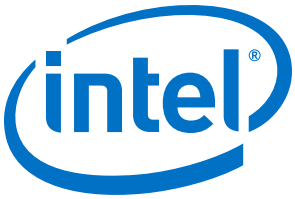


H-tile Hard IP for Ethernet Intel® Stratix® 10 FPGA IP Design Example User Guide

Updated for Intel® Quartus® Prime Design Suite: **18.0**

Subscribe
Send Feedback

UG-20122 | 2018.08.10
Latest document on the web: **PDF** | **HTML**

Contents

1. Quick Start Guide

The H-tile Hard IP for Ethernet Intel® FPGA IP core provides a simulation testbench and a hardware design example that supports compilation and hardware testing. When you generate the design example, the parameter editor automatically creates the files necessary to simulate, compile, and test the design in hardware.

In addition, You can download the compiled hardware design to the Intel Stratix® 10 GX Transceiver Signal Integrity Development Kit. Intel provides a compilation-only example project that you can use to quickly estimate IP core area and timing.

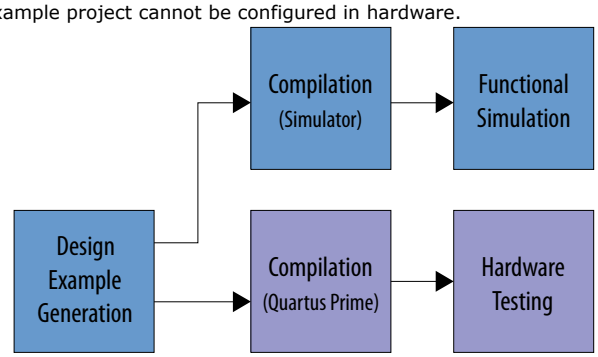
You can generate the design example H-tile Hard IP for Ethernet Intel FPGA IP core variation:

- 50 or 100-Gbps MAC+PCS (supports simulation testbench, compilation-only example project, and hardware design example)
- 50 or 100-Gbps PCS only (supports simulation testbench and compilation-only example project)
- 50 or 100-Gbps OTN (supports simulation testbench and compilation-only example project)
- 50 or 100-Gbps FlexE (supports simulation testbench and compilation-only example project)

Note: The H-tile Hard IP for Ethernet Intel FPGA IP provides preliminary support for the OTN feature. For further inquiries, contact your nearest Intel sales representative or file an Intel Premier Support (IPS) case on https://www.intel.com/content/www/us/en/programmable/my-intel/mal-home.html.

Figure 1. Development Steps for the Design Example

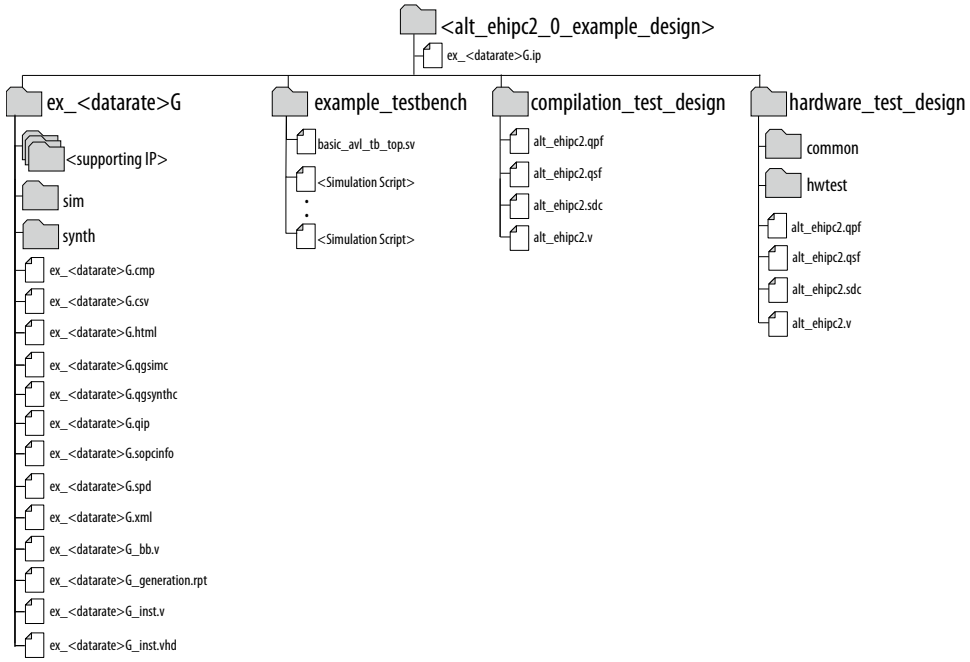
Future releases of the IP core also provide a hardware design example you can compile and test in hardware. The compilation-only example project cannot be configured in hardware.



1.1. Directory Structure

Figure 2. H-tile Hard IP for Ethernet Intel FPGA Design Example Directory Structure

<*datarate*> is either "50" or "100", depending on your IP core variation.



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.

Table 1. H-tile Hard IP for Ethernet Intel FPGA IP Core Testbench File Descriptions

File Names	Description
Key Testbench and Simulation Files	
basic_avl_tb_top.sv	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_ncsim.sh	The Cadence NCSim* script to run the testbench.

Table 2. H-tile Hard IP for Ethernet Intel FPGA IP Core Hardware Design Example File Descriptions

File Names	Description
alt_ehip2.qpf	Intel Quartus® Prime project file
alt_ehip2.qsf	Intel Quartus Prime project settings file

continued...

