# Contents


2. E-tile Ethernet IP for Intel Agilex FPGA Design Example ....................................................... 4  
   2.1. Quick Start Guide ............................................................................................................... 4  
      2.1.1. Directory Structure ................................................................................................. 5  
      2.1.2. Generating the Design ............................................................................................ 8  
      2.1.3. Simulating the E-tile Ethernet IP for Intel Agilex FPGA Design Example Testbench ................................................................................................................................. 9  
      2.1.4. Compiling the Compilation-Only Project .................................................................. 10  
   2.2. 10GE/25GE with Optional RS-FEC Design Examples ...................................................... 10  
      2.2.1. Simulation Design Examples ................................................................................. 11  
   2.3. 100GE with Optional RS-FEC Design Example ................................................................ 19  
      2.3.1. Simulation Design Examples ................................................................................. 19  

3. E-Tile Dynamic Reconfiguration Design Example .................................................................. 32  
   3.1. Quick Start Guide ............................................................................................................ 32  
      3.1.1. Directory Structure ................................................................................................. 32  
      3.1.2. Generating the Design ............................................................................................ 34  
      3.1.3. Simulating the E-tile Dynamic Reconfiguration Design Example Testbench ........... 36  
   3.2. 10G/25G Ethernet Dynamic Reconfiguration Design Examples ........................................ 39  
      3.2.1. Functional Description .......................................................................................... 40  
      3.2.2. Simulation Design Examples ................................................................................. 41  
   3.3. 25G Ethernet to CPRI Dynamic Reconfiguration Design Example .................................... 50  
      3.3.1. Functional Description .......................................................................................... 50  
      3.3.2. Simulation Design Examples ................................................................................. 50  
   3.4. Document Revision History for the E-tile Dynamic Reconfiguration Design Example ...... 57
1. About E-tile Hard IP Intel® Agilex™ Design Example
User Guide

This user guide consists of design examples for the following IP cores:

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

2.1. Quick Start Guide

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

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

Note: Intel Quartus® Prime Pro Edition software version 19.3 does not support hardware design examples for Intel Agilex devices.

Table 1. List of Supported Design Example Variants

<table>
<thead>
<tr>
<th>Data Rate</th>
<th>Variant Description</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>X</td>
</tr>
<tr>
<td></td>
<td>Single channel PCS</td>
<td>✓</td>
<td>✓</td>
<td>X</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 with optional Reed-Solomon forward error correction (RS-FEC)</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>25GE</td>
<td>Single or multi channels MAC + PCS with optional RS-FEC and optional PTP</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Single channel PCS with optional RS-FEC</td>
<td>✓</td>
<td>✓</td>
<td>X</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>

continued...
<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>X</td>
<td></td>
</tr>
</tbody>
</table>
| 100GE | MAC+ PCS with optional:  
• (528,514) RS-FEC  
• PTP | ✓ | ✓ | X |
| | MAC+PCS with (544, 514) RS-FEC | ✓ | ✓ | X |
| | PCS with optional (528,514) or (544, 514) RS-FEC | ✓ | ✓ | X |
| | OTN with optional (528,514) or (544, 514) RS-FEC | ✓ | ✓ | X |
| | FlexE with optional (528,514) or (544, 514) RS-FEC | ✓ | ✓ | X |

Figure 1. Development Steps for the Design Example

2.1.1. Directory Structure

The E-tile Ethernet IP for Intel Agilex FPGA design example file directories contain the following generated files for the design examples.
Figure 2. **E-tile Ethernet IP for Intel Agilex FPGA 10GE/25GE with Optional RS-FEC and Optional PTP Design Example Directory Structure**

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

```
<alt_ehipc3_fm_0_example_design>
  ex_<datarate>G
    sim
      ex_<datarate>G.smp
    synth
      ex_<datarate>G.syn
    alt_ehipc3_fm_gen
      ex_<datarate>G_gen
    alt_ehipc3_fm
      ex_<datarate>G.sdc

<example_testbench>
  basic_avl_tb_top.sv

<compilation_test_design>
  alt_ehipc3_fm_hw.qpf
  ex_<datarate>G.ip

<hardware_test_design (1)>
  ex_<datarate>G_qip
  ex_<datarate>G_sopcinfo

Note:
1. Intel Quartus® Prime Pro Edition software version 19.1 does not support hardware design examples for the E-Tile Hard IP for Ethernet Intel FPGA IP core for Intel Agilex devices.
Figure 3. E-tile Ethernet IP for Intel Agilex FPGA 100GE with Optional RS-FEC Design Example Directory Structure

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

Table 2. E-tile Ethernet IP for Intel Agilex FPGA Core Testbench File Descriptions

<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>
<tr>
<td><code>&lt;design_example_dir&gt;/example_testbench/run_vsims.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 Agilex IP Core Hardware Design Example File Descriptions**

<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_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 Agilex 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>
<tr>
<td>hwtest_sl/main_script.tcl (10GE/25GE) hwtest/main.tcl (100GE)</td>
<td>Main file for accessing System Console.</td>
</tr>
</tbody>
</table>

### 2.1.2. Generating the Design

**Figure 4. Procedure**

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

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

1. In the IP Catalog, locate and select **E-tile Ethernet IP for Intel Agilex FPGA**. The **New IP Variation** window appears.
2. Specify a top-level name `<your_ip>` for your custom IP variation. The parameter editor saves the IP variation settings in a file named `<your_ip>.ip`.
3. Click **OK**. The parameter editor appears.
4. On the **IP**, 100GE, or 10GE/25GE tabs, specify the parameters for your IP core variation.
5. On the Example Design tab, under Example Design Files, select the Simulation option to generate the testbench and the compilation-only project. You must select at least one of the Simulation and Synthesis options to generate the design example.

6. On the Example Design tab, under Generated HDL Format, select Verilog HDL or VHDL. If you select VHDL, you must simulate the testbench with a mixed-language simulator. The device under test in the ex_<datarate> directory is a VHDL model, but the main testbench file is a System Verilog file.

7. Under Target Development Kit, select None. The compilation-only design examples target your project device.

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

9. If you want to modify the design example directory path or name from the defaults displayed (alt_ehipc3_fm_0_example_design), browse to the new path and type the new design example directory name (<design_example_dir>).

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

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

Figure 6. Procedure

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>
<tbody>
<tr>
<td>Mentor Graphics</td>
<td>In the command line, type vsim -do run_vsim.do</td>
</tr>
<tr>
<td>ModelSim*</td>
<td>If you prefer to simulate without bringing up the ModelSim GUI, type vsim -c -do run_vsim.do</td>
</tr>
</tbody>
</table>
2. 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

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

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

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

<table>
<thead>
<tr>
<th>Variant</th>
<th>Intel Agilex Design Example Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAC+PCS with Optional RS-FEC(^{(1)})</td>
<td>Simulation and compilation-only project</td>
</tr>
<tr>
<td>MAC+PCS with Optional RS-FEC and PTP(^{(1)})</td>
<td>Simulation and compilation-only project</td>
</tr>
<tr>
<td>PCS Only with Optional RS-FEC(^{(1)})</td>
<td>Simulation and compilation-only project</td>
</tr>
<tr>
<td>OTN with Optional RS-FEC(^{(1)})</td>
<td>Simulation and compilation-only project</td>
</tr>
<tr>
<td>FlexE with Optional RS-FEC(^{(1)})</td>
<td>Simulation and compilation-only project</td>
</tr>
<tr>
<td>Custom PCS with Optional RS-FEC(^{(1)})</td>
<td>Simulation and compilation-only project</td>
</tr>
</tbody>
</table>

\(^{(1)}\) 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.

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

Figure 7. Simulation Block Diagram for Non-PTP E-tile Ethernet IP for Intel Agilex FPGA 10GE/25GE MAC+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.

To speed up simulation, the IP core simulation model sends alignment marker tags at shorter intervals than required by the IEEE Ethernet standard. The standard specifies an alignment marker interval of 16,384 words in each virtual lane. The simulation model with the testbench implements an alignment marker interval of 512 words.

The successful test run displays output confirming the following behavior:

1. Waiting for PLL to lock.
2. Waiting for RX transceiver reset to complete.
3. Waiting for RX alignment.
4. Sending 10 packets.
5. Receiving those packets.
6. Displaying Testbench complete.
The following sample output illustrates a successful simulation test run for a 25GE, MAC+PCS with RS-FEC, non-PTP IP core variation.

