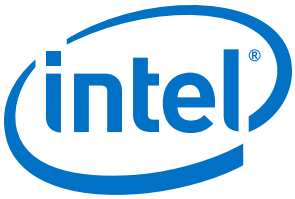


Low Latency 100G Ethernet Intel® Stratix® 10 FPGA IP Design Example User Guide

Updated for Intel® Quartus® Prime Design Suite: **20.1**

IP Version: **19.2.0**

UG-20086 | 2020.04.13
Latest document on the web: **PDF** | **HTML**

Subscribe
Send Feedback

Contents

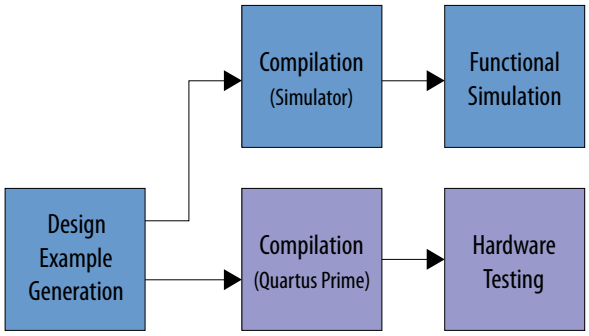
1. Quick Start Guide

The IP core provides a design example which allows the user to:

- Compile the design — to get an estimate IP core area and timing
- Simulate the design — to verify the IP core functionality through simulation
- Test the design on hardware — to test the design on the Intel® Stratix® 10 GX Transceiver Signal Integrity Development Kit

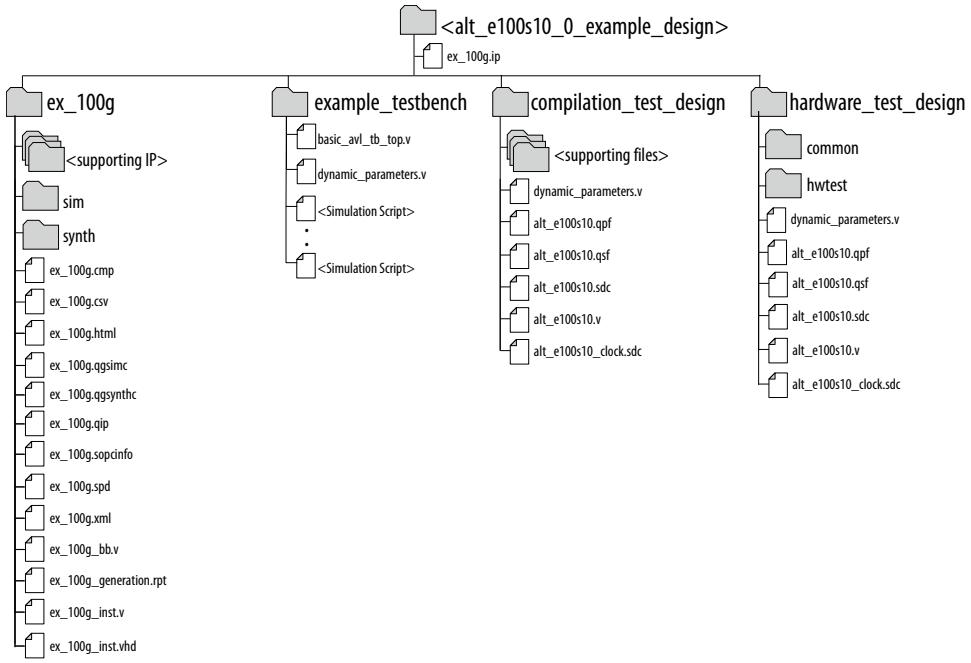
When you generate the design example, the parameter editor automatically creates the files necessary to simulate, compile, and test the design in hardware.

Figure 1. Development Steps for the Design Example



1.1. Directory Structure

Figure 2. Low Latency 100G Ethernet Intel Stratix 10 FPGA Design Example Directory Structure



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

1.2. Simulation Design Example Components

Figure 3. Low Latency 100G Ethernet Intel Stratix 10 FPGA Simulation Design Example Block Diagram

