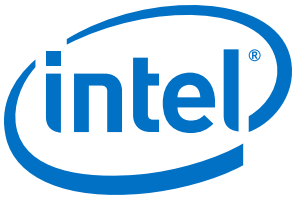


AN 757: 1G/2.5G Ethernet Design Examples

AN-757 | 2018.11.12
Latest document on the web: PDF | HTML

Contents

AN 757: 1G/2.5G Ethernet Design Examples

This document describes two design examples that demonstrate the 1G/2.5G Ethernet operations, one of which includes the IEEE 1588v2 feature.

Features

- Dual-speed Ethernet operation—1G and 2.5G.
- Support for two channels.
- Option to generate the design example with the IEEE 1588v2 feature.
- Testbench and simulation script.
- Tested with the Spirent TestCenter.

Hardware and Software Requirements

Intel uses the following hardware and software to test the design example in a Linux system:

- Intel® Quartus® Prime software
- ModelSim®-AE, ModelSim-SE, NCSim (Verilog only), and VCS simulators
- For hardware testing:
  - Arria® V GT FPGA Development Board (5AGTFD7K3F40I3)

Functional Description

The design example consists of various components. The following block diagrams shows the design components and the top-level signals of the design example.

Figure 1. Block Diagram—1G/2.5G Ethernet Design Example with IEEE 1588v2 Feature

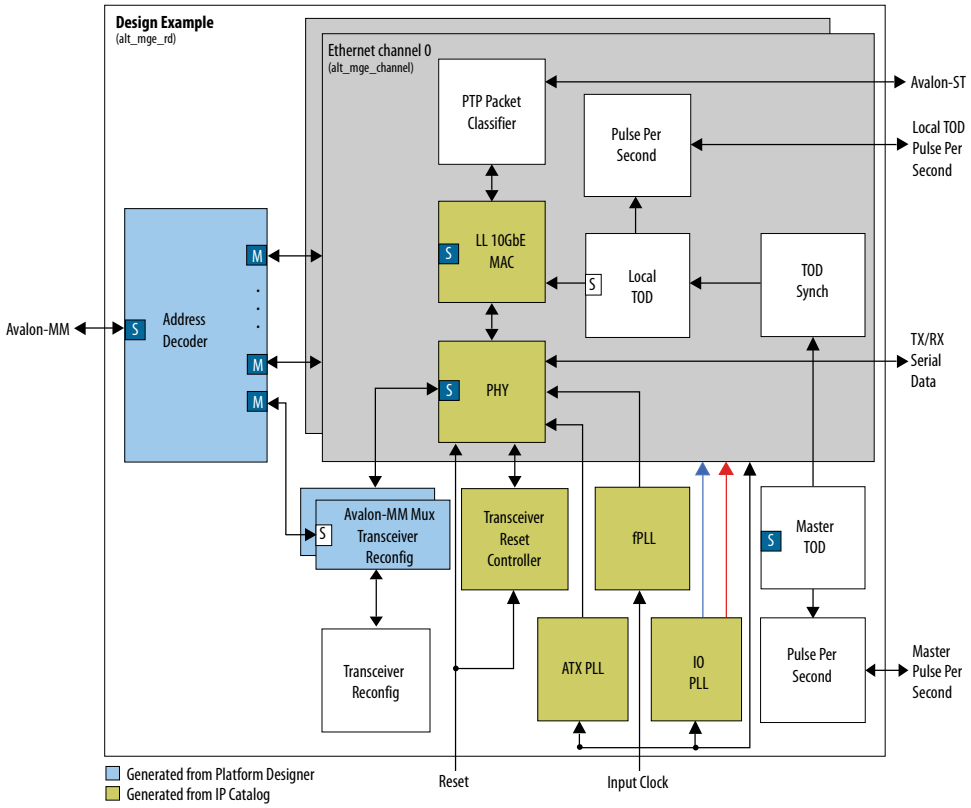
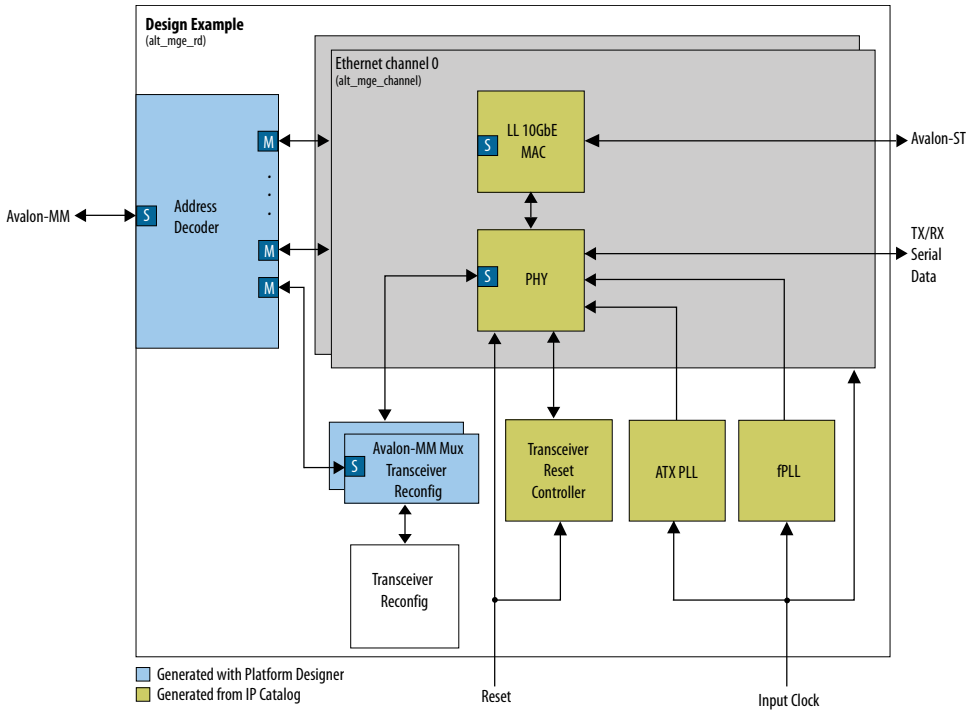


