E-tile Hard IP Intel® Agilex™ Design Example User Guide

Ethernet, E-tile CPRI PHY and Dynamic Reconfiguration

Updated for Intel® Quartus® Prime Design Suite: **19.4**

Subscribe
Send Feedback

UG-20210 | 2019.12.30
Latest document on the web: PDF | HTML

Contents

1. About E-tile Hard IP Intel® Agilex™ Design Example User Guide

This document consists of the following design examples:

- E-tile Ethernet IP for Intel Agilex FPGA design example
- E-tile CPRI PHY Intel® FPGA IP design example
- E-Tile Dynamic Reconfiguration Design Example

Related Information

- E-tile Ethernet IP for Intel Agilex FPGA Design Example on page 5
- E-tile CPRI PHY Intel FPGA IP Design Example on page 72
- E-Tile Dynamic Reconfiguration Design Example on page 86

2. E-tile Ethernet IP for Intel Agilex™ FPGA Design Example

Related Information

About E-tile Hard IP Intel Agilex Design Example User Guide on page 4

2.1. Quick Start Guide

The E-Tile Ethernet IP for Intel Agilex™ FPGA core for Intel Agilex devices provides a simulation testbench. When you generate the design example, the parameter editor automatically creates the files necessary to simulate, compile, and test the design.

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

Table 1. List of Supported Design Example Variants

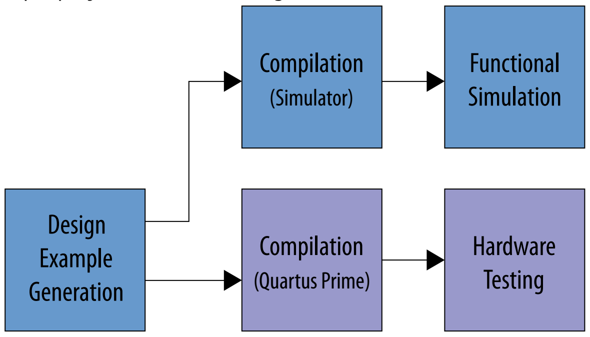
Data Rate	Variant	Simulation	Compilation-Only Project	Hardware Design Example
10GE	Single or multi channels Media Access Controller (MAC) + Physical Coding Sublayer (PCS) with optional 1588 Precision Time Protocol (PTP)	√	√	√
	Single channel PCS	√	√	√
	Single channel Optical Transport Network (OTN)	√	√	X
	Single channel Flexible Ethernet (FlexE)	√	√	X
	Single or multi channels custom PCS	√	√	√
25GE	Single or multi channels MAC + PCS with optional RS-FEC and optional PTP	√	√	√
	Single channel PCS with optional RS-FEC	√	√	√
	Single channel OTN with optional RS-FEC	√	√	X
	Single channel FlexE with optional RS-FEC	√	√	X

continued...

Data Rate	Variant	Simulation	Compilation-Only Project	Hardware Design Example
	Single or multi channels custom PCS with optional RS-FEC	√	√	√
100GE	MAC+ PCS with optional: • (528,514) RS-FEC • PTP	√	√	√
	MAC+PCS with (544, 514) RS-FEC	√	√	√
	PCS with optional (528,514) or (544, 514) RS-FEC	√	√	√
	OTN with optional (528,514) or (544, 514) RS-FEC	√	√	X
	FlexE with optional (528,514) or (544, 514) RS-FEC	√	√	X

Figure 1. Development Steps for the Design Example

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

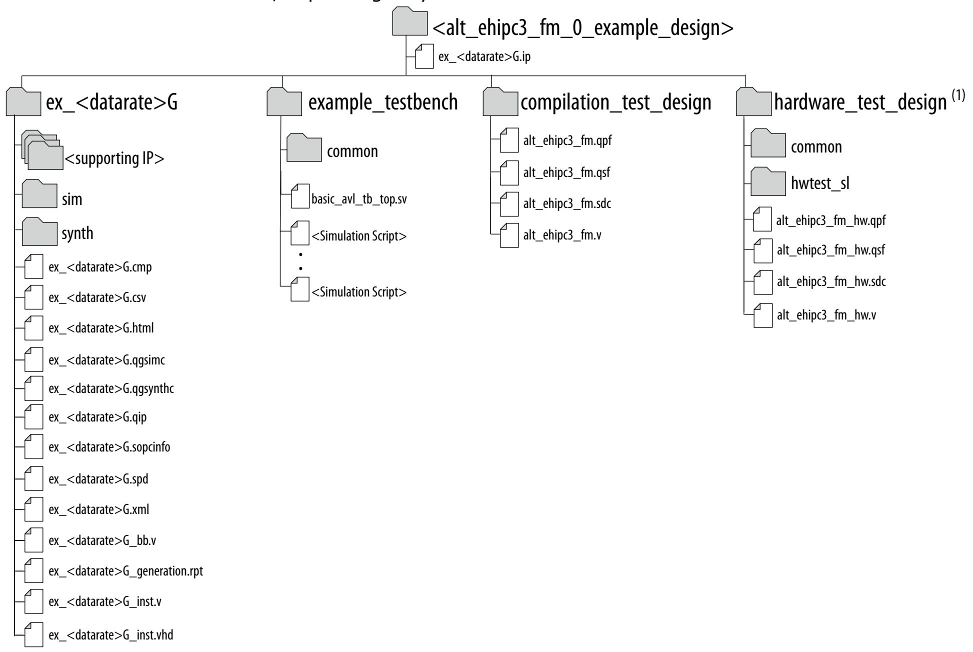


2.1.1. Directory Structure

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

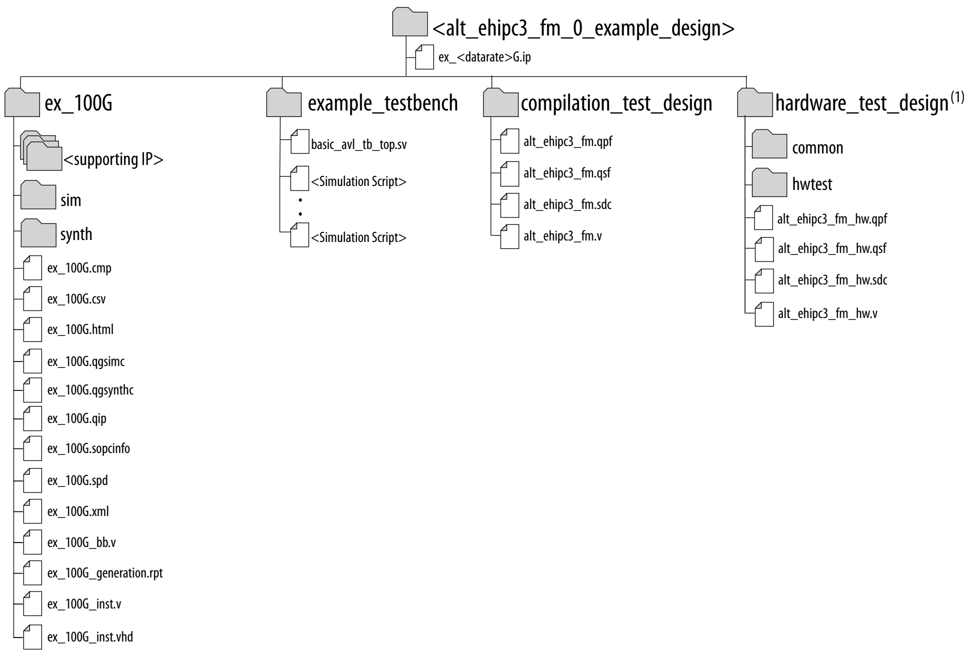
Figure 2. E-tile Ethernet IP for Intel Agilex FPGA 10GE/25GE with Optional RS-FEC and Optional PTP Design Example Directory Structure

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



Note:
1. Intel Quartus® Prime Pro Edition software version 19.1 does not support hardware design examples for the E-Tile Hard IP for Ethernet Intel FPGA IP core for Intel Agilex devices.

Figure 3. E-tile Ethernet IP for Intel Agilex FPGA 100GE with Optional RS-FEC Design Example Directory Structure



Note:
1. Intel Quartus® Prime Pro Edition software version 19.1 does not support hardware design examples for the E-Tile Hard IP for Ethernet Intel FPGA IP core for Intel Agilex devices.

Table 2. E-tile Ethernet IP for Intel Agilex FPGA 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 testbench.
`<design_example_dir>/example_testbench/run_vcsmx.sh`	The Synopsys VCS MX* script (combined Verilog HDL and System Verilog with VHDL) to run the testbench.
`<design_example_dir>/example_testbench/run_ncsim.sh`	The Cadence NCSim* script to run the testbench.
`<design_example_dir>/example_testbench/run_xcelium.sh`	The Xcelium* script to run the testbench.

Table 3. Intel Agilex IP Core Hardware Design Example File Descriptions

File Names	Description
`<design_example_dir>/hardware_test_design/alt_ehipc3_hw.qpf`	Intel Quartus® Prime project file.
`<design_example_dir>/hardware_test_design/alt_ehipc3_hw.qsf`	Intel Quartus Prime project settings file.
`<design_example_dir>/hardware_test_design/alt_ehipc3_hw.sdc`	Synopsys Design Constraints files. You can copy and modify these files for your own Intel Agilex design.
`<design_example_dir>/hardware_test_design/alt_ehipc3_hw.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) `hwtest/main.tcl` (100GE)	Main file for accessing System Console.

2.1.2. Generating the Design

Figure 4. Procedure

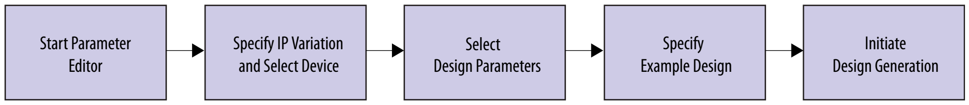
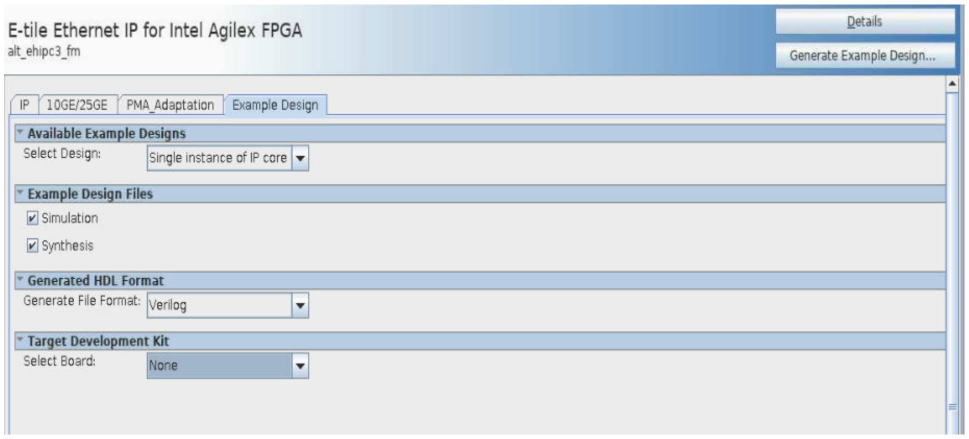


Figure 5. Example Design Tab in the E-tile Ethernet IP for Intel Agilex FPGA Parameter Editor


E-tile Ethernet IP for Intel Agilex FPGA
alt_ehipc3_fm

Available Example Designs
Select Design: Single instance of IP core

Example Design Files
☑ Simulation
☑ Synthesis

Generated HDL Format
Generate File Format: Verilog

Target Development Kit
Select Board: None

Follow these steps to generate the E-tile Ethernet IP for Intel Agilex FPGA testbench:

1. In the IP Catalog, locate and select **E-tile Ethernet IP for Intel Agilex 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. Change PMA adaptation setting. To change the PMA adaptation setting for the optimal performance, go to **PMA Adaptation** tab. This step is optional.
   a. Select a PMA adaptation preset for **PMA adaptation Select** parameter.
   b. Click **PMA Adaptation Preload** to load the initial and continuous adaptation parameters.
   c. Specify the number of PMA configurations to support when multiple PMA configurations are enabled using **Number of PMA configuration** parameter.
   d. Select which PMA configuration to load or store using **Select a PMA configuration to load or store**.
   e. Click **Load adaptation from selected PMA configuration** to load the selected PMA configuration settings.

   For more information about the PMA adaptation parameters, refer to the *E-Tile Transceiver PHY User Guide*.

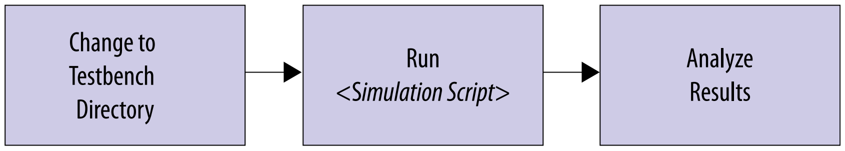
   Note: If you require more information about the PMA adaptation parameters, contact My Intel support.
6. On the **Example Design** tab, under **Example Design Files**, select the **Simulation** option to generate the testbench and the compilation-only project. You must select at least one of the **Simulation** and **Synthesis** options to generate the design example.
7. On the **Example Design** tab, under **Generated HDL Format**, 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.
8. Under **Target Development Kit**, select **None**. The compilation-only design examples target your project device.
9. Click the **Generate Example Design** button. The **Select Example Design Directory** window appears.
10. If you want to modify the design example directory path or name from the defaults displayed (`alt_ehipc3_fm_0_example_design`), browse to the new path and type the new design example directory name (*<design_example_dir>*).

Related Information

E-tile Ethernet IP for Intel Agilex FPGA Core Parameters
Provides more information about customizing your IP core.

2.1.3. Simulating the E-tile Ethernet IP for Intel Agilex FPGA Design Example Testbench

Figure 6. Procedure



Send Feedback

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 4. 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*/VCS MX*	In the command line, type `sh run_vcs.sh` or `sh run_vcsmx.sh` *Note:* `run_vcs.sh` is only available if you select **Verilog** as the **Generated HDL Format**. If you select **VHDL** as the **Generated HDL Format**, you must simulate the testbench with a mixed language simulator using `run_vcsmx.sh`.
Xcelium*	In the command line, type `sh run_xcelium.sh`

2.1.4. 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

2.1.5. 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 `<design_example_dir>/hardware_test_design/output_files` directory. Follow these steps to program the hardware design example on the Intel Agilex device:
   a. Connect Agilex TX Transceiver Signal Integrity Development Kit to the host computer.
   b. Launch the Clock Control application, which is part of the development kit, and set new frequencies for the design example. Below is the frequency setting in the Clock Control application:
      - 10GE/25GE MAC+PCS and 10GE/25GE PCS Only design examples:

        Y1—322.265625 MHz

        U3, OUT3—100 MHz
      - 10GE/25GE Custom PCS design example:

        Y1—X MHz (Set to the frequency set in the Clock Control user interface for PHY_REFCLK)

        U3, OUT3 — 100 MHz
   c. On the **Tools** menu, click **Programmer**.
   d. In the Programmer, click **Hardware Setup**.
   e. Select a programming device.
   f. Select and add the Agilex TX Transceiver Signal Integrity Development Kit to which your Intel Quartus Prime Pro Edition session can connect.
   g. Ensure that **Mode** is set to **JTAG**.
   h. Select the Intel Agilex device and click **Add Device**. The Programmer displays a block diagram of the connections between the devices on your board.
   i. In the row with your `.sof`, check the box for the `.sof`.
   j. Check the box in the **Program/Configure** column.
   k. Click **Start**.

Related Information

- Block-Based Design Flows
- Programming Intel FPGA Devices
- Analyzing and Debugging Designs with System Console

2.1.6. Testing the E-tile Ethernet IP for Intel Agilex FPGA Hardware Design Example

After you compile the E-tile Ethernet IP for Intel Agilex FPGA core design example and configure it on your Intel Agilex device, you can use the System Console to program the IP core and its embedded Native PHY IP core registers.

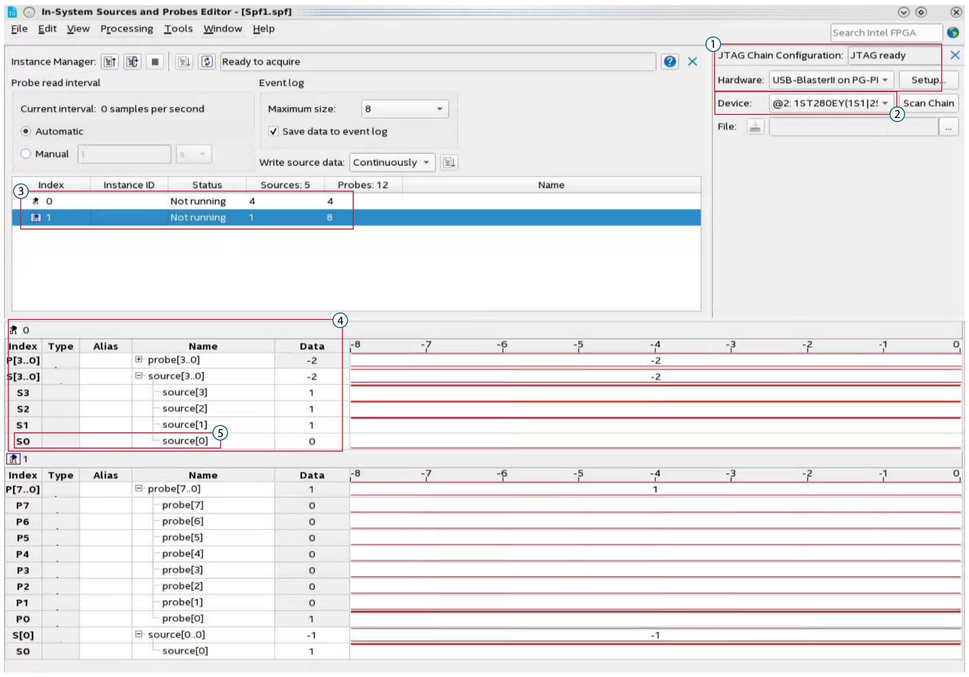
2.1.6.1. 10GE/25GE Design Example

This section applies to 10G/25G Ethernet MAC+PCS with optional RS-FEC and optional PTP, 10G/25G Ethernet PCS only with optional RS-FEC, and 10G/25G Ethernet custom PCS with optional RS-FEC hardware design examples.

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 Agilex device, in the Intel Quartus Prime Pro Edition software, on the **Tools** menu, click **In-System Sources and Probes Editor**.

Figure 7. In-System Sources and Probes Editor


In-System Sources and Probes Editor - [Spf1.spf]
File Edit View Processing Tools Window Help
Search Intel FPGA
Setup...
Scan Chain
-8 -7 -6 -5 -4 -3 -2 -1 0
1
2
3
4
5

Instance Manager: Ready to acquire

Probe read interval

Current interval: 0 samples per second

(●) Automatic

( ) Manual

Event log

Maximum size: 8

[✓] Save data to event log

Write source data: Continuously

JTAG Chain Configuration: JTAG ready

Hardware: USB-BlasterII on PG-PI

Device: @2: 1ST280EY(1S1|2!

File:

Index	Instance ID	Status	Sources: 5	Probes: 12	Name
0		Not running	4	4	
1		Not running	1	8	

0

Index	Type	Alias	Name	Data
P[3..0]			probe[3..0]	-2
S[3..0]			source[3..0]	-2
S3			source[3]	1
S2			source[2]	1
S1			source[1]	1
S0			source[0]	0

1

Index	Type	Alias	Name	Data
P[7..0]			probe[7..0]	1
P7			probe[7]	0
P6			probe[6]	0
P5			probe[5]	0
P4			probe[4]	0
P3			probe[3]	0
P2			probe[2]	0
P1			probe[1]	0
P0			probe[0]	1
S[0]			source[0..0]	-1
S0			source[0]	1

1. JTAG Chain Configuration shows USB-BlasterII connection to the development kit.
2. Shows the device used on the development kit.
3. Shows the number of instances connected to the JTAG chain.
4. Shows the number of probes and sources connected to instance 0.
5. S0 connected to i_reconfig_reset

2. In the **JTAG Chain Configuration** window, select the USB connection that is connected to the development kit.
3. Next, from the **Device** list, select the device with 1ST280EY string in the name. The **Ready to acquire** status appears at the bottom of the **Instance Manager** window if the correct device is selected.

4. A list of instances appears once the connection is acquired. There are four sources under index 0. These sources have the following connections:

Source	Signal
source[3]	`sl_csr_rst_n` (active low)
source[2]	`sl_tx_rst_n` (active low)
source[1]	`sl_rx_rst_n` (active low)
source[0]	`i_reconfig_reset` (active high)

5. Toggle **source[0]** to initiate reset for the transceiver and Ethernet reconfiguration interfaces.
6. Once the reset is initiated, on the **Tools** menu, click **System Debugging Tools ➤ System Console**.
7. In the Tcl Console pane, type `cd hwtest_sl` to change directory to *`<design_example_dir>`*`/hardware_test_design/hwtest_sl`.
8. Type `set <command_setting>` to configure the test according to your design configuration:

Command Setting	Description
`totalChannel`	Set this value according to the value of **Number of Channels of 10GE/25GE** parameter in your design. The default value is 1. Example, in the system console type `set totalChannel 2` to change the number of channels to 2. *Note:* E-tile Ethernet IP for Intel Agilex FPGA does not support multichannel PCS variation.
`jtag_port_id`	Set this value to the JTAG port ID that is connected to the development kit. Example, in the system console type `set jtag_port_id 0` to change the JTAG ID to 0.
`enableILB`	Set this to 1 to enable Internal Serial Loopback. The default value is 1. Example, in the system console, type `set enableILB 0` to disable Internal Serial Loopback.
`enablePTP`	Set this to 1 if PTP is enabled in the design. Otherwise set the value to 0. The default value is 0. Example, in the system console type `set enablePTP 1` to enable PTP.
`speed`	Choose the following option according to the design example variation: • `10G` for 10 Gbps data rate • `25G` for 25 Gbps data rate • `25G_fec` for 25 Gbps data rate with RS-FEC enabled • `pcsonly` for PCS only and custom PCS designs • `pcsonly_fec` for PCS only and custom PCS designs with RS-FEC enabled Example, in the system console type `set speed 25G_fec` to set the data rate to 25G with RS-FEC enabled.

Command Setting	Description
PMAadaptation	Set this to 1 if **Enable adaptation load soft IP** parameter is enabled in your design. Otherwise, set the value to 0. The default value is 0.
PMAConfig	Set the PMA configuration number to enable PMA adaptation. The PMA configuration number set must be one of the PMA configurations defined in your design.

9. Type `source main_script.tcl` to enable the internal loopback and run the test.

Configuring the 10GE/25GE MAC+PCS with optional RS-FEC and optional PTP hardware test in System Console:

```
% set totalChannel 1
1
% set jtag_port_id 0
0
% set enablePTP 0
0
% set speed 25G
25G
% set PMAadaptation 1
1
% set recipe 0
0
% source main_script.tcl
Info: Number of Channels = 1
Info: JTAG Port ID       = 0
Info: PTP Enable         = 0
Info: Speed              = 25G
Info: PMA Adaptation    = 1
Info: PMAConfig Number   = 0
```

Set the speed to `pcsonly` to configure 10GE/25GE PCS only with optional RS-FEC hardware test. Set the speed to `pcsonly_fec` to configure 10G/25G custom PCS with optional RS-FEC hardware test.

Related Information

Intel Quartus Prime Pro Edition User Guide: Debug Tools - In-System Sources and Probes

2.1.6.2. 100GE MAC+PCS with Optional (528,514) RS-FEC or (544,514) RS-FEC and Adaptation Flow Hardware Design Example

This hardware design example enables internal serial loopback mode by default. To run the hardware design with external loopback mode, select **Enable adaptation load soft IP** in the parameter editor before generating the 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 Agilex 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 use the following design example commands to configure the 100GE hardware design example test with internal serial loopback mode. For example, in the system console, type `run_test` and press `Enter`.

- `run_test`(1)/`run_test_pam4`(2): To run hardware design example tests.
- `start_pma_init_adaptation`(1)/`start_pma_02_init_adaptation`(2): To perform PMA adaptation.
- `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`(1)/`loop_on_pam4`(2): Turns on internal serial loopback.
- `loop_off`: Turns off internal serial loopback.
- `reg_read` *<addr>*: Returns the IP core register value at *<addr>*. Example, to read TX datapath PCS ready register, type `reg_read 0x322`.
- `reg_write` *<addr> <data>*: Writes *<data>* to the IP core register at address *<addr>*. Example, to initiate soft reset on RX PCS, type `reg_write 0x310 0x0004>`
- `chk_init_adaptation_status`(1)/`chk_init_adaptation_status02`(2): Check for PAM4 PMA adaptation status.

4. Optional step: To run the MAC+PCS with (528,514) RS-FEC or (544, 514) RS-FEC and PMA adaptation design example in external loopback mode, open `hardware_test_design/hwtest/main.tcl` file and uncomment `start_pma_init_adaptation`(1)/`start_pma_02_init_adaptation`(2) command.

   Make sure the **Enable adaptation load soft IP** is selected and the **PMA adaptation Select** is set to:

   - **NRZ_28Gbps_LR**, **NRZ_28Gbps_VSR**, or **NRZ_10Gbps** before generating the design example(1)
   - **PAM4_56Gbps_LR** or **PAM4_56Gbps_VSR** before generating the design example(2)

5. Disable the internal serial loopback mode by using `loop_off` command.

(1) Applicable for 100GE MAC+PCS with optional (528,514) RS-FEC and PMA adaptation hardware design example.

(2) Applicable for 100GE MAC+PCS with optional (544,514) RS-FEC and PMA adaptation hardware design example.

You can use the following design example commands to configure the 100GE hardware design example test with external loopback mode.

- `start_pma_init_adaptation`(1)/ `start_pma_02_init_adaptation_ex`(2): Performs PMA adaptation on external loopback or external devices connection tests.
- `start_pma_anlg_rst03`(1)/`start_pma_anlg_02`(2): Performs NRZ transceiver PMA reset.
- `init_adaptation_16_NoPrbsNoLdEL03`(1)/ `init_adaptation_16_NoPrbsNoLdELCntPC02`(2): Performs NRZ PMA adaptation.

  Important: All the values set in this design example are tested with Agilex TX Transceiver Signal Integrity Development Kit. You may need to customize the PMA adaptation configuration values if you are running this design example on boards other than the Agilex TX Transceiver Signal Integrity Development Kit.
- `chk_init_adaptation_status`(1)/ `chk_init_adaptation_status_02`(2): Checks for PAM4 PMA adaptation status.
- `ld_rcp`: Loads PMA configuration settings based on the selection set in the **Select a PMA configuration to load or store** in the parameter editor.

  Important: All the values set in this design example are tested with Agilex TX Transceiver Signal Integrity Development Kit. You may need to customize the PMA adaptation configuration values if you are running this design example on boards other than the Agilex TX Transceiver Signal Integrity Development Kit.
- `chk_rcp_status`(1): Checks PMA configuration settings load status and retry if necessary.

Related Information

- Intel Quartus Prime Pro Edition User Guide: Debug Tools - In-System Sources and Probes
- E-Tile Transceiver PHY User Guide
  More information on parameters in **PMA Adaptation** tab.

2.1.6.3. 100GE PCS Only with Optional (528,514) RS-FEC or (544,514) RS-FEC, and Optional PTP 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 Agilex 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.
4. Type `pcs_only_traffic_test <number of iteration>` to run the specified iteration of PCS only with (528,514) RS-FEC hardware design example test. If no value is specified, the test runs only 1 iteration. Each packet generated for every iterations are in random number of frames, size, and types.
5. Type `pcs_only_traffic_test_pam4 <number of interation>` to run the specified iteration of PCS only with (544,514) RS-FEC hardware design example test. If no value is specified, the test runs only 1 iteration. Each packet generated for every iterations are in random number of frames, size, and types.

Related Information

Intel Quartus Prime Pro Edition User Guide: Debug Tools - In-System Sources and Probes

2.2. 10GE/25GE with Optional RS-FEC Design Examples

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

Table 5. Supported Design Example Variants for 10GE/25GE

All variant supports up to 4 channels.

Variant	Intel Agilex Design Example Support
MAC+PCS with Optional RS-FEC[3]	Simulation and compilation-only project
MAC+PCS with Optional RS-FEC and PTP[3]	Simulation and compilation-only project
PCS Only with Optional RS-FEC[3]	Simulation and compilation-only project
OTN with Optional RS-FEC[3]	Simulation and compilation-only project
FlexE with Optional RS-FEC[3]	Simulation and compilation-only project
Custom PCS with Optional RS-FEC[3]	Simulation and compilation-only project

(3) RS-FEC is not supported in 10GE variant.

2.2.1. Simulation Design Examples

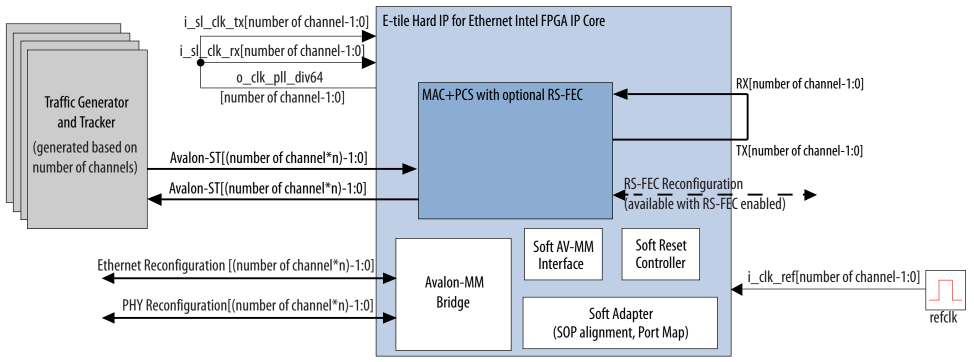
2.2.1.1. Non-PTP 10GE/25GE MAC+PCS with Optional RS-FEC Simulation Design Example

The simulation block diagram below is generated using the following settings in the IP parameter editor:

1. Under the **IP** tab:
   a. **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.
   b. **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.
   c. **Enable RSFEC** to use the RS-FEC feature.
2. Under the **10GE/25GE** tab:
   a. **10G** or **25G** as the Ethernet rate.
3. **Enable asynchronous adapter clocks** to use the asynchronous adapter feature.

Note: RS-FEC is not supported in 10GE variant.

Figure 8. Simulation Block Diagram for Non-PTP E-tile Ethernet IP for Intel Agilex FPGA 10GE/25GE MAC+PCS with Optional RS-FEC Design Example


Traffic Generator and Tracker
(generated based on number of channels)
i_sl_clk_tx[number of channel-1:0]
i_sl_clk_rx[number of channel-1:0]
o_clk_pll_div64 [number of channel-1:0]
E-tile Hard IP for Ethernet Intel FPGA IP Core
MAC+PCS with optional RS-FEC
RX[number of channel-1:0]
TX[number of channel-1:0]
Avalon-ST[(number of channel*n)-1:0]
Avalon-ST[(number of channel*n)-1:0]
RS-FEC Reconfiguration (available with RS-FEC enabled)
Soft AV-MM Interface
Soft Reset Controller
Ethernet Reconfiguration [(number of channel*n)-1:0]
PHY Reconfiguration[(number of channel*n)-1:0]
Avalon-MM Bridge
Soft Adapter (SOP alignment, Port Map)
i_clk_ref[number of channel-1:0]
refclk

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 16,384 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 25GE, MAC+PCS with RS-FEC, non-PTP IP core variation.

```
# Ref clock is 156.25 MHz
# Channel 0 - waiting for EHIP Ready....
# Channel 0 - EHIP READY is 1 at time           2472365000
# Channel 0 - Waiting for RX Block Lock
# Channel 0 - EHIP RX Block Lock  is high at time           2507639043
# Channel 0 - Waiting for RX alignment
# Channel 0 - RX deskew locked
# Channel 0 - RX lane alignmnent locked
# Channel 0 - 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...
# ** Sending Packet           10...
# Channel 0 - Received Packet          1...
# Channel 0 - Received Packet          2...
# Channel 0 - Received Packet          3...
# Channel 0 - Received Packet          4...
# Channel 0 - Received Packet          5...
# Channel 0 - Received Packet          6...
# Channel 0 - Received Packet          7...
# Channel 0 - Received Packet          8...
# Channel 0 - Received Packet          9...
# Channel 0 - Received Packet         10...
# **
# ** Reading KR CSR -C0
# ** Address offset = 000c0, ReadData  = 737d0381
# ** AVMM access CSR registers read/write check for ETH amd XCVR CH0
# ** Address offset = 00301, ReadData  = 00000000
# ** 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, WriteData = ffffffff
# ** Address offset = 00084, ReadData  = 000000ff
# ** Address offset = 00084, WriteData = 00000000
# ** Address offset = 00230, WriteData = ffffffff
# ** Address offset = 00230, ReadData  = 000000ff
# ** Address offset = 00230, WriteData = 0000007b
# **
# ** AVMM access CSR registers read/write check for ETH RSFEC
# ** Address offset = 10000, ReadData  = 00000001
# ** Address offset = 10000, WriteData = ffffffff
# ** Address offset = 10000, ReadData  = 000000fd
# ** Address offset = 10004, ReadData  = 00000004
# ** Address offset = 10010, ReadData  = 00000061
# ** Address offset = 10011, ReadData  = 00000066
# ** Address offset = 10000, WriteData = 00000001
# ** Check KR CSR Status - C0
# ** Address offset = 000b1, ReadData  = 00040801
# ** Address offset = 000d2, ReadData  = 00000001
# **
# ** Testbench complete.
# **
# *****************************************
# ** Note: $finish    : ./basic_avl_tb_top.sv(415)
#    Time: 2628595 ns  Iteration: 0  Instance: /basic_avl_tb_top
```

Related Information

Simulating the E-tile Ethernet IP for Intel Agilex FPGA Design Example Testbench on page 10

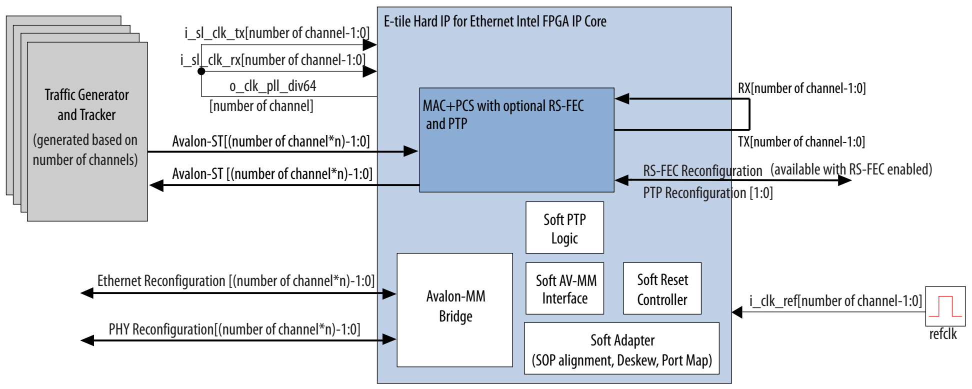
2.2.1.2. PTP 10GE/25GE MAC+PCS with Optional RS-FEC Simulation Design Example

The simulation block diagram below is generated using the following settings:

1. Under the **IP** tab:
    a. **100GE or 1 to 4 channel 10GE/25GE with optional RSFEC and PTP** as the core variant.
    b. **10G/25GE channels** as **Active channel(s) at startup**.
    c. **Enable IEEE 1588 PTP**.
    d. **Enable RSFEC** to use the RS-FEC feature.
2. Under the **10GE/25GE** tab:
    a. **10G** or **25G** as the Ethernet rate.

Note: RS-FEC is not supported in 10GE variant.

