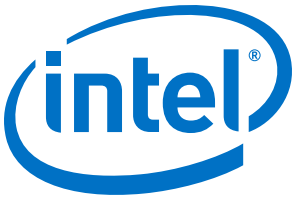


25G Ethernet Intel Stratix 10 FPGA IP Design Example User Guide

Updated for Intel® Quartus® Prime Design Suite: **18.1**

Subscribe
Send Feedback

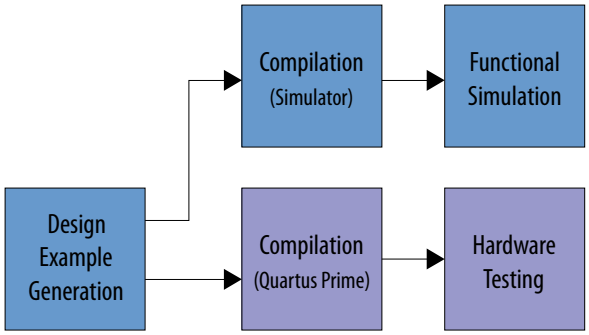
UG-20110 | 2019.01.07
Latest document on the web: PDF | HTML

Contents

1. 25G Ethernet Intel FPGA IP Quick Start Guide

The Intel® 25G Ethernet (25GbE) Intel FPGA IP core for Intel Stratix® 10 devices provides the capability of generating design examples for selected configurations.

Figure 1. Development Stages for the Design Example

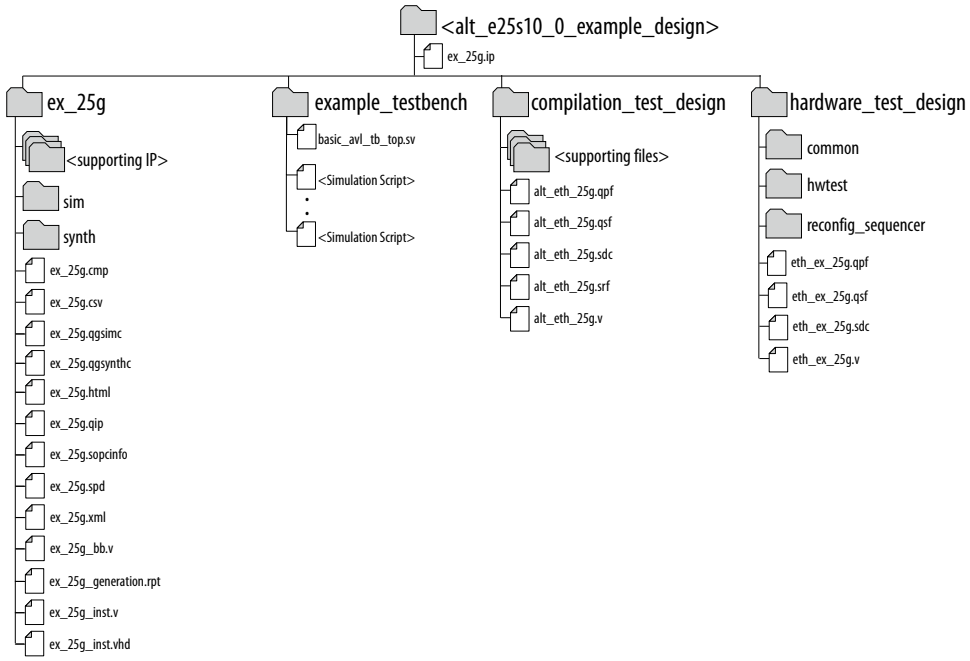


Related Information

- 25G Ethernet Single-Channel Design Example for Intel Stratix 10 Devices on page 24
  Provides details for the 25G Ethernet single-channel design example.
- 25G Ethernet Multi-Channel Design Example for Intel Stratix 10 Devices on page 35
  Provides details for the 25G Ethernet multi-channel design example.
- 10G/25G Ethernet Single-Channel Design Example for Intel Stratix 10 Devices on page 10
  Provides details for the 10G/25G Ethernet single-channel design example.

1.1. Directory Structure

Figure 2. Directory Structure for the 25G and 10G/25G Ethernet Design Examples



- The simulation files (testbench for simulation only) are located in `<design_example_dir>/example_testbench`.
- The compilation-only design example is located in `<design_example_dir>/compilation_test_design`.
- The hardware configuration and test files (the design example in hardware) are located in `<design_example_dir>/hardware_test_design`.

Table 1. Directory and File Descriptions

File Names	Description
`eth_ex_25g.qpf`	Intel Quartus® Prime project file.
`eth_ex_25g.qsf`	Intel Quartus Prime project settings file.
`eth_ex_25g.sdc`	Synopsys Design Constraints file. You can copy and modify this file for your own 25GbE Intel FPGA IP core design.
`eth_ex_25g.v`	Top-level Verilog HDL design example file. Single-channel design uses Verilog file.
`common/`	Hardware design example support files.
`hwtest/main.tcl`	Main file for accessing System Console.

1.2. Generating the Design Example

Figure 3. Procedure

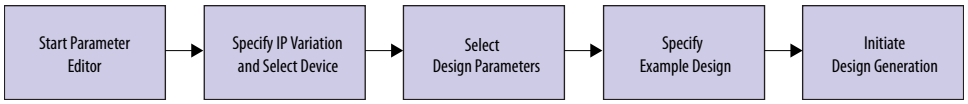
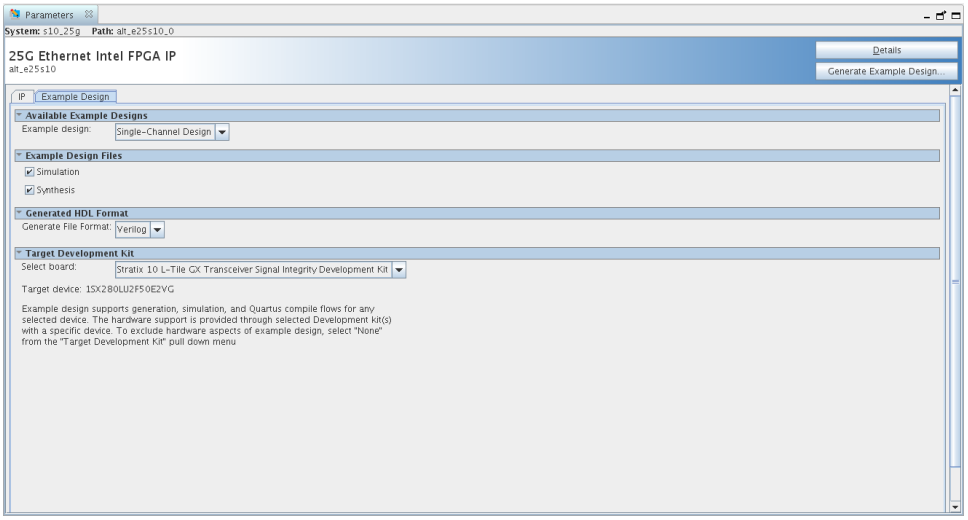


Figure 4. Example Design Tab in the 25G Ethernet Intel FPGA IP Parameter Editor



Follow these steps to generate the hardware design example and testbench:

1. In the Intel Quartus Prime Pro Edition software, click **File ➤ New Project Wizard** to create a new Quartus Prime project, or **File ➤ Open Project** to open an existing Quartus Prime project. The wizard prompts you to specify a device.
2. In the IP Catalog, locate and select **25G Ethernet Intel FPGA IP**. The **New IP Variation** window appears.
3. Specify a top-level name for your IP variation and click **OK**. The parameter editor adds the top-level `.ip` file to the current project automatically. If you are prompted to manually add the `.ip` file to the project, click **Project ➤ Add/ Remove Files in Project** to add the file.
4. In the Intel Quartus Prime Pro Edition software, you must select a specific Intel Stratix 10 device in the **Device** field, or keep the default device the Quartus Prime software proposes.

   Note: The hardware design example overwrites the selection with the device on the target board. You specify the target board from the menu of design example options in the **Example Design** tab (Step 8).
5. Click **OK**. The parameter editor appears.
6. On the **IP** tab, specify the parameters for your IP core variation.
7. On the **Example Design** tab, for **Example Design Files**, select the **Simulation** option to generate the testbench, and select the **Synthesis** option to generate the hardware design example. Only Verilog HDL files are generated.

Note: A functional VHDL IP core is not available. Specify Verilog HDL only, for your IP core design example.

8. For **Target Development Kit** select the **Intel Stratix 10 L-Tile GX Transceiver Signal Integrity Development Kit**.

   Note: Contact your Intel FPGA representative for information about a platform suitable to run this hardware example.

9. Click **Generate Example Design**. The **Select Example Design Directory** window appears.
10. If you want to modify the design example directory path or name from the defaults displayed (`alt_e25s10_0_example_design`), browse to the new path and type the new design example directory name (<*design_example_dir*>).
11. Click **OK**.

1.2.1. Design Example Parameters

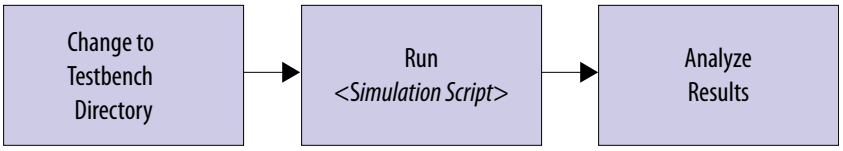
Table 3. Parameters in the Example Design Tab

Parameter	Description
Example Design	Available example designs for the IP parameter settings.
Example Design Files	The files to generate for the different development phase. • Simulation—generates the necessary files for simulating the example design. • Synthesis—generates the synthesis files. Use these files to compile the design in the Intel Quartus Prime Pro Edition software for hardware testing and perform static timing analysis.
Generate File Format	The format of the RTL files for simulation—Verilog.
Select Board	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. **Intel Stratix 10 L-Tile GX Transceiver Signal Integrity Development Kit:** This option allows you to test the design example on the selected Intel FPGA IP development kit. This option automatically selects the *Target Device* to match the device on the Intel FPGA IP development kit. If your board revision has a different device grade, you can change the target device. **None**: This option excludes the hardware aspects for the design example.

1.3. Simulating the 25G Ethernet Intel FPGA IP Design Example Testbench

1.3.1. Procedure

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



1. At the command prompt, change the working directory to `<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.

Table 4. Steps to Simulate the Testbench

Simulator	Instructions
ModelSim*	In the command line, type `vsim do run_vsim.do` *Note:* The ModelSim-AE and ModelSim-ASE simulatorscannot simulate this IP core. You must use another supported ModelSim simulator such as ModelSim SE.
VCS*	In the command line, type `sh run_vcs.sh`
NCSim	In the command line, type `sh run_ncsim.sh`
Xcelium*	In the command line, type `sh run_xcelium.sh`

A successful simulation ends with the following message:

```
Simulation Passed.
```

or

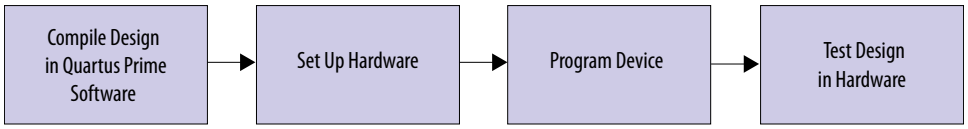
```
Testbench complete.
```

After successful completion, you can analyze the results.

