E-tile Hard IP Intel® Stratix® 10 Design Examples User Guide

Ethernet, CPRI PHY, and Dynamic Reconfiguration

Updated for Intel® Quartus® Prime Design Suite: 20.1
# Contents

1. **About E-tile Hard IP Design Examples** ........................................................................................................ 4

2. **E-Tile Hard IP for Ethernet Intel FPGA IP Design Example** ........................................................................ 5
   2.1. E-Tile Hard IP for Ethernet Intel FPGA IP Quick Start Guide ............................................................... 5
   2.1.1. Directory Structure .............................................................................................................................. 6
   2.1.2. Generating the Design ......................................................................................................................... 8
   2.1.3. Simulating the E-Tile Hard IP for Ethernet Intel FPGA IP Design Example Testbench .................................................. 11
   2.1.4. Compiling the Compilation-Only Project ............................................................................................... 11
   2.1.5. Compiling and Configuring the Design Example in Hardware .............................................................. 12
   2.1.6. Testing the E-Tile Hard IP for Ethernet Intel FPGA IP Hardware Design Example ........................................ 13
   2.2. 10GE/25GE with Optional RS-FEC Design Examples .................................................................... 18
   2.2.1. Simulation Design Examples .............................................................................................................. 19
   2.2.2. Hardware Design Examples .............................................................................................................. 27
   2.2.3. 10GE/25GE Design Example Interface Signals .............................................................................. 37
   2.2.4. 10GE/25GE Design Examples Registers ............................................................................................. 38
   2.3. 100GE with Optional RS-FEC Design Example .................................................................................. 41
   2.3.1. Simulation Design Examples .............................................................................................................. 41
   2.3.2. Hardware Design Examples .............................................................................................................. 53
   2.3.3. 100GE MAC+PCS with Optional RS-FEC Design Example Interface Signals .................................. 65
   2.3.4. 100GE PCS with Optional RS-FEC Design Example Interface Signals ............................................. 66
   2.3.5. 100GE MAC+PCS with Optional RS-FEC Design Example Registers .............................................. 67
   2.3.6. 100GE PCS with Optional RS-FEC Design Example Registers ....................................................... 68
   2.4. Document Revision History for the E-Tile Hard IP for Ethernet Intel FPGA IP Design Example .............. 70

3. **E-tile CPRI PHY Intel FPGA IP Design Example** .................................................................................. 74
   3.1. E-tile CPRI PHY Intel FPGA IP Quick Start Guide ................................................................................ 74
   3.1.1. Hardware and Software Requirements .............................................................................................. 74
   3.1.2. Generating the Design ......................................................................................................................... 75
   3.1.3. Directory Structure .............................................................................................................................. 76
   3.1.4. Simulating the Design Example Testbench ........................................................................................... 78
   3.1.5. Compiling the Compilation-Only Project ............................................................................................... 80
   3.1.6. Compiling and Configuring the Design Example in Hardware .............................................................. 81
   3.1.7. Testing the E-tile CPRI PHY Intel FPGA IP Hardware Design Example .............................................. 82
   3.2. E-tile CPRI PHY Design Example Description .................................................................................. 83
   3.2.1. Features ............................................................................................................................................ 83
   3.2.2. Simulation Design Example .............................................................................................................. 83
   3.2.3. Hardware Design Example .............................................................................................................. 85
   3.2.4. Interface Signals ................................................................................................................................. 86
   3.2.5. Design Example Register Map for Reconfiguration ........................................................................... 86
   3.3. Document Revision History for the E-tile CPRI PHY Intel FPGA IP Design Example ........................ 87

4. **E-Tile Dynamic Reconfiguration Design Example** .................................................................................. 88
   4.1. Quick Start Guide .................................................................................................................................. 88
   4.1.1. Directory Structure .............................................................................................................................. 89
   4.1.2. Generating the Design ......................................................................................................................... 92
4.1.3. Simulating the E-Tile Dynamic Reconfiguration Design Example Testbench........96
4.1.4. Compiling and Configuring the Design Example in Hardware..................100
4.1.5. Testing the E-tile Dynamic Reconfiguration Hardware Design Example......100
4.2. 10G/25G Ethernet Dynamic Reconfiguration Design Examples....................104
  4.2.1. Functional Description.................................................................104
  4.2.2. Simulation Design Examples..........................................................105
  4.2.3. Hardware Design Examples............................................................110
  4.2.4. 10GE/25GE Design Example Interface Signals..................................112
  4.2.5. 10GE/25GE Design Examples Registers........................................112
4.3. CPRI Dynamic Reconfiguration Design Examples.......................................115
  4.3.1. Functional Description.................................................................115
  4.3.2. Simulation Design Examples..........................................................117
  4.3.3. Hardware Design Examples............................................................121
  4.3.4. CPRI Design Example Interface Signals........................................123
  4.3.5. CPRI Design Example Registers...................................................124
4.4. 25G Ethernet to CPRI Dynamic Reconfiguration Design Example ................125
  4.4.1. Functional Description.................................................................125
  4.4.2. Simulation Design Examples..........................................................126
  4.4.3. Hardware Design Examples............................................................128
  4.4.4. 25G Ethernet to CPRI Design Example Interface Signals....................130
  4.4.5. 25G Ethernet to CPRI Design Examples Registers............................130
4.5. 100G Ethernet Dynamic Reconfiguration Design Example............................134
  4.5.1. Functional Description.................................................................134
  4.5.2. Testing the 100G Ethernet Dynamic Reconfiguration Hardware Design
        Example............................................................................................136
  4.5.3. Simulation Design Examples..........................................................137
  4.5.4. 100GE DR Hardware Design Examples...........................................140
  4.5.5. 100G Ethernet Dynamic Reconfiguration Design Example Interface Signals...
        144
  4.5.6. 100G Ethernet Dynamic Reconfiguration Examples Registers................144
4.6. Document Revision History for the E-tile Dynamic Reconfiguration Design Example...146

1. About E-tile Hard IP Design Examples

This document consists of the following design examples:

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

Related Information

- E-Tile Hard IP for Ethernet Intel FPGA IP Design Example on page 5
- E-tile CPRI PHY Intel FPGA IP Design Example on page 74
- E-Tile Dynamic Reconfiguration Design Example on page 88
2. E-Tile Hard IP for Ethernet Intel FPGA IP Design Example

2.1. E-Tile Hard IP for Ethernet Intel FPGA IP Quick Start Guide

The E-tile Hard IP for Ethernet Intel FPGA IP core for Intel Stratix® 10 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 Stratix 10 TX Transceiver Signal Integrity Development Kit. Intel provides a compilation-only example project that you can use to quickly estimate IP core area and timing.

Table 1. List of Supported Design Example Variants

<table>
<thead>
<tr>
<th>Data Rate</th>
<th>Variant</th>
<th>Simulation</th>
<th>Compilation-Only Project</th>
<th>Hardware Design Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>10GE</td>
<td>Single or multi channels Media Access Controller (MAC) + Physical Coding Sublayer (PCS) with optional 1588 Precision Time Protocol (PTP)</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td></td>
<td>Single channel PCS</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td></td>
<td>Single channel Optical Transport Network (OTN)</td>
<td>√</td>
<td>√</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Single channel Flexible Ethernet (FlexE)</td>
<td>√</td>
<td>√</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Single or multi channels custom PCS</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>25GE</td>
<td>Single or multi channels MAC + PCS with optional RS-FEC and optional PTP • Asynchronous Adapter clock</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td></td>
<td>Single channel PCS with optional RS-FEC</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td></td>
<td>Single channel OTN with optional RS-FEC</td>
<td>√</td>
<td>√</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Single channel FlexE with optional RS-FEC</td>
<td>√</td>
<td>√</td>
<td>X</td>
</tr>
</tbody>
</table>

*Other names and brands may be claimed as the property of others.*
<table>
<thead>
<tr>
<th>Data Rate</th>
<th>Variant</th>
<th>Simulation</th>
<th>Compilation-Only Project</th>
<th>Hardware Design Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single or multi channels custom PCS with optional RS-FEC</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>100GE</td>
<td>MAC+ PCS with optional:</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td></td>
<td>• (528,514) RS-FEC</td>
<td>√</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• PTP</td>
<td>√</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>MAC+PCS with (544, 514) RS-FEC</td>
<td>√</td>
<td></td>
<td>√</td>
</tr>
<tr>
<td></td>
<td>PCS with optional (528,514) or (544, 514) RS-FEC</td>
<td>√</td>
<td></td>
<td>√</td>
</tr>
<tr>
<td></td>
<td>OTN with optional (528,514) or (544, 514) RS-FEC</td>
<td>√</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>FlexE with optional (528,514) or (544, 514) RS-FEC</td>
<td>√</td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

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

**Figure 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 Hard IP for Ethernet Intel FPGA IP design example file directories contain the following generated files for the design examples.
Figure 2.  **E-Tile Hard IP for Ethernet Intel FPGA IP 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.

Figure 3.  **E-Tile Hard IP for Ethernet Intel FPGA IP 100GE with Optional RS-FEC Design Example Directory Structure**
Table 2. E-Tile Hard IP for Ethernet Intel FPGA IP Core Testbench File Descriptions

<table>
<thead>
<tr>
<th>File Names</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;design_example_dir&gt;/example_testbench/basic_avl_tb_top.sv</code></td>
<td>Top-level testbench file. The testbench instantiates the DUT and runs</td>
</tr>
<tr>
<td></td>
<td>Verilog HDL tasks to generate and accept packets.</td>
</tr>
<tr>
<td>Testbench Scripts</td>
<td></td>
</tr>
<tr>
<td><code>&lt;design_example_dir&gt;/example_testbench/run_vsim.do</code></td>
<td>The Mentor Graphics ModelSim* script to run the testbench.</td>
</tr>
<tr>
<td><code>&lt;design_example_dir&gt;/example_testbench/run_vcs.sh</code></td>
<td>The Synopsys VCS* script to run the testbench.</td>
</tr>
<tr>
<td><code>&lt;design_example_dir&gt;/example_testbench/run_vcsmx.sh</code></td>
<td>The Synopsys VCS MX* script (combined Verilog HDL and System Verilog with VHDL) to run the testbench.</td>
</tr>
<tr>
<td><code>&lt;design_example_dir&gt;/example_testbench/run_ncsim.sh</code></td>
<td>The Cadence NCsim* script to run the testbench.</td>
</tr>
<tr>
<td><code>&lt;design_example_dir&gt;/example_testbench/run_xcelium.sh</code></td>
<td>The Xcelium* script to run the testbench.</td>
</tr>
</tbody>
</table>

Table 3. Intel Stratix 10 IP Core Hardware Design Example File Descriptions

<table>
<thead>
<tr>
<th>File Names</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;design_example_dir&gt;/hardware_test_design/alt_ehipc3_hw.qpf</code></td>
<td>Intel Quartus® Prime project file.</td>
</tr>
<tr>
<td><code>&lt;design_example_dir&gt;/hardware_test_design/alt_ehipc3_hw.qsf</code></td>
<td>Intel Quartus Prime project settings file.</td>
</tr>
<tr>
<td><code>&lt;design_example_dir&gt;/hardware_test_design/alt_ehipc3_hw.sdc</code></td>
<td>Synopsys Design Constraints files. You can copy and modify these files for your own Intel Stratix 10 design.</td>
</tr>
<tr>
<td><code>&lt;design_example_dir&gt;/hardware_test_design/alt_ehipc3_hw.v</code></td>
<td>Top-level Verilog HDL design example file.</td>
</tr>
<tr>
<td><code>&lt;design_example_dir&gt;/hardware_test_design/common/</code></td>
<td>Hardware design example support files.</td>
</tr>
<tr>
<td><code>hwtest_sl/main_script.tcl (10GE/25GE)</code></td>
<td>Main file for accessing System Console.</td>
</tr>
<tr>
<td><code>hwtest/main.tcl (100GE)</code></td>
<td></td>
</tr>
</tbody>
</table>

2.1.2. Generating the Design

Figure 4. Procedure
If you do not already have an Intel Quartus Prime Pro Edition project in which to integrate your E-Tile Hard IP for Ethernet Intel FPGA IP core, you must create one.

1. In the Intel Quartus Prime Pro Edition software, click File ➤ New Project Wizard to create a new Quartus Prime project, or File ➤ Open Project to open an existing Intel Quartus Prime project. The wizard prompts you to specify a device.

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

3. Click Finish.

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

1. In the IP Catalog, locate and select E-Tile Hard IP for Ethernet 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, 100GE, or 10GE/25GE tabs, specify the parameters for your IP core variation.

5. The hardware design examples provide enable internal serial loopback by default.

6. 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.

7. 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.

8. 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.

9. Under **Target Development Kit**, select the **Stratix 10 TX Transceiver Signal Integrity Development Kit-1ST280EY2F55E2VGSI**, **Stratix 10 TX Transceiver Signal Integrity Development Kit-1ST280EY2F55E2VG** or select **None**. If you select a **specific Development Kit** as the **Target Development Kit**, the design example is generated based on a specific device and it overwrites the device you selected in your project file. If you select **None** as the **Target Development Kit**, ensure the selected device is your targeted device and adjust the pins assignment in the .qsf file. By default, .qsf file is generated based on the device used in the development kit.

10. Click the **Generate Example Design** button. The **Select Example Design Directory** window appears.

11. If you want to modify the design example directory path or name from the defaults displayed (**alt_ehipc3_0_example_design**), browse to the new path and type the new design example directory name (**<design_example_dir>**).

**Related Information**

- **E-Tile Hard IP for Ethernet Intel FPGA IP Core Parameters**
  Provides more information about customizing your IP core.

  More information parameters in **PMA Adaptation** tab.

- **Intel Stratix 10 TX Signal Integrity Development Kit Webpage**
2.1.3. Simulating the E-Tile Hard IP for Ethernet Intel FPGA IP Design Example Testbench

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*

<table>
<thead>
<tr>
<th>Simulator</th>
<th>Instructions</th>
</tr>
</thead>
</table>
| Mentor Graphics ModelSim* | In the command line, type `vlsim -do run_vsim.do`  
If you prefer to simulate without bringing up the ModelSim GUI, type `vlsim -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`  
*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`. |
| 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 Stratix 10 device:
   a. Connect Stratix 10 E-Tile 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 Stratix 10 E-Tile 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 Stratix 10 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 Hard IP for Ethernet Intel FPGA IP Hardware Design Example

After you compile the E-Tile Hard IP for Ethernet Intel FPGA IP core design example and configure it on your Intel Stratix 10 device, you can use the System Console to program the IP core and its embedded Native PHY IP core registers.

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 Stratix 10 device, in the Intel Quartus Prime Pro Edition software, on the Tools menu, click In-System Sources and Probes Editor.

