AN-744: Scalable Triple Speed Ethernet Reference Designs for Arria 10 Devices

2016-5-13

AN-744 Subscribe Send Feedback

The Altera® scalable Triple-Speed Ethernet reference designs demonstrate the functionalities of the Altera Triple-Speed Ethernet MegaCore® in Arria 10 devices. The reference designs offer the following features:

- Multi-channel Triple-speed Ethernet MAC IP operating at 10/100/1000 Mbps.
- Scability, up to 10 Ethernet channels.
- Implementation of the IEEE 1588v2 feature in one of the designs.
- Packet monitoring on the TX and RX datapaths.
- Reporting of the MAC TX and RX statistics counters.
- Testing of various types of Ethernet packet transfer.

Block Diagrams

The following diagrams show the components of the reference designs, and their clocking and reset schemes.

Figure 1: Block Diagram of the Reference Design Without the IEEE 1588v2 Feature

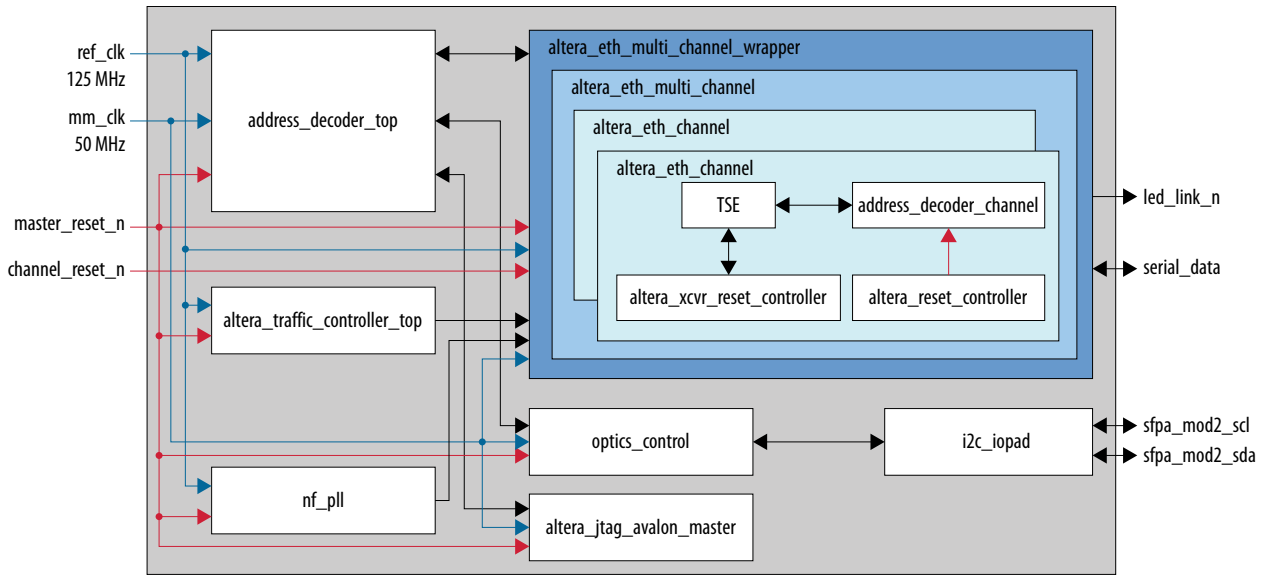
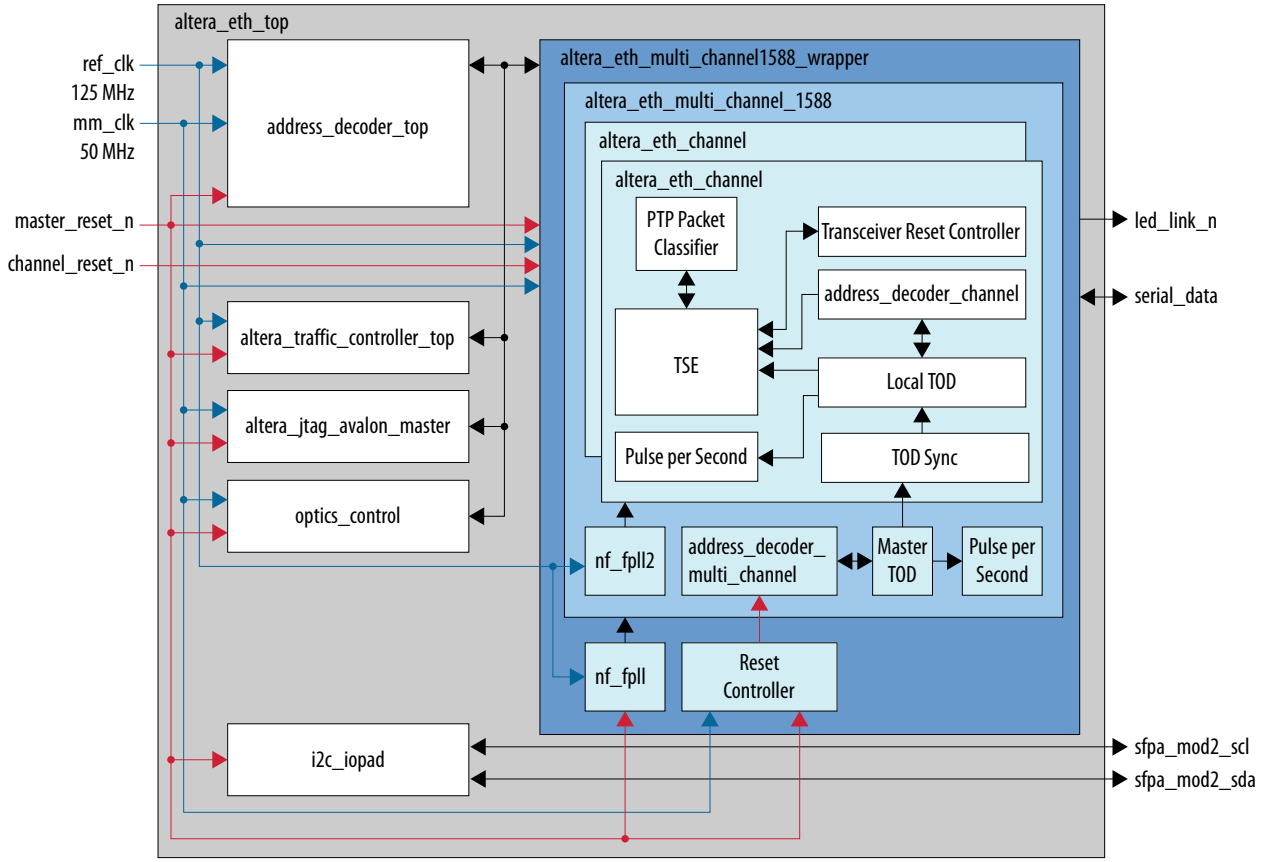


