

Release Information

Table 1 provides information about this release of the Altera® 10-Gbps Ethernet Reference Design.

Table 1. Release Information

Item	Description
Version	8.1
Ordering Code	IP-10ETHERNET
Product ID(s)	E003
Vendor ID(s)	6AF7

Device Family Support


Table 2 shows the level of support offered by the 10-Gbps Ethernet reference design to each Altera device family.

Table 2. Device Family Support

Device Family	Support
Arria® GX	Full
Stratix® II	Full
Stratix II GX	Full
Stratix III	Full
Stratix IV	Preliminary
Other device families	No support

Features

- 10 Gbps Ethernet receiver and transmitter media access controller (MAC) in full-duplex mode, which conforms to the *IEEE 802.3 2005* standard
- Passed the University of New Hampshire Interoperability Lab (UNH) 10 Gb Ethernet tests of MAC, 10GBASE-X physical coding sublayer (PCS), and physical medium attachment (PMA)
- Verified and hardware tested with standard 10-Gbps Ethernet test equipment

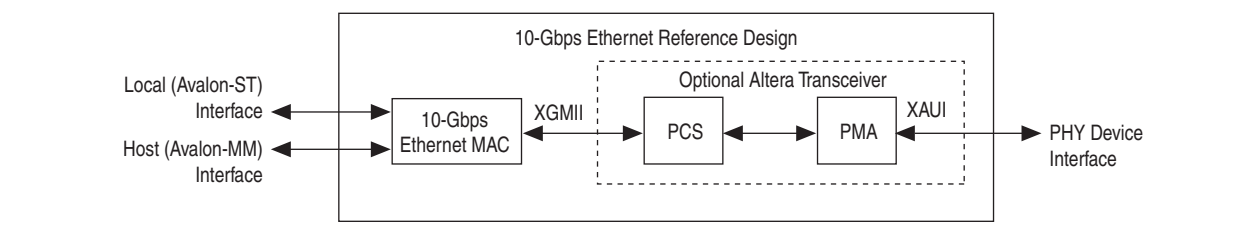
- Flexible standard interfaces:
 - Seamless interface to commercial Ethernet PHY devices via 10-Gbps media independent interface (XGMII) or 10-Gbps attachment unit interface (XAUI)
 - Management data input/output (MDIO) master interface for PHY device management
 - 64-bit wide FIFO or no-FIFO interface to application logic based on the Avalon® Streaming (Avalon-ST) interface
-  For more information on Avalon-ST and Avalon-MM interfaces, refer to the [Avalon Interface Specifications](#).
- 10-Gbps full-duplex throughput rate
 - Flow control by programmable pause quanta
 - Pause frame generation controllable by user applications, enabling flexible traffic flow control
 - Parameterizable FIFO size (64 bytes to 64 Kbytes) and programmable threshold levels
 - Programmable MAC addresses and receive packet filtering based on up to two unicast and broadcast destination MAC address, and one destination VLAN address
- Programmable maximum receiving frame length up to 16 Kbytes, including jumbo frames (1,519 to 9,600 bytes)
- Support for promiscuous (transparent) and non-promiscuous (filtered) modes
- Support for virtual local area network (VLAN) and stacked VLAN tagged frames according to the *IEEE 802.1Q* standard
- Management interface and loopback for system test—local or line loopback on the XGMII
- Statistics counters supporting RMON (*RFC 2819*), Ethernet type MIB (*RFC 3635*), and interface group MIB (*RFC 2863*)
- Filtering of frames with CRC errors, length-check error, or oversized errors
- Easy-to-use MegaWizard® interface for parameterization
- IP functional simulation models for use in Altera-supported VHDL and Verilog HDL simulators
- Verilog HDL and VHDL testbench or verification environment

General Description

The 10-Gbps Ethernet reference design provides an integrated Ethernet MAC, PCS, and PMA solution for Ethernet applications. The reference design works in full-duplex mode and supports both switching and network interface card (NIC) or line-card applications, by providing transparent and full Ethernet frame termination and generation.

Figure 1 shows the 10-Gbps Ethernet Reference Design block diagram. The function comprises the following main blocks: MAC and PCS. All blocks are parameterizable at synthesis time. A memory-mapped register interface controls the MAC and PCS blocks.

Figure 1. Block Diagram



For the local interface, the reference design provides standard Avalon-ST interfaces, to allow interoperability with other Avalon-ST components. A host processor can manage the reference design via a separate Avalon Memory-Mapped (Avalon-MM) interface that provides access to the control register space. For the PHY interface, the reference design can seamlessly connect to any industry standard 10-Gb Ethernet PHY or MAC device via XGMII or XAUI (Arria GX, Stratix II GX, or Stratix IV devices only).


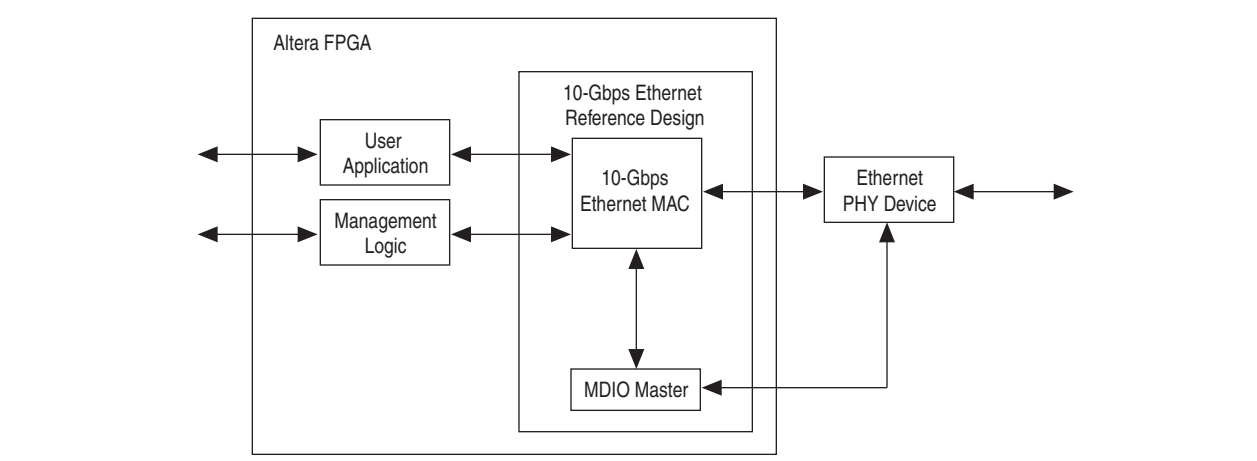
 For more information on the Avalon-ST interface, refer to the [Avalon Interface Specifications](#).

Figure 2 shows an example application using the reference design as a stand-alone MAC without the PCS and PMA, serving as a bridge between the user application and Ethernet PHY devices.

Figure 2. Stand-Alone 10-Gbps Ethernet MAC



Verification

Altera verified the 10-Gbps Ethernet reference design through extensive in-house simulation and internal hardware verification.

Altera used a highly parameterizeable transaction-based testbench to test the following aspects of the reference design:

- Register access
- Management data input and output (MDIO) access
- Frame transmission and error handling
- Frame reception and error handling
- Received ethernet frame MAC address filtering
- Flow control

Altera has also validated the reference design with copper and optical media using the Spirent SmartBits and the following Altera development boards:

- Stratix II GX video development board
- Stratix II GX PCI Express development board

In the copper medium, Altera tested the reference design with an external 10GBASE-CX4 PHY device, a CX4 switch and CX4 test equipment; in the optical medium, Altera tested the 10-Gbps Ethernet reference design with a 10-Gbps X2 PHY board and various X2 optical pluggable modules interfacing to standard test equipment.



For more information on X2, refer to www.X2msa.org.

For hardware testing, a 10-Gbps Ethernet MAC, PCS, PMA, and an internal system design are implemented in the FPGA. The internal system retrieves all frames received from the test equipment by the MAC function and returns them to the sender by manipulating the address fields, implementing a loopback at the internal system interface of the MAC.

Performance and Resource Utilization

Table 3 shows the typical expected performance for different parameters, using the Quartus II software version 8.1 targeting a Stratix II GX (EP2SGX30DF780C3) device.

For C4 and C5 device speed grades the f_{MAX} is 156.25 MHz.

Table 3. 10-Gbps Ethernet Performance and Resource Utilization—Stratix II GX Device

XAUI	FIFO (eight-byte words)	MDIO	Statistics Counter	Combinational ALUTs	Logic Registers	f_{MAX} (MHz)			Memory	
						Avalon-MM	Avalon-ST	System Clock	M4K	M512
Yes	64	Yes	Yes	4,141	3,759	144	200	183	9	13
Yes	64	Yes	No	3,348	2,962	150	200	183	8	3
Yes	64	No	No	3,300	2,828	150	200	183	8	3
Yes	No	No	No	2,973	2,509	150	200	171	4	3

Table 4 shows the typical expected performance, using the Quartus II software v8.1 targeting a Stratix IV (EP4SGX70DF29C3) device.

Table 4. 10-Gbps Ethernet Performance and Resource Utilization—Stratix IV Device

XAUI	FIFO (eight-byte words)	MDIO	Statistics Counter	Combinational ALUTs	Logic Registers	f _{MAX} (MHz)			Memory	
						Avalon-MM	Avalon-ST	System Clock	M9K	
Yes	64	Yes	Yes	4,148	3,787	142	200	179	16	

Table 5 shows the typical expected performance, using the Quartus II software v8.1 targeting a Arria GX (EP1AGX20CF780C6) device.

Table 5. 10-Gbps Ethernet Performance and Resource Utilization—Arria GX Device

XAUI	FIFO (eight-byte words)	MDIO	Statistics Counter	Combinational ALUTs	Logic Registers	f _{MAX} (MHz)			Memory	
						Avalon-MM	Avalon-ST	System Clock	M4K	M512
Yes	64	Yes	Yes	4,218	3,785	120	160	156.25	11	13

Installation and Licensing

To install the reference design, unzip the **.tar.gz** file to a directory on your PC.



Do not unzip to the standard Altera IP directory **altera/<version>/ip**.

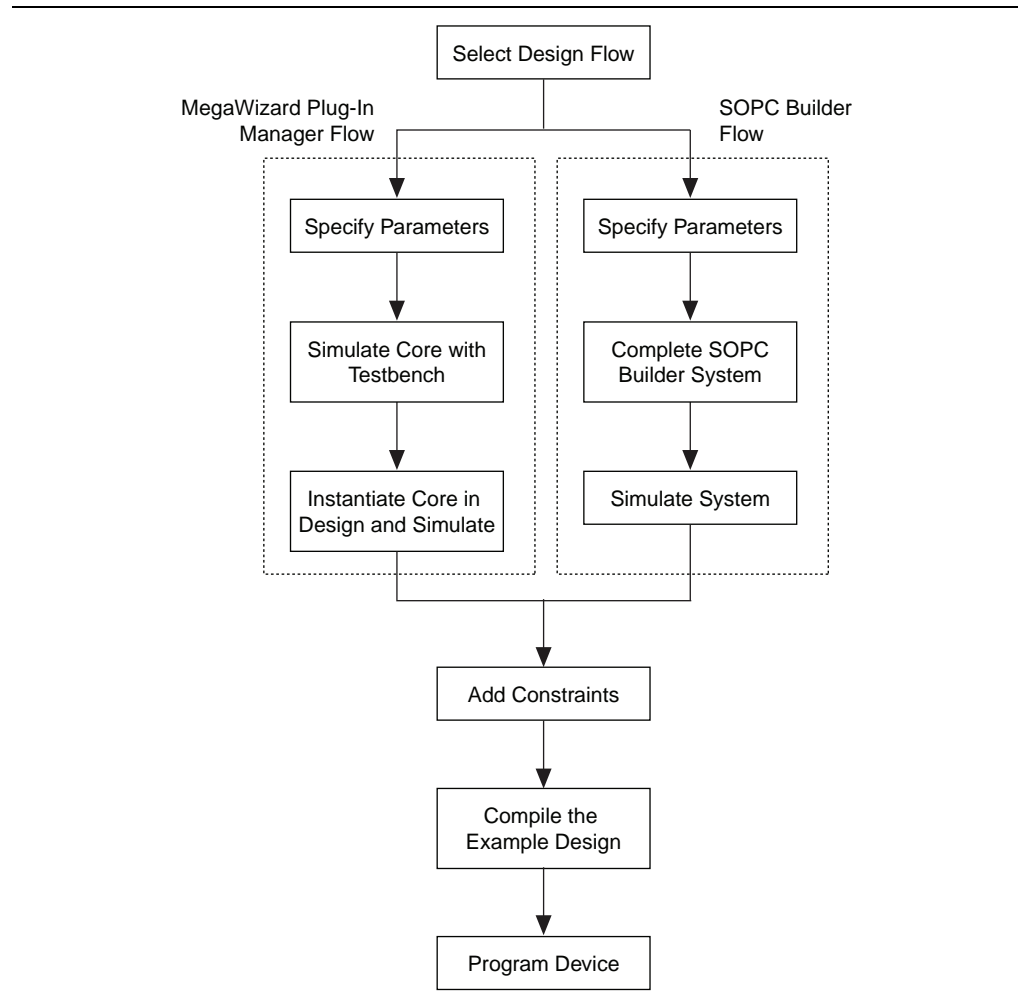
In the Quartus II software add the 10-Gbps Ethernet library to the user library:

1. In the Quartus II software, on the Tools menu click **Options**.
2. Click **Global User Libraries**.
3. In Library Name specify the **eth_10g\lib** directory.

After you purchase a license for the reference design, you can request a license file from the Altera website at www.altera.com/licensing and install it on your computer. When you request a license file, Altera emails you a **license.dat** file. If you do not have Internet access, contact your local Altera representative.

Getting Started

Figure 3 shows the stages for creating a system with the 10-Gbps Ethernet Reference Design and the Quartus® II software. Each of the stages is described in detail in subsequent sections.

Figure 3. 10-Gbps Ethernet Design Flow

Design Flow Selection

You customize the 10-Gbps Ethernet reference design by specifying parameters using the 10-Gbps Ethernet MegaWizard interface, launched from either the MegaWizard Plug-in Manager or SOPC Builder in the Quartus II software.

MegaWizard Plug-in Manager Flow

Use the MegaWizard Plug-in Manager flow to create a variant that you can instantiate manually in your design.

SOPC Builder Flow

Use the SOPC Builder flow for the following advantages:

- You want to rapidly create a new SOPC Builder system design that includes an Ethernet interface.
- You want to use the reference design in conjunction with other components available in SOPC Builder such as the Nios II processor, external memory controllers and the scatter/gather DMA controller.

MegaWizard Plug-in Manager Flow

The MegaWizard Plug-in Manager flow allows you to customize the 10-Gbps Ethernet reference design, and manually integrate the function into your design.

Specify Parameters

Follow the steps below to specify 10-Gbps Ethernet parameters using the MegaWizard Plug-in Manager flow.

1. Create a Quartus II project using the **New Project Wizard** available from the File menu.
2. Launch **MegaWizard Plug-in Manager from the Tools menu, and follow the prompts in the MegaWizard Plug-in Manager interface to create a custom megafunction variation.**



You can find **10-Gbps Ethernet** by expanding **Installed Plug-Ins > Interfaces > Ethernet.**

3. Specify the parameters on all pages in the **Parameter Settings** tab.

For detailed explanation of the parameters, refer to the [“Parameter Settings” on page 9.](#)

4. On the EDA tab, turn on **Generate simulation model** to generate an IP functional simulation model for the reference design in the selected language.

An IP functional simulation model is a cycle-accurate VHDL or Verilog HDL model produced by the Quartus II software.



Some third-party synthesis tools can use a netlist that contains only the structure of the reference design, but not detailed logic, to optimize performance of the design that contains the reference design. If your synthesis tool supports this feature, turn on **Generate netlist.**



Use the simulation models only for simulation and not for synthesis or any other purposes. Using these models for synthesis creates a non-functional design.

5. On the **Summary** tab, select the files you want to generate. A grey checkmark indicates a file that is automatically generated. All other files are optional.
6. Click **Finish** to generate the reference design and supporting files.

Simulate with Provided Testbench

You can simulate the reference design using the IP functional simulation model and testbench generated by the 10-Gbps Ethernet wizard.



For more information on the testbench, see [“Testbench” on page 51.](#)

To run a simulation using the ModelSim simulator, follow these steps:

1. Start the ModelSim simulator.
2. Change the working directory to `<project directory>\tb\verilog` or `<project directory>\tb\vhdl.`

- Run the following command to set up the required libraries, compile the generated IP functional simulation model, and exercise the simulation model with the provided testbench:

```
demo_run_modelsim.tcl ←
```

The ModelSim transcript pane (in Main window) displays messages from the testbench reflecting the current task being performed.

Instantiate the Reference Design in your Design

You can now integrate your 10-Gbps Ethernet reference design variation into your design, and simulate the system with your custom testbench.

