Scalable 10G Ethernet MAC using 1G/10G PHY

2016-05-13

AN-705 Subscribe Send Feedback

The following design examples demonstrate Altera 10G Ethernet MAC using 1G/10G PHY.

- Scalable Multispeed 10G Ethernet MAC using 1G/10G PHY without IEEE 1588v2
- Scalable Multispeed 10G Ethernet MAC using 1G/10G PHY with IEEE 1588v2

Related Information

- **Scalable Multispeed 10G Ethernet MAC using 1G/10G PHY without IEEE 1588v2**
- **Scalable Multispeed 10G Ethernet MAC using 1G/10G PHY with IEEE 1588v2**

Features

These design examples offer the following features:

- Support multi speed operation of 10 Megabits per second (Mbps) to 10 Gigabits per second (Gbps) with 1G/10G PHY.
- Support scalability from 1 to 12 channels Ethernet MAC and PHY.
- Provide packet monitoring system on transmit and receive data paths and report Ethernet MAC statistics counters for transmit and receive datapaths.
- Support testing using different types of Ethernet packet transfer with or without IEEE 1588v2 features.

Parameters

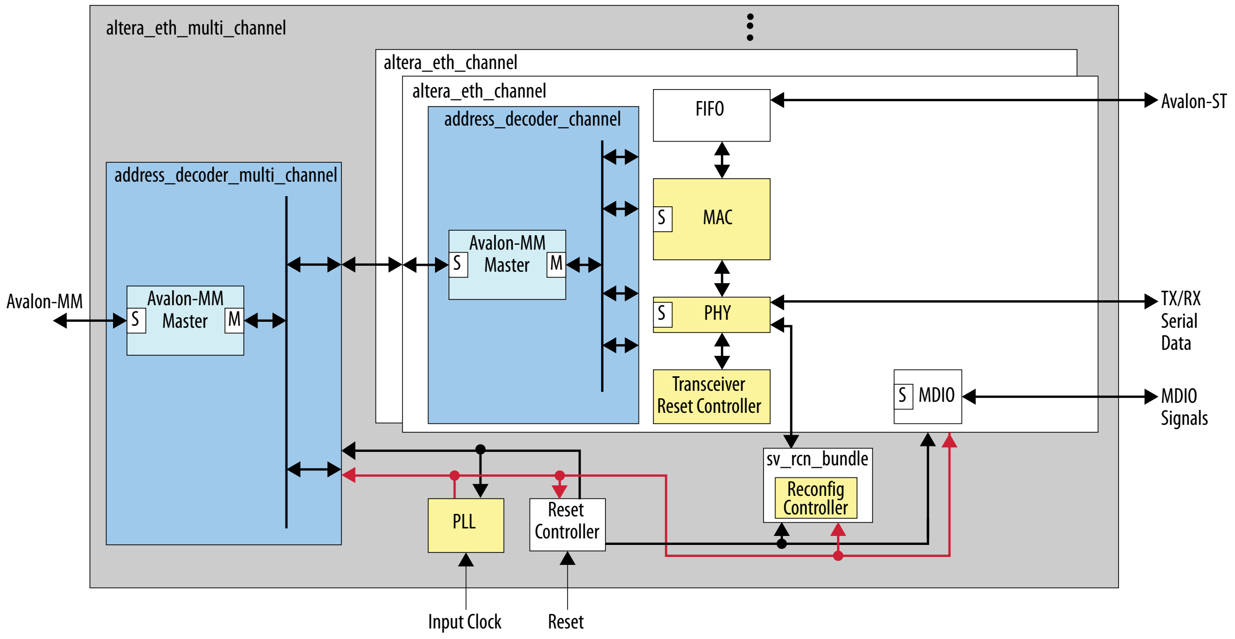
Table 1: Parameters for Design Example Customization

Parameter	Description	Default Value
`NUM_CHANNELS`	Specify the number of channels of 1-Gbps Ethernet(GbE)/ 10GbE that will be instantiated in the design example. Range from 1 to 12.	2
`MDIO_MDC_CLOCK_DIVISOR`	Use this parameter to set the management data input/ output (MDIO) clock divisor. Range from 8 to 64.	32
`SHARED_REFCLK_EN`	Use this parameter to enable the sharing of reference clock `refclk` between all channels. • 0 : disable sharing • 1 : enable sharing	1

Parameter	Description	Default Value
`SV_RCN_BUNDLE_MODE`	Use this parameter to set the Reconfiguration Bundle operation mode. • 0: 10GBaseKR • 1: 1G10G without IEEE 1588v2 • 2: 1G10G with IEEE 1588v2 **Note:** 0 and 1 are disabled in the design example with IEEE 1588v2.	• 1 (without IEEE 1588v2) • 2 (with IEEE 1588v2)
`FIFO_OPTIONS`	Use this parameter to enable the FIFO in between user Avalon-ST and MAC interface. • 0: disable FIFO • 1: enable SC FIFO • 2: enable DC FIFO • 3: enable SC + DC FIFO **Note:** This parameter is available only for design example without IEEE 1588v2.	1
`TSTAMP_FP_WIDTH`	Use this parameter to set the timestamp fingerprint width which follows the setting in 1G/10GbE MAC. You must regenerate the MAC IP if this parameter is changed. Enter the new width value in MAC IP regeneration page. **Note:** This parameter is available only for design example with IEEE 1588v2.	4

Block Diagrams

Figure 1: Block Diagram for Design Example without IEEE 1588v2

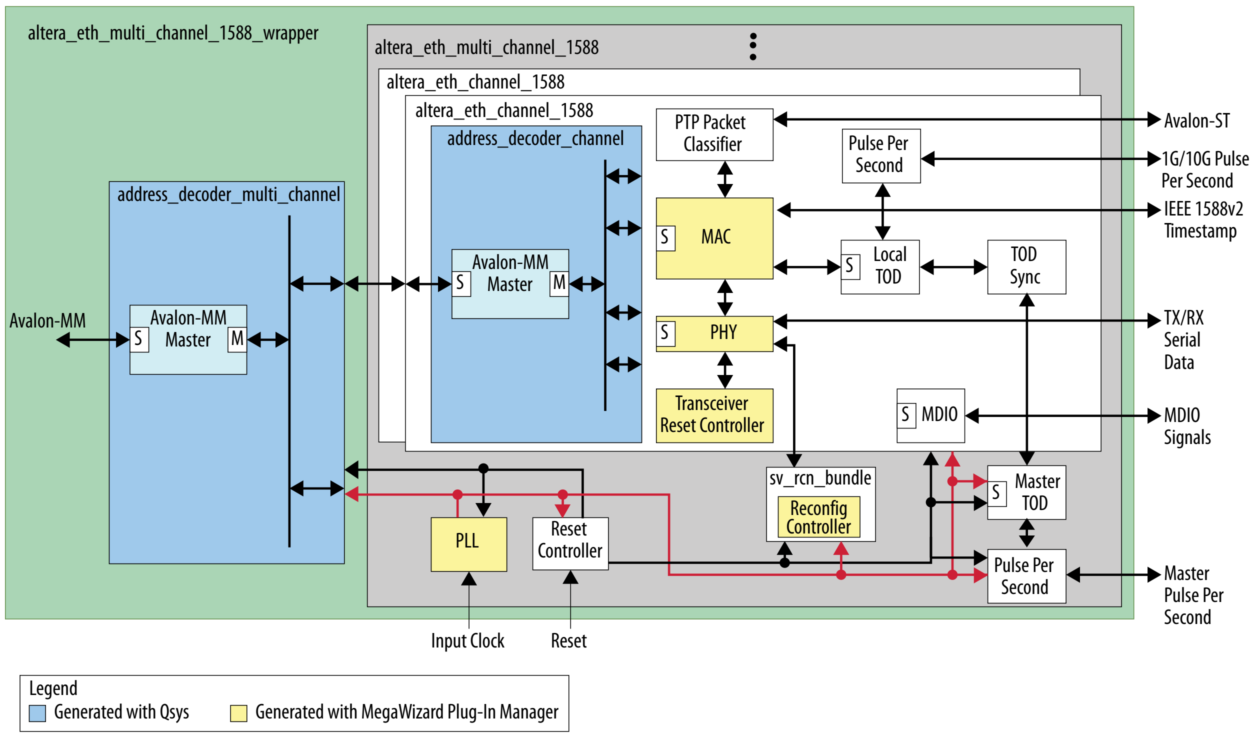


Legend
🟦 Generated with Qsys 🟨 Generated with MegaWizard Plug-In Manager

Figure 2: Block Diagram for Design Example with IEEE 1588v2

You can use one of the following module:

- **altera_eth_multi_channel_1588_wrapper**—includes `address_decoder_multi_channel` block which consolidate `address_decoder_channel` of all channels and Master TOD into a single Avalon-MM interface.
- **altera_eth_multi_channel_1588**—exposes Avalon-MM interface of `address_decoder_channel` of every channels and Master TOD to provide more flexible access and register map address space allocation.



Components

Table 2: Design Examples Components

Component	Design Example without IEEE 1588v2	Design Example with IEEE 1588v2
10G MAC	Ethernet MAC IP core.	
1G/10G PHY	Altera 1G/10G and 10GBASE-KR PHY IP.	
MDIO	Provides MDIO interface to connect Ethernet MAC to external PHY.	
Address decoder channel	Address decoder module for each component within the channel, for example, MAC and PHY.	

Component	Design Example without IEEE 1588v2	Design Example with IEEE 1588v2
Address decoder multi-channel	Address decoder module for all channels and components within multi-channel level, for example Master TOD.	
Reset controller	Reset modules which handle reset synchronization for the components in the design example.	
Master PLL	Generates clocks for all the components in the design example.	
Reconfig Controller	Reconfigure the transceiver channel speed from 1G to 10G and vice versa.	
Master Time-of-Day (TOD)	—	Provides a master TOD for all channels.
TOD Sync	—	Module to synch time of day from Master TOD to Local TOD for all channels.
Local TOD	—	TOD module in each channel.
Master Pulse Per Second module	—	Returns pulse per second (pps) to user for all channels.
1G/10G Pulse Per Second module	—	Returns pulse per second (pps) to user in each channel.
PTP packet classifier	—	Decodes the packet type of incoming PTP packets and returns the decoded information to the Ethernet MAC.
FIFO	Avalon Streaming (Avalon-ST) single-clock or dual-clock FIFO that buffers the receive and transmit data between the MAC and client.	—

Requirements

Altera uses the following hardware and software to test the design examples and testbench in Linux platform:

- Altera Complete Design Suite (ACDS) version 16.0
- Stratix V GX Transceiver SI Development Board (EP5SGXEA7N2F40C2N)
- ModelSim-SE 10.3d
- Synopsys VCS Version I-2014.03-SP1

Design Example Walkthrough

The following design examples come with pre-generated RTL files for two channels:

Design Example	File Name
Scalable Multispeed 10G Ethernet MAC using 1G/10G PHY without IEEE 1588v2	**altera_eth_1g10g_lineside.tar.gz**
Scalable Multispeed 10G Ethernet MAC using 1G/10G PHY with IEEE 1588v2	**altera_eth_1g10g_lineside_w_1588.tar.gz**

Setting Up the Design Examples

To set up the design examples, follow these steps:

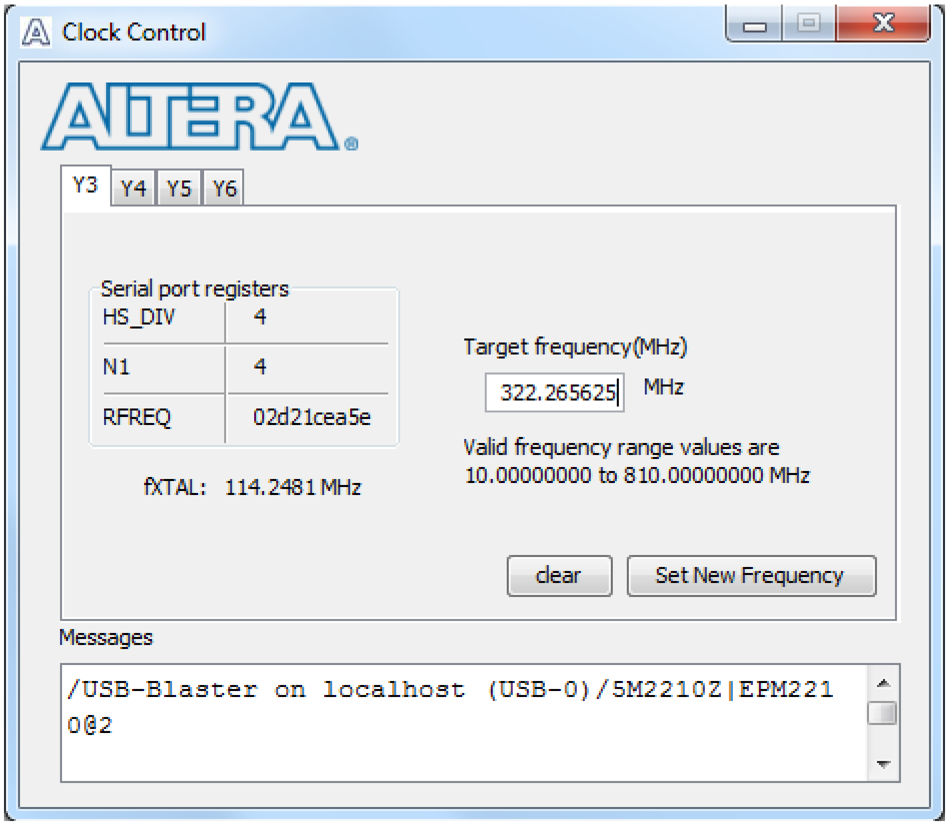
1. Unzip and untar the design examples at the project directory.

