



| F-Tile Ethernet Intel® FPGA Hard IP User Guide

Updated for Intel® Quartus® Prime Design Suite: **23.4**



Online Version



Send Feedback

UG-20313

683023

2023.12.04

Contents

1. Overview.....	5
1.1. Ethernet System in F-Tile Overview.....	9
1.2. F-Tile Ethernet Intel FPGA Hard IP Overview.....	10
1.2.1. Device Family Support.....	11
1.2.2. Device Speed Grade Support.....	11
1.2.3. Resource Utilization.....	13
1.2.4. Round-trip Latency.....	13
1.2.5. Release Information.....	16
2. Getting Started.....	17
2.1. Installing and Licensing F-Tile Ethernet Intel FPGA Hard IP Cores.....	17
2.2. Specifying the IP Core Parameters and Options.....	17
2.3. Reference and System PLL Clock for your IP Design.....	18
2.4. Generated File Structure.....	19
2.4.1. Generating IP-XACT File.....	21
2.5. Generating Tile Files.....	22
2.6. IP Core Testbenches.....	22
3. F-Tile Ethernet Intel FPGA Hard IP Parameters.....	23
4. Functional Description.....	31
4.1. Datapath Description.....	31
4.2. F-Tile Ethernet Intel FPGA Hard IP MAC	31
4.2.1. MAC TX Datapath.....	31
4.2.2. MAC RX Datapath.....	36
4.2.3. Congestion and Flow Control Using PAUSE or Priority Flow Control (PFC).....	39
4.2.4. Link Fault Signaling.....	41
4.2.5. Order of Ethernet Transmission.....	42
4.3. PCS, OTN, and FlexE Modes.....	44
4.3.1. PCS Mode.....	45
4.3.2. OTN Mode.....	45
4.3.3. FlexE Mode.....	45
4.4. Precision Time Protocol.....	46
4.4.1. Features.....	46
4.4.2. PTP Timestamp Accuracy.....	47
4.4.3. PTP Client Flow.....	50
4.4.4. RX Virtual Lane Offset Calculation for No FEC Variants.....	62
4.4.5. Virtual Lane Order and Offset Values.....	63
4.4.6. UI Adjustment.....	64
4.4.7. Reference Time Interval.....	69
4.4.8. Minimum and Maximum Reference Time (TAM) Interval for UI Measurement (Hardware).....	71
4.4.9. UI Value and PMA Delay.....	73
4.4.10. Routing Delay Adjustment for Advanced Timestamp Accuracy Mode.....	74
4.4.11. Routing Delay Adjustment for Basic Timestamp Accuracy Mode.....	75
4.5. Auto-Negotiation and Link Training.....	76

5. Clocks	77
5.1. Clock Connections in Single Instance Operation	81
5.2. Clock Connections in Multiple Instance Operation	82
5.3. Clock Connections in MAC Asynchronous FIFO Operation	84
5.4. Clock Connections in PTP-Based Synchronous and Asynchronous Operation	85
5.5. Clock Connections in Synchronous Ethernet Operation	86
5.6. Custom Cadence	90
6. Resets	94
6.1. Reset Signals	95
6.2. Reset Sequence	96
7. Interface Overview	98
7.1. Status Interface	98
7.2. TX MAC Avalon ST Client Interface	100
7.2.1. TX MAC Avalon ST Client Interface with Disabled Preamble Passthrough	102
7.2.2. TX MAC Avalon ST Client Interface with Enabled Preamble Passthrough	103
7.2.3. Using MAC Avalon ST <code>skip_crc</code> Signal to Control Source Address, PAD, and CRC Insertion	105
7.2.4. Using MAC Avalon ST <code>i_tx_error</code> Signal to Mark Packets Invalid	106
7.3. RX MAC Avalon ST Aligned Client Interface	107
7.3.1. RX MAC Avalon ST Client Interface with Disabled Preamble Passthrough	109
7.3.2. RX MAC Avalon ST Client Interface with Enabled Passthrough and RX CRC Forwarding	110
7.3.3. RX MAC Adapter Limitations	110
7.3.4. RX MAC Avalon ST Client Interface Status	111
7.4. TX MAC Segmented Client Interface	111
7.4.1. TX MAC Segmented Client Interface with Disabled Preamble Passthrough	113
7.4.2. TX MAC Segmented Client Interface with Enabled Preamble Passthrough	115
7.4.3. Using MAC Segmented <code>skip_crc</code> Signal to Control Source Address, PAD, and CRC Insertion	116
7.4.4. Using MAC Segmented <code>i_tx_mac_error</code> to Mark Packets Invalid	116
7.5. RX MAC Segmented Client Interface	117
7.5.1. RX MAC Segmented Client Interface with Disabled Preamble Passthrough	119
7.5.2. RX MAC Segmented Client Interface with Enabled Passthrough and RX CRC Forwarding	120
7.5.3. RX MAC Segmented Client Interface Status and Errors	121
7.6. MAC Flow Control Interface	121
7.7. PCS Mode TX Interface	122
7.8. PCS Mode RX Interface	125
7.9. FlexE and OTN Mode TX Interface	127
7.10. FlexE and OTN Mode RX Interface	129
7.11. Custom Rate Interface	131
7.12. 32-bit Soft CWBIN Counters	132
7.13. Reconfiguration Interfaces	132
7.13.1. Ethernet Reconfiguration Interfaces	132
7.13.2. Transceiver Reconfiguration Interfaces	134
7.14. Precision Time Protocol Interface	135
7.14.1. Time-of-Day Interface	136
7.14.2. TX 2-Step Timestamp Interface	137
7.14.3. TX 1-Step Timestamp Interface	139

7.14.4. RX Timestamp Interface.....	146
7.14.5. PTP Status Interface.....	147
7.14.6. PTP Tile Interface.....	148
8. Configuration Registers.....	149
8.1. Ethernet Avalon Memory-Mapped Interface Address Space.....	149
8.1.1. Ethernet Hard IP Core CSRs.....	150
8.1.2. FEC and Transceiver Control and Status Registers.....	150
8.2. Transceiver Avalon Memory-Mapped Interface Address Space.....	152
9. Supported Modules and IPs.....	153
9.1. F-Tile Auto-Negotiation and Link Training For Ethernet Intel FPGA IP.....	153
9.1.1. Overview.....	153
9.1.2. Release Information.....	154
9.1.3. Functional Description.....	155
9.1.4. Parameters.....	156
9.1.5. Clocks and Resets.....	161
9.1.6. Registers.....	161
9.2. PTP Tile Adapter.....	162
9.2.1. Overview.....	162
9.2.2. Clocks, Reset, and Interface Ports.....	162
9.2.3. PTP Asymmetry Delay Reconfiguration Interface.....	163
9.2.4. PTP Peer-to-Peer MeanPathDelay Reconfiguration Interface.....	163
10. Supported Tools.....	165
10.1. F-Tile Channel Placement Tool.....	165
10.2. Ethernet Toolkit Overview.....	166
10.2.1. Features.....	166
11. F-Tile Ethernet Intel FPGA Hard IP User Guide Archives.....	168
12. Document Revision History for the F-Tile Ethernet Intel FPGA Hard IP User Guide...	169

1. Overview

The F-Tile Ethernet Intel® FPGA Hard IP is an Ethernet-based IP that includes a configurable, hardened protocol stack for Ethernet. The IP is compatible with the *IEEE 802.3-2018 - IEEE Standard for Ethernet* and the *25G/50G Ethernet Specification* from the 25Gigabit Ethernet Consortium.

Table 1. Features

Features	Description
Ethernet modes with number of supported PMAs for each, where <i>10GE-1</i> is 10GE mode supporting one physical medium attachment (PMA) ⁽¹⁾	<ul style="list-style-type: none"> • 10GE-1 • 25GE-1 • 40GE-4 • 50GE-2, 50GE-1 • 100GE-4, 100GE-2, 100GE-1 • 200GE-8, 200GE-4, 200GE-2 • 400GE-8, 400GE-4
PMA types	<ul style="list-style-type: none"> • FHT: up to four per tile⁽²⁾ • FGT: up to 16 per tile
IP core variations	<ul style="list-style-type: none"> • Media Access Control (MAC) Avalon® ST • MAC segmented • Physical Coding Sublayer (PCS) • Optical Transportation Network (OTN) • Flexible Ethernet (FlexE)
Types of client interface	<ul style="list-style-type: none"> • MAC Avalon streaming interface • MAC segmented interface • Media Independent Interface (MII) • PCS66
Forward error correction (FEC) and Reed-Solomon FEC (RS-FEC)	<ul style="list-style-type: none"> • IEEE 802.3 BASE-R Firecode (CL74) • IEEE 802.3 RS(528,514) (CL91) • IEEE 802.3 RS(544,514) (CL134) • Ethernet Technology Consortium (ETC) RS(272, 258)
Other	<ul style="list-style-type: none"> • Support for the 1588 Precision Time Protocol (PTP) • Support for Auto-negotiation (AN) and Link training (LT) • Support for PCS lane re-ordering

⁽¹⁾ Ethernet modes without specified number of supported lanes assume support for all available variations for that Ethernet mode.

⁽²⁾ Not all FHT PMAs are bonded out in every tile. Refer to the *Intel Agilex® 7 Device Family Pin Connection Guidelines* and *Intel Agilex 7 Device Overview* for exact PMA bond-out information by tile location.

The F-Tile Ethernet Intel FPGA Hard IP core is available in the following configurations. For any variant, choose a MAC Avalon streaming interface, a MAC segmented client interface, a PCS variation, a FlexE variation, or an OTN variation.

Table 2. Variant Selection

FEC Selection supports the following FEC types:

- No FEC: No FEC
- CL74: IEEE 802.3 BASE-R Firecode (CL74)
- CL91: IEEE 802.3 RS(528,514) (CL91)
- CL134: IEEE 802.3 RS(544,514) (CL134)
- ETC: Ethernet Technology Consortium ETC RS(272, 258)

Abbreviations:

- MAC AvST: MAC Avalon ST
- MAC Seg: MAC Segmented

Ethernet Mode	Modulation	PMA Type	FEC Selection					MAC AvST	MAC Seg	PCS (MII)	PCS (OTN/FlexE)	PTP	AN/LT
			No FEC	CL74	CL91	CL134	ETC						
10GE-1	NRZ	FGT	✓	—	—	—	—	✓	✓	✓	✓	✓	✓
25GE-1	NRZ	FGT FHT	✓	✓	✓ ⁽³⁾	✓	—	✓	✓	✓	✓	✓ ⁽⁴⁾ — ⁽⁵⁾	— ⁽⁷⁾ ✓ ⁽⁸⁾
40GE-4	NRZ	FGT	✓	—	—	—	—	✓	✓	—	✓	—	✓
50GE-2	NRZ	FGT FHT	✓	—	✓	✓	—	✓	✓	✓	✓	✓	✓ ⁽⁶⁾ — ⁽⁷⁾
50GE-1	PAM4	FGT FHT	—	—	—	✓	✓	✓	✓	✓	✓	✓	✓
100GE-4	NRZ	FGT FHT	✓	—	✓	✓	—	✓	✓	✓	✓	✓	✓ ⁽⁶⁾ — ⁽⁷⁾
100GE-2	PAM4	FGT FHT	—	—	—	✓	✓	✓	✓	✓	✓	✓	✓
100GE-1	PAM4	FHT	—	—	—	✓	—	✓	✓	✓	✓	✓	✓
200GE-8	NRZ	FGT	—	—	—	✓	—	—	✓	✓	✓	✓	—

continued...

⁽³⁾ This Ethernet mode is also compliant with IEEE 802.3 25GBASE-R RS-FEC (CL108).

⁽⁴⁾ PTP is available for no FEC, IEEE 802.3 BASE-R Firecode (CL74), and IEEE 802.3 RS(528,514) (CL91).

⁽⁵⁾ PTP is not available for IEEE 802.3 RS(544,514) (CL134).

⁽⁶⁾ Auto-negotiation and link training is available for no FEC and IEEE 802.3 RS(528,514) (CL91).

⁽⁷⁾ Auto-negotiation and link training is not available for IEEE 802.3 RS(544,514) (CL134).

⁽⁸⁾ Auto-negotiation and link training is available for no FEC, IEEE 802.3 BASE-R Firecode (CL74), and IEEE 802.3 RS(528,514) (CL91).

Ethernet Mode	Modulation	PMA Type	FEC Selection					MAC AvST	MAC Seg	PCS (MII)	PCS (OTN/FlexE)	PTP	AN/LT
			No FEC	CL74	CL91	CL134	ETC						
200GE-4	PAM4	FGT FHT	—	—	—	✓	✓	—	✓	✓	✓	✓	✓
200GE-2	PAM4	FHT	—	—	—	✓	—	—	✓	✓	✓	✓	✓
400GE-8	PAM4	FGT	—	—	—	✓	✓	—	✓	✓	✓	✓	✓
400GE-4	PAM4	FHT	—	—	—	✓	—	—	✓	✓	✓	✓	✓

Note: For reference about fractured type, refer to the *Fracture Type Used by Mode* table in the *F-Tile Architecture User Guide*.

F-Tile Ethernet Intel FPGA Hard IP supports a variety of protocol implementations.

Table 3. Supported Ethernet Protocols

Ethernet Channel	Protocol	Number of Lanes and Line Rate
10GE	10GBASE-KR	1x10.3125 Gbps NRZ lane for Copper Backplane
	10GBASE-CR	1x10.3125 Gbps NRZ lane for Direct Attach Copper Cable
	10GBASE-LR	1x10.3125 Gbps NRZ lane for optical fiber
25GE	25GBASE-KR	1x25.78125 Gbps NRZ lane for Copper Backplane
	25GBASE-CR	1x25.78125 Gbps NRZ lane for Direct Attach Copper Cable
	25GBASE-R	1x25.78125 Gbps NRZ lane based on the <i>25G Ethernet Consortium</i> specification
	25GAUI-1	1x25.78125 Gbps NRZ lanes for Low Loss Links: Chip-to-Chip, Chip-to-Module
40GE	40GBASE-KR4	4x10.3125 Gbps NRZ lanes for Copper Backplane
	40GBASE-CR4	4x10.3125 Gbps NRZ lanes for Direct Attach Copper Cable
	40GBASE-SR4	4x10.3125 Gbps NRZ lanes for optical fiber
50GE	50GBASE-KR1	1x53.125 Gbps NRZ lane for Copper Backplane
	50GBASE-CR1	1x53.125 Gbps NRZ lane for Direct Attach Copper Cable
	50GBASE-KR2	2x25.78125 Gbps NRZ lane for Copper Backplane
	50GBASE-CR2	2x25.78125 Gbps NRZ lane for Direct Attach Copper Cable
	50GAUI-1	1x53.125 Gbps NRZ lanes for Low Loss Links: Chip-to-Chip, Chip-to-Module
	50GAUI-2	2x25.78125 Gbps NRZ lanes for Low Loss Links: Chip-to-Chip, Chip-to-Module
100GE	100GBASE-KR1	1x106.25 Gbps PAM4 lanes for Copper Backplane
	100GBASE-CR1	1x106.25 Gbps PAM4 lanes for Direct Attach Copper Cable
	100GBASE-KR2	2x53.125 Gbps PAM4 lanes for Copper Backplane
	100GBASE-CR2	2x53.125 Gbps PAM4 lanes for Direct Attach Copper Cable
	100GBASE-KR4	4x25.78125 Gbps Non-Return-to-Zero (NRZ) lanes for Copper Backplane
	100GBASE-CR4	4x25.78125 Gbps NRZ lanes for Direct Attach Copper Cable
continued...		

Ethernet Channel	Protocol	Number of Lanes and Line Rate
	100GAUI-1	1x106.25 Gbps NRZ lanes for Low Loss Links: Chip-to-Chip, Chip-to-Module
	100GAUI-2	2x53.125 Gbps NRZ lanes for Low Loss Links: Chip-to-Chip, Chip-to-Module
	100GAUI-4	4x26.5625 Gbps NRZ lanes for Low Loss Links: Chip-to-Chip, Chip-to-Module
	CAUI-2	2x53.125 Gbps PAM4 lanes for Low Loss Links: Chip-to-Chip, Chip-to-Module
	CAUI-4	4x25.78125 Gbps NRZ lanes for Low Loss Links: Chip-to-Chip, Chip-to-Module
200GE	200GBASE-KR2	2x106.25 Gbps PAM4 lanes for Copper Backplane
	200GBASE-CR2	2x106.25 Gbps PAM4 lanes for Direct Attach Copper Cable
	200GBASE-KR4	4x53.125 Gbps PAM4 lanes for Copper Backplane
	200GBASE-CR4	4x53.125 Gbps PAM4 lanes for Direct Attach Copper Cable
	200GAUI-2	2x106.25 Gbps PAM4 for Low Loss Links: Chip-to-Chip or Chip-to-Module
	200GAUI-4	4x53.125 Gbps PAM4 for Low Loss Links: Chip-to-Chip or Chip-to-Module
	200GAUI-8	8x26.5265 Gbps PAM4 for Low Loss Links: Chip-to-Chip or Chip-to-Module
400GE	400GBASE-KR4	4x106.25 Gbps PAM4 lanes for Copper Backplane
	400GBASE-CR4	4x106.25 Gbps PAM4 lanes for Direct Attach Copper Cable
	400GAUI-4	4x106.25 Gbps PAM4 lanes for Low Loss Links: Chip-to-Chip or Chip-to-Module
	400GBASE-KR8	8x53.125 Gbps PAM4 lanes for Copper Backplane (Ethernet Consortium)
	400GBASE-CR8	8x53.125 Gbps PAM4 lanes for Direct Attach Copper Cable
	400GAUI-8	8x53.125 Gbps PAM4 lanes for Low Loss Links: Chip-to-Chip or Chip-to-Module

Related Information

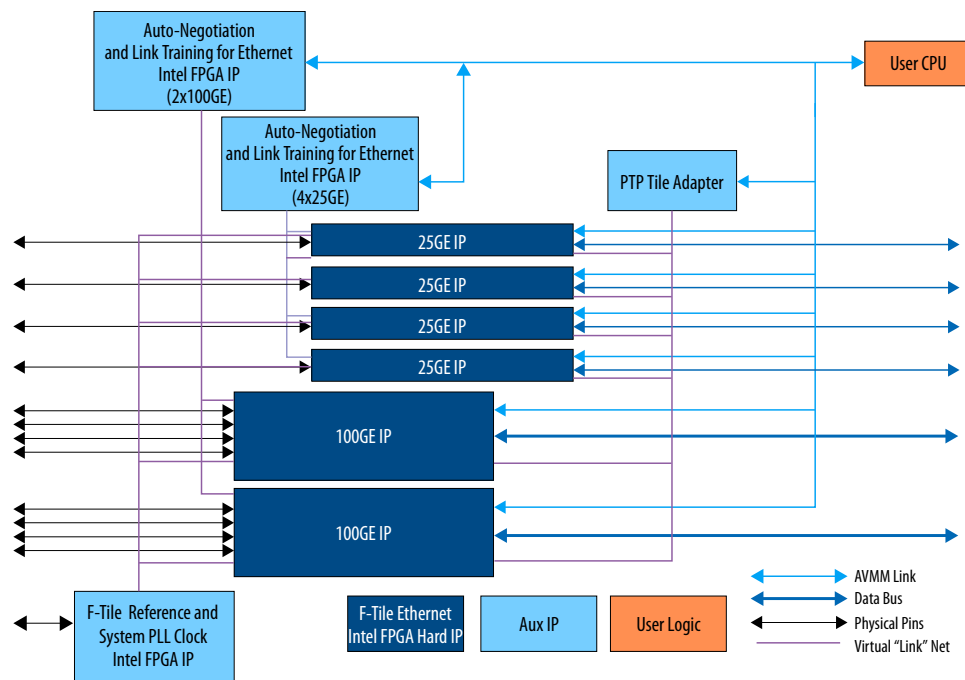
- [F-Tile Architecture User Guide](#)
- [F-Tile Ethernet Intel FPGA Hard IP Design Example User Guide](#)
- [F-Tile Ethernet Intel FPGA Hard IP Release Notes](#)
- [25G Ethernet Consortium](#)

1.1. Ethernet System in F-Tile Overview

The F-Tile Ethernet Intel FPGA Hard IP core along with other supporting IPs allows you to create various Ethernet F-Tile solutions. Note that the F-Tile Ethernet Intel FPGA Hard IP supports one port per IP instance.

The figure below shows one available Ethernet F-Tile configuration. The figure displays instantiation of six F-Tile Ethernet Intel FPGA Hard IP cores with 2 instances configured for 100GE-4 Ethernet rate and 4 instances configured with 25GE-1 Ethernet rate. Two F-Tile Auto-Negotiation and Link Training for Ethernet Intel FPGA IPs are generated to support each specified Ethernet rate. The PTP multiplexer block enables PTP functionality. The F-Tile Reference and System PLL Clock Intel FPGA IP enables you to specify clocking topology. In your design, you assign the serial and reference clock pins to the device physical pins.

Figure 1. Conceptual Overview of Ethernet System in F-Tile



For more information about F-tile architecture and supported hard IP topologies, refer to the *F-Tile Architecture and PMA and FEC Direct PHY IP User Guide*.

Related Information

- [Overview](#)
- [F-Tile Architecture and PMA and FEC Direct PHY IP User Guide](#)

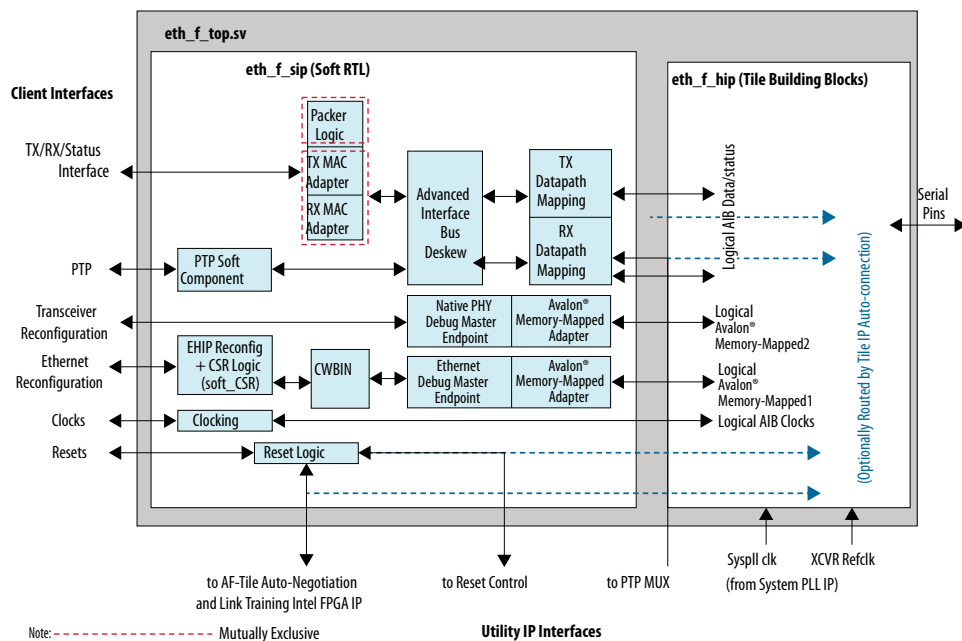
1.2. F-Tile Ethernet Intel FPGA Hard IP Overview

The F-Tile Ethernet Intel FPGA Hard IP core consists of synthesizable soft-logic and the hardened IP core block. Each F-Tile Ethernet Intel FPGA Hard IP core consists of a single Ethernet port, configurable for 10GE, 25GE, 40GE, 50GE, 100GE, 200GE, or 400GE data rate.

The F-Tile Ethernet Intel FPGA Hard IP does not support multiport configuration. You can enable multiport configuration by instantiating multiple IP instances.

The figure below displays the IP core block diagram, showing significant blocks and connections. The same implementation applies to all supported data rate IP options.

Figure 2. F-Tile Ethernet Intel FPGA Hard IP Block Diagram



The TX/RX MAC Adapters provide an optional MAC Avalon ST interface for 10GE/25GE/40GE/50GE/100GE ports, which can also provide an asynchronous interface, and converts from multiple segments to a wide MAC Avalon ST datapath. The MAC Avalon ST client interface is not available for 200GE and 400GE ports.

The F-tile is connected to the FPGA fabric using Intel's embedded multi-die interconnect bridge (EMIB) technology. The EMIB Deskew block corrects for possible skew over the EMIB interfaces between the main FPGA die and the F-Tile. Typically, the 40GE/50GE/100GE/200GE/400GE ports access the EMIB Deskew block. The 10GE/25GE ports that use PTP can also access this block.

The TX/RX Data Path (DP) mapping functions map the Ethernet IP signals to the EMIB datapath.

The PTP soft component logic block enables the PTP interface. The block performs the soft logic operations required for the F-Tile timestamp system for 1588 PTP support and connects to the time-of-day (TOD) module.

The PCS interface and the PCS66 interface follows the path through the EMIB Deskew and DP Mapping stages. The interface does not use adapters.

The Auto-negotiation and Link Training (AN/LT) port connects to a separate F-Tile Auto-Negotiation and Link Training for Ethernet Intel FPGA IP. When enabled, the IP provides the status and control information.

The Reconfiguration and reset logic implement the reconfiguration interfaces and resets for the core, respectively.

Avalon memory-mapped interface (Avalon MM) Adapters communicate with the F-tile raw Avalon memory-mapped interface, allowing an 8-to-32 bit conversion on the transactions.

Optional Debug Master Endpoints instantiate the Avalon MM interfaces via GUI options to enable Transceiver Toolkit and Ethernet Toolkit software utilities. This feature is planned for future release.

32-bit soft CWBIN counters are implemented in Soft IP and this parameter can be enabled for all FEC modes. This soft logic converts the 8-bit CWBin0-3 register (FEC block of F-Tile) of the Hard IP to 32-bit soft logic registers.

1.2.1. Device Family Support

Table 4. Intel FPGA IP Core Device Support Levels

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

Table 5. F-Tile Ethernet Intel FPGA Hard IP Core Device Family Support

Shows the level of support offered by the F-Tile Ethernet Intel FPGA Hard IP for each Intel FPGA device family.

Device Family	Support
Intel Agilex 7	Advance
Other device families	No support

1.2.2. Device Speed Grade Support

The F-Tile Ethernet Intel FPGA Hard IP supports the following speed grades for Intel Agilex 7 devices:

- Core speed grade: -1 or -2 or -3

Note: Intel recommends -1 or -2 core speed grades for IP core variation with PTP, if the system PLL frequency is higher than 805.664062 MHz for NRZ mode or 830.078125 MHz for PAM4 mode

Ethernet IEEE 1588 TOD Synchronizer Intel FPGA IP has lower timing margin for Intel Agilex 7 I-Series OPNs with device density code of 041.

Table 6. Core speed grade/Transceiver speed grade table

Core Speed	XCVR Speed	FHT (BK)		FGT (UX)	
		PAM4	NRZ	PAM4	NRZ
-1V	-1	50GE-1 100GE-1 100GE-2 200GE-2 200GE-4 400GE-4	25GE-1 50GE-2 100GE-4	50GE-1 100GE-2 200GE-4 400GE-8	10GE-1 25GE-1 40GE-4 50GE-2 100GE-4 200GE-8
	-2	50GE-1 100GE-2 200GE-4	25GE-1 50GE-2 100GE-4	50GE-1 100GE-2 200GE-4 400GE-8	10GE-1 25GE-1 40GE-4 50GE-2 100GE-4 200GE-8
-2V	-1	50GE-1 100GE-2 100GE-1 200GE-2 200GE-4 400GE-4	25GE-1 50GE-2 100GE-4	50GE-1 100GE-2 200GE-4 400GE-8	10GE-1 25GE-1 40GE-4 50GE-2 100GE-4 200GE-8
	-2	50GE-1 100GE-2 200GE-4	25GE-1 50GE-2 100GE-4	50GE-1 100GE-2 200GE-4 400GE-8	10GE-1 25GE-1 40GE-4 50GE-2 100GE-4 200GE-8
	-3	-	-	-	10GE-1
	-4	-	-	-	-
-3V/3E	-1	-	-	-	-
	-2	50GE-1 100GE-2 200GE-4	25GE-1 50GE-2 100GE-4	50GE-1 100GE-2 200GE-4 400GE-8	10GE-1 25GE-1 40GE-4 50GE-2 100GE-4 200GE-8
	-3	50GE-1 100GE-2 200GE-4	25GE-1 50GE-1 100GE-4	-	10GE-1
-4F	-1	-	-	-	-
	-2	-	-	-	10GE-1 25GE-1
	-3	-	-	-	10GE-1

For information about the applicable transceiver speed grades based on the target data rates, refer to the following tables in *Intel Agilex 7 Device Data Sheet*:

- Table: F-Tile FHT Transmitter and Receiver Data Rate Performance for Intel Agilex 7 Devices
- Table: F-Tile FGT Transmitter and Receiver Data Rate Performance for Intel Agilex 7 Devices

Related Information

[Intel Agilex 7 F-Tile Transceiver Performance Device Data Sheet](#)

1.2.3. Resource Utilization

The resources for the F-Tile Ethernet Intel FPGA Hard IP are obtained from the Intel Quartus® Prime Pro Edition software version 21.3.

Table 7. Resource Utilization for Intel Agilex 7 Devices

These results were obtained using the Intel Quartus Prime software version 21.3 with the following conditions:

- PTP core variations is enabled with **Timestamp accuracy mode** set to **Advanced**.
- The resource utilization excludes the soft logic utilization for the tile files, generated by Quartus after the Logic Generation phase.
 - The tiles files use approx. 5,000 combinatorial ALUTs, 6,000 logic registers, and 164,000 bits of block memory bits.
 - The PTP tile adapter uses approx. 216 combinatorial ALUTs, 174 logic registers, and 0 block memory bits.

Ethernet Rate	IP Core Variation	Combinatorial ALUTs	Logic Registers	Block Memory Bits
10G	MAC Avalon ST	1370	2353	0
	MAC Avalon ST with PTP	2,446	6,177	3,264
25G	MAC Avalon ST	1370	2353	0
	MAC Avalon ST with PTP	2,452	6,180	3,264
40G	MAC Avalon ST	2914	6858	0
50G	MAC Avalon ST	2144	4260	0
	MAC Avalon ST with PTP	3,320	7,992	1,024
100G	MAC Avalon ST	4515	9204	0
	MAC Avalon ST with PTP	7,964	17,578	1,024
200G	MAC segmented	2164	8650	0
	MAC segmented with PTP	4,531	16,119	1,024
400G	MAC segmented	1998	7241	0
	MAC segmented with PTP	11,033	34,269	2,048

1.2.4. Round-trip Latency

The following table includes the round-trip latency numbers for specific variants. The latency numbers for the F-Tile Ethernet Intel FPGA Hard IP are obtained from the Intel Quartus Prime Pro Edition software version 22.1.

Table 8. Round-trip Latency for Intel Agilex 7 Devices

Client Interface	Data Rate	PMA Type	Ethernet Mode	FEC Mode	Enable Asynchronous Adaptor Clocks	Round Trip (TX+RX) of MAC +PHY	
						Hardware Measurement (PS)	Simulation Number (PS)
MAC Segmented / MAC Avalon ST	10GE	FGT	10GE-1	None	Off	407118	377280
					On	461731	452982
	25GE	FGT	25GE-1	IEEE 802.3 RS(528,514) (CL91)	Off	494002	530974
				IEEE 802.3 BASE-R Firecode (CL 74)	Off	352504	349956
				IEEE 802.3 RS(528,514) (CL91))	On	533721	530974
				IEEE 802.3 BASE-R Firecode (CL 74)	On	414565	402526
				None	Off	181217	217838
				None	On	233348	216991
	40GE	FGT	40GE-4	None	Off	491520	469224
			40GE-4	None	On	491520	474275
	50GE	FGT	50GE-2	None	Off	228383	225948
				None	On	243278	229309
				IEEE 802.3 RS(528,514) (CL91)	Off	372364	359896
				IEEE 802.3 RS(528,514) (CL91)	On	392223	383178
		FHT	50GE-1	IEEE 802.3 RS(544,514) (CL134)	Off	392734	380688
	100GE	FGT	100GE-4	None	Off	285479	270596
				None	Off	297891	283032
		FGT	100GE-4	None	On	305338	291675
				None	On	317750	309593
		FGT	100GE-4	IEEE 802.3 RS(528,514) (CL91)	Off	347539	360849
				IEEE 802.3 RS(528,514) (CL91)	Off	357469	359900
		FHT	100GE-4	IEEE 802.3 RS(528,514) (CL91)	On	382293	367930
				IEEE 802.3 RS(528,514) (CL91)	On	367399	360849
		FGT	100GE-2	IEEE 802.3 RS(544,514) (CL134)	Off	356593	351812
				Ethernet Technology Consortium RS(272, 258)	Off	313224	301156

continued...

Client Interface	Data Rate	PMA Type	Ethernet Mode	FEC Mode	Enable Asynchronous Adaptor Clocks	Round Trip (TX+RX) of MAC +PHY	
						Hardware Measurement (PS)	Simulation Number (PS)
	200GE	FHT	100GE-1	IEEE 802.3 RS(544,514) (CL134)	Off	431942	448708
		FGT	200GE-4	IEEE 802.3 RS(544,514) (CL134)	N/A	296358	291496
		FGT	200GE-4	Ethernet Technology Consortium RS(272, 258)	N/A	260216	252988
		FHT	200GE-2	IEEE 802.3 RS(544,514) (CL134)	N/A	310814	306000
	400GE	FGT	400GE-8	IEEE 802.3 RS(544,514) (CL134)	N/A	272264	260184
		FGT	400GE-8	Ethernet Technology Consortium RS(272, 258)	N/A	248169	243376
		FHT	400GE-4	IEEE 802.3 RS(544,514) (CL134)	N/A	279492	272224
MII PCS Only	25GE	FGT	25GE-1	IEEE 802.3 RS(544,514) (CL134)	N/A	424056	399600
		FHT	25GE-1	IEEE 802.3 RS(544,514) (CL134)	N/A	433694	422088
	50GE	FGT	50GE-2	IEEE 802.3 RS(544,514) (CL134)	N/A	274673	284280
	10GE	FGT	10GE-1	None	N/A	302856	178748
	100GE	FGT	100GE-2	Ethernet Technology Consortium RS(272, 258)	N/A	216847	183148
		FHT	100GE-2	Ethernet Technology Consortium RS(272, 258)	N/A	178296	233740
	200GE	FGT	200GE-4	Ethernet Technology Consortium RS(272, 258)	N/A	171068	187920
	400GE	FGT	400GE-8	Ethernet Technology Consortium RS(272, 258)	N/A	173477	212048
PCS66 OTN	25GE	FGT	25GE-1	Ethernet Technology Consortium RS(272, 258)	N/A	404635	397247
	50GE	FGT	50GE-2	IEEE 802.3 RS(544,514) (CL134)	N/A	296357	274712
	100GE	FGT	100GE-4	IEEE 802.3 RS(528,514) (CL91)	N/A	203558	206016
continued...							

Client Interface	Data Rate	PMA Type	Ethernet Mode	FEC Mode	Enable Asynchronous Adaptor Clocks	Round Trip (TX+RX) of MAC +PHY	
						Hardware Measurement (PS)	Simulation Number (PS)
PCS66 FlexE	100GE	FGT	100GE-4	IEEE 802.3 RS(528,514) (CL91)	N/A	168804	210964
	200GE	FGT	200GE-4	IEEE 802.3 RS(544,514) (CL 134)	N/A	214438	180671
	400GE	FHT	400GE-4	IEEE 802.3 RS(544,514) (CL 134)	N/A	284311	209584

1.2.5. Release Information

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

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

Table 9. F-Tile Ethernet Intel FPGA Hard IP Core Release Information

Item	Description
IP Version	12.0.0
Intel Quartus Prime Version	23.4
Release Date	2023.12.04
Ordering Code	IP-ETH-FTILEHIP
Ordering Code with enabled PTP	IP-ETH-FTILEHIP

Related Information

[F-Tile Ethernet Intel FPGA Hard IP Release Notes](#)

2. Getting Started

The following sections explain how to install, parameterize, simulate, and initialize the F-Tile Ethernet Intel FPGA Hard IP:

2.1. Installing and Licensing F-Tile Ethernet Intel FPGA Hard IP Cores

The Intel Quartus Prime Pro Edition 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. .

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

Figure 3. IP Core Installation Path

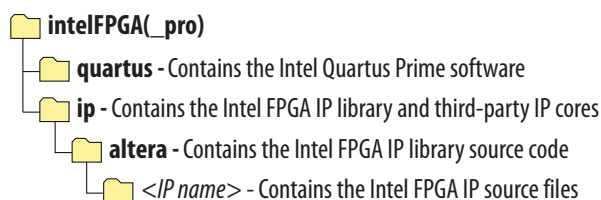


Table 10. IP Core Installation Locations

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

2.2. Specifying the IP Core Parameters and Options

The F-Tile Ethernet Intel FPGA Hard IP parameter editor allows you to quickly configure your custom IP variation. Use the following steps to specify IP core options and parameters in the Intel Quartus Prime Pro Edition software.

1. If you do not already have an Intel Quartus Prime Pro Edition project in which to integrate your F-Tile Ethernet Intel FPGA Hard IP, you must create one.
 - a. In the Intel Quartus Prime Pro Edition, click **File > New Project Wizard** to create a new Quartus Prime project, or **File > Open Project** to open an existing Quartus Prime project. The wizard prompts you to specify a device.
 - b. Specify the device family and select a production F-tile device.

- c. Click **Finish**.
2. In the IP Catalog, locate and select **F-Tile Ethernet Intel FPGA Hard IP**. The **New IP Variation** window appears.
3. Specify a top-level name for your new custom IP variation. The parameter editor saves the IP variation settings in a file named `<your_ip>.ip`.
4. Click **OK**. The parameter editor appears.
5. Specify the parameters for your IP core variation. Refer to [F-Tile Ethernet Intel FPGA Hard IP Parameters](#) on page 23 for information about specific IP core parameters.
6. Optionally, to generate a simulation testbench or hardware design example, follow the instructions in the *F-Tile Ethernet Intel FPGA Hard IP Design Example User Guide*.
7. Click **Generate HDL**. The **Generation** dialog box appears.
8. Select the **IP-XACT** setting to generate the `.ipxact` file containing register map information.
9. Specify output file generation options, and then click **Generate**. The IP variation files generate according to your specifications.
10. Click **Close**.
11. After generating and instantiating your IP variation, make appropriate pin assignments to connect ports and set any appropriate per-instance RTL parameters.

Related Information

[F-Tile Ethernet Intel FPGA Hard IP Design Example User Guide](#)

2.3. Reference and System PLL Clock for your IP Design

Each F-Tile system must instantiate one F-Tile Reference and System PLL Clocks Intel FPGA IP. The F-Tile Reference and System PLL Clocks Intel FPGA IP performs three main functions:

1. Configure reference clock for FHT PMA:
 - Enable the FHT common PLLs and select the reference clock source for FHT common PLL
 - Specify the FHT reference clock source frequency
2. Configure reference clock for FGT PMA:
 - Enable FGT reference clocks and specify the reference clock frequency
 - Specify FGT CDR output
3. Configure system PLL:
 - Enable system PLL and specify its mode
 - Specify the reference clock source and frequency for system PLL

Note: In your IP design, you must include an F-Tile Reference and System PLL Clocks Intel FPGA IP core to pass logic generation flow.

The F-Tile Reference and System PLL Clocks Intel FPGA IP must always connect to a protocol based Intel FPGA IP. The F-Tile Reference and System PLL Clocks Intel FPGA IP cannot be compiled or simulated as a standalone IP. For more information on parameters and port list for F-Tile Reference and System PLL Clocks Intel FPGA IP core, refer to the *F-Tile Architecture and PMA/FEC Direct PHY IP User Guide*.

When you design multiple interfaces or protocol-based IP cores within a single F-Tile, you must use only one instance of the F-Tile Reference and System PLL Clocks Intel FPGA IP core to configure:

- All required reference clocks for FGT PMA (up to 10) and FHT PMA (up to 2) to implement multiple interfaces within a single F-Tile.
- All required FHT common PLLs (up to 2) to implement multiple interfaces within a single F-Tile.
- All required System PLLs (up to 3) to implement multiple interfaces within a single F-Tile.
- All required reference clocks for system PLLs (up to 8 – shared with FGT PMA) to implement multiple interfaces within a single F-Tile.

When you design multiple interfaces or protocol-based IP cores within a single F-Tile, you can only use three System PLLs. For example, you can use one System PLL for PCIe and two for Ethernet and other protocols. However, there are other use cases where you can use all three for various interfaces within the Ethernet and PMA-Direct digital blocks. As there are only three System PLLs, multiple interfaces or protocol-based IP cores with different line rates may have to share a System PLL. While sharing a System PLL, the interface with the highest line rate determines the system PLL frequency, and the interfaces with the lower line rates must be overclocked. For more information, refer to the *F-Tile Architecture and PMA/FEC Direct PHY IP User Guide*.

Related Information

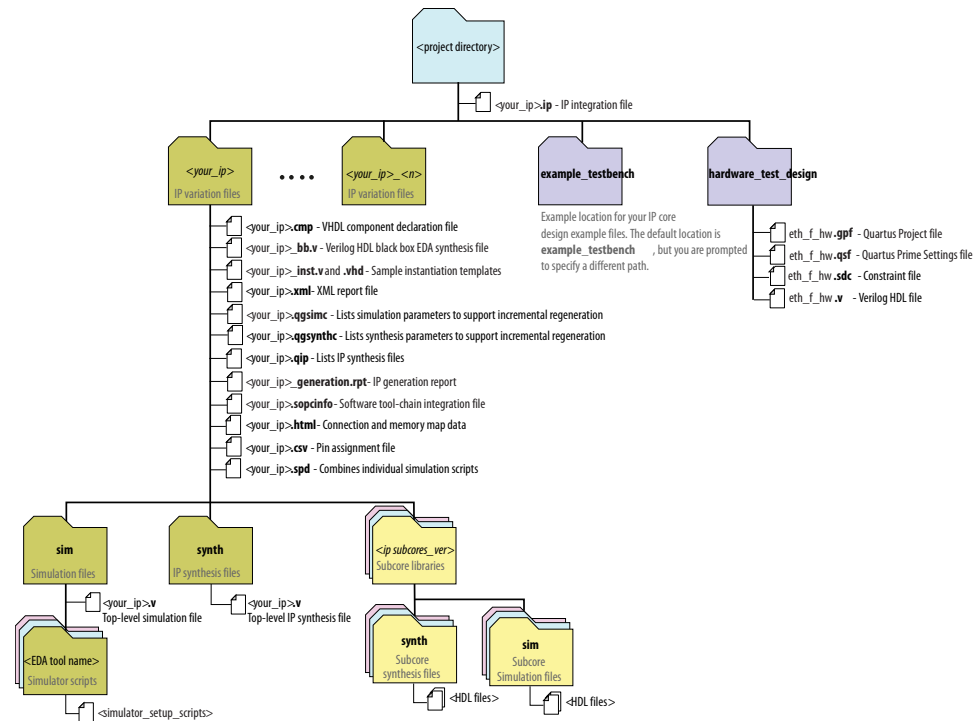
[F-Tile Architecture and PMA/FEC Direct PHY IP User Guide](#)

2.4. Generated File Structure

The Intel Quartus Prime Pro Edition software generates the following IP core output file structure.

For information about the file structure of the design example, refer to the *F-Tile Ethernet Intel FPGA Hard IP Design Example User Guide*.

Figure 4. F-Tile Ethernet Intel FPGA Hard IP Generated Files



Note: The hardware_test_design folder contains files used in the simulation.

Table 11. IP Core Generated Files

File Name	Description
<your_ip>.ip	The Platform Designer system or top-level IP variation file. <your_ip> is the name that you give your IP variation.
<your_ip>.cmp	The VHDL Component Declaration (.cmp) file is a text file that contains local generic and port definitions that you can use in VHDL design files.
<your_ip>.html	A report that contains connection information, a memory map showing the address of each slave with respect to each master to which it is connected, and parameter assignments.
<your_ip>_generation.rpt	IP or Platform Designer generation log file. A summary of the messages during IP generation.
<your_ip>.qgsimc	Lists simulation parameters to support incremental regeneration.
<your_ip>.qgsynthc	Lists synthesis parameters to support incremental regeneration.
<your_ip>.qip	Contains all the required information about the IP component to integrate and compile the IP component in the Intel Quartus Prime software.
<your_ip>.sopcinfo	Describes the connections and IP component parameterizations in your Platform Designer system. You can parse its contents to get requirements when you develop software drivers for IP components. Downstream tools such as the Nios® II tool chain use this file. The .sopcinfo file and the system.h file generated for the Nios II tool chain include address map information for each slave relative to each master that accesses the slave. Different masters may have a different address map to access a particular slave component.

continued...

File Name	Description
<your_ip>.csv	Contains information about the upgrade status of the IP component.
<your_ip>.spd	Required input file for ip-make-simscript to generate simulation scripts for supported simulators. The .spd file contains a list of files generated for simulation, along with information about memories that you can initialize.
<your_ip>_bb.v	You can use the Verilog black-box (_bb.v) file as an empty module declaration for use as a black box.
<your_ip>_inst.v or _inst.vhd	HDL example instantiation template. You can copy and paste the contents of this file into your HDL file to instantiate the IP variation.
<your_ip>.svd	Allows hard processor system (HPS) System Debug tools to view the register maps of peripherals connected to HPS in a Platform Designer system. During synthesis, the .svd files for slave interfaces visible to System Console masters are stored in the .sof file in the debug section. System Console reads this section, which Platform Designer can query for register map information. For system slaves, Platform Designer can access the registers by name.
<your_ip>.v or <your_ip>.vhd	HDL files that instantiate each submodule or child IP core for synthesis or simulation.
<your_ip>.xml	Contains information about interfaces and parameters of the IP component.
mentor/	Contains a ModelSim* script msim_setup.tcl to set up and run a simulation.
synopsys/vcs/ synopsys/vcsmx/	Contains a shell script vcs_setup.sh to set up and run a VCS* simulation. Contains a shell script vcsmx_setup.sh and synopsys_ sim.setup file to set up and run a VCS MX* simulation.
xcelium/	Contains a shell script xcelium_setup.sh and other setup files to set up and run an Xcelium simulation.
submodules/	Contains HDL files for the IP core submodules.
<child IP cores>/	For each generated child IP core directory, Platform Designer generates synth/ and sim/ sub-directories.

2.4.1. Generating IP-XACT File

You can generate the IP-XACT information for the F-Tile Ethernet Intel FPGA Hard IP. This IP-XACT information is included in the <ip_name>.ip file. The generated IP-XACT information includes the register map for your IP. It contains generic information about your IP. The IP variant-specific information such as reset and some register values may vary across the IP variants.

Use the following steps to enable IP-XACT generation in the <ip_name>.ip file:

1. In the IP Parameter Editor window, click **Generate HDL**.
2. In the **Generation** dialog box, select the **IP-XACT** setting.
3. Click **Generate**.
4. Check your <ip_name>.ip file for the IP-XACT information.

Generating IP-XACT Files for Designs with enabled PTP

When you select **Enable IEEE 1588 PTP** setting in the IP Parameter Editor, the PTP-specific registers information is available as follows:

- PTP-related registers are IP-specific. These registers are available in the F-Tile Ethernet Intel FPGA Hard IP's generated .ipxact file.
- PTP asymmetry delay registers and P2P delay registers are tile-specific, not IP-specific registers.

In the **Generation** dialog box, ensure the **Create HDL design files for synthesis** parameter is set to **Verilog** or **VHDL**.

The IP synthesis file directory contains the generated .xml files:

- <variant_name>/eth_f_<version>/synth/eth_ptp_adpt_f_p2p_ipxact.xml
- <variant_name>/eth_f_<version>/synth/eth_ptp_adpt_f_asm_ipxact.xml

2.5. Generating Tile Files

The **Support-Logic Generation** is a pre-synthesis step used to generate tile-related files needed for simulation and hardware design. The tile generation is a required step before simulation.

You can use the **Support-Logic Generation** command on the Processing menu in the Intel Quartus Prime Pro Edition software to generate the F-Tile specific tiles your design. Alternatively, you can run `quartus_tlg` command prompt to generate these files.

Starting with the Intel Quartus Prime software version 21.4, the **Support-Logic Generation** command is run automatically when you generate your design using F-Tile Ethernet Intel FPGA Hard IP Example Design IP Parameter Editor.

A successful tile file generation results in the `eth_f_hw_auto_tiles` files where x represents necessary file extensions. The generated files are located in your project directory and contain the full netlist for simulation and synthesis.

2.6. IP Core Testbenches

Intel provides a testbench design example that you can generate for the F-Tile Ethernet Intel FPGA Hard IP.

To generate the simulation testbench, follow these steps:

1. In the F-Tile Ethernet Intel FPGA Hard IP parameter editor, you must first set the parameter values for the IP core variation you intend to generate in your end product. If you do not set the parameter values for your design to match the parameter values in your end product, the testbench you generate does not exercise the IP core variation you intend.
2. Generate the example design.
3. In the Intel Quartus Prime software, run logic generation to generate the tile-related files. The process generates full netlist for simulation and synthesis.

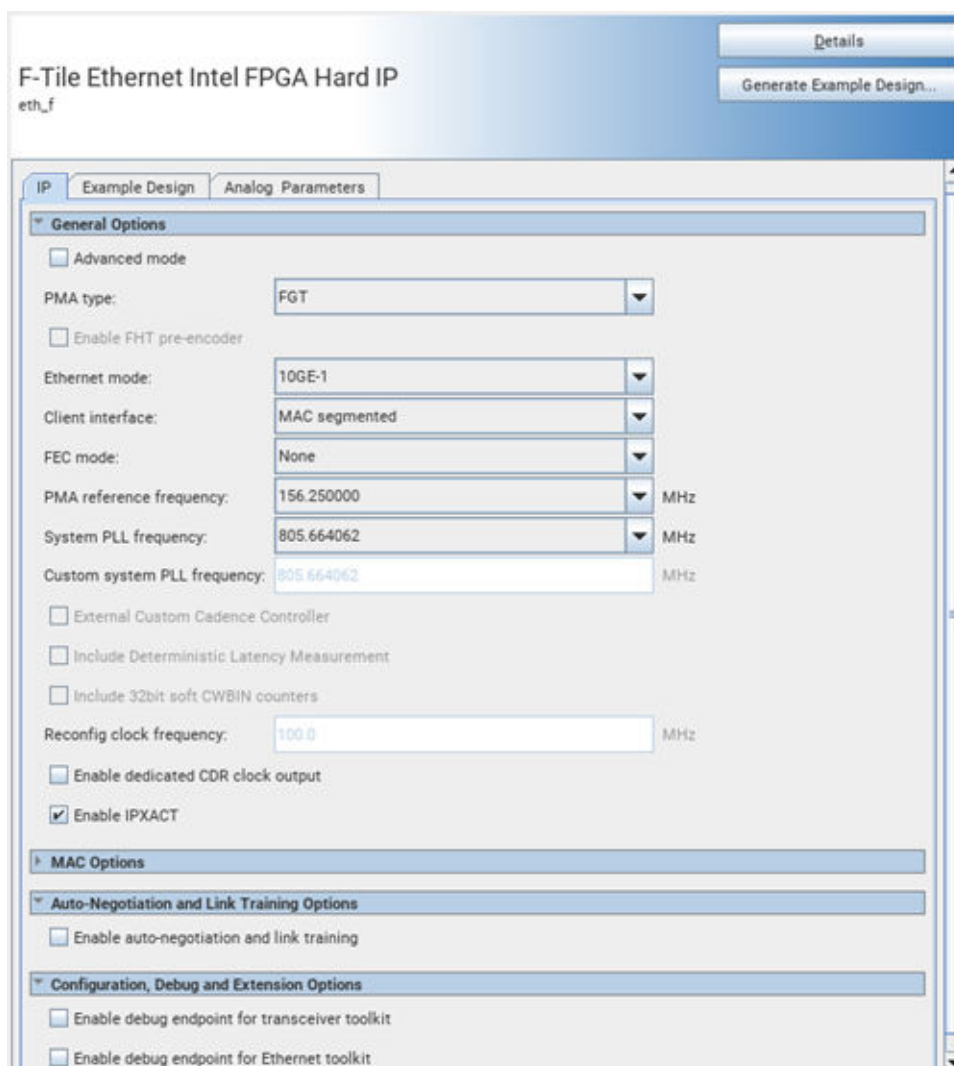
The testbench demonstrates a basic test of the IP core. It is not intended to be a substitute for a full verification environment.

3. F-Tile Ethernet Intel FPGA Hard IP Parameters

The F-Tile Ethernet Intel FPGA Hard IP parameter editor provides the parameters you can set to configure your F-Tile Ethernet Intel FPGA Hard IP variation and simulation and hardware design examples.

The F-Tile Ethernet Intel FPGA Hard IP parameter has an **IP** tab, **Example Design** tab, and **Analog Parameters** tab.

Figure 5. F-Tile Ethernet Intel FPGA Hard IP Parameters: IP Tab



The screenshot shows the 'F-Tile Ethernet Intel FPGA Hard IP' parameter editor window. The 'IP' tab is selected, showing various configuration options. The window has a title bar with 'Details' and 'Generate Example Design...' buttons. The main area is divided into sections: 'General Options', 'MAC Options', 'Auto-Negotiation and Link Training Options', and 'Configuration, Debug and Extension Options'. The 'General Options' section includes checkboxes for 'Advanced mode', 'Enable FHT pre-encoder', 'External Custom Cadence Controller', 'Include Deterministic Latency Measurement', 'Include 32bit soft CWBIN counters', 'Enable dedicated CDR clock output', and 'Enable IPXACT'. It also has dropdown menus for 'PMA type' (FGT), 'Ethernet mode' (10GE-1), 'Client interface' (MAC segmented), 'FEC mode' (None), 'PMA reference frequency' (156.250000 MHz), 'System PLL frequency' (805.664062 MHz), and 'Custom system PLL frequency' (805.664062 MHz). A text input field for 'Reconfig clock frequency' is set to 100.0 MHz. The 'MAC Options' section is collapsed. The 'Auto-Negotiation and Link Training Options' section includes a checkbox for 'Enable auto-negotiation and link training'. The 'Configuration, Debug and Extension Options' section includes checkboxes for 'Enable debug endpoint for transceiver toolkit' and 'Enable debug endpoint for Ethernet toolkit'.

Intel Corporation. All rights reserved. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

*Other names and brands may be claimed as the property of others.

ISO
9001:2015
Registered

For information about the **Example Design** tab and **Analog Parameters** tab, refer to the *F-Tile Ethernet Intel FPGA Hard IP Design Example User Guide* and *Analog Parameters Options in F-Tile Architecture and PMA and FEC Direct PHY Intel FPGA IP User Guide*.

Table 12. F-Tile Ethernet Intel FPGA Hard IP Parameters: IP Tab

This table does not provide information about invalid parameter value combinations. If you make selections that create a conflict, the parameter editor generates error messages in the **System Messages** pane.