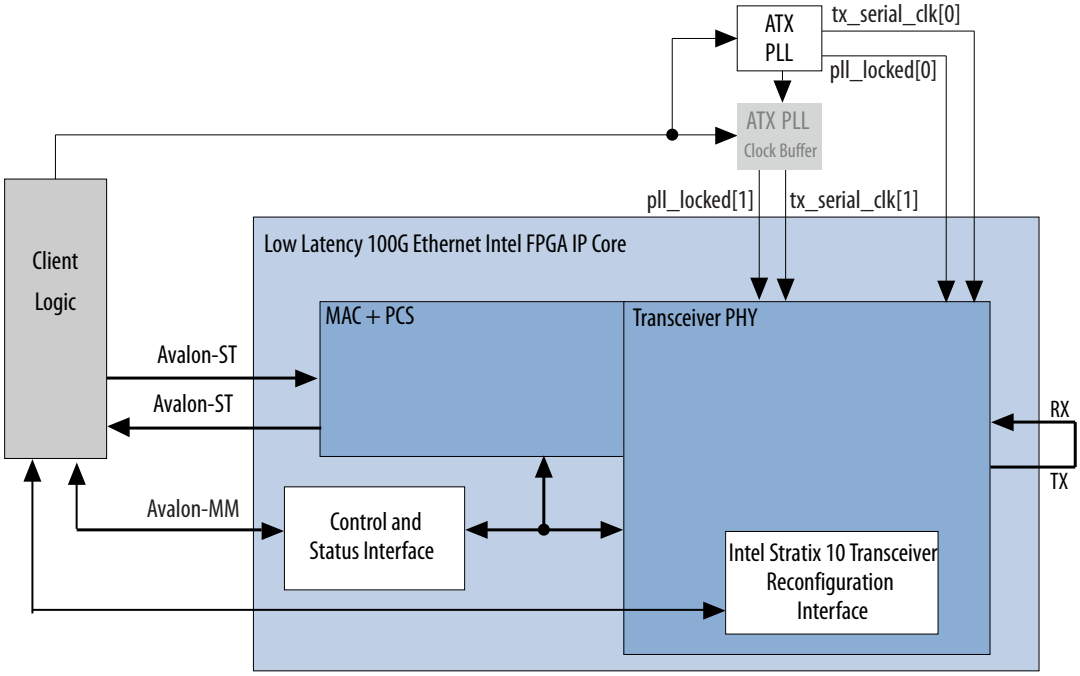
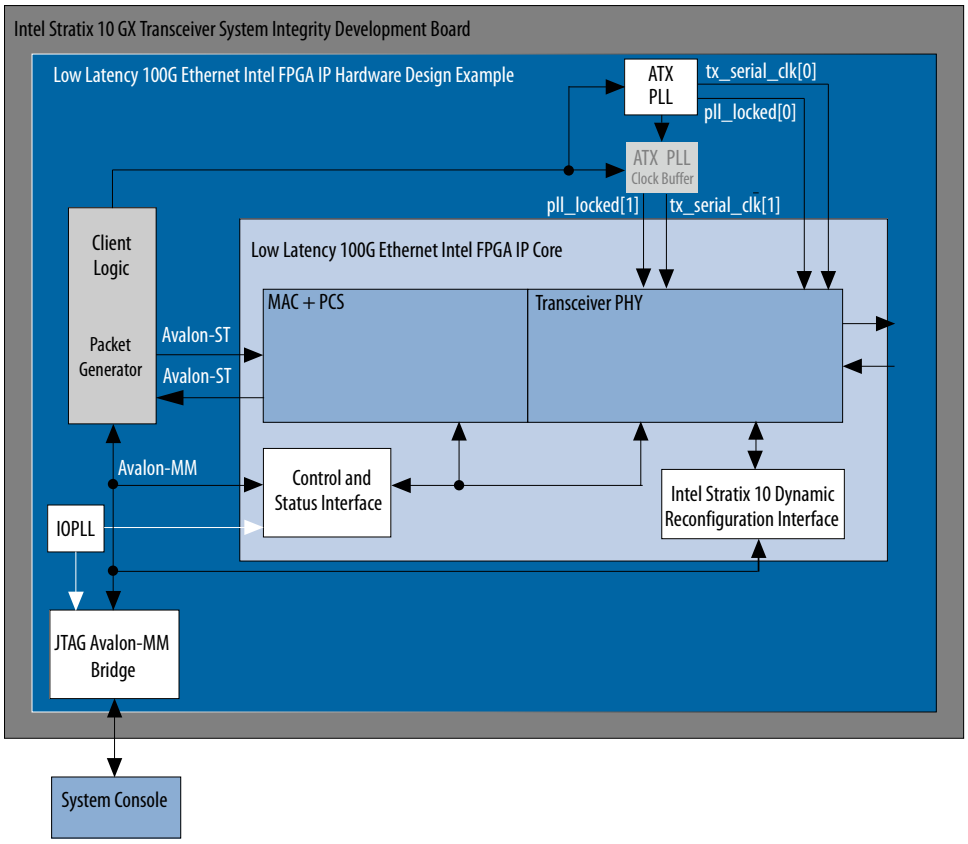


Table 1. Low Latency 100G Ethernet Intel Stratix 10 FPGA Core Testbench File Descriptions

File Names	Description
Key Testbench and Simulation Files	
`basic_avl_tb_top.v`	Top-level testbench file. The testbench instantiates the DUT and runs Verilog HDL tasks to generate and accept packets.
Testbench Scripts	
`run_vsim.do`	The Mentor Graphics* ModelSim* script to run the testbench.
`run_vcs.sh`	The Synopsys* VCS* script to run the testbench.
`run_vcsmx.sh`	The Synopsys VCS MX script (combined Verilog HDL and SystemVerilog with VHDL) to run the testbench. *Note:* Use `run_vcsmx.sh` when RS-FEC is enabled.
`run_ncsim.sh`	The Cadence NCSim script to run the testbench.
`run_xcelium.sh`	The Cadence Xcelium* script to run the testbench.

