Low Latency 40G Ethernet Example Design User Guide

UG-20025
2017.11.08

101 Innovation Drive
San Jose, CA 95134
www.altera.com

Contents

Quick Start Guide..... 1-1
Directory Structure..... 1-2
Design Components..... 1-3
Generating the Design..... 1-3
Simulating the Design..... 1-5
Compiling and Testing the Example Design in Hardware..... 1-6

Example Design Description..... 2-1
Features..... 2-1
Hardware and Software Requirements..... 2-1
Functional Description..... 2-2
Design Variations..... 2-3
Standard IP Core Variation..... 2-3
40GBASE-KR4 IP Core Variation..... 2-4
Interface Signals..... 2-4
Example Design Registers..... 2-4

Document Revision History..... A-1

Quick Start Guide 1

2017.11.08

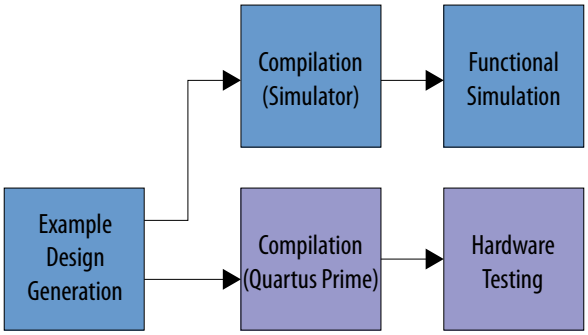
UG-20025

Subscribe Send Feedback

The Arria 10 variations of the LL 40GbE IP core feature a simulatable testbench and a hardware example design that supports compilation and hardware testing, to help you understand usage. When you generate the example design, the parameter editor automatically creates the files necessary to simulate, compile, and test the design in hardware. You can download the compiled hardware design to the Arria 10 GX Transceiver Signal Integrity Development Kit. The testbench and demonstration example design are available for a wide range of parameters. However, they do not cover all possible parameterizations of the LL 40GbE IP Core.

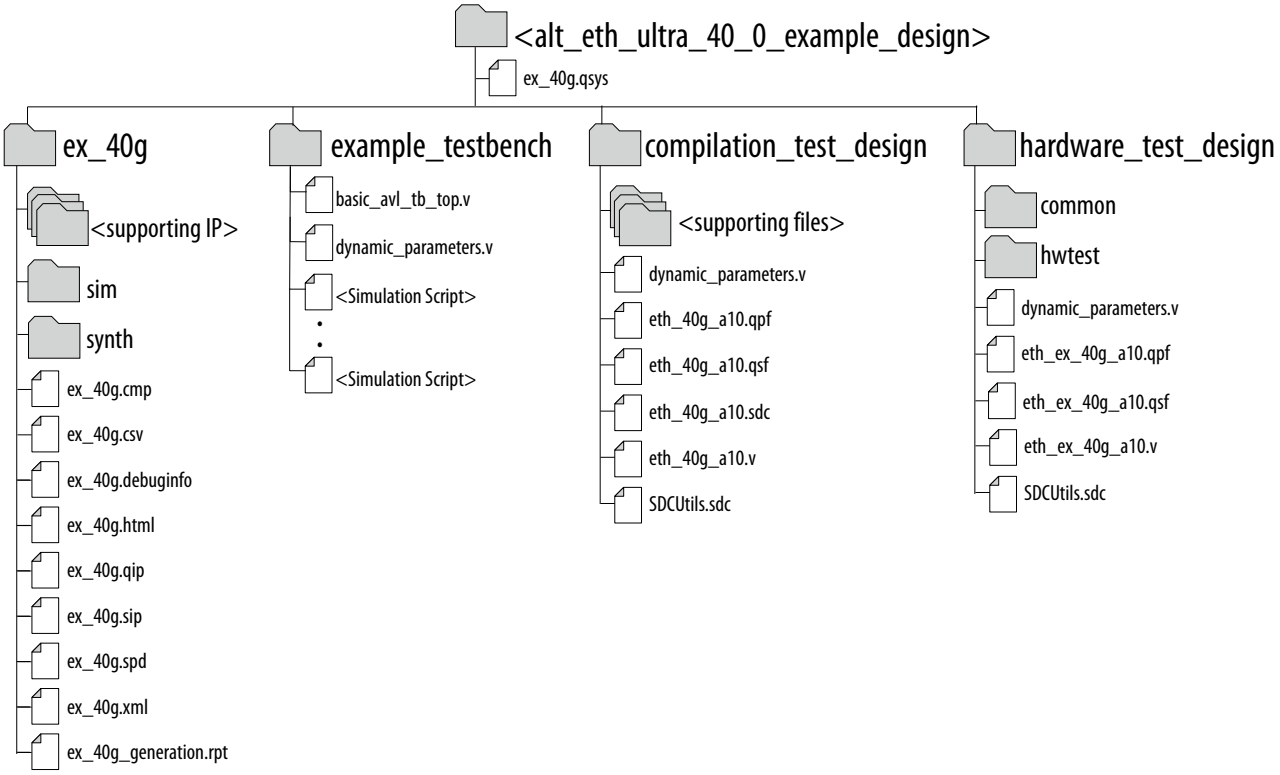
In addition, for most IP core variations, Altera provides a compilation-only example project that you can use to quickly estimate IP core area and timing.

