F-Tile Ethernet Intel® FPGA Hard IP Design Example User Guide

Updated for Intel® Quartus® Prime Design Suite: **21.3**

IP Version: **3.0.0**

Subscribe

Send Feedback

UG-20326 | 2021.10.11

Latest document on the web: **PDF** | **HTML**

Contents

1. Quick Start Guide

The F-Tile Ethernet Intel® FPGA Hard IP core provides a simulation testbench and a hardware design example that supports compilation and hardware testing. When you generate the design example, the parameter editor automatically creates an example design with all files necessary to compile and test the design in hardware. The simulation example design contains a simple testbench used to exercise the hardware example design.

You can generate a design example for any supported variants including design examples with Media Access Controller (MAC) interface, Physical Coding Sublayer (PCS) interface, Optical Transport Network (OTN) interface, and Flexible Ethernet (FlexE) interface for various Ethernet modes and optional FEC mode. In your design example, you can also enable the Precision Time Protocol (PTP) and auto-negotiation and link training options. For a list of supported configurations in the current Intel Quartus® Prime Pro Edition software version, refer to the Variant Selection table in the *F-Tile Ethernet Intel FPGA Hard IP User Guide*.

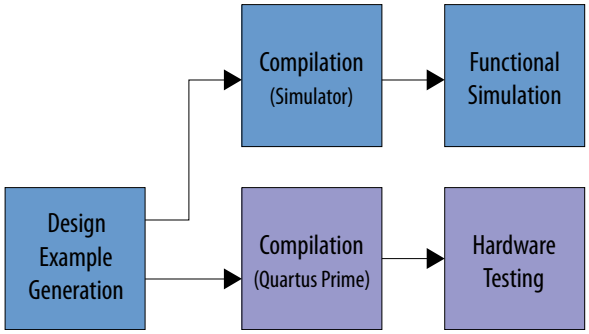
This user guide describes the following design examples:

- Design Example: Single IP Core Instantiation on page 14
- Design Example: Single IP Core Instantiation with Precision Time Protocol on page 23
- Design Example: Single IP Core Instantiation with Auto-Negotiation and Link Training on page 32
- Design Example: Multiple IP Core Instantiation on page 41

Note: The current Intel Quartus Prime Pro Edition software release does not support 40GE/50GE/100GE OTN design examples with no FEC.

Figure 1. Development Stages for the Design Example

Future IP core releases also provide a hardware design example you can compile and test in hardware.



Related Information

- F-Tile Ethernet Intel FPGA Hard IP User Guide

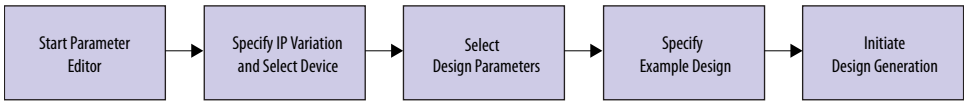
- F-Tile Ethernet Intel FPGA Hard IP Release Notes
- F-Tile Auto-Negotiation and Link Training for Ethernet Release Notes

1.1. Generating the Design

Intel Quartus Prime software supports both, single IP instance generation and multiple IP instances generation. Per your design needs, follow one of the following design generation flows.

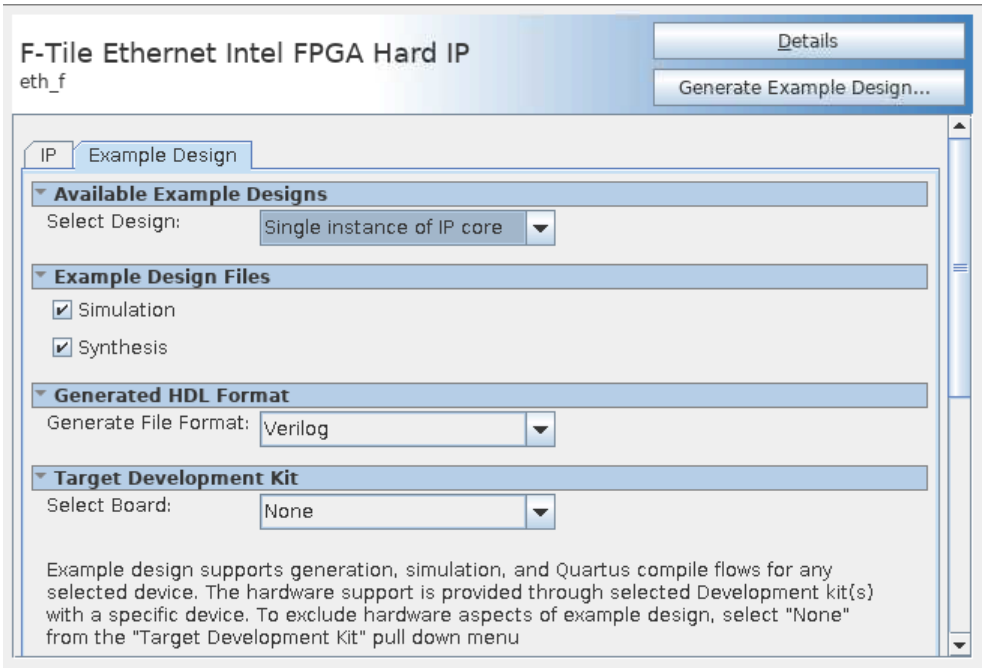
1.1.1. Generating Single IP Instance Design

Figure 2. Procedure



1. In the Intel Quartus Prime Pro Edition, click **File ➤ New Project Wizard** to create a new Intel Quartus Prime project, or **File ➤ Open Project** to open an existing Intel Quartus Prime project. The wizard prompts you to specify a device.
2. Specify the device family **Agilex (F-Series/I-Series)** and select device with F-tile for your design.
3. Select **Tools ➤ IP Catalog** to open the IP Catalog and select **F-Tile Ethernet Intel FPGA Hard IP**.
4. Specify a top-level name <*your_ip*> and the folder for your custom IP variation. The parameter editor saves the IP variation settings in a file named <*your_ip*>.ip.
5. Click **Create**. The IP parameter editor appears.

Figure 3. Example Design Tab



6. On the **IP** tab, specify the parameters for your IP core variation. For exact IP parameter setting, refer to the *Selected IP Parameter Settings* table in the desired *Design Example* chapter.
7. Specify the parameters in the **Example Design** tab.
   a. Under **Available Example Designs**, select **Single instance of IP core**.
   b. Under **Example Design Files**, select the **Simulation** option to generate the testbench and the compilation-only project. Select the **Synthesis** option to generate the hardware design example.
   c. Under **Generated HDL Format**, select **Verilog**.
   d. Under **Target Development Kit**, select **None**.
8. Click the **Generate Example Design** button.

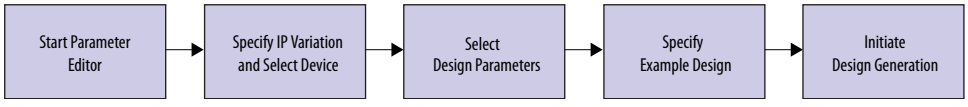
The software generates all design files in sub-directories. You require these files to run simulation, compilation, and hardware testing.

Related Information

F-Tile Ethernet Intel FPGA Hard IP User Guide: IP Parameters

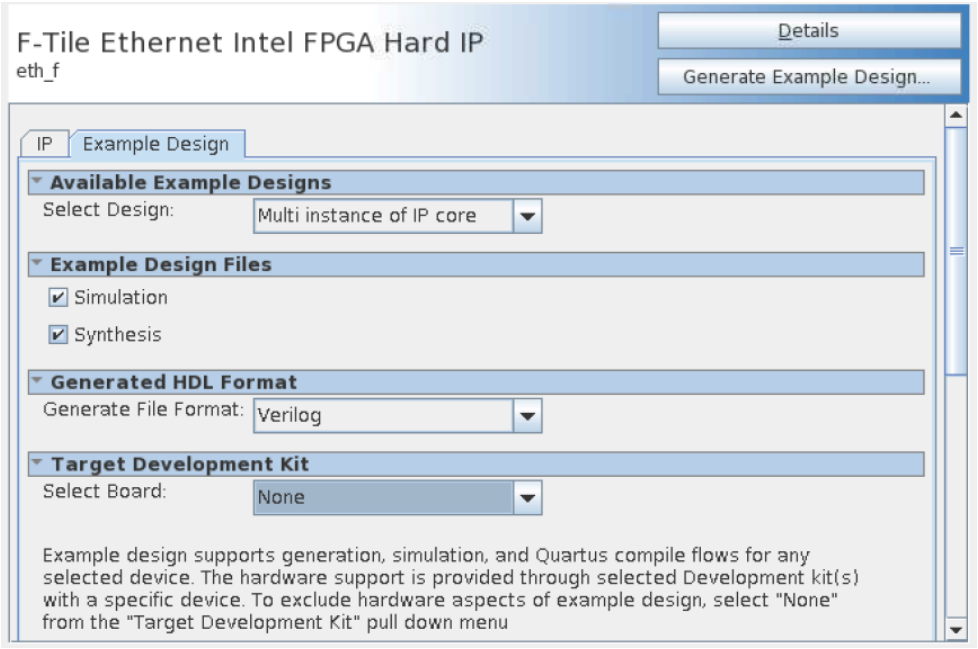
1.1.2. Generating Multiple IP Instance Design

Figure 4. Procedure



1. In the Intel Quartus Prime Pro Edition, click **File ➤ New Project Wizard** to create a new Intel Quartus Prime project, or **File ➤ Open Project** to open an existing Intel Quartus Prime project. The wizard prompts you to specify a device.
2. Specify the device family **Agilex (F-Series/I-Series)** and select device with F-tile for your design.
3. Select **Tools ➤ IP Catalog** to open the IP Catalog and select **F-Tile Ethernet Intel FPGA Hard IP**.
4. Specify a top-level name <*your_ip*> and the folder for your custom IP variation. The parameter editor saves the IP variation settings in a file named <*your_ip*>.ip.
5. Click **Create**. The IP parameter editor appears.

Figure 5. Example Design Tab



6. Specify the parameters in the **IP** tab. For exact IP parameter setting, refer to the *Selected IP Parameter Settings* table in the desired *Design Example* chapter.
7. Specify the parameters in the **Example Design** tab.
   a. Under **Available Example Designs**, select **Multi instance of IP core**.
   b. Under **Example Design Files**, select the **Simulation** option to generate the testbench and the compilation-only project. Select the **Synthesis** option to generate the hardware design example.
   c. Under **Generated HDL Format**, select **Verilog**.
   d. Under **Target Development Kit**, select **None**.
8. Click the **Generate Example Design** button.

The software generates all design files in sub-directories. You require these files to run simulation, compilation, and hardware testing.

Related Information

F-Tile Ethernet Intel FPGA Hard IP User Guide: IP Parameters

1.2. Directory Structure

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

Figure 6. Directory Structure for F-Tile Ethernet Intel FPGA Hard IP Design Example

The <ethernet_mode> refers to the selected Ethernet mode in the **IP** tab of the IP parameter editor.

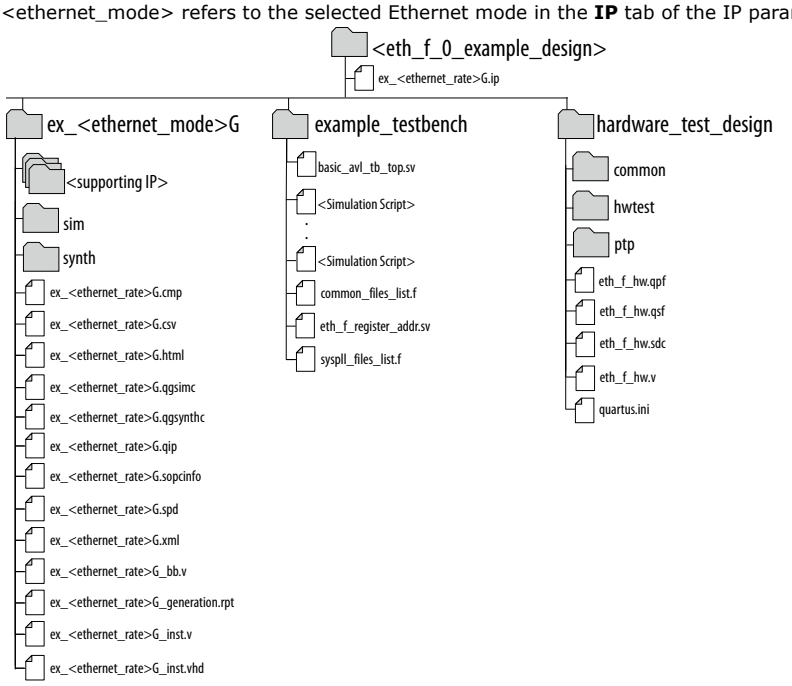


Table 1. Directory and File Description

Directory/File	Description
`<design_example_dir>/hardware_test_design/eth_f_hw.qpf`	Intel Quartus Prime project file.
`<design_example_dir>/hardware_test_design/eth_f_hw.qsf`	Intel Quartus Prime setting file.
`<design_example_dir>/hardware_test_design/eth_f_hw.v`	Design example top-level HDL.
`<design_example_dir>/hardware_test_design/eth_f_hw.sdc`	Synopsys Design Constraints (SDC) file.
`<design_example_dir>/hardware_test_design/common`	Hardware design example support files.
`<design_example_dir>/hardware_test_design/hwtest/main.tcl`	Main file for accessing System Console.

