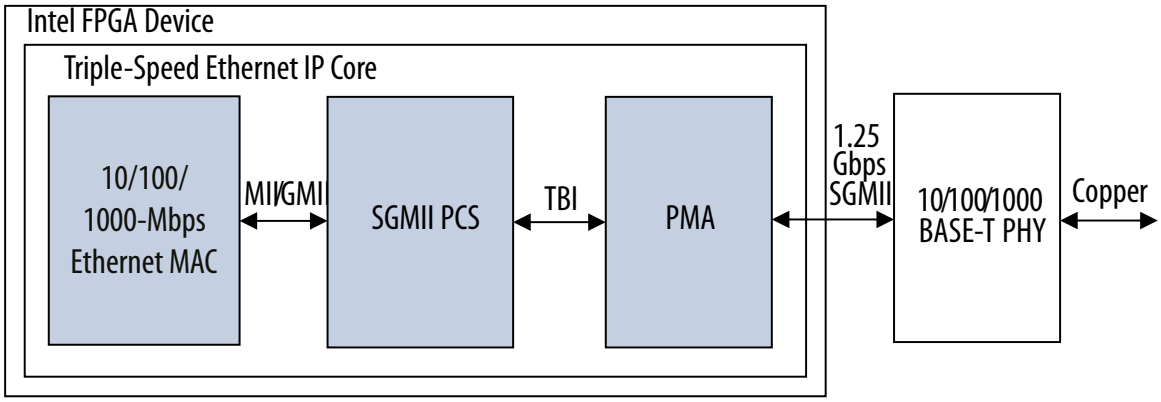



**Figure 10. 10/100/1000 Mbps Ethernet MAC and SGMII PCS with Embedded PMA—GMII/MII to 1.25-Gbps Serial Bridge Mode**

Example application using the Triple-Speed Ethernet Intel FPGA IP core with 1000BASE-X and PMA, 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).




## 1.6. IP Core Verification

For each release, Intel verifies the Triple-Speed Ethernet Intel FPGA IP core through extensive simulation and internal hardware verification in various Intel FPGA device families. The University of New Hampshire (UNH) InterOperability Lab also successfully verified the IP core prior to its release.

Intel used a highly parameterizeable transaction-based testbench to test the following aspects of the IP core:

- 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

Intel has also validated the Triple-Speed Ethernet Intel FPGA IP core in both optical and copper platforms using the following development kits:

- Nios® II Development Kit, Cyclone II Edition (2C35)
- Intel Arria 10 FPGA Development Kit
- Intel Cyclone 10 LP FPGA Development Kit
- Stratix III FPGA Development Kit
- Stratix IV FPGA Development Kit
- Stratix V FPGA Development Kit
- Intel Stratix 10 FPGA Development Kit
- Quad 10/100/1000 Marvell PHY
- MorethanIP 10/100 and 10/100/1000 Ethernet PHY Daughtercards

### 1.6.1. 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.

### 1.6.2. Copper Platform

In the copper platform, Intel FPGA tested the Triple-Speed Ethernet Intel FPGA IP core with an external 1000BASE-T PHY devices. The IP core 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.

## 1.7. Performance and Resource Utilization

The estimated resource utilization and performance of the Triple-Speed Ethernet Intel FPGA IP core are obtained by compiling the Triple-Speed Ethernet Intel FPGA IP core using the Intel Quartus® Prime software targeting a given device. The $f_{MAX}$ of all configurations is more than 125 MHz.

## Table 4. Resource Utilization for Triple-Speed Ethernet for Intel Agilex Devices

The following estimates are obtained by targeting the Intel Agilex (AGFB014R24A3E3VR0) device.

| IP Core Variation | Settings | FIFO Buffer Size (Bits) | Combinational ALUTs | Logic Registers | Memory (M20K) |
|---|---|---|---|---|---|
| 10/100/1000-Mbps Ethernet MAC | MII/GMII.<br>All MAC options enabled.<br>Full- and half-duplex. | 2048x32 | 4051 | 5634 | 21 |
| | | 2048x8 | 3865 | 5442 | 16 |
| 10/100-Mbps Small MAC | MII.<br>Full- and half-duplex only. | 2048x32 | 1445 | 2120 | 11 |
| 1000-Mbps Small MAC | GMII.<br>Full-duplex only. | 2048x32 | 1178 | 1937 | 10 |
| 1000BASE-X/ SGMII PCS | SGMII bridge enabled. | N/A | 898 | 1448 | 0 |
| | 1000BASE-X.<br>SGMII bridge enabled.<br>PMA block (LVDS_IO). | N/A | 967 | 1638 | 1 |
| 1000BASE-X/ SGMII 2XTBI PCS only | SGMII bridge enabled. | N/A | 1329 | 2003 | 2 |
| | 1000BASE-X. | N/A | 1267 | 1917 | 2 |
| 10/100/1000 Mb Ethernet MAC with 1000BASEX/ SGMII 2XTBI PCS | All MAC options enabled.<br>SGMII bridge enabled.<br>PMA block (GXB). | 2048x32 | 5415 | 7882 | 22 |

## Table 5. Resource Utilization for Triple-Speed Ethernet in Intel Stratix 10 Devices

The following estimates are obtained by targeting the Intel Stratix 10 (1SG280HN3F43E3VG) device with speed grade -3.

| IP Core Variation | Settings | FIFO Buffer Size (Bits) | Combinational ALUTs | Logic Registers | Memory (M20K) |
|---|---|---|---|---|---|
| 10/100/1000-Mbps Ethernet MAC | MII/GMII.<br>All MAC options enabled.<br>Full- and half-duplex. | 2048x32 | 4055 | 5398 | 21 |
| | | 2048x8 | 3831 | 5139 | 16 |
| 10/100-Mbps Small MAC | MII.<br>Full- and half-duplex only. | 2048x32 | 1460 | 2078 | 11 |
| 1000-Mbps Small MAC | GMII.<br>Full-duplex only. | 2048x32 | 1158 | 1838 | 10 |
| 1000BASE-X/ SGMII PCS | SGMII bridge enabled. | N/A | 901 | 1333 | 0 |
| | 1000BASE-X.<br>SGMII bridge enabled.<br>PMA block (LVDS_IO). | N/A | 945 | 1500 | 0 |

## Table 6. Resource Utilization for Triple-Speed Ethernet with E-Tile Transceiver in Intel Stratix 10 Devices

The following estimates are obtained by targeting the Intel Stratix 10 (1ST280EY3F55E3VG) device with speed grade -3.

| IP Core Variation | Settings | FIFO Buffer Size (Bits) | Combinational ALUTs | Logic Registers | Memory (M20K) |
|---|---|---|---|---|---|
| 1000BASE-X/ SGMII 2XTBI PCS only | SGMII bridge enabled. | N/A | 1306 | 1609 | 2 |
| | 1000BASE-X | N/A | 1246 | 1543 | 2 |
| 10/100/1000 Mb Ethernet MAC with 1000BASE-X/SGMII 2XTBI PCS | All MAC options enabled.<br>SGMII bridge enabled.<br>PMA block (GXB). | 2048x32 | 5292 | 7244 | 22 |

## Table 7. Resource Utilization for Triple-Speed Ethernet for Intel Arria 10 Devices

The following estimates are obtained by targeting the Intel Arria 10 GX (10AX115R4F40I3SG) device with speed grade -3.

| IP Core Variation | Settings | FIFO Buffer Size (Bits) | Combinational ALUTs | Logic Registers | Memory (M20K) |
|---|---|---|---|---|---|
| 10/100/1000-Mbps Ethernet MAC | MII/GMII.<br>All MAC options enabled.<br>Full- and half-duplex. | 2048x32 | 3817 | 5316 | 22 |
| | | 2048x8 | 3646 | 5142 | 17 |
| 10/100-Mbps Small MAC | MII.<br>Full- and half-duplex. | 2048x32 | 1354 | 1986 | 12 |
| 1000-Mbps Small MAC | GMII.<br>Full-duplex only. | 2048x32 | 1079 | 1769 | 10 |
| 1000BASE-X/ SGMII PCS | SGMII bridge enabled. | N/A | 841 | 1162 | 2 |
| | 1000BASE-X.<br>SGMII bridge enabled.<br>PMA block (GXB). | N/A | 662 | 995 | 2 |
| | 1000BASE-X. | N/A | 641 | 800 | 0 |

## Table 8. Resource Utilization for Triple-Speed Ethernet for Intel Cyclone 10 GX Devices

The following estimates are obtained by targeting the Intel Cyclone 10 GX (10CX220YU484I6G) device with speed grade -3.

| IP Core Variation | Settings | FIFO Buffer Size (Bits) | Combinational ALUTs | Logic Registers | Memory (M20K) |
|---|---|---|---|---|---|
| 10/100/1000-Mbps Ethernet MAC | MII/GMII.<br>All MAC options enabled.<br>Full- and half-duplex. | 2048x32 | 3819 | 5346 | 22 |
| | | 2048x8 | 3646 | 5185 | 17 |
| 10/100-Mbps Small MAC | MII.<br>Full- and half-duplex. | 2048x32 | 1358 | 1992 | 12 |
| 1000-Mbps Small MAC | GMII.<br>Full-duplex only. | 2048x32 | 1083 | 1761 | 10 |
| 1000BASE-X/ SGMII PCS | SGMII bridge enabled. | N/A | 841 | 1175 | 2 |
| | 1000BASE-X. | N/A | 662 | 992 | 2 |

***continued...***

| IP Core Variation | Settings | FIFO Buffer Size (Bits) | Combinational ALUTs | Logic Registers | Memory (M20K) |
|---|---|---|---|---|---|
| | SGMII bridge enabled.<br>PMA block (GXB). | | | | |
| | 1000BASE-X.<br>SGMII bridge enabled.<br>PMA block (LVDS_IO). | N/A | 877 | 1285 | 2 |

**Table 9. Resource Utilization for Triple-Speed Ethernet for Intel Cyclone 10 LP Devices**

The following estimates are obtained by targeting the Intel Cyclone 10 LP (10CL120YF780I7G) device.

| IP Core Variation | Settings | FIFO Buffer Size (Bits) | Combinational ALUTs | Logic Registers | Memory (M9K) |
|---|---|---|---|---|---|
| 10/100/1000-Mbps Ethernet MAC | MII/GMII.<br>All MAC options enabled.<br>Full- and half-duplex. | 2048x8 | 6724 | 4840 | 17 |
| | | N/A | 5863 | 4204 | 8 |
| 1000BASE-X/SGMII PCS | 1000BASE-X.<br>SGMII bridge enabled. | N/A | 1628 | 1133 | 2 |
| 10/100-Mbps Small MAC | MII.<br>Full and half-duplex only. | N/A | 2416 | 1933 | 24 |
| 1000-Mbps Small MAC | GMII.<br>Full-duplex only. | N/A | 1998 | 1645 | 24 |

**Table 10. Resource Utilization for Triple-Speed Ethernet for Intel MAX 10 Devices**

The following estimates are obtained by targeting the Intel MAX 10 (10M08DAF484C8G) device with speed grade -8.

| IP Core Variation | Settings | FIFO Buffer Size (Bits) | Logic Elements | Logic Registers | Memory (M9K) |
|---|---|---|---|---|---|
| 10/100/1000-Mbps Ethernet MAC | MII/GMII.<br>Full- and half-duplex. | 2048x32 | 6806 | 4943 | 30 |
| | | 2048x8 | 6593 | 4767 | 17 |
| 10/100-Mbps Small MAC | MII.<br>Full- and half-duplex. | 2048x32 | 2650 | 2117 | 24 |
| 1000-Mbps Small MAC | RGMII<br>Full-duplex only. | 2048x32 | 2286 | 1862 | 24 |

**Table 11. Resource Utilization for Triple-Speed Ethernet for Stratix V Devices**

The following estimates are obtained by targeting the Stratix V GX (5SGXMA7N3F45C3) device with speed grade -3.

| IP Core Variation | Settings | FIFO Buffer Size (Bits) | Combinational ALUTs | Logic Registers | Memory (M20K Blocks/ MLAB Bits) |
|---|---|---|---|---|---|
| 10/100-Mbps Small MAC | MII.<br>Full- and half-duplex. | 2048x32 | 1261 | 2018 | 11/0 |
| | MII.<br>All MAC options enabled. | 2048x32 | 1261 | 2018 | 11/0 |
| 1000-Mbps Small MAC | GMII.<br>All MAC options enabled. | 2048x32 | 1227 | 1959 | 10/128 |

*continued...*

| IP Core Variation | Settings | FIFO Buffer Size (Bits) | Combinational ALUTs | Logic Registers | Memory (M20K Blocks/ MLAB Bits) |
|---|---|---|---|---|---|
| | RGMII.<br>All MAC options enabled. | 2048x32 | 1237 | 1984 | 10/128 |
| 10/100/1000-Mbps Ethernet MAC | MII/GMII.<br>Full- and half-duplex. | N/A | 3137 | 4298 | 5/2048 |
| | | 2048x8 | 3627 | 4971 | 10/2048 |
| | | 2048x32 | 3777 | 5145 | 16/2048 |
| | MII/GMII.<br>All MAC options enabled. | 2048x32 | 3454 | 4928 | 16/768 |
| | RGMII.<br>All MAC options enabled. | 2048x32 | 3466 | 4933 | 16/768 |
| 1000BASE-X/ SGMII PCS | 1000BASE-X. | N/A | 614 | 786 | 0/0 |
| | 1000BASE-X.<br>SGMII bridge enabled. | N/A | 839 | 1160 | 0/480 |
| | 1000BASE-X.<br>SGMII bridge enabled.<br>PMA block (LVDS_IO). | N/A | 857 | 1250 | 0/480 |
| | 1000BASE-X.<br>SGMII bridge enabled.<br>PMA block (GXB). | N/A | 2203 | 1991 | 5/2208 |
| | 1000BASE-X.<br>SGMII bridge enabled.<br>PMA block (GXB).<br>The reconfig controller is compiled with this variation. | N/A | 1441 | 903 | 4/2048 |
| 10/100/1000-Mbps Ethernet MAC and 1000BASE-X/ SGMII PCS | All MAC options enabled.<br>SGMII bridge enabled. | 2048×32 | 4306 | 6132 | 16/1248 |
| | Default MAC options.<br>SGMII bridge enabled.<br>IEEE 1588v2 feature enabled. | 0 | 5062 | 5318 | 4/1536 |

## Table 12. Resource Utilization for Triple-Speed Ethernet for Cyclone V Devices

The following estimates are obtained by targeting the Cyclone V GX (5CGXFC7C7F23C8) device with speed grade -8.

| IP Core Variation | Settings | FIFO Buffer Size (Bits) | Logic Elements | Logic Registers | Memory (M10K) |
|---|---|---|---|---|---|
| 10/100/1000-Mbps Ethernet MAC | MII/GMII.<br>Full- and half-duplex. | 2048x32 | 3644 | 5340 | 27 |
| | | 2048x8 | 3489 | 5185 | 15 |
| 10/100-Mbps Small MAC | MII.<br>Full- and half-duplex. | 2048x32 | 1539 | 2295 | 21 |
| 1000-Mbps Small MAC | RGMII.<br>Full-duplex only. | 2048x32 | 1265 | 2060 | 20 |
| 1000BASE-X/ SGMII PCS | SGMII bridge enabled | N/A | 844 | 1241 | 2 |
| | 1000BASE-X. | N/A | 656 | 947 | 9 |

***continued...***

| IP Core Variation | Settings | FIFO Buffer Size (Bits) | Logic Elements | Logic Registers | Memory (M10K) |
|---|---|---|---|---|---|
| | SGMII bridge enabled.<br>PMA block (GXB). | | | | |
| | 1000BASE-X.<br>SGMII bridge enabled. | N/A | 836 | 1097 | 9 |

## Table 13. Resource Utilization for Triple-Speed Ethernet for Stratix IV Devices

The following estimates are obtained by targeting the Stratix IV GX (EP4SGX530NF45C4) device with speed grade -4.

| IP Core Variation | Settings | FIFO Buffer Size (Bits) | Combinational ALUTs | Logic Registers | Memory (M9K Blocks/ M144K Blocks/MLAB Bits) |
|---|---|---|---|---|---|
| 10/100-Mbps Small MAC | MII.<br>Full- and half-duplex. | 2048x32 | 1410 | 2127 | 12/1/1408 |
| | MII.<br>All MAC options enabled. | 2048x32 | 1157 | 1894 | 12/1/128 |
| 1000-Mbps Small MAC | GMII.<br>All MAC options enabled. | 2048x32 | 1160 | 1827 | 12/1/176 |
| | RGMII.<br>All MAC options enabled. | 2048x32 | 1170 | 1861 | 12/1/176 |
| 10/100/1000-Mbps Ethernet MAC | MII/GMII.<br>Full- and half-duplex. | N/A | 2721 | 3395 | 0/0/3364 |
| | | 2048x8 | 3201 | 3977 | 8/0/3620 |
| | | 2048x32 | 3345 | 4425 | 12/1/3364 |
| | MII/GMII.<br>All MAC options enabled. | 2048x32 | 3125 | 3994 | 12/1/2084 |
| | RGMII.<br>All MAC options enabled. | 2048x32 | 3133 | 4021 | 12/1/2084 |
| 1000BASE-X/ SGMII PCS | 1000BASE-X. | N/A | 624 | 661 | 0/0/0 |
| | 1000BASE-X.<br>SGMII bridge enabled. | N/A | 808 | 986 | 2/0/0 |
| | 1000BASE-X.<br>SGMII bridge enabled.<br>PMA block (LVDS_IO). | N/A | 819 | 1057 | 2/0/0 |
| | 1000BASE-X.<br>SGMII bridge enabled.<br>PMA block (GXB). | N/A | 1189 | 1212 | 1/0/160 |
| 10/100/1000-Mbps Ethernet MAC and 1000BASE-X/ SGMII PCS | All MAC options enabled.<br>SGMII bridge enabled. | 2048×32 | 3971 | 4950 | 14/1/2084 |

### Table 14. Resource Utilization for Triple-Speed Ethernet for Cyclone IV GX Devices

The following estimates are obtained by targeting the Cyclone IV GX (EP4CGX150DF27C7) device with speed grade -7.

| IP Core Variation | Settings | FIFO Buffer Size (Bits) | Logic Elements | Logic Registers | Memory (M9K Blocks/ M144K Blocks/ MLAB Bits) |
|---|---|---|---|---|---|
| 1000-Mbps Small MAC | RGMII<br>Full-duplex only. | 2048x32 | 2161 | 1699 | 24/0/0 |
| 10/100/1000-Mbps Ethernet MAC | MII/GMII<br>Full- and half-duplex. | 2048x32 | 5614 | 3666 | 31/0/0 |
| 1000BASE-X/ SGMII PCS | 1000BASE-X. | N/A | 1149 | 661 | 0/0/0 |
| | 1000BASE-X.<br>SGMII bridge enabled.<br>PMA block (GXB). | N/A | 2001 | 1127 | 2/0/0 |

### Table 15. Resource Utilization for Triple-Speed Ethernet for Arria II GX Devices

The following estimates are obtained by targeting the Arria II GX (EP2AGX260EF29I3) device with speed grade -3.

| IP Core Variation | Settings | FIFO Buffer Size (Bits) | Combinational ALUTs | Logic Registers | Memory (M9K Blocks/ M144K Blocks/MLAB Bits) |
|---|---|---|---|---|---|
| 10/100/1000-Mbps Ethernet MAC | RGMII.<br>All MAC options enabled.<br>Full- and half-duplex. | 2048x32 | 3357 | 3947 | 26/0/1828 |
| 1000BASE-X/ SGMII PCS | 1000BASE-X. | N/A | 624 | 661 | 0/0/0 |
| | 1000BASE-X.<br>SGMII bridge enabled.<br>PMA block (GXB). | N/A | 1191 | 1214 | 1/0/160 |

## 1.8. Release Information

IP versions are the same as the Intel Quartus Prime Design Suite software versions up to v19.1. From Intel Quartus Prime Design Suite software version 19.2 or later, IP cores have a new IP versioning scheme.

The IP versioning scheme (X.Y.Z) number changes from one software version to another. A change in:

- X indicates a major revision of the IP. If you update your Intel Quartus Prime software, you must regenerate the IP.
- Y indicates the IP includes new features. Regenerate your IP to include these new features.
- Z indicates the IP includes minor changes. Regenerate your IP to include these changes.

**Table 16. Triple-Speed Ethernet Intel FPGA IP Core Release Information**

| Item | Description |
|---|---|
| IP Version | 19.4.0 |
| Intel Quartus Prime Version | 19.4 |
| Release Date | 2019.12.16 |
| Ordering Code | Triple-Speed Ethernet: IP-TRIETHERNET<br>IEEE 1588v2 for Triple-Speed Ethernet: IP-TRIETHERNETF |
| Product ID(s) | Triple-Speed Ethernet: 00BD<br>IEEE 1588v2 for Triple-Speed Ethernet: 0104 |
| Vendor ID(s) | 6AF7 |

Intel verifies that the current version of the Intel Quartus Prime software compiles the previous version of each IP core. The *Intel FPGA IP Release Notes* report any exceptions to this verification. Intel does not verify compilation with IP core versions older than one release.

### Related Information

- Intel FPGA IP Release Notes
- Triple-Speed Ethernet Design Examples
  Available design examples in Intel FPGA Design Store.
- AN-744: Scalable Triple-Speed Ethernet Reference Designs for Arria 10 Device
- AN-830: Intel FPGA Triple-Speed Ethernet and On-Board PHY Chip Reference Design for Intel Stratix 10 Devices

# 2. Getting Started with Intel FPGA IP Cores

Intel and strategic IP partners offer a broad portfolio of off-the-shelf, configurable IP cores optimized for Intel FPGA devices. The Intel Quartus Prime software installation includes the Intel FPGA IP library.

For more information on how to install and use Intel FPGA IP Cores, refer to the *Introduction to Intel FPGA IP Cores*.

**Related Information**

Introduction to Intel FPGA IP Cores

## 2.1. Design Walkthrough

This walkthrough explains how to create a Triple-Speed Ethernet IP core design using Platform Designer in the Intel Quartus Prime software. After you generate a custom variation of the Triple-Speed Ethernet IP core, you can incorporate it into your overall project.

This walkthrough includes the following steps:

1. Creating a New Intel Quartus Prime Project on page 21
2. Generating a Design Example or Simulation Model on page 22
3. Simulate the System on page 22
4. Compiling the Triple-Speed Ethernet IP Core Design on page 23
5. Programming an FPGA Device on page 23

### 2.1.1. Creating a New Intel Quartus Prime Project

You need to create a new Intel Quartus Prime project with the **New Project Wizard**, which specifies the working directory for the project, assigns the project name, and designates the name of the top-level design entity.

To create a new project, follow these steps:

1. Launch the Intel Quartus Prime software on your PC. Alternatively, you can use the Intel Quartus Prime Lite Edition software.
2. On the **File** menu, click **New Project Wizard**.
3. In the **New Project Wizard: Directory, Name, Top-Level Entity** page, specify the working directory, project name, and top-level design entity name. Click **Next**.
4. In the **New Project Wizard: Add Files** page, select the existing design files (if any) you want to include in the project.(2) Click **Next**.

5. In the **New Project Wizard: Family & Device Settings** page, select the device family and specific device you want to target for compilation. Click **Next**.
6. In the **EDA Tool Settings** page, select the EDA tools you want to use with the Intel Quartus Prime software to develop your project.
7. The last page in the **New Project Wizard** window shows the summary of your chosen settings. Click **Finish** to complete the Intel Quartus Prime project creation.

## 2.1.2. Generating a Design Example or Simulation Model

After you have parameterized the IP core, you can also generate a design example, in addition to generating the IP core component files.

In the parameter editor, click **Example Design** to create a functional simulation model (design example that includes a testbench). The testbench and the automated script are located in the <*variation name*>**_testbench** directory.

*Note:* Generating a design example can increase processing time.

*Note:* **Generate Example Design** option only generates the design for functional simulation.

You can now integrate your custom IP core instance in your design, simulate, and compile. While integrating your IP core instance into your design, you must make appropriate pin assignments. You can create a virtual pin to avoid making specific pin assignments for top-level signals while you are simulating and not ready to map the design to hardware.

### Related Information

- Testbench
  More information about the IP core simulation model.
- Intel Quartus Prime Help
  More information about the Intel Quartus Prime software, including virtual pins.

## 2.1.3. Simulate the System

During system generation, Platform Designer generates a functional simulation model —or design example that includes a testbench—which you can use to simulate your system in any Intel FPGA-supported simulation tool.

### Related Information

- Intel Quartus Prime Pro Edition Software and Device Support Release Notes
  More information about the latest Intel FPGA-supported simulation tools.
- Intel Quartus Prime Standard Edition Software and Device Support Release Notes
  More information about the latest Intel FPGA-supported simulation tools.
- Simulating Intel FPGA Designs
  More information in the Intel Quartus Prime Pro Edition User Guide: Third-party Simulation about simulating Intel FPGA IP cores.

---

(2) To include existing files, you must specify the directory path to where you installed the IP core. You must also add the user libraries if you installed the IP Library in a different directory from where you installed the Intel Quartus Prime software.

- Simulating Intel FPGA Designs
  More information in the Intel Quartus Prime Standard Edition User Guide: Third-party Simulation about simulating Intel FPGA IP cores.
- System Design with Platform Designer
  More information in the Intel Quartus Prime Pro Edition User Guide: Platform Designer about simulating Platform Designer systems.
- System Design with Platform Designer (Standard)
  More information in the Intel Quartus Prime Standard Edition User Guide: Platform Designer about simulating Platform Designer (Standard) systems.

### 2.1.4. Compiling the Triple-Speed Ethernet IP Core Design

Refer to Design Considerations on page 151 chapter before compiling the Triple-Speed Ethernet IP core design.

To compile your design, click **Start Compilation** on the Processing menu in the Intel Quartus Prime software. You can use the generated **.qip** file to include relevant files into your project.

#### Related Information

Intel Quartus Prime Help
More information about compilation in Intel Quartus Prime software.

### 2.1.5. Programming an FPGA Device

After successfully compiling your design, program the targeted Intel FPGA device with the Intel Quartus Prime Programmer and verify the design in hardware. For instructions on programming the FPGA device, refer to the *Device Programming* section in volume 3 of the Intel Quartus Prime Handbook.

#### Related Information

Device Programming

## 2.2. Generated Files

The type of files generated in your project directory and their names may vary depending on the custom variation of the IP core you created.

**Table 17. Generated Files**

| File Name | Description |
|---|---|
| <*variation_name*>.v or <*variation_name*>.vhd | A IP core variation file, which defines a VHDL or Verilog HDL top-level description of the custom IP core. Instantiate the entity defined by this file inside your design. Include this file when compiling your design in the Intel Quartus Prime software. |
| <*variation_name*>.bsf | Intel Quartus Prime symbol file for the IP core variation. You can use this file in the Intel Quartus Prime block diagram editor. |
| <*variation_name*>.qip and <*variation_name*>**.sip** | Contains Intel Quartus Prime project information for your IP core variations. |
| <*variation_name*>.cmp | A VHDL component declaration file for the IP core variation. Add the contents of this file to any VHDL architecture that instantiates the IP core. |

*continued...*

| File Name | Description |
|---|---|
| <*variation_name*>.spd | Simulation Package Descriptor file. Specifies the files required for simulation. |
| Testbench Files (in <variation_name>**_testbench** folder) | |
| README.txt | Read me file for the testbench design. |
| **generate_sim.qpf** and generate_sim.qsf | Dummy Intel Quartus Prime project and project setting file. Use this to start the Intel Quartus Prime in the correct directory to launch the generate_sim_verilog.tcl and generate_sim_vhdl.tcl files. |
| **generate_sim_verilog.tcl** and generate_sim_vhdl.tcl | A Tcl script to generate the DUT VHDL or Verilog HDL simulation model for use in the testbench. |
| **/testbench_vhdl/**<variation_name>**/**<variation_name>**_tb.vhd** or **/testbench_verilog/**<variation_name>**/**<variation_name>**_tb.v** | VHDL or Verilog HDL testbench that exercises your IP core variation in a third party simulator. |
| **/testbench_vhdl/**<variation_name>**/run_**<variation_name>**_tb.tcl** or **/testbench_verilog/**<variation_name>**/run_**<variation_name>**_tb.tcl** | A Tcl script for use with the ModelSim simulation software. |
| **/testbench_vhdl/**<variation_name>**/**<variation_name>**_wave.do** or **/testbench_verilog/**<variation_name>**/**<variation_name>**_wave.do** | A signal tracing macro script used with the ModelSim simulation software to display testbench signals. |
| **/testbench_vhdl/models** or **/testbench_verilog/models** | A directory containing VHDL and Verilog HDL models of the Ethernet generators and monitors used by the generated testbench. |

## 2.2.1. Design Constraint File No Longer Generated

For a new Triple-Speed Ethernet IP core created using the Intel Quartus Prime software version 13.0 or later, the software no longer generate the <*variation_name*>_constraints.tcl file that contains the necessary constraints for the compilation of your IP core variation.

The following table lists the recommended Quartus pin assignments that you can set in your design.

**Table 18. Recommended Quartus Pin Assignments**

| Pin Assignment | Assignment Value | Description | Design Pin |
|---|---|---|---|
| FAST_INPUT_REGISTER | ON | To optimize I/O timing for MII, GMII and TBI interface. | MII, GMII, RGMII, TBI input pins. |
| FAST_OUTPUT_REGISTER | ON | To optimize I/O timing for MII, GMII and TBI interface. | MII, GMII, RGMII, TBI output pins. |
| IO_STANDARD | 1.4-V PCML or 1.5-V PCML | I/O standard for GXB serial input and output pins. | GXB transceiver serial input and output pins. |
| IO_STANDARD | LVDS | I/O standard for LVDS/IO serial input and output pins. | LVDS/IO transceiver serial input and output pins. |

*continued...*

| Pin Assignment | Assignment Value | Description | Design Pin |
|---|---|---|---|
| GLOBAL_SIGNAL | Global clock | To assign clock signals to use the global clock network. Use this setting to guide the Intel Quartus Prime software in the fitter process for better timing closure. | • `clk` and `reset` pins for MAC only (without internal FIFO).<br>• `clk` and `ref_clk` input pins for MAC and PCS with transceiver (without internal FIFO). |
| GLOBAL_SIGNAL | Regional clock | To assign clock signals to use the regional clock network. Use this setting to guide the Intel Quartus Prime software in the fitter process for better timing closure. | • `rx_clk` <n> and `tx_clk` <n> input pins for MAC only using MII/GMII interface (without internal FIFO).<br>• `rx_clk` <n> input pin for MAC only using RGMII interface (without internal FIFO). |
| GLOBAL_SIGNAL | OFF | To prevent a signal to be used as a global signal. | Signals for Arria V devices:<br>• `*reset_ff_wr` and `*reset_ff_rd`<br>• `*\|altera_tse_reset_synchronizer_chain_out` |

# 3. Parameter Settings

You customize the Triple-Speed Ethernet IP core by specifying parameters using the Triple-Speed Ethernet parameter editor, launched from Platform Designer in the Intel Quartus Prime software. The customization enables specific core features during synthesis and generation.

This chapter describes the parameters and how they affect the behavior of the IP core. Each section corresponds to a page in the **Parameter Settings** tab in the parameter editor interface.

## 3.1. Core Configuration

**Table 19. Core Configuration Parameters**

| Name | Value | Description |
|---|---|---|
| **Core Variation** | • 10/100/1000Mb Ethernet MAC<br>• 10/100/1000Mb Ethernet MAC with 1000BASE-X/SGMII PCS<br>• 10/100/1000Mb Ethernet MAC with 1000BASE-X/SGMII 2XTBI PCS<br>• 1000BASE-X/SGMII PCS only<br>• 1000BASE-X/SGMII 2XTBI PCS only<br>• 1000Mb Small MAC<br>• 10/100Mb Small MAC | Determines the primary blocks to include in the variation.<br>*Note:* 10/100/1000 Mb Ethernet MAC with 1000BASE-X/SGMII 2XTBI PCS and 1000BASE-X/SGMII 2XTBI PCS only are only supported when you select E-Tile as the transceiver tile in Intel Quartus Prime Pro Edition software version 19.2 onwards. |
| **Enable ECC protection** | On/Off | Turn on this option to enable ECC protection for internal memory blocks. |
| **Interface** | • MII<br>• GMII<br>• RGMII<br>• MII/GMII | Determines the Ethernet-side interface of the MAC block.<br>• **MII**—The only option available for 10/100 Mb Small MAC core variations.<br>• **GMII**—Available only for 1000 Mb Small MAC core variations.<br>• **RGMII**—Available for 10/100/1000 Mb Ethernet MAC and 1000 Mb Small MAC core variations.<br>• **MII/GMII**—Available only for 10/100/1000 Mb 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.<br>*Note:* The RGMII interface is not supported in Intel Stratix 10, Intel Arria 10, and Intel Cyclone 10 GX devices from Intel Quartus Prime software version 17.1 onwards. |

*continued...*

| Name | Value | Description |
|---|---|---|
| **Use clock enable for MAC** | On/Off | Turn on this option to include clock enable signals for the MAC. This option is only applicable for 10/100/1000Mb Ethernet MAC and 1000Mb Small MAC core variations. |
| **Use internal FIFO** | On/Off | Turn on this option to include internal FIFO buffers in the core. You can only include internal FIFO buffers in single-port MACs. |
| **Number of ports** | **1**, **4**, **8**, **12**, **16**, **20**, and **24** | Specifies the number of Ethernet ports supported by the IP core. This parameter is enabled if the parameter **Use internal FIFO** is turned off. A multiport MAC does not support internal FIFO buffers.<br>*Note:* For Intel Quartus Prime software version 17.1 onwards, the number of ports supported for Triple-Speed Ethernet designs targeting Intel Stratix 10, Intel Arria 10, and Intel Cyclone 10 GX devices is 8. This is applicable only when you select LVDS I/O for the **Transceiver type** option. |
| **Transceiver type** | • None<br>• LVDS I/O<br>• GXB | This option is only available for variations that include the PCS block.<br>• **None**—the PCS block does not include an integrated transceiver module. The PCS block implements a ten-bit interface (TBI) to an external SERDES chip.<br>• **LVDS I/O** or **GXB**—the IP core 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. For Intel Stratix 10 devices, the GXB option is only available in the Intel Stratix 10 devices with E-tile transceivers. GXB and LVDS I/O options are not available in Intel Cyclone 10 LP devices.<br>*Note:* There may be a performance risk if you use the Triple-Speed Ethernet IP variant with LVDS I/O for PMA implementation in Intel Arria 10 devices for Intel Quartus Prime software version 17.0.2 and earlier. To avoid the performance risk, Intel recommends that you regenerate the Triple-Speed Ethernet IP core and recompile the design in the Intel Quartus Prime software version 17.1 or later. To download and install the software patch for Intel Quartus Prime version 17.0.2, refer to KDB link: *Performance Risk Running Triple Speed Ethernet LVDS in Arria 10 Devices*. |

### Related Information

- MAC and PCS With LVDS Soft-CDR I/O on page 154
- KDB Link: Performance Risk Running Triple Speed Ethernet LVDS in Intel Arria 10 Devices

## 3.2. Ethernet MAC Options

These options are enabled when your variation includes the MAC function. In small MACs, only the following options are available:

- **Enable MAC 10/100 half duplex support** (10/100 Small MAC variations)
- **Align packet headers to 32-bit boundary** (10/100 and 1000 Small MAC variations)

## Table 20. MAC Options Parameters

| Name | Value | Description |
|---|---|---|
| **Ethernet MAC Options** | | |
| **Enable MAC 10/100 half duplex support** | On/Off | Turn on this option to include support for half duplex operation on 10/100 Mbps connections. |
| **Enable local loopback on MII/GMII/RGMII** | On/Off | 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** | On/Off | Turn on this option to include support for supplementary destination MAC unicast addresses for fast hardware-based received frame filtering. |
| **Include statistics counters** | On/Off | 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.<br>By default, the width of all statistics counters are 32 bits. |
| **Enable 64-bit statistics byte counters** | On/Off | Turn on this option to extend the width of selected statistics counters— `aOctetsTransmittedOK`, `aOctetsReceivedOK`, and `etherStatsOctets`—to 64 bits. |
| **Include multicast hashtable** | On/Off | 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 boundary** | On/Off | Turn on this option to include logic that aligns all packet headers to a 32-bit boundary. This helps reduce software overhead processing in realignment of data buffers.<br>This option is available for MAC variations with 32 bits wide internal FIFO buffers and MAC variations without internal FIFO buffers.<br>You must turn on this option if you intend to use the Triple-Speed Ethernet IP core with the Interniche TCP/IP protocol stack. |
| **Enable full-duplex flow control** | On/Off | Turn on this option to include the logic for full-duplex flow control that includes pause frames generation and termination. |
| **Enable VLAN detection** | On/Off | Turn on this option to include the logic for VLAN and stacked VLAN frame detection. When turned off, the MAC does not detect VLAN and staked VLAN frames. The MAC forwards these frames to the user application without processing them. |
| **Enable magic packet detection** | On/Off | Turn on this option to include logic for magic packet detection (Wake-on LAN). |
| **MDIO Module** | | |
| **Include MDIO module (MDC/MDIO)** | On/Off | Turn on this option if you want to access external PHY devices connected to the MAC function. When turned off, the core does not include the logic or signals associated with the MDIO interface. |
| **Host clock divisor** | — | Clock divisor to divide the MAC control interface clock to produce the MDC clock output on the MDIO interface. The default value is 40.<br>For example, if the MAC control 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.<br>Intel recommends that the division factor is defined such that the MDC frequency does not exceed 2.5 MHz. |

## 3.3. FIFO Options

The FIFO options are enabled only for MAC variations that include internal FIFO buffers.

**Table 21. FIFO Options Parameters**

| Name | Value | Parameter |
|---|---|---|
| **Width** | | |
| **Width** | **8 Bits** and **32 Bits** | Determines the data width in bits of the transmit and receive FIFO buffers. |
| **Depth** | | |
| **Transmit** | Between 64 and 64K | Determines the depth of the internal FIFO buffers. |
| **Receive** | | |

## 3.4. Timestamp Options

**Table 22. Timestamp Options Parameters**

| Name | Value | Parameter |
|---|---|---|
| **Timestamp** | | |
| **Enable timestamping** | On/Off | Turn on this parameter to enable time stamping on the transmitted and received frames. |
| **Enable PTP 1-step clock** | On/Off | Turn on this parameter to insert timestamp on PTP messages for 1-step clock based on the TX Timestamp Insert Control interface.<br>This parameter is disabled if you do not turn on **Enable timestamping**. |
| **Timestamp fingerprint width** | — | Use this parameter to set the width in bits for the timestamp fingerprint on the TX path. The default value is 4 bits. |

## 3.5. PCS/Transceiver Options

The PCS/Transceiver options are enabled only if your core variation includes the PCS function.

**Table 23. PCS/Transceiver Options Parameters**

| Name | Value | Parameter |
|---|---|---|
| **PCS Options** | | |
| **PHY ID (32 bit)** | — | Configures the PHY ID of the PCS block. |
| **Enable SGMII bridge** | On/Off | Turn on this option to add the SGMII clock and rate-adaptation logic to the PCS block. This option allows you to configure the PCS either in SGMII mode or 1000Base-X mode. If your application only requires 1000BASE-X PCS, turning off this option reduces resource usage.<br>In Cyclone IV GX devices, `REFCLK[0,1]` and `REFCLK[4,5]` cannot connect directly to the GCLK network. If you enable the SGMII bridge, you must connect `ref_clk` to an alternative dedicated clock input pin. |
| **Transceiver Options**—apply only to variations that include GXB transceiver blocks | | |

*continued...*

| Name | Value | Parameter |
|---|---|---|
| **Export transceiver powerdown signal** | On/Off | This option is not supported in Stratix V, Arria V, Arria V GZ, and Cyclone V devices.<br>Turn on this option to export the powerdown signal of the GX transceiver to the top-level of your design. Powerdown is shared among the transceivers in a quad. Therefore, turning on this option in multiport Ethernet configurations maximizes efficient use of transceivers within the quad.<br>Turn off this option to connect the powerdown signal internally to the PCS control register interface. This connection allows the host processor to control the transceiver powerdown in your system. |
| **Enable transceiver dynamic reconfiguration** | On/Off | This option is always turned on in devices other than Arria GX and Stratix II GX. When this option is turned on, the Intel FPGA IP core includes the dynamic reconfiguration signals.<br>For designs targeting devices other than Arria V, Cyclone V, Stratix V, Intel Arria 10, and Intel Cyclone 10 GX, Intel recommends that you instantiate the ALTGX_RECONFIG megafunction and connect the megafunction to the dynamic reconfiguration signals to enable offset cancellation.<br>For Arria V, Cyclone V, and Stratix V designs, Intel recommends that you instantiate the Transceiver Reconfiguration Controller megafunction and connect the megafunction to the dynamic reconfiguration signals to enable offset cancellation. The transceivers in the Arria V, Cyclone V, and Stratix V designs are configured with Intel FPGA Custom PHY IP core. The Custom PHY IP core require two reconfiguration interfaces for external reconfiguration controller. For more information on the reconfiguration interfaces required, refer to the V-Series Transceiver PHY IP Core User Guide and the respective device handbook.<br>For more information about quad sharing considerations, refer to Sharing PLLs in Devices with GIGE PHY on page 157. |
| **Starting channel number** | **0 – 284** | Specifies the channel number for the GXB transceiver block. In a multiport MAC, this parameter specifies the channel number for the first port. Subsequent channel numbers are in four increments.<br>In designs with multiple instances of GXB transceiver block (multiple instances of Triple-Speed Ethernet IP core with GXB transceiver block or a combination of Triple-Speed Ethernet IP core and other IP cores), Intel recommends that you set a unique starting channel number for each instance to eliminate conflicts when the GXB transceiver blocks share a transceiver quad.<br>This option is not supported in Arria V, Cyclone V, Stratix V, Intel Arria 10, and Intel Cyclone 10 GX devices. For these devices, the channel numbers depends on the dynamic reconfiguration controller. |
| **Series V GXB Transceiver Options** | | |
| **TX PLLs type** | • **CMU**<br>• **ATX** | This option is only available for variations that include the PCS block for Stratix V and Arria V GZ devices.<br>Specifies the TX phase-locked loops (PLLs) type—CMU or ATX—in the GXB transceiver for Series V devices. |
| **Enable SyncE Support** | On/Off | Turn on this option to enable SyncE support by separating the TX PLL and RX PLL reference clock. |
| **TX PLL clock network** | • **x1**<br>• **xN** | This option is only available for variations that include the PCS block for Arria V and Cyclone V devices.<br>Specifies the TX PLL clock network type. |
| **Intel Arria 10 or Intel Cyclone 10 GX GXB Transceiver Options** | | |

*continued...*

| Name | Value | Parameter |
|---|---|---|
| **Enable Intel Arria 10 or Intel Cyclone 10 GX transceiver dynamic reconfiguration** | On/Off | Turn on this option for the Intel FPGA IP core to include the dynamic reconfiguration signals. |
| **Intel Stratix 10 GXB Transceiver Options** | | |
| **Enable E-tile transceiver dynamic reconfiguration** | On/Off | Turn on this option for the Intel FPGA IP core to include the dynamic reconfiguration signals. |

*Note:*

1. You must configure the Intel Arria 10/Intel Cyclone 10 GX Transceiver ATX PLL with an output clock frequency of 1250.0 MHz (instead of applying the default value of 625 MHz) when using the Intel Arria 10/Intel Cyclone 10 GX Transceiver Native PHY with the Triple-Speed Ethernet Intel FPGA IP core.
2. The **Transceiver Options** and **Series V GXB Transceiver Options** parameters are not available in the Triple-Speed Ethernet Intel FPGA IP parameter editor interface of the Intel Quartus Prime Pro Edition software version 19.2 onwards. These options are only present in the Intel Quartus Prime Standard Edition software.

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

### Related Information

- E-Tile Transceiver PHY User Guide
  More information about dynamic reconfiguration in the Intel Stratix 10 and Intel Agilex devices.
- Intel Arria 10 Transceiver PHY User Guide
  More information about the Intel Arria 10 Transceiver ATX PLL.
- Intel Cyclone 10 GX Transceiver PHY User Guide
  More information about the Intel Cyclone 10 GX Transceiver ATX PLL.

# 4. Functional Description

The Triple-Speed Ethernet IP core includes the following functions:

- 10/100/1000 Ethernet MAC
- 1000BASE-X/SGMII PCS With Optional Embedded PMA
- Intel FPGA IEEE 1588v2

## 4.1. 10/100/1000 Ethernet MAC

The Intel 10/100/1000 Ethernet MAC function handles the flow of data between user applications and Ethernet network through an internal or external Ethernet PHY. Intel offers the following MAC variations:

- Variations with internal FIFO buffers—supports only single port.
- Variations without internal FIFO buffers—supports up to 24 ports (except for Intel Arria 10, Intel Cyclone 10 GX, and Intel Stratix 10 devices) and the ports can operate at different speeds.
- Small MAC—provides basic functionalities of a MAC function using minimal resources.

  Refer to 10/100/1000 Ethernet MAC Versus Small MAC on page 8 for a feature comparison between the 10/100/1000 Ethernet MAC and small MAC.

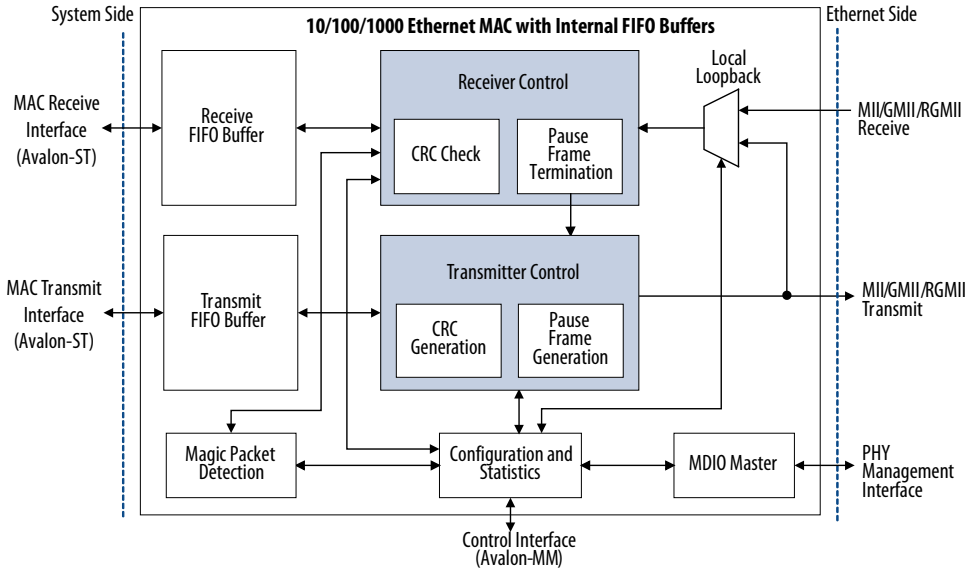
The MAC function supports the following Ethernet frames: basic, VLAN and stacked VLAN, jumbo, and control frames.

**Related Information**

Ethernet Frame Format on page 195

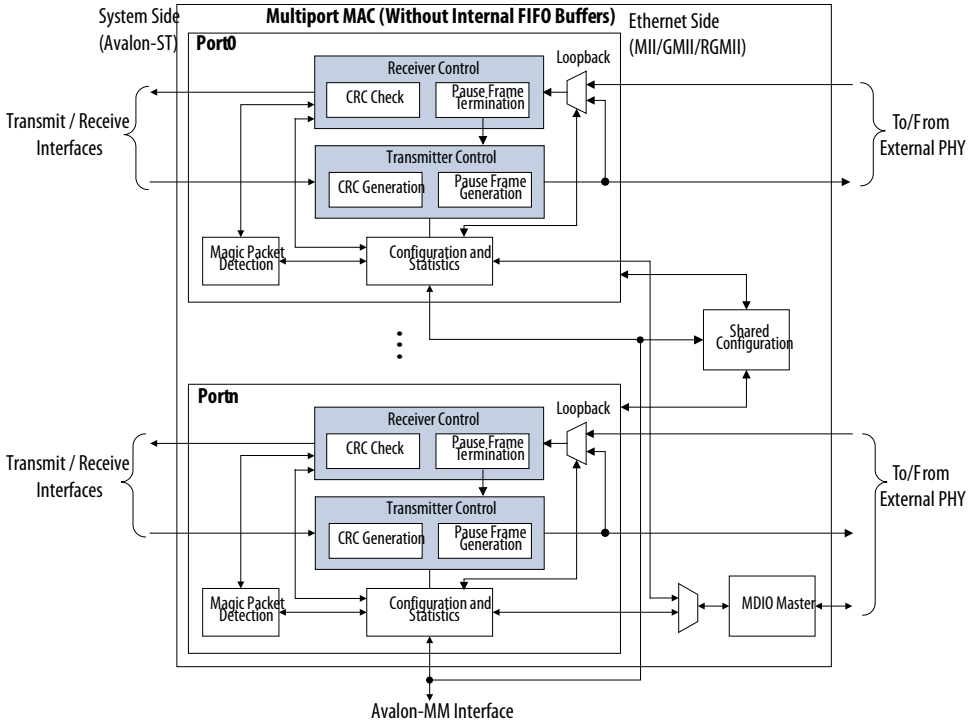
## 4.1.1. MAC Architecture

**Figure 11. 10/100/1000 Ethernet MAC With Internal FIFO Buffers**




The FIFO buffers, which you can configure to 8- or 32-bits wide, store the transmit and receive data. The buffer width determines the data width on the Avalon Streaming receive and transmit interfaces. You can configure the FIFO buffers to operate in cut-through or store-and-forward mode using the `rx_section_full` and `tx_section_full` registers.

**Figure 12. Multiport MAC Without Internal FIFO Buffers**




In a multiport MAC, the instances share the MDIO master and some configuration registers. You can use the Avalon Streaming Multi-Channel Shared Memory FIFO core in Platform Designer to store the transmit and receive data.

### Related Information

MAC Configuration Register Space on page 81

## 4.1.2. MAC Interfaces

The MAC function implements the following interfaces:

- Avalon Streaming on the system side.
  - Avalon Streaming sink port on transmit with the following properties:
    - Fixed data width, 8 bits, in MAC variations without internal FIFO buffers; configurable data width, 8 or 32 bits, in MAC variations with internal FIFO buffers.
    - Packet support using start-of-packet (SOP) and end-of-packet (EOP) signals, and partial final packet signals.
    - Error reporting.
    - Variable-length ready latency specified by the `tx_almost_full` register.
  - Avalon Streaming source port on receive with the following properties:
    - Fixed data width of 8 bits in MAC variations without internal FIFO buffers; configurable data width, 8 or 32 bits, in MAC variations with internal FIFO buffers.
    - Backpressure is supported only in MAC variations with internal FIFO buffers. Transmission stops when the level of the FIFO buffer reaches the respective programmable thresholds.
    - Packet support using SOP and EOP signals, and partial final packet signals.
    - Error reporting.
    - Ready latency is zero in MAC variations without internal FIFO buffers. In MAC variations with internal FIFO buffers, the ready latency is two.
- Media independent interfaces on the network side—select MII, GMII, or RGMII by setting the **Interface** option on the **Core Configuration** page or the ETH_SPEED bit in the `command_config` register.
- Control interface—an Avalon Memory-Mapped slave port that provides access to 256 32-bit configuration and status registers, and statistics counters. This interface supports the use of `waitrequest` to stall the interconnect fabric for as many cycles as required.
- PHY management interface—implements the standard MDIO specification, IEEE 803.2 standard Clause 22, to access the PHY device management registers. This interface supports up to 32 PHY devices.

MAC variations without internal FIFO buffers implement the following additional interfaces:

- FIFO status interface—an Avalon Streaming sink port that streams in the fill level of an external FIFO buffer. Only MAC variations without internal buffers implement this interface.
- Packet classification interface—an Avalon Streaming source port that streams out receive packet classification information. Only MAC variations without internal buffers implement this interface.

### Related Information

- Transmit Thresholds on page 47
- Interface Signals on page 106
- MAC Configuration Register Space on page 81
- Avalon Interface Specifications
  More information about the Avalon interfaces.

### 4.1.3. MAC Transmit Datapath

On the transmit path, the MAC function accepts frames from a user application and constructs Ethernet frames before forwarding them to the PHY. Depending on the MAC configuration, the MAC function could perform the following tasks: realigns the payload, modifies the source address, calculates and appends the CRC-32 field, and inserts interpacket gap (IPG) bytes. In half-duplex mode, the MAC function also detects collision and attempts to retransmit frames when a collision occurs. The following conditions trigger transmission:

- In MAC variations with internal FIFO buffers:
  - Cut-through mode—transmission starts when the level of the FIFO level hits the transmit section-full threshold.
  - Store and forward mode—transmission starts when a full packet is received.

    *Note:* To use the internal FIFO buffer in the store and forward mode, ensure that the internal FIFO buffer size is larger than the transmitted packet size. If the transmitted packet size is larger than the internal FIFO buffer size, the Triple-Speed Ethernet IP core locks up with no packet transmission. The lock-up of the IP core can only be cleared through a hard reset to the IP core.
- In MAC variations without internal FIFO buffers, transmission starts as soon as data is available on the Avalon Streaming transmit interface.

**Related Information**

Ethernet Frame Format on page 195

#### 4.1.3.1. IP Payload Re-alignment

If you turn the **Align packet headers to 32-bit boundaries** option, the MAC function removes the additional two bytes from the beginning of Ethernet frames.

**Related Information**

IP Payload Alignment on page 43

#### 4.1.3.2. Address Insertion

By default, the MAC function retains the source address received from the user application. You can configure the MAC function to replace the source address with the primary MAC address or any of the supplementary addresses by setting the TX_ADDR_INS bit in the `command_config` register to 1. The TX_ADDR_SEL bits in the `command_config` register determines the address selection.

**Related Information**

Command_Config Register (Dword Offset 0x02) on page 85

#### 4.1.3.3. Frame Payload Padding

The MAC function inserts padding bytes (`0x00`) when the payload length does not meet the minimum length required:

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

### 4.1.3.4. CRC-32 Generation

To turn on CRC-32 generation, you must set the `OMIT_CRC` bit in the `tx_cmd_stat` register to 0 and send the frame to the MAC function with the `ff_tx_crc_fwd` signal deasserted.

The following equation shows the CRC polynomial, as specified in the IEEE 802.3 standard:

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

The 32-bit CRC value occupies the FCS field with X31 in the least significant bit of the first byte. The CRC bits are thus transmitted in the following order: X31, X30,..., X1, X0.

### 4.1.3.5. Interpacket Gap Insertion

In full-duplex mode, the MAC function maintains the minimum number of IPG configured in the `tx_ipg_length` register between transmissions. You can configure the minimum IPG 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. Transmission starts only when 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 when the `m_rx_crs` signal is deasserted.

### 4.1.3.6. Collision Detection in Half-Duplex Mode

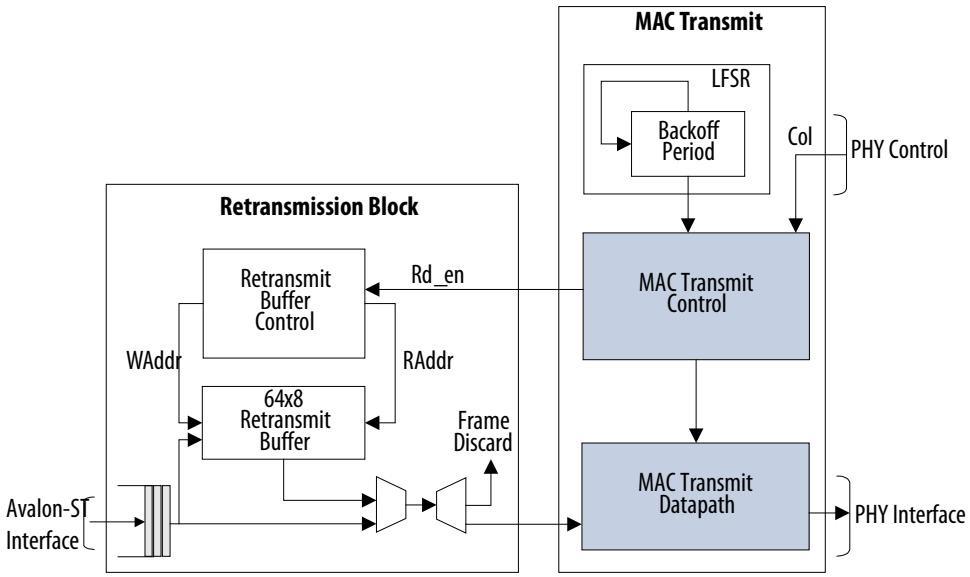
Collision occurs only in a half-duplex network. It occurs when two or more nodes transmit concurrently. The PHY device asserts the `m_rx_col` signal to indicate collision.

