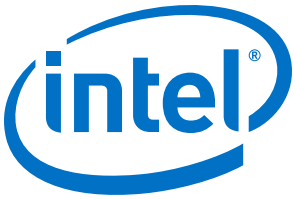


E-tile Hard IP for Ethernet Intel® Stratix® 10 FPGA IP Design Example User Guide

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

UG-20172 | 2018.08.10
Latest document on the web: **PDF** | **HTML**

Subscribe
Send Feedback

Contents

1. Quick Start Guide

The E-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 TX 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 simulation testbench and compilation-only example project for any E-Tile Hard IP for Ethernet Intel FPGA IP core variation as below

- Single channel 10Gigabit Ethernet (GE)/25GE Media Access Controller (MAC) + Physical Coding Sublayer (PCS)
- Single channel 10GE/25GE MAC+ PCS with optional Reed Solomon Forward Error Correction (RSFEC) and 1588 Precision Time Protocol (PTP)
- Single channel 100GE MAC+PCS
- Single channel 100GE MAC+PCS with optional RSFEC
- Single channel 100GE PCS_Only
- Single channel 100GE OTN
- Single channel 100GE FlexE

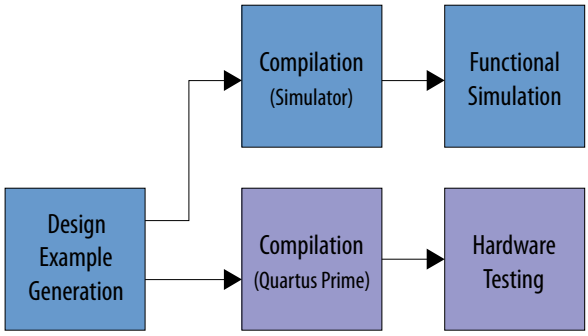
Note: The E-Tile Hard IP for Ethernet Intel FPGA IP provides preliminary support for the OTN feature. For further inquiries, contact your nearest Intel sales representative or file an Intel Premier Support (IPS) case on https://www.intel.com/content/www/us/en/programmable/my-intel/mal-home.html.

You can generate the E-Tile Hard IP for Ethernet Intel FPGA hardware design example with the variations below:

- Single channel 10GE/25GE MAC+PCS
- Single channel 10GE/25GE MAC+PCS with optional RSFEC and optional PTP
- Single channel 100GE MAC+PCS with optional RSFEC

Figure 1. Development Steps for the Design Example

The compilation-only example project cannot be configured in hardware.



1.1. Directory Structure

The E-Tile Hard IP for Ethernet Intel FPGA design example file directories contain the following generated files for the design examples.

Figure 2. E-Tile Hard IP for Ethernet Intel FPGA 10/25-GE with Optional RSFEC and Optional PTPDesign Example Directory Structure

<datarate> is either "10" or "25", depending on your IP core variation.

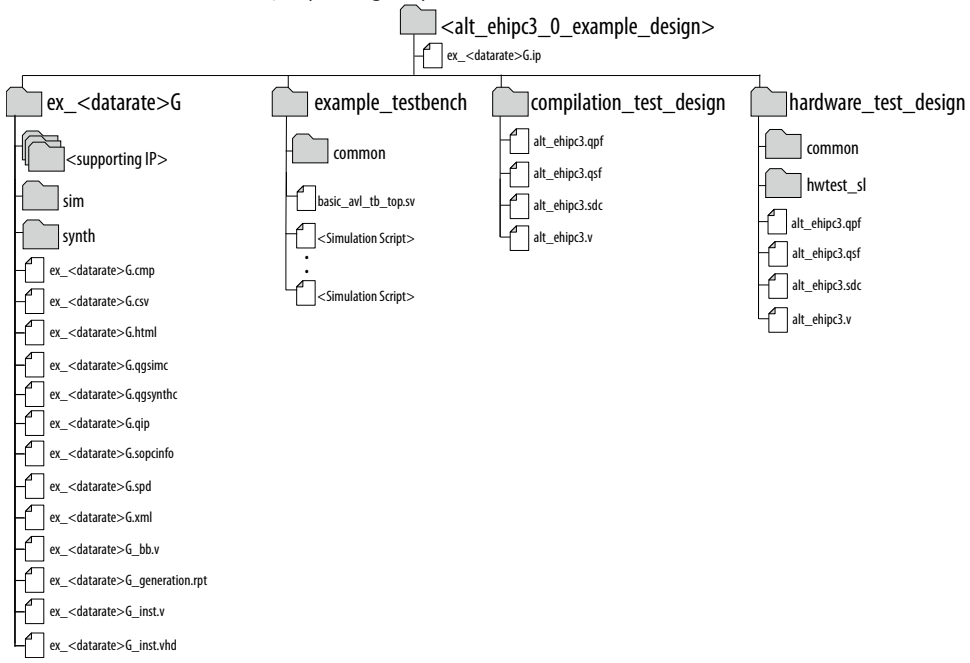


Figure 3. E-Tile Hard IP for Ethernet Intel FPGA 100GE with Optional RSFEC Design Example Directory Structure

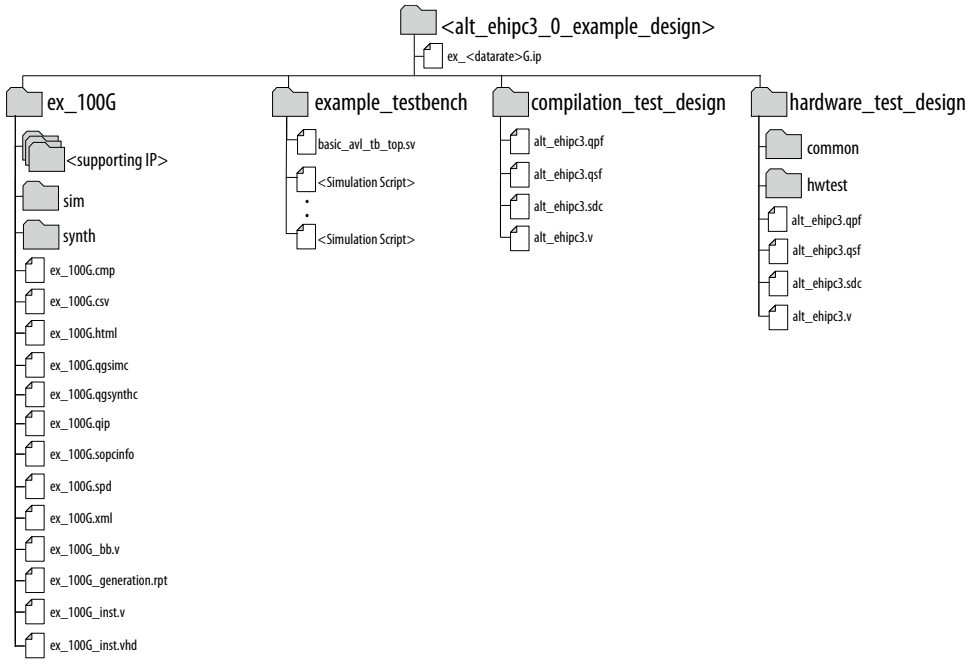


Table 1. E-Tile Hard IP for Ethernet Intel FPGA IP Core Testbench File Descriptions

File Names	Description
Key Testbench and Simulation Files	
<design_example_dir>/example_testbench/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	
<design_example_dir>/example_testbench/run_vsim.do	The Mentor Graphics ModelSim* script to run the testbench.
<design_example_dir>/example_testbench/run_vcs.sh	The Synopsys VCS* script to run the Verilog testbench.
<design_example_dir>/example_testbench/run_vcs.sh (VHDL)	The Synopsys VCS script to run the VHDL testbench.
<design_example_dir>/example_testbench/run_ncsim.sh	The Cadence NCSim* script to run the testbench.