SOPC Builder Flow

The SOPC Builder flow allows you to add the reference design directly to a new or existing SOPC Builder system. You can also easily add other available components to quickly create an SOPC Builder system with an Ethernet interface, such as the Nios II processor, external memory controllers and scatter/gather DMA controllers. SOPC Builder automatically creates the system interconnect logic and system simulation environment.



For more information on SOPC Builder, refer to *Volume 4* of the *Quartus II Handbook*.

Specify Parameters

Follow the steps below to specify 10-Gbps Ethernet parameters using the SOPC Builder flow.

- Create a new Quartus II project using the **New Project Wizard** available from the File menu.
- Launch **SOPC Builder** from the Tools menu.
- For a new system, specify the system name and language.
- Add the reference design to the SOPC Builder component list:
 - On the Tools menu, click **Options**.
 - In the **Category** list click **IP Search Path**.
 - Click **Add**.
 - Add the `<path>eth_10g\lib\ip_toolbench`.
 - Click **Finish**.
 - On the File menu, click **Refresh Component List**.
- Add 10-Gbps Ethernet to your system from the System Contents tab.



You can find **10-Gbps Ethernet** by expanding **Interface Protocols > Ethernet**.

- Specify the required parameters on **all pages in the Parameter Settings** tab.



For detailed explanation of the parameters, refer to the “**Parameter Settings**” on page 9.

7. Click **Finish** to complete the reference design and add it to the system.

Complete the SOPC Builder System

Follow the steps below to complete the SOPC Builder system.

1. Add and parameterize any additional components to the system.



A typical SOPC builder system that enables Ethernet connectivity uses a scatter/gather DMA controller on each of the transmit and receive paths, and a Nios II processor for configuration and control.

2. Connect the components using the SOPC Builder patch panel.
3. If you intend to simulate your SOPC builder system, select **Simulate** on the System Generation tab to generate a functional simulation model for the system.
4. Click **Generate** to generate the system.

Simulate the System

During system generation, SOPC Builder optionally generates a simulation model and testbench for the entire system which you can use to easily simulate your system in any of Altera's supported simulation tools. SOPC Builder also generates a set of ModelSim Tcl scripts and macros that you can use to compile the testbench, IP Functional simulation models and plain-text RTL design files that describe your system in the ModelSim simulation software.



For more information about simulating SOPC Builder systems, refer to *Volume 4* of the *Quartus II Handbook* and *AN 351: Simulating Nios II Systems*.

Design Compilation and Device Programming

You can use the Quartus II software to compile your design. After a successful compilation, you can program the targeted Altera device and verify the design in hardware.



For more information on compiling a design and programming Altera devices, refer to Quartus II Help.

Parameter Settings

This section describes the parameters and how they affect the behavior of the reference design. Each section corresponds to a page in the **Parameter Settings** tab in the 10-Gbps Ethernet MegaWizard interface.

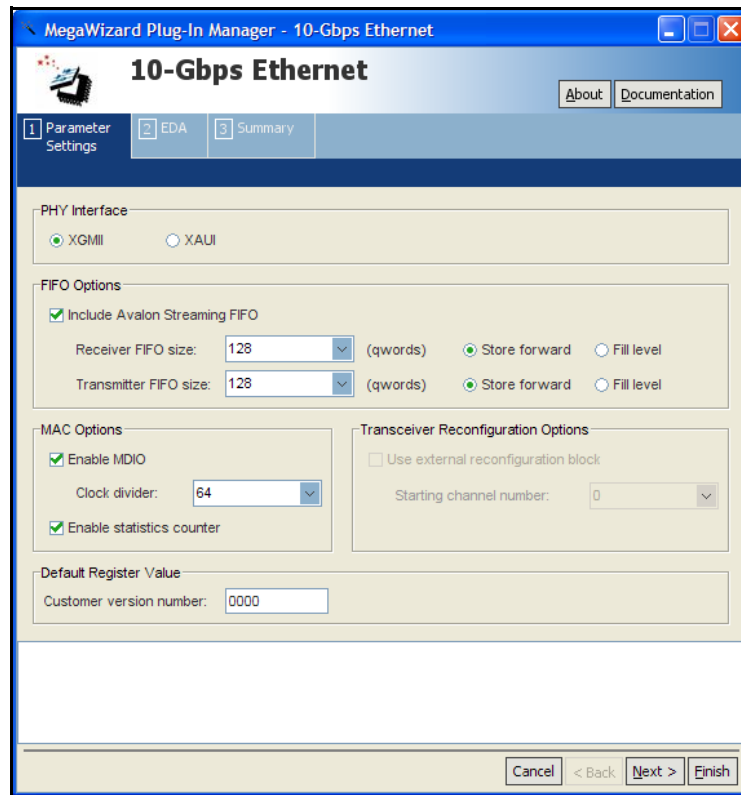


When parameterizing a reference design using the SOPC Builder flow, the **EDA** and **Summary** tabs are not visible. In the SOPC Builder flow, simulation model settings are inherited from options specified in SOPC Builder.



For more information on setting simulation options, refer to the Quartus II Help.

Figure 4. Options



PHY Interface

Select **XGMII** or **XAUI**. XGMII is a 32-bit double data rate interface at 156.25-MHz with four bits of control; XAUI is a four-lane serialized interface at 3.125 Gbps per lane (Arria GX, Stratix II GX, or Stratix IV devices only).

FIFO Options

Turn on **Include Avalon Streaming FIFO Buffer** to include a FIFO buffer in the local interface.


Select the receiver and transmitter **FIFO size**. FIFO size options are: 8, 64, 128, 256, 512, 1,024, 2,048, 4,096, or 8,192 eight-byte words.




The usable FIFO size is (wizard size – 1). For example, if you select a FIFO size of 128 eight-byte words, the size you can use is 127 eight-byte words.

Select **Fill level** (cut-through mode) and the read data available (`avl_st_rx_dav`) signal goes high when a threshold is reached or an end of packet (EOP). The fill level is passed continuously from the write side to the read side.

Select **Store forward** and the packet information is passed from the write side to the read side only when the EOP is reached on the write side. So data available only triggers on EOP. When you select **Store forward**, select the FIFO size to hold the longest possible frame in the system. Altera recommends twice the maximum possible frame size.

 For more information on thresholds, see [“Receiver FIFO Thresholds” on page 17](#) and [“Transmit FIFO Thresholds” on page 21](#).

 If you turn off **Include Avalon Streaming FIFO**, for the receiver there is no ready signal, so the data is sent automatically; for the transmitter, the read latency is 1. You must provide a full packet when the design indicates data is present.

MAC Options


Turn on **Enable MDIO**, so the reference design instantiates an MDIO master, see [“MDIO Interface” on page 27](#).

Turn on **Enable Statistics Counter** to allow reporting of statistics transmitted and received.

Enter a number for the **Customer version number**, see rev bit in [Table 14 on page 33](#).

Transceiver Reconfiguration Options

For Stratix II GX devices with XAUI, if you turn on **Use external reconfiguration block**, you can modify analog properties of the transceiver with an external reconfiguration block (ALT2GXB_RECONFIG). For Stratix IV devices with XAUI, you must have an external reconfiguration block, so this option is always on. For Arria GX devices, this option is not available.

 If you select this option, you must generate and use an external reconfiguration block. The external configuration block is an ALT2GXB_RECONFIG megafunction that you can edit and generate in the Quartus II software.

 For more information on the ALT2GXB_RECONFIG megafunction, refer to the [ALT2GXB_RECONFIG Megafunction User Guide](#).

In addition, when you turn on **Use external reconfiguration block**, you can select the starting channel number of the transceivers.


 For more information on starting channel number, refer to the [ALT2GXB_RECONFIG Megafunction User Guide](#).

Table 6 shows the reconfiguration block signals.

Table 6. External Reconfiguration Block Signals

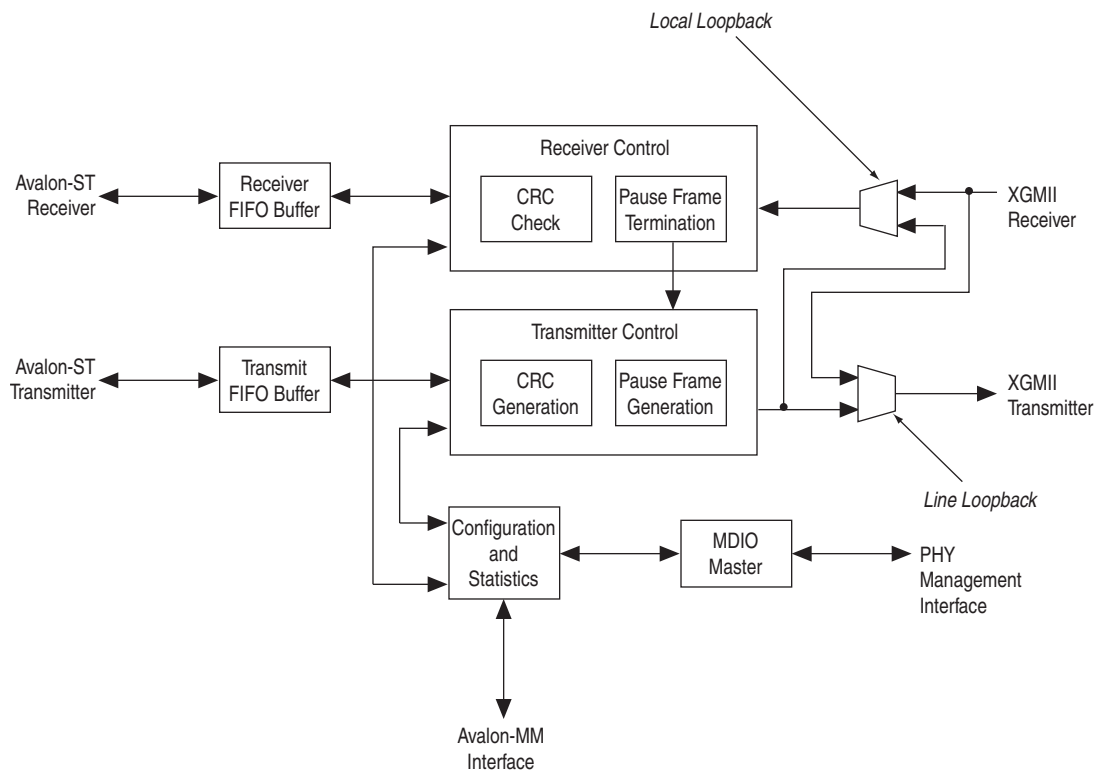
Signal	Size (Bits)		Direction
	Stratix II GX Devices	Stratix IV Devices	
reconfig_fromgxb	1	17	Output
reconfig_togxb	3	4	Input
reconfig_clk	1	1	Input

Functional Description

The 10-Gbps Ethernet reference design provides an Avalon® Streaming (Avalon-ST) interface to user applications and an industry standard XGMII or XAUI to external PHY devices.

Figure 5 shows a block diagram of the 10-Gbps Ethernet reference design.

Figure 5. Block Diagram



Frame Format

A basic IEEE 802.3 Ethernet frame comprises the following fields:

- Start—1 byte.
- Preamble—6 bytes.
- Start frame delimiter (SFD)—a 1-byte fixed value of 0xD5 which marks the beginning of a frame.
- Destination and source addresses—6 bytes each. The least significant byte is transmitted first.
- Length or type—a 2-byte value equal to or greater than 1,536 (0x600) indicates a type field. Otherwise, this field contains the length of the payload data. The most significant byte of this field is transmitted first on the serial line to the Ethernet network.
- Payload data and pad—variable length data and padding of 00 data if frame length is less than 64 bytes long.

- Frame check sequence (FCS)—a 4-byte cyclical redundancy check (CRC) value for detecting frame errors during transmission.
- End of frame delimiter (EFD)

Figure 6 shows the format of a basic frame.

Figure 6. Frame Format

Frame Length	1 byte	Start
	6 bytes	Preamble
	1 bytes	SFD
	6 bytes	Destination Address
	6 bytes	Source Address
	2 bytes	Length/Type
	0..1500/9000 bytes	Payload Data
	0..46 bytes	Pad
	4 bytes	Frame Check Sequence

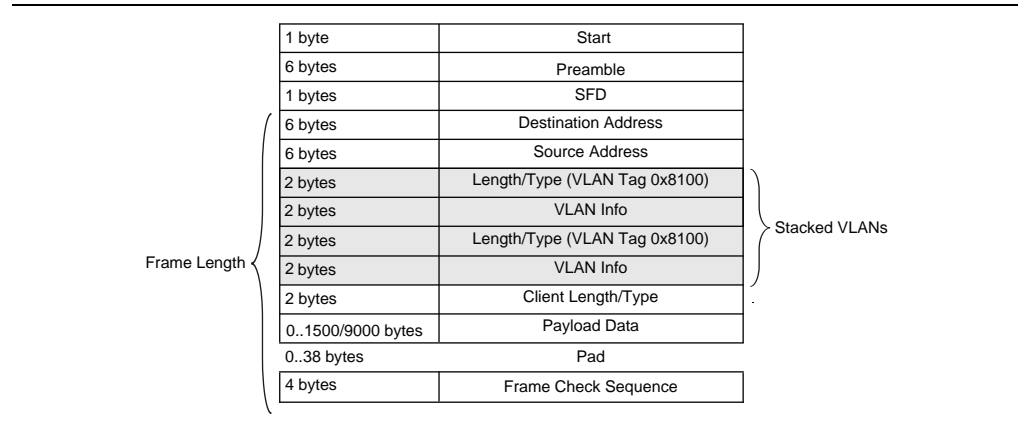
The extension of a basic frame is a virtual local area network (VLAN) tagged frame, which contains an additional 4-byte field for the VLAN tag and information between the source address and length/type fields. VLAN tagging is defined by the *IEEE 802.1Q* standard. VLANs can identify and separate many groups' network traffic from each other in enterprises and also in metro networks. Each VLAN group can consist of many users with varied MAC address in different geographical locations of a network. VLANs increase and scale the network performance and add privacy and safety to various groups or customers' network traffic.


VLAN tagged frames have a maximum length of 1,522 bytes, excluding the preamble and the SFD bytes. Figure 7 shows the format of a VLAN tagged frame.

Figure 7. VLAN Tagged Frame Format

Frame Length	1 byte	Start
	6 bytes	Preamble
	1 bytes	SFD
	6 bytes	Destination Address
	6 bytes	Source Address
	2 bytes	Length/Type (VLAN Tag 0x8100)
	2 bytes	VLAN Info
	2 bytes	Client Length/Type
	0..1500/9000 bytes	Payload Data
	0..42 bytes	Pad
4 bytes	Frame Check Sequence	

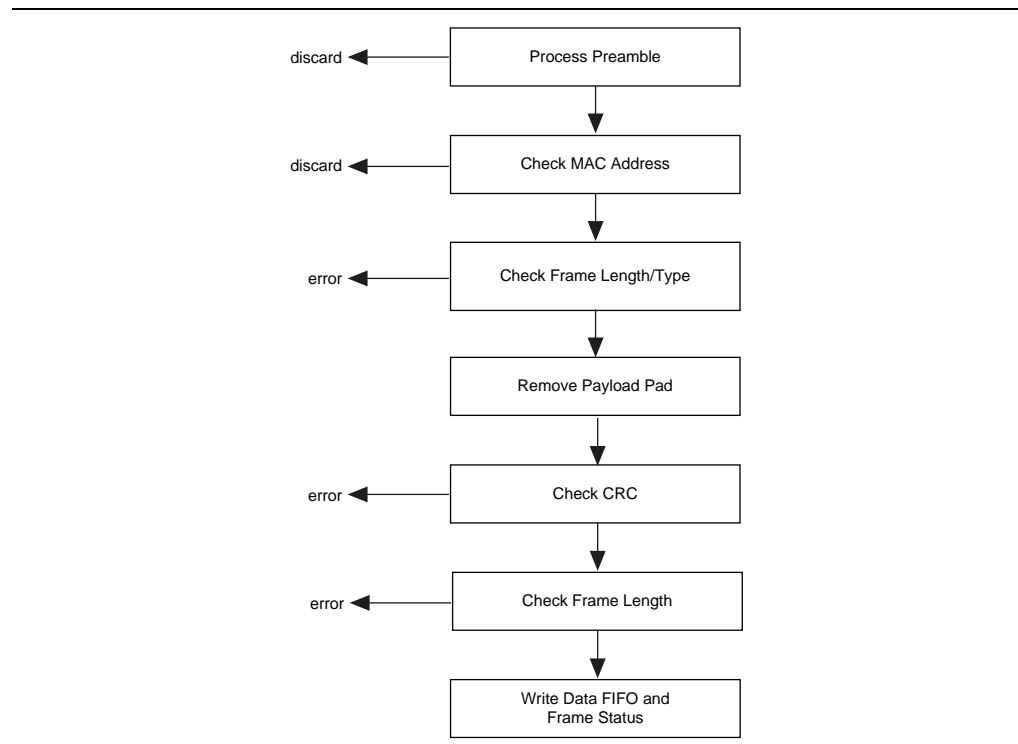
In some applications, frames can be tagged with two consecutive VLAN tags (stacked VLAN). Stacked VLAN frames contain an additional 8-byte field between the source address and client length/type fields, see Figure 8. Stacked VLANs allow virtual private networks (VPN) in metro Ethernet and across multiple carriers.

