25G Ethernet Intel® Arria® 10 FPGA IP User Guide

Updated for Intel® Quartus® Prime Design Suite: **21.1**

IP Version: **19.4.0**

Online Version
Send Feedback

UG-20015

ID: 683639
Version: 2021.03.29

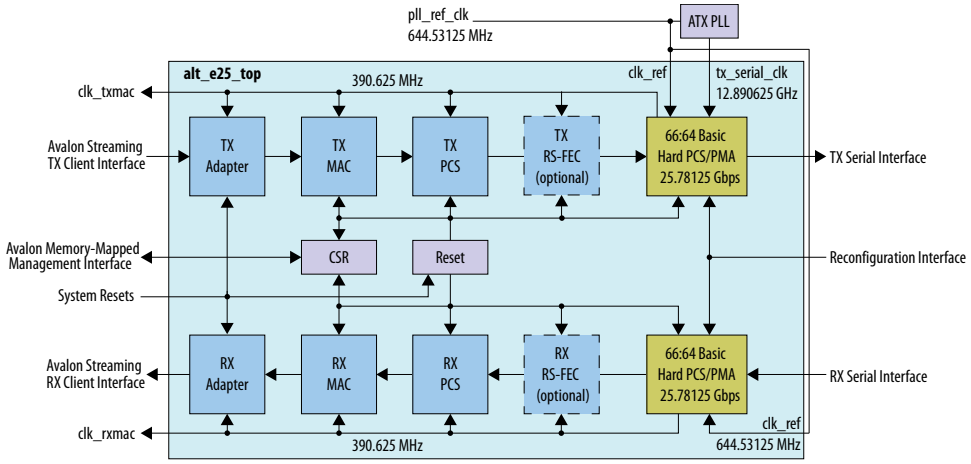
Contents

1. About the 25G Ethernet Intel FPGA IP

The 25G Ethernet Intel FPGA IP implements the *25G & 50G Ethernet Specification, Draft 1.6* from the 25 Gigabit Ethernet Consortium and the *IEEE 802.3by 25Gb Ethernet* specification. The IP includes an option to support unidirectional transport as defined in *Clause 66* of the *IEEE 802.3-2012 Ethernet Standard*. The MAC client side interface for the 25G Ethernet Intel FPGA IP is a 64-bit Avalon® streaming interface. It maps to one 25.78125 Gbps transceiver. The IP optionally includes the *IEEE 802.3-2018 Clause 108* Reed-Solomon forward error correction (RS-FEC) for support of *IEEE802.3-2018 Clause 107* 25GBASE-R PCS. *IEEE 802.3 Clause 73* Auto-Negotiation and *IEEE 802.3 Clause 74* CR/KR-FEC are not supported. Transceiver interface to 25GBASE-SR optical Physical Medium Dependent (PMD) transceiver is supported.

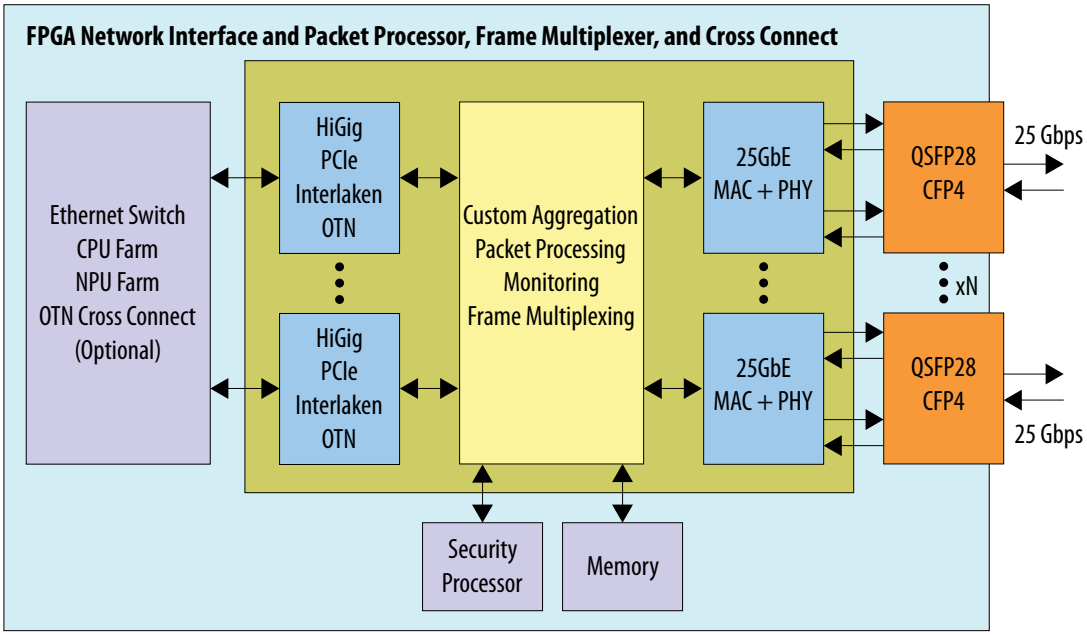
The IP core provides standard media access control (MAC) and physical coding sublayer (PCS), RS-FEC, and PMA functions shown in the following block diagram. The PHY comprises the PCS, optional RS-FEC, and PMA.

Figure 1. 25G Ethernet Intel FPGA IP MAC and PHY IP Clock Diagram



The following block diagram shows an example of a network application with 25G Ethernet Intel FPGA IP MAC and PHY.

Figure 2. Example Network Application



Related Information

25 Gigabit Ethernet Consortium

1.1. Release Information

Intel® FPGA IP versions match the Intel Quartus® Prime Design Suite software versions until v19.1. Starting in Intel Quartus Prime Design Suite software version 19.2, Intel FPGA IP has a new versioning scheme.

The Intel FPGA IP version (X.Y.Z) number can change with each Intel Quartus Prime software version. A change in:

- X indicates a major revision of the IP. If you update the 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 1. 25G Ethernet Intel FPGA IP Core Current Release Information

Item	Description
IP Version	19.4.0
Intel Quartus Prime Version	19.4
Release Date	2019.12.16
Ordering Codes	IP-25GEUMACPHY (IPR-25GEUMACPHY for renewal)

Related Information

25G Ethernet Intel FPGA IP Release Note
Describes changes to the IP in a particular release.

1.2. 25G Ethernet Intel FPGA IP Supported Features

The 25G Ethernet Intel FPGA IP is designed to the *25G & 50G Ethernet Specification, Draft 1.6* from the 25 Gigabit Ethernet Consortium and designed to the *IEEE 802.3by 25Gb Ethernet* specification, as well as the *IEEE 802.3ba-2012 High Speed Ethernet Standard* available on the IEEE website (www.ieee.org). The MAC provides RX cut-through frame processing to optimize latency. The IP supports the following features:

- PHY features:
  - Soft PCS logic that interfaces seamlessly to Intel Arria® 10 FPGA 25.78125 gigabits per second (Gbps) serial transceivers.
  - *IEEE 802.3-2018 Ethernet Standard Clause 108* optional soft Reed-Solomon forward error correction (RS-FEC).
- Frame structure control features:
  - Support for jumbo packets, defined as packets greater than 1500 bytes.
  - Receive (RX) CRC removal and pass-through control.
  - Transmit (TX) CRC generation and insertion.
  - RX and TX preamble pass-through option for applications that require proprietary user management information transfer.
  - TX automatic frame padding to meet the 64-byte minimum Ethernet frame length.
- Frame monitoring and statistics:
  - RX CRC checking and error reporting.
  - RX malformed packet checking per IEEE specification.
  - Optional statistics counters.
  - Optional fault signaling detects and reports local fault and generates remote fault, with *IEEE 802.3ba-2012 Ethernet Standard Clause 46* support.
  - Unidirectional transport as defined in *Clause 66 of the IEEE 802.3-2012 Ethernet Standard*.
- Flow control:
  - Standard *IEEE 802.3 Clause 31* and Priority-Based *IEEE 802.1Qbb* flow control.

- Precision Time Protocol support:
  - Optional support for the IEEE Standard 1588-2008 Precision Clock Synchronization Protocol (1588 PTP). This feature supports PHY operating speed with a constant timestamp accuracy of ± 3 ns and a dynamic timestamp accuracy of ± 1 ns.
- Debug and testability features:
  - Programmable serial PMA local loopback (TX to RX) at the serial transceiver for self-diagnostic testing.
  - TX error insertion capability.
  - Optional access to Native PHY Debug Master Endpoint (NPDME) for serial link debugging or monitoring PHY signal integrity.
- User system interfaces:
  - Avalon memory-mapped management interface to access the IP control and status registers.
  - Avalon streaming data path interface connects to client logic.
  - Configurable ready latency of 0 or 3 clock cycles for Avalon streaming TX interface.
  - Hardware and software reset control.

For a detailed specification of the Ethernet protocol refer to the *IEEE 802.3 Ethernet Standard*.

Related Information

IEEE website
The *IEEE 802.3 Ethernet Standard* is available on the IEEE website.

1.3. 25G Ethernet Intel FPGA IP Core Device Family and Speed Grade Support

1.3.1. Device Family Support

Table 2. Intel IP Core Device Support Levels

Device Support Level	Definition
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 3. 25G Ethernet Intel FPGA IP Core Device Family Support

Shows the level of support offered by the 25G Ethernet Intel FPGA IP core for each Intel device family.

Device Family	Support
Intel Arria 10 GT	Default support level provided in the Intel Quartus Prime software. Refer to the *Intel Quartus Prime Standard Edition Software and Device Support Release Notes* and the *Intel Quartus Prime Pro Edition Software and Device Support Release Notes*.
Other device families	Not supported.

Related Information

- Timing and Power Models
  Reports the default device support levels in the current version of the Intel Quartus Prime Standard Edition software.
- Timing and Power Models
  Reports the default device support levels in the current version of the Intel Quartus Prime Pro Edition software.

1.3.2. 25G Ethernet Intel FPGA IP Core Device Speed Grade Support

Table 4. Slowest Supported Device Speed Grades

IP Core	Device Family	Supported Speed Grades
25G Ethernet Intel FPGA IP	Intel Arria 10 GT	E2

1.4. IP Core Verification

To ensure functional correctness of the 25G Ethernet Intel FPGA IP core, Intel performs extensive validation through both simulation and hardware testing. Before releasing a version of the 25G Ethernet Intel FPGA IP core, Intel runs comprehensive regression tests in the current version of the Intel Quartus Prime software.

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

Related Information

- Knowledge Base Issues for IP core
  Exceptions to functional correctness are documented in the 25G Ethernet Intel FPGA IP core errata.
- 25G Ethernet Intel FPGA IP Release Notes
- Intel Quartus Prime Design Suite Update Release Notes
  Includes changes in minor releases (updates).

1.4.1. Simulation Environment

Intel performs the following tests on the 25G Ethernet Intel FPGA IP core in the simulation environment using internal and third-party standard bus functional models (BFM):

- Constrained random tests that cover randomized frame size and contents.
- Assertion based tests to confirm proper behavior of the IP core with respect to the specification.
- Extensive coverage of our runtime configuration space and proper behavior in all possible modes of operation.

1.4.2. Compilation Checking

Intel performs compilation testing on an extensive set of 25G Ethernet Intel FPGA IP core variations and designs to ensure the Intel Quartus Prime software places and routes the IP core ports correctly.

1.4.3. Hardware Testing

Intel performs hardware testing of the key functions of the 25G Ethernet Intel FPGA IP core using internal loopback and standard 25 Gbps Ethernet network test equipment. The hardware tests also ensure reliable solution coverage for hardware related areas such as performance, link synchronization, and reset recovery.

1.5. Performance and Resource Utilization

The following table shows the typical device resource utilization for selected configurations using the current version of the Intel Quartus Prime software. With the exception of M20K memory blocks, the numbers of ALMs and logic registers are rounded up to the nearest 100. The timing margin for this IP core is a minimum of 15%.

Table 5. IP Core Variation Encoding for Resource Utilization Table

"On" indicates the parameter is turned on. The symbol "—" indicates the parameter is turned off or not available.

IP Core Variation / Parameter	A	B	C
Read Latency	0	0	3
Enable RS-FEC	—	On	—
Enable flow control	—	Standard flow control, 1 queue	Standard flow control, 1 queue
Enable link fault generation	—	—	On
Enable preamble passthrough	—	—	On
Enable TX CRC passthrough	On	—	—
Enable MAC statistics counters	—	On	On
Enable IEEE 1588	—	—	On

Table 6. IP Core FPGA Resource Utilization for 25G Ethernet Intel FPGA IP Core for Intel Arria 10 Devices

Lists the resources and expected performance for selected variations of the 25G Ethernet Intel FPGA IP core.

These results were obtained using the Intel Quartus Prime software v19.4.

- The numbers of ALMs and logic registers are rounded up to the nearest 100.
- The numbers of ALMs, before rounding, are the **ALMs needed** numbers from the Intel Quartus Prime Fitter Report.

IP Core Variation	ALMs	Dedicated Logic Registers	M20K Memory Blocks
A	3100	7200	0
B	13300	30100	19
C	11400	25300	32

Related Information

- 25G Ethernet Intel FPGA IP Parameters on page 25
  Information about the parameters and values in the IP core variations.
- Fitter Resources Reports in the Quartus Prime Pro Edition Help

2. Getting Started

Related Information

- Introduction to Intel FPGA IP Cores
  Provides general information about all Intel FPGA IP cores, including parameterizing, generating, upgrading, and simulating IP cores.
- Creating Version-Independent IP and Qsys Simulation Scripts
  Create simulation scripts that do not require manual updates for software or IP version upgrades.
- Project Management Best Practices
  Guidelines for efficient management and portability of your project and IP files.

2.1. Installing and Licensing Intel FPGA IP Cores

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

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

Figure 3. IP Core Installation Path

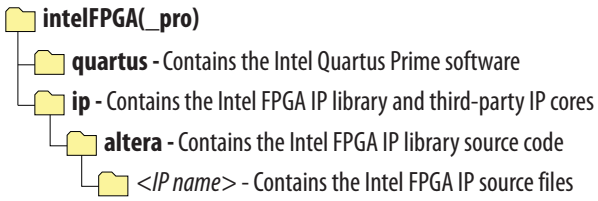


Table 7. IP Core Installation Locations

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

Note: The Intel Quartus Prime software does not support spaces in the installation path.

2.1.1. Intel FPGA IP Evaluation Mode

The free Intel FPGA IP Evaluation Mode allows you to evaluate licensed Intel FPGA IP cores in simulation and hardware before purchase. Intel FPGA IP Evaluation Mode supports the following evaluations without additional license:

- Simulate the behavior of a licensed Intel FPGA IP core in your system.
- Verify the functionality, size, and speed of the IP core quickly and easily.
- Generate time-limited device programming files for designs that include IP cores.
- Program a device with your IP core and verify your design in hardware.

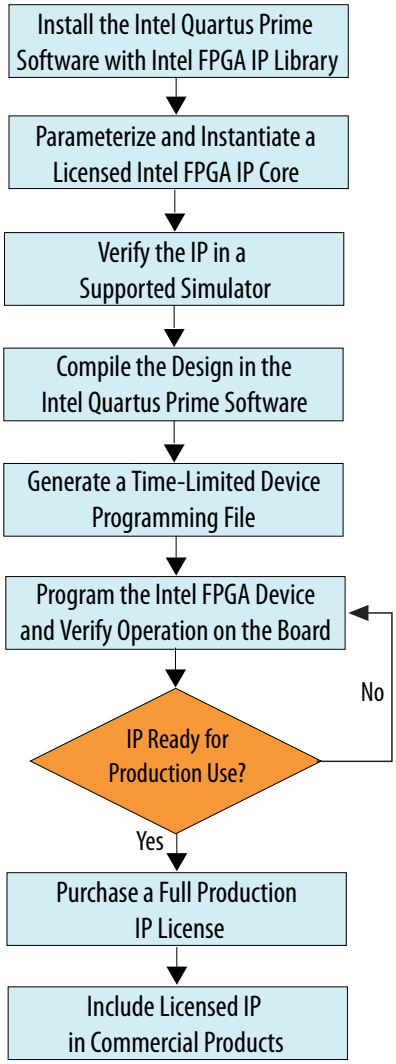
Intel FPGA IP Evaluation Mode supports the following operation modes:

- **Tethered**—Allows running the design containing the licensed Intel FPGA IP indefinitely with a connection between your board and the host computer. Tethered mode requires a serial joint test action group (JTAG) cable connected between the JTAG port on your board and the host computer, which is running the Intel Quartus Prime Programmer for the duration of the hardware evaluation period. The Programmer only requires a minimum installation of the Intel Quartus Prime software, and requires no Intel Quartus Prime license. The host computer controls the evaluation time by sending a periodic signal to the device via the JTAG port. If all licensed IP cores in the design support tethered mode, the evaluation time runs until any IP core evaluation expires. If all of the IP cores support unlimited evaluation time, the device does not time-out.
- **Untethered**—Allows running the design containing the licensed IP for a limited time. The IP core reverts to untethered mode if the device disconnects from the host computer running the Intel Quartus Prime software. The IP core also reverts to untethered mode if any other licensed IP core in the design does not support tethered mode.

When the evaluation time expires for any licensed Intel FPGA IP in the design, the design stops functioning. All IP cores that use the Intel FPGA IP Evaluation Mode time out simultaneously when any IP core in the design times out. When the evaluation time expires, you must reprogram the FPGA device before continuing hardware verification. To extend use of the IP core for production, purchase a full production license for the IP core.

You must purchase the license and generate a full production license key before you can generate an unrestricted device programming file. During Intel FPGA IP Evaluation Mode, the Compiler only generates a time-limited device programming file (`<project name>_time_limited.sof`) that expires at the time limit.

Figure 4. Intel FPGA IP Evaluation Mode Flow



Note: Refer to each IP core's user guide for parameterization steps and implementation details.

Intel licenses IP cores on a per-seat, perpetual basis. The license fee includes first-year maintenance and support. You must renew the maintenance contract to receive updates, bug fixes, and technical support beyond the first year. You must purchase a full production license for Intel FPGA IP cores that require a production license, before generating programming files that you may use for an unlimited time. During Intel FPGA IP Evaluation Mode, the Compiler only generates a time-limited device programming file (`<project name>_time_limited.sof`) that expires at the time limit. To obtain your production license keys, visit the Self-Service Licensing Center.

The Intel FPGA Software License Agreements govern the installation and use of licensed IP cores, the Intel Quartus Prime design software, and all unlicensed IP cores.

Related Information

- Intel FPGA Licensing Support Center
- Introduction to Intel FPGA Software Installation and Licensing

2.2. Specifying the 25G Ethernet Intel FPGA IP Core Parameters and Options

The 25G Ethernet Intel FPGA 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 software.

1. Depending on whether you are using the Intel Quartus Prime Pro Edition software or the Intel Quartus Prime Standard Edition software, perform one of the following actions:
   - 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.
   - In the Intel Quartus Prime Standard Edition software, in the IP Catalog (**Tools ➤ IP Catalog**), select the Arria 10 target device family.
2. In the IP Catalog (**Tools ➤ IP Catalog**), locate and double-click the name of the IP core to customize. The New IP Variation window appears.
3. In the **New IP Variation** dialog box, specify a top-level name for your custom IP variation. The parameter editor saves the IP variation settings in a file named *<your_ip>*`.qsys` (in Intel Quartus Prime Standard Edition) or *<your_ip>*`.ip` (in Intel Quartus Prime Pro Edition).
4. In the Intel Quartus Prime Standard Edition software, you must select a specific Intel Arria 10 device in the **Device** field, or keep the default device the Quartus Prime software proposes.
5. Click **OK**. The parameter editor appears.
6. On the **IP** tab, specify the parameters for your IP core variation. Refer to 25G Ethernet Intel FPGA IP Parameters on page 25 for information about specific IP core parameters.
7. Optionally, to generate a simulation testbench or compilation and hardware design example, follow the instructions in the *Intel Arria 10 25G Ethernet Design Example User Guide*.
8. Click **Generate HDL**. The **Generation** dialog box appears.
9. Specify output file generation options, and then click **Generate**. The IP variation files generate according to your specifications.

   Note: A functional VHDL IP core is not available. Specify Verilog HDL only, for your IP core variation.
10. Click **Finish**. The parameter editor adds the top-level `.qsys` or `.ip` file to the current project automatically. If you are prompted to manually add the `.qsys` or `.ip` file to the project, click **Project ➤ Add/Remove Files in Project** to add the file.
11. After generating and instantiating your IP variation, make appropriate pin assignments to connect ports.

Related Information

Intel Arria 10 25G Ethernet Design Example User Guide
Information about the **Example Design** tab in the 25G Ethernet Intel FPGA IP parameter editor.

2.3. Simulating the IP Core

You can simulate your 25G Ethernet Intel FPGA IP core variation with the functional simulation model and the testbench generated with the IP core. The functional simulation model is a cycle-accurate model that allows for fast functional simulation of your IP core instance using industry-standard Verilog HDL simulators. You can simulate the Intel-provided testbench or create your own testbench to exercise the IP core functional simulation model.

The functional simulation model and testbench files are generated in project subdirectories. These directories also include scripts to compile and run the design example.

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

In the top-level wrapper file for your simulation project, you can set the the following RTL parameters to enable simulation optimization. These optimizations significantly decrease the time to reach link initialization.

- SIM_SHORT_RST: Shortens the reset times to speed up simulation.
- SIM_SHORT_AM: Shortens the interval between alignment markers to accelerate alignment marker lock. Alignment markers are used when Reed-Solomon FEC is enabled.
  - SIM_SHORT_AM = 1'b1: The TX RS-FEC inserts alignment marker at every 1280 64b/66b blocks or 320 257-bit transcoded blocks. The RX RS-FEC expects alignment marker at every 1280 64b/66b blocks or 320 257-bit transcoded blocks.
  - SIM_SHORT_AM = 1'b0: The TX RS-FEC inserts alignment marker at every 81920 64b/66b blocks or 20480 257-bit transcoded blocks. The RX RS-FEC expects alignment marker at every 81920 64b/66b blocks or 20480 257-bit transcoded blocks.
- SIM_SIMPLE_RATE: Sets the PLL reference clock (`clk_ref`) to 625 MHz instead of 644.53125 MHz to optimize PLL simulation model behavior

In general, parameters are set through the IP core parameter editor and you should not change them manually. The only exceptions are these simulation optimization parameters.

To set these parameters on the PHY blocks, add the following lines to the top-level wrapper file:

```
defparam <dut instance>.SIM_SHORT_RST = 1'b1;
defparam <dut instance>.SIM_SHORT_AM = 1'b1;
defparam <dut instance>.SIM_SIMPLE_RATE = 1'b1;
```

Note: You can use the example testbench as a guide for setting the simulation parameters in your own simulation environment. These lines are already present in the Intel-provided testbench for the IP core.

Related Information

- Simulating Intel FPGA Designs
  Quartus Prime Standard Edition Handbook Volume 3: Verification chapter that provides information about simulating Intel FPGA IP cores.
- Intel Arria 10 25G Ethernet Design Example User Guide
  Information about generating and simulating the Intel-provided 25G Ethernet Intel FPGA IP testbench. This testbench demonstrates a basic test of the IP core. It is not intended to be a substitute for a full verification environment.

2.4. Generated File Structure

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

For information about the file structure of the design example, refer to the *Arria 10 25G Ethernet Intel FPGA IP Design Example User Guide*.

Figure 5. IP Core Generated Files

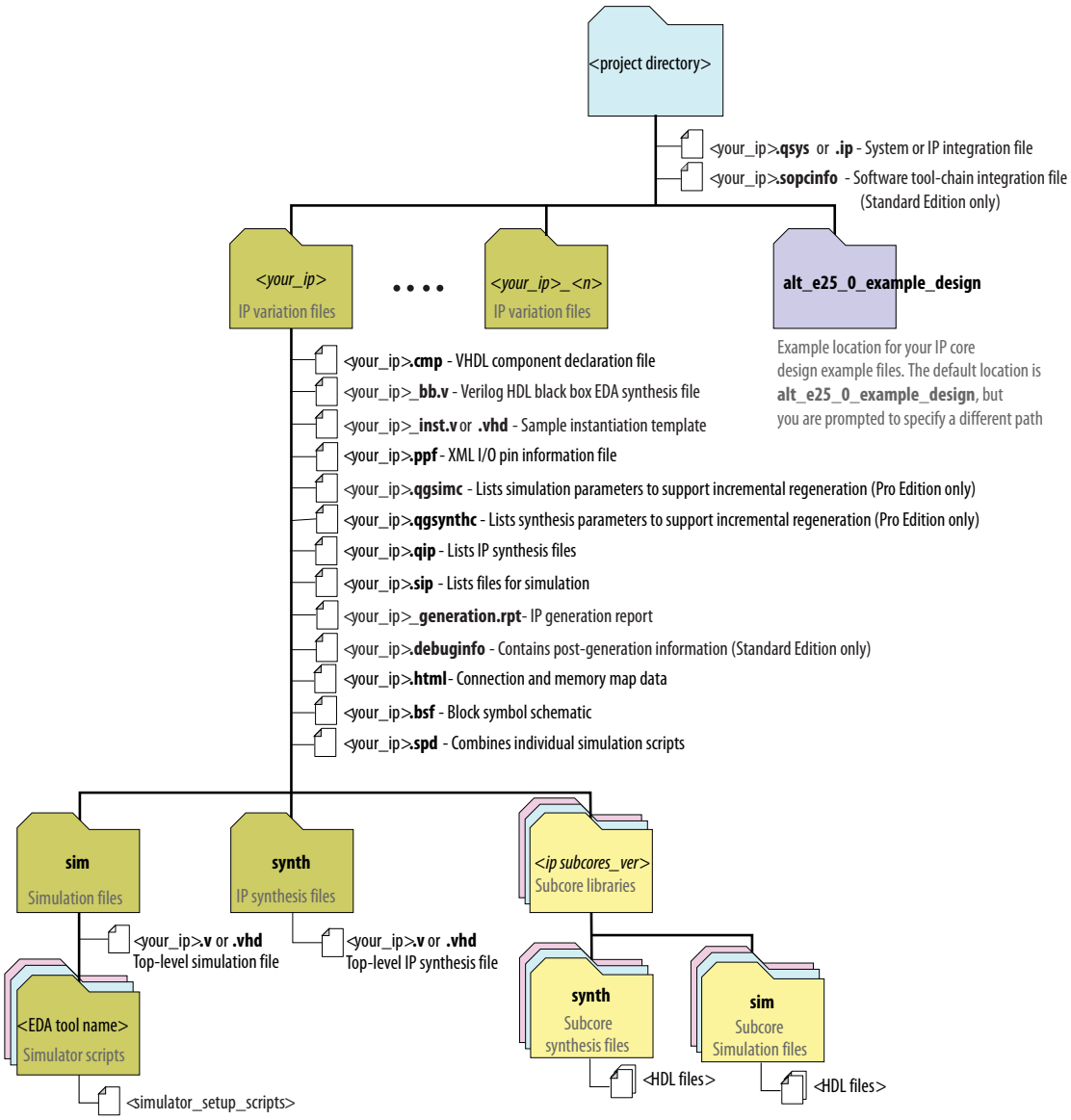


Table 8. IP Core Generated Files

File Name	Description
<*your_ip*>.qsys (Intel Quartus Prime Standard Edition only)	The Platform Designer system or top-level IP variation file. <*your_ip*> is the name that you give your IP variation.
<*your_ip*>.ip (Intel Quartus Prime Pro Edition only)	
<*system*>.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. (Intel Quartus Prime Standard Edition only)

continued...