```
# Ref clock is 156.25 MHz
# Channel 0 - waiting for EHIP Ready....
# Channel 0 - EHIP READY is 1 at time 2472365000
# Channel 0 - Waiting for RX Block Lock
# Channel 0 - RX block lock is high at time 2507639043
# Channel 0 - RX alignment
# 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 = 00000000
** Address offset = 00300, ReadData = 11112015
** Address offset = 00400, ReadData = 11112015
** Address offset = 00836, ReadData = 0000000a
** Address offset = 00804, ReadData = 00000000
** Address offset = 00904, ReadData = 00000000
** Address offset = 00322, ReadData = fffffff
** Address offset = 00084, WriteData = ffffffff
** Address offset = 00084, ReadData = 000000ff
** Address offset = 00084, WriteData = 00000000
** Address offset = 00230, WriteData = ffffffff
** Address offset = 00230, WriteData = 0000007b
** AVMM access CSR registers read/write check for ETH RSFEC
** Address offset = 10000, ReadData = 00000001
** Address offset = 10000, WriteData = ffffffff
** Address offset = 10000, ReadData = 00000000
** 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.
**
```
2.2.1.2. 10GE/25GE MAC+PCS with Optional RS-FEC and PTP 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 8.** Simulation Block Diagram for E-tile Ethernet IP for Intel Agilex FPGA 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)

.  

### Channel 3 - Configure TX extra latency
### writedata = 0004267a

### Channel 3 - Configure RX extra latency
### writedata = 800369d0

### Channel 3 - TX enabled
### Channel 3 - Sending Packet  1
### Channel 3 - Sending Packet  2
### Channel 3 - Sending Packet  3
### Channel 3 - Sending Packet  4
### Channel 3 - Sending Packet  5
### Channel 3 - Sending Packet  6
### Channel 3 - Sending Packet  7
### Channel 3 - Sending Packet  8
### Channel 3 - Sending Packet  9
### Channel 3 - Sending Packet 10
### Channel 3 - Received Packet  1
### Channel 3 - Received Packet  2
### Channel 3 - Received Packet  3
### Channel 3 - Received Packet  4
### Channel 3 - Received Packet  5
### Channel 3 - Received Packet  6
### Channel 3 - Received Packet  7
### Channel 3 - Received Packet  8
### Channel 3 - Received Packet  9
### Channel 3 - Received Packet 10

** Testbench complete.

Related Information

Simulating the E-tile Ethernet IP for Intel Agilex FPGA Design Example Testbench on page 9

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.

```
# 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.

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.

```plaintext
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
```
<table>
<thead>
<tr>
<th>Channel</th>
<th>Status</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>RX lane alignment locked</td>
<td>389768227</td>
</tr>
</tbody>
</table>
| 0       | TX enabled | |*
| 0       | Sending packets *** | |*
| 0       | Start frame detected, byteslip 0 | 395241712 |
| 0       | RX checker has received packets correctly! | |*
| 0       | RX checker is reset. | |*
| 0       | Second attempt of sending packets *** | |*
| 0       | Start frame detected, byteslip 0 | 400721512 |
| 0       | RX checker has received packets correctly! | |*
| 0       | RX checker is reset. | |*
| 0       | Third attempt of sending packets *** | |*
| 0       | Start frame detected, byteslip 0 | 406113519 |
| 0       | RX checker has received packets correctly! | |*
| 0       | RX checker is reset. | |*
| 1       | RX lane alignment locked | |*
| 1       | TX enabled | |*
| 1       | Sending packets *** | |*
| 1       | Start frame detected, byteslip 0 | 406605943 |
| 1       | RX checker has received packets correctly! | |*
| 1       | RX checker is reset. | |*
| 1       | Second attempt of sending packets *** | |*
| 1       | Start frame detected, byteslip 0 | 411907712 |
| 1       | RX checker has received packets correctly! | |*
| 1       | RX checker is reset. | |*
| 1       | Third attempt of sending packets *** | |*
| 1       | Start frame detected, byteslip 0 | 417092055 |
| 1       | RX checker has received packets correctly! | |*
| 2       | RX lane alignment locked | |*
| 2       | TX enabled | |*
| 2       | Sending packets *** | |*
| 2       | Start frame detected, byteslip 0 | 422502903 |
| 2       | RX checker has received packets correctly! | |*
| 2       | RX checker is reset. | |*
| 2       | Second attempt of sending packets *** | |*
| 2       | Start frame detected, byteslip 0 | 428007954 |
| 2       | RX checker has received packets correctly! | |*
| 2       | RX checker is reset. | |*
| 2       | Third attempt of sending packets *** | |*
| 2       | Start frame detected, byteslip 0 | 433494066 |
| 2       | RX checker has received packets correctly! | |*
| 3       | RX lane alignment locked | |*
| 3       | TX enabled | |*
| 3       | Sending packets *** | |*
| 3       | Start frame detected, byteslip 0 | 438905013 |
| 3       | RX checker has received packets correctly! | |*
| 3       | RX checker is reset. | |*
| 3       | Second attempt of sending packets *** | |*
| 3       | Start frame detected, byteslip 0 | 444384812 |
| 3       | RX checker has received packets correctly! | |*
| 3       | RX checker is reset. | |*
| 3       | Third attempt of sending packets *** | |*
2.3. 100GE with Optional RS-FEC Design Example

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

Table 6. Supported Design Example Variants for 100GE

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

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

2.3.1. Simulation Design Examples

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

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

1. Under the **IP** tab:
a. **Single 100GE with optional RSFEC** or **100GE or 1 to 4 channel 10GE/25GE with optional RSFEC and PTP** as the core variant.

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

c. **Enable RSFEC** to use the RS-FEC feature.

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

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

---

**Figure 11.** Simulation Block Diagram for E-tile Ethernet IP for Intel Agilex FPGA 100GE MAC+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.

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.

```text
# o_tx_lanes_stable is 1 at time             345651500
# waiting for tx_dll_lock....
# TX DLL LOCK is 1 at time             398849563
# waiting for tx_transfer_ready....
# TX transfer ready is 1 at time             399169435
# waiting for rx_transfer_ready....
# RX transfer ready is 1 at time             410719813
# EHIP PLD Ready out is 1 at time             410776000
# EHIP reset out is 0 at time             411040000
# EHIP reset ack is 0 at time             412282101
# EHIP TX reset out is 0 at time             413160000
# EHIP TX reset ack is 0 at time             462643731
# waiting for EHIP Ready....
# EHIP READY is 1 at time             462750387
# EHIP RX reset out is 0 at time             463088000
# waiting for rx reset ack....
# EHIP RX reset ack is 0 at time             463283667
# Waiting for RX Block Lock
# EHIP RX Block Lock  is high at time             467376591
# Waiting for AM lock
# EHIP RX AM Lock  is high at time             468643131
# Waiting for RX alignment
# RX deskew locked
# RX lane alignment locked
# ** Sending Packet     1...
# ** Sending Packet     2...
# ** Sending Packet     3...
# ** Sending Packet     4...
# ** Sending Packet     5...
# ** Sending Packet     6...
# ** Sending Packet     7...
# ** Received Packet    1...
# ** Sending Packet     8...
# ** Received Packet    2...
# ** Sending Packet     9...
# ** Received Packet    3...
# ** Received Packet    4...
# ** Sending Packet    10...
# ** Received Packet    5...
# ** Received Packet    6...
# ** Received Packet    7...
# ** Received Packet    8...
# ** Received Packet    9...
# ** Received Packet   10...
# =====>MATCH!    ReaddataValid = 1 Readdata = 11112015 Expected_Readdata = 11112015
# =====> writedata = ffff0000
# =====>MATCH!    ReaddataValid = 1 Readdata = 11112015 Expected_Readdata = 11112015
# =====> writedata = 4321abcd
# =====>MATCH!    ReaddataValid = 1 Readdata = 4321abcd Expected_Readdata = 4321abcd
# =====> writedata = a5a51234
# =====>MATCH!    ReaddataValid = 1 Readdata = a5a51234 Expected_Readdata = a5a51234
```
2. E-tile Ethernet IP for Intel Agilex FPGA 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.
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.

```plaintext
# o_tx_lanes_stable is 1 at time 346295000
# waiting for tx_dll_lock....
# TX DLL LOCK is 1 at time 405180363
# waiting for tx_transfer_ready....
# TX transfer ready is 1 at time 405500235
# waiting for rx_transfer_ready....
# RX transfer ready is 1 at time 416575803
# EHIP PLD Ready out is 1 at time 416632000
# EHIP reset out is 0 at time 416768000
# EHIP reset ack is 0 at time 416844540
# EHIP TX reset out is 0 at time 417184000
# EHIP TX reset ack is 0 at time 468265476
# waiting for EHIP Ready....
# EHIP READY is 1 at time 468389597
# EHIP RX reset out is 0 at time 470288000
# waiting for rx reset ack....
# EHIP RX reset ack is 0 at time 470301064
# Waiting for RX Block Lock
# EHIP RX Block Lock is high at time 511027716
# Waiting for AM lock
```
# EHIP RX AM Lock is high at time 511027716
# Waiting for RX alignment
# RX deskew locked
# RX lane 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_cw_pos_rx_3
# rsfec_cw_pos_rx_3 = 32'h7dd
# min skew value = 32'h1
# lane_skew_adjust = 32'h1
# Tlat_final = 32'h4
# Generate VL offset data
# before-rotation: VL[PL] 0[0], deskew_delay = 4 UI, vl_offset_bits = 4
# After rotation: VL_OFFSET for RVL[PL] 4[0] = 0 ns 27b8 Fns, Sign bit = 0
# 
# before-rotation: VL[PL] 19[0], deskew_delay = 4 UI, vl_offset_bits = 8
# after-rotation: VL_OFFSET for RVL[PL] 3[0] = c ns 7d5d Fns, Sign bit = 1
# Writing VL offset data for VL 0
# -> writedata = 00000004
# -> writedata = 000027b8
# Writing VL offset data for VL 19
# -> writedata = 00000003
# -> writedata = 800c7d5d
# Waiting for RX PTP Ready
# RX PTP ready
** Sending Packet 1...
** Sending Packet 2...
** Sending Packet 3...
** Sending Packet 4...
** Sending Packet 5...
** Sending Packet 6...
** Sending Packet 7...
** Sending Packet 8...
** Sending Packet 9...
** Received Packet 1...
** Received Packet 2...
** Received Packet 3...
** Received Packet 4...
** Received Packet 5...
** Received Packet 6...
** Received Packet 7...
** Received Packet 8...
** Received Packet 9...
** Received Packet 10...
**
** Testbench complete.
**
Related Information
Simulating the E-tile Ethernet IP for Intel Agilex FPGA Design Example Testbench on page 9