Figure 1-1: Development Steps for the Example Design



Directory Structure

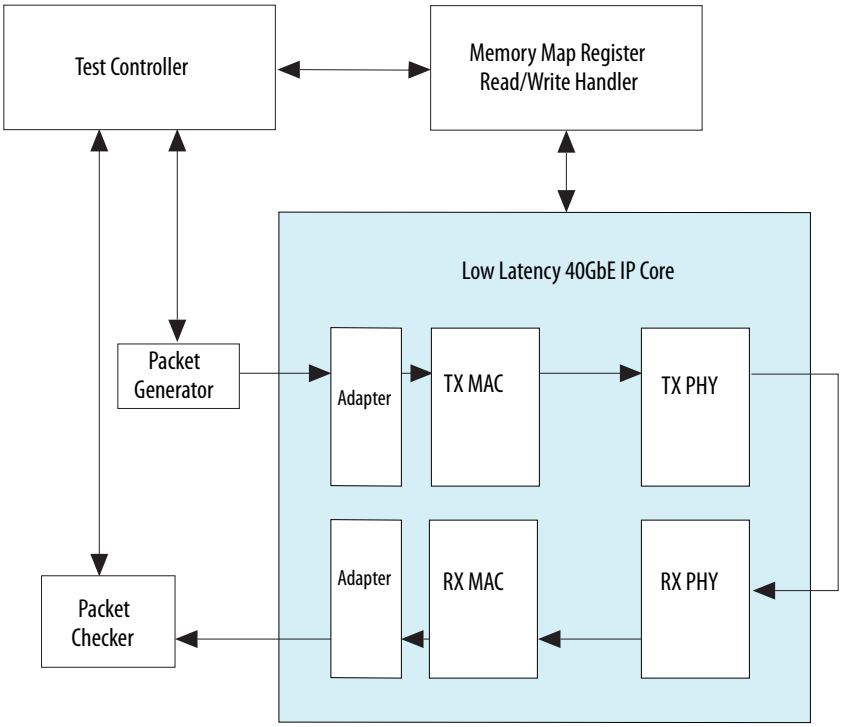
Figure 1-2: Directory Structure for the Generated Example Design



The hardware comfiguration and test files (the hardware design example) are located in `<example_design_install_dir>/hardware_test_design`. The simulation files (testbench for simulation only) are located in `<example_design_install_dir>/example_testbench`. The compilation-only example design is located in `<example_design_install_dir>/compilation_test_design`.

