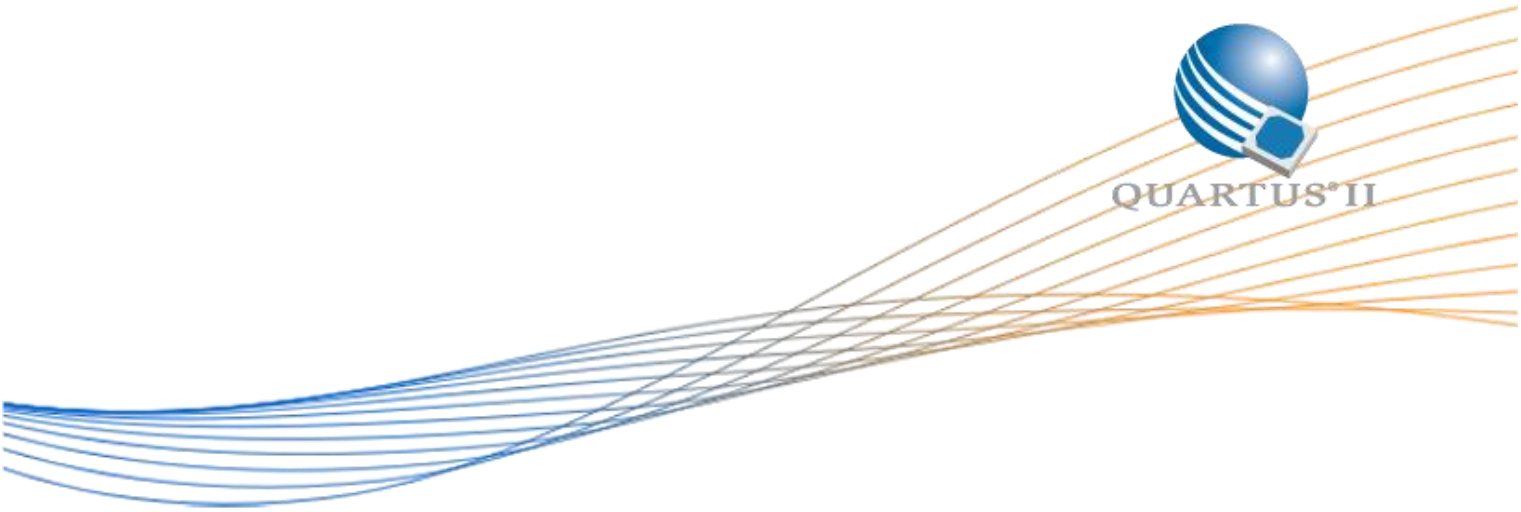


Arria 10 Triple Speed Ethernet and Native PHY Design Example User Guide

Date: May 2017

Revision: 1.0

Table of Contents

Introduction

This design example demonstrates the functionalities of the Intel® Arria 10 Triple-Speed Ethernet (TSE) and Intel® Arria 10 Transceiver Native PHY IP cores on Intel® Arria 10 SI development board.

This reference design offers the following features:

- Multi-channel design operating at 10/100/1000 Mbps.
- Scalable up to 10 Ethernet channels.
- Packet statistics for traffic generator, monitor, MAC transmitter (TX) and MAC receiver (RX).
- Throughput for the traffic received by the traffic monitor.
- Reporting of the MAC TX and RX statistics counters.
- System Console user interface. This TCL-based user interface allows you to dynamically configure and monitor any registers in the reference design.

System Architecture

Figure 1 shows the block diagrams on clocking and reset scheme for the design example.