Figure 2: Block Diagram of the Reference Design With the IEEE 1588v2 Feature



The designs consist of two asynchronous reset domains:

- Master reset—resets all channels when the active-low signal, `master_reset_n`, is asserted. Altera recommends that you trigger the master reset upon power-up, when the FPGA is successfully configured.
- Channel reset—resets channel *n* when the active low signal, `channel_reset_n[n]`, is asserted.

Design Components

Table 1: Reference Design Components

Component	Description
Altera Triple-Speed Ethernet IP core	Provides the MAC and PCS functionalities and features.
I^2C Controller	Monitors and controls the SFP module. This controller consists of the `optics_control` and `i2c_iopad` modules.
Traffic Controller	Generates and monitors Ethernet packets for hardware verification.

Component	Description
Altera JTAG Avalon Master	Provides an interface to the Qsys tool for users to initiate Avalon Memory-Mapped transactions through System Console.
Address Decoder Top	The address decoder module for the design.
Address Decoder Channel	The address decoder module for each component within the channel.
Address Decoder Multi-Channel	The address decoder module for all channels and the components used for all channels, such as the Master TOD.
Reset Controller	Synchronizes the reset of all design components.
fPLL	Generates the `tx_serial_clk`, `pcs_phase_measure_clk`, and `tod_sync_sampling_clk clocks` for the designs.
Components for the IEEE 1588v2 feature	
Master Time-of-Day (TOD)	The master time-of-day for all channels.
TOD Synch	Synchronizes the time-of-day from the Master TOD module to the Local TOD module for all channels.
Local TOD	The time-of-day module for each channel.
Pulse per Second Module	Returns the pulse per second (pps) value for all channels.
PTP Packet Classifier	Decodes the packet type of each incoming PTP packet and returns the information to the Triple-Speed Ethernet IP core.

Design Parameters

The reference designs allow you to set the total number of channels. The design with the IEEE 1588v2 allows you to also set the width of the timestamp fingerprint. You can specify these parameters by editing the **altera_eth_top.sv** file.