   Note: Altera recommends you to untar the example design in Linux environment. The VCS simulation script may become unrecognized or non-executable if the project archive was unzip in Windows environment.
2. Launch the Quartus II software and open the following project file:
   - **altera_eth_1g10g_top.qpf**—for Scalable Multispeed 10G Ethernet MAC using 1G/10G PHY without IEEE 1588v2.
   - **altera_eth_top.qpf**—for Scalable Multispeed 10G Ethernet MAC using 1G/10G PHY with IEEE 1588v2.
3. Click **Start Compilation** on the Processing menu to compile the design example.
4. Configure the FPGA on Stratix V GX Transceiver SI Development Board using either:
   - **altera_eth_1g10g_top.sof**—for Scalable Multispeed 10G Ethernet MAC using 1G/10G PHY without IEEE 1588v2.
   - **altera_eth_top.sof**—for Scalable Multispeed 10G Ethernet MAC using 1G/10G PHY with IEEE1588v2.
5. After configuration is done, open the **Clock Control** tool from **kits\stratixVGX_5sgxea7nf40_si\examples\board_test_system\ClockControl.exe**

   The **Clock Control** tool that is shipped with the "Installation Kit" for Stratix V GX Transceiver SI Development Board.
6. Set the new frequency for Y3 and Y4 as shown below:

   a. Y3: 322.265625 MHz
   b. Y4: 125.00MHz

Figure 3: Clock Control Settings for Y3



Serial port registers

HS_DIV	4
N1	4
RFREQ	02d21cea5e

fXTAL: 114.2481 MHz

Target frequency(MHz)

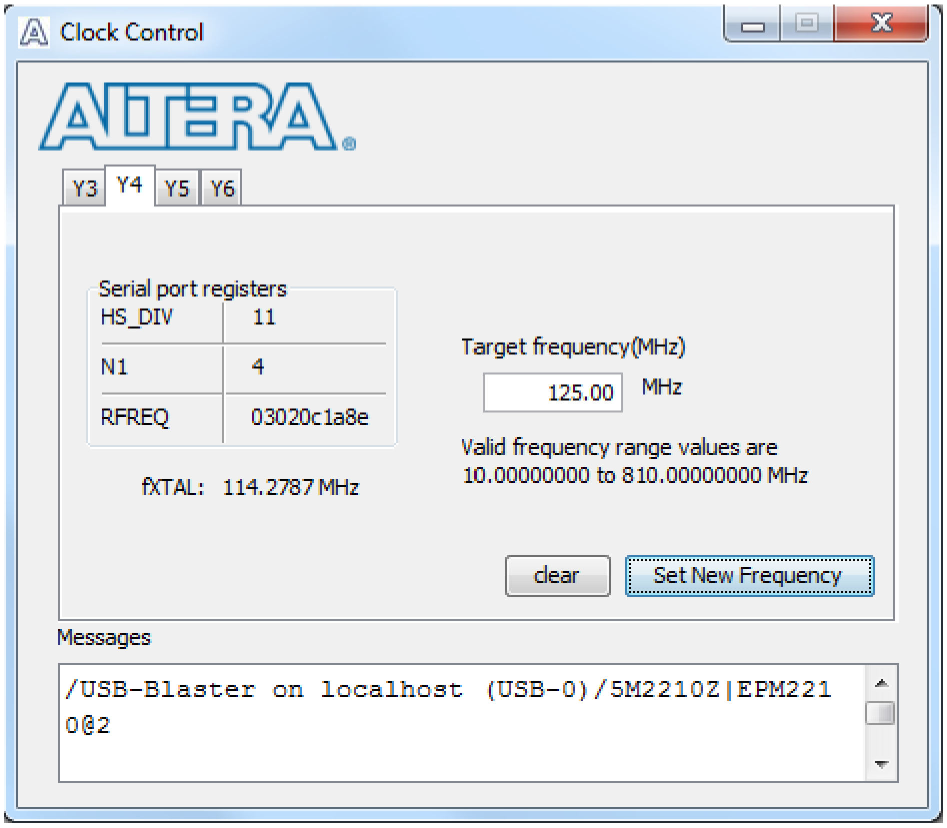
322.265625 MHz

Valid frequency range values are
10.00000000 to 810.00000000 MHz

Messages

```
/USB-Blaster on localhost (USB-0)/5M2210Z|EPM221
0@2
```

Figure 4: Clock Control Settings for Y4



Serial port registers

HS_DIV	11
N1	4
RFREQ	03020c1a8e

fXTAL: 114.2787 MHz

Target frequency(MHz)

125.00 MHz

Valid frequency range values are
10.00000000 to 810.00000000 MHz

Messages

```
/USB-Blaster on localhost (USB-0)/5M2210Z|EPM221
0@2
```

7. Press PB0 push button to reset the system.

   Note: System must be hard reset after configuration done.
8. On Quartus II Tools menu, click on **System Debugging Tools** and then launch **System Console**.
9. In the **System Console** command shell, change the directory to either:
   - `SystemConsole_wo1588`—for Scalable Multispeed 10G Ethernet MAC using 1G/10G PHY without IEEE 1588v2
   - `SystemConsole`—for Scalable Multispeed 10G Ethernet MAC using 1G/10G PHY with IEEE 1588v2
10. Run the command `source main.tcl` to initialize the reference design command list.
11. Perform the following test by running the command in the **System Console** command shell:
    a. PHY internal serial loopback

    Command: TEST_PHYSERIAL_LOOPBACK {channel speed_test burst_size}

    Example: `TEST_PHYSERIAL_LOOPBACK 0 1G 1000`

    b. SMA loopback

    Command:

- TEST_SMA_LOOPBACK {channel speed_test burst_size}—for Scalable Multispeed 10G Ethernet MAC using 1G/10G PHY without IEEE 1588v2

  Note: Channel 1 is assigned to SFP+ by default. To test Channel 1, you need to reassign it to SMA port.
- TEST_SMA_LB {channel speed_test burst_size}—for Scalable Multispeed 10G Ethernet MAC using 1G/10G PHY with IEEE 1588v2

Example: `TEST_SMA_LOOPBACK 0 1G 1000`

c. SMA loopback between 2 channels (for Scalable Multispeed 10G Ethernet MAC using 1G/10G PHY with IEEE 1588v2)

Command: TEST_1588 {from_channel to_channel speed_test}

Example: `TEST_1588 0 1 1G`

Note: For Scalable Multispeed 10G Ethernet MAC using 1G/10G PHY without IEEE 1588v2, Channel 0 is assigned to SMA (GXB_RXL_17 & GXB_TXL_17) and Channel 1 is assigned to SFP+ by default.

For Scalable Multispeed 10G Ethernet MAC using 1G/10G PHY with IEEE 1588v2, Channel 0 is assigned to SMA (GXB_RXL_17 & GXB_TXL_17) and Channel 1 is assigned to SMA (GXB_RXL_16 & GXB_TXL_16) by default.

Related Information

Transceiver Signal Integrity Development Kit, Stratix V GX Edition
Kit installation for Transceiver Signal Integrity Development Kit, Stratix V GX Edition

Changing Number of Channels

The design examples are configured to have two channels by default. To change the number of channels, modify the `NUM_CHANNELS` parameter of either:

- **altera_eth_top.sv** and **altera_eth_top.sdc** —for design example without IEEE 1588v2
- **altera_eth_top.sv** and **altera_eth_top_1588.sdc**—for design example with IEEE 1588v2

Configuring PHY Speed

After reset, all ports are in 10G and auto speed detection mode. Use the PHY memory map to change to other modes: 10G SFI, 1G 1000Base-X, or 1G/100M/10M SGMII.

Changing Speed between 10G and 1G in 1000BaseX mode

The software can turn off auto speed detection and force the PHY to either 1G or 10G by writing a different value to the PHY register address at offset 0x02C0.

Table 3: Register Value for Speed Change in 1000BaseX Mode

Value	Description
0x01	Reset back to auto speed detection mode
0x11	Turn off auto speed detection and force the PHY to 1G
0x41	Turn off auto speed detection and force the PHY to 10G

Forcing Port 0 to 1000Base-X mode

- Set Port 0 to 1000Base-X: `write_32 0x02_42C0 0x11`

Reset Port 0 to auto speed detection mode

- Set Port 0 to 1000Base-X: `write_32 0x02_42C0 0x01`

Changing Speed between 1G, 100M, and 10M SGMII

To enable SGMII, the software needs to write a different value to the PHY register address offset `0x0290`. Set the port to 1000Base-X mode first before you select any SGMII modes.

Table 4: Register Value for Speed Change in 1000BaseX Mode

Value	Description
0x01	Enable SGMII mode and force speed to 10M
0x03	Enable SGMII mode and use SGMII auto negotiation
0x05	Enable SGMII mode and force speed to 100M
0x09	Enable SGMII mode and force speed to 1G

Forcing Port 0 to SGMII 100M mode

1. Set Port 0 to 1000Base-X: `write_32 0x02_42C0 0x11`
2. Set Port 0 to SGMII 100M: `write_32 0x02_4290 0x05`

IP Regeneration

Regeneration of IP files is required when upgrading to a new version of ACDS. This process involves 2 different tools: Qsys and Megawizard. The following table shows the IPs that need regeneration and the tools involved.

Table 5: List of IPs Require Regeneration

IP	Tools	IP File Locations
address_decoder_channel	Qsys	ADDRESS_DECODER/address_decoder_channel.qsys
address_decoder_multi_ channel	Qsys	ADDRESS_DECODER/address_decoder_multi_ channel.qsys
address_decoder_top	Qsys	ADDRESS_DECODER/address_decoder_top.qsys
altera_eth_1588_tod(1)	Qsys	altera_eth_1588_tod/altera_eth_1588_tod.qsys
altera_eth_1588_tod_ synchronizer (1)	Qsys	altera_eth_1588_tod_synchronizer/ altera_eth_1588_tod_ synchronizer.qsys
altera_eth_packet_classifier(1)	Qsys	altera_eth_packet_classifier/ altera_eth_packet_ classifier.qsys

IP	Tools	IP File Locations
altera_eth_10g_mac	Megawizard	• Without IEEE 1588v2: MAC/altera_eth_10g_mac.v • With IEEE 1588v2: MAC_W_1588/altera_eth_10g_mac.v
altera_eth_10gkr_phy	Megawizard	• Without IEEE 1588v2: PHY/altera_eth_10gkr_phy.v • With IEEE 1588v2: PHY_W_1588/altera_eth_10gkr_phy.v
altera_xcvr_reset_controller	Megawizard	XCVR_RESET_CONTROLLER/altera_xcvr_reset_controller.v
reconfig	Megawizard	RECONFIG/reconfig.v
pll	Megawizard	PLL/pll.v
pll_2[1]	Megawizard	PLL/pll_2.v

Launch the tool and open the IP files listed in the table to regenerate the IP.

Design Example Testbench

Altera provides testbenches to verify the design examples, with or without IEEE 1588v2.

Testbench Components

The testbench operates in loopback mode. The following figure shows the flow of the packets in the design examples.

[1] This IP is only available for design example with IEEE 1588v2.