1.3. Hardware Design Example Components

Figure 4. Low Latency 100G Ethernet Intel Stratix 10 FPGA Hardware Design Example High Level Block Diagram



The Low Latency 100G Ethernet Intel Stratix 10 FPGA hardware design example includes the following components:

- Low Latency 100G Ethernet Intel Stratix 10 FPGA core.
- Client logic that coordinates the programming of the IP core and packet generation.
- One ATX PLL and one clock buffer to drive the device transceiver channels.
- IOPLL to generate a 100 MHz clock from a 50 MHz input clock to the hardware design example.
- JTAG controller that communicates with the System Console. You communicate with the client logic through the System Console.

The hardware design example uses `run_test` command to initiate packet transmission from packet generator to the IP core. Use the `loop_on` command to turn on internal serial loopback for this design example. If the internal serial loopback is turned on, the IP core receives the packets and transmit to the packet generator. The MAC checks the received packets and updates the statistic counters. Use the `chkmac_stats` command in the system console to read and print out the MAC statistic registers once the packet transmissions completed.

Table 2. Low Latency 100G Ethernet Intel Stratix 10 FPGA Core Hardware Design Example File Descriptions

File Names	Description
`alt_e100s10.qpf`	Intel Quartus® Prime project file
`alt_e100s10.qsf`	Intel Quartus Prime project settings file
`alt_e100s10.sdc`, `alt_e100s10_clock.sdc`	Synopsys Design Constraints files. You can copy and modify these files for your own Low Latency 100G Ethernet Intel Stratix 10 FPGA design.
`alt_e100s10.v`	Top-level Verilog HDL design example file
`common/`	Hardware design example support files
`hwtest/main.tcl`	Main file for accessing System Console

Related Information

- Intel Stratix 10 GX Signal Integrity Development Kit Webpage
- Testing the Low Latency 100G Ethernet Intel Stratix 10 FPGA Hardware Design Example on page 12
  For more information on available commands to test the hardware design example.

1.4. Generating the Design

Figure 5. Procedure

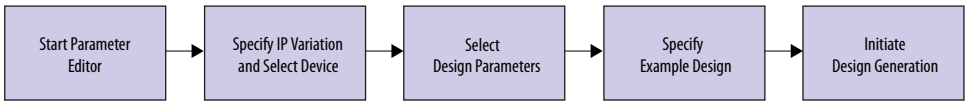
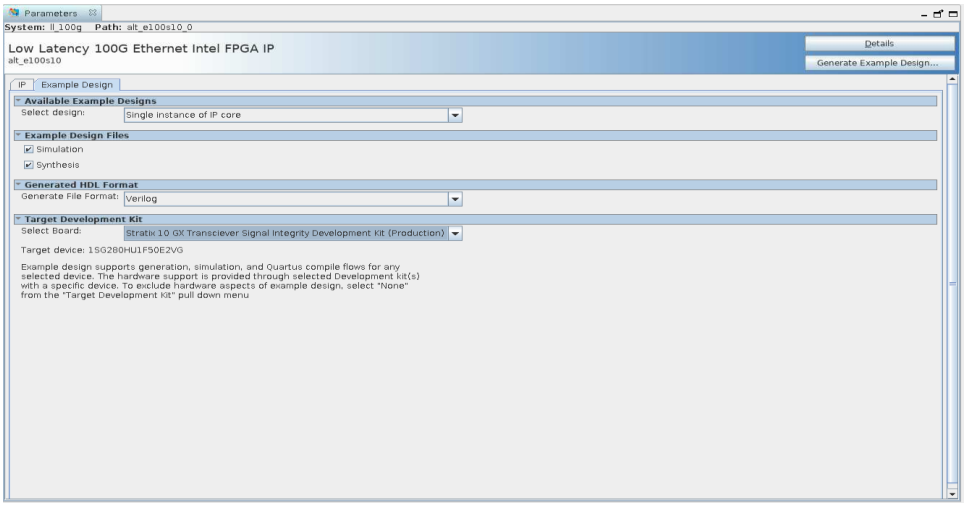


Figure 6. Example Design Tab in the Low Latency 100G Ethernet Intel Stratix 10 FPGA Parameter Editor



Follow these steps to generate the Low Latency 100G Ethernet Intel Stratix 10 FPGA hardware design example and testbench:

1. If you do not already have an Intel Quartus Prime Pro Edition project in which to integrate your Low Latency 100G Ethernet Intel Stratix 10 FPGA core, you must create one.
   a. In the Intel Quartus Prime Pro Edition, click **File ➤ New Project Wizard** to create a new Quartus Prime project, or **File ➤ Open Project** to open an existing Intel Quartus Prime project. The wizard prompts you to specify a device.
   b. Specify the device family **Intel Stratix 10** and select a device that meets all of these requirements:
      - Transceiver tile is L-tile or H-tile (any transceiver tile)
      - Transceiver speed grade is –1 or –2
      - Core speed grade is –1 or –2
      - Production version devices
   c. Click **Finish**.
2. In the IP Catalog, locate and select **Low Latency 100G Ethernet**. The **New IP Variation** window appears.
3. Specify a top-level name <*your_ip*> for your custom IP variation. The parameter editor saves the IP variation settings in a file named <*your_ip*>.ip.
4. Click **OK**. The parameter editor appears.
5. On the **IP** tab, specify the parameters for your IP core variation.