Figure 2. Block Diagram—1G/2.5G Ethernet Design Example without IEEE 1588v2 Feature



Design Components

Table 1. Design Components

Component	Description
LL 10GbE MAC	The Low Latency Ethernet 10G MAC Intel FPGA IP with the following configuration: • **Speed**: 1G/2.5G • **Datapath options**: TX & RX • **Enable ECC on memory blocks**: Not selected • **Enable supplementary address**: Selected • **Enable statistics collection**: Selected • **Statistics counters**: Memory-based • All **Legacy Ethernet 10G MAC Interfaces** options: Selected For the design example with the IEEE 1588v2 feature, the following additional parameters are configured: • **Enable time stamping**: Selected • **Enable PTP one-step clock support**: Selected • **Timestamp fingerprint width**: 4 • **Time Of Day format**: Enable both 96b and 64b Time of Day Format
PHY	The 1G/2.5G/5G/10G Multi-rate Ethernet PHY Intel FPGA IP.
Transceiver Reset Controller	The Transceiver PHY Reset Controller IP core. Resets the transceiver.
Avalon®-MM Mux Transceiver Reconfig	Provides the transceiver reconfig block and system console access to the PHY's Avalon-MM interface.
Transceiver Reconfig	Reconfigures the transceiver channel speed from 1 Gbps to 2.5 Gbps, and vice versa.

continued...

Component	Description
ATX PLL	Generates a TX serial clock for the Arria V 2.5G transceiver.
fPLL	Generates a TX serial clock for the Arria V 1G transceiver.
Design Components for the IEEE 1588v2 Feature	
IO PLL	Generates the clocks for the 1588 design components.
Master TOD	The master time-of-day (TOD) for all channels.
TOD Synch	Synchronizes the master TOD to all local TODs.
Local TOD	The TOD for each channel.
Master Pulse Per Second	Returns pulse per second (pps) for all channels.
Pulse Per Second	Returns pulse per second (pps) for each channel.
PTP Packet Classifier	Decodes the packet type of incoming PTP packets and returns the decoded information to the LL 10GbE MAC Intel FPGA IP core.

Reset Scheme

The global reset signal of the design example is asynchronous and active-low. Asserting this signal resets all channels and their components. Upon power-up, reset the design example.

Figure 3. Reset Scheme for the 1G/2.5G Ethernet Design Example without IEEE 1588v2 Feature

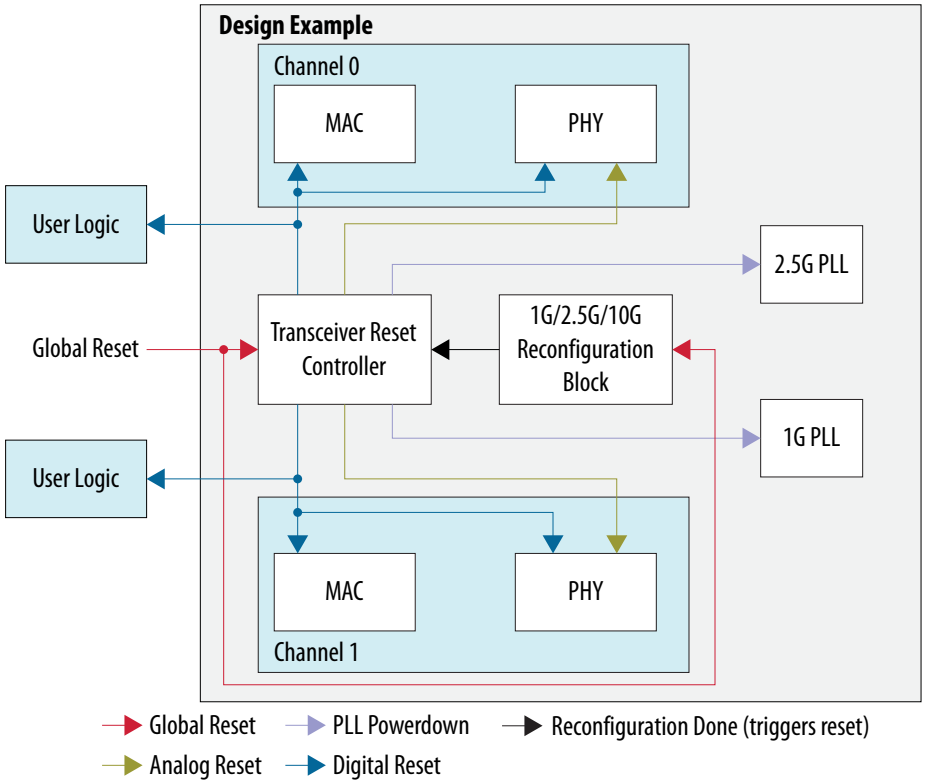
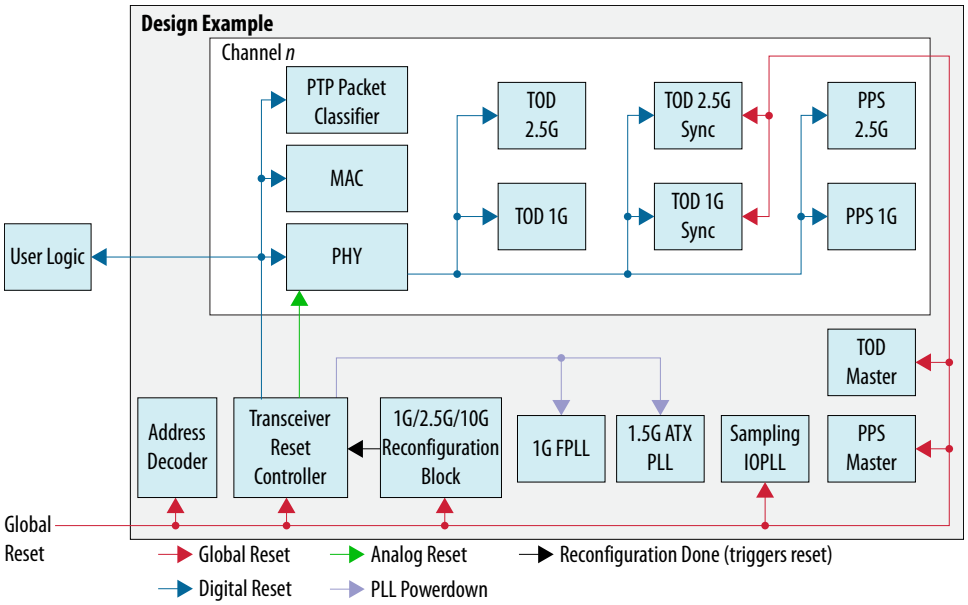