1.4. Compiling and Configuring the Design Example in Hardware

The 25G Ethernet Intel FPGA IP core parameter editor allows you to compile and configure the design example on a target development kit.

1.4.1. Procedure



To compile and configure a design example on hardware, follow these steps:

1. Launch the Intel Quartus Prime Pro Edition software and select **Processing ➤ Start Compilation** to compile the design.
2. After you generate an SRAM object file `.sof`, follow these steps to program the hardware design example on the Intel Stratix 10 device:
   a. On the **Tools** menu, click **Programmer**.
   b. In the Programmer, click **Hardware Setup**.
   c. Select a programming device.
   d. Select and add the Intel Stratix 10 GX board to your Intel Quartus Prime Pro Edition session.

e. Ensure that **Mode** is set to **JTAG**.

f. Select the Intel Stratix 10 device and click **Add Device**. The Programmer displays a block diagram of the connections between the devices on your board.

g. In the row with your `.sof`, check the box for the `.sof`.

h. Check the box in the **Program/Configure** column.

i. Click **Start**.

Note: This design targets the Intel Stratix 10 device. Please contact your Intel FPGA representative to inquire about a platform suitable to run this hardware example.

Related Information

- Incremental Compilation for Hierarchical and Team-Based Design
- Programming Intel FPGA Devices

1.5. Testing the 25G Ethernet Intel FPGA IP Design in Hardware

1.5.1. Procedure

After you compile the 25G Ethernet Intel FPGA IP core design example and configure it on your Intel Stratix 10 device, you can use the System Console to program the IP core and its embedded Native PHY IP core registers.

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

1. In the Intel Quartus Prime Pro Edition software, select **Tools ➤ System Debugging Tools ➤ System Console** to launch the system console.
2. In the Tcl Console pane, type `cd hwtest` to change directory to `/hardware_test_design/hwtest`.
3. Type `source main.tcl` to open a connection to the JTAG master.

You can now run any of the predefined hardware tests from the System Console. Observe the test results displayed.

Related Information

Analyzing and Debugging Designs with System Console

2. 10G/25G Ethernet Single-Channel Design Example for Intel Stratix 10 Devices

The 10G/25G Ethernet single-channel design example demonstrates an Ethernet solution for Intel Stratix 10 devices using the 25G Ethernet Intel FPGA IP core.

Generate the design example from the **Example Design** tab of the 25G Ethernet Intel FPGA IP parameter editor. You can choose to generate the design with or without the IEEE 1588v2 feature. You can also choose to generate the design with or without the Reed-Solomon Forward Error Correction (RS-FEC) feature.

2.1. Features

- Supports single Ethernet channel operating at either 10G or 25G.
- Generate design example with IEEE 1588v2 feature.
- Generate design example with RS-FEC feature.
- Generates design example separately from Intel Stratix 10 Transceiver Native PHY.
- Provides testbench and simulation script.

2.2. Hardware and Software Requirements

Intel uses the following hardware and software to test the design example in a Linux system:

- Intel Quartus Prime Pro Edition software
- ModelSim-SE, NCSim (Verilog only), VCS, and Xcelium simulator
- Intel Stratix 10 L-Tile GX Transceiver Signal Integrity Development Kit (1SX280LU2F50E2VG) for hardware testing

2.3. Functional Description

The 10G/25G Ethernet single-channel design example consists of two core variants—MAC+PCS+PMA and MAC+PCS. The following block diagrams show the design components and the top-level signals of the two core variants in the 10G/25G Ethernet single-channel design example.

Figure 5. Block Diagram—10G/25G Ethernet Single-Channel Design Example (MAC +PCS+PMA Core Variant) Without the IEEE 1588v2 Feature

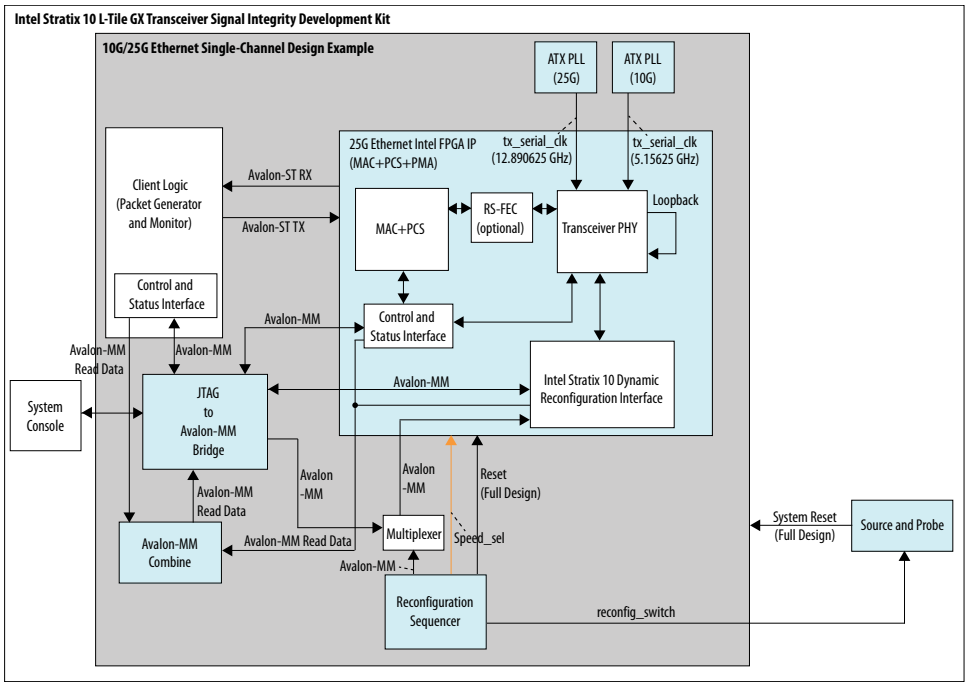


Figure 6. Block Diagram—10G/25G Ethernet Single-Channel Design Example (MAC +PCS+PMA Core Variant) with the IEEE 1588v2 Feature

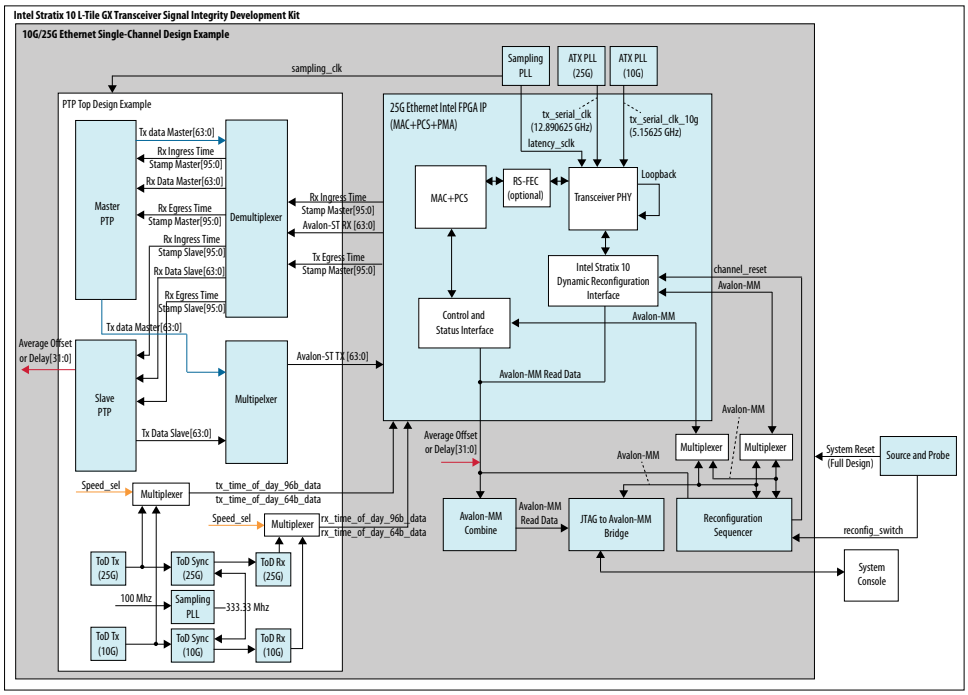
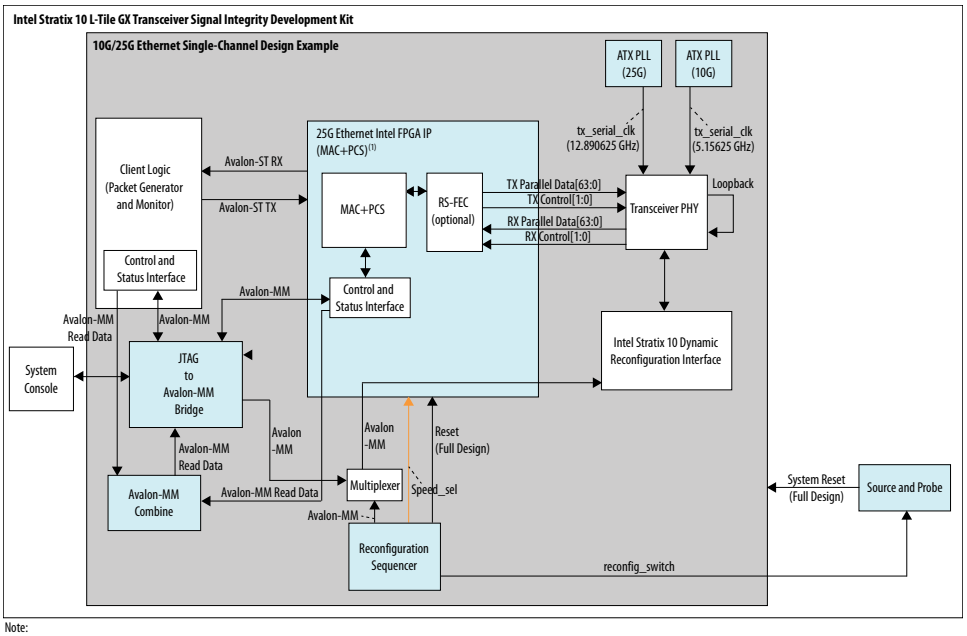
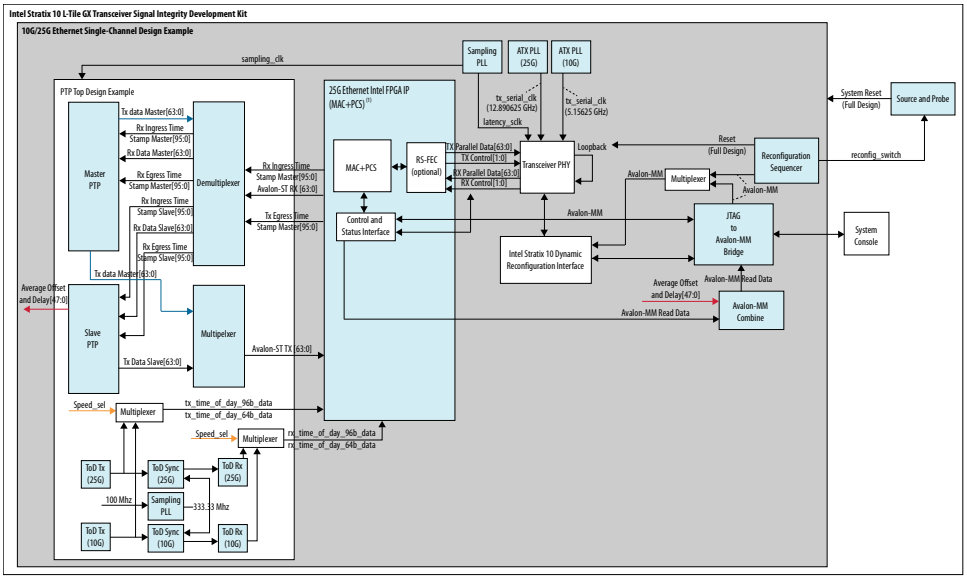


Figure 7. Block Diagram—10G/25G Ethernet Single-Channel Design Example (MAC +PCS Core Variant) Without the IEEE 1588v2 Feature



Note:
1. Components outside of this block are part of the design example only.