The Intel Quartus Prime software generates the design example files in the following folders:

- `<design_example_dir>/ex_<ethernet_rate>G`: IP core files
- `<design_example_dir>/example_testbench`: simulation files for testbench
- `<design_example_dir>/hardware_test_design`: hardware test design files

1.3. Generating Tile Files

The **Support-Logic Generation** is a pre-synthesis step used to generate tile-related files required for simulation and hardware design. The tile generation is required for all F-tile based design simulations. You must complete this step before simulation.

Using Graphical User Interface:

1. In the Intel Quartus Prime Pro Edition, navigate to the **Compilation Dashboard** window for your project overview.
2. Click **Support-Logic Generation**.

Using command prompt window:

1. At the command prompt, navigate to the `hardware_test_design` folder in your example design:

   ```
   cd <your_design_path>/hardware_test_design
   ```

2. If you enabled auto-negotiation and link training, you must specify the pin assignments. Append the `eth_f_hw.qsf` file with the recommended pin location assignments described in QSF Assignments on page 38.

   Note: This step is required only when you enabled the AN/LT feature in the F-Tile Ethernet Intel FPGA Hard IP and instantiated the F-Tile Ethernet Intel FPGA IP.

3. Run the following command:

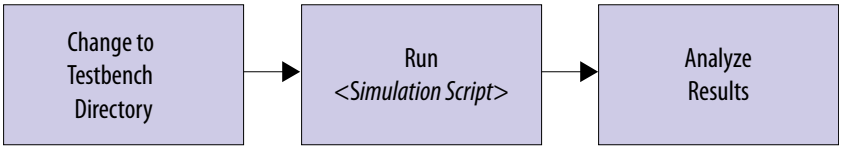
   ```
   quartus_tlg eth_f_hw
   ```

This step generates `eth_f_hw_tiles` files. The generated files are located in the `<your_design>/hardware_test_design` directory and contain the full netlist for simulation and synthesis.

1.4. Simulating the Design Example Testbench

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

Figure 7. Procedure



Note: Prior the simulation, you must generate tile-related files described in Generating Tile Files on page 8.

1. At the command prompt, change to the testbench simulation directory `cd <design_example_dir>/ex_*G/sim`.
2. Run the IP setup simulation:

   ```
   ip-setup-simulation -quartus-project=../../hardware_test_design/eth_f_hw.qpf
   ```

3. At the command prompt, change the working directory to `<design_example_dir>/example_testbench`.
4. 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*.
5. Analyze the results. The successful testbench displays `"Simulation Passed"`.

Table 2. Steps to Simulate the Testbench

Simulator	Instructions
Synopsys* VCS*	In the command line, type: `sh run_vcs.sh`
Synopsys VCS MX	In the command line, type: `sh run_vcsmx.sh` Use this script when the design contains Verilog HDL and System Verilog with VHDL.

ModelSim* SE or Questa* or Questa-Intel FPGA Edition	In the command line, type:
	`vsim -do run_vsim.do`
	If you prefer to simulate without bringing up the GUI, type:
	`vsim -c -do run_vsim.do`

A successful simulation ends with the following message:

```
Simulation Passed.
```

or

```
Testbench complete.
```

After successful completion, you can analyze the results.

1.4.1. Fast Sim Model for FGT Variants

To provide a reduction in a real-time simulation duration, you can utilize a Fast Sim model in your design example testbench. Available for the FGT variants only, the model is enabled by a macro in the simulation run script.

To enable the Fast Sim model, add the following macro to your simulation run script:

```
+define+IP7581SERDES_UX_SIMSPEED
```

The design example simulation script enables the macro by default for all variants with the exception of variants with enabled PTP or auto-negotiation and link training.

- In PTP variants, the macro is not enabled by default as it affects the timestamp accuracy in simulation. If you want to perform a general functionality check, you can enable the macro in your PTP simulation scripts.
- The macro is not available for designs with enabled auto-negotiation and link training.

You can also add the macro to your simulation script for your own testbench.

Attention: In 53G PAM4 Ethernet IPs, you must set the serial clock input to the IP to the exact value of 37.648 ps for simulation to work correctly. This limitation does not apply to the design example simulations since the serial lines are in a loopback.

1.5. Hardware and Software Requirements

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

- Intel Quartus Prime Pro Edition software
- System Console
- Supported Simulators:
  - Synopsys VCS
  - Synopsys VCS MX
  - Siemens* EDA ModelSim SE or Questa
  - Questa-Intel FPGA Edition
- Synopsys Verdi*: Optional waveform viewer used with the Synopsys VCS simulator.

1.6. Testing the Hardware Design Example

Follow these steps to test Ethernet-based design examples in hardware:

1. Generate design example as described in Generating the Design on page 4.
2. Modify the `.qsf` settings:
   - Set device to match the appropriate ordering part number (OPN) for your design.
   - Update the pinout to match the board and the design function.
   - Assign the appropriate VID settings in your `.qsf` file to match your board.
3. Generate the `.sof` file.
4. Update board clock settings. The default value for the PHY reference clock is 156.25 MHz. The default value for the reconfiguration clock is 100 MHz.
5. Insert appropriate electrical loopback plug into the Ethernet port.
6. Program the design.
7. Open **Tools ➤ System Debugging Tools ➤ System Console**.
   a. Navigate to the hardware directory `<design_example>/hardware_test_design/hwtest` directory.
   b. Type `source main_<variant_type>.tcl`.
   c. Type `set_jtag<number_of appropriate_JTAG_master>`

      Note: If you enabled auto-negotiation and link training in your design, follow the hardware design example steps described in Hardware Design Example on page 39.
   d. Run the following command:

```
run_test_without_loopback
```

The hardware design example uses `run_test` command to initiate packet transmission from packet generator to the IP core. The script enables internal loopback, checks the PMA PHY status, sends 16 packets, and displays the MAC statistics. When you use `run_test_without_loopback` command to run the hardware test, the script disables the internal loopback.

The following sample output illustrates a successful hardware test run:

```
% run_test_without_loopback
--- Turning off packet generation ----
--------------------------------------
--- Wait for RX clock to settle... ---
--------------------------------------
-------- Printing PHY status ---------
--------------------------------------
 RX PHY Register Access: Checking Clock Frequencies (KHz)
    TXCLK         :41505  (KHZ)
    RXCLK         :41503  (KHZ)

 TX PLL Lock Status           0x000000ff
 RX Frequency Lock Status     0x000000ff
 RX PCS Ready                 0x00000001
 TX Lanes Stable              0x00000001
 Deskew status                0x00000001
 Link Fault Status            0x00000000
 RX Frame Error               0x00000000
 RX AM LOCK Condition         0x00000001

---- Clearing MAC stats counters -----
---- Initialize PKT ROM Read address for IP_INST[0] ----
--------------------------------------
--------- Sending packets... ---------
--------------------------------------
----- Reading MAC stats counters -----
--------------------------------------

================================================================================
==========
                        STATISTICS FOR BASE 20480
(Rx)

================================================================================
==========
Fragmented Frames               : 0
Jabbered Frames                 : 0
Any Size with FCS Err Frame     : 0
Right Size with FCS Err Fra     : 0
Multicast data  Err Frames      : 0
Broadcast data Err  Frames      : 0
Unicast data Err  Frames        : 0
Multicast control  Err Frame    : 0
Broadcast control Err  Frame    : 0
Unicast control Err  Frames     : 0
Pause control Err  Frames       : 0
64 Byte Frames                  : 0
65 - 127 Byte Frames            : 16
128 - 255 Byte Frames           : 0
256 - 511 Byte Frames           : 0
512 - 1023 Byte Frames          : 0
1024 - 1518 Byte Frames         : 0
1519 - MAX Byte Frames          : 0
> MAX Byte Frames               : 0
Rx Frame Starts                 : 16
Multicast data  OK  Frame       : 16
Broadcast data OK   Frame       : 0
Unicast data OK   Frames        : 0
Multicast Control Frames        : 0
Broadcast Control Frames        : 0
Unicast Control Frames          : 0
```

```
Pause Control Frames              : 0
Data and padding octets           : 800
Frame octets:                     : 1088
================================================================================
==========
                        STATISTICS FOR BASE 20480
(Tx)

================================================================================
==========
Fragmented Frames                 : 0
Jabbered Frames                   : 0
Any Size with FCS Err Frame       : 0
Right Size with FCS Err Fra       : 0
Multicast data  Err Frames        : 0
Broadcast data Err  Frames        : 0
Unicast data Err  Frames          : 0
Multicast control  Err Frame      : 0
Broadcast control Err  Frame      : 0
Unicast control Err  Frames       : 0
Pause control Err  Frames         : 0
64 Byte Frames                    : 0
65 - 127 Byte Frames              : 16
128 - 255 Byte Frames             : 0
256 - 511 Byte Frames             : 0
512 - 1023 Byte Frames            : 0
1024 - 1518 Byte Frames           : 0
1519 - MAX Byte Frames            : 0
> MAX Byte Frames                 : 0
Tx Frame Starts                   : 16
Multicast data  OK  Frame         : 16
Broadcast data OK   Frame         : 0
Unicast data OK   Frames          : 0
Multicast Control Frames          : 0
Broadcast Control Frames          : 0
Unicast Control Frames            : 0
Pause Control Frames              : 0
Data and padding octets           : 800
Frame octets:                     : 1088
------------ Done --------------------
```

1.7. Register Maps

This section displays the available register address range for F-Tile Ethernet Intel FPGA Hard IP and F-Tile Auto-Negotiation and Link Training for Ethernet Intel FPGA IPs.

The F-Tile Ethernet Intel FPGA Hard IP supports up to 16 Ethernet IP instances. The address range for each IP instance is shown in the *Register Map for Multi-Rate IP Instances, PTP, and AN/LT* table. The table also displays the address range for PTP-related blocks and auto-negotiation and link training (AN/LT).

Note: The PTP and AN/LT address range is the same regardless of whether the PTP and AN/LT features are enabled or not.

Table 3. Register Map for Multi-Instance IP Design Examples, PTP, and AN/LT

The address range is specified as a byte address.

Address Range [28:0]	Module
0x0000_0000 - 0x00FF_FFFC	Ethernet IP : Instance 0
0x0100_0000 - 0x01FF_FFFC	Ethernet IP : Instance 1
0x0200_0000 - 0x02FF_FFFC	Ethernet IP : Instance 2

continued...

Address Range [28:0]	Module
0x0300_0000 - 0x03FF_FFFC	Ethernet IP : Instance 3
0x0400_0000 - 0x04FF_FFFC	Ethernet IP : Instance 4
0x0500_0000 - 0x05FF_FFFC	Ethernet IP : Instance 5
0x0600_0000 - 0x06FF_FFFC	Ethernet IP : Instance 6
0x0700_0000 - 0x07FF_FFFC	Ethernet IP : Instance 7
0x0800_0000 - 0x08FF_FFFC	Ethernet IP : Instance 8
0x0900_0000 - 0x09FF_FFFC	Ethernet IP : Instance 9
0x0A00_0000 - 0x0AFF_FFFC	Ethernet IP : Instance 10
0x0B00_0000 - 0x0BFF_FFFC	Ethernet IP : Instance 11
0x0C00_0000 - 0x0CFF_FFFC	Ethernet IP : Instance 12
0x0D00_0000 - 0x0DFF_FFFC	Ethernet IP : Instance 13
0x0E00_0000 - 0x0EFF_FFFC	Ethernet IP : Instance 14
0x0F00_0000 - 0x0FFF_FFFC	Ethernet IP : Instance 15
0x1000_0000 - 0x1000_FFFC	PTP Master TOD registers
0x1001_5000 - 0x1001_5FFC	PTP Adapter: Asymmetry Delay
0x1002_5000 - 0x1002_5FFC	PTP Adapter: Peer-to-Peer MeanPathDelay
0x1003_0000 - 0x100F_FFFC	Reserved
0x1010_0000 - 0x101F_FFFC	Auto-negotiation and link training

The table below displays the register address map within a single Ethernet IP instance.

Table 4. Register Address Map for Single Ethernet IP Instance

The address range is specified as a byte address.

Address Range [28:0]	Module
0x0000_0000 - 0x0000_FFFC	Ethernet reconfiguration interface
0x0001_0000 - 0x000F_FFFC	Reserved
0x0010_0000 - 0x007F_FFFC	Packet Client
0x0080_0000 - 0x008F_FFFC	Transceiver reconfiguration interface for lane 0
0x0090_0000 - 0x009F_FFFC	Transceiver reconfiguration interface for lane 1
0x00A0_0000 - 0x00AF_FFFC	Transceiver reconfiguration interface for lane 2
0x00B0_0000 - 0x00BF_FFFC	Transceiver reconfiguration interface for lane 3
0x00C0_0000 - 0x00CF_FFFC	Transceiver reconfiguration interface for lane 4
0x00D0_0000 - 0x00DF_FFFC	Transceiver reconfiguration interface for lane 5
0x00E0_0000 - 0x00EF_FFFC	Transceiver reconfiguration interface for lane 6
0x00F0_0000 - 0x00FF_FFFC	Transceiver reconfiguration interface for lane 7

For example, the Packet Client base address in the 2nd Ethernet IP instance is equivalent to 0x0100_0000 + 0x0010_0000 = 0x0110_0000.