Parameter	Range	Default Setting	Parameter Description
General Options			
Advanced mode	<ul style="list-style-type: none"> On Off 	Off	When turned on, enables the Custom Ethernet line rate option and allows you to access the following FEC modes for 25GE-1. <ul style="list-style-type: none"> Fibre Channel RS(528,514) Ethernet Technology Consortium RS(528, 514)
PMA type	<ul style="list-style-type: none"> FHT FGT 	FGT	PMA Channel Type. Selects the F-Tile based targeted PMA type. Each PMA has a different data rate range and compliance specifications. <ul style="list-style-type: none"> FHT: <ul style="list-style-type: none"> 24-29 Gbps NRZ 48-58 Gbps NRZ and PAM4 96-116 Gbps PAM4 FGT: <ul style="list-style-type: none"> 1-32 Gbps NRZ 20-58.125 PAM4⁽⁹⁾
FHT precoding enable	<ul style="list-style-type: none"> Disabled Enabled 	Disabled	Enables the FHT precoding. Available for PAM4 modes, always disabled for NRZ modes. <i>Note:</i> This feature is used during the auto-negotiation and link training. When link training is disabled, the FHT precoding is enabled.
Ethernet mode	<ul style="list-style-type: none"> 10GE-1 25GE-1 40GE-4 50GE-2 50GE-1 100GE-4 100GE-2 100GE-1 200GE-8 200GE-4 200GE-2 400GE-8 400GE-4 	10GE-1	Ethernet Configuration. Specifies the overall port bandwidth across the number of physical lanes used by the port. Term xGE-y represents: <ul style="list-style-type: none"> x is the overall bandwidth of the port y is the number of physical lanes used by port
Custom Ethernet line rate	Varies based on the selected Ethernet mode.	25.78125 Gbps	When turned on, this option allows you to choose the custom Ethernet line rate up to the maximum ethernet rate supported. This parameter is available when Advanced mode is enabled in the IP parameter editor.

continued...

⁽⁹⁾ FGT **Quad0** can only support 20-32 Gbps PAM4. FGT **Quad1**, **Quad2**, and **Quad3** can support 20-58 Gbps PAM4.

Parameter	Range	Default Setting	Parameter Description
Client interface	<ul style="list-style-type: none"> MAC segmented MAC Avalon ST MII PCS only PCS66 OTN PCS66 FlexE 	MAC segmented	<p>Selects data interface exposed to a client. Selected interface determines the Ethernet functional blocks enabled in the design.</p> <ul style="list-style-type: none"> MAC Avalon ST supports up to 100GE Ethernet rates. MAC segmented interface supports all Ethernet rates.
FEC mode	<ul style="list-style-type: none"> None IEEE 802.3 BASE-R Firecode (CL74)⁽¹⁰⁾ IEEE 802.3 RS(528,514) (CL91) IEEE 802.3 RS(544,514) (CL134) Ethernet Technology Consortium RS(272, 258) 	None	<p>Selects the FEC mode for each port.</p> <p>The IP core supports the following FEC types</p> <ul style="list-style-type: none"> IEEE 802.3 BASE-R Firecode is available only for 25GE MAC modes. IEEE 802.3 RS(528,514) is typically used for NRZ modes. IEEE 802.3 RS(544,514) is typically used for PAM4 modes. Ethernet Technology Consortium RS(272,258) is a low-latency substitute for RS(544,514). <p>For more information about FEC modes and the supported protocols, refer to the F-Tile Supported FEC Modes and Compliance Specifications table in the <i>F-Tile Architecture and PMA and FEC Direct PHY IP User Guide</i>.</p>
PMA reference frequency	<ul style="list-style-type: none"> 156.250000 312.500000 322.265625 	156.250000	<p>Selects the reference clock frequency used by the transceiver.</p> <p>156.25 MHz is the recommended frequency for all Ethernet modes. This is the only supported frequency when using FHT or when auto-negotiation and link training (AN/LT) is enabled.</p> <p>312.5 MHz is also supported when using FGT without AN/LT.</p> <p>322.265625 MHz is supported when you select IEEE 802.3 BASE-R Firecode or RS(528,514), while using FGT without AN/LT.</p>
System PLL frequency	<ul style="list-style-type: none"> 830.078125 805.664062⁽¹¹⁾ 322.265625 Custom 	805.664062	<p>Selects the System PLL frequency. The core clock is equivalent to this frequency divided by 2.</p> <p>Recommended frequency based on selected FEC mode:</p>

continued...

⁽¹⁰⁾ This mode is only available for 25G Ethernet mode.

⁽¹¹⁾ The IP GUI specifies the 805.664062 MHz frequency. The actual frequency is 805.6640625 MHz.

Parameter	Range	Default Setting	Parameter Description
			<ul style="list-style-type: none"> Use 830.078125 MHz or higher when you select IEEE 802.3 RS(544,514) (CL134) or Ethernet Technology Consortium RS(272, 258) Use 805.6640625 MHz or higher when you select IEEE 802.3 RS(528,514) (CL91), IEEE 802.3 BASE-R Firecode (CL74), or no FEC. Also, use this frequency or higher when PTP is enabled, including the 10GE PTP option. Use 322.265625 MHz or higher when you select 10GE without PTP. Use Custom when you require other frequencies or if system PLL reference clock source and PMA reference clock source are different. You must define Custom System PLL Frequency parameter value. <ul style="list-style-type: none"> When you select Custom, the IP internally includes a custom cadence controller. When you require an external custom cadence controller, enable External Custom Cadence Controller parameter.
Custom system PLL Frequency	805.6640625 - 903.125 MHz (with enabled PTP) 322.265625 - 1GHz (with disabled PTP)	N/A	If you choose the Custom option in the System PLL Frequency parameter, the IP core clock <code>o_clk_pll</code> is equivalent to half of the specified rate.
External Custom Cadence Controller	<ul style="list-style-type: none"> On Off 	Off	When turned on, enables the external custom cadence controller option and allows you to drive the <code>i_custom_cadence</code> port to DUT. You can use the parameter when sharing custom cadence controller with multiple IP instances. This parameter is available if you choose the Custom option in the System PLL Frequency parameter.
Include Deterministic Latency Measurement	<ul style="list-style-type: none"> On Off 	Off	When turned on, enables the built-in Deterministic Latency Measurement module within the IP. This parameter is available if you choose the PCS66 FlexE option in the Client interface parameter. <i>Note:</i> This parameter and Include Deterministic Latency Interface cannot be enabled at the same time.
Include 32-bit soft CWBIN counters	<ul style="list-style-type: none"> On Off 	Off	This parameter is available when FEC mode is enabled in the IP parameter editor. This soft logic converts the 8-bit CWBin0-3 registers in Hard IP (FEC block of F-Tile) to the 32-bit registers in Soft logic.
Reconfig Clock Frequency	100 to 250 MHz	100 MHz	Avalon® memory-mapped interface reconfiguration clock. The interface uses this clock to access control status registers (CSRs). The clock supports 100 to 250 MHz frequency.
Enable dedicated CDR Clock Output	<ul style="list-style-type: none"> On Off 	Off	When turned on, enables the dedicated CDR clock output. When there is more than one channel number, the CDR clock output is
continued...			

Parameter	Range	Default Setting	Parameter Description
			connected to channel 0. This option is only applicable if the channel 0 is placed within FGT QUAD3 or UX FGT Quad2.
Enable IPXACT	<ul style="list-style-type: none"> On Off 	On	When turned on, IPXACT/CSR register information is included in the generated IP file.
MAC Options			
Basic Tab			
TX maximum frame size	65 – 65535	1518	Maximum packet size (in bytes) the IP core can transmit on the Ethernet link without reporting an oversized packet in the TX statistics counters. In PCS Only, OTN, and FlexE variations, this parameter has no effect and remains at the default value of 1518.
RX maximum frame size	65 – 65535	1518	Maximum packet size (in bytes) the IP core can receive on the Ethernet link without reporting an oversized packet in the RX statistics counters. If you turn on Enforce Maximum Frame Size parameter, the IP core truncates incoming Ethernet packets that exceed this size. In PCS Only, OTN, and FlexE variations, this parameter has no effect and remains at the default value of 1518.
Enforce maximum frame size	<ul style="list-style-type: none"> On Off 	Off	Specifies whether the IP core is able to receive an oversized packet or truncates these packets. In a truncated packet, error signal indicates oversize and FCS error.
Link fault generation option	<ul style="list-style-type: none"> Off Unidirectional Bidirectional 	Off	Specifies the IP core response to link fault events. Bidirectional link fault handling complies with the Ethernet specification, specifically IEEE 802.3 Figure 81-11. Unidirectional link fault handling implements IEEE 802.3 Clause 66: in response to local faults, the IP core transmits Remote Fault ordered sets in interpacket gaps but does not respond to incoming Remote Fault ordered sets. The OFF option is provided for backward compatibility.
Bytes to remove from RX frames	<ul style="list-style-type: none"> None Remove CRC bytes Remove CRC and PAD bytes 	Remove CRC bytes	Selects whether the RX MAC should remove CRC bytes, or remove CRC and PAD bytes, or do not remove anything from incoming RX frames before passing them to the RX MAC Client. If the PAD bytes and CRC are not needed downstream, this option can reduce the need for downstream packet processing logic
Forward RX pause requests	<ul style="list-style-type: none"> On Off 	Off	Selects whether the RX MAC forwards incoming PAUSE and PFC frames on the RX client interface, or drops them after internal processing. <i>Note:</i> If flow control is turned off, the IP core forwards all incoming PAUSE and PFC frames directly to the RX client interface and performs no internal processing. In that case this parameter has no effect.
Use source address insertion	-	-	When enabled, the IP inserts source address in outgoing packets.
continued...			

Parameter	Range	Default Setting	Parameter Description
			<p><i>Note:</i> Use Hexadecimal values to insert source address.</p> <p>Source address insertion applies to PAUSE and PFC packets provided on the TX MAC client interface, but does not apply to PAUSE and PFC packets the IP core transmits in response to the assertion of <code>i_tx_pause</code> or <code>i_tx_pfc[n]</code> on the TX MAC client interface.</p>
TX VLAN detection	<ul style="list-style-type: none"> On Off 	Off	Specifies whether the IP core TX statistics block treats TX VLAN and Stacked VLAN Ethernet frames as regular control frames, or performs Length/Type field decoding, includes these frame in VLAN statistics, and counts the payload bytes instead of the full Ethernet frame in the <code>TxFramOctetsOK</code> counter. If turned on, the IP core identifies these frames in TX statistics as VLAN or Stacked VLAN frames. If turned off, the IP core treats these frames as regular control frames.
RX VLAN detection	<ul style="list-style-type: none"> On Off 	Off	Specifies whether the IP core RX statistics block treats RX VLAN and Stacked VLAN Ethernet frames as regular control frames, or performs Length/Type field decoding, includes these frames in VLAN statistics, and counts the payload bytes instead of the full Ethernet frame in the <code>RxFramOctetsOK</code> counter. If turned on, the IP core identifies these frames in RX statistics as VLAN or Stacked VLAN frames. If turned off, the IP core treats these frames as regular control frames.
Stop TX traffic when link partner sends PAUSE	<ul style="list-style-type: none"> Yes No-PFC only No Disable flow control 	No	When set to Yes , both SFC and PFC are supported. When a pause frame is received, the TX MAC stops sending traffic. When set to No , only SDC is supported. When a pause frame is received, the TX MAC does not stop sending traffic. When set to Disable, flow control is disabled entirely.
Ready latency	0 - 3	0	<p>Selects the ready Latency value on the TX client interface. <code>readyLatency</code> is an Avalon ST interface property that defines the number of clock cycles of delay from when the IP core asserts the <code>o_tx_ready</code> signal to the clock cycle in which the IP core can accept data on the TX client interface. Refer to the <i>Avalon Interface Specifications</i>.</p> <p>In MII PCS Only and MAC segmented variations, this parameter has no effect.</p> <p>Selecting a longer latency (higher number) eases timing closure at the expense of increased latency for the TX datapath in MAC +PCS variations.</p>
Enable TX Packing	<ul style="list-style-type: none"> On Off 	Off	<p>This parameter is available when Client interface is set to MAC segmented mode without PTP enabled for all rates ranging from 40G to 400G and with PTP enabled for all rates ranging from 50GE to 400GE</p> <p>When set to On, packing logic is inserted in TX direction. It removes idle segments in between the packets and maximizes throughput of the MAC.</p>
continued...			

Parameter	Range	Default Setting	Parameter Description
Enable asynchronous adapter clocks	<ul style="list-style-type: none"> On Off 	Off	When turned on, you may drive the <code>i_clk_rx</code> and <code>i_clk_tx</code> clock signals different from <code>o_clk_pll</code> clock. Available only when Client interface is set to MAC Avalon ST .
PTP Tab			
Enable IEEE 1588 PTP	<ul style="list-style-type: none"> On Off 	Off	Enable this option to add IEEE 1588 PTP Timestamp offload functions to the IP core. The IP core can generate TX timestamps and RX timestamps.
Timestamp accuracy mode	<ul style="list-style-type: none"> Basic Advanced 	Advanced	Select PTP TX and RX timestamps accuracy mode. ⁽¹²⁾ In Basic mode, supports the following timestamps accuracy: <ul style="list-style-type: none"> ± 3ns for 10GE, and 25GE Ethernet rate ± 8ns for 50GE, 100GE, 200GE, and 400GE Ethernet rate In Advanced mode, supports the following timestamps accuracy: <ul style="list-style-type: none"> ± 1.5ns for 10GE, 25GE, 50GE, 100GE, 200GE, and 400GE Ethernet rate
Timestamp fingerprint width	8 - 32	8	Specify the width of the timestamp fingerprint in bits on the TX path. The default value is 8 bits.
Specialized Tab			
Enable strict preamble check	<ul style="list-style-type: none"> On Off 	Off	If turned on, the IP core rejects RX packets whose preamble is not the standard Ethernet preamble (0x55_55_55_55_55_55). This option provides an additional layer of protection against spurious Start frames that can occur at startup or when bit errors occur.
Enable strict SFD check	<ul style="list-style-type: none"> On Off 	Off	If turned on, the IP core rejects RX packets whose SFD byte is not the standard Ethernet SFD (0xD5). This option provides an additional layer of protection against spurious Start frames that can occur at startup or when bit errors occur.
Average inter-packet gap	<ul style="list-style-type: none"> 1 8 10 12 	12	Specifies the average minimum inter-packet gap (IPG) the IP core maintains on the TX Ethernet link. Specifies the average minimum inter-packet gap (IPG) the IP core maintains on the TX Ethernet link. The default value of 12 complies with the Ethernet standard. The remaining values support increased throughput. The value of 1 specifies that the IP core transmits Ethernet packets as soon as the data is available, with the minimum possible gap. The IPG depends on the space you leave between frame data as you write it to the core. The IP core no longer complies with the
continued...			

⁽¹²⁾ Ethernet modes without specified lane apply to all lane variants.

Parameter	Range	Default Setting	Parameter Description
			Ethernet standard but the application has control over the average gap and maximizing the throughput.
Enable preamble passthrough	<ul style="list-style-type: none"> On Off 	Off	If turned on, the IP core is in RX and TX preamble pass-through mode. In RX preamble pass-through mode, the IP core passes the preamble and SFD to the client instead of stripping them out of the Ethernet packet. In TX preamble pass-through mode, the client specifies the preamble to be sent in the Ethernet frame.
Additional IPG removed per AM period	0-16536	0	<p>Specifies the number of inter-packet gaps the IP core removes per alignment marker period, in addition to the default number required for protocol compliance.</p> <p>Each increment of 1 in the value of Additional IPG removed per AM period increases throughput by 3ppm in 100GE variations. To specify larger throughput increases, use the Average Inter-packet Gap parameter.</p>
Auto-Negotiation and Link Training Options			
Enable auto-negotiation and link training	<ul style="list-style-type: none"> On Off 	Off	<p>Enables auto-negotiation and link training for the Ethernet port.</p> <p>You must instantiate the F-Tile Auto-Negotiation and Link Training for Ethernet Intel FPGA IP to support this feature.</p>
Configuration, Debug and Extension Options			
Enable Ethernet Debug Endpoint	<ul style="list-style-type: none"> On Off 	Off	<p>Enables the Ethernet debug endpoint.</p> <p>You must enable this parameter to allow the System Console access the Ethernet toolkit.</p>
Enable Native PHY Debug Endpoint	<ul style="list-style-type: none"> On Off 	Off	<p>Enables the Native PHY debug endpoint.</p> <p>You must enable this parameter to allow the System Console access the Transceiver toolkit.</p>

Related Information

- [Analog parameters Option](#)
- [F-Tile Architecture and PMA and FEC Direct PHY IP User Guide](#)
- [F-Tile Ethernet Intel FPGA Hard IP Design Example User Guide](#)



4. Functional Description

4.1. Datapath Description

In the transmit direction, the MAC accepts client frames, and inserts inter-packet gap (IPG), preamble, start of frame delimiter (SFD), padding, and CRC bits before passing them to the PHY. You can configure the MAC to accept some of the additions with the client frame. The MAC also updates the TX statistics counters. The PHY encodes the MAC frame as required for reliable transmission over the media to the remote end.

In the receive direction, the PHY passes frames to the MAC. The MAC accepts frames from the PHY, performs checks, updates statistics counters, strips out the CRC, preamble, and SFD, and passes the rest of the frame to the client. In RX preamble pass-through mode, the MAC passes on the preamble and SFD to the client instead of stripping them out. You can configure the MAC to provide the full RX frame at the client interface, the frame with CRC bytes removed, or the frame with CRC and RX PAD bytes removed.

The F-Tile Ethernet Intel FPGA Hard IP also supports PCS, FlexE, and OTN variations. The PCS variations provide an MII interface to the client and transmit and receive Ethernet packets through a 10-Gbps, 25-Gbps, 50-Gbps, 100-Gbps, 200-Gbps, and 400-Gbps Ethernet PHY implemented in hard IP. The FlexE and OTN variations use PCS66 interface for transmitting and receiving 66b blocks, bypassing the MAC.

The IP core handles the frame encapsulation and flow of data between client logic and an Ethernet network through a 10-Gbps, 25-Gbps, 40-Gbps, 50-Gbps, 100-Gbps, 200-Gbps, and 400-Gbps Ethernet PHY implemented in hard IP, with optional various Forward Error Corrections (FEC).

4.2. F-Tile Ethernet Intel FPGA Hard IP MAC

4.2.1. MAC TX Datapath

The below sections are applicable for both, TX MAC Avalon ST interface and TX MAC segmented interface.

When the TX MAC module in a channel is enabled, it receives the client payload data with the destination and source addresses and then adds, appends, or updates various header fields in accordance with the configuration specified. The MAC does not modify the destination address or the payload received from the client. However, the TX MAC module adds a preamble (if the IP core is not configured to receive the preamble from user logic), pads the payload of frames greater than eight bytes to satisfy the minimum Ethernet frame payload of 46 bytes, and if you enable source address insertion, replaces the bytes in the source address field position of your data with a stored source address you provide as a parameter.

Note: The TX MAC interface does not support non-contiguous transfer. The `i_tx_valid` or `i_tx_mac_valid` must be continuously asserted between the assertions of the start of packet and end of packet signals for the same packet. You must implement store and forward packet mechanism when transferring non-contiguous packets.

When TX MAC interface ready signal indicates low, if you use MAC Avalon ST interface, the valid signal may go low. If you use MAC segmented interface, the valid signal must go low.

The client interface includes a port named `i_tx_skip_crc` or `i_tx_mac_skip_crc`, which when asserted during a frame, makes the MAC skip the insertion of source address, padding, and CRC.

- When CRC insertion is skipped, the client must provide a CRC for the frame data it writes in the last 4 bytes of the frame.
- When padding is skipped, the frame data must be large enough to include a fully formed frame header (at least 14 bytes long) or the MAC will automatically mark it as an error frame.

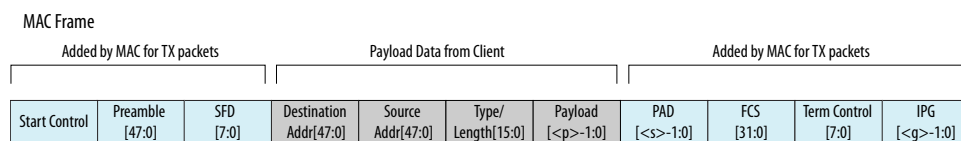
The TX MAC module always inserts IDLE bytes to maintain an average IPG.

The F-Tile Ethernet Intel FPGA Hard IP drops incoming frames of less than nine bytes.

Figure 6. Typical Client Frame at the Transmit Interface

The figure illustrates the changes that the TX MAC makes to the client frame when **Enable Preamble Passthrough** is turned off. This figure uses the following notational conventions:

- `<p>` = payload size, which is arbitrarily large.
- `<s>` = number of padding bits (0–46 bytes)
- `<g>` = number of IPG bits (full bytes)



The following sections describe the functions performed by the TX MAC:

[TX Preamble, Start, and SFD Insertion](#) on page 32

[Source Address Insertion](#) on page 33

[Length/Type Field Processing](#) on page 33

[Frame Padding](#) on page 33

[Frame Check Sequence \(CRC-32\) Insertion](#) on page 33

[Inter-Packet Gap Generation and Insertion](#) on page 34

[TX Packing Logic](#) on page 34

4.2.1.1. TX Preamble, Start, and SFD Insertion

In the TX datapath the MAC appends an eight-byte preamble that begins with a Start byte (0xFB) to the client frame. If you turn on **Link fault generation option**, this MAC module also incorporates the functions of the reconciliation sublayer.

The source of the preamble depends on whether you enable the preamble pass-through feature by turning on **Enable preamble passthrough** in the F-Tile Ethernet Intel FPGA Hard IP parameter editor.

If the preamble pass-through feature is turned on, the client must provide 8 preamble bytes (including an SFD byte) on the data bus. The MAC will automatically replace the Start Control byte.

4.2.1.2. Source Address Insertion

If you configure the IP core to use source address insertion, the MAC replaces the bytes in the `Source Addr` field provided by the client interface with the source address given by the `txmac_saddr` register.

To enable source address insertion, turn on **Use Source Address Insertion** in the F-Tile Ethernet Intel FPGA Hard IP parameter editor.

4.2.1.3. Length/Type Field Processing

This two-byte header field represents either the length of the payload or the type of MAC frame. When the value of this field is equal to or greater than 1536 (0x600) it indicates a type field. Otherwise, this field provides the length of the payload data that ranges from 0–1500 bytes. The TX MAC does not modify this field before forwarding it to the network; it uses this field to generate TX Statistics.

4.2.1.4. Frame Padding

When the length of client frame is less than 64 bytes and greater than eight bytes, the TX MAC module inserts pad bytes after the payload to create a frame length equal to the minimum size of 64 bytes. If the `i_tx_skip_crc` or `i_tx_mac_skip_crc` signal is asserted while writing frame data, the core does not insert PAD bytes even if the frame is shorter than 64 bytes long.

Caution: When using the MAC segmented interface, if the **Enable preamble passthrough** parameter and the `i_tx_mac_skip_crc` signal are both disabled, the payload length must not be less than 9 bytes. However, if the **Enable preamble passthrough** parameter and the `i_tx_mac_skip_crc` signal are both enabled, the entire frame, including Preamble and CRC, must not be less than 9 bytes. You must ensure that such frames do not reach the TX client interface as the F-Tile Ethernet Intel FPGA Hard IP discards them.

4.2.1.5. Frame Check Sequence (CRC-32) Insertion

If the `i_tx_skip_crc` or `i_tx_mac_skip_crc` signal on the TX client interface is not asserted, the TX MAC computes and inserts a frame check sequence (FCS) in the transmitted MAC frame. The FCS field contains a 32-bit Cyclic Redundancy Check (CRC32) value. The MAC computes the CRC32 over the frame bytes that include the source address, destination address, length/type field, data, and pad (if applicable). The FCS computation excludes the preamble and SFD. The encoding is defined by the following generating polynomial:

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

CRC bits are transmitted with MSB (X32) first.

If `i_tx_skip_crc` or `i_tx_mac_skip_crc` is asserted while writing frame data, the TX MAC does not append an FCS to the end of the frame. This causes the resulting packet to be invalid unless the last 4 bytes of frame data are a correctly computed FCS value.

4.2.1.6. Inter-Packet Gap Generation and Insertion

If you set **Average Inter-packet Gap** to **12** in the F-Tile Ethernet Intel FPGA Hard IP parameter editor, the TX MAC maintains the minimum inter-packet gap (IPG) between transmitted frames required by the *IEEE 802.3 Ethernet standard*. The standard requires an average minimum IPG of 96 bit times (or 12-byte times). The MAC uses a deficit idle counter to allow the actual gap between frames to vary as needed to meet the maximum throughput requirements of the link.

If you set **Average Inter-packet Gap** to **10** or **8**, the TX MAC maintains a minimum average IPG of 10 or 8 bytes accordingly. This option is provided as an intermediate option to allow you to enforce an IPG that does not conform to the Ethernet standard, but which increases the throughput of your IP core.

If you set **Average Inter-packet Gap** to **1**, the IP core transmits Ethernet packets as soon as the data is available, with the minimum possible gap. The IPG depends on the space you leave between frame data as you write it to the core. If you select this parameter value, the core no longer complies with the Ethernet standard, but your application has control over the average gap and throughput can be maximized. For a packet of size (P) bytes, the number of idles bytes (G) inserted after is specified by the following formula $G = 8 - (P \% 8)$. A few examples are depicted below:

Packet Size (P)	Gap Idle Bytes (G)
64	8
65	7
66	6
67	5
68	4
69	3
70	2
71	1
72	8

Note: Even when you set the Average Inter-packet Gap to 1, the 10GE/25GE channels still enforces an effective IPG of 5. This is because the protocol specifically prohibits IPG lower than 5 for 10GE/25GE links to prevent MACs from producing packets that cannot be encoded using 64B/66B encoders.

4.2.1.7. TX Packing Logic

Attention: To achieve the maximum throughput when using the TX MAC segmented interface, the input packets must be packed tightly, leaving no idle segments in between.

TX Packing Logic Features

The F-Tile Ethernet Intel FPGA Hard IP TX packing for transmit direction can be implemented in MAC segmented mode. This feature is implemented in soft logic.

Figure 7. Enable TX Packing IP Parameter Editor

The screenshot shows the 'MAC Options' parameter editor with the 'Basic' tab selected. The 'Enable TX Packing' checkbox is highlighted with a red box. Other visible options include 'TX maximum framesize' (1518), 'RX maximum framesize' (1518), 'Enforce maximum frame size' (unchecked), 'Link fault generation option' (Off), 'Bytes to remove from RX frames' (Remove CRC bytes), 'Forward RX pause requests' (unchecked), 'Use source address insertion' (unchecked), 'TX VLAN detection' (checked), 'RX VLAN detection' (checked), 'Stop TX traffic when link partner sends PAUSE' (No), 'Ready latency' (0), and 'Enable asynchronous adapter clocks' (unchecked).

- TX packing in MAC segmented mode is implemented for all rates ranging from 50G to 400G (MAC segmented interface width of 128b and above) with PTP enabled, and 40G to 400G without PTP enabled. This TX packing logic increases IP resource utilization, as shown in the table below:

Table 13. IP resource utilization

With PTP Enabled	Without PTP Enabled
216 ALMs for 50GE	205 ALMs for 40GE/50GE
809 ALMs for 100GE	791 ALMs for 100GE
2606 ALMs for 200GE	2583 ALMs for 200GE
7457 ALMs for 400GE	7281 ALMs for 400GE

- The TX packing logic adds an additional ~30 cycles of `i_clk_tx` latency.
- The TX Packing feature is disabled by default to ensure backward compatibility with the previous IP version. If you do not want to run the IP at full traffic throughput, use the disable option for lower IP utilization and latency. The maximum TX MAC throughput is reduced when sending unpacked data with idle segments in between packets while TX packing is disabled.
- When unpacked data is generated on the TX MAC segmented interface, the TX packing enabled IP to meet close to 100% line rate of all traffic with no drops or corruptions.

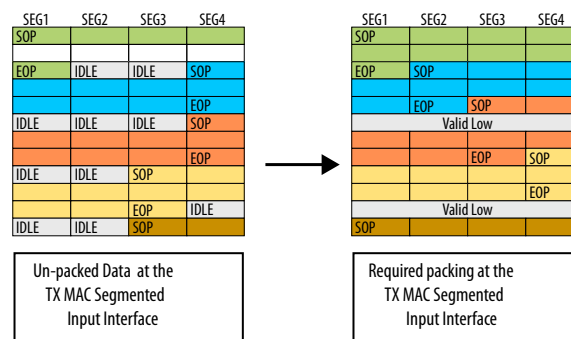
TX MAC Segmented Client Interface with Disabled TX Packing

To achieve close to 100% line rate without using the TX packing logic, you must implement your own packing logic on the MAC segmented interface as shown below.

The figure below depicts an example of 100GE packet data on the MAC segmented interface.

- The box on the left shows an example of loosely packed data with 2-3 idle segments in between two packets.
- The box on the right shows that data is packed with no idle segments between two packets. Pack the IPs input as shown in the right figure.

Figure 8. TX MAC Segmented Client Interface with Disabled TX Packing



4.2.2. MAC RX Datapath

The below sections are applicable for both, RX MAC Avalon ST interface and RX MAC segmented interface.

When the RX MAC in the channel is enabled, it receives Ethernet frames from the PHY and forwards them to the client with framing information together with the results of header and error checking functions.

You can configure whether to include or remove the PAD bytes and FCS using the **Bytes to remove from RX frames** parameter.

Figure 9. Flow of Frame Through the MAC Avalon ST RX Without Preamble Pass-Through

The figure illustrates the typical flow of frame through the MAC RX when the preamble pass-through feature is turned off. In this figure, <p> is payload size, and <s> is the number of pad bytes (0-46 bytes).

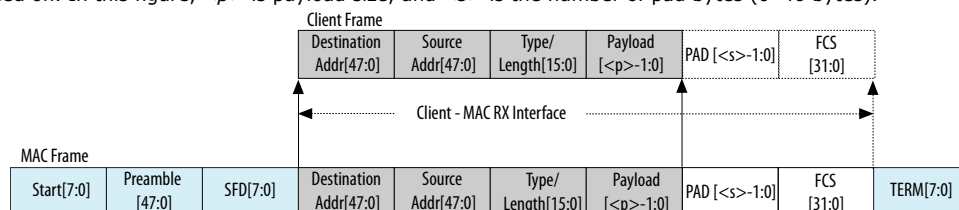
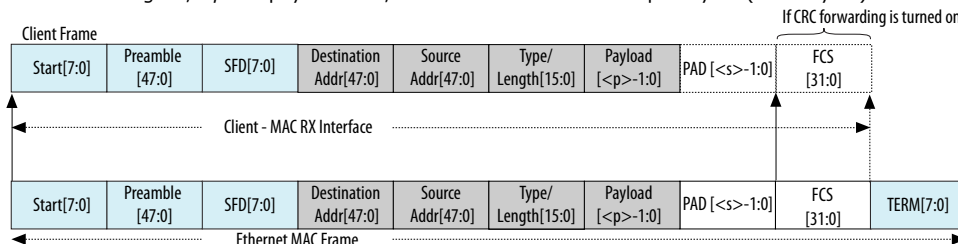


Figure 10. Flow of Frame Through the MAC Avalon ST RX With Preamble Pass-Through

The figure illustrates the typical flow of frame through the MAC RX when the preamble pass-through feature is turned on. In this figure, <p> is payload size, and <s> is the number of pad bytes (0–46 bytes)..



The following sections describe the functions performed by the RX MAC:

[RX Preamble Processing](#) on page 37

[RX Strict SFD Checking](#) on page 37

[RX FCS Checking](#) on page 38

[RX Malformed Packet Handling](#) on page 38

[Removing PAD Bytes and FCS Bytes from RX Frames](#) on page 38

[RX Undersized Frames, Oversized Frames, and Frames with Length Errors](#) on page 38

[Inter-Packet Gap](#) on page 38

4.2.2.1. RX Preamble Processing

The preamble sequence is Start, six preamble bytes, and SFD. The Start byte must be on receive lane 0 of the MII, which means byte [7:0] of the data decoded from a 66b block. The IP core uses the Start Control byte (0xFB, with the corresponding MII control bit set to 1) to identify the start of the Ethernet packet, and the location of the preamble.

By default, the MAC RX removes all Start, SFD, preamble, and IPG bytes from accepted frames. However, if you turn on **Enable preamble passthrough** in the F-Tile Ethernet Intel FPGA Hard IP parameter editor, the MAC RX does not remove the eight-byte preamble sequence.

4.2.2.2. RX Strict SFD Checking

The F-Tile Ethernet Intel FPGA Hard IP RX MAC checks all incoming packets for a correct Start byte (0xFB).

If you turn on **Enable strict preamble check** in the F-Tile Ethernet Intel FPGA Hard IP parameter editor, the RX MAC requires all RX packets to have an Ethernet standard preamble (0x55_55_55_55_55_55). If you turn on **Enable strict SFD check**, the RX MAC requires all RX packets to have an Ethernet standard Start Frame Delimiter (0xD5).

Strict checking reduces the incidence of runt packets caused by bit errors on the line. However, do not use strict checking in applications where custom preamble values or SFD values are needed.

4.2.2.3. RX FCS Checking

The RX MAC checks the FCS of all incoming packets that are minimum sized or larger. If the RX MAC Avalon ST detects an FCS error, it marks the frame invalid by asserting `o_rx_error[1]`. If the RX MAC segmented detects an FCS error, it marks the frame invalid by asserting `o_rx_mac_rcs_error`. FCS errors are also indicated for arriving packets containing an Error control block.

4.2.2.4. RX Malformed Packet Handling

While receiving an incoming packet from the Ethernet link, the RX MAC expects packets to end with a Terminate Control byte. Packets that contain a control byte other than Terminate are malformed packets. The RX MAC asserts `o_rx_error[0]` or sets `o_rx_mac_error[1:0] = 2'b1` on the EOP segment when the frame ends to indicate that it was a malformed packet.

4.2.2.5. Removing PAD Bytes and FCS Bytes from RX Frames

The **Bytes to remove from RX frames** parameter in the parameter editor offers the option of removing bytes from the end of RX frames. You can program the RX MAC to present all the bytes that arrive at the end of an RX frame, remove the RX FCS bytes only, or remove the RX FCS bytes and any RX PAD bytes that were added to the frame.

4.2.2.6. RX Undersized Frames, Oversized Frames, and Frames with Length Errors

The RX MAC flags RX frames that arrive with fewer than 64 bytes as undersized, and are not checked for FCS. The RX MAC marks undersized frames by asserting `o_rx_error[2]` or sets `o_rx_mac_error[2:0] = 2'b2` on the EOP segment when the frame ends.

The RX MAC marks RX frames that arrive with more bytes than the **RX Maximum Frame Size** value you specify in the parameter editor as oversized. The RX MAC marks oversized frames by asserting `o_rx_error[2]` or sets `o_rx_mac_error[1:0] = 2'b2` when the frame ends.

If you turn on **Enforce maximum frame size** in the parameter editor, oversized frames are not allowed through the RX client interface. When the frame reaches the maximum size, it is ended, and the RX MAC asserts both `o_rx_error[2]` and `o_rx_error[1]` to indicate the frame was truncated.

RX Frames that arrive with a valid Length field ($\text{Length/Type} \leq 1500$) are checked for length errors. If the length of the packet advertised in the Length/Type field is larger than the length of the frame that actually arrived, the RX MAC asserts `o_rx_error[4]` or sets `o_rx_mac_error[1:0] = 2'b3` on the EOP segment to indicate that there was a length error.

4.2.2.7. Inter-Packet Gap

The MAC RX removes all IPG octets received, and does not forward them to the client interface. For 10GE/25GE rates, it can sustain a stream with IPG of 5. For rates higher than 25GE, it can tolerate a sustained stream of packets with an IPG of 1.

4.2.3. Congestion and Flow Control Using PAUSE or Priority Flow Control (PFC)

If you do not select **Disable Flow Control** in the Stop TX traffic when link partner sends pause parameter, the F-Tile Ethernet Intel FPGA Hard IP provides flow control to reduce congestion at the local or remote link partner. When either link partner experiences congestion, the respective TX MAC can be instructed to send PAUSE or PFC frames to regulate the flow of data from the other side of the link.

- PAUSE frames instruct the remote transmitter to stop sending data for the duration that the congested receiver specified in an incoming XOFF frame.
- PFC frames instruct the receiver to halt the flow of packets assigned to a specific Priority Queue for a specified duration.

4.2.3.1. Conditions Triggering XOFF Frame Transmission

The F-Tile Ethernet Intel FPGA Hard IP supports retransmission. In retransmission, the IP core retransmits a XOFF frame periodically, extending the pause time, based on signal values.

The TX MAC transmits PAUSE XOFF frames when one of the following conditions occurs:

- Client requests XOFF transmission—A client can explicitly request that XOFF frames be sent using the `i_tx_pause` and `i_tx_pfc[7:0]` signals. When `i_tx_pause` is asserted, a PAUSE XOFF frame is sent to the Ethernet network when the current frame transmission completes. When `i_tx_pfc` is asserted, a PFC XOFF packet is transmitted with XOFF requests for each of the Queues that has a bit high in the signal. For example, setting `i_tx_pfc` to 0x03 sends XOFF requests for Queues 0 and 1.
- Host (software) requests PAUSE XOFF transmission—Setting the pause request register triggers a request that a PAUSE XOFF frame be sent. Similarly, setting the PFC request register triggers PFC XOFF frame requests for the selected Priority Queues.
- Retransmission mode—If the retransmit hold-off enable bit has the value of 1, and the `i_tx_pause` signal remains asserted or the pause request register value remains high, when the time duration specified in the hold-off quanta register has lapsed after the previous PAUSE XOFF transmission, the TX MAC sends another PAUSE XOFF frame to the Ethernet network. The same mechanism applies to PFC. While the IP core is paused in retransmission mode, you cannot use either of the other two methods to trigger a new XOFF frame: the signal or register value is already high.

Note: Intel recommends that you use the flow control ports to backpressure the remote Ethernet node.

4.2.3.2. Conditions Triggering XON Frame Transmission

The TX MAC transmits PAUSE or PFC XON frames when one of the following conditions occurs:

- Client requests XON transmission—A client can explicitly request that XON frames be sent using the pause control interface signal. When `i_tx_pause` is deasserted, a PAUSE XON frame is sent to the Ethernet network when the current frame transmission completes. Similarly, when `i_tx_pfc[n]` is deasserted, a PFC frame is sent with a PFC XON message for queue *n*. If multiple PFC queues are deasserted, the TX MAC will pack the requests into the same PFC packet if possible.
- Host (software) requests XON transmission—Resetting the pause request register triggers a request that an XON frame be sent.

4.2.3.3. Pause Control and Generation Interface

The flow control interface implements PAUSE as specified by the *IEEE 802.3ba 2010 High Speed Ethernet Standard*, PFC as specified by the *IEEE Standard 802.1Qbb*.

You can configure the PAUSE logic to automatically stop local packet transmission when the link partner sends a PAUSE XOFF packet. The PAUSE logic can pass the PAUSE packets through as normal packets or drop the packets before they reach the RX client.

As for PFC frames, you can configure the PFC logic to pass the PFC packets through as normal packets or drop them before they reach the RX client. However, you don't have an option to stop traffic automatically when a PFC XOFF frame arrives.

Table 14. Pause Control and Generation Signals

Describes the signals that implement pause control. These signals are available only if you turn on flow control in the F-Tile Ethernet Intel FPGA Hard IP parameter editor.

Signal Name	Direction	Description
<code>i_tx_pause</code> (PAUSE) <code>i_tx_pfc</code> (PFC)	Input	Level signal which directs the IP core to insert a PAUSE or PFC frame for priority traffic class [n] on the Ethernet link. If bit [n] of the <code>TX_PAUSE_EN</code> register has the value of 1, the IP core transmits an XOFF frame when this signal is first asserted. If you enable retransmission, the IP core continues to transmit XOFF frames periodically until the signal is de-asserted. When the signal is deasserted, the IP core inserts an XON frame.
<code>o_rx_pause</code> (PAUSE) <code>o_rx_pfc</code> (PFC)	Output	Asserted to indicate an RX PAUSE or PFC signal match. The IP core asserts bit [n] of this signal when it receives a pause request with an address match, to signal the TX MAC to throttle its transmissions from priority queue [n] on the Ethernet link.

4.2.3.4. Pause Control Frame Filtering

The F-Tile Ethernet Intel FPGA Hard IP supports options to enable or disable the following features for incoming pause control frames. These options are available if you do not set the **Stop TX traffic when link partner sends pause** parameter to **Disable Flow Control**.

For filtering, the PAUSE and PFC packets are only processed if their destination address matches the address given in the `rx_pause_daddr` register.

- If you turn on **Forward RX Pause Requests** in the parameter editor, the RX PAUSE and PFC frames are always passed along the RX client, even if they are processed.
- If you turn off **Forward RX Pause Requests** in the parameter editor, the RX PAUSE and PFC packets are processed internally, and not presented to the RX client as valid packets.

A PAUSE or PFC packet must have a destination address that matches the `rx_pause_daddr` register, a Length/Type field that is set to 0x8808, and the first 2 bytes of the packet set to 0x0001 or 0x0101.

To actually trigger PAUSE or PFC, you must also ensure that the packets are of the correct length and have no FCS error. Because these conditions are not known until the whole packet has arrived, if you turn off **Forward RX Pause Requests**, you may have packets that are filtered because they look like PAUSE or PFC packets, but not processed because they are of the wrong size or have an error.

4.2.4. Link Fault Signaling

If you enable **Link Fault Generation Mode** in the F-Tile Ethernet Intel FPGA Hard IP parameter editor, the IP core provides link fault signaling as defined in the *IEEE 802.3-2018 IEEE Standard for Ethernet*.

The Ethernet MAC includes a Reconciliation Sublayer (RS) located between the MAC and the MII to manage local and remote faults. Link fault signaling on the Ethernet link is disabled by default but can be enabled by bit [0] of the `link_fault_config` register. When the `link_fault_config` register bits [1:0] have the value of 2'b01, link fault signaling is enabled in normal bidirectional mode. In this mode, the local RS TX logic transmits remote fault sequences in case of a local fault and transmits IDLE control words in case of a remote fault.

If you turn on bit [1] of the `link_fault_config` register, the IP core conforms to Clause 66 of the *IEEE 802.3-2018 IEEE Standard for Ethernet*. When `link_fault_config[1:0]` has the value of 2'b11, the IP core transmits the fault sequence ordered sets in the interpacket gaps according to the clause requirements.

The RS RX logic sets `remote_fault_status` or `local_fault_status` to 1 when the RS RX block receives remote fault or local fault sequence ordered sets. When valid data is received in more than 127 columns, the RS RX logic resets the relevant fault status (`remote_fault_status` or `local_fault_status`) to 0.

The IEEE standard specifies RS monitoring of `RXC<7:0>` and `RXD<63:0>` for Sequence ordered_sets. For more information, refer to *Figure 81-9—Link Fault Signaling state diagram* and *Table 81-5—Sequence ordered_sets* in the *IEEE 802.3 2018 IEEE Standard for Ethernet*. The variable `link_fault` is set to indicate the value of an RX Sequence ordered_set when four fault_sequences containing the same fault value are received with fault sequences separated by less than 128 columns and with no intervening fault_sequences of different fault values. The variable `link_fault` is set to OK following any interval of 128 columns not containing a remote fault or local fault Sequence ordered_set.

4.2.5. Order of Ethernet Transmission

The TX MAC transmits bytes on the Ethernet link starting with the preamble and ending with the FCS in accordance with the IEEE 802.3 standard. On the transmit client interface, the IP core expects the client to send the most significant bytes of the frame first, and to send each byte in big-endian format. Similarly, on the receive client interface, the IP core sends the client the most significant bytes of the frame first, and orders each byte in big-endian format.

The figure below describes the byte order on the Avalon streaming interface when the preamble pass-through feature is turned off. Destination Address[40] is the broadcast/multicast bit (a type bit), and Destination Address[41] is a locally administered address bit.

Figure 11. Byte Order on the Client Interface Lanes Without Preamble Pass-Through

	Destination Address (DA)						Source Address (SA)						Type/ Length (TL)		Data (D)		
Octet	5	4	3	2	1	0	5	4	3	2	1	0	1	0	00	...	NN
Bit	[47:40]	[39:32]	[31:24]	[23:16]	[15:8]	[7:0]	[47:40]	[39:32]	[31:24]	[23:16]	[15:8]	[7:0]	[15:8]	[7:0]	MSB[7 :0]	...	LSB[7:0]

For example, the destination MAC address includes the following six octets AC-DE-48-00-00-80. The first octet transmitted (octet 0 of the MAC address described in the 802.3 standard) is AC and the last octet transmitted (octet 7 of the MAC address) is 80. The first bit transmitted is the low-order bit of AC, a zero. The last bit transmitted is the high order bit of 80, a one.

The preceding table and the following figure show that in this example, 0xAC is driven on DA5 (DA[47 : 40]) and 0x80 is driven on DA0 (DA[7 : 0]).

Figure 12. Octet Transmission on the Avalon Streaming Interface Signals Without Preamble Pass-Through

The figure illustrates how the octets of the client frame are transferred over the TX datapath when preamble pass-through is turned off.

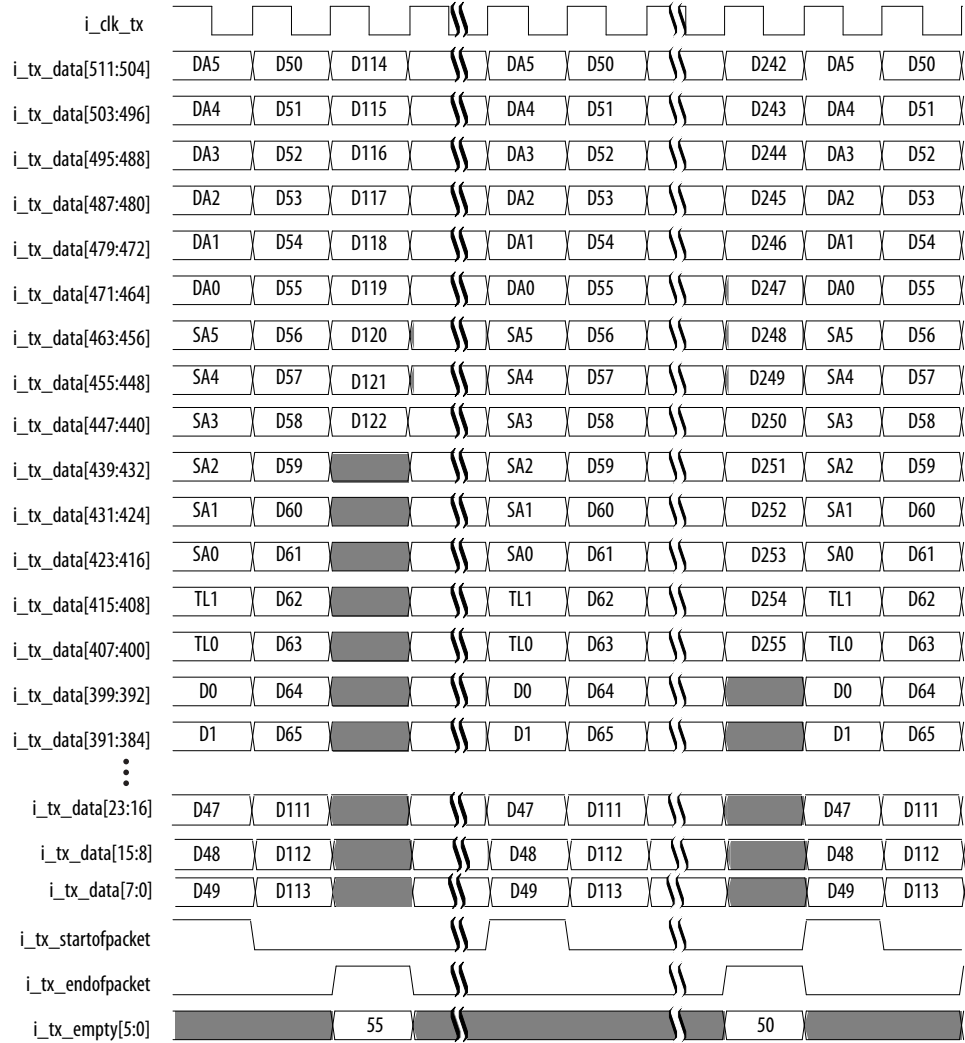


Figure 13. Byte Order on the Avalon Streaming Interface Lanes With Preamble Pass-Through

The figure describes the byte order on the Avalon streaming interface when the preamble pass-through feature is turned on.

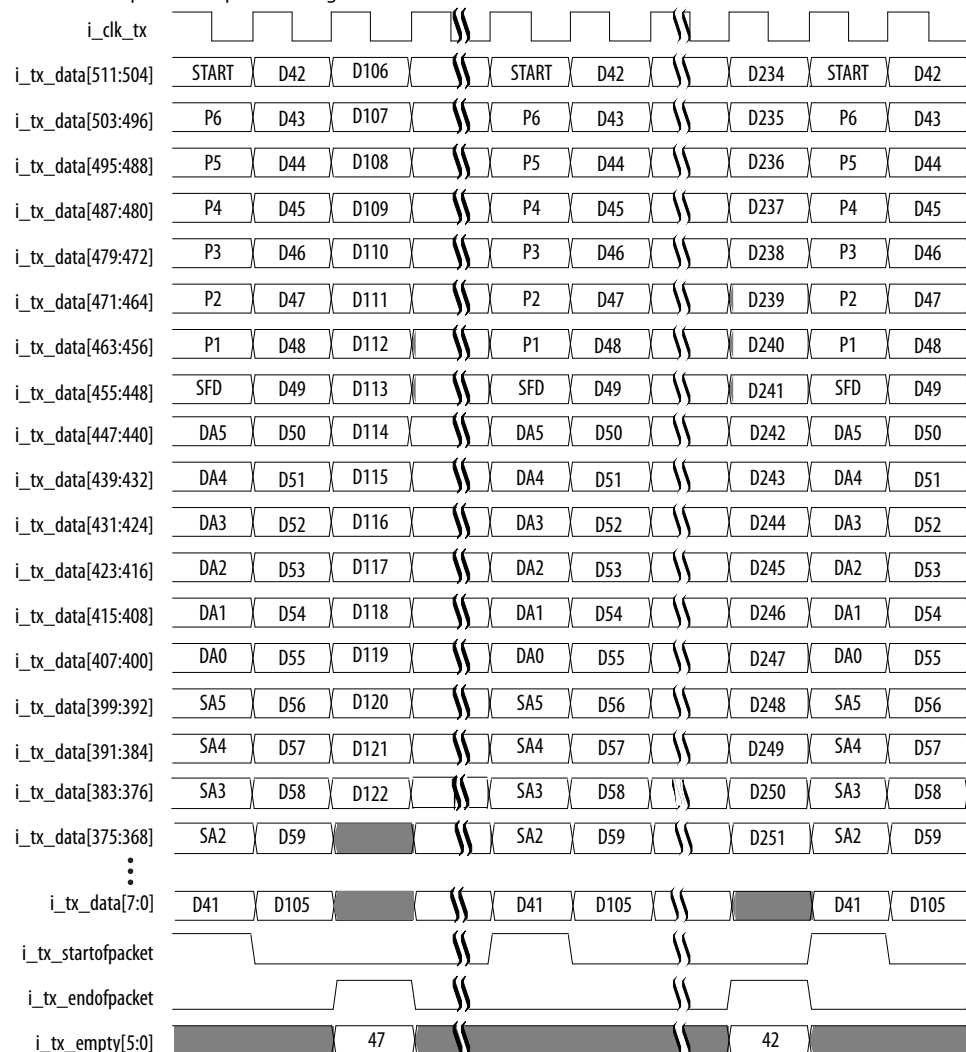
Destination Address[40] is the broadcast/multicast bit (a type bit), and Destination Address[41] is a locally administered address bit.

	SFD		Preamble					Start	Destination Address (DA)						Source Address (SA)						Type/ Length	Data (D)			
Octet	7	6	5	4	3	2	1	0	5	4	3	2	1	0	5	4	3	2	1	0	1	0	00	...	NN
Bit	[63:56]	[55:48]	[47:40]	[39:32]	[31:24]	[23:16]	[15:8]	[7:0]	[47:40]	[39:32]	[31:24]	[23:16]	[15:8]	[7:0]	[47:40]	[39:32]	[31:24]	[23:16]	[15:8]	[7:0]	[15:8]	[7:0]	MSB[7:0]	:	LSB[7:0]

Figure 14. Octet Transmission on the Avalon Streaming Interface Signals With Preamble Pass-Through

The figure illustrates how the octets of the client frame are transferred over the TX datapath when preamble pass-through is turned on. The eight preamble bytes precede the destination address bytes. The preamble bytes are reversed: the application must drive the SFD byte on `i_tx_data[455:448]` and the START byte on `i_tx_data[511:504]`.

The destination address and source address bytes follow the preamble pass-through in the same order as in the case without preamble pass-through.



4.3. PCS, OTN, and FlexE Modes

Each F-Tile Ethernet Intel FPGA Hard IP instance contains a full featured multi-lane PCS layer, which offers a number of interfacing options from the FPGA fabric.

The IP offers the following PCS options:

- PCS - This mode uses the MII interface to transmit and receive Ethernet packets for data rate of 10GE/25GE/50GE/100GE/200GE/400GE.
- OTN and FlexE - This mode uses the PCS66 interface to read and write 66 bit blocks data, from and to the PMA block.

4.3.1. PCS Mode

The F-Tile Ethernet Intel FPGA Hard IP supports PCS only mode in 10GE/25GE/50GE/100GE/200GE/400GE Ethernet rate variants with optional RS-FEC feature.

The TX PCS datapath consists of:

- TX PCS encoder—encodes the data from the PMA interface.
- TX PCS scrambler—enables the data to be scrambled. Channels does not lock correctly if the data is not scrambled.
- Alignment insertion—the TX PCS interface inserts alignment markers.
- Striper—enables logically sequential data to be segmented to increase data throughput.

The RX PCS datapath consists of:

- Aligner—enables the alignment of incoming data.
- RX PCS descrambler—enables the incoming scrambled data to be descrambled.
- RX PCS decoder—decodes the incoming encoded data from the PMA interface.

4.3.2. OTN Mode

The F-Tile Ethernet Intel FPGA Hard IP supports OTN mode in all Ethernet modes with optional RS-FEC feature.

The TX OTN datapath consists of:

- Alignment insertion—Alignment Marker and its position are made available for OTN based on the Ethernet mode of operation. Refer to [Signals of the PCS66 TX Interface](#) table for more information.
- Striper—enables logically sequential data to be segmented to increase data throughput.

Note: In OTN mode in 10GE/20GE/40GE/50GE/100GE Ethernet modes, scrambler is bypassed because the input data is expected to be scrambled. In 200GE/400GE Ethernet modes, RS-FEC block descrambles the data.

The RX OTN datapath consists of an aligner block that enables the alignment of the incoming data.

4.3.3. FlexE Mode

The F-Tile Ethernet Intel FPGA Hard IP supports all flexible Ethernet mode variants with optional RS-FEC feature. This mode bypassed the Ethernet MAC and uses PCS66 interface to read and write to the PMA block.

The TX FlexE datapath consists of:

- TX PCS scrambler—enables the data to be scrambled. Channels does not lock correctly if the data is not scrambled.
- Alignment insertion—the TX PCS interface inserts alignment markers
- Striper—enables logically sequential data to be segmented to increase data throughput.