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

The simulation block diagram below is generated using the following settings in the IP parameter editor:
1. Under the IP tab:
   a. **Single 100GE with optional RSFEC** or **100GE or 1 to 4 channel 10GE/25GE with optional RSFEC and PTP** as the core variant.
   b. **100GE Channel** as **Active channel(s) at startup** if you choose **100GE or 1 to 4 channel 10GE/25GE with optional RSFEC and PTP** as the core variant.
2. Under the **100GE** tab:
   a. **100G** as the Ethernet rate.
   b. **PCS_Only**, **PCS+(528,514)RSFEC**, or **PCS+(544,514)RSFEC** as the Ethernet IP layer.

**Figure 13.** Simulation Block Diagram for E-tile Ethernet IP for Intel Agilex FPGA 100GE PCS Only Design Example

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

To speed up simulation, the IP core simulation model sends alignment marker tags at shorter intervals than required by the IEEE Ethernet standard. The standard specifies an alignment marker interval of 16,384 words in each virtual lane. The simulation model with the testbench implements an alignment marker interval of 512 words.
The successful test run displays output confirming the following behavior:

1. The client logic resets the IP core.
2. Waits for RX datapath to align.
3. Once alignment is complete, client logic transmits a series of packets to the IP core through TX MII interface.
4. A counter drives \texttt{i\_tx\_mii\_am} port with alignment marker insertion requests at the correct intervals.
5. The client logic receives the same series of packets through RX MII interface.
6. The client logic then checks the number of packets received.
7. Displaying Testbench complete.

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

```
o_tx_lanes_stable is 1 at time 354775000
waiting for tx_dll_lock....
TX DLL LOCK is 1 at time 413726943
waiting for tx_transfer_ready....
TX transfer ready is 1 at time 414046815
waiting for rx_transfer_ready....
RX transfer ready is 1 at time 425122383
EHIP PLD Ready out is 1 at time 425184000
EHIP reset out is 0 at time 425320000
EHIP reset ack is 0 at time 426016853
EHIP TX reset out is 0 at time 426232000
EHIP TX reset ack is 0 at time 476830347
waiting for EHIP Ready....
EHIP READY is 1 at time 476910363
EHIP RX reset out is 0 at time 478680000
waiting for rx reset ack....
EHIP RX reset ack is 0 at time 478777403
Waiting for RX Block Lock
EHIP Rx Block Lock is high at time 481444603
Waiting for AM lock
Waiting for RX alignment
RX deskew locked
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 Ethernet IP for Intel Agilex FPGA Design Example Testbench on page 9

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

The simulation block diagram below is generated using the following settings in the IP parameter editor:
1. Under the **IP** tab:
   a. **Single 100GE with optional RSFEC** or **100GE or 1 to 4 channel 10GE/25GE with optional RSFEC and PTP** as the core variant.
   b. **100GE Channel** as **Active channel(s) at startup** if you choose **100GE or 1 to 4 channel 10GE/25GE with optional RSFEC and PTP** as the core variant.

2. Under the **100GE** tab:
   a. **100G** as the Ethernet rate.
   b. **OTN, OTN+(528,514)RSFEC**, or **OTN+(544,514)RSFEC** as the Ethernet IP layer.

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

**Figure 14. Simulation Block Diagram for E-tile Ethernet IP for Intel Agilex FPGA 100GE OTN Design Example**

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

The successful test run displays output confirming the following behavior:
1. The client logic resets both the IP cores.
2. The stimulus client logic waits for the stimulus RX datapath and OTN RX datapath to align.
3. Once alignment is complete, the stimulus client logic transmits a series of packets to the OTN IP core.
4. The OTN IP core receives the series of packets and transmits back to the stimulus MAC IP core.
5. The stimulus client logic then checks the number of packets received and verify that the packets have no errors.
6. Displaying **Testbench complete**.
The following sample output illustrates a successful simulation test run for a 100GE OTN IP core variation.

```
# test_dut: def_100G_o_tx_lanes_stable is 1 at time 345685000
# test_dut: waiting for tx_dll_lock....
# dut: o_tx_lanes_stable is 1 at time 345685000
# dut: waiting for tx_dll_lock....
# dut: TX DLL LOCK is 1 at time 398849563
# dut: waiting for tx_transfer_ready....
# dut: TX transfer ready is 1 at time 399169435
# dut: waiting for rx_transfer_ready....
# dut: RX transfer ready is 1 at time 410719813
# dut: waiting for TX DLL LOCK is 1 at time 411040000
# dut: waiting for EHIP PLD Ready out is 1 at time 410776000
# dut: EHIP reset out is 0 at time 411040000
# dut: EHIP reset ack is 0 at time 412282101
# dut: EHIP TX reset out is 0 at time 413160000
# dut: EHIP TX reset ack is 0 at time 462643731
# dut: waiting for EHIP Ready....
# dut: EHIP READY is 1 at time 462750387
# test_dut: RX transfer ready is 1 at time 463088000
# dut: waiting for rx reset ack....
# dut: EHIP RX reset out is 0 at time 463283667
# dut: Waiting for RX Block Lock
# test_dut: TX DLL LOCK is 1 at time 475452243
# test_dut: waiting for tx_transfer_ready....
# test_dut: TX transfer ready is 1 at time 475772115
# test_dut: waiting for rx_transfer_ready....
# test_dut: RX transfer ready is 1 at time 487164223
# test_dut: EHIP PLD Ready out is 1 at time 487224000
# test_dut: EHIP reset out is 0 at time 487488000
# test_dut: EHIP reset ack is 0 at time 488907771
# test_dut: EHIP TX reset out is 0 at time 489784000
# test_dut: EHIP TX reset ack is 0 at time 539116083
# test_dut: waiting for EHIP Ready....
# test_dut: EHIP READY is 1 at time 539169411
# test_dut: RX transfer ready is 1 at time 539512000
# test_dut: waiting for rx reset ack....
# test_dut: EHIP RX reset out is 0 at time 539512000
# test_dut: waiting for rx reset ack....
# test_dut: EHIP RX reset ack is 0 at time 539702691
# test_dut: Waiting for RX Block Lock
# dut: RX Block Lock is high at time 542102451
# dut: Waiting for AM lock
# test_dut: RX Block Lock is high at time 542735721
# test_dut: Waiting for AM lock
# dut: EHIP RX AM Lock is high at time 543368991
# dut: Waiting for RX alignment
# dut: RX deskew locked
# dut: RX lane alignment locked
# dut: ******************************************************
# test_dut: EHIP RX AM Lock is high at time 549068421
# test_dut: waiting for RX alignment
# test_dut: RX deskew locked
# test_dut: RX lane alignment locked
# test_dut: ** Sending Packet 1...
# test_dut: ** Sending Packet 2...
# test_dut: ** Sending Packet 3...
# test_dut: ** Sending Packet 4...
# test_dut: ** Sending Packet 5...
# test_dut: ** Sending Packet 6...
# test_dut: ** Sending Packet 7...
# test_dut: ** Sending Packet 8...
# test_dut: ** Sending Packet 9...
# test_dut: ** Sending Packet 10...
# test_dut: ** Testbench complete.
# test_dut: ******************************************************
```
2.3.1.5. E-tile Ethernet IP for Intel Agilex FPGA 100GE FlexE with Optional RS-FEC Simulation Design Example

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

1. Under the **IP** tab:
   a. **Single 100GE with optional RSFEC** or **100GE or 1 to 4 channel 10GE/25GE with optional RSFEC and PTP** as the core variant.
   b. **100GE Channel** as **Active channel(s) at startup** if you choose **100GE or 1 to 4 channel 10GE/25GE with optional RSFEC and PTP** as the core variant.

2. Under the **100GE** tab:
   a. **100G** as the Ethernet rate.
   b. **FlexE**, **FlexE+(528,514)RSFEC**, or **FlexE+(544,514)RSFEC** as the Ethernet IP layer.

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.

