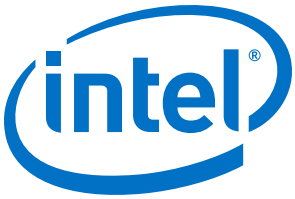


Intel® FPGA P-Tile Avalon® Streaming IP for PCI Express* Design Example User Guide

Updated for Intel® Quartus® Prime Design Suite: **20.3**

IP Version: **3.1.0**

UG-20234 | 2020.10.05
Latest document on the web: **PDF** | **HTML**

Contents

1. Design Example Description

1.1. Functional Description for the Programmed Input/Output (PIO) Design Example

The PIO design example performs memory transfers from a host processor to a target device. In this example, the host processor requests single-dword MemRd and MemWr TLPs.

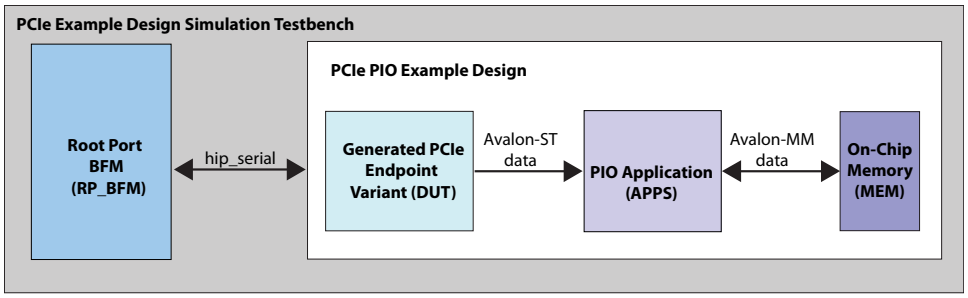
The PIO design example automatically creates the files necessary to simulate and compile in the Intel® Quartus® Prime software. The design example covers a wide range of parameters. However, it does not cover all possible parameterizations of the P-Tile Hard IP for PCIe.

This design example includes the following components:

- The generated P-Tile Avalon-ST Hard IP Endpoint variant (DUT) with the parameters you specified. This component drives TLP data received to the PIO application.
- The PIO Application (APPS) component, which performs the necessary translation between the PCI Express TLPs and simple Avalon-MM writes and reads to the on-chip memory.
- An on-chip memory (MEM) component.
- Reset Release IP: This IP holds the control circuit in reset until the device has fully entered user mode. The FPGA asserts the `INIT_DONE` output to signal that the device is in user mode. The Reset Release IP generates an inverted version of the internal `INIT_DONE` signal to create the `nINIT_DONE` output that you can use for your design.The `nINIT_DONE` signal is high until the entire device enters user mode. After `nINIT_DONE` asserts (low), all logic is in user mode and operates normally. You can use the `nINIT_DONE` signal in one of the following ways:
  - To gate an external or internal reset.
  - To gate the reset input to the transceiver and I/O PLLs.
  - To gate the write enable of design blocks such as embedded memory blocks, state machine, and shift registers.
  - To synchronously drive register reset input ports in your design.

The simulation testbench instantiates the PIO design example and a Root Port BFM to interface with the target Endpoint.

Figure 1. Block Diagram for the Platform Designer PIO Design Example Simulation Testbench



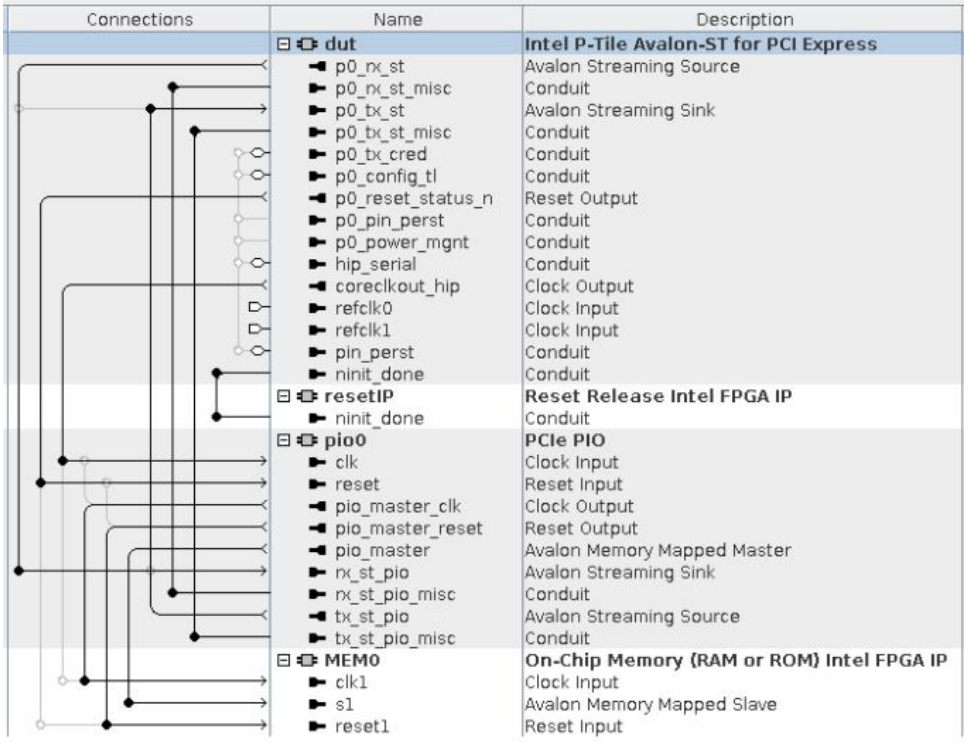
The test program writes to and reads back data from the same location in the on-chip memory. It compares the data read to the expected result. The test reports, "Simulation stopped due to successful completion" if no errors occur.