   Note: **Enable RX/TX statistics counters** parameter is enabled by default.
6. On the **Example Design** tab, under **Example Design Files**, select the **Simulation** option to generate the testbench, and select the **Synthesis** option to generate the compilation-only and hardware design examples.

   Note: You must select at least one of the **Simulation** and **Synthesis** options to generate the design example.
7. On the **Example Design** tab, under **Generated HDL Format**, only Verilog HDL is available. This IP core does not support VHDL.
8. Under **Target Development Kit** select the **Stratix 10 GX Transceiver Signal Integrity Development Kit (Production)**, **Stratix 10 GX Transceiver Signal Integrity Development Kit (ES)**, or **None**.

Note: When you select **Stratix 10 GX Transceiver Signal Integrity Development Kit (Production)** or **Stratix 10 GX Transceiver Signal Integrity Development Kit (ES)** as the **Target Development Kit**, the design example is generated based on a specific device and it overwrites the device you selected in your project file. If you select **None** as the **Target Development Kit**, ensure the device you selected is the correct device and make changes to the pins assignment in the `.qsf` file. By default, the `.qsf` file is generated based on the device used in Stratix 10 GX Transceiver Signal Integrity Development Kit.

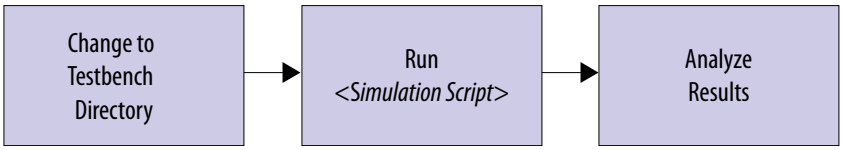
9. Click the **Generate Example Design** button. The **Select Example Design Directory** window appears.
10. If you want to modify the design example directory path or name from the defaults displayed (`alt_e100s10_0_example_design`), browse to the new path and type the new design example directory name (*<design_example_dir>*).

Related Information

- IP Core Parameters
  Provides more information about customizing your IP core.
- Intel Stratix 10 GX Signal Integrity Development Kit Webpage

1.5. Simulating the Design Example Testbench

Figure 7. Procedure



Follow these steps to simulate the testbench:

1. Change to the testbench simulation directory *<design_example_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 3. Steps to Simulate the Testbench

Simulator	Instructions
Mentor Graphics ModelSim	In the command line, type `vsim -do run_vsim.do` If you prefer to simulate without bringing up the ModelSim GUI, type `vsim -c -do run_vsim.do`

continued...

Simulator	Instructions
	Note: The ModelSim - Intel FPGA Edition simulator does not have the capacity to simulate this IP core. You must use another supported ModelSim simulator such as ModelSim SE.
Cadence NCSim	In the command line, type `sh run_ncsim.sh`
Cadence Xcelium	In the command line, type `sh run_xcelium.sh`
Synopsys VCS/ VCS MX	In the command line, type `sh run_vcs.sh` or `sh run_vcsmx.sh` *Note:* `run_vcs.sh` is only available if you select **Verilog** as the **Generated HDL Format**. If **RS-FEC** is enabled or you select **VHDL** as the **Generated HDL Format**, you must simulate the testbench with a mixed language simulator using `run_vcsmx.sh`.

The successful test run displays output confirming the following behavior:

1. Waiting for the ATX PLLs to lock.
2. Waiting for RX transceiver reset to complete.
3. Waiting for RX alignment.
4. Sending ten packets.
5. Receiving ten packets.
6. Displaying `Testbench complete.`

The following sample output illustrates a successful simulation test run:

```
ATX PLLs Locked
***************************************************
***************  Transmit Ready   *****************
***************************************************
Waiting for the receiver to be ready
Receive transceivers are out of reset
Waiting for RX alignment
Rx Alignment Status Update 1/4: Word/Block lock is acquired over all virtual
lanes
Rx Alignment Status Update 2/4: Virtual lanes Ordered
Rx Alignment Status Update 3/4: RX deskew lock acquired
Rx Alignment Status Update 4/4: RX alignment lock acquired
Rx is fully aligned with Tx
***************************************************
*************   Receive Ready   *****************
***************************************************
Transmitting test data
** Sending Packet           1...
** Sending Packet           2...
** Sending Packet           3...
** Sending Packet           4...
** Sending Packet           5...
** Sending Packet           6...
** Sending Packet           7...
** Sending Packet           8...
** Sending Packet           9...
** Sending Packet          10...
** Received Packet          1...
** Received Packet          2...
** Received Packet          3...
** Received Packet          4...
** Received Packet          5...
** Received Packet          6...
** Received Packet          7...
** Received Packet          8...
** Received Packet          9...
** Received Packet         10...
**
```

```
** Testbench complete.
**
**************************************
```

1.6. Compiling the Compilation-Only Project

To compile the compilation-only example project, follow these steps:

1. Ensure compilation design example generation is complete.
2. In the Intel Quartus Prime software, open the Intel Quartus Prime project `<design_example_dir>/compilation_test_design/alt_e100s10.qpf`.
3. On the Processing menu, click **Start Compilation**.