```plaintext
# test_dut: def_100G_o_tx_lanes_stable is 1 at time 34568500
# test_dut: waiting for tx_dll_lock....
# dut: o_tx_lanes_stable is 1 at time 34568500
# 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 410719813
# dut: waiting for rx_transfer_ready....
# dut: RX transfer ready is 1 at time 413160000
# dut: waiting for EHIP Ready....
# dut: EHIP READY is 1 at time 462750387
# dut: EHIP RX reset out is 0 at time 463088000
# dut: waiting for rx reset ack....
# dut: EHIP RX reset ack is 0 at time 463283667
# dut: Waiting for RX Block Lock
# test_dut: TX DLL LOCK is 1 at time 475452243
# test_dut: waiting for tx_transfer_ready....
# test_dut: TX transfer ready is 1 at time 475772115
# test_dut: waiting for rx_transfer_ready....
# test_dut: RX transfer ready is 1 at time 487164223
# test_dut: EHIP PLD Ready out is 1 at time 487224000
# test_dut: EHIP reset out is 0 at time 487488000
# test_dut: EHIP reset ack is 0 at time 488907771
# test_dut: EHIP TX reset out is 0 at time 489784000
# test_dut: EHIP TX reset ack is 0 at time 539116083
# test_dut: waiting for EHIP Ready....
# test_dut: EHIP READY is 1 at time 539169411
# test_dut: EHIP RX reset out is 0 at time 539512000
# test_dut: waiting for rx reset ack....
# test_dut: EHIP RX reset ack is 0 at time 539702691
# test_dut: Waiting for RX Block Lock
# dut: EHIP RX Block Lock is high at time 542102451
# dut: Waiting for AM lock
# dut: EHIP RX AM Lock is high at time 543368991
# dut: Waiting for RX alignment
# dut: RX deskew locked
# dut: RX lane alignment locked
dut: *****************************************
# test_dut: EHIP RX Block Lock is high at time 546535341
# test_dut: Waiting for AM lock
# test_dut: EHIP RX AM Lock is high at time 547801881
# test_dut: Waiting for RX alignment
# dut: RX deskew locked
test_dut: RX lane alignment locked
test_dut: ** Sending Packet 1...
# test_dut: ** Sending Packet 9...
# test_dut: ** Sending Packet 10...
# test_dut: ** Received Packet 1...
# test_dut: ** Received Packet 9...
# test_dut: ** Received Packet 10...
# test_dut: **
# test_dut: ** Testbench complete.
test_dut: **
test_dut: *****************************************
```
2. E-tile Ethernet IP for Intel Agilex FPGA Design Example

UG-20210 | 2019.10.18

Related Information
Simulating the E-tile Ethernet IP for Intel Agilex FPGA Design Example Testbench on page 9


<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>2019.10.18</td>
<td>19.3</td>
<td>19.3.0</td>
<td>Initial Release.</td>
</tr>
</tbody>
</table>
3. E-Tile Dynamic Reconfiguration Design Example

3.1. Quick Start Guide

Starting with Intel Quartus Prime software version 19.3, the E-tile Dynamic Reconfiguration design example provides simulation testbench that supports compilation and timing-only testing for Intel Agilex devices. Hardware design examples are not available.

Table 7. List of Supported Dynamic Reconfiguration Design Example Variants

<table>
<thead>
<tr>
<th>Dynamic Reconfiguration Protocol</th>
<th>Variant</th>
<th>Simulation</th>
<th>Compilation-Only Project</th>
<th>Hardware Design Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>25G Ethernet to CPRI Protocol</td>
<td>25G with PTP and optional RS-FEC</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
</tbody>
</table>

Figure 16. Development Steps for the Design Example

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

3.1.1. Directory Structure

The E-tile dynamic reconfiguration design example file directories contain the following generated files for the design examples.
Figure 17.  E-tile Dynamic Reconfiguration 10G/25G Ethernet and 25G Ethernet to CPRI Design Example Directory Structure