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:

<table>
<thead>
<tr>
<th>Source</th>
<th>Signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>source[3]</td>
<td>sl_csr_rst_n (active low)</td>
</tr>
<tr>
<td>source[2]</td>
<td>sl_tx_rst_n (active low)</td>
</tr>
<tr>
<td>source[1]</td>
<td>sl_rx_rst_n (active low)</td>
</tr>
<tr>
<td>source[0]</td>
<td>i_reconfig_reset (active high)</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Command Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>totalChannel</td>
<td>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 <code>set totalChannel 2</code> to change the number of channels to 2. Note: E-Tile Hard IP for Ethernet Intel FPGA IP does not support multichannel PCS variation.</td>
</tr>
<tr>
<td>jtag_port_id</td>
<td>Set this value to the JTAG port ID that is connected to the development kit. Example, in the system console type <code>set jtag_port_id 0</code> to change the JTAG ID to 0.</td>
</tr>
<tr>
<td>enableILB</td>
<td>Set this to 1 to enable Internal Serial Loopback. The default value is 1. Example, in the system console, type <code>set enableILB 0</code> to disable Internal Serial Loopback.</td>
</tr>
<tr>
<td>enablePTP</td>
<td>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 <code>set enablePTP 1</code> to enable PTP.</td>
</tr>
<tr>
<td>speed</td>
<td>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 <code>set speed 25G_fec</code> to set the data rate to 25G with RS-FEC enabled.</td>
</tr>
</tbody>
</table>
**Command Setting**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMAadaptation</td>
<td>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.</td>
</tr>
<tr>
<td>PMAConfig</td>
<td>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.</td>
</tr>
</tbody>
</table>

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:

```plaintext
% 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 Stratix 10 device, in the Intel Quartus Prime Pro Edition software, on the **Tools** menu, click **System Debugging Tools** ➤ **System Console**.
2. In the Tcl Console pane, type `cd hwtest` to change directory to `<design_example_dir>/hardware_test_design/hwtest`.
3. Type source main.tcl to open a connection to the JTAG master.
You can 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 Stratix 10 E-Tile 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 Stratix 10 E-Tile 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 Stratix 10 E-Tile 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 Stratix 10 E-Tile 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 Stratix 10 device, in the Intel Quartus Prime Pro Edition software, on the Tools menu, click System Debugging Tools ➤ System Console.

2. In the Tcl Console pane, type cd hwtest to change directory to <design_example_dir>/hardware_test_design/hwtest.

3. Type source main.tcl to open a connection to the JTAG master.

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 iteration> 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 Stratix 10 devices using the E-Tile Hard IP for Ethernet Intel FPGA IP core with the following variants:

Table 5. Supported Design Example Variants for 10GE/25GE
All variant supports up to 4 channels.

<table>
<thead>
<tr>
<th>Variant</th>
<th>Intel Stratix 10 Design Example Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAC+PCS with Optional RS-FEC(3)</td>
<td>Simulation and compilation-only project, and hardware design example</td>
</tr>
<tr>
<td>MAC+PCS with Optional RS-FEC and PTP(3)</td>
<td>Simulation and compilation-only project, and hardware design example</td>
</tr>
<tr>
<td>PCS Only with Optional RS-FEC(3)</td>
<td>Simulation and compilation-only project, and hardware design example</td>
</tr>
<tr>
<td>OTN with Optional RS-FEC(3)</td>
<td>Simulation and compilation-only project</td>
</tr>
<tr>
<td>FlexE with Optional RS-FEC(3)</td>
<td>Simulation and compilation-only project</td>
</tr>
<tr>
<td>Custom PCS with Optional RS-FEC(3)</td>
<td>Simulation and compilation-only project, and hardware design example</td>
</tr>
</tbody>
</table>

(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.

![Simulation Block Diagram for Non-PTP E-Tile Hard IP for Ethernet Intel FPGA IP Core](image)

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.

```plaintext
# 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 alignment 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 = 00836, ReadData = 0000000a
# ** Address offset = 00936, ReadData = 0000000a
# ** Address offset = 00804, ReadData = 00000000
# ** Address offset = 00904, ReadData = 00000000
# ** Address offset = 00322, ReadData = 00000000
# ** Address offset = 00084, WriteData = ffffffff
# ** Address offset = 00084, ReadData = 000000ff
# ** Address offset = 00084, WriteData = 00000000
# ** Address offset = 00230, WriteData = ffffffff
# ** Address offset = 00230, WriteData = 00000007b
# **
# ** AVMM access CSR registers read/write check for ETH RSFEC
# ** Address offset = 10000, ReadData = 00000001
# ** Address offset = 10000, WriteData = ffffffff
# ** Address offset = 10000, ReadData = 0000000f
# ** 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 Hard IP for Ethernet Intel FPGA IP Design Example Testbench on page 11

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 Hard IP for Ethernet Intel FPGA IP 10GE/25GE with Optional RS-FEC and PTP Design Example

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 alignment 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)

---

- During the test, the following steps were executed:
  - Channel 3 - Configure TX extra latency
  - Channel 3 - Configure RX extra latency
  - Channel 3 - TX enabled
  - Channel 3 - Sending Packet 1 to 10
  - Channel 3 - Received Packet 1 to 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 Hard IP for Ethernet Intel FPGA IP Design Example Testbench on page 11

---

### 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.
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.

```bash
# 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
```
## 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 Hard IP for Ethernet Intel FPGA IP 10GE/25GE Custom PCS with Optional RS-FEC Design Example**

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 alignment 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 – TX enabled locked
Channel 1 – RX lane alignment locked
Channel 1 – TX enabled
*** Channel 1 – Sending packets ***
Start frame detected, byteslip 0, time 406135119
** 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 – TX enabled locked
Channel 2 – RX lane alignment 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 – RX lane alignment 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
2.2.2. Hardware Design Examples

Hardware Design examples are supported for Intel Stratix 10 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

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

- E-Tile Hard IP for Ethernet Intel FPGA IP core.
- Client logic that coordinates the programming of the IP core and packet generation.
- 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
Info: Successfully Write EHIPLANE Channel 0, User Register

   phy_ehip_csr_soft_reset               , offset = 0x310, data = 0x0
   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

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_s1/c3_elane_ptp_traffic_basic_test.log.

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 = 0x0
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 = 00000000003331b311e971d6, RX Timestamp = 00000000003331b311e9df10, 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

- Intel Stratix 10 TX Signal Integrity Development Kit Webpage
- Compiling and Configuring the Design Example in Hardware on page 12
- Testing the E-Tile Hard IP for Ethernet Intel FPGA IP Hardware Design Example on page 13
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

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

- E-Tile Hard IP for Ethernet Intel FPGA IP core.
- Client logic that coordinates the programming of the IP core and packet generation.
- 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

- Intel Stratix 10 TX Signal Integrity Development Kit Webpage
- Compiling and Configuring the Design Example in Hardware on page 12
• Testing the E-Tile Hard IP for Ethernet Intel FPGA IP 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

The E-Tile Hard IP for Ethernet Intel FPGA IP hardware design example includes the following components:
• E-Tile Hard IP for Ethernet Intel FPGA IP core.
• Client logic that coordinates the programming of the IP core and packet generation.
• 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  

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  

Successfully Write EHIPLANE Channel 1, User Register phy_ehip_csr_soft_reset  

Successfully Write EHIPLANE Channel 1, User Register phy_ehip_csr_soft_reset  

Successfully Write EHIPLANE Channel 1, User Register phy_ehip_csr_soft_reset  

Successfully Read EHIPLANE Channel 1, User Register phy_ehip_csr_soft_reset  

Successfully Write EHIPLANE Channel 1, User Register phy_ehip_csr_soft_reset  

Successfully Write EHIPLANE Channel 1, User Register phy_ehip_csr_soft_reset  

Successfully Write EHIPLANE Channel 1, User Register phy_ehip_csr_soft_reset  

Successfully Read EHIPLANE Channel 1, User Register phy_ehip_csr_soft_reset  

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
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
- Intel Stratix 10 TX Signal Integrity Development Kit Webpage
- Compiling and Configuring the Design Example in Hardware on page 12
- Testing the E-Tile Hard IP for Ethernet Intel FPGA IP 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

<table>
<thead>
<tr>
<th>Signal</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clk100</td>
<td>Input</td>
<td>Drive at 100 to 161.13 MHz. Input clock for CSR access on all the AVMM interfaces.</td>
</tr>
<tr>
<td>i_clk_ref</td>
<td>Input</td>
<td>Drive at 322.265625 MHz.</td>
</tr>
<tr>
<td>cpu_resetn</td>
<td>Input</td>
<td>Resets the IP core. Active low. Drives the global hard reset csr_reset_n to the IP core.</td>
</tr>
<tr>
<td>o_tx_serial[number of channels-1:0]</td>
<td>Output</td>
<td>Transceiver PHY output serial data.</td>
</tr>
<tr>
<td>i_rx_serial[number of channels-1:0]</td>
<td>Input</td>
<td>Transceiver PHY input serial data.</td>
</tr>
</tbody>
</table>
2.2.4. 10GE/25GE Design Examples Registers

Table 7. E-Tile Hard IP for Ethernet Intel FPGA IP 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.

<table>
<thead>
<tr>
<th>Channel Number</th>
<th>Word Offset</th>
<th>Register Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0x000000</td>
<td>KR4 registers</td>
</tr>
<tr>
<td></td>
<td>0x000300</td>
<td>RX PCS registers</td>
</tr>
<tr>
<td></td>
<td>0x000400</td>
<td>TX MAC registers</td>
</tr>
<tr>
<td></td>
<td>0x000500</td>
<td>RX MAC registers</td>
</tr>
<tr>
<td></td>
<td>0x000800</td>
<td>TX Statistics Counter registers</td>
</tr>
<tr>
<td></td>
<td>0x000900</td>
<td>RX Statistics Counter registers</td>
</tr>
<tr>
<td></td>
<td>0x001000</td>
<td>Packet Client and Packet Generator registers</td>
</tr>
<tr>
<td></td>
<td>0x002000</td>
<td>PTP monitoring registers</td>
</tr>
<tr>
<td></td>
<td>0x010000</td>
<td>RS-FEC configuration registers</td>
</tr>
<tr>
<td></td>
<td>0x010000</td>
<td>Transceiver registers</td>
</tr>
<tr>
<td>1</td>
<td>0x200000</td>
<td>KR4 registers</td>
</tr>
<tr>
<td></td>
<td>0x200300</td>
<td>RX PCS registers</td>
</tr>
<tr>
<td></td>
<td>0x200400</td>
<td>TX MAC registers</td>
</tr>
<tr>
<td></td>
<td>0x200500</td>
<td>RX MAC registers</td>
</tr>
<tr>
<td></td>
<td>0x200800</td>
<td>TX Statistics Counter registers</td>
</tr>
<tr>
<td></td>
<td>0x200900</td>
<td>RX Statistics Counter registers</td>
</tr>
<tr>
<td></td>
<td>0x201000</td>
<td>Packet Client registers</td>
</tr>
<tr>
<td></td>
<td>0x202000</td>
<td>PTP monitoring registers</td>
</tr>
<tr>
<td></td>
<td>0x210000</td>
<td>RS-FEC configuration registers</td>
</tr>
<tr>
<td></td>
<td>0x300000</td>
<td>Transceiver registers</td>
</tr>
<tr>
<td>2</td>
<td>0x400000</td>
<td>KR4 registers</td>
</tr>
<tr>
<td></td>
<td>0x400300</td>
<td>RX PCS registers</td>
</tr>
<tr>
<td></td>
<td>0x400400</td>
<td>TX MAC registers</td>
</tr>
<tr>
<td></td>
<td>0x400500</td>
<td>RX MAC registers</td>
</tr>
<tr>
<td></td>
<td>0x400800</td>
<td>TX Statistics Counter registers</td>
</tr>
<tr>
<td></td>
<td>0x400900</td>
<td>RX Statistics Counter registers</td>
</tr>
<tr>
<td></td>
<td>0x401000</td>
<td>Packet Client registers</td>
</tr>
<tr>
<td></td>
<td>0x402000</td>
<td>PTP monitoring registers</td>
</tr>
</tbody>
</table>
Table 8. Packet Client Registers

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

<table>
<thead>
<tr>
<th>Addr</th>
<th>Name</th>
<th>Bit</th>
<th>Description</th>
<th>HW Reset Value</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x1000</td>
<td>PKT_CL_SCRATCH_TCH</td>
<td>[31:0]</td>
<td>Scratch register available for testing.</td>
<td></td>
<td>RW</td>
</tr>
<tr>
<td>0x1001</td>
<td>PKT_CL_CLNT</td>
<td>[31:0]</td>
<td>Four characters of IP block identification string “CLNT”</td>
<td></td>
<td>RO</td>
</tr>
<tr>
<td>0x1008</td>
<td>Packet Size Configure</td>
<td>[29:0]</td>
<td>Specifies the transmit packet size in bytes. These bits have dependencies</td>
<td>0x25800040</td>
<td>RW</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>to PKT_GEN_TX_CTRL register.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Bit [29:16]: Specify the upper limit of the packet size in bytes.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Bit [13:0]:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>— For fixed mode, these bits specify the transmit packet size in bytes.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>— For incremental mode, these bits specify the incremental bytes for a packet.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1009</td>
<td>Packet Number Control</td>
<td>[31:0]</td>
<td>Specifies the number of packets to transmit from the packet generator.</td>
<td>0xA</td>
<td>RW</td>
</tr>
<tr>
<td>0x1010</td>
<td>PKT_GEN_TX_CTRL</td>
<td>[7:0]</td>
<td>• Bit [0]: Reserved.</td>
<td>0x6</td>
<td>RW</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Bit [1]: Packet generator disable bit. Set this bit to the value of 1 to</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>turn off the packet generator, and reset it to the value of 0 to turn</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>on the packet generator.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Bit [2]: Reserved.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Bit [3]: Has the value of 1 if the IP core is in MAC loopback mode; has</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>the value of 0 if the packet client uses the packet generator.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

continued...
### Table 9. MII Packet Generator Registers

<table>
<thead>
<tr>
<th>Addr</th>
<th>Name</th>
<th>Bit</th>
<th>Description</th>
<th>HW Reset Value</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x1011</td>
<td>Destination address lower 32 bits</td>
<td>[31:0]</td>
<td>Destination address (lower 32 bits)</td>
<td>0x56780ADD</td>
<td>RW</td>
</tr>
<tr>
<td>0x1012</td>
<td>Destination address upper 16 bits</td>
<td>[15:0]</td>
<td>Destination address (upper 16 bits)</td>
<td>0x1234</td>
<td>RW</td>
</tr>
<tr>
<td>0x1013</td>
<td>Source address lower 32 bits</td>
<td>[31:0]</td>
<td>Source address (lower 32 bits)</td>
<td>0x43210ADD</td>
<td>RW</td>
</tr>
<tr>
<td>0x1014</td>
<td>Source address upper 16 bits</td>
<td>[15:0]</td>
<td>Source address (upper 16 bits)</td>
<td>0x8765</td>
<td>RW</td>
</tr>
<tr>
<td>0x1016</td>
<td>PKT_CL_LOOP BACK_RESET</td>
<td>[0]</td>
<td>MAC loopback reset. Set to the value of 1 to reset the design example MAC loopback.</td>
<td>1'b0</td>
<td>RW</td>
</tr>
</tbody>
</table>

### Related Information

E-Tile Hard IP for Ethernet Intel FPGA IP core register descriptions
2.3. 100GE with Optional RS-FEC Design Example

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

Table 10. Supported Design Example Variants for 100GE

<table>
<thead>
<tr>
<th>Variant</th>
<th>Design Example Support</th>
</tr>
</thead>
</table>
| 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, compilation-only project, and hardware design example |
| 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, compilation-only project, and hardware design example |
| 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, compilation-only project, and hardware design example |
| 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 Hard IP for Ethernet Intel FPGA IP provides preliminary support for the OTN feature. For further inquiries, contact your nearest Intel sales representative or file an Intel Premier Support (IPS) case on https://www.intel.com/content/www/us/en/programmable/my-intel/etal-home.html.

2.3.1. Simulation Design Examples

2.3.1.1. Non-PTP E-Tile Hard IP for Ethernet Intel FPGA IP 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.
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 Hard IP for Ethernet Intel FPGA IP 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.