Figure 8. Stacked VLAN Tagged Frame Format

 See also [Figure 14](#).

Receiver Operation

This section describes the receiver operation. [Figure 9](#) shows the flow of the receiver operation.

Figure 9. Receiver Flow

Preamble Processing

The preamble sequence is Start, six preamble bytes, and SFD. If this sequence is incorrect the frame is ignored. The Start word must be on the receiving lane 0 (most significant byte).

The reference design uses the SFD byte (0xD5) to identify the last byte of the preamble. The reference design looks for the Start, 6 preambles and SFD. If not found the frame is ignored.

The IEEE standard specifies that frames must be separated by an interframe gap (IFG) of an average of 96 bit times.

The reference design removes all preamble and SFD bytes from accepted frames.

Address Checking

Bit 0 in the destination address field specifies the type of address.

- If bit 0 is 0, the destination address is a unicast (individual) address.
- If bit 0 is 1, the destination address defines a multicast (group) address.
- If all 48 bits in the destination address are 1, it is a broadcast address.

The reference design accepts broadcast frames if bit 28 (BROAD_FILTER_ENA) in the `command_config` register is set to 0, which is the default. If you enable promiscuous mode (PROMIS_EN bit in the `command_config` register = 1), address checking is omitted and all received frames are accepted.

A frame is accepted only if its unicast destination address matches any of the following addresses:

- The primary address, configured in the registers `mac_0` and `mac_1`—these 32-bit registers build a 48-bit MAC address
- The supplemental addresses, configured in the following registers:
`smac_0_0`, `smac_0_1`

Otherwise, the reference design discards the frame. The multicast address matching works the same way as the unicast. If the supplemental address is multicast, it is matched. For more information on the address registers, refer to [Table 14 on page 33](#).

Frame Length/Type Checking

If the length/type field represents the payload length, the reference design checks the payload length and reports any error in the frame status, `avl_st_rx_err`. Otherwise, the reference design forwards the frame to your application. The frame is forwarded even with the error except if filtered by the FIFO.

The maximum frame size is defined in a register, the value of the register determines the maximum length. The maximum frame size is 16 Kbyte. If the maximum frame size is exceeded, the packet is marked as error, and the error signal is set high.

VLAN Frame Processing

A value of 0x8100 in the length/type field denotes a VLAN tagged frame. A 2-byte VLAN tag follows the length/type field. VLAN tagged frames are received in the same manner as basic frames, and the entire frame, including the VLAN tag, is forwarded to the user application. The reference design sets the `avl_st_vlan_tag` signal to indicate that the current frame is a VLAN tagged frame.

The reference design removes the padding from VLAN tagged frames only when the value of the client length/type field, which comes after the VLAN control information field, is less than 42 and the `PAD_EN` bit in the `command_config` register is set to 1.

Stacked VLAN Frame Processing

A value of 0x8100 in the length/type field following the additional 2 bytes in VLAN tagged frames denotes a stacked VLAN tagged frame (see “Frame Format” on page 12). The reference design sets the `avl_st_vlan_vlan_tag` signal to indicate that the current frame is a stacked VLAN tagged frame.

The reference design removes the padding from stacked VLAN tagged frames only when the value of the client length/type field, which comes after the second VLAN tag field, is less than 38 and the `PAD_EN` bit in the `command_config` register is set to 1.

Pause Frame Termination

Pause frames are terminated within the receive engine; they are not forwarded to the receive FIFO. The reference design determines if a pause frame is valid by checking its CRC and frame length. The pause quanta in a valid pause frame is extracted and forwarded to the transmit engine. Invalid pause frames are ignored.



For the destination address, both the multicast and the unicast are taken. The frame must be valid, and be at least 64 bytes long and have a correct CRC.

The statistics counter `aPAUSEMACCtrlFramesReceive` is incremented each time a valid pause frame is received.

Payload Pad Removal

The padding removal can be optional, depending on the payload length and the value of the `PAD_EN` bit in the `command_config` register.

The reference design removes the padding, prior to sending the frames to the receive FIFO, when the `PAD_EN` bit is set to 1 and the payload length is less than the following values for the different frame types:

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

If the `PAD_EN` bit is set to 0, complete frames including the padding are forwarded to the receive FIFO.

CRC Checking

The following equation shows the CRC polynomial, as specified in *IEEE 802.3*:

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

The 32-bit CRC value occupies the FCS field with X^{32} in the least significant bit of the first byte. The CRC bits are thus received in the following order: $X^{32}, X^{30}, \dots, X^1, X^0$.

If a CRC-32 error is detected, the reference design marks the frame invalid by setting the frame status, `avl_st_rx_err`, to 1.

The CRC-32 field is forwarded to the receive FIFO if the `CRC_FWD` and `PAD_EN` bits in the `command_config` register are 1 and 0 respectively.

Frame Length Checking

The reference design checks the complete frame length to ensure that the length conforms to the following conditions:

- The length of all frame types is not less than 64 bytes.
- The length of basic frames is not greater than the maximum length specified in the `frm_length` register.
- The length of VLAN tagged frames is not greater than the maximum length specified in the `frm_length` register plus four.
- The length of stacked VLAN tagged frames is not greater than the maximum length specified in the `frm_length` register plus eight.

The reference design keeps track of the actual frame payload length as it receives a frame. The actual frame payload length is checked against the length/type or client length/type field, depending on the frame type, when the `NO_LGTH_CHECK` bit in the `command_config` register is set to 0 and a valid frame length is received for the following frame types:

- Basic frames—the length/type field is between 0x2E and 0x0600 (46 to 1,536), excluding 0x600.
- VLAN tagged frames—the client length/type field is between 0x2A and 0x0600 (42 to 1,536), excluding 0x600.
- Stacked VLAN tagged frames—the client length/type field is between 0x26 and 0x0600 (38 to 1,536), excluding 0x600.

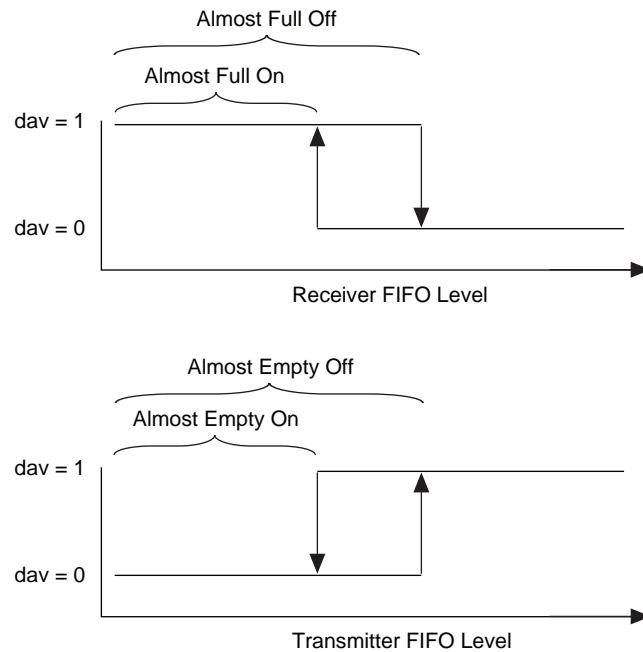
If the actual frame payload length and frame length received do not match, the reference design sets the received frame status bit, `avl_st_rx_err`, to 1, indicating a length error.

Receiver FIFO Thresholds

If you turn on **Include Avalon Streaming FIFO** in the wizard, you can configure the following receiver FIFO thresholds to dynamically change the receive FIFO operations and effectively manage potential FIFO overflow or underflow:

- Almost full off
- Almost full on
- Almost empty off
- Almost empty on

The FIFO is 64 bits wide. All the thresholds are in bytes (8 bits). [Figure 10](#) illustrates the receiver FIFO thresholds.

Figure 10. Receive FIFO Thresholds

You set the receive FIFO thresholds via registers. [Table 7](#) describes how each threshold can be used to change and manage FIFO operations.



-  Minimum value is 0; maximum value is $(N - 1) \times 8$ eight-byte words (where N is the number of bytes in the FIFO buffer). The on and off values must be different.
-  If you turn off the **Include Avalon Streaming FIFO** for the receiver there is no ready signal, so the data is sent automatically.

Table 7. Receive Threshold Registers Description

Threshold	Register Name	Description
Almost full off	rx_almost_full_off	The number of unread entries in the FIFO before the FIFO is empty. When the FIFO reaches this level, user_rx_dav is asserted.
Almost full on	rx_almost_full_on	The number of unwritten entries in the FIFO before the FIFO is full. When the FIFO reaches this level, user_rx_dav is deasserted.
Almost empty off	rx_almost_empty_off	Indicates that there are enough entries to read. When the FIFO reaches this level, avl_st_rx_dav is asserted.
Almost empty on	rx_almost_empty_on	Indicates that there are insufficient entries in the FIFO for the user application to start reading from it. When the FIFO reaches this level, avl_st_rx_dav is deasserted unless an EOP is present in the FIFO.

Link Fault Signaling

The receiver monitors the traffic for local and remote fault signaling. On receiving these faults, it requests the transmission of remote fault or idles respectively. On detection of local or remote faults, the receiver transmits remote fault or idle, which overrides the sent data.

The state machine for detecting and transmitting follows *IEEE 802.3 Clause 46* state diagram.

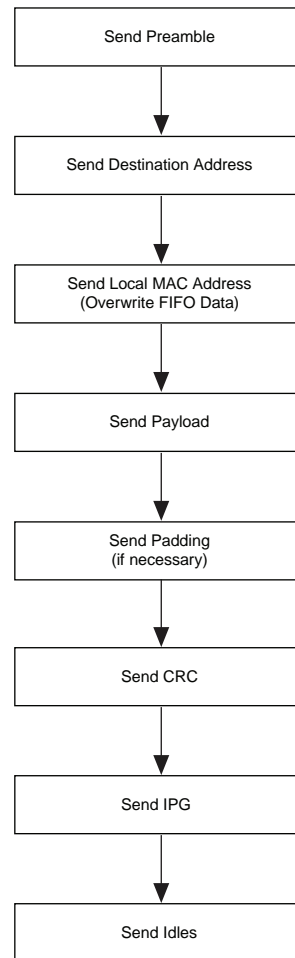
Transmitter Operation

Frame transmission starts when the transmit FIFO buffer holds enough data and initiates the following tasks:

- Generates Start, preamble, and SFD field before frame transmission
- Inserts source address if required
- Adds padding to the frame, if required
- Calculates and appends CRC-32 to the transmitted frame, if required
- Sends frame with correct interframe gap (IFG)
- Generates XOFF pause frames if the receive FIFO reports a congestion or if the `xoff_gen` signal is asserted
- Generates XON pause frames if the receive FIFO congestion condition is cleared or if the `xon_gen` signal is asserted
- Suspends Ethernet frame transmission (XOFF) if a non-zero pause quanta is received from the receive path
- Checks and generates the link fault condition

 For more information on XON and XOFF, see [“Flow Control Operation”](#) on page 23.

[Figure 11](#) shows the transmitter flow.

Figure 11. Transmitter Flow

Address Insertion

If address insertion is enabled (TX_ADDR_INS bit in the `command_config` register = 1), the source address in frames received from the transmit FIFO is replaced with the primary address.

If address insertion is disabled (TX_ADDR_INS bit in the `command_config` register = 0), the source address is forwarded to the Ethernet-side interface.

Frame Payload Padding

IEEE 802.3 defines a minimum frame length of 64 bytes. To avoid violating this specification, the reference design automatically inserts padding bytes (0x00) if it receives frames with payload length less than 46 bytes from the user application:

CRC-32 Generation

The CRC polynomial, as specified in the *IEEE 802.3*, is shown in the following equation:

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

The 32-bit CRC value occupies the FCS field with x^{31} in the least significant bit of the first byte. The CRC bits are thus transmitted in the following order: $x^{31}, x^{30}, \dots, x^1, x^0$.

IFG

The IFG configured in the `tx_ipg_length` register is maintained between transmissions. The minimum IFG can be configured to any value between 64 and 216 bit times, where 64 bit times is the time it takes to transmit 64 bits of raw data on the medium.

The deficit idle counter maintains an average IFG. The number is a multiple of 4 with a minimum of 8 and a maximum of 252. The default (IEEE required value) is 12.

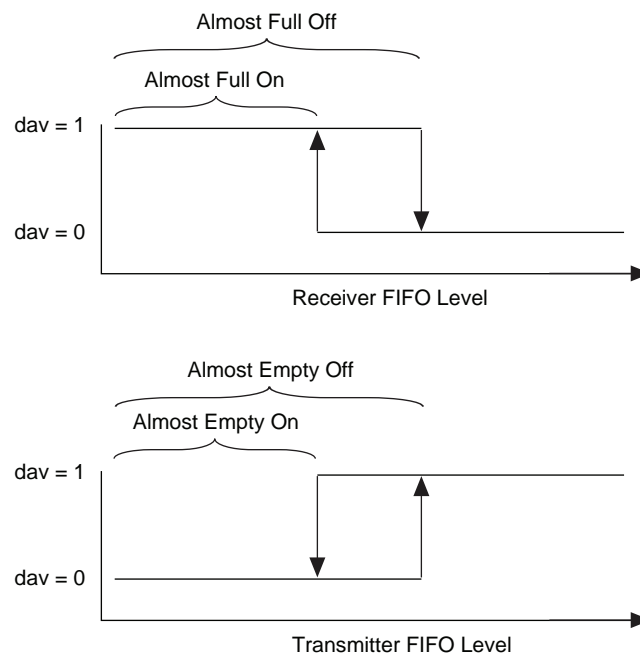
Transmit FIFO Thresholds

You can configure the following transmit FIFO thresholds to dynamically change the transmit FIFO operation and effectively manage potential FIFO overflow or underflow:

- Almost full off
- Almost full on
- Almost empty off
- Almost empty on

The FIFO is 64 bits wide. All the thresholds are in bytes (8 bits). [Figure 12](#) illustrates the transmit thresholds.

Figure 12. Transmit FIFO Thresholds



The transmit FIFO thresholds are configured via the registers. [Table 8](#) describes how you can use each threshold to change and manage FIFO operations.



-  Minimum value is 0; maximum value is $(N - 1) \times 8$ eight-byte words, (where N is the number of bytes in the FIFO buffer). The on and off values must be different.
-  If you turn off the **Include Avalon Streaming FIFO** for the transmitter, the read latency is 1. You must provide a full packet when the design indicates data is present.

Table 8. Threshold Registers Description

Threshold	Register Name	Description
Almost full off	tx_almost_full_off	The number of unread entries in the FIFO before the FIFO is empty. When the FIFO reaches this level, <code>avl_st_tx_dav</code> is deasserted.
Almost full on	tx_almost_full_on	The number of unwritten entries in the FIFO before the FIFO is full. When the FIFO reaches this level, <code>avl_st_tx_dav</code> is asserted.
Almost empty off	tx_almost_empty_off	Indicates that there are enough entries to read. When the FIFO reaches this level, <code>user_tx_dav</code> is asserted.
Almost empty on	tx_almost_empty_on	Indicates that there are insufficient entries in the FIFO for the user application to start reading from it. When the FIFO reaches this level, <code>user_tx_dav</code> is deasserted unless there is another EOP in the FIFO.

Transmit FIFO Underflow

During a frame transmission, if the transmit FIFO reaches the almost-empty threshold with no end of frame indication stored in the FIFO, the transmit control stops reading data from the FIFO and initiates the following actions:

1. The reference design indicates that the fragment transferred is not valid.
2. The reference design terminates the frame transmission.
3. After the underflow, the application completes the frame transmission.
4. The transmit control discards any new data in the FIFO until the EOP is reached.
5. The reference design starts to transfer data on the XGMII when the application sends a new frame with a start of frame indication.

Errors

The transmitter and receiver allow errors, coming in the form of an error command or due to FIFO over/underflow.

On the transmit side, if the data runs dry on the input side without an EOP, an error is sent out and the reference design pulls data until the EOP is read. If an error signal arrives alongside the data, the reference design ensures the error is propagated.

On the receive side, if an FCS error occurs, `avl_st_rx_err` goes high. Error in the reception due to error commands is signalled down towards the local interface. If you select a store forward FIFO and set bit 26 (`FIFO_ERR_DIS`) in the `command_config` register to 1, error packets are discarded.

The following errors are possible:

- Malformed frame (missing end of frame)
- Error during the packet signalled by the error command
- Incorrect FCS

- Incorrect length

Flow Control Operation

The full duplex flow control manages three congestion types:

- Remote device congestion—the remote device connected to the same Ethernet segment as the reference design reports congestion and requests the reference design to stop sending data.
- Receive FIFO congestion—when the receive FIFO reaches a user defined threshold (`rx_almost_full_on`), the reference design sends a pause frame to the remote device requesting the remote device to stop sending data.
- Local device congestion—any device connected to the reference design can request the remote device to stop data transmission, which is typically done via the host processor.

Remote Device Congestion

When the transmit control receives a valid pause quanta from the receive path, the reference design completes the transfer of the current frame and stops sending data for the amount of time specified by the pause quanta in 512 bit times increments.