Figure 5: Testbench Block Diagram for Design Examples

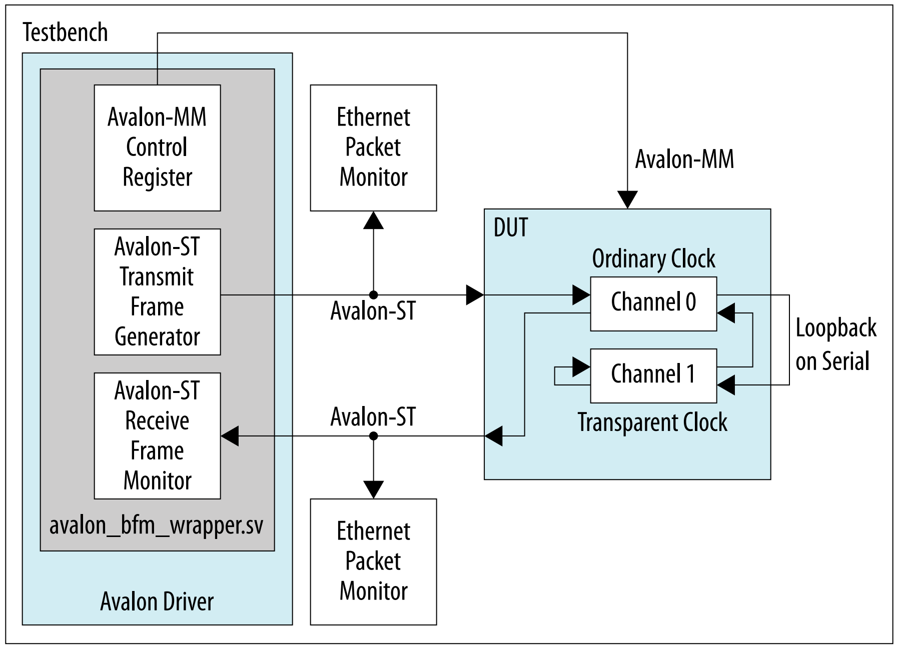


Table 6: List of Testbench Components and Description

Component	Description
Device under test (DUT)	The design example.
Avalon driver	Uses Avalon-ST master bus functional models (BFMs) to form transmit and receive paths. The driver also uses the master Avalon-MM BFM to access the Avalon-MM interfaces of the design example components.
Packet monitors	Monitor transmit and receive datapaths, and display the frames in the simulator console.

Testbench Files

The following table lists the location of the testbench files.

Table 7: Testbench Files Location

Design Examples	Location
Design example without IEEE 1588v2	*<project directory>* **/altera_eth_1g10g_lineside/testbench/<Modelsim or VCS>/testcase<n>**
Design example with IEEE 1588v2	*<project directory>***/altera_eth_1g10g_lineside_w_1588/testbench/<Modelsim or VCS>/testcase<n>**

Table 8: Testbench Files

File Name	Description
all_modes.mif	Memory initialization file (MIF) used for reconfiguration to change speed.
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 transmit and receive path, 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 transmit and receive 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.
• Without IEEE 1588v2: **tb_testcase.sv** • With IEEE 1588v2: **tb_testcase_1588.sv**	A SystemVerilog HDL testbench file that controls the flow of the testbench.
tb_top_n.sv • Without IEEE 1588v2: **tb_top_n.sv** • With IEEE 1588v2: **tb_top_n_1588.sv**	The top-level testbench file. This file includes the customized 1G/10GbE MAC, which 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 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.

Using ModelSim Simulator

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

1. Change the directory.

- For design example without IEEE 1588v2, change directory to **/altera_eth_1g10g_lineside/testbench/<Modelsim>/testcase<n>**
- For design example with IEEE 1588v2, change directory to **/altera_eth_1g10g_lineside_w_1588/testbench/<Modelsim>/testcase<n>**.

2. Launch Modelsim, and run **do tb_run.tcl** to set up the required libraries, to compile the generated IP functional simulation model, and to exercise the simulation model with the provided testbench.

Using VCS Simulator

Follow these steps if you choose to use the VCS simulator to simulate the testbench designs:

1. Change the directory.
   - For design example without IEEE 1588v2, change the directory to: **/altera_eth_1g10g_lineside/testbench/<VCS>/testcase<n>**
   - For design example with IEEE 1588v2, change the directory to: **/altera_eth_1g10g_lineside_w_1588/testbench/<VCS>/testcase<n>**
2. Run **./run.sh** to set up the required libraries, to compile the generated IP functional simulation model, and to exercise the simulation model with the provided testbench.

Test Case

The test cases are included to demonstrate how to change the channel speed to 10G/1G/100M/10M and MAC & PHY configuration.

Test Scenario for Design Example without IEEE 1588v2

This test case uses the following configuration:

- 2 channels
- Circular loopback

1. The channel is configured to 10G mode by default during start-up.
2. Perform basic MAC configuration, PHY speed configuration and FIFO configuration for all 2 channels.
3. Wait for the design example to assert the `channel_ready` signals for all 2 channels.
4. Send the following packets:
   - Normal data frame, 64Bytes
   - VLAN data frame, multicast, 1500Bytes
   - Normal data frame, 1500Bytes
   - SVLAN data frame, broadcast, 64Bytes
   - VLAN data frame, unicast, 500Bytes
   - SVLAN data frame, 1500Bytes
5. Repeat Step 2 to Step 4 for 1G, 100M and 10M speed mode.

   Note: The Avalon_st_rxstatus_valid and Avalon_st_txstatus_valid signals are not aligned to the Avalon_st_rx_endofpacket and Avalon_st_tx_endofpacket signals as stated in the 10Gbps Ethernet MAC MegaCore Function User Guide. This is due to the Avalon_st_rx_endofpacket and Avalon_st_tx_endofpacket signals are coming from an internal SC FIFO which creates a certain delay. You should observe the Avalon_st_rxstatus_valid and Avalon_st_txstatus_valid signals that correspond to the Avalon_st_rx_endofpacket and Avalon_st_tx_endofpacket signals directly from the MAC.

6. When the simulation ends, refer to the transcript window for channel 0 MAC TX and RX statistic counter results.

Figure 6: MAC TX and RX Statistic Counter Results

```
# ------------------------
# Channel 0: TX Statistics
# ------------------------
#     framesOK                         = 24
#     framesErr                        = 0
#     framesCRCErr                     = 0
#     octetsOK                         = 20080
#     pauseMACCtrlFrames               = 0
#     ifErrors                         = 0
#     unicastFramesOK                  = 16
#     unicastFramesErr                 = 0
#     multicastFramesOK                = 4
#     multicastFramesErr               = 0
#     broadcastFramesOK                = 4
#     broadcastFramesErr               = 0
#     etherStatsOctets                 = 20608
#     etherStatsPkts                   = 24
#     etherStatsUndersizePkts          = 0
#     etherStatsOversizePkts           = 0
#     etherStatsPkts64Octets           = 0
#     etherStatsPkts65to127Octets      = 8
#     etherStatsPkts128to255Octets     = 0
#     etherStatsPkts256to511Octet      = 4
#     etherStatsPkts512to1023Octets    = 0
#     etherStatsPkts1024to1518Octets   = 12
#     etherStatsPkts1519OtoXOctets     = 0
#     etherStatsFragments              = 0
#     etherStatsJabbers                = 0
#     etherStatsCRCErr                 = 0
#     unicastMACCtrlFrames             = 0
#     multicastMACCtrlFrames           = 0
#     broadcastMACCtrlFrames           = 0
#
# ------------------------
# Channel 0: RX Statistics
# ------------------------
#     framesOK                         = 24
#     framesErr                        = 0
#     framesCRCErr                     = 0
#     octetsOK                         = 20080
#     pauseMACCtrlFrames               = 0
#     ifErrors                         = 0
#     unicastFramesOK                  = 16
#     unicastFramesErr                 = 0
#     multicastFramesOK                = 4
#     multicastFramesErr               = 0
#     broadcastFramesOK                = 4
#     broadcastFramesErr               = 0
#     etherStatsOctets                 = 20608
#     etherStatsPkts                   = 24
#     etherStatsUndersizePkts          = 0
#     etherStatsOversizePkts           = 0
#     etherStatsPkts64Octets           = 0
#     etherStatsPkts65to127Octets      = 8
#     etherStatsPkts128to255Octets     = 0
#     etherStatsPkts256to511Octet      = 4
#     etherStatsPkts512to1023Octets    = 0
#     etherStatsPkts1024to1518Octets   = 12
#     etherStatsPkts1519OtoXOctets      = 0
#     etherStatsFragments               = 0
#     etherStatsJabbers                 = 0
#     etherStatsCRCErr                  = 0
#     unicastMACCtrlFrames              = 0
#     multicastMACCtrlFrames            = 0
#     broadcastMACCtrlFrames            = 0
#
#
# Simulation PASSED
```

If all the total 24 packets have been received successfully to channel 0 Avalon_st RX interface, the transcript will display `Simulation PASSED`.

Test Scenario for Design Example with IEEE 1588v2

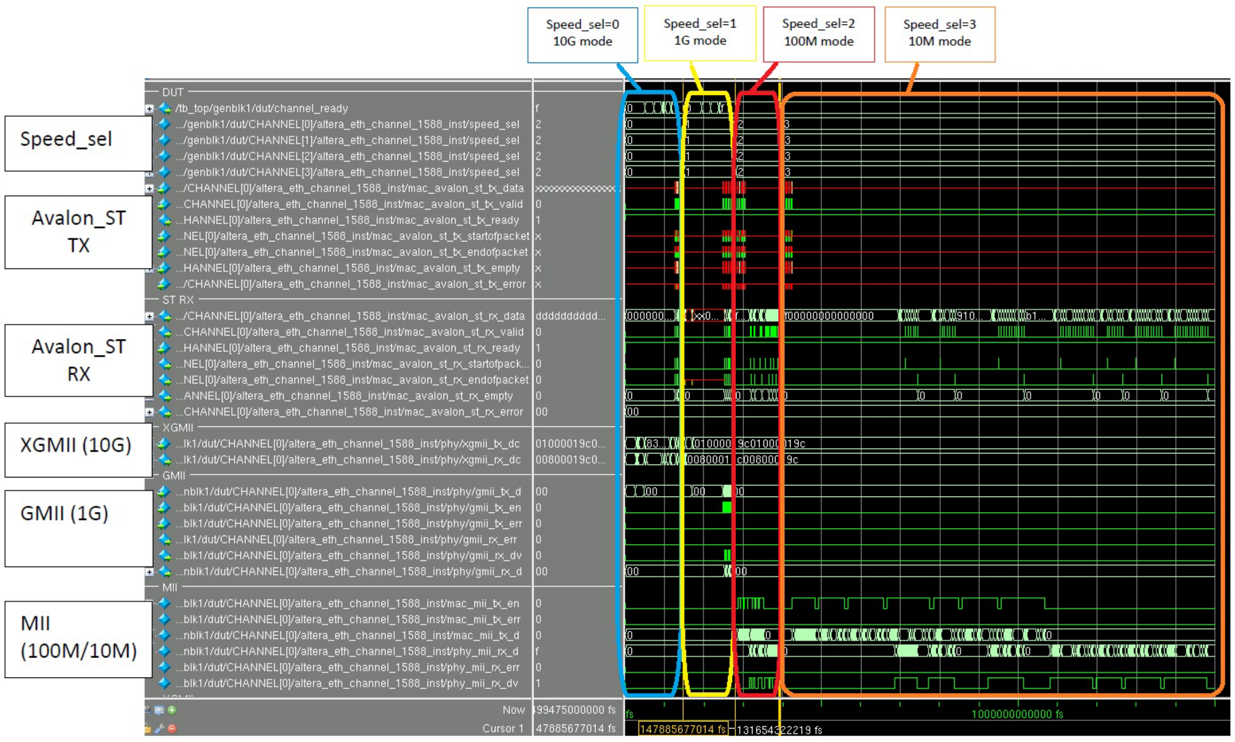
There are three test cases available to showcase how to change channel speed to 10G/1G/100M/10M.

Testcase 1

This testcase simulates manual speed change between 10G/1G/100M/10M in SGMII/1000Base-X mode with auto negotiation disabled using circular loopback configuration.

- Parameter in default_test_parameter.sv file: SGMII_1000BASEX (1-SGMII, 0-1000BaseX) to select between SGMII and 1000BaseX mode.
- To change number of channels, modify the parameter NUM_CHANNELS in default_test_params_pkg.sv file

Figure 7: Channel Waveform for Speed_sel, Avalon_st, XGMII ,GMII and MII Interface



1. Force all of the channel to 10G mode by setting speed_sel =0. You can observe the data through the XGMII interface .
2. After all 7 data packets sent and received by avalon_st RX of the channel, the design issues write operation to the csr interface to change to another speed mode.
3. Continue with 1G mode by setting the speed_sel=1 and the data will be observed in GMII interface.
4. For 100M, set speed_sel=2 and 10M, speed_sel=3. These mode transmissions of data happens in MII interfaces.
5. For each speed mode, the test case runs in circular loopback sequence. The data flows starting with channel0 followed by channel1, channel 2, channel 3, then loopback to channel 2, channel 1 and channel 0.
6. After the simulation stop, user can refer to the transcript window for channel0 MAC TX and RX Statistic counter result.