After successful compilation, reports for timing and for resource utilization are available in your Intel Quartus Prime Pro Edition session.

Related Information

Block-Based Design Flows

1.7. Compiling and Configuring the Design Example in Hardware

To compile the hardware design example and configure it on your Intel Stratix 10 device, follow these steps:

1. Ensure hardware design example generation is complete.
2. In the Intel Quartus Prime Pro Edition software, open the Intel Quartus Prime project `<design_example_dir>/hardware_test_design/alt_e100s10.qpf`.
3. On the Processing menu, click **Start Compilation**.
4. After successful compilation, a `.sof` file is available in your <*design_example_dir*>/hardware_test_design/output_files directory. Follow these steps to program the hardware design example on the Intel Stratix 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 Intel Stratix 10 Transceiver Signal Integrity Development Kit to which your Intel Quartus Prime Pro Edition session can connect.
   e. Ensure that **Mode** is set to **JTAG**.
   f. Select the Intel Stratix 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**.

Related Information

- Block-Based Design Flows
- Programming Intel FPGA Devices

- Analyzing and Debugging Designs with System Console

1.8. Testing the Hardware Design Example

After you compile the Low Latency 100G Ethernet Intel Stratix 10 FPGA core design example and configure it on your Intel Stratix 10 device, you can use the System Console to program the IP core and its embedded Native PHY IP core registers.

To turn on the System Console and test the hardware design example, follow these steps:

1. After the hardware design example is configured on the Intel Stratix 10 device, in the Intel Quartus Prime Pro Edition software, on the **Tools** menu, click **System Debugging Tools ➤ System Console**.
2. In the Tcl Console pane, type `cd hwtest` to change directory to `<design_example_dir>/hardware_test_design/hwtest`.
3. Type `source main.tcl` to open a connection to the JTAG master.

You can program the IP core with the following design example commands:

- `chkphy_status`: Displays the clock frequencies and PHY lock status.
- `chkmac_stats`: Displays the values in the MAC statistics counters.
- `clear_all_stats`: Clears the IP core statistics counters.
- `start_pkt_gen`: Starts the packet generator.
- `stop_pkt_gen`: Stops the packet generator.
- `loop_on`: Turns on internal serial loopback
- `loop_off`: Turns off internal serial loopback.
- `reg_read <addr>`: Returns the IP core register value at *<addr>*.
- `reg_write <addr> <data>`: Writes *<data>* to the IP core register at address *<addr>*.

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               :0 (KHZ)
          TXCLK                :39063   (KHZ)
          RXCLK                :39064   (KHZ)
          RX RECOV CLK         :31250   (KHZ)
          TX-IO CLOCK          :31251   (KHZ)
 RX PHY Status Polling

 Rx Frequency Lock Status       0x0000000f

 Mac Clock in OK Condition?     0x00000007

 Rx Frame Error                 0x00000000

 Rx PHY Fullly Aligned?         0x00000001

 Rx AM LOCK Condition?          0x00000001

 Rx Lanes Deskewed Condition?   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                   : 8971
65 - 127 Byte Frames             : 7995
128 - 255 Byte Frames            : 15074
256 - 511 Byte Frames            : 28808
512 - 1023 Byte Frames           : 57749
1024 - 1518 Byte Frames          : 55550
1519 - MAX Byte Frames           : 1664270
> MAX Byte Frames                : 0
Rx Frame Starts                  : 0
Multicast data OK  Frame         : 0
Broadcast data OK  Frame         : 0
Unicast data OK  Frames          : 1838417
Multicast Control Frames         : 0
Broadcast Control Frames         : 0
Unicast Control Frames           : 415
Pause Control Frames             : 0
Payload Octets OK                : 12416446222
Frame Octets OK                  : 12449532850
 ============================================================================
```