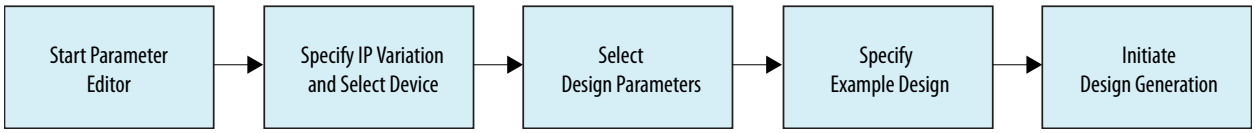
Design Components

Figure 1-3: Block Diagram



Generating the Design

Figure 1-4: Procedure



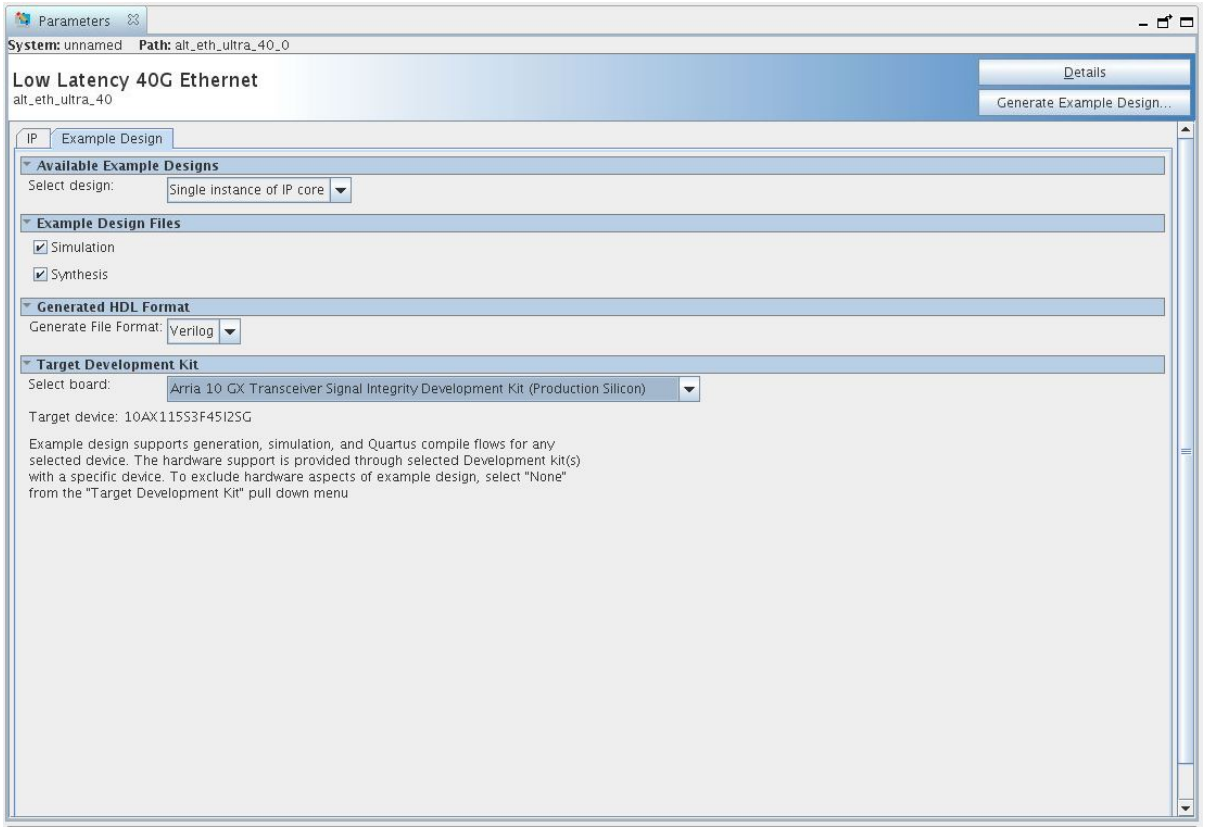
Follow these steps to generate the Arria 10 hardware example design and testbench::

1. In the IP Catalog (`Tools > IP Catalog`), select the Arria 10 target device family.

   Note: The Quick Start hardware design example is only supported in Arria 10 devices. The testbench is available for variations that target Arria 10 devices or Stratix V devices.
2. In the IP Catalog, locate and select **Low Latency 40G Ethernet**. The New IP Variation window appears.
3. Specify a top-level name for your custom IP variation. The parameter editor saves the IP variation settings in a file named .
4. You must select a specific Arria 10 device in the **Device** field, or keep the default Quartus Prime software device selection.

5. Click **OK**. The parameter editor appears.

Figure 1-5: Example Design Tab in LL 40GbE Parameter Editor



6. On the **IP** tab, specify the parameters for your IP core variation.

   Note: The LL 40GbE example design is not available for following selections:

   - Use external MAC PLL
   - Custom streaming client interface

7. On the **Example Design** tab, select the **Simulation** option to generate the testbench, and select the **Synthesis** option to generate the hardware example design.

   Note: At least one of the Simulation and Synthesis check boxes from Example Design Files must be selected to allow generation of Example Design Files.

8. For **Generated HDL Format**, only Verilog is available.
9. For **Target Development Kit** select the **Arria 10 GX Transceiver Signal Integrity Development Kit**. The hardware example design overwrites the selection (in **step 4**) with the device on the target board.
10. Click the **Generate Example Design** button.
11. Refer to the KDB Answer *How do I compensate for the jitter of PLL cascading or non-dedicated clock path for Arria 10 PLL reference clock?* for a workaround you should apply in the `hardware_test_design` directory in the `.sdc` file.

    Note: You must consult this KDB Answer because the RX path in the IP core includes cascaded PLLs. Therefore, the IP core clocks might experience additional jitter in Arria 10 devices. This KDB Answer clarifies the software releases in which the workaround is necessary.