File Names	Description
`alt_ehip2.sdc`	Synopsys Design Constraints files. You can copy and modify these files for your own H-tile Hard IP for Ethernet Intel FPGA design.
`alt_ehip2.v`	Top-level Verilog HDL design example file
`common/`	Hardware design example support files
`hwtest/main.tcl`	Main file for accessing System Console

1.2. Generating the Design

Figure 3. Procedure

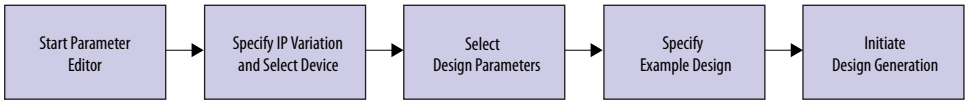
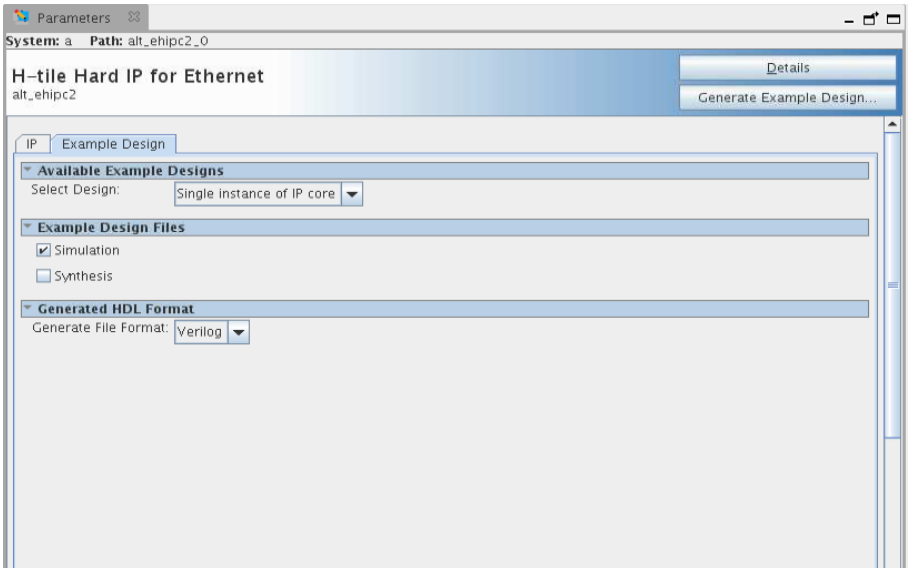


Figure 4. Example Design Tab in the H-tile Hard IP for Ethernet Intel FPGA Parameter Editor



Follow these steps to generate the H-tile Hard IP for Ethernet Intel 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 H-tile Hard IP for Ethernet Intel FPGA IP 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 H-tile
- Transceiver speed grade is 1 or 2
- Core speed grade is 1 or 2

c. Click **Finish**.

2. In the IP Catalog, locate and select **H-tile Hard IP for Ethernet Intel FPGA IP**. 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.

   Important: The design example testbench supports only the default IP parameter settings listed below. Any changes to these settings may cause simulation failure in the testbench.

IP Parameter Settings	Default Value
Ready latency	0
TX maximum frame size	1518
RX maximum frame size	1518
Enforce maximum frame size	Disable
Link fault generation option	Off
Stope TX traffic when link partner send pause	No
Bytes to remove from RX frames	Remove CRC bytes
Forward RX pause requests	Disable
Use source address insertion	Disable
TX VLAN detection	Enable
RX VLAN detection	Enable
Enable AN/LT	Disable
Enable Altera Debug Master endpoint (ADME)	Disable

6. On the **Example Design** tab, under **Example Design Files**, select the **Simulation** option to generate the testbench and the compilation-only project. Select the **Synthesis** option to generate the hardware design example.

   Note: You must select at least one of the **Simulation** and **Synthesis** options to generate the design example.

   Note: You must select the **Simulation** option to generate the testbench.

7. On the **Example Design** tab, under **Generated HDL Format**, select **Verilog** HDL or **VHDL**

   Note: If you select **VHDL**, you must simulate the testbench with a mixed-language simulator. The device under test in the ex_50G or ex_100G directory is a VHDL model, but the main testbench file is a System Verilog file.

8. Under **Target Development Kit** select the **Stratix 10 GXT Transceiver SOC Dev Kit**, or **Stratix 10 GXT Transceiver Signal Integrity Development Kit**to generate the hardware design example. Selecting **None** generates only the simulation and compilation-only design examples.

   Note: The compilation-only and hardware design examples target your project device. For correct hardware design functionality out of the box, you must ensure your project device is the device on your development kit.

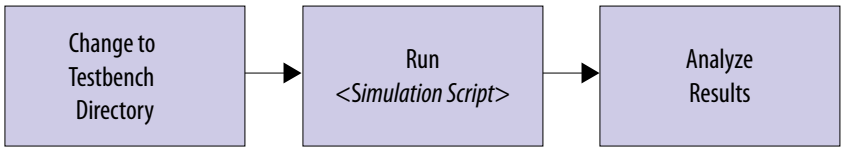
9. Click the **Generate Example Design** button. The **Select Example Design Directory** window appears.
10. If you wish to modify the design example directory path or name from the defaults displayed (`alt_ehipc2_0_example_design`), browse to the new path and type the new design example directory name (<*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

1.3. Simulating the H-tile Hard IP for Ethernet Intel FPGA Design Example Testbench

Figure 5. 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 or fourteen packets, receives the same number of 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` *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`
Synopsys VCS	In the command line, type `sh run_vcs.sh`

The successful test run displays output confirming the following behavior:

1. Waiting for the ATX PLL or ATX PLLs to lock.
2. Waiting for RX transceiver reset to complete.
3. Waiting for RX alignment.
4. Sending ten (MAC+PCS, OTN, and FlexE) or thirteen (PCS Only) packets.
5. Receiving those packets (MAC+PCS, OTN, and FlexE) or receiving and checking those packets (PCS Only).
6. Displaying `Testbench complete.`

1.4. 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 Pro Edition software, open the Intel Quartus Prime Pro Edition project `<design_example_dir>/compilation_test_design/alt_ehipc2.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

1.5. 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.