```
<design_example>
  software
    dynamic_reconfiguration_hardware
      c3_reconfig.c
      c3_reconfig.h
      c3_function.c
      flow.c
      main.c
      packet_gen.c
      packet_gen.h
    dynamic_reconfiguration_sim
      c3_reconfig.c
      c3_reconfig.h
      c3_function.c
      main.c
      nios_system_onchip_memory2_0_onchip_memory2_0.hex
      packet_gen.c
      packet_gen.h
  example_testbench
    common
    mentor
    synopsys
    cadence
    xcelium
    setup_scripts
      basic_avl_tb_top.sv
  hardware_test_design
    common
    ex_25G
    eth_25g_channel_pll(1)
    ip
    nios_system
    reset_release
      alt_ehpic3.qpf
      alt_ehpic3.qsf
      alt_ehpic3_fm.sdc
      alt_ehpic3.sv
      eth_25g_channel_pll.ip(1)
      ex_25G.ip
      nios_system.qsys
      reset_release.ip
```

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

Table 8.  E-tile Dynamic Reconfiguration Design Example Testbench File Descriptions

<table>
<thead>
<tr>
<th>File Names</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key Testbench and Simulation Files</td>
<td></td>
</tr>
<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>
<tr>
<td>Testbench Scripts</td>
<td></td>
</tr>
<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>
</tbody>
</table>

*continued...*
### 3.1.2. Generating the Design

#### Figure 18. 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 19. Example Design Tab in the E-tile Dynamic Reconfiguration Design Example Parameter Editor

If you do not already have an Intel Quartus Prime Pro Edition project in which to integrate your IP core, you must create one.

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

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

3. Click **Finish**.

Follow these steps to generate the E-tile Dynamic Reconfiguration design example hardware design example and testbench:

<table>
<thead>
<tr>
<th>File Names</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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 Xcelium script to run the testbench.</td>
</tr>
</tbody>
</table>
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 **Ethernet Protocol** or **Ethernet to CPRI Protocol**.
   - If you select **Ethernet Protocol**, click the **10G/25G Ethernet Protocol** tab.
   - If you select **Ethernet to CPRI Protocol**, click the **25G Ethernet to CPRI Protocol** tab.

5. Under **Select DR Design**, select a starting base variant IP for the selected DR Protocol design.

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

7. 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>`).

8. Click **OK**.

### 3.1.2.1. Design Example Parameters

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Options</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Select DR Protocol** | • 10G/25G Ethernet Protocol  
• 25G Ethernet to CPRI Protocol | Available protocols for dynamic reconfiguration design example generation. |

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

| Select DR Design | 25G 1588 PTP RS-FEC  
25G RS-FEC | Available base variants for Ethernet Dynamic Reconfiguration design example generation. |

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

| Select DR Design | 25GE PTP RS-FEC | Available base variant for Ethernet to CPRI Dynamic Reconfiguration design example generation. |

*continued...*
### Parameter Settings: 10G/25G Ethernet Protocol and 25G Ethernet to CPRI Protocol (The parameters below are available in both tabs)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify Number of Channels</td>
<td>1</td>
<td>Specify the number of channels. The valid number of channels is 1 and this parameter is not selectable.</td>
</tr>
<tr>
<td>Select Board</td>
<td>• Other Development Kits</td>
<td>Supported hardware for design implementation. When you select an Intel FPGA development board, the Target Device is the one that matches the device on the Development Kit. If this menu is not available, there is no supported board for the options that you select. Other Development Kits: This option allows the design example to be tested on development kits other than 1ST280EY2F55E2VG. You need to set the pin assignments based on the board used to run this design example.</td>
</tr>
</tbody>
</table>

#### 3.1.3. 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 20. Procedure**

1. Change to Testbench Directory
2. Run `<Simulation Script>`
3. Analyze Results

#### 3.1.3.1. 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 10. Steps to Simulate the Testbench

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

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

### 3.1.3.2. Running the Simulation with New HEX File

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

1. In the Intel Quartus Prime Pro Edition software, select **Tools ➤ Nios II Software Build Tools for Eclipse**.
2. Create a new workspace when the **Workspace Launcher** window prompt appears. Click **OK** to open the workspace.
3. In the **Nios II - Eclipse** window, select **File ➤ New ➤ Nios II Application and BSP from Template**. A **Nios II Application and BSP from Template** appears.
4. In the **Nios II Application and BSP from Template** window, fill in the following information:
• For SOPC Information File name, browse to `<design_example_dir>/hardware_test_design/nios_system` and open the SOPC Information File (nios_system.sopcinfo) for your design. Click OK to select the file and Eclipse automatically loads all CPU settings.

• For Project name, specify your desired project name. This example uses dynamic_reconfiguration_simulation.

5. Click Finish to generate the project. The Intel Quartus Prime Pro Edition software creates a new directory named software in the specified project location.

6. Replace the C-code source files located in your new software directory (`<design_example_dir>/hardware_test_design/software/dynamic_reconfiguration_simulation`) with the following C-code source files from the `<design_example_dir>/software/dynamic_reconfiguration_sim` design:
   - c3_reconfig.c
   - c3_reconfig.h
   - c3_function.c
   - flow.c
   - main.c
   - packet_gen.c
   - packet_gen.h

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

7. In the Nios II - Eclipse window, press F5 or right-click your project and select Refresh to refresh the window and reload the new files into the project.

8. On the Project Explorer view, right-click the dynamic_reconfiguration_simulation and select Build Project. Ensure the dynamic_reconfiguration_simulation.elf file is generated in the new `<design_example_dir>/hardware_test_design/software/dynamic_reconfiguration_simulation` directory.

9. To generate a new HEX file, right-click the dynamic_reconfiguration_simulation in the Project Explorer view, point to Make Targets and select Build. A Make Targets dialog box appears.

10. In the Make Targets dialog box, select mem_init_generate.

11. Click Build. The mem_init_generate creates the new HEX (nios_system_onchip_memory2_0_onchip_memory2_0.hex) file. The new HEX file resides in the `<design_example_dir>/hardware_test_design/software/dynamic_reconfiguration_simulation/mem_init` directory.
Follow these steps to simulate the testbench:

1. Open the `<simulator_name>_files.tcl` script in the `<design_example_dir>/example_testbench/setup_scripts/common` directory.

2. Edit the TCL script to change the existing `nios_system_onchip_memory2_0_onchip_memory2_0.hex` file directory to the new HEX file generated from the Nios II SBT for Eclipse:

   For example, change the following line in the TCL script from:
   ```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/software/dynamic_reconfiguration_simulation/mem_init/nios_system_onchip_memory2_0_onchip_memory2_0.hex"]"
   ```

3. Using the supported simulator of your choice, change to the testbench simulation directory to `<design_example_dir>/example_testbench/`.<simulator_name>.

4. Run the simulation script for the simulator. The script compiles and runs the testbench in the simulator. Refer to the table Table 10 on page 37.

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.

### 3.2. 10G/25G Ethernet Dynamic Reconfiguration Design Examples

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

<table>
<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>
3.2.1. Functional Description

3.2.1.1. Clocking Scheme