When the MAC function detects collision during transmission, it stops the transmission and sends a 32-bit jam pattern instead. A jam pattern is a fixed pattern, 0x648532A6, and is not compared to the CRC of the frame. The probability of a jam pattern to be identical to the CRC is very low, 0.532%.

If the MAC function detects collision while transmitting the preamble or SFD field, it sends the jam pattern only after transmitting the SFD field, which subsequently results in a minimum of 96-bit fragment.

If the MAC function detects collision while transmitting the first 64 bytes, including the preamble and SFD fields, the MAC function waits for an interval equal to the backoff period and then retransmits the frame. The frame is stored in a 64-byte retransmit buffer. The backoff period is generated from a pseudo-random process, truncated binary exponential backoff.

**Figure 13. 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:

$0 \leq r < 2k$

$k = \min(n, N)$, 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 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 transmit FIFO buffer.

In networks that violate standard requirements, collision may occur after the transmission of the first 64 bytes. If this happens, the MAC function stops transmitting the current frame, discards the rest of the frame from the transmit FIFO buffer, and resumes transmitting the next available frame. You can check the LATE_COL register (`command_config [12]`) to verify if the MAC has discarded any frame due to collision.

## 4.1.4. MAC Receive Datapath

The MAC function receives Ethernet frames from the network via a PHY and forwards the payload with relevant frame fields to the user application after performing checks, filtering invalid frames, and removing the preamble and SFD.

#### 4.1.4.1. Preamble Processing

The MAC function uses the SFD (0xD5) to identify the last byte of the preamble. If an SFD is not found after the seventh byte, 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 8-bytes and 6-bytes in RGMII/GMII (1000 Mbps operation) and RGMII/MII (10/100 Mbps operation) respectively.

The MAC function removes the preamble and SFD fields from valid frames.

#### 4.1.4.2. Collision Detection in Half-Duplex Mode

In half-duplex mode, the MAC function checks for collisions during frame reception. When 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 MAC function forwards the frame to the user application with error.

#### 4.1.4.3. Address Checking

The MAC function can accept frames with the following address types:

- Unicast address—bit 0 of the destination address is 0.
- Multicast address—bit 0 of the destination address is 1.
- Broadcast address—all 48 bits of the destination address are 1.

  The MAC function always accepts broadcast frames. If promiscuous mode is enabled (PROMIS_EN bit in the command_config register = 1), the MAC function omits address filtering and accepts all frames.

##### 4.1.4.3.1. Unicast Address Checking

When promiscuous mode is disabled, the MAC function only accepts unicast frames if the destination address matches any of the following addresses:

- The primary address, configured in the registers mac_0 and mac_1
- The supplementary 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.

##### 4.1.4.3.2. Multicast Address Resolution

You can use either a software program running on the host processor or a hardware multicast address resolution engine to resolve multicast addresses. Address resolution using a software program can affect the system performance, especially in gigabit mode.

The MAC function uses a 64-entry hash table in the register space, multicast hash table, to implement the hardware multicast address resolution engine as shown in figure below. 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 below. This code represents the address of an entry in the hash table. Write one to the most significant bit in the table entry. All multicast addresses that hash to the address of this entry are valid and accepted.

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 MHASH_SEL bit to 1. The latter option is provided if you want 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 14. Hardware Multicast Address Resolution Engine**

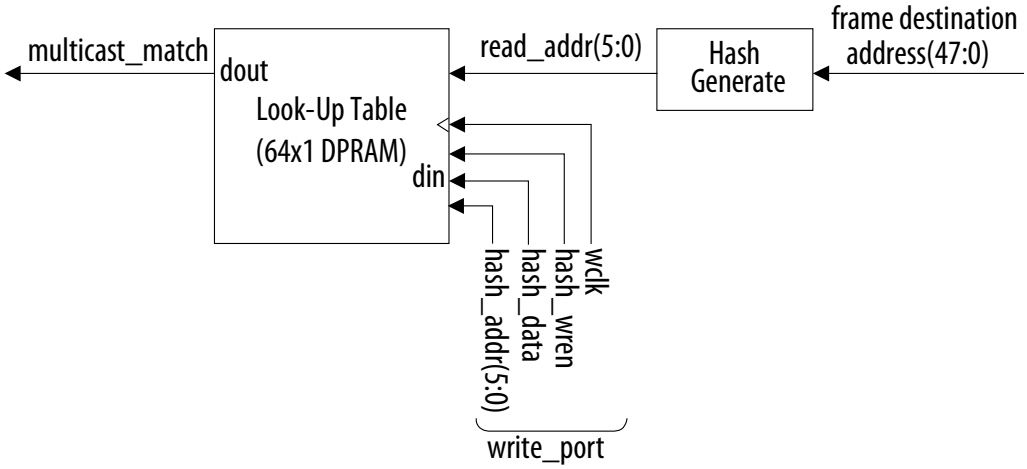



**Table 24. Hash Code Generation—Full Destination Address**

Algorithm for generating the 6-bit code from the entire 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 25. Hash Code Generation—Lower 24 Bits of Destination Address**

Algorithm for generating the 6-bit code from the lower 24 bits of the 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 |

*continued...*

| Hash Code Bit | Value |
|---|---|
| 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.

*Note:*

1. All entries in the hash table must be filled.
2. RX_ENA needs to be set to '0' prior to filling the hash table.

Failure to follow notes (1) and (2) can result in error during multicast packet detection.

### 4.1.4.4. Frame Type Validation

The MAC function checks the length/type field to determine the frame type:

- Length/type < 0x600—the field represents the payload length of a basic Ethernet frame. The MAC function continues to check the frame and payload lengths.
- Length/type >= 0x600—the field represents the frame type.
  - Length/type = 0x8100—VLAN or stacked VLAN tagged frames. The MAC function continues to check the frame and payload lengths, and asserts the following signals:
    - for VLAN frames, `rx_err_stat[16]` in MAC variations with internal FIFO buffers or `pkt_class_data[1]` in MAC variations without internal FIFO buffers
    - for stacked VLAN frames, `rx_err_stat[17]` in MAC variations with internal FIFO buffers or `pkt_class_data[0]` in MAC variations without internal FIFO buffers.
  - Length/type = 0x8088—control frames. The next two bytes, the Opcode field, indicate the type of control frame.
    - For pause frames (Opcode = 0x0001), the MAC function continues to check the frame and payload lengths. For valid pause frames, the MAC function proceeds with pause frame processing. The MAC function forwards pause frames to the user application only when the PAUSE_FWD bit in the `command_config` register is set to 1.
    - For other types of control frames, the MAC function accepts the frames and forwards them to the user application only when the CNTL_FRM_ENA bit in the `command_config` register is set to 1.
- For other field values, the MAC function forwards the receive frame to the user application.

**Related Information**

Remote Device Congestion on page 49

### 4.1.4.5. Payload Pad Removal

You can turn on padding removal by setting the PAD_EN bit in the `command_config` register to 1. The MAC function removes the padding, prior to forwarding the frames to the user application, when 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

Padding removal is not applicable when:

- the length or type field is equal to or greater than 1536 (0x600) for basic frame.
- the client length or type field is equal to or greater than 1536 (0x600) for VLAN and stacked VLAN frames.

When padding removal is turned off, complete frames including the padding are forwarded to the Avalon Streaming receive interface.

### 4.1.4.6. CRC Checking

The following equation shows the CRC polynomial, as specified in the IEEE 802.3 standard:

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

The 32-bit CRC value occupies the FCS field with X31 in the least significant bit of the first byte. The CRC bits are thus received in the following order: X31, X30,..., X1, X0.

If the MAC function detects CRC-32 error, it marks the frame invalid by asserting the following signals:

- `rx_err[2]` in MAC variations with internal FIFO buffers.
- `data_rx_error[1]` in MAC variations without internal FIFO buffers.

The MAC function discards frames with CRC-32 error if the RX_ERR_DISC bit in the `command_config` register is set to 1.

For frames less than the required minimum length, the MAC function forwards the CRC-32 field to the user application if the CRC_FWD and PAD_EN bits in the `command_config` register are 1 and 0 respectively. Otherwise, the CRC-32 field is removed from the frame.

### 4.1.4.7. Length Checking

The MAC function checks the frame and payload lengths of basic, VLAN tagged, and stacked VLAN tagged frames.

The frame length must be at least 64 (0x40) bytes and not exceed the following maximum value for the different frame types:

- Basic frames—the value specified in the `frm_length` register
- VLAN tagged frames—the value specified in the `frm_length` register plus four
- Stacked VLAN tagged frames—the value specified in the `frm_length` register plus eight

To prevent FIFO buffer overflow, the MAC function truncates the frame if it is more than 11 bytes longer than the allowed maximum length.

For frames of a valid length, the MAC function continues to check the payload length if the `NO_LGTH_CHECK` bit in the `command_config` register is set to 0. The MAC function keeps track of the payload length as it receives a frame, and checks the length against the length/type field in basic MAC frames or the client length/type field in VLAN tagged frames. The payload length is valid if it satisfies the following conditions:

- The actual payload length matches the value in the length/type or client length/type field.
- Basic frames—the payload length is between 46 (0x2E)and 1536 (0x0600) bytes, excluding 1536.
- VLAN tagged frames—the payload length is between 42 (0x2A)and 1536 (0x0600), excluding 1536.
- Stacked VLAN tagged frames—the payload length is between 38 (0x26) and 1536 (0x0600), excluding 1536.

If the frame or payload length is not valid, the MAC function asserts one of the following signals to indicate length error:

- `rx_err[1]` in MACs with internal FIFO buffers.
- `data_rx_error[0]` in MACs without internal FIFO buffers.

### 4.1.4.8. Frame Writing

The IP core removes the preamble and SFD fields from the frame. The CRC field and padding bytes may be removed depending on the configuration.

For MAC variations with internal FIFO buffers, the MAC function writes the frame to the internal receive FIFO buffers. For MAC variations without internal FIFO buffers, it forwards the frame to the Avalon Streaming receive interface.

MAC variations without internal FIFO buffers do not support backpressure on the Avalon Streaming receive interface. In this variation, if the receiving component is not ready to receive data from the MAC function, the frame gets truncated with error and subsequent frames are also dropped with error.

### 4.1.4.9. IP Payload Alignment

The network stack makes frequent use of the IP addresses stored in Ethernet frames. When you turn on the **Align packet headers to 32-bit boundaries** option, the MAC function aligns the IP payload on a 32-bit boundary by adding two 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 26. 32-Bit Interface Data Structure — Non-IP Aligned Ethernet Frame**

| Bits | | | |
|---|---|---|---|
| **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 27. 32-Bit Interface Data Structure — IP Aligned Ethernet Frame**

| Bits | | | |
|---|---|---|---|
| **31...24** | **23...16** | **15...8** | **7...0** |
| padded with zeros | | Byte 0 | Byte 1 |
| Byte 2 | Byte 3 | Byte 4 | Byte 5 |

## 4.1.5. MAC Transmit and Receive Latencies

Intel uses the following definitions for the transmit and receive latencies:

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

**Table 28. Transmit and Receive Nominal Latency**

The transmit and receive nominal latencies in various modes. The FIFO buffer thresholds are set to the typical values specified in this user guide when deriving the latencies. Under *MAC Options* tab, only the following options are selected when deriving the latencies shown in the table below: **Enable MAC 10/100 half duplex support**, **Include statistics counters**, and **Enable magic packet detection**.

| MAC Configuration | Latency (Clock Cycles) (3) (4) | |
|---|---|---|
| | Transmit | Receive |
| **MAC with Internal FIFO Buffers** (5) | | |
| GMII in gigabit and cut-through mode | 38 | 99 |
| MII in 100M and cut-through mode | 40 | 184 |
| MII in 10M and cut-through mode | 34 | 183 |
| RGMII in gigabit and cut-through mode | 40 | 102 |
| RGMII in 10 Mbps and cut-through mode | 41 | 187 |
| RGMII in 100 Mbps and cut-through mode | 36 | 186 |
| **MAC without Internal FIFO Buffers** (6) | | |

*continued...*

(3) The clocks in all domains are running at the same frequency.

(4) The numbers in this table are from simulation.

(5) The data width is set to 32 bits

| MAC Configuration | Latency (Clock Cycles) (3) (4) | |
|---|---|---|
| | Transmit | Receive |
| GMII | 15 | 28 |
| MII | 26 | 56 |
| RGMII in gigabit mode | 16 | 31 |
| RGMII in10/100 Mbps | 27 | 59 |

**Related Information**

Base Configuration Registers (Dword Offset 0x00 – 0x17) on page 82

## 4.1.6. FIFO Buffer Thresholds

For MAC variations with internal FIFO buffers, you can change the operations of the FIFO buffers, and manage potential FIFO buffer overflow or underflow by configuring the following thresholds:

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

These thresholds are defined in bytes for 8-bit wide FIFO buffers and in words for 32-bit wide FIFO buffers. The FIFO buffer thresholds are configured via the registers.

(3) The clocks in all domains are running at the same frequency.

(4) The numbers in this table are from simulation.

(6) The data width is set to 8 bits.

### 4.1.6.1. Receive Thresholds

**Figure 15. Receive FIFO Thresholds**

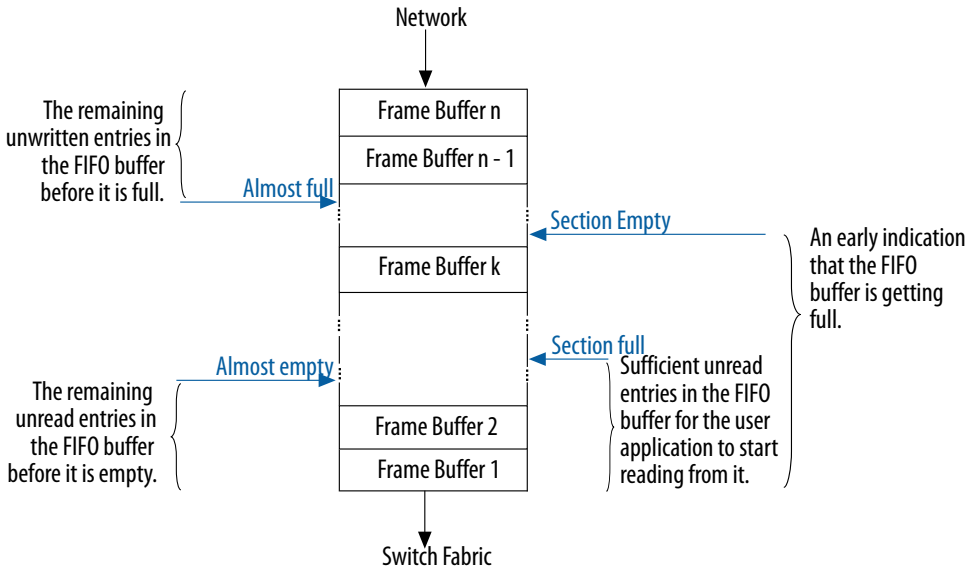



**Table 29. Receive Thresholds**

| Threshold | Register Name | Description |
|---|---|---|
| Almost empty | `rx_almost_empty` | The number of unread entries in the FIFO buffer before the buffer is empty. When the level of the FIFO buffer reaches this threshold, the MAC function asserts the `ff_rx_a_empty` signal. The MAC function stops reading from the FIFO buffer and subsequently stops transferring data to the user application to avoid buffer underflow.<br>When the MAC function detects an EOP, it 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 buffer before the buffer is full. When the level of the FIFO buffer reaches this threshold, the MAC function asserts the `ff_rx_a_full` signal. If the user application is not ready to receive data (`ff_rx_rdy = 0`), the MAC function performs the following operations:<br>• Stops writing data to the FIFO buffer.<br>• Truncates received frames to avoid FIFO buffer overflow.<br>• Asserts the `rx_err[0]` signal when the `ff_rx_eop` signal is asserted.<br>• Marks the truncated frame invalid by setting the `rx_err[3]` signal to 1.<br>If the RX_ERR_DISC bit in the `command_config` register is set to 1 and the section-full (`rx_section_full`) |

*continued...*

| Threshold | Register Name | Description |
|---|---|---|
| | | threshold is set to 0, the MAC function discards frames with error received on the Avalon Streaming interface. |
| Section empty | `rx_section_empty` | An early indication that the FIFO buffer is getting full. When the level of the FIFO buffer hits this threshold, the MAC function generates an XOFF pause frame to indicate FIFO congestion to the remote Ethernet device. When the FIFO level goes below this threshold, the MAC function generates an XON pause frame to indicate its readiness to receive new frames.<br>To avoid data loss, you can use this threshold as an early warning to the remote Ethernet device on the potential FIFO buffer congestion before the buffer level hits the almost-full threshold. The MAC function truncates receive frames when the buffer level hits the almost-full threshold. |
| Section full | `rx_section_full` | The section-full threshold indicates that there are sufficient entries in the FIFO buffer for the user application to start reading from it. The MAC function asserts the `ff_rx_dsav` signal when the buffer level hits this threshold.<br>Set this threshold to 0 to enable store and forward on the receive datapath. In the store and forward mode, the `ff_rx_dsav` signal remains deasserted. The MAC function asserts the `ff_rx_dval` signal as soon as a complete frame is written to the FIFO buffer. |

### 4.1.6.2. Transmit Thresholds

**Figure 16. Transmit FIFO Thresholds**

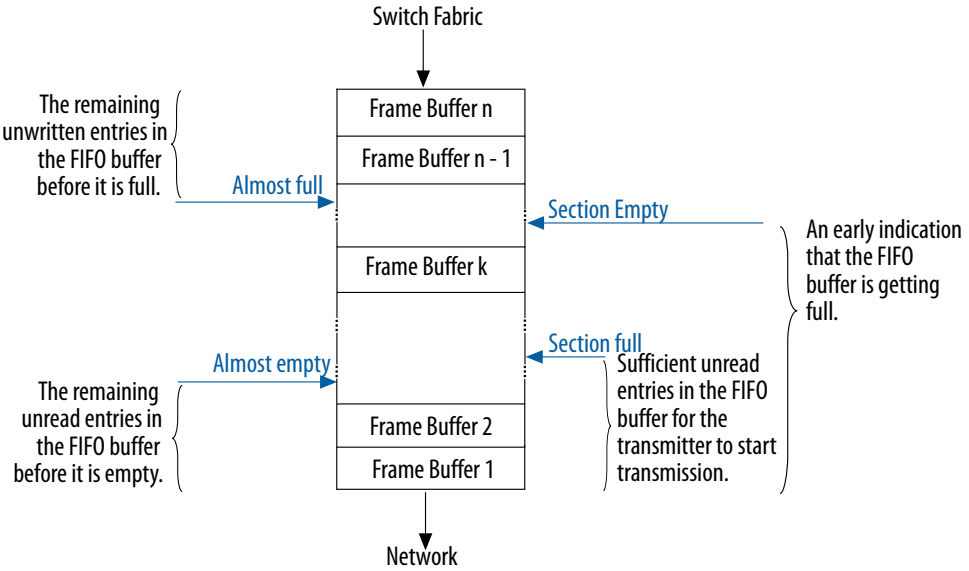


**Table 30. Transmit Thresholds**

| Threshold | Register Name | Description |
|---|---|---|
| Almost empty | `tx_almost_empty` | The number of unread entries in the FIFO buffer before the buffer is empty. When the level of the FIFO buffer reaches this threshold, the MAC function asserts the `ff_tx_a_empty` signal. The MAC function stops reading from the FIFO buffer and sends the Ethernet frame with GMII / MII/ RGMII error to avoid FIFO underflow. |
| Almost full | `tx_almost_full` | The number of unwritten entries in the FIFO buffer before the buffer is full. When the level of the FIFO buffer reaches this threshold, the MAC function asserts the `ff_tx_a_full` signal. The MAC function deasserts the `ff_tx_rdy` signal to backpressure the Avalon Streaming transmit interface. |
| Section empty | `tx_section_empty` | An early indication that the FIFO buffer is getting full. When the level of the FIFO buffer reaches this threshold, the MAC function deasserts the `ff_tx_septy` signal. This threshold can serve as a warning about potential FIFO buffer congestion. |
| Section full | `tx_section_full` | This threshold indicates that there are sufficient entries in the FIFO buffer to start frame transmission.<br>Set this threshold to 0 to enable store and forward on the transmit path. When you enable the store and forward mode, the MAC function forwards each frame as soon as it is completely written to the transmit FIFO buffer. |

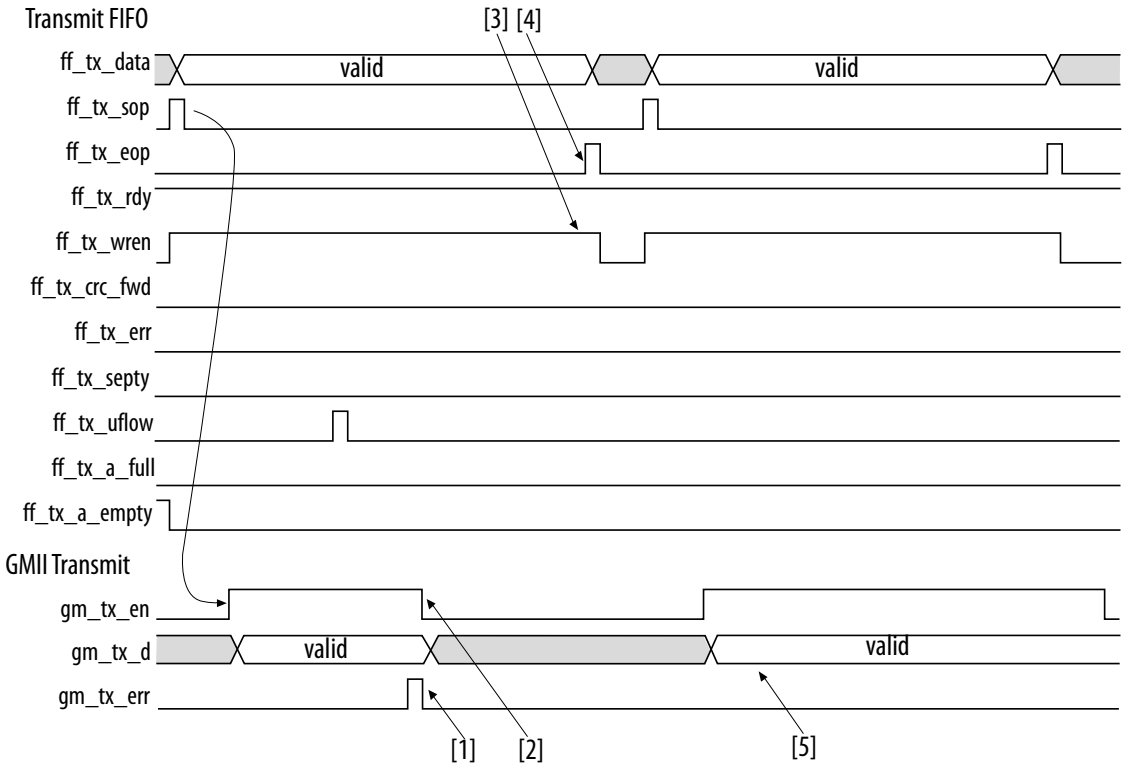
### 4.1.6.3. Transmit FIFO Buffer Underflow

If the transmit FIFO buffer hits the almost-empty threshold during transmission and the FIFO buffer does not contain the end-of-packet indication, the MAC function stops reading data from the FIFO buffer and initiates the following actions:

1. The MAC function asserts the RGMII/GMII/MII error signals (`tx_control`/ `gm_tx_err`/`m_tx_err`) to indicate that the fragment transferred is not valid.
2. The MAC function deasserts the RGMII/GMII/MII transmit enable signals (`tx_control`/`gm_tx_en`/`m_tx_en`) to terminate the frame transmission.
3. After the underflow, the user application completes the frame transmission.
4. The transmitter control discards any new data in the FIFO buffer until the end of frame is reached.
5. The MAC function starts to transfer data on the RGMII/GMII/MII when the user application sends a new frame with an SOP.

**Figure 17. Transmit FIFO Buffer Underflow**

Figure illustrates the FIFO buffer underflow protection algorithm for gigabit Ethernet system.




## 4.1.7. Congestion and Flow Control

In full-duplex mode, the MAC function implements flow control to manage the following types of congestion:

- Remote device congestion—the receiving device experiences congestion and requests the MAC function to stop sending data.
- Receive FIFO buffer congestion—when the receive FIFO buffer is 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, such as a processor, can request the remote device to stop data transmission.

### Related Information

MAC Configuration Register Space on page 81

### 4.1.7.1. Remote Device Congestion

When the MAC function receives an XOFF pause frame and the PAUSE_IGNORE bit in the `command_config` register is set to 0, the MAC function completes the transfer of the current frame and stops transmission for the amount of time specified by the pause quanta in 512 bit times increments. Transmission resumes when the timer expires or when the MAC function receives an XON frame.

You can configure the MAC function to ignore pause frames by setting the PAUSE_IGNORE bit in the command_config register is set to 1.

### 4.1.7.2. Receive FIFO Buffer and Local Device Congestion

Pause frames generated are compliant to the IEEE Standard 802.3 annex 31A & B. The MAC function generates pause frames when the level of the receive FIFO buffer hits a level that can potentially cause an overflow, or at the request of the user application. The user application can trigger the generation of an XOFF pause frame by setting the XOFF_GEN bit in the command_config register to 1 or asserting the xoff_gen signal.

For MAC variations with internal FIFO buffers, the MAC function generates an XOFF pause frame when the level of the FIFO buffer reaches the section-empty threshold (rx_section_empty). If transmission is in progress, the MAC function waits for the transmission to complete before generating the pause frame. The fill level of an external FIFO buffer is obtained via the Avalon Streaming receive FIFO status interface.

When generating a pause frame, the MAC function fills the pause quanta bytes P1 and P2 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).

The MAC function automatically generates an XON pause frame when the FIFO buffer section-empty flag is deasserted and the current frame transmission is completed. The user application can trigger the generation of an XON pause frame by clearing the XOFF_GEN bit and signal, and subsequently setting the XON_GEN bit to 1 or asserting the xon_gen signal.

When generating an XON pause frame, the MAC function fills the pause quanta (payload bytes P1 and P2) 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).

In addition to the flow control mechanism, the MAC function prevents an overflow by truncating excess frames. The status bit, rx_err[3], is set to 1 to indicate such errors. The user application should subsequently discard these frames by setting the RX_ERR_DISC bit in the command_config register to 1.

## 4.1.8. 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 is a stream of six consecutive 0xFF bytes followed by a sequence of 16 consecutive unicast MAC addresses. The unicast address is the address 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.

#### 4.1.8.1. Sleep Mode

You can only put a node to sleep (set SLEEP bit in the `command_config` register to 1 and deassert the `magic_sleep_n` signal) if magic packet detection is enabled (set the `MAGIC_ENA` bit in the `command_config` register to 1).

Intel recommends that you do not put a node to sleep if you disable magic packet detection.

Network transmission is disabled when a node is put to sleep. The receiver remains enabled, but it ignores all traffic from the line except magic packets to allow a remote agent to wake up the node. In the sleep mode, only `etherStatsPkts` and `etherStatsOctets` count the traffic statistics.

#### 4.1.8.2. Magic Packet Detection

Magic packet detection wakes up a node that was put to sleep. The MAC function detects magic packets with any of the following destination addresses:

- Any multicast address
- A broadcast address
- The primary MAC address configured in the `mac_0` and `mac_1` registers
- Any of the supplementary 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 the MAC function detects a magic packet, the `WAKEUP` bit in the `command_config` register is set to 1, and the `etherStatsPkts` and `etherStatsOctets` statistics registers are incremented.

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

### 4.1.9. MAC Local Loopback

You can enable local loopback on the MII/GMII/RGMII of the MAC function to exercise the 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 local loopback:

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.

## 4.1.10. MAC Error Correction Code (ECC)

The error correction code (ECC) feature is implemented to the memory instances in the IP core. This feature is capable of detecting single and double bit errors, and can fix single bit errors in the corrupted data.

*Note:* This feature is only applicable for Arria V GZ, Stratix V, and Intel Arria 10 devices.

**Table 31. Core Variation and ECC Protection Support**

| Core Variation | ECC Protection Support |
|---|---|
| 10/100/1000 Mb Ethernet MAC | Protects the following options:<br>transmit and receive FIFO buffer<br>Retransmit buffer (if half duplex is enabled)<br>Statistic counters (if enabled)<br>Multicast hashtable (if enabled) |
| 10/100/1000 Mb Ethernet MAC with 1000BASE-X/SGMII PCS | Protects the following options:<br>transmit and receive FIFO buffer<br>Retransmit buffer (if half duplex is enabled)<br>Statistic counters (if enabled)<br>Multicast hashtable (if enabled)<br>SGMII bridge (if enabled) |
| 1000BASE-X/SGMII PCS only | Protects the SGMII bridge (if enabled) |
| 1000 Mb Small MAC | Protects the transmit and receive FIFO buffer |
| 10/100 Mb Small MAC | Protects the following options:<br>transmit and receive FIFO buffer<br>Retransmit buffer (if half duplex is enabled) |

When you enable this feature, the following output ports are added for 10/100/1000 Mb Ethernet MAC and 1000BASE-X/SGMII PCS variants to provide ECC status of all the memory instances in the IP core.

- Single channel core configuration—`eccstatus[1:0]` output ports.
- Multi-channel core configuration—`eccstatus_<n>[1:0]` output ports, where `eccstatus_0[1:0]` is for channel 0, eccstatus_1[1:0] for channel 1, and so on.

## 4.1.11. MAC Reset

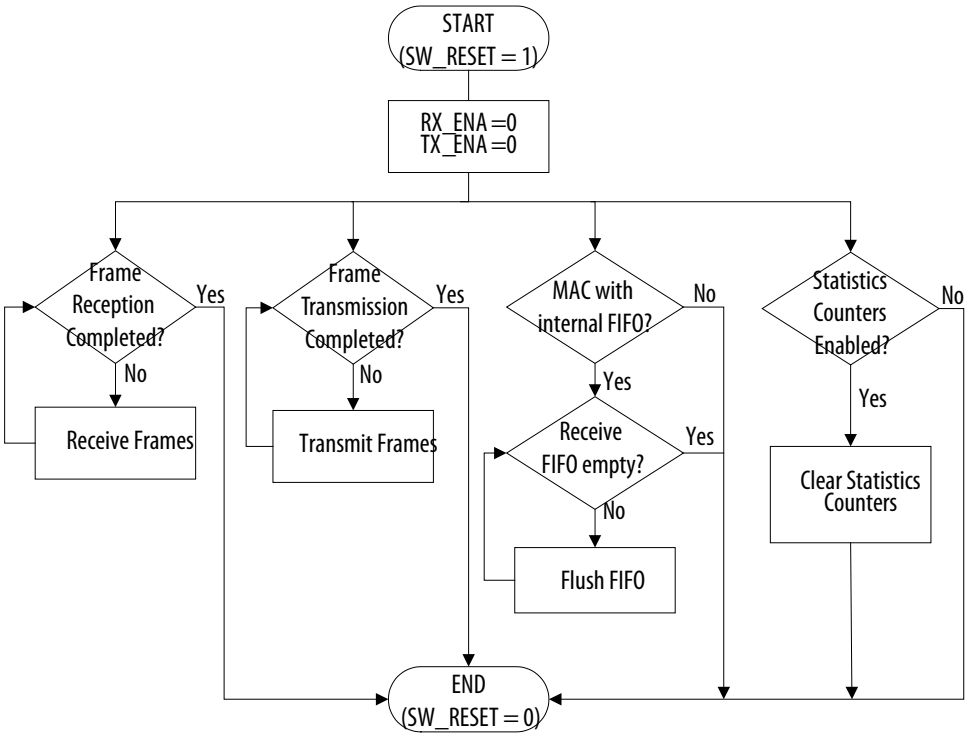
A hardware reset resets all logic. A software reset only disables the transmit and receive paths, clears all statistics registers, and flushes the receive FIFO buffer. The values of configuration registers, such as the MAC address and thresholds of the FIFO buffers, are preserved during a software reset.

When you trigger a software reset, the MAC function sets the TX_ENA and RX_ENA bits in the `command_config` register to 0 to disable the transmit and receive paths. However, the transmit and receive paths are only disabled when the current frame transmission and reception complete.

- To trigger a hardware reset, assert the `reset` signal.
- To trigger a software reset, set the SW_RESET bit in the `command_config` register to 1. The SW_RESET bit is cleared automatically when the software reset ends.

Intel recommends that you perform a software reset and wait for the software reset sequence to complete before changing the MAC operating speed and mode (full/half duplex). If you want to change the operating speed or mode without changing other configurations, preserve the `command_config` register before performing the software reset and restore the register after the changing the MAC operating speed or mode.

**Figure 18. Software Reset Sequence**




*Note:* 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.

## 4.1.12. PHY Management (MDIO)

This module implements the standard MDIO specification, IEEE 803.2 standard Clause 22, to access the PHY device management registers, and supports up to 32 PHY devices.

To access each PHY device, write the PHY address to the MDIO register (`mdio_addr0/1`) followed by the transaction data (MDIO Space 0/1). 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. You can access the MDIO registers via the Avalon Memory-Mapped interface.

For more information about the registers of a PHY device, refer to the specification provided with the device.

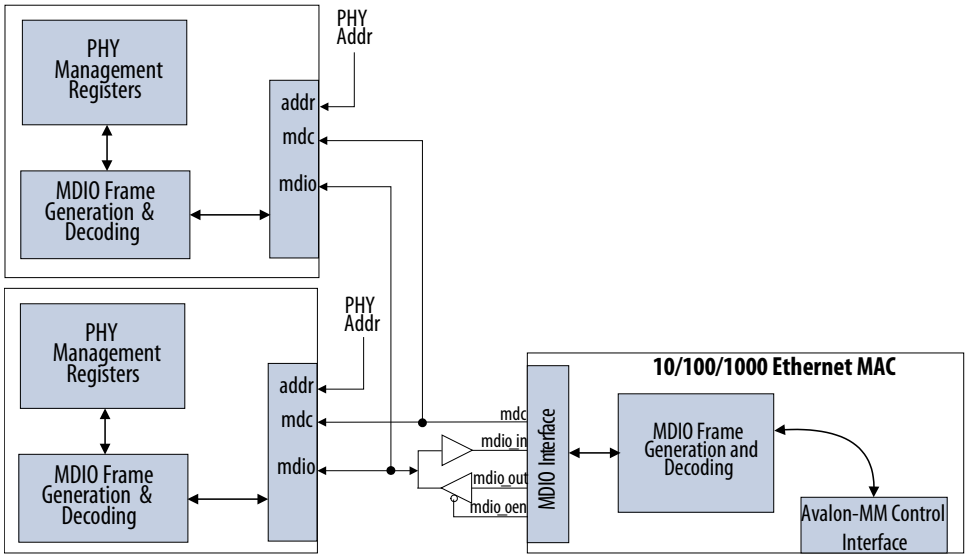
For more information about the MDIO registers, refer to MAC Configuration Register Space on page 81.

**Related Information**

MAC Configuration Register Space on page 81

### 4.1.12.1. MDIO Connection

**Figure 19. MDIO Interface**




### 4.1.12.2. MDIO Frame Format

The MDIO master communicates with the slave PHY device using MDIO frames. 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, `mdc`.

### Table 32. MDIO Frame Formats (Read/Write)

Field settings for MDIO transactions.

| Type | PRE | Command | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | ST<br>MSB LSB | OP<br>MSB LSB | Addr1<br>MSB LSB | Addr2<br>MSB LSB | TA | Data<br>MSB LSB | Idle |
| Read | 1 ... 1 | 01 | 10 | xxxxx | xxxxx | Z0 | xxxxxxxxxxxxxxxx | Z |
| Write | 1 ... 1 | 01 | 01 | xxxxx | xxxxx | 10 | xxxxxxxxxxxxxxxx | Z |

### Table 33. 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 | Opcode. Defines the transaction type. |
| 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. |

## 4.1.13. Connecting MAC to External PHYs

The MAC function implements a flexible network interface—MII for 10/100-Mbps interfaces, RGMII or GMII for 1000-Mbps interfaces—that you can use in multiple applications. This section provides the guidelines for implementing the following network applications:

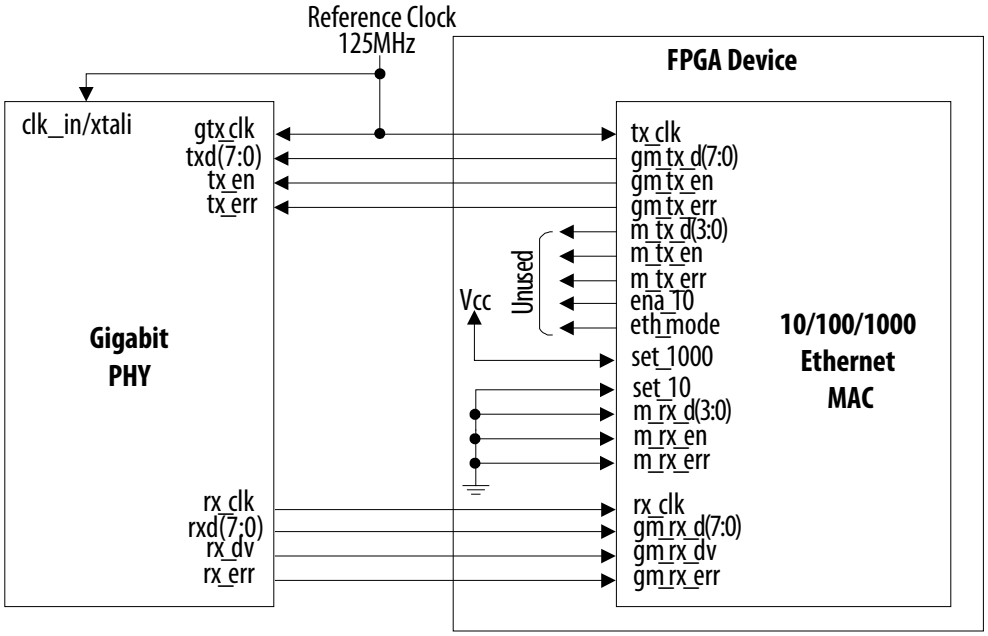
- Gigabit Ethernet operation
- Programmable 10/100 Ethernet operation
- Programmable 10/100/1000 Ethernet operation

### 4.1.13.1. Gigabit Ethernet

You can connect gigabit Ethernet PHYs to the MAC function via GMII or RGMII. On the receive path, connect the 125-MHz clock provided by the PHY device to the MAC clock, `rx_clk`. On transmit, drive a 125-MHz clock to the PHY GMII or RGMII. Connect a 125-MHz clock source to the MAC 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.

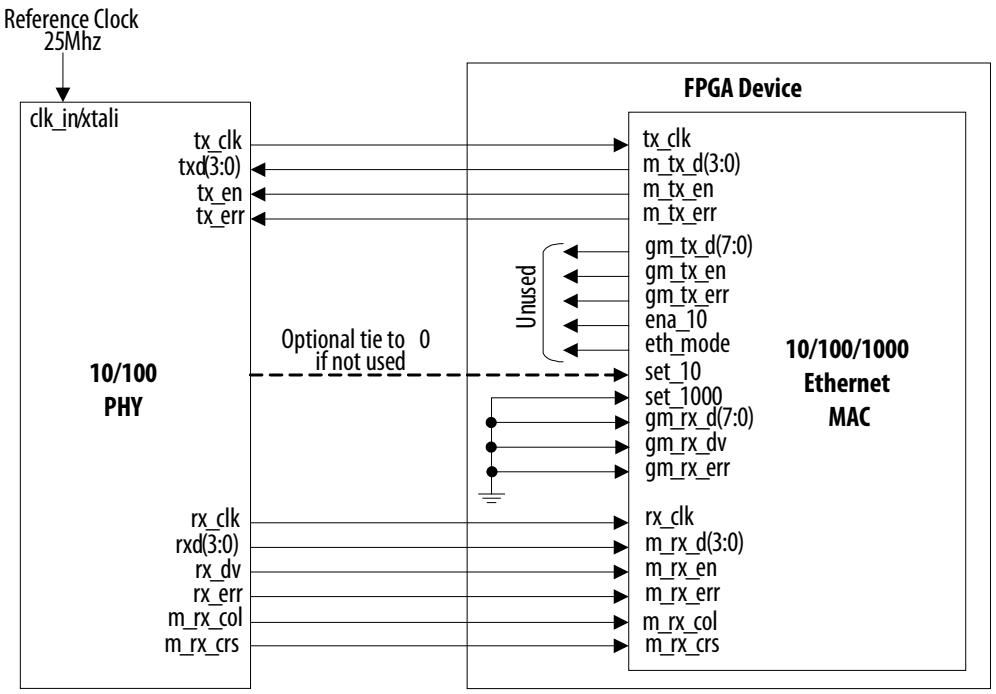
**Figure 20. Gigabit PHY to MAC via GMII**




### 4.1.13.2. Programmable 10/100 Ethernet

Connect 10/100 Ethernet PHYs to the MAC function via MII. On the receive path, connect the 25-MHz (100 Mbps) or 2.5-MHz (10 Mbps) clock provided by the PHY device to the MAC clock, `rx_clk`. On the transmit path, connect the 25 MHz (100 Mbps) or a 2.5 MHz (10 Mbps) clock provided by the PHY to the MAC clock, `tx_clk`.

**Figure 21. 10/100 PHY Interface**




### 4.1.13.3. Programmable 10/100/1000 Ethernet Operation

Typically, 10/100/1000 Ethernet PHY devices implement a shared interface that you connect to a 10/100-Mbps MAC via MII/RGMII or to a gigabit MAC via GMII/RGMII.

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

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. In gigabit mode, the PHY device expects a 125-MHz clock from the MAC function. 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, the MAC control signal `eth_mode` selects either MII or GMII. The MAC function asserts the `eth_mode` signal when the MAC function operates in gigabit mode, which subsequently drives the MAC GMII to the PHY interface. The `eth_mode` signal is deasserted when the MAC function operates in 10/100 mode. In this mode, the MAC MII is driven to the PHY interface.

**Figure 22. 10/100/1000 PHY Interface via MII/GMII**

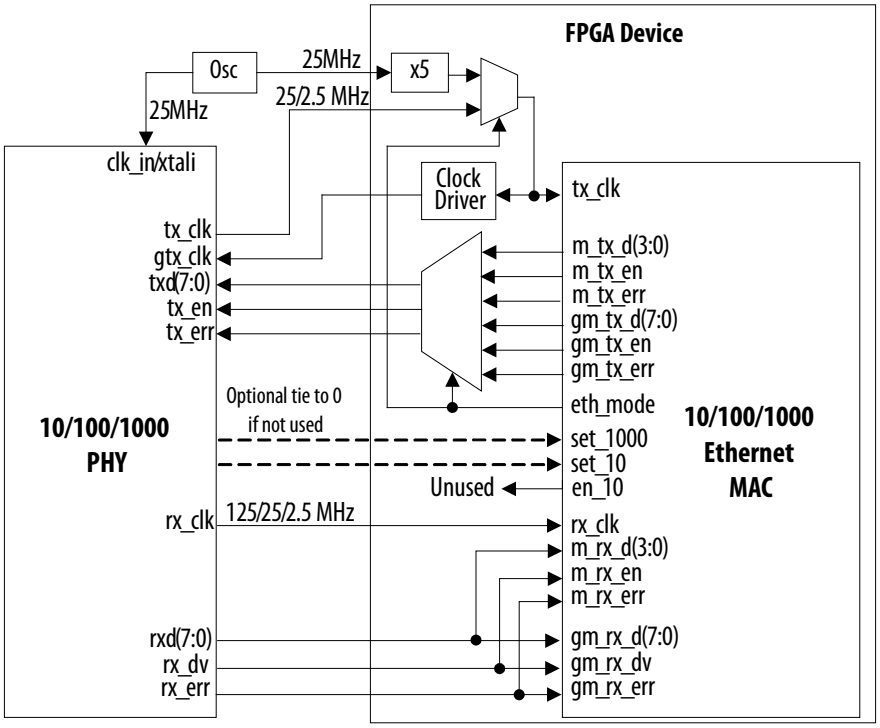
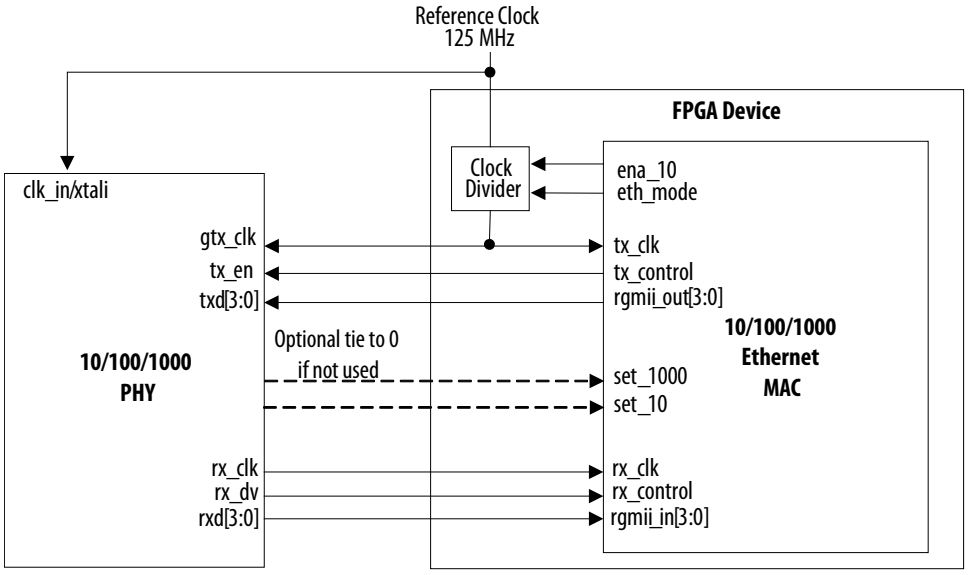



**Figure 23. 10/100/1000 PHY Interface via RGMII**



## 4.2. 1000BASE-X/SGMII PCS With Optional Embedded PMA

The Intel FPGA 1000BASE-X/SGMII PCS function implements the functionality specified by IEEE 802.3 Clause 36. It does not support carrier extension, frame bursting, and low power idle features. The PCS function is accessible via MII (SGMII) or GMII (1000BASE-X/SGMII). The PCS function interfaces to an on- or off-chip SERDES component via the industry standard ten-bit interface (TBI).

You can configure the PCS function to include an embedded physical medium attachment (PMA) with a a serial transceiver or LVDS I/O and soft CDR. The PMA interoperates with an external physical medium dependent (PMD) device, which drives the external copper or fiber network. The interconnect between Intel FPGA and PMD devices can be TBI or 1.25 Gbps serial.

The PCS function supports the following external PHYs:

- 1000 BASE-X PHYs as is.
- 10BASE-T, 100BASE-T and 1000BASE-T PHYs if the PHYs support SGMII.

### 4.2.1. 1000BASE-X/SGMII PCS Architecture

**Figure 24. 1000BASE-X/SGMII PCS**

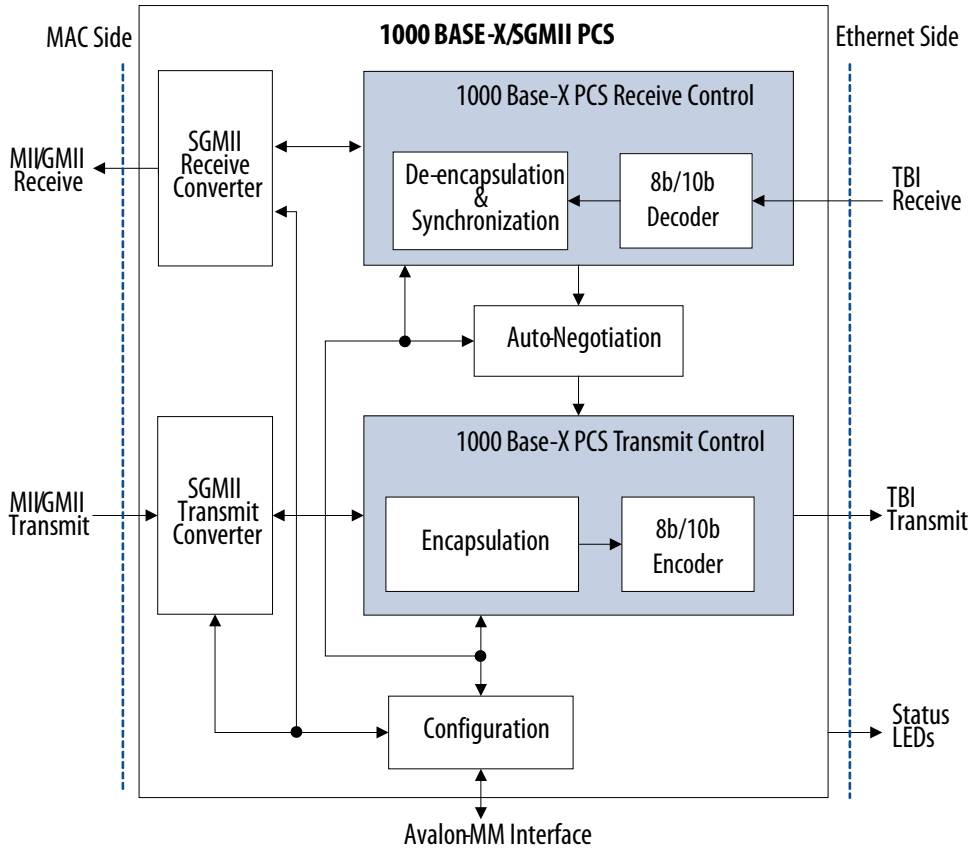


**Figure 25. 1000BASE-X/SGMII 2XTBI PCS**

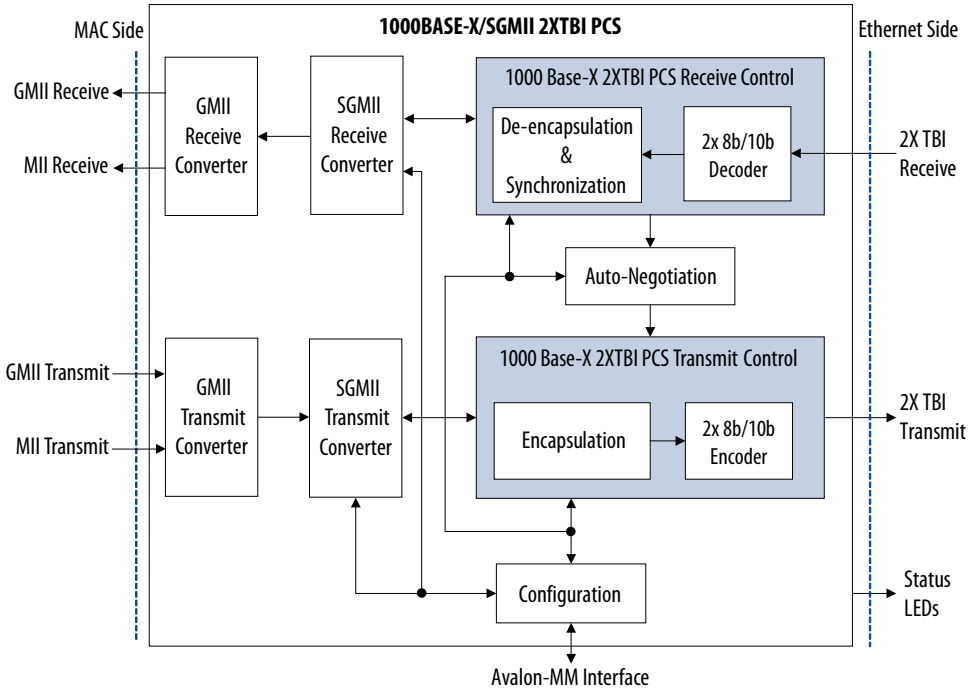



**Figure 26. 1000BASE-X/SGMII PCS with Embedded PMA**

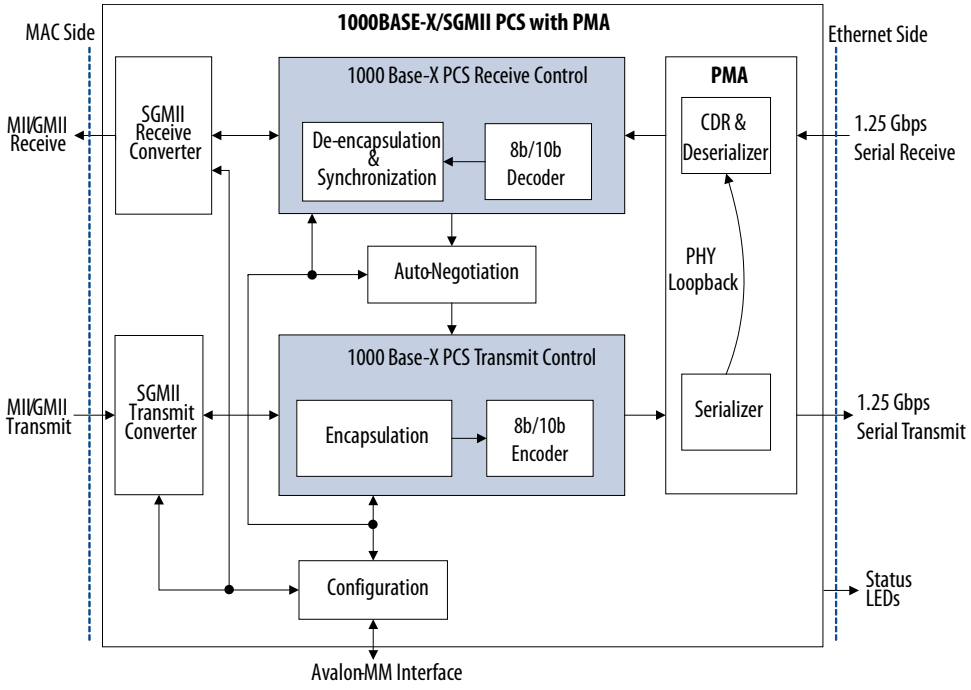
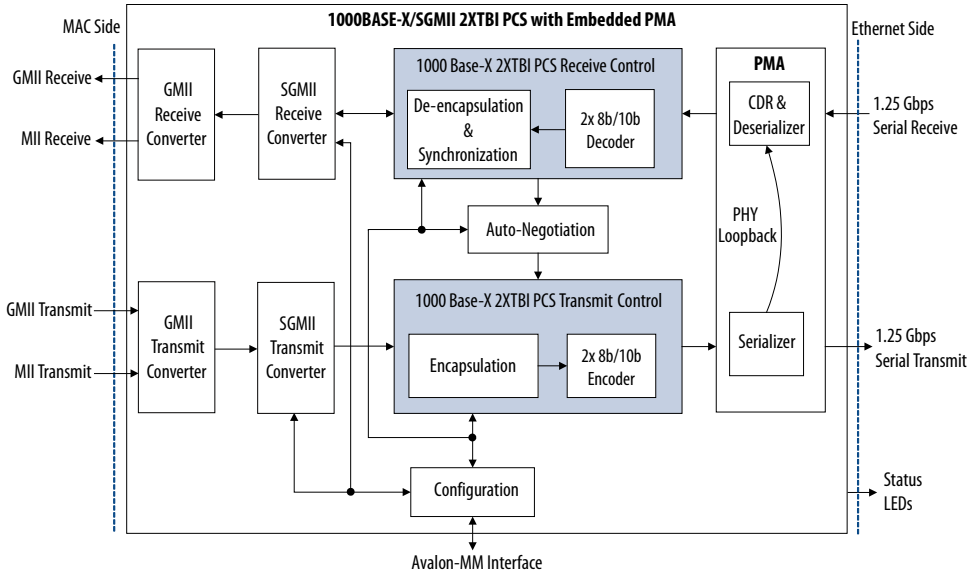


**Figure 27. 1000BASE-X/SGMII 2XTBI PCS with Embedded PMA**




## 4.2.2. Transmit Operation

The transmit operation includes frame encapsulation and encoding.

