50 Gbps Ethernet IP Core User Guide

Updated for Intel® Quartus® Prime Design Suite: **17.0**

Online Version
Send Feedback

UG-20021

ID: 683158
Version: 2017.05.08

Contents

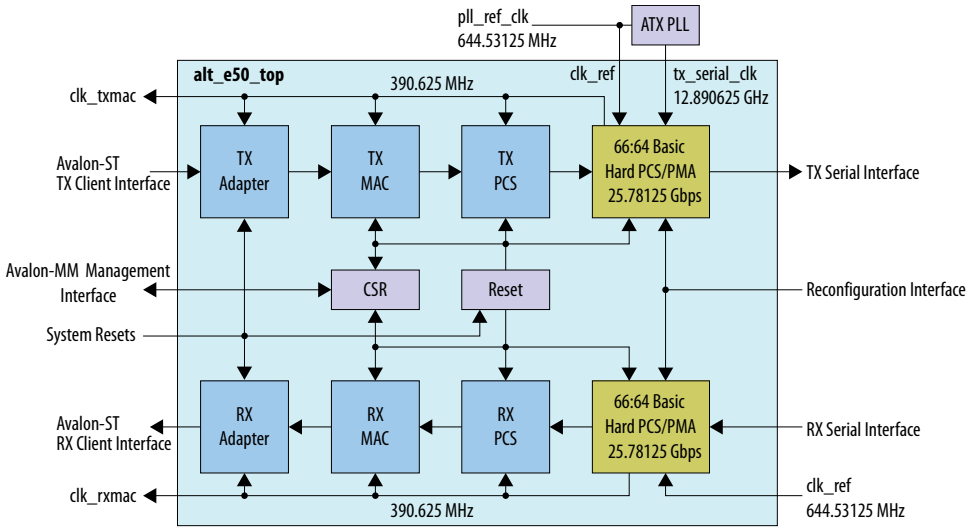
1. About the 50GbE Core

The 50GbE core 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 core 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 50GbE core is a 64-bit Avalon® streaming interface. It maps to one 25.78125 Gbps transceiver. The IP core 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 Intel® 50G Ethernet IP core 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 core 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 50G Ethernet IP core is a 128-bit Avalon Streaming (Avalon-ST) interface. It maps to two 25.78125 Gbps transceivers.

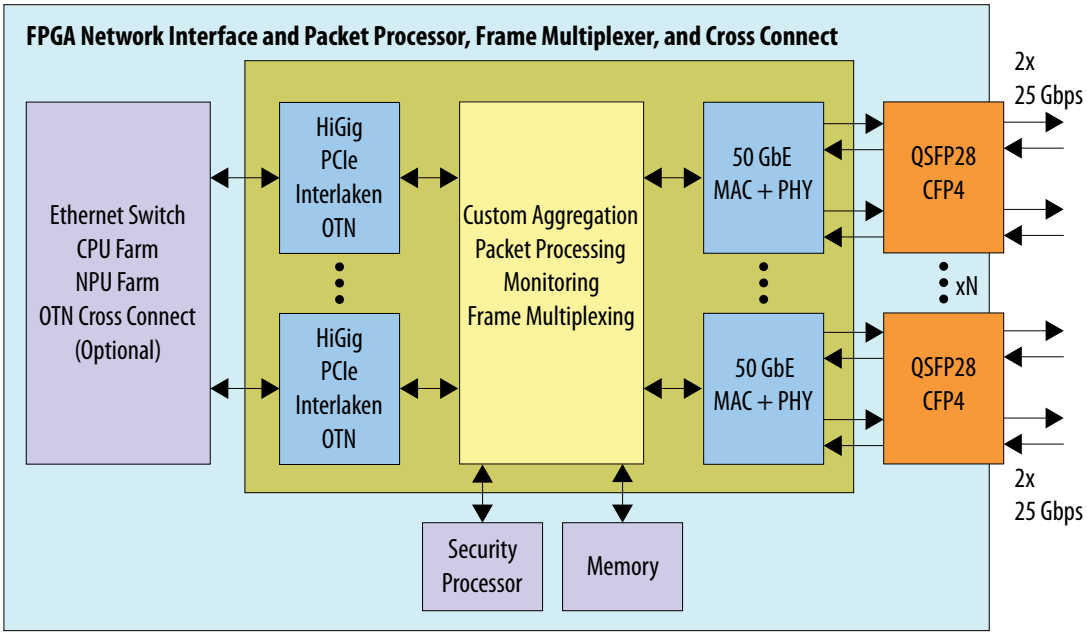
The IP core provides standard media access control (MAC) and physical coding sublayer (PCS), and PMA functions shown in the following block diagram. The PHY comprises the PCS and PMA.

Figure 1. 50GbE MAC and PHY IP Clock Diagram



The following block diagram shows an example of a network application with 50GbE MAC and PHY.

Figure 2. Example Network Application



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.
-

1.1. 50GbE Supported Features

The 50GbE 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.
- 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.
  - 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.
  - Optional RX strict SFD checking per IEEE specification.
  - RX malformed packet checking per IEEE specification.
  - Optional fault signaling detects and reports local fault and generates remote fault, with *IEEE 802.3ba-2012 Ethernet Standard Clause 66* support.
  - Unidirectional transport as defined in *Clause 66 of the IEEE 802.3-2012 Ethernet Standard*.
- Debug and testability features:
  - Programmable serial PMA local loopback (TX to RX) at the serial transceiver for self-diagnostic testing.
  - 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.
  - Ready latency of 0 clock cycles for Avalon-ST 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.

1.2. 50GbE Core Device Family and Speed Grade Support

1.2.1. Device Family Support

Table 1. 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 2. 25G Ethernet Intel FPGA IP50G Ethernet IP Core Device Family Support

Shows the level of support offered by the 50GbE 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 Quartus Prime Standard Edition software.
- Timing and Power Models
  Reports the default device support levels in the current version of the Quartus Prime Pro Edition software.

1.2.2. 50GbE Core Device Speed Grade Support

Table 3. Slowest Supported Device Speed Grades

IP Core	Device Family	Supported Speed Grades
50GbE	Intel Arria 10 GT	E2

1.3. IP Core Verification

To ensure functional correctness of the 50GbE core, Intel performs extensive validation through both simulation and hardware testing. Before releasing a version of the 50GbE 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

- 50G Ethernet IP Core Release Notes
  Changes to the IP core in major releases are noted in the 50GbE Release Notes starting from the Intel Quartus Prime software.
- Intel Quartus Prime Design Suite Update Release Notes
  Includes changes in minor releases (updates).

1.3.1. Simulation Environment

Intel performs the following tests on the 50GbE 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.

1.3.2. Compilation Checking

Intel performs compilation testing on an extensive set of 50GbE core variations and designs to ensure the Intel Quartus Prime software places and routes the IP core ports correctly.