Table 2: Parameters

Name	Description	Default Value
`NUM_CHANNELS`	Use this parameter to specify the total number of Ethernet channels to be instantiated in the design. You can specify between 1 to 10 channels.	2
`TSTAMP_FP_WIDTH`	Use this parameter to set the width of the timestamp fingerprint width for the design. The value of this parameter must match the value of the **Timestamp fingerprint width** parameter of the Triple-Speed Ethernet MAC IP. Each time you change the value of this parameter, you must reconfigure and regenerate the MAC IP.	4

Hardware and Software Requirements

Altera uses the following hardware and software to test the designs and testbench in Linux:

- Altera Complete Design Suite (ACDS) version 15.0
- ModelSim-SE version 10.3d
- Arria 10 GX SI Development Board (10AX115S4F45I3SGE2)
- Clock Control module, which is shipped with the Installation Kit for the Arria 10 GX SI Development Board

Related Information

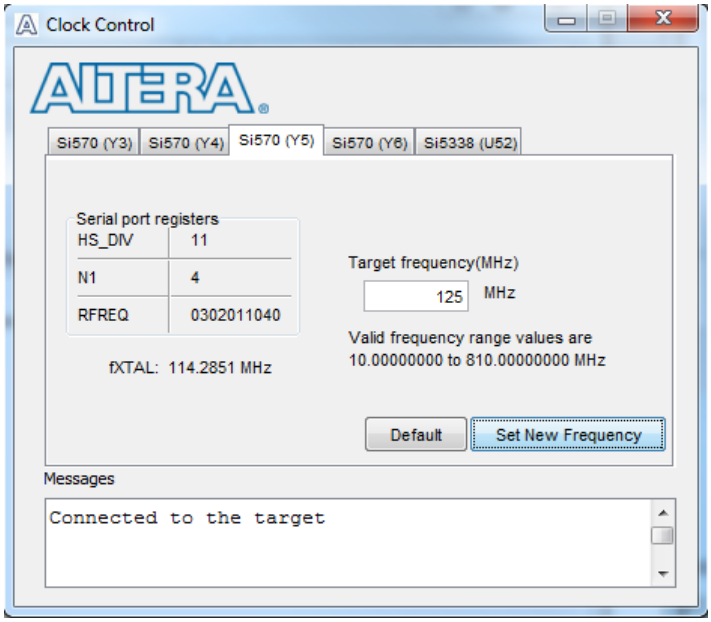
- **Reference Design With the IEEE 1588v2 Feature**
- **Reference Design Without the IEEE 1588v2 Feature**
  Click to download the design files.

Using the Reference Design

Follow these steps to start using the reference design:

1. Unzip the design files in the project directory.
   - **altera_eth_tse_wo_1588.tar.gz**—the design without the IEEE 1588v2 feature.
   - **altera_eth_tse_w_1588.tar.gz**—the design with the IEEE 1588v2 feature.
2. Change to one of the following working directories:
   - **altera_eth_tse_wo_1588** if you are using the design without the IEEE 1588v2 feature.
   - **altera_eth_tse_w_1588** if you are using the design with the IEEE 1588v2 feature.
3. Launch the Quartus II software and open the project file, **altera_eth_top.qpf**.
4. The default number of channels for the designs is two. To change the number of channels, open the **altera_eth_top.sv** file and change the value of the `NUM_CHANNELS` parameter.
5. Select **Processing > Start Compilation** to compile the design example.
   You may see a net delay violation at the end of the compilation, which you can ignore. The cause is the violation of a lower boundary in the **altera_avalon_mm_clock_crossing bridge** module.
6. Configure the FPGA on the Arria 10 GX SI development board using the generated programming file, **altera_eth_top.sof**.
7. When the FPGA is successfully configured, launch the **Clock Control** tool. The tool's executable—**\examples\board_test_system\ClockControl.exe**—is shipped with the Installation Kit for the Arria 10 GX SI development board.
8. Set Y5 to 125 MHz.

Figure 3: Clock Control



9. Reset the system by pressing the user push button `S1`.
10. In the Quartus II software, select **Tool > System Debugging Tools > System Console**.
11. In the **System Console** command shell, change to the **SystemConsole** directory.
12. Initialize the reference design command list by running this command, `source main.tcl`.
13. You can now perform the tests listed below. In these tests, the Triple-Speed Ethernet IP core operates in SGMII mode.

Option	Description
PHY internal serial loopback	`TEST_PHYSERIAL_LOOPBACK` <*channel_number*> <*speed_test*> <*burst_size*> Example: `TEST_PHYSERIAL_LOOPBACK 0 1G 1000`
SMA loopback	`TEST_SMA_LOOPBACK` <*channel_number*> <*speed_test*> <*burst_size*> Example: `TEST_SMA_LOOPBACK 0 1G 1000`
SPF+ loopback between 2 channels	`TEST_1588` <*from_channel*> <*to_channel*> <*speed_test*> Example: `TEST_1588 0 0 1G`

Upgrading the Components

You must upgrade the components each time you upgrade the ACDS to a new version. To upgrade the components, follow these steps:

1. Launch the Quartus II software and open the project file, **altera_eth_top.qpf**.
2. Select **Project > Upgrade IP** Components.
3. Click the **Perform automatic upgrade** button. Ensure the version displayed is updated accordingly.