Figure 9. Simulation Block Diagram for E-tile Ethernet IP for Intel Agilex FPGA 10GE/25GE with Optional RS-FEC and PTP Design Example


Traffic Generator and Tracker
(generated based on number of channels)
i_sl_clk_tx[number of channel-1:0]
i_sl_clk_rx[number of channel-1:0]
o_clk_pll_div64
[number of channel]
Avalon-ST[(number of channel*n)-1:0]
Avalon-ST [(number of channel*n)-1:0]
E-tile Hard IP for Ethernet Intel FPGA IP Core
MAC+PCS with optional RS-FEC and PTP
RX[number of channel-1:0]
TX[number of channel-1:0]
RS-FEC Reconfiguration (available with RS-FEC enabled)
PTP Reconfiguration [1:0]
Soft PTP Logic
Ethernet Reconfiguration [(number of channel*n)-1:0]
PHY Reconfiguration[(number of channel*n)-1:0]
Avalon-MM Bridge
Soft AV-MM Interface
Soft Reset Controller
Soft Adapter
(SOP alignment, Deskew, Port Map)
i_clk_ref[number of channel-1:0]
refclk

In this design example, 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 16,384 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 25GE, MAC+PCS, RS-FEC, PTP IP core variation.

```
# Channel 0 - EHIP Ready is high
# Channel 0 - Waiting for RX Block Lock
# Channel 0 - RX Block Lock is high
# Channel 0 - Waiting for RX alignment
# Channel 0 - RX lane aligmnent locked
# Channel 0 - Waiting for TX PTP Ready
# Channel 0 - TX PTP ready
# Channel 0 - Training RX PTP AIB deskew and waiting for RX PTP ready
# Channel 0 - Sending  Packet          1
# Channel 0 - Received Packet          1
# Channel 0 - Sending  Packet          2
# Channel 0 - Received Packet          2
# Channel 0 - Sending  Packet          3
# Channel 0 - Received Packet          3
# Channel 0 - Sending  Packet          4
# Channel 0 - Received Packet          4
# Channel 0 - RX PTP ready
.
.
(Repeat tests for Channel 1, Channel 2, and Channel 3)
.
.
# ====> writedata = 00000000
#
# Channel 0 - Configure TX extra latency
# ====> writedata = 0004267a
#
# Channel 0 - Configure RX extra latency
# ====> writedata = 8002d4de
#
# Channel 0 - TX enabled
# Channel 0 - Sending  Packet          1
# Channel 0 - Sending  Packet          2
# Channel 0 - Sending  Packet          3
# Channel 0 - Sending  Packet          4
# Channel 0 - Sending  Packet          5
# Channel 0 - Sending  Packet          6
# Channel 0 - Sending  Packet          7
# Channel 0 - Sending  Packet          8
# Channel 0 - Sending  Packet          9
# Channel 0 - Sending  Packet         10
# Channel 0 - Received Packet          1
# Channel 0 - Received Packet          2
# Channel 0 - Received Packet          3
# Channel 0 - Received Packet          4
# Channel 0 - Received Packet          5
# Channel 0 - Received Packet          6
# Channel 0 - Received Packet          7
# Channel 0 - Received Packet          8
# Channel 0 - Received Packet          9
# Channel 0 - Received Packet         10
# ====> writedata = 00000000
.
.
```

```
(Send and receive packets for Channel 1 and Channel 2)
.
.
# ====> writedata = 00000000
#
# Channel 3 - Configure TX extra latency
# ====> writedata = 0004267a
#
# Channel 3 - Configure RX extra latency
# ====> writedata = 800369d0
#
# Channel 3 - TX enabled
# Channel 3 - Sending  Packet          1
# Channel 3 - Sending  Packet          2
# Channel 3 - Sending  Packet          3
# Channel 3 - Sending  Packet          4
# Channel 3 - Sending  Packet          5
# Channel 3 - Sending  Packet          6
# Channel 3 - Sending  Packet          7
# Channel 3 - Sending  Packet          8
# Channel 3 - Sending  Packet          9
# Channel 3 - Sending  Packet         10
# Channel 3 - Received Packet          1
# Channel 3 - Received Packet          2
# Channel 3 - Received Packet          3
# Channel 3 - Received Packet          4
# Channel 3 - Received Packet          5
# Channel 3 - Received Packet          6
# Channel 3 - Received Packet          7
# Channel 3 - Received Packet          8
# Channel 3 - Received Packet          9
# Channel 3 - Received Packet         10
# *****************************************
# ** Testbench complete.
# *****************************************
# ** Note: $finish    : ./basic_avl_tb_top.sv(484)
#    Time: 473545955 ps  Iteration: 0  Instance: /basic_avl_tb_top
```

Related Information

Simulating the E-tile Ethernet IP for Intel Agilex FPGA Design Example Testbench on page 10

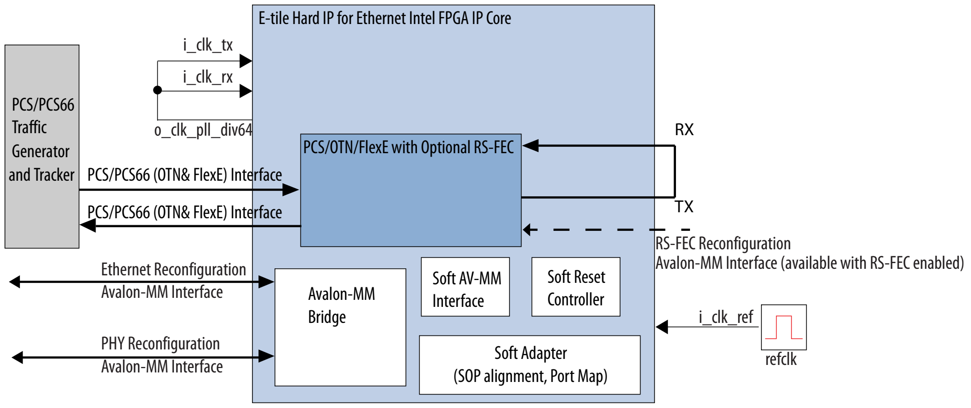
2.2.1.3. 10GE/25GE PCS Only, OTN, or FlexE with Optional RS-FEC Simulation Design Example

The simulation block diagram below is generated using the following settings in the IP parameter editor:

1. Under the **IP** tab:
    a. **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.
    b. **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.
    c. **Enable RSFEC** to use the RS-FEC feature.
2. Under the **10GE/25GE** tab:
    a. **10G** or **25G** as the Ethernet rate.
    b. Select **PCS Only**, **OTN**, or **FlexE** as Ethernet IP layers.

Note: RS-FEC is not supported in 10GE variant.

Figure 10. Simulation Block Diagram for E-tile Ethernet IP for Intel Agilex FPGA 10GE/25GE PCS Only, OTN, or FlexE with Optional RS-FEC Design Examples


E-tile Hard IP for Ethernet Intel FPGA IP Core
PCS/PCS66 Traffic Generator and Tracker
i_clk_tx
i_clk_rx
o_clk_pll_div64
PCS/OTN/FlexE with Optional RS-FEC
RX
TX
PCS/PCS66 (OTN& FlexE) Interface
PCS/PCS66 (OTN& FlexE) Interface
RS-FEC Reconfiguration Avalon-MM Interface (available with RS-FEC enabled)
Ethernet Reconfiguration Avalon-MM Interface
Avalon-MM Bridge
Soft AV-MM Interface
Soft Reset Controller
i_clk_ref
refclk
PHY Reconfiguration Avalon-MM Interface
Soft Adapter (SOP alignment, Port Map)

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. Wait for PLL to lock.
2. Wait for RX transceiver reset to complete.
3. Wait for RX alignment.
4. Send three sets of packet.
5. Receive and verify the packets.
6. Displaying `Testbench complete.`

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

```
# Ref clock is 322.265625 MHz
# waiting for EHIP Ready....
# EHIP READY is 1 at time             425955000
# Waiting for RX Block Lock
# EHIP RX Block Lock  is high at time             429395673
# Waiting for RX alignment
# RX deskew locked
# RX lane alignment locked
# TX enabled
# *** Sending packets ***
# Start frame detected, byteslip 0, time 431948219
# ** RX checker has received packets correctly!
# ** RX checker is reset.
# *** Second attempt of sending packets ***
# Start frame detected, byteslip 0, time 437204752
# ** RX checker has received packets correctly!
# ** RX checker is reset.
# *** Third attempt of sending packets ***
# Start frame detected, byteslip 0, time 442467492
# ** RX checker has received packets correctly!
# ** PASSED
# **
# *****************************************
# ** Note: $finish    : ./basic_avl_tb_top.sv(246)
#    Time: 445329189 ps  Iteration: 0  Instance: /basic_avl_tb_top
# 1
# Break in Module basic_avl_tb_top at ./basic_avl_tb_top.sv line 246
```

Related Information

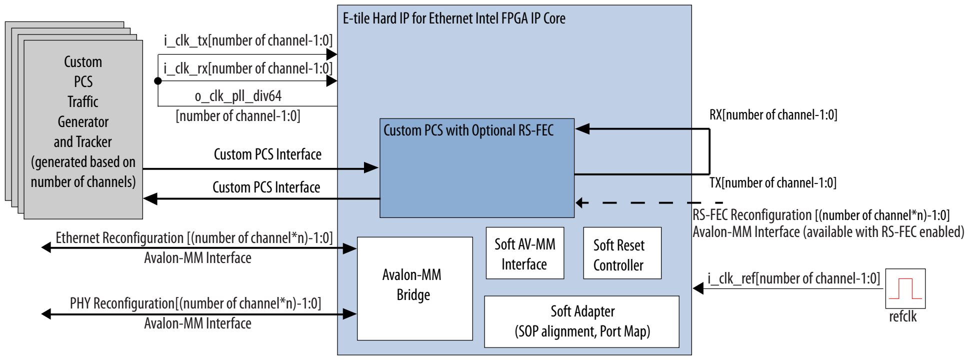
Simulating the E-tile Ethernet IP for Intel Agilex FPGA Design Example Testbench on page 10

2.2.1.4. 10GE/25GE Custom PCS with Optional RS-FEC Simulation Design Example

The simulation block diagram below is generated using the following settings in the IP parameter editor:

1. Under the **IP** tab:
   a. **Custom PCS with optional RSFEC** as the core variant.
   b. **Enable RSFEC** to use the RS-FEC feature.
2. Under the **Custom PCS Channel(s)** tab:
   a. **PCS+RSFEC** as the custom PCS mode.

Figure 11. Simulation Block Diagram for E-tile Ethernet IP for Intel Agilex FPGA 10GE/25GE Custom PCS with Optional RS-FEC Design Example


Custom PCS Traffic Generator and Tracker (generated based on number of channels)
i_clk_tx[number of channel-1:0]
i_clk_rx[number of channel-1:0]
o_clk_pll_div64 [number of channel-1:0]
Custom PCS Interface
Custom PCS Interface
Ethernet Reconfiguration [(number of channel*n)-1:0] Avalon-MM Interface
PHY Reconfiguration[(number of channel*n)-1:0] Avalon-MM Interface
E-tile Hard IP for Ethernet Intel FPGA IP Core
Custom PCS with Optional RS-FEC
Avalon-MM Bridge
Soft AV-MM Interface
Soft Reset Controller
Soft Adapter (SOP alignment, Port Map)
RX[number of channel-1:0]
TX[number of channel-1:0]
RS-FEC Reconfiguration [(number of channel*n)-1:0] Avalon-MM Interface (available with RS-FEC enabled)
i_clk_ref[number of channel-1:0]
refclk

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. Wait for PLL to lock.
2. Wait for RX transceiver reset to complete.
3. Wait for RX alignment.
4. Send three sets of packet.
5. Receive and verify the packets.
6. Displaying `Testbench complete.`

The following sample output illustrates a successful simulation test run for a 10GE, custom PCS, RS-FEC IP core variation.

```
Ref clock is 184.320000 MHz
Channel 0 - waiting for EHIP Ready....
Channel 0 - EHIP READY is 1 at time            382745000
Channel 0 - Waiting for RX Block Lock
Channel 0 - EHIP RX Block Lock  is high at time            387137583
Channel 0 - Waiting for RX alignment
Channel 0 - RX deskew locked
Channel 0 - RX lane aligmnent locked
Channel 0 - TX enabled
*** Channel 0 - Sending packets ***
Start frame detected, byteslip 0, time 389768227
** Channel 0 - RX checker has received packets correctly!
** Channel 0 - RX checker is reset.
*** Channel 0 - Second attempt of sending packets ***
Start frame detected, byteslip 0, time 395241712
** Channel 0 - RX checker has received packets correctly!
** Channel 0 - RX checker is reset.
*** Channel 0 - Third attempt of sending packets ***
Start frame detected, byteslip 0, time 400721512
** Channel 0 - RX checker has received packets correctly!
Channel 1 - waiting for EHIP Ready....
Channel 1 - EHIP READY is 1 at time            403524543
Channel 1 - Waiting for RX Block Lock
Channel 1 - EHIP RX Block Lock  is high at time            403524543
Channel 1 - Waiting for RX alignment
Channel 1 - RX deskew locked
Channel 1 - RX lane aligmnent locked
Channel 1 - TX enabled
*** Channel 1 - Sending packets ***
Start frame detected, byteslip 0, time 406113519
** Channel 1 - RX checker has received packets correctly!
** Channel 1 - RX checker is reset.
*** Channel 1 - Second attempt of sending packets ***
Start frame detected, byteslip 0, time 411605943
** Channel 1 - RX checker has received packets correctly!
** Channel 1 - RX checker is reset.
*** Channel 1 - Third attempt of sending packets ***
Start frame detected, byteslip 0, time 417092055
** Channel 1 - RX checker has received packets correctly!
Channel 2 - waiting for EHIP Ready....
Channel 2 - EHIP READY is 1 at time            419907712
Channel 2 - Waiting for RX Block Lock
Channel 2 - EHIP RX Block Lock  is high at time            419907712
Channel 2 - Waiting for RX alignment
Channel 2 - RX deskew locked
Channel 2 - RX lane aligmnent locked
Channel 2 - TX enabled
*** Channel 2 - Sending packets ***
Start frame detected, byteslip 0, time 422502903
** Channel 2 - RX checker has received packets correctly!
** Channel 2 - RX checker is reset.
*** Channel 2 - Second attempt of sending packets ***
Start frame detected, byteslip 0, time 428007954
** Channel 2 - RX checker has received packets correctly!
** Channel 2 - RX checker is reset.
*** Channel 2 - Third attempt of sending packets ***
Start frame detected, byteslip 0, time 433494066
** Channel 2 - RX checker has received packets correctly!
Channel 3 - waiting for EHIP Ready....
Channel 3 - EHIP READY is 1 at time            436322349
Channel 3 - Waiting for RX Block Lock
Channel 3 - EHIP RX Block Lock  is high at time            436322349
Channel 3 - Waiting for RX alignment
Channel 3 - RX deskew locked
Channel 3 - RX lane aligmnent locked
Channel 3 - TX enabled
*** Channel 3 - Sending packets ***
Start frame detected, byteslip 0, time 438905013
** Channel 3 - RX checker has received packets correctly!
** Channel 3 - RX checker is reset.
*** Channel 3 - Second attempt of sending packets ***
Start frame detected, byteslip 0, time 444384812
** Channel 3 - RX checker has received packets correctly!
** Channel 3 - RX checker is reset.
*** Channel 3 - Third attempt of sending packets ***
Start frame detected, byteslip 0, time 449864611
** Channel 3 - RX checker has received packets correctly!
** PASSED
```

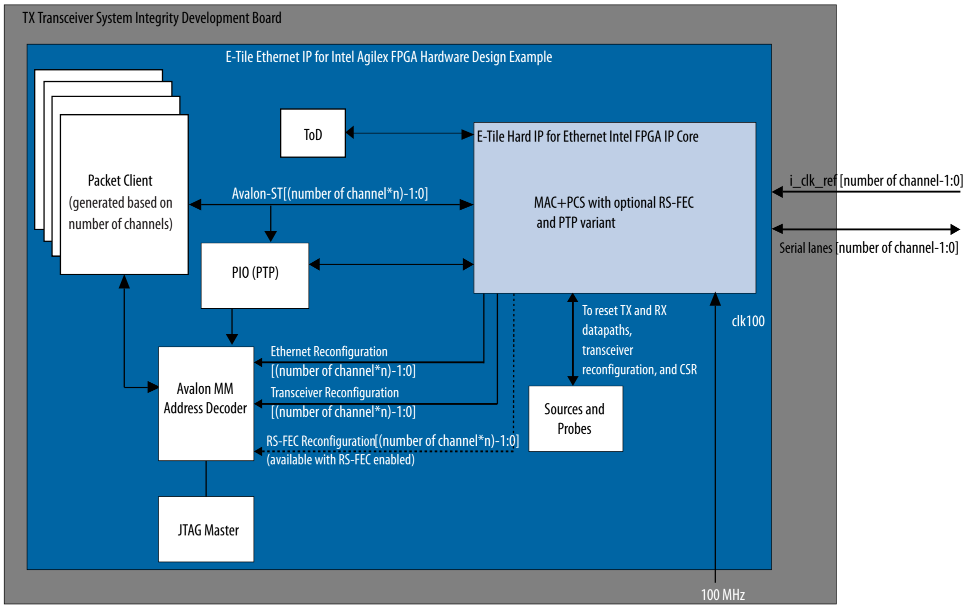
```
**
*****************************************
$finish called from file "basic_avl_tb_top.sv", line 285.
$finish at simulation time        452773953718
```

2.2.2. Hardware Design Examples

Hardware Design examples are supported for Intel Agilex devices.

2.2.2.1. 10GE/25GE MAC+PCS with Optional RS-FEC and PTP Hardware Design Example Components

Figure 12. 10GE/25GE MAC+PCS with Optional RS-FEC and PTP Hardware Design Example High Level Block Diagram


TX Transceiver System Integrity Development Board
E-Tile Ethernet IP for Intel Agilex FPGA Hardware Design Example
Packet Client (generated based on number of channels)
ToD
E-Tile Hard IP for Ethernet Intel FPGA IP Core
MAC+PCS with optional RS-FEC and PTP variant
Avalon-ST[(number of channel*n)-1:0]
PIO (PTP)
i_clk_ref [number of channel-1:0]
Serial lanes [number of channel-1:0]
To reset TX and RX datapaths, transceiver reconfiguration, and CSR
clk100
Avalon MM Address Decoder
Ethernet Reconfiguration [(number of channel*n)-1:0]
Transceiver Reconfiguration [(number of channel*n)-1:0]
RS-FEC Reconfiguration[(number of channel*n)-1:0] (available with RS-FEC enabled)
Sources and Probes
JTAG Master
100 MHz

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

- E-tile Ethernet IP for Intel Agilex FPGA core.
- Client logic that coordinates the programming of the IP core and packet generation.
- Time-of-day (ToD) module to provide a continuous flow of current time-of-day information to the IP core.
- PIO block to store RX and TX PTP timestamp for accuracy calculation and to send PTP 2-step timestamp request.
- Avalon® memory-mapped interface address decoder to decode reconfiguration address space for MAC, transceiver, and RS-FEC modules during reconfiguration accesses.
- JTAG controller that communicates with the System Console. You communicate with the client logic through the System Console.

The following sample output illustrates a successful hardware test run for a 25GE, 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: 13:08:58
    Test Start date is: 03/12/2019

    Successfully Write XCVR Channel 0, CSR Register offset = 0x84, data = 0x0
    Successfully Write XCVR Channel 0, CSR Register offset = 0x85, data = 0x0
.
.
.
    Successfully Read  XCVR  Channel 0, CSR Register offset = 0x89, data = 0x0
Info: ELANE Channel 0 Internal Loopback initialAdaptation Status
    Successfully Write XCVR Channel 0, CSR Register offset = 0x84, data = 0x0
    Successfully Write XCVR Channel 0, CSR Register offset = 0x85, data = 0xb
.
.
.
    Successfully Read  XCVR  Channel 0, CSR Register offset = 0x89, data = 0x0
Info: initialAdaptation is done successfully on channel 0
    Successfully Write XCVR Channel 0, CSR Register offset = 0x84, data = 0x0
    Successfully Write XCVR Channel 0, CSR Register offset = 0x85, data = 0x8f
.
.
.
    Successfully Read  XCVR  Channel 0, CSR Register offset = 0x89, data = 0x0
    Successfully Write EHIPLANE Channel 0, User Register
phy_ehip_csr_soft_reset               , offset = 0x310, data = 0x0
    Successfully Write EHIPLANE Channel 0, 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

    Test End time is: 13:09:02
    Test End date is: 03/12/2019

Info: Closed JTAG Master Service



Info: Test <c3_elane_xcvr_loopback_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: 13:09:02
    Test Start date is: 03/12/2019
```

```
Info: Read all ELANE CSR registers

    Successfully Read  EHIPLANE Channel 0, User Register
phy_revid                              , offset = 0x300, data = 0x11112015
    Successfully Read  EHIPLANE Channel 0, User Register
phy_scratch                           , offset = 0x301, data = 0x0
.
.
.
    Successfully Read  EHIPLANE Channel 0, User Register
phy_ehip_csr_soft_reset               , offset = 0x310, data = 0x0

    C3 ELANE Channel 0 System Reset is successfully

    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: Stopping the traffic generator

    Successfully Write EHIPLANE Channel 0, Traffic GEN/CHK Register
pkt_tx_ctrl                , offset = 0x10, data = 0x87

Info: clearing the traffic generator statistics

    Successfully Write EHIPLANE Channel 0, Traffic GEN/CHK Register
pkt_clear_dropped_counter  , offset = 0x7, data = 0x3
    Successfully Write EHIPLANE Channel 0, Traffic GEN/CHK Register
pkt_clear_dropped_counter  , offset = 0x7, data = 0x0

Info: clearing the statistics

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

Info: Enabling the statistics

    Successfully Write EHIPLANE Channel 0, User Register
cntr_tx_config                        , offset = 0x845, data = 0x0
    Successfully Write EHIPLANE Channel 0, User Register
cntr_rx_config                        , offset = 0x945, data = 0x0

Info: Starting the traffic generator

    Successfully Write EHIPLANE Channel 0, Traffic GEN/CHK Register
pkt_tx_ctrl                , offset = 0x10, data = 0x85
    Successfully Read  EHIPLANE Channel 0, User Register
cntr_tx_fragments_lo                  , offset = 0x800, data = 0x0

Info: Stopping the traffic generator

    Successfully Write EHIPLANE Channel 0, Traffic GEN/CHK Register
pkt_tx_ctrl                , offset = 0x10, data = 0x87
    Successfully Read  EHIPLANE Channel 0, Traffic GEN/CHK Register
pkt_tx_ctrl                , offset = 0x10, data = 0x87
.
.
.
    Successfully Read  EHIPLANE Channel 0, Traffic GEN/CHK Register
pkt_rx_pkt_cnt             , offset = 0x5, data = 0x463f3f

Info: Channel 0 test is completed

    Successfully Read  RSFEC Register rsfec_top_rx_cfg                     ,
offset = 0x14, data = 0x1
    Successfully Read  RSFEC Register arbiter_base_cfg                      ,
offset = 0x0, data = 0x1
.
```

```
.
.
    Successfully Read  RSFEC Register rsfec_top_tx_cfg                     ,
offset = 0x10, data = 0x6661

    Test End time is: 13:09:13
    Test End date is: 03/12/2019

Info: Closed JTAG Master Service


Info: Test <c3_elane_traffic_basic_test> Passed
```

The following sample output illustrate a successful hardware test run for a 25GE, MAC +PCS, with PTP IP core variation. The test result is located at `<design_example_dir>/hardware_test_design/hwtest_sl/c3_elane_ptp_traffic_basic_test.log`.

```

Info: Set JTAG Master Service Path


Info: Opened JTAG Master Service

    Test Start time is: 17:50:05
    Test Start date is: 03/12/2019

    Successfully Write EHIPLANE Channel 0, User Register
phy_ehip_csr_soft_reset               , offset = 0x310, data = 0x0
    Successfully Write EHIPLANE Channel 0, 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: Stopping the traffic generator

    Successfully Write EHIPLANE Channel 0, Traffic GEN/CHK Register
pkt_tx_ctrl                , offset = 0x10, data = 0x57

Info: clearing the traffic generator statistics

    Successfully Write EHIPLANE Channel 0, Traffic GEN/CHK Register
pkt_clear_dropped_counter   , offset = 0x7, data = 0x3
    Successfully Write EHIPLANE Channel 0, Traffic GEN/CHK Register
pkt_clear_dropped_counter   , offset = 0x7, data = 0x0

Info: clearing the statistics

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

Info: Enabling the statistics

    Successfully Write EHIPLANE Channel 0, User Register
cntr_tx_config                        , offset = 0x845, data = 0x0
    Successfully Write EHIPLANE Channel 0, User Register
cntr_rx_config                        , offset = 0x945, data = 0x0
.
.
.
    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: Training PTP RX AIB deskew and waiting for PTP RX ready...

    Successfully Read  EHIPLANE Channel 0, PIO Register, offset = 0x0, data =
0x5
.
.
.
    Successfully Read  EHIPLANE Channel 0, PIO Register, offset = 0x0, data =
0x7

Info: PTP RX AIB Deskew Done


Info: clearing the traffic generator statistics

    Successfully Write EHIPLANE Channel 0, Traffic GEN/CHK Register
pkt_clear_dropped_counter   , offset = 0x7, data = 0x3
    Successfully Write EHIPLANE Channel 0, Traffic GEN/CHK Register
pkt_clear_dropped_counter   , offset = 0x7, data = 0x0

Info: clearing the statistics

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

Info: Enabling the statistics

    Successfully Write EHIPLANE Channel 0, User Register
cntr_tx_config                        , offset = 0x845, data = 0x0
    Successfully Write EHIPLANE Channel 0, User Register
cntr_rx_config                        , offset = 0x945, data = 0x0

Info: Accuracy measurement settings

    Successfully Read  RSFEC Register rsfec_cw_pos_rx_3                    ,
offset = 0x1cc, data = 0x2e

Info: RX slip count = 0xe


Info: UI Value = 0x0009EE01


Info: TX Extra Latency = 0x2c10247


Info: RX Extra Latency = 0x5d17496

    Successfully Write EHIPLANE Channel 0, User Register
tx_ptp_extra_latency                  , offset = 0xa0a, data = 0x2c102
.
.
.
    Successfully Read  EHIPLANE Channel 0, PIO Register, offset = 0xc, data =
0x101

Info: Iteration = 1 : TX Timestamp = 000000000011274d263fa436,  RX Timestamp =
000000000011274d263d4680,  Accuracy Difference = 2.36605835 ns

    Successfully Write EHIPLANE Channel 0, PIO Register, offset = 0xc, data =
0x0
    Successfully Write EHIPLANE Channel 0, Traffic GEN/CHK Register
pkt_tx_ctrl                 , offset = 0x10, data = 0x57
    Successfully Write EHIPLANE Channel 0, PIO Register, offset = 0xc, data =
0x102
    Successfully Write EHIPLANE Channel 0, Traffic GEN/CHK Register
pkt_tx_ctrl                , offset = 0x10, data = 0x55
    Successfully Read  EHIPLANE Channel 0, User Register
cntr_tx_64b_lo                        , offset = 0x816, data = 0x2
    Successfully Read  EHIPLANE Channel 0, User Register
cntr_rx_64b_lo                        , offset = 0x916, data = 0x2
    Successfully Read  EHIPLANE Channel 0, PIO Register, offset = 0x4, data =
0x17137aad
    Successfully Read  EHIPLANE Channel 0, PIO Register, offset = 0x5, data =
0x11284d
    Successfully Read  EHIPLANE Channel 0, PIO Register, offset = 0x6, data =
0x0
    Successfully Read  EHIPLANE Channel 0, PIO Register, offset = 0x8, data =
0x17111cf7
    Successfully Read  EHIPLANE Channel 0, PIO Register, offset = 0x9, data =
0x11284d
    Successfully Read  EHIPLANE Channel 0, PIO Register, offset = 0xa, data =
0x0
    Successfully Read  EHIPLANE Channel 0, PIO Register, offset = 0x7, data =
0x2
    Successfully Read  EHIPLANE Channel 0, PIO Register, offset = 0xc, data =
0x102
.
.
.

Info: Iteration = 1000 : TX Timestamp = 000000000003331b311e971d6,  RX Timestamp
= 000000000003331b311e9df10,  Accuracy Difference = -0.42666626 ns


Info: Stopping the traffic generator

    Successfully Write EHIPLANE Channel 0, PIO Register, offset = 0xc, data =
0x0
    Successfully Write EHIPLANE Channel 0, Traffic GEN/CHK Register
pkt_tx_ctrl                , offset = 0x10, data = 0x57
.
.
    Successfully Read  EHIPLANE Channel 0, User Register
cntr_rx_badlt_hi                      , offset = 0x969, data = 0x0

    Test End time is: 17:50:40
    Test End date is: 03/12/2019

Info: Closed JTAG Master Service



Info: Test <c3_elane_ptp_traffic_basic_test> Passed
```

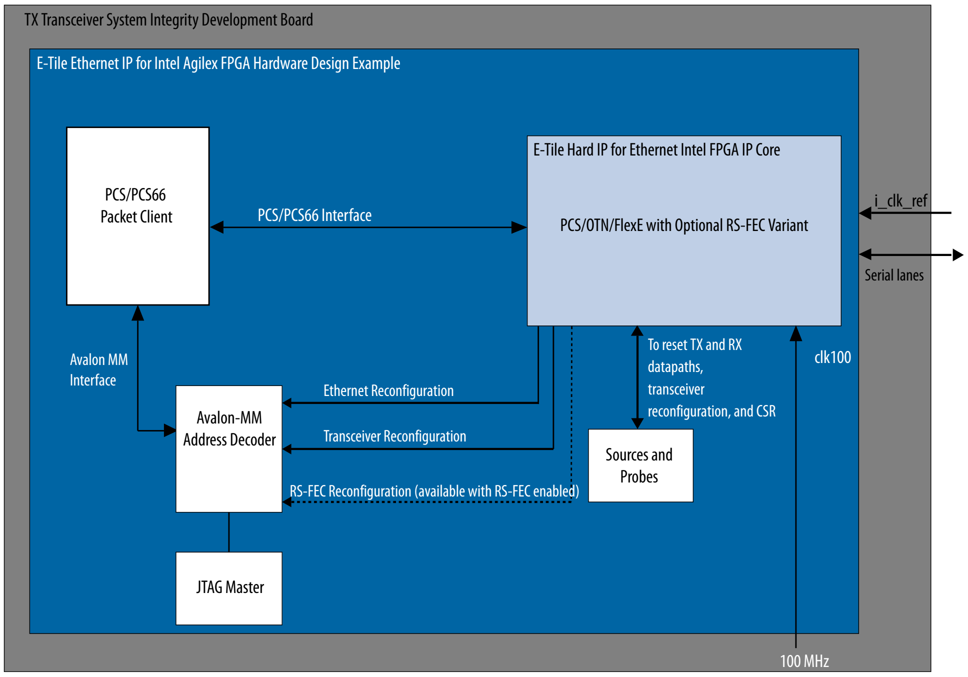
Related Information

- Compiling and Configuring the Design Example in Hardware on page 12
- Testing the E-tile Ethernet IP for Intel Agilex FPGA Hardware Design Example on page 13

Send Feedback

2.2.2.2. 10GE/25GE PCS Only with Optional RS-FEC Hardware Design Example Components

Figure 13. 10GE/25GE PCS Only with Optional RS-FEC Hardware Design Example High Level Block Diagram


TX Transceiver System Integrity Development Board
E-Tile Ethernet IP for Intel Agilex FPGA Hardware Design Example
PCS/PCS66 Packet Client
PCS/PCS66 Interface
E-Tile Hard IP for Ethernet Intel FPGA IP Core
PCS/OTN/FlexE with Optional RS-FEC Variant
i_clk_ref
Serial lanes
Avalon MM Interface
Avalon-MM Address Decoder
Ethernet Reconfiguration
Transceiver Reconfiguration
RS-FEC Reconfiguration (available with RS-FEC enabled)
To reset TX and RX datapaths, transceiver reconfiguration, and CSR
clk100
Sources and Probes
JTAG Master
100 MHz

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

- E-tile Ethernet IP for Intel Agilex FPGA 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.

Result from `c3_elane_pcsonly_traffic_basic_test.log` file:

```
Info: Set JTAG Master Service Path


Info: Opened JTAG Master Service

    Test Start time is: 12:15:27
    Test Start date is: 03/12/2019


Info: Read all ELANE CSR registers

    Successfully Read  EHIPLANE Channel 0, User Register
phy_revid                              , offset = 0x300, data = 0x11112015
.
.
.
    Successfully Read  EHIPLANE Channel 0, User Register
phy_ehip_csr_soft_reset               , offset = 0x310, data = 0x0
```

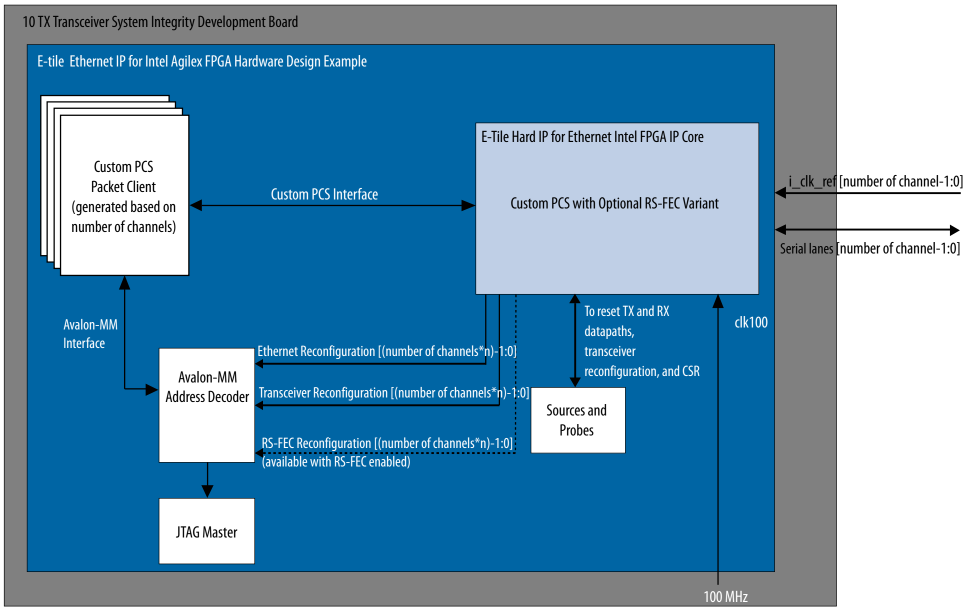
```
    C3 ELANE Channel 0 System Reset is successfully


Info: Stopping the Channel 0 XGMII traffic generator

    Successfully Read  EHIPLANE Channel 0, XGMII Traffic GEN/CHK Register,
offset = 0x0, data = 0x0
    Successfully Write EHIPLANE Channel 0, XGMII Traffic GEN/CHK Register,
offset = 0x0, data = 0x0

Info: Starting the Channel 0 XGMII traffic generator

    Successfully Write EHIPLANE Channel 0, XGMII Traffic GEN/CHK Register,
offset = 0x0, data = 0x1

Info: Comparing the Channel 0 XGMII traffic checker results

    Successfully Read  EHIPLANE Channel 0, XGMII Traffic GEN/CHK Register,
offset = 0x2, data = 0x2

Info: Channel 0, Iteration 1 is completed successfully
.
.
.
Info: Starting the Channel 0 XGMII traffic generator

    Successfully Write EHIPLANE Channel 0, XGMII Traffic GEN/CHK Register,
offset = 0x0, data = 0x1

Info: Comparing the Channel 0 XGMII traffic checker results

    Successfully Read  EHIPLANE Channel 0, XGMII Traffic GEN/CHK Register,
offset = 0x2, data = 0x2

Info: Channel 0, Iteration 4 is completed successfully

Info: Stopping the Channel 0 XGMII traffic generator

    Successfully Read  EHIPLANE Channel 0, XGMII Traffic GEN/CHK Register,
offset = 0x0, data = 0x1
    Successfully Write EHIPLANE Channel 0, XGMII Traffic GEN/CHK Register,
offset = 0x0, data = 0x0

Info: Starting the Channel 0 XGMII traffic generator

    Successfully Write EHIPLANE Channel 0, XGMII Traffic GEN/CHK Register,
offset = 0x0, data = 0x1

Info: Comparing the Channel 0 XGMII traffic checker results

    Successfully Read  EHIPLANE Channel 0, XGMII Traffic GEN/CHK Register,
offset = 0x2, data = 0x2

Info: Channel 0, Iteration 5 is completed successfully

Info: Channel 0 test is completed

    Test End time is: 12:17:08
    Test End date is: 03/12/2019

Info: Closed JTAG Master Service

Info: Test <c3_elane_pcsonly_traffic_basic_test> Passed
```

Related Information

- Compiling and Configuring the Design Example in Hardware on page 12
- Testing the E-tile Ethernet IP for Intel Agilex FPGA Hardware Design Example on page 13

2.2.2.3. 10GE/25GE Custom PCS with Optional RS-FEC Hardware Design Example

Figure 14. 10GE/25GE Custom PCS with Optional RS-FEC Hardware Design Example High Level Block Diagram


10 TX Transceiver System Integrity Development Board
E-tile Ethernet IP for Intel Agilex FPGA Hardware Design Example
Custom PCS Packet Client (generated based on number of channels)
Custom PCS Interface
E-Tile Hard IP for Ethernet Intel FPGA IP Core
Custom PCS with Optional RS-FEC Variant
i_clk_ref [number of channel-1:0]
Serial lanes [number of channel-1:0]
Avalon-MM Interface
Avalon-MM Address Decoder
Ethernet Reconfiguration [(number of channels*n)-1:0]
Transceiver Reconfiguration [(number of channels*n)-1:0]
RS-FEC Reconfiguration [(number of channels*n)-1:0] (available with RS-FEC enabled)
To reset TX and RX datapaths, transceiver reconfiguration, and CSR
Sources and Probes
clk100
JTAG Master
100 MHz

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

- E-tile Ethernet IP for Intel Agilex FPGA 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.

Result from `c3_elane_pcsonly_traffic_basic_test.log` file:

```
Info: Set JTAG Master Service Path


Info: Opened JTAG Master Service

    Test Start time is: 05:47:37
    Test Start date is: 03/21/2019


Info: Read all ELANE CSR registers

    Successfully Read  EHIPLANE Channel 0, User Register
phy_revid                              , offset = 0x300, data = 0x11112015
    Successfully Read  EHIPLANE Channel 0, User Register
phy_scratch                           , offset = 0x301, data = 0x0
.
.
.
    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: Stopping the Channel 0 XGMII traffic generator

    Successfully Read  EHIPLANE Channel 0, XGMII Traffic GEN/CHK Register,
offset = 0x0, data = 0x0
    Successfully Write EHIPLANE Channel 0, XGMII Traffic GEN/CHK Register,
offset = 0x0, data = 0x0

Info: Starting the Channel 0 XGMII traffic generator

    Successfully Write EHIPLANE Channel 0, XGMII Traffic GEN/CHK Register,
offset = 0x0, data = 0x1

Info: Comparing the Channel 0 XGMII traffic checker results

    Successfully Read  EHIPLANE Channel 0, XGMII Traffic GEN/CHK Register,
offset = 0x2, data = 0x2