Table 2. IP Core Hardware Design Example File Descriptions

File Names	Description
<design_example_dir>/hardware_test_design/alt_ehipc3.qpf	Intel Quartus® Prime project file
<design_example_dir>/hardware_test_design/alt_ehipc3.qsf	Intel Quartus Prime project settings file
<design_example_dir>/hardware_test_design/alt_ehipc3.sdc	Synopsys Design Constraints files. You can copy and modify these files for your own design.

continued...

File Names	Description
<design_example_dir>/hardware_test_design/alt_ehipc3.v	Top-level Verilog HDL design example file
<design_example_dir>/hardware_test_design/common/	Hardware design example support files
hwtest_sl/main_script.tcl (10GE/25GE with optional RSFEC) hwtest/main.tcl (100GE)	Main file for accessing System Console

1.2. Generating the Design

Figure 4. Procedure

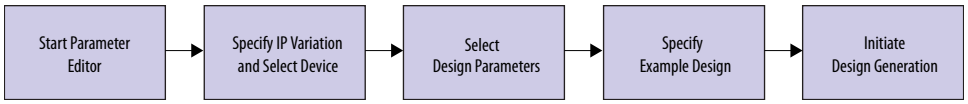
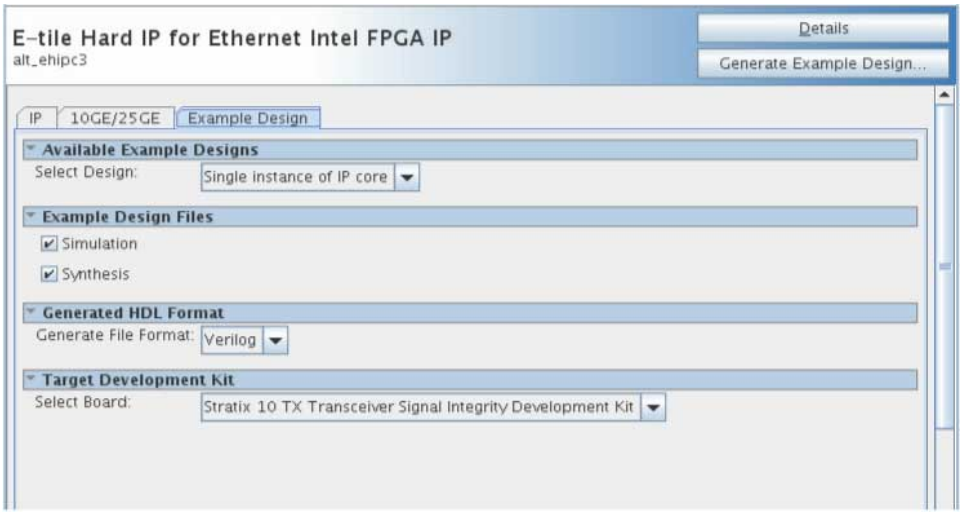


Figure 5. Example Design Tab in the E-Tile Hard IP for Ethernet Intel FPGA Parameter Editor



If you do not already have an Intel Quartus Prime Pro Edition project in which to integrate your E-Tile Hard IP for Ethernet Intel FPGA IP core, you must create one.

1. 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.
2. Specify the device family **Intel Stratix 10** and select a device that meets all of these requirements:
   - Transceiver tile is E-tile
   - Transceiver speed grade is 1 or 2
   - Core speed grade is 1 or 2
3. Click **Finish**.

Follow these steps to generate the E-Tile Hard IP for Ethernet Intel FPGA hardware design example and testbench:

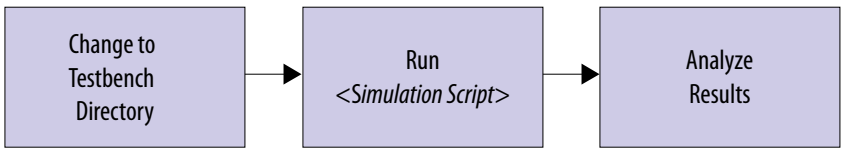
1. In the IP Catalog, locate and select **E-Tile Hard IP for Ethernet Intel FPGA**. The **New IP Variation** window appears.
2. Specify a top-level name <*your_ip*> for your custom IP variation. The parameter editor saves the IP variation settings in a file named <*your_ip*>.ip.
3. Click **OK**. The parameter editor appears.
4. On the **IP**, 100GE, or 10GE/25GE tabs, specify the parameters for your IP core variation.
5. 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. You must select at least one of the **Simulation** and **Synthesis** options to generate the design example.
6. On the **Example Design** tab, under **Generated HDL Format**, select **Verilog** HDL or **VHDL**. If you select **VHDL**, you must simulate the testbench with a mixed-language simulator. The device under test in the ex_<datarate> directory is a VHDL model, but the main testbench file is a System Verilog file.
7. Under **Target Development Kit**, select the **Stratix 10 TX Transceiver Signal Integrity Development Kit** or select **None**. The compilation-only and hardware design examples target your project device. For the hardware design to functional correctly, you must ensure your project device is the same device on your development kit.
8. Click the **Generate Example Design** button. The **Select Example Design Directory** window appears.
9. If you want to modify the design example directory path or name from the defaults displayed (alt_ehipc3_0_example_design), browse to the new path and type the new design example directory name (<*design_example_dir*>).

Related Information

- E-Tile Hard IP for Ethernet Intel FPGA IP Core Parameters
  Provides more information about customizing your IP core.
- Intel Stratix 10 TX Signal Integrity Development Kit Webpage

1.3. Simulating the E-Tile Hard IP for Ethernet Intel FPGA Design Example Testbench

Figure 6. 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` (Verilog) or `sh run_vcsmx.sh` (VHDL)

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_ehipc3.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.
2. In the Intel Quartus Prime Pro Edition software, open the Intel Quartus Prime project <*design_example_dir*>/hardware_test_design/alt_ehip3.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 E-Tile Hard IP for Ethernet Intel FPGA Hardware Design Example

After you compile the E-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.

1.6.1. 10GE/25GE with Optional RSFEC Hardware Design Example

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_sl` to change directory to `<design_example_dir>/hardware_test_design/hwtest_sl`.
3. Type `source main_script.tcl` to open a connection to the JTAG master.

   Note: 25GE variant only supports random packet size transfer in Intel Quartus Prime version 18.0.

Related Information

Analyzing and Debugging Designs with System Console

1.6.2. 100GE with Optional RSFEC Hardware Design Example

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 for 100GE design example:

   - `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*>.