Related Information

- **IP Core Parameters**
  Provides more information about IP core customization by specifying parameters.
- **Arria 10 GX Transceiver Signal Integrity Development Kit Webpage**
- **KDB Answer: How do I compensate for the jitter of PLL cascading or non-dedicated clock path for Arria 10 PLL reference clock?**

Simulating the Design

Figure 1-6: Procedure

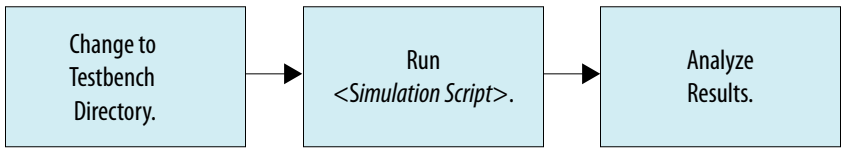


Table 1-1: Low Latency 40GbE IP Core Testbench File Descriptions

Lists the key files that implement the example testbenches.

File Names	Description
Testbench and Simulation Files	
`basic_avl_tb_top.v`	Top-level testbench file for non-40GBASE-KR4 variations. The testbench instantiates the DUT and runs Verilog HDL tasks to generate and accept packets.
`alt_e40_avalon_kr4_tb.sv`	Top-level testbench file for 40GBASE-KR4 variations.
`alt_e40_avalon_tb_packet_gen.v`, `alt_e40_avalon_tb_packet_gen_sanity_check.v`, `alt_e40_stat_cntr_1port.v`	Packet generator and checkers for 40GBASE-KR4 variations.
Testbench Scripts	
`run_vsim.do`	The ModelSim script to run the testbench.
`run_vcs.sh`	The Synopsys VCS script to run the testbench.
`run_ncsim.sh`	The Cadence NCSim script to run the testbench.

Follow these steps to simulate the testbench:

1. Change to the testbench simulation directory `<example_design_install_dir>/example_testbench`.
2. Run the simulation script for the supported simulator of your choice. The script compiles and runs the testbench in the simulator. Refer to the table "Steps to Simulate the Testbench".
3. Analyze the results. The successful testbench sends ten packets, receives ten packets, and displays `"Testbench complete."`

Table 1-2: Steps to Simulate the Testbench

Simulator	Instructions
Modelsim	In the command line, type `vsim -c -do run_vsim.do` **Note:** The ModelSim-AE simulator does not have the capacity to simulate this IP core. You must use another supported ModelSim simulator.
NCSim	In the command line, type `sh run_ncsim.sh`
VCS	In the command line, type `sh run_vcs.sh`

Compiling and Testing the Example Design in Hardware

To compile and run a demonstration test on the hardware design example, follow these steps:

1. Ensure hardware design example generation is complete.
2. In the Quartus® Prime software, open the Quartus Prime project `<example_design_install_dir>/hardware_test_design/eth_ex_40g_a10.qpf`, as appropriate.
3. Before compiling, ensure you have implemented the workaround from the KDB Answer *How do I compensate for the jitter of PLL cascading or non-dedicated clock path for Arria 10 PLL reference clock?* if relevant for your software release.
4. On the Processing menu, click **Start Compilation**.
5. After successful compilation, a **.sof** file will be generated in your specified directory. Follow these steps to program the hardware design example on the Arria 10 device:
   a. On the **Tools** menu, click **Programmer**.
   b. In the Programmer, click **Hardware Setup**.
   c. Select a programming device.
   d. Select and add the Arria 10 GX Transceiver Signal Integrity Development Kit to which your Quartus Prime session can connect.
   e. Ensure that **Mode** is set to **JTAG**.
   f. Select the Arria 10 device and click **Add Device**. The Programmer displays a block diagram of the connections between the devices on your board.
   g. In the row with your `.sof`, check the box for the `.sof`.
   h. Check the box in the **Program/Configure** column.
   i. Click **Start**.
   j. After the hardware example design is configured on the Arria 10 device, in the Quartus Prime software, on the **Tools** menu, click **System Debugging Tools** > **System Console**.

6. In the Tcl Console pane, type `cd hwtest` to change directory to `<example_design_install_dir>/hardware_test_design/hwtest`.
7. Type `source main.tcl`.
8. Type `run_test`.

The successful test run displays output confirming the following behavior:

1. Turning off packet generation
2. Enabling loopback
3. Waiting for RX clock to settle
4. Printing PHY status
5. Clearing MAC statistics counters
6. Sending packets
7. Reading MAC statistics counters
8. Printing MAC statistics counters, which show 0 in all error counters

