Intel FPGA Low Latency Ethernet 10G MAC User Guide

Updated for Intel® Quartus® Prime Design Suite: **17.1**

Subscribe
Send Feedback

UG-01144 | 2017.12.25
Latest document on the web: PDF | HTML

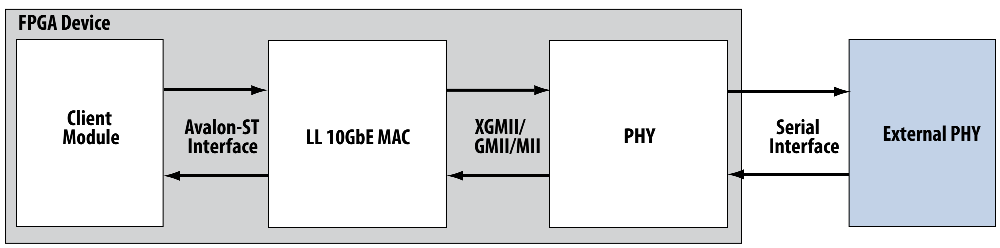
Contents

1 About LL Ethernet 10G MAC

The Low Latency (LL) Ethernet 10G (10GbE) Media Access Controller (MAC) IP core is a configurable component that implements the IEEE 802.3-2008 specification. To build a complete Ethernet subsystem in an Intel FPGA device and connect it to an external device, you can use the LL 10GbE IP core with an Intel FPGA PHY IP core or any of the supported PHYs.

The following figure shows a system with the LL 10GbE MAC IP core.

Figure 1. Typical Application of LL 10GbE MAC



Note: Intel FPGAs implement and support the LL 10GbE Media Access Control (MAC) and Multi-Rate Ethernet PHY (PCS + PMA) IP to interface in a chip-to-chip or chip-to-module channel with external MGBASE-T and NBASE-T (1G/2.5G/5G/10Gb Ethernet) PHY standard devices. However, Intel FPGAs do not comply with or support these interface specifications to directly interface with the required twisted-pair copper cables such as CAT-5/6/7.

Related Links

- Intel FPGA Low Latency Ethernet 10G MAC User Guide Archives on page 109
  Provides a list of user guides for previous versions of the Low Latency Ethernet 10G MAC IP core.
- Low Latency 10G Ethernet MAC Design Example User Guide for Intel Arria 10 Devices
- Low Latency 10G Ethernet MAC Design Example User Guide for Intel Stratix 10 Devices
- Intel Stratix 10 Multi-rate Ethernet PHY IP Core User Guide

1.1 Features

This IP core is designed to the *IEEE 802.3–2008 Ethernet Standard* available on the IEEE website (www.ieee.org). All LL 10GbE IP core variations include MAC only and are in full-duplex mode. These IP core variations offer the following features:

- MAC features:
  - Full-duplex MAC in eight operating modes: 10G, 1G/10G, 1G/2.5G, 1G/2.5G/10G, 1G/2.5G/5G/10G (USXGMII), 10M/100M/1G/10G, 10M/100M/1G/2.5G, and 10M/100M/1G/2.5G/10G.
  - Three variations for selected operating modes: MAC TX only block, MAC RX only block, and both MAC TX and MAC RX block.
  - 10GBASE-R register mode on the TX and RX datapaths, which enables lower latency.
  - Programmable promiscuous (transparent) mode.
  - Unidirectional feature as specified by IEEE 802.3 (Clause 66).
  - Priority-based flow control (PFC) with programmable pause quanta. PFC supports 2 to 8 priority queues.
- Interfaces:
  - Client-side—32-bit Avalon®-ST interface.
  - Management—32-bit Avalon-MM interface.
  - PHY-side—32-bit XGMII for 10GbE, 16-bit GMII for 2.5GbE, 8-bit GMII for 1GbE, or 4-bit MII for 10M/100M.
- Frame structure control features:
  - Virtual local area network (VLAN) and stacked VLAN tagged frames decoding (type 'h8100).
  - Cyclic redundancy code (CRC)-32 computation and insertion on the TX datapath. Optional CRC checking and forwarding on the RX datapath.
  - Deficit idle counter (DIC) for optimized performance with average inter-packet gap (IPG) for LAN applications.
  - Ethernet flow control using pause frames.
  - Programmable maximum length of TX and RX data frames up to 64 Kbytes (KB).
  - Preamble passthrough mode on TX and RX datapaths, which allows user defined preamble in the client frame.
  - Optional padding insertion on the TX datapath and termination on the RX datapath.
- Frame monitoring and statistics:
  - Optional CRC checking and forwarding on the RX datapath.
  - Optional statistics collection on TX and RX datapaths.
- Optional timestamping as specified by the IEEE 1588v2 standard for the following configurations:
  - 10GbE MAC with 10GBASE-R PHY IP core
  - 1G/10GbE MAC with 1G/10GbE PHY IP core
  - 1G/2.5GbE MAC with 1G/2.5GbE Multi-rate Ethernet PHY IP core
  - 1G/2.5G/10GbE MAC with 1G/2.5G/10GbE (MGBASE-T) Multi-rate Ethernet PHY IP core
  - 10M/100M/1G/10GbE MAC with 10M-10GbE PHY IP core

1.1.1 LL Ethernet 10G MAC and Legacy 10-Gbps Ethernet MAC

Current users of the legacy 10-Gbps Ethernet MAC IP core can use the following table to consider migrating to the LL Ethernet 10G MAC IP core.

Table 1. Features Comparison

Feature	LL 10GbE MAC	Legacy 10GbE MAC
Operating mode	10G, 1G/10G, 10M/100M/1G/10G, 1G/ 2.5G, 1G/2.5G/10G (MGBASE-T), 1G/ 2.5G/5G/10G (USXGMII), 10M/ 100M/1G/2.5G, 10M/100M/1G/ 2.5G/10G (MGBASE-T)	10G, 1G/10G, 10M/100M/1G/10G
Device support [1]	Intel® Arria® 10, Arria V, Intel Cyclone® 10 GX, Intel Stratix® 10, Stratix V	Arria V, Arria II, Cyclone V, Cyclone IV, Stratix V, Stratix IV
Operating frequency	• 312.5 MHz • 322.265625 MHz (10GBASE-R register mode enabled)	• 156.25MHz
Latency (TX + RX)	For Intel Arria 10 and Intel Cyclone 10 GX devices: • 60.8 ns (10G MAC) • 307 ns (1G MAC) For Intel Stratix 10 devices: • 70.4 ns (10G MAC) • 319.9 ns (1G MAC)	• 140.8 ns (10G MAC) • 422.4 ns (1G MAC)
Resource utilization	For Intel Arria 10 and Intel Cyclone 10 GX devices: 1600 ALMs, 2400 ALUTs, 2600 Registers (10G with all options disabled) For Intel Stratix 10 devices: 2000 ALMs, 2700 ALUTs, 2900 Registers (10G with all options disabled)	2300 ALMs, 3100 ALUTs, 4400 Registers, 2 M20Ks (10G with all options disabled)
Avalon-ST interface data width	• 32 bits • 64 bits, when the backward compatibility to the legacy MAC is enabled.	• 64 bits
XGMII data width	• 32 bits • Supports backward compatibility with the legacy MAC	• 64 bits
Configuration registers	• 10-bit address bus • Supports backward compatibility with the legacy MAC	• 13-bit address bus
Error detection and correction (ECC)	Supported	Not supported
10GBASE-R register mode	Supported	Not supported
96-bit and 64-bit ToD clock formats	Supported	Not supported
Programmable IPG	Supported	Not supported

[1] Device support depends on the operating mode. Refer to the individual user guides for further details.

Related Links

Altera Low Latency Ethernet 10G MAC IP Core Migration Guidelines
Provides more information on migrating from the legacy 10G Ethernet MAC IP Core to the Low Latency Ethernet 10G MAC IP Core.

1.2 Release Information

Table 2. Release Information of the LL Ethernet 10G MAC IP Core

Item	Description
Version	17.1
Release Date	November 2017
Ordering Code (without the IEEE 1588v2 feature)	IP-10GEUMAC
Ordering Code (with the IEEE 1588v2 feature)	IP-10GEUMACF
Vendor ID	6AF7

Intel verifies that the current version of the Intel Quartus® Prime software compiles the previous version of each Intel FPGA IP core function, if this Intel FPGA IP core function was included in the previous release. Any exceptions to this verification are reported in the *Intel FPGA IP Release Notes*. Intel does not verify compilation with Intel FPGA IP core function versions older than the previous release.

Related Links

- Intel FPGA IP Release Notes
- Errata for Low Latency Ethernet 10G MAC MegaCore function in the Knowledge Base

1.3 Device Family Support

Table 3. Intel FPGA IP Core Device Support Levels

Device Support Level	Definition
Preliminary	Intel verifies the IP core 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. This IP core can be used in production designs with caution.
Final	Intel verifies the IP core with final timing models for this device family. The IP core meets all functional and timing requirements for the device family. This IP core is ready to be used in production designs.

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

Table 4. Device Family Support for LL 10GbE MAC

Device Family	Support	Minimum Speed Grade	
		With 1588 Feature	Without 1588 Feature
Intel Stratix 10	Preliminary	-I2, -E2	-I3, -C3
Intel Arria 10	Final	-I2, -E2	-I3, -E3
Intel Cyclone 10 GX	Final	-I2, -E2	-I3, -E3
Stratix V	Final	-I3, -C3	-I4, -C4
Arria V	Final	-I3, -C3	-I4, -C4

The following table lists possible configurations and the devices each configuration supports:

Table 5. Device Family Support for Configurations

Configuration	Arria V	Intel Arria 10	Intel Cyclone 10 GX	Stratix V	Intel Stratix 10
10G MAC with 10GBASE-R PHY	Arria V GZ	—	—	Yes	—
10G MAC with 10GBASE-R PHY and IEEE 1588v2	Arria V GZ	—	—	Yes	—
10G MAC with Intel Arria 10/Intel Cyclone 10 GX Transceiver Native PHY presets: 10GBASE-R, 10GBASE-R Low Latency, 10GBASE-R Register Mode, and 10GBASE-R w/KR-FEC.	—	Yes	Yes	—	—
10G MAC with Intel Stratix 10 Transceiver Native PHY preset: 10GBASE-R, 10BASE-R Low Latency, 10GBASE-R 1588, and 10GBASE-R w/KR-FEC.	—	—	—	—	Yes
1G/2.5G/10G MAC with 1G/2.5G/10G (MGBASE-T) Multi-rate Ethernet PHY connected to an external MGBASE-T PHY	—	Yes	—	—	Yes
1G/2.5G/10G MAC with 1G/2.5G/10G (MGBASE-T) Multi-rate Ethernet PHY and IEEE 1588v2	—	—	—	—	Yes

continued...

Configuration	Arria V	Intel Arria 10	Intel Cyclone 10 GX	Stratix V	Intel Stratix 10
1G/2.5G/5G/10G (USXGMII) MAC with 1G/2.5G/5G/10G (USXGMII) Multi-rate Ethernet PHY connected to an external NBASE-T PHY	—	Yes	—	—	Yes
1G/2.5G MAC with 1G/2.5G Multi-rate Ethernet PHY	Arria V GX/GT/SX/ST	Yes	—	—	Yes
1G/2.5G MAC with 1G/2.5G Multi-rate Ethernet PHY and IEEE 1588v2	Arria V GX/GT/SX/ST	—	—	—	Yes
1G/2.5G MAC with 2.5G Multi-rate Ethernet PHY	Arria V GX/GT/SX/ST	—	—	—	Yes
10M/100M/1G/10G MAC	Arria V GZ	Yes	Yes	Yes	Yes
10M/100M/1G/10G MAC with IEEE 1588v2	Arria V GZ	Yes	Yes	Yes	Yes
10M/100M/1G/10G MAC with Backplane Ethernet 10GBASE-KR PHY	Arria V GZ	Yes	—	Yes	—
1G/10G MAC with Backplane Ethernet 10GBASE-KR PHY	—	—	—	—	Yes
10M/100M/1G/10G MAC with 1G/10GbE PHY IP core and IEEE 1588v2	Arria V GZ	Yes	—	Yes	—
10M/100M/1G/2.5G MAC with 1G/2.5G Multi-rate PHY	—	—	—	—	Yes
10M/100M/1G/2.5G/10G MAC with 1G/2.5G/10G (MGBASE-T) Multi-rate PHY	—	—	—	—	Yes

1.4 Performance and Resource Utilization

1.4.1 Resource Utilization

The estimates for operating modes other than 1G/2.5G are obtained by compiling the LL Ethernet 10G MAC with the Intel Quartus Prime software targeting a commercial Stratix V. For 1G/2.5G, the target devices are Intel Stratix 10, Intel Arria 10 and Intel Cyclone 10 GX. These estimates are generated by the fitter, excluding the virtual I/Os.

Table 6. Resource Utilization for LL Ethernet 10G MAC for Intel Stratix 10 Devices

<table>
<tr><th colspan="3">MAC Settings</th><th rowspan="2">ALMs</th><th rowspan="2">ALUTs</th><th rowspan="2">Logic Registers</th><th rowspan="2">Memory Block (M20K)</th></tr>
<tr><th>Operating Mode</th><th colspan="2">Enabled Options</th></tr>
<tr><td>10G</td><td colspan="2">None.</td><td>2,000</td><td>2,700</td><td>2,900</td><td>0</td></tr>
<tr><td>10G</td><td colspan="2">Memory-based statistics counters.</td><td>2,700</td><td>3,600</td><td>4,300</td><td>4</td></tr>
<tr><td>1G/2.5G</td><td colspan="2">Supplementary addresses.
Memory-based statistics counters.</td><td>3,500</td><td>4,300</td><td>5,900</td><td>4</td></tr>
<tr><td>1G/2.5G</td><td colspan="2">Supplementary addresses.
Memory-based statistics counters.
Timestamping. Time of day: 96b and 64b.</td><td>7,000</td><td>8,100</td><td>13,100</td><td>19</td></tr>
<tr><td>1G/2.5G/10G</td><td colspan="2">Supplementary addresses.
Memory-based statistics counters.</td><td>3,600</td><td>4,500</td><td>6,200</td><td>4</td></tr>
<tr><td>1G/ 2.5G/5G/10G (USXGMII)</td><td colspan="2">Supplementary addresses.
Memory-based statistics counters.</td><td>3,000</td><td>4,000</td><td>4,800</td><td>4</td></tr>
<tr><td>10M/100M/ 1G/10G</td><td colspan="2">Memory-based statistics counters.</td><td>3,400</td><td>4,400</td><td>5,500</td><td>4</td></tr>
<tr><td rowspan="3">10M/100M/ 1G/10G</td><td rowspan="3">Timestamping.
Memory-based statistics counters.</td><td>Time of day: 96b and 64b.</td><td>6,800</td><td>8,300</td><td>12,100</td><td>17</td></tr>
<tr><td>Time of day: 96b</td><td>6,300</td><td>7,700</td><td>11,000</td><td>17</td></tr>
<tr><td>Time of day format: 64b</td><td>5,600</td><td>6,700</td><td>10,000</td><td>13</td></tr>
<tr><td>10M/100M/ 1G/10G</td><td colspan="2">All options enabled except the options to maintain compatibility with the legacy Ethernet 10G MAC.</td><td>7,200</td><td>8,600</td><td>12,400</td><td>27</td></tr>
<tr><td>10M/100M/1G/ 2.5G</td><td colspan="2">Supplementary addresses.
Memory-based statistics counters.</td><td>3,500</td><td>4,600</td><td>6,100</td><td>4</td></tr>
<tr><td>10M/100M/1G/ 2.5G/10G</td><td colspan="2">Supplementary addresses.
Memory-based statistics counters.</td><td>3,600</td><td>4,800</td><td>6,800</td><td>4</td></tr>
</table>

Table 7. Resource Utilization for LL Ethernet 10G MAC for Intel Arria 10 and Intel Cyclone 10 GX Devices

<table>
<tr><th colspan="3">MAC Settings</th><th rowspan="2">ALMs</th><th rowspan="2">ALUTs</th><th rowspan="2">Logic Registers</th><th rowspan="2">Memory Block (M20K)</th></tr>
<tr><th>Operating Mode</th><th colspan="2">Enabled Options</th></tr>
<tr><td>10G</td><td colspan="2">None.</td><td>1,600</td><td>2,400</td><td>2,600</td><td>0</td></tr>
<tr><td>10G</td><td colspan="2">Memory-based statistics counters.</td><td>2,000</td><td>3,200</td><td>3,700</td><td>4</td></tr>
<tr><td>1G/2.5G</td><td colspan="2">Supplementary addresses.
Memory-based statistics counters.</td><td>2,700</td><td>4,000</td><td>5,000</td><td>5</td></tr>
<tr><td>1G/2.5G</td><td colspan="2">Supplementary addresses.
Memory-based statistics counters.
Timestamping. Time of day: 96b and 64b.</td><td>5,300</td><td>7,600</td><td>12,000</td><td>20</td></tr>
<tr><td>1G/2.5G/10G</td><td colspan="2">Supplementary addresses.
Memory-based statistics counters.</td><td>2,900</td><td>4,100</td><td>5,600</td><td>4</td></tr>
<tr><td>1G/2.5G/5G/10G (USXGMII)</td><td colspan="2">Supplementary addresses.
Memory-based statistics counters.</td><td>2,200</td><td>3,400</td><td>4,300</td><td>4</td></tr>
<tr><td>10M/100M/1G/10G</td><td colspan="2">Memory-based statistics counters.</td><td>2,600</td><td>3,900</td><td>4,900</td><td>4</td></tr>
<tr><td rowspan="3">10M/100M/1G/10G</td><td rowspan="3">Timestamping.
Memory-based statistics counters.</td><td>Time of day: 96b and 64b.</td><td>5,000</td><td>7,200</td><td>11,800</td><td>19</td></tr>
<tr><td>Time of day: 96b</td><td>4,700</td><td>6,800</td><td>10,700</td><td>17</td></tr>
<tr><td>Time of day format: 64b</td><td>4,200</td><td>6,000</td><td>9,700</td><td>13</td></tr>
<tr><td>10M/100M/1G/10G</td><td colspan="2">All options enabled except the options to maintain compatibility with the legacy Ethernet 10G MAC.</td><td>5,400</td><td>7,600</td><td>12,200</td><td>27</td></tr>
<tr><td>10M/100M/1G/2.5G</td><td colspan="2">Supplementary addresses.
Memory-based statistics counters.</td><td>2,900</td><td>4,300</td><td>5,500</td><td>5</td></tr>
<tr><td>10M/100M/1G/2.5G/10G</td><td colspan="2">Supplementary addresses.
Memory-based statistics counters.</td><td>3,000</td><td>4,400</td><td>5,700</td><td>5</td></tr>
</table>

1.4.2 TX and RX Latency

The TX and RX latency values are based on the following definitions and assumptions:

- TX latency is the time taken for the data frame to move from the Avalon-ST interface to the PHY-side interface.
- RX latency is the time taken for the data frame to move from the PHY-side interface to the Avalon-ST interface.
- No backpressure on the Avalon-ST TX and RX interfaces.
- All options under **Legacy Ethernet 10G MAC interfaces**, that allow compatibility with the legacy MAC are disabled.

Table 8. TX and RX Latency Values for Intel Stratix 10 Devices

These latency values are MAC-only latencies and do not include the PHY latencies.

MAC Operating Mode	Speed	Latency (ns)		
		TX	RX	Total
10G	10 Gbps	28.8	41.6	70.4
1G/10G	1 Gbps	156	163.9	319.9
1G/2.5G/10G	1 Gbps	182.7	199.1	381.8
1G/2.5G/10G	2.5 Gbps	106	92.8	198.8
1G/2.5G/10G	10 Gbps	35.2	35.2	70.4
1G/2.5G	1 Gbps	235.2	222.4	457.6
1G/2.5G	2.5 Gbps	140.8	121.7	262.5
10M/100M/1G/10G	10 Mbps	1484.9	20827.7	22312.6
10M/100M/1G/10G	100 Mbps	245.3	2106.1	2351.4
1G/2.5G/5G/10G (USXGMII)	10 Gbps	28.8	41.6	70.4
1G/2.5G/5G/10G (USXGMII)	5 Gbps	41.6	67.2	108.8
1G/2.5G/5G/10G (USXGMII)	2.5 Gbps	57.6	118.4	176
1G/2.5G/5G/10G (USXGMII)	1 Gbps	128	272	400
10M/100M/1G/2.5G	100 M	1360	1664	3024
10M/100M/1G/2.5G	10 M	12561	17662	30223
10M/100M/1G/2.5G/10G	100 M	1289	1640	2929
10M/100M/1G/2.5G/10G	10 M	14230	17641	31871

Table 9. TX and RX Latency Values for Intel Arria 10 and Intel Cyclone 10 GX Devices

These latency values are MAC-only latencies and do not include the PHY latencies.

MAC Operating Mode	Speed	Latency (ns)		
		TX	RX	Total
10G	10 Gbps	22.4	38.4	60.8
1G/10G	1 Gbps	148	159	307
1G/2.5G/10G	1 Gbps	182.7	195.9	378.6
1G/2.5G/10G	2.5 Gbps	93.2	89.7	182.9
1G/2.5G/10G	10 Gbps	28.8	32	60.8
1G/2.5G	1 Gbps	238.4	219.2	457.6
1G/2.5G	2.5 Gbps	137.1	112.5	249.6
10M/100M/1G/10G	10 Mbps	1492.9	20822.9	22315.8
10M/100M/1G/10G	100 Mbps	253.3	2104.5	2357.8
1G/2.5G/5G/10G (USXGMII)	10 Gbps	22.4	38.4	60.8
1G/2.5G/5G/10G (USXGMII)	5 Gbps	32	64	96
1G/2.5G/5G/10G (USXGMII)	2.5 Gbps	54.4	115.2	169.6

continued...

MAC Operating Mode	Speed	Latency (ns)		
		TX	RX	Total
1G/2.5G/5G/10G (USXGMII)	1 Gbps	118.4	268.9	387.3
10M/100M/1G/2.5G	100 M	1360	1657.6	3017.6
10M/100M/1G/2.5G	10 M	12561	17656	30217
10M/100M/1G/2.5G/10G	100 M	1289	1637	2926
10M/100M/1G/2.5G/10G	10 M	14227	17638	31865

2 Getting Started

This chapter provides a general overview of the Intel FPGA IP core design flow to help you quickly get started with LL Ethernet 10G MAC.

2.1 Introduction to Intel FPGA IP Cores

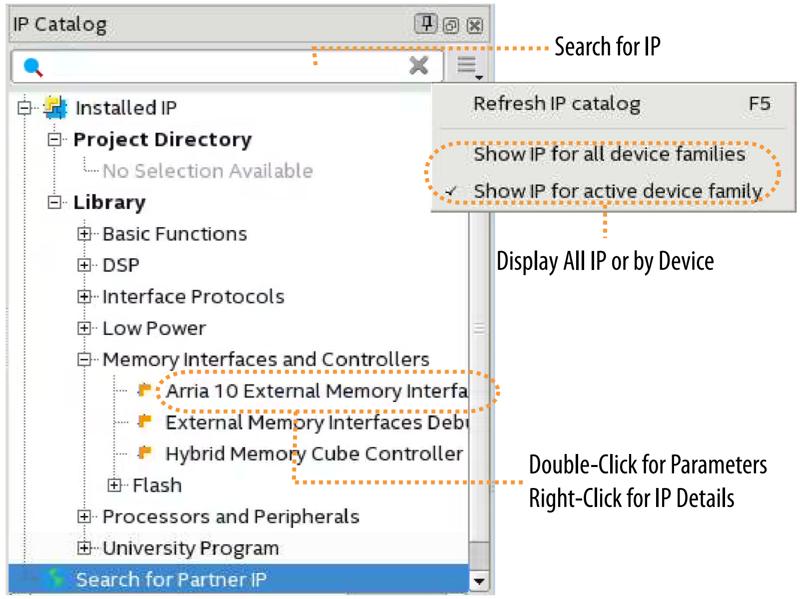
Intel and strategic IP partners offer a broad portfolio of configurable IP cores optimized for Intel FPGA devices.

The Intel Quartus Prime software installation includes the Intel FPGA IP library. Integrate optimized and verified Intel FPGA IP cores into your design to shorten design cycles and maximize performance. The Intel Quartus Prime software also supports integration of IP cores from other sources. Use the IP Catalog (**Tools ➤ IP Catalog**) to efficiently parameterize and generate synthesis and simulation files for your custom IP variation. The Intel FPGA IP library includes the following types of IP cores:

- Basic functions
- DSP functions
- Interface protocols
- Low power functions
- Memory interfaces and controllers
- Processors and peripherals

This document provides basic information about parameterizing, generating, upgrading, and simulating stand-alone IP cores in the Intel Quartus Prime software.

Figure 2. IP Catalog



2.2 Installing and Licensing Intel FPGA IP Cores

The Intel Quartus Prime software installation includes the Intel FPGA IP library. This library provides many useful IP cores for your production use without the need for an additional license. Some Intel FPGA IP cores require purchase of a separate license for production use. The Intel FPGA IP Evaluation Mode allows you to evaluate these licensed Intel FPGA IP cores in simulation and hardware, before deciding to purchase a full production IP core license. You only need to purchase a full production license for licensed Intel IP cores after you complete hardware testing and are ready to use the IP in production.

The Intel Quartus Prime software installs IP cores in the following locations by default:

Figure 3. IP Core Installation Path

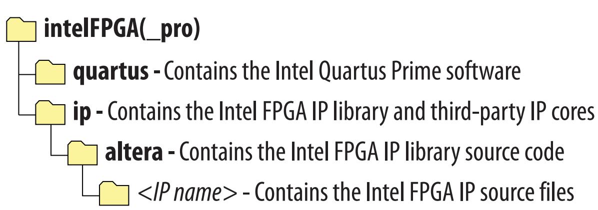


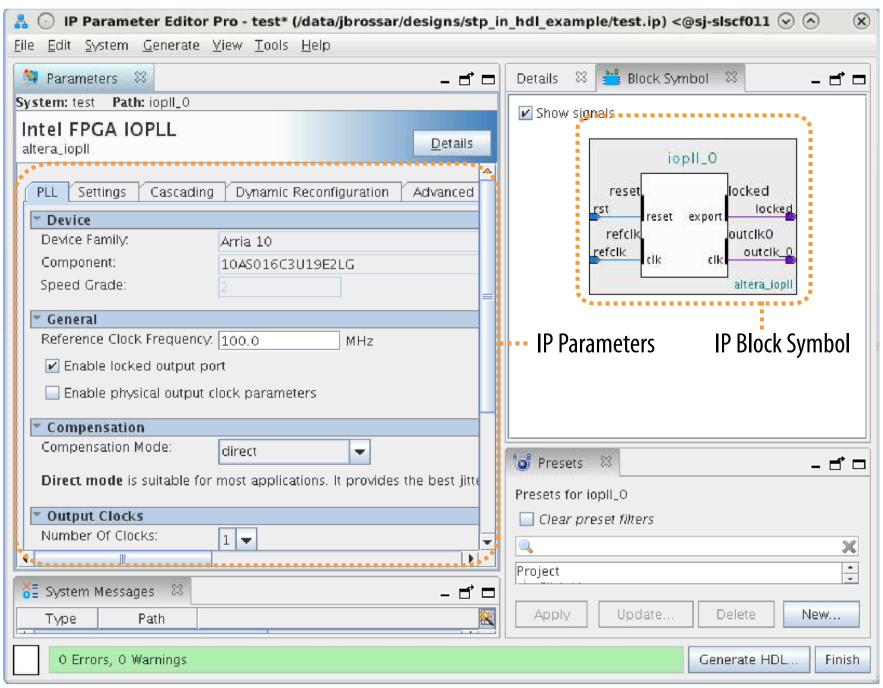
Table 10. IP Core Installation Locations

Location	Software	Platform
<*drive*>:\intelFPGA_pro\quartus\ip\altera	Intel Quartus Prime Pro Edition	Windows*
<*drive*>:\intelFPGA\quartus\ip\altera	Intel Quartus Prime Standard Edition	Windows
<*home directory*>:/intelFPGA_pro/quartus/ip/altera	Intel Quartus Prime Pro Edition	Linux*
<*home directory*>:/intelFPGA/quartus/ip/altera	Intel Quartus Prime Standard Edition	Linux

2.3 Generating IP Cores (Intel Quartus Prime Pro Edition)

Quickly configure Intel FPGA IP cores in the Intel Quartus Prime parameter editor. Double-click any component in the IP Catalog to launch the parameter editor. The parameter editor allows you to define a custom variation of the IP core. The parameter editor generates the IP variation synthesis and optional simulation files, and adds the `.ip` file representing the variation to your project automatically.

Figure 4. IP Parameter Editor (Intel Quartus Prime Pro Edition)



Follow these steps to locate, instantiate, and customize an IP core in the parameter editor:

1. Create or open an Intel Quartus Prime project (`.qpf`) to contain the instantiated IP variation.
2. In the IP Catalog (**Tools ➤ IP Catalog**), locate and double-click the name of the IP core to customize. To locate a specific component, type some or all of the component's name in the IP Catalog search box. The New IP Variation window appears.
3. Specify a top-level name for your custom IP variation. Do not include spaces in IP variation names or paths. The parameter editor saves the IP variation settings in a file named <*your_ip*>`.ip`. Click **OK**. The parameter editor appears.
4. Set the parameter values in the parameter editor and view the block diagram for the component. The **Parameterization Messages** tab at the bottom displays any errors in IP parameters:
    - Optionally, select preset parameter values if provided for your IP core. Presets specify initial parameter values for specific applications.
    - Specify parameters defining the IP core functionality, port configurations, and device-specific features.
    - Specify options for processing the IP core files in other EDA tools.