Frame transmission resumes when the time specified by the quanta expires, and no new quanta value or pause frame with a quanta value set to `0x0000` is received.

Receive FIFO and Local Device Congestion

The transmit control generates pause frames at the request of the user application or the following signals derived from the receiver write signals:

- Almost full on generates XOFF
- Almost full off generates XON

You can enable or disable this behavior (bit 30 in the `command_config` register). For example, the FIFOs may be small and depending on their threshold, you may not want to always send XOFF and XOFF.

To generate an XOFF pause frame, the user application sets the XOFF_GEN bit in the `command_config` register to 1. An XOFF pause frame is generated when one of the following two conditions are met:

- The receive FIFO asserts the almost full on flag
- The XOFF_GEN bit is set to 1 in the `command_config` register

Then the XOFF frame is sent when the current frame transmission completes.

When an XOFF pause frame is generated, the pause quanta bytes P1 and P2 (see [“Pause Frames” on page 24](#)) are filled with the value configured in the `pause_quant` register. The source address is set to the primary address configured in the `mac_0` and `mac_1` registers, and the destination address is set to a fixed multicast address, `01-80-C2-00-00-01` (`0x010000c28001`).

An XON pause frame is generated when one of the following two conditions are met:

- The receive FIFO almost full off threshold is reached
- The XON_GEN bit is set to 1 in the `command_config` register.

Then the XON frame is sent when the current transmission completes.

When an XON pause frame is generated, the pause quanta (payload bytes P1 and P2) is filled with 0x0000 (zero quanta). The source address is set to the primary address configured in the `mac_0` and `mac_1` registers and the destination address is set to a fixed multicast address, 01-80-C2-00-00-01 (0x010000c28001).

Pause frames generated are compliant to the *IEEE 802.3 Annex 31A and B*.


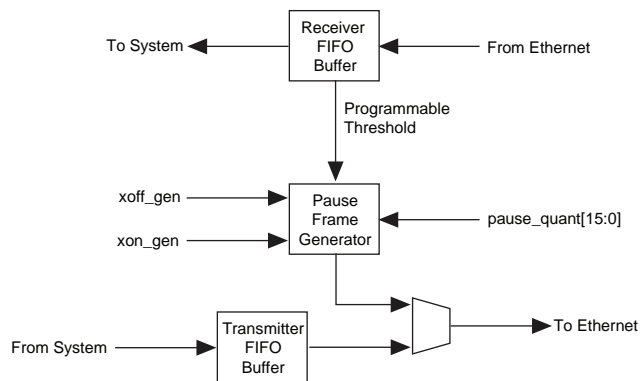

 For more information on pause frames, refer to “Pause Frames” on page 24.

Figure 13. Pause Frame Generation



 The flow control mechanism can prevent any FIFO overflow on the receive path and protects the receive FIFO. When an overflow is detected on the receive FIFO, if you select **Fill level** in the wizard, the current frame is truncated with an error indication set in the frame status, `avl_st_rx_err`. If you select a store forward FIFO and set bit 26 in the `command_config` register to 1, error packets are discarded.

Pause Frames

A pause frame is generated by the receiving device to indicate congestion to the source device. The source device should stop sending data upon receiving pause frames if it supports flow control.

Figure 14 shows the format of pause frames. The length/type field has a fixed value of 0x8808, followed by a 2-byte opcode field of 0x0001. A 2-byte pause quanta is defined in the frame payload bytes 2 (P1) and 3 (P2). The pause quanta, P1, is the most significant. A pause frame has no payload length field, and is always padded with 42 bytes of 0x00.

Figure 14. Pause Frame Format

1 bytes	Start
6 bytes	Preamble
1 bytes	SFD
6 bytes	Destination Address
6 bytes	Source Address
2 bytes	Type (0x8808)
2 bytes	Opcode (0x0001)
2 bytes	Pause Quanta ((0xP1, 0xP2))
42bytes	PAD
4 bytes	CRC

If a pause frame with a pause quanta greater than zero is received, the reference design completes the current frame transmission, and subsequently suspends data transmission for a duration specified by the pause quanta. One pause quanta fraction is equivalent to 512 bit times, which equates to $512/64$ (the width of our bus), which is the eight-clock cycle of the system clock.

Data transmission resumes when a pause quanta of zero is received.

The transmitter can insert `XON` and `XOFF` commands based on the thresholds given by the receiver FIFO.

The receiver detects the `XON` and `XOFF` commands and stops the transmission of data. The pause is triggered on one of the following address options:

- The unicast address corresponding to the MAC
- The reserved multicast address

In both cases, action is only taken between packets. If a packet is being transmitted, the action is only taken after the end of the packet.

Interfaces

This section discusses the reference design interfaces.

FIFO Interfaces

The receiver and transmitter FIFO interfaces comply with the Avalon-ST interface specification.



For more information on Avalon-ST interfaces, refer to the [Avalon Interface Specification](#).

The receiver FIFO interface has a ready fixed latency of one; the transmitter FIFO interface has a ready fixed latency of one.

Frames received on the FIFO interface do not contain any preamble or SFD bytes. These are inserted and discarded by the reference design on transmit and receive respectively.



The standard requires that the least significant byte of the address is transmitted first. For the other header fields, such as the length/type, VLAN tag, VLAN info, and pause quanta, the most significant byte is transmitted first.

The data bus is 64-bits wide and the first byte of the destination address is on bit 65 : 56.

Figure 15 shows the timing diagram for the receiver local interface (without FIFO buffer).


 For a description of the signal names, see “Signals” on page 44.

Figure 15. Local Interface Timing Diagram—Receiver

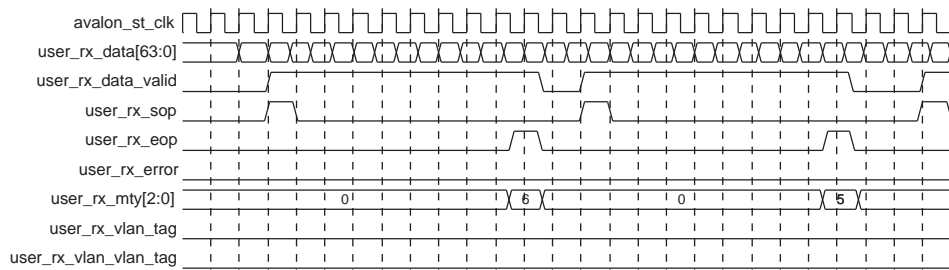


Figure 16 shows the timing diagram for the transmitter local interface (without FIFO buffer).

Figure 16. Local Interface Timing Diagram—Transmitter

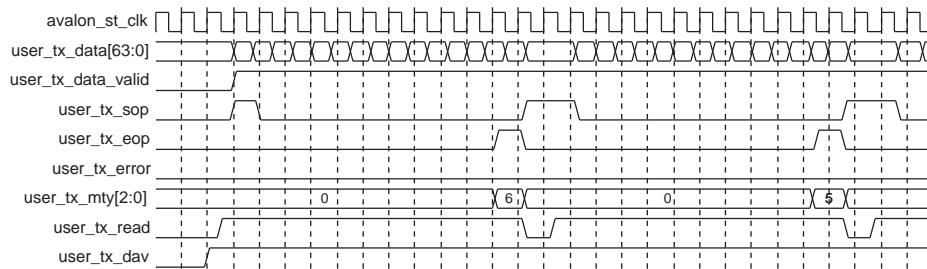


Figure 17 shows the timing diagram for the local interface (with FIFO buffer).

Figure 17. Local Interface Timing Diagram

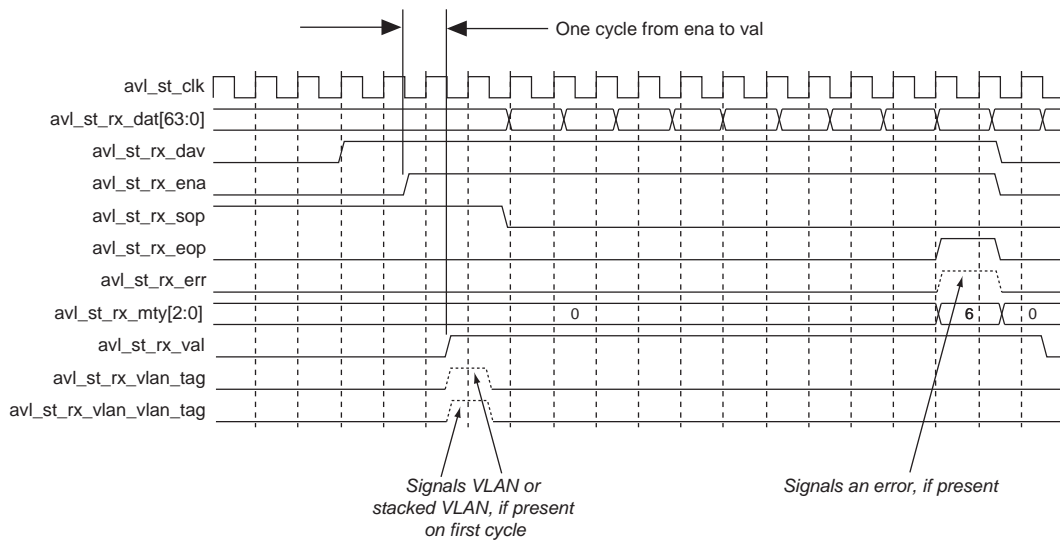


Figure 18 shows the timing diagram for Avalon-MM interface reads.

Figure 18. Avalon-MM Interface Timing Diagram—Reads

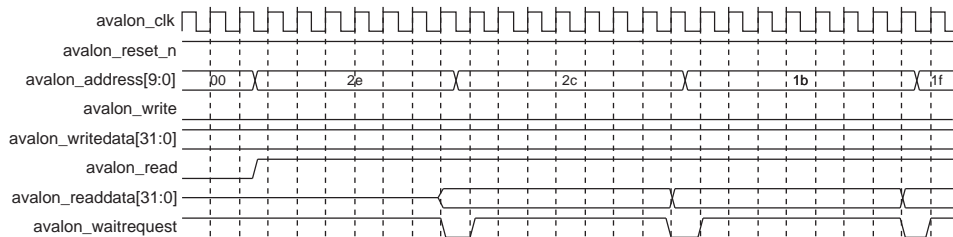
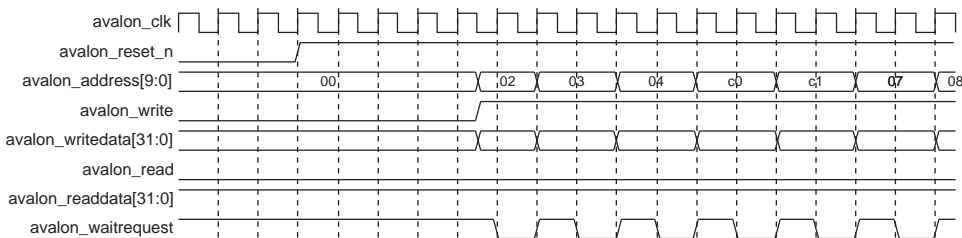


Figure 19 shows the timing diagram for Avalon-MM interface multiple writes.

Figure 19. Avalon-MM Interface Timing Diagram—Multiple Writes

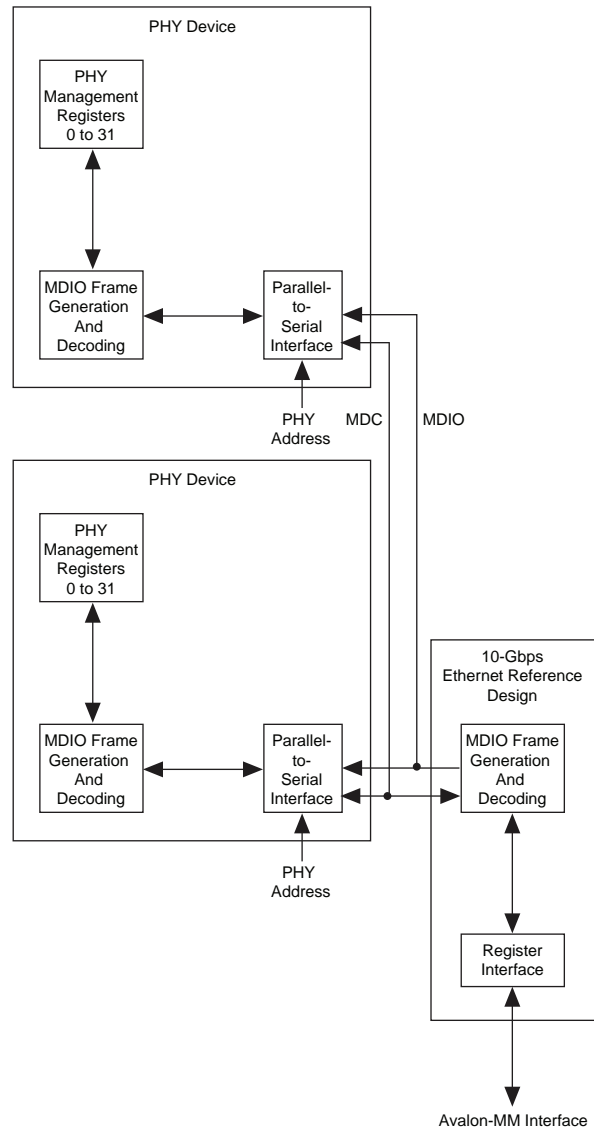


MDIO Interface

The management data input/output (MDIO) interface is a two-wire management interface. The MDIO interface implements a standardized method to access the PHY device management registers. The reference design MDIO interface supports *IEEE 802.3 Clause 22* (see [Figure 20](#)) and *Clause 45*.

The PHY device MDIO registers are mapped in the register space and can be read and written from the Avalon-MM register interface. The reference design provides the flexibility to access PHY devices with the MDIO address set to any legal value.

Figure 20. MDIO Interface (Clause 22)



MDIO Frame Format (Clause 22)

The MDIO master controller communicates with the slave PHY device using frames, see [Table 9](#). A complete frame is 64-bits long and consists of 32-bit preamble, 14-bit command, 2-bit bus direction change, and 16-bit data. Each bit is transferred on the rising edge of the MDIO clock (MDC). The PHY management interface supports the standard MDIO specification (*IEEE803.2 Clause 22*).

Table 9. MDIO Frame Formats (Read/Write)

Type	Command							
	PRE	ST	OP	PHYAD	REGAD	TA	Data	Idle
Read	1 ... 1	01	10	AAAAA	RRRRR	Z0	DDDDDDDDDDDDDDDD	Z
Write	1 ... 1	01	01	AAAAA	RRRRR	10	DDDDDDDDDDDDDDDD	Z

Table 10 describes the fields of the MDIO frame (clause 22).

Table 10. MDIO Frame Field Descriptions—Clause 22

Name	Description
PRE	Preamble. 32 bits of logical 1 sent prior to every transaction.
ST	Start indication. Standard MDIO (Clause 22): 0b01.
OP	The opcode defines whether a read or write operation is performed: <ul style="list-style-type: none"> ■ 0b10: a read operation is performed. ■ 0b01: a write operation is performed.
PHYAD	The PHY device address (PHYAD). Up to 32 devices can be addressed. For PHY device 0, the <code>addr1</code> field is set to the value configured in the <code>mdio_addr0</code> register.
REDAD	Register address. Each PHY can have up to 32 registers.
TA	Turnaround time. Two bit times are reserved for read operations to switch the data bus from write to read for read operations. The PHY device presents its register contents in the data phase and drives the bus from the 2 nd bit of the turnaround phase.
Data	16-bit data written to or read from the PHY device.
Idle	Between frames, the MDIO data signal is tristated.

MDIO Frame Format (Clause 45)

The MDIO master controller communicates with the slave PHY device using frames, see Table 11. A complete frame is 64-bits long and consists of 32-bit preamble, 14-bit command, 2-bit bus direction change, and 16-bit data. Each bit is transferred on the rising edge of the MDIO clock (MDC). The PHY management interface supports the standard MDIO specification (*IEEE803.2 Clause 45*).

Table 11. MDIO Frame Formats—Clause 45

Type	Command							
	PRE	ST	OP	PRTAD	DEVAD	TA	Address/Data	Idle
Address	1 ... 1	00	00	PPPPP	EEEE	10	AAAAAAAAAAAAAAAA	Z
Write	1 ... 1	00	01	PPPPP	EEEE	10	DDDDDDDDDDDDDDDD	Z
Read	1 ... 1	00	11	PPPPP	EEEE	Z0	DDDDDDDDDDDDDDDD	Z
Post-read-increment-address	1 ... 1	00	10	PPPPP	EEEE	Z0	DDDDDDDDDDDDDDDD	Z

Table 12 describes the fields of the MDIO frame (clause 45).