The P-Tile Avalon®-ST design example supports the following configurations:

- Gen4 x16 Endpoint
- Gen3 x16 Endpoint
- Gen4 x8 Endpoint
- Gen3 x8 Endpoint

Figure 2. Platform Designer System Contents for P-Tile Avalon-ST PCI Express PIO Design Example

The Platform Designer generates this design for up to Gen4 x16 variants.



1.2. Functional Description for the Single Root I/O Virtualization (SR-IOV) Design Example

The SR-IOV design example performs memory transfers from a host processor to a target device. It supports up to two PFs and 32 VFs per PF.

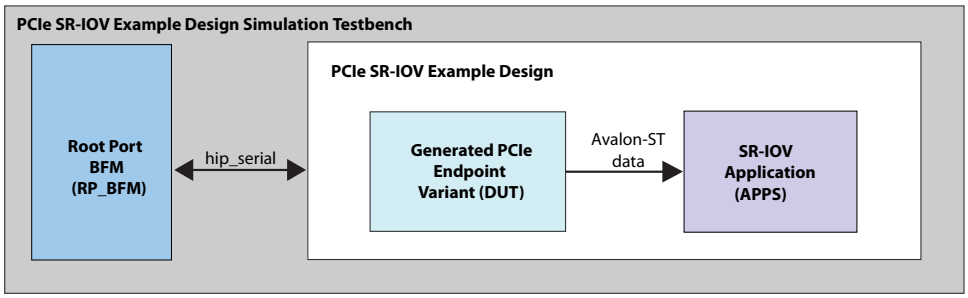
The SR-IOV design example automatically creates the files necessary to simulate and compile in the Intel Quartus Prime software. You can download the compiled design to an Intel Stratix® 10 DX Development Kit.

This design example includes the following components:

- The generated P-Tile Avalon Streaming (Avalon-ST) IP Endpoint variant (DUT) with the parameters you specified. This component drives the received TLP data to the SR-IOV application.
- The SR-IOV Application (APPS) component, which performs the necessary translation between the PCI Express TLPs and simple Avalon-ST writes and reads to the on-chip memory.
- A Reset Release IP.

The simulation testbench instantiates the SR-IOV design example and a Root Port BFM to interface with the target Endpoint.

Figure 3. Block Diagram for the Platform Designer SR-IOV Design Example Simulation Testbench

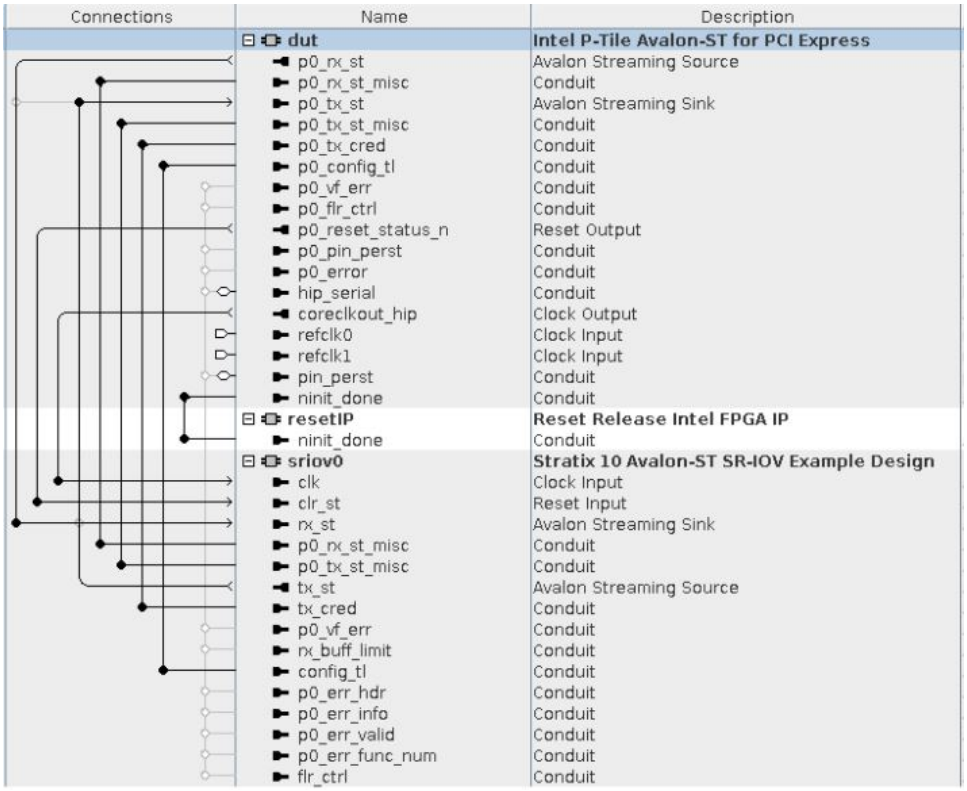


The test program writes to and reads back data from the same location in the on-chip memory across 2 PFs and 32 VFs per PF. It compares the data read to the expected result. The test reports, "Simulation stopped due to successful completion" if no errors occur.