Testbench

You can use the provided testbench to verify the reference designs. The testbench loops back Ethernet packet into the designs.

Figure 4: Testbench Block Diagram

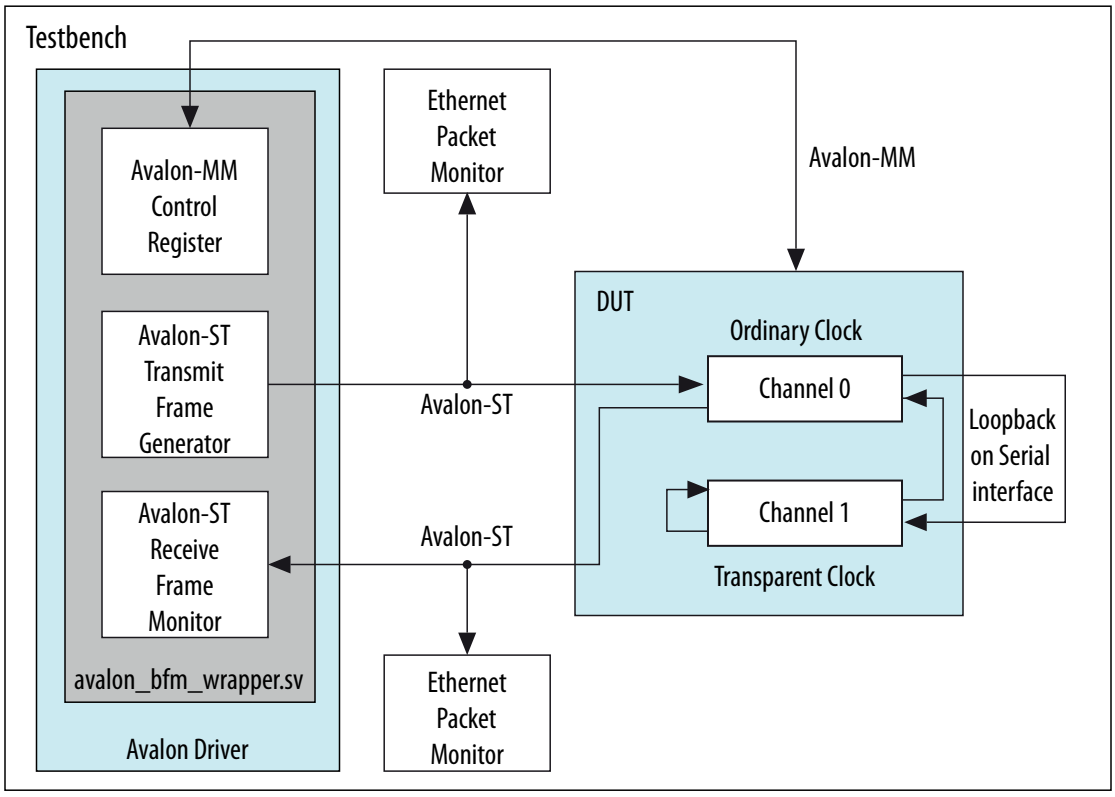


Table 3: Testbench Components

Component	Description
Device under test (DUT)	The reference design.

Component	Description
Avalon driver	Consists of Avalon Streaming (Avalon-ST) master bus functional models (BFMs) to generate and monitor ethernet packets. The driver also connects with the design components through the Avalon Memory-Mapped (Avalon-MM) interface.
Ethernet packet monitors	Monitor transmit and receive datapaths, and display the frames in the simulator console.

Testbench Files

The testbench files are located in the following folders:

- **<*project directory*>/altera_eth_tse_wo_1588/testbench/Modelsim/testcase<*n*>** for the reference design without the IEEE 1588v2 feature.
- **<*project directory*>/altera_eth_tse_w_1588/testbench/Modelsim/testcase<*n*>** for the reference design with the IEEE 1588v2 feature.