    Note: Refer to your IP core user guide for information about specific IP core parameters.
5. Click **Generate HDL**. The **Generation** dialog box appears.
6. Specify output file generation options, and then click **Generate**. The synthesis and simulation files generate according to your specifications.
7. To generate a simulation testbench, click **Generate ➤ Generate Testbench System**. Specify testbench generation options, and then click **Generate**.
8. To generate an HDL instantiation template that you can copy and paste into your text editor, click **Generate ➤ Show Instantiation Template**.
9. Click **Finish**. Click **Yes** if prompted to add files representing the IP variation to your project.
10. After generating and instantiating your IP variation, make appropriate pin assignments to connect ports.

    Note: Some IP cores generate different HDL implementations according to the IP core parameters. The underlying RTL of these IP cores contains a unique hash code that prevents module name collisions between different variations of the IP core. This unique code remains consistent, given the same IP settings and software version during IP generation. This unique code can change if you edit the IP core's parameters or upgrade the IP core version. To avoid dependency on these unique codes in your simulation environment, refer to *Generating a Combined Simulator Setup Script*.

Related Links

- Intel FPGA IP Release Notes
- IP User Guide Documentation
- Low Latency 10G Ethernet MAC Design Example User Guide for Intel Arria 10 Devices

- Low Latency 10G Ethernet MAC Design Example User Guide for Intel Stratix 10 Devices

2.4 Generated Files

The following table describes the generated files and other files that might be in your project directory. The names and types of generated files specified in the IP parameter editor report vary depending on whether you create your design with VHDL or Verilog HDL.

Table 11. Generated Files

Extension	Description
<*variation name*>**.v** or **.vhd**	A Intel FPGA IPcore variation file, which defines a VHDL or Verilog HDL description of the custom Intel FPGA IP core. Instantiate the entity defined by this file inside of your design. Include this file when compiling your design in the Intel Quartus Prime software.
<*variation name*>**.cmp**	A VHDL component declaration file for the Intel FPGA IP core variation. Add the contents of this file to any VHDL architecture that instantiates the Intel FPGA IP core.
<*variation name*>**.qsys**	A Platform Designer file for the MAC IP core design.
<*variation name*>**.qip**	Contains Intel Quartus Prime project information for your Intel FPGA IP core variation.
<*variation name*>**.bsf**	Intel Quartus Prime symbol file for the Intel FPGA IP core variation. Use this file in the Intel Quartus Prime block diagram editor.
<*variation name*>**.sip**	Contains IP core library mapping information required by the Intel Quartus Prime software. The Intel Quartus Prime software generates a **. sip** file during generation of some Intel FPGA IP cores. You must add any generated **.sip** file to your project for use by NativeLink simulation and the Intel Quartus Prime Archiver.
<*variation name*>**.spd**	Contains a list of required simulation files for your Intel FPGA IP core.

2.5 Simulating Intel FPGA IP Cores

The Intel Quartus Prime software supports IP core RTL simulation in specific EDA simulators. IP generation creates simulation files, including the functional simulation model, any testbench (or example design), and vendor-specific simulator setup scripts for each IP core. Use the functional simulation model and any testbench or example design for simulation. IP generation output may also include scripts to compile and run any testbench. The scripts list all models or libraries you require to simulate your IP core.

The Intel Quartus Prime software provides integration with many simulators and supports multiple simulation flows, including your own scripted and custom simulation flows. Whichever flow you choose, IP core simulation involves the following steps:

1. Generate simulation model, testbench (or example design), and simulator setup script files.
2. Set up your simulator environment and any simulation scripts.
3. Compile simulation model libraries.
4. Run your simulator.

2.6 Creating a Signal Tap Debug File to Match Your Design Hierarchy

For Intel Arria 10 and Intel Cyclone 10 GX devices, the Intel Quartus Prime software generates two files, `build_stp.tcl` and `<ip_core_name>.xml`. You can use these files to generate a Signal Tap file with probe points matching your design hierarchy.

The Intel Quartus Prime software stores these files in the `<IP core directory>/synth/debug/stp/` directory.

Synthesize your design using the Intel Quartus Prime software.

1. To open the Tcl console, click **View ➤ Utility Windows ➤ Tcl Console**.
2. Type the following command in the Tcl console:
   `source <IP core directory>/synth/debug/stp/build_stp.tcl`
3. To generate the STP file, type the following command:
   `main -stp_file <output stp file name>.stp -xml_file <input xml_file name>.xml -mode build`
4. To add this Signal Tap file (**.stp**) to your project, select **Project ➤ Add/Remove Files in Project**. Then, compile your design.
5. To program the FPGA, click **Tools ➤ Programmer**.
6. To start the Signal Tap Logic Analyzer, click **Quartus Prime ➤ Tools ➤ Signal Tap Logic Analyzer**.

   The software generation script may not assign the Signal Tap acquisition clock in `<output stp file name>.stp`. Consequently, the Intel Quartus Prime software automatically creates a clock pin called `auto_stp_external_clock`. You may need to manually substitute the appropriate clock signal as the Signal Tap sampling clock for each STP instance.
7. Recompile your design.
8. To observe the state of your IP core, click **Run Analysis**.

   You may see signals or Signal Tap instances that are red, indicating they are not available in your design. In most cases, you can safely ignore these signals and instances. They are present because software generates wider buses and some instances that your design does not include.

2.7 Parameter Settings for the LL Ethernet 10G MAC IP Core

You customize the MAC IP core by specifying the parameters on the parameter editor in the Intel Quartus Prime software. The parameter editor enables only the parameters that are applicable to the selected speed.

Parameter	Value	Description
Speed	10G, 1G/10G, 10M/100M/1G/10G, 1G/2.5G, 1G/2.5G/10G, 1G/2.5G/5G/10G (USXGMII), 10M/100M/1G/2.5G, 10M/100M/1G/2.5G/10G	Select the desired speed. By default, **10G** is selected.
Datapath options	TX only, RX only, TX & RX	Select the MAC variation to instantiate.

continued...

Parameter	Value	Description
		• TX only—instantiates MAC TX. • RX only—instantiates MAC RX. • TX & RX—instantiates both MAC TX and RX.
Enable ECC on memory blocks	On, Off	Turn on this option to enable error detection and correction on memory blocks. This option is available to designs that target Intel Stratix 10, Stratix V, Arria V GZ, Intel Arria 10, and Intel Cyclone 10 GX devices only.
Enable preamble pass-through mode	On, Off	Turn on this option to enable preamble pass-through mode. You must also set the `tx_preamble_control`, `rx_preamble_control`, and `rx_custom_preamble_forward` registers to 1. When enabled, the MAC IP core allows custom preamble in data frames on the transmit and receive datapaths. This option is available only for **10G**.
Enable priority-based flow control (PFC)	On, Off	Turn on this option to enable PFC. You must also set the `tx_pfc_priority_enable[n]`bit to 1 and specify the number of priority queues in the **Number of PFC queues** field. This option is available only for **10G**.
Number of PFC queues	2—8	Specify the number of PFC queues. This option is only enabled if you turn **Enable priority-based flow control (PFC)**.
Enable unidirectional feature	On, Off	Turn on this option to enable unidirectional feature as specified in the IEEE802.3 specification (Clause 66). This feature is only supported in 10Gbps speed mode.
Enable 10GBASE-R register mode	On, Off	Turn on this option to enable 10GBASE-R register mode on the TX and RX datapaths to further reduce the round-trip latency between the MAC and PHY. In this mode, the MAC datapaths must run at 322.265625 MHz. This option is available only for **10G TX & RX** variation. It is not available with the following features: • preamble passthrough, • priority-based flow control, • unidirectional, • timestamping, and • 64-bit compatibility options on XGMII and Avalon-ST interface.
Enable supplementary address	On, Off	Turn on this option to enable supplementary addresses. You must also set the EN_SUPP0/1/2/3 bits in the `rx_frame_control` register to 1.
Enable statistics collection	On, Off	Turn on this option to collect statistics on the TX and RX datapaths.
Statistics counters	Memory-based, Register-based	Specify the implementation of the statistics counters. When you turn on **Statistics collection**, the default implementation of the counters is **Memory-based**. • Memory-based—selecting this option frees up logic elements. The MAC IP core does not clear memory-based counters after they are read. • Register-based—selecting this option frees up the memory. The MAC IP core clears register-based statistic counters after the counters are read.

continued...

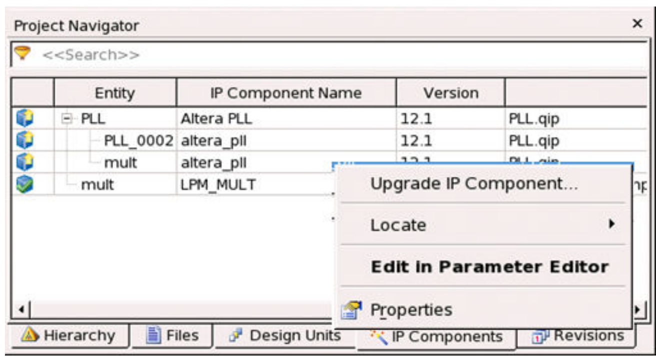
Parameter	Value	Description
		Memory-based statistics counters may not be accurate when the MAC IP core receives or transmits back-to-back undersized frames. On the TX datapath, you can enable padding to avoid this situation. Undersized frames are frames with less than 64 bytes.
Enable time stamping	On, Off	Turn on this option to enable time stamping on the TX and RX datapaths. This option is not available in 1G/2.5G/10G (Intel Arria 10 and Intel Cyclone 10 GX devices only), 1G/2.5G/5G/10G (USXGMII), 10M/100M/1G/2.5G, and 10M/100M/1G/2.5G/10G configurations.
Enable PTP one-step clock support	On, Off	Turn on this option to enable 1-step time stamping. This option is enabled only when you turn on time stamping. This option is not available in 1G/2.5G/10G (Intel Arria 10 and Intel Cyclone 10 GX devices only), 1G/2.5G/5G/10G (USXGMII), 10M/100M/1G/2.5G, and 10M/100M/1G/2.5G/10G configurations.
Enable asymmetry support	On, Off	Turn on this option to enable asymmetry support on TX datapath. This option is enabled only when you turn on time stamping and PTP one-step clock support. This option is not available in 1G/2.5G/10G (Intel Arria 10 and Intel Cyclone 10 GX devices only), 1G/2.5G/5G/10G (USXGMII), 10M/100M/1G/2.5G, and 10M/100M/1G/2.5G/10G configurations.
Enable peer-to-peer	On, Off	Turn on this option to enable peer-to-peer support on TX datapath. This option is enabled only when you turn on time stamping and PTP one-step clock support. This option is not available in 1G/2.5G/10G (Intel Arria 10 and Intel Cyclone 10 GX devices only), 1G/ 2.5G/5G/10G (USXGMII), 10M/100M/1G/2.5G, and 10M/100M/1G/2.5G/10G configurations. *Note:* This option is only available from Intel Quartus Prime version 17.1 onwards.
Timestamp fingerprint width	1–32	Specify the width of the timestamp fingerprint in bits on the TX path. The default value is 4 bits. This option is not available in 1G/2.5G/10G (Intel Arria 10 and Intel Cyclone 10 GX devices only), 1G/2.5G/5G/10G (USXGMII), 10M/100M/1G/2.5G, and 10M/100M/1G/2.5G/10G configurations.
Time of Day Format	Enable 96b Time of Day Format only, Enable 64b Time of Day Format only, Enable both 96b and 64b Time of Day Format	Specify the time of day format. This option is not available in 1G/2.5G/10G (Intel Arria 10 and Intel Cyclone 10 GX devices only), 1G/2.5G/5G/10G (USXGMII), 10M/100M/1G/2.5G, and 10M/100M/1G/2.5G/10G configurations.
Use legacy Ethernet 10G MAC XGMII Interface	On, Off	Turn on this option to maintain compatibility with the 64-bit Ethernet 10G MAC on the XGMII.
Use legacy Ethernet 10G MAC Avalon Memory-Mapped Interface	On, Off	Turn on this option to maintain compatibility with the 64-bit Ethernet 10G MAC on the Avalon-MM Interface.
Use legacy Ethernet 10G MAC Avalon Streaming Interface	On, Off	Turn on this option to maintain compatibility with the 64-bit Ethernet 10G MAC on the Avalon-ST interface.

2.8 Upgrading the LL Ethernet 10G MAC IP Core

The Intel Quartus Prime software alerts you when your IP core is not upgraded to the current version. Click **Project > Upgrade IP Components** to identify and upgrade the IP cores. To successfully upgrade the IP core, you must ensure that the file structure of your project that was generated by an older version of the software is preserved. Failure to upgrade IP cores can result in a mismatch between the IP core variation and the current supporting libraries.

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* reports any verification exceptions. Intel does not verify the compilation of IP cores older than the previous release.

Figure 5. Upgrading IP Components in Project Navigator



Related Links

Intel FPGA IP Release Notes

2.9 Design Considerations for the LL Ethernet 10G MAC IP Core

2.9.1 Migrating from Legacy Ethernet 10G MAC to LL Ethernet 10G MAC

Intel recommends that you opt for the following migration paths. These migration paths allow you to take advantage of the benefits of LL Ethernet 10G MAC—low resource count and low latency.

2.9.1.1 Migration—32-bit Datapath on Avalon-ST Interface

Follow these steps to implement 32-bit datapath on the Avalon ST and Avalon-MM interfaces.

1. Instantiate the LL Ethernet 10G MAC IP core in your design. If you are using a PHY with 64-bit SDR XGMII interface, turn on the **Use legacy Ethernet 10G MAC XGMII Interface** option.
2. Modify your user logic to accommodate 32-bit datapaths on Avalon-ST TX and RX data interfaces.
3. Ensure that `tx_312_5_clk` and `rx_312_5_clk` are connected to 312.5 MHz clock sources. Intel recommends that you use the same clock source for these clock signals.
4. Update the register offsets to the offsets of the LL Ethernet 10G MAC. The configuration registers of the LL Ethernet 10G MAC allow access to new features such as error correction and detection on memory blocks.
5. If you turn on the **Use legacy Ethernet 10G MAC XGMII Interface** option, add a 156.25 MHz clock source for `tx_156_25_clk` and `rx_156_25_clk`. This 156.25 MHz clock source must be rise-to-rise synchronous to the 312.5 MHz clock source.
6. Ensure that `csr_clk` is within 125 MHz to 156.25 MHz. Otherwise, some statistic counters may not be accurate.

2.9.1.2 Migration—Maintains 64-bit on Avalon-ST Interface

Follow these steps to implement 32-bit to 64-bit adapters on the Avalon ST interface and XGMII, and uses the same register offsets to maintain backward compatibility with the legacy 10-Gbps Ethernet (10GbE) MAC IP Core.

1. Instantiate the LL Ethernet 10G MAC IP core in your design. To maintain compatibility on the interfaces, turn on the **Use legacy Ethernet 10G MAC XGMII Interface**, **Use legacy Ethernet 10G MAC Avalon Memory-Mapped Interface**, and **Use legacy Ethernet 10G MAC Avalon Streaming Interface** options.
2. Ensure that `tx_312_5_clk` and `rx_312_5_clk` are connected to 312.5 MHz clock sources. Intel recommends that you use the same clock source for these clock signals.
3. Add a 156.25 MHz clock source for `tx_156_25_clk` and `rx_156_25_clk`. This 156.25 MHz clock source must be rise-to-rise synchronous to the 312.5 MHz clock source.
4. Ensure that `csr_clk` is within 125 MHz to 156.25 MHz. Otherwise, some statistic counters may not be accurate.

Related Links

Altera Low Latency Ethernet 10G MAC IP Core Migration Guidelines

2.9.2 Timing Constraints

Intel provides timing constraint files (`.sdc`) to ensure that the IP core meets the design timing requirements in Intel FPGA devices. The files constraint the false paths and multicycle paths in the 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.

The timing constraints files are in the IP directory. You can edit these files as necessary. They are for clock crossing logic and grouped as below:

- Pseudo-static CSR fields
- Clock crosser
- Dual clock FIFO

Note: For the IP to work correctly, there must be no other timing constraints files cutting or overriding the paths, for example, `set_false_path`, `set_clock_groups`, at the project level.

2.9.2.1 Pseudo-Static CSR Fields

Most of the configuration registers in the MAC IP core must not be programmed when the MAC is in operation. As such, they are not synchronized to reduce resource usage. These registers are all in the `set_false_path` constraint.

2.9.2.2 Clock Crosser

Clock crossers perform multi-bit signals crossing from one clock domain to another.

The working principle of the clock crosser is to let the crossed-over data stabilize first before indicating that the data is valid in the latched clock domain. Using such structure, the data bits must not skew for more than one latched clock period. The timing constraint file applies a common timing check over all the clock crossers irrespective of their latched clock domain. This is over-pessimistic for signals crossing into the CSR clock, but there are no side-effects, like significant run-time impact and false violations, during the internal testing. If your design runs into clock crosser timing violation paths within the IP and the latched clock domain is `csr_clk`, you can dismiss the violation manually or by editing the `.sdc` file if the violation is less than one `csr_clk` period.

The timing constraint file uses the `set_net_delay` to constraint the fitter placement and `set_max_skew` to perform timing check on the paths. For a project with very high device utilization, Intel recommends that you implement addition steps like floor planning or Logic Lock to aid the place-and-route process. The additional steps can give a more consistent timing closure along these paths instead of only relying on the `set_net_delay`.

A caveat of using `set_max_skew` is that it does not analyze whether the insertion delay of the path in concern exceeds a limit. In other words, a path could meet skew requirement but have longer than expected insertion delay. If this is not checked, it may cause functional failure in certain latency-sensitive paths. Therefore, a custom script (`alt_em10g32_clock_crosser_timing_info.tcl`) is available for you to check that the round-trip clock crosser delay is within expectation. To use this script, manually add it to the user flow and run it. To ensure that the IP core operates correctly, the results must be positive (no error).

2.9.2.3 Dual Clock FIFO

The bit skew of the dual clock FIFO gray-coded pointers must be within one 312.5 MHz clock period.

The timing constraint file uses the `set_net_delay` to constraint the fitter placement and `set_max_skew` to perform timing check on the paths. For a project with very high device utilization, Intel recommends that you implement addition steps like floor planning or Logic Lock to aid the place-and-route process. The additional steps can give a more consistent timing closure along these paths instead of only relying on the `set_net_delay`.

2.9.2.4 Jitter on PLL Clocks

To minimize jitter, the advanced transmit (ATX) PLL and the fractional PLL (fPLL) can source the input reference clock directly from the reference clock buffer without passing through the reference clock network. You must ensure that a location constraint is added to your design to achieve a correct placement on the device.

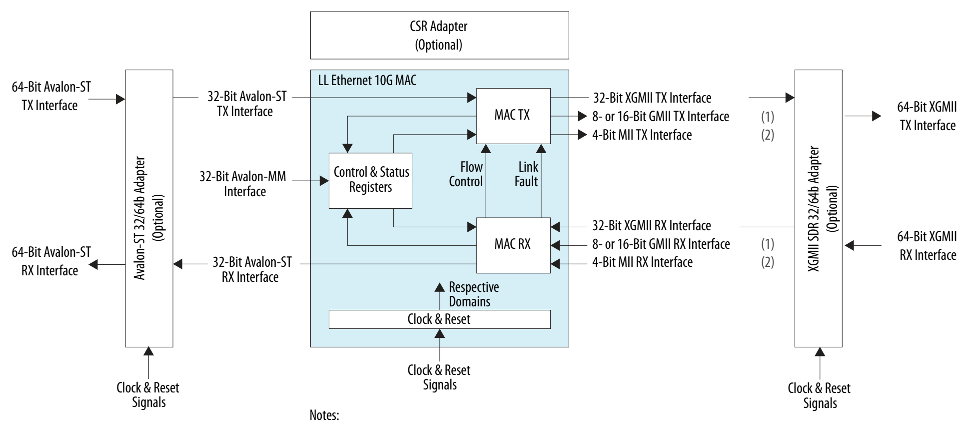
3 Functional Description

The Low Latency (LL) Ethernet 10G MAC IP core handles the flow of data between a client and an Ethernet network through an Ethernet PHY. On the transmit path, the MAC IP core accepts client frames and constructs Ethernet frames by inserting various control fields, such as checksums before forwarding them to the PHY. Similarly, on the receive path, the MAC accepts Ethernet frames via a PHY, performs checks, and removes the relevant fields before forwarding the frames to the client. You can configure the MAC IP core to collect statistics on both transmit and receive paths.

3.1 Architecture

The LL Ethernet 10G MAC IP core is a composition of the following blocks: MAC receiver (MAC RX), MAC transmitter (MAC TX), configuration and status registers, and clock and reset.

Figure 6. LL Ethernet 10G MAC Block Diagram



Notes:
(1) 8-bit applies to 1G/10G and 10M/100M/1G/10G MAC only. 16-bit applies to 1G/2.5G.
(2) Applies to 10M/100M/1G/10G MAC only.

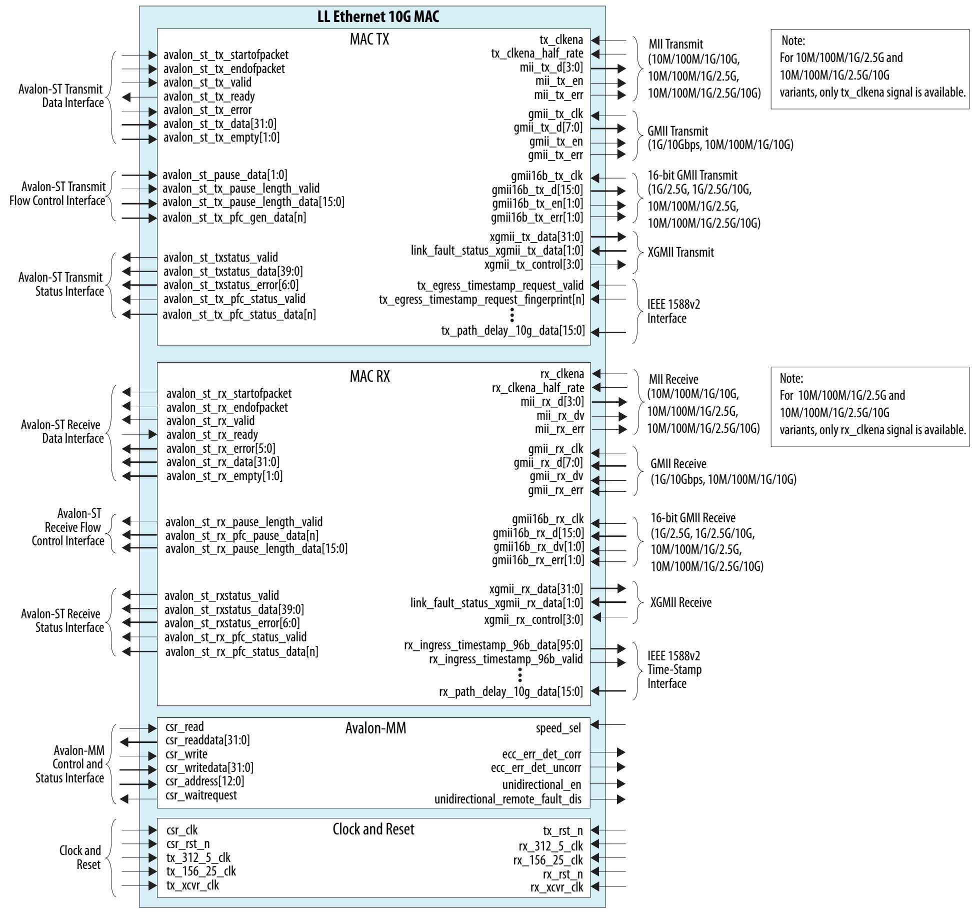
3.2 Interfaces

Table 12. Interfaces

Interfaces	Description
Avalon-ST Interface	The client-side interface of the MAC employs the Avalon-ST protocol, which is a synchronous point-to-point, unidirectional interface that connects the producer of a data stream (source) to a consumer of the data (sink). The key properties of this interface include: • Frame transfers marked by `startofpacket` and `endofpacket` signals. • Signals from source to sink are qualified by the `valid` signal. • Errors marking a current packet are aligned with the end-of-packet cycle. • Use of the `ready` signal by the sink to backpressure the source. In the MAC IP core, the Avalon-ST interface acts as a sink in the TX datapath and source in the RX datapath. This interface supports packets, backpressure, and error detection. It operates at either 312.5 MHz or 156.25 MHz depending on the operating mode. The ready latency on this interface is 0.
Avalon-MM Control and Status Register Interface	The Avalon-MM control and status register interface is an Avalon-MM slave port. This interface uses word addressing which provides access to the configuration and status registers, and statistics counters.
XGMII	In 10G mode, the network-side interface of the MAC IP core implements the XGMII protocol. Depending on the configuration, the XGMII consists of 32- or 64-bit data bus and 4- or 8-bit control bus operating at 312.5 MHz. This interface operates at 322.265625 MHz if the 10GBASE-R register mode is enabled. The data bus carries the MAC frame with the most significant byte occupying the least significant lane.
GMII	In 1G/10G and 10M/100M/1G/10G operating modes, the network-side interface of the MAC IP core implements 8 bits wide GMII protocol when the MAC operates at 1 Gbps. This 8-bit interface supports gigabit operations at 125 MHz. In 1G/2.5G operating mode, the network-side interface of the MAC IP core implements 16 bits wide GMII protocol. This 16-bit interface supports 2.5G operations at 156.25 MHz and 1G operations at 62.5 MHz. For 10M/100M/1G/2.5G and 10M/100M/1G/2.5G/10G variants, the 10M/100M operating mode uses 16 bits wide GMII protocol with TX /RX clock enabled to downsample the data rate /10 and /100. No MII interfaces are available.
MII	In 10M or 100M mode, the network-side interface of the MAC IP core implements the MII protocol. This 4-bit MII supports 10-Mbps and 100-Mbps operations at 125 MHz, with a clock enable signal that divides the clock to effective rates of 2.5 MHz for 10 Mbps and 25 MHz for 100 Mbps.

Figure 7. Interface Signals

The inclusion and width of some signals depend on the operating mode and features selected.



Related Links

Interface Signals on page 87
Describes each signal in detail.

3.3 Frame Types

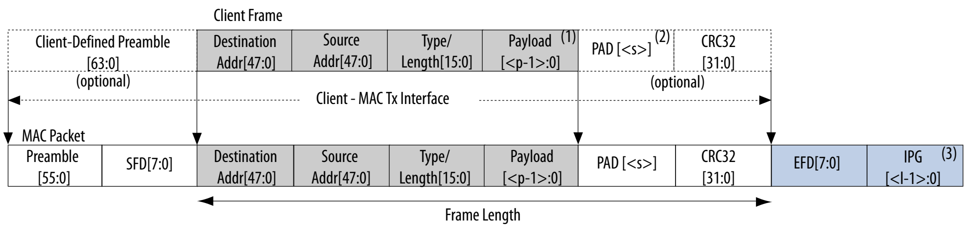
The MAC IP core supports the following frame types:

- Basic Ethernet frames, including jumbo frames.
- VLAN and stacked VLAN frames.
- Control frames, which include pause and PFC frames.

3.4 TX Datapath

The MAC TX receives the client payload data with the destination and source addresses, and appends various control fields depending on the MAC configuration.

Figure 8. Typical Client Frame at TX Interface



3.4.1 Padding Bytes Insertion

By default, the MAC TX inserts padding bytes (0x00) into TX frames to meet the following minimum payload length:

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

Ensure that CRC-32 insertion is enabled when padding bytes insertion is enabled.

You can disable padding bytes insertion by setting the `tx_pad_control` register to 0. When disabled, the MAC IP core forwards the frames to the PHY-side interface without padding. Ensure that the minimum payload length is met; otherwise the current frame may get corrupted. You can check for undersized frames by referring to the statistics collected.

3.4.2 Address Insertion

By default, the MAC TX retains the source address received from the client. You can configure the MAC TX to replace the source address with the primary MAC address specified in the `tx_addrins_macaddr0` and `tx_addrins_macaddr1` registers by setting the bit `tx_src_addr_override[0]` to 1.