2. Design Example: Single IP Core Instantiation

The single instance IP core design example supports all F-tile supported Ethernet rates and demonstrates the basic functions of the F-Tile Ethernet Intel FPGA Hard IP.

To generate the design example, you must first set the parameter values for the IP core variation you intend to generate in your end product. Generating the design example creates a copy of the IP core; the testbench and hardware design example use this variation as the DUT. If your parameter values for the DUT don't match the parameter values in your end product, the design example you generate does not exercise the IP core variation you intend.

The following IP parameter settings were used to generate this design example:

Table 5. IP Parameters for 200G Ethernet Mode with 2 Lanes Design Example

Table specifies parameter settings used to generate this design example.

Selected IP Parameter Settings	Value
General Options	
PMA type	FGT
Ethernet mode	200GE-2
Client interface	MAC segmented
FEC mode	IEEE 802.3 RS(544,514) (CL134)
PMA reference frequency	156.25
System PLL frequency	830.078125

For more information about steps on how to generate a design example, refer to the *Generating the Design Example*.

Related Information

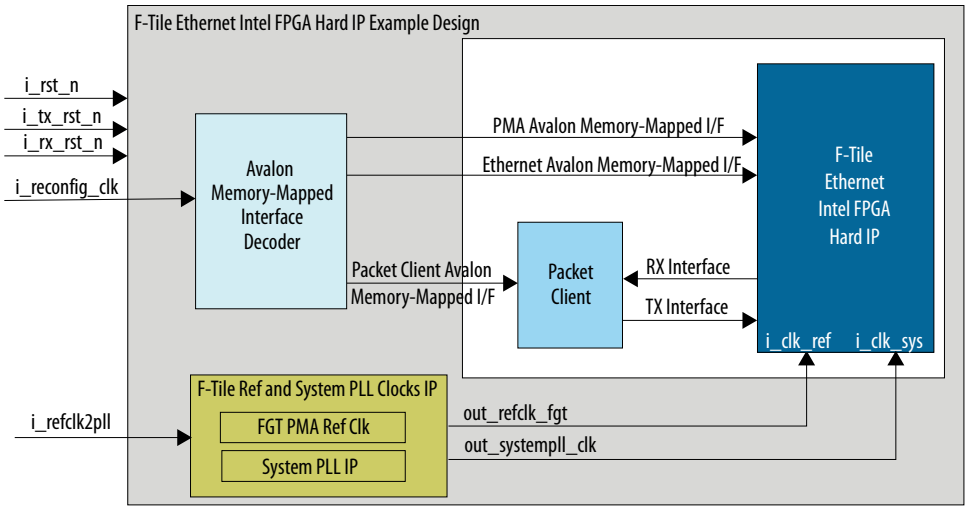
Generating Single IP Instance Design on page 4

2.1. Features

- Supports 10G, 25G, 40G, 50G, 100G, 200G, and 400G Ethernet rates
- Supports Avalon® streaming interface for 10G, 25G, 40G, 50G, and 100G Ethernet rates with synchronized or asynchronous adapter
- Supports MAC segmented interface
- Instantiates F-Tile Reference and System PLL Clocks Intel FPGA IP based on Ethernet configuration

2.2. Functional Description

Figure 8. F-Tile Ethernet Intel FPGA Hard IP Simulation Design Example Block Diagram with FGT PMA



The F-Tile Ethernet Intel FPGA Hard IP design example includes the following components:

- **F-Tile Ethernet Intel FPGA Hard IP**: Generated IP core.
- **F-Tile Reference and System PLL Clocks Intel FPGA IP**: Instantiated reference clock and system PLL clock IP. The F-Tile Reference and System PLL Clocks Intel FPGA IP parameter editor settings align with the **System PLL frequency** and **PMA reference frequency** parameter settings in the F-Tile Ethernet Intel FPGA Hard IP. If you generate the design example using **Generate Example Design** button in the IP parameter editor, the IP instantiates automatically. If you create your own design example, you must manually instantiate this IP and connect all I/O ports.

  For information about supported system PLL modes, refer to *F-Tile Ethernet Intel FPGA Hard IP User Guide*. For information about this IP, refer to *F-Tile Architecture and PMA and FEC Direct PHY IP User Guide*.
- **Packet Client**: Consists of a packet generator, a packet checker and a loopback client. The Packet Client generates various ROM-based traffic patterns for MAC mode and can loopback the RX and TX client side.
- **Avalon memory-mapped interface Decoder**: Decodes the Avalon memory-mapped interface address to Hardware IP Top. For base address for each of the Avalon memory-mapped interface accessed instances, refer to Register Maps on page 12.

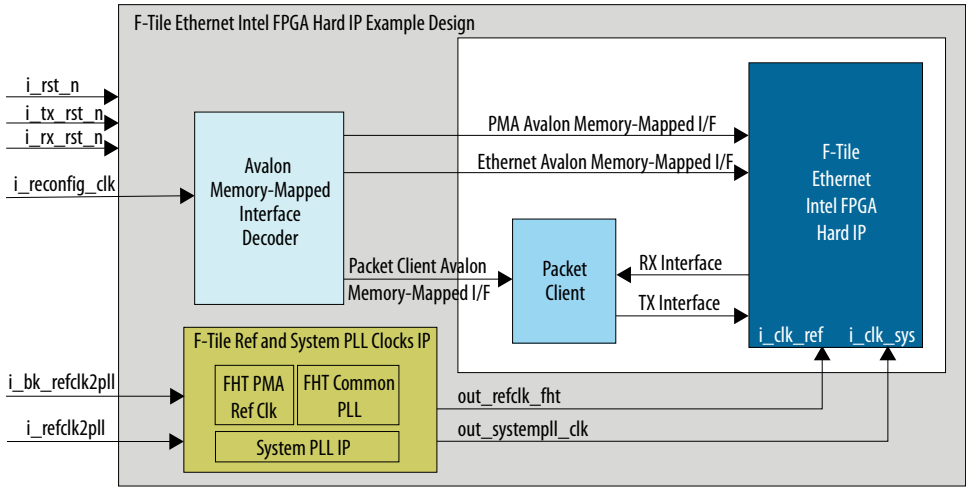
Related Information

- Variation: F-Tile Ethernet Intel FPGA Hard IP with FHT PMA on page 16
- F-Tile Architecture and PMA and FEC Direct PHY IP User Guide

2.2.1. Variation: F-Tile Ethernet Intel FPGA Hard IP with FHT PMA

This section displays F-Tile Ethernet Intel FPGA Hard IP block diagram when you select **FHT** for PMA type in the IP Parameter Editor. In the FHT PMA variation, a separate clock feeds the FHT PMA reference clock block.

Figure 9. F-Tile Ethernet Intel FPGA Hard IP Simulation Design Example Block Diagram with FHT PMA



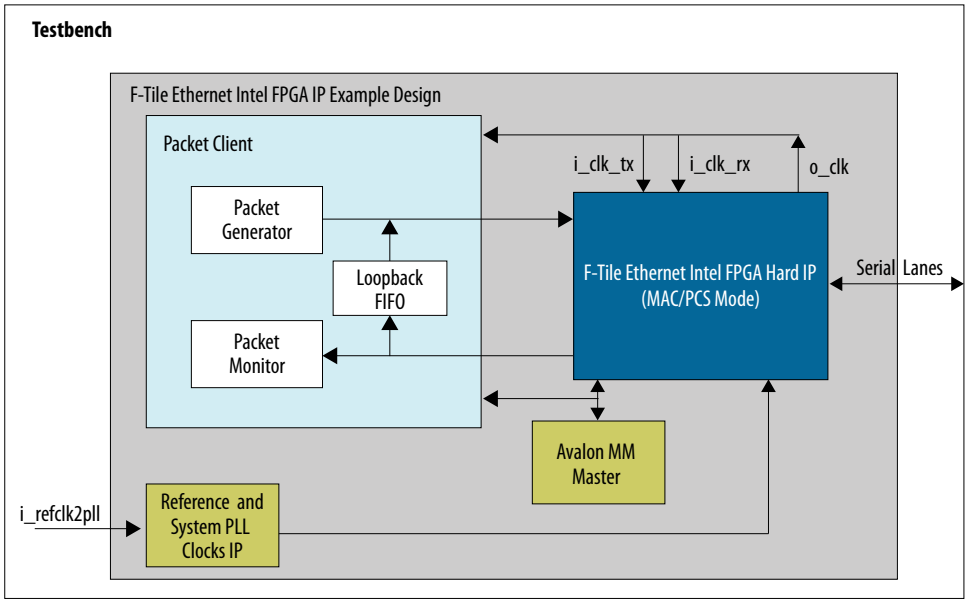
In this variation, the system PLL include additional FHL common PLL block.

2.3. Simulation

The testbench provides basic functionality such as the startup and wait for lock and send and receive a few packets using the ROM-based packet generator.

You can enable the Fast Sim model to speed up the duration of your simulation. For more information, refer to Fast Sim Model for FGT Variants on page 9.

Figure 10. F-Tile Ethernet Intel FPGA Hard IP Simulation Design Example Block Diagram



The following sections describe the simulation testbench flow variations based on the selected client interface.

Related Information

Fast Sim Model for FGT Variants on page 9

2.3.1. Simulation Testbench Flow for MAC Mode

The following steps show the simulation testbench flow for MAC mode:

1. Assert global reset (`i_rst_n`) to reset the F-Tile Ethernet Intel FPGA Hard IP.
2. Wait until resets acknowledgment. The `o_rst_ack_n` signal goes low.
3. Deasserts the global reset.
4. Wait until `o_tx_lanes_stable` bit is set to `1`, indicating TX path is ready.
5. Wait until `o_rx_pcs_ready` bit is set to `1`, indicating RX path is ready.
6. Read TX packet data information from 0x00 - 0x34 registers in sequential order.

- 0x00: Set hw_pc_ctrl[6] = 1'b1 to enable snapshot bit to read the TX packet statistics.
- 0x020/0x24: TX start of packet counter (LSB/MSB)
- 0x28/0x2C: TX end of packet counter (LSB/MSB)
- 0x30/0x34: TX error counter (LSB/MSB)
- 0x00: Set hw_pc_ctrl[6] = 1'b0 to disable snapshot bit.

7. Read RX packet data information from 0x38 - 0x4C registers in sequential order.
   - 0x00: Set hw_pc_ctrl[6] = 1'b1 to enable snapshot bit to read the RX packet statistics.
   - 0x38/0x3C: RX start of packet counter (LSB/MSB)
   - 0x40/0x44: RX end of packet counter (LSB/MSB)
   - 0x48/0x4C: RX error counter (LSB/MSB)
   - 0x00: Set hw_pc_ctrl[6] = 1'b0 to disable snapshot bit.
8. Compare read counters to ensure 16 packets were sent and received.
9. Instruct packet client to stop data transmission by writing hw_pc_ctrl[2:0]=3'b100 to stop the packet generator. Clear counters.
10. Perform Avalon memory-mapped interface test. Write and read Ethernet IP registers.
    - 0x104: Scratch register
    - 0x108: Ethernet IP soft reset register
    - 0x214: TX MAC source address register [31:0]
    - 0x218: TX MAC source address register [47:32]
    - 0x21C: RX MAC frame size register
11. Perform Avalon memory-mapped interface 2 test. Write and read transceiver registers.

The following sample output illustrates a successful simulation test run.

```
---SRC IP sequence started -----
---SRC IP sequence TX completed -----
---SRC IP sequence RX completed -----
---Test    0;   ---Total     16 packets to send-----
------Start pkt gen TX-----
------Checking Packet TX/RX result-----
------------  16 packets Sent;     0 packets Received--------
------ALL   16  packets Sent out---
------------  16 packets Sent;    16 packets Received--------
------ALL   16  packets Received---
------TX/RX packet check OK---
****Starting AVMM Read/Write****
===>MATCH! ReaddataValid = 1 Readdata = abcdef01 Expected_Readdata = abcdef01
===>MATCH! ReaddataValid = 1 Readdata = 00000007 Expected_Readdata = 00000007
===>MATCH! ReaddataValid = 1 Readdata = 00000000 Expected_Readdata = 00000000
===>MATCH! ReaddataValid = 1 Readdata = 9d228c3a Expected_Readdata = 9d228c3a
===>MATCH! ReaddataValid = 1 Readdata = 4338b586 Expected_Readdata = 4338b586
===>MATCH! ReaddataValid = 1 Readdata = deadc0de Expected_Readdata = deadc0de
===>MATCH! ReaddataValid = 1 Readdata = deadc0de Expected_Readdata = deadc0de
===>MATCH! ReaddataValid = 1 Readdata = 00000000 Expected_Readdata = 00000000
===>MATCH! ReaddataValid = 1 Readdata = 22334455 Expected_Readdata = 22334455
===>MATCH! ReaddataValid = 1 Readdata = 00000011 Expected_Readdata = 00000011
===>MATCH! ReaddataValid = 1 Readdata = 000005ee Expected_Readdata = 000005ee
```

```
===>MATCH! ReaddataValid = 1 Readdata = 01234567 Expected_Readdata = 01234567
===>MATCH! ReaddataValid = 1 Readdata = 000089ab Expected_Readdata = 000089ab
743830ns  Try to access AVMM2 begin...
743830ns  write 0x00000065 to xcvr  0 address 0x103c004
744795ns  Try to access AVMM2 end...
744890ns  read from address 0x103c004
====>MATCH!     ReaddataValid = 1 Readdata = 00000065 Expected_Readdata =
00000065

...

758740ns  Try to access AVMM2 end...
758840ns  Try to access AVMM2 begin...
758840ns  write 0x0000006c to xcvr  7 address 0x103c00b
759825ns  Try to access AVMM2 end...
759920ns  read from address 0x103c00b
====>MATCH!     ReaddataValid = 1 Readdata = 0000006c Expected_Readdata =
0000006c
760900ns  Try to access AVMM2 end...
**** AVMM Read/Write Operation Completed ****
** Testbench complete
**
```

2.3.2. Simulation Testbench Flow for PCS, OTN, and FlexE Modes

The following steps show the simulation testbench flow for PCS, OTN, and FlexE modes:

1. Assert global reset (`i_rst_n`) to reset the F-Tile Ethernet Intel FPGA Hard IP.
2. Wait until resets acknowledgment. The `o_rst_ack_n` signal goes low.
3. Deasserts the global reset.
4. Wait until `o_tx_lanes_stable` bit is set to 1, indicating TX path is ready.
5. Wait until `o_rx_pcs_ready` bit is set to 1, indicating RX path is ready.