File Name	Description
	Downstream tools such as the Nios® II Gen 2 tool chain use this file. The `.sopcinfo` file and the `system.h` file generated for the Nios II Gen 2 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.
<*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. This IP core does not support VHDL. However, the Intel Quartus Prime software generates this file.
<*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*>.debuginfo	Contains post-generation information. Used to pass System Console and Bus Analyzer Toolkit information about the Platform Designer interconnect. The Bus Analysis Toolkit uses this file to identify debug components in the Platform Designer interconnect. (Intel Quartus Prime Standard Edition only)
<*your_ip*>.qgsimc	Lists simulation parameters to support incremental regeneration. (Intel Quartus Prime Pro Edition only)
<*your_ip*>.qgsynthc	Lists synthesis parameters to support incremental regeneration. (Intel Quartus Prime Pro Edition only)
<*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*>.csv	Contains information about the upgrade status of the IP component.
<*your_ip*>.bsf	A Block Symbol File (.**bsf**) representation of the IP variation for use in Intel Quartus Prime Block Diagram Files (**.bdf**).
<*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*>.ppf	The Pin Planner File (**.ppf**) stores the port and node assignments for IP components created for use with the Pin Planner.
<*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*>.sip	Contains information required for NativeLink simulation of IP components. You must add the **.sip** file to your Quartus Prime project. (Intel Quartus Prime Standard Edition only)
<*your_ip*>_inst.v and _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. This IP core does not support VHDL. However, the Intel Quartus Prime software generates the `_inst.vhd` file.
<*your_ip*>.regmap	If IP contains register information, **.regmap** file generates. The **.regmap** file describes the register map information of master and slave interfaces. This file complements the **.sopcinfo** file by providing more detailed register information about the system. This enables register display views and user customizable statistics in the System Console.
<*your_ip*>.svd	Allows hard processor system (HPS) System Debug tools to view the register maps of peripherals connected to HPS within 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.

continued...

File Name	Description
<*your_ip*>.v and <*your_ip*>.vhd	HDL files that instantiate each submodule or child IP core for synthesis or simulation. This IP core does not support VHDL. However, the Intel Quartus Prime software generates this file.
mentor/	Contains a ModelSim script msim_setup.tcl to set up and run a simulation.
aldec/	Contains a Riviera-PRO script rivierapro_setup.tcl to setup 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.
cadence/	Contains a shell script ncsim_setup.sh and other setup files to set up and run an NCSIM simulation.
submodules/	Contains HDL files for the IP core submodule.
<*child IP cores*>/	For each generated child IP core directory, Platform Designer generates synth/ andsim/ sub-directories.

Related Information

Intel Arria 10 25G Ethernet Design Example User Guide
Information about the 25G Ethernet Intel FPGA IP design example file structure.

2.5. Integrating Your IP Core in Your Design

2.5.1. Pin Assignments

When you integrate your 25G Ethernet Intel FPGA IP core instance in your design, you must make appropriate pin assignments. While compiling the IP core alone, you can create virtual pins to avoid making specific pin assignments for top-level signals. When you are ready to map the design to hardware, you can change to the correct pin assignments.

Related Information

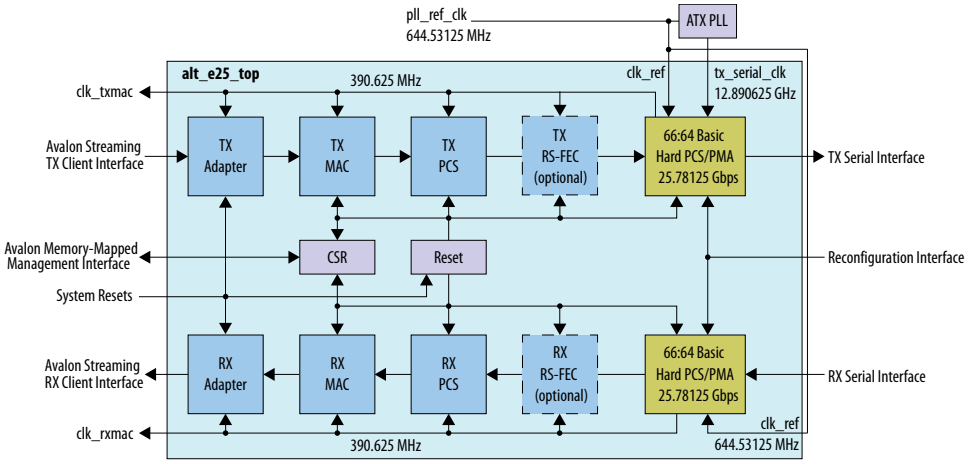
Intel Quartus Prime Help
For information about the Intel Quartus Prime software, including virtual pins.

2.5.2. Adding the Transceiver PLL

The 25G Ethernet Intel FPGA IP core targets Arria 10 GT devices. Intel Arria 10 GT devices require an external PLL to drive the TX transceiver serial clock, in order to compile and to function correctly in hardware. In many cases, the same PLL can be shared with an additional transceiver in your design.

Figure 6. PLL Configuration Example

The TX transceiver PLL is instantiated with an ATX PLL IP core. The TX transceiver PLL must always be instantiated outside the 25G Ethernet Intel FPGA IP core.



You can use the IP Catalog to create a transceiver PLL.

- Select **Intel Arria 10 Transceiver ATX PLL**.
- In the parameter editor, set the following parameter values:
  - **PLL output frequency** to **12890.625 MHz**. The transceiver performs dual edge clocking, using both the rising and falling edges of the input clock from the PLL. Therefore, this PLL output frequency setting supports a 25.78125 Gbps data rate through the transceiver.
  - **PLL reference clock frequency** to **644.53125** MHz.

You must connect the ATX PLL to the 25G Ethernet Intel FPGA IP core as follows:

- Connect the clock output port of the ATX PLL to the `tx_serial_clk` input port of the 25G Ethernet Intel FPGA IP core.
- Connect the `pll_locked` output port of the ATX PLL to the `tx_pll_locked` input port of the 25G Ethernet Intel FPGA IP core.
- Drive the ATX PLL reference clock port and the 25G Ethernet Intel FPGA IP core `clk_ref` input port with the same clock. The clock frequency must be the frequency you specify for the ATX PLL IP core **PLL reference clock frequency** parameter.

Related Information

- Transceivers on page 58
- Intel Arria 10 Transceiver PHY User Guide
  Information about the correspondence between PLLs and transceiver channels, and information about how to configure an external transceiver PLL for your own design. You specify the clock network to which the PLL output connects by setting the clock network in the PLL parameter editor.

2.5.3. Handling Potential Jitter in Intel Arria 10 Devices

The RX path in the 25G Ethernet Intel FPGA IP core includes cascaded PLLs. Therefore, the IP core clocks might experience additional jitter in Intel Arria 10 devices.

Refer to the KDB Answer *How do I compensate for the jitter of PLL cascading or non-dedicated clock path for Arria 10 PLL reference clock?* for a workaround you should apply to the IP core, in your design.

Related Information

https://www.altera.com/support/support-resources/knowledge-base/tools/2017/fb470823.html

KDB Answer: How do I compensate for the jitter of PLL cascading or non-dedicated clock path for Arria 10 PLL reference clock?

2.5.4. Adding the External Time-of-Day Module for Variations with 1588 PTP Feature

25G Ethernet Intel FPGA IP cores that include the 1588 PTP module require an external time-of-day (TOD) module to provide a continuous flow of current time-of-day information. The TOD module must update the time-of-day output value on every clock cycle, and must provide the TOD value in the V2 format (96 bits) or the 64-bit TOD format, or both.

Intel provides the following components that you can combine to create the TOD module the 25G Ethernet Intel FPGA IP core requires:

- A simple TOD clock module, available from the IP Catalog (**Interface Protocols > Ethernet > Reference Design Components > Ethernet IEEE 1588 Time of Day Clock Intel FPGA IP**). You can instantiate two of these clock modules and connect one to the TX MAC and the other to the RX MAC.
- A single-format TOD synchronizer, available from the IP Catalog (**Interface Protocols > Ethernet > Reference Design Components > Ethernet IEEE 1588 TOD Synchronizer Intel FPGA IP**). This component can handle only a single TOD format. Therefore, if you set the **Time of day format** parameter to the value of **Enable both formats**, you must instantiate and connect two TOD synchronizer modules. If your IP core supports only a single TOD format, your design requires only a single TOD synchronizer module.

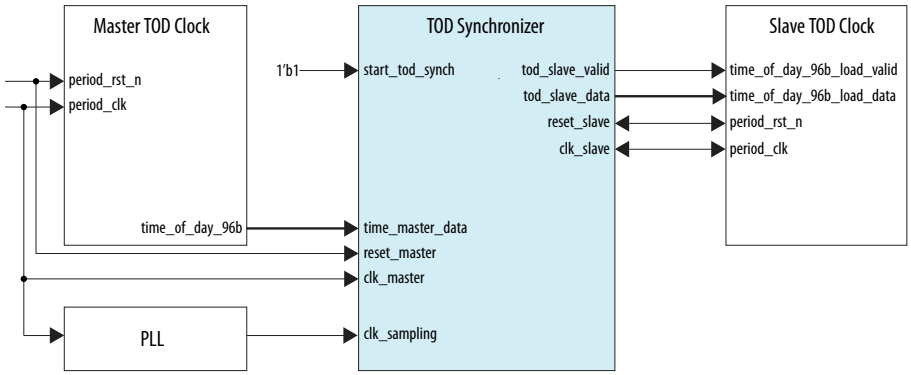
Each TOD synchronizer connects a master TOD clock and a slave TOD clock.

- If you create your TOD module with a single TOD synchronizer, the master TOD clock connects to the TX MAC of the 25G Ethernet Intel FPGA IP core and the slave TOD clock connects to the RX MAC of the 25G Ethernet Intel FPGA IP core.
- Alternatively, you can drive both the TX and RX TOD clocks from a single master TOD clock. In that case, your design must include two TOD synchronizers, one to connect the master TOD clock and the slave TX TOD clock and one to connect the master TOD clock and the slave RX TOD clock.

If your IP core supports both TOD formats, double the number of TOD synchronizers in your TOD module. The configuration you implement depends on your system design requirements for 1588 PTP functionality.

Figure 7. TOD Synchronizer and TOD Clocks in 96-Bit TOD Format Design

Shows the required connections between two TOD clock components and a TOD synchronizer component in a single TOD format design. In a simple TOD module, the master TOD clock connects to the TX MAC of the IP core, and the slave TOD clock connects to the RX MAC of the IP core. If your 25G Ethernet Intel FPGA IP core supports both TOD formats, a second TOD synchronizer connects to the corresponding 64-bit time-of-day signals of the same master and slave TOD clocks.



For information about the Ethernet IEEE 1588 Time of Day Clock and Ethernet IEEE 1588 TOD Synchronizer components, and the requirements for the PLL that connects to the TOD synchronizer, refer to the *Ethernet Design Example Components User Guide*.

Table 9. TOD Module Required Connections to 25G Ethernet Intel FPGA IP Core

Lists the required connections between the TOD module and the 25G Ethernet Intel FPGA IP core, using signal names for TOD modules that provide both a 96-bit TOD and a 64-bit TOD. If you create your own TOD module it must have the output signals required by the 25G Ethernet Intel FPGA IP core. However, its signal names could be different than the TOD module signal names in the table. The signals that the IP core includes depend on the value you set for **Time of day format** in the parameter editor. For example, an RX TOD module might require only a 96-bit TOD out signal. This table does not list required connections between the TOD module and additional parts of your design.

TOD Module Signal	25GbE IP Core Signal
`rst_n` (input to TX and RX TOD clocks)	Drive this signal from the same source as the `csr_rst_n` input signal to the 25G Ethernet Intel FPGA IP core.
`period_rst_n` (input to RX TOD clock) `reset_slave` (input to Synchronizer)	Drive these signals from the same source as the `rx_rst_n` input signal to the 25G Ethernet Intel FPGA IP core.
`period_rst_n` (input to TX TOD clock) `reset_master` (input to Synchronizer)	Drive these signals from the same source as the `tx_rst_n` input signal to the 25G Ethernet Intel FPGA IP core.
`time_of_day_96b[95:0]` (output from TX TOD clock)	`tx_time_of_day_96b_data[95:0]` (input)
`time_of_day_64b[63:0]` (output from TX TOD clock)	`tx_time_of_day_64b_data[63:0]` (input)
`time_of_day_96b[95:0]` (output from RX TOD clock)	`rx_time_of_day_96b_data[95:0]` (input)
`time_of_day_64b[63:0]` (output from RX TOD clock)	`rx_time_of_day_64b_data[63:0]` (input)
`period_clk` (input to TX TOD clock) `clk_master` (input to Synchronizer)	`clk_txmac` (output)
`period_clk` (input to RX TOD clock) `clk_slave` (input to Synchronizer)	`clk_rxmac` (output)

Related Information

- External Time-of-Day Module for 1588 PTP Variations on page 48

- Ethernet Design Example Components User Guide
  Describes the Ethernet IEEE 1588 Time of Day Clock component and the Ethernet IEEE 1588 TOD Synchronizer component available in the Intel Quartus Prime software from the IP Catalog.

2.5.5. Placement Settings for the 25G Ethernet Intel FPGA IP Core

The Quartus Prime software provides the options to specify design partitions and Logic Lock (Standard) or Logic Lock regions for incremental compilation, to control placement on the device. To achieve timing closure for your design, you might need to provide floorplan guidelines using one or both of these features.

The appropriate floorplan is always design-specific, and depends on your design.

Related Information

- Intel Quartus Prime Standard Edition User Guide: Design Constraints
  Describes incremental compilation, design partitions, and Logic Lock (Standard) regions.
- Intel Quartus Prime Pro Edition User Guide: Design Constraints
  Describes incremental compilation, design partitions, and Logic Lock regions.

2.6. Compiling the Full Design and Programming the FPGA

You can use the **Start Compilation** command on the Processing menu in the Intel Quartus Prime software to compile your design. After successfully compiling your design, program the targeted Intel FPGA with the Programmer and verify the design in hardware.

Note: The 25G Ethernet Intel FPGA IP core design example synthesis directories include Synopsys Constraint (`.sdc`) files that you can copy and modify for your own design.

Note: For additional `.sdc` file requirements, please refer to the KDB Answer at https://www.altera.com/support/support-resources/knowledge-base/tools/2017/fb470823.html.

Related Information

- Incremental Compilation for Hierarchical and Team-Based Design
- Programming Intel Devices
- 25G Ethernet Intel Arria 10 FPGA IP Design Example User Guide
  Information about generating the design example and the design example directory structure.

3. 25G Ethernet Intel FPGA IP Parameters

The 25G Ethernet Intel FPGA IP parameter editor provides the parameters you can set to configure the 25G Ethernet Intel FPGA IP core and simulation testbenches.

The 25G Ethernet Intel FPGA IP parameter editor includes an **Example Design** tab. For information about that tab, refer to the *25G Ethernet Design Example User Guide*.

Table 10. IP Core Parameters

Parameter	Range	Default Setting	Description
General Options			
Device Family	**Arria 10**	**Arria 10**	Selects the device family.
Ready Latency	**0**, **3**	0	Selects the readyLatency value on the TX client interface. readyLatency is an Avalon streaming interface property that defines the number of clock cycles of delay from when the IP core asserts the `l1_tx_ready` signal to the clock cycle in which the IP core can accept data on the TX client interface. Refer to the *Avalon Interface Specifications*. Selecting a latency of 3 eases timing closure at the expense of increased latency for the datapath. If you set the readyLatency to 3 and turn on standard flow control, data might be delayed in the IP core while the IP core is backpressured.
PCS/PMA Options			
Enable RS-FEC	**Enabled**, **Disabled**	**Disabled**	When enabled, the IP core implements Reed-Solomon forward error correction (FEC). This parameter is not available if you turn on **Enable IEEE 1588**.
Flow Control Options			
Enable flow control	**Enabled**, **Disabled**	**Disabled**	When enabled, the IP core implements flow control. When either link partner experiences congestion, the respective transmit control sends pause frames. Register settings control flow control behavior, including whether the IP core implements standard flow control or priority-based flow control. If you turn on standard flow control and set the readyLatency to 3, data might be delayed in the IP core while the IP core is backpressured.
Number of queues	**1**-**8**	**8**	Specifies the number of queues used in managing flow control.
MAC Options			
Enable link fault generation	**Enabled**, **Disabled**	**Disabled**	When enabled, the IP core implements link fault signaling as defined in the *IEEE 802.3-2012 IEEE Standard for Ethernet*. The MAC includes a Reconciliation Sublayer (RS) to manage local and remote faults. When enabled, the local RS TX logic can

continued...

Parameter	Range	Default Setting	Description
			transmit remote fault sequences in case of a local fault and can transmit IDLE control words in case of a remote fault.
Enable preamble passthrough	**Enabled**, **Disabled**	**Disabled**	When enabled, 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 Start Frame Delimiter (SFD) to the client instead of stripping them out of the Ethernet packet. In TX preamble pass-through mode, the client specifies the preamble and provides the SFD to be sent in the Ethernet frame.
Enable TX CRC passthrough	**Enabled**, **Disabled**	**Disabled**	When enabled, TX MAC does not insert the CRC-32 checksum in the out-going frame. In pass-through mode, the client must provide frames with at least 64 bytes, including the Frame Check Sequence (FCS). When disabled, the TX MAC computes and inserts a 32-bit FCS in the TX MAC frame. This parameter is not available if you turn on **Enable IEEE 1588**.
Enable MAC statistics counters	**Enabled**, **Disabled**	**Enabled**	When enabled, the IP core includes statistics counters that characterize TX and RX traffic.
IEEE 1588 Options			
Enable IEEE 1588	**Enabled**, **Disabled**	**Disabled**	If enabled, the IP core supports the IEEE Standard 1588-2008 Precision Clock Synchronization Protocol, by providing the hooks to implement the Precise Timing Protocol (PTP). This parameter is not available if you turn on **Enable TX CRC passthrough**. This parameter is not available if you turn on **Enable RS-FEC**.
Time of day format	**Enable 96-bit timestamp format**, **Enable 64-bit timestamp format**, **Enable both formats**	**Enable both formats**	Specifies the interface to the Time of Day module. If you select **Enable both formats**, the IP core includes both the 64-bit interface and the 96-bit interface. This parameter is available only in variations with **Enable IEEE 1588** turned on. The IP core provides the Time of Day interface; the IP core does not include Time of Day and synchronizer modules to connect to this interface.
Fingerprint width	**1–32**	**4**	Specifies the number of bits in the fingerprint that the IP core handles. This parameter is available only in variations with **Enable IEEE 1588** turned on.
Configuration, Debug and Extension Options			
Enable Native PHY Debug Master Endpoint (NPDME)	**Enabled**, **Disabled**	**Disabled**	If enabled, the IP core turns on the following features in the Intel Arria 10 PHY IP core that is included in the 25G Ethernet Intel FPGA IP core: • **Enable Native PHY Debug Master Endpoint (NPDME)** • **Enable capability registers** If turned off, the IP core is configured without these features. For information about these Intel Arria 10 features, refer to the *Intel Arria 10 Transceiver PHY User Guide*. .

Related Information

- 25G Ethernet Design Example User Guide
  Information about the **Example Design** tab in the 25G Ethernet Intel FPGA IP parameter editor.

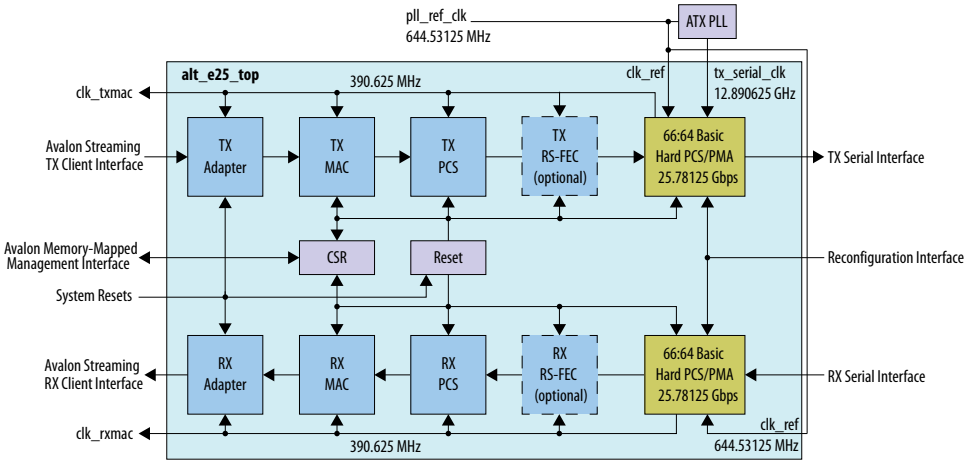
- Avalon Interface Specifications
  Detailed information about Avalon streaming interfaces and the Avalon streaming readyLatency parameter.
- Intel Arria 10 Transceiver PHY User Guide
  Information about Intel Arria 10 Native PHY IP core features, including NPDME.

4. Functional Description

4.1. 25G Ethernet Intel FPGA IP Core Functional Description

The 25G Ethernet Intel FPGA IP core implements an Ethernet MAC in accordance with the *25G & 50G Ethernet Specification*. The IP core implements an Ethernet PCS and PMA (PHY) that handles the frame encapsulation and flow of data between a client logic and Ethernet network.

Figure 8. 25G Ethernet Intel FPGA IP Core MAC and PHY Clock Diagram



In the TX direction, the MAC assembles packets and sends them to the PHY. It completes the following tasks:

- Accepts client frames.
- Inserts the inter-packet gap (IPG), preamble, start of frame delimiter (SFD), and padding. The source of the preamble and SFD depends on whether the IP core is in preamble-pass-through mode.
- Adds the CRC bits if enabled.
- Updates statistics counters if enabled.

The PHY encodes MAC frames for reliable transmission over the media to the remote end.

In the RX direction, the PMA passes frames to the PCS that sends them to the MAC. The MAC completes the following tasks:

- Performs CRC and malformed packet checks.
- Updates statistics counters if enabled.
- Strips out the CRC, preamble, and SFD.
- Passes the remainder of the frame to the client.

In preamble pass-through mode, the MAC passes on the preamble and SFD to the client instead of stripping them out. In RX CRC pass-through mode, the MAC passes on the CRC bytes to the client and asserts the end-of-packet signal in the same clock cycle as the final CRC byte.

4.1.1. 25G Ethernet Intel FPGA IP Core TX MAC Datapath

The TX MAC module receives the client payload data with the destination and source addresses. It then adds, appends, or updates various header fields in accordance with the configuration specified. The MAC does not modify the destination address, the source 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. It pads the payload of frames greater than eight bytes to satisfy the minimum Ethernet frame payload of 46 bytes. By default, the MAC inserts the CRC bytes. The TX MAC module inserts IDLE bytes to maintain an average IPG of 12.

Figure 9. Typical Client Frame at the Transmit Interface

Illustrates the changes that the TX MAC makes to the client frame. This figure uses the following notational conventions:

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

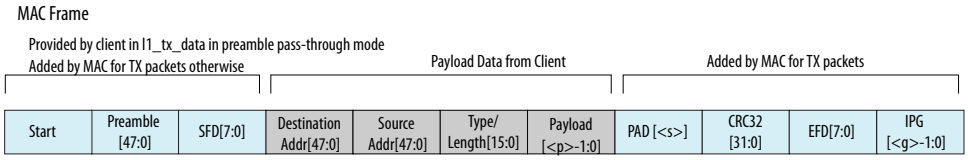
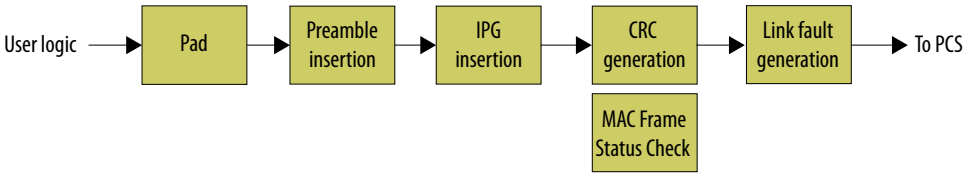


Figure 10. TX MAC Functions



4.1.1.1. Frame Padding

When the length of the client frame is less than 64 bytes, the TX MAC module inserts pad bytes (0x00) after the payload to create a frame length equal to the minimum size of 64 bytes (including CRC).

The IP core filters out all client frames with lengths less than 9 bytes. The IP core drops these frames silently.

4.1.1.2. Preamble Insertion

In the TX datapath the MAC prepends an eight-byte preamble to the client frame. If you turn on **Enable link fault generation**, this MAC module also incorporates the functions of the reconciliation sublayer (RS).

The source of the 7-byte preamble (including a Start byte) and 1-byte SFD depends on whether you turn on **Enable preamble passthrough** in the parameter editor.

If the preamble pass-through feature is enabled, the client provides the eight-byte preamble (including the 0xFB Start byte and final 1-byte SFD) on `l1_tx_data`. The client is responsible for providing the correct Start byte (0xFB) and an appropriate SFD byte. If the preamble pass-through feature is disabled, the MAC inserts the standard Ethernet preamble in the transmitted Ethernet frame.

Note that a single parameter in the 25G Ethernet Intel FPGA IP parameter editor turns on both RX and TX preamble passthrough.

4.1.1.3. Inter-Packet Gap Generation and Insertion

The TX MAC maintains the minimum inter-packet gap (IPG) between transmitted frames required by the IEEE 802.3 Ethernet standard. The deficit idle counter (DIC) maintains the average IPG of 12 bytes.