Related Information

Analyzing and Debugging Designs with System Console

2. 10GE/25GE with Optional RSFEC Design Examples

The 10GE/25GE design example demonstrates an Ethernet solution for Intel Stratix 10 devices using the E-Tile Hard IP for Ethernet Intel FPGA IP core with the following variants:

Number of Channel	Data Rate	Feature	Available Design Example
1	10GE	MAC + PCS	Simulation, compilation-only project, and hardware design example
		MAC + PCS + PTP	Simulation, compilation-only project, and hardware design example
	25GE	MAC + PCS	Simulation, compilation-only project, and hardware design example
		MAC + PCS + RSFEC	Simulation, compilation-only project, and hardware design example
		MAC + PCS + PTP	Simulation, compilation-only project, and hardware design example
		MAC + PCS + RSFEC + PTP	Simulation, compilation-only project, and hardware design example

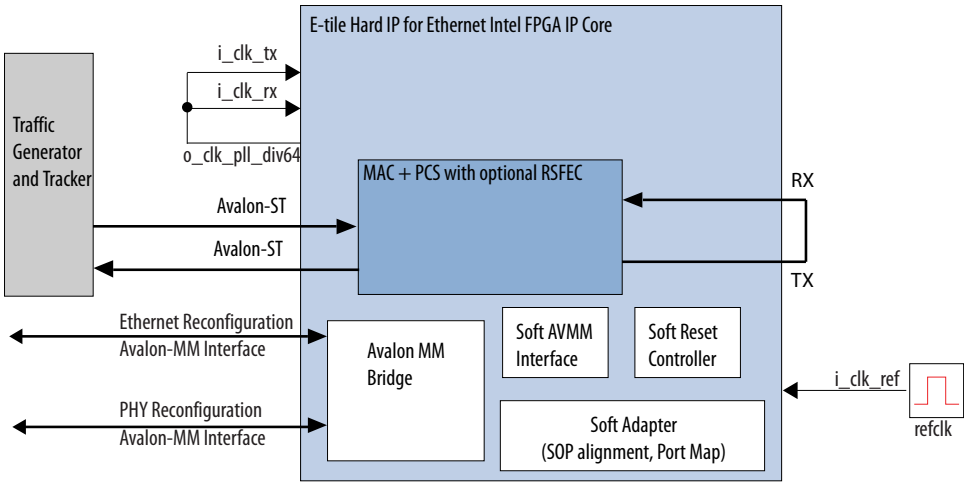
2.1. Non-PTP 10GE/25GE with Optional RSFEC Simulation Design Example

To generate this design example, choose the following settings in your IP parameter editor:

1. **1 to 4 10GE/25GE with optional RSFEC** or **100GE or 1 to 4 channel 10GE/25GE with optional RSFEC and PTP** as the core variant.
2. **10GE/25GE Channel(s)** as **Active channel(s) at startup** if you choose **100GE or 1 to 4 channel 10GE/25GE with optional RSFEC and PTP** as the core variant.
3. **Enable RSFEC** to use RSFEC feature.
4. **10G** or **25G** as the Ethernet rate.

   Note: RSFEC is not supported in 10GE variant.

Figure 7. Non-PTP E-Tile Hard IP for Ethernet Intel FPGA 10GE/25GE with Optional RSFEC Variations Simulation Design Example Block Diagram



The testbench sends traffic through the IP core, exercising the transmit side and receive side of the IP core. For 25GE variant, the testbench only supports random packet size transfer.

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.

The successful test run displays output confirming the following behavior:

1. Waiting for PLL to lock.
2. Waiting for RX transceiver reset to complete.
3. Waiting for RX alignment.
4. Sending 10 packets.
5. Receiving those packets.
6. Displaying `Testbench complete.`

The following sample output illustrates a successful simulation test run for a 10GE, MAC+PCS, non-PTP IP core variation.

```
# Ref clock is 322.265625 MHz
# waiting for EHIP Ready....
# EHIP READY is 1 at time             274765883
# Waiting for RX Block Lock
# EHIP RX Block Lock  is high at time             278880903
# Waiting for RX alignment
# RX deskew locked
# RX lane aligmnent locked
# TX enabled
# ** Packet Generator Configuration
# ** Address offset = 00000, WriteData = 0000000a
# ** Address offset = 00001, WriteData = 00000000
# ** Address offset = 00002, WriteData = 00000000
# ** Address offset = 00009, WriteData = 0000000a
# ** Address offset = 0000d, WriteData = 00000040
# ** Address offset = 00003, WriteData = 00000001
# ** INFO : Sending Packet            1...
```

```
# ** INFO : Sending Packet            2...
# ** INFO : Sending Packet            3...
# ** INFO : Sending Packet            4...
# ** INFO : Sending Packet            5...
# ** INFO : Sending Packet            6...
# ** INFO : Sending Packet            7...
# ** INFO : Sending Packet            8...
# ** INFO : Sending Packet            9...
# ** INFO : Sending Packet           10...
# ** INFO : Received Packet           1...
# ** INFO : Received Packet           2...
# ** INFO : Received Packet           3...
# ** INFO : Received Packet           4...
# ** INFO : Received Packet           5...
# ** INFO : Received Packet           6...
# ** INFO : Received Packet           7...
# ** INFO : Received Packet           8...
# ** INFO : Received Packet           9...
# ** INFO : Received Packet          10...
# **
# ** AVMM access CSR registers read/write check
# ** Address offset = 00301, WriteData = c3ec3ec3
# ** Address offset = 00301, ReadData  = c3ec3ec3
# ** Address offset = 00301, WriteData = 00000000
# ** Address offset = 00300, ReadData  = 11112015
# ** Address offset = 00400, ReadData  = 11112015
# ** Address offset = 00a00, ReadData  = 11112015
# ** Address offset = 00b00, ReadData  = 11112015
# ** Address offset = 00836, ReadData  = 0000000a
# ** Address offset = 00936, ReadData  = 0000000a
# ** Address offset = 00804, ReadData  = 00000000
# ** Address offset = 00904, ReadData  = 00000000
# ** Address offset = 00322, ReadData  = 00000001
# ** Address offset = 00084, ReadData  = xxxxxx00
# ** Address offset = 00084, WriteData = ffffffff
# ** Address offset = 00084, ReadData  = xxxxxxff
# ** Address offset = 00084, WriteData = 00000000
# ** Address offset = 00230, WriteData = ffffffff
# ** Address offset = 00230, ReadData  = xxxxxxff
# ** Address offset = 00230, WriteData = 0000007b
# **
# ** Testbench complete.
# **
# ****************************************
# ** Note: $finish    : ./basic_avl_tb_top.sv(383)
#    Time: 298225 ns  Iteration: 0  Instance: /basic_avl_tb_top
```

Related Information

Simulating the E-Tile Hard IP for Ethernet Intel FPGA Design Example Testbench on page 7