Info: Channel 0, Iteration 1 is completed successfully
.
.
.
Info: Channel 0, Iteration 5 is completed successfully


Info: Channel 0 test is completed

    Successfully Write EHIPLANE Channel 1, User Register
phy_ehip_csr_soft_reset               , offset = 0x310, data = 0x0
    Successfully Write EHIPLANE Channel 1, User Register
phy_ehip_csr_soft_reset               , offset = 0x310, data = 0x1
    Successfully Write EHIPLANE Channel 1, User Register
phy_ehip_csr_soft_reset               , offset = 0x310, data = 0x3
    Successfully Write EHIPLANE Channel 1, User Register
phy_ehip_csr_soft_reset               , offset = 0x310, data = 0x7
    Successfully Read  EHIPLANE Channel 1, User Register
phy_ehip_csr_soft_reset               , offset = 0x310, data = 0x7
    Successfully Write EHIPLANE Channel 1, User Register
phy_ehip_csr_soft_reset               , offset = 0x310, data = 0x6
    Successfully Write EHIPLANE Channel 1, User Register
phy_ehip_csr_soft_reset               , offset = 0x310, data = 0x4
    Successfully Write EHIPLANE Channel 1, User Register
phy_ehip_csr_soft_reset               , offset = 0x310, data = 0x0
    Successfully Read  EHIPLANE Channel 1, User Register
phy_ehip_csr_soft_reset               , offset = 0x310, data = 0x0

    C3 ELANE Channel 1 System Reset is successfully


Info: Stopping the Channel 1 XGMII traffic generator

    Successfully Read  EHIPLANE Channel 1, XGMII Traffic GEN/CHK Register,
offset = 0x0, data = 0x0
    Successfully Write EHIPLANE Channel 1, XGMII Traffic GEN/CHK Register,
offset = 0x0, data = 0x0

Info: Starting the Channel 1 XGMII traffic generator

    Successfully Write EHIPLANE Channel 1, XGMII Traffic GEN/CHK Register,
offset = 0x0, data = 0x1

Info: Comparing the Channel 1 XGMII traffic checker results

    Successfully Read  EHIPLANE Channel 1, XGMII Traffic GEN/CHK Register,
offset = 0x2, data = 0x2

Info: Channel 1, Iteration 1 is completed successfully
.
.
.
Info: Channel 1, Iteration 5 is completed successfully
```

```
Info: Channel 1 test is completed

    Successfully Read  RSFEC Register rsfec_top_rx_cfg                     ,
offset = 0x14, data = 0x11
    Successfully Read  RSFEC Register arbiter_base_cfg                      ,
offset = 0x0, data = 0x1
    Successfully Read  RSFEC Register rsfec_top_clk_cfg                    ,
offset = 0x4, data = 0x304
    Successfully Read  RSFEC Register rsfec_top_tx_cfg                     ,
offset = 0x10, data = 0x6611
    Successfully Write RSFEC Register rsfec_top_tx_cfg                     ,
offset = 0x10, data = 0x10001666
    Successfully Read  RSFEC Register rsfec_top_tx_cfg                     ,
offset = 0x10, data = 0x10001666
    Successfully Write RSFEC Register rsfec_top_tx_cfg                     ,
offset = 0x10, data = 0x6611
    Successfully Read  RSFEC Register rsfec_top_tx_cfg                     ,
offset = 0x10, data = 0x6611

    Test End time is: 05:51:01
    Test End date is: 03/21/2019

Info: Closed JTAG Master Service



Info: Test <c3_elane_pcsonly_traffic_basic_test> Passed
```

Related Information

- Compiling and Configuring the Design Example in Hardware on page 12
- Testing the E-tile Ethernet IP for Intel Agilex FPGA Hardware Design Example on page 13

2.2.3. 10GE/25GE Design Example Interface Signals

The following signals are hardware design example signals for all 10GE/25GE variants.

Table 6. 10GE/25GE Hardware Design Example Interface Signals

Signal	Direction	Description
`clk100`	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.
`cpu_resetn`	Input	Resets the IP core. Active low. Drives the global hard reset `csr_reset_n` to the IP core.
`o_tx_serial[(number of channels-1:0]`	Output	Transceiver PHY output serial data.
`i_rx_serial[number of channels-1:0]`	Input	Transceiver PHY input serial data.

Related Information

E-tile Ethernet IP for Intel Agilex FPGA Interfaces and Signal Descriptions

2.2.4. 10GE/25GE Design Examples Registers

Table 7. E-tile Ethernet IP for Intel Agilex FPGA Hardware Design Examples Register Map

Lists the memory mapped register ranges for all 10GE/25GE hardware design example variants. You access these registers with the `reg_read` and `reg_write` functions in the System Console.

Channel Number	Word Offset	Register Type
0	0x000000	KR4 registers
	0x000300	RX PCS registers
	0x000400	TX MAC registers
	0x000500	RX MAC registers
	0x000800	TX Statistics Counter registers
	0x000900	RX Statistics Counter registers
	0x001000	Packet Client and Packet Generator registers
	0x002000	PTP monitoring registers
	0x010000	RS-FEC configuration registers
	0x100000	Transceiver registers
1	0x200000	KR4 registers
	0x200300	RX PCS registers
	0x200400	TX MAC registers
	0x200500	RX MAC registers
	0x200800	TX Statistics Counter registers
	0x200900	RX Statistics Counter registers
	0x201000	Packet Client registers
	0x202000	PTP monitoring registers
	0x210000	RS-FEC configuration registers
	0x300000	Transceiver registers
2	0x400000	KR4 registers
	0x400300	RX PCS registers
	0x400400	TX MAC registers
	0x400500	RX MAC registers
	0x400800	TX Statistics Counter registers
	0x400900	RX Statistics Counter registers
	0x401000	Packet Client registers
	0x402000	PTP monitoring registers
	0x410000	RS-FEC configuration registers
	0x500000	Transceiver registers
3	0x600000	KR4 registers

continued...

Send Feedback

Channel Number	Word Offset	Register Type
	0x600300	RX PCS registers
	0x600400	TX MAC registers
	0x600500	RX MAC registers
	0x600800	TX Statistics Counter registers
	0x600900	RX Statistics Counter registers
	0x601000	Packet Client registers
	0x602000	PTP monitoring registers
	0x610000	RS-FEC configuration registers
	0x700000	Transceiver registers

Table 8. Packet Client Registers

You can customize the E-tile Ethernet IP for Intel Agilex 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]	Specifies 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]	Specifies 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 upper 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

Table 9. MII Packet Generator Registers

Addr	Name	Bit	Description	HW Reset Value	Access
0x0	XGMII_PKTGEN_START	[0]	Start or stop packet generator for MII interface. Valid for custom PCS, OTN, FlexE, and PCS_only modes. • 1’b0: Stop • 1’b1: Start	0	RW
0x2	XGMII_PKTGEN_PASS	[1]	Checks for pass or fail status of MII interface packet generation. • 1’b0: Fail • 1’b1: Pass	0	RO

Related Information

E-tile Ethernet IP for Intel Agilex FPGA core register descriptions

2.3. 100GE with Optional RS-FEC Design Example

The 100GE design example demonstrates an Ethernet solution for Intel Agilex devices using the E-tile Ethernet IP for Intel Agilex FPGA core with the following variants:

Table 10. Supported Design Example Variants for 100GE

Variant	Design Example Support
Non-PTP MAC+PCS with Optional RS-FEC (528,514)/(544,514) • For (528,514) RS-FEC variant, the design example consists of 4 transceiver channels • For (544,514) RS-FEC variant, the design example consists of 2 transceiver channels	Simulation and compilation-only project
MAC+PCS with Optional RS-FEC and PTP (528,514) • For (528,514) RS-FEC variant, the design example consists of 4 transceiver channels	Simulation and compilation-only project
PCS Only with Optional RS-FEC (528,514)/(544,514) • For (528,514) RS-FEC variant, the design example consists of 4 transceiver channels • For (544,514) RS-FEC variant, the design example consists of 2 transceiver channels	Simulation and compilation-only project
OTN with Optional RS-FEC (528,514)/(544,514) • For (528,514) RS-FEC variant, the design example consists of 4 transceiver channels • For (544,514) RS-FEC variant, the design example consists of 2 transceiver channels	Simulation and compilation-only project
FlexE with Optional RS-FEC (528,514)/(544,514) • For (528,514) RS-FEC variant, the design example consists of 4 transceiver channels • For (544,514) RS-FEC variant, the design example consists of 2 transceiver channels	Simulation and compilation-only project

Note: The E-Tile Ethernet IP for Intel Agilex FPGA 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.

2.3.1. Simulation Design Examples

2.3.1.1. Non-PTP E-tile Ethernet IP for Intel Agilex FPGA 100GE MAC+PCS with Optional RS-FEC Simulation Design Example

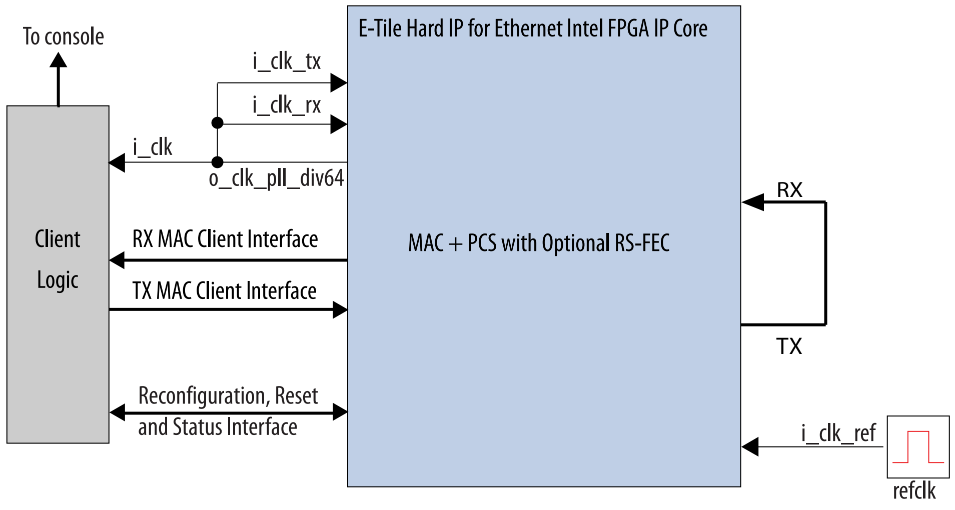
The simulation block diagram below is generated using the following settings in the IP parameter editor:

1. Under the **IP** tab:
   a. **Single 100GE with optional RSFEC** or **100GE or 1 to 4 channel 10GE/25GE with optional RSFEC and PTP** as the core variant.
   b. **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.
   c. **Enable RSFEC** to use the RS-FEC feature.

Note: The RS-FEC feature is only available when you select **100GE or 1 to 4 channel 10GE/25GE with optional RSFEC and PTP** as the core variant.

2. Under the **100GE** tab:
   a. **100G** as the Ethernet rate.
   b. **MAC+PCS** as **Select Ethernet IP Layers** to use instantiate MAC and PCS layer or **MAC+PCS+(528,514)RSFEC**/**MAC+PCS+(544,514)RSFEC** to instantiate MAC and PCS with RS-FEC feature.
3. **Enable asynchronous adapter clocks** to use the asynchronous adapter feature.

Figure 15. Simulation Block Diagram for E-tile Ethernet IP for Intel Agilex FPGA 100GE MAC+PCS with Optional RS-FEC Design Example



Note: If **Enable asynchronous adapter clocks** is enabled, the `o_clk_div66` feeds the `i_clk_tx` and `i_clk_rx` clocks.

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 16,384 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 RS-FEC 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

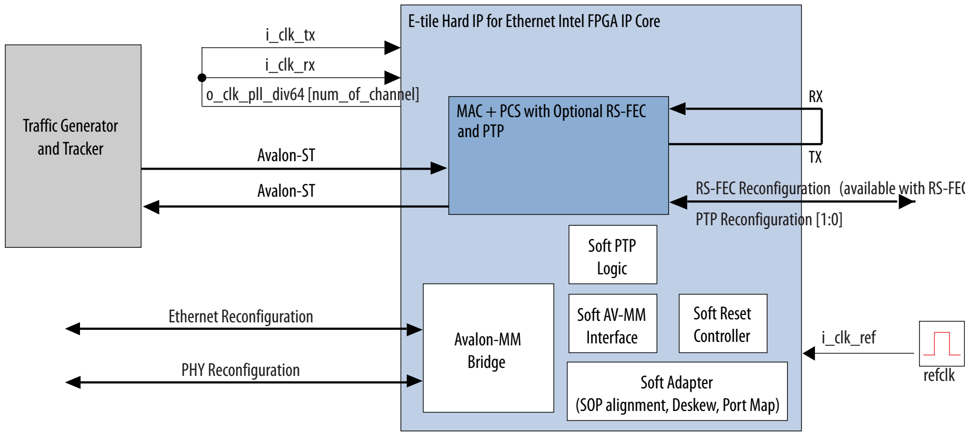
Simulating the E-tile Ethernet IP for Intel Agilex FPGA Design Example Testbench on page 10

2.3.1.2. E-tile Ethernet IP for Intel Agilex FPGA 100GE MAC+PCS with Optional RS-FEC and PTP Simulation Design Example

The simulation block diagram below is generated using the following settings in the IP parameter editor:

1. Under the **IP** tab:
    a. **100GE or 1 to 4 channel 10GE/25GE with optional RSFEC and PTP** as the core variant.
    b. **100GE Channel** as **Active channel(s) at startup**.
    c. **Enable IEEE 1588 PTP**.
    d. **Enable RSFEC** to use the RS-FEC feature.
2. Under the **100GE** tab:
    a. **100G** as the Ethernet rate.
    b. **MAC+1588PTP+PCS+(528,514)RSFEC** as the Ethernet IP layer.

Figure 16. Simulation Block Diagram for E-tile Ethernet IP for Intel Agilex FPGA 100GE MAC+PCS with Optional RS-FEC and PTP Design Example


E-tile Hard IP for Ethernet Intel FPGA IP Core
i_clk_tx
i_clk_rx
o_clk_pll_div64 [num_of_channel]
Traffic Generator and Tracker
MAC + PCS with Optional RS-FEC and PTP
RX
TX
Avalon-ST
Avalon-ST
RS-FEC Reconfiguration (available with RS-FEC
PTP Reconfiguration [1:0]
Soft PTP Logic
Ethernet Reconfiguration
Avalon-MM Bridge
Soft AV-MM Interface
Soft Reset Controller
i_clk_ref
refclk
PHY Reconfiguration
Soft Adapter (SOP alignment, Deskew, Port Map)

In this design example, 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 16,384 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 100GE, MAC+PCS, RS-FEC, PTP IP core variation.

```
# o_tx_lanes_stable is 1 at time             346295000
# waiting for tx_dll_lock....
# TX DLL LOCK is 1 at time             405180363
# waiting for tx_transfer_ready....
# TX transfer ready is 1 at time             405500235
# waiting for rx_transfer_ready....
# RX transfer ready is 1 at time             416575803
# EHIP PLD Ready out is 1 at time             416632000
# EHIP reset out is 0 at time             416768000
# EHIP reset ack is 0 at time             416844540
# EHIP TX reset out is 0 at time             417184000
# EHIP TX reset ack is 0 at time             468265476
# waiting for EHIP Ready....
# EHIP READY is 1 at time             468389597
# EHIP RX reset out is 0 at time             470288000
# waiting for rx reset ack....
# EHIP RX reset ack is 0 at time             470301064
# Waiting for RX Block Lock
# EHIP RX Block Lock  is high at time             511027716
# Waiting for AM lock
# EHIP RX AM Lock  is high at time             511027716
# Waiting for RX alignment
# RX deskew locked
# RX lane aligmnent locked
# Configure TX extra latency
# ====> writedata = 0004267a
#
# Configure RX extra latency
# ====> writedata = 8003af52
#
# Waiting for TX PTP Ready
# TX PTP ready
# Waiting for RSFEC alignment locked
# Reading rsfec_ln_mapping_rx_0
# rsfec_ln_mapping_rx_0 = 32'h0
.
.
.
# Reading rsfec_cw_pos_rx_3
# rsfec_cw_pos_rx_3 = 32'h7dd
# min skew value = 32'h1
# lane_skew_adjust = 32'h1
# Tlat_final = 32'h4
# Generate VL offset data
# before-rotation: VL[PL] 0[0], deskew_delay = 4 UI, vl_offset_bits = 4
# After rotation: VL_OFFSET for RVL[PL] 4[0] = 0 ns 27b8 Fns, Sign bit= 0
.
.
.
# before-rotation: VL[PL] 19[0], deskew_delay = 4 UI, vl_offset_bits = 8
# before-rotation: VL[PL] 19[0], deskew_delay = 4 UI, vl_offset_bits_shifted =
-322
# After rotation: VL_OFFSET for RVL[PL] 3[0] = c ns 7d5d Fns, Sign bit= 1
# Writing VL offset data for VL 0
# ====> writedata = 00000004
#
# ====> writedata = 000027b8
.
.
.
# Writing VL offset data for VL 19
# ====> writedata = 00000003
#
# ====> writedata = 800c7d5d
#
# Waiting for RX PTP Ready
# RX PTP ready
# ** 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.
# **
```

Send Feedback

```
# ******************************************
# ** Note: $finish    : ./basic_avl_tb_top.sv(674)
#    Time: 530600 ns  Iteration: 0  Instance: /basic_avl_tb_top
```

Related Information

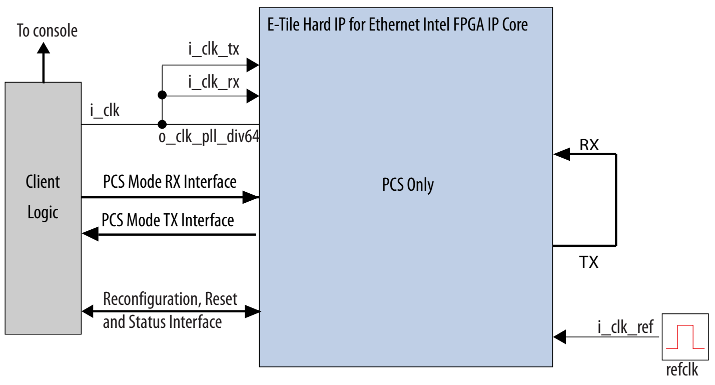
Simulating the E-tile Ethernet IP for Intel Agilex FPGA Design Example Testbench on page 10

2.3.1.3. E-tile Ethernet IP for Intel Agilex FPGA 100GE PCS Only with Optional RS-FEC Simulation Design Example

The simulation block diagram below is generated using the following settings in the IP parameter editor:

1. Under the **IP** tab:
   a. **Single 100GE with optional RSFEC** or **100GE or 1 to 4 channel 10GE/25GE with optional RSFEC and PTP** as the core variant.
   b. **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.
2. Under the **100GE** tab:
   a. **100G** as the Ethernet rate.
   b. **PCS_Only**, **PCS+(528,514)RSFEC**, or **PCS+(544,514)RSFEC** as the Ethernet IP layer.

Figure 17. Simulation Block Diagram for E-tile Ethernet IP for Intel Agilex FPGA 100GE PCS Only Design Example



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 16,384 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             354775000
waiting for tx_dll_lock....
TX DLL LOCK is 1 at time             413726943
waiting for tx_transfer_ready....
TX transfer ready is 1 at time             414046815
waiting for rx_transfer_ready....
RX transfer ready is 1 at time             425122383
EHIP PLD Ready out is 1 at time             425184000
EHIP reset out is 0 at time             425320000
EHIP reset ack is 0 at time             426016853
EHIP TX reset out is 0 at time             426232000
EHIP TX reset ack is 0 at time             476830347
waiting for EHIP Ready....
EHIP READY is 1 at time             476910363
EHIP RX reset out is 0 at time             478680000
waiting for rx reset ack....
EHIP RX reset ack is 0 at time             478777403
Waiting for RX Block Lock
EHIP Rx Block Lock  is high at time             481444603
Waiting for AM lock
EHIP Rx am Lock  is high at time             482711523
Waiting for RX alignment
RX deskew locked
RX lane alignmnent locked
Sending Packets and Receiving Packets
====> writedata = 00000001

====>MATCH!     ReaddataValid = 1 Readdata = 00000053 Expected_Readdata =
00000053

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

Related Information

Simulating the E-tile Ethernet IP for Intel Agilex FPGA Design Example Testbench on page 10

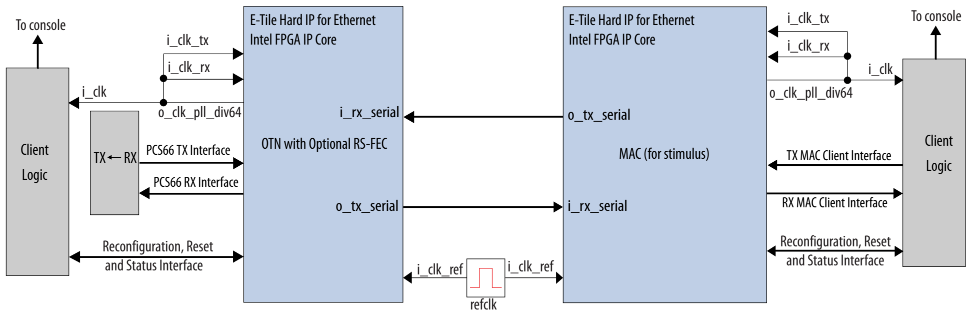
2.3.1.4. E-tile Ethernet IP for Intel Agilex FPGA 100GE OTN with Optional RS-FEC Simulation Design Example

The simulation block diagram below is generated using the following settings in the IP parameter editor:

1. Under the **IP** tab:
    a. **Single 100GE with optional RSFEC** or **100GE or 1 to 4 channel 10GE/ 25GE with optional RSFEC and PTP** as the core variant.
    b. **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.
2. Under the **100GE** tab:
    a. **100G** as the Ethernet rate.
    b. **OTN**, **OTN+(528,514)RSFEC**, or **OTN+(544,514)RSFEC** as the Ethernet IP layer.

Note: The E-tile Ethernet IP for Intel Agilex FPGA 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 18. Simulation Block Diagram for E-tile Ethernet IP for Intel Agilex FPGA 100GE OTN Design Example



The testbench sends traffic through the IP core with OTN mode, exercising the transmit side and receive interface using a separate E-tile Ethernet IP for Intel Agilex 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           9...
# test_dut: ** Sending Packet          10...
# test_dut: ** Received Packet          1...

.
.
.
# test_dut: ** Received Packet          9...
# test_dut: ** Received Packet         10...
# test_dut: **
# test_dut: ** Testbench complete.
# test_dut: **
# test_dut: *****************************************
```

Send Feedback

Related Information

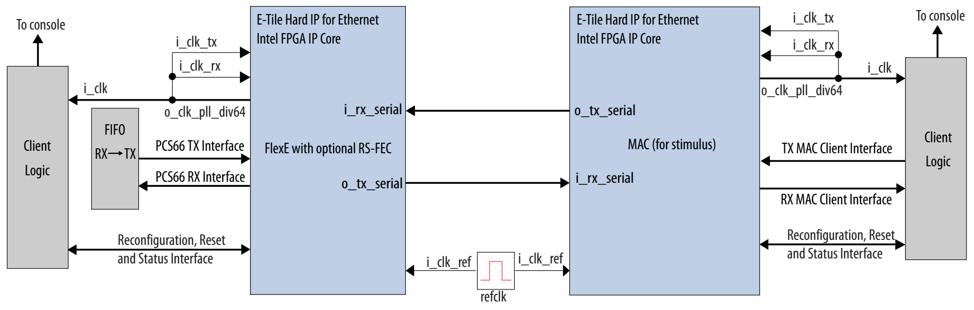
Simulating the E-tile Ethernet IP for Intel Agilex FPGA Design Example Testbench on page 10

2.3.1.5. E-tile Ethernet IP for Intel Agilex FPGA 100GE FlexE with Optional RS-FEC Simulation Design Example

The simulation block diagram below is generated using the following settings in the IP parameter editor:

1. Under the **IP** tab:
    a. **Single 100GE with optional RSFEC** or **100GE or 1 to 4 channel 10GE/25GE with optional RSFEC and PTP** as the core variant.
    b. **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.
2. Under the **100GE** tab:
    a. **100G** as the Ethernet rate.
    b. **FlexE**, **FlexE+(528,514)RSFEC**, or **FlexE+(544,514)RSFEC** as the Ethernet IP layer.

Figure 19. Simulation Block Diagram for E-tile Ethernet IP for Intel Agilex FPGA 100GE FlexE 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          9...
# test_dut: ** Sending Packet         10...
# test_dut: ** Received Packet         1...
.
.
.
# test_dut: ** Received Packet         9...
# test_dut: ** Received Packet        10...
# test_dut: **
# test_dut: ** Testbench complete.
# test_dut: **
# test_dut: *****************************************
```

Send Feedback

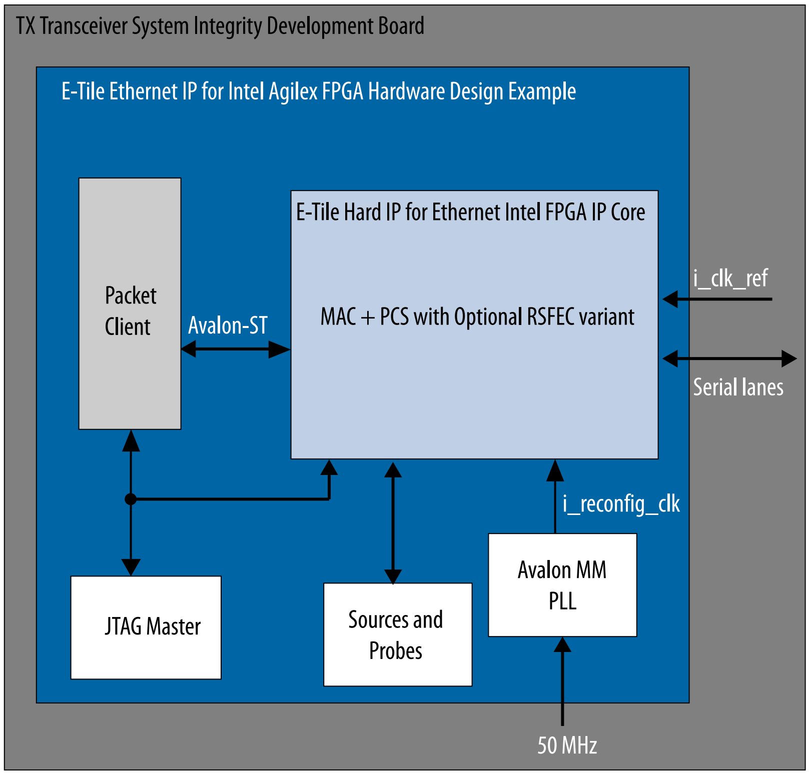
Related Information

Simulating the E-tile Ethernet IP for Intel Agilex FPGA Design Example Testbench on page 10

2.3.2. Hardware Design Examples

2.3.2.1. 100GE MAC+PCS with Optional RS-FEC and PMA Adaptation Flow Hardware Design Example Components

Figure 20. 100GE MAC+PCS with Optional RS-FEC Hardware Design Example High Level Block Diagram


TX Transceiver System Integrity Development Board
E-Tile Ethernet IP for Intel Agilex FPGA Hardware Design Example
Packet Client
Avalon-ST
E-Tile Hard IP for Ethernet Intel FPGA IP Core
MAC + PCS with Optional RSFEC variant
i_clk_ref
Serial lanes
i_reconfig_clk
JTAG Master
Sources and Probes
Avalon MM PLL
50 MHz

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

- E-tile Ethernet IP for Intel Agilex FPGA core. The IP core consists of 4 channels if you select (528,514) RS-FEC option, and 2 transceiver channels if you select (544,514) RS-FEC option and enabled asynchronous adapter.
- 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 you use the `run_test` or the `run_test_pam4` commands to run the hardware test in the design examples, the script enables internal loopback.

Send Feedback

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.

The following sample output illustrates a successful hardware test run for 100GE, MAC +PCS with (528,514) RS-FEC variation:

```
% run_test
--- Turning off packet generation ----
--------------------------------------
--------- Enabling loopback ----------
--------------------------------------
--- Wait for RX clock to settle... ---
--------------------------------------
-------- Printing PHY status ---------
--------------------------------------
 RX PHY Register Access: Checking Clock Frequencies (KHz)
    REFCLK         :0 (KHZ)
    TXCLK         :40285  (KHZ)
    RXCLK         :40284  (KHZ)
    TXRSCLK     :0  (KHZ)
    RXRSCLK     :0  (KHZ)
 RX PHY Status Polling
 Rx Frequency Lock Status     0x0000000f
 Mac Clock in OK Condition?   0x00000001
 Rx Frame Error               0x00000000
 Rx PHY Fully Aligned?        0x00000001
 Rx AM LOCK Condition?        0x00000001
 Rx Lanes Deskewed Condition? 0x00000001
---- Clearing MAC stats counters -----
--------------------------------------
--------- Sending packets... ---------
--------------------------------------
----- Reading MAC stats counters -----
--------------------------------------

================================================================================
==========
                        STATISTICS FOR BASE 0x000900
(Rx)

================================================================================
==========
Fragmented Frames                 : 0
Jabbered Frames                   : 0
Any Size with FCS Err Frame       : 0
Right Size with FCS Err Fra       : 0
Multicast data  Err Frames        : 0
Broadcast data Err  Frames        : 0
Unicast data Err  Frames          : 0
Multicast control  Err Frame      : 0
Broadcast control Err  Frame      : 0
Unicast control Err  Frames       : 0
Pause control Err  Frames         : 0
64 Byte Frames                    : 7190
65 - 127 Byte Frames              : 6965
128 - 255 Byte Frames             : 14338
256 - 511 Byte Frames             : 28779
512 - 1023 Byte Frames            : 57548
1024 - 1518 Byte Frames           : 55880
1519 - MAX Byte Frames            : 0
> MAX Byte Frames                 : 1669560
Rx Frame Starts                   : 1840260
Multicast data  OK  Frame         : 0
Broadcast data OK   Frame         : 0
Unicast data OK   Frames          : 1836399
Multicast Control Frames          : 0
Broadcast Control Frames          : 0
Unicast Control Frames            : 0
```

```
Pause Control Frames              : 0

================================================================================
==========
                        STATISTICS FOR BASE 0x000800
(Tx)

================================================================================
==========
Fragmented Frames                 : 0
Jabbered Frames                   : 0
Any Size with FCS Err Frame       : 0
Right Size with FCS Err Fra       : 0
Multicast data  Err Frames        : 0
Broadcast data Err  Frames        : 0
Unicast data Err  Frames          : 0
Multicast control  Err Frame      : 0
Broadcast control Err  Frame      : 0
Unicast control Err  Frames       : 0
Pause control Err  Frames         : 0
64 Byte Frames                    : 7190
65 - 127 Byte Frames              : 6965
128 - 255 Byte Frames             : 14338
256 - 511 Byte Frames             : 28779
512 - 1023 Byte Frames            : 57548
1024 - 1518 Byte Frames           : 55880
1519 - MAX Byte Frames            : 0
> MAX Byte Frames                 : 1669560
Tx Frame Starts                   : 1840260
Multicast data  OK  Frame         : 0
Broadcast data OK   Frame         : 0
Unicast data OK   Frames          : 1836399
Multicast Control Frames          : 0
Broadcast Control Frames          : 0
Unicast Control Frames            : 0
Pause Control Frames              : 0
```

The following sample output illustrates a successful hardware test run for 100GE, MAC +PCS with (544,512) RS-FEC variation:

```
% run_test_pam4
--- Turning off packet generation ----
--------------------------------------
--------- Enabling loopback ----------
--------------------------------------
--- Performing PMA adaptation... ---
--------------------------------------
------------ Starting PMA Adaptation ------------
------- Checking PMA Adaptation Status-------
------- PMA Adaptation Done for ch0x0 -------
------- PMA Adaptation Done for ch0x2 -------
------------ Applying TX and RX Reset ----------
 wait for phy lock=50, locked=1
--Iteration:0 - PMA Adaptaion is Successful--
--- Wait for RX clock to settle... ---
--------------------------------------
-------- Printing PHY status ---------
--------------------------------------
 RX PHY Register Access: Checking Clock Frequencies (KHz)
    REFCLK         :0 (KHZ)
    TXCLK         :41504  (KHZ)
    RXCLK         :41505  (KHZ)
    TXRSCLK     :0  (KHZ)
    RXRSCLK     :0  (KHZ)
 RX PHY Status Polling
 Rx Frequency Lock Status     0x0000000f
 Mac Clock in OK Condition?   0x00000001
 Rx Frame Error               0x00000000
 Rx AM LOCK Condition?        0x00000001
 Rx Lanes Deskewed Condition? 0x00000001
```

```
---- Clearing MAC stats counters -----
--------------------------------------
--------- Sending packets... ---------
--------------------------------------
----- Reading MAC stats counters -----
--------------------------------------

================================================================================
==========
                        STATISTICS FOR BASE 0x000900
(Rx)

================================================================================
==========
Fragmented Frames                : 0
Jabbered Frames                  : 0
Any Size with FCS Err Frame      : 0
Right Size with FCS Err Fra      : 0
Multicast data  Err Frames       : 0
Broadcast data Err  Frames       : 0
Unicast data Err  Frames         : 0
Multicast control  Err Frame     : 0
Broadcast control Err  Frame     : 0
Unicast control Err  Frames      : 0
Pause control Err  Frames        : 0
64 Byte Frames                   : 7114
65 - 127 Byte Frames             : 6925
128 - 255 Byte Frames            : 14418
256 - 511 Byte Frames            : 28563
512 - 1023 Byte Frames           : 57313
1024 - 1518 Byte Frames          : 56067
1519 - MAX Byte Frames           : 0
> MAX Byte Frames                : 1670068
Rx Frame Starts                  : 1840468
Multicast data  OK  Frame        : 0
Broadcast data OK   Frame        : 0
Unicast data OK   Frames         : 1836559
Multicast Control Frames         : 0
Broadcast Control Frames         : 0
Unicast Control Frames           : 0
Pause Control Frames             : 0

================================================================================
==========
                        STATISTICS FOR BASE 0x000800
(Tx)