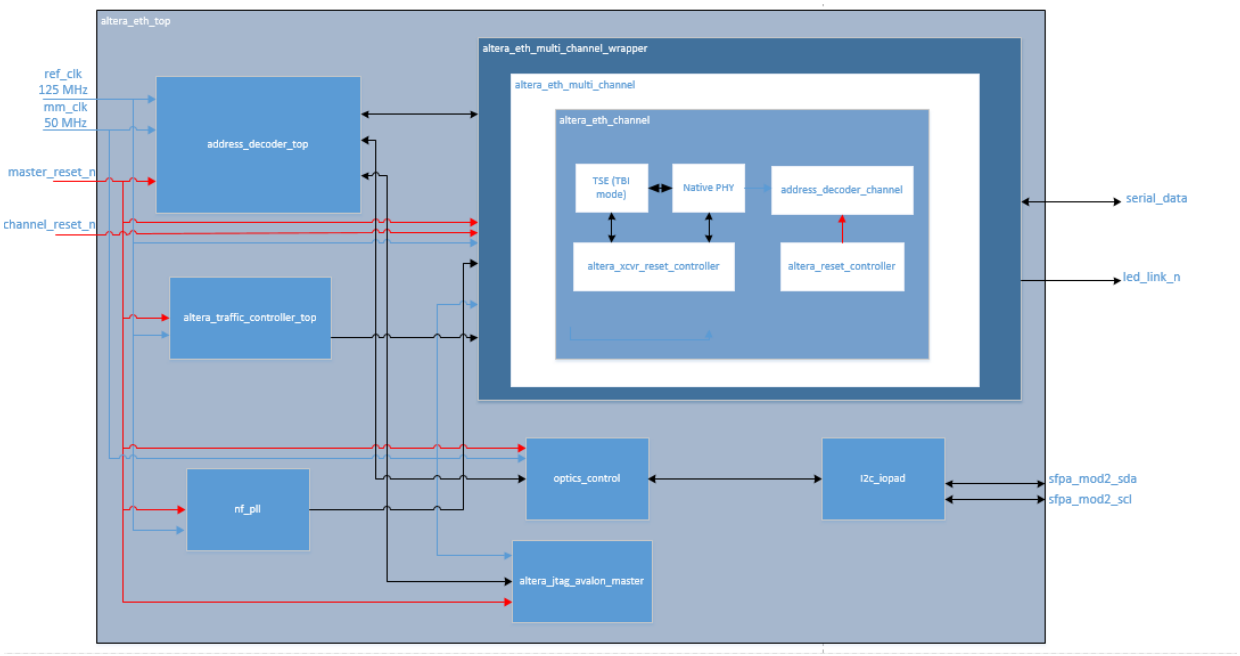


Figure 1: Block Diagram of Design Example

Design Components

Triple-Speed Ethernet IP

The Triple-Speed Ethernet IP core (TSE) consist of 10/100/1000-Mbps Ethernet MAC, 1000BASE-X/SGMII PCS. Ten-bit interface (TBI) is enabled in this design with PMA implemented with serial transceiver (Native PHY IP) in Intel FPGA support 1.25Gbps serial interface.

Transceiver Native PHY IP

The transceiver PHY used in this design is at basic (standard PCS) mode. The data rate of PHY is 1.25Gbps. For both PCS and PMA width are set as 10. PCFIFO configured as register mode, Byte SERDES and RMFIFO are disabled. 8B/10B Encoder and Decoder are both disabled in this PHY configuration.

Traffic Controller

The traffic controller consists of traffic generator and traffic monitor. The traffic generator injects client packet bursts into MAC TX and traffic monitor received packet bursts from MAC RX. This traffic controller connects to the Avalon single-clock FIFO in Ethernet subsystem through Avalon-ST interface.

Ethernet Packet Generator

This module consists of Avalon-MM registers, Ethernet packet generation block, CRC generator and shift register.

Ethernet Packet Monitor

This module will verify the payload of received packets and collect the statistic counter information. It consists of CRC checker and Avalon-MM registers.

JTAG to Avalon Master Bridge

This IP core provides a connection between the System Console and Qsys system through the physical interfaces. The System Console can initiate Avalon Memory-Mapped (Avalon-MM) transactions by sending encoded streams of bytes through the bridge's physical interfaces.

Reset Controller

This module is used to synchronize and generate signals as per design requirements.

I2C Controller

Monitor and control the SFP module.

PLL

Generated clock sources for the design.

System Register Map

Generator Register Map

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

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

Table 1: Generator Register Map

Monitor Register Map

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

Table 2: Monitor Register Map

Interface Signals

This section describes the signals available on the top level for the reference design.