Figure 4. Reset Scheme for the 1G/2.5G Ethernet Design Example with IEEE 1588v2 Feature



Clocking Scheme

Figure 5. Clocking Scheme for the 1G/2.5G Ethernet Design Example with IEEE 1588v2 Feature

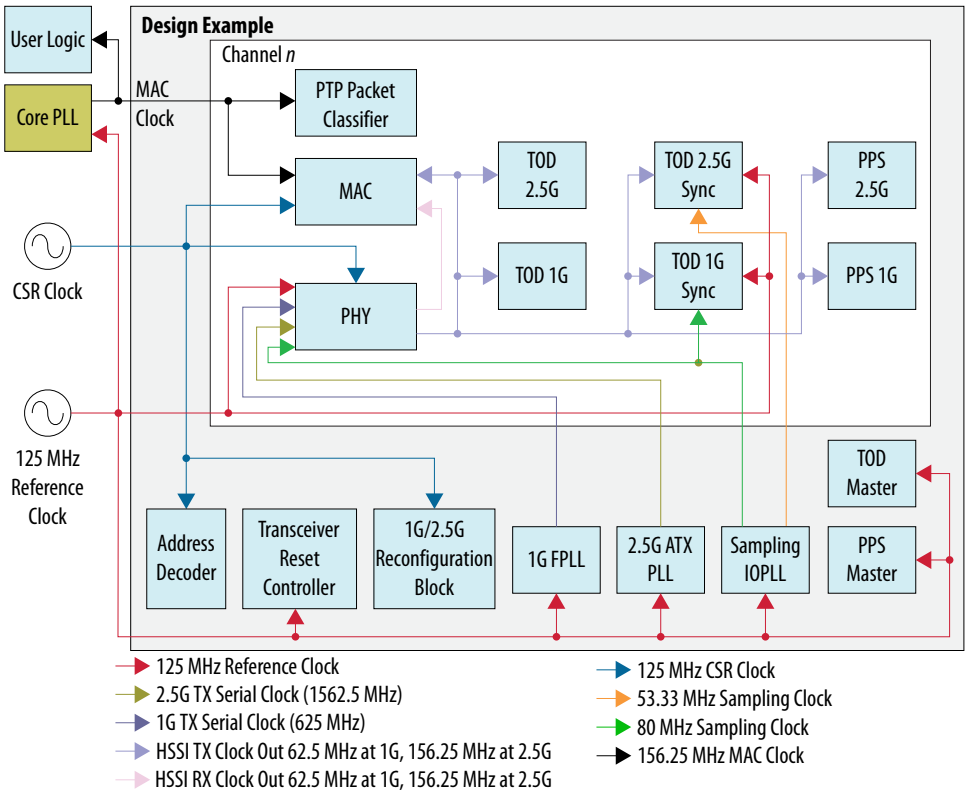
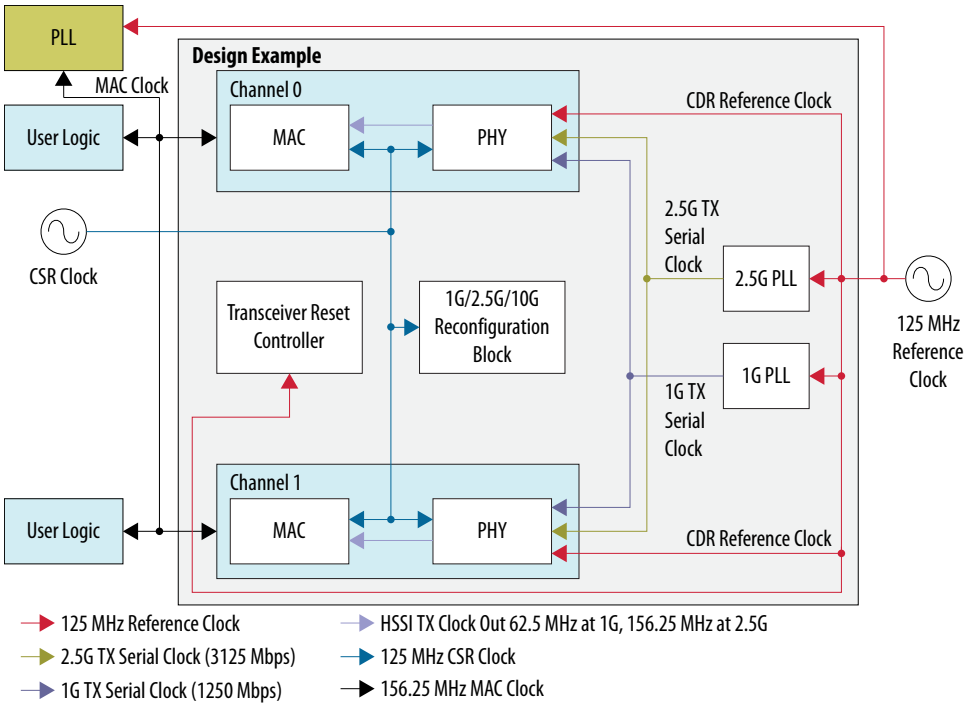