================================================================================
==========
Fragmented Frames                : 0
Jabbered Frames                  : 0
Any Size with FCS Err Frame      : 0
Right Size with FCS Err Fra      : 0
Multicast data  Err Frames       : 0
Broadcast data Err  Frames       : 0
Unicast data Err  Frames         : 0
Multicast control  Err Frame     : 0
Broadcast control Err  Frame     : 0
Unicast control Err  Frames      : 0
Pause control Err  Frames        : 0
64 Byte Frames                   : 7114
65 - 127 Byte Frames             : 6925
128 - 255 Byte Frames            : 14418
256 - 511 Byte Frames            : 28563
512 - 1023 Byte Frames           : 57313
1024 - 1518 Byte Frames          : 56067
1519 - MAX Byte Frames           : 0
> MAX Byte Frames                : 1670068
Tx Frame Starts                  : 1840468
Multicast data  OK  Frame        : 0
Broadcast data OK   Frame        : 0
```

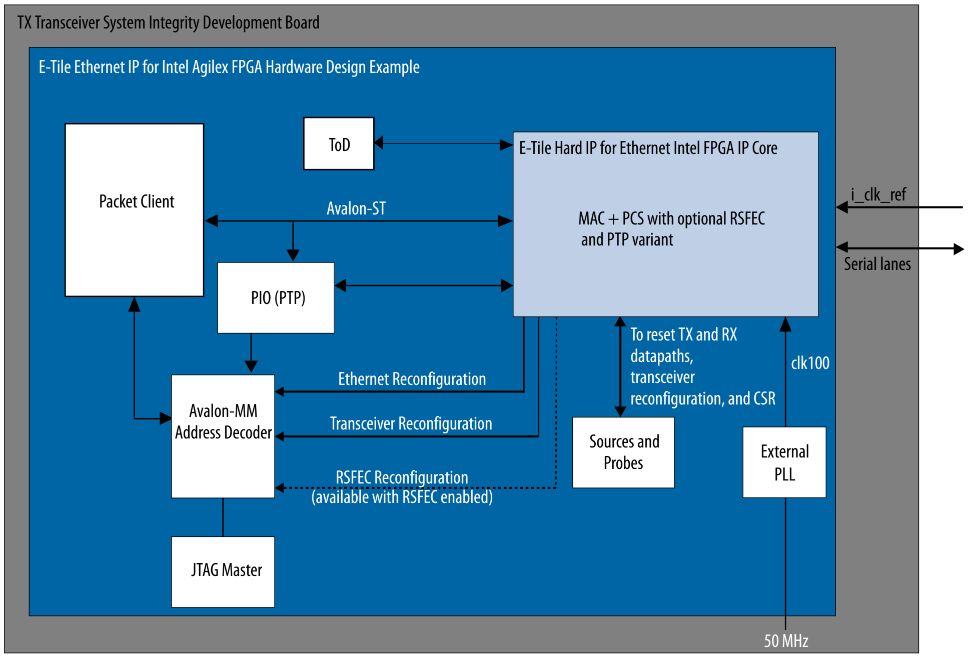
```
Unicast data OK   Frames          : 1836559
Multicast Control Frames          : 0
Broadcast Control Frames          : 0
Unicast Control Frames            : 0
Pause Control Frames              : 0
```

Related Information

- Compiling and Configuring the Design Example in Hardware on page 12
- Testing the E-tile Ethernet IP for Intel Agilex FPGA Hardware Design Example on page 13

2.3.2.2. 100GE MAC+PCS with Optional RS-FEC and PTP Hardware Design Example

Figure 21. 100GE MAC + PCS with Optional RS-FEC and PTP Hardware Design Examples High Level Block Diagram


TX Transceiver System Integrity Development Board
E-Tile Ethernet IP for Intel Agilex FPGA Hardware Design Example
Packet Client
ToD
E-Tile Hard IP for Ethernet Intel FPGA IP Core
MAC + PCS with optional RSFEC and PTP variant
Avalon-ST
PIO (PTP)
i_clk_ref
Serial lanes
To reset TX and RX datapaths, transceiver reconfiguration, and CSR
clk100
Ethernet Reconfiguration
Transceiver Reconfiguration
Avalon-MM Address Decoder
Sources and Probes
External PLL
RSFEC Reconfiguration (available with RSFEC enabled)
JTAG Master
50 MHz

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

- E-tile Ethernet IP for Intel Agilex FPGA core.
- Client logic that coordinates the programming of the IP core and packet generation.
- Time-of-day (ToD) module to provide a continuous flow of current time-of-day information to the IP core.

- PIO block to store RX and TX PTP timestamp for accuracy calculation and to send PTP 2-step timestamp request.
- Avalon memory-mapped interface address decoder to decode reconfiguration address space for MAC, transceiver, and RS-FEC modules during reconfiguration accesses.
- JTAG controller that communicates with the System Console. You communicate with the client logic through the System Console.

The following sample output illustrates a successful hardware test run for a 100GE, MAC+PCS with RS-FEC, non-PTP IP core variation. The test results are located at `<`*`design_example_dir`*`>/hardware_test_design/hwtest_ptp/c3_elane_xcvr_loopback_test.log` or `<`*`design_example_dir`*`>/hardware_test_design/hwtest_ptp/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: 13:25:08
    Test Start date is: 03/04/2019


Info: Cycling reset ...
    Successfully Write Channel 0 XCVR CSR Register offset = 0x84, data = 0x1
.
.
.
    Successfully Read  Channel 0 XCVR CSR Register offset = 0x88, data = 0x8

    C3 EHIP XCVR Channel 0 Loopback mode is successfully enabled

    Successfully Write Channel 1 XCVR CSR Register offset = 0x84, data = 0x1
.
.
.
    Successfully Read  Channel 1 XCVR CSR Register offset = 0x88, data = 0x8

    C3 EHIP XCVR Channel 1 Loopback mode is successfully enabled
.
.
.
    Successfully Read  Channel 2 XCVR CSR Register offset = 0x88, data = 0x8

    C3 EHIP XCVR Channel 2 Loopback mode is successfully enabled

    Successfully Write Channel 3 XCVR CSR Register offset = 0x84, data = 0x1
.
.
.
    Successfully Write Channel 3 XCVR CSR Register offset = 0x8a, data = 0x80
    Successfully Read  Channel 3 XCVR CSR Register offset = 0x88, data = 0x8

    C3 EHIP XCVR Channel 3 Loopback mode is successfully enabled

    Successfully Write EHIP User Register
phy_ehip_csr_soft_reset               , offset = 0x310, data = 0x0
.
.
.
    Successfully Read  EHIP User Register
phy_ehip_csr_soft_reset               , offset = 0x310, data = 0x0
```

```
    C3 EHIP System Reset is successfully

    Test End time is: 13:25:09
    Test End date is: 03/04/2019

Info: Closed JTAG Master Service


Info: Test <c3_ehip_xcvr_loopback_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: 13:25:09
    Test Start date is: 03/04/2019


Info: Read all EHIP CSR registers

    Successfully Read  EHIP User Register
phy_revid                              , offset = 0x300, data = 0x11112015
    Successfully Read  EHIP User Register
phy_scratch                           , offset = 0x301, data = 0x0
.
.
.
    Successfully Read  EHIP User Register
phy_ehip_csr_soft_reset               , offset = 0x310, data = 0x0

    C3 EHIP System Reset is successfully


Info: Stopping the traffic generator

    Successfully Write EHIP Traffic GEN/CHK Register, offset = 0x10, data =
0x87

Info: clearing the statistics

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

Info: Enabling the statistics

    Successfully Write EHIP User Register
cntr_tx_config                        , offset = 0x845, data = 0x0
    Successfully Write EHIP User Register
cntr_rx_config                        , offset = 0x945, data = 0x0

Info: Starting the traffic generator

    Successfully Write EHIP Traffic GEN/CHK Register, offset = 0x10, data =
0x85
    Successfully Read  EHIP User Register
cntr_tx_fragments_lo                  , offset = 0x800, data = 0x0

Info: Stopping the traffic generator

    Successfully Write EHIP Traffic GEN/CHK Register, offset = 0x10, data =
0x87
    Successfully Read  EHIP Traffic GEN/CHK Register, offset = 0x10, data =
0x87
.
```

```
.
.
    Successfully Read  EHIP User Register
cntr_rx_badlt_hi                      , offset = 0x969, data = 0x0

Info: Test iteration 1 is completed

    Successfully Read  RSFEC Register rsfec_top_rx_cfg                     ,
offset = 0x14, data = 0x1111
    Successfully Read  RSFEC Register arbiter_base_cfg                      ,
offset = 0x0, data = 0x1
    Successfully Read  RSFEC Register rsfec_top_clk_cfg                    ,
offset = 0x4, data = 0xf00
    Successfully Read  RSFEC Register rsfec_top_tx_cfg                     ,
offset = 0x10, data = 0x0
    Successfully Write RSFEC Register rsfec_top_tx_cfg                     ,
offset = 0x10, data = 0x10001666
    Successfully Read  RSFEC Register rsfec_top_tx_cfg                     ,
offset = 0x10, data = 0x10001666
    Successfully Write RSFEC Register rsfec_top_tx_cfg                     ,
offset = 0x10, data = 0x0
    Successfully Read  RSFEC Register rsfec_top_tx_cfg                     ,
offset = 0x10, data = 0x0

    Test End time is: 13:25:21
    Test End date is: 03/04/2019

Info: Closed JTAG Master Service


Info: Test <c3_ehip_traffic_basic_test> Passed
```

The following sample output illustrates a successful hardware test run for a 100GE, MAC+PCS with RS-FEC, PTP IP core variation. The test result is located at `<`*`design_example_dir`*`>/hardware_test_design/hwtest_ptp/c3_elane_ptp_traffic_basic_test.log`.

```
Info: Set JTAG Master Service Path


Info: Opened JTAG Master Service

    Test Start time is: 13:25:21
    Test Start date is: 03/04/2019

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

    C3 EHIP System Reset is successfully

    Successfully Write Channel 0 XCVR CSR Register offset = 0x84, data = 0x0
    Successfully Write Channel 0 XCVR CSR Register offset = 0x85, data = 0x0
.
.
.
    Successfully Write Channel 3 XCVR CSR Register offset = 0x93, data = 0x0
Internal Loopback iCal Status
    Successfully Write Channel 0 XCVR CSR Register offset = 0x84, data = 0x0
.
.
.
    Successfully Write Channel 0 XCVR CSR Register offset = 0x93, data = 0x0
```

```
iCal is done successfully on channel 0
    Successfully Write Channel 1 XCVR CSR Register offset = 0x84, data = 0x0
.
.
.
    Successfully Write Channel 3 XCVR CSR Register offset = 0x93, data = 0x0

Info: Cycling reset ...
    Successfully Write EHIP Traffic GEN/CHK Register, offset = 0x8, data = 0x40
    Successfully Write EHIP Traffic GEN/CHK Register, offset = 0x9, data = 0x1
    Successfully Read  EHIP Traffic GEN/CHK Register, offset = 0x9, data = 0x1

Info: clearing the statistics

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

Info: Enabling the statistics

    Successfully Write EHIP User Register
cntr_tx_config                        , offset = 0x845, data = 0x0
    Successfully Write EHIP User Register
cntr_rx_config                        , offset = 0x945, data = 0x0

Info: Accuracy measurement settings

Info: UI Value = 0x0009EE01

Info: TX Extra Latency = 0xc69814

Info: RX Extra Latency = 0x5467088

    Successfully Write EHIP User Register
tx_ptp_extra_latency                  , offset = 0xa0a, data = 0xc698
    Successfully Read  EHIP User Register
tx_ptp_extra_latency                  , offset = 0xa0a, data = 0xc698
    Successfully Write EHIP User Register
rx_ptp_extra_latency                  , offset = 0xb06, data = 0x80054670
    Successfully Read  EHIP User Register
rx_ptp_extra_latency                  , offset = 0xb06, data = 0x80054670

Info: Waiting for VL offset data ready

    Successfully Read  EHIP Soft PTP Register
vl_offset_data0_lo                    , offset = 0xc10, data = 0xc000008c

Info: All VL data reading, calculation of VL offset and reloading new VL
offset...

Reading FEC lane mapping and deskew ...
Lane map 0 = 0
Lane map 1 = 1
Lane map 2 = 2
Lane map 3 = 3
Lane 0 skew = 1
Lane 1 skew = 2
Lane 2 skew = 1
Lane 3 skew = 2

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
gen_vl_data_fec: Input Deskew_delay = 0x00000001
gen_vl_data_fec: Input Selected_pl  = 0
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

================================================================================
============

before-rotation: VL[PL] 0[0], deskew_delay = 0x1 UI, vl_offset_bits = 1
```

```
After Rotation: calc_vl_offset done - RVL 4, LPL 0, LVL 0 Sign=0, NS=0, FNS=2542


For LOCAL_VL=0 --> CALC_VL_OFFSET=0x000009EE, LOCAL_PL=0, REMOTE_VL=4
Final Calculated value - 325380

================================================================================
============
.
.
.
================================================================================
============

before-rotation: VL[PL] 19[0], deskew_delay = 0x1 UI, vl_offset_bits = 5
before-rotation: VL[PL] 19[0], deskew_delay = 0x1 UI, vl_offset_bits_shifted =
-325
After Rotation: calc_vl_offset done - RVL 3, LPL 0, LVL 19 Sign=1, NS=12,
FNS=39719


For LOCAL_VL=19 --> CALC_VL_OFFSET=0x800C9B27, LOCAL_PL=0, REMOTE_VL=3
Final Calculated value - 274983654275

================================================================================
============



Writing new VL offsets ...
write_vl_offset Loading vls data.....
    Successfully Write EHIP Soft PTP Register
vl_offset0_lo                         , offset = 0xc40, data = 0x4
.
.
.
    Successfully Write EHIP Soft PTP Register
vl_offset19_hi                        , offset = 0xc67, data = 0x800c9b27

Info: Waiting for PTP RX ready...

    Successfully Read  EHIP PIO Register, offset = 0x0, data = 0x7
.
.
.
    Successfully Read  EHIP PIO Register, offset = 0xc, data = 0x101

Info: Iteration = 1 : TX Timestamp = 0000000000060ca82f0f8fa7,  RX Timestamp =
0000000000060ca82f0e78eb,  Accuracy Difference = -1.08880615 ns

    Successfully Write EHIP PIO Register, offset = 0xc, data = 0x0
    Successfully Write EHIP Traffic GEN/CHK Register, offset = 0x10, data =
0x57
    Successfully Write EHIP PIO Register, offset = 0xc, data = 0x102
    Successfully Write EHIP Traffic GEN/CHK Register, offset = 0x10, data =
0x55
    Successfully Read  EHIP User Register
cntr_tx_64b_lo                        , offset = 0x816, data = 0x2
    Successfully Read  EHIP User Register
cntr_rx_64b_lo                        , offset = 0x916, data = 0x2
    Successfully Read  EHIP PIO Register, offset = 0x4, data = 0x90e52f43
    Successfully Read  EHIP PIO Register, offset = 0x5, data = 0x60f25
    Successfully Read  EHIP PIO Register, offset = 0x6, data = 0x0
    Successfully Read  EHIP PIO Register, offset = 0x8, data = 0x90e68e57
    Successfully Read  EHIP PIO Register, offset = 0x9, data = 0x60f25
    Successfully Read  EHIP PIO Register, offset = 0xa, data = 0x0
    Successfully Read  EHIP PIO Register, offset = 0x7, data = 0x2
    Successfully Read  EHIP PIO Register, offset = 0xc, data = 0x102

.
.
```

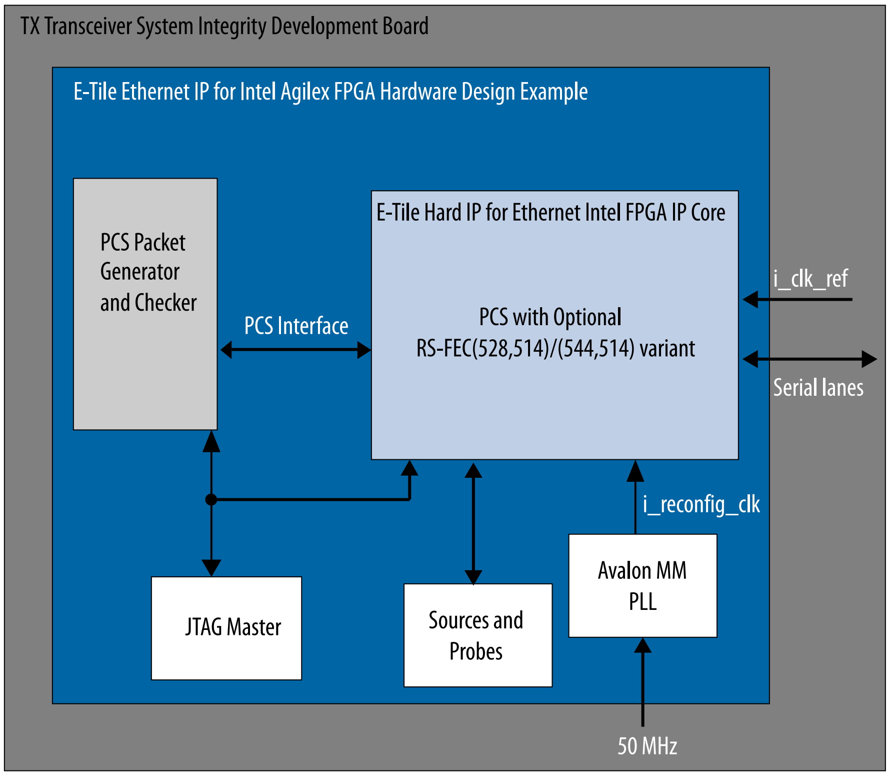
```
.
Info: Iteration = 100 : TX Timestamp = 00000000000a1d8d0ad81ed6,  RX Timestamp
= 00000000000a1d8d0ad982d9,  Accuracy Difference = 1.39067078 ns


Info: Stopping the traffic generator

    Successfully Write EHIP PIO Register, offset = 0xc, data = 0x0
.
.
.
    Successfully Read  EHIP User Register
cntr_rx_badlt_hi                      , offset = 0x969, data = 0x0

    Test End time is: 13:25:39
    Test End date is: 03/04/2019

Info: Closed JTAG Master Service



Info: Test <c3_ehip_ptp_traffic_basic_test> Passed
```

Related Information

- Compiling and Configuring the Design Example in Hardware on page 12
- Testing the E-tile Ethernet IP for Intel Agilex FPGA Hardware Design Example on page 13

2.3.2.3. 100GE PCS with Optional RS-FEC Hardware Design Example Components

Figure 22. 100GE MAC + PCS with Optional RS-FEC Hardware Design Example High Level Block Diagram


TX Transceiver System Integrity Development Board
E-Tile Ethernet IP for Intel Agilex FPGA Hardware Design Example
PCS Packet Generator and Checker
PCS Interface
E-Tile Hard IP for Ethernet Intel FPGA IP Core
PCS with Optional RS-FEC(528,514)/(544,514) variant
i_clk_ref
Serial lanes
i_reconfig_clk
Avalon MM PLL
JTAG Master
Sources and Probes
50 MHz

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

- E-tile Ethernet IP for Intel Agilex FPGA core.
- PCS packet generator and checker that coordinates the programming of the IP core, packet generation, and verify the packets.
- 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 test initiates media-independent interface (MII) packet transmission from packet generator to the IP core. The packet generator supports incremental packet mode, fixed-size packet mode, and random packet content mode. Once reset is completed, the packet generator generates the number of packets requested to the IP core. The IP core transfers the packets through internal PMA loopback to the packet generator and checker for verification. This test only works with internal PMA loopback mode.

The following sample output illustrates a successful hardware test run for 100GE, PCS only with (528,514) RS-FEC variation:

```
% pcs_only_traffic_test
Running pcs_only_traffic_test test
 RX PHY Register Access: Checking Clock Frequencies (KHz)

    REFCLK         :2 (KHZ)
    TXCLK         :40284  (KHZ)
    RXCLK         :40284  (KHZ)
    TXRSCLK     :0  (KHZ)
    RXRSCLK     :0  (KHZ)
 RX PHY Status Polling
 Rx Frequency Lock Status     0x0000000f
 Mac Clock in OK Condition?   0x00000001
 Rx Frame Error               0x00000000
Rx PHY Fully Aligned?        0x00000001
 Rx AM LOCK Condition?        0x00000001
 Rx Lanes Deskewed Condition? 0x00000001
Setting Number of frames to 6767
Setting Size of frames to 8588
Setting Size of frames to constant
-------------------------------------
PCS TRAFFIC = 0
pcs_only_traffic_test:pass
0
```

The following sample output illustrates a successful hardware test run for 100GE, PCS only with (544,512) RS-FEC variations:

```
% % pcs_only_traffic_test_pam4
Running pcs_only_traffic_test_pam4 test
RX PHY Register Access: Checking Clock Frequencies (KHz)
    REFCLK         :1 (KHZ)
    TXCLK         :41504  (KHZ)
    RXCLK         :41505  (KHZ)
    TXRSCLK     :0  (KHZ)
    RXRSCLK     :0  (KHZ)
 RX PHY Status Polling
 Rx Frequency Lock Status     0x0000000f
 Mac Clock in OK Condition?   0x00000001
 Rx Frame Error               0x00000000
 Rx AM LOCK Condition?        0x00000001
 Rx Lanes Deskewed Condition? 0x00000001
-------------------------------------
PCS TRAFFIC = 0
Setting Number of frames to 5340
Setting Size of frames to 635
Setting Size of frames to random
pcs_only_traffic_test_pam4:pass
```

Related Information

- Compiling and Configuring the Design Example in Hardware on page 12
- Testing the E-tile Ethernet IP for Intel Agilex FPGA Hardware Design Example on page 13

2.3.3. 100GE MAC+PCS with Optional RS-FEC Design Example Interface Signals

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

Table 11. 100GE MAC+PCS with Optional RS-FEC 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.

Related Information

E-tile Ethernet IP for Intel Agilex FPGA Interfaces and Signal Descriptions

2.3.4. 100GE PCS with Optional RS-FEC Design Example Interface Signals

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

Table 12. 100GE PCS with Optional RS-FEC 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.

Related Information

E-tile Ethernet IP for Intel Agilex FPGA Interfaces and Signal Descriptions

2.3.5. 100GE MAC+PCS with Optional RS-FEC Design Example Registers

Table 13. E-tile Ethernet IP for Intel Agilex 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 PCS 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
0x010000	RS-FEC configuration registers
0x100000	Transceiver registers

Table 14. Packet Client Registers

You can customize the E-tile Ethernet IP for Intel Agilex 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.	0x6	RW

continued...

Addr	Name	Bit	Description	HW Reset Value	Access
			• 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.		
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 upper 16bits	[15:0]	Source address (upper 16 bits)	0x8765	RW
0x1016	PKT_CL_LOOP BACK_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 Ethernet IP for Intel Agilex FPGA core register descriptions

2.3.6. 100GE PCS with Optional RS-FEC Design Example Registers

Table 15. E-tile Ethernet IP for Intel Agilex 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 PCS registers

continued...

Word Offset	Register Type
0x00F000	Packet Generator and Checker registers
0x010000	RS-FEC configuration registers
0x100000	Transceiver registers

Table 16. Packet Generator and Checker Registers

You can customize the E-tile Ethernet IP for Intel Agilex FPGA hardware design example by programming the packet client registers.

Addr	Name	Bit	Description	HW Reset Value	Access
0xF000	`Control Register 0`	[0]	Write 1 to start transmitting PCS packets.	0x0	RWC
0xF001	`Control Register 1`	[0]	Write 1 to reset the channel.	0x0	RW
0xF002	`XGMII Status register`	[6:0]	• Bit [0]: value 1 indicates the RX path is ready to receive packet • Bit [1]: Value 1 indicates the packets are verified and passed. • Bit [2]: Value 1 indicates there is an error with the received packets. • Bit [3]: Value 1 indicates the FIFO is full. • Bit [4]: Value 1 indicates the test is completed. • Bit [5]: Value 1 indicates all frames completed transmission and reception. • Bit [6]: value 1 indicates the test has passed. .	0x0	RO
0xF003	`GMII Status register`	[5:0]	• Bit [0]: value 1 indicates the GMII RX path is ready to receive packet • Bit [1]: Value 1 indicates the auto-negotiation completed. • Bit [2]: Value 1 indicates packet generation completed. • Bit [3]: Value 1 indicates packet verification completed. • Bit [4]: Value 1 indicates is an error with the received packets. • Bit [5]: value 1 indicates the test has passed.	0x0	RO
0xF006	`max_frame register`	[31:0]	Specify the maximum number of frames for transmission.	0x0	RW
0xF007	`frame_length register`	[31:0]	Specify the packet size.	0x0	RW
0xF008	`XGMII_data_match_count`	[255:0]	Report the number of XGMII passed packets.	0x0	RO
0xF009	`XGMII_data_mismatch_count`	[255:0]	Reports the number of XGMII error packets.	0x0	RO
0xF00A	`frame_type`	[2:0]	• 001: Fixed mode • 010: Incremental mode • 100: Random mode	0x0	RW
0xF00B	`PXGMII_client_loopback`	[0]	Set the value to 1 to enable XGMII RX loopback to XGMII TX.	0x0	RW

Related Information

E-tile Ethernet IP for Intel Agilex FPGA core register descriptions

2.4. Document Revision History for the E-tile Hard IP for Ethernet Intel Agilex FPGA IP Design Example User Guide

Document Version	Intel Quartus Prime Version	IP Version	Changes
2019.12.30	19.4	19.4.0	• Updated Table: *List of Supported Design Example Variants* to include hardware design example support. • Added hardware design example support for 10GE/25GE with optional RS-FEC design example. • Added hardware design example support for 100GE with optional RS-FEC design example. • Updated description of PMA adaptation setting in the *Generating the Design* section. • Added Asynchronous clock support for the 100GE MAC+PCS with (528,514) RS-FEC and PTP variant. • Restructured topics to improve the content flow.
2019.10.18	19.3	19.3.0	Initial Release.

3. E-tile CPRI PHY Intel FPGA IP Design Example

Related Information

About E-tile Hard IP Intel Agilex Design Example User Guide on page 4

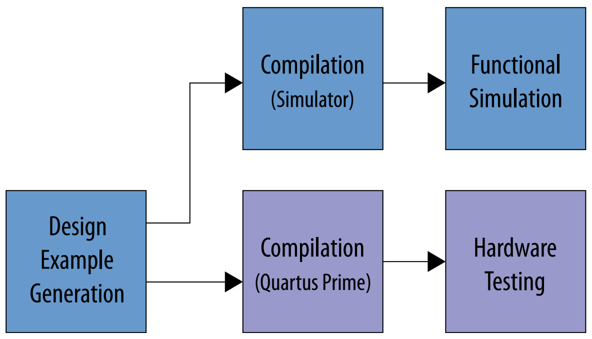
3.1. E-tile CPRI PHY Intel FPGA IP Quick Start Guide

The E-tile CPRI PHY Intel FPGA IP core for Intel Agilex devices 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 Agilex 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.

The E-tile CPRI PHY Intel FPGA IP core provides the capability of generating design examples for all supported combinations of number of CPRI channels and CPRI line bit rates. The testbench and design example support numerous parameter combinations of the E-tile CPRI PHY Intel FPGA IP core.

Figure 23. Development Steps for the Design Example



Related Information

- E-tile Hard IP User Guide
  For detailed information on E-tile CPRI PHY IP.
- About the E-Tile CPRI PHY Intel FPGA IP
  For more information about CPRI channels and supported CPRI line rates.

3.1.1. Hardware and Software Requirements

To test the example design, use the following hardware and software:

- Intel Quartus Prime Pro Edition software
- System Console
- Modelsim-SE*, VCS, NCSim and Xcelium Parallel Simulator*
- Intel Agilex TX Transceiver Signal Integrity Development Kit

3.1.2. Generating the Design

Figure 24. Procedure

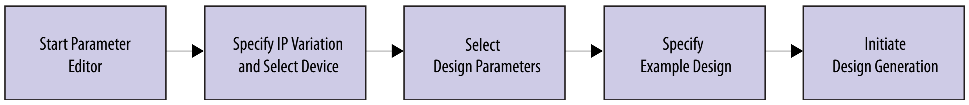
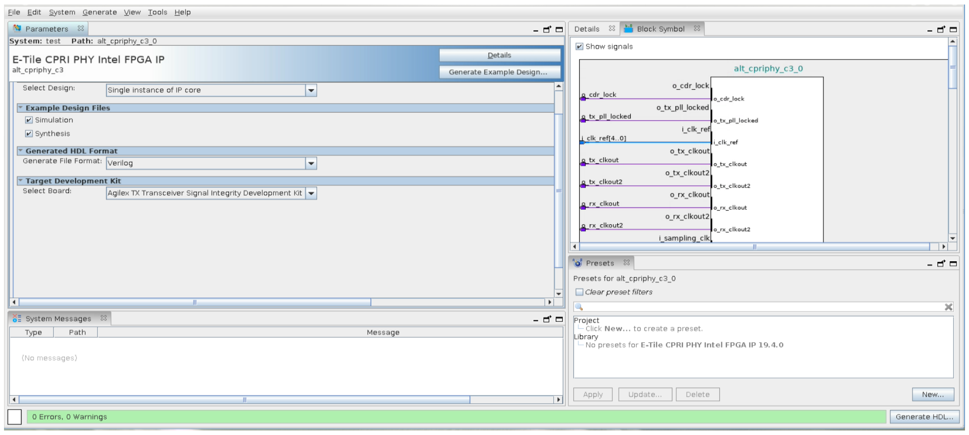


Figure 25. Example Design Tab in the E-tile CPRI PHY Intel FPGA IP Parameter Editor



System: test Path: alt_cpriphy_c3_0

E-Tile CPRI PHY Intel FPGA IP
alt_cpriphy_c3

Select Design: Single instance of IP core

Example Design Files
- [✓] Simulation
- [✓] Synthesis

Generated HDL Format
Generate File Format: Verilog

Target Development Kit
Select Board: Agilex TX Transceiver Signal Integrity Development Kit

Presets for alt_cpriphy_c3_0
Project
Click New... to create a preset.
Library
No presets for E-Tile CPRI PHY Intel FPGA IP 19.4.0

System Messages
(No messages)

0 Errors, 0 Warnings

If you do not already have an Intel Quartus Prime Pro Edition project in which to integrate your E-tile CPRI PHY 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 **Agilex (FB/FA)** 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 CPRI PHY IP hardware design example and testbench:

1. In the IP Catalog, locate and select **E-tile CPRI PHY Intel FPGA IP**. 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** tab, specify the parameters for your IP core variation.
5. The hardware design example provided with enable internal serial loopback by default.
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. You must select at least one of the **Simulation** and **Synthesis** options to generate the design example.
7. On the **Example Design** tab, under **Generated HDL Format**, 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.
8. Under **Target Development Kit**, select the **Agilex 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 function correctly, you must ensure that your project device is the same device on your development kit.
9. Click the **Generate Example Design** button. The **Select Example Design Directory** window appears.
10. If you want to modify the design example directory path or name from the defaults displayed (`alt_cpriphy_c3_0_example_design`), browse to the new path and type the new design example directory name (*<design_example_dir>*).

Related Information

About the E-Tile CPRI PHY Intel FPGA IP

3.1.3. Directory Structure

The E-tile CPRI PHY IP core design example file directories contain the following generated files for the design example.

Figure 26. Directory Structure of the Generated Example Design

<datarate> is either "2_4", "3", "4_9", "6", "9_8", "10", "12" or "24", depending on your IP core variation.

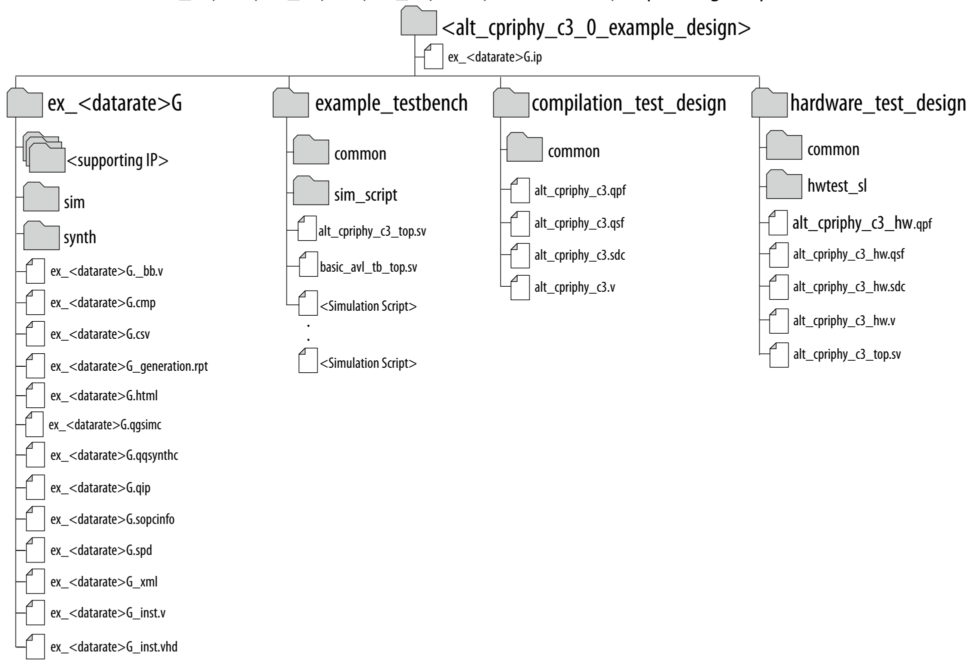


Table 17. E-tile CPRI PHY Intel FPGA IP Core Testbench File Descriptions

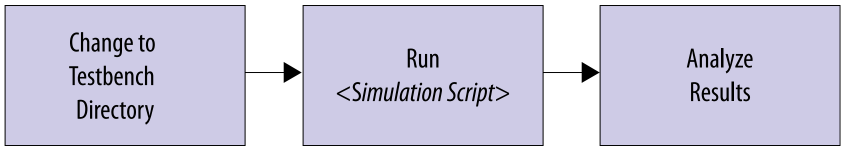
File Names	Description
Key Testbench and Simulation Files	
`<design_example_dir>/example_testbench/basic_avl_tb_top.sv`	Top-level testbench file. The testbench instantiates the DUT wrapper and runs Verilog HDL tasks to generate and accept packets.
`<design_example_dir>/example_testbench/alt_cpriphy_c3_top.sv`	DUT wrapper that instantiates DUT and other testbench components.
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 testbench.
`<design_example_dir>/example_testbench/run_vcsmx.sh`	The Synopsys VCS MX* script (combined Verilog HDL and SystemVerilog with VHDL) to run the testbench.
`<design_example_dir>/example_testbench/run_ncsim.sh`	The Cadence NCSim script to run the testbench.
`<design_example_dir>/example_testbench/run_xcelium.sh`	The Xcelium script to run the testbench.

Table 18. E-tile CPRI PHY Intel FPGA IP Core Hardware Design Example File Descriptions

File Names	Descriptions
`<design_example_dir>/hardware_test_design/alt_cpriphy_c3_hw.qpf`	Intel Quartus Prime project file.
`<design_example_dir>/hardware_test_design/alt_cpriphy_c3_hw.qsf`	Intel Quartus Prime project setting file.
`<design_example_dir>/hardware_test_design/alt_cpriphy_c3_hw.sdc`	Synopsys Design Constraints files. You can copy and modify these files for your own Intel Agilex design.
`<design_example_dir>/hardware_test_design/alt_cpriphy_c3_hw.v`	Top-level Verilog HDL design example file.
`<design_example_dir>/hardware_test_design/alt_cpriphy_c3_top.sv`	DUT wrapper that instantiates DUT and other testbench components.
`<design_example_dir>/hardware_test_design/hwtest_sl/main_script.tcl`	Main file for accessing System Console.

3.1.4. Simulating the Design Example Testbench

Figure 27. Procedure



Follow these steps to simulate the testbench:

1. At the command prompt, 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 received five hyperframes, and displays "PASSED".

Table 19. 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`
Xcelium*	In the command line, type `sh run_xcelium.sh`

Send Feedback

The following sample output illustrates a successful simulation test run for 24.33024 Gbps with 4 CPRI channels:

```
waiting for EHIP Ready....
EHIP READY is 1 at time             424915000
Enable internal serial loopback...
** Address offset = 0x84, WriteData = 0x00000001
** Address offset = 0x85, WriteData = 0x00000001
** Address offset = 0x86, WriteData = 0x00000008
** Address offset = 0x87, WriteData = 0x00000000
** Address offset = 0x90, WriteData = 0x00000001
** Reading address 0x8a[7] until it changes to 1...
** Address offset = 0x8a[7], ReadData  = 0x1
** Reading address 0x8b[0] until it changes to 0...
** Address offset = 0x8b[0], ReadData  = 0x0
** Address offset = 0x8a, WriteData = 0x00000080
** Address offset = 0x84, WriteData = 0x00000001
** Address offset = 0x85, WriteData = 0x00000001
** Address offset = 0x86, WriteData = 0x00000008
** Address offset = 0x87, WriteData = 0x00000000
** Address offset = 0x90, WriteData = 0x00000001
** Reading address 0x8a[7] until it changes to 1...
** Address offset = 0x8a[7], ReadData  = 0x1
** Reading address 0x8b[0] until it changes to 0...
** Address offset = 0x8b[0], ReadData  = 0x0
** Address offset = 0x8a, WriteData = 0x00000080
** Address offset = 0x84, WriteData = 0x00000001
** Address offset = 0x85, WriteData = 0x00000001
** Address offset = 0x86, WriteData = 0x00000008
** Address offset = 0x87, WriteData = 0x00000000
** Address offset = 0x90, WriteData = 0x00000001
** Reading address 0x8a[7] until it changes to 1...
** Address offset = 0x8a[7], ReadData  = 0x1
** Reading address 0x8b[0] until it changes to 0...
** Address offset = 0x8b[0], ReadData  = 0x0
** Address offset = 0x8a, WriteData = 0x00000080
** Address offset = 0x84, WriteData = 0x00000001
** Address offset = 0x85, WriteData = 0x00000001
** Address offset = 0x86, WriteData = 0x00000008
** Address offset = 0x87, WriteData = 0x00000000
** Address offset = 0x90, WriteData = 0x00000001
** Reading address 0x8a[7] until it changes to 1...
** Address offset = 0x8a[7], ReadData  = 0x1
** Reading address 0x8b[0] until it changes to 0...
** Address offset = 0x8b[0], ReadData  = 0x0
** Address offset = 0x8a, WriteData = 0x00000080
Internal serial loopback is enabled
Waiting for RX Block Lock
RX Block Lock is high at time             523408053
Waiting for RX ready
RX is ready is high at time             523450000
*** sending packets in progress, waiting for checker pass ***
*** waiting for measure_valid to assert...
** Address offset = 0xc01[0], ReadData  = 0x1
** measure_valid is asserted.
** Address offset = 0xc02, ReadData  = 0x0000280a
** Address offset = 0xc03, ReadData  = 0x000073c2
** Address offset = 0x29, ReadData  = 0x00000026
*** waiting for hyperframe sync to assert...
** hyperframe sync is asserted.
*** waiting for round trip measure...
-> 722269000ps: Channel 0: Round trip measure done with count 5058
** Channel 0: RX checker has received packets correctly!
** PASSED
*** waiting for measure_valid to assert...
** Address offset = 0xc01[0], ReadData  = 0x1
** measure_valid is asserted.
** Address offset = 0xc02, ReadData  = 0x00002709
** Address offset = 0xc03, ReadData  = 0x000072ad
** Address offset = 0x29, ReadData  = 0x00000066
```

```
*** waiting for hyperframe sync to assert...
** hyperframe sync is asserted.
*** waiting for round trip measure...
-> 729769000ps: Channel 1: Round trip measure done with count 4992
** Channel 1: RX checker has received packets correctly!
** PASSED
*** waiting for measure_valid to assert...
** Address offset = 0xc01[0], ReadData  = 0x1
** measure_valid is asserted.
** Address offset = 0xc02, ReadData  = 0x000025af
** Address offset = 0xc03, ReadData  = 0x000072ad
** Address offset = 0x29, ReadData  = 0x00000046
*** waiting for hyperframe sync to assert...
** hyperframe sync is asserted.
*** waiting for round trip measure...
-> 736725000ps: Channel 2: Round trip measure done with count 4949
** Channel 2: RX checker has received packets correctly!
** PASSED
*** waiting for measure_valid to assert...
** Address offset = 0xc01[0], ReadData  = 0x1
** measure_valid is asserted.
** Address offset = 0xc02, ReadData  = 0x00002836
** Address offset = 0xc03, ReadData  = 0x00007590
** Address offset = 0x29, ReadData  = 0x00000002
*** waiting for hyperframe sync to assert...
** hyperframe sync is asserted.
*** waiting for round trip measure...
-> 786573000ps: Channel 3: Round trip measure done with count 5123
** Channel 3: RX checker has received packets correctly!
** PASSED
**
*****************************************
$finish called from file "basic_avl_tb_top.sv", line 320.
$finish at simulation time 786593000ps
Simulation complete, time is 786593000000 fs.
```

Related Information

Reconfiguring the Duplex PMA Using the Reset Controller in Automatic Mode
Refer to this section to know more about the address offsets.

