



50 Gbps Ethernet IP Core User Guide

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



[Subscribe](#)

[Send Feedback](#)

UG-20021 | 2017.11.07

Latest document on the web: [PDF](#) | [HTML](#)

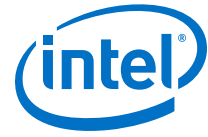


Contents

- 1 Datasheet..... 4**
 - 1.1 50GbE IP Core Supported Features..... 6
 - 1.2 50GbE IP Core Device Family and Speed Grade Support.....7
 - 1.2.1 Device Family Support.....7
 - 1.2.2 50GbE IP Core Device Speed Grade Support.....7
 - 1.3 IP Core Verification..... 7
 - 1.3.1 Simulation Environment.....8
 - 1.3.2 Compilation Checking.....8
 - 1.3.3 Hardware Testing.....8
 - 1.4 Performance and Resource Utilization..... 8
 - 1.5 Release Information.....9
- 2 Getting Started..... 10**
 - 2.1 Installing and Licensing IP Cores.....10
 - 2.1.1 OpenCore Plus IP Evaluation.....10
 - 2.2 Specifying the 50GbE IP Core Parameters and Options..... 11
 - 2.3 Simulating the IP Core.....12
 - 2.4 Generated File Structure..... 13
 - 2.5 Integrating Your IP Core in Your Design..... 15
 - 2.5.1 Pin Assignments..... 15
 - 2.5.2 Adding the Transceiver PLL15
 - 2.5.3 Handling Potential Jitter in Arria 10 Devices.....17
 - 2.5.4 Placement Settings for the 50GbE IP Core.....17
 - 2.6 Compiling the Full Design and Programming the FPGA..... 17
- 3 50GbE IP Core Parameters..... 19**
- 4 Functional Description..... 20**
 - 4.1 50GbE IP Core Functional Description..... 20
 - 4.1.1 50GbE IP Core TX Datapath.....21
 - 4.1.2 50GbE IP Core RX Datapath..... 22
 - 4.1.3 Link Fault Signaling Interface.....26
 - 4.1.4 50 GbE RX PCS.....28
 - 4.2 User Interface to Ethernet Transmission..... 28
 - 4.2.1 Order of Transmission.....29
 - 4.2.2 Bit Order For TX and RX Datapaths.....30
- 5 Reset..... 31**
- 6 Interfaces and Signal Descriptions..... 32**
 - 6.1 TX MAC Interface to User Logic.....32
 - 6.2 RX MAC Interface to User Logic.....34
 - 6.3 Transceivers.....36
 - 6.4 Transceiver Reconfiguration Signals.....36
 - 6.5 Avalon-MM Management Interface.....38
 - 6.6 Miscellaneous Status and Debug Signals.....38
 - 6.7 Reset Signals..... 39



| | |
|--|-----------|
| 7 Control and Status Register Descriptions..... | 40 |
| 7.1 PHY Registers..... | 40 |
| 7.2 TX MAC Registers..... | 43 |
| 7.3 RX MAC Registers..... | 44 |
| 8 Debugging the Link..... | 46 |
| 9 Document Revision History..... | 48 |

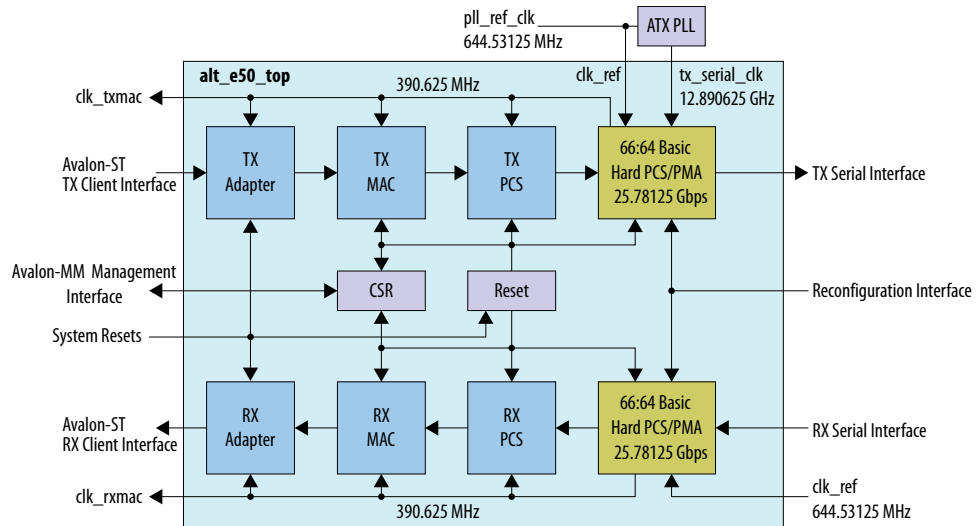


1 Datasheet

The Intel® 50 Gbps Ethernet (50GbE) IP core implements the *25G & 50G Ethernet Specification, Draft 1.4* from the 25 Gigabit Ethernet Consortium and the *IEEE 802.3by 25Gb Ethernet* draft. 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 50 Gbps Ethernet IP core is a 128-bit Avalon® Streaming (Avalon-ST) interface. It maps to two 25.78125 Gbps transceivers.

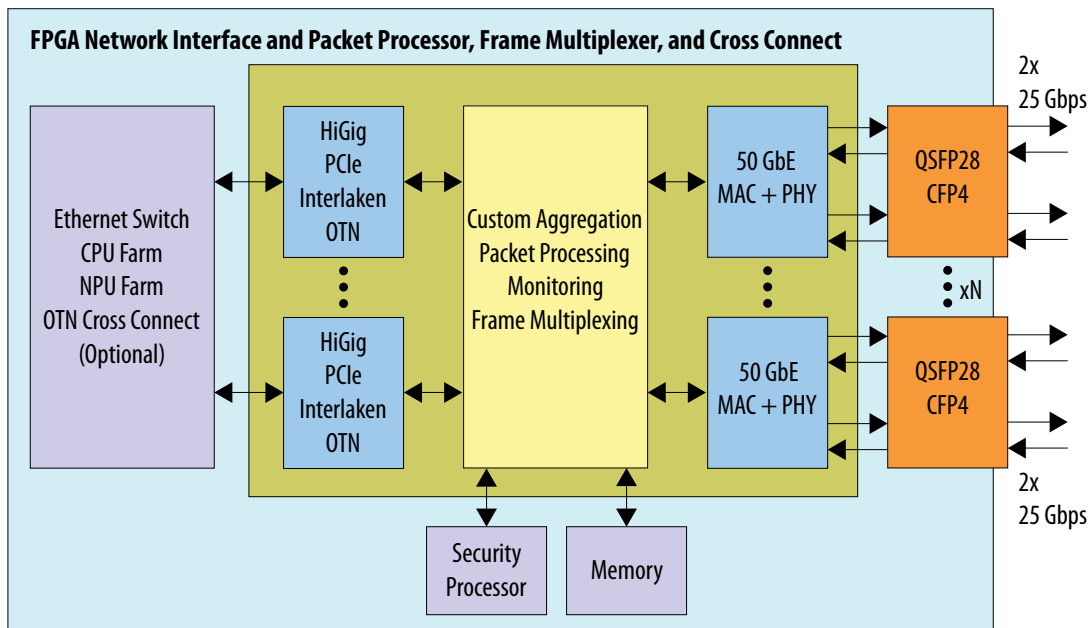
The IP core provides standard media access control (MAC) and physical coding sublayer (PCS), and PMA functions shown in the following block diagram. The PHY comprises the PCS and PMA.