4.1.1.4. Frame Check Sequence (CRC32) Insertion

The component GUI includes the **Enable TX CRC passthrough** parameter to control CRC generation. When enabled, TX MAC does not insert the CRC32 checksum in the out-going frame. In pass-through mode, the client must provide frames with at least 64 bytes, so that the IP core does not pad them. When disabled, the TX MAC computes and inserts a 32-bit Frame Check Sequence (FCS) in the TX MAC frame. The MAC computes the CRC32 over the frame bytes that include the source address, destination address, length, data, and pad (if applicable). The CRC checksum computation excludes the preamble, SFD, and FCS.

In pass-through mode, the `l1_tx_endofpacket`, `l1_rx_endofpacket`, `l1_tx_empty[2:0]`, and `l1_rx_empty` are asserted in the same clock cycle with the final FCS byte. When pass-through mode is disabled, the `l1_tx_endofpacket`, `l1_rx_endofpacket`, `l1_tx_empty[2:0]`, and `l1_rx_empty` are asserted in the same clock cycle with the byte before the first FCS bytes.

The encoding is defined by the following generating polynomial:

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

CRC bits are transmitted with MSB first.

Note that you control whether the IP core implements TX CRC insertion or passthrough with a parameter in the 25G Ethernet Intel FPGA IP parameter editor. You control RX CRC forwarding dynamically with the `MAC_CRC_CONFIG` register.

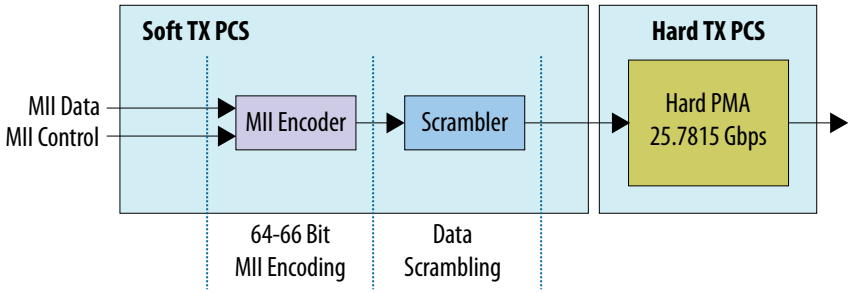
Related Information

Order of Transmission on page 50

4.1.2. 25 GbE TX PCS

The soft TX PCS implements MII encoding and scrambling. The 66-bit output stream is input to the hard PCS and PMA block.

Figure 11. High Level Block Diagram of the Soft TX PCS



The Hard PCS and PMA blocks are configured in 66:64 bit basic generic 10G PCS mode whose status can be read through Control and Status registers. These blocks use FIFOs in elastic-buffer mode. The PMA operates at 25.78125 Gbps.

Related Information

Ethernet section of the Intel Arria 10 Transceiver PHY User Guide
Provides more information about the PMA and PCS for Ethernet protocols.

4.1.2.1. TX RSFEC

If you turn on **Enable RS-FEC** in the 25G Ethernet Intel FPGA IP parameter editor, the IP core includes Reed-Solomon forward error correction (FEC) in both the receive and transmit datapaths.

The IP core implements Reed-Solomon FEC per Clause 108 of the IEEE Standard 802.3by. The Reed-Solomon FEC algorithm includes the following modules:

- 64B/66B to 256B/257B Transcoding
- 257:80 gearbox
- High-Speed Reed-Solomon Encoder
- 80:66 gearbox

You cannot turn on both **Enable RS-FEC** and **Enable IEEE 1588** in the 25G Ethernet Intel FPGA IP parameter editor.

4.1.3. 25G Ethernet Intel FPGA IP Core RX MAC Datapath

The RX MAC receives Ethernet frames and forwards the payload with relevant header bytes to the client after performing some MAC functions on header bytes. The RX MAC processes all incoming valid frames.

Figure 12. Flow of Client Frame With Preamble Pass-Through Turned On

This figure uses the following notational conventions:

- <*p*> = payload size, which is arbitrarily large.
- <*s*> = number of padding bytes (0–46).

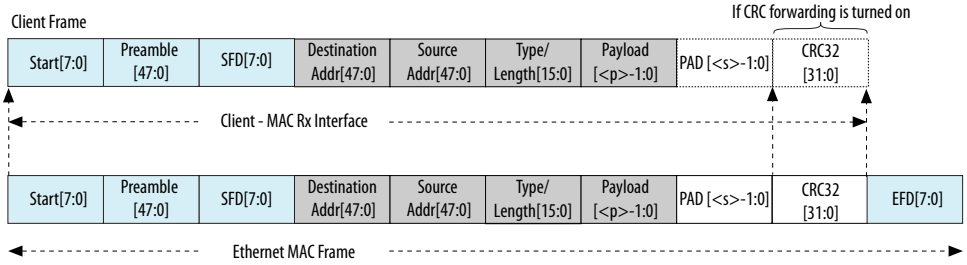


Figure 13. Flow of Client Frame With Preamble Pass-Through Turned Off

This figure uses the following notational conventions:

- <*p*> = payload size, which is arbitrarily large.
- <*s*> = number of padding bytes (0–46).

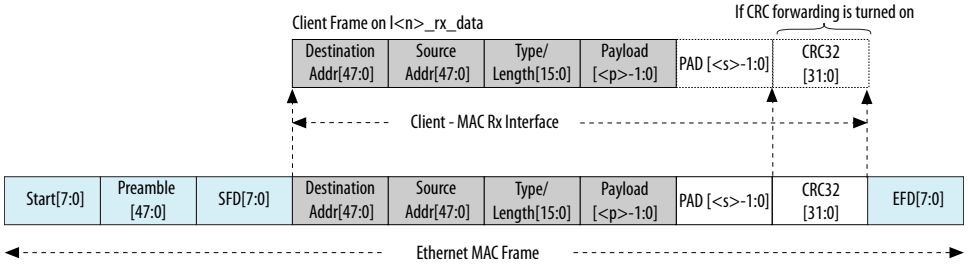
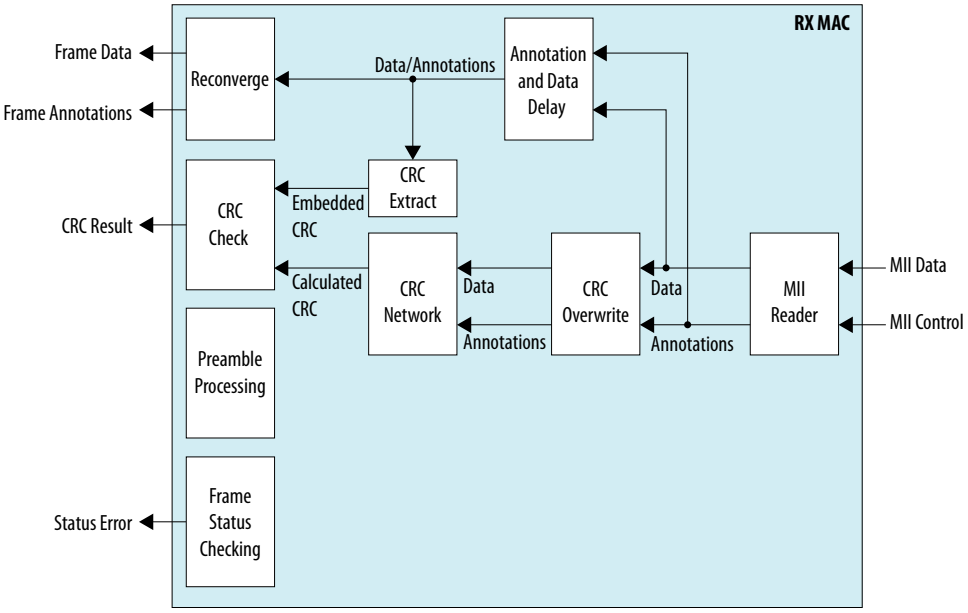


Figure 14. RX MAC Datapath



4.1.3.1. IP Core Preamble Processing

If you turn on **Enable preamble passthrough** in the parameter editor, the RX MAC forwards preamble bytes. The TX MAC requires the preamble bytes to be included in the frames at the Avalon Streaming interface.

If you turn off **Enable preamble passthrough**, the IP core removes the preamble bytes. `l1_rx_startofpacket` is aligned to the MSB of the destination address.

Note that a single parameter in the 25G Ethernet Intel FPGA IP parameter editor turns on both RX and TX preamble passthrough.

4.1.3.2. IP Core Malformed Packet Handling

While receiving an incoming packet from the Ethernet link, the 25G Ethernet Intel FPGA IP core expects to detect a terminate character at the end of the packet. When it detects an expected terminate character, the IP core generates an EOP on the client interface. However, sometimes the IP core detects an unexpected control character when it expects a terminate character.

If the 25G Ethernet Intel FPGA IP core detects an Error character, a Start character, an IDLE character, or any other non-terminate control character, when it expects a terminate character, it performs the following actions:

- Generates an EOP.
- Asserts a malformed packet error (`l1_rx_error[0]`).
- Asserts an FCS error (`l1_rx_error[1]`).

If the IP core subsequently detects a terminate character, it does not generate another EOP indication.

When the IP core receives a packet that contains an error deliberately introduced on the Ethernet link using the 25G Ethernet Intel FPGA IP TX error insertion feature, the IP core identifies it as a malformed packet.

At this time, the 25G Ethernet Intel FPGA IP core does not recognize non-zero 4-bit ordered set types as an error.

4.1.3.3. Length/Type Field Processing

This two-byte header represents either the length of the payload or the type of MAC frame.

- Length/type < 0x600—The field represents the payload length of a basic Ethernet frame. The MAC RX continues to check the frame and payload lengths.
- Length/type >= 0x600—The field represents the frame type. The following frame types are possible:
  - Length/type = 0x8100—VLAN or stacked VLAN tagged frames (up to a total of two tags with value 0x8100). The MAC RX continues to check the frame and payload lengths.
  - Length/type = 0x8808—Control frames. The next two bytes are the Opcode field that indicates the type of control frame. For pause frames (Opcode = 0x0001) and PFC frames (Opcode = 0x0101), the MAC RX proceeds with pause frame processing. In addition to processing any pause request, the IP core passes these frames to the RX client interface and updates the appropriate `l1_rxstatus_data` bits.
  - For other field values, the MAC RX forwards the receive frame to the client.

4.1.3.3.1. Length Checking

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

The IP core checks that the frame length is valid—is neither undersized nor oversized. A valid frame length is at least 64 (0x40) bytes and does not exceed the following maximum value for the different frame types:

- Basic frames—The number of bytes specified in the `MAX_RX_SIZE_CONFIG` register.
- VLAN tagged frames—The value specified in the `MAX_RX_SIZE_CONFIG` register plus four bytes.
- Stacked VLAN tagged frames—The value specified in the `MAX_RX_SIZE_CONFIG` register plus eight bytes.

If the length/type field in a basic MAC frame or the client length/type field in a VLAN tagged frame has a value less than 0x600, the IP core also checks the payload length. The IP core keeps track of the payload length as it receives a frame, and checks the length against the relevant frame field. The payload length is valid if it satisfies the following conditions:

- The actual payload length matches the value in the length/type or client length/type field.
- Normal frames:
  - Basic frames—the payload length is between 46 (0x2E)and 1536 (0x0600) bytes, excluding 1536.
  - VLAN tagged frames—the payload length is between 42 (0x2A)and 1536 (0x0600), excluding 1536.
  - Stacked VLAN tagged frames—the payload length is between 38 (0x26) and 1536 (0x0600), excluding 1536.
- Jumbo frames:
  - Jumbo basic frames—the payload length is between 46 (0x2E) and the value specified in the MAX_RX_SIZE_CONFIG register minus 18 bytes.
  - Jumbo VLAN tagged frames—the payload length is between 42 (0x2A) and the value specified in the MAX_RX_SIZE_CONFIG register minus 22 bytes.
  - Jumbo stacked VLAN tagged frames—the payload length is between 38 (0x26) and the value specified in the MAX_RX_SIZE_CONFIG register minus 26 bytes.

The RX MAC does not drop frames with invalid length or invalid payload length. If the frame or payload length is not valid, the MAC function asserts output error bits.

- `l1_rx_error[2]`—Undersized frame.
- `l1_rx_error[3]`—Oversized frame.
- `l1_rx_error[4]`—Payload length error.

If the length field value is greater than the actual payload length, the IP core asserts `l1_rx_error[4]`. If the length field value is less than the actual payload length, the MAC RX considers the frame to have excessive padding and does not assert `l1_rx_error[4]`.

Note: Starting from Intel Quartus Prime Pro Edition software version 20.3 onwards, the MAC has a counter limit of 0xFFFF. For frames with size larger than this value, the MAC asserts oversized frame error regardless of the value programmed into MAX_RX_SIZE_CONFIG. This applies to the TX datapath as well.

4.1.3.4. RX CRC Checking and Dynamic Forwarding

The RX MAC checks the incoming CRC32 for errors. It asserts `l1_rx_error[1]` in the same cycle as `l1_rx_endofpacket` when it detects an error. CRC checking takes several cycles. The packet frame is delayed to align the CRC output with the end of the frame.

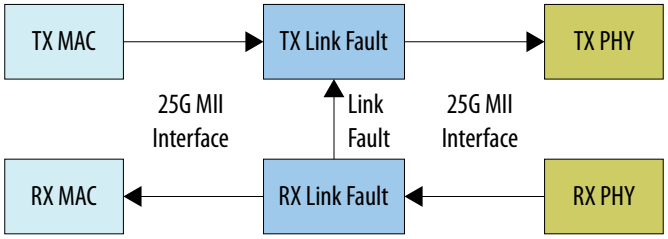
By default, the RX MAC strips off the CRC bytes before forwarding the packet to the MAC client. You can configure the core to retain the RX CRC and forward it to the client by updating the `MAC_CRC_CONFIG` register.

4.1.4. Link Fault Signaling Interface

Link fault signaling reflects the health of the link. It operates between the remote Ethernet device Reconciliation Sublayer (RS) and the local Ethernet device RS. The link fault modules communicate status during the interframe period.

You enable link fault signaling by turning on **Enable link fault generation** in the parameter editor. For bidirectional fault signaling, the IP core implements the functionality defined in the *IEEE 802.3ba 10G Ethernet Standard* and *Clause 46* based on the `LINK_FAULT` configuration register settings.
For unidirectional fault signaling, the core implements *Clause 66 of the IEEE 802.3-2012 Ethernet Standard*.

Figure 15. Link Fault Block Diagram



Local Fault (LF)

If an Ethernet PHY sublayer detects a fault that makes the link unreliable, it notifies the RS of the local fault condition. If unidirectional is not enabled, the core follows *Clause 46*. The RS stops sending MAC data, and continuously generates a remote fault status on the TX datapath. After a local fault is detected, the RX PCS modifies the MII data and control to send local fault sequence ordered sets. Refer to *Link Fault Signaling Based On Configuration and Status* below.

The RX PCS cannot recognize the link fault under the following conditions:

- The RX PCS is not fully aligned.
- The bit error rate (BER) is high.

Remote Fault (RF)

If unidirectional is not enabled, the core follows *Clause 46*. If the RS receives a remote fault status, the TX datapath stops sending MAC data and continuously generates idle control characters. If the RS stops receiving fault status messages, it returns to normal operation, sending MAC client data. Refer to *Link Fault Signaling Based On Configuration and Status* below.

Link Status Signals

The MAC RX generates two link fault signals: `local_fault_status` and `remote_fault_status`.

Note: These signals are real time status signals that reflect the status of the link regardless of the settings in the link fault configuration register.

This register is generated only if you turn on **Enable link fault generation**. The MAC TX interface uses the link fault status signals for additional link fault signaling.

Table 11. Link Fault Signaling Based On Configuration and Status

For more information about the LINK_FAULT register, refer to TX MAC Registers.

LINK_FAULT Register (0x405)				Real Time Link Status		Configured TX Behavior		Comment
Bit [0]	Bit [3]	Bit [1]	Bit [2]	LF Received	RF Received	TX Data	TX RF	
1'b0	Don't care	Don't care	Don't care	Don’t care	Don’t care	On	Off	Disable Link fault signaling on TX. RX still reports link status. TX side Link fault signaling disabled on the link. TX data and idle.
1'b1	1'b1	Don't care	Don't care	Don't care	Don't care	Off	On	Force RF. TX: Stop data. Transmit RF only
1'b1	1'b0	1'b1	1'b1	Don't care	Don't care	On	Off	Unidir: Backwards compatible. TX: Transmit data and idle. No RF.
1'b1	1'b0	1'b1	1'b0	1'b1	1'b0	On	On	Unidir: LF received. TX: Transmit data 1 column IDLE after end of packet and RF
1'b1	1'b0	1'b1	1'b0	1'b0	1'b1	On	Off	Unidir: RF receives TX: Transmit data and idle. No RF.
1'b1	1'b0	1'b1	1'b0	1'b0	1'b0	On	Off	Unidir: No link fault TX: Transmit data and idle. No RF.
1'b1	1'b0	1'b0	Don't care	1'b1	1'b0	Off	On	Bidir: LF received TX: Stop data. Transmit RF only.
1'b1	1'b0	1'b0	Don't care	1'b0	1'b1	Off	Off	Bidir: RF received TX: Stop data. Idle only. No RF.
1'b1	1'b0	1'b0	Don’t care	1'b0	1'b0	On	Off	Bidir: No link fault TX: Transmit data and idle. No RF.

At this time, the 25G Ethernet Intel FPGA IP core does not recognize received non-zero 4-bit ordered set types as an error.

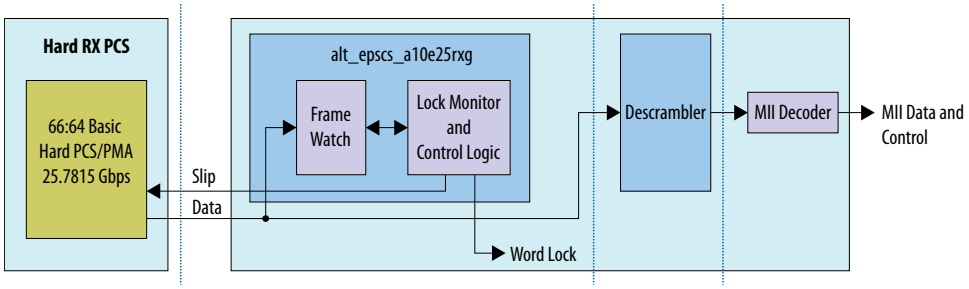
Related Information

- TX MAC Registers on page 70
  Information about the LINK_FAULT register.
- IEEE website
  The Ethernet specifications are available on the IEEE website.

4.1.5. 25 GbE RX PCS

The soft RX PCS interfaces to the hard PCS and PMA blocks configured in 66:64 10G PCS Basic Generic Mode with bitslip enabled. The hard PCS drives a 66-bit output stream to the soft RX PCS. The soft RX PCS implements word lock, descrambling, and MII decoding. It drives output data to the MAC. You can read the status of FIFOs at the interface of Hard RX PCS using the Control and Status registers.

Figure 16. High Level Block Diagram of the Soft RX PCS



4.1.5.1. RX RSFEC

If you turn on **Enable RS-FEC** in the 25G Ethernet Intel FPGA IP parameter editor, the IP core includes Reed-Solomon forward error correction (FEC) in both the receive and transmit datapaths.

The IP core implements Reed-Solomon FEC per Clause 108 of the IEEE Standard 802.3by. The Reed-Solomon FEC algorithm includes the following modules:

- Alignment marker lock
- 66:80 gearbox
- High-speed Reed-Solomon decoder
- 80:257 gearbox
- 256B/257B to 64B/66B Transcoding

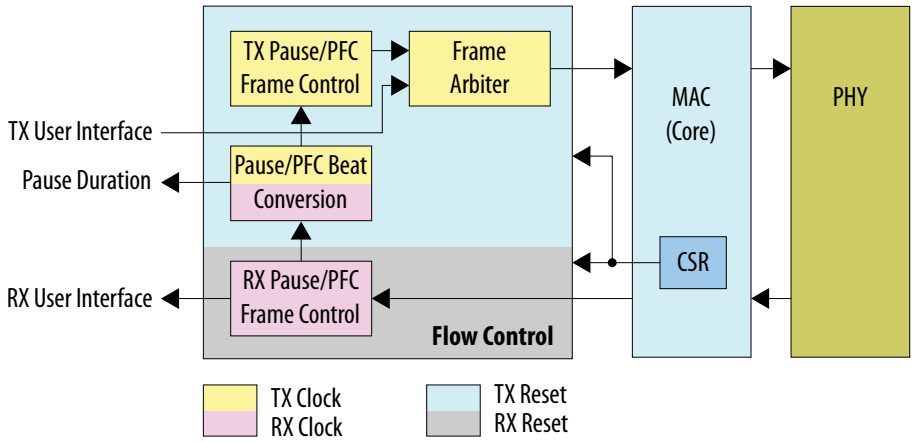
You cannot turn on both **Enable RS-FEC** and **Enable IEEE 1588** in the 25G Ethernet Intel FPGA IP parameter editor.

4.1.6. Flow Control

Flow control reduces congestion at the local or remote link partner. When either link partner experiences congestion, the respective transmit control sends pause frames. XOFF Pause frames stop the remote transmitter. XON Pause frames let the remote transmitter resume data transmission. The 25G Ethernet Intel FPGA IP core supports both standard and Priority-based Flow Control (PFC) control frames.

Figure 17. Flow Control Module Conceptual Overview

The flow control module acts as a buffer between client logic and the TX and RX MAC.



Standard Flow Control (Pause Frame Flow Control):

- Inhibits the next client frame transmission on the reception of a valid Pause frame.

Priority-based Flow Control (PFC):

- PFC frame transmission follows a priority-based arbitration scheme, where the Frame Type indication is provided for the usage of external downstream logic.
- Inhibits the per queue client frame transmission on the reception of a valid PFC frame from the client. Includes per-queue PFC Pause quanta duration indicator

Flow Control includes the following features:

Feature	Standard Flow Control	Priority-based Flow Control (PFC)
Generation and Transmission		
Programmable 1-bit or 2-bit XON/XOFF request mode	Supported	Supported
In 2-bit request mode, programmable selection of register or signal-based control	Supported	Supported
Programmable destination and source addresses	Supported	Supported
Programmable pause quanta	Supported	Supported
Programmable per-queue XOFF frame separation	—	Supported
Reception and Decode		
Programmable destination address for filtering incoming pause and PFC frames	Supported	Supported
Configurable enable, directing the IP core to ignore incoming flow control frames	Supported	Supported
Per-queue client frame transmission pause duration indicator	—	Supported

Caution: The 25G Ethernet Intel FPGA IP core supports the flow control feature for either value of the **Ready Latency** parameter. However, in standard flow control you might experience data delay if you select the value of 3 for this parameter. The IP core might still hold user data packet in its internal buffer if transmission of the IP core stops due to flow control. This issue does not occur in priority-based flow control.

Related Information

Pause/PFC Flow Control Registers on page 72
Describes the registers that the IP core uses to implement the flow control functionality.

4.1.6.1. TX Pause/PFC Flow Control Frame Transmission Request

An XON/XOFF request triggers the IP core to transmit a Pause or PFC flow control frame on the Ethernet link. You can control XON/XOFF requests using the TX flow control registers or the `pause_insert_tx0` and `pause_insert_tx1` input signals.

You can specify whether the IP core accepts XON/XOFF requests in 1-bit or 2-bit format by updating the `TX Flow Control Request Mode` register field. By default the IP core assumes 1-bit requests.

4.1.6.2. XON/XOFF Pause Frames

Priority-based Flow Control

You can trigger the 25G Ethernet Intel FPGA IP core to transmit PFC XOFF frame with a pause duration that is specified in TX Flow Control Quanta register by updating the `pause_insert_tx0` and `pause_insert_tx1` input signals or TX flow control registers. If an enabled priority queue is in the XOFF condition, a new PFC frame is transmitted after the minimum time gap. You specify the minimum time gap in the per priority queue `TX Flow Control Signal XOFF Request Hold Quanta` register. The minimum time gap between two consecutive PFC frames is 1 pause quanta or 512-bit times. PFC frame transmission ends when none of the PFC interfaces of all enabled priority queues is requesting PFC frames.

A transition from XOFF to XON in any enabled priority queue triggers the IP core to transmit a PFC frame with pause quanta of 0 for the associated priority queue. The IP core sends a single XON flow control frame. In the rare case that the XON frame is lost or corrupted, the remote partner should still be able to resume transmission. The remote partner resumes transmission after the duration or the pause quanta value specified in the previous XOFF flow control frame expires.

Standard Flow Control

In the case of standard flow control, the IP core transmits Pause frames instead of PFC frames. The transmission behavior is identical.

When the IP core is in standard flow control mode and receives a Pause frame, the IP core stops processing TX client data, either immediately or at the next frame boundary. Client data transmission resumes when all of the following conditions are true:

- The time specified by the pause quanta has elapsed and there is no new quanta value.
- A valid pause frame with 0 pause duration has been received.

A Pause frame has no effect if the associated `TX Flow Control Enable` register bit is set to disable XON and XOFF flow control.

4.1.7. 1588 Precision Time Protocol Interfaces

If you turn on **Enable IEEE 1588**, the 25G Ethernet Intel FPGA IP core processes and provides 1588 Precision Time Protocol (PTP) timestamp information as defined in the *IEEE 1588-2008 Precision Clock Synchronization Protocol for Networked Measurement and Control Systems Standard*. This feature supports PHY operating speed with a constant timestamp accuracy of ± 3 ns and a dynamic timestamp accuracy of ± 1 ns.

1588 PTP packets carry timestamp information. The 25G Ethernet Intel FPGA IP core updates the incoming timestamp information in a 1588 PTP packet to transmit a correct updated timestamp with the data it transmits on the Ethernet link, using a one-step or two-step clock.

A fingerprint can accompany a 1588 PTP packet. You can use this information for client identification and other client uses. If provided 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. Because the module is outside the 25G Ethernet Intel FPGA IP core, you can use the same module to provide the current time of day for multiple modules in your system.

Related Information

- 1588 PTP Registers on page 84
- IEEE website
  The *IEEE 1588-2008 Precision Clock Synchronization Protocol for Networked Measurement and Control Systems Standard* is available on the IEEE website.

4.1.7.1. Implementing a 1588 System That Includes a 25G Ethernet Intel FPGA IP Core

The 1588 specification in *IEEE 1588-2008 Precision Clock Synchronization Protocol for Networked Measurement and Control Systems Standard* describes various systems you can implement in hardware and software to synchronize clocks in a distributed system by communicating offset and frequency correction information between master and slave clocks in arbitrarily complex systems. A 1588 system that includes the 25G Ethernet Intel FPGA IP core with 1588 PTP functionality uses the incoming and outgoing timestamp information from the IP core and the other modules in the system to synchronize clocks across the system.