<table>
<thead>
<tr>
<th>Event Description</th>
<th>Time (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>o_tx_lanes_stable is 1</td>
<td>345651500</td>
</tr>
<tr>
<td>waiting for tx_dll_lock...</td>
<td>398849563</td>
</tr>
<tr>
<td>TX DLL LOCK is 1</td>
<td>399169435</td>
</tr>
<tr>
<td>waiting for tx_transfer_ready...</td>
<td>410719813</td>
</tr>
<tr>
<td>TX transfer ready is 1</td>
<td>410776000</td>
</tr>
<tr>
<td>RX transfer ready is 1</td>
<td>411040000</td>
</tr>
<tr>
<td>EHIP PLD Ready out is 1</td>
<td>412282101</td>
</tr>
<tr>
<td>EHIP reset out is 0</td>
<td>462643731</td>
</tr>
<tr>
<td>EHIP reset ack is 0</td>
<td>462750387</td>
</tr>
<tr>
<td>EHIP TX reset out is 0</td>
<td>463088000</td>
</tr>
<tr>
<td>EHIP TX reset ack is 0</td>
<td>463283667</td>
</tr>
<tr>
<td>Waiting for EHIP Ready...</td>
<td>467376591</td>
</tr>
<tr>
<td>EHIP READY is 1</td>
<td>468643131</td>
</tr>
<tr>
<td>EHIP RX reset out is 0</td>
<td>469169435</td>
</tr>
<tr>
<td>waiting for rx reset ack...</td>
<td>469476591</td>
</tr>
<tr>
<td>EHIP RX ready is 1</td>
<td>469789000</td>
</tr>
<tr>
<td>RX deskew locked</td>
<td>469890000</td>
</tr>
<tr>
<td>RX lane alignment locked</td>
<td>469990000</td>
</tr>
<tr>
<td>** Sending Packet 1...</td>
<td>470090000</td>
</tr>
<tr>
<td>** Sending Packet 2...</td>
<td>470190000</td>
</tr>
<tr>
<td>** Sending Packet 3...</td>
<td>470290000</td>
</tr>
<tr>
<td>** Sending Packet 4...</td>
<td>470390000</td>
</tr>
<tr>
<td>** Sending Packet 5...</td>
<td>470490000</td>
</tr>
<tr>
<td>** Sending Packet 6...</td>
<td>470590000</td>
</tr>
<tr>
<td>** Sending Packet 7...</td>
<td>470690000</td>
</tr>
<tr>
<td>** Received Packet 1...</td>
<td>470790000</td>
</tr>
<tr>
<td>** Sending Packet 8...</td>
<td>470890000</td>
</tr>
<tr>
<td>** Received Packet 2...</td>
<td>470990000</td>
</tr>
<tr>
<td>** Sending Packet 9...</td>
<td>471090000</td>
</tr>
<tr>
<td>** Received Packet 3...</td>
<td>471190000</td>
</tr>
<tr>
<td>** Received Packet 4...</td>
<td>471290000</td>
</tr>
<tr>
<td>** Sending Packet 10...</td>
<td>471390000</td>
</tr>
<tr>
<td>** Received Packet 5...</td>
<td>471490000</td>
</tr>
<tr>
<td>** Received Packet 6...</td>
<td>471590000</td>
</tr>
<tr>
<td>** Received Packet 7...</td>
<td>471690000</td>
</tr>
<tr>
<td>** Received Packet 8...</td>
<td>471790000</td>
</tr>
<tr>
<td>** Received Packet 9...</td>
<td>471890000</td>
</tr>
<tr>
<td>** Received Packet 10...</td>
<td>471990000</td>
</tr>
<tr>
<td>=====MATCH! ReaddataValid = 1 Readdata = 11112015 Expected_Readdata = 11112015</td>
<td>472090000</td>
</tr>
<tr>
<td>=====MATCH! ReaddataValid = 1 Readdata = 11112015 Expected_Readdata = 11112015</td>
<td>472190000</td>
</tr>
<tr>
<td>=====MATCH! ReaddataValid = 1 Readdata = 11112015 Expected_Readdata = 11112015</td>
<td>472290000</td>
</tr>
<tr>
<td>=====MATCH! ReaddataValid = 1 Readdata = 11112015 Expected_Readdata = 11112015</td>
<td>472390000</td>
</tr>
<tr>
<td>=====MATCH! ReaddataValid = 1 Readdata = 11112015 Expected_Readdata = 11112015</td>
<td>472490000</td>
</tr>
<tr>
<td>=====MATCH! ReaddataValid = 1 Readdata = 11112015 Expected_Readdata = 11112015</td>
<td>472590000</td>
</tr>
<tr>
<td>=====MATCH! ReaddataValid = 1 Readdata = 11112015 Expected_Readdata = 11112015</td>
<td>472690000</td>
</tr>
<tr>
<td>=====MATCH! ReaddataValid = 1 Readdata = 11112015 Expected_Readdata = 11112015</td>
<td>472790000</td>
</tr>
<tr>
<td>=====MATCH! ReaddataValid = 1 Readdata = 11112015 Expected_Readdata = 11112015</td>
<td>472890000</td>
</tr>
<tr>
<td>=====MATCH! ReaddataValid = 1 Readdata = 11112015 Expected_Readdata = 11112015</td>
<td>472990000</td>
</tr>
<tr>
<td>=====MATCH! ReaddataValid = 1 Readdata = 11112015 Expected_Readdata = 11112015</td>
<td>473090000</td>
</tr>
<tr>
<td>=====MATCH! ReaddataValid = 1 Readdata = 11112015 Expected_Readdata = 11112015</td>
<td>473190000</td>
</tr>
<tr>
<td>=====MATCH! ReaddataValid = 1 Readdata = 11112015 Expected_Readdata = 11112015</td>
<td>473290000</td>
</tr>
</tbody>
</table>
2. E-Tile Hard IP for Ethernet Intel FPGA IP Design Example

Simulating the E-Tile Hard IP for Ethernet Intel FPGA IP Design Example Testbench on page 11

2.3.1.2. E-Tile Hard IP for Ethernet Intel FPGA IP 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.

---

**Related Information**

Simulating the E-Tile Hard IP for Ethernet Intel FPGA IP Design Example Testbench on page 11

---

```plaintext
# =====> 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.
**
**************************************************
Figure 16. Simulation Block Diagram for E-Tile Hard IP for Ethernet Intel FPGA IP 100GE MAC+PCS with Optional RS-FEC and PTP Design Example

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.

```
Waiting for RX alignment
RX deskew locked
RX lane alignment 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_ln_skew_rx_0
rsfec_ln_skew_rx_0 = 32'h0
Reading rsfec_cw_pos_rx_0
rsfec_cw_pos_rx_0 = 32'h1c5

Testbench complete.
```
Reading rsfec_ln_skew_rx_3
rsfec_ln_skew_rx_3 = 32'h1
Reading rsfec_cw_pos_rx_3
rsfec_cw_pos_rx_3 = 32'h1c5
min skew value = 32'h0
lane_skew_adjust = 32'h0
Tlat_final = 32'h0
Generate VL offset data
before-rotation: VL[PL] 0[0], deskew_delay = 0 UI, vl_offset_bits = 0
After rotation: VL_OFFSET for RVL[PL] 4[0] = 0 ns 0 Fns, Sign bit= 0
  .
  .
before-rotation: VL[PL] 19[0], deskew_delay = 0 UI, vl_offset_bits = 0
before-rotation: VL[PL] 19[0], deskew_delay = 0 UI, vl_offset_bits_shifted = -326
After rotation: VL_OFFSET for RVL[PL] 3[0] = c ns a515 Fns, Sign bit= 1
Writing VL offset data for VL 0
====> writedata = 00000004
====> writedata = 00000000
 .
  .
Writing VL offset data for VL 19
====> writedata = 00000000
====> writedata = 800ca515
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...
RX and TX timestamp range of difference is from -2.875549 ns to -2.870483 ns
**
** Testbench complete.
**
******************************************************************************
$finish called from file "basicavl_top.sv", line 713.
$finish at simulation time 5323700000

Related Information

Simulating the E-Tile Hard IP for Ethernet Intel FPGA IP Design Example Testbench on page 11
2.3.1.3. E-Tile Hard IP for Ethernet Intel FPGA IP 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.

![Simulation Block Diagram for E-Tile Hard IP for Ethernet Intel FPGA IP 100GE PCS Only Design Example](image)

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 \textit{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....               413726943
TX DLL LOCK is 1 at time                  414046815
waiting for tx_transfer_ready....         425122383
TX transfer ready is 1 at time             425184000
RX transfer ready is 1 at time             425320000
EHIP reset out 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....                476910363
EHIP READY is 1 at time                   478680000
EHIP RX reset out is 0 at time             482711523
EHIP RX reset ack is 0 at time             482723200
EHIP Rx Block Lock  is high at time        481446403
Waiting for AM lock                       482711523
EHIP Rx am Lock  is high at time           482723200
Waiting for RX alignment                  476830347
RX deskew locked                          478680000
RX lane alignment locked
Sending Packets and Receiving Packets
====> writedata = 00000001
====>MATCH!     ReaddataValid = 1 Readdata = 00000053 Expected_Readdata = 00000053
** Testbench complete.
** ******************************************************
```

**Related Information**

Simulating the E-Tile Hard IP for Ethernet Intel FPGA IP Design Example Testbench on page 11

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

Figure 18. Simulation Block Diagram for E-Tile Hard IP for Ethernet Intel FPGA IP 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 Hard IP for Ethernet Intel FPGA IP 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.

```plaintext
# 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
```
2.3.1.5. E-Tile Hard IP for Ethernet Intel FPGA IP 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 Hard IP for Ethernet Intel FPGA IP 100GE FlexE Design Example**

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 399849563
# 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 0 at time 410776000
# dut: EHIP reset out is 0 at time 411040000
# dut: EHIP reset ack is 0 at time 412282101
```
<table>
<thead>
<tr>
<th>Event Description</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>dut: EHIP TX reset out is 0 at time</td>
<td>41316000</td>
</tr>
<tr>
<td>dut: EHIP TX reset ack is 0 at time</td>
<td>46264371</td>
</tr>
<tr>
<td>dut: waiting for EHIP Ready....</td>
<td></td>
</tr>
<tr>
<td>dut: RX PLL READY is 1 at time</td>
<td>462750387</td>
</tr>
<tr>
<td>dut: EHIP RX reset out is 0 at time</td>
<td>463088000</td>
</tr>
<tr>
<td>dut: waiting for rx reset ack....</td>
<td></td>
</tr>
<tr>
<td>dut: EHIP RX reset ack is 0 at time</td>
<td>463283667</td>
</tr>
<tr>
<td>dut: Waiting for RX Block Lock</td>
<td></td>
</tr>
<tr>
<td>test_dut: TX PLL LOCK is 1 at time</td>
<td>475452243</td>
</tr>
<tr>
<td>test_dut: waiting for TX_transfer_ready...</td>
<td></td>
</tr>
<tr>
<td>test_dut: TX transfer ready is 1 at time</td>
<td>475772115</td>
</tr>
<tr>
<td>test_dut: waiting for RX_transfer_ready...</td>
<td></td>
</tr>
<tr>
<td>test_dut: RX transfer ready is 1 at time</td>
<td>487164223</td>
</tr>
<tr>
<td>test_dut: EHIP PLL Ready out is 1 at time</td>
<td>487224000</td>
</tr>
<tr>
<td>test_dut: EHIP reset out is 0 at time</td>
<td>487488000</td>
</tr>
<tr>
<td>test_dut: EHIP reset ack is 0 at time</td>
<td>488907771</td>
</tr>
<tr>
<td>test_dut: EHIP TX reset out is 0 at time</td>
<td>489784000</td>
</tr>
<tr>
<td>test_dut: EHIP TX reset ack is 0 at time</td>
<td>539116083</td>
</tr>
<tr>
<td>test_dut: waiting for EHIP Ready....</td>
<td></td>
</tr>
<tr>
<td>test_dut: RX PLL READY is 1 at time</td>
<td>539169411</td>
</tr>
<tr>
<td>test_dut: EHIP RX reset out is 0 at time</td>
<td>539512000</td>
</tr>
<tr>
<td>test_dut: waiting for rx reset ack....</td>
<td></td>
</tr>
<tr>
<td>test_dut: EHIP RX reset ack is 0 at time</td>
<td>539702691</td>
</tr>
<tr>
<td>test_dut: Waiting for RX Block Lock</td>
<td></td>
</tr>
<tr>
<td>dut: EHIP RX Block Lock is high at time</td>
<td>542102451</td>
</tr>
<tr>
<td>dut: Waiting for AM lock</td>
<td></td>
</tr>
<tr>
<td>dut: EHIP RX AM Lock is high at time</td>
<td>543368991</td>
</tr>
<tr>
<td>dut: Waiting for RX alignment</td>
<td></td>
</tr>
<tr>
<td>dut: RX deskew locked</td>
<td></td>
</tr>
<tr>
<td>dut: RX lane alignment locked</td>
<td></td>
</tr>
<tr>
<td>dut: *****************************************</td>
<td></td>
</tr>
<tr>
<td>test_dut: EHIP RX Block Lock is high at time</td>
<td>546535341</td>
</tr>
<tr>
<td>test_dut: EHIP RX AM Lock is high at time</td>
<td>547801881</td>
</tr>
<tr>
<td>test_dut: Waiting for RX alignment</td>
<td></td>
</tr>
<tr>
<td>test_dut: RX deskew locked</td>
<td></td>
</tr>
<tr>
<td>test_dut: RX lane alignment locked</td>
<td></td>
</tr>
<tr>
<td>test_dut: ** Sending Packet 1...</td>
<td></td>
</tr>
<tr>
<td>test_dut: ** Sending Packet 9...</td>
<td></td>
</tr>
<tr>
<td>test_dut: ** Sending Packet 10...</td>
<td></td>
</tr>
<tr>
<td>test_dut: ** Received Packet 1...</td>
<td></td>
</tr>
<tr>
<td>test_dut: ** Received Packet 9...</td>
<td></td>
</tr>
<tr>
<td>test_dut: ** Received Packet 10...</td>
<td></td>
</tr>
<tr>
<td>test_dut: **</td>
<td></td>
</tr>
<tr>
<td>test_dut: ** Testbench complete.</td>
<td></td>
</tr>
<tr>
<td>test_dut: **</td>
<td></td>
</tr>
<tr>
<td>test_dut: *****************************************</td>
<td></td>
</tr>
</tbody>
</table>

Related Information

Simulating the E-Tile Hard IP for Ethernet Intel FPGA IP Design Example Testbench on page 11
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

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

- E-Tile Hard IP for Ethernet Intel FPGA IP 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.
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:

```bash
% 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 ----- 
---------------------------
(Rx)
STATISTICS FOR BASE 0x000900