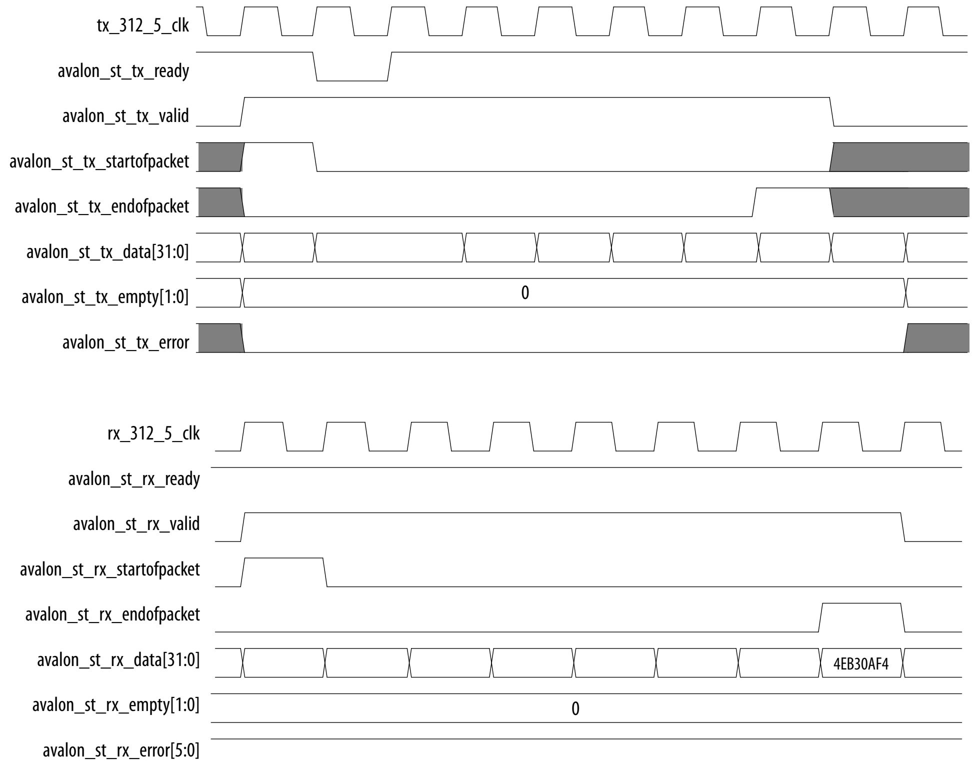
3.4.3 CRC-32 Insertion

By default, the MAC TX computes and inserts CRC-32 checksum into TX frames. The MAC TX computes the CRC-32 checksum over frame bytes that include the source address, destination address, length, data, and padding bytes. The computation excludes the preamble and SFD bytes. The MAC TX then inserts the CRC-32 checksum into the TX frame. Bit 31st of the checksum occupies the least significant bit of the first byte in the CRC field.

You can disable this function by setting the `tx_crc_control[1]` register bit to 0.

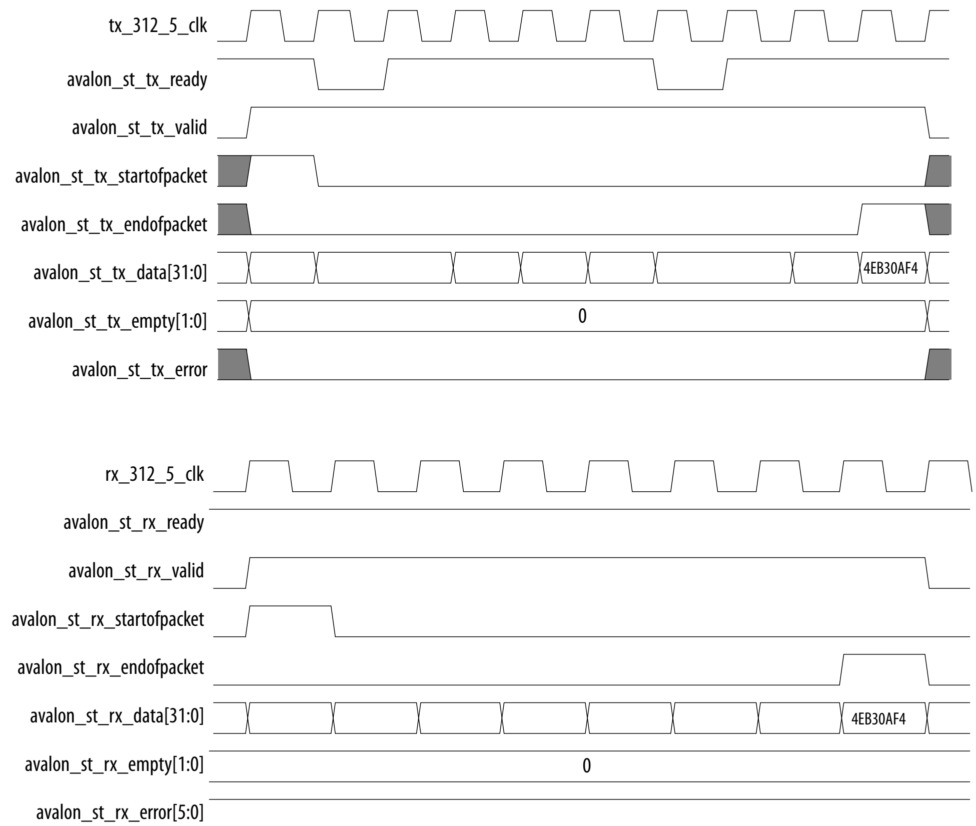
The following figure shows the timing diagram on the Avalon-ST data interfaces where CRC insertion is enabled on transmit and CRC removal is disabled on receive. The frame from the client is without CRC-32 checksum. The MAC TX inserts the CRC-32 checksum (4EB00AF4) into the frame. The frame is then looped back to the RX datapath with the CRC-32 checksum.

Figure 9. Avalon-ST TX and RX Interfaces with CRC Insertion Enabled



The following figure shows the timing diagram on the Avalon-ST data interfaces where CRC insertion is disabled on transmit and CRC removal is disabled on receive. The MAC TX receives the frame from the client with a CRC-32 checksum (4EB00AF4). The frame with the same CRC-32 checksum is then looped back to the RX datapath.

Figure 10. Avalon-ST TX and RX Interface with CRC Insertion Disabled



3.4.4 XGMII Encapsulation

By default, the MAC TX inserts 7-byte preamble, 1-byte SFD and 1-byte EFD (0xFD) into frames received from the client.

The MAC TX also supports custom preamble in 10G operations. To use custom preamble, set the `tx_preamble_control` register to 1. Behavior of the MAC TX in custom preamble mode:

- The MAC TX accepts the first eight bytes in the frame from the client as custom preamble.
- The MAC TX inserts 1-byte EFD (0xFD) into the frame.
- The MAC TX replaces the first byte of the preamble with 1-byte START (0xFB).
- The MAC TX converts the eighth byte of the preamble to a 1-byte SFD (0xD5).

An underflow could occur on the Avalon-ST TX interface. An underflow occurs when the `avalon_st_tx_valid` signal is deasserted in the middle of frame transmission. When this happens, the 10GbE MAC TX inserts an error character |E| into the frame and forwards the frame to the XGMII.

3.4.5 Inter-Packet Gap Generation and Insertion

The MAC TX maintains an average IPG between TX frames as required by the IEEE 802.3 Ethernet standard. The average IPG is maintained at 96 bit times (12 byte times) using the deficit idle count (DIC). The MAC TX inserts or deletes idle bytes depending on the value of the DIC; the DIC must be between 9 to 15 bytes. Averaging the IPG ensures that the MAC utilizes the maximum available bandwidth.

For 10M/100M/1G/2.5G operations, however, the MAC TX maintains a minimum IPG of 12 bytes time.

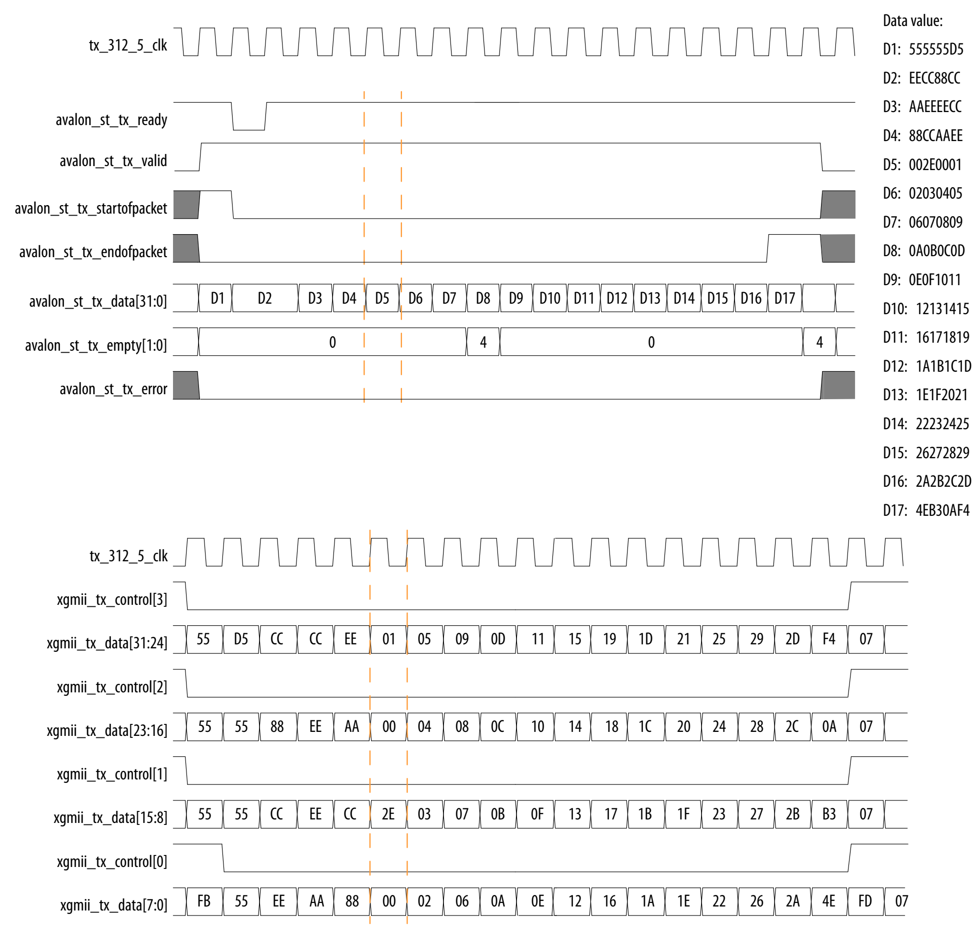
3.4.6 XGMII Transmission

On the XGMII, the MAC TX performs the following:

- Aligns the first byte of the frame to lane 0 of the interface.
- Performs endian conversion. Transmit frames received from the client on the Avalon-ST interface are big endian. Frames transmitted on the XGMII are little endian; the MAC TX therefore transmits frames on this interface from the least significant byte.

The following figure shows the timing on the Avalon-ST TX data interface and XGMII. The least significant byte of the value in D5 is transmitted first on the XGMII.

Figure 11. Endian Conversion



3.4.7 Unidirectional Feature

The MAC TX implements the unidirectional feature as specified by clause 66 in the IEEE802.3 specification. This is an optional feature supported only in 10G operations. When you enable this feature, two output ports—`unidirectional_en`, `unidirectional_remote_fault_dis`— and two register fields—`UniDir_En` (Bit 0), `UniDirRmtFault_Dis` (Bit 1)— are accessible to control the TX XGMII interface.

Table 13. Register Field and Link Status

Bit 0 Register Field	Bit 1 Register Field	Link Status	TX XGMII Interface Behavior
Don't care	*Don't care*	No link fault	Continue to allow normal packet transmission.
0	*Don't care*	Local fault	Immediately override the current content with remote fault sequence.
1	0	Local fault	Continue to send packet if there is one. Otherwise, override the IPG/IDLE bytes with remote fault sequence.(2)

continued...

Bit 0 Register Field	Bit 1 Register Field	Link Status	TX XGMII Interface Behavior
1	1	Local fault	Continue to allow normal packet transmission (similar to no link fault).
0	*Don't care*	Remote fault	Immediately override the current content with IDLE control characters.
1	*Don't care*	Remote fault	Continue to allow normal packet transmission (similar to no link fault).

3.4.8 TX Timing Diagrams

Figure 12. Normal Frame

The following diagram shows the transmission of a normal frame.

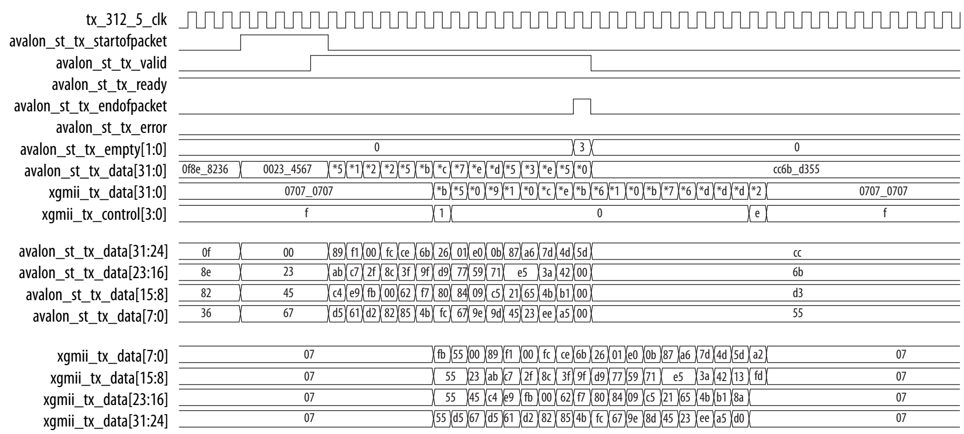
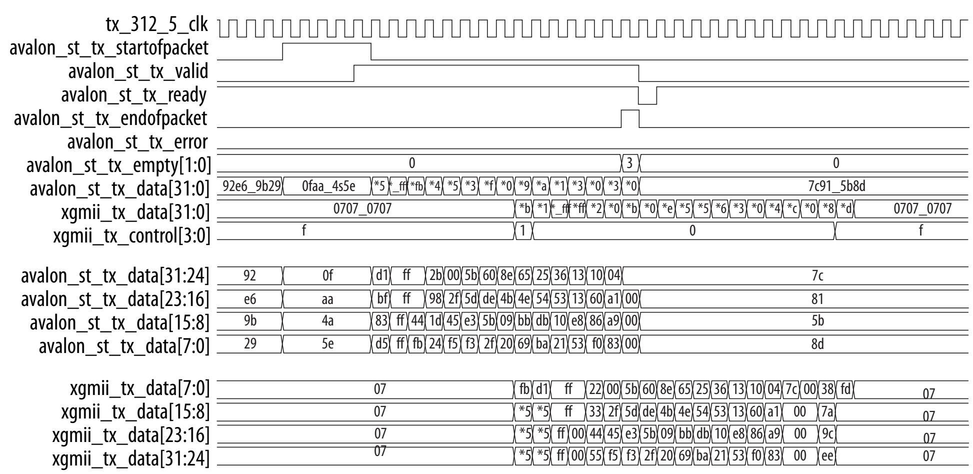


Figure 13. Normal Frame with Preamble Passthrough Mode, Padding Bytes Insertion, and Source Address Insertion Enabled

The following diagram shows the transmission of good frames with preamble passthrough mode, padding bytes insertion, and source address insertion enabled.



(2) At least a full column of IDLE (four IDLE characters) must precede the remote fault sequence.

Figure 14. Back-to-back Transmission of Normal Frames with Source Address Insertion Enabled.

The following diagram shows back-to-back transmission of normal frames with source address insertion enabled. The MAC primary address registers are set to 0x000022334455.

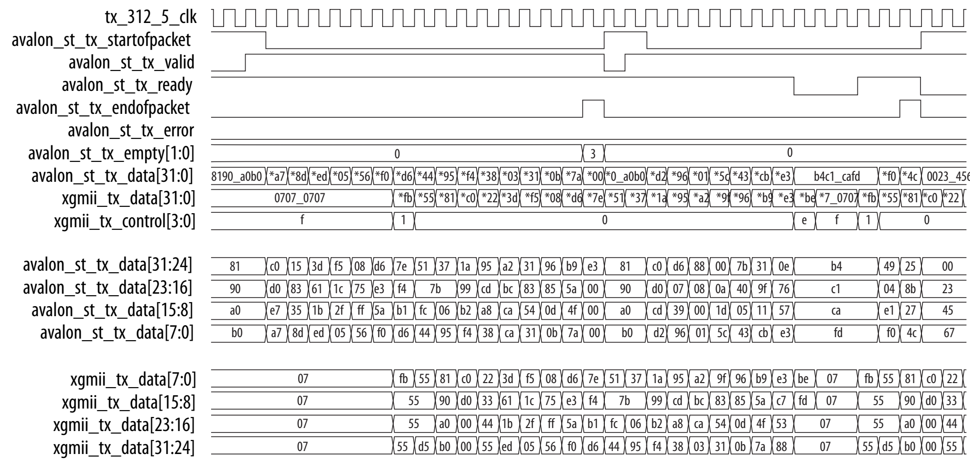


Figure 15. Back-to-back Transmission of Normal Frames with Preamble Passthrough Mode Enabled

The following diagram shows back-to-back transmission of normal frames with preamble passthrough mode enabled.

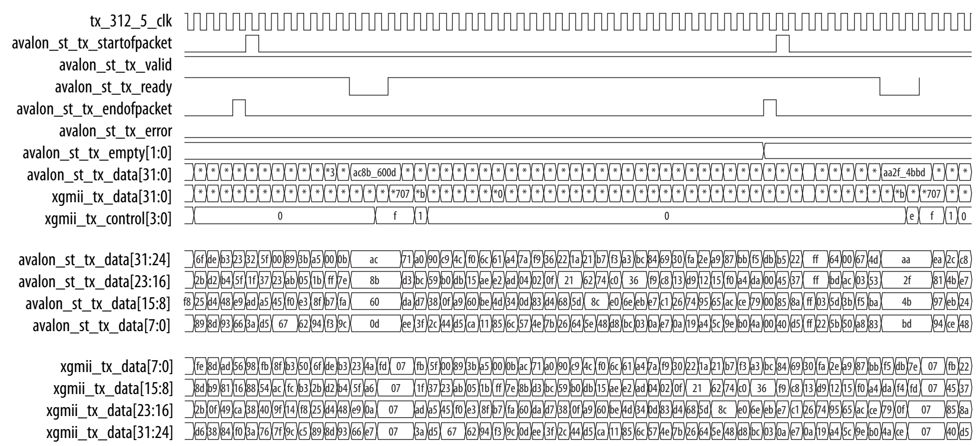
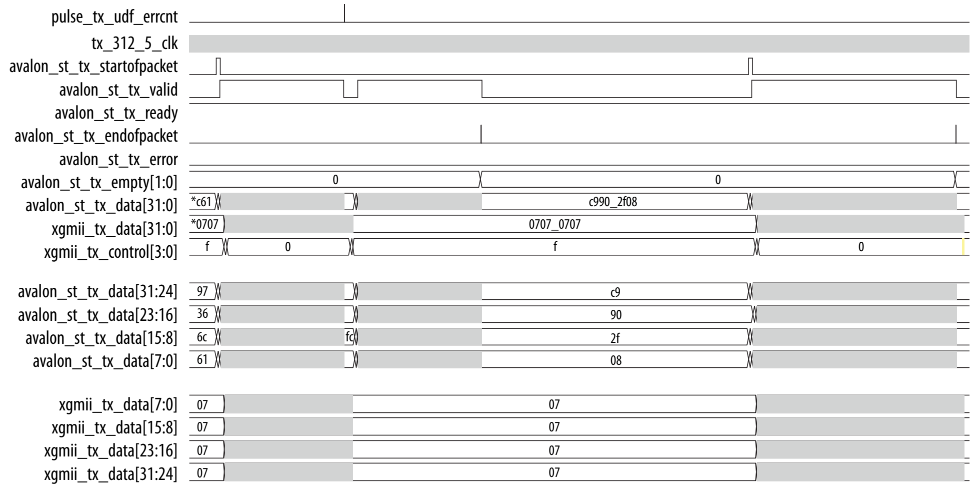


Figure 16. Error Condition—Underflow

The following diagrams show an underflow on the transmit datapath followed by the transmission of a normal frame.



An underflow happens in the middle of a frame that results in a premature termination on the XGMII. The remaining data from the Avalon-ST transmit interface is still received after the underflow but the data is dropped. The transmission of the next frame is not affected by the underflow.

Figure 17. Error Condition—Underflow, continued

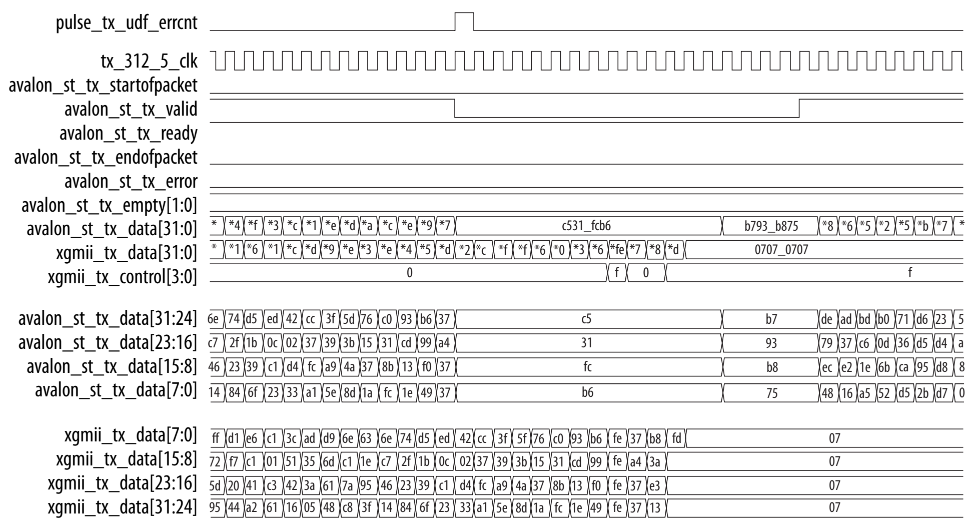
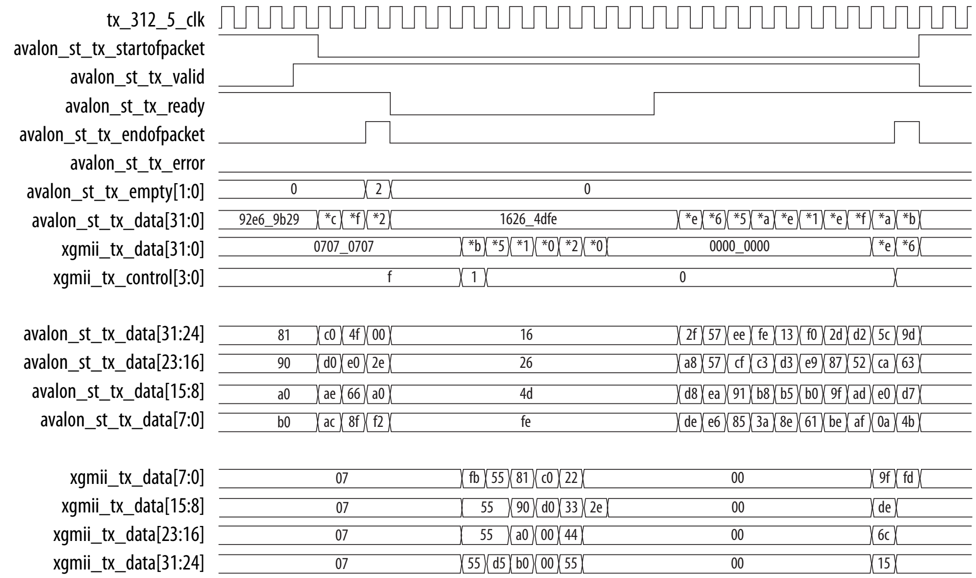


Figure 18. Short Frame with Padding Bytes Insertion Enabled

The following diagram shows the transmission of a short frame with no payload data. Padding bytes insertion is enabled.

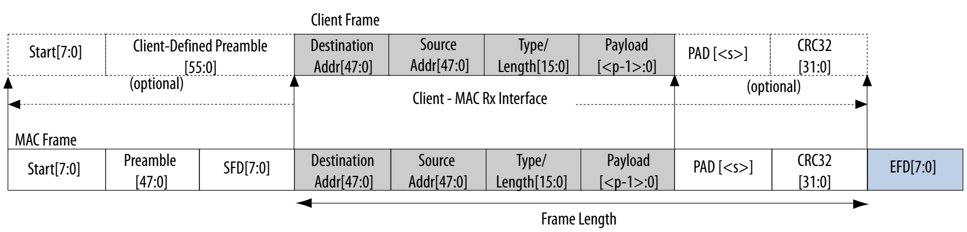


3.5 RX Datapath

The MAC RX receives Ethernet frames from the XGMII and forwards the payload with relevant frame fields to the client after performing checks and filtering invalid frames. Some frame fields are optionally removed from the frame before MAC RX forwards the frame to the client.

The following figure shows the typical flow of frame through the MAC RX.

Figure 19. Typical Client Frame at Receive Interface



3.5.1 XGMII Decapsulation

The MAC RX expects the first byte of receive packets to be in lane 0, `xgmii_rx_data[7:0]`. If the 32-bit/64-bit adapter on the XGMII is present, the first byte of receive packets must be in lane 0 or lane 4, `xgmii_rx_data[39:32]`. Receive packets must also be preceded by a column of idle bytes or an ordered set such as a local fault. Packets that do not satisfy these conditions are invalid and the MAC RX drops them.

By default, the MAC RX only accepts packets that begin with a 1-byte START, 6-byte preamble, and 1-byte SFD. Packets that do not satisfy this condition are invalid and the MAC RX drops them.

When you enable the preamble passthrough mode (`rx_preamble_control` register = 1), the MAC RX only checks packets that begin with a 1-byte START. In this mode, the MAC RX does not remove the START and custom preamble, but passes the bytes along with the frame to the client.

After examining the packet header bytes in the correct order, the MAC IP retrieves the frame data from the packet. If the frame data starting from the destination address field is less than 17 bytes, the MAC IP may or may not drop the frame. If the erroneous frame is not dropped but forwarded, an undersized error is flagged to the external logic to drop the frame. If the frame is more than 17 bytes, the MAC forwards the frame as normal and flags error whenever applicable.

3.5.2 CRC Checking

The MAC RX computes the CRC-32 checksum over frame bytes received and compares the computed value against the CRC field in the receive frame. If the values do not match, the MAC RX marks the frame invalid by setting `avalon_st_rx_error[1]` to 1 and forwards the receive frame to the client. When the CRC error indicator is asserted, the external logic is expected to drop the frame bytes.

3.5.3 Address Checking

The MAC RX 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 RX always accepts broadcast frames. By default, the MAC RX also receives all unicast and multicast frames unless configured otherwise in the EN_ALLUCAST and EN_ALLMCAST bits of the `rx_frame_control` register.

When the EN_ALLUCAST bit is set to 0, the MAC RX filters unicast frames received. The MAC RX accepts only unicast frames with a destination address that matches the primary MAC address specified in the `primary_mac_addr0` and `primary_mac_addr1` registers. If any of the supplementary address bits are set to 1 (EN_SUPP0/1/2/3 in the `rx_frame_control` register), the MAC RX also checks the destination address against the supplementary addresses in the `rx_frame_spaddr*_*` registers.

When the EN_ALLMCAST bit is set to 0, the MAC RX drops all multicast frames. This condition does not apply to global multicast pause frames.

3.5.4 Frame Type Checking

The MAC RX 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 RX 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 RX continues to check the frame and payload lengths.
  - Length/type = 0x8808—Control frames. The next two bytes are the Opcode field which indicates the type of control frame. For pause frames (Opcode = 0x0001) and PFC frames (Opcode = 0x0101), the MAC RX proceeds with pause frame processing. By default, the MAC RX drops all control frames. If configured otherwise (`FWD_CONTROL` bit in the `rx_frame_control` register = 1), the MAC RX forwards control frames to the client.
  - For other field values, the MAC RX forwards the receive frame to the client.

Table 14. MAC Behavior for Different Frame Types

Category	Packet Size	Length/Type = Payload	Length/Type > Payload	Length/Type < Payload	MAC Behavior	
					Frame Drop	avalon_st_rx_error[]
Normal packet	65–1518	Yes	No	No	No	—
		No	No	Yes	No	
		No	Yes	No	No	avalon_st_rx_error[4] = 1
Undersized	Packet < 64	Yes	No	No	No	avalon_st_rx_error[2] = 1
		No	No	Yes	No	
		No	Yes	No	No	avalon_st_rx_error[2] = 1 avalon_st_rx_error[4] = 1
Oversized	1518 < Packet < 1535	Yes	No	No	No	avalon_st_rx_error[3] = 1
		No	No	Yes	No	
		No	Yes	No	No	avalon_st_rx_error[3] = 1 avalon_st_rx_error[4] = 1

Related Links

Avalon-ST RX Data Interface Signals on page 92

3.5.5 Length Checking

The MAC RX checks the frame and payload lengths of basic, VLAN tagged, and stacked VLAN tagged frames. The MAC RX does not drop frames with invalid length but sets the error bits accordingly.

3.5.5.1 Frame Length

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

- Basic—The value in the rx_frame_maxlength register.
- VLAN tagged—The value in the rx_frame_maxlength register plus four bytes when the rx_vlan_detection[0] register bit is 0; or the value in the rx_frame_maxlength register when the rx_vlan_detection[0] register bit is set to 1.
- Stacked VLAN tagged—The value in the rx_frame_maxlength register plus eight bytes when the rx_vlan_detection[0] register bit is 0; or the value in the rx_frame_maxlength register when the rx_vlan_detection[0] register bit is set to 1.

The following error bits represent frame length violations:

- avalon_st_rx_error[2]—undersized frames.
- avalon_st_rx_error[3]—oversized frames.