1.3.3. Hardware Testing

Intel performs hardware testing of the key functions of the 50GbE core using internal loopback and also with other 50G switches. The hardware tests also ensure reliable solution coverage for hardware related areas such as performance, link synchronization, and reset recovery.

1.4. 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 4. 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

continued...

IP Core Variation / Parameter	A	B	C
Enable preamble passthrough	—	—	On
Enable TX CRC passthrough	On	—	—
Enable MAC statistics counters	—	On	On
Enable IEEE 1588	—	—	On

Table 5. IP Core FPGA Resource Utilization for 50GbE Core for Intel Arria 10 Devices

Lists the resources and expected performance for selected variations of the 50GbE 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

Fitter Resources Reports in the Quartus Prime Pro Edition Help

2. Getting Started

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

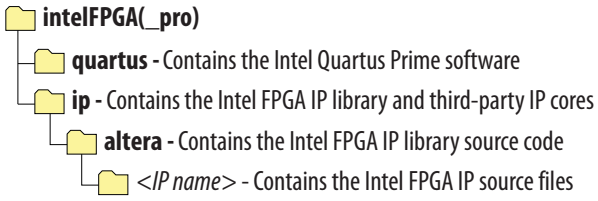


Table 6. IP Core Installation Locations

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.

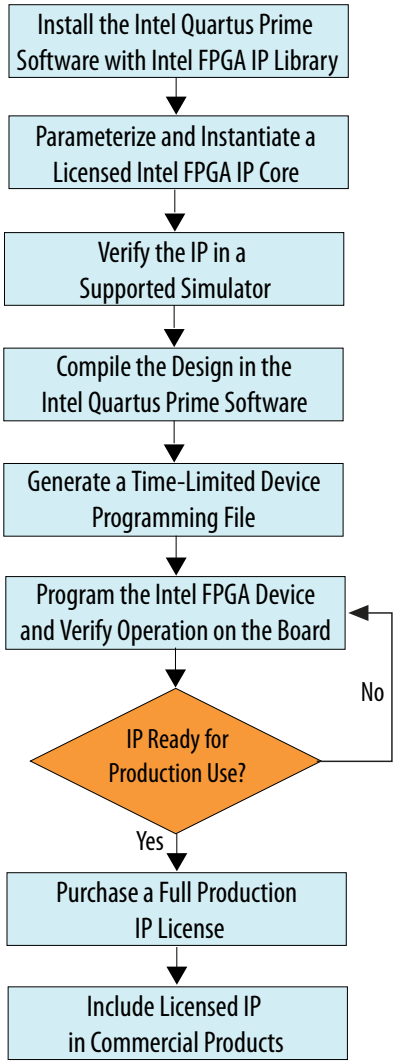
Intel FPGA IP Evaluation Mode supports the following operation modes:

- **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 Intel FPGA 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 50GbE Core Parameters and Options

The 50GbE 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 50GbE Parameters on page 21 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 50G 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 50G Ethernet Design Example User Guide
Information about the **Example Design** tab in the 50GbE parameter editor.

2.3. Simulating the IP Core

You can simulate your 50GbE 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.
- 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.

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 .

Figure 5. IP Core Generated Files

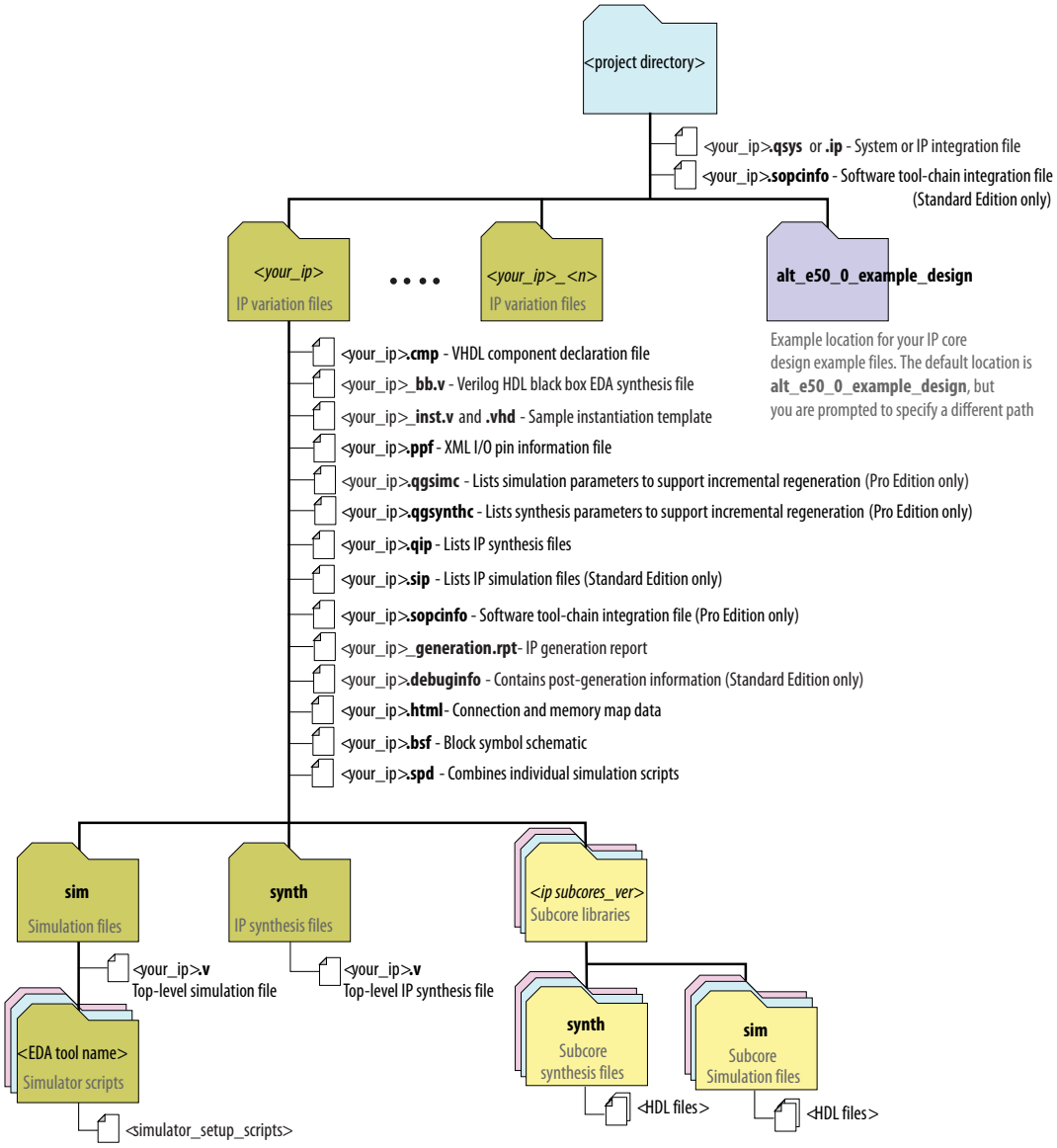


Table 7. IP Core Generated Files

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.

