



AN 701: Scalable Low Latency Ethernet 10G MAC using Intel Arria 10 1G/10G PHY



Contents

Scalable Low Latency Ethernet 10G MAC using Intel Arria 10 1G/10G PHY.....	3
Features.....	3
Requirements.....	3
Design Example Walkthrough.....	4
Setting Up the Design Examples.....	4
Changing Number of Channels.....	4
Configuring PHY Speed.....	4
IP Regeneration.....	5
Components.....	5
Parameters.....	7
Design Example Testbench.....	8
Testbench Components.....	8
Testbench Files.....	8
Simulating Testbench.....	9
Test Case.....	10
Clocking Scheme.....	15
Clocking Diagram.....	16
Sync-E Support.....	17
Reset Scheme.....	19
Design Example without IEEE 1588v2.....	20
Design Example with IEEE 1588v2.....	21
Interface Signals.....	23
Clock and Reset Interface Signals.....	23
Avalon-MM Interface Signals.....	24
Avalon-ST Interface Signals.....	24
PHY Interface Signals.....	27
MDIO Interface Signals.....	28
IEEE 1588v2 Timestamp Interface Signals.....	28
Packet Classifier Interface Signals.....	29
ToD Interface Signals.....	30
Register Map.....	31
Master TOD.....	32
1G TOD.....	33
10G TOD.....	34
PHY.....	35
1G/10G MAC.....	38
Document Revision History for Scalable Low Latency Ethernet 10G MAC using Intel Arria 10 1G/10G PHY.....	43



Scalable Low Latency Ethernet 10G MAC using Intel Arria 10 1G/10G PHY

The following design examples demonstrate Intel® FPGA Low Latency Ethernet 10G MAC IP systems using Intel Arria® 10 PHY.

- Scalable Low Latency Ethernet 10G MAC using Intel Arria 10 1G/10G PHY without IEEE 1588v2
- Scalable Low Latency Ethernet 10G MAC using Intel Arria 10 1G/10G PHY with IEEE 1588v2

These design examples support only Intel Arria 10 devices.

Related Links

- [Scalable Low Latency Ethernet 10G MAC using Arria 10 1G/10G PHY without IEEE 1588v2](#)
- [Scalable Low Latency Ethernet 10G MAC using Arria 10 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 Intel Arria 10 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.

Requirements

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

- Intel Quartus® Prime version 15.0 for software and hardware simulation
- ModelSim-SE 10.3d
- Synopsys VCS Version I-2014.03-SP1

Note: Upgrading prior versions of the design example is not supported in the current ACDS version. Please use the latest design example files.



Design Example Walkthrough

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

Design Example	File Name
Scalable Low Latency Ethernet 10G MAC using Intel Arria 10 1G/10G PHY without IEEE 1588v2	LL_Ethernet_10G_A10_phy_lineside.tar.gz
Scalable Low Latency Ethernet 10G MAC using Intel Arria 10 1G/10G PHY with IEEE 1588v2	LL_Ethernet_10G_A10_phy_lineside_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.
2. Launch the Intel Quartus Prime software and open the project file, **altera_eth_top.qpf**.
3. Click **Start Compilation** on the Processing menu to compile the design example.

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 **altera_eth_multi_channel** or **altera_eth_multi_channel_1588**.

Configuring PHY Speed

After reset, all ports are set in 10G and auto speed detection mode. Use the PHY memory map to change to other modes: 10G SerDes Framer Interface (SFI), 1G1000Base-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 0x12C0.

Table 1. Register Value for Speed Change in 1000BaseX Mode in Intel Arria 10 Transceiver PHY IP

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

Example 1. Forcing Port 0 to 1000Base-X mode

- Set Port 0 to 1000Base-X: `write_32 0x02_52C0 0x11`

Example 2. Reset Port 0 to auto speed detection mode

- Set Port 0 to 1000Base-X: `write_32 0x02_52C0 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 0x1290. Set the port to 1000Base-X mode first before you select any SGMII modes.

Table 2. Register Value for Speed Change in SGMII Mode in Intel Arria 10 Transceiver PHY IP

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

Example 3. Forcing Port 0 to SGMII 100M mode

1. Set Port 0 to 1000Base-X: `write_32 0x02_52C0 0x11`
2. Set Port 0 to SGMII 100M: `write_32 0x02_5290 0x05`

IP Regeneration

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

Table 3. IP Requires Regeneration

IP	Tools	IP File Locations
address_decoder_channel	Platform Designer	ADDRESS_DECODER/address_decoder_channel.qsys
address_decoder_multi_channel	Platform Designer	ADDRESS_DECODER/address_decoder_multi_channel.qsys
altera_eth_10g_mac	IP Catalog	MAC_NO_1588/altera_eth_10g_mac.qsys
altera_eth_10gkr_phy	IP Catalog	PHY_NO_1588/altera_eth_10gkr_phy.qsys
altera_xcvr_reset_controller	IP Catalog	XCVR_RESET_CONTROLLER/altera_xcvr_reset_controller.qsys
nf_xcvr_10g_pll	IP Catalog	nf_atx_pll/nf_xcvr_10g_pll.qsys
nf_xcvr_1g_pll	IP Catalog	nf_xcvr_fpll/nf_xcvr_1g_pll.qsys
pll	IP Catalog	PLL/pll.qsys
pll_2 <i>Note: This IP is only available for design example with IEEE 1588v2.</i>	IP Catalog	PLL/pll_2.qsys