Figure 6. Clocking Scheme for the 1G/2.5G Ethernet Design Example without IEEE 1588v2 Feature



Using the Design Example

To start using the design example, follow these steps:

1. Unzip the design files in the project directory.

   ```
   tar -zxvf LL_Ethernet_1G_2p5G_A5.tar.gz
   ```

2. Change directory to the following directory:
   - `LL10G_A5_1G_2_5G` for the design example without the IEEE 1588v2 feature, or
   - `LL10G_A5_1G_2_5G_1588v2` for the design example with the IEEE 1588v2 feature.
3. Launch the Intel Quartus Prime software and open the project file, `altera_eth_top.qpf`.
4. Select **Processing** > **Start Compilation** to compile the design example.

Related Information

Low Latency Ethernet 1G/2.5G Design Files for Arria V

Simulation

Testbench

Figure 7. Block Diagram of the Testbench

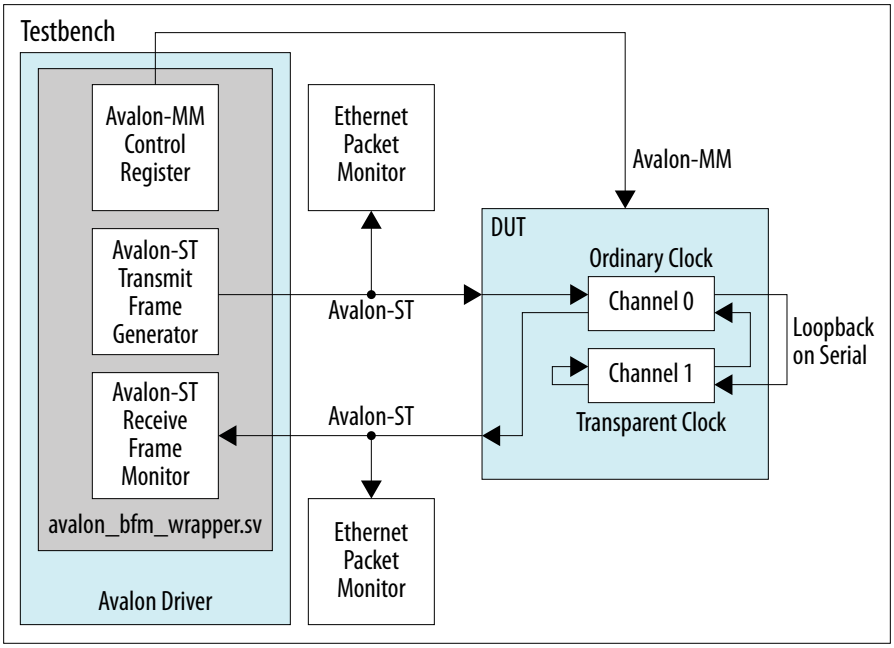


Table 2. Testbench Components

Component	Description
Device under test (DUT)	The design example.
Avalon driver	Consists of Avalon-ST master bus functional models (BFMs). This driver forms the TX and RX paths. The driver also provides access the Avalon-MM interface of the DUT.
Ethernet packet monitors	Monitor TX and RX datapaths, and display the frames in the simulator console.

Using the Simulator

To simulate the design example using the ModelSim simulator, follow these steps:

1. Ensure that the QUARTUS_ROOTDIR environment variable points to the Intel Quartus Prime installation path.
2. Change the working directory to <*project directory*>/`simulation/ed_sim/mentor`.
3. Launch the ModelSim simulator.
4. Run the following command to set up the required libraries, compile the functional simulation model, and exercise the simulation model with the provided testbench:

```
do tb_run.tcl
```

Test Case—Design Example without the IEEE 1588v2 Feature

The simulation test case performs the following steps:

1. Starts up the example design with an operating speed of 2.5G.
2. Configures the MAC, PHY, and FIFO buffer for both channels.
3. Waits until the design example asserts the `channel_tx_ready` and `channel_rx_ready` signals for both channels.
4. Sends the following packets:
   - 64-byte packet
   - 1518-byte packet
   - 100-byte packet
5. Repeats steps 2 to 4 for 1G.