The 25G Ethernet Intel FPGA IP core with 1588 PTP functionality provides the timestamp manipulation and basic update capabilities required to integrate your IP core in a 1588 system. You can specify that packets are PTP packets, and how the IP core should update incoming timestamps from the client interface before transmitting them on the Ethernet link. The IP core does not implement the event messaging layers of the protocol, but rather provides the basic hardware capabilities that support a system in implementing the full 1588 protocol.

Figure 18. Example Ethernet System with Ordinary Clock Master and Ordinary Clock Slave

You can implement both master and slave clocks using the 25G Ethernet Intel FPGA IP core with 1588 PTP functionality. Refer to Adding the External Time-of-Day Module for Variations with 1588 PTP Feature for implementation of the TOD module.

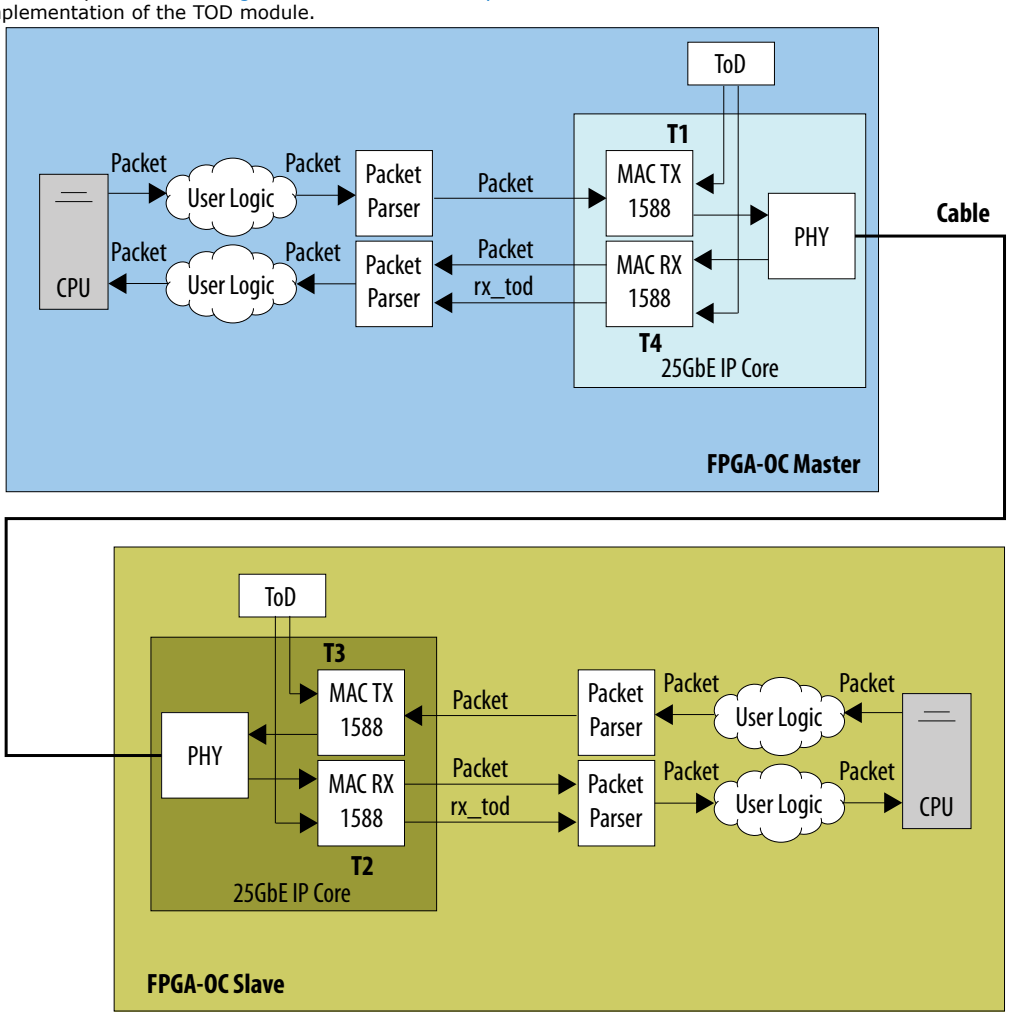


Figure 19. Hardware Configuration Example Using 25G Ethernet Intel FPGA IP core in a 1588 System in Transparent Clock Mode

Refer to Adding the External Time-of-Day Module for Variations with 1588 PTP Feature for implementation of the TOD module.

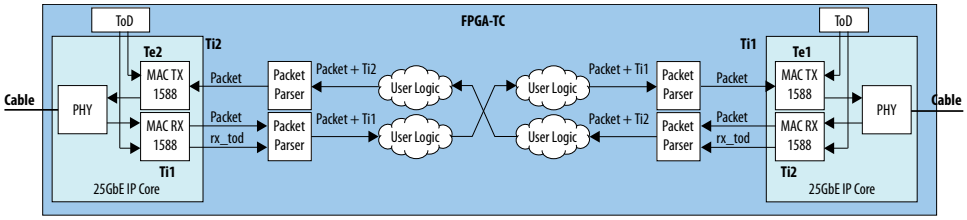


Figure 20. Software Flow Using Transparent Clock Mode System

This figure from the 1588 standard is augmented with the timestamp labels shown in the transparent clock system figure. A precise description of the software requirements is beyond the scope of this document. Refer to the 1588 standard.

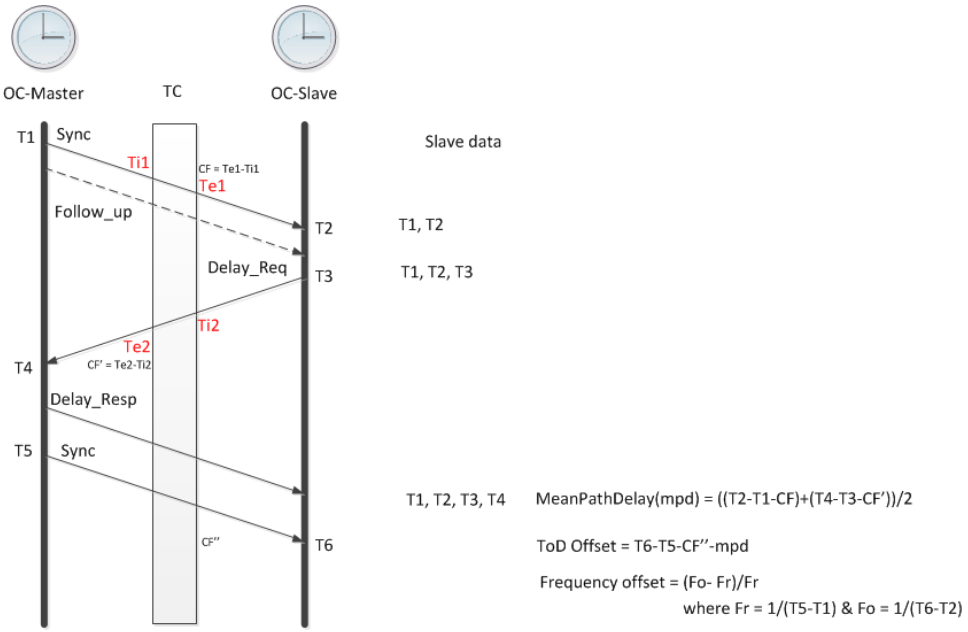
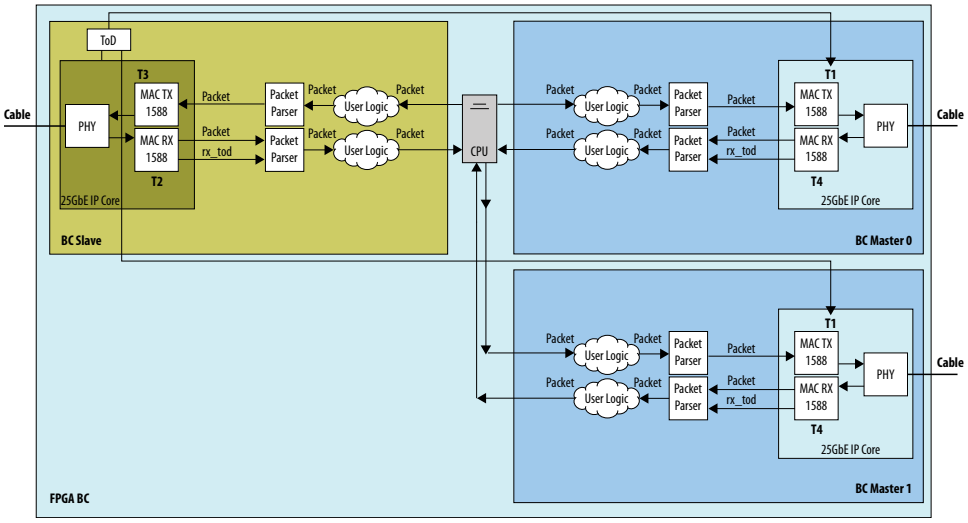


Figure 21. Example Boundary Clock with One Slave Port and Two Master Ports

You can implement a 1588 system in boundary clock mode using the 25G Ethernet Intel FPGA IP core with 1588 PTP functionality. Refer to Adding the External Time-of-Day Module for Variations with 1588 PTP Feature for implementation of the TOD module.



Related Information

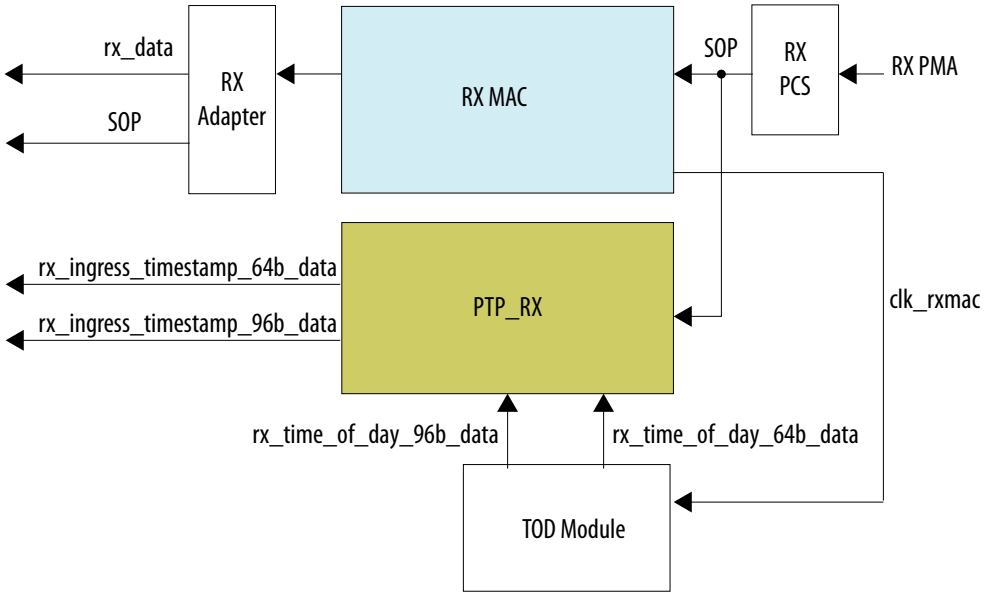
IEEE website

The *IEEE 1588-2008 Precision Clock Synchronization Protocol for Networked Measurement and Control Systems Standard* is available on the IEEE website.

4.1.7.2. PTP Receive Functionality

If you turn on **Enable IEEE 1588** in the 25G Ethernet Intel FPGA IP parameter editor, the IP core provides a 96-bit (V2 format) or 64-bit timestamp with every packet on the RX client interface, whether it is a 1588 PTP packet or not. The value on the timestamp bus (`rx_ingress_timestamp_96b_data[95:0]` or `rx_ingress_timestamp_64b_data[63:0]` or both, if present) is valid in the same clock cycle as the RX SOP signal. The value on the timestamp bus is not the current timestamp; instead, it is the timestamp from the time when the IP core received the packet on the Ethernet link. The IP core captures the time-of-day from the TOD module on `rx_time_of_data_96b_data` or `rx_time_of_day_64b_data` at the time it receives the packet on the Ethernet link, and sends that timestamp to the client on the RX SOP cycle on the timestamp bus `rx_ingress_timestamp_96b_data[95:0]` or `rx_ingress_timestamp_64b_data[63:0]` or both, if present. User logic can use this timestamp or ignore it.

Figure 22. PTP Receive Block Diagram



Related Information

IEEE website

The *IEEE 1588-2008 Precision Clock Synchronization Protocol for Networked Measurement and Control Systems Standard* is available on the IEEE website.

4.1.7.3. PTP Transmit Functionality

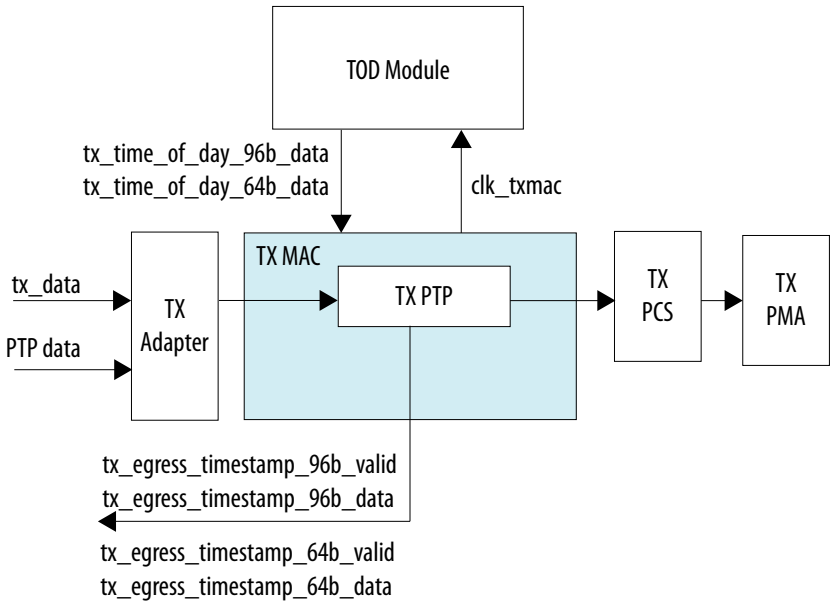
When you send a 1588 PTP packet to a 25G Ethernet Intel FPGA IP core with **Enable IEEE 1588** turned on in the parameter editor, you should assert the following respective input signals with the TX SOP signal to tell the IP core the PTP operations or processes that the IP core should perform to the packet:

- `tx_egress_timestamp_request_valid`: assert this signal to tell the IP core to process the current packet in two-step processing mode.
- `tx_etstamp_ins_ctrl_timestamp_insert`: assert this signal to tell the IP core to process the current packet in one-step processing mode and to insert the exit timestamp for the packet in the packet (insertion mode).
- `tx_etstamp_ins_ctrl_residence_time_update`: assert this signal to tell the IP core to process the current packet in one-step processing mode and to update the timestamp in the packet by adding the latency through the IP core (the residence time in the IP core) to the cumulative delay field maintained in the packet (correction mode). This mode supports transparent clock systems.

Note: If `tx_etstamp_ins_ctrl_residence_time_update` is asserted, you should not assert `tx_egress_timestamp_request_valid` or `tx_etstamp_ins_ctrl_timestamp_insert` as the result will be undefined.

The IP core transmits the 1588 PTP packet in an Ethernet frame after PTP processing.

Figure 23. PTP Transmit Block Diagram



In one-step mode, the IP core either overwrites the timestamp information provided at the user-specified offset with the packet exit timestamp (insertion mode), or adds the residence time in this system to the value at the specified offset (correction mode). You tell the IP core how to process the timestamp by asserting the appropriate signal with the TX SOP signal. You must specify the offset of the timestamp in the packet (`tx_etstamp_ins_ctrl_offset_timestamp`) in insertion mode, or the offset of the correction field in the packet

(`tx_etstamp_ins_ctrl_offset_correction_field`) in correction mode. In addition, the IP core zeroes out or updates the UDP checksum, or leaves the UDP checksum as is, depending on the mutually exclusive `tx_etstamp_ins_ctrl_checksum_zero` and `tx_etstamp_ins_ctrl_checksum_correct` signals. Checksum calculation is mandatory for the UDP/IPv6 protocol. You must extend 2 bytes at the end of the UDP payload of the PTP packet. The MAC function modifies the extended bytes to ensure that the UDP checksum remains uncompromised.

Two-step PTP processing ignores the values on the one-step processing signals. In two-step processing mode, the IP core does not modify the current timestamp in the packet. Instead, the IP core transmits a two-step derived timestamp on the separate `tx_egress_timestamp_96b_data[95:0]` or `tx_egress_timestamp_64b_data[63:0]` bus, when it begins transmitting the Ethernet frame. The value on the `tx_egress_timestamp_{96b,64b}_data` bus is the packet exit timestamp. The `tx_egress_timestamp_{96b,64b}_data` bus holds a valid value when the corresponding `tx_egress_timestamp_{96b,64b}_valid` signal is asserted.

In addition, to help the client to identify the packet, you can specify a fingerprint to be passed by the IP core in the same clock cycle with the timestamp. To specify the number of distinct fingerprint values the IP core can handle, set the **Fingerprint width** parameter to the desired number of bits `W`. You provide the fingerprint value to the IP core in the `tx_egress_timestamp_request_fingerprint[(W-1):0]` signal. The IP core then drives the fingerprint on the appropriate `tx_egress_timestamp_{96b,64b}_fingerprint[(W-1):0]` port with the corresponding output timestamp, when it asserts the `tx_egress_timestamp_{96b,64b}_valid` signal.

The IP core calculates the packet exit timestamp.

exit TOD = entry TOD + IP core maintained expected latency + user-configured PMA latency

- entry TOD is the value in `tx_time_of_day_96b_data` or `tx_time_of_day_64b_data` when the packet enters the IP core.
- The expected latency through the IP core is a static value. The IP core maintains this value internally.
- The IP core reads the user-configured PMA latency from the TX_PTP_PMA_LATENCY register. This option is provided for user flexibility.

The IP core provides the exit TOD differently in different processing modes.

- In two-step mode, the IP core drives the exit TOD on `tx_egress_timestamp_96b_data` and on `tx_egress_timestamp_64b_data`, as available.
- In one-step processing insertion mode, the IP core inserts the exit TOD in the timestamp field of the packet at the offset you specify in `tx_etstamp_ins_ctrl_offset_timestamp`.
- In one-step processing correction mode, the IP core calculates the exit TOD and uses it only to calculate the residence time.

In one-step processing correction mode, the IP core calculates the updated correction field value:

exit correction field value = entry correction field value + residence time + asymmetry extra latency

- residence time = exit TOD – entry (ingress) timestamp.
- entry (ingress) timestamp is the value on `tx_etstamp_ins_ctrl_ingress_timestamp_{95,64}b` in the SOP cycle when the IP core received the packet on the TX client interface. The application is responsible to drive this signal with the correct value for the cumulative calculation. The correct value depends on system configuration.
- The IP core reads the asymmetry extra latency from the TX_PTP_ASYM_DELAY register if the `tx_egress_asymmetry_update` signal is asserted. This option is provided for additional user-defined precision. You can set the value of this register and set the `tx_egress_asymmetry_update` signal to indicate the register value should be included in the latency calculation.

Related Information

- 1588 PTP Registers on page 84
- IEEE website
  The *IEEE 1588-2008 Precision Clock Synchronization Protocol for Networked Measurement and Control Systems Standard* is available on the IEEE website.

4.1.7.4. External Time-of-Day Module for 1588 PTP Variations

25G Ethernet Intel FPGA IP cores that include the 1588 PTP module require an external time-of-day (TOD) module to provide the current time-of-day in each clock cycle, based on the incoming clock. The TOD module must update the time-of-day output value on every clock cycle, and must provide the TOD value in the V2 format (96 bits) or the 64-bit TOD format, or both.

Related Information

Adding the External Time-of-Day Module for Variations with 1588 PTP Feature on page 22

4.1.7.5. PTP Timestamp and TOD Formats

The 25G Ethernet Intel FPGA IP core supports a 96-bit timestamp (V2 format) or a 64-bit timestamp (correction-field format) in PTP packets. The 64-bit timestamp and TOD signals of the IP core are in an Intel-defined 64-bit format that is distinct from the V1 format, for improved efficiency in one-step processing correction mode. Therefore, if your system need not handle any packets in one-step processing correction mode, you should set the **Time of day format** parameter to the value of **Enable 96-bit timestamp format**.

You control the format or formats the IP core supports with the **Time of day format** parameter. If you set the value of this parameter to **Enable 96-bit timestamp format** or **Enable both formats**, your IP core can support two-step processing mode, one-step processing insertion mode, and one-step processing correction mode, and can support both V1 and V2 formats. You can set the parameter value to **Enable 64-bit timestamp format** to support one-step processing correction mode more efficiently. However, if you do so, your IP core variation cannot support two-step processing mode and cannot support one-step processing insertion mode. If you turn

on both of these parameters, the value you drive on the `tx_estamp_ins_ctrl_timestamp_format` or `tx_etstamp_ins_ctrl_residence_time_calc_format` signal determines the format the IP core supports for the current packet.

The IP core completes all internal processing in the V2 format. However, if you specify V1 format for a particular PTP packet in one-step insertion mode, the IP core inserts the appropriate V1-format timestamp in the outgoing packet on the Ethernet link.

V2 Format

The IP core maintains the time-of-day (TOD) in V2 format according to the IEEE specification:

- Bits [95:48]: Seconds (48 bits).
- Bits [47:16]: Nanoseconds (32 bits). This field overflows at 1 billion.
- Bits [15:0]: Fractions of nanosecond (16 bits). This field is a true fraction; it overflows at 0xFFFF.

The IP core can receive time-of-day information from the TOD module in V2 format or in 64-bit TOD format, or both, depending on your setting for the **Time of day format** parameter.

V1 Format

V1 timestamp format is specified in the IEEE specification:

- Bits [63:32]: Seconds (32 bits).
- Bits [31:0]: Nanoseconds (32 bits). This field overflows at 1 billion.

Intel 64-Bit TOD Format

The Intel 64-bit TOD format is distinct from the V1 format and supports a longer time delay. It is intended for use in transparent clock systems, in which each node adds its own residence time to a running total latency through the system. This format matches the format of the correction field in the packet, as used in transparent clock mode.

- Bits [63:16]: Nanoseconds (48 bits). This field can specify a value greater than 4 seconds.
- Bits [15:0]: Fractions of nanosecond (16 bits). This field is a true fraction; it overflows at 0xFFFF.

The TOD module provides 64-bit TOD information to the IP core in this 64-bit TOD format. The expected format of all 64-bit input timestamp and TOD signals to the IP core is the Intel 64-bit TOD format. The format of all 64-bit output timestamp and TOD signals from the IP core is the Intel 64-bit TOD format. If you build your own TOD module that provides 64-bit TOD information to the IP core, you must ensure it provides TOD information in the Intel 64-bit TOD format.

Related Information

IEEE website
The *IEEE 1588-2008 Precision Clock Synchronization Protocol for Networked Measurement and Control Systems Standard* is available on the IEEE website.

4.2. User Interface to Ethernet Transmission

The IP core reverses the bit stream for transmission per Ethernet requirements. The transmitter handles the insertion of the inter-packet gap, frame delimiters, and padding with zeros as necessary. The transmitter also handles FCS computation and insertion.

The IP core transmits complete packets. After transmission begins, it must complete with no IDLE insertions. Between the end of one packet and the beginning of the next packet, the data input is not considered and the transmitter sends IDLE characters. An unbounded number of IDLE characters can be sent between packets.

4.2.1. Order of Transmission

The IP core 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.

Figure 24. Byte Order on the Client Interface Lanes

Describes the byte order on the Avalon streaming interface. Destination Address[40] is the broadcast/multicast bit (a type bit), and Destination Address[41] is a locally administered address bit.

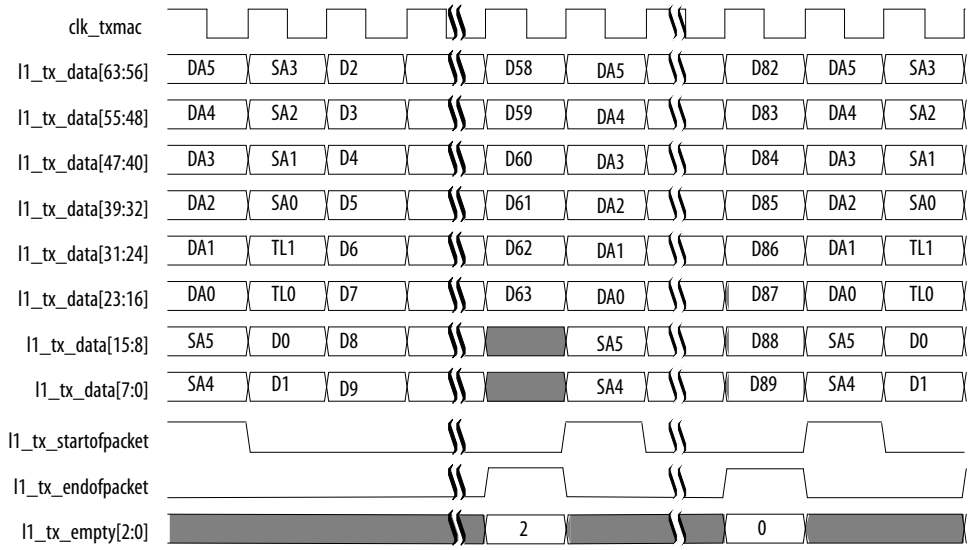
	Destination Address (DA)						Source Address (SA)						Type/ Length (TL)		Data (D)		
Octet	5	4	3	2	1	0	5	4	3	2	1	0	1	0	00	...	NN
Bit	[47:40]	[39:32]	[31:24]	[23:16]	[15:8]	[7:0]	[47:40]	[39:32]	[31:24]	[23:16]	[15:8]	[7:0]	[15:8]	[7:0]	MSB[7 :0]	...	LSB[7:0]

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 5 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 25. Octet Transmission on the 25GbE Avalon Streaming Signals

In the following diagram Preamble pass through and CRC pass through mode are disabled.



4.2.2. Bit Order For TX and RX Datapaths

The TX bit order matches the placement shown in the PCS lanes as illustrated in *IEEE Standard for Ethernet, Section 4, Figure 49-5*. The RX bit order matches the placement shown in *IEEE Standard for Ethernet, Section 4, Figure 49-6*.