Clock and Reset Signals

Signal	Direction	Width	Description
ref_clk	Input	1	125-MHz reference clock
mm_clk	Input	1	50-MHz clock for Avalon-MM interface
channel_reset_n[]	Input	NUM_CHANNELS	An asynchronous and active-low reset signal that resets individual Ethernet channels. This signal does not reset the components shared by all channels, such as the master TOD, master PPS, reconfiguration bundle, and fPLLs.
master_reset_n	Input	1	An asynchronous and active-low reset signal that resets the entire design.

Table 3: Clock and reset signals

Using the Design Example

This section describes the required hardware and software setup.

Hardware Requirements

To run this reference design, you need the following hardware available:

- Arria 10® GX SI Development Board
- USB-Blaster cable
- SFP+ module with loopback cable
- 2.4mm SMA cables.

Software Requirements

To run this reference design, you also require the following software:

- Quartus II® version 16.1
- Windows or Linux based system console
- USB-Blaster driver
- ModelSim® Simulator

Setting up the Arria 10® GX SI Development Kit

Figure 3 shows the Arria 10® GX SI development kit. The development kit has the hard-reset button for the Ethernet subsystem master reset and reset buttons for the channel resets. User Push button labeled as S1 on the board is the master reset. Push button from S2 to S8 will be assigned for each channel resets. (Description on the reset signals may refer to Table 3 above.)

Figure 2: Arria 10® GX Transceiver SI Development Kit\

Hardware Test on the Design Example

To perform hardware rest, follow these steps:

1. The design used 2 transceiver channels by default. (Pin assignment have been set to two channels, shown as below:
   PIN_K44 – TX channel 0 (p)
   PIN_K43 – TX channel 0 (n)
   PIN_R42 – RX channel 0 (p)
   PIN_R41 – RX channel 0 (n)
   PIN_J42 – TX channel 1 (p)
   PIN_J41 – TX channel 1 (n)
   PIN_P40 – RX channel 1 (p)
   PIN_P39 – RX channel 1 (n)

(User required to connect between TX channel 0 to RX channel 0 or TX channel 1 to RX channel 1 through SMA 2.5mm cables for **SMA loopback test**, depend on which channel user is using.)