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

Components



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

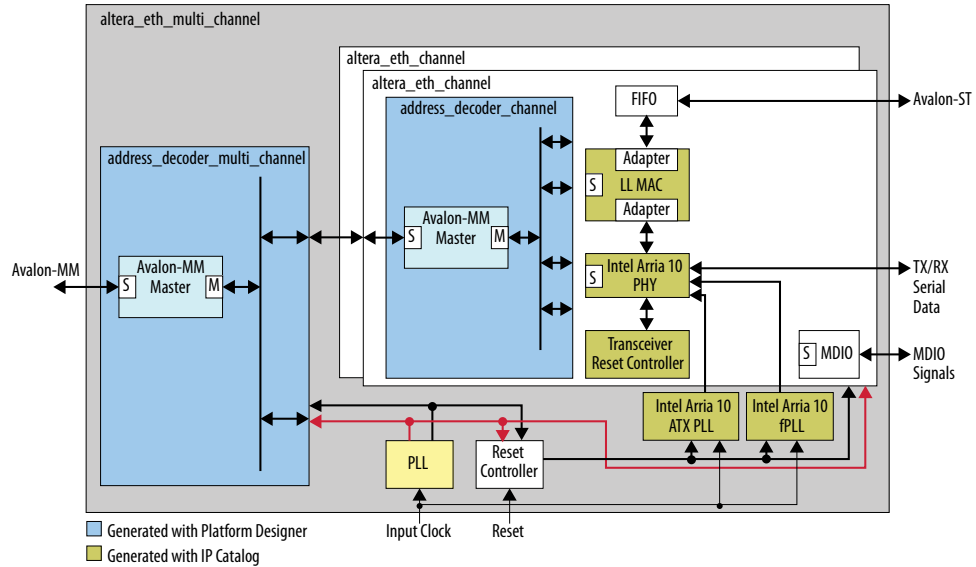


Figure 2. Block Diagram for Design Example with IEEE 1588v2

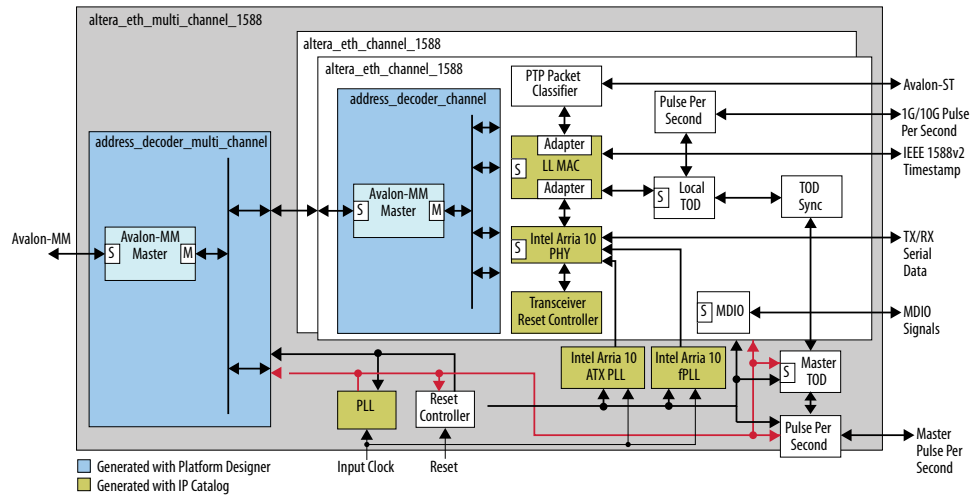


Table 4. Design Examples Components

Component	Design Example without IEEE 1588v2	Design Example with IEEE 1588v2
Low latency Ethernet 10G MAC	Ethernet MAC IP core	
Intel Arria 10 1G/10G PHY	Intel FPGA 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.	
Address decoder multi-channel	Address decoder module for all channels and components within multi-channel level, for example Master TOD.	

continued...



Component	Design Example without IEEE 1588v2	Design Example with IEEE 1588v2
Reset controller	Reset modules which handle reset synchronisation for the components in the design example.	
Master PLL	Generates clocks for all the components in the design example.	
Intel Arria 10 ATX PLL	Generates a TX serial clock for Intel Arria 10 10G transceiver.	
Intel Arria 10 fractional PLL	Generates a TX serial clock for Intel Arria 10 1G transceiver.	
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.	—

Parameters

Table 5. 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 <code>refclk</code> between all channels. <ul style="list-style-type: none"> • 0 : disable sharing • 1 : enable sharing 	1
FIFO_OPTIONS	Use this parameter to enable the FIFO in between user Avalon-ST and MAC interface. <ul style="list-style-type: none"> • 0: disable FIFO • 1: enable SC FIFO • 2: enable DC FIFO • 3: enable SC + DC FIFO <i>Note:</i> 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. <i>Note:</i> This parameter is available only for design example with IEEE 1588v2.	4

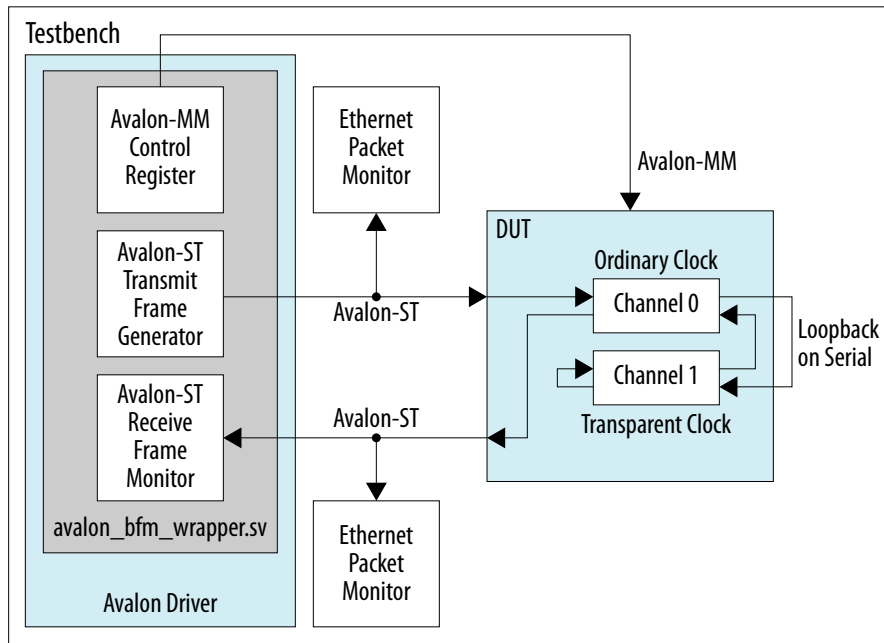
Design Example Testbench

Intel 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.

Figure 3. Testbench Block Diagram for Design Examples



The following table lists the components in the testbench.

Table 6. 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>/LL_Ethernet_10G_A10_phy_lineside/testbench/<Modelsim or VCS>/testcase<n>
Design example with IEEE 1588v2	<project directory>/LL_Ethernet_10G_A10_phy_lineside_1588/testbench/<Modelsim or VCS>/testcase<n>

The following table describes the files that implement the design example testbench.

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 BFM that the avalon_driver.sv file uses.
avalon_driver.sv	A SystemVerilog HDL driver that uses the BFM 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 BFM. 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.
tb_testcase.sv	A SystemVerilog HDL testbench file that controls the flow of the testbench.
tb_top_n.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 Intel FPGA simulation model libraries. You have to set the **QUARTUS_ROOTDIR** to point to the Intel Quartus Prime installation path after installation. If this environment variable is missing, then you must set the variable manually.

Using ModelSim Simulator

To use the ModelSim simulator to simulate the testbench designs, follow these steps:

1. Change the directory.



- For design example without IEEE 1588v2, change directory to <project directory>/LL_Ethernet_10G_A10_phy_lineside/testbench/Modelsim/testcase<n>
 - For design example with IEEE 1588v2, change directory to <project directory>/LL_Ethernet_10G_A10_phy_lineside_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

To use the VCS simulator to simulate the testbench designs, follow these steps:

1. Change the directory.
 - For design example without IEEE 1588v2, change the directory to: <project directory>/LL_Ethernet_10G_A10_phy_lineside/testbench/VCS/testcase<n>.
 - For design example with IEEE 1588v2, change the directory to: <project directory>/LL_Ethernet_10G_A10_phy_lineside_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.

This test case uses the following configuration:

- 2 channels
- Circular loopback

Test Scenario for Design Example without IEEE 1588v2

To perform test case, follow these steps:

1. Set the start up with channel configured to 10G mode.
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
 - SVLAN data frame, broadcast, 64Bytes
 - VLAN data frame, unicast, 500Bytes



- Repeat steps 2 to 4 for 1G, 100M and 10M speed mode.
- When the simulation ends, refer to the transcript window for channel 0 MAC TX and RX statistic counter results.