The SR-IOV design example supports the following configurations:

- Gen4 x16 Endpoint
- Gen3 x16 Endpoint

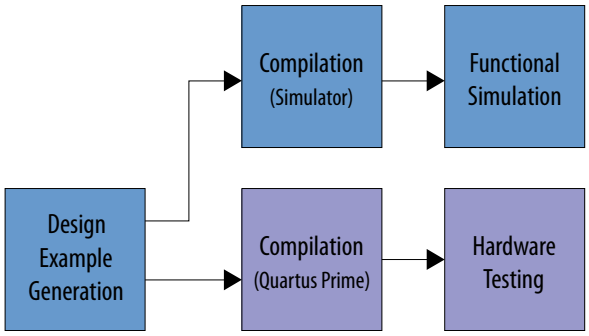
Figure 4. Platform Designer System Contents for P-Tile Avalon-ST with SR-IOV for PCI Express Design Example



2. Quick Start Guide

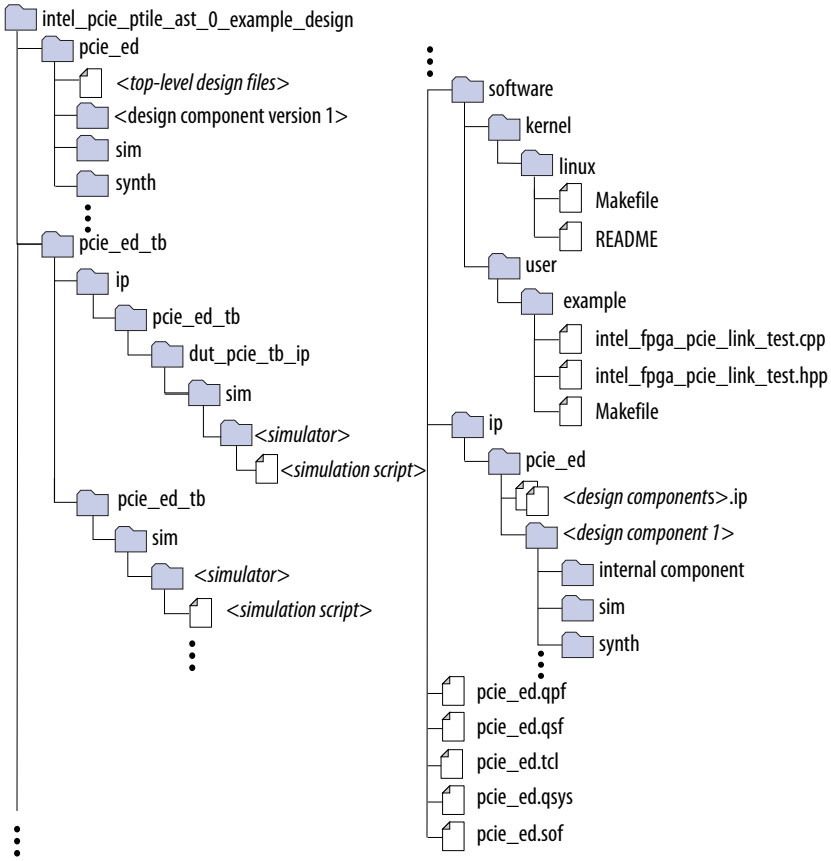
Using Intel Quartus Prime software, you can generate a programmed I/O (PIO) design example for the Intel FPGA P-Tile Avalon-ST Hard IP for PCI Express* IP core. The generated design example reflects the parameters that you specify. The PIO example transfers data from a host processor to a target device. It is appropriate for low-bandwidth applications. This design example automatically creates the files necessary to simulate and compile in the Intel Quartus Prime software. You can download the compiled design to your FPGA Development Board. To download to custom hardware, update the Intel Quartus Prime Settings File (`.qsf`) with the correct pin assignments .