Table 12. MDIO Frame Field Descriptions—Clause 45

Name	Description
PRE	Preamble. 32 bits of logical 1 sent prior to every transaction.
ST	The start of frame for indirect access cycles is indicated by the <00> pattern. This pattern assures a transition from the default one and identifies the frame as an indirect access. Frames that contain the ST=01 pattern defined in Clause 22 are ignored by the devices specified in Clause 45.
OP	The operation code field indicates the type of transaction being performed by the frame. <ul style="list-style-type: none"> ■ 00 indicates that the frame payload contains the address of the register to access. ■ 01 indicates that the frame payload contains data to be written to the register whose address was provided in the previous address frame. ■ 11 indicates that the frame is a read operation. ■ 10 indicates that the frame is a post-read-increment-address operation.
PRTAD	The port address (PRTAD). The port address is five bits, allowing 32 unique port addresses. The first port address bit to be transmitted and received is the MSB of the address. A station management entity must have a prior knowledge of the appropriate port address for each port to which it is attached, whether connected to a single port or to multiple ports.
DEVAD	The device address is five bits, allowing 32 unique MDIO manageable devices (MMDs) per port. The first device address bit transmitted and received is the MSB of the address.
TA	The turnaround time is a 2-bit time spacing between the device address field and the data field of a management frame to avoid contention during a read transaction. For a read or post-read-increment-address transaction, both the STA and the MMD remain in a high-impedance state for the first bit time of the turnaround. The MMD drives a zero bit during the second bit time of the turnaround of a read or postread-increment-address transaction. During a write or address transaction, the STA drives a one bit for the first bit time of the turnaround and a zero bit for the second bit time of the turnaround.
Address/ Data	The address/data field is 16 bits. For an address cycle, it contains the address of the register to be accessed on the next cycle. For the data cycle of a write frame, the field contains the data to be written to the register. For a read or post-read-increment-address frame, the field contains the contents of the register. The first bit transmitted and received is bit 15.
Idle	The idle condition on MDIO is a high-impedance state. All three state drivers shall be disabled and the MMD's pull-up resistor pulls the MDIO line to a one.

MDIO Registers

The host processor can access the MDIO registers of a PHY device via an Avalon-MM interface. The PHY MDIO registers are mapped in the address space. Each PHY device has 32 registers.

The reference design supports both clause 22 and clause 45, but you cannot use both at the same time due to electrical restrictions. Only one clause should be accessed in a design.

For clause 22, follow these steps:


1. Set up the `mdio_addr0` register at address `0x03C`, where:
 - Bits 31:5 are unused
 - Bits 4:0 are PHYAD
2. Read or write to addresses `0x200` to `0x27C` for direct access to the 32 register addresses present (REGAD).

For clause 45, follow these steps:

- Set up the `mdio_addr0` register at address `0x03C`, where:
 - Bits 31:16 are the Address
 - Bits 12:8 are the PRTAD
 - Bits 4:0 are the DEVAD
- Read or write the register of the device of the port selected by reading or writing to address `0x320`.

MDIO Clock Generation

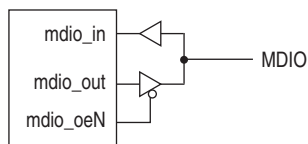
The management data clock (MDC) is generated from the Avalon-MM interface clock signal, `clk`.

 The division factor must be defined such that the MDC frequency does not exceed 2.5 MHz.

MDIO Buffer Connection

Figure 21 illustrates the buffers you can implement for the MDIO tristate bus.

Figure 21. MDIO Buffer Connection



Configuration and Statistics Interfaces

The configuration and statistics control interfaces have a register space of 256 registers, providing access to all functional blocks within the reference design.


 When the statistics counters reach their maximum value they roll over. Your software must calculate the difference between the current and the last read values to measure the change and should perform an accumulation operation in software.

Table 13 shows an overview of the register space for the reference design.

Table 13. Register Overview (Part 1 of 2)

Addr Offset	Section	Description
0x000 to 0x05C	Block configuration	Base register set to parameterize the reference design.
0x060 to 0x0DC	Statistics counters	Counters collecting traffic statistics.
0x200 to 0x27C	MDIO space 0 or PCS block configuration	MDIO registers for the first PHY device. These registers map directly to the 32 MDIO registers in a connected device. If the configuration includes the PCS function, these registers control the PCS function.
0x280 to 0x2FC	Reserved (1)	Unused.

Table 13. Register Overview (Part 2 of 2)

Addr Offset	Section	Description
0x300 to 0x31C	Addresses	Supplemental unicast addresses.
0x320 to 0x338	MDIO clause 45	Statistic counters,
0x33C	Reserved (1)	Unused.
0x340 to 0x348	Transceiver status	Transceiver status for XAUI (Stratix II GX or Stratix IV devices only).

Note to Table 13:

(1) Altera recommends that you set all bits in reserved registers to 0 and ignore them on reads.

Control Interface Register Map

This section defines the control interface register map. Table 14 shows each usable register. The following list describes the columns HW reset, SW reset and access:

- The HW reset column specifies the value after hardware reset, which is controlled by the `reset` signal.
- The SW reset column specifies the value or influence after a software reset, which is controlled by the `SW_RESET` bit in the `command_config` register.
 - “—” indicates that `reset` is not relevant to this register and has no influence.
 - “X” indicates that the value is unknown, which is typical for memory-based registers.
- The access column indicates whether you can only read a register (RO), write it (WO) or read and write it (RW).



You must perform a software reset, before you write to a register, then re-enable the design.

Table 14. Control Interface Register Map (Part 1 of 5)

Address Offset	Name	Description	Access	HW Reset	SW Reset
0x000	rev	Revision. This register is divided into two 16-bit fields: <ul style="list-style-type: none"> ■ Bits 15:0: reference design revision, set to 0x0702 ■ Bit 31:16: Customer specific revision, set to 0 during reference design configuration. This field is controlled by the parameter <code>CUST_VERSION</code> defined in the top level generated for the 10-Gbps Ethernet reference design instance. 	RO	0x00000702	—
0x004	scratch	Scratch register. Provides a memory location for user applications to test the device memory operation.	RW	0	—
0x008	command_config	Command register. The host processor uses this register to control and configure the reference design.	RW	0	bit 0 = 0 bit 1 = 0 Others not modified
0x00C	mac_0	32-bit primary address word 0—bits 0 to 31 of the primary address. Bit 0 maps to bit 0 of the address, bit 1 maps to bit 1 of the address, and so on.	RW	0	—
0x010	mac_1	32-bit primary address word 1—bits 32 to 47 of the primary address. Bits 16 to 31 are reserved. Bit 0 maps to bit 32 of the address.	RW	0	—
0x014	frm_length	14-bit maximum frame length. The receive logic uses this value to check frames. Typical value is 1518. Bits 14 to 31 are reserved.	RW	1518	—
0x018	pause_quant	16-bit pause quanta. The pause quanta is used in each pause frame sent to a remote Ethernet device, in increments of 512 Ethernet bit times. Bits 16 to 31 are reserved.	RW	0	—
0x01C	rx_almost_empty_off	12-bit receive FIFO almost empty off threshold. Bits 12 to 31 are unused.	RW	0	—

Table 14. Control Interface Register Map (Part 2 of 5)

Address Offset	Name	Description	Access	HW Reset	SW Reset
0x020	rx_almost_empty_on	12-bit receive FIFO almost empty on threshold. Bits 12 to 31 are unused.	RW	0	—
0x024	TX_almost_EMPTY_off	12-bit transmit FIFO almost empty off threshold. Bits 12 to 31 are unused.	RW	0	—
0x028	TX_almost_empty_on	12-bit transmit FIFO almost empty on threshold. Bits 12 to 31 are unused.	RW	0	—
0x02C	RX_ALMOST_full_off	12-bit receive FIFO almost-empty threshold. Bits 12 to 31 are unused.	RW	0	—
0x030	RX_almost_full_on	12-bit receive FIFO almost-full threshold. Bits 12 to 31 are unused.	RW	0	—
0x034	TX_almost_full_off	12-bit transmit FIFO almost-empty threshold. Bits 12 to 31 are unused.	RW	0	—
0x038	TX_almost_full_on	12-bit transmit FIFO almost-full threshold. Bits 12 to 31 are unused.	RW	0	—
0x03C	MDIO_ADDR0	MDIO address of PHY device 0. <ul style="list-style-type: none"> ■ 4:0 are the PHY address (clause 22) or the device address (clause 45) ■ 12:8 are the port address (clause 45 only) ■ 31:16 are the register address (clause 45 only). To read and write for clause 45, the read and write must be done at address 0x320 	RW	0	—
0x040	Unused	Unused.	—	—	—
0x044 to 0x054	Reserved	Reserved for user defined registers.	—	0	—
0x058	Reserved	Reserved for user defined registers.	RO	0	—
0x05C	tx_ipg_length	Minimum IFG. Valid values are between 8 and 252 bytes. If this register is set to an invalid value, it defaults to 12 byte-times which is a typical value of minimum IFG. Bits 5 to 31 are reserved and set to read-only value 0.	RW	12	—

Table 14. Control Interface Register Map (Part 3 of 5)

Address Offset	Name	Description	Access	HW Reset	SW Reset
0x060 0x064	aMacID	This register is wired to mac_0 and mac_1 addresses respectively.	RO	0	—
0x068	aFramesTransmittedOK	See Table 16 on page 40.	RO	0	0
0x06C	aFramesReceivedOK	See Table 16 on page 40.	RO	0	0
0x070	aFrameCheckSequenceErrors	See Table 16 on page 40.	RO	0	0
0x074	aAlignmentErrors	See Table 16 on page 40.	RO	0	0
0x078	aOctetsTransmittedOK	See Table 16 on page 40.	RO	0	0
0x07C	aOctetsReceivedOK	See Table 16 on page 40.	RO	0	0
0x080	aTxPAUSEMACCtrlFrames	See Table 16 on page 40.	RO	0	0
0x084	aRxPAUSEMACCtrlFrames	See Table 16 on page 40.	RO	0	0
0x088	ifInErrors	See Table 17 on page 41.	RO	0	0
0x08C	ifOutErrors	See Table 17 on page 41.	RO	0	0
0x090	ifInUcastPkts	See Table 17 on page 41.	RO	0	0
0x094	ifInMulticastPkts	See Table 17 on page 41.	RO	0	0
0x098	ifInBroadcastPkts	See Table 17 on page 41.	RO	0	0
0x09C	ifOutDiscards	See Table 19 on page 42.	RO	0	0
0x0A0	ifOutUcastPkts	See Table 17 on page 41.	RO	0	0
0x0A4	ifOutMulticastPkts	See Table 17 on page 41.	RO	0	0
0x0A8	ifOutBroadcastPkts	See Table 17 on page 41.	RO	0	0
0x0AC	etherStatsDropEvents	See Table 18 on page 42.	RO	0	0
0x0B0	etherStatsOctets	See Table 18 on page 42.	RO	0	0
0x0B4	etherStatsPkts	See Table 18 on page 42.	RO	0	0
0x0B8	etherStatsUndersizePkts	See Table 18 on page 42.	RO	0	0
0x0BC	etherStatsOversizePkts	See Table 18 on page 42.	RO	0	0
0x0C0	etherStatsPkts64Octets	See Table 18 on page 42.	RO	0	0
0x0C4	etherStatsPkts65to127Octets	See Table 18 on page 42.	RO	0	0
0x0C8	etherStatsPkts128to255Octets	See Table 18 on page 42.	RO	0	0
0x0CC	etherStatsPkts256to511Octets	See Table 18 on page 42.	RO	0	0
0x0D0	etherStatsPkts512to1023Octets	See Table 18 on page 42.	RO	0	0
0x0D4	etherStatsPkts1024to1518Octets	See Table 18 on page 42.	RO	0	0

Table 14. Control Interface Register Map (Part 4 of 5)

Address Offset	Name	Description	Access	HW Reset	SW Reset
0x0D8	etherStatsPkts1519toXOctets	Any frame length from 1519 to the maximum length configured in the <code>frm_length</code> register, if it is greater than 1518.	RO	0	0
0x0DC	etherStatsJabbers	Too long frames with CRC error.	RO	0	0
0x0E0	etherStatsFragments	Too short frames with CRC error.	RO	0	0
0x0E4	Reserved	Unused	RO	—	—
0x0EA	linkFaulDetect	<p>Link remote and local fault detection according to <i>IEEE 802.3ae</i>:</p> <ul style="list-style-type: none"> ■ Bit[0] = 1'b1 and indicates a local fault has been detected ■ Bit[1] = 1'b1 and indicates a remote fault has been detected ■ Bit [1:0] = 2'b00 and indicates link is OK <p>All other bits are currently unused.</p>	RO	—	—
0x200 to 0x27C	PHY Device 0 Internal Registers	<p>Registers 0 to 31 within PHY device 0 connected to the MDIO PHY management interface. Reading or writing immediately causes a corresponding MDIO transaction to read or write the underlying PHY device register. For configurations that include Ethernet and PCS blocks, the internal PCS is always device 0. In this case, reading and writing does not require an MDIO module as the application reads/writes directly to the PHY registers through the register interface.</p> <p>The register at address offset 0x200 corresponds to register 0 of PHY device 0. The register at address offset 0x204 corresponds to register 1 of PHY device 0.</p> <p>For all registers, bits 15:0 are significant. Bits 31:16 should be written with 0 and ignored on read.</p>	RW	—	—
0x280 to 0x2FC	Reserved	Reserved.	—	—	—

Table 14. Control Interface Register Map (Part 5 of 5)

Address Offset	Name	Description	Access	HW Reset	SW Reset
0x300	smac_0_0	Supplemental address 0, bits 31:0. Register bit 0 maps to bit 0 of the address, bit 1 maps to bit 1 of the address, and so on.	RW	0	—
0x304	smac_0_1	Supplemental address 0, bits 47:32. Register bit 0 maps to bit 32 of the address. Register bits 30:16 are reserved. register bit 31 enables address: <ul style="list-style-type: none"> ■ 0 to disable ■ 1 to enable 	RW	0	—
0x320	MDIO clause 45	—	RW	—	—
0x324	FrameTXok	Packet and byte/octet counts for both transmitter and receiver are 64 bit wide. To access this register, you must access the top 32 bits of the 64 bits. When you read the lower 32 bits then the upper 32 bits, the upper 32 bits may have incremented by 1. Ensure the software can workaround this behavior.	—	—	—
0x328	FrameRXok		—	—	—
0x32C	OctetTxOK		—	—	—
0x330	OctetRXok		—	—	—
0x334	etherInOctets		—	—	—
0x338	etherInPkt		—	—	—
0x33C	Reserved	Reserved.	—	—	—
0x340	GXB status0	Status of the transceivers (XAUI only). <ul style="list-style-type: none"> ■ 0 pll_locked ■ 1 rx_channelaligned ■ 7:4 rx_freqlocked ■ 11:8 rx_pll_locked 	RO	—	—
0x344	GXB status1	Status of the transceivers (XAUI only). <ul style="list-style-type: none"> ■ 7:0 rx_disperr ■ 15:8 rx_errdetect ■ 23:16 rx_patterndetect ■ 31:24 rx_syncstatus 	RO	—	—
0x348	Shadow MDIO	Stores the last read value from the MDIO access.	—	—	—

Command_Config Register

Table 15 describes the function of each bit and field in the `command_config` register.

Table 15. Command_Config Register Bit Descriptions (Part 1 of 2)

Bit(s)	Bit Name	Access	Description
0	TX_ENA	RW	Transmit enable. Setting this bit to 1 enables the transmit datapath. This bit is cleared following a hardware or software reset. See the SW_RESET bit description.
1	RX_ENA	RW	Receive enable. Setting this bit to 1 enables the receive datapath. This bit is cleared following a hardware or software reset. See the the SW_RESET bit description.
2	XON_GEN	RW	Pause frames generation. When this bit is set to 1, the reference design generates a pause frame with a pause quanta of 0, independent of the receive FIFO status.
3	Reserved	—	Self clear.
4	PROMIS_EN	RW	Promiscuous enable. Setting this bit to 1 enables promiscuous operation whereby the destination address of the receive frame is not checked.
5	PAD_EN	RW	Pad enable. Setting this bit to 1 enables pad removal in receive frames. The reference design removes receive frames padding before forwarding the frames to the user application. Transmit frames are always padded and this bit has no effect.
6	CRC_FWD	RW	Receive CRC forwarding. <ul style="list-style-type: none"> ■ If this bit is set to 1, the reference design forwards the CRC field to the user application. ■ If this bit is set to 0, the reference design removes the CRC field from the frame before forwarding the frame to the user application. ■ This bit is ignored if the PAD_EN bit is 1. In this case, the reference design checks the CRC field and removes it from the frame before forwarding the frame to the user application.
7	PAUSE_FWD	RW	Receive pause frame forwarding. Terminates or forwards pause frames. <ul style="list-style-type: none"> ■ If this bit is set to 1, the reference design forwards pause frames to the user application. ■ If this bit is set to 0, the reference design terminates and discards pause frames.
8	PAUSE_IGNORE	RW	Ignore pause frame quanta. <ul style="list-style-type: none"> ■ Setting this bit to 1 causes the reference design to ignore received pause frames. ■ Setting this bit to 0 causes the transmit process to stop for an amount of time specified in the pause quanta within the pause frame.
9	TX_ADDR_INS	RW	Set address on transmit. <ul style="list-style-type: none"> ■ If this bit is set to 1, the reference design overwrites the transmit frame source address with the address configured in the mac_0 and mac_1 registers, or in any of the supplemental address registers. ■ If this bit is set to 0, the reference design does not modify the source address.
12:10	Reserved	—	Reserved.
13	SW_RESET	RW	Software reset command. Setting this bit to 1 causes the reference design to disable the transmit and receive logic, flush the receive FIFO, and reset the statistics counters. This bit is automatically cleared when the software reset sequence completes.