### 4.2.2.1. Frame Encapsulation

The PCS function replaces the first preamble byte in the MAC frame with the start of frame /S/ symbol. Then, the PCS function encodes the rest of the bytes in the MAC frame with standard 8B/10B encoded characters. After the last FCS byte, the PCS function inserts the end of frame sequence, /T/ /R/ /R/ or /T/ /R/, depending on the number of character transmitted. Between frames, the PCS function transmits /I/ symbols.

If the PCS function receives a frame from the MAC function with an error (`gm_tx_err` asserted during frame transmission), the PCS function encodes the error by inserting a /V/ character.

### 4.2.2.2. 8b/10b Encoding

The 8B/10B encoder maps 8-bit words to 10-bit symbols to generate a DC balance and ensure disparity of the stream with a maximum run length of 5.

## 4.2.3. Receive Operation

The receive operation includes comma detection, decoding, de-encapsulation, synchronization, and carrier sense.

#### 4.2.3.1. Comma Detection

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 PCS function realigns the data stream on a valid 10-bit character boundary. A standard 8b/10b decoder can subsequently decodes the aligned stream.

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

#### 4.2.3.2. 8b/10b Decoding

The 8b/10b decoder performs the disparity checking to ensure DC balancing and produces a decoded 8-bit stream of data for the frame de-encapsulation function.

#### 4.2.3.3. Frame De-encapsulation

The frame de-encapsulation state machine detects the start of frame when the /I/ /S/ sequence is received and replaces the /S/ with a preamble byte (0x55). It continues decoding the frame bytes and transmits them to the MAC function. The /T/ /R/ /R/ or the /T/ /R/ sequence is decoded as an end of frame.

A /V/ character is decoded and sent to the MAC function as frame error. The state machine decodes sequences other than /I/ /I/ (Idle) or /I/ /S/ (Start of Frame) as wrong carrier.

During frame reception, the de-encapsulation state machine checks for invalid characters. When the state machine detects invalid characters, it indicates an error to the MAC function.

#### 4.2.3.4. 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 the state machine receives three code groups with comma consecutively without error.

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

The PCS function drives the `led_panel_link` signal to 1 when link synchronization is acquired for the PCS operating in 1000 Base-X without auto negotiation and SGMII mode without auto negotiation.

#### 4.2.3.5. 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 `mii_rx_crs` and `led_crs` signals to 1 when it detects an activity. The `led_crs` signal can be used as a common visual activity check using a board LED.

#### 4.2.3.6. Collision Detection

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

When a collision happens, the collision detection state machine drives the `mii_rx_col` and `led_col` signals to 1. You can use the `led_col` signal as a visual check using a board LED.

### 4.2.4. Transmit and Receive Latencies

Intel uses the following definitions for the transmit and receive latencies for the PCS function with an embedded PMA:

- Transmit latency is the time the PCS function takes to transmit the first bit on the PMA-PCS interface after the bit was first available on the MAC side interface (MII/GMII).
- Receive latency is the time the PCS function takes to present the first bit on the MAC side interface (MII/GMII) after the bit was received on the PMA-PCS interface.

**Table 34. PCS Transmit and Receive Latency**

For GXB, the TX latencies are obtained from `sim:/tb/dut/gmii_tx_d` or `sim:/tb/dut/mii_tx_d` (after `clkena` is asserted) to `sim:/tb/dut/i_tse_pcs_0/tx_frame`. The RX latencies are obtained from `sim:/tb/dut/gmii_rx_d` or `sim:/tb/dut/mii_rx_d` to `sim:/tb/dut/i_tse_pcs_0/tx_frame`.

For LVDS, the TX latencies are obtained from the TX latencies are obtained from `sim:/tb/dut/gmii_tx_d` or `sim:/tb/dut/mii_tx_d` (after `clkena` is asserted) to `sim:/tb/dut/i_tse_pcs_0/tbi_tx_d_muxed`. The RX latencies are obtained from `sim:/tb/dut/gmii_rx_d` or `sim:/tb/dut/mii_rx_d` to `sim:/tb/dut/i_tse_pcs_0/tbi_rx_d_lvds`.

For 2XTBI PCS variant, the TX latencies are obtained from `sim:/tb/gmii_tx_d` to `sim:/tb/tbi2x_tx_d`. The RX latencies are obtained from `sim:/tb/tbi2x_rx_d to sim:/tb/gmii_rx_d`.

For 2XTBI PCS with GXB variant, the TX latencies are obtained from `sim:/tb/dut/eth_tse_0_testbench/i_tse_pcs_0/gmii_tx_d` to `sim:/tb/dut/eth_tse_0_testbench/i_tse_pcs_0/tbi2x_tx_d`. The RX latencies are obtained from `sim:/tb/dut/eth_tse_0_testbench/i_tse_pcs_0/tbi2x_rx_d` to `sim:/tb/dut/eth_tse_0_testbench/i_tse_pcs_0/gmii_rx_d`.

| PCS Configuration | Latency (ns) | |
|---|---|---|
| | Transmit | Receive |
| **Stratix IV** | | |
| 10-Mbps SGMII PCS with GXB | 3456 | 1454.85 |
| 100-Mbps SGMII PCS with GXB | 376 | 214.8 |
| 1000-Mbps SGMII PCS with GXB | 104 | 142.8 |
| 1000BASE-X with GXB | 8 | 48 |
| 10-Mbps SGMII PCS with LVDS I/O | 3064 | 1720 |

*continued...*

| PCS Configuration | Latency (ns) | |
|---|---|---|
| | Transmit | Receive |
| 100-Mbps SGMII PCS with LVDS I/O | 384 | 280 |
| 1000-Mbps SGMII PCS with LVDS I/O | 136 | 192 |
| 1000BASE-X PCS with LVDS I/O | 40 | 96 |
| **Intel Arria 10** | | |
| 10-Mbps SGMII PCS with GXB | 3600 | 1867.65 |
| 100-Mbps SGMII PCS with GXB | 360 | 187.65 |
| 1000-Mbps SGMII PCS with GXB | 104 | 139.65 |
| 1000BASE-X with GXB | 8 | 48 |
| 10-Mbps SGMII PCS with LVDS I/O | 3208 | 1176 |
| 100-Mbps SGMII PCS with LVDS I/O | 368 | 256 |
| 1000-Mbps SGMII PCS with LVDS I/O | 136 | 192 |
| 1000BASE-X PCS with LVDS I/O | 40 | 96 |
| **Intel Cyclone 10 GX** | | |
| 10-Mbps SGMII PCS with GXB | 3600 | 1867.65 |
| 100-Mbps SGMII PCS with GXB | 360 | 187.65 |
| 1000-Mbps SGMII PCS with GXB | 104 | 139.65 |
| 1000BASE-X with GXB | 8 | 48 |
| 10-Mbps SGMII PCS with LVDS I/O | 3208 | 1176 |
| 100-Mbps SGMII PCS with LVDS I/O | 368 | 256 |
| 1000-Mbps SGMII PCS with LVDS I/O | 136 | 192 |
| 1000BASE-X PCS with LVDS I/O | 40 | 96 |
| **Intel Stratix 10** | | |
| 10-Mbps SGMII PCS with LVDS I/O | 3336 | 1840 |
| 100-Mbps SGMII PCS with LVDS I/O | 456 | 280 |
| 1000-Mbps SGMII PCS with LVDS I/O | 112 | 208 |
| 1000BASE-X PCS with LVDS I/O with no Enable SGMII | 40 | 104 |
| **Intel Stratix 10 E-tile** | | |
| 10-Mbps SGMII 2XTBI PCS | 4872 | 6328 |
| 100-Mbps SGMII 2XTBI PCS | 752 | 968 |
| 1000-Mbps SGMII 2XTBI PCS | 300 | 416 |
| 1000BASE-X 2XTBI PCS without enabling SGMII | 300 | 416 |
| 10-Mbps SGMII 2XTBI PCS with GXB | 6466.401 | 6160.280 |
| 100-Mbps SGMII 2XTBI PCS with GXB | 906.401 | 880.272 |
| 1000-Mbps SGMII 2XTBI PCS with GXB | 306.401 | 424.281 |
| 1000BASE-X 2XTBI PCS with GXB without enabling SGMII | 306.401 | 424.281 |

*continued...*

| PCS Configuration | Latency (ns) | |
|---|---|---|
| | Transmit | Receive |
| **Intel Agilex** | | |
| 10-Mbps SGMII PCS with LVDS I/O | 2512 | 2512 |
| 100-Mbps SGMII PCS with LVDS I/O | 352 | 232 |
| 1000-Mbps SGMII PCS with LVDS I/O | 116 | 192 |
| 1000BASE-X PCS with LVDS I/O without enabling SGMII | 44 | 88 |
| 10-Mbps SGMII 2XTBI PCS | 4872 | 6328 |
| 100-Mbps SGMII 2XTBI PCS | 752 | 968 |
| 1000-Mbps SGMII 2XTBI PCS | 300 | 416 |
| 1000BASE-X 2XTBI PCS without enabling SGMII | 300 | 416 |
| 10-Mbps SGMII 2XTBI PCS with GXB | 6466.401 | 6160.280 |
| 100-Mbps SGMII 2XTBI PCS with GXB | 906.401 | 880.272 |
| 1000-Mbps SGMII 2XTBI PCS with GXB | 306.401 | 424.281 |
| 1000BASE-X 2XTBI PCS with GXB without enabling SGMII | 306.401 | 424.281 |

## 4.2.5. GMII Converter

- The GMII transmit converter converts the GMII (1000Mbps)/MII (10/100 Mbps) datapath to 16-bit GMII data.
- The GMII receive converter converts the 16-bit GMII data to GMII (1000 Mbps)/MII (10/100 Mbps).

## 4.2.6. SGMII Converter

You can configure the PCS layer to use SGMII mode or 1000BASE-X mode. To configure to SGMII mode, enable the SGMII converter by setting the SGMII_ENA bit in the `if_mode` register to 1. When enabled and 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, Intel recommends that you configure the SGMII converter with the SGMII_SPEED bits in the `if_mode` register.

In 1000BASE-X mode, the PCS function always operates in gigabit mode and data duplication is disabled.

### 4.2.6.1. Transmit

In gigabit mode, the PCS and MAC functions must 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.

#### 4.2.6.2. Receive

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

### 4.2.7. 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.

When simulating your design, you can disable auto-negotiation to reduce the simulation time. If you enable auto-negotiation in your design, set the `link_timer` time to a smaller value to reduce the auto-negotiation link timer in the simulation.

**Related Information**

PCS Configuration Register Space on page 93

#### 4.2.7.1. 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.

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.

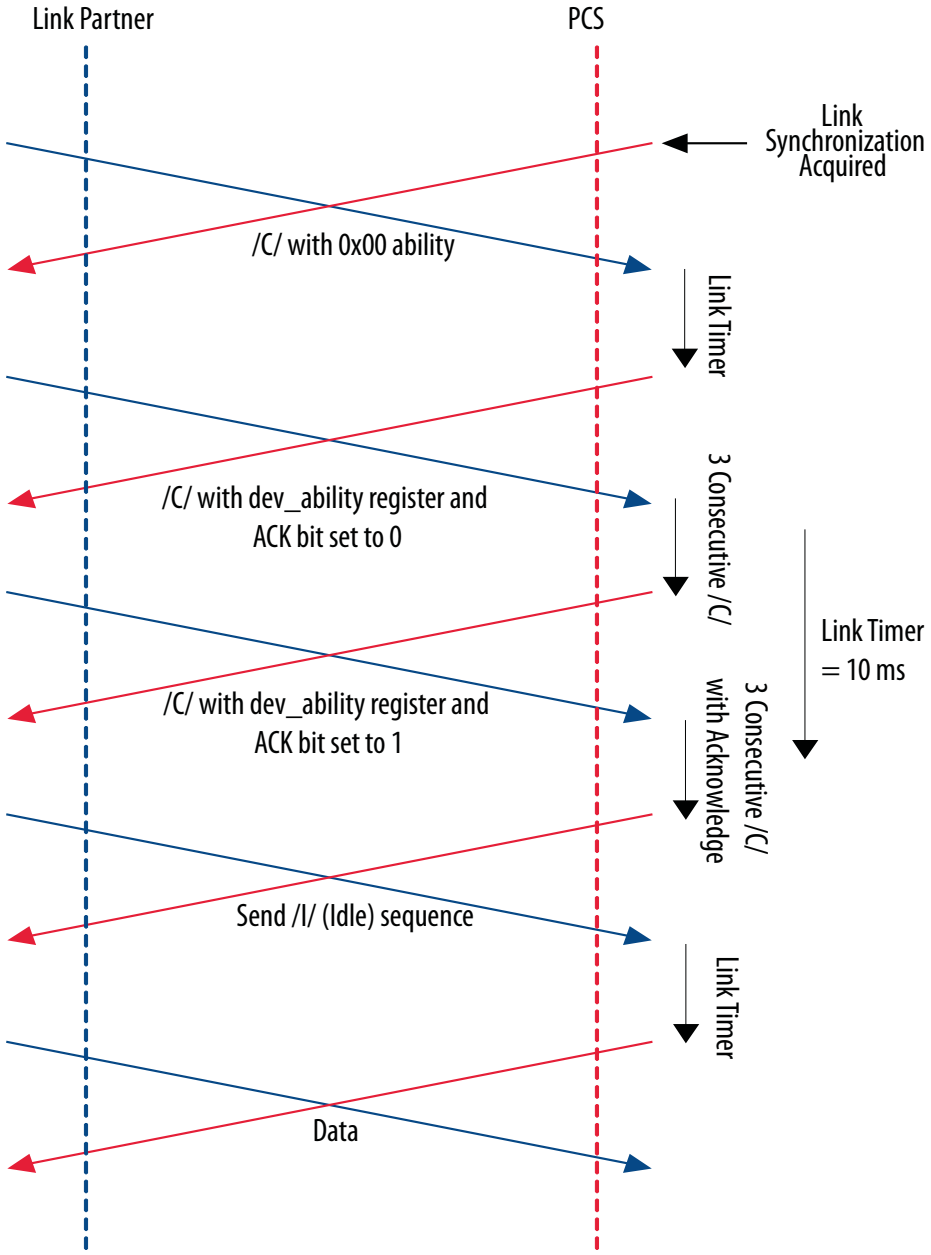
The auto-negotiation state machine then sets the ACK bit to 1 in the advertised `dev_ability` register and checks if three consecutive /C/ sequences are received from the link partner with the ACK bit set to 1.

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.

After auto-negotiation completes successfully, the user software reads both the `dev_ability` and `partner_ability` register and proceed to resolve priority for duplex mode and pause mode. If the design contains a MAC and PCS, the user software configures the MAC with a proper resolved pause mode by setting the

PAUSE_IGNORE bit in command_config register. To disable pause frame generation based on the receive FIFO buffer level, you should set the rx_section_empty register accordingly.

**Figure 28. Auto-Negotiation Activity (Simplified)**




Once auto-negotiation completes successfully, the ability advertised by the link partner device is available in the partner_ability register and the AUTO_NEGOTIATION_COMPLETE bit in the status register is set to 1.

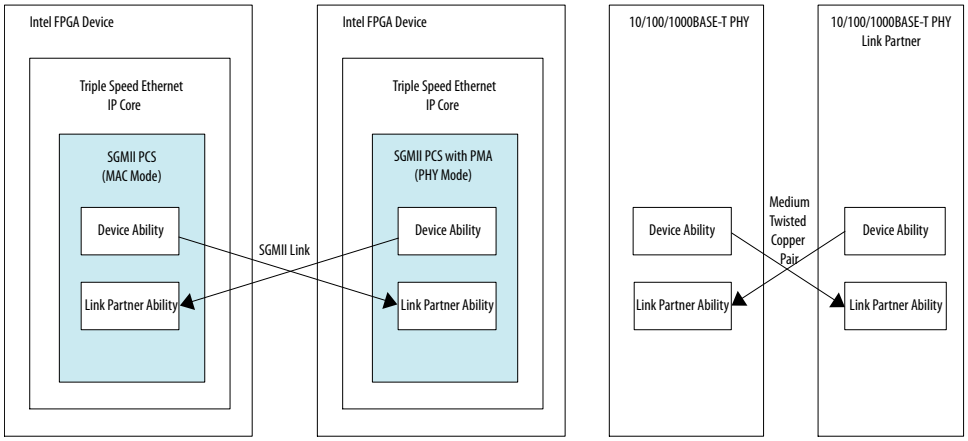
The PCS function restarts auto-negotiation when link synchronization is lost and reacquired, or when you set the RESTART_AUTO_NEGOTIATION bit in the PCS control register to 1.

### 4.2.7.2. SGMII Auto-Negotiation

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

Possible application of SGMII auto-negotiation in MAC mode and PHY mode.

**Figure 29. SGMII Auto-Negotiation in MAC Mode and PHY Mode**

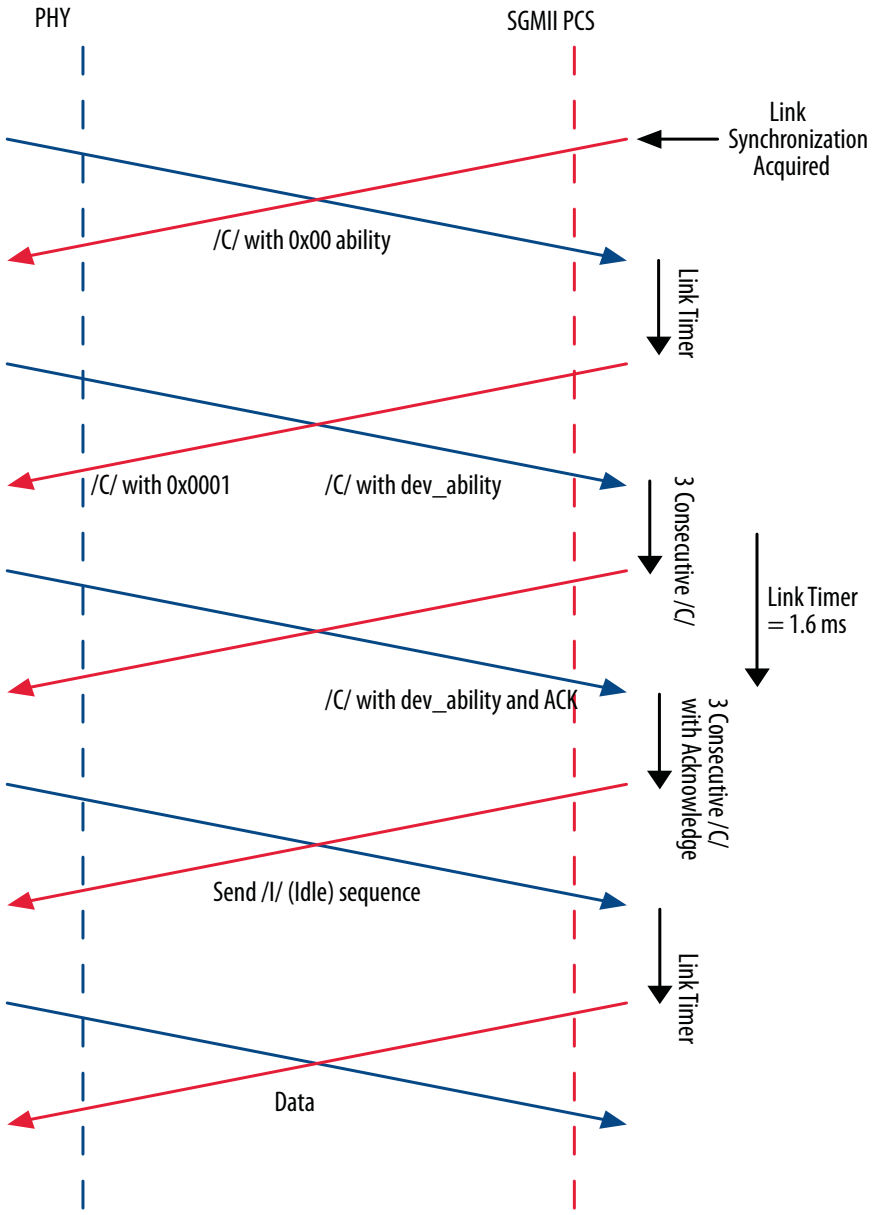



If the SGMII_ENA and USE_SGMII_AN bits in the if_mode register are 1, the PCS function is automatically configured with the capabilities advertised by the PHY device once the auto-negotiation completes.

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.

If the SGMII_ENA bit is 1 and the SGMII_AN_MODE bit is 1 (SGMII PHY Mode auto-negotiation is enabled) the speed and duplex mode resolution will be resolved based on the value that you set in the dev_ability register once auto negotiation is done. You should use set to the PHY mode if you want to advertise the link speed and duplex mode to the link partner.

**Figure 30. SGMII Auto-Negotiation Activity**



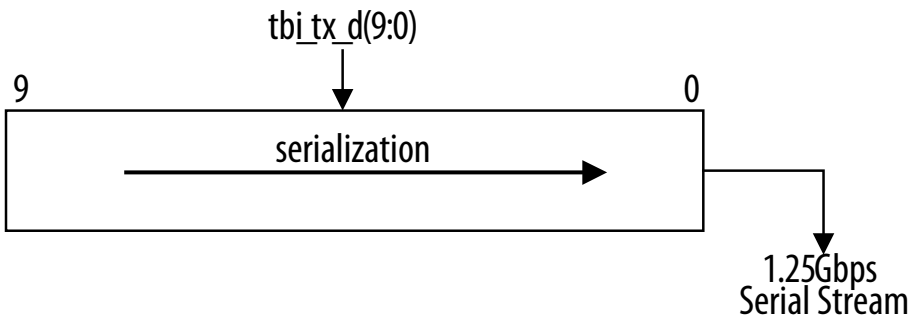

For more information, refer to *CISCO Serial-GMII Specifications*.

## 4.2.8. Ten-bit Interface

In PCS variations with embedded PMA, the PCS function implements a TBI to an external SERDES.

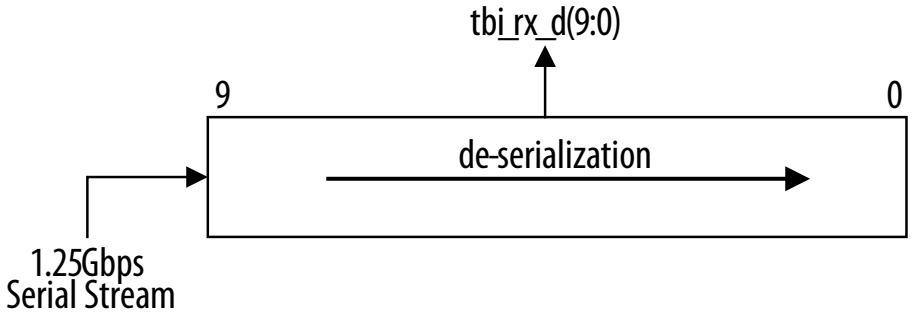
On transmit, the SERDES must serialize `tbi_tx_d[0]`, the least significant bit of the TBI output bus first and `tbi_tx_d[9]`, the most significant bit of the TBI output bus last to ensure the remote node receives the data correctly, as figure below illustrates.

**Figure 31. SERDES Serialization Overview**




On receive, the SERDES must serialize the TBI least significant bit first and the TBI most significant bit last, as figure below illustrates.

**Figure 32. SERDES De-Serialization Overview**

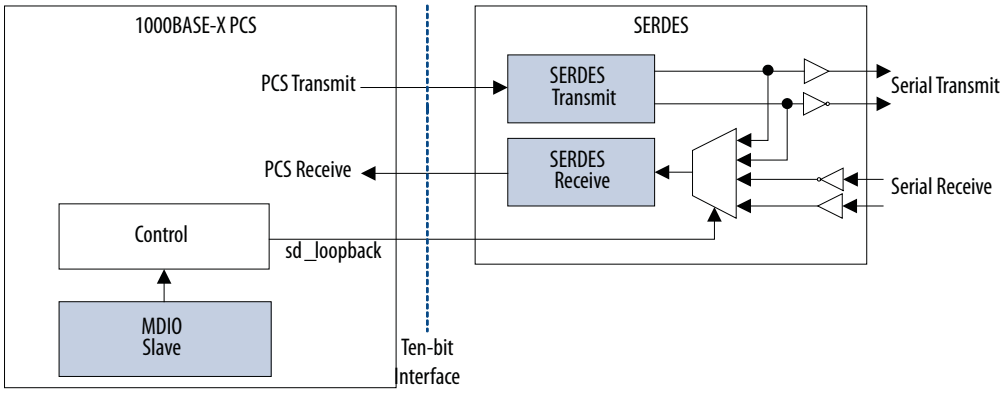



## 4.2.9. PHY Loopback

In PCS variations with embedded PMA targeting devices with GX transceivers, you can enable loopback on the serial interface to test the PCS and embedded PMA functions in isolation of the PMD. To enable loopback, set the `sd_loopback` bit in the PCS `control` register to 1.

The serial loopback option is not supported in Cyclone IV devices with GX transceiver.

**Figure 33. Serial Loopback**

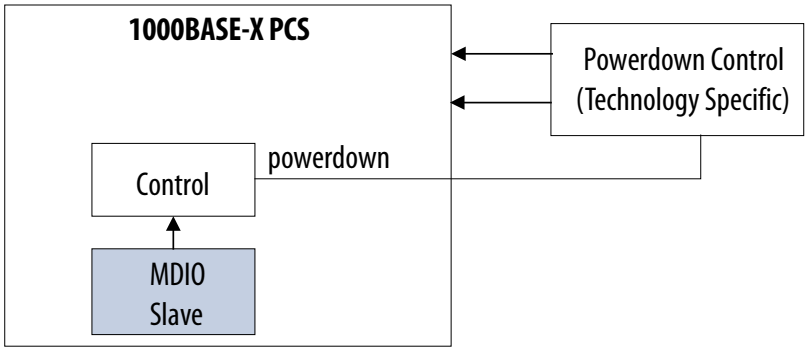


## 4.2.10. PHY Power-Down

Power-down is controlled by the POWERDOWN bit in the PCS `control` register. When the system management agent enables 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 34. Power-Down**

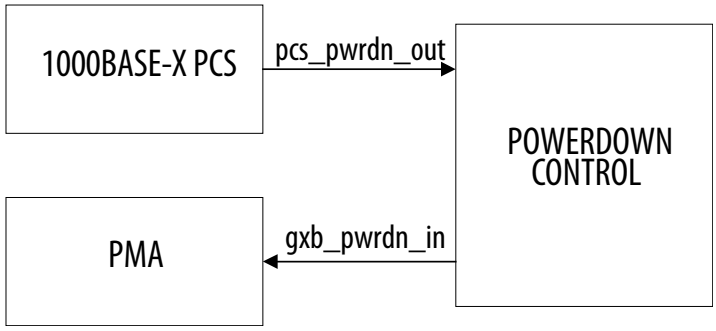



### 4.2.10.1. Power-Down in PCS Variations with Embedded PMA

In PCS variations with embedded PMA targeting devices with GX transceivers, the power-down signal is internally connected to the power-down of the GX transceiver. In these devices, the power-down functionality is shared across quad-port transceiver blocks. Ethernet designs must share a common `gxb_pwrdn_in` signal to use the same quad-port transceiver block.

For designs targeting devices other than Stratix V, you can export the power-down signals to implement your own power-down logic to efficiently use the transceivers within a particular transceiver quad. Turn on the **Export transceiver powerdown signal** parameter to export the signals.

**Figure 35. Power-Down with Export Transceiver Power-Down Signal**

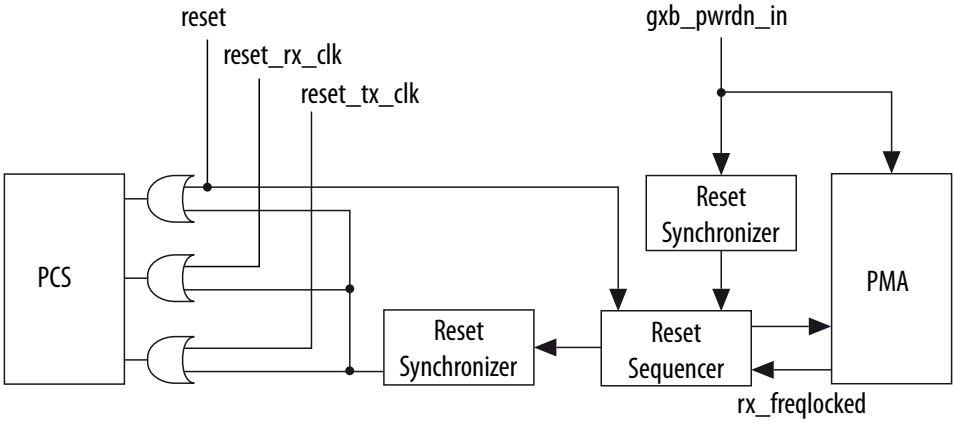


## 4.2.11. 1000BASE-X/SGMII PCS Reset

A hardware reset resets all logic synchronized to the respective clock domains whereas a software reset only resets the PCS state machines, comma detection function, and 8B10B encoder and decoder. To trigger a hardware reset on the PCS, assert the respective reset signals: `reset_reg_clk`, `reset_tx_clk`, and `reset_rx_clk`. To trigger a software reset, set the RESET bit in the `control` register to 1.

In PCS variations with embedded PMA, assert the respective reset signals or the power-down signal to trigger a hardware reset. You must assert the `reset` signal subsequent to asserting the `reset_rx_clk`, `reset_tx_clk`, or `gxb_pwrdn_in` signal. The reset sequence is also initiated when the active-low `rx_freqlocked` signal goes low.

**Figure 36. Reset Distribution in PCS with Embedded PMA**

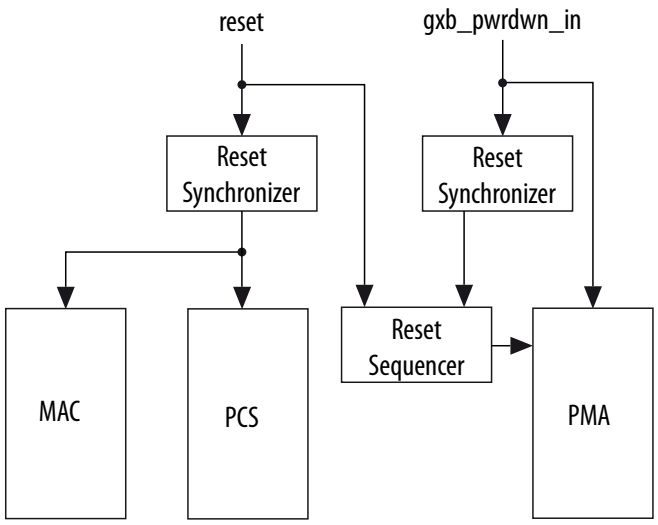



For more information about the `rx_freqlocked` signal and transceiver reset, refer to the transceiver handbook of the respective device family.

Assert the `reset` or `gxb_pwrdn_in` signals to perform a hardware reset on MAC with PCS and embedded PMA variation.

*Note:* You must assert the `reset` signal for at least three clock cycles.

**Figure 37. Reset Distribution in MAC with PCS and Embedded PMA**




## 4.2.12. Intel FPGA IEEE 1588v2 Feature

The Intel FPGA IEEE 1588v2 feature provides timestamp for receive and transmit frames in the Triple-Speed Ethernet IP core designs. The feature consists of Precision Time Protocol (PTP). PTP is a layer-3 protocol that accurately synchronizes all real time-of-day clocks in a network to a master clock.

### 4.2.12.1. IEEE 1588v2 Supported Configurations

The Triple-Speed Ethernet IP core supports the IEEE 1588v2 feature only in the following configurations:

- 10/100/1000-Mbps MAC with 1000BASE-X/SGMII PCS and embedded serial PMA without FIFO buffer in full-duplex mode
- 10/100/1000-Mbps MAC with 1000BASE-X/SGMII PCS and embedded LVDS I/O without FIFO buffer in full-duplex mode
- 10/100/1000-Mbps MAC without FIFO buffer in full-duplex mode

### 4.2.12.2. IEEE 1588v2 Features

- Supports 4 types of PTP clock on the transmit datapath:
  - Master and slave ordinary clock
  - Master and slave boundary clock
  - End-to-end (E2E) transparent clock
  - Peer-to-peer (P2P) transparent clock
- Supports PTP message types:
  - PTP event messages—Sync, Delay_Req, Pdelay_Req, and Pdelay_Resp.
  - PTP general messages—Follow_Up, Delay_Resp, Pdelay_Resp_Follow_Up, Announce, Management, and Signaling.
- Supports simultaneous 1-step and 2-step clock synchronizations on the transmit datapath.
  - 1-step clock synchronization—The MAC function inserts accurate timestamp in Sync PTP message or updates the correction field with residence time.
  - 2-step clock synchronization—The MAC function provides accurate timestamp and the related fingerprint for all PTP message.
- Supports the following PHY operating speed accuracy:
  - random error:
    - 10Mbps—NA
    - 100Mbps—timestamp accuracy of ± 5 ns
    - 1000Mbps—timestamp accuracy of ± 2 ns
  - static error—timestamp accuracy of ± 3 ns
- Supports IEEE 802.3, UDP/IPv4, and UDP/IPv6 transfer protocols for the PTP frames.
- Supports untagged, VLAN tagged, Stacked VLAN Tagged PTP frames, and any number of MPLS labels.
- Supports configurable register for timestamp correction on both transmit and receive datapaths.
- Supports Time-of-Day (ToD) clock that provides a stream of 64-bit and 96-bit timestamps.

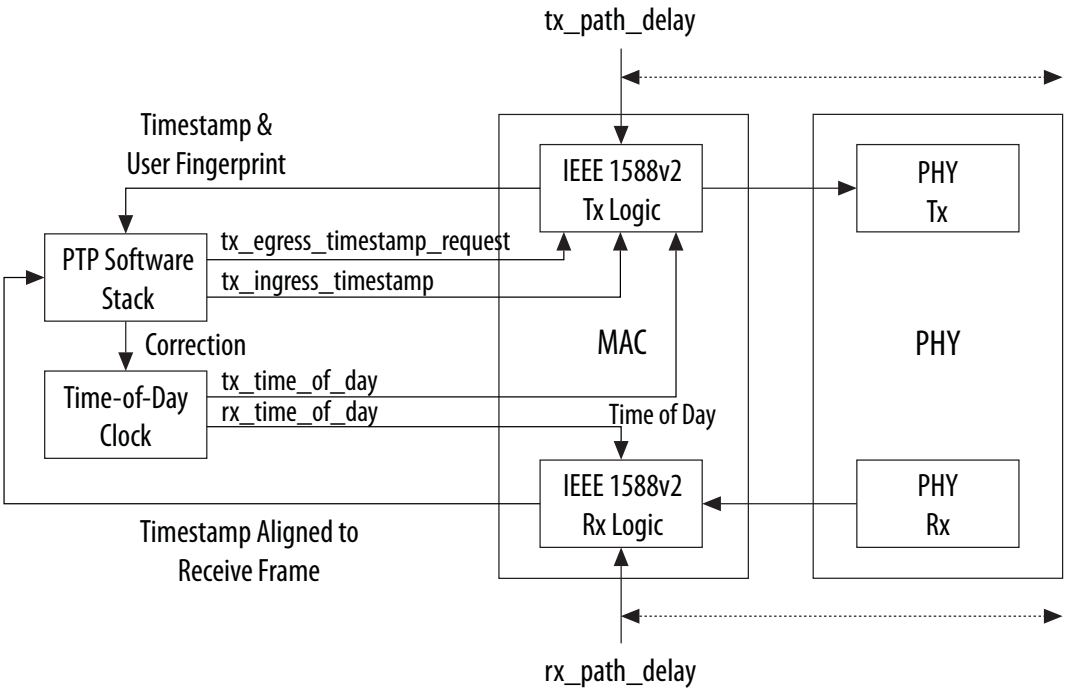
**Related Information**

Altera 1588 System Solution

### 4.2.12.3. IEEE 1588v2 Architecture

**Figure 38. Overview of the IEEE 1588v2 Feature**

This figure shows only the datapaths related to the IEEE 1588v2 feature.




### 4.2.12.4. IEEE 1588v2 Transmit Datapath

The IEEE 1588v2 feature supports 1-step and 2-step clock synchronizations on the transmit datapath.

- For 1-step clock synchronization:
  - Timestamp insertion depends on the PTP device and message type.
  - The MAC function inserts a timestamp in the Sync PTP message if the PTP clock operates as ordinary or boundary clock.
  - Depending on the PTP device and message type, the MAC function updates the residence time in the correction field of the PTP frame when the client asserts `tx_etstamp_ins_ctrl_residence_time_update`. The residence time is the difference between the egress and ingress timestamps.
  - For PTP frames encapsulated using the UDP/IPv6 protocol, the MAC function performs UDP checksum correction using extended bytes in the PTP frame.
  - The MAC function re-computes and re-inserts CRC-32 into the PTP frames after each timestamp or correction field insertion.
- For 2-step clock synchronization, the MAC function returns the timestamp and the associated fingerprint for all transmit frames when the client asserts `tx_egress_timestamp_request_valid`.

**Table 35. Timestamp and Correction Insertion for 1-Step Clock Synchronization**

This table summarizes the timestamp and correction field insertions for various PTP messages in different PTP clocks.

| PTP Message | Ordinary Clock | | Boundary Clock | | E2E Transparent Clock | | P2P Transparent Clock | |
|---|---|---|---|---|---|---|---|---|
| | Insert Time stamp | Insert Correction | Insert Time stamp | Insert Correction | Insert Time stamp | Insert Correction | Insert Time stamp | Insert Correction |
| Sync | Yes (1) | No | Yes (1) | No | No | Yes (2) | No | Yes (2) |
| Delay_Req | No | No | No | No | No | Yes (2) | No | Yes (2) |
| Pdelay_Req | No | No | No | No | No | Yes (2) | No | No |
| Pdelay_Resp | No | Yes (1), (2) | No | Yes (1), (2) | No | Yes (2) | No | Yes (1), (2) |
| Delay_Resp | No | No | No | No | No | No | No | No |
| Follow_Up | No | No | No | No | No | No | No | No |
| Pdelay_Resp_Follow_Up | No | No | No | No | No | No | No | No |
| Announce | No | No | No | No | No | No | No | No |
| Signaling | No | No | No | No | No | No | No | No |
| Management | No | No | No | No | No | No | No | No |

Notes to Table 35 on page 76 :

1. Applicable only when 2-step flag in `flagField` of the PTP frame is 0.
2. Applicable when you assert `tx_ingress_timestamp_request_valid`.

### 4.2.12.5. IEEE 1588v2 Receive Datapath

In the receive datapath, the IEEE 1588v2 feature provides a timestamp for all receive frames. The timestamp is aligned with the `avalon_st_rx_startofpacket` signal.

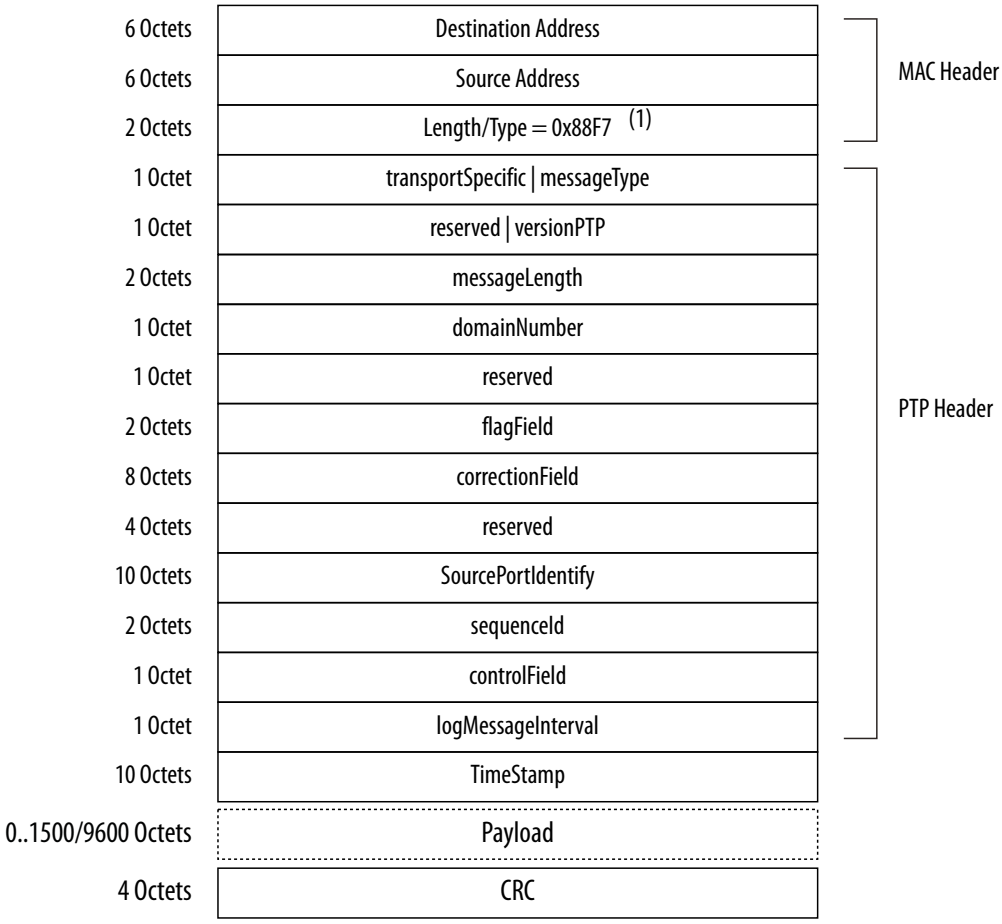
### 4.2.12.6. IEEE 1588v2 Frame Format

The MAC function, with the IEEE 1588v2 feature, supports PTP frame transfer for the following transport protocols:

- IEEE 802.3
- UDP/IPv4
- UDP/IPv6

### 4.2.12.6.1. PTP Frame in IEEE 802.3

**Figure 39. PTP Frame in IEEE 802.3**



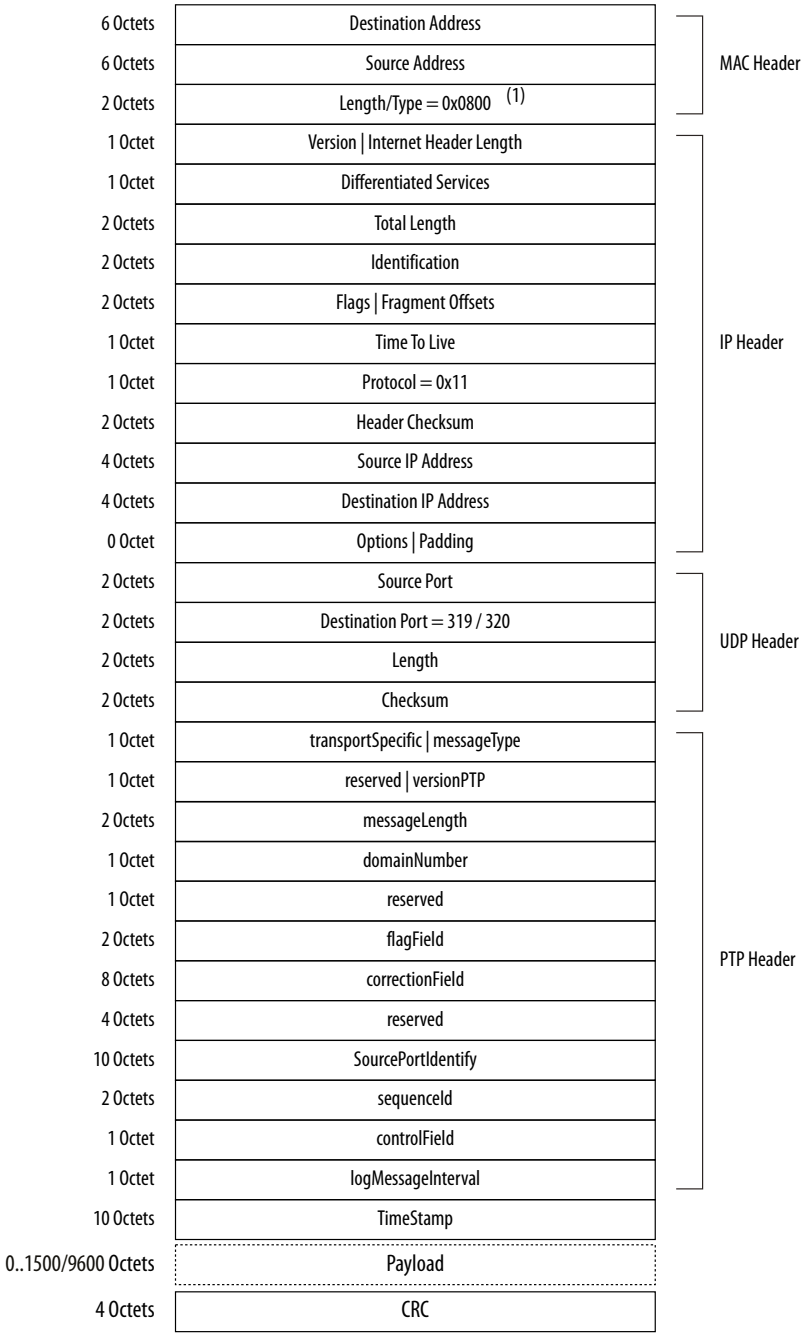

Note to Figure 39 on page 77 :

1. For frames with VLAN or Stacked VLAN tag, add 4 or 8 octets offsets before the length/type field.

### 4.2.12.6.2. PTP Frame over UDP/IPv4

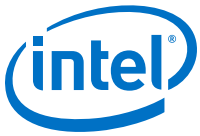
Checksum calculation is optional for the UDP/IPv4 protocol. The 1588v2 Tx logic should set the checksum to zero.

**Figure 40. PTP Frame over UDP/IPv4**




Note to Figure 40 on page 78 :

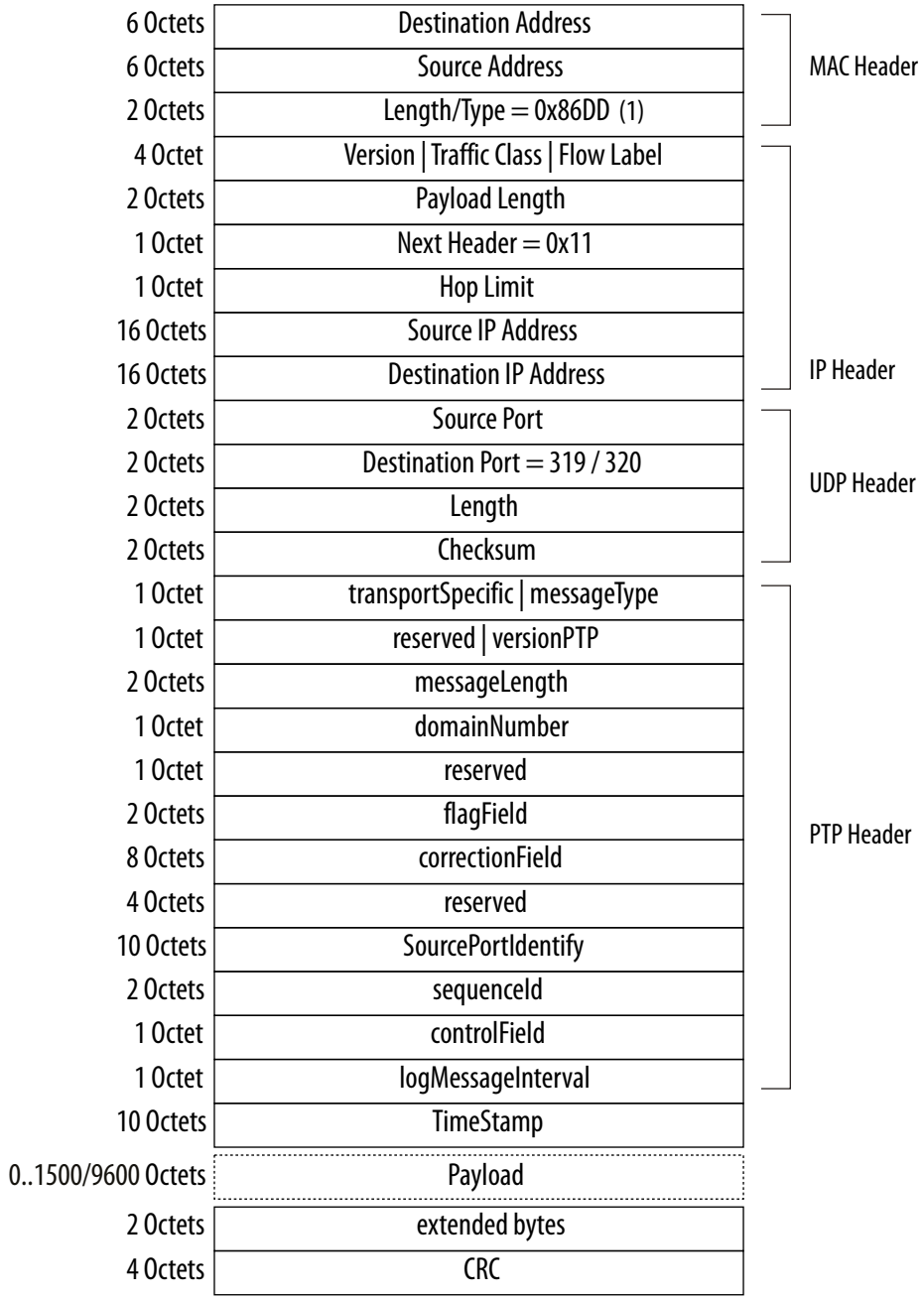
1. For frames with VLAN or Stacked VLAN tag, add 4 or 8 octets offsets before the length/type field.



#### 4.2.12.6.3. PTP Frame over UDP/IPv6

Checksum calculation is mandatory for the UDP/IPv6 protocol. You must extend 2 bytes at the end of the UDP payload of the PTP frame. The MAC function modifies the extended bytes to ensure that the UDP checksum remains uncompromised.

**Figure 41. PTP Frame over UDP/IPv6**




Note to Figure 41 on page 80 :

1. For frames with VLAN or Stacked VLAN tag, add 4 or 8 octets offsets before the length/type field.

# 5. Configuration Register Space

## 5.1. MAC Configuration Register Space

Use the registers to configure the different aspects of the MAC function and retrieve its status and statistics counters.

In multiport MACs, a contiguous register space is allocated for all ports and accessed via the Avalon Memory-Mapped control interface. For example, if the register space base address for the first port is 0x00, the base address for the next port is 0x100 and so forth. The registers that are shared among the instances occupy the register space of the first port. Updating these registers in the register space of other ports has no effect on the configuration.

*Note:* If you select **10/100/1000Mb Ethernet MAC** core variant and deselect **Use clock enable for MAC** in the parameter editor of the Triple-Speed Ethernet Intel FPGA IP core, the CSR access latency will be dependent on the datapath clock frequency.

**Table 36. Overview of MAC Register Space**

| Dword Offset | Section | Description |
|---|---|---|
| 0x00 – 0x17 | Base Configuration | Base registers to configure the MAC function. At the minimum, you must configure the following functions:<br>• Primary MAC address (`mac_0/mac_1`)<br>• Enable transmit and receive paths (TX_ENA and RX_ENA bits in the `command_config` register)<br>The following registers are shared among all instances of a multiport MAC:<br>• `rev`<br>• `scratch`<br>• `frm_length`<br>• `pause_quant`<br>• `mdio_addr0` and `mdio_addr1`<br>• `tx_ipg_length`<br>For more information about the base configuration registers, refer to Base Configuration Registers (Dword Offset 0x00 – 0x17) on page 82. |
| 0x18 – 0x38 | Statistics Counters | Counters collecting traffic statistics. For more information about the statistics counters, refer to Statistics Counters (Dword Offset 0x18 – 0x38) on page 88. |
| 0x3A | Transmit Command | Transmit and receive datapaths control register. For more information about these registers, see Transmit and Receive Command Registers (Dword Offset 0x3A – 0x3B) on page 91. |
| 0x3B | Receive Command | |
| 0x3C – 0x3E | Extended Statistics Counters | Upper 32 bits of selected statistics counters. These registers are used if you turn on the option to use extended statistics counters. For more information about these counters, refer to Statistics Counters (Dword Offset 0x18 – 0x38) on page 88 . |
| 0x3F | Reserved | Unused. |

*continued...*

| Dword Offset | Section | Description |
|---|---|---|
| 0x40 – 0x7F | Multicast Hash Table | 64-entry write-only hash table to resolve multicast addresses. Only bit 0 in each entry is significant. When you write a 1 to a dword offset in the hash table, the MAC accepts all multicast MAC addresses that hash to the value of the address (bits 5:0). Otherwise, the MAC rejects the multicast address. This table is cleared during reset.<br>Hashing is not supported in 10/100 and 1000 Mbps Small MAC core variations. |
| 0x80 – 0x9F | MDIO Space 0<br>or PCS Function Configuration | MDIO Space 0 and MDIO Space 1 map to registers 0 to 31 of the PHY devices whose addresses are configured in the `mdio_addr0` and `mdio_addr1` registers respectively. For example, register 0 of PHY device 0 maps to dword offset 0x80, register 1 maps to dword offset 0x81 and so forth.<br>Reading or writing to MDIO Space 0 or MDIO Space 1 immediately triggers a corresponding MDIO transaction to read or write the PHY register. Only bits [15:0] of each register are significant. Write 0 to bits [31:16] and ignore them on reads.<br>If your variation does not include the PCS function, you can use MDIO Space 0 and MDIO Space 1 to map to two PHY devices.<br>If your MAC variation includes the PCS function, the PCS function is always device 0 and its configuration registers (PCS Configuration Register Space on page 93) occupy MDIO Space 0. You can use MDIO Space 1 to map to a PHY device. |
| 0xA0 – 0xBF | MDIO Space 1 | |
| 0xC0 – 0xC7 | Supplementary Address | Supplementary unicast addresses. For more information about these addresses, refer to Supplementary Address (Dword Offset 0xC0 – 0xC7) on page 91. |
| 0xC8 – 0xCF | Reserved (1) | Unused. |
| 0xD0 – 0xD6 | IEEE 1588v2 Feature | Registers to configure the IEEE 1588v2 feature. For more information about these registers, refer to IEEE 1588v2 Feature (Dword Offset 0xD0 – 0xD6) on page 92. |
| 0xD7 – 0xFF | Reserved (1) | Unused. |

Note to Table 36 on page 81:
1. Intel recommends that you set all bits in the reserved registers to 0 and ignore them on reads.

## 5.1.1. Base Configuration Registers (Dword Offset 0x00 – 0x17)

The following table lists the base registers you can use to configure the MAC function. A software reset does not reset these registers except the first two bits (TX_ENA and RX_ENA = 0) in the `command_config` register.