Figure 4. Channel 0 MAC TX Statistic Counter

```
# -----  
# Channel 0: TX Statistics  
# -----  
# framesOK = 12  
# framesErr = 0  
# framesCRCErr = 0  
# octetsOK = 2296  
# pauseMACCtrlFrames = 0  
# ifErrors = 0  
# unicastFramesOK = 8  
# unicastFramesErr = 0  
# multicastFramesOK = 0  
# multicastFramesErr = 0  
# broadcastFramesOK = 4  
# broadcastFramesErr = 0  
# etherStatsOctets = 2560  
# etherStatsPkts = 12  
# etherStatsUndersizePkts = 0  
# etherStatsOversizePkts = 0  
# etherStatsPkts64Octets = 0  
# etherStatsPkts65to127Octets = 8  
# etherStatsPkts128to255Octets = 0  
# etherStatsPkts256to511Octet = 4  
# etherStatsPkts512to1023Octets = 0  
# etherStatsPkts1024to1518Octets = 0  
# etherStatsPkts15190toXOctets = 0  
# etherStatsFragments = 0  
# etherStatsJabbers = 0  
# etherStatsCRCErr = 0  
# unicastMACCtrlFrames = 0  
# multicastMACCtrlFrames = 0  
# broadcastMACCtrlFrames = 0  
#
```



Figure 5. Channel 0 MAC RX Statistic Counter

```
# -----  
# Channel 0: RX Statistics  
# -----  
# framesOK = 12  
# framesErr = 0  
# framesCRCErr = 0  
# octetsOK = 2296  
# pauseMACCtrlFrames = 0  
# ifErrors = 0  
# unicastFramesOK = 8  
# unicastFramesErr = 0  
# multicastFramesOK = 0  
# multicastFramesErr = 0  
# broadcastFramesOK = 4  
# broadcastFramesErr = 0  
# etherStatsOctets = 2560  
# etherStatsPkts = 12  
# etherStatsUndersizePkts = 0  
# etherStatsOversizePkts = 0  
# etherStatsPkts64Octets = 0  
# etherStatsPkts65to127Octets = 8  
# etherStatsPkts128to255Octets = 0  
# etherStatsPkts256to511Octet = 4  
# etherStatsPkts512to1023Octets = 0  
# etherStatsPkts1024to1518Octets = 0  
# etherStatsPkts1519OtoXOctets = 0  
# etherStatsFragments = 0  
# etherStatsJabbers = 0  
# etherStatsCRCErr = 0  
# unicastMACCtrlFrames = 0  
# multicastMACCtrlFrames = 0  
# broadcastMACCtrlFrames = 0  
#
```

7. If channel 0 Avalon_st RX interface successfully receives all 12 packets, the transcript displays PASSED.



Test Scenario for Design Example with IEEE 1588v2

To perform test case, follow these steps:

1. Set the start up with channel configured to 10G mode.
2. Perform basic MAC configuration, PHY speed configuration, 1588 component configuration and PTP clock mode configuration for all 2 channels.
3. Wait for the design example to assert the `channel1_ready` signals for all 2 channels.
4. Send the following packets:
 - Non-PTP
 - No VLAN, PTP over Ethernet, PTP Sync Message, 1-step PTP
 - VLAN, PTPover UDP/IPv4, PTP Sync Message, 1-step PTP
 - Stacked VLAN, PTP over UDP/IPv6, PTP Sync Message, 2-step PTP
 - No VLAN, PTP over Ethernet, PTP Delay Request Message, 1-step PTP
 - VLAN, PTPover UDP/IPv4, PTP Delay Request Message, 2-step PTP
 - Stacked VLAN, PTP over UDP/IPv6, PTP Delay Request Message, 1-step PTP
5. Repeat steps 2 to 4 for 1G, 100M and 10M speed mode.
6. When the simulation ends, refer to the transcript window for channel 0 MAC TX and RX statistic counter results.



Figure 6. Channel 0 MAC TX Statistic Counter

```
# -----  
# 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  
#
```



Figure 7. Channel 0 MAC RX Statistic Counter

```
# -----  
# 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  
# etherStatsPkts15190toXOctets = 0  
# etherStatsFragments = 0  
# etherStatsJabbers = 0  
# etherStatsCRCErr = 0  
# unicastMACCtrlFrames = 0  
# multicastMACCtrlFrames = 0  
# broadcastMACCtrlFrames = 0  
# display_statistics
```

7. If channel 0 Avalon_st RX interface successfully receives all 28 packets, the transcript displays PASSED.

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 8. Clocking Scheme for the Design Example without IEEE 1588v2

Note: The IOPLL input reference clock is sourcing from input clock through the global clock network. Sourcing reference clock from a cascaded PLL output, global clock or core clock network may introduce additional jitter to the ATX/FPLL/IOPLL output. Refer to this [KDB Answer](#) for a workaround you should apply to the IP core in your design.

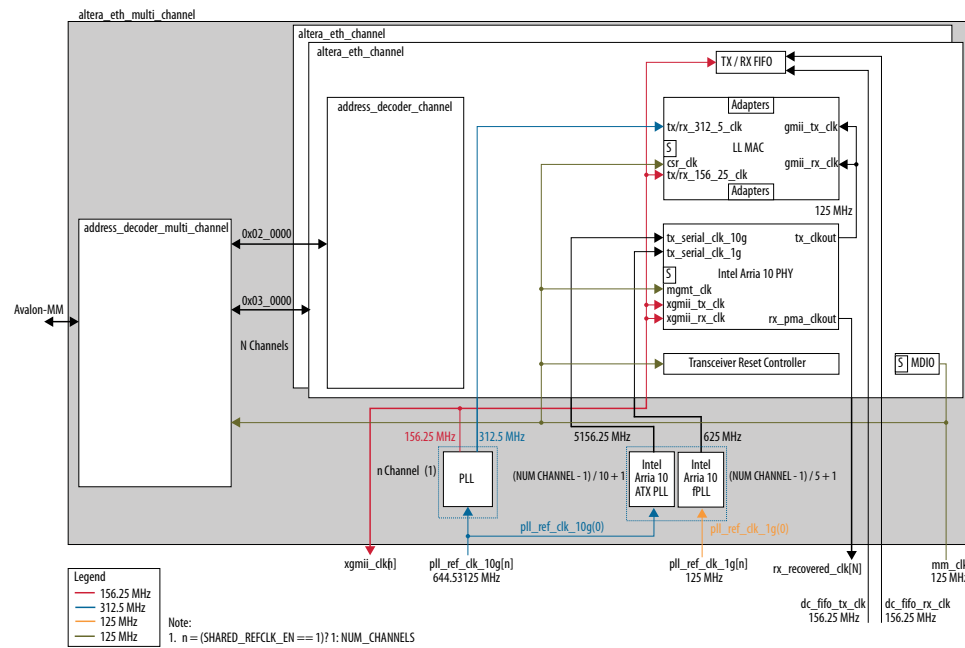
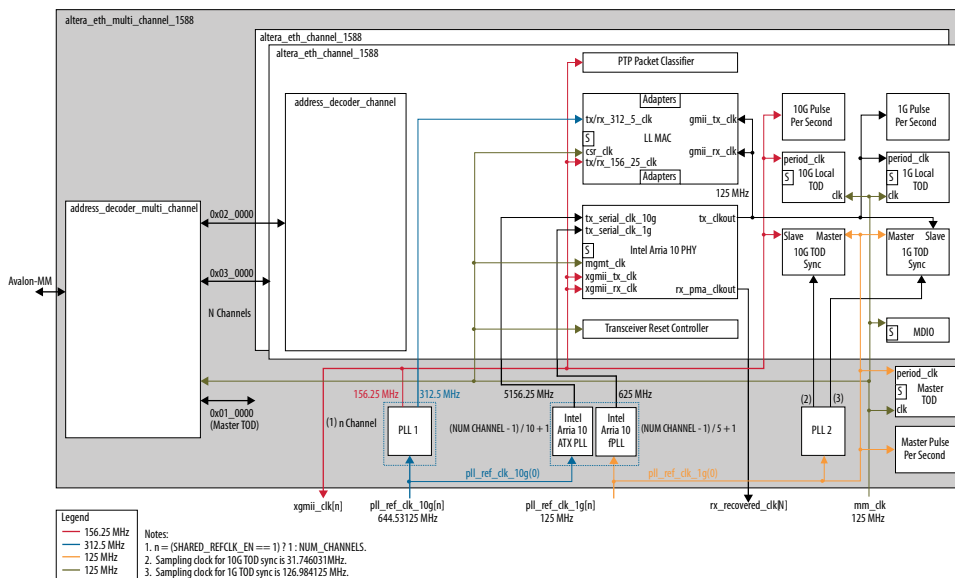




Figure 9. Clocking Scheme for the Design Example with IEEE 1588v2

Note: The IOPLL input reference clock is sourcing from input clock through the global clock network. Sourcing reference clock from a cascaded PLL output, global clock or core clock network may introduce additional jitter to the ATX/FPLL/IOPLL output. Refer to this [KDB Answer](#) for a workaround you should apply to the IP core in your design.