   Note: The hardware design example in Intel Quartus Prime version 18.0 only supports MAC+PCS variant.
2. In the Intel Quartus Prime Pro Edition software, open the Intel Quartus Prime project `<design_example_dir>/hardware_test_design/alt_ehip2.qpf`.
3. On the Processing menu, click **Start Compilation**.
4. After successful compilation, a `.sof` file is available in your specified 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

1.6. Testing the H-tile Hard IP for Ethernet Intel FPGA Hardware Design Example

After you compile the H-tile Hard IP for Ethernet Intel FPGA IP 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

Related Information

- Hardware Design Example Components on page 20
- Analyzing and Debugging Designs with System Console

2. Design Example Description

The design example demonstrates the basic functions of the H-tile Hard IP for Ethernet Intel FPGA IP core with the following variants:

Note: The hardware design example in Intel Quartus Prime version 18.0 only supports MAC +PCS variant.

- 50-Gbps datarate:
  - MAC + PCS
  - PCS Only
  - OTN
  - FlexE
- 100-Gbps datarate:
  - MAC + PCS
  - PCS Only
  - OTN
  - FlexE

Note: The H-tile Hard IP for Ethernet Intel FPGA IP provides preliminary support for the OTN feature. For further inquiries, contact your nearest Intel sales representative or file an Intel Premier Support (IPS) case on https://www.intel.com/content/www/us/en/programmable/my-intel/mal-home.html.

You can generate the design from the **Example Design** tab in the H-tile Hard IP for Ethernet Intel 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, compilation-only, 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.

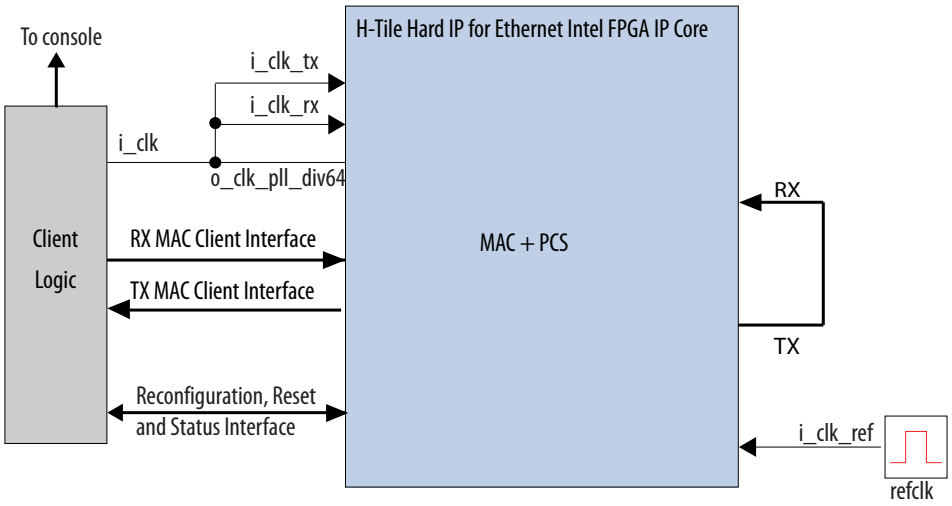
Note: The testbench demonstrates a basic test of the IP core. It is not intended to be a substitute for a full verification environment. You must perform more extensive verification of your own H-tile Hard IP for Ethernet Intel FPGA design in simulation and in hardware.

Related Information

Intel Stratix 10 H-Tile Hard IP for Ethernet IP Core User Guide

2.1. H-tile Hard IP for Ethernet Intel FPGA MAC + PCS Simulation Design Example

Figure 6. H-tile Hard IP for Ethernet Intel FPGA MAC + PCS Simulation Design Example Block Diagram



The testbench sends traffic through the IP core, exercising the transmit and receive MAC client interfaces of the IP core.

For 50-Gbps datarate, the simulation design example instantiates an ATX PLL to drive the transceiver channel.

For 100-Gbps datarate, the simulation design example instantiates a main ATX PLL for transceiver channel 0 and 1, and a clock buffer for channel 2 and 3.

The testbench in this design example performs the following:

1. The client logic resets the IP core.
2. Client logic waits for RX datapath to align.
3. Once alignment is complete, client logic transmits a series of packets to the IP core.
4. The client logic receives the same series of packets through RX MAC interface.
5. The client logic then checks the number of packets received and verifies that the packets have no errors.