Table 15. Command_Config Register Bit Descriptions (Part 2 of 2)

Bit(s)	Bit Name	Access	Description
14	Reserved	—	Reserved.
15	LOCAL_LOOP_ENA	RW	<p>Local loopback enable. Setting this bit to 1 enables a loopback. Frames sent through the transmitter interface are looped back into the receiver interface.</p> <p>The transition from straight-through to loop-back and from loop-back to straight-through mode only occurs between frames. If the reference design is transmitting a packet, the transition occurs after the EOP and when the frame is complete without corrupting any processed frame.</p> <p>Only idle characters are transmitted to the Ethernet when in local loopback.</p>
21:16	Reserved	—	Reserved.
22	XOFF_GEN	RW	Pause frame generation. If this bit is set to 1, the reference design generates a pause frame with the pause quanta set to the value configured in the <code>pause_quant</code> register, independent of the receive FIFO status.
23	CNTL_FRM_ENA	RW	<p>Receive control frame enable bit self clear.</p> <ul style="list-style-type: none"> ■ If this bit is set to 1, the receive control frames with any opcode other than 0x0001 are accepted and forwarded to the Avalon-ST interface. ■ If this bit is set to 0, the receive control frames with any opcode other than 0x0001 are discarded.
24	NO_LGTH_CHECK	RW	<p>Receive payload length check disable.</p> <ul style="list-style-type: none"> ■ If this bit is set to 0, the reference design checks the actual payload length of received frames against the length/type field in the received frames. ■ No checking is done if this bit is set to 1.
25	Reserved	—	Reserved.
26	FIFO_ERR_DISC	RW	Enable discard in FIFO. In store forward mode only, discards error and overflow frames if set to 1. Default value is 0.
27	LINE_LOOP_ENA	RW	<p>Line loopback enable. Set to 1 to loopback the receiver traffic onto the transmitter path. Default value is 0.</p> <p>To get correct loop-back frames, you should first stop transmitting data, put the reference design into line loopback mode and then resume transmission.</p>
28	BROAD_FILTER_ENA	RW	Broadcast filtering enable. 0 lets the receive broadcast frames through; 1 filters them out. Default value is 0.
29	INS_CRC_ERR	RW	Insert one CRC error in the next transmit frame. For a generate fault, set to 1. An error is inserted in the next (or the one after due to latency to propagate the error) packet. The bit self clears.
30	NO_PAUSE_FIFO	RW	Disable FIFO pause. When set to 1, the FIFO does not trigger the generation of pause frames on the transmitter path. Otherwise, pause frames are triggered depending on thresholds.
31	CNT_RESET	WC	Self-clearing counter reset command. Setting this bit to 1 clears the statistics counters. This bit is automatically cleared when the counter reset sequence is completed.

Software Reset

A software application can reset the reference design by setting the SW_RESET bit in the command_config register to 1. During a software reset, the reference design clears all statistics registers, flushes the receive FIFO, and disables the transmitter and receiver by setting the TX_ENA and RX_ENA bits in the command_config register to 0.

The value of configuration registers, such as the address and FIFO thresholds are preserved. The SW_RESET bit is cleared automatically when the software reset ends. For more information about the reset signal, refer to [“Command_Config Register” on page 37](#).

Statistics Block

The statistics block accumulates statistics required in *IEEE802.3 Basic, Mandatory and Recommended Management Information Packages, IEEE 802.3ah, Clause 30*.

In addition, the reference design provides all signals to generate the applicable objects of the *Management Information Base (MIB, MIB-II)* according to *IETF RFC2665 and Remote Network Monitoring (RMON)* according to *IETF RFC 2819 for SNMP Managed Environments*.

The statistics block implements *RMON (RFC 2819), MIB (RFC 3635), and MIB (RFC 2863)*.

IEEE 802.3 Management Packages


[Table 16](#) lists the resources available to implement the *IEEE 802.3* mandatory management packages defined in the *Ethernet Standard 802.3 Clause 30* for the managed objects oMacEntity and oPauseEntity. [Table 17](#) mentions objects and attributes only applicable to the reference design; other objects and attributes are derived from your application or higher layers.

Table 16. IEEE 802.3 oEntity and oPauseEntity Managed Object Support (Part 1 of 2)

IEEE802.3 Attribute	IEEE Management Packages			Description
	Basic	Mandatory	Recommended (Optional)	
oEntity Managed Object Support				
aMACID	X	—	—	The addresses.
aFramesTransmittedOK	—	X	—	Number of frames transmitted without error including pause frames.
aFramesReceivedOK	—	X	—	Number of frames received without error including pause frames.
aFrameCheckSequence Errors	—	X	—	Number of frames received with a CRC error.
aAlignmentErrors	—	X	—	Frame received with an alignment error.
aOctetsTransmittedOK	—	—	X	Sum of payload and padding octets of frames transmitted without error.
aOctetsReceivedOK	—	—	X	Sum of payload and padding octets of frames received without error.
oPauseEntity Managed Object Support				

Table 16. IEEE 802.3 oEntity and oPauseEntity Managed Object Support (Part 2 of 2)

IEEE802.3 Attribute	IEEE Management Packages			Description
	Basic	Mandatory	Recommended (Optional)	
aPAUSEMACCtrlFrames Transmitted	—	—	X	Number of transmitted pause frames.
aPAUSEMACCtrlFrames Received	—	—	X	Number of received pause frames.

 For more information on attributes and objects, refer to *IEEE 802.3 Standard Clause 30*.

IETF Management Information Base—(MIB, MIB-II) Objects Support

The *Internet Engineering Task Force (IETF) Request for Comments 2665 (RFC2665)* defines the Management Information Base (MIB, MIB-II) objects for the Ethernet-like interface types. *RFC 2665* details the MIB (MIB-II) objects for Ethernet interfaces, which are defined in a more generic manner within *RFC 2863*.

Table 17. IETF MIB (MIB-II) Objects Support

MIB Object Name	Description
ifInUcastPkts	Number of valid received unicast frames.
ifInMulticastPkts	Number of valid received multicast frames (without pause).
ifInBroadcastPkts	Number of valid received broadcast frames.
ifOutUcastPkts	Number of valid transmitted unicast frames.
ifOutMulticastPkts	Number of valid transmitted multicast frames.
ifOutBroadcastPkts	Number of valid transmitted broadcast frames.
ifInErrors	Number of frames received with error: <ul style="list-style-type: none"> ■ FIFO overflow error ■ CRC error ■ Length error ■ Alignment error
ifOutErrors	Number of frames transmitted with error: <ul style="list-style-type: none"> ■ FIFO overflow error ■ FIFO underflow error ■ User application defined error

IETF Remote Network Monitoring Support

The *IETF RFC 2819* defines objects for managing remote network monitoring devices. These objects are usually implemented in a dedicated device (monitor/probe) for traffic monitoring and analysis within a network segment. Such a probe usually samples the values in a periodic manner to give relative usage estimations rather than absolute values. [Table 18](#) lists the defined objects. The remote monitoring (RMON) MIB counts good and bad packets, defined as:

- Good packets (valid frames): Good packets are error-free packets that have a valid frame length. A valid frame length is defined as between 64 bytes long and the value set in the `frm_length` register. The length does not include framing bits (preamble, SFD) but includes the FCS field.
- Bad packets (invalid frames): Bad packets are packets that have proper framing and are therefore recognized as packets, but contain errors within the packet or have an invalid length. On the Ethernet, bad packets have a valid preamble and SFD, but have a bad CRC, or are either shorter than 64 bytes or longer than and the value set in the `frm_length` register.

Table 18. IETF RMON MIB Object Support

MIB Object Name	Support
<code>etherStatsDropEvents</code>	Counts the number of dropped packets due to internal errors of the client. Occurs when FIFO overflow condition persists.
<code>etherStatsOctets</code>	Total number of bytes received. Good and bad frames.
<code>etherStatsPkts</code>	Total number of packets received. Counts good and bad packets.
<code>etherStatsUndersizePkts</code>	Number of packets received with less than 64 bytes.
<code>etherStatsOversizePkts</code>	Incremented with each well-formed packet that exceeds the valid maximum programmed frame length.
<code>etherStatsPkts64Octets</code>	Incremented when a packet of 64 bytes length is received (good and bad frames are counted).
<code>etherStatsPkts65to127Octets</code>	Frames (good and bad) with 65 to 127 bytes.
<code>etherStatsPkts128to255Octets</code>	Frames (good and bad) with 128 to 255 bytes.
<code>etherStatsPkts256to511Octets</code>	Frames (good and bad) with 256 to 511 bytes.
<code>etherStatsPkts512to1023Octets</code>	Frames (good and bad) with 512 to 1023 bytes.
<code>etherStatsPkts1024to1518Octets</code>	Frames (good and bad) with 1,024 to 1,518 bytes.
<code>etherStatsPkts1519toXOctets</code>	Frames (good and bad) with 1,519 to maximum frame size defined address.

Software Derived MIB Objects

To extend the management information base, the following counters and objects can be derived by the management or driver software, based on the counters available.

Table 19 describes three derived IETF MIB (MIB-II) objects.

Table 19. Derived IETF MIB (MIB-II) Objects

MIB Object	Description
<code>ifInOctets</code>	Sum of bytes received except preamble (for example, header, payload, pad and FCS) of all valid received frames. = $18 \times aFramesReceivedOK + aOctetsReceivedOK$
<code>ifOutOctets</code>	Sum of bytes transmitted except preamble (for example, header, payload, pad and FCS) of all valid transmitted frames. = $18 \times aFramesTransmittedOK + aOctetsTransmittedOK$
<code>ifOutDiscards</code>	Not applicable. The reference design does not discard frames that you write into the FIFO. If a higher layer discards frames to be transmitted it implements this counter.

Table 20 describes three derived IETF RMON MIB objects.

Table 20. Derived IETF RMON MIB Objects

Object	Support
etherStatsBroadcastPkts	Any valid frame with Broadcast address: = ifInBroadcastPkts
etherStatsMulticastPkts	Any valid multicast frame, including pause frames: = ifInMulticastPkts + aPAUSEMACCtrlFramesReceived
etherStatsCRCAlignErrors	Incremented when frames of correct length but with CRC error are received: = aFrameCheckSequenceErrors

XAUI Receive

This section describes the XAUI receive operation, which includes comma detection, decoding, de-encapsulation, synchronization, and carrier sense.

Comma Detection

Ten-bit data received from PMA devices may not align on a valid 10-bit character. The comma detection function searches for the 10-bit encoded comma character, K28.1/K28.5/K28.7, in consecutive samples received from PMA devices. When the K28.1/K28.5/K28.7 comma code group is detected, the stream is realigned on a valid 10-bit character boundary. The aligned stream can subsequently be decoded with a standard 8b/10b decoder.

The comma detection function restarts the search for a valid comma character if the receive synchronization state machine loses the link synchronization.

8b/10b Decoding

The 8b/10b decoder checks the DC balancing (disparity check) and produces a decoded 8-bit stream of data for the frame de-encapsulation function.

Frame De-encapsulation

The frame de-encapsulation state machine detects the start of frame when the /I/ /S/ sequence is received. The frame bytes are decoded and transmitted to the reference design. The /T/ /R/ /R/ or the /T/ /R/ sequence is decoded as an EOP indication for the reference design.

The reception of a /V/ character is decoded as a frame error indication for the reference design. A wrong carrier is decoded when a sequence different from /I/ /I/ (Idle) or /I/ /S/ (Start of Frame) is detected.

During frame reception, the de-encapsulation state machine checks for invalid characters. If invalid characters are detected, the de-encapsulation state machine indicates an error to the reference design.

Synchronization

The link synchronization constantly monitors the decoded data stream and determines if the underlying receive channel is ready for operation. The link synchronization state machine acquires link synchronization if three code groups with comma are received consecutively without error.

Once link synchronization is acquired, the link synchronization state machine counts the number of invalid characters received. The state machine increments an internal error counter for each invalid character received and incorrectly positioned comma character. The internal error counter is decremented when four consecutive valid characters are received. When the counter reaches 4, the link synchronization is lost.

XAUI Transmit

This section describes the XAUI transmit operation, which includes frame encapsulation and encoding.

Frame Encapsulation

During transmission, the PCS function encapsulates frames according to the specification in *IEEE 802.3 Clause 36*. The first byte of the preamble in the frame is replaced with the start of frame /S/ symbol. Following the insertion on /S/, all of the frame is encoded with standard 8B/10B encoded characters. After the last FCS byte, the EOP /T/ /R/ /R/ or the /T/ /R/ sequence is inserted. The selection of the end packet sequence is based on odd/even number of character transmission. Between frames, /I/ symbols are transmitted.

If a frame is received from the reference design with an error indication, the encapsulation function inserts a /V/ character to encode an error that can be decoded by the remote end PHY device.

8b/10b Encoding

The 8B/10B encoder maps 8-bit words to 10-bit symbols to generate a DC balanced stream with a maximum run length of 5.

Signals

This section describes all interface signals.

Clocks

The reference design has a single clock domain—both the receiver and the transmitter run on the same system clock. You can share this clock with other components on the device but it must be tightly controlled to the exact required frequency of 156.25MHz.

Table 21 shows the clocks.

Table 21. Clocks (Part 1 of 2)

Name	Description
sysclk	156.25-MHz system clock for the state machines and datapath. Can be shared with other 10-Gbps Ethernet reference designs in the same device.
avalon_clk	Avalon-MM clock that controls the Avalon-MM bus (for the statistics and configuration). This clock can be shared with other modules, an SOPC system, or can be tied to the system clock.
avl_st_clk	Avalon-ST clock input for the datapath, which can be different if a FIFO is present. This clock can be tied to the system clock.
xgmii_rx_clk xgmii_tx_clk	XGMII clocks. To capture the data coming on the XGMII receive interface, a PLL and local clock is required. The incoming clock is shifted by 90 degrees to capture the data. The xgmii_tx_clk clock is the same as sysclk.

Table 21. Clocks (Part 2 of 2)

Name	Description
serdes_sysclk	XAUI clock. When using the XAUI block, the transceiver module has its own PLL to derive the required output clocks. You can feed both the transmit and receive data clock, which are connected to the system clock, and the clock domain crossing from network clock to system clock occurs within the transceiver block.
phy_mdc	MDIO clock. The MDIO clock runs very slowly (less than 2.5 MHz) and is derived from the Avalon-MM clock. The MDIO clock cannot be shared between modules.
cal_blk_clk	Serializer/deserializer (SERDES) calibration clock.

Figure 22 shows the clocks with XGMII; Figure 22 shows the clocks with XAUI.

Figure 22. Clocks—XGMII

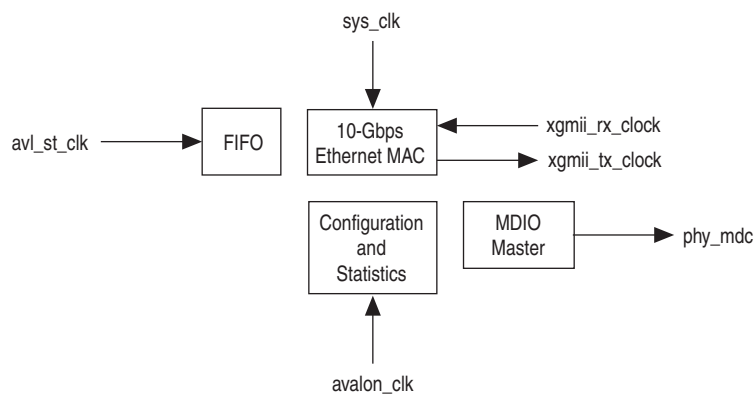
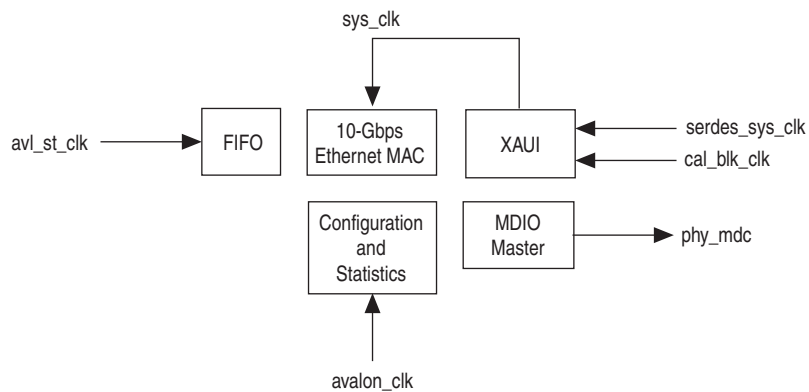



Figure 23. Clocks—XAUI



Control Interface Signals

The control interface is an Avalon Memory Mapped (Avalon-MM) slave port. This interface controls both the Ethernet and PCS blocks in the reference design. The slave port has the following properties:

- 32-bit readdata and writedata signals
- 8-bit address signal
- Variable wait states

 For more information on the Avalon-MM interface, refer to the *Avalon Interface Specifications*.