Figure 8: MAC TX and RX Statistic Counter Results

```
# ------------------------
# Channel         0: TX Statistics
# ------------------------
#     framesOK                     = 28
#     framesErr                    = 0
#     framesCRCErr                 = 0
#     octetsOK                     = 2088
#     pauseMACCtrlFrames           = 0
#     ifErrors                     = 0
#     unicastFramesOK              = 28
#     unicastFramesErr             = 0
#     multicastFramesOK            = 0
#     multicastFramesErr               = 0
#     broadcastFramesOK                = 0
#     broadcastFramesErr               = 0
#     etherStatsOctets                 = 2688
#     etherStatsPkts                   = 28
#     etherStatsUndersizePkts          = 0
#     etherStatsOversizePkts           = 0
#     etherStatsPkts64Octets           = 4
#     etherStatsPkts65to127Octets      = 16
#     etherStatsPkts128to255Octets     = 8
#     etherStatsPkts256to511Octet      = 0
#     etherStatsPkts512to1023Octets    = 0
#     etherStatsPkts1024to1518Octets   = 0
#     etherStatsPkts1519OtoXOctets     = 0
#     etherStatsFragments              = 0
#     etherStatsJabbers                = 0
#     etherStatsCRCErr                 = 0
#     unicastMACCtrlFrames             = 0
#     multicastMACCtrlFrames           = 0
#     broadcastMACCtrlFrames           = 0
#
# ------------------------
# Channel         0: RX Statistics
# ------------------------
#     framesOK                         = 28
#     framesErr                        = 0
#     framesCRCErr                     = 0
#     octetsOK                         = 2088
#     pauseMACCtrlFrames               = 0
#     ifErrors                         = 0
#     unicastFramesOK                  = 28
#     unicastFramesErr                 = 0
#     multicastFramesOK                = 0
#     multicastFramesErr               = 0
#     broadcastFramesOK                = 0
#     broadcastFramesErr               = 0
#     etherStatsOctets                 = 2688
#     etherStatsPkts                   = 28
#     etherStatsUndersizePkts          = 0
#     etherStatsOversizePkts           = 0
#     etherStatsPkts64Octets           = 4
#     etherStatsPkts65to127Octets      = 16
#     etherStatsPkts128to255Octets     = 8
#     etherStatsPkts256to511Octet      = 0
#     etherStatsPkts512to1023Octets    = 0
#     etherStatsPkts1024to1518Octets   = 0
#     etherStatsPkts1519OtoXOctets     = 0
#     etherStatsFragments              = 0
#     etherStatsJabbers                = 0
#     etherStatsCRCErr                 = 0
#     unicastMACCtrlFrames             = 0
#     multicastMACCtrlFrames           = 0
#     broadcastMACCtrlFrames           = 0
#
#
# Simulation PASSED
```

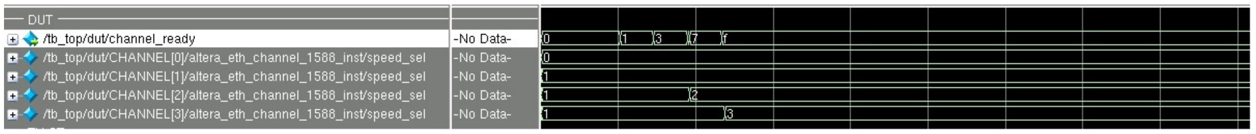
If all the total 28 packets have been received successfully to channel 0 Avalon_st RX interface, the transcript will display `Simulation PASSED`.

Testcase 2

This testcase simulates manual speed change between 10G/1G/100M/10M in SGMII/1000Base-X mode with auto negotiation disabled using serial loopback configuration.

- To choose speed for each channel, go to tb_testcase_1588, change the value of reg [1:0] speed[NUM_CHANNELS] = {}
- To change number of channels: modify the parameter NUM_CHANNELS in default_test_parameter.sv file
- To choose SGMII or 1000 Base-X mode for 1G, go to default_test_parameter.sv file, change parameter SGMII_1000BASEX (1-SGMII, 0-1000 Base-X)
- PHY serial loopback for each channel setup:
  - channel0: 10G
  - channel1: 1G
  - channel2: 100M
  - channel3: 10M

Figure 9: Waveform for Channel_ready and speed_sel for All 4 Channels



```
DUT
/tb_top/dut/channel_ready                                   -No Data-   0  1  3  7  f
/tb_top/dut/CHANNEL[0]/altera_eth_channel_1588_inst/speed_sel  -No Data-   0
/tb_top/dut/CHANNEL[1]/altera_eth_channel_1588_inst/speed_sel  -No Data-   1
/tb_top/dut/CHANNEL[2]/altera_eth_channel_1588_inst/speed_sel  -No Data-   1  2
/tb_top/dut/CHANNEL[3]/altera_eth_channel_1588_inst/speed_sel  -No Data-   1  3
```

1. Force the channel0 to 10G mode and the data transmission can be observed in XGMII interface.
2. Force the channel1 to 1G mode and the data transmission can be observed in GMII interface.
3. To set channel2 to 100M mode, first set the channel to 1G mode. Next, enable the SGMII mode and set it to 100M mode. Data transmission can be observed in MII interface signals.
4. Repeat Step 3 for channel3, only this time set the speed to 10M after enabling SGMII mode. Data transmission can be observed in MII interface signals.
5. The data transmission for each channel will only start when the channel_ready signal is up. MAC TX and RX statistic will be printed out in transcript window for each channel.

Testcase 3

This testcase simulates 1G in 1000Base-X mode with auto negotiation enabled in serial loopback configuration.

- NUM_CHANNELS = 1
- Link timer = 8 micro second

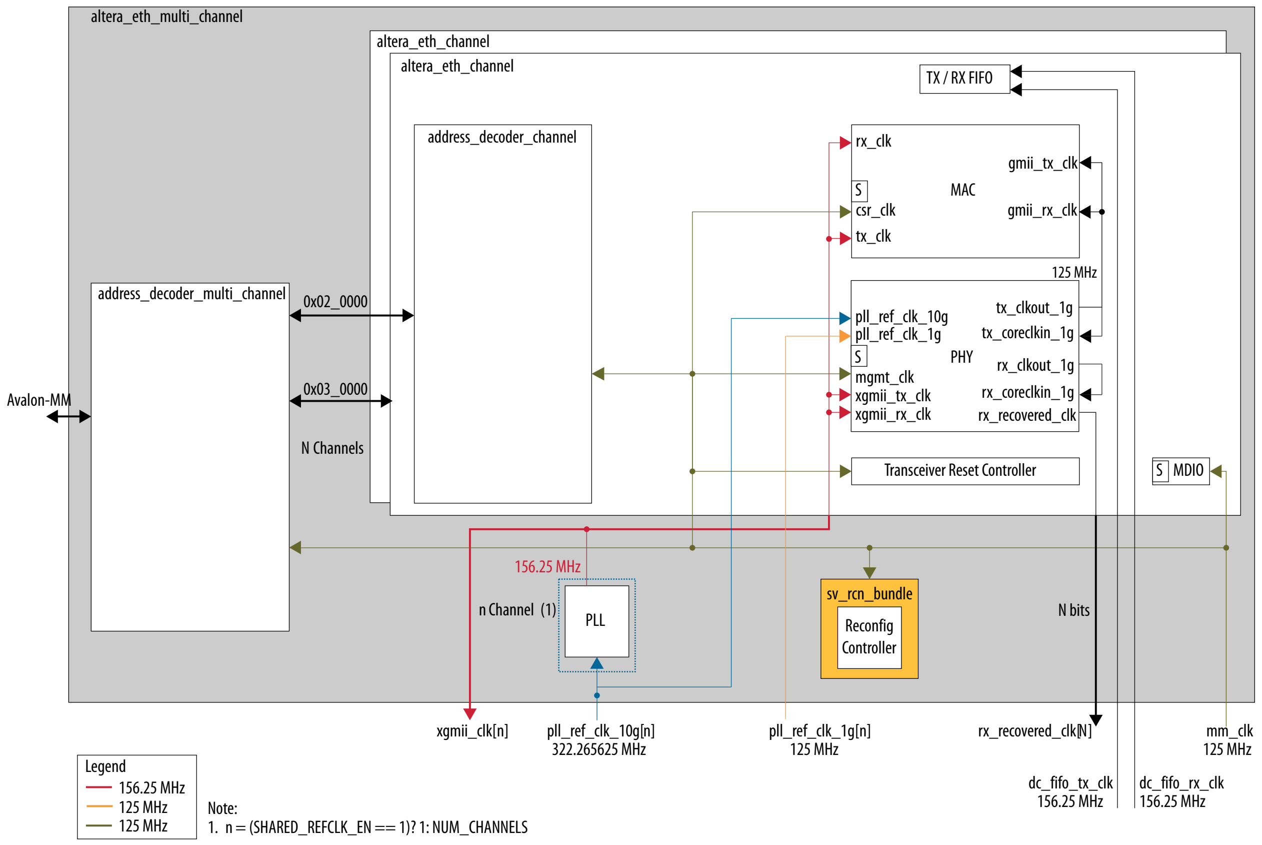
Clocking Scheme

There are *n* instances of PLL 1 and it is merged into 1 if `SHARED_REFCLK_EN` = 1.

Clocking Diagram

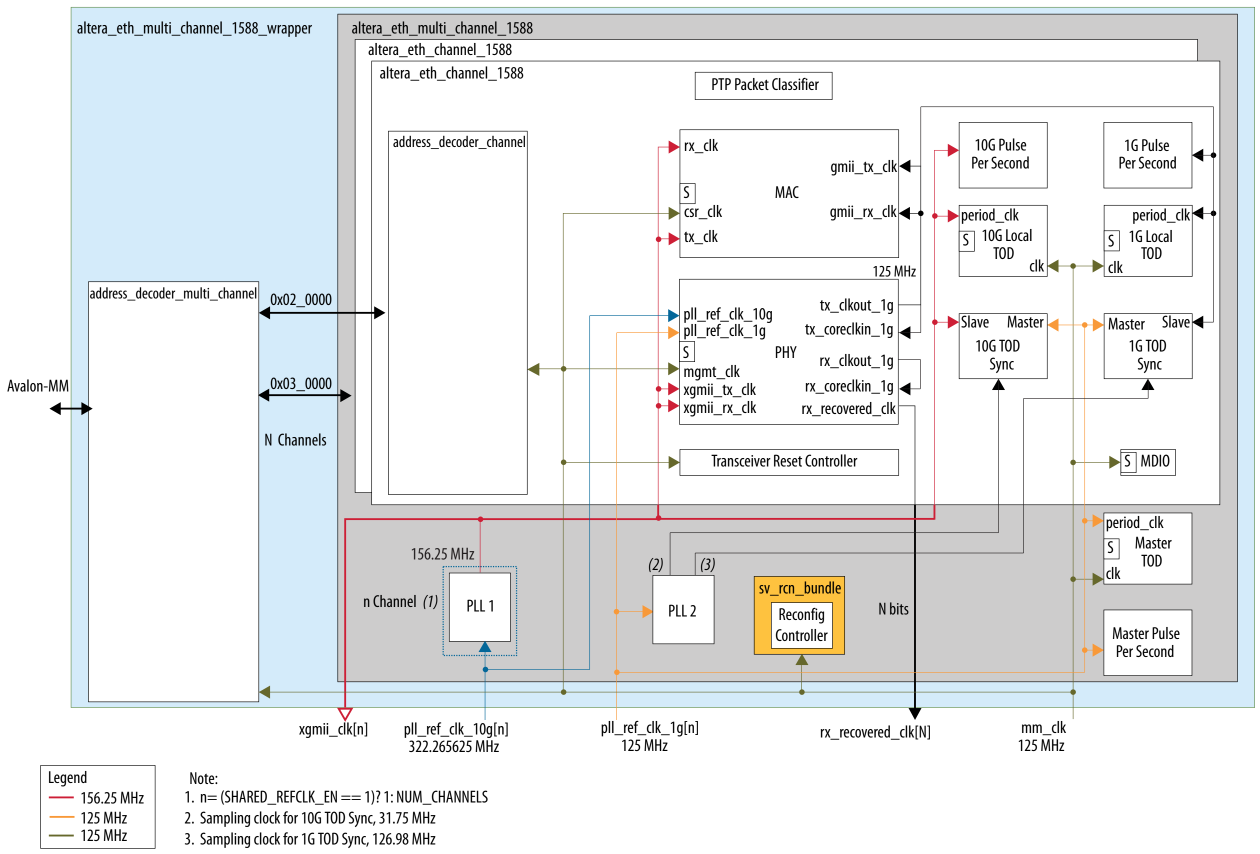
The following diagrams show the clocking scheme for the design example without IEEE 1588v2 and design example with IEEE 1588v2 respectively.