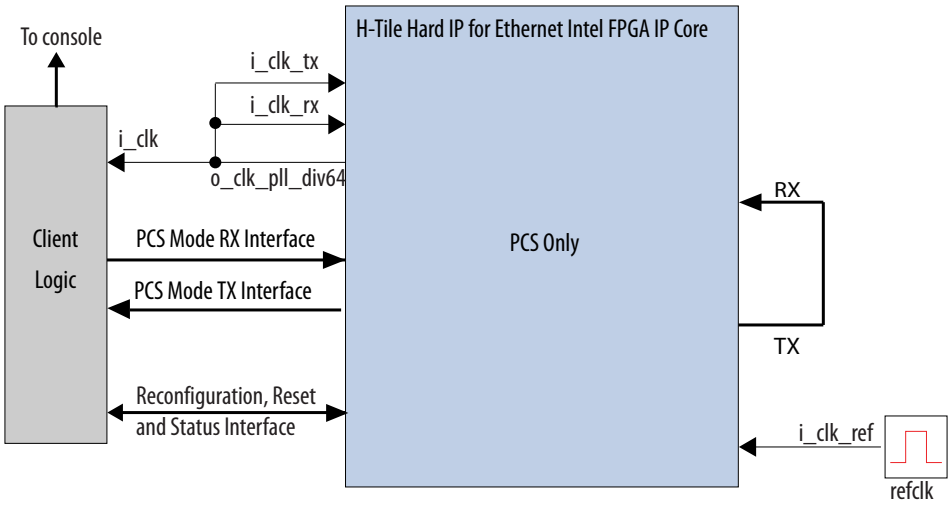
The following sample output illustrates a successful simulation test run for a 100-Gbps, MAC+PCS IP core variation. Times are in picoseconds. The times required for simulation of 100-Gbps variations are slightly longer than the times required for 50-Gbps variations.

```
Ref clock is 644.53125 MHz

================================================
Module ct1_hssi_cr2_ehip_pcs_interface
================================================
silicon_rev = 14nm5bcr2ea

waiting for o_tx_lanes_stable...
```

```
o_tx_lanes_stable is 1 at time                525000
waiting for tx_dll_lock....
TX DLL LOCK is 1 at time              25332363
waiting for tx_transfer_ready....
TX transfer ready is 1 at time              25652235
waiting for rx_transfer_ready....
RX transfer ready is 1 at time              31979703
EHIP PLD Ready out is 1 at time              32040000
EHIP reset out is 0 at time              32304000
EHIP reset ack is 0 at time              32612783
EHIP TX reset out is 0 at time              32928000
EHIP TX reset ack is 0 at time              82562795
waiting for EHIP Ready....
EHIP READY is 1 at time              82629435
EHIP RX reset out is 0 at time              82968000
waiting for rx reset ack....
EHIP RX reset ack is 0 at time              83029275
Waiting for RX Block Lock
EHIP RX Block Lock  is high at time             90223063
Waiting for AM lock
EHIP RX AM Lock  is high at time              91172683
Waiting for RX alignment
RX deskew locked
RX lane aligmnent locked
TX enabled
** Sending Packet           1...
** Sending Packet           2...
** Sending Packet           3...
** Sending Packet           4...
** Sending Packet           5...
** Sending Packet           6...
** Received Packet          1...
** Sending Packet           7...
** Received Packet          2...
** Sending Packet           8...
** Received Packet          3...
** Sending Packet           9...
** Received Packet          4...
** Sending Packet          10...
** Received Packet          5...
** Received Packet          6...
** Received Packet          7...
** Received Packet          8...
** Received Packet          9...
** Received Packet         10...
**
** Testbench complete.
**
****************************************
```

2.2. H-tile Hard IP for Ethernet Intel FPGA PCS Only Simulation Design Example

Figure 7. H-tile Hard IP for Ethernet Intel FPGA PCS Only Simulation Design Example Block Diagram



The testbench sends traffic through the IP core, exercising the transmit and receive Media Independent Interface (MII) of the IP core.

For 50-Gbps datarate, the simulation design example instantiates an ATX PLL to drive the transceiver channel.

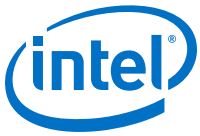
For 100-Gbps datarate, the simulation design example instantiates a main ATX PLL for transceiver channel 0 and 1, and a clock buffer for channel 2 and 3.

The testbench in this design example performs the following:

1. The client logic resets the IP core.
2. Client logic waits for RX datapath to align.
3. Once alignment is complete, client logic transmits a series of packets to the IP core through TX MII interface.
4. A counter drives `i_tx_mii_am` port with alignment marker insertion requests at the correct intervals.
5. The client logic receives the same series of packets through RX MII interface.
6. The client logic then checks the number of packets received.

The following sample output illustrates a successful simulation test run for a 100-Gbps, PCS Only IP core variation. Times are in picoseconds. The times required for simulation of 100-Gbps variations are slightly longer than the times required for 50-Gbps variations.

```
Ref clock is run at 625 MHz so whole numbers can used for all clock periods.
Multiply reported frequencies by 33/32 to get actual clock frequencies.

================================================
Module ct1_hssi_cr2_ehip_pcs_interface
================================================
```