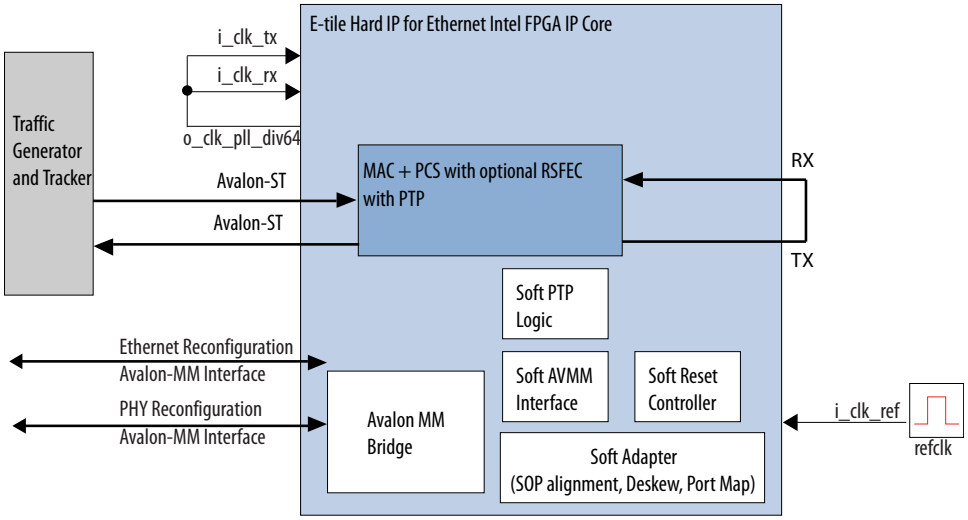
2.2. 10GE/25GE with Optional RSFEC and PTP Simulation Design Example

To generate this design example, choose the following settings in your IP parameter editor:

1. **100GE or 1 to 4 channel 10GE/25GE with optional RSFEC and PTP** as the core variant.
2. **10G/25GE channels** as **Active channel(s) at startup**.
3. **Enable IEEE 1588 PTP**.
4. **Enable RSFEC** to use RSFEC feature.
5. **10G** or **25G** as the Ethernet rate.

   Note: RSFEC is not supported in 10GE variant.

Figure 8. E-Tile Hard IP for Ethernet Intel FPGA 10GE/25GE with Optional RSFEC with PTP Variations Simulation Design Example Block Diagram



In this design example, the testbench sends traffic through the IP core, exercising the transmit side and receive side of the IP core. For 25GE variant, the testbench only supports random packet size transfer.

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.

The successful test run displays output confirming the following behavior:

1. Waiting for PLL to lock.
2. Waiting for RX transceiver reset to complete.
3. Waiting for RX alignment.
4. Sending 10 packets.
5. Receiving those packets.
6. Displaying `Testbench complete.`

The following sample output illustrates a successful simulation test run for a 10GE, MAC+PCS, PTP IP core variation.

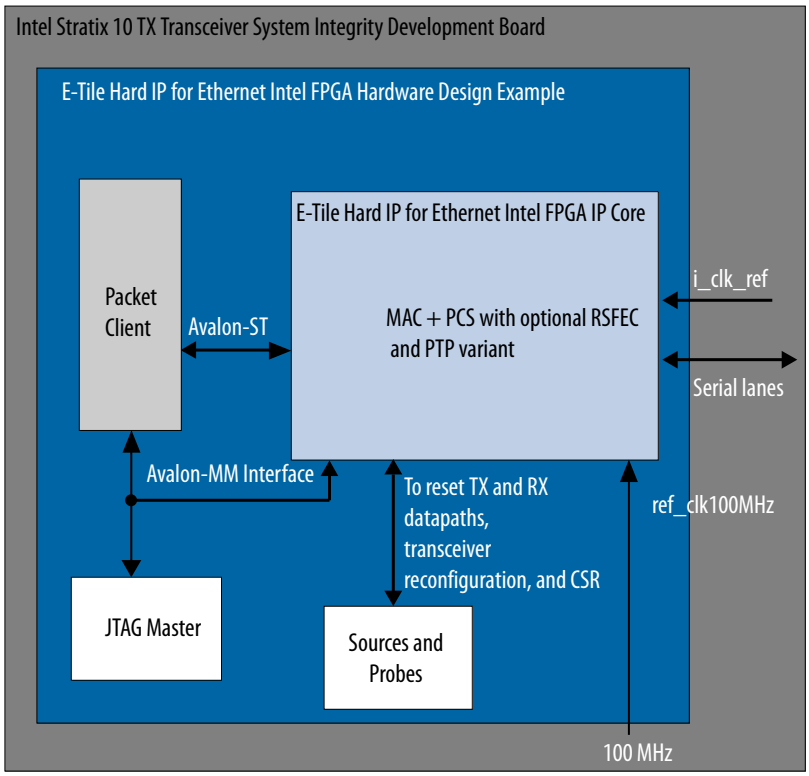
```
# EHIP READY is 1 at time             270492593
# Waiting for RX Block Lock
# EHIP RX Block Lock  is high at time             274449343
# Waiting for RX alignment
# RX deskew locked
# RX lane aligmnent locked
# Waiting for PTP Ready
# PTP Ready
# Configure TX extra latency
# ====> writedata = 0004267a
#
# Configure RX extra latency
# ====> writedata = 80054670
#
# TX enabled
# ** 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...
# ** Received Packet           1...
# ** Sending Packet           10...
# ** 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.
# **
# ****************************************
```

Related Information

Simulating the E-Tile Hard IP for Ethernet Intel FPGA Design Example Testbench on page 7

2.3. 10GE/25GE with Optional RSFEC and PTP Hardware Design Example Components

Figure 9. 10GE/25GE with Optional RSFEC and PTP Hardware Design Examples High Level Block Diagram



The E-Tile Hard IP for Ethernet Intel FPGA hardware design examples include the following components:

- E-Tile Hard IP for Ethernet Intel FPGA IP core.
- Client logic that coordinates the programming of the IP core and packet generation.
- JTAG controller that communicates with the System Console. You communicate with the client logic through the System Console.

The following sample outputs illustrate a successful hardware test run for a 10GE, MAC+PCS, non-PTP IP core variation. The test results are located at `<design_example_dir>/hardware_test_design/hwtest_sl/c3_elane_xcvr_loopback_test_log` or `<design_example_dir>/hardware_test_design/hwtest_sl/c3_elane_traffic_basic_test_log`.

Result from `c3_elane_xcvr_loopback_test_log` file:

```
Info: Set JTAG Master Service Path

Info: Opened JTAG Master Service

    Test Start time is: 12:33:49
    Test Start date is: 05/02/2018
```

```
Info: Read all ELANE CSR registers

    Successfully Read  EHIPLANE User Register phy_revid     , offset = 0x300,
data = 0x11112015
    Successfully Read  EHIPLANE User Register phy_scratch   , offset = 0x301,
data = 0x0
.
.
.

    Successfully Read  EHIPLANE User Register cntr_rx_badlt_hi , offset =
0x969, data = 0x0

    Test End time is: 12:33:54
    Test End date is: 05/02/2018

Info: Closed JTAG Master Service


Info: Test <c3_elane_traffic_basic_test> Passed
```