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, all statistics error counters are zero, and the RX MAC statistics counters are equal to the TX MAC statistics counters.

Hardware Testing

Follow these steps to compile and test the design in the supported Altera development kit:

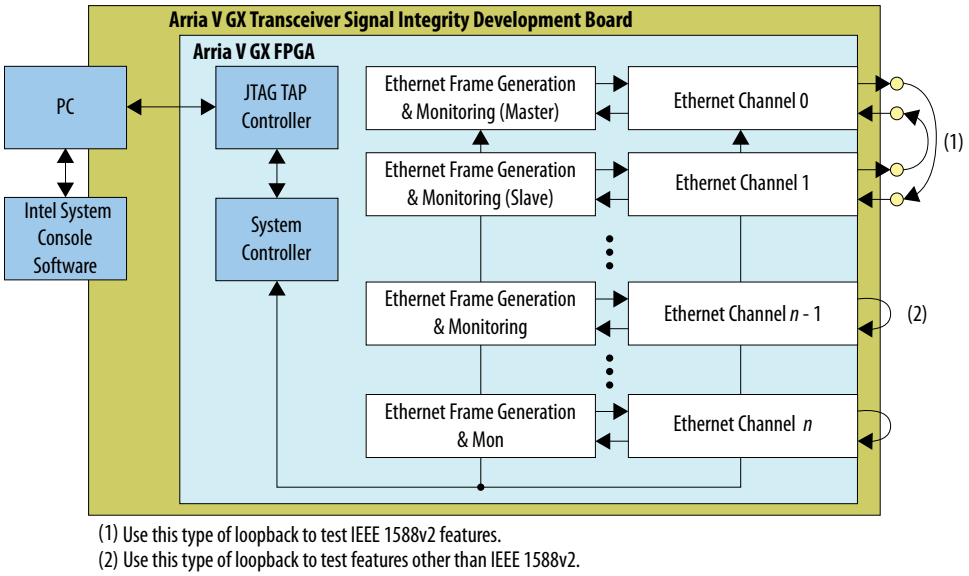
1. Launch the Quartus Prime software and compile the design (**Processing** > **Start Compilation**).

   The timing contraints for the design example and the design components are automatically loaded during compilation.
2. Connect the development board to the host computer.
3. Configure the FPGA on the development board using the generated `.sof` file (**Tools** > **Programmer**).
4. Launch the Clock Control tool, which is part of the development kit, and set new frequencies for the design example.

   Note: The chapter that describes the design example states the frequencies to set.
5. Reset the system by pressing the `PB0` push button.
6. In the Quartus Prime software, launch the system console (**Tools** > **System Debugging Tools** > **System Console**).
7. Change the working directory to *<Example Design>*`\hwtesting\system_console`.
8. Initialize the design command list by running this command, `source main.tcl`.
9. You can now run any of the predefined hardware tests from the System Console. Observe the test results displayed.

Setup

Figure 8. Hardware Setup



(1) Use this type of loopback to test IEEE 1588v2 features.
(2) Use this type of loopback to test features other than IEEE 1588v2.

Test Procedure

Follow these steps to test the design examples in hardware:

1. Run the following command in the system console to start the test.

   TEST_EXT_LB <*channel*> <*speed*> <*burst_size*>

   Example: `TEST_EXT_LB 0 2.5G 1000000000`

Table 3. Command Parameters

Parameter	Valid Values	Description
`channel`	0, 1	The channel number to test.
`speed`	1G, 2.5G	The PHY speed.
`burst_size`	An integer value	The number of packets to generate for the test.

2. When the test is completed, observe the output displayed. The following diagrams show excerpts of the output, which shows that the Ethernet packet monitor block receives the same number of packets generated without error, and the TX and RX statistics counters.

Figure 9. Sample Test Output—Ethernet Packet Monitor

```
TEST_INFO: 2.5G board trace Loopback Test
CONFIGURE CHANNEL 1
Configure to 2.5G
Setting up mac with a basic working config
Setting 0xC5C4 into rxmac primary address Reg-1
Setting 0xC3C2C1C0 into rxmac primary address Reg-0
Enabling: crc insertion in tx mac
Enabling: pad and crc stripping in rx mac
Setting 1518 into rxmac max frame length
Setting 1518 into txmac max frame length
Clearing mac stats registers
Select std ethernet traffic controller
Disable Avalon ST Loopback


============================================================
       B E G I N     C O N F I G U R A T I O N
============================================================
payload length = variable (random) ....
payload bytes = random bytes ....
burst size =  1000000000 ....
payload length =  1518 ....
frame source addres field = F0F1F2F3F4F5 ....
frame destination addres field = C5C4C3C2C1C0 ....
reseting monitor Packet Counters
number of Packets Expected By Monitor = 0x3b9aca00
burst being injected into device ....
    -- MONITOR processing frames received .....


    -- MONITOR Received  Packet#   4067046]

    -- MONITOR Received  Packet#   8143458]
```