The RX FlexE datapath consists of:

- Aligner—enables the alignment of incoming data.
- RX PCS descrambler—enables the incoming scrambled data to be descrambled.

4.4. Precision Time Protocol

The Precision Time Protocol (PTP) provides 1588 Precision Time Protocol timestamp information as defined in the *IEEE 1588-2008 Precision Clock Synchronization Protocol for Networked Measurement and Control System Standard*. 1588 PTP logic generates 96-bit timestamp information for transmitted and received packets. The generated timestamps represent the time when the first bit after the Start of Frame Delimiter (SFD) byte crosses the FPGA serial pins (Medium Dependent Interface, MDI).

In a 1-step TX operation, the IP core updates the PTP message content on-the-fly during the packet transmission. The update includes a 96-bit 1588 v2 format timestamp, updated correction field with residence time, asymmetry delay, and the peer-to-peer mean path delay. The IP core sets the UDP/IPv4 checksum field to zero and updates extended bytes to a correct checksum of the UDP/IPv6 packets.

A fingerprint can accompany a 1588 PTP packet. You can use this information to associate returned TX timestamp to the transmitted PTP packet. If you provide fingerprint information, the IP core passes it through unchanged.

The IP core connects to a time-of-day (TOD) module that continuously provides the current time of day based on the input clock frequency.

Once PTP is enabled, you must instantiate PTP tile adapter module (`eth_ptp_adpt_f`) per each F-tile. A single PTP tile adapter instance accommodates all PTP variants on the same F-tile. You must manually connect the bus interface between the PTP tile adapter and the associated F-Tile Ethernet Intel FPGA Hard IP. For more information about the PTP tile adapter, refer to [PTP Tile Adapter](#) on page 162.

Note: The PTP tile adapter requires dedicated use of EMIB 6 and EMIB 7 channels.

For proper connectivity between the F-Tile Ethernet Intel FPGA Hard IP in PTP mode and the supporting logic blocks, refer to the *F-Tile Ethernet Intel FPGA Hard IP Design Example*.

Related Information

- [PTP Tile Adapter](#)
- [F-Tile Ethernet Intel FPGA Hard IP Design Example User Guide](#)

4.4.1. Features

F-Tile Ethernet Intel FPGA Hard IP supports the following PTP features:

- Latency registers to accommodate for delay of external PHY components
- 10GE, 25GE, 50GE, 100GE, 200GE, and 400GE operating speed
- 1-step update 1588v2 96-bit timestamp
- 1-step update residence time in correction field
- 1-step set UPD/IPv4 checksum to zero
- 1-step update 2 byte of extended byte to ensure the UDP checksum remains correct
- 1-step asymmetry delay adjustment in correction field
- 1-step peer-to-peer mean path delay adjustment in correction field
- PTP statistics to keep track of number of packets with a PTP timestamp operation in TX and RX path
- Avalon memory-mapped interface accessible configuration, debug, and status registers
- Timestamp accuracy in Basic mode:
 - ± 3 ns for 10GE and 25GE modes
 - ± 8 ns for 50GE, 100GE, 200GE, and 400GE modes
- Timestamp accuracy in Advanced mode:
 - ± 1.5 ns for 10GE, 25GE, 50GE, 100GE, 200GE and 400GE modes

4.4.2. PTP Timestamp Accuracy

The IP core supports two timestamps accuracy modes, **Basic** and **Advanced**. In advanced mode, additional logic generates the high accuracy PTP timestamps.

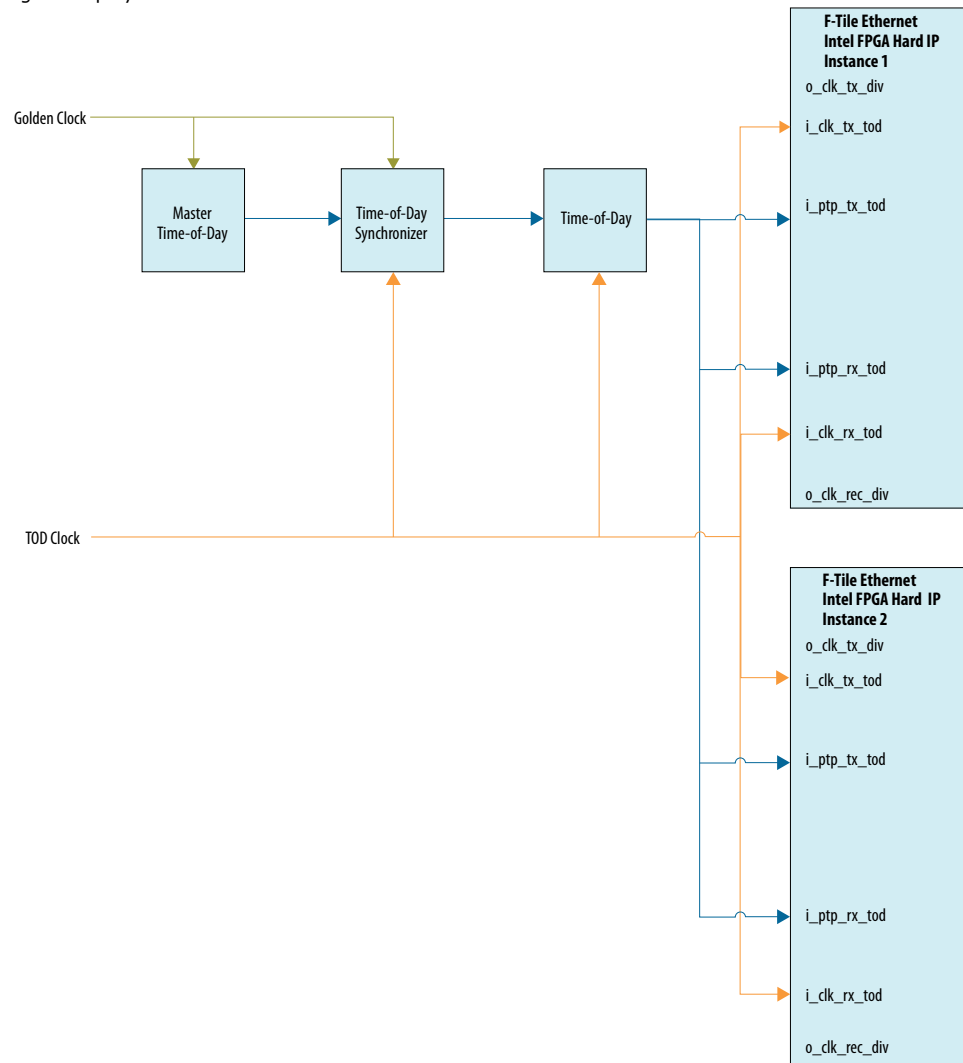
PTP Timestamp Accuracy in Basic Mode

When you select **Basic** in the **Timestamp accuracy mode** under PTP tab in the parameter editor for F-Tile Ethernet Intel FPGA Hard IP core, TX and RX data paths of multiple Ethernet IP instances share a single TOD clock output.

- The time-of-day (TOD) module can be shared across multiple IP instances of different data rates
- The F-Tile Ethernet Intel FPGA Hard IP instance 1 and instance 2, shown below, can be located on different F-tiles
- Any 390.625 MHz frequency clock source can drive the TOD clock

Figure 15. TOD Clock and TOD Connections in Basic Accuracy Mode

The figure displays recommended TOD clock connection. The TOD clock source should be 390.625 MHz.

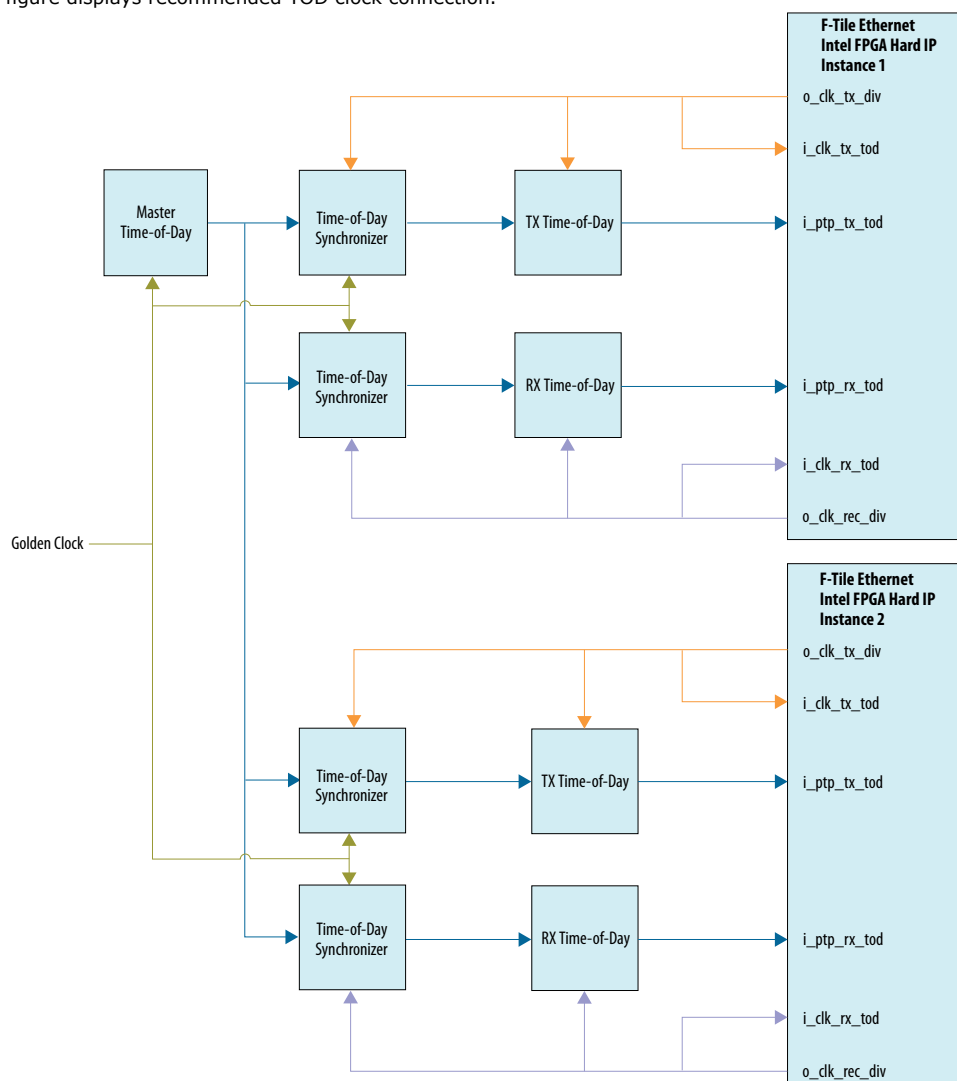


PTP Timestamp Accuracy in Advanced Mode

When you select **Advanced** in the **Timestamp accuracy mode** under PTP tab in the parameter editor, the IP core requires a single Master TOD module and dedicated TOD Synchronizer and TOD modules for both, TX and RX PTP instances.

Figure 16. TOD Clock and TOD Connections in Advanced Accuracy Mode

The figure displays recommended TOD clock connection.

**Table 15. PTP Timestamp Accuracy per Ethernet Data Rate Measured in Simulation**

Ethernet Data Rate	PTP Timestamp Accuracy	
	Basic Mode	Advanced Mode
10GE	± 3ns	± 1.5ns
25GE	± 3ns	± 1.5ns
50GE	± 8ns	± 1.5ns
100GE	± 8ns	± 1.5ns
200GE	± 8ns	± 1.5ns
400GE	± 8ns	± 1.5ns

4.4.3. PTP Client Flow

You must implement proper TX and RX data flows prior to sending PTP commands to the IP core and utilizing the timestamps.

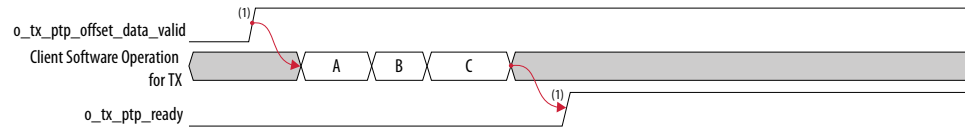
Attention: The following flows depict pseudo-code meant for conceptual, illustrative purposes. For definitive software routines, consult the design example.

The figures below depict the TX and RX client flows described in the *PTP TX Client Flow* and *PTP RX Client Flow* sections.

Figure 17. PTP TX Client Flow

The figure displays the following events in PTP TX client flow. For more information, refer to [PTP TX Client Flow](#) on page 52.

- A: Reading TX raw offset data from IP
- B: Calculating the TX offset value
- C: Writing calculated TX offset value to the IP

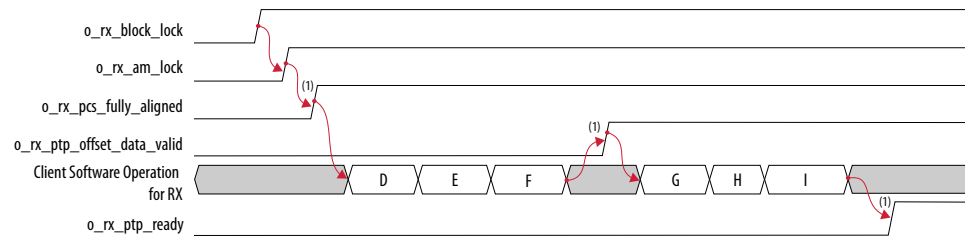


Note:
(1) The events are also available in status register, which client software can monitor by polling.

Figure 18. PTP RX with RS-FEC Client Flow

The figure displays the following events in PTP RX with RS-FEC client flow. For more information, refer to [PTP TX Client Flow](#) on page 52

- D: Reading RS-FEC `cw_pos` value from the IP
- E: Calculating `cw_pos` adjustment value
- F: Writing adjustment value to the IP
- G: Reading RX raw offset data from IP
- H: Calculating the RX offset value
- I: Writing calculated RX offset value to the IP

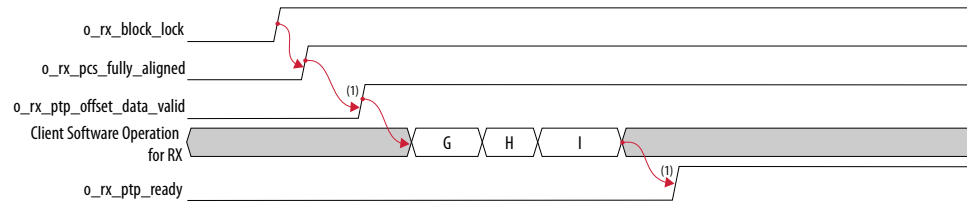


Note:
(1) The events are also available in status register, which client software can monitor by polling.

Figure 19. PTP RX without RS-FEC Client Flow

The figure displays the following events in PTP RX with no RS-FEC client flow. For more information, refer to [PTP RX Client Flow](#) on page 55.

- G: Reading RX raw offset data from IP
- H: Calculating the RX offset value
- I: Writing calculated RX offset value to the IP



Note:

(1) The events are also available in status register, which client software can monitor by polling.

Table 16. Client Flow Glossary

Term	Meaning
UI	Unit interval. Indicates bit time of one serial bit for specific speed. Unit interval is defined in 32-bit format, where bit [31:28] represent bit time in nanoseconds (ns) and bit [27:0] represent bit time in fractional nanoseconds (fns).
PL	Total number of PMA physical lanes of the variant
FL	Total number of FEC lanes of the variant
VL	Total number of virtual lanes of the variant
apl	Actual number of specific physical lane in a PMA quad. Possible values are 0, 1, 2, and 3. For more information, refer to the <i>Building Blocks</i> chapter in the <i>F-tile Architecture and PMA and FEC Direct PHY IP User Guide</i> .
pl	Logical number corresponding to a specific physical lane. pl=0 typically refers to the top-most active physical lane.
fl	Logical number of a specific FEC lane
vl	Logical number of a specific virtual lane
read(reg_name)	Performs CSR read from the reg_name register
write(reg_name, value)	Performs CSR write with value to the reg_name register.
tx_pma_delay_ui	Specifies TX serializer latency (in UI)
rx_pma_delay_ui	Specifies RX deserializer latency (in UI)
tx_external_phy_delay	Specifies TX external Ethernet PHY latency and board trace delay (in ns)
rx_external_phy_delay	Specifies RX external Ethernet PHY latency and board trace delay (in ns)

UI Format

The UI format differs from the format of other variables. UI uses the {4-bit ns, 28-bit fractional ns} format. Other variables defined in the PTP TX/RX client flows use the {N-bit ns, 16-bit fns} format, where N is the largest number to store the calculation's max value.

If you use UI format in your calculation, you must convert the result to a 16-bit fractional nanoseconds (fns) format.

Related Information

- [F-Tile Architecture and PMA and FEC Direct PHY IP User Guide](#)
- [PTP RX Client Flow](#) on page 55

4.4.3.1. PTP TX Client Flow

In this section, the acronyms PL and VL stand for Physical Lane and Virtual Lane respectively.

The following flows depict pseudo-code meant for the conceptual, illustrative purposes. For definitive software routines, refer to the design example.

Important: If IP undergoes TX reset at any point in this flow, you must restart the entire PTP TX client flow.

1. After power up or a TX reset, wait until TX raw offset data are ready.

You can monitor the status via one of the following:

- Output port:

```
o_tx_ptp_offset_data_valid = 1'b1
```

- Polling via Avalon memory-mapped interface register until it is asserted:

```
csr_read(ptp_status.tx_ptp_offset_data_valid) = 1'b1
```

2. Read TX raw offset data from IP:

```
tx_const_delay      = csr_read(ptp_tx_lane_calc_data_constdelay[30:0])
tx_const_delay_sign = csr_read(ptp_tx_lane_calc_constdelay[31])

for (pl = 0; pl < PL; pl++) {
    tx_apulse_offset[pl] = csr_read(ptp_tx_lane<pl>_calc_data_offset[30:0])
    tx_apulse_offset_sign[pl] = csr_read(ptp_tx_lane<pl>_calc_data_offset[31])
    tx_apulse_wdelay[pl] = csr_read(ptp_tx_lane<pl>_calc_data_wiredelay[19:0])
    tx_apulse_time[pl]   = csr_read(ptp_tx_lane<pl>_calc_data_time[27:0])
}
```

3. Determine TX reference lane:

The following sub-steps apply for designs with multiple lanes since any lane can be used as reference lane. You can skip the sub-steps for designs with one single PMA lane by setting the `tx_ref_pl = 0`.

- a. Detect rollover of asynchronous pulse time:

The `tx_apulse_time[pl]` signal represents an asynchronous time of each physical lane in a 28-bit format, where bit [27:16] represent asynchronous pulse time in nanoseconds (ns) and bit [15:0] represent asynchronous pulse time in fractional nanoseconds (fns).

Two types of rollover are possible:

- i. Natural rollover from bit 27 to bit 28 when the value reaches 28'hFFF_FFFF. Before rollover, bit [27:24] is 4'hF.
- ii. Billion rollover when the TOD reaches one billion ns or 48'h3B9A_CA00_0000 in hex value. Before rollover, bit [27:24] is 4'h9.

```

Given tx_apulse_time_max is largest tx_apulse_time from all physical
lanes,
for (pl = 0; pl < PL; pl++){
    if (tx_apulse_time_max - tx_apulse_time[pl] > 29'h01F4_0000){
        tx_apulse_time[pl] = tx_apulse_time[pl] + 29'h1000_0000
    } else {
        tx_apulse_time[pl] = tx_apulse_time[pl] + 29'h0A00_0000
    }
}

```

- b. Calculate the actual time of TX Alignment Marker at TX PMA parallel data interface.

```

for (pl = 0; pl < PL; pl++) {
    tx_am_actual_time[pl] =
        (tx_apulse_time[pl])
        + (tx_apulse_offset_sign[pl] ? -tx_apulse_offset[pl]
          : tx_apulse_offset[pl])
        - (tx_apulse_wdelay[pl])
}

```

- c. Determine TX reference lane:

```
tx_ref_pl = pl
```

The TX reference lane is the TX physical lane containing the largest tx_am_actual_time when comparing among all physical lanes.

4. Calculate TX offsets:

Attention: Step 4c is not applicable for 10G and 25G Ethernet data rates. You must skip step 4c for these rates.

- a. Calculate TX TAM adjust:

```

tx_tam_adjust_sim =
    (tx_const_delay_sign ? -tx_const_delay : tx_const_delay)
    + (tx_apulse_offset_sign[tx_ref_pl] ?
      -tx_apulse_offset[tx_ref_pl] : tx_apulse_offset[tx_ref_pl])
    - (tx_apulse_wdelay[tx_ref_pl])

```

For hardware run with PTP **Timestamp accuracy mode** set to **Advanced**:

```

tx_tam_adjust = (tx_tam_adjust_sim)
    + (tx_routing_adj_sign[tx_ref_pl] ? - tx_routing_adj[tx_ref_pl]
      : tx_routing_adj[tx_ref_pl])

```

For routing delay adjustment information, refer to the *Routing Delay Adjustment for Advanced Timestamp Accuracy Mode* section.

For all other cases:

```
tx_tam_adjust = tx_tam_adjust_sim
```

Convert TAM adjust to a 32-bit 2's complement number:

```
tx_tam_adjust_2c = tx_tam_adjust
where tx_tam_adjust is a 32-bit 2's complement number
```

- b. Calculate TX extra latency:

Convert unit of TX PMA delay from UI to nanoseconds. For UI value, refer to tables specified in *UI Value and PMA Delay*.

```
tx_pma_delay_ns = tx_pma_delay_ui * UI(13)
```

TX extra latency is a positive adjustment. To indicate the positive adjustment, set the most-significant register bit to 0. Total up all extra latency together:

```
tx_extra_latency[31] = 0
tx_extra_latency[30:0] = tx_pma_delay_ns + tx_external_phy_delay
```

c. Calculate TX virtual lane offsets:

Use VL0 as the reference virtual lane. Assign TX virtual lane offset values according to virtual lane order.

- For KP-FEC or LL-FEC variants:

Note: % is the modulo operator.

```
for (vl = 0; vl < VL; vl++) {
    tx_vl_offset[vl] = [vl - (vl % PL)] / PL * 68 * UI(13)
}
```

- For KR-FEC variants:

```
for (vl = 0; vl < VL; vl++) {
    tx_vl_offset[vl] = [vl - (vl % PL)] / PL * 66 * UI(13)
}
```

- For no FEC variants:

```
for (vl = 0; vl < VL; vl++) {
    tx_vl_offset[vl] = [vl - (vl % PL)] / PL * 1 * UI(13)
}
```

5. Write the determined TX reference lane into IP:

```
csr_write (ptp_ref_lane.tx_ref_lane, tx_ref_pl)
```

6. Write the calculated TX offsets to IP:

Attention: Step 6a is not applicable for 10G and 25G Ethernet data rates. You must skip step 6a for these rates.

a. Write TX virtual lane offsets:

```
for (vl = 0; vl < VL; vl++) {
    csr_write(tx_ptp_vl_offset_vl, tx_vl_offset[vl])
}
```

b. Write TX extra latency:

```
csr_write(tx_ptp_extra_latency, tx_extra_latency)
```

c. Write TX TAM adjust:

```
csr_write(ptp_tx_tam_adjust, tx_tam_adjust_2c)
```

⁽¹³⁾ The UI format differs from the format of other variables. UI uses the {4-bit ns, 28-bit fractional ns} format. Other variables defined in this flow use the {N-bit ns, 16-bit fractional ns} format, where N is the largest number to store the calculation's max value. If you use UI format in your calculation, you must convert your result to a 16-bit fractional ns format.

7. Notify soft PTP that uses flow configuration is completed.

```
csr_write(ptp_tx_user_cfg_status.tx_user_cfg_done, 1'b1)
```

8. UI value measurement. Follow the steps mentioned in the *TX UI Adjustment* section..

For simulation or hardware run with 0 ppm setup, you can skip the measurement and program 0 ppm UI value defined in *UI Adjustment*.

9. Wait until TX PTP is ready.

You can monitor the status via one of the following:

- Output port:

```
o_tx_ptp_ready = 1'b1
```

- Polling via CSR:

```
csr_read(ptp_status.tx_ptp_ready) = 1'b1
```

10. TX PTP is up and running.

- a. Adjust TX UI value.

Perform the TX UI adjustment occasionally to prevent time counter drift from golden time-of-day in the system. Follow the steps described in *TX UI Adjustment*.

Note: UI measurement is a long process in simulation. Therefore, for simulation, Intel recommends skipping this step and program a 0 ppm value.

Related Information

- [UI Adjustment](#) on page 64
- [TX UI Adjustment](#) on page 65
- [UI Value and PMA Delay](#) on page 73
- [Routing Delay Adjustment for Advanced Timestamp Accuracy Mode](#) on page 74

4.4.3.2. PTP RX Client Flow

In this section, the acronyms PL and VL stand for Physical Lane and Virtual Lane respectively.

The following flows depict pseudo-code meant for the conceptual, illustrative purposes. For definitive software routines, refer to the design example.

Note: RX PTP Ready signal goes down when the TX is reset. RX PTP Ready signal retunes to up when the TX reset is released, RX link is not lost or RX Reset is not applied during the TX Reset applied interval, without the need to run the initialization flow.

1. After power on or RX reset or reachieve a lost RX link, wait until RX PCS is fully aligned.

You can monitor the status via one of the following:

- Output port:

```
o_rx_pcs_fully_aligned = 1'b1
```

- Polling via Avalon memory-mapped interface register until it is asserted:

- For 10GE - 100GE no FEC variants and 25G FEC variants:

```
csr_read(phy_rxpcs_status.rx_aligned) = 1'b1
```

- For 50GE - 400GE FEC variants:

```
csr_read(rsfec_aggr_rx_stat.not_align) = 1'b0
```

2. For FEC variants, configure RX FEC codeword position into the transceiver.

Attention: You must skip this step for non-FEC variants.

- a. Read RX FEC codeword position and FEC lane mapping for each PMA lane.

Determine mapping from a PMA lane to a FEC lane.

```
pl_fl_map = Speed / 25 / PL
where Speed = {25, 50, 100, 200, 400}

for (fl = 0; fl < FL; fl++) {
    rx_fec_cw_pos[fl] = csr_read(rsfec_cw_pos_rx[fl][14:0])
}
```

- b. Calculate pulse adjustments and check for FEC cw_pos rollover between FEC lane received from the same transceiver lane.

```
for (fl = 0; fl < FL; fl++) {
    if ((rx_fec_cw_pos[fl] >= rx_fec_cw_pos[fl-(fl%pl_fl_map)])
        && (rx_fec_cw_pos[fl]-rx_fec_cw_pos[fl-(fl%pl_fl_map)] > 15'h4E20))
    {
        KR FEC Variants:
        rx_xcvr_if_pulse_adj[fl] = 15'h5280 - rx_fec_cw_pos[fl]

        KP & LL FEC Variants:
        rx_xcvr_if_pulse_adj[fl] = 15'h5500 - rx_fec_cw_pos[fl]

        rx_xcvr_if_pulse_adj_sign[fl] = 1'b1
    } else if ((rx_fec_cw_pos[fl-(fl%pl_fl_map)] > rx_fec_cw_pos[fl])
        && (rx_fec_cw_pos[fl - (fl%pl_fl_map)]
            - rx_fec_cw_pos[fl] > 15'h4E20))
    {
        KR FEC Variants:
        rx_xcvr_if_pulse_adj[fl] = 15'h5280 + rx_fec_cw_pos[fl]

        KP & LL FEC Variants:
        rx_xcvr_if_pulse_adj[fl] = 15'h5500 + rx_fec_cw_pos[fl]

        rx_xcvr_if_pulse_adj_sign[fl] = 1'b0
    } else {
        rx_xcvr_if_pulse_adj[fl] = rx_fec_cw_pos[fl]
        rx_xcvr_if_pulse_adj_sign[fl] = 1'b0
    }
}
```

- c. Write the pulse adjustments into the IP:

For multiple FEC lanes that interleave within single transceiver, select adjustment value from FEC lane with lowest index. You must ensure correct mapping between PMA lane and the corresponding CSR registers.

- For FGT transceiver:

```
for (pl = 0; pl < PL; pl++) {
    csr_write(ux_q_dl_ctrl_a_l<apl>.cfg_rx_lat_bit_for_async[17:0],
              rx_xcvr_if_pulse_adj[pl*pl_fl_map])
}
```

Note: There are 4 FGT quads with 4 apl lanes each. You must program registers of all active quad lanes. For information on accessing different FGT quads, refer to the *User Guide*.

- For FHT transceiver:

```
for (pl = 0; pl < PL; pl++) {
    csr_write(rxdl_async_l<apl>.cfg_rx_lat_bit_for_async_lane<pl>[17:0],
              rx_xcvr_if_pulse_adj[pl*pl_fl_map])
}
```

Note: Both apl and pl are used in this step. For more information about the physical lanes, refer to Client Flow Glossary table in the *PTP Client Flow* section.

- Notify soft PTP that pulse adjustments have been configured.

```
csr_write(ptp_rx_user_cfg_status.rx_fec_cw_pos_cfg_done, 1'b1)
```

- Wait until RX raw offset data are ready.

You can monitor the status via one of the following:

- Output port:

```
o_rx_ptp_offset_data_valid = 1'b1
```

- Polling via CSR:

```
csr_read(ptp_status.rx_ptp_offset_data_valid) = 1'b1
```

- Read RX raw offset data from IP:

- All variants:

```
rx_const_delay = csr_read(ptp_rx_lane_calc_data_constdelay[30:0])
rx_const_delay_sign = csr_read(ptp_rx_lane_calc_data_constdelay[31])

for (pl = 0; pl < PL; pl++) {
    rx_apulse_offset[pl] = csr_read(ptp_rx_lane<pl>_calc_data_offset[30:0])
    rx_apulse_offset_sign[pl] =
        csr_read(ptp_rx_lane<pl>_calc_data_offset[31])
    rx_apulse_wdelay[pl] =
        csr_read(ptp_rx_lane<pl>_calc_data_wiredelay[19:0])
    rx_apulse_time[pl] = csr_read(ptp_rx_lane<pl>_calc_data_time[27:0])
}
```

- 10GE/25GE no FEC variants:

```
rx_bitslip_cnt = csr_read(bitslip_cnt.bitslip_cnt[6:0])
rx_dlpulse_alignment = csr_read(bitslip_cnt.dlpulse_alignment)
```

- Determine RX reference lane.

Skip steps 5b, 5c, and 5d for single lane 10G and 25G Ethernet modes. For these variants, use:

```
rx_ref_pl = 0
rx_ref_fl = 0
rx_ref_vl = 0
```

- a. Determine synchronous pulse (Alignment Marker (AM)) offsets with reference to asynchronous pulse.

- FEC variants:

```
for (fl = 0; fl < FL; fl++) {
    if (rx_xcvr_if_pulse_adj_sign[fl]){
        rx_spulse_offset[fl] = rx_xcvr_if_pulse_adj[fl] * UI * pl_fl_map
        rx_spulse_offset_sign[fl] = 1'b1;
    } else
    {
        if ((rx_xcvr_if_pulse_adj[fl]
        + rx_xcvr_if_pulse_adj[fl-(fl%pl_fl_map)][4:0])
        > rx_xcvr_if_pulse_adj[fl-(fl%pl_fl_map)])
        {
            rx_spulse_offset[fl] =
            (rx_xcvr_if_pulse_adj[fl]
            - rx_xcvr_if_pulse_adj[fl-(fl%pl_fl_map)]
            + rx_xcvr_if_pulse_adj[fl-(fl%pl_fl_map)][4:0]) * UI(14) * pl_fl_map
            rx_spulse_offset_sign[fl] = 1'b0;
        } else
        {
            rx_spulse_offset[fl] =
            (rx_xcvr_if_pulse_adj[fl-(fl%pl_fl_map)]
            - rx_xcvr_if_pulse_adj[fl]
            - rx_xcvr_if_pulse_adj[fl-(fl%pl_fl_map)][4:0]) * UI(14) * pl_fl_map
            rx_spulse_offset_sign[fl] = 1'b1;
        }
    }
}
```

- 50GE/100GE no FEC variants:

```
for (vl = 0; vl < VL; vl++) {
    rx_spulse_post_am[vl] = <Refer to RX Virtual Lane Offset
    Calculation for No FEC Variants>
    rx_spulse_offset[vl] = (AM_INTERVAL - rx_spulse_post_am[vl])
    * UI(14)
    rx_spulse_offset_sign[vl] = 1'b0;
}
```

The offset calculated in *RX Virtual Lane Offset Calculation for No FEC Variants* measures the bit distance from PCS internal AM to synchronous pulse of each virtual lane. To determine the bit distance from synchronous pulse to the next AM, subtract the calculated value from the AM interval.

Variants	AM INTERVAL in Simulation	AM INTERVAL in Hardware
50GE-2	2,560 * 66 = 168,960	32,768 * 66 = 2,162,688
100GE-4	2,560 * 66 = 168,960	81,920 * 66 = 5,406,720

- For 10GE/25GE no FEC variants:

```
rx_spulse_offset[0] = (rx_bitslip_cnt + rx_dlpulse_alignment*33)*UI(14)
rx_spulse_offset_sign[0] = 1'b0;
```

- b. Detect rollover of asynchronous pulse time:

⁽¹⁴⁾ The UI format differs from the format of other variables. UI uses the {4-bit ns, 28-bit fractional ns} format. Other variables defined in this flow use the {N-bit ns, 16-bit fractional ns} format, where N is the largest number to store the calculation's max value. If you use UI format in your calculation, you must convert your result to a 16-bit fractional ns format.

The `rx_apulse_time[pl]` signal represents an asynchronous pulse time of each physical lane in a 28-bit format, where bit [27:16] represent asynchronous pulse time in nanoseconds (ns) and bit [15:0] represent asynchronous pulse time in fractional nanoseconds (fns).

Two types of rollover are possible:

- i. Natural rollover from bit 27 to bit 28 when the value reaches `28'hFFF_FFFF`. Before rollover, bit [27:24] is `4'hF`.
- ii. Billion rollover when the TOD reaches one billion ns or `48'h3B9A_CA00_0000` in hex value. Before rollover, bit [27:24] is `4'h9`.

```
Given rx_apulse_time_max is largest rx_apulse_time from all physical lanes,
for (pl = 0; pl < PL; pl++){
    if (rx_apulse_time_max - rx_apulse_time[pl] > 29'h01F4_0000){
        if (rx_apulse_time_max[27:24] == 4'hF) {
            rx_apulse_time[pl] = rx_apulse_time[pl] + 29'h1000_0000
        } else {
            rx_apulse_time[pl] = rx_apulse_time[pl] + 29'h0A00_0000
        }
    }
}
```

- c. Calculate the actual time of RX Alignment Marker (AM) at RX PMA parallel data interface:

```
// fl to pl map
pl = fl to pl map(f) = (fl-(fl%pl_fl_map))/pl_fl_map
```

```
// vl to pl map
<See PL in RX Virtual Lane Offset Calculation for No FEC Variants>
```

- FEC variants:

```
for (fl = 0; fl < FL; fl++) {
    local_pl = fl_to_pl_map(fl)
    rx_am_actual_time[fl] = (rx_apulse_time[local_pl])
        + (rx_apulse_offset_sign[local_pl] ?
            -rx_apulse_offset[local_pl] : rx_apulse_offset[local_pl])
        - (rx_apulse_wdelay[local_pl])
        + (rx_spulse_offset_sign[fl] ?
            -rx_spulse_offset[fl] : rx_spulse_offset[fl])
}
```

- No FEC variants:

```
for (vl = 0; vl < VL; vl++) {
    local_pl = vl_to_pl_map(vl)
    rx_am_actual_time[vl] = (rx_apulse_time[local_pl])
        + (rx_apulse_offset_sign[local_pl] ?
            -rx_apulse_offset[local_pl] : rx_apulse_offset[local_pl])
        - (rx_apulse_wdelay[local_pl])
        + (rx_spulse_offset_sign[vl] ?
            -rx_spulse_offset[vl] : rx_spulse_offset[vl])
}
```

- d. Determine RX reference lane:

- FEC variants:

```
rx_am_actual_time[rx_ref_fl] is maximum value out of
rx_am_actual_time[FL-1:0]
rx_ref_pl = fl_to_pl_map(rx_ref_fl)

where rx_am_actual_time[fl] is max(rx_am_actual_time[FL-1:0])
```

- No FEC variants:

```
rx_am_actual_time[rx_ref_vl] is maximum value out of
rx_am_actual_time[VL-1:0]
rx_ref_pl = vl_to_pl_map(rx_ref_vl)

where rx_am_actual_time[vl] is max(rx_am_actual_time[VL-1:0])
```

6. Calculate RX offsets:

a. Calculate RX TAM adjust:

- FEC variants:

```
rx_tam_adjust_sim =
(rx_const_delay_sign ? -rx_const_delay : rx_const_delay)
+ (rx_apulse_offset_sign[rx_ref_pl] ?
  -rx_apulse_offset[rx_ref_pl] : rx_apulse_offset[rx_ref_pl])
- (rx_apulse_wdelay[rx_ref_pl])
+ (rx_spulse_offset_sign[rx_ref_fl] ?
  -rx_spulse_offset[rx_ref_fl] : rx_spulse_offset[rx_ref_fl])
```

- No FEC variants:

```
rx_tam_adjust_sim =
(rx_const_delay_sign ? -rx_const_delay : rx_const_delay)
+ (rx_apulse_offset_sign[rx_ref_pl] ?
  -rx_apulse_offset[rx_ref_pl] : rx_apulse_offset[rx_ref_pl])
- (rx_apulse_wdelay[rx_ref_pl])
+ (rx_spulse_offset_sign[rx_ref_vl] ?
  -rx_spulse_offset[rx_ref_vl] : rx_spulse_offset[rx_ref_vl])
```

For hardware run with PTP **Timestamp accuracy mode** set to **Advanced**:

```
rx_tam_adjust = (rx_tam_adjust_sim)
+ (rx_routing_adj_sign[rx_ref_pl] ? - rx_routing_adj[rx_ref_pl]
  : rx_routing_adj[rx_ref_pl])
```

For routing delay adjustment information, refer to *Routing Delay Adjustment for Advanced Timestamp Accuracy Mode*.

For all other cases:

```
rx_tam_adjust = rx_tam_adjust_sim
```

Convert TAM adjust to a 32-bit 2's complement number:

```
rx_tam_adjust_2c = rx_tam_adjust
where rx_tam_adjust is a 32-bit 2's complement number
```

b. Calculate RX extra latency:

Convert unit of RX PMA delay from UI to nanoseconds:

```
rx_pma_delay_ns = rx_pma_delay_ui * UI(14)
```

RX extra latency is a negative adjustment. To indicate the negative adjustment, set the most-significant register bit to 1. Total up all extra latency together:

```
rx_extra_latency[31] = 1
rx_extra_latency[30:0] = rx_pma_delay_ns + rx_external_phy_delay
```

- c. Calculate RX virtual lane offsets:

Attention: This step is not applicable for 10G and 25G Ethernet data rates. You must skip this step for these rates.

Using determined reference virtual lane, assign RX virtual lane offset values as described in *Virtual Lane Order and Offset Values*.

- For FEC KP-FEC or FEC LL-FEC variants:

```
for (vl = 0; vl < VL; vl++) {
    rx_vl_offset[vl] = [vl - (vl % PL)] / PL * 68 * UI(14)
}
```

- For FEC KR-FEC variants:

```
for (vl = 0; vl < VL; vl++) {
    rx_vl_offset[vl] = [vl - (vl % PL)] / PL * 66 * UI(14)
}
```

- For no FEC 100G Ethernet rate variants:

```
for (vl = 0; vl < VL; vl++) {
    rx_vl_offset[vl] = 2 * UI(14)
}
```

- For no FEC 50G Ethernet rate variants:

```
for (vl = 0; vl < VL; vl++) {
    rx_vl_offset[vl] = 0.5 * UI(14)
}
```

7. Write the determined RX reference lane into IP:

Attention: This step is not applicable for 10G and 25G Ethernet data rates. You must skip this step for the specified rates.

```
csr_write(ptp_ref_lane.rx_ref_lane, rx_ref_pl)
```

8. Write the calculated RX offsets to IP:

- a. Write RX virtual lane offsets:

Attention: This step is not applicable for 10G and 25G Ethernet data rates. You must skip this step for the specified rates.

```
for (vl = 0; vl < VL; vl++) {
    csr_write(rx_ptp_vl_offset_vl, rx_vl_offset[vl])
}
```

- b. Write RX extra latency:

```
csr_write(rx_ptp_extra_latency, rx_extra_latency)
```

- c. Write RX TAM adjust:

```
csr_write(ptp_rx_tam_adjust, rx_tam_adjust_2c)
```

9. Notify soft PTP that uses flow configuration is completed.

```
csr_write(ptp_rx_user_cfg_status.rx_user_cfg_done, 1'b1)
```

10. Continue UI value measurement. Follow steps 1 through 7 mentioned in the *RX UI Adjustment* section.

For simulation or hardware run with 0 ppm setup, you can skip the measurement and program 0 ppm UI value defined in *UI Adjustment*.

11. Wait until RX PTP is ready.

You can monitor the status via one of the following:

- Output port:

```
o_rx_ptp_ready = 1'b1
```

- Polling via CSR:

```
csr_read(ptp_status.rx_ptp_ready) = 1'b1
```

12. RX PTP is up and running.

- a. Adjust RX UI value.

Perform the RX UI adjustment occasionally to prevent time counter drift from golden time-of-day in the system. Follow steps 1 through 8 described in *RX UI Adjustment*.

Note: UI measurement is a long process in simulation. Therefore, for simulation, Intel recommends skipping this step and program a 0 ppm value. For more details, refer to *UI Value and PMA Delay*.

Related Information

- [PTP Client Flow](#) on page 50
- [UI Adjustment](#) on page 64
- [RX Virtual Lane Offset Calculation for No FEC Variants](#) on page 62
- [RX UI Adjustment](#) on page 67
- [UI Value and PMA Delay](#) on page 73
- [Routing Delay Adjustment for Advanced Timestamp Accuracy Mode](#) on page 74
- [Virtual Lane Order and Offset Values](#) on page 63

4.4.4. RX Virtual Lane Offset Calculation for No FEC Variants

1. Lock the RX PCS.

The RX PCS must be fully aligned before extracting VL Offset data. Wait for `o_rx_pcs_fully_aligned` to be asserted.

2. Read the VL data for each of the Local Virtual Lanes (VL):

Perform the read operation from `o_rx_pcs_status[mode]` register via CSR interface. This returns VL data field data for all 20 virtual lanes.

3. Set the Physical Lanes for each Remote Virtual Lane:

Step through the data provided by each Local Virtual lane:

- Each lane provides a `REMOTE_VL` and a `LOCAL_PL` value.

For each of the 20 possible Remote virtual lanes, set $PL[REMOTE_VL] = LOCAL_PL$.

For example, if you read Local Virtual lane 12, and get back $REMOTE_VL = 5$, $LOCAL_PL = 2$, that means the data for Virtual lane 5 from the link partner is coming in on our Physical lane 2. You store that as $PL[5] = 2$.

4. Calculate the Virtual Lane Offsets for each of the Remote Virtual Lanes in bits:

```
For 100GE rate:
Virtual Lane Offset = sync_pulse to last AM
= gb_33_66_occupancy + gb_66_110_occupancy + blk_align_occupancy
+ am_detect_occupancy + am_count - local_lane_adjust
```

```
For 50GE-2 rate:
Virtual Lane Offset = sync_pulse to last AM
= gb_33_66_occupancy + sep50_occupancy + blk_align_occupancy
+ am_detect_occupancy + am_count - local_lane_adjust
```

5. Apply shifts to the offsets for $REMOTE_VLs \{18..19\}$ to account for the `mii_am/mii_data` reordering that happens in the PCS MII Decoder. This step is applicable for 50GE-2 and 100GE rates only.

- Generate `vl_offset_bits` shifted for each of the $REMOTE_VLs$.
- When using RX PTP on data from the RX PCS, shift the times for $REMOTE_VLs$ 18 to 19 by -330 bits:

```
vl_offset_bits_shifted[REMOTE_VL] = vl_offset_bits[REMOTE_VL] - 330 for
REMOTE_VL == {18..19}
vl_offset_bits_shifted[REMOTE_VL] = vl_offset_bits[REMOTE_VL] for
REMOTE_VL == {0..17}
```

This shift accounts for a shift that is applied by the RX PCS to incoming data.

For example, if `vl_offset_bits[18] == 887`, set `vl_offset_bits_shifted[18]` to 557. If `vl_offset_bits[17] == 887`, set `vl_offset_bits_shifted[17]` to 887.

Note: For 50GE-2 rates, follow same instructions except only apply the shifting to VL3.

6. Convert the `vl_offset_bits` to `VL_OFFSET` (in ns):

```
For each REMOTE_VL, VL_OFFSET[REMOTE_VL] = vl_offset_bits_shifted[REMOTE_VL]
* RX_UI
```

Related Information

[PTP RX Client Flow](#) on page 55

4.4.5. Virtual Lane Order and Offset Values

As bit interleaving and lane distribution of virtual lanes and FEC lanes may be implemented differently in PCS and FEC of other devices, it is not possible to know the ordering implementation of data transmitted by link partner. Therefore the PTP implementation uses logical lane arrangement without interleaving. The tables below show examples of the virtual lane arrangement in 100GE (25GE-4 and 50GE-2) Ethernet rates.

Table 17. Virtual Lane Distribution in Physical Lanes of 100GE (25GE-4) Ethernet Rate

Physical Lane	T + 0	T + 1	T + 2	T + 3	T + 4
Lane 0	VL0	VL4	VL8	VL12	VL16
Lane 1	VL1	VL5	VL9	VL13	VL17
Lane 2	VL2	VL6	VL10	VL14	VL18
Lane 3	VL3	VL7	VL11	VL15	VL19

Table 18. Virtual Lane Distribution in Physical Lanes of 100GE (50GE-2) Ethernet Rate

Physical Lane	T + 0	T + 1	T + 2	T + 3	T + 4	T + 5	T + 6	T + 7	T + 8	T + 9
Lane 0	VL0	VL2	VL4	VL6	VL8	VL10	VL12	VL14	VL16	VL18
Lane 1	VL1	VL3	VL5	VL7	VL9	VL11	VL13	VL15	VL17	VL19

Related Information

PTP RX Client Flow on page 55

4.4.6. UI Adjustment

This section provides steps to measure UI value to prevent time drift due to clock PPM differences as referenced to golden time-of-day (TOD) in the platform.

Tip: The TOD value increments constantly based on the golden TOD running at the ideal clock with 0 ppm difference. If you observe a large adjustment to the TOD value, don't perform measurement or discard any performed measurement as it impacts the measurement result.

TAM is the reference time for Alignment Marker (AM) for variants with AM, and an arbitrary selected reference bit for variants without AM.

For simulation or hardware run with 0 ppm setting, you can skip the measurement and program 0 ppm UI value as defined in the table.

Table 19. Default or 0 ppm UI Value

Attention: The UI value provided in this table is different from the UI value at serial line in the *UI Value and PMA Delay* section.

Mode	0 ppm UI value (ps)	0 ppm UI value {4-bit ns, 28-bit fractional ns}
10GE-1	96.9697	32'h018D3019
25GE-1	38.7878	32'h 009EE00A
50GE-2		
100GE-4		
200GE-8		
50GE-1	19.3939	32'h004F7005
100GE-2		

continued...

Mode	0 ppm UI value (ps)	0 ppm UI value {4-bit ns, 28-bit fractional ns}
200GE-4	9.6969	32'h0027B802
400GE-8		
100GE-1		
200GE-2		
400GE-4		

Related Information

- [PTP TX Client Flow](#) on page 52
- [PTP RX Client Flow](#) on page 55
- [Reference Time Interval](#) on page 69
- [Minimum and Maximum Reference Time \(TAM\) Interval for UI Measurement \(Hardware\)](#) on page 71
- [UI Value and PMA Delay](#) on page 73

4.4.6.1. TX UI Adjustment

1. Request snapshot of initial TX TAM:

```
csr_write (ptp_uim_tam_snapshot.tx_tam_snapshot, 1'b1)
```

2. Read snapshotted initial TAM and counter values:

```
tx_tam_0_31_0 = csr_read (ptp_tx_uim_tam_info0.tam_31_0[31:0])
tx_tam_0_47_32 = csr_read (ptp_tx_uim_tam_info1.tam_47_32[15:0])
tx_tam_0_cnt = csr_read (ptp_tx_uim_tam_info1.tam_cnt[30:16])
tx_tam_0_valid = csr_read (ptp_tx_uim_tam_info1.tam_valid[31])
```

- If tx_tam_0_valid = 1, complete TAM by concatenating the initial TAM values:

```
tx_tam_0 = {tx_tam_0_47_32, tx_tam_0_31_0};
```

- If tx_tam_0_valid = 0, restart from Step1.

3. Starting from time when step 1 is executed, wait for time duration as specified in section *Minimum and Maximum Reference Time (TAM) Interval for UI Measurement (Hardware)*.

4. Request snapshot of Nth TX TAM:

```
csr_write (ptp_uim_tam_snapshot.tx_tam_snapshot, 1'b1)
```

5. Read snapshotted Nth TAM and counter values:

```
tx_tam_n_31_0 = csr_read (ptp_tx_uim_info0.tam_31_0[31:0])
tx_tam_n_47_32 = csr_read (ptp_tx_uim_tam_info1.tam_47_32[15:0])
tx_tam_n_cnt = csr_read (ptp_tx_uim_tam_info1.tam_cnt[30:16])
tx_tam_n_valid = csr_read (ptp_tx_uim_tam_info1.tam_valid[31])
```

Form the TAM by concatenating snapshotted Nth TAM values:

```
tx_tam_n = {tx_tam_n_47_32, tx_tam_n_31_0};
```

6. Check if there was a large change to TOD value impacting TAM value:

```
tx_tam_n_valid = csr_read (ptp_tx_uim_tam_infol.tam_valid[31])
```

If `tx_tam_n_valid = 0`, restart from Step 1. If you used `tx_tam_n` as `tx_tam_0` and `tx_tam_n_cnt` as `tx_tam_0_cnt`, you can skip Steps 1 and 2. Then, you can start the wait time in Step 3 when the Step 4 executes.

7. Calculation:

- a. Get TAM interval:

```
tx_tam_interval = <Refer to Reference Time Interval>
tx_tam_interval_per_pl = tx_tam_interval / PL
```

- b. Calculate time elapsed:

```
tx_tam_delta =
    (tx_tam_n <= tx_tam_0) ? [(tx_tam_n + 10^9 ns) - tx_tam_0]
    : (tx_tam_n - tx_tam_0)
```

Per Step 3, `tx_tam_0` and `tx_tam_n` difference must be within the expected time range.

- If `tx_tam_delta` (in ms) is lesser that the minimum time value specified by Time (ms) column of Table: Minimum and Maximum Reference Time (TAM) Interval Allowed for UI Measurement (Hardware Basic Mode) and Table: Minimum and Maximum Reference Time (TAM) Interval Allowed for UI Measurement (Hardware Advanced Mode) in the *Minimum and Maximum Reference Time (TAM) Interval for UI Measurement (Hardware)* section, discard the result and restart from Step 3.
- If `tx_tam_delta` (in ms) is greater than the maximum value specified by Time (ms) column of Table: Minimum and Maximum Reference Time (TAM) Interval Allowed for UI Measurement (Hardware Basic Mode) and Table: Minimum and Maximum Reference Time (TAM) Interval Allowed for UI Measurement (Hardware Advanced Mode) in the *Minimum and Maximum Reference Time (TAM) Interval for UI Measurement (Hardware)* section, discard the result and restart from Step 1.

Note: $10^9 \text{ ns} = 48'h \text{ 3B9A_CA00_0000}$

- c. Calculate TAM count value:

```
tx_tam_cnt = (tx_tam_n_cnt < tx_tam_0_cnt) ? [(tx_tam_n_cnt + 2^15) -
tx_tam_0_cnt]
                                                : (tx_tam_n_cnt -
tx_tam_0_cnt)
```

Per Step 3, `tx_tam_0` and `tx_tam_n` difference must be within the expected time range.

- If `tx_tam_cnt` (in ms) is lesser than the minimum time value specified by Number of Count column of Table: Minimum and Maximum Reference Time (TAM) Interval Allowed for UI Measurement (Hardware Basic Mode) and Table: Minimum and Maximum Reference Time (TAM) Interval Allowed for UI Measurement (Hardware Advanced Mode) in the *Minimum and Maximum Reference Time (TAM) Interval for UI Measurement (Hardware)* section, discard the result and restart from Step 3.
- If `tx_tam_cnt` (in ms) is greater than the maximum value specified by Number of Count column of Table: Minimum and Maximum Reference Time (TAM) Interval Allowed for UI Measurement (Hardware Basic Mode) and Table: Minimum and Maximum Reference Time (TAM) Interval Allowed for UI Measurement (Hardware Advanced Mode) in the *Minimum and Maximum Reference Time (TAM) Interval for UI Measurement (Hardware)* section, discard the result and restart from Step 1.

d. Calculate UI value:

```
tx_ui = (tx_tam_delta) / (tx_tam_cnt * tx_tam_interval_per_pl)
```

8. Write the calculated UI value to IP:

```
csr_write (tx_ptp_ui, tx_ui)
```

Ensure the format is {4-bit nanoseconds, 28-bit fractional nanoseconds}.

9. After first UI measurement, for every minimum TAM interval or longer duration, repeat step 1 to 8. This is to prevent time counter drift from golden time-of-day in the system whenever clock ppm changes.

Related Information

- [PTP TX Client Flow](#) on page 52
- [Minimum and Maximum Reference Time \(TAM\) Interval for UI Measurement \(Hardware\)](#) on page 71
- [Reference Time Interval](#) on page 69

4.4.6.2. RX UI Adjustment

1. Request snapshot of initial RX TAM:

```
csr_write (ptp_uim_tam_snapshot.rx_tam_snapshot, 1'b1)
```

2. Read snapshotted initial TAM and counter values:

```
rx_tam_0_31_0 = csr_read (ptp_rx_uim_tam_info0.tam_31_0[31:0])
rx_tam_0_47_32 = csr_read (ptp_rx_uim_tam_info1.tam_47_32[15:0])
rx_tam_0_cnt = csr_read (ptp_rx_uim_tam_info1.tam_cnt[30:16])
rx_tam_0_valid = csr_read (ptp_rx_uim_tam_info1.tam_valid[31])
```

- If `rx_tam_0_valid = 1`, complete TAM by concatenating the initial TAM values:

```
rx_tam_0 = {rx_tam_0_47_32, rx_tam_0_31_0};
```

- If `rx_tam_0_valid = 0`, restart from Step1.

3. Starting from time when step 1 is executed, wait for time duration as specified in the *Minimum and Maximum Reference Time (TAM) Interval for UI Measurement (Hardware)* section.