   Note: In non-MAC mode, the packet client sends out idle data in MII/PCS66 format.
6. Read TX packet data information from 0x00 - 0x34 registers in sequential order.
   - 0x00: Set `hw_pc_ctrl[6]` = 1'b1 to enable snapshot bit to read the TX packet statistics.
   - 0x020/0x24: TX start of packet counter (LSB/MSB)
   - 0x28/0x2C: TX end of packet counter (LSB/MSB)
   - 0x00: Set `hw_pc_ctrl[6]` = 1'b0 to disable snapshot bit.
7. Read RX packet data information from 0x38 - 0x4C registers in sequential order.
   - 0x00: Set `hw_pc_ctrl[6]` = 1'b1 to enable snapshot bit to read the RX packet statistics.
   - 0x38/0x3C: RX start of packet counter (LSB/MSB)
   - 0x40/0x44: RX end of packet counter (LSB/MSB)
   - 0x48/0x4C: RX error counter (LSB/MSB)
   - 0x00: Set `hw_pc_ctrl[6]` = 1'b0 to disable snapshot bit.
8. Compare the counters to ensure 16 packets were sent and received.

9. Instruct packet client to stop data transmission. Write `cfg_start_pkt_gen[0]=1'b1` to stop the packet generator. Clear counters.
10. Perform Avalon memory-mapped interface test. Write and read Ethernet IP registers.
    - 0x104: Scratch register
    - 0x108: Ethernet IP soft reset register
    - 0x004: Ethernet IP debug configuration control register
    - 0x008: Ethernet IP enable/clock gating configuration register
11. Perform Avalon memory-mapped interface 2 test. Write and read transceiver registers.

2.4. Packet Client Registers

Table 6. Packet Client Registers for MAC Segmented Interface and MAC Avalon ST Interface

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

Note: All address offsets are specified in the byte address format.

Address	Name	Bit Offset	Default Value	Access	Description
0x00	hw_pc_ctrl	0	1'b0	RW	Indicates start and stop of the TX packet. • 0: Stop packet generator • 1: Start packet generator
		2	1'b0	RW	Indicates packet transmission mode. • 0: One time mode • 1: Continuous mode When you switch from continuous to one time mode, you must set `hw_pc_ctrl[0]` to 1 to ensure the pending packet transmission is completed.
		4	1'b0	RW	Indicates packet client loopback. • 0: Packet generator data path • 1: Loopback client data path
		6	1'b0	RW	Indicates the snapshot status for all statistics counter registers. • 0: No snapshot • 1: Snapshot available
		7	1'b0	RW	Indicates the clear status of the snapshot registers. • 0: Do not clear the internal registers • 1: Clear the internal registers
		8	1'b0	RW	Indicates the EOP of TX packet and clears counters.
0x04	hw_test_loop_cnt	[15:0]	16'd1	RW	Indicates the number of times the ROM packet data are sent.

continued...

Address	Name	Bit Offset	Default Value	Access	Description
0x08	hw_test_rom_addr	[15:0]	16'd0	RW	ROM packet data start address
		[31:16]	16'd0	RW	ROM packet data end address
0x20	stat_tx_sop_cnt_lsb	[31:0]	32'b0	RO	Lower 32-bits of the TX start-of-packet (SOP) counter
0x24	stat_tx_sop_cnt_msb	[31:0]	32'b0	RO	Upper 32-bits of the TX start-of-packet (SOP) counter
0x28	stat_tx_eop_cnt_lsb	[31:0]	32'b0	RO	Lower 32-bits of the TX end-of-packet (EOP) counter
0x2C	stat_tx_eop_cnt_msb	[31:0]	32'b0	RO	Upper 32-bits of the TX end-of-packet (EOP) counter
0x30	stat_tx_err_cnt_lsb	[31:0]	32'b0	RO	Lower 32-bits of the TX error counter
0x34	stat_tx_err_cnt_msb	[31:0]	32'b0	RO	Upper 32-bits of the TX error counter
0x38	stat_rx_sop_cnt_lsb	[31:0]	32'b0	RO	Lower 32-bits of the RX start-of-packet (SOP) counter
0x3C	stat_rx_sop_cnt_msb	[31:0]	32'b0	RO	Upper 32-bits of the RX start-of-packet (SOP) counter
0x40	stat_rx_eop_cnt_lsb	[31:0]	32'b0	RO	Lower 32-bits of the RX end-of-packet (EOP) counter
0x44	stat_rx_eop_cnt_msb	[31:0]	32'b0	RO	Upper 32-bits of the RX end-of-packet (EOP) counter
0x48	stat_rx_err_cnt_lsb	[31:0]	32'b0	RO	Lower 32-bits of the RX error counter
0x4C	stat_rx_err_cnt_lsb	[31:0]	32'b0	RO	Upper 32-bits of the RX error counter

Table 7. Packet Client Registers for PCS, OTN, and FlexE Interface

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

Note: All address offsets are specified in the byte address format.

Address	Name	Bit Offset	Default Value	Access	Description
0x10	cfg_start_pkt_gen	0	1'b0	RW	Indicates start and stop of the TX packet. • 0: Stop packet generator. • 1: Start packet generator.
0x58	cfg_clear_counters	0	1'b0	RW	When set, clears packet generator counters.
0x18	stat_tx_sop_cnt	[31:0]	32'b0	RO	Lower 32-bits of the TX start-of-packet (SOP) counter
0x1C	stat_tx_sop_cnt	[63:32]	32'b0	RO	Upper 32-bits of the TX start-of packet (SOP) counter
0x20	stat_tx_eop_cnt	[31:0]	32'b0	RO	Lower 32-bits of the TX end-of-packet (EOP) counter
0x24	stat_tx_eop_cnt	[63:32]	32'b0	RO	Upper 32-bits of the TX end-of-packet (EOP) counter

continued...

Address	Name	Bit Offset	Default Value	Access	Description
0x30	stat_rx_sop_cnt	[31:0]	32'b0	RO	Lower 32-bits of the RX start-of-packet (SOP) counter
0x34	stat_rx_sop_cnt	[63:32]	32'b0	RO	Upper 32-bits of the RX start-of-packet (SOP) counter
0x38	stat_rx_eop_cnt	[31:0]	32'b0	RO	Lower 32-bits of the RX end-of-packet (EOP) counter
0x3C	stat_rx_eop_cnt	[63:32]	32'b0	RO	Upper 32-bits of the RX end-of-packet (EOP) counter
0x50	stat_rx_err_cnt	[31:0]	32'b0	RO	Lower 32-bits of the RX error status
0x54	stat_rx_err_cnt	[63:32]	32'b0	RO	Upper 32-bits of the RX error status

For information about register base and offset addresses, refer to the *F-Tile Ethernet Intel FPGA Hard IP User Guide*.

Related Information

F-Tile Ethernet Intel FPGA Hard IP User Guide

3. Design Example: Single IP Core Instantiation with Precision Time Protocol

The single instance IP core design example supports Ethernet rates with enabled Precision Time Protocol (PTP) and demonstrates the basic functions of the F-Tile Ethernet Intel FPGA Hard IP.

To generate the design example, you must first set the parameter values for the IP core variation you intend to generate in your end product. Generating the design example creates a copy of the IP core; the testbench and hardware design example use this variation as the DUT. If your parameter values for the DUT don't match the parameter values in your end product, the design example you generate does not exercise the IP core variation you intend.

The following IP parameter settings were used to generate this design example:

Table 8. Selected IP Parameter Settings

Table specifies parameter settings used to generate this design example.

Selected IP Parameter Settings	Value
General Options	
PMA type	FGT
Ethernet mode	400GE-8
Client interface	MAC segmented
FEC mode	Ethernet Technology Consortium RS(272,514)
PMA reference frequency	156.25
System PLL frequency	830.078125
MAC Options	
PTP	
Enable IEEE 1588 PTP	√
Timestamp accuracy mode	Advanced
Timestamp fingerprint width	8

For more information about steps of how to generate a design example, refer to the *Generating the Design Example*.

Related Information

Generating Single IP Instance Design on page 4

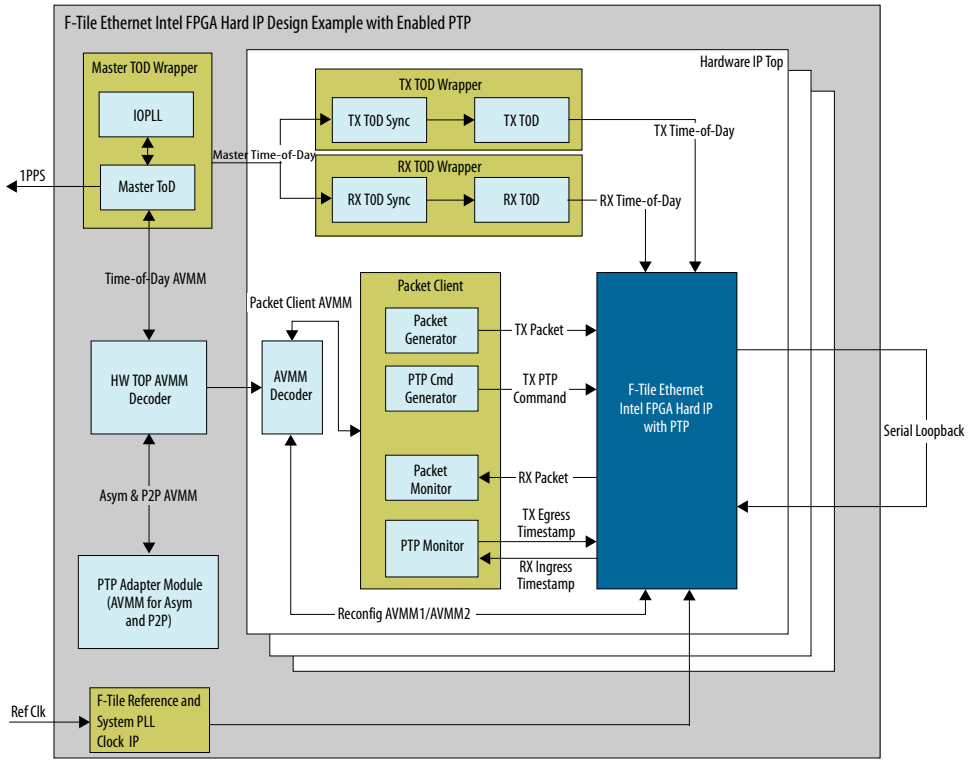
3.1. Features

- Supports 10G, 25G, 50G, 100G, 200G, and 400G Ethernet rates
- Supports Avalon streaming interface for 10G, 25G, 50G, and 100G Ethernet rates with synchronous or asynchronous adapter
- Supports MAC segmented interface for all Ethernet rates
- Instantiates PTP adapter module (`eth_ptp_adpt_f`)
- Instantiates Ethernet IEEE 1588 TOD and TOD Synchronizer Intel FPGA IP based on Ethernet configuration
- Instantiates F-Tile Reference and System PLL Clocks Intel FPGA IP based on Ethernet configuration

3.2. Functional Description

When generating a design example with PTP option enabled, the software instantiates PTP-specific components.

Figure 11. F-Tile Ethernet Intel FPGA Hard IP: Design Example with Enabled PTP



The F-Tile Ethernet Intel FPGA Hard IP design example includes the following components:

- **F-Tile Ethernet Intel FPGA Hard IP**: Generated IP core with enabled PTP option.
- **F-Tile Reference and System PLL Clocks Intel FPGA IP**: Instantiate reference clock and system PLL. For information about supported system PLL modes, refer to *F-Tile Ethernet Intel FPGA Hard IP*. For information about this IP, refer to *F-Tile Architecture and PMA and FEC Direct PHY IP User Guide*.
- **Time-of-Day (TOD)**: Provides a continuous flow of a current time-of-day information to the IP core. The master TOD runs at 125 MHz clock frequency. TX TOD and RX TOD, which are clocked by `div66` or `div68` clock of the F-Tile Ethernet Intel FPGA Hard IP, synchronize to the master TOD through their respective TOD synchronizers.

  Master TOD also generates a one pulse per second (1 pps) to monitor the accuracy and allows synchronization between multiple devices.