3.5.5.2 Payload Length

The MAC IP core checks the payload length for frames other than control frames when the VLAN and stacked VLAN detection is disabled. The MAC RX keeps track of the actual payload length upon receiving a frame and checks the actual payload length against the length/type or client length/type field. The payload length must be between 46 (0x2E) and 1500 (0x5DC). For VLAN and stacked VLAN frames, the minimum payload length is 42 (0x2A) or 38 (0x26) respectively and not exceeding the maximum value of 1500 (0x5DC).

For an invalid payload length, the MAC RX sets the avalon_st_rx_error[4] bit to 1. This error occurs when the actual payload length is less than the value of the length/type field. If the actual payload length is more than the value of the length/type field, the MAC RX assumes that the frame contains excessive padding and does not set this error bit to 1. If the value of the length/type field is more than the actual payload length, the MAC RX sets the avalon_st_rx_error[4] bit to 1, and the packet content will not be accurate.

3.5.6 CRC and Padding Bytes Removal

By default, the MAC RX forwards receive frames to the client without removing the CRC field and padding bytes from the frames. You can configure the MAC RX to remove the CRC field by setting the rx_padcrc_control register to 1. To remove both the CRC field and padding bytes, set the rx_padcrc_control register to 3.

When enabled, the MAC RX removes padding bytes from receive frames whose payload length is less than the following values for the different frame types:

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

The MAC RX removes padding bytes only when the VLAN and stacked VLAN detection is enabled (`rx_vlan_detection[0]` = 0). Otherwise, the MAC RX does not remove padding bytes even if padding bytes removal is enabled.

3.5.7 Overflow Handling

When an overflow occurs on the client side, the client can backpressure the Avalon-ST receive interface by deasserting the `avalon_st_rx_ready` signal. If an overflow occurs, the MAC RX sets the error bit, `avalon_st_rx_error[5]`, to 1 to indicate an overflow. The MAC RX drops subsequent frames if the overflow condition persists. The MAC RX then continues to receive data when the overflow condition ceases.

3.5.8 RX Timing Diagrams

Figure 20. Back-to-back Transmission of Normal Frames with CRC Removal Enabled

The following diagram shows back-to-back reception of normal frames with CRC removal enabled.

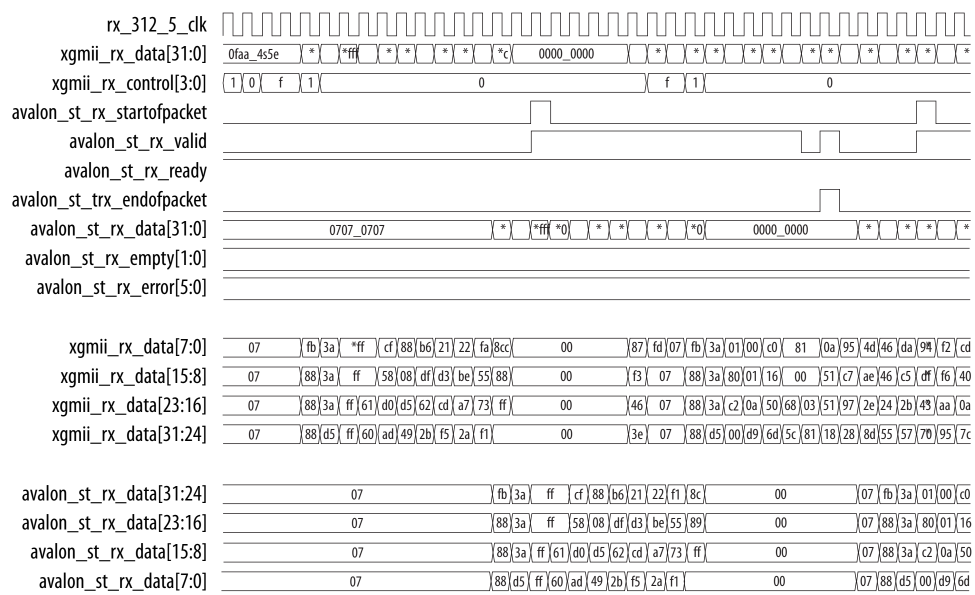
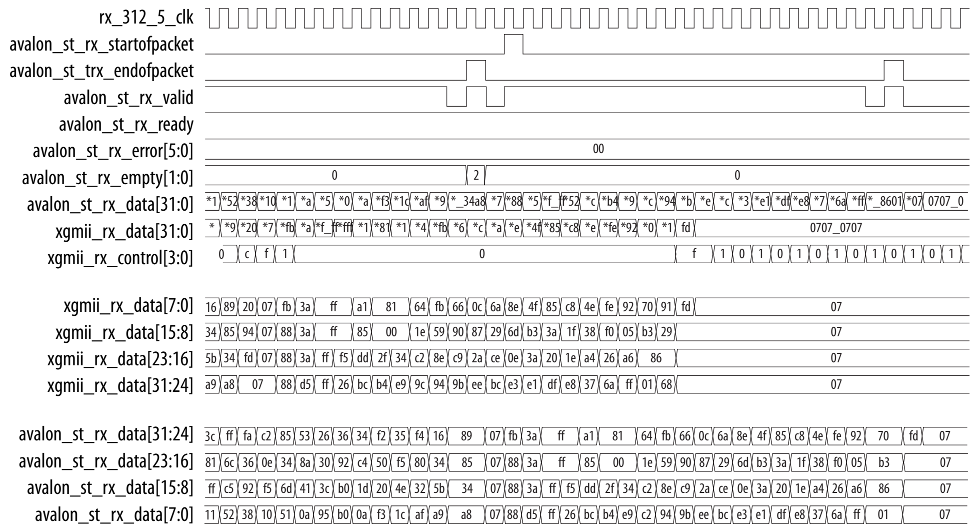


Figure 21. Back-to-back Transmission of Normal Frames with Preamble Passthrough Mode Enabled

The following diagram shows back-to-back reception of normal frames with preamble passthrough mode and padding bytes and CRC removal enabled.



3.6 Flow Control

The MAC IP core implements the following flow control mechanisms:

- IEEE 802.3 flow control—implements the IEEE 802.3 Annex 31B standard to manage congestion. When the MAC IP core experiences congestion, the core sends a pause frame to request its link partner to suspend transmission for a given period of time. This flow control is a mechanism to manage congestion at the local or remote partner. When the receiving device experiences congestion, it sends an XOFF pause frame to the emitting device to instruct the emitting device to stop sending data for a duration specified by the congested receiver. Data transmission resumes when the emitting device receives an XON pause frame (pause quanta = zero) or when the timer expires.
- Priority-based flow control (PFC)—implements the IEEE 802.1Qbb standard. PFC manages congestion based on priority levels. It supports up to 8 priority queues. When the receiving device experiences congestion on a priority queue, it sends a PFC frame requesting the emitting device to stop transmission on the priority queue for a duration specified by the congested receiver. When the receiving device is ready to receive transmission on the priority queue again, it sends a PFC frame instructing the emitting device to resume transmission on the priority queue.

Note: Intel recommends that you enable only one type of flow control at any one time.

3.6.1 IEEE 802.3 Flow Control

To use the IEEE 802.3 flow control, set the following registers:

- On the TX datapath:
  - Set `tx_pfc_priority_enable[7:0]` to 0 to disable the PFC. The rest of the bits are unused.
  - Set `tx_pauseframe_enable[0]` to 1 to enable the IEEE 802.3 flow control.
- On the RX datapath:
  - Set `rx_pfc_control[7:0]` to 1 to disable the PFC. The rest of the bits are mostly unused.
  - Set the `IGNORE_PAUSE` bit in the `rx_frame_control` register to 0 to enable the IEEE 802.3 flow control.

3.6.1.1 Pause Frame Reception

When the MAC receives an XOFF pause frame, it stops transmitting frames to the remote partner for a period equal to the pause quanta field of the pause frame. If the MAC receives a pause frame in the middle of a frame transmission, the MAC finishes sending the current frame and then suspends transmission for a period specified by the pause quanta. The MAC resumes transmission when it receives an XON pause frame or when the timer expires. The pause quanta received overrides any counter currently stored. When the remote partner sends more than one pause quanta, the MAC sets the value of the pause to the last quanta it received from the remote partner. You have the option to configure the MAC to ignore pause frames and continue transmitting frames by setting the `IGNORE_PAUSE` bit in the `rx_frame_control` register to 1.

3.6.1.2 Pause Frame Transmission

Use one of the following methods to trigger pause frame transmission:

- avalon_st_pause_data signal (tx_pauseframe_enable[2:1] set to 0)—You can connect this 2-bit signal to a FIFO buffer or a client. Bit setting:
  - avalon_st_pause_data[1]: 1—triggers the transmission of XOFF pause frames.
  - avalon_st_pause_data[0]: 1—triggers the transmission of XON pause frames. The transmission of XON pause frames only trigger for one time after XOFF pause frames regardless of how long the avalon_st_pause_data[0] signal is asserted.

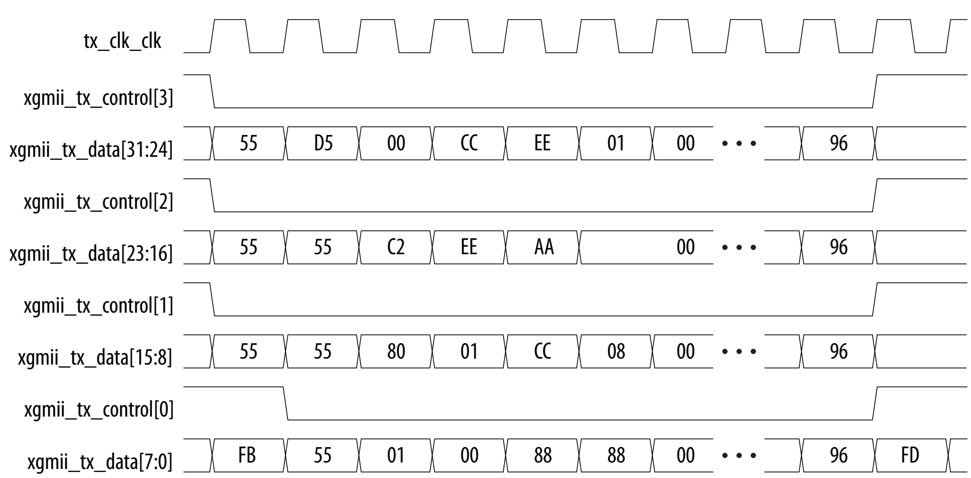
  If pause frame transmission is triggered when the MAC is generating a pause frame, the MAC ignores the incoming request and completes the generation of the pause frame. Upon completion, if the avalon_st_pause_data signal remains asserted, the MAC generates a new pause frame and continues to do so until the signal is deasserted. You can also configure the gap between successive XOFF requests for using the tx_pauseframe_quanta register. XON pause frames will only be generated if the MAC generates XOFF pause frames.
- tx_pauseframe_control register (tx_pauseframe_enable[2:0] set to 0x1)—A host (software) can set this register to trigger pause frames transmission. Setting tx_pauseframe_control[1] to 1 triggers the transmission of XOFF pause frames; setting tx_pauseframe_control[0] to 1 triggers the transmission of XON pause frames. The register clears itself after the request is executed.

You can configure the pause quanta in the tx_pauseframe_quanta register. The MAC sets the pause quanta field in XOFF pause frames to this register value.

Note: The new register field determines which pause interface takes effect.

The following figure shows the transmission of an XON pause frame. The MAC sets the destination address field to the global multicast address, 01-80-C2-00-00-01 (0x010000c28001) and the source address to the MAC primary address configured in the tx_addrins_macaddr0 and tx_addrins_madaddr1 registers.

Figure 22. XON Pause Frame Transmission



3.6.2 Priority-Based Flow Control

Follow these steps to use the PFC:

1. Turn on the **Priority-based flow control (PFC)** parameter and specify the number of priority levels using the **Number of PFC priorities** parameter. You can specify between 2 to 8 PFC priority levels.
2. Set the following registers.
   - On the TX datapath:
     — Set `tx_pauseframe_enable` to 0 to disable the IEEE 802.3 flow control.
     — Set `tx_pfc_priority_enable[n]` to 1 to enable the PFC for priority queue *n*.
   - On the RX datapath:
     — Set the `IGNORE_PAUSE` bit in the `rx_frame_control` register to 1 to disable the IEEE 802.3 flow control.
     — Set the `rx_pfc_control[7:0]` register bits to 0 to enable the PFC. Most of the rest of the bits are unused.
3. Connect the `avalon_st_tx_pfc_gen_data` signal to the corresponding RX client logic and the `avalon_st_rx_pfc_pause_data` signal to the corresponding TX client logic.
4. You have the option to configure the MAC RX to forward the PFC frame to the client by setting the `rx_pfc_control[16]` register to 1. By default, the MAC RX drops the PFC frame after processing it.

3.6.2.1 PFC Frame Reception

When the MAC RX receives a PFC frame from the remote partner, it asserts the `avalon_st_rx_pfc_pause_data[n]` signal if Pause Quanta n is valid (Pause Quanta Enable [n] = 1) and greater than 0. The client suspends transmission from the TX priority queue n for the period specified by Pause Quanta n. If the MAC RX asserts the `avalon_st_rx_pfc_pause_data[n]` signal in the middle of a client frame transmission for the TX priority queue n, the client finishes sending the current frame and then suspends transmission for the queue.

When the MAC RX receives a PFC frame from the remote partner, it deasserts the `avalon_st_rx_pfc_pause_data[n]` signal if Pause Quanta n is valid (Pause Quanta Enable [n] = 1) and equal to 0. The MAC RX also deasserts this signal when the timer expires. The client resumes transmission for the suspended TX priority queue when the `avalon_st_rx_pfc_pause_data[n]` signal is deasserted.

When the remote partner sends more than one pause quanta for the TX priority queue n, the MAC RX sets the pause quanta n to the last pause quanta received from the remote partner.

3.6.2.2 PFC Frame Transmission

PFC frame generation is triggered through the `avalon_st_tx_pfc_gen_data` signal. Set the respective bits to generate XOFF or XON requests for the priority queues.

For XOFF requests, you can configure the pause quanta for each priority queue using the `pfc_pause_quanta_n` registers. For an XOFF request for priority queue n, the MAC TX sets bit n in the Pause Quanta Enable field to 1 and the Pause Quanta n field to the value of the `pfc_pause_quanta_n` register. You can also configure the gap between successive XOFF requests for a priority queue using the `pfc_holdoff_quanta_n` register.

For XON requests, the MAC TX sets the pause quanta to 0. You must generate a XOFF request before generating a XON request.

3.7 Reset Requirements

The MAC IP core consists of the following reset domains:

- CSR reset—global reset,
- MAC TX reset, and
- MAC RX reset.

These resets are asynchronous events. When the MAC or any part of it goes into reset, the user application must manage possible asynchronous changes to the states of the MAC interface signals. The MAC does not guarantee any reset sequence. Intel recommends the sequence shown in the following diagram and table for CSR reset, and TX and RX datapaths reset respectively.

Figure 23. CSR Reset

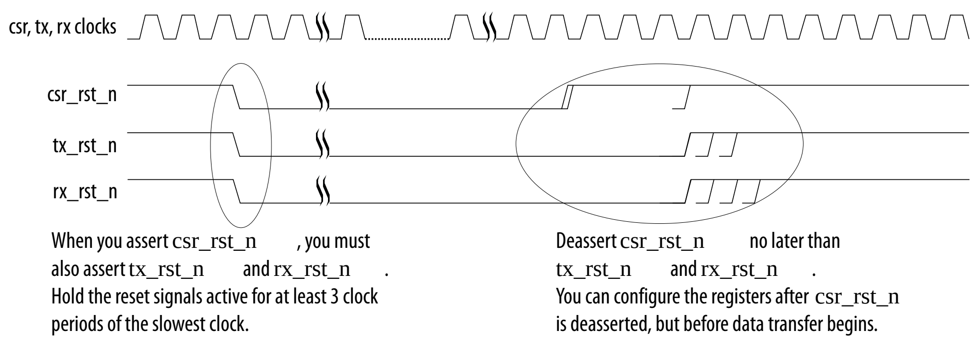


Table 15. TX and RX Datapaths Reset

No	Stage	Steps
1	Ensure no data transfer in progress.	1. Set the `tx_packet_control[0]` bit to 1 to disable the TX datapath; the `rx_transfer_control[0]` bit to disable the RX datapath. 2. Check the `tx_transfer_status[8]` bit for a value of 0 to ensure that no TX data transfer is in progress; the `rx_transfer_status[8]` bit for RX path. Alternatively, wait for a period of time.
2	Trigger reset.	1. Ensure that the respective TX and RX clocks are stable. 2. Assert the `tx_rst_n` signal or the `rx_rst_n` signal to reset the MAC TX or MAC RX respectively. You can also trigger the reset by setting the `mac_reset_control[0]` bit or the `mac_reset_control[8]` bit to 1 to reset the MAC TX or MAC RX respectively. 3. Hold the reset signal active for at least three clock cycles.
3	Stop reset.	1. Release the reset signal only when the clocks are stable. 2. Wait for at least 500 ns to ensure the reset is fully complete. 3. Clear the statistics counters.
4	Resume data transfer.	1. Clear the `tx_packet_control[0]` bit to enable the TX datapath; the `rx_transfer_control[0]` bit to enable the RX datapath.

3.8 Supported PHYs

You can connect the LL 10GbE MAC IP core to a PHY IP core using XGMII, GMII, or MII interfaces.

Table 16. Supported PHYs

Operating Mode	PHY
10G	10GBASE-R PHY, XAUI PHY
1G/10G	10GBASE-KR or 1G/10G PHY
10M/100M/1G/10G	
1G/2.5G	1G/2.5G/10G Multi-rate Ethernet PHY
1G/2.5G/10G	
10M/100M/1G/2.5G	1G/2.5G/10G Multi-rate Ethernet PHY (with SGMII bridge enabled)
10M/100M/1G/2.5G/10G	

To connect the MAC IP core to 64-bit PHYs, ensure that you enable the **Use legacy Ethernet 10G MAC XGMII Interface** option.

Related Links

AN 701: Scalable Low Latency Ethernet 10G MAC using Intel Arria 10 1G/10G PHY
design examples to demonstrate the Low Latency Ethernet 10G MAC IP systems using Intel Arria 10 PHY.

3.8.1 10GBASE-R Register Mode

The MAC IP core supports this feature for use with the Intel Arria 10, Intel Cyclone 10 GX, and Intel Stratix 10 Transceiver Native PHY IP core preset configurations. When operating in this mode, the round-trip latency for the MAC and PHY is reduced to 140 ns (for Intel Arria 10 and Intel Cyclone 10 GX devices) or 168 ns (for Intel Stratix 10 devices) with a slight increase in resource count and clock frequencies.

When you enable this feature, the MAC IP core implements two additional signals to determine the validity of the data on the TX and RX XGMII. These signals, `xgmii_tx_valid` and `xgmii_rx_valid`, ensure that the effective data rate of the MAC is 10 Gbps. You must also observe the following guidelines when using the register mode:

- For Intel Arria 10 and Intel Cyclone 10 GX devices, the selected preset is **10GBASE-R Register Mode**.
- For Intel Stratix 10 devices, the selected preset is **10GBASE-R 1588**.
- The PHY must expose the TX and RX parallel clocks.
- The PHY must expose data valid signals, with MAC/PHY TX/RX interfaces in register mode, as in the IEEE 1588v2 configuration.
- The MAC and PHY run at the parallel clock frequency of 322.265625 MHz (the PCS/PMA width equals to 32).

Figure 24. PHY Configuration with 10GBASE-R Register Mode Enabled for Intel Arria 10 and Intel Cyclone 10 GX Devices

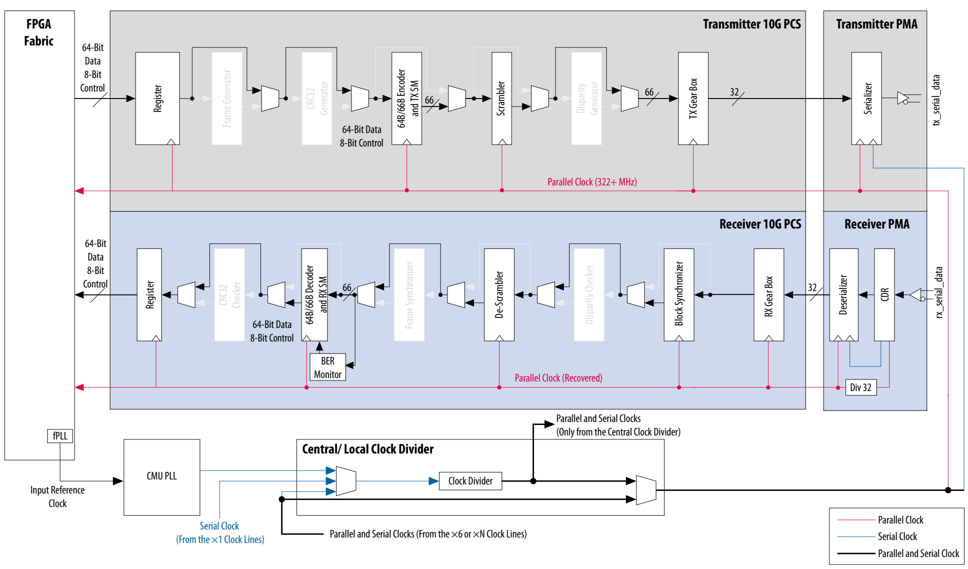
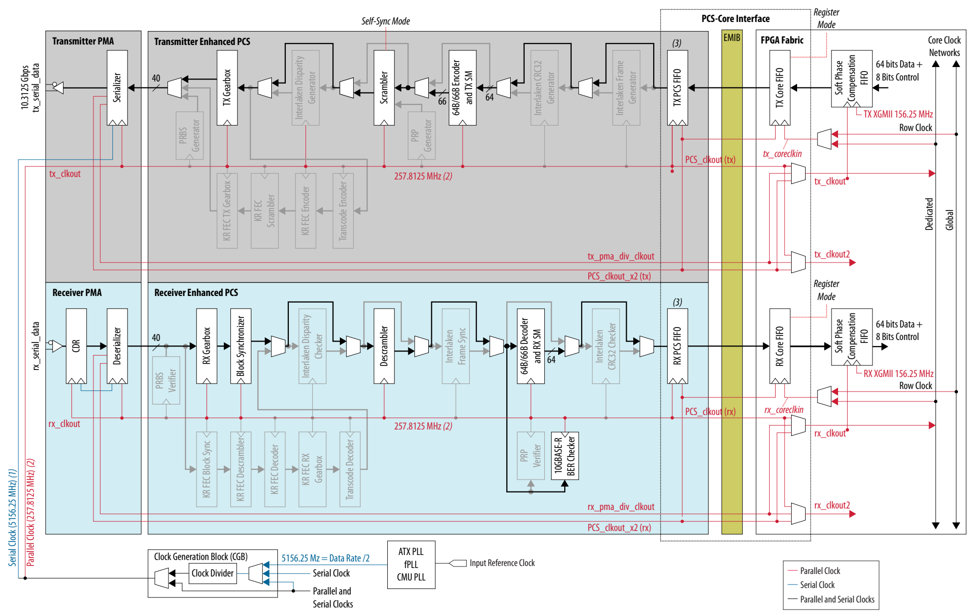


Figure 25. PHY Configuration with 10GBASE-R with IEEE 1588v2 Enabled for Intel Stratix 10 Devices



Notes:
1. Value based on the clock division factor chosen.
2. Value calculated as data rate/PCS-PMA interface width.
3. This block is in phase measuring FIFO mode for the 10GBASE-R with 1588 configuration.

Related Links

- Intel Arria 10 Transceiver PHY User Guide
  More information on how to configure the transceivers to implement 10GBASE-R functionality by using the preset of the Intel Arria 10 Transceiver Native IP core.
- Intel Cyclone 10 GX Transceiver PHY User Guide
  More information on how to configure the transceivers to implement 10GBASE-R functionality by using the preset of the Intel Cyclone 10 Transceiver Native IP core.
- Intel Stratix 10 L- and H-Tile Transceiver PHY User Guide
  More information on how to configure the transceivers to implement 10GBASE-R functionality by using the preset of the Intel Stratix 10 L- and H-Tile Transceiver Native IP cores.

3.9 XGMII Error Handling (Link Fault)

The LL Ethernet 10G MAC supports link fault generation and detection.

When the MAC RX receives a local fault, the MAC TX starts sending remote fault status (0x9c000002) on the XGMII. If the packet transmission was in progress at the time, the remote fault bytes will override the packet bytes until the fault condition ceases.

When the MAC RX receives a remote fault, the MAC TX starts sending IDLE bytes (0x07070707) on its XGMII. If packet transmission was in progress at the time, the IDLE bytes will override the packet bytes until the fault condition ceases.

The MAC considers the link fault condition has ceased if the client and the remote partner both receive valid data in more than 127 columns.

Figure 26. Fault Signaling

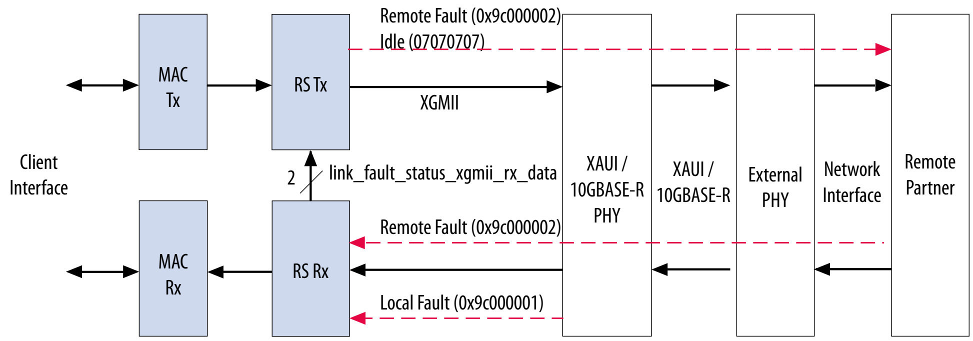
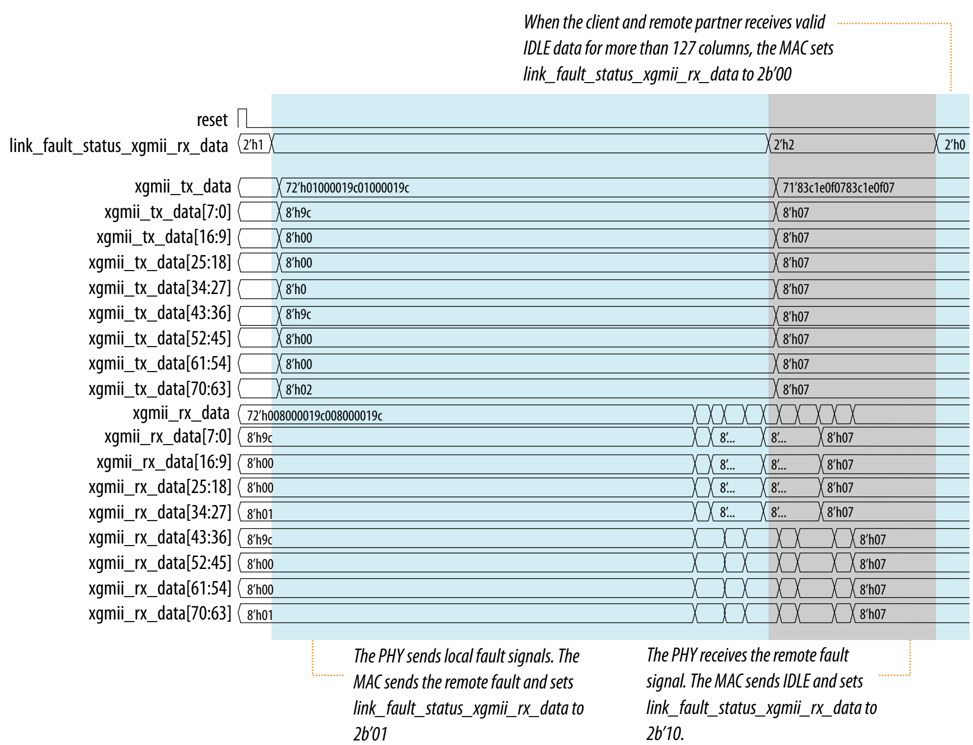


Figure 27. XGMII TX interface Transmitting Remote Fault Signal

Figure shows the timing for the XGMII TX interface transmitting the remote fault signal.



When you instantiate the MAC RX only variation, connect the `link_fault_status_xgmii_rx_data` signal to the corresponding RX client logic to handle the link fault. Similarly, when you instantiate the MAC TX only variation, connect the `link_fault_status_xgmii_tx_data` signal to the corresponding TX client logic.

3.10 IEEE 1588v2

The IEEE 1588v2 option provides time stamp for receive and transmit frames in the LL Ethernet 10G MAC IP core designs. The feature consists of Precision Time Protocol (PTP). PTP is a protocol that accurately synchronizes all real time-of-day clocks in a network to a master clock.

The IEEE 1588v2 option has the following 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 with the following 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 random error:
  - 10 Gbps—Timestamp accuracy of ± 1 ns
  - 2.5 Gbps—Timestamp accuracy of ± 2 ns
  - 1 Gbps—Timestamp accuracy of ± 2 ns
  - 100 Mbps—Timestamp accuracy of ± 5 ns
- Supports static error of ± 3 ns across all speeds.
- Supports IEEE 802.3, UDP/IPv4, and UDP/IPv6 protocol encapsulations for the PTP packets.
- Supports untagged, VLAN tagged, and Stacked VLAN Tagged PTP packets, and any number of MPLS labels. The packet classifier under user control parses the packet (Ethernet packet or MPLS packet) and gives the IP core the required offset, at which either the Time of Day (ToD) or correction factor (CF) update can happen.
- Supports configurable register for timestamp correction on both transmit and receive datapaths.
- Supports ToD clock that provides streams of 64-bit and 96-bit timestamps. The 64-bit timestamp is for transparent clock devices and the 96-bit timestamp is for ordinary clock and boundary clock devices.