**Table 37. Base Configuration Register Map**

| Dword Offset | Name | R/W | Description | HW Reset |
|---|---|---|---|---|
| 0x00 | `rev` | RO | • Bits[15:0]—Set to the current version of the IP core.<br>• Bits[31:16]—Customer specific revision, specified by the `CUST_VERSION` parameter defined in the top-level file generated for the instance of the IP core. These bits are set to 0 during the configuration of the IP core. | <IP version number> |
| 0x01 | `scratch`(1) | RW | Scratch register. Provides a memory location for you to test the device memory operation. | 0 |
| 0x02 | `command_config` | RW | MAC configuration register. Use this register to control and configure the MAC function. The MAC function starts operation as soon as the transmit and receive enable bits in this register are turned on. Intel, therefore, recommends that you configure this register last. | 0 |

*continued...*

| Dword Offset | Name | R/W | Description | HW Reset |
|---|---|---|---|---|
| | | | See Command_Config Register (Dword Offset 0x02) on page 85 for the bit description. | |
| 0x03 | mac_0 | RW | 6-byte MAC primary address. The first four most significant bytes of the MAC address occupy mac_0 in reverse order. The last two bytes of the MAC address occupy the two least significant bytes of mac_1 in reverse order.<br>For example, if the MAC address is 00-1C-23-17-4A-CB, the following assignments are made:<br>mac_0 = 0x17231c00<br>mac_1 = 0x0000CB4a<br>Ensure that you configure these registers with a valid MAC address if you disable the promiscuous mode (PROMIS_EN bit in command_config = 0). | 0 |
| 0x04 | mac_1 | RW | | 0 |
| 0x05 | frm_length | RW/ RO | • Bits[15:0]—16-bit maximum frame length in bytes. The IP core checks the length of receive frames against this value. Typical value is 1518.<br>In 10/100 and 1000 Small MAC core variations, this register is RO and the maximum frame length is fixed to 1518.<br>• Bits[31:16]—unused. | 1518 |
| 0x06 | pause_quant | RW | • Bits[15:0]—16-bit pause quanta. Use this register to specify the pause quanta to be sent to remote devices when the local device is congested. The IP core sets the pause quanta (P1, P2) field in pause frames to the value of this register.<br>10/100 and 1000 Small MAC core variations do not support flow control.<br>• Bits[31:16]—unused. | 0 |
| 0x07 | rx_section_empty | RW/ RO | Variable-length section-empty threshold of the receive FIFO buffer. Use the depth of your FIFO buffer to determine this threshold. This threshold is typically set to (FIFO Depth – 16).<br>Set this threshold to a value that is below the rx_almost_full threshold and above the rx_section_full or rx_almost_empty threshold.<br>In 10/100 and 1000 Small MAC core variations, this register is RO and the register is set to a fixed value of (FIFO Depth – 16). | 0 |
| 0x08 | rx_section_full | RW/ RO | Variable-length section-full threshold of the receive FIFO buffer. Use the depth of your FIFO buffer to determine this threshold.<br>For cut-through mode, this threshold is typically set to 16. Set this threshold to a value that is above the rx_almost_empty threshold.<br>For store-and-forward mode, set this threshold to 0.<br>In 10/100 and 1000 Small MAC core variations, this register is RO and the register is set to a fixed value of 16. | 0 |
| 0x09 | tx_section_empty | RW/ RO | Variable-length section-empty threshold of the transmit FIFO buffer. Use the depth of your FIFO buffer to determine this threshold. This threshold is typically set to (FIFO Depth – 16).<br>Set this threshold to a value below the rx_almost_full threshold and above the rx_section_full or rx_almost_empty threshold. | 0 |

*continued...*

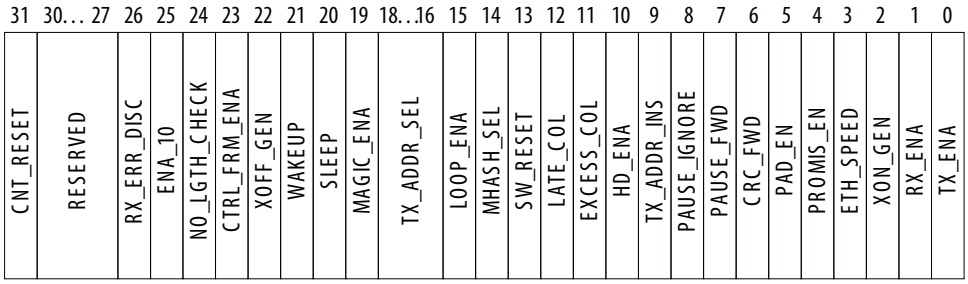
| Dword Offset | Name | R/W | Description | HW Reset |
|---|---|---|---|---|
| | | | In 10/100 and 1000 Small MAC core variations, this register is RO and the register is set to a fixed value of (FIFO Depth – 16). | |
| 0x0A | tx_section_full | RW/ RO | Variable-length section-full threshold of the transmit FIFO buffer. Use the depth of your FIFO buffer to determine this threshold.<br>For cut-through mode, this threshold is typically set to 16. Set this threshold to a value above the tx_almost_empty threshold.<br>For store-and-forward mode, set this threshold to 0.<br>In 10/100 and 1000 Small MAC core variations, this register is RO and the register is set to a fixed value of 16. | 0 |
| 0x0B | rx_almost_empty | RW/ RO | Variable-length almost-empty threshold of the receive FIFO buffer. Use the depth of your FIFO buffer to determine this threshold.<br>Due to internal pipeline latency, you must set this threshold to a value greater than 3. This threshold is typically set to 8.<br>In 10/100 and 1000 Small MAC core variations, this register is RO and the register is set to a fixed value of 8. | 0 |
| 0x0C | rx_almost_full | RW/ RO | Variable-length almost-full threshold of the receive FIFO buffer. Use the depth of your FIFO buffer to determine this threshold.<br>Due to internal pipeline latency, you must set this threshold to a value greater than 3. This threshold is typically set to 8.<br>In 10/100 and 1000 Small MAC core variations, this register is RO and the register is set to a fixed value of 8. | 0 |
| 0x0D | tx_almost_empty | RW/ RO | Variable-length almost-empty threshold of the transmit FIFO buffer. Use the depth of your FIFO buffer to determine this threshold.<br>Due to internal pipeline latency, you must set this threshold to a value greater than 3. This threshold is typically set to 8.<br>In 10/100 and 1000 Small MAC core variations, this register is RO and the register is set to a fixed value of 8. | 0 |
| 0x0E | tx_almost_full | RW/ RO | Variable-length almost-full threshold of the transmit FIFO buffer. Use the depth of your FIFO buffer to determine this threshold.<br>You must set this register to a value greater than or equal to 3. A value of 3 indicates 0 ready latency; a value of 4 indicates 1 ready latency, and so forth. Because the maximum ready latency on the Avalon Streaming interface is 8, you can only set this register to a maximum value of 11. This threshold is typically set to 3.<br>In 10/100 and 1000 Small MAC core variations, this register is RO and the register is set to a fixed value of 3. | 0 |
| 0x0F | mdio_addr0 | RW | • Bits[4:0]—5-bit PHY address. Set these registers to the addresses of any connected PHY devices you want to access. The mdio_addr0 and mdio_addr1 registers contain the addresses of the PHY whose registers are mapped to MDIO Space 0 and MDIO Space 1 respectively.<br>• Bits[31:5]—unused. Set to read-only value of 0. | 0 |
| 0x10 | mdio_addr1 | RW | | 1 |

continued...

| Dword Offset | Name | R/W | Description | HW Reset |
|---|---|---|---|---|
| 0x11 | holdoff_quant | RW | • Bit[15:0]—16-bit holdoff quanta. When you enable the flow control, use this register to specify the gap between consecutive XOFF requests.<br>• Bits[31:16]—unused. | 0xFFFF |
| 0x12 – 0x16 | Reserved | — | — | 0 |
| 0x17 | tx_ipg_length | RW | • Bits[4:0]—minimum IPG. Valid values are between 8 and 26 byte-times. If this register is set to an invalid value, the MAC still maintains a typical minimum IPG value of 12 bytes between packets, although a read back to the register reflects the invalid value written.<br>In 10/100 and 1000 Small MAC core variations, this register is RO and the register is set to a fixed value of 12.<br>Bits[31:5]—unused. Set to read-only value 0. | 0 |
| Note to Table 37 on page 82 :<br>1. Register is not available in 10/100 and 1000 Small MAC variations. | | | | |

### 5.1.1.1. Command_Config Register (Dword Offset 0x02)

**Figure 42. Command_Config Register Fields**




At the minimum, you must configure the TX_ENA and RX_ENA bits to 1 to start the MAC operations. When configuring the command_config register, Intel recommends that you configure the TX_ENA and RX_ENA bits the last because the MAC function immediately starts its operations once these bits are set to 1.

**Table 38. Command_Config Register Field Descriptions**

| Bit(s) | Name | R/W | Description | HW Reset |
|---|---|---|---|---|
| 0 | TX_ENA | RW | Transmit enable. Set this bit to 1 to enable the transmit datapath. The MAC function clears this bit following a hardware or software reset. See the SW_RESET bit description. | 0 |
| 1 | RX_ENA | RW | Receive enable. Set this bit to 1 to enable the receive datapath. The MAC function clears this bit following a hardware or software reset. See the SW_RESET bit description. | 0 |
| 2 | XON_GEN | RW | Pause frame generation. When you set this bit to 1, the MAC function generates a pause frame with a pause quanta of 0, independent of the status of the receive FIFO buffer. | 0 |
| | | | | *continued...* |

| Bit(s) | Name | R/W | Description | HW Reset |
|---|---|---|---|---|
| 3 | ETH_SPEED | RW | Ethernet speed control.<br>• Set this bit to 1 to enable gigabit Ethernet operation. The set_1000 signal is masked and does not affect the operation.<br>• If you set this bit to 0, gigabit Ethernet operation is enabled only if the set_1000 signal is asserted. Otherwise, the MAC function operates in 10/100 Mbps Ethernet mode.<br>When the MAC operates in gigabit mode, the eth_mode signal is asserted. This bit is not available in the small MAC variation. | 0 |
| 4 | PROMIS_EN | RW | Promiscuous enable. Set this bit to 1 to enable promiscuous mode. In this mode, the MAC function receives all frames without address filtering. | 0 |
| 5 | PAD_EN | RW | Padding removal on receive. Set this bit to 1 to remove padding from receive frames before the MAC function forwards the frames to the user application. This bit has no effect on transmit frames.<br>This bit is not available in the small MAC variation. | 0 |
| 6 | CRC_FWD | RW | CRC forwarding on receive.<br>• Set this bit to 1 to forward the CRC field to the user application.<br>• Set this bit to 0 to remove the CRC field from receive frames before the MAC function forwards the frame to the user application.<br>• The MAC function ignores this bit when it receives a padded frame and the PAD_EN bit is 1. 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. | 0 |
| 7 | PAUSE_FWD | RW | Pause frame forwarding on receive.<br>• Set this bit to 1 to forward receive pause frames to the user application.<br>• Set this bit to 0 to terminate and discard receive pause frames. | 0 |
| 8 | PAUSE_IGNORE | RW | Pause frame processing on receive.<br>• Set this bit to 1 to ignore receive pause frames.<br>• Set this bit to 0 to process receive pause frames. The MAC function suspends transmission for an amount of time specified by the pause quanta. | 0 |
| 9 | TX_ADDR_INS | RW | MAC address on transmit.<br>• Set this bit to 1 to overwrite the source MAC address in transmit frames received from the user application with the MAC primary or supplementary address configured in the registers. The TX_ADDR_SEL bit determines the address selection.<br>• Set this bit to 0 to retain the source MAC address in transmit frames received from the user application. | 0 |
| 10 | HD_ENA | RW | Half-duplex enable.<br>• Set this bit to 1 to enable half-duplex.<br>• Set this bit to 0 to enable full-duplex.<br>• The MAC function ignores this bit if you set the ETH_SPEED bit to 1. | 0 |
| 11 | EXCESS_COL | RO | Excessive collision condition. | 0 |

*continued...*

| Bit(s) | Name | R/W | Description | HW Reset |
|---|---|---|---|---|
| | | | • The MAC function sets this bit to 1 when it discards a frame after detecting a collision on 16 consecutive frame retransmissions.<br>• The MAC function clears this bit following a hardware or software reset. See the SW_RESET bit description. | |
| 12 | LATE_COL | RO | Late collision condition.<br>• The MAC function sets this bit to 1 when it detects a collision after transmitting 64 bytes and discards the frame.<br>• The MAC function clears this bit following a hardware or software reset. See the SW_RESET bit description. | 0 |
| 13 | SW_RESET | RW | Software reset. Set this bit to 1 to trigger a software reset. The MAC function clears this bit when it completes the software reset sequence.<br>When software reset is triggered, the MAC function completes the current transmission or reception, and subsequently disables the transmit and receive logic, flushes the receive FIFO buffer, and resets the statistics counters. | 0 |
| 14 | MHASH_SEL | RW | Hash-code mode selection for multicast address resolution.<br>• Set this bit to 0 to generate the hash code from the full 48-bit destination address.<br>• Set this bit to 1 to generate the hash code from the lower 24 bits of the destination MAC address. | 0 |
| 15 | LOOP_ENA | RW | Local loopback enable. Set this bit to 1 to enable local loopback on the RGMII/GMII/MII of the MAC. The MAC function sends transmit frames back to the receive path.<br>This bit is not available in the small MAC variation. | 0 |
| 16 – 18 | TX_ADDR_SEL[2:0] | RW | Source MAC address selection on transmit. If you set the TX_ADDR_INS bit to 1, the value of these bits determines the MAC address the MAC function selects to overwrite the source MAC address in frames received from the user application.<br>• 000 = primary address configured in the mac_0 and mac_1 registers.<br>• 100 = supplementary address configured in the smac_0_0 and smac_0_1 registers.<br>• 101 = supplementary address configured in the smac_1_0 and smac_1_1 registers.<br>• 110 = supplementary address configured in the smac_2_0 and smac_2_1 registers.<br>• 111 = supplementary address configured in the smac_3_0 and smac_3_1 registers. | 000 |
| 19 | MAGIC_ENA | RW | Magic packet detection. Set this bit to 1 to enable magic packet detection.<br>This bit is not available in the small MAC variation. | 0 |
| 20 | SLEEP | RW | Sleep mode enable. When the MAGIC_ENA bit is 1, set this bit to 1 to put the MAC function to sleep and enable magic packet detection.<br>This bit is not available in the small MAC variation. | 0 |
| 21 | WAKEUP | RO | Node wake-up request. Valid only when the MAGIC_ENA bit is 1. | 0 |

***continued...***

| Bit(s) | Name | R/W | Description | HW Reset |
|---|---|---|---|---|
| | | | • The MAC function sets this bit to 1 when a magic packet is detected.<br>• The MAC function clears this bit when the SLEEP bit is set to 0. | |
| 22 | XOFF_GEN | RW | Pause frame generation. Set this bit to 1 to generate a pause frame independent of the status of the receive FIFO buffer. The MAC function sets the pause quanta field in the pause frame to the value configured in the pause_quant register. | 0 |
| 23 | CNTL_FRM_ENA | RW | MAC control frame enable on receive.<br>• Set this bit to 1 to accept control frames other than pause frames (opcode = 0x0001) and forward them to the user application.<br>• Set this bit to 0 to discard control frames other than pause frames. | 0 |
| 24 | NO_LGTH_CHECK | RW | Payload length check on receive.<br>• Set this bit to 0 to check the actual payload length of receive frames against the length/type field in receive frames.<br>• Set this bit to 1 to omit length checking.<br>This bit is not available in the small MAC variation | 0<br>1 (for small MAC variation) |
| 25 | ENA_10 | RW | 10-Mbps interface enable. Set this bit to 1 to enable the 10-Mbps interface. The MAC function asserts the ena_10 signal when you enable the 10-Mbps interface. You can also enable the 10-Mbps interface by asserting the set_10 signal. | 0 |
| 26 | RX_ERR_DISC | RW | Erroneous frames processing on receive.<br>• Set this bit to 1 to discard erroneous frames received. This applies only when you enable store and forward operation in the receive FIFO buffer by setting the rx_section_full register to 0.<br>• Set this bit to 0 to forward erroneous frames to the user application with rx_err[0] asserted. | 0 |
| 27 | DISABLE_READ_TIMEOUT | RW | By default, this bit is set to 0. Set this bit to 1 to disable MAC configuration register read timeout.<br>To ensure the configuration register does not wait for read timeout when an error occurs, set this bit to 1. | 0 |
| 28 – 30 | Reserved | — | — | 000 |
| 31 | CNT_RESET | RW | Statistics counters reset. Set this bit to 1 to clear the statistics counters. The MAC function clears this bit when the reset sequence completes. | 0 |

## 5.1.2. Statistics Counters (Dword Offset 0x18 – 0x38)

The following table describes the read-only registers that collect the statistics on the transmit and receive datapaths. A hardware reset clears these registers; a software reset also clears these registers except aMacID. The statistics counters roll up when the counter is full.

The register description uses the following definitions:

- Good frame—error-free frames with valid frame length.
- Error frame—frames that contain errors or whose length is invalid.
- Invalid frame—frames that are not addressed to the MAC function. The MAC function drops this frame.

**Table 39. Statistics Counters**

| Dword Offset | Name | R/W | Description |
|---|---|---|---|
| 0x18 – 0x19 | aMacID | RO | The MAC address. This register is wired to the primary MAC address in the mac_0 and mac_1 registers. |
| 0x1A | aFramesTransmittedOK | RO | The number of frames that are successfully transmitted including the pause frames. |
| 0x1B | aFramesReceivedOK | RO | The number of frames that are successfully received including the pause frames. |
| 0x1C | aFrameCheckSequenceErrors | RO | The number of receive frames with CRC error. |
| 0x1D | aAlignmentErrors | RO | The number of receive frames with alignment error. |
| 0x1E | aOctetsTransmittedOK | RO | The number of data and padding octets that are successfully transmitted.<br>This register contains the lower 32 bits of the aOctetsTransmittedOK counter. The upper 32 bits of this statistics counter reside at the dword offset 0x0F. |
| 0x1F | aOctetsReceivedOK | RO | The number of data and padding octets that are successfully received, including pause frames.<br>The lower 32 bits of the aOctetsReceivedOK counter. The upper 32 bits of this statistics counter reside at the dword offset 0x3D. |
| 0x20 | aTxPAUSEMACCtrlFrames | RO | The number of pause frames transmitted. |
| 0x21 | aRxPAUSEMACCtrlFrames | RO | The number received pause frames received. |
| 0x22 | ifInErrors | RO | The number of errored frames received. |
| 0x23 | ifOutErrors | RO | The number of transmit frames with one the following errors:<br>• FIFO overflow error<br>• FIFO underflow error<br>• Frames that encounter late or excessive collision occasions<br>• Errors defined by the user application |
| 0x24 | ifInUcastPkts | RO | The number of valid unicast frames received. |
| 0x25 | ifInMulticastPkts | RO | The number of valid multicast frames received. The count does not include pause frames. |
| 0x26 | ifInBroadcastPkts | RO | The number of valid broadcast frames received. |
| 0x27 | Reserved | — | Unused. |
| 0x28 | ifOutUcastPkts | RO | The number of valid unicast and erroneous frames transmitted, as well as unicast frames transmitted during late and excessive collision occasions. |

*continued...*

| Dword Offset | Name | R/W | Description |
|---|---|---|---|
| 0x29 | ifOutMulticastPkts | RO | The number of valid multicast frames transmitted, as well as multicast frames transmitted during late and excessive collision occasions, excluding pause frames. |
| 0x2A | ifOutBroadcastPkts | RO | The number of valid and erroneous broadcast frames transmitted, as well as broadcast frames transmitted during late and excessive collision occasions. |
| 0x2B | etherStatsDropEvents | RO | The number of frames that are dropped due to MAC internal errors when FIFO buffer overflow persists. |
| 0x2C | etherStatsOctets | RO | The total number of octets received. This count includes both good and errored frames.<br>This register is the lower 32 bits of `etherStatsOctets`. The upper 32 bits of this statistics counter reside at the dword offset 0x3E. |
| 0x2D | etherStatsPkts | RO | The total number of good and errored frames received. |
| 0x2E | etherStatsUndersizePkts | RO | The number of frames received with length less than 64 bytes, including pause frames. This count does not include errored frames. |
| 0x2F | etherStatsOversizePkts | RO | The number of frames received that are longer than the value configured in the `frm_length` register. This count does not include errored frames. |
| 0x30 | etherStatsPkts64Octets | RO | The number of 64-byte frames received. This count includes good and errored frames. |
| 0x31 | etherStatsPkts65to127Octets | RO | The number of received good and errored frames between the length of 65 and 127 bytes. |
| 0x32 | etherStatsPkts128to255Octets | RO | The number of received good and errored frames between the length of 128 and 255 bytes. |
| 0x33 | etherStatsPkts256to511Octets | RO | The number of received good and errored frames between the length of 256 and 511 bytes. |
| 0x34 | etherStatsPkts512to1023Octets | RO | The number of received good and errored frames between the length of 512 and 1023 bytes. |
| 0x35 | etherStatsPkts1024to1518Octets | RO | The number of received good and errored frames between the length of 1024 and 1518 bytes. |
| 0x36 | etherStatsPkts1519toXOctets | RO | The number of received good and errored frames between the length of 1519 and the maximum frame length configured in the `frm_length` register. |
| 0x37 | etherStatsJabbers | RO | Too long frames with CRC error. |
| 0x38 | etherStatsFragments | RO | Too short frames with CRC error. |
| 0x39 | Reserved | — | Unused. |
| **Extended Statistics Counters (0x3C – 0x3E)** | | | |
| 0x3C | msb_aOctetsTransmittedOK | RO | 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 **Enable 64-bit byte counters** parameter.<br>To read the counter, read the lower 32 bits first, then followed by the extended statistic counter bits. |
| 0x3D | msb_aOctetsReceivedOK | RO | |
| 0x3E | msb_etherStatsOctets | RO | |

## 5.1.3. Transmit and Receive Command Registers (Dword Offset 0x3A – 0x3B)

The following table describes the registers that determine how the MAC function processes transmit and receive frames. A software reset does not change the values in these registers.

**Table 40. Transmit and Receive Command Registers**

| Dword Offset | Name | R/W | Description |
|---|---|---|---|
| 0x3A | `tx_cmd_stat` | RW | Specifies how the MAC function processes transmit frames. When you turn on the **Align packet headers to 32-bit boundaries** option, this register resets to 0x00040000 upon a hardware reset. Otherwise, it resets to 0x00.<br>• Bits 0 to 16—unused.<br>• Bit 17 (`OMIT_CRC`)—Set this bit to 1 to omit CRC calculation and insertion on the transmit path. The user application is therefore responsible for providing the correct data and CRC. This bit, when set to 1, always takes precedence over the `ff_tx_crc_fwd` signal.<br>• Bit 18 (`TX_SHIFT16`)—Set this bit to 1 if the frames from the user application are aligned on 32-bit boundary. For more information, refer to IP Payload Re-alignment on page 36.<br>This setting applies only when you turn on the **Align packet headers to 32-bit boundary** option and in MAC variations with 32-bit internal FIFO buffers. Otherwise, reading this bit always return a 0.<br>In MAC variations without internal FIFO buffers, this bit is a read-only bit and takes the value of the **Align packet headers to 32-bit boundary** option.<br>• Bits 19 to 31—unused. |
| 0x3B | `rx_cmd_stat` | RW | Specifies how the MAC function processes receive frames. When you turn on the **Align packet headers to 32-bit boundaries** option, this register resets to 0x02000000 upon a hardware reset. Otherwise, it resets to 0x00.<br>• Bits 0 to 24—unused.<br>• Bit 25 (`RX_SHIFT16`)—Set this bit to 1 to instruct the MAC function to align receive frames on 32-bit boundary. For more information on frame alignment, refer to IP Payload Alignment on page 43.<br>This setting applies only when you turn on the **Align packet headers to 32-bit boundary** option and in MAC variations with 32-bit internal FIFO buffers. Otherwise, reading this bit always return a 0.<br>In MAC variations without internal FIFO buffers, this bit is a read-only bit and takes the value of the **Align packet headers to 32-bit boundary** option.<br>• Bits 26 to 31—unused. |

## 5.1.4. Supplementary Address (Dword Offset 0xC0 – 0xC7)

A software reset has no impact on these registers. MAC supplementary addresses are not available in 10/100 and 1000 Small MAC variations.

**Table 41. Supplementary Address Registers**

| Dword Offset | Name | R/W | Description | HW Reset |
|---|---|---|---|---|
| 0xC0 | smac_0_0 | RW | You can specify up to four 6-byte supplementary addresses:<br>• smac_0_0/1<br>• smac_1_0/1<br>• smac_2_0/1<br>• smac_3_0/1<br>Map the supplementary addresses to the respective registers in the same manner as the primary MAC address. Refer to the description of mac_0 and mac_1.<br>The MAC function uses the supplementary addresses for the following operations:<br>• to filter unicast frames when the promiscuous mode is disabled (refer to Command_Config Register (Dword Offset 0x02) on page 85 for the description of the PROMIS_EN bit).<br>• to replace the source address in transmit frames received from the user application when address insertion is enabled (refer to Command_Config Register (Dword Offset 0x02) on page 85 for the description of the TX_ADDR_INS and TX_ADDR_SEL bits).<br>If you do not require the use of supplementary addresses, configure them to the primary address. | 0 |
| 0xC1 | smac_0_1 | | | |
| 0xC2 | smac_1_0 | | | |
| 0xC3 | smac_1_1 | | | |
| 0xC4 | smac_2_0 | | | |
| 0xC5 | smac_2_1 | | | |
| 0xC6 | smac_3_0 | | | |
| 0xC7 | smac_3_1 | | | |

## 5.1.5. IEEE 1588v2 Feature (Dword Offset 0xD0 – 0xD6)

**Table 42. IEEE 1588v2 MAC Registers**

| Dword Offset | Name | R/W | Description | HW Reset |
|---|---|---|---|---|
| 0xD0 | tx_period | RW | Clock period for timestamp adjustment on the transmit datapath. The period register is multiplied by the number of stages separating actual timestamp and the GMII bus.<br>• Bits 0 to 15: Period in fractional nanoseconds (TX_PERIOD_FNS).<br>• Bits 16 to 24: Period in nanoseconds (TX_PERIOD_NS).<br>• Bits 25 to 31: Not used.<br>The default value for the period is 0. For 125-MHz clock, set this register to 8 ns. | 0x0 |
| 0xD1 | tx_adjust_fns | RW | Static timing adjustment in fractional nanoseconds for outbound timestamps on the transmit datapath.<br>• Bits 0 to 15: Timing adjustment in fractional nanoseconds.<br>• Bits 16 to 31: Not used. | 0x0 |
| 0xD2 | tx_adjust_ns | RW | Static timing adjustment in nanoseconds for outbound timestamps on the transmit datapath.<br>• Bits 0 to 15: Timing adjustment in nanoseconds.<br>• Bits 16 to 23: Not used. | 0x0 |
| 0xD3 | rx_period | RW | Clock period for timestamp adjustment on the receive datapath. The period register is multiplied by the number of stages separating actual timestamp and the GMII bus.<br>• Bits 0 to 15: Period in fractional nanoseconds (RX_PERIOD_FNS).<br>• Bits 16 to 24: Period in nanoseconds (RX_PERIOD_NS).<br>• Bits 25 to 31: Not used. | 0x0 |

*continued...*

| Dword Offset | Name | R/W | Description | HW Reset |
|---|---|---|---|---|
| | | | The default value for the period is 0. For 125-MHz clock, set this register to 8 ns. | |
| 0xD4 | rx_adjust_fns | RW | Static timing adjustment in fractional nanoseconds for outbound timestamps on the receive datapath.<br>• Bits 0 to 15: Timing adjustment in fractional nanoseconds.<br>• Bits 16 to 31: Not used. | 0x0 |
| 0xD5 | rx_adjust_ns | RW | Static timing adjustment in nanoseconds for outbound timestamps on the receive datapath.<br>• Bits 0 to 15: Timing adjustment in nanoseconds.<br>• Bits 16 to 23: Not used. | 0x0 |

### 5.1.6. IEEE 1588v2 Feature PMA Delay

PMA digital and analog delay of hardware for the IEEE 1588v2 feature and the register timing adjustment. 1 UI is equivalent to 800 ps.

**Table 43. IEEE 1588v2 Feature PMA Delay—Hardware**

| Delay | Device | Timing Adjustment | |
|---|---|---|---|
| | | TX register | RX register |
| Digital | Stratix V or Arria V GZ | 53 UI | 26 UI |
| | Arria V GX, Arria V GT, or Arria V SoC | 52 UI | 34 UI |
| | Cyclone V GX or Cyclone V SoC | 32 UI | 44 UI |
| | Intel Arria 10 | 43 UI | 24.5 UI |
| Analog | Stratix V | -1.1 ns | 1.75 ns |
| | Arria V | -1.1 ns | 1.75 ns |
| | Cyclone V | -1.1 ns | 1.75 ns |

**Table 44. IEEE 1588v2 Feature LVDS I/O Delay—Hardware**

| Delay | Device | Timing Adjustment | |
|---|---|---|---|
| | | TX register | RX register |
| Digital | Stratix V or Arria V GZ | 21 UI | 26 UI |
| | Arria V GX, Arria V GT, or Arria V SoC | 21 UI | 26 UI |
| | Intel Arria 10 | 21 UI | 26 UI |
| | Intel Stratix 10 (L-tile and H-tile) | 21 UI | 26 UI |

## 5.2. PCS Configuration Register Space

This section describes the PCS registers. Use the registers to configure the PCS function or retrieve its status.

*Note:* In MAC and PCS variations, the PCS registers occupy the MAC register space and you access these registers via the MAC 32-bit Avalon Memory-Mapped control interface. PCS registers are 16 bits wide, they therefore occupy only the lower 16 bits and the upper 16 bits are set to 0. The offset of the first PCS register in this variation is mapped to dword offset 0x80.

If you instantiate the IP core using the IP Catalog flow, use word addressing to access the register spaces. When you instantiate MAC and PCS variations, map the PCS registers to the respective dword offsets in the MAC register space by adding the PCS word offset to the offset of the first PCS. For example,

- In PCS only variation, you can access the `if_mode` register at word offset 0x14.
- In MAC and PCS variations, map the `if_mode` register to the MAC register space:
  - Offset of the first PCS register = 0x80
  - `if_mode` word offset = 0x14
  - `if_mode` dword offset = 0x80 + 0x14 = 0x94

If you instantiate the MAC and PCS variation using the Platform Designer system, access the register spaces using byte addressing. Convert the dword offsets to byte offsets by multiplying the dword offsets by 4. For example,

- For MAC registers:
  - `comand_config` dword offset = 0x02
  - `comand_config` byte offset = 0x02 × 4 = 0x08
- For PCS registers, map the registers to the dword offsets in the MAC register space before you convert the dword offsets to byte offsets:
  - `if_mode` word offset = 0x14
  - `if_mode` dword offset = 0x80 + 0x14 = 0x94
  - `if_mode` byte offset = 0x94 × 4 = 0x250

**Table 45. PCS Configuration Registers**

| Word Offset | Register Name | R/W | Description | HW Reset |
|---|---|---|---|---|
| 0x00 | `control` | RW | PCS control register. Use this register to control and configure the PCS function. For the bit description, see Control Register (Word Offset 0x00) on page 95. | 0x1140 |
| 0x01 | `status` | RO | Status register. Provides information on the operation of the PCS function. | 0x0089 |
| 0x02 | `phy_identifier` | RO | 32-bit PHY identification register. This register is set to the value of the **PHY ID** parameter. Bits 31:16 are written to word offset 0x02. Bits 15:0 are written to word offset 0x03. | 0x0101 |
| 0x03 | | | | 0x0101 |
| 0x04 | `dev_ability` | RW | Use this register to advertise the device abilities to a link partner during auto-negotiation. In SGMII MAC mode, the PHY does not use this register during auto-negotiation. For the register bits description in 1000BASE-X and SGMII mode, see 1000BASE-X on page 97 and SGMII PHY Mode Auto Negotiation on page 99. | 0x01A0 |

*continued...*

| Word Offset | Register Name | R/W | Description | HW Reset |
|---|---|---|---|---|
| 0x05 | partner_ability | RO | 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 1000BASE-X on page 97 and SGMII PHY Mode Auto Negotiation on page 99, respectively. | 0x0000 |
| 0x06 | an_expansion | RO | Auto-negotiation expansion register. Contains the PCS function capability and auto-negotiation status. | 0x0000 |
| 0x07 | device_next_page | RO | The PCS function does not support these features. These registers are always set to 0x0000 and any write access to the registers is ignored. | 0x0000 |
| 0x08 | partner_next_page | | | 0x0000 |
| 0x09 | master_slave_cntl | | | 0x0000 |
| 0x0A | master_slave_stat | | | 0x0000 |
| 0x0B – 0x0E | Reserved | — | — | — |
| 0x0F | extended_status | RO | The PCS function does not implement extended status registers. | — |
| **Specific Extended Registers** | | | | |
| 0x10 | scratch | RW | Scratch register. Provides a memory location to test register read and write operations. | 0x0000 |
| 0x11 | rev | RO | The PCS function revision. Always set to the current version of the IP core. | <IP version number> |
| 0x12 | link_timer | RW | 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.<br>• Bits 15:0 are written to word offset 0x12. Bit 0 of word offset 0x12 is always set to 0, thus any value written to it is ignored.<br>• Bits 20:16 are written to word offset 0x13. The remaining bits are reserved and always set to 0. | 0x8968 |
| 0x13 | | | | 0x0009 |
| 0x14 | if_mode | RW | Interface mode. Use this register to specify the operating mode of the PCS function; 1000BASE-X or SGMII. | 0 |
| 0x17 – 0x1F | Reserved | — | — | 0 |

## 5.2.1. Control Register (Word Offset 0x00)

### Table 46. PCS Control Register Bit Descriptions

| Bit(s) | Name | R/W | Description |
|---|---|---|---|
| 0:4 | Reserved | — | — |
| 5 | UNIDIRECTIONAL_ENABLE | RW | Enables the unidirectional function. This bit depends on bit 12. When bit 12 is one, this bit is ignored. |

***continued...***

| Bit(s) | Name | R/W | Description |
|---|---|---|---|
| | | | When bit 12 is zero, bit 5 indicates the unidirectional function:<br>• A value of 1 enables transmit from media independent interface regardless of whether the PHY has determined that a valid link has been established.<br>• A value of 0 enables transmit from media independent interface only when the PHY has determined that a valid link has been established.<br>The reset value of this bit is zero. |
| 6, 13 | SPEED_SELECTION | RO | Indicates the operating mode of the PCS function. Bits 6 and 13 are set to 1 and 0 respectively. This combination of values represent the gigabit mode.<br>Bit [6, 13]:<br>• 00: 10 Mbps<br>• 01: 100 Mbps<br>• 10: 1 Gigabit<br>• 11: Reserved |
| 7 | COLLISION_TEST | RO | The 1000BASE-X PCS function does not support half-duplex mode. This bit is always set to 0.<br>Ignore this bit if you are using SGMII PCS function. |
| 8 | DUPLEX_MODE | RO | The 1000BASE-X PCS function only supports full-duplex mode. This bit is always set to 1.<br>Ignore this bit if you are using SGMII PCS function. |
| 9 | RESTART_AUTO_NEGOTIATION | RW | Set this bit to 1 to restart the auto-negotiation sequence. For normal operation, set this bit to 0 (reset value). |
| 10 | ISOLATE | RW | Set this bit to 1 to isolate the PCS function from the MAC layer device. For normal operation, set this bit to 0 (reset value). |
| 11 | POWERDOWN | RW | Set this bit to 1 to power down the transceiver quad. The PCS function then asserts the powerdown signal to indicate the state it is in. |
| 12 | AUTO_NEGOTIATION_ENABLE | RW | Set this bit to 1 (reset value) to enable auto-negotiation. |
| 14 | LOOPBACK | RW | PHY loopback. Set this bit to 1 to implement loopback in the GX transceiver. For normal operation, set this bit to 0 (reset value). This bit is ignored if reduced ten-bit interface (RTBI) is implemented.<br>This feature is supported in all device families except the Cyclone IV GX device families. |
| 15 | RESET | RW | Self-clearing reset bit. Set this bit to 1 to generate 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). |

## 5.2.2. Status Register (Word Offset 0x01)

### Table 47. Status Register Bit Descriptions

| Bit | Name | R/W | Description |
|---|---|---|---|
| 0 | EXTENDED_CAPABILITY | RO | A value of 1 indicates that the PCS function supports extended registers. |
| 1 | JABBER_DETECT | — | Unused. Always set to 0. |
| 2 | LINK_STATUS | RO | A value of 1 indicates that a valid link is established. A value of 0 indicates an invalid link.<br>If the link synchronization is lost, a 0 is latched. |

*continued...*

| Bit | Name | R/W | Description |
|---|---|---|---|
| 3 | AUTO_NEGOTIATION_ABILITY | RO | A value of 1 indicates that the PCS function supports auto-negotiation. |
| 4 | REMOTE_FAULT | — | Unused. Always set to 0. |
| 5 | AUTO_NEGOTIATION_COMPLETE | RO | A value of 1 indicates the following status:<br>• The auto-negotiation process is completed.<br>• The auto-negotiation control registers are valid. |
| 6 | MF_PREAMBLE_SUPPRESSION | — | Unused. Always set to 0. |
| 7 | UNIDIRECTIONAL_ABILITY | RO | A value of 1 indicates that the PCS is able to transmit from MII/GMII regardless of whether the PCS has established a valid link. |
| 8 | EXTENDED_STATUS | — | Unused. Always set to 0. |
| 9 | 100BASET2_HALF_DUPLEX | RO | The PCS function does not support 100Base-T2, 10-Mbps, 100BASE-X, and 100Base-T4 operation. Always set to 0. |
| 10 | 100BASET2_FULL_DUPLEX | | |
| 11 | 10MBPS_HALF_DUPLEX | | |
| 12 | 10MBPS_FULL_DUPLEX | | |
| 13 | 100BASE-X_HALF_DUPLEX | | |
| 14 | 100BASE-X_FULL_DUPLEX | | |
| 15 | 100BASE-T4 | | |

## 5.2.3. Dev_Ability and Partner_Ability Registers (Word Offset 0x04 – 0x05)

The definition of each field in the `partner_ability` registers depends on the mode in which the PCS function operates.

In this mode, the definition of the fields in the `dev_ability` register are the same as the 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).

### 5.2.3.1. 1000BASE-X

**Table 48. Dev_Ability and Partner_Ability Registers Bits Description in 1000BASE-X**

| Bit(s) | Name | R/W | Description |
|---|---|---|---|
| 0:4 | Reserved | — | Always set these bits to 0. |
| 5 | FD | RW/RO (1), (2) | Full-duplex mode enable. A value of 1 indicates support for full duplex. |
| 6 | HD | | Half-duplex mode enable. A value of 1 indicates support for half duplex. |

*continued...*

| Bit(s) | Name | R/W | Description |
|---|---|---|---|
| 7 | PS1 | | Pause support.<br>• PS1=0 / PS2=0: Pause is not supported.<br>• PS1=0 / PS2=1: Asymmetric pause toward link partner.<br>• PS1=1 / PS2=0: Symmetric pause.<br>• PS1=1/ PS2=1: Pause is supported on transmit and receive. |
| 8 | PS2 | | |
| 9:11 | Reserved | — | Always set these bits to 0. |
| 12 | RF1 | RW/RO (1), (2) | Remote fault condition:<br>• RF1=0 / RF2=0: No error, link is valid (reset condition).<br>• RF1=0 / RF2=1: Offline.<br>• RF1=1 / RF2=0: Failure condition.<br>• RF1=1 / RF2=1: Auto-negotiation error. |
| 13 | RF2 | | |
| 14 | ACK | RO | Acknowledge. A value of 1 indicates that the device has received three consecutive matching ability values from its link partner. |
| 15 | NP | RW/RO(1) (2) | Next page. In `dev_ability` register, this bit is always set to 0. |
| Notes to Table 48 on page 97 :<br>1. All bits in the `dev_ability` register have RW access.<br>2. All bits in the `partner_ability` register are read-only. | | | |

### 5.2.3.2. SGMII MAC Mode Auto Negotiation

When the SGMII mode and the SGMII MAC mode auto-negotiation are enabled, the Triple-Speed Ethernet IP core ignores the value in the `dev_ability` register and automatically sets the value to `16'h4001` as specified in the SGMII specification for SGMII auto-negotiation.

When the auto-negotiation is complete, the Triple-Speed Ethernet IP core speed and the duplex mode will be resolved based on the value in the `partner_ability` register. The `partner_ability` register is received from the link partner during the auto-negotiation process.

**Table 49. Partner_Ability Register Bits Description in SGMII MAC Mode**

| Bit(s) | Name | R/W | Description |
|---|---|---|---|
| 9:0 | Reserved | — | — |
| 11:10 | `COPPER_SPEED[1:0]` | RO | Link partner interface speed:<br>• 00: copper interface speed is 10 Mbps<br>• 01: copper interface speed is 100 Mbps<br>• 10: copper interface speed is 1 gigabit<br>• 11: reserved |
| 12 | `COPPER_DUPLEX_STATUS` | RO | Link partner duplex capability:<br>• 1: copper interface is capable of operating in full-duplex mode<br>• 0: copper interface is capable of operating in half-duplex mode |
| 13 | Reserved | — | — |
| 14 | ACK | RO | Acknowledge. A value of 1 indicates that the link partner has received 3 consecutive matching ability values from the device. |
| 15 | `COPPER_LINK_STATUS` | RO | Copper link partner status:<br>• 1: copper interface link is up<br>• 0: copper interface link is down |

### 5.2.3.3. SGMII PHY Mode Auto Negotiation

When the SGMII mode and the SGMII PHY mode auto-negotiation is enabled, set the `dev_ability` register before the auto-negotiation process so that the link partner can identify the copper speed, duplex status, and link status.

When the auto-negotiation is complete, Triple-Speed Ethernet IP core speed and the duplex mode will be resolved based on the value that you set in the `dev_ability` register. You can get the value for the `dev_ability` register from the system level where the Triple-Speed Ethernet IP core is integrated. If the IP core is integrated in the system level with another IP that resolves the copper speed and duplex information, use these values to set the `dev_ability` register.

**Table 50. Dev_Ability Register Bits Description in SGMII PHY Mode**

| Bit(s) | Name | R/W | Description |
|---|---|---|---|
| 9:0 | Reserved | — | Always set bit 0 to 1 and bits1–9 to 0. |
| 11:10 | `SPEED[1:0]` | RW | Link partner interface speed:<br>• 00: copper interface speed is 10 Mbps<br>• 01: copper interface speed is 100 Mbps<br>• 10: copper interface speed is 1 gigabit<br>• 11: reserved |
| 12 | `COPPER_DUPLEX_STATUS` | RW | Link partner duplex capability:<br>• 1: copper interface is capable of operating in full-duplex mode<br>• 0: copper interface is capable of operating in half-duplex mode<br>• 1 Gbps speed does not support half-duplex mode. |
| 13 | Reserved | — | Always set this bit to 0. |
| 14 | ACK | RO | Acknowledge. Value as specified in the IEEE 802.3z standard. |
| 15 | `COPPER_LINK_STATUS` | RW | Copper link partner status:<br>• 1: copper interface link is up<br>• 0: copper interface link is down |

## 5.2.4. An_Expansion Register (Word Offset 0x06)

**Table 51. An_Expansion Register Description**

| Bit(s) | Name | R/W | Description |
|---|---|---|---|
| 0 | `LINK_PARTNER_AUTO_NEGOTIATION_ABLE` | RO | A value of 1 indicates that the link partner supports auto-negotiation. The reset value is 0. |
| 1 | `PAGE_RECEIVE` | RO | 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` | — | Unused. Always set to 0. |
| 15:3 | Reserved | — | — |

### 5.2.5. If_Mode Register (Word Offset 0x14)

**Table 52. IF_Mode Register Description**

| Bit(s) | Name | R/W | Description |
|---|---|---|---|
| 0 | SGMII_ENA | RW | Determines the PCS function operating mode. Setting this bit to 1 enables SGMII mode. Setting this bit to 0 enables 1000BASE-X gigabit mode. |
| 1 | USE_SGMII_AN | RW | 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. |
| 3:2 | SGMII_SPEED[1:0] | RW | SGMII speed. When the PCS function operates in SGMII mode (SGMII_ENA = 1) and programed not to be automatically configured (USE_SGMII_AN = 0), set the speed as follows:<br>• 00: 10 Mbps<br>• 01: 100 Mbps<br>• 10: 1 Gigabit<br>• 11: Reserved<br>These bits are ignored when SGMII_ENA is 0 or USE_SGMII_AN is 1. These bits are only valid if you only enable the SGMII mode and not the auto-negotiation mode. |
| 4 | SGMII_DUPLEX | RW | 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. These bits are only valid if you only enable the SGMII mode and not the auto-negotiation mode. |
| 5 | SGMII_AN_MODE | RW | SGMII auto-negotiation mode:<br>• 1: enable SGMII PHY mode<br>• 0: enable SGMII MAC mode<br>This bit resets to 0, which defaults to SGMII MAC mode. |
| 15:6 | Reserved | — | — |

## 5.3. Register Initialization

The Triple-Speed Ethernet IP core supports various types of interface commonly used by the following Ethernet solutions:

- MII/GMII
- RGMII
- 10-bit Interface
- SGMII
- 1000BASE-X
- Management Data Input/Output (MDIO) for external PHY register configuration

When using the Triple-Speed Ethernet IP core with an external interface, you must understand the requirements and initialize the registers.

Register initialization mainly performed in the following configurations:

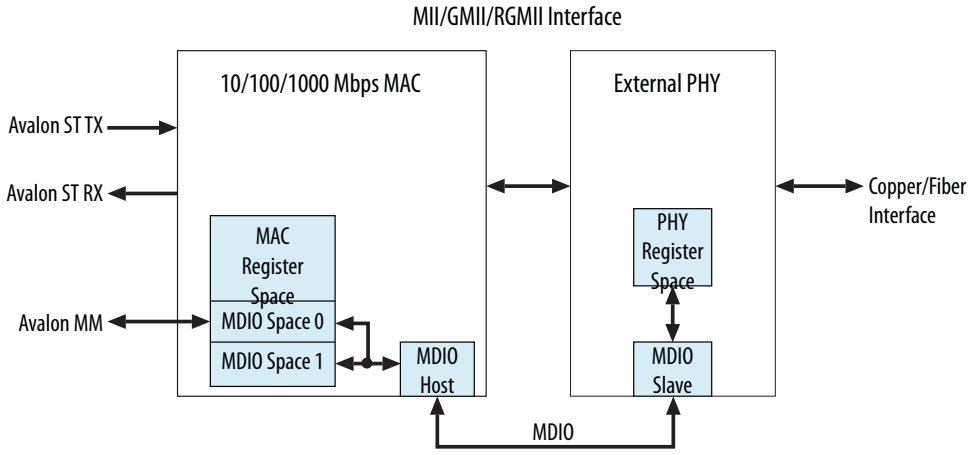
- External PHY Initialization using MDIO (Optional)
- PCS Configuration Register Initialization
- MAC Configuration Register Initialization

This section discusses the register initialization for the following examples of the Ethernet system using different MAC interfaces with recommended initialization sequences:

- Triple-Speed Ethernet System with MII/GMII or RGMII on page 101
- Triple-Speed Ethernet System with SGMII on page 103
- Triple-Speed Ethernet System with 1000BASE-X Interface on page 104

## 5.3.1. Triple-Speed Ethernet System with MII/GMII or RGMII

**Figure 43. Triple-Speed Ethernet System with MII/GMII or RGMII with Register Initialization Recommendation**




Use the following recommended initialization sequences for the example shown in the figure above.

1. External PHY Initialization using MDIO

   //Assume the External PHY Address is 0x0A

   mdio_addr0 = 0x0A

   //External PHY Register will Map to MDIO Space 0

   Read/write to MDIO space 0 (dword offset 0x80 - 0x9F) = Read/write to PHY Register 0 to 31

2. MAC Configuration Register Initialization

   a. Disable MAC Transmit and Receive DatapathDisable the MAC transmit and receive datapath before performing any changes to configuration.

      //Set TX_ENA and RX_ENA bit to 0 in Command Config Register

      Command_config Register = 0x00802220

      //Read the TX_ENA and RX_ENA bit is set 0 to ensure TX and RX path is disable

      Wait Command_config Register = 0x00802220

   b. MAC FIFO Configuration

      Tx_section_empty = Max FIFO size - 16

Tx_almost_full = 3

Tx_almost_empty = 8

Rx_section_empty = Max FIFO size - 16

Rx_almost_full = 8

Rx_almost_empty = 8

//Cut Through Mode, Set this Threshold to 0 to enable Store and Forward Mode

Tx_section_full = 16

//Cut Through Mode, Set this Threshold to 0 to enable Store and Forward Mode

Rx_section_full = 16

c. MAC Address Configuration

   //MAC address is 00-1C-23-17-4A-CB

   mac_0 = 0x17231C00

   mac_1 = 0x0000CB4A

d. MAC Function Configuration

   //Maximum Frame Length is 1518 bytes

   Frm_length = 1518

   //Minimum Inter Packet Gap is 12 bytes

   Tx_ipg_length = 12

   //Maximum Pause Quanta Value for Flow Control

   Pause_quant = 0xFFFF

   //Set the MAC with the following option:

   // 100Mbps, User can get this information from the PHY status/PCS status

   //Full Duplex, User can get this information from the PHY status/PCS status

   //Padding Removal on Receive

   //CRC Removal

   //TX MAC Address Insertion on Transmit Packet

   //Select mac_0 and mac_1 as the source MAC Address

   Command_config Register = 0x00800220

e. Reset MAC

   Intel recommends that you perform a software reset when there is a change in the MAC speed or duplex. The MAC software reset bit self-clears when the software reset is complete.

   //Set SW_RESET bit to 1

   Command_config Register = 0x00802220

   Wait Command_config Register = 0x00800220

f. Enable MAC Transmit and Receive Datapath

   //Set TX_ENA and RX_ENA to 1 in Command Config Register

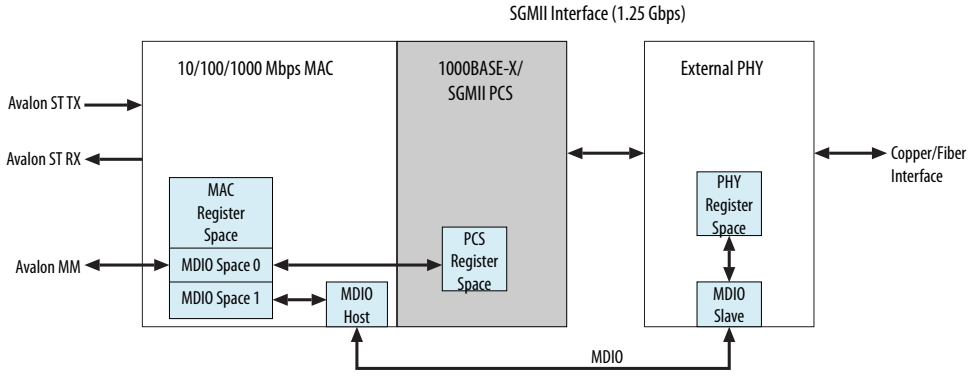
Command_config Register = 0x00800223

//Read the TX_ENA and RX_ENA bit is set 1 to ensure TX and RX path is enable

Wait Command_config Register = 0x00800223

## 5.3.2. Triple-Speed Ethernet System with SGMII

**Figure 44. Triple-Speed Ethernet System with SGMII with Register Initialization Recommendation**




Use the following recommended initialization sequences for the example shown in the figure above.

1. External PHY Initialization using MDIO

   Refer to step 1 in Triple-Speed Ethernet System with MII/GMII or RGMII on page 101.

2. PCS Configuration Register Initialization
   a. Set Auto Negotiation Link Timer

      //Set Link timer to 1.6ms for SGMII

      link_timer (address offset 0x12) = 0x0D40

      Link_timer (address offset 0x13) = 0x03

   b. Configure SGMII

      //Enable SGMII Interface and Enable SGMII Auto Negotiation

      //SGMII_ENA = 1, USE_SGMII_AN = 1

      if_mode = 0x0003

   c. Enable Auto Negotiation

      //Enable Auto Negotiation

      //AUTO_NEGOTIATION_ENA = 1, Bit 6,8,13 can be ignore

      PCS Control Register = 0x1140

   d. PCS Reset

      //PCS Software reset is recommended where there any configuration changed

      //RESET = 1

      PCS Control Register = 0x9140

Wait PCS Control Register RESET bit is clear

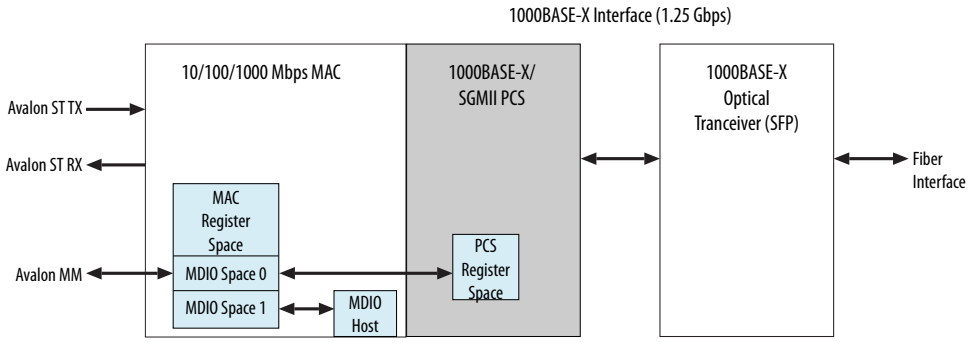
3. MAC Configuration Register Initialization

   Refer to step 2 in Triple-Speed Ethernet System with MII/GMII or RGMII on page 101.

*Note:* If 1000BASE-X/SGMII PCS is initialized, set the ETH_SPEED (bit 3) and ENA_10 (bit 25) in `command_config` register to 0. If half duplex is reported in the PHY/PCS status register, set the HD_ENA (bit 10) to 1 in `command_config` register.

## 5.3.3. Triple-Speed Ethernet System with 1000BASE-X Interface

**Figure 45. Triple-Speed Ethernet System with 1000BASE-X Interface with Register Initialization Recommendation**




Use the following recommended initialization sequences for the example shown in the figure above.

1. External PHY Initialization using MDIO

   Refer to step 1 in Triple-Speed Ethernet System with MII/GMII or RGMII on page 101.

2. PCS Configuration Register Initialization

   a. Set Auto Negotiation Link Timer

      //Set Link timer to 10ms for 1000BASE-X

      link_timer (address offset 0x12) = 0x12D0

      link_timer (address offset 0x13) = 0x13

   b. Configure SGMII

      //1000BASE-X/SGMII PCS is default in 1000BASE-X Mode

      //SGMII_ENA = 0, USE_SGMII_AN = 0

      if_mode = 0x0000

   c. Enable Auto Negotiation

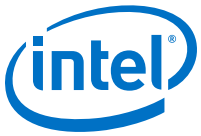
      //Enable Auto Negotiation

      //AUTO_NEGOTIATION_ENA = 1, Bit 6,8,13 is Read Only

      PCS Control Register = 0x1140

   d. PCS Reset

      //PCS Software reset is recommended where there any configuration changed



```
//RESET = 1
PCS Control Register = 0x9140
Wait PCS Control Register RESET bit is clear
```

3. MAC Configuration Register Initialization

   Refer to step 2 in Triple-Speed Ethernet System with MII/GMII or RGMII on page 101.

*Note:* If 1000BASE-X/SGMII PCS is initialized, set the ETH_SPEED (bit 3) and ENA_10 (bit 25) in `command_config` register to 0. If half duplex is reported in the PHY/PCS status register, set the HD_ENA (bit 10) to 1 in `command_config` register.

# 6. Interface Signals

## 6.1. Interface Signals

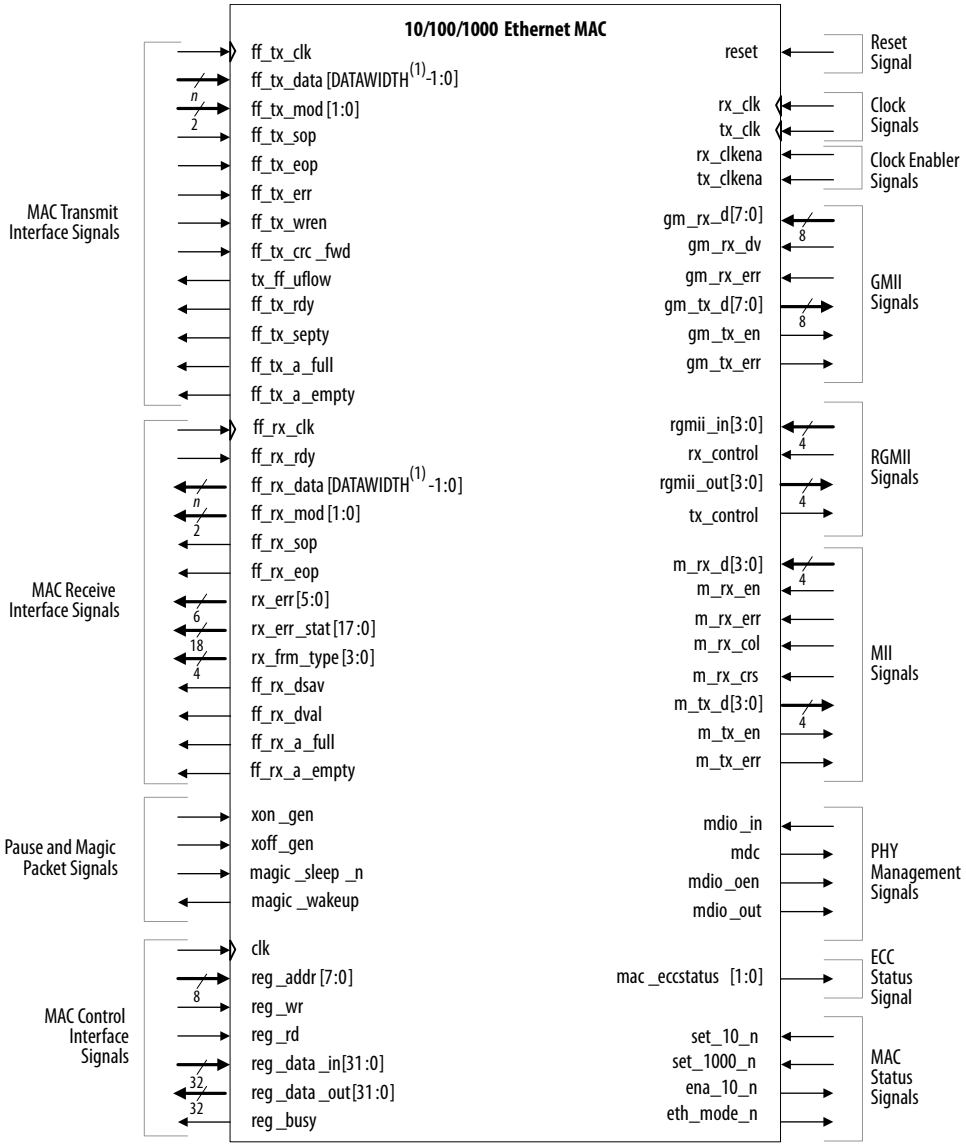
The following sections describe the Triple-Speed Ethernet Intel FPGA IP core interface signals:


- 10/100/1000 Ethernet MAC Signals on page 107
- 10/100/1000 Multiport Ethernet MAC Signals on page 115
- 10/100/1000 Ethernet MAC with 1000BASE-X/SGMII PCS Signals on page 118
- 10/100/1000 Multiport Ethernet MAC with 1000BASE-X/SGMII PCS Signals on page 124
- 10/100/1000 Ethernet MAC with 1000BASE-X/SGMII 2XTBI PCS Signals on page 122
- 10/100/1000 Ethernet MAC with 1000BASE-X/SGMII PCS and Embedded PMA Signals on page 126
- 10/100/1000 Multiport Ethernet MAC with 1000BASE-X/SGMII PCS and Embedded PMA on page 129
- 1000BASE-X/SGMII PCS Signals on page 134
- 1000BASE-X/SGMII 2XTBI PCS Signals on page 138
- 1000BASE-X/SGMII PCS and PMA Signals on page 140


*Note:* To view all the interface signal names, turn on **Show Signals** in the **Block Diagram** tab in the Triple-Speed Ethernet Intel FPGA IP parameter editor interface. Otherwise, only the connection signal names are shown.