Result from `c3_elane_traffic_basic_test_log` file:

```
Info: Set JTAG Master Service Path


Info: Opened JTAG Master Service

    Test Start time is: 18:34:19
    Test Start date is: 04/24/2018


    Successfully Write EHIPLANE User Register phy_ehip_csr_soft_reset ,
offset = 0x310, data = 0x0
    Successfully Write EHIPLANE User Register phy_ehip_csr_soft_reset ,
offset = 0x310, data = 0x1
.
.
.
    Successfully Read  EHIPLANE Channel 0, User Register
phy_ehip_csr_soft_reset   , offset = 0x310, data = 0x0

    C3 ELANE Channel 0 System Reset is successfully

Info: Read all ELANE CSR registers

    Successfully Read  EHIPLANE User Register phy_revid  , offset = 0x300,
data = 0x11112015
.
.
.
    Successfully Read  EHIPLANE User Register rx_ptp_dp_latency , offset =
0xb07, data = 0x0

Info: Stopping the traffic generator

    Successfully Write EHIPLANE Traffic GEN/CHK Register tg_stop , offset =
0x4, data = 0x0

Info: clearing the statistics

    Successfully Write EHIPLANE User Register cntr_tx_config , offset =
0x845, data = 0x1
    Successfully Write EHIPLANE User Register cntr_rx_config , offset =
0x945, data = 0x1

    Successfully Write EHIPLANE Channel 0, Traffic GEN/CHK Register
pkt_end_addr_start_addr , offset = 0x8, data = 0x25800040
    Successfully Write EHIPLANE Channel 0, Traffic GEN/CHK Register
```

```
pkt_tx_num   , offset = 0x9, data = 0xa

Info: Enabling the statistics

    Successfully Write EHIPLANE User Register cntr_tx_config , offset =
0x845, data = 0x0
.
.
.
     Successfully Write EHIPLANE Traffic GEN/CHK Register
tm_sticky_rx_err_reg , offset = 0x10b, data = 0x0

    Traffic Generator Configuration <traffic_gen_config> completed
successfully

    Successfully Write EHIPLANE Traffic GEN/CHK Register tm_mon_csr , offset
= 0x107, data = 0x1
    Successfully Write EHIPLANE Traffic GEN/CHK Register tm_mon_csr , offset
= 0x107, data = 0x0

Info: Stopping the traffic generator

    Successfully Write EHIPLANE Traffic GEN/CHK Register tg_start  , offset =
0x3, data = 0x1
.
.
.
     Successfully Read  EHIPLANE User Register cntr_tx_bcast_data_ok_lo  ,
offset = 0x828, data = 0xc
.
.
.
     Successfully Read  EHIPLANE User Register cntr_rx_bcast_data_ok_lo ,
offset = 0x928, data = 0xc
.
.
.
     Successfully Read  EHIPLANE Traffic GEN/CHK Register tg_num_packets ,
offset = 0x0, data = 0xc

    Test End time is: 18:34:20
    Test End date is: 04/24/2018

Info: Closed JTAG Master Service



Info: Test <c3_elane_traffic_basic_test> Passed
```

Related Information

- Intel Stratix 10 TX Signal Integrity Development Kit Webpage
- Compiling and Configuring the Design Example in Hardware on page 8
- Testing the E-Tile Hard IP for Ethernet Intel FPGA Hardware Design Example on page 9

2.4. 10GE/25GE with Optional RSFEC and PTP Design Example Interface Signals

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

Table 4. 10GE/25GE with Optional RSFEC and PTP Hardware Design Example Interface Signals

Signal	Direction	Description
`ref_clk100MHz`	Input	Drive at 100 to 161.13 MHz. Input clock for CSR access on all the AVMM interfaces.
`i_clk_ref`	Input	Drive at 322.265625 MHz.
`o_clk_pll_div64`	Output	Output 402.83203125 MHz +/- 100ppm.
`o_tx_serial[3:0]`	Output	Transceiver PHY output serial data.
`i_rx_serial[3:0]`	Input	Transceiver PHY input serial data.

Related Information

E-Tile Hard IP for Ethernet Intel FPGA Interfaces and Signal Descriptions

2.5. 10GE/25GE with Optional RSFEC Design Examples Registers

Table 5. E-Tile Hard IP for Ethernet Intel FPGA Hardware Design Examples Register Map

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

Word Offset	Register Type
0x00000000	MAC and PCS registers
0x00100000	PHY reconfiguration registers
0x00200000	Frame generator and checker registers
0x00300000	RSFEC configuration registers

Related Information

E-Tile Hard IP for Ethernet Intel FPGA IP core register descriptions

3. 100GE with Optional RSFEC Design Example

The 100GE design example demonstrates an Ethernet solution for Intel Stratix 10 devices using the E-Tile Hard IP for Ethernet Intel FPGA IP core with the following variants:

Number of Channel	Data Rate	Feature	Available Design Example
1	100GE	MAC + PCS	Simulation, compilation-only project, and hardware design example
		MAC + PCS + RSFEC	Simulation, compilation-only project, and hardware design example
		PCS Only	Simulation and compilation-only project
		OTN	Simulation and compilation-only project
		FlexE	Simulation and compilation-only project

Note: The E-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.

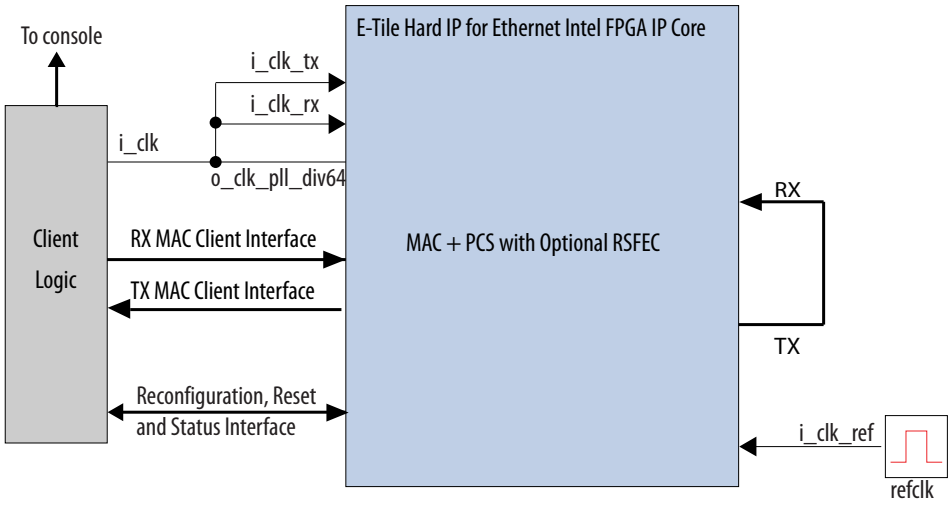
3.1. E-Tile Hard IP for Ethernet Intel FPGA 100GE MAC + PCS with Optional RSFEC Simulation Design Example

To generate this design example, choose the following settings in your IP parameter editor:

1. **Single 100GE with optional RSFEC** or **100GE or 1 to 4 channel 10GE/25GE with optional RSFEC and PTP** as the core variant.
2. **100GE Channel** as **Active channel(s) at startup** if you choose **100GE or 1 to 4 channel 10GE/25GE with optional RSFEC and PTP** as the core variant.
3. **Enable RSFEC** to use RSFEC feature.

   Note: The RSFEC feature is only available when you select **100GE or 1 to 4 channel 10GE/25GE with optional RSFEC and PTP** as the core variant.
4. **100G** as the Ethernet rate.
5. **MAC + PCS** as **Select Ethernet IP Layers** to use instantiate MAC and PCS layer or **MAC+PCS+(528,514)RSFEC** to instantiate MAC and PCS with RSFEC feature.

Figure 10. E-Tile Hard IP for Ethernet Intel FPGA 100GE MAC + PCS with Optional RSFEC Simulation Design Example Block Diagram



The testbench sends traffic through the IP core, exercising the transmit side and receive side of the IP core.