Figure 8. Block Diagram—10G/25G Ethernet Single-Channel Design Example (MAC +PCS Core Variant) with the IEEE 1588v2 Feature



Note:
1. Components outside of this block are part of the design example only.

2.3.1. Design Components

Table 5. Design Components

Component	Description
25G Ethernet Intel FPGA IP	The Consists of MAC, PCS, and Transceiver PHY, with the following configuration: • **Core Variant**: MAC+PCS+PMA • **Enable flow control**: Optional • **Enable link fault generation**: Optional • **Enable preamble passthrough**: Optional • **Enable statistics collection**: Optional • **Enable MAC statistics counters**: Optional • **Enable 10G/25G dynamic rate switching**: Selected • **Enable Enable Altera Debug Master Endpoint (ADME)**: Optional • **Reference clock frequency**: 644.531250/322.265625 For the design example with the IEEE 1588 feature, the following additional parameters are configured: • **Enable IEEE 1588**: Selected • **Time of day format**: Enable 96-bit timestamp format (1) For the design example with the RS-FEC feature, the following additional parameter is configured: • **Enable RS-FEC**: Selected For the design example with separated transceiver native PHY, the following additional parameter is configured: • **Core Variant**: MAC+PCS
Reconfiguration Sequencer	Reconfigures the transceiver channel speed from 10 Gbps to 25 Gbps, and vice versa.
ATX PLL	Generates TX serial clocks for the 10G and 25G transceivers.
Client logic	Consists of: • Traffic generator, which generates burst packets to the 25G Ethernet Intel FPGA IP core for transmission. • Traffic monitor, which receives burst packets from 25G Ethernet Intel FPGA IP core.
Source and Probe	Source and probe signals, including system reset input signal, which you can use for debugging.
Design Components for the IEEE 1588v2 Feature	
Sampling PLL	Generates the clocks for the IEEE 1588v2 design components. • `latency_sclk`: 156.25 MHz for latency measurement. • `sampling_clk`: 250 MHz for ToD synchronization
Time-of-day (ToD) Sync	Synchronizes the 10G and 25G ToDs.
ToD Tx	ToD for transmit paths for the 10G and 25G transceivers.
ToD Rx	ToD for receive paths for the 10G and 25G transceivers.
Master Precision Time Protocol (PTP)	Master PTP consists of a packet generator and a packet receiver. • Packet generator: Obtains timestamp information from 25G Ethernet Intel FPGA IP core and generates Avalon®-ST packets such as Sync packet and Delay Response packet. • Packet receiver: Obtains the delay request packet information from 25G Ethernet Intel FPGA IP core and produces timestamp values.
Slave PTP	Slave PTP consists of a packet generator, a packet receiver, and packet compute.

continued...

(1) The 10G/25G Ethernet single-channel design example with IEEE 1588v2 feature only supports 96-bit timestamp format.

Component	Description
	• Packet generator: Obtains timestamp information from 25G Ethernet Intel FPGA IP core and generates Avalon-ST packets such as Delay Request packet. • Packet receiver: Obtains the Sync and Delay Response packets information from 25G Ethernet Intel FPGA IP core and produces timestamp values. • Packet compute: Calculates and produces the delay and offsets value based on the timestamp values.

2.4. Simulation

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

Related Information

Simulating the 25G Ethernet Intel FPGA IP Design Example Testbench on page 7

2.4.1. Testbench

Figure 9. Block Diagram of the 10G/25G Ethernet Single-Channel Design Example Simulation Testbench

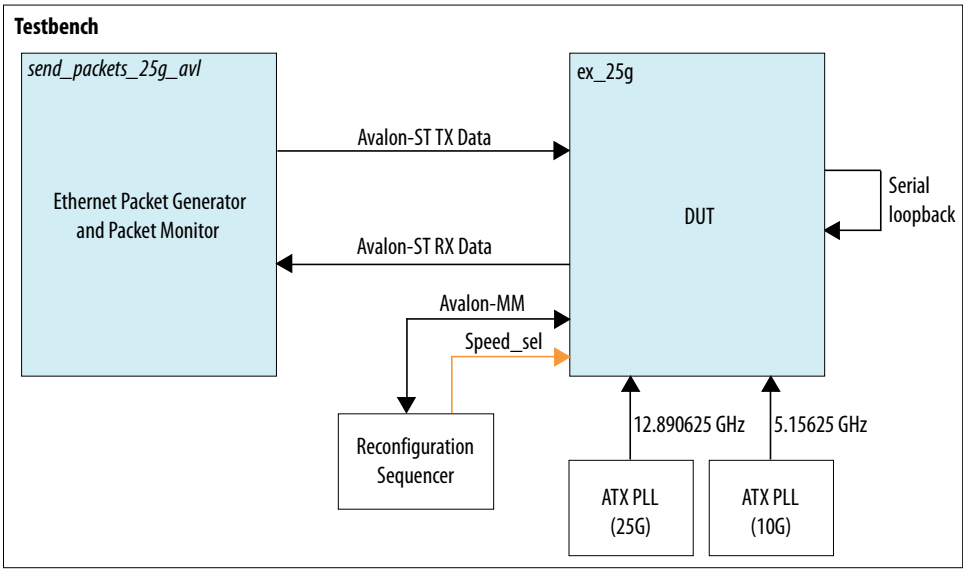


Table 6. Testbench Components

Component	Description
Device under test (DUT)	25G Ethernet Intel FPGA IP core.
Reconfiguration Sequencer	Reconfigures the transceiver channel speed from 10 Gbps to 25 Gbps, and vice versa.
Ethernet Packet Generator and Packet Monitor	Packet generator generates frames and transmit to the DUT. Packet Monitor monitors TX and RX datapaths and displays the frames in the simulator console.
ATX PLL	Generates a TX serial clock for the Intel Stratix 10 10G/25G transceiver which is wrapped in 25G Ethernet Intel FPGA IP core.

Note: For 10G/25G Ethernet single-channel design example with IEEE 1588v2 feature simulation testbench, refer to Figure 6 on page 11.

2.4.2. Simulation Design Example Components

Table 7. 10G/25G Ethernet Single-Channel Design Example Testbench File Descriptions

File Name	Description
Testbench and Simulation Files	
`basic_avl_tb_top..sv`	Top-level testbench file. The testbench instantiates the DUT, performs Avalon-MM configuration on design components and client logic, and sends and receives packet to or from 25G Ethernet Intel FPGA IP.
Testbench Scripts	
`run_vsim.do`	The ModelSim script to run the testbench.
`run_vcs.sh`	The Synopsys VCS script to run the testbench.
`run_ncsim.sh`	The Cadence NCSim script to run the testbench.
`run_xcelium.sh`	The Xcelium script to run the testbench.

2.4.3. Test Case—Design Example Without the IEEE 1588v2 Feature

The simulation test case performs the following actions:

1. Instantiates 25G Ethernet Intel FPGA IP and ATX PLL.
2. Starts up the design example with an operating speed of 25G.
3. Waits for RX clock and PHY status signal to settle.
4. Prints PHY status.
5. Sends and receives 10 valid data on 25G speed.
6. Performs channel reset and switches to 10G speed.
7. Waits for RX clock and PHY status signal to settle.
8. Prints PHY status.
9. Sends and receives another 10 valid data on 10G speed.
10. Performs channel reset and switches to 25G speed.
11. Waits for RX clock and PHY status signal to settle.
12. Prints PHY status.
13. Sends and receives another 10 valid data on 25G speed.
14. Analyzes the results. The successful testbench displays "`Simulation PASSED.`".

The following sample output illustrates a successful simulation test run:

```
Waiting for RX alignment
RX deskew locked
RX lane alignmnet locked
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...
** Received Packet           1...
** Received Packet           2...
** Sending Packet            9...
** Sending Packet           10...
** Received Packet           3...
** Received Packet           4...
** Received Packet           5...
** Received Packet           7...
** Received Packet           8...
** Received Packet           9...
** Received Packet          10...
Switching to 10G mode: 10G Reconfig start
Switching to 10G mode: 10G Reconfig End
Waiting for RX alignment
RX deskew locked
RX lane alignment locked
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...
** Received Packet           1...
** Received Packet           2...
** Sending Packet            9...
** Sending Packet           10...
** Received Packet           3...
** Received Packet           4...
** Received Packet           5...
** Received Packet           7...
** Received Packet           8...
** Received Packet           9...
** Received Packet          10...
Switching to 25G mode: 25G Reconfig start
Switching to 25G mode: 25G Reconfig End
Waiting for RX alignment
RX deskew locked
RX lane alignment locked
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...
** Received Packet           1...
** Received Packet           2...
** Sending Packet            9...
** Sending Packet           10...
** Received Packet           3...
** Received Packet           4...
** Received Packet           5...
** Received Packet           7...
** Received Packet           8...
** Received Packet           9...
** Received Packet          10...
**
** Testbench complete.
**
```

2.4.4. Test Case—Design Example with the IEEE 1588v2 Feature

Note: For 10G/25G Ethernet single-channel design example with IEEE 1588v2 feature simulation testbench, refer to Figure 6 on page 11.

The simulation test case performs the following actions:

1. Instantiates 25G Ethernet Intel FPGA IP, ATX PLL, and IO PLL (sampling PLL).
2. Starts up the design example with an operating speed of 25G.
3. Waits for RX clock and PHY status signal to settle.
4. Prints PHY status.
5. Checks for 10 valid data on 25G speed.
6. Switches to 10G speed.
7. Waits for RX clock and PHY status signal to settle.
8. Prints PHY status.
9. Checks for another 10 valid data on 10G speed.
10. Switches to 25G speed once all 10 valid data passes.
11. Waits for RX clock and PHY status signal to settle.
12. Prints PHY status.
13. Checks for another 10 valid data on 25G speed.
14. Analyzes the results. The successful testbench displays "`Simulation PASSED.`" when the PTP delay and offset data is within the threshold value.