Table 4: Testbench Files

Name	Description
avalon_bfm_wrapper.sv	A wrapper for the Avalon BFMs that the **avalon_driver.sv** file uses.
avalon_driver.sv	A SystemVerilog HDL driver that uses the BFMs to form the TX and RX datapaths, and access the Avalon-MM interface.
avalon_if_params_pkg.sv	A SystemVerilog HDL testbench that contains parameters to configure the BFMs. Because the configuration is specific to the DUT, you must not change the contents of this file.
avalon_st_eth_packet_monitor.sv	A SystemVerilog HDL testbench that monitors the Avalon-ST TX and RX interfaces.
default_test_params_pkg.sv	A SystemVerilog HDL package that contains the default parameter settings of the testbench.
eth_mac_frame.sv	A SystemVerilog HDL class that defines the Ethernet frames. The **avalon_driver.sv** file uses this class.
eth_register_map_params_pkg.sv	A SystemVerilog HDL package that maps addresses to the Avalon-MM control registers.
ptp_timestamp.sv	A SystemVerilog HDL class that defines the timestamp in the testbench.
tb_run.tcl	A Tcl script that starts a simulation session in the ModelSim simulation software.
tb_testcase.sv **tb_testcase_1588.sv**	A SystemVerilog HDL testbench file that controls the flow of the testbench.

Name	Description
tb_top_n.sv **tb_top_n_1588.sv**	The top-level testbench file consists of the device under test (DUT), a client packet generator, and a client packet monitor along with other logic blocks.
wave.do	A signal tracing macro script that the ModelSim simulation software uses to display testbench signals.

Simulating the Testbench

The simulation script uses **QUARTUS_ROOTDIR** environment variable to access Altera simulation model libraries. You have to set the **QUARTUS_ROOTDIR**to point to the Quartus II installation path after installation. If this environment variable is missing, then you must set the variable manually.

Follow these steps to use the ModelSim simulator to simulate the testbench designs:

1. Ensure that the `QUARTUS_ROOTDIR` environment variable is pointing to the Quartus II installation path. If this environment variable is missing, you must set the variable manually.

   The simulation script uses this environment variable to access Altera simulation model libraries.
2. Change to one of the following directories.
   - **<*project directory*>/altera_eth_tse_wo_1588/testbench/Modelsim/testcase<*n*>** for the design without the IEEE 1588v2 feature.
   - **<*project directory*>/altera_eth_tse_w_1588/testbench/Modelsim/testcase<*n*>** for the design with the IEEE 1588v2 feature.
3. Launch the Modelsim simulator, and run **do tb_run.tcl** to set up the required libraries, compile the generated IP functional simulation model, and exercise the simulation model with the provided testbench.

Test Case: Design With the IEEE 1588v2 Feature

The simulation performs the following steps.

1. Starts up the design with an operating speed of 1 Gbps.
2. Configures the MAC, PHY speed, and FIFO buffer for the channels.
3. Waits until the design asserts the `channel_ready` signal for each channel.
4. Sends the following packets:
   - Basic data frame with PTP over Ethernet, PTP Sync Message, and 1-step PTP.
   - VLAN data frame with PTPover UDP/IPv4, PTP Sync Message, and 1-step PTP.
   - Stacked VLAN data frame with PTP over UDP/IPv6, PTP Sync Message, and 2-step PTP.
   - Basic data frame with PTP over Ethernet, PTP Delay Request Message, and 1-step PTP.
   - VLAN data frame with PTPover UDP/IPv4, PTP Delay Request Message, and 2-step PTP.
   - SVLAN data frame with PTP over UDP/IPv6, PTP Delay Request Message, and 1-step PTP.
5. Repeats steps 2 to 4 for operating speeds 100 Mbps and 10 Mbps.

When simulation ends, the values of the MAC statistics counters are displayed in the transcript window. The transcript window also displays PASSED if the RX Avalon-ST interface of channel 0 received all packets successfully.

Figure 5: Simulation Results

```
# ------------------------
# Channel         1: MAC Statistics
# ------------------------
#       aFramesTransmittedOK              = 6
#       aFramesReceivedOK                 = 6
#       aFrameCheckSequenceErrors         = 0
#       aAlignmentErrors                  = 0
#       aOctetsTransmittedOK              = 3836
#       aOctetsReceivedOK                 = 3836
#       aTxPAUSEMACCtrlFrames             = 0
#       aRxPAUSEMACCtrlFrames             = 0
#       ifInErrors                        = 0
#       ifOutErrors                       = 0
#       ifInUcastPkts                     = 4
#       ifInMulticastPkts                 = 1
#       ifInBroadcastPkts                 = 1
#       ifOutDiscards                     = 0
#       ifOutUcastPkts                    = 4
#       ifOutMulticastPkts                = 1
#       ifOutBroadcastPkts                = 1
#       etherStatsDropEvents              = 0
#       etherStatsOctets                  = 3944
#       etherStatsPkts                    = 6
#       etherStatsUndersizePkts           = 0
#       etherStatsOversizePkts            = 0
#       etherStatsPkts64Octets            = 1
#       etherStatsPkts65to127Octets       = 1
#       etherStatsPkts128to255Octets      = 0
#       etherStatsPkts256to511Octet       = 1
#       etherStatsPkts512to1023Octets     = 2
#       etherStatsPkts1024to1518Octets    = 1
#       etherStatsPkts1519OtoXOctets      = 0
#       etherStatsFragments               = 0
#       etherStatsJabbers                 = 0
#
#
# Simulation PASSED
#
# ** Note: $finish    : tb_testcase.sv(199)
#    Time: 102465 ns  Iteration: 1  Instance: /tb_top/TESTCASE
```