Figure 21. Clocking Scheme for 10G/25GE MAC+PCS with RS-FEC and PTP Dynamic Reconfiguration Design Example

Figure 22. Clocking Scheme for 10G/25GE MAC+PCS with RS-FEC Dynamic Reconfiguration Design Example

Note: i_channel_PLL signal is E-tile Transceiver PHY specific signal that utilizes additional transceiver E-tile channel.
3.2.2. Simulation Design Examples

3.2.2.1. 10GE/25GE MAC+PCS with RS-FEC and PTP Simulation Dynamic Reconfiguration Design Example Components

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

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

**Figure 23.** Simulation Block Diagram for E-Tile Ethernet IP for Intel Agilex FPGA 10GE/25GE with RS-FEC and PTP Dynamic Reconfiguration Design Example

The successful test run displays output confirming the following behavior:

1. Asserting all reset signals and deasserting `sl_csr_rst_n`, `sl_tx_rst_n`, `sl_rx_rst_n`, and `i_reconfig_reset` signals.

2. Performing internal loopback test:
   a. Waiting for `PIO_OUT[0] = 0x1` (o_ehip_ready asserted).
   b. Enabling SERDES loopback.
   c. Waiting 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).
   d. Continuously sending packets for the clock data recover (CDR) receiver (RX) deskew training and waiting until `PIO_OUT[4] = 0x1` (o_rx_ptp_ready asserted).
e. Clearing Ethernet statistic counters.

f. Enabling the packet generator to send packets of data, checking the transmitter (TX) packet count statistic counter to ensure all the packets are sent, and stopping the packet generator.

g. Checking for expected packets to be received by the packet checker.

3. Performing dynamic reconfiguration (DR) test from 25G PTP with RS-FEC to 25G PTP without RS-FEC:

a. Asserting reset signals (sl_tx_rst_n and sl_rx_rst_n).

b. Disabling SERDES. Use PMA attribute code 0x0001 in the E-tile Transceiver PHY User Guide: PMA Attribute Codes section.

c. Triggering PMA analog reset. For more information about register descriptions, refer to the E-tile Transceiver PHY User Guide.

d. Reconfiguring 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.

e. Adjusting the phase offset of a recovered clock. Use PMA attribute code 0x000E in the E-tile Transceiver PHY User Guide: PMA Attribute Codes section.


g. Enabling SERDES loopback.

h. Deasserting the reset signals (sl_tx_rst_n and sl_rx_rst_n).

i. Waiting 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).

j. Continuously sending packets for the CDR RX deskew training and waiting until PIO_OUT[4] = 0x1 (o_rx_ptp_ready asserted).

k. Clearing Ethernet statistic counters.

l. Enabling the packet generator to send packets of data, checking the TX packet count statistic counter to ensure all the packets are sent, and stopping the packet generator.

m. Checking for expected packets to be received by the packet checker.

4. Performing DR test from 25G PTP without RS-FEC to 10G PTP:

a. Asserting reset signals (sl_tx_rst_n and sl_rx_rst_n).

b. Disabling SERDES.

c. Triggering PMA analog reset.

d. Changing the transceiver TX bit/refclk ratio to 10G (based on 156.25 MHz refclk).

e. Changing the transceiver RX bit/refclk ratio to 10G (based on 156.25 MHz refclk).

f. Reconfiguring registers for the Ethernet, RS-FEC, and transceiver blocks.

g. Adjusting the phase offset of a recovered clock.

h. Enabling SERDES.
i. Enabling SERDES loopback.

j. Deasserting the reset signals (sl_tx_rst_n and sl_rx_rst_n).

k. Waiting 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).

l. Continuously sending packets for the CDR RX deskew training and waiting until PIO_OUT[4] = 0x1 (o_rx_ptp_ready asserted).

m. Clearing Ethernet statistic counters.

n. Enabling the packet generator to send packets of data, checking the TX packet count statistic counter to ensure all the packets are sent, and stopping the packet generator.

o. Checking for expected packets to be received by the packet checker.

5. Performing DR test from 10G PTP to 25G PTP without RS-FEC:

a. Asserting reset signals (sl_tx_rst_n and sl_rx_rst_n).

b. Disabling SERDES.

c. Triggering PMA analog reset.

d. Changing the transceiver TX bit/refclk ratio to 25G (based on 156.25 MHz refclk).

e. Changing the transceiver RX bit/refclk ratio to 25G (based on 156.25 MHz refclk).

f. Reconfiguring the registers for the Ethernet, RS-FEC, and transceiver blocks.

g. Adjusting the phase offset of a recovered clock.

h. Enabling SERDES.

i. Enabling SERDES loopback.

j. Deasserting the reset signals (sl_tx_rst_n and sl_rx_rst_n).

k. Waiting 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).

l. Continuously sending packets for the CDR RX deskew training and waiting until PIO_OUT[4] = 0x1 (o_rx_ptp_ready asserted).

m. Clearing Ethernet statistic counters.

n. Enabling the packet generator to send packets of data, checking the TX packet count statistic counter to ensure all the packets are sent, and stopping the packet generator.

o. Checking for expected packets to be received by the packet checker.

6. Performing DR test from 25G PTP without RS-FEC to 25G PTP with RS-FEC:

a. Asserting reset signals (sl_tx_rst_n and sl_rx_rst_n).

b. Disabling SERDES.

c. Triggering PMA analog reset.

d. Reconfiguring registers for the Ethernet, RS-FEC, and transceiver blocks.

e. Adjusting the phase offset of a recovered clock.

f. Enabling SERDES.
g. Enabling SERDES loopback.

h. Deasserting the reset signals (sl_tx_rst_n and sl_rx_rst_n).

i. Waiting for \text{PIO\_OUT}[3:0] = 0xF (o\_tx\_ptp\_ready, o\_sl\_rx\_pcs\_ready, o\_sl\_rx\_block\_lock, and o\_ehip\_ready asserted).

j. Continuously sending packets for the CDR RX deskew training and waiting until \text{PIO\_OUT}[4] = 0x1 (o\_rx\_ptp\_ready asserted).

k. Clearing Ethernet statistic counters.

l. Enabling the packet generator to send packets of data, checking the TX packet count statistic counter to ensure all the packets are sent, and stopping the packet generator.

m. Checking for expected packets to be received by the packet checker.

7. Performing DR test from 25G PTP with RS-FEC to 10G PTP:
   a. Asserting sl\_tx\_rst\_n and sl\_rx\_rst\_n resets.
   b. Disabling SERDES.
   c. Triggering PMA analog reset.
   d. Changing the transceiver TX bit/refclk ratio to 10G (based on 156.25 MHz refclk).
   e. Changing the transceiver RX bit/refclk ratio to 10G (based on 156.25 MHz refclk).
   f. Reconfiguring registers for the Ethernet, RS-FEC, and transceiver blocks.
   g. Adjusting the phase offset of a recovered clock.
   h. Enabling SERDES.
   i. Enabling SERDES loopback.
   j. Deasserting the reset signals (sl\_tx\_rst\_n and sl\_rx\_rst\_n).
   k. Waiting for \text{PIO\_OUT}[3:0] = 0xF (o\_tx\_ptp\_ready, o\_sl\_rx\_pcs\_ready, o\_sl\_rx\_block\_lock, and o\_ehip\_ready asserted).
   l. Continuously sending packets for the CDR RX deskew training and waiting until \text{PIO\_OUT}[4] = 0x1 (o\_rx\_ptp\_ready asserted).
   m. Clearing Ethernet statistic counters.
   n. Enabling the packet generator to send packets of data, checking the TX packet count statistic counter to ensure all the packets are sent, and stopping the packet generator.
   o. Checking for expected packets to be received by the packet checker.

8. Performing DR test from 10G PTP to 25G PTP with RS-FEC:
   a. Asserting sl\_tx\_rst\_n and sl\_rx\_rst\_n resets.
   b. Disabling SERDES.
   c. Triggering PMA analog reset.
   d. Changing the transceiver TX bit/refclk ratio to 25G (based on 156.25 MHz refclk).
   e. Changing the transceiver RX bit/refclk ratio to 25G (based on 156.25 MHz refclk).
f. Reconfiguring registers for the Ethernet, RS-FEC, and transceiver blocks.
g. Adjusting the phase offset of a recovered clock.
h. Enabling SERDES.
i. Enabling SERDES loopback.
j. Deasserting the reset signals (si_tx_rst_n and sl_rx_rst_n).
k. Waiting 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).
l. Continuously sending packets for the CDR RX deskew training and waiting until PIO_OUT[4] = 0x1 (o_rx_ptp_ready asserted).
m. Clearing Ethernet statistic counters.
n. Enabling the packet generator to send packets of data, checking the TX packet count statistic counter to ensure all the packets are sent, and stopping the packet generator.
o. Checking for expected packets to be received by the packet checker.

9. Displaying Simulation PASSED.

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