The following sample output illustrates a successful simulation test run:

```
# Running at 25G mode...
#
#
# Waiting for RX alignment...
# iatpg_pipeline_global_en is set
# iatpg_pipeline_global_en is set
# RX deskew locked.
# RX lane aligmnent locked.
#
# Sending packets...
#
# Delay (sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000064457
# Offset(sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000000000
# Offset within tolerence range.
#
#
# Delay (sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000064bb4
# Offset(sec[95:48],ns[47:16],fns[15:0]): 0x0000000000ffffffff8a2
# Offset within tolerence range.
#
#
# Delay (sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000643b5
# Offset(sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000000520
# Offset within tolerence range.
#
#
# Delay (sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000634fb
# Offset(sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000000000
# Offset within tolerence range.
#
#
# Delay (sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000063f3b
```

```
# Offset(sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000000000000
# Offset within tolerence range.
#
#
# Delay (sec[95:48],ns[47:16],fns[15:0]): 0x0000000000000000063a1a
# Offset(sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000000000000
# Offset within tolerence range.
#
#
# Delay (sec[95:48],ns[47:16],fns[15:0]): 0x0000000000000000006445a
# Offset(sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000000000000
# Offset within tolerence range.
#
#
# Delay (sec[95:48],ns[47:16],fns[15:0]): 0x0000000000000000063e95
# Offset(sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000000000000
# Offset within tolerence range.
#
#
# Delay (sec[95:48],ns[47:16],fns[15:0]): 0x0000000000000000000648d5
# Offset(sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000000000000
# Offset within tolerence range.
#
#
# Delay (sec[95:48],ns[47:16],fns[15:0]): 0x0000000000000000000643b5
# Offset(sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000000000520
# Offset within tolerence range.
#
#
#
# Finished sending packets.
#
#
#
# Switching to 10G mode: 10G Reconfig starts...
# Switching to 10G mode: 10G Reconfig End.
#
# Waiting for RX alignment...
# RX deskew locked.
# RX lane aligmnent locked.
#
# Configuring 1588 period...
# Configuring 1588 period done.
#
# Sending packets...
#
# Delay (sec[95:48],ns[47:16],fns[15:0]): 0x0000000000000000000e5a7d
# Offset(sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000000002013
# Offset within tolerence range.
#
#
# Delay (sec[95:48],ns[47:16],fns[15:0]): 0x0000000000000000000e0764
# Offset(sec[95:48],ns[47:16],fns[15:0]): 0x0000000000000fffffffff99a0
# Offset within tolerence range.
#
#
# Delay (sec[95:48],ns[47:16],fns[15:0]): 0x0000000000000000000e0764
# Offset(sec[95:48],ns[47:16],fns[15:0]): 0x0000000000000fffffffffa66d
# Offset within tolerence range.
#
#
# Delay (sec[95:48],ns[47:16],fns[15:0]): 0x0000000000000000000dfa97
# Offset(sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000000006660
# Offset within tolerence range.
#
#
# Delay (sec[95:48],ns[47:16],fns[15:0]): 0x0000000000000000000e1431
# Offset(sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000000005993
# Offset within tolerence range.
#
#
```

```
# Delay (sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000e2db7
# Offset(sec[95:48],ns[47:16],fns[15:0]): 0x0000000000fffffffccc0
# Offset within tolerence range.
#
#
# Delay (sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000e1431
# Offset(sec[95:48],ns[47:16],fns[15:0]): 0x0000000000fffffff8006
# Offset within tolerence range.
#
#
# Delay (sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000e60e4
# Offset(sec[95:48],ns[47:16],fns[15:0]): 0x0000000000fffffffff320
# Offset within tolerence range.
#
#
# Delay (sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000dfa97
# Offset(sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000005993
# Offset within tolerence range.
#
#
# Delay (sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000e874b
# Offset(sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000000ce0
# Offset within tolerence range.
#
#
#
# Finished sending packets.
#
#
#
# Switching to 25G mode: 25G Reconfig start...
# Switching to 25G mode: 25G Reconfig end.
#
# Waiting for RX alignment...
# RX deskew locked.
# RX lane aligmnent locked.
#
# Configuring 1588 period...
# Configuring 1588 period done.
#
# Sending packets...
#
# Delay (sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000063c58
# Offset(sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000000000
# Offset within tolerence range.
#
#
# Delay (sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000063c58
# Offset(sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000000000
# Offset within tolerence range.
#
#
# Delay (sec[95:48],ns[47:16],fns[15:0]): 0x0000000000000000006502f
# Offset(sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000000000
# Offset within tolerence range.
#
#
# Delay (sec[95:48],ns[47:16],fns[15:0]): 0x0000000000000000006502f
# Offset(sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000000000
# Offset within tolerence range.
#
#
# Delay (sec[95:48],ns[47:16],fns[15:0]): 0x0000000000000000006554d
# Offset(sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000000000
# Offset within tolerence range.
#
#
# Delay (sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000064b10
# Offset(sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000000000
# Offset within tolerence range.
#
```

```
#
# Delay (sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000064b10
# Offset(sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000000000
# Offset within tolerence range.
#
#
# Delay (sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000064bb4
# Offset(sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000000000
# Offset within tolerence range.
#
#
# Delay (sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000064bb4
# Offset(sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000000000
# Offset within tolerence range.
#
#
# Delay (sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000065a6c
# Offset(sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000000000
# Offset within tolerence range.
#
#
#
# Finished sending packets.
#
# **
# ** Testbench complete.
# **
```

2.5. Compilation

Follow the procedure in Compiling and Configuring the Design Example in Hardware on page 8 to compile and configure the design example in the selected hardware.

You can estimate resource utilization and Fmax using the compilation-only design example. You can compile your design using the **Start Compilation** command on the **Processing** menu in the Intel Quartus Prime Pro Edition software. A successful compilation generates the compilation report summary.

For more information, refer to *Design Compilation* in the *Compiler User Guide: Intel Quartus Prime Pro Edition* document.

Related Information

Design Compilation
In Compiler User Guide: Intel Quartus Prime Pro Edition

2.6. Hardware Testing

In the hardware design example, you can program the IP core in internal serial loopback mode and generate traffic on the transmit side that loops back through the receive side.

Follow the procedure at the provided related information link to test the design example in the selected hardware.

Related Information

Testing the 25G Ethernet Intel FPGA IP Design in Hardware on page 9
More information on the procedure and hardware setup.

2.6.1. Test Procedure—Design Example Without the IEEE 1588v2 Feature

Follow these steps to test the design examples in hardware using PMA serial loopback:

Note: The design example starts with default data rate of 25G.

1. Perform data rate switching to 10G:
   a. In Intel Quartus Prime Pro Edition software, go to **Tools ➤ In-System Sources & Probes Editor** tool to open the default source and probe GUI.
   b. Set the source bit[1] in source and probe to 1.
2. Perform data rate switching to 25G:
   a. In Intel Quartus Prime Pro Edition software, go to **Tools ➤ In-System Sources & Probes Editor** tool to open the default source and probe GUI.
   b. Set the source bit[1] in source and probe to 0.
3. Perform system reset release after executing the data rate reconfiguration:
   a. Click **Tools ➤ In-System Sources & Probes Editor** tool for the default Source and Probe GUI.
   b. Toggle the system reset signal (`Source[0]`) from 0 to 1 to apply the reset and return the system reset signal back to 0 to release the system from the reset state.
   c. Monitor the Probe signals and ensure that the status is valid.
4. Run the following command parameters in the system console to start the test.

Table 8. Command Parameters

Parameter	Description
`chkphy_status`	Displays the clock frequencies and PHY lock status.
`chkmac_stats`	Displays the values in the MAC statistics counters.
`clear_all_stats`	Clears the IP core statistics counters.
`start_gen`	Starts the packet generator.
`stop_gen`	Stops the packet generator.
`loop_on`	Turns on internal serial loopback.
`loop_off`	Turns off internal serial loopback.
`reg_read <addr>`	Returns the IP core register value at <*addr*>.
`reg_write <addr> <data>`	Writes <*data*> to the IP core register at address <*addr*>.

Note: The above configuration is applied to the default 25G mode for the first time.

When the test is completed, observe the output displayed in the System Console.

Related Information

Design Example Registers on page 45

2.6.2. Test Procedure—Design Example with the IEEE 1588v2 Feature

Follow these steps to test the design examples in hardware using PMA serial loopback:

Note: The design example starts with default data rate of 25G.

1. Perform data rate switching to 10G:
   a. In Intel Quartus Prime Pro Edition software, go to **Tools ➤ In-System Sources & Probes Editor** tool to open the default source and probe GUI.
   b. Set the source bit[1] in source and probe to 1.
   c. In the System Console panel, type the following commands as below to set the correct clock period for the required TX and RX MAC clock frequency in 10G speed mode:

```
reg_write 0xA05 0x66666
reg_write 0xB05 0x66666
```

2. Perform data rate switching to 25G:
   a. In Intel Quartus Prime Pro Edition software, go to **Tools ➤ In-System Sources & Probes Editor** tool to open the default source and probe GUI.
   b. Set the source bit[1] in source and probe to 0.
   c. In the System Console panel, type the following commands as below to set the correct clock period for the required TX and RX MAC clock frequency in 25G speed mode:

```
reg_write 0xA05 0x28F5C
reg_write 0xB05 0x28F5C
```

   Note: 0xA05 is register that configure TX_PTP_CLK_PERIOD. 0xB05 is register that configure RX_PTP_CLK_PERIOD.
3. Perform system reset release after executing the data rate reconfiguration:
   a. Click **Tools ➤ In-System Sources & Probes Editor** tool for the default Source and Probe GUI.
   b. Toggle the system reset signal (`Source[0]`) from 0 to 1 to apply the reset and return the system reset signal back to 0 to release the system from the reset state.
   c. Monitor the Probe signals and ensure that the status is valid.
4. Run the following command parameters in the system console to start the test.

Table 9. Command Parameters

Parameter	Description
`chkphy_status`	Displays the clock frequencies and PHY lock status.
`chkmac_stats`	Displays the values in the MAC statistics counters.
`clear_all_stats`	Clears the IP core statistics counters.
`start_gen`	Starts the packet generator.
`stop_gen`	Stops the packet generator.
`loop_on`	Turns on internal serial loopback.

continued...

Parameter	Description
`loop_off`	Turns off internal serial loopback.
`reg_read <addr>`	Returns the IP core register value at <*addr*>.
`reg_write <addr> <data>`	Writes <*data*> to the IP core register at address <*addr*>.

Note: The above configuration is applied to the default 25G mode for the first time.

When the test is completed, observe the output displayed in the System Console.