The following sample output illustrates a successful test run:

```
--- Turning off packet generation ----
--------------------------------------

--------- Enabling loopback ----------
--------------------------------------

--- Wait for RX clock to settle... ---
--------------------------------------

-------- Printing PHY status ---------
--------------------------------------

 RX PHY Register Access: Checking Clock Frequencies (KHz)

          REFCLK                :644530 (KHZ)
          TXCLK                 :390624   (KHZ)
          RXCLK                 :390625   (KHZ)
          RX RECOV CLK          :322265   (KHZ)
          TX-IO CLOCK           :322265   (KHZ)
 RX PHY Status Polling

 Tx PLL Lock Status             0x000003ff

 Rx Frequency Lock Status       0x000003ff

 Mac Clock in OK Condition?     0x00000007

 Rx Frame Error                 0x00000000

 Rx PHY Fullly Aligned?         0x00000001

---- Clearing MAC stats counters -----
--------------------------------------

--------- Sending packets... ---------
--------------------------------------

----- Reading MAC stats counters -----
--------------------------------------


 ======================================================================
```

```
                        STATISTICS FOR BASE 0x0900 (Rx)
 ============================================================================
Fragmented Frames                : 0
Jabbered Frames                  : 0
Any Size with FCS Err Frame      : 0
Right Size with FCS Err Fra      : 0
Multicast data Err Frames        : 0
Broadcast data Err Frames        : 0
Unicast data Err  Frames         : 0
Multicast control  Err Frame     : 0
Broadcast control Err  Frame     : 0
Unicast control Err Frames       : 0
Pause control Err Frames         : 0
64 Byte Frames                   : 0
65 - 127 Byte Frames             : 6894742
128 - 255 Byte Frames            : 9147409
256 - 511 Byte Frames            : 8089346
512 - 1023 Byte Frames           : 3411180
1024 - 1518 Byte Frames          : 347630
1519 - MAX Byte Frames           : 40042
> MAX Byte Frames                : 0
Rx Frame Starts                  : 27930349
Multicast data OK  Frame         : 0
Broadcast data OK  Frame         : 0
Unicast data OK  Frames          : 27929934
Multicast Control Frames         : 0
Broadcast Control Frames         : 0
Unicast Control Frames           : 415
Pause Control Frames             : 0
 ============================================================================
                        STATISTICS FOR BASE 0x0800 (Tx)
 ============================================================================
Fragmented Frames                : 0
Jabbered Frames                  : 0
Any Size with FCS Err Frame      : 0
Right Size with FCS Err Fra      : 0
Multicast data Err Frames        : 0
Broadcast data Err Frames        : 0
Unicast data Err Frames          : 0
Multicast control  Err Frame     : 0
Broadcast control Err  Frame     : 0
Unicast control Err Frames       : 0
Pause control Err Frames         : 0
64 Byte Frames                   : 0
65 - 127 Byte Frames             : 6894742
128 - 255 Byte Frames            : 9147409
256 - 511 Byte Frames            : 8089346
512 - 1023 Byte Frames           : 3411180
1024 - 1518 Byte Frames          : 347630
1519 - MAX Byte Frames           : 40042
> MAX Byte Frames                : 0
Tx Frame Starts                  : 27930349
Multicast data OK  Frame         : 0
Broadcast data OK  Frame         : 0
Unicast data OK  Frames          : 27929934
Multicast Control Frames         : 0
Broadcast Control Frames         : 0
Unicast Control Frames           : 415
Pause Control Frames             : 0
---------------- Done ----------------
```

Related Information

- **KDB Answer: How do I compensate for the jitter of PLL cascading or non-dedicated clock path for Arria 10 PLL reference clock?**

- **Incremental Compilation for Hierarchical and Team-Based Design**
- **Programming Altera Devices**
- **Analyzing and Debugging Designs with System Console**

Example Design Description 2

2017.11.08

UG-20025

The example design demonstrate the functionalities of the Low Latency 40G Ethernet IP core. You can generate the design from the **Example Design** tab of the LL 40G Ethernet graphical user interface (GUI) in the IP parameter editor.

Features

- Standard XLAUI or CAUI external interface consisting of FPGA hard serial transceiver lanes operating at 10.3125 Gbps , or the CAUI-4 external interface consisting of four FPGA hard serial transceiver lanes operating at 25.78125 Gbps.
- Supports 40GBASE-KR4 PHY based on 64B/66B encoding with data striping and alignment markers to align data from multiple lanes.
- Avalon Memory-Mapped (Avalon-MM) management interface to access the IP core control and status registers.
- Avalon-ST data path interface connects to client logic with the start of frame in the most significant byte (MSB) when optional adapters are used. Interface has data width 256 or 512 bits depending on the data rate.
- RX CRC checking and error reporting.
- TX error insertion capability supports test and debug.
- Hardware and software reset control.