Test Case: Design Without the IEEE 1588v2 Feature

The simulation performs the following steps.

1. Starts up the design with an operating speed of 1 Gbps.
2. Configures the MAC, PHY speed, and FIFO buffer for the channels.
3. Waits until the design asserts the `channel_ready` signal for each channel.
4. Sends the following packets:

- Basic data frame, 64 bytes.
- VLAN multicast data frame, 1000 bytes.
- Basic data frame, 800 bytes.
- SVLAN broadcast data frame, 80 bytes.
- VLAN unicast data frame, 500 bytes.
- SVLAN data frame, 1500 bytes.

5. Repeats steps 2 to 4 for operating speeds 100 Mbps and 10 Mbps.

When simulation ends, the values of the MAC statistics counters are displayed in the transcript window. The transcript window also displays PASSED if the RX Avalon-ST interface of channel 0 received all packets successfully.

Figure 6: Simulation Results

```
# -----------------------
# Channel         1: MAC Statistics
# -----------------------
#       aFramesTransmittedOK              = 6
#       aFramesReceivedOK                 = 6
#       aFrameCheckSequenceErrors         = 0
#       aAlignmentErrors                  = 0
#       aOctetsTransmittedOK              = 500
#       aOctetsReceivedOK                 = 500
#       aTxPAUSEMACCtrlFrames             = 0
#       aRxPAUSEMACCtrlFrames             = 0
#       ifInErrors                        = 0
#       ifOutErrors                       = 0
#       ifInUcastPkts                     = 0
#       ifInMulticastPkts                 = 6
#       ifInBroadcastPkts                 = 0
#       ifOutDiscards                     = 0
#       ifOutUcastPkts                    = 0
#       ifOutMulticastPkts                = 6
#       ifOutBroadcastPkts                = 0
#       etherStatsDropEvents              = 0
#       etherStatsOctets                  = 608
#       etherStatsPkts                    = 6
#       etherStatsUndersizePkts           = 0
#       etherStatsOversizePkts            = 0
#       etherStatsPkts64Octets            = 0
#       etherStatsPkts65to127Octets       = 4
#       etherStatsPkts128to255Octets      = 2
#       etherStatsPkts256to511Octet       = 0
#       etherStatsPkts512to1023Octets     = 0
#       etherStatsPkts1024to1518Octets    = 0
#       etherStatsPkts1519ToXOctets       = 0
#       etherStatsFragments               = 0
#       etherStatsJabbers                 = 0
#
#
# Simulation PASSED
#
# ** Note: $finish    : tb_testcase_1588.sv(436)
#    Time: 77655 ns  Iteration: 1  Instance: /tb_top/TESTCASE
```

Interface Signals

Table 5: Top-level Design Signals

Signal	Direction	Width	Description
ref_clk	In	1	125-MHz reference clock for the design.
mm_clk	In	1	50-MHz clock for Avalon-MM interface.
channel_reset_n[]	In	NUM_CHANNELS	An asynchronous and active-low reset signal that resets individual Ethernet channels. This signal does not reset the components shared by all channels, such as the master TOD, master PPS, reconfig bundle, and fPLLs.
master_reset_n	In	1	An asynchronous and active-low reset signal that resets the entire design.
led_link_n	Out	NUM_CHANNELS	When asserted, this active-low signal indicates a successful link synchronization.
rx_serial_data	In	NUM_CHANNELS	RX serial data.
tx_serial_data	Out	NUM_CHANNELS	TX serial data.
sfpa_mod1_scl	Bidir	1	Serial clock line of the I2C interface.
sfpa_mod2_sda	Bidir	1	Serial data line of the I2C interface.
sfp_txdisable	Out	1	Deasserted to enable the transmitter optical output.
xfp_txdisable	Out	1	Deasserted to enable the transmitter optical output.