The 8-bit address provides access to a register space of 256, 32-bit registers. The complete register map is described in [Table 13 on page 31](#).

[Table 22](#) describes the signals that comprise the register interface.

Table 22. Register Interface Signals

Signal Name	Avalon-MM Signal Type	Direction	Description
avalon_write	write	In	Register write enable.
avalon_read	read	In	Register read enable.
avalon_address(9:0)	address	In	32-bit word-aligned register address.
avalon_writedata(31:0)	writedata	In	Register write data. Bit 0 is the least significant bit.
avalon_readdata(31:0)	readdata	Out	Register read data. Bit 0 is the least significant bit.
avalon_waitrequest	waitrequest	Out	Register interface busy. Asserted (1) during register read or register write access. Set to 0 to indicate the completion of the current register access.

Resets

[Table 23](#) describes the reset signals.

Table 23. Reset Signal

Signal Name	Direction	Description
serdes_reset_n	In	XAUI reset.
reset_n	In	Global asynchronous reset.
avalon_reset_n	In	Avalon-MM interface reset.
avalon_st_reset_n	In	Avalon-ST interface reset.

PHY Interface Signals

[Table 24](#) shows the XGMII signals.

Table 24. XGMII Signals

Name	Direction	Description
xgmii_rx_clk	Input	156.25-MHz input clock for the data.
xgmii_rx_data(31:0)	Input	Data input.
xgmii_rx_ctrl(3:0)	Input	Control input alongside data.
xgmii_tx_clk	Output	156.25-MHz clock output for the data (suing both edges).
xgmii_tx_data(31:0)	Output	Data output.
xgmii_tx_ctrl(3:0)	Output	Control output alongside data.

[Table 25](#) shows the XAUI signals.

Table 25. XAUI Signals

Name	Direction	Description
<code>xalui_rx_data(3:0)</code>	Input	Data input at 3.125-Gbps per lane.
<code>xalui_tx_data(3:0)</code>	Output	Data output at 3.125-Gbps per lane.
<code>cal_blk_clk</code>	Input	Calibration clock for the SERDES. Maximum frequency depends on device, but usually under 50 MHz.
<code>serdes_sysclk</code>	Output	Clock derived from the PLL of the SERDES.

Local Interface Signals

In the wizard, you can turn on **Include Avalon-ST FIFO buffer** for the local interface. Both interfaces (with or without FIFO buffer) offer Avalon-ST interfaces.

The optional FIFO buffer provides more flexibility. It is 64-bits wide (plus any bits used for control but less than 72 bits).



You can also enter the FIFO size.

The local interface is implemented as two Avalon Streaming (Avalon-ST) ports. An Avalon-ST source port provides an interface to the receive FIFO; an Avalon-ST sink port provides an interface to the transmit FIFO. There are a few additional signals that are not associated with either Avalon-ST port.



For more information on the Avalon-ST interface protocol, refer to the *Avalon Interface Specification*.

When instantiating the reference design, the Avalon-ST signals appear at the top-level of the variant HDL file, and you must manually connect them.

Table 26 lists the local interface signals (without FIFO buffer).

Table 26. Local Interface Signals (without FIFO Buffer) (Part 1 of 2)

Signal Name	Direction	Avalon-ST Type	Description
<code>user_rx_data(63:0)</code>	Output	data	Receive data.
<code>user_rx_data_valid</code>	Output	valid	Receive data valid. Asserted (set to 1) by the reference design to indicate that data is valid on <code>user_rx_data</code> , <code>user_rx_sop</code> , <code>user_rx_eop</code> , and <code>user_rx_error</code> .
<code>user_rx_sop</code>	Output	startofpacket	Receive SOP. Set to 1 when the first byte or word of a frame is driven on <code>user_rx_data</code> .
<code>user_rx_eop</code>	Output	endofpacket	Receive EOP. Set to 1 when the last byte or word of frame data is driven on <code>user_rx_data</code> .
<code>user_rx_error</code>	Output	error	Receive error. Asserted with the final byte in the frame to indicate that an error was detected when receiving the frame.
<code>user_rx_mty(2:0)</code>	Output	empty	Indicates non-valid bytes in cycle containing EOP.
<code>user_rx_vlan_tag</code>	Output	export	VLAN tag.
<code>user_tx_data(63:0)</code>	Input	data	Transmit data.

Table 26. Local Interface Signals (without FIFO Buffer) (Part 2 of 2)

Signal Name	Direction	Avalon-ST Type	Description
user_tx_data_valid	Input	valid	Transmit data write enable. Asserted by the transmit application to write data to the FIFO buffer. Indicates that user_tx_data, user_tx_sop, user_tx_eop are valid.
user_tx_sop	Input	startofpacket	Transmit SOP. Set to 1 when the first byte in the frame (the first byte of the destination address) is driven on user_tx_data.
user_tx_eop	Input	endofpacket	Transmit EOP. Set to 1 when the last byte in the frame (the last byte of the FCS field) is driven on user_tx_data.
user_tx_error	Input	error	Transmit frame error. Asserted with the final byte in the frame to indicate that the transmitted frame is invalid. When user_tx_error is asserted, the frame is transmitted to the XGMII with an error.
user_tx_mty(2:0)	Input	empty	Indicates non-valid bytes in cycle containing EOP.
user_tx_read	Output	ready	Ready. Asserted by the reference design to indicate that it is ready to accept data from the user application.
user_tx_dav	Input	export	Data available—a threshold or EOP is present.

Table 27 lists all the local interface signals (with a FIFO buffer).

Table 27. Local Interface Signals (with FIFO Buffer) (Part 1 of 2)

Signal Name	Direction	Avalon-ST Type	Description
avl_st_rx_dat(63:0)	Output	data	Receive data.
avl_st_rx_dav	Output	export	Read data available. Indicates FIFO threshold is reached.
avl_st_rx_ena	Input	ready	User ready. Asserted by the user to indicate that you are ready to accept data from the reference design.
avl_st_rx_sop	Output	startofpacket	Receive SOP. Set to 1 when the first byte or word of a frame is driven on avl_st_rx_dat.
avl_st_rx_eop	Output	endofpacket	Receive EOP. Set to 1 when the last byte or word of frame data is driven on avl_st_rx_dat.
avl_st_rx_err	Output	error	Frame error.
avl_st_rx_mty(2:0)	Output	empty	Indicates non-valid bytes in cycle containing EOP.
avl_st_rx_val	Output	valid	Data and SOP and EOP are valid.
avl_st_rx_vlan_tag	Output	export	Indicates that the packet contains a VLAN tag. The signal is active during SOP.
avl_st_rx_vlan_vlan_tag	Output	export	Indicates that the packet contains a stacked VLAN tag. The signal is active during SOP.
avl_st_tx_dat(63:0)	Input	data	Transmit data.
avl_st_tx_dav	Output	ready	Indicates space in FIFO in transmitter.

Table 27. Local Interface Signals (with FIFO Buffer) (Part 2 of 2)

Signal Name	Direction	Avalon-ST Type	Description
avl_st_tx_ena	Input	valid	Transmit data write enable. Asserted by the transmit application to write data to the FIFO buffer. Indicates that avl_st_tx_dat, avl_st_tx_sop, avl_st_tx_eop are valid.
avl_st_tx_sop	Input	startofpacket	Transmit SOP. Set to 1 when the first byte in the frame (the first byte of the destination address) is driven on avl_st_tx_dat.
avl_st_tx_eop	Input	endofpacket	Transmit EOP. Set to 1 when the last byte in the frame (the last byte of the FCS field) is driven on avl_st_tx_dat.
avl_st_tx_err	Input	error	Transmit frame error. Asserted with the final byte in the frame to indicate that the transmitted frame is invalid. When avl_st_tx_err is asserted, the frame is transmitted to the XGMII with an error.
avl_st_tx_mty(2:0)	Input	empty	Indicates non-valid bytes in cycle containing EOP.

MDIO Signals

Table 28 lists the MDIO signals.

Table 28. MDIO Signals

Signal Name	Direction	Description
phy_mdio_in	Input	Management data input.
phy_mdio_out	Output	Management data output.
phy_mdio_oen	Output	Management data active low output enable.

Software Programming Interface

This section describes the software programming interface.

Constants

Table 29 shows all constants defined for MAC register manipulation.

Table 29. Constants Mapping

Constant	Value	Description
ALT_ETH_10G_CMD_TX_ENA_OFST	0	Configures the TX_ENA bit.
ALT_ETH_10G_CMD_TX_ENA_MSK	0x1	
ALT_ETH_10G_CMD_RX_ENA_OFST	1	Configures the RX_ENA bit.
ALT_ETH_10G_CMD_RX_ENA_MSK	0x2	
ALT_ETH_10G_CMD_XON_GEN_OFST	2	Configures the XON_GEN bit.
ALT_ETH_10G_CMD_XON_GEN_MSK	0x4	
ALT_ETH_10G_CMD_PROMIS_EN_OFST	4	Configures the PROMIS_EN bit.
ALT_ETH_10G_CMD_PROMIS_EN_MSK	0x10	

Table 29. Constants Mapping

Constant	Value	Description
ALT_ETH_10G_CMD_PAD_EN_OFST	5	Configures the PAD_EN bit.
ALT_ETH_10G_CMD_PAD_EN_MSK	0x20	
ALT_ETH_10G_CMD_CRC_FWD_OFST	6	Configures the CRC_FWD bit.
ALT_ETH_10G_CMD_CRC_FWD_MSK	0x40	
ALT_ETH_10G_CMD_PAUSE_FWD_OFST	7	Configures the PAUSE_FWD bit.
ALT_ETH_10G_CMD_PAUSE_FWD_MSK	0x80	
ALT_ETH_10G_CMD_PAUSE_IGNORE_OFST	8	Configures the PAUSE_IGNORE bit.
ALT_ETH_10G_CMD_PAUSE_IGNORE_MSK	0x100	
ALT_ETH_10G_CMD_TX_ADDR_INS_OFST	9	Configures the TX_ADDR_INS bit.
ALT_ETH_10G_CMD_TX_ADDR_INS_MSK	0x200	
ALT_ETH_10G_CMD_SW_RESET_OFST	13	Configures the SW_RESET bit.
ALT_ETH_10G_CMD_SW_RESET_MSK	0x2000	
ALT_ETH_10G_CMD_LOOP_ENA_OFST	15	Configures the LOOP_ENA bit.
ALT_ETH_10G_CMD_LOOP_ENA_MSK	0x8000	
ALT_ETH_10G_CMD_XOFF_GEN_OFST	22	Configures the XOFF_GEN bit.
ALT_ETH_10G_CMD_XOFF_GEN_MSK	0x400000	
ALT_ETH_10G_CMD_CNTL_FRM_ENA_OFST	23	Configures the CNTL_FRM_ENA bit.
ALT_ETH_10G_CMD_CNTL_FRM_ENA_MSK	0x800000	
ALT_ETH_10G_CMD_NO_LGTH_CHECK_OFST	24	Configures the NO_LGTH_CHECK bit.
ALT_ETH_10G_CMD_NO_LGTH_CHECK_MSK	0x1000000	
ALT_ETH_10G_CMD_FIFO_ERR_DISC_OFST	26	Configures the FIFO_ERR_DISC bit.
ALT_ETH_10G_CMD_FIFO_ERR_DISC_MSK	0x4000000	
ALT_ETH_10G_CMD_LINE_LOOP_ENA_OFST	27	Configures the LINE_LOOP_ENA bit.
ALT_ETH_10G_CMD_LINE_LOOP_ENA_MSK	0x8000000	
ALT_ETH_10G_CMD_BROAD_FILTER_ENA_OFST	28	Configures the BROAD_FILTER_ENA bit.
ALT_ETH_10G_CMD_BROAD_FILTER_ENA_MSK	0x10000000	
ALT_ETH_10G_CMD_INS_CRC_ERROR_OFST	29	Configures the INS_CRC_ERROR bit.
ALT_ETH_10G_CMD_INS_CRC_ERROR_MSK	0x20000000	
ALT_ETH_10G_CMD_NO_FIFO_PAUSE_OFST	30	Configures the NO_FIFO_PAUSE bit.
ALT_ETH_10G_CMD_NO_FIFO_PAUSE_MSK	0x40000000	
ALT_ETH_10G_CMD_CNT_RESET_OFST	31	Configures the CNT_RESET bit.
ALT_ETH_10G_CMD_CNT_RESET_MSK	0x80000000	

64-Bit Statistics Counters

Some of the statistics counters are 64-bits wide. To access this register you must access the top 32-bits of the counter as a separate read operation.



If you read the lower 32-bits first followed by the upper 32 bits, the upper 32-bit value may have incremented by one.

The following C-code sample shows how to work around this behavior:

```
alt_u32 Cnt_Upper_First_Time;
alt_u32 Cnt_Lower;
alt_u32 Cnt_Upper_Second_Time;
alt_u64 Cnt_Value;
Cnt_Upper_First_Time = IORD_ALT_ETH_10G_A_OCTETS_RX_OK_HI(base);
Cnt_Lower = IORD_ALT_ETH_10G_A_OCTETS_RX_OK(base);
Cnt_Upper_Second_Time = IORD_ALT_ETH_10G_A_OCTETS_RX_OK_HI(base);
if (Cnt_Upper_Second_Time != Cnt_Upper_First_Time)
{
    if (Cnt_Lower > 0x80000000)
    {
        Cnt_Value = (alt_u64)Cnt_Upper_First_Time << 32;
        Cnt_Value += Cnt_Lower;
    }
    else
    {
        Cnt_Value = (alt_u64)Cnt_Upper_Second_Time << 32;
        Cnt_Value += Cnt_Lower;
    }
}
else
{
    Cnt_Value = (alt_u64)Cnt_Upper_First_Time << 32;
    Cnt_Value += Cnt_Lower;
}
```

Testbench

You can use the testbench provided with the 10-Gbps Ethernet reference design to exercise your reference design. The testbench includes the following features:

- Easy-to-use simulation environment for any standard HDL simulator
- Simulation of all basic Ethernet packet transactions
- Open source Verilog HDL testbench files

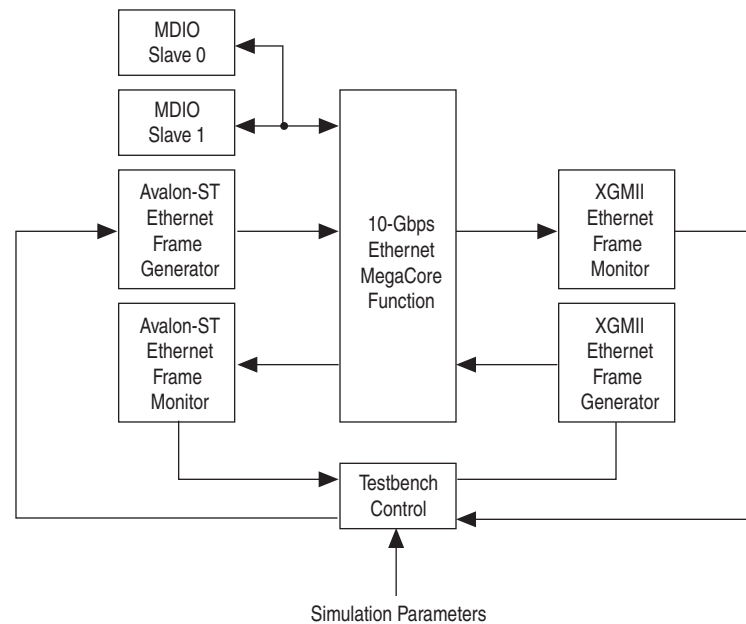


For more information on simulation, see [“Simulate with Provided Testbench” on page 7](#).

Testbench Architecture

This section describes the testbench architecture for each 10-Gbps Ethernet reference design.

[Figure 24](#) shows the 10-Gbps Ethernet testbench architecture.

Figure 24. Testbench Architecture

Overview

The testbench demonstrates and verifies the following functionality:

- Transmit and receive datapaths are functionally correct.
- Ethernet frames of valid length (greater than 64 bytes) received on the Avalon-ST transmit interface are sent out through the XGMII transmit interface with correct CRC-32 inserted by the reference design. Short frames are always padded by the reference design up to at least 64 bytes in length.
- The CRC-32 is optionally discarded before forwarding the frames onto the Avalon-ST receive interface.

Default Testbench

The default testbench:

- Sets the `command_config` register to `0x80000203`.
- Configures the interframe gap to 12.
- Simulates the receive and transmit datapaths independently.
- Sends five Ethernet frames of payload length 100, 101, 102, 103 and 104 bytes to the Avalon-ST transmit interface and the XGMII (and XAUI) receive interface without errors.
- Forwards all received frames that are not filtered based on the destination address of the received frames.



The testbench works with a FIFO size of equal to or greater than 64 bytes.