4. Request snapshot of Nth RX TAM:

```
csr_write (ptp_uim_tam_snapshot.rx_tam_snapshot, 1'b1)
```

5. Read snapshotted Nth TAM and counter values:

```
rx_tam_n_31_0 = csr_read (ptp_rx_uim_info0.tam_31_0[31:0])
rx_tam_n_47_32 = csr_read (ptp_rx_uim_tam_infol.tam_47_32[15:0])
rx_tam_n_cnt = csr_read (ptp_rx_uim_tam_infol.tam_cnt[30:16])
rx_tam_n_valid = csr_read (ptp_rx_uim_tam_infol.tam_valid[31])
```

Form the TAM by concatenating snapshotted Nth TAM values:

```
rx_tam_n = {rx_tam_n_47_32, rx_tam_n_31_0};
```

6. Check if there was a large change to TOD value impacting TAM value:

```
rx_tam_n_valid = csr_read (ptp_rx_uim_tam_infol.tam_valid[31])
```

If `rx_tam_n_valid = 0`, restart Step 1. If you used `rx_tam_n` as a new `rx_tam_0` and `rx_tam_n_cnt` as a new `rx_tam_0_cnt`, you can skip Step 1 and 2. Then, you can start the wait time in Step 3 when Step 4 executes.

7. Calculation:

a. Get TAM interval

```
rx_tam_interval = <Refer to Reference Time Interval>
rx_tam_interval_per_pl = rx_tam_interval / PL
```

b. Calculate time elapsed:

```
rx_tam_delta =
  (rx_tam_n <= rx_tam_0) ? [(rx_tam_n + 10^9ns) - rx_tam_0]
                        : (rx_tam_n - rx_tam_0)
```

Per Step 3, `rx_tam_0` and `rx_tam_n` difference must be within the expected time range.

- If `rx_tam_delta` (in ms) is lesser that the minimum time value specified by Time (ms) column of Table: Minimum and Maximum Reference Time (TAM) Interval Allowed for UI Measurement (Hardware Basic Mode) and Table: Minimum and Maximum Reference Time (TAM) Interval Allowed for UI Measurement (Hardware Advanced Mode) in the *Minimum and Maximum Reference Time (TAM) Interval for UI Measurement (Hardware)* section, discard the result and restart from Step 3.
- If `rx_tam_delta` (in ms) is greater than the maximum value specified by Time (ms) column of Table: Minimum and Maximum Reference Time (TAM) Interval Allowed for UI Measurement (Hardware Basic Mode) and Table: Minimum and Maximum Reference Time (TAM) Interval Allowed for UI Measurement (Hardware Advanced Mode) in the *Minimum and Maximum Reference Time (TAM) Interval for UI Measurement (Hardware)* section, discard the result and restart from Step 1 or Step 3 by using `rx_tam_n` as a new `rx_tam_0`.

Note: $10^9\text{ns} = 48'h\ 3B9A_CA00_0000$

c. Calculate TAM count value:

```
rx_tam_cnt = (rx_tam_n_cnt < rx_tam_0_cnt) ? [(rx_tam_n_cnt + 2^15) -
rx_tam_0_cnt]
                                                : (rx_tam_n_cnt -
rx_tam_0_cnt)
```

Per Step 3, `rx_tam_0` and `rx_tam_n` difference must be within the expected time range.

- If `rx_tam_cnt` (in ms) is lesser than the minimum time value specified by Number of Count column of Table: Minimum and Maximum Reference Time (TAM) Interval Allowed for UI Measurement (Hardware Basic Mode) and Table: Minimum and Maximum Reference Time (TAM) Interval Allowed for UI Measurement (Hardware Advanced Mode) in the *Minimum and Maximum Reference Time (TAM) Interval for UI Measurement (Hardware)* section, discard the result and restart from Step 3.
- If `rx_tam_cnt` (in ms) is greater than the maximum value specified by Number of Count column of Table: Minimum and Maximum Reference Time (TAM) Interval Allowed for UI Measurement (Hardware Basic Mode) and Table: Minimum and Maximum Reference Time (TAM) Interval Allowed for UI Measurement (Hardware Advanced Mode) in the *Minimum and Maximum Reference Time (TAM) Interval for UI Measurement (Hardware)* section, discard the result and restart from Step 1 or Step 3 by using `rx_tam_n` as a new `rx_tam_0`.

d. Calculate UI value:

```
rx_ui = (rx_tam_delta) / (rx_tam_cnt * rx_tam_interval_pl)
```

8. Write the calculated UI value to IP:

```
csr_write (rx_ptp_ui, rx_ui)
```

Ensure the format is {4-bit nanoseconds, 28-bit fractional nanoseconds}.

9. After first UI measurement, for every minimum TAM interval or longer duration, repeat step 1 to 8. This is to prevent time counter drift from the golden time-of-day in the system whenever the clock ppm changes.

Related Information

- [PTP RX Client Flow](#) on page 55
- [Minimum and Maximum Reference Time \(TAM\) Interval for UI Measurement \(Hardware\)](#) on page 71
- [Reference Time Interval](#) on page 69

4.4.7. Reference Time Interval

Table below displays the number of bits between two subsequent reference time captures. The UI adjustment calculation uses these numbers. To speed up the simulation, the number for simulation is smaller.

Table 20. Reference Time (TAM) Interval

FEC type:

- No FEC: No FEC
- CL74: IEEE 802.3 BASE-R Firecode (CL74)
- CL91: IEEE 802.3 RS(528,514) (CL91)
- CL134: IEEE 802.3 RS(544,514) (CL134)
- ETC: Ethernet Technology Consortium ETC RS(272, 258)

Mode	FEC Type	Simulation (bit)		Hardware (bit)	
		TX	RX ⁽¹⁵⁾	TX	RX
10GE	No FEC	168,960	168,960	5,406,720	168,960
25GE	No FEC	168,960	168,960	5,406,720	168,960
	CL74				
	CL91	168,960	168,960	5,406,720	5,406,720
50GE	No FEC (ETC)	337,920	337,920	4,325,376	4,325,376
	CL91	337,920	337,920	5,406,72	5,406,720
	CL134				
	ETC				
100GE	No FEC	675,840	675,840	21,626,880	21,626,880
	CL91				
	CL134				
	ETC				
200GE	CL134	1,351,680	1,351,680	21,626,880	21,626,880
	ETC				
400GE	CL134	2,703,360	2,703,360	43,253,760	43,253,760
	ETC				

Related Information

- [TX UI Adjustment](#) on page 65
- [RX UI Adjustment](#) on page 67
- [UI Adjustment](#) on page 64

⁽¹⁵⁾ Depends on the link partner AM. The numbers assume serial loopback.

4.4.8. Minimum and Maximum Reference Time (TAM) Interval for UI Measurement (Hardware)

Table 21. Minimum and Maximum Reference Time (TAM) Interval Allowed for UI Measurement (Hardware Basic Mode)

FEC type:

- No FEC: No FEC
- CL74: IEEE 802.3 BASE-R Firecode (CL74)
- CL91: IEEE 802.3 RS(528,514) (CL91)
- CL134: IEEE 802.3 RS(544,514) (CL134)
- ETC: Ethernet Technology Consortium ETC RS(272, 258)

Mode	FEC Type	TX Minimum Number of Reference Time Load Interval		TX Maximum Number of Reference Time Load Interval		RX Minimum Number of Reference Time Load Interval		RX Maximum Number of Reference Time Load Interval	
		Number of Count, N	Time (ms)	Number of Count, N	Time (ms)	Number of Count, N	Time (ms)	Number of Count, N	Time (ms)
10GE-1	No FEC	6	3.15	32,767	17,179.34	6	0.10	32,767	536.85
25GE-1	No FEC	6	1.26	32,767	6,871.74	6	0.04	32,767	214.74
	CL74								
	CL91	6	1.26	32,767	6,871.74	6	1.26	32,767	6,871.74
50GE-2	No FEC (ETC)	6	0.5	32,767	2,748.7	6	0.5	32,767	2,748.70
	CL91	6	0.63	32,767	3,435.87	6	0.63	32,767	3,435.87
	CL134								
50GE-1	CL134	6	0.63	32,767	3,435.87	6	0.63	32,767	3,435.87
	ETC								
100GE-4	No FEC	6	1.26	32,767	6,871.74	6	1.26	32,767	6,871.74
	CL91								
	CL134								
100GE-2	CL134	6	1.26	32,767	6,871.74	6	1.26	32,767	6,871.74
	ETC								
100GE-1	CL134	6	1.26	32,767	6,871.74	6	1.26	32,767	6,871.74
200GE-8	CL134	6	0.63	32,767	3,435.87	6	0.63	32,767	3,435.87
200GE-4	CL134	6	0.63	32,767	3,435.87	6	0.63	32,767	3,435.87
	ETC								
200GE-2	CL134	6	0.63	32,767	3,435.87	6	0.63	32,767	3,435.87
400GE-8	CL134	6	0.63	32,767	3,435.87	6	0.63	32,767	3,435.87
	ETC								
400GE-4	CL134	6	0.63	32,767	3,435.87	6	0.63	32,767	3,435.87

Table 22. Minimum and Maximum Reference Time (TAM) Interval Allowed for UI Measurement (Hardware Advanced Mode)

FEC type:

- No FEC: No FEC
- CL74: IEEE 802.3 BASE-R Firecode (CL74)
- CL91: IEEE 802.3 RS(528,514) (CL91)
- CL134: IEEE 802.3 RS(544,514) (CL134)
- ETC: Ethernet Technology Consortium ETC RS(272, 258)

Mode	FEC Type	TX Minimum Number of Reference Time Load Interval		TX Maximum Number of Reference Time Load Interval		RX Minimum Number of Reference Time Load Interval		RX Maximum Number of Reference Time Load Interval	
		Number of Count, N	Time (ms)	Number of Count, N	Time (ms)	Number of Count, N	Time (ms)	Number of Count, N	Time (ms)
10GE-1	No FEC	10	5.24	32,767	17,179.34	10	0.16	32,767	536.85
25GE-1	No FEC	10	2.1	32,767	6,871.74	10	0.07	32,767	214.74
	CL74								
	CL91	10	2.1	32,767	6,871.74	10	2.1	32,767	6,871.74
50GE-2	No FEC (ETC)	10	0.84	32,767	2,748.7	10	0.84	32,767	2,748.70
	CL91	10	1.05	32,767	3,435.87	10	1.05	33,767	3,435.87
	CL134								
50GE-1	CL134	10	1.05	32,767	3,435.87	10	1.05	32,767	3,435.87
	ETC								
100GE-4	No FEC	10	2.1	32,767	6,871.74	10	2.1	32,767	6,871.74
	CL91								
	CL134								
100GE-2	CL134	10	2.1	32,767	6,871.74	10	2.1	32,767	6,871.74
	ETC								
100GE-1	CL134	10	2.1	32,767	6,871.71	10	2.1	32,767	6,871.74
200GE-8	CL134	10	1.05	32,767	3,435.87	10	1.05	32,767	3,435.87
200GE-4	CL134	10	1.05	32,767	3,435.87	10	1.05	32,767	3,435.87
	ETC								
200GE-2	CL134	10	1.05	32,767	3,435.87	10	1.05	32,767	3,435.87
400GE-8	CL134	10	1.05	32,767	3,435.87	10	1.05	32,767	3,435.87
	ETC								
400GE-4	CL134	10	1.05	32,767	3,435.87	10	1.05	32,767	3,435.87

Related Information

- [TX UI Adjustment](#) on page 65
- [RX UI Adjustment](#) on page 67
- [UI Adjustment](#) on page 64

4.4.9. UI Value and PMA Delay

Table 23. UI Value and PMA Delay of Ethernet Modes

FEC type:

- No FEC: No FEC
- CL74: IEEE 802.3 BASE-R Firecode (CL74)
- CL91: IEEE 802.3 RS(528,514) (CL91)
- CL134: IEEE 802.3 RS(544,514) (CL134)
- ETC: Ethernet Technology Consortium ETC RS(272, 258)

PMA Rate (Gbps)	Ethernet Mode	FEC Type	UI Value		Simulation PMA Delay (UI)				Hardware PMA Delay ⁽¹⁶⁾ (UI)			
			ps	{4-bit ns, 28-bit fractional ns}	FGT		FHT		FGT		FHT	
					TX	RX	TX	RX	TX	RX	TX	RX
10.3125	10GE-1	No FEC	96.9697	32'h018D3019	80/33 ⁽¹⁷⁾	88/96 ⁽¹⁷⁾	276	295	79	88	320	282
25.78125	25GE-1	No FEC CL74 CL91	38.7878	32'h009EE00A								
25.78125	50GE-2 100GE-4	No FEC CL91										
26.5625	50GE-2 100GE-4 200GE-8	CL134	37.6471	32'h009A33CD								
53.125	50GE-1 100GE-2 200GE-4 400GE-8	CL134 ETC	18.8235	32'h004D19EC	158/63 ⁽¹⁷⁾	176/193 ⁽¹⁷⁾	552	589	158	175	640	584
106.25	100GE-1 200GE-2 400GE-4	CL134	9.4118	32'h00268CF3	N/A	N/A	1,094	1,037	N/A	N/A	1,186	1,124

Note: The FGT Fast Sim Model with PTP enabled can support up to 10,000 μ s simulation time. If you run the simulation beyond this point, the simulation behaves unpredictably.

Table 24. Number of Virtual Lanes per Ethernet Speed

Ethernet Speed	Number of Virtual Lanes (VL)
10GE	1
25GE	1
50GE	4
continued...	

⁽¹⁶⁾ The hardware PMA delay values are preliminary and subject to change.

⁽¹⁷⁾ This is the PMA delay UI value for the Fast Sim Model.

Ethernet Speed	Number of Virtual Lanes (VL)
100GE	20
200GE	8
400GE	16

Related Information

- [PTP TX Client Flow](#) on page 52
- [PTP RX Client Flow](#) on page 55
- [UI Adjustment](#) on page 64

4.4.10. Routing Delay Adjustment for Advanced Timestamp Accuracy Mode

In hardware, the existing fabric routing delays can impact the accuracy by a few hundred picoseconds. Follow the steps below to generate the routing delay information and apply the delay information to the TX/RX TAM adjust calculations described in the *PTP TX/RX Client Flow* sections.

For hardware run with **Timestamp accuracy mode** set to **Advanced**, use the `eth_f_ptp_report_dl_path_delay.tcl` script to generate the routing delay information.

1. Copy `<generated HDL directory>/ex_<speed>/eth_f_<version>/synth/eth_f_ptp_report_dl_path_delay.tcl` to your Intel Quartus Prime software `<proj dir>` folder.
2. Run full compilation.
3. In your `<proj dir>` folder, run the following command:

```
quartus_sta -t eth_f_ptp_report_dl_path_delay.tcl <project_name>
```

4. The generated TX and RX routing delay adjustment information is available in these views:

- In the Command Prompt window:

```
Info: PTP DL Path Routing delay adjustment summary:
Info: Mean TX Routing delay adjustment for IP_INST[0].hw_ip_top|dut|
eth_f_0 lane[0] = 0.702833333 ns
Info: Mean TX Routing delay adjustment for IP_INST[0].hw_ip_top|dut|
eth_f_0 lane[1] = 0.8525 ns
Info: Mean RX Routing delay adjustment for IP_INST[0].hw_ip_top|dut|
eth_f_0 lane[0] = 0.730833333 ns
Info: Mean RX Routing delay adjustment for IP_INST[0].hw_ip_top|dut|
eth_f_0 lane[1] = 0.803 ns
Info (23030): Evaluation of Tcl script
eth_f_ptp_report_dl_path_delay.tcl was successful
```

- In the generated `ptp_hw_adv_adj.tcl` script:

```
#Mean Routing Delay Adjustment (ns)
#IP0: IP_INST[0].hw_ip_top|dut|eth_f_0
set IP0_TX0_ROUTING_ADJ 0.702833333
```

```
set IP0_TX1_ROUTING_ADJ 0.8525
set IP0_RX0_ROUTING_ADJ 0.730833333
set IP0_RX1_ROUTING_ADJ 0.803
```

- In `ptp_hw_adv_adj.csv` file:

	A	B	C	D
1	Mean Routing Delay Adjustment Table			
2	Instance Name	Instance	Lane	Mean Routing Delay Adjustment (ns)
3	IP_INST[0].hw_ip_top dut eth_f_0	IP0	TX0	0.702833333
4	IP_INST[0].hw_ip_top dut eth_f_0	IP0	TX1	0.8525
5	IP_INST[0].hw_ip_top dut eth_f_0	IP0	RX0	0.730833333
6	IP_INST[0].hw_ip_top dut eth_f_0	IP0	RX1	0.803

5. Use the generated routing delay values in the `tx/rx_tam_adjust` calculation specified in [PTP TX Client Flow](#) on page 52 and [PTP RX Client Flow](#) on page 55.

Note: If you modify your project and rerun the compilation, you also must regenerate the routing delay information by following the steps above.

Related Information

- [PTP TX Client Flow](#) on page 52
- [PTP RX Client Flow](#) on page 55

4.4.11. Routing Delay Adjustment for Basic Timestamp Accuracy Mode

Follow the steps below to generate the routing delay information and apply the delay information to the TX/RX TAM adjust calculations described in the *PTP TX/RX Client Flow* sections.

For hardware run with **Timestamp accuracy mode** set to **Basic**, use the `eth_f_ptp_report_dl_path_delay.tcl` script to generate the routing delay information.

The steps to execute .tcl script is same as advanced accuracy mode. For more information to execute .tcl scripts, refer to the steps 1 to 3 in the [Routing Delay Adjustment for Advanced Timestamp Accuracy Mode](#) on page 74.

1. The generated TX and RX routing delay adjustment information is available in these views:

- In the Command Prompt window:

```
Info: PTP DL Path Routing delay adjustment summary:
Info: Mean TX Routing delay adjustment for IP_INST[0].hw_ip_top|dut|
eth_f_0 lane[0] = 0.48525 ns
Info: Mean TX Routing delay adjustment for IP_INST[0].hw_ip_top|dut|
eth_f_0 lane[1] = 0.3595 ns
Info: Mean RX Routing delay adjustment for IP_INST[0].hw_ip_top|dut|
eth_f_0 lane[0] = 0.56175 ns
Info: Mean RX Routing delay adjustment for IP_INST[0].hw_ip_top|dut|
eth_f_0 lane[1] = 0.67025 ns
Info (23030): Evaluation of Tcl script
eth_f_ptp_report_dl_path_delay.tcl was successful
```

- In the generated `ptp_hw_basic_adj.tcl` script:

```
#Mean Routing Delay Adjustment (ns)
#IP0: IP_INST[0] hw_ip_top|dut|eth_f_0
set IP0_TX0_ROUTING_ADJ 0.48525
```

```
set IP0_TX1_ROUTING_ADJ 0.3595
set IP0_RX0_ROUTING_ADJ 0.56175
set IP0_RX1_ROUTING_ADJ 0.67025
```

- In `ptp_hw_basic_adj.csv` file:

	A	B	C	D
1	Mean Routing Delay Adjustment Table			
2	Instance Name	Instance	Lane	Mean Routing Delay Adjustment (ns)
3	IP_INST[0].hw_ip_top dut eth_f_0	IP0	TX0	0.48525
4	IP_INST[0].hw_ip_top dut eth_f_0	IP0	TX1	0.3595
5	IP_INST[0].hw_ip_top dut eth_f_0	IP0	RX0	0.56175
6	IP_INST[0].hw_ip_top dut eth_f_0	IP0	RX1	0.67025

2. Use the generated routing delay values in the `tx/rx_tam_adjust` calculation specified in [PTP TX Client Flow](#) on page 52 and [PTP RX Client Flow](#) on page 55.

Note: If you modify your project and rerun the compilation, you also must regenerate the routing delay information by following the steps above.

4.5. Auto-Negotiation and Link Training

The Auto-negotiation and link training (AN/LT) implements auto-negotiation and link training protocols for the F-Tile Ethernet Intel FPGA Hard IP.

In F-tile architecture, the AN/LT functionality is decoupled from the F-Tile Ethernet Intel FPGA Hard IP into the F-Tile Auto-Negotiation and Link Training for Ethernet Intel FPGA IP.

Turn on the **Enable auto-negotiation and link training** parameter in the F-Tile Ethernet Intel FPGA Hard IP parameter editor to configure support for auto-negotiation and link training. Once enabled, you must instantiate one F-Tile Auto-Negotiation and Link Training for Ethernet Intel FPGA IP instance per each type of F-Tile Ethernet Intel FPGA Hard IP Ethernet port.

For more information, refer to the *F-Tile Auto-Negotiation and Link Training for Ethernet Intel FPGA IP* section.

Related Information

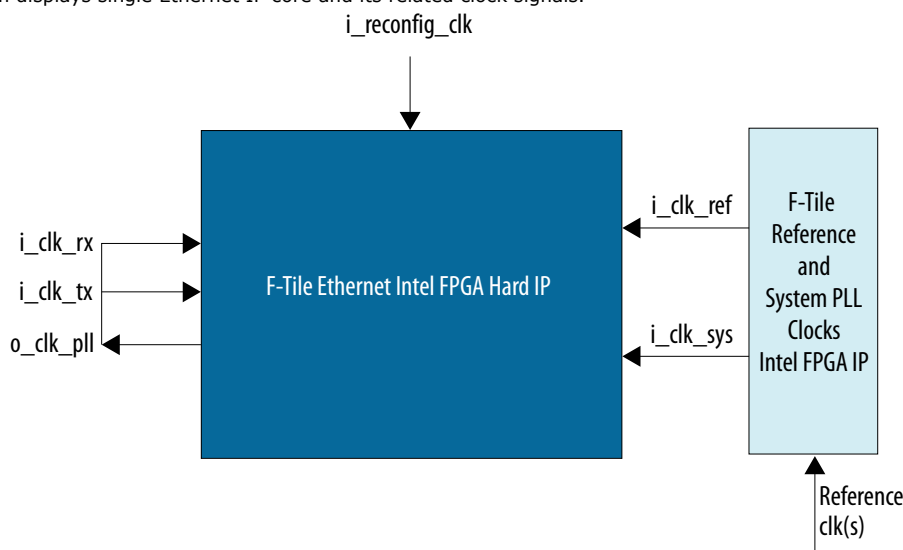
[F-Tile Auto-Negotiation and Link Training For Ethernet Intel FPGA IP](#) on page 153

5. Clocks

This section describes required clock connections and clock signals for various F-Tile Ethernet Intel FPGA Hard IP core variations.

Figure 20. Conceptual Overview of General IP Core Clock Connection

Diagram displays single Ethernet IP core and its related clock signals.



The F-Tile Reference and System PLL Clocks Intel FPGA IP generates `i_clk_ref` and `i_clk_sys` clocks that drive the F-Tile Ethernet Intel FPGA Hard IP (IP core).

The IP core supports the `i_reconfig_clk` frequency range of 100 to 250 MHz. The IP core output clock (`o_clk_pll`) drives both `i_clk_rx` and `i_clk_tx` input signals.

All IP core variations support the Synchronous Ethernet (SyncE) standard.

The table below describes necessary input and output clocks with required clock frequencies, and the clock-related status signals. You can use the clock status ports to hold circuits in reset or until the PLLs driving the clocks are locked.

Table 25. Clock Signals

Describes the input clocks you must provide, and the output clocks that the IP core provides.

Name	Description
Clock Inputs	
<code>i_clk_tx</code>	TX datapath clock
<i>continued...</i>	

Intel Corporation. All rights reserved. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

*Other names and brands may be claimed as the property of others.

Name	Description
	<p>This clock drives the active TX interface for the port.</p> <p>This clock source is:</p> <ul style="list-style-type: none"> o_clk_pll clock unless you enabled Enable asynchronous adapter clocks parameter o_clk_pll of the PTP tile adapter when Enable IEEE 1588 PTP parameter is enabled
i_clk_rx	<p>RX datapath clock</p> <p>This clock drives the active RX interface for the port.</p> <p>This clock source is:</p> <ul style="list-style-type: none"> o_clk_pll clock unless you enabled Enable asynchronous adapter clocks parameter o_clk_pll of the PTP tile adapter when Enable IEEE 1588 PTP parameter is enabled
i_clk_pll	<p>PTP-related datapath clock</p> <p>This clock drives the internal datapath clock for the port when both, Enable IEEE 1588 PTP and Enable asynchronous adapter clocks parameters, are enabled.</p> <p>This clock source is the o_clk_pll output of the PTP tile adapter. You must use a single clock source when using a multiple PTP ports in your design.</p> <p>Supports the following frequencies:</p> <ul style="list-style-type: none"> 402.83203125 MHz or higher for all Ethernet modes without FEC, with IEEE 802.3 BASE-R Firecode (CL74), or IEEE 802.3 RS(528,514) (CL91). The system PLL must be of 805.6640625 MHz frequency or higher. 415.0390625 MHz or higher for all Ethernet modes with IEEE 802.3 RS(544,514) (CL134), with Ethernet Technology Consortium RS(272, 258). The system PLL must be of 830.078125 MHz frequency or higher. Custom system PLL frequency supports 402.83203125 MHz or higher frequency <p>When Enable IEEE 1588 PTP parameter is disabled, connect this port to 1'b0.</p>
i_reconfig_clk	<p>Avalon memory-mapped interface reconfiguration clock</p> <p>The interface uses this clock to access control status registers (CSRs). The clock supports 100 to 250 MHz frequency.</p>
i_clk_ref	<p>PMA reference clock</p> <p>F-Tile Reference and System PLL Clock Intel FPGA IP drives this clock.</p> <ul style="list-style-type: none"> 156.25 MHz is the recommended frequency. Supported when using FHT PMA or when auto-negotiation and link training is enabled. 312.5 MHz when using FGT PMA without auto-negotiation and link training 322.265625 MHz when using FGT PMA without auto-negotiation and link training <p>You must specify this frequency in the F-Tile Ethernet Intel FPGA Hard IP PMA reference frequency IP parameter and in the F-Tile Reference and System PLL Clock Intel FPGA IP FGT refclk frequency/FHT refclk frequency IP parameter.</p> <p><i>Note:</i> The i_clk_ref is a virtual signal. In simulation, the signal displays as 0.</p> <p>The clock source depends on the F-Tile Ethernet Intel FPGA Hard IP PMA selection.</p> <ul style="list-style-type: none"> When using FGT PMA, the clock source is the out_refclk_fgt_i output signal from the F-Tile Reference and System PLL Clocks Intel FPGA IP. When using FHT PMA, the clock source is the out_fht_cmppll_clk_i output signal from the F-Tile Reference and System PLL Clocks Intel FPGA IP. <p>Unless the Custom cadence parameter is enabled, the clock must be PPM matched to the i_clk_sys clock.</p>
i_clk_sys	<p>Ethernet system clock</p> <p>F-Tile Reference and System PLL Clock Intel FPGA IP drives this clock.</p> <p>Unless Custom cadence parameter is enabled, the clock frequency depends on the FEC type:</p> <ul style="list-style-type: none"> 805.6640625 MHz or higher for all Ethernet modes without FEC, or with IEEE 802.3 BASE-R Firecode (CL74), or IEEE 802.3 RS(528,514) (CL91) 830.078125 MHz or higher for all Ethernet modes with IEEE 802.3 RS(544,514) (CL134), Ethernet Technology Consortium RS(272, 258) 322.265625 MHz or higher is also supported for 10GE without PTP

continued...

Name	Description
	<p>You must specify this frequency in the F-Tile Ethernet Intel FPGA Hard IP System PLL frequency IP parameter and in the F-Tile Reference and System PLL Clock Intel FPGA IP Mode of system PLL IP parameter.</p> <p><i>Note:</i> The <code>i_clk_sys</code> is a virtual signal. In simulation, the signal displays as 0.</p> <p>Connect to the <code>out_systempll_clk_i</code> signal from the F-Tile Reference and System PLL Clocks Intel FPGA IP.</p>
<code>i_sampling_clk</code>	<p>Input sampling clock</p> <p>This clock drives the Deterministic Latency Measurement module. The clock frequency is 250MHz.</p>
Clock Outputs	
<code>o_clk_pll</code>	<p>System PLL clock</p> <p>Clock derived from the F-Tile System PLL associated with the Ethernet IP port. The <code>o_clk_pll</code> frequency is equal to PLL frequency divided by 2. The following shows the <code>o_clk_pll</code> frequency unless you enabled custom system PLL frequency.</p> <p>Supports the following frequencies:</p> <ul style="list-style-type: none"> 402.83203125 MHz or higher for all Ethernet modes without FEC, with IEEE 802.3 BASE-R Firecode (CL74), or IEEE 802.3 RS(528,514) (CL91). The system PLL must be of 805.6640625 MHz frequency or higher. 415.0390625 MHz or higher for all Ethernet modes with IEEE 802.3 RS(544,514) (CL134), with Ethernet Technology Consortium RS(272, 258). The system PLL must be of 830.078125 MHz frequency or higher. 161.1328125 MHz or higher for 10GE without enabled PTP. The system PLL must be of 322.265625 MHz frequency or higher. Custom system PLL frequency divided by 2, if custom system PLL frequency is used
<code>o_clk_tx_div</code>	<p>Supports the following frequencies:</p> <ul style="list-style-type: none"> 156.25 MHz for 10GE 312.5 MHz for 40GE 390.625 MHz for all other Ethernet modes <p>Clock recovered from the TX SERDES rate divided by either 33/66/68, depending on the FEC mode and Ethernet mode parameters. The <code>o_clk_tx_div</code> is equal to:</p> <ul style="list-style-type: none"> TX SERDES rate divided by 33 for 40GE. TX SERDES rate divided by 66 when FEC mode parameter is set to one of the following: <ul style="list-style-type: none"> None except for 40GE IEEE 802.3 BASE-R Firecode (CL74) IEEE 802.3 RS(528,514) (CL91) TX SERDES rate divided by 68 when FEC mode parameter is set to one of the following: <ul style="list-style-type: none"> IEEE 802.3 RS(544,514) (CL134) Ethernet Technology Consortium RS(272, 258)
<code>o_clk_rec_div64</code>	<p>Supports the following frequencies:</p> <ul style="list-style-type: none"> 161.1328125 MHz \pm 200 PPM for 10GE/40GE 402.83203125 MHz \pm 200 PPM for Ethernet modes without FEC (except 10GE and 40GE), with IEEE 802.3 BASE-R Firecode (CL74), and IEEE 802.3 RS(528,514) (CL91) 415.0390625 MHz \pm 200 PPM for Ethernet modes with IEEE 802.3 RS(544,514) (CL134) and Ethernet Technology Consortium RS(272, 258) <p>Clock derived from RX recovered clock, divided by 64.</p>
<code>o_clk_rec_div</code>	<p>Supports the following frequencies:</p> <ul style="list-style-type: none"> 156.25 MHz \pm 200PPM for 10GE 312.50 MHz \pm 200PPM for 40GE 390.625 MHz \pm 200PPM for all other Ethernet modes
<i>continued...</i>	

Name	Description
	<p>Clock derived from the RX recovered clock divided by either 33/66/68, depending on the FEC mode parameter. The <code>o_clk_rec_div</code> is equal to:</p> <ul style="list-style-type: none"> RX SERDES rate divided by 33 for 40GE RX SERDES rate divided by 66 when FEC mode parameter is set to one of the following: <ul style="list-style-type: none"> None except for 40GE IEEE 802.3 BASE-R Firecode (CL74) IEEE 802.3 RS(528,514) (CL91) RX SERDES rate divided by 68 when FEC mode parameter is set to one of the following: <ul style="list-style-type: none"> IEEE 802.3 RS(544,514) (CL134) Ethernet Technology Consortium RS(272, 258)
Clock Status	
<code>o_tx_pll_locked</code>	<p>This clock indicates that the TX SERDES PLLs are locked.</p> <p><i>Note:</i> Do not use the <code>o_clk_tx_div</code> output clock until the <code>o_tx_pll_locked</code> signal is high.</p>
<code>o_cdr_lock</code>	<p>This clock indicates that the recovered clocks are locked to data.</p> <p><i>Note:</i> Do not use the <code>o_clk_rec_div64</code> output clock until the <code>o_cdr_lock</code> signal is high.</p>

Important Design Considerations

- In most Ethernet IP configurations, use the output clock `o_clk_pll` or an equivalent clock to drive the `i_clk_tx` and `i_clk_rx` signals. In asynchronous adapter option, you can use slower clocks to drive these signals.
- PTP channel only system clock divided by 2 at frequency of 402.83 MHz or higher. When PTP is enabled, all ports with PTP enabled share the same system clock.
- Recovered frequencies from a remote link partner are shown with ± 200 ppm range, assuming that local oscillator is ± 100 ppm and remote oscillator is (unrelated) ± 100 ppm. For SyncE applications, local oscillator must match recovered clock within ± 4.6 ppm.

You must configure the modes in the F-Tile Reference and System PLL Clock Intel FPGA IP. The table below displays the reference clock and output frequency based on a selected System PLL mode.

Table 26. Mode of System PLL: System PLL Reference Clock and Output Frequencies

Mode of System PLL	Reference Clock (MHz)	Output Frequency (MHz)
ETHERNET_FREQ_805_156	156.25	805.6640625
ETHERNET_FREQ_805_312	312.5	805.6640625
ETHERNET_FREQ_805_322	322.265625	805.6640625
ETHERNET_FREQ_830_156	156.25	830.078125
ETHERNET_FREQ_830_312	312.5	830.078125

5.1. Clock Connections in Single Instance Operation

This clock connection describes a single IP core instantiation in your design.

This is a typical clock connection requirement in a single IP core.

You must make the following clock connections:

- The `i_clk_ref` and the `i_clk_sys` clocks drive the IP core.
- The output clock `o_clk_pll` drives both the `i_clk_rx` and the `i_clk_tx` input signals.

Figure 21. Typical Clock Connections

This diagram displays single Ethernet IP core and its related clock signals.

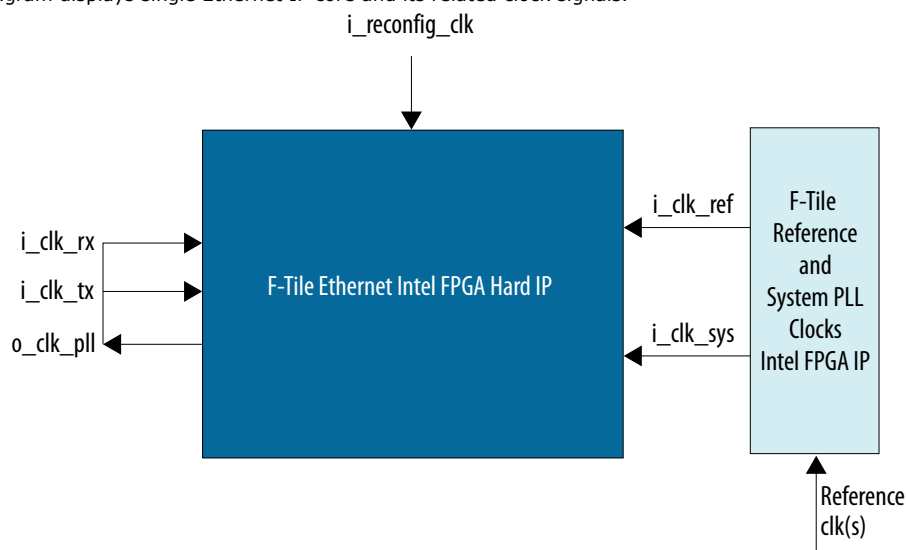


Table 27. Port Connection Guidelines between F-Tile Auto-Negotiation and Link Training for Ethernet Intel FPGA IP and F-Tile Ethernet Intel FPGA Hard IP

F-Tile Auto-Negotiation and Link Training for Ethernet Intel FPGA IP	F-Tile Ethernet Intel FPGA Hard IP
System PLL	
<code>out_systempll_clk</code>	<code>i_clk_sys</code>
FGT	
<code>out_refclk_fgt</code>	<code>i_clk_ref</code>
FHT	
<code>out_fht_cmmpll_clk</code>	<code>i_clk_ref</code>

5.2. Clock Connections in Multiple Instance Operation

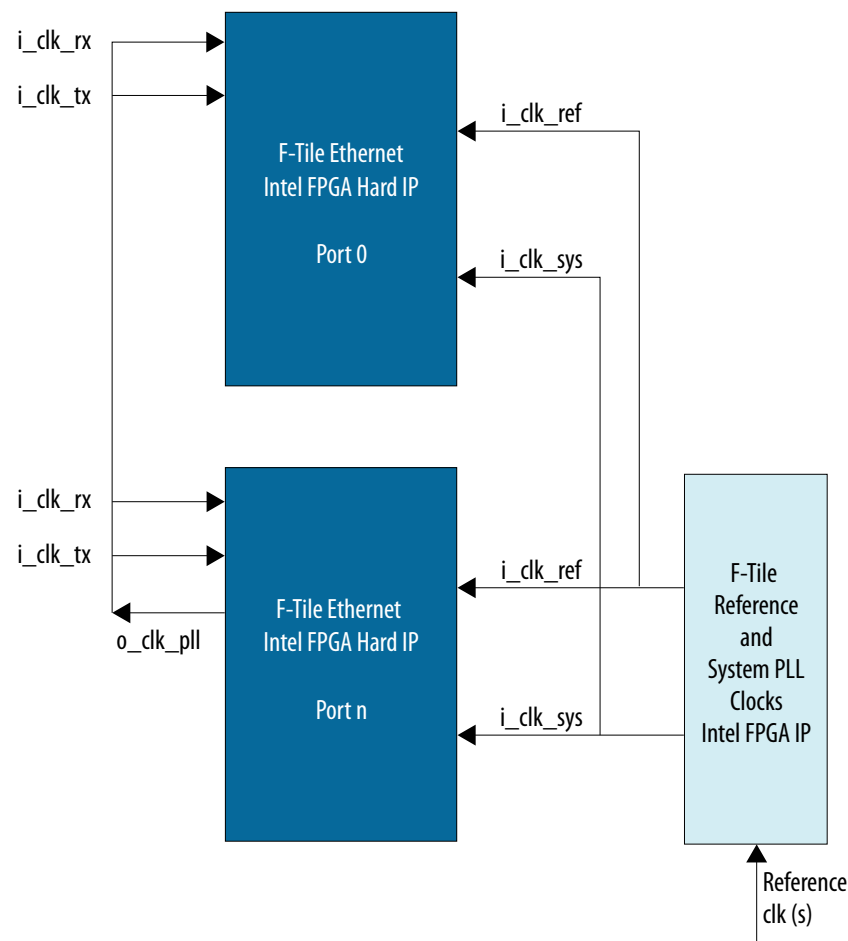
This clock connection describes multiple IP core instantiations in your design.

This is a recommended clocking for multiple IP core clock connections.

You must make the following clock connections:

- The `i_clk_ref` and the `i_clk_sys` clocks drives all instantiated IP cores.
- The output clock `o_clk_pll` of a single IP core can drive all instantiated IP core `i_clk_rx` and the `i_clk_tx` input signals under the following conditions:
 - The shared clock is traceable to a common source reference clock.
 - The F-Tile Ethernet Intel FPGA Hard IP uses the same rate to configure the its port system clocks.

Figure 22. Clock Connections for Multiple IP Cores



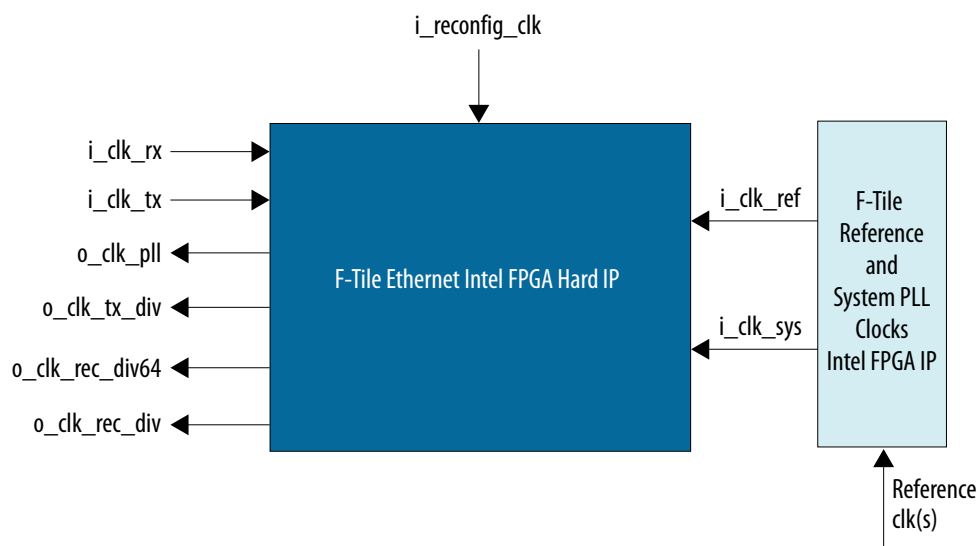
The following are examples of alternative clock sources satisfying the clock connections requirements:

- Another IP instance's `o_clk_pll` output clock can drive an IP core specific `i_clk_rx` and the `i_clk_tx` input signals provided that their respective reference clocks are configured at the same rate.
- An IO PLL can drive an IP core related input clock signals provided that the PLL and IP core derive their `i_clk_ref` reference clock from the same reference clock source.
- A GPIO, directly connected to the reference clock, with frequency of 161.1328125 MHz, can directly drive the `i_clk_rx` and the `i_clk_tx` input signals.

5.3. Clock Connections in MAC Asynchronous FIFO Operation

When you enable **Enable asynchronous adapter clocks**, `i_clk_tx` and `i_clk_rx` input clock signals can be asynchronous from each other and from `o_clk_pll` clock provided that the clocks are fast enough to ensure the IP core channel processes all data.

Figure 23. Clock Connections in MAC Asynchronous Client FIFO Operation



The table below summarizes minimum frequencies required for `i_clk_tx` and `i_clk_rx` during the Asynchronous mode.

Table 28. Minimum Supported Clock Rates for MAC Client Asynchronous FIFO Operation

Ethernet Data Rate	Clock Rate	
	Min <code>i_clk_tx</code>	Min <code>i_clk_rx</code>
10G	156.25 MHz	<code>o_clk_rec_div</code> or 156.25 MHz + 100 PPM
25G/50G	390.625 MHz	<code>o_clk_rec_div</code> or 390.625 MHz + 100 PPM
40G	312.5 MHz	312.5 MHz + 100 PPM
100G with enabled Preamble Passthrough	380 MHz	380 MHz
100G with disabled Preamble Passthrough	340 MHz	340 MHz

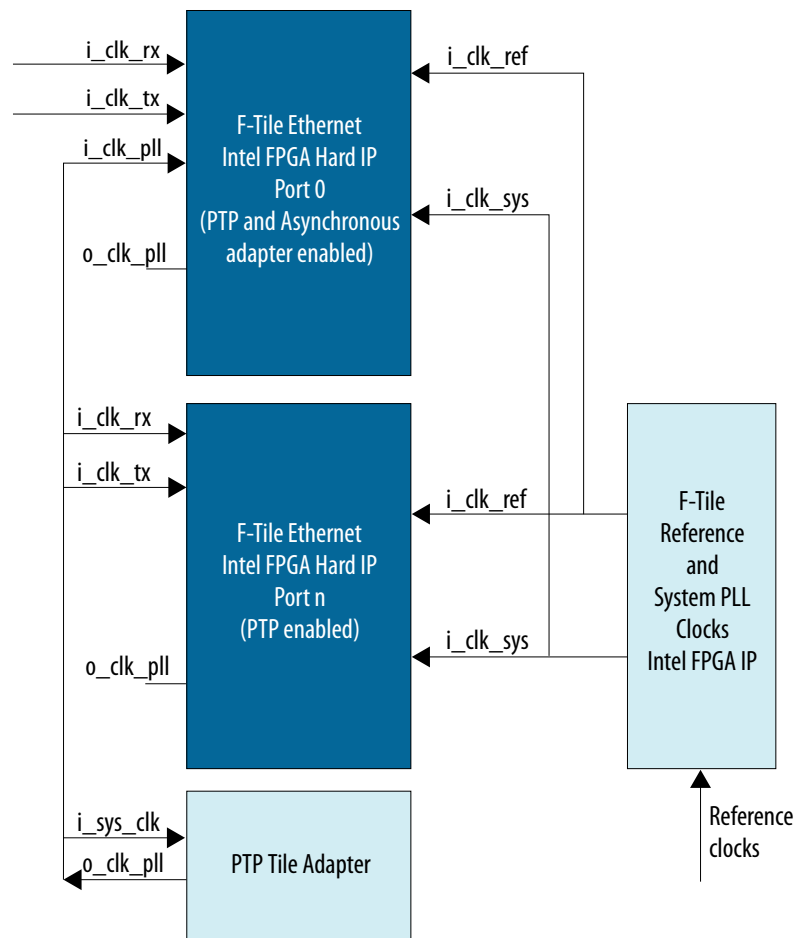
5.4. Clock Connections in PTP-Based Synchronous and Asynchronous Operation

When you enable **Enable IEEE 1588 PTP** in your IP, all Ethernet IP cores must be clocked by the same system clock source `o_clk_pll` of the PTP tile adapter. The required input clock source is system clock source divided by 2, with a minimum frequency of 402.83 MHz.

When you enable **Enable asynchronous adapter clocks** along with the **Enable IEEE 1588 PTP** in your IP, the `i_clk_pll` signal must connect to the same system clock source. The `i_clk_tx` and `i_clk_rx` input clock signals can be asynchronous from each other and from `o_clk_pll` clock provided that the clocks are fast enough to ensure the IP core channel processes all data.

The PTP tile adapter's `i_sys_clk` clock is sourced from its own `o_clk_pll` clock.

Figure 24. Clock Connections in PTP-Based Synchronous and Asynchronous Operation



5.5. Clock Connections in Synchronous Ethernet Operation

When you enable the Synchronous Ethernet (SyncE) operation, two or more channels can share the off-chip cleanup PLL clock output.

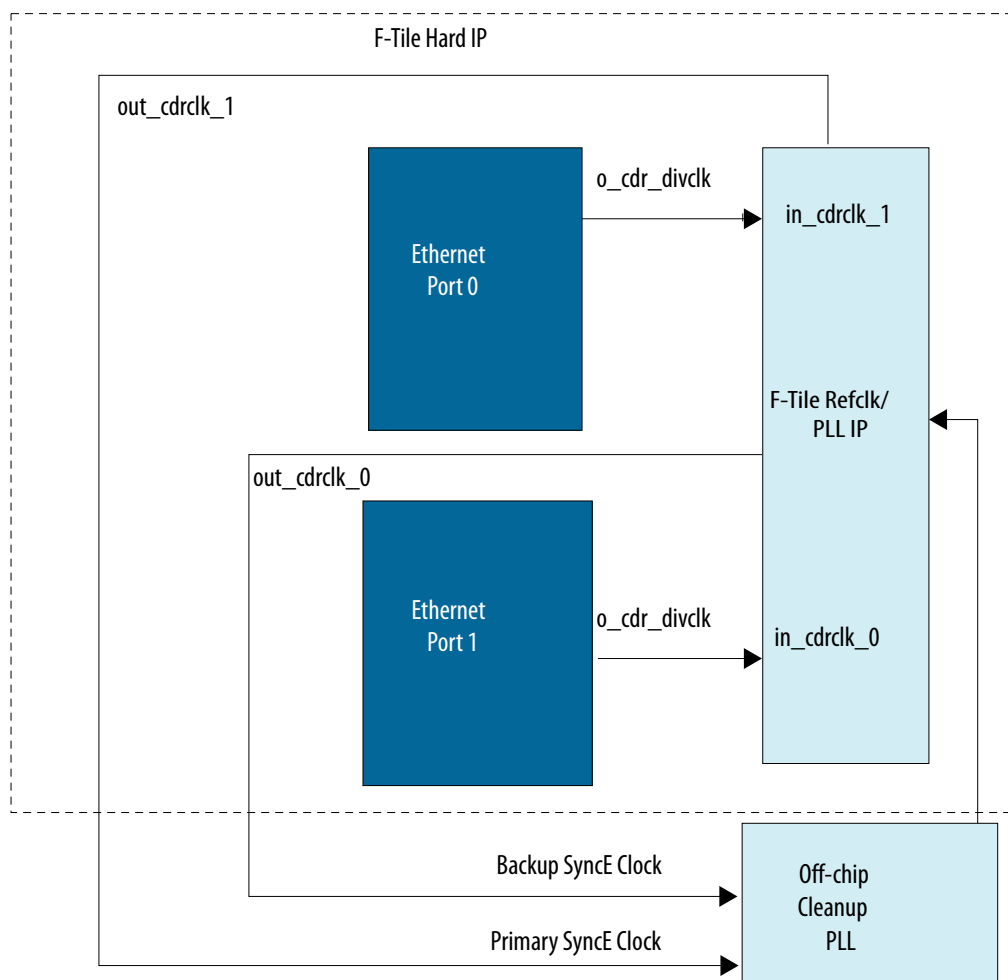
The Synchronous Ethernet standard, described in the ITU-T G.8261, G.8262, and G.8264 recommendations, requires that the TX clock be filtered to maintain synchronization with the RX reference clock through a sequence of nodes. The expected usage is that user logic drives the transceiver reference clocks with a filtered version of the RX recovered clock signal, to ensure the receive and transmit functions remain synchronized. In this usage model, a design component outside the IP core performs the filtering off chip.

In the diagram below, the recovered clock outputs from the IPs can be connected to the off-chip cleanup PLL using tile PIN. In F-tile, the recovered clock from the FGT PMA can be driven to dedicated clock output pins on the tile itself. Each F-tile has only two such clock outputs: FGT QUADs 2 and 3. Each has one dedicated clock output pin (Refclk8/9).

The primary and backup cleanup clocks come from recovered clock output pins from a pair of ports connected to remote stations on the same SyncE network, with the transceiver reference clock is sourced from the output of the cleanup PLL.

In the diagram below, `out_cdr_clk0` is the clock that goes to off-chip cleanup PLL.

Figure 25. Clock Connection of Sync-E clock through CDR clock out pin



Note: You must set the **Custom Cadence** mode to match the PPM difference between clocks when the Ethernet IP system clock is derived from a different reference clock than the transceiver clock.

Implementation of Synchronous Ethernet (SyncE) Operation

To enable the recovered clock output from F-tile, select the **Enable dedicated CDR Clock Output** in the IP parameter editor as shown below.

Figure 26. Enable dedicated CDR Clock Output IP Parameter Editor

F-Tile Ethernet Intel FPGA Hard IP
eth_f

General Options

PMA type: FGT

☐ Enable FHT pre-encoder

Ethernet mode: 10GE-1

Client interface: MAC segmented

FEC mode: None

PMA reference frequency: 156.250000 MHz

System PLL frequency: 805.664062 MHz

Custom system PLL frequency: 805.664062 MHz

☐ External Custom Cadence Controller

☐ Include Deterministic Latency interface

☐ Include 32bit soft CWBIN counters

Reconfig clock frequency: 100.0 MHz

☒ Enable dedicated CDR clock output

The output frequency is equal to the nominal incoming `refclk` divided by the pre-divider on the RX path: $o_cdr_divclk = refclk / N$.

To retrieve the N divider value, follow the steps below:

1. Compile the design.
2. Open **Compilation Report**, then go to **Logic Generation Tool > IP Parameter Settings Report**.
3. Search **cdr_n_counter**.
4. [optional] You can also search **cdr_f_ref_hz** to double confirm the input reference clock frequency.

Figure 27. Retrieve the N divider Value

Table of Contents

- Flow Summary
- Flow Settings
- Flow Non-Default Global Settings
- Flow Elapsed Time
- Flow OS Summary
- Flow Log
- Design Analysis
- Logic Generation Tool
 - Settings
 - IP Parameter Settings Report**

Logic Generation Tool IP Parameter Settings Report

Search: cdr_n_counter

	Name	Value
1	XGPON directphy_f_0	
1	-- BB_F_UX_RX	XGPON directphy_f...x_ux_x_bb_f_ux_rx
1	-- cdr_n_counter	6'b000110

The table below shows the recovered clock frequencies with respect to input reference clock.

Table 29. Recovered Clock Frequency

NRZ/PAM4	Input Refclk (MHz)	N divider	Output Recovered Clk (MHz)
NRZ	156.25	4	39.0625
	156.25	6	26.0417
	312.5	8	39.0625
	312.5	12	26.0417
	322.265625	12	26.8554688
PAM4	156.25	6	26.0417
	312.5	12	

Note: Make sure the DUT top signal o_cdr_divclk is connected to the system PLL IP.

To connect to the system PLL, select **Enable FGT CDR Output** in the IP parameter editor, as shown in the figure below.

Figure 28. System PLL Clocks Intel FPGA IP

Parameters

System: suja223 Path: systemclk_f_0

F-Tile Reference and System PLL Clocks Intel FPGA IP

systemclk_f

System PLL

System PLL #0 System PLL #1 System PLL #2

Mode of System PLL: ETHERNET_FREQ_805_156

Refclk source: RefClk #0

Output frequency: 805.6640625 MHz

☒ Refclk is available at power-on

FHT Common PLL

Controller source: Auto

FHT Common PLL A FHT Common PLL B

☐ Enable FHT Common PLL A

FHT refclk source: FHT RefClk #0

RefClk

FGT/System PLL Refclk FGT CDR Clock-out FHT Refclk

☒ Enable FGT CDR Output #0

☐ Enable FGT CDR Output #1

Note: In the IP Parameter Editor, you can **Enable the dedicated CDR clock output** for multiple IPs. However, if you want to enable two dedicated CDR clock outputs from different IPs, **Enable FGT CDR Output #0** and **Enable FGT CDR Output #1** must be enabled in the system PLL.

Example Design Generation of Synchronous Ethernet Operation

1. To generate the example design, after selecting the **Enable dedicated CDR Clock Output** in the IP parameter editor, go to **Example Design** tab.
2. In the **Select Design** parameter under **Available example Designs**, you can select the following options:

Figure 29. Select Designs Example Design

- 2x25G ED for SyncE
- Single instance of IP core

Note: The multi instance of IP core is not supported when **Enable dedicated CDR Clock Output** is set to on.

5.6. Custom Cadence

You can use custom cadence to ensure that the TX PMA interface FIFO doesn't overflow due to the over clocking of the datapath. You must use the custom cadence in the following 2 scenarios:

- When the reference clock for TX PMA and the reference clock for system PLL are different. Although, system PLL frequency is configured to be same as PMA parallel frequency, there might be PPM differences between TX PMA parallel clock frequency and system PLL frequency.
- When the system PLL frequency is more than the PMA parallel clock frequency. This non-standard system PLL frequency may require when several interfaces share the same system clock (some interfaces need to run at over-clocked system PLL frequency greater than PMA parallel clock frequency). This scenario may occur due to a limited number of system PLLs or when dynamic reconfiguration is enabled.

If your interface requires custom cadence, you must enable custom cadence. To enable custom cadence, select **System PLL frequency** IP parameter to **Custom** and input your required custom system PLL frequency.

When **System PLL frequency** IP parameter is set to **Custom**, the IP internally includes a custom cadence controller.

If you require an external custom cadence controller instead, you must enable it by selecting **External Custom Cadence Controller** IP parameter. When selected, this option enables the external custom cadence controller and `i_custom_cadence` input port becomes available in the IP port list. You must drive the `i_custom_cadence` input port producing a steady data valid cadence.

In multiple IP instance designs, you may use the external custom cadence controller option to share the logic with multiple IPs and reduce the resource utilization.