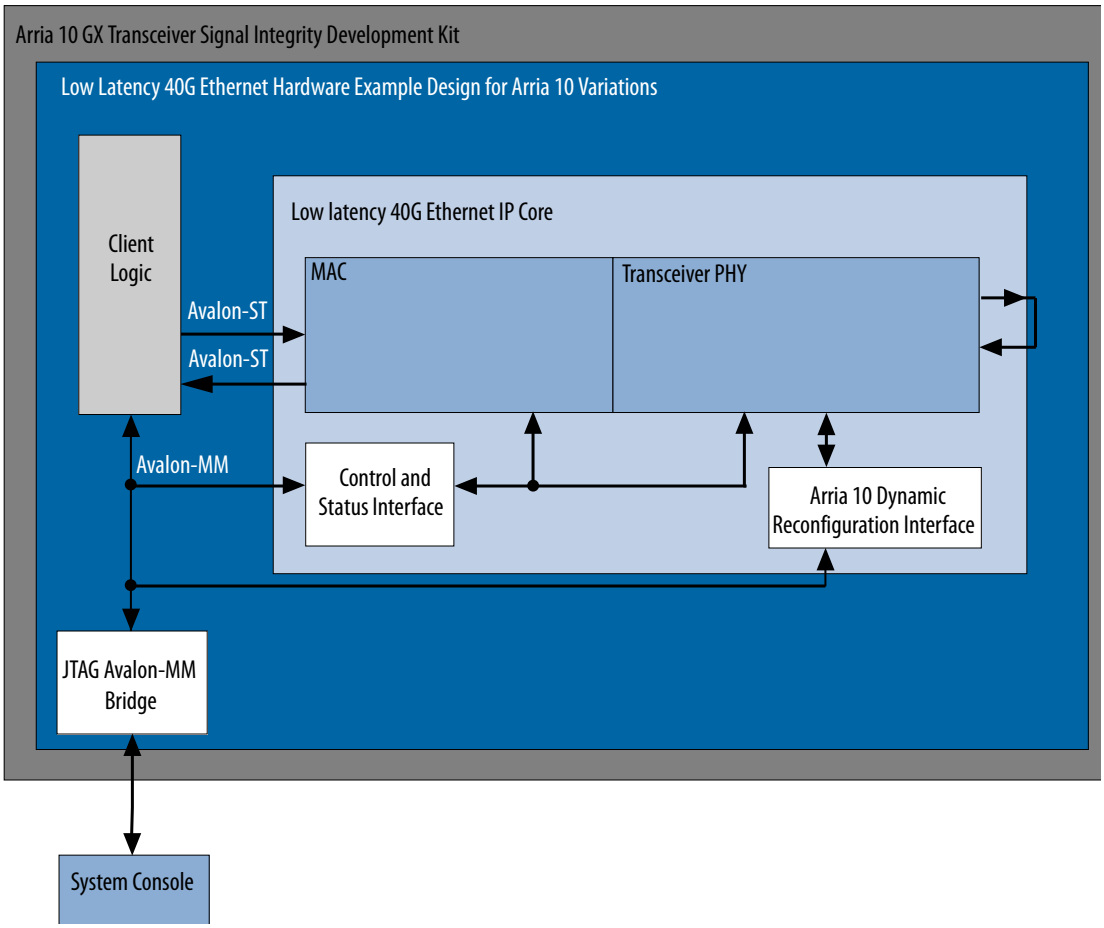
Hardware and Software Requirements

Altera uses the following hardware and software to test the example design:

- Quartus Prime software
- System Console
- ModelSim-AE, Modelsim-SE, NCsim (Verilog only), or VCS simulator
- Arria 10 GX Transceiver Signal Integrity Development Kit for hardware testing

Functional Description

Figure 2-1: High Level Block Diagram for the LL 40GbE Hardware Design Example



The Arria 10 Low Latency 40GbE hardware design example includes the following components:

- Low Latency 40GbE IP core with Avalon-ST user interfaces. The IP core does not support a hardware example design for variations with custom streaming user interfaces.
- Client logic that coordinates the programming of the IP core, and packet generation and checking.
- JTAG controller that communicates with the Altera System Console. You communicate with the client logic through the System Console.

Table 2-1: Low Latency GbE IP Core Hardware Design Example File Descriptions

File Names	Description
`eth_ex_40g_a10.qpf`	Quartus Prime project file
`eth_ex_40g_a10.qsf`	Quartus project settings file
`eth_ex_40g_a10.v`	Top-level Verilog HDL design example file

File Names	Description
`common/`	Hardware design example support files
Scripts	
`hwtest/`	System Console testing scripts
`hwtest/main.tcl`	Main file for accessing System Console