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.

The successful test run displays output confirming the following behavior:

1. The client logic resets the IP core.
2. 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 verify that the data matches with the transmitted packets.
6. Displaying `Testbench complete.`

The following sample output illustrates a successful simulation test run for a 100GE, MAC+PCS with optional RSFEC IP core variation.

```
# o_tx_lanes_stable is 1 at time             345651500
# waiting for tx_dll_lock....
# TX DLL LOCK is 1 at time             398849563
# waiting for tx_transfer_ready....
# TX transfer ready is 1 at time             399169435
# waiting for rx_transfer_ready....
# RX transfer ready is 1 at time             410719813
# EHIP PLD Ready out is 1 at time             410776000
# EHIP reset out is 0 at time            411040000
# EHIP reset ack is 0 at time            412282101
# EHIP TX reset out is 0 at time             413160000
# EHIP TX reset ack is 0 at time             462643731
# waiting for EHIP Ready....
# EHIP READY is 1 at time            462750387
# EHIP RX reset out is 0 at time             463088000
```

```
# waiting for rx reset ack....
# EHIP RX reset ack is 0 at time             463283667
# Waiting for RX Block Lock
# EHIP RX Block Lock  is high at time             467376591
# Waiting for AM lock
# EHIP RX AM Lock  is high at time              468643131
# Waiting for RX alignment
# RX deskew locked
# RX lane aligmnent locked
# ** Sending Packet           1...
# ** Sending Packet           2...
# ** Sending Packet           3...
# ** Sending Packet           4...
# ** Sending Packet           5...
# ** Sending Packet           6...
# ** Sending Packet           7...
# ** Received Packet          1...
# ** Sending Packet           8...
# ** Received Packet          2...
# ** Sending Packet           9...
# ** Received Packet          3...
# ** Received Packet          4...
# ** Sending Packet          10...
# ** Received Packet          5...
# ** Received Packet          6...
# ** Received Packet          7...
# ** Received Packet          8...
# ** Received Packet          9...
# ** Received Packet         10...
# ====>MATCH!     ReaddataValid = 1 Readdata = 11112015 Expected_Readdata =
11112015
#
# ====> writedata = ffff0000
#
# ====>MATCH!     ReaddataValid = 1 Readdata = 11112015 Expected_Readdata =
11112015
#
# ====> writedata = 4321abcd
#
# ====>MATCH!     ReaddataValid = 1 Readdata = 4321abcd Expected_Readdata =
4321abcd
#
# ====> writedata = a5a51234
#
# ====>MATCH!     ReaddataValid = 1 Readdata = a5a51234 Expected_Readdata =
a5a51234
#
# ====> writedata = abcda5a5
#
# ====>MATCH!     ReaddataValid = 1 Readdata = abcda5a5 Expected_Readdata =
abcda5a5
#
# ====> writedata = 4321abcd
#
# ====>MATCH!     ReaddataValid = 1 Readdata = 4321abcd Expected_Readdata =
4321abcd
#
# ====> writedata = a5a51234
#
# ====>MATCH!     ReaddataValid = 1 Readdata = a5a51234 Expected_Readdata =
a5a51234
#
# ====> writedata = abcda5a5
#
# ====>MATCH!     ReaddataValid = 1 Readdata = abcda5a5 Expected_Readdata =
abcda5a5
#
# TX enabled
# **
```

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

Related Information

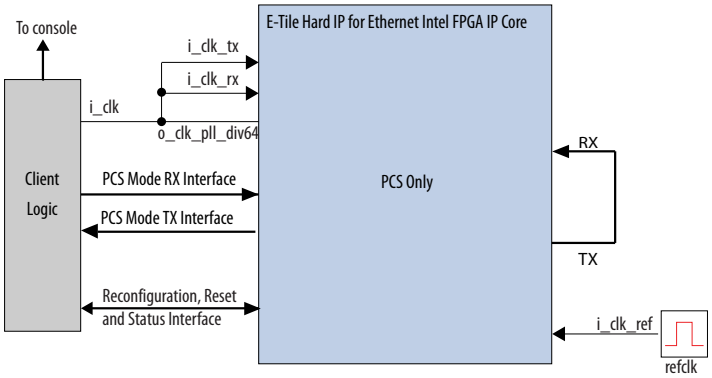
Simulating the E-Tile Hard IP for Ethernet Intel FPGA Design Example Testbench on page 7

3.2. E-Tile Hard IP for Ethernet Intel FPGA 100GE PCS Only Simulation Design Example

To generate this design example, choose the following settings in your IP parameter editor:

1. **Single 100GE with optional RSFEC** or **100GE or 1 to 4 channel 10GE/25GE with optional RSFEC and PTP** as the core variant.
2. **100GE Channel** as **Active channel(s) at startup** if you choose **100GE or 1 to 4 channel 10GE/25GE with optional RSFEC and PTP** as the core variant.
3. **100G** as the Ethernet rate.
4. **PCS_Only** as the Ethernet IP layer.

Figure 11. E-Tile Hard IP for Ethernet Intel FPGA 100GE PCS Only Simulation Design Example Block Diagram



The testbench sends traffic through the IP core, exercising the transmit side and receive side of the IP core.

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.

The successful test run displays output confirming the following behavior:

1. The client logic resets the IP core.
2. 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.
7. Displaying `Testbench complete.`

The following sample output illustrates a successful simulation test run for a 100GE, PCS only IP core variation.

```
# o_tx_lanes_stable is 1 at time             345685000
# waiting for tx_dll_lock....
# TX DLL LOCK is 1 at time             398849563
# waiting for tx_transfer_ready....
# TX transfer ready is 1 at time             399169435
# waiting for rx_transfer_ready....
# RX transfer ready is 1 at time             410719813
# EHIP PLD Ready out is 1 at time             410776000
# EHIP reset out is 0 at time            411040000
# EHIP reset ack is 0 at time            412282101
# EHIP TX reset out is 0 at time             413160000
# EHIP TX reset ack is 0 at time             462643731
# waiting for EHIP Ready....
# EHIP READY is 1 at time            462750387
# EHIP RX reset out is 0 at time             463088000
# waiting for rx reset ack....
# EHIP RX reset ack is 0 at time             463283667
# Waiting for RX Block Lock
# EHIP Rx Block Lock  is high at time             466743321
# Waiting for AM lock
# EHIP Rx am Lock  is high at time             468009861
# 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.
# **
# ****************************************
```

Related Information

Simulating the E-Tile Hard IP for Ethernet Intel FPGA Design Example Testbench on page 7

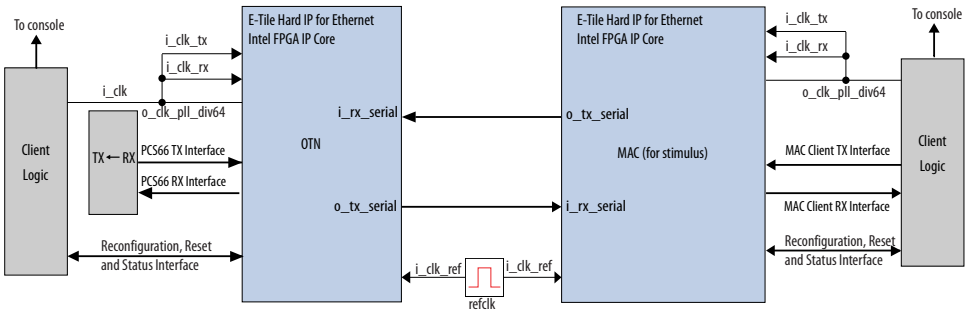
3.3. E-Tile Hard IP for Ethernet Intel FPGA 100GE OTN Simulation Design Example

To generate this design example, choose the following settings in your IP parameter editor:

1. **Single 100GE with optional RSFEC** or **100GE or 1 to 4 channel 10GE/25GE with optional RSFEC and PTP** as the core variant.
2. **100GE Channel** as **Active channel(s) at startup** if you choose **100GE or 1 to 4 channel 10GE/25GE with optional RSFEC and PTP** as the core variant.
3. **100G** as the Ethernet rate.
4. **OTN** as the Ethernet IP layer.