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 : 55800
1519 - MAX Byte Frames : 1659560
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
```

--- 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 ----- 
---------------------------
Pause Control Frames : 0

----------------------
STATISTICS FOR BASE 0x000800

----------------------
Fragmented Frames : 0
Jabbered Frames : 0
Any Size with PCS Err Frame : 0
Right Size with PCS Err Frame : 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 : 55880
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 ------
-------- Appyling TX and RX Reset --------
wait for phy lock=50, locked=1
--Iteration:0 - PMA Adaptation 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

---

<table>
<thead>
<tr>
<th>Fragmented Frames</th>
<th>: 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jabbered Frames</td>
<td>: 0</td>
</tr>
<tr>
<td>Any Size with FCS Err Frame</td>
<td>: 0</td>
</tr>
<tr>
<td>Right Size with FCS Err Frame</td>
<td>: 0</td>
</tr>
<tr>
<td>Multicast data Err Frames</td>
<td>: 0</td>
</tr>
<tr>
<td>Broadcast data Err Frames</td>
<td>: 0</td>
</tr>
<tr>
<td>Unicast data Err Frames</td>
<td>: 0</td>
</tr>
<tr>
<td>Multicast control Err Frame</td>
<td>: 0</td>
</tr>
<tr>
<td>Broadcast control Err Frame</td>
<td>: 0</td>
</tr>
<tr>
<td>Unicast control Err Frames</td>
<td>: 0</td>
</tr>
<tr>
<td>Pause control Err Frames</td>
<td>: 0</td>
</tr>
</tbody>
</table>

| 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 |
| > MAX Byte Frames | : 0 |
| 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 0x000900

---

<table>
<thead>
<tr>
<th>Fragmented Frames</th>
<th>: 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jabbered Frames</td>
<td>: 0</td>
</tr>
<tr>
<td>Any Size with FCS Err Frame</td>
<td>: 0</td>
</tr>
<tr>
<td>Right Size with FCS Err Frame</td>
<td>: 0</td>
</tr>
<tr>
<td>Multicast data Err Frames</td>
<td>: 0</td>
</tr>
<tr>
<td>Broadcast data Err Frames</td>
<td>: 0</td>
</tr>
<tr>
<td>Unicast data Err Frames</td>
<td>: 0</td>
</tr>
<tr>
<td>Multicast control Err Frame</td>
<td>: 0</td>
</tr>
<tr>
<td>Broadcast control Err Frame</td>
<td>: 0</td>
</tr>
<tr>
<td>Unicast control Err Frames</td>
<td>: 0</td>
</tr>
<tr>
<td>Pause control Err Frames</td>
<td>: 0</td>
</tr>
</tbody>
</table>

| 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 |
| > MAX Byte Frames | : 0 |
| Tx Frame Starts | : 1840468 |
| Multicast data OK Frame | : 0 |
| Broadcast data OK Frame | : 0 |

---

### STATISTICS FOR BASE 0x000800

---

<table>
<thead>
<tr>
<th>Fragmented Frames</th>
<th>: 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jabbered Frames</td>
<td>: 0</td>
</tr>
<tr>
<td>Any Size with FCS Err Frame</td>
<td>: 0</td>
</tr>
<tr>
<td>Right Size with FCS Err Frame</td>
<td>: 0</td>
</tr>
<tr>
<td>Multicast data Err Frames</td>
<td>: 0</td>
</tr>
<tr>
<td>Broadcast data Err Frames</td>
<td>: 0</td>
</tr>
<tr>
<td>Unicast data Err Frames</td>
<td>: 0</td>
</tr>
<tr>
<td>Multicast control Err Frame</td>
<td>: 0</td>
</tr>
<tr>
<td>Broadcast control Err Frame</td>
<td>: 0</td>
</tr>
<tr>
<td>Unicast control Err Frames</td>
<td>: 0</td>
</tr>
<tr>
<td>Pause control Err Frames</td>
<td>: 0</td>
</tr>
</tbody>
</table>

| 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 |
| > MAX Byte Frames | : 0 |

---
### Related Information

- Intel Stratix 10 TX Signal Integrity Development Kit Webpage
- Compiling and Configuring the Design Example in Hardware on page 12
- Testing the E-Tile Hard IP for Ethernet Intel FPGA IP 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

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

- E-Tile Hard IP for Ethernet Intel FPGA IP core.
- Client logic that coordinates the programming of the IP core and packet generation.
- 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 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
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.
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 = 0x00009EE01
Info: TX Extra Latency = 0xc69814
Info: RX Extra Latency = 0x5467088

Successfully Write 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 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
cnter_tx_64b_lo                        , offset = 0x816, data = 0x2
Successfully Read  EHIP User Register
cnter_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_badt_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
- Intel Stratix 10 TX Signal Integrity Development Kit Webpage
- Compiling and Configuring the Design Example in Hardware on page 12
- Testing the E-Tile Hard IP for Ethernet Intel FPGA IP Hardware Design Example on page 13
2.3.2.3. 100GE PCS with Optional RS-FEC Hardware Design Example Components

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

- E-Tile Hard IP for Ethernet Intel FPGA IP 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:

```shell
% 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)
  TXRSLCK : 0 (KHZ)
  RXRSLCK : 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
```

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

```shell
% 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)
  TXRSLCK : 0 (KHZ)
  RXRSLCK : 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
Setting Number of frames to 5340
Setting Size of frames to random
```

### Related Information

- Intel Stratix 10 TX Signal Integrity Development Kit Webpage
- Compiling and Configuring the Design Example in Hardware on page 12
- Testing the E-Tile Hard IP for Ethernet Intel FPGA IP Hardware Design Example on page 13

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

The E-Tile Hard IP for Ethernet Intel FPGA IP 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

<table>
<thead>
<tr>
<th>Signal</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clk50</td>
<td>Input</td>
<td>Drive at 50 MHz. The intent is to drive this from a 50 Mhz oscillator on the board.</td>
</tr>
<tr>
<td>i_clk_ref</td>
<td>Input</td>
<td>Drive at 156.25 MHz.</td>
</tr>
<tr>
<td>cpu_resetn</td>
<td>Input</td>
<td>Resets the IP core. Active low. Drives the global hard reset csr_reset_n to the IP core.</td>
</tr>
<tr>
<td>i_rx_serial[3:0]</td>
<td>Input</td>
<td>Transceiver PHY input serial data.</td>
</tr>
<tr>
<td>o_tx_serial[3:0]</td>
<td>Output</td>
<td>Transceiver PHY output serial data.</td>
</tr>
<tr>
<td>user_led[3:0]</td>
<td>Output</td>
<td>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.</td>
</tr>
</tbody>
</table>

**Related Information**

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

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

The E-Tile Hard IP for Ethernet Intel FPGA IP 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

<table>
<thead>
<tr>
<th>Signal</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clk50</td>
<td>Input</td>
<td>Drive at 50 MHz. The intent is to drive this from a 50 Mhz oscillator on the board.</td>
</tr>
<tr>
<td>i_clk_ref</td>
<td>Input</td>
<td>Drive at 156.25 MHz.</td>
</tr>
<tr>
<td>cpu_resetn</td>
<td>Input</td>
<td>Resets the IP core. Active low. Drives the global hard reset csr_reset_n to the IP core.</td>
</tr>
<tr>
<td>i_rx_serial[3:0]</td>
<td>Input</td>
<td>Transceiver PHY input serial data.</td>
</tr>
<tr>
<td>o_tx_serial[3:0]</td>
<td>Output</td>
<td>Transceiver PHY output serial data.</td>
</tr>
<tr>
<td>user_led[3:0]</td>
<td>Output</td>
<td>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.</td>
</tr>
</tbody>
</table>

**Related Information**

E-Tile Hard IP for Ethernet Intel FPGA IP Interfaces and Signal Descriptions
2.3.5. 100GE MAC+PCS with Optional RS-FEC Design Example Registers

Table 13. E-Tile Hard IP for Ethernet Intel FPGA IP 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.

<table>
<thead>
<tr>
<th>Word Offset</th>
<th>Register Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000000</td>
<td>KR4 registers</td>
</tr>
<tr>
<td>0x000300</td>
<td>RX PCS registers</td>
</tr>
<tr>
<td>0x000400</td>
<td>TX MAC registers</td>
</tr>
<tr>
<td>0x000500</td>
<td>RX MAC registers</td>
</tr>
<tr>
<td>0x000800</td>
<td>TX Statistics Counter registers</td>
</tr>
<tr>
<td>0x000900</td>
<td>RX Statistics Counter registers</td>
</tr>
<tr>
<td>0x001000</td>
<td>Packet Client registers</td>
</tr>
<tr>
<td>0x002000</td>
<td>Packet monitoring registers</td>
</tr>
<tr>
<td>0x010000</td>
<td>RS-FEC configuration registers</td>
</tr>
<tr>
<td>0x100000</td>
<td>Transceiver registers</td>
</tr>
</tbody>
</table>

Table 14. Packet Client Registers

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

<table>
<thead>
<tr>
<th>Addr</th>
<th>Name</th>
<th>Bit</th>
<th>Description</th>
<th>HW Reset Value</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x1000</td>
<td>Pkt_CL_SCRA</td>
<td>[31:0]</td>
<td>Scratch register available for testing.</td>
<td></td>
<td>RW</td>
</tr>
<tr>
<td>0x1001</td>
<td>Pkt_CL_CLNT</td>
<td>[31:0]</td>
<td>Four characters of IP block identification string “CLNT”</td>
<td></td>
<td>RO</td>
</tr>
</tbody>
</table>
| 0x1008 | Packet Size       | [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     | [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...
## Related Information

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

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

#### Table 15. E-Tile Hard IP for Ethernet Intel FPGA IP 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.

<table>
<thead>
<tr>
<th>Word Offset</th>
<th>Register Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000000</td>
<td>KR4 registers</td>
</tr>
<tr>
<td>0x000300</td>
<td>RX PCS registers</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Addr</th>
<th>Name</th>
<th>Bit</th>
<th>Description</th>
<th>HW Reset Value</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x1011</td>
<td>Destination address lower 32 bits</td>
<td>[31:0]</td>
<td>Destination address (lower 32 bits)</td>
<td>0x56780ADD</td>
<td>RW</td>
</tr>
<tr>
<td>0x1012</td>
<td>Destination address upper 16 bits</td>
<td>[15:0]</td>
<td>Destination address (upper 16 bits)</td>
<td>0x1234</td>
<td>RW</td>
</tr>
<tr>
<td>0x1013</td>
<td>Source address lower 32 bits</td>
<td>[31:0]</td>
<td>Source address (lower 32 bits)</td>
<td>0x43210ADD</td>
<td>RW</td>
</tr>
<tr>
<td>0x1014</td>
<td>Source address upper 16 bits</td>
<td>[15:0]</td>
<td>Source address (upper 16 bits)</td>
<td>0x8765</td>
<td>RW</td>
</tr>
<tr>
<td>0x1016</td>
<td>PKT_CL_LOOP BACK_RESET</td>
<td>[0]</td>
<td>MAC loopback reset. Set to the value of 1 to reset the design example MAC loopback.</td>
<td>1'b0</td>
<td>RW</td>
</tr>
</tbody>
</table>
### Table 16. Packet Generator and Checker Registers
You can customize the E-Tile Hard IP for Ethernet Intel FPGA IP hardware design example by programming the packet client registers.

<table>
<thead>
<tr>
<th>Addr</th>
<th>Name</th>
<th>Bit</th>
<th>Description</th>
<th>HW Reset Value</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xF000</td>
<td>Control Register 0</td>
<td>[0]</td>
<td>Write 1 to start transmitting PCS packets.</td>
<td>0x0</td>
<td>RWC</td>
</tr>
<tr>
<td>0xF001</td>
<td>Control Register 1</td>
<td>[0]</td>
<td>Write 1 to reset the channel.</td>
<td>0x0</td>
<td>RW</td>
</tr>
</tbody>
</table>
| 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     |
2.4. Document Revision History for the E-Tile Hard IP for Ethernet Intel FPGA IP Design Example

<table>
<thead>
<tr>
<th>Document Version</th>
<th>Intel Quartus Prime Version</th>
<th>IP Version</th>
<th>Changes</th>
</tr>
</thead>
</table>
| 2019.12.23       | 19.4                       | 19.4.0     | • 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. |
| 2019.09.30       | 19.3                       | 19.3.0     | • Updated the List of Supported Design Example Variants table in E-tile Hard IP for Ethernet Intel FPGA IP Quick Start Guide:  
— Added support for Single or multi channels custom PCS with optional RS-FEC for 10GE variant.  
— Removed Asynchronous clock support from the 100GE MAC+PCS with (528,514) RS-FEC and PTP variant  
— Added support for 100GE MAC+PCS with (544,514) RS-REC variant  
• Updated the List of Supported Design Example Variants in 10GE/25GE with Optional RS-FEC Design Examples.  
• Updated register name Source address upper, 16 bits in the Packet Client Register table for all variants. |
| 2019.05.17       | 19.1                       | 19.1       | • Renamed the document as E-tile Hard IP Intel Stratix 10 Design Examples User Guide.  
• Added 10GE/25GE custom PCS with optional RS-FEC simulation, compilation-only project, and hardware design examples.  
• Added E-Tile Hard IP for Ethernet Intel FPGA IP 100GE MAC+PCS with optional RS-FEC and PTP simulation, compilation-only project, and hardware design examples. |

continued...
<table>
<thead>
<tr>
<th>Document Version</th>
<th>Intel Quartus Prime Version</th>
<th>IP Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Updated the Testing the E-Tile Hard IP for Ethernet Intel FPGA IP Hardware Design Example topic:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>— Added a new subtopic: 10GE/25GE Custom PCS with Optional RS-FEC Hardware Design Example.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>— Updated the 10GE/25GE MAC+PCS with Optional RS-FEC and Optional PTP Hardware Design Example and 10GE/25GE PCS Only with Optional RS-FEC Hardware Design Example subtopics.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>— Updated the following commands in the 100GE MAC+PCS with Optional (528,514) RS-FEC and PMA Calibration Hardware Design Example and 100GE MAC+PCS with Optional (544,514) RS-FEC and PMA Calibration Hardware Design Example topics:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• start_pma_init_adaptation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• start_pma_anlg_rst03</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• init_adaptation_16_NoPrbsNoLdEL03</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• chk_init_adaptation_status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• chk_init_adaptation_status_02</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>— Updated Figure: In-System Sources and Probes Editor in the following topics:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 10GE/25GE MAC+PCS with Optional RS-FEC and Optional PTP Hardware Design Example</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 10GE/25GE PCS Only with Optional RS-FEC Hardware Design Example</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 10GE/25GE Custom PCS with Optional RS-FEC Hardware Design Example</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Updated Table: List of Supported Design Example Variants.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Updated Table: Packet Generator and Checker Registers of the 100GE PCS with Optional RS-FEC Design Example Registers topic to update the register names for address 0xF000 and 0xF001.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Updated the following Figures:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>— Simulation Block Diagram for Non-PTP E-Tile Hard IP for Ethernet Intel FPGA IP 10GE/25GE MAC+PCS with Optional RS-FEC Design Example.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>— Simulation Block Diagram for E-Tile Hard IP for Ethernet Intel FPGA IP 10GE/25GE with Optional RS-FEC and PTP Design Example.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>— Simulation Block Diagram for E-Tile Hard IP for Ethernet Intel FPGA IP 10GE/25GE with Optional RS-FEC and PTP Design Example.</td>
</tr>
</tbody>
</table>

continued...
• Updated the 10GE/25GE with Optional RS-FEC Design Example chapter:
  — Updated Table: Supported Design Example Variants for 10GE/25GE.
  — Updated the following simulation design example topics:
    • Non-PTP 10GE/25GE MAC+PCS with Optional RS-FEC Simulation Design Example
    • 10GE/25GE MAC+PCS with Optional RS-FEC and PTP Simulation Design Example
    • 10GE/25GE PCS Only, OTN, or FlexE with Optional RS-FEC Simulation Design Example
    • 10GE/25GE Custom PCS with Optional RS-FEC Simulation Design Example
  — Added new Table: Packet Generator Registers.
• Updated the 100GE with Optional RS-FEC Design Example chapter:
  — Updated the following simulation design example topics:
    • Non-PTP E-Tile Hard IP for Ethernet Intel FPGA IP 100GE MAC+PCS with Optional RS-FEC Simulation Design Example.
    • E-Tile Hard IP for Ethernet Intel FPGA IP 100GE MAC+PCS with Optional RS-FEC and PTP Simulation Design Example
    • E-Tile Hard IP for Ethernet Intel FPGA IP 100GE PCS Only with Optional RS-FEC Simulation Design Example
    • E-Tile Hard IP for Ethernet Intel FPGA IP 100GE OTN with Optional RS-FEC Simulation Design Example
    • E-Tile Hard IP for Ethernet Intel FPGA IP 100GE FlexE with Optional RS-FEC Simulation Design Example
  — Updated the following Figures:
    • Simulation Block Diagram for E-Tile Hard IP for Ethernet Intel FPGA IP 100GE MAC+PCS with Optional RS-FEC Design Example.
    • Simulation Block Diagram for E-Tile Hard IP for Ethernet Intel FPGA IP 100GE OTN Design Example.
  — Made editorial updates throughout the document.

<table>
<thead>
<tr>
<th>Document Version</th>
<th>Intel Quartus Prime Version</th>
<th>IP Version</th>
<th>Changes</th>
</tr>
</thead>
</table>
| 2019.02.14        | 18.1.1                      | 18.1.1     | Updated Table: Steps to Simulate the Testbench to include instruction for Xcelium simulator.

continued...
<table>
<thead>
<tr>
<th>Document Version</th>
<th>Intel Quartus Prime Version</th>
<th>IP Version</th>
<th>Changes</th>
</tr>
</thead>
</table>
| 2019.01.04       | 18.1.1                     | 18.1.1     | • Added information for the following design examples:  
|                  |                            |            | — 10GE/25GE PCS Only, OTN, and FlexE RSFEC simulation and compilation-only project design examples.  
|                  |                            |            | — 10GE/25GE PCS Only hardware design example.  
|                  |                            |            | — Multi channel 10GE/25GE MAC + PCS with optional RSFEC and PTP simulation, compilation-only project, and hardware design examples.  
|                  |                            |            | — 100GE MAC + PCS with optional RSFEC(544,514) simulation, compilation-only project, and hardware design examples.  
|                  |                            |            | — 100GE PCS Only with optional RSFEC(528,514) and RSFEC(544,514) simulation, compilation-only, and hardware design examples.  
|                  |                            |            | • Updated steps to test 10GE/25GE MAC + PCS with optional RSFEC and optional PTP and 100GE MAC +PCS with optional RSFEC and PMA calibration hardware design examples.  
|                  |                            |            | • Updated result log for 10GE/25GE MAC + PCS with optional RSFEC and optional PTP and 100GE MAC +PCS with optional RSFEC and PMA calibration hardware design examples.  
|                  |                            |            | • Updated simulation and hardware design example block diagram for 10GE/25GE MAC + PCS with optional RSFEC and non-PTP 10GE/25GE MAC + PCS with optional RSFEC variants.  
|                  |                            |            | • Updated register map for 10GE/25GE and 100GE design examples. |
| 2018.08.10       | 18.0                       | 18.0       | Added a note to clarify that the E-Tile Hard IP for Ethernet Intel FPGA IP provides preliminary support for the OTN feature in the following sections:  
|                  |                            |            | • Quick Start Guide  
|                  |                            |            | • 100GE with Optional RSFEC Design Example  
|                  |                            |            | • 100GE OTN Simulation Design Example |
| 2018.07.19       | 18.0                       | 18.0       | Initial release. |
3. E-tile CPRI PHY Intel FPGA IP Design Example

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

The E-tile CPRI PHY Intel FPGA IP core for Intel Stratix 10 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 Stratix 10 TX Transceiver Signal Integrity Development Kit. Intel provides a compilation-only example project that you can use to quickly estimate IP core area and timing.

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.

Related Information

- E-tile Hard IP User Guide
  For detailed information on E-tile CPRI PHY IP.
- About the E-Tile CPRI PHY
  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:
3.1.2. Generating the Design

**Figure 24.** Procedure

1. **Start Parameter Editor**
2. **Specify IP Variation and Select Device**
3. **Select Design Parameters**
4. **Specify Example Design**
5. **Initiate Design Generation**

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

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

1. In the Intel Quartus Prime Pro Edition, click **File ➤ New Project Wizard** to create a new Quartus Prime project, or **File ➤ Open Project** to open an existing Intel Quartus Prime project. The wizard prompts you to specify a device.

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

3. Click **Finish**.

Follow these steps to generate the E-tile 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 **Stratix 10 TX Transceiver Signal Integrity Development Kit** or select **None**. The compilation-only and hardware design examples target your project device. For the hardware design to 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

### 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

<table>
<thead>
<tr>
<th>File Names</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;design_example_dir&gt;/example_testbench/basic_avl_tb_top.sv</td>
<td>Top-level testbench file. The testbench instantiates the DUT wrapper and runs Verilog HDL tasks to generate and accept packets.</td>
</tr>
<tr>
<td>&lt;design_example_dir&gt;/example_testbench/alt_cpriphy_c3_top.sv</td>
<td>DUT wrapper that instantiates DUT and other testbench components.</td>
</tr>
<tr>
<td>&lt;design_example_dir&gt;/example_testbench/run_vsims.do</td>
<td>The Mentor Graphics ModelSim script to run the testbench.</td>
</tr>
<tr>
<td>&lt;design_example_dir&gt;/example_testbench/run_vcs.sh</td>
<td>The Synopsys VCS script to run the testbench.</td>
</tr>
<tr>
<td>&lt;design_example_dir&gt;/example_testbench/run_vcsmx.sh</td>
<td>The Synopsys VCS MX* script (combined Verilog HDL and SystemVerilog with VHDL) to run the testbench.</td>
</tr>
<tr>
<td>&lt;design_example_dir&gt;/example_testbench/run_nxcsim.sh</td>
<td>The Cadence NCSim script to run the testbench.</td>
</tr>
<tr>
<td>&lt;design_example_dir&gt;/example_testbench/run_xcelium.sh</td>
<td>The Xcelium script to run the testbench.</td>
</tr>
</tbody>
</table>
Table 18. E-tile CPRI PHY Intel FPGA IP Core Hardware Design Example File Descriptions

<table>
<thead>
<tr>
<th>File Names</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;design_example_dir&gt;/hardware_test_design/alt_cpriphy_c3_hw.qpf</code></td>
<td>Intel Quartus Prime project file.</td>
</tr>
<tr>
<td><code>&lt;design_example_dir&gt;/hardware_test_design/alt_cpriphy_c3_hw.qsf</code></td>
<td>Intel Quartus Prime project setting file.</td>
</tr>
<tr>
<td><code>&lt;design_example_dir&gt;/hardware_test_design/alt_cpriphy_c3_hw.sdc</code></td>
<td>Synopsys Design Constraints files. You can copy and modify these files for your own Intel Stratix 10 design.</td>
</tr>
<tr>
<td><code>&lt;design_example_dir&gt;/hardware_test_design/alt_cpriphy_c3_hw.v</code></td>
<td>Top-level Verilog HDL design example file.</td>
</tr>
<tr>
<td><code>&lt;design_example_dir&gt;/hardware_test_design/alt_cpriphy_c3_top.sv</code></td>
<td>DUT wrapper that instantiates DUT and other testbench components.</td>
</tr>
<tr>
<td><code>&lt;design_example_dir&gt;/hardware_test_design/hwtest_sl/main_script.tcl</code></td>
<td>Main file for accessing System Console.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Simulator</th>
<th>Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mentor Graphics</td>
<td>In the command line, type <code>vsim -do run_vsim.do</code></td>
</tr>
<tr>
<td>ModelSim*</td>
<td>If you prefer to simulate without bringing up the ModelSim GUI, type <code>vsim -c -do run_vsim.do</code></td>
</tr>
<tr>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>Cadence NCSim*</td>
<td>In the command line, type <code>sh run_ncsim.sh</code></td>
</tr>
<tr>
<td>Synopsys VCS*</td>
<td>In the command line, type <code>sh run_vcs.sh</code></td>
</tr>
<tr>
<td>Xcelium*</td>
<td>In the command line, type <code>sh run_xcelium.sh</code></td>
</tr>
</tbody>
</table>
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 = 0x8a[0], ReadData = 0x0
** Address offset = 0x8a, WriteData = 0x00000000
** 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 = 0x00000000
** 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 = 0x00000000
** 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 = 0x00000000
** 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
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 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 = 0xc09, 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 = 0xc09, ReadData = 0x00000066
** 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 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_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 Stratix 10 device:
   a. Connect Intel Stratix 10 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:
      - Y1—156.25 MHz
      - U3, OUT3—100 MHz
      - U3, OUT5— Set this value to 184.32 MHz for the CPRI designs that target 10.1, 12.1 and 24.3 Gbps (with and without RS-FEC) line rates and 153.6 MHz for the CPRI designs that target 2.4/3/4.9/6.1/9.8 Gbps CPRI line rates.
   c. On the **Tools** menu, click **Programmer**.
   d. In the Programmer, click **Hardware Setup**.
   e. Select a programming device.
   f. Select and add the **Stratix 10 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 Stratix 10 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
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 Stratix 10 device, you can use the System Console to program the IP core and its embedded Native PHY IP core registers.

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

1. After the hardware design example is configured on the Intel Stratix 10 device, in the Intel Quartus Prime Pro Edition software, on the **Tools** menu, click **System Debugging Tools ➤ System Console**.

2. In the Td Console pane, type `cd hwtest` to change directory to `<design_example_dir>/hardware_test_design/hwtest`.

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 10.1376 Gbps CPRI line bit rate with 1 CPRI channel:

```bash
source main_script.tcl
Info: Number of Channels = 1
Info: JTAG Port ID = 0
Info: Speed = 10G
Info: Start of c3 cpri test
Info: Basic CPRI test
INFO: Checking PLL Lock status...
    iopll_sclk_Locked I,channel_pll_locked I
INFO: PLL is Locked
INFO: Set Reconfig Reset
INFO: Release Reconfig Reset
INFO: Release CSR Reset
INFO: Release TX Reset
INFO: Release RX Reset
INFO: Release Reset Done!
INFO: Turn on serial Loopback

INFO: Start of C3 ELANE XCVR Channel 0 Loopback mode
INFO: Pooling for PMA Register: Read XCVR CSR Register offset = 0x8a, data= 0x84
INFO: Pooling for PMA Register: Read XCVR CSR Register offset = 0x8b, data= 0x8e
INFO: C3 ELANE XCVR Channel 0 Loopback mode is successfuly enabled
Loop 0
Channel 0 : Wait for measure_valid to assert
Channel 0 : Get checker_pass status:
    Checker value = 1
    Checker status = Passed!
Channel 0 : Read Deterministic latency counts
Info: Loop 0 passed
End of loop 0

Info: End of c3_cpri_test
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 Intel FPGA 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 Simulation Block Diagram for 10.1376, 12.1651, and 24.33024 Gbps (with and without RS-FEC) Line Rates
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 checks for the content and correctness of the hyperframes once the counter completes the round trip latency count.
5. 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

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.
- IOPPLL 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 Stratix 10 TX Transceiver Signal Integrity Development Kit.

Related Information

- Intel Stratix 10 TX Transceiver Signal Integrity Development Kit Web Page
- Intel Stratix 10 TX Transceiver Signal Integrity Development Kit User Guide
3.2.4. Interface Signals

Table 20. Design Example Interface Signals

<table>
<thead>
<tr>
<th>Signal</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ref_clk100MHz</td>
<td>Input</td>
<td>Input clock for CSR access on all the AV-MM interfaces. Drive at 100 MHz.</td>
</tr>
<tr>
<td>ref_clk156MHz</td>
<td>Input</td>
<td>Reference clock for channel PLL. Drive at 156.25 MHz.</td>
</tr>
<tr>
<td>i_clk_ref</td>
<td>Input</td>
<td>Transceiver reference clock. Drive at 153.6 MHz for CPRI line rates 2.4/3/4.9/6.1/9.8 Gbps. Drive at 184.32 MHz for CPRI line rates 10.1, 12.1, and 24.3 Gbps with and without RS-FEC.</td>
</tr>
<tr>
<td>i_rx_serial[n]</td>
<td>Input</td>
<td>Transceiver PHY input serial data.</td>
</tr>
<tr>
<td>o_tx_serial[n]</td>
<td>Output</td>
<td>Transceiver PHY output serial data.</td>
</tr>
</tbody>
</table>

3.2.5. Design Example Register Map for Reconfiguration

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

<table>
<thead>
<tr>
<th>Channel Number</th>
<th>Word Offset</th>
<th>Register Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0x000000</td>
<td>CPRI registers</td>
</tr>
<tr>
<td></td>
<td>0x010000</td>
<td>RS-FEC configuration registers</td>
</tr>
<tr>
<td></td>
<td>0x100000</td>
<td>Transceiver registers</td>
</tr>
<tr>
<td>1</td>
<td>0x200000</td>
<td>CPRI registers</td>
</tr>
<tr>
<td></td>
<td>0x300000</td>
<td>Transceiver registers</td>
</tr>
<tr>
<td>2</td>
<td>0x400000</td>
<td>CPRI registers</td>
</tr>
<tr>
<td></td>
<td>0x500000</td>
<td>Transceiver registers</td>
</tr>
<tr>
<td>3</td>
<td>0x600000</td>
<td>CPRI registers</td>
</tr>
<tr>
<td></td>
<td>0x700000</td>
<td>Transceiver registers</td>
</tr>
</tbody>
</table>
## 3.3. Document Revision History for the E-tile CPRI PHY Intel FPGA IP Design Example

<table>
<thead>
<tr>
<th>Document Version</th>
<th>Intel Quartus Prime Version</th>
<th>IP Version</th>
<th>Changes</th>
</tr>
</thead>
</table>
| 2019.08.07       | 19.2                        | 19.2.0     | • Added Figure: E-tile CPRI PHY Intel FPGA IP Simulation Block Diagram for 2.4/4.9/9.8 Line Rates.  
• Updated E-tile CPRI PHY Intel FPGA IP Core Hardware Design Examples High Level Block Diagram for new supported CPRI line rates.  
• Added the frequency value for the 2.4/4.8/9.8 Gbps line rates in section Compiling and Configuring the Design Example in Hardware.  
• Clarified i_clk_ref frequency value for different CPRI line rates. |
| 2019.05.17       | 19.1                        | 19.1       | Initial release. |
4. E-Tile Dynamic Reconfiguration Design Example

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 Stratix 10 E-Tile TX Transceiver Signal Integrity Development Kit.

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

<table>
<thead>
<tr>
<th>Dynamic Reconfiguration Protocol</th>
<th>Variant</th>
<th>Simulation</th>
<th>Hardware Design Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>10G/25G Ethernet Protocol</td>
<td>10G/25G with PTP and optional RS-FEC</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>10G/25G with optional RS-FEC</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CPRI</td>
<td>10G/24G CPRI with optional RS-FEC</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>9.8G CPRI</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>25G Ethernet to CPRI Protocol</td>
<td>25G with PTP and optional RS-FEC</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>100G Ethernet Protocol</td>
<td>100G Ethernet MAC+PCS with optional RS-FEC</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>File Names</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;design_example_dir&gt;/example_testbench/basic_avl_tb_top.sv</code></td>
<td>Top-level testbench file. The testbench instantiates the DUT and runs Verilog HDL tasks to generate and accept packets.</td>
</tr>
</tbody>
</table>

Testbench Scripts

<table>
<thead>
<tr>
<th>File Names</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;design_example_dir&gt;/example_testbench/mentor/run_vsim.do</code></td>
<td>The Mentor Graphics ModelSim script to run the testbench.</td>
</tr>
<tr>
<td><code>&lt;design_example_dir&gt;/example_testbench/synopsys/run_vcs.sh</code></td>
<td>The Synopsys VCS script to run the testbench.</td>
</tr>
<tr>
<td><code>&lt;design_example_dir&gt;/example_testbench/synopsys/run_vcsmx.sh</code></td>
<td>The Synopsys VCS MX script (combined Verilog HDL and SystemVerilog with VHDL) to run the testbench.</td>
</tr>
<tr>
<td><code>&lt;design_example_dir&gt;/example_testbench/run_ncsim.sh</code></td>
<td>The Cadence NCSim script to run the testbench.</td>
</tr>
<tr>
<td><code>&lt;design_example_dir&gt;/example_testbench/run_xcelium.sh</code></td>
<td>The Cadence Xcelium script to run the testbench.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>File Names</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;design_example_dir&gt;/hardware_test_design/alt_ehipc3.qpf</code></td>
<td>Intel Quartus Prime project file</td>
</tr>
<tr>
<td><code>&lt;design_example_dir&gt;/hardware_test_design/alt_ehipc3.qsf</code></td>
<td>Intel Quartus Prime project settings file</td>
</tr>
</tbody>
</table>

continued...
### 4. E-Tile Dynamic Reconfiguration Design Example

#### 4.1.2. Generating the Design

**Figure 35. Procedure**

![Diagram of the design generation procedure]

- **Start Parameter Editor**
- Specify IP Variation and Select Device
- Select Design Parameters
- Specify Example Design
- Initiate Design Generation

---

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

<table>
<thead>
<tr>
<th>File Names</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;design_example_dir&gt;/hardware_test_design/alt_ehipc3.sdc</td>
<td>Synopsys Design Constraints files. You can copy and modify these files for your own Intel Stratix 10 design.</td>
</tr>
<tr>
<td>&lt;design_example_dir&gt;/hardware_test_design/alt_ehipc3.sv</td>
<td>Top-level Verilog HDL design example file</td>
</tr>
<tr>
<td>&lt;design_example_dir&gt;/hardware_test_design/alt_ehipc3_hw.qpf</td>
<td>Intel Quartus Prime project file</td>
</tr>
<tr>
<td>&lt;design_example_dir&gt;/hardware_test_design/alt_ehipc3_hw.qsf</td>
<td>Intel Quartus Prime project settings file</td>
</tr>
<tr>
<td>&lt;design_example_dir&gt;/hardware_test_design/alt_ehipc3_hw.sdc</td>
<td>Synopsys Design Constraints files. You can copy and modify these files for your own Intel Stratix 10 design.</td>
</tr>
<tr>
<td>&lt;design_example_dir&gt;/hardware_test_design/alt_ehipc3_hw.v</td>
<td>Top-level Verilog HDL design example file</td>
</tr>
<tr>
<td>&lt;design_example_dir&gt;/hardware_test_design/common/</td>
<td>Hardware design example support files</td>
</tr>
</tbody>
</table>
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 Stratix 10** and select a device that meets all of these requirements:
   - Transceiver tile is E-tile
   - Transceiver speed grade is –1 or –2
   - Core speed grade is –1 or –2

3. Click **Finish**.

Follow these steps to generate the E-tile 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 **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 **Stratix 10 E-Tile TX Transceiver Signal Integrity Development Kit**, available for Intel Stratix 10 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**.