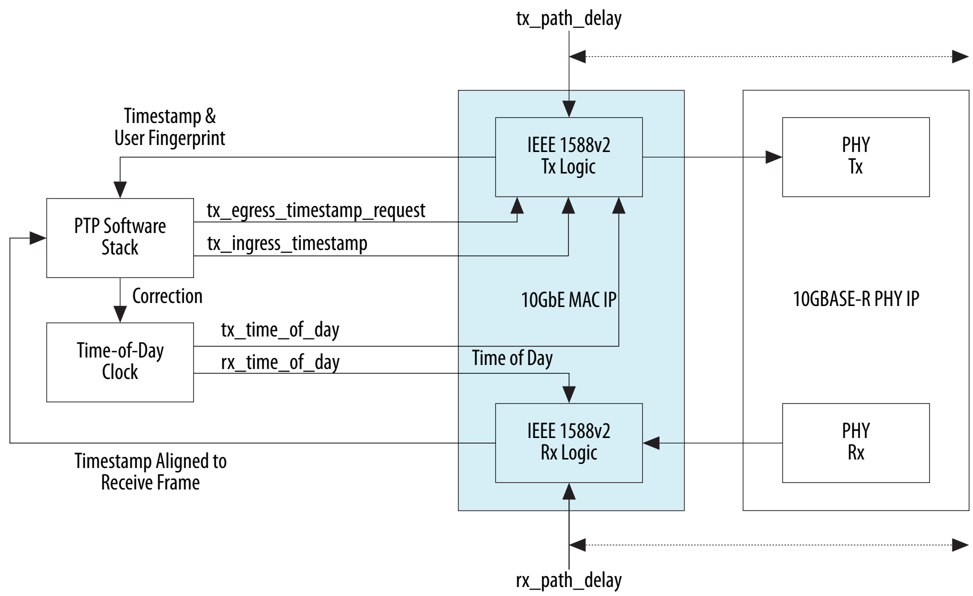
Related Links

Intel 1588 System Solution
Describes the implementation of the IEEE 1588v2 feature.

3.10.1 Architecture

The following figure shows the overview of the IEEE 1588v2 feature.

Figure 28. Overview of IEEE 1588v2 Feature



3.10.2 TX Datapath

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

- For 1-step clock synchronization,
  - Timestamp insertion depends on the PTP device and message type.
  - The MAC function inserts a timestamp in the PTP packet when the client specifies the Timestamp field offset and asserts Timestamp Insert Request.
  - Depending on the PTP device and message type, the MAC function updates the residence time in the correction field of the PTP packet when the client asserts `tx_etstamp_ins_ctrl_residence_time_update` and Correction Field Update. The residence time is the difference between the egress and ingress timestamps.
  - For PTP packets encapsulated using the UDP/IPv6 protocol, the MAC function performs UDP checksum correction using extended bytes in the PTP packet.
  - The MAC function recomputes and reinserts CRC-32 into PTP packets each time the timestamp or correction field is updated, even when CRC insertion is disabled using the `tx_crc_control[1]` register bit.
  - The format of timestamp supported includes 1588v1 and 1588v2.
  - The MAC function update `tx_egress_p2p_val[45:0]` into correction field of PTP packet when the client assert `tx_egress_p2p_update`.
  - The MAC function update asymmetry value into correction field of PTP packet when the client assert `tx_egress_asymmetry_update`. The asymmetry value is retrieved from configuration registers.
- For 2-step clock synchronization, the MAC function returns the timestamp and the associated fingerprint for all TX frames when the client asserts `tx_egress_timestamp_request_valid`.

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

Table 17. Timestamp and Correction Insertion for 1-Step Clock Synchronization

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 Correcti on
Sync	Yes (3)	No	Yes(3)	No	No	Yes (4)	No	Yes (4)
Delay_Req	No	No	No	No	No	Yes (4)	No	Yes (4)
Pdelay_Req	No	No	No	No	No	Yes (4)	No	No
Pdelay_Resp	No	Yes (3) (4)	No	Yes (3) (4)	No	Yes (4)	No	Yes (3) (4)

continued...

(3) Applicable only when 2-step flag in `flagField` of the PTP packet is 0.

(4) Applicable when you assert the `tx_etstamp_ins_ctrl_residence_time_update` signal.

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 Correcti on
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

3.10.3 RX Datapath

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

3.10.4 Frame Format

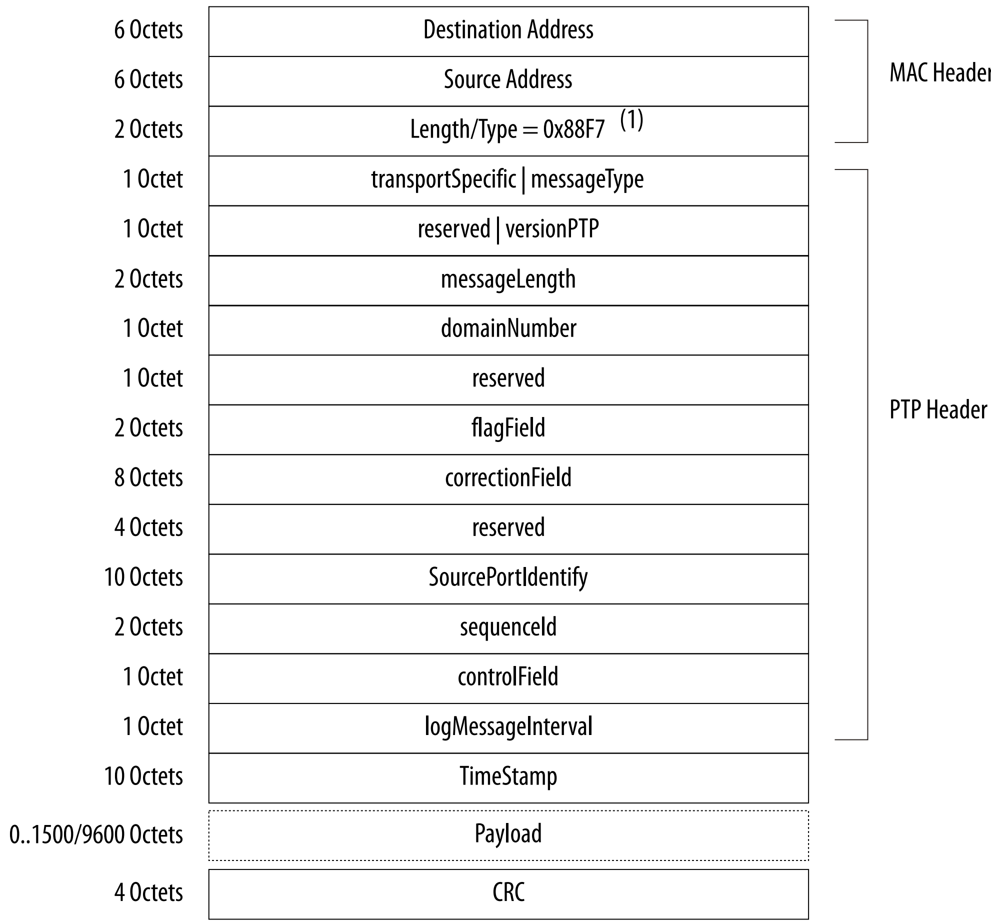
The MAC function, with the IEEE 1588v2 feature, supports PTP packet transfer for the following transport protocols:

- IEEE 802.3
- UDP/IPv4
- UDP/IPv6

3.10.4.1 PTP Packet in IEEE 802.3

The following figure shows the format of the PTP packet encapsulated in IEEE 802.3.

Figure 29. PTP Packet in IEEE 802.3



Note:
(1) For packets with VLAN or Stacked VLAN tag, add 4 or 8 octets offsets before the length/type field.

3.10.4.2 PTP Packet over UDP/IPv4

The following figure shows the format of the PTP packet encapsulated in UDP/IPv4. Checksum calculation is optional for the UDP/IPv4 protocol. The 1588v2 TX logic should set the checksum to zero.

Figure 30. PTP Packet over UDP/IPv4

6 Octets	Destination Address	MAC Header
6 Octets	Source Address	
2 Octets	Length/Type = 0x0800 (1)	
1 Octet	Version | Internet Header Length	IP Header
1 Octet	Differentiated Services	
2 Octets	Total Length	
2 Octets	Identification	
2 Octets	Flags | Fragment Offsets	
1 Octet	Time To Live	
1 Octet	Protocol = 0x11	
2 Octets	Header Checksum	
4 Octets	Source IP Address	
4 Octets	Destination IP Address	
0 Octet	Options | Padding	
2 Octets	Source Port	UDP Header
2 Octets	Destination Port = 319 / 320	
2 Octets	Length	
2 Octets	Checksum	
1 Octet	transportSpecific | messageType	PTP Header
1 Octet	reserved | versionPTP	
2 Octets	messageLength	
1 Octet	domainNumber	
1 Octet	reserved	
2 Octets	flagField	
8 Octets	correctionField	
4 Octets	reserved	
10 Octets	SourcePortIdentify	
2 Octets	sequenceId	
1 Octet	controlField	
1 Octet	logMessageInterval	
10 Octets	TimeStamp	
0..1500/9600 Octets	Payload	
4 Octets	CRC	

Note:
(1) For packets with VLAN or Stacked VLAN tag, add 4 or 8 octets offsets before the length/type field.

3.10.4.3 PTP Packet over UDP/IPv6

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

Figure 31. PTP Packet over UDP/IPv6

6 Octets	Destination Address	MAC Header
6 Octets	Source Address	
2 Octets	Length/Type = 0x86DD (1)	
4 Octet	Version | Traffic Class | Flow Label	IP Header
2 Octets	Payload Length	
1 Octet	Next Header = 0x11	
1 Octet	Hop Limit	
16 Octets	Source IP Address	
16 Octets	Destination IP Address	
2 Octets	Source Port	UDP Header
2 Octets	Destination Port = 319 / 320	
2 Octets	Length	
2 Octets	Checksum	
1 Octet	transportSpecific | messageType	PTP Header
1 Octet	reserved | versionPTP	
2 Octets	messageLength	
1 Octet	domainNumber	
1 Octet	reserved	
2 Octets	flagField	
8 Octets	correctionField	
4 Octets	reserved	
10 Octets	SourcePortIdentify	
2 Octets	sequenceId	
1 Octet	controlField	
1 Octet	logMessageInterval	
10 Octets	TimeStamp	
0..1500/9600 Octets	Payload	
2 Octets	extended bytes	
4 Octets	CRC	

Note:
(1) For packets with VLAN or Stacked VLAN tag, add 4 or 8 octets offsets before the length/type field.

4 Configuration Registers

The LL Ethernet 10G MAC IP core provides a total of 4Kb register space that is accessible via the Avalon-MM interface. Each register is 32 bits wide. Access only registers that apply to the variation of the MAC IP core you are using and enabled features. For example, if you are using the MAC RX only variation, avoid accessing registers specific to the MAC TX only variation. Accessing reserved registers or specific registers to variations that you are not using may produce non-deterministic behavior.

4.1 Register Map

Table 18. Register Map

Word Offset	Purpose	Variation
0x0000: 0x000F	Reserved	—
0x0010: 0x0011	Primary MAC Address	MAC TX, MAC RX
0x0012: 0x001D	Reserved	—
0x001F	MAC Reset Control Register	
0x0020: 0x003F	TX Configuration and Status Registers	MAC TX
0x0040: 0x005F	TX Flow Control Registers	MAC TX
0x0060: 0x006F	Reserved	—
0x0070	TX Unidirectional Control Register	MAC TX
0x0071: 0x009F	Reserved	—
0x00A0: 0x00FF	RX Configuration and Status Registers	MAC RX
0x0100: 0x010C	TX Timestamp Registers	MAC TX
0x0120: 0x012C	RX Timestamp Registers	MAC RX
0x0140: 0x023F	Statistics Registers	MAC TX, MAC RX
0x0240: 0x0241	ECC Registers	MAC TX, MAC RX

4.1.1 Mapping 10-Gbps Ethernet MAC Registers to LL Ethernet 10G MAC Registers

Use this table to map the legacy Ethernet 10-Gbps MAC registers to the LL Ethernet 10G MAC registers.

Table 19. Register Mapping

Register Names (10-Gbps Ethernet MAC)	Offset (10-Gbps Ethernet MAC)	Offset (LL Ethernet 10G MAC)
MAC TX Configuration Registers		
TX Packet Control	1000	020
TX Transfer Status	1001	Not used.
TX Pad Insertion Control	1040	024
TX CRC Insertion Control	1080	026
TX Packet Underflow Count[31:0]	10C0	03E
TX Packet Underflow Count[35:32]	10C1	03F
TX Preamble Pass-Through Mode Control	1100	028
TX Unidirectional	1120	070
TX Pause Frame Control	1140	040
TX Pause Frame Quanta	1141	042
TX Pause Frame Enable	1142	044
TX PFC0 Pause Quanta	1180	048
TX PFC1 Pause Quanta	1181	049
TX PFC2 Pause Quanta	1182	04A
TX PFC3 Pause Quanta	1183	04B
TX PFC4 Pause Quanta	1184	04C
TX PFC5 Pause Quanta	1185	04D
TX PFC6 Pause Quanta	1186	04E
TX PFC7 Pause Quanta	1187	04F
TX PFC0 Hold-off Quanta	1190	058
TX PFC1 Hold-off Quanta	1191	059
TX PFC2 Hold-off Quanta	1192	05A
TX PFC3 Hold-off Quanta	1193	05B
TX PFC4 Hold-off Quanta	1194	05C
TX PFC5 Hold-off Quanta	1195	05D
TX PFC6 Hold-off Quanta	1196	05E
TX PFC7 Hold-off Quanta	1197	05F
TX PFC Enable	11A0	046
TX Address Insertion Control	1200	02A
TX Address Insertion MAC Address[31:0]	1201	010
TX Address Insertion MAC MAC Address[47:32]	1202	011
TX Maximum Frame Length	1801	02C
MAC RX Configuration Registers		

continued...

Register Names (10-Gbps Ethernet MAC)	Offset (10-Gbps Ethernet MAC)	Offset (LL Ethernet 10G MAC)
RX Transfer Control	0000	0A0
RX Transfer Status	0001	Not used
RX Pad/CRC Control	0040	0A4
RX CRC Check Control	0080	0A6
RX Overflow Truncated Packet Count[31:0]	00C0	0FC
RX Overflow Truncated Packet Count[35:32]	00C1	0FD
RX Overflow Dropped Packet Count[31:0]	00C2	0FE
RX Overflow Dropped Packet Count[35:32]	00C3	0FF
RX Preamble Forward Control	0100	0A8
RX Preamble Pass-Through Mode Control	0140	0AA
RX Frame Filtering Control	0800	0AC
RX Maximum Frame Length	0801	0AE
RX Frame MAC Address[31:0]	0802	010
RX Frame MAC Address[47:32]	0803	011
RX Supplementary Address 0[31:0]	0804	0B0
RX Supplementary Address 0[47:32]	0805	0B1
RX Supplementary Address 1[31:0]	0806	0B2
RX Supplementary Address 1[47:32]	0807	0B3
RX Supplementary Address 2[31:0]	0808	0B4
RX Supplementary Address 2[47:32]	0809	0B5
RX Supplementary Address 3[31:0]	080A	0B6
RX Supplementary Address 3[47:32]	080B	0B7
RX PFC Control	0818	0C0
TX Time Stamp Registers		
TX Period for 10G	1110	100
TX Fractional Nano-second Adjustment for 10G	1112	102
TX Nano-second Adjustment for 10G	1113	104
TX Period for 10M/100M/1G	1118	108
TX Fractional Nano-second Adjustment for 10M/100M/1G/ 2.5G	111A	10A
TX Nano-second Adjustment for 10M/100M/1G/2.5G	111B	10C
RX Time Stamp Registers		
RX Period for 10G	0110	120
RX Fractional Nano-second Adjustment for 10G	0112	122
RX Nano-second Adjustment for 10G	0113	124
RX Period for 10M/100M/1G	0118	128

continued...

Register Names (10-Gbps Ethernet MAC)	Offset (10-Gbps Ethernet MAC)	Offset (LL Ethernet 10G MAC)
RX Fractional Nano-second Adjustment for 10M/100M/1G/2.5G	011A	12A
RX Nano-second Adjustment for 10M/100M/1G/2.5G	011B	12C
All TX Statistics Registers	1Cxx	14x
All RX Statistics Registers	0Cxx	1Cx
Status Registers		
ECC Error Status	Not applicable	240
ECC Error Enable	Not applicable	241

4.2 Register Access

Table 20. Types of Register Access

Access	Definition
RO	Read only.
RW	Read and write.
RW1C	Read, and write and clear. The user application writes 1 to the register bit(s) to invoke a defined instruction. The IP core clears the bit(s) upon executing the instruction.

4.3 Primary MAC Address

Table 21. Primary MAC Address

Word Offset	Register Name	Description	Access	HW Reset Value
0x0010	`primary_mac_addr0`	6-byte primary MAC address. Configure this register with a non-zero value before you enable the MAC IP core for operations. Map the primary MAC address as follows: • `primary_mac_addr0`: Lower four bytes of the address. • `primary_mac_addr1[15:0]`: Upper two bytes of the address. • `primary_mac_addr1[31:16]`: Reserved. Example If the primary MAC address is 00-1C-23-17-4A-CB, set `primary_mac_addr0` to 0x23174ACB and `primary_mac_addr1` to 0x0000001C. Usage On transmit, the MAC IP core uses this address to fill the source address field in control frames. For data frames from the client, the MAC IP core replaces the source address field with the primary MAC address when the `tx_src_addr_override` register is set to 1. On receive, the MAC IP core uses this address to filter unicast frames when the EN_ALLUCAST bit of the `rx_frame_control` register is set to 0.	RW	0x0
0x0011	`primary_mac_addr1`			

continued...

Word Offset	Register Name	Description	Access	HW Reset Value
		The MAC IP core drops frames whose destination address is different from the value of the primary MAC address.		

4.4 MAC Reset Control Register

This register is used only in 10G, 1G/10G, and 10M/100M/1G/10G operating modes.

Table 22. MAC Reset Control Register

Word Offset	Register Name	Description	Access	HW Reset Value
0x001F 0x08FF	`mac_reset_control`	The user application can use the specified bits in this register to reset the MAC datapaths. The effect is the same as asserting the `tx_rst_n` or `rx_rst_n` signals. • Bit 0—TX datapath reset. 0: Stops the reset process. 1: Starts the reset process. • Bits 7:1—reserved. • Bit 8—RX datapath reset. 0: Stops the reset process. 1: Starts the reset process. • Bits 31:9—reserved. If you turn on **Use legacy Ethernet 10G MAC Avalon Memory-Mapped interface**, the word offset is `0x08FF`. Otherwise, the word offset is `0x001F`.	RW	0x0

4.5 TX_Configuration and Status Registers

Table 23. TX Configuration and Status Registers

Word Offset	Register Name	Description	Access	HW Reset Value
0x0020	`tx_packet_control`	• Bit 0—configures the TX path. 0: Enables the TX path. 1: Disables the TX path. The MAC IP core indicates a backpressure on the Avalon-ST transmit data interface by deasserting the `avalon_st_tx_ready` signal. When disabled, the IP core stops generating new pause and PFC frames. • Bits 31:1—reserved. You can change the value of this register as necessary. If the TX path is disabled while a frame is being transmitted, the MAC IP core completes the transmission before disabling the TX path.	RW	0x0
0x0022	`tx_transfer_status`	The MAC sets the following bits to indicate the status of the TX datapath. • Bits 7:0—reserved. • Bit 8: TX datapath status. 0: The TX datapath is idle. 1: A TX data transfer is in progress. • Bits 11:9—reserved. • Bit 12: TX datapath reset status. 0: The TX datapath is not in reset. 1: The TX datapath is in reset.	RO	0x0

continued...

Word Offset	Register Name	Description	Access	HW Reset Value
0x0024	tx_pad_control	• Bit 0—padding insertion enable on transmit. 0: Disables padding insertion. The client must ensure that the length of the data frame meets the minimum length as required by the IEEE 802.3 specifications. 1: Enables padding insertion. The MAC IP core inserts padding bytes into the data frames from the client to meet the minimum length as required by the IEEE 802.3 specifications. When padding insertion is enabled, you must set tx_crc_control[] to 0x3 to enable CRC insertion. • Bits 31:1—reserved. Configure this register before you enable the MAC IP core for operations.	RW	0x1
0x0026	tx_crc_control	• Bit 0—always set this bit to 1. • Bit 1—configures CRC insertion. 0: Disables CRC insertion. The client must provide the CRC field and ensure that the length of the data frame meets the minimum required length. 1: Enables CRC insertion. The MAC IP core computes the CRC field and inserts it into the data frame. • Bits 31:2—reserved. Configure this register before you enable the MAC IP core for operations.	RW	0x3
0x0028	tx_preamble_control(5)	• Bit 0—configures the preamble passthrough mode on transmit. 0: Disables preamble passthrough. The MAC IP core inserts the standard preamble specified by the IEEE 802.3 specifications into the data frame. 1: Enables preamble passthrough. The MAC IP core identifies the first 8 bytes of the data frame from the client as a custom preamble. • Bits 31:1—reserved. Configure this register before you enable the MAC IP core for operations.	RW	0x0
0x002A	tx_src_addr_override	• Bit 0—configures source address override. 0: Disables source address override. The client must fill the source address field with a valid address.. 1: Enables source address override. The MAC IP core overwrites the source address field in data frames with the primary MAC address specified in the tx_primary_mac_addr0 and tx_primary_mac_addr1 registers. • Bits 31:1—reserved. Configure this register before you enable the MAC IP core for operations.	RW	0x0

continued...

(5) This register is used only when you turn on **Enable preamble pass-through mode** option. It is reserved when not used.

Word Offset	Register Name	Description	Access	HW Reset Value
0x002C	`tx_frame_maxlength`	• Bits 15:0—specify the maximum allowable frame length. The MAC IP core uses this register only for the purpose of collecting statistics. When the length of the data frame from the client exceeds this value, the MAC IP core asserts the `avalon_st_txstatus_error[1]` signal to flag the frame as oversized. The MAC IP core then forwards the oversized frame through the transmit datapath as is. • Bits 31:16—reserved. Configure this register before you enable the MAC IP core for operations.	RW	0x5EE (1518)
0x002D	`tx_vlan_detection`	• Bit 0—TX VLAN detection disable. 0: The MAC detects VLAN and stacked VLAN frames. 1: The MAC does not detect VLAN and stacked VLAN frames. When received, the MAC treats them as basic frames and considers their tags as payload bytes. • Bits 31:1—reserved.	RW	0x0
0x002E 0x081E	`tx_ipg_10g`	• Bit 0—use this bit to specify the average IPG for operating speed of 10 Gbps. 0: Sets the average IPG to 8 bytes. 1: Sets the average IPG to 12 bytes. • Bits 31:1—reserved. The Unidirectional feature does not support an average IPG of 8 bytes. If you turn on **Use legacy Ethernet 10G MAC Avalon Memory-Mapped interface**, the word offset is `0x081E`. Otherwise, the word offset is `0x002E`.	RW	0x1
0x002F 0x081F	`tx_ipg_10M_100M_1G`	• Bits 3:0—use these bits to specify the average IPG for operating speed of 10 Mbps, 100 Mbps or 1 Gbps. Valid values are between 8 to 15 bytes. • Bits 31:4—reserved. If you turn on **Use legacy Ethernet 10G MAC Avalon Memory-Mapped interface**, the word offset is `0x081F`. Otherwise, the word offset is `0x002F`.	RW	0x0C
0x003E	`tx_underflow_counter0`	36-bit error counter that collects the number of truncated TX frames when TX buffer underflow persists. • `tx_underflow_counter0`: Lower 32 bits of the error counter. • `tx_underflow_counter1[3:0]`: Upper 4 bits of the error counter. • `tx_underflow_counter1[31:4]`—reserved. To read the counter, read the lower 32 bits followed by the upper 4 bits. The IP core clears the counter after a read.	RO	0x0
0x003F	`tx_underflow_counter1`			

4.6 Flow Control Registers

Table 24. Flow Control Registers

Word Offset	Register Name	Description	Access	HW Reset Value
0x0040	tx_pauseframe_control	• Bits 1:0—configures the transmission of pause frames. 00: No pause frame transmission. 01: Trigger the transmission of an XON pause frame (pause quanta = 0), if the transmission is not disabled by other conditions. 10: Trigger the transmission of an XOFF pause frame (pause quanta = tx_pauseframe_quanta register), if the transmission is not disabled by other conditions. 11: Reserved. This setting does not trigger any action. • Bits 31:2—reserved. Changes to this self-clearing register affects the next transmission of a pause frame.	RW	0x0
0x0042	tx_pauseframe_quanta	• Bits 15:0—pause quanta in unit of quanta, 1 unit = 512 bits time. The MAC IP core uses this value when it generates XOFF pause frames. An XOFF pause frame with a quanta value of 0 is equivalent to an XON frame. • Bits 31:16—reserved. Configure this register before you enable the MAC IP core for operations.	RW	0x0
0x0043	tx_pauseframe_holdoff_quanta	• Bits 15:0—specifies the gap between two consecutive transmissions of XOFF pause frames in unit of quanta, 1 unit = 512 bits time. The gap prevents back-to-back transmissions of pause frames, which may affect the transmission of data frames. • Bits 31:16—reserved. Configure this register before you enable the MAC IP core for operations.	RW	0x1
0x0044	tx_pauseframe_enable	• Bit 0—configures the transmission of pause frames. This bit affects pause frame requests from both register and vector settings. 0: Disables pause frame transmission. 1: Enables pause frame transmission, if TX path is enabled by tx_packet_control. • Bits 2:1—specifies the trigger for pause frame requests. 00: Accepts pause frame requests only from vector setting, avalon_st_pause_data. 01: Accepts pause frame requests only from register setting, tx_pauseframe_control. 10 / 11: Reserved. • Bits 31:3—reserved. Configure this register before you enable the MAC IP core for operations.	RW	0x1
0x0046	tx_pfc_priority_enable[6]	Enables priority-based flow control on the TX datapath.	RW	0x0

continued...

Word Offset	Register Name	Description	Access	HW Reset Value
		• Bits 7:0—setting bit *n* enables priority-based flow control for priority queue *n*. For example, setting `tx_pfc_priority_enable[0]` enables queue 0. • Bits 31:8—reserved. Configure this register before you enable the MAC IP core for operations.		
0x0048	`pfc_pause_quanta_0`(6)	Specifies the pause quanta for each priority queue. • Bits 15:0—`pfc_pause_quanta_`*n*`[15:0]` specifies the pause length for priority queue *n* in quanta unit, where 1 unit = 512 bits time. • Bits 31:16—reserved. Configure these registers before you enable the MAC IP core for operations.	RW	0x0
0x0049	`pfc_pause_quanta_1`(6)			
0x004A	`pfc_pause_quanta_2`(6)			
0x004B	`pfc_pause_quanta_3`(6)			
0x004C	`pfc_pause_quanta_4`(6)			
0x004D	`pfc_pause_quanta_5`(6)			
0x004E	`pfc_pause_quanta_6`(6)			
0x004F	`pfc_pause_quanta_7`(6)			
0x0058	`pfc_holdoff_quanta_0`(6)	Specifies the gap between two consecutive transmissions of XOFF pause frames in unit of quanta, 1 unit = 512 bits time. The gap prevents back-to-back transmissions of pause frames, which may affect the transmission of data frames. • Bits 15:0— `pfc_holdoff_quanta_`*n*`[15:0]` specifies the gap for priority queue *n*. • Bits 31:16—reserved. Configure these registers before you enable the MAC IP core for operations.	RW	0x1
0x0059	`pfc_holdoff_quanta_1`(6)			
0x005A	`pfc_holdoff_quanta_2`(6)			
0x005B	`pfc_holdoff_quanta_3`(6)			
0x005C	`pfc_holdoff_quanta_4`(6)			
0x005D	`pfc_holdoff_quanta_5`(6)			
0x005E	`pfc_holdoff_quanta_6`(6)			
0x005F	`pfc_holdoff_quanta_7`(6)			

(6) This register is used only when you turn on the **Enable preamble pass-through mode** option. It is reserved when not used.