Note: The E-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 12. E-Tile Hard IP for Ethernet Intel FPGA 100GE OTN Simulation Design Example Block Diagram



The testbench sends traffic through the IP core with OTN mode, exercising the transmit side and receive interface using a separate E-Tile Hard IP for Ethernet Intel FPGA MAC as a stimulus generator.

The successful test run displays output confirming the following behavior:

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 verify that the packets have no errors.
6. Displaying `Testbench complete.`

The following sample output illustrates a successful simulation test run for a 100GE OTN IP core variation.

```
# test_dut: def_100G_o_tx_lanes_stable is 1 at time            345685000
# test_dut: waiting for tx_dll_lock....
# dut: o_tx_lanes_stable is 1 at time            345685000
# dut: waiting for tx_dll_lock....
# dut: TX DLL LOCK is 1 at time            398849563
# dut: waiting for tx_transfer_ready....
# dut: TX transfer ready is 1 at time            399169435
# dut: waiting for rx_transfer_ready....
# dut: RX transfer ready is 1 at time            410719813
# dut: EHIP PLD Ready out is 1 at time            410776000
# dut: EHIP reset out is 0 at time            411040000
# dut: EHIP reset ack is 0 at time            412282101
# dut: EHIP TX reset out is 0 at time            413160000
# dut: EHIP TX reset ack is 0 at time            462643731
# dut: waiting for EHIP Ready....
# dut: EHIP READY is 1 at time            462750387
# dut: EHIP RX reset out is 0 at time            463088000
# dut: waiting for rx reset ack....
# dut: EHIP RX reset ack is 0 at time            463283667
# dut: Waiting for RX Block Lock
# test_dut: TX DLL LOCK is 1 at time            475452243
# test_dut: waiting for tx_transfer_ready....
# test_dut: TX transfer ready is 1 at time            475772115
# test_dut: waiting for rx_transfer_ready....
# test_dut: RX transfer ready is 1 at time            487164223
# test_dut: EHIP PLD Ready out is 1 at time            487224000
# test_dut: EHIP reset out is 0 at time            487488000
# test_dut: EHIP reset ack is 0 at time            488907771
# test_dut: EHIP TX reset out is 0 at time            489784000
# test_dut: EHIP TX reset ack is 0 at time            539116083
# test_dut: waiting for EHIP Ready....
# test_dut: EHIP READY is 1 at time            539169411
# test_dut: EHIP RX reset out is 0 at time            539512000
# test_dut: waiting for rx reset ack....
# test_dut: EHIP RX reset ack is 0 at time            539702691
# test_dut: Waiting for RX Block Lock
# dut: EHIP RX Block Lock  is high at time            542102451
# dut: Waiting for AM lock
# test_dut: EHIP RX Block Lock  is high at time            542735721
# test_dut: Waiting for AM lock
# dut: EHIP RX AM Lock  is high at time            543368991
# dut: Waiting for RX alignment
# dut: RX deskew locked
# dut: RX lane aligmnent locked
# dut: ***************************************
# test_dut: EHIP RX AM Lock  is high at time            549068421
# test_dut: Waiting for RX alignment
# test_dut: RX deskew locked
# test_dut: RX lane aligmnent locked
```

```
# 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: ****************************************
```

Related Information

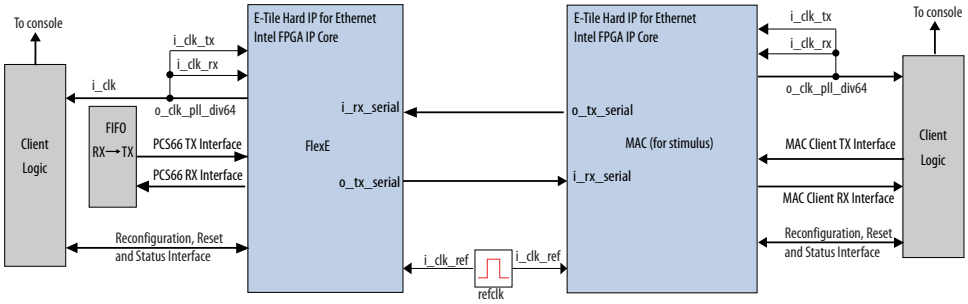
Simulating the E-Tile Hard IP for Ethernet Intel FPGA Design Example Testbench on page 7

3.4. E-Tile Hard IP for Ethernet Intel FPGA 100GE FlexE Simulation Design Example

To generate this design example, choose the following settings in your IP parameter editor:

1. **Single 100GE with optional RSFEC** or **100GE or 1 to 4 channel 10GE/25GE with optional RSFEC and PTP** as the core variant.
2. **100GE Channel** as **Active channel(s) at startup** if you choose **100GE or 1 to 4 channel 10GE/25GE with optional RSFEC and PTP** as the core variant.
3. **100G** as the Ethernet rate.
4. **FlexE** as the Ethernet IP layer.

Figure 13. E-Tile Hard IP for Ethernet Intel FPGA 100GE FlexE Simulation Design Example Block Diagram



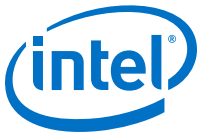
The testbench sends traffic through the IP core, exercising the transmit side and receive side of the IP core.