```
                        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                   : 8971
65 - 127 Byte Frames             : 7995
128 - 255 Byte Frames            : 15074
256 - 511 Byte Frames            : 28808
512 - 1023 Byte Frames           : 57749
1024 - 1518 Byte Frames          : 55550
1519 - MAX Byte Frames           : 1664270
> MAX Byte Frames                : 0
Tx Frame Starts                  : 0
Multicast data OK  Frame         : 0
Broadcast data OK  Frame         : 0
Unicast data OK  Frames          : 1838417
Multicast Control Frames         : 0
Broadcast Control Frames         : 0
Unicast Control Frames           : 0
Pause Control Frames             : 0
Payload Octets OK                : 12416446222
Frame Octets OK                  : 12449532850
---------------- Done ----------------
```

Related Information

- Analyzing and Debugging Designs with System Console
- Testing the Hardware Design Example using Ethernet Link Inspector on page 14

1.8.1. Testing the Hardware Design Example using Ethernet Link Inspector

You can also test your design using the Ethernet Link Inspector (ELI) tool available in System console.

Design examples have built-in JTAG to AVMM bridge allowing you to use the Ethernet Link Inspector. Refer to the user guide on how to use the ELI to test your design. The ELI tool is accessible via **System Console** in the **Tools ➤ Legacy Toolkits** in the Intel Quartus Prime Pro Edition software.

Related Information

Ethernet Link Inspector User Guide for Intel Stratix 10 Devices

1.8.2. Testing the Hardware Design Example using Ethernet Toolkit

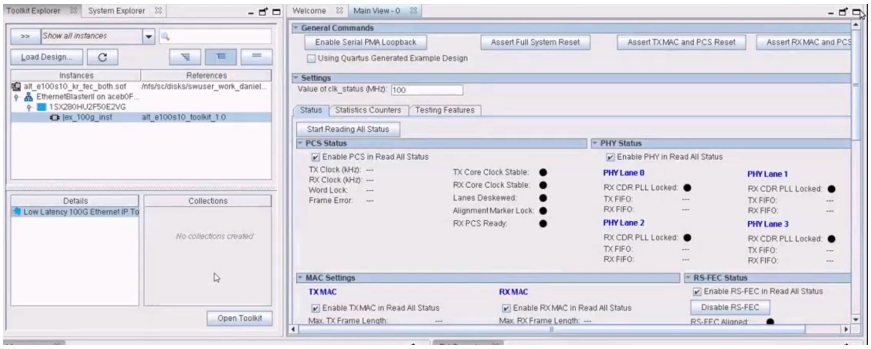
The Ethernet Toolkit (ETK) is a tool that continuously monitors an Ethernet link.

Available in the Intel Quartus Prime Pro Edition software version 19.4 and later, the toolkit features list follows the standalone Ethernet Link Inspector tool. The toolkit allows you to review the PHY status, PCS status, MAC status, RS-FEC status, and AN and LT status.

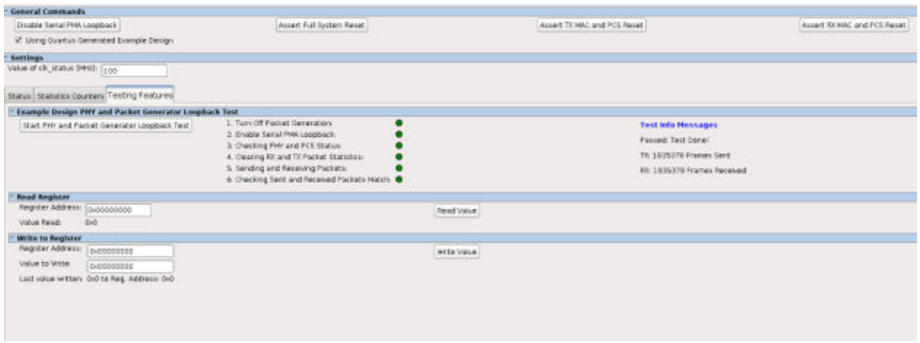
Follow these steps to test your design using the Ethernet Toolkit:

1. In the **IP** tab of the Low Latency 100G Ethernet Intel Stratix 10 FPGA IP, select **Enable JTAG to Avalon Master Bridge**.
2. Compile your design.
3. In the Intel Quartus Prime Pro Edition, click **Tools ➤ Programmer**, click **Hardware Setup**.
   a. Select a programming hardware setup from currently selected hardware or add new hardware. Click **Close.**
   b. Click **Start** to program your design onto the device.
4. Click **Tools ➤ System Debugging Tools ➤ System Console** to open a new system console window.
5. In the **System Console**, follow these steps to open the Ethernet Toolkit:
   a. Select **File ➤ Load Design** to load the programming file (*.sof). Click **OK**.
   b. In the Instances pane, select the Ethernet IP instance.
   c. In the Details pane, select **Low Latency 100G EthernetIP Toolkit** from the toolkit list.

Figure 8. Ethernet Toolkit Explorer View



   d. In the **General Commands** tab, click **Enable Serial PMA Loopback** to establish the serial loopback.
   e. The **Settings ➤ Status** tab provides link status enable and control options such as PCS and PHY Status, MAC settings, and AN and LT status. Click **Start Reading All Status** to start reading the link status.
   f. The **Settings ➤ Statistics Counters** tab provides the example design packet generator settings, transmitter and receiver statistics.
      i. Deselect **MAC Loopback Mode**.
      ii. Select your preferred **Packet Generator Modes** option.
      iii. Click **Start Packet Generator**. This enables the packet generator.
      iv. Click **Start Reading Transmitter Statistics** and **Start Reading Receiver Statistics**.
   g. The **Settings ➤ Testing Features** tab provides a series of testing features to test the Ethernet link.
      i. Click **Start PHY and Packet Generator Loopback Test** to test your Ethernet link.


General Commands
Disable Serial PMA Loopback
Assert Full System Reset
Assert TX MAC and PCS Reset
Assert RX MAC and PCS Reset
Using Quartus Generated Example Design
Settings
Value of clk_status (MHz):
100
Status
Statistics Counters
Testing Features
Example Design PHY and Packet Generator Loopback Test
Start PHY and Packet Generator Loopback Test
1. Turn-Off Packet Generation:
2. Enable Serial PMA Loopback:
3. Checking PHY and PCS Status:
4. Clearing RX and TX Packet Statistics:
5. Sending and Receiving Packets:
6. Checking Sent and Received Packets Match:
Test Info Messages
Passed: Test Done!
TX: 1835378 Frames Sent
RX: 1835378 Frames Received
Read Register
Register Address:
0x00000000
Value Read:
0x0
Read Value
Write to Register
Register Address:
0x00000000
Value to Write:
0x00000000
Last value written: 0x0 to Reg. Address: 0x0
Write Value

2. Design Example Description