3.1.5. 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_cpriphy_c3.qpf`.
3. On the Processing menu, click **Start Compilation**.
4. 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

3.1.6. Compiling and Configuring the Design Example in Hardware

To compile the hardware design example and configure it on your Intel Agilex 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_cpriphy_c3_hw.qpf`.
3. On the Processing menu, click **Start Compilation**.
4. After successful compilation, a `.sof` file is available in `<design_example_dir>/hardware_test_design/output_files` directory. Follow these steps to program the hardware design example on the Intel Agilex device:
   a. Connect Intel Agilex TX Transceiver Signal Integrity Development Kit to the host computer.

      Note: The development kit is preprogrammed with the correct clock frequencies by default. You do not need to use the Clock Control application to set the frequencies.
   b. On the **Tools** menu, click **Programmer**.
   c. In the Programmer, click **Hardware Setup**.
   d. Select a programming device.
   e. Select and add the **Agilex TX Transceiver Signal Integrity Development kit** to which your Intel Quartus Prime Pro Edition session can connect.
   f. Ensure that **Mode** is set to **JTAG**.
   g. Select the Intel Agilex device and click **Add Device**. The Programmer displays a block diagram of the connections between the devices on your board.
   h. In the row with your `.sof`, check the box for the `.sof`.
   i. Check the box in the **Program/Configure** column.
   j. Click **Start**.

Related Information

- Block-Based Design Flows
- Programming Intel FPGA Devices
- Analyzing and Debugging Designs with System Console

3.1.7. Testing the E-tile CPRI PHY Intel FPGA IP Hardware Design Example

After you compile the E-tile CPRI PHY Intel FPGA IP core design example and configure it on your Intel Agilex 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 Agilex 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_sl`.
3. Type `source main_script.tcl` to open a connection to the JTAG master and start the test.

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

   The following sample output illustrates a successful test run for 24.33024 Gbps CPRI line bit rate with 1 CPRI channel:

```
Info: Number of Channels = 1
Info: JTAG Port ID       = 1
Info: Speed              = 24G

Info: Start of c3_cpri_test

INFO: Basic CPRI test

INFO: Checking PLL lock status...
    iopll_sclk_locked 1, channel_pll_locked 1
INFO: PLL is locked

Loop 0
INFO: Set Reconfig Reset
INFO: Channel 0: Set CSR Reset
INFO: Channel 0: Set TX Reset
INFO: Channel 0: Set RX Reset
INFO: Release Reconfig Reset
INFO: Channel 0: Release CSR Reset
INFO: Channel 0: Release TX Reset
INFO: Channel 0: Release RX Reset
INFO: Wait for master channel to stable
INFO: Release Reset Done!
INFO: Turn on serial loopback

    INFO: Start of C3 ELANE XCVR Channel 0 Loopback mode

    INFO: Polling For PMA Register: Read  XCVR CSR Register offset = 0x8a,
data = 0x80
    INFO: Polling For PMA Register: Read  XCVR CSR Register offset = 0x8b,
data = 0x8e

    INFO: C3 ELANE XCVR Channel 0 Loopback mode is successfully enabled

INFO: Running calibration...
INFO: Channel 0

INFO: Assert TX RX Digital Reset

INFO: Channel 0: Set TX Reset
INFO: Channel 0: Set RX Reset
INFO: Reset PMA

    INFO: Waiting PMA reset . . .
    INFO: Waiting 3
    INFO: Waiting 4
    INFO: Waiting 5
    INFO: Waiting 6
    INFO: Waiting 8
    INFO: Waiting 9
    INFO: Waiting 11
    INFO: Waiting 12
    INFO: Waiting 13

INFO: De-assert TX Digital Reset

INFO: Channel 0: Release TX Reset
    INFO: Internal loopback


    INFO: Start of C3 ELANE XCVR Channel 0 Loopback mode

    INFO: Polling For PMA Register: Read  XCVR CSR Register offset = 0x8a,
data = 0x80
    INFO: Polling For PMA Register: Read  XCVR CSR Register offset = 0x8b,
data = 0x8e

    INFO: C3 ELANE XCVR Channel 0 Loopback mode is successfully enabled

    INFO: Channel 0 initial adaptation

    INFO: Polling For PMA Register: Read  XCVR CSR Register offset = 0x8a,
data = 0x80
    INFO: Polling For PMA Register: Read  XCVR CSR Register offset = 0x8b,
data = 0x8c
    INFO: Channel 0 initial adaptation status
    INFO: Polling For PMA Register: Read  XCVR CSR Register offset = 0x8a,
data = 0x80
    INFO: Polling For PMA Register: Read  XCVR CSR Register offset = 0x8b,
data = 0x8e
    INFO: Polling For PMA Register: Read  XCVR CSR Register offset = 0x88,
data = 0x80
    INFO: Initial adaptation is done successfully on channel 0

INFO: De-assert RX Digital Reset

INFO: Channel 0: Release RX Reset
Channel 0 : Wait for measure_valid to assert
    measure_valid is asserted
Channel 0 : Get checker_pass status:
    Checker value = 1
    Checker status = Passed!
Channel 0 : Read Determenistic latency counts
Info: Loop 0 passed
End of loop 0



Info: End of c3_cpri_test


Info: Total loop passed = 1/1


Info: Test <c3_cpri_test> Passed
```

3.2. E-tile CPRI PHY Design Example Description

The design example demonstrates the basic functionality of the E-tile CPRI PHY IP core. You can generate the design from the Example Design tab in the E-tile CPRI PHY IP 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. You can choose to generate the design example with or without the RS-FEC feature. The RS-FEC feature is available with 10.1376, 12.1651 and 24.33024 Gbps CPRI line bit rates.

3.2.1. Features

- TX and RX serial loopback mode
- Generate the design example with RS-FEC feature
- PMA adaptation
- Supports TX and RX external loopback mode when you turn on PMA adaptation feature
- Basic packet checking capabilities including round trip latency count
- Ability to use System Console to reset the design for re-testing purpose

3.2.2. Simulation Design Example

The E-tile CPRI PHY design example generates a simulation testbench and simulation files that instantiates the E-tile CPRI PHY Intel FPGA IP core when you select the **Simulation** option.

Figure 28. E-tile CPRI PHY Intel FPGA IP for 10.1316, 12.1651, and 24.33024 Gbps (with and without RS-FEC) Line Rates

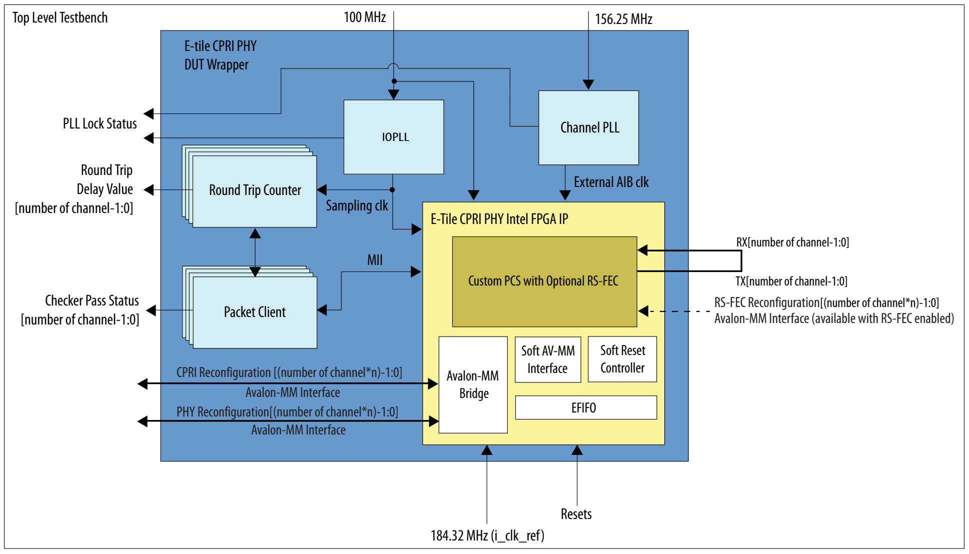
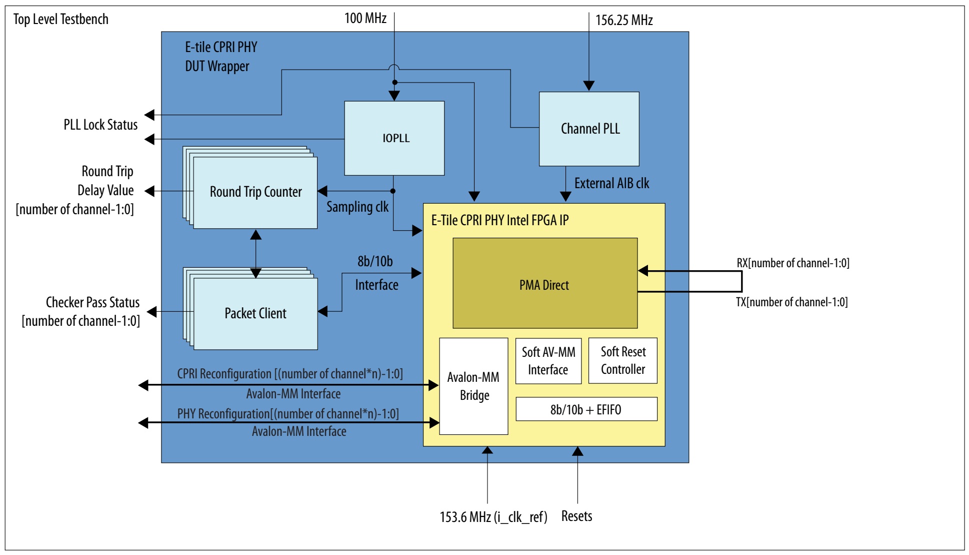

Top Level Testbench
100 MHz
156.25 MHz
E-tile CPRI PHY DUT Wrapper
PLL Lock Status
IOPLL
Channel PLL
Round Trip Delay Value [number of channel-1:0]
Round Trip Counter
Sampling clk
External AIB clk
E-Tile CPRI PHY Intel FPGA IP
RX[number of channel-1:0]
TX[number of channel-1:0]
MII
Custom PCS with Optional RS-FEC
RS-FEC Reconfiguration[(number of channel*n)-1:0] Avalon-MM Interface (available with RS-FEC enabled)
Checker Pass Status [number of channel-1:0]
Packet Client
Soft AV-MM Interface
Soft Reset Controller
Avalon-MM Bridge
EFIFO
CPRI Reconfiguration [(number of channel*n)-1:0] Avalon-MM Interface
PHY Reconfiguration[(number of channel*n)-1:0] Avalon-MM Interface
Resets
184.32 MHz (i_clk_ref)

Figure 29. E-tile CPRI PHY Intel FPGA IP for 2.4376, 3.0720, 4.9152, 6.144, 9.8304 Gbps Line Rates


Top Level Testbench
100 MHz
156.25 MHz
E-tile CPRI PHY DUT Wrapper
PLL Lock Status
IOPLL
Channel PLL
Round Trip Delay Value [number of channel-1:0]
Round Trip Counter
Sampling clk
External AIB clk
E-Tile CPRI PHY Intel FPGA IP
8b/10b Interface
PMA Direct
RX[number of channel-1:0]
TX[number of channel-1:0]
Checker Pass Status [number of channel-1:0]
Packet Client
Avalon-MM Bridge
Soft AV-MM Interface
Soft Reset Controller
CPRI Reconfiguration [(number of channel*n)-1:0] Avalon-MM Interface
PHY Reconfiguration[(number of channel*n)-1:0] Avalon-MM Interface
8b/10b + EFIFO
153.6 MHz (i_clk_ref)
Resets

In this design example, the simulation testbench provides basic functionality such as startup and wait for lock, transmit and receive packets.

The successful test run displays output confirming the following behavior:

1. The client logic resets the IP core.
2. The client logic waits for the RX datapath alignment.
3. The client logic transmits hyperframes on the TX MII interface and waits for five hyperframes to be received on RX MII interface. Hyperframes are transmitted and received on MII interface according to the CPRI v7.0 specifications.

   Note: The CPRI designs that target 2.4/3/4.9/6.1/9.8 Gbps line rates use 8b/10b interface and the designs that target 10.1, 12.1 and 24.3 Gbps (with and without RS-FEC) use MII interface.

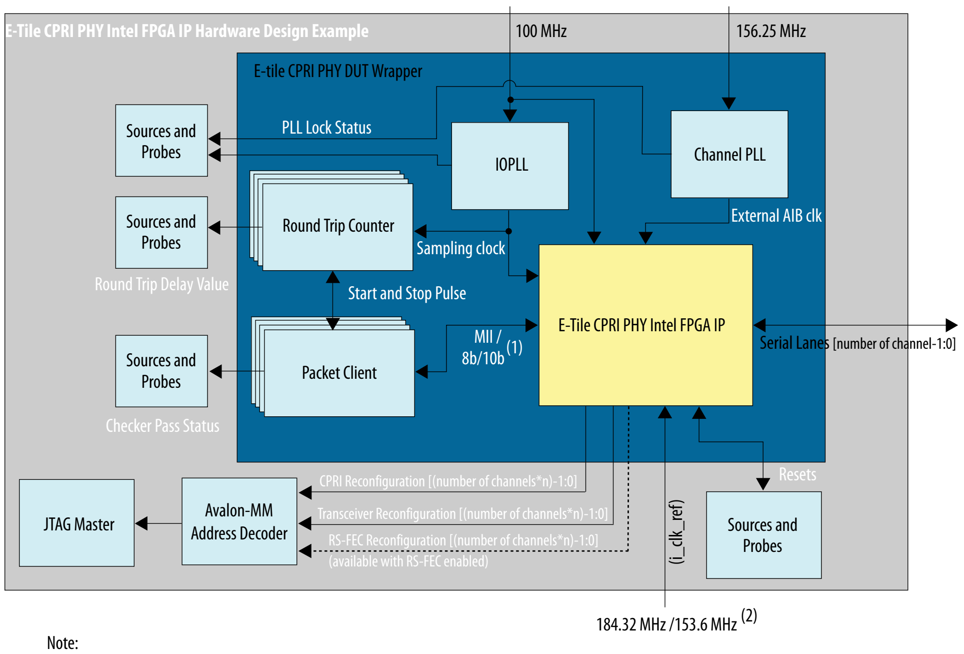
   Note: This design example includes a round trip counter to count the round trip latency from TX to RX.
4. The client logic reads the round trip latency value and checks for the content and correctness of the hyperframes data on the RX MII side once the counter completes the round trip latency count.

Related Information

CPRI Specifications

3.2.3. Hardware Design Example

Figure 30. E-tile CPRI PHY Intel FPGA IP Core Hardware Design Examples High Level Block Diagram


E-Tile CPRI PHY Intel FPGA IP Hardware Design Example
100 MHz
156.25 MHz
E-tile CPRI PHY DUT Wrapper
Sources and Probes
PLL Lock Status
IOPLL
Channel PLL
Sources and Probes
Round Trip Counter
External AIB clk
Sampling clock
Round Trip Delay Value
Start and Stop Pulse
E-Tile CPRI PHY Intel FPGA IP
MII / 8b/10b (1)
Sources and Probes
Packet Client
Serial Lanes [number of channel-1:0]
Checker Pass Status
Resets
CPRI Reconfiguration [(number of channels*n)-1:0]
JTAG Master
Avalon-MM Address Decoder
Transceiver Reconfiguration [(number of channels*n)-1:0]
RS-FEC Reconfiguration [(number of channels*n)-1:0]
(available with RS-FEC enabled)
(i_clk_ref)
Sources and Probes
184.32 MHz /153.6 MHz (2)

Note:
(1) The CPRI designs with 2.4/3.07/4.9/6.1/9.8 Gbps CPRI line rates use 8b/10b interface and all other CPRI line rates designs use MII interface.
(2) The CPRI designs with 2.4/3.07/4.9/6.1/9.8 Gbps CPRI line rates need 153.6 MHz transceiver reference clock and all other CPRI line rates need 184.32 MHz.

The E-tile CPRI PHY Intel FPGA IP core hardware design example includes the following components:

- E-tile CPRI PHY Intel FPGA IP core.
- Packet client logic block that generates and receives traffic.
- Round trip counter.
- IOPLL to generate sampling clock for deterministic latency logic inside the IP, and round trip counter component at testbench.
- Channel PLL to generate external AIB clocks for the IP.
- Avalon-MM address decoder to decode reconfiguration address space for CPRI, transceiver, and RS-FEC modules during reconfiguration accesses.
- Sources and probes for asserting resets and monitoring the clocks and a few status bits.
- JTAG controller that communicates with the System Console. You communicate with the client logic through System Console.
- The example design targets an Intel Agilex TX Transceiver Signal Integrity Development Kit.

3.2.4. Interface Signals

Table 20. Design Example Interface Signals

Signal	Direction	Description
`ref_clk100MHz`	Input	Input clock for CSR access on all the AV-MM interfaces. Drive at 100 MHz.
`ref_clk156MHz`	Input	Reference clock for channel PLL. Drive at 156.25 MHz.
`i_clk_ref`	Input	Transceiver reference clock. Drive at • 153.6 MHz for CPRI line rates 2.4,3.07,4.9,6.1, and 9.8 Gbps. • 184.32 MHz for CPRI line rates 10.1,12.1, and 24.3 Gbps with and without RS-FEC.
`i_rx_serial[n]`	Input	Transceiver PHY input serial data.
`o_tx_serial[n]`	Output	Transceiver PHY output serial data.

3.2.5. Design Example Register Map for Reconfiguration

Table 21. E-tile CPRI PHY Intel FPGA IP Hardware Design Example PHY Register Map

Channel Number	Word Offset	Register Type
0	0x000000	CPRI registers
	0x010000	RS-FEC configuration registers
	0x100000	Transceiver registers
1	0x200000	CPRI registers
	0x300000	Transceiver registers
2	0x400000	CPRI registers
	0x500000	Transceiver registers
3	0x600000	CPRI registers
	0x700000	Transceiver registers

3.3. Document Revision History for the E-tile CPRI PHY Intel FPGA IP Design Example User Guide

Document Version	Intel Quartus Prime Version	IP Version	Changes
2019.12.30	19.4	19.4.0	Initial Release.

4. E-Tile Dynamic Reconfiguration Design Example

Related Information

About E-tile Hard IP Intel Agilex Design Example User Guide on page 4

4.1. Quick Start Guide

The E-Tile Dynamic Reconfiguration Design Example 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 the design in hardware.

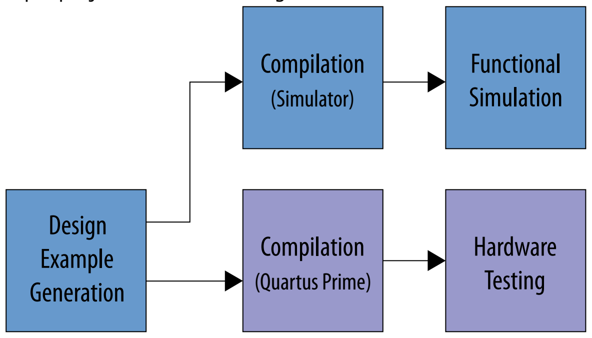
In addition, you can download the compiled hardware design to the Agilex TX Transceiver Signal Integrity Development Kit.

Table 22. List of Supported E-Tile Dynamic Reconfiguration Design Example Variants

Dynamic Reconfiguration Protocol	Variant	Simulation	Hardware Design Example
10G/25G Ethernet Protocol	10G/25G with PTP and optional RS-FEC	√	√
	10G/25G with optional RS-FEC	√	√
CPRI	10G/24G CPRI with optional RS-FEC	√	√
	9.8G CPRI	√	√
25G Ethernet to CPRI Protocol	25G with PTP and optional RS-FEC	√	√
100G Ethernet Protocol	100G Ethernet MAC+PCS with optional RS-FEC	√	√

Figure 31. Development Steps for the Design Example

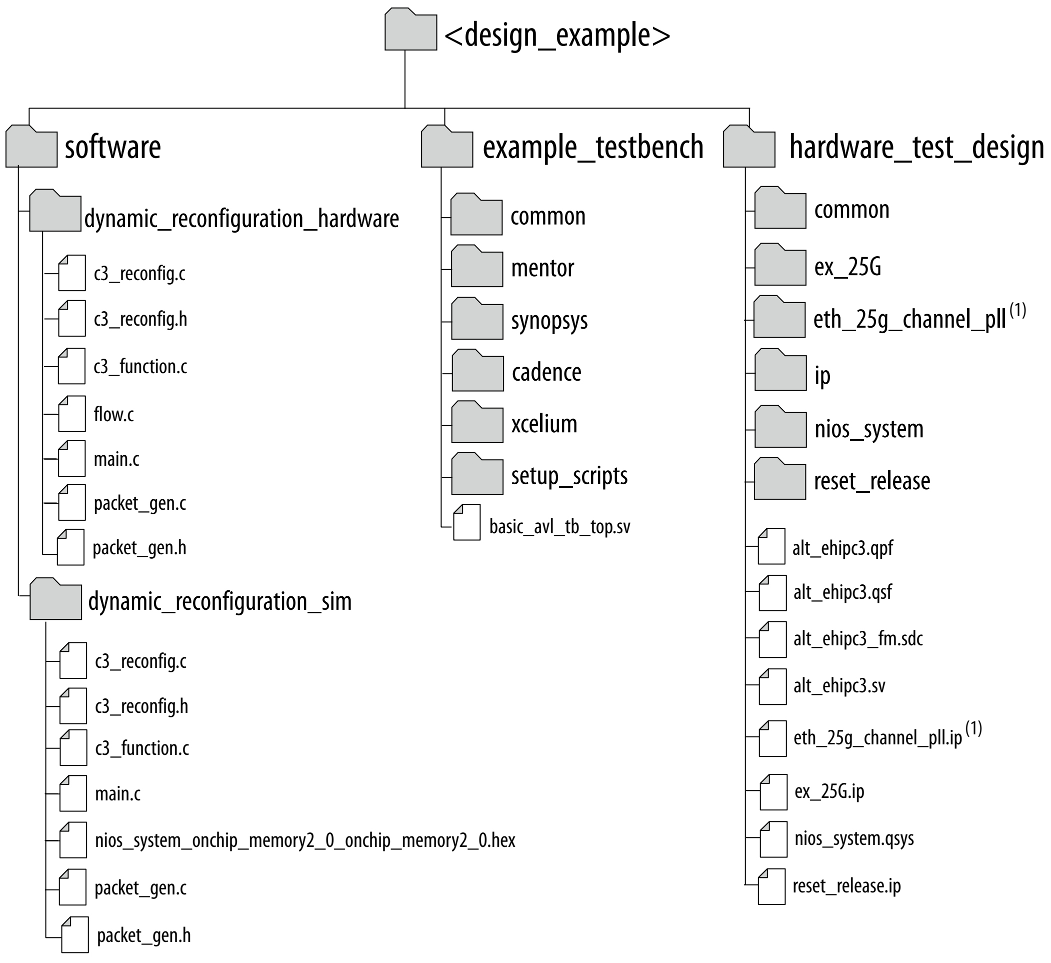
The compilation-only example project cannot be configured in hardware.



4.1.1. Directory Structure

The E-Tile Dynamic Reconfiguration Design Example file directories contain the following generated files for the design examples.

Figure 32. E-tile Dynamic Reconfiguration 10G/25G Ethernet and 25G Ethernet to CPRI Design Example Directory Structure



Note:
1. Only applicable for 25G+RS-FEC design.

Figure 33. E-tile Dynamic Reconfiguration 24G CPRI Design Example Directory Structure

The example directory structure applies to all CPRI variants. <datarate> is either "24G" or "9P8G", depending on your IP core variation.

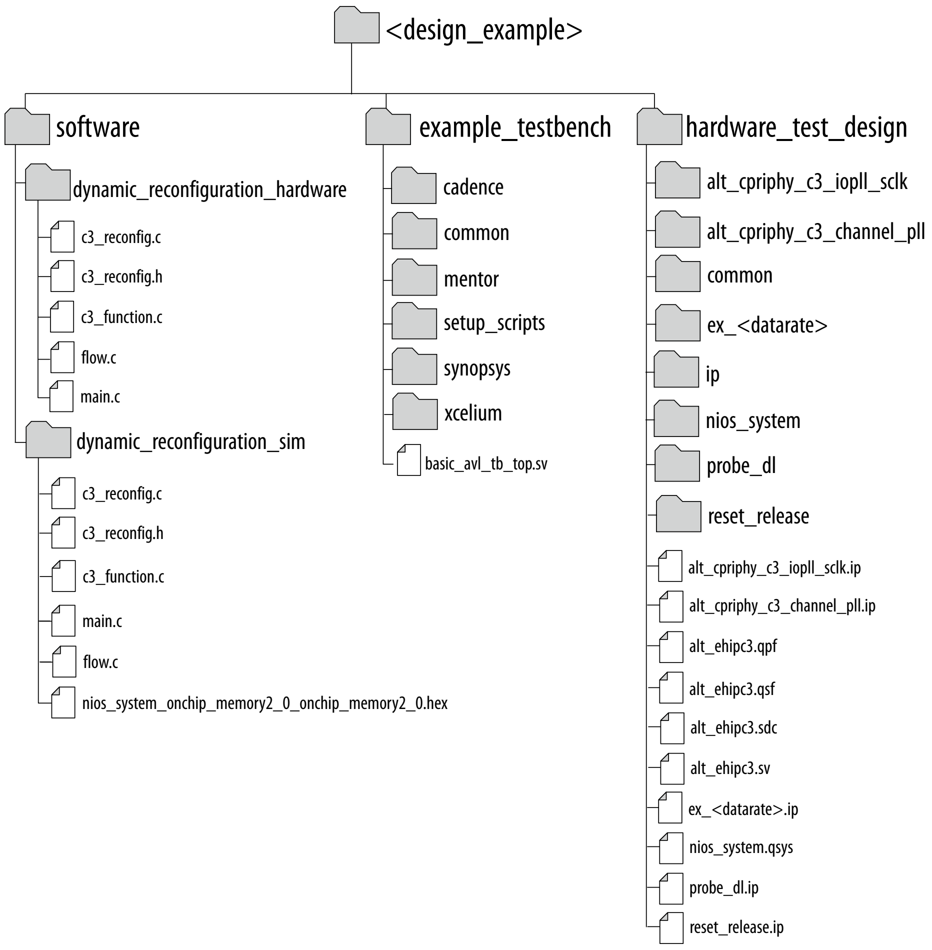


Figure 34. E-tile Dynamic Reconfiguration 100G Ethernet Design Example Directory Structure

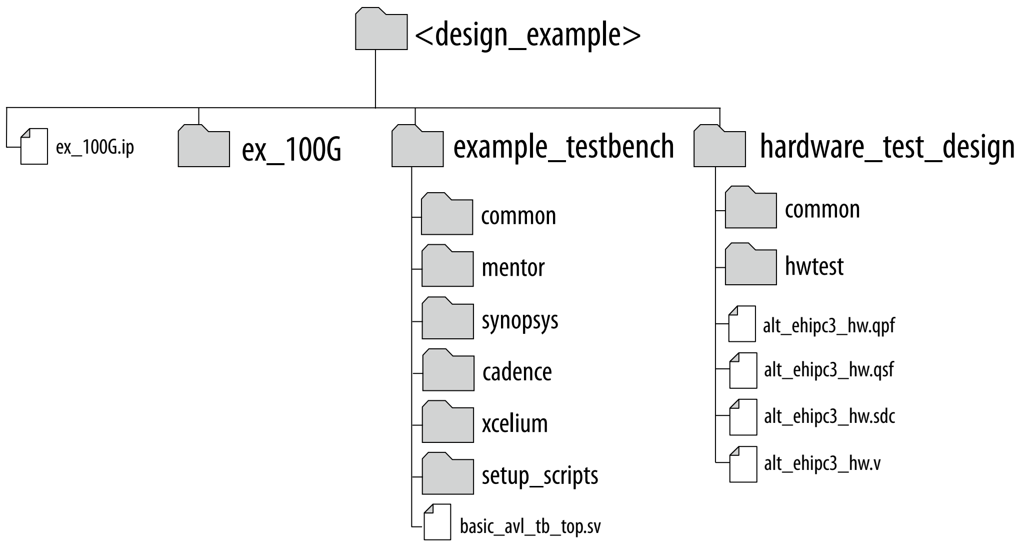


Table 23. E-Tile Dynamic Reconfiguration Design Example Testbench File Descriptions

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/mentor/run_vsim.do`	The Mentor Graphics ModelSim script to run the testbench.
`<design_example_dir>/example_testbench/synopsys/run_vcs.sh`	The Synopsys VCS script to run the testbench.
`<design_example_dir>/example_testbench/synopsys/run_vcsmx.sh`	The Synopsys VCS MX script (combined Verilog HDL and SystemVerilog with VHDL) to run the testbench.
`<design_example_dir>/example_testbench/run_ncsim.sh`	The Cadence NCSim script to run the testbench.
`<design_example_dir>/example_testbench/run_xcelium.sh`	The Cadence Xcelium script to run the testbench.

Table 24. E-Tile Dynamic Reconfiguration Design Example Hardware Design Example File Descriptions for 10G/25G Ethernet and CPRI Protocols

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

continued...

File Names	Description
`<design_example_dir>/ hardware_test_design/alt_ehipc3.sdc`	Synopsys Design Constraints files. You can copy and modify these files for your own Intel Agilex design.
`<design_example_dir>/ hardware_test_design/alt_ehipc3.sv`	Top-level Verilog HDL design example file
`<design_example_dir>/ hardware_test_design/common/`	Hardware design example support files

Table 25. E-Tile Dynamic Reconfiguration Design Example Hardware Design Example File Descriptions for 100G Ethernet Protocol

File Names	Description
`<design_example_dir>/ hardware_test_design/alt_ehipc3_hw.qpf`	Intel Quartus Prime project file
`<design_example_dir>/ hardware_test_design/alt_ehipc3_hw.qsf`	Intel Quartus Prime project settings file
`<design_example_dir>/ hardware_test_design/alt_ehipc3_hw.sdc`	Synopsys Design Constraints files. You can copy and modify these files for your own Intel Agilex design.
`<design_example_dir>/ hardware_test_design/alt_ehipc3_hw.v`	Top-level Verilog HDL design example file
`<design_example_dir>/ hardware_test_design/common/`	Hardware design example support files

4.1.2. Generating the Design

Figure 35. Procedure

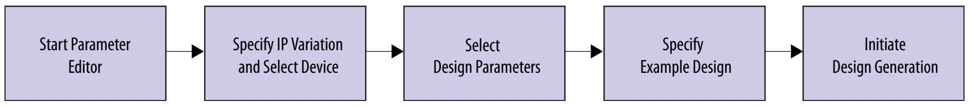
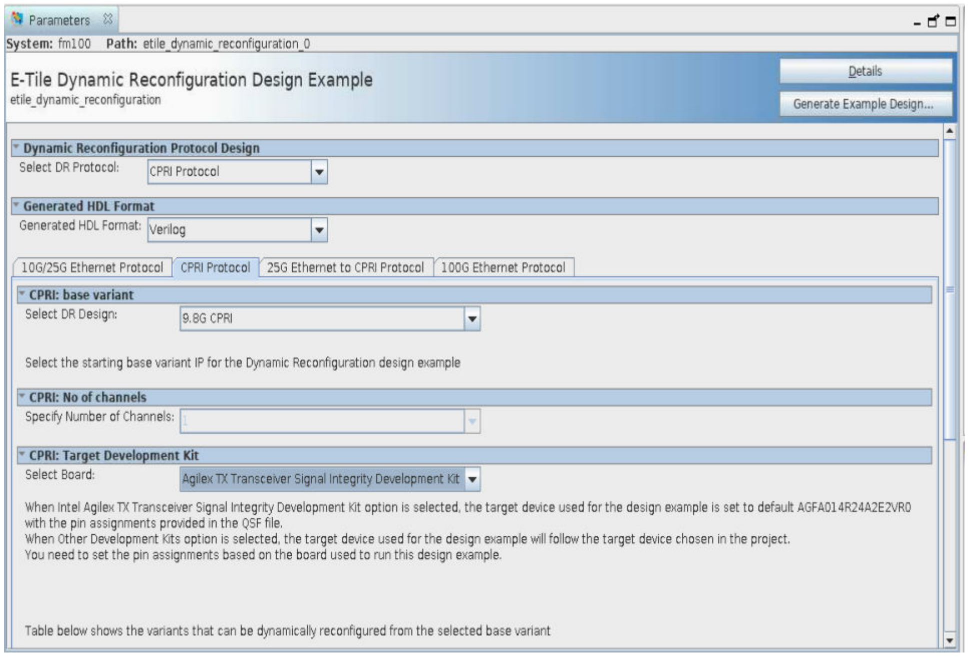


Figure 36. Example Design Tab in the E-Tile Dynamic Reconfiguration Design Example Parameter Editor


Parameters
System: fm100 Path: etile_dynamic_reconfiguration_0
E-Tile Dynamic Reconfiguration Design Example
etile_dynamic_reconfiguration
Details
Generate Example Design...
10G/25G Ethernet Protocol
CPRI Protocol
25G Ethernet to CPRI Protocol
100G Ethernet Protocol

Dynamic Reconfiguration Protocol Design
Select DR Protocol: CPRI Protocol

Generated HDL Format
Generated HDL Format: Verilog

CPRI: base variant
Select DR Design: 9.8G CPRI

Select the starting base variant IP for the Dynamic Reconfiguration design example

CPRI: No of channels
Specify Number of Channels:

CPRI: Target Development Kit
Select Board: Agilex TX Transceiver Signal Integrity Development Kit

When Intel Agilex TX Transceiver Signal Integrity Development Kit option is selected, the target device used for the design example is set to default AGFA014R24A2E2VR0 with the pin assignments provided in the QSF file.
When Other Development Kits option is selected, the target device used for the design example will follow the target device chosen in the project.
You need to set the pin assignments based on the board used to run this design example.

Table below shows the variants that can be dynamically reconfigured from the selected base variant

If you do not already have an Intel Quartus Prime Pro Edition project in which to integrate your 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 Agilex** and select a device that meets all of these requirements:
   - 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 Dynamic Reconfiguration design example hardware design example and testbench:

1. In the IP Catalog, locate and select **E-Tile Dynamic Reconfiguration Design Example**. 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. Under **Select DR Protocol**, select one of the protocols:

- If you select **10G/25G Ethernet Protocol**, click the **10G/25G Ethernet Protocol** tab.
- If you select **CPRI Protocol**, click the **CPRI Protocol** tab.
- If you select **25G Ethernet to CPRI Protocol**, click the **25G Ethernet to CPRI Protocol** tab.
- If you select **100G Ethernet**, click the **100G Ethernet Protocol** tab.

5. Under **Select DR Design**, select a starting base variant IP for the selected DR Protocol design.
6. Under **Target Development Kit**, select the **Agilex TX Transceiver Signal Integrity Development Kit**, available for Intel Agilex devices, or select **Other Development Kits**. The compilation-only and hardware design examples target your project device. For the hardware design to function correctly, you must ensure that your project device is the same device on your development kit.
7. Click the **Generate Example Design** button. The **Select Example Design Directory** window appears.
8. If you want to modify the design example directory path or name from the defaults displayed (`etile_dynamic_reconfiguration_0_EXAMPLE_DESIGN`), browse to the new path and type the new design example directory name (*<design_example_dir>*).
9. Click **OK**.

4.1.2.1. Design Example Parameters

The E-Tile Dynamic Reconfiguration Design Example parameter editor allows you to specify certain parameters before generating the design example.

Table 26. Parameters in the E-Tile Dynamic Reconfiguration Design Example Parameter Editor

Parameter	Options	Description
Select DR Protocol	• 10G/25G Ethernet Protocol • CPRI Protocol • 25G Ethernet to CPRI Protocol • 100G Ethernet	Available protocols for dynamic reconfiguration design example generation.
Parameter Settings: 10G/25G Ethernet Protocol (This tab is only applicable when you select 10G/25G Ethernet Protocol)		
Select DR Design	• 25G 1588 PTP RS-FEC • 25G RS-FEC	Available base variants for Ethernet Dynamic Reconfiguration design example generation.
Parameter Settings: CPRI Protocol (This tab is only applicable when you select CPRI Protocol)		
Select DR Design	• 24G CPRI RS-FEC • 9.8G CPRI	Available base variant for CPRI Dynamic Reconfiguration design example generation.
Parameter Settings: 25G Ethernet to CPRI Protocol (This tab is only applicable when you select 25G Ethernet to CPRI Protocol)		
Select DR Design	25GE PTP RS-FEC	Available base variant for Ethernet to CPRI Dynamic Reconfiguration design example generation.
Parameter Settings: 100G Ethernet Protocol (This tab is only applicable when you select 100G Ethernet Protocol)		

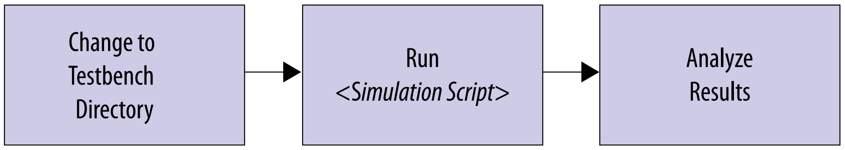
continued...