Test Flow

The testbench performs the following operations upon a simulated power-on reset:

- Initialize the reference design, which consists the following operations:
 - Set the operation mode via the `command_config` register.
 - Set the address via the `mac_0` and `mac_1` registers.
 - Set the IFG for transmit frames via the `tx_ipg_length` register.
 - Set the Avalon-ST FIFO threshold registers.
- Set the supplemental unicast addresses.
- Start transmission and clear receive and transmit FIFO.

Using the Testbench

The testbench can send and check received frames. The testbench supports the following basic functions:

- Generates valid Ethernet frames and checks their validity
- Generates Ethernet frames with CRC errors
- Generates frames on the Avalon-ST interface
- Allows access to the register interface
- Generates frames with vlan or stacked vlan tags
- Allows MDIO reads and writes

The maximum supported frame size is 16 Kbytes.

The functions are similar in Verilog HDL and VHDL, but have slightly different names. Also the VHDL functions have more ports.



The procedures are VHDL; tasks are Verilog HDL.

The VHDL functions are part of the package in the `eth_10g_rd_lib` library. The Verilog HDL functions are called with the command (for example, `demo_eth_gen.gen_crc_errored_frame`).

Ethernet Frame Generation

```
procedure gen_valid_eth_frame (pkt_type, pkt_length, packet_number,
current_idc, pkt_ifg, clk, eth_generated_data, eth_generated_ctrl)
task demo_eth_gen.gen_valid_frame; (pkt_length, pkt_type, min_ipg)
```

where:

- `pkt_type` is the two octet type or length
- `pkt_length` is the physical size, which includes the payload only and must be greater than 4
- `packet_number` is a 32-bit vector that you must define in the testbench
- `current_idc` is a two-bit vector deficit idle counter

- *pkt_ifg*: 16-bit vector packet interframe gap, which you must specify. The minimum value is 8 bytes
- *clk* is the clock for the generation, which should be 156.25 MHz
- *eth_generated_data* is a 32-bit vector DDR data signal sent to the MAC
- *eth_generated_ctrl* is a four-bit DDR control signal sent to the MAC

```

procedure gen_crc_errored_eth_frame (pkt_type, pkt_length,
packet_number, current_idc, pkt_ifg, clk, eth_generated_data,
eth_generated_ctrl)
task demo_eth_gen.gen_crc_errored_frame; (pkt_length, pkt_type,
min_ipg)
procedure gen_vlan_eth_frame (pkt_type, pkt_length, packet_number,
current_idc, pkt_ifg, clk, eth_generated_data, eth_generated_ctrl)
task demo_eth_gen.gen_vlan_frame; (pkt_length, pkt_type, min_ipg)
procedure gen_vlan_vlan_eth_frame (pkt_type, pkt_length, packet_number,
current_idc, pkt_ifg, clk, eth_generated_data, eth_generated_ctrl)
task demo_eth_gen.gen_vlan_vlan_frame; (pkt_length, pkt_type, min_ipg)
procedure gen_pause_eth_frame (pause_length, use_multicast,
packet_number, current_idc, clk, eth_generated_data,
eth_generated_ctrl)
task demo_eth_gen.gen_pause_frame; (pause_length, use_multicast)

```

where:

- *pause_length* is the 16-bit pause length, which is the length of the pause to transmit in 512-bit time units
- *use_multicast* is 1 to use the pause multicast address; 0 to use the MAC unicast address

```

procedure gen_eth_idle (number_of_idle, clk, eth_generated_data,
eth_generated_ctrl)

```

where:

- *number_of_idle* is an integer number of idle bytes, which must be a multiple of 4

```

task demo_eth_gen.gen_idle; (number_of_idle)

```

where:

- *number_of_idle* is the number of idle cycles sent

Avalon-ST Frame Generation

```

procedure gen_valid_avl_st_frame (pkt_type, pkt_length, packet_number,
clk, avl_st_tx_dat, avl_st_tx_sop, avl_st_tx_eop, avl_st_tx_err,
avl_st_tx_ena, avl_st_tx_dav, avl_st_tx_mty)

```

where:

- *pkt_type* is the two octet type or length
- *pkt_length* is the physical size, which includes the payload only and must be greater than 4.
- *packet_number* is a 32-bit vector that you must define in the testbench
- *clk* is the clock for the generation, which should be 156.25 MHz
- *avl_st_tx_**** are the Avalon-ST interface signals (see [Table 27](#))

```

task demo_avl_st_gen.gen_valid_frame; (pkt_length, pkt_type, min_ipg)

```

where:

- *pkt_type* is the data that you entered in the length and type field of the frame
- *pkt_length* is the length of the payload in bytes
- *packet_number* is a number on each packet that is incremented after it is sent

```
procedure gen_avl_st_idle (number_of_idle, clk, avl_st_tx_dat,
avl_st_tx_sop, avl_st_tx_eop, avl_st_tx_err, avl_st_tx_ena,
avl_st_tx_dav, avl_st_tx_mty)
```

where:

- *number_of_idle* is in clock cycles

```
task demo_avl_st_gen.gen_idle; (idle_length)
```

Avalon MM Interface

The following function writes to the configuration and MDIO interface:

```
procedure write_avalon (data_to_write, address_to_write,
avalon_writedata, avalon_address, avalon_write, avalon_waitrequest,
avalon_clk)
task demo_config_eth10g.write_avalon; (data_to_write,
address_to_write)
```

where:

- *data_to_write* is the data to write
- *address_to_write* is the address of the register to write to
- *avalon_**** are the Avalon-MM interface signals (see [Table 22](#))

The following function reads to the configuration and MDIO interface:

```
procedure read_avalon (address_to_read, data_read, avalon_readdata,
avalon_address, avalon_read, avalon_waitrequest, avalon_clk)
task demo_config_eth10g.read_avalon; (address_to_read, data_read)
```

where:

- *address_to_read* is the address of the register to read from
- *data_read* is the data to read

The following function performs a write to an MDIO register:

```
procedure write_mdio (clause45, prtadr, devphyadr, regadr, data,
avalon_writedata, avalon_address, avalon_write, avalon_waitrequest,
avalon_clk)
task demo_config_eth10g.write_mdio; (clause45, prtadr, devphyadr,
regadr, data)
```

where:

- *clause45* is set to 1 if the MDIO is for clause 45; set to 0 if the MDIO is for clause 22
- *portadr* is the port address (clause 45 only)
- *devphyadr* is the device or PHY address (device in clause 45; PHY in clause 22)
- *regadr* is the register address. 16-bit address in clause 45; only the bottom 5-bits for clause 22

The following function performs a read an MDIO register:

```

procedure read_mdio (clause45, prtadr, devphyadr, regadr, data_read,
avalon_writedata, avalon_readdata, avalon_address, avalon_write,
avalon_read, avalon_waitrequest, avalon_clk)
taskdemo_config_eth10g.read_mdio; (clause45, prtadr, devphyadr, regadr,
data_read)

```

Testbench Errors

By default, the testbench stops after it encounters the first error. You can change this setting by editing the value of `tb.err_limit` to a higher number.

External Reconfiguration Block Example

This section describes an example design where the 10 Gbps Ethernet MAC uses an external reconfiguration block. This example shows a way of connecting the blocks for a design targeting Stratix II GX devices. The example design is in the `eth_10g\reconfig_example` directory.


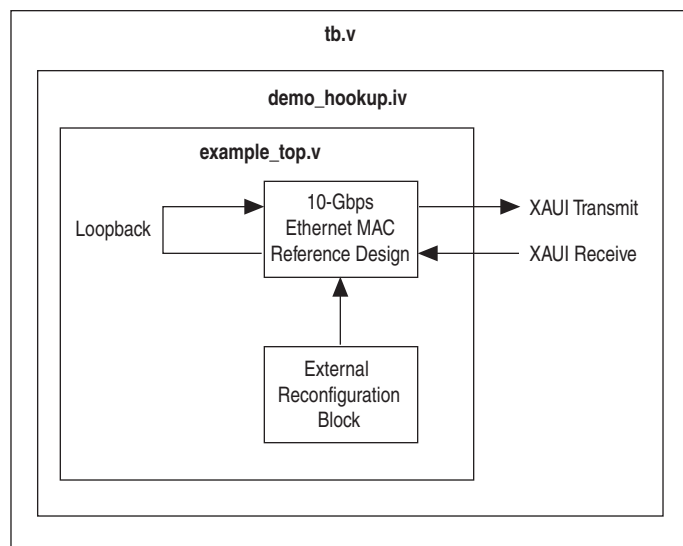
 The external reconfiguration block is optional when targeting Stratix II GX devices but mandatory when targeting Stratix IV GX devices.

Figure 25 shows the example overview.

Figure 25. External Reconfiguration Block Example Overview



You can use this example in real hardware but you may need to map the Avalon MM interfaces to an internal bus structure as the number of pins required may be high.

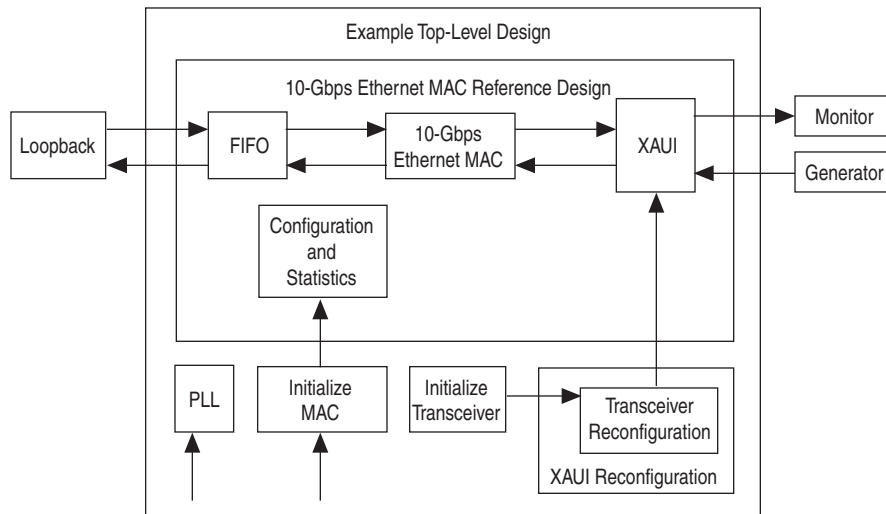
The example includes an Altera-generated a 10-Gbps Ethernet reference design, `mac_10g_example.v`, and the following files:

- `demo_hookup.iv`, which uses `example_top.v` and has two new Avalon-MM interfaces and does not have local Avalon-ST interfaces
- `demo_run_modelsim.tcl` includes modifications to compile the new files
- `do_msim.do` is the ModelSim script that runs the simulation

- **tb.v** does not configure the MAC and only sends data on the receiver channel. The loopback function sends the packets sent on the receiver back on the transmitter.

Figure 26 shows the example block diagram.

Figure 26. Example Block Diagram



The example has the following blocks:

- Example top-level design (**example_top.v**)
- Reference design (**mac_10g_example.v**)
- MAC initialization (**init_mac.v**)
- XAUI reconfiguration (**xaui_reconfig.v**)
- Transceiver reconfiguration (**gxb_reconfig_sii.v**)
- Transceiver initialization (**init_gxb.v**)
- Loopback (**loopback_module.v**)
- PLL (**eth_pll.v**)

Example Top-Level Design

The top-level design file, which has the following interfaces:

- XAUI for its data channel
- Avalon-MM interface for the MAC. The clock for this interface is provided by the example top block
- Avalon-MM for the XAUI reconfiguration block. The clock for this interface is provided by the example top. It is the same as the one for the Avalon MM for the MAC and must run at exactly 1/4 of the system clock.
- Input clock, which is fed to a PLL
- Reset signal

Reference Design

The 10-Gbps Ethernet reference design has the following main parameters:

- XAUI with External reconfiguration block
- FIFO support

MAC Initialization

The MAC initialization for the MAC registers waits a certain amount of time at startup, then configures various registers such as FIFO thresholds and the configuration register (by turning on the receiver and transmitter and setting the receiver to pass through). When the configuration finishes, the commands are then taken from the Avalon-MM interface.

XAUI Reconfiguration

The XAUI reconfiguration instantiates the transceiver reconfiguration block. This block converts Avalon-MM instructions into a format supported by the transceiver configuration module. In this design example, the clock relationship between the system clock and the reconfiguration clock is critical.

Transceiver Reconfiguration

The transceiver reconfiguration is an instantiation of the transceiver reconfiguration block. It has access to the relevant parameters to the XAUI interface. You can view the parameters by opening the the file with the MegaWizard Plug-In.

Transceiver Initialization

Similarly to the MAC initialization, the transceiver initialization first waits for the system to stabilize (reset the blocks). Then it writes then reads access to the reconfiguration block. When complete, the Avalon-MM interface can control the reconfiguration block. The read after configuration is important.

For more information on transceiver initialization, refer to the [Stratix II GX ALT2GXB User Guide](#).

Table 30 shows the interface mapping.

Table 30. Interface Mapping (Part 1 of 2)

Address	Access	Bit	Mapping
0x0	RW	0	Start operation.
		1	0 perform read; 1 perform write.
0x4	RO	0	Busy.

Table 30. Interface Mapping (Part 2 of 2)

Address	Access	Bit	Mapping
0x8	RW	2:0	Transmitter vodctrl.
		7:4	Transmitter preamphasis setting 0.
		11:8	Transmitter preamphasis setting 1.
		15:12	Transmitter preamphasis setting 2.
		19:16	Receiver equalizer.
		21:20	Receiver equalizer DC gain.
		27:24	RW mode selection.
		29:28	Channel number.
0xC	RO	—	Channel 0 status (see Table 31 for mapping).
0x10		—	Channel 1 status (see Table 31 for mapping).
0x14		—	Channel 2 status (see Table 31 for mapping).
0x18		—	Channel 3 status (see Table 31 for mapping).

[Table 31](#) shows the channel status mapping per channel.

Table 31. Channel Status Mapping Per Channel

Bit	Mapping
2:0	vodctrl.
7:4	Preamphasis setting 0.
11:8	Preamphasis setting 1.
15:12	Preamphasis setting 2.
19:16	Equalizer control.
21:20	Equalizer DC gain.

Loopback

The loopback reads the data from the Avalon-ST receiver interface of the 10-Gbps Ethernet MAC reference design, invert the source and destination addresses, and then forwards the data back to the transmitter channel. It expects a FIFO interface and does not work if the reference design has no FIFO.

PLL

The PLL generates all the clocks required for the example. It assumes that the input clock frequency is 100 MHz. The example has the following output clocks:

- C0 is the Avalon-ST clock (for the loopback) running at 200 MHz

- C1 is the Avalon-MM clock running at 80 MHz
- C2 is the reconfiguration block running at 1/4 of the Avalon-MM clock at 20 MHz

Table 32 shows the example design's signals.

Table 32. Design Example Signals

Port Name	Direction	Size	Avalon Type	Description
reset_n	Input	1	reset_n	Reset signal.
inclk	Input	1	export	Main input for the example top, expected to run at 100 MHz.
Serdes_refclock	Input	1	export	Clock that runs the MAC and SERDES, running at 156.25 MHz.
avlclk	Output	1	export	The clock that connects to the Avalon-MM interfaces.
avl_mm_mac_address	Input	32	address	MAC Avalon-MM address.
avl_mm_mac_write_n	Input	1	write_n	MAC Avalon-MM write.
avl_mm_mac_writedata	Input	32	writedata	MAC Avalon-MM write data.
avl_mm_mac_read_n	Input	1	read_n	MAC Avalon-MM read.
avl_mm_mac_readdata	Output	32	readdata	MAC Avalon-MM read data.
avl_mm_mac_waitrequest	Output	1	waitrequest	MAC Avalon-MM wait request.
avl_mm_gxb_address	Input	32	address	Reconfiguration Avalon-MM address.
avl_mm_gxb_write_n	Input	1	write_n	Reconfiguration Avalon-MM write.
avl_mm_gxb_writedata	Input	32	writedata	Reconfiguration Avalon-MM write data.
avl_mm_gxb_read_n	Input	1	read_n	Reconfiguration Avalon-MM read.
avl_mm_gxb_readdata	Output	32	readdata	Reconfiguration Avalon-MM read data.
avl_mm_gxb_waitrequest	Output	1	waitrequest	Reconfiguration Avalon-MM wait request.
xau_i_rx_data	Input	4	export	XAUI receive data.
xau_i_tx_data	Output	4	export	XAUI transmit data.

Simulate the Example Design

To simulate the example design, follow these steps:

1. Start the ModelSim simulator.
2. Change the working directory to the `eth_10g\reconfig_example` directory.
3. Run the following command:

```
do_modelsim.do
```

Document Revision History

Table 33 shows the revision history for this application note.

Table 33. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
October 2008 v1.1	<ul style="list-style-type: none">■ Added clocking diagram■ Added external reconfiguration block information	—
June 2008 v1.0	First release.	—



101 Innovation Drive
San Jose, CA 95134
www.altera.com
Technical Support
www.altera.com/support

Copyright © 2008 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera 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 Altera Corporation. Altera 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.



I.S. EN ISO 9001