Figure 10. Sample Test Output—Statistics Counters

```
=================================================================
|  MAC TX STATS REGISTER CHECK
=================================================================
|# FRAMES_RECEIVED_WITH_ERROR               = 0
|# UNICAST_FRAMES_WITH_ERROR                = 0
|# MULTICAST_FRAMES_RECEIVED_WITH_ERROR     = 0
|# BRDCAST_FRAMES_WITH_ERROR                = 0
|# FRAMES_RECEIVED_WITH_ONLY_CRCERROR       = 0
|# VALID_LENGTH_FRAMES_WITH_CRC_ERROR       = 0
|# JABBER_FRAMES                            = 0
|# FRAGMENTED_FRAMES                        = 0
|# INVALID_FRAMES_RECEIVED                  = 0
|# FRAMES_RECEIVED_GOOD                     = 1000000000
|# PAUSE_FRAMES_RECEIVED                    = 0
|# UNICAST_CONTROL_FRAMES                   = 0
|# MULTICAST_CONTROL_FRAMES                 = 0
|# UNICAST_FRAMES_RECEIVED_GOOD             = 0
|# MULTICAST_FRAMES_RECEIVED_GOOD           = 1000000000
```

Interface Signals

Clock and Reset Interface Signals

Table 4. Clock and Reset Interface Signals

Signal	Direction	Width	Description
csr_clk	In	1	125-MHz configuration clock for the Avalon-MM interface.
mac_clk	In	1	156.25-MHz clock for the Avalon-ST interface. This clock must have 0 ppm frequency difference to refclk.
refclk	In	1	125-MHz reference clock for the TX PLLs.
rx_pma_clkout	Out	1	Recovered clock from CDR.
reset	In	1	Asserting this signal resets the whole design example. Asynchronous and active low signal.
tx_digital_reset	Out	2	Asserting this signal resets the TX datapath. Asynchronous and active low signal.
rx_digital_reset	Out	2	Asserting this signal resets the RX datapath. Asynchronous and active low signal.

Avalon-MM Interface Signals

Use the prefixes to identify the Avalon-MM interface signals for the following design components:

- MAC—csr_mac_*
- PHY—csr_phy_*
- Transceiver Reconfiguration—csr_rcfg_*
- Arria V Native PHY Reconfiguration—csr_native _phy_rcfg_*
- Master TOD—csr_master_tod_*

Table 5. Avalon MM Interface Signals

Signal	Direction	Width	Description
csr_mac_write[]	In	2	Assert the signal to request a write. For the MAC, PHY, and Master TOD, assert bit 0 to request a write to channel 0; bit 1 for channel 1.
csr_phy_write[]		2	
csr_rcfg_write		1	
csr_native _phy_rcfg_write		1	
csr_master_tod_write[]		2	
csr_mac_read[]	In	2	Assert the signal bit to request a read. For the MAC, PHY, and Master TOD, assert bit 0 to request a read from channel 0; bit 1 for channel 1.
csr_phy_read[]		2	
csr_rcfg_read		1	
csr_native _phy_rcfg_read		1	
csr_master_tod_read[]		2	

continued...

Signal	Direction	Width	Description
csr_mac_address[][]	In	[2][10]	Use this bus to specify the register address you want to read from or write to.
csr_phy_address[][]		[2][5]	
csr_rcfg_address[]		2	
csr_native_phy_rcfg_address[]		10	
csr_master_tod_address[][]		[2][4]	
csr_mac_writedata[][]	In	[2][32]	Data to be written to the specified register.
csr_phy_writedata[][]		[2][32]	
csr_rcfg_writedata[]		32	
csr_native_phy_rcfg_writedata[]		32	
csr_master_tod_writedata[][]		[2][32]	
csr_mac_readdata[][]	Out	[2][32]	Data read from the specified register.
csr_phy_readdata[][]		[2][32]	
csr_rcfg_readdata[]		32	
csr_native_phy_rcfg_readdata[]		32	
csr_master_tod_readdata[][]		[2][32]	
csr_mac_waitrequest[]	Out	2	When asserted, this respective signal bit indicates that the channel is busy and not ready to accept any read or write requests.
csr_phy_waitrequest[]		2	
csr_native_phy_rcfg_waitrequest		2	
csr_master_tod_waitrequest[]		2	

Avalon-ST Interface Signals

Table 6. Avalon-ST Interface Signals

Signal	Direction	Width	Description
avalon_st_tx_startofpacket[]	In	2	Assert this signal to indicate the beginning of the TX data.
avalon_st_tx_endofpacket[]	In	2	Assert this signal to indicate the end of the TX data.
avalon_st_tx_valid[]	In	2	Assert this signal to indicate that avalon_st_tx_data[] and other signals on this interface are valid.
avalon_st_tx_ready[]	Out	2	When asserted, indicates that the MAC IP core is ready to accept data. The reset value of this signal is non-deterministic.
avalon_st_tx_error[]	In	2	Assert this signal to indicate that the current TX packet contains errors.

continued...