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 10. Signals from PHY to Support Sync-E Implementation for Design Example without IEEE 1588v2

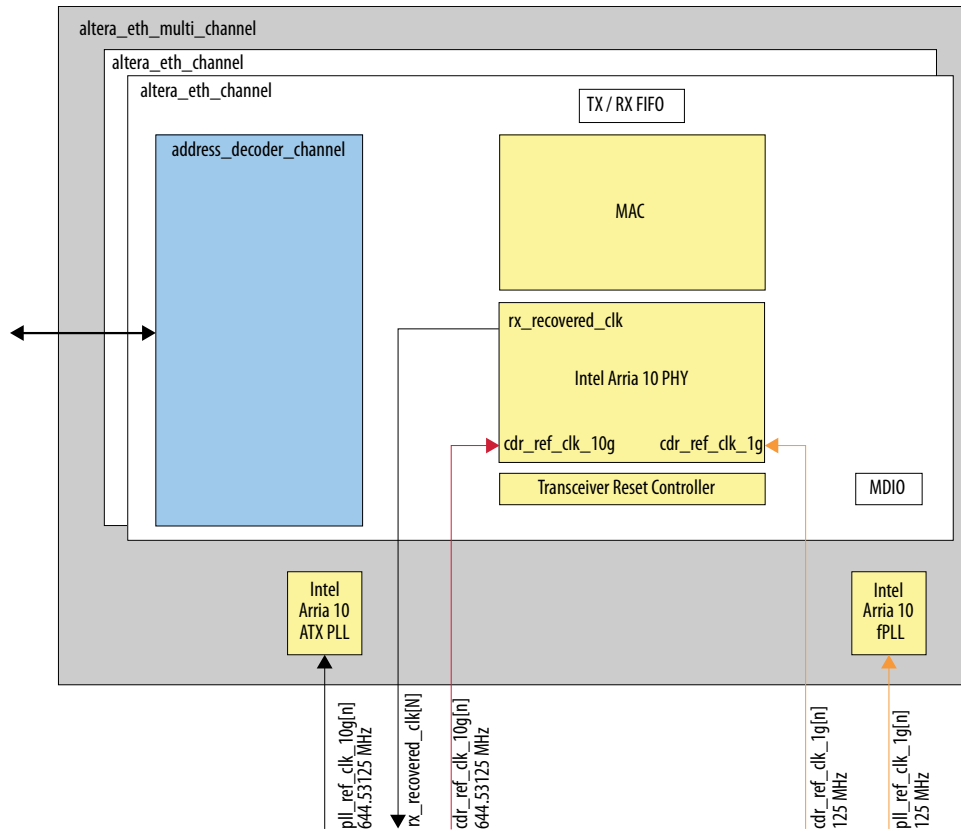
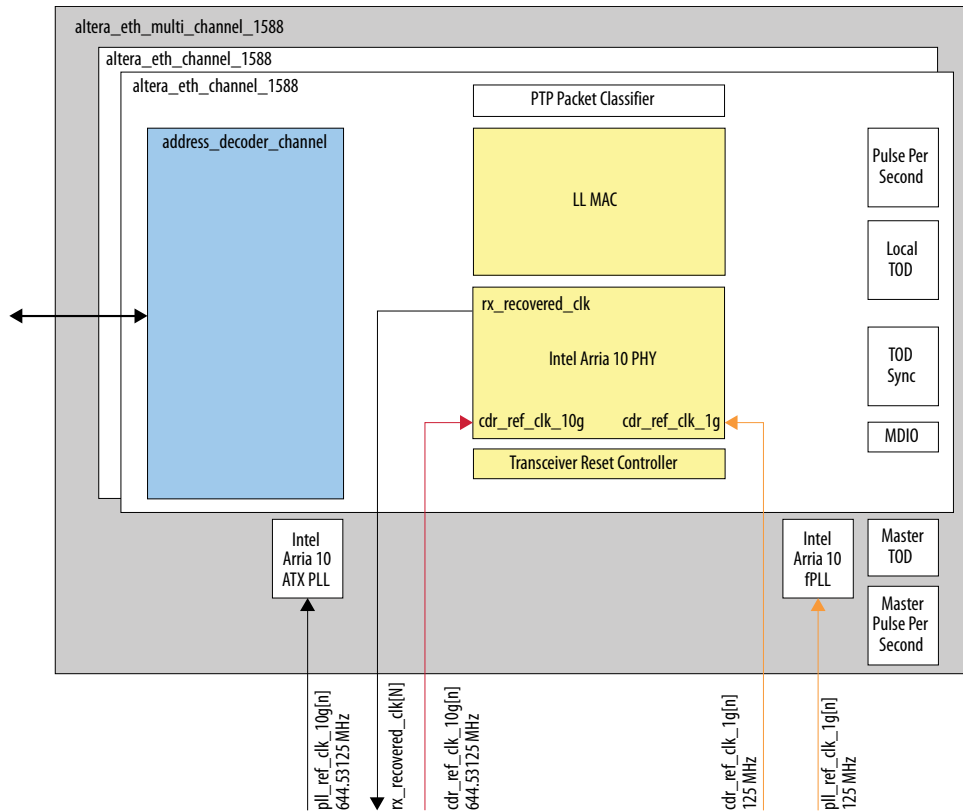




Figure 11. Signals from PHY to Support Sync-E Implementation for Design Example with IEEE 1588v2



Enable Ref Clock Sharing

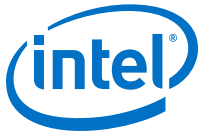
When user set the parameter `SHARED_REFCLK_EN` to 1, this will enable the ref clock sharing and only 1 set of `pll_ref_clk_10g`, `pll_ref_clk_1g`, `cdr_ref_clk_10g` and `cdr_ref_clk_1g` is needed. These ref clock signals will be used across all channels. There will be N number of `rx_recovered_clk` regardless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

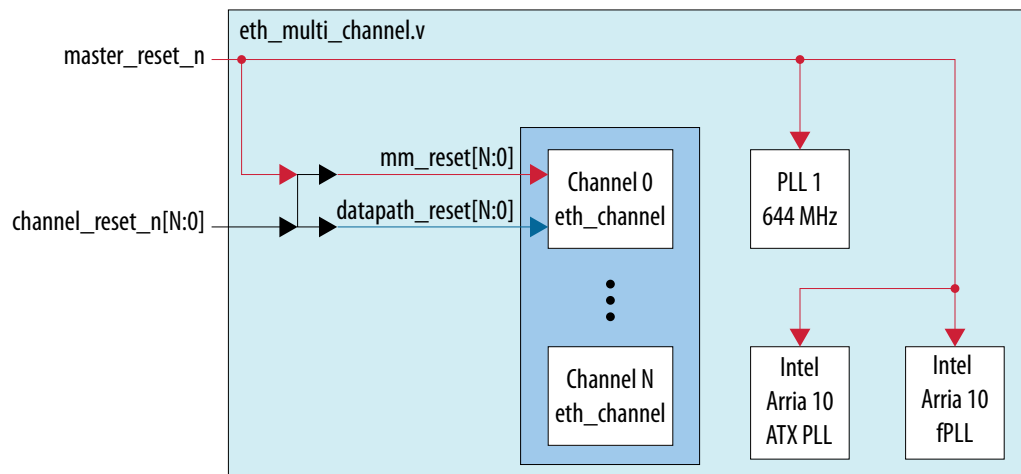
Master reset is needed when the design example is powered up.