  In this user guide, the generated design example assumes 0 ppm delay. In your design, drive the master TOD with the most accurate clock.
- **Packet Client**: Consists of multiple PTP related modules. The Packet Client does not support the PTP functionality when packet loopback is set from RX to TX in client side.
- **Packet Generator**: Configures to loop over various patterns in the ROM. Alternatively, can perform a single transmission.
- **PTP Command Generator**: The PTP command generation module in the Packet Client generates PTP command for the packet in transmission. The generated command aligns to start-of-packet (SOP) for Avalon streaming interface and MAC segmented based interface.
- **Packet Monitor**: Stores sent and received packet information between packet client and the IP core.
- **PTP Monitor**: The module stores the PTP information sent from/to the Packet Client to/from the F-Tile Ethernet Intel FPGA Hard IP when packet loopbacks from TX serial to RX serial.
- **PTP Adapter Module (Avalon memory-mapped interface for Asymmetry and Peer-to-Peer (P2P)**: If you enabled PTP option in the F-Tile Ethernet Intel FPGA Hard IP, you must instantiate this module and connect it to all associated Ethernet IP cores. The design allows only one instance per tile.

  When instantiated, the module provides the access to the Avalon memory-mapped interface registers, specifically `Asymmetry Delay` and `P2P MeanPathDelay` registers.
- **Avalon memory-mapped interface Decoder**: Decodes the Avalon memory-mapped interface address to Hardware IP Top, master TOD, and PTP adapter. For base address for each of the Avalon memory-mapped interface accessed instances, refer to Register Maps on page 12.

Related Information

- F-Tile Architecture and PMA and FEC Direct PHY IP User Guide
- Register Maps on page 12

3.3. Simulation

The testbench provides basic functionality such as the startup and wait for lock and send and receive a few packets utilizing the PTP feature.

Since Fast Sim model does not support the PTP timestamp accuracy, the model is not enabled by default in the designs with enabled PTP feature. To utilize the Fast Sim model for functional purpose simulation, refer to Fast Sim Model for FGT Variants on page 9.

Note: In PTP design examples, the simulation printout displays the timestamp accuracy as RX ITS - TX ETS information.

1. Assert global reset (`i_rst_n`) to reset the F-Tile Ethernet Intel FPGA Hard IP.
2. Wait until resets acknowledgment. The `o_rst_ack_n` signal goes low.
3. Deasserts the global reset.
4. Wait until `o_tx_lanes_stable` bit is set to 1, indicating TX path is ready.
5. Wait until `o_rx_pcs_ready` bit is set to 1, indicating RX path is ready.
6. Perform the PTP user flow until `tx_ptp_ready` and `rx_ptp_ready` signals are set to 1.
7. Write to PTP Asymmetry Delay and peer-to-peer MeanPathDelay Avalon memory-mapped interface registers.
8. Reset the PTP monitor.
9. Read TX packet data information from 0x00 - 0x34 registers in sequential order.
   - 0x00: Set `hw_pc_ctrl[6]` = 1'b1 to enable snapshot bit to read the TX packet statistics.
   - 0x020/0x24: TX start of packet counter (LSB/MSB)
   - 0x28/0x2C: TX end of packet counter (LSB/MSB)
   - 0x30/0x34: TX error counter (LSB/MSB)
   - 0x00: Set `hw_pc_ctrl[6]` = 1'b0 to disable snapshot bit.
10. Read RX packet data information from 0x38 - 0x4C registers in sequential order.
    - 0x00: Set `hw_pc_ctrl[6]` = 1'b1 to enable snapshot bit to read the RX packet statistics.
    - 0x38/0x3C: RX start of packet counter (LSB/MSB)
    - 0x40/0x44: RX end of packet counter (LSB/MSB)
    - 0x48/0x4C: RX error counter (LSB/MSB)
    - 0x00: Set `hw_pc_ctrl[6]` = 1'b0 to disable snapshot bit.
11. Compare read counters to ensure 16 packets were sent and received.
12. Instruct packet client to stop data transmission and clear the counters by writing `hw_pc_ctrl[2:0]=3'b100` to stop the packet generator.
13. Start the PTP checker.

a. Wait until the packet transmission is complete. Poll the TX_PKT_VALID bit to monitor the transmission status.

b. Read TX packet data information from 0x102 - 0x104 registers in sequential order, starting from the 0x102 register. The PTP monitor logic refreshes the 0x102 - 0x104 registers content to next data when read from 0x104 register.

c. Repeat the previous step until the TX_PKT_EOP bit is set to 1 indicating the read operation reached the end of the packet.

d. Read TX PTP command information from 0x105 - 0x10A registers in sequential order, starting from the 0x105 register. The PTP monitor logic refreshes the 0x105 - 0x10A registers content to next data when read from 0x10A register.

e. If TX PTP command indicates a PTP packet:

   i. Wait until the TX egress timestamp is available. Use the TX_PTP_ETS_VALID signal to monitor the status.

   ii. Read the TX egress timestamp from the 0x10C - 0x10F registers in sequential order, starting with the 0x10C register.

   The PTP monitor logic refreshes content of 0x10C - 0x10F registers to next data when read from the 0x10F register.

f. Wait until the packets are loop backed in the RX data path. Poll the RX_PKT_VALID bit to monitor the transmission status.

g. Read RX packet data information from 0x110 - 0x112 registers in sequential order, starting with the 0x110 register. The PTP monitor logic refreshes the 0x110 - 0x112 registers content to next data when read from 0x112 register.

h. Repeat the previous step until the RX_PKT_EOP bit is set to 1 indicating the read operation reached the end of the packet.

i. Read RX ingress timestamp.

   i. Read the RX ingress timestamp from the 0x114 - 0x116 registers in sequential order, starting with the 0x14 register. The PTP monitor logic refreshes the 0x114 - 0x116 registers content to next data when read from 0x116 register.

j. Display the TX packet content, RX packet content, TX PTP commands, TX egress timestamp, and RX ingress timestamp information.

k. Display the comparison information. Note that in 1-step commands, the TX/RX packets and PTP commands field content is the same.

l. Repeat steps b through step k until the system processes all packets.

   Note: The design example simulation performs step 14 for only first six packets to reduce long simulation time.

14. Perform Avalon memory-mapped interface test on PTP-related registers.
    - Selective PTP Assymmetry Delay and P2P MeanPathDelay registers
    - Selective Master TOD Avalon memory-mapped interface registers
15. Perform Avalon memory-mapped interface test. Write and read Ethernet IP registers.

- 0x104: Scratch register
- 0x108: Ethernet IP soft reset register
- 0x214: TX MAC source address register [31:0]
- 0x218: TX MAC source address register [47:32]
- 0x21C: RX MAC frame size register

16. Perform Avalon memory-mapped interface 2 test. Write and read transceiver registers.

The following sample output illustrates a successful simulation test.

```
--------------------
TX Packet #1
--------------------
------------------------------------------------------------
| | 0 1 2 3 4 5 6 7 8 9 A B C D E F |
------------------------------------------------------------
| 0000 | 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F |
| 0010 | 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F |
| 0020 | 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F |
| 0030 | 30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F |
| 0040 | 40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F |
| 0050 | 50 51 52 53 54 55 56 57 |
------------------------------------------------------------
--------------------
RX Packet #1
--------------------
------------------------------------------------------------
| | 0 1 2 3 4 5 6 7 8 9 A B C D E F |
------------------------------------------------------------
| 0000 | 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F |
| 0010 | 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F |
| 0020 | 20 21 22 23 24 25 26 27 00 00 2A 2B 2C 2D 2E 2F |
| 0030 | 30 31 32 33 34 35 36 37 AB CD 3A 3B 3C 3D 3E 3F |
| 0040 | 40 41 42 43 44 45 46 47 48 49 4A 4B 00 00 00 00 |
| 0050 | 00 23 45 67 89 AB 56 57 CC 12 34 CC |
------------------------------------------------------------
--------------------
TX PTP Command #1
--------------------
Egress Timestamp Request : 0
Insert Timestamp : 1
Update CorrectionField : 0
Clear UDP/IPv4 Checksum : 1
Update UPD/IPv6 Extended Bytes : 0
Add Peer-to-Peer (P2P) MinPathDelay : 0
Add Asymmetry Delay : 0
Asymmetry Delay Sign : 0
Asymmetry & P2P MinPathDelay Select : 123
Timestamp Field Offset : 0x004C
Correction Field Offset : 0x0032
Checksum Field Offset : 0x0028
Timestamp Fingerprint : 0x02
TX Ingress Timestamp : 0x0
--------------------
Timestamps #1
--------------------
TX Egress Timestamp : 0x000000000023456789AB7394
RX Ingress Timestamp : 0x000000000023456789BCDEF0
TX User Fingerprint : 0x02
TX Returned Fingerprint : 0x02
TX Egress Virtual Lane : 2
RX Ingress Virtual Lane : 2
--------------------
Comparison #1
--------------------
RX ITS - TX ETS : 0x116B5C / 17.419ns
```

```
TX Timestamp Fields : 0x4C4D4E4F505152535455
RX Timestamp Fields : 0x000000000023456789AB
TX Correction Fields : 0x3233343536373839
RX Correction Fields : 0x323334353637ABCD
TX Checksum Fields : 0x2829
RX Checksum Fields : 0x0000
```

Related Information

- Register Maps on page 12
- Registers on page 29

3.4. Registers

The PTP monitor registers are part of the Packet Client block of each Ethernet IP instance. The Packet Client base address for each IP instance is available in Register Maps on page 12.

For example, the PTP monitor registers address in the third Ethernet IP instance is equivalent to 0x0210_0400, where:

- 0x0200_0000 is the Ethernet IP: Instance 2 base address
- 0x0010_0000 is the Packet Client base address for the specified IP instance
- 4*0x100 is the PTP monitor base address for the specified IP instance

Table 9. PTP Monitor Registers

All address offsets are specified in the word address format.

Word Address	Name	Bit Offset	Default Value	Access	Description
0x100	Soft Reset	0	1'b0	RW	Soft reset for PTP monitor. Prior to every burst iteration, you must write 1'b1 followed by 1'b0.
0x101	TX_PKT_VALID	0	1'b0	RO	Indicates the TX packet data is available. • 0: No packet content is available for read out. • 1: Packet content is available.
	TX_PTP_ETS_VALID	1	1'b0	RO	Indicates the TX egress timestamp is available. • 0: No timestamp is available for read out. • 1: Timestamp is available.
	RX_PKT_VALID	2	1'b0	RO	Indicates the RX packet data is available. • 0: No packet content is available for read out. • 1: Packet content is available.
0x102	TX_PKT_DATA_63_32	[31:0]	32'h0	RO	Indicates the TX packet data for 64-bit segment.
0x103	TX_PKT_DATA_31_0	[31:0]	32'h0	RO	
0x104	TX_PKT_INFRAME	0	1'b0	RO	Specifies TX packet inframe.
	TX_PKT_SOP	1	1'b0	RO	Indicates the start-of-packet (SOP) for TX packet.

continued...

Word Address	Name	Bit Offset	Default Value	Access	Description
	TX_PKT_EOP	2	1'b0	RO	Indicates the end-of-packet (EOP) for TX packet.
	TX_PKT_EMPTY	[5:3]	3'b000	RO	Indicates that TX packet is empty.
	TX_PKT_ERROR	6	1'b0	RO	Indicates a TX packet error.
	TX_PKT_SKIP_CRC	7	1'b0	RO	Indicates the setting of skip_crc signal for TX packet.
0x105	TX_PTP_TS_REQ	0	1'b0	RO	Indicates a received request for TX PTP timestamp.
	TX_PTP_INS_ETS	1	1'b0	RO	Indicates TX PTP insert egress timestamp.
	TX_PTP_INS_CF	2	1'b0	RO	Indicates that TX PTP updated the Correction Field with residence time.
	TX_PTP_ZERO_CSUM	3	1'b0	RO	TX PTP clear checksum field for IPv4 packet.
	TX_PTP_UPDATE_EB	4	1'b0	RO	Indicates that TX PTP updated extended bytes for IPv6 packets.
	TX_PTP_P2P	5	1'b0	RO	Indicates that TX PTP inserted peer-to-peer value for MeanPathDelay signal.
	TX_PTP_ASYM	6	1'b0	RO	Indicates that TX PTP inserted asymmetry delay.
	Reserved				
	TX_PTP_ASYM_SIGN	8	1'b0	RO	Indicates the asymmetry delay sign bit for TX PTP packets.
	TX_PTP_ASYM_PTP_IDX	[15:9]	7'h0	RO	TX PTP index from asymmetry and peer-to-peer lookup table.
	TX_PTP_TS_OFFSET	[31:16]	16'h0	RO	Indicates the TX PTP timestamp offset.
0x106	TX_PTP_CF_OFFSET	[15:0]	16'h0	RO	Indicates offset for the TX PTP CorrectionField signal.
	TX_PTP_CSUM_OFFSET	[31:16]	16'h0	RO	Indicates offset for the TX PTP ChecksumField signal.
0x107	TX_PTP_USR_FP	[31:0]	32'h0	RO	Specifies your TX PTP fingerprint. The valid range depends on the **PTP_FP_WIDTH** setting in the IP Parameter Editor.
0x108	TX_PTP_ITS_95_64	[31:0]	32'h0	RO	Specifies the 96-bit TX PTP ingress timestamp.
0x109	TX_PTP_ITS_63_32	[31:0]	32'h0	RO	
0x10A	TX_PTP_ITS_31_0	[31:0]	32'h0	RO	
0x10C	TX_PTP_ETS_FP	[31:0]	32'h0	RO	Specifies TX PTP fingerprint. The valid range depends on the **PTP_FP_WIDTH** setting in the IP Parameter Editor.
0x10D	TX_PTP_ETS_95_64	[31:0]	32'h0	RO	Specifies the 96-bit TX PTP egress timestamp.
0x10E	TX_PTP_ETS_63_32	[31:0]	32'h0	RO	
0x10F	TX_PTP_ETS_31_0	[31:0]	32'h0	RO	

continued...