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.
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

50G Ethernet Design Example User Guide
Information about the 50GbE design example file structure.

2.5. Integrating Your IP Core in Your Design

2.5.1. Pin Assignments

When you integrate your 50GbE 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

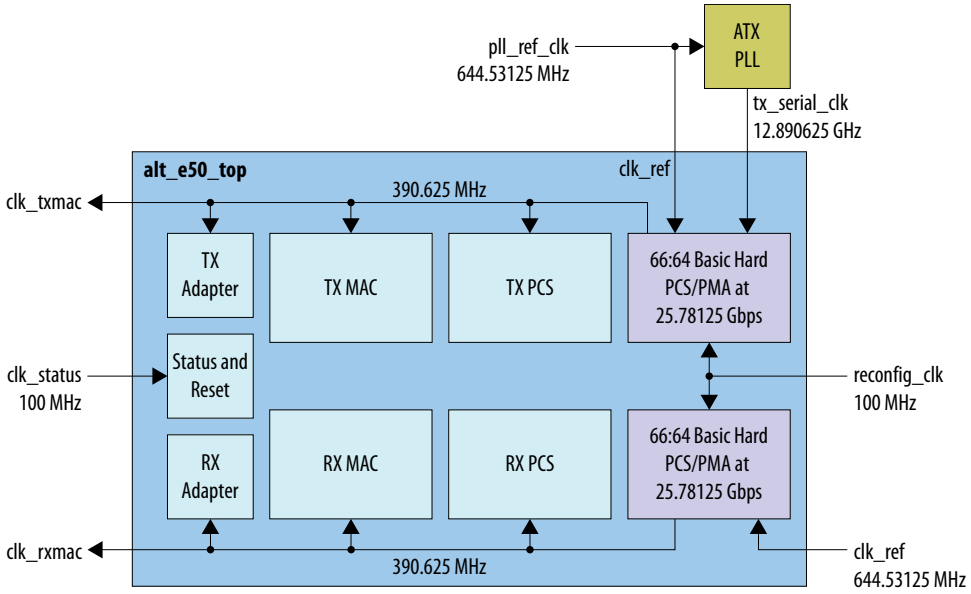
Intel Quartus Prime Help
For information about the Intel Quartus Prime software, including virtual pins.

2.5.2. Adding the Transceiver PLL

The 50GbE 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.

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 50GbE 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 50GbE core as follows:

- Connect the clock output port of the ATX PLL to the `tx_serial_clk` input pin for each 50GbE core PHY link to the same output port of the ATX PLL. Connect the ATX PLL output port to both `tx_serial_clk[1]` and `tx_serial_clk[0]`.
- Connect the `pll_locked` output port of the ATX PLL to the `tx_pll_locked` input port of the 50GbE core.
- Drive the ATX PLL reference clock port and the 50GbE 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 38
- 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 50GbE 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. Placement Settings for the 50GbE 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 50GbE 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
- 50G Ethernet Design Example User Guide
  Information about generating the design example and the design example directory structure.

- 25G Ethernet Intel Arria 10 FPGA IP Design Example User Guide
  Information about generating the design example and the design example directory structure.

3. 50GbE Parameters

The 50GbE parameter editor provides the parameters you can set to configure the 50GbE core and simulation testbenches.

The 50GbE parameter editor includes an **Example Design** tab. For information about that tab, refer to the *50G Ethernet Design Example User Guide*.

Table 8. IP Core Parameters

Parameter	Range	Default Setting	Description
General Options			
Device Family	**Arria 10**	**Arria 10**	Selects the device family.
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 transmit remote fault sequences in case of a local fault and can transmit IDLE control words in case of a remote fault.
Enable strict SFD checking	**Enabled**, **Disabled**	**Disabled**	When enabled, the IP core can implement strict SFD checking, depending on register settings.
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.
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 50GbE 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

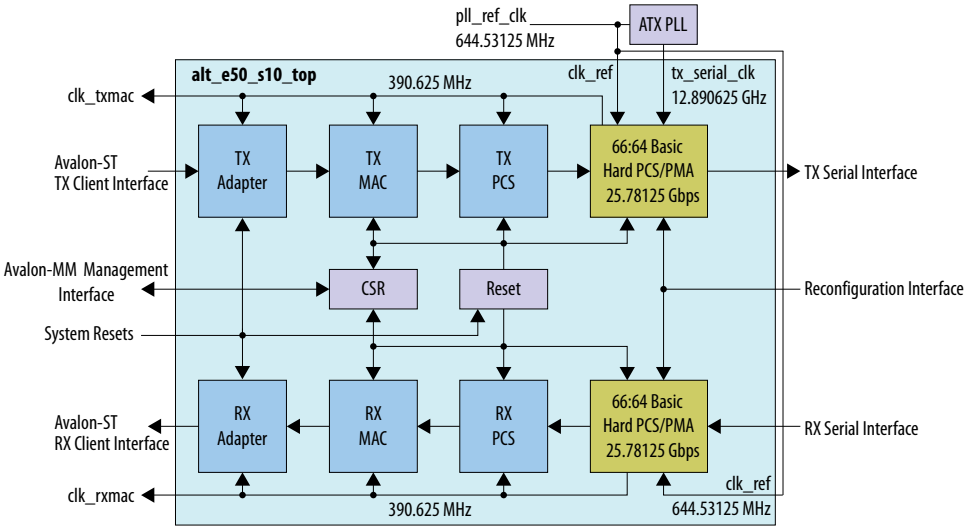
- 50G Ethernet Design Example User Guide
  Information about the **Example Design** tab in the 50GbE parameter editor.
- Intel Arria 10 Transceiver PHY User Guide
  Information about Intel Arria 10 Native PHY IP core features, including NPDME.

4. Functional Description

4.1. 50GbE Core Functional Description

The 50GbE 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 7. 50GbE 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.

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.
- 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. 50GbE 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. The MAC inserts the CRC bytes. The TX MAC module inserts IDLE bytes to maintain an average IPG of 12.