Design Example without IEEE 1588v2

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.

Figure 12. Reset scheme at `altera_eth_multi_channel`



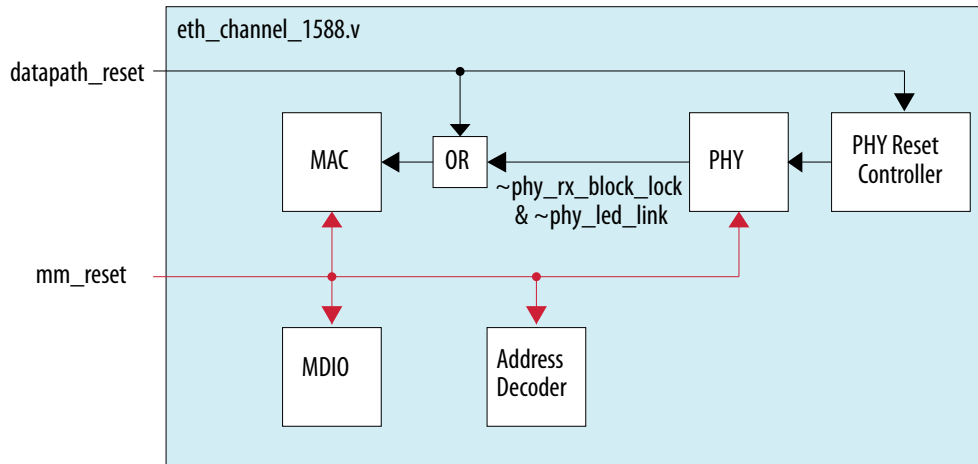
Note:
1. N is the number of channels.

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 13. Reset scheme at altera_eth_channel



Notes:

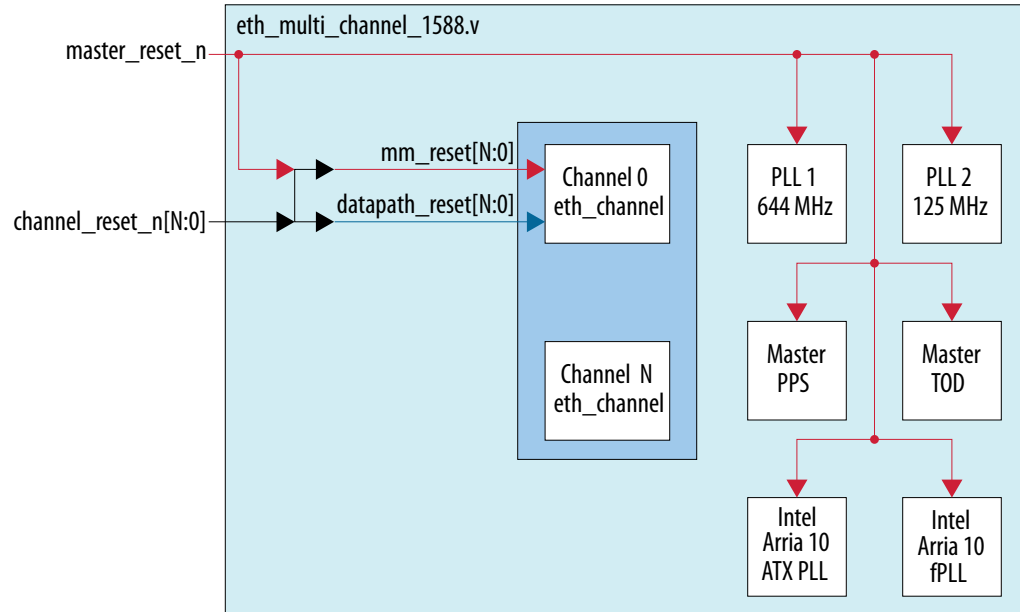
1. The mm_reset signal resets all Avalon-MM control blocks.
2. The datapath_reset signal resets all digital blocks, including the PHY reset controller.

Design Example with IEEE 1588v2

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.

Figure 14. Reset Scheme at altera_eth_multi_channel_1588



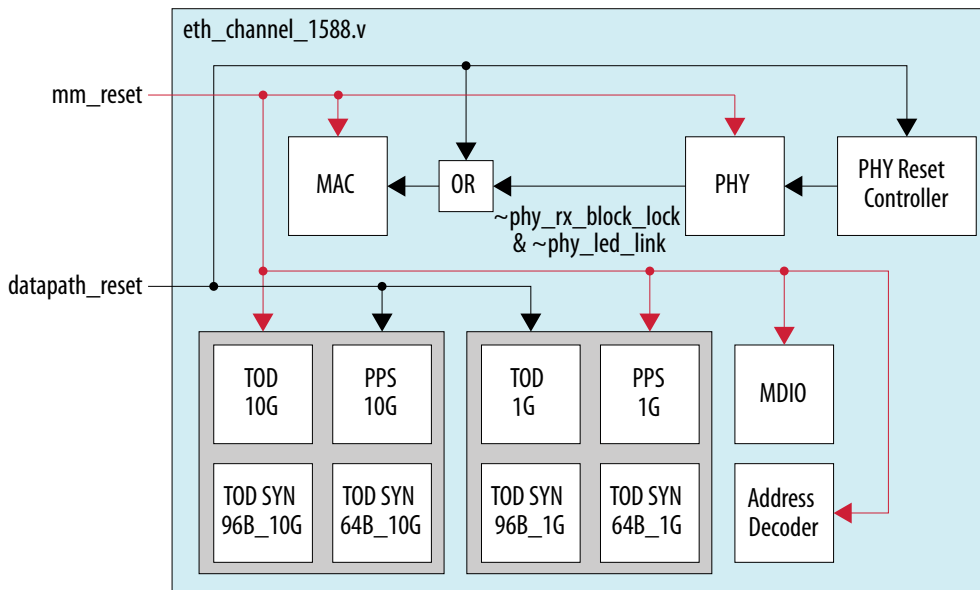
Note:
1. N is the number of channels.

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 15. Reset scheme at altera_eth_channel_1588



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

Interface Signals

This section describes the interface signals at design example level.

The NUM_UNSHARED_CHANNELS are determined by the equation below:

$$\text{NUM_UNSHARED_CHANNELS} = (\text{SHARED_REFCLK_EN} == 1) ? 1 : \text{NUM_CHANNELS}$$

NUM_CHANNELS and SHARED_REFCLK_EN are parameters set by user.

Clock and Reset Interface Signals

The following table lists the clock and reset interface signals. These interface signals are applicable to both design examples.

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.
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 644.53125 MHz.
cdr_ref_clk_1g[]	input	[NUM_UNSHARED_CHANNELS]	Reference clock for the RX PLL in 1G mode. Frequency is 125 MHz.

continued...



Signal	Direction	Width	Description
cdr_ref_clk_10g[]	input	[NUM_UNSHARED_CHANNELS]	Reference clock for the RX PLL in 10G mode. Frequency is 644.53125 MHz.
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 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

The following table lists the Avalon-MM interface signals. These interface signals are applicable to both design examples.

Table 10. Avalon MM Interface Signals

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

Avalon-ST Interface Signals

The following table lists the Avalon-ST interface signals. These interface signals are applicable to both design examples.