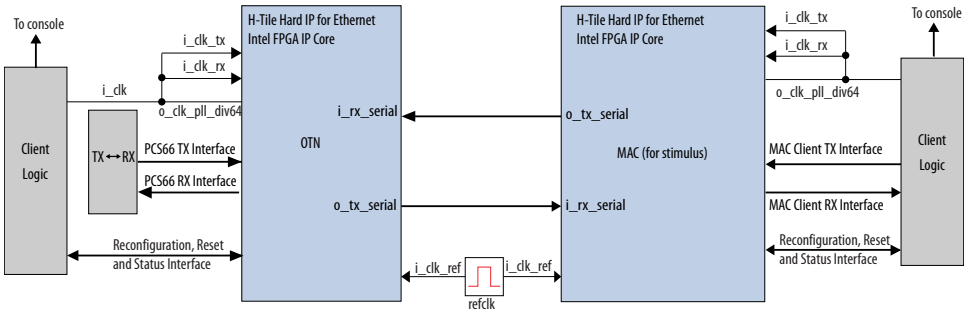


```
silicon_rev = 14nm5bcr2ea

waiting for o_tx_lanes_stable...
o_tx_lanes_stable is 1 at time                525000
waiting for tx_dll_lock....
TX DLL LOCK is 1 at time              25332363
waiting for tx_transfer_ready....
TX transfer ready is 1 at time              25652235
waiting for rx_transfer_ready....
RX transfer ready is 1 at time              31979703
EHIP PLD Ready out is 1 at time              32040000
EHIP reset out is 0 at time              32304000
EHIP reset ack is 0 at time              32612783
EHIP TX reset out is 0 at time              32928000
EHIP TX reset ack is 0 at time              82562795
waiting for EHIP Ready....
EHIP READY is 1 at time              82669419
EHIP RX reset out is 0 at time              83008000
waiting for rx reset ack....
EHIP RX reset ack is 0 at time              83055931
Waiting for RX Block Lock
EHIP Rx Block Lock  is high at time              90697873
Waiting for AM lock
EHIP Rx am Lock  is high at time              92122303
Waiting for RX alignment
RX deskew locked
RX lane aligmnent locked
Sending Packets
Receiving Packets
** Received Packet
0101010101010101fefefefefefefefe0101010101010101fefefefefefefefe...
** Matched packet   1 ...
** Received Packet
0202020202020202fdfdfdfdfdfdfdfd0202020202020202fdfdfdfdfdfdfdfd...
** Matched packet   2 ...
** Received Packet
0303030303030303fcfcfcfcfcfcfcfc0303030303030303fcfcfcfcfcfcfcfc...
** Matched packet   3 ...
** Received Packet
0404040404040404fbfbfbfbfbfbfbfb0404040404040404fbfbfbfbfbfbfbfb...
** Matched packet   4 ...
** Received Packet
0505050505050505fafafafafafafafa0505050505050505fafafafafafafafa...
** Matched packet   5 ...
** Received Packet
0606060606060606f9f9f9f9f9f9f9f906060606060606f9f9f9f9f9f9f9f9...
** Matched packet   6 ...
** Received Packet
0707070707070707f8f8f8f8f8f8f8f807070707070707f8f8f8f8f8f8f8f8...
** Matched packet   7 ...
** Received Packet
0808080808080808f7f7f7f7f7f7f7f708080808080808f7f7f7f7f7f7f7f7...
** Matched packet   8 ...
** Received Packet
0909090909090909f6f6f6f6f6f6f6f609090909090909f6f6f6f6f6f6f6f6...
** Matched packet   9 ...
** Received Packet
0a0a0a0a0a0a0a0af5f5f5f5f5f5f5f50a0a0a0a0a0a0a0af5f5f5f5f5f5f5f5...
** Matched packet  10 ...
** Received Packet
0b0b0b0b0b0b0b0bf4f4f4f4f4f4f4f40b0b0b0b0b0b0b0bf4f4f4f4f4f4f4f4...
** Matched packet  11 ...
** Received Packet
0c0c0c0c0c0c0c0cf3f3f3f3f3f3f30c0c0c0c0c0c0c0cf3f3f3f3f3f3f3f3...
** Matched packet  12 ...
** Received Packet
0d0d0d0d0d0d0d0df2f2f2f2f2f2f20d0d0d0d0d0d0d0df2f2f2f2f2f2f2f2...
** Matched packet  13 ...
** Received Packet
0e0e0e0e0e0e0e0ef1f1f1f1f1f1f10e0e0e0e0e0e0e0ef1f1f1f1f1f1f1f1...
** Matched packet  14 ...
```

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

2.3. H-tile Hard IP for Ethernet Intel FPGA OTN Simulation Design Example

Figure 8. H-tile Hard IP for Ethernet Intel FPGA OTN Simulation Design Example Block Diagram



Note: The H-tile Hard IP for Ethernet Intel FPGA IP provides preliminary support for the OTN feature. For further inquiries, contact your nearest Intel sales representative or file an Intel Premier Support (IPS) case on https://www.intel.com/content/www/us/en/programmable/my-intel/mal-home.html.

The testbench sends traffic through the IP core with OTN mode, exercising the transmit and receive PCS66 interfaces using a separate H-tile Hard IP for Ethernet Intel FPGA MAC as a stimulus generator.

For 50-Gbps datarate, the simulation design example instantiates an ATX PLL to drive the transceiver channel.

For 100-Gbps datarate, the simulation design example instantiates a main ATX PLL for transceiver channel 0 and 1, and a clock buffer for channel 2 and 3.

The testbench in this design example performs the following:

1. The client logic resets both the IP cores.
2. The stimulus client logic waits for the stimulus RX datapath and OTN RX datapath to align.
3. Once alignment is complete, the stimulus client logic transmits a series of packets to the OTN IP core.
4. The OTN IP core receives the series of packets and transmits back to the stimulus MAC IP core.
5. The stimulus client logic then checks the number of packets received and verifies that the packets have no errors.