Figure 8. 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

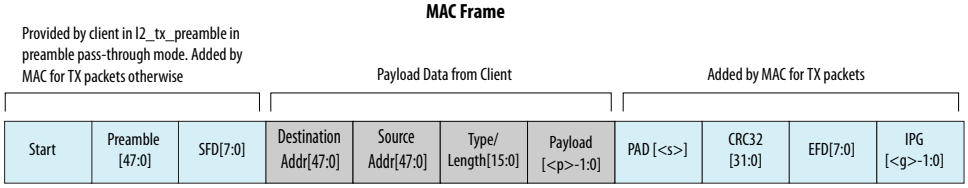
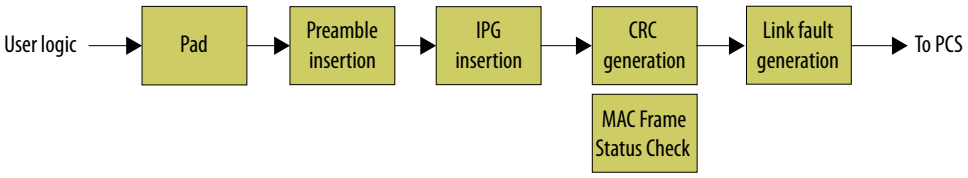


Figure 9. TX MAC Functions



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 (comprising seven bytes of preamble, and final 1-byte SFD) on a dedicated preamble bus, `l2_tx_preamble[63:0]`. In this case, 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.

When `l2_tx_startofpacket` is asserted, `l2_tx_preamble[63:0]` contains the preamble data and `l2_tx_data[127:0]` contains the first 16 bytes of frame data, starting from destination address.

Note that a single parameter in the 50GbE 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.

4.1.2. 50GbE 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 10. 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).

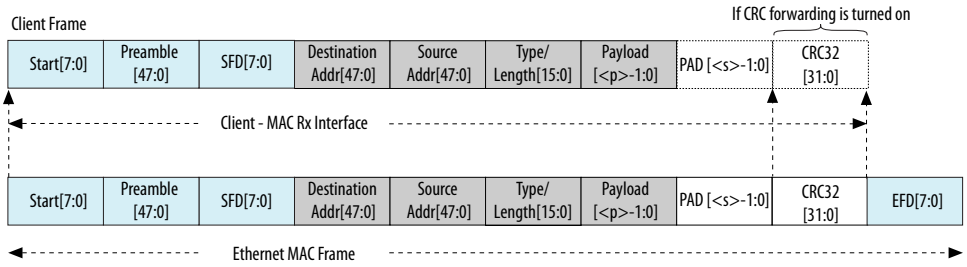


Figure 11. Flow of Client Frame With Preamble Pass-Through Turned Off

This figure uses the following notational conventions:

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

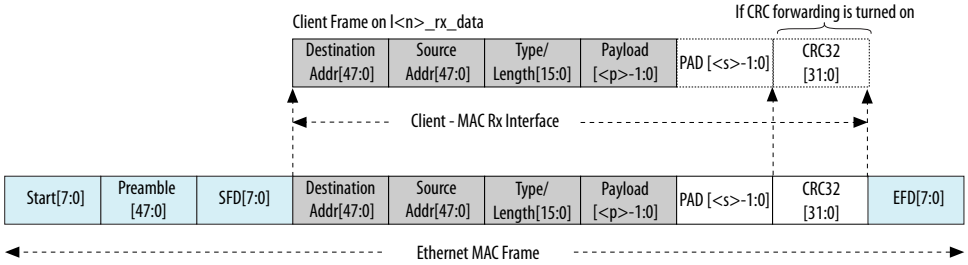
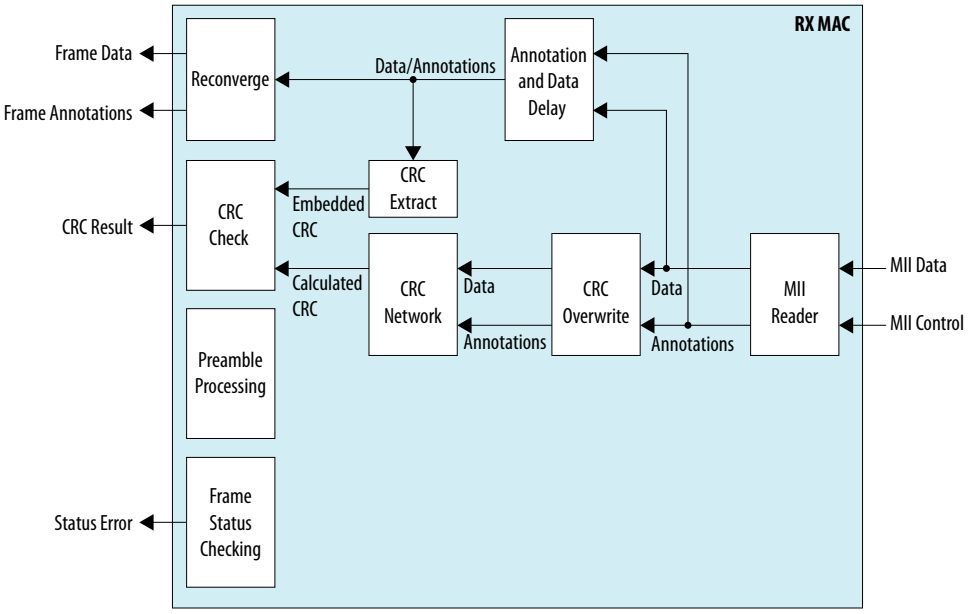


Figure 12. RX MAC Datapath



4.1.2.1. 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.

In preamble pass-through mode, `l2_rx_preamble[63:0]` provides the received preamble data to the client. When `l2_rx_startofpacket` is asserted, the `l2_rx_preamble[63:0]` drives the preamble data and `l2_rx_data[127:0]` drives the first 16 bytes of frame data, starting from destination address. The `l2_rx_preamble[63:0]` bus is only present in preamble pass-through mode.

If you turn off **Enable preamble passthrough**, the IP core removes the preamble bytes. `l2_rx_startofpacket` is aligned to the MSB of the destination address.

Note that a single parameter in the 50GbE parameter editor turns on both RX and TX preamble passthrough.

4.1.2.2. IP Core Strict SFD Checking

The 50GbE core RX MAC checks all incoming packets for a correct Start byte (0xFB). If you turn on **Enable strict SFD check** in the 50GbE parameter editor, you enable the RX MAC to check the incoming preamble and SFD for the following values:

- SFD = 0xD5
- Preamble = 0x555555555555

The RX MAC checks one or both of these values depending on the values in bits [4:3] of the RXMAC_CONTROL register at offset 0x50A.

Table 9. Strict SFD Checking Configuration

Enable strict SFD check	0x50A[4]: Preamble Check	0x50A[3]: SFD Check	Fields Checked	Behavior if Check Fails
Off	Don't Care	Don't Care	Start byte	IP core does not recognize a malformed Start byte as a Start byte
On	0	0	Start byte	
	0	1	Start byte and SFD	IP Core drops the packet
	1	0	Start byte and preamble	
	1	1	Start byte and preamble and SFD	

Related Information

RX MAC Registers on page 46
Describes the RXMAC_CONTROL register at offset 0x50A.