Table 11. Avalon-ST Interface Signals

Signal	Direction	Width	Description
avalon_st_tx_startofpacket[]	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.
<i>continued...</i>			



Signal	Direction	Width	Description
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.
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_startofpacket[]	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 <code>avalon_st_rx_data[]</code> 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.
avalon_st_rx_error[][]	output	[NUM_CHANNELS][64]	When set to 1, the respective bits indicate an error type: <ul style="list-style-type: none"> • Bit 0—PHY error. For 10 Gbps, the data on <code>xgmii_rx_data</code> contains a control error character (FE). For 10 Mbps, 100 Mbps, 1 Gbps, <code>gmii_rx_err</code> or <code>mii_rx_err</code> 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 <code>MAX_FRAME_SIZE</code>. • 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 <code>avalon_st_txstatus_data[]</code> and <code>avalon_st_txstatus_error[]</code> .
avalon_st_tx_status_data[][]	output	[NUM_CHANNELS][40]	Contains information about the transmit frame.

continued...



Signal	Direction	Width	Description
			<ul style="list-style-type: none"> • 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]	<p>When set to 1, the respective bit indicates the following error type in the receive frame.</p> <ul style="list-style-type: none"> • 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. <p>The error status is invalid when an overflow occurs.</p>
avalon_st_rxstatus_valid[]	output	[NUM_CHANNELS]	<p>When asserted, this signal qualifies <code>avalon_st_txstatus_data[]</code> and <code>avalon_st_txstatus_error[]</code>. The MAC IP core asserts this signal in the same clock cycle <code>avalon_st_rx_endofpacket</code> is asserted.</p>
avalon_st_rxstatus_data[][]	output	[NUM_CHANNELS][40]	<p>Contains information about the transmit frame.</p> <ul style="list-style-type: none"> • 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.

continued...



Signal	Direction	Width	Description
			<ul style="list-style-type: none"> • 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. <ul style="list-style-type: none"> • 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[][]	input	[NUM_CHANNELS][2]	Set this signal to the following values to trigger the corresponding actions. <ul style="list-style-type: none"> • 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. <i>Note:</i> This signal only takes effect if tx_pauseframe_enable[2:1] is 00 (default)

PHY Interface Signals

The following table lists the PHY interface signals. These interface signals are applicable to both design examples.

Table 12. 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_lg_an[]	output	[NUM_CHANNELS]	Clause 37 Auto-Negotiation status. The PCS function asserts this signal when auto-negotiation completes.
ethernet_lg_char_err[]	output	[NUM_CHANNELS]	10-bit character error
ethernet_lg_disp_err[]	output	[NUM_CHANNELS]	Disparity error signal indicating a 10-bit running disparity error.
channel_ready[]	output	[NUM_CHANNELS]	This signal is asserted when the channel is ready for data transmission.



MDIO Interface Signals

The following table lists the MDIO interface signals. These interface signals are applicable to both design examples.

Table 13. 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

The following table lists the IEEE 1588v2 timestamp interface signals. These interface signals are only applicable to design examples with IEEE 1588v2.

Table 14. 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[].
tx_egress_timestamp_96b_data[][]	output	[NUM_CHANNELS][96]	Carries the 96-bit egress timestamp in the following format: <ul style="list-style-type: none"> • 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:

continued...



Signal	Direction	Width	Description
			<ul style="list-style-type: none"> 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.
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: <ul style="list-style-type: none"> 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: <ul style="list-style-type: none"> Bits 16 to 63: 48-bit nanoseconds field Bits 0 to 15: 16-bit fractional nanoseconds field

Packet Classifier Interface Signals

The following table lists the packet classifier interface signals. These interface signals are only applicable to design examples with IEEE 1588v2.

Table 15. Packet Classifier Interface Signals

Signal	Direction	Width	Description
tx_egress_timestamp_request_in_valid[]	input	[NUM_CHANNELS]	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[][]	input	[NUM_CHANNELS] [TSTAMP_FP_WIDTH]	Use this bus to specify the fingerprint that validates the timestamp for the incoming packet.
<i>continued...</i>			



Signal	Direction	Width	Description
clock_operation_mode_mode[][]	input	[NUM_CHANNELS][2]	Determines the clock mode. <ul style="list-style-type: none"> • 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. <ul style="list-style-type: none"> • 0: Packet contains CRC • 1: Packet does not contain CRC
tx_ingress_timestamp_valid[]	input	[NUM_CHANNELS]	Indicates the update for residence time. <ul style="list-style-type: none"> • 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_time_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.
tx_ingress_timestamp_format[]	input	[NUM_CHANNELS]	The format of the timestamp for calculating the residence time. <ul style="list-style-type: none"> • 0: 96 bits • 1: 64 bits This signal must be aligned to the start of an incoming packet.

ToD Interface Signals

The following table lists the ToD interface signals. These interface signals are only applicable to design examples with IEEE 1588v2.



Table 16. ToD Interface Signals

Signal	Direction	Width	Description
master_pulse_per_second	output	1	Pulse per second (PPS) from master PPS module. The pulse per second output asserts for 10 ms.
start_tod_sync[]	input	[NUM_CHANNELS]	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.
pulse_per_second_10g[]	output	[NUM_CHANNELS]	PPS from the 10G PPS module of channel <i>n</i> . The signal stay asserted for 10 ms.
pulse_per_second_1g[]	output	[NUM_CHANNELS]	PPS from the 1G PPS module of channel <i>n</i> . The signal stay asserted for 10 ms.

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.

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

Table 17. Design Example Block Register Map

Byte Offset	Block
0x00_0000 - 0x00_EFFF	Client Logic
0x00_F000 - 0x00_FFFF	Intel FPGA Logic
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

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



Table 18. Port Sub-block Register Map

Byte Offset	Sub-block
0x0000 - 0x3FFF	Intel FPGA 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 19. Register Description and Address Offset for 1588 TOD Clock

Byte Offset	R/W	Name	Description	HW Reset
0x0000	RW	SecondsH	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • 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. <ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • 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. <ul style="list-style-type: none"> • 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. <ul style="list-style-type: none"> • 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 only applicable to design examples with IEEE 1588v2.

The base address of the 1G TOD registers are defined as follows:

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 20. Register Description and Address Offset for 1588 TOD Clock

Byte Offset	R/W	Name	Description	HW Reset
0x0000	RW	SecondsH	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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. <ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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. <ul style="list-style-type: none"> 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.	0x0

continued...



Byte Offset	R/W	Name	Description	HW Reset
			<ul style="list-style-type: none"> 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. 	

10G TOD

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

The base address of the 10G TOD registers are defined as follows:

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 21. Register Description and Address Offset for 1588 TOD Clock

Byte Offset	R/W	Name	Description	HW Reset
0x0000	RW	SecondsH	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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. <ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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.	0x0

continued...



Byte Offset	R/W	Name	Description	HW Reset
			<ul style="list-style-type: none"> 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. 	
0x0020	RW	DriftAdjustRate	The count of clock cycles for each ToD's drift adjustment to take effect. <ul style="list-style-type: none"> 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.

The base address of the PHY registers are defined as follows:

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 [Intel Arria 10 Transceiver PHY User Guide](#).

Note: The address offset in the following tables is in byte, while the register map table in the [Intel Arria 10 Transceiver PHY User Guide](#) is in word.