Signal	Direction	Width	Description
avalon_st_tx_data[][]	In	[2][32]	TX data from the client.
avalon_st_tx_empty[][]	In	[2][2]	Use this signal to specify the number of empty bytes in the cycle that contain the end of the TX data. 0x0=All bytes are valid. 0x1=The last byte is invalid. 0x2=The last two bytes are invalid. 0x3=The last three bytes are invalid.
avalon_st_rx_startofpacket[]	Out	2	When asserted, indicates the beginning of the RX data.
avalon_st_rx_endofpacket[]	Out	2	When asserted, indicates the end of the RX data.
avalon_st_rx_valid[]	Out	2	When asserted, indicates that the avalon_st_rx_ data[] signal and other signals on this interface are valid.
avalon_st_rx_ready[]	In	2	Assert this signal when the client is ready to accept data.
avalon_st_rx_error[][]	Out	[2][6]	When set to 1, the respective bits indicate an error type: • Bit 0—PHY error. For 10 Gbps, the data on xgmii_rx_data contains a control error character (FE). For 10 Mbps,100 Mbps,1 Gbps, gmii_rx_err or mii_rx_err is asserted. • Bit 1—CRC error. The computed CRC value differs from the received CRC. • Bit 2—Undersized frame. The receive frame length is less than 64 bytes. • Bit 3—Oversized frame. The receive frame length is more than MAX_FRAME_SIZE. • Bit 4—Payload length error. The actual frame payload length is different from the value in the length/type field. • Bit 5—Overflow error. The receive FIFO buffer is full while it is still receiving data from the MAC IP core.
avalon_st_rx_data[][]	Out	[2][32]	RX data to the client.
avalon_st_rx_empty[][]	Out	[2][2]	Contains the number of empty bytes during the cycle that contain the end of the RX data.
avalon_st_tx_status_valid[]	Out	2	When asserted, this signal qualifies the avalon_st_txstatus_data[] and avalon_st_txstatus_error[] signals.
avalon_st_tx_status_data[][]	Out	[2][40]	Contains information about the TX frame. • Bits 0 to 15: Payload length. • Bits 16 to 31: Packet length. • Bit 32: When set to 1, indicates a stacked VLAN frame. • Bit 33: When set to 1, indicates a VLAN frame. • Bit 34: When set to 1, indicates a control frame. • Bit 35: When set to 1, indicates a pause frame. • Bit 36: When set to 1, indicates a broadcast frame. • Bit 37: When set to 1, indicates a multicast frame. • Bit 38: When set to 1, indicates a unicast frame. • Bit 39: When set to 1, indicates a PFC frame.
avalon_st_tx_status_error[][]	Out	[2][7]	When set to 1, the respective bit indicates the following error type in the RX frame. • Bit 0: Undersized frame. • Bit 1: Oversized frame. • Bit 2: Payload length error.

continued...

Signal	Direction	Width	Description
			• Bit 3: Unused. • Bit 4: Underflow. • Bit 5: Client error. • Bit 6: Unused. The error status is invalid when an overflow occurs.
avalon_st_rx_status_valid[]	Out	2	When asserted, this signal qualifies the avalon_st_txstatus_data[] and avalon_st_txstatus_error[] signals. The MAC IP core asserts this signal in the same clock cycle the avalon_st_rx_endofpacket signal is asserted.
avalon_st_rx_status_data[][]	Out	[2][40]	Contains information about the RX frame. • Bits 0 to 15: Payload length. • Bits 16 to 31: Packet length. • Bit 32: When set to 1, indicates a stacked VLAN frame. • Bit 33: When set to 1, indicates a VLAN frame. • Bit 34: When set to 1, indicates a control frame. • Bit 35: When set to 1, indicates a pause frame. • Bit 36: When set to 1, indicates a broadcast frame. • Bit 37: When set to 1, indicates a multicast frame. • Bit 38: When set to 1, indicates a unicast frame. • Bit 39: When set to 1, indicates a PFC frame.
avalon_st_rx_status_error[][]	Out	[2][7]	When set to 1, the respective bit indicates the following error type in the RX frame. • Bit 0: Undersized frame. • Bit 1: Oversized frame. • Bit 2: Payload length error. • Bit 3: Unused. • Bit 4: Underflow. • Bit 5: Client error. • Bit 6: Unused. The error status is invalid when an overflow occurs.
avalon_st_pause_data[][]	In	[2][2]	This signal takes effect when the register bits, tx_pauseframe_enable[2:1], are both set to the default value 0. Set this signal to the following values to trigger the corresponding actions. • 0x0: Stops pause frame generation. • 0x1: Generates an XON pause frame. • 0x2: Generates an XOFF pause frame. The MAC IP core sets the pause quanta field in the pause frame to the value in the tx_pauseframe_quanta register. • 0x3: Reserved.

PHY Interface Signals

Table 7. PHY Interface Signals

Signal	Direction	Width	Description
rx_serial_data[]	In	2	RX serial input data
tx_serial_data[]	Out	2	TX serial output data
led_link[]	Out	2	Asserted when the link synchronization for 1GbE or 2.5GbE is successful.

continued...

Signal	Direction	Width	Description
led_char_err[]	Out	2	Asserted when a 10-bit character error is detected in the RX data.
led_disp_err[]	Out	2	Asserted when a 10-bit running disparity error is detected in the RX data.
led_an[]		2	Asserted when auto-negotiation is completed.
channel_tx_ready[]	Out	2	The signal bit is asserted when the TX datapath of the channel is ready for data transmission.
channel_rx_ready[]	Out	2	The signal bit is asserted when the RX datapath of the channel is ready for data transmission.

IEEE 1588v2 Timestamp Interface Signals

Table 8. IEEE 1588v2 Timestamp Interface Signals

Signal	Direction	Width	Description
tx_egress_timestamp_96b_valid[]	Out	2	When asserted, this signal qualifies the timestamp, tx_egress_timestamp_96b_data[], and the fingerprint, tx_egress_timestamp_96b_fingerprint[], of the TX frame.
tx_egress_timestamp_96b_data[][]	Out	[2][96]	Carries the 96-bit egress timestamp 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_egress_timestamp_96b_fingerprint[][]	Out	[2] [TSTAMP_FP_WIDTH]	Specifies the fingerprint of the TX frame that the 96-bit timestamp is for.
tx_egress_timestamp_64b_valid[]	Out	2	When asserted, this signal qualifies the timestamp, tx_egress_timestamp_64b_data[], and the fingerprint, tx_egress_timestamp_64b_fingerprint[], of the TX frame.
tx_egress_timestamp_64b_data[][]	Out	[2][64]	Carries the 64-bit egress timestamp in the following format: • Bits 16 to 63: 48-bit nanoseconds field • Bits 0 to 15: 16-bit fractional nanoseconds field
tx_egress_timestamp_64b_fingerprint[][]	Out	[2] [TSTAMP_FP_WIDTH]	Specifies the fingerprint of the TX frame that the 64-bit timestamp is for.
rx_ingress_timestamp_96b_valid[]	Out	2	When asserted, this signal qualifies the timestamp, rx_ingress_timestamp_96b_data[]. The MAC IP core asserts this signal in the same clock cycle it asserts avalon_st_rx_startofpacket.
rx_ingress_timestamp_96b_data[][]	Out	[2][96]	Carries the 96-bit ingress timestamp in the following format:

continued...