Related Information

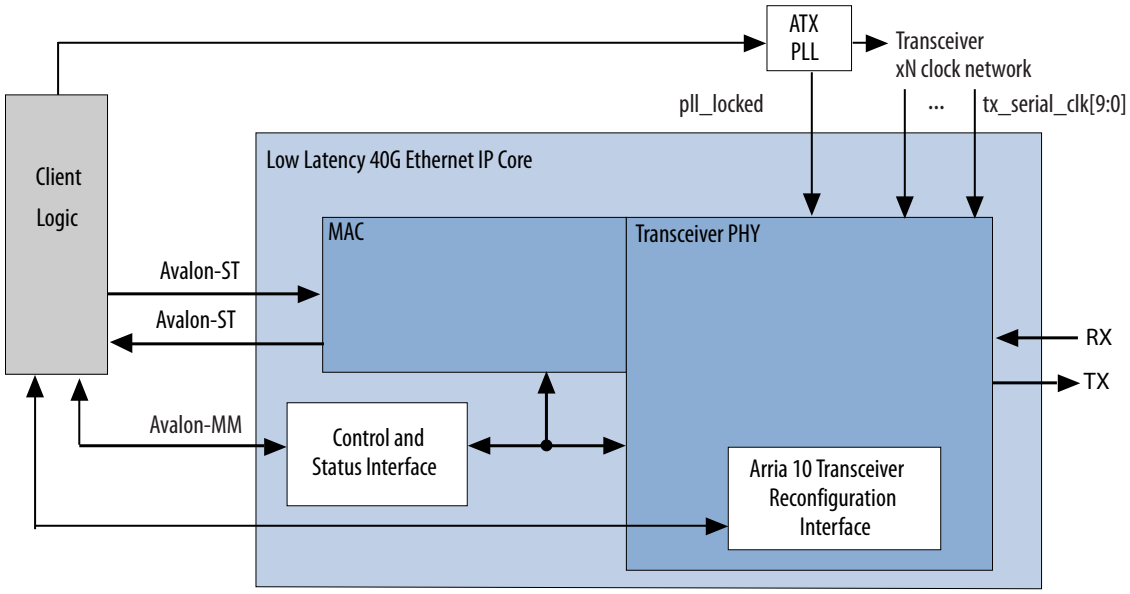
Arria 10 GX Transceiver Signal Integrity Development Kit Webpage

Design Variations

Standard IP Core Variation

The example design for standard 40GbE IP core variations that target an Arria 10 device configure a single ATX PLL and connect it to the xN clock network, which distributes the output `tx_serial_clk` signal to all four or ten individual transceiver channels. If this arrangement is not available for your design, you can use multiple external ATX and CMU PLLs to generate and distribute the `tx_serial_clk` signals for the individual channels. It also includes client logic to exercise the IP core. The client logic includes logic to ensure each packet is sent to the Avalon-ST interface without any intermediate idle cycles, in other words, that the data sent to this interface complies with the IP core requirements.

Figure 2-2: IP Core Variation Testbench

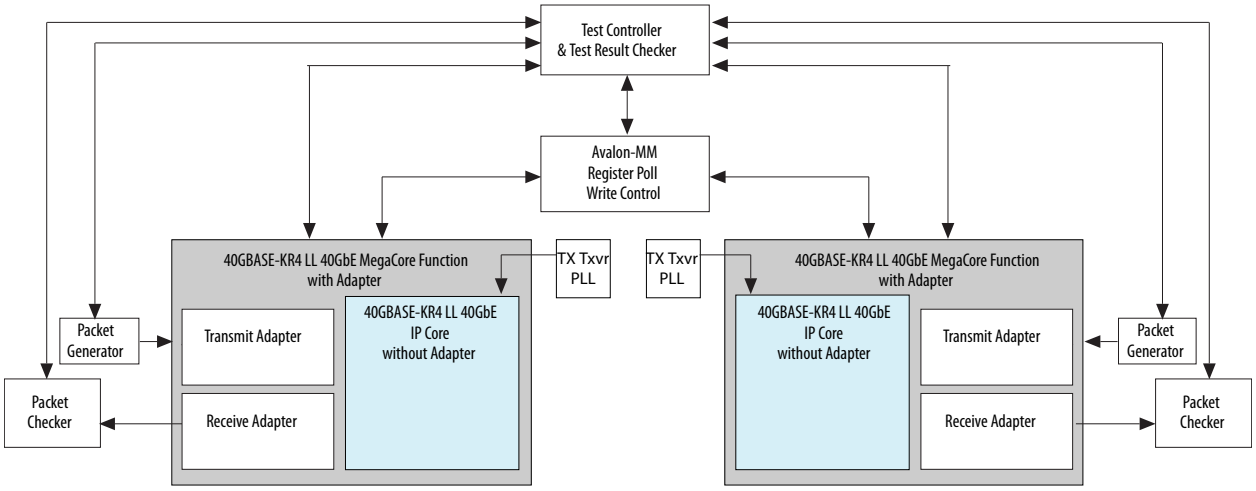


The simulation testbench instantiates the IP core and necessary PLLs. It interfaces directly with the Avalon-ST port to provide basic packet sending and receiving. The TX and RX lanes will be connected together to provide loopback testing capabilities.