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 Links

- [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.
- [25 Gigabit Ethernet Consortium](#)



1.1 50GbE IP Core Supported Features

The 50GbE IP core 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 core supports the following features:

- PHY features:
 - Soft PCS logic that interfaces seamlessly to 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 Altera Debug Master Endpoint (ADME) for serial link debugging or monitoring PHY signal integrity.
- User system interfaces:
 - Avalon Memory-Mapped (Avalon-MM) management interface to access the IP core control and status registers.
 - Avalon Streaming (Avalon-ST) 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 Links

[IEEE website](#)

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



1.2 50GbE IP 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. 50GbE IP Core Device Family Support

Shows the level of support offered by the 50GbE IP core for each Intel device family.

| Device Family | Support |
|-----------------------|--|
| Arria 10 GT | Default support level provided in the Quartus® Prime software. Refer to the <i>Quartus Prime Standard Edition Software and Device Support Release Notes</i> and the <i>Quartus Prime Pro Edition Software and Device Support Release Notes</i> . |
| Other device families | Not supported |

Related Links

- [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 IP Core Device Speed Grade Support

Table 3. Slowest Supported Device Speed Grades

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

1.3 IP Core Verification

To ensure functional correctness of the 50GbE IP core, Intel performs extensive validation through both simulation and hardware testing. Before releasing a version of the 50GbE IP core, Intel runs comprehensive regression tests in the current version of the Quartus Prime software.

Intel verifies that the current version of the 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 Links

- [Intel FPGA IP Release Notes](#)
Changes to the IP core in major releases are noted in the 50GbE Intel FPGA IP Release Notes starting from the Quartus Prime software v17.0
- [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 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

1.3.2 Compilation Checking

Intel performs compilation testing on an extensive set of 50GbE IP core variations and designs to ensure the 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 IP 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 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. Performance and Resource Utilization for 50GbE IP Core

The following results are for a variant that excludes all optional features.

| Product | ALMs | Dedicated Logic Registers | M20K Memory Blocks |
|---------|------|---------------------------|--------------------|
| 50GbE | 9300 | 18600 | 3 |

Related Links

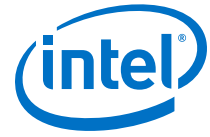
- [50GbE IP Core Parameters](#) on page 19
Information about the IP core parameters.
- [Fitter Resources Reports in the Quartus Prime Help](#)



1.5 Release Information

Table 5. 50GbE IP Core Current Release Information

| Item | Description |
|----------------|--|
| Version | 17.0 |
| Release Date | 2017.05.08 |
| Ordering Codes | IP-50GEUMACPHY (IPR-50GEUMACPHY for renewal) |



2 Getting Started

2.1 Installing and Licensing IP Cores

The Intel Quartus Prime software installation includes the Intel FPGA IP library. This library provides useful IP core functions for your production use without the need for an additional license. Some IP cores in the library require that you purchase a separate license for production use. The OpenCore® feature allows evaluation of any Intel FPGA IP core in simulation and compilation in the Quartus Prime software. Upon satisfaction with functionality and performance, visit the Self Service Licensing Center to obtain a license number for any Intel FPGA product.

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

Figure 3. IP Core Installation Path

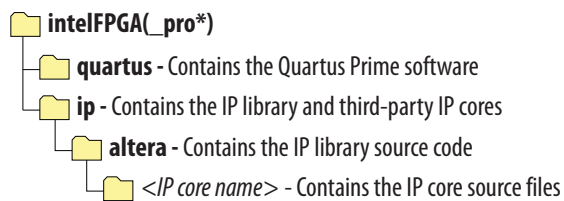


Table 6. IP Core Installation Locations

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

2.1.1 OpenCore Plus IP Evaluation

The free OpenCore Plus feature allows you to evaluate licensed MegaCore IP cores in simulation and hardware before purchase. Purchase a license for MegaCore IP cores if you decide to take your design to production. OpenCore Plus supports the following evaluations:

- Simulate the behavior of a licensed 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.



OpenCore Plus evaluation supports the following two operation modes:

- Untethered—run the design containing the licensed IP for a limited time.
- Tethered—run the design containing the licensed IP for a longer time or indefinitely. This operation requires a connection between your board and the host computer.

Note: All IP cores that use OpenCore Plus time out simultaneously when any IP core in the design times out.

Related Links

- [Quartus Prime Licensing Site](#)
- [Quartus Prime Installation and Licensing](#)

2.2 Specifying the 50GbE IP 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 Quartus Prime software.

1. Depending on whether you are using the Quartus Prime Pro Edition software or the Quartus Prime Standard Edition software, perform one of the following actions:
 - In the 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 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 Quartus Prime Standard Edition) or `<your_ip>.ip` (in Quartus Prime Pro Edition).
4. In the Quartus Prime Standard Edition software, you must select a specific 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 IP Core Parameters](#) on page 19 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 *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 Links

[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 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.
- **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 Links