Word Address	Name	Bit Offset	Default Value	Access	Description
0x110	RX_PKT_INFRAME	0	1'b0	RO	Specifies RX packet inframe.
	RX_PKT_SOP	1	1'b0	RO	Indicates the start-of-packet (SOP) for the RX packet.
	RX_PKT_EOP	2	1'b0	RO	Indicates the end-of-packet (EOP) for the RX packet.
	RX_PKT_EMPTY	[5:3]	3'b000	RO	Indicates that RX packet is empty.
	RX_PKT_ERROR	[11:6]		RO	Indicates a RX packet error.
	RX_PKT_FCS_ERROR	12	1'b0	RO	Indicates a RX packet FCS error.
0x111	RX_PKT_DATA_63_32	[31:0]	32'h0	RO	Specifies the 64-bit segment for RX packet data.
0x112	RX_PKT_DATA_31_0	[31:0]	32'h0	RO	
0x114	RX_PTP_ITS_95_64	[31:0]	32'h0	RO	Specifies the 96-bit RX PTP ingress timestamp.
0x115	RX_PTP_ITS_63_32	[31:0]	32'h0	RO	
0x116	RX_PTP_ITS_31_0	[31:0]	32'h0	RO	

4. Design Example: Single IP Core Instantiation with Auto-Negotiation and Link Training

This design example describes necessary steps to enable auto-negotiation and link training (AN/LT) in your design example.

When you enable auto-negotiation and link training parameter in the IP and generate a design example, the design example instantiates two separate IPs, F-Tile Ethernet Intel FPGA Hard IP and the F-Tile Auto-Negotiation and Link Training for Ethernet. You must connect required signals at the top level of your testbench. To successfully simulate your design, you must specify appropriate tile placement assignments in the `.qsf` file. The tile placement assignment information is unique to each device's OPN.

The following IP parameter settings were used to generate this design example:

Table 10. F-Tile Ethernet Intel FPGA Hard IP Parameter Settings

Table specifies parameter settings used to generate this design example.

Selected IP Parameter Settings	Value
General Options	
PMA type	FGT
Ethernet mode	100GE-4
Client interface	MAC segmented
FEC mode	IEEE 802.3 RS(528,514) (CL91)
PMA reference frequency	156.25 MHz
System PLL frequency	805.6640625 MHz

Table 11. F-Tile Auto-Negotiation and Link Training for Ethernet Parameter Settings

Table specifies parameter settings used to generate this design example.

Selected IP Parameter Settings	Value
Mode Selection	
Enable auto-negotiation on reset	On
Enable link training on reset	On
PMA type	FGT
Ethernet mode	100GE-4
KR or CR mode	KR mode

continued...

Selected IP Parameter Settings	Value
Number of ports	1
FEC mode	IEEE 802.3 RS(528,514)
Status clock frequency	100 MHz

For more information about steps of how to generate a design example, refer to the *Generating the Design Example*.

Related Information

Generating Single IP Instance Design on page 4

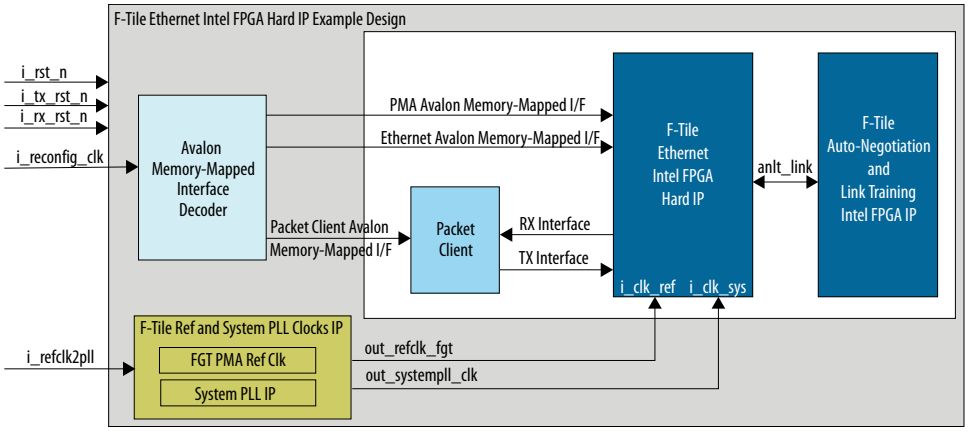
4.1. Features

- Instantiates F-Tile Auto-Negotiation and Link Training for Ethernet Intel FPGA IP
- Instantiates F-Tile Reference and System PLL Clocks Intel FPGA IP based on Ethernet configuration

4.2. Functional Description

Figure 12. F-Tile Ethernet Intel FPGA Hard IP Simulation Design Example Block Diagram with Enabled Auto-Negotiation and Link Training

The diagram depicts the FGT PMA option.



The design example includes the following components:

- **F-Tile Ethernet Intel FPGA Hard IP**: Generated IP core.
- **F-Tile Auto-Negotiation and Link Training for Ethernet Intel FPGA IP**: Generated IP core when auto-negotiation and link training is enabled.
- **F-Tile Reference and System PLL Clocks Intel FPGA IP**: Instantiated reference clock and system PLL clock IP. The F-Tile Reference and System PLL Clocks Intel FPGA IP parameter editor settings align with the **System PLL frequency** and **PMA reference frequency** parameter settings in the F-Tile Ethernet Intel FPGA Hard IP. If you generate the design example using **Generate Example Design** button in the IP parameter editor, the IP instantiates automatically. If you create your own design example, you must manually instantiate this IP and connect all I/O ports.

  For information about supported system PLL modes, refer to *F-Tile Ethernet Intel FPGA Hard IP User Guide*. For information about this IP, refer to *F-Tile Architecture and PMA and FEC Direct PHY IP User Guide*.
- **Packet Client**: Consists of a packet generator, a packet checker and a loopback client. The Packet Client generates various ROM-based traffic patterns for MAC mode and can loopback the RX and TX client side.
- **Avalon memory-mapped interface Decoder**: Decodes the Avalon memory-mapped interface address to Hardware IP Top. For base address for each of the Avalon memory-mapped interface accessed instances, refer to Register Maps on page 12.

Related Information

- Supported System PLL Modes on page 0
- F-Tile Architecture and PMA and FEC Direct PHY IP User Guide

4.3. Simulation

The testbench provides basic functionality such as the startup and waits for lock and send and receive a few packets using the ROM-based packet generator.

Important: Before the simulation, you must generate tile-related files described in Generating Tile Files on page 8 and specify the location assignment in the `.qsf` file for successful simulation.

For selected OPN, aside from updating the `.qsf` assignments for your device, you must update the tile location in the `example_testbench/basic_avl_tb_top.sv` test bench file with the F-tile location for the selected OPN.

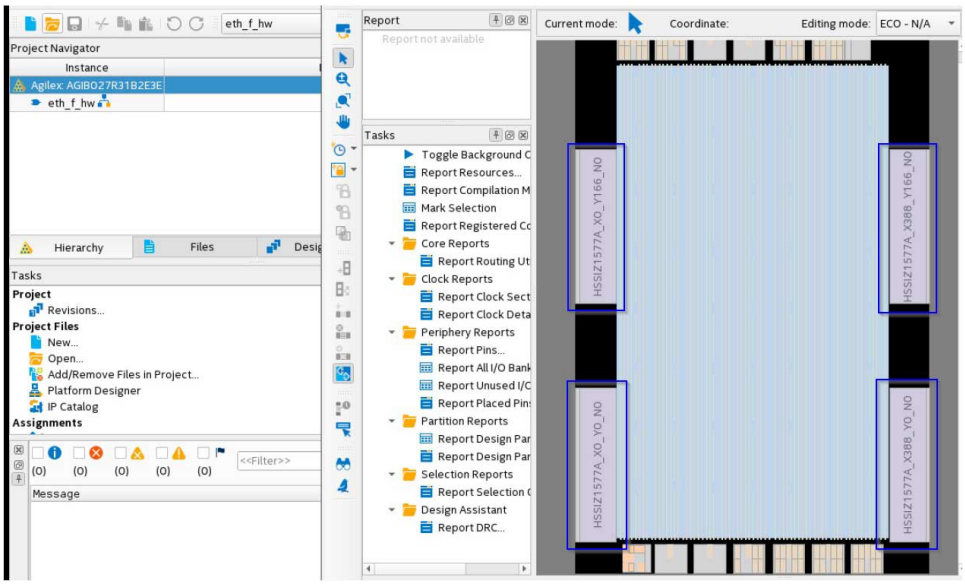
```
`define QUARTUS_TOP_LEVEL_ENTITY_INSTANCE_PATH eth_f_hw_tiles.<tile_location>
```

Then, you must also update the `.qsf` file to include both the F-Tile Auto-Negotiation and Link Training for Ethernet Intel FPGA IP and the F-Tile Ethernet Intel FPGA Hard IP, instance to the same tile location as shown below:

```
set_instance_assigment -name IP_TILE_ASSIGNMENT <tile_location> -to <Ethernet
IP INSTANCE>
set_instance_assigment -name IP_TILE_ASSIGNMENT <tile_location> -to <ANLT IP
INSTANCE>
```

To find the tile location, in the Intel Quartus Prime software window, select **Tools ➤ Chip Planner** and use the appropriate tile location as shown in the figure.

Figure 13. Chip Planner: Tile Location for the Selected OPN



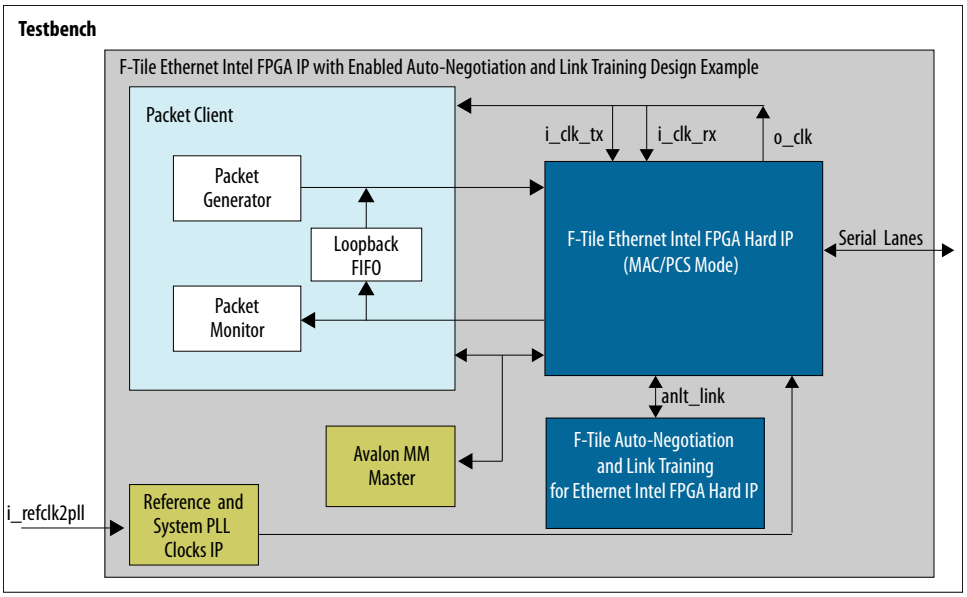
For example, if you selected the AGIA027R29A1E2VR0 ordering part number (OPN), the `.qsf` assignments are defined in QSF Assignments on page 38. The following example depicts the selection of the Z1577A_X0_Y0_N0 tile location in the AGIA027R29A1E2VR0 ordering part number (OPN):

```
`define QUARTUS_TOP_LEVEL_ENTITY_INSTANCE_PATH eth_f_hw_tiles.z1577a_x0_y0_n0
```

```
set_instance_assigment -name IP_TILE_ASSIGNMENT Z1577A_X0_Y0_N0 -to
IP_INST[0].hw_ip_top|dut
set_instance_assignment -name IP_TILE_ASSIGNMENT Z1577A_X0_Y0_N0 -to kr_dut
```

Important: If you use a different OPN, you must update your F-tile location for the selected OPN.