Parameter	Options	Description
Select DR Design	100G Ethernet MAC+PCS RS-FEC	Available base variants for 100G Ethernet Dynamic Reconfiguration design example generation.
Select DR Controller Location	• Internal • External	Internal Dynamic Reconfiguration selection. • 0: Enables the soft CPU Dynamic Reconfiguration controller internally within the IP • 1: Enables external hardware reconfiguration.
Parameter Settings: 10G/25G Ethernet Protocol, CPRI, 25G Ethernet to CPRI Protocol, and 100G Ethernet Protocol (The parameters below are available in both tabs)		
Specify Number of Channels	1	Specify the number of channels. The valid number of channels is 1 and this parameter is not selectable. *Note:* This parameter is not available in the 100G Ethernet protocol.
Select Board	• Agilex TX Transceiver Signal Integrity Development Kit • Other Development Kits	Supported hardware for design implementation. When you select an Intel FPGA development board, the *Target Device* is the one that matches the device on the Development Kit. If this menu is not available, there is no supported board for the options that you select. **Other Development Kits**: This option allows the design example to be tested on development kits other than 1ST280EY2F55E2VG. You need to set the pin assignments based on the board used to run this design example.

4.1.3. Simulating the E-Tile Dynamic Reconfiguration Design Example Testbench

You can compile and simulate the design by running a simulation script from the command prompt.

Figure 37. Procedure



4.1.3.1. Running the Simulation with Default HEX File

You can run and simulate the default Nios® II-based testbench of the design example using a pre-generated HEX file (`nios_system_onchip_memory2_0_onchip_memory2_0.hex`) that provided in the `<design_example_dir>/software/dynamic_reconfiguration_sim` directory.

Note: The HEX file is generated based on the C-code design example simulation source files in the `dynamic_reconfiguration_sim` folder. If you modify the source files, you need to generate a new HEX file using Nios II Software Build Tools (SBT) for Eclipse. Refer to the *Running the Simulation with New HEX File* section for the steps on generating a new HEX file and simulating the testbench using the new HEX file.

Follow these steps to simulate the testbench:

1. Open the <*simulator_name*>`_files.tcl` script in the <*design_example_dir*>`/example_testbench/setup_scripts/common` directory.
2. Edit the TCL script to change the existing `nios_system_onchip_memory2_0_onchip_memory2_0.hex` file directory to the pre-generated HEX file directory.

   For example, change the following line in the TCL script from:

   ```
   lappend memory_files "[normalize_path "$QSYS_SIMDIR/../<design_example_dir>/
   hardware_test_design/ip/nios_system/nios_system_onchip_memory2_0/
   altera_avalon_onchip_memory2_191/sim/
   nios_system_onchip_memory2_0_onchip_memory2_0.hex"]"
   ```

   to

   ```
   lappend memory_files "[normalize_path "$QSYS_SIMDIR/../<design_example_dir>/
   software/dynamic_reconfiguration_sim/
   nios_system_onchip_memory2_0_onchip_memory2_0.hex"]"
   ```

3. Using the supported simulator of your choice, change to the testbench simulation directory to <*design_example_dir*>`/example_testbench/`<*simulator_name*>.
4. Run the simulation script for the simulator. The script compiles and runs the testbench in the simulator. Refer to the table *Steps to Simulate the Testbench*.
5. Analyze the results. The successful testbench performs the dynamic reconfiguration (DR) operations, sends and transmits packets for each DR operation, and displays "`Nios has completed its transactions`" and "`Simulation PASSED`" after completing the simulation.

Table 27. 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`
Cadence Xcelium	In the command line, type `sh run_xcelium.sh`
Synopsys VCS/VCS MX	In the command line, type `sh run_vcs.sh` or `sh run_vcsmx.sh` *Note:* `run_vcs.sh` is only available if you select **Verilog** as the **Generated HDL Format**. If you select **VHDL** as the **Generated HDL Format**, you must simulate the testbench with a mixed language simulator using `run_vcsmx.sh`.

Notice: For Nios II-based testbench, the simulation runs for more than 5 hours.

4.1.3.2. Running the Simulation with New HEX File

If you modify the C-code design example simulation source files, you must generate a `.HEX` file using Nios II Software Build Tools (SBT) for Eclipse.

1. In the Intel Quartus Prime Pro Edition software, select **Tools ➤ Nios II Software Build Tools for Eclipse**.
2. Create a new workspace when the **Workspace Launcher** window prompt appears. Click **OK** to open the workspace.
3. In the **Nios II - Eclipse** window, select **File ➤ New ➤ Nios II Application and BSP from Template**. A **Nios II Application and BSP from Template** appears.
4. In the **Nios II Application and BSP from Template** window, fill in the following information:
   - For `SOPC Information File name`, browse to *<design_example_dir>*/`hardware_test_design/nios_system` and open the SOPC Information File (`nios_system.sopcinfo`) for your design. Click **OK** to select the file and Eclipse automatically loads all CPU settings.
   - For `Project name`, specify your desired project name. This example uses `dynamic_reconfiguration_simulation`.
5. Click **Finish** to generate the project. The Intel Quartus Prime Pro Edition software creates a new directory named `software` in the specified project location.
6. Replace the C-code source files located in your new software directory (*<design_example_dir>*`/hardware_test_design/software/dynamic_reconfiguration_simulation`) with the following C-code source files from the `<design_example_dir>/software/dynamic_reconfiguration_sim` design:
   - `c3_reconfig.c`
   - `c3_reconfig.h`
   - `c3_function.c`
   - `flow.c`
   - `main.c`
   - `packet_gen.c`
   - `packet_gen.h`

   Note: The `packet_gen.c` and `packet_gen.h` files are only applicable for Ethernet dynamic reconfiguration (DR) design example and Ethernet to CPRI DR design example variants.
7. In the **Nios II - Eclipse** window, press **F5** or right-click your project and select **Refresh** to refresh the window and reload the new files into the project.
8. On the **Project Explorer** view, right-click the `dynamic_reconfiguration_simulation` and select **Build Project**. Ensure the `dynamic_reconfiguration_simulation.elf` file is generated in the new *<design_example_dir>*`/hardware_test_design/software/dynamic_reconfiguration_simulation` directory.

9. To generate a new HEX file, right-click the dynamic_reconfiguration_simulation in the **Project Explorer** view, point to **Make Targets** and select **Build**. A **Make Targets** dialog box appears.
10. In the **Make Targets** dialog box, select mem_init_generate.
11. Click **Build**. The mem_init_generate creates the new HEX (nios_system_onchip_memory2_0_onchip_memory2_0.hex) file. The new HEX file resides in the <*design_example_dir*>/hardware_test_design/software/dynamic_reconfiguration_simulation/mem_init directory.

Follow these steps to simulate the testbench:

1. Open the <*simulator_name*>_files.tcl script in the <*design_example_dir*>/example_testbench/setup_scripts/common directory.
2. Edit the TCL script to change the existing nios_system_onchip_memory2_0_onchip_memory2_0.hex file directory to the new HEX file generated from the Nios II SBT for Eclipse:

   For example, change the following line in the TCL script from:

   ```
   lappend memory_files "[normalize_path "$QSYS_SIMDIR/../<design_example_dir>/
   hardware_test_design/ip/nios_system/nios_system_onchip_memory2_0/
   altera_avalon_onchip_memory2_191/sim/
   nios_system_onchip_memory2_0_onchip_memory2_0.hex"]"
   ```

   to

   ```
   lappend memory_files "[normalize_path "$QSYS_SIMDIR/../<design_example_dir>/
   hardware_test_design/software/dynamic_reconfiguration_simulation/mem_init/
   nios_system_onchip_memory2_0_onchip_memory2_0.hex"]"
   ```
3. Using the supported simulator of your choice, change to the testbench simulation directory to <*design_example_dir*>/example_testbench/<*simulator_name*>.
4. Run the simulation script for the simulator. The script compiles and runs the testbench in the simulator. Refer to Table 27 on page 94.
5. Analyze the results. The successful testbench performs the DR operations, sends and transmits packets for each DR operation, and displays "Nios has completed its transactions" and "Simulation PASSED" after completing the simulation.

   Notice: For Nios II-based testbench, the simulation runs for more than 5 hours.

4.1.3.3. Performing the Link Initialization

The link initialization is part of the default simulation test. You should perform link initialization before each simulation test. The default HEX file provided for the simulation contains this step.

Follow these steps to perform link initialization:

1. Wait for PIO_OUT[0] (o_ehip_ready) goes high.
2. Enable PMA loopback.
3. Wait for PIO_OUT[3:0] = 0xF (o_tx_ptp_ready, o_sl_rx_pcs_ready, o_sl_rx_block_lock, and o_ehip_ready asserted).

4. Continuously send packets to the clock data recover (CDR) receiver (RX) deskew training and wait until PIO_OUT[4] (`o_rx_ptp_ready` goes high.
5. Clear Ethernet statistic counters.
6. Enable the packet generator to start sending packets of data. Check the transmitter (TX) packet count statistic counter to confirm all packets are sent.
7. Check that the packet generator received all expected packets. Confirm the `checker_pass` status and wait for `PIO_OUT[3:0]` = 0xF (`checker_pass`, `o_sl_rx_pcs_ready`, `o_sl_rx_block_lock`, and `o_ehip_ready` asserted).
8. Disable the packet generator to stop sending packets.

4.1.4. Compiling and Configuring the Design Example in Hardware

To compile the hardware design example and configure it on your Intel Agilexdevice, 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_ehipc3.qpf`.
3. On the Processing menu, click **Start Compilation**.
4. After successful compilation, a `.sof` file is available in `<design_example_dir>/hardware_test_design` directory. Follow these steps to program the hardware design example on the Intel Agilexdevice:
    a. On the **Tools** menu, click **Programmer**.
    b. In the Programmer, click **Hardware Setup**.
    c. Select a programming device.
    d. Select and add the Agilex TX 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 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

4.1.5. Testing the E-tile Dynamic Reconfiguration Hardware Design Example

After you compile theE-Tile Dynamic Reconfiguration Design Example and configure it on your device, you can use the Nios II Software Build Tools (SBT) for Eclipse to compile and test the design in hardware.

4.1.5.1. Running the Design Example in Hardware

If you select **Agilex TX Transceiver Signal Integrity Development Kit** option as the **Target Development Kit** in the E-Tile Dynamic Reconfiguration Design Example parameter editor in Intel Quartus Prime Pro Edition software, refer to Power Management Setting for Agilex Transceiver Signal Integrity Development Kit on page 100 on how to configure the power management setting that can be included in the Quartus Setting File (`.qsf`) for the Agilex TX Transceiver Signal Integrity Development Kit.

Follow the steps below to run the design example in hardware:

1. In the Intel Quartus Prime Pro Edition software, compile the design example with the power management setting included to obtain a working SRAM Object File (`.sof`) file.
2. Download the `.sof` file to the Agilex TX Transceiver Signal Integrity Development Kit.
3. Configure the board clock control. Open on-board clock control. Select Si5341A(U3) to program OUT0 = 153.6Mhz, OUT5 = 184.32Mhz. This step is only applicable for Ethernet to CPRI DR design example variants.
4. In the Intel Quartus Prime Pro Edition software, select **Tools ➤ Nios II Software Build Tools for Eclipse**.
5. Create a new workspace when the **Workspace Launcher** window prompt appears. Click **OK** to open the workspace.
6. In the **Nios II - Eclipse** window, select **File ➤ New ➤ Nios II Application and BSP from Template**. A **Nios II Application and BSP from Template** appears.
7. In the **Nios II Application and BSP from Template** window, fill in the following information:
    - For `SOPC Information File name`, browse to `<`*`design_example_dir`*`>/hardware_test_design/nios_system` and open the SOPC Information File (`nios_system.sopcinfo`) for your design. Click **OK** to select the file and Eclipse automatically loads all CPU settings.
    - For `Project name`, specify your desired project name. This example uses `dynamic_reconfiguration_hardware`.
8. Click **Finish** to generate the project. The Intel Quartus Prime Pro Edition software creates a new directory named `software` in the specified project location.
9. Replace the C-code source files located in your new software directory (`<`*`design_example_dir`*`>/hardware_test_design/software/dynamic_reconfiguration_hardware`) with the following C-code source files from the `<design_example_dir>/software/dynamic_reconfiguration_hardware` design:
    - `c3_reconfig.c`
    - `c3_reconfig.h`
    - `c3_function.c`
    - `flow.c`

- `main.c`
- `packet_gen.c`
- `packet_gen.h`

  Note: The `packet_gen.c` and `packet_gen.h` files are only applicable for Ethernet dynamic reconfiguration (DR) design example and Ethernet to CPRI DR design example variants.

10. In the **Nios II - Eclipse** window, press **F5** or right-click your project and select **Refresh** to refresh the window and reload the new files into the project.
11. On the **Project Explorer** view, right-click `dynamic_reconfiguration_hardware` and select **Build Project**. Ensure the `dynamic_reconfiguration_hardware.elf` file is generated in the new `<`*`design_example_dir`*`>/hardware_test_design/software/dynamic_reconfiguration_hardware` directory.
12. To run the hardware test, right-click `dynamic_reconfiguration_hardware` in the **Project Explorer** view, point to **Run As** and select **Nios II Hardware**.

    If the **Run Configurations** dialog box appears, verify that `Project name` and `ELF file` name contain relevant data, then click **Run**.

In the Interactive GUI dialog box, select the dynamic reconfiguration hardware test.

Note: The GUI dialog box varies based on the selected dynamic reconfiguration hardware test variant.

The following is a hardware test example for the 25G Ethernet with PTP and RS-FEC variant.

```
CPU is alive!


             Dynamic Reconfiguration Hardware Test

By default, the starting mode is 25G_PTP_FEC.
      Please choose one of Dynamic reconfiguration:
    0) 25G_PTP_FEC    -> 25G_PTP_noFEC -> 10G_PTP -> 25G_PTP_noFEC ->
25G_PTP_FEC -> 10G_PTP -> 25G_PTP_FEC
    1) 25G_PTP_FEC    -> 25G_PTP_noFEC
    2) 25G_PTP_noFEC  -> 25G_PTP_FEC
    3) 25G_PTP_FEC    -> 10G_PTP
    4) 10G_PTP        -> 25G_PTP_FEC
    5) 25G_PTP_noFEC  -> 10G_PTP
    6) 10G_PTP        -> 25G_PTP_noFEC
    9) Terminate test
       If you terminate test halfway, you must reload the .sof file before
retrigger the hardware test.

Enter a Valid Selection (0,1,3,9):
```

The following is a hardware test example for CPRI variants.

```
CPU is alive!


             Dynamic Reconfiguration Hardware Test

By default, the starting mode is CPRI24G_FEC.
      Please choose the Targeted mode available:
    1) CPRI24G
    2) CPRI12GFEC
    3) CPRI12G
    4) CPRI10GFEC
    5) CPRI10G
    6) CPRI9.8G
    7) CPRI6.0G
    8) CPRI4.9G
    9) CPRI3.0G
    a) CPRI2.4G
    9) Terminate test  -> If you terminate test halfway, you must reload
the .sof file before retrigger the hardware test.

Enter a Valid Selection:
```

4.1.5.2. Power Management Setting for Agilex Transceiver Signal Integrity Development Kit

If you select Agilex TX Transceiver Signal Integrity Development Kit option as the **Target Development Kit** in the E-Tile Dynamic Reconfiguration Design Example parameter editor in Intel Quartus Prime Pro Edition software, the target device used for the design example is set to default AGFB014R24A2E2VR0 with the pin assignments provided in the `.qsf` file.

The Agilex TX Transceiver Signal Integrity Development Kit (AGFB014R24A2E2VR0) is a voltage identification (VID) device. The `.qsf` file includes the power management setting. The following is an example of the specific power management setting that can be included in the `.qsf` file for the Agilex TX Transceiver Signal Integrity Development Kit:

```
set_global_assignment -name USE_PWRMGT_SCL SDM_IO0
set_global_assignment -name USE_PWRMGT_SDA SDM_IO12
set_global_assignment -name VID_OPERATION_MODE "PMBUS MASTER"
set_global_assignment -name PWRMGT_BUS_SPEED_MODE "400 KHZ"
set_global_assignment -name PWRMGT_SLAVE_DEVICE_TYPE OTHER
set_global_assignment -name PWRMGT_SLAVE_DEVICE0_ADDRESS 42
set_global_assignment -name PWRMGT_SLAVE_DEVICE1_ADDRESS 43
set_global_assignment -name PWRMGT_SLAVE_DEVICE2_ADDRESS 44
set_global_assignment -name PWRMGT_SLAVE_DEVICE3_ADDRESS 00
set_global_assignment -name PWRMGT_SLAVE_DEVICE4_ADDRESS 00
set_global_assignment -name PWRMGT_SLAVE_DEVICE5_ADDRESS 00
set_global_assignment -name PWRMGT_SLAVE_DEVICE6_ADDRESS 00
set_global_assignment -name PWRMGT_SLAVE_DEVICE7_ADDRESS 00
set_global_assignment -name PWRMGT_PAGE_COMMAND_ENABLE ON
set_global_assignment -name PWRMGT_VOLTAGE_OUTPUT_FORMAT "AUTO DISCOVERY"
set_global_assignment -name PWRMGT_TRANSLATED_VOLTAGE_VALUE_UNIT VOLTS”
```

However, if you select the `Other Development Kits` option as the **Target Development Kit**, the target device used for the design example follows the target device chosen in the project. You must set the pin assignment based on the base variant used.

Note: The E-Tile Dynamic Reconfiguration Design Example is a Nios II-based design. You can use the Nios II Software Build Tools (SBT) for Eclipse to perform the hardware test.

4.2. 10G/25G Ethernet Dynamic Reconfiguration Design Examples

The 10G/25G Ethernet Dynamic Reconfiguration design example demonstrates a dynamic reconfiguration solution for Intel Agilex devices using the E-Tile Ethernet IP for Intel Agilex FPGA core with the following variants:

Table 28. List of Supported Design Example Variants for 10G/25G Ethernet Dynamic Reconfiguration

Base Operation	Dynamic Reconfiguration Variants
25GE with RS-FEC and PTP	25GE with RS-FEC and PTP
	25GE with PTP
	10GE with PTP
25GE with RS-FEC	25GE with RS-FEC
	25GE
	10GE

4.2.1. Functional Description

4.2.1.1. Clocking Scheme

Figure 38. Clocking Scheme for 10G/25GE MAC+PCS with RS-FEC and PTP Dynamic Reconfiguration Design Example

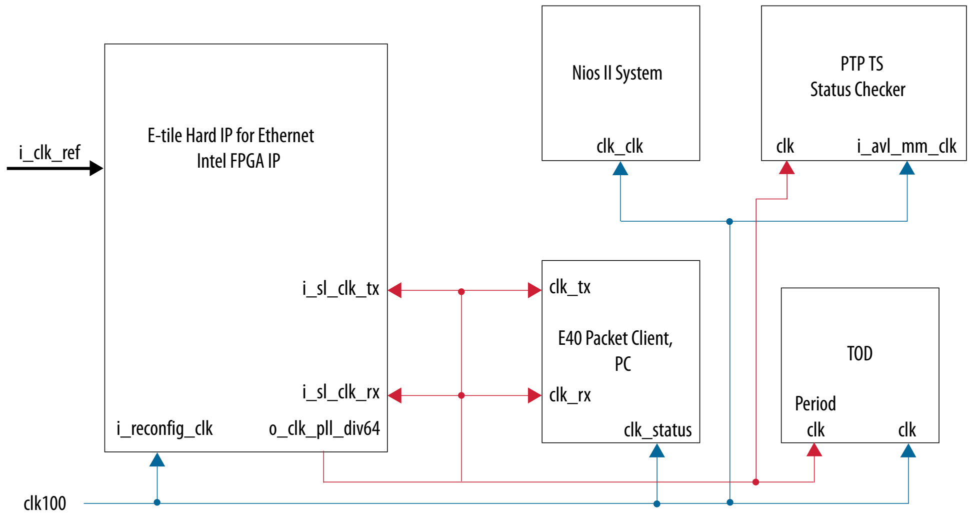
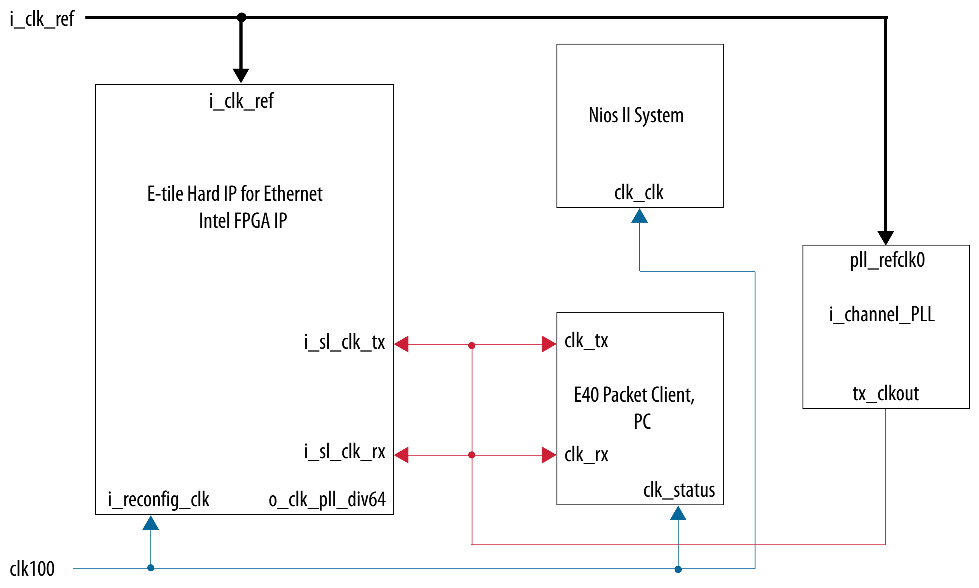


Figure 39. Clocking Scheme for 10G/25GE MAC+PCS with RS-FEC Dynamic Reconfiguration Design Example



Note: `i_channel_PLL` module is E-tile Transceiver PHY specific module that utilizes additional transceiver E-tile channel.

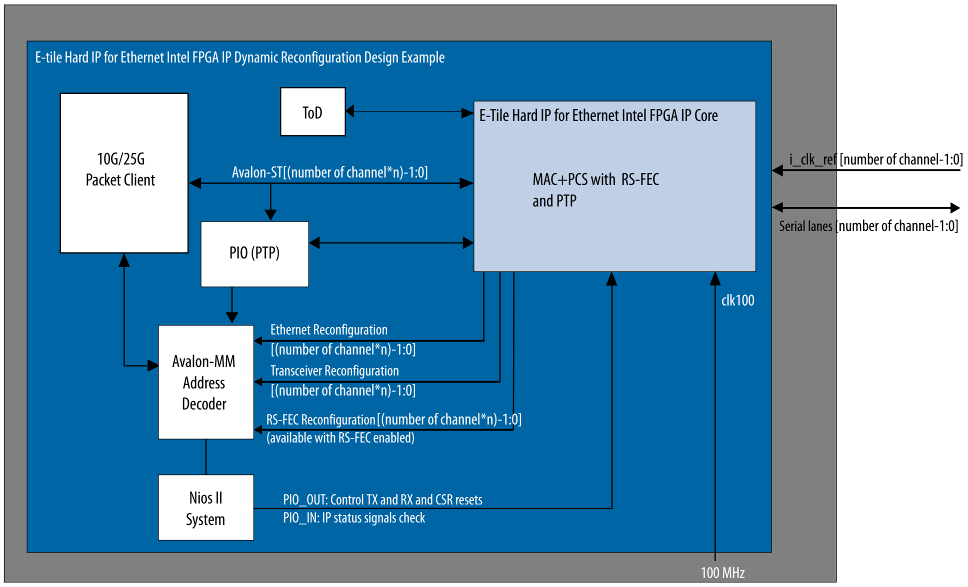
4.2.2. Simulation Design Examples

4.2.2.1. 10GE/25GE MAC+PCS with RS-FEC and PTP Simulation Dynamic Reconfiguration Design Example Components

The simulation block diagram below is generated using the following settings in the IP parameter editor:

1. **Ethernet Protocol** as **DR Protocol**.
2. Under the **10G/25G Ethernet Protocol** tab:
   a. **25G 1588PTP RS-FEC** as **Select DR Design**.
   b. **Other Development Kits** as the target development kit.

Figure 40. Simulation Block Diagram for E-Tile Ethernet IP for Intel Agilex FPGA 10GE/25GE with RS-FEC and PTP Dynamic Reconfiguration Design Example


E-tile Hard IP for Ethernet Intel FPGA IP Dynamic Reconfiguration Design Example
10G/25G Packet Client
ToD
E-Tile Hard IP for Ethernet Intel FPGA IP Core
MAC+PCS with RS-FEC and PTP
Avalon-ST[(number of channel*n)-1:0]
PIO (PTP)
i_clk_ref [number of channel-1:0]
Serial lanes [number of channel-1:0]
clk100
Avalon-MM Address Decoder
Ethernet Reconfiguration [(number of channel*n)-1:0]
Transceiver Reconfiguration [(number of channel*n)-1:0]
RS-FEC Reconfiguration [(number of channel*n)-1:0] (available with RS-FEC enabled)
Nios II System
PIO_OUT: Control TX and RX and CSR resets
PIO_IN: IP status signals check
100 MHz

The successful test displays the dynamic reconfiguration transition flow between various modes. Use preset HEX file provided for each design example or modify provided C code to enable specific transition simulation. For more information on HEX file, refer to Simulating the E-Tile Dynamic Reconfiguration Design Example Testbench on page 93.

To test a specific transition, reorder the dynamic reconfiguration transition flow tests in the `main.c` file and regenerate a new HEX file. Each test describes a transition from the starting rate to the destination rate.

This is the default simulation test sequence based on the provided HEX file.

1. Toggle `sl_tx_rst_n` and `sl_rx_rst_n` reset signals.
2. Link Initialization. For more information, refer to Performing the Link Initialization on page 96.
3. Dynamic reconfiguration (DR) test from 25G PTP with RS-FEC to 25G PTP without RS-FEC
4. DR test from 25G PTP without RS-FEC to 10G PTP
5. DR test from 10G PTP to 25G PTP without RS-FEC
6. DR test from 25G PTP without RS-FEC to 25G PTP with RS-FEC
7. DR test from 25G PTP with RS-FEC to 10G PTP
8. DR test from 10G PTP to 25G PTP with RS-FEC

Each of the dynamic reconfiguration tests follows these steps:

1. Assert `sl_tx_rst_n` and `sl_rx_rst_n` reset signals.
2. Disable SERDES. Use PMA attribute code 0x0001 in the *E-tile Transceiver PHY User Guide: PMA Attribute Codes* section.
3. Trigger PMA analog reset. For more information about register descriptions, refer to the *E-tile Transceiver PHY User Guide*.
4. Change transceiver TX bit/`refclk` ratio to the destination rate. The `refclk` is 156.25 MHz.
5. Change transceiver RX bit/`refclk` ratio to the destination rate. The `refclk` is 156.25 MHz.
6. Reconfigure the following registers for the Ethernet, RS-FEC, and transceiver blocks. For more information about the details of the changed register values, refer to the `c3_reconfig.c` file. For more information about the register descriptions, refer to the *E-tile Hard IP for Ethernet and CPRI PHY Intel FPGA IPs User Guide*.
7. Adjust the phase offset of a recovered clock. Use PMA attribute code 0x000E in the *E-tile Transceiver PHY User Guide: PMA Attribute Codes* section.
8. Enable SERDES. Use PMA attribute code 0x0001 in the *E-tile Transceiver PHY User Guide: PMA Attribute Codes* section.
9. Enable internal serial loopback. Use PMA attribute code 0x0008 in the *E-tile Transceiver PHY User Guide: PMA Attribute Codes* section.
10. Deassert `sl_tx_rst_n` and `sl_rx_rst_n` reset signals.
11. Wait for `PIO_OUT[4:0]` = 0x1F (`o_sl_rx_ptp_ready`, `o_sl_rx_pcs_ready`, `o_sl_rx_block_lock`, and `o_ehip_ready` asserted).
12. Clear Ethernet statistic counters.
13. Enable the packet generator to start sending packets of data.
14. Check for `checker_pass` status and waiting for `PIO_OUT[3:0]` = 0xF (`checker_pass`, `o_sl_rx_pcs_ready`, `o_sl_rx_block_lock`, and `o_ehip_ready` asserted).

The following sample output illustrates a successful simulation test run for a 25GE MAC+PCS with RS-FEC and PTP IP core variation.

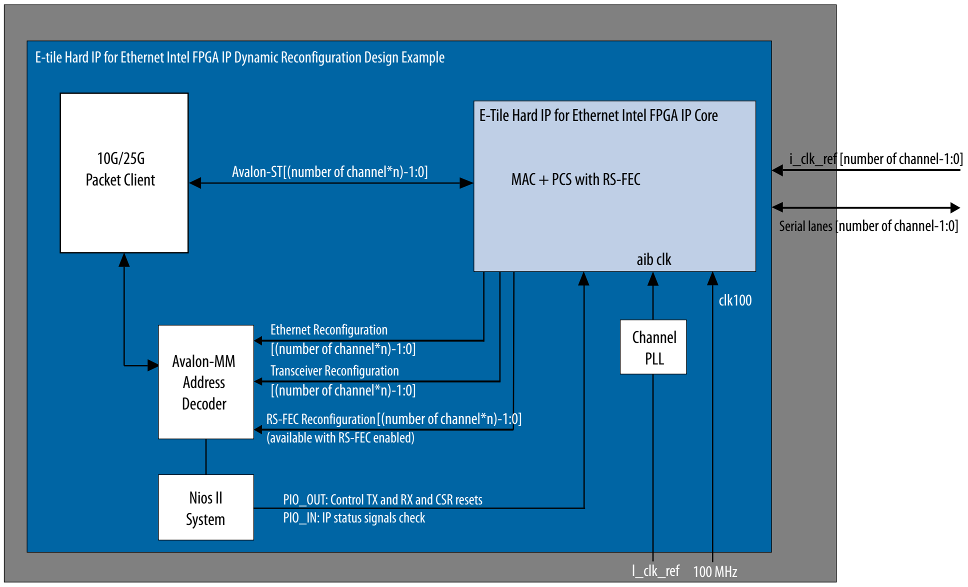
```
# CPU is alive!
# INFO:  PKT_RX_CNT received = 10
# INFO:  PKT_RX_CNT received = 20
# INFO:  PKT_RX_CNT received = 30
# INFO:  PKT_RX_CNT received = 40
# INFO:  PKT_RX_CNT received = 50
# INFO:  PKT_RX_CNT received = 60
# INFO:  PKT_RX_CNT received = 70
# End of test
# Nios has completed its transactions          4794387104
# Simulation PASSED          4794387104
# ** Note: $finish    : ./../basic_avl_tb_top.sv(587)
#    Time: 4794387104 ps  Iteration: 9  Instance: /basic_avl_tb_top
```

4.2.2.2. 10GE/25GE MAC+PCS with RS-FEC Simulation Dynamic Reconfiguration Design Example Components

The simulation block diagram below is generated using the following settings in the IP parameter editor:

1. **Ethernet Protocol** as **DR Protocol**.
2. Under the **10G/25G Ethernet Protocol** tab:
   a. **25G RS-FEC** as **Select DR Design**.
   b. **Other Development Kits** as the target development kit.

Figure 41. Simulation Block Diagram for E-Tile Ethernet IP for Intel Agilex FPGA 10GE/25GE with RS-FEC Dynamic Reconfiguration Design Example


E-tile Hard IP for Ethernet Intel FPGA IP Dynamic Reconfiguration Design Example
10G/25G Packet Client
Avalon-ST[(number of channel*n)-1:0]
E-Tile Hard IP for Ethernet Intel FPGA IP Core
MAC + PCS with RS-FEC
i_clk_ref [number of channel-1:0]
Serial lanes [number of channel-1:0]
aib clk
clk100
Channel PLL
Avalon-MM Address Decoder
Ethernet Reconfiguration [(number of channel*n)-1:0]
Transceiver Reconfiguration [(number of channel*n)-1:0]
RS-FEC Reconfiguration [(number of channel*n)-1:0] (available with RS-FEC enabled)
Nios II System
PIO_OUT: Control TX and RX and CSR resets
PIO_IN: IP status signals check
I_clk_ref
100 MHz

The successful test displays the dynamic reconfiguration transition flow between various modes. Use preset HEX file provided for each design example or modify provided C code to enable specific transition simulation. For more information on HEX file, refer to Simulating the E-Tile Dynamic Reconfiguration Design Example Testbench on page 93.

To test a specific transition, reorder the dynamic reconfiguration transition flow tests in the `main.c` file and regenerate a new HEX file. Each test describes a transition from the starting rate to the destination rate.

This is the default simulation test sequence based on the provided HEX file.

1. Toggle `sl_tx_rst_n` and `sl_rx_rst_n` reset signals.
2. Link Initialization. For more information, refer to Performing the Link Initialization on page 96.
3. Dynamic reconfiguration (DR) test from 25G PTP with RS-FEC to 25G PTP without RS-FEC
4. DR test from 25G PTP without RS-FEC to 10G PTP
5. DR test from 10G PTP to 25G PTP without RS-FEC
6. DR test from 25G PTP without RS-FEC to 25G PTP with RS-FEC
7. DR test from 25G PTP with RS-FEC to 10G PTP
8. DR test from 10G PTP to 25G PTP with RS-FEC

Each of the dynamic reconfiguration tests follows these steps:

1. Assert `sl_tx_rst_n` and `sl_rx_rst_n` reset signals.
2. Disable SERDES. Use PMA attribute code 0x0001 in the *E-tile Transceiver PHY User Guide: PMA Attribute Codes* section.
3. Trigger PMA analog reset. For more information about register descriptions, refer to the *E-tile Transceiver PHY User Guide*.
4. Change transceiver TX bit/`refclk` ratio to the destination rate. The `refclk` is 156.25 MHz.
5. Change transceiver RX bit/`refclk` ratio to the destination rate. The `refclk` is 156.25 MHz.
6. Reconfigure the following registers for the Ethernet, RS-FEC, and transceiver blocks. For more information about the details of the changed register values, refer to the `c3_reconfig.c` file. For more information about the register descriptions, refer to the *E-tile Hard IP for Ethernet and CPRI PHY Intel FPGA IPs User Guide*.
7. Adjust the phase offset of a recovered clock. Use PMA attribute code 0x000E in the *E-tile Transceiver PHY User Guide: PMA Attribute Codes* section.
8. Enable SERDES. Use PMA attribute code 0x0001 in the *E-tile Transceiver PHY User Guide: PMA Attribute Codes* section.
9. Enable internal serial loopback. Use PMA attribute code 0x0008 in the *E-tile Transceiver PHY User Guide: PMA Attribute Codes* section.
10. Deassert `sl_tx_rst_n` and `sl_rx_rst_n` reset signals.
11. Wait for `PIO_OUT[4:0]` = 0x1F (`o_sl_rx_ptp_ready`, `o_sl_rx_pcs_ready`, `o_sl_rx_block_lock`, and `o_ehip_ready` asserted).
12. Clear Ethernet statistic counters.
13. Enable the packet generator to start sending packets of data.
14. Check for `checker_pass` status and waiting for `PIO_OUT[3:0]` = 0xF (`checker_pass`, `o_sl_rx_pcs_ready`, `o_sl_rx_block_lock`, and `o_ehip_ready` asserted).

The following sample output illustrates a successful simulation test run for a 25GE MAC+PCS with RS-FEC IP core variation.

```
# CPU is alive!
# INFO:  PKT_RX_CNT received = 10
# INFO:  PKT_RX_CNT received = 20
# INFO:  PKT_RX_CNT received = 30
# INFO:  PKT_RX_CNT received = 40
# INFO:  PKT_RX_CNT received = 50
# INFO:  PKT_RX_CNT received = 60
# INFO:  PKT_RX_CNT received = 70
# End of test
# Nios has completed its transactions          4535480000
# Simulation PASSED          4535480000
# ** Note: $finish    : ./../basic_avl_tb_top.sv(522)
#    Time: 4535480 ns  Iteration: 1  Instance: /basic_avl_tb_top
```

4.2.3. Hardware Design Examples

In general, simulation design examples and hardware design examples follow the same flow except for a PMA adaptation flow.

Intel Quartus Prime Pro Edition 19.4 version supports switching between internal serial loopback without PMA adaptation, the internal serial loopback with PMA adaptation, and the external loopback with PMA adaptation. To select the loopback mode, configure TEST_MODE parameter in the flow.c.

TEST_MODE	Mode
0	Internal serial loopback without PMA adaptation
1	Internal serial loopback with PMA adaptation
Others	External serial loopback with PMA adaptation

For speed switching to 24G, 12G, 10G, and 9.8G speed modes, setting TEST_MODE to a non-zero value enables the general PMA adaptation. This PMA adaptation with zero effort configuration to shorten the link up time to less than 100 ms as per CPRI specifications requirement.

For speed switching to 6G speed modes or lower, the hardware design examples use the manual CTLE function to shorten the link up time to less than 100 ms per CPRI specification requirement. For more information about manual CTLE configuration, refer to the *E-Tile Transceiver PHY User Guide*.