**Related Information**
Intel Stratix 10 TX Signal Integrity Development Kit Webpage

### 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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Select DR Protocol</strong></td>
<td>• 10G/25G Ethernet Protocol&lt;br&gt;• CPRI Protocol&lt;br&gt;• 25G Ethernet to CPRI Protocol&lt;br&gt;• 100G Ethernet</td>
<td>Available protocols for dynamic reconfiguration design example generation.</td>
</tr>
</tbody>
</table>

**Parameter Settings: 10G/25G Ethernet Protocol (This tab is only applicable when you select 10G/25G Ethernet Protocol)**

*continued...*
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Options</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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)**

| 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 | • Stratix 10 E-Tile 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.  
**Stratix 10 E-Tile TX Transceiver Signal Integrity Development Kit:** This option allows you to test the design example on the selected Intel FPGA IP development kit. The target device used is 1ST280EY2FS5E2VG. This option automatically selects the Target Device to match the device on the Intel FPGA IP development kit. If your board revision has a different device grade, you can change the target device.  
**Other Development Kits:** This option allows the design example to be tested on development kits other than 1ST280EY2FS5E2VG. 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

![Procedure Diagram]

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

<table>
<thead>
<tr>
<th>Simulator</th>
<th>Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mentor Graphics</td>
<td>In the command line, type <code>vsim -do run_vsim.do</code></td>
</tr>
<tr>
<td>ModelSim</td>
<td>If you prefer to simulate without bringing up the ModelSim GUI, type <code>vsim -c -do run_vsim.do</code></td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> 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.</td>
</tr>
<tr>
<td>Cadence NCSim</td>
<td>In the command line, type <code>sh run_ncsim.sh</code></td>
</tr>
<tr>
<td>Cadence Xcelium</td>
<td>In the command line, type <code>sh run_xcelium.sh</code></td>
</tr>
<tr>
<td>Synopsys VCS/VCS</td>
<td>In the command line, type <code>sh run_vcs.sh</code> or <code>sh run_vcsmx.sh</code></td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> <code>run_vcs.sh</code> 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 <code>run_vcsmx.sh</code>.</td>
</tr>
</tbody>
</table>

**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 
\(<\text{design_example_dir}/\text{hardware_test_design/software/dynamic_reconfiguration_simulation}>\) with the following C-code source files from the \(<\text{design_example_dir}/\text{software/dynamic_reconfiguration_sim}>\) 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 \(\text{dynamic_reconfiguration_simulation}\) and select Build Project. Ensure the \(\text{dynamic_reconfiguration_simulation.elf}\) file is generated in the new \(<\text{design_example_dir}/\text{hardware_test_design/software/dynamic_reconfiguration_simulation}>\) directory.

9. To generate a new HEX file, right-click the \(\text{dynamic_reconfiguration_simulation}\) 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 \(\text{mem_init_generate}\).

11. Click Build. The \(\text{mem_init_generate}\) creates the new HEX \((\text{nios_system_onchip_memory2_0_onchip_memory2_0.hex})\) file. The new HEX file resides in the \(<\text{design_example_dir}/\text{hardware_test_design/software/dynamic_reconfiguration_simulation/mem_init}>\) directory.

Follow these steps to simulate the testbench:

1. Open the \(<\text{simulator_name}>\_files.tcl\) script in the \(<\text{design_example_dir}/\text{example_testbench/setup_scripts/common}>\) directory.

2. Edit the TCL script to change the existing \(\text{nios_system_onchip_memory2_0_onchip_memory2_0.hex}\) 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:

```tcl
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:

```tcl
lappend memory_files *[normalize_path "$QSYS_SIMDIR/../\design_example_dir/\hardware_test_design/ip/nios_system/nios_system_onchip_memory2_0_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/`.  

4. Run the simulation script for the simulator. The script compiles and runs the testbench in the simulator. Refer to Table 27 on page 97.  

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).  
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 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_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 Stratix 10 device:
   a. On the Tools menu, click Programmer.
   b. In the Programmer, click Hardware Setup.
   c. Select a programming device.
   d. Select and add the Stratix 10 E-Tile 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 the E-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 Stratix 10 E-Tile 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 Stratix 10 E-Tile TX Transceiver Signal Integrity Development Kit on page 103 on how to configure the power management setting that can be included in the Quartus Setting File (.qsf) for the Stratix 10 E-Tile 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 Stratix 10 E-Tile TX Transceiver Signal Integrity Development Kit.

3. Launch the Clock Control application and set new frequencies for the design example. Below is the frequency setting in the Clock Control application. Select Si5341A(U3) to program:
   - OUT0 = 184.32 MHz
   - OUT5 = 153.6 MHz

   This step is only applicable for CPRI protocol and Ethernet to CPRI protocol related dynamic reconfiguration design examples.


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
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
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 Stratix 10 E-Tile TX Transceiver Signal Integrity Development Kit

If you select Stratix 10 E-Tile 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 1ST280EY2F55E2VG with the pin assignments provided in the .qsf file.

The Stratix 10 E-Tile TX Transceiver Signal Integrity Development Kit (1ST280EY2F55E2VG) 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 Stratix 10 E-Tile TX Transceiver Signal Integrity Development Kit:

```
set_global_assignment -name USE_PWRMGT_SCL SDM_IO14
set_global_assignment -name USE_PWRMGT_SDA SDM_IO11
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_DEVICE0_ADDRESS 47
set_global_assignment -name PWRMGT_SLAVE_DEVICE1_ADDRESS 48
set_global_assignment -name PWRMGT_SLAVE_DEVICE2_ADDRESS 00
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 Stratix 10 devices using the E-Tile Hard IP for Ethernet Intel FPGA IP core with the following variants:

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

<table>
<thead>
<tr>
<th>Base Operation</th>
<th>Dynamic Reconfiguration Variants</th>
</tr>
</thead>
<tbody>
<tr>
<td>25GE with RS-FEC and PTP</td>
<td>25GE with RS-FEC and PTP</td>
</tr>
<tr>
<td></td>
<td>25GE with PTP</td>
</tr>
<tr>
<td></td>
<td>10GE with PTP</td>
</tr>
<tr>
<td>25GE with RS-FEC</td>
<td>25GE with RS-FEC</td>
</tr>
<tr>
<td></td>
<td>25GE</td>
</tr>
<tr>
<td></td>
<td>10GE</td>
</tr>
</tbody>
</table>

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. **Stratix 10 E-Tile TX Transceiver Signal Integrity Development Kit** as the target development kit.
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 96.

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 99.
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.
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.
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
```

**Related Information**

- E-Tile Transceiver PHY User Guide
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. **Stratix 10 E-Tile TX Transceiver Signal Integrity Development Kit** as the target development kit.

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

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 96.

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 99.
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.
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.
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
```


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 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 \texttt{TEST\_MODE} parameter in the flow.c.

<table>
<thead>
<tr>
<th>TEST_MODE</th>
<th>Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Internal serial loopback without PMA adaptation</td>
</tr>
<tr>
<td>1</td>
<td>Internal serial loopback with PMA adaptation</td>
</tr>
<tr>
<td>2</td>
<td>External serial loopback with PMA adaptation</td>
</tr>
</tbody>
</table>

For speed switching to 24G, 12G, 10G, and 9.8G speed modes, setting \texttt{TEST\_MODE} to a non-zero value enables the general PMA adaptation. This PMA adaptation with zero effort configuration is used 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 \textit{E-Tile Transceiver PHY User Guide}.

Related Information

\textit{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 Hard IP for Ethernet Intel FPGA IP 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 Hard IP for Ethernet Intel FPGA IP 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 Hard IP for Ethernet Intel FPGA IP 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 Hard IP for Ethernet Intel FPGA IP 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 Hard IP for Ethernet Intel FPGA IP core.

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

CPU is alive!
By default, the starting mode is 25G_FEC.

Please choose one of Dynamic reconfiguration:
0) 25G_FEC   -> 25G_noFEC  -> 10G   -> 25G_FEC -> 25G_noFEC -> 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

<table>
<thead>
<tr>
<th>Signal</th>
<th>Direction</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>clk100</td>
<td>Input</td>
<td>Input clock for reconfiguration. Drive at 100 MHz. The intent is to drive this from a 100 MHz oscillator on the board.</td>
</tr>
<tr>
<td>cpu_resetn</td>
<td>Input</td>
<td>Global reset for Nios II system.</td>
</tr>
<tr>
<td>i_clk_ref</td>
<td>Input</td>
<td>Reference clock 25G IP core. Drive at 156.25MHz.</td>
</tr>
<tr>
<td>o_tx_serial</td>
<td>Output</td>
<td>Transmit serial data.</td>
</tr>
<tr>
<td>i_rx_serial</td>
<td>Input</td>
<td>Receiver serial data.</td>
</tr>
</tbody>
</table>

4.2.5. 10GE/25GE Design Examples Registers

Table 30. E-Tile Hard IP for Ethernet Intel FPGA IP Hardware Design Examples Register Map

<table>
<thead>
<tr>
<th>Word Offset</th>
<th>Register Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000000 - 0x000FFF</td>
<td>Ethernet MAC and PCS registers</td>
</tr>
<tr>
<td>0x001000 - 0x001FFF</td>
<td>Packet Generator and Checker registers</td>
</tr>
<tr>
<td>0x002000 - 0x002FFF</td>
<td>PTP monitoring registers</td>
</tr>
<tr>
<td>0x010000 - 0x0107FF</td>
<td>RS-FEC configuration registers</td>
</tr>
<tr>
<td>0x100000 - 0x1FFFFFF</td>
<td>Transceiver registers</td>
</tr>
</tbody>
</table>

(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.
### Table 31. Packet Client Registers

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

<table>
<thead>
<tr>
<th>Addr</th>
<th>Name</th>
<th>Bit</th>
<th>Description</th>
<th>HW Reset Value</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x1000</td>
<td>PKT_CL_SCRA TCH</td>
<td>[31:0]</td>
<td>Scratch register available for testing.</td>
<td>N/A</td>
<td>RW</td>
</tr>
<tr>
<td>0x1001</td>
<td>PKT_CL_CLNT</td>
<td>[31:0]</td>
<td>Four characters of IP block identification string CLNT.</td>
<td>N/A</td>
<td>RO</td>
</tr>
<tr>
<td>0x1008</td>
<td>Packet Size Configure</td>
<td>[29:0]</td>
<td>Specify the transmit packet size in bytes. These bits have dependencies to PKT_GEN_TX_CTRL register.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Bit[29:11]: Reserved.</td>
<td>0x25800040</td>
<td>RW</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Bit[10:0]: These bits specify the transmit packet size in bytes.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1009</td>
<td>Packet Number Control</td>
<td>[31:0]</td>
<td>Specify the number of packets to transmit from the packet generator.</td>
<td>0xA</td>
<td>RW</td>
</tr>
<tr>
<td>0x1010</td>
<td>PKT_GEN_TX_CTRL</td>
<td>[7:0]</td>
<td></td>
<td>0x6</td>
<td>RW</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Bit [0]: Reserved.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 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.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Bit [2]: Reserved.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 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.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Bit [5:4]:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 00: Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 01: Fixed mode</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 10: Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 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.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Bit [7]:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 1: For transmission without gap in between packets.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 0: For transmission with random gap in between packets.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1011</td>
<td>Destination address lower 32</td>
<td>[31:0]</td>
<td>Destination address (lower 32 bits).</td>
<td>0x56780ADD</td>
<td>RW</td>
</tr>
<tr>
<td></td>
<td>bits</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*continued...*
<table>
<thead>
<tr>
<th>Addr</th>
<th>Name</th>
<th>Bit</th>
<th>Description</th>
<th>HW Reset Value</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x1012</td>
<td>Destination address upper 16 bits</td>
<td>[15:0]</td>
<td>Destination address (upper 16 bits).</td>
<td>0x1234</td>
<td>RW</td>
</tr>
<tr>
<td>0x1013</td>
<td>Source address lower 32 bits</td>
<td>[31:0]</td>
<td>Source address (lower 32 bits).</td>
<td>0x43210ADD</td>
<td>RW</td>
</tr>
<tr>
<td>0x1014</td>
<td>Source address upper 16 bits</td>
<td>[15:0]</td>
<td>Source address (upper 16 bits).</td>
<td>0x8765</td>
<td>RW</td>
</tr>
</tbody>
</table>
4.3. 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 32. Supported Design Example Variants for CPRI Dynamic Reconfiguration

<table>
<thead>
<tr>
<th>Base Operation</th>
<th>Variants that Supports Dynamic Reconfiguration</th>
</tr>
</thead>
<tbody>
<tr>
<td>24G CPRI with RS-FEC</td>
<td>24G CPRI with RS-FEC</td>
</tr>
<tr>
<td></td>
<td>24G CPRI</td>
</tr>
<tr>
<td></td>
<td>12G CPRI with RS-FEC</td>
</tr>
<tr>
<td></td>
<td>12G CPRI</td>
</tr>
<tr>
<td></td>
<td>10G CPRI with RS-FEC</td>
</tr>
<tr>
<td></td>
<td>10G CPRI</td>
</tr>
<tr>
<td></td>
<td>9.8G CPRI</td>
</tr>
<tr>
<td></td>
<td>6G CPRI</td>
</tr>
<tr>
<td></td>
<td>4.9G CPRI</td>
</tr>
<tr>
<td></td>
<td>3G CPRI</td>
</tr>
<tr>
<td></td>
<td>2.4G CPRI</td>
</tr>
<tr>
<td>9.8G CPRI</td>
<td>9.8G CPRI</td>
</tr>
<tr>
<td></td>
<td>6G CPRI</td>
</tr>
<tr>
<td></td>
<td>4.9G CPRI</td>
</tr>
<tr>
<td></td>
<td>3G CPRI</td>
</tr>
<tr>
<td></td>
<td>2.4G CPRI</td>
</tr>
</tbody>
</table>

4.3.1. Functional Description

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

Figure 42. Block Diagram—CPRI Dynamic Reconfiguration Design Example

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

Figure 43. Clocking Scheme for 24G CPRI with RS-FEC Dynamic Reconfiguration Design Example

Figure 44. Clocking Scheme for 9.8G CPRI Dynamic Reconfiguration Design Example
4.3.2. Simulation Design Examples

4.3.2.1. 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. **Stratix 10 E-Tile TX Transceiver Signal Integrity Development Kit** as the target development kit.

**Figure 45. Simulation Block Diagram for 24G CPRI PHY with RS-FEC Dynamic Reconfiguration Design Example**

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 96.

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.


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 (_clk_ref[0]) to 153.6 MHz (_clk_ref[1]).
   c. Switch the PMA controller clock to the transceiver refclk0 clock.

   Note: Steps 3a and 3c 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.


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
```

Related Information

E-tile Hard IP User Guide: E-Tile Hard IP for Ethernet and E-Tile CPRI PHY Intel FPGA IPs

4.3.2.2. 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. Stratix 10 E-Tile TX Transceiver Signal Integrity Development Kit as
      the target development kit.
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 96.

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.


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.