4.1.2.3. IP Core Malformed Packet Handling

While receiving an incoming packet from the Ethernet link, the 50GbE 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 50GbE 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 (`l2_rx_error[0]`).
- Asserts an FCS error (`l2_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, the IP core identifies it as a malformed packet.

4.1.2.4. 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. 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.

    Note: The 50G Ethernet IP core passes these frames to the RX client interface and updates the appropriate `l2_rxstatus_data` bits. However, the IP core does not implement flow control.
  - For other field values, the MAC RX forwards the receive frame to the client.

4.1.2.4.1. 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.
- 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.

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.

- `l2_rx_error[2]`—Undersized frame.
- `l2_rx_error[3]`—Oversized frame.
- `l2_rx_error[4]`—Payload length error.

If the length field value is greater than the actual payload length, the IP core asserts `l2_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 `l2_rx_error[4]`.

4.1.2.5. RX CRC Checking and Dynamic Forwarding

The RX MAC checks the incoming CRC32 for errors. It asserts `l2_rx_error[1]` in the same cycle as `l2_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.

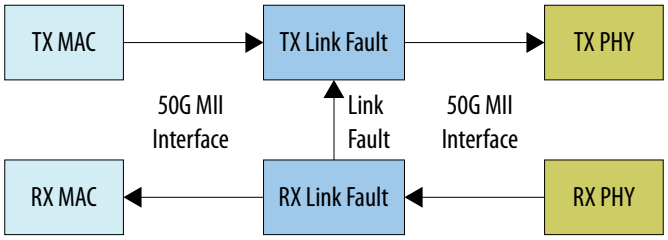
By default, the RX MAC strips off the CRC bytes before forwarding the packet to the MAC client. You can configure the core to retain the RX CRC and forward it to the client by updating the `MAC_CRC_CONFIG` register.

4.1.3. 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 40G/100G Ethernet Standard* and *Ethernet Clause 81.3.4* 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 13. 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 10. 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

continued...

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	
								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.

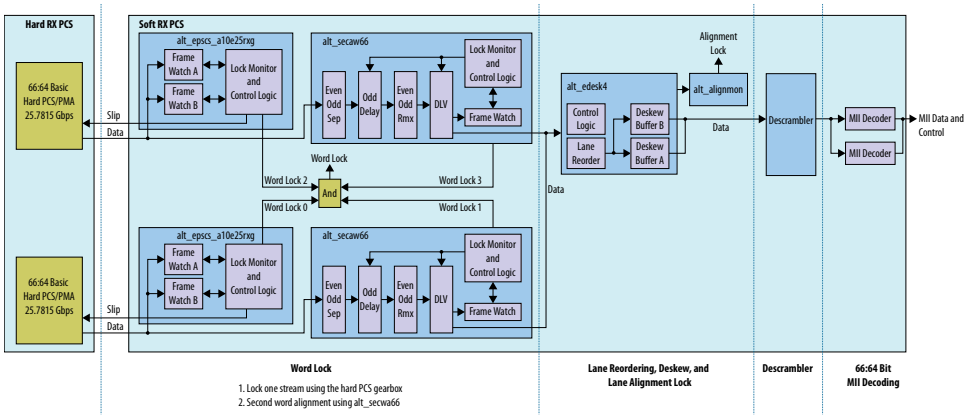
Related Information

- TX MAC Registers on page 45
  Information about the LINK_FAULT register.
- IEEE website
  The Ethernet specifications are available on the IEEE website.

4.1.4. 50 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 two, 66-bit output streams containing four virtual lanes to the soft RX PCS. The soft RX PCS implements word lock, lane reordering, descrambling, and MII decoding.

Figure 14. High Level Block Diagram of the Soft TX PCS



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 15. 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.

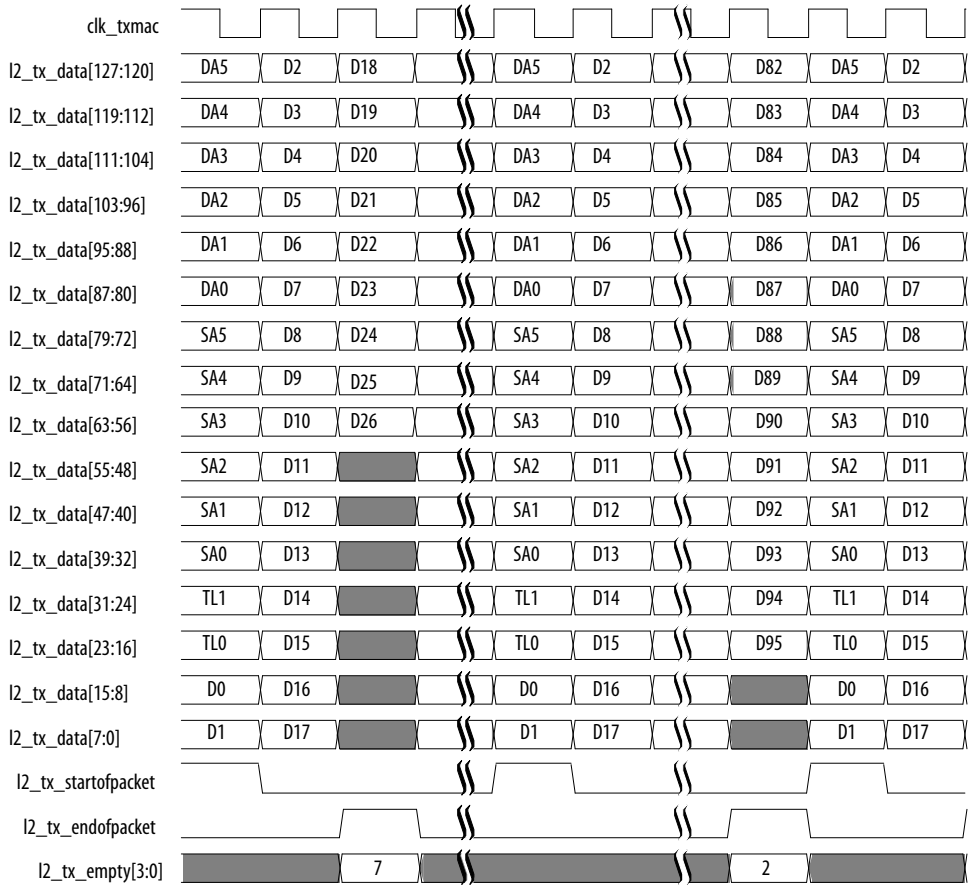
	Destination Address (DA)						Source Address (SA)						Type/ Length (TL)		Data (D)		
Octet	5	4	3	2	1	0	5	4	3	2	1	0	1	0	00	...	NN
Bit	[47:40]	[39:32]	[31:24]	[23:16]	[15:8]	[7:0]	[47:40]	[39:32]	[31:24]	[23:16]	[15:8]	[7:0]	[15:8]	[7:0]	MSB[7 :0]	...	LSB[7:0]