4.2.3.1. 10GE/25GE MAC+PCS with RS-FEC and PTP Hardware Dynamic Reconfiguration Design Example Components

The 10GE/25GE hardware dynamic reconfiguration design example includes the following components:

- E-tile Ethernet IP for Intel Agilex FPGA core.
- Client logic that coordinates the programming of the IP core and packet generation.
- Time-of-day (ToD) module to provide a continuous flow of current time-of-day information to the IP core.
- PIO block to store RX and TX PTP timestamp for accuracy calculation and to send PTP 2-step timestamp request.
- Avalon-MM address decoder to decode reconfiguration address space for MAC, transceiver, and RS-FEC modules during reconfiguration accesses.
- Nios II System that communicates with the Nios II Software Build Tools (SBT) for Eclipse. You communicate with the client logic and E-tile Ethernet IP for Intel Agilex FPGA through the tool.

By default, the hardware test run uses the internal serial loopback mode. The following sample outputs illustrate a successful hardware test run for a 25GE, MAC +PCS, RS-FEC, with PTP IP core variation. The hardware test uses this user control GUI to switch to any supported mode.

```
CPU is alive!


                Dynamic Reconfiguration Hardware Test
```

```
By default, the starting mode is 25G_PTP_FEC.
      Please choose one of Dynamic reconfiguration:
    0) 25G_PTP_FEC    -> 25G_PTP_noFEC -> 10G_PTP -> 25G_PTP_noFEC ->
25G_PTP_FEC -> 10G_PTP -> 25G_PTP_FEC
    1) 25G_PTP_FEC    -> 25G_PTP_noFEC
    2) 25G_PTP_noFEC  -> 25G_PTP_FEC
    3) 25G_PTP_FEC    -> 10G_PTP
    4) 10G_PTP        -> 25G_PTP_FEC
    5) 25G_PTP_noFEC  -> 10G_PTP
    6) 10G_PTP        -> 25G_PTP_noFEC
    9) Terminate test
       If you terminate test halfway, you must reload the .sof file before
retrigger the hardware test.

Enter a Valid Selection (0,1,3,9):
```

4.2.3.2. 10GE/25GE MAC+PCS with RS-FEC Hardware Dynamic Reconfiguration Design Example Components

The 10GE/25GE hardware dynamic reconfiguration design example includes the following components:

- E-tile Ethernet IP for Intel Agilex FPGA core.
- Client logic that coordinates the programming of the IP core and packet generation.
- Avalon-MM address decoder to decode reconfiguration address space for MAC, transceiver, and RS-FEC modules during reconfiguration accesses.
- Nios II System that communicates with the Nios II Software Build Tools (SBT) for Eclipse. You communicate with the client logic and E-tile Ethernet IP for Intel Agilex FPGA through the tool.
- Native PHY in PMA Direct mode that acts as a channel PLL to provide EMIB clocks (for example, 402.8 MHz and 805.6 MHz), as required by the E-tile Ethernet IP for Intel Agilex FPGA core.

The following sample outputs illustrate a successful hardware test run for a 25GE, MAC+PCS, RS-FEC IP core variation:

```
CPU is alive!


             Dynamic Reconfiguration Hardware Test

By default, the starting mode is 25G_FEC.
      Please choose one of Dynamic reconfiguration:
    0) 25G_FEC   -> 25G_noFEC -> 10G -> 25G_noFEC -> 25G_FEC -> 10G -> 25G_FEC
    1) 25G_FEC   -> 25G_noFEC
    2) 25G_noFEC -> 25G_FEC
    3) 25G_FEC   -> 10G
    4) 10G       -> 25G_FEC
    5) 25G_noFEC -> 10G
    6) 10G       -> 25G_noFEC
    9) Terminate test
       If you terminate test halfway, you must reload the .sof file before
retrigger the hardware test.

Enter a Valid Selection (0,1,3,9):
```

4.2.4. 10GE/25GE Design Example Interface Signals

The following signals are hardware dynamic reconfiguration design example signals for all 10GE/25GE variants.

Table 29. 10GE/25GE Dynamic Reconfiguration Design Example Hardware Design Example Interface Signals

Signal	Direction	Comments
`clk100`	Input	Input clock for reconfiguration. Drive at 100 MHz. The intent is to drive this from a 100 Mhz oscillator on the board.
`cpu_resetn`	Input	Global reset for Nios II system.
`i_clk_ref` [(4)]	Input	Reference clock 25G IP core. Drive at 156.25MHz.
`o_tx_serial`	Output	Transmit serial data.
`i_rx_serial`	Input	Receiver serial data.

4.2.5. 10GE/25GE Design Examples Registers

Table 30. E-tile Ethernet IP for Intel Agilex FPGA Hardware Design Examples Register Map

Word Offset	Register Category
0x000000 – 0x000FFF	Ethernet MAC and PCS registers
0x001000 – 0x001FFF	Packet Generator and Checker registers
0x002000 – 0x002FFF	PTP monitoring registers
0x010000 – 0x0107FF	RS-FEC configuration registers
0x100000 – 0x1FFFFF	Transceiver registers

Table 31. Packet Client Registers

You can customize the E-tile Ethernet IP for Intel Agilex 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.	N/A	RW
0x1001	PKT_CL_CLNT	[31:0]	Four characters of IP block identification string CLNT.	N/A	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[29:11]: Reserved. • Bit[10:0]: These bits specify the transmit packet size in bytes.	0x25800040	RW
0x1009	`Packet Number Control`	[31:0]	Specify the number of packets to transmit from the packet generator.	0xA	RW

continued...

(4) `i_clk_ref` is also used in the 25G + RS-FEC design to provide clock to to a PMA direct module, which acts as a channel PLL to supply the required E-tile Ethernet TX/RX clocks and EMIB clocks.

Addr	Name	Bit	Description	HW Reset Value	Access
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: Reserved — 01: Fixed mode — 10: Reserved • 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 32 bits	[31:0]	Source address (lower 32 bits).	0x43210ADD	RW
0x1014	Source address upper 16 bits	[15:0]	Source address (upper 16 bits).	0x8765	RW

4.3. 25G Ethernet to CPRI Dynamic Reconfiguration Design Example

The 25G Ethernet to CPRI Dynamic Reconfiguration design example demonstrates a dynamic reconfiguration solution for Intel Agilex devices using the E-tile Ethernet IP for Intel Agilex FPGA IP core with the following variants:

Table 32. Supported Design Example Variants for 25G Ethernet to CPRI Dynamic Reconfiguration

Base Operation	Variants that Supports Dynamic Reconfiguration
25GE with RS-FEC and PTP	25GE with RS-FEC and PTP
	24GE CPRI with RS-FEC
	10GE CPRI

continued...

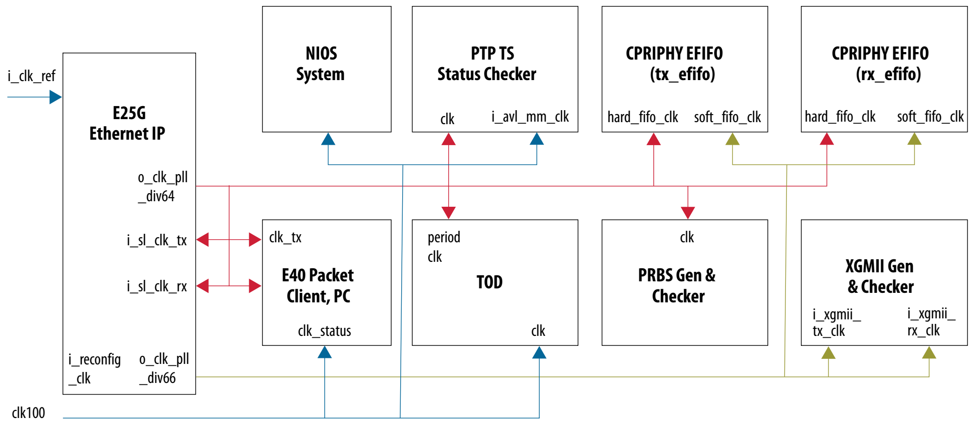
Send Feedback

Base Operation	Variants that Supports Dynamic Reconfiguration
	9.8GE CPRI
	4.9GE CPRI
	2.4GE CPRI

4.3.1. Functional Description

4.3.1.1. Clocking Scheme

Figure 42. Clocking Scheme 25G Ethernet to CPRI Dynamic Reconfiguration Design Example



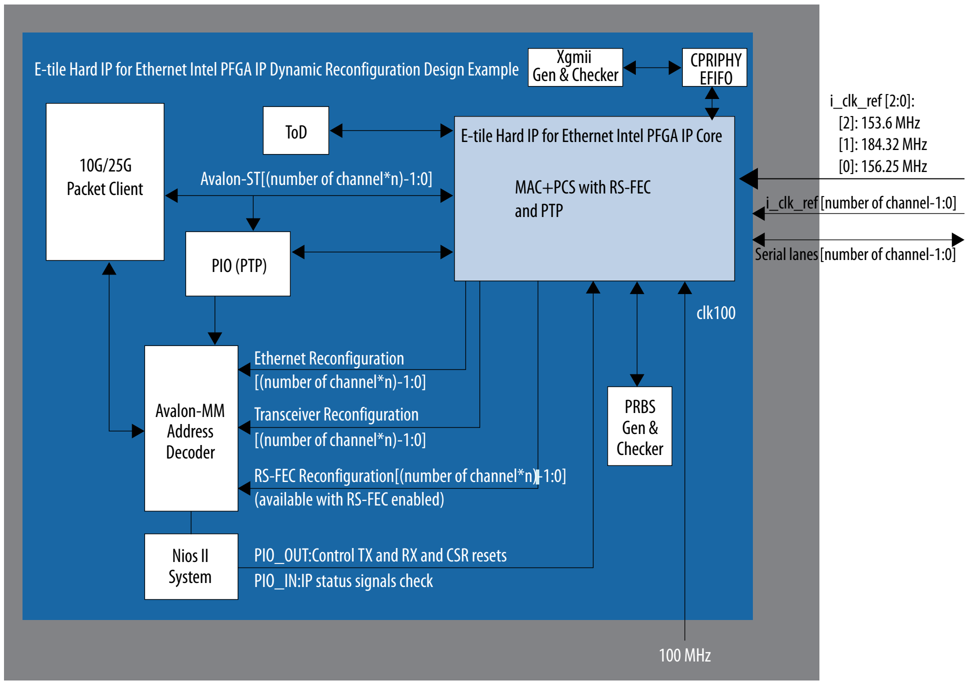
4.3.2. Simulation Design Examples

4.3.2.1. 25GE MAC+PCS with RS-FEC and PTP to CPRI Simulation Dynamic Reconfiguration Design Example Components

The simulation block diagram below is generated using the following settings in the IP parameter editor:

1. **25G Ethernet to CPRI Protocol** as **DR Protocol**.
2. Under the **25G Ethernet to CPRI Protocol** tab:
   a. **25G PTP RS-FEC** as **Select DR Design**.
   b. **Other Development Kits** as the target development kit.

Figure 43. Simulation Block Diagram for E-Tile Ethernet IP for Intel Agilex FPGA 25G Ethernet to CPRI Dynamic Reconfiguration Design Example


E-tile Hard IP for Ethernet Intel PFGA IP Dynamic Reconfiguration Design Example
Xgmii Gen & Checker
CPRIPHY EFIFO
10G/25G Packet Client
ToD
Avalon-ST[(number of channel*n)-1:0]
E-tile Hard IP for Ethernet Intel PFGA IP Core
MAC+PCS with RS-FEC and PTP
i_clk_ref [2:0]:
[2]: 153.6 MHz
[1]: 184.32 MHz
[0]: 156.25 MHz
i_clk_ref [number of channel-1:0]
Serial lanes [number of channel-1:0]
PIO (PTP)
clk100
Avalon-MM Address Decoder
Ethernet Reconfiguration [(number of channel*n)-1:0]
Transceiver Reconfiguration [(number of channel*n)-1:0]
RS-FEC Reconfiguration[(number of channel*n)-1:0] (available with RS-FEC enabled)
PRBS Gen & Checker
Nios II System
PIO_OUT:Control TX and RX and CSR resets
PIO_IN:IP status signals check
100 MHz

The successful test displays the dynamic reconfiguration transition flow between various modes. Use preset HEX file provided for each design example or modify provided C code to enable specific transition simulation. For more information on HEX file, refer to Simulating the E-Tile Dynamic Reconfiguration Design Example Testbench on page 93.

To test a specific transition, reorder the dynamic reconfiguration transition flow tests in the `main.c` file and regenerate a new HEX file. Each test describes a transition from the starting rate to the destination rate.

This is the default simulation test sequence based on the provided HEX file.

1. Toggle `sl_tx_rst_n` and `sl_rx_rst_n` reset signals.
2. Link Initialization. For more information, refer to Performing the Link Initialization on page 96.
3. Dynamic reconfiguration (DR) test from 25G PTP with RS-FEC to 24G CPRI with RS-FEC
4. DR test from 24G CPRI with RS-FEC to 25G PTP with RS-FEC
5. DR test from 25G PTP with RS-FEC to 10G CPRI
6. DR test from 10G CPRI to 25G PTP with RS-FEC
7. DR test from 25G PTP with RS-FEC to 9.8G CPRI
8. DR test from 9.8G CPRI to 25G PTP with RS-FEC
9. DR test from 25G PTP with RS-FEC to 4.9G CPRI

10. DR test from 4.9G CPRI to 25G PTP with RS-FEC
11. DR test from 25G PTP with RS-FEC to 2.4G CPRI
12. DR test from 2.4G CPRI to 25G PTP with RS-FEC

Each of the dynamic reconfiguration tests follows these steps:

1. Assert `sl_tx_rst_n` and `sl_rx_rst_n` reset signals.
2. Disable SERDES. Use PMA attribute code 0x0001 in the *E-tile Transceiver PHY User Guide: PMA Attribute Codes* section.
3. Perform reference clock mux switching. For more information about the details of the changed register values, refer to the `c3_reconfig.c` file.
    a. Switch the PMA controller clock to the transceiver `refclk1` clock.
    b. Change `refclk` reference clock from the original speed mode clock to the destination speed mode clock.

        Note: For information on speed mode clocks, refer to 25G Ethernet to CPRI Design Example Interface Signals on page 115.

    c. Switch the PMA controller clock to the transceiver `refclk0` clock.

    Note: Steps i and iii are only applicable for Ethernet dynamic reconfiguration hardware test to avoid potential hardware glitch due to the reference clock switch operation. These steps are available in the hardware test code but skip in the simulation test code.
4. Trigger PMA analog reset. For more information about register descriptions, refer to the *E-tile Transceiver PHY User Guide*.
5. Reconfigure the registers for the Ethernet, RS-FEC, and transceiver blocks. For more information about register descriptions, refer to the *E-tile Transceiver PHY User Guide*.
6. Adjust the phase offset of a recovered clock. Use PMA attribute code 0x000E in the *E-tile Transceiver PHY User Guide: PMA Attribute Codes* section.
7. Enable SERDES. Use PMA attribute code 0x0001 in the *E-tile Transceiver PHY User Guide: PMA Attribute Codes* section.
8. Enable internal serial loopback. Use PMA attribute code 0x0008 in the *E-tile Transceiver PHY User Guide: PMA Attribute Codes* section.

4.3.3. Hardware Design Examples

In general, simulation design examples and hardware design examples follow the same flow except for a PMA adaptation flow.

Intel Quartus Prime Pro Edition 19.4 version supports switching between internal serial loopback without PMA adaptation, the internal serial loopback with PMA adaptation, and the external loopback with PMA adaptation. To select the loopback mode, configure TEST_MODE parameter in the flow.c.

`TEST_MODE`	Mode
0	Internal serial loopback without PMA adaptation
1	Internal serial loopback with PMA adaptation
Others	External serial loopback with PMA adaptation

For speed switching to 24G, 12G, 10G, and 9.8G speed modes, setting TEST_MODE to a non-zero value enables the general PMA adaptation. This PMA adaptation with zero effort configuration to shorten the link up time to less than 100 ms as per CPRI specifications requirement.

For speed switching to 6G speed modes or lower, the hardware design examples use the manual CTLE function to shorten the link up time to less than 100 ms per CPRI specification requirement. For more information about manual CTLE configuration, refer to the *E-Tile Transceiver PHY User Guide*.

4.3.3.1. 25GE MAC+PCS with RS-FEC and PTP to CPRI Hardware Dynamic Reconfiguration Design Example Components

The 25G Ethernet to CPRI hardware dynamic reconfiguration design example includes the following components:

- E-tile Ethernet IP for Intel Agilex FPGA core.
- Client logic that coordinates the programming of the IP core and packet generation.
- Client XGMII Pattern Generator and Checker that coordinates the programming of the IP core and packet generation.
- Client PRBS Pattern Generator and Checker that coordinates the programming of the IP core and packet generation.
- CPRI PHY E-FIFO that coordinates between XGMII Pattern Generator and checker and E-tile Ethernet IP for Intel Agilex FPGA core.
- Time-of-day (ToD) module to provide a continuous flow of current time-of-day information to the IP core.
- PIO block to store RX and TX PTP timestamp for accuracy calculation and to send PTP 2-step timestamp request.
- Avalon-MM address decoder to decode reconfiguration address space for MAC, transceiver, and RS-FEC modules during reconfiguration accesses.
- Nios II System that communicates with the Nios II Software Build Tools (SBT) for Eclipse. You communicate with the client logic and E-tile Ethernet IP for Intel Agilex FPGA through the tool.

The following sample outputs illustrate a successful hardware test run for a 25GE, MAC+PCS, RS-FEC, with PTP IP core variation:

```
CPU is alive!


             Dynamic Reconfiguration Hardware Test

By default, the starting mode is 25G_PTP_FEC.
      Please choose one of Dynamic reconfiguration:
    0) 25G_PTP_FEC -> 10G_PTP -> 25G_PTP_FEC -> CPRI_24G -> 25G_PTP_FEC ->
CPRI_10G -> 25G_PTP_FEC -> CPRI_9p8G -> 25G_PTP_FEC -> CPRI_4p9G -> 25G_PTP_FEC
-> CPRI_2p4G -> 25G_PTP_FEC
    1) 25G_PTP_FEC -> CPRI_24G
    2) CPRI_24G    -> 25G_PTP_FEC
    3) 25G_PTP_FEC -> CPRI_10G
    4) CPRI_10G    -> 25G_PTP_FEC
    5) 25G_PTP_FEC -> CPRI_9p8G
    6) CPRI_9p8G   -> 25G_PTP_FEC
    7) 25G_PTP_FEC -> CPRI_4p9G
    8) CPRI_4p9G   -> 25G_PTP_FEC
    9) 25G_PTP_FEC -> CPRI_2p4G
    a) CPRI_2p4G   -> 25G_PTP_FEC
    b) 25G_PTP_FEC -> 10G_PTP
    c) 10G_PTP     -> 25G_PTP_FEC
    d) Terminate test
       If you terminate test halfway, you must reload the .sof file before
retrigger the hardware test.

Enter a Valid Selection (0,1,3,5,7,9,b,d):
```

4.3.4. 25G Ethernet to CPRI Design Example Interface Signals

The following signals are hardware dynamic reconfiguration design example signals for 25G Ethernet to CPRI variants.

Table 33. 25G Ethernet to CPRI Dynamic Reconfiguration Design Example Hardware Interface Signals

Signal	Direction	Comments
`clk100`	Input	Input clock for reconfiguration. Drive at 100 MHz. The intent is to drive this from a 100 MHz oscillator on the board.
`cpu_resetn`	Input	Input reset for Nios II System.
`ref_clk156MHz`	Input	156.25 MHz input clock for the 25G Ethernet IP core. Connect to `i_clk_ref[0]` in 25G Ethernet IP core.
`ref_clk184MHz`	Input	184.32 MHz input clock for the 10G/24G CPRI mode. Connect to the `i_clk_ref[1]` in 25G Ethernet IP core.
`ref_clk153MHz`	Input	153.6 MHz input clock for the 2.4G/4.9G/9.8G CPRI mode. Connect to the `i_clk_ref[2]` in 25G Ethernet IP core.
`tx_serial_data/_n`	Output	Transmit serial data for channel PLL (PMA direct mode).
`rx_serial_data/_n`	Input	Receiver serial data for channel PLL (PMA direct mode).
`o_tx_serial`	Output	Transmit serial data.
`i_rx_serial`	Input	Receiver serial data.

4.3.5. 25G Ethernet to CPRI Design Examples Registers

Table 34. E-tile Ethernet IP for Intel Agilex FPGA Hardware Design Examples Register Map

Word Offset	Register Category
0x000000 – 0x000FFF	Ethernet MAC and PCS registers
0x001000 – 0x001FFF	Packet Generator and Checker registers
0x002000 – 0x002FFF	PTP monitoring registers
0x010000 – 0x0107FF	RS-FEC configuration registers
0x100000 – 0x1FFFFF	Transceiver registers

Table 35. Packet Client Registers

You can customize the E-tile Ethernet IP for Intel Agilex 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.	N/A	RW
0x1001	PKT_CL_CLNT	[31:0]	Four characters of IP block identification string CLNT.	N/A	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[29:11]: Reserved. • Bit[10:0]: These bits specify the transmit packet size in bytes.	0x25800040	RW
0x1009	Packet Number Control	[31:0]	Specify the number of packets to transmit from the packet generator.	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

continued...

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

4.4. CPRI Dynamic Reconfiguration Design Examples

The CPRI dynamic reconfiguration design example demonstrates a dynamic reconfiguration solution for devices using the E-tile CPRI PHY Intel FPGA IP core with the following variants.

Table 36. Supported Design Example Variants for CPRI Dynamic Reconfiguration

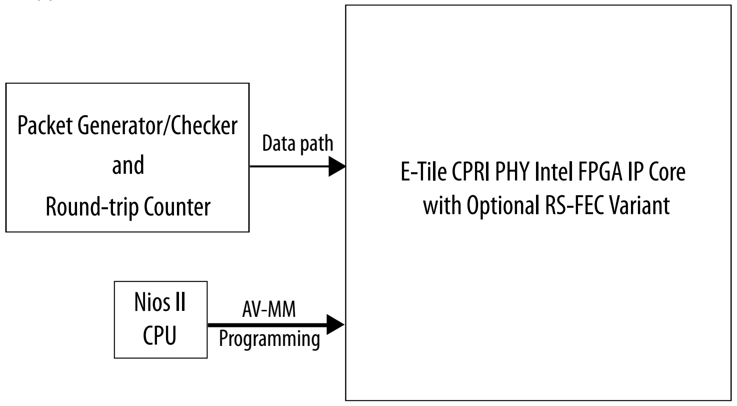
Base Operation	Variants that Supports Dynamic Reconfiguration
24G CPRI with RS-FEC	24G CPRI with RS-FEC
	24G CPRI
	12G CPRI with RS-FEC
	12G CPRI
	10G CPRI with RS-FEC
	10G CPRI
	9.8G CPRI
	6G CPRI
	4.9G CPRI
	3G CPRI
	2.4G CPRI
9.8G CPRI	9.8G CPRI
	6G CPRI
	4.9G CPRI
	3G CPRI
	2.4G CPRI

4.4.1. Functional Description

The design example consists of various components. The following block diagram shows the design components of the design example.

Figure 44. Block Diagram—CPRI Dynamic Reconfiguration Design Example

This block diagram applies to 24G CPRI with RS-FEC variant and 9.8G CPRI variant.



4.4.1.1. Clocking Scheme

Figure 45. Clocking Scheme for 24G CPRI with RS-FEC Dynamic Reconfiguration Design Example

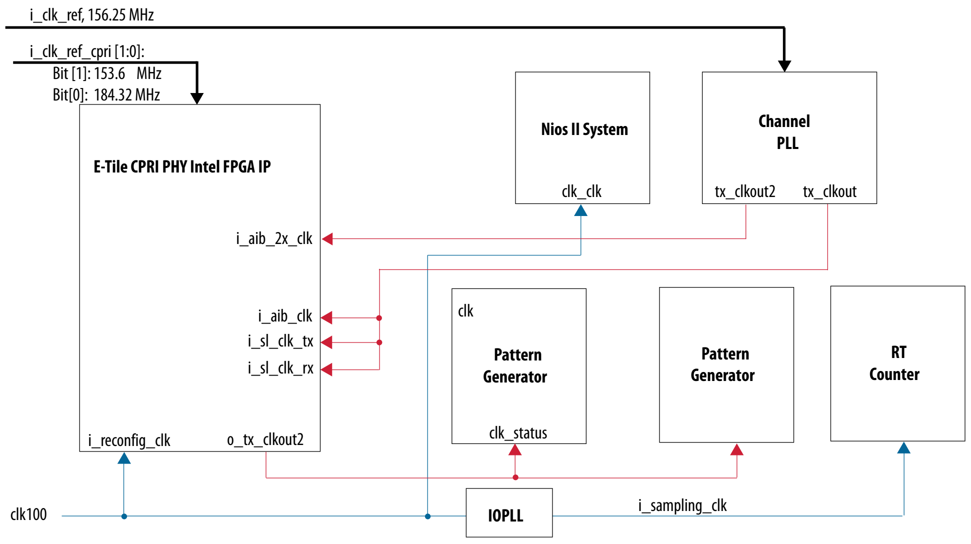
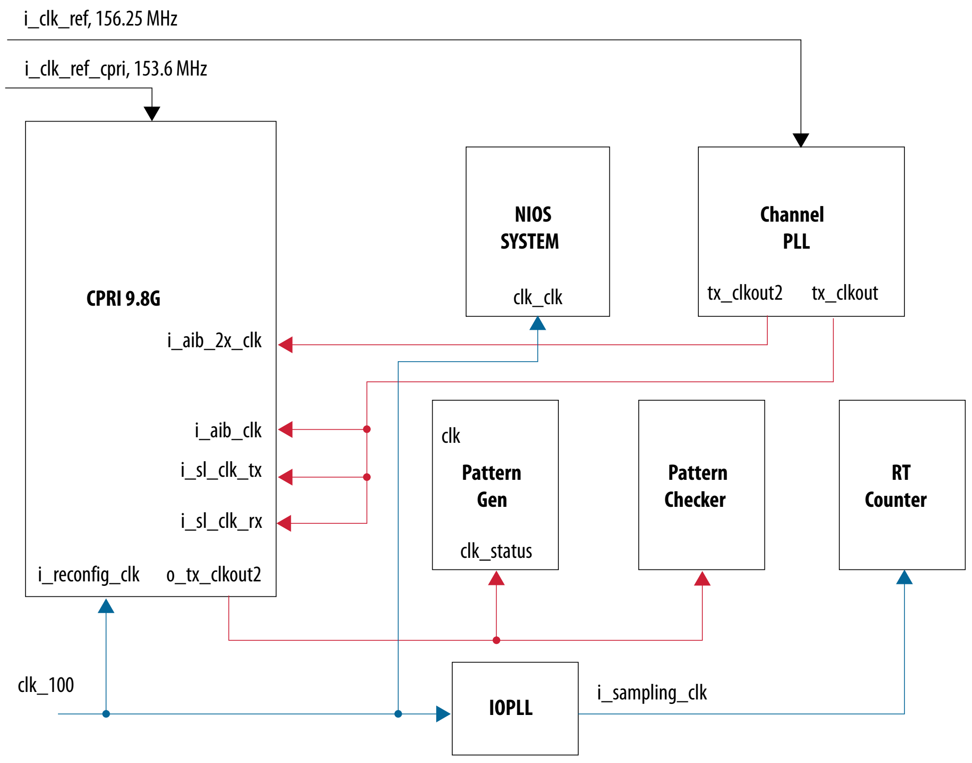


Figure 46. Clocking Scheme for 9.8G CPRI Dynamic Reconfiguration Design Example



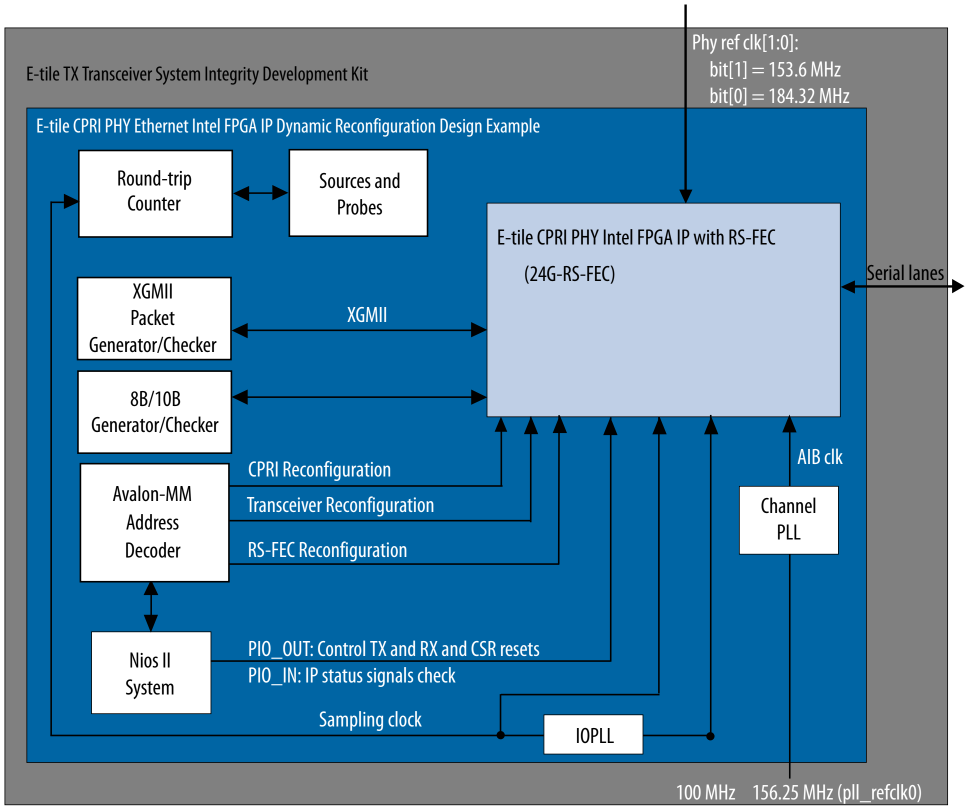
4.4.2. Simulation Design Examples

4.4.2.1. 24G CPRI PHY with RS-FEC Simulation Dynamic Reconfiguration Design Example Components

The simulation block diagram below is generated using the following settings in the IP parameter editor:

1. **CPRI Protocol** as **DR Protocol**.
2. Under the **CPRI Protocol** tab:
    a. **24G CPRI RS-FEC** as **Select DR Design**.
    b. **Agilex TX Transceiver Signal Integrity Development Kit** as the target development kit.

Figure 47. Simulation Block Diagram for 24G CPRI PHY with RS-FEC Dynamic Reconfiguration Design Example


E-tile TX Transceiver System Integrity Development Kit
Phy ref clk[1:0]:
bit[1] = 153.6 MHz
bit[0] = 184.32 MHz
E-tile CPRI PHY Ethernet Intel FPGA IP Dynamic Reconfiguration Design Example
Round-trip Counter
Sources and Probes
E-tile CPRI PHY Intel FPGA IP with RS-FEC
(24G-RS-FEC)
Serial lanes
XGMII Packet Generator/Checker
XGMII
8B/10B Generator/Checker
AIB clk
CPRI Reconfiguration
Avalon-MM Address Decoder
Transceiver Reconfiguration
Channel PLL
RS-FEC Reconfiguration
Nios II System
PIO_OUT: Control TX and RX and CSR resets
PIO_IN: IP status signals check
Sampling clock
IOPLL
100 MHz
156.25 MHz (pll_refclk0)

The successful test displays the dynamic reconfiguration transition flow between various modes. Use preset HEX file provided for each design example or modify provided C code to enable specific transition simulation. For more information on HEX file, refer to Simulating the E-Tile Dynamic Reconfiguration Design Example Testbench on page 93.

To test a specific transition, reorder the dynamic reconfiguration transition flow tests in the `main.c` file and regenerate a new HEX file. Each test describes a transition from the starting rate to the destination rate.

This is the default simulation test sequence based on the provided HEX file.

1. Toggle `sl_tx_rst_n` and `sl_rx_rst_n` reset signals.
2. Dynamic reconfiguration (DR) test from 24G CPRI with RS-FEC to 12G CPRI with RS-FEC
3. DR test from 12G CPRI with RS-FEC to 10G CPRI with RS-FEC
4. DR test from 10G CPRI with RS-FEC to 9.8G CPRI
5. DR test from 9.8G CPRI to 6G CPRI
6. DR test from 6G CPRI to 4.9G CPRI

7. DR test from 4.9G CPRI to 3G CPRI
8. DR test from 3G CPRI to 2.4G CPRI
9. DR test from 2.4G CPRI to 24G CPRI with RS-FEC

Each of the dynamic reconfiguration tests follows these steps:

1. Assert `sl_tx_rst_n` and `sl_rx_rst_n` reset signals.
2. Disable SERDES. Use PMA attribute code 0x0001 in the *E-tile Transceiver PHY User Guide: PMA Attribute Codes* section.
3. Perform reference clock mux switching. Use this step when reconfiguring from a high-speed mode (10G/ 12G/24G) to a PMA direct low-speed mode (2.4G/3G/ 4.9G/6G/9.8G) and vice versa. For more information about the details of the changed register values, refer to the `c3_reconfig.c` file.
    a. Switch the PMA controller clock to the transceiver `refclk1` clock.
    b. Change `refclk` reference clock from 184.32 MHz (`i_clk_ref[0]`) to 153.6 MHz (`i_clk_ref[1]`).
    c. Switch the PMA controller clock to the transceiver `refclk0` clock.

    Note: Steps i and iii are only applicable for Ethernet dynamic reconfiguration hardware tests to avoid potential hardware glitch due to the reference clock switch operation. These steps are available in the hardware test code but skip in the simulation test code.
4. Trigger PMA analog reset. For more information about register descriptions, refer to the *E-tile Transceiver PHY User Guide*.
5. Reconfigure the following registers for the Ethernet, RS-FEC, and transceiver blocks. For more information about the details of the changed register values, refer to the `c3_reconfig.c` file. For more information about the register descriptions, refer to the *E-tile Hard IP for Ethernet and CPRI PHY Intel FPGA IPs User Guide*.
6. Adjust the phase offset of a recovered clock. Use PMA attribute code 0x000E in the *E-tile Transceiver PHY User Guide: PMA Attribute Codes* section.
7. Enable SERDES. Use PMA attribute code 0x0001 in the *E-tile Transceiver PHY User Guide: PMA Attribute Codes* section.
8. Enable internal serial loopback. Use PMA attribute code 0x0008 in the *E-tile Transceiver PHY User Guide: PMA Attribute Codes* section.
9. Deassert `sl_tx_rst_n` and `sl_rx_rst_n` reset signals.
10. Wait for `PIO_OUT[3:0]` = 0x7 (`o_sl_rx_pcs_ready`, `o_sl_rx_block_lock`, and `o_ehip_ready` asserted).
11. Clear Ethernet statistic counters.
12. Enable the packet generator to start sending packets of data.
13. Check for `checker_pass` status and waiting for `PIO_OUT[3:0]` = 0xF (`checker_pass`, `o_sl_rx_pcs_ready`, `o_sl_rx_block_lock`, and `o_ehip_ready` asserted).
14. Disable the packet generator to stop sending packets.

The following sample output illustrates a successful simulation test run for a 24G MAC +PCS with RS-FEC IP core variation.

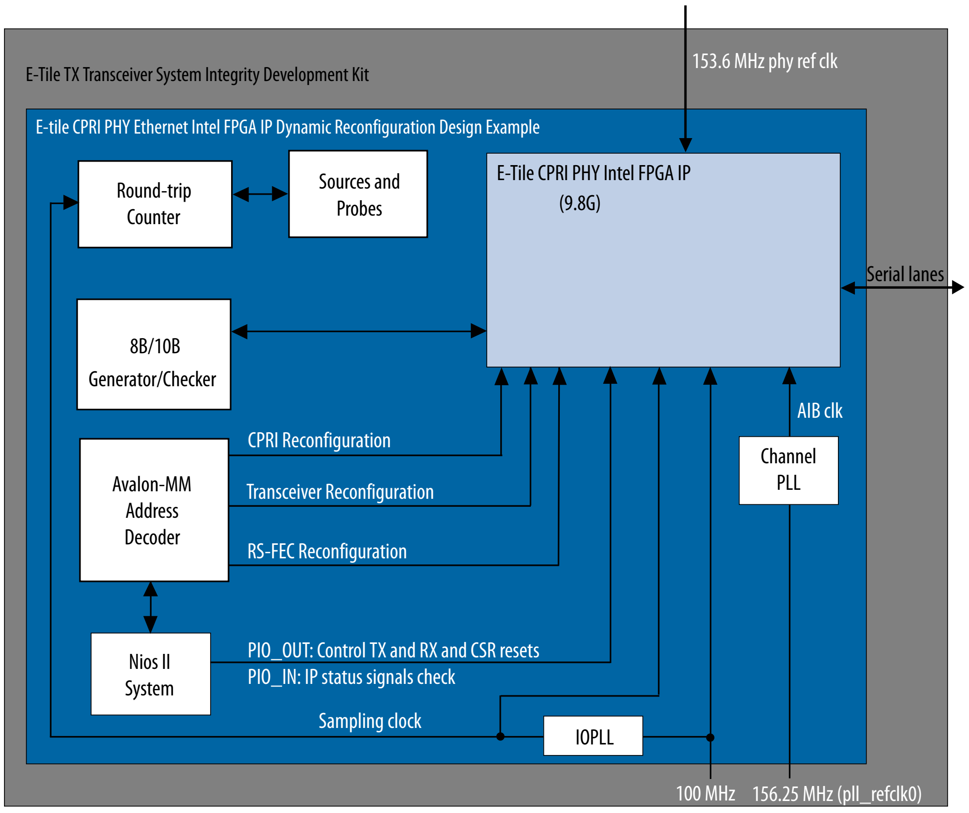
```
# CPU is alive!
# End of test
# Nios has completed its transactions          1995670000
# Simulation PASSED          1995670000
# ** Note: $finish    : ./../basic_avl_tb_top.sv(634)
#    Time: 1995670 ns  Iteration: 1  Instance: /basic_avl_tb_top
```

4.4.2.2. 9.8G CPRI PHY Simulation Dynamic Reconfiguration Design Example Components

The simulation block diagram below is generated using the following settings in the IP parameter editor:

1. **CPRI Protocol** as **DR Protocol**.
2. Under the **CPRI Protocol** tab:
    a. **9.8G CPRI** as **Select DR Design**.
    b. **Agilex TX Transceiver Signal Integrity Development Kit** as the target development kit.