Simulating Intel FPGA Designs

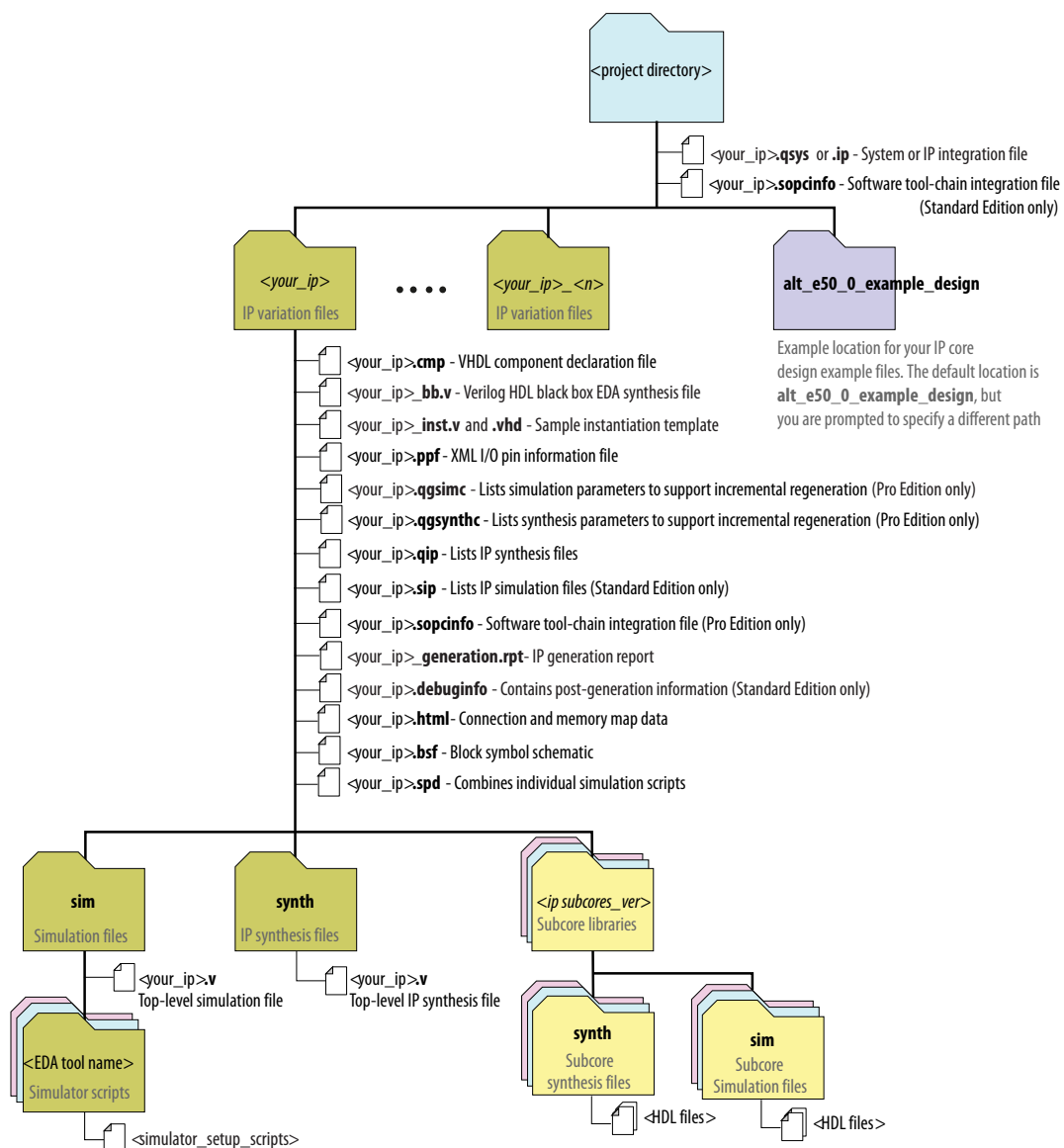
Quartus Prime Standard Edition Handbook Volume 3: Verification chapter that provides information about simulating Intel FPGA IP cores.

2.4 Generated File Structure

The 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 50GbE Design Example User Guide*.

Figure 4. IP Core Generated Files



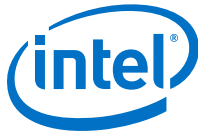


Table 7. IP Core Generated Files

| File Name | Description |
|--|---|
| <your_ip>.qsys (Quartus Prime Standard Edition only) | The Qsys system or top-level IP variation file. <your_ip> is the name that you give your IP variation. |
| <your_ip>.ip (Quartus Prime Pro Edition only) | |
| <system>.sopcinfo | Describes the connections and IP component parameterizations in your Qsys system. You can parse its contents to get requirements when you develop software drivers for IP components. Downstream tools such as the Intel Nios® II tool chain use this file. The .sopcinfo file and the system.h file generated for the Nios II tool chain include address map information for each slave relative to each master that accesses the slave. Different masters may have a different address map to access a particular slave component. |
| <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 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 Qsys 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 Qsys interconnect. The Bus Analysis Toolkit uses this file to identify debug components in the Qsys interconnect. (Quartus Prime Standard Edition only) |
| <your_ip>.qgsimc | Lists simulation parameters to support incremental regeneration. (Quartus Prime Pro Edition only) |
| <your_ip>.qgsynthc | Lists synthesis parameters to support incremental regeneration. (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 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 Quartus Prime Block Diagram Files (.bdf). |
| <your_ip>.spd | Required input file for ip-make-simscrip 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. (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 Quartus Prime software generates the _inst.vhd file. |

continued...



| File Name | Description |
|-------------------------------------|--|
| <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 Qsys 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 Qsys can query for register map information. For system slaves, Qsys can access the registers by name. |
| <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 Quartus Prime software generates this file. |
| mentor/ | Contains a ModelSim script <code>msim_setup.tcl</code> to set up and run a simulation. |
| aldec/ | Contains a Riviera-PRO script <code>rivierapro_setup.tcl</code> to setup and run a simulation. |
| synopsys/vcs/ synopsys/vcsmx/ | Contains a shell script <code>vcs_setup.sh</code> to set up and run a VCS® simulation. Contains a shell script <code>vcsmx_setup.sh</code> and <code>synopsys_sim.setup</code> file to set up and run a VCS MX® simulation. |
| cadence/ | Contains a shell script <code>ncsim_setup.sh</code> 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, Qsys generates <code>synth/</code> and <code>andsim/</code> sub-directories. |

Related Links

[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 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 Links

[Quartus Prime Help](#)

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

2.5.2 Adding the Transceiver PLL

The 50GbE IP core targets Arria 10 GT devices. 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.



2.5.3 Handling Potential Jitter in Arria 10 Devices

The RX path in the 50GbE IP core includes cascaded PLLs. Therefore, the IP core clocks might experience additional jitter in 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 Links

<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 IP Core

The Quartus Prime software provides the options to specify design partitions and LogicLock® or LogicLock Plus 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 full design.

Related Links

- [Quartus Prime Standard Edition Handbook Volume 2: Design Implementation and Optimization](#)
Describes incremental compilation, design partitions, and LogicLock regions.
- [Quartus Prime Pro Edition Handbook Volume 2: Design Implementation and Optimization](#)
Describes incremental compilation, design partitions, and LogicLock Plus regions.

2.6 Compiling the Full Design and Programming the FPGA

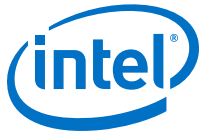
You can use the **Start Compilation** command on the Processing menu in the Quartus Prime software to compile your design. After successfully compiling your design, program the targeted Intel FPGA with the Programmer and verify the design in hardware.

Note: The 50GbE 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 Links

- [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.