Figure 5. Development Steps for the Design Example



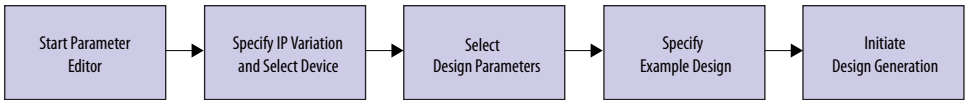
2.1. Directory Structure

Figure 6. Directory Structure for the Generated Design Example



2.2. Generating the Design Example

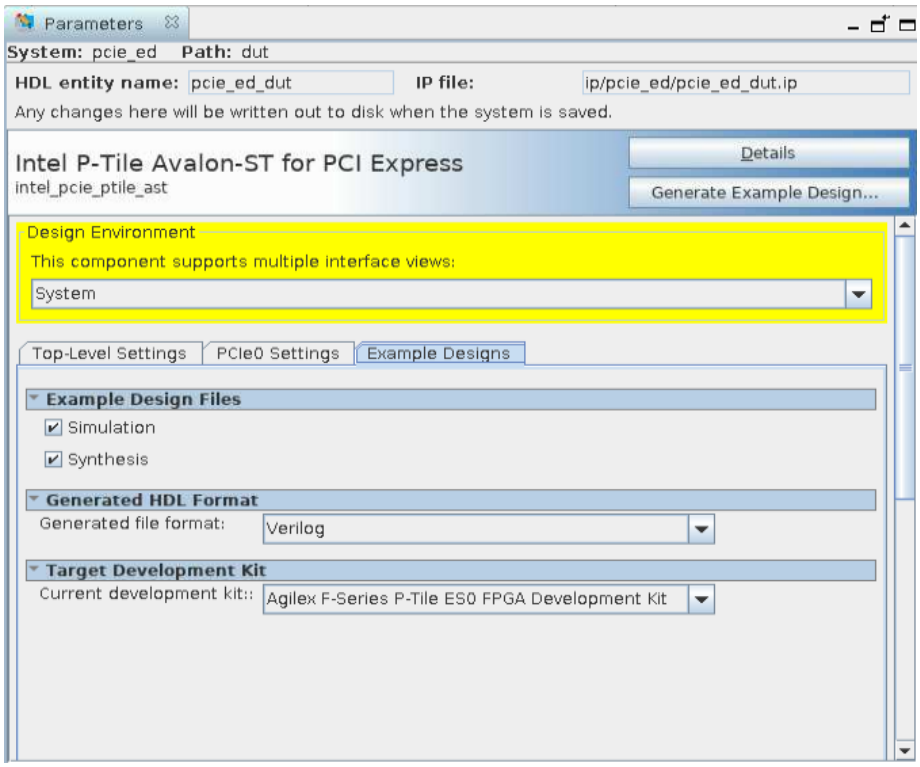
Figure 7. Procedure



1. In the Intel Quartus Prime Pro Edition software, create a new project (**File ➤ New Project Wizard**).
2. Specify the **Directory, Name**, and **Top-Level Entity.**
3. For **Project Type**, accept the default value, **Empty project**. Click **Next**.
4. For **Add Files** click **Next**.
5. For **Family, Device & Board Settings** under **Family**, select **Intel Agilex™** or **Intel Stratix 10**.
6. If you selected **Intel Stratix 10** in the last step, select **Stratix 10 DX** in the **Device** pull-down menu.
7. Select the **Target Device** for your design.
8. Click **Finish**.
9. In the IP Catalog locate and add the **Intel P-Tile Avalon-ST Hard IP for PCI Express**.
10. In the **New IP Variant** dialog box, specify a name for your IP. Click **Create**.
11. On the **Top-Level Settings** and **PCIe* Settings** tabs, specify the parameters for your IP variation. If you are using the SR-IOV design example, do the following steps to enable SR-IOV:
    a. On the **PCIe0 Device** tab under the **PCIe0 PCI Express / PCI Capabilities** tab, check the box **Enable multiple physical functions**.
    b. On the **PCIe0 Multifunction and SR-IOV System Settings** tab, check the box **Enable SR-IOV support** and specify the number of PFs and VFs.
    c. On the **PCIe0 MSI-X** tab under the **PCIe0 PCI Express / PCI Capabilities** tab, enable the MSI-X feature as required.
    d. On the **PCIe0 Base Address Registers** tab, enable BAR0 for both PF and VF.
12. On the **Example Designs** tab, make the following selections:
    a. For **Example Design Files**, turn on the **Simulation** and **Synthesis** options. If you do not need these simulation or synthesis files, leaving the corresponding option(s) turned off significantly reduces the example design generation time.
    b. For **Generated HDL Format**, only Verilog is available in the current release.
    c. For **Target Development Kit**, select either the **Intel Stratix 10 DX P-Tile ES1 FPGA Development Kit** or the **Intel Agilex F-Series P-Tile ES0 FPGA Development Kit**.
13. Select **Generate Example Design** to create a design example that you can simulate and download to hardware. If you select one of the P-Tile development boards, the device on that board overwrites the device previously selected in the Intel Quartus Prime project if the devices are different. When the prompt asks you

to specify the directory for your example design, you can accept the default directory, `./intel_pcie_ptile_ast_0_example_design`, or choose another directory.

Figure 8. Example Designs Tab

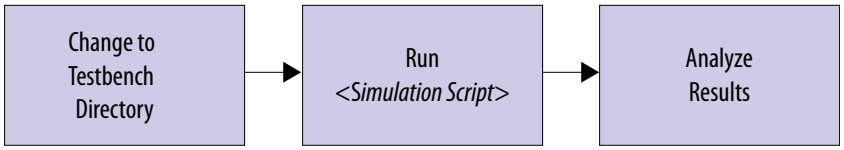


14. Click **Finish**. You may save your `.ip` file when prompted, but it is not required to be able to use the example design.
15. Open the example design project.
16. Compile the example design project to generate the `.sof` file for the complete example design. This file is what you download to a board to perform hardware verification.
17. Close your example design project.

    Note that you cannot change the PCIe pin allocations in the Intel Quartus Prime project. However, to ease PCB routing, you can take advantage of the lane reversal and polarity inversion features supported by this IP.

2.3. Simulating the Design Example

Figure 9. Procedure



1. Change to the testbench simulation directory, `<project_directory>/pcie_ed_tb/pcie_ed_tb/sim/<EDA_vendor>/simulator`.
2. Run the simulation script for the simulator of your choice. Refer to the table below.
3. Analyze the results.