Table 30. Custom Cadence Use Cases

Abbreviations:

- CL74: IEEE 802.3 BASE-R Firecode (CL74)
- CL91: IEEE 802.3 RS(528,514) (CL91)
- CL134: IEEE 802.3 RS(544,514) (CL134)
- ETC: Ethernet Technology Consortium ETC RS(272, 258)
- MAC AvST: MAC Avalon ST
- MAC Seg: MAC Segmented

Configuration	System PLL frequency (in MHz)	Use Case	Custom Cadence Controller Selection
NRZ modes <ul style="list-style-type: none"> • with FEC mode: <ul style="list-style-type: none"> — None — CL74 — CL91 • with client interface: <ul style="list-style-type: none"> — MAC Seg — MAC AvST 	830.078125	<ul style="list-style-type: none"> • If system PLL is shared with FEC mode configuration CL134 or ETC and system PLL reference clock and PMA reference clock sources are same and use the same clock frequency (0 ppm). • If system PLL reference clock and PMA reference clock sources are different, enable Custom system PLL Frequency and set the input to 830.078125 MHz. 	The IP enables internal custom cadence controller logic when 830.078125 MHz is selected instead of typical 805.6640625 MHz.
PCS OTN FlexE	805.6640625 or 830.078125	<ul style="list-style-type: none"> • If system PLL reference clock and PMA reference clock sources are same and use the same clock frequency (0 ppm). • If system PLL reference clock and PMA reference clock sources are different, enable Custom system PLL Frequency and set the input to 805.6640625 or 830.078125 MHz. 	The IP enables internal custom cadence controller logic when 830.078125 or 805.6640625 MHz.
Any variant with client interface: <ul style="list-style-type: none"> • MAC Seg • MAC AvST • MII PCS only • PCS66 OTN • PCS66 FlexE 	Custom PLL	<ul style="list-style-type: none"> • Requires a non-standard system PLL frequency, for example 900 MHz. • System PLL reference clock and PMA reference clock sources are different. 	The IP enables internal custom cadence controller when the external custom cadence controller is disabled. When the external custom cadence controller is enabled, you must drive the <code>i_custom_cadence</code> port.

Table 31. Various Custom Cadence Scenarios for 25GE-1 with IEEE 802.3 RS(528,514) (CL91) FEC Mode

The example assumes system PLL frequency of 805.6640625 MHz.

Client Interface	System PLL Frequency (in MHz)	PMA/System PLL Reference Clock Source	Custom Cadence Required	Custom Cadence Controller	Client Interface Frequency	FIFO Requirement
MAC Avalon ST	805.6640625	Same	No	—	Same as system PLL frequency/2 (402.83203125 MHz)	No
MAC Avalon ST	805.6640625	Same	No	—	Different as system PLL frequency/2 (For example, 390.625 MHz)	Enable asynchronous adapter clocks parameter is On
MAC Avalon ST	Custom (For example, 805.6640625)	Different (PPM)	Yes	Internal or External	Same as system PLL frequency/2 (For example, 402.83203125 MHz)	No
MAC Avalon ST	Custom (For example, 805.6640625)	Different (PPM)	Yes	Internal or External	Different as system PLL frequency/2 (For example, 390.625 MHz)	Enable asynchronous adapter clocks parameter is On ⁽¹⁸⁾
MAC Avalon ST	830.078125	Same	Yes	Internal	Same as system PLL frequency/2 (415.078125 MHz)	No
MAC Avalon ST	830.078125	Same	Yes	Internal	Different as system PLL frequency/2 (For example, 390.625 MHz/402.83203125 MHz)	Enable asynchronous adapter clocks parameter is On
MAC Avalon ST	Custom (For example, 830.078125 or 900)	Same or Different (PPM)	Yes	Internal or External	Same as system PLL frequency/2 (For example, 415.078125 MHz/450 MHz)	No
MAC Avalon ST	Custom (For example, 830.078125 or 900)	Same or Different (PPM)	Yes	Internal or External	Different as system PLL frequency/2 (For example, 390.625 MHz/402.83203125 MHz)	Enable asynchronous adapter clocks parameter is On ⁽¹⁸⁾
MAC segmented	805.6640625	Same	No	—	Same as system PLL frequency/2 (402.83203125 MHz)	No
MAC segmented	805.6640625	Same	No	—	Different as system PLL frequency/2 (For example, 390.625 MHz)	Client Custom FIFO instantiated outside of the Ethernet IP
continued...						

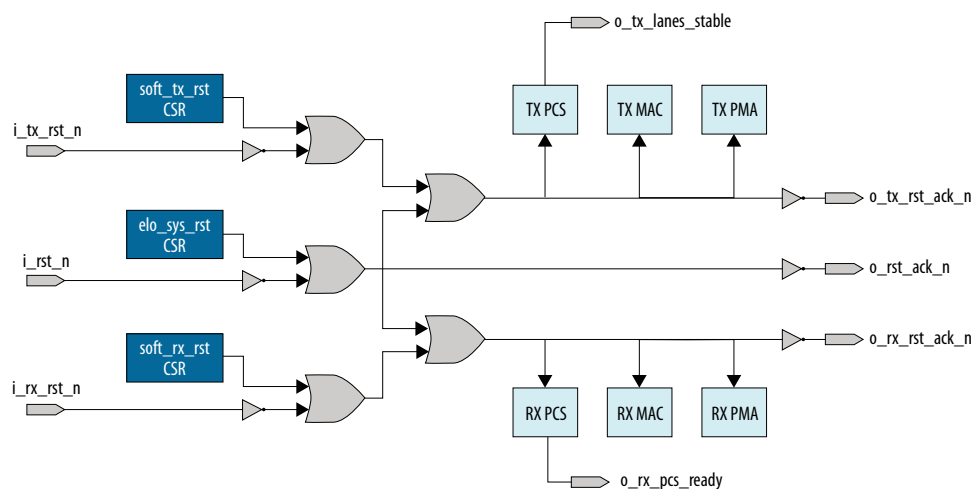
⁽¹⁸⁾ When using multiple IP instances, if both **Enable asynchronous adapter clocks** and **External Custom Cadence Controller** are enabled, you must generate each IP's `i_custom_cadence` signal using corresponding `o_clk_pll` clock.

Client Interface	System PLL Frequency (in MHz)	PMA/System PLL Reference Clock Source	Custom Cadence Required	Custom Cadence Controller	Client Interface Frequency	FIFO Requirement
MAC segmented	Custom (For example, 805.6640625)	Different (PPM)	Yes	Internal or External	Same as system PLL frequency/2 (For example, 402.83203125 MHz)	No
MAC segmented	Custom (For example, 805.6640625)	Different (PPM)	Yes	Internal or External	Different as system PLL frequency/2 (For example, 390.625 MHz)	Client Custom FIFO instantiated outside of the Ethernet IP
MAC segmented	830.078125	Same	Yes	Internal	Same as system PLL frequency/2 (415.078125 MHz)	No
MAC segmented	830.078125	Same	Yes	Internal	Different as system PLL frequency/2 (For example, 390.625 MHz/402.83203125 MHz)	Client Custom FIFO instantiated outside of the Ethernet IP
MAC segmented	Custom (For example, 830.078125 or 900)	Same or Different (PPM)	Yes	Internal or External	Same as system PLL frequency/2 (For example, 415.078125 MHz/450 MHz)	No
MAC segmented	Custom (For example, 830.078125 or 900)	Same or Different (PPM)	Yes	Internal or External	Different as system PLL frequency/2 (For example, 390.625 MHz/402.83203125 MHz)	Client Custom FIFO instantiated outside of the Ethernet IP

6. Resets

Ethernet reset ports control for the F-Tile Ethernet Intel FPGA Hard IP consists of four main reset ports and five soft datapath and statistics register resets.

Figure 30. Conceptual Overview of General IP Core Reset Logic



The general reset signals reset the following functions:

- `i_reconfig_reset`: Resets the entire reconfiguration clock domain, including the soft CSR registers and Avalon memory-mapped interface.
- `i_tx_rst_n`: Resets the TX datapath, TX transceivers, and TX EMIB adapters.
- `i_rx_rst_n`: Resets the RX datapath, RX transceivers, and RX EMIB adapters.

Note: When RX MAC is in reset, TX MAC is only able to transmit idles or remote fault indications if link fault signaling is enabled. You are unable to transmit the data. The `o_tx_ready/o_tx_mac_ready` remains low.

- `i_rst_n`: Resets TX/RX datapaths, transceivers, and EMIB adapters.

Note: The system PLL cannot be reset.

Table 32. Reset Signals Functions

In this table, a tick (✓) represents the block is reset by the specified reset signal.

Important: The F-Tile Ethernet Intel FPGA Hard IP does not support clearing hard CSR registers back to the default values.

Reset Signal	PHY		Datapath				Stats		Soft CSRs
	TX	RX	PCS TX	PCS RX	MAC TX	MAC RX	MAC TX	MAC RX	
Port Resets									
i_rst_n	✓	✓	✓	✓	✓	✓	✓	✓	
i_tx_rst_n	✓		✓		✓		✓		
i_rx_rst_n		✓		✓		✓		✓	
i_reconfig_reset									✓
Register Resets									
eio_sys_rst	✓	✓	✓	✓	✓	✓	✓	✓	
soft_tx_rst	✓		✓		✓		✓		
soft_rx_rst		✓		✓		✓		✓	
rst_tx_stats							✓		
rst_rx_stats								✓	

6.1. Reset Signals

The IP core has four soft reset inputs. These resets are asynchronous and are internally synchronized.

Table 33. Reset Signals

All specified resets are asynchronous.

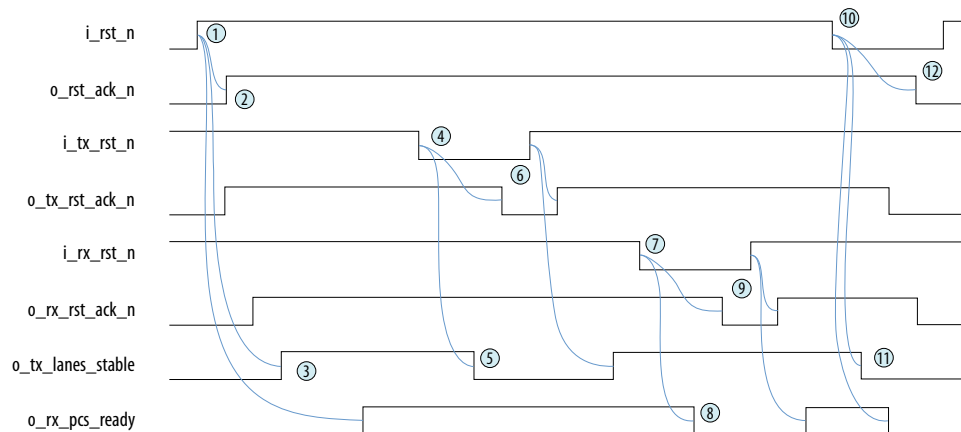
Signal	Description
Input Signals	
i_rst_n	Active-low reset asynchronous signal. Do not deassert until the o_rst_ack_n deasserts. <ul style="list-style-type: none"> Resets the TX interface, including the TX PCS and TX MAC. Resets the RX interface, including the RX PCS and RX MAC. Resets the TX PMA and TX EMIB. Resets the RX PMA and RX EMIB. This reset leads to assertion of the o_rst_ack_n output signal.
i_tx_rst_n	Active-low reset asynchronous signal. Resets the entire TX datapath, including the TX PCS, TX MAC, TX PMA, and TX EMIB. Do not deassert until the o_rst_ack_n asserts.
i_rx_rst_n	Active-low reset asynchronous signal. Resets the entire RX datapath, including the RX PCS, RX MAC, RX PMA, and RX EMIB. Do not deassert until the o_rst_ack_n asserts.
i_reconfig_reset	Active-high reconfiguration reset signal. Reset the entire reconfiguration clock domain, including the soft registers (CSRs). You must assert this reset after power-on or during the configuration. The i_reconfig_clk must be stable before deasserting this reset.
<i>Note:</i>	
continued...	

Signal	Description
When you enabled the AN/LT feature in the F-Tile Ethernet Intel FPGA Hard IP, you must set the <code>kr_pause</code> AN/LT CSR register bit to 1 for a subsequent assertion of <code>i_reconfig_reset</code> after power-on and wait for <code>kr_paused</code> to be set to 1. Toggling <code>i_reconfig_reset</code> is now safe.	
Output Signals	
<code>o_rst_ack_n</code>	Active-low asynchronous acknowledgement signal for the <code>i_rst_n</code> reset. Do not deassert <code>i_rst_n</code> reset until the <code>o_rst_ack_n</code> asserts.
<code>o_tx_rst_ack_n</code>	Active-low asynchronous acknowledgement signal for the <code>i_tx_rst_n</code> reset. Do not deassert <code>i_tx_rst_n</code> reset until the <code>o_tx_rst_ack_n</code> asserts.
<code>o_rx_rst_ack_n</code>	Active-low asynchronous acknowledgement signal for the <code>i_rx_rst_n</code> reset. Do not deassert <code>i_rx_rst_n</code> reset until the <code>o_rx_rst_ack_n</code> asserts.
Status Signals	
<code>o_tx_lanes_stable</code>	Active-high asynchronous status signal for the TX datapath. <ul style="list-style-type: none"> Asserts when the TX datapath is ready to send data. Deasserts when <code>i_tx_rst_n</code>/<code>i_rst_n</code> signal asserts or during the auto-negotiation and link training operation.
<code>o_rx_pcs_ready</code>	Active-high asynchronous status signal for the RX datapath. <ul style="list-style-type: none"> Asserts when the RX datapath is ready to receive data. Deasserts when <code>i_rx_rst_n</code>/<code>i_rst_n</code> signal asserts or during the auto-negotiation and link training operation.

6.2. Reset Sequence

This section shows the sequencing of signals for several typical reset scenarios.

Figure 31. Reset Sequence



The following steps describe IP core reset sequence as shown in the waveform.

1. Drive the `i_rst_n` reset signal high while `i_tx_rst_n` and `i_rx_rst_n` reset signals are already deasserted.
2. The `o_rst_ack_n` reset signal deasserts. This indicates that the IP core is no longer in the full reset.

Note: This step doesn't indicate that the IP core is in fully functional state.

Note: The `o_tx_rst_ack_n` and `o_rx_rst_ack_n` reset signals also deassert. The exact sequence and timing is not guaranteed.

3. The IP core is fully out of reset. Assert `o_tx_lanes_stable` and `o_rx_pcs_ready` to indicate that the TX and RX datapaths are ready for use.
4. Assert the `i_tx_rst_n` reset signal.
5. The `o_tx_lanes_stable` signal deasserts to indicate that the TX datapath is no longer operational.
6. The `o_tx_rst_ack_n` signal asserts indicating that the TX datapath is fully in reset. Then, deassert the `i_tx_rst_n` signal to bring the TX datapath out of the reset.
7. Assert the `i_rx_rst_n` reset signal.
8. The `o_rx_pcs_ready` signal deasserts to indicate that the RX datapath is no longer operational.
9. The `o_rx_rst_ack_n` signal asserts indicating that the RX datapath is fully in reset. Then, deassert the `i_rx_rst_n` signal to bring the RX datapath out of the reset.
10. Assert the `i_rst_n` reset signal.
11. The `o_tx_lanes_stable` and `o_rx_pcs_ready` signals deassert to indicate that TX and RX datapath are no longer operational.
12. The `o_rst_ack_n` signals assert to indicate the IP core is fully in reset. To bring the IP core out of the reset, deassert the `i_rst_n` reset signal.

System Considerations

- During the startup state, the system does not require asserting `i_rst_n`, `i_tx_rst_n`, and `i_rx_rst_n` reset signals.
- After power on, configuration, partial reconfiguration, you must assert `i_reconfig_reset` signal at least once to ensure the soft CSR registers contains the reset values.
- For external custom cadence, the custom cadence signal must be toggling before `tx_lanes_stable` signal comes up.
- Similarly for PCS and PCS66 interfaces, alignment marker insertion must occur at the proper interval before `tx_lanes_stable` comes up.
- During the reset, hold the `i_reconfig_reset` signal asserted for several valid reconfiguration clock cycles to ensure the Avalon memory-mapped interface and soft CSRs are fully reset.
- Access to any Avalon memory-mapped interface is available while the `i_reconfig_reset` signal is low.

7. Interface Overview

All input signal names begin with `i_` and all output signal names begin with `o_`.

7.1. Status Interface

The F-Tile Ethernet Intel FPGA Hard IP core provides a handful of status signals to support visibility into the actions of the IP core and the stability of IP core output clocks.

Table 34. Status Signals

All of the status signals except the `i_stats_snapshot` signal are asynchronous.

Signal	Description
<code>o_rx_block_lock</code>	In non-FEC and Firecode FEC variant, asserted when the IP core completes 66-bit block boundary alignment on all PCS lanes. Otherwise, asserted when the IP core completes the codeword alignment on all FEC lanes.
<code>o_rx_am_lock</code>	Asserted when the RX PCS completes detection of alignment markers and deskew of the PCS lanes. Not supported for 10G/25G variants.
<code>o_local_fault_status</code>	Asserted when the RX MAC detects a local fault: the RX PCS detected a problem that prevents it from receiving data. This signal is functional only if you set the Client Interface parameter to the value of MAC segmented or MAC Avalon ST in the parameter editor.
<code>o_remote_fault_status</code>	Asserted when the RX MAC detects a remote fault: the remote link partner sent remote fault ordered sets indicating that it is unable to receive data. This signal is functional only if you set the Client Interface parameter to the value of MAC segmented or MAC Avalon ST in the parameter editor.
<code>i_stats_snapshot</code>	Directs the IP core to record a snapshot of the current state of the statistics registers. Assert this signal to perform the function of both the TX and RX statistics register shadow request fields at the same time, or to perform that function for multiple instances of the IP core simultaneously. This signal is synchronous with the <code>i_clk_tx</code> clock.
<code>o_rx_hi_ber</code>	Asserted to indicate the RX PCS is in a HI BER state according to Figure 82-15 in the <i>IEEE 802.3-2015 Standard</i> . The IP core uses this signal in auto negotiation and link training.
<code>o_rx_pcs_fully_aligned</code>	Asserts when RX PCS is ready to receive data.

Figure 32. Status Interface Behavior during Link Startup with Bidirectional Link Fault

The waveform displays the status signal behavior in the IP core at the startup.

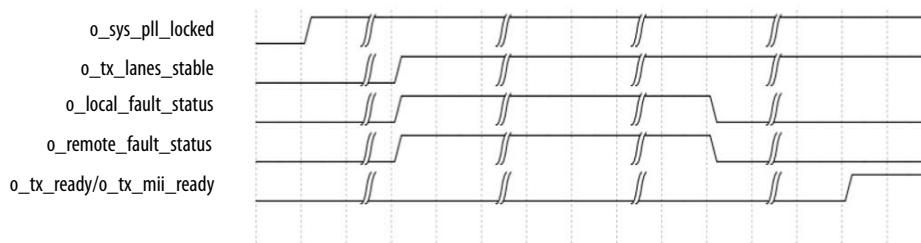


Figure 33. Status Interface Behavior during Link Startup with Unidirectional or Link Fault Disable

The waveform displays status signals behavior in the IP core at the startup.

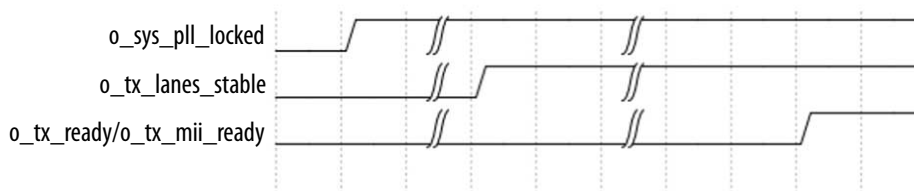
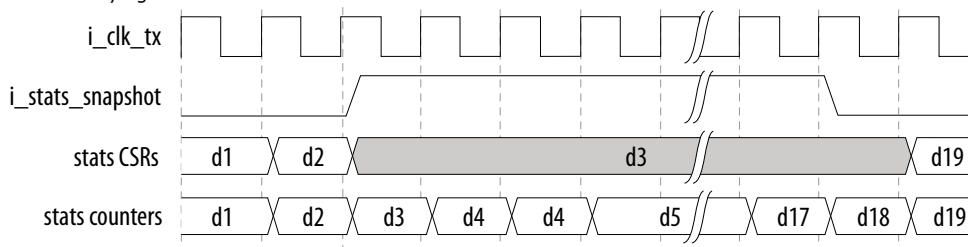


Figure 34. Status Interface Behavior during Freezing i_stats_snapshot

The waveform displays an event when i_stats_snapshot signal is used to freeze the statistics CSRs. Note that the underlying counters continue to track the IP core events.



Related Information

[Reset Signals](#) on page 95

7.2. TX MAC Avalon ST Client Interface

The F-Tile Ethernet Intel FPGA Hard IP TX client interface in MAC+PCS variations employs the Avalon-ST protocol. The Avalon ST protocol is a synchronous point-to-point, unidirectional interface that connects the producer of a data stream (source) to a consumer of data (sink). The key properties of this interface include:

- Start of packet (SOP) and end of packet (EOP) signals delimit frame transfers.
- The SOP must always be in the MSB, simplifying the interpretation and processing of incoming data.
- A valid signal qualifies signals from source to sink.
- The sink applies backpressure to the source by using the ready signal. The source typically responds to the deassertion of the ready signal from the sink by driving the same data until the sink can accept it. The **Ready latency** defines the relationship between assertion and deassertion of the ready signal, and cycles which are considered to be *ready* for data transfer.

The client acts as a source and the TX MAC acts as a sink in the transmit direction.

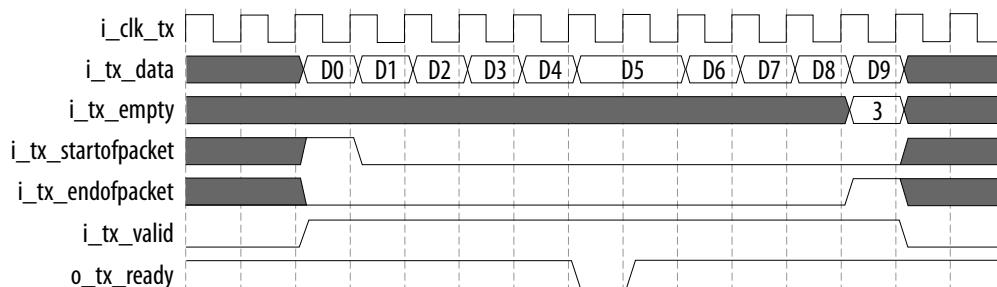
Table 35. Signals of the Avalon Streaming TX Client Interface

All interface signals are clocked by the TX clock. The signal names are standard Avalon streaming interface signals with slight differences to indicate the variations. For example:

Signal Name	Width	Description
i_tx_data[511:0] i_tx_data[127:0] i_tx_data[63:0]	512 bits (100GE) 128 bits (50GE/40GE) 64 bits (25GE/10GE)	Input data to the MAC when the rate is 10GE/25/40GE/50GE/100GE. Bit 0 is the LSB.
i_tx_valid	1 bit	When asserted, the TX data signal is valid. This signal must be continuously asserted between the assertions of the start of packet and end of packet signals for the same packet.
i_tx_startofpacket	1 bit	Start of Packet (SOP). When asserted, indicates that the TX data holds the first clock cycle of data in a packet (start of packet). Assert for only a single clock cycle for each packet. When the SOP signal is asserted, the MSB of the TX data drives the start of packet.
i_tx_endofpacket	1 bit	End of Packet (EOP). When asserted, indicates that the TX data holds the final clock cycle of data in a packet (end of packet). Assert for only a single clock cycle for each packet. For some legitimate packets, the SOP and EOP signals are asserted on the same clock cycle.
i_tx_empty[5:0] i_tx_empty[3:0] i_tx_empty[2:0]	6 bits (100GE) 4 bits (50GE/40GE) 3 bits (10GE/25GE)	Indicates the number of empty bytes on the TX data when the EOP signal is asserted.
o_tx_ready	1 bit	The ready signal indicates the MAC is ready to receive data in normal operational mode.
i_tx_preamble[63:0]	64 bits	Writes the preamble value of a TX frame. This signal is valid when the i_tx_valid and the i_tx_startofpacket signals are asserted. This signal is only available when you turn on Preamble Passthrough in the parameter editor for 40GE/50GE channels.
continued...		

Signal Name	Width	Description
i_tx_error	1 bit	When asserted in an EOP cycle (while the EOP signal is asserted), directs the IP core to insert an error in the packet before sending it on the Ethernet link.
i_tx_skip_crc	1 bit	<p>Specifies how the TX MAC should process the current TX MAC client interface packet. Use this signal to temporarily turn off source insertion for a specific packet and to override the default behaviors of padding to minimum packet size and inserting CRC.</p> <p>If this signal is asserted, directs the TX MAC to not insert CRC, not add padding bytes, and not implement source address insertion. You can use this signal to indicate the data on the TX data signal includes CRC, padding bytes (if relevant), and the correct source address.</p> <p>If this signal is not asserted, and source address insertion is enabled, directs the TX MAC to overwrite the source address. The MAC copies the new source address from the TXMAC_SADDR register.</p> <p>If this signal is not asserted, whether or not source address insertion is enabled, the TX MAC inserts padding bytes if needed and inserts CRC in the packet.</p> <p>The client must maintain the same value on this signal for the duration of the packet (from the cycle in which it asserts the SOP signal through the cycle in which it asserts the EOP signal, inclusive).</p>
o_tx_serial[3:0]	4 bits	Output TX serial transceiver signal indicates how TX serial pins on the board are connected to transceiver RX serial pins for lane reverse ordering.

Figure 35. Transmitting Data Using the TX MAC Avalon ST Client Interface



The figure above shows how to transmit data using the TX MAC Avalon ST client interface. The interface complies with the Avalon streaming interface specification.

- Data valid (`i_tx_valid`) must be held high from the start to end of a packet, and must be low outside of a packet.
- Packets always start on the MSB of the byte of `i_tx_data` (SOP aligned).
- You can set the **Ready latency** through the parameter editor.
 - When `o_tx_ready` deasserts, `i_tx_data` must be paused for as many cycles as `o_tx_ready` is deasserted, starting **Ready latency** cycles later. In this example, **Ready latency** is 1. So the cycle after `o_tx_ready` deasserts for 1 cycle, `i_tx_data` is paused for 1 cycle.
- When the frame ends, `i_tx_empty` is set to the number of unused bytes in `i_tx_data`, starting from the LSB (byte 0).
 - In this example, `i_tx_data` on the last cycle of the packet has 3 empty bytes.
 - The minimum number of bytes on the last cycle is 1.

7.2.1. TX MAC Avalon ST Client Interface with Disabled Preamble Passthrough

Figure 36. Fields and Frame Boundaries in an Ethernet Packet

When you turn off **Preamble Passthrough** in the parameter editor, `i_tx_data` must be written as shown below for the first cycle of data presented to the MAC.

Note: For 10GE/25GE channels, multiple cycles are required to write the header data.

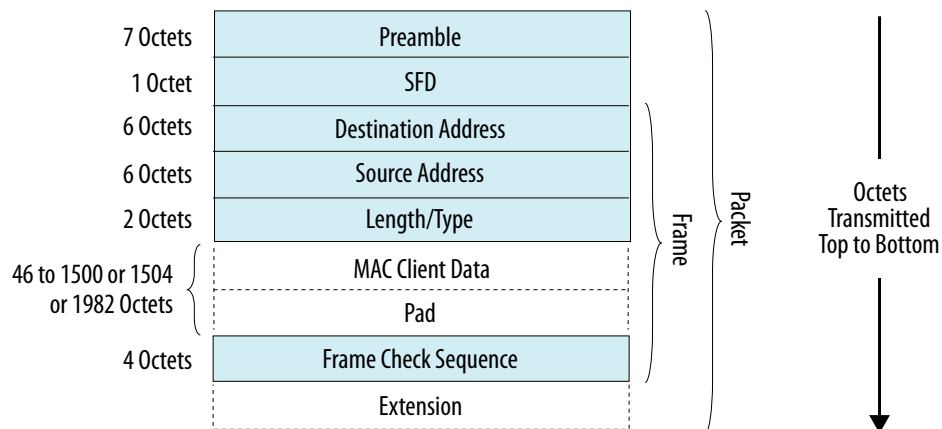


Table 36. TX MAC Field Positions in i_tx_data with Preamble Passthrough Disabled

Attention: 10GE/25GE requires multiple transfer cycle for header data.

The (') symbol in the **10GE/25GE i_tx_data** column represents transfer on the subsequent cycle.

100GE i_tx_data	40GE/50GE i_tx_data	10GE/25GE i_tx_data	MAC Field	Note
[511:504]	[127:120]	[63:56]'	Dest Addr[47:40]	The first octet of the Destination Address, follows Start Frame Delimiter (SFD).
[503:496]	[119:112]	[55:48]'	Dest Addr[39:32]	
[495:488]	[111:104]	[47:40]'	Dest Addr[31:24]	
[495:480]	[103:96]	[39:32]'	Dest Addr[23:16]	
[479:472]	[95:88]	[31:24]'	Dest Addr[15:8]	
[471:464]	[87:80]	[23:16]'	Dest Addr[7:0]	
[463:456]	[79:72]	[15:8]'	Src Addr[47:40]	When you turn on Source Address Insertion , contents are replaced by txmac_saddr unless i_tx_skip_crc is high.
[455:448]	[71:64]	[7:0]'	Src Addr[39:32]	
[447:440]	[63:56]	[63:56]	Src Addr[31:24]	
[439:432]	[55:48]	[55:48]	Src Addr[23:16]	
[431:424]	[39:32]	[47:40]	Src Addr[15:8]	
[423:416]	[39:32]	[39:32]	Src Addr[7:0]	
[415:408]	[31:24]	[31:24]	Length/Type[15:8]	
[407:400]	[23:16]	[23:16]	Length/Type[7:0]	
[399:0]	[15:0]	[15:0]	...	

Note:

- In the table above, the byte order on the bus flows from MSB to LSB, the first byte of the MAC destination address is the MSB. The MAC considers this to be the first byte after the Start Frame Delimiter (SFD).
- The bit numbered 0 is always the least significant bit of each byte.
 - For example, on 100GE interface, **i_tx_data[504]** transmits after the SFD, and corresponds to the Ethernet destination address unicast/multicast bit.

7.2.2. TX MAC Avalon ST Client Interface with Enabled Preamble Passthrough

When you turn on **Preamble Passthrough** in the parameter editor, you must provide 8 preamble bytes to the TX MAC interface.

The table below describes TX MAC field positions with enabled Preamble Passthrough parameter.

- MII Start of Packet control byte always replaces the first preamble byte.
- Bits[55:8] are the preamble bits, typically set to the 0x55 value.
- Bits[7:0] is the last preamble byte. In a standard preamble, it is set to the Start Frame Delimiter 0xD5 value.

Table 37. TX MAC Field Positions in i_tx_data with Preamble Passthrough Enabled for 10GE/25GE/100GE Ports

Attention: 10GE/25GE requires multiple transfer cycle for header data.

The (') symbol in the **10GE/25GE i_tx_data** column represents transfer on the subsequent cycle.

The (") symbol in the **10GE/25GE i_tx_data** column represents transfer on the 2nd subsequent cycle.

100GE i_tx_data	10GE/25GE i_tx_data	MAC Field	Note
[511:504]	[63:56]"	Custom Preamble [63:56]	MII SOP control channel replaces it.
[503:496]	[55:48]"	Custom Preamble [55:48]	0x55
[495:488]	[47:40]"	Custom Preamble [47:40]	0x55
[495:480]	[39:32]"	Custom Preamble [39:32]	0x55
[479:472]	[31:24]"	Custom Preamble [31:24]	0x55
[471:464]	[23:16]"	Custom Preamble [23:16]	0x55
[463:456]	[15:8]"	Custom Preamble [15:8]	0x55
[455:448]	[7:0]"	Custom Preamble [7:0]	0xD5 (SFD)
[447:440]	[63:56]'	Dest Addr[47:40]	
[439:432]	[55:48]'	Dest Addr[39:32]	
[431:424]	[47:40]'	Dest Addr[31:24]	
[423:416]	[39:32]'	Dest Addr[23:16]	
[415:408]	[31:24]'	Dest Addr[15:8]	
[407:400]	[23:16]'	Dest Addr[7:0]	
[399:392]	[15:8]'	Src Addr[47:40]	When you turn on Source Address Insertion , contents are replaced by txmac_saddr unless i_tx_skip_crc is high.
[391:384]	[7:0]'	Src Addr[39:32]	
[383:376]	[63:56]	Src Addr[31:24]	
[375:368]	[55:48]	Src Addr[23:16]	
[367:360]	[47:40]	Src Addr[15:8]	
[359:352]	[39:32]	Src Addr[7:0]	
[351:344]	[31:24]	Length/Type[15:8]	
[343:336]	[23:16]	Length/Type[7:0]	
[335:0]	[15:0]	...	

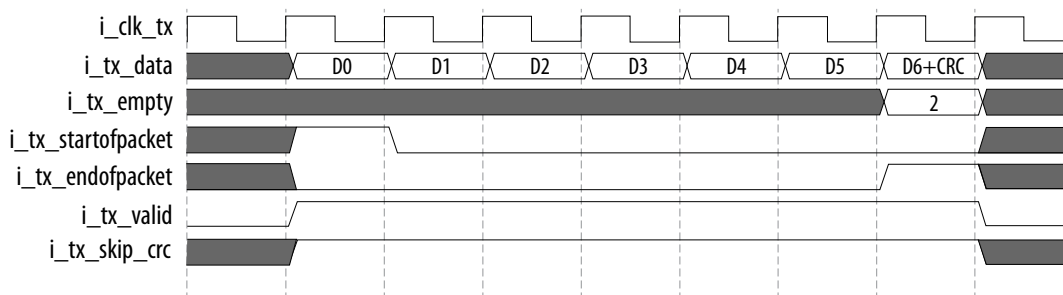
Table 38. TX MAC Field Positions in i_tx_data with Preamble Passthrough Enabled for 40GE/50GE Ports

40GE/50GE i_tx_preamble/i_tx_data	MAC Field	Note
TX Preamble [63:56]	Custom Preamble [63:56]	MII SOP control channel replaces it.
TX Preamble [55:48]	Custom Preamble [55:48]	0x55
TX Preamble [47:40]	Custom Preamble [47:40]	0x55
TX Preamble [39:32]	Custom Preamble [39:32]	0x55
TX Preamble [31:24]	Custom Preamble [31:24]	0x55
TX Preamble [23:16]	Custom Preamble [23:16]	0x55
TX Preamble [15:8]	Custom Preamble [15:8]	0x55
TX Preamble [7:0]	Custom Preamble [7:0]	0xD5 (SFD)
TX Data[127:120]	MAC Dest Addr[47:40]	
TX Data [119:112]	MAC Dest Addr[39:32]	
TX Data [111:104]	MAC Dest Addr[31:24]	
TX Data [103:96]	MAC Dest Addr[23:16]	
TX Data[95:88]	MAC Dest Addr[15:8]	
TX Data [87:80]	MAC Dest Addr[7:0]	
TX Data [79:72]	MAC Src Addr[47:40]	When you turn on Source Address Insertion , contents are replaced by txmac_saddr unless i_tx_skip_crc is high.
TX Data [71:64]	MAC Src Addr[39:32]	
TX Data [63:56]	MAC Src Addr[31:24]	
TX Data [55:48]	MAC Src Addr[23:16]	
TX Data [47:40]	MAC Src Addr[15:8]	
TX Data [39:32]	MAC Src Addr[7:0]	
TX Data [31:24]	Length/Type[15:8]	
TX Data [23:16]	Length/Type[7:0]	
TX Data [15:0]	MAC Client data or VLAN header	

7.2.3. Using MAC Avalon ST skip_crc Signal to Control Source Address, PAD, and CRC Insertion

The i_tx_skip_crc port allows the CRC to turn on and off per packet basis.

Figure 37. Using tx_skip_crc



Some systems have multiple streams of data feeding the same Ethernet link. If some of those streams provide packets with CRC already calculated, you can use the `i_tx_skip_crc` signal to not allow their recalculation. Some systems use frames smaller than the Ethernet minimum size. In this case, you must disable the frame padding.

Some protocols modify the CRC to indicate special conditions. You can use the `i_tx_skip_crc` signal to pass these special CRCs to the line without turning CRC off for all packets. Use the `i_tx_skip_crc` signal when bridging packets and respacing them without changing the received CRC.

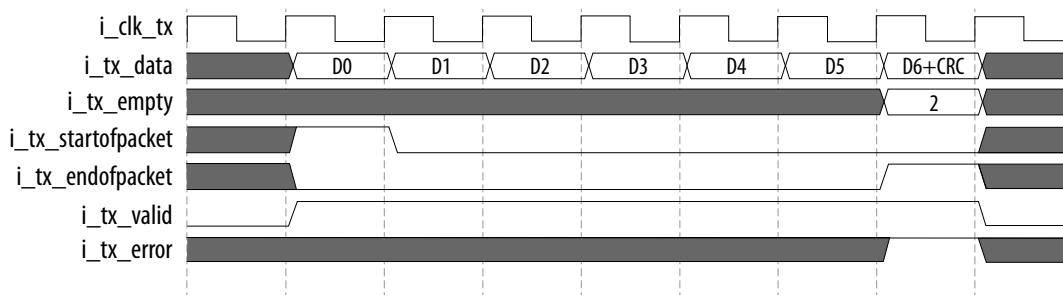
Table 39. Consequences of Asserting the tx_skip_crc Signal

MAC Field	<code>i_tx_skip_crc = 0</code>	<code>i_tx_skip_crc = 1</code>
Source Address	When you enable Use Source Address Insertion , <code>i_txmac_saddr</code> replaces the content of the source address bytes.	Regardless of Use Source Address Insertion status, the source address bytes are not replaced.
Padding	Frames with size of 64 bytes or less are padded to 64 bytes.	No padding is added.
CRC (Frame Check Sequence)	Calculates the packet's CRC and append it to the end.	No CRC calculation. Instead, uses last four bytes of the <code>i_tx_data</code> as a CRC value.

7.2.4. Using MAC Avalon ST `i_tx_error` Signal to Mark Packets Invalid

The `i_tx_error` port allows packets to be marked as errored when they are complete.

Figure 38. Using i_tx_error



Because the core uses a cut-through interface, the core starts transmitting the packet data it is given as soon as possible. If the core discovers an error after the packet starts, e.g. in a bridging system where the receiver also uses a cut-through interface, you can use `i_tx_error` to invalidate the packet. You can also use `i_tx_error` for testing, to generate errored packets, and confirm that the other end of the link is able to reject the errored packets.

To invalidate an errored frame, end it with `i_tx_endofpacket` and assert `i_tx_error`. If the frame is good, deassert `i_tx_error`.

Note: Using `i_tx_error` does not provide a robust test of the remote CRC, because it uses MII Error Control bytes to indicate error, rather than relying on corrupted CRC bits.

7.3. RX MAC Avalon ST Aligned Client Interface

The F-Tile Ethernet Intel FPGA Hard IP RX client interface in MAC+PCS variations employs the Avalon streaming interface protocol. The Avalon streaming interface protocol is a synchronous point-to-point, unidirectional interface that connects the producer of a data stream (source) to a consumer of data (sink). The key properties of this interface include:

- Start of packet (SOP) and end of packet (EOP) signals delimit frame transfers.
- The SOP must always be in the MSB, simplifying the interpretation and processing of data you receive on this interface.
- A valid signal qualifies signals from source to sink.

The RX MAC acts as a source and the client acts as a sink in the receive direction.

Table 40. Signals of the RX MAC Avalon ST Client Interface

All interface signals are clocked by the RX clock (`i_clk_rx`). The signal names are standard Avalon streaming interface signals.

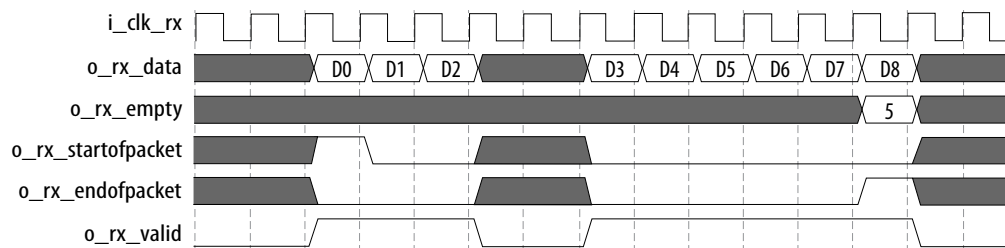
Name	Width	Description
o_rx_data[511:0] o_rx_data[127:0] o_rx_data[63:0]	512 bits (100GE) 128 bits (50GE/40GE) 64 bits (25GE/10GE)	Output data to the MAC when the rate is 10GE/25/40GE/50GE/100GE. Bits 0 is the LSB.
o_rx_valid	1 bit	When asserted, indicates that RX data is valid. Only valid between the SOP and EOP signals. This signal might be deasserted between the assertion of the SOP and EOP signals.
o_rx_empty[5:0] o_rx_empty[3:0] o_rx_empty[2:0]	6 bits (100GE) 4 bits (50GE/40GE) 3 bits (10GE/25GE)	Indicates the number of empty bytes on the RX data signal when EOP signal is asserted, starting from the least significant byte (LSB).
o_rx_startofpacket	1 bit	When asserted, indicates that the RX data signal holds the first clock cycle of data in a packet (start of packet). The IP core asserts this signal for only a single clock cycle for each packet. When the SOP signal is asserted, the MSB of the RX data signal drives the start of packet.
o_rx_endofpacket	1 bit	When asserted, indicates that the RX data signal holds the final clock cycle of data in a packet (end of packet). The IP core asserts this signal for only a single clock cycle for each packet.

continued...

Name	Width	Description
		In the case of an undersized frame or in the case of a frame that is exactly 64 bytes long, the SOP and EOP signals might be asserted in the same clock cycle.
o_rx_error[5:0]	6 bits	<p>Reports certain types of errors in the Ethernet frame whose contents are currently being transmitted on the client interface. This signal is valid in EOP cycles only.</p> <p>The individual bits report different types of errors:</p> <ul style="list-style-type: none"> • Bit [0]: Malformed packet error. If this bit has the value of 1, the packet is malformed. The IP core identifies a malformed packet when it receives a control character that is not a terminate character. • Bit [1]: CRC error. If this bit has the value of 1, the IP core detected a CRC error, error character in the frame, malformed, undersized, or truncated packets.. • Bit [2]: undersized or oversized frame. The IP core does not recognize an incoming frame of size eight bytes or less as a frame, and those cases are not reported here. If the preamble-passthrough and CRC forwarding settings cause the RX MAC to strip out bytes such that only eight bytes or less remain in the frame, the IP core also does not recognize the frame, and those cases are not reported here. If the frame is malformed, the case is not reported here. • Bit [3]: Reserved. • Bit [4]: payload length error. If this bit has the value of 1, the payload received in the frame is shorter than the length field value, and the value in the length field is less than or equal 1500 bytes. If the frame is oversized or undersized, the case is not reported here. If the frame is malformed, the case is not reported here. • Bit [5]: Reserved.
o_rxstatus_valid	1 bit	When asserted, indicates that o_rxstatus_data is driving valid data.
o_rxstatus_data[39:0]	40 bits	<p>Specifies information about the received frame. The following fields are defined:</p> <ul style="list-style-type: none"> • Bits [39:38]: Reserved. • Bit [37]: When asserted, indicates a BCAST/MCAST frame • Bit [36]: Reserved • Bit [35]: When asserted, indicates a PAUSE frame • Bit [34]: When asserted, indicates a Control (Type is 0x8808) frame • Bit[33]: When asserted, indicates a VLAN frame and a stacked VLAN frame • Bits[32:0]: Reserved
o_rx_preamble[63:0]	64 bits	<p>Reads the preamble value of an RX frame. This signal is valid when the o_rx_valid and the o_rx_startofpacket signals are asserted.</p> <p>This signal is only available when you turn on Preamble Passthrough in the parameter editor for 40GE/50GE channels.</p>
continued...		

Name	Width	Description
		<i>Note:</i> For preamble passthrough with other rates, use the first eight bytes of the o_rx_data signal.
i_rx_serial[3:0]	4 bits	Input RX serial transceiver signal to connect from RX pins on board to RX serial pins on the transceiver for lane reverse ordering.

Figure 39. Receiving Data Using the RX MAC Avalon ST Client Interface



The figure above shows how to receive data using the RX MAC Avalon ST client interface. The interface complies with the Avalon streaming interface specification.

- Packets always start on the MSB of o_rx_data (SOP aligned).
- When the frame ends, o_rx_empty is set to the number of unused bytes in o_rx_data, starting from the right (byte 0).
 - In this example, o_rx_data on the last cycle of the packet has 5 empty bytes.
 - The minimum number of bytes on the last cycle is 1.
- The framing and data ports are only valid when o_rx_data is high.

Note: The interface does not take direct backpressure.

Related Information

[RX MAC Segmented Client Interface Status and Errors](#) on page 121

7.3.1. RX MAC Avalon ST Client Interface with Disabled Preamble Passthrough

Table 41. RX MAC Field Positions in o_rx_data with Preamble Passthrough Disabled

100GE o_rx_data	40GE/50GE o_rx_data	10GE/25GE o_rx_data	MAC Field	Note
[511:504]	[127:120]	[63:56]	Dest Addr[47:40]	The first octet of the Destination Address, follows Start Frame Delimiter (SFD).
[503:496]	[119:112]	[55:48]	Dest Addr[39:32]	
[495:488]	[111:104]	[47:40]	Dest Addr[31:24]	
[495:480]	[103:96]	[39:32]	Dest Addr[23:16]	
[479:472]	[95:88]	[31:24]	Dest Addr[15:8]	
continued...				

100GE o_rx_data	40GE/50GE o_rx_data	10GE/25GE o_rx_data	MAC Field	Note
[471:464]	[87:80]	[23:16]	Dest Addr[7:0]	
[463:456]	[79:72]	[15:8]	Src Addr[47:40]	
[455:448]	[71:64]	[7:0]	Src Addr[39:32]	
[447:440]	[63:56]	[63:56]	Src Addr[31:24]	
[439:432]	[55:48]	[55:48]	Src Addr[23:16]	
[431:424]	[47:40]	[47:40]	Src Addr[15:8]	
[423:416]	[39:32]	[39:32]	Src Addr[7:0]	
[415:408]	[31:24]	[31:24]	Length/Type[15:8]	
[407:400]	[23:16]	[23:16]	Length/Type[7:0]	
[399:0]	[15:0]	[15:0]	...	

In the table above, the byte order on the bus and the data bit order follow the TX MAC Avalon ST client interface. For example,

- For 100GE, the first received bit after the SFD was bit 504, i.e. bit 0 of the first received byte.
- For 40GE/50GE, the first bit of the first received byte is bit 120.
- For 10GE/25GE, the first bit of the first received byte that is bit 56.

7.3.2. RX MAC Avalon ST Client Interface with Enabled Passthrough and RX CRC Forwarding

When you turn on **Preamble Passthrough** in the parameter editor, the IP cores with **Ethernet Mode** set to 100GE data rate presents the preamble received on the first eight bytes of the o_rx_data signal; specifically, o_rx_data[511:448]. The IP cores with **Ethernet Mode** set to 40GE or 50GE data rates present preamble received on the o_rx_preamble port with each packet, while the o_rx_startofpacket signal is high.

The IP cores with **Ethernet Mode** set to 10GE or 25GE data rates present the preamble data received on the first cycle, o_rx_data[63:0].

7.3.3. RX MAC Adapter Limitations

The RX MAC Adapter has limitations on the pattern of undersized frames passed to the MAC Avalon streaming interface.

In the 40GE/50GE RX MAC client interface with enabled CRC passthrough, you may observe dropped frames with fewer than nine octets. When CRC passthrough is disabled, you may observe the dropped frames with fewer than 13 octets.

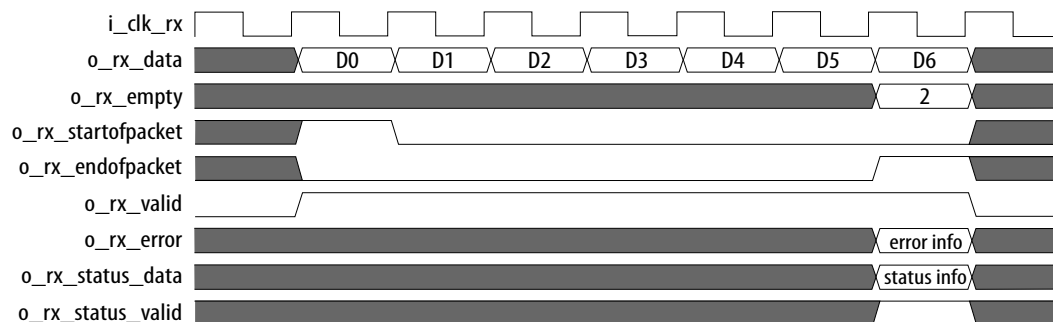
In the 100GE RX MAC Avalon ST client interface, if two frames received with frame starts less than 5 blocks apart, the first frame is dropped. A start is equivalent to the start of the frame. A block references a 64 bit encoded block, as defined in the IEEE 802.3 Clause 49 standard.

You can view the number of dropped frames by reading the statistics registers.

7.3.4. RX MAC Avalon ST Client Interface Status

Figure 40. RX MAC Status and Errors

The figure shows how status and error information are presented with RX packets as they arrive.



The status valid port is provided for backward compatibility, but always asserts when `o_rx_endofpacket` is asserted and valid.

7.4. TX MAC Segmented Client Interface

The F-Tile Ethernet Intel FPGA Hard IP TX MAC segmented client interface allows you to write frame data to the TX MAC for transmission. The packets may start on any 8-byte segment.

Attention: To achieve the maximum throughput when using the TX MAC segmented interface, the input packets need to be packed tightly, leaving no idle segments in between.

Table 42. Signals of the TX MAC Segmented Client Interface

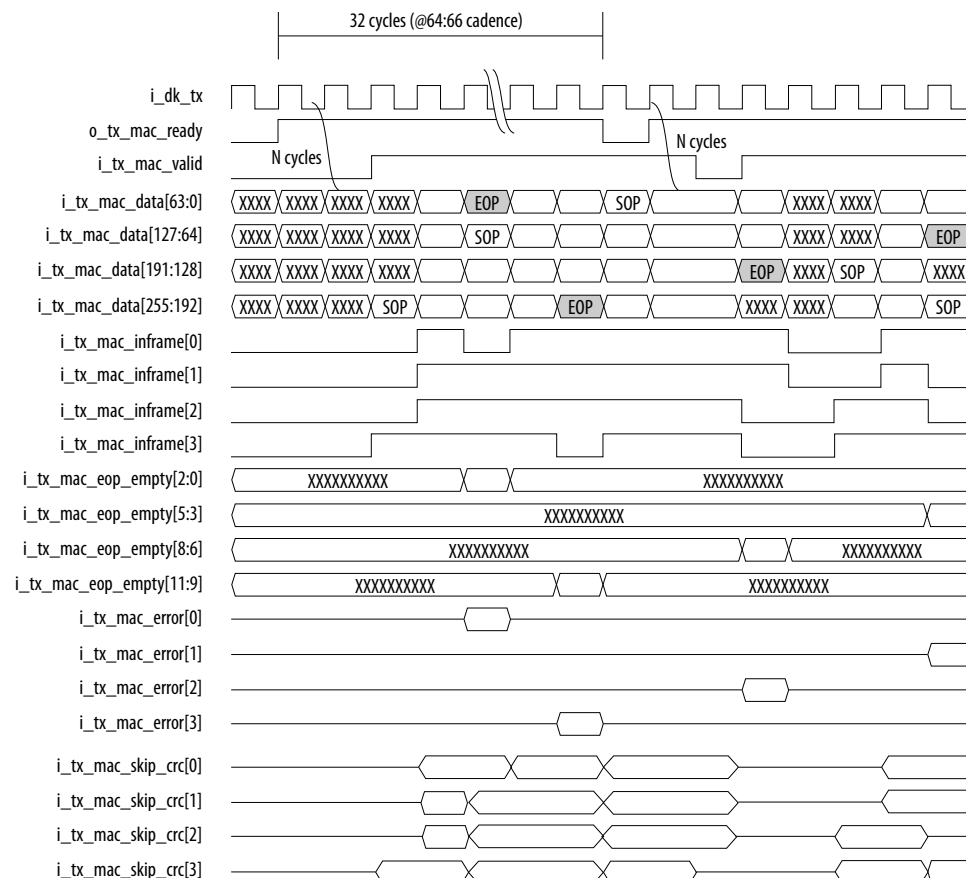
All interface signals are clocked by the TX clock.

Signal Name	Width	Description
i_tx_mac_data[1023:0] i_tx_mac_data[511:0] i_tx_mac_data[255:0] i_tx_mac_data[127:0] i_tx_mac_data[63:0]	1024 bits (400GE) 512 bits (200GE) 256 bits (100GE) 128 bits (50GE/40GE) 64 bits (25GE/10GE)	Input data to the MAC when the rate is 10GE/25/40GE/50GE/100GE/200GE/400GE. Bit 0 is the LSB.
i_tx_mac_valid	1 bit	When asserted, the TX data signal is valid on this cycle. Must follow at a fixed latency relative to <code>o_tx_mac_ready</code> , and may go low during transmission of a packet. When deasserted, values of input signals on the interface are hold constant.
i_tx_mac_inframe[15:0] i_tx_mac_inframe[7:0] i_tx_mac_inframe[3:0] i_tx_mac_inframe[1:0] i_tx_mac_inframe[0]	16 bits (400GE) 8 bits (200GE) 4 bits (100GE) 2 bits (50GE/40GE) 1 bits (25GE/10GE)	Indicates valid data in each segment for specific rate. Along with the previous segment's inframe signal, this signal indicates the SOP and EOP location.
i_tx_mac_eop_empty[47:0] i_tx_mac_eop_empty[23:0]	48 bits (400GE) 24 bits (200GE)	The empty signal indicates the number of empty bytes in that segment at the end of the MAC frame.

continued...

Signal Name	Width	Description
i_tx_mac_eop_empty[11:0] i_tx_mac_eop_empty[5:0] i_tx_mac_eop_empty[2:0]	12 bits (100GE) 6 bits (50GE/40GE) 3 bits (25GE/10GE)	
o_tx_mac_ready	1 bit	The ready signal indicates the MAC is ready to receive data in normal operational mode.
i_tx_mac_error[15:0] i_tx_mac_error[7:0] i_tx_mac_error[3:0] i_tx_mac_error[1:0] i_tx_mac_error[0]	16 bits (400GE) 8 bits (200GE) 4 bits (100GE) 2 bits (50GE/40GE) 1 bits (25GE/10GE)	The error signal cause the TX frame treated as an error. Must assert in the segment where the frame ends.
i_tx_mac_skip_crc[15:0] i_tx_mac_skip_crc[7:0] i_tx_mac_skip_crc[3:0] i_tx_mac_skip_crc[1:0] i_tx_mac_skip_crc[0]	16 bits (400GE) 8 bits (200GE) 4 bits (100GE) 2 bits (50GE/40GE) 1 bits (25GE/10GE)	The skip CRC signal instructs the MAC to not insert CRC/PAD in a current packet. Must be asserted along with all valid data segments for the packet.