Table 22. PMA Registers

Byte Offset	Bit	R/W	Name
0x1110	1	RW	reset_tx_digital
	2	RW	reset_rx_analog
	3	RW	reset_rx_digital
<i>continued...</i>			



Byte Offset	Bit	R/W	Name
0x1184		RW	phy_serial_loopback
0x1190		RW	pma_rx_set_locktodata
0x1194		RW	pma_rx_set_locktoref
0x1198		RO	pma_rx_is_lockedtodata
0x119C		RO	pma_rx_is_lockedtoref
0x12A0	0	RW	tx_invpolarity
	1	RW	rx_invpolarity
	2	RW	rx_bitreversal_enable
	3	RW	rx_bytereversal_enable
	4	RW	force_electrical_idle
0x12A4	0	R	rx_syncstatus
	1	R	rx_patterndetect
	2	R	rx_rlv
	3	R	rx_rmfifoatainserted
	4	R	rx_rmfiwodatadeleted
	5	R	rx_disperr
	6	R	rx_errdetect

Table 23. PCS Registers

Byte Offset	Bit	R/W	Name
0x1200		RW	Indirect_addr
0x1204	2	RW	RCLR_ERRBLK_CNT
	3	RW	RCLR_BER_COUNT
0x1208	1	RO	HI_BER
	2	RO	BLOCK_LOCK
	3	RO	TX_FULL
	4	RO	RX_FULL
	7	RO	Rx_DATA_READY

Table 24. Intel Arria 10 GMII PCS Registers

Byte Offset	Bit	R/W	Name
0x1240	9	RW	RESTART_AUTO_ NEGOTIATION
	12	RW	AUTO_NEGOTIATION_ ENABLE
	15	RW	Reset
0x1244	2	R	LINK_STATUS

continued...



Byte Offset	Bit	R/W	Name
	3	R	AUTO_NEGOTIATION_ ABILITY
	5	R	AUTO_NEGOTIATION_ COMPLETE
0x1250	5	RW	FD
	6	RW	HD
	8:7	RW	PS2, PS1
	13:12	RW	RF2, RF1
	14	RO	ACK
	15	RW	NP
0x1254	5	R	FD
	6	R	HD
	8:7	R	PS2, PS1
	13:12	R	RF2, RF1
	14	R	ACK
	15	R	NP
0x1258	0	R	LINK_PARTNER_AUTO_NEGOTIATION_ABLE
	1	R	PAGE_RECEIVE
0x1288	15:0	RW	AN link timer[15:0]
0x128C	4:0	RW	AN link timer[4:0]
0x1290	0	RW	SGMII_ENA
	1	RW	USE_SGMII_AN
	3:2	RW	SGMII_SPEED

Table 25. 10GBASE-KR Register Definitions

Byte Offset	Bit	R/W	Name
0x12C0	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
0x12C4	0	R	SEQ Link Ready
	1	R	SEQ AN timeout
	2	R	SEQ LT timeout

continued...



Byte Offset	Bit	R/W	Name
	13:8	RW	SEQ Reconfig Mode[5:0]
	16	R	KR FEC ability
	17	R	KR FEC err ind ability

1G/10G MAC

1G/10G MAC registers are applicable to both design examples.

The base address of the 1G/10G MAC registers are defined as follows:

Channel	1G/10G MAC 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 1/10G MAC register, refer to the [Low Latency Ethernet 10G MAC User Guide](#).

Note: The address offset in this table is in byte, while the register map table in the [Low Latency Ethernet 10G MAC User Guide](#) is in word.

Table 26. Primary MAC Address

Byte Offset	R/W	Name	HW Reset
0x2008	RW	primary_mac_addr0	0x0
0x200C	RW	primary_mac_addr1	0x0

Table 27. Transmit Configuration and Status Registers

Byte Offset	R/W	Name	HW Reset
0x4000	RW	tx_packet_control	0x0
0x4004	RO	tx_packet_status	0x0
0x4100	RW	tx_pad_control	0x1
0x4200	RW	tx_crc_control	0x3
<i>continued...</i>			



Byte Offset	R/W	Name	HW Reset
0x4400	RW	tx_preamble_control	0x0
0x6004	RW	tx_frame_maxlength	0x5EE(1518)
0x4300	RO	tx_underflow_counter0	0x0
0x4304	RO	tx_underflow_counter1	0x0

Table 28. Flow Control Registers

Byte Offset	R/W	Name	HW Reset
0x4500	RW	tx_pauseframe_control	0x0
0x4504	RW	tx_pauseframe_quanta	0x0
0x4508	RW	tx_pauseframe_enable	0x1
0x4680	RW	tx_pfc_priority_enable	0x0
0x4600	RW	pfc_pause_quanta_0	0x0
0x4604	RW	pfc_pause_quanta_1	0x0
0x4608	RW	pfc_pause_quanta_2	0x0
0x460C	RW	pfc_pause_quanta_3	0x0
0x4610	RW	pfc_pause_quanta_4	0x0
0x4614	RW	pfc_pause_quanta_5	0x0
0x4618	RW	pfc_pause_quanta_6	0x0
0x461C	RW	pfc_pause_quanta_7	0x0
0x4640	RW	pfc_holdoff_quanta_0	0x1
0x4644	RW	pfc_holdoff_quanta_1	0x1
0x4648	RW	pfc_holdoff_quanta_2	0x1
0x464C	RW	pfc_holdoff_quanta_3	0x1
0x4650	RW	pfc_holdoff_quanta_4	0x1
0x4654	RW	pfc_holdoff_quanta_5	0x1
0x4658	RW	pfc_holdoff_quanta_6	0x1
0x465C	RW	pfc_holdoff_quanta_7	0x1

Table 29. Receive Configuration and Status Registers

Byte Offset	R/W	Name	HW Reset
0x0000	RW	rx_transfer_control	0x0
0x0004	RO	rx_transfer_status	0x0
0x0100	RW	rx_padcrc_control	0x1
0x0200	RW	rx_crccheck_control	0x2
0x0400	RW	rx_custom_preamble_forward	0x0
<i>continued...</i>			



Byte Offset	R/W	Name	HW Reset
0x0500	RW	rx_preamble_control	0x0
0x2000	RW	rx_frame_control	0x3
0x2004	RW	rx_frame_maxlength	1518
0x2010	RW	rx_frame_spaddr0_0	0x0
0x2014	RW	rx_frame_spaddr0_1	0x0
0x2018	RW	rx_frame_spaddr1_0	0x0
0x201C	RW	rx_frame_spaddr1_1	0x0
0x2020	RW	rx_frame_spaddr2_0	0x0
0x2024	RW	rx_frame_spaddr2_1	0x0
0x2028	RW	rx_frame_spaddr3_0	0x0
0x202C	RW	rx_frame_spaddr3_1	0x0
0x2060	RW	rx_pfc_control	0x1
0x0300	RO	rx_pktovrfloerror	0x0

Table 30. Transmit Timestamp Registers

Byte Offset	R/W	Name	HW Reset
0x4440	RW	tx_period_10G	0x33333
0x4448	RW	tx_fns_adjustment_10G	0x0
0x444C	RW	tx_ns_adjustment_10G	0x0
0x4460	RW	tx_period_mult_speed	0x80000
0x4468	RW	tx_fns_adjustment_mult_speed	0x0
0x446C	RW	tx_ns_adjustment_mult_speed	0x0

Table 31. Receive Timestamp Registers

Byte Offset	R/W	Name	HW Reset
0x0440	RW	rx_period_10G	0x33333
0x0448	RW	rx_fns_adjustment_10G	0x0
0x044C	RW	rx_ns_adjustment_10G	0x0
0x0460	RW	rx_period_mult_speed	0x80000
0x0468	RW	rx_fns_adjustment_mult_speed	0x0
0x046C	RW	rx_ns_adjustment_mult_speed	0x0

Table 32. Transmit and Receive Statistics Registers

Byte Offset	R/W	Name	HW Reset
0x7000	RO	tx_stats_clr	0x0
0x3000	RO	rx_stats_clr	0x0
<i>continued...</i>			