The design example demonstrates the functions of the Low Latency 100G Ethernet Intel Stratix 10 FPGA core with transceiver interface compliant with the IEEE 802.3ba standard CAUI-4 specification. You can generate the design from the **Example Design** tab in the Low Latency 100G Ethernet Intel Stratix 10 FPGA parameter editor.

To generate the design example, you must first set the parameter values for the IP core variation you intend to generate in your end product. Generating the design example creates a copy of the IP core; the testbench and hardware design example use this variation as the DUT. If you do not set the parameter values for the DUT to match the parameter values in your end product, the design example you generate does not exercise the IP core variation you intend.

Note: The testbench demonstrates a basic test of the IP core. It is not intended to be a substitute for a full verification environment. You must perform more extensive verification of your own Low Latency 100G Ethernet Intel Stratix 10 FPGA design in simulation and in hardware.

Related Information

Low Latency 100G Ethernet Intel Stratix 10 FPGA IP User Guide

2.1. Features

DUT features:

- Standard CAUI-4 external interface consisting of four FPGA hard serial transceiver lanes operating at 25.78125 Gbps.
- Avalon Memory-Mapped (Avalon-MM) management interface to access the IP core control and status registers.
- RX CRC checking and error reporting.
- TX error insertion capability to transmit error frame at the end of a packet cycle.
- Hardware and software reset control.

2.2. Design Example Behavior

The testbench sends traffic through the IP core, exercising the transmit side and receive side of the IP core. In the hardware design example, you can program the IP core in internal serial loopback mode and generate traffic on the transmit side that loops back through the receive side.

To speed up simulation, the IP core simulation model sends alignment marker tags at shorter intervals than required by the IEEE Ethernet standard. The standard specifies an alignment marker interval of 16384 words in each virtual lane. The simulation model with the testbench implements an alignment marker interval of 512 words.

2.3. Design Example Interface Signals

The Low Latency 100G Ethernet Intel Stratix 10 FPGA testbench is self-contained and does not require you to drive any input signals.

Table 4. Low Latency 100G Ethernet Intel Stratix 10 FPGA Hardware Design Example Interface Signals

Signal	Direction	Comments
clk50	Input	Drive at 50 MHz. The intent is to drive this input from a 50 MHz oscillator on the board.
clk_ref_r	Input	Drive at 644.53125 or 322.265625 MHz.
cpu_resetn	Input	Resets the IP core. Active low. Drives the global hard reset csr_reset_n to the IP core.
tx_serial[3:0]	Output	Transceiver PHY output serial data.
rx_serial[3:0]	Input	Transceiver PHY input serial data.
user_led[7:0]	Output	Status signals. Currently the design example drives all of these signals to a constant value of 0.

Related Information

Interfaces and Signal Descriptions
Provides detailed descriptions of the Low Latency 100G Ethernet Intel Stratix 10 FPGA core signals and the interfaces to which they belong.

2.4. Design Example Registers

Table 5. Low Latency 100G Ethernet Intel Stratix 10 FPGA Hardware Design Example Register Map

Lists the memory mapped register ranges for the hardware design example. You access these registers with the reg_read and reg_write functions in the System Console.

Word Offset	Register Type
0x000000	KR4 registers
0x000300	PHY registers
0x000400	TX MAC registers
0x000500	RX MAC registers
0x000800	TX Statistics Counter registers
0x000900	RX Statistics Counter registers
0x001000	Packet Client registers
0x004000	PMA registers(1)

(1) Each PHY channel has its PMA registers set. PMA registers sets are offset from each other by 0x800 channel offset, starting at the 0x4000 word offset.

Table 6. Packet Client Registers

You can customize the Low Latency 100G Ethernet Intel Stratix 10 FPGA 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
0x1008	Packet Size Configure	[29:0]	Specify the transmit packet size in bytes. These bits have dependencies to PKT_GEN_TX_CTRL register. • Bit [29:16]: Specify the upper limit of the packet size in bytes. This is only applicable to incremental mode. • Bit [13:0]: — For fixed mode, these bits specify the transmit packet size in bytes. — For incremental mode, these bits specify the lower limit of the transmit packet size in bytes.	0x25800040	RW
0x1009	Packet Number Control	[31:0]	Specify the number of packets to transmit from the packet generator. These bits have dependencies to PKT_GEN_TX_CTRL register. • Bit [31]: — 1: For fixed/incremental mode with fixed number of packets, this is the number of packets enable bit. — 0: For random mode or fixed/incremental mode without fixed number of packets. • Bits [30:0]: — If bit [31] is set, specifies the number of packets to be sent by the packet generator. For more information, refer to the Packet Generator Programming Sequence on page 20.	0xA	RW