Related Information

Design Example Registers on page 45

3. 25G Ethernet Single-Channel Design Example for Intel Stratix 10 Devices

The 25G Ethernet single-channel design example demonstrates an Ethernet solution for Intel Stratix 10 devices using the 25G Ethernet Intel FPGA IP core.

Generate the design example from the **Example Design** tab of the 25G Ethernet Intel FPGA IP parameter editor. You can choose to generate the design with or without the IEEE 1588v2 feature. You can also choose to generate the design with or without the Reed-Solomon Forward Error Correction (RS-FEC) feature.

3.1. Features

- Supports single Ethernet channel operating at 25G.
- Generates design example with IEEE 1588v2 feature.
- Generates design example with RS-FEC feature.
- Generates design example separately from Intel Stratix 10 Transceiver Native PHY.
- Provides testbench and simulation script.

3.2. Hardware and Software Requirements

Intel uses the following hardware and software to test the design example in a Linux system:

- Intel Quartus Prime Pro Edition software
- ModelSim-SE, NCSim (Verilog only), VCS, and Xcelium simulator
- Intel Stratix 10 L-Tile GX Transceiver Signal Integrity Development Kit (1SX280LU2F50E2VG) for hardware testing

3.3. Functional Description

The 25G Ethernet single-channel design example consists of two core variants—MAC +PCS+PMA and MAC+PCS. The following block diagrams show the design components and the top-level signals of the two core variants in the 25G Ethernet single-channel design example.

Figure 10. Block Diagram—25G Ethernet Single-Channel Design Example (MAC+PCS +PMA Core Variant) Without the IEEE 1588v2 Feature

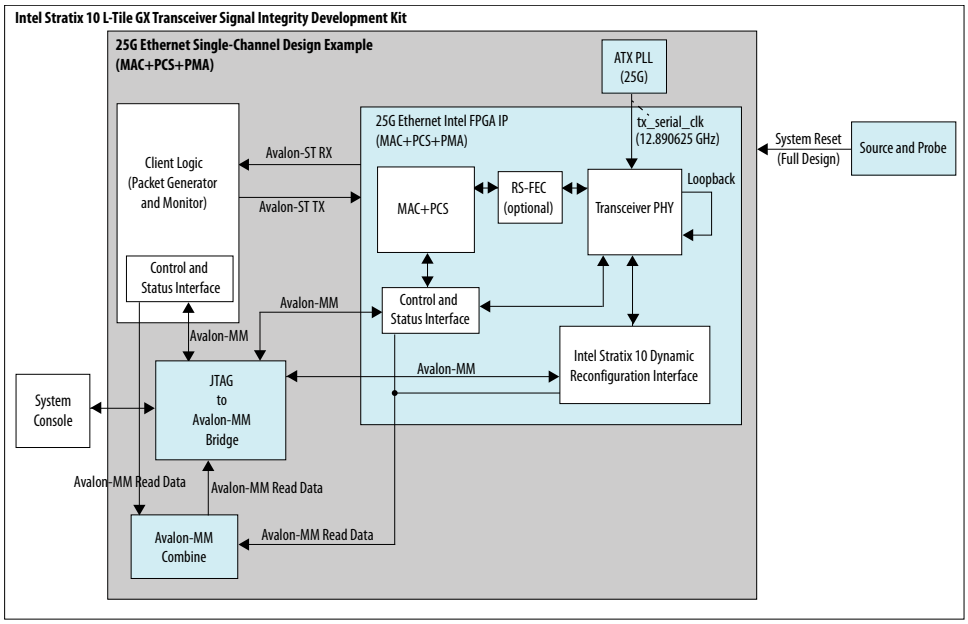


Figure 11. Block Diagram—25G Ethernet Single-Channel Design Example (MAC+PCS +PMA Core Variant) with the IEEE 1588v2 Feature

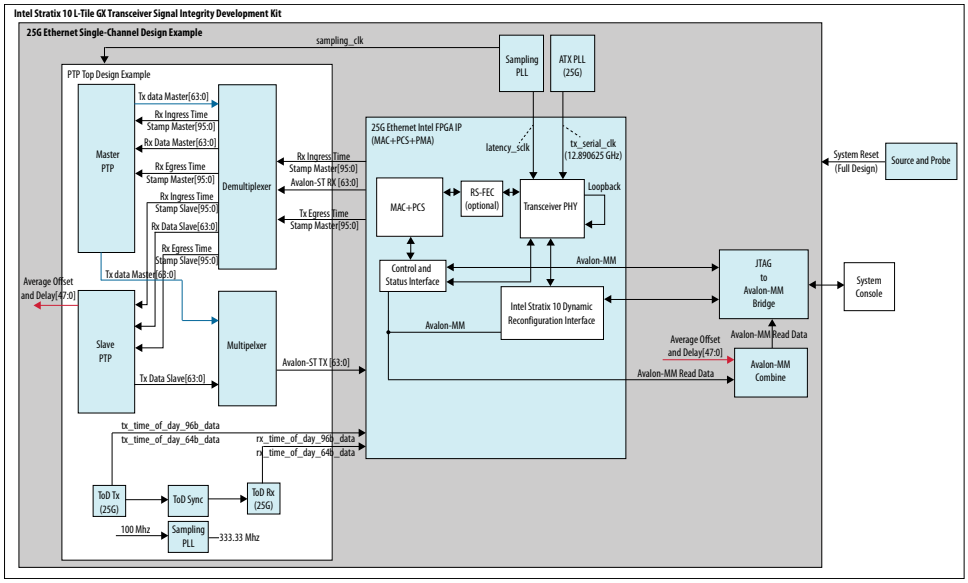
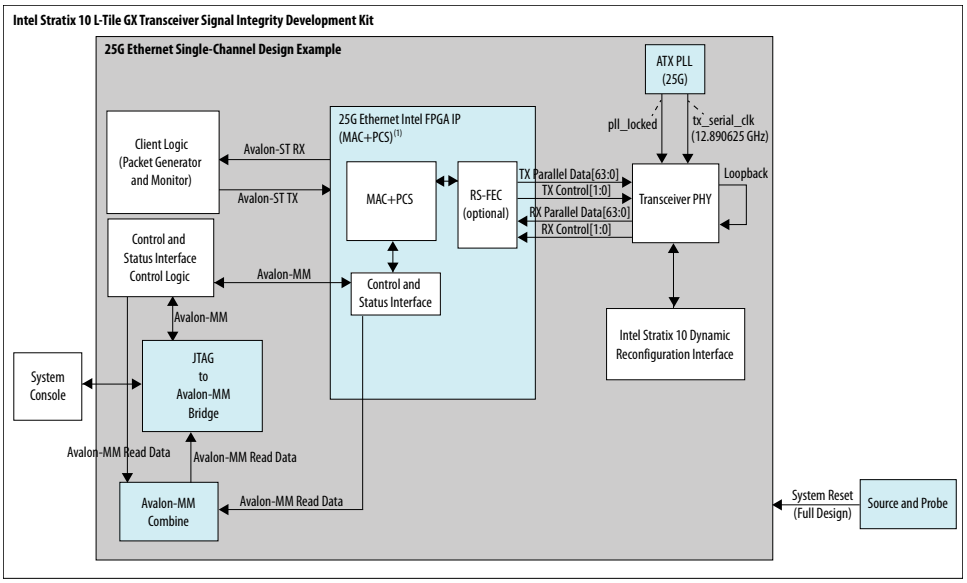
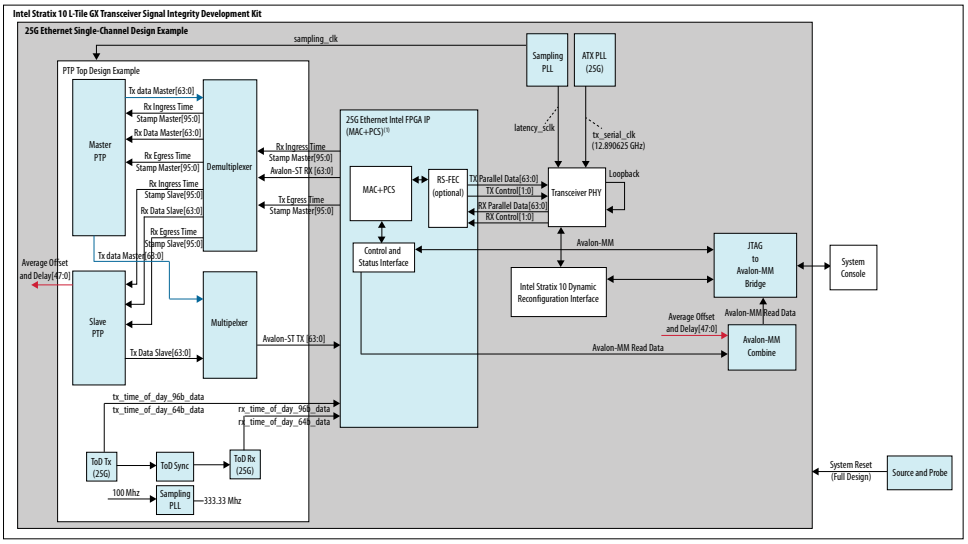


Figure 12. Block Diagram—25G Ethernet Single-Channel Design Example (MAC+PCS Core Variant) Without the IEEE 1588v2 Feature

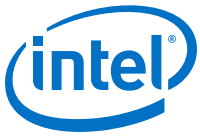


Note:
1. Components outside of this block are part of the design example only.

Figure 13. Block Diagram—25G Ethernet Single-Channel Design Example (MAC+PCS Core Variant) with the IEEE 1588v2 Feature



Note:
1. Components outside of this block are part of the design example only.



3.3.1. Design Components

Table 10. Design Components

Component	Description
25G Ethernet Intel FPGA IP	Consists of MAC, PCS, and Transceiver PHY, with the following configuration: • **Core Variant**: MAC+PCS+PMA • **Enable flow control**: Optional • **Enable link fault generation**: Optional • **Enable preamble passthrough**: Optional • **Enable statistics collection**: Optional • **Enable MAC statistics counters**: Optional • **Enable 10G/25G dynamic rate switching**: Not selected • **Enable Enable Altera Debug Master Endpoint (ADME)**: Optional • **Reference clock frequency**: 644.531250/322.265625 For the design example with the IEEE 1588 feature, the following additional parameters are configured: • **Enable IEEE 1588**: Selected • **Time of day format**: Enable 96-bit timestamp format (2) For the design example with the RS-FEC feature, the following additional parameter is configured: • **Enable RS-FEC**: Selected For the design example with separated transceiver native PHY, the following additional parameter is configured: • **Core Variant**: MAC+PCS
ATX PLL	Generates TX serial clocks for the 25G transceiver.
Client logic	Consists of: • Traffic generator, which generates burst packets to the 25G Ethernet Intel FPGA IP core for transmission. • Traffic monitor, which monitors burst packets that are coming from the 25G Ethernet Intel FPGA IP core.
Source and Probe	Source and probe signals, including system reset input signal, which you can use for debugging.
Design Components for the IEEE 1588v2 Feature	
Sampling PLL	Generates the clocks for the IEEE 1588v2 design components. • `latency_sclk`: 156.25 MHz for latency measurement. • `sampling_clk`: 250 MHz for ToD synchronization
Time-of-day (ToD) Sync	Synchronizes the 25G ToD.
ToD Tx	ToD for transmit paths for the 25G transceiver.
ToD Rx	ToD for receive paths for the 25G transceiver.
Master Precision Time Protocol (PTP)	Master PTP consists of a packet generator and a packet receiver. • Packet generator: Obtains timestamp information from 25G Ethernet Intel FPGA IP core and generates Avalon-ST packets such as Sync packet and Delay Response packet. • Packet receiver: Obtains the delay request packet information from 25G Ethernet Intel FPGA IP core and produces timestamp values.
Slave PTP	Slave PTP consists of a packet generator, a packet receiver, and a packet compute.

continued...