Byte Offset	R/W	Name	HW Reset
0x7008:0x700C	RO	tx_stats_framesOK	0x0
0x3008:0x300C	RO	rx_stats_framesOK	0x0
0x7010:0x7014	RO	tx_stats_framesErr	0x0
0x3010:0x3014	RO	rx_stats_framesErr	0x0
0x7018:0x701C	RO	tx_stats_framesCRCErr	0x0
0x3018:0x301C	RO	rx_stats_framesCRCErr	0x0
0x7020:0x7024	RO	tx_stats_octetsOK	0x0
0x3020:0x3024	RO	rx_stats_octetsOK	0x0
0x7028:0x702C	RO	tx_stats_pauseMACCtrl_Frames	0x0
0x3028:0x302C	RO	rx_stats_pauseMACCtrl_Frames	0x0
0x7030:0x7034	RO	tx_stats_ifErrors	0x0
0x3030:0x3034	RO	rx_stats_ifErrors	0x0
0x7038:0x703C	RO	tx_stats_unicast_FramesOK	0x0
0x3038:0x303C	RO	rx_stats_unicast_FramesOK	0x0
0x7040:0x7044	RO	tx_stats_unicast_FramesErr	0x0
0x3040:0x3044	RO	rx_stats_unicast_FramesErr	0x0
0x7048:0x704C	RO	tx_stats_multicast_FramesOK	0x0
0x3048:0x304C	RO	rx_stats_multicast_FramesOK	0x0
0x7050:0x7054	RO	tx_stats_multicast_FramesErr	0x0
0x3050:0x3054	RO	rx_stats_multicast_FramesErr	0x0
0x7058:0x705C	RO	tx_stats_broadcast_FramesOK	0x0
0x3058:0x305C	RO	rx_stats_broadcast_FramesOK	0x0
0x7060:0x7064	RO	tx_stats_broadcast_FramesErr	0x0
0x3060:0x3064	RO	rx_stats_broadcast_FramesErr	0x0
0x7068:0x706C	RO	tx_stats_etherStatsOctets	0x0
0x3068:0x306C	RO	rx_stats_etherStatsOctets	0x0
0x7070:0x7074	RO	tx_stats_etherStatsPkts	0x0
0x3070:0x3074	RO	rx_stats_etherStatsPkts	0x0
0x7078:0x707C	RO	tx_stats_etherStatsUndersizePkts	0x0
0x3078:0x307C	RO	rx_stats_etherStatsUndersizePkts	0x0
0x7080:0x7084	RO	tx_stats_etherStatsOversizePkts	0x0
0x3080:0x3084	RO	rx_stats_etherStatsOversizePkts	0x0
0x7088:0x708C	RO	tx_stats_etherStatsPkts64Octets	0x0
0x3088:0x308C	RO	rx_stats_etherStatsPkts64Octets	0x0
			<i>continued...</i>



Byte Offset	R/W	Name	HW Reset
0x7090:0x7094	RO	tx_stats_etherStatsPkts65to127Octets	0x0
0x3090:0x3094	RO	rx_stats_etherStatsPkts65to127Octets	0x0
0x7098:0x709C	RO	tx_stats_etherStatsPkts128to255Octets	0x0
0x3098:0x309C	RO	rx_stats_etherStatsPkts128to255Octets	0x0
0x70A0:0x70A4	RO	tx_stats_etherStatsPkts256to511Octets	0x0
0x30A0:0x30A4	RO	rx_stats_etherStatsPkts256to511Octets	0x0
0x70A8:0x70AC	RO	tx_stats_etherStatsPkts512to1023Octets	0x0
0x30A8:0x30AC	RO	rx_stats_etherStatsPkts512to1023Octets	0x0
0x70B0:0x70B4	RO	tx_stats_etherStatPkts1024to1518Octets	0x0
0x30B0:0x30B4	RO	rx_stats_etherStatPkts1024to1518Octets	0x0
0x70B8:0x70BC	RO	tx_stats_etherStatsPkts1519toXOctets	0x0
0x30B8:0x30BC	RO	rx_stats_etherStatsPkts1519toXOctets	0x0
0x70C0:0x70C4	RO	tx_stats_etherStatsFragments	0x0
0x30C0:0x30C4	RO	rx_stats_etherStatsFragments	0x0
0x70C8:0x70CC	RO	tx_stats_etherStatsJabbers	0x0
0x30C8:0x30CC	RO	rx_stats_etherStatsJabbers	0x0
0x70D0:0x70D4	RO	tx_stats_etherStatsCRCErr	0x0
0x30D0:0x30D4	RO	rx_stats_etherStatsCRCErr	0x0
0x70D8:0x70DC	RO	tx_stats_unicastMACCtrlFrames	0x0
0x30D8:0x30DC	RO	rx_stats_unicastMACCtrlFrames	0x0
0x70E0:0x70E4	RO	tx_stats_multicastMACCtrlFrames	0x0
0x30E0:0x30E4	RO	rx_stats_multicastMACCtrlFrames	0x0
0x70E8:0x70EC	RO	tx_stats_broadcastMACCtrlFrames	0x0
0x30E8:0x30EC	RO	rx_stats_broadcastMACCtrlFrames	0x0
0x70F0:0x70F4	RO	tx_stats_PFCMACCtrlFrames	0x0
0x30F0:0x30F4	RO	rx_stats_PFCMACCtrlFrames	0x0



Document Revision History for Scalable Low Latency Ethernet 10G MAC using Intel Arria 10 1G/10G PHY

Table 33. Document Revision History

Date	Version	Changes
November 2017	2017.11.06	<ul style="list-style-type: none"> Rebranded as Intel. Added a note to the "Clocking Diagram" topic to clarify that the IOPLL input reference clock is sourcing from input clock through global clock network. Updated Figures: <ul style="list-style-type: none"> Block Diagram for Design Example without IEEE 1588v2 Block Diagram for Design Example with IEEE 1588v2 Updated "Packet Classifier Interface Signals" table: Updated the description for the <code>tx_egress_timestamp_request_in_valid[]</code>, <code>tx_egress_timestamp_request_in_fingerprint[][]</code>, and <code>tx_ingress_timestamp_format[]</code> signals. Updated "ToD Interface Signals" table: Updated the descriptions for all signals.
September 2015	2015.09.30	Added <code>DriftAdjust</code> and <code>DriftAdjustRate</code> registers to Master, 1G and 10G TOD register tables.
June 2015	2015.06.15	<ul style="list-style-type: none"> Updated supported ACDS, Modelsim and Synopsys versions. Updated Clocking scheme for design example with and without IEEE 1588v2 from <code>tx_pma_clkout</code> to <code>tx_clkout</code>. Removed timing violation notification during compilation. Added note about upgrading older design example versions is not supported. Added foot note Master TOD, 1G TOD and 10G TOD register maps about the default value for 'Period'. Updated the steps in 'Setting Up the Design Examples' by combining unzip and setting directory steps. Updated 'Channel 0 MAC RX Statistic Counter' by correcting number of packets from 28 to 12. Updated sampling clock for 10G and 1G TOD from 31.75MHz and 126.98MHz to 31.746031MHz and 126.984125MHz respectively. Updated 'Reset scheme at altera_eth_channel' and 'Reset scheme at altera_eth_channel_1588' figures from '<code>~phy_rx_block_lock ~phy_led_link</code>' to '<code>~phy_rx_block_lock & ~phy_led_link</code>'. Updated <code>tx_frame_maxlength</code> and <code>tx_frame_maxlength</code> hardware reset value to hexadecimal value.
January 2015	2015.01.22	Updated supported ACDS release for software simulation.
December 2014	2014.12.29	<ul style="list-style-type: none"> Updated supported ACDS release for simulation. Added tested ACDS release version for hardware simulation. Added timing violation notification during compilation. Updated clocking diagram for both design example with and without IEEE 1588v2. Replaced CMU PLL to fractional PLL for entire document.
May 2014	2014.05.29	Initial release.