3 50GbE IP Core Parameters

The 50GbE parameter editor provides the parameters you can set to configure the 50GbE IP 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 <i>IEEE 802.3-2012 IEEE Standard for Ethernet</i> . 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 Altera Debug Master Endpoint (ADME) | Enabled, Disabled | Disabled | If enabled, the IP core turns on the following features in the Arria 10 PHY IP core that is included in the 50GbE IP core: <ul style="list-style-type: none"> • Enable Altera Debug Master Endpoint (ADME) • Enable capability registers If turned off, the IP core is configured without these features. For information about these Arria 10 features, refer to the <i>Arria 10 Transceiver PHY User Guide</i> . |

Related Links

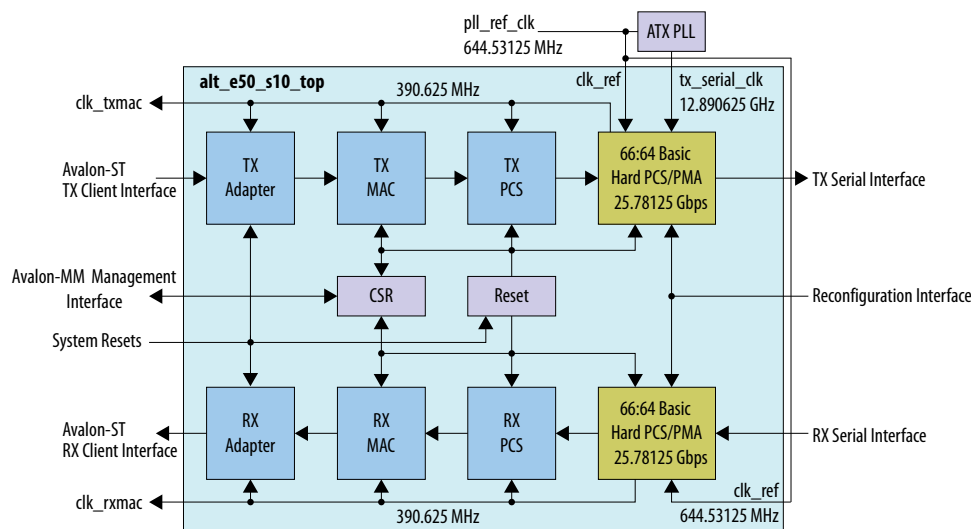
- [50G Ethernet Design Example User Guide](#)
Information about the **Example Design** tab in the 50GbE parameter editor.
- [Arria 10 Transceiver PHY User Guide](#)
Information about Arria 10 Native PHY IP core features, including ADME.

4 Functional Description

4.1 50GbE IP Core Functional Description

The Intel FPGA 50GbE 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 6. 50GbE 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 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 IP Core TX 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 7. 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:

- $\langle p \rangle$ = payload size, which is arbitrarily large
- $\langle s \rangle$ = number of padding bytes (0–46)
- $\langle g \rangle$ = number of IPG bytes

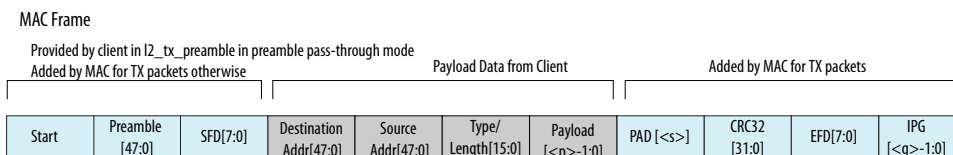
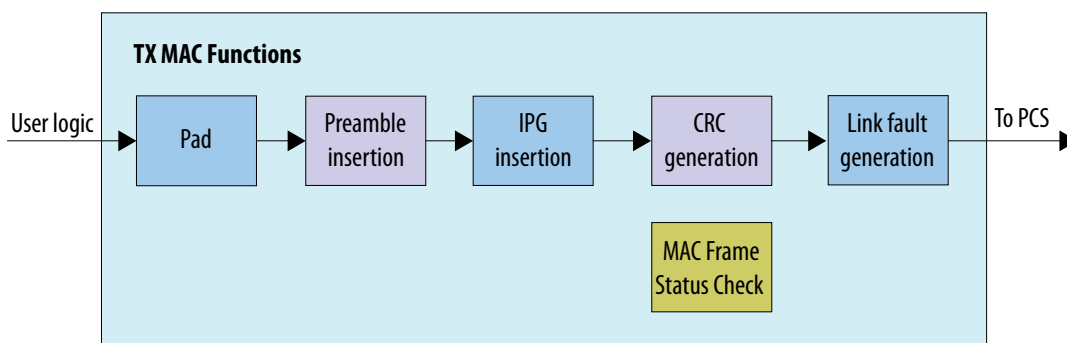


Figure 8. TX MAC Functions



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 IP Core RX 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 9. 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).

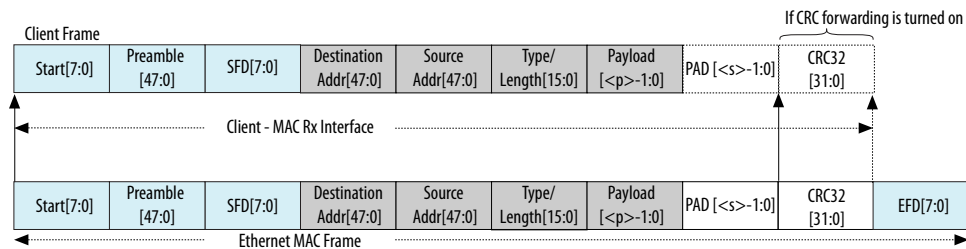




Figure 10. Flow of Client Frame With Preamble Pass-Through Turned Off

This figure uses the following notational conventions:

- $\langle p \rangle$ = payload size, which is arbitrarily large.
- $\langle s \rangle$ = number of padding bytes (0–46).