(2) The 25G Ethernet single-channel design example with IEEE 1588v2 feature only supports 96-bit timestamp format.

Component	Description
	• Packet generator: Obtains timestamp information from 25G Ethernet Intel FPGA IP core and generates Avalon-ST packets such as Delay Request packet. • Packet receiver: Obtains the Sync and Delay Response packets information from 25G Ethernet Intel FPGA IP core and produces timestamp values. • Packet compute: Calculates and produces the delay and offsets value based on the timestamp values.

3.4. Simulation

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

Related Information

Simulating the 25G Ethernet Intel FPGA IP Design Example Testbench on page 7

3.4.1. Testbench

Figure 14. Block Diagram of the 25G Ethernet Single-Channel Design Example Simulation Testbench

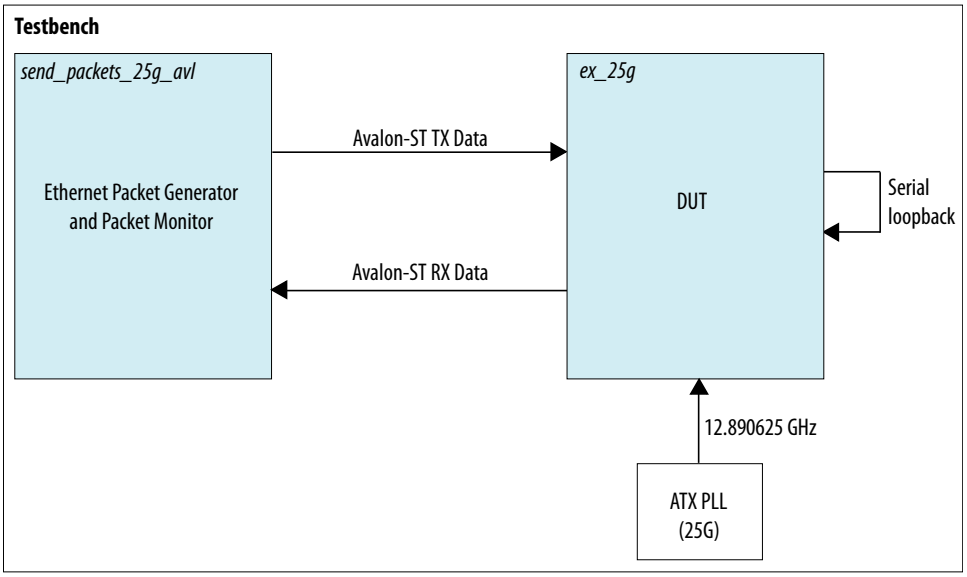


Table 11. Testbench Components

Component	Description
Device under test (DUT)	25G Ethernet Intel FPGA IP core.
Reconfiguration Sequencer	Reconfigures the transceiver channel speed from 10 Gbps to 25 Gbps, and vice versa.
Ethernet Packet Generator and Packet Monitor	Packet generator generates frames and transmit to the DUT. Packet Monitor monitors TX and RX datapaths and displays the frames in the simulator console.
ATX PLL	Generates a TX serial clock for the Intel Stratix 10 10G/25G transceiver which is wrapped in 25G Ethernet Intel FPGA IP core.

Note: For 25G Ethernet single-channel design example with IEEE 1588v2 feature simulation testbench, refer to Figure 11 on page 25.

3.4.2. Simulation Design Example Components

Table 12. 25G Ethernet Single-Channel Design Example Testbench File Descriptions

File Name	Description
Testbench and Simulation Files	
`basic_avl_tb_top.v`	Top-level testbench file. The testbench instantiates the DUT, performs Avalon-MM configuration on design components and client logic, and sends and receives packet to or from 25G Ethernet Intel FPGA IP.
Testbench Scripts	
`run_vsim.do`	The ModelSim script to run the testbench.
`run_vcs.sh`	The Synopsys VCS script to run the testbench.
`run_ncsim.sh`	The Cadence NCSim script to run the testbench.
`run_xcelium.sh`	The Xcelium script to run the testbench.

3.4.3. Test Case—Design Example Without the IEEE 1588v2 Feature

The simulation test case performs the following actions:

1. Instantiates 25G Ethernet Intel FPGA IP and ATX PLL.
2. Waits for RX clock and PHY status signal to settle.
3. Prints PHY status.
4. Analyzes the results. The successful testbench sends ten packets, receives ten packets, and displays "`Testbench complete.`"

Figure 15. Sample Simulation Output for Design Example Without the IEEE 1588v2 Feature

```
#rx_pcs_ready[ 0]
#RX deskew locked
#RX lane alignment locked
#TX enabled
#** Link           0  Sending Packet            1...
#** Link           0  Sending Packet            2...
#** Link           0  Sending Packet            3...
#** Link           0  Sending Packet            4...
#** Link           0  Sending Packet            5...
#** Link           0  Sending Packet            6...
#** Link           0  Sending Packet            7...
#** Link           0  Sending Packet            8...
#** Link           0  Received Packet           1...
#** Link           0  Received Packet           2...
#** Link           0  Sending Packet            9...
#** Link           0  Sending Packet           10...
#** Link           0  Received Packet           3...
#** Link           0  Received Packet           4...
#** Link           0  Received Packet           5...
#** Link           0  Received Packet           6...
#** Link           0  Received Packet           7...
#** Link           0  Received Packet           8...
#** Link           0  Received Packet           9...
#** Link           0  Received Packet          10...
#**
#** Testbench complete.
#**
```

3.4.4. Test Case—Design Example with the IEEE 1588v2 Feature

Note: For 25G Ethernet single-channel design example with IEEE 1588v2 feature simulation testbench, refer to Figure 11 on page 25.

The simulation test case performs the following actions:

1. Instantiates 25G Ethernet Intel FPGA IP, ATX PLL, and IO PLL (sampling PLL).
2. Waits for RX clock and PHY status signal to settle.
3. Prints PHY status.
4. Checks for 10 valid data.
5. Analyzes the results. The successful testbench displays "`Testbench complete.`" when the PTP delay and offset data are within the threshold values.

The following sample output illustrates a successful simulation test run:

```
#
# Waiting for RX alignment...
# iatpg_pipeline_global_en is set
# iatpg_pipeline_global_en is set
# RX deskew locked.
# RX lane aligmnent locked.
#
# Sending packets...
#
# Delay (sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000064457
# Offset(sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000000000
# Offset within tolerence range.
#
#
# Delay (sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000064bb4
# Offset(sec[95:48],ns[47:16],fns[15:0]): 0x0000000000fffffffff8a2
# Offset within tolerence range.
#
#
# Delay (sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000643b5
# Offset(sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000000520
# Offset within tolerence range.
#
#
# Delay (sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000634fb
# Offset(sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000000000
# Offset within tolerence range.
#
#
# Delay (sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000063f3b
# Offset(sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000000000
# Offset within tolerence range.
#
#
# Delay (sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000063a1a
# Offset(sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000000000
# Offset within tolerence range.
#
#
# Delay (sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000006445a
# Offset(sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000000000
# Offset within tolerence range.
#
#
# Delay (sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000063e95
# Offset(sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000000000
# Offset within tolerence range.
#
#
# Delay (sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000648d5
# Offset(sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000000000
# Offset within tolerence range.
#
#
# Delay (sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000643b5
# Offset(sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000000520
# Offset within tolerence range.
#
#
#
# Finished sending packets.
#
# **
# ** Testbench complete.
# **
```

Figure 16. Sample Simulation Output for Design Example with the IEEE 1588v2 Feature (Part 1 of 2)

```
#
# Waiting for RX alignment...
# iatpg_pipeline_global_en is set
# iatpg_pipeline_global_en is set
# RX deskew locked.
# RX lane aligmnent locked.
#
# Sending packets...
# Delay (sec[95:48],ns[47:16],fns[15:0]): 0x0000000000000000003c3b7
# Offset(sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000000000
# Offset within tolerence range.
#
#
# Delay (sec[95:48],ns[47:16],fns[15:0]): 0x0000000000000000003d2c2
# Offset(sec[95:48],ns[47:16],fns[15:0]): 0x000000000000fffffffffb34
# Offset within tolerence range.
#
#
# Delay (sec[95:48],ns[47:16],fns[15:0]): 0x0000000000000000003cdf7
# Offset(sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000000000
# Offset within tolerence range.
#
#
# Delay (sec[95:48],ns[47:16],fns[15:0]): 0x0000000000000000003cdf7
# Offset(sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000000000
# Offset within tolerence range.
#
#
# Delay (sec[95:48],ns[47:16],fns[15:0]): 0x0000000000000000003c363
# Offset(sec[95:48],ns[47:16],fns[15:0]): 0x000000000000000000009ec
# Offset within tolerence range.
```

Figure 17. Sample Simulation Output for Design Example with the IEEE 1588v2 Feature (Part 2 of 2)

```
#
#
# Delay (sec[95:48],ns[47:16],fns[15:0]): 0x0000000000000000003c883
# Offset(sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000000004cc
# Offset within tolerence range.
#
#
# Delay (sec[95:48],ns[47:16],fns[15:0]): 0x0000000000000000003c883
# Offset(sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000000004cc
# Offset within tolerence range.
#
#
# Delay (sec[95:48],ns[47:16],fns[15:0]): 0x0000000000000000003cd50
# Offset(sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000000000000
# Offset within tolerence range.
#
#
# Delay (sec[95:48],ns[47:16],fns[15:0]): 0x0000000000000000003c883
# Offset(sec[95:48],ns[47:16],fns[15:0]): 0x000000000000fffffffffb33
# Offset within tolerence range.
#
#
# Delay (sec[95:48],ns[47:16],fns[15:0]): 0x0000000000000000003d2c3
# Offset(sec[95:48],ns[47:16],fns[15:0]): 0x00000000000000000000004cc
# Offset within tolerence range.
#
#
#
# Finished sending packets.
#
# **
# ** Testbench complete.
# **
# ****************************************
# ** Note: $finish    : ./basic_avl_tb_top.sv(129)
#    Time: 150355 ns  Iteration: 2  Instance: /basic_avl_tb_top
```

3.5. Compilation

Follow the procedure in Compiling and Configuring the Design Example in Hardware on page 8 to compile and configure the design example in the selected hardware.