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 16. Octet Transmission on the 50GbE Avalon Streaming Interface

In the following diagram Preamble pass through is 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

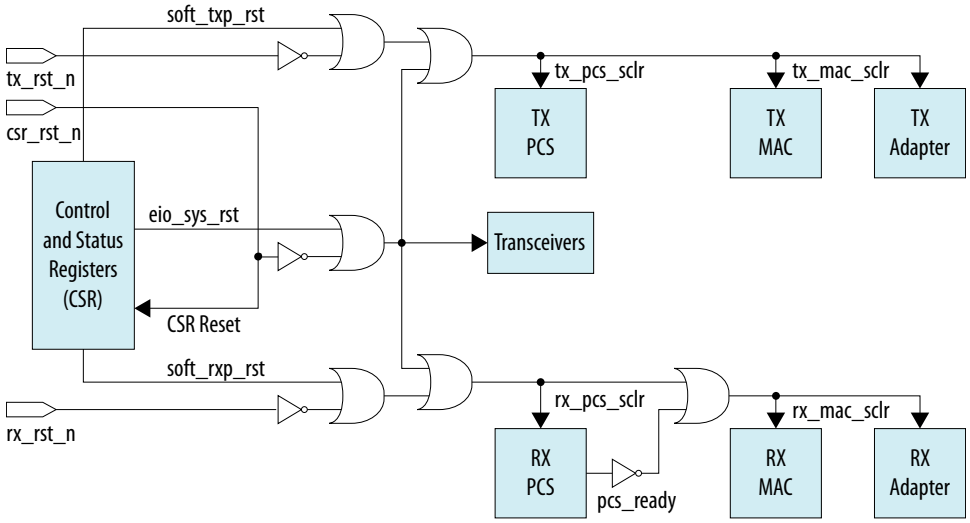
The *IEEE Standard for Ethernet, Section 4* is available on the IEEE website.

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 17. 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:

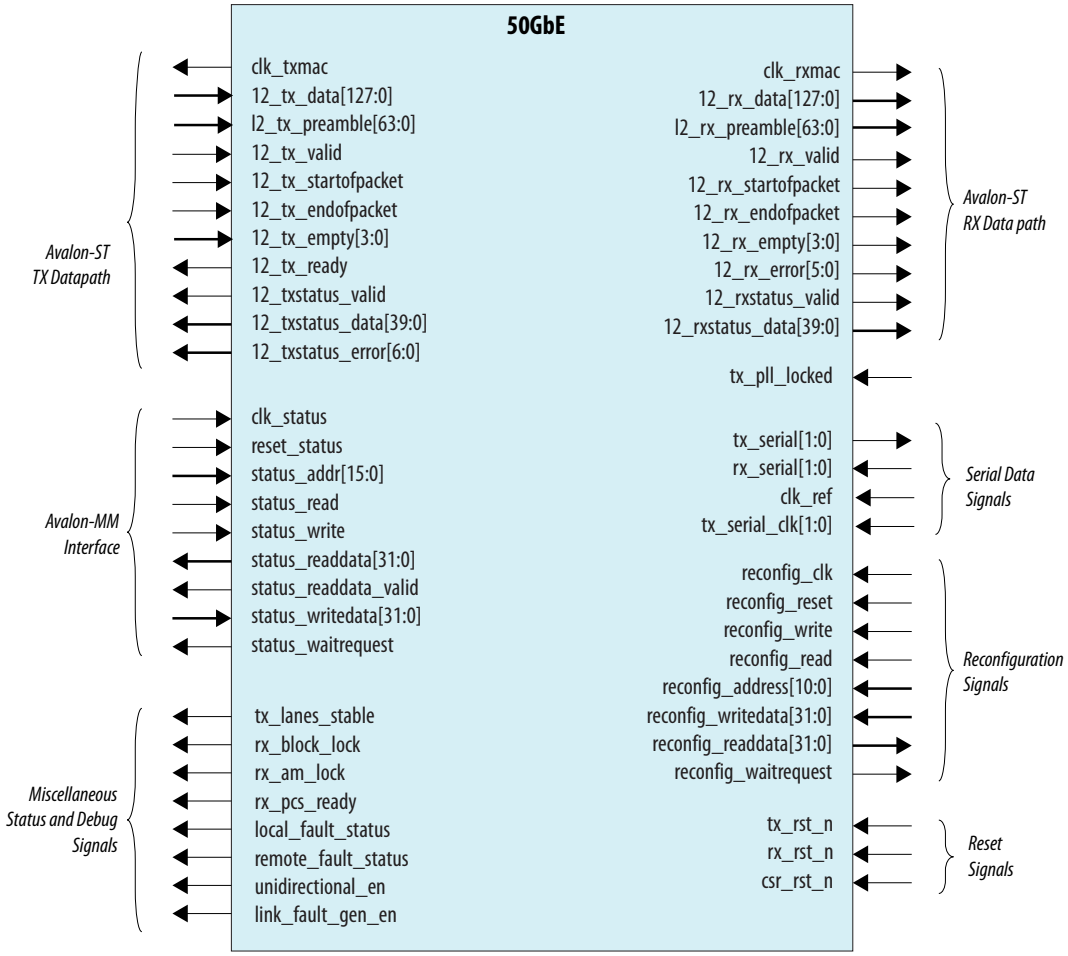
- `soft_txp_rst`: Resets the IP core in TX direction. Resets the TX PCS, MAC, and adapter.This soft reset leads to deassertion of `tx_lanes_stable` output signal.
- `soft_rxp_rst`: Resets the IP core in RX direction. Resets the RX PCS, MAC, and adapter. This soft reset leads to the deassertion of `rx_pcs_ready` 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

- PHY Registers on page 42
- Reset Signals on page 41

6. Interfaces and Signal Descriptions

Figure 18. 50GbE 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 11. Avalon Streaming TX Datapath