Figure 41. Transmitting Data Using the TX MAC Segmented Client Interface



The figure above shows how to transmit data using the TX MAC segmented client interface. In this example, the figure shows a 100GE port transmitting several packets, represented by 4 segments.

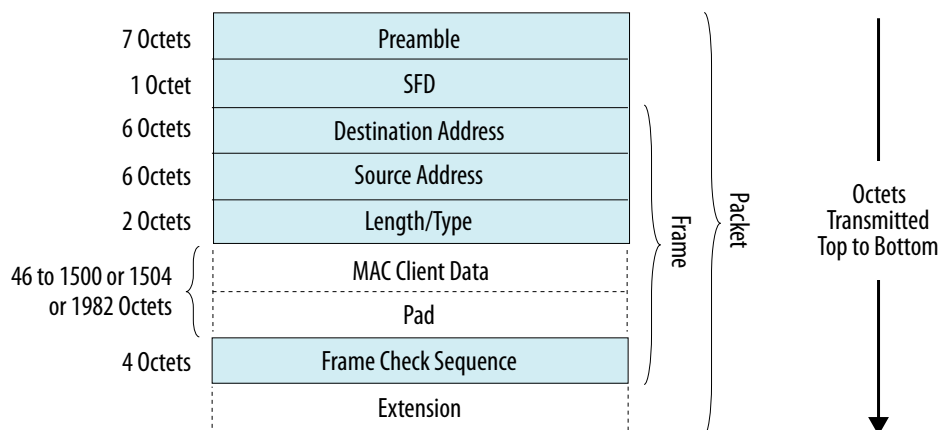
- Packets may start on any 8-byte segment of `i_tx_mac_data` (Segmented). The `i_tx_mac_inframe` transition from 0 to 1 (between two consecutive segments) indicates a start of packet (SOP). The SOP begins at a segment where `i_tx_mac_frame` is set to 1.
- For multisegmented interfaces, a new packet may start and the previous packet end are within the same cycle.
- The byte order for the TX MAC segmented client interface is **the reverse** of the MAC Avalon ST client byte order. The bytes flow from right to left, LSB to MSB. The first transmitted byte from the interface is the `i_tx_mac_data[7:0]`.
- The bit order for the TX MAC segmented client interface is **the same** as the MAC Avalon ST client bit order. The first transmitted bit from the interface is the `i_tx_mac_data[0]`.
- - The `i_tx_mac_valid` signal asserts only when the `o_tx_mac_ready` signal is asserted. The `i_tx_mac_valid` signal deasserts only when the `o_tx_mac_ready` signal is deasserted. Also, the `i_tx_mac_valid` may go low during the packet transmission.
 - The `i_tx_mac_valid` and the `o_tx_mac_ready` signals can be spaced by a fixed latency between 1 to 8 clock cycles.
 - When `i_tx_mac_valid` deasserts, `i_tx_mac_data`, `i_tx_mac_inframe`, `i_tx_mac_eop_empty`, `i_tx_mac_error`, and `i_tx_skip_crc` signals must be paused for as many cycles as `o_tx_mac_ready` is deasserted.
- When the frame ends, `i_tx_mac_eop_empty` is set to the number of unused bytes in `i_tx_mac_data`.
 - The `i_tx_mac_eop_empty` transition from 1 to 0 (between two consecutive segments) indicates the end of packet (EOP). The EOP ends in the segment where `i_tx_mac_inframe` is set to 0.
 - The minimum number of bytes on the last cycle is 1.

7.4.1. TX MAC Segmented Client Interface with Disabled Preamble Passthrough

Figure 42. Fields and Frame Boundaries in an Ethernet Packet

When you turn off **Preamble Passthrough** in the parameter editor, `i_tx_mac_data` must be written as shown below for the first cycle of data presented to the MAC.

Note: For 10GE/25GE channels, multiple cycles are required to write the header data.


Table 43. TX MAC Field Positions in `i_tx_mac_data` with Preamble Passthrough Disabled

Attention: 10GE/25GE requires multiple transfer cycle for header data.

The (') symbol in the **10GE/25GE `i_tx_mac_data`** column represents transfer on the subsequent cycle.

40GE/50GE/100GE/ 200GE/400G <code>i_tx_mac_data</code>	10GE/25GE <code>i_tx_mac_data</code>	MAC Field	Note
[7:0]	[7:0]'	Dest Addr[47:40]	The first octet of the Destination Address, follows Start Frame Delimiter (SFD).
[15:8]	[15:8]'	Dest Addr[39:32]	
[23:16]	[23:16]'	Dest Addr[31:24]	
[31:24]	[31:24]'	Dest Addr[23:16]	
[39:32]	[39:32]'	Dest Addr[15:8]	
[47:40]	[47:40]'	Dest Addr[7:0]	
[55:48]	[55:48]'	Src Addr[47:40]	When you turn on Source Address Insertion , contents are replaced by txmac_saddr unless <code>i_tx_mac_skip_crc</code> is high.
[63:56]	[63:56]'	Src Addr[39:32]	
[71:64]	[7:0]	Src Addr[31:24]	
[79:72]	[15:8]	Src Addr[23:16]	
[87:80]	[23:16]	Src Addr[15:8]	
[95:88]	[31:24]	Src Addr[7:0]	
[103:96]	[39:32]	Length/Type[15:8]	
[111:104]	[47:40]	Length/Type[7:0]	
[...:112]	[63:48]	...	

Note:

- In the table above, the byte order on the bus flows from LSB to MSB, the first byte of the MAC destination address is the LSB. The MAC considers this to be the first byte after the Start Frame Delimiter (SFD).
- The bit numbered 0 is always the LSB of each byte.
 - For example, on 40GE/50GE/100GE/200GE/400GE interface, `i_tx_data[0]` transmits after the SFD, and corresponds to the Ethernet destination address unicast/multicast bit.

7.4.2. TX MAC Segmented Client Interface with Enabled Preamble Passthrough

When you turn on **Preamble Passthrough** in the parameter editor, you must provide 8 preamble bytes to the TX MAC segmented client interface.

The table below describes TX MAC segmented field positions with enabled Preamble Passthrough parameter.

- MII Start of Packet control byte always replaces the first preamble byte.
- Bits[55:8] are the preamble bits, typically set to the 0x55 value.
- Bits[63:56] are the last preamble byte. In a standard preamble, it is set to the Start Frame Delimiter 0xD5 value.

Table 44. TX MAC Segmented Field Positions in `i_tx_mac_data` with Preamble Passthrough Enabled

Attention:

10GE/25GE requires multiple transfer cycle for header data.

The (') symbol in the **10GE/25GE `i_tx_mac_data`** column represents transfer on the subsequent cycle.

The (") symbol in the **10GE/25GE `i_tx_mac_data`** column represents transfer on the 2nd subsequent cycle.

100GE/400GE <code>i_tx_mac_data</code>	40GE/50GE <code>i_tx_mac_data</code>	10GE/25GE <code>i_tx_mac_data</code>	MAC Field	Note
[7:0]	[7:0]'	[7:0]"	Custom Preamble [7:0]	MII SOP control channel replaces it.
[15:8]	[15:8]'	[15:8]"	Custom Preamble [15:8]	0x55
[23:16]	[23:16]'	[23:16]"	Custom Preamble [23:16]	0x55
[31:24]	[31:24]'	[31:24]"	Custom Preamble [31:24]	0x55
[39:32]	[39:32]'	[39:32]"	Custom Preamble [39:32]	0x55
[47:40]	[47:40]'	[47:40]"	Custom Preamble [47:40]	0x55
[55:48]	[55:48]'	[55:48]"	Custom Preamble [55:48]	0x55
[63:56]	[63:56]'	[63:56]"	Custom Preamble [63:56]	0xD5 (SFD)
continued...				

100GE/400GE i_tx_mac_data	40GE/50GE i_tx_mac_data	10GE/25GE i_tx_mac_data	MAC Field	Note
[71:64]	[71:64]'	[7:0]'	Dest Addr[47:40]	
[79:72]	[79:72]'	[15:8]'	Dest Addr[39:32]	
[87:80]	[87:80]'	[23:16]'	Dest Addr[31:24]	
[95:88]	[95:88]'	[31:24]'	Dest Addr[23:16]	
[103:96]	[103:96]'	[39:32]'	Dest Addr[15:8]	
[111:104]	[111:104]'	[47:40]'	Dest Addr[7:0]	
[119:112]	[119:112]'	[55:48]'	Src Addr[47:40]	When you turn on Source Address Insertion , contents are replaced by txmac_saddr unless i_tx_skip_crc is high.
[127:120]	[127:120]'	[63:56]'	Src Addr[39:32]	
[135:128]	[7:0]	[7:0]	Src Addr[31:24]	
[143:136]	[15:8]	[15:8]	Src Addr[23:16]	
[151:144]	[23:16]	[23:16]	Src Addr[15:8]	
[159:152]	[31:24]	[31:24]	Src Addr[7:0]	
[167:160]	[39:32]	[39:32]	Length/Type[15:8]	
[175:168]	[47:40]	[47:40]	Length/Type[7:0]	
[...:176]	[127:48]	[63:48]	...	

7.4.3. Using MAC Segmented skip_crc Signal to Control Source Address, PAD, and CRC Insertion

The MAC segmented i_tx_skip_crc signal behaves identically to the TX MAC Avalon ST client interface i_tx_skip_crc signal.

For more information, refer to the *Using MAC Avalon ST skip_crc Signal to Control Source Address, PAD, and CRC Insertion* section.

Related Information

[Using MAC Avalon ST skip_crc Signal to Control Source Address, PAD, and CRC Insertion](#) on page 105

7.4.4. Using MAC Segmented i_tx_mac_error to Mark Packets Invalid

The i_tx_mac_error signal behaves identically in the TX MAC segmented client interface and the TX MAC Avalon ST client interface with the exception that the i_tx_mac_error signal must be asserted for the appropriate segment in which the EOP occurs.

For more information, refer to the *Using MAC Avalon ST i_tx_error Signal to Mark Packet Invalid* section.

Related Information

[Using MAC Avalon ST i_tx_error Signal to Mark Packets Invalid](#) on page 106

7.5. RX MAC Segmented Client Interface

The F-Tile Ethernet Intel FPGA Hard IP RX client interface allows the application to read frame data from the RX MAC segmented when it is received. Packets may start on any 8-byte segment.

Table 45. Signals of the Avalon Streaming RX Client Interface

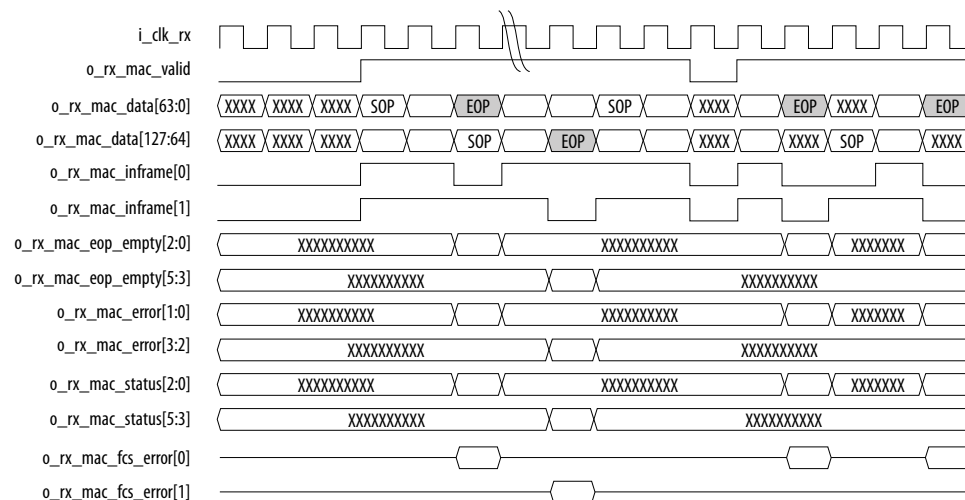
All interface signals are clocked by the RX clock.

Name	Width	Description
o_rx_mac_data[1023:0] o_rx_mac_data[511:0] o_rx_mac_data[255:0] o_rx_mac_data[127:0] o_rx_mac_data[63:0]	1024 bits (400GE) 512 bits (200GE) 256 bits (100GE) 128 bits (50GE/40GE) 64 bits (25GE/10GE)	Output data to the MAC when the rate is 10GE/25/40GE/50GE/100GE/200GE/400GE. Bit 0 is the LSB.
o_rx_mac_valid	1 bit	When asserted, indicates that RX data is valid on this cycle When deasserted, ignore the signals value.
i_rx_mac_inframe[15:0] i_rx_mac_inframe[7:0] i_rx_mac_inframe[3:0] i_rx_mac_inframe[1:0] i_rx_mac_inframe[0]	16 bits (400GE) 8 bits (200GE) 4 bits (100GE) 2 bits (50GE/40GE) 1 bit (25GE/10GE)	Indicates the valid data in each EMIB. With the previous segment's inframe signal, indicates the SOP and EOP location. Each segment is 64 bits.
o_rx_mac_eop_empty[47:0] o_rx_mac_eop_empty[23:0] o_rx_mac_eop_empty[11:0] o_rx_mac_eop_empty[5:0] o_rx_mac_eop_empty[2:0]	48 bits (400GE) 24 bits (200GE) 12 bits (100GE) 6 bits (50GE/40GE) 3 bits (25GE/10GE)	Indicates the number of empty bytes on the RX data signal where EOC occurs, starting from the most significant byte (MSB).
o_rx_mac_fcs_error[15:0] o_rx_mac_fcs_error[7:0] o_rx_mac_fcs_error[3:0] o_rx_mac_fcs_error[1:0] o_rx_mac_fcs_error[0]	16 bits (400GE) 8 bits (200GE) 4 bits (100GE) 2 bits (50GE/40GE) 1 bit (25GE/10GE)	Indicates that the currently ending frame has an error. Valid only on EOP segments. A valid high on this signal indicates that the frame that is ending either had an FCS error, was malformed, was undersized, or was oversized and truncated because the MTU enforcement feature of the RX MAC was enabled.
o_rx_mac_error[31:0] o_rx_mac_error[15:0] o_rx_mac_error[7:0] o_rx_mac_error[3:0] o_rx_mac_error[1:0]	32 bits (400GE) 16 bits (200GE) 8 bits (100GE) 4 bits (50GE/40GE) 1 bit (25GE/10GE)	Indicates that the currently ending frame has a decoding error.

continued...

Name	Width	Description
		<p>The [1:0] bits for each segment accept values 0, 1, 2, and 3, and report different types of errors:</p> <ul style="list-style-type: none"> 2'd0: If FCS error is deasserted, the packet contains no error. 2'd1: Malformed packet error. The IP core identifies a malformed packet when it receives a control character that is not a terminate character. 2'd2: Undersized (frame shorter than 64b) or oversized (frame larger than the programmed maximum frame rate) frame. 2'd3: Payload length error. The payload received in the frame is shorter than the length field value, and the value in the length field is less than or equal 1500 bytes. If the frame is oversized or undersized, the case is not reported here. If the frame is malformed, the case is not reported here.
o_rx_mac_status_data[47:0] o_rx_mac_status_data[23:0] o_rx_mac_status_data[11:0] o_rx_mac_status_data[5:0] o_rx_mac_status_data[2:0]	48 bits (400GE) 24 bits (200GE) 12 bits (100GE) 6 bits (50GE/40GE) 3 bits (25GE/10GE)	<p>Specifies information about the received frame. If multiple status occur, the status is reported in 5, 4, 7, 6, 1, 2, 3, 0 priority order.</p> <p>The individual bits report different information:</p> <ul style="list-style-type: none"> 3'd7: When asserted, indicates a FC (Length/Type is 0x8808) frame 3'd6: When asserted, indicates illegal length frame 3'd5: When asserted, indicates a SVLAN frame and a stacked VLAN frame 3'd4: When asserted, indicates a SFC frame or PFC frame 3'd3: Reserved 3'd2: When asserted, indicates a BCAST frame and a MCAST frame 3'd1: When asserted, indicates an Ethernet type frame that is not FC frame 3'd0: When asserted, indicates a valid length frame

Figure 43. Receiving Data Using the RX MAC Client Interface



The figure above shows how to receive data using the RX MAC segmented client interface.

- Packets may start on any 8-byte segment of `i_tx_mac_data` (Segmented). The `i_tx_mac_inframe` transition from 0 to 1 (between two consecutive segments) indicates a start of packet (SOP). The SOP begins at a segment where `o_rx_mac_frame` is set to 1.
- For multisegmented interfaces, a new packet may start and the previous packet end are within the same cycle.
- When the packet ends, `o_rx_mac_eop_empty` is set to the number of unused bytes in `o_rx_mac_data`.
 - The `o_rx_mac_eop_empty` transition from 1 to 0 (between two consecutive segments) indicates the end of packet (EOP). The EOP ends in the segment where `o_rx_mac_inframe` is set to 0.
 - The minimum number of bytes on the last cycle is 1.

Note: The interface does not take direct backpressure.

7.5.1. RX MAC Segmented Client Interface with Disabled Preamble Passthrough

Table 46. RX MAC Field Positions in `o_rx_mac_data` with Preamble Passthrough Disabled

40GE/50GE/100GE/ 200GE/400GE <code>o_rx_mac_data</code>	10GE/25GE <code>o_rx_mac_data</code>	MAC Field	Note
[7:0]	[7:0]	Dest Addr[47:40]	The first octet of the Destination Address, follows Start Frame Delimiter (SFD).
[15:8]	[15:8]	Dest Addr[39:32]	
[23:16]	[23:16]	Dest Addr[31:24]	
[31:24]	[31:24]	Dest Addr[23:16]	
[39:32]	[39:32]	Dest Addr[15:8]	
[47:40]	[47:40]	Dest Addr[7:0]	
[55:48]	[55:48]	Src Addr[47:40]	
[63:56]	[7:0]	Src Addr[39:32]	
[71:64]	[7:0]	Src Addr[31:24]	
[79:72]	[15:8]	Src Addr[23:16]	
[87:80]	[23:16]	Src Addr[15:8]	
[95:88]	[31:24]	Src Addr[7:0]	
[103:96]	[39:32]	Length/Type[15:0]	
[111:104]	[47:40]	Length/Type[7:0]	
[...:112]	[63:48]	...	

For 10GE/25GE variants, the header arrives over 2 clock cycles.

7.5.2. RX MAC Segmented Client Interface with Enabled Passthrough and RX CRC Forwarding

The RX MAC segmented client interface is available when you select **Enable Preamble Passthrough** in the parameter editor.

For all data rates, the IP core presents the preamble bits received on the first 8 bytes of the `o_rx_mac_data`, that is `o_rx_mac_data[63:0]`. When **Remove CRC Bytes** is deselected, the CRC bytes are equivalent to the last 4 bytes of data while in the segment containing the EOP `o_rx_endofpacket` is asserted.

Table 47. RX MAC Field Positions in `o_rx_mac_data` with Preamble Passthrough Enabled

Attention: 10G/25G requires multiple transfer cycle for header data.

The (') symbol in the **10GE/25GE `o_rx_mac_data`** column represents transfer on the subsequent cycle.

The (") symbol in the **10GE/25GE `o_rx_mac_data`** column represents transfer on the 2nd subsequent cycle.

100GE/200GE/ 400GE o_rx_mac_data	40GE/50GE o_rx_mac_data	10GE/25GE o_rx_mac_data	MAC Field	Note
[7:0]	[7:0]'	[7:0]"	Preamble [7:0]	0x55
[15:8]	[15:8]'	[15:8]"	Preamble [15:8]	0x55
[23:16]	[23:16]'	[23:16]"	Preamble [23:16]	0x55
[31:24]	[31:24]'	[31:24]"	Preamble [31:24]	0x55
[39:32]	[39:32]'	[39:32]"	Preamble [39:32]	0x55
[47:40]	[47:40]'	[47:40]"	Preamble [47:40]	0x55
[55:48]	[55:48]'	[55:48]"	Preamble [55:48]	0x55
[63:56]	[63:56]'	[63:56]"	Preamble [63:56]	0xD5 (SFD)
[71:64]	[71:64]'	[7:0]'	Dest Addr[47:40]	
[79:72]	[79:72]'	[15:8]'	Dest Addr[39:32]	
[87:80]	[87:80]'	[23:16]'	Dest Addr[31:24]	
[95:88]	[95:88]'	[31:24]'	Dest Addr[23:16]	
[103:96]	[103:96]'	[39:32]'	Dest Addr[15:8]	
[111:104]	[111:104]'	[47:40]'	Dest Addr[7:0]	
[119:112]	[119:112]'	[55:48]'	Src Addr[47:40]	When you turn on Source Address Insertion , contents are replaced by txmac_saddr unless i_tx_skip_crc is high.
[127:120]	[127:120]'	[63:56]'	Src Addr[39:32]	
[135:128]	[7:0]	[7:0]	Src Addr[31:24]	
[143:136]	[15:8]	[15:8]	Src Addr[23:16]	
[151:144]	[23:16]	[23:16]	Src Addr[15:8]	
[159:152]	[31:24]	[31:24]	Src Addr[7:0]	
continued...				

100GE/200GE/ 400GE o_rx_mac_data	40GE/50GE o_rx_mac_data	10GE/25GE o_rx_mac_data	MAC Field	Note
[167:160]	[39:32]	[39:32]	Length/Type[15:0]	
[175:168]	[47:40]	[47:40]	Length/Type[7:0]	
[...:176]	[127:48]	[63:48]	...	

7.5.3. RX MAC Segmented Client Interface Status and Errors

As with the Avalon streaming interface, MAC Status and Error o_rx_mac_status, and o_rx_mac_error signals are valid at the end of packet. For more information, refer to *RX MAC Avalon ST Client Interface Status*.

In the RX MAC segmented interface, these signals are aligned to the appropriate segment in which EOP occurs. In addition, decoding of the status and error signals is different for the MAC segmented Interface. For more information, refer to the *RX Aligned Client Interface*.

Related Information

- [RX MAC Avalon ST Aligned Client Interface](#) on page 107
- [RX MAC Avalon ST Client Interface Status](#) on page 111

7.6. MAC Flow Control Interface

The F-Tile Ethernet Intel FPGA Hard IP MAC flow control interface in MAC Avalon ST or MAC segmented variation indicates and receives flow control events notifications.

Table 48. MAC Flow Control Interface

All interface signals are clocked by the i_clk_tx clock. For 10GE/25GE channels, all interface signals are asynchronous.

Signal Name	Width	Description
i_tx_pause	1 bit	When asserted, directs the IP core to send a PAUSE XOFF frame on the Ethernet link. <i>Note:</i> For 10GE/25GE channels, you should hold the i_tx_pause signal more than 205 ns to get the request captured by the MAC.
i_tx_pfc[7:0]	8 bits	When a bit is asserted, directs the IP core to send a PFC XOFF frame on the Ethernet link for the corresponding priority queue. <i>Note:</i> For 10GE/25GE channels, you should hold the i_tx_pfc signal more than 205 ns to get the request captured by the MAC. The rising edge triggers the request. You must maintain this signal at the value of 1 until you want the IP core to end the pause period. The IP core sends a PFC XOFF frame after it completes processing of the current in-flight TX packet, and periodically thereafter, until you deassert the i_tx_pfc bit. When you deassert the bit, the IP core sends a

continued...

Signal Name	Width	Description
		PFC XON frame on the Ethernet link for the corresponding priority queue. This signal is functional only if priority flow control is enabled.
o_rx_pause	1 bit	When asserted, indicates the IP core received a PAUSE XOFF frame on the Ethernet link.
o_rx_pfc[7:0]	8 bits	When a bit is asserted, indicates the IP core received a PFC XOFF frame on the Ethernet link for the corresponding priority queue.

7.7. PCS Mode TX Interface

The F-Tile Ethernet Intel FPGA Hard IP TX client interface in PCS variations employs the Media Independent Interface (MII) protocol.

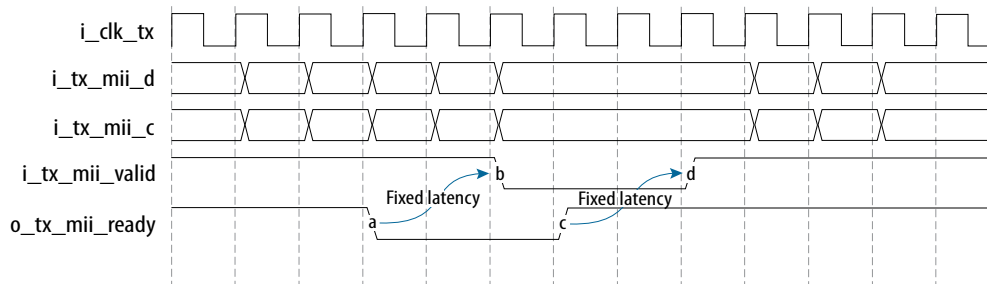
The client acts as a source and the TX PCS acts as a sink in the transmit direction.

Table 49. Signals of the MII TX Client Interface

All interface signals are clocked by the TX clock. The signal names are standard Avalon streaming interface signals.

Signal Name	Width	Description
i_tx_mii_d[1023:0] i_tx_mii_d[511:0] i_tx_mii_d[255:0] i_tx_mii_d[127:0] i_tx_mii_d[63:0]	1024 bits (400GE) 512 bits (200GE) 256 bits (100GE) 128 bits (40GE/50GE) 64 bits(10GE/25GE)	TX MII data. Data must be in MII encoding. i_tx_mii_d[7:0] holds the first byte the IP core transmits on the Ethernet link. i_tx_mii_d[0] holds the first bit the IP core transmits on the Ethernet link. While the TX MII valid signal has the value of 0 or the alignment marker insertion bit signal has the value of 1, and for one additional clock cycle, you must hold the value of this signal stable, a behavior called freezing the signal value.
i_tx_mii_c[127:0] i_tx_mii_c[63:0] i_tx_mii_c[31:0] i_tx_mii_c[15:0] i_tx_mii_c[7:0]	128 bits (400GE) 64 bits (200GE) 32 bits (100GE) 16 bits (40GE/50GE) 8 bits(10GE/25GE)	TX MII control bits. Each bit corresponds to a byte of the TX MII data signal. For example, i_tx_mii_c[0] corresponds to i_tx_mii_d[7:0], i_tx_mii_c[1] corresponds to i_tx_mii_d[15:8], and so on. If the value of a bit is 1, the corresponding data byte is a control byte. If the value of a bit is 0, the corresponding data byte is data.
i_tx_mii_valid	1 bit	Indicates that the TX MII data signal is valid. You must assert this signal a fixed number of clock cycles after the IP core raises ready signal, and must deassert this signal the same number of clock cycles after the IP core deasserts the ready signal. The number must be in the range of 1–6 clock cycles.
o_tx_mii_ready	1 bit	Indicates the PCS is ready to receive new data.
i_tx_mii_am	1 bit	Alignment marker insertion bit.

Figure 44. Transmitting Data Using the PCS Mode TX Interface



The figure above shows how to write packets directly to the PCS mode TX interface.

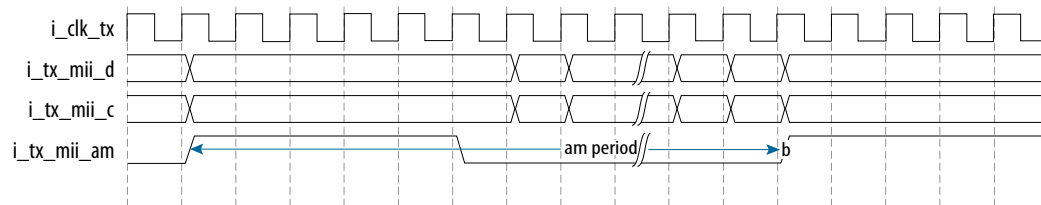
- The packets are written using MII.
 - Each byte in `i_tx_mii_d` has a corresponding bit in `i_tx_mii_c` that indicates whether the byte is a control byte or a data byte; for example, `i_tx_mii_c[1]` is the control bit for `i_tx_mii_d[15:8]`.
- `i_tx_mii_valid` should conform to these conditions:
 - Assert the valid signal only when the ready signal is asserted, and deassert only when the ready signal is deasserted.
 - The two signals can be spaced by a fixed latency between 1 and 6 cycles.
 - When the valid signal deasserts, `i_tx_mii_d` and `i_tx_mii_c` must be paused.
- The byte order for the PCS mode TX interface is opposite of the byte order for the MAC Avalon streaming interface. Bytes flow from LSB to MSB; the first byte to be transmitted from the interface is `i_tx_mii_d[7:0]`.
- The bit order for the PCS mode TX interface is the same as the bit order of the MAC client. The first bit to be transmitted from the interface is `i_tx_mii_d[0]`.

Note: The PCS mode TX interface is not SOP aligned. Any legal ordering of packets in MII format is accepted.

Table 50. Writing a Start Packet Block with Preamble to the PCS Mode TX Interface

MII Data		MII Control		Ethernet Packet Byte
<code>i_tx_mii_d[7:0]</code>	0xFB	<code>i_tx_mii_c[0]</code>	1	Start of Packet
<code>i_tx_mii_d[15:8]</code>	0x55	<code>i_tx_mii_c[1]</code>	0	Preamble
<code>i_tx_mii_d[23:16]</code>	0x55	<code>i_tx_mii_c[2]</code>	0	Preamble
<code>i_tx_mii_d[31:24]</code>	0x55	<code>i_tx_mii_c[3]</code>	0	Preamble
<code>i_tx_mii_d[39:32]</code>	0x55	<code>i_tx_mii_c[4]</code>	0	Preamble
<code>i_tx_mii_d[47:40]</code>	0x55	<code>i_tx_mii_c[5]</code>	0	Preamble
<code>i_tx_mii_d[55:48]</code>	0x55	<code>i_tx_mii_c[6]</code>	0	Preamble
<code>i_tx_mii_d[63:56]</code>	0xD5	<code>i_tx_mii_c[7]</code>	0	SFD

Figure 45. Inserting Alignment Markers



The fabric controls the timing of alignment marker insertion. Alignment markers cannot be delayed without disrupting the Ethernet link. Use valid cycles to count the alignment markers. When `i_tx_mii_valid` is low, the alignment marker counters and input must freeze.

For alignment marker counts, you only use valid cycles. When `i_tx_mii_valid` is low, the alignment marker counters and input data must freeze.

The number of cycles for `i_tx_mii_am` to remain high depends on the rate of the interface.

- 400GE: 2 cycles
- 200GE: 2 cycles
- 100GE: 5 cycles
- 50GE-2 with RS-FEC or 50GE-1 for IEEE: 2 cycles
- 50GE-2 without RS-FEC or 40GE: 2 cycles
- 25GE with RS-FEC: 4 cycles)

The number of cycles for am period depends on the rate of the interface and whether in simulation or hardware.

- In simulation, it is common to use a reduced am period for both sides of the link to increase lock-time speed.
 - 400GE: 640
 - 200GE: 640
 - 100GE: 1280
 - 50GE-2 with RS-FEC: 512
 - 50GE-2 without RS-FEC or 50GE: 640
 - 40GE: 128
 - 25GE with RS-FEC: 1280
- In hardware.
 - 400GE: 40960
 - 200GE: 40960
 - 100GE: 81920
 - 50GE-2 with RS-FEC RS(272,258) or 50GE-1 for IEEE: 40960
 - 40GE or 50GE-2 without RS-FEC: 32768
 - 25GE with RS-FEC: 81920

For proper operation, program the MAC, PCS, and optional FEC to use the same am period.

In FEC modes, the TX datapath does not come completely out of reset until at least 2 alignment marker periods passed. You must start driving `i_tx_mii_am` at the proper interval before `o_tx_pll_lanes_stable` goes high. You may drive the signal as soon as `o_tx_pll_locked` has gone high and `o_tx_mii_ready` starts toggling. When the external custom rate interface is enabled, you must start driving `i_custom_cadence`. For more information, refer to the [Custom Rate Interface](#) on page 131.

Related Information

[Custom Rate Interface](#) on page 131

7.8. PCS Mode RX Interface

The F-Tile Ethernet Intel FPGA Hard IP RX client interface in PCS Only variations employs the Media Independent Interface (MII) protocol.

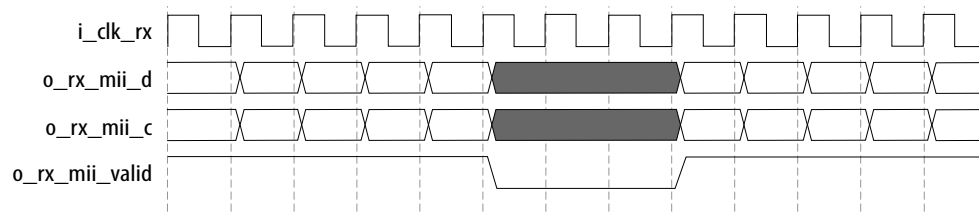
The RX PCS acts as a source and the client acts as a sink in the receive direction.

Table 51. Signals of the MII RX Client Interface

All interface signals are clocked by the RX clock. The signal names are standard Avalon streaming interface signals.

Signal Name	Width	Description
<code>o_rx_mii_d[1023:0]</code> <code>o_rx_mii_d[511:0]</code> <code>o_rx_mii_d[255:0]</code> <code>o_rx_mii_d[127:0]</code> <code>o_rx_mii_d[63:0]</code>	1024 bits (400GE) 512 bits (200GE) 256 bits (100GE) 128 bits (40GE/50GE) 64 bits(10GE/25GE)	RX MII data. Data is in MII encoding. <code>o_rx_mii_d[7:0]</code> holds the first byte the IP core received on the Ethernet link. <code>o_rx_mii_d[0]</code> holds the first bit the IP core received on the Ethernet link. When RX MII valid signal has the value of 0 or the RX valid alignment marker signal has the value of 1, the value on this signal is invalid.
<code>o_rx_mii_c[127:0]</code> <code>o_rx_mii_c[63:0]</code> <code>o_rx_mii_c[31:0]</code> <code>o_rx_mii_c[15:0]</code> <code>o_rx_mii_c[7:0]</code>	128 bits (400GE) 64 bits (200GE) 32 bits (100GE) 16 bits (40GE/50GE) 8 bits(10GE/40GE)	RX MII control bits. Each bit corresponds to a byte of RX MII data. <code>o_rx_mii_c[0]</code> corresponds to <code>o_rx_mii_d[7:0]</code> , <code>o_rx_mii_c[1]</code> corresponds to <code>o_rx_mii_d[15:8]</code> , and so on. If the value of a bit is 1, the corresponding data byte is a control byte. If the value of a bit is 0, the corresponding data byte is data.
<code>o_rx_mii_valid</code>	1 bit	Indicates that the RX MII data, RX MII control bits, and the RX valid alignment marker signals are valid.
<code>o_rx_mii_am_valid</code>	1 bit	Indicates the IP core received a valid alignment marker on the Ethernet link. When the RX MII valid signal has the value of 0, the value on this signal is invalid. The value of the RX MII valid signal may fall while the IP core is asserting this signal.

Figure 46. Receiving Data Using the PCS Mode RX Interface



The figure above shows how to read packets from the RX PCS using the PCS mode RX interface.

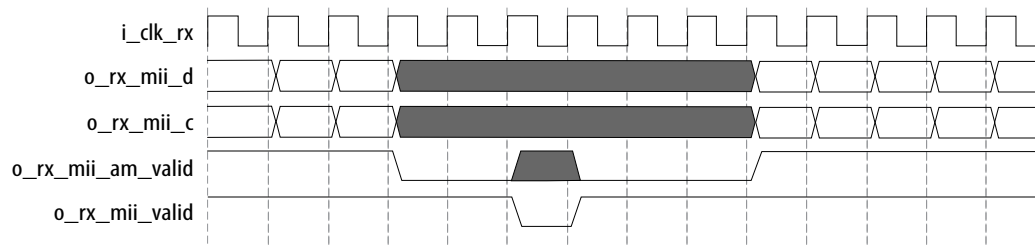
- The packets are MII encoded.
 - Each byte in `o_rx_mii_d` has a corresponding bit in `o_rx_mii_c` that indicates whether the byte is a control byte or a data byte; for example, `o_rx_mii_c[2]` is the control bit for `o_rx_mii_d[23:16]`.
- The data is only valid when `o_rx_mii_valid` is high. The contents of the `o_rx_mii_d` and `o_rx_mii_c` buses are not defined when `o_rx_mii_valid` is low.
- The byte order for the PCS mode RX interface is opposite of the byte order for the MAC segmented client interface. Bytes flow from LSB to MSB; the first byte that the core receives is `o_rx_mii_d[7:0]`.
- The bit order for the PCS mode RX interface is the same as the bit order of the MAC segmented client interface. The first bit that the core receives is `o_rx_mii_d[0]`.

Note: The PCS mode RX interface is not SOP aligned. New packets can begin on any byte position that is divisible by 8 (PCS data is transferred in 8-byte blocks).

Table 52. Reading a Start Packet Block with Preamble from a PCS Mode TX Interface

MII Data		MII Control		Ethernet Packet Byte
<code>o_rx_mii_d[7:0]</code>	0xFB	<code>o_rx_mii_c[0]</code>	1	Start of Packet
<code>o_rx_mii_d[15:8]</code>	0x55	<code>o_rx_mii_c[1]</code>	0	Preamble
<code>o_rx_mii_d[23:16]</code>	0x55	<code>o_rx_mii_c[2]</code>	0	Preamble
<code>o_rx_mii_d[31:24]</code>	0x55	<code>o_rx_mii_c[3]</code>	0	Preamble
<code>o_rx_mii_d[39:32]</code>	0x55	<code>o_rx_mii_c[4]</code>	0	Preamble
<code>o_rx_mii_d[47:40]</code>	0x55	<code>o_rx_mii_c[5]</code>	0	Preamble
<code>o_rx_mii_d[55:48]</code>	0x55	<code>o_rx_mii_c[6]</code>	0	Preamble
<code>o_rx_mii_d[63:56]</code>	0xD5	<code>o_rx_mii_c[7]</code>	0	SFD

Figure 47. Receiving Alignment Markers



`o_rx_mii_am_valid` indicates the arrival of the alignment markers from the RX PCS. The alignment markers also depend on `o_rx_mii_valid`. When `o_rx_mii_valid` is low, `o_rx_mii_am_valid` is not valid.

The contents of the `o_rx_mii_d` and `o_rx_mii_c` buses are not defined when `o_rx_mii_valid` is low. This is because alignment markers are not part of the 64b/66b encoding, and do not have an MII equivalent.

7.9. FlexE and OTN Mode TX Interface

The F-Tile Ethernet Intel FPGA Hard IP TX client interface in FlexE and OTN variations employs the PCS66 interface protocol.

The FlexE and OTN variations allow the application to write 66b blocks to the TX PCS, bypassing the TX MAC.

- In FlexE mode and 200GE/400GE OTN modes, the TX encoder in the PCS is also bypassed.
- In 10GE/25GE/40GE/50GE/100GE OTN mode, both the TX encoder and the scrambler are bypassed.

The client acts as a source and the TX PCS acts as a sink in the transmit direction.

Table 53. Signals of the PCS66 TX Interface

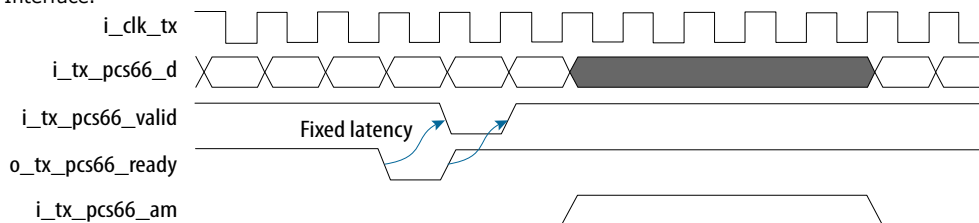
All interface signals are clocked by the TX clock. The signal names are standard Avalon streaming interface signals.

Signal Name	Width	Description
<code>i_tx_pcs66_d[1055:0]</code> <code>i_tx_pcs66_d[527:0]</code> <code>i_tx_pcs66_d[263:0]</code> <code>i_tx_pcs66_d[131:0]</code> <code>i_tx_pcs66_d[65:0]</code>	1056 bits (400GE) 528 bits (200GE) 264 bits (100GE) 132 bits (40GE/50GE) 66 bits (10GE/25GE)	TX PCS 66b data for 1 block. <ul style="list-style-type: none"> • In FlexE mode, the data presented is scrambled. • In OTN mode, the data goes directly to the RS-FEC or PMA.
<code>i_tx_pcs66_valid</code>	1 bit	When asserted, indicates that the TX PCS 66b data is valid. Must be asserted when the TX PCS 66b ready signal is asserted.
<code>o_tx_pcs66_ready</code>	1 bit	TX PCS 66b ready signal. When asserted, indicates the PCS is ready to receive new data.
<code>i_tx_pcs66_am</code>	1 bit for each channel	Alignment marker insertion bit.
<i>continued...</i>		

Signal Name	Width	Description
		<p>In FlexE mode, asserting this signal causes the PCS to allow gaps for the alignment markers in place of the data presented on the TX PCS data signal. The application marks the block as an alignment marker and the scrambler does not process the data.</p> <p>In OTN mode, RS-FEC must be aware of the alignment marker location.</p> <p>In OTN mode without FEC, the <code>i_tx_pcs66_am</code> signal is optional. You can tie it low. The Transcode Decoder inserts and removes the alignment markers for Ethernet modes with FEC (200GE/400GE) using 400GE PCS. For 100GE, the PCS inserts the alignment marker and remaps it using the FEC Transcode Decoder. In this mode FEC is simulated as a sublayer between 100GE PCS and PMA. Therefore, the 100GE alignment markers exist outside of the FEC.</p> <p>Note: When the Deterministic Latency Interface or Deterministic Latency Measurement parameter is turned on, sync pulses should be provided for 25GE and 10GE variants. When Deterministic Latency Measurement parameter is turned on, the internal module handles this generation.</p>

Figure 48. Transmitting Data Using the PCS66 TX Interface

The figure shows how to write the 66b blocks directly to the TX PCS in FlexE and OTN mode using the PCS66 TX Interface.



TX data is written as 66b blocks. The blocks are expected to be 66b encoded, with the sync header bits in the rightmost bit positions (bits 1 and 0).

- In FlexE mode, the PCS scrambles and stripes the blocks for transmission.
- In OTN mode, the PCS only stripes the blocks for transmission. The input data is expected to be already scrambled.

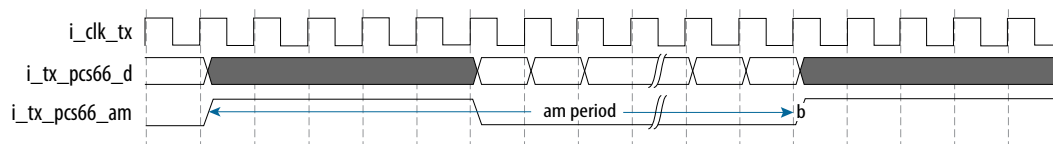
`i_tx_pcs66_valid` should conform to these conditions:

- Assert the valid signal only when the ready signal is asserted, and deassert only when the ready signal is deasserted.
- The two signals can be spaced by a fixed latency between 1 and 6 cycles.
- When the valid signal deasserts, `i_tx_pcs66_d` must be paused.

The block order for the PCS66 mode TX interface is the same as the TX PCS interface. Blocks are transmitted from LSB to MSB; the first block to be transmitted from the interface is `i_tx_pcs66_d[65:0]`.

The bit order for the PCS66 mode TX interface is the same as the TX PCS interface. Bits are transmitted from least to most significant; the first bit to be transmitted from the interface is `i_tx_pcs66_d[0]`.

Figure 49. Inserting Alignment Markers



When the PCS66 TX interface is used for FlexE mode, the timing of alignment marker insertion can be controlled from the fabric. The same operations can be performed on single lane variants, with slight variance:

- For 40GE/50GE/100GE/200GE/400GE channels and 25GE channel with RS-FEC, the signal causes the alignment markers to be inserted.
- For 10GE/25GE channels with no RS-FEC or Firecode, the signal causes the cycle to be treated as invalid for PCS processing (no changes to scramble).

In FlexE mode, the timing of alignment marker insertion is very rigid. Alignment markers cannot be delayed without disrupting the Ethernet link. Use valid cycles to count the alignment markers. When `i_tx_pcs66_valid` is low, the alignment marker counters and input must freeze.

OTN streams are expected to include their own alignment markers. In OTN mode with FEC, you must assert `i_tx_pcs66_am` to indicate the position of the alignment markers. In OTN mode without FEC, `i_tx_pcs66_am` is optional and you can tie the signal low.

In FEC modes, the TX datapath does not come completely out of reset until at least 2 alignment marker periods passed. You must start driving `i_tx_pcs66_am` at the proper interval before `o_tx_lanes_stable` goes high. You may drive the signal as soon as `o_tx_pll_locked` has gone high and `o_tx_pcs66_ready` starts toggling. When the external custom rate interface is enabled, you must start driving `i_custom_cadence`. For more information, refer to the [Custom Rate Interface](#) on page 131.

Related Information

[Custom Rate Interface](#) on page 131

7.10. FlexE and OTN Mode RX Interface

The F-Tile Ethernet Intel FPGA Hard IP RX client interface in FlexE and OTN variations employs the PCS66 interface protocol.

The FlexE and 200GE/400GE OTN modes, the application allows to read 66b blocks from the RX PCS, bypassing the RX MAC.

The RX PCS acts as a source and the client acts as a sink in the receive direction.

Additionally, the OTN mode enables the collection of RX MAC statistics.

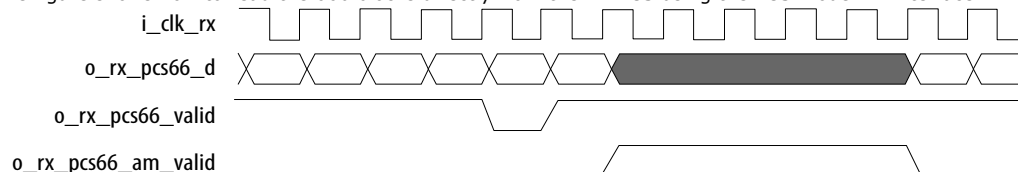
Table 54. Signals of the PCS66 RX Interface

All interface signals are clocked by the RX clock. The signal names are standard Avalon-ST signals.

Name	Width	Description
o_rx_pcs66_d[1055:0] o_rx_pcs66_d[527:0] o_rx_pcs66_d[263:0] o_rx_pcs66_d[131:0] o_rx_pcs66_d[65:0]	1056 bits (400GE) 528 bits (200GE) 264 bits (100GE) 132 bits (40GE/50GE) 66 bits(10GE/25GE)	RX PCS 66b data for 1 block. <ul style="list-style-type: none"> In FlexE mode, the RX PCS 66b data is aligned and descrambled but not decoded. In OTN mode, the RX PCS 66b data is aligned and descrambled in 200GE/400GE.
o_rx_pcs66_valid	1 bit	When asserted, indicates that the RX PCS 66b data is valid.
o_rx_pcs66_am_valid	1 bit	Alignment marker indicator. When asserted, Indicates the blocks on the RX PCS 66b data signal are identified as RS-FEC codeword markers. <i>Note:</i> This signal cannot be driven properly for 25GE with RS-FEC variant. When the Deterministic latency Interface or Deterministic latency Measurement parameter is turned on, sync pulses are seen even for 25GE without RS-FEC variant and 10GE variant. When Deterministic latency Measurement parameter is turned on, the internal module handles this generation.

Figure 50. Receiving Data Using the PCS66 RX Interface

The figure shows how to read the 66b blocks directly from the RX PCS using the PCS mode RX Interface.



The 66b blocks follow Ethernet 64b/66b convention. The least significant 2 bits of each 66 block is a 2b sync header and the remaining 64b are data.

- In FlexE mode, the data is aligned and descrambled..
- In OTN mode, the data is aligned and descrambled in 200GE/400GE.

The data is only valid when o_rx_pcs66_valid is high. The contents of the o_rx_pcs66_d bus are not defined when o_rx_pcs66_valid is low.

The block order for the PCS66 mode RX interface is the same as the RX PCS interface. Blocks flow from LSB to MSB; the first block that the core receives is o_rx_pcs66_d[65:0].

The bit order for the PCS66 mode RX interface is the same as the RX PCS interface. Bits flow from least to most significant; the first bit that the core receives is o_rx_pcs66_d[0].

`o_rx_pcs66_am_valid` indicates the arrival of the alignment markers from the RX PCS. The alignment markers also depend on `o_rx_pcs66_valid`. When `o_rx_pcs66_valid` is low, `o_rx_pcs66_am_valid` is not valid.

- In FlexE mode, when `o_rx_pcs66_am_valid` is high, `o_rx_pcs66_d` is undefined because the alignment markers do not get descrambled.
- In OTN mode, when `o_rx_pcs66_am_valid` is high, `o_rx_pcs66_d` presents the received alignment markers.

Figure 51. Receiving Alignment Markers for FlexE Mode

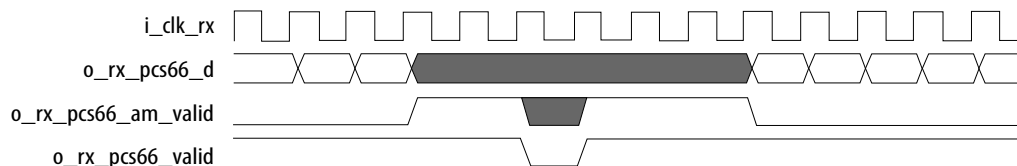
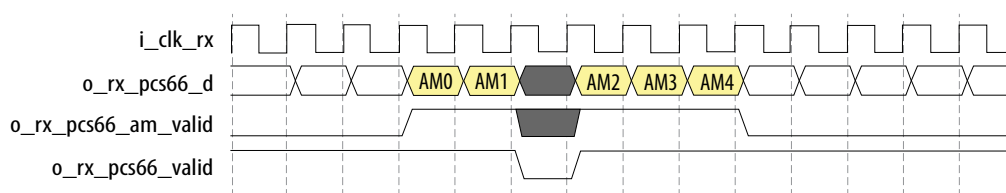


Figure 52. Receiving Alignment Markers for OTN Mode



7.11. Custom Rate Interface

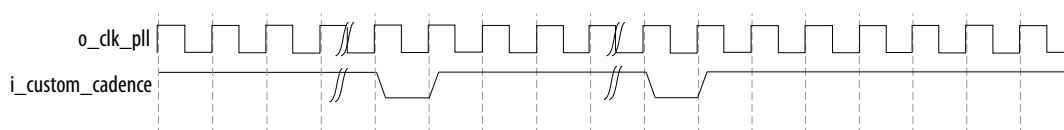
The F-Tile Ethernet Intel FPGA Hard IP Custom Rate Interface is available when you enable **Use external custom cadence controller** option. The interface accounts for differences between the system clock and the TX transceiver PLL rate.

Table 55. Signals of the Custom Rate Interface

All of the Custom Rate Interface signals except the `i_custom_cadence` signal are asynchronous.

Signal Name	Width	Description
<code>i_custom_cadence</code>	1	<p>Custom data valid signal.</p> <ul style="list-style-type: none"> • 1'b1: Set cadence data valid high for this cycle • 1'b0: Set cadence data valid low for this cycle <p>Connect this signal either to a counter that produces a steady data valid cadence that corresponds to the ratio between the clock rate used and the clock rate required, or a system that increases or decreases the data valid cadence based on the current occupancy of transceiver TX FIFO or an external TX FIFO.</p> <p><i>Note:</i> The TX reset sequence requires valid custom cadence pulses. You must start driving <code>i_custom_cadence</code> before <code>o_tx_lanes_stable</code> assert. You may drive <code>i_custom_cadence</code> as soon as <code>o_tx_pll_locked</code> asserts, and any external cadence generation logic and clocks are out of reset.</p>

Figure 53. Custom rate Interface Behavior with a Fixed Data Valid Ratio



A counter producing a steady ratio of high and low pulses to balance the flow through a channel can drive the custom cadence rate interface.

7.12. 32-bit Soft CWBIN Counters

The F-Tile Ethernet Intel FPGA Hard IP 32-bit Soft CWBIN Counters support all the FEC mode options available in the IP Parameter editor. This soft logic converts 8-bit CWBin0-3 registers in Ethernet Hard IP to 32-bit registers in soft logic. When the FEC codeword receives zero errors, the CWBin0 register counts up; When the FEC codeword receives 1 error, the CWBin1 register counts up, and so on.

Only the CWBin0-3 counters are expected to increase at a high rate. As a result, this Soft logic performs fast reads of these 8-bit counter values from Hard IP and accumulates these counts to 32-bit counter registers that you can access.

The F-Tile Ethernet Intel FPGA Hard IP read rate is set to 6.5µs for 200/400G and 13µs for remaining rates to prevent the 8-bit counts from overflowing.

7.13. Reconfiguration Interfaces

Both Ethernet reconfiguration interfaces and Transceiver reconfiguration interfaces have a `readdata_valid` port, which makes them compatible with pipelined read bus master. Enabling it has no effect on the throughput.

For more information, refer to the *Pipelined Transfers of Avalon Interface Specifications* section.

Related Information

[Pipelined Transfers](#)

7.13.1. Ethernet Reconfiguration Interfaces

You access Ethernet control and status registers of the F-Tile Ethernet Intel FPGA Hard IP during normal operation using an Avalon memory-mapped interface. The interface responds regardless of the link status. It also responds when the IP core is in a reset state driven by any reset signal or soft reset other than the `i_reconfig_reset` signal.

Note: The Avalon memory-mapped interfaces are word-addressed. The address always aligns to 32-bit words. All register access described in this user guide is byte-based access. You need to convert each word address to a byte address by shifting right by two (dividing by 4). To access individual bytes, use the byte enable signals.

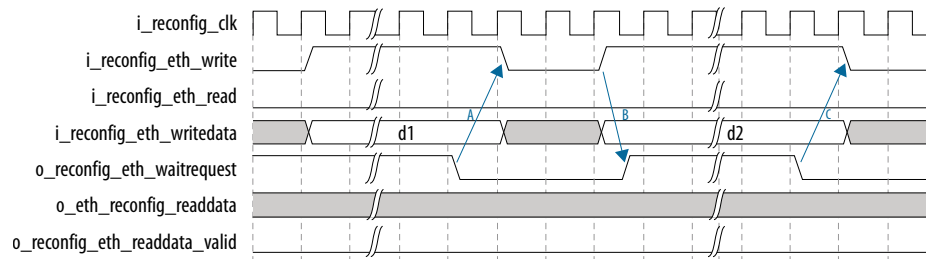
Table 56. Ethernet Reconfiguration Interface

The signals in this interface are clocked by the `i_reconfig_clk` clock and reset by the `i_reconfig_reset` signal. This clock and reset are used for all the reconfiguration interfaces in the IP core.

Port Name	Width	Description
<code>i_reconfig_eth_addr[13:0]</code>	14 bits	Address bus for Ethernet control and status registers.
<code>i_reconfig_eth_read</code>	1 bit	Read request signal for Ethernet control and status registers.
<i>continued...</i>		

Port Name	Width	Description
i_reconfig_eth_write	1 bit	Write request signal for Ethernet control and status registers.
i_reconfig_eth_byteenable[3:0]	4 bits	Byte enable for Ethernet read and write request signals.
o_reconfig_eth_readdata[31:0]	32 bits	Read data from reads to Ethernet control and status registers.
o_reconfig_eth_readdata_valid	1 bit	Read data from Ethernet control and status registers is valid.
i_reconfig_eth_writedata[31:0]	32 bits	Write data for Ethernet control and status registers.
o_reconfig_eth_waitrequest	1 bit	Avalon memory-mapped interface stalling signal for operations on Ethernet control and status registers.