4.7 Unidirectional Control Registers

Table 25. Unidirectional Control Registers

Word Offset	Register Name	Description	Access	HW Reset Value
0x0070	`tx_unidir_control`(7)	• Bit 0—configures the unidirectional feature on the TX path. 0: Disables unidirectional feature. 1: Enables unidirectional feature. • Bit 1—configures remote fault sequence generation when the unidirectional feature is enabled on the TX path. 0: Enable remote fault sequence generation on detecting local fault. 1: Disable remote fault sequence generation. • Bit 2—configures user-triggered remote fault notification when the unidirectional feature is enabled on the TX path. 0: Default setting. 1: The IP core sends remote fault notifications continuously until this bit is cleared. • Bits 31:3—reserved. Configure this register before you enable the MAC IP core for operations.	RW	0x0

4.8 RX Configuration and Status Registers

Table 26. RX Configuration and Status Registers

Word Offset	Register Name	Description	Access	HW Reset Value
0x00A0	`rx_transfer_control`	• Bit 0—RX path enable. 0: Enables the RX path. 1: Disables the RX path. The MAC IP core drops all incoming frames. • Bits 31:1—reserved. A change of value in this register takes effect at a packet boundary. Any transfer in progress is not affected.	RW	0x0
0x00A2	`rx_transfer_status`	The MAC sets the following bits to indicate the status of the RX datapath. • Bits 7:0—reserved. • Bit 8: RX datapath status. 0: The RX datapath is idle. 1: An RX data transfer is in progress. • Bits 11:9—reserved. • Bit 12: RX datapath reset status. 0: The RX datapath is not in reset. 1: The RX datapath is in reset.	RO	0x0

continued...

(7) This register is used when you turn on **Enable unidirectional feature**. It is reserved when not used.

Word Offset	Register Name	Description	Access	HW Reset Value
0x00A4	`rx_padcrc_control`	• Bits [1:0]—Padding and CRC removal on receive. 00: Retains the padding bytes and CRC field, and forwards them to the client. 01: Retains only the padding bytes. The MAC IP core removes the CRC field before it forwards the RX frame to the client. 11: Removes the padding bytes and CRC field before the RX frame is forwarded to the client. 10: Reserved. • Bits 31:2—reserved. Configure this register before you enable the MAC IP core for operations.	RW	0x1
0x00A6	`rx_crccheck_control`	CRC checking on receive. • Bit 0—always set this bit to 0. • Bit 1—CRC checking enable. 0: Ignores the CRC field. 1: Checks the CRC field and reports the status to `avalon_st_rx_error[1]` and `avalon_st_rxstatus_error`. • Bits 31:2—reserved. Configure this register before you enable the MAC IP core for operations.	RW	0x2
0x00A8	`rx_custom_preamble_forward` (8)	• Bit 0—configures the forwarding of the custom preamble to the client. 0: Removes the custom preamble from the RX frame. 1: Retains and forwards the custom preamble to the client. • Bits 31:1—reserved. Configure this register before you enable the MAC IP core for operations.	RW	0x0
0x00AA	`rx_preamble_control`(8)	• Bit 0—preamble passthrough enable on receive. 0: Disables preamble passthrough. The MAC IP core checks for START and SFD during packet decapsulation process. 1: Enables preamble passthrough. The MAC IP core checks only for START during packet decapsulation process. • Bits 31:1—reserved. Configure this register before you enable the MAC IP core for operations.	RW	0x0
0x00AC	`rx_frame_control`	Configure this register before you enable the MAC IP core for operations. Bit 0—EN_ALLUCAST 0: Filters RX unicast frames using the primary MAC address. The MAC IP core drops unicast frames with a destination address other than the primary MAC address. 1: Accepts all RX unicast frames.	RW	0x3

continued...

(8) This register is used only when you turn on the **Enable preamble pass-through mode** option. It is reserved when not used.

Word Offset	Register Name	Description	Access	HW Reset Value
		Setting this bit and the EN_ALLMCAST to 1 puts the MAC IP core in the promiscuous mode.		
		Bit 1—EN_ALLMCAST 0: Drops all RX multicast frames. 1: Accepts all RX multicast frames. Setting this bit and the EN_ALLUCAST bit to 1 is equivalent to setting the MAC IP core to the promiscuous mode.		
		Bit 2—reserved.		
		Bit 3—FWD_CONTROL. When you turn on the **Priority-based Flow Control** parameter, this bit affects all control frames except the IEEE 802.3 pause frames and priority-based control frames. When the **Priority-based Flow Control** parameter is not enabled, this bit affects all control frames except the IEEE 802.3 pause frames. 0: Drops the control frames. 1: Forwards the control frames to the client.		
		Bit 4—FWD_PAUSE 0: Drops pause frames. 1: Forwards pause frames to the client.		
		Bit 5—IGNORE_PAUSE 0: Processes pause frames. 1: Ignores pause frames.		
		Bits 15:6—reserved.		
		Bit 16—EN_SUPP0 0: Disables the use of supplementary address 0. 1: Enables the use of supplementary address 0.		
		Bit 17—EN_SUPP1 0: Disables the use of supplementary address 1. 1: Enables the use of supplementary address 1.		
		Bit 18—EN_SUPP2 0: Disables the use of supplementary address 2. 1: Enables the use of supplementary address 2.		
		Bit 19—EN_SUPP3 0: Disables the use of supplementary address 3. 1: Enables the use of supplementary address 3.		
		Bits 31:20—reserved.		
0x00AE	rx_frame_maxlength	• Bits 15:0—specify the maximum allowable frame length. The MAC asserts the avalon_st_rx_error[3] signal when the length of the RX frame exceeds the value of this register. • Bits 16:31—reserved. Configure this register before you enable the MAC IP core for operations.	RW	1518

continued...

Word Offset	Register Name	Description	Access	HW Reset Value
0x00AF	rx_vlan_detection	• Bit 0—RX VLAN detection disable. 0: The MAC detects VLAN and stacked VLAN frames. 1: The MAC does not detect VLAN and stacked VLAN frames. When received, the MAC treats them as basic frames and considers their tags as payload bytes. • Bits 31:1—reserved.	RW	0x0
0x00B0	rx_frame_spaddr0_0	You can specify up to four 6-byte supplementary addresses: • rx_framedecoder_spaddr0_0/1 • rx_framedecoder_spaddr1_0/1 • rx_framedecoder_spaddr2_0/1 • rx_framedecoder_spaddr3_0/1 Configure the supplementary addresses before you enable the MAC RX datapath. Map the supplementary addresses to the respective registers in the same manner as the primary MAC address. Refer to the description of primary_mac_addr0 and primary_mac__addr1.The MAC IP core uses the supplementary addresses to filter unicast frames when the following conditions are set: • The use of the supplementary addresses are enabled using the respective bits in the rx_frame_control register. • The en_allucast bit of the rx_frame_control register is set to 0.	RW	0x0
0x00B1	rx_frame_spaddr0_1			
0x00B2	rx_frame_spaddr1_0			
0x00B3	rx_frame_spaddr1_1			
0x00B4	rx_frame_spaddr2_0			
0x00B5	rx_frame_spaddr2_1			
0x00B6	rx_frame_spaddr3_0			
0x00B7	rx_frame_spaddr3_1			
0x00C0	rx_pfc_control(9)	• Bits 7:0—enables priority-based flow control on the RX datapath. Setting bit *n* to 0 enables priority-based flow control for priority queue *n*. For example, setting rx_pfc_control[0] to 0 enables queue 0. • Bits 15:9—reserved. • Bit 16—configures the forwarding of priority-based control frames to the client. 0: Drops the control frames. 1: Forwards the control frames to the client. • Bits 31:17—reserved. Configure this register before you enable the MAC IP core for operations.	RW	0x1
0x00FC	rx_pktovrflow_error	36-bit error counter that collects the number of RX frames that are truncated when a FIFO buffer overflow persists: • 0x00FC = Lower 32 bits of the error counter. • 0x00FD = Upper 4 bits of the error counter occupy bits [3:0]. Bits [31:4] are unused.	RO	0x0
0x00FD				

continued...

(9) This register is used only when you turn on the **Enable priority-based flow control (PFC)** option. It is reserved when not used.

Word Offset	Register Name	Description	Access	HW Reset Value
		To read the counter, read the lower 32 bits followed by the upper 4 bits. The IP core clears the counter after a read.		
0x00FE 0x00FF	`rx_pktovrflow_etherStatsDropEvents`	36-bit error counter that collects the number of RX frames that are dropped when FIFO buffer overflow persists: • 0x00FE = Lower 32 bits of the error counter. • 0x00FF = Upper 4 bits of the error counter occupy bits [3:0]. Bits [31:4] are unused. To read the counter, read the lower 32 bits followed by the upper 4 bits. The IP core clears the counter after a read.	RO	0x0

Related Links

- Length Checking on page 41
- Statistics Registers on page 82

4.9 Timestamp Registers

The TX and RX timestamp registers are available when you turn on the **Enable time stamping** parameter. Otherwise, these registers are reserved.

Table 27. Timestamp Registers

Word Offset	Register Name	Description	Access	HW Reset Value
0x0100	`tx_period_10G`	Specifies the clock period for the timestamp adjustment on the datapaths for 10G operations. The MAC IP core multiplies the value of this register by the number of stages separating the actual timestamp and XGMII bus. • Bits 0 to 15—period in fractional nanoseconds. • Bits 16 to 19—period in nanoseconds. • Bits 20 to 31—reserved. Set these bits to 0. The default value is 3.2 ns for 312.5 MHz clock. Configure this register before you enable the MAC IP core for operations.	RW	0x33333
0x0102	`tx_fns_adjustment_10G`	Static timing adjustment in fractional nanoseconds on the datapaths for 10G operations. • Bits 0 to 15—adjustment period in fractional nanoseconds. • Bits 16 to 31—reserved. Set these bits to 0. Configure this register before you enable the MAC IP core for operations. For timing adjustment calculations, refer to the related links.	RW	0x0
0x0104	`tx_ns_adjustment_10G`	Static timing adjustment in nanoseconds on the datapaths for 10G operations. • Bits 0 to 15—adjustment period in nanoseconds. • Bits 16 to 31—reserved. Set these bits to 0. Configure this register before you enable the MAC IP core for operations. For timing adjustment calculations, refer to the related links.	RW	0x0
0x0108	`tx_period_mult_speed`	Specifies the clock period for timestamp adjustment on the datapaths for 10M/100M/1G operations. The MAC IP core multiplies the value of this register by the number of stages separating the actual timestamp and GMII/MII bus. • Bits 0 to 15—period in fractional nanoseconds. • Bits 16 to 19—period in nanoseconds. • Bits 20 to 31—reserved. Set these bits to 0. The default value is 8 ns for 125 MHz clock. Configure this register before you enable the MAC IP core for operations.	RW	0x80000

continued...

Word Offset	Register Name	Description	Access	HW Reset Value
		The IP core automatically sets the clock period for 1G/2.5G configurations. For 1G, the clock period is set to 16 ns for 62.5 MHz clock; for 2.5G, the clock period is 6.4 ns for 156.25 MHz clock.		
0x0120	`rx_period_10G`	Specifies the clock period for the timestamp adjustment on the datapaths for 10G operations. The MAC IP core multiplies the value of this register by the number of stages separating the actual timestamp and XGMII bus. • Bits 0 to 15—period in fractional nanoseconds. • Bits 16 to 19—period in nanoseconds. • Bits 20 to 31—reserved. Set these bits to 0. The default value is 3.2 ns for 312.5 MHz clock. Configure this register before you enable the MAC IP core for operations.	RW	0x33333
0x0122	`rx_fns_adjustment_10G`	Static timing adjustment in fractional nanoseconds on the datapaths for 10G operations. • Bits 0 to 15—adjustment period in fractional nanoseconds. • Bits 16 to 31—reserved. Set these bits to 0. Configure this register before you enable the MAC IP core for operations. For timing adjustment calculations, refer to the related links.	RW	0x0
0x0124	`rx_ns_adjustment_10G`	Static timing adjustment in nanoseconds on the datapaths for 10G operations. • Bits 0 to 15—adjustment period in nanoseconds. • Bits 16 to 31—reserved. Set these bits to 0. Configure this register before you enable the MAC IP core for operations. For timing adjustment calculations, refer to the related links.	RW	0x0
0x0128	`rx_period_mult_speed`	Specifies the clock period for timestamp adjustment on the datapaths for 10M/100M/1G operations. The MAC IP core multiplies the value of this register by the number of stages separating the actual timestamp and GMII/MII bus. • Bits 0 to 15—period in fractional nanoseconds. • Bits 16 to 19—period in nanoseconds. • Bits 20 to 31—reserved. Set these bits to 0. The default value is 8 ns for 125 MHz clock. Configure this register before you enable the MAC IP core for operations. The IP core automatically sets the clock period for 1G/2.5G configurations. For 1G, the clock period is set to 16 ns for 62.5 MHz clock; for 2.5G, the clock period is 6.4 ns for 156.25 MHz clock.	RW	0x80000

continued...

Word Offset	Register Name	Description	Access	HW Reset Value
0x10A	`tx_fns_adjustment_mult_speed`	Static timing adjustment in fractional nanoseconds on the datapaths for 10M/100M/1G/2.5G operations. • Bits 0 to 15—adjustment period in fractional nanoseconds. • Bits 16 to 31—reserved. Set these bits to 0. Configure this register before you enable the MAC IP core for operations. For timing adjustment calculations, refer to the related links.	RW	0x0
0x10C	`tx_ns_adjustment_mult_speed`	Static timing adjustment in nanoseconds on the datapaths for 10M/100M/1G/2.5G operations. • Bits 0 to 15—adjustment period in nanoseconds. • Bits 16 to 31—reserved. Set these bits to 0. Configure this register before you enable the MAC IP core for operations. For timing adjustment calculations, refer to the related links.	RW	0x0
0x12A	`rx_fns_adjustment_mult_speed`	Static timing adjustment in fractional nanoseconds on the datapaths for 10M/100M/1G/2.5G operations. • Bits 0 to 15—adjustment period in fractional nanoseconds. • Bits 16 to 31—reserved. Set these bits to 0. Configure this register before you enable the MAC IP core for operations. For timing adjustment calculations, refer to the related links.	RW	0x0
0x12C	`rx_ns_adjustment_mult_speed`	Static timing adjustment in nanoseconds on the datapaths for 10M/100M/1G/2.5G operations. • Bits 0 to 15—adjustment period in nanoseconds. • Bits 16 to 31—reserved. Set these bits to 0. Configure this register before you enable the MAC IP core for operations. For timing adjustment calculations, refer to the related links.	RW	0x0
0x110	`tx_asymmetry`	Specifies the asymmetry value and direction of arithmetic operation. • Bits 0 to 16—asymmetry value. • Bit 17—direction. — Set to 0—add asymmetry value to correction field (CF). — Set to 1—minus asymmetry value from CF. • Bit 18—enable bit.	RW	0x0

continued...

Word Offset	Register Name	Description	Access	HW Reset Value
0x112	tx_p2p	Specifies the direction of arithmetic operation for meanPathDelay. • Bit 0— direction. — Set to 0—add meanPathDelay value to CF. — Set to 1—minus meanPathDelay value from CF. • Bits 1 to 30—reserved.	RW	0x0
0x114	tx_cf_err_stat	• Bits 0—error status bit to indicate that ingress correction field is equal to the absolute maximum, 64'h7FF_FFFF_FFFF_FFFF. • Bits 0 to 15—reserved. • Bit 16—error status bit to indicate that egress correction field is equal or larger than absolute maximum, 64'h7FFF_FFFF_FFFF_FFFF. • Bit 17—error status bit to indicate that residence time is equal or larger than 4 seconds. • Bit 18—error status bit to indicate that residence time is a negative value. • Bits 19 to 31—reserved.	RW1C	0x0
0x12E	rx_p2p_mpd_ns	meanPathDelay valid and value in ns. The peer-to-peer mechanism delivers meanPathDelay for each ingress port. This needs to be added to the Sync packet's correction field before the packet is sent out on egress port. Thus, the egress port might add any of the ingress ports' 'meanPathDelay'. The value to be added at the egress port should correspond to the ingress port on which the Sync packet has arrived. • Bit 30—Indicates meanPathDelay is valid. • Bits 0 to 29—meanPathDelay value in nanosecond. • Bit 31—reserved.	RW	0x0
0x130	rx_p2p_mpd_fns	• Bits 0 to 15—meanPathDelay value in fractional nanosecond. • Bits 16 to 31—reserved.	RW	0x0

Related Links

Calculating Timing Adjustments on page 79

4.9.1 Calculating Timing Adjustments

You can derive the required timing adjustments in ns and fns from the hardware PMA delay.

Table 29. Hardware PMA Delay

<table>
<tr><th rowspan="2">Type</th><th rowspan="2">Device</th><th rowspan="2">PMA Mode (bit)</th><th colspan="2">Latency</th><th rowspan="2">MAC Configurations</th></tr>
<tr><th>TX</th><th>RX</th></tr>
<tr><td rowspan="16">Digital[10]</td><td rowspan="3">Arria V GZ
Stratix V</td><td>40</td><td>123 UI</td><td>87 UI</td><td>10GbE
10G of 10M-10GbE</td></tr>
<tr><td>32</td><td>99 UI</td><td>84 UI</td><td>10GbE</td></tr>
<tr><td>10</td><td>53 UI</td><td>26 UI</td><td>1G/100M/10M of 10M-10GbE</td></tr>
<tr><td>Arria V GX/GT/SX/ST</td><td>10</td><td>42 UI</td><td>44 UI</td><td>1G/2.5GbE</td></tr>
<tr><td rowspan="3">Intel Arria 10</td><td>40</td><td>147 UI</td><td>66.5 UI</td><td>10GbE
10G of 10M-10GbE</td></tr>
<tr><td>32</td><td>123 UI</td><td>58.5 UI</td><td>10GbE
10G of 10M-10GbE</td></tr>
<tr><td>10</td><td>43 UI</td><td>24.5 UI</td><td>1G/100M/10M of 10M-10GbE
1G/2.5GbE</td></tr>
<tr><td rowspan="3">Intel Cyclone 10 GX</td><td>40</td><td>147 UI</td><td>66.5 UI</td><td>10GbE
10G of 10M-10GbE</td></tr>
<tr><td>32</td><td>123 UI</td><td>58.5 UI</td><td>10GbE
10G of 10M-10GbE</td></tr>
<tr><td>10</td><td>43 UI</td><td>24.5 UI</td><td>1G/100M/10M of 10M-10GbE
1G/2.5GbE</td></tr>
<tr><td rowspan="3">Intel Stratix 10</td><td>40</td><td>147 UI</td><td>68.5 UI</td><td>10GbE
10G of 10M-10GbE</td></tr>
<tr><td>32</td><td>123 UI</td><td>60.5 UI</td><td>10GbE
10G of 10M-10GbE</td></tr>
<tr><td>10</td><td>43 UI</td><td>26.5 UI</td><td>1G/100M/10M of 10M-10GbE
1G/2.5GbE</td></tr>
<tr><td rowspan="6">Analog[11]</td><td>Arria V</td><td rowspan="6">—</td><td rowspan="4">-1.1 ns</td><td rowspan="4">1.75 ns</td><td rowspan="4">All</td></tr>
<tr><td>Stratix V</td></tr>
<tr><td>Intel Arria 10</td></tr>
<tr><td>Intel Cyclone 10 GX</td></tr>
<tr><td rowspan="2">Intel Stratix 10</td><td>0.69 ns</td><td>3.54 ns</td><td>10G of 1G/2.5G/10Gbe</td></tr>
<tr><td>0.18 ns</td><td>3.03 ns</td><td>1G/2.5GbE</td></tr>
</table>

(10) For 10G, 1 UI is 97 ps. For 2.5G, 1 UI is 320 ps. For 10M/100M/1G,1 UI is 800 ps.

(11) Valid for the HSSI clock routing using periphery clock. Other clocking scheme might result in deviation of a few ns.

The example below shows the required calculation for a 10M – 10GbE design targeting a Stratix V device.

Table 30. Example: Calculating RX Timing Adjustments for 10M – 10GbE Design in Stratix V Device

Step	Description	10G	10M, 100M or 1G
1	Identify the digital latency for the device.	For Stratix V using the PMA mode of 40 bits, the digital latency is 87 UI.	For Stratix V using the PMA mode of 10 bits, the digital latency is 26 UI.
2	Convert the digital latency in UI to ns.	87 UI * 0.097 = 8.439 ns	26 UI * 0.8 = 20.8 ns
3	Add the analog latency to the digital latency in ns.	8.439 ns + 1.75 ns = 10.189 ns	20.8 ns + 1.75 ns = 22.55 ns
4	Add any external PHY delay to the total obtained in step 3. In this example, an external PHY delay of 1 ns is assumed.	10.189 ns + 1 ns = 11.189 ns	22.55 ns + 1 ns = 23.55 ns
5	Convert the total latency to ns and fns in hexadecimal.	ns: 0xB fns: 0.189 * 65536 = 0x3062	ns: 0x17 fns: 0.55 * 65536 = 0x8CCC
6	Configure the respective registers.	`rx_ns_adjustment_10G` = 0xB `rx_fns_adjustment_10G` = 0x3062	`rx_ns_adjustment_mult_speed` = 0x17 `rx_fns_adjustment_mult_speed` = 0x8CCC

Related Links

Timestamp Registers on page 75

4.10 ECC Registers

The ECC registers are used when you turn on **Enable ECC on memory blocks**. They are reserved when not used.

Table 31. ECC Registers

Word Offset	Register Name	Description	Access	HW Reset Value
0x0240 0x0820	`ecc_status`	• Bit 0—a value of '1' indicates that an ECC error was detected and corrected. The user application must write 1 to this bit to clear it. • Bit 1—a value of '1' indicates that an ECC error was detected but not corrected. The user application must write 1 to this bit to clear it. • Bits 31:2—reserved. If you turn on **Use legacy Ethernet 10G MAC Avalon Memory-Mapped interface**, the word offset is `0x0820`. Otherwise, the word offset is `0x0240`.	RWC	0x0
0x0241 0x0821	`ecc_enable`	• Bit 0—specifies how detected and corrected ECC errors are reported. 0: Reported by the `ecc_status[0]` register bit only. 1: Reported by the `ecc_status[0]` register bit and the `ecc_err_det_corr` signal. • Bit 1—specifies how detected and uncorrected ECC errors are reported. 0: Reported by the `ecc_status[0]` register bit only. 1: Reported by the `ecc_status[0]` register bit and the `ecc_err_det_uncorr` signal. • Bits 31:2—reserved. If you turn on **Use legacy Ethernet 10G MAC Avalon Memory-Mapped interface**, the word offset is `0x0821`. Otherwise, the word offset is `0x0241`.	RW	0x0

4.11 Statistics Registers

Statistics counters with prefix `tx_` collect statistics on the TX datapath; prefix `rx_` collect statistics on the RX datapath. The counters collect statistics for the following frames:

- Good frame—error-free frames with a valid frame length.
- Error frame—frames that contain errors or with an invalid frame length.
- Invalid frame—frames that are not supported by the MAC IP core or its current configuration. For example, if the MAC is configured to receive all unicast frames, unicast frames are considered valid because address filtering is disabled. The MAC drops invalid frames.

Most of the statistics counters are 36 bits wide and occupy two offsets. The user application must first read the lower 32 bits followed by the upper 4 bits.

- The lower 32 bits of the counter occupy the first offset.
- The upper 4 bits of the counter occupy bits 3:0 at the second offset.
- Bits 31:5 at the second offset are reserved.

Consider the following guidelines when using the statistics counters:

- Memory-based statistics counters may not be accurate when the MAC IP core receives or transmits back-to-back undersized frames. On the TX datapath, you can enable padding to avoid this situation. Undersized frames are frames with less than 64 bytes.
- Do not access the statistics counters when the TX and RX datapaths reset are in progress. Doing so can lead to unpredictable results.

Table 32. TX and RX Statistics Registers

Word Offset	Register Name	Description	Access	HW Reset Value
0x0140	`tx_stats_clr`	• Bit 0—Set this register to 1 to clear all TX statistics counters. The IP core clears this bit when all counters are cleared. • Bits 31:1—reserved.	RWC	0x0
0x01C0	`rx_stats_clr`	• Bit 0—Set this register to 1 to clear all RX statistics counters. The IP core clears this bit when all counters are cleared. • Bits 31:1—reserved.	RWC	0x0
0x0142	`tx_stats_framesOK`	36-bit statistics counter that collects the number of frames that are successfully received or transmitted, including control frames.	RO	0x0
0x0143				
0x01C2	`rx_stats_framesOK`			
0x01C3				
0x0144	`tx_stats_framesErr`	36-bit statistics counter that collects the number of frames received or transmitted with error, including control frames.	RO	0x0
0x0145				
0x01C4	`rx_stats_framesErr`			

continued...

Word Offset	Register Name	Description	Access	HW Reset Value
0x01C5				
0x01C6	rx_stats_framesCRCErr	36-bit statistics counter that collects the number of RX frames with CRC error.	RO	0x0
0x01C7				
0x0148	tx_stats_octetsOK	64-bit statistics counter that collects the payload length, including the bytes in control frames. The payload length is the number of data and padding bytes received or transmitted. If the `tx_vlan_detection[0]` or `rx_vlan_detection[0]` register bit is set to 1, the VLAN and stacked VLAN tags are counted as part of the TX payload or RX payload respectively.	RO	0x0
0x0149				
0x01C8	rx_stats_octetsOK			
0x01C9				
0x014A	tx_stats_pauseMACCtrl_Frames	36-bit statistics counter that collects the number of valid pause frames received or transmitted.	RO	0x0
0x014B				
0x01CA	rx_stats_pauseMACCtrl_Frames			
0x01CB				
0x014C	tx_stats_ifErrors	36-bit statistics counter that collects the number of frames received or transmitted that are invalid and with error.	RO	0x0
0x014D				
0x01CC	rx_stats_ifErrors			
0x01CD				
0x014E	tx_stats_unicast_FramesOK	36-bit statistics counter that collects the number of good unicast frames received or transmitted, excluding control frames.	RO	0x0
0x014F				
0x01CE	rx_stats_unicast_FramesOK			
0x01CF				
0x0150	tx_stats_unicast_FramesErr	36-bit statistics counter that collects the number of unicast frames received or transmitted with error, excluding control frames.	RO	0x0
0x0151				
0x01D0	rx_stats_unicast_FramesErr			
0x01D1				
0x0152	tx_stats_multicast_FramesOK	36-bit statistics counter that collects the number of good multicast frames received or transmitted, excluding control frames.	RO	0x0
0x0153				
0x01D2	rx_stats_multicast_FramesOK			
0x01D3				
0x0154	tx_stats_multicast_FramesErr	36-bit statistics counter that collects the number of multicast frames received or transmitted with error, excluding control frames.	RO	0x0
0x0155				
0x01D4	rx_stats_multicast_FramesErr			
0x01D5				

continued...

Word Offset	Register Name	Description	Access	HW Reset Value
0x0156	tx_stats_broadcast_FramesOK	36-bit statistics counter that collects the number of good broadcast frames received or transmitted, excluding control frames.	RO	0x0
0x0157				
0x01D6	rx_stats_broadcast_FramesOK			
0x01D7				
0x0158	tx_stats_broadcast_FramesErr	36-bit statistics counter that collects the number of broadcast frames received or transmitted with error, excluding control frames.	RO	0x0
0x0159				
0x01D8	rx_stats_broadcast_FramesErr			
0x01D9				
0x015A	tx_stats_etherStatsOctets	64-bit statistics counter that collects the total number of octets received or transmitted. This count includes good, errored, and invalid frames.	RO	0x0
0x015B				
0x01DA	rx_stats_etherStatsOctets			
0x01DB				
0x015C	tx_stats_etherStatsPkts	36-bit statistics counter that collects the total number of good, errored, and invalid frames received or transmitted.	RO	0x0
0x015D				
0x01DC	rx_stats_etherStatsPkts			
0x01DD				
0x015E	tx_stats_etherStatsUndersizePkts	36-bit statistics counter that collects the number of undersized TX or RX frames.	RO	0x0
0x015F				
0x01DE	rx_stats_etherStatsUndersizePkts			
0x01DF				
0x0160	tx_stats_etherStatsOversizePkts	36-bit statistics counter that collects the number of TX or RX frames whose length exceeds the maximum frame length specified.	RO	0x0
0x0161				
0x01E0	rx_stats_etherStatsOversizePkts			
0x01E1				
0x0162	tx_stats_etherStatsPkts64Octets	36-bit statistics counter that collects the number of 64-byteTX or RX frames, including the CRC field but excluding the preamble and SFD bytes. This count includes good, errored, and invalid frames.	RO	0x0
0x0163				
0x01E2	rx_stats_etherStatsPkts64Octets			
0x01E3				
0x0164	tx_stats_etherStatsPkts65to127Octets	36-bit statistics counter that collects the number of TX or RX frames between the length of 65 and 127 bytes, including the CRC field but excluding the preamble and SFD bytes. This count includes good, errored, and invalid frames.	RO	0x0
0x0165				
0x01E4	rx_stats_etherStatsPkts65to127Octets			
0x01E5				
0x0166	tx_stats_etherStatsPkts128to255Octets	36-bit statistics counter that collects the number of TX or RX frames between the length of 128 and 255 bytes, including the CRC field but	RO	0x0
0x0167				

continued...