The successful test run displays output confirming the following behavior:

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 verify that the packets have no errors.
6. Displaying `Testbench complete.`

The following sample output illustrates a successful simulation test run for a 100GE, FlexE only IP core variation.

```
# test_dut: def_100G_o_tx_lanes_stable is 1 at time            345685000
# test_dut: waiting for tx_dll_lock....
# dut: o_tx_lanes_stable is 1 at time            345685000
# dut: waiting for tx_dll_lock....
# dut: TX DLL LOCK is 1 at time            398849563
# dut: waiting for tx_transfer_ready....
# dut: TX transfer ready is 1 at time            399169435
# dut: waiting for rx_transfer_ready....
# dut: RX transfer ready is 1 at time            410719813
# dut: EHIP PLD Ready out is 1 at time            410776000
# dut: EHIP reset out is 0 at time            411040000
# dut: EHIP reset ack is 0 at time            412282101
# dut: EHIP TX reset out is 0 at time            413160000
# dut: EHIP TX reset ack is 0 at time            462643731
# dut: waiting for EHIP Ready....
# dut: EHIP READY is 1 at time            462750387
# dut: EHIP RX reset out is 0 at time            463088000
# dut: waiting for rx reset ack....
# dut: EHIP RX reset ack is 0 at time            463283667
# dut: Waiting for RX Block Lock
# test_dut: TX DLL LOCK is 1 at time            475452243
# test_dut: waiting for tx_transfer_ready....
# test_dut: TX transfer ready is 1 at time            475772115
# test_dut: waiting for rx_transfer_ready....
# test_dut: RX transfer ready is 1 at time            487164223
# test_dut: EHIP PLD Ready out is 1 at time            487224000
# test_dut: EHIP reset out is 0 at time            487488000
# test_dut: EHIP reset ack is 0 at time            488907771
# test_dut: EHIP TX reset out is 0 at time            489784000
# test_dut: EHIP TX reset ack is 0 at time            539116083
# test_dut: waiting for EHIP Ready....
# test_dut: EHIP READY is 1 at time            539169411
# test_dut: EHIP RX reset out is 0 at time            539512000
# test_dut: waiting for rx reset ack....
# test_dut: EHIP RX reset ack is 0 at time            539702691
# test_dut: Waiting for RX Block Lock
# dut: EHIP RX Block Lock  is high at time            542102451
# dut: Waiting for AM lock
# dut: EHIP RX AM Lock  is high at time            543368991
# dut: Waiting for RX alignment
# dut: RX deskew locked
# dut: RX lane aligmnent locked
# dut: ***************************************
# test_dut: EHIP RX Block Lock  is high at time            546535341
# test_dut: Waiting for AM lock
```



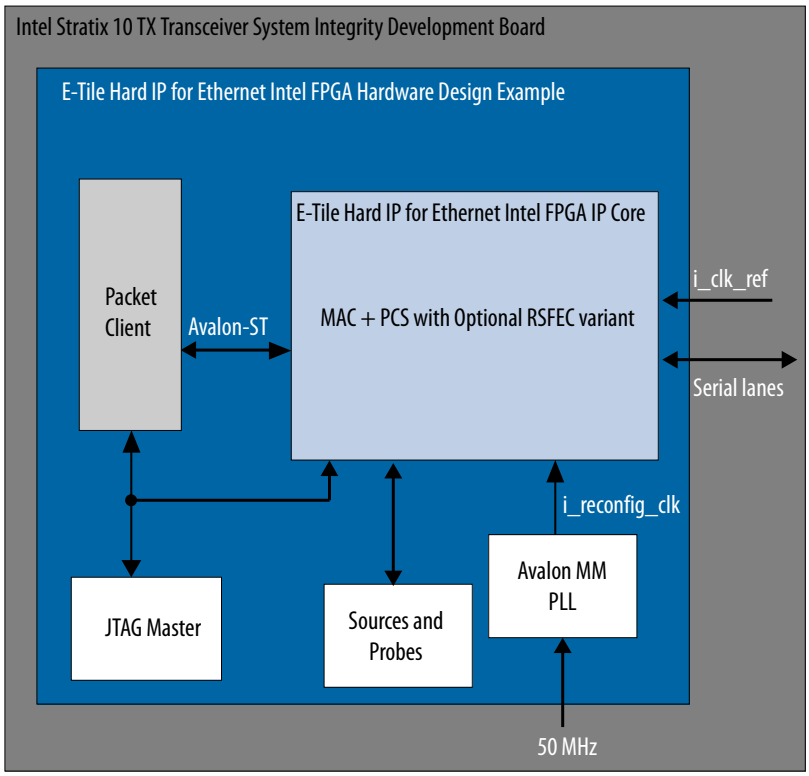
```
# test_dut: EHIP RX AM Lock  is high at time            547801881
# test_dut: Waiting for RX alignment
# test_dut: RX deskew locked
# test_dut: RX lane aligmnent locked
# test_dut: ** Sending Packet           1...
# test_dut: ** Sending Packet           2...
# test_dut: ** Sending Packet           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: ****************************************
```

Related Information

Simulating the E-Tile Hard IP for Ethernet Intel FPGA Design Example Testbench on page 7

3.5. 100GE MAC + PCS with Optional RSFEC Hardware Design Example Components

Figure 14. 100GE MAC + PCS with Optional RSFEC Hardware Design Example High Level Block Diagram



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

- E-Tile Hard IP for Ethernet Intel FPGA IP core.
- Client logic that coordinates the programming of the IP core and packet generation.
- 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. By default, the internal serial loopback is disabled in this design example. Use the `loop_on` command to enable the internal serial loopback. When the internal serial loopback is enabled, the IP core receives the packets and transmit to the packet generator. The client logic reads and print out the MAC statistic registers when the packet transmissions are complete.

Related Information

- Intel Stratix 10 TX Signal Integrity Development Kit Webpage

- Compiling and Configuring the Design Example in Hardware on page 8
- Testing the E-Tile Hard IP for Ethernet Intel FPGA Hardware Design Example on page 9

3.6. 100GE MAC + PCS with Optional RSFEC Design Example Interface Signals

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

Table 6. 100GE MAC + PCS with Optional RSFEC Hardware Design Example Interface Signals

Signal	Direction	Description
`clk50`	Input	Drive at 50 MHz. The intent is to drive this from a 50 Mhz oscillator on the board.
`i_clk_ref`	Input	Drive at 156.25 MHz.
`cpu_resetn`	Input	Resets the IP core. Active low. Drives the global hard reset `csr_reset_n` to the IP core.
`i_rx_serial[3:0]`	Input	Transceiver PHY input serial data.
`o_tx_serial[3:0]`	Output	Transceiver PHY output serial data.
`user_led[3:0]`	Output	Status signals. Currently the design example drives all of these signals to a constant value of 0. The hardware design example connects these bits to drive LEDs on the target board.
`user_io[9:0]`	Output	Status signals. The hardware design example connects these bits to physical pins on the target board for debugging purposes. Individual bits reflect the following signal values and clock behavior: • [0]: `clk_txmac_cnt` • [1]: `clk_rxmac_cnt` • [2]: `clk_rx_rs_cnt` • [3]: `clk_tx_rs_cnt` • [4]: `rx_sop_cnt` • [5]: `rx_eop_cnt` • [6]: `tx_sop_cnt` • [7]: `tx_eop_cnt` • [8]: `prescale` • [9]: `khz_all`

Related Information

E-Tile Hard IP for Ethernet Intel FPGA Interfaces and Signal Descriptions

3.7. 100GE MAC+PCS with Optional RSFEC Design Example Registers

Table 7. E-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
0x002000	Packet monitoring registers

Table 8. Packet Client Registers

You can customize the E-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. • 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.	0x25800040	RW
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.	0x6	RW

continued...

Addr	Name	Bit	Description	HW Reset Value	Access
			• 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.		
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

E-Tile Hard IP for Ethernet Intel FPGA IP core register descriptions

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

Document Version	Intel Quartus Prime Version	Changes
2018.08.10	18.0	Added a note to clarify that the E-Tile Hard IP for Ethernet Intel FPGA IP provides preliminary support for the OTN feature in the following sections: • *Quick Start Guide* • *100GE with Optional RSFEC Design Example* • *100GE OTN Simulation Design Example*
2018.07.19	18.0	Initial release.