Related Information

IEEE website

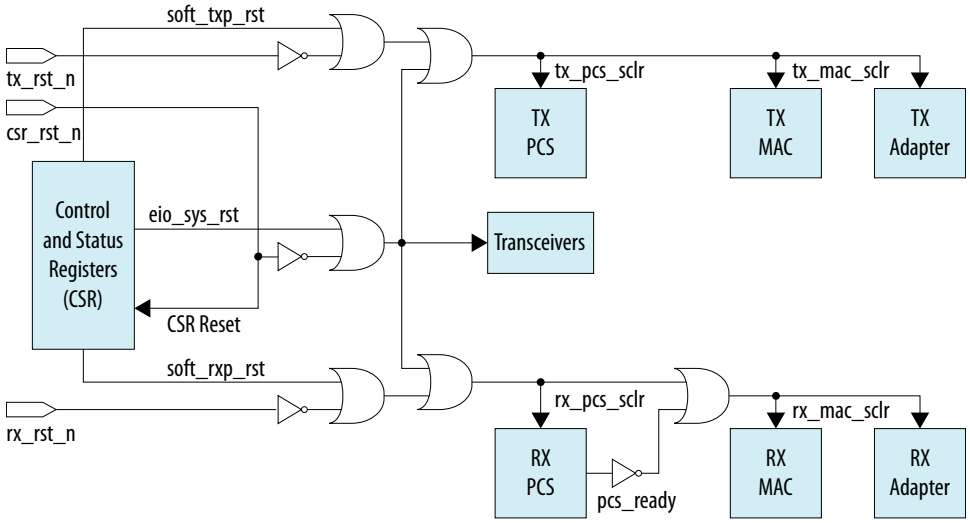
The *IEEE Standard for Ethernet, Section 4* is available on the IEEE website.

5. Reset

Control and Status registers control three parallel soft resets. These soft resets are not self-clearing. Software clears them by writing to the appropriate register. Asserting the external hard reset `csr_rst_n` returns Control and Status registers to their original values.

Figure 26. Conceptual Overview of Reset Logic

The three hard resets are top-level ports. The soft resets are internal signals which are outputs of the `PHY_CONFIG` register. Software writes the appropriate bit of the `PHY_CONFIG` to assert a soft reset.



The internal soft reset signals reset the following functions:

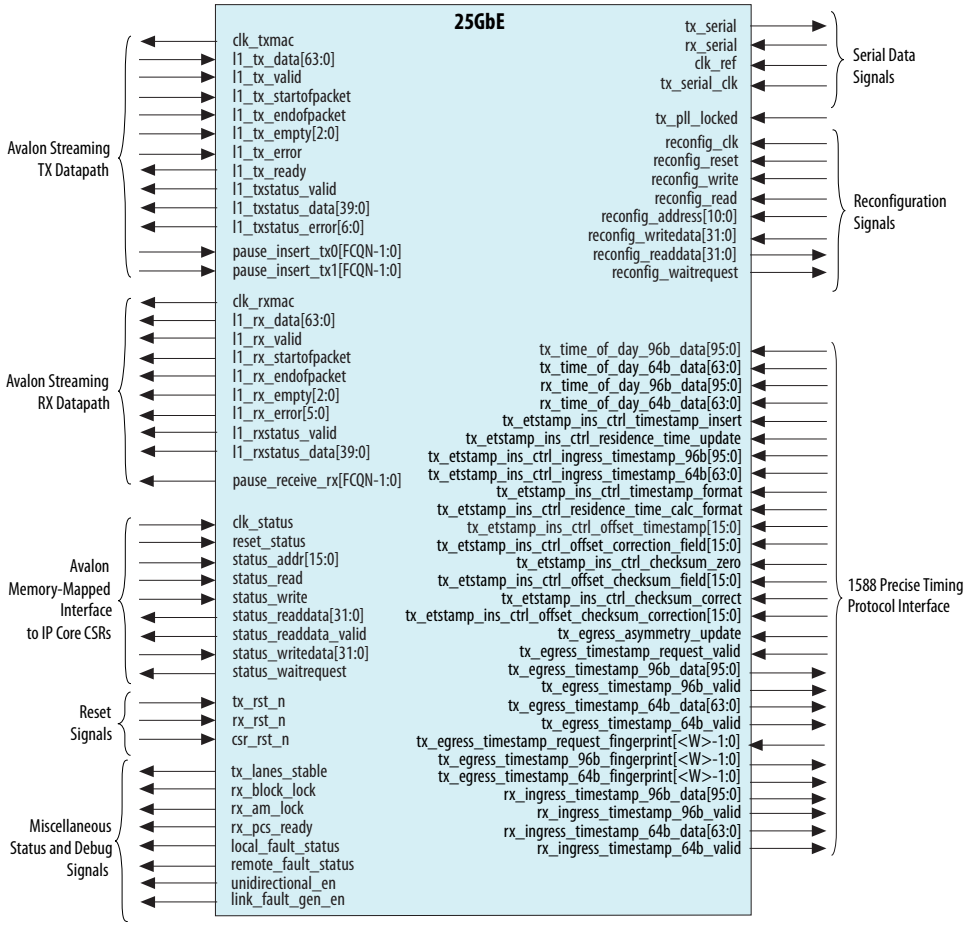
- `soft_txp_rst`: Resets the IP core in TX direction. Resets the TX PCS, MAC, and adapter.This soft reset leads to deassertion of `tx_lanes_stable` output signal.
- `soft_rxp_rst`: Resets the IP core in RX direction. Resets the RX PCS, MAC, and adapter. This soft reset leads to the deassertion of `rx_pcs_ready` output signal.
- `eio_sys_rst`: Resets the IP core. Resets the TX and RX MACs, PCS, adapters, and transceivers. Does not reset the Control and Status registers. This soft reset leads to the deassertion of `tx_lanes_stable` and `rx_pcs_ready` output signal.

Related Information

- Reset Signals on page 66
- PHY Registers on page 68

6. Interfaces and Signal Descriptions

Figure 27. 25G Ethernet Intel FPGA IP Signals and Interfaces



6.1. TX MAC Interface to User Logic

The TX MAC provides an Avalon streaming interface to the FPGA fabric. The minimum packet size is nine bytes.

Table 12. Avalon Streaming TX Datapath

All interface signals are clocked by the clk_txmac clock. The value you specify for **Ready Latency** in the 25G Ethernet Intel FPGA IP parameter editor is the Avalon streaming readyLatency value on this interface.

Signal	Direction	Description
clk_txmac	Output	Clock for the TX logic. Derived from pll_refclk, and is an output from the 25G Ethernet Intel FPGA IP core. clk_txmac is guaranteed to be stable when tx_lanes_stable is asserted. The frequency of this clock is 390.625 MHz. **All TX MAC interface signals are synchronous to clk_txmac**.
l1_tx_data[63:0]	Input	Data input to MAC. Bit 63 is the MSB and bit 0 is the LSB. Bytes are read in the usual left to right order. The 25G Ethernet Intel FPGA IP core does not process incoming frames of less than nine bytes correctly. You must ensure such frames do not reach the TX client interface. You must send each TX data packet without intermediate idle cycles. Therefore, you must ensure your application can provide the data for a single packet in consecutive clock cycles. If data might not be available otherwise, you must buffer the data in your design and wait to assert l1_tx_startofpacket when you are assured the packet data to send on l1_tx_data[63:0] is available or will be available on time. If readyLatency = 0, ensure that no data transition at the l1_tx_data bus at the same clock cycle l1_tx_ready is deasserted. You can transition the data at the l1_tx_data bus at the same clock cycle l1_tx_ready is asserted. If readyLatency = 3, ensure that no data transition at the l1_tx_data bus at the third clock cycle after l1_tx_ready is deasserted. You can transition the data at the l1_tx_data bus at the third clock cycles after l1_tx_ready is asserted.
l1_tx_valid	Input	When asserted, indicates valid data is available on l1_tx_data[63:0]. You must assert this signal continuously between the assertions of l1_tx_startofpacket and l1_tx_endofpacket for the same packet regardless of the l1_tx_ready status.
l1_tx_startofpacket	Input	When asserted, indicates the first byte of a frame. When l1_tx_startofpacket is asserted, the MSB of l1_tx_data drives the start of packet. Packets that drive l1_tx_startofpacket and l1_tx_endofpacket in the same cycle are ignored.
l1_tx_endofpacket	Input	When asserted, indicates the end of a packet. Packets that drive l1_tx_startofpacket and l1_tx_endofpacket in the same cycle are ignored.
l1_tx_empty[2:0]	Input	Specifies the number of empty bytes on l1_tx_data when l1_tx_endofpacket is asserted.
l1_tx_error	Input	When asserted in the same cycle as l1_tx_endofpacket, indicates the current packet should be treated as an error packet. Assertion at any other position in the packet is ignored. The TX statistics counters do not reflect errors the IP core creates in response to this signal.
l1_tx_ready	Output	When asserted, indicates that the MAC can accept the data. When the readyLatency = 0, the IP core accepts valid data in the same clock cycle in which it asserts l1_tx_ready. When the readyLatency = 3, the IP core accepts valid data 3 clock cycles after it asserts l1_tx_ready.
l1_txstatus_valid	Output	When asserted, indicates that l1_txstatus_data[39:0] is driving valid data.

continued...

Signal	Direction	Description
`l1_txstatus_data[39:0]`	Output	Specifies information about the transmit frame. The following fields are defined: • Bit[39]: When asserted, indicates a PFC frame • Bit[38]: When asserted, indicates a unicast frame • Bit[37]: When asserted, indicates a multicast frame • Bit[36]: When asserted, indicates a broadcast frame • Bit[35]: When asserted, indicates a pause frame • Bit[34]: When asserted, indicates a control frame • Bit[33]: When asserted, indicates a VLAN frame • Bit[32]: When asserted, indicates a stacked VLAN frame • Bits[31:16]: Specifies the frame length from the first byte of the destination address to the last bye of the FCS • Bits[15:0]: Specifies the payload length
`l1_txstatus_error[6:0]`	Output	Specifies the error type in the transmit frame. The following fields are defined: • Bits[6:3]: Reserved • Bit[2]: Payload length error • Bit[1]: Oversized frame • Bit[0]: Reserved
`pause_insert_tx0[FCQN-1:0]` `pause_insert_tx1[FCQN-1:0]`	Input	Available if you specify Pause or PFC. Indicates to the MAC if an XON, XOFF, Pause or PFC frame should be sent. FCQN equals 1 for Pause and 1-8 for PFC. In 1-bit programming mode, the IP core ignores `pause_insert_tx1[FCQN-1:0]`. In 2-bit programming mode, the higher-order bit is in `pause_insert_tx1[FCQN-1:0]` and the lower-order bit is in `pause_insert_tx0[FCQN-1:0]`. The following encodings are defined for 1-bit programming mode: • 0 = No request • 0 to 1 = Generate XOFF request • 1 = Continue to generate XOFF request • 1 to 0 = Generate XON request The following encodings are defined for the 2-bit programming model: • 2'b00: No further XON/XOFF request. If there is a XON/XOFF flow control frame in progress, it is sent • 2'b01: Generate XON flow control frame • 2'b10: Generate XOFF request • 2'b11: Invalid

Figure 28. Client to 25G Ethernet Intel FPGA IP MAC Avalon Streaming Interface when Ready Latency is 0 (1 of 2)

This timing diagram shows `l1_tx_ready` asserts before `l1_tx_valid` asserts. TX MAC captures data at `l1_tx_data` bus at the same clock cycle as `l1_tx_valid` asserts.

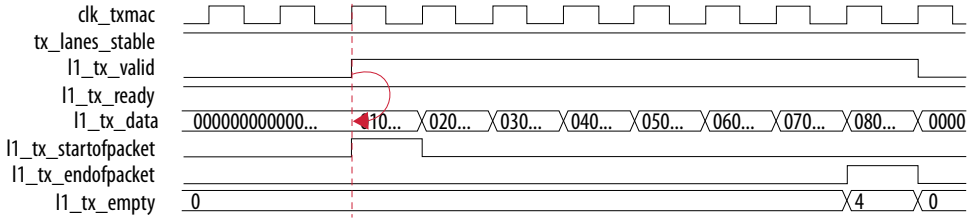


Figure 29. Client to 25G Ethernet Intel FPGA IP MAC Avalon Streaming Interface when Ready Latency is 0 (2 of 2)

This timing diagram shows `l1_tx_valid` asserts before `l1_tx_ready` asserts. TX MAC captures data at `l1_tx_data` bus at the same clock cycle as `l1_tx_ready asserts`.

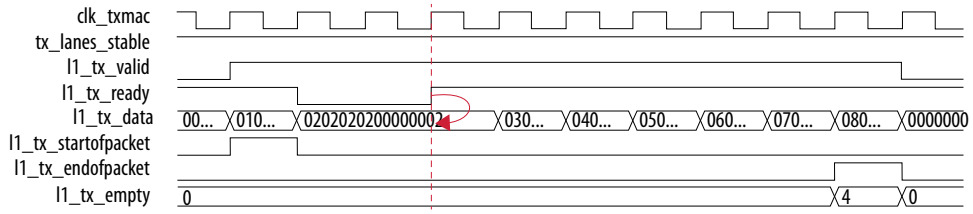


Figure 30. Client to 25G Ethernet Intel FPGA IP MAC Avalon Streaming Interface when Ready Latency is 3 (1 of 2)

This timing diagram shows `l1_tx_ready` asserts before `l1_tx_valid` asserts. TX MAC captures data at `l1_tx_data` bus at the same clock cycle as `l1_tx_valid` asserts.

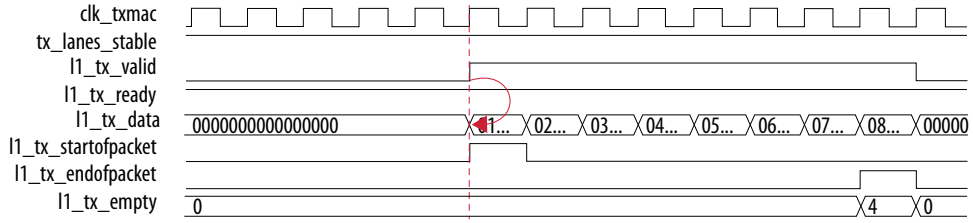
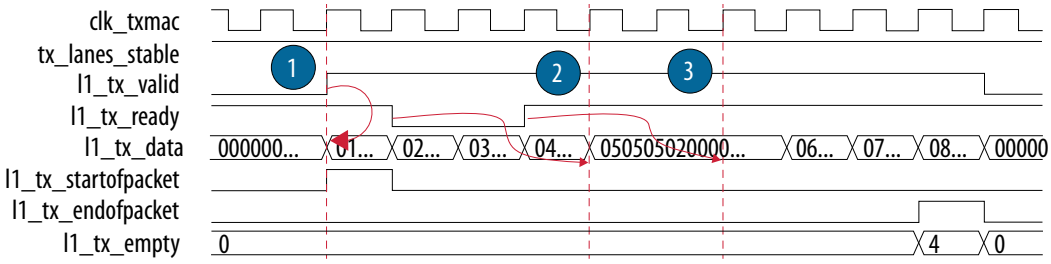


Figure 31. Client to 25G Ethernet Intel FPGA IP MAC Avalon Streaming Interface when Ready Latency is 3 (2 of 2)



Notes:

1. TX MAC captures data at l1_tx_data bus at the same clock cycle as l1_tx_valid asserts.
2. TX MAC stops capturing data at l1_tx_data bus 3 clock cycles after l1_tx_ready de-asserts.
3. TX MAC captures data at l1_tx_data bus 3 clock cycles after l1_tx_ready asserts.

Related Information

Avalon Interface Specifications

Detailed information about Avalon streaming interfaces and the Avalon streaming readyLatency parameter.

6.2. RX MAC Interface to User Logic

The RX MAC provides an Avalon streaming interface to the FPGA fabric. The datapath consists of a single 64-bit word.

Table 13. Avalon Streaming RX Datapath

All interface signals are clocked by the `clk_rxmac` clock.

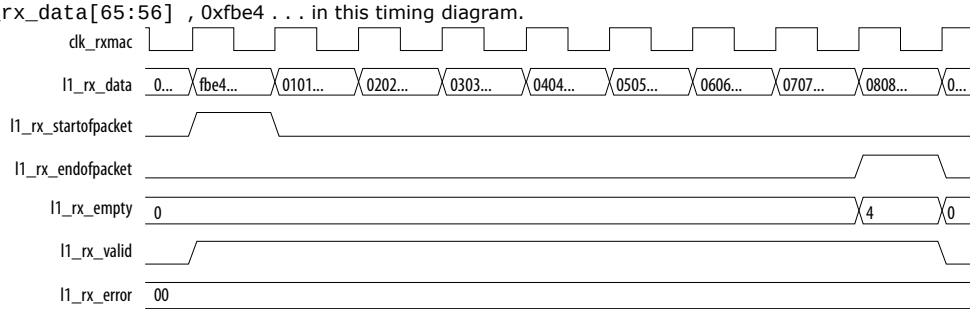
Signal	Direction	Description
`clk_rxmac`	Output	Clock for the RX MAC. Recovered from the incoming data. This clock is guaranteed stable when `rx_pcs_ready` is asserted. The frequency of this clock is 390.625 MHz. **All RX MAC interface signals are synchronous to `clk_rxmac`.**
`l1_rx_data[63:0]`	Output	Data output from the MAC. Bit[63] is the MSB and bit[0] is the LSB. Bytes are read in the usual left to right order. The IP core reverses the byte order to meet the requirements of the Ethernet standard.
`l1_rx_valid`	Output	When asserted, indicates that `l1_rx_data[63:0]` is driving valid data. If you turn off **Enable RS-FEC**, the IP core asserts this signal continuously between the assertions of `l1_tx_startofpacket` and `l1_tx_endofpacket` for the same packet. However, if you turn on **Enable RS-FEC**, the IP core drives IDLE cycles during alignment marker cycles.
`l1_rx_startofpacket`	Output	When asserted, indicates the first byte of a frame.
`l1_rx_endofpacket`	Output	When asserted, indicates the last data byte of a frame, before the frame check sequence (FCS). In CRC pass-through mode, it is the last byte of the FCS. The packet can end at any byte position.
`l1_rx_empty[2:0]`	Output	Specifies the number of empty bytes when `l1_rx_endofpacket` is asserted. The packet can end at any byte position. The empty bytes are the low-order bytes.
`l1_rx_error[5:0]`	Output	When asserted in the same cycle as `l1_rx_endofpacket`, indicates the current packet should be treated as an error packet. The 6 bits of `l1_rx_error` specify the following errors: • `l1_rx_error[5]`: Unused. • `l1_rx_error[4]`: Payload length error. If the length field is <1535 bytes (0x600 bytes), the received payload length is less than what is advertised in the payload length field. • `l1_rx_error[3]`: Oversized frame. The frame size is greater than the value specified in the `MAX_RX_SIZE_CONFIG` register. • `l1_rx_error[2]`: Undersized frame – The frame size is less than 64 bytes. Frame size = header size + payload size. • `l1_rx_error[1]`: CRC Error. The computed CRC value differs from the received CRC. • `l1_rx_error[0]`: Malformed packet. The packet is terminated with a non-terminate control character. When this bit is asserted, `l1_rx_error[1]` is also asserted.
`l1_rxstatus_valid`	Output	When asserted, indicates that `l1_rxstatus_data` is driving valid data.
`l1_rxstatus_data[39:0]`	Output	Specifies information about the received frame. The following fields are defined: • Bit[39]: When asserted, indicates a PFC frame • Bit[38]: When asserted, indicates a unicast frame • Bit[37]: When asserted, indicates a multicast frame • Bit[36]: When asserted, indicates a broadcast frame • Bit[35]: When asserted, indicates a pause frame • Bit[34]: When asserted, indicates a control frame • Bit[33]: When asserted, indicates a VLAN frame

continued...

Signal	Direction	Description
		• Bit[32]: When asserted, indicates a stacked VLAN frame • Bits[31:16]: Specifies the frame length from the first byte of the destination address to the last bye of the FCS • Bits[15:0]: Specifies the payload length
pause_receive_rx[FCQN-1:0]	Output	Each bit of pause_receive_rx[FCQN-1:0] indicates that the corresponding queue is being paused.

Figure 32. 25G Ethernet Intel FPGA IP MAC to Client Avalon Streaming Interface

l1_rx_data reception order is highest byte to lowest byte. The first byte of the destination address is on l1_rx_data[65:56] , 0xfbe4 . . . in this timing diagram.



Related Information

Avalon Interface Specifications
Detailed information about Avalon streaming interfaces.

6.3. Transceivers

The transceivers require a separately instantiated advanced transmit (ATX) PLL to generate the high speed serial clock. In many cases, the same ATX PLL can serve as input to an additional transceiver that has similar input clocking requirements. In comparison to the fractional PLL (fPLL) and clock multiplier unit PLL, the ATX PLL has the best jitter performance and supports the highest frequency operation.

Table 14. Transceiver Signals

Signal	Direction	Description
tx_serial	Output	TX transceiver signal. Each tx_serial bit becomes two physical pins that form a differential pair.
rx_serial	Input	RX transceiver signals. Each rx_serial bit becomes two physical pins that form a differential pair.
clk_ref	Input	The PLL reference clock. Input to the clock data recovery (CDR) circuitry in the RX PMA. The frequency of this clock is 644.53125 MHz.
tx_serial_clk	Input	High speed serial clock driven by the ATX PLL. The frequency of this clock is 12.890625 GHz.
tx_pll_locked	Input	Lock signal from ATX PLL. Indicates all ATX PLL(s) are locked.

Related Information

- Adding the Transceiver PLL on page 20
- Ethernet section of the *Intel Arria 10 Transceiver PHY User Guide*
  Provides more information about the PMA and PCS for Ethernet protocols.

6.4. Transceiver Reconfiguration Signals

You access the transceiver control and status registers using the transceiver reconfiguration interface. This is an Avalon memory-mapped interface.

The Avalon memory-mapped interface implements a standard memory-mapped protocol. You can connect an Avalon master to this bus to access the registers of the embedded Transceiver PHY IP core.

Table 15. Reconfiguration Interface Ports with Shared Native PHY Reconfiguration Interface

All interface signals are clocked by the `reconfig_clk` clock.

Port Name	Direction	Description
`reconfig_clk`	Input	Avalon clock. The clock frequency is 100-125 MHz. **All signals transceiver reconfiguration interface signals are synchronous to `reconfig_clk`.**
`reconfig_reset`	Input	Resets the Avalon memory-mapped interface and all of the registers to which it provides access.
`reconfig_write`	Input	Write enable signal. Signal is active high.
`reconfig_read`	Input	Read enable signal. Signal is active high.
`reconfig_address[10:0]`	Input	Address bus.
`reconfig_writedata[31:0]`	Input	A 32-bit data write bus. `reconfig_address` specifies the address.
`reconfig_readdata[31:0]`	Output	A 32-bit data read bus. Drives read data from the specified address. Signal is valid after `reconfig_waitrequest` is deasserted.
`reconfig_waitrequest`	Output	Indicates the Avalon memory-mapped interface is busy. Keep the `reconfig_write` or `reconfig_read` asserted until `reconfig_waitrequest` is deasserted.

Related Information

Intel Arria 10 Transceiver PHY User Guide
Provides more information about the transceiver reconfiguration interface, including timing diagrams for reads and writes.

6.5. Avalon Memory-Mapped Management Interface

You access control and status registers using an Avalon memory-mapped management 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 csr_rst_n signal. Asserting the csr_rst_n signal resets all control, status, and statistics registers; while this reset is in process, the Avalon memory-mapped management interface does not respond.

Table 16. Avalon Memory-Mapped Management Interface

Note: All status_* signals are synchronous to clk_status signal.

Signal	Direction	Description
clk_status	Input	The clock that drives the control and status registers. The frequency of this clock is 100 MHz.
reset_status	Input	Connect this signal to 1'b0. This signal remains visible as a top-level signal for backward compatibility.
status_addr[15:0]	Input	Drives the Avalon memory-mapped register address.
status_read	Input	When asserted, specifies a read request.
status_write	Input	When asserted, specifies a write request.
status_readdata[31:0]	Output	Drives read data. Valid when status_readdata_valid is asserted.
status_readdata_valid	Output	When asserted, indicates that status_read_data[31:0] is valid.
status_writedata[31:0]	Input	Drives the write data. The packet can end at any byte position. The empty bytes are the low-order bytes.
status_waitrequest	Output	Indicates that the control and status interface is not ready to complete the transaction. status_waitrequest is only used for read transactions.

Related Information

- Control, Status, and Statistics Register Descriptions on page 67
  Information about the 25G Ethernet Intel FPGA IP core registers you can access through the Avalon memory-mapped management interface.
- Typical Read and Write Transfers section in the *Avalon Interface Specifications*
  Describes typical Avalon memory-mapped read and write transfers with a slave-controlled waitrequest signal.

6.6. 1588 PTP Interface Signals

Table 17. Signals of the 1588 Precision Time Protocol Interface

Signals are clocked by clk_rxmac or clk_txmac, as specified. All 64-bit output signals are in the Intel 64-bit TOD format, and you are expected to drive all 64-bit input signals in this format.

Signal Name	Direction	Description
PTP Interface to TOD module		
tx_time_of_day_96b_data[95:0]	Input	Current V2-format (96-bit) TOD in clk_txmac clock domain. Connect this signal to the external TOD module.

continued...