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
```

### 4.3.3. 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 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.

<table>
<thead>
<tr>
<th>TEST_MODE</th>
<th>Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Internal serial loopback without PMA adaptation</td>
</tr>
<tr>
<td>1</td>
<td>Internal serial loopback with PMA adaptation</td>
</tr>
<tr>
<td>2</td>
<td>External serial loopback with PMA adaptation</td>
</tr>
</tbody>
</table>
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 is used 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.

Related Information
E-Tile Transceiver PHY User Guide

4.3.3.1. 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 Hard IP for Ethernet Intel FPGA IP 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

*Terminate test* -> If you terminate test halfway, you must reload the .sof file before retrigger the hardware test.

Enter a Valid Selection:

### 4.3.4. 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 33. CPRI Hardware Dynamic Reconfiguration Design Example Interface Signals**

<table>
<thead>
<tr>
<th>Signal</th>
<th>Direction</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>clk100</td>
<td>Input</td>
<td>Input clock for reconfiguration. Drive at 100 MHz. The intent is to drive this from a 100 MHz oscillator on the board.</td>
</tr>
<tr>
<td>cpu_resetn</td>
<td>Input</td>
<td>Global reset for Nios II system.</td>
</tr>
<tr>
<td>i_clk_ref (5)</td>
<td>Input</td>
<td>156.25 MHz input clock for channel PLL.</td>
</tr>
<tr>
<td>tx_serial_data/_n</td>
<td>Output</td>
<td>Transmit serial data for channel PLL (PMA direct mode).</td>
</tr>
<tr>
<td>rx_serial_data/_n</td>
<td>Input</td>
<td>Receiver serial data for channel PLL (PMA direct mode).</td>
</tr>
<tr>
<td>i_clk_ref_cpri[1:0]</td>
<td>Input</td>
<td>Input clock for CPRI IP core. In 24G CPRI IP:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• [0]: 184.32MHz for high speed mode for 10G/25G Ethernet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• [1]: 153.6MHz for PMA direct low speed mode for 9.8G CPRI, 6G CPRI, 4.9G CPRI, 3G CPRI, and 2.4G CPRI</td>
</tr>
<tr>
<td>o_tx_serial</td>
<td>Output</td>
<td>Transmit serial data</td>
</tr>
<tr>
<td>i_rx_serial</td>
<td>Input</td>
<td>Receiver serial data</td>
</tr>
</tbody>
</table>

(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.
4.3.5. CPRI Design Example Registers

Table 34. E-tile CPRI PHY Intel FPGA IP Hardware Design Example Register Map

<table>
<thead>
<tr>
<th>Word Offset</th>
<th>Register Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000000 – 0x000FFF</td>
<td>CPRI PCS registers</td>
</tr>
<tr>
<td>0x010000 – 0x0107FF</td>
<td>RS-FEC configuration registers</td>
</tr>
<tr>
<td>0x100000 – 0x1FFFFFF</td>
<td>Transceiver registers</td>
</tr>
</tbody>
</table>
4.4. 25G Ethernet to CPRI Dynamic Reconfiguration Design Example

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

Table 35. Supported Design Example Variants for 25G Ethernet to CPRI Dynamic Reconfiguration

<table>
<thead>
<tr>
<th>Base Operation</th>
<th>Variants that Supports Dynamic Reconfiguration</th>
</tr>
</thead>
<tbody>
<tr>
<td>25GE with RS-FEC and PTP</td>
<td>25GE with RS-FEC and PTP</td>
</tr>
<tr>
<td></td>
<td>24GE CPRI with RS-FEC</td>
</tr>
<tr>
<td></td>
<td>10GE CPRI</td>
</tr>
<tr>
<td></td>
<td>9.8GE CPRI</td>
</tr>
<tr>
<td></td>
<td>4.9GE CPRI</td>
</tr>
<tr>
<td></td>
<td>2.4GE CPRI</td>
</tr>
</tbody>
</table>

4.4.1. Functional Description

4.4.1.1. Clocking Scheme

Figure 47. Clocking Scheme 25G Ethernet to CPRI Dynamic Reconfiguration Design Example

4.4.1.2. Reset

Nios system controls the resets implemented in the design example via PIO. To reset the design under test (DUT) IP, always deassert the i_sl_csr_rst_n first before the sl_tx/rx_rst_n and i_reconfig_reset signals. When performing dynamic reconfiguration, i_sl_csr_rsn_n and i_reconfig_reset should not be toggled.
4.4.2. Simulation Design Examples

4.4.2.1. 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. **Stratix 10 E-Tile TX Transceiver Signal Integrity Development Kit** as the target development kit.

**Figure 48. Simulation Block Diagram for 25G Ethernet to CPRI Dynamic Reconfiguration Design Example**

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 96.

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 \texttt{sl\_tx\_rst\_n} and \texttt{sl\_rx\_rst\_n} reset signals.

2. Link Initialization. For more information, refer to Performing the Link Initialization on page 99.

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 \texttt{sl\_tx\_rst\_n} and \texttt{sl\_rx\_rst\_n} reset signals.


3. Perform reference clock mux switching. For more information about the details of the changed register values, refer to the \texttt{c3\_reconfig.c} file.
   a. Switch the PMA controller clock to the transceiver \texttt{refclk1} clock.
   b. Change \texttt{refclk} reference clock from the original speed mode clock to the destination speed mode clock.
      
      \textit{Note:} For information on speed mode clocks, refer to 25G Ethernet to CPRI Design Example Interface Signals on page 130.

   c. Switch the PMA controller clock to the transceiver \texttt{refclk0} clock.
      
      \textit{Note:} Steps 3a and 3c 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.


Related Information

• E-tile Hard IP User Guide: E-Tile Hard IP for Ethernet and E-Tile CPRI PHY Intel FPGA IPs
• E-Tile Transceiver PHY User Guide

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 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.

<table>
<thead>
<tr>
<th>TEST_MODE</th>
<th>Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Internal serial loopback without PMA adaptation</td>
</tr>
<tr>
<td>1</td>
<td>Internal serial loopback with PMA adaptation</td>
</tr>
<tr>
<td>2</td>
<td>External serial loopback with PMA adaptation</td>
</tr>
</tbody>
</table>

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 is used 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.

Related Information

E-Tile Transceiver PHY User Guide

4.4.3.1. 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 Hard IP for Ethernet Intel FPGA IP core.
• Client 10G/25G 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 8B/10B Pattern Generator and Checker that coordinates the programming of the IP core and packet generation.
• Roundtrip counter to measure the total roundtrip delay value via the DUT.
• IO PLL to provide sampling clock, 250 MHz for DUT and the Roundtrip counter.
• 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 Hard IP for Ethernet Intel FPGA IP through the tool.

When switching from Ethernet to CPRI protocol mode, the CPRI PHY Soft wrapper module reads the TX and RX datapath latency and round-trip counter values and displays them in the dynamic reconfiguration hardware test. For more information about CPRI PHY registers, refer to the *E-tile Hard IP User Guide: E-Tile CPRI PHY Intel FPGA IP* section.

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_RSFEC.
Please choose one of Dynamic reconfiguration:
  0) 25G_PTP_RSFEC  -> 10G_PTP  -> 25G_PTP_RSFEC  -> CPRI_24G_RSFEC  ->
  25G_PTP_RSFEC  -> CPRI_10G  -> 25G_PTP_RSFEC  -> CPRI_9p8G  -> 25G_PTP_RSFEC  ->
  CPRI_4p9G  -> 25G_PTP_RSFEC  -> CPRI_2p4G  -> 25G_PTP_RSFEC
  1) 25G_PTP_RSFEC  -> CPRI_24G_RSFEC
  2) CPRI_24G_RSFEC  -> 25G_PTP_RSFEC
  3) 25G_PTP_RSFEC  -> CPRI_10G
  4) CPRI_10G  -> 25G_PTP_RSFEC
  5) 25G_PTP_RSFEC  -> CPRI_9p8G
  6) CPRI_9p8G  -> 25G_PTP_RSFEC
  7) 25G_PTP_RSFEC  -> CPRI_4p9G
  8) CPRI_4p9G  -> 25G_PTP_RSFEC
  9) 25G_PTP_RSFEC  -> CPRI_2p4G
     a) CPRI_2p4G  -> 25G_PTP_RSFEC
     b) 25G_PTP_RSFEC  -> 10G_PTP
     c) 10G_PTP  -> 25G_PTP_RSFEC
     d) Terminate test

If you terminate the test halfway, you must reload the .sof file before retry the hardware test.

Enter a Valid Selection (0,1,3,5,7,9,b,d):
0
You entered: 0. Execute five pairs of dynamic reconfiguration.

INFO: Dynamic reconfiguration: 25G_PTP_RSFEC  -> 10G_PTP

INFO: End of dynamic reconfiguration: 25G_PTP_RSFEC  -> 10G_PTP
INFO: PKT_RX_CNT received = 20

INFO: Dynamic reconfiguration: 25G_PTP_RSFEC  -> CPRI_24G_RSFEC

INFO: End of dynamic reconfiguration: 25G_PTP_RSFEC  -> CPRI_24G_RSFEC

INFO: tx_delay info: 26fb
INFO: rx_delay info: 7908
INFO: 24G with FEC rx bitslip value info: 4c
INFO: Total Output RT count value (sample sizes: hardware: 128, sim: 32): 5115
Test Pass!
```
4.4.4. 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>
<thead>
<tr>
<th>Signal</th>
<th>Direction</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>clk100</td>
<td>Input</td>
<td>Input clock for reconfiguration. Drive at 100 MHz. The intent is to drive this from a 100 MHz oscillator on the board.</td>
</tr>
<tr>
<td>cpu_resetn</td>
<td>Input</td>
<td>Input reset for Nios II System.</td>
</tr>
<tr>
<td>ref_clk156MHz</td>
<td>Input</td>
<td>156.25 MHz input clock for the 25G Ethernet IP core. Connect to i_clk_ref[0] in 25G Ethernet IP core.</td>
</tr>
<tr>
<td>ref_clk184MHz</td>
<td>Input</td>
<td>184.32 MHz input clock for the 10G/24G CPRI mode. Connect to the i_clk_ref[1] in 25G Ethernet IP core.</td>
</tr>
<tr>
<td>ref_clk153MHz</td>
<td>Input</td>
<td>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.</td>
</tr>
<tr>
<td>tx_serial_data_/n</td>
<td>Output</td>
<td>Transmit serial data for channel PLL (PMA direct mode).</td>
</tr>
<tr>
<td>rx_serial_data_/n</td>
<td>Input</td>
<td>Receiver serial data for channel PLL (PMA direct mode).</td>
</tr>
<tr>
<td>o_tx_serial</td>
<td>Output</td>
<td>Transmit serial data.</td>
</tr>
<tr>
<td>i_rx_serial</td>
<td>Input</td>
<td>Receiver serial data.</td>
</tr>
</tbody>
</table>

4.4.5. 25G Ethernet to CPRI Design Examples Registers

<table>
<thead>
<tr>
<th>Word Offset</th>
<th>Register Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000000 – 0x000FFF</td>
<td>Ethernet MAC and PCS registers</td>
</tr>
<tr>
<td>0x001000 – 0x001FFF</td>
<td>Packet Generator and Checker registers</td>
</tr>
<tr>
<td>0x002000 – 0x002FFF</td>
<td>PTP monitoring registers</td>
</tr>
<tr>
<td>0x010000 – 0x0107FF</td>
<td>RS-FEC configuration registers</td>
</tr>
<tr>
<td>0x100000 – 0x1FFFFFF</td>
<td>Transceiver registers</td>
</tr>
<tr>
<td>0x003000 – 0x003FFF</td>
<td>CPRI PHY soft registers</td>
</tr>
</tbody>
</table>
You can customize the E-Tile Hard IP for Ethernet to CPRI Intel FPGA IP hardware design example by programming the packet client registers.

<table>
<thead>
<tr>
<th>Addr</th>
<th>Name</th>
<th>Bit</th>
<th>Description</th>
<th>HW Reset Value</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x1000</td>
<td>PKT_CL_SCRATCH TCH</td>
<td>[31:0]</td>
<td>Scratch register available for testing.</td>
<td>N/A</td>
<td>RW</td>
</tr>
<tr>
<td>0x1001</td>
<td>PKT_CL_CLNT</td>
<td>[31:0]</td>
<td>Four characters of IP block identification string CLNT.</td>
<td>N/A</td>
<td>RO</td>
</tr>
</tbody>
</table>
| 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: 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     |

continued...
<table>
<thead>
<tr>
<th>Addr</th>
<th>Name</th>
<th>Bit</th>
<th>Description</th>
<th>HW Reset Value</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x1012</td>
<td>Destination address upper 16 bits</td>
<td>[15:0]</td>
<td>Destination address (upper 16 bits).</td>
<td>0x1234</td>
<td>RW</td>
</tr>
<tr>
<td>0x1013</td>
<td>Source address lower 32 bits</td>
<td>[31:0]</td>
<td>Source address (lower 32 bits).</td>
<td>0x43210ADD</td>
<td>RW</td>
</tr>
<tr>
<td>0x1014</td>
<td>Source address upper 16 bits</td>
<td>[15:0]</td>
<td>Source address (upper 16 bits).</td>
<td>0x8765</td>
<td>RW</td>
</tr>
</tbody>
</table>

Table 39. CPRI PHY Soft Registers

You can view the status ports and TX and RX datapath latencies of the CPRI PHY soft module by accessing the CPRI PHY registers.

<table>
<thead>
<tr>
<th>Addr</th>
<th>Name</th>
<th>Bit</th>
<th>Description</th>
<th>HW Reset Value</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x3000</td>
<td>rx_bitslip boundary_sel</td>
<td>[9:5]</td>
<td>Indicates the number of bits that the 8B/10B RX PCS block slipped to achieve a deterministic latency.</td>
<td>0x0</td>
<td>RO</td>
</tr>
<tr>
<td></td>
<td>cpri_fec_en</td>
<td>[4]</td>
<td>Used by deterministic latency, this bit indicates whether the RS-FEC block is enabled.</td>
<td>0x1</td>
<td>RW</td>
</tr>
<tr>
<td></td>
<td>cpri_rate_sel</td>
<td>[3:0]</td>
<td>Used by EFIFO and deterministic latency, this bit indicates the CPRI PHY speed selection.</td>
<td>0xB</td>
<td>RW</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bit [3:0]:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 ~ 1: Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2: 2.4G</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 3: 3G</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 4: 4.9G</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 5: 6G</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 6: 9.8G</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 7 ~ 8: Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 9: 10G</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 10: 12G</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 11: 24G</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 12 ~ 15: Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Note: The TX/RX datapath must be reconfigured after every setting change.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x3001</td>
<td>measure_valid</td>
<td>[0]</td>
<td>Indicates whether the deterministic latency measurement values are valid.</td>
<td>0x0</td>
<td>RO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0: Not valid</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1: Valid</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x3002</td>
<td>tx_delay</td>
<td>[20:0]</td>
<td>TX Datapath Latency</td>
<td>0x0</td>
<td>RO</td>
</tr>
</tbody>
</table>

Displays the TX datapath deterministic latency measurement values measured in sampling_clk cycles.
<table>
<thead>
<tr>
<th>Addr</th>
<th>Name</th>
<th>Bit</th>
<th>Description</th>
<th>HW Reset Value</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>measure_valid must be set prior taking the measurement.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x3003</td>
<td>rx_delay</td>
<td>[20:0]</td>
<td>RX Datapath Latency Displays the RX datapath deterministic latency measurement values measured in sampling_clk cycles. measure_valid must be set prior taking the measurement.</td>
<td>0x0</td>
<td>RO</td>
</tr>
</tbody>
</table>


4.5. 100G Ethernet Dynamic Reconfiguration Design Example

The 100G Ethernet E-Tile Dynamic Reconfiguration Design Example demonstrates a dynamic reconfiguration solution for Intel Stratix 10 devices using the E-Tile Hard IP for Ethernet Intel FPGA IP 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 40. List of Supported Design Example Variants for 100G Ethernet Dynamic Reconfiguration

<table>
<thead>
<tr>
<th>Base Operation</th>
<th>Dynamic Reconfiguration Variants</th>
</tr>
</thead>
<tbody>
<tr>
<td>100GE MAC + PCS with RS-FEC</td>
<td>100G MAC + PCS with RS-FEC</td>
</tr>
<tr>
<td></td>
<td>100G MAC + PCS</td>
</tr>
<tr>
<td></td>
<td>4x25G MAC + PCS with RS-FEC</td>
</tr>
<tr>
<td></td>
<td>4x25G MAC + PCS</td>
</tr>
</tbody>
</table>

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.