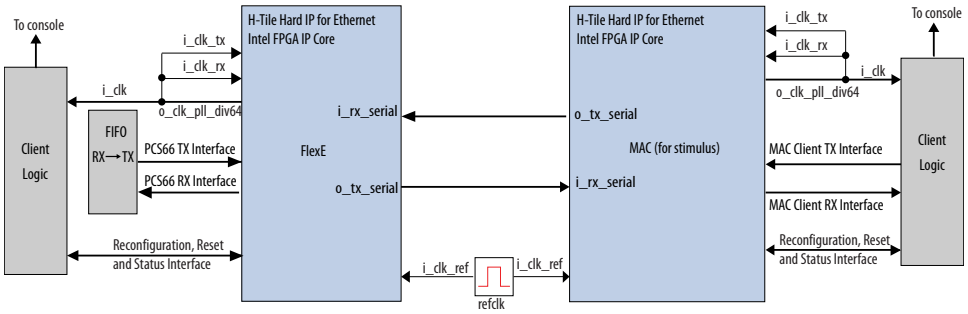
The following sample output illustrates a successful simulation test run for a 100-Gbps OTN IP core variation. Times are in picoseconds. The times required for simulation of 100-Gbps variations are slightly longer than the times required for 50-Gbps variations.

```
# Ref clock is 644.53125 MHz
# Ref clock is 644.53125 MHz
# iatpg_pipeline_global_en is set
.
.
.
# iatpg_pipeline_global_en is set
# test_dut:waiting for o_tx_lanes_stable...
# dut:waiting for o_tx_lanes_stable...
# test_dut:o_tx_lanes_stable is 1 at time                525000
# test_dut:waiting for tx_dll_lock....
# dut:o_tx_lanes_stable is 1 at time                525000
# dut:waiting for tx_dll_lock....
# dut:TX DLL LOCK is 1 at time             47806703
# dut:waiting for tx_transfer_ready....
# dut:TX transfer ready is 1 at time              48126575
# dut:waiting for rx_transfer_ready....
# dut:RX transfer ready is 1 at time              59518683
# dut:EHIP PLD Ready out is 1 at time             59576000
# dut:EHIP reset out is 0 at time             59840000
# dut:EHIP reset ack is 0 at time             60151763
# dut:EHIP TX reset out is 0 at time             60488000
# dut:EHIP TX reset ack is 0 at time            110085115
# dut:waiting for EHIP Ready....
# dut:EHIP READY is 1 at time            110178411
# dut:EHIP RX reset out is 0 at time            110520000
# dut:waiting for rx reset ack....
# dut:EHIP RX reset ack is 0 at time            110578251
# dut:Waiting for RX Block Lock
# test_dut:TX DLL LOCK is 1 at time            124725923
# test_dut:waiting for tx_transfer_ready....
# test_dut:TX transfer ready is 1 at time             125045795
# test_dut:waiting for rx_transfer_ready....
# test_dut:RX transfer ready is 1 at time             136437903
# test_dut:EHIP PLD Ready out is 1 at time             136496000
# test_dut:EHIP reset out is 0 at time            136760000
# test_dut:EHIP reset ack is 0 at time            137070983
# test_dut:EHIP TX reset out is 0 at time             137408000
# test_dut:EHIP TX reset ack is 0 at time             187001003
# test_dut:waiting for EHIP Ready....
# test_dut:EHIP READY is 1 at time            187107627
# test_dut:EHIP RX reset out is 0 at time             187448000
# test_dut:waiting for rx reset ack....
# test_dut:EHIP RX reset ack is 0 at time             187507467
# test_dut:Waiting for RX Block Lock
# dut:EHIP RX Block Lock  is high at time             189300083
# dut:Waiting for AM lock
# dut:EHIP RX AM Lock  is high at time            190566243
# dut:Waiting for RX alignment
# dut:RX deskew locked
# dut:RX lane aligmnent locked
# dut:**
# dut:** Testbench complete.
# dut:**
# dut:****************************************
# test_dut:EHIP RX Block Lock  is high at time             194839533
# test_dut:Waiting for AM lock
# test_dut:EHIP RX AM Lock  is high at time            196580503
# test_dut:Waiting for RX alignment
# test_dut:RX deskew locked
# test_dut:RX lane aligmnent locked
# dut:RX deskew locked
# dut:RX lane aligmnent locked
# test_dut:TX enabled
# test_dut: ** Sending Packet          1...
# test_dut: ** Sending Packet          2...
```

```
# test_dut: ** Sending Packet           3...
# test_dut: ** Sending Packet           4...
# test_dut: ** Sending Packet           5...
# test_dut: ** Sending Packet           6...
# test_dut: ** Sending Packet           7...
# test_dut: ** Sending Packet           8...
# test_dut: ** Sending Packet           9...
# test_dut: ** Received Packet          1...
# test_dut: ** Sending Packet          10...
# test_dut: ** Received Packet          2...
# test_dut: ** Received Packet          3...
# test_dut: ** Received Packet          4...
# test_dut: ** Received Packet          5...
# test_dut: ** Received Packet          6...
# test_dut: ** Received Packet          7...
# test_dut: ** Received Packet          8...
# test_dut: ** Received Packet          9...
# test_dut: ** Received Packet         10...
# test_dut:**
# test_dut:** Testbench complete.
# test_dut:**
# test_dut:****************************************
```

2.4. H-tile Hard IP for Ethernet Intel FPGA FlexE Simulation Design Example

Figure 9. H-tile Hard IP for Ethernet Intel FPGA 100-Gbps FlexE Simulation Design Example Block Diagram



The testbench sends traffic through the IP core with FlexE mode, exercising the transmit and receive PCS66 interfaces using a separate H-tile Hard IP for Ethernet Intel FPGA MAC as a stimulus generator. The PCS66 interface of the FlexE core is connected to a synchronous FIFO that receives PCS66 data and alignment marker valid signals. The FIFO then writes back to the PCS66 transmit interface,

For 50-Gbps datarate, the simulation design example instantiates an ATX PLL to drive the transceiver channel.

For 100-Gbps datarate, the simulation design example instantiates a main ATX PLL for transceiver channel 0 and 1, and a clock buffer for channel 2 and 3.

The testbench in this design example performs the following:

1. The client logic resets both the IP cores.
2. The stimulus client logic waits for the stimulus RX datapath and FlexE RX datapath to align.
3. Once alignment is complete, the stimulus client logic transmits a series of packets to the FlexE IP core.
4. The FlexE IP core receives the series of packets and transmits back to the stimulus MAC IP core.
5. The stimulus client logic then checks the number of packets received and verifies that the packets have no errors.