Signal	Direction	Width	Description
			• 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
rx_ingress_timestamp_64b_valid[]	Out	2	When asserted, this signal qualifies the timestamp, rx_ingress_timestamp_64b_data[]. The MAC IP core asserts this signal in the same clock cycle it asserts avalon_st_rx_startofpacket.
rx_ingress_timestamp_64b_data[][]	Out	[2][64]	Carries the 64-bit ingress timestamp in the following format: • Bits 16 to 63: 48-bit nanoseconds field • Bits 0 to 15: 16-bit fractional nanoseconds field

Packet Classifier Interface Signals

Table 9. Packet Classifier Interface Signals

Signal	Direction	Width	Description
tx_egress_timestamp_request_in_valid[]	In	2	Assert this signal to request timestamping for the TX frame. This signal must be asserted in the same clock cycle avalon_st_tx_startofpacket is asserted.
tx_egress_timestamp_request_in_fingerprint[][]	In	[2][TSTAMP_FP_WIDTH]	Use this bus to specify the fingerprint that validates the timestamp for the incoming packet.
clock_operation_mode_mode[][]	In	[2][2]	Use this signal to specify 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	2	Use this signal to specify whether or not a packet contains CRC. • 0: Packet contains CRC • 1: Packet does not contain CRC
tx_ingress_timestamp_valid[]	In	2	Indicates whether or not the residence time can be updated. • 0: Prevents update for residence time • 1: Allows update for residence time based on decoded results When this signal is deasserted, the tx_etstamp_ins_ctrl_out_residence_ti me_update signal also gets deasserted.

continued...

Signal	Direction	Width	Description
tx_ingress_timestamp_96b_data[][]	In	[2][96]	96-bit format of ingress timestamp that holds the data so that the output can align with the start of an incoming packet.
tx_ingress_timestamp_64b_data[][]	In	[2][64]	64-bit format of ingress timestamp that holds the data so that the output can align with the start of an incoming packet.
tx_ingress_timestamp_format[]	In	2	The format of the timestamp for calculating the residence time. • 0: 96 bits • 1: 64 bits This signal must be aligned to the start of an incoming packet.

TOD Interface Signals

Table 10. TOD Interface Signals

Signal	Direction	Width	Description
master_pulse_per_second	Out	1	Pulse per second (PPS) from the master PPS module. The signal stay asserted for 10 ms.
start_tod_sync[]	In	2	Use this signal to trigger the TOD synchronization process. The time of day of the local TOD is synchronized to the time of day of the master TOD. The synchronization process continues as long as this signal remains asserted.
pps[]	Out	2	PPS from the 1G/2.5G PPS module of channel *n*. The signal stay asserted for 10ms.

Configuration Registers

You can access the 32-bit configuration registers of the design components through the Avalon-MM interface.

Table 11. Register Map

Byte Offset	Block
0x00_0000	Transceiver Reconfiguration
0x00_4000	Reserved
Channel 0	
0x01_0000	MAC
0x01_8000	PHY
0x01_A000	Native PHY Reconfiguration
Channel 1	
0x02_0000	MAC
0x02_8000	PHY
0x02_A000	Native PHY Reconfiguration
Traffic Controller	
0x10_0000	Traffic Controller

Transceiver Reconfiguration

Table 12. Transceiver Reconfiguration Register Map

Word Offset	Name	Bits	Description	Access	HW Reset
0x00	`logical_channel_number`	[9:0]	The logical number of the reconfiguration block.	RW	0x000
		[31:10]	Reserved	—	—
0x01	`control`	[1:0]	Specify the new operating speed: • 00: 1 Gbps • 01: 2.5 Gbps • 10: Reserved • 11: 10 Gbps	RW	0x00
		[15:2]	Reserved	—	0x000
		[16]	Writing 1 to this bit when it is 0 starts the reconfiguration process. The bit clears when the process is completed.	RWC	0x0
		[31:17]	Reserved	—	0x000000
0x02	`status`	[0]	When set to 1, indicates the reconfiguration process is in progress.	RO	0x0
		[31:1]	Reserved	—	—

Document Revision History for AN 757: 1G/2.5G Ethernet Design Examples

Document Version	Changes
2018.12.11	• Renamed the document as AN 757: 1G/2.5G Ethernet Design Examples. • Updated the *Hardware and Software Requirements* topic. • Corrected a broken link to the design files in the *Using the Design Example* topic. • Made minor editorial updates to the document.

Date	Version	Changes
June 2017	2017.06.19	• Rebranded to Intel. • Changed Intel Arria 10 references to Arria V.
May 2016	2016.05.20	• Updated the Hardware Setup diagram. • Added Arria V support.
February 2016	2016.02.05	Initial release