Figure 10: Clocking Scheme for the Design Example without IEEE 1588v2



Note:
1. n = (SHARED_REFCLK_EN == 1)? 1: NUM_CHANNELS

Figure 11: Clocking Scheme for the Design Example with IEEE 1588v2



Note:
1. n= (SHARED_REFCLK_EN == 1)? 1: NUM_CHANNELS
2. Sampling clock for 10G TOD Sync, 31.75 MHz
3. Sampling clock for 1G TOD Sync, 126.98 MHz

Sync-E Support

To support Sync-E implementation, separate `refclk` signals to RX PLL and TX PLL and expose them at design example. The following diagrams show the signals per channel for design example without IEEE 1588v2 and design example with IEEE 1588v2 respectively.

Figure 12: Signals from PHY to Support Sync-E Implementation for Design Example without IEEE 1588v2

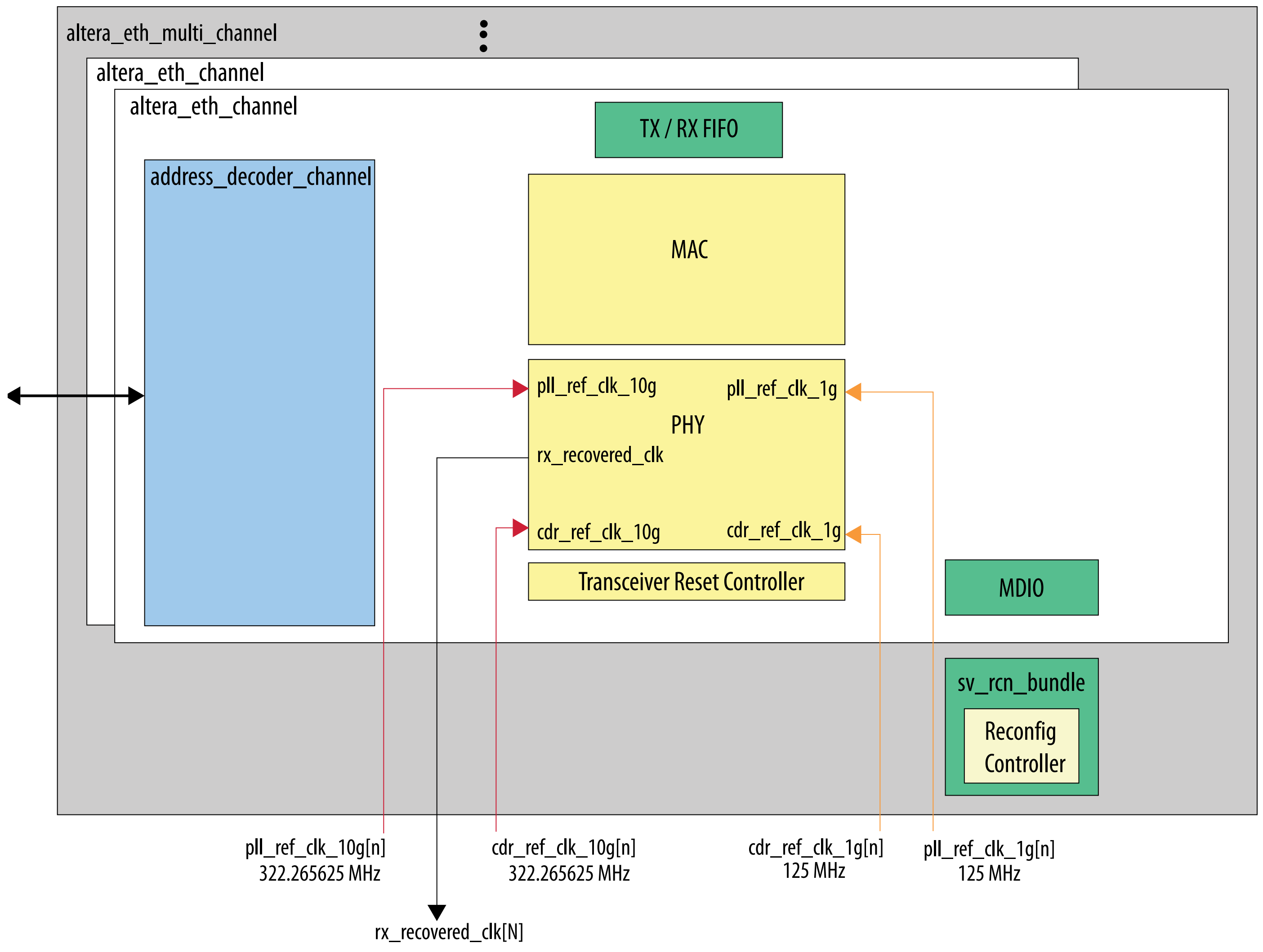
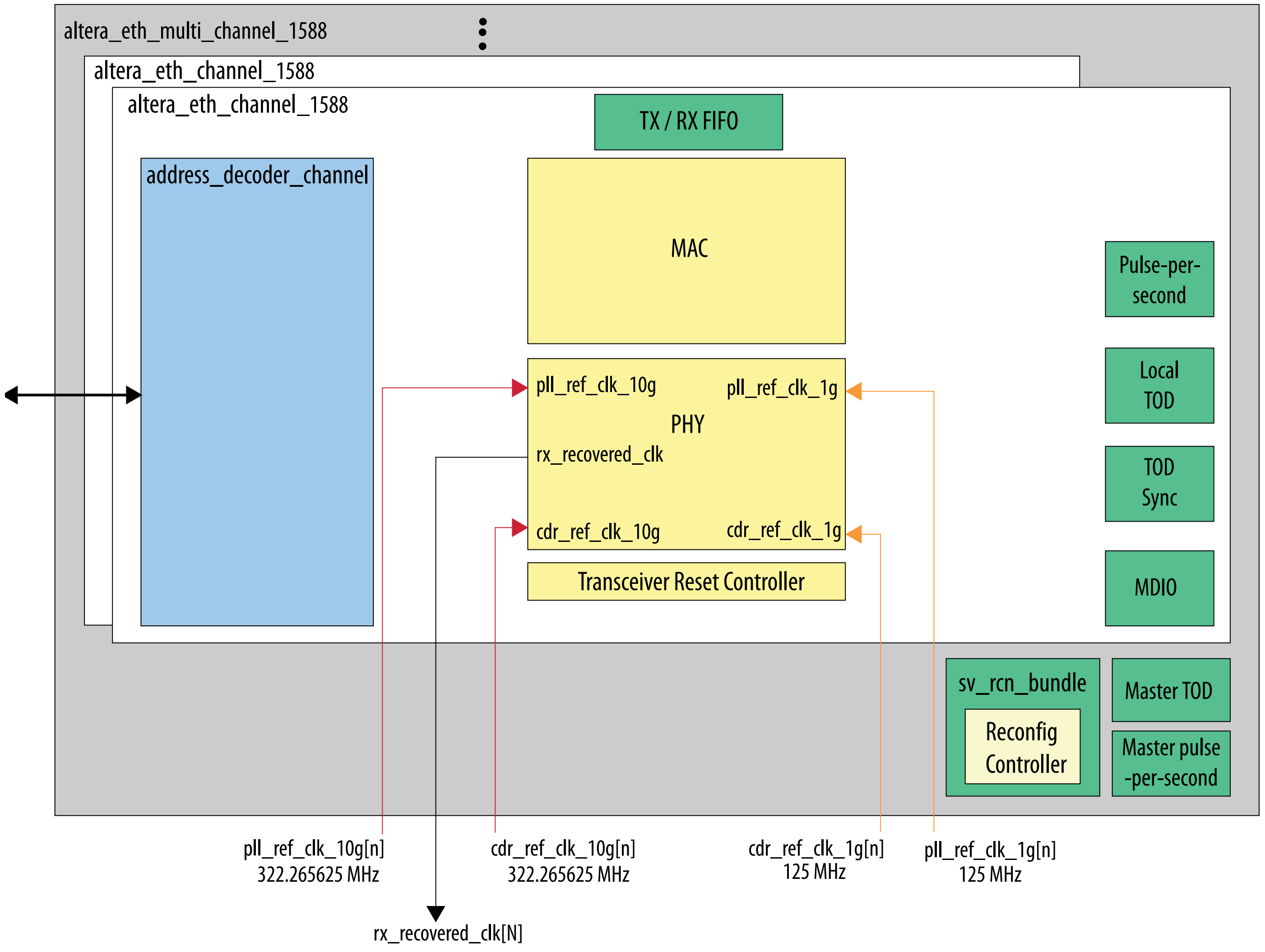


Figure 13: Signals from PHY to Support Sync-E Implementation for Design Example with IEEE 1588v2



Enable Ref Clock Sharing

When you set the parameter `SHARED_REFCLK_EN` to `1`, the design example will enable the ref clock sharing and only 1 set of `pll_ref_clk_10g`, `pll_ref_clk_1g`, `cdr_ref_clk_10g` and `cdr_ref_clk_1g` is needed. These ref clock signals will be used across all channels. There will be N number of `rx_recovered_clk` regardless of ref clock sharing setting, where N=number of channels.

Disable Ref Clock Sharing

When you set the parameter `SHARED_REFCLK_EN` to `0`, this will disable the ref clock sharing and N set of `pll_ref_clk_10g`, `pll_ref_clk_1g`, `cdr_ref_clk_10g` and `cdr_ref_clk_1g` are needed, where N=number of channels. These ref clock signals will be connected to their individual channel respectively.

Reset Scheme

At the design example level, there are one `master_reset_n` and <*N*> `channel_reset_n` signals. All the signals are asynchronous and active-low signal. The signals are synced to different clock domain internally. When the `master_reset_n` is de-asserted, the signal will bring down all <*N*> Ethernet channels and all modules in the design example.

The `channel_reset_n[0..N]` only reset all the components in the individual channel.

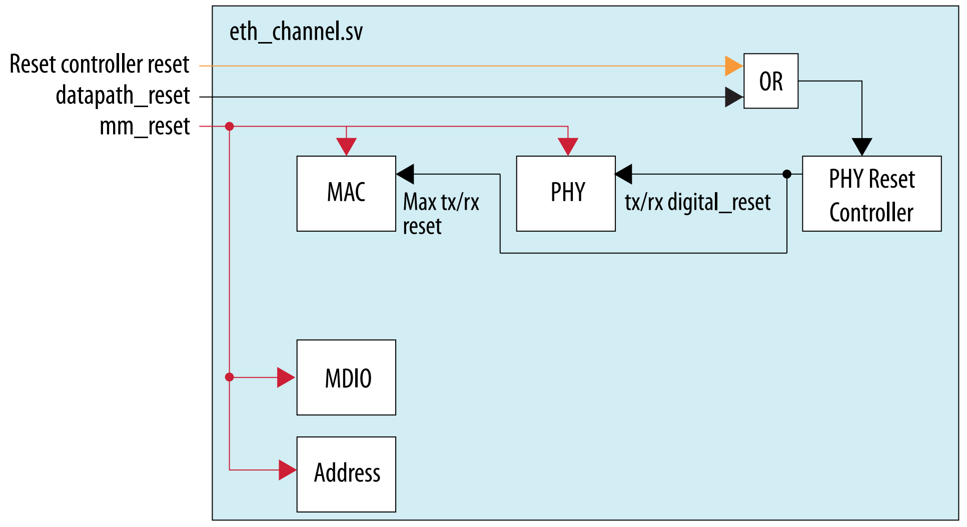
Master reset is needed when the design example is powered up.

Design Example without IEEE 1588v2

Channel Level Reset Scheme

The following diagram shows the reset scheme per channel. `mm_reset` is used to reset the registers in MAC, PHY, MDIO and address_decoder block while `datapath_reset` is used to reset all digital blocks including PHY reset controller. However, `mm_reset` and `datapath_reset` are tied together at multi channel level in the design example, therefore they can't be triggered separately.