Figure 14. F-Tile Ethernet Intel FPGA Hard IP Simulation Design Example Block Diagram with Enabled Auto-Negotiation and Link Training



The following steps show the simulation testbench flow:

1. Assert global resets (`i_rst_n` and `i_reconfig_rst`) to reset the F-Tile Ethernet Intel FPGA Hard IP and F-Tile Auto-Negotiation and Link Training for Ethernet Intel FPGA IP.
2. Wait until configuration settings load.
3. Wait until resets acknowledgment. The `o_rst_ack_n` signal goes low.
4. Deasserts the global resets, `i_rst_n` and `i_reconfig_rst`.
5. Reconfigure PLL for an auto-negotiation frequency by setting **set_base_freq** for the selected Ethernet variant.
6. Wait until auto-negotiation is complete. The data mode begins.
7. Reconfigure PLL for link training frequency by setting **set_base_freq** for the selected Ethernet variant.
8. Wait until link training is complete.
9. Wait until `o_tx_lanes_stable` bit is set to `1`, indicating TX path is ready.
10. Wait until `o_rx_pcs_ready` bit is set to `1`, indicating RX path is ready.
11. Instruct packet client to transmit data. Write `hw_pc_ctrl[0]=1'b1` to start the packet generator.
12. Read TX packet data information from 0x20 - 0x34 registers. Read register in sequencial order.
13. Read RX packet data information from 0x38 - 0x4C registers. Read register in sequencial order.
14. Compare the counters to ensure 16 packets were sent and received.

15. Instruct packet client to stop data transmission. Write `hw_pc_ctrl[2:0]=3'b100` to stop the packet generator. Clear counters.
16. Perform Avalon memory-mapped interface test. Write and read Ethernet IP registers.
    - 0x104: Scratch register
    - 0x108: IP soft reset register
    - 0x214: TX MAX source address register [31:0]
    - 0x218: TX MAX source address register [47:32]
    - 0x21C: RX MAX frame size register
17. Perform Avalon memory-mapped interface 2 test to read and write operation transceiver registers.

The following sample output illustrates a successful simulation test run.

```
---TX reset sequence completed -----
---RX reset sequence completed -----
---Starting Data mode after completing AN ----
---IP_INST[ 0] Test    0;   ---Total     16 packets to send-----
------IP_INST[ 0] Start pkt gen TX-----
------Checking Packet TX/RX result-----
------------  16 packets Sent;     0 packets Received--------
------ALL   16  packets Sent out---
------------  16 packets Sent;    16 packets Received--------
------ALL   16  packets Received---
------TX/RX packet check OK---
****Starting AVMM Read/Write****
====>MATCH!     ReaddataValid = 1 Readdata = abcdef01 Expected_Readdata =
abcdef01
====>MATCH!     ReaddataValid = 1 Readdata = 00000007 Expected_Readdata =
00000007
====>MATCH!     ReaddataValid = 1 Readdata = 00000000 Expected_Readdata =
00000000
====>MATCH!     ReaddataValid = 1 Readdata = 9d228c3a Expected_Readdata =
9d228c3a
====>MATCH!     ReaddataValid = 1 Readdata = 4338b586 Expected_Readdata =
4338b586
====>MATCH!     ReaddataValid = 1 Readdata = deadc0de Expected_Readdata =
deadc0de
====>MATCH!     ReaddataValid = 1 Readdata = deadc0de Expected_Readdata =
deadc0de
====>MATCH!     ReaddataValid = 1 Readdata = 00000000 Expected_Readdata =
00000000
====>MATCH!     ReaddataValid = 1 Readdata = 22334455 Expected_Readdata =
22334455
====>MATCH!     ReaddataValid = 1 Readdata = 00000011 Expected_Readdata =
00000011
====>MATCH!     ReaddataValid = 1 Readdata = 000005ee Expected_Readdata =
000005ee
====>MATCH!     ReaddataValid = 1 Readdata = 01234567 Expected_Readdata =
01234567
====>MATCH!     ReaddataValid = 1 Readdata = 000089ab Expected_Readdata =
000089ab
743830ns  Try to access AVMM2 begin...
743830ns  write 0x00000065 to xcvr  0 address 0x103c004
744795ns  Try to access AVMM2 end...
744890ns  read from address 0x103c004
====>MATCH!     ReaddataValid = 1 Readdata = 00000065 Expected_Readdata =
00000065

...

758740ns  Try to access AVMM2 end...
```

```
758840ns  Try to access AVMM2 begin...
758840ns  write 0x0000006c to xcvr  7 address 0x103c00b
759825ns  Try to access AVMM2 end...
759920ns  read from address 0x103c00b
====>MATCH!     ReaddataValid = 1 Readdata = 0000006c Expected_Readdata =
0000006c
760900ns  Try to access AVMM2 end...
**** AVMM Read/Write Operation Completed for IP_INST[ 0]****
** Testbench complete
**
```

Note: The simulation completion may take a longer time. To confirm the simulation is progressing successfully, verify the intermediate outputs from the System Console such as bringing the base and AN/LT IP out of resets, IP resetting sequence, Auto-negotiation and link training auto-connection completion, and others.

4.4. QSF Assignments

For successful logic generation/compilation and simulation, you must specify lane location assignment and tile location assignment in the `.qsf` file in your design. The exact `.qsf` assignments are OPN-specific and must be updated for each OPN. For information about updating tile location, refer to Simulation on page 34.

This section provides an example of `.qsf` assignments for FGT and FHT PMA types for simulation.

Table 12. FGT `.qsf` Assignments

The following `.qsf` assignments are specific to the `AGIA027R29A1E2VR0` OPN. If you use a device with a different OPN, you must update these location assignments to match pin out of the corresponding OPN.

Ethernet Mode	QSF Assignments

All -1 modes	`set_location_assignment PIN_FP74 -to i_refclk2pll`
	`set_location_assignment PIN_GG78 -to o_tx_serial[0]`
	`set_location_assignment PIN_GK83 -to i_rx_serial[0]`

All -2 modes	`set_location_assignment PIN_FP74 -to i_refclk2pll`
	`set_location_assignment PIN_GG78 -to o_tx_serial[0]`
	`set_location_assignment PIN_GW80 -to o_tx_serial[1]`
	`set_location_assignment PIN_GK83 -to i_rx_serial[0]`
	`set_location_assignment PIN_HB83 -to i_rx_serial[1]`

All -4 modes	`set_location_assignment PIN_FP74 -to i_refclk2pll`
	`set_location_assignment PIN_GG78 -to o_tx_serial[0]`
	`set_location_assignment PIN_GW80 -to o_tx_serial[1]`
	`set_location_assignment PIN_HB77 -to o_tx_serial[2]`
	`set_location_assignment PIN_HL80 -to o_tx_serial[3]`
	`set_location_assignment PIN_GK83 -to i_rx_serial[0]`
	`set_location_assignment PIN_HB83 -to i_rx_serial[1]`
	`set_location_assignment PIN_HP83 -to i_rx_serial[2]`
	`set_location_assignment PIN_JD80 -to i_rx_serial[3]`

All -8 modes	`set_location_assignment PIN_FP74 -to i_refclk2pll`
	`set_location_assignment PIN_GG78 -to o_tx_serial[0]`
	`set_location_assignment PIN_GW80 -to o_tx_serial[1]`
	`set_location_assignment PIN_HB77 -to o_tx_serial[2]`
	`set_location_assignment PIN_HL80 -to o_tx_serial[3]`
	`set_location_assignment PIN_HP77 -to o_tx_serial[4]`
	`set_location_assignment PIN_JG77 -to o_tx_serial[5]`
	`set_location_assignment PIN_JT74 -to o_tx_serial[6]`
	`set_location_assignment PIN_JW77 -to o_tx_serial[7]`
	`set_location_assignment PIN_GK83 -to i_rx_serial[0]`
	`set_location_assignment PIN_HB83 -to i_rx_serial[1]`
	`set_location_assignment PIN_HP83 -to i_rx_serial[2]`
	`set_location_assignment PIN_JD80 -to i_rx_serial[3]`
	`set_location_assignment PIN_JG83 -to i_rx_serial[4]`

continued...

Ethernet Mode	QSF Assignments

	`set_location_assignment PIN_JT80 -to i_rx_serial[5]`
	`set_location_assignment PIN_JW83 -to i_rx_serial[6]`
	`set_location_assignment PIN_KH80 -to i_rx_serial[7]`

Table 13. FHT `.qsf` Assignments

The following `.qsf` assignments are specific to the AGIA027R29A1E2VR0 OPN. If you use a device with a different OPN, you must update these location assignments to match pin out of the corresponding OPN.

Ethernet Mode	QSF Assignments

All -1 modes	`set_location_assignment PIN_EC76 -to i_bk_refclk2pll`
	`set_location_assignment PIN_HY68 -to i_refclk2pll`
	`set_location_assignment PIN_DU82 -to o_tx_serial[0]`
	`set_location_assignment PIN_DL78 -to i_rx_serial[0]`

All -2 modes	`set_location_assignment PIN_EC76 -to i_bk_refclk2pll`
	`set_location_assignment PIN_HY68 -to i_refclk2pll`
	`set_location_assignment PIN_DU82 -to o_tx_serial[0]`
	`set_location_assignment PIN_EK82 -to o_tx_serial[1]`
	`set_location_assignment PIN_DL78 -to i_rx_serial[0]`
	`set_location_assignment PIN_EC78 -to i_rx_serial[1]`

All -4 modes	`set_location_assignment PIN_EC76 -to i_bk_refclk2pll`
	`set_location_assignment PIN_HY68 -to i_refclk2pll`
	`set_location_assignment PIN_DU82 -to o_tx_serial[0]`
	`set_location_assignment PIN_EK82 -to o_tx_serial[1]`
	`set_location_assignment PIN_FB82 -to o_tx_serial[2]`
	`set_location_assignment PIN_FP82 -to o_tx_serial[3]`
	`set_location_assignment PIN_DL78 -to i_rx_serial[0]`
	`set_location_assignment PIN_EC78 -to i_rx_serial[1]`
	`set_location_assignment PIN_ET78 -to i_rx_serial[2]`
	`set_location_assignment PIN_FH78 -to i_rx_serial[3]`

Related Information

Simulation on page 34

4.5. Hardware Design Example

Follow these steps to test Ethernet-based design examples with enabled auto-negotiation and link training in hardware:

1. Generate design example as described in Generating the Design on page 4.
2. Modify the `.qsf` settings:
   - Set device to match the appropriate ordering part number (OPN) for your design.
   - Update the pinout to match the board and the design function.
   - Assign the appropriate VID settings in your `.qsf` file to match your board.
3. Generate the `.sof` file.
4. Update board clock settings. The default value for the PHY reference clock is 156.25 MHz. The default value for the reconfiguration clock is 100 MHz.
5. Insert appropriate electrical loopback plug into the Ethernet port.
6. Program the design.
7. Open **Tools ➤ System Debugging Tools ➤ System Console**.
8. Navigate to the hardware directory `<design_example>/hardware_test_design/hwtest` directory.

9. Type `source main_<variant_type>.tcl`.
10. Type `set_jtag<number_of appropriate_JTAG_master>`
11. Perform this step if external loopback module is connected. Skip this step if the design connects to link partner.
    a. Type command to read the `seq cfg` register:
    ```
    reg_read 0x101002C0
    ```
    b. Type command to set the ingore nonce value to 1:
    ```
    reg_write 0x10100300 0x737d0281
    ```
    c. Type command to restart the AN sequencer:
    ```
    reg_write 0x101002c0 0x00002003
    ```
    d. Type command to read the debug status:
    ```
    reg_read 0x101003c0
    ```
    The link is up if the command returns value `1f0`.
12. Type the following command to check the PHY status:
    ```
    chkphy_status
    ```
13. Type the command to start and stop the packet generator. The function sends 16 packets.
    ```
    start_pkt_gen
    stop_pkt_gen
    ```
14. Type the command to check the MAC status:
    ```
    chkmac_status
    ```

5. Design Example: Multiple IP Core Instantiation

The multiple IP core design example demonstrates the ability to instantiate same F-Tile Ethernet Intel FPGA Hard IP multiple times in your design.

The number of instantiated IP instances depends on the Ethernet mode. For more details, refer to Table 15 on page 42. To generate the design example, you must first set the parameter values for the IP core variation you intend to generate in your end product. Generating the design example creates multiple copies of your IP core; the testbench design example uses this variation as the DUT. If your parameter values for the DUT don't match the parameter values in your end product, the design example you generate does not exercise the IP core variation you intend.

The multi IP core design example supports PTP feature for FGT PMA type.

The following IP parameter settings were used to generate this design example:

Table 14. IP Parameters for 25G Ethernet Mode with 1 Lane Design Example

Table specifies parameter settings used to generate this design example.

Selected IP Parameter Settings	Value
IP Tab: General Options	
PMA type	FGT
Ethernet mode	25GE-1
Client interface	MAC segmented
FEC mode	IEEE 802.3 RS(528,514) (CL91)
PMA reference frequency	156.25
System PLL frequency	805.664062
Example Design Tab: Available Example Designs	
Select Design	Multi instance of IP core

For more information about steps on how to generate a design example, refer to the *Generating the Design Example*.