```verbatim
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
```

3.2.2.2. 10GE/25GE MAC+PCS with RS-FEC Simulation Dynamic Reconfiguration Design Example Components

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

1. **Ethernet Protocol** as DR Protocol.
2. Under the 10G/25G Ethernet Protocol tab:
   a. **25G RS-FEC** as Select DR Design.
   b. **Other Development Kits** as the target development kit.
The successful test run displays output confirming the following behavior:

1. **Asserting all reset signals and deasserting** \( \text{sl}_\text{csr}_\text{rst}_\text{n} \), \( \text{sl}_\text{tx}_\text{rst}_\text{n} \), and \( \text{sl}_\text{rx}_\text{rst}_\text{n} \) signals.

2. Performing internal loopback test:
   a. Enabling SERDES.
   b. Waiting for \( \text{PIO.OUT}[3:0] = 0x7 \), \( \text{o}_\text{sl}_\text{rx}_\text{pcs}\text{ready} \), \( \text{o}_\text{sl}_\text{rx}_\text{block}\text{lock} \), and \( \text{o}_\text{ehip}_\text{ready} \) asserted.
   c. Clearing Ethernet statistic counters.
   d. Enabling the packet generator to send packets of data, checking the transmitter (TX) packet count statistic counter to ensure all the packets are sent, and stopping the packet generator.
   e. Checking for expected packets to be received by the packet checker.

3. Performing dynamic reconfiguration (DR) test from 25G with RS-FEC to 25G without RS-FEC:
   a. Asserting reset signals \( \text{sl}_\text{tx}_\text{rst}_\text{n} \) and \( \text{sl}_\text{rx}_\text{rst}_\text{n} \).
   b. Disabling SERDES. Use PMA attribute code 0x0001 in the *E-tile Transceiver PHY User Guide: PMA Attribute Codes* section.
   c. Triggering PMA analog reset. For more information about register descriptions, refer to the *E-tile Transceiver PHY User Guide*.
   d. Reconfiguring 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*. 
3. E-Tile Dynamic Reconfiguration Design Example

47
e. Changing the transceiver RX bit/refclk ratio to 25G (based on 156.25 MHz refclk).

f. Reconfiguring registers for the Ethernet, RS-FEC, and transceiver blocks.

g. Adjusting the phase offset of a recovered clock.

h. Enabling SERDES.

i. Enabling SERDES loopback.

j. Deasserting the reset signals (sl_tx_rst_n and sl_rx_rst_n).

k. Waiting for PIO_OUT[4:0] = 0x7 (o_sl_rx_pcs_ready, o_sl_rx_block_lock, and o_ehip_ready asserted).

l. Clearing Ethernet statistic counters.

m. Enabling the packet generator to send packets of data, checking the TX packet count statistic counter to ensure all the packets are sent, and stopping the packet generator.

n. Checking for expected packets to be received by the packet checker.

6. Performing DR test from 25G without RS-FEC to 25G with RS-FEC:

a. Asserting reset signals (sl_tx_rst_n and sl_rx_rst_n).

b. Disabling SERDES.

c. Triggering PMA analog reset.

d. Reconfiguring registers for the Ethernet, RS-FEC, and transceiver blocks.

e. Adjusting the phase offset of a recovered clock.

f. Enabling SERDES.

g. Enabling SERDES loopback.

h. Deasserting the reset signals (sl_tx_rst_n and sl_rx_rst_n).

i. Waiting for PIO_OUT[4:0] = 0x7 (o_sl_rx_pcs_ready, o_sl_rx_block_lock, and o_ehip_ready asserted).

j. Clearing Ethernet statistic counters.

k. Enabling the packet generator to send packets of data, checking the TX packet count statistic counter to ensure all the packets are sent, and stopping the packet generator.

l. Checking for expected packets to be received by the packet checker.

7. Performing DR test from 25G with RS-FEC to 10G:

a. Asserting reset signals (sl_tx_rst_n and sl_rx_rst_n).

b. Disabling SERDES.

c. Triggering PMA analog reset.

d. Changing the transceiver TX bit/refclk ratio to 10G (based on 156.25 MHz refclk).

e. Changing the transceiver RX bit/refclk ratio to 10G (based on 156.25 MHz refclk).

f. Reconfiguring registers for the Ethernet, RS-FEC, and transceiver blocks.

g. Adjusting the phase offset of a recovered clock.
h. Enabling SERDES.
   i. Enabling SERDES loopback.
   j. Deasserting the reset signals (sl_tx_rst_n and sl_rx_rst_n).
   k. Waiting for PIO_OUT[4:0] = 0x7 (o_sl_rx_pcs_ready, o_sl_rx_block_lock, and o_ehip_ready asserted).
   l. Clearing Ethernet statistic counters.
   m. Enabling the packet generator to send packets of data, checking the TX packet count statistic counter to ensure all the packets are sent, and stopping the packet generator.
   n. Checking for expected packets to be received by the packet checker.

8. Performing DR test from 10G to 25G with RS-FEC:
   a. Asserting reset signals (sl_tx_rst_n and sl_rx_rst_n).
   b. Disabling SERDES.
   c. Triggering PMA analog reset.
   d. Changing the transceiver TX bit/refclk ratio to 25G (based on 156.25 MHz refclk).
   e. Changing the transceiver RX bit/refclk ratio to 25G (based on 156.25 MHz refclk).
   f. Reconfiguring registers for the Ethernet, RS-FEC, and transceiver blocks.
   g. Adjusting the phase offset of a recovered clock.
   h. Enabling SERDES.
   i. Enabling SERDES loopback.
   j. Deasserting the reset signals (sl_tx_rst_n and sl_rx_rst_n).
   k. Waiting for PIO_OUT[4:0] = 0x7 (o_sl_rx_pcs_ready, o_sl_rx_block_lock, and o_ehip_ready asserted).
   l. Clearing Ethernet statistic counters.
   m. Enabling the packet generator to send packets of data, checking the TX packet count statistic counter to ensure all the packets are sent, and stopping the packet generator.
   n. Checking for expected packets to be received by the packet checker.

9. Displaying Simulation PASSED.

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