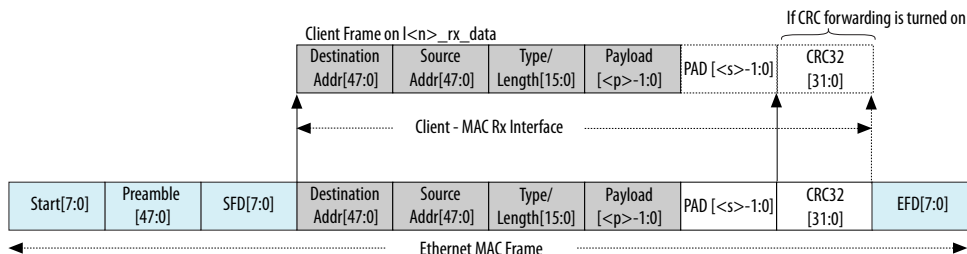
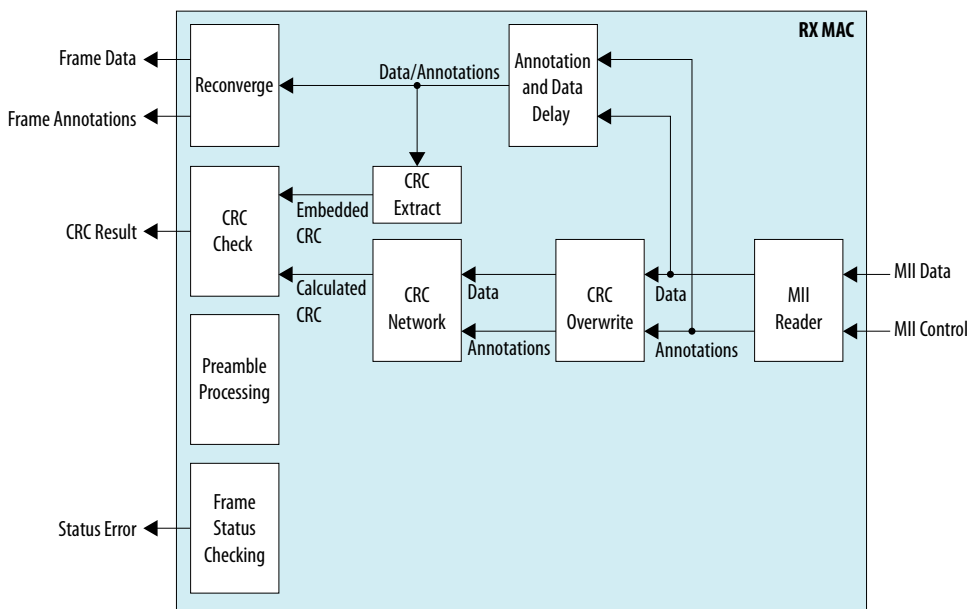


Figure 11. RX MAC Datapath



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-ST 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 IP 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 Links

[RX MAC Registers](#) on page 44

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 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 50GbE 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 (`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.

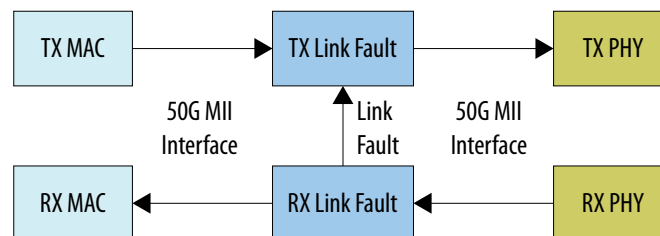
By default, the RX MAC strips off the CRC bytes before forwarding the packet to the MAC client. You can configure the core to retain the RX CRC and forward it to the client by updating the `MAC_CRC_CONFIG` register.

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 12. 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. |

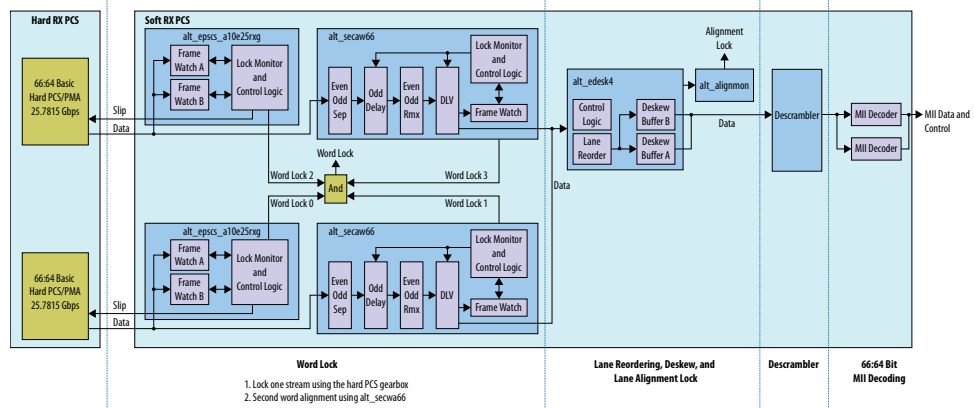
Related Links

- [TX MAC Registers](#) on page 43
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 bit-slip 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 13. 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 14. Byte Order on the Client Interface Lanes

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

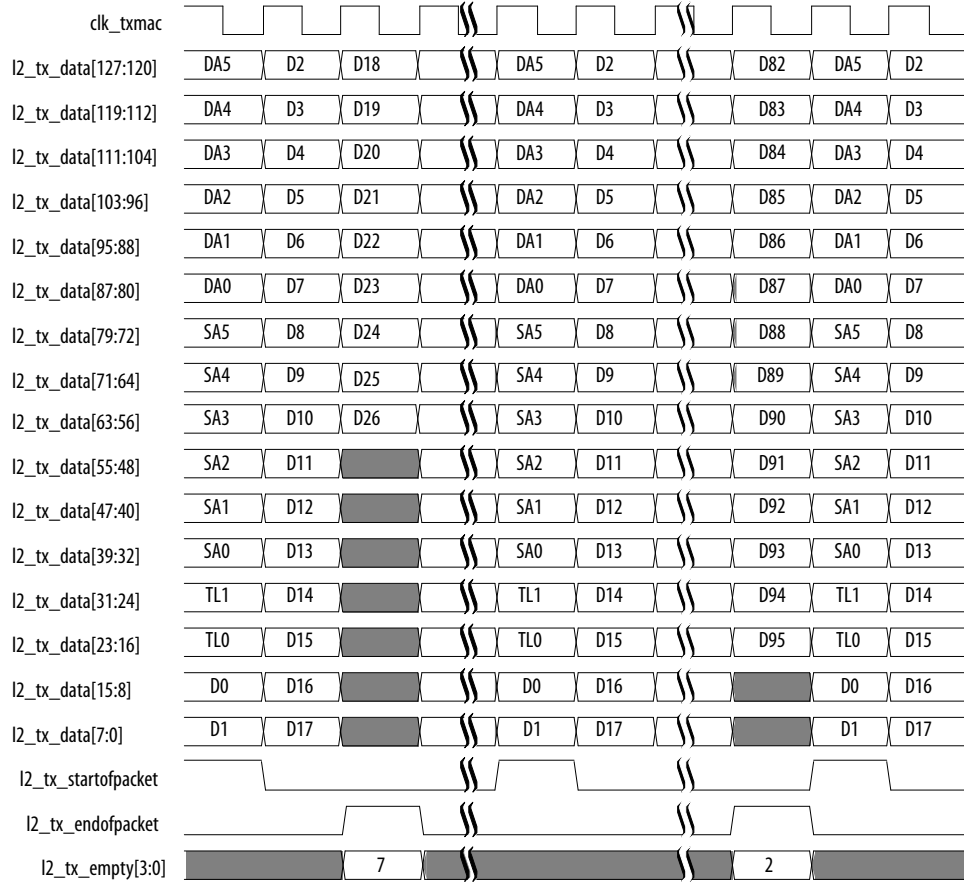
| | Destination Address (DA) | | | | | | Source Address (SA) | | | | | | Type/ Length (TL) | | Data (D) | | |
|-------|--------------------------|---------|---------|---------|--------|-------|---------------------|---------|---------|---------|--------|-------|----------------------|-------|----------|-----|----------|
| 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 15. Octet Transmission on the 50GbE Avalon-ST 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 Links