40GBASE-KR4 IP Core Variation

Illustrates the top-level modules of the LL 40GBASE-KR4 IP core variation testbench. To support the simulation of auto-negotiation, the testbench uses two instances of the IP core instead of configuring the IP core in loopback mode.

Figure 2-3: 40GBASE-KR4 LL 40G Ethernet IP Core Testbench



Interface Signals

Table 2-2: Example Design Interface Signals

Signal	Direction	Interface
clk_ref	Input	Clocks
reset_async	Input	Reset
tx_serial[3:0] / tx_serial[9:0]	Output	Transceiver PHY serial data interface

Related Information

IP Core Signals
Provides detailed description of the signals and the interfaces to which they belong.

Example Design Registers

Table 2-3: Packet Client Registers

You can customize the LL 40GbE hardware design example by programming the packet client registers.

Addr	Name	Bit	Description	HW Reset Value	Access
0x1000	PKT_CL_SCRATCH	[31:0]	Scratch register available for testing.		RW
0x1001	PKT_CL_CLNT	[31:0]	Four characters of IP block identification string "CLNT"		RO
0x1002	PKT_CL_FEATURE	[9:0]	Feature vector to match DUT. Bits [8:3] have the value of 0 to indicate the DUT does not have the property or the value of 1 to indicate the DUT has the property. • Bit [0]: Has the value of 1 to indicate the DUT targets an Arria 10 device. • Bit [1]: Has the value of 0 if the DUT is a LL 40GbE IP core; has the value of 1 if the DUT is a LL 100GbE IP core. • Bit [2]: Reference clock frequency. Has the value 0 for 322 MHz; has the value of 1 for 644 MHz. • Bit [3]: Indicates whether the DUT is a LL 40-GBASE KR4 IP core. • Bit [4]: Inidcates whether the DUT is a CAUI-4 IP core. • Bit [5]: Indicates whether the DUT includes PTP support. • Bit [6]: Indicates whether the DUT includes pause support • Bit [7]: Indicates whether the DUT provides local fault signaling. • Bit [8]: Indicates whether the DUT has **Use external MAC TX PLL** turned on. Must have the value of 0. • Bit [9]: Value 0 if the DUT has a custom streaming client interface; value 1 if the DUT has an Avalon-ST client interface. Must have the value of 1.		RO
0x1006	PKT_CL_TSD	[7:0]	Arria 10 device temperature sensor diode readout in Fahrenheit.		RO
0x1010	PKT_GEN_TX_CTRL	[3:0]	• Bit [0]: Reserved. • Bit [1]: Packet generator disable bit. set this bit to the value of 1 to turn off the packet generator, and reset it to the value of 0 to turn on the packet generator. • Bit [2]: Reserved. • Bit [3]: Has the value of 1 if the IP core is in MAC loopback mode; has the value of 0 if the packet client uses the packet generator.	4'b0101	RW

Addr	Name	Bit	Description	HW Reset Value	Access
0x1015	PKT_CL_LOOPBACK_FIFO_ERR_CLR	[2:0]	Reports MAC loopback errors. • Bit [0]: FIFO underflow. Has the value of 1 if the FIFO has underflowed. This bit is sticky. Has the value of 0 if the FIFO has not underflowed. • Bit [1]: FIFO overflow. Has the value of if the FIFO has overflowed. This bit is sticky. Has the value of 0 if the FIFO has not overflowed. • Bit [2]: Assert this bit to clear bits [0] and [1].	3'b0	RO
0x1016	PKT_CL_LOOPBACK_RESET	[0]	MAC loopback reset. Set to the value of 1 to reset the design example MAC loopback.	1'b0	RW

Related Information

IP Core Registers

Provides description of the registers included in the Low Latency 40GbE IP core.

Document Revision History

A

2017.11.08

UG-20025 Subscribe Send Feedback

Table A-1: Revision History

Date	Changes
2017.11.08	Added link to KDB Answer that provides workaround for potential jitter on Arria® 10 devices due to cascading ATX PLLs in the IP core. Refer to **Generating the Design** on page 1-3 and **Compiling and Testing the Example Design in Hardware** on page 1-6. **Note:** This design example user guide has not been updated to reflect minor changes in design generation in Quartus Prime releases later than the Quartus Prime software release v16.0.
2016.05.02	Initial release