Word Offset	Register Name	Description	Access	HW Reset Value
0x01E6	rx_stats_etherStatsPkts128to255Octets	excluding the preamble and SFD bytes. This count includes good, errored, and invalid frames.		
0x01E7				
0x0168	tx_stats_etherStatsPkts256to511Octets	36-bit statistics counter that collects the number of TX or RX frames between the length of 256 and 511 bytes, including the CRC field but excluding the preamble and SFD bytes. This count includes good, errored, and invalid frames.	RO	0x0
0x0169				
0x01E8	rx_stats_etherStatsPkts256to511Octets			
0x01E9				
0x016A	tx_stats_etherStatsPkts512to1023Octets	36-bit statistics counter that collects the number of TX or RX frames between the length of 512 and 1,023 bytes, including the CRC field but excluding the preamble and SFD bytes. This count includes good, errored, and invalid frames.	RO	0x0
0x016B				
0x01EA	rx_stats_etherStatsPkts512to1023Octets			
0x01EB				
0x016C	tx_stats_etherStatPkts1024to1518Octets	36-bit statistics counter that collects the number of TX or RX frames between the length of 1,024 and 1,518 bytes, including the CRC field but excluding the preamble and SFD bytes. This count includes good, errored, and invalid frames.	RO	0x0
0x016D				
0x01EC	rx_stats_etherStatPkts1024to1518Octets			
0x01ED				
0x016E	tx_stats_etherStatsPkts1519toXOctets	36-bit statistics counter that collects the number of TX or RX frames equal or more than the length of 1,519 bytes, including the CRC field but excluding the preamble and SFD bytes. This count includes good, errored, and invalid frames.	RO	0x0
0x016F				
0x01EE	rx_stats_etherStatsPkts1519toXOctets			
0x01EF				
0x01F0	rx_stats_etherStatsFragments	36-bit statistics counter that collects the total number of RX frames with length less than 64 bytes and CRC error. The MAC does not drop these frames.	RO	0x0
0x01F1				
0x01F2	rx_stats_etherStatsJabbers	36-bit statistics counter that collects the number of oversized RX frames with CRC error. The MAC does not drop these frames.	RO	0x0
0x01F3				
0x01F4	rx_stats_etherStatsCRCErr	36-bit statistics counter that collects the number of RX frames with CRC error, whose length is between 64 and the maximum frame length specified in the register. The MAC does not drop these frames.	RO	0x0
0x01F5				
0x0176	tx_stats_unicastMACCtrlFrames	36-bit statistics counter that collects the number of valid TX or RX unicast control frames.	RO	0x0
0x0177				
0x01F6	rx_stats_unicastMACCtrlFrames			
0x01F7				
0x0178	tx_stats_multicastMACCtrlFrames	36-bit statistics counter that collects the number of valid TX or RX multicast control frames.	RO	0x0
0x0179				
0x01F8	rx_stats_multicastMACCtrlFrames			

continued...

Word Offset	Register Name	Description	Access	HW Reset Value
0x01F9				
0x017A	tx_stats_broadcastMACCtrlFrames	36-bit statistics counter that collects the number of valid TX or RX broadcast control frames.	RO	0x0
0x017B				
0x01FA	rx_stats_broadcastMACCtrlFrames			
0x01FB				
0x017C	tx_stats_PFCMACCtrlFrames	36-bit statistics counter that collects the number of valid TX or RX PFC frames.	RO	0x0
0x017D				
0x01FC	rx_stats_PFCMACCtrlFrames			
0x01FD				

5 Interface Signals

Related Links

Interfaces on page 29
Overview of the interfaces and signals.

5.1 Clock and Reset Signals

The LL Ethernet 10G MAC IP core operates in multiple clock domains. You can use different sources to drive the clock and reset domains. You can also use the same clock source as specified in the description of each signal.

Table 33. Clock and Reset Signals

Signal	Operating Mode	Direction	Width	Description
`tx_312_5_clk`	10G, 1G/10G, 1G/2.5G/10G, 1G/2.5G/5G/10G (USXGMII), 10M/100M/1G/10G, 10M/100M/1G/2.5G/10G	In	1	312.5 MHz clock for the MAC TX datapath when the **Enable 10GBASE-R register mode** is disabled. You may use the same clock source for this clock and `rx_312_5_clk`.
`tx_xcvr_clk`	10G	In	1	322.265625 MHz clock for the MAC TX datapath when the **Enable 10GBASE-R register mode** is enabled.
`tx_156_25_clk`	10G, 1G/10G, 1G/2.5G/10G, 1G/2.5G/5G/10G (USXGMII), 10M/100M/1G/10G, 10M/100M/1G/2.5G/10G	In	1	156.25 MHz clock for the MAC TX datapath when you choose to maintain compatibility with the 64-bit Ethernet 10G MAC on the Avalon-ST TX data interface or XGMII. This feature is not available when the **Enable 10GBASE-R register mode** is enabled. Intel recommends that this clock and `tx_312_5_clk` share the same clock source. This clock must be synchronous to `tx_312_5_clk`. Their rising edges must align and must have 0 ppm and phase-shift.
	1G/2.5G, 10M/100M/1G/2.5G	In	1	156.25 MHz clock for the Avalon-ST TX data interface.
`tx_rst_n`	All	In	1	Active-low asynchronous reset in the `tx_312_5_clk` clock domain for the MAC TX datapath. For the reset requirements, refer to the related links.

continued...

Signal	Operating Mode	Direction	Width	Description
rx_312_5_clk	10G, 1G/10G, 1G/2.5G/10G, 1G/ 2.5G/5G/10G (USXGMII), 10M/100M/1G/ 10G, 10M/ 100M/1G/ 2.5G/10G	In	1	312.5 MHz clock for the MAC RX datapath when the **Enable 10GBASE-R register mode** is disabled. You may use the same clock source for this clock and tx_312_5_clk.
rx_xcvr_clk	10G	In	1	322.265625 MHz clock for the MAC RX datapath when the **Enable 10GBASE-R register mode** is enabled.
rx_156_25_clk	10G, 1G/10G, 1G/2.5G/10G, 1G/ 2.5G/5G/10G (USXGMII), 10M/100M/1G/ 10G, 10M/ 100M/1G/ 2.5G/10G	In	1	156.25MHz clock for the MAC RX datapath when you choose to maintain compatibility with the 64-bit Ethernet 10G MAC on the Avalon-ST RX data interface or XGMII. This feature is not available when the **Enable 10GBASE-R register mode** is enabled. Intel recommends that you use the same clock source for this clock and rx_312_5_clk. This clock must be synchronous to rx_312_5_clk. Their rising edges must align and must have 0 ppm and phase-shift.
	1G/2.5G, 10M/ 100M/1G/2.5G	In	1	156.25 MHz clock for the Avalon-ST RX data interface.
rx_rst_n	All	In	1	Active-low reset in the rx_312_5_clk clock domain for the MAC RX datapath. For the reset requirements, refer to the related links.
csr_clk	10G, 1G/10G, 1G/2.5G/10G, 1G/ 2.5G/5G/10G (USXGMII), 10M/100M/1G/ 10G, 10M/ 100M/1G/ 2.5G/10G	In	1	Clock for the Avalon-MM control and status interface. Intel recommends that this clock operates within 125 - 156.25 MHz. A lower frequency might result in inaccurate statistics especially when you are using register-based statistics counters.
	1G/2.5G, 10M/ 100M/1G/2.5G	In	1	125 MHz clock for the Avalon-MM control and status interface.
csr_rst_n	All	In	1	Active-low asynchronous reset signal for the csr_clk domain. This signal acts as a global reset for the MAC IP core. For the reset requirements, refer to the related links.

Related Links

- Reset Requirements on page 48
- Avalon-ST Data Interface Clocks on page 93
- IEEE 1588v2 Interface Clocks on page 108

5.2 Speed Selection Signal

Table 34. Speed Selection Signal

Signal	Operating Mode	Direction	Width	Description
`speed_sel`	10G, 1G/10G,10M/100M/1G/10G	In	2	Connect this asynchronous signal to the PHY to obtain the PHY's speed: • 0x0 = 10 Gbps • 0x1 = 1 Gbps • 0x2 = 100 Mbps • 0x3 = 10 Mbps • 0x4 = 2.5 Gbps • 0x5 = 5 Gbps The speed_sel signal can be synchronized to TX or RX clock of the LL Ethernet 10G MAC IP. Before the speed change, make sure the MAC TX and RX datapaths are idle with no packet transmission. After the line rate changes, trigger a reset on the TX and RX datapaths by asserting these active-low reset signals, `tx_rst_n` and `rx_rst_n`.
	1G/2.5G, 1G/2.5G/10G, 1G/2.5G/5G/10G (USXGMII), 10M/100M/1G/2.5G, 10M/100M/1G/2.5G/10G	In	3	

5.3 Error Correction Signals

The error correction signals are present only when you turn on the ECC option.

Table 35. Error Correction Signals

Signal	Direction	Width	Description
ecc_err_det_corr	Out	1	The MAC IP core can indicate detected and corrected ECC errors using the ecc_status register, or both the register and this signal. This signal indicates the state of the ecc_status[0] register bit when the ecc_enable[0] register bit is set to 1. This signal is 0 when the ecc_enable[0] register bit is set to 1.
ecc_err_det_uncorr	Out	1	The MAC IP core can indicate detected and uncorrected ECC errors using the ecc_status register, or both the register and this signal. This signal indicates the state of the ecc_status[1] register bit when the ecc_enable[1] register bit is set to 1. This signal is 0 when the ecc_enable[1] register bit is set to 1.

5.4 Unidirectional Signals

The signals below are present when you turn on the **Unidirectional feature** option. This feature is available only in 10G operating mode.

Table 36. Unidirectional Signals

Signal	Direction	Width	Description
unidirectional_en	Out	1	When asserted, this signal indicates the state of the tx_unidir_control register bit 0.
unidirectional_remote_fault_dis	Out	1	When asserted, this signal indicates the state of the tx_unidir_control register bit 1.
unidirectional_force_remote_fault	Out	1	When asserted, this signal indicates the state of the tx_unidir_control register bit 2.

5.5 Avalon-MM Programming Signals

Table 37. Avalon-MM Programming Signals

Signal	Direction	Width	Description
csr_address[]	In	10/2	Use this bus to specify the register address to read from or write to. The width is 13 bits when you enable the **Use Legacy Ethernet 10G MAC Avalon Memory-Mapped Interface** option.
csr_read	In	1	Assert this signal to request a read.
csr_readdata[]	Out	32	Data read from the specified register. The data is valid when thecsr_waitrequest signal is deasserted.

continued...

Signal	Direction	Width	Description
csr_write	In	1	Assert this signal to request a write.
csr_writedata[]	In	32	Data to be written to the specified register. The data is written when the csr_waitrequest signal is deasserted.
csr_waitrequest	Out	1	When asserted, this signal indicates that the MAC IP core is busy and not ready to accept any read or write requests. • When you have requested for a read or write, keep the control signals to the Avalon-MM interface constant while this signal is asserted. The request is complete when it is deasserted. • This signal can be high or low during idle cycles and reset. Therefore, the user application must not make any assumption of its assertion state during these periods.

5.6 Avalon-ST Data Interfaces

5.6.1 Avalon-ST TX Data Interface Signals

Table 38. Avalon-ST TX Data Interface Signals

Signal	Direction	Width	Description
avalon_st_tx_startofpacket	In	1	Assert this signal to indicate the beginning of the TX data.
avalon_st_tx_endofpacket	In	1	Assert this signal to indicate the end of the TX data.
avalon_st_tx_valid	In	1	Assert this signal to indicate that the avalon_st_tx_data[] signal and other signals on this interface are valid.
avalon_st_tx_ready	Out	1	When asserted, indicates that the MAC IP core is ready to accept data. The reset value of this signal is non-deterministic.
avalon_st_tx_error	In	1	Assert this signal to indicate that the current TX packet contains errors.
avalon_st_tx_data[]	In	32/64	TX data from the client. The client sends the TX data to the MAC IP core in this order: avalon_st_tx_data[31:24], avalon_st_tx_data[23:16], and so forth. The width is 64 bits when you enable the **Use 64-bit Ethernet 10G MAC Avalon Streaming Interface** option. Otherwise, it is 32 bits
avalon_st_tx_empty[]	In	2/3	Use this signal to specify the number of empty bytes in the cycle that contain the end of the TX data. • 0x0: All bytes are valid. • 0x1: The last byte is invalid. • 0x2: The last two bytes are invalid. • 0x3: The last three bytes are invalid. The width is 3 bits when you enable the **Use 64-bit Ethernet 10G MAC Avalon Streaming Interface** option. Otherwise, it is 2 bits.

5.6.2 Avalon-ST RX Data Interface Signals

Table 39. Avalon-ST RX Data Interface Signals

Signal	Direction	Width	Description
avalon_st_rx_startofpacket	Out	1	When asserted, indicates the beginning of the RX data.
avalon_st_rx_endofpacket	Out	1	When asserted, indicates the end of the RX data.
avalon_st_rx_valid	Out	1	When asserted, indicates that the avalon_st_rx_data[] signal and other signals on this interface are valid.
avalon_st_rx_ready	In	1	Assert this signal when the client is ready to accept data.
avalon_st_rx_error[]	Out	6	This signal indicates one or more errors in the current packet being transferred on the Avalon-ST RX interface. It is qualified by the avalon_st_rx_valid and avalon_st_rx_ready signals and aligned to the end of packet. • Bit 0—PHY error. — For 10 Gbps, the data on xgmii_rx_data contains a control error character (FE). — For 10 Mbps,100 Mbps,1 Gbps, gmii_rx_err or mii_rx_err is asserted. — For 1G/2.5G, gmii16b_rx_err is asserted. • Bit 1—CRC error. The computed CRC value does not match the CRC received. • Bit 2—Undersized frame. The RX frame length is less than 64 bytes. • Bit 3—Oversized frame. • Bit 4—Payload length error. • Bit 5—Overflow error. The user application is not ready to receive more data while still receiving incoming data from the MAC IP core.
avalon_st_rx_data[]	Out	32/64	RX data to the client. The MAC IP core sends the RX data to the client in this order: avalon_st_rx_data[31:24], avalon_st_rx_data[23:16], and so forth. The width is 64 bits when you enable the **Use 64-bit Ethernet 10G MAC Avalon Streaming Interface** option. Otherwise, it is 32 bits
avalon_st_rx_empty[]	Out	2/3	Contains the number of empty bytes during the cycle that contain the end of the RX data. The width is 3 bits when you enable the **Use 64-bit Ethernet 10G MAC Avalon Streaming Interface** option. Otherwise, it is 2 bits.

Related Links

Frame Type Checking on page 41

5.6.3 Avalon-ST Data Interface Clocks

Table 40. Clock Signals for the Avalon-ST Data Interfaces

Interface Signal	Mode	Use legacy Ethernet 10G MAC Avalon Streaming interface Option	Clock Signal
avalon_st_tx_*	1G	On	tx_156_25_clk
		Off	tx_312_5_clk
	10G	On	tx_156_25_clk
		Off	tx_312_5_clk
avalon_st_rx_*	1G	On	rx_156_25_clk
		Off	rx_312_5_clk
	10G	On	rx_156_25_clk
		Off	rx_312_5_clk

Related Links

Clock and Reset Signals on page 87

5.7 Avalon-ST Flow Control Signals

Table 41. Avalon-ST Flow Control Signals

Signal	Operating Mode	Direction	Width	Description
avalon_st_pause_data[]	All	In	2	This signal takes effect when the register bits, tx_pauseframe_enable[2:1], are both set to the default value 0. Set this signal to the following values to trigger the corresponding actions. • 0x0: Stops pause frame generation. • 0x1: Generates an XON pause frame. • 0x2: Generates an XOFF pause frame. The MAC IP core sets the pause quanta field in the pause frame to the value in the tx_pauseframe_quanta register. • 0x3: Reserved.
avalon_st_tx_pause_length_valid	All	In	1	This signal is present in the MAC TX only variation. Assert this signal to request the MAC IP core to suspend data transmission. When you assert this signal, ensure that a valid pause quanta is available on the avalon_st_tx_pause_length_data bus.
avalon_st_tx_pause_length_data[]	All	In	16	This signal is present in the MAC TX only variation. Use this bus to specify the pause quanta in unit of quanta, where 1 unit = 512 bits time.
avalon_st_tx_pfc_gen_data[]	10G	In	*n* (4–16)	*n* = 2 x **Number of PFC queues** parameter. Each pair of bits is associated with a priority queue. Bits 0 and 1 are for priority queue 0, bits 2 and 3 are for priority queue 1, and so forth. Set the respective pair of bits to the following values to trigger the specified actions for the corresponding priority queue.

continued...

Signal	Operating Mode	Direction	Width	Description
				• 0x0: Stops pause frame generation for the corresponding queue. • 0x1: Generates an XON pause frame for the corresponding queue. • 0x2: Generates an XOFF pause frame for the corresponding queue. The MAC IP core sets the pause quanta field in the pause frame to the value in the `tx_pauseframe_quanta` register. • 0x3: Reserved.
`avalon_st_rx_pfc_pause_data[]`	10G	Out	n (2–8)	n = **Number of PFC queues** parameter. When the MAC RX receives a pause frame, it asserts bit n of this signal when the pause quanta for the n^{th} queue is valid (Pause Quanta Enable [n] = 1) and greater than 0. For each quanta unit, the MAC RX asserts bit n for eight clock cycle. The MAC RX deasserts bit n of this signal when the pause quanta for the n^{th} queue is valid (Pause Quanta Enable [n] = 1) and equal to 0. The MAC RX also deasserts bit n when the timer expires.
`avalon_st_rx_pause_length_valid`	All	Out	1	This signal is present in the MAC RX only variation. The MAC IP core asserts this signal to request its link partner to suspend data transmission. When asserted, a valid pause quanta is available on the `avalon_st_rx_pause_length_data` bus.
`avalon_st_rx_pause_length_data[]`	All	Out	16	This signal is present only in the MAC RX only variation. Specifies the pause quanta in unit of quanta, where 1 unit = 512 bits time.

5.8 Avalon-ST Status Interface

5.8.1 Avalon-ST TX Status Signals

Table 42. Avalon-ST TX Status Signals

Signal	Direction	Width	Description
`avalon_st_txstatus_valid`	Out	1	When asserted, this signal qualifies the `avalon_st_txstatus_data[]` and `avalon_st_txstatus_error[]` signals.
`avalon_st_txstatus_data[]`	Out	40	Contains information about the TX frame. • Bits 0 to 15: Payload length. • Bits 16 to 31: Packet length. • Bit 32: When set to 1, indicates a stacked VLAN frame. Ignore this bit when the MAC is configured not to detect stacked VLAN frames (`tx_vlan_detection[0]` = 1). • Bit 33: When set to 1, indicates a VLAN frame. Ignore this bit when the MAC is configured not to detect VLAN frames (`tx_vlan_detection[0]` = 1). • Bit 34: When set to 1, indicates a control frame. • Bit 35: When set to 1, indicates a pause frame. • Bit 36: When set to 1, indicates a broadcast frame. • Bit 37: When set to 1, indicates a multicast frame. • Bit 38: When set to 1, indicates a unicast frame. • Bit 39: When set to 1, indicates a PFC frame.

continued...

Signal	Direction	Width	Description
			This status signal is valid only if the TX frame is valid. For example, bit 35 is not asserted if a pause frame is oversized.
`avalon_st_txstatus_error[]`	Out	7	When set to 1, the respective bit indicates the following error type in the TX frame: • Bit 0: Undersized frame. • Bit 1: Oversized frame. • Bit 2: Payload length error. • Bit 3: Unused. • Bit 4: Underflow. • Bit 5: The `avalon_st_tx_error` input signal from client is asserted. • Bit 6: Unused. The error status is invalid when an overflow occurs.
`avalon_st_tx_pfc_status_valid`	Out	1	When asserted, this signal qualifies the `avalon_st_tx_pfc_status_data[]` signal. This signal applies only to 10G operating mode.
`avalon_st_tx_pfc_status_data[]`	Out	*n* (4 - 16)	*n* = 2 × **Number of PFC queues** parameter. When set to 1, the respective bit indicates the flow control request to the remote partner, for example: • Bit 0: XON request for priority queue 0 • Bit 1: XOFF request for priority queue 0 • Bit 2: XON request for priority queue 1 • Bit 3: XOFF request for priority queue 1 • Bit 4: XON request for priority queue 2 • Bit 5: XOFF request for priority queue 2 This signal applies only to the 10G operating mode.

Related Links

Length Checking on page 41
Describes how the MAC IP core checks the frame and payload lengths.

5.8.2 Avalon-ST RX Status Signals

Table 43. Avalon-ST RX Status Signals

Signal	Direction	Width	Description
`avalon_st_rxstatus_valid`	Out	1	When asserted, this signal qualifies the `avalon_st_rxstatus_data[]` and `avalon_st_rxstatus_error[]` signals. The MAC IP core asserts this signal in the same clock cycle the `avalon_st_rx_endofpacket` signal is asserted.
`avalon_st_rxstatus_data[]`	Out	40	Contains information about the RX frame. • Bits 0 to 15: Payload length. • Bits 16 to 31: Packet length. • Bit 32: When set to 1, indicates a stacked VLAN frame. Ignore this bit when the MAC is configured not to detect stacked VLAN frames (`tx_vlan_detection[0]` = 1). • Bit 33: When set to 1, indicates a VLAN frame. Ignore this bit when the MAC is configured not to detect VLAN frames (`tx_vlan_detection[0]` = 1). • Bit 34: When set to 1, indicates a control frame. • Bit 35: When set to 1, indicates a pause frame.

continued...

Signal	Direction	Width	Description
			• Bit 36: When set to 1, indicates a broadcast frame. • Bit 37: When set to 1, indicates a multicast frame. • Bit 38: When set to 1, indicates a unicast frame. • Bit 39: When set to 1, indicates a PFC frame.
`avalon_st_rxstatus_error[]`	Out	7	When set to 1, the respective bit indicates the following error type in the RX frame. • Bit 0: Undersized frame. • Bit 1: Oversized frame. • Bit 2: Payload length error. • Bit 3: CRC error. • Bit 4: Unused. • Bit 5: Unused. • Bit 6: PHY error. The IP core presents the error status on this bus in the same clock cycle it asserts the `avalon_st_rxstatus_valid` signal. The error status is invalid when an overflow occurs.
`avalon_st_rx_pfc_status_valid`	Out	1	When asserted, this signal qualifies the `avalon_st_rx_pfc_status_data[]` signal. This signal applies only to 10G operating mode.
`avalon_st_rx_pfc_status_data[]`	Out	*n* (4 - 16)	*n* = 2 x **Number of PFC queues** parameter When set to 1, the respective bit indicates the flow control request from the remote partner, for example: • Bit 0: XON request for priority queue 0. • Bit 1: XOFF request for priority queue 0. • Bit 2: XON request for priority queue 1. • Bit 3: XOFF request for priority queue 1. • Bit 4: XON request for priority queue 2. • Bit 5: XOFF request for priority queue 2. This signal applies only to 10G operating mode.

Related Links

Length Checking on page 41
Describes how the MAC IP core checks the frame and payload lengths.

5.9 PHY-side Interfaces

5.9.1 XGMII TX Signals

The signals below are present in the following operating modes: 10G, 1G/10G, 1G/2.5G/10G, 1G/2.5G/5G/10G (USXGMII), 10M/100M/1G/10G, and 10M/100M/1G/2.5G/10G.

Table 44. XGMII TX Signals

Signal	Condition	Direction	Width	Description
xgmii_tx_data[]	**Use legacy Ethernet 10G MAC XGMII interface** disabled. **Enable 10GBASE-R register mode** disabled.	Out	32	4-lane data bus. Lane 0 starts from the least significant bit. • Lane 0: xgmii_tx_data[7:0] • Lane 1: xgmii_tx_data[15:8] • Lane 2: xgmii_tx_data[23:16] • Lane 3: xgmii_tx_data[31:24]
	Use legacy Ethernet 10G MAC XGMII interface disabled. **Enable 10GBASE-R register mode** enabled.	Out	64	8-lane SDR XGMII transmit data. This signal connects directly to the NativePHY IP core. • Lane 0: xgmii_tx_data[7:0] • Lane 1: xgmii_tx_data[15:8] • Lane 2: xgmii_tx_data[23:16] • Lane 3: xgmii_tx_data[31:24] • Lane 4: xgmii_tx_data[39:32] • Lane 5: xgmii_tx_data[47:40] • Lane 6: xgmii_tx_data[55:48] • Lane 7: xgmii_tx_data[63:56]
xgmii_tx_control[]	**Use legacy Ethernet 10G MAC XGMII interface** disabled. **Enable 10GBASE-R register mode** disabled.	Out	4	Control bits for each lane in xgmii_tx_data[]. • Lane 0: xgmii_tx_control[0] • Lane 1: xgmii_tx_control[1] • Lane 2: xgmii_tx_control[2] • Lane 3: xgmii_tx_control[3]
	Use legacy Ethernet 10G MAC XGMII interface disabled. **Enable 10GBASE-R register mode** enabled.	Out	8	8-lane SDR XGMII transmit control. This signal connects directly to the NativePHY IP core. • Lane 0: xgmii_tx_control[0] • Lane 1: xgmii_tx_control[1] • Lane 2: xgmii_tx_control[2] • Lane 3: xgmii_tx_control[3] • Lane 4: xgmii_tx_control[4] • Lane 5: xgmii_tx_control[5] • Lane 6: xgmii_tx_control[6] • Lane 7: xgmii_tx_control[7]
xgmii_tx_valid	**Use legacy Ethernet 10G MAC XGMII interface** disabled.	Out	1	When asserted, indicates that the data and control buses are valid.

continued...

Signal	Condition	Direction	Width	Description
	(**Enable 10GBASE-R register mode** enabled or **Speed** is set to **1G/ 2.5G/5G/10G (USXGMII)**)			
`xgmii_tx[]`	**Use legacy Ethernet 10G MAC XGMII interface** enabled.	Out	72	8-lane SDR XGMII transmit data and control bus. Each lane contains 8 data plus 1 control bits. The signal mapping is compatible with the 64b MAC. • Lane 0 data: `xgmii_tx[7:0]` • Lane 0 control: `xgmii_tx[8]` • Lane 1 data: `xgmii_tx[16:9]` • Lane 1 control: `xgmii_tx[17]` • Lane 2 data: `xgmii_tx[25:18]` • Lane 2 control: `xgmii_tx[26]` • Lane 3 data: `xgmii_tx[34:27]` • Lane 3 control: `xgmii_tx[35]` • Lane 4 data: `xgmii_tx[43:36]` • Lane 4 control: `xgmii_tx[44]` • Lane 5 data: `xgmii_tx[52:45]` • Lane 5 control: `xgmii_tx[53]` • Lane 6 data: `xgmii_tx[61:54]` • Lane 6 control: `xgmii_tx[62]` • Lane 7 data: `xgmii_tx[70:63]` • Lane 7 control: `xgmii_tx[71]`
`link_fault_status_xgmii_tx_data[]`	—	In	2	This signal is present in the MAC TX only variation. Connect this signal to the corresponding RX client logic to handle the local and remote faults. The following values indicate the link fault status: • 0x0: No link fault • 0x1: Local fault • 0x2: Remote fault

5.9.2 XGMII RX Signals

The signals below are present in the following operating modes: 10G, 1G/10G, 1G/2.5G/10G, 1G/2.5G/5G/10G (USXGMII), 10M/100M/1G/10G, and 10M/100M/1G/2.5G/10G.

Table 45. XGMII Receive Signals

Signal	Condition	Direction	Width	Description
xgmii_rx_data[]	**Use legacy Ethernet 10G MAC XGMII interface** disabled. **Enable 10GBASE-R register mode** disabled.	In	32	4-lane RX data bus. Lane 0 starts from the least significant bit. • Lane 0: xgmii_rx_data[7:0] • Lane 1: xgmii_rx_data[15:8] • Lane 2: xgmii_rx_data[23:16] • Lane 3: xgmii_rx_data[31:24]
	Use legacy Ethernet 10G MAC XGMII interface disabled. **Enable 10GBASE-R register mode** enabled.	In	64	8-lane SDR XGMII receive data. This signal connects directly to the Native PHY IP core. • Lane 0: xgmii_rx_data[7:0] • Lane 1: xgmii_rx_data[15:8] • Lane 2: xgmii_rx_data[23:16] • Lane 3: xgmii_rx_data[31:24] • Lane 4: xgmii_rx_data[39:32] • Lane 5: xgmii_rx_data[47:40] • Lane 6: xgmii_rx_data[55:48] • Lane 7: xgmii_rx_data[63:56]
xgmii_rx_control[]	**Use legacy Ethernet 10G MAC XGMII interface** disabled. **Enable 10GBASE-R register mode** disabled.	In	4	Control bits for each lane in xgmii_rx_data[]. • Lane 0: xgmii_rx_control[0] • Lane 1: xgmii_rx_control[1] • Lane 2: xgmii_rx_control[2] • Lane 3: xgmii_rx_control[3]
	Use legacy Ethernet 10G MAC XGMII interface disabled. **Enable 10GBASE-R register mode** enabled.	In	8	8-lane SDR XGMII receive control. This signal connects directly to the NativePHY IP core. • Lane 0: xgmii_rx_control[0] • Lane 1: xgmii_rx_control[1] • Lane 2: xgmii_rx_control[2] • Lane 3: xgmii_rx_control[3] • Lane 4: xgmii_rx_control[4] • Lane 5: xgmii_rx_control[5] • Lane 6: xgmii_rx_control[6] • Lane 7: xgmii_rx_control[7]
xgmii_rx_valid	**Use legacy Ethernet 10G MAC XGMII interface** disabled.	In	1	When asserted, indicates that the data and control buses are valid.

continued...