Signal Name	Direction	Description
		This signal is available only if you set the **Time of day format** parameter to the value of **Enable 96-bit timestamp format** or **Enable both formats**.
tx_time_of_day_64b_data[63:0]	Input	Current 64-bit TOD in clk_txmac clock domain. Connect this signal to the external TOD module. This signal is available only if you set the **Time of day format** parameter to the value of **Enable 64-bit timestamp format** or **Enable both formats**.
rx_time_of_day_96b_data[95:0]	Input	Current V2-format (96-bit) TOD in clk_rxmac clock domain. Connect this signal to the external TOD module. This signal is available only if you set the **Time of day format** parameter to the value of **Enable 96-bit timestamp format** or **Enable both formats**.
rx_time_of_day_64b_data[63:0]	Input	Current 64-bit TOD in clk_rxmac clock domain. Connect this signal to the external TOD module. This signal is available only if you set the **Time of day format** parameter to the value of **Enable 64-bit timestamp format** or **Enable both formats**.
PTP Interface to Client		
TX Signals Related to One Step Processing		
tx_etstamp_ins_ctrl_timestamp_insert	Input	Indicates the current packet on the TX client interface is a 1588 PTP packet, and directs the IP core to process the packet in one-step processing insertion mode. In this mode, the IP core overwrites the timestamp of the packet with the timestamp field when the packet appears on the TX Ethernet link. The TX client must assert and deassert this signal synchronously with the TX SOP signal for the 1588 PTP packet. If the TX client asserts this signal simultaneously with tx_etstamp_ins_ctrl_residence_time_update, the results are undefined.
tx_etstamp_ins_ctrl_residence_time_update	Input	Indicates the current packet on the TX client interface is a 1588 PTP packet, and directs the IP core to process the packet in one-step processing correction mode. In this mode, the IP core adds the latency through the IP core (residence time) to the current contents of the timestamp field. The TX client must assert and deassert this signal synchronously with the TX SOP signal for the 1588 PTP packet. If the TX client asserts this signal simultaneously with either of tx_etstamp_ins_ctrl_timestamp_insert or tx_egress_timestamp_request_valid, the results are undefined.
tx_etstamp_ins_ctrl_ingress_timestamp_96b[95:0]	Input	Indicates the V2-format TOD when the packet entered the system. The TX client must ensure this signal is valid in each TX SOP cycle when it asserts tx_etstamp_ins_ctrl_residence_time_update. The TX client must maintain the desired value on this signal while the TX SOP signal is asserted. This signal is useful only in transparent clock mode when the TX client asserts tx_etstamp_ins_ctrl_residence_time_update. This signal is available only if you set the **Time of day format** parameter to the value of **Enable 96-bit timestamp format** or **Enable both formats**.
tx_etstamp_ins_ctrl_ingress_timestamp_64b[63:0]	Input	Indicates the TOD (in Intel 64-bit format) when the packet entered the system. The TX client must ensure this signal is valid in each TX SOP cycle when it asserts tx_etstamp_ins_ctrl_residence_time_update. The TX client must maintain the desired value on this signal while the TX SOP

continued...

Signal Name	Direction	Description
		signal is asserted. This signal is useful only in transparent clock mode when the TX client asserts `tx_etstamp_ins_ctrl_residence_time_update`. This signal is available only if you set the **Time of day format** parameter to the value of **Enable 64-bit timestamp format** or **Enable both formats**.
`tx_etstamp_ins_ctrl_timestamp_format`	Input	Specifies the timestamp format (V1 or V2 format) for the current packet if the TX client simultaneously asserts `tx_etstamp_ins_ctrl_timestamp_insert`. Values are: • 1'b0: 96-bit timestamp format (V2) • 1'b1: 64-bit timestamp format (V1) The TX client must maintain the desired value on this signal while the TX SOP signal is asserted. If the client specifies the V1 format, you read and write the V1 format TOD (32 bits of seconds and 32 bits of nanoseconds) in bits [79:16] of the 96-bit timestamp and TOD signals. *Note:* If you set the **Time of day format** parameter to the value of **Enable 64-bit timestamp format**, the results of asserting `tx_etstamp_ins_ctrl_timestamp_insert` are undefined. Therefore, the timestamp in any case maps to the 96-bit signals.
`tx_etstamp_ins_ctrl_residence_time_calc_format`	Input	Specifies the TOD format (64-bit TOD format or the V2 96-bit TOD format) for the current packet if the TX client simultaneously asserts `tx_etstamp_ins_ctrl_residence_time_update`. Values are: • 1'b0: 96-bit TOD format (V2) • 1'b1: 64-bit TOD format (48-bit ns and 16-bit fns) The TX client must maintain the desired value on this signal while the TX SOP signal is asserted. If you set the **Time of day format** parameter to the value of **Enable 64-bit timestamp format** or **Enable both formats**, and the client specifies the 64-bit TOD format, the IP core uses the 64-bit TOD format for residence time calculation. If you set the **Time of day format** parameter to the value of **Enable 64-bit timestamp format** and the client specifies the 96-bit format (V2), the results are undefined.
`tx_etstamp_ins_ctrl_offset_timestamp[15:0]`	Input	Specifies the byte offset of the timestamp information in the current packet if the TX client simultaneously asserts `tx_etstamp_ins_ctrl_timestamp_insert`. The IP core overwrites the value at this offset. The TX client must maintain the desired value on this signal while the TX SOP signal is asserted. If the packet supports V2 format, the timestamp has 96 bits. In this case, the IP core inserts ten bytes (bits [95:16]) of the timestamp at this offset and the remaining two bytes (bits [15:0]) of the timestamp at the offset specified in `tx_etstamp_ins_ctrl_offset_correction_field`. The TX client must ensure that: • The offset includes the entire timestamp in a single packet. • If the packet is more than 256 bytes, the offset supports inclusion of the entire timestamp in the first 256 bytes of the packet. • The timestamp bytes do not overlap with the bytes in any other field, including the UDP checksum field. (If these particular two fields overlap, the result is undefined).
`tx_etstamp_ins_ctrl_offset_correction_field[15:0]`	Input	If the TX client simultaneously asserts `tx_etstamp_ins_ctrl_residence_time_update`, this signal specifies the byte offset of the correction field in the current packet. If the TX client simultaneously asserts `tx_etstamp_ins_ctrl_timestamp_insert` and deasserts (sets to the value of 0) the `tx_etstamp_ins_ctrl_timestamp_format` signal, this signal specifies the byte offset of bits [15:0]] of the timestamp.

continued...

Signal Name	Direction	Description
		The TX client must maintain the desired value on this signal while the TX SOP signal is asserted. In addition, the TX client must ensure that: • The offset includes the entire correction field or timestamp in a single packet. • If the packet is more than 256 bytes, the offset supports inclusion of the entire timestamp or correction field in the first 256 bytes of the packet. • The correction field or timestamp bytes do not overlap with the bytes in any other field, including the UDP checksum field. (If these particular two fields overlap, the result is undefined).
`tx_etstamp_ins_ctrl_checksum_zero`	Input	The TX client asserts this signal during a TX SOP cycle to tell the IP core to zero the UDP checksum in the current packet. If the TX client asserts the `tx_etstamp_ins_ctrl_checksum_correct` signal, it cannot assert this signal. This signal is meaningful only in one-step clock mode. A zeroed UDP checksum indicates the checksum value is not necessarily correct. This information is useful to tell the application to skip checksum checking of UDP IPv4 packets. This function is illegal for UDP IPv6 packets.
`tx_etstamp_ins_ctrl_offset_checksum_field[15:0]`	Input	Indicates the byte offset of the UDP checksum in the current packet. The TX client must ensure this signal has a valid value during each TX SOP cycle when it also asserts the `tx_etstamp_ins_ctrl_checksum_zero` signal. Holds the byte offset of the two bytes in the packet that the IP core should reset. This signal is meaningful only in one-step clock mode. The TX client must ensure that: • The offset includes the entire checksum in a single packet. • The checksum bytes do not overlap with the bytes in any other field, including the timestamp bytes. (If these particular two fields overlap, the result is undefined).
`tx_etstamp_ins_ctrl_checksum_correct`	Input	The TX client asserts this signal during a TX SOP cycle to tell the IP core to update (correct) the UDP checksum in the current packet. If the TX client asserts the `tx_etstamp_ins_ctrl_checksum_zero` signal, it cannot assert this signal. This signal is meaningful only in one-step clock mode. The application must assert this signal for correct processing of UDP IPv6 packets.
`tx_etstamp_ins_ctrl_offset_checksum_correction[15:0]`	Input	Indicates the byte offset of the UDP checksum correction field in the current packet represented by the extended bytes before CRC. The TX client must ensure this signal has a valid value during each TX SOP cycle when it also asserts the `tx_etstamp_ins_ctrl_checksum_correct` signal. Holds the byte offset of the two bytes in the packet that the IP core should correct. This signal is meaningful only in one-step clock mode. The TX client must ensure that: • The offset and length of the checksum correction field includes the entire two bytes of the checksum correction field in a single packet. • The checksum bytes do not overlap with the bytes in any other field, including the timestamp bytes. (If these particular two fields overlap, the result is undefined). • The end of the UDP payload of the PTP packet is extended by 2 bytes. The MAC function modifies the extended bytes to ensure that the UDP checksum remains uncompromised.
`tx_egress_asymmetry_update`	Input	Indicates the IP core should include the value in the TX_PTP_ASYM_DELAY register in its correction calculations. The TX client must maintain the desired value on this signal while the TX SOP signal is asserted. This option is useful in one-step correction mode.

continued...

Signal Name	Direction	Description
TX Signals Related to Two Step Processing		
tx_egress_timestamp_request_valid	Input	Indicates the current packet on the TX client interface is a 1588 PTP packet, and directs the IP core to process the packet in two-step processing mode. In this mode, the IP core outputs the timestamp of the packet when it exits the IP core, and does not modify the packet timestamp information. The TX client must assert and deassert this signal synchronously with the TX SOP signal for the 1588 PTP packet. If the TX client asserts this signal simultaneously with tx_etstamp_ins_ctrl_residence_time_update, the results are undefined.
tx_egress_timestamp_96b_data[95:0]	Output	Provides the V2-format timestamp when a 1588 PTP frame begins transmission on the Ethernet link. Value is valid when the tx_egress_timestamp_96b_valid signal is asserted. This signal is meaningful only in two-step clock mode. This signal is available only if you set the **Time of day format** parameter to the value of **Enable 96-bit timestamp format** or **Enable both formats**.
tx_egress_timestamp_96b_valid	Output	Indicates that the tx_egress_timestamp_96b_data and tx_egress_timestamp_96b_fingerprint signals are valid in the current clk_txmac clock cycle. This signal is meaningful only in two-step clock mode. This signal is available only if you set the **Time of day format** parameter to the value of **Enable 96-bit timestamp format** or **Enable both formats**.
tx_egress_timestamp_64b_data[63:0]	Output	Provides the timestamp when a 1588 PTP frame begins transmission on the Ethernet link. Value is valid when the tx_egress_timestamp_64b_valid signal is asserted. This signal is meaningful only in two-step clock mode. This signal is available only if you set the **Time of day format** parameter to the value of **Enable 64-bit timestamp format** or **Enable both formats**.
tx_egress_timestamp_64b_valid	Output	Indicates that the tx_egress_timestamp_64b_data and tx_egress_timestamp_64b_fingerprint signals are valid in the current clk_txmac clock cycle. This signal is meaningful only in two-step clock mode. This signal is available only if you set the **Time of day format** parameter to the value of **Enable 64-bit timestamp format** or **Enable both formats**.
tx_egress_timestamp_request_fingerprint[(W-1):0] where W is the value between 1 and 32, inclusive, that you specify for the **Fingerprint width** parameter	Input	Fingerprint of the current packet. The TX client must assert and deassert this signal synchronously with the TX SOP signal for the 1588 PTP packet.
tx_egress_timestamp_96b_fingerprint[(W-1):0] where W is the value between 1 and 32, inclusive, that you specify for the **Fingerprint width** parameter	Output	Provides the fingerprint of the 1588 PTP frame currently beginning transmission on the Ethernet link. Value is valid when the tx_egress_timestamp_96b_valid signal is asserted. This signal is available only if you set the **Time of day format** parameter to the value of **Enable 96-bit timestamp format** or **Enable both formats**.
tx_egress_timestamp_64b_fingerprint[(W-1):0] where W is the value between 1 and 32, inclusive, that you specify for the **Fingerprint width** parameter	Output	Provides the fingerprint of the 1588 PTP frame currently beginning transmission on the Ethernet link. Value is valid when the tx_egress_timestamp_64b_valid signal is asserted. This signal is available only if you set the **Time of day format** parameter to the value of **Enable 64-bit timestamp format** or **Enable both formats**.

continued...

Signal Name	Direction	Description
RX Signals		
`rx_ingress_timestamp_96b_data[95:0]`	Output	Whether or not the current packet on the RX client interface is a 1588 PTP packet, indicates the V2-format timestamp when the IP core received the packet on the Ethernet link. The IP core provides a valid value on this signal in the same cycle it asserts the RX SOP signal for 1588 PTP packets. This signal is available only if you set the **Time of day format** parameter to the value of **Enable 96-bit timestamp format** or **Enable both formats**.
`rx_ingress_timestamp_96b_valid`	Output	Indicates that the `rx_ingress_timestamp_96b_data` signal is valid in the current cycle. This signal is redundant with the RX SOP signal for 1588 PTP packets. This signal is available only if you set the **Time of day format** parameter to the value of **Enable 96-bit timestamp format** or **Enable both formats**.
`rx_ingress_timestamp_64b_data[63:0]`	Output	Whether or not the current packet on the RX client interface is a 1588 PTP packet, indicates the 64-bit TOD (in Intel 64-bit format) when the IP core received the packet on the Ethernet link. The IP core provides a valid value on this signal in the same cycle it asserts the RX SOP signal for 1588 PTP packets. This signal is available only if you set the **Time of day format** parameter to the value of **Enable 64-bit timestamp format** or **Enable both formats**.
`rx_ingress_timestamp_64b_valid`	Output	Indicates that the `rx_ingress_timestamp_64b_data` signal is valid in the current cycle. This signal is redundant with the RX SOP signal for 1588 PTP packets. This signal is available only if you set the **Time of day format** parameter to the value of **Enable 64-bit timestamp format** or **Enable both formats**.

Related Information

1588 PTP Registers on page 84

6.7. Miscellaneous Status and Debug Signals

The miscellaneous status and debug signals are asynchronous.

Table 18. Miscellaneous Status and Debug Signals

Signal	Direction	Description
`tx_lanes_stable`	Output	Asserted when all TX lanes are stable and ready to transmit data.
`rx_block_lock`	Output	Signal is asserted when 64B/66B sync header is found consecutively for at least 64 clock cycles by the RX PCS.
`rx_am_lock`	Output	If you turn on **Enable RS-FEC** in the parameter editor, this signal is asserted when alignment marker lock status is achieved. If you turn off **Enable RS-FEC** in the parameter editor, this signal behaves the same as the `rx_block_lock` signal.
`rx_pcs_ready`	Output	Signal is asserted when `rx_block_lock` is asserted.
`local_fault_status`	Output	Asserted when the RX MAC detects a local fault. This signal is available if you turn on **Enable link fault generation** in the parameter editor.

continued...

Signal	Direction	Description
remote_fault_status	Output	Asserted when the RX MAC detects a remote fault. This signal is available if you turn on **Enable link fault generation** in the parameter editor.
unidirectional_en	Output	Asserted if the IP core includes *Clause 66* for unidirectional support. This signal is available if you turn on **Enable link fault generation** in the parameter editor.
link_fault_gen_en	Output	Asserted if the IP core includes *Clause 66* for unidirectional support. This signal is available if you turn on **Enable link fault generation** in the parameter editor.

Related Information

Debugging the Link on page 88

6.8. Reset Signals

The IP core has three external hard reset inputs. These resets are asynchronous and are internally synchronized. Assert resets for ten cycles or until you observe the effect of their specific reset. Asserting the external hard reset csr_rst_n returns control and status registers to their original values. rx_pcs_ready and tx_lanes_stable are asserted when the IP core has exited reset successfully.

Table 19. Reset Signals

Signal	Direction	Description
tx_rst_n	Input	Active low hard reset signal. Resets the TX interface, including the TX PCS and TX MAC. This reset leads to the deassertion of the tx_lanes_stable output signal.
rx_rst_n	Input	Active low hard reset signal. Resets the RX interface, including the RX PCS and RX MAC. This reset leads to the deassertion of the rx_pcs_ready output signal.
csr_rst_n	Input	Active low hard global reset. Resets the full IP core. Resets the TX MAC, RX MAC, TX PCS, RX PCS, adapters, transceivers, and control, status, and statistic registers. This reset leads to the deassertion of the tx_lanes_stable and rx_pcs_ready output signals.

7. Control, Status, and Statistics Register Descriptions

This section provides information about the memory-mapped registers. You access these registers using the IP core Avalon memory-mapped control and status interface. The registers use 32-bit addresses; they are not byte addressable.

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 no 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.

Table 20. Register Base Addresses

Word Offset	Register Type
0x300-0x3FF	PHY registers
0x400-0x4FF	TX MAC registers
0x500-0x5FF	RX MAC registers
0x600-0x708	Pause and Priority-Based Flow Control registers
0x800-0x8FF	Statistics Counter registers - TX direction
0x900-0x9FF	Statistics Counter registers - RX direction
0xA00-0xAFF	TX 1588 PTP registers
0xB00-0xBFF	RX 1588 PTP registers
0xC00-0xCFF	TX Reed-Solomon FEC registers
0xD00-0xDFF	RX Reed-Solomon FEC registers

Note: Do not attempt to access any register address that is Reserved or undefined. Accesses to registers that do not exist in your IP core variation have unspecified results.

Related Information

Avalon Memory-Mapped Management Interface on page 60
Interface to access the 25G Ethernet Intel FPGA IP core registers.

7.1. PHY Registers

Table 21. PHY Registers

The global hard reset `csr_rst_n` resets all of these registers. The TX reset `tx_rst_n` and RX reset `rx_rst_n` signals do not reset these registers.

Addr	Name	Description	Reset	Access
0x300	REVID	IP core PHY module revision ID.	0x0916 2016	RO
0x301	SCRATCH	Scratch register available for testing.	0x0000 0000	RW
0x302	PHY_NAME_0	First characters of IP core variation identifier string, "0025". The "00" is unprintable.	0x0000 3235	RO
0x303	PHY_NAME_1	Next characters of IP core variation identifier string, "00GE". The "00" is unprintable.	0x0000 4745	RO
0x304	PHY_NAME_2	Final characters of IP core variation identifier string, "0pcs". The "0" is unprintable.	0x0070 6373	RO
0x310	PHY_CONFIG	PHY configuration registers. The following bit fields are defined: • Bit[0]: : `eio_sys_rst`. Full system reset (except registers). Set this bit to initiate the internal reset sequence. • Bit[1]: `soft_txp_rst`. TX soft reset. Resets TX PCS, MAC, and adapter. • Bit[2]: `soft_rxp_rst`. RX soft reset. Resets RX PCS, MAC, and adapter. • Bit[3]: Reserved. • Bit[4]: `set_ref_lock`. Directs the RX CDR PLL to lock to the reference clock. • Bit[5]: `set_data_lock`. Directs the RX CDR PLL to lock to data. • Bits[31:6]: Reserved.	26'hX_2'b0_1'bX_3'b0 (1)	RW
0x312	WORD_LOCK	When asserted, indicates that the virtual channel has identified 66 bit block boundaries in the serial data stream.	31'hX1'b0 (1)	RO
0x313	EIO_SLOOP	Serial PMA loopback. Setting a bit puts the corresponding transceiver in serial loopback mode. In serial loopback mode, the TX lane loops back to the RX lane on an internal loopback path.	31'hX1'b0 (1)	RW
0x314	EIO_FLAG_SEL	Supports indirect addressing of individual FIFO flags in the PCS Native PHY IP core. Program this register with the encoding for a specific FIFO flag. The flag values (one per transceiver) are then accessible in the EIO_FLAGS register. The value in the EIO_FLAG_SEL register directs the IP core to make available the following FIFO flag: • 3'b000: TX FIFO full • 3'b001: TX FIFO empty • 3b010: TX FIFO partially full	29'hX3'b0 (1)	RW

continued...

(1) X means "Don't Care".

(2) Register value convert in decimal.

Addr	Name	Description	Reset	Access
		• 3'b011: TX FIFO partially empty • 3b100: RX FIFO full • 3b101: RX FIFO empty • 3b110: RX FIFO partially full • 3b111: RX FIFO partially empty		
0x315	EIO_FLAGS	PCS indirect data. To read a FIFO flag, set the value in the EIO_FLAG_SEL register to indicate the flag you want to read. After you specify the flag in the EIO_FLAG_SEL register, each bit [n] in the EIO_FLAGS register has the value of that FIFO flag for the transceiver channel for lane [n].	31'hX1'b0 (1)	RO
0x321	EIO_FREQ_LOCK	Each asserted bit indicates that the corresponding lane RX clock data recovery (CDR) phase-locked loop (PLL) is locked.	31'hX1'b0 (1)	RO
0x322	PHY_CLK	The following encodings are defined: • Bit[0]: Indicates if the TX PCS is ready • Bit [1]: Indicates if the TX MAC PLL is locked. • Bit[2]: Indicates if the RX CDR PLL is locked.	29'hX3'b00 (1)	RO
0x323	FRM_ERR	The IP core asserts bit [0] if it identifies a frame error. You can read this register to determine if the IP core sustains a low number of frame errors, below the threshold to lose word lock. This bit is sticky, unless the IP core loses word lock. Write 1'b1 to the SCLR_FRM_ERR register to clear. If the IP core loses word lock, it clears this register.	31'hX1'b0 (1)	RO
0x324	SCLR_FRM_ERR	Synchronous clear for FRM_ERR register. Write 1'b1 to this register to clear the FRM_ERR register and bit [1] of the LANE_DESKEWED register. A single bit clears all sticky framing errors. This bit does not auto-clear. Write a 1'b0 to continue logging frame errors.	0x0	RW
0x325	EIO_RX_SOFT_PURGE_S	Reserved.	0x0000	RO
0x326	RX_PCS_FULLY_ALIGNED_S	Indicates the RX PCS is fully aligned and ready to accept traffic. • Bit[0]: RX PCS fully aligned status. • Bit[1]: RX PCS bit error rate status. A bit value of 1 indicates a bit error rate higher than 10^{-4} or there are at least 16 errors within 50 us. This bit value is only valid when the link fault generation is enabled.	31'hX1'b0 (1)	RO
0x329	LANE_DESKEWED	The following encodings are defined:	30'hX2'b00 (1)	RO

continued...

(1) X means "Don't Care".

(2) Register value convert in decimal.

Addr	Name	Description	Reset	Access
		• Bit [0]: Indicates all lanes are deskewed. • Bit [1]: When asserted indicates a change in lanes deskewed status. To clear this sticky bit, write 1'b1 to the corresponding bit of the SCLR_FRM_ERR register. This is a latched signal.		
0x340	KHZ_REF	The register indicates the value of reference clock frequency. Apply the following definition for the frequency value: [(Register value [2] * clk_status)/10] KHZ	0x0000 0000	RO
0x341	KHZ_RX	The register indicates the value of RX clock (clk_rxmac) frequency. Apply the following definition for the frequency value: [(Register value [2] * clk_status)/10] KHZ	0x0000 0000	RO
0x342	KHZ_TX	The register indicates the value of TX clock (clk_txmac) frequency. Apply the following definition for the frequency value: [(Register value [2] * clk_status)/10] KHZ	0x0000 0000	RO

7.2. TX MAC Registers

Table 22. TX MAC Registers

Addr	Name	Description	Reset	Access
0x400	TXMAC_REVID	TX MAC revision ID for 25G TX MAC CSRs.	0x0916 2016	RO
0x401	TXMAC_SCRATCH	Scratch register available for testing.	0x0000 0000	RW
0x402	TXMAC_NAME_0	First 4 characters of IP core variation identifier string, "25gMACTxCSR".	0x3235 674D	RO
0x403	TXMAC_NAME_1	Next 4 characters of IP core variation identifier string, "ACTx".	0x4143 5478	RO
0x404	TXMAC_NAME_2	Final 4 characters of IP core variation identifier string, "0CSR". The "0" is unprintable.	0x0043 5352	RO

continued...

(1) X means "Don't Care".

(2) Register value convert in decimal.

(3) X means "Don't Care".

Addr	Name	Description	Reset	Access
0x405	LINK_FAULT	Link Fault Configuration Register. The following bits are defined: • `Force Remote Fault` bit[3]: When link fault generation is enabled, stops data transmission and forces transmission of a remote fault. • `Disable Remote Fault` bit[2]: When both link fault reporting and unidirectional transport are enabled, the core transmits data and does not transmit remote faults (RF). This bit takes effect when the value of this register is 28'hX4'b0111. • `Unidir Enable` bit[1]: When asserted, the core includes Clause 66 support for the remote link fault reporting on the Ethernet link. • `Link Fault Reporting Enable` bit[0]: The following encodings are defined: — 1'b1: The PCS generates the proper fault sequence on Ethernet link, when conditions are met. — 1'b0: The PCS does not generate the fault sequence.	28'hX_4'b0001 (3)	RW
0x407	MAX_TX_SIZE_CONFIG	Specifies the maximum TX frame length. Frames that are longer are considered oversized. They are transmitted, but also increment the CNTR_TX_OVERSIZE register. Bits [31:16] of this register are Reserved.	0xXXXX 2580 (3)	RW
0x40A	TXMAC_CONTROL	TX MAC Control Register. A single bit is defined: • Bit[1]: VLAN detection disabled. This bit is deasserted by default, implying VLAN detection is enabled.	30'hX2'b0X (3)	RW

7.3. RX MAC Registers

Table 23. RX MAC Registers

Addr	Name	Description	Reset	Access
0x500	RXMAC_REVID	RX MAC revision ID for 25G Ethernet IP core.	0x0916 2016	RO
0x501	RXMAC_SCRATCH	Scratch register available for testing.	0x0000 0000	RW
0x502	RXMAC_NAME_0	First 4 characters of IP core variation identifier string, "25gMACRxCSR".	0x3235 674D	RO
0x503	RXMAC_NAME_1	Next 4 characters of IP core variation identifier string, "ACRx".	0x4143 5278	RO
0x504	RXMAC_NAME_2	Final 4 characters of IP core variation identifier string, "0CSR". The "0" is unprintable.	0x0043 5352	RO

continued...

(3) X means "Don't Care".

(4) X means "Don't Care".

Addr	Name	Description	Reset	Access
0x506	MAX_RX_SIZE_CONFIG	Specifies the maximum frame length available. The MAC asserts l1_rx_error[3] when the length of the received frame exceeds the value of this register. If the IP core receives an Ethernet frame of size greater than the number of bytes specified in this register, and the IP core includes statistics registers, the IP core increments the 64-bit CNTR_RX_OVERSIZE counter.	0xXXXX 2580 (4)	RW
0x507	MAC_CRC_CONFIG	The RX CRC forwarding configuration register. The following encodings are defined: • 1'b0: Remove RX CRC, do not forward it to the RX client interface • 1'b1: Retain RX CRC, forward it to the RX client interface In either case, the IP core checks the incoming RX CRC and flags errors.	31'hX1'b0 (4)	RW
0x508	LINK_FAULT	Link Fault Status Register. For regular (non-unidirectional) Link Fault, implements *IEEE 802.3 Ethernet Clause 46.* For unidirectional Link Fault, implements *IEEE 802.3 Ethernet Clause 66.* If you turn on **Enable link fault generation**, the following bit fields are defined: • Bit[0]: A bit value of 1 indicates local fault is detected. • Bit[1]: A bit value of 1 indicates remote fault is detected. If you disable **Enable link fault generation**, bit[0] and [1] are always to zero.	30'hX2'b00 (4)	RO
0x50A	RXMAC_CONTROL	RX MAC Control Register. A single bit is defined: • Bit [1]: VLAN detection disabled. This bit is deasserted by default implying VLAN detection is enabled. • Bit [4]: Enable check for Preamble. By default, Preamble check is turned off. Write 1'b1 to this bit to enable preamble checking. This bit is a don't care when you turn on **Enable Preamble Passthrough**.	27'hX_5'b0XX0X (4)	RW

7.4. Pause/PFC Flow Control Registers

Some of the registers in this table cannot be updated during normal operation. To ensure correct operation, perform a soft reset by writing Bit[0] of the PHY_CONFIG (0x310) after updating registers that cannot be changed dynamically.