## 6.1.1. 10/100/1000 Ethernet MAC Signals

**Figure 46. 10/100/1000 Ethernet MAC Function with Internal FIFO Buffers Signals**




Note to 10/100/1000 Ethernet MAC Function with Internal FIFO Buffers Signals:

1. The DATAWIDTH value depends on the FIFO width that you select in the parameter editor. Options available are 8 and 32 bits.

### 6.1.1.1. Clock and Reset Signals

Data transfers on the MAC Ethernet-side interface are synchronous to the receive and transmit clocks.

### Table 53. GMII/RGMII/MII Clock Signals

| Name | I/O | Description |
|---|---|---|
| tx_clk (In Platform Designer: pcs_mac_tx_clock_connection) | I | 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 Platform Designer: pcs_mac_rx_clock_connection) | I | 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 54. Reset Signal

| Name | I/O | Description |
|---|---|---|
| reset | I | Assert this signal to reset all logic in the MAC and PCS control interface. The signal must be asserted for at least three clock cycles. |

## 6.1.1.2. Clock Enabler Signals

### Table 55. Clock Enabler Signals

| Name | I/O | Description |
|---|---|---|
| tx_clkena | I | Clock enable from the PHY IP. When you turn on the **Use clock enable for MAC** parameter, this signal is used together with tx_clk and rx_clk to generate 125 MHz, 25 MHz, and 2.5 MHz clocks. (7) |
| rx_clkena | I | Clock enable from the PHY IP. When you turn on the **Use clock enable for MAC** parameter, this signal is used together with tx_clk and rx_clk to generate 125 MHz, 25 MHz, and 2.5 MHz clocks. (8) |

## 6.1.1.3. MAC Control Interface Signals

The MAC control interface is an Avalon Memory-Mapped slave port that provides access to the register space.

### Table 56. MAC Control Interface Signals

| Name | Avalon Memory-Mapped Signal Type | I/O | Description |
|---|---|---|---|
| clk | clk | I | Register access reference clock. Set the signal to a value less than or equal to 125 MHz. |
| reg_wr | write | I | Register write enable. |
| reg_rd | read | I | Register read enable. |
| reg_addr[7:0] | address | I | 32-bit word-aligned register address. |

*continued...*

(7) For configurations without internal FIFO, this signal is called tx_clkena_<n>.

(8) For configurations without internal FIFO, this signal is called rx_clkena_<n>.

| Name | Avalon Memory-Mapped Signal Type | I/O | Description |
|---|---|---|---|
| reg_data_in[31:0] | writedata | I | Register write data. Bit 0 is the least significant bit. |
| reg_data_out[31:0] | readdata | O | Register read data. Bit 0 is the least significant bit. |
| reg_busy | waitrequest | O | Register interface busy. Asserted during register read or register write access; deasserted when the current register access completes. |

### 6.1.1.4. MAC Status Signals

The MAC status signals which allow you to set the transfer mode of the Ethernet-side interface.

**Table 57. MAC Status Signals**

| Name | I/O | Description |
|---|---|---|
| eth_mode | O | Ethernet mode. This signal is set to 1 when the MAC function is configured to operate at 1000 Mbps; set to 0 when it is configured to operate at 10/100 Mbps. |
| ena_10 | O | 10 Mbps enable. This signal is set to 1 to indicate that the PHY interface should operate at 10 Mbps. Valid only when the eth_mode signal is set to 0. |
| set_1000 | I | Gigabit mode selection. Can be driven to 1 by an external device, for example a PHY device, to set the MAC function to operate in gigabit. When set to 0, the MAC is set to operate in 10/100 Mbps. This signal is ignored when the ETH_SPEED bit in the command_config register is set to 1. |
| set_10 | I | 10 Mbps selection. Can be driven to 1 by an external device, for example a PHY device, to indicate that the MAC function is connected to a 10-Mbps PHY device. When set to 0, the MAC function is set to operate in 100-Mbps or gigabit mode. This signal is ignored when the ETH_SPEED or ENA_10 bit in the command_config register is set to 1. The ENA_10 bit has a higher priority than this signal. |

### 6.1.1.5. MAC Receive Interface Signals

**Table 58. MAC Receive Interface Signals**

| Name | Avalon Streaming Signal Type | I/O | Description |
|---|---|---|---|
| **Avalon Streaming Signals** | | | |
| ff_rx_clk (In Platform Designer: receive_clock_connection) | clk | I | Receive clock. All signals on the Avalon Streaming receive interface are synchronized on the rising edge of this clock. Set this clock to the frequency required to get the desired bandwidth on this interface. This clock can be completely independent from rx_clk. |
| ff_rx_dval | valid | O | Receive data valid. When asserted, this signal indicates that the data on the following signals are valid: 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]. |
| ff_rx_data [(DATAWIDTH-1):0] | data | O | 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. |

*continued...*

| Name | Avalon Streaming Signal Type | I/O | Description |
|---|---|---|---|
| ff_rx_mod[1:0] | empty | O | Receive data modulo. Indicates invalid bytes in the final frame word:<br>• 11: ff_rx_data[23:0] is not valid<br>• 10: ff_rx_data[15:0] is not valid<br>• 01: ff_rx_data[7:0] is not valid<br>• 00: ff_rx_data[31:0] is valid<br>This signal applies only when DATAWIDTH is set to 32. |
| ff_rx_sop | startofpacket | O | Receive start of packet. Asserted when the first byte or word of a frame is driven on ff_rx_data[(DATAWIDTH-1):0]. |
| ff_rx_eop | endofpacket | O | Receive end of packet. Asserted when the last byte or word of frame data is driven on ff_rx_data[(DATAWIDTH-1):0]. |
| ff_rx_rdy | ready | I | Receive application ready. Assert this signal on the rising edge of ff_rx_clk when the user application is ready to receive data from the MAC function. |
| rx_err[5:0] | error | O | Receive error. Asserted with the final byte in the frame to indicate that an error was detected when receiving the frame. See Table 60 on page 111 for the bit description. |
| **Component-Specific Signals** | | | |
| ff_rx_dsav | — | O | Receive frame available. When asserted, this signal indicates that the internal receive FIFO buffer contains some data to be read but not necessarily a complete frame. The user application may want to start reading from the FIFO buffer.<br>This signal remains deasserted in the store and forward mode. |
| rx_frm_type[3:0] | — | O | Frame type. See Table 59 on page 110 for the bit description. |
| ff_rx_a_full | — | O | Asserted when the FIFO buffer reaches the almost-full threshold. |
| ff_rx_a_empty | — | O | Asserted when the FIFO buffer goes below the almost-empty threshold. |
| rx_err_stat[17:0] | — | O | rx_err_stat[17]: One indicates that the receive frame is a stacked VLAN frame.<br>rx_err_stat[16]: One indicates that the receive frame is either a VLAN or stacked VLAN frame.<br>rx_err_stat[15:0]: The value of the length/type field of the receive frame. |

## Table 59. rx_frm_type Bit Description

| Bit | Description |
|---|---|
| 3 | Indicates VLAN frames. Asserted with ff_rx_sop and remains asserted until the end of the frame. |
| 2 | Indicates broadcast frames. Asserted with ff_rx_sop and remains asserted until the end of the frame. |
| 1 | Indicates multicast frames. Asserted with ff_rx_sop and remains asserted until the end of the frame. |
| 0 | Indicates unicast frames. Asserted with ff_rx_sop and remains asserted until the end of the frame. |

### Table 60. rx_err Bit Description

| Bit | Description |
|---|---|
| 5 | Collision error. Asserted when the frame was received with a collision. |
| 4 | Corrupted receive frame caused by PHY or PCS error. Asserted when the error is detected on the MII/GMII/RGMII. |
| 3 | Truncated receive frame. Asserted when the receive frame is truncated due to an overflow in the receive FIFO buffer. |
| 2 (1) | CRC error. Asserted when the frame is received with a CRC-32 error. This error bit applies only to frames with a valid length. Refer to Length Checking on page 42. |
| 1 (1) | Invalid length error. Asserted when the receive frame has an invalid length as defined by the IEEE Standard 802.3. For more information on the frame length, refer to Length Checking on page 42 . |
| 0 | Receive frame error. Indicates that an error has occurred. It is the logical OR of `rx_err[5:1]`. |

Note to Table 60 on page 111 :

1. Bits 1 and 2 are not mutually exclusive. Ignore CRC error `rx_err[2]` signal if it is asserted at the same time as the invalid length error `rx_err[1]` signal.

## 6.1.1.6. MAC Transmit Interface Signals

### Table 61. MAC Transmit Interface Signals

| Name | Avalon Streaming Signal Type | I/O | Description |
|---|---|---|---|
| **Avalon Streaming Signals** | | | |
| `ff_tx_clk`<br>(In Platform Designer: `transmit_clock_connection`) | `clk` | I | Transmit clock. All transmit signals are synchronized on the rising edge of this clock.<br>Set this clock to the required frequency to get the desired bandwidth on the Avalon Streaming transmit interface. This clock can be completely independent from `tx_clk`. |
| `ff_tx_wren` | `valid` | I | Transmit data write enable. Assert this signal to indicate that the data on the following signals are valid: `ff_tx_data[(DATAWIDTH-1):0]`, `ff_tx_sop`, and `ff_tx_eop`.<br>In cut-through mode, keep this signal asserted throughout the frame transmission. Otherwise, the frame is truncated and forwarded to the Ethernet-side interface with an error. |
| `ff_tx_data[(DATAWIDTH-1):0]` | `data` | I | Transmit data. `DATAWIDTH` can be either 8 or 32 depending on the FIFO data width configured. 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` | I | Transmit data modulo. Indicates invalid bytes in the final frame word:<br>• 11: `ff_tx_data[23:0]` is not valid<br>• 10: `ff_tx_data[15:0]` is not valid<br>• 01: `ff_tx_data[7:0]` is not valid<br>• 00: `ff_tx_data[31:0]` is valid<br>This signal applies only when `DATAWIDTH` is set to 32. |
| `ff_tx_sop` | `startofpacket` | I | Transmit start of packet. Assert this signal when the first byte in the frame (the first byte of the destination address) is driven on `ff_tx_data`. |

*continued...*

| Name | Avalon Streaming Signal Type | I/O | Description |
|---|---|---|---|
| ff_tx_eop | endofpacket | I | Transmit end of packet. Assert this signal when the last byte in the frame (the last byte of the FCS field) is driven on ff_tx_data. |
| ff_tx_err | error | I | Transmit frame error. Assert this signal with the final byte in the frame to indicate that the transmit frame is invalid. The MAC function forwards the invalid frame to the GMII with an error. |
| ff_tx_rdy | ready | O | MAC ready. When asserted, the MAC function is ready to accept data from the user application. |
| **Component-Specific Signals** | | | |
| ff_tx_crc_fwd | — | I | Transmit CRC insertion. Set this signal to 0 when ff_tx_eop is set to 1 to instruct the MAC function to compute a CRC and insert it into the frame. If this signal is set to 1, the user application is expected to provide the CRC. |
| tx_ff_uflow | — | O | Asserted when an underflow occurs on the transmit FIFO buffer. |
| ff_tx_septy | — | O | Deasserted when the FIFO buffer is filled to or above the section-empty threshold defined in the tx_section_empty register. User applications can use this signal to indicate when to stop writing to the FIFO buffer and initiate backpressure. |
| ff_tx_a_full | — | O | Asserted when the transmit FIFO buffer reaches the almost- full threshold. |
| ff_tx_a_empty | — | O | Asserted when the transmit FIFO buffer goes below the almost-empty threshold. |

### 6.1.1.7. Pause and Magic Packet Signals

The pause and magic packet signals are component-specific signals.

**Table 62. Pause and Magic Packet Signals**

| Name | I/O | Description |
|---|---|---|
| xon_gen | I | Assert this signal for at least 1 tx_clk clock cycle to trigger the generation of a pause frame with a 0 pause quanta. The MAC function generates the pause frame independent of the status of the receive FIFO buffer.<br>This signal is not in use in the following conditions:<br>• Ignored when the xon_gen bit in the command_config register is set to 1.<br>• Absent when the **Enable full duplex flow control** option is turned off. |
| xoff_gen | I | Assert this signal for at least one tx_clk clock cycle to trigger the generation of a pause frame with a pause quanta configured in the pause_quant register. The MAC function generates the pause frame independent of the status of the receive FIFO buffer.<br>This signal is not in use in the following conditions:<br>• Ignored if the xoff_gen bit in the command_config register is set to 1.<br>• Absent when the **Enable full duplex flow control** option is turned off. |
| magic_sleep_n | I | Assert this active-low signal to put the node into a power-down state.<br>If magic packets are supported (the MAGIC_ENA bit in the command_config register is set to 1), the receiver logic stops writing data to the receive FIFO buffer and the magic packet detection logic is enabled. Setting this signal to 1 restores the normal frame reception mode. |

*continued...*

| Name | I/O | Description |
|---|---|---|
| | | This signal is present only if the **Enable magic packet detection** option is turned on. |
| magic_wakeup | O | If the MAC function is in the power-down state, the MAC function asserts this signal to indicate that a magic packet has been detected and the node is requested to restore its normal frame reception mode.<br>This signal is present only if the **Enable magic packet detection** option is turned on. |

### 6.1.1.8. MII/GMII/RGMII Signals

**Table 63. GMII/RGMII/MII Signals**

| Name | I/O | Description |
|---|---|---|
| **GMII Transmit** | | |
| gm_tx_d[7:0] | I | GMII transmit data bus. |
| gm_tx_en | O | Asserted to indicate that the data on the GMII transmit data bus is valid. |
| gm_tx_err | O | Asserted to indicate to the PHY that the frame sent is invalid. |
| **GMII Receive** | | |
| gm_rx_d[7:0] | I | GMII receive data bus. |
| gm_rx_dv | I | Assert this signal to indicate that the data on the GMII receive data bus is valid. Keep this signal asserted during frame reception, from the first preamble byte until the last byte of the CRC field is received. |
| gm_rx_err | I | The PHY asserts this signal to indicate that the receive frame contains errors. |
| **RGMII Transmit** | | |
| rgmii_out[3:0] | O | 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 | O | Control output signal. Drives gm_tx_en on the 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] | I | 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 | I | RGMII control input signal. Expects gm_rx_dv on the 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] | O | MII transmit data bus. |
| m_tx_en | O | Asserted to indicate that the data on the MII transmit data bus is valid. |
| m_tx_err | O | Asserted to indicate to the PHY device that the frame sent is invalid. |
| **MII Receive** | | |
| m_rx_d[3:0] | I | MII receive data bus. |
| m_rx_en | I | Assert this signal to indicate that the data on the MII receive data bus is valid. Keep this signal asserted during frame reception, from the first preamble byte until the last byte of the CRC field is received. |

*continued...*

| Name | I/O | Description |
|---|---|---|
| m_rx_err | I | The PHY asserts this signal to Indicate that the receive frame contains errors. |
| **MII PHY Status** | | |
| m_rx_col | I | Collision detection. The PHY asserts this signal to indicate a collision during frame transmission. This signal is not used in full- duplex or gigabit mode. |
| m_rx_crs | I | Carrier sense detection. The PHY asserts this signal to indicate that it has detected transmit or receive activity on the Ethernet line. This signal is not used in full-duplex or gigabit mode. |

### 6.1.1.9. PHY Management Signals

**Table 64. PHY Management Interface Signals**

| Name | I/O | Description |
|---|---|---|
| mdio_in | I | Management data input. |
| mdio_out | O | Management data output. |
| mdio_oen | O | An active-low signal that enables mdio_in or mdio_out. For more information about the MDIO connection, refer to MDIO Connection on page 54. |
| mdc | O | Management data clock. Generated from the Avalon Memory-Mapped interface clock signal, clk. Specify the division factor using the **Host clock divisor** parameter such that the frequency of this clock does not exceed 2.5 MHz. For more information about the parameters, refer to Ethernet MAC Options on page 27.<br>A data bit is shifted in/out on each rising edge of this clock. All fields are shifted in and out starting from the most significant bit. |

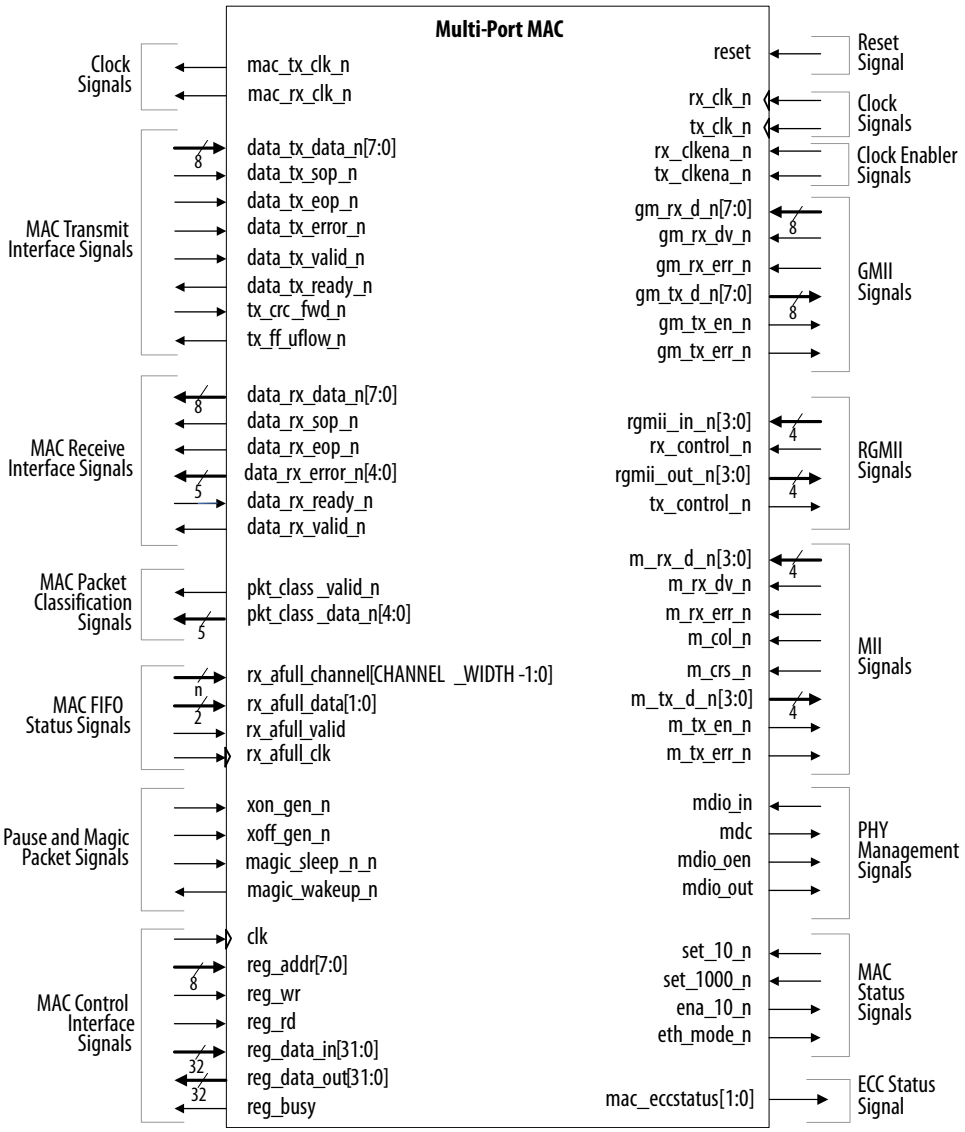
### 6.1.1.10. ECC Status Signals

**Table 65. ECC Status Signals**

| Name | I/O | Description |
|---|---|---|
| mac_eccstatus[1:0] | O | Indicates the ECC status. This signal is synchronized to the reg_clk clock domain.<br>• 11: An uncorrectable error occurred and the error data appears at the output.<br>• 10: A correctable error occurred and the error has been corrected at the output. However, the memory array has not been updated.<br>• 01: Not valid.<br>• 00: No error. |

## 6.1.2. 10/100/1000 Multiport Ethernet MAC Signals

**Figure 47. 10/100/1000 Multiport Ethernet MAC Function without Internal FIFO Buffers Signals**




### 6.1.2.1. Multiport MAC Clock and Reset Signals

**Table 66. Clock Signals**

| Name | Avalon Streaming Signal Type | I/O | Description |
|---|---|---|---|
| mac_rx_clk_n | clk | O | Receive MAC clock (2.5/25/125 MHz) for the Avalon Streaming receive data and receive packet classification interfaces. |
| mac_tx_clk_n | clk | O | Transmit MAC clock (2.5/25/125 MHz) for the Avalon Streaming transmit data interface. |

### 6.1.2.2. Multiport MAC Receive Interface Signals

**Table 67. MAC Receive Interface Signals**

| Name | Avalon Streaming Signal Type | I/O | Description |
|---|---|---|---|
| data_rx_valid_n | valid | O | Receive data valid. When asserted, this signal indicates that the data on the following signals are valid: data_rx_data_n, data_rx_sop_n, data_rx_eop_n, and data_rx_error_n. |
| data_rx_data_n[7:0] | data | O | Receive data. |
| data_rx_sop_n | startofpacket | O | Receive start of packet. Asserted when the first byte or word of a frame is driven on data_rx_data_n. |
| data_rx_eop_n | endofpacket | O | Receive end of packet. Asserted when the last byte or word of frame data is driven on data_rx_data_n. |
| data_rx_ready_n | ready | I | Receive application ready. Assert this signal on the rising edge of data_rx_clk_n when the user application is ready to receive data from the MAC function.<br>If the user application is not ready to receive data, the packet is dropped or truncated with an error. |
| data_rx_error_n[4:0] | error | O | 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 MAC Receive Interface Signals on page 109 . Bit 4 of this signal maps to bit 5 in the table and so forth. |

### 6.1.2.3. Multiport MAC Transmit Interface Signals

**Table 68. MAC Transmit Interface Signals**

| Name | Avalon Streaming Signal Type | I/O | Description |
|---|---|---|---|
| **Avalon Streaming Signals** | | | |
| data_tx_valid_n | valid | I | Transmit data valid. Assert this signal to indicate that the data on the following signals are valid: data_tx_data_n, data_tx_sop_n, data_tx_eop_n, and data_tx_error_n. |
| data_tx_data_n[7:0] | data | I | Transmit data. |
| data_tx_sop_n | startofpacket | I | Transmit start of packet. Assert this signal when the first byte in the frame is driven on data_tx_data_n. |
| data_tx_eop_n | endofpacket | I | Transmit end of packet. Assert this signal 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[4:0] | error | I | Transmit frame error. Assert this signal with the final byte in the frame to indicate that the transmit frame is invalid. The MAC function then forwards the frame to the GMII with error. |
| data_tx_ready_n | ready | O | MAC ready. When asserted, this signal indicates that the MAC function is ready to accept data from the user application. |
| **Component-Specific Signal** | | | |
| tx_crc_fwd_n | – | I | Transmit CRC insertion. Assert this active-low signal when data_tx_eop_n is asserted for the MAC function to compute the CRC and insert it into the frame. Otherwise, the user application is expected to provide the CRC. |

### 6.1.2.4. Multiport MAC Packet Classification Signals

The MAC packet classification interface is an Avalon Streaming source port which streams out receive packet classifications.

**Table 69. MAC Packet Classification Signals**

| Name | Avalon Streaming Signal Type | I/O | Description |
|---|---|---|---|
| pkt_class_valid_n | valid | O | When asserted, this signal indicates that the data on the following signals are valid: data_tx_data_n, data_tx_sop_n, data_tx_eop_n, and data_tx_error_n. |
| pkt_class_data_n[4:0] | data | O | Classification presented at the beginning of each packet:<br>Bit 4—Set to 1 for unicast frames.<br>Bit 3—Set to 1 for broadcast frames.<br>Bit 2—Set to 1 for multicast frames.<br>Bit 1—Set to 1 for VLAN frames.<br>Bit 0—Set to 1 for stacked VLAN frames. |

### 6.1.2.5. Multiport MAC FIFO Status Signals

The MAC FIFO status interface is an Avalon Streaming sink port which streams in information on the fill level of the external FIFO buffer to the MAC function.

**Table 70. MAC FIFO Status Signals**

| Signal Name | Avalon Streaming Signal Type | I/O | Description |
|---|---|---|---|
| rx_afull_valid_n | valid | I | Assert this signal to indicate that the fill level of the external FIFO buffer, rx_afull_data_n[1:0], is valid. |
| rx_afull_data_n[1:0] | data | I | Carries the fill level of the external FIFO buffer:<br>rx_afull_data_n[1]—Set to 1 if the external receive FIFO buffer reaches the initial warning level indicating that it is almost full. Upon detecting this, the MAC function generates pause frames.<br>rx_afull_data_n[0]—Set to 1 if the external receive FIFO buffer reaches the critical level before it overflows. The FIFO buffer can be considered overflow if this bit is set to 1 in the middle of a packet transfer. |
| rx_afull_channel[(CHANNEL_WIDTH-1):0] | channel | I | The port number the status applies to. |
| rx_afull_clk | clk | I | The clock that drives the MAC FIFO status interface. |

**Table 71. References**

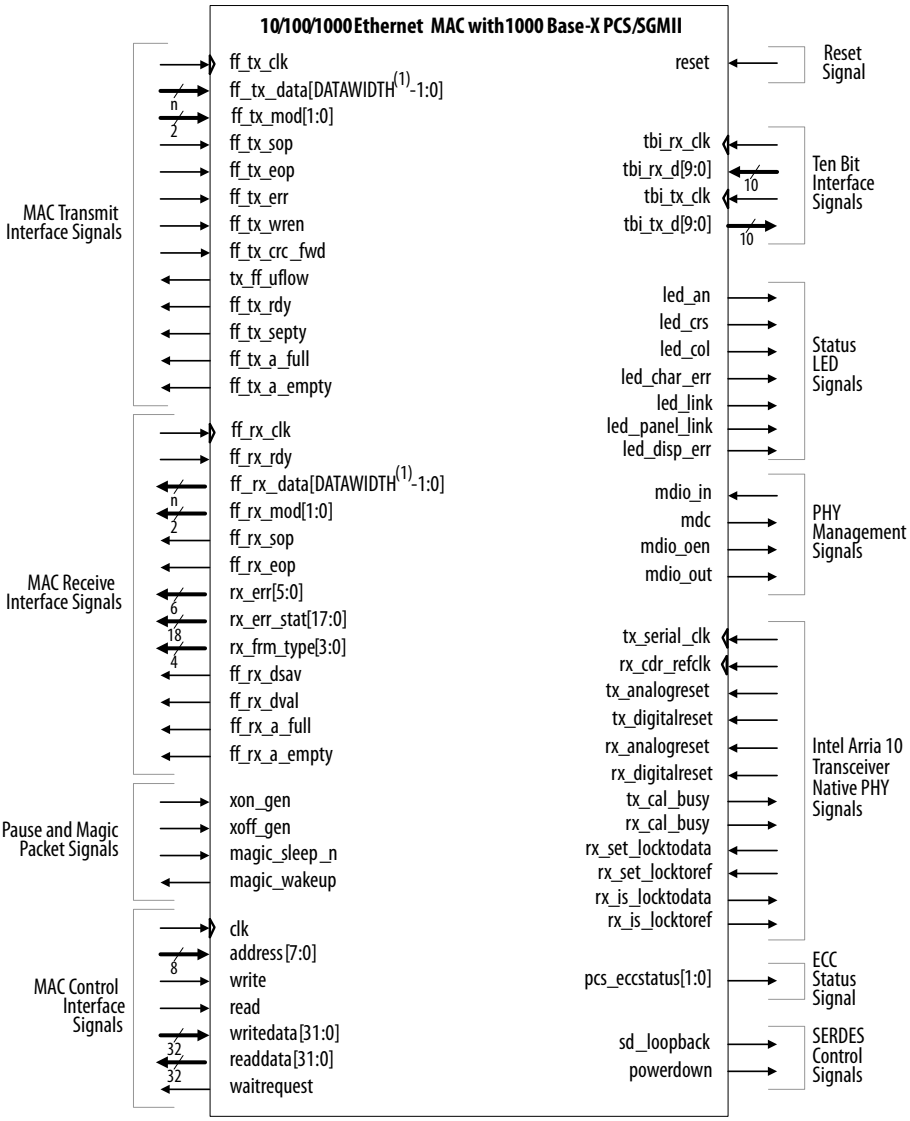
| Interface Signal | Section |
|---|---|
| Clock and reset signals | Clock and Reset Signals on page 107 |
| MAC control interface | MAC Control Interface Signals on page 108 |
| MAC transmit interface | MAC Transmit Interface Signals on page 111 |
| MAC receive interface | MAC Receive Interface Signals on page 109 |
| Status signals | MAC Status Signals on page 109 |
| Pause and magic packet signals | Pause and Magic Packet Signals on page 112 |

continued...

| Interface Signal | Section |
|---|---|
| MII/GMII/RGMII interface | MII/GMII/RGMII Signals on page 113 |
| PHY management signals | PHY Management Signals on page 114 |
| ECC status signals | ECC Status Signals on page 114 |

## 6.1.3. 10/100/1000 Ethernet MAC with 1000BASE-X/SGMII PCS Signals

**Figure 48. 10/100/1000 Ethernet MAC Function with Internal FIFO Buffers, with 1000BASE-X/SGMII PCS Signals**



Note to 10/100/1000 Ethernet MAC Function with Internal FIFO Buffers, with 1000BASE-X/SGMII PCS Signals:

1. The DATAWIDTH value depends on the FIFO width that you select in the parameter editor. Options available are 8 and 32 bits.

### 6.1.3.1. 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 72. TBI Interface Signals for External SERDES Chip**

| Name | I/O | Description |
|---|---|---|
| tbi_tx_d[9:0] | O | TBI transmit data. The PCS function transmits data on this bus synchronous to tbi_tx_clk. |
| tbi_tx_clk | I | 125-MHz TBI transmit clock from external SERDES, typically sourced by the local reference clock oscillator. |
| tbi_rx_clk | I | 125-MHz TBI receive clock from external SERDES, typically sourced by the line clock recovered from the encoded line stream. |
| tbi_rx_d[9:0] | I | TBI receive data. This bus carries the data from the external SERDES. Synchronize the bus with tbi_rx_clk. The data can be arbitrary aligned. |

### 6.1.3.2. Status LED Control Signals

**Table 73. Status LED Interface Signals**

| Name | I/O | Description | |
|---|---|---|---|
| led_link | O | When asserted, this signal indicates a successful link synchronization. | |
| led_panel_link | O | When asserted, this signal indicates the following behavior: | |
| | | **Mode** | **Signal Behavior** |
| | | 1000 Base-X without auto negotiation | Similar to led_link |
| | | SGMII mode without auto negotiation | Similar to led_link |
| | | 1000 Base-X with auto negotiation | Similar to led_an |
| | | SGMII mode with MAC mode auto negotiation | Similar to led_an and partner_ability [15] |
| | | SGMII mode with PHY mode auto negotiation | Similar to led_an and dev_ability [15] |
| led_crs | O | When asserted, this signal indicates some activities on the transmit and receive paths. When deasserted, it indicates no traffic on the paths. | |
| led_col | O | When asserted, this signal indicates that a collision was detected during frame transmission. This signal is always deasserted when the PCS function operates in standard 1000BASE-X mode or in full-duplex mode when SGMII is enabled. | |

*continued...*

| Name | I/O | Description |
|---|---|---|
| led_an | O | Auto-negotiation status. The PCS function asserts this signal when an auto-negotiation completes. |
| led_char_err | O | 10-bit character error. Asserted for one tbi_rx_clk cycle when an erroneous 10-bit character is detected. |
| led_disp_err | O | 10-bit running disparity error. Asserted for one 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. |

### 6.1.3.3. SERDES Control Signals

**Table 74. SERDES Control Signal**

| Name | I/O | Description |
|---|---|---|
| powerdown | O | Power-down enable. Asserted when the PCS function is in power-down mode; deasserted when the PCS function is operating in normal mode. This signal is implemented only when an external SERDES is used. |
| sd_loopback | O | SERDES Loopback Control. Asserted when the PCS function operates in loopback mode. You can use this signal to configure an external SERDES device to operate in loopback mode. |

### 6.1.3.4. Intel Arria 10 Transceiver Native PHY Signals

**Table 75. Intel Arria 10 Transceiver Native PHY Signals**

| Name | I/O | Description |
|---|---|---|
| tx_serial_clk | I | Serial clock input from the transceiver PLL. The frequency of this clock is 1250 MHz and the division factor is fixed to divide by 2. |
| rx_cdr_refclk | I | Reference clock input to the receive clock data recovery (CDR) circuitry. The frequency of this clock is 125 MHz. |
| tx_analogreset | I | Resets the analog transmit portion of the transceiver PHY. |
| tx_digitalreset | I | Resets the digital transmit portion of the transceiver PHY. |
| rx_analogreset | I | Resets the analog receive portion of the transceiver PHY. |
| rx_digitalreset | I | Resets the digital receive portion of the transceiver PHY. |
| tx_cal_busy | O | When asserted, this signal indicates that the transmit channel is being calibrated. |
| rx_cal_busy | O | When asserted, this signal indicates that the receive channel is being calibrated. |
| rx_set_locktodata | I | Force the receiver CDR to lock to the incoming data. |
| rx_set_locktoref | I | Force the receiver CDR to lock to the phase and frequency of the input reference clock. |
| rx_is_lockedtodata | O | When asserted, this signal indicates that the CDR PLL is locked to the incoming data rx_serial_data. |
| rx_is_lockedtoref | O | When asserted, this signal indicates that the CDR PLL is locked to the incoming reference clock, rx_cdr_refclk. |

**Related Information**

Intel Arria 10 Transceiver PHY User Guide
More information about Gigabit Ethernet (GbE) and GbE with 1588, the connection guidelines for a PHY design, and how to implement GbE/GbE with 1588 in Intel Arria 10 Transceivers

### 6.1.3.5. ECC Status Signals

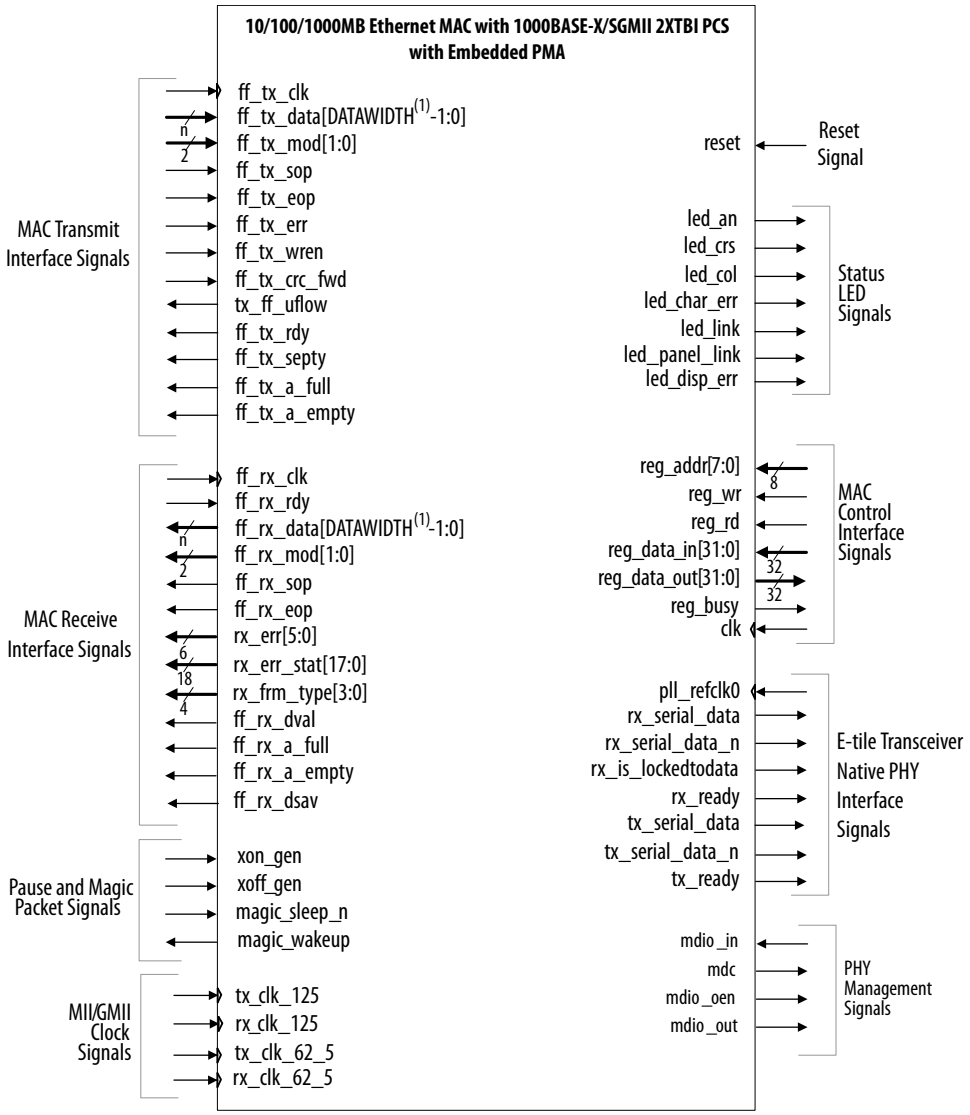
**Table 76. ECC Status Signals**

| Name | I/O | Description |
|---|---|---|
| `pcs_eccstatus[1:0]` | O | Indicates the ECC status. This signal is synchronized to the `reg_clk` clock domain.<br>11: An uncorrectable error occurred and the error data appears at the output.<br>10: A correctable error occurred and the error has been corrected at the output. However, the memory array has not been updated.<br>01: Not valid.<br>00: No error. |

For more information on the signals, refer to the respective sections shown in Table 71 on page 117.

## 6.1.4. 10/100/1000 Ethernet MAC with 1000BASE-X/SGMII 2XTBI PCS Signals

**Figure 49. 10/100/1000 Ethernet MAC Function with Internal FIFO Buffers, and 1000BASE-X/SGMII 2XTBI PCS Signals With Embedded PMA Signals**




Note to 10/100/1000 Ethernet MAC Function with Internal FIFO Buffers, and 1000BASE-X/SGMII 2XTBI PCS Signals With Embedded PMA Signals:

1. The DATAWIDTH value depends on the FIFO width that you select in the parameter editor. Options available are 8 and 32 bits.

### Table 77. References

| Interface Signal | Section |
|---|---|
| Reset signal | Table 54 on page 108 |
| MAC control interface signals | MAC Control Interface Signals on page 108 |
| Pause and magic packet signals | Pause and Magic Packet Signals on page 112 |
| Status LED signals | Status LED Control Signals on page 119 |
| MAC transmit interface | MAC Transmit Interface Signals on page 111 |
| MAC receive interface | MAC Receive Interface Signals on page 109 |
| PHY management signals | PHY Management Signals on page 114 |

## 6.1.4.1. GMII Clock Signals

### Table 78. GMII Clocks

| Name | I/O | Description |
|---|---|---|
| `rx_clk_125` | I | 125 MHz receive clock for the RX datapath on MAC side |
| `tx_clk_125` | I | 125 MHz transmit clock for the TX datapath on MAC side. |
| `rx_clk_62_5` | I | 62.5 MHz receive clock for the RX datapath on PCS side.<br>Intel recommends that this clock and `rx_clk_125` share the same clock source. This clock must be synchronous to `rx_clk_125`. Their rising edges must align and must have 0 ppm and phase shift. |
| `tx_clk_62_5` | I | 62.5 MHz transmit clock for the TX datapath on PCS side.<br>Intel recommends that this clock and `tx_clk_125` share the same clock source. This clock must be synchronous to `tx_clk_125`. Their rising edges must align and must have 0 ppm and phase shift. |

For more information about the clock signals, refer to Clocking Scheme of MAC with 2XTBI PCS and Embedded PMA on page 159.

## 6.1.4.2. E-tile Transceiver Native PHY Signals

### Table 79. E-tile Transceiver Native PHY Signals

| Name | I/O | Description |
|---|---|---|
| `pll_refclk0` | I | Reference clock for the E-tile Native PHY transceiver. Set this clock to 156.25 MHz. |
| `rx_serial_data` | I | Positive signal for the receiver serial data. |
| `rx_serial_data_n` | I | Negative signal for the receiver serial data. |
| `rx_is_lockedtodata` | O | When asserted, this signal indicates that the CDR PLL is locked to the incoming `rx_serial` data. |
| `tx_serial_data` | O | Positive signal for the transmitter serial data. |
| `tx_serial_data_n` | O | Negative signal for the transmitter serial data. |
| `rx_ready` | O | Status signal from E-tile Native PHY. It is asserted when Native PHY RX datapath resets sequencing is complete. |
| `tx_ready` | O | Status signal from E-tile Native PHY. It is asserted when Native PHY TX datapath resets sequencing is complete. |

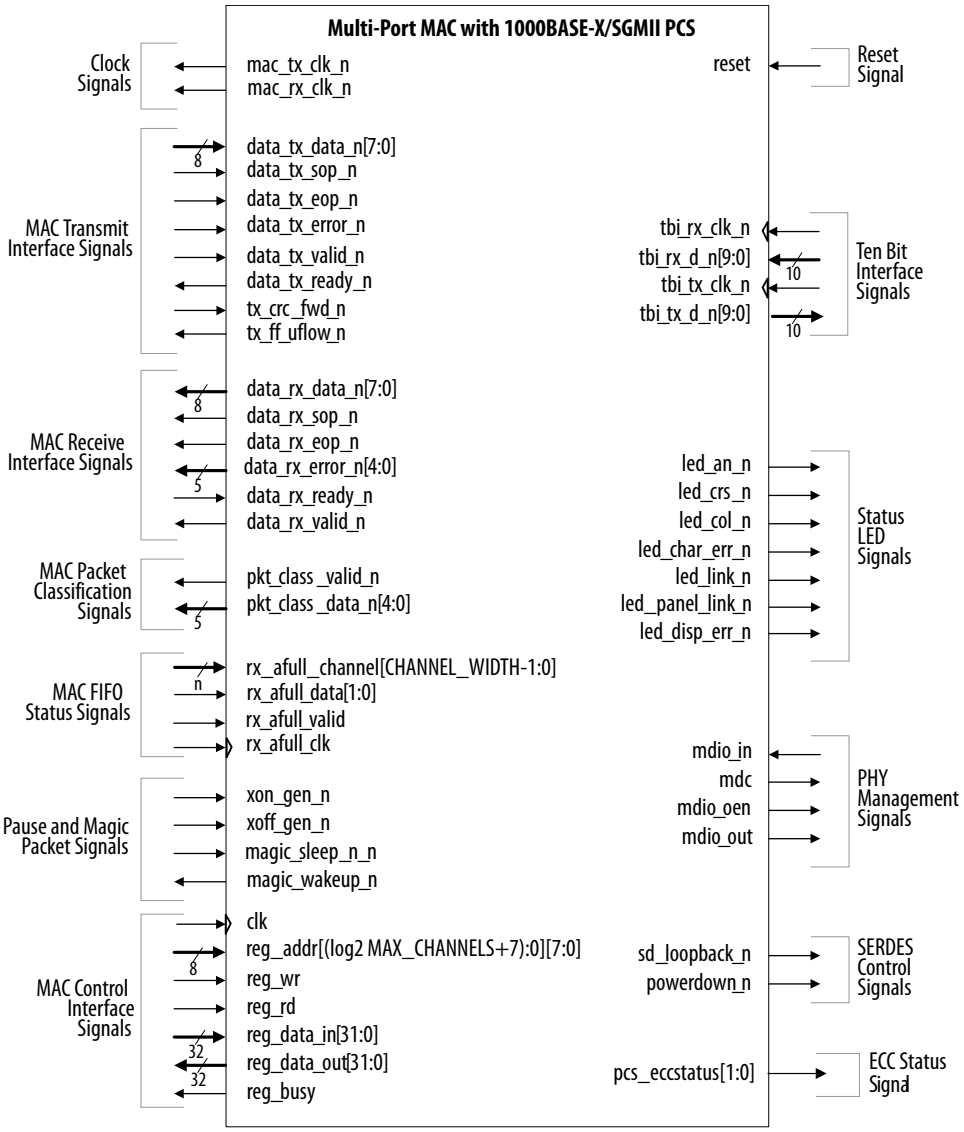
*Note:* For Intel Stratix 10 E-tile and Intel Agilex devices, the `reconfig_avmm` interface signals are present when the reconfiguration feature is enabled. Refer to the *E-Tile Transceiver PHY User Guide* for more information on the interface signals.

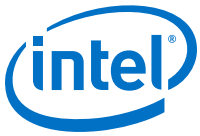
**Related Information**

E-Tile Transceiver PHY User Guide

## 6.1.5. 10/100/1000 Multiport Ethernet MAC with 1000BASE-X/SGMII PCS Signals

**Figure 50. 10/100/1000 Multiport Ethernet MAC Function without Internal FIFO Buffers with 1000BASE-X/SGMII PCS Signals**



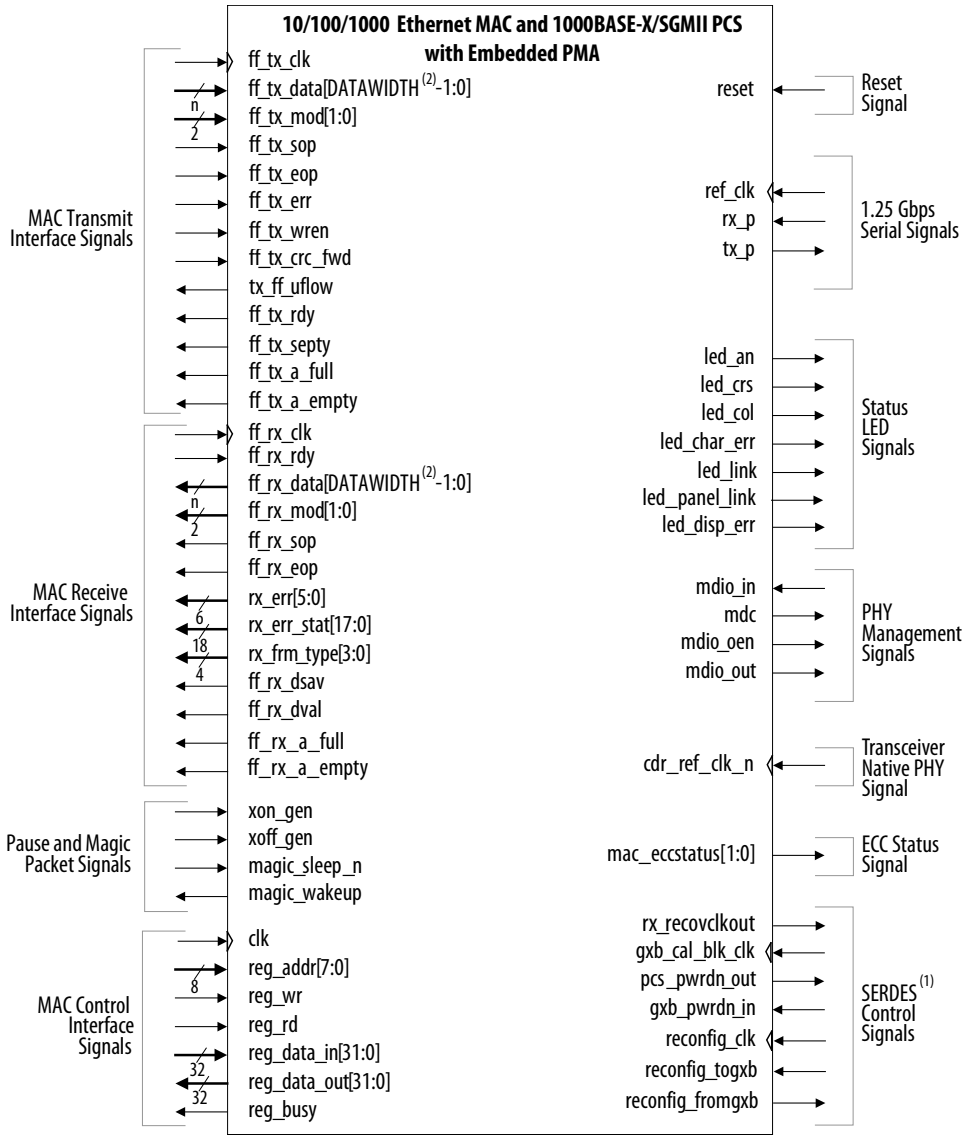


### Table 80. References

| Interface Signal | Section |
|---|---|
| Clock and reset signals | Clock and Reset Signals on page 107 |
| MAC control interface | MAC Control Interface Signals on page 108 |
| MAC transmit interface | MAC Transmit Interface Signals on page 111 |
| MAC receive interface | MAC Receive Interface Signals on page 109 |
| MAC packet classification signals | Multiport MAC Packet Classification Signals on page 117 |
| MAC FIFO status signals | Multiport MAC FIFO Status Signals on page 117 |
| Pause and magic packet signals | Pause and Magic Packet Signals on page 112 |
| PHY management signals | PHY Management Signals on page 114 |
| Ten-bit interface | TBI Interface Signals on page 119 |
| Status LED signals | Status LED Control Signals on page 119 |
| SERDES control signals | SERDES Control Signals on page 120 |

## 6.1.6. 10/100/1000 Ethernet MAC with 1000BASE-X/SGMII PCS and Embedded PMA Signals

**Figure 51. 10/100/1000 Ethernet MAC Function with Internal FIFO Buffers, and 1000BASE-X/SGMII PCS With Embedded PMA Signals**




Note to Figure 51 on page 126:

1. The SERDES control signals are present in variations targeting devices with GX transceivers. For device families prior to the Stratix V device, the reconfiguration signals—`reconfig_clk`, `reconfig_togxb`, and `reconfig_fromgxb`—are always present in the interface. For Stratix V GX device, the reconfiguration signals—`reconfig_togxb` and `reconfig_fromgxb`—are always present in the interface. The `reconfig_clk` reconfiguration signal is embedded in the

reconfig_togxb reconfiguration signal. For Intel Arria 10 GX device, the reconfig_avmm interface signal is present when reconfiguration feature is enabled.

2. The DATAWIDTH value depends on the FIFO width that you select in the parameter editor. Options available are 8 and 32 bits.

#### 6.1.6.1. 1.25 Gbps Serial Interface

If the variant includes an embedded PMA, the PMA provides a 1.25-GHz serial interface.

**Table 81. 1.25 Gbps MDI Interface Signals**

| Name | I/O | Description |
|---|---|---|
| ref_clk | I | 125 MHz local reference clock oscillator. |
| rx_p | I | Serial Differential Receive Interface. |
| tx_p | O | Serial Differential Transmit Interface. |

### 6.1.6.2. Transceiver Native PHY Signal

**Table 82. Transceiver Native PHY Signal**

| Name | I/O | Description |
|---|---|---|
| cdr_ref_clk_n | I | Port to connect the RX PLL reference clock with a frequency of 125 MHz when you enable SyncE support. |

### 6.1.6.3. SERDES Control Signals

These signals apply only to PMA blocks implemented in devices with GX transceivers.

**Table 83. SERDES Control Signal**

| Name | I/O | Description |
|---|---|---|
| rx_recovclkout | O | Recovered clock from the PMA block. |
| pcs_pwrdn_out | O | Power-down status. Asserted when the PCS function is in power-down mode; deasserted when the PCS function is operating in normal mode. This signal is implemented only when an internal SERDES is used with the option to export the power-down signal.<br>This signal is not present in PMA blocks implemented in Intel Arria 10, Stratix V, Arria V, and Arria V devices with GX transceivers. |
| gxb_pwrdn_in | I | Power-down enable. Assert this signal to power down the transceiver quad block. This signal is implemented only when an internal SERDES is used with the option to export the power-down signal.<br>This signal is not present in PMA blocks implemented in Intel Arria 10, Stratix V, Arria V, and Arria V devices with GX transceivers. |
| gxb_cal_blk_clk | I | 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.<br>This signal is not present in PMA blocks implemented in Intel Arria 10, Stratix V, Arria V, and Cyclone V devices with GX transceivers. |

*continued...*

| Name | I/O | Description |
|---|---|---|
| reconfig_clk | I | Reference clock for the dynamic reconfiguration controller. If you use a dynamic reconfiguration controller in your design to dynamically control the transceiver, both the reconfiguration controller and the IP core require this clock. This clock must operate between 37.5–50 MHz. Tie this clock low if you are not using an external reconfiguration controller.<br>This signal is not present in PMA blocks implemented in Intel Arria 10, Stratix V, Arria V, and Cyclone V devices with GX transceivers. |
| reconfig_togxb[n:0] | I | Driven from an external dynamic reconfiguration controller. Supports the selection of multiple transceiver channels for dynamic reconfiguration.<br>For PMA blocks implemented in Stratix V devices with GX transceivers, the bus width is [139:0]. For more information about the bus width for PMA blocks implemented in each device, refer to the Dynamic Reconfiguration chapter of the respective device handbook. |
| reconfig_fromgxb[n:0] | O | Connects 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 you are not using an external reconfiguration controller.<br>For more information about the bus width for PMA blocks implemented in each device, refer to the Dynamic Reconfiguration chapter of the respective device handbook. |
| reconfig_busy | I | Driven from an external dynamic reconfiguration controller. This signal will indicate the busy status of the dynamic reconfiguration controller during offset cancellation. Tie this signal to 1'b0 if you are not using an external reconfiguration controller.<br>This signal is not present in PMA blocks implemented in Intel Arria 10, Stratix V, Arria V, and Arria V devices with GX transceivers. |

For more information on the signals, refer to the respective sections shown in Table 80 on page 125.