[IEEE website](#)

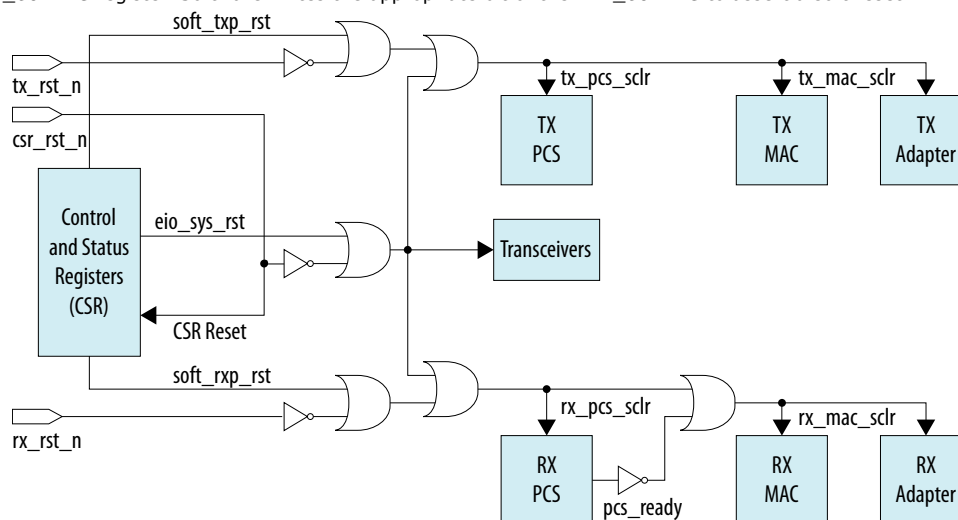
The *IEEE Standard for Ethernet, Section 4* is available on the IEEE website.

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 16. 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.

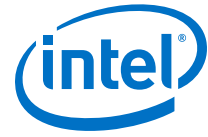


The internal soft reset signals reset the following functions:

- `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.