Related Information

Triple-Speed Ethernet MegaCore Function User Guide
Describes the interface signals of the Triple-Speed Ethernet IP Core.

Packet Classifier Interface Signals

Table 6: Packet Classifier Interface Signals

The packet classifier is one of the components in the design with the IEEE 1588v2 feature.

Signal	Direction	Width	Description
tx_egress_timestamp_request_in_valid[]	In	NUM_CHANNELS	Assert this signal if a timestamp is required for the TX frame. Align this signal to the start of the packet.

Signal	Direction	Width	Description
tx_egress_timestamp_request_in_fingerprint[][]	In	NUM_CHANNELS x TSTAMP_ FP_WIDTH	Specifies the fingerprint of the TX frame for a given channel. You configure the width of the fingerprint using the TSTAMP_FP_WIDTH parameter.
clock_operation_mode[][]	In	NUM_CHANNELS x 2	Specifies the clock mode. • 00: Ordinary clock • 01: Boundary clock • 10: End to end transparent clock • 11: Peer to peer transparent clock
pkt_with_crc_mode[]	In	NUM_CHANNELS	Assert the respective signal bit to indicate that the packet for the channel contains a CRC field. Otherwise, deassert the signal bit.
tx_ingress_timestamp_valid[]	In	NUM_CHANNELS	Assert the respective signal bit to allow the residence time of the channel to be updated based on the decoded results. Otherwise, deassert the signal bit.
tx_ingress_timestamp_96b_data[][]	In	NUM_CHANNELS x 96	Specify the 96-bit ingress timestamp for the channel in the following format: • Bits 48 to 95: 48-bit seconds field. • Bits 16 to 47: 32-bit nanoseconds field. • Bits 0 to 15: 16-bit fractional nanoseconds field.
tx_ingress_timestamp_64b_data[][]	In	NUM_CHANNELS x 64	Carries the 64-bit ingress timestamp for the channel in the following format: • Bits 16 to 63: 48-bit nanoseconds field. • Bits 0 to 15: 16-bit fractional nanoseconds field.
tx_ingress_timestamp_format[]	In	NUM_CHANNELS	Assert the respective signal bit to use the 64-bit timestamp format for the channel. When deasserted, the 96-bit timestamp format is used. Align this signal to the start of the packet.

ToD Interface Signals

Table 7: ToD Interface Signals

The Master and Slave ToDs are one of the components in the design with the IEEE 1588v2 feature.

Signal	Direction	Width	Description
master_pulse_per_second	Out	1	Pulse per second, the output from the Master PPS module. This signal asserts for 10ms.
start_tod_sync[]	In	NUM_CHANNELS	Assert the respective signal bit to start TOD synchronization process for the channel. The synchronization process continues as long as this signal stays asserted.
pulse_per_second_10g[]	Out	NUM_CHANNELS	Pulse per second for each channel, the output from10G PPS module. This signals asserts for 10ms.
pulse_per_second_1g[]	Out	NUM_CHANNELS	Pulse per second for each channel, the output from1G PPS module. This signals asserts for 10ms.

Configuration Registers

You can access the configuration registers of the designs and their components through the Avalon-MM interface. All registers are 32 bits wide.

Table 8: Register Map

Byte Offset	Component
0x00_0000	Client Logic
0x01_0000	Master ToD
0x02_0000	Channel 0—Triple-Speed Ethernet IP core
0x02_0600	Channel 0—Slave ToD
0x03_0000	Channel 1—Triple-Speed Ethernet IP core
0x03_0600	Channel 1—Slave ToD
0x04_0000	Channel 2—Triple-Speed Ethernet IP core
0x04_0600	Channel 2—Slave ToD
... and so forth up until Channel 9	... and so forth.

Registers of the Triple-Speed Ethernet IP Core

You can access the register space of the MAC and PCS functions via the Avalon-MM interface of the Triple-Speed Ethernet IP. Because the reference designs used the Qsys tool to instantiate this IP core, the register space is accessed using byte addressing. The Triple-Speed Ethernet MegaCore Function User

Guide lists the registers in dword offsets for the MAC registers; word offsets for PCS registers. The MAC registers are 32 bits wide. The PCS registers, however, are 16 bits wide. The 16-bit PCS register occupies the lower 16 bits and the remaining 16 bits are initialized to 0.

The examples below show how byte offsets are derived for both the MAC and PCS registers.

Example 1—accesing the command_config register of the MAC.

Dword offset of the command_config register = 0x02.

Byte offset of the command_config register = (0x02 x 4) = 0x08.

Example 2—accessing the `if_mode` register of the PCS function.

Word offset of the `if_mode` register = 0x14.