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

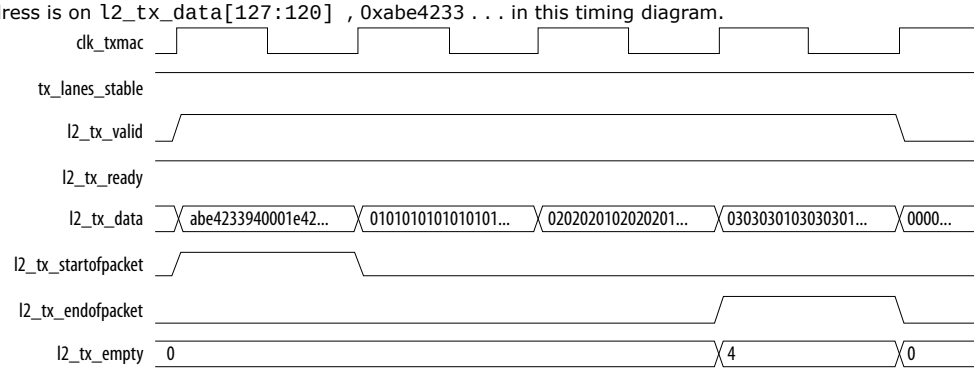
Signal	Direction	Description
`clk_txmac`	Output	Clock for the TX logic. Derived from `pll_refclk`, and is an output from the 50GbE core. `clk_txmac` is guaranteed to be stable when `tx_lanes_stable` is asserted. The frequency of this clock is 390.625 MHz.All TX MAC interface signals are synchronous to `clk_txmac`.
`l2_tx_data[127:0]`	Input	Data Input to MAC. Bit 127 is the MSB and bit 0 is the LSB. Bytes are read in the usual left to right order. The 50GbE 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 `l2_tx_startofpacket` when you are assured the packet data to send on `l2_tx_data[127:0]` is available or will be available on time.
`l2_tx_preamble[63:0]`	Input	User preamble data. Available when you select PREAMBLE PASS-THROUGH mode. User logic drives the custom preamble data when `l2_tx_startofpacket` is asserted.
`l2_tx_valid`	Input	When asserted, indicates valid data is available on `l2_tx_data[127:0]`. You must assert this signal continuously between the assertions of `l2_tx_startofpacket` and `l2_tx_endofpacket` for the same packet.
`l2_tx_startofpacket`	Input	When asserted, indicates the first byte of a frame. When `l2_tx_startofpacket` is asserted, the MSB of `l2_tx_data` drives the first byte of the packet.
`l2_tx_endofpacket`	Input	When asserted, indicates the end of a packet. The IP core ignores packets with length less than nine bytes.
`l2_tx_empty[3:0]`		Specifies the number of empty bytes on `l2_tx_data` when `l2_tx_endofpacket` is asserted.
`l2_tx_ready`	Output	When asserted, indicates that the MAC can accept the data.
`l2_txstatus_valid`	Output	When asserted, indicates that `l2_txstatus_error[6:0]` is driving valid data.
`l2_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

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
l2_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

Figure 19. Client to 50GbE MAC Avalon Streaming Interface

The IP core expects data order in l2_tx_data is highest byte to lowest byte. The first byte of the destination address is on l2_tx_data[127:120] , 0xabe4233 . . . in this timing diagram.



Related Information

Avalon Interface Specifications
Detailed information about Avalon streaming interfaces.

6.2. RX MAC Interface to User Logic

The RX MAC provides an Avalon streaming interface to the FPGA fabric. The datapath comprises 2, 64-bit words.

Table 12. Avalon Streaming RX Datapath

All interface signals are clocked by the clk_rxmac clock.

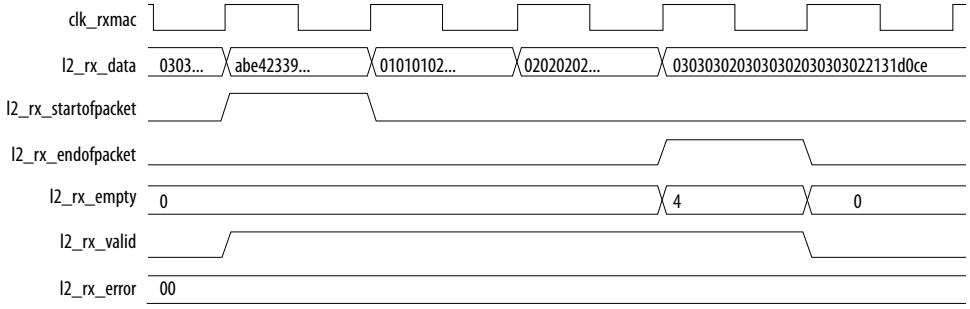
Signal	Direction	Description
clk_rxmac	Output	Clock for the RX MAC. Recovered from the incoming data. This clock is guaranteed stable when rx_pcs_ready is asserted. The frequency of this clock is 390.625 MHz. All RX MAC interface signals are synchronous to clk_rxmac.
l2_rx_data[127:0]	Output	Data output from the MAC. Bit 127 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.
l2_rx_preamble[63:0]	Output	Received preamble data. Available when you select PREAMBLE PASS-THROUGH mode. Valid when l2_rx_startofpacket is asserted.

continued...

Signal	Direction	Description
l2_rx_valid	Output	When asserted, indicates that l2_rx_data[127:0] is driving data. The IP core need not assert this signal continuously between the assertions of l2_tx_startofpacket and l2_tx_endofpacket for the same packet. During alignment marker cycles, the IP core drives IDLE cycles.
l2_rx_startofpacket	Output	When asserted, indicates the first byte of a frame. In PREAMBLE PASS-THROUGH mode, marks the first byte of the preamble.
l2_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.
l2_rx_empty[3:0]	Output	Specifies the number of empty bytes when l2_rx_endofpacket is asserted. The packet can end at any byte position. The empty bytes are the low-order bytes.
l2_rx_error[5:0]	Output	When asserted in the same cycle as l2_rx_endofpacket, indicates the current packet should be treated as an error packet. The 6 bits of l2_rx_error specify the following errors: • l2_rx_error[5]: Unused. • l2_rx_error[4]: Payload length error. If the length field is <1535 bytes, the received payload length is less than what is advertised in payload length field. • l2_rx_error[3]: Oversized frame. The frame size is greater than the value specified in the RXMAC_SIZE_CONFIG register. • l2_rx_error[2]: Undersized frame – The frame size is less than 64 bytes. Frame size = header size + payload size. • l2_rx_error[1]: CRC Error. The computed CRC value differs from the received CRC. • l2_rx_error[0]: Malformed packet. The packet is terminated with a non-terminate control character. When this bit is asserted, l2_rx_error[1] is also asserted.
l2_rxstatus_valid	Output	When asserted, indicates that l2_rxstatus_data is driving valid data.
l2_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 • 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

Figure 20. 50GbE MAC to Client Avalon Streaming Interface

`l2_rx_data` reception order is highest byte to lowest byte. The first byte of the destination address is on `l2_rx_data[127:120]` , 0xabe42339 . . . 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. For the 50GbE core, you can use the same ATX PLL for both transceivers. 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 13. Transceiver Signals

Signal	Direction	Description
`tx_serial[1:0]`	Output	TX transceiver signal. Each `tx_serial` bit becomes two physical pins that form a differential pair.
`rx_serial[1:0]`	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[1:0]`	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 17
- 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 14. 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 and status registers except the statistics registers; while this reset is in process, the Avalon memory-mapped management interface does not respond.