Figure 14: Reset scheme at altera_eth_channel

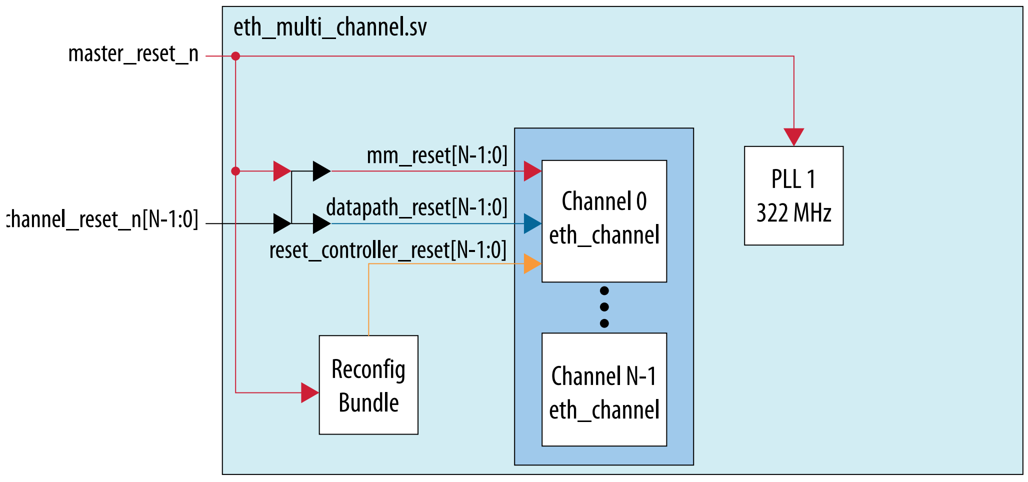


Notes:
1. Reset_controller_reset starts the PHY reset controller.
2. The mm_reset signal resets all Avalon-MM control blocks.
3. The datapath_reset signal resets all digital blocks, including the PHY reset controller.

Multi-Channel Level Reset Scheme

The following diagram shows the reset scheme at **altera_eth_multi_channel** level. `master_reset_n` is used to reset the whole design example, while `channel_reset_n` is used to reset the individual Ethernet channel. When you change the speed of the ethernet channel, the reconfig bundle will trigger the `reset_controller_reset` for individual channel after reconfiguration is done.

Figure 15: Reset scheme at altera_eth_multi_channel



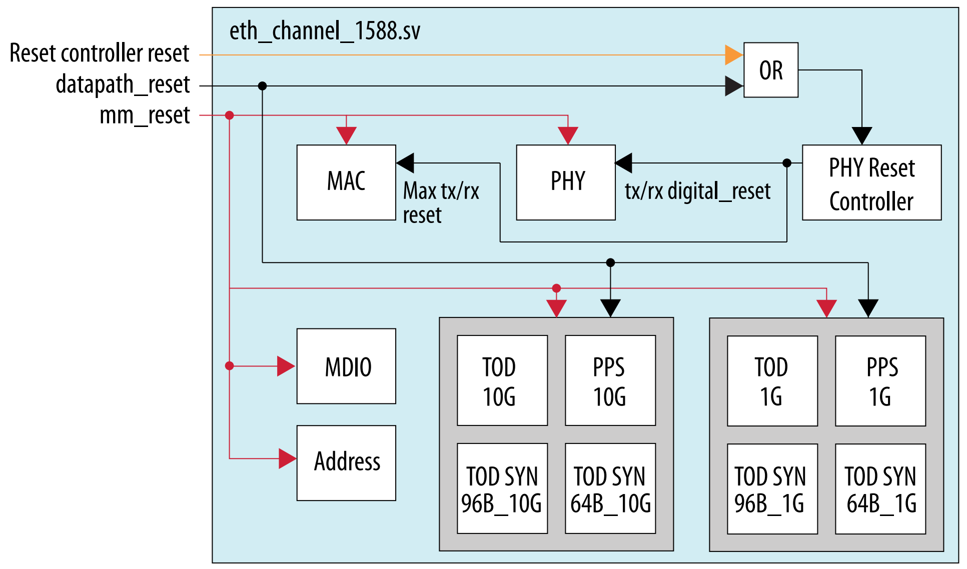
Note:
1. N is the number of channels.

Design Example with IEEE 1588v2

Channel Level Reset Scheme

The following diagram shows the reset scheme per channel. `mm_reset` is used to reset the registers in MAC, PHY, TOD, MDIO and address decoder block, while `datapath_reset` is used to reset MAC, PHY reset controller and TOD. However, `mm_reset` and `datapath_reset` are tied together at multi-channel level in the design example, therefore it is not possible to trigger them separately.

Figure 16: Reset scheme at altera_eth_channel_1588

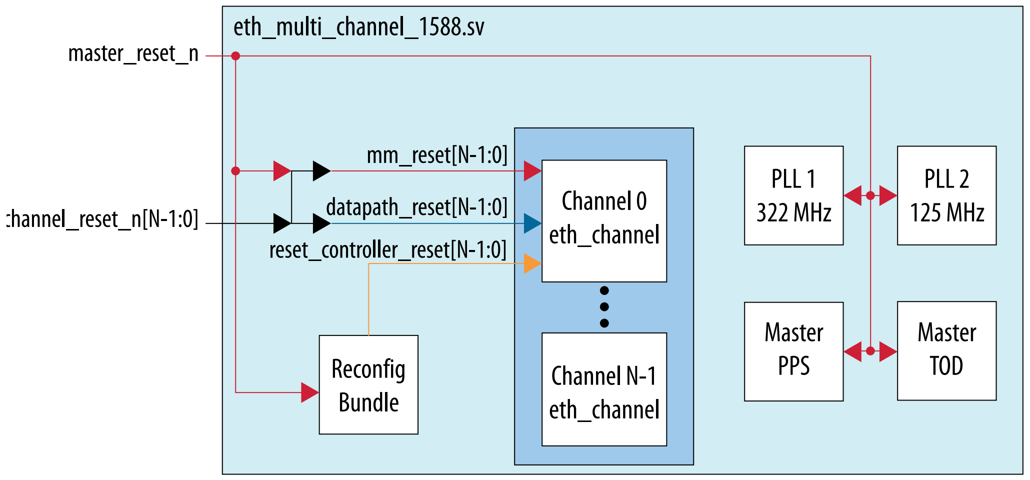


Notes:
1. Reset_controller_reset starts the PHY reset controller.
2. The mm_reset signal resets all Avalon-MM control blocks.
3. The datapath_reset signal resets all digital blocks, including the PHY reset controller.

Multi-Channel Level Reset Scheme

The following diagram shows the reset scheme at **altera_eth_multi_channel_1588** level. `master_reset_n` is used to reset the whole design example, while `channel_reset_n` is used to reset the individual Ethernet channel. When you change the speed of the ethernet channel, the reconfig bundle will trigger the `reset_controller_reset` for individual channel after reconfiguration is done.

Figure 17: Reset scheme at altera_eth_multi_channel_1588



Note:
1. N is the number of channels.

Internal PHY PLL Powerdown Connection Scheme

All transceiver reset input signals are connected to Transceiver(XCVR) Reset Controller in individual channel except `pll_powerdown`. `pll_powerdown` port of every transceiver channels are connected to `master_reset_n`. This setup enables all channels of the transceiver to be placed in contiguous banks and fewer ATX PLLs and CMU PLLs is used as they are being merged after fitter process.

Interface Signals

This section describes the interface signals at design example level. All design example has the same interface signals except the Avalon-MM interface for design example with IEEE 1588v2 with wrapper file.

The `NUM_UNSHARED_CHANNELS` are determined by the equation below:

`NUM_UNSHARED_CHANNELS` = (`SHARED_REFCLK_EN` == 1) ? 1: `NUM_CHANNELS`

`NUM_CHANNELS` and `SHARED_REFCLK_EN` are parameters set by user.

Clock and Reset Interface Signals

Table 9: Clock and Reset Interface Signals

Signal	Direction	Width	Description
`mm_clk`	input	1	Configuration clock for Avalon-MM interface. Frequency is 125 MHz. The clock runs at 100MHz to 125MHz for Stratix V.
`pll_ref_clk_1g[]`	input	`[NUM_UNSHARED_CHANNELS]`	Reference clock for the TX PLL in 1G mode. Frequency is 125 MHz.
`pll_ref_clk_10g[]`	input	`[NUM_UNSHARED_CHANNELS]`	Reference clock for the TX PLL in 10G mode. Frequency is 322.265625 MHz.
`cdr_ref_clk_1g[]`	input	`[NUM_UNSHARED_CHANNELS]`	Reference clock for the RX PLL in 1G mode. Frequency is 125 MHz.
`cdr_ref_clk_10g[]`	input	`[NUM_UNSHARED_CHANNELS]`	Reference clock for the RX PLL in 10G mode. Frequency is 322.265625 MHz.

Signal	Direction	Width	Description
`channel_reset_n`	input	`[NUM_CHANNELS]`	To reset individual Ethernet channel. This does not impact the components running at multi_channel level, e.g. master TOD, master PPS, reconfig bundle, and fPLLs. Asynchronous and active low signal.
`master_reset_n`	input	1	To reset the whole design example. Asynchronous and active low signal.
`xgmii_clk`	output	`[NUM_UNSHARED_CHANNELS]`	Clock used for single data rate (SDR) XGMII TX & RX interface in between MAC and PHY. This clock is also used for Avalon-ST interface. Frequency is 156.25MHz.
`rx_recovered_clk`	output	`[NUM_CHANNELS]`	This is the RX clock, which is recovered from the received data.

Avalon-MM Interface Signals

Table 10: Avalon-MM Interface Signals for Design Example with and without IEEE 1588v2.

This table is applicable to:

- Design Example without IEEE 1588v2—**altera_eth_multi_channel**
- Design Example with IEEE 1588v2—**altera_eth_multi_channel_1588**

Signal	Direction	Width	Description
`write`	input	1	Assert this signal to request a write.
`read`	input	1	Assert this signal to request a read.
`address[]`	input	20	Use this bus to specify the register address you want to read from or write to.
`writedata[]`	input	32	Carries the data to be written to the specified register.
`readdata[]`	output	32	Carries the data read from the specified register.

Signal	Direction	Width	Description
`waitrequest`	output	1	When asserted, this signal indicates that the IP core is busy and not ready to accept any read or write requests.

Table 11: Avalon-MM Interface Signals for Design Example IEEE 1588v2 with wrapper.

This table is applicable to:

- Design Example with IEEE 1588v2 with wrapper—**altera_eth_multi_channel_1588_wrapper**

Signal	Direction	Width	Description
`multi_channel_write[]`	input	`[NUM_CHANNELS]`	Assert this signal to request a write toEthernet channel <n>.
`multi_channel_read[]`	input	`[NUM_CHANNELS]`	Assert this signal to request a read toEthernet channel <n>.
`multi_channel_address[][]`	input	`[NUM_CHANNELS][16]`	Use this bus to specify the register addressyou ant to read from or write to Ethernet channel <n>.
`multi_channel_writedata[][]`	input	`[NUM_CHANNELS][32]`	Carries the data to be written to thespecified register of Ethernet channel <n>.
`multi_channel_readdata[][]`	output	`[NUM_CHANNELS][32]`	Carries the data read from the specified register of Ethernet channel <n>.
`multi_channel_waitrequest[]`	output	`[NUM_CHANNELS]`	When asserted, this signal indicates that the IP core of Ethernet channel <n> is busy and not ready to accept any read or write requests.
`master_tod_write`	input	1	Assert this signal to request a write toMaster TOD.
`master_tod_read`	input	1	Assert this signal to request a read toMaster TOD.
`master_tod_address[]`	input	6	Use this bus to specify the register addressyou want to read from or write to Master TOD
`master_tod_writedata[]`	input	32	Carries the data to be written to the specified register of Master TOD.

Signal	Direction	Width	Description
`master_tod_readdata[]`	output	32	Carries the data read from the specified register of Master TOD.
`master_tod_waitrequest`	output	1	When asserted, this signal indicates that the IP core of Master TOD is busy and not ready to accept any read or write requests.

Avalon-ST Interface Signals

Table 12: Avalon-ST Interface Signals

Signal	Direction	Width	Description
`avalon_st_tx_startof-packet[]`	input	`[NUM_CHANNELS]`	Assert this signal to mark the beginning of the transmit data on the Avalon-ST interface.
`avalon_st_tx_endofpacket[]`	input	`[NUM_CHANNELS]`	Assert this signal to mark the end of the transmit data on the Avalon-ST interface.
`avalon_st_tx_valid[]`	input	`[NUM_CHANNELS]`	Assert this signal to indicate that `avalon_st_tx_data[]` and other signals on this interface are valid.
`avalon_st_tx_ready[]`	output	`[NUM_CHANNELS]`	When asserted, this signal indicates that the MAC IP core is ready to accept data.
`avalon_st_tx_error[][]`	input	`[NUM_CHANNELS][64]`	Assert this signal to indicate the current transmit packet contains errors.
`avalon_st_tx_data[][]`	input	`[NUM_CHANNELS][3]`	Carries the transmit data from the client.