0x1010	PKT_GEN_TX_CTRL	[7:0]	• Bit [0]: Reserved. • Bit [1]: Packet generator status control bit. — 0: Packet generator starts sending out the packets. — 1: Packet generator stops sending out the packets. • 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. • Bit [5:4]: — 00: Random mode — 01: Fixed mode — 10: Incremental mode • Bit [6]: — 1: For transmission without gap in between packets. — 0: For transmission with random gap in between packets. • Bit [7]: Reserved.	0x6	RW

continued...

Addr	Name	Bit	Description	HW Reset Value	Access
0x1011	Destination address lower 32 bits	[31:0]	Destination address (lower 32 bits)	0x56780ADD	RW
0x1012	Destination address upper 16 bits	[15:0]	Destination address (upper 16 bits)	0x1234	RW
0x1013	Source address lower 32 bits	[31:0]	Source address (lower 32 bits)	0x43210ADD	RW
0x1014	Source address upper 16 bits	[15:0]	Source address (upper 16 bits)	0x8765	RW
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

- Packet Generator Programming Sequence on page 20
- Low Latency 100G Ethernet Intel Stratix 10 FPGA core register descriptions

2.4.1. Packet Generator Programming Sequence

You can initiate the packet transmission from the packet generator to the IP core by programming `Packet Size Configure`, `Packet Number Control`, and `PKT_GEN_TX_CTRL` registers.

The programming sequence varies based on a selected mode `PKT_GEN_TX_CTRL[5:4]`:

In Random Mode:

1. Set `Packet Number Control[31]` to 0.
2. Set `PKT_GEN_TX_CTRL[1]` to 0 to start the packet transmission through the packet generator.
3. Set `PKT_GEN_TX_CTRL[1]` to 1 to stop the packet generator.

In Fixed/Incremental Mode—with fixed number of packets:

1. Configure `Packet Size Configure`.
2. Configure the number of packets: Bit `[31]` must be set to specify `Packet Number Control[30:0]`
3. Set `PKT_GEN_TX_CTRL[1]` to 0 to start the packet transmission through the packet generator.
4. The packet generator stops when the number of packets reaches 0.

In Fixed/Incremental Mode— without fixed number of packets:

1. Configure `Packet Size Configure`.
2. Set `Packet Number Control[31]` to 0.
3. Set `PKT_GEN_TX_CTRL[1]` to 0 to start the packet transmission through the packet generator.
4. Set `PKT_GEN_TX_CTRL[1]` to 1 to stop the packet generator.

3. Low Latency 100G Ethernet Intel Stratix 10 FPGA IP Design Example User Guide Archives

IP versions are the same as the Intel Quartus Prime Design Suite software versions up to v19.1. From Intel Quartus Prime Design Suite software version 19.2 or later, IPs have a new IP versioning scheme.

Intel Quartus Prime Version	IP Core Version	User Guide
19.4	19.1.1	Low Latency 100G Ethernet Intel Stratix 10 FPGA IP Design Example User Guide
19.3	19.1.1	Low Latency 100G Ethernet Intel Stratix 10 FPGA IP Design Example User Guide
19.1	19.1	Low Latency 100G Ethernet Intel Stratix 10 FPGA IP Design Example User Guide
18.0	18.0	Low Latency 100G Ethernet Intel Stratix 10 FPGA IP Design Example User Guide
17.1	17.1	Intel Stratix 10 Low Latency 100G Ethernet Design Example User Guide

4. Document Revision History for the Low Latency 100G Ethernet Intel Stratix 10 FPGA IP Design Example User Guide

Document Version	Intel Quartus Prime Version	IP version	Changes
2020.04.13	20.1	19.2.0	• Updated the *Low Latency 100G Ethernet Intel Stratix 10 FPGA Simulation Design Example Block Diagram* and the *Low Latency 100G Ethernet Intel Stratix 10 FPGA Hardware Design Example High Level Block Diagram* to emphasize that the 2nd ATX PLL configures as a clock buffer. • Corrected the register type in the *Design Example Registers* section. The 0x000300 word offset represents PHY registers, not RX PHY registers. • Added PMA registers word offset in the *Design Example Registers* section.
2019.12.16	19.4	19.1.1	• Added note to clarify `run_vcs.sh` and `run_vcsmx.sh` usage. • Updated sample output in the *Testing the Hardware Design Example* section. • Added a new topic: *Testing the Hardware Design Example using System Debugging Toolkits*
2019.09.30	19.3	19.1.1	• Added Synopsys VCS script option `run_vcsmx.sh` in *Low Latency 100G Ethernet Intel Stratix 10 FPGA Core Testbench File Descriptions* and *Steps to Simulate the Testbench* tables. • Added support for Xcelium* simulator. • Updated *Generating the Design* section. • Updated .sof file path in the *Compiling and Configuring the Design Example in Hardware* section. • Added Ethernet Link Inspector in the *Testing the Hardware Design Example* section. • Updated `Packet Number Control` and PKT_GEN_TX_CTRL registers in the *Packet Client Registers* table. • Added *Packet Generator Programming Sequence* section.
2019.05.15	19.1	19.1	Changed word "lower" to "upper" in the `Source address` register.
2018.06.29	18.0	18.0	• Added flow control feature in the *DUT features* list. • Added 322.2625 MHz support for PHY reference clock. • Added packet client registers description in the *Packet Client Registers* table.
2017.11.06	17.1	17.1	Initial release.