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 41. 100G Ethernet Dynamic Reconfiguration Hardware Design Example Functions

<table>
<thead>
<tr>
<th>Command Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>start_random_pkt_gen_4ch</td>
<td>Starts the packet generator in a random size mode for all four channel lanes. Example: %start_random_pkt_gen_4ch</td>
</tr>
<tr>
<td>stop_pkt_gen_4ch</td>
<td>Immediately stops the packet generator for all four channel lanes.</td>
</tr>
<tr>
<td>chkmac_stats $ch</td>
<td>Checks the mac stats counter for the specified channel.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>• In 100GE mode: %chkmac_stats</td>
</tr>
<tr>
<td></td>
<td>• In 25GE mode: $chkmac_stats 2 for lane 2</td>
</tr>
<tr>
<td>run_test_dr</td>
<td>Switches between all available modes and performs the traffic test for each reconfiguration. In 25GE mode, performs four traffic tests, one per each lane.</td>
</tr>
<tr>
<td>run_test_dr_sw</td>
<td>Switches to a specified mode and performs the traffic test in a loopback mode.</td>
</tr>
<tr>
<td>dr_calib_switch $mode_curr $mode_target</td>
<td>Reconfigures to a different mode based on the configuration and a $mode_target variable. Performs the PMA adaptation for the specific mode. $mode_target options:</td>
</tr>
<tr>
<td></td>
<td>• 100G_fec</td>
</tr>
<tr>
<td></td>
<td>• 100G_nofec</td>
</tr>
<tr>
<td></td>
<td>• 4x25G_fec</td>
</tr>
<tr>
<td></td>
<td>• 4x25G_nofec</td>
</tr>
<tr>
<td></td>
<td>$mode_curr variable supports all target modes.</td>
</tr>
<tr>
<td></td>
<td>Note: $mode_curr is not a required parameter.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>• In 100GE mode, use this command to switch to 4x25GE:</td>
</tr>
<tr>
<td></td>
<td>dr_calib_switch 0 &quot;100g_fec&quot; &quot;4x25G_nofec&quot;</td>
</tr>
<tr>
<td></td>
<td>• In 25GE mode, use this command to switch to 4x25GE:</td>
</tr>
<tr>
<td></td>
<td>dr_calib_switch 0 &quot;&quot; &quot;4x25G_nofec&quot; as $mode_curr isn't required.</td>
</tr>
<tr>
<td>dr_reset</td>
<td>Resets all signals except the PMA and E-tile Hard IP for Ethernet CSRs.</td>
</tr>
</tbody>
</table>

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 42. Reset sequence assertion

This table illustrates $\text{dr\_reset[3:0]}$ assertion sequence.

<table>
<thead>
<tr>
<th>Assertion Sequence</th>
<th>$\text{dr_reset[3:0]}$={Channel3, Channel2, Channel1, Channel0}</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Channel 3</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>

### Table 43. Reset sequence deassertion

This table illustrates $\text{dr\_reset[3:0]}$ deassertion sequence.

<table>
<thead>
<tr>
<th>Assertion Sequence</th>
<th>$\text{dr_reset[3:0]}$={Channel3, Channel2, Channel1, Channel0}</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Channel 3</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
</tr>
</tbody>
</table>

### 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. **Stratix 10 E-Tile TX Transceiver Signal Integrity Development Kit** as the target development kit.
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.... 407396143
# waiting for rx_transfer_ready.... 407716015
# RX transfer ready is 1 at time 418791583
# 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.... 470536467
# EHIP RX reset out is 0 at time 472496000
```
Sending Packet 1...

** Received Packet 10...

DR -> 100G NoFEC

** DR STARTING

--- wrotevdata = 00000000
--- wrotevdata = 00000020
--- wrotevdata = 00010000
--- wrotevdata = 00000001

** RECONFIG CALLED, WAITING FOR DR

--- MATCH!  ReaddataValid = 1 Readdata = 00000001 Expected_Readdata = 00000001
--- AVMM READ MISMATCH!  ReaddataValid = 1 Readdata = 00000000 Expected_Readdata = 00000000
--- MATCH!  ReaddataValid = 1 Readdata = 00000000 Expected_Readdata = 00000000

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 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 EIHP 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 alignment locked
** Sending Packet 1...
** Received Packet 10...
** Testbench complete.

********************************************************************************
Related Information
Simulating the E-Tile Hard IP for Ethernet Intel FPGA IP Design Example Testbench on page 11

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 programming 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

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:

```tcl
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_write $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:

```plaintext
% 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                0x0000fffff
     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?         0x00000000
     Rx AM LOCK Condition?         0x00000000
     Rx Lanes Deskewed Condition?  0x00000000
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?         0x00000000
     Rx AM LOCK Condition?         0x00000000
```

4. E-Tile Dynamic Reconfiguration Design Example
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 Frame : 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 Frame : 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
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>
<thead>
<tr>
<th>Signal</th>
<th>Direction</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>clk100</td>
<td>Input</td>
<td>Input clock for reconfiguration. Drive at 100 MHz. The intent is to drive this from a 100 Mhz oscillator on the board.</td>
</tr>
<tr>
<td>cpu_resetn</td>
<td>Input</td>
<td>Input reset for the dynamic reconfiguration controller.</td>
</tr>
<tr>
<td>i_csr_rst_n</td>
<td></td>
<td>Resets the entire IP core.</td>
</tr>
<tr>
<td>refclk</td>
<td>Input</td>
<td>156.25 MHz clock for the 100G Ethernet IP core.</td>
</tr>
<tr>
<td>o_tx_serial</td>
<td>Output</td>
<td>Transmit serial data.</td>
</tr>
<tr>
<td>i_rx_serial</td>
<td>Input</td>
<td>Receiver serial data.</td>
</tr>
</tbody>
</table>

4.5.6. 100G Ethernet Dynamic Reconfiguration Examples Registers

<table>
<thead>
<tr>
<th>Channel Number</th>
<th>Word Offset</th>
<th>Register Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0x100000</td>
<td>Transceiver registers</td>
</tr>
<tr>
<td></td>
<td>0x000000</td>
<td>10G/25G Ethernet registers</td>
</tr>
<tr>
<td></td>
<td>0x010000</td>
<td>RS-FEC configuration registers</td>
</tr>
<tr>
<td></td>
<td>0x004000</td>
<td>100G Ethernet registers</td>
</tr>
<tr>
<td></td>
<td>0x005000</td>
<td>100G Packet Client and Packet Generator registers</td>
</tr>
<tr>
<td></td>
<td>0x001000</td>
<td>10G/25G Packet client and Packet Generator registers</td>
</tr>
<tr>
<td>1</td>
<td>0x300000</td>
<td>Transceiver registers</td>
</tr>
</tbody>
</table>

continued...
<table>
<thead>
<tr>
<th>Channel Number</th>
<th>Word Offset</th>
<th>Register Type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0x200000</td>
<td>10G/25G Ethernet registers</td>
</tr>
<tr>
<td></td>
<td>0x201000</td>
<td>10G/25G Packet Client and Packet Generator registers</td>
</tr>
<tr>
<td>2</td>
<td>0x500000</td>
<td>Transceiver registers</td>
</tr>
<tr>
<td></td>
<td>0x400000</td>
<td>10G/25G Ethernet registers</td>
</tr>
<tr>
<td></td>
<td>0x401000</td>
<td>10G/25G Packet Client and Packet Generator registers</td>
</tr>
<tr>
<td>3</td>
<td>0x700000</td>
<td>Transceiver registers</td>
</tr>
<tr>
<td></td>
<td>0x600000</td>
<td>10G/25G Ethernet registers</td>
</tr>
<tr>
<td></td>
<td>0x601000</td>
<td>10G/25G Packet Client and Packet Generator registers</td>
</tr>
</tbody>
</table>

Table 46. **100G Ethernet Dynamic Reconfiguration Registers**

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

<table>
<thead>
<tr>
<th>Addr</th>
<th>Name</th>
<th>Bit</th>
<th>Description</th>
<th>HW Reset Value</th>
<th>Access</th>
</tr>
</thead>
</table>
| 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_channel | [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_channel  | [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_channel  | [3:0]    | Ethernet channel rate  
• [0]: Selects 25G/100G | 0x0            | RW     |
4.6. Document Revision History for the E-tile Dynamic Reconfiguration Design Example

<table>
<thead>
<tr>
<th>Document Version</th>
<th>Intel Quartus Prime Version</th>
<th>IP Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2020.04.13</td>
<td>20.1</td>
<td>20.1.0</td>
<td>• Added support for deterministic latency feature in the 25G Ethernet to CPRI protocol.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>— Updated the Clocking Scheme for 24G CPRI with RS-FEC Dynamic Reconfiguration Design Example figure.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>— Updated the Simulation Block Diagram for 25G Ethernet to CPRI Dynamic Reconfiguration Design Example figure.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>— Updated components described in the 25GE MAC+PCS with RS-FEC and PTP to CPRI Hardware Dynamic Reconfiguration Design Example Components section.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>— Added CPRI PHY soft registers table in the 25G Ethernet to CPRI Design Examples Registers section.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Updated Clock Control frequency setting in the Testing the E-tile Dynamic Reconfiguration Hardware Design Example section. This setting applies to CPRI and Ethernet to CPRI protocols.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Updated TEST_MODE selection in the 10GE/25GE MAC+PCS with RS-FEC Simulation Dynamic Reconfiguration Design Example Components section. TEST_MODE options are 0, 1, and 2.</td>
</tr>
<tr>
<td>2019.12.23</td>
<td>19.4</td>
<td>19.4.0</td>
<td>• Added simulation, compilation, and timing support for Intel Stratix 10 dynamic reconfiguration design examples:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>— 9.8G CPRI with direct PMA</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>— 100G Ethernet</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Restructured topics to improve the content flow.</td>
</tr>
<tr>
<td>2019.09.30</td>
<td>19.3</td>
<td>19.3.0</td>
<td>• Added List of Supported Dynamic Reconfiguration Design Example variants table.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Added a note to clarify run_vcs.sh and run_vcsmx.sh usage in the Steps to Simulate the Testbench table.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Updated Running the Design Example in Hardware section:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>— Added the board control configuration step</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>— Added screenshot of a successful hardware test</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Updated Power Management Setting for Intel Stratix 10 E-tile TX Transceiver Signal Integrity Development Kit section.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Added Note: i_channel_PLL is E-tile Transceiver PHY specific signal that utilizes additional transceiver E-tile channel, in the Clocking Scheme for 10G/25GE MAC+PCS with RS-FEC Dynamic Reconfiguration Design Example figure.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Updated test run sequence in the 10GE/25GE MAC+PCS with RS-FEC Simulation Dynamic Reconfiguration Design Example Components section.</td>
</tr>
</tbody>
</table>

continued...
<table>
<thead>
<tr>
<th>Document Version</th>
<th>Intel Quartus Prime Version</th>
<th>IP Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Replaced clk_100 with clk100 in clocking scheme figures:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>— Clocking Scheme for 10G/25GE MAC+PCS with RS-FEC and PTP Dynamic Reconfiguration Design Example</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>— Clocking Scheme for 10G/25GE MAC+PCS with RS-FEC Dynamic Reconfiguration Design Example</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>— Clocking Scheme for CPRI with RS-FEC Dynamic Reconfiguration Design Example</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Added the Calibration Flow for the Hardware Test Script section for all dynamic reconfiguration variants.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Added screenshot of the Dynamic Reconfiguration Hardware test for all dynamic reconfiguration variants.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Updated Dynamic Reconfiguration Design Example Components sections to include the following steps in the DR tests. Updates are applicable to all DR variants.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>— Disabling SERDES.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>— Triggering PMA analog reset.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>— Adjusting the phase offset of a recovered clock.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>— Enabling SERDES.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Added variants: 24G CPRI, 9.8G CPRI, 4.9G CPRI, and 2.4G CPRI in CPRI Dynamic Reconfiguration Design example.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Added 25G Ethernet to CPRI Dynamic Reconfiguration simulation, compilation-only project, and hardware design examples.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>— Added flow.c in software/dynamic_reconfiguration_hardware.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>— Added c3_function.c file in software/dynamic_reconfiguration_hardware and software/dynamic_reconfiguration_sim.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>— Renamed c3_config.c file with c3_reconfig.c in software/dynamic_reconfiguration_hardware and software/dynamic_reconfiguration_sim.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>— Renamed c3_config.h file with c3_reconfig.h file in software/dynamic_reconfiguration_hardware and software/dynamic_reconfiguration_sim.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>— Added cadence and xcelium folders in example_testbench.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>— Renamed eth_25g_pma_direct folder with eth_25g_channel_pll in hardware_test_design.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>— Added reset_release folder in hardware_test_design.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>— Added reset_release.ip file in the hardware_test_design.</td>
</tr>
</tbody>
</table>

continued...
4. E-Tile Dynamic Reconfiguration Design Example

UG-20172 | 2020.04.13

<table>
<thead>
<tr>
<th>Document Version</th>
<th>Intel Quartus Prime Version</th>
<th>IP Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2019.05.17</td>
<td>19.1</td>
<td>19.1</td>
<td>Initial release.</td>
</tr>
</tbody>
</table>

- **Added** `c3_function.c` and `flow.c` functions to `dynamic_reconfiguration_sim` and `dynamic_reconfiguration_hardware` directories.
- **Added** Ethernet to CPRI Protocol in the E-Tile Dynamic Reconfiguration Design Example: Generating the Design section.
- **Added** Xcelium and NCSim simulators support.
- **Renamed** 10G/24G CPRI with CPRI globally.
- **Added** `cpu_resetn` and updated `i_clk_ref` signal’s description in the 10GE/25GE Design Example Interface Signals section.
- **Added** test cases in the CPRI Protocol reconfiguration flow section.
  - 24G CPRI with RS-FEC to 24G CPRI without RS-FEC
  - 24G CPRI without RS-FEC to 10G CPRI
  - 10G CPRI to 24G CPRI without RS-FEC
  - 24G CPRI without RS-FEC to 24G CPRI with RS-FEC
  - 10G CPRI to 9.8G CPRI
  - 9.8G CPRI to 4.9G CPRI
  - 4.9G CPRI to 2.4G CPRI
- **Updated** `phy_ref_clk` to `phy_ref_clk[1:0]` signal in the Simulation Block Diagram for CPRI PHY with RS-FEC Dynamic Reconfiguration Design Example figure.
- **Updated** bit size to `i_clk_ref_cpri[1:0]` in the CPRI Hardware Dynamic Reconfiguration Design Example Interface Signals table.
5. E-tile Hard IP Intel Stratix 10 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.

<table>
<thead>
<tr>
<th>Intel Quartus Prime Version</th>
<th>User Guide</th>
</tr>
</thead>
<tbody>
<tr>
<td>19.4</td>
<td>E-tile Hard IP Intel Stratix 10 Design Examples User Guide</td>
</tr>
<tr>
<td>19.3</td>
<td>E-tile Hard IP Intel Stratix 10 Design Examples User Guide</td>
</tr>
<tr>
<td>19.2</td>
<td>E-tile Hard IP Intel Stratix 10 Design Examples User Guide</td>
</tr>
<tr>
<td>19.1</td>
<td>E-tile Hard IP Intel Stratix 10 Design Examples User Guide</td>
</tr>
<tr>
<td>18.1.1</td>
<td>E-tile Hard IP for Ethernet Intel Stratix 10 FPGA IP Design Example User Guide</td>
</tr>
<tr>
<td>18.0</td>
<td>E-tile Hard IP for Ethernet Intel Stratix 10 FPGA IP Design Example User Guide</td>
</tr>
</tbody>
</table>