Note: **P-Tile does not support parallel PIPE simulations.**

Table 1. Steps to Run Simulation

Simulator	Working Directory	Instructions
ModelSim*	`<example_design>/pcie_ed_tb/pcie_ed_tb/sim/mentor/`	1. Invoke vsim (by typing `vsim`, which brings up a console window where you can run the following commands). 2. `do msim_setup.tcl` *Note:* Alternatively, instead of doing Steps 1 and 2, you can type: `vsim -c -do msim_setup.tcl`. 3. `ld_debug` 4. `run -all` 5. A successful simulation ends with the following message, "Simulation stopped due to successful completion!"
VCS*	`<example_design>/pcie_ed_tb/pcie_ed_tb/sim/synopsys/vcs`	1. Type `sh vcs_setup.sh USER_DEFINED_COMPILE_OPTIONS="" USER_DEFINED_ELAB_OPTIONS="-xlrm\ uniq_prior_final" USER_DEFINED_SIM_OPTIONS=""` *Note:* The command above is a single-line command. 2. A successful simulation ends with the following message, "Simulation stopped due to successful completion!" *Note:* To run a simulation in interactive mode, use the following steps: (if you already generated a `simv` executable in non-interactive mode, delete the `simv` and `simv.diadir`) 1. Open the `vcs_setup.sh` file and add a debug option to the VCS command: `vcs -debug_access+r` 2. Compile the design example: `sh vcs_setup.sh USER_DEFINED_ELAB_OPTIONS="-xlrm\ uniq_prior_final" SKIP_SIM=1` 3. Start the simulation in interactive mode: `simv -gui &`

This testbench simulates up to a Gen4 x16 variant.

The simulation reports, "Simulation stopped due to successful completion" if no errors occur.

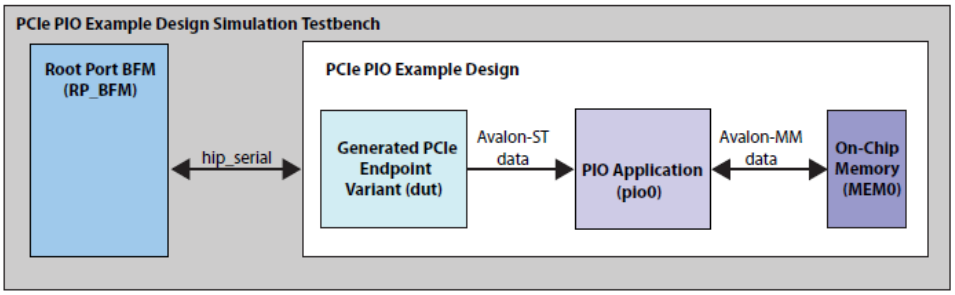
2.3.1. Testbench

The testbench uses a test driver module, `altpcietb_bfm_rp_gen4_x16.sv`, to initiate the configuration and memory transactions. At startup, the test driver module displays information from the Root Port and Endpoint Configuration Space registers, so that you can correlate to the parameters you specified using the Parameter Editor.

The example design and testbench are dynamically generated based on the configuration that you choose for the P-Tile IP for PCIe. The testbench uses the parameters that you specify in the Parameter Editor in Intel Quartus Prime.

This testbench simulates up to a ×16 PCI Express link using the serial PCI Express interface. The testbench design does allow more than one PCI Express link to be simulated at a time. The following figure presents a high level view of the PIO design example.

Figure 10. PIO Design Example Simulation Testbench



The top-level of the testbench instantiates the following main modules:

- `altpcietb_bfm_rp_gen4x16.sv` —This is the Root Port PCIe BFM.

```
//Directory path
<project_dir>/intel_pcie_ptile_ast_0_example_design/pcie_ed_tb/ip/
pcie_ed_tb/dut_pcie_tb_ip/intel_pcie_ptile_tbed_<ver>/sim
```

- `pcie_ed_dut.ip`: This is the Endpoint design with the parameters that you specify.

```
//Directory path
<project_dir>/intel_pcie_ptile_ast_0_example_design/ip/pcie_ed
```

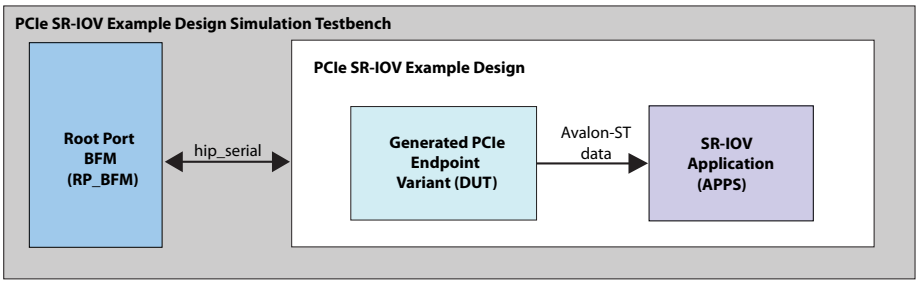
- `pcie_ed_pio0.ip`: This module is a target and initiator of transactions for the PIO design example.

```
//Directory path
<project_dir>/intel_pcie_ptile_ast_0_example_design/ip/pcie_ed
```

- `pcie_ed_sriov0.ip`: This module is a target and initiator of transactions for the SR-IOV design example.

```
//Directory path
<project_dir>/intel_pcie_ptile_ast_0_example_design/ip/pcie_ed
```

Figure 11. SR-IOV Design Example Simulation Testbench



In addition, the testbench has routines that perform the following tasks:

- Generates the reference clock for the Endpoint at the required frequency.
- Provides a PCI Express reset at start up.

For more details on the Root Port BFM, refer to the *TestBench* chapter of the *Intel FPGA P-Tile Avalon streaming IP for PCI Express User Guide*.

Related Information

Intel FPGA P-Tile Avalon streaming IP for PCI Express User Guide

2.3.1.1. Test Driver Module

The test driver module, `intel_pcie_ptile_tbed_hwtcl.v`, instantiates the top-level BFM,`altpcietb_bfm_top_rp.v`.