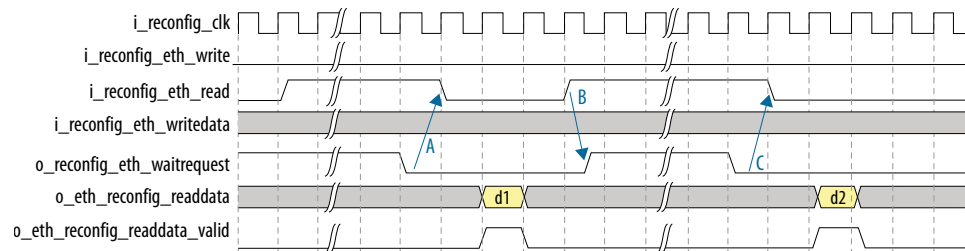
Figure 54. Performing a Avalon Memory-Mapped Interface Write on the Ethernet Reconfiguration Interface



The figure above shows how to write data using the Ethernet reconfiguration Avalon memory-mapped interface.

- When write begins while `o_reconfig_eth_waitrequest` is high, you must hold the write request (`i_reconfig_eth_write`) until `o_reconfig_eth_waitrequest` deasserts.
 - A: When `o_reconfig_eth_waitrequest` deasserts, also deassert write on the next cycle.
- When write begins while `o_reconfig_eth_waitrequest` is low:
 - B: The `o_reconfig_eth_waitrequest` signal asserts on the same clock cycle.
 - The write request holds until `o_reconfig_eth_waitrequest` deasserts.
 - C: On the next clock cycle after `o_reconfig_eth_waitrequest` deasserts, the write request (`i_reconfig_eth_write`) deasserts.
- Write request can take a variable amount of time to complete.
- You cannot perform read and write requests at the same time.
- When multiple configuration bits are at the same address, a you need to perform a Read-Modify-Write operation to change the desired bits without changing the remaining configurations at the same location.

Figure 55. Performing a Avalon Memory-Mapped Interface Read on the Ethernet Reconfiguration Interface



The figure above shows how to read data using the Ethernet reconfiguration Avalon memory-mapped interface.

- — A: When read begins while `o_reconfig_eth_waitrequest` is high, you must hold the read request (`i_reconfig_eth_read`) until `o_reconfig_eth_waitrequest` deasserts.
Then, the requested read data is available on the read port on the cycle when `o_reconfig_eth_readdata_valid` is high.
- When read begins while `o_reconfig_eth_waitrequest` is low:
 - B: The `o_reconfig_eth_waitrequest` signal asserts on the same clock cycle as the read request (`i_reconfig_eth_read`).
 - C: The read request holds until `o_reconfig_eth_waitrequest` deasserts.
Then, the requested read data is available on the read cycle when `o_reconfig_eth_readdata_valid` is high.
- Read request can take a variable amount of time to complete.
- You cannot perform read and write requests at the same time.
- The Avalon memory-mapped interface processes one read request at a time.

Related Information

[F-Tile Architecture and PMA and FEC Direct PHY IP User Guide](#)

Provides more information about the transceiver reconfiguration interface in F-tile devices, including timing diagrams for reads and writes.

7.13.2. Transceiver Reconfiguration Interfaces

You access the control and status registers of the Intel Agilex 7 F-tile transceivers during normal operation using an Avalon memory-mapped interface. The interface responds regardless of the link status. It also responds when the IP core is in a reset state driven by any reset signal or soft reset other than the `i_reconfig_reset` signal.

Asserting the `i_reconfig_reset` signal resets all transceiver reconfiguration control and status registers, including the statistics counters; while this reset is in process, reads and writes to addresses in the F-Tile Ethernet Intel FPGA Hard IP is delayed.

Table 57. Transceiver Reconfiguration Interface Ports to Native PHY Reconfiguration Interfaces

The signals in this interface are clocked by the `i_reconfig_clk` clock and reset by the `i_reconfig_reset` signal. The signal names are standard Avalon memory-mapped interface signals.

- n = index of transceiver associated with the reconfiguration interface, from 0 to (number of lanes - 1)

Port Name	Width	Description
<code>i_reconfig_xcvrn_addr[17:0]</code>	18 bits	Address bus for transceiver control and status registers.
<code>i_reconfig_xcvrn_read</code>	1 bit	Transceiver read signal. When asserted, starts a read cycle.
<code>i_reconfig_xcvrn_write</code>	1 bit	Transceiver write signal. When asserted, writes data on the reconfiguration write data bus.
<code>i_reconfig_xcvrn_byteenable[3:0]</code>	4 bits	Transceiver byte enable signal for read and write request.
<code>o_reconfig_xcvrn_readdata[31:0]</code>	32 bits	Transceiver read data bus. When asserted, presents transceiver data read on a read cycle.
<code>o_reconfig_xcvrn_readdata_valid</code>	1 bit	Read data from Transceiver read data bus is valid.
<code>i_reconfig_xcvrn_writedata[31:0]</code>	32 bits	Transceiver write data bus. When asserted, presents transceiver data written on a write cycle.
<code>o_reconfig_xcvrn_waitrequest</code>	1 bit	Indicates the Avalon memory-mapped interface is busy. The read or write cycle is only complete when this signal goes low.

Related Information

F-Tile Architecture and PMA and FEC Direct PHY IP User Guide

Provides more information about the transceiver reconfiguration interface in F-tile devices, including timing diagrams for reads and writes.

7.14. Precision Time Protocol Interface

The Precision Time Protocol (PTP) interface is available when you enable **Enable IEEE 1588 PTP** option in the **PTP** tab. When selected, the IP generates PTP based 1-step or 2-step TX and RX timestamps. The IP requires the IEEE 1588 96-bit time-of-day (TOD) input.

Table 58. PTP Clock Ports

Signal Name	Width	Description
<code>i_clk_tx_tod</code>	1	TX TOD input clock. The clock frequencies vary based on the Ethernet rate and the selected Timestamp accuracy mode . In Basic timestamp accuracy mode, connect this signal to any clock with below frequency. <ul style="list-style-type: none"> 10GE: 390.625 MHz 25GE/50GE/100GE/200GE/400GE: 390.625 MHz In Advanced timestamp accuracy mode, connect this signal to the <code>o_clk_tx_div</code> clock.

continued...

Signal Name	Width	Description
		<ul style="list-style-type: none"> 10GE: 156.25 MHz 25GE/50GE/100GE/200GE/400GE: 390.625 MHz
i_clk_rx_tod	1	<p>RX TOD input clock. The clock frequencies vary based on the Ethernet rate and the selected Timestamp accuracy mode.</p> <p>In Basic timestamp accuracy mode, connect this signal to any clock with below frequency.</p> <ul style="list-style-type: none"> 10GE: 390.625 MHz 25GE/50GE/100GE/200GE/400GE: 390.625 MHz <p>In Advanced timestamp accuracy mode, connect this signal to the o_clk_rec_div clock.</p> <ul style="list-style-type: none"> 10GE: 156.25 MHz 25GE/50GE/100GE/200GE/400GE: 390.625 MHz
i_clk_ptp_sample	1	<p>Sample clock for the PTP measurement. The clock frequency is 114.2857 MHz.</p> <p>The same sample clock feeds the PTP logics of designs instantiating multiple F-Tile Ethernet Intel FPGA Hard IPs.</p>

7.14.1. Time-of-Day Interface

The shared time-of-day (TOD) allows the IP core to reference all timestamps to the local time-of-day. The 96-bit timestamps are in IEEE 1588 v2 format.

Table 59. Signals of the Time-of-Day (TOD) Interface

The time-of-day interface allows the IP core to reference all of its timestamps to the time of day as it is known locally.

All interface signals are clocked by the i_clk_tx_tod clock.

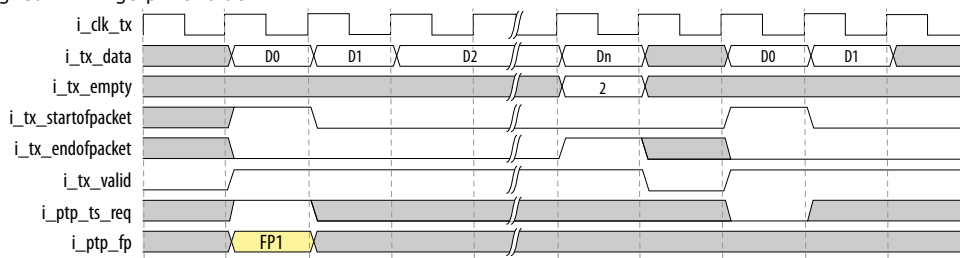
Signal Name	Width	Description
i_ptp_tx_tod[95:0]	96	<p>Time-of-day (TOD), according to the local clock for the TX clock domain. The timestamp is in IEEE 1588 v2 format.</p> <ul style="list-style-type: none"> [95:48]: represent seconds [47:16]: represent nanoseconds [15:0]: represent fractional nanoseconds
i_ptp_tx_tod_valid	1	<p>Indicates the TX TOD signal is valid.</p> <ul style="list-style-type: none"> You assert the signal when i_ptp_tx_tod bus contains a valid time. You deassert the signal for at least one clock cycle to indicate a significant change in the i_ptp_tx_tod value such as TOD reset or first TOD adjustment after system power up. The IP core deasserts o_tx_ptp_ready to indicate the TX egress timestamp is not valid.
i_ptp_rx_tod[95:0]	96	<p>Time-of-day, according to the local clock for the RX clock domain. The timestamp is in IEEE 1588 v2 format (96 bits).</p> <ul style="list-style-type: none"> [95:48]: represent seconds [47:16]: represent nanoseconds [15:0]: represent fractional nanoseconds
i_ptp_rx_tod_valid	1	<p>Indicates the RX TOD signal is valid.</p> <ul style="list-style-type: none"> You assert the signal when i_ptp_rx_tod bus contains a valid time. You deassert the signal for at least one clock cycle to indicate a significant change in the i_ptp_rx_tod value such as TOD reset or first TOD adjustment after system power up. The IP core deasserts o_rx_ptp_ready to indicate the RX egress timestamp is not valid.

7.14.2. TX 2-Step Timestamp Interface

The timestamp interface signals are valid when `i_tx_valid` and `i_tx_startofpacket` signals are high. To initiate a timestamp request, the `i_ptp_ts_req` signal must be high. You also must specify a fingerprint value in the `i_ptp_fp` signal. The fingerprint values are re-usable once the IP core returns a packet with the timestamp.

Figure 56. Using the IEEE 1588 TX 2-Step Timestamp Interface

The figure depicts transmission of two packets. The first packet contains a 2-step timestamp request with assigned FP1 fingerprint value.



When the IP core returns the timestamp for a packet with a 2-step timestamp request, it asserts the `o_ptp_ets_valid` signal to indicate that the `o_ptp_ets` and `o_ptp_ets_fp` signals contain the timestamp information. The `o_ptp_ets` contains the timestamp and the `o_ptp_ets_fp` signal indicates the specific packet the timestamp is for. The IP processes the timestamp requests in the order they were received.

Figure 57. 2-Step Timestamp: Receiving Packet with Timestamp

The figure shows that the IP received the timestamp for packet FP1 and then sometime later, IP returns the timestamp for packet FP2.

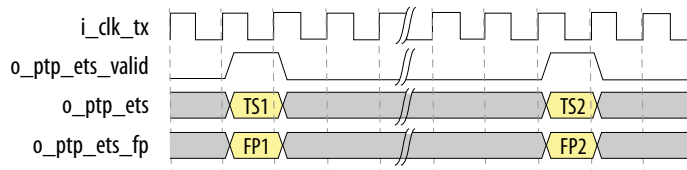


Table 60. Signals of the 2-Step TX Timestamp Interface

The timestamp is always in 1588 v2 format.

All interface signals are synchronous and clocked by the `i_clk_tx` clock.

Signal Name	Width	Description
<code>i_ptp_ts_req</code> <code>i_ptp_ts_req[1:0]</code>	1 bit (10GE/25GE/50GE/ 100GE/200GE) 2 bits (400GE)	Request a 2-step timestamp signal for the current TX packet when Ethernet rate is 10GE/25GE/50GE/ 100GE/200GE/400GE.
<i>continued...</i>		

Signal Name	Width	Description
		<ul style="list-style-type: none"> For TX MAC Avalon ST client interface, <code>i_ptp_ts_req[0]</code> is valid only when both, <code>i_tx_valid</code> and <code>i_tx_startofpacket</code> signals, are equal to 1. For TX MAC segmented client interface, <code>i_ptp_ts_req[0]</code> is valid only when <code>i_tx_mac_valid</code> is equal to 1 and <code>i_tx_mac_inframe[7:0]</code> contains the start of the packet (SOP). The <code>i_ptp_ts_req[1]</code> is valid only when <code>i_tx_mac_valid</code> is equal to 1 and <code>i_tx_mac_inframe[15:8]</code> contains the SOP.
<code>i_ptp_fp[w]</code> , where <code>w</code> represents timestamp fingerprint width <code>i_ptp_fp[w*2]</code> , where <code>w</code> is 8-12	<code>w</code> bit (10GE/25GE/50GE/ 100GE/200GE) (<code>w*2</code>) bits (400GE)	<p>Fingerprint signal for current TX packet when Ethernet rate is 10GE/ 25GE/50GE/100GE/200GE/ 400GE.</p> <p>Assigns an 8 bit fingerprint to a TX packet that is being transmitted so the returned TX egress timestamp can be identified with respective TX packet.</p> <ul style="list-style-type: none"> For TX MAC Avalon ST client interface, <code>i_ptp_fp[(w-1):0]</code> is valid only when both, <code>i_tx_valid</code> and <code>i_tx_startofpacket</code> signals, are equal to 1. For TX MAC segmented client interface, <code>i_ptp_fp[(w-1):0]</code> is valid only when <code>i_tx_mac_valid</code> is equal to 1 and <code>i_tx_mac_inframe[7:0]</code> contains the start of the packet (SOP). The <code>i_ptp_fp[w*2-1:w]</code> is valid only when <code>i_tx_mac_valid</code> is equal to 1 and <code>i_tx_mac_inframe[15:8]</code> contains the SOP. <p>Used for both 2-step and 1-step PTP packets. Intel recommends to use large range of fingerprints to avoid assigning same fingerprint to two currently processed TX packets.</p>
<code>o_ptp_ets_valid</code> <code>o_ptp_ets_valid[1:0]</code>	1 bit (10GE/25GE/50GE/ 100GE/200GE) 2 bits (400GE)	TX egress timestamp valid signal when Ethernet rate is 10GE/ 25GE/50GE/100GE/200GE/400GE.
<code>o_ptp_ets[95:0]</code> <code>o_ptp_ets[191:0]</code>	96 bits (10GE/25GE/ 50GE/100GE/200GE) 192 bits (400GE)	<p>TX egress timestamp bus when Ethernet rate is 10GE/ 25GE/50GE/100GE/200GE/400GE.</p> <p>The bus presents an egress timestamp for the TX packet transmitted with the fingerprint given by <code>o_ptp_ets_fp</code> signal.</p> <ul style="list-style-type: none"> <code>o_ptp_ets[95:0]</code> is valid only when <code>o_ptp_ets_valid[0]=1</code>. <code>o_ptp_ets[191:96]</code> is valid only when <code>o_ptp_ets_valid[1]=1</code>. <p>The egress timestamp applies for 2-step and 1-step PTP packets.</p>
<code>o_ptp_ets_fp[w]</code> <code>o_ptp_ets_fp[w*2]</code> , where <code>w</code> is 8-12	<code>w</code> bits (10GE/25GE/ 50GE/100GE/200GE) (<code>w*2</code>) bits (400GE)	<p>TX egress timestamp fingerprint when Ethernet rate is 10GE/25GE/ 50GE/100GE/200GE/400GE.</p> <p>The bus presents the fingerprint for the current 2-step egress timestamp. You use the fingerprint to determine which TX packet the timestamp belongs to.</p>
continued...		

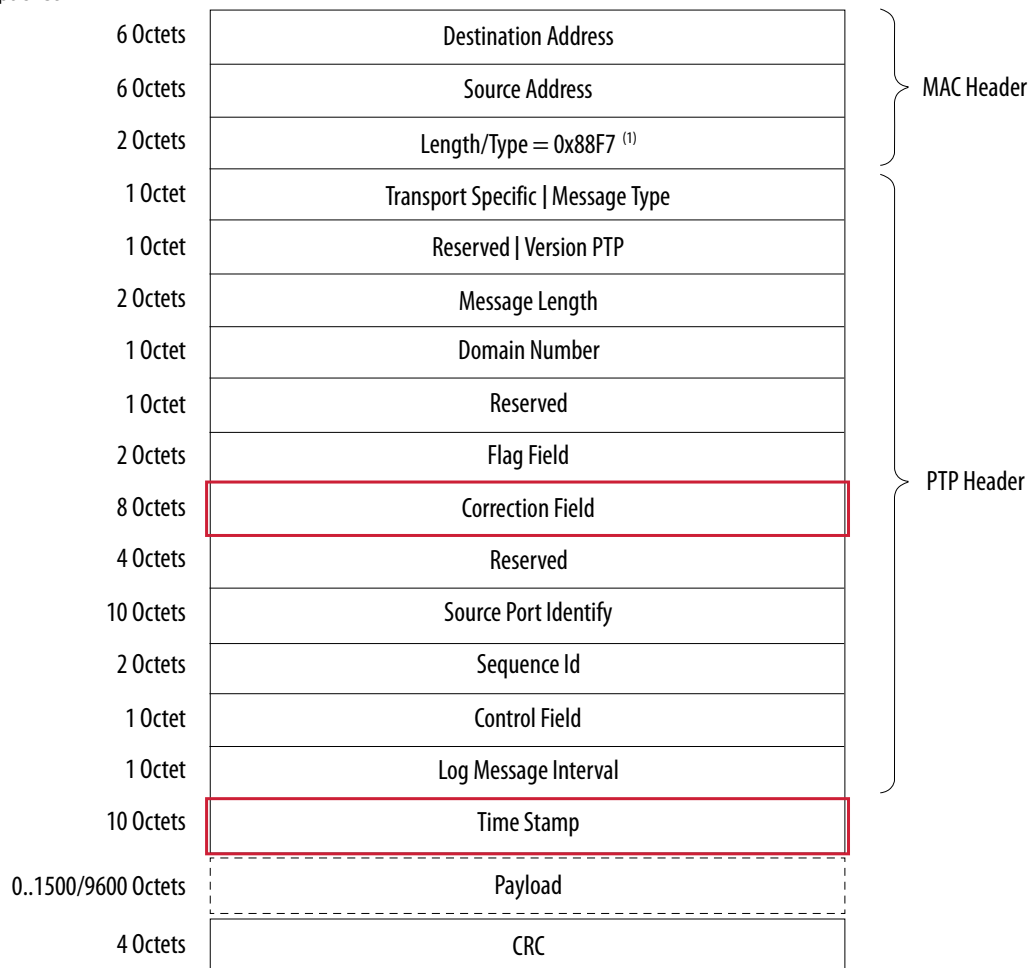
Signal Name	Width	Description
		<ul style="list-style-type: none"> o_ptp_ets_fp[w-1:0] is valid only when o_ptp_ets_valid[0] = 1. o_ptp_ets_fp[w*2-1:w] is valid only when o_ptp_ets_valid[1] = 1. <p>The egress timestamp applies for 2-step and 1-step PTP packets.</p>

7.14.3. TX 1-Step Timestamp Interface

When transmitting a 1-step PTP packet, the IP core modifies the PTP field on-the-fly. The IP core can change the UDP checksum or add extension bytes to adjust it. PTP headers can be part of the VLAN packets and can be encapsulated by UDP datagrams inside of the IPv4 and IPv6 packets.

Figure 58. Example of Ethernet Packet with a PTP Header

The figure depicts an Ethernet packet with PTP header embedded as an Ethernet payload in a simple non-VLAN packet.



The figure below shows how the PTP 1-step interface writes a 1-step timestamp to a PTP header. The timestamp signals are valid when `i_tx_valid` and `i_tx_startofpacket` signals are high. To initiate a timestamp request, the `i_ptp_ins_ets` signal must be high.

The PTP header must provide the offset of the starting octet of the timestamp field, compared to the first byte of the frame, as the `i_ptp_ts_offset` signal.

- In above diagram, first byte of timestamp field is 49th byte, the byte offset (`i_ptp_ts_offset`) is 48.
- The timestamp must not overwrite other fields such as headers or packet CRCs.

Figure 59. Using the IEEE 1588 TX 1-Step Timestamp Interface

The figure depicts writing a 1-step timestamp to a PTP header. The actual timestamp offset for this packet is 48 octets. Offset of the correction field starting octet is 22 octets.]

Note:

The byte offset (`i_ptp_ts_offset` and `i_ptp_cf_offset`) values are decimal.

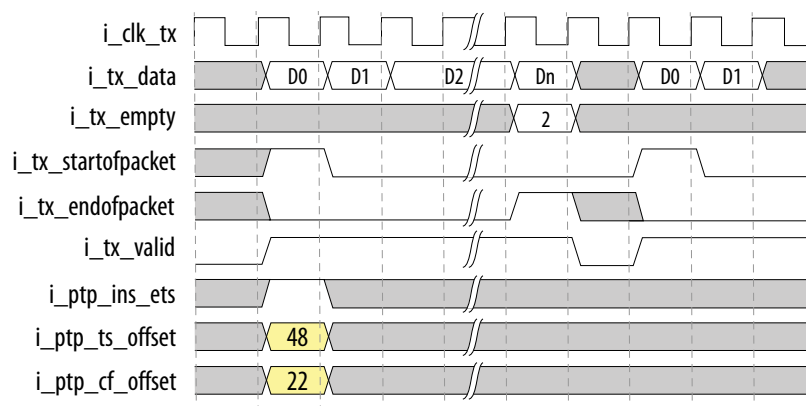


Table 61. Signals of the 1-Step TX Timestamp Interface

All interface signals are clocked by `i_clk_tx` clock. The timestamp is always in 1588 v2 format.

Signal Name	Width	Description
<code>i_ptp_ins_ets</code> <code>i_ptp_ins_ets[1:0]</code>	1 bit (10GE/25GE, 50GE/100GE/200GE) 2 bits (400GE)	<p>Insert an egress timestamp signal for the current TX packet when Ethernet rate is 10GE/25GE/50GE/100GE/200GE/400GE.</p> <ul style="list-style-type: none"> • For TX MAC Avalon ST client interface, <code>i_ptp_ins_ets[0]</code> is valid only when both TX valid (<code>i_tx_valid</code>) and TX SOP (<code>i_tx_startofpacket</code>) signals are asserted. • For TX MAC segmented client interface, <code>i_ptp_ins_ets[0]</code> is valid only when TX valid (<code>i_tx_mac_valid</code>) is asserted and <code>i_tx_mac_inframe[7:0]</code> contains the start of the packet (SOP). <p>The <code>i_ptp_ins_ets[1]</code> is valid only when TX valid (<code>i_tx_mac_valid</code>) is asserted and <code>i_tx_mac_inframe[15:8]</code> contains the SOP.</p> <ul style="list-style-type: none"> • Must not use when <code>i_tx_skip_crc</code> = 1. To recalculate the CRC for the TX packet, the TX skip CRC signal must be 0.

continued...

Signal Name	Width	Description
		<ul style="list-style-type: none"> Must not use when <code>i_ptp_ins_cf = 1</code>. You cannot update residence time and insert an egress timestamp on the same packet. Set the PTP timestamp field in the TX packet (<code>i_ptp_ts_offset</code>) to the byte position of the start of the timestamp field in the PTP header. Set the PTP correction field in the TX packet (<code>i_ptp_cf_offset</code>) to the byte position of the start of the correction field in the PTP header.
<code>i_ptp_ins_cf</code> <code>i_ptp_ins_cf[1:0]</code>	1 bit (10GE/25GE,50GE/100GE/200GE) 2 bits (400GE)	<p>Update the correction field with residence time in the current TX packet when Ethernet rate is 10GE/25GE/50GE/100GE/200GE/400GE.</p> <ul style="list-style-type: none"> For TX MAC Avalon ST client interface, <code>i_ptp_ins_cf[0]</code> is valid only when both TX valid (<code>i_tx_valid</code>) and TX SOP (<code>i_tx_startofpacket</code>) signals are asserted. For TX MAC segmented client interface, <code>i_ptp_ins_cf[0]</code> is valid only when TX valid (<code>i_tx_mac_valid</code>) is asserted and <code>i_tx_mac_inframe[7:0]</code> contains the start of the packet (SOP). The <code>i_ptp_ins_cf[1]</code> is valid only when TX valid (<code>i_tx_mac_valid</code>) is asserted and <code>i_tx_mac_inframe[15:8]</code> contains the SOP. Must not use when <code>i_tx_skip_crc = 1</code>. To recalculate the CRC for the TX packet, the TX skip CRC signal must be 0. Do not use when the egress timestamp signal (<code>i_ptp_ins_ets</code>) is asserted. You cannot update residence time and insert an egress timestamp on the same packet. To calculate the residence time, use the <code>i_ptp_tx_its</code> signal to provide IP core with the ingress timestamp of the current packet when it entered the system. Set the position of the PTP correction field in the TX packet (<code>i_ptp_cf_offset</code>) to the byte position of the start of the correction field in the PTP header.
<code>i_ptp_zero_csum</code> <code>i_ptp_zero_csum[1:0]</code>	1 bit (10GE/25GE,50GE/100GE/200GE) 2 bits (400GE)	<p>Overwrites the UDP checksum with zeros in the current TX packet when Ethernet rate is 10GE/25GE/50GE/100GE/200GE/400GE.</p>
continued...		

Signal Name	Width	Description
		<ul style="list-style-type: none"> For TX MAC Avalon ST client interface, <code>i_ptp_zero_csum[0]</code> is valid only when both TX valid (<code>i_tx_valid</code>) and TX SOP (<code>i_tx_startofpacket</code>) signals are asserted. For TX MAC segmented client interface, <code>i_ptp_zero_csum[0]</code> is valid only when TX valid (<code>i_tx_mac_valid</code>) is asserted and <code>i_tx_mac_inframe[7:0]</code> contains the start of the packet (SOP). The <code>i_ptp_zero_csum[1]</code> is valid only when TX valid (<code>i_tx_mac_valid</code>) is asserted and <code>i_tx_mac_inframe[15:8]</code> contains the SOP. Must not use when <code>i_tx_skip_crc = 1</code>. To recalculate the CRC for the TX packet, the TX skip CRC signal must be 0. Do not use when the update extended bytes field signal (<code>i_ptp_update_eb</code>) is asserted. You cannot set a UDP checksum to 0, and update an extension field to cancel out checksum changes on the same packet. Set the position of the UDP checksum field in the TX packet (<code>i_ptp_csum_offset</code>) to the byte position of the start of the UDP checksum in the TX packet.
<code>i_ptp_update_eb</code> <code>i_ptp_update_eb[1:0]</code>	1 bit (10GE/25GE,50GE/ 100GE/200GE) 2 bits (400GE)	Update the extended bytes field to cancel out checksum changes in the current TX packet when Ethernet rate is 10GE/25GE/50GE/100GE/200GE/400GE. When asserted, overwrites the extended bytes field in an IPv6 packet carried inside the current TX packet with a value that cancels out changes to the checksum due to changes to the UDP packet. <ul style="list-style-type: none"> For TX MAC Avalon ST client interface, <code>i_ptp_update_eb[0]</code> is valid only when both TX valid (<code>i_tx_valid</code>) and TX SOP (<code>i_tx_startofpacket</code>) signals are asserted. For TX MAC segmented client interface, <code>i_ptp_update_eb[0]</code> is valid only when TX valid (<code>i_tx_mac_valid</code>) is asserted and <code>i_tx_mac_inframe[7:0]</code> contains the start of the packet (SOP). The <code>i_ptp_update_eb[1]</code> is valid only when TX valid (<code>i_tx_mac_valid</code>) is asserted and <code>i_tx_mac_inframe[15:8]</code> contains the SOP. Must not use when <code>i_tx_skip_crc = 1</code>. To recalculate the CRC for the TX packet, the TX skip CRC signal must be 0. Do not use when the overwrite a UDP checksum with zeros signal (<code>i_ptp_zero_csum</code>) is asserted. You cannot set a UDP checksum to 0, and update an extension field to cancel out checksum changes on the same packet. You do not have to provide the byte position of the start of the UDP extension field in the TX packet. The IP core assumes that the byte position starts two bytes before the CRC field.
<code>i_ptp_p2p</code> <code>i_ptp_p2p[1:0]</code>	1 bit (10GE/25GE,50GE/ 100GE/200GE) 2 bits (400GE)	Add peer-to-peer mean path delay to correction field for the current TX packet when Ethernet rate is 10GE/25GE/50GE/100GE/200GE/400GE.
continued...		

Signal Name	Width	Description
		<ul style="list-style-type: none"> For TX MAC Avalon ST client interface, <code>i_ptp_p2p[0]</code> is valid only when both TX valid (<code>i_tx_valid</code>) and TX SOP (<code>i_tx_startofpacket</code>) signals are asserted. For TX MAC segmented client interface, <code>i_ptp_p2p[0]</code> is valid only when TX valid (<code>i_tx_mac_valid</code>) is asserted and <code>i_tx_mac_inframe[7:0]</code> contains the start of the packet (SOP). The <code>i_ptp_p2p[1]</code> is valid only when TX valid (<code>i_tx_mac_valid</code>) is asserted and <code>i_tx_mac_inframe[15:8]</code> contains the SOP. Must not use when <code>i_tx_skip_crc</code> = 1. To recalculate the CRC for the TX packet, the TX skip CRC signal must be 0.
<code>i_ptp_asym</code> <code>i_ptp_asym[1:0]</code>	1 bit (10GE/25GE,50GE/ 100GE/200GE) 2 bits (400GE)	<p>Add asymmetric latency to the correction field for the current TX packet when Ethernet rate is 10GE/25GE/50GE/100GE/200GE/400GE.</p> <ul style="list-style-type: none"> For TX MAC Avalon ST client interface, <code>i_ptp_asym[0]</code> is valid only when both TX valid (<code>i_tx_valid</code>) and TX SOP (<code>i_tx_startofpacket</code>) signals are asserted. For TX MAC segmented client interface, <code>i_ptp_asym[0]</code> is valid only when TX valid (<code>i_tx_mac_valid</code>) is asserted and <code>i_tx_mac_inframe[7:0]</code> contains the start of the packet (SOP). The <code>i_ptp_asym[1]</code> is valid only when TX valid (<code>i_tx_mac_valid</code>) is asserted and <code>i_tx_mac_inframe[15:8]</code> contains the SOP. Must not use when <code>i_tx_skip_crc</code> = 1. To recalculate the CRC for the TX packet, the TX skip CRC signal must be 0.
<code>i_ptp_asym_sign</code> <code>i_ptp_asym_sign[1:0]</code>	1 bit (10GE/25GE,50GE/ 100GE/200GE) 2 bits (400GE)	<p>Sign of an asymmetry delay added to the correction field for the current TX packet when Ethernet rate is 10GE/25GE/50GE/100GE/200GE/400GE.</p> <ul style="list-style-type: none"> 0: Asymmetry delay is a positive value. 1: Asymmetry delay is a negative value. For TX MAC Avalon ST client interface, <code>i_ptp_asym_sign[0]</code> is valid only when both TX valid (<code>i_tx_valid</code>) and TX SOP (<code>i_tx_startofpacket</code>) signals are asserted. For TX MAC segmented client interface, <code>i_ptp_asym_sign[0]</code> is valid only when TX valid (<code>i_tx_mac_valid</code>) is asserted and <code>i_tx_mac_inframe[7:0]</code> contains the start of the packet (SOP). The <code>i_ptp_asym_sign[1]</code> is valid only when TX valid (<code>i_tx_mac_valid</code>) is asserted and <code>i_tx_mac_inframe[15:8]</code> contains the SOP.
<code>i_ptp_asym_p2p_idx[6:0]</code> <code>i_ptp_asym_p2p_idx[13:0]</code>	7 bits (10GE/ 25GE,50GE/100GE/ 200GE) 14 bits (400GE)	<p>Index of asymmetry delay and peer-to-peer mean path delay in the configurable lookup table⁽¹⁹⁾ when Ethernet rate is 10GE/25GE/50GE/100GE/200GE/400GE.</p>

continued...

Signal Name	Width	Description
		<ul style="list-style-type: none"> For TX MAC Avalon ST client interface, <code>i_ptp_asym_p2p_idx[6:0]</code> is valid only when both TX valid (<code>i_tx_valid</code>) and TX SOP (<code>i_tx_startofpacket</code>) signals are asserted. For TX MAC segmented client interface, <code>i_ptp_asym_p2p_idx[6:0]</code> is valid only when TX valid (<code>i_tx_mac_valid</code>) is asserted and <code>i_tx_mac_inframe[7:0]</code> contains the start of the packet (SOP). The <code>i_ptp_asym_p2p_idx[13:7]</code> is valid only when TX valid (<code>i_tx_mac_valid</code>) is asserted and <code>i_tx_mac_inframe[15:8]</code> contains the SOP.
<code>i_ptp_ts_offset[15:0]</code> <code>i_ptp_ts_offset[31:0]</code>	16 bits (10GE/ 25GE,50GE/100GE/ 200GE) 32 bits (400GE)	Presents the position of the PTP timestamp field in the TX packet when Ethernet rate is 10GE/25GE/50GE/100GE/200GE/400GE. <ul style="list-style-type: none"> For TX MAC Avalon ST client interface, <code>i_ptp_ts_offset[15:0]</code> is valid only when both TX valid (<code>i_tx_valid</code>) and TX SOP (<code>i_tx_startofpacket</code>) signals are asserted. For TX MAC segmented client interface, <code>i_ptp_ts_offset[15:0]</code> is valid only when TX valid (<code>i_tx_mac_valid</code>) is asserted and <code>i_tx_mac_inframe[7:0]</code> contains the start of the packet (SOP). The <code>i_ptp_ts_offset[31:16]</code> is valid only when TX valid (<code>i_tx_mac_valid</code>) is asserted and <code>i_tx_mac_inframe[15:8]</code> contains the SOP. It is the offset of the first octet of the field from the start of the frame, where the first byte of the frame (the first destination MAC address octet) is position 0. The PTP timestamp field is 10 octets long, starting from the position given by the offset. PTP adds the lower 16 bits of the 96-bit timestamp in the lower 2 octets of the correction field. Caution: You must set the offset to a position within the TX packet, or the PTP insertion operation fails. Also, you must not overlap the PTP fields.
<code>i_ptp_cf_offset[15:0]</code> <code>i_ptp_cf_offset[31:0]</code>	16 bits (10GE/ 25GE,50GE/100GE/ 200GE) 32 bits (400GE)	Presents the position of the PTP correction field in the TX packet when Ethernet rate is 10GE/25GE,50GE/100GE/200GE/400GE.
continued...		

(19) The lookup table is accessible via the PTP tile adapter module's Avalon memory-mapped interface. For more information, refer to [PTP Asymmetry Delay Reconfiguration Interface](#) on page 163 and [PTP Peer-to-Peer MeanPathDelay Reconfiguration Interface](#) on page 163.

Signal Name	Width	Description
		<ul style="list-style-type: none"> For TX MAC Avalon ST client interface, <code>i_ptp_cf_offset[15:0]</code> is valid only when both TX valid (<code>i_tx_valid</code>) and TX SOP (<code>i_tx_startofpacket</code>) signals are asserted. For TX MAC segmented client interface, <code>i_ptp_cf_offset[15:0]</code> is valid only when TX valid (<code>i_tx_mac_valid</code>) is asserted and <code>i_tx_mac_inframe[7:0]</code> contains the start of the packet (SOP). The <code>i_ptp_cf_offset[31:16]</code> is valid only when TX valid (<code>i_tx_mac_valid</code>) is asserted and <code>i_tx_mac_inframe[15:8]</code> contains the SOP. It is the offset of the first octet of the field from the start of the frame, where the first byte of the frame (the first destination MAC address octet) is position 0. The PTP correction field is 8 octets long, starting from the position given by the offset. PTP adds the lower 16 bits of the 96-bit timestamp in the lower 2 octets of the correction field.. <p>Caution: You must set the offset to a position within the TX packet, or the PTP insertion operation fails. Also, you must not overlap the PTP fields.</p>
<code>i_ptp_csum_offset[15:0]</code> <code>i_ptp_csum_offset[31:0]</code>	16 bits (10GE/ 25GE,50GE/100GE/ 200GE) 32 bits (400GE)	Position of the UDP checksum field in the TX packet when Ethernet rate is 10GE/25GE,50GE/100GE/200GE/400GE. <ul style="list-style-type: none"> For TX MAC Avalon ST client interface, <code>i_ptp_csum_offset[15:0]</code> is valid only when both TX valid (<code>i_tx_valid</code>) and TX SOP (<code>i_tx_startofpacket</code>) signals are asserted. For TX MAC segmented client interface, <code>i_ptp_csum_offset[15:0]</code> is valid only when TX valid (<code>i_tx_mac_valid</code>) is asserted and <code>i_tx_mac_inframe[7:0]</code> contains the start of the packet (SOP). The <code>i_ptp_csum_offset[31:16]</code> is valid only when TX valid (<code>i_tx_mac_valid</code>) is asserted and <code>i_tx_mac_inframe[15:8]</code> contains the SOP. It is the offset of the first octet of the field from the start of the frame, where the first byte of the frame (the first destination MAC address octet) is position 0. <p>Caution: You must set the offset to a position within the TX packet, or the PTP insertion operation fails. Also, you must not overlap the PTP fields.</p>
<code>i_ptp_tx_its[95:0]</code> <code>i_ptp_tx_its[191:0]</code>	96 bits (10GE/ 25GE,50GE/100GE/ 200GE) 192 bits (400GE)	Presents the ingress timestamp for the TX packet residence time calculation when Ethernet rate is 10GE/25GE,50GE/100GE/200GE/400GE.

Signal Name	Width	Description
		<p>The timestamp represents time when packet arrives in the system. To generate the residence time, the TX packet compares this timestamp to the time when packet left the system.</p> <ul style="list-style-type: none"> For TX MAC Avalon ST client interface, <code>i_ptp_tx_its[95:0]</code> is valid only when both TX valid (<code>i_tx_valid</code>) and TX SOP (<code>i_tx_startofpacket</code>) signals are asserted. For TX MAC segmented client interface, <code>i_ptp_tx_its[95:0]</code> is valid only when TX valid (<code>i_tx_mac_valid</code>) is asserted and <code>i_tx_mac_inframe[7:0]</code> contains the start of the packet (SOP). <p>The <code>i_ptp_tx_its[191:96]</code> is valid only when TX valid (<code>i_tx_mac_valid</code>) is asserted and <code>i_tx_mac_inframe[15:8]</code> contains the SOP.</p> <ul style="list-style-type: none"> The residence time value updated in the correction field is not valid if TX egress timestamp is larger than the TX ingress timestamp by 4 seconds.

7.14.3.1. PTP Field Offset for 1-Step Operation

The PTP field offset (`i_ptp_ts_offset`, `i_ptp_cf_offset`, and `i_ptp_csum_offset`) for a 1-step operation must follow one of the following formats:

- eCPRI type 5 message
- PTP message transported over Ethernet, UDP with IPv4 protocol, UDP with IPv6 protocol, with a VLAN, stacked VLAN, or MPLS header encapsulation.

7.14.4. RX Timestamp Interface

The RX timestamp interface provides RX timestamps for any received packets.

Each RX packet contains a timestamp, recorded in the `o_ptp_rx_its` bus. The timestamp signals are valid when `o_rx_valid` and `o_rx_startofpacket` signals are high.

Figure 60. Using the IEEE 1588 RX Timestamp Interface

The figure depicts two RX packet containing two RX packets and their respective ingress timestamps.

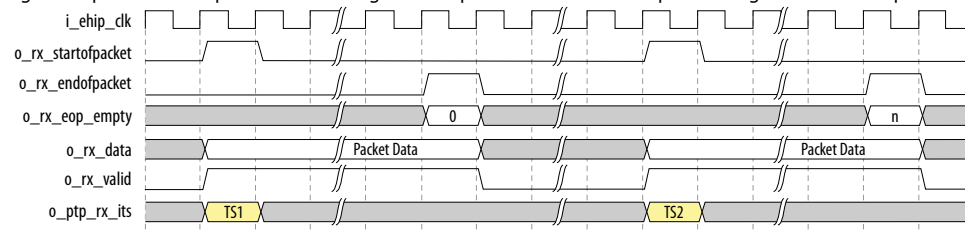


Table 62. Signals of the RX Timestamp Interface

All interface signals are clocked by `i_clk_rx` clock. The timestamp is always in 1588 v2 format.

Signal Name	Width	Description
o_ptp_rx_its[95:0] o_ptp_rx_its[191:0]	96 bits (10GE/ 25GE,50GE/100GE/ 200GE) 192 bits (400GE)	Ingress timestamp signal for the incoming RX packets when Ethernet rate is 10GE/25GE,50GE/100GE/200GE/400GE. <ul style="list-style-type: none"> For RX MAC Avalon ST client interface, o_ptp_rx_its[95:0] is valid only when both, o_rx_valid and o_rx_startofpacket signals, are equal to 1. For RX MAC segmented client interface, o_ptp_rx_its[95:0] is valid only when o_rx_valid is equal to 1 and o_rx_mac_inframe[7:0] contains the start of the packet (SOP). The o_ptp_rx_its[191:96] is valid only when o_rx_valid is equal to 1 and o_rx_mac_inframe[15:8] contains the SOP.

7.14.5. PTP Status Interface

The PTP status interface lets applications using PTP functions know when the PTP timestamp logic is ready for use.

Table 63. Signals of the PTP Status Interface

All interface signals are asynchronous.

Signal Name	Width	Description
o_tx_ptp_offset_data_valid	1	TX PTP offset data is valid. When asserted, indicates that the PTP offset data for TX data path is available to read from the Avalon memory-mapped interface registers: <ul style="list-style-type: none"> ptp_tx_lane_calc_data_constdelay ptp_tx_lane<n>_calc_data_offset ptp_tx_lane<n>_calc_data_time ptp_tx_lane<n>_calc_data_wiredelay, where n is a range from 0 to number of active SERDES lanes. The ptp_tx_lane0_calc_data_time signal is the captured time relative to the first TX SERDES lane.
o_rx_ptp_offset_data_valid	1	RX PTP offset data is valid. When asserted, indicates that the PTP offset data for RX data path is available to read from the Avalon memory-mapped interface registers: <ul style="list-style-type: none"> ptp_rx_lane_calc_data_constdelay ptp_rx_lane<n>_calc_data_offset ptp_rx_lane<n>_calc_data_time ptp_rx_lane<n>_calc_data_wiredelay, where n is a range from 0 to number of active SERDES lanes. The ptp_rx_lane0_calc_data_time signal is the captured time relative to the first RX SERDES lane.
o_tx_ptp_ready	1	TX PTP logic is ready for use.
<i>continued...</i>		

Signal Name	Width	Description
		When asserted, indicates that the PTP for TX data path is fully functional and the TX egress timestamp is valid within the supported accuracy range.
o_rx_ptp_ready	1	RX PTP logic is ready for use. When asserted, indicates that the PTP for RX data path is fully functional and the RX ingress timestamp is valid within the supported accuracy range.

7.14.6. PTP Tile Interface

The PTP tile interface creates a connection between PTP Tile Adapter and one or more Ethernet IP. When PTP is enabled, you must connect the PTP link port of one or more Ethernet IP to the PTP Tile Adapter.

Table 64. Signal of the PTP Tile Interface

Name	Description
ptp_link	Represents a logical connection between PTP Tile Adapter and the Ethernet IP with enabled PTP. During Support-Logic Generation , if the ptp_link is connected, the IP flow generates a PTP signal bus between the PTP Tile Adapter and one or more Ethernet IP with enabled PTP option within the same F-tile design.

8. Configuration Registers

You access the Ethernet registers for the F-Tile Ethernet Intel FPGA Hard IP using the Avalon memory-mapped interface Ethernet reconfiguration interface on each channel. These registers use 32-bit addresses; for accessing individual bytes, use `byteenable` signals.

Write operations to a read-only register field have no effect. Read operations that address a Reserved register return an unspecified result. Write operations to Reserved registers have an undefined effect. Accesses to registers that do not exist in your IP core variation, or to register bits that are not defined in your IP core variation, have an unspecified result. You should consider these registers and register bits Reserved. Although you can only access registers in 32-bit read and write operations, you should not attempt to write or ascribe meaning to values in undefined register bits.

For more information about specific `reconfig_eth` address register description, refer to the *F-Tile Ethernet Intel FPGA Hard IP Register Map* and *F-Tile Auto-Negotiation and Link Training Register Map* IPXACT files.

Refer to [Specifying the IP Core Parameters and Options](#) on page 17 for information about generating the `.ipxact` containing the register information.

Note: Unauthorized access to register sets outside of the configured Ethernet fractures is not recommended. For example, if your design is configured for 25G Ethernet, you should not be able to access additional variations such as 100G/400G.

Related Information

- [F-Tile Ethernet Intel FPGA Hard IP Register Map](#)
- [F-Tile Auto-Negotiation and Link Training Register Map](#)
- [Specifying the IP Core Parameters and Options](#) on page 17

8.1. Ethernet Avalon Memory-Mapped Interface Address Space

The Reconfiguration Ethernet interface (`reconfig_eth`) provides access to the Ethernet Hard IP Avalon memory-mapped interface space for the local Ethernet Hard IP fracture, including MAC, PCS, and FEC interface, the interface to the PMA, as well as soft CSRs implemented in the FPGA fabric. All addresses are byte-based address even though the register description specifies 32 bit boundary.

Refer to the *F-Tile Ethernet Hard IP Register Map* to view the register map and registers description.

Table 65. Reconfiguration Ethernet Avalon Memory-Mapped Interface Address Ranges

The description for the reconfiguration ethernet interface is provided in IP-XACT format upon IP core generation.

Register Type	Address Range
F-Tile EMIB Configuration Registers	0x0000-0x00FC
Soft Control Status Registers (Soft CSRs)	0x0100-0x0FFC
EHIP Registers	0x1000-0x5FFC
FEC/Transceiver Interface Registers	0x6000-0x9FFC

Related Information

[F-Tile Ethernet Intel FPGA Hard IP Register Map](#)

8.1.1. Ethernet Hard IP Core CSRs

The Ethernet Hard IP CSRs consist of the MAC, PCS, and PTP registers implemented in the hardened Ethernet core on the F-Tile. These CSRs are set at device configuration time and cannot be reset back to their default values other than by re-writing.

The register addresses in the EHIP consist of a base address that is set based on Ethernet mode, and an offset from that address which is the same for all Ethernet modes. Below table displays all `eth_reconfig` base addresses. For example, the same registers from 0x1000-0x1FFC for a 25GE port are located at 0x2000-0x2FFC for a 50GE port, etc. All register addresses in the following sections are Offset Addresses and must be combined with the appropriate Base Address for the current Ethernet mode.

Table 66. Ethernet Hard IP Base Address per Ethernet Mode

Ethernet Mode	<code>eth_reconfig</code> Base Address
10GE/25GE	0x1000
50GE	0x2000
40GE/100GE	0x3000
200GE	0x4000
400GE	0x5000

Table 67. Ethernet Hard IP Offset Range

Ethernet Hard IP Function	Ethernet Hard IP Offset Address Range
PCS Configuration	0x000-0x07C
PCS Status	0x080-0x1FC
MAC/PTP Configuration	0x200-0x7FC
MAC/PTP Statistics	0x800-0xFFC

8.1.2. FEC and Transceiver Control and Status Registers

Lane segments organize the FEC and Transceivers registers. Each lane segment includes set of registers, replicated multiple times depending on the number of Ethernet fractures or FEC lanes required for a specific Ethernet mode.

Table 68. FEC/Transceiver Lane Segment Base Address per Ethernet Mode

The table describes the layout and a base address for each segment.

Ethernet Mode	FEC/Transceiver Lane Segment	eth_reconfig Base Address
25GE	Segment 0	0x6000
50GE	Segment 0	0x6200
	Segment 1	0x6400
100GE	Segment 0	0x6600
	Segment 1	0x6800
	Segment 2	0x6A00
	Segment 3	0x6C00
200GE	Segment 0	0x6E00
	Segment 1	0x7000
	Segment 2	0x7200
	Segment 3	0x7400
	Segment 4	0x7600
	Segment 5	0x7800
	Segment 6	0x7A00
	Segment 7	0x7C00
400GE	Segment 0	0x7E00
	Segment 1	0x8000
	Segment 2	0x8200
	Segment 3	0x8400
	Segment 4	0x8600
	Segment 5	0x8800
	Segment 6	0x8A00
	Segment 7	0x8C00
	Segment 8	0x8E00
	Segment 9	0x9000
	Segment 10	0x9200
	Segment 11	0x9400
	Segment 12	0x9600
	Segment 13	0x9800
	Segment 14	0x9A00
	Segment 15	0x9C00

The offset address describes the registers within one lane segment. The offset address combined with the lane segment base address creates the registers eth_reconfig address.

Table 69. FEC/Transceiver Lane Segment Offset Address Range

The table displays the offset address space for a specific lane segment.

FEC/Transceiver Interface Function	FEC/Transceiver Interface Lane Segment Offset Address Range
Transceiver Interface Control	0x000 - 0x0BC
FEC Configuration	0x0C0 - 0x0FC
Transceiver Status	0x100 - 0x13C
FEC Status	0x140 - 0x1FC

The lane segment in a given Ethernet mode does not support all FEC control and stats registers. Some registers are only valid in segment 0 or in every forth segment.

8.2. Transceiver Avalon Memory-Mapped Interface Address Space

Refer to the *F-Tile Architecture and PMA and FEC Direct PHY IP User Guide* for information about transceiver register map and descriptions.

Related Information

[F-Tile Architecture and PMA and FEC Direct PHY IP User Guide](#)

9. Supported Modules and IPs

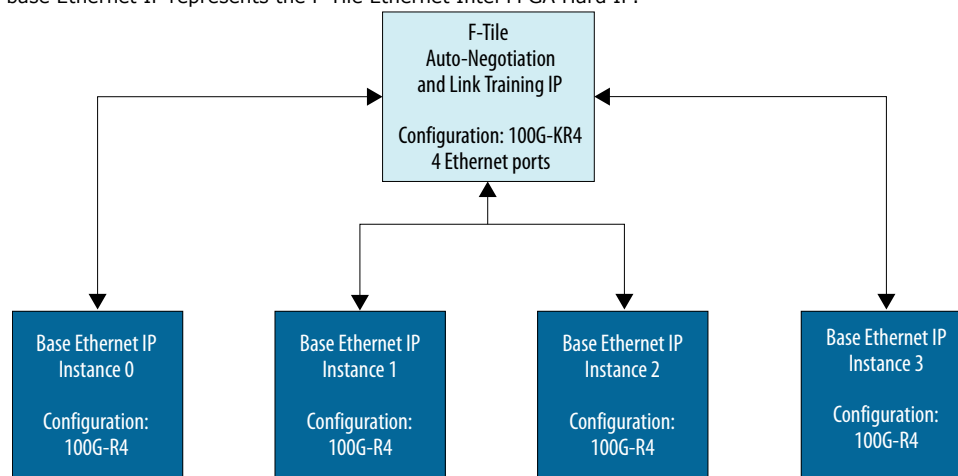
9.1. F-Tile Auto-Negotiation and Link Training For Ethernet Intel FPGA IP

9.1.1. Overview

The F-Tile Auto-Negotiation and Link Training for Ethernet Intel FPGA IP (F-tile AN/LT IP) implements the auto-negotiation and link training for F-tile Ethernet ports. You must instantiate the F-Tile Auto-Negotiation and Link Training for Ethernet Intel FPGA IP and connect it to the Base Ethernet IP⁽²⁰⁾. Each F-tile AN/LT IP supports one Ethernet rate with same PMA type and FEC mode and can be shared with up to 16 Ethernet ports.

Figure 61. F-Tile Auto-Negotiation and Link Training for Ethernet Intel FPGA IP: Single Protocol and Multiple Ethernet Rates Example

The base Ethernet IP represents the F-Tile Ethernet Intel FPGA Hard IP.

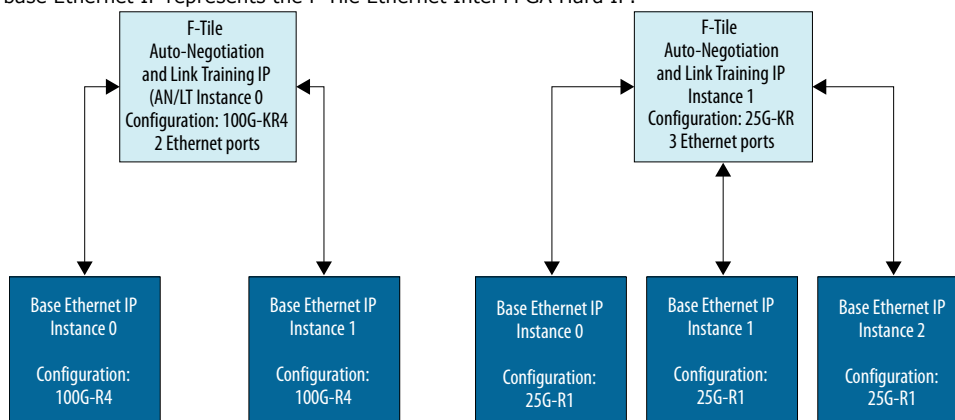


If you plan to integrate multiple Ethernet rates on your tile, you must instantiate multiple F-tile AN/LT IPs. For instance, to support 50G and 100G Ethernet rates with auto-negotiation and link training features, you must instantiate two F-tile AN/LT IP instances.

⁽²⁰⁾ Base Ethernet IP is equivalent to the F-Tile Ethernet Intel FPGA Hard IP.

Figure 62. F-Tile Auto-Negotiation and Link Training for Ethernet Intel FPGA IP: Two Ethernet Rates Example

The base Ethernet IP represents the F-Tile Ethernet Intel FPGA Hard IP.