The following sample output illustrates a successful simulation test run for a 100-Gbps FlexE IP core variation. Times are in picoseconds. The times required for simulation of 100-Gbps variations are slightly longer than the times required for 50-Gbps variations.

```
# Ref clock is 644.53125 MHz
# Ref clock is 644.53125 MHz
# iatpg_pipeline_global_en is set
.
.
.
# iatpg_pipeline_global_en is set
# test_dut:waiting for o_tx_lanes_stable...
# dut:waiting for o_tx_lanes_stable...
# test_dut:o_tx_lanes_stable is 1 at time               525000
# test_dut:waiting for tx_dll_lock....
# dut:o_tx_lanes_stable is 1 at time               525000
# dut:waiting for tx_dll_lock....
# dut:TX DLL LOCK is 1 at time             47806703
# dut:waiting for tx_transfer_ready....
# dut:TX transfer ready is 1 at time             48126575
# dut:waiting for rx_transfer_ready....
# dut:RX transfer ready is 1 at time             59518683
# dut:EHIP PLD Ready out is 1 at time             59576000
# dut:EHIP reset out is 0 at time             59840000
# dut:EHIP reset ack is 0 at time             60151763
# dut:EHIP TX reset out is 0 at time             60488000
# dut:EHIP TX reset ack is 0 at time            110085115
# dut:waiting for EHIP Ready....
# dut:EHIP READY is 1 at time            110178411
# dut:EHIP RX reset out is 0 at time            110520000
# dut:waiting for rx reset ack....
# dut:EHIP RX reset ack is 0 at time            110578251
# dut:Waiting for RX Block Lock
# test_dut:TX DLL LOCK is 1 at time            124725923
# test_dut:waiting for tx_transfer_ready....
# test_dut:TX transfer ready is 1 at time             125045795
# test_dut:waiting for rx_transfer_ready....
# test_dut:RX transfer ready is 1 at time             136437903
# test_dut:EHIP PLD Ready out is 1 at time             136496000
# test_dut:EHIP reset out is 0 at time            136760000
# test_dut:EHIP reset ack is 0 at time            137070983
# test_dut:EHIP TX reset out is 0 at time             137408000
# test_dut:EHIP TX reset ack is 0 at time             187001003
# test_dut:waiting for EHIP Ready....
# test_dut:EHIP READY is 1 at time            187107627
# test_dut:EHIP RX reset out is 0 at time             187448000
# test_dut:waiting for rx reset ack....
# test_dut:EHIP RX reset ack is 0 at time             187507467
# test_dut:Waiting for RX Block Lock
# dut:EHIP RX Block Lock  is high at time             189300083
# dut:Waiting for AM lock
# dut:EHIP RX AM Lock  is high at time            190566243
# dut:Waiting for RX alignment
```

```
# dut:RX deskew locked
# dut:RX lane aligmnent locked
# dut:**
# dut:** Testbench complete.
# dut:**
# dut:****************************************
# test_dut:EHIP RX Block Lock  is high at time            195472613
# test_dut:Waiting for AM lock
# test_dut:EHIP RX AM Lock  is high at time            196580503
# test_dut:Waiting for RX alignment
# test_dut:RX deskew locked
# test_dut:RX lane aligmnent locked
# dut:RX deskew locked
# dut:RX lane aligmnent locked
# test_dut:TX enabled
# test_dut: ** Sending Packet          1...
# test_dut: ** Sending Packet          2...
# test_dut: ** Sending Packet          3...
# test_dut: ** Sending Packet          4...
# test_dut: ** Sending Packet          5...
# test_dut: ** Sending Packet          6...
# test_dut: ** Sending Packet          7...
# test_dut: ** Sending Packet          8...
# test_dut: ** Sending Packet          9...
# test_dut: ** Sending Packet         10...
# test_dut: ** Received Packet         1...
# test_dut: ** Received Packet         2...
# test_dut: ** Received Packet         3...
# test_dut: ** Received Packet         4...
# test_dut: ** Received Packet         5...
# test_dut: ** Received Packet         6...
# test_dut: ** Received Packet         7...
# test_dut: ** Received Packet         8...
# test_dut: ** Received Packet         9...
# test_dut: ** Received Packet        10...
# test_dut:**
# test_dut:** Testbench complete.
# test_dut:**
# test_dut:****************************************
```

2.5. Hardware Design Example Components

The H-tile Hard IP for Ethernet Intel FPGA hardware design example supports only the MAC + PCS variant.

Figure 10. Intel Stratix 10 GXT Transceiver Signal Integrity Development Kit Hardware Design Example High Level Block Diagram

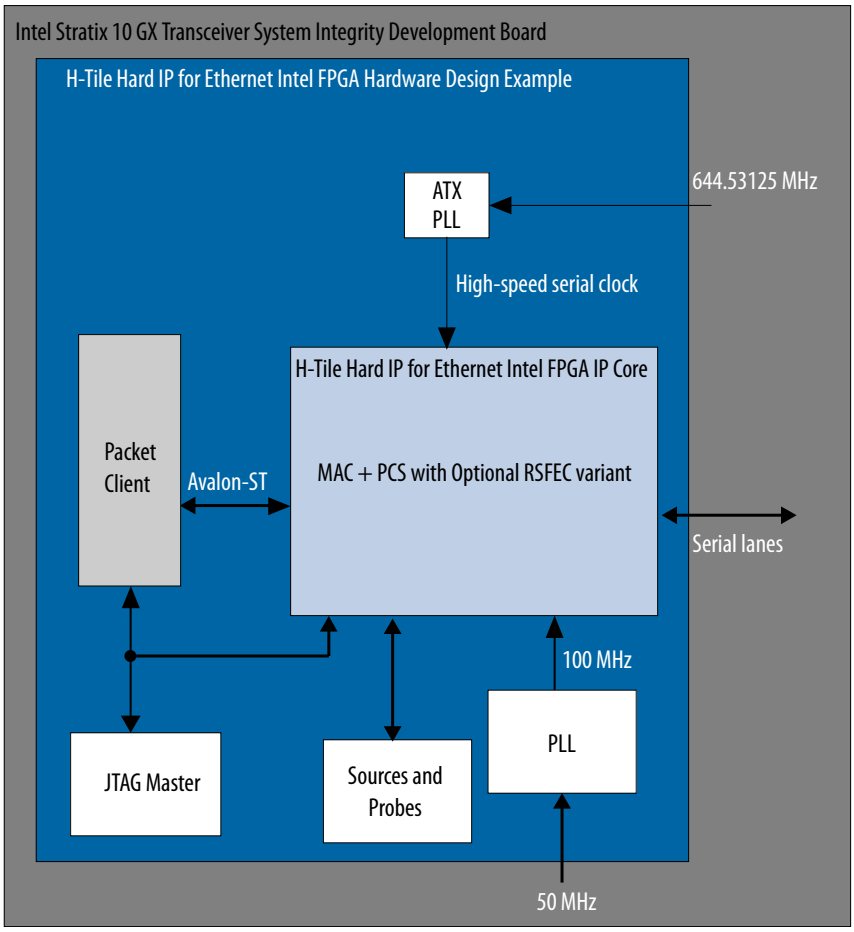
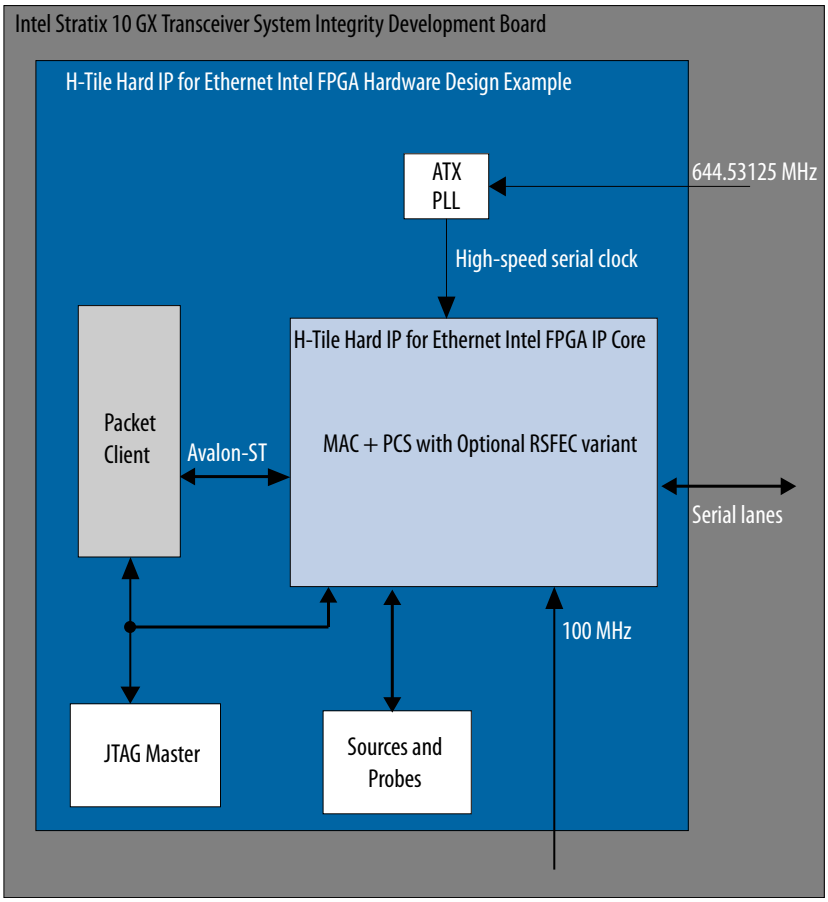