The top-level BFM completes the following tasks:

1. Instantiates the driver and monitor.
2. Instantiates the Root Port BFM.
3. Instantiates the serial interface.

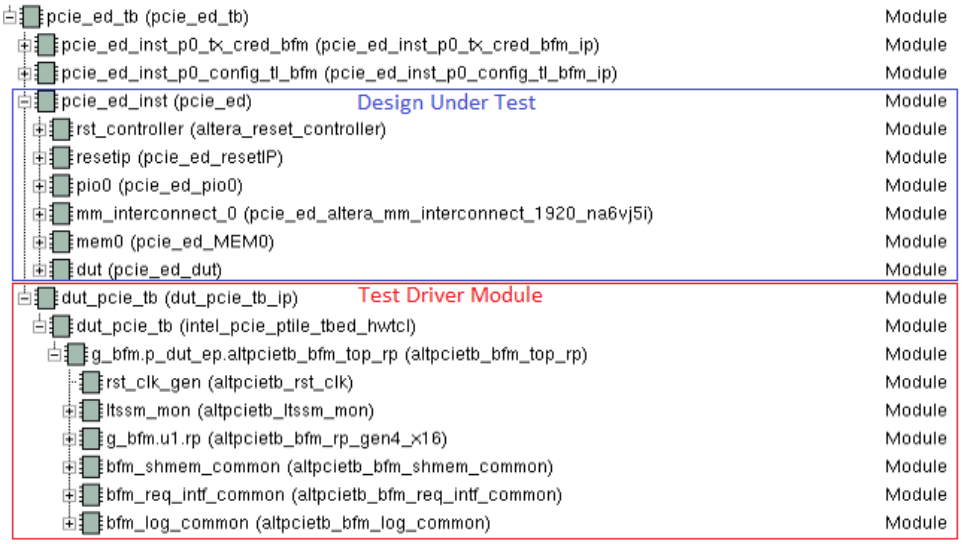
The configuration module, `altpcietb_g3bfm_configure.v`, performs the following tasks:

1. Configures and assigns the BARs.
2. Configures the Root Port and Endpoint.
3. Displays comprehensive Configuration Space, BAR, MSI, MSI-X, and AER settings.

2.3.1.2. PIO Design Example Testbench

The figure below shows the PIO design example simulation design hierarchy. The tests for the PIO design example are defined with the `apps_type_hwtcl` parameter set to 3. The tests run under this parameter value are defined in `ebfm_cfg_rp_ep_rootport`, `find_mem_bar` and `downstream_loop`.

Figure 12. PIO Design Example Simulation Design Hierarchy

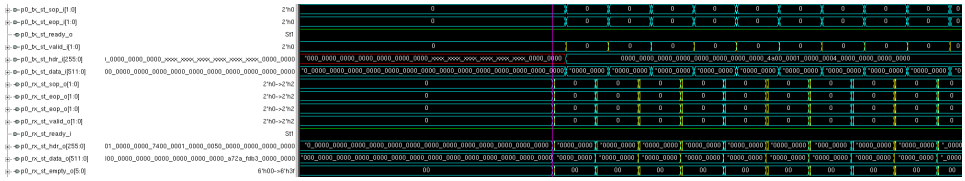


The testbench starts with link training and then accesses the configuration space of the IP for enumeration. A task called `downstream_loop` (defined in the Root Port PCIe BFM `altpcietb_bfm_rp_gen4_x16.sv`) then performs the PCIe link test. This test consists of the following steps:

1. Issue a memory write command to write a single dword of data into the on-chip memory behind the Endpoint.
2. Issue a memory read command to read back data from the on-chip memory.
3. Compare the read data with the write data. If they match, the test counts this as a Pass.
4. Repeat Steps 1, 2 and 3 for 10 iterations.

The first memory write takes place around 219 us. It is followed by a memory read at the Avalon-ST RX interface of the P-tile Hard IP for PCIe. The Completion TLP appears shortly after the memory read request at the Avalon-ST TX interface. The memory write and read transactions and the Completion TLP are shown in the following waveforms.

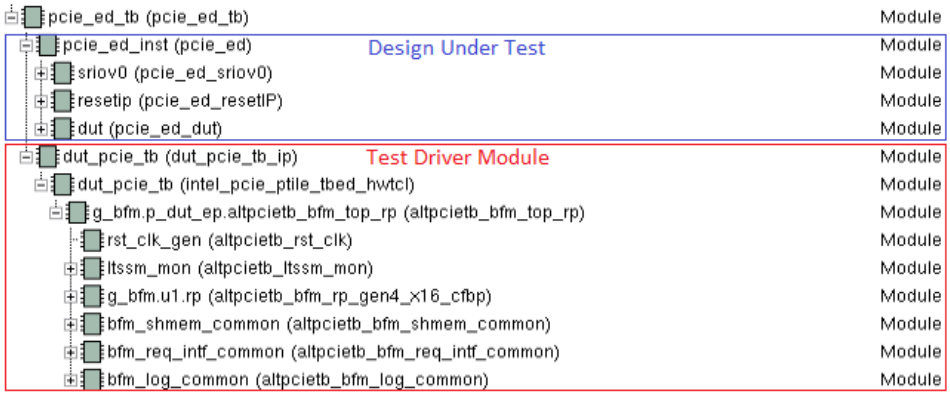
Figure 13. Simulation Waveforms for the PIO Design Example for the P-Tile Avalon-ST IP for PCIe



2.3.1.3. SR-IOV Design Example Testbench

The figure below shows the SR-IOV design example simulation design hierarchy. The tests for the SR-IOV design example are performed by the task called `sriov_test`, which is defined in `altpcietb_bfm_cfbp.sv`.