(4) X means "Don't Care".

Table 24. TX Flow Control Registers

Addr	Bit	Name	Description	Reset	Access
0x600	31:0	TX Flow Control Revision ID	Specifies the revision ID, "25GEFCTx CSR".	0x0916_2016	RO
0x601	31:0	TX Flow Control Scratch Pad	Scratch register for testing.	0	RW
0x602	31:0	TX Flow Control IP Core Variant 0	Specifies first 4 characters of IP core variation identifier ASCII string, "25GE ".	0x3235_4745	RO
0x603	31:0	TX Flow Control IP Core Variant 1	Next 4 characters of IP core variation identifier ASCII string, "FCTx".	0x4643_5478	RO
0x604	31:0	TX Flow Control IP Core Variant 2	Final 4 characters of IP core variation identifier ASCII string, "0CSR". The "0" is unprintable.	0x0043_5352	RO
0x605	(FCQN-1): 0	TX Flow Control Enable	Enables the IP core to generate XON and XOFF Pause/PFC flow control frames to the remote partner. The following encodings are defined: • 1'b0: XON or XOFF Pause/PFC flow control is disabled. • 1'b1: XON or XOFF Pause/PFC flow control is enabled. You can change this field dynamically.	1'b1 in each bit that corresponds to a queue	RW
	31:FCQN	Reserved	Reserved	0	RO
0x606	(FCQN-1): 0	TX Flow Control CSR XON/XOFF Request 1 One bit per queue	XON/XOF flow control frame request bit 0. Interpretation depends on whether the IP core is in 1-bit FC request mode or in 2-bit FC request mode. This register affects a flow control queue only if the corresponding bit of the TX Flow Control Enable register has the value of 1. In 2-bit mode, in addition, this register is active for a specific flow control queue only if the corresponding bit in the TX 2-bit Flow Control Request Mode register field (bits [(FCQN-1):0] of the register at offset 0x641) specifies that the flow control logic accepts input from this register. The following encodings are defined for 1-bit mode. The IP core reads the 1-bit mode value in TX Flow Control CSR XON/XOFF Request 0. • 0 = No request • 0 to 1 = Generate XOFF request • 1 = Continue to generate XOFF request • 1 to 0 = Generate XON request	0	RW

continued...

Addr	Bit	Name	Description	Reset	Access
			The following encodings are defined for 2-bit mode. The IP core reads the 2-bit mode value in {TX Flow Control CSR XON/XOFF Request 1, TX Flow Control CSR XON/XOFF Request 1}. • 00 = No request • 01 = XON request • 10 = XOFF request • 11 = Invalid You can modify the value of this field dynamically.		
	15:FCQN	Reserved	Reserved	0	RO
	(FCQN +15):16	TX Flow Control CSR XON/XOFF Request 1 1-bit per queue	In conjunction with Flow Control XON/XOFF Request 0 specifies a 2-bit request for XON/XOFF flow control frame transmission. This bit is the upper bit of the 2-bit control field. You can change the value of this field dynamically.	0	RW
	31:(FCQN +16)	Reserved	Reserved	0	RO
0x607	31:0	Reserved	Reserved	N/A	RO
0x608	31:0	Reserved	Reserved	N/A	RO
0x609	31:0	Reserved	Reserved	N/A	RO
0x60A	0	TX Pause Enable 1-bit	Determines whether receiving a valid Pause frame stops TX user data transmission. 1'b0: Transmission is not stopped 1'b1: Transmission stops You cannot change the value of this field dynamically.	0	RW
	31:1	Reserved	Reserved	0	RO
0x60B	31:0	Reserved	Reserved	N/A	RO
0x60C	31:0	Reserved	Reserved	N/A	RO
0x60D	31:0	TX Flow Control Destination Address Lower	Specifies the 48-bit Destination Address of the flow control frame. Contains the 32 LSB of the address field. You cannot modify the value of this field dynamically.	0xC2000001	RW
0x60E	15:0	TX Flow Control Destination Address Upper	Specifies the 48-bit Destination Address of flow control frame. Contains the 16 MSB of the address field. You cannot modify the value of this field dynamically.	0x0180	RW
	31:16	Reserved	Reserved	0	RO
0x60F	31:0	TX Flow Control Source Address Lower	Specifies the 48-bit Source Address of flow control frame. Contains the 32 LSB of the address field.	0xCBFC5ADD	RW
0x610	15:0	TX Flow Control Source Address Upper	Specifies the 48-bit Source Address of flow control frame. Contains the 16 MSB of the address field.	0xE100	RW

continued...

Addr	Bit	Name	Description	Reset	Access
			You cannot modify the value of this field dynamically.		
	31:16	Reserved	Reserved	0	RO
0x620, 0x621, …, 0x620+ (FCQN-1)	15:0	`TX Flow Control Quanta` 16-bit per queue	Specifies the pause quanta of Pause/PFC flow control frames to be sent to remote partner. You cannot modify the value of this field dynamically.	0xFFFF	RW
	31:16	Reserved	Reserved	0	RO
0x628, 0x629, …, 0x628+ (FCQN-1)	15:0	`TX Flow Control Signal XOFF Request Hold Quanta` 16-bit per queue	Specifies the separation between 2 consecutive XOFF flow control frames. You cannot modify the value of this field dynamically.	0xFFFF	RW
	31:16	Reserved	Reserved	0	RO
0x640	0	`TX Flow Control Select` 1-bit	Specifies whether the TX hardware generates Pause or PFC frames. Affects only PFC Queue 0. Usage example: You can synthesize a single PFC queue and use it for both Pause and PFC purpose. 1'b0: Pause 1'b1: PFC You cannot modify the value of this field dynamically.	1	RW
	31:1	Reserved.	Reserved.	0	RO
0x641	(FCQN-1): 0	`TX 2-bit Flow Control Request Mode` 1-bit per queue	Determines whether the `TX Flow Control CSR XON/XOFF Request` register or the `pause_insert_tx0` and `pause_insert_tx1` signals control XON/XOFF mode in 2-bit control mode. 1'b0: The `pause_insert_tx0` and `pause_insert_tx1` signals control requests 1'b1: The `TX Flow Control CSR XON/XOFF Request` register fields control requests You cannot modify the value of this field dynamically.	0	RW
	16	`TX Flow Control Request Mode` 1 bit for all queues	Determines whether the IP core is in TX flow control 1-bit mode or 2-bit mode. 1'b0: Use 1-bit mode to make TX flow control requests 1'b1: Use 2-bit mode to make TX flow control requests	0	RW
	31:17	Reserved	Reserved	0	RO

Table 25. RX Flow Control Registers

Addr	Bit	Name	Description	Reset	Access
0x700	31:0	RX Flow Control Revision ID	Provides the flow control revision, "25GEFCRx CSR".	0x0916_2016	RO
0x701	31:0	RX Flow Control Scratch Pad	Provides a register for debug.	0	RW
0x702	31:0	RX Flow Control IP Core Variant 0	First 4 characters of IP core variation identifier ASCII string, "25GE".	0x3235_4745	RO
0x703	31:0	RX Flow Control IP Core Variant 1	Next 4 characters of IP core variation identifier ASCII string, "FCRx".	0x4643_5278	RO
0x704	31:0	RX Flow Control IP Core Variant 2	Final 4 characters of IP core variation identifier ASCII string, "0CSR". The "0" is unprintable.	0x0043_5352	RO
0x705	(FCQN-1): 0]	RX PFC Enable 1 bit per queue	Determines whether receiving a valid PFC frame causes the PFC duration user interface to indicate a valid pause quanta duration to the user logic. 1'b0: Disable 1'b1: Enable You cannot modify the value of this field dynamically.	1'b1 in each bit that corresponds to a queue	RW
	31:FCQN8	Reserved	Reserved	0	RO
0x706	31:0	Reserved	Reserved	N/A	RO
0x707	31:0	RX Flow Control Destination Address Lower	Specifies the 48-bit Destination Address of the flow control frame. Contains the 32 LSB of the address field. You cannot modify the value of this field dynamically.	0xC2000001	RW
0x708	15:0	RX Flow Control Destination Address Upper	Specifies the 48-bit Destination Address of flow control frame. Contains the 16 MSB of the address field. You cannot modify the value of this field dynamically.	0x0180	RW
	31:16	Reserved	Reserved	0	RO

Related Information

Flow Control on page 38
Describes how the IP core uses the information in these registers to provide flow control functionality.

7.5. Statistics Registers

The 25G Ethernet Intel FPGA IP statistics registers count Ethernet traffic and errors. The 64-bit statistics registers are designed to roll over, to ensure timing closure on the FPGA. However, these registers should never roll over if the link is functioning properly. The statistics registers check the size of frames, which includes the following fields:

- Size of the destination address
- Size of the source address
- Size of the data
- Four bytes of CRC

The statistics counters module is a synthesis option. The statistics registers are counters that are implemented inside the CSR. When you turn on the **Enable MAC statistics counters** parameter in the 25G Ethernet Intel FPGA IP parameter editor, the counters are implemented in the CSR. When you turn off the **Enable MAC statistics counters** parameter in the 25G Ethernet Intel FPGA IP parameter editor, the counters are not implemented in the CSR, and read access to the counters returns undefined results.

After system power-up, the statistics counters have random values. You must clear these registers and confirm the system is stable before using their values. To clear the registers, use any of the following methods:

1. Assert `csr_rst_n` to clear both the TX and RX statistic counters.
2. Assert `tx_rst_n` to clear the TX statistic counters.
3. Assert `rx_rst_n` to clear the RX statistic counters.
4. Write 1'b1 to bit[0], `eio_sys_rst` of the PHY_CONFIG (0x310) register to clear both the TX and RX statistic counters.
5. Write 1'b1 to bit[1], `soft_txp_rst` of the PHY_CONFIG (0x310) register to clear the TX statistic counters.
6. Write 1'b1 to bit[2], `soft_rxp_rst` of the PHY_CONFIG (0x310) register to clear the RX statistic counters.
7. Write 1'b1 to bit[0] of the CNTR_TX_CONFIG (0x845) to clear the TX statistic counters.
8. Write 1'b1 to bit[0] of the CNTR_RX_CONFIG (0x945) to clear the RX statistic counters.

The configuration register at offset 0x845 allows you to clear all of the TX statistics counters. The configuration register at offset 0x945 allows you to clear all of the RX statistics counters. If you exclude these registers, you can monitor the statistics counter increment vectors that the IP core provides at the client side interface and maintain your own counters.

Reading the value of a statistics register does not affect its value.

To ensure that the counters you read are consistent, you should issue a shadow request to create a snapshot of all of the TX or RX statistics registers, by setting bit [2] of the configuration register at offset 0x845 or 0x945, respectively. Until you reset

this bit, the counters continue to increment but the readable values remain constant. You can read bit [1] of the status register at offset 0x846 or 0x946, respectively, to confirm your shadow request has been granted or released.

7.5.1. TX Statistics Registers

Table 26. Transmit Side Statistics Registers

The TX statistics counters do not reflect TX CRC errors the user forces by asserting the `l1_tx_error` signal.

Address	Name-	Description	Access
0x800	CNTR_TX_FRAGMENTS_LO	Number of transmitted frames less than 64 bytes and reporting a CRC error (lower 32 bits).	RO
0x801	CNTR_TX_FRAGMENTS_HI	Number of transmitted frames less than 64 bytes and reporting a CRC error (upper 32 bits).	RO
0x802	CNTR_TX_JABBERS_LO	Number of transmitted oversized frames reporting a CRC error (lower 32 bits).	RO
0x803	CNTR_TX_JABBERS_HI	Number of transmitted oversized frames reporting a CRC error (upper 32 bits).	RO
0x804	CNTR_TX_FCS_LO	Number of transmitted packets with FCS errors. (lower 32 bits).	RO
0x805	CNTR_TX_FCS_HI	Number of transmitted packets with FCS errors. (upper 32 bits).	RO
0x806	CNTR_TX_CRCERR_LO	Number of transmitted frames with a frame of length at least 64 reporting a CRC error (lower 32 bits).	RO
0x807	CNTR_TX_CRCERR_HI	Number of transmitted frames with a frame of length at least 64 reporting a CRC error (upper 32 bits).	RO
0x808	CNTR_TX_MCAST_DATA_ERR_LO	Number of errored multicast frames transmitted, excluding control frames (lower 32 bits).	RO
0x809	CNTR_TX_MCAST_DATA_ERR_HI	Number of errored multicast frames transmitted, excluding control frames (upper 32 bits).	RO
0x80A	CNTR_TX_BCAST_DATA_ERR_LO	Number of errored broadcast frames transmitted, excluding control frames (lower 32 bits).	RO
0x80B	CNTR_TX_BCAST_DATA_ERR_HI	Number of errored broadcast frames transmitted, excluding control frames (upper 32 bits).	RO
0x80C	CNTR_TX_UCAST_DATA_ERR_LO	Number of errored unicast frames transmitted, excluding control frames (lower 32 bits).	RO
0x80D	CNTR_TX_UCAST_DATA_ERR_HI	Number of errored unicast frames transmitted, excluding control frames (upper 32 bits).	RO
0x80E	CNTR_TX_MCAST_CTRL_ERR_LO	Number of errored multicast control frames transmitted (lower 32 bits).	RO
0x80F	CNTR_TX_MCAST_CTRL_ERR_HI	Number of errored multicast control frames transmitted (upper 32 bits).	RO
0x810	CNTR_TX_BCAST_CTRL_ERR_LO	Number of errored broadcast control frames transmitted (lower 32 bits).	RO
0x811	CNTR_TX_BCAST_CTRL_ERR_HI	Number of errored broadcast control frames transmitted (upper 32 bits).	RO
0x812	CNTR_TX_UCAST_CTRL_ERR_LO	Number of errored unicast control frames transmitted (lower 32 bits).	RO

continued...

Address	Name-	Description	Access
0x813	CNTR_TX_UCAST_CTRL_ERR_HI	Number of errored unicast control frames transmitted (upper 32 bits).	RO
0x814	CNTR_TX_PAUSE_ERR_LO	Number of errored pause frames transmitted (lower 32 bits).	RO
0x815	CNTR_TX_PAUSE_ERR_HI	Number of errored pause frames transmitted (upper 32 bits).	RO
0x816	CNTR_TX_64B_LO	Number of 64-byte transmitted frames (lower 32 bits), including the CRC field but excluding the preamble and SFD bytes.	RO
0x817	CNTR_TX_64B_HI	Number of 64-byte transmitted frames (upper 32 bits), including the CRC field but excluding the preamble and SFD bytes.	RO
0x818	CNTR_TX_65to127B_LO	Number of transmitted frames between 65–127 bytes (lower 32 bits).	RO
0x819	CNTR_TX_65to127B_HI	Number of transmitted frames between 65–127 bytes (upper 32 bits).	RO
0x81A	CNTR_TX_128to255B_LO	Number of transmitted frames between 128–255 bytes (lower 32 bits).	RO
0x81B	CNTR_TX_128to255B_HI	Number of transmitted frames between 128–255 bytes (upper 32 bits).	RO
0x81C	CNTR_TX_256to511B_LO	Number of transmitted frames between 256–511 bytes (lower 32 bits).	RO
0x81D	CNTR_TX_256to511B_HI	Number of transmitted frames between 256–511 bytes (upper 32 bits).	RO
0x81E	CNTR_TX_512to1023B_LO	Number of transmitted frames between 512–1023 bytes (lower 32 bits).	RO
0x81F	CNTR_TX_512to1023B_HI	Number of transmitted frames between 512–1023 bytes (upper 32 bits).	RO
0x820	CNTR_TX_1024to1518B_LO	Number of transmitted frames between 1024–1518 bytes (lower 32 bits).	RO
0x821	CNTR_TX_1024to1518B_HI	Number of transmitted frames between 1024–1518 bytes (upper 32 bits).	RO
0x822	CNTR_TX_1519toMAXB_LO	Number of transmitted frames of size between 1519 bytes and the number of bytes specified in the MAX_TX_SIZE_CONFIG register (lower 32 bits).	RO
0x823	CNTR_TX_1519toMAXB_HI	Number of transmitted frames of size between 1519 bytes and the number of bytes specified in the MAX_TX_SIZE_CONFIG register (upper 32 bits).	RO
0x824	CNTR_TX_OVERSIZE_LO	Number of oversized frames (frames with more bytes than the number specified in the MAX_TX_SIZE_CONFIG register) transmitted (lower 32 bits).	RO
0x825	CNTR_TX_OVERSIZE_HI	Number of oversized frames (frames with more bytes than the number specified in the MAX_TX_SIZE_CONFIG register) transmitted (upper 32 bits).	RO
0x826	CNTR_TX_MCAST_DATA_OK_LO	Number of valid multicast frames transmitted, excluding control frames (lower 32 bits).	RO
0x827	CNTR_TX_MCAST_DATA_OK_HI	Number of valid multicast frames transmitted, excluding control frames (upper 32 bits).	RO

continued...

Address	Name-	Description	Access
0x828	CNTR_TX_BCAST_DATA_OK_LO	Number of valid broadcast frames transmitted, excluding control frames (lower 32 bits).	RO
0x829	CNTR_TX_BCAST_DATA_OK_HI	Number of valid broadcast frames transmitted, excluding control frames (upper 32 bits).	RO
0x82A	CNTR_TX_UCAST_DATA_OK_LO	Number of valid unicast frames transmitted, excluding control frames (lower 32 bits).	RO
0x82B	CNTR_TX_UCAST_DATA_OK_HI	Number of valid unicast frames transmitted, excluding control frames (upper 32 bits).	RO
0x82C	CNTR_TX_MCAST_CTRL_LO	Number of valid multicast frames transmitted, excluding data frames (lower 32 bits).	RO
0x82D	CNTR_TX_MCAST_CTRL_HI	Number of valid multicast frames transmitted, excluding data frames (upper 32 bits).	RO
0x82E	CNTR_TX_BCAST_CTRL_LO	Number of valid broadcast frames transmitted, excluding data frames (lower 32 bits).	RO
0x82F	CNTR_TX_BCAST_CTRL_HI	Number of valid broadcast frames transmitted, excluding data frames (upper 32 bits).	RO
0x830	CNTR_TX_UCAST_CTRL_LO	Number of valid unicast frames transmitted, excluding data frames (lower 32 bits).	RO
0x831	CNTR_TX_UCAST_CTRL_HI	Number of valid unicast frames transmitted, excluding data frames (upper 32 bits).	RO
0x832	CNTR_TX_PAUSE_LO	Number of valid pause frames transmitted (lower 32 bits).	RO
0x833	CNTR_TX_PAUSE_HI	Number of valid pause frames transmitted (upper 32 bits).	RO
0x834	CNTR_TX_RUNT_LO	Number of transmitted runt packets (lower 32 bits). The IP core does not transmit frames of length less than nine bytes. The IP core pads frames of length nine bytes to 64 bytes to extend them to 64 bytes. Therefore, this counter does not increment in normal operating conditions.	RO
0x835	CNTR_TX_RUNT_HI	Number of transmitted runt packets (upper 32 bits). The IP core does not transmit frames of length less than nine bytes. The IP core pads frames of length nine bytes to 64 bytes to extend them to 64 bytes. Therefore, this counter does not increment in normal operating conditions.	RO
0x836–0x844	Reserved		
0x845	CNTR_TX_CONFIG	Bits[2:0]: Configuration of TX statistics counters: • Bit[2]: Shadow request (active high): When set to the value of 1, TX statistics collection is paused. The underlying counters continue to operate, but the readable values reflect a snapshot at the time the pause flag was activated. Write a 0 to release. • Bit[1]: Parity-error clear. When software sets this bit, the IP core clears the parity bit CNTR_TX_STATUS[0]. This bit (CNTR_TX_CONFIG[1]) is self-clearing. • Bit[0]: Software can set this bit to the value of 1 to reset all of the TX statistics registers at the same time. This bit is self-clearing. Bits[31:3] are Reserved.	RW
0x846	CNTR_TX_STATUS	• Bit[1]: Indicates that the TX statistics registers are paused (while CNTR_TX_CONFIG[2] is asserted). • Bit[0]: Indicates the presence of at least one parity error in the TX statistics counters. Bits[31:2] are Reserved.	RO

continued...

Address	Name-	Description	Access
0x847–0x85F	Reserved		
0x860	TxPayloadOctetsOK_LO	Number of transmitted payload bytes in frames with no FCS, undersized, oversized, or payload length errors. If VLAN detection is turned off for the TX MAC (bit[1] of the TX_MAC_CONTROL register at offset 0x40A has the value of 1), the IP core counts the VLAN header bytes (4 bytes for VLAN and 8 bytes for stacked VLAN) as payload bytes. This register is compliant with the requirements for a OctetsTransmittedOK in section 5.2.2.1.8 of the *IEEE Standard 802.3-2008*.	RO
0x861	TxPayloadOctetsOK_HI		RO
0x862	TxFrameOctetsOK_LO	Number of transmitted bytes in frames with no FCS, undersized, oversized, or payload length errors. This register is compliant with the requirements for ifOutOctets in RFC3635 (Managed Objects for Ethernet-like Interface Types) and TX etherStatsOctets in RFC2819(Remote Network Monitoring Management Information Base (RMON)).	RO
0x863	TxFrameOctetsOK_HI		RO

7.5.2. RX Statistics Registers

Table 27. Receive Side Statistics Registers

Address	Name	Description	Access
0x900	CNTR_RX_FRAGMENTS_LO	Number of received frames less than 64 bytes and reporting a CRC error (lower 32 bits)	RO
0x901	CNTR_RX_FRAGMENTS_HI	Number of received frames less than 64 bytes and reporting a CRC error (upper 32 bits)	RO
0x902	CNTR_RX_JABBERS_LO	Number of received oversized frames reporting a CRC error (lower 32 bits)	RO
0x903	CNTR_RX_JABBERS_HI	Number of received oversized frames reporting a CRC error (upper 32 bits)	RO
0x904	CNTR_RX_FCS_LO	Number of received packets with FCS errors. This register maintains a count of the number of pulses on the l<n>_rx_fcs_error or rx_fcs_error output signal (lower 32 bits)	RO
0x905	CNTR_RX_FCS_HI	Number of received packets with FCS errors. This register maintains a count of the number of pulses on the l<n>_rx_fcs_error output signal (upper 32 bits)	RO
0x906	CNTR_RX_CRCERR_LO	Number of received frames with a frame of length at least 64, with CRC error (lower 32 bits)	RO
0x907	CNTR_RX_CRCERR_HI	Number of received frames with a frame of length at least 64, with CRC error (upper 32 bits)	RO
0x908	CNTR_RX_MCAST_DATA_ERR_LO	Number of errored multicast frames received, excluding control frames (lower 32 bits)	RO
0x909	CNTR_RX_MCAST_DATA_ERR_HI	Number of errored multicast frames received, excluding control frames (upper 32 bits)	RO
0x90A	CNTR_RX_BCAST_DATA_ERR_LO	Number of errored broadcast frames received, excluding control frames (lower 32 bits)	RO
0x90B	CNTR_RX_BCAST_DATA_ERR_HI	Number of errored broadcast frames received, excluding control frames (upper 32 bits)	RO
0x90C	CNTR_RX_UCAST_DATA_ERR_LO	Number of errored unicast frames received, excluding control frames (lower 32 bits)	RO

continued...

Address	Name	Description	Access
0x90D	CNTR_RX_UCAST_DATA_ERR_HI	Number of errored unicast frames received, excluding control frames (upper 32 bits)	RO
0x90E	CNTR_RX_MCAST_CTRL_ERR_LO	Number of errored multicast control frames received (lower 32 bits)	RO
0x90F	CNTR_RX_MCAST_CTRL_ERR_HI	Number of errored multicast control frames received (upper 32 bits)	RO
0x910	CNTR_RX_BCAST_CTRL_ERR_LO	Number of errored broadcast control frames received (lower 32 bits)	RO
0x911	CNTR_RX_BCAST_CTRL_ERR_HI	Number of errored broadcast control frames received (upper 32 bits)	RO
0x912	CNTR_RX_UCAST_CTRL_ERR_LO	Number of errored unicast control frames received (lower 32 bits)	RO
0x913	CNTR_RX_UCAST_CTRL_ERR_HI	Number of errored unicast control frames received (upper 32 bits)	RO
0x914	CNTR_RX_PAUSE_ERR_LO	Number of errored pause frames received (lower 32 bits)	RO
0x915	CNTR_RX_PAUSE_ERR_HI	Number of errored pause frames received (upper 32 bits)	RO
0x916	CNTR_RX_64B_LO	Number of 64-byte received frames (lower 32 bits), including the CRC field but excluding the preamble and SFD bytes	RO
0x917	CNTR_RX_64B_HI	Number of 64-byte received frames (upper 32 bits), including the CRC field but excluding the preamble and SFD bytes	RO
0x918	CNTR_RX_65to127B_LO	Number of received frames between 65–127 bytes (lower 32 bits)	RO
0x919	CNTR_RX_65to127B_HI	Number of received frames between 65–127 bytes (upper 32 bits)	RO
0x91A	CNTR_RX_128to255B_LO	Number of received frames between 128 –255 bytes (lower 32 bits)	RO
0x91B	CNTR_RX_128to255B_HI	Number of received frames between 128 –255 bytes (upper 32 bits)	RO
0x91C	CNTR_RX_256to511B_LO	Number of received frames between 256 –511 bytes (lower 32 bits)	RO
0x91D	CNTR_RX_256to511B_HI	Number of received frames between 256 –511 bytes (upper 32 bits)	RO
0x91E	CNTR_RX_512to1023B_LO	Number of received frames between 512–1023 bytes (lower 32 bits)	RO
0x91F	CNTR_RX_512to1023B_HI	Number of received frames between 512 –1023 bytes (upper 32 bits)	RO
0x920	CNTR_RX_1024to1518B_LO	Number of received frames between 1024–1518 bytes (lower 32 bits)	RO
0x921	CNTR_RX_1024to1518B_HI	Number of received frames between 1024–1518 bytes (upper 32 bits)	RO
0x922	CNTR_RX_1519toMAXB_LO	Number of received frames between 1519 bytes and the maximum size defined in the MAX_RX_SIZE_CONFIG register (lower 32 bits)	RO

continued...