Related Information

- Generating Single IP Instance Design on page 4
- Available Number of Multiple IP Instances per Ethernet Mode table on page 42

5.1. Features

- Supports multiple instantiations of the same F-Tile Ethernet Intel FPGA Hard IP. For more information, refer to the *Available Number of Multiple IP Instances per Ethernet Mode* table.
- Each instance supports 10G, 25G, 40G, 50G, 100G, 200G, and 400G Ethernet rates
- Each instance supports Avalon streaming interface for 10G, 25G, 40G, 50G, and 100G Ethernet rates with synchronized or asynchronous adapter
- Each instance supports MAC segmented interface
- Instantiates F-Tile Reference and System PLL Clocks Intel FPGA IP based on Ethernet configuration

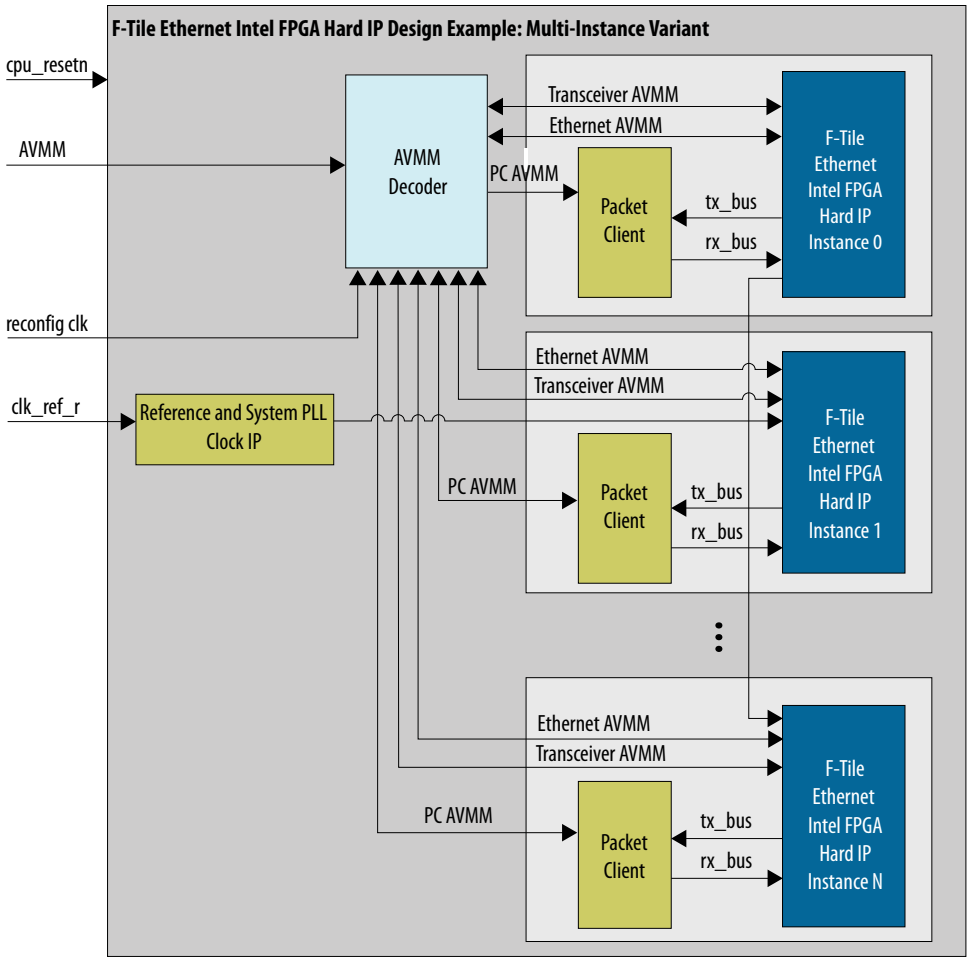
When you enable the multi instance IP option, the design example generates multiple instances of the same IP. The table specifies the number of instantiated F-Tile Ethernet Intel FPGA Hard IPs per Ethernet mode.

Table 15. Available Number of Multiple IP Instances per Ethernet Mode

Ethernet Mode	Modulation	PMA Type	Number of Multiple IP Instances
10GE-1	NRZ	FGT	16
25GE-1	NRZ	FGT	16
25GE-1	NRZ	FHT	4
40GE-4	NRZ	FGT	4
50GE-2	NRZ	FGT	8
50GE-2	NRZ	FHT	2
50GE-1	PAM4	FGT	8
50GE-1	PAM4	FHT	4
100GE-4	NRZ	FGT	4
100GE-4	NRZ	FHT	1
100GE-2	NRZ	FGT	4
100GE-2	NRZ	FHT	2
100GE-1	PAM4	FHT	4
200GE-8	NRZ	FGT	2
200GE-4	PAM4	FGT	2
200GE-4	PAM4	FHT	1
200GE-2	PAM4	FHT	2
400GE-8	PAM4	FGT	1
400GE-4	PAM4	FHT	1

5.2. Functional Description

Figure 15. Simulation Design Example Block for Multiple F-Tile Ethernet Intel FPGA Hard IPs



Legend:
AVMM: Avalon memory-mapped interface
N: Number of F-Tile Ethernet Intel FPGA IP Instances

The F-Tile Ethernet Intel FPGA Hard IP design example includes the following components:

- **F-Tile Ethernet Intel FPGA Hard IP**: Generated IP core.
- **F-Tile Reference and System PLL Clocks Intel FPGA IP**: Instantiated reference clock and system PLL clock IP. The F-Tile Reference and System PLL Clocks Intel FPGA IP parameter editor settings align with the **System PLL frequency** and **PMA reference frequency** parameter settings in the F-Tile Ethernet Intel FPGA Hard IP. If you generate the design example using **Generate Example Design** button in the IP parameter editor, the IP instantiates automatically. If you create your own design example, you must manually instantiate this IP and connect all I/O ports.

  For information about supported system PLL modes, refer to *F-Tile Ethernet Intel FPGA Hard IP User Guide*. For information about this IP, refer to *F-Tile Architecture and PMA and FEC Direct PHY IP User Guide*.
- **Packet Client**: Consists of a packet generator, a packet checker and a loopback client. The Packet Client generates various ROM-based traffic patterns for MAC mode and can loopback the RX and TX client side.
- **Avalon memory-mapped interface Decoder**: Decodes the Avalon memory-mapped interface address to Hardware IP Top and PTP blocks if PTP is enabled. For base address for each of the Avalon memory-mapped interface accessed instances, refer to Register Maps on page 12.

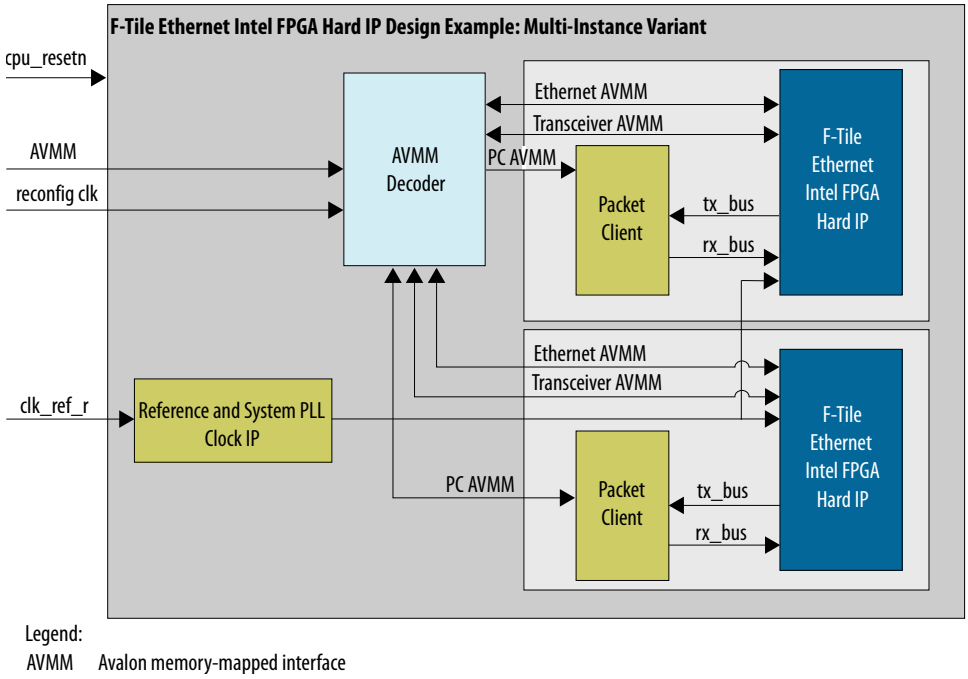
Related Information

F-Tile Architecture and PMA and FEC Direct PHY IP User Guide

5.3. Simulation

The testbench provides basic functionality such as the startup and wait for lock, sending and receiving of a few packets per each instantiated IP using the ROM-based packet generator.

Figure 16. Simulation Design Example Block Diagram with 2 instantiated F-Tile Ethernet Intel FPGA Hard IPs



The following steps show the simulation testbench flow:

1. Assert global reset (`i_rst_n`) to reset each F-Tile Ethernet Intel FPGA Hard IP instance (IP instance).
2. Wait until resets acknowledgment from all IP instances. The `o_rst_ack_n` signals go low.
3. Deasserts the global reset.
4. Wait until `o_tx_lanes_stable` bit is set to 1, indicating TX path is ready.
5. Wait until `o_rx_pcs_ready` bit is set to 1, indicating RX path is ready.
6. Repeat steps 4 and 5 to complete the reset sequence for all IP instances.
7. Instruct packet client to transmit data. Write `hw_pc_ctrl[0]=1'b1` to start the packet generator.
8. Read TX packet data information from 0x20 - 0x34 registers. Read registers in a sequential order.
9. Read RX packet data information from 0x38 - 0x4C registers. Read registers in a sequential order.
10. Compare the counters to ensure 16 packets were sent and received.
11. Instruct packet client to stop data transmission. Write `hw_pc_ctrl[2:0]=3'b100` to stop the packet generator. Clear counters.

12. Repeat steps 7 - 11 to simulate packet transfer for each of the instantiated IPs.
13. Perform Avalon memory-mapped interface test. Write and read Ethernet IP registers.
14. Perform Avalon memory-mapped interface 2 test to read and write operation transceiver registers.
15. Repeat steps 13 and 14 for each instantiated IP in a sequential order.

The following sample output illustrates a successful simulation test run.

```
---SRC IP sequence started -----
---SRC IP sequence TX completed -----
---SRC IP sequence RX completed -----
---Test    0;   ---Total     16 packets to send-----
------Start pkt gen TX-----
------Checking Packet TX/RX result-----
------------  16 packets Sent;     0 packets Received--------
------ALL   16  packets Sent out---
------------  16 packets Sent;    16 packets Received--------
------ALL   16  packets Received---
------TX/RX packet check OK---
****Starting AVMM Read/Write****
====>MATCH!     ReaddataValid = 1 Readdata = abcdef01 Expected_Readdata =
abcdef01
====>MATCH!     ReaddataValid = 1 Readdata = 00000007 Expected_Readdata =
00000007
====>MATCH!     ReaddataValid = 1 Readdata = 00000000 Expected_Readdata =
00000000
====>MATCH!     ReaddataValid = 1 Readdata = 9d228c3a Expected_Readdata =
9d228c3a
====>MATCH!     ReaddataValid = 1 Readdata = 4338b586 Expected_Readdata =
4338b586
====>MATCH!     ReaddataValid = 1 Readdata = deadc0de Expected_Readdata =
deadc0de
====>MATCH!     ReaddataValid = 1 Readdata = deadc0de Expected_Readdata =
deadc0de
====>MATCH!     ReaddataValid = 1 Readdata = 00000000 Expected_Readdata =
00000000
====>MATCH!     ReaddataValid = 1 Readdata = 22334455 Expected_Readdata =
22334455
====>MATCH!     ReaddataValid = 1 Readdata = 00000011 Expected_Readdata =
00000011
====>MATCH!     ReaddataValid = 1 Readdata = 000005ee Expected_Readdata =
000005ee
====>MATCH!     ReaddataValid = 1 Readdata = 01234567 Expected_Readdata =
01234567
====>MATCH!     ReaddataValid = 1 Readdata = 000089ab Expected_Readdata =
000089ab
743830ns  Try to access AVMM2 begin...
743830ns  write 0x00000065 to xcvr  0 address 0x103c004
744795ns  Try to access AVMM2 end...
744890ns  read from address 0x103c004
====>MATCH!     ReaddataValid = 1 Readdata = 00000065 Expected_Readdata =
00000065

...

758740ns  Try to access AVMM2 end...
758840ns  Try to access AVMM2 begin...
758840ns  write 0x0000006c to xcvr  7 address 0x103c00b
759825ns  Try to access AVMM2 end...
759920ns  read from address 0x103c00b
====>MATCH!     ReaddataValid = 1 Readdata = 0000006c Expected_Readdata =
0000006c
760900ns  Try to access AVMM2 end...
```

```
**** AVMM Read/Write Operation Completed ****
** Testbench complete
**
```

6. F-Tile Ethernet Intel FPGA Hard IP Archives

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

Intel Quartus Prime Version	IP Core Version	User Guide
21.2	2.0.0	F-Tile Ethernet Intel FPGA Hard IP Design Example User Guide

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

Document Version	Intel Quartus Prime Version	IP Version	Changes
2021.10.11	21.3	3.0.0	• Added pin assignment requirement for AN/LT designs in *Generating Tile Files*. • Updated the list of supported simulators in *Simulating the Design Example Testbench*. • Added new topics: — *Fast Sim Model* — *Testing the Hardware Design Example* — *Register Maps* — *Simulation Testbench Flow for PCS, OTN, and FlexE Modes* • Updated register descriptions in *Packet Client Registers*. • Updated PTP-related *Registers* section. Address offset is specified as a byte address. • Added new design examples: *Single IP Core Instantiation with Auto-Negotiation and Link Training*
2021.07.01	21.2	2.0.0	Initial release.