Figure 14. SR-IOV Design Example Simulation Design Hierarchy



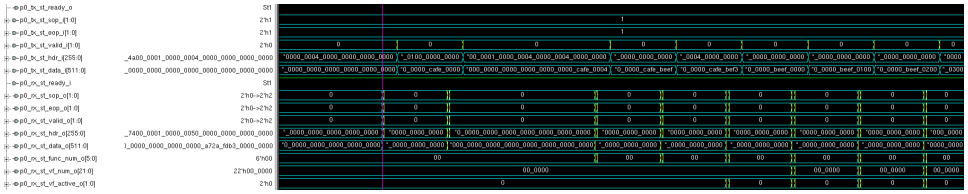
The SR-IOV testbench supports up to two Physical Functions (PFs) and 32 Virtual Functions (VFs) per PF.

The testbench starts with link training and then accesses the configuration space of the IP for enumeration. After that, it performs the following steps:

1. Send a memory write request to a PF followed by a memory read request to read back the same data for comparison. If the read data matches the write data, it is a Pass. This test is performed by the task called `my_test` (defined in `altpcietb_bfm_cfbp.v`). This test is repeated twice for each PF.
2. Send a memory write request to a VF followed by a memory read request to read back the same data for comparison. If the read data matches the write data, it is a Pass. This test is performed by the task called `cfbp_target_test` (defined in `altpcietb_bfm_cfbp.v`). This test is repeated for each VF.

The first memory write takes place around 263 us. It is followed by a memory read at the Avalon-ST RX interface of PF0 of the P-tile Hard IP for PCIe. The Completion TLP appears shortly after the memory read request at the Avalon-ST TX interface. The memory write and read transactions and the Completion TLP associated with both PFs and VFs are shown in the following waveforms.

Figure 15. Simulation Waveforms for the SR-IOV Design Example for the P-Tile Avalon-ST IP for PCIe



2.4. Compiling the Design Example

1. Navigate to `<project_dir>/intel_pcie_ptile_ast_0_example_design/` and open `pcie_ed.qpf`.
2. If you select either of the two following development kits, the VID-related settings are included in the `.qsf` file of the generated design example, and you are not required to add them manually. Note that these settings are board-specific.
   - Intel Stratix 10 DX P-Tile ES1 FPGA development kit
   - Intel Agilex F-Series P-Tile ES0 FPGA development kit
3. On the Processing menu, select **Start Compilation**.

2.5. Installing the Linux Kernel Driver

Before you can test the design example in hardware, you must install the Linux kernel driver. You can use this driver to perform the following tests:

- A PCIe link test that performs 100 writes and reads
- Memory space DWORD(1) reads and writes
- Configuration Space DWORD reads and writes

In addition, you can use the driver to change the value of the following parameters:

- The BAR being used
- The selected device (by specifying the bus, device and function (BDF) numbers for the device)

Complete the following steps to install the kernel driver:

1. Navigate to `./software/kernel/linux` under the example design generation directory.
2. Change the permissions on the `install`, `load`, and `unload` files:

   ```
   $ chmod 777 install load unload
   ```
3. Install the driver:

   ```
   $ sudo ./install
   ```
4. Verify the driver installation:

(1) Throughout this user guide, the terms word, DWORD and QWORD have the same meaning that they have in the PCI Express Base Specification. A word is 16 bits, a DWORD is 32 bits, and a QWORD is 64 bits.

```
$ lsmod | grep intel_fpga_pcie_drv
```

Expected result:

```
intel_fpga_pcie_drv 17792 0
```

5. Verify that Linux recognizes the PCIe design example:

```
$ lspci -d 1172:000 -v | grep intel_fpga_pcie_drv
```

Note: If you have changed the Vendor ID, substitute the new Vendor ID for Intel's Vendor ID in this command.

Expected result:

```
Kernel driver in use: intel_fpga_pcie_drv
```

2.6. Running the Design Example

Here are the test operations you can perform on the P-Tile Avalon-ST PCIe design examples:

Table 2. Test Operations Supported by the P-Tile Avalon-ST PCIe Design Examples

Operations	Required BAR	Supported by P-Tile Avalon-ST PCIe Design Example
0: Link test - 100 writes and reads	0	Yes
1: Write memory space	0	Yes
2: Read memory space	0	Yes
3: Write configuration space	N/A	Yes
4: Read configuration space	N/A	Yes
5: Change BAR	N/A	Yes
6: Change device	N/A	Yes
7: Enable SR-IOV	N/A	Yes (*)
8: Do a link test for every enabled virtual function belonging to the current device	N/A	Yes (*)
9: Perform DMA	N/A	No
10: Quit program	N/A	Yes

Note: (*) These test operations are available only when the SR-IOV design example is selected.

2.6.1. Running the PIO Design Example

1. Navigate to `./software/user/example` under the design example directory.
2. Compile the design example application:

```
$ make
```

3. Run the test:

```
$ sudo ./intel_fpga_pcie_link_test
```

You can run the Intel FPGA IP PCIe link test in manual or automatic mode. Choose from:

- In automatic mode, the application automatically selects the device. The test selects the Intel PCIe device with the lowest BDF by matching the Vendor ID. The test also selects the lowest available BAR.
- In manual mode, the test queries you for the bus, device, and function number and BAR.