```plaintext
# 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
```
3.3. 25G Ethernet to CPRI Dynamic Reconfiguration Design Example

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

<table>
<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>24GE CPRI with RS-FEC</td>
<td>10GE CPRI</td>
</tr>
<tr>
<td>9.8GE CPRI</td>
<td>4.9GE CPRI</td>
</tr>
<tr>
<td>2.4GE CPRI</td>
<td></td>
</tr>
</tbody>
</table>

3.3.1. Functional Description

3.3.1.1. Clocking Scheme

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

3.3.2. Simulation Design Examples

3.3.2.1. 25GE MAC+PCS with RS-FEC and PTP to CPRI Simulation Dynamic Reconfiguration Design Example Components

The simulation block diagram below is generated using the following settings in the IP parameter editor:
1. **25G Ethernet to CPRI Protocol** as DR Protocol.
2. Under the **25G Ethernet to CPRI Protocol** tab:
   a. **25G PTP RS-FEC** as Select DR Design.
   b. **Other Development Kits** as the target development kit.

**Figure 26. Simulation Block Diagram for E-Tile Ethernet IP for Intel Agilex FPGA 25G Ethernet to CPRI Dynamic Reconfiguration Design Example**

The successful test run displays output confirming the following behavior:
1. Asserting all reset signals and deasserting `csr_rst_n`, `sl_tx_rst_n`, and `sl_rx_rst_n` signals.
2. Performing internal loopback test:
   a. Waiting for `PIO_OUT[3:0] = 0x1 (o_ehip_ready asserted).
   b. Enabling SERDES loopback.
   c. Waiting for `PIO_OUT[3:0] = 0xF (o_sl_tx_ptp_ready, o_sl_rx_pcs_ready, o_sl_rx_block_lock, and o_ehip_ready asserted).
   d. Continuously sending packets for the clock data recover (CDR) receiver (RX) deskew training and waiting until `PIO_OUT[4] = 0x1 (o_sl_rx_ptp_ready asserted).
3. Performing dynamic reconfiguration (DR) test from 25G PTP with RS-FEC to 24G CPRI with RS-FEC:
   a. Asserting reset signals (sl_tx_rst_n and sl_rx_rst_n).
   b. Disabling SERDES. Use PMA attribute code 0x0001 in the *E-tile Transceiver PHY User Guide: PMA Attribute Codes* section.
   c. Performing reference clock mux switching. For more information about the details of the changed register values, refer to the c3_reconfig.c file.
      i. Switching the PMA controller clock to the transceiver refclk1 clock.
      ii. Changing refclk reference clock from 156 MHz (i_clk_ref[0]) to 184.32 MHz (i_clk_ref[1]).
      iii. Switching the PMA controller clock to the transceiver refclk0 clock.
   Note: Steps i and iii are only applicable for Ethernet dynamic reconfiguration hardware test to avoid potential hardware glitch due to the reference clock switch operation. These steps are available in the hardware test code but skip in the simulation test code.
   d. Triggering PMA analog reset. For more information about register descriptions, refer to the *E-tile Transceiver PHY User Guide*.
   e. Reconfiguring 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*.
   f. Adjusting the phase offset of a recovered clock. Use PMA attribute code 0x000E in the *E-tile Transceiver PHY User Guide: PMA Attribute Codes* section.
   g. Enabling SERDES. Use PMA attribute code 0x0001 in the *E-tile Transceiver PHY User Guide: PMA Attribute Codes* section.
   h. Enabling SERDES loopback. Use PMA attribute code 0x0008 in the *E-tile Transceiver PHY User Guide: PMA Attribute Codes* section.

4. Performing dynamic reconfiguration (DR) test from 24G CPRI with RS-FEC to 25G PTP with RS-FEC:
   a. Asserting reset signals (sl_tx_rst_n and sl_rx_rst_n).
   b. Disabling SERDES.
   c. Performing reference clock mux switching. For more information about the details of the changed register values, refer to the c3_reconfig.c file.
      i. Switching the PMA controller clock to the transceiver refclk1 clock.
      ii. Changing refclk reference clock from 184.32 MHz (i_clk_ref[1]) to 156 MHz (i_clk_ref[0]).
      iii. Switching the PMA controller clock to the transceiver refclk0 clock.
3. E-Tile Dynamic Reconfiguration Design Example

Note: Steps i and iii are only applicable for Ethernet dynamic reconfiguration hardware test to avoid potential hardware glitch due to the reference clock switch operation. These steps are available in the hardware test code but skip in the simulation test code.

d. Triggering PMA analog reset.
e. Reconfiguring the registers for the Ethernet, RS-FEC, and transceiver blocks.
f. Adjusting the phase offset of a recovered clock.
g. Enabling SERDES.
h. Enabling SERDES loopback.

5. Performing dynamic reconfiguration (DR) test from 25G PTP with RS-FEC to 10G CPRI:

a. Asserting reset signals (sl_tx_rst_n and sl_rx_rst_n).
b. Disabling SERDES.
c. Performing reference clock mux switching. For more information about the details of the changed register values, refer to the c3_reconfig.c file.
   i. Switching the PMA controller clock to the transceiver refclk1 clock.
   ii. Changing refclk reference clock from 156 MHz (i_clk_ref[0]) to 184.32 MHz (i_clk_ref[1]).
   iii. Switching the PMA controller clock to the transceiver refclk0 clock.

   Note: Steps i and iii are only applicable for Ethernet dynamic reconfiguration hardware test to avoid potential hardware glitch due to the reference clock switch operation. These steps are available in the hardware test code but skip in the simulation test code.

d. Triggering PMA analog reset.
e. Reconfiguring the registers for the Ethernet, RS-FEC, and transceiver blocks.
f. Adjusting the phase offset of a recovered clock.
g. Enabling SERDES.
h. Enabling SERDES loopback.

6. Performing dynamic reconfiguration (DR) test from 10G CPRI to 25G PTP with RS-FEC:

a. Asserting reset signals (sl_tx_rst_n and sl_rx_rst_n).
b. Disabling SERDES.
c. Performing reference clock mux switching. For more information about the details of the changed register values, refer to the c3_reconfig.c file.
   i. Switching the PMA controller clock to the transceiver refclk1 clock.
   ii. Changing refclk reference clock from 184.32 MHz (i_clk_ref[1]) to 156 MHz (i_clk_ref[0]).
   iii. Switching the PMA controller clock to the transceiver refclk0 clock.

   Note: Steps i and iii are only applicable for Ethernet dynamic reconfiguration hardware test to avoid potential hardware glitch due to the reference clock switch operation. These steps are available in the hardware test code but skip in the simulation test code.
d. Triggering PMA analog reset.
e. Reconfiguring the registers for the Ethernet, RS-FEC, and transceiver blocks.
f. Adjusting the phase offset of a recovered clock.
g. Enabling SERDES.
h. Enabling SERDES loopback.

7. Performing dynamic reconfiguration (DR) test from 25G PTP with RS-FEC to 9.8G CPRI:
   a. Asserting reset signals (sl_tx_rst_n and sl_rx_rst_n).
   b. Disabling SERDES.
   c. Performing reference clock mux switching. For more information about the
details of the changed register values, refer to the c3_reconfig.c file.
      i. Switching the PMA controller clock to the transceiver refclk1 clock.
      ii. Changing refclk reference clock from 156 MHz (i_clk_ref[0]) to
          153.6 MHz (i_clk_ref[2]).
      iii. Switching the PMA controller clock to the transceiver refclk0 clock.

   Note: Steps i and iii are only applicable for Ethernet dynamic reconfiguration
   hardware test to avoid potential hardware glitch due to the reference
clock switch operation. These steps are available in the hardware test
code but skip in the simulation test code.

d. Triggering PMA analog reset.
e. Reconfiguring the registers for the Ethernet, RS-FEC, and transceiver blocks.
f. Adjusting the phase offset of a recovered clock.
g. Asserting EMIB reset. Use PMA register 0x400E2 in the E-tile Transceiver PHY
   User Guide: PMA Control and Status Registers section.
h. Deasserting EMIB reset. Use PMA register 0x400E2 in the E-tile Transceiver
   PHY User Guide: PMA Control and Status Registers section.
i. Enabling SERDES.
j. Enabling SERDES loopback.
k. Deasserting reset signals sl_tx_rst_n and sl_rx_rst_n.

8. Performing dynamic reconfiguration (DR) test from 9.8G CPRI to 25G PTP with RS-
   FEC:
   a. Asserting reset signals (sl_tx_rst_n and sl_rx_rst_n).
   b. Disabling SERDES.
   c. Performing reference clock mux switching. For more information about the
details of the changed register values, refer to the c3_reconfig.c file.
      i. Switching the PMA controller clock to the transceiver refclk1 clock.
      ii. Changing refclk reference clock from 153.6 MHz (i_clk_ref[2]) to
          156 MHz (i_clk_ref[0]).
      iii. Switching the PMA controller clock to the transceiver refclk0 clock.
Note: Steps i and iii are only applicable for Ethernet dynamic reconfiguration hardware test to avoid potential hardware glitch due to the reference clock switch operation. These steps are available in the hardware test code but skip in the simulation test code.

d. Triggering PMA analog reset.

e. Reconfiguring the registers for the Ethernet, RS-FEC, and transceiver blocks.

f. Adjusting the phase offset of a recovered clock.

g. Asserting EMIB reset.

h. Deasserting EMIB reset.

i. Enabling SERDES.

j. Enabling SERDES loopback.

k. Deasserting reset signals sl_tx_rst_n and sl_rx_rst_n.

9. Performing dynamic reconfiguration (DR) test from 25G PTP with RS-FEC to 4.9G CPRI:

a. Asserting reset signals (sl_tx_rst_n and sl_rx_rst_n).

b. Disabling SERDES.

c. Performing reference clock mux switching. For more information about the details of the changed register values, refer to the c3_reconfig.c file.

i. Switching the PMA controller clock to the transceiver refclk1 clock.

ii. Changing refclk reference clock from 156 MHz (i_clk_ref[0]) to 153.6 MHz (i_clk_ref[2]).

iii. Switching the PMA controller clock to the transceiver refclk0 clock.

Note: Steps i and iii are only applicable for Ethernet dynamic reconfiguration hardware test to avoid potential hardware glitch due to the reference clock switch operation. These steps are available in the hardware test code but skip in the simulation test code.

d. Triggering PMA analog reset.

e. Reconfiguring the registers for the Ethernet, RS-FEC, and transceiver blocks.

f. Adjusting the phase offset of a recovered clock.

g. Asserting EMIB reset.

h. Deasserting EMIB reset.

i. Enabling SERDES.

j. Enabling SERDES loopback.

k. Deasserting reset signals sl_tx_rst_n and sl_rx_rst_n.

10. Performing dynamic reconfiguration (DR) test from 4.9G CPRI to 25G PTP with RS-FEC:

a. Asserting reset signals (sl_tx_rst_n and sl_rx_rst_n).

b. Disabling SERDES.

c. Performing reference clock mux switching. For more information about the details of the changed register values, refer to the c3_reconfig.c file.
i. Switching the PMA controller clock to the transceiver refclk1 clock.

ii. Changing refclk reference clock from 153.6 MHz (i_clk_ref[2]) to 156 MHz (i_clk_ref[0]).

iii. Switching the PMA controller clock to the transceiver refclk0 clock.

Note: Steps i and iii are only applicable for Ethernet dynamic reconfiguration hardware test to avoid potential hardware glitch due to the reference clock switch operation. These steps are available in the hardware test code but skip in the simulation test code.

d. Triggering PMA analog reset.

e. Reconfiguring the registers for the Ethernet, RS-FEC, and transceiver blocks.

f. Adjusting the phase offset of a recovered clock.

g. Asserting EMIB reset.

h. Deasserting EMIB reset.

i. Enabling SERDES.

j. Enabling SERDES loopback.

k. Deasserting reset signals sl_tx_rst_n and sl_rx_rst_n.

11. Performing dynamic reconfiguration (DR) test from 25G PTP with RS-FEC to 2.4G CPRI:

   a. Asserting reset signals (sl_tx_rst_n and sl_rx_rst_n).

   b. Disabling SERDES.

   c. Performing reference clock mux switching. For more information about the details of the changed register values, refer to the c3_reconfig.c file.

      i. Switching the PMA controller clock to the transceiver refclk1 clock.

      ii. Changing refclk reference clock from 156 MHz (i_clk_ref[0]) to 153.6 MHz (i_clk_ref[2]).

      iii. Switching the PMA controller clock to the transceiver refclk0 clock.

      Note: Steps i and iii are only applicable for Ethernet dynamic reconfiguration hardware test to avoid potential hardware glitch due to the reference clock switch operation. These steps are available in the hardware test code but skip in the simulation test code.

   d. Triggering PMA analog reset.

   e. Reconfiguring the registers for the Ethernet, RS-FEC, and transceiver blocks.

   f. Adjusting the phase offset of a recovered clock.

   g. Asserting EMIB reset.

   h. Deasserting EMIB reset.

   i. Enabling SERDES.

   j. Enabling SERDES loopback.

   k. Deasserting reset signals sl_tx_rst_n and sl_rx_rst_n.

12. Performing dynamic reconfiguration (DR) test from 2.4G CPRI to 25G PTP with RS-FEC:
a. Asserting reset signals (sl_tx_rst_n and sl_rx_rst_n).
b. Disabling SERDES.
c. Performing reference clock mux switching. For more information about the details of the changed register values, refer to the c3_reconfig.c file.
   i. Switching the PMA controller clock to the transceiver refclk1 clock.
   ii. Changing refclk reference clock from 153.6 MHz (i_clk_ref[2]) to 156 MHz (i_clk_ref[0]).
   iii. Switching the PMA controller clock to the transceiver refclk0 clock.
   Note: Steps i and iii are only applicable for Ethernet dynamic reconfiguration hardware test to avoid potential hardware glitch due to the reference clock switch operation. These steps are available in the hardware test code but skip in the simulation test code.
d. Triggering PMA analog reset.
e. Reconfiguring the registers for the Ethernet, RS-FEC, and transceiver blocks.
f. Adjusting the phase offset of a recovered clock.
g. Asserting EMIB reset.
h. Deasserting EMIB reset.
i. Enabling SERDES.
j. Enabling SERDES loopback.
k. Deasserting reset signals sl_tx_rst_n and sl_rx_rst_n.

3.4. 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>2019.10.18</td>
<td>19.3</td>
<td>19.3.0</td>
<td>Initial release.</td>
</tr>
</tbody>
</table>