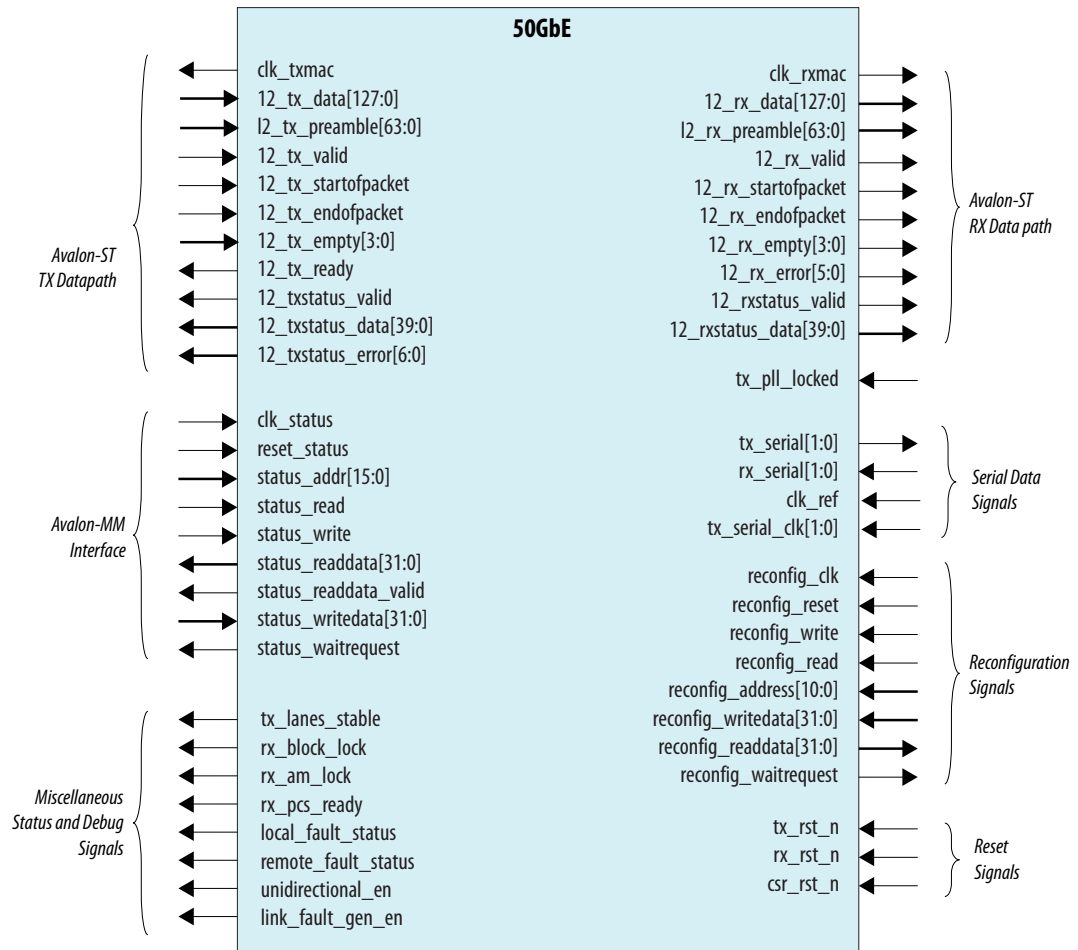
Related Links

- [PHY Registers](#) on page 40
- [Reset Signals](#) on page 39



6 Interfaces and Signal Descriptions

Figure 17. 50GbE Signals and Interfaces



6.1 TX MAC Interface to User Logic

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



Table 11. Avalon-ST TX Datapath

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

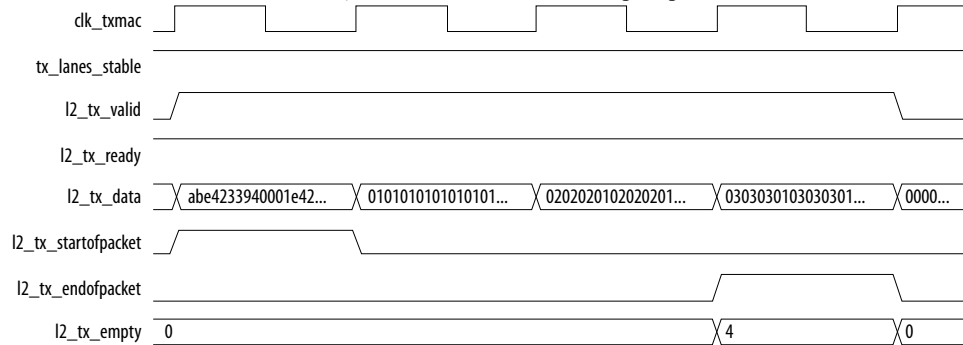
| Signal | Direction | Description |
|-------------------------------------|-----------|--|
| <code>clk_txmac</code> | Output | Clock for the TX logic. Derived from <code>pll_refclk</code> , and is an output from the 50GbE IP core. <code>clk_txmac</code> is guaranteed to be stable when <code>tx_lanes_stable</code> is asserted. The frequency of this clock is 390.625 MHz. All TX MAC interface signals are synchronous to <code>clk_txmac</code> . |
| <code>l2_tx_data[127:0]</code> | 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 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 <code>l2_tx_startofpacket</code> when you are assured the packet data to send on <code>l2_tx_data[127:0]</code> is available or will be available on time. |
| <code>l2_tx_preamble[63:0]</code> | Input | User preamble data. Available when you select <code>PREAMBLE_PASS-THROUGH</code> mode. User logic drives the custom preamble data when <code>l2_tx_startofpacket</code> is asserted. |
| <code>l2_tx_valid</code> | Input | When asserted, indicates valid data is available on <code>l2_tx_data[127:0]</code> . You must assert this signal continuously between the assertions of <code>l2_tx_startofpacket</code> and <code>l2_tx_endofpacket</code> for the same packet. |
| <code>l2_tx_startofpacket</code> | Input | When asserted, indicates the first byte of a frame. When <code>l2_tx_startofpacket</code> is asserted, the MSB of <code>l2_tx_data</code> drives the first byte of the packet. |
| <code>l2_tx_endofpacket</code> | Input | When asserted, indicates the end of a packet. The IP core ignores packets with length less than nine bytes. |
| <code>l2_tx_empty[3:0]</code> | | Specifies the number of empty bytes on <code>l2_tx_data</code> when <code>l2_tx_endofpacket</code> is asserted. |
| <code>l2_tx_ready</code> | Output | When asserted, indicates that the MAC can accept the data. |
| <code>l2_txstatus_valid</code> | Output | When asserted, indicates that <code>l2_txstatus_error[6:0]</code> is driving valid data. |
| <code>l2_txstatus_data[39:0]</code> | Output | Specifies information about the transmit frame. The following fields are defined: <ul style="list-style-type: none"> [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 |
|------------------------|-----------|--|
| | | <ul style="list-style-type: none"> 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 byte 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: <ul style="list-style-type: none"> Bits[6:3]: Reserved Bit[2]: Payload length error Bit[1]: Oversized frame Bit[0]: Reserved |

Figure 18. Client to 50GbE MAC Avalon-ST 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], 0xabe4233940001e42... in this timing diagram.



Related Links

[Avalon Interface Specifications](#)

Detailed information about Avalon-ST interfaces.

6.2 RX MAC Interface to User Logic

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

Table 12. Avalon-ST 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. |

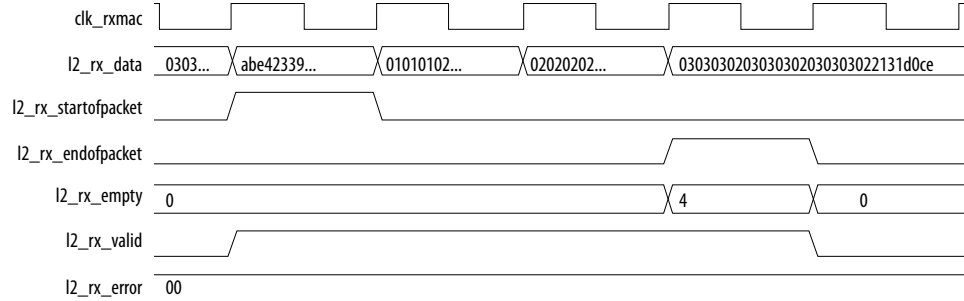
continued...



| 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: <ul style="list-style-type: none"> • 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: <ul style="list-style-type: none"> • [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 byte of the FCS • Bits[15:0]: Specifies the payload length |

Figure 19. 50GbE MAC to Client Avalon-ST 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 Links

[Avalon Interface Specifications](#)

Detailed information about Avalon-ST interfaces.

6.3 Transceivers

The transceivers require a separately instantiated advanced transmit (ATX) PLL to generate the high speed serial clock. For the 50GbE IP 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 Links

- [Adding the Transceiver PLL on page 15](#)
- [Ethernet section of the 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-MM interface.



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

Related Links

[\[Transceiver Reconfiguration\] Ports and Parameters section of the *Arria 10 Transceiver PHY User Guide*](#)

Provides more information about the transceiver reconfiguration interface, including timing diagrams for reads and writes.

6.5 Avalon-MM Management Interface

You access control and status registers using an Avalon-MM 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 counters; while this reset is in process, the Avalon-MM management interface does not respond.

Table 15. Avalon-MM Management Interface

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

Related Links

- [Control and Status Register Descriptions](#) on page 40
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-MM 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 |
|------------------------------|-----------|--|
| <code>tx_lanes_stable</code> | Output | Asserted when all TX lanes are stable and ready to transmit data. |
| <code>rx_block_lock</code> | Output | Asserted when all lanes have identified 66-bit block boundaries in the serial data stream. |
| <code>rx_am_lock</code> | Output | Asserted when all lanes have identified alignment markers in the data stream. |
| <code>rx_pcs_ready</code> | Output | Asserted when the RX lanes are fully aligned and ready to receive data. |
| <i>continued...</i> | | |



| Signal | Direction | Description |
|---------------------|-----------|--|
| 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 <i>Clause 66</i> 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 <i>Clause 66</i> for unidirectional support. This signal is available if you turn on Enable link fault generation in the parameter editor. |

Related Links

[Debugging the Link](#) on page 46

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 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 <code>tx_lanes_stable</code> 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 <code>rx_pcs_ready</code> 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 <code>tx_lanes_stable</code> and <code>rx_pcs_ready</code> 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-MM 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 |

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 Links

[Avalon-MM Management Interface](#) on page 38
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, "50". | 0x0000 3530 | RO |
| <i>continued...</i> | | | | |