Signal	Direction	Width	Description
`avalon_st_tx_empty[]`	input	`[NUM_CHANNELS]`	Use this signal to specify the number of bytes that are empty (not used) during cycles that contain the end of a packet. 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_startof-packet[]`	output	`[NUM_CHANNELS]`	When asserted, this signal marks the beginning of the receive data on the Avalon-ST interface.
`avalon_st_rx_endofpacket[]`	output	`[NUM_CHANNELS]`	When asserted, this signal marks the end of the receive data on the Avalon-ST interface.
`avalon_st_rx_valid[]`	output	`[NUM_CHANNELS]`	When asserted, this signal indicates that `avalon_st_rx_data[]`and other signals on this interface are valid.
`avalon_st_rx_ready[]`	input	`[NUM_CHANNELS]`	Assert this signal when the client is ready to accept data.

Signal	Direction	Width	Description
`avalon_st_rx_error[][]`	output	`[NUM_CHANNELS][64]`	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[][]`	output	`[NUM_CHANNELS][3]`	Carries the receive data to the client.
`avalon_st_rx_empty[][]`	output	`[NUM_CHANNELS][6]`	Contains the number of bytes that are empty (not used) during cycles that contain the end of a packet.
`avalon_st_tx_status_valid[]`	output	`[NUM_CHANNELS]`	When asserted, this signal qualifies `avalon_st_txstatus_data[]` and `avalon_st_txstatus_error[]`.

Signal	Direction	Width	Description
`avalon_st_tx_status_data[][]`	output	`[NUM_CHANNELS][40]`	Contains information about the transmit 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[][]`	output	`[NUM_CHANNELS][7]`	When set to 1, the respective bit indicates the following error type in the receive 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_rxstatus_valid[]`	output	`[NUM_CHANNELS]`	When asserted, this signal qualifies `avalon_st_txstatus_data[]` and `avalon_st_txstatus_error[]`. The MAC IP core asserts this signal in the same clock cycle `avalon_st_rx_endofpacket` is asserted.

Signal	Direction	Width	Description
`avalon_st_rxstatus_data[][]`	output	`[NUM_CHANNELS][40]`	Contains information about the transmit 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_rxstatus_error[][]`	output	`[NUM_CHANNELS][7]`	When set to 1, the respective bit indicates the following error type in the receive 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.

Signal	Direction	Width	Description
`avalon_st_pause_data[][]`	input	`[NUM_CHANNELS][2]`	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. **Note:** This signal only takes effect if `tx_pauseframe_enable[2:1]` is 00 (default)

PHY Interface Signals

Table 13: PHY Interface Signals

Signal	Direction	Width	Description
`rx_serial_data[]`	input	`[NUM_CHANNELS]`	RX serial input data
`tx_serial_data[]`	output	`[NUM_CHANNELS]`	TX serial output data
`ethernet_1g_an[]`	output	`[NUM_CHANNELS]`	Clause 37 Auto-Negotiation status. The PCS function asserts this signal when auto-negotiation completes.
`ethernet_1g_char_err[]`	output	`[NUM_CHANNELS]`	10-bit character error
`ethernet_1g_disp_err[]`	output	`[NUM_CHANNELS]`	Disparity error signal indicating a 10-bit running disparity error.
`channel_ready[]`	output	`[NUM_CHANNELS]`	Asserted when the channel is ready for data transmission.

Signal	Direction	Width	Description
`ethernet_link_ready[]`	output	`[NUM_CHANNELS]`	Asserted after PHY reconfiguration is done for the ethernet channel and ready for register configuration.
`ethernet_speed_sel[][]`	output	`[NUM_CHANNELS][1:0]`	Indicates the link fault status from the RS RX to the RS TX. • 2'b00: 10G • 2'b01: 1G • 2'b10: 100M • 2'b11: 10M
`ethernet_10g_link_fault_status[][]`	output	`[NUM_CHANNELS][1:0]`	Indicates the link fault status from the RS RX to the RS TX. • 00: No link fault • 01: Local fault • 10: Remote fault

MDIO Interface Signals

Table 14: MDIO Interface Signals

Signal	Direction	Width	Description
`mdio_mdc[]`	output	`[NUM_CHANNELS]`	Management Data clock
`mdio_in[]`	input	`[NUM_CHANNELS]`	Input to MDIO interface
`mdio_out[]`	output	`[NUM_CHANNELS]`	Output from MDIO interface
`mdio_oen[]`	output	`[NUM_CHANNELS]`	Output enable signal

IEEE 1588v2 Timestamp Interface Signals

Table 15: IEEE 1588v2 Timestamp Interface Signals

Signal	Direction	Width	Description
`tx_egress_timestamp_96b_valid[]`	output	`[NUM_CHANNELS]`	When asserted, this signal qualifies the timestamp on `tx_egress_timestamp_96b_data[]` for the transmit frame whose fingerprint is specified by `tx_egress_timestamp_96b_fingerprint[]`.

Signal	Direction	Width	Description
`tx_egress_timestamp_96b_data[][]`	output	`[NUM_CHANNELS][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[][]`	output	`[NUM_CHANNELS][TSTAMP_FP_WIDTH]`	The fingerprint of the transmit frame, which is received on `tx_egress_timestamp_request_data[]`. This fingerprint specifies the transmit frame the egress timestamp on `tx_egress_timestamp_96b_data[]` is for.
`tx_egress_timestamp_64b_valid[]`	output	`[NUM_CHANNELS]`	When asserted, this signal qualifies the timestamp on `tx_egress_timestamp_64b_data[]` for the transmit frame whose fingerprint is specified by `tx_egress_timestamp_64b_fingerprint[]`.
`tx_egress_timestamp_64b_data[][]`	output	`[NUM_CHANNELS][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[][]`	output	`[NUM_CHANNELS][TSTAMP_FP_WIDTH]`	The fingerprint of the transmit frame, which is received on `tx_egress_timestamp_request_data[]`. This fingerprint specifies the transmit frame the egress timestamp on `tx_egress_timestamp_64b_data[]` is for.

Signal	Direction	Width	Description
rx_ingress_timestamp_96b_valid[]	output	[NUM_CHANNELS]	When asserted, this signal qualifies the timestamp on 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[][]	output	[NUM_CHANNELS][96]	Carries the 96-bit ingress 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
rx_ingress_timestamp_64b_valid[]	output	[NUM_CHANNELS]	When asserted, this signal qualifies the timestamp on 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[][]	output	[NUM_CHANNELS][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 16: Packet Classifier Interface Signals

Signal	Direction	Width	Description
tx_egress_timestamp_request_in_valid[]	input	[NUM_CHANNELS]	Assert this signal when timestamp is required for the particular frame. This signal must be aligned to the start of an incoming packet.

Signal	Direction	Width	Description
tx_egress_timestamp_ request_in_fingerprint[][]	input	[NUM_CHANNELS][TSTAMP_ FP_WIDTH]	A width-configurable fingerprint that correlates timestamps for incoming packets.
clock_operation_mode_ mode[][]	input	[NUM_CHANNELS][2]	Determines 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[]	input	[NUM_CHANNELS]	Indicates whether or not a packet contains CRC. • 0: Packet contains CRC • 1: Packet does not contain CRC
tx_ingress_timestamp_ valid[]	input	[NUM_CHANNELS]	Indicates the update for residence time. • 0: Prevents update for residence time • 1: Allows update for residence time based on decoded results When this signal is deasserted, tx_etstamp_ins_ctrl_out_ residence_ti me_update also gets deasserted.
tx_ingress_timestamp_96b_ data[][]	input	[NUM_CHANNELS][96]	96-bit format of ingress timestamp that holds data so that the output can align with the start of an incoming packet.
tx_ingress_timestamp_64b_ data[][]	input	[NUM_CHANNELS][64]	64-bit format of ingress timestamp that holds data so that the output can align with the start of an incoming packet.

Signal	Direction	Width	Description
`tx_ingress_timestamp_format[]`	input	`[NUM_CHANNELS]`	Format of the timestamp to be used for calculating residence time. This signal must be aligned to the start of an incoming packet. A value of 0 indicates 96 bit timestamp format while 1 indicates 64 bit timestamp format.

ToD Interface Signals

Table 17: ToD Interface Signals

Signal	Direction	Width	Description
`master_pulse_per_second`	output	1	Pulse per second output from Master PPS module. The pulse per second output asserts for 10ms.
`start_tod_sync[]`	input	`[NUM_CHANNELS]`	Start TOD synchronization process. As long as this signal is asserted high, the synchronization process will continue and time of day from master TOD will be repeatedly synchronized to local TOD.
`pulse_per_second_10g[]`	output	`[NUM_CHANNELS]`	Pulse per second output from 10G PPS module in channel-n. The pulse per second output asserts for 10ms.
`pulse_per_second_1g[]`	output	`[NUM_CHANNELS]`	Pulse per second output from 1G PPS module in channel-n. The pulse per second output asserts for 10ms.

Register Map

MSA0 is a 32-bit memory space address that provides access to all the client logic and scalable 1G/10G design example configuration registers. All registers in this space are 32-bit registers. Access smaller than 32 bits are not supported.

Table 18: Design Example Block Register Map

The following table shows the address offset for the design example and client logic at the design example level.

Byte Offset	Block
0x00_0000 - 0x00_EFFF	Client Logic
0x00_F000 - 0x00_FFFF	1G/10G Ethernet Reconfig Controller
0x01_0000	Master TOD
0x02_0000	Port 0
0x03_0000	Port 1
0x04_0000	Port 2
0x05_0000	Port 3
0x06_0000	Port 4
0x07_0000	Port 5
0x08_0000	Port 6
0x09_0000	Port 7
0x0A_0000	Port 8
0x0B_0000	Port 9
0x0C_0000	Port 10
0x0D_0000	Port 11
0x0E_0000 onwards	Client Logic

Table 19: Port Sub-block Register Map

The following table shows the address offset for the design example logic for each port.

Byte Offset	Sub-Block
0x0000 - 0x6FFF	Altera Logic
0x4000	PHY
0x7800	10G TOD
0x7900	1G TOD
0x8000	1G/10G MAC

Master TOD

Master TOD registers are applicable only to design examples with IEEE 1588v2.

The base address of the Master ToD registers are defined as follows:

- Master TOD Base Address = MSA0 + 0x01_0000

Table 20: Register Description and Address Offset for 1588 TOD Clock

Byte Offset	R/W	Name	Description	HW Reset
0x0000	RW	SecondsH	• Bits 0 to 15: High-order 16-bit second field • Bits 16 to 31: Not used.	0x0
0x0004	RW	SecondsL	Bits 0 to 32: Low-order 32-bit second field.	0x0
0x0008	RW	NanoSec	Bits 0 to 32: 32-bit nanosecond field.	0x0
0x0010	RW	Period	• Bits 0 to 15: Period in fractional nanosecond • Bits 16 to 19: Period in nanosecond • Bits 20 to 31: Not used.	N
0x0014	RW	AdjustPeriod	The period for the offset adjustment. • Bits 0 to 15: Period in fractional nanosecond • Bits 16 to 19: Period in nanosecond • Bits 20 to 31: Not used.	0x0
0x0018	RW	AdjustCount	• Bits 0 to 19: The number of AdjustPeriod clock cycles used during offset adjustment • Bits 20 to 31: Not used.	0x0
0x001C	RW	DriftAdjust	The drift of ToD adjusted periodically by adding a correction value as configured in this register space. • Bits 0 to 15: Adjustment value in fractional nanosecond (DRIFT_ADJUST_FNS). This value is added into the current ToD during the adjustment. • Bits 16 to 19: Adjustment value in nanosecond (DRIFT_ADJUST_NS). This value is added into the current ToD during the adjustment. • Bits 20 to 32: Not used.	0x0
0x0020	RW	DriftAdjustRate	The count of clock cycles for each ToD's drift adjustment to take effect. • Bits 0 to 15: The number of clock cycles (ADJUST_RATE). The ToD adjustment happens once after every period in number of clock cycles as indicated by this register space. • Bits 20 to 32: Not used.	0x0

1G TOD

1G TOD registers are applicable only to design examples with IEEE 1588v2.