### 6.1.6.4. Intel LVDS Transmitter and Receiver Soft-CDR I/O Signals

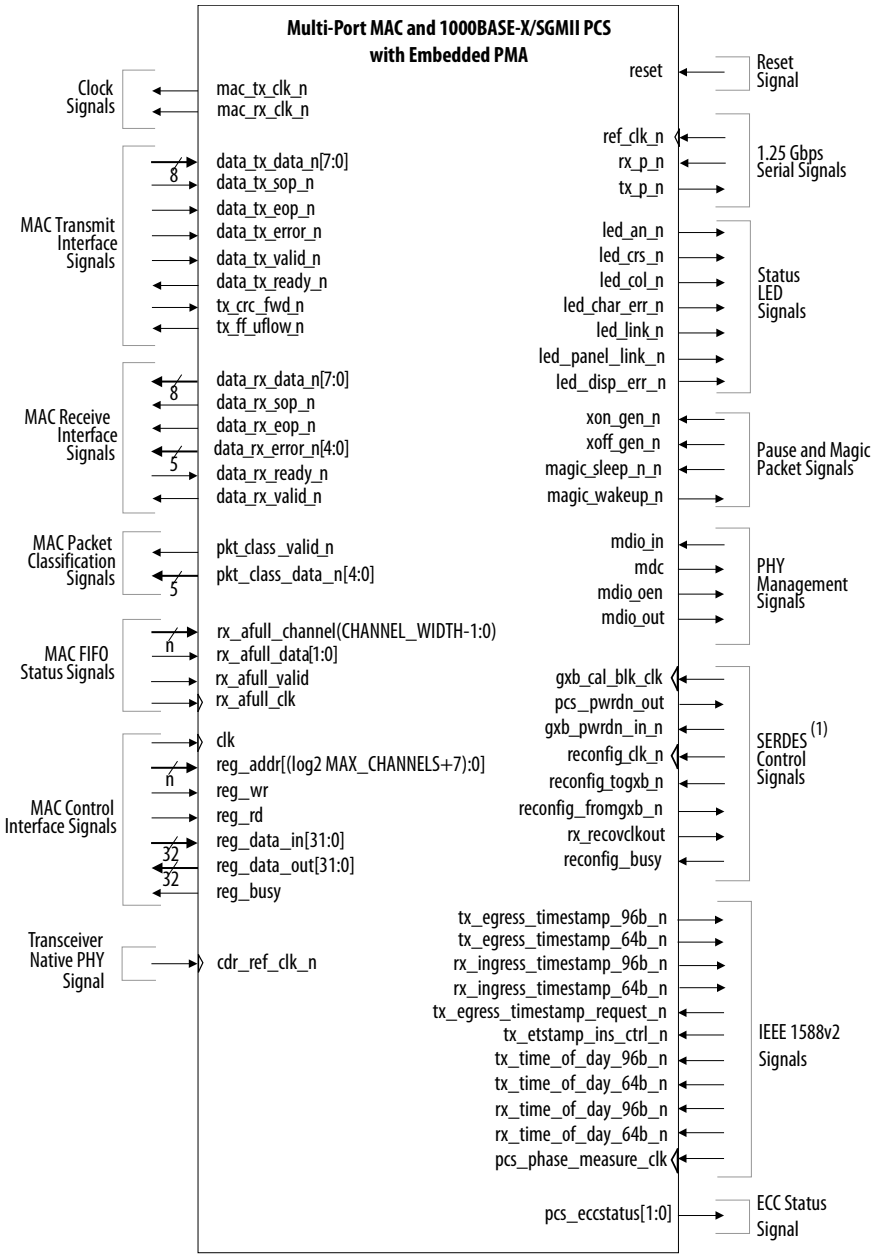
These signals apply only to Intel Agilex devices.

**Table 84. SERDES Control Signal**

| Name | I/O | Description |
|---|---|---|
| refclk | I | 125 MHz local reference clock oscillator. |
| txp | O | Positive signal for the transmitter serial data. |
| txn | O | Negative signal for the transmitter serial data. |
| rxp | I | Positive signal for the receiver serial data. |
| rxn | I | Negative signal for the receiver serial data. |
| rx_recovclkout | O | Recovered clock from LVDS receiver core. |
| lvds_tx_pll_locked | O | PLL locked indicator from LVDS transmitter core. |

## 6.1.7. 10/100/1000 Multiport Ethernet MAC with 1000BASE-X/SGMII PCS and Embedded PMA

**Figure 52. 10/100/1000 Multiport Ethernet MAC Function without Internal FIFO Buffers, with IEEE 1588v2, 1000BASE-X/SGMII PCS and Embedded PMA Signals**




Note to Figure 52 on page 129:

1. The SERDES control signals are present in variations targeting devices with GX transceivers. For Stratix II GX and Arria GX devices, the reconfiguration signals—`reconfig_clk`, `reconfig_togxb`, and `reconfig_fromgxb`—are included only when the **Enable transceiver dynamic reconfiguration** option is turned on.

The reconfiguration signals—`gxb_cal_blk_clk`, `pcs_pwrdwn_out`, `gxb_pwrdn_in`, `reconfig_clk`, and `reconfig_busy`—are not present in variations targeting Intel Arria 10, Stratix V, Arria V, and Cyclone V devices with GX transceivers.

**Table 85. References**

| Interface Signal | Section |
|---|---|
| Clock and reset signals | Clock and Reset Signals on page 107 |
| MAC control interface | MAC Control Interface Signals on page 108 |
| MAC transmit interface | MAC Transmit Interface Signals on page 111 |
| MAC receive interface | MAC Receive Interface Signals on page 109 |
| MAC packet classification signals | Multiport MAC Packet Classification Signals on page 117 |
| MAC FIFO status signals | Multiport MAC FIFO Status Signals on page 117 |
| Pause and magic packet signals | Pause and Magic Packet Signals on page 112 |
| PHY management signals | PHY Management Signals on page 114 |
| 1.25 Gbps Serial Signals | 1.25 Gbps Serial Interface on page 127 |
| Status LED signals | Status LED Control Signals on page 119 |
| SERDES control signals | SERDES Control Signals on page 120 |
| Transceiver Native PHY signal | Transceiver Native PHY Signal on page 127 |
| IEEE 1588v2 RX Timestamp Signals | IEEE 1588v2 RX Timestamp Signals on page 130 |
| IEEE 1588v2 TX Timestamp Signals | IEEE 1588v2 TX Timestamp Signals on page 131 |
| IEEE 1588v2 TX Timestamp Request Signals | IEEE 1588v2 TX Timestamp Request Signals on page 131 |
| IEEE 1588v2 TX Insert Control Timestamp Signals | IEEE 1588v2 TX Insert Control Timestamp Signals on page 132 |
| IEEE 1588v2 ToD Clock Interface Signals | IEEE 1588v2 Time-of-Day (ToD) Clock Interface Signals on page 133 |

### 6.1.7.1. IEEE 1588v2 RX Timestamp Signals

**Table 86. IEEE 1588v2 RX Timestamp Interface Signals**

| Signal | I/O | Width | Description |
|---|---|---|---|
| `rx_ingress_timestamp_96b_data_n` | O | 96 | Carries the ingress timestamp on the receive datapath. Consists of 48-bit seconds field, 32-bit nanoseconds field, and 16-bit fractional nanoseconds field.<br>The MAC presents the timestamp for all receive frames and asserts this signal in the same clock cycle it asserts `rx_ingress_timestamp_96b_valid`. |
| `rx_ingress_timestamp_96b_valid` | O | 1 | When asserted, this signal indicates that `rx_ingress_timestamp_96b_data` contains valid timestamp.<br>For all receive frame, the MAC asserts this signal in the same clock cycle it receives the start of packet (`avalon_st_rx_startofpacket` is asserted). |
| `rx_ingress_timestamp_64b_data` | O | 64 | Carries the ingress timestamp on the receive datapath. Consists of 48-bit nanoseconds field and 16-bit fractional nanoseconds field. |

*continued...*

| Signal | I/O | Width | Description |
|---|---|---|---|
| | | | The MAC presents the timestamp for all receive frames and asserts this signal in the same clock cycle it asserts `rx_ingress_timestamp_64b_valid`. |
| `rx_ingress_timestamp_64b_valid` | O | 1 | When asserted, this signal indicates that `rx_ingress_timestamp_64b_data` contains valid timestamp.<br>For all receive frame, the MAC asserts this signal in the same clock cycle it receives the start of packet (`avalon_st_rx_startofpacket` is asserted). |

### 6.1.7.2. IEEE 1588v2 TX Timestamp Signals

**Table 87. IEEE 1588v2 TX Timestamp Interface Signals**

| Signal | I/O | Width | Description |
|---|---|---|---|
| `tx_egress_timestamp_96b_data_n` | O | 96 | A transmit interface signal. This signal requests timestamp of frames on the TX path. The timestamp is used to calculate the residence time.<br>Consists of 48-bit seconds field, 32-bit nanoseconds field, and 16-bit fractional nanoseconds field. |
| `tx_egress_timestamp_96b_valid` | O | 1 | A transmit interface signal. Assert this signal to indicate that a timestamp is obtained and a timestamp request is valid for the particular frame.<br>Assert this signal in the same clock cycle as the start of packet (`avalon_st_tx_startofpacket` is asserted). |
| `tx_egress_timestamp_96b_fingerprint` | O | *n* | Configurable width fingerprint that returns with correlated timestamps.<br>The signal width is determined by the TSTAMP_FP_WIDTH parameter (default parameter value is 4). |
| `tx_egress_timestamp_64b_data` | O | 64 | A transmit interface signal. This signal requests timestamp of frames on the TX path. The timestamp is used to calculate the residence time.<br>Consists of 48-bit nanoseconds field and 16-bit fractional nanoseconds field. |
| `tx_egress_timestamp_64b_valid` | O | 1 | A transmit interface signal. Assert this signal to indicate that a timestamp is obtained and a timestamp request is valid for the particular frame.<br>Assert this signal in the same clock cycle as the start of packet (`avalon_st_tx_startofpacket` or `avalon_st_tx_startofpacket_n` is asserted). |
| `tx_egress_timestamp_64b_fingerprint` | O | *n* | Configurable width fingerprint that returns with correlated timestamps.<br>The signal width is determined by the TSTAMP_FP_WIDTH parameter (default parameter value is 4). |

### 6.1.7.3. IEEE 1588v2 TX Timestamp Request Signals

**Table 88. IEEE 1588v2 TX Timestamp Request Signals**

| Signal | I/O | Width | Description |
|---|---|---|---|
| `tx_egress_timestamp_request_valid_n` | I | 1 | Assert this signal when a user-defined `tx_egress_timestamp` is required for a transmit frame.<br>Assert this signal in the same clock cycle as the start of packet (`avalon_st_tx_startofpacket` or `avalon_st_tx_startofpacket_n` is asserted). |
| `tx_egress_timestamp_request_fingerprint` | I | *n* | Use this bus to specify fingerprint for the user-defined `tx_egress_timestamp`. The fingerprint is used to identify the user-defined timestamp.<br>The signal width is determined by the TSTAMP_FP_WIDTH parameter (default parameter value is 4).<br>The value of this signal is mapped to `user_fingerprint`.<br>This signal is only valid when you assert `tx_egress_timestamp_request_valid`. |

### 6.1.7.4. IEEE 1588v2 TX Insert Control Timestamp Signals

**Table 89. IEEE 1588v2 TX Insert Control Timestamp Interface Signals**

| Signal | I/O | Width | Description |
|---|---|---|---|
| tx_etstamp_ins_ctrl_timestamp_insert_n | I | 1 | Assert this signal to insert egress timestamp into the associated frame.<br>Assert this signal in the same clock cycle as the start of packet (avalon_st_tx_startofpacket is asserted). |
| tx_etstamp_ins_ctrl_timestamp_format | I | 1 | Timestamp format of the frame, which the timestamp inserts.<br>0: 1588v2 format (48-bits second field + 32-bits nanosecond field + 16-bits correction field for fractional nanosecond)<br>Required offset location of timestamp and correction field.<br>1: 1588v1 format (32-bits second field + 32-bits nanosecond field)<br>Required offset location of timestamp.<br>Assert this signal in the same clock cycle as the start of packet (avalon_st_tx_startofpacket is asserted). |
| tx_etstamp_ins_ctrl_residence_time_update | I | 1 | Assert this signal to add residence time (egress timestamp –ingress timestamp) into correction field of PTP frame.<br>Required offset location of correction field.<br>Assert this signal in the same clock cycle as the start of packet (avalon_st_tx_startofpacket is asserted). |
| tx_etstamp_ins_ctrl_ingress_timestamp_96b[] | I | 96 | 96-bit format of ingress timestamp.<br>(48 bits second + 32 bits nanosecond + 16 bits fractional nanosecond).<br>Assert this signal in the same clock cycle as the start of packet (avalon_st_tx_startofpacket is asserted). |
| tx_etstamp_ins_ctrl_ingress_timestamp_64b[] | I | 64 | 64-bit format of ingress timestamp.<br>(48-bits nanosecond + 16-bits fractional nanosecond).<br>Assert this signal in the same clock cycle as the start of packet (avalon_st_tx_startofpacket is asserted). |
| tx_etstamp_ins_ctrl_residence_time_calc_format | I | 1 | Format of timestamp to be used for residence time calculation.<br>0: 96-bits (96-bits egress timestamp - 96-bits ingress timestamp).<br>1: 64-bits (64-bits egress timestamp - 64-bits ingress timestamp).<br>Assert this signal in the same clock cycle as the start of packet (avalon_st_tx_startofpacket is asserted). |
| tx_etstamp_ins_ctrl_checksum_zero | I | 1 | Assert this signal to set the checksum field of UDP/IPv4 to zero.<br>Required offset location of checksum field.<br>Assert this signal in the same clock cycle as the start of packet (avalon_st_tx_startofpacket is asserted). |
| tx_etstamp_ins_ctrl_checksum_correct | I | 1 | Assert this signal to correct UDP/IPv6 packet checksum, by updating the checksum correction, which is specified by checksum correction offset.<br>Required offset location of checksum correction.<br>Assert this signal in the same clock cycle as the start of packet (avalon_st_tx_startofpacket is asserted). |
| tx_etstamp_ins_ctrl_offset_timestamp | I | 1 | The location of the timestamp field, relative to the first byte of the packet.<br>Assert this signal in the same clock cycle as the start of packet (avalon_st_tx_startofpacket is asserted). |
| tx_etstamp_ins_ctrl_offset_correction_field[] | I | 16 | The location of the correction field, relative to the first byte of the packet.<br>Assert this signal in the same clock cycle as the start of packet (avalon_st_tx_startofpacket is asserted). |
| tx_etstamp_ins_ctrl_offset_checksum_field[] | I | 16 | The location of the checksum field, relative to the first byte of the packet. |

*continued...*

| Signal | I/O | Width | Description |
|---|---|---|---|
| | | | Assert this signal in the same clock cycle as the start of packet (`avalon_st_tx_startofpacket` is asserted). |
| `tx_etstamp_ins_ctrl_offset_checksum_correction[]` | I | 16 | The location of the checksum correction field, relative to the first byte of the packet.<br>Assert this signal in the same clock cycle as the start of packet (`avalon_st_tx_startofpacket` is asserted). |

### 6.1.7.5. IEEE 1588v2 Time-of-Day (ToD) Clock Interface Signals

**Table 90. IEEE 1588v2 ToD Clock Interface Signals**

| Signal | I/O | Width | Description |
|---|---|---|---|
| `tx_time_of_day_96b_data_n` | I | 96 | Use this bus to carry the time-of-day from external ToD module to 96-bit MAC TX clock.<br>Consists of 48 bits seconds field, 32 bits nanoseconds field, and 16 bits fractional nanoseconds field |
| `rx_time_of_day_96b_data` | I | 96 | Use this bus to carry the time-of-day from external ToD module to 96-bit MAC RX clock.<br>Consists of 48 bits seconds field, 32 bits nanoseconds field, and 16 bits fractional nanoseconds field |
| `tx_time_of_day_64b_data` | I | 64 | Use this bus to carry the time-of-day from external ToD module to 64-bit MAC TX clock.<br>Consists of 48-bit nanoseconds field and 16-bit fractional nanoseconds field |
| `rx_time_of_day_64b_data` | I | 64 | Use this bus to carry the time-of-day from external ToD module to 64-bit MAC RX clock.<br>Consists of 48-bit nanoseconds field and 16-bit fractional nanoseconds field |

### 6.1.7.6. IEEE 1588v2 PCS Phase Measurement Clock Signal

**Table 91. IEEE 1588v2 PCS Phase Measurement Clock Signal**

| Signal | I/O | Width | Description |
|---|---|---|---|
| `pcs_phase_measure_clk` | I | 1 | Sampling clock to measure the latency through the PCS FIFO buffer. The recommended frequency is 80 MHz. |

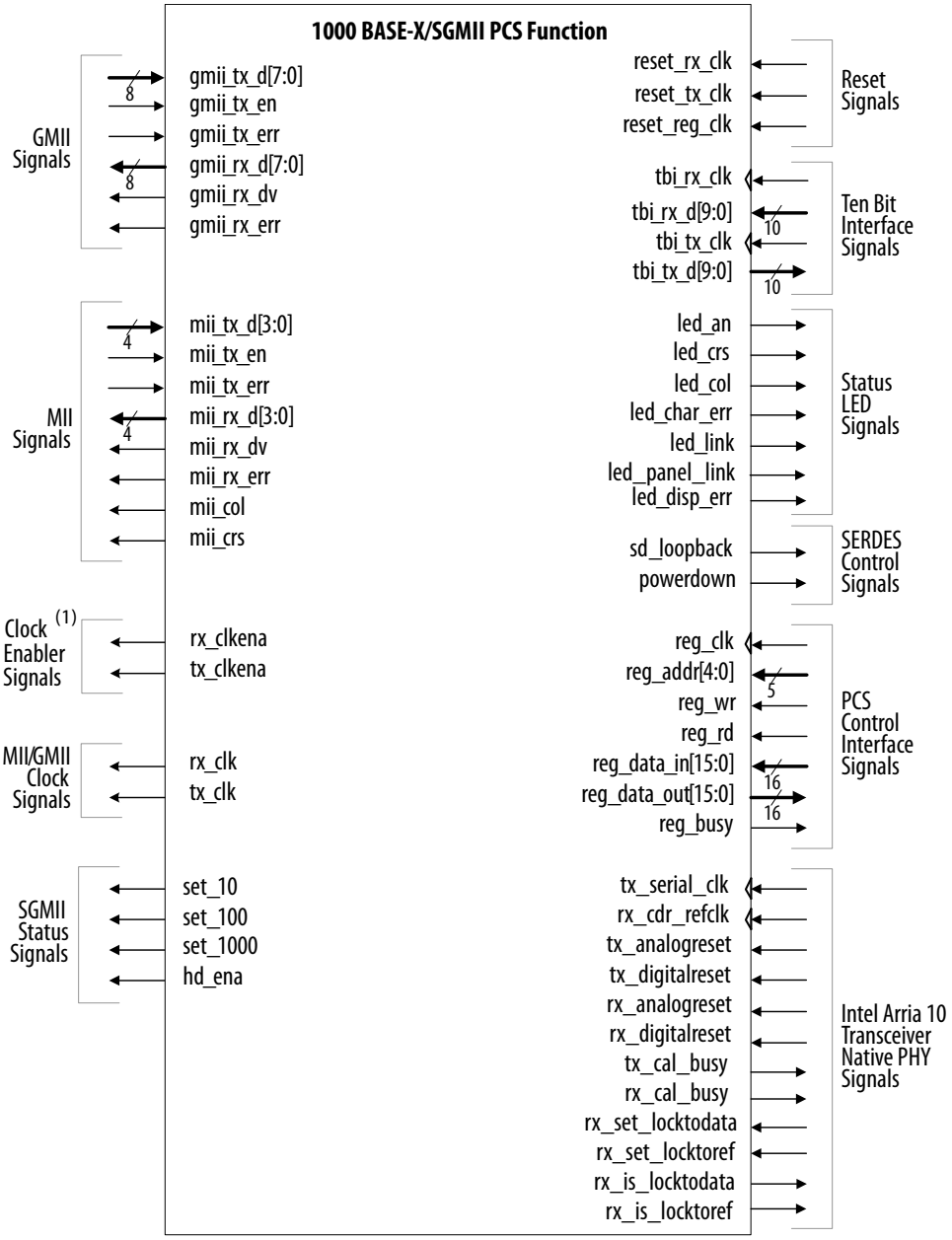
### 6.1.7.7. IEEE 1588v2 PHY Path Delay Interface Signals

**Table 92. IEEE 1588v2 PHY Path Delay Interface Signals**

| Signal | I/O | Width | Description |
|---|---|---|---|
| `tx_path_delay_data` | I | 22 | Use this bus to carry the path delay on the transmit datapath. The delay is measured between the physical network and MII/GMII to adjust the egress timestamp.<br>Bits 0 to 9—Fractional number of clock cycles<br>Bits 10 to 21—Number of clock cycles |
| `rx_path_delay_data` | I | 22 | Use this bus to carry the path delay on the receive datapath. The delay is measured between the physical network and MII/GMII to adjust the ingress timestamp.<br>Bits 0 to 9—Fractional number of clock cycles<br>Bits 10 to 21—Number of clock cycles |

## 6.1.8. 1000BASE-X/SGMII PCS Signals

**Figure 53. 1000BASE-X/SGMII PCS Function Signals**




Note to Figure 53 on page 134:

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

### Table 93. References

| Interface Signal | Section |
|---|---|
| Ten-bit interface | TBI Interface Signals on page 119 |
| Status LED signals | Status LED Control Signals on page 119 |
| SERDES control signals | SERDES Control Signals on page 120 |
| Intel Arria 10 Transceiver Native PHY signals | Intel Arria 10 Transceiver Native PHY Signals on page 120 |

## 6.1.8.1. PCS Control Interface Signals

### Table 94. Register Interface Signals

| Name | Avalon Memory-Mapped Signal Type | I/O | Description |
|---|---|---|---|
| reg_clk | clk | I | Register access reference clock. Set the signal to a value less than or equal to 125-MHz. |
| reset_reg_clk | reset | I | Active-high reset signal for reg_clk clock domain. |
| reg_wr | write | I | Register write enable. |
| reg_rd | read | I | Register read enable. |
| reg_addr[4:0] | address | I | 16-bit word-aligned register address. |
| reg_data_in[15:0] | writedata | I | Register write data. Bit 0 is the least significant bit. |
| reg_data_out[15:0] | readdata | O | Register read data. Bit 0 is the least significant bit. |
| reg_busy | waitrequest | O | Register interface busy. Asserted during register read or register write. A value of 0 indicates that the read or write is complete. |

## 6.1.8.2. PCS Reset Signals

### Table 95. Reset Signals

| Name | I/O | Description |
|---|---|---|
| reset_rx_clk | I | Active-high reset signal for PCS rx_clk clock domain. Assert this signal to reset the logic synchronized by rx_clk. |
| reset_tx_clk | I | Active-high reset signal for PCS tx_clk clock domain. Assert this signal to reset the logic synchronized by tx_clk. |

## 6.1.8.3. MII/GMII Clocks and Clock Enablers

Data transfers on the MII/GMII interface are synchronous to the receive and transmit clocks.

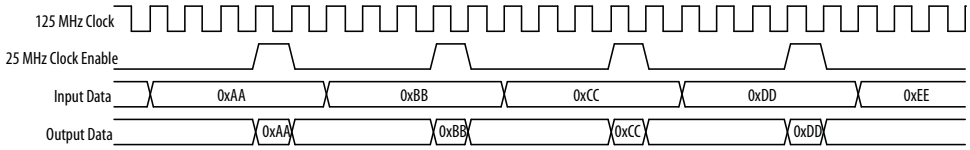
### Table 96. MAC Clock Signals

| Name | I/O | Description |
|---|---|---|
| rx_clk | O | Receive clock. This clock is derived from the TBI clock tbi_rx_clk and set to 125 MHz. |
| tx_clk | O | Transmit clock. This clock is derived from the TBI clock tbi_tx_clk and set to 125 MHz. |

*continued...*

| Name | I/O | Description |
|---|---|---|
| **Clock Enabler Signals**<br>*Note:* The clock enabler signals are present only in SGMII mode. | | |
| rx_clkena | O | Receive clock enabler. In SGMII mode, this signal enables rx_clk. |
| tx_clkena | O | Transmit clock enabler. In SGMII mode, this signal enables tx_clk. |

**Figure 54. Clock Enabler Signal Behavior**




### 6.1.8.4. GMII

**Table 97. GMII Signals**

| Name | I/O | Description |
|---|---|---|
| **GMII Transmit Interface** | | |
| gmii_tx_d[7:0] | I | GMII transmit data bus. |
| gmii_tx_en | I | Assert this signal to indicate that the data on gmii_tx_d[7:0]is valid. |
| gmii_tx_err | I | Assert this signal to indicate to the PHY device that the current frame sent is invalid. |
| **GMII Receive Interface** | | |
| gmii_rx_d[7:0] | O | GMII receive data bus. |
| gmii_rx_dv | O | Asserted to indicate that the data on gmii_rx_d[7:0] 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 | O | Asserted by the PHY to indicate that the current frame contains errors. |

### 6.1.8.5. MII

**Table 98. MII Signals**

| Name | I/O | Description |
|---|---|---|
| **MII Transmit Interface** | | |
| mii_tx_d[3:0] | I | MII transmit data bus. |
| mii_tx_en | I | Assert this signal to indicate that the data on mii_tx_d[3:0]is valid. |
| mii_tx_err | I | Assert this signal to indicate to the PHY device that the frame sent is invalid. |
| **MII Receive Interface** | | |
| mii_rx_d[3:0] | O | MII receive data bus. |
| mii_rx_dv | O | Asserted to indicate that the data on mii_rx_d[3:0]is valid. The signal stays asserted during frame reception, from the first preamble byte until the last byte of the CRC field is received. |

*continued...*

| Name | I/O | Description |
|---|---|---|
| mii_rx_err | O | Asserted by the PHY to indicate that the current frame contains errors. |
| mii_col | O | Collision detection. Asserted by the PCS function to indicate that a collision was detected during frame transmission. |
| mii_crs | O | Carrier sense detection. Asserted by the PCS function to indicate that a transmit or receive activity is detected on the Ethernet line. |

### 6.1.8.6. SGMII Status Signals

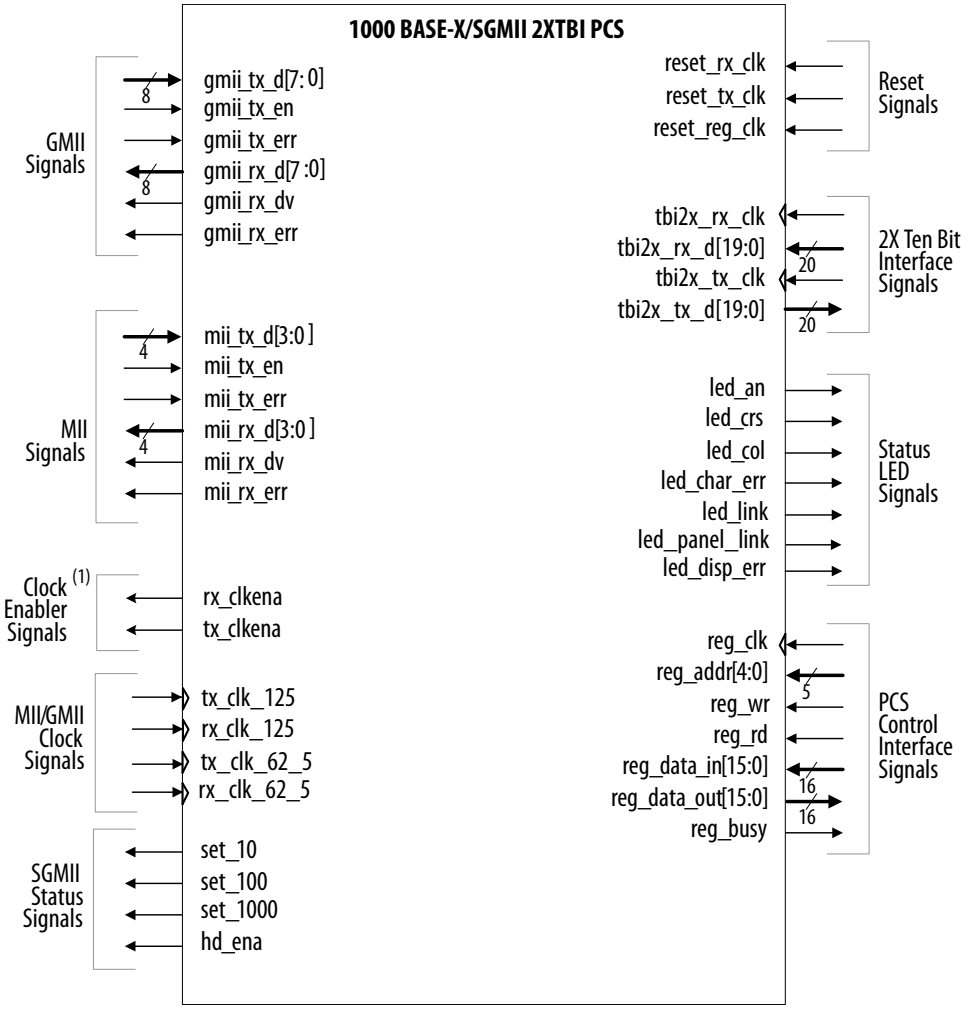
The SGMII status signals provide status information to the PCS block. When the PCS is instantiated standalone, these signals are inputs to the MAC and serve as interface control signals for that block.

**Table 99. SGMII Status Signals**

| Name | I/O | Description |
|---|---|---|
| set_1000 | O | 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:<br>• 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.<br>• The USE_SGMII_AN bit is set to 0 and the SGMII_SPEED bit is set to 10. |
| set_100 | O | 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:<br>• 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.<br>• The USE_SGMII_AN bit is set to 0 and the SGMII_SPEED bit is set to 01. |
| set_10 | O | 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:<br>• 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.<br>• The USE_SGMII_AN bit is set to 0 and the SGMII_SPEED bit is set to 00. |
| hd_ena | O | 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:<br>• 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.<br>• The USE_SGMII_AN bit is set to 0 and the SGMII_DUPLEX bit is set to 1. |

## 6.1.9. 1000BASE-X/SGMII 2XTBI PCS Signals

**Figure 55. 1000BASE-X/SGMII 2XTBI PCS Function Signals**




Note to Figure 55 on page 138:

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

**Table 100. References**

| Interface Signal | Section |
|---|---|
| GMII Clock Signals | GMII Clock Signals on page 123 |
| GMII signals | GMII on page 136 |
| MII signals | MII on page 136 |
| PCS control interface signals | PCS Control Interface Signals on page 135 |
| SGMII status signals | SGMII Status Signals on page 137 |
| Status LED signals | Status LED Control Signals on page 119 |

### 6.1.9.1. Clock Enablers

#### Table 101. Clock Enablers

*Note:* The clock enabler signals are present only in SGMII mode.

| Name | I/O | Description |
|---|---|---|
| rx_clkena | O | Receive clock enabler for SGMII 10M/100M operating speeds. |
| tx_clkena | O | Transmit clock enabler for SGMII 10M/100M operating speeds. |

### 6.1.9.2. PCS Reset Signals

#### Table 102. Reset Signals

| Name | I/O | Description |
|---|---|---|
| reset_rx_clk | I | Active-high reset signal for PCS receive clock domain. Assert this signal to reset the logic synchronized by rx_clk_125 and rx_clk_62_5. |
| reset_tx_clk | I | Active-high reset signal for PCS transmit clock domain. Assert this signal to reset the logic synchronized by tx_clk_125 and tx_clk_62_5. |

### 6.1.9.3. TBI Interface Signals

If the core variation does not include an embedded PMA, the PCS block provides a 62.5-MHz 2 ten-bit interface (TBI) to an external SERDES chip.

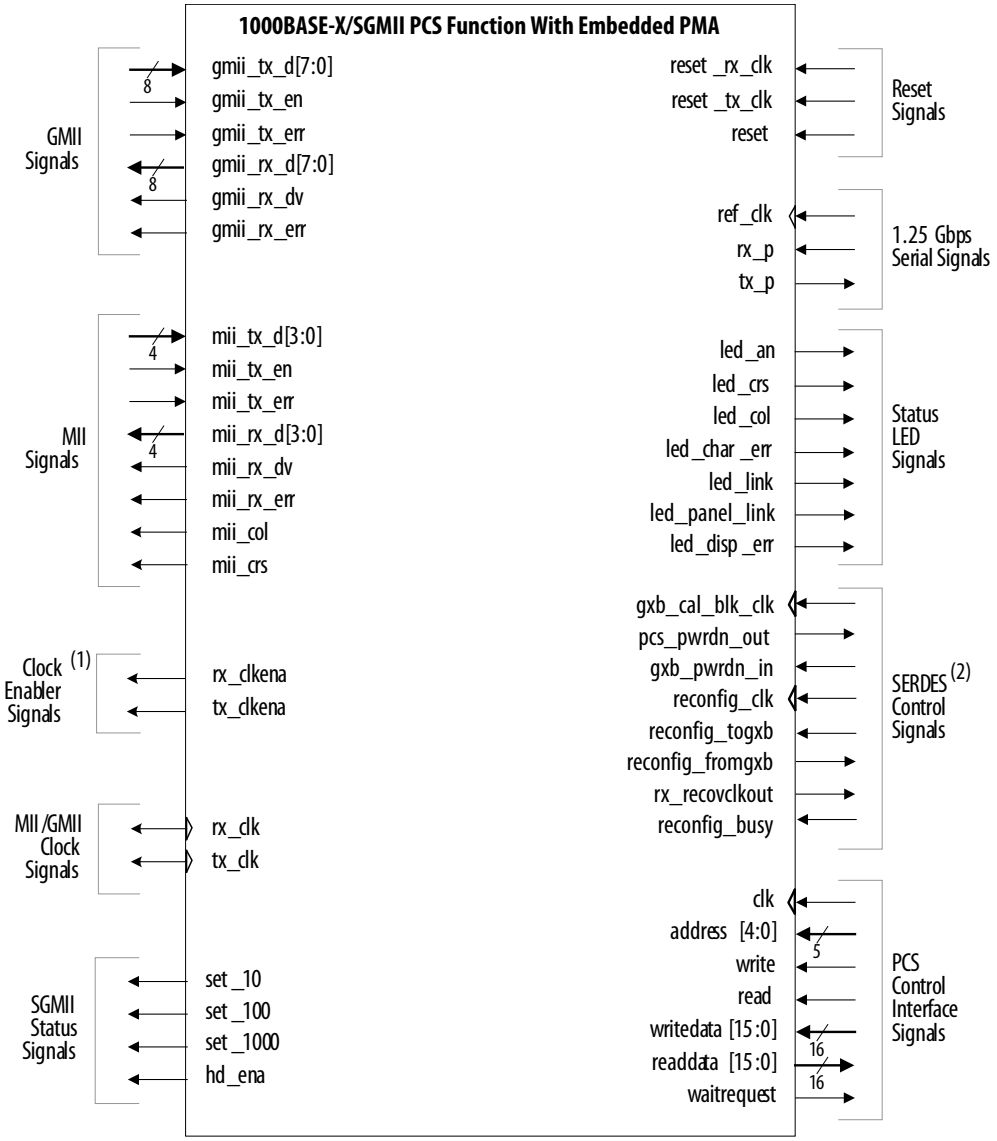
#### Table 103. TBI Interface Signals for External SERDES Chip

| Name | I/O | Description [9] |
|---|---|---|
| tbi2x_tx_d[19:0] | O | 2X TBI transmit data. The PCS function transmits data on this bus synchronous to tbi2x_tx_clk. |
| tbi2x_tx_clk | I | 62.5-MHz TBI transmit clock from external SERDES, typically sourced by the local reference clock oscillator.<br>*Note:* If connected to the E-tile Native PHY, this signal can be sourced from tx_clkout. |
| tbi2x_rx_clk | I | 62.5-MHz TBI receive clock from external SERDES, typically sourced by the clock recovered from the encoded serial data.<br>*Note:* If connected to the E-tile Native PHY, this signal can be sourced from rx_clkout. |
| tbi2x_rx_d[19:0] | I | 2X TBI receive data. This bus carries the data from the external SERDES. Synchronize the bus with tbi2x_rx_clk. |

(9) All the signals in this table are not exported to the top level for 10/100/1000Mb Ethernet MAC with 1000BASE-X/SGMII 2XTBI PCS with E-tile GXB variant.

## 6.1.10. 1000BASE-X/SGMII PCS and PMA Signals

**Figure 56. 1000BASE-X/SGMII PCS Function and PMA Signals**




Notes to Figure 56 on page 140:

1. The clock enabler signals are present only in SGMII mode.
2. The SERDES control signals are present in variations targeting devices with GX transceivers. For Stratix II GX and Arria GX devices, the reconfiguration signals—`reconfig_clk`, `reconfig_togxb`, and `reconfig_fromgxb`—are included only when the option, **Enable transceiver dynamic reconfiguration**, is turned on. The reconfiguration signals—`gxb_cal_blk_clk`, `pcs_pwrdwn_out`, `gxb_pwrdn_in`, `reconfig_clk`, and `reconfig_busy`—are not present in variations targeting Stratix V devices with GX transceivers.

**Table 104. References**

| Interface Signal | Section |
|---|---|
| Reset signals | PCS Reset Signals on page 135 |
| MII/GMII clocks and clock enablers | MII/GMII Clocks and Clock Enablers on page 135 |
| PCS control interface | PCS Control Interface Signals on page 135 |
| GMII signals | GMII on page 136 |
| MII signals | MII on page 136 |
| SGMII status signals | SGMII Status Signals on page 137 |
| 1.25 Gbps Serial Signals | 1.25 Gbps Serial Interface on page 127 |
| Status LED signals | Status LED Control Signals on page 119 |
| SERDES control signals | SERDES Control Signals on page 120 |
| Transceiver Native PHY signal | Transceiver Native PHY Signal on page 127 |

## 6.2. Timing

This section shows the timing on the Triple-Speed Ethernet transmit and receive interfaces as well as the timestamp signals for the IEEE 1588v2 feature.

### Related Information

Avalon Interface Specifications
More information on Avalon Memory-Mapped control interface timing.

## 6.2.1. Avalon Streaming Receive Interface

**Figure 57. Receive Operation—MAC With Internal FIFO Buffers**

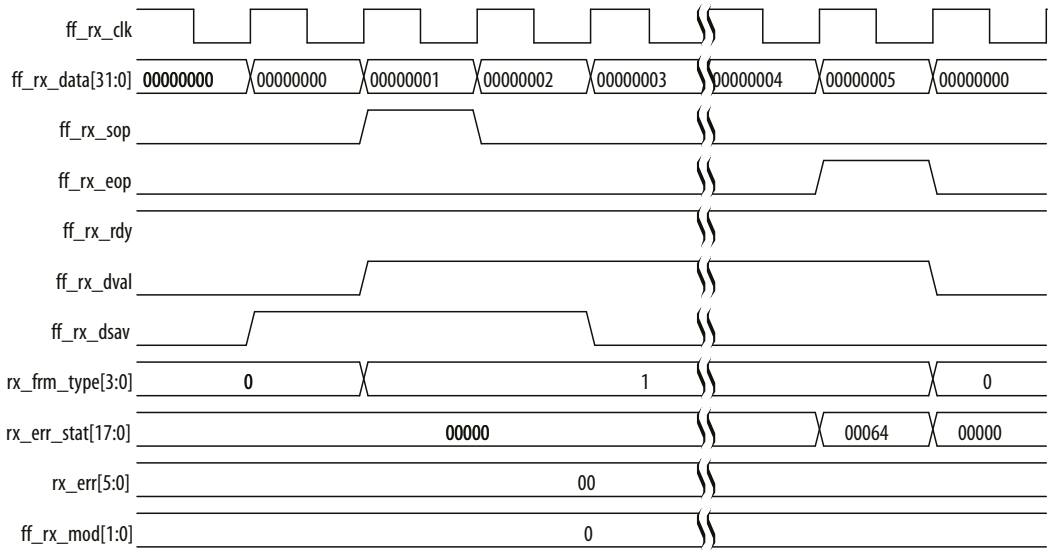


**Figure 58. Receive Operation—MAC Without Internal FIFO Buffers**

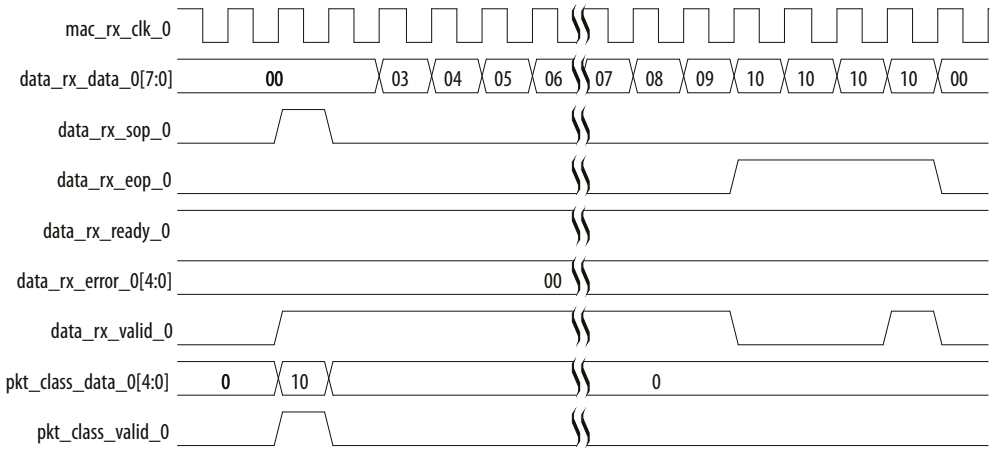



**Figure 59. Invalid Length Error During Receive Operation—MAC With Internal FIFO Buffer**

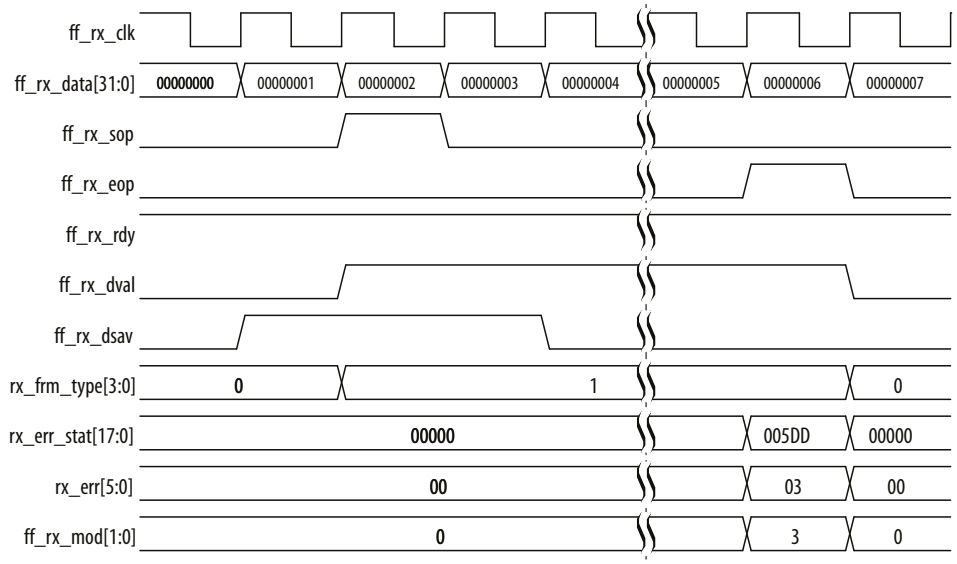
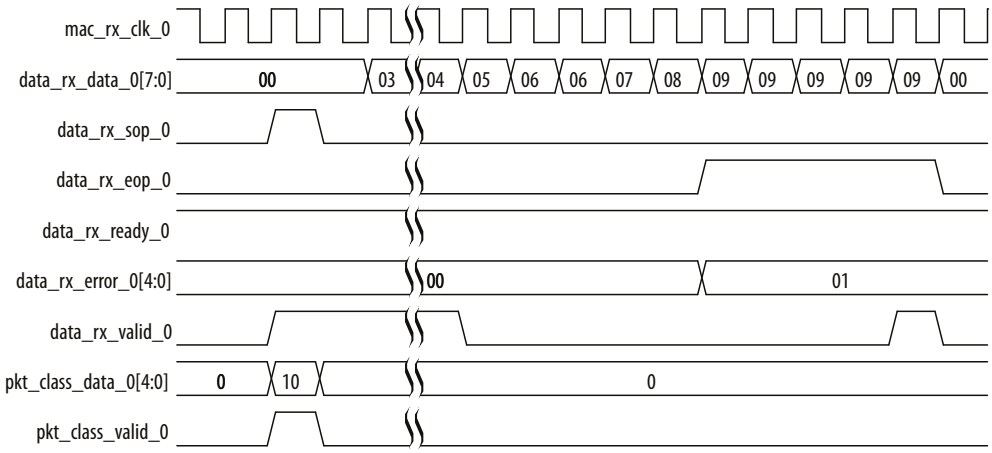


**Figure 60. Invalid Length Error During Receive Operation—MAC Without Internal FIFO Buffers**




## 6.2.2. Avalon Streaming Transmit Interface

**Figure 61. Transmit Operation—MAC With Internal FIFO Buffers**

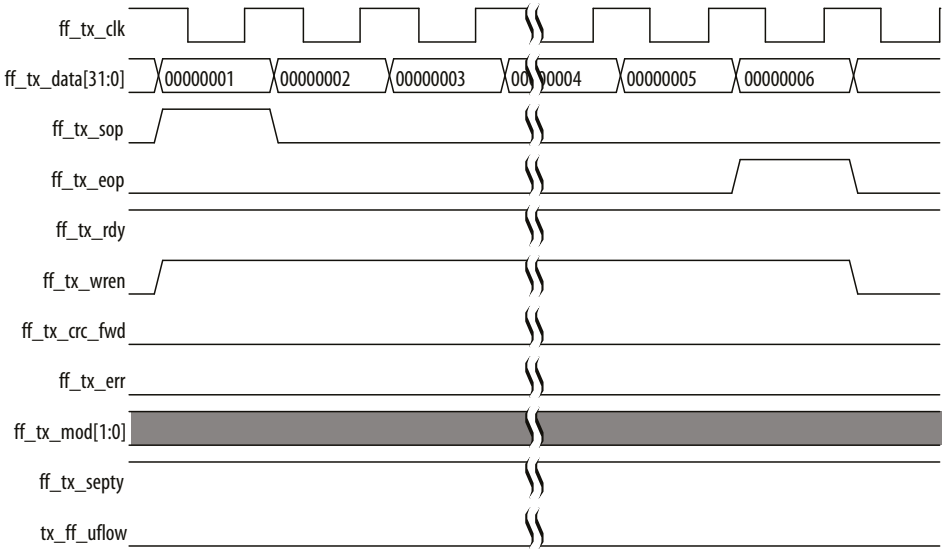
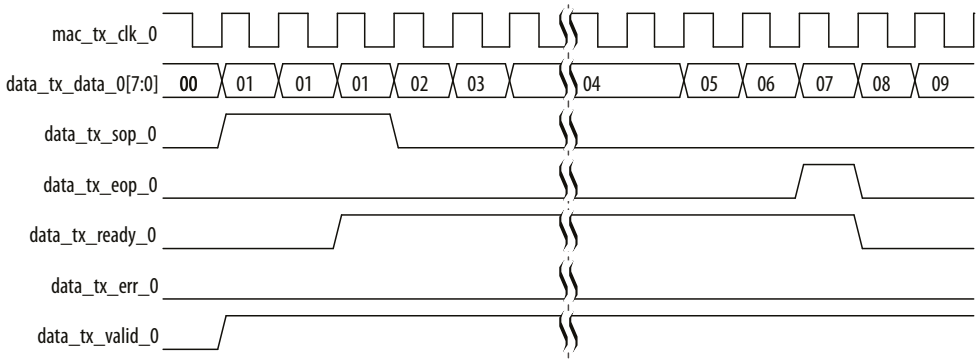


**Figure 62. Transmit Operation—MAC Without Internal FIFO Buffers**




## 6.2.3. GMII 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 Avalon Streaming 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.

## 6.2.4. GMII 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 Avalon Streaming interface with the error signal `rx_err[0]` asserted.

## 6.2.5. RGMII 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 63. RGMII Transmit in 10/100 Mbps**

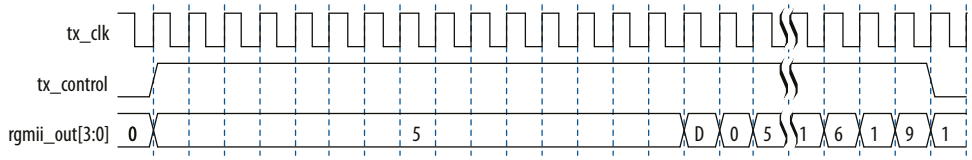
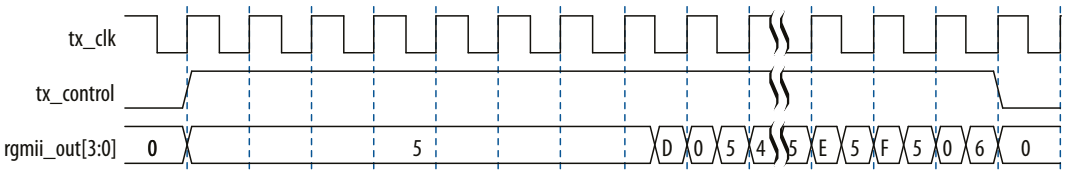
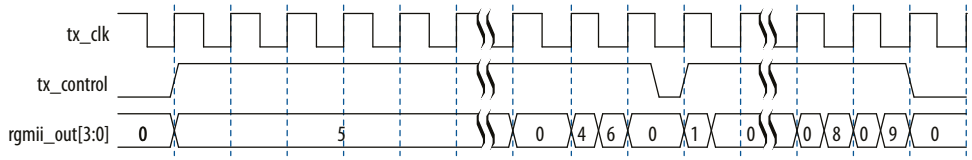



**Figure 64. RGMII Transmit in Gigabit Mode**




If a frame is received on the Avalon Streaming 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 65. RGMII Transmit with Error in 1000 Mbps**




## 6.2.6. RGMII 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 66. RGMII Receive in 10/100 Mbps**

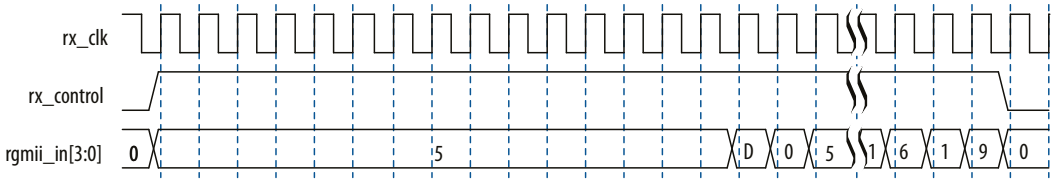
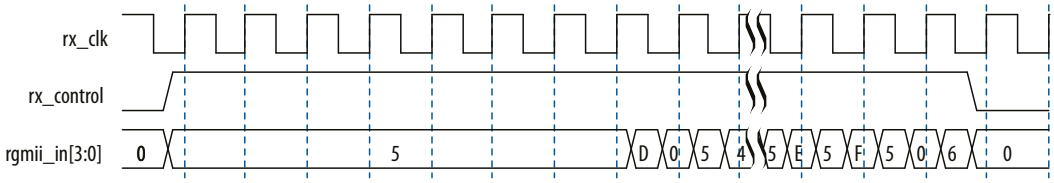
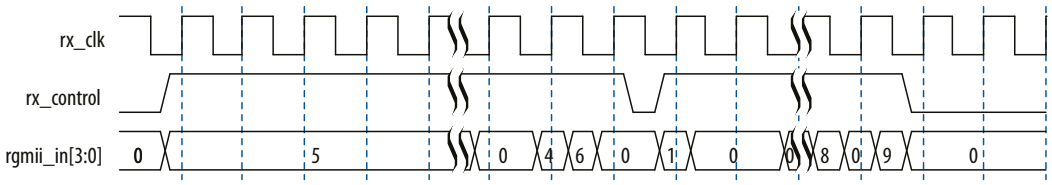



**Figure 67. RGMII Receive in 1000 Mbps**



A frame received on the RGMII interface with a PHY error indication is subsequently transferred on the Avalon Streaming interface with the error signal `rx_err[0]`asserted.

**Figure 68. 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).

## 6.2.7. MII 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.

## 6.2.8. MII 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.

## 6.2.9. IEEE 1588v2 Timestamp

The following timing diagrams show the timestamp of frames observed on TX path for the IEEE 1588v2 feature.

Figure below shows the TX timestamp signals for the IEEE 1588v2 feature in a 1-step operation.

In a 1-step operation, a TX egress timestamp is inserted into timestamp field of the PTP frame in the MAC. You need to drive the 1-step related signal appropriately so that the timestamp can be inserted into the correct location of the packet. The input signals related to the 2-step operation are not important and can be driven low or ignored.

**Figure 69. Egress Timestamp Insert for IEEE 1588v2 PTP Packet Encapsulated in IEEE 802.3**

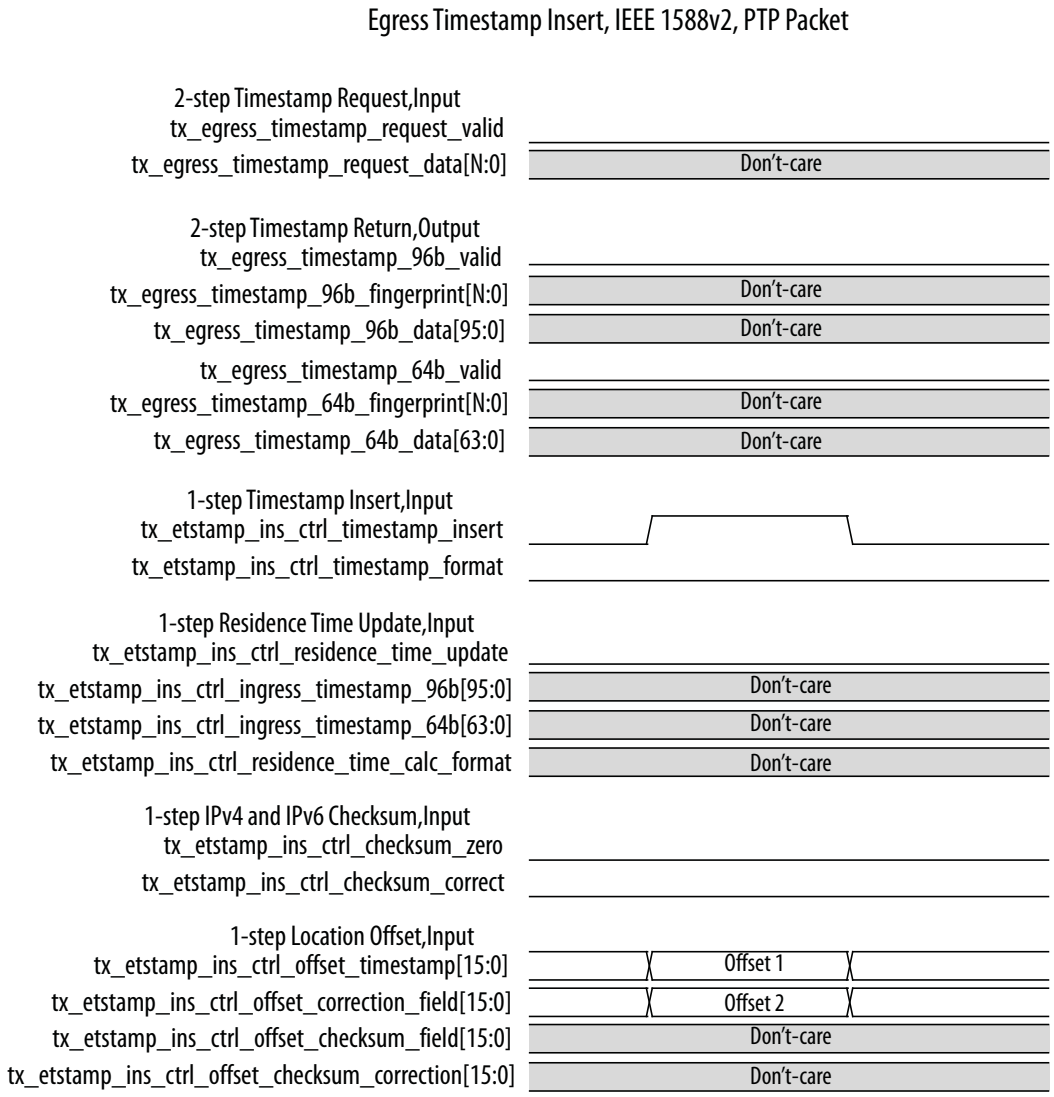



Figure 70 on page 148 shows the TX timestamp signals for the first type of egress correction field update, where the residence time is calculated by subtracting 96 bit ingress timestamp from 96 bit egress timestamp. The result is updated in the correction field of the PTP frame encapsulated over UDP/IPv4.

The `tx_etstamp_ins_ctrl_residence_time_calc_format` signal is driven low to indicate that this is a 96b residence time calculation. The `tx_etstamp_ins_ctrl_checksum_zero` signal is driven high to clear the UDP/IPv4 checksum field to all 0.

**Figure 70. Type 1 Egress Correction Field Update**

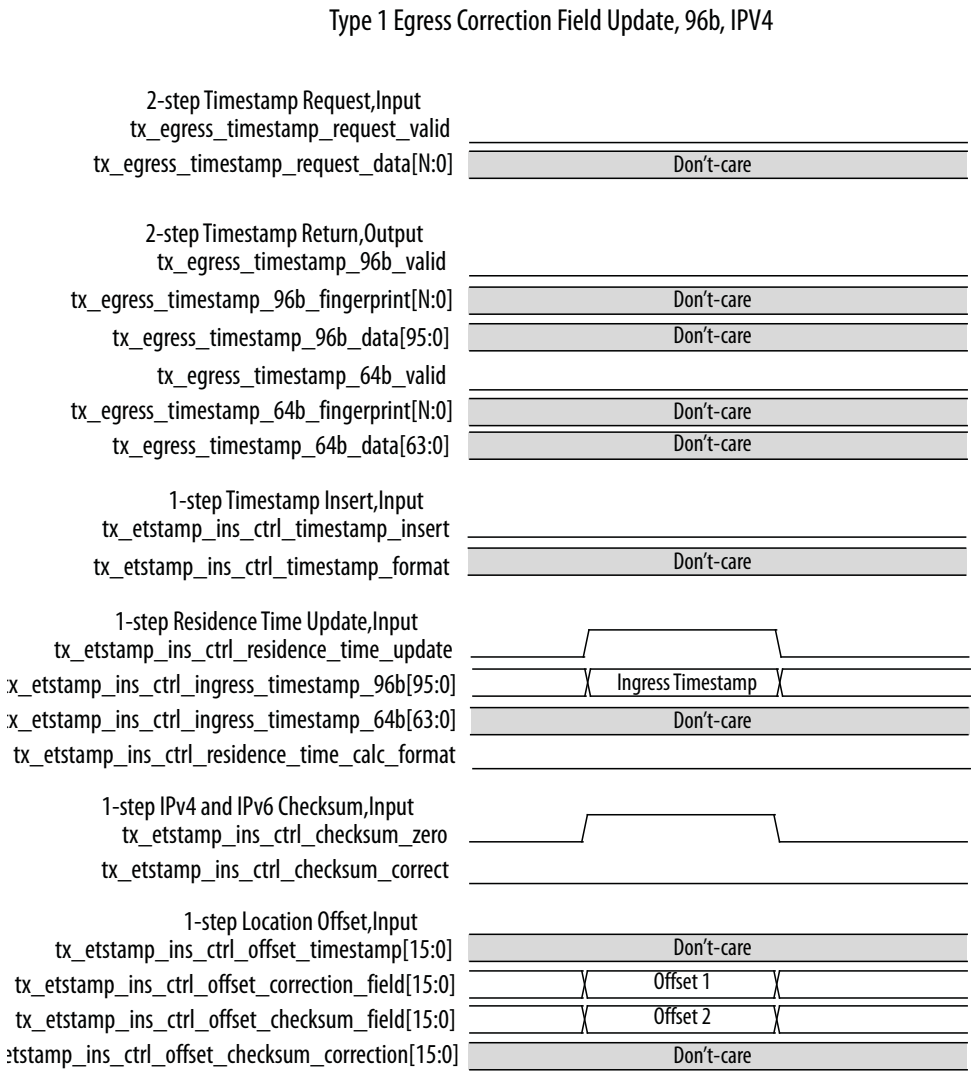



Figure 71 on page 149 shows the TX timestamp signals for the second type of egress correction field update, where the 64 bit ingress timestamp has been pre-subtracted from the correction field at the ingress port. At the egress port, the 64 bit egress timestamp is added into the correction field and the correct residence time is updated in the correction field. This is the example of PTP frame encapsulated over UPD/IPV6.

The `tx_etstamp_ins_ctrl_residence_time_calc_format` signal is driven high to indicate that this is a 64b residence time calculation. The `tx_etstamp_ins_ctrl_checksum_correct` signal is driven high to correct the packet UPD/IPV6 checksum by updating the checksum correction field.

**Figure 71. Type 2 Egress Correction Field Update**

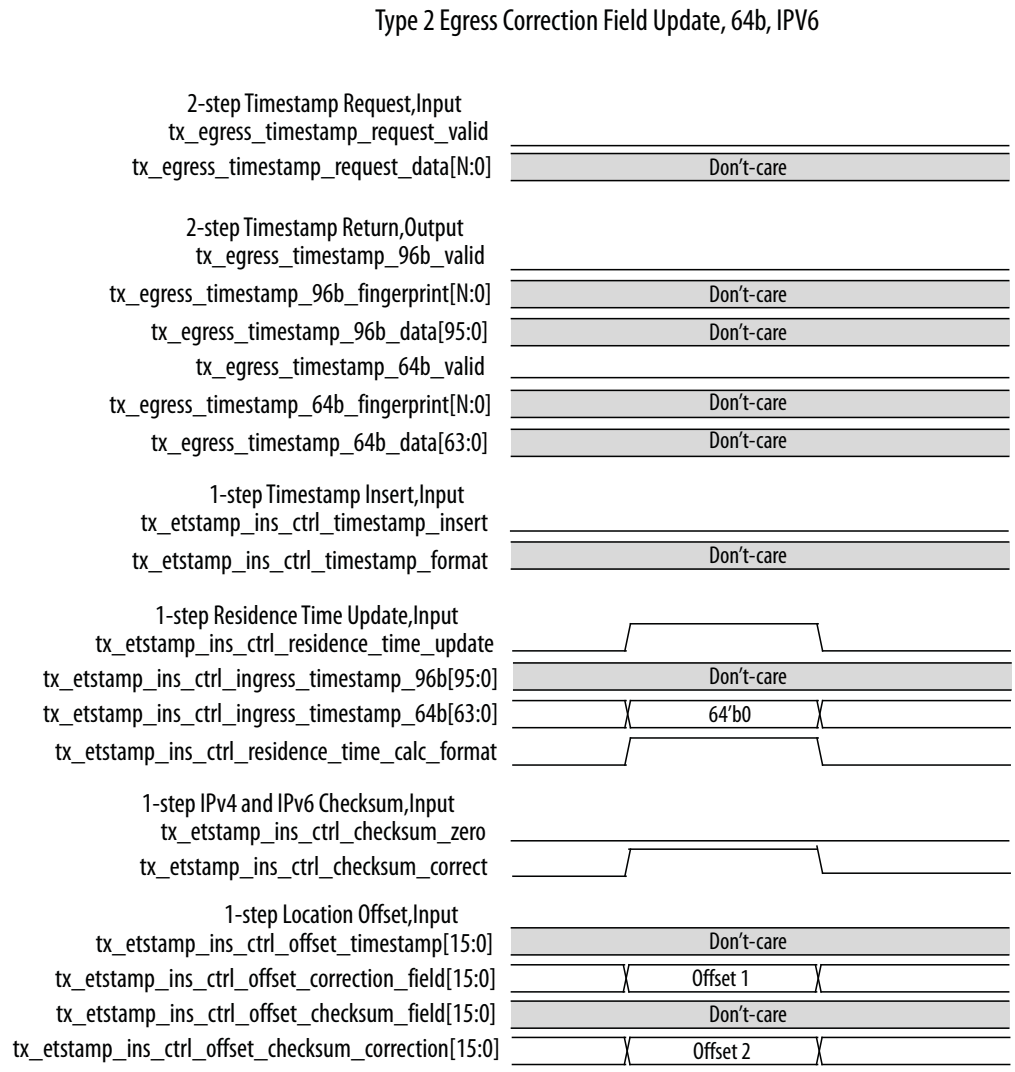
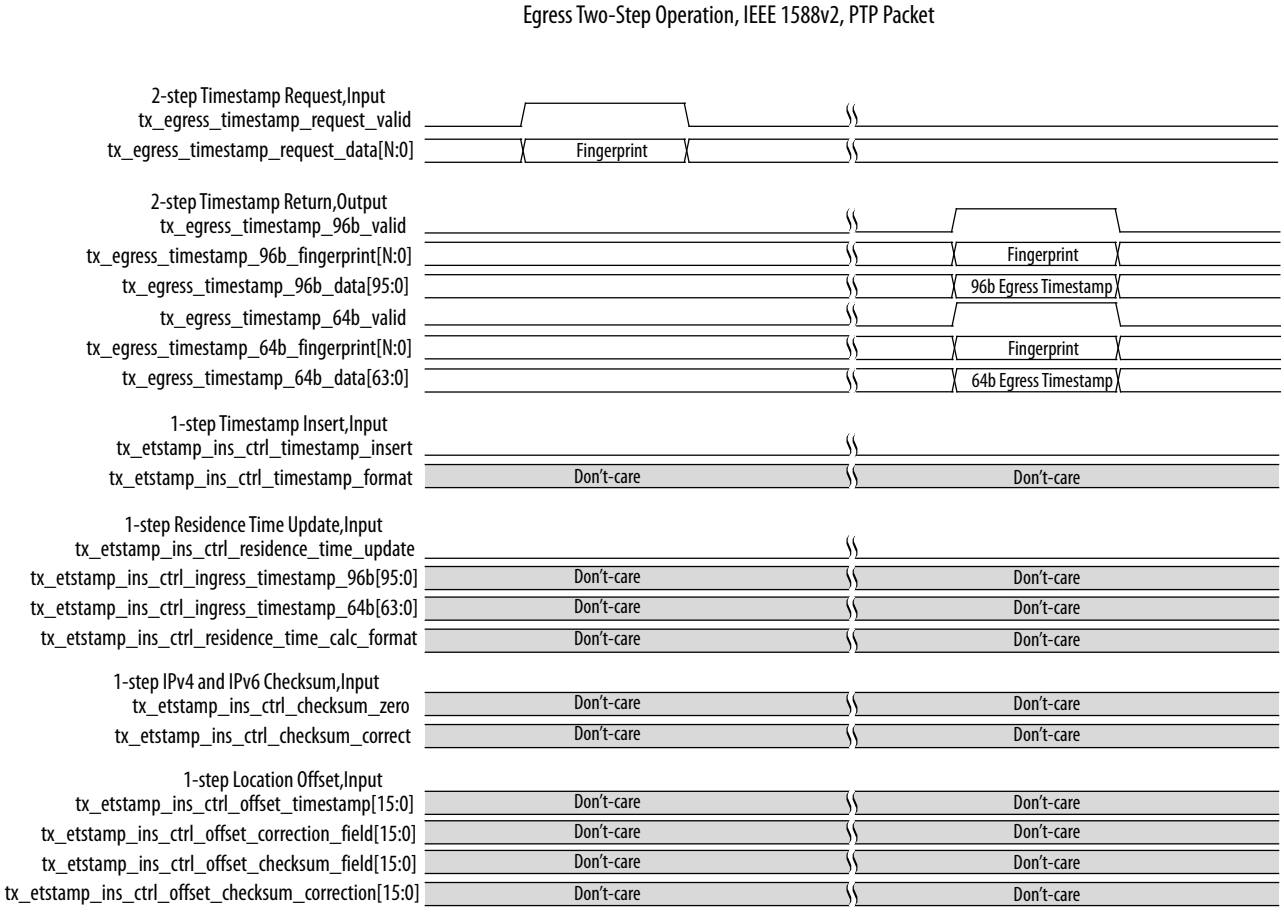



Figure 72 on page 150 shows the TX timestamp signals for the IEEE 1588v2 feature in a two step operation.

When the `tx_egress_timestamp_request_valid` signal is driven high with a unique fingerprint, the MAC returns an egress timestamp associated with that unique fingerprint. The signals related to the 1-step operation can be driven low or ignored. There is no modification to the packet content.

## Figure 72. Egress 2-Step Operation



# 7. Design Considerations

## 7.1. Optimizing Clock Resources in Multiport MAC with PCS and Embedded PMA

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

- Configuration of the Triple-Speed Ethernet IP core and 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 105. Clock Signals Visible at Top-Level Design**

Clock signals that are visible at the top-level design for each possible configuration.

| Clocks | Configurations (1) | | |
|---|---|---|---|
| | MAC Only | MAC and PCS | MAC and PCS with PMA |
| rx_recovclkout | — | — | Yes |
| ref_clk | — (2) | — (2) | Yes |
| clk | Yes | Yes | Yes |
| ff_tx_clk (3) | Yes | Yes | Yes |
| ff_rx_clk (3) | Yes | Yes | Yes |
| tx_clk | Yes | No | No |
| rx_clk | Yes | No | No |
| tbi_rx_clk | — | Yes | No |
| tbi_tx_clk | — | Yes | No |
| gxb_cal_blk_clk (4) | — | — | Yes |
| reconfig_clk | — | — | Yes |
| Notes to Table 105 on page 151 : | | | |

*continued...*

| Clocks | Configurations (1) | | |
|---|---|---|---|
| | MAC Only | MAC and PCS | MAC and PCS with PMA |

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. This clock is visible at the top-level design and needs to be connected to a clock source for the multiport fifoless Ethernet MAC ONLY and fifoless Ethernet MAC with SGMII PCS configurations. For single channel fifoless MAC configuration ONLY and fifoless MAC with SGMII disabled in PCS, this clock is not used and can be tied to GND.
3. These signals are only visible on the single port MAC with internal FIFO buffers. They are only visible when the multiport Ethernet design is implemented through the replication of the single port MAC with internal FIFO buffers.
4. Applies to GX transceiver.

### 7.1.1. MAC and PCS With GX Transceivers

In configurations that contain the MAC, PCS, and 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.

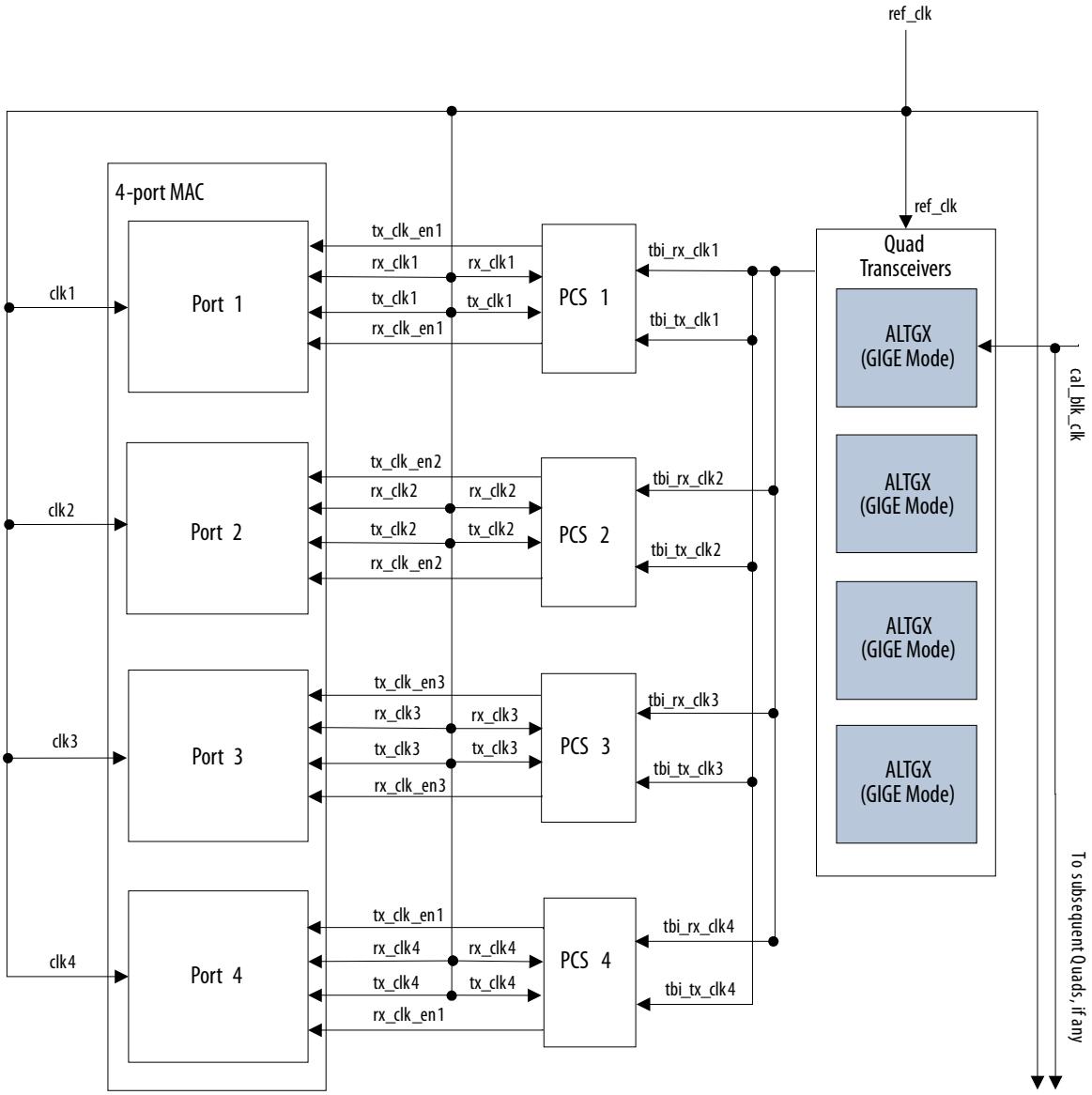
The Intel Quartus Prime software automatically optimizes the TBI transmit clocks. Only one clock source drives the TBI transmit clocks from each PMA quad.

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.

*Note:* If you do not constrain the PLL inputs and outputs in your design, add `derive_pll_clocks` in the timing constraint file to ensure that the Timing Analyzer automatically creates derived clocks for the PLL outputs.

## Figure 73. Clock Distribution in MAC and SGMII PCS with GXB Configuration—Optimal Case

Figure shows the optimal clock distribution scheme you can achieve in configurations that contain the 10/100/1000 Ethernet MAC, SGMII PCS, and GX transceivers.



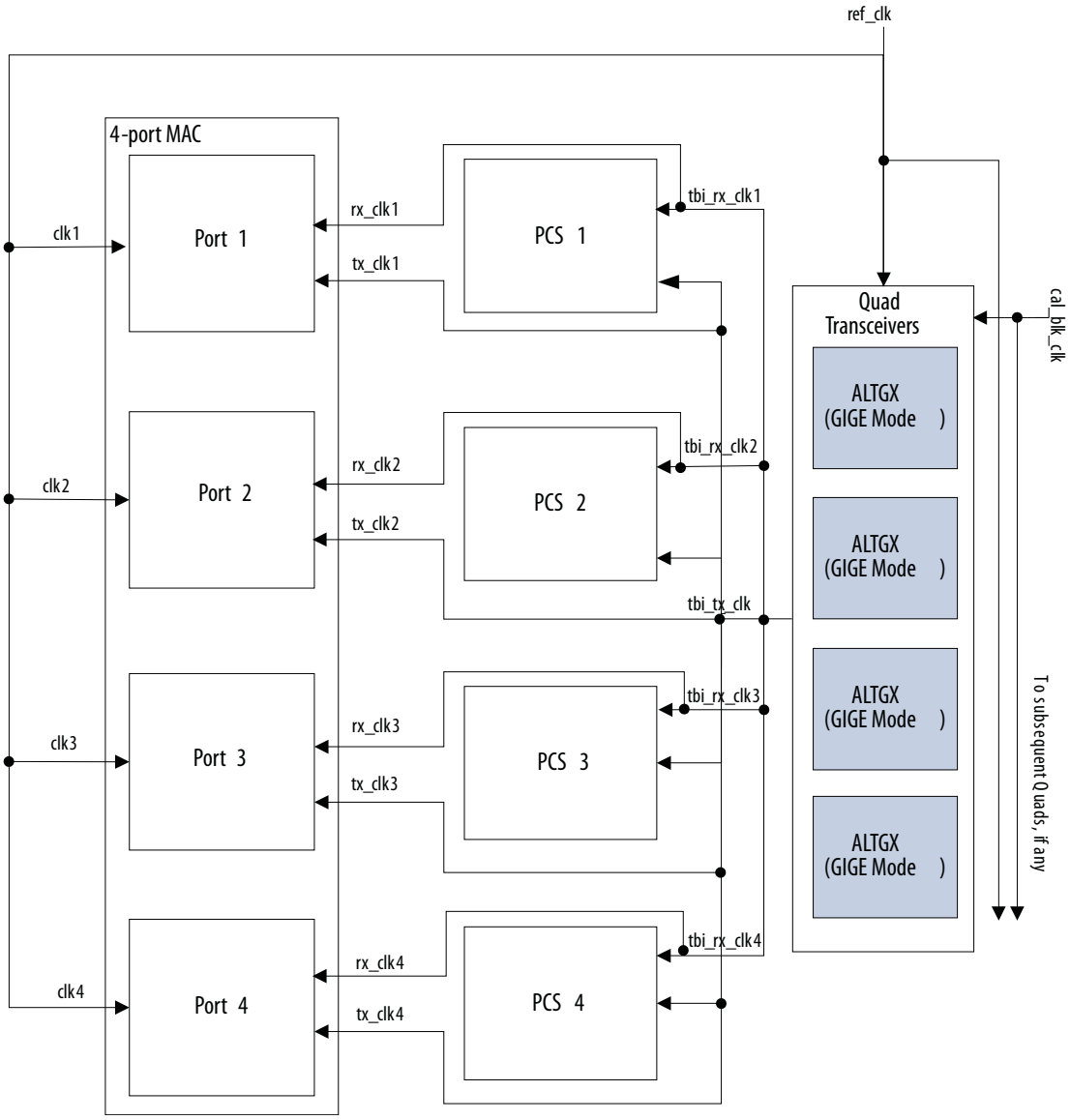

Note to Figure 73 on page 153:

1. The PMA layer in devices with GX transceivers uses ALTGX IP.

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 74. Clock Distribution in MAC and 1000BASE-X PCS with GXB Configuration—Optimal Case**

Figure shows the optimal clock distribution scheme you can achieve in configurations that contain the 10/100/1000 Ethernet MAC, 1000Base-X PCS, and GX transceivers.




Note to Figure 74 on page 154 :

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

## 7.1.2. MAC and PCS With LVDS Soft-CDR I/O

In configurations that contain the MAC, PCS, and 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 do not 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 75. Clock Distribution in MAC and SGMII PCS with LVDS Configuration—Optimal Case

Figure shows the optimal clock distribution scheme you can achieve in configurations that contain the MAC, SGMII PCS and LVDS Soft-CDR I/O.

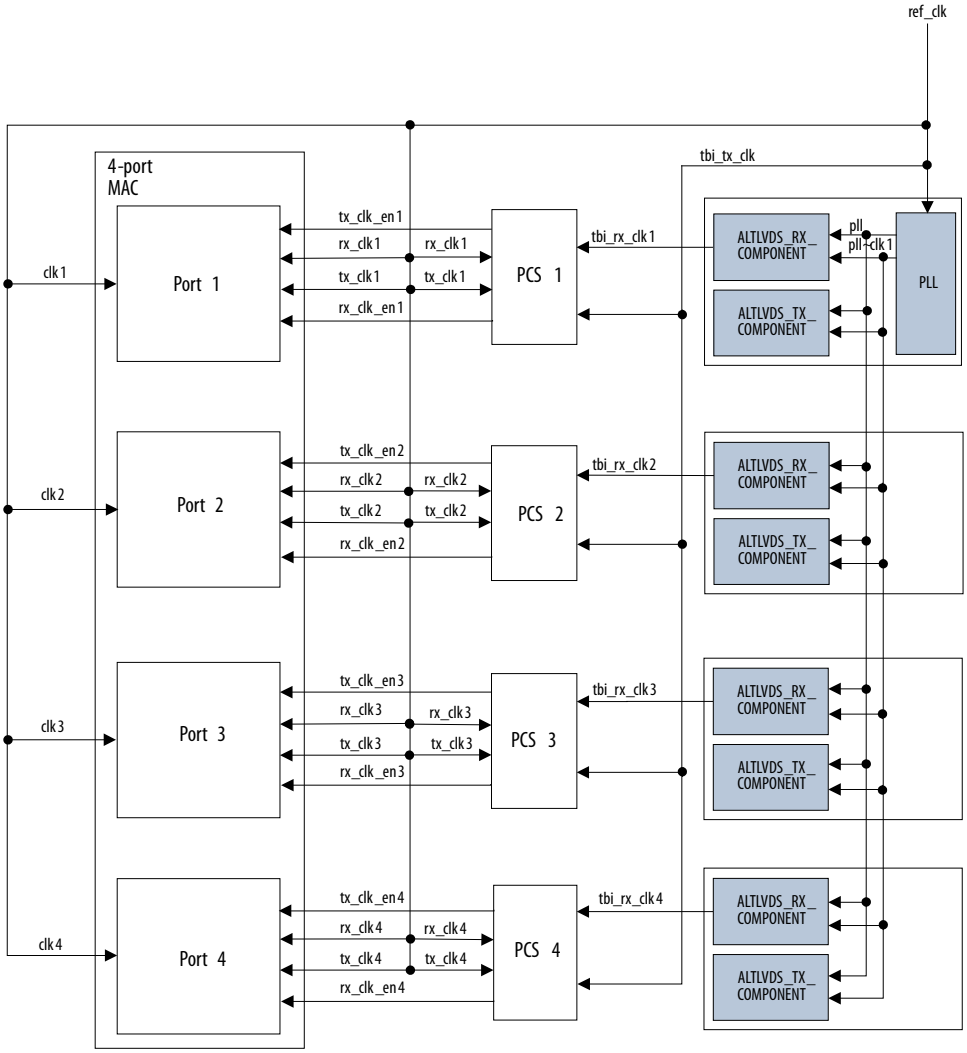
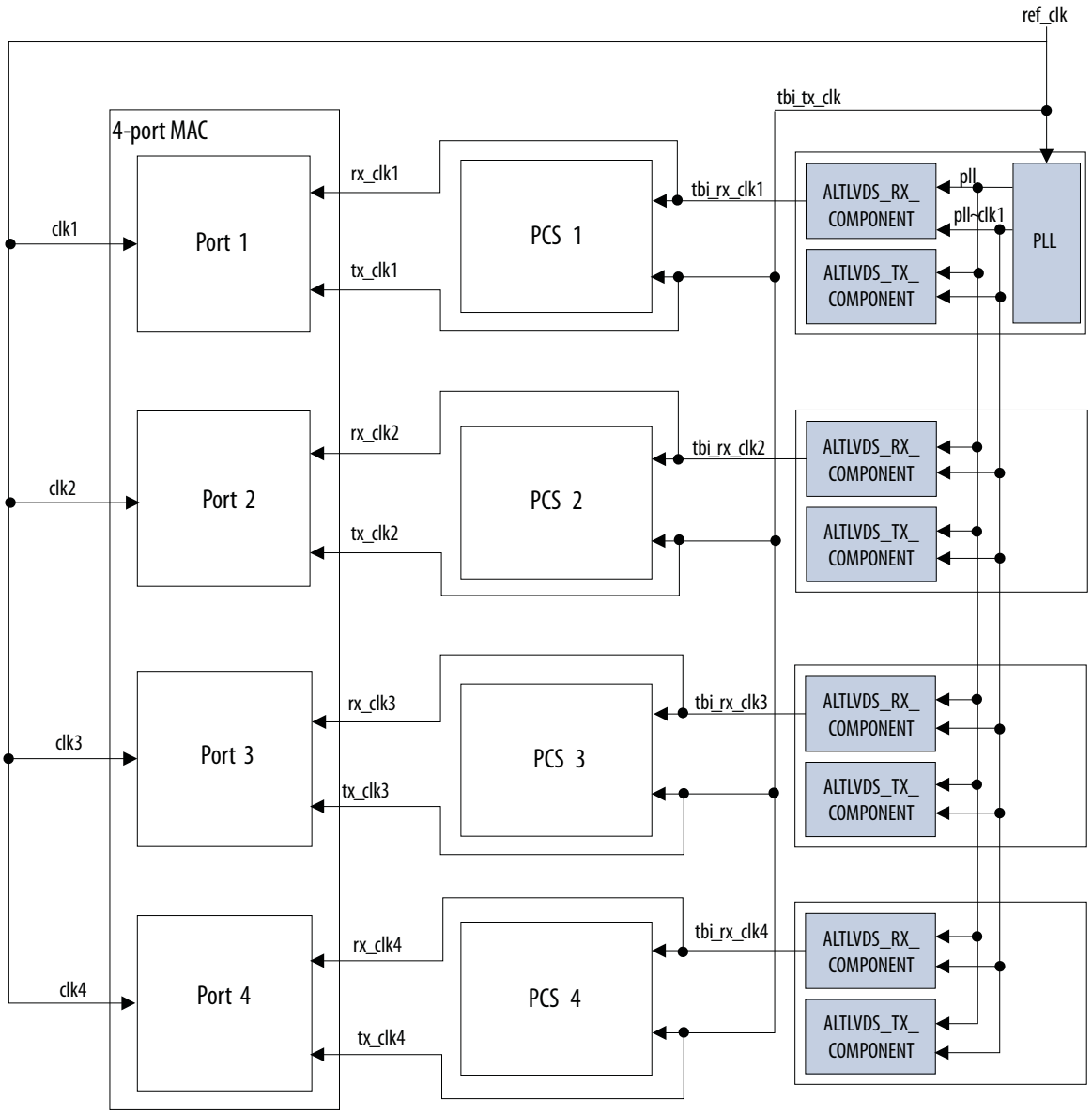


**Figure 76. Clock Distribution in MAC and 1000BASE-X PCS with LVDS Configuration—Optimal Case**

Figure shows the optimal clock distribution scheme you can achieve in configurations that contain the MAC, 1000BASE-X PCS, and LVDS Soft-CDR I/O.




Notes to Figure 75 on page 155 and Figure 76 on page 156:

1. There may be a performance risk if you use the Triple-Speed Ethernet IP variant with LVDS I/O for PMA implementation in Intel Arria 10 devices for Intel Quartus Prime software versions 17.0.2 and earlier. To avoid the performance risk, Intel recommends that you regenerate the Triple-Speed Ethernet IP core and recompile

the design in the Intel Quartus Prime software version 17.1 or later. To download and install the software patch for Intel Quartus Prime version 17.0.2, refer to KDB Link: *Performance Risk Running Triple Speed Ethernet LVDS in Arria 10 Devices*.

2. For Intel Quartus Prime software version 17.1 onwards, the number of ports supported for Triple-Speed Ethernet design targeting Intel Stratix 10, Intel Arria 10, and Intel Cyclone 10 GX devices is 8 per instance. To avoid performance risk, you must not promote the reference clock to global clock manually. Assign the number of ports supported and its reference clock to the same I/O bank as inter-bank clock sharing is not allowed.

**Related Information**

- Core Configuration on page 26
- KDB Link: Performance Risk Running Triple Speed Ethernet LVDS in Intel Arria 10 Devices
- Sharing PLLs in Devices with LVDS Soft-CDR I/O on page 157

## 7.2. Sharing PLLs in Devices with LVDS Soft-CDR I/O

For designs that contain multiple instances of MAC and PCS with PMA or PCS with PMA variation targeting devices with LVDS soft-CDR I/O, you can optimize resource utilization by sharing the PLLs.

The Intel Quartus Prime software merges the PLLs for these instances if you implement the following items in your design:

- Connect the reference clock of each instance to the same source.
- Place the LVDS I/O pins on the same side of the FPGA.

*Note:* For Intel Quartus Prime software version 17.1 onwards, the number of ports supported for Triple-Speed Ethernet design targeting Intel Stratix 10, Intel Arria 10, and Intel Cyclone 10 GX is 8 per instance. Assign the number of ports supported and its reference clock to the same I/O bank as inter-bank clock sharing is not allowed.

**Related Information**

MAC and PCS With LVDS Soft-CDR I/O on page 154

## 7.3. Sharing PLLs in Devices with GIGE PHY

For Cyclone V designs that contain multiple instances of MAC and PCS with PMA or PCS with PMA variation targeting devices with GIGE PHY, you can share the PLLs by placing the associated signals (`tx_p`, `rx_p`, and `ref_clk`) to the same I/O block of transceiver bank through pin assignment. Additionally, the `rx_recovclkout` clock must be buffered by two levels of inverter in the top level module so that it can be fitted to the general I/O pins.

## 7.4. Sharing Transceiver Quads

For designs that contain multiple PMA blocks targeting Intel FPGA device families with GX transceivers, you can combine the transceiver channels in the same quad. To share the same transceiver quad, the transceiver channels must have the same dynamic

reconfiguration setting. In other words, you must turn on dynamic reconfiguration capabilities in all channels in a quad even though you only intend to use these capabilities in some of the channels.

The dynamic reconfiguration is always turned on in devices other than Arria GX and Stratix II GX. When the dynamic reconfiguration is turned on in designs targeting devices other than Intel Arria 10, Stratix V, Arria V, Intel Cyclone 10 GX, and Cyclone V, Intel recommends that you connect the dynamic reconfiguration signals to the ALTGX_RECONFIG megafunction.

In Stratix V, Arria V, and Cyclone V devices, Intel recommends that you connect the dynamic reconfiguration signals to the Transceiver Reconfiguration Controller megafunction. For transceiver quad sharing between Triple-Speed Ethernet IP core and other IP cores that target these devices, reset signal for all the cores must be from the same source.

Refer to the respective device handbook for more information on dynamic reconfiguration signals in Intel FPGA devices.

## 7.5. Migrating From Old to New User Interface For Existing Designs

In the Intel Quartus Prime software version 13.0, the old Triple-Speed Ethernet IP core user interface is deprecated. Existing Triple-Speed Ethernet designs generated prior to version 13.0 can still load properly in the Intel Quartus Prime software version 13.0. However, starting from version 13.1, the old Triple-Speed Ethernet interface and design generated using the old interface will not be supported.

You need to manually migrate your design to the new user interface. Reopening and saving the existing design created with the old user interface will not automatically convert the design to the new user interface.

To migrate your design to the new user interface, launch the Intel Quartus Prime software version 13.0 or later, create a new project, and specify the parameters as described in Design Walkthrough on page 21.

*Note:* For target devices with LVDS I/O such as Intel Stratix 10, Intel Arria 10, and Intel Cyclone 10 GX, Intel recommends that you migrate your Triple-Speed Ethernet designs to Intel Quartus Prime software version 17.1 and later.

### 7.5.1. Exposed Ports in the New User Interface

In the new user interface in Platform Designer, for a design that has a MAC function, you have to manually connect the exposed ports or terminate them.

In MAC variation with internal FIFO buffers, the ready latency is two in both standalone and Platform Designer flow. The Platform Designer system inserts a timing adapter to change the ready latency to zero.

**Table 106. Exposed Ports and Recommended Termination Value for MAC Variation With Internal FIFO Buffers**

| Port Name | I/O | Width | Recommended Termination Value |
|---|---|---|---|
| xon_gen | I | 1 | 1'b0 |
| xoff_gen | I | 1 | 1'b0 |
| magic_wakeup | O | 1 | Left open |
| magic_sleep_n | I | 1 | 1'b1 |
| ff_tx_crc_fwd | I | 1 | 1'b0 |
| ff_tx_septy | O | 1 | Left open |
| tx_ff_uflow | O | 1 | Left open |
| ff_tx_a_full | O | 1 | Left open |
| ff_tx_a_empty | O | 1 | Left open |
| rx_err_stat | O | 18 | Left open |
| rx_frm_type | O | 4 | Left open |
| ff_rx_dsav | O | 1 | Left open |
| ff_rx_a_full | O | 1 | Left open |
| ff_rx_a_empty | O | 1 | Left open |

The following table lists the following ports that are exposed in the Platform Designer system for a design that has MAC variation without internal FIFO buffers.

**Table 107. Exposed Ports and Recommended Termination Value for MAC Variation Without Internal FIFO Buffers**

| Port Name | I/O | Width | Recommended Termination Value |
|---|---|---|---|
| xon_gen_<n> | I | 1 | 1'b0 |
| xoff_gen_<n> | I | 1 | 1'b0 |
| magic_wakeup_<n> | O | 1 | Left open |
| magic_sleep_n_<n> | I | 1 | 1'b1 |
| ff_tx_crc_fwd_<n> | I | 1 | 1'b0 |

## 7.6. Clocking Scheme of MAC with 2XTBI PCS and Embedded PMA

The following is the clocking scheme of the design that contains MAC with 2XTBI and embedded PMA on E-tile:

- 2XTBI PCS runs on 125 MHz and 62.5 MHz clocks while the same 125 MHz is used by MAC.
- The 125 MHz and 62.5 MHz must be synchronous, where their rising edges must align and must have 0 ppm and phase shift.
- The E-tile Native PHY is the embedded PMA in this variant. The `tx_clkout` and `rx_clkout` on the E-tile Native PHY are used as clock source for 2XTBI PCS `tbi2x_tx_clk` and `tbi2x_rx_clk`.
- Logics are implemented in the PCS block for clock rate matching by default regardless whether the ENABLE_SGMII option is selected. Therefore, the 125 MHz and 62.5 Mhz clocks do not need to be at 0 ppm in comparison with `tx_clkout` and `rx_clkout`, which are usually provided by external SERDES.
- The E-tile Native PHY transceiver is driven by the 156.25 MHz clock.

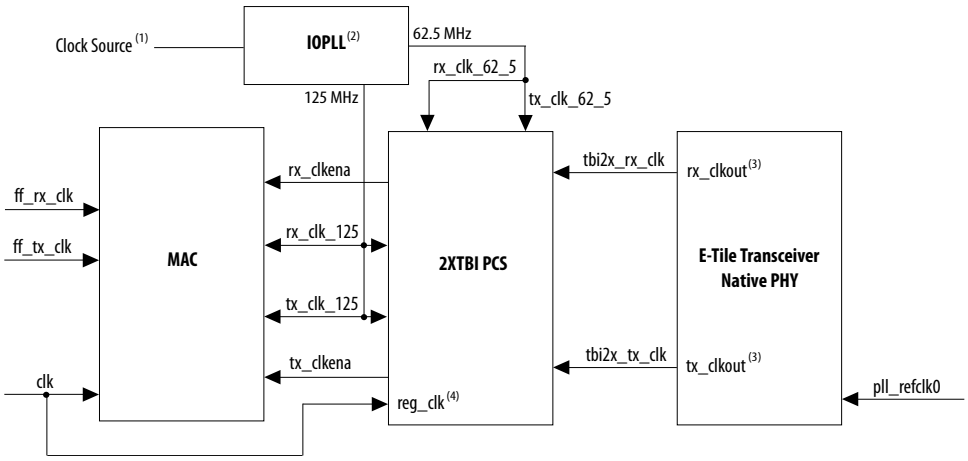
## Table 108. Clock Signals Visible at Top-Level Design

Clock signals that are visible at the top-level design for each possible configuration.

| Clocks | Configurations (1) | |
|---|---|---|
| | MAC and 2XTBI PCS with PMA | 2XTBI PCS Only |
| `clk` | Yes | N/A |
| `reg_clk` | No | Yes |
| `ff_tx_clk` | Yes | N/A |
| `ff_rx_clk` | Yes | N/A |
| `tx_clk_125` | Yes | Yes |
| `rx_clk_125` | Yes | Yes |
| `tx_clk_62_5` | Yes | Yes |
| `rx_clk_62_5` | Yes | Yes |
| `tbi2x_tx_clk` | No | Yes |
| `tbi2x_rx_clk` | No | Yes |
| `pll_refclk0` (2) | Yes | N/A |
| `tx_clkout` (2) | No | N/A |
| `rx_clkout` (2) | No | N/A |

Note to Table 108 on page 160:
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.
   N/A indicates that the clock is not applicable for the given configuration.
2. Clock signals of E-tile transceiver Native PHY.

## Figure 77. Clock Connectivity in MAC with 2XTBI PCS and Embedded PMA




Notes to Figure 77 on page 161:

1. Intel recommends that the `rx_clk_125`, `tx_clk_125`, `rx_clk_62_5`, and `tx_clk_62_5` share the same clock source.
2. Therefore, Intel recommends you to use one IOPLL with two output clocks to get the 125 MHz and 62.5 MHz clocks and connect to both the TX and RX datapaths.
3. `rx_clkout` and `tx_clkout` are output clocks generated by the E-tile transceiver Native PHY and internally connected to `tbi2x_rx_clk` and `tbi2x_tx_clk` in the variant MAC with 2XTBI and embedded PMA.
4. The `reg_clk` clock is internally connected to `clk` in the variant MAC with 2XTBI and embedded PMA. Refer to Table 94 on page 135 for more information about `reg_clk`.

# 8. Timing Constraints

Intel provides timing constraint files (`.sdc`) to ensure that the Triple-Speed Ethernet IP core meets the design timing requirements in Intel FPGA devices. The files constraints the false paths and multi-cycle paths in the Triple-Speed Ethernet IP core. The timing constraints files are specified in the `<variation_name>.qip` file and is automatically included in the Intel Quartus Prime project files.

You may need to add timing constraints that are external to the IP core. The following sections describe the procedure to create the timing constraint file.

## 8.1. Creating Clock Constraints

After you generate and integrate the Triple-Speed Ethernet IP core into the system, you need to create a timing constraints file to specify the clock constraint requirement.

You can specify the clock requirement in the timing constraint file using the following command:

```
create_clock
```

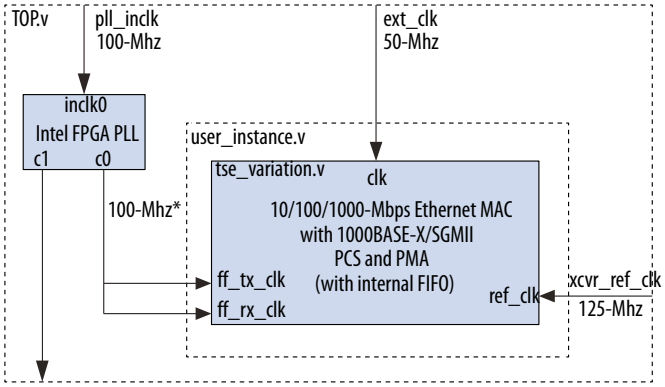
For example, for a new clock named "`reg_clk`", with a 50 MHz clock targeted to the top level input port "`clk`", enter the following command line:

```
create_clock -name "reg_clk" -period "50 MHz" [get_ports "clk"]
```

Figure below shows an example of how you can create a timing constraint file to constrain the Triple-Speed Ethernet IP core clocks.

**Figure 78. Triple-Speed Ethernet Timing Constraint Example**

The `reconfig_clk` signal is not shown in this example. Constrain the `reconfig_clk` based on your design implementation.




* For 32-bit FIFO width, Intel recommneds to use 100 MHz for c0 PLL output clock.
For 8-bit FIFO width, Intel recommneds to use 125 MHz for c0 PLL output clock.

The example above consists of the following Verilog modules:

- `TOP.v`—The top level design module which contains an Intel FPGA PLL and a user-defined instance. The top level input clocks consist of `pll_inclk`, `ext_clk`, and `xcvr_ref_clk`.
- `user_instance.v`—The user-defined instance that instantiates the Triple-Speed Ethernet IP core.
- `tse_variation.v`—A Triple-Speed Ethernet IP core variation. This example uses a 10/100/1000-Mbps Ethernet MAC with an internal FIFO buffer, a 1000BASE-X/SGMII PCS, and an embedded PMA.

The frequency for the PLL clock input, `inclk0`, is 100 MHz, and the frequency for the PLL clock output, `c0`, is 100 MHz. The Triple-Speed Ethernet MAC Avalon Streaming clocks, `ff_tx_clk` and `ff_rx_clk`, use `c0` as the clock source. The input clock frequency for the transceiver reference clock, `xcvr_ref_clk`, is 125 MHz.

Example of the Triple-Speed Ethernet IP core timing constraint file:

```
# PLL clock input, 100 MHz
create_clock -name pll_inclk -period 10.000 [get_ports {pll_inclk}]

# ext_clk, 50 MHz
create_clock -name ext_clk -period 20.000 [get_ports {ext_clk}]

# xcvr_ref_clk, 125 MHz
create_clock -name xcvr_ref_clk -period 8.000 [get_ports {xcvr_ref_clk}]

# Derive PLL generated output clocks.
derive_pll_clocks
```

*Note:* The `derive_pll_clocks` command is not supported in Intel Stratix 10 devices because all PLL clocks are automatically generated by the SDC files generated alongside the PLL IP.

## 8.2. Recommended Clock Frequency

**Table 109. Recommended Clock Input Frequency For Each IP Core Variant**

| IP Core Variant | Clock | Recommended Frequency (MHz) |
|---|---|---|
| 10/100/1000-Mbps Ethernet MAC (with Internal FIFO buffers) | CLK | 50–125 |
| | TX_CLK | 125 |
| | RX_CLK | 125 |
| | FF_TX_CLK | • For 32 bits FIFO: 100<br>• For 8 bits FIFO: 125 |
| | FF_RX_CLK | • For 32 bits FIFO: 100<br>• For 8 bits FIFO: 125 |
| 10/100/1000-Mbps Ethernet MAC (without Internal FIFO buffers) | CLK | 50–125 |
| | TX_CLK <N> | 125 |
| | RX_CLK <N> | 125 |

*continued...*

| IP Core Variant | Clock | Recommended Frequency (MHz) |
|---|---|---|
| | RX_AFULL_CLK | 100 |
| 10/100/1000-Mbps Ethernet MAC with 1000BASE-X/SGMII PCS (with Internal FIFO buffers) | CLK | 50–125 |
| | FF_TX_CLK | • For 32 bits FIFO: 100<br>• For 8 bits FIFO: 125 |
| | FF_RX_CLK | • For 32 bits FIFO: 100<br>• For 8 bits FIFO: 125 |
| | TBI_TX_CLK | 125 |
| | TBI_RX_CLK | 125 |
| | REF_CLK | 125 |
| | RECONFIG_CLK (10) | 37.5–50 |
| | GXB_CAL_BLK_CLK | 125 |
| 10/100/1000-Mbps Ethernet MAC with 1000BASE-X/SGMII PCS (without Internal FIFO buffers) | CLK | 50–125 |
| | RX_AFULL_CLK | 100 |
| | TBI_TX_CLK <N> | 125 |
| | TBI_RX_CLK <N> | 125 |
| | REF_CLK | 125 |
| | RECONFIG_CLK <N> (10) | 37.5–50 |
| | GXB_CAL_BLK_CLK | 125 |
| 1000BASE-X/SGMII PCS only | CLK | 50–125 |
| | REF_CLK | 125 |
| | TBI_TX_CLK | 125 |
| | TBI_RX_CLK | 125 |
| 10/100/1000 Ethernet MAC with 1000BASE-X/SGMII 2XTBI (with Internal FIFO buffers) | CLK | 50–125 |
| | TX_CLK_125 | 125 |
| | RX_CLK_125 | 125 |
| | FF_TX_CLK | • For 32 bits FIFO: 100<br>• For 8 bits FIFO: 125 |
| | FF_RX_CLK | • For 32 bits FIFO: 100<br>• For 8 bits FIFO: 125 |
| | TX_CLK_62_5 | 62.5 |
| | RX_CLK_62_5 | 62.5 |
| | PLL_REFCLK0 | 156.25 |
| 1000BASE-X/SGMII 2XTBI PCS only | CLK | 50–125 |

*continued...*

(10) This signal is only applicable to all device family prior to the 28-nm devices, which consists of the Stratix V, Arria V, Arria V GZ, and Cyclone V devices.

| IP Core Variant | Clock | Recommended Frequency (MHz) |
|---|---|---|
| | TX_CLK_125 | 125 |
| | RX_CLK_125 | 125 |
| | TX_CLK_62_5 | 62.5 |
| | RX_CLK_62_5 | 62.5 |

# 9. Testbench

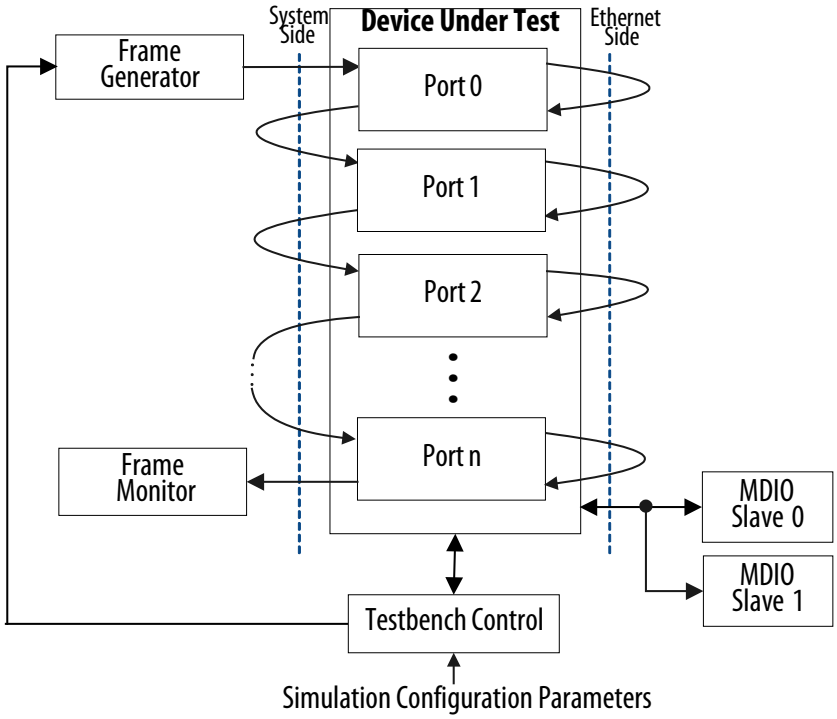
You can use the testbench provided with the Triple-Speed Ethernet IP core to exercise your custom IP core variation. The testbench includes the following features:

- 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 IP core variations created using Platform Designer.

## 9.1. Triple-Speed Ethernet Testbench Architecture

**Figure 79. Triple-Speed Ethernet Testbench Architecture**




## 9.2. Testbench Components

The testbench comprises the following modules:

- Device under test (DUT)—Your custom IP core variation
- Avalon Streaming Ethernet frame generator—Simulates a user application connected to the MAC system-side interface. It generates frames on the Avalon Streaming transmit interface.
- Avalon Streaming Ethernet frame monitor—Simulates a user application receiving frames from the MAC system-side interface. It monitors the Avalon Streaming 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 110. Testbench Components**

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

## 9.3. Testbench 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.

## 9.4. Testbench 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 FIFO buffers. 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.

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 a single-channel MAC with internal FIFO buffers, the transmit FIFO buffer is set to start data transmission as soon as its level reaches `tx_section_full`. The receive FIFO buffer is set to begin forwarding Ethernet frames to the Avalon Streaming receive interface when its 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.

## 9.5. Test Flow

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

- Initializes the DUT registers.
- Starts transmission. For a single-channel MAC with internal FIFO buffers, 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.

## 9.6. Simulation Model

This section describes the step-by-step instructions for generating the simulation model and simulating your design using the ModelSim simulator or other simulators.

### 9.6.1. Generate the Simulation Model

The generated design example includes both Verilog HDL and VHDL testbench files for the device under test (DUT)—your custom IP core variation.

To generate a Verilog functional simulation model, use the command prompt and run the `quartus_sh -t generate_sim_verilog.tcl` file. Alternatively, perform the following steps:

1. Launch the Intel Quartus Prime software and browse to the <*variation name*>**_testbench** directory.
2. Open the **generate_sim.qpf** file from the project directory.
3. On the **Tools** menu, select **Tcl Scripts** and select the **generate_sim_verilog.tcl** file.
4. Click **Run**.

To generate a VHDL functional simulation model, you can use the command prompt and run the `quartus_sh -t generate_sim_vhdl.tcl` file. Alternatively, perform the following steps:

1. Launch the Intel Quartus Prime software and browse to the <*variation name*>**_testbench** directory.
2. Open the **generate_sim.qpf** file from the project directory.
3. On the **Tools** menu, select **Tcl Scripts** and browse to the **generate_sim_vhdl.tcl** file.
4. Click **Run**.

## 9.6.2. Simulate the IP Core

You can simulate your IP core variation with the functional simulation model and the testbench or design example generated with your IP core. The functional simulation model and testbench files are generated in a project subdirectory. This directory may also include scripts to compile and run the testbench.

For a complete list of models or libraries required to simulate your IP core, refer to the scripts provided with the testbench in Simulation Model Files on page 171.

Generate the simulation model as shown in Generate the Simulation Model on page 169 before simulating the testbench design.

To use the ModelSim® simulation software to simulate the testbench design, follow these steps:

1. For Verilog testbench design:
   a. Browse to the following project directory: <*variation name*>**_testbench/ testbench_verilog/**<*variation name*>
   b. Run the following command to set up the required libraries, to compile the generated IP Functional simulation model, and to exercise the simulation model with the provided testbench:

      ```
      do run_<variation_name>_tb.tcl
      ```
2. For VHDL testbench design:
   a. Browse to the following project directory: <*variation name*>**_testbench/ testbench_vhdl/**<*variation name*>**_testbench**
   b. Run the following command to set up the required libraries, to compile the generated IP Functional simulation model, and to exercise the simulation model with the provided testbench:

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

For more information about simulating Intel FPGA IP cores, refer to the *Simulating Intel FPGA Designs* section in the respective *Intel Quartus Prime Pro Edition User Guide: Third-party Simulation* and *Intel Quartus Prime Standard Edition User Guide: Third-party Simulation*.

*Note:* Use the simulation models only for simulation and not for synthesis or any other purposes. Using these models for synthesis creates a nonfunctional design.

### Related Information

- Simulating Intel FPGA Designs
  More information in the Intel Quartus Prime Pro Edition User Guide: Third-party Simulation about simulating Intel FPGA IP cores.
- Simulating Intel FPGA Designs
  More information in the Intel Quartus Prime Standard Edition User Guide: Third-party Simulation about simulating Intel FPGA IP cores.

## 9.6.3. Simulation Model Files

Previously, the Triple-Speed Ethernet IP core generates a <variation_name>.vho or <variation_name>.vo file for VHDL or Verilog HDL IP functional simulation model.

For the new Triple-Speed Ethernet IP core created in Intel Quartus Prime software version 13.0, the simulation model will be generated using the industrial standard IEEE simulation encryption.

The following table lists the scripts available for you to compile the simulation model files in a standalone flow.

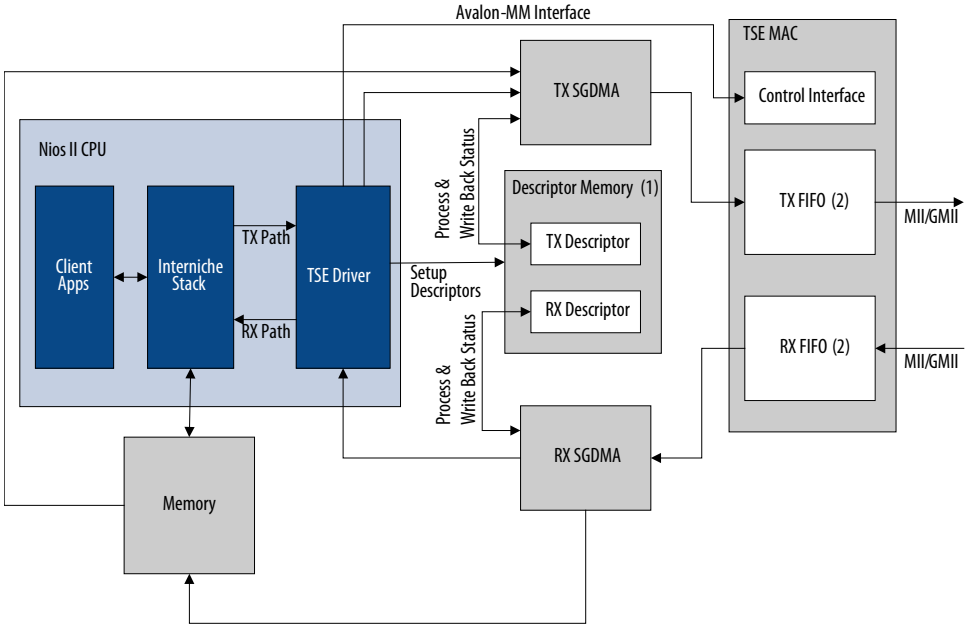
**Table 111. Simulation Model Files**

| Directory Name | Description |
|---|---|
| **sim/mentor/** | Contains a ModelSim script `msim_setup.tcl` to set up and run a simulation.<br>*Note:* The ModelSim-AE simulator is not supported for 10/100/1000 Ethernet MAC with 1000BASE-X/SGMII 2XTBI PCS variant. You must use another supported ModelSim simulator such as ModelSim SE. |
| **sim/synopsys/vcs/** | Contains a shell script `vcs_setup.sh` to set up and run a VCS* simulation. |
| **sim/synopsys/vcsmx/** | Contains a shell script `vcsmx_setup.sh` and **synopsys_sim.setup** to set up and run a VCS MX simulation. |
| **sim/cadence/** | Contains a shell script `ncsim_setup.sh` and other setup files to set up and run an NCSim simulation. |
| **sim/aldec/** | Contains a Riviera-PRO* script `rivierapro_setup.tcl` to set up and run a simulation.<br>*Note:* Riviera simulator is not supported for 10/100/1000 Ethernet MAC with 1000BASE-X/SGMII 2XTBI PCS variant. |
| **sim/xcelium/** | Contains a shell script `xcelium_setup.tcl` and other setup files to set up and run a simulation. |

# 10. Software Programming Interface

## 10.1. Driver Architecture

**Figure 80. Triple-Speed Ethernet Software Driver Architecture**



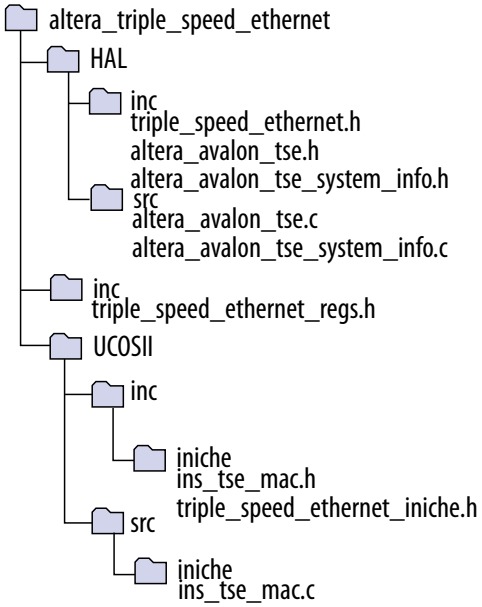

Notes to Figure 80 on page 172:

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 FIFO buffers, the transmit and receive FIFOs are external to the MAC function.