Dword offset of the `if_mode` register = 0x80 + 0x14 = 0x94.

Byte offset of the `if_mode` register = 0x94 x 4 = 0x250.

Related Information

Triple-Speed Ethernet MegaCore Function User Guide
Describes the configuration registers of the Triple-Speed Ethernet IP Core.

ToD Registers

The Master and Slave ToDs are present in the design with the IEEE 1588v2 feature.

Table 9: ToD Register Map

Byte Offset	Name	Description	Access	Reset Value
0x00	`SecondsH`	• Bits [15:0]: The upper 16 bits of the second field. • Bits [31:16]: Reserved.	RW	0xffffffff
0x04	`SecondsL`	The lower 32 bits of the second field.	RW	0x0
0x08	`NanoSec`	The nanosecond field.	RW	0x0
0x10	`Period`	• Bits [15:0]: The time-of-day in fractional nanosecond. • Bits [19:16]: The time-of-day in nanosecond. • Bits [31:20]: Reserved	RW	0x0
0x14	`AdjustPeriod`	The offset adjustment period. • Bits [15:0]: The period in fractional nanosecond. • Bits [19:16]: The period in nanosecond. • Bits [31:20]: Reserved	RW	0x0

Byte Offset	Name	Description	Access	Reset Value
0x18	AdjustCount	• Bits [19:0]: The number of adjusted period in clock cycles. • Bits [31:20]: Reserved	RW	0x0

Packet Generator Registers

The packet generator is a sub-block of the traffic controller and the traffic monitor.

Table 10: Packet Generator Register Map

Byte Offset	Name	Description	Access	Reset Value
0x00	NUMPKTS	The total number of Ethernet packets that the traffic generator generates and transmits to the design components.	RW	0x0
0x04	RANDOMLENGTH	Enables random packet length up to the value of the PKTLENGTH register. • 0x00: Fixed length. • 0x01: Random length.	RW	0x0
0x08	RANDOMPAYLOAD	Enables random contents of the payload. • 0x00: Incremental. • 0x01: Random.	RW	0x0
0x0C	START	Start the generation of the Ethernet traffic by writing 0x01 to this register.	RW	0x0
0x10	STOP	Stops the generation of the Ethernet traffic by writing 0x01 to this register.	RW	0x0
0x14	MACSA0	The lower 32 bits of the source address.	RW	0x0
0x18	MACSA1	The upper 16 bits of the source address. The remaining 16 bits are not used.	RW	0x0
0x1C	MACDA0	The lower 32 bits of the destination address.	RW	0x0
0x20	MACDA1	The upper 16 bits of the source address. The remaining 16 bits are not used.	RW	0x0
0x24	TXPKTCNT	The number of packets that the traffic generator transmitted. Read this register when the traffic generator is not active, for example, when the testing has completed.	RO	0x0

Byte Offset	Name	Description	Access	Reset Value
0x34	PKTLENGTH	When random-sized packets are enabled, this register specifies the maximum payload length. Otherwise, it specifies the length of the packet to be generated.	RW	0x0

Traffic Monitor Registers

Table 11: Traffic Monitor Register Map

Byte Offset	Name	Description	Access	Reset Value
0x00	RXPKTCNT_EXPT	The number of packets that the traffic monitor expects to receive.	RW	0xffffffff
0x04	RXPKTCNT_GOOD	The number of good packets received by the traffic monitor.	RO	0x0
0x08	RXPKTCNT_BAD	The number of packets received with CRC error.	RO	0x0
0x0C	RXBYTECNT_LO32	The lower 32 bits of the counter that keeps track of the total number of bytes the traffic monitor received.	RO	0x0
0x10	RXBYTECNT_HI32	The upper 32 bits of the counter that keeps track of the total number of bytes the traffic monitor received.	RO	0x0
0x14	RXCYCLCNT_LO32	The lower 32-bit of the counter that keeps track of the total number of clock cycles required by the traffic monitor to receive the expected number of packets.	RO	0x0
0x18	RXCYCLCNT_HI32	The upper 32-bit of the counter that keeps track of the total number of clock cycles required by the traffic monitor to receive the expected number of packets.	RO	0x0
0x1C	RXCTRL_STATUS	Monitors the configuration and status register. • Bit [0]: Set to 1 to initialize all of the traffic monitor counters. • Bit [1]: Reserved. • Bit [2]: When set to 1, indicates that the traffic monitor has received the total number of expected packets. This bit is a read-only bit. • Bits [31:3]: Reserved.	RW	0x0

Document Revision History

Date	Version	Changes
May 2016	2016.05.13	Corrected the titles of the test cases.
June 2015	2015.06.24	Initial release.