2. After configuration done, open the **Clock Control** tool to change the frequency for **Y4** to **125 MHz**.

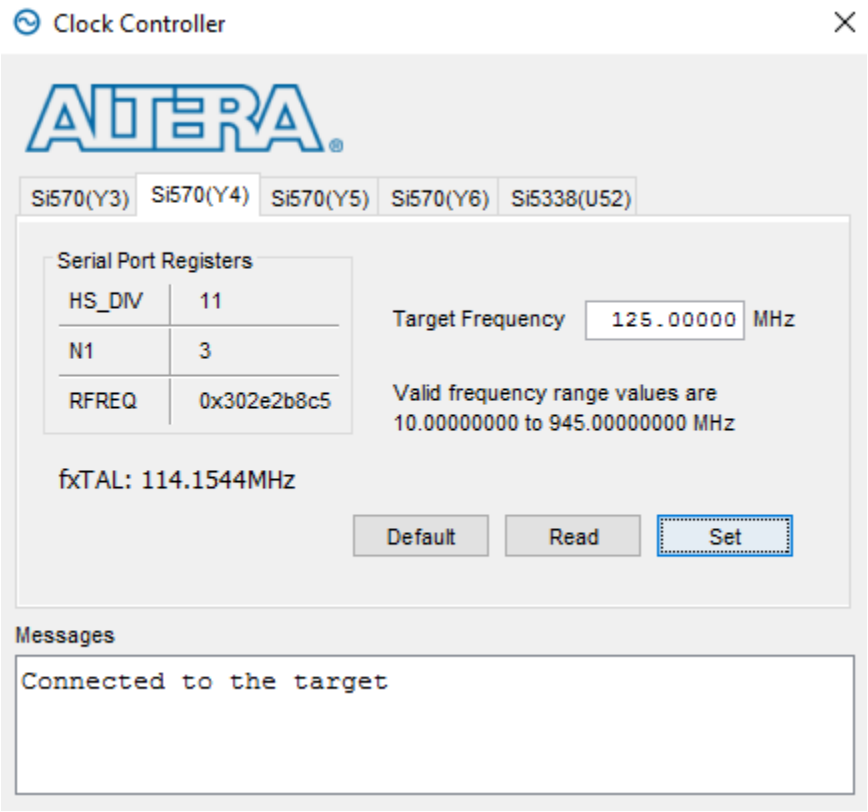


Figure 3: Clock Controller GUI

3. Download and restore the design example from Design Store.
4. Launch the Quartus II software and open the project file (top.qpf).
5. Click **Start Compilation** on the Processing menu to compile the design example.
6. Configure the FPGA using the generated configuration file (top.sof).

7. Reset the Ethernet system using the push button S1.

8. On Quartus II menu, click on **Tools>System Debugging Tools** and then launch the **System Console**.
9. In the System Console command shell, change the directory to "**system_console**" directory.
10. Run the command **source main.tcl**
11. Perform the following tests by running the command in the command shell.
    a. **To perform PHY internal serial loopback test**
        i. Format: TEST_PHYSERIAL_LOOPBACK <channel_number> <speed_test> <burst_size>
        ii. Example Command: TEST_PHYSERIAL_LOOPBACK 0 1G 10000
            - View the log to ensure that the traffic monitor does not receive bad packets. It also provides packet classification and statistics by the MAC TX and RX.
            - Perform **step 7** reset using push button after the test is completed.
    b. **To perform SMA loopback test**
        i. Format: TEST_SMA_LB <channel_number> <speed_test> <burst_size>
        ii. Example Command: TEST_SMA_LB 0 1G 10000
            - View the log to ensure that the traffic monitor does not receive bad packets. It also provides packet classification and statistics by the MAC TX and RX.
            - Perform **step 7** reset using push button after the test is completed.
            - This test only function able with connecting the TX and RX channels through the SMA cables. Refer to **step 1** for the connectivity details.
    c. **To perform MAC loopback test**
        i. Format: TEST_MAC_LOOPBACK <channel_number> <speed_test> <burst_size>
        ii. Example Command: TEST_MAC_LOOPBACK 0 1G 10000

- View the log to ensure that the traffic monitor does not receive bad packets. It also provides packet classification and statistics by the MAC TX and RX.
- Perform **step 7** reset using push button after the test is completed.

To perform hardware rest (on SFP+ loopback test), follow these steps:

12. Modification on top.qsf need to be perform to change the design to SFP+ mode.
13. Open the top.qsf file through any text editor software and make the following changes:
    a. **# Y5 (1G OSC) - For SFP+**
       set_location_assignment PIN_AC8 -to ref_clk

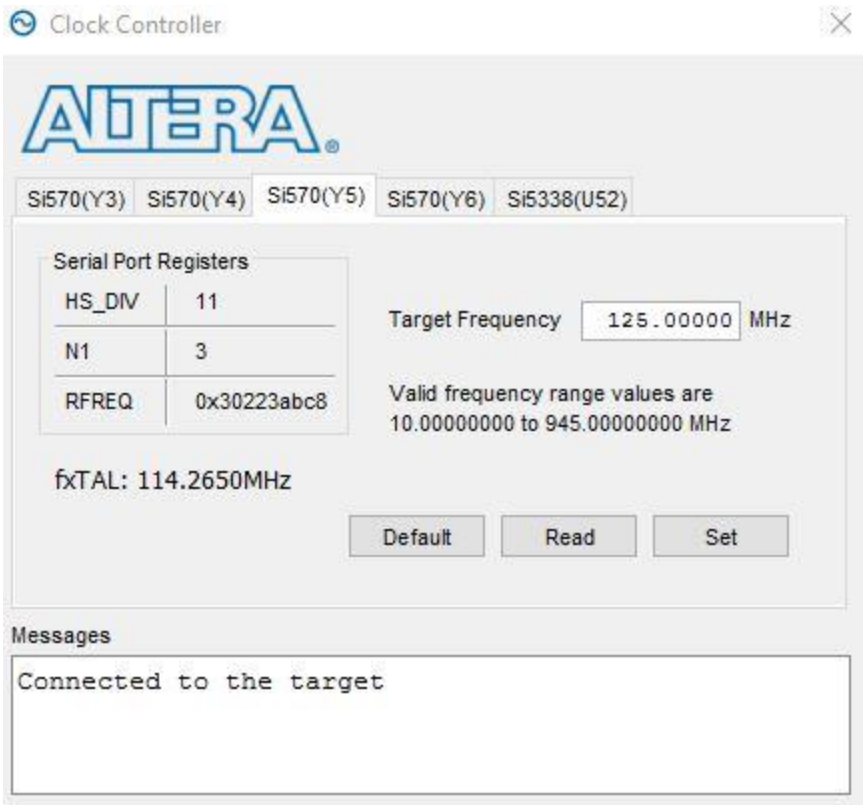
       # Y4 (1G OSC) - For Board Loopback
       # set_location_assignment PIN_U37 -to ref_clk