Figure 11. Intel Stratix 10 GXT Transceiver SOC Development Kit Hardware Design Example High Level Block Diagram



The H-tile Hard IP for Ethernet Intel FPGA hardware design example includes the following components:

- H-tile Hard IP for Ethernet Intel FPGA IP core.
- Client logic that coordinates the programming of the IP core and packet generation.
- One ATX PLL to generate the high speed serial clock to drive the device transceiver channels.
- For Intel Stratix 10 GXT Transceiver Signal Integrity Development Kit, the design example also generates an I/O PLL 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. The IP core receives the packets and transmit to the packet generator through the serial loopback. The client logic reads and print out the MAC statistic registers when the packet transmissions are complete.

Related Information

Intel Stratix 10 GX Signal Integrity Development Kit Webpage

2.6. Design Example Interface Signals

The H-tile Hard IP for Ethernet Intel FPGA testbench is self-contained and does not require you to drive any input signals.

Related Information

Interfaces and Signal Descriptions
Provides detailed descriptions of the H-tile Hard IP for Ethernet Intel FPGA IP core signals and the interfaces to which they belong.

2.7. H-tile Hard IP for Ethernet Intel FPGA Design Example Registers

Table 4. H-tile Hard IP for Ethernet Intel 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	RX PHY registers
0x000400	TX MAC registers
0x000500	RX MAC registers
0x000800	TX Statistics Counter registers
0x000900	RX Statistics Counter registers
0x001000	Packet Client registers

Table 5. Packet Client Registers

You can customize the H-tile Hard IP for Ethernet Intel 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.	0x25800040	RW

continued...

Addr	Name	Bit	Description	HW Reset Value	Access
			• 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 incremental bytes for a packet.		
0x1009	Packet Number Control	[31:0]	Specify the number of packets to transmit from the packet generator.	0xA	RW
0x1010	PKT_GEN_TX_CTRL	[7:0]	• 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. • Bit [5:4]: — 00: Random mode — 01: Fixed mode — 10: Incremental mode • Bit [6]: Set this bit to 1 to use 0x1009 register to turn off packet generator based on a fixed number of packets to transmit. Otherwise, bit [1] of PKT_GEN_TX_CTRL register is used to turn off the packet generator. • Bit [7]: — 1: For transmission without gap in between packets. — 0: For transmission with random gap in between packets.	0x6	RW
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 32bits	[31:0]	Source address (lower 32 bits)	0x43210ADD	RW
0x1014	Source address lower 16bits	[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

H-tile Hard IP for Ethernet Intel FPGA IP core register descriptions

3. H-Tile Hard IP for Ethernet Intel Stratix 10 FPGA IP Design Example User Guide Document Archives

If an IP core version is not listed, the user guide for the previous IP core version applies.

IP Core Version	User Guide
17.1	H-Tile Hard IP for Ethernet Intel Stratix 10 FPGA IP Design Example User Guide

4. Document Revision History for the H-Tile Hard IP for Ethernet Intel Stratix 10 FPGA IP Design Example User Guide

Document Version	Intel Quartus Prime Version	Changes
2018.08.10	18.0	• Added design example testbench components and test behavior for OTN and FlexE variations. • Added hardware design example components and test behavior. • Added hardware design example register description. • Rebranded the IP core name from Intel Stratix 10 H-Tile Hard IP for Ethernet IP core to H-tile Hard IP for Ethernet Intel FPGA per Intel rebranding.
2017.11.29	17.1	Initial release.