Figure 48. Simulation Block Diagram for 9.8G CPRI PHY Dynamic Reconfiguration Design Example


E-Tile TX Transceiver System Integrity Development Kit
153.6 MHz phy ref clk
E-tile CPRI PHY Ethernet Intel FPGA IP Dynamic Reconfiguration Design Example
Round-trip Counter
Sources and Probes
E-Tile CPRI PHY Intel FPGA IP (9.8G)
Serial lanes
8B/10B Generator/Checker
CPRI Reconfiguration
Transceiver Reconfiguration
RS-FEC Reconfiguration
Avalon-MM Address Decoder
AIB clk
Channel PLL
Nios II System
PIO_OUT: Control TX and RX and CSR resets
PIO_IN: IP status signals check
Sampling clock
IOPLL
100 MHz
156.25 MHz (pll_refclk0)

The successful test displays the dynamic reconfiguration transition flow between various modes. Use preset HEX file provided for each design example or modify provided C code to enable specific transition simulation. For more information on HEX file, refer to Simulating the E-Tile Dynamic Reconfiguration Design Example Testbench on page 93.

To test a specific transition, reorder the dynamic reconfiguration transition flow tests in the `main.c` file and regenerate a new HEX file. Each test describes a transition from the starting rate to the destination rate.

This is the default simulation test sequence based on the provided HEX file.

1. Toggle `sl_tx_rst_n` and `sl_rx_rst_n` reset signals.
2. Dynamic reconfiguration (DR) test from 9.8G CPRI to 6G CPRI
3. DR test from 6G CPRI to 4.9G CPRI
4. DR test from 4.9G CPRI to 3G CPRI
5. DR test from 3G CPRI to 2.4G CPRI
6. DR test from 2.4G CPRI to 9.8G CPRI

Each of the dynamic reconfiguration tests follows these steps:

1. Assert `sl_tx_rst_n` and `sl_rx_rst_n` reset signals.
2. Disable SERDES. Use PMA attribute code 0x0001 in the *E-tile Transceiver PHY User Guide: PMA Attribute Codes* section.
3. Trigger PMA analog reset. For more information about register descriptions, refer to the *E-tile Transceiver PHY User Guide*.
4. Reconfigure the following registers for the Ethernet and transceiver blocks. For more information about the details of the changed register values, refer to the `c3_reconfig.c` file. For more information about the register descriptions, refer to the *E-tile Hard IP for Ethernet and CPRI PHY Intel FPGA IPs User Guide*.
5. Adjust the phase offset of a recovered clock. Use PMA attribute code 0x000E in the *E-tile Transceiver PHY User Guide: PMA Attribute Codes* section.
6. Enable SERDES. Use PMA attribute code 0x0001 in the *E-tile Transceiver PHY User Guide: PMA Attribute Codes* section.
7. Enable internal serial loopback. Use PMA attribute code 0x0008 in the *E-tile Transceiver PHY User Guide: PMA Attribute Codes* section.
8. Deassert `sl_tx_rst_n` and `sl_rx_rst_n` reset signals.
9. Wait for `PIO_OUT[3:0]` = 0x7 (`o_sl_rx_pcs_ready`, `o_sl_rx_block_lock`, and `o_ehip_ready` asserted).
10. Clear Ethernet statistic counters.
11. Enable the packet generator to start sending packets of data.
12. Check for `checker_pass` status and waiting for `PIO_OUT[3:0]` = 0xF (`checker_pass`, `o_sl_rx_pcs_ready`, `o_sl_rx_block_lock`, and `o_ehip_ready` asserted).
13. Disable the packet generator to stop sending packets.

The following sample output illustrates a successful simulation test run for a 9.8G CPRI PHY IP core variation.

```
# CPU is alive!
# End of test
# Nios has completed its transactions          1995670000
# Simulation PASSED          1995670000
# ** Note: $finish    : ./../basic_avl_tb_top.sv(634)
#    Time: 1995670 ns  Iteration: 1  Instance: /basic_avl_tb_top
```

4.4.3. Hardware Design Examples

In general, simulation design examples and hardware design examples follow the same flow except for a PMA adaptation flow.

Intel Quartus Prime Pro Edition 19.4 version supports switching between internal serial loopback without PMA adaptation, the internal serial loopback with PMA adaptation, and the external loopback with PMA adaptation. To select the loopback mode, configure TEST_MODE parameter in the flow.c.

TEST_MODE	Mode
0	Internal serial loopback without PMA adaptation
1	Internal serial loopback with PMA adaptation
Others	External serial loopback with PMA adaptation

For speed switching to 24G, 12G, 10G, and 9.8G speed modes, setting TEST_MODE to a non-zero value enables the general PMA adaptation. This PMA adaptation with zero effort configuration to shorten the link up time to less than 100 ms as per CPRI specifications requirement.

For speed switching to 6G speed modes or lower, the hardware design examples use the manual CTLE function to shorten the link up time to less than 100 ms per CPRI specification requirement. For more information about manual CTLE configuration, refer to the *E-Tile Transceiver PHY User Guide*.

4.4.3.1. CPRI PHY with RS-FEC Hardware Dynamic Reconfiguration Design Example Components

The 24G CPRI PHY hardware dynamic reconfiguration design example and 9.8G CPRI PHY hardware dynamic reconfiguration design example include the following components:

- E-tile CPRI PHY Intel FPGA IP core.
  - E-tile CPRI PHY Intel FPGA IP core - 24G CPRI
  - E-tile CPRI PHY Intel FPGA IP core - 9.8G CPRI PMA direct mode
- XGMII packet generator and checker that coordinates the programming of the IP core and packet generation.

  Note: This component is only available for 24G CPRI variant.
- 8B/10B pattern generator and checker that coordinates the programming of the IP core and packet generation.
- Avalon memory-mapped interface address decoder to decode reconfiguration address space for E-tile CPRI PHY Intel FPGA IP core, transceiver, and RS-FEC modules during reconfiguration accesses.

- Nios II System that communicates with the Nios II Software Build Tools (SBT) for Eclipse. You communicate with the client logic and E-tile Ethernet IP for Intel Agilex FPGA through the tool.
- Native PHY in PMA Direct mode that acts as a channel PLL to provide EMIB clocks (for example, 402.8 MHz and 805.6 MHz), as required by the E-tile CPRI PHY Intel FPGA IP core.
- IOPLL to provide sampling clock (for example, 250 MHz for E-tile CPRI PHY Intel FPGA IP core) and round-trip (RT) counter.
- Sources and Probes module to measure the round-trip value of the E-tile CPRI PHY Intel FPGA IP core in all supported speed modes.

The following sample outputs illustrate a successful hardware test run for a 24G CPRI PHY with RS-FEC IP core variation:

```
CPU is alive!


             Dynamic Reconfiguration Hardware Test

By default, the starting mode is CPRI24G_FEC.
      Please choose the Targeted mode available:
    1) CPRI24G
    2) CPRI12GFEC
    3) CPRI12G
    4) CPRI10GFEC
    5) CPRI10G
    6) CPRI9.8G
    7) CPRI6.0G
    8) CPRI4.9G
    9) CPRI3.0G
    a) CPRI2.4G
    9) Terminate test  -> If you terminate test halfway, you must reload
the .sof file before retrigger the hardware test.

Enter a Valid Selection:
```

4.4.4. CPRI Design Example Interface Signals

The following signals are hardware dynamic reconfiguration design example signals for the 2.4G/3G/4.9G/6G/9.8G/10G/12G/24G variants.

Table 37. CPRI Hardware Dynamic Reconfiguration Design Example Interface Signals

Signal	Direction	Comments
clk100	Input	Input clock for reconfiguration. Drive at 100 MHz. The intent is to drive this from a 100 Mhz oscillator on the board.
cpu_resetn	Input	Global reset for Nios II system.
i_clk_ref [5]	Input	156.25 MHz input clock for channel PLL.
tx_serial_data/_n	Output	Transmit serial data for channel PLL (PMA direct mode).
rx_serial_data/_n	Input	Receiver serial data for channel PLL (PMA direct mode).

continued...

[5] i_clk_ref is used to provide clock to a PMA direct module, which acts as a channel PLL to supply the required CPRI TX/RX clocks and EMIB clocks.

Signal	Direction	Comments
`i_clk_ref_cpri[1:0]`	Input	Input clock for CPRI IP core. In 24G CPRI IP: • [0]: 184.32MHz for high speed mode for 10G/25G Ethernet • [1]: 153.6MHz for PMA direct low speed mode for 9.8G CPRI, 6G CPRI, 4.9G CPRI, 3G CPRI, and 2.4G CPRI In 9.8G CPRI IP: • [0]: 153.6 MHz for direct PMA • [1]: unused
`o_tx_serial`	Output	Transmit serial data
`i_rx_serial`	Input	Receiver serial data

4.4.5. CPRI Design Example Registers

Table 38. E-tile CPRI PHY Intel FPGA IP Hardware Design Example Register Map

Word Offset	Register Category
0x000000 – 0x000FFF	CPRI PCS registers
0x010000 – 0x0107FF	RS-FEC configuration registers
0x100000 – 0x1FFFFF	Transceiver registers

4.5. 100G Ethernet Dynamic Reconfiguration Design Example

The 100G Ethernet E-Tile Dynamic Reconfiguration Design Example demonstrates a dynamic reconfiguration solution for Intel Agilex devices using the E-tile Ethernet IP for Intel Agilex FPGA core with the following variants. The 100G Ethernet E-Tile Dynamic Reconfiguration Design Example supports four PMA channels to create either a single 100G Ethernet channel, or four single 10G/25G Ethernet channels.

Table 39. List of Supported Design Example Variants for 100G Ethernet Dynamic Reconfiguration

All variants support 156.25 MHz `refclk` and optional RS-FEC. The external AIB clocking, PTP, and asynchronous clock support are not available in the current implementation.

Base Operation	Dynamic Reconfiguration Variants
100GE MAC + PCS with RS-FEC	100G MAC + PCS with RS-FEC
	100G MAC + PCS
	4x25G MAC + PCS with RS-FEC
	4x25G MAC + PCS

4.5.1. Functional Description

The 100G Ethernet E-Tile Dynamic Reconfiguration Design Example is built from the hardened E-Tile Hard IP for Ethernet IP core to enable run-time reconfiguration between different protocols, rates, and stack layers. The 100G Ethernet E-Tile Dynamic Reconfiguration Design Example supports four PMA channels to create either a single 100G Ethernet channel or four single 10G/25G Ethernet channels. The dynamic reconfiguration interface provides a selection of Ethernet modes to reconfigure your design. Once you select a mode rate, the firmware manages all register space updates to facilitate the rate change.

The dynamic reconfiguration interface enables you to reconfigure the design by selecting specific Ethernet reconfiguration modes. The firmware processes the register space modifications needed to switch between the selected modes. Alternatively, you can reconfigure the individual components by direct register programming.

The IP parameter editor allows you to select the CPU location for the 100G Ethernet E-Tile Dynamic Reconfiguration Design Example. The below figures depict the design examples block diagram with internal and external CPUs.

Figure 49. 100G Ethernet Dynamic Reconfiguration Design Example with Internal CPU Block Diagram

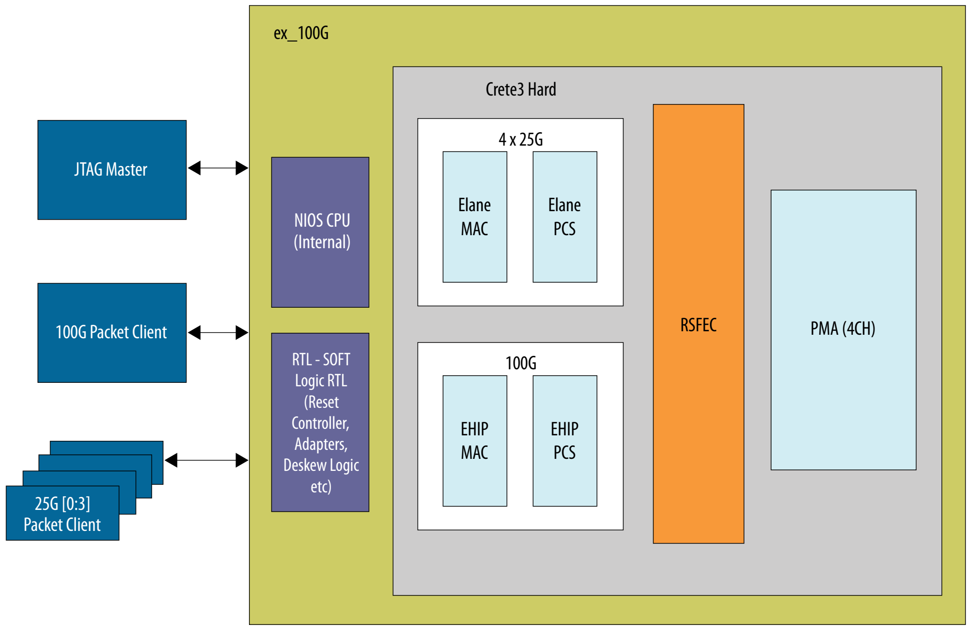
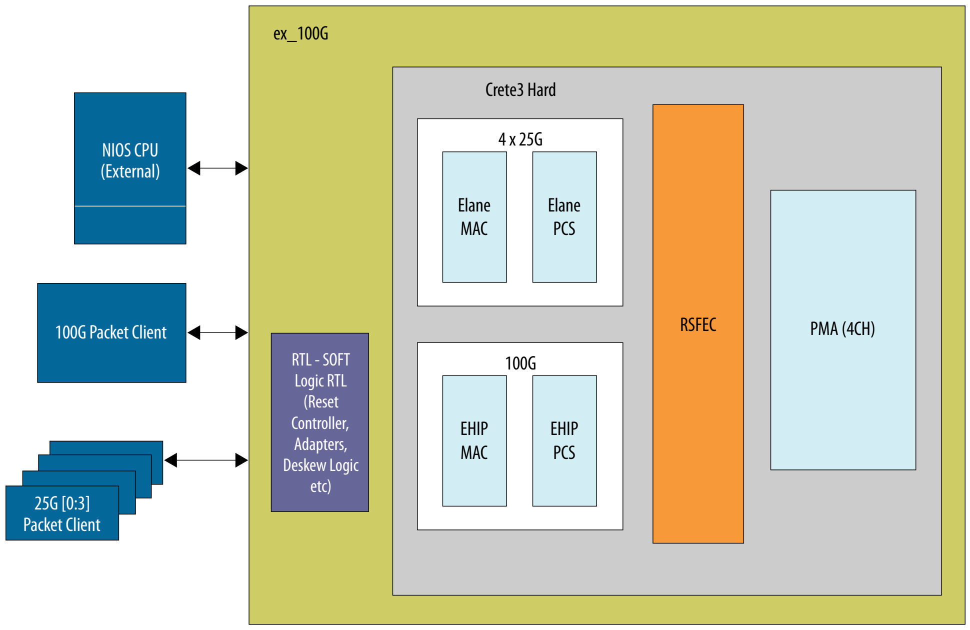


Figure 50. 100G Ethernet Dynamic Reconfiguration Design Example with External CPU Block Diagram



4.5.2. Testing the 100G Ethernet Dynamic Reconfiguration Hardware Design Example

After you compile the 100G Ethernet E-Tile Dynamic Reconfiguration Design Example and configure it on your device, you can use the procedures to program the IP core.

Table 40. 100G Ethernet Dynamic Reconfiguration Hardware Design Example Functions

Command Setting	Description
`start_random_pkt_gen_4ch`	Starts the packet generator in a random size mode for all four channel lanes. Example: `%start_random_pkt_gen_4ch`
`stop_pkt_gen_4ch`	Immediately stops the packet generator for all four channel lanes.
`chkmac_stats $ch`	Checks the mac stats counter for the specified channel. Example: • In 100GE mode: `%chkmac_stats` • In 25GE mode: `%chkmac_stats 2` for lane 2
`run_test_dr`	Switches between all available modes and performs the traffic test for each reconfiguration. In 25GE mode, performs four traffic tests, one per each lane.
`run_test_dr_sw`	Switches to a specified mode and performs the traffic test in a loopback mode.
`dr_calib_switch $mode_curr $mode_target`	Reconfigures to a different mode based on the configuration and a `$mode_target` variable. Performs the PMA adaptation for the specific mode.

continued...

Command Setting	Description
	`$mode_target` options: • 100G_fec • 100G_nofec • 4x25G_fec • 4x25G_nofec `$more_curr` variable supports all target modes. *Note:* `$mode_curr` is not a required parameter. Example: • In 100GE mode, use this command to switch to 4x25GE: `dr_calib_switch 0 "100g_fec" "4x25G_nofec"` • In 25GE mode, use this command to switch to 4x25GE: `dr_calib_switch 0 "" "4x25G_nofec"` as `$mode_curr` isn't required.
`dr_reset`	Resets all signals except the PMA and E-tile Hard IP for Ethernet CSRs.

Below tables describe `dr_reset` sequence. You need to assert the 4-bit register in a step pattern: **0x8 ➤ 0xC ➤ 0xE ➤ 0xF ➤ 0xE ➤ 0xC ➤ 0x8 ➤ 0x0**. Assume 1 ms delay between each step.

Table 41. Reset sequence assertion

This table illustrates `dr_reset[3:0]` assertion sequence.

Assertion Sequence	dr_reset[3:0]={Channel3, Channe2, Channel1, Channel0}			
	Channel 3	Channel 2	Channel 1	Channel 0 (Master Channel)
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1

Table 42. Reset sequence deassertion

This table illustrates `dr_reset[3:0]` deassertion sequence.

Assertion Sequence	dr_reset[3:0]={Channel3, Channe2, Channel1, Channel0}			
	Channel 3	Channel 2	Channel 1	Channel 0 (Master Channel)
1	1	1	1	1
2	1	1	1	0
3	1	1	0	0
4	0	0	0	0

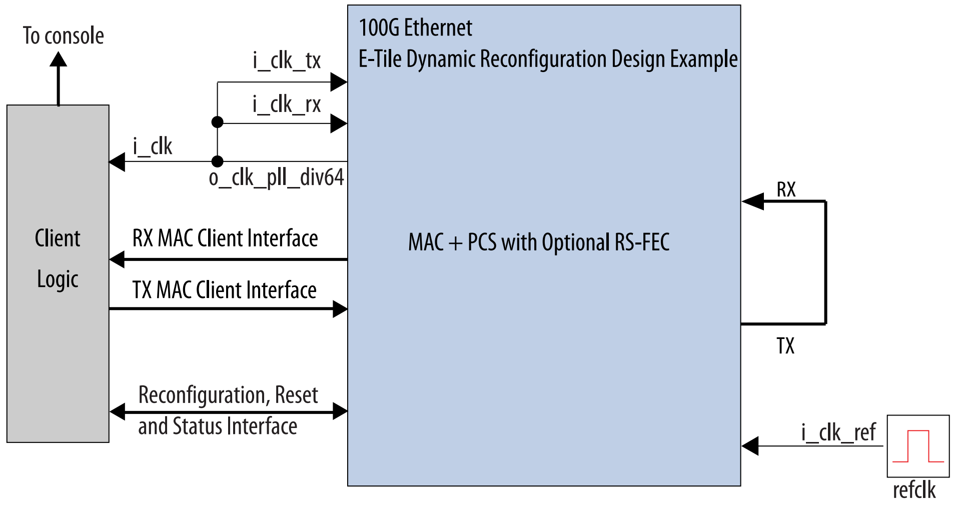
4.5.3. Simulation Design Examples

4.5.3.1. 100GE MAC+PCS with Optional RS-FEC Dynamic Reconfiguration Simulation Design Example

The simulation block diagram below is generated using the following settings in the IP parameter editor:

1. **100G Ethernet** as **DR Protocol**.
2. Under the **100G Ethernet Protocol** tab:
    a. **100G Ethernet MAC+PCS RS-FEC** as **DR Design**.
    b. **Internal** as **DR Controller Location**.
    c. **Agilex TX Transceiver Signal Integrity Development Kit** as the target development kit.

Figure 51. Simulation Block Diagram for 100GE MAC+PCS with Optional RS-FEC E-Tile Dynamic Reconfiguration Design Example



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 16,384 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 portion of successful simulation test run for a 100GE, MAC+PCS without RS-FEC IP core variation.

```
# o_tx_lanes_stable is 1 at time             348403500
# waiting for tx_dll_lock....
# TX DLL LOCK is 1 at time             407396143
# waiting for tx_transfer_ready....
# TX transfer ready is 1 at time             407716015
# waiting for rx_transfer_ready....
# RX transfer ready is 1 at time             418791583
# EHIP PLD Ready out is 1 at time             418848000
# EHIP reset out is 0 at time             418992000
# EHIP reset ack is 0 at time             419070847
# EHIP TX reset out is 0 at time             419416000
# EHIP TX reset ack is 0 at time             470466959
# waiting for EHIP Ready....
# EHIP READY is 1 at time             470536467
# EHIP RX reset out is 0 at time             472496000
# waiting for rx reset ack....
# EHIP RX reset ack is 0 at time             472509994
# Waiting for RX Block Lock
# EHIP RX Block Lock  is high at time             503401281
# Waiting for AM lock
# EHIP RX AM Lock  is high at time             503401281
# Waiting for RX alignment             503401281
# RX deskew locked             503403000
# RX lane aligmnent locked
# ** Sending Packet          1...
# ...
# ** Received Packet         10...
#
# DR -> 100G NoFEC
# ** DR STARTING
#
# ===> writedata = 00000000
# ===> writedata = 000000020
# ===> writedata = 00010000
# ===> writedata = 00000001
#
# ** RECONFIG CALLED, WAITING FOR DR
#
# ===>MATCH!     ReaddataValid = 1 Readdata = 00000001 Expected_Readdata =
00000001
# ===>AVMM READ MISMATCH!     ReaddataValid = 1 Readdata = 00000001
Expected_Readdata = 00000000
# ===>MATCH!     ReaddataValid = 1 Readdata = 00000000 Expected_Readdata =
000000000
#
# Reconfig Done
#
# ** RECONFIG DONE
#
# waiting for o_tx_lanes_stable...
# o_tx_lanes_stable is 1 at time             804564000
# waiting for tx_dll_lock....
# TX DLL LOCK is 1 at time             804564000
# waiting for tx_transfer_ready....
# TX transfer ready is 1 at time             804564000
# waiting for rx_transfer_ready....
# RX transfer ready is 1 at time             808135783
# EHIP PLD Ready out is 1 at time             808135783
# EHIP reset out is 0 at time             808135883
# EHIP reset ack is 0 at time             808135883
# EHIP TX reset out is 0 at time             808632000
# EHIP TX reset ack is 0 at time             808645131
# waiting for EHIP Ready....
# EHIP READY is 1 at time             808754358
# EHIP RX reset out is 0 at time             810672000
# waiting for rx reset ack....
# EHIP RX reset ack is 0 at time             810685684
# Waiting for RX Block Lock
# EHIP RX Block Lock  is high at time             813021645
# Waiting for AM lock
# EHIP RX AM Lock  is high at time             814548336
# Waiting for RX alignment             814548336
# RX deskew locked             815377000
# RX lane aligmnent locked
# ** Sending Packet           1...
# ...
# ** Received Packet         10...
# **
# ** Testbench complete.
# **
# *****************************************
```

Related Information

Simulating the E-tile Ethernet IP for Intel Agilex FPGA Design Example Testbench on page 10

4.5.4. 100GE DR Hardware Design Examples

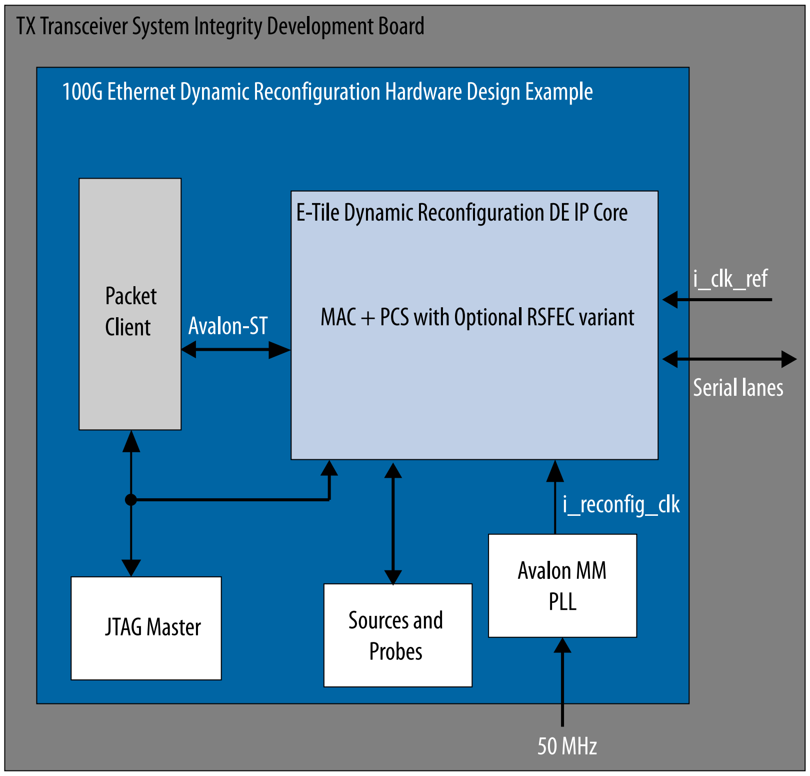
This section describes high-level flow guidelines for the E-tile reconfigurable Ethernet core.

You can follow these steps to configure the E-tile reconfigurable Ethernet IP core:

1. Create a hardware project.
   - Instantiate the E-Tile Dynamic Reconfiguration Design Example for 100G Ethernet protocol.
   - Configure the IP parameters and generate design in the Intel Quartus Prime.
2. Reconfigure the hardware project as needed during the run time.
   - Define next configuration by programing CSR registers.
   - Trigger the reconfiguration. Both variants, four 25G and one 100G Ethernet, are supported.
   - Repeat step 2.

4.5.4.1. 100GE MAC+PCS with Optional RS-FEC Dynamic Reconfiguration Hardware Design Example Components

Figure 52. 100GE MAC+PCS with Optional RS-FEC Dynamic Reconfiguration Hardware Design Example High Level Block Diagram


TX Transceiver System Integrity Development Board
100G Ethernet Dynamic Reconfiguration Hardware Design Example
Packet Client
Avalon-ST
E-Tile Dynamic Reconfiguration DE IP Core
MAC + PCS with Optional RSFEC variant
i_clk_ref
Serial lanes
i_reconfig_clk
Avalon MM PLL
JTAG Master
Sources and Probes
50 MHz

The E-Tile Dynamic Reconfiguration Design Example includes the following components:

- E-Tile Dynamic Reconfiguration Design Example core. The IP core consists of four 25G channels with optional RS-FEC or one 100G channel.
- Client logic that coordinates the programming of the IP core and packet generation.
- Avalon memory-mapped interface address decoder to decode reconfiguration address space for E-Tile Hard IP for Ethernet core and RS-FEC modules during reconfiguration accesses.
- 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 you use the `run_test` command to run the hardware test in the design examples, the script tests 100GE with RS-FEC. Use the `run_test_dr` to run the hardware test to perform all reconfigurable switches. The client logic reads and print out the MAC statistic registers when the packet transmissions are complete.

The following sample script illustrates a reconfiguration sequence:

```
source hwtest/main.tcl
set BASE_EHIP 0x400
#DR to 25GNF
# configure dr_cfg_ch_en register
reg_write $BASE_EHIP 0x13 0xf;
# configure dr_cfg_fec_en register
reg_wrtie $BASE_EHIP 0x15 0x0;
# configure dr_control and trigger reconfig registers
reg_write 0x4009 0x1;
```

The following sample output illustrates a successful hardware test run for 100GE, switching from 100G Ethernet with RS-FEC to 100G Ethernet variation:

```
% cd hwtest/altera_dr
% run_test_dr_sw "100G_rsfec" "100G_nofec"
-----------------------------------
----- Switching to 100G_nofec -----
-----------------------------------
- Checking init_adaptation status -
-----------------------------------
channel 0 init_adaptation status is 0
channel 1 init_adaptation status is 0
channel 2 init_adaptation status is 0
channel 3 init_adaptation status is 0

Running Traffic_test_100G_nofec test

 RX PHY Register Access: Checking Clock Frequencies (KHz)
    REFCLK        :2 (KHZ)
    TXCLK         :40283  (KHZ)
    RXCLK         :40285  (KHZ)
    TXRSCLK       :0  (KHZ)
    RXRSCLK       :0  (KHZ)
 RX PHY Status Polling
 Rx Frequency Lock Status     0x0000000f
 Mac Clock in OK Condition?   0x00000001
 Rx Frame Error               0x000fffff
 Rx PHY Fully Aligned?        0x00000001
 Rx AM LOCK Condition?        0x00000001
 Rx Lanes Deskewed Condition? 0x00000001
 wait for phy lock 0, locked=0x00000001
 RX PHY Register Access: Checking Clock Frequencies (KHz)
    REFCLK        :0 (KHZ)
    TXCLK         :40283  (KHZ)
    RXCLK         :40284  (KHZ)
    TXRSCLK       :0  (KHZ)
    RXRSCLK       :0  (KHZ)
 RX PHY Status Polling
 Rx Frequency Lock Status     0x0000000f
 Mac Clock in OK Condition?   0x00000001
 Rx Frame Error               0x00000000
 Rx PHY Fully Aligned?        0x00000001
 Rx AM LOCK Condition?        0x00000001
 Rx Lanes Deskewed Condition? 0x00000001

 RX PHY Register Access: Checking Clock Frequencies (KHz)
    REFCLK        :1 (KHZ)
    TXCLK         :40282  (KHZ)
    RXCLK         :40285  (KHZ)
    TXRSCLK       :0  (KHZ)
    RXRSCLK       :0  (KHZ)
 RX PHY Status Polling
 Rx Frequency Lock Status     0x0000000f
 Mac Clock in OK Condition?   0x00000001
 Rx Frame Error               0x00000000
 Rx PHY Fully Aligned?        0x00000001
 Rx AM LOCK Condition?        0x00000001
 Rx Lanes Deskewed Condition? 0x00000001

================================================================================
==========
                        STATISTICS FOR BASE 18688
(Rx)

================================================================================
==========
Fragmented Frames                : 0
Jabbered Frames                  : 0
Any Size with FCS Err Frame      : 0
Right Size with FCS Err Fra      : 0
Multicast data  Err Frames       : 0
Broadcast data Err  Frames       : 0
Unicast data Err  Frames         : 0
Multicast control  Err Frame     : 0
Broadcast control Err  Frame     : 0
Unicast control Err  Frames      : 0
Pause control Err  Frames        : 0
64 Byte Frames                   : 14620
65 - 127 Byte Frames             : 14148
128 - 255 Byte Frames            : 28658
256 - 511 Byte Frames            : 57110
512 - 1023 Byte Frames           : 115595
1024 - 1518 Byte Frames          : 111182
1519 - MAX Byte Frames           : 0
> MAX Byte Frames                : 3342259
Rx Frame Starts                  : 3683572
Multicast data  OK  Frame        : 0
Broadcast data OK   Frame        : 0
Unicast data OK   Frames         : 3675761
Multicast Control Frames         : 0
Broadcast Control Frames         : 0
Unicast Control Frames           : 0
Pause Control Frames             : 0

================================================================================
==========
                        STATISTICS FOR BASE 18432
(Tx)

================================================================================
==========
Fragmented Frames                : 0
Jabbered Frames                  : 0
Any Size with FCS Err Frame      : 0
Right Size with FCS Err Fra      : 0
Multicast data  Err Frames       : 0
Broadcast data Err  Frames       : 0
Unicast data Err  Frames         : 0
Multicast control  Err Frame     : 0
Broadcast control Err  Frame     : 0
Unicast control Err  Frames      : 0
Pause control Err  Frames        : 0
64 Byte Frames                   : 14620
65 - 127 Byte Frames             : 14148
128 - 255 Byte Frames            : 28658
256 - 511 Byte Frames            : 57110
512 - 1023 Byte Frames           : 115595
1024 - 1518 Byte Frames          : 111182
1519 - MAX Byte Frames           : 0
> MAX Byte Frames                : 3342259
Tx Frame Starts                  : 3683572
Multicast data  OK  Frame        : 0
Broadcast data OK   Frame        : 0
Unicast data OK   Frames         : 3675761
Multicast Control Frames         : 0
Broadcast Control Frames         : 0
```

```
Unicast Control Frames          : 0
Pause Control Frames            : 0
Traffic_test_100G_nofec: Pass
```

Related Information

- Compiling and Configuring the Design Example in Hardware on page 12
- Testing the E-tile Ethernet IP for Intel Agilex FPGA Hardware Design Example on page 13

4.5.5. 100G Ethernet Dynamic Reconfiguration Design Example Interface Signals

The following signals are hardware dynamic reconfiguration design example signals for 100G Ethernet Dynamic Reconfiguration variants.

Table 43. 100G Ethernet Dynamic Reconfiguration Design Example Hardware Interface Signals

Signal	Direction	Comments
clk100	Input	Input clock for reconfiguration. Drive at 100 MHz. The intent is to drive this from a 100 Mhz oscillator on the board.
cpu_resetn	Input	Input reset for the dynamic reconfiguration controller.
i_csr_rst_n		Resets the entire IP core.
refclk	Input	156.25 MHz clock for the 100G Ethernet IP core..
o_tx_serial	Output	Transmit serial data.
i_rx_serial	Input	Receiver serial data.

4.5.6. 100G Ethernet Dynamic Reconfiguration Examples Registers

Table 44. E-Tile Dynamic Reconfiguration Design Example Hardware Design Examples Register Map for 100G Ethernet Protocol

Lists the memory mapped register ranges for all 100G Ethernet dynamic reconfiguration hardware design example variants. You access these registers with the reg_read and reg_write functions in the System Console.

Channel Number	Word Offset	Register Type
0	0x100000	Transceiver registers
	0x000000	10G/25G Ethernet registers
	0x010000	RS-FEC configuration registers
	0x004000	100G Ethernet registers
	0x005000	100G Packet Client and Packet Generator registers
	0x001000	10G/25G Packet client and Packet Generator registers
1	0x300000	Transceiver registers
	0x200000	10G/25G Ethernet registers

continued...

Send Feedback

Channel Number	Word Offset	Register Type
	0x201000	10G/25G Packet Client and Packet Generator registers
2	0x500000	Transceiver registers
	0x400000	10G/25G Ethernet registers
	0x401000	10G/25G Packet Client and Packet Generator registers
3	0x700000	Transceiver registers
	0x600000	10G/25G Ethernet registers
	0x601000	10G/25G Packet Client and Packet Generator registers

Table 45. 100G Ethernet Dynamic Reconfiguration Registers

For a specific address, use the 100G Ethernet registers word offset.

Addr	Name	Bit	Description	HW Reset Value	Access
0x00	`dr_status`	[0]	**Reconfiguration controller status** Indicates the reconfiguration controller is busy. Don't modify the configuration while busy.	0x0	RW
0x09	`dr_control`	[0]	**Reconfiguration process control** Set to 1 to trigger the reconfiguration process.	0x0	RW
0x0E	`dr_reset`	[3:0]	**Reset sequence** Reset all signals except the PMA and E-Tile Hard IP for Ethernet CSRs.	0x0	RW
0x13	`cdr_cfg_ch_en`	[16,3:0]	**Channel enable** • [16]: Enables lane 0 in 100G Ethernet variant • [3:0]: Enables lane 3 ~ lane 0 in 25G Ethernet variant	0x0	RW
0x14	`dr_cfg_ch_mode`	[26:24, 18:16, 10:8, 2:0]	**Channel mode** MAC+PCS: 0x5 • [26:24]: Selects channel 3 • [18:16]: Selects channel 2 • [10:8]: Selects channel 1 • [2:0]: Selects channel 0	0x0	RW
0x15	`dr_cfg_fec_en`	[3:0]	Enable RS-FEC on Nth channel	0x0	RW
0x16	`dr_cfg_ch_rate`	[3:0]	**Ethernet channel rate** • [0]: Selects 25G/100G	0x0	RW

4.6. Document Revision History for the E-tile Dynamic Reconfiguration Design Example

Document Version	Intel Quartus Prime Version	IP Version	Changes
2019.12.30	19.4	19.4.0	• Added new PMA adaptation flow for 10G/25G variant. • Added simulation, compilation, and hardware support for Intel Agilex dynamic reconfiguration design examples for CPRI protocols: — 24G CPRI with RS-FEC — 9.8G CPRI with RS-FEC • Added simulation, compilation, and hardware support for Intel Agilex dynamic reconfiguration design examples for 100G Ethernet protocol
2019.10.18	19.3	19.3.0	Initial release.

5. E-tile Hard IP Intel Agilex Design Examples User Guide Archives

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

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

Intel Quartus Prime Version	User Guide
19.3	E-tile Hard IP Intel Agilex Design Examples User Guide