Table 15. 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 and Status Register Descriptions on page 42
  Information about the 50GbE IP core registers you can access through the Avalon-MM 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. Miscellaneous Status and Debug Signals

The miscellaneous status and debug signals are asynchronous.

Table 16. 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	Asserted when all lanes have identified 66-bit block boundaries in the serial data stream.

continued...

Signal	Direction	Description
rx_am_lock	Output	Asserted when all lanes have identified alignment markers in the data stream.
rx_pcs_ready	Output	Asserted when the RX lanes are fully aligned and ready to receive data.
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.
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.

6.7. 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 17. 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 and status registers. This reset leads to the deassertion of the tx_lanes_stable and rx_pcs_ready output signals.

7. Control and Status 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 18. Register Base Addresses

Word Offset	Register Type
0x300-0x3FF	PHY registers
0x400-0x4FF	TX MAC registers
0x500-0x5FF	RX MAC registers

Related Information

Avalon Memory-Mapped Management Interface on page 40
Interface to access the 50GbE IP core registers.

7.1. PHY Registers

Table 19. 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, "0050". The "00" is unprintable.	0x0000 3530	RO
0x303	PHY_NAME_1	Next characters of IP core variation identifier string, "00GE". The "00" is unprintable.	0x0000 4745	RO

continued...

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

(2) Register value convert in decimal.

Addr	Name	Description	Reset	Access
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'b 0 (1)	RW
0x312	WORD_LOCK	When asserted, indicates that the virtual channel has identified 66 bit block boundaries in the serial data stream.	30'hX4'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.	30'hX2'b00 (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 • 3'b011: TX FIFO partially empty • 3b100: RX FIFO full • 3b101: RX FIFO empty • 3b110: RX FIFO partially full • 3b111: RX FIFO partially empty	29'hX3'b0 (1)	RW
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].	30'hX2'b0 (1)	RO

continued...

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

(2) Register value convert in decimal.

Addr	Name	Description	Reset	Access
0x321	EIO_FREQ_LOCK	Each asserted bit indicates that the corresponding lane RX clock data recovery (CDR) phase-locked loop (PLL) is locked.	30'hX2'b00 (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	Each asserted bit indicates that the corresponding virtual lane has 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. These bits are 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.	28'hX_4'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	Set bit [0] to clear the RX FIFO for both physical lanes.	31'bX_1'b0 (1)	RW
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
0x328	AM_LOCK	When asserted indicates that the physical channel has identified virtual lane alignment markers in the data stream.	28'hX_4'b0 (1)	RO
0x329	LANE_DESKEWED	The following encodings are defined: • 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.	30'hX2'b00 (1)	RO
0x330	PCS_VLANE	The following encodings are defined:	24'bX8'b0 (1)	RO

continued...

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

(2) Register value convert in decimal.

Addr	Name	Description	Reset	Access
		• Bit [1:0]: First virtual index for physical lane 0. • Bit [3:2]: Second virtual index for physical lane 0. • Bit [5:4]: First virtual index for physical lane 1. • Bit [7:6]: Second virtual index for physical lane 1.		
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 20. TX MAC Registers

Addr	Name	Description	Reset	Access
0x400	TXMAC_REVID	TX MAC revision ID for 50G 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, "50gMACTxCSR".	0x3530 674D	RO
0x403	TXMAC_NAME_1	Next 4 characters of IP core variation identifier string, "ACTx".	0x4143 5278 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
0x406	IPG_COL_REM	Specifies the number of IDLE columns to be removed in every Alignment Marker period to compensate for alignment marker insertion. You can program this register to a larger value than the default value, for clock compensation. Bits [31:8] of this register are Reserved.	0xXXXX 0004 (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 21. RX MAC Registers

Addr	Name	Description	Reset	Access
0x500	RXMAC_REVID	RX MAC revision ID for 50G 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, "50gMACRxCSR".	0x3530 674D	RO

continued...

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

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

Addr	Name	Description	Reset	Access
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
0x506	RXMAC_SIZE_CONFIG	Specifies the maximum frame length available. The MAC asserts `l2_rx_error[3]` when the length of the received frame exceeds the value of this register.	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 BA Ethernet Clause 81.3.4.* For unidirectional Link Fault, implements *IEEE 802.3 Ethernet Clause 66.*	30'hX2'b00 (4)	RO
0x50A	RXMAC_CONTROL	RX MAC Control Register. The following bits are defined: • Bit[4]: Preamble check. Strict SFD checking option to compare each packet preamble to 0x555555555555. This field is available only if you turn on **Enable strict SFD check**. • Bit[3]: SFD check. Strict SFD checking option to compare each SFD byte to 0x5D. This field is available only if you turn on **Enable strict SFD check**. • Bit [1]: VLAN detection disabled. This bit is deasserted by default implying VLAN detection is enabled.	27'h0_5'b11X0X (4)3 0'h0_2'b0X (4)	RW

Related Information

IP Core Strict SFD Checking on page 26

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

8. Debugging the Link

The following steps should help you identify and resolve common problems that occur when bringing up a 50GbE 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 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 deskew status by reading bit[0] of the LANE_DESKEWED register at address 0x329. Bit[1] of the LANE_DESKEWED register indicates if the deskew lock has ever toggled since reset or SCLR_FRM_ERR cleared an error. 50GbE currently supports up to 16 words of skew among the virtual lanes. The inability to deskew implies a higher than supported skew between the virtual lanes.
5. Establish the alignment marker lock—Virtual lane alignment marker lock requires a moderate quality transceiver connection. If the lock is completely absent, recheck the alignment marker period. If the lock is intermittent, recheck the transceiver physical connection and analog settings.

   To check for alignment marker lock: Check the value of the rx_pcs_ready signal or read bit [0] of the RX_PCS_FULLY_ALIGNED_S register at offset 0x326. The value of 1 on the signal or in the register bit indicates the RX PCS is fully aligned.

6. 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. The bit interleaved parity (BIP) mechanism is a diagonal parity computation that enables tracing a protocol error back to a physical lane.
7. 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.
8. Tuning—You can adjust transceiver analog parameters to improve the bit error rate. If you turn on NPDME 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 0x308. 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`.

9. Document Revision History

Date	Quartus Prime Version	Changes
2017.11.09	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 19 and to Compiling the Full Design and Programming the FPGA on page 19. Reorganized Features list for reading clarity. Refer to 50GbE Supported Features on page 6. 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 and Status Register Descriptions on page 42. Clarified that the `reset_status` signal remains visible in future releases for backward compatiblity, rather than being removed. The design must tie this input signal low. Refer to Avalon Memory-Mapped Management Interface on page 40.
2017.05.08	17.0	Initial public release.