You can estimate resource utilization and Fmax using the compilation-only design example. You can compile your design using the **Start Compilation** command on the **Processing** menu in the Intel Quartus Prime Pro Edition software. A successful compilation generates the compilation report summary.

For more information, refer to *Design Compilation* in the *Compiler User Guide: Intel Quartus Prime Pro Edition* document.

Related Information

Design Compilation
In Compiler User Guide: Intel Quartus Prime Pro Edition

3.6. Hardware Testing

In the hardware design example, you can program the IP core in internal serial loopback mode and generate traffic on the transmit side that loops back through the receive side.

Follow the procedure at the provided related information link to test the design example in the selected hardware.

Related Information

Testing the 25G Ethernet Intel FPGA IP Design in Hardware on page 9
More information on the procedure and hardware setup.

3.6.1. Test Procedure—Design Example With and Without the IEEE 1588v2 Feature

Follow these steps to test the design example in hardware:

1. Before you run the hardware testing for this design example, you must reset the system:
   a. Click **Tools ➤ In-System Sources & Probes Editor** tool for the default Source and Probe GUI.
   b. Toggle the system reset signal (`Source[0]`) from 0 to 1 to apply the reset and return the system reset signal back to 0 to release the system from the reset state.
   c. Monitor the Probe signals and ensure that the status is valid.
2. Run the following command parameters in the system console to start the test.

Table 13. Command Parameters

Parameter	Description
`chkphy_status`	Displays the clock frequencies and PHY lock status.
`chkmac_stats`	Displays the values in the MAC statistics counters.
`clear_all_stats`	Clears the IP core statistics counters.
`start_gen`	Starts the packet generator.
`stop_gen`	Stops the packet generator.
`loop_on`	Turns on internal serial loopback.
`loop_off`	Turns off internal serial loopback.
`reg_read <addr>`	Returns the IP core register value at <*addr*>.
`reg_write <addr> <data>`	Writes <*data*> to the IP core register at address <*addr*>.

Related Information

Design Example Registers on page 45

4. 25G Ethernet Multi-Channel Design Example for Intel Stratix 10 Devices

The 25G Ethernet multi-channel design example demonstrates an Ethernet solution for Intel Stratix 10 devices using the 25G Ethernet Intel FPGA IP core.

Generate the design example from the **Example Design** tab of the 25G Ethernet Intel FPGA IP parameter editor.

4.1. Features

- Supports up to four Ethernet channels operating at 25G.
- Provides testbench and simulation script.

4.2. Hardware and Software Requirements

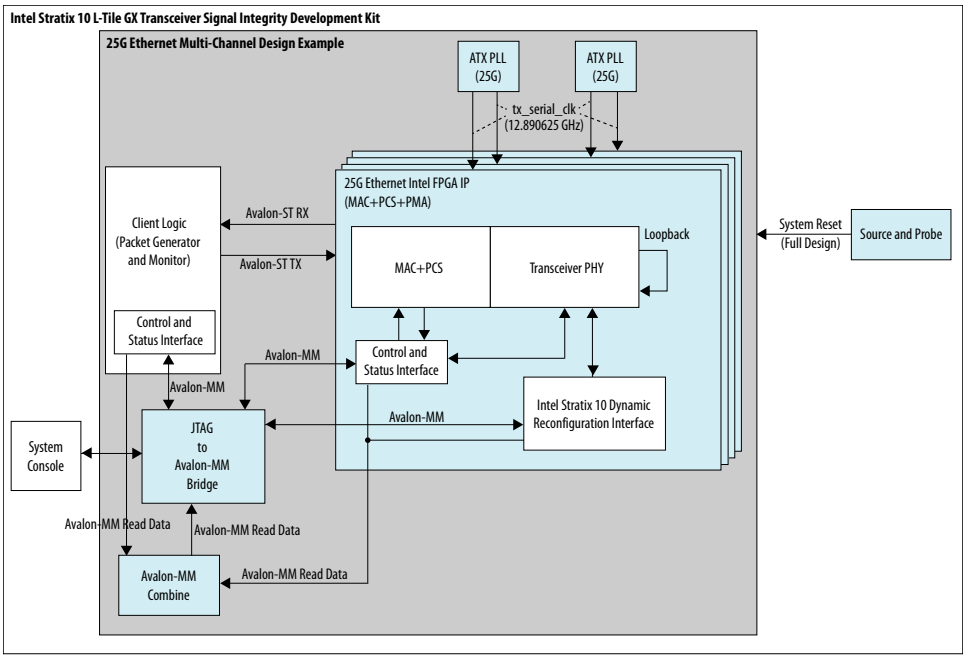
Intel uses the following hardware and software to test the design example in a Linux system:

- Intel Quartus Prime Pro Edition software
- ModelSim-SE, NCSim (Verilog only), VCS, and Xcelium simulator
- Intel Stratix 10 L-Tile GX Transceiver Signal Integrity Development Kit (1SX280LU2F50E2VG) for hardware testing

4.3. Functional Description

The 25G Ethernet multi-channel design example consists of various components. The following block diagram shows the design components and the top-level signals of the design example.

Figure 18. Block Diagram—25G Ethernet Multi-Channel Design Example (MAC+PCS+PMA Core Variant)



4.3.1. Design Components

Table 14. Design Components

Component	Description
25G Ethernet Intel FPGA IP	Consists of MAC, PCS, and Transceiver PHY, with the following configuration: • **Core Variant**: MAC+PCS+PMA • **Enable RS-FEC**: Not selected • **Enable flow control**: Optional • **Enable link fault generation**: Optional • **Enable preamble passthrough**: Optional • **Enable statistics collection**: Optional • **Enable MAC statistics counters**: Optional • **Enable IEEE 1588**: Not selected • **Enable 10G/25G dynamic rate switching**: Not selected • **Enable Enable Altera Debug Master Endpoint (ADME)**: Optional • **Reference clock frequency**: 644.531250/322.265625
ATX PLL	Generates TX serial clocks for the 25G transceiver.
Client logic	Consists of: • Traffic generator, which generates burst packets to the 25G Ethernet Intel FPGA IP core for transmission. • Traffic monitor, which receives burst packets from 25G Ethernet Intel FPGA IP core.
Source and Probe	Source and probe signals, including system reset input signal, which you can use for debugging.

4.4. Simulation

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

Related Information

Simulating the 25G Ethernet Intel FPGA IP Design Example Testbench on page 7

4.4.1. Testbench

Figure 19. Block Diagram of the 25G Ethernet Multi-Channel Design Example Simulation Testbench

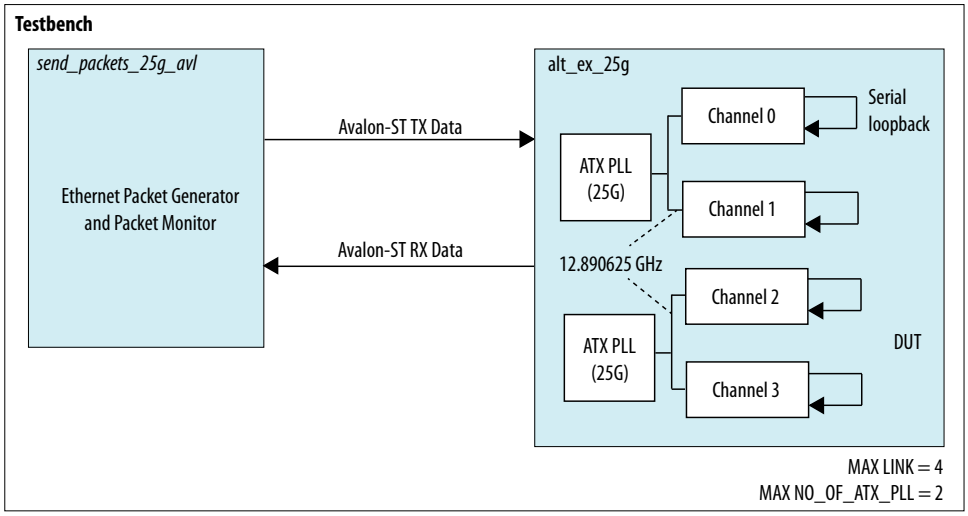


Table 15. Testbench Components

Component	Description
Device under test (DUT)	25G Ethernet Intel FPGA IP core.
Reconfiguration Sequencer	Reconfigures the transceiver channel speed from 10 Gbps to 25 Gbps, and vice versa.
Ethernet Packet Generator and Packet Monitor	Packet generator generates frames and transmit to the DUT. Packet Monitor monitors TX and RX datapaths and displays the frames in the simulator console.
ATX PLL	Generates a TX serial clock for the Intel Stratix 10 10G/25G transceiver which is wrapped in 25G Ethernet Intel FPGA IP core.

4.4.2. Simulation Design Example Components

Table 16. 25G Ethernet Multi-Channel Design Example Testbench File Descriptions

File Name	Description
Testbench and Simulation Files	
basic_avl_tb_top.v	Top-level testbench file. The testbench instantiates the DUT, performs Avalon-MM configuration on design components and client logic, and sends and receives packet to or from 25G Ethernet Intel FPGA IP.
Testbench Scripts	
run_vsim.do	The ModelSim script to run the testbench.
run_vcs.sh	The Synopsys VCS script to run the testbench.
run_ncsim.sh	The Cadence NCSim script to run the testbench.
run_xcelium.sh	The Xcelium script to run the testbench.

4.4.3. Test Case

The simulation test case performs the following steps:

1. Instantiates 25G Ethernet Intel FPGA IP and ATX PLL.
2. Waits for PHY status signal to settle.
3. Prints PHY status.
4. Analyzes the results. The successful testbench sends and receives packets, and displays "Testbench complete."