Related Information

[F-Tile Ethernet Intel FPGA Hard IP Design Example User Guide](#)

9.1.2. Release Information

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

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

Table 70. F-Tile Auto-Negotiation and Link Training for Ethernet Intel FPGA IP Core Release Information

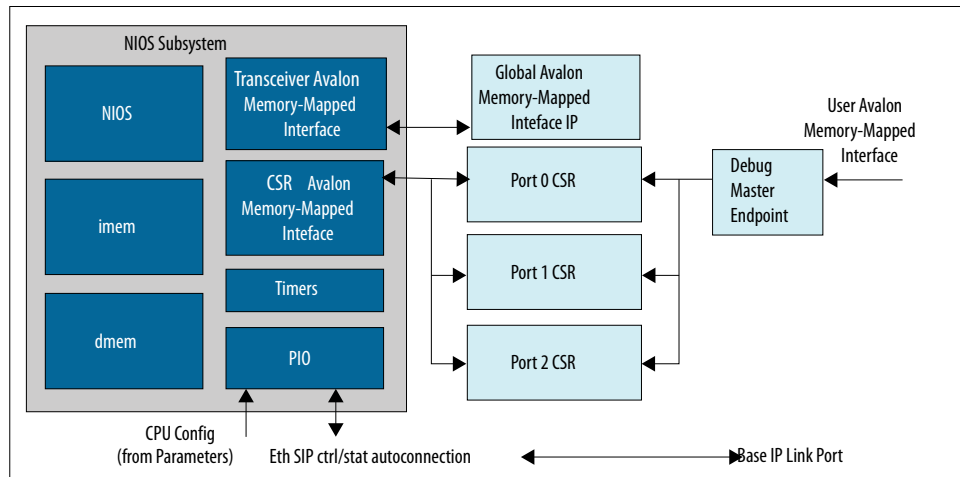
Item	Description
IP Version	12.0.0
Intel Quartus Prime Version	23.4
Release Date	2023.12.04
Ordering Code	IP-ETH-F-ANLT

Related Information

[F-Tile Auto-Negotiation and Link Training for Ethernet Intel FPGA IP Release Notes](#)

9.1.3. Functional Description

Figure 63. F-Tile Auto-Negotiation and Link Training Intel FPGA IP Block Diagram



Nios CPU subsystem executes the AN/LT firmware. CSR blocks provide interface to the client logic.

Once configured and brought out of reset, the AN/LT functionality is automatic. To ensure the auto connection expected behavior, connect `anlt_link` port from F-Tile AN/LT IP to the Ethernet IP.

Based on configuration, the F-Tile AN/LT IP begins the auto-negotiation and link training flows and the Ethernet IP function is temporarily disabled. Once AN/LT is complete, the Ethernet IP re-enables into a data mode and behaves as a standard Ethernet port. If the RX Ethernet link goes down and the LX timer is enabled but the timer is expired, the AN/LT IP may restart the auto-negotiation flow. If the RX link goes down and the LF timer is disabled, the link remains in data mode.

You can access the F-Tile AN/LT IP CSR registers at any time to monitor status, change configuration, or interrupt or restart the flow for any of the Ethernet ports connected to that AN/LT IP instance.

AN/LT designs support both internal and external serial loopback for FGT PMAs. However, for FHT PMAs, only external serial loopback is supported natively with AN/LT designs. Internal serial loopback with FHT PMAs require a specific TX EQ setting to function properly. The NIOS set these settings once the internal serial loopback is enabled. When the internal serial loopback is disabled, the NIOS restores the original TX EQ settings, therefore negating the TX EQ settings specified during LT. To support internal serial loopback with AN/LT designs, you should first disable the AN/LT via CSR register settings. When the link is forced into data mode, you can enable the internal serial loopback. Alternatively, if the link already passed through AN/LT and is in data mode, you can enable the internal serial loopback.

Note: B0 FHT multi-lane designs support bonding by default in F-Tile AN/LT IP, and non-bonded FHT multi-lane designs are not supported.

9.1.4. Parameters

The F-Tile Auto-Negotiation and Link Training for Ethernet Intel FPGA IP parameter editor provides the parameters you can set to configure your IP variation.

The F-Tile Auto-Negotiation and Link Training for Ethernet Intel FPGA IP parameter has the following tabs:

- **AN/LT Options**
- **AN Channel Map**
- **Target Profile Settings**

Figure 64. F-Tile Auto-Negotiation and Link Training for Ethernet Intel FPGA IP Parameters: IP Tab

F-Tile Auto-Negotiation and Link Training for Ethernet Intel FPGA IP
eth_anlt_f

AN/LT Options AN Channel Map Target Profile Settings

General Options

☒ Enable auto-negotiation on reset

☒ Enable link training on reset

☐ Enable ECC protection

PMA type: FGT

Ethernet mode: 10GE-1

KR or CR mode: CR mode

Number of ports: 4

FEC mode: None

Link fail inhibit time: 505 ms (Accepted range: 0-20000 ms)

Status clock frequency: 100.0 MHz (Accepted range: 100.0-250.0 MHz)

☐ Enable ANLT Debug Endpoint for Ethernet toolkit

☒ Enable fast simulation

☐ Enable Dynamic AN/LT

Note:

The **Target Profile Settings** tab is available only when **Enable Dynamic AN/LT** option is selected in the **AN/LT Options** tab in the F-Tile Auto-Negotiation and Link Training for Ethernet Intel FPGA IP GUI.

Table 71. F-Tile Auto-Negotiation and Link Training for Ethernet Intel FPGA IP Parameters: IP Tab

This table does not provide information about invalid parameter value combinations. If you make selections that create a conflict, the parameter editor generates error messages in the **System Messages** pane.

Parameter	Range	Default Setting	Parameter Description
Mode selection			
Enable auto-negotiation on reset	<ul style="list-style-type: none"> On Off 	On	Enable Auto-negotiation.
Enable link training on reset	<ul style="list-style-type: none"> On Off 	On	Enable Link Training.
Enable ECC protection	<ul style="list-style-type: none"> On Off 	Off	Enable Error Correction Code (ECC) for Nios II memory. To enable this feature, you must acquire Nios II license. Contact Intel Sales for information about acquiring a license.
PMA type	<ul style="list-style-type: none"> FHT FGT 	FGT	PMA Type. Selects the PMA type. Each PMA has a different data rate range and compliance specifications. The selected mode must match with PMA mode selected in Base IP.
Ethernet mode	<ul style="list-style-type: none"> 10GE-1 25GE-1 40GE-4 50GE-2 50GE-1 100GE-4 100GE-2 100GE-1 200GE-8 200GE-4 200GE-2 400GE-8 400GE-4 	10GE-1	Ethernet Configuration. Specifies the overall port bandwidth across the number of physical lanes used by the port. Term XGbE-Y represents: <ul style="list-style-type: none"> X is the overall bandwidth of the port Y is the number of physical lanes used by port The selected mode must match with Ethernet mode selected in Base IP.
KR or CR mode	<ul style="list-style-type: none"> KR mode CR mode 	CR mode	Selects the option during auto-negotiation.
Number of ports	1-16	4	Selects number of base Ethernet IP ports connected to the IP.
FEC mode	<ul style="list-style-type: none"> None IEEE 802.3 BASE-R Firecode (CL74) IEEE 802.3 RS(528,514) (CL91) or IEEE 802.3 RS(544,514) (CL134) Ethernet Technology Consortium RS(272, 258) 	None	Selects the FEC mode for each port. The IP core supports the following FEC types <ul style="list-style-type: none"> IEEE 802.3 BASE-R Firecode (CL74) is available only for 25GE-1. IEEE 802.3 RS(528,514) (CL91) IEEE 802.3 RS(544,514) (CL134) Ethernet Technology Consortium RS(272,258) is a low-latency substitute for RS(544,514).
<i>continued...</i>			

Parameter	Range	Default Setting	Parameter Description
Link fail inhibit time	0-20000	<ul style="list-style-type: none"> 505 (for NRZ) 3150 (for 50G PAM4) 12350 (for 100G PAM4) 	Sets the link fail inhibit timeout for auto-negotiation in milliseconds. Default value: <ul style="list-style-type: none"> 505 ms for NRZ modes 3150 ms for 50G PAM4 modes 12350 ms for 100G PAM4 modes
Status clock frequency	100-250	100	Selects the auto-negotiation and link training status clock frequency. Must be set to the frequency of the <code>i_clk</code> input for the correct timer functionality.
Enable ANLT Debug Endpoint for Ethernet Toolkit	<ul style="list-style-type: none"> On Off 	Off	Enables JTAG interface to external terminal so that Ethernet Tool Kit (ETK) can access the AN/LT configuration and status registers.
Enable Fast Simulation	<ul style="list-style-type: none"> On Off 	On	Enable fast simulation to skip auto-negotiation and link training functionality and FHT or FGT features. Additionally, it updates the AN/LT status registers to reflect AN/LT completion. <i>Note:</i> Enable Fast Simulation is not supported when Enable Dynamic AN/LT is enabled.
Enable Dynamic AN/LT	<ul style="list-style-type: none"> On Off 	Off	Enables the Dynamic reconfiguration feature for AN/LT IP. Make sure the correct combination of settings are used based on the base ethernet IP secondary profiles associated with the ANLT port. Selecting this parameter enables the Target Profile Settings tab. <i>Note:</i> The Enable AN/LT you select depends on the base ethernet IP secondary profiles associated with the AN/LT port.
AN Channel Map			
AN channel			
AN channel location <n>	0-7	0	Selects the AN lane default location in each Ethernet ports. <n> is an integer from 0 to (number of lanes - 1). For example, for 100GE-4, AN channel location <n> parameter can be 0 to 3. <i>Note:</i> <ul style="list-style-type: none"> When Enable Dynamic AN/LT is set, only AN_CHAN = 0 is supported for all multi-lane designs. The AN channel you select depends on the PMA physical channel connections on your board and the desired physical channel to enable AN/LT. This option is not available for single lane (For example, 10GE-1, 25GE-1).
Target Profile Settings			
<i>Note:</i> Make sure the correct combination of settings are used based on the base ethernet IP secondary profiles associated with the ANLT port.			
Advertise Low Latency FEC Request	<ul style="list-style-type: none"> On Off 	Off	Sets the value of the override consortium low-latency FEC request field, bit 3 of Register 0xCC.
Advertise RS-FEC Request	<ul style="list-style-type: none"> On Off 	Off	Must be set to the value of the override RS-FEC ability field, bit 10 of Register 0xCC.
continued...			

Parameter	Range	Default Setting	Parameter Description
Advertise RS-FEC Ability	<ul style="list-style-type: none"> On Off 	Off	Sets the value of the override RS-FEC ability field bit 8 of Register 0xCC.
Advertise BASER-FEC Request	<ul style="list-style-type: none"> On Off 	Off	Sets the value of the override BASER-FEC request field, bit 11 of Register 0xCC.
Advertise BASER-FEC Ability	<ul style="list-style-type: none"> On Off 	Off	Sets the value of the override BASER-FEC ability field, bit 9 of Register 0xCC.
Advertise Low Latency FEC Ability			
LL FEC Ability for 50G BASE-CR1/KR1	<ul style="list-style-type: none"> On Off 	Off	Sets the value of the override consortium low-latency FEC ability field, bit 4 of Register 0xCC.
LL FEC Ability for 100G BASE-CR2/KR2	<ul style="list-style-type: none"> On Off 	Off	Sets the value of the override consortium low-latency FEC ability field, bit 5 of Register 0xCC.
LL FEC Ability for 200G BASE-CR4/KR4	<ul style="list-style-type: none"> On Off 	Off	Sets the value of the override consortium low-latency FEC ability field, bit 6 of Register 0xCC.
Advertise IEEE Port Ability			
10G BASE-KR(A2)	<ul style="list-style-type: none"> On Off 	Off	Sets the value of the override IEEE port ability fields, bit 12 of Register 0xCC.
40G BASE-KR4 (A3)	<ul style="list-style-type: none"> On Off 	Off	Sets the value of the override IEEE port ability fields, bit 13 of Register 0xCC.
40G BASE-CR4 (A4)	<ul style="list-style-type: none"> On Off 	Off	Sets the value of the override IEEE port ability fields, bit 14 of Register 0xCC.
100G BASE-KR4(A7)	<ul style="list-style-type: none"> On Off 	Off	Sets the value of the override IEEE port ability fields, bit 15 of Register 0xCC.
100G BASE-CR4(A8)	<ul style="list-style-type: none"> On Off 	Off	Sets the value of the override IEEE port ability fields, bit 16 of Register 0xCC.
25G BASE-KR-S/CR-S/A9	<ul style="list-style-type: none"> On Off 	Off	Sets the value of the override IEEE port ability fields, bit 17 of Register 0xCC.
25G BASE-KR-CR-A10	<ul style="list-style-type: none"> On Off 	Off	Sets the value of the override IEEE port ability fields, bit 18 of Register 0xCC.
50G BASE-KR-CR(A13)	<ul style="list-style-type: none"> On Off 	Off	Sets the value of the override IEEE port ability fields Bit 19 of Register 0xCC.
100G BASE-KR2/CR2(A14)	<ul style="list-style-type: none"> On Off 	Off	Sets the value of the override IEEE port ability fields, bit 20 of Register 0xCC.
200G BASE-KR/CR4(A15)	<ul style="list-style-type: none"> On Off 	Off	Sets the value of the override IEEE port ability fields, bit 21 of Register 0xCC.
100G BASE-KR/CR(A16)	<ul style="list-style-type: none"> On Off 	Off	Sets the value of the override IEEE port ability fields, bit 22 of Register 0xCC.
200G BASE - KR2/CR2 (A17)	<ul style="list-style-type: none"> On Off 	Off	Sets the value of the override IEEE port ability fields, bit 23 of Register 0xCC.
400G BASE - KR4/CR4	<ul style="list-style-type: none"> On Off 	Off	Sets the value of the override IEEE port ability fields, bit 29 of Register 0xCC.
Advertise Consortium Port Ability			
<i>continued...</i>			

Parameter	Range	Default Setting	Parameter Description
25G BASE-KR1 (D20)	<ul style="list-style-type: none"> On Off 	Off	Sets the value of the override consortium port ability fields, bit 24 of Register 0xCC.
25G BASE-CR1 (D21)	<ul style="list-style-type: none"> On Off 	Off	Sets the value of the override consortium port ability fields, bit 25 of Register 0xCC.
50 BASE-KR2 (D24)	<ul style="list-style-type: none"> On Off 	Off	Sets the value of the override consortium port ability fields, bit 26 of Register 0xCC.
50 BASE-CR2 (D25)	<ul style="list-style-type: none"> On Off 	Off	Sets the value of the override consortium port ability fields, bit 27 of Register 0xCC.
400G BASR-KR8/CR8 (D34)	<ul style="list-style-type: none"> On Off 	Off	Sets the value of the override consortium port ability fields, bit 28 of Register 0xCC.

9.1.5. Clocks and Resets

Table 72. Clock and Reset Ports

Name	Description
i_clk	Clock source with 100 MHz - 250 MHz frequency. When AN/LT is enabled, i_clk can be driven at 1GHz for faster simulation times.
i_reset	Active high reset, synchronous to i_clk clock.

Table 73. Avalon Memory-Mapped Interface Ports

The interface signals are clocked by the i_clk clock.

Name	Width	Description
i_kr_reconfig_addr[11:0]	12	Address bus for auto-negotiation and link training control and status registers (AN/LT CSRs). <ul style="list-style-type: none"> Bits [11:8]: Port number [7:0]: CSR space for each port
i_kr_reconfig_read	1	Read enable for AN/LT CSRs.
i_kr_reconfig_write	1	Write enable for AN/LT CSRs.
i_kr_reconfig_byte_en[3:0]	4	AN/LT byte enable signal for writing data.
i_kr_reconfig_writedata[31:0]	32	Write data for AN/LT CSRs.
o_kr_reconfig_readdata[31:0]	32	Read data from AN/LT CSRs.
o_kr_reconfig_readdata_valid	1	Valid signal for AN/LT CSRs read data. When asserted, the register is valid.
o_kr_reconfig_waitrequest	1	Indicates that the Avalon memory-mapped interface is busy. The read or write cycle is only complete when this signal goes low.

Table 74. Base IP Ports Connection

Name	Width	Description
anlt_link	[NUMPORTS_GUI-1:0]	Used to connect to NUMPORTS_GUI Ethernet IP Instances. You must connect the port to the anlt_link port of the F-Tile Ethernet Intel FPGA Hard IP. <i>Note:</i> This is a virtual wire that carries no signal information used by the Intel Quartus Prime Tile Logic Generation flow to correctly connect the AN/LT IP to the Ethernet IP.

9.1.6. Registers

The auto-negotiation and link training registers are available per each Ethernet port the F-Tile Auto-Negotiation and Link Training for Ethernet Intel FPGA IP controls.

Refer to the *F-tile Auto-Negotiation and Link Training Register Map* to access register map and registers description. The register map documents the registers using the byte address offset. The physical Avalon Memory-Mapped Interface (AVMM) is based on 32-bit word addresses. However this document refers to the registers as byte addresses, you can convert to word addresses by shifting 2 bits to the right (divide by 4). You can use a byte enabled signal to address individual bytes.

Related Information

F-Tile Auto-Negotiation and Link Training Register Map

9.2. PTP Tile Adapter

9.2.1. Overview

PTP tile adapter transfers the PTP signals between the Ethernet IP and the tile.

The PTP tile adapter module is generated along with the Ethernet IP when you enable PTP option in the Ethernet IP parameter editor. In your design, you must instantiate PTP tile adapter (`eth_ptp_adpt_f`) instance and connect it to the F-Tile Ethernet Intel FPGA Hard IP. A single PTP tile adapter instance accommodates all PTP variants on the same F-tile.

The module transfers the signals through dedicated EMIB6 and EMIB7 channels. While connection between adapter and tile is automatic, you must manually connect the bus interface between the PTP tile adapter and the associated F-Tile Ethernet Intel FPGA Hard IP. You access the EMIB6 and EMIB7 registers for Asymmetry Delay and P2P Mean Path Delay through the Avalon memory-mapped interface. To align data transfer across multiple EMIBs from the FPGA fabric to the tile, the PTP Tile Adapter also generates TX deskew pulse for EMIBs of the Ethernet IP with PTP enabled.

9.2.2. Clocks, Reset, and Interface Ports

Table 75. Clock Ports

Name	Description
<code>i_sys_clk</code>	Clock source to drive TX deskew pulse generation logic. Must be connected to the F-Tile Reference and System PLL Clock Intel FPGA IP clock source (<code>o_clk_pll</code>). The clock frequency is equivalent to a half of a system PLL frequency, specified by the System PLL frequency parameter. The minimal frequency is 402.83 MHz.
<code>i_reconfig_clk</code>	Reconfiguration clock for Avalon memory-mapped interface.
<code>o_clk_pll</code>	Clock derived from the F-Tile Reference and System PLL Clock Intel FPGA IP associated with the Ethernet IP port. The <code>o_clk_pll</code> frequency is equal to PLL frequency divided by 2. When PTP is enabled, the PTP tile adapter's <code>o_clk_pll</code> is a clock input to the <code>i_clk_tx</code> , <code>i_clk_rx</code> Ethernet clocks, and <code>i_clk_pll</code> for Asynchronous mode, for all Ethernet modes.

Table 76. Reset Ports

Name	Description
<code>i_rst_n</code>	Active-low reset signal synchronous to the <code>i_sys_clk</code> clock. Intel recommends connecting the reset to the synchronized version of the <code>o_tx_pll_locked</code> output signal of an active Ethernet IP port. You must assert this reset when the <code>i_sys_clk</code> clock is not stable.
<code>i_reconfig_reset</code>	Active-high reconfiguration reset signal. Resets the entire reconfiguration clock domain. You must assert this reset after power-on or during configuration. The <code>i_reconfig_clk</code> must be stable before deasserting this reset.

Table 77. Interface Ports

Name	Description
ptp_link	Represents a logical connection between PTP Tile Adapter and the Ethernet IP with enabled PTP. During Support-Logic Generation , if the ptp_link is connected, the IP flow generates a PTP signal bus between the PTP Tile Adapter and one or more Ethernet IP with enabled PTP option within the same F-tile design.

9.2.3. PTP Asymmetry Delay Reconfiguration Interface

Table 78. PTP Asymmetry Reconfiguration Interface

The signals in this interface are clocked by the i_reconfig_clk clock and reset by the i_reconfig_reset signal of the PTP tile adapter. This clock and reset are used for all the reconfiguration interfaces in the IP core.

Port Name	Width	Description
i_reconfig_ptp_asym_addr[16:0]	17 bits	Word address bus for PTP asymmetry delay and status registers.
i_reconfig_ptp_asym_read	1 bit	Read request signal for PTP asymmetry delay and status registers.
i_reconfig_ptp_asym_write	1 bit	Write request signal for PTP asymmetry delay and status registers.
i_reconfig_ptp_asym_byteenable[3:0]	4 bits	Byte enable for PTP asymmetry read and write request signals.
o_reconfig_ptp_asym_readdata[31:0]	32 bits	Read data from reads to PTP asymmetry delay and status registers.
o_reconfig_ptp_asym_readdata_valid	1 bit	When set, read data from PTP asymmetry delay and status registers is valid.
i_reconfig_ptp_asym_writedata[31:0]	32 bits	Write data for PTP asymmetry delay and status registers.
o_reconfig_ptp_asym_waitrequest	1 bit	Avalon memory-mapped interface stalling signal for operations on PTP asymmetry delay and status registers.

9.2.4. PTP Peer-to-Peer MeanPathDelay Reconfiguration Interface

Table 79. PTP Peer-to-Peer (P2P) MeanPathDelay Reconfiguration Interface

The signals in this interface are clocked by the i_reconfig_clk clock and reset by the i_reconfig_reset signal of the PTP tile adapter. This clock and reset are used for all the reconfiguration interfaces in the IP core.

Port Name	Width	Description
i_reconfig_ptp_p2p_addr[16:0]	17 bits	Word address bus for PTP P2P MeanPathDelay and status registers.
i_reconfig_ptp_p2p_read	1 bit	Read request signal for PTP P2P MeanPathDelay and status registers.
i_reconfig_ptp_p2p_write	1 bit	Write request signal for PTP P2P MeanPathDelay and status registers.
i_reconfig_ptp_p2p_byteenable[3:0]	4 bits	Byte enable for PTP P2P MeanPathDelay read and write request signals.
o_reconfig_ptp_p2p_readdata[31:0]	32 bits	Read data from reads to PTP P2P MeanPathDelay and status registers.
continued...		

Port Name	Width	Description
o_reconfig_ptp_p2p_readdata_valid	1 bit	When set, read data from PTP P2P MeanPathDelay and status registers is valid.
i_reconfig_ptp_p2p_writedata[31:0]	32 bits	Write data for PTP P2P MeanPathDelay and status registers.
o_reconfig_ptp_p2p_waitrequest	1 bit	Avalon memory-mapped interface stalling signal for operations on PTP P2P MeanPathDelay and status registers.

10. Supported Tools

10.1. F-Tile Channel Placement Tool

F-tile supports datacenters, 5G networks, Smart Grid and other market segments. Ethernet, CPRI and OTN are the backbone of these emerging and traditional technologies. The *F-Tile Channel Placement Tool*, in conjunction with the *Device Family Pin Connection Guidelines*, allows you to swiftly plan protocol placements in the product prior to reading comprehensive documentation and implementing designs in Intel Quartus Prime software.

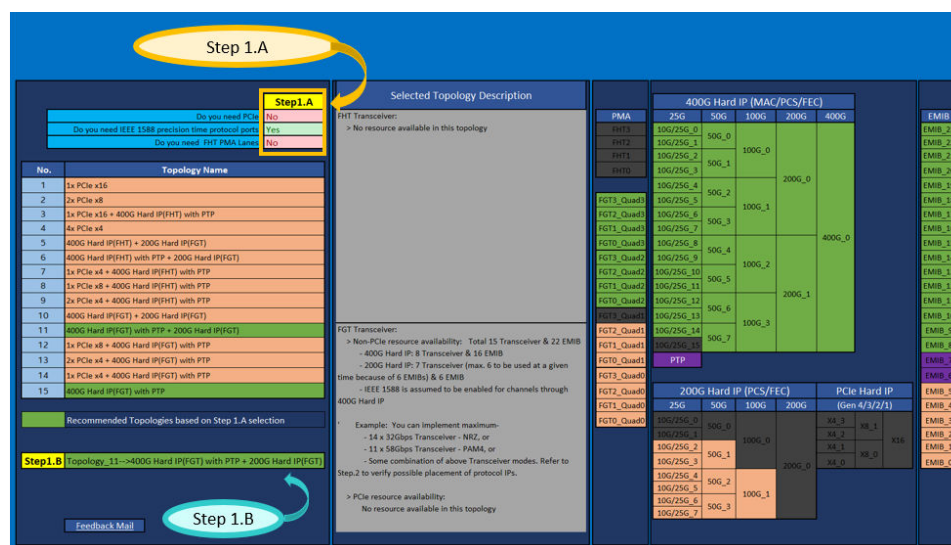
The Excel-based *F-Tile Channel Placement Tool*, supplemented with the following tabs is available for download at [F-Tile Channel Placement Tool](#):

- **Instructions**
- **PMA Hard IP Mapping (Reference)**
- **Step1_Topology Selection**
- **Step2 Hard IP Transceiver Placement**
- **Revision**

Note:

If your design requires use of the 200G hard IP block, use Intel Agilex 7 production device OPNs with the "C" suffix. If you are using Intel Agilex 7 production devices with OPNs that have no suffix (blank) or "B" suffix, and your design includes the 200G hard IP block, please contact the Intel customer support team for additional information.

Figure 65. F-Tile Channel Placement Tool



The screenshot displays the F-Tile Channel Placement Tool interface. A yellow callout bubble labeled "Step 1.A" points to the "Selected Topology Description" section. Another yellow callout bubble labeled "Step 1.B" points to the "Recommended Topologies based on Step 1.A selection" section.

Selected Topology Description

Do you need IEEE 1588 precision time protocol ports? ☐ No
Do you need PMA Hard IP? ☐ No

Topology Name

No.	Topology Name
1	1x PCIe x16
2	2x PCIe x8
3	1x PCIe x16 + 400G Hard IP (PHT) with PTP
4	4x PCIe x4
5	400G Hard IP (PHT) + 200G Hard IP (FGT)
6	400G Hard IP (PHT) with PTP + 200G Hard IP (FGT)
7	1x PCIe x8 + 400G Hard IP (PHT) with PTP
8	1x PCIe x8 + 400G Hard IP (PHT) with PTP
9	2x PCIe x8 + 400G Hard IP (PHT) with PTP
10	400G Hard IP (PHT) + 200G Hard IP (FGT)
11	400G Hard IP (PHT) with PTP + 200G Hard IP (FGT)
12	1x PCIe x8 + 400G Hard IP (PHT) with PTP
13	2x PCIe x8 + 400G Hard IP (PHT) with PTP
14	1x PCIe x8 + 400G Hard IP (PHT) with PTP
15	400G Hard IP (PHT) with PTP

Recommended Topologies based on Step 1.A selection

Step 1.B Topology 11 - 400G Hard IP (PHT) with PTP + 200G Hard IP (FGT)

Feedback Mail

Selected Topology Description

PHT Transceiver:
> No resource available in this topology

FGT Transceiver:
> Non-PCIe resource availability: Total 15 Transceiver & 22 EMB
> 400G Hard IP: 8 Transceiver & 16 EMB
> 200G Hard IP: 7 Transceiver (max. 6 to be used at a given time because of 6 EMBs) & 6 EMB
> IEEE 1588 is assumed to be enabled for channels through 400G Hard IP

Example: You can implement maximum:
- 14 x 32Gbps Transceiver - NR2, or
- 11 x 56Gbps Transceiver - PAMA, or
- Some combination of above Transceiver modes. Refer to Step 2 to verify possible placement of protocol IPs.

PCIe resource availability:
> No resource available in this topology

400G Hard IP (MAC/PCS/FEC)

PMA	25G	50G	100G	200G	400G	EMB
PHT1	100/25G_0	50G_0	100G_0	200G_0	400G_0	EMB_21
PHT2	100/25G_1	50G_1	100G_1	200G_1	400G_1	EMB_22
PHT3	100/25G_2	50G_2	100G_2	200G_2	400G_2	EMB_23
PHT4	100/25G_3	50G_3	100G_3	200G_3	400G_3	EMB_24
PHT5	100/25G_4	50G_4	100G_4	200G_4	400G_4	EMB_25
PHT6	100/25G_5	50G_5	100G_5	200G_5	400G_5	EMB_26
PHT7	100/25G_6	50G_6	100G_6	200G_6	400G_6	EMB_27
PHT8	100/25G_7	50G_7	100G_7	200G_7	400G_7	EMB_28
PHT9	100/25G_8	50G_8	100G_8	200G_8	400G_8	EMB_29
PHT10	100/25G_9	50G_9	100G_9	200G_9	400G_9	EMB_30
PHT11	100/25G_10	50G_10	100G_10	200G_10	400G_10	EMB_31
PHT12	100/25G_11	50G_11	100G_11	200G_11	400G_11	EMB_32
PHT13	100/25G_12	50G_12	100G_12	200G_12	400G_12	EMB_33
PHT14	100/25G_13	50G_13	100G_13	200G_13	400G_13	EMB_34
PHT15	100/25G_14	50G_14	100G_14	200G_14	400G_14	EMB_35
PHT16	100/25G_15	50G_15	100G_15	200G_15	400G_15	EMB_36
PHT17	100/25G_16	50G_16	100G_16	200G_16	400G_16	EMB_37
PHT18	100/25G_17	50G_17	100G_17	200G_17	400G_17	EMB_38
PHT19	100/25G_18	50G_18	100G_18	200G_18	400G_18	EMB_39
PHT20	100/25G_19	50G_19	100G_19	200G_19	400G_19	EMB_40
PHT21	100/25G_20	50G_20	100G_20	200G_20	400G_20	EMB_41
PHT22	100/25G_21	50G_21	100G_21	200G_21	400G_21	EMB_42
PHT23	100/25G_22	50G_22	100G_22	200G_22	400G_22	EMB_43
PHT24	100/25G_23	50G_23	100G_23	200G_23	400G_23	EMB_44
PHT25	100/25G_24	50G_24	100G_24	200G_24	400G_24	EMB_45
PHT26	100/25G_25	50G_25	100G_25	200G_25	400G_25	EMB_46
PHT27	100/25G_26	50G_26	100G_26	200G_26	400G_26	EMB_47
PHT28	100/25G_27	50G_27	100G_27	200G_27	400G_27	EMB_48
PHT29	100/25G_28	50G_28	100G_28	200G_28	400G_28	EMB_49
PHT30	100/25G_29	50G_29	100G_29	200G_29	400G_29	EMB_50
PHT31	100/25G_30	50G_30	100G_30	200G_30	400G_30	EMB_51
PHT32	100/25G_31	50G_31	100G_31	200G_31	400G_31	EMB_52
PHT33	100/25G_32	50G_32	100G_32	200G_32	400G_32	EMB_53
PHT34	100/25G_33	50G_33	100G_33	200G_33	400G_33	EMB_54
PHT35	100/25G_34	50G_34	100G_34	200G_34	400G_34	EMB_55
PHT36	100/25G_35	50G_35	100G_35	200G_35	400G_35	EMB_56
PHT37	100/25G_36	50G_36	100G_36	200G_36	400G_36	EMB_57
PHT38	100/25G_37	50G_37	100G_37	200G_37	400G_37	EMB_58
PHT39	100/25G_38	50G_38	100G_38	200G_38	400G_38	EMB_59
PHT40	100/25G_39	50G_39	100G_39	200G_39	400G_39	EMB_60
PHT41	100/25G_40	50G_40	100G_40	200G_40	400G_40	EMB_61
PHT42	100/25G_41	50G_41	100G_41	200G_41	400G_41	EMB_62
PHT43	100/25G_42	50G_42	100G_42	200G_42	400G_42	EMB_63
PHT44	100/25G_43	50G_43	100G_43	200G_43	400G_43	EMB_64
PHT45	100/25G_44	50G_44	100G_44	200G_44	400G_44	EMB_65
PHT46	100/25G_45	50G_45	100G_45	200G_45	400G_45	EMB_66
PHT47	100/25G_46	50G_46	100G_46	200G_46	400G_46	EMB_67
PHT48	100/25G_47	50G_47	100G_47	200G_47	400G_47	EMB_68
PHT49	100/25G_48	50G_48	100G_48	200G_48	400G_48	EMB_69
PHT50	100/25G_49	50G_49	100G_49	200G_49	400G_49	EMB_70
PHT51	100/25G_50	50G_50	100G_50	200G_50	400G_50	EMB_71
PHT52	100/25G_51	50G_51	100G_51	200G_51	400G_51	EMB_72
PHT53	100/25G_52	50G_52	100G_52	200G_52	400G_52	EMB_73
PHT54	100/25G_53	50G_53	100G_53	200G_53	400G_53	EMB_74
PHT55	100/25G_54	50G_54	100G_54	200G_54	400G_54	EMB_75
PHT56	100/25G_55	50G_55	100G_55	200G_55	400G_55	EMB_76
PHT57	100/25G_56	50G_56	100G_56	200G_56	400G_56	EMB_77
PHT58	100/25G_57	50G_57	100G_57	200G_57	400G_57	EMB_78
PHT59	100/25G_58	50G_58	100G_58	200G_58	400G_58	EMB_79
PHT60	100/25G_59	50G_59	100G_59	200G_59	400G_59	EMB_80
PHT61	100/25G_60	50G_60	100G_60	200G_60	400G_60	EMB_81
PHT62	100/25G_61	50G_61	100G_61	200G_61	400G_61	EMB_82
PHT63	100/25G_62	50G_62	100G_62	200G_62	400G_62	EMB_83
PHT64	100/25G_63	50G_63	100G_63	200G_63	400G_63	EMB_84
PHT65	100/25G_64	50G_64	100G_64	200G_64	400G_64	EMB_85
PHT66	100/25G_65	50G_65	100G_65	200G_65	400G_65	EMB_86
PHT67	100/25G_66	50G_66	100G_66	200G_66	400G_66	EMB_87
PHT68	100/25G_67	50G_67	100G_67	200G_67	400G_67	EMB_88
PHT69	100/25G_68	50G_68	100G_68	200G_68	400G_68	EMB_89
PHT70	100/25G_69	50G_69	100G_69	200G_69	400G_69	EMB_90
PHT71	100/25G_70	50G_70	100G_70	200G_70	400G_70	EMB_91
PHT72	100/25G_71	50G_71	100G_71	200G_71	400G_71	EMB_92
PHT73	100/25G_72	50G_72	100G_72	200G_72	400G_72	EMB_93
PHT74	100/25G_73	50G_73	100G_73	200G_73	400G_73	EMB_94
PHT75	100/25G_74	50G_74	100G_74	200G_74	400G_74	EMB_95
PHT76	100/25G_75	50G_75	100G_75	200G_75	400G_75	EMB_96
PHT77	100/25G_76	50G_76	100G_76	200G_76	400G_76	EMB_97
PHT78	100/25G_77	50G_77	100G_77	200G_77	400G_77	EMB_98
PHT79	100/25G_78	50G_78	100G_78	200G_78	400G_78	EMB_99
PHT80	100/25G_79	50G_79	100G_79	200G_79	400G_79	EMB_100
PHT81	100/25G_80	50G_80	100G_80	200G_80	400G_80	EMB_101
PHT82	100/25G_81	50G_81	100G_81	200G_81	400G_81	EMB_102
PHT83	100/25G_82	50G_82	100G_82	200G_82	400G_82	EMB_103
PHT84	100/25G_83	50G_83	100G_83	200G_83	400G_83	EMB_104
PHT85	100/25G_84	50G_84	100G_84	200G_84	400G_84	EMB_105
PHT86	100/25G_85	50G_85	100G_85	200G_85	400G_85	EMB_106
PHT87	100/25G_86	50G_86	100G_86	200G_86	400G_86	EMB_107
PHT88	100/25G_87	50G_87	100G_87	200G_87	400G_87	EMB_108
PHT89	100/25G_88	50G_88	100G_88	200G_88	400G_88	EMB_109
PHT90	100/25G_89	50G_89	100G_89	200G_89	400G_89	EMB_110
PHT91	100/25G_90	50G_90	100G_90	200G_90	400G_90	EMB_111
PHT92	100/25G_91	50G_91	100G_91	200G_91	400G_91	EMB_112
PHT93	100/25G_92	50G_92	100G_92	200G_92	400G_92	EMB_113
PHT94	100/25G_93	50G_93	100G_93	200G_93	400G_93	EMB_114
PHT95	100/25G_94	50G_94	100G_94	200G_94	400G_94	EMB_115
PHT96	100/25G_95	50G_95	100G_95	200G_95	400G_95	EMB_116
PHT97	100/25G_96	50G_96	100G_96	200G_96	400G_96	EMB_117
PHT98	100/25G_97	50G_97	100G_97	200G_97	400G_97	EMB_118
PHT99	100/25G_98	50G_98	100G_98	200G_98	400G_98	EMB_119
PHT100	100/25G_99	50G_99	100G_99	200G_99	400G_99	EMB_120
PHT101	100/25G_100	50G_100	100G_100	200G_100	400G_100	EMB_121
PHT102	100/25G_101	50G_101	100G_101	200G_101	400G_101	EMB_122
PHT103	100/25G_102	50G_102	100G_102	200G_102	400G_102	EMB_123
PHT104	100/25G_103	50G_103	100G_103	200G_103	400G_103	EMB_124
PHT105	100/25G_104	50G_104	100G_104	200G_104	400G_104	EMB_125
PHT106	100/25G_105	50G_105	100G_105	200G_105	400G_105	EMB_126
PHT107	100/25G_106	50G_106	100G_106	200G_106	400G_106	EMB_127
PHT108	100/25G_107	50G_107	100G_107	200G_107	400G_107	EMB_128
PHT109	100/25G_108	50G_108	100G_108	200G_108	400G_108	EMB_129
PHT110	100/25G_109	50G_109	100G_109	200G_109	400G_109	EMB_130
PHT111	100/25G_110	50G_110	100G_110	200G_110	400G_110	EMB_131
PHT112	100/25G_111	50G_111	100G_111	200G_111	400G_111	EMB_132
PHT113	100/25G_112	50G_112	100G_112	200G_112	400G_112	EMB_133
PHT114	100/25G_113	50G_113	100G_113	200G_113	400G_113	EMB_134
PHT115	100/25G_114	50G_114	100G_114	200G_114	400G_114	EMB_135
PHT116	100/25G_115	50G_115	100G_115	200G_115	400G_115	EMB_136
PHT117	100/25G_116	50G_116	100G_116	200G_116	400G_116	EMB_137
PHT118	100/25G_117	50G_117	100G_117	200G_117	400G_117	EMB_138
PHT119	100/25G_118	50G_118	100G_118	200G_118	400G_118	EMB_139
PHT120	100/25G_119	50G_119	100G_119	200G_119	400G_119	EMB_140
PHT121	100/25G_120	50G_120	100G_120	200G_120	400G_120	EMB_141
PHT122	100/25G_121	50G_121	100G_121	200G_121	400G_121	EMB_142
PHT123	100/25					

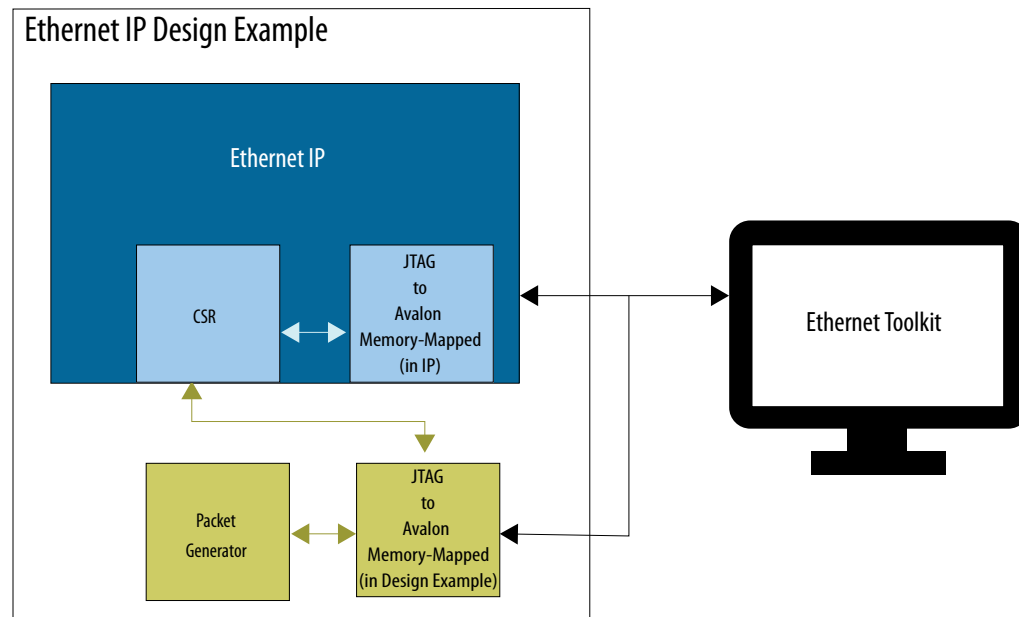
Related Information

- [F-Tile Channel Placement Tool](#)
- [Intel Agilex 7 Device Family Pin Connection Guidelines](#)

10.2. Ethernet Toolkit Overview

The Ethernet Toolkit is a TCL based debugging tool that allows you to interact with an Ethernet Intel FPGA IP in real time.

Figure 66. Block Diagram of the Ethernet Toolkit



You can use the Ethernet Toolkit with hardware design that has standalone Ethernet IP. You can also use the Ethernet Toolkit with an Intel Quartus Prime generated Ethernet IP design example.

10.2.1. Features

The Ethernet Toolkit offers the following features when used with hardware design that has standalone Ethernet IP as well as with an Intel Quartus Prime generated Ethernet IP design example:

- Verifies the status of the Ethernet link.
- Reads and writes to status and configuration registers of the IP.
- Displays the values of TX/RX status and statistics registers.
- Ability to assert and deassert IP resets.
- Verifies the IPs error correction capability.

The Ethernet Toolkit also offers some additional features when used with an Intel Quartus Prime generated Ethernet IP design example:

- Provides access to the example design packet generator.
- Execute testing procedures to verify the functionality of Ethernet IPs.
- Enable and disable MAC loopback.
- Set source and destination MAC addresses.



11. F-Tile Ethernet Intel FPGA Hard IP User Guide Archives

For the latest and previous versions of this user guide, refer to [F-Tile Ethernet Intel FPGA Hard IP User Guide](#). If an IP or software version is not listed, the user guide for the previous IP or software version applies.

12. Document Revision History for the F-Tile Ethernet Intel FPGA Hard IP User Guide

Document Version	Intel Quartus Prime Version	Changes
2023.12.04	23.4	<p>Made the following changes:</p> <ul style="list-style-type: none"> Updated resource utilization table due to changes from Nios II to Nios V. Corrected the description for <i>Stop TX traffic when link partner sends PAUSE</i> parameter in <i>Parameters</i>. Added new IP Parameter: <i>Enable IPXACT</i> in <i>Parameters</i>. Updated TX and RX table fields for UI Value and PMA Delay of Ethernet Modes in <i>UI Value and PMA Delay</i>. Removed the restrictions on PTP support for Agilex 7 041 devices in <i>Device Speed Grade Support</i>.
2023.10.02	23.3	<p>Made the following changes:</p> <ul style="list-style-type: none"> Corrected the <code>i_ptp_ts_offset</code> description and image in <i>TX 1-Step Timestamp Interface</i>. Added F-Tile Ethernet Intel FPGA Hard IP Parameters: IP Tab screenshot with Analog parameters tab in <i>F-Tile Ethernet Intel FPGA Hard IP Parameters</i>. Removed the IP Parameter: Include Deterministic Latency Interface in <i>Parameter</i> section. Removed <i>Deterministic latency Interface</i> topic in <i>Interface Overview</i> section. Added F-Tile Auto-Negotiation and Link Training for Ethernet Intel FPGA IP parameter screenshot with <i>Target Profile Settings</i> tab. Added the following parameters in the F-Tile Auto-Negotiation and Link Training for Ethernet Intel FPGA IP <i>Parameters</i> section: <ul style="list-style-type: none"> — Enable AN/LT Debug Endpoint for Ethernet Toolkit — Enable Fast Simulation — Enable Dynamic AN/LT
2023.06.26	23.2	<ul style="list-style-type: none"> Updated alignment marker verbiage in <i>OTN Mode</i>. Added note in <i>PTP RX Client Flow</i>. Added description about alignment marker for the signal name <code>i_tx_pcs66_am</code> in <i>Signals of the PCS66 TX Interface</i> table. Updated ANLT functional description in <i>Supported Modules and IPs</i>. Added note under AN channel in <i>AN/LT parameters</i> IP tab. Added a verbiage about converting byte addresses to word addresses by shifting 2 bits to the right (divide by 4) in <i>Registers</i>. Added a note verbiage "unauthorized access to register sets outside of the configured Ethernet fractures is not recommended" in <i>Configuration Registers</i>.
continued...		

Document Version	Intel Quartus Prime Version	Changes
2023.04.03	23.1	<ul style="list-style-type: none"> Updated the following in the <i>Round-trip latency table</i>. <ul style="list-style-type: none"> All the IEEE 802.3 RS(544,514) replaced with IEEE 802.3 RS(528,514) (CL91). All the IEEE 802.3 RS(528,514) (CL91) replaced with IEEE 802.3 RS(544,514) (CL134). Added steps to retrieve the N divider value in <i>Clock Connections in Synchronous Ethernet Operation</i>. Removed the note "Enable TX packing feature is not available when PTP is turned on" in <i>TX Packing Logic</i>. Updated ALMs numbers in <i>TX Packing Logic</i> section. Updated <i>Features</i> in <i>Precision Time Protocol</i> section to include 200GE and 400GE modes for timestamp accuracy in advance mode. Updated 200GE and 400GE Ethernet Data Rate to ± 1.5ns for advanced accuracy mode in <i>PTP Timestamp Accuracy</i>. Added new topic: <i>Routing Delay Adjustment for Basic Timestamp Accuracy Mode</i>. Updated the product family name to "Intel Agilex 7."
2022.12.19	22.4	<p>Made the following changes:</p> <ul style="list-style-type: none"> Updated F-Tile Ethernet Intel FPGA Hard IP Block Diagram to remove filler text in <i>F-Tile Ethernet Intel FPGA Hard IP Overview</i>. Added the following footnotes in <i>Variant Selection</i> table. <ul style="list-style-type: none"> The Auto-negotiation and link training hardware is available for no FEC, IEEE 802.3 BASE-R Firecode (CL74), and IEEE 802.3 RS(528,514) (CL91). The Auto-negotiation and link training is not available for IEEE 802.3 RS(544,514) (CL134). Updated note about core speed grades for IP core variation with PTP in <i>Device Speed Grade Support</i>. Added note about only 10GE with PTP support in advanced mode for OPNs AGIB041R29D1E2VR0, AGID041R29D1E2VR0, or any device density code of 041 in <i>Device Speed Grade Support</i>. Added new topic: <i>Round-trip Latency</i> in <i>F-Tile Ethernet Intel FPGA Hard IP Overview</i>. Updated <i>Release Information</i>. Added the following parameters: <ul style="list-style-type: none"> <i>Include Deterministic Latency Measurement</i> <i>Advanced Mode</i> <i>Custom Ethernet line rate</i> Updated <i>Include 32-bit soft CWBIN Counters</i> default setting to <i>Off</i> in <i>Parameters</i>. Updated <i>Enable TX packing</i> Parameter Description in <i>Parameters</i>. Added <i>i_sampling_clk</i> in <i>Clocks</i>. Added content for <i>Example Design Generation of Synchronous Ethernet Operation</i> in <i>Clock Connections in Synchronous Ethernet Operation</i>. Updated the code for calculate pulse adjustment and FEC variants in <i>PTP RX Client Flow</i>. Added a description for <i>i_tx_pcs66_am</i> in <i>FlexE and OTN Mode TX Interface</i>. Added a description for <i>o_rx_pcs66_am_valid</i> in <i>FlexE and OTN Mode RX Interface</i>.

continued...

Document Version	Intel Quartus Prime Version	Changes
		<ul style="list-style-type: none"> Added new subsection <i>Deterministic Latency Measurement</i> Added auto-negotiation and link training feature note in <i>Reset Signals</i>. Added reference to <i>Pipelined Transfers</i> of Avalon Interface Specifications in <i>Reconfiguration Interfaces</i>.
2022.09.26	22.3	<p>Made the following changes:</p> <ul style="list-style-type: none"> Added <i>Core Speed Grade/Transceiver Speed Grade</i> table in <i>Device Speed Grade Support</i>. Updated IP-XACT file information in <i>Generating IP-XACT File</i>. Updated F-Tile Ethernet Intel FPGA Hard IP Block Diagram to include CWBIN in <i>F-Tile Ethernet Intel FPGA Hard IP Overview</i>. Added <i>32-bit soft CWBIN counters</i> parameter description in <i>F-Tile Ethernet Intel FPGA Hard IP Overview</i>. Updated <i>Release Information</i>. Added <i>Include 32bit soft CWBIN counters</i> and <i>Reconfig clock frequency</i> parameters in <i>F-Tile Ethernet Intel FPGA Hard IP Parameters</i>. Added <i>Enable IEEE 1588 PTP</i> parameter description "When disabled, connect this port to 1'b0" in <i>Clock Signals</i> table. Added new topic: <i>32-bit Soft CWBIN Counters</i>. Added note about auto-negotiation and link training bonding support for B0 FHT multi-lane designs in <i>AN/LT Functional Description</i>. Added new topic: <i>Implementation of Synchronous Ethernet (syncE) operation</i> in <i>Clocks</i> section. Corrected <i>o_rx_mac_status_data</i>'s value 3'd3 to reserved in <i>RX MAC Segmented Client Interface</i> section.
2022.06.20	22.2	<p>Made the following changes:</p> <ul style="list-style-type: none"> Updated note in <i>Device Speed Grade Support</i>. Updated the <i>PCS Mode RX Interface</i> section to fix Hardware AM Cycle numbers. Updated the <i>Client Flow Glossary</i> table in <i>PTP Client Flow</i> section. Added a row FL-Total number of FEC lanes of the variant. Added new topic: <i>Deterministic latency Interface</i> Added new topic: <i>TX Packing Logic</i>. Added <i>Enable TX Packing</i> and <i>Include Deterministic Latency Interface</i> parameters in <i>F-Tile Ethernet Intel FPGA Hard IP Parameters</i>. Fixed the broken link <i>F-Tile Channel Placement Tool</i> in <i>Supported Tools</i>.
2022.03.28	22.1	<p>Made the following changes:</p> <ul style="list-style-type: none"> Updated the <i>Variant Selection</i> table. The Auto-negotiation and link training hardware is available for FHT NRZ 25GE-1, 50GE-2, and 100GE-4. Added transceiver speed grade-related information in <i>Device Speed Grade Support</i>. Added IP-XACT statement in <i>Specifying the IP Core Parameters and Options</i>. Added new topic: <i>Generating IP-XACT File</i> Updated caution note in <i>Frame Padding</i> to clarify 9-byte payload requirements. Re-ordered step 7 and step 8 in <i>PTP TX Client Flow</i>. Re-ordered step 9 and step 10 in <i>PTP RX Client Flow</i>. Updated TX/RX hardware PMA delay for FHT transceivers in the <i>UI Value and PMA Delay of Ethernet Modes</i> table.

continued...

Document Version	Intel Quartus Prime Version	Changes
		<ul style="list-style-type: none"> Updated the <i>Clocks</i> section to remove the PTP clock limitation. The PTP clock runs at 100 to 250 MHz. Removed the CRC bytes statement in <i>RX MAC Avalon ST Client Interface with Enabled Passthrough and RX CRC Forwarding</i>. Added reference to the .ipxact file generation in <i>Configuration Registers</i>. Revised the i_rst_n reset description in the <i>PTP Tile Adapter: Clocks, Reset, and Interface Ports</i> section. Removed Verilog file format specific note from the Auto-Negotiation and Link Training <i>Parameters</i> section. The IP supports VHDL for synthesis and simulation. Updated <i>F-Tile Ethernet Intel FPGA Hard IP User Guide Archives</i>.
2022.01.07	21.4	<p>Made the following changes:</p> <ul style="list-style-type: none"> Updated <i>F-Tile Ethernet Intel FPGA Hard IP</i>: <ul style="list-style-type: none"> Added note about auto-negotiation and link training hardware support in <i>Overview</i>. Added new section: <i>Device Speed Grade Support</i>. Updated <i>Generating Tile Files</i> to include quartus_tlg updates. Added the FHT precoding enable and Enable Native PHY Debug Endpoint parameters in the <i>F-Tile Ethernet Intel FPGA Hard IP Parameters</i>. Added PTP-specific i_reconfig_clk frequency range limitation. Updated <i>F-Tile Auto-Negotiation and Link Training for Ethernet Intel FPGA IP</i>: <ul style="list-style-type: none"> Removed outdated 21.3 specific statement in <i>Overview</i>. Updated <i>Release Information</i>. Added note about VHDL limitation in <i>F-Tile Auto-Negotiation and Link Training for Ethernet Intel FPGA IP Parameters</i>.
2021.10.04	21.3	<p>Made the following changes:</p> <ul style="list-style-type: none"> Added support for PCS lane re-ordering in the <i>Features</i> table. Removed 400GE-8 specific footnote from the <i>Variant Selection</i> table. 400GE-4 mode now supports auto-negotiation and link training. Revised PTP description in <i>F-Tile Ethernet Intel FPGA Hard IP Overview</i>. Added <i>Resource Utilization</i>. Updated <i>F-Tile Ethernet Intel FPGA Hard IP Parameters: IP Tab</i> table: <ul style="list-style-type: none"> Updated PMA reference frequency description. Added PTP-related frequency requirements in the System PLL frequency description. Added important note about hardware accuracy values in the Timestamp accuracy mode description. Added new Enable Ethernet Debug Master Endpoint parameter Added note in the <i>TX MAC Segmented Client Interface</i>. Updated the TX TAM adjust calculation in <i>PTP TX Client Flow</i>. Updated the RX TAM adjust calculation in <i>PTP RX Client Flow</i>. Added new section: <i>Routing Delay Adjustment for Advanced Timestamp Accuracy Mode</i>

continued...

Document Version	Intel Quartus Prime Version	Changes
		<ul style="list-style-type: none"> Updated <i>Precision Time Protocol Interface</i>: <ul style="list-style-type: none"> Removed 40G Ethernet rate from the PTP Clocks table. 40G Ethernet rate does not support PTP feature Corrected <code>i_clk_tx_tod</code> clock frequency from 250 MHz to 114.2857 MHz.. Revised step 6 in <i>RX UI Adjustment</i>. Update simulation-based RX TAM values for 50G Ethernet rate in <i>Reference Time (TAM) Interval</i>. Updated Hardware UX PMA delay values in the <i>UI Value and PMA Delay of Ethernet Modes</i> table. Globally added a note about PTP timestamp accuracy. The note emphasizes that the specified timestamp accuracy values in base and advanced modes represent simulation-based results. Updated <i>Clock Signals</i> descriptions for the following clock signals: <ul style="list-style-type: none"> <code>i_clk_tx</code> <code>i_clk_rx</code> <code>i_clk_pll</code> <code>i_clk_ref</code> Revised text and updated figure in <i>Clock Connections in PTP-Based Synchronous and Asynchronous Operation</i>. Corrected signal name from <code>i_sl_tx_valid</code> to <code>i_tx_valid</code> in the <i>TX MAC Avalon ST Client Interface</i>. Revised fixed latency range in <i>TX MAC Segmented Client Interface</i>. The <code>i_tx_mac_valid</code> and <code>o_tx_mac_ready</code> signals can be spaced by a fixed latency between 1 to 8 clock cycles. Revised <code>i_clk_ptp_sample</code> clock description in the <i>PTP Clock Ports</i> table. Added new topic: <i>Ethernet Toolkit Overview</i>
2021.06.28	21.2	Initial release

Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

[Altera:](#)

[IP-ETH-FTILEHIP](#) [IP-ETH-F-ANLT](#)