For the Intel Stratix 10 DX or Intel Agilex Development Kit, you can determine the BDF by typing the following command:

```
$ lspci -d 1172
```

4. Here are sample transcripts for automatic and manual modes:

   Automatic mode:

```
Intel FPGA PCIe Link Test - Automatic Mode
Version 2.0
0: Automatically select a device
1: Manually select a device
************************************************
>0
Opened a handle to BAR 0 of a device with BDF 0x100
************************************************
0: Link test - 100 writes and reads
1: Write memory space
2: Read memory space
3: Write configuration space
4: Read configuration space
5: Change BAR
6: Change device
7: Enable SR-IOV
8: Do a link test for every enabled virtual function
   belonging to the current device
9: Perform DMA
10: Quit program
************************************************
> 0
Doing 100 writes and 100 reads . .
Number of write errors:     0
Number of read errors:      0
Number of DWORD mismatches: 0
```

Manual mode:

```
************************************************************
Intel FPGA PCIe Link Test
Version 2.0
0: Automatically select a device
1: Manually select a device
************************************************************
> 1
Enter bus number, in hex:
> 4b
Enter device number, in hex:
> 0
Enter function number, in hex:
> 0
BDF is 0x4b00
B:D.F, in hex, is 4b:0.0
Enter BAR number (-1 for none):
> 0
Opened a handle to BAR 0 of a device with BDF 0x4b00
```

Related Information

PCIe Link Inspector Overview
Use the PCIe Link Inspector to monitor the link at the Physical, Data Link and Transaction Layers.

2.6.2. Running the SR-IOV Design Example

Here are the steps to test the SR-IOV design example on hardware:

1. Run the Intel FPGA IP PCIe link test by running the `sudo ./intel_fpga_pcie_link_test` command and then select the option **1: Manually select a device**.
2. Enter the BDF of the physical function for which the virtual functions are allocated.
3. Enter BAR "0" to proceed to the test menu.
4. Enter option 7 to enable SR-IOV for the current device.
5. Enter the number of virtual functions to be enabled for the current device.

```
*******************************************************
Intel FPGA PCIe Link Test
Version 2.0
0: Automatically select a device
1: Manually select a device
*******************************************************
> 1
Enter bus number, in hex:
> 4b
Enter device number, in hex:
> 0
Enter function number, in hex:
> 1
BDF is 0x4b01
B:D.F, in hex, is 4b:0.1
Enter BAR number (-1 for none):
> 0
Opened a handle to BAR 0 of a device with BDF 0x4b01

*******************************************************
 0: Link test - 100 writes and reads
 1: Write memory space
 2: Read memory space
 3: Write configuration space
 4: Read configuration space
 5: Change BAR
 6: Change device
 7: Enable SRIOV
 8: Do a link test for every enabled virtual function
    belonging to the current device
 9: Perform DMA
10: Quit program
*******************************************************
> 7
Enter the number of VFs to enable for the current device:
> 2048
Enabled 2048 VFs.
Type 'lspci -d 1172:' in a new terminal to determine newly enabled devices' BDFs.
```

6. Enter option 8 to perform a link test for every enabled virtual function allocated for the physical function. The link test application will do 100 memory writes with a single dword of data each and then read the data back for checking. The application will print the number of virtual functions that failed the link test at the end of the testing.

```
Number of write errors:       0
Number of read errors:        0
Number of dword mismatches:   0
Testing VF with BDF 0x52fe...
Doing 100 writes and 100 reads..
Number of write errors:       0
Number of read errors:        0
Number of dword mismatches:   0
Testing VF with BDF 0x52ff...
Doing 100 writes and 100 reads..
Number of write errors:       0
Number of read errors:        0
Number of dword mismatches:   0
Testing VF with BDF 0x5300...
Doing 100 writes and 100 reads..
Number of write errors:       0
Number of read errors:        0
Number of dword mismatches:   0
Testing VF with BDF 0x5301...
Doing 100 writes and 100 reads..
Number of write errors:       0
Number of read errors:        0
Number of dword mismatches:   0
Test failed for 0 VFs out of 2048 VFs
```

7. In a new terminal, run the `lspci –d 1172: | grep -c “Altera”` command to verify the enumeration of PFs and VFs. The expected result is the sum of the number of physical functions and number of virtual functions.

```
bash $ lspci -d 1172: | grep -c "Altera"
2050
```

A. Document Revision History for the Intel P-Tile Avalon-ST Hard IP for PCIe Design Example User Guide

A.1. Intel FPGA P-Tile Avalon Streaming IP for PCI Express Design Example User Guide Revision History

Document Version	Intel Quartus Prime Version	IP Version	Changes
2020.10.05	20.3	3.1.0	Removed the *Registers* section since the Avalon Streaming design examples have no control register.
2020.07.10	20.2	3.0.0	Added simulation waveforms, test case descriptions and test result descriptions for the design examples. Added simulation instructions for the ModelSim simulator to the *Simulating the Design Example* section.
2020.05.07	20.1	2.0.0	Updated the document title to *Intel FPGA P-Tile Avalon streaming IP for PCI Express Design Example User Guide* to meet new legal naming guidelines. Updated the VCS interactive mode simulation command.
2019.12.16	19.4	1.1.0	Added SR-IOV design example description.
2019.11.13	19.3	1.0.0	Added Gen4 x8 Endpoint and Gen3 x8 Endpoint to the list of supported configurations.
2019.05.03	19.1.1	1.0.0	Initial release.