       # Board Loopback
       # set_location_assignment PIN_J42 -to tx_serial_data[1]
       # set_location_assignment PIN_P40 -to rx_serial_data[1]

       # set_location_assignment PIN_K44 -to tx_serial_data[0]
       # set_location_assignment PIN_R42 -to rx_serial_data[0]

       # SFP+
       set_location_assignment PIN_BC7 -to tx_serial_data[0]
       set_location_assignment PIN_AW7 -to rx_serial_data[0]

14. Save the top.qsf file after the modification and repeat step 4 -5.
15. After configuration done, open the **Clock Control** tool to change the frequency for **Y5** to **125 MHz**.



16. Next, follow the steps 7 until 11 for all the hardware test. (Note: For **SMA loopback test**, it only support for one channel (channel 0) as only one SFP+ slot is available for the development kit)

Simulation Test on the Design Example

To run simulation on this reference design, follow these steps:

1. Download and restore the design example from Design Store.
2. Start the ModelSim® SE 10.4b simulator software.
3. Go to the "**testbench/Modelsim/testcase<x>**" directory. We have two different test cases available in this design.
4. testcase1 is for single channel design and testcase2 will simulate 2 channels loopback testing.
5. In the TCL Console window, type the following commands:
    a. Command: do tb_run.tcl

6. At the end of the simulation, ModelSim simulator will generate statistics of transmitted packets and received packets in the Transcript window.

```
# ------------------------
# Channel         1: MAC Statistics
# ------------------------
#       aFramesTransmittedOK              = 6
#       aFramesReceivedOK                 = 6
#       aFrameCheckSequenceErrors         = 0
#       aAlignmentErrors                  = 0
#       aOctetsTransmittedOK              = 500
#       aOctetsReceivedOK                 = 500
#       aTxPAUSEMACCtrlFrames             = 0
#       aRxPAUSEMACCtrlFrames             = 0
#       ifInErrors                        = 0
#       ifOutErrors                       = 0
#       ifInUcastPkts                     = 0
#       ifInMulticastPkts                 = 6
#       ifInBroadcastPkts                 = 0
#       ifOutDiscards                     = 0
#       ifOutUcastPkts                    = 0
#       ifOutMulticastPkts                = 6
#       ifOutBroadcastPkts                = 0
#       etherStatsDropEvents              = 0
#       etherStatsOctets                  = 608
#       etherStatsPkts                    = 6
#       etherStatsUndersizePkts           = 0
#       etherStatsOversizePkts            = 0
#       etherStatsPkts64Octets            = 0
#       etherStatsPkts65to127Octets       = 4
#       etherStatsPkts128to255Octets      = 2
#       etherStatsPkts256to511Octet       = 0
#       etherStatsPkts512to1023Octets     = 0
#       etherStatsPkts1024to1518Octets    = 0
#       etherStatsPkts1519OtoXOctets      = 0
#       etherStatsFragments               = 0
#       etherStatsJabbers                 = 0
#
#
# Simulation PASSED
#
# ** Note: $finish    : tb_testcase_1588.sv(436)
#    Time: 77655 ns  Iteration: 1  Instance: /tb_top/TESTCASE
```

Figure 4: Simulation Results

Document Revision History

Date	Version	Changes
May 2017	1.0	Initial release