Table 21: Base Address of 1G TOD Registers

Channel	1G TOD Register Base Address
0	MSA0 + 0x02_7900
1	MSA0 + 0x03_7900
2	MSA0 + 0x04_7900
3	MSA0 + 0x05_7900
4	MSA0 + 0x06_7900
5	MSA0 + 0x07_7900
6	MSA0 + 0x08_7900
7	MSA0 + 0x09_7900
8	MSA0 + 0x0A_7900
9	MSA0 + 0x0B_7900
10	MSA0 + 0x0C_7900
11	MSA0 + 0x0D_7900

Table 22: Register Description and Address Offset for 1588 TOD Clock

Byte Offset	R/W	Name	Description	HW Reset
0x0000	RW	SecondsH	• Bits 0 to 15: High-order 16-bit second field • Bits 16 to 31: Not used.	0x0
0x0004	RW	SecondsL	Bits 0 to 32: Low-order 32-bit second field.	0x0
0x0008	RW	NanoSec	Bits 0 to 32: 32-bit nanosecond field.	0x0
0x0010	RW	Period	• Bits 0 to 15: Period in fractional nanosecond • Bits 16 to 19: Period in nanosecond • Bits 20 to 31: Not used.	N
0x0014	RW	AdjustPeriod	The period for the offset adjustment. • Bits 0 to 15: Period in fractional nanosecond • Bits 16 to 19: Period in nanosecond • Bits 20 to 31: Not used.	0x0
0x0018	RW	AdjustCount	• Bits 0 to 19: The number of AdjustPeriod clock cycles used during offset adjustment • Bits 20 to 31: Not used.	0x0

Byte Offset	R/W	Name	Description	HW Reset
0x001C	RW	DriftAdjust	The drift of ToD adjusted periodically by adding a correction value as configured in this register space. • Bits 0 to 15: Adjustment value in fractional nanosecond (DRIFT_ADJUST_FNS). This value is added into the current ToD during the adjustment. • Bits 16 to 19: Adjustment value in nanosecond (DRIFT_ADJUST_NS). This value is added into the current ToD during the adjustment. • Bits 20 to 32: Not used.	0x0
0x0020	RW	DriftAdjustRate	The count of clock cycles for each ToD's drift adjustment to take effect. • Bits 0 to 15: The number of clock cycles (ADJUST_RATE). The ToD adjustment happens once after every period in number of clock cycles as indicated by this register space. • Bits 20 to 32: Not used.	0x0

10G TOD

10G TOD registers are applicable only to design examples with IEEE 1588v2.

Table 23: Base Address of 10G TOD Registers

Channel	10G TOD Register Base Address
0	MSA0 + 0x02_7800
1	MSA0 + 0x03_7800
2	MSA0 + 0x04_7800
3	MSA0 + 0x05_7800
4	MSA0 + 0x06_7800
5	MSA0 + 0x07_7800
6	MSA0 + 0x08_7800
7	MSA0 + 0x09_7800
8	MSA0 + 0x0A_7800
9	MSA0 + 0x0B_7800
10	MSA0 + 0x0C_7800
11	MSA0 + 0x0D_7800

Table 24: Register Description and Address Offset for 1588 TOD Clock

Byte Offset	R/W	Name	Description	HW Reset
0x0000	RW	SecondsH	• Bits 0 to 15: High-order 16-bit second field • Bits 16 to 31: Not used.	0x0
0x0004	RW	SecondsL	Bits 0 to 32: Low-order 32-bit second field.	0x0
0x0008	RW	NanoSec	Bits 0 to 32: 32-bit nanosecond field.	0x0
0x0010	RW	Period	• Bits 0 to 15: Period in fractional nanosecond • Bits 16 to 19: Period in nanosecond • Bits 20 to 31: Not used.	N
0x0014	RW	AdjustPeriod	The period for the offset adjustment. • Bits 0 to 15: Period in fractional nanosecond • Bits 16 to 19: Period in nanosecond • Bits 20 to 31: Not used.	0x0
0x0018	RW	AdjustCount	• Bits 0 to 19: The number of AdjustPeriod clock cycles used during offset adjustment • Bits 20 to 31: Not used.	0x0
0x001C	RW	DriftAdjust	The drift of ToD adjusted periodically by adding a correction value as configured in this register space. • Bits 0 to 15: Adjustment value in fractional nanosecond (DRIFT_ADJUST_FNS). This value is added into the current ToD during the adjustment. • Bits 16 to 19: Adjustment value in nanosecond (DRIFT_ADJUST_NS). This value is added into the current ToD during the adjustment. • Bits 20 to 32: Not used.	0x0
0x0020	RW	DriftAdjustRate	The count of clock cycles for each ToD's drift adjustment to take effect. • Bits 0 to 15: The number of clock cycles (ADJUST_RATE). The ToD adjustment happens once after every period in number of clock cycles as indicated by this register space. • Bits 20 to 32: Not used.	0x0

PHY

PHY registers are applicable to both design examples.

Table 25: Base Address of PHY Registers

Channel	PHY Register Base Address
0	MSA0 + 0x02_4000
1	MSA0 + 0x03_4000
2	MSA0 + 0x04_4000
3	MSA0 + 0x05_4000
4	MSA0 + 0x06_4000
5	MSA0 + 0x07_4000
6	MSA0 + 0x08_4000
7	MSA0 + 0x09_4000
8	MSA0 + 0x0A_4000
9	MSA0 + 0x0B_4000
10	MSA0 + 0x0C_4000
11	MSA0 + 0x0D_4000

Note: For the description of each PHY register, refer to the Altera Transceiver PHY IP Core User Guide. The address offset in the following tables is in byte, while the register map table in the Altera Transceiver PHY IP Core User Guide is in word.

Table 26: PMA Registers

Byte Offset	Bit	R/W	Name
0x0088		RO	`pma_tx_pll_is_locked`
0x0110	1	RW	`reset_tx_digital`
	2	RW	`reset_rx_analog`
	3	RW	`reset_rx_digital`
0x0184		RW	`phy_serial_loopback`
0x0190		RW	`pma_rx_set_locktodata`
0x0194		RW	`pma_rx_set_locktoref`
0x0198		RO	`pma_rx_is_lockedtodata`
0x019C		RO	`pma_rx_is_lockedtoref`

Byte Offset	Bit	R/W	Name
0x02A0	0	RW	`tx_invpolarity`
	1	RW	`rx_invpolarity`
	2	RW	`rx_bitreversal_enable`
	3	RW	`rx_bytereversal_enable`
	4	RW	`force_electrical_idle`
0x02A4	0	R	`rx_syncstatus`
	1	R	`rx_patterndetect`
	2	R	`rx_rlv`
	3	R	`rx_rmfifodatainserted`
	4	R	`rx_rmfifodatadeleted`
	5	R	`rx_disperr`
	6	R	`rx_errdetect`

Table 27: PCS Registers

Byte Offset	Bit	R/W	Name
0x0200		RW	`Indirect_addr`
0x0204	2	RW	`RCLR_ERRBLK_CNT`
	3	RW	`RCLR_BER_COUNT`
0x0208	1	RO	`HI_BER`
	2	RO	`BLOCK_LOCK`
	3	RO	`TX_FULL`
	4	RO	`RX_FULL`
	5	RO	`RX_SYNC_HEAD_ERROR`
	6	RO	`RX_SCRAMBLER_ERROR`
	7	RO	`Rx_DATA_READY`

Table 28: 1G/10GbE GMII PCS Registers

Byte Offset	Bit	R/W	Name
0x0240	9	RW	`RESTART_AUTO_ NEGOTIATION`
	12	RW	`AUTO_NEGOTIATION_ ENABLE`
	15	RW	`Reset`

Byte Offset	Bit	R/W	Name
0x0244	2	R	`LINK_STATUS`
	3	R	`AUTO_NEGOTIATION_ ABILITY`
	5	R	`AUTO_NEGOTIATION_ COMPLETE`
0x0250	5	RW	`FD`
	6	RW	`HD`
	8:7	RW	`PS2,PS1`
	13:12	RW	`RF2,RF1`
	14	R0	`ACK`
	15	RW	`NP`
0x0254	5	R	`FD`
	6	R	`HD`
	8:7	R	`PS2,PS1`
	13:12	R	`RF2,RF1`
	14	R	`ACK`
	15	R	`NP`
0x0258	0	R	`LINK_PARTNER_AUTO_NEGOTIATION_ABLE`
	1	R	`PAGE_RECEIVE`
0x0288	15:0	RW	`AN link timer[15:0]`
0x028C	4:0	RW	`AN link timer[4:0]`
0x0290	0	RW	`SGMII_ENA`
	1	RW	`USE_SGMII_AN`
	3:2	RW	`SGMII_SPEED`

Table 29: 1G/10GbE Register Definitions

Byte Offset	Bit	R/W	Name
0x02C0	0	RW	`Reset SEQ`
	1	RW	`Disable AN Timer`
	2	RW	`Disable LF Timer`
	6:4	RW	`SEQ Force Mode[2:0]`
	16	RW	`FEC ability`
	18	RW	`FEC request`

Byte Offset	Bit	R/W	Name
0x02C4	0	R	`SEQ Link Ready`
	1	R	`SEQ AN timeout`
	2	R	`SEQ LT timeout`
	13:8	RW	`SEQ Reconfig Mode[5:0]`
	16	R	`KR FEC ability`
	17	R	`KR FEC err ind ability`

Related Information

Altera Transceiver PHY IP Core User Guide

1G/10G MAC

MAC registers are applicable to both design examples.

Table 30: Base Address of 1G/10G MAC Registers

Channel	PHY Register Base Address
0	MSA0 + 0x02_8000
1	MSA0 + 0x03_8000
2	MSA0 + 0x04_8000
3	MSA0 + 0x05_8000
4	MSA0 + 0x06_8000
5	MSA0 + 0x07_8000
6	MSA0 + 0x08_8000
7	MSA0 + 0x09_8000
8	MSA0 + 0x0A_8000
9	MSA0 + 0x0B_8000
10	MSA0 + 0x0C_8000
11	MSA0 + 0x0D_8000

Note: For the description of each MAC register, refer to the 10-Gbps Ethernet MAC Megacore Function User Guide. The address offset in the following tables is in byte, while the register map table in the 10-Gbps Ethernet MAC Megacore Function User Guide is in word.

Table 31: MAC Register Components and Offset Range

Component	Byte Offset Range
RX Packet Transfer	0x0000:0x00FF
RX Pad/CRC Remover	0x0100:0x01FF

Component	Byte Offset Range
RX CRC Checker	0x0200:0x02FF
RX Packet Overflow	0x0300:0x03FF
RX Preamble Control	0x0400:0x04FF
RX Lane Decoder	0x0500:0x1FFF
RX Frame Decoder	0x2000:0x2FFF
RX Statistics Counters	0x3000:0x3FFF
TX Packet Transfer	0x4000:0x40FF
TX Pad Inserter	0x4100:0x41FF
TX CRC Inserter	0x4200:0x42FF
TX Packet Underflow	0x4300:0x43FF
TX Preamble Control	0x4400:0x44FF
TX Pause Frame Control and Generator	0x4500:0x45FF
TX PFC Generator	0x4600:0x47FF
TX Address Inserter	0x4800:0x5FFF
TX Frame Decoder	0x6000:0x6FFF
TX Statistics Counters	0x7000:0x7FFF

Table 32: 1G/10G MAC Registers for IEEE 1588v2 Feature

Register	Byte Offset
rx_period	0x0440 (10G)
	0x0460 (1G)
rx_adjust_fns	0x0448 (10G)
	0x0468 (1G)
rx_adjust_ns	0x044C (10G)
	0x046C (1G)
tx_period	0x4440 (10G)
	0x4460 (1G)
tx_adjust_fns	0x4448 (10G)
	0x4468 (1G)
tx_adjust_ns	0x444C (10G)
	0x446C (1G)

Related Information

10-Gbps Ethernet MAC MegaCore Function User Guide

Known Issues

There are 3 illegal clocks reported in TimeQuest which can be ignored because it does not affect the functionality of the hardware.

- 10g pcs rxclkout
- 10g pcs txclkout
- clk33pcs

Document Revision History

Date	Version	Changes
May 2016	2016.05.13	Update the supported software version.
September 2015	2015.09.30	• Added note to a step in *Setting Up the Design Example* channel 0 and channel 1 default connection. • Added `DriftAdjust` and `DriftAdjustRate` registers to Master, 1G and 10G TOD register tables.
June 2015	2015.06.15	Updated supported ACDS, ModelSim and Synopsys versions.
January 2015	2015.01.22	Updated supported ACDS version.
December 2014	2014.12.29	Updated supported ACDS version.
October 2014	2014.10.23	Initial release