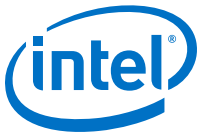


Figure 20. Sample Simulation Output when Ethernet Channel is Configured to 1

This figure shows a successful simulation test run when the Ethernet channel (i.e., LINK) is configured to 1.

```
#rx_pcs_ready[ 0]
#RX deskew locked
#RX lane alignment locked
#TX enabled
#** Link           0  Sending Packet           1...
#** Link           0  Sending Packet           2...
#** Link           0  Sending Packet           3...
#** Link           0  Sending Packet           4...
#** Link           0  Sending Packet           5...
#** Link           0  Sending Packet           6...
#** Link           0  Sending Packet           7...
#** Link           0  Sending Packet           8...
#** Link           0  Received Packet          1...
#** Link           0  Received Packet          2...
#** Link           0  Sending Packet           9...
#** Link           0  Sending Packet          10...
#** Link           0  Received Packet          3...
#** Link           0  Received Packet          4...
#** Link           0  Received Packet          5...
#** Link           0  Received Packet          6...
#** Link           0  Received Packet          7...
#** Link           0  Received Packet          8...
#** Link           0  Received Packet          9...
#** Link           0  Received Packet         10...
#**
#** Testbench complete.
#**
```

Figure 21. Sample Simulation Output when Ethernet Channel is Configured to 4 (Part 1 of 2)

This figure shows a successful simulation test run when the Ethernet channel (i.e., LINK) is configured to 4.

```
rx_pcs_ready[          0]
rx_pcs_ready[          1]
rx_pcs_ready[          2]
rx_pcs_ready[          3]
RX deskew locked
RX lane aligmnent locked
TX enabled
** Link           0  Sending Packet           1...
** Link           0  Sending Packet           2...
** Link           0  Sending Packet           3...
** Link           0  Sending Packet           4...
** Link           0  Sending Packet           5...
** Link           0  Sending Packet           6...
** Link           0  Sending Packet           7...
** Link           0  Sending Packet           8...
** Link           0  Received Packet          1...
** Link           0  Received Packet          2...
** Link           0  Sending Packet           9...
** Link           0  Sending Packet          10...
** Link           0  Received Packet          3...
** Link           0  Received Packet          4...
** Link           0  Received Packet          5...
** Link           0  Received Packet          6...
** Link           0  Received Packet          7...
** Link           0  Received Packet          8...
** Link           0  Received Packet          9...
** Link           0  Received Packet         10...
** Link           1  Sending Packet           1...
** Link           1  Sending Packet           2...
** Link           1  Sending Packet           3...
** Link           1  Sending Packet           4...
** Link           1  Sending Packet           5...
** Link           1  Sending Packet           6...
** Link           1  Sending Packet           7...
** Link           1  Sending Packet           8...
** Link           1  Received Packet          1...
** Link           1  Received Packet          2...
** Link           1  Received Packet          9...
** Link           1  Received Packet         10...
** Link           1  Received Packet          3...
** Link           1  Received Packet          4...
** Link           1  Received Packet          5...
** Link           1  Received Packet          6...
** Link           1  Received Packet          7...
** Link           1  Received Packet          8...
** Link           1  Received Packet          9...
** Link           1  Received Packet         10...
```

Figure 22. Sample Simulation Output when Ethernet Channel is Configured to 4 (Part 2 of 2)

This figure shows a successful simulation test run when the Ethernet channel (i.e., LINK) is configured to 4.

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

4.5. Compilation

Follow the procedure in Compiling and Configuring the Design Example in Hardware on page 8 to compile and configure the design example in the selected hardware.

You can estimate resource utilization and Fmax using the compilation-only design example. You can compile your design using the **Start Compilation** command on the **Processing** menu in the Intel Quartus Prime Pro Edition software. A successful compilation generates the compilation report summary.

For more information, refer to *Design Compilation* in the *Compiler User Guide: Intel Quartus Prime Pro Edition* document.

Related Information

Design Compilation
In Compiler User Guide: Intel Quartus Prime Pro Edition

4.6. Hardware Testing

In the hardware design example, you can program the IP core in internal serial loopback mode and generate traffic on the transmit side that loops back through the receive side.

Follow the procedure at the provided related information link to test the design example in the selected hardware.

Related Information

Testing the 25G Ethernet Intel FPGA IP Design in Hardware on page 9
More information on the procedure and hardware setup.

4.6.1. Test Procedure

Follow these steps to test the design example in hardware:

1. Before you run the hardware testing for this design example, you must reset the system:
   a. Click **Tools ➤ In-System Sources & Probes Editor** tool for the default Source and Probe GUI.
   b. Toggle the system reset signal (`Source[0]`) from 0 to 1 to apply the reset and return the system reset signal back to 0 to release the system from the reset state.
   c. Monitor the Probe signals and ensure that the status is valid.
2. Run the following command parameters in the system console to start the test.

Table 17. Command Parameters

Parameter	Description
`chkphy_status <link_num>`	Displays the clock frequencies and PHY lock status.
`chkmac_stats <link_num>`	Displays the values in the MAC statistics counters.
`clear_all_stats <link_num>`	Clears the IP core statistics counters.
`start_gen <link_num>`	Starts the packet generator.
`stop_gen <link_num>`	Stops the packet generator.
`loop_on <link_num>`	Turns on internal serial loopback.

continued...

Parameter	Description
`loop_off <link_num>`	Turns off internal serial loopback.
`reg_read <addr>`	Returns the IP core register value at <*addr*>.
`reg_write <addr> <data>`	Writes <*data*> to the IP core register at address <*addr*>.

Note: `link_num` is valid for 0 to 3 only.

Related Information

Design Example Registers on page 45

5. 25G Ethernet Intel FPGA IP Design Example References

This section provides information about the 25G Ethernet Intel FPGA IP core interface signals and registers in the design examples.

5.1. Design Example Interface Signals

The 25G Ethernet Intel FPGA IP core testbench is self-contained and does not require you to drive any input signals.

Table 18. Hardware Design Example Interface Signals for 25GbE Intel FPGA IP Core for Intel Stratix 10 Devices

Signal	Direction	Comments
clk100	Input	Drive at 100 MHz. The intent is to drive this from a 100 Mhz oscillator on theboard.
clk_ref	Input	Drive at 644.53125 MHz or 322.265625 MHz from an oscillator on the board.
cpu_resetn	Input	Resets the IP core. Active low. Drives the global hard reset csr_reset_n to the IP core.
tx_serial	Output	Transceiver PHY output serial data.
rx_serial	Input	Transceiver PHY input serial data.
user_led[7:0]	Output	Status signals. The hardware design example connects these bits to drive LEDs on the target board. Individual bits reflect the following signal values and clock behavior: • [0]: Main reset signal to IP core • [1]: Reserved • [2]: Divided version of clk50 • [3]: Divided version of 100 MHz status clock • [4]: tx_lanes_stable • [5]: rx_block_lock • [6]: rx_am_lock • [7]: rx_pcs_ready

Related Information

Interfaces and Signal Descriptions
Provides detailed descriptions of the Intel Stratix 10 25G Ethernet Intel FPGA IP core signals and the interfaces to which they belong.

5.2. Design Example Registers

Table 19. Hardware Design Example Register Map for 25G Ethernet Intel FPGA IP Core for Intel Stratix 10 Devices

You access these registers with the `reg_read` and `reg_write` functions in the System Console.

Word Offset	Register Category
Variant: Single-Channel	
0X0000–0X0DFF	Register range to access the Status Registers.
0X4000–0X7FFF	Register range to access the Reconfiguration Registers.
0X10000–0X10001	Register range to access the Reconfiguration Registers module for 10G/25G switching.
0x1020	32-bit `average_offset_fnsec_r` register: • Read this register to obtain average offset value [47:16] in fractional nanosecond, which is derived from the offset adjustment data [96:0] of the PTP slave.
0x1021	32-bit `average_offset_fnsec_to_mem` register: • Read this register to obtain average offset value [15:0] in fractional nanosecond, which is derived from the offset adjustment data [96:0] of the PTP slave. • Bit [31:16]: Reserved.
0x1030	32-bit `average_delay_fnsec_r` register: • Read this register to obtain average delay value [47:16] in fractional nanosecond, which is derived from the offset adjustment data [191:96] of the PTP slave.
0x1031	32-bit `average_delay_fnsec_to_mem` register: • Read this register to obtain average delay value [15:0] in fractional nanosecond, which is derived from the offset adjustment data [191:96] of the PTP slave. • Bit [31:16]: Reserved.
Variant: Multi-Channel	
0x00000–0x30DFF	For multi-channel design examples, the base address of all channels are incremented with 0x10000. This corresponds to: • Channel 0 Range: 0x00300–00DFF • Channel 1 Range: 0x10300–10DFF • Channel 2 Range: 0x20300–20DFF • Channel 3 Range: 0x30300–30DFF

Note: For C2E_REVB device prior to accessing the XCVR core reconfiguration register, disable the background calibration as described in the *Background Calibration* section of the *Intel Stratix 10 L- and H-Tile Transceiver PHY User Guide*.

Related Information

- 25G Ethernet Intel Stratix 10 FPGA IP User Guide
  Describes the 25G Ethernet Intel FPGA IP control, status, and statistics registers.
- Intel Stratix 10 L- and H-Tile Transceiver PHY User Guide

6. 25G Ethernet Intel Stratix 10 FPGA IP Design Example User Guide Archives

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

IP Core Version	User Guide
18.0	25G Ethernet Intel Stratix 10 FPGA IP Design Example User Guide

7. Document Revision History for the 25G Ethernet Intel Stratix 10 FPGA IP Design Example User Guide

Document Version	Intel Quartus Prime Version	Changes
2019.01.07	18.1	• Updated the *Generating the Design Example* topic to correct target development kit in Step 8 from **Stratix 10 GX FPGA Development Kit** to **Intel Stratix 10 L-Tile GX Transceiver Signal Integrity Development Kit**. • Updated Figure: *Block Diagram—10G/25G Ethernet Single-Channel Design Example (MAC+PCS+PMA Core Variant) with the IEEE 1588v2 Feature*. • Added a note to *Design Example Registers* topic.
2018.10.03	18.1	• Updated Table: *Parameters in the **Example Design** Tab* to update the description for **Select Board**. • Updated the *Hardware and Software Requirements* topics for all design example chapters. • Updated the *Design Components* topics for all design example chapters. • Added new Figures: — *Block Diagram—25G Ethernet Single-Channel Design Example (MAC +PCS Core Variant) Without the IEEE 1588v2 Feature* — *Block Diagram—25G Ethernet Single-Channel Design Example (MAC +PCS Core Variant) Without the IEEE 1588v2 Feature* — *Block Diagram—25G Ethernet Single-Channel Design Example (MAC +PCS Core Variant) Without the IEEE 1588v2 Feature* — *Block Diagram—25G Ethernet Single-Channel Design Example (MAC +PCS Core Variant) with the IEEE 1588v2 Feature* • Updated Figures: — *Example Design Tab in the 25G Ethernet Intel FPGA IP Parameter Editor* — *Block Diagram—25G Ethernet Single-Channel Design Example (MAC +PCS+PMA Core Variant) with the IEEE 1588v2 Feature* — *Block Diagram—25G Ethernet Single-Channel Design Example (MAC +PCS+PMA Core Variant) Without the IEEE 1588v2 Feature* — *Block Diagram—25G Ethernet Multi-Channel Design Example (MAC +PCS+PMA Core Variant)* — *Block Diagram—10G/25G Ethernet Single-Channel Design Example (MAC+PCS+PMA) Without the IEEE 1588v2 Feature* — *Block Diagram—10G/25G Ethernet Single-Channel Design Example (MAC+PCS+PMA) with the IEEE 1588v2 Feature* • Removed Figure: *Sample Simulation Output for Design Example with the IEEE 1588v2 Feature (Part 1 of 2)* and *Sample Simulation Output for Design Example with the IEEE 1588v2 Feature (Part 2 of 2)*. • Updated the simulation sample output of the *Test Case—Design Example with the IEEE 1588v2 Feature* topic for 25G Ethernet Single-Channel design example.

continued...

Document Version	Intel Quartus Prime Version	Changes
		• Updated the simulation sample output of the *Test Case—Design Example with the IEEE 1588v2 Feature* topic for the 10G/25G Ethernet design example. • Restructured descriptions for *Features* topics for all design example chapters. • Streamlined the contents and document organization.
2018.06.25	18.0	Initial release.