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

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

ISO
9001:2008
Registered



| Addr | Name | Description | Reset | Access |
|-------|---------------|--|---|--------|
| 0x303 | PHY_NAME_1 | Next characters of IP core variation identifier string, "GE". | 0x0000 4745 | RO |
| 0x304 | PHY_NAME_2 | Final characters of IP core variation identifier string, "pcs". | 0x0070 6373 | RO |
| 0x310 | PHY_CONFIG | PHY configuration registers. The following bit fields are defined: <ul style="list-style-type: none"> Bit[0]: : <code>eio_sys_rst</code>. Full system reset (except registers). Set this bit to initiate the internal reset sequence. Bit[1]: <code>soft_txp_rst</code>. TX soft reset. Resets TX PCS, MAC, and adapter. Bit[2]: <code>soft_rxp_rst</code>. RX soft reset. Resets RX PCS, MAC, and adapter. Bit[3]: Reserved. Bit[4]: <code>set_ref_lock</code>. Directs the RX CDR PLL to lock to the reference clock. Bit[5]: <code>set_data_lock</code>. Directs the RX CDR PLL to lock to data. Bits[31:6]: Reserved. | 26'hX_2'b0_1'bX_3'b0 (X= don't care) | 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 | 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 | 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 <code>EIO_FLAGS</code> register. The value in the <code>EIO_FLAG_SEL</code> register directs the IP core to make available the following FIFO flag: <ul style="list-style-type: none"> 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 | RW |
| 0x315 | EIO_FLAGS | PCS indirect data. To read a FIFO flag, set the value in the <code>EIO_FLAG_SEL</code> register to indicate the flag you want to read. After you specify the flag in the <code>EIO_FLAG_SEL</code> register, each bit [n] in the <code>EIO_FLAGS</code> register has the value of that FIFO flag for the transceiver channel for lane [n]. | 30'hX2'b0 | RO |
| 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 | RO |
| 0x322 | PHY_CLK | The following encodings are defined: | 29'hX3'b00 | RO |

continued...



| Addr | Name | Description | Reset | Access |
|-------|------------------------|--|-------------|--------|
| | | <ul style="list-style-type: none"> • 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 | | |
| 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 | 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 | RW |
| 0x326 | RX_PCS_FULLY_ALIGNED_S | Indicates the RX PCS is fully aligned and ready to accept traffic. <ul style="list-style-type: none"> • Bit[0] : RX PCS fully aligned status. | 31'hX1'b0 | 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 | RO |
| 0x329 | LANE_DESKEWED | The following encodings are defined: <ul style="list-style-type: none"> • 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 | RO |
| 0x330 | PCS_VLANE | The following encodings are defined: <ul style="list-style-type: none"> • 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. | 24'bX8'b0 | RO |
| 0x340 | KHZ_REF | Reference clock frequency in KHz, assuming the clk_status clock is 100 MHz. The reference clock frequency is the value in KHZ times the frequency of the clk_status clock, divided by 100. | 0x0000 0000 | RO |
| 0x341 | KHZ_RX | RX clock (clk_rxmac) frequency in KHz, assuming the clk_status clock has the frequency of 100 MHz. The RX clock | 0x0000 0000 | RO |

continued...



| Addr | Name | Description | Reset | Access |
|-------|--------|---|-------------|--------|
| | | frequency is the value in KHZ times the frequency of the <code>clk_status</code> clock, divided by 100. | | |
| 0x342 | KHZ_TX | TX clock (<code>clk_txmac</code>) frequency in KHz, assuming the <code>clk_status</code> clock has the frequency of 100 MHz. The TX clock frequency is the value in KHZ times the frequency of the <code>clk_status</code> clock, divided by 100. | 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 5478 | RO |
| 0x404 | TXMAC_NAME_2 | Final 4 characters of IP core variation identifier string, "00CSR". The "00" is unprintable. | 0x0043 5352 | RO |
| 0x405 | LINK_FAULT | <p>Link Fault Configuration Register. The following bits are defined:</p> <ul style="list-style-type: none"> 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: <ul style="list-style-type: none"> 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 | 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. | 0xXXXX 0004 | RW |

continued...



| Addr | Name | Description | Reset | Access |
|-------|--------------------|---|-------------|--------|
| | | Bits [31:8] of this register are Reserved. | | |
| 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 | RW |
| 0x40A | TXMAC_CONTROL | TX MAC Control Register. A single bit is defined: <ul style="list-style-type: none"> Bit [1] – VLAN detection disabled. This bit is deasserted by default, implying VLAN detection is enabled. | 30'hX2'b0X | 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 |
| 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, "00CSR". The "00" is unprintable. | 0x0043 5352 | RO |
| 0x506 | RXMAC_SIZE_CONFIG | Specifies the maximum frame length available. The MAC asserts <code>l2_rx_error[3]</code> when the length of the received frame exceeds the value of this register. | 0xXXXX 2580 | RW |
| 0x507 | MAC_CRC_CONFIG | The RX CRC forwarding configuration register. The following encodings are defined: <ul style="list-style-type: none"> 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 | RW |
| 0x508 | LINK_FAULT | Link Fault Status Register. For regular (non-unidirectional) Link Fault, implements <i>IEEE 802.3 BA Ethernet Clause 81.3.4</i> . For unidirectional Link Fault, implements <i>IEEE 802.3 Ethernet Clause 66</i> . | 30'hX2'b00 | RO |
| 0x50A | RXMAC_CONTROL | RX MAC Control Register. The following bits are defined: | 27'h0_5'b11X0X | RW |

continued...



| Addr | Name | Description | Reset | Access |
|------|------|--|-------|--------|
| | | <ul style="list-style-type: none">• 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. | | |

Related Links

[IP Core Strict SFD Checking](#) on page 24



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 50GbE 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 in the Control and Status registers.

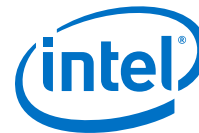
To check for word lock: Clear the `FRM_ERR` register by writing the value of 1 followed by another write of 0 to the `SCLR_FRM_ERR` register at offset 0x324. Then read the `FRM_ERR` register at offset 0x323. If the value is zero, the core has word lock. If non-zero the status is indeterminate

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 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.

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 | <p>Added link to KDB Answer that provides workaround for potential jitter on Arria 10 devices due to cascading ATX PLLs in the IP core. Refer to Handling Potential Jitter in Arria 10 Devices on page 17 and to Compiling the Full Design and Programming the FPGA on page 17.</p> <p>Reorganized Features list for reading clarity. Refer to 50GbE IP Core Supported Features on page 6.</p> <p>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 40.</p> <p>Clarified that the <code>reset_status</code> signal remains visible in future releases for backward compatibility, rather than being removed. The design must tie this input signal low. Refer to Avalon-MM Management Interface on page 38.</p> |
| 2017.05.08 | 17.0 | Initial public release. |

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

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

ISO
9001:2008
Registered