Address	Name	Description	Access
0x923	CNTR_RX_1519toMAXB_HI	Number of received frames between 1519 bytes and the maximum size defined in the MAX_RX_SIZE_CONFIG register (upper 32 bits)	RO
0x924	CNTR_RX_OVERSIZE_LO	Number of oversized frames (frames with more bytes than the number specified in the MAX_RX_SIZE_CONFIG register) received (lower 32 bits)	RO
0x925	CNTR_RX_OVERSIZE_HI	Number of oversized frames (frames with more bytes than the number specified in the MAX_RX_SIZE_CONFIG register) received (upper 32 bits)	RO
0x926	CNTR_RX_MCAST_DATA_OK_LO	Number of valid multicast frames received, excluding control frames (lower 32 bits)	RO
0x927	CNTR_RX_MCAST_DATA_OK_HI	Number of valid multicast frames received, excluding control frames (upper 32 bits)	RO
0x928	CNTR_RX_BCAST_DATA_OK_LO	Number of valid broadcast frames received, excluding control frames (lower 32 bits)	RO
0x929	CNTR_RX_BCAST_DATA_OK_HI	Number of valid broadcast frames received, excluding control frames (upper 32 bits)	RO
0x92A	CNTR_RX_UCAST_DATA_OK_LO	Number of valid unicast frames received, excluding control frames (lower 32 bits)	RO
0x92B	CNTR_RX_UCAST_DATA_OK_HI	Number of valid unicast frames received, excluding control frames (upper 32 bits)	RO
0x92C	CNTR_RX_MCAST_CTRL_LO	Number of valid multicast frames received, excluding data frames (lower 32 bits)	RO
0x92D	CNTR_RX_MCAST_CTRL_HI	Number of valid multicast frames received, excluding data frames (upper 32 bits)	RO
0x92E	CNTR_RX_BCAST_CTRL_LO	Number of valid broadcast frames received, excluding data frames (lower 32 bits)	RO
0x92F	CNTR_RX_BCAST_CTRL_HI	Number of valid broadcast frames received, excluding data frames (upper 32 bits)	RO
0x930	CNTR_RX_UCAST_CTRL_LO	Number of valid unicast frames received, excluding data frames (lower 32 bits)	RO
0x931	CNTR_RX_UCAST_CTRL_HI	Number of valid unicast frames received, excluding data frames (upper 32 bits)	RO
0x932	CNTR_RX_PAUSE_LO	Number of received pause frames, with or without error (lower 32 bits)	RO
0x933	CNTR_RX_PAUSE_HI	Number of received pause frames, with or without error (upper 32 bits)	RO
0x934	CNTR_RX_RUNT_LO	Number of received runt packets (lower 32 bits) A runt is a packet of size less than 64 bytes but greater than eight bytes. If a packet is eight bytes or smaller, it is considered a decoding error and not a runt frame, and the IP core does not flag it nor count it as a runt.	RO
0x935	CNTR_RX_RUNT_HI	Number of received runt packets (upper 32 bits) A runt is a packet of size less than 64 bytes but greater than eight bytes. If a packet is eight bytes or smaller, it is considered a decoding error and not a runt frame, and the IP core does not flag it nor count it as a runt.	RO
0x936–0x944	Reserved		

continued...

Address	Name	Description	Access
0x945	CNTR_RX_CONFIG	Bits[2:0]: Configuration of RX statistics counters: • Bit[2]: Shadow request (active high): When set to the value of 1, RX statistics collection is paused. The underlying counters continue to operate, but the readable values reflect a snapshot at the time the pause flag was activated. Write a 0 to release. • Bit[1]: Parity-error clear. When software sets this bit, the IP core clears the parity bit CNTR_RX_STATUS[0]. This bit (CNTR_RX_CONFIG[1]) is self-clearing. • Bit[0]: Software can set this bit to the value of 1 to reset all of the RX statistics registers at the same time. This bit is self-clearing. Bits[31:3] are Reserved.	RW
0x946	CNTR_RX_STATUS	• Bit[1]: Indicates that the RX statistics registers are paused (while CNTR_RX_CONFIG[2] is asserted). • Bit[0]: Indicates the presence of at least one parity error in the RX statistics counters. Bits [31:2] are Reserved.	RO
0x947–0x95F	Reserved		
0x960	RxPayloadOctetsOK_LO	Number of received payload bytes in frames with no FCS, undersized, oversized, or payload length errors. If VLAN detection is turned off for the RX MAC (bit [1] of the RXMAC_CONTROL register at offset 0x50A has the value of 1), the IP core counts the VLAN header bytes (4 bytes for VLAN and 8 bytes for stacked VLAN) as payload bytes. This register is compliant with the requirements for aOctetsReceivedOK in section 5.2.2.1.14 of the *IEEE Standard 802.3-2008*.	RO
0x961	RxPayloadOctetsOK_HI		RO
0x962	RxFrameOctetsOK_LO	Number of received bytes in frames with no FCS, undersized, oversized, or payload length errors. This register is compliant with the requirements for ifInOctets in RFC3635 (Managed Objects for Ethernet-like Interface Types) and RX etherStatsOctets in RFC2819 (Remote Network Monitoring Management Information Base (RMON)).	RO
0x963	RxFrameOctetsOK_HI		RO

7.6. 1588 PTP Registers

The 1588 PTP registers together with the 1588 PTP signals process and provide Precision Time Protocol (PTP) timestamp information as defined in the *IEEE 1588-2008 Precision Clock Synchronization Protocol for Networked Measurement and Control Systems Standard*. The 1588 PTP module provides you the support to implement the 1588 Precision Time Protocol in your design.

Table 28. TX 1588 PTP Registers

Addr	Name	Bit	Description	HW Reset Value	Access
0xA00	TXPTP_REVID	[31:0]	IP core revision ID.	0x0916_2016	RO
0xA01	TXPTP_SCRATCH	[31:0]	Scratch register available for testing.	32'b0	RW
0xA02	TXPTP_NAME_0	[31:0]	First 4 characters of IP core variation identifier string "25GETXPTPCSR"	0x3235_4745	RO
0xA03	TXPTP_NAME_1	[31:0]	Next 4 characters of IP core variation identifier string "25GETXPTPCSR"	0x5458_5054	RO

continued...

Addr	Name	Bit	Description	HW Reset Value	Access
0xA04	TXPTP_NAME_2	[31:0]	Final 4 characters of IP core variation identifier string"25GETXPTPCSR"	0x5043_5352	RO
0xA05	TX_PTP_CLK_PERIOD	[19:0]	clk_txmac clock period. Bits [19:16]: nanoseconds Bits [15:0]: fraction of nanosecond	0x28F5C	RW
0xA06–0xA0A	Reserved		Reserved	96'b0	RO
0xA0B	TX_PTP_ASYM_DELAY	[18:0]	Asymmetry adjustment as required for delay measurement. The IP core adds this value to the final delay. • Bit[18]: The value of 1 enables the feature and the value of 0 disables the feature. • Bit[17]: The value of 1 indicates subtraction and the value of 0 indicates addition. Depending on the value of this bit, the value in bits [16:0] is added to or subtracted from the final delay. • Bits[16:0]: Asymmetry adjustment in nanoseconds.	19'b0	RW
0xA0C	TX_PTP_PMA_LATENCY	[31:0]	Latency through the TX PMA. This is the delay from the TX PCS output to the tx_serial pin. • Bits [31:16]: Full nanoseconds • Bits [15:0]: Fraction of a nanosecond In Intel Arria 10 devices, the TX PMA latency is 187 UI. One UI is approximately 38.8 ps. Therefore, Intel recommends that you set this register to the value of 0x0007_428F. This is a device-dependent value that is sufficiently accurate in most cases. Intel recommends that you modify this value with extreme caution.	32'b0	RW

Table 29. RX 1588 PTP Registers

Addr	Name	Bit	Description	HW Reset Value	Access
0xB00	RXPTP_REVID	[31:0]	IP core revision ID.	0x0916_2016	RO
0xB01	RXPTP_SCRATCH	[31:0]	Scratch register available for testing.	32'b0	RW
0xB02	RXPTP_NAME_0	[31:0]	First 4 characters of IP core variation identifier string "25GERXPTPCSR"	0x3235_4745	RO
0xB03	RXPTP_NAME_1	[31:0]	Next 4 characters of IP core variation identifier string"25GERXPTPCSR"	0x5258_5054	RO
0xB04	RXPTP_NAME_2	[31:0]	Final 4 characters of IP core variation identifier string "25GERXPTPCSR"	0x5043_5352	RO
0xB05	RX_PTP_CLK_PERIOD	[19:0]	clk_rxmac clock period. Bits [19:16]: Full nanoseconds Bits [15:0]: Fraction of a nanosecond	0x28F5C	RW
0xB06	RX_PTP_PMA_LATENCY	[31:0]	Latency through the RX PMA. This is the delay from the rx_serial pin to the RX PCS input. • Bits [31:16]: Full nanoseconds • Bits [15:0]: Fraction of a nanosecond	32'b0	RW

continued...

Addr	Name	Bit	Description	HW Reset Value	Access
			In Intel Arria 10 devices, the RX PMA latency is 102.5 UI. One UI is approximately 38.8 ps. Therefore, Intel recommends that you set this register to the value of 0x0003_FA1C. This is a device-dependent value that is sufficiently accurate in most cases. Intel recommends that you modify this value with extreme caution.		

Related Information

- 1588 PTP Interface Signals on page 60
- 1588 Precision Time Protocol Interfaces on page 41
- PTP Transmit Functionality on page 46

7.7. TX Reed-Solomon FEC Registers

Table 30. TX Reed-Solomon FEC Registers

Addr	Name	Description	Reset	Access
0xC00	REVID	Reed-Solomon FEC TX module revision ID.	0x0916_2016	RO
0xC01	TX_RSFEC_NAME_0	First 4 characters of IP core variation identifier string, "25geRSFECoTX".	0x3235_6765	RO
0xC02	TX_RSFEC_NAME_1	Middle 4 characters of IP core variation identifier string, "25geRSFECoTX".	0x5253_4645	RO
0xC03	TX_RSFEC_NAME_2	Final 4 characters of IP core variation identifier string, "25geRSFECoTX".	0x436F_5458	RO
0xC04	ERR_INS_EN	Configuration register to enable error insertion in RS-FEC transmitter. Writing 1'b1 enables the feature. Writing 1'b0 disables it. The following encodings are defined: • Bit[4]: Enable error insertion for single FEC codeword. Bit self-clears after error is inserted. • Bit[0]: Enable error insertion for every FEC codeword. • All other bits: Reserved.	0x00000000	RW
0xC05	ERR_MASK	Specifies the bit masks for symbols and bits in a group for error injection. Each FEC codeword consists of 528 symbols of 10 bits each. The encoder works on groups of 8 symbols (80 bits). Therefore, each FEC codeword consists of 66 groups. Writing 1'b1 enables the feature. Writing 1'b0 disables it. The following encodings are defined: • Bits[25:16]: Bit mask. • Bits[15:8]: Symbol mask. • Bits[6:0]: Group number (0-65). • Other bits: Reserved.	0x00000000	RW
0xC06	BYPASS_RSFEC	Bypass RS-FEC core. Used by both TX and RX RS-FEC cores. Writing 1'b1 enables the feature. Writing 1'b0 disables it. The following encodings are defined: • Bit[0]: Bypass RS-FEC core. • All other bits: Reserved.	0x00000000	RW

7.8. RX Reed-Solomon FEC Registers

Table 31. RX Reed-Solomon FEC Registers

Addr	Name	Description	Reset	Access
0xD00	REVID	RS-FEC TX module revision ID	0x0916_2016	RO
0xD01	RX_RSFEC_NAME0	First 4 characters of IP core variation identifier string, "25geRSFECoRX".	0x3235_6765	RO
0xD02	RX_RSFEC_NAME1	Middle 4 characters of IP core variation identifier string, "25geRSFECoRX".	0x5253_4645	RO
0xD03	RX_RSFEC_NAME2	Final 4 characters of IP core variation identifier string, "25geRSFECoRX".	0x436F_5258	RO
0xD04	BYPASS_RESTART	Configuration register to bypass error correction and to restart alignment marker synchronization. Writing 1'b1 enables the feature. Writing 1'b0 disables it.The following encodings are defined: • Bit[0]: Bypass error correction. The RS-FEC core remains enabled but does not correct errors. • Bit[4]: Restarts FEC alignment marker synchronization. Bit clears after alignment marker synchronization is restarted. • All other bits: Reserved.	0x0000 0000	RW
0xD05	FEC_ALIGN_STATUS	Alignment marker lock status. The following encodings are defined: • Bit[0]: Indicates alignment marker lock status. When 1'b1, indicates alignment has been achieved. • All other bits: reserved	0x0000 0000	RO
0xD06	CORRECTED_CW	32-bit counter that contains the number of corrected FEC codewords processed. The value resets to zero upon read and holds at max count.	0x0000 0000	RO
0xD07	UNCORRECTED_CW	32-bit counter that contains the number of uncorrected FEC codewords processed. The value resets to zero upon read and holds at max count.	0x0000 0000	RO

8. Debugging the Link

Begin debugging your link at the most basic level, with word lock. Then, consider higher level issues.

The following steps should help you identify and resolve common problems that occur when bringing up a 25G Ethernet Intel FPGA IP core link:

1. Establish word lock—The RX lanes should be able to achieve word lock even in the presence of extreme bit error rates. If the IP core is unable to achieve word lock, check the transceiver clocking and data rate configuration. Check for cabling errors such as the reversal of the TX and RX lanes. Check the clock frequency monitors (KHZ_REF, KHZ_TX, KHZ_RX PHY registers) in the Control and Status registers.

   To check for word lock: Clear the FRM_ERR register by writing the value of 1 followed by another write of 0 to the SCLR_FRM_ERR register at offset 0x324.Then read the FRM_ERR register at offset 0x323. If the value is zero, the core has word lock. If non-zero the status is indeterminate
2. When having problems with word lock, check the EIO_FREQ_LOCK register at address 0x321. The values in this register define the status of the recovered clock. In normal operation, all the bits should be asserted. A non-asserted (value-0) or toggling logic value on the bit that corresponds to any lane, indicates a clock recovery problem. Clock recovery difficulties are typically caused by the following problems:
   - Bit errors
   - Failure to establish the link
   - Incorrect clock inputs to the IP core
3. Check the PMA FIFO levels by selecting appropriate bits in the EIO_FLAG_SEL register and reading the values in the EIO_FLAGS register. During normal operation, the TX and RX FIFOs should be nominally filled. Observing a the TX FIFO is either empty or full typically indicates a problem with clock frequencies. The RX FIFO should never be full, although an empty RX FIFO can be tolerated.
4. Establish lane integrity—When operating properly, the lanes should not experience bit errors at a rate greater than roughly one per hour per day. Bit errors within data packets are identified as FCS errors. Bit errors in control information, including IDLE frames, generally cause errors in XL/CGMII decoding.
5. Verify packet traffic—The Ethernet protocol includes automatic lane reordering so the higher levels should follow the PCS. If the PCS is locked, but higher level traffic is corrupted, there may be a problem with the remote transmitter virtual lane tags.
6. Tuning—You can adjust transceiver analog parameters to improve the bit error rate. If you turn on ADME in the IP core parameter editor, you can use the Transceiver Toolkit for this purpose.

In addition, your IP core can experience loss of signal on the Ethernet link after it is established. In this case, the TX functionality is unaffected, but the RX functionality is disrupted. The following symptoms indicate a loss of signal on the Ethernet link:

- The IP core deasserts the `rx_pcs_ready` signal, indicating the IP core has lost alignment marker lock.
- The IP core deasserts the RX PCS fully aligned status bit (bit [0]) of the RX_PCS_FULLY_ALIGNED_S register at offset 0x326. This change is linked to the change in value of the `rx_pcs_ready` signal.
- If **Enable link fault generation** is turned on, the IP core sets `local_fault_status` to the value of 1.
- The IP core asserts the `Local Fault Status` bit (bit [0]) of the `Link_Fault` register at offset 0x508. This change is linked to the change in value of the `local_fault_status` signal.
- The IP core triggers the RX digital reset process by asserting `soft_rxp_rst`.

Related Information

Intel Arria 10 Transceiver PHY User Guide
For information about the analog parameters for Intel Arria 10 devices.

8.1. Error Insertion Test and Debugging

Error insertion allows you to test 25G Ethernet Intel FPGA IP core test error handling.

To use this feature, the Avalon streaming TX client asserts `l1_tx_error` in the same cycle as `l1_tx_endofpacket`. The error appears as a 66-bit error block that consists of eight /E/ characters (EBLOCK_T) in the Ethernet frame. The 25G Ethernet Intel FPGA IP core overwrites Ethernet frame data with an EBLOCK_T error block when it transmits the Ethernet frame that corresponds to the packet EOP. The RX interface detects the frame corruption resulting in a CRC error output.

9. 25G Ethernet Intel Arria 10 FPGA IP User Guide Archive

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

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

Intel Quartus Prime Version	IP Version	User Guide
20.3	19.4.0	25G Ethernet Intel Arria 10 FPGA IP User Guide
19.4	19.4.0	25G Ethernet Intel Arria 10 FPGA IP User Guide
17.0	17.0	25G Ethernet Intel Arria 10 FPGA IP User Guide

10. Document Revision History for the 25G Ethernet Intel Arria 10 FPGA IP User Guide

Document Version	Intel Quartus Prime Version	IP Version	Changes
2021.03.29	21.1	19.4.0	Updated the descriptions for the following signals in Table: *Signals of the 1588 Precision Time Protocol Interface*: • `tx_etstamp_ins_ctrl_residence_time_calc_format` • `tx_egress_timestamp_64b_data[63:0]` • `tx_egress_timestamp_96b_fingerprint[(W-1):0]` • `tx_egress_timestamp_64b_fingerprint[(W-1):0]`
2020.10.12	20.3	19.4.0	• Updated the description in the *About the 25G Ethernet Intel FPGA IP* section. • Updated the description in the *25G Ethernet Intel FPGA IP Supported Features* section.25G Ethernet Intel FPGA IP Supported Features • Added a note to the *Length Checking* section to state that the MAC has a counter limit of 0xFFFF starting from Intel Quartus Prime Pro Edition software version 20.3 onward. • Made minor editorial updates throughout the document.
2020.06.22	19.4	19.4.0	• Updated the *Length/Type Field Processing* section. • Update the descriptions to the following signals in Table: *Avalon Streaming TX Datapath*: — `l1_tx_data[63:0]` — `l1_tx_valid` — `l1_tx_ready` • Removed Figure: *Client to 25G Ethernet Intel FPGA IP MAC Avalon Streaming Interface*. • Added the following Figures: — *Client to 25G Ethernet Intel FPGA IP MAC Avalon Streaming Interface when Ready Latency is 0 (1 of 2)* — *Client to 25G Ethernet Intel FPGA IP MAC Avalon Streaming Interface when Ready Latency is 0 (2 of 2)* — *Client to 25G Ethernet Intel FPGA IP MAC Avalon Streaming Interface when Ready Latency is 3 (1 of 2)* — *Client to 25G Ethernet Intel FPGA IP MAC Avalon Streaming Interface when Ready Latency is 3 (2 of 2)*

continued...

Document Version	Intel Quartus Prime Version	IP Version	Changes
2020.04.13	19.4	19.4.0	• Updated the *Simulating the IP Core* section. • Updated the *Length Checking* section. • Updated Table: Avalon Streaming RX Datapath to update the description for `l1_rx_error[5:0]`. • Updated Table: *Receive Side Statistics Registers* to update the descriptions for `CNTR_RX_1519toMAXB_HI`, `CNTR_RX_OVERSIZE_LO`, and `CNTR_RX_OVERSIZE_HI`. • Updated the description and reset value for `RXMAC_CONTROL` and description for `LINK_FAULT` in Table: *RX MAC Registers*.
2020.02.20	19.4	19.4.0	• Updated the description for frame monitoring and statistics in the *25G Ethernet Intel FPGA IP Core Supported Features* section. • Updated the *Debugging the Link* section.

continued...

Document Version	Intel Quartus Prime Version	IP Version	Changes
2019.12.16	19.4	19.4.0	• Updated the description in the *About the 25G Ethernet Intel FPGA IP Core* section. • Updated Table: *IP Core FPGA Resource Utilization for 25G Ethernet Intel FPGA IP Core for Intel Arria 10 Devices*. • Updated the description in the *PTP Transmit Functionality* section. • Updated the descriptions for the following signals in Table: *Signals of the 1588 Precision Time Protocol Interface*: — `tx_etstamp_ins_ctrl_offset_timestamp[15:0]` — `tx_etstamp_ins_ctrl_offset_correction_field[15:0]` — `tx_etstamp_ins_ctrl_offset_checksum_field[15:0]` — `tx_etstamp_ins_ctrl_offset_checksum_correction[15:0]` • Added `rx_am_lock` to Table: *Miscellaneous Status and Debug Signals*. • Updated the description and reset value for `RXMAC_CONTROL` in Table: *RX MAC Registers*. • Updated the description of the *Avalon Memory-Mapped Management Interface* section. • Updated the *Statistics Registers* section. • Renamed Altera Debug Master Endpoint (ADME) to Native PHY Debug Master Endpoint (NPDME). • Updated for latest Intel branding standards.
2018.10.23	17.0	17.0	• Updated Table: PHY Registers: — Added bit[1] description for `RX_PCS_FULLY_ALIGNED_S`. — Updated the descriptions for `KHZ_REF`, `KHZ_RX`, and `KHZ_TX`. • Made minor editorial updates to the document.
2018.07.17	17.0	17.0	• Renamed the document as *25G Ethernet Intel Arria 10 FPGA IP User Guide*. • Renamed "25G Ethernet" IP core to "25G Ethernet Intel FPGA IP" as per Intel rebranding. • Updated Table: *TX 1588 PTP Registers* to correct the HW reset value of the `TX_PTP_CLK_Period` register to 0x28F5C. • Updated Table: *RX 1588 PTP Registers* to correct the HW reset value of the `RX_PTP_CLK_Period` register to 0x28F5C. • Updated for latest Intel branding standards.

Date	Quartus Prime Version	Changes
2017.11.07	17.0	Added link to KDB Answer that provides workaround for potential jitter on Intel Arria 10 devices due to cascading ATX PLLs in the IP core. Refer to Handling Potential Jitter in Intel Arria 10 Devices on page 22.
2017.08.28	17.0	• Updated for new Quartus Prime software v17.0 release — Updated instructions to generate the IP core. Refer to Specifying the 25G Ethernet Intel FPGA IP Core Parameters and Options on page 15 — Updated IP Cores Generated Files figure. Refer to Generated File Structure on page 17. • Updated information about specification compliance. The IP core now complies with Draft 1.6 of the *25G & 50G Ethernet Specification* from the 25 Gigabit Ethernet Consortium, and the *IEEE 802.3by 25Gb Ethernet* specification is no longer a draft. Refer to About the 25G Ethernet Intel FPGA IP on page 4 and 25G Ethernet Intel FPGA IP Supported Features on page 7. • Added shadow feature for reading statistics registers. Replaced the `CLEAR_TX_STATS` and `CLEAR_RX_STATS` registers with new `CNTR_TX_CONFIG` and `CNTR_RX_CONFIG` registers at offsets 0x845 and 0x945. Added new `CNTR_TX_STATUS` and `CNTR_RX_STATUS` registers at offsets 0x846 and 0x946, respectively . Refer to Statistics Registers on page 77. • Removed statements that parameter editor fails to enforce certain incompatible parameter constraints. The IP core parameter editor no longer allows you to select unsupported parameter value combinations. Refer to 25G Ethernet Intel FPGA IP Parameters on page 25. • Removed Vendor ID from Release Information on page 6. • Clarified that the IP core `clk_ref` input signal and the ATX PLL reference clock input signal must be driven by the same clock. Refer to Adding the Transceiver PLL on page 20. • Clarified that the design example includes SDC files that you can modify for your own design. Refer to Compiling the Full Design and Programming the FPGA on page 24. • Clarified that the IP core does not support VHDL models. Refer to Specifying the 25G Ethernet Intel FPGA IP Core Parameters and Options on page 15 and Generated File Structure on page 17. • Clarified that the TX statistics counters do not count the errors you generate by asserting the `l1_tx_error` signal. Refer to TX MAC Interface to User Logic on page 53 and TX Statistics Registers on page 78. • Clarified that both read and write accesses to undefined registers or register fields, including accesses to registers and register fields not defined in the current IP core variation, return unspecified results. Refer to Control, Status, and Statistics Register Descriptions on page 67. • Removed the section "Creating a Signal Tap II Debug File to Match Your Design Hierarchy" from the Debugging chapter. The files cited (`build_stp.tcl`) do not exist in the file hierarchy generated by the Intel Quartus Prime Standard Edition v17.0 software. • Reworded the definitions of the TX flow control/PFC registers at offsets 0x606 and 0x641 for clarity. The register functionality has not changed. Refer to Pause/PFC Flow Control Registers on page 72. • Updated flow control register descriptions to use the variable FCQN as it is defined in TX MAC Interface to User Logic on page 53: the number of flow control queues in the IP core. Refer to Pause/PFC Flow Control Registers on page 72. • Updated REV_ID register values in Pause/PFC Flow Control Registers on page 72, 1588 PTP Registers on page 84,TX Reed-Solomon FEC Registers on page 86 and RX Reed-Solomon FEC Registers on page 87.

continued...

Date	Quartus Prime Version	Changes
		• Added clarification that non-character bytes in NAME registers are not printable, in register sections that previously indicated only the value zero for these bytes. Refer to PHY Registers on page 68 and Pause/PFC Flow Control Registers on page 72. • Mentioned that the IP core does not correctly flag received non-zero 4-bit ordered set types as erroneous. Refer to IP Core Malformed Packet Handling on page 33 and Link Fault Signaling Interface on page 36. • Added reference to Intel FPGA Answer that explains the requirement for certain additions to .sdc file. Refer to Compiling the Full Design and Programming the FPGA on page 24. • Reorganized Features list for reading clarity. Refer to 25G Ethernet Intel FPGA IP Supported Features on page 7. • Fixed assorted typos and minor errors.
2016.10.31	16.1	Initial release.