## 10.2. Directory Structure

Structure of the **altera_triple_speed_ethernet** directory.

**Figure 81. Directory Structure**




## 10.3. PHY Definition

By default, the software driver only supports the following PHYs:

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

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 112 on page 174.

**Table 112. PHY Speed Bit Values**

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

For PHYs that do not 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 of 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 of 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;
```

## 10.4. 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, Intel 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, Intel recommends that you use this memory space for only one purpose, that is storing unprocessed packets for the Triple-Speed Ethernet IP core. 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 IP core to optimize performance and resource usage, use the modified value to compute the new value of NUMBIGBUFS.

## 10.5. Using Jumbo Frames

To use jumbo frames, set the `frm_length` register to 9600 and edit the files and definitions.

**Table 113. Jumbo Frames Definitions**

| File | Definition |
|---|---|
| ip\altera\ethernet \altera_eth_tse\src \software\lib\UCOSII | #define ALTERA_TSE_PKT_INIT_LEN 8206<br>#define ALTERA_TSE_MAX_MTU_SIZE 8192<br>#define ALTERA_TSE_MIN_MTU_SIZE 14 |

*continued...*

| File | Definition |
|---|---|
| \inc\iniche\altera_eth_tse_iniche.h | |
| ip\altera\ethernet\altera_eth_tse\src\software\lib\HAL\inc\altera_avalon_tse.h | #define ALTERA_TSE_MAC_MAX_FRAME_LENGTH 8196 (1) |
| <*BSP project directory*>\iniche\src\h\nios2\ipport.h | #ifndef BIGBUFSIZE<br>#define BIGBUFSIZE 1536<br>#endif |

Note to Table 113 on page 175:
1. The maximum value for ALTERA_TSE_MAC_MAX_FRAME_LENGTH is defined by the frm_length register.

## 10.6. API Functions

This section describes each provided API function in alphabetical order.

### 10.6.1. alt_tse_mac_get_common_speed()

| | Details |
|---|---|
| **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 multiport 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.<br>TSE_PHY_SPEED_100 if the PHYs common speed is 100 Mbps.<br>TSE_PHY_SPEED_10 if the PHYs common speed is 10 Mbps.<br>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() |

### 10.6.2. alt_tse_mac_set_common_speed()

| | Details |
|---|---|
| **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> |

*continued...*

| | Details |
|---|---|
| **Description:** | The `alt_tse_mac_set_common_speed()` sets the speed of a multiport MAC and the PHYs connected to it. |
| **Parameter:** | `pmac`—A pointer to the base of the MAC control interface.<br>`common_speed`—The speed to set. |
| **Return:** | `TSE_PHY_SPEED_1000` if the PHYs common speed is 1000 Mbps.<br>`TSE_PHY_SPEED_100` if the PHYs common speed is 100 Mbps.<br>`TSE_PHY_SPEED_10` if the PHYs common speed is 10 Mbps.<br>`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()` |

## 10.6.3. alt_tse_phy_add_profile()

| | Details |
|---|---|
| **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. |

## 10.6.4. alt_tse_system_add_sys()

| | Details |
|---|---|
| **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><system.h><altera_avalon_tse_system_info.h> <altera_avalon_tse.h><altera_avalon_tse_system_info.h> <altera_avalon_tse_system_info.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. |

*continued...*

| | Details |
|---|---|
| | psys_sgdma—A pointer to the scatter-gather DMA structure.<br>psys_mem—A pointer to the memory structure.<br>psys_shared_fifo—A pointer to the FIFO structure.<br>psys_phy—A pointer to the PHY structure. |
| **Return:** | SUCCESS if the operation is successful.SUCCESS if the operation is successful.<br>ALTERA_TSE_MALLOC_FAILED if the operation fails.<br>ALTERA_TSE_SYSTEM_DEF_ERROR if one or more of the definitions are incorrect, or empty. |

## 10.6.5. triple_speed_ethernet_init()

| | Details |
|---|---|
| **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 initializes the Triple-Speed Ethernet driver. Initialization involves the following operations:<br>• Set up the NET structure of the MAC device instance.<br>• Configure the MAC PHY Address.<br>• Register and open the SGDMA RX and TX Module of the MAC device instance.<br>• Enable the SGDMA RX interrupt and register it to the Operating System.<br>• Register the SGDMA RX callback function.<br>• Obtains the PHY Speed of the MAC.<br>• Set up the Ethernet MAC Register settings for the Triple-Speed Ethernet driver operation.<br>• 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() |

## 10.6.6. tse_mac_close()

| | Details |
|---|---|
| **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: |

*continued...*

| | Details |
|---|---|
| | • Configure the admin and operation status of the NET structure of the Triple-Speed Ethernet driver instance to ALTERA_TSE_ADMIN_STATUS_DOWN.<br>• De-register the SGDMA RX interrupt from the operating system.<br>• 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.<br>An error code if de-registration of SGDMA RX from the operating system failed. |
| **See also:** | triple_speed_ethernet_init() |

## 10.6.7. tse_mac_raw_send()

| | Details |
|---|---|
| **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.<br>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.<br>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.<br>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.<br>SEND_DROPPED if the number of data bytes is less than the Ethernet header size.<br>ENP_RESOURCE if the SGDMA TX engine is busy. |

## 10.6.8. tse_mac_setGMII mode()

| | Details |
|---|---|
| **Prototype:** | int tse_mac_setGMIImode(np_tse_mac *pmac) |
| **Thread-safe:** | No |
| **Available from ISR:** | No |
| **Include:** | <triple_speed_ethernet_iniche.h> |

*continued...*

| | Details |
|---|---|
| **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()` |

### 10.6.9. tse_mac_setMIImode()

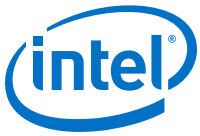
| | Details |
|---|---|
| **Prototype:** | `int tse_mac_setMIImode(np_tse_mac *pmac)` |
| **Thread-safe:** | No |
| **Available from ISR:** | No |
| **Include:** | `<triple_speed_ethernet_iniche.h>` |
| **Description:** | The `tse_mac_setMIImode()` function sets the MAC function operation mode to MII (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()` |

### 10.6.10. tse_mac_SwReset()

| | Details |
|---|---|
| **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 |

## 10.7. Constants

The following 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 114. Constants Mapping

| Constant | Value | Description |
|---|---|---|
| ALTERA_TSE_DUPLEX_MODE_DEFAULT | 1 | 0: Half-duplex<br>1:Full-duplex |
| ALTERA_TSE_MAC_SPEED_DEFAULT | 0 | 0: 10 Mbps<br>1: 100 Mbps<br>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** (Command_Config Register (Dword Offset 0x02) on page 85) | | |
| ALTERA_TSEMAC_CMD_TX_ENA_OFST | 0 | Configures the TX_ENA bit. |
| ALTERA_TSEMAC_CMD_TX_ENA_MSK | 0x1 | |
| ALTERA_TSEMAC_CMD_RX_ENA_OFST | 1 | Configures the RX_ENA bit. |
| ALTERA_TSEMAC_CMD_RX_ENA_MSK | 0x2 | |
| ALTERA_TSEMAC_CMD_XON_GEN_OFST | 2 | Configures the XON_GEN bit. |
| ALTERA_TSEMAC_CMD_XON_GEN_MSK | 0x4 | |
| ALTERA_TSEMAC_CMD_ETH_SPEED_OFST | 3 | Configures the ETH_SPEED bit. |
| ALTERA_TSEMAC_CMD_ETH_SPEED_MSK | 0x8 | |
| ALTERA_TSEMAC_CMD_PROMIS_EN_OFST | 4 | Configures the PROMIS_EN bit. |
| ALTERA_TSEMAC_CMD_PROMIS_EN_MSK | 0x10 | |
| ALTERA_TSEMAC_CMD_PAD_EN_OFST | 5 | Configures the PAD_EN bit. |
| ALTERA_TSEMAC_CMD_PAD_EN_MSK | 0x20 | |
| ALTERA_TSEMAC_CMD_CRC_FWD_OFST | 6 | Configures the CRC_FWD bit. |
| ALTERA_TSEMAC_CMD_CRC_FWD_MSK | 0x40 | |
| ALTERA_TSEMAC_CMD_PAUSE_FWD_OFST | 7 | Configures the PAUSE_FWD bit. |

*continued...*

| Constant | Value | Description |
|---|---|---|
| ALTERA_TSEMAC_CMD_PAUSE_FWD_MSK | 0x80 | |
| ALTERA_TSEMAC_CMD_PAUSE_IGNORE_OFST | 8 | Configures the PAUSE_IGNORE bit. |
| ALTERA_TSEMAC_CMD_PAUSE_IGNORE_MSK | 0x100 | |
| ALTERA_TSEMAC_CMD_TX_ADDR_INS_OFST | 9 | Configures the TX_ADDR_INS bit. |
| ALTERA_TSEMAC_CMD_TX_ADDR_INS_MSK | 0x200 | |
| ALTERA_TSEMAC_CMD_HD_ENA_OFST | 10 | Configures the HD_ENA bit. |
| ALTERA_TSEMAC_CMD_HD_ENA_MSK | 0x400 | |
| ALTERA_TSEMAC_CMD_EXCESS_COL_OFST | 11 | Configures the EXCESS_COL bit. |
| ALTERA_TSEMAC_CMD_EXCESS_COL_MSK | 0x800 | |
| ALTERA_TSEMAC_CMD_LATE_COL_OFST | 12 | Configures the LATE_COL bit. |
| ALTERA_TSEMAC_CMD_LATE_COL_MSK | 0x1000 | |
| ALTERA_TSEMAC_CMD_SW_RESET_OFST | 13 | Configures the SW_RESET bit. |
| ALTERA_TSEMAC_CMD_SW_RESET_MSK | 0x2000 | |
| ALTERA_TSEMAC_CMD_MHASH_SEL_OFST | 14 | Configures the MHASH_SEL bit. |
| ALTERA_TSEMAC_CMD_MHASH_SEL_MSK | 0x4000 | |
| 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 MAGIC_ENA bit. |
| ALTERA_TSEMAC_CMD_MAGIC_ENA_MSK | 0x80000 | |
| ALTERA_TSEMAC_CMD_SLEEP_OFST | 20 | Configures the SLEEP bit. |

*continued...*

| Constant | Value | Description |
|---|---|---|
| ALTERA_TSEMAC_CMD_SLEEP_MSK | 0x100000 | |
| ALTERA_TSEMAC_CMD_WAKEUP_OFST | 21 | Configures the WAKEUP bit. |
| ALTERA_TSEMAC_CMD_WAKEUP_MSK | 0x200000 | |
| ALTERA_TSEMAC_CMD_XOFF_GEN_OFST | 22 | Configures the XOFF_GEN bit. |
| ALTERA_TSEMAC_CMD_XOFF_GEN_MSK | 0x400000 | |
| ALTERA_TSEMAC_CMD_CNTL_FRM_ENA_OFST | 23 | Configures the CNTL_FRM_ENA bit. |
| ALTERA_TSEMAC_CMD_CNTL_FRM_ENA_MSK | 0x800000 | |
| ALTERA_TSEMAC_CMD_NO_LENGTH_CHECK_OFST | 24 | Configures the NO_LENGTH_CHECK bit. |
| ALTERA_TSEMAC_CMD_NO_LENGTH_CHECK_MSK | 0x1000000 | |
| ALTERA_TSEMAC_CMD_ENA_10_OFST | 25 | Configures the ENA_10 bit. |
| ALTERA_TSEMAC_CMD_ENA_10_MSK | 0x2000000 | |
| ALTERA_TSEMAC_CMD_RX_ERR_DISC_OFST | 26 | Configures the RX_ERR_DISC bit. |
| ALTERA_TSEMAC_CMD_RX_ERR_DISC_MSK | 0x4000000 | |
| ALTERA_TSEMAC_CMD_CNT_RESET_OFST | 31 | Configures the CNT_RESET bit. |
| ALTERA_TSEMAC_CMD_CNT_RESET_MSK | 0x80000000 | |
| **Tx_Cmd_Stat Register** (Transmit and Receive Command Registers (Dword Offset 0x3A – 0x3B) on page 91) | | |
| 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 | |

***continued...***

| Constant | Value | Description |
|---|---|---|
| **Rx_Cmd_Stat Register** (Transmit and Receive Command Registers (Dword Offset 0x3A – 0x3B) on page 91) | | |
| ALTERA_TSEMAC_RX_CMD_STAT_RXSHIFT16_OFST | 25 | Configures the RX_SHIFT16 bit |
| ALTERA_TSEMAC_RX_CMD_STAT_RXSHIFT16_MSK | 0x2000000 | |

# 11. Triple-Speed Ethernet Intel FPGA IP User Guide Archives

IP versions are the same as the Intel Quartus Prime Design Suite software versions up to v19.1. From Intel Quartus Prime Design Suite software version 19.2 or later, IP cores have a new IP versioning scheme.

If an IP core version is not listed, the user guide for the previous IP core version applies.

| IP Core Version | User Guide |
|---|---|
| 19.3.0 | Triple-Speed Ethernet Intel FPGA IP User Guide |
| 19.2.0 | Triple-Speed Ethernet Intel FPGA IP User Guide |
| 17.1 | Triple-Speed Ethernet Intel FPGA IP User Guide |
| 16.0 | Triple-Speed Ethernet MegaCore Function User Guide |
| 15.1 | Triple-Speed Ethernet MegaCore Function User Guide |
| 15.0 | Triple-Speed Ethernet MegaCore Function User Guide |

# 12. Document Revision History for the Triple-Speed Ethernet Intel FPGA IP User Guide

| Document Version | Intel Quartus Prime Version | IP Version | Changes |
|---|---|---|---|
| 2020.02.27 | 19.4 | 19.4.0 | • Updated the *Payload Pad Removal* topic. |
| 2019.12.16 | 19.4 | 19.4.0 | • Added support for Intel Agilex devices.<br>• Added new section—*Clocking Scheme of MAC with 2XTBI PCS and Embedded PMA*.<br>• Added a foot note to the Minimum Speed Grade with 1588 Feature value of the Intel Stratix 10 (E-tile) in Table: *Device Family Support for Triple-Speed Ethernet MAC*.<br>• Updated the *Features* topic.<br>• Updated the *Performance and Resource Utilization* topic:<br>— Added new Table: *Resource Utilization for Triple-Speed Ethernet for Intel Agilex Devices*.<br>— Update Table: *Resource Utilization for Triple-Speed Ethernet in Intel Stratix 10 Devices*.<br>— Updated Table: *Resource Utilization for Triple-Speed Ethernet with E-Tile Transceiver in Intel Stratix 10 Devices*.<br>— Updated Table: *Resource Utilization for Triple-Speed Ethernet for Intel Arria 10 Devices*.<br>— Updated Table: *Resource Utilization for Triple-Speed Ethernet for Intel Cyclone 10 GX Devices*.<br>• Remove the note in the *Generating a Design Example or Simulation Model* topic.<br>• Updated the description of the **Transceiver type** parameter in Table: *Core Configuration Parameters*.<br>• Updated the parameter name **Enable Stratix 10 transceiver dynamic reconfiguration** to **Enable E-tile transceiver dynamic reconfiguration** in Table: *PCS/Transceiver Options Parameters*.<br>• Added a new Topic—*Intel LVDS Transmitter and Receiver Soft-CDR I/O Signals*.<br>• Updated the description in the *Receive FIFO Buffer and Local Device Congestion* topic.<br>• Updated Table: *PCS Transmit and Receive Latency* to include latency information for the following **Intel Stratix 10 E-tile** configuration:<br>— 10-Mbps SGMII 2XTBI PCS<br>— 100-Mbps SGMII 2XTBI PCS<br>— 10-Mbps SGMII 2XTBI PCS with GXB<br>— 100-Mbps SGMII 2XTBI PCS with GXB<br>• Updated the description in the *1000BASE-X/SGMII PCS With Optional Embedded PMA* topic.<br>• Updated the *FPGA IEEE 1588v2 Feature* topic.<br>• Updated the *GMII Clock Signals* topic. |

***continued...***

| Document Version | Intel Quartus Prime Version | IP Version | Changes |
|---|---|---|---|
| | | | • Updated the following figures:<br>— Updated figure and figure title *10/100/1000-Mbps Ethernet MAC and 1000BASE-X/SGMII PCS with Optional PMA* to *10/100/1000-Mbps Ethernet MAC and 1000BASE-X/SGMII PCS with PMA*.<br>— Updated figure and figure title *1000BASE-X/SGMII 2XTBI PCS with Optional PMA* to *1000BASE-X/SGMII 2XTBI PCS*.<br>— *Clock Distribution in MAC and SGMII PCS with GXB Configuration—Optimal Case*<br>— *Clock Distribution in MAC and SGMII PCS with LVDS Configuration—Optimal Case*.<br>— *Power-Down*.<br>— Corrected figure title *PTP Frame in IEEE 8002.3* to *PTP Frame in IEEE 802.3*<br>— Updated Figure: *10/100/1000 Ethernet MAC Function with Internal FIFO Buffers Signals*.<br>— Updated Figure: *10/100/1000 Multiport Ethernet MAC Function without Internal FIFO Buffers Signals*.<br>— Updated Figure: *10/100/1000 Ethernet MAC Function with Internal FIFO Buffers, with 1000BASE-X/SGMII PCS Signals*.<br>— Updated Figure: *0/100/1000 Ethernet MAC Function with Internal FIFO Buffers, and 1000BASE-X/SGMII 2XTBI PCS Signals With Embedded PMA Signals*.<br>— Updated Figure: *10/100/1000 Multiport Ethernet MAC Function without Internal FIFO Buffers with 1000BASE-X/SGMII PCS Signals*.<br>— Updated Figure: *10/100/1000 Ethernet MAC Function with Internal FIFO Buffers, and 1000BASE-X/SGMII PCS With Embedded PMA Signals*.<br>— Updated Figure: *10/100/1000 Multiport Ethernet MAC Function without Internal FIFO Buffers, with IEEE 1588v2, 1000BASE-X/SGMII PCS and Embedded PMA Signals*.<br>— Updated Figure: *1000BASE-X/SGMII PCS Function Signals*.<br>— Updated Figure: *1000BASE-X/SGMII 2XTBI PCS Function Signals*.<br>— Updated Figure: *1000BASE-X/SGMII PCS Function and PMA Signals*.<br>— Updated Figure: *Reset Distribution in MAC with PCS and Embedded PMA*.<br>• Updated the Table: *GMII/RGMII/MII Clock Signals* in the *10/100/1000 Ethernet MAC Signals* section to remove `tx_clkena` and `rx_clkena` signals.<br>• Added a new topic for the for the *10/100/1000 Ethernet MAC Signals* section—*Clock Enabler Signals*.<br>• Updated Table: *Clock Signals Visible at Top-Level Design*:<br>— Updated the table description.<br>— Added footnotes for `ff_tx_clk` and `ff_rx_clk`. |

***continued...***

| Document Version | Intel Quartus Prime Version | IP Version | Changes |
|---|---|---|---|
| | | | • Updated the note in the *Creating Clock Constraints* topic to state that the `derive_pll_clocks` command is not supported in Intel Stratix 10 devices because all PLL clocks are automatically generated by the SDC files generated alongside the PLL IP.<br>• Corrected the term `gbx_pwrdn_in` to `gxb_pwrdn_in`.<br>• Updated the term "MegaWizard Plug-In Manager" to "IP Catalog".<br>• Renamed topic title *Intel Stratix 10 E-tile Transceiver Native PHY Signals* to *E-tile Transceiver Native PHY Signals*.<br>• Updated for latest Intel branding standards. |
| 2019.11.01 | 19.3 | 19.3.0 | • Corrected the minimum speed grade with 1588 feature for Intel Stratix 10 E-tile devices from "-I3" to "Not supported".<br>• Updated the descriptions in the following topics:<br>— *Features*<br>— *Intel FPGA IP IEEE 1588v2 Feature*<br>• Updated Table: *IEEE 1588v2 Feature LVDS I/O Delay—Hardware*. |
| 2019.09.30 | 19.3 | 19.3.0 | • Updated Table: *Device Family Support for Triple-Speed Ethernet MAC*.<br>• Updated Table: *Simulation Model Files*.<br>• Updated the notes for Figure: *Clock Distribution in MAC and 1000BASE-X PCS with LVDS Configuration —Optimal Case*. |
| 2019.07.24 | 19.2 | 19.2.0 | • Added support for two new core variants for Intel Stratix 10 E-tile devices:<br>— 10/100/1000-Mbps Ethernet MAC with 1000BASE-X/SGMII 2XTBI PCS<br>— 1000BASE-X/SGMII 2XTBI PCS<br>• Added a new Topic: *GMII Converter*.<br>• Added a new Table: *Resource Utilization for Triple-Speed Ethernet with E-tile Transceiver in Intel Stratix 10 Devices*.<br>• Updated Table: *PCS/Transceiver Options Parameters*:<br>— Added **Intel Stratix 10 GXB Transceiver Options** parameter to the table.<br>— Added a second note to clarify that the **Transceiver Options** and **Series V GXB Transceiver Options** parameters are not available in the Triple-Speed Ethernet Intel FPGA IP parameter editor interface of the Intel Quartus Prime Pro Edition software version 19.2 onwards. These options are only present in the Intel Quartus Prime Standard Edition software.<br>• Updated Table: *PCS Transmit and Receive Latency* to include latency information for **Intel Stratix 10 E-tile** configuration.<br>• Added a note to the *MAC Configuration Register Space* topic.<br>• Updated Table: *PCS Control Register Bit Descriptions* to update the descriptions for `COLLISION_TEST` and `DUPLEX_MODE`. |

*continued...*

| Document Version | Intel Quartus Prime Version | IP Version | Changes |
|---|---|---|---|
| | | | • Updated `c0` clock frequency from 110 MHz to 100 MHz and added a note of recommended clock frequencies for different FIFO widths for `ff_tx_clk` and `ff_rx_clk` signals in Figure: *Triple-Speed Ethernet Timing Constraint Example*.<br>• Updated recommended clock frequencies for `ff_tx_clk` and `ff_rx_clk` signals in Table: *Recommended Clock Input Frequency For Each IP Core Variant*.<br>• Updated Table: *Statistics Counters* to change the `ifOutDiscards` register to Reserved.<br>• Restructured the document. |
| 2019.03.29 | 17.1 | 17.1 | Updated the note in the *MAC Error Correction Code (ECC)* topic to state that the error correction code (ECC) feature is applicable to Arria V GZ, Stratix V, and Intel Arria 10 devices. |
| 2019.02.21 | 17.1 | 17.1 | Updated the project directory path for VHDL design in the *Simulate the IP Core* topic. |
| 2019.01.28 | 17.1 | 17.1 | Added notes to the *Multicast Address Resolution* topic. |
| 2018.11.28 | 17.1 | 17.1 | Updated Table: *Clock Signals Visible at Top-Level Design* to add notes for `ref_clk` under MAC Only and MAC+PCS configurations. |
| 2018.08.01 | 17.1 | 17.1 | • Renamed the document as *Triple-Speed Ethernet Intel FPGA IP Core User Guide*.<br>• Updated the description of the *MAC Transmit Datapath* topic.<br>• Updated the "Command_Config Register Field Descriptions" and "PCS Configuration Registers" tables: Added HW reset values to all registers.<br>• Updated Table: *Transmit and Receive Nominal Latency*.<br>• Updated Table: *Statistics Counters* to update the descriptions for `aOctetsReceivedOK`, `ifOutErrors`, `ifOutUcastPkts`, `ifOutMulticastPkts`, and `ifOutBroadcastPkts` registers.<br>• Updated for latest Intel branding standards. |

| Date | Version | Changes |
|---|---|---|
| November 2017 | 2017.11.06 | • Rebranded as Intel.<br>• Renamed the document as *Intel FPGA Triple-Speed Ethernet IP Core User Guide*.<br>• Added support for the Intel Stratix 10, Intel Cyclone 10 GX, and and Intel Cyclone 10 LP device families.<br>• Updated the description of the *About This IP Core* topic.<br>• Added "Intel FPGA IP Core Device Support Levels" table to the *Device Family Support* topic.<br>• Removed the *Definition: Device Support Level* topic.<br>• Updated the "Intel Arria 10 Resource Utilization", "Cyclone V Resource Utilization" table: Updated the IP core name from 1000BASE-X/SGMII PCS with PMA to 1000BASE-X/SGMII PCS. |

***continued...***

| Date | Version | Changes |
|---|---|---|
| | | • Updated the *Generating a Design Example or Simulation Model* topic:<br>— Added a note to clarify that the **Generate Example Design** option only generates the design for functional simulation.<br>— Added a note to clarify that the dynamically generated design example for functional simulation is available only in Intel Arria 10, Intel Cyclone GX, and Intel Stratix 10 devices.<br>• Updated the "Recommended Quartus Pin Assignments" table: Updated the Design Pin information for `GLOBAL_SIGNAL` pin assignment.<br>• Updated the "Core Configuration Parameters" table:<br>— Added a note to the description of **Interface** parameter to clarify that RGMII interface is not supported in Intel Arria 10, Intel Cyclone 10 GX, and Intel Stratix 10 devices from Intel Quartus Prime software version 17.1 onwards.<br>— Added a note to the description of **Number of ports** to clarify that the number of ports supported for Triple-Speed Ethernet designs targeting Intel Stratix 10, Intel Arria 10, and Intel Cyclone 10 GX devices is 8 in Intel Quartus Prime software version 17.1 onwards.<br>— Added a note to the description of **Transceiver type** parameter to clarify on the performance risk when using Triple-Speed Ethernet IP variant with LVDS I/O for PMA implementation in Intel Arria 10 devices for Intel Quartus Prime software versions 17.0.2 and earlier.<br>• Updated Figure: Hardware Multicast Address Resolution Engine<br>• Updated the "PCS Transmit and Receive Latency" table:<br>— Added PCS transmit and receive latency for Intel Stratix 10 and Intel Cyclone 10 GX devices.<br>— Added a footnote under the Latency (Clock Cycles) column to clarify that the latency numbers are from simulation.<br>• Updated the description in the *CRC Checking* topic.<br>• Updated the *Configuration Register Space* section:<br>— Updated the "IEEE 1588v2 Feature PMA Delay—Hardware" table to include digital delay information for Intel Arria 10 devices.<br>— Updated the "IEEE 1588v2 Feature LVDS I/O Delay—Hardware" table to include digital delay information for Intel Arria 10 and Intel Stratix 10 devices.<br>• Updated the description of the *MAC and PCS With LVDS Soft-CDR I/O* topic: Added a note to clarify on the performance risk when using Triple-Speed Ethernet IP variant with LVDS I/O for PMA implementation in Intel Arria 10 devices for Intel Quartus Prime software versions 17.0.2 and earlier.<br>• Added a note to the *Sharing PLLs in Devices with LVDS Soft-CDR I/O* topic.<br>• Updated the *Creating Clock Constraints* topic: Added a note to clarify that the `derive_pll_clocks` command is not supported in Intel Stratix 10 devices.<br>• Made editorial updates throughout the document. |
| March 2017 | 2017.03.08 | • Updated the note below Figure 6-5 in the 10/100/1000 Ethernet MAC with 1000BASE-X/SGMII PCS and Embedded PMA Signals topic.<br>• Updated the Arria 10 and Cyclone V Resource Utilization tables to include information about 10/100/1000-Mbps Ethernet MAC and 1000BASE-X/SGMII PCS MegaCore Function.<br>• Updated the link in the Related Information section for the Altera IEEE 1588v2 Features topic.<br>• Editorial fix to the notes in Figures 6-5, 6-6, 6-7, and 7-2. |
| January 2017 | 2017.01.05 | • Added ordering code IP-TRIETHERNETF for IEEE 1588v2 and product ID(s) 00BD and 0104 for Triple-Speed Ethernet and IEEE 1588v2. |
| January 2017 | 2017.01.05 | Corrected typo in the Configuration Register Space topic. |

*continued...*

| Date | Version | Changes |
|---|---|---|
| October 2016 | 2016.10.31 | • Corrected the Device Family Support topic to include all supported devices, including devices that do not have the 1588 feature support.<br>• Removed mention of `read_timeout` in the topic about MAC reset.<br>• Updated the description of `DISABLE_READ_TIMEOUT` in the topic about the command_config register.<br>• Removed `read_timeout` and `disable_read_timeout` registers from the table that lists the PCS configuration registers. |
| May 2016 | 2016.05.02 | • Updated the Device Family Support topic.<br>• Updated the Performance and Resource Utilization topic.<br>• Updated the Release Information topic.<br>• Removed the Design Example topic and the appendices that described the design components, Time-of-Day (ToD) Clock, ToD Synchronizer, and Packet Classifier. Added a link to the application note for the design example.<br>• Added the Document Archives topic that lists documents for the past releases.<br>• Removed the PMA and LVDS I/O Delay—Simulation Model tables from the IEEE 1588v2 Feature PMA Delay topic because simulation data is not deterministic. |
| November 2015 | 2015.11.02 | • ToD Clock chapter:<br>— Updated the device family support.<br>— Added a new parameter—**PERIOD_CLOCK_FREQUENCY**.<br>— Updated the CSR description for `SecondsH`, `SecondsL`, `NanoSec`, `Period`, `AdjustPeriod`, `DriftAdjust`, and `DriftAdjustRate`.<br>• ToD Synchronizer chapter:<br>— Updated the device family support.<br>— Changed the frequency range to 390.625 MHz (from 312.5 MHz)<br>— Added a new table—"Sampling Clock Frequency According to the Selected Parameter Settings".<br>— Updated the "Settings to Achieve the Recommended Factors for Stratix V PLL" table with more sampling clock factors.<br>— Updated the parameter value of `SYNC_MODE` to "Between 0 to 15" (from "Between 0 to 6").<br>— Added a new parameter—**SAMPLE_SIZE**.<br>• Updated the description for `tx_serial_clk` to state that the clock frequency is 1250 MHz.<br>• Changed instances of *Quartus II* to *Quartus Prime*. |
| June 2015 | 2015.06.15 | • Added a new parameter, , in the Core Configuration Parameters table.<br>• Added description for new signals—`tx_clkena`, `rx_clkena`, and `led_panel_link`.<br>• Added Qsys-equivalent signal names for the following signals:Use clock enable for MAC<br>— control_port_clock_connection: clk<br>— pcs_mac_tx_clock_connection: tx_clk<br>— pcs_mac_rx_clock_connection: rx_clk<br>— receive_clock_connection: ff_rx_clk<br>— transmit_clock_connection: ff_tx_clk<br>• Revised the `Command_config` register field descriptions for bits 0, 1, and 13.<br>• Corrected the `Command_config` register setting for Enable MAC Transmit and Receive Datapath register initialization sequence from 0x00802223 to 0x00800223.<br>• Corrected the bit width for `pkt_class_data`Use clock enable signal in the following timing diagrams:<br>— Receive Operation—MAC Without Internal FIFO Buffers.<br>— Invalid Length Error During Receive Operation—MAC Without Internal FIFO Buffers. |

***continued...***

| Date | Version | Changes |
|---|---|---|
| | | • Updated the following sections to indicate that the reconfiguration signals are not present in variations targeting Arria 10, Stratix V, Arria V, and Cyclone V devices with GX transceivers.<br>— note in Figure 51 on page 126<br>— SERDES control signals description in Table 83 on page 127.<br>— note in Figure 52 on page 129<br>— Sharing Transceiver Quads on page 157<br>• Updated the description for **Extended Statistics Counters (0x3C – 0x3E)** to state the specific order for reading counters.<br>• Removed "10/100/1000-Mbps MAC with 1000BASE-X/SGMII PCS" configuration from the list of supported configurations in IEEE 1588v2 feature.<br>• Added a new topic—Using ToD Clock SecondsH, SecondsL, and NanoSecRegisters. |
| June 2014 | 14.0 | • Added a link to the Altera website that provides the latest device support information for Altera IP.<br>• Added a note in PCS/Transceiver Options on page 29—You must configure the Arria 10 Transceiver ATX PLL output clock frequency to 1250.0 MHz when using the Arria 10 Transceiver Native PHY with the Triple-Speed Ethernet IP core.<br>• Added MAC Error Correction Code (ECC) on page 52 section.<br>• Added new support configuration for IEEE 1588v2 feature.<br>• Updated the `tx_period` and `rx_period` register bits in IEEE 1588v2 Feature (Dword Offset 0xD0 – 0xD6) on page 92.<br>• Updated the timing adjustment for the IEEE 1588v2 feature PMA delay in IEEE 1588v2 Feature PMA Delay on page 93.<br>• Revised the control interface signal names to `reg_rd`, `reg_data_in`, `reg_wr`, `reg_busy`, and `reg_addr` in MAC Control Interface Signals on page 108.<br>• Added ECC status signals in ECC Status Signals on page 114 and ECC Status Signals on page 121.<br>• Added Arria 10 Transceiver Native PHY signals in Intel Arria 10 Transceiver Native PHY Signals on page 120.<br>• Added Transceiver Native PHY signal in Transceiver Native PHY Signal on page 127.<br>• Updated the following the signal diagrams:<br>— 10/100/1000 Ethernet MAC Signals<br>— 1000BASE-X/SGMII PCS Function Signals<br>— 10/100/1000 Ethernet MAC with 1000BASE-X/SGMII PCS Signals<br>— 10/100/1000 Multiport Ethernet MAC Function without Internal FIFO Buffers, with IEEE 1588v2, 1000BASE-X/SGMII PCS and Embedded PMA Signals<br>• Added IEEE 1588v2 feature PHY path delay interface signals in IEEE 1588v2 PHY Path Delay Interface Signals on page 133.<br>• Updated the `Period` and `AdjustPeriod` register bits in ToD Clock Configuration Register Space.<br>• Added two new conditions that the ToD synchronizer module supports in ToD Synchronizer chapter.<br>• Added three new recommended sampling clock frequencies in ToD Synchronizer chapter.<br>• Added a new setting of 32/63 in ToD Synchronizer Block.<br>• Updated the SYNC_MODE parameter value and description in ToD Synchronizer Parameter Settings. |
| December 2013 | 13.1 | • Added support for Arria 10 device.<br>• Added device family support list for IEEE 1588v2 variant.<br>• Updated the PCS/Transceiver options parameters in PCS/Transceiver Options on page 29. |

*continued...*

| Date | Version | Changes |
|---|---|---|
| | | • Updated the bit order in Table 49 on page 98 , Table 50 on page 99 and Table 52 on page 100.<br>• Added information on how to view all the signal names when implementing the IP in Qsys in Interface Signals.<br>• Added a section about exposed ports in the new user interface in Design Considerations. |
| May 2013 | 13.0 | • Updated the MegaWizard Plug-In Manager flow in Getting Started with Altera IP Cores.<br>• Added information about generating a design example and simulation testbench in Generating a Design Example or Simulation Model on page 22.<br>• Updated the list of Quartus II generated files.<br>• Added information about the recommended pin assignments in Design Constraint File No Longer Generated on page 24.<br>• Updated the MegaCore parameter names and description in Parameter Settings.<br>• Updated the IEEE 1588v2 feature list in Functional Description.<br>• Updated the SGMII auto-negotiation description in Functional Description.<br>• Added information about the IEEE 1588v2 feature PMA delay in IEEE 1588v2 Feature PMA Delay on page 93.<br>• Updated the Multiport Ethernet MAC with IEEE 1588v2, 1000BASE-X/SGMII PCS and Embedded PMA Signals.<br>• Updated the IEEE 1588v2 timestamp signal names.<br>• Added timing diagrams for IEEE 1588v2 timestamp signals.<br>• Added a section about migrating existing design to the Quartus II software new MegaCore user interface in Design Considerations.<br>• Updated Timing Constraints chapter, to describe the new timing constraint files and the recommended clock input frequency for each MegaCore Function variant.<br>• Added information about the simulation model files generated using IEEE simulation encryption in Simulation Model Files on page 171.<br>• Updated the jumbo frames file directory in Using Jumbo Frames on page 175.<br>• Updated the ToD configuration parameters in ToD Clock Parameter Setting and ToD interface signals, ToD Clock Avalon-ST Transmit Interface Signals and ToD Clock Avalon-MM Control Interface Signals.<br>• Added information to describe the ToD's drift adjustment in the Adjusting ToD Clock Drift.<br>• Added ToD Synchronizer and Packet Classifier chapters.<br>• Removed SOPC Builder information. |
| January 2013 | 12.1 | • Added Altera IEEE 1588v2 Feature section in Chapter 4.<br>• Added information for the following GUI parameters: Enable timestamping, Enable PTP 1-step clock, and Timestamp fingerprint width in "Timestamp Options".<br>• Added MAC registers with IEEE 1588v2 feature.<br>• Added IEEE 1588v2 feature signals tables.<br>• Added Triple-Speed Ethernet with IEEE 1588v2 Design Example section.<br>• Added Time-of-Day Clock section. |
| June 2012 | 12.0 | • Added support for Cyclone V.<br>• Updated the Congestion and Flow Control section in Chapter 4.<br>• Added Register Initialization section in Chapter 5.<br>• Added `holdoff_quant` register description.<br>• Added `UNIDIRECTIONAL_ENABLE` bit description.<br>• Revised and moved the section on Timing Constraint to a new chapter.<br>• Added information about how to customize the SDC file in Chapter 8.<br>• Added Pause Frame Generation section. |

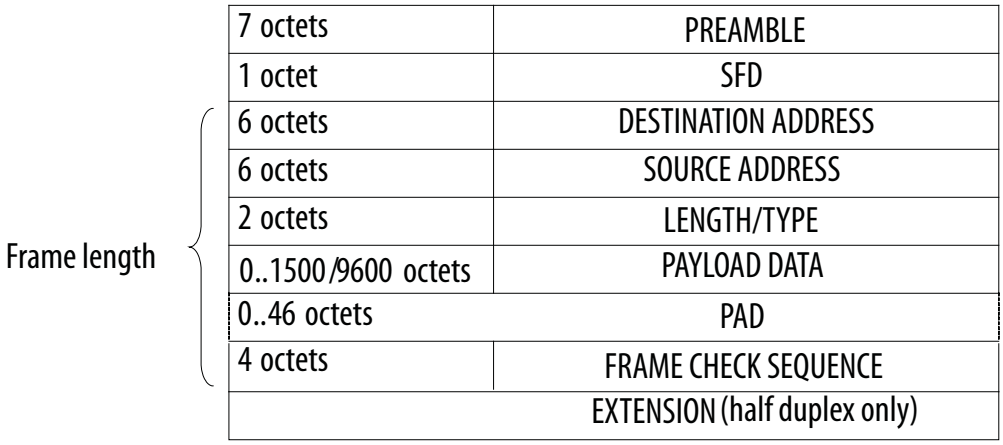
***continued...***

| Date | Version | Changes |
|---|---|---|
| November 2011 | 11.1 | • Added support for Arria V.<br>• Revised the Device Family Support section in Chapter 1.<br>• Added `disable_read_timeout` and `read_timeout` registers at address 0x15 and 0x16. |
| June 2011 | 11.0 | • Updated support for Cyclone IV GX, Cyclone III LS, Aria II GZ, HardCopy IV GX/E and HardCopy III E devices.<br>• Revised Performance and Resource Utilization section in Chapter 1.<br>• Updated Chapter 3 to include Qsys System Integration Tool Design Flow.<br>• Added Transmit and Receive Latencies section in Chapter 4.<br>• Updated all MAC register address to dbyte addressing. |
| December 2010 | 10.1 | • Added support for Arria II GZ.<br>• Added a new parameter, Starting Channel Number.<br>• Streamlined the contents and document organization. |
| August 2010 | 10.0 | • Added support for Stratix V.<br>• Revised the nomenclature of device support types.<br>• Added chapter 5, Design Considerations. Moved the Clock Distribution section to this chapter and renamed it to Optimizing Clock Resources in Multiport MAC and PCS with Embedded PMA. Added sections on PLL Sharing and Transceiver Quad Sharing.<br>• Updated the description of Enable transceiver dynamic reconfiguration. |
| November 2009 | 9.1 | • Added support for Cyclone IV, Hardcopy III, and Hardcopy IV, and updated support for Hardcopy II to full.<br>• Updated chapter 1 to include a feature comparison between 10/100/1000 Ethernet MAC and small MAC.<br>• Updated chapter 4 to revise the 10/100/1000 Ethernet MAC description, Length checking, Reset, and Control Interface sections. |
| March 2009 | 9.0 | • Added support for Arria II GX.<br>• Updated chapter 3 to include a new parameter that enables wider statistics counters.<br>• Updated chapter 4 to reflect support for different speed in multiport MACs and gated clocks elimination.<br>• 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.<br>• Updated Chapter 6 to include a procedure to add unsupported PHYs. |
| May 2008 | 8.0 | • Revised the performance tables and device support.<br>• Updated Chapters 3 and 4 to include information on MAC with multi ports and without internal FIFOs.<br>• Revised the clock distribution section in Chapter 4.<br>• Reorganized Chapter 5 to remove redundant information and to include the new testbench architecture.<br>• Updated Chapter 6 to include new public APIs. |
| October 2007 | 7.2 | • Updated Chapter 1 to reflect new device support.<br>• Updated Chapters 3 and 4 to include information on Small MAC. |
| May 2007 | 7.1 | • Added Chapters 2, 3, 5 and 6.<br>• 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.<br>• Initial release of document on Web. |
| December 2006 | 6.1 | Initial release of document on DVD. |

# A. Ethernet Frame Format

## A.1. Basic Frame Format

**Figure 82. MAC Frame Format**




A basic Ethernet frame comprises the following fields:

- Preamble—a maximum of 7-octet fixed value of 0x55.
- Start frame delimiter (SFD)—a 1-octet fixed value of 0xD5 which marks the beginning of a frame.
- Destination and source addresses—6 octets each. The least significant byte is transmitted first.
- Length or type—a 2-octet 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)—a 4-octet cyclic redundancy check (CRC) value for detecting frame errors during transmission.
- An extension field—Required only for gigabit Ethernet operating in half-duplex mode. The MAC function does not support this implementation.

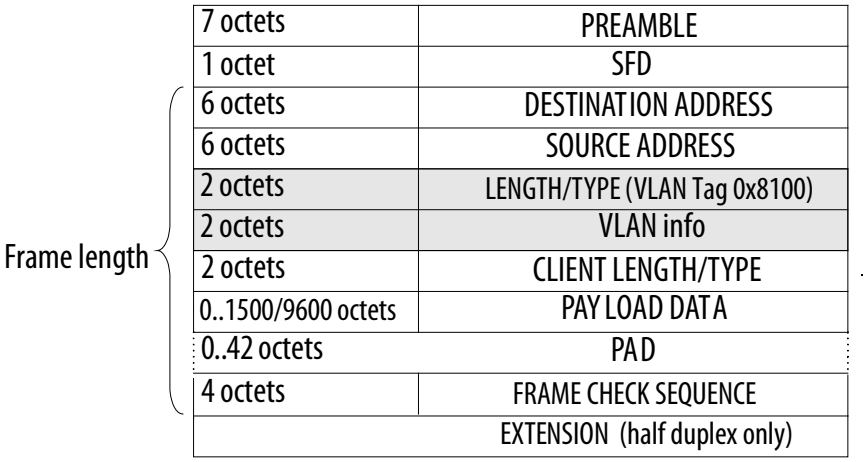
## A.2. VLAN and Stacked VLAN Frame Format

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 source address and length/type fields. VLAN tagging is defined by the IEEE Standard 802.1Q. VLAN tagging can identify and separate many groups' network

traffic from each other in enterprise and metro networks. Each VLAN group can consist of many users with varied MAC address in different geographical locations of a network. VLAN tagging increases and scales the network performance and add privacy and safety to various groups and customers' network traffic.

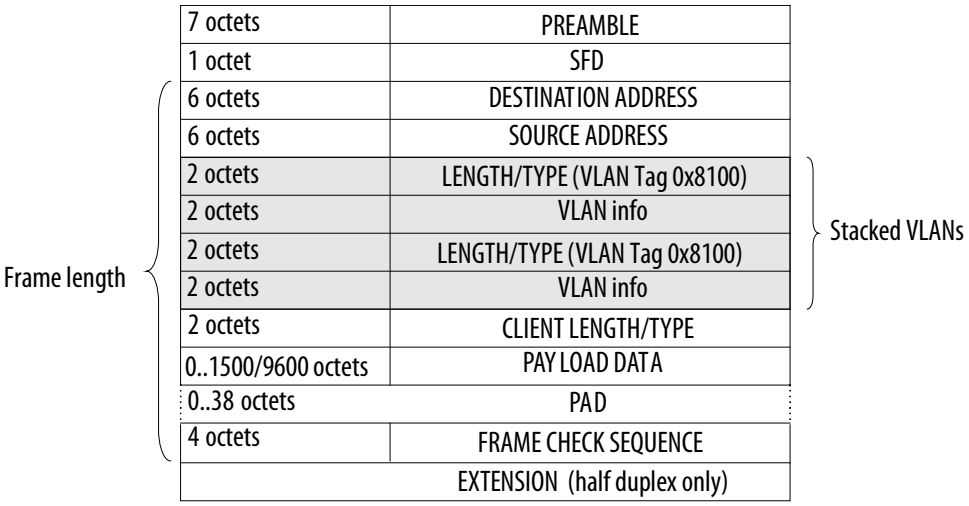
VLAN tagged frames have a maximum length of 1522 bytes, excluding the preamble and the SFD fields.

**Figure 83. 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 source address and client length/type fields, as illustrated.

**Figure 84. Stacked VLAN Tagged MAC Frame Format**

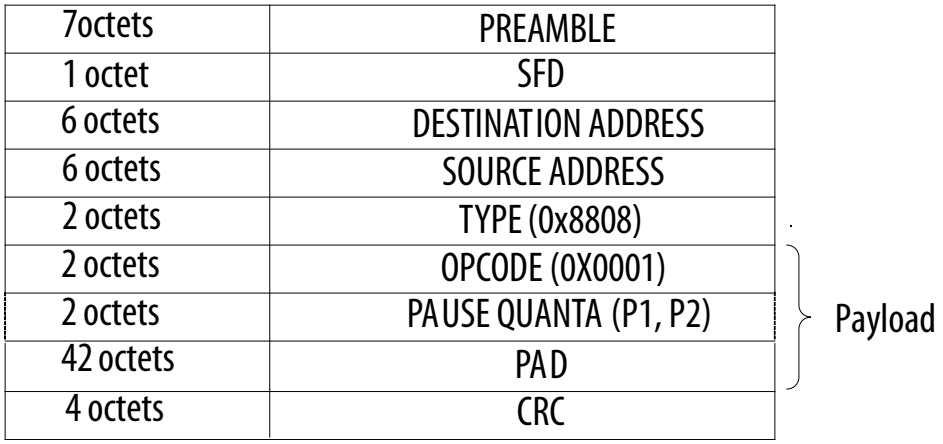


## A.3. Pause Frame Format

A pause frame is generated by the receiving device to indicate congestion to the emitting device. If flow control is supported, the emitting device should stop sending data upon receiving pause frames.

The length/type field has a fixed value of 0x8808, followed by a 2-octet opcode field of 0x0001. A 2-octet pause quanta is defined in the second and third bytes of the frame payload (P1 and P2). The pause quanta, P1, is the most significant byte. A pause frame has no payload length field, and is always padded with 42 bytes of 0x00.

**Figure 85. Pause Frame Format**




### A.3.1. Pause Frame Generation

When you turn on the **Enable full-duplex flow control** option, pause frame generation is triggered by the following events:

- RX FIFO fill level hits the `rx_section_empty` threshold.
- XOFF register write.
- XON register write.
- XOFF I/O pin (`xoff_gen`) assertion.
- XON I/O pin (`xon_gen`) assertion.

If the RX FIFO buffer is almost full, the MAC function triggers the pause frame generation to the remote Ethernet device.

If the local Ethernet device needs to generate pause frame via XOFF or XON register write or I/O pin assertion, it is recommended to set the `rx_section_empty` register to a larger value to avoid non-deterministic result.

The following table summarizes the pause frame generation based on the above events.

## Table 115. Pause Frame Generation

| Register Write or I/O Pin Assertion (1) | | Description |
|---|---|---|
| XOFF_GEN | XON_GEN | |
| 1 | 0 | If the XOFF_GEN bit is set to 1, the XOFF pause frames are continuously generated and sent to the MII/GMII TX interface until the XOFF_GEN bit is cleared. |
| 0 | 1 | If the XON_GEN bit is set to 1, the XON pause frames are continuously generated and sent to the MII/GMII TX interface until the XON_GEN bit is cleared. |
| 1 | 1 | This event is not recommended as it will produce non-deterministic result. |

Note to Table 115 on page 198 :
1. Set the XON and XOFF registers to 0 when you use the I/O pin to generate the pause frame and vice versa.

# B. Simulation Parameters

## B.1. Functionality Configuration Parameters

You can use these parameters to enable or disable specific functionality in the MAC and PCS.

**Table 116. IP Core Functionality Configuration Parameters**

| Parameter | Description | Default |
|---|---|---|
| **Supported in configurations that contain the 10/100/1000 Ethernet MAC** | | |
| ETH_MODE | 10: Enables MII.<br>100: Enables MII.<br>1000: Enables GMII. | 1000 |
| HD_ENA | Sets the HD_ENA bit in the command_config register. See Command_Config Register (Dword Offset 0x02) on page 85. | 0 |
| TB_MACPAUSEQ | Sets the pause_quant register. See Base Configuration Registers (Dword Offset 0x00 – 0x17) on page 82. | 15 |
| TB_MACIGNORE_PAUSE | Sets the PAUSE_IGNORE bit in the command_config register. See Command_Config Register (Dword Offset 0x02) on page 85. | 0 |
| TB_MACFWD_PAUSE | Sets the PAUSE_FWD bit in the command_config register. See Command_Config Register (Dword Offset 0x02) on page 85. | 0 |
| TB_MACFWD_CRC | Sets the CRC_FWD bit in the command_config register. See Command_Config Register (Dword Offset 0x02) on page 85. | 0 |
| TB_MACINSERT_ADDR | Sets the ADDR_INS bit in the command_config register. See Command_Config Register (Dword Offset 0x02) on page 85. | 0 |
| TB_PROMIS_ENA | Sets the PROMIS_EN bit in the command_config register. See Command_Config Register (Dword Offset 0x02) on page 85. | 1 |
| TB_MACPADEN | Sets the PAD_EN bit in the command_config register. See Command_Config Register (Dword Offset 0x02) on page 85. | 1 |
| TB_MACLENMAX | Maximum frame length. | 1518 |
| TB_IPG_LENGTH | Sets the tx_ipg_length register. See Base Configuration Registers (Dword Offset 0x00 – 0x17) on page 82. | 12 |
| TB_MDIO_ADDR0 | Sets the mdio_addr0 register. See Base Configuration Registers (Dword Offset 0x00 – 0x17) on page 82. | 0 |
| TB_MDIO_ADDR1 | Sets the mdio_addr1 register. See Base Configuration Registers (Dword Offset 0x00 – 0x17) on page 82. | 1 |
| TX_FIFO_AE | Sets the tx_almost_empty register. See Base Configuration Registers (Dword Offset 0x00 – 0x17) on page 82. | 8 |
| TX_FIFO_AF | Sets the tx_almost_full register. See Base Configuration Registers (Dword Offset 0x00 – 0x17) on page 82. | 10 |

*continued...*

| Parameter | Description | Default |
|---|---|---|
| RX_FIFO_AE | Sets the rx_almost_empty register. See Base Configuration Registers (Dword Offset 0x00 – 0x17) on page 82. | 8 |
| RX_FIFO_AF | Sets the rx_almost_full register. See Base Configuration Registers (Dword Offset 0x00 – 0x17) on page 82. | 8 |
| TX_FIFO_SECTION_EMPTY | Sets the tx_section_empty register. See Base Configuration Registers (Dword Offset 0x00 – 0x17) on page 82. | 16 |
| TX_FIFO_SECTION_FULL | Sets the tx_section_full register. See Base Configuration Registers (Dword Offset 0x00 – 0x17) on page 82. | 16 |
| RX_FIFO_SECTION_EMPTY | Sets the rx_section_empty register. See Base Configuration Registers (Dword Offset 0x00 – 0x17) on page 82. | 0 |
| RX_FIFO_SECTION_FULL | Sets the rx_section_full register. See Base Configuration Registers (Dword Offset 0x00 – 0x17) on page 82. | 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<br>0x886644352611<br>0xABCDEF012313<br>0x92456545AB15<br>0x432680010217<br>0xADB589215439<br>0xFFEACFE3434B<br>0xFFCCDDAA3123<br>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 If_Mode Register (Word Offset 0x14) on page 100. | 0 |
| TB_SGMII_AUTO_CONF | Sets the USE_GMII_AN bit in the if_mode register. See If_Mode Register (Word Offset 0x14) on page 100. | 0 |

## B.2. Test Configuration Parameters

You can use these parameters to create custom test scenarios.

### Table 117. Test Configuration Parameters

| 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 Streaming Ethernet frame generator. | 5 |
| TB_RXIPG | IPG on the receive path. | 12 |

*continued...*

| Parameter | Description | Default |
|---|---|---|
| TB_ENA_VAR_IPG | 0: A constant IPG, TB_RXIPG, is used by the GMII/RGMII/MII Ethernet frame generator.<br><br>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.<br><br>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.<br>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 Streaming monitor stops reading from the receive FIFO. | 1000 |
| TB_TX_FF_ERR | 0: Normal behavior.<br><br>1: Drives the Avalon Streaming 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 cycles 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.<br>1: Enables flow control in the GMII frame generator. | 1 |
| TB_MDIO_SIMULATION | Enable / Disable MDIO simulation. | 0 |
| **Supported in configurations that contain the 1000BASE-X/SGMII PCS** | | |
| TB_SGMII_HD | 0: Disables half-duplex mode. | 0 |

***continued...***

| Parameter | Description | Default |
|---|---|---|
| | 1: Enables half-duplex mode. | |
| TB_SGMII_1000 | 0: Disables gigabit operation.<br>1: Enables gigabit operation. | 1 |
| TB_SGMII_100 | 0: Disables 100 Mbps operation.<br>1: Enables 100 Mbps operation. | 0 |
| TB_SGMII_10 | 0: Disables 10 Mbps operation.<br>1: Enables 10 Mbps operation. | 0 |
| TB_TX_ERR | 0: Disables error generation.<br>1: Enables error generation. | 0 |

# Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

Altera:

IP-TRIETHERNETF IPR-TRIETHERNETF