Signal	Condition	Direction	Width	Description
	(**Enable 10GBASE-R register mode** enabled or **Speed** is set to **1G/ 2.5G/5G/10G (USXGMII)**)			
xgmii_rx[]	**Use legacy Ethernet 10G MAC XGMII interface** enabled.	In	72	8-lane SDR XGMII receive data and control bus. Each lane contains 8 data plus 1 control bits. The signal mapping is compatible with the 64-bit MAC. • Lane 0 data: xgmii_rx[7:0] • Lane 0 control: xgmii_rx[8] • Lane 1 data: xgmii_rx[16:9] • Lane 1 control: xgmii_rx[17] • Lane 2 data: xgmii_rx[25:18] • Lane 2 control: xgmii_rx[26] • Lane 3 data: xgmii_rx[34:27] • Lane 3 control: xgmii_rx[35] • Lane 4 data: xgmii_rx[43:36] • Lane 4 control: xgmii_rx[44] • Lane 5 data: xgmii_rx[52:45] • Lane 5 control: xgmii_rx[53] • Lane 6 data: xgmii_rx[61:54] • Lane 6 control: xgmii_rx[62] • Lane 7 data: xgmii_rx[70:63] • Lane 7 control: xgmii_rx[71]
link_fault_status_ xgmii_rx_data[]	—	Out	2	The following values indicate the link fault status: • 0x0 = No link fault • 0x1 = Local fault • 0x2 = Remote fault

5.9.3 GMII TX Signals

Table 46. GMII TX Signals

Signal	Operating Mode	Direction	Width	Description
gmii_tx_clk	• 1G/10G • 10M/ 100M/1G/ 10G	In	1	125 MHz TX clock.
gmii_tx_d[]		Out	8	TX data.
gmii_tx_en		Out	1	When asserted, indicates the TX data is valid.
gmii_tx_err		Out	1	When asserted, indicates the TX data contains error.

continued...

Signal	Operating Mode	Direction	Width	Description
gmii16b_tx_clk	• 1G/2.5G • 1G/ 2.5G/10G (MGBASE-T) • 10M/ 100M/1G/ 2.5G • 10M/ 100M/1G/ 2.5G/10G (MGBASE-T)	In	1	156.25 MHz TX clock for 2.5G; 62.5 MHz TX clock for 1G; 62.5 MHz TX clock for 10M/100M/1G.
gmii16b_tx_d[]		Out	16	TX data.
gmii16b_tx_en		Out	2	When asserted, indicates the TX data is valid.
gmii16b_tx_err		Out	2	When asserted, indicates the TX data contains error.

5.9.4 GMII RX Signals

Table 47. GMII RX Signals

Signal	Operating Mode	Direction	Width	Description
gmii_rx_clk	• 1G/10G • 10M/ 100M/1G/ 10G	In	1	125 MHz RX clock.
gmii_rx_d[]		In	8	RX data.
gmii_rx_dv		In	1	When asserted, indicates the RX data is valid.
gmii_rx_err		In	1	When asserted, indicates the RX data contains error.
gmii16b_rx_clk	• 1G/2.5G • 1G/ 2.5G/10G (MGBASE-T) • 10M/ 100M/1G/ 2.5G • 10M/ 100M/1G/ 2.5G/10G (MGBASE-T)	In	1	156.25 MHz RX clock for 2.5G; 62.5 MHz RX clock for 1G; 62.5 MHz RX clock for 10M/100M/1G.
gmii16b_rx_d[]		In	16	RX data.
gmii16b_rx_dv		In	2	When asserted, indicates the RX data is valid.
gmii16b_rx_err		In	2	When asserted, indicates the RX data contains error.

5.9.5 MII TX Signals

The signals below are present in the 10M/100M/1G/10G, 10M/100M/1G/2.5G, and 10M/100M/1G/2.5G/10G operating modes.

Note: For 10M/100M/1G/2.5G and 10M/100M/1G/2.5G/10G variants, only tx_clkena signal is available.

Table 48. MII TX Signals

Signal	Direction	Width	Description
tx_clkena	In	1	Clock enable from the PHY IP. This clock effectively divides gmii_tx_clk to 25 MHz for 100 Mbps and 2.5 MHz for 10 Mbps. For 10M/100M/1G/2.5G/10G and 10M/100M/1G/2.5G variants, this clock effectively divides gmii16b_tx_clk to 6.25 Mhz for 100 Mbps and 0.625 MHz for 10 Mbps.
tx_clkena_half_rate	In	1	Clock enable from the PHY IP. This clock effectively divides gmii_tx_clk to 12.5 MHz for 100 Mbps and 1.25 MHz for 10 Mbps.
mii_tx_d[]	Out	4	TX data bus.
mii_tx_en	Out	1	When asserted, indicates the TX data is valid.
mii_tx_err	Out	1	When asserted, indicates the TX data contains error.

5.9.6 MII RX Signals

The signals below are present in the 10M/100M/1G/10G, 10M/100M/1G/2.5G, and 10M/100M/1G/2.5G/10G operating modes.

Note: For 10M/100M/1G/2.5G and 10M/100M/1G/2.5G/10G variants, only rx_clkena signal is available.

Table 49. MII RX Signals

Signal	Direction	Width	Description
rx_clkena	In	1	Clock enable from the PHY IP for 100 Mbps and 10 Mbps operations. This clock effectively divides gmii_rx_clk to 25 MHz for 100 Mbps and 2.5 MHz for 10 Mbps. For 10M/100M/1G/2.5G/10G and 10M/100M/1G/2.5G variants, this clock effectively divides gmii16b_rx_clk to 6.25 Mhz for 100 Mbps and 0.625 MHz for 10 Mbps.
rx_clkena_half_rate	In	1	Clock enable from the PHY IP for 100 Mbps and 10 Mbps operations. This clock effectively runs at half the rate of rx_clkena and divides gmii_rx_clk to 12.5 MHz for 100 Mbps and 1.25 MHz for 10 Mbps. The rising edges of this signal and rx_clkena must align.
mii_rx_d[]	Out	4	RX data bus.
mii_rx_dv	Out	1	When asserted, indicates the RX data is valid.
mii_rx_err	Out	1	When asserted, indicates the RX data contains error.

5.10 IEEE 1588v2 Interfaces

5.10.1 IEEE 1588v2 Egress TX Signals

The signals below are present when you select the **Enable time stamping** option. This feature is available in the following operating modes: 10G, 1G/10G, 10M/100M/1G/10G, 1G/2.5G, and 1G/2.5G/10G (Intel Stratix 10 devices only).

Table 50. IEEE 1588v2 Egress TX Signals

Signal	Direction	Width	Description
tx_egress_timestamp_request_valid	In	1	Assert this signal to request for a timestamp for the transmit frame. This signal must be asserted in the same clock cycle avalon_st_tx_startofpacket is asserted.
tx_egress_timestamp_request_fingerprint[]	In	*n*	*n* = value of the **Timestamp fingerprint width** parameter. Use this bus to specify the fingerprint of the transmit frame that you are requesting a timestamp for. This bus must carry a valid fingerprint at the same time tx_egress_timestamp_request_valid is asserted. The purpose of the fingerprint is to associate the timestamp with the packet. Thus, it can be the sequence ID field from the PTP packet or some other unique field of the packet, to validate both the fingerprint and timestamp collected from the CPU.
tx_egress_timestamp_96b_valid	Out	1	When asserted, this signal qualifies the timestamp on tx_egress_timestamp_96b_data[] for the transmit frame whose fingerprint is specified by tx_egress_timestamp_96b_fingerprint[] .
tx_egress_timestamp_96b_data[]	Out	96	Carries the 96-bit egress timestamp in the following format: • For 1588v2 format: — Bits 48 to 95: 48-bit seconds field — Bits 16 to 47: 32-bit nanoseconds field — Bits 0 to 15: 16-bit fractional nanoseconds field • For 1588v1 format: — Bits 64 to 95: 32-bit seconds field — Bits 32 to 63: 32-bit nanoseconds field — Bits 0 to 31: Unused
tx_egress_timestamp_96b_fingerprint[]	Out	*n*	*n* = value of the **Timestamp fingerprint width** parameter. The fingerprint of the transmit frame, which is received on tx_egress_timestamp_request_data[]. This fingerprint specifies the transmit frame the egress timestamp on tx_egress_timestamp_96b_data[] is for.

continued...

Signal	Direction	Width	Description
`tx_egress_timestamp_64b_valid`	Out	1	When asserted, this signal qualifies the timestamp on `tx_egress_timestamp_64b_data[]` for the transmit frame whose fingerprint is specified by `tx_egress_timestamp_64b_fingerprint[]`.
`tx_egress_timestamp_64b_data[]`	Out	64	Carries the 64-bit egress timestamp in the following format: • Bits 16 to 63: 48-bit nanoseconds field • Bits 0 to 15: 16-bit fractional nanoseconds field
`tx_egress_timestamp_64b_fingerprint[]`	Out	*n*	*n* = value of the **Timestamp fingerprint width** parameter. The fingerprint of the transmit frame, which is received on `tx_egress_timestamp_request_data[]`. This fingerprint specifies the transmit frame the egress timestamp on `tx_egress_timestamp_64b_data[]` signal is for.
`tx_time_of_day_96b_10g_data` (for 10 Gbps)	In	96	Carries the time of day (ToD) from an external ToD module to the MAC IP core in the following format: • Bits 48 to 95: 48-bit seconds field • Bits 16 to 47: 32-bit nanoseconds field • Bits 0 to 15: 16-bit fractional nanoseconds field This is required for noting the timestamp ToD which is of 80-bit, consisting of seconds and nanoseconds, in the respective field of the PTP packet. The remaining 16-bit fractional nanoseconds value, if used, is for updating the CF of the PTP packet.
`tx_time_of_day_96b_1g_data` (for 10 Mbps, 100 Mbps, 1, and 2.5 Gbps)			
`tx_time_of_day_64b_10g_data` (for 10 Gbps)	In	64	Carries the ToD from an external ToD module to the MAC IP core in the following format: • Bits 16 to 63: 48-bit nanoseconds field • Bits 0 to 15: 16-bit fractional nanoseconds field The 64-bit timestamp is required to update the CF in the PTP header. Updating the CF is fundamental to the transparent clock operation.
`tx_time_of_day_64b_1g_data` (for 10 Mbps, 100 Mbps, 1, and 2.5 Gbps)			
`tx_path_delay_10g_data` (for 10 Gbps)	In	16	Connect this bus to the Intel FPGA PHY IP. This bus carries the path delay, which is measured between the physical network and the PHY side of the MAC IP Core (XGMII, GMII, or MII). The MAC IP core uses this value when generating the egress timestamp to account for the delay. The path delay is in the following format: • Bits 0 to 9: Fractional number of clock cycle • Bits 10 to 15/21: Number of clock cycle
`tx_path_delay_1g_data` (for 10 Mbps, 100 Mbps, 1, and 2.5 Gbps)		22	
`tx_egress_p2p_update`	In	1	Assert this signal when the CF needs to be added with <meanPathDelay> given by `tx_egress_p2p_val` for a transmit frame, as part of peer-to-peer mechanism.

continued...

Signal	Direction	Width	Description
			Assert this signal in the same clock cycle as the start of packet (`avalon_st_tx_startofpacket` is asserted).
`tx_egress_p2p_val[]`	In	46	This represents <`meanPathDelay`> for peer to peer operations. • Bits 16 to 45: Link delay in nanoseconds field • Bits 0 to 15: Link delay in fractional nanoseconds field

Table 51. IEEE 1588v2 Egress TX Signals—1-step Mode

These signals apply to 1-step operation mode only.

Signal	Direction	Width	Description
`tx_etstamp_ins_ctrl_timestamp_insert`	In	1	Assert this signal to insert egress timestamp into the associated frame. Assert this signal in the same clock cycle `avalon_st_tx_startofpacket` is asserted.
`tx_etstamp_ins_ctrl_timestamp_format`	In	1	Use this signal to specify the format of the timestamp to be inserted. • 0: 1588v2 format (48-bits second field + 32-bits nanosecond field + 16-bits correction field for fractional nanosecond). Required offset location of timestamp and correction field. • 1: 1588v1 format (32-bits second field + 32-bits nanosecond field). Required offset location of timestamp. 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`	In	1	Assert this signal to add residence time (egress timestamp –ingress timestamp) into correction field of PTP frame. Required offset location of correction field. 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[]`	In	96	96-bit format of ingress timestamp.(48 bits second + 32 bits nanosecond + 16 bits fractional nanosecond). 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[]`	In	64	64-bit format of ingress timestamp. (48-bits nanosecond + 16-bits fractional nanosecond). 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`	In	1	Format of timestamp to be used for residence time calculation. 0: 96-bits (96-bits egress timestamp - 96-bits ingress timestamp). 1: 64-bits (64-bits egress timestamp - 64-bits ingress timestamp).

continued...

Signal	Direction	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_checksum_zero`	In	1	Assert this signal to set the checksum field of UDP/IPv4 to zero. Required offset location of checksum field. 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`	In	1	Assert this signal to correct UDP/IPv6 packet checksum, by updating the checksum correction, which is specified by checksum correction offset. Required offset location of checksum correction. 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[]`	In	16	The location of the timestamp field, relative to the first byte of the packet. 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[]`	In	16	The location of the correction field, relative to the first byte of the packet. 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[]`	In	16	The location of the checksum field, relative to the first byte of the packet. 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[]`	In	16	The location of the checksum correction field, relative to the first byte of the packet. Assert this signal in the same clock cycle as the start of packet (`avalon_st_tx_startofpacket` is asserted).
`tx_egress_asymmetry_update`	In	1	Assert this signal to update the CF in the PTP header of transmit frame with asymmetry value. Assert this signal in the same clock cycle as the start of packet (`avalon_st_tx_startofpacket`) is asserted. For more details, refer to related links about the `tx_asymmetry` register.

Related Links

Timestamp Registers on page 75
Provides more information about the `tx_asymmetry` register.

5.10.2 IEEE 1588v2 Ingress RX Signals

The signals below are present when you select the **Enable time stamping** option. This feature is available in the following operating modes: 10G, 1G/10G, 10M/100M/1G/10G, and 1G/2.5G, and 1G/2.5G/10G (Intel Stratix 10 devices only).

Table 52. IEEE 1588v2 Ingress RX Signals

Signal	Direction	Width	Description
`rx_ingress_timestamp_96b_valid`	Out	1	When asserted, this signal qualifies the timestamp on `rx_ingress_timestamp_96b_data[]`. The MAC IP core asserts this signal in the same clock cycle it asserts `avalon_st_rx_startofpacket`.
`rx_ingress_timestamp_96b_data[]`	Out	96	Carries the 96-bit ingress timestamp in the following format: • Bits 48 to 95: 48-bit seconds field • Bits 16 to 47: 32-bit nanoseconds field • Bits 0 to 15: 16-bit fractional nanoseconds field The 96-bit timestamp is usually for noting the complete ToD and is useful in ordinary clock and boundary clock devices. The transparent clock typically uses 64-bit timestamp.
`rx_ingress_timestamp_64b_valid`	Out	1	When asserted, this signal qualifies the timestamp on `rx_ingress_timestamp_64b_data[]`. The MAC IP core asserts this signal in the same clock cycle it asserts `avalon_st_rx_startofpacket`.
`rx_ingress_timestamp_64b_data[]`	Out	64	Carries the 64-bit ingress timestamp in the following format: • Bits 16 to 63: 48-bit nanoseconds field • Bits 0 to 15: 16-bit fractional nanoseconds field This timestamp is used in transparent clock devices.
`rx_time_of_day_96b_10g_data` (for 10 Gbps)	In	96	Carries the time of day (ToD) from an external ToD module to the MAC IP core in the following format: • Bits 48 to 95: 48-bit seconds field • Bits 16 to 47: 32-bit nanoseconds field • Bits 0 to 15: 16-bit fractional nanoseconds field
`rx_time_of_day_96b_1g_data` (for 10 Mbps, 100 Mbps, 1 Gbps, and 2.5 Gbps)			
`rx_time_of_day_64b_10g_data` (for 10 Gbps)	In	64	Carries the ToD from an external ToD module the MAC IP core in the following format: • Bits 16 to 63: 48-bit nanoseconds field • Bits 0 to 15: 16-bit fractional nanoseconds field
`rx_time_of_day_64b_1g_data` (for 10 Mbps, 100 Mbps, 1 Gbps, and 2.5 Gbps)			
`rx_path_delay_10g_data` (for 10 Gbps)	In	16	Connect this bus to the Intel FPGA PHY IP. This bus carries the path delay (residence time), measured between the physical network and the PHY side of the MAC IP Core (XGMII, GMII, or MII). The MAC IP
`rx_path_delay_1g_data` (for 10 Mbps, 100 Mbps, 1 Gbps, and 2.5 Gbps)		22	

continued...

Signal	Direction	Width	Description
			core uses this value when generating the ingress timestamp to account for the delay. The path delay is in the following format: • Bits 0 to 9: Fractional number of clock cycle • Bits 10 to 15/21: Number of clock cycle
`rx_ingress_p2p_val[]`	Out	46	Represents `<meanPathDelay>` for the current ingress port, which is used for peer-to-peer operations. • Bits 16 to 45: Link delay in nanoseconds field • Bits 0 to 15: Link delay in fractional nanoseconds field
`rx_ingress_p2p_val_valid`	Out	1	When asserted, this signal indicates the `rx_ingress_p2p_val` is valid.

5.10.3 IEEE 1588v2 Interface Clocks

Table 53. Clock Signals for the IEEE 1588V2 Interfaces

Interface Signal	Use legacy Ethernet 10G MAC Avalon Streaming interface Option	Clock Signal
`tx_egress_*` `tx_time_of_day_*_10G_*` `tx_etstamp_ins_*`	On	`tx_156_25_clk`
	Off	`tx_312_5_clk`
`tx_time_of_day_*_1G_*`	On	`gmii_tx_clk`
	Off	
`rx_ingress_*` `rx_time_of_day_*_10G_*`	On	`rx_156_25_clk`
	Off	`rx_312_5_clk`
`rx_time_of_day_*_1G_*`	On	`gmii_rx_clk`
	Off	

Related Links

Clock and Reset Signals on page 87

A Intel FPGA Low Latency Ethernet 10G MAC User Guide Archives

If an IP core version is not listed, the user guide for the previous IP core version applies.

IP Core Version	User Guide
17.0	Low Latency Ethernet 10G MAC User Guide
16.1	Low Latency Ethernet 10G MAC User Guide
16.0	Low Latency Ethernet 10G MAC User Guide
15.1	Low Latency Ethernet 10G MAC User Guide
15.0	Low Latency Ethernet 10G MAC User Guide
14.1	Low Latency Ethernet 10G MAC User Guide

B Document Revision History for Intel FPGA Low Latency Ethernet 10G MAC User Guide

Date	Version	Changes
December 2017	2017.12.25	• Updated the decription in the *10GBASE-R Register Mode* topic: — Added support for Intel Stratix 10 devices. — Added Figure: PHY Configuration with 10GBASE-R with IEEE 1588v2 Enabled for Intel Stratix 10 Devices.
November 2017	2017.11.06	• Renamed the document as *Intel FPGA Low Latency Ethernet 10G MAC User Guide*. • Added support for the Intel Cyclone 10 GX device family. • Added a new feature—Peer-to-Peer: — Added a new parameter—**Enable peer-to-peer support**. — Updated the description in the *TX Datapath* sub-topic of the *IEEE 1588v2* topic. — Added new timestamp registers: • Added new IEEE 1588v2 Egress TX signals— `tx_egress_p2p_update` and `tx_egress_p2p_val[]`. • Added new IEEE 1588v2 Ingress RX signals— `rx_ingress_p2p_val[]` and `rx_ingress_p2p_val_valid`. • Updated the *About LL Ethernet 10G MAC* section: — Updated the *Features* topic. — Updated the "Device Family Support for LL 10GbE MAC" table: Updated the support for Intel Arria 10 and Intel Cyclone 10 GX device families from Preliminary to Final. — Updated the "Device Family Support for Configurations" table: Updated the configuration support for Intel Stratix 10 and Intel Cyclone 10 GX device families. • Updated the *Getting Started* section: — Updated *Creating a Signal Tap Debug File to Match Your Design Hierarchy* topic. — Updated the *Parameter Settings for the LL Ethernet 10G MAC IP Core* topic: • Updated the descriptions of the **Enable ECC on memory blocks**, **Enable time stamping**, **Enable PTP one-step clock support**, **Enable asymmetry support**, **Timestamp fingerprint width**, and **Time of Day Format** parameters. — Added *Jitter on PLL Clocks* topic. • Updated the *Functional Description* section: — Updated the *Payload Length* topic. — Updated the *10GBASE-R Register Mode* topic: Added a note to clarify that 10GBASE-R register mode is not supported in Intel Stratix 10 devices.

continued...

Date	Version	Changes
		• Updated the *Configuration Registers* section: — Updated the "Timestamp Registers" table: Updated the descriptions for the `tx_fns_adjustment_10G`, `tx_ns_adjustment_10G`, `rx_fns_adjustment_10G`, `rx_ns_adjustment_10G`, `tx_fns_adjustment_mult_speed`, `tx_ns_adjustment_mult_speed`, `rx_fns_adjustment_mult_speed`, and `rx_ns_adjustment_mult_speed` registers. — Updated "Hardware PMA Delay" table: Updated the RX latency values for Intel Stratix 10 devices. • Updated for latest branding standards. • Made editorial updates throughout the document.
June 2017	2017.06.19	• Added support for two new variants: — 10M/100M/1G/2.5G — 10M/100M/1G/2.5G/10G • Updated the following tables: — Features Comparison — Device Family Support for Configurations • Updated the Interface Signals figure. • Updated the operating modes for the clock and reset signals in the Clock and Reset Signals table. • Updated descriptions and tables for the following topics: — Parameter Settings for the LL Ethernet 10G MAC IP Core — Unidirectional Signals — XGMII TX Signals — XGMII RX Signals — GMII TX Signals — GMII RX Signals — MII TX Signals — MII RX Signals — IEEE 1588v2 Ingress TX Signals — IEEE 1588v2 Ingress RX Signals • Added `unidirectional_force_remote_fault` signal in the Unidirectional Signals table. • Made minor editorial updates.
May 2017	2017.05.08	• Clarified the device family support for Stratix 10 devices. • Updated the Features topic. • Updated the Device Family Support topic: — Added Intel FPGA IP Core Device Support Levels table. — Updated the LL Ethernet 10G MAC table to include Stratix 10 speed grades with 1588 feature. • Removed the Definition: Device Support Level topic. • Updated the Device Family Support for Configurations table to include additional configurations that support Stratix 10 device family: — 1G/2.5G/10G MAC with 1G/2.5G/10G Multi-rate Ethernet PHY — 1G/2.5G/10G MAC with 1G/2.5G/10G Multi-rate Ethernet PHY and IEEE 1588v2 — 1G/2.5G MAC with 1G/2.5G Multi-rate Ethernet PHY — 1G/2.5G MAC with 2.5G Multi-rate Ethernet PHY — 10M/100M/1G/10G MAC with IEEE 1588v2 — 1G/10G MAC with Backplane Ethernet 10GBASE-KR PHY • Added tables listing resource utilization for Arria 10 and Stratix 10 devices in the Resource Utilization topic. • Updated the Hardware PMA Delay table to include support for Stratix 10 device family.

continued...

Date	Version	Changes
		• Updated the TX Configuration and Status Registers table: — Revised the HW Reset Value of tx_ipg_10g from 0x0 to 0x1. — Revised the HW Reset Value of tx_ipg_10M_100M_1G from 0x0 to 0x0C. • Added tables listing LX and RX latency values for Arria 10 and Stratix 10 devices in the TX and RX latency topic. • Updated the Timestamp Registers table to map registers in order of address. • Updated the Clock and Reset Signals table. • Updated the description for speed_sel signal in the Speed Selection table. • Updated the width and description for csr_address[] in the Avalon-MM Programming Signals table. • Updated the width and description for avalon_st_tx_data[] signal in both Avalon-ST TX Data Interface Signals and Avalon-ST RX Data Interface Signals tables. • Updated Typical Client Frame at TX Interface figure, Endian Conversion figure, and Typical Client Frame at Receive Interface figure.
October 2016	2016.10.31	• Added support for the Stratix 10 device family. • Added 1588 asymmetry support feature. • Corrected the Intel Arria 10 device speed grades from –C2 and –C3 to –E2 and –E3. • Updated the topic about XGMII encapsulation in the TX datapath to clarify that the MAC TX converts the eighth byte of the preamble to a 1-byte SFD. • Added a table listing the MAC behavior for different frame types in the topic about frame type checking. • Updated the topic listing the clock and reset signals to specify the clock domains of the tx_rst_n and rx_rst_n signals. • Updated the avalon_st_txstatus_data[] signal description to clarify that the status is only valid if the TX frame is valid. • Updated the description of avalon_st_txstatus_error[5] to clarify that this bit asserts if the avalon_st_tx_error input signal from client is asserted. • Added tables listing the clocks for the Avalon-ST and IEEE 1588v2 interface signals. • Added the different word offsets for the tx_ipg_10g, tx_ipg_10M_100M_1G, ecc_status, ecc_enable, and mac_reset_control registers if you turn on **Use legacy Ethernet 10G MAC Avalon Memory-Mapped interface**. • Updated document template.
May 2016	2016.05.02	• Updated the following topics to include the new speed mode 1G/2.5G/5G/10G (USXGMII): Features, Device Family Support, Parameter Settings, XGMII TX, and XGMII RX. • Added a new topic: LL Ethernet 10G MAC and Legacy 10-Gbps Ethernet MAC. • Added a new topic: Creating a SignalTap II Debug File to Match Your Design Hierarchy. • Updated the description of the Overflow Handling. • Replace the timing diagram in the XGMII Error Handling topic. • Revised the description of invalid frames in the Statistics Registers topic and removed the tx_stats_etherStatsCRCErr, tx_stats_etherStatsJabbers, tx_stats_etherStatsFragments, and tx_stats_framesCRCErr from the topic. • Removed the PMA Delay from Simulation Model table from the Calculating Timing Adjustment topic because simulation data is not deterministic.

continued...

Date	Version	Changes
November 2015	2015.11.02	• Updated the Features, Device Family Support, Configuration Registers, and Interface Signals topics for 1G/2.5G and 1G/2.5G/10G operating speeds. • Updated the Resource Utilization table. • Revised the description in the Upgrading Outdate IP Cores topic. • Updated the Reset topic, added a step in stage 2. • Updated the Register Access topic, ECC status, and statistics clear register definitions. • Updated the `tx_unidir_control` register to include support for user-triggered remote fault notification. • Removed the Migrating IP Cores to a Different Device topic.
May 2015	2015.05.04	• Update the Device Support table. • Updated the Resource Utilization table. • Updated the Parameter Settings table. • Added instruction on how to read statistics counters in the Statistics Registers topic. • Added new registers: `tx_vlan_detection`, `rx_vlan_detection`, `tx_ipg_10g`, `tx_ipg_10M_100M_1G`, `tx_transfer_status`, and `rx_transfer_status`. • Updated the description of the `rx_stats_octetsOK` and `tx_stats_octetsOK` statistics counters. • Update the Length Checking topic. • Added the Reset Requirements topic. • Added the Deriving TX Timing Adjustments and Deriving RX Timing Adjustments topics. • Removed the Minimum Inter-packet Gap topic.
December 2014	2014.12.15	• Updated the Performance and Resource Utilization table—improved the resource utilization for IEEE 1588v2 feature. • Added a new feature—10GBASE-R register mode: — Added a new parameter—**Enable 10GBASE-R register mode**. — Added new signals—`tx_xcvr_clk`, `rx_xcvr_clk`, `xgmii_tx_valid`, `xgmii_rx_valid`. • Added new parameter options for **Time of Day Format**. • Added a new table in the Frame Type Checking topic to describe the MAC behavior for different frame types. • Added a new table—Register Access Type Convention—to describe the access type for the IP core registers. • Added a new section about timing constraints. • Revised the receive timestamp registers word offset to start from 0x0120 to 0x012C.

continued...

Date	Version	Changes
		• Added a recommendation for the csr_rst_n signal—deassert the csr_rst_n signal at least once after tx_clk and rx_clk are stable. • Revised the number of bits for fractional number of clock cycle for rx_path_delay_10g_data and rx_path_delay_1g_data signals to Bit [9:0]: Fractional number of clock cycle, Bit [21/15:10]:Number of clock cycle. • Updated the signals description for: — tx_egress_timestamp_request_fingerprint[] — tx_egress_timestamp_96b_data[] — tx_egress_timestamp_64b_data[] — tx_time_of_day_96b_1g_data — tx_time_of_day_64b_1g_data
June 2014	2014.06.30	• Improved the performance and resource utilization. • Added a new feature—Unidirectional Ethernet. — Added a new parameter—**Enable Unidirectional feature**. — Added Unidirectional registers and signals. • Added information about PMA analog and digital delay for IEEE 1588v2 MAC registers. • Edited the bit description of avalon_st_rxstatus_error[] signal. • Added more information about the avalon_st_pause_data[0] bit signal to indicate that the transmission of XON pause frames only trigger for one time after XOFF pause frames regardless of how long the avalon_st_pause_data[0] is asserted. • Updated the statistics registers description. • Edited the bit description of tx_underflow_counter0, tx_underflow_counter1, rx_pktovrflow_etherStatsDropEvents,rx_pktovrflow_error signals. • Edited the bit description of csr_clk signal to state that the recommended clock frequency for this signal is 125 Mhz–156.25 Mhz regardless of whether you select register-based or memory-based statistics counter. • Updated the tx_rst_n and rx_rst_n signals description to reflect the change from asynchronous reset to synchronous reset. • Updated the csr_waitrequest